Минобрнауки России

Юго-Западный государственный университет

Кафедра программной инженерии

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА ПО ПРОГРАММЕ БАКАЛАВРИАТА

09.03.04 Программная инженерия

(код, наименование ОПОП ВО: направление подготовки, направленность (профиль))

«Разработка программно-информационных систем»

Программно-информационная система для удаленного администрирования компьютеров организации

(название темы)

Дипломный проект

(вид ВКР: дипломная работа или дипломный проект)

Автор ВКР		Я. В. Арнаутов
	(подпись, дата)	(инициалы, фамилия)
Группа ПО-01б		
Руководитель ВКР		Е. А. Петрик
	(подпись, дата)	(инициалы, фамилия)
Нормоконтроль		А. А. Чаплыгин
	(подпись, дата)	(инициалы, фамилия)
ВКР допущена к защите:		
Заведующий кафедрой		А. В. Малышев
	(подпись, дата)	(инициалы, фамилия)

Курск 2024 г.

Минобрнауки России

Юго-Западный государственный университет

Кафедра программной инженерии

УТВЕРЖДАЮ:

Заведующий кафедрой

_____

(подпись, инициалы, фамилия)

«____» _____ 20____ г.

ЗАДАНИЕ НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ ПО ПРОГРАММЕ БАКАЛАВРИАТА

Студента Арнаутов Я. В., шифр 20-06-0420, группа ПО-01б

1. Тема «Программно-информационная система для удаленного администрирования компьютеров организации» утверждена приказом ректора ЮЗГУ от «07» апреля 2023 г. № 1505-с.

2. Срок предоставления работы к защите «13» июня 2023 г.

3. Исходные данные для создания программной системы:

3.1. Перечень решаемых задач:

1) проанализировать IT-инфраструктуру организации;

2) разработать концептуальную модель системы управления IT-инфраструктурой предприятия на основе подхода к управлению и расположению компьютеров в организации;

3) спроектировать программную систему удалённого администрирования IT-инфраструктурой организации;

4) сконструировать и протестировать программную систему удалённого администрирования компьютеров организации.

3.2. Входные данные и требуемые результаты для программы:

1) Входными данными для программной системы являются: данные по IP адресам компьютеров организации, введённые команды, данные о подключении и отключение от компьютера.

2) Выходными данными для программной системы являются: ответ компьютера-сервера на подключение или отключение, ответ на введённую выполненную команду.

4. Содержание работы (по разделам):

4.1. Введение

4.2. Анализ предметной области

4.3. Техническое задание: основание для разработки, назначение разработки, требования к программной системе, требования к оформлению документации.

4.4. Технический проект: общие сведения о программной системе, проект данных программной системы, проектирование архитектуры программной системы, проектирование пользовательского интерфейса программной системы.

4.5. Рабочий проект: спецификация компонентов и классов программной системы, тестирование программной системы, сборка компонентов программной системы.

4.6. Заключение

4.7. Список использованных источников

5. Перечень графического материала:

Лист 1. Сведения о ВКРБ

Лист 2. Цель и задачи разработки

Лист 3. Диаграмма архитектуры локальной сети

Лист 4. Диаграмма прецедентов

Лист 5. UML диаграмма логики действия приложения

Лист 6. Диаграмма классов клиента

Лист 7. Диаграмма классов сервера

Лист 8. Заключение

Руководитель ВКР		Е. А. Петрик
	(подпись, дата)	(инициалы, фамилия)
Задание принял к исполнению		Я. В. Арнаутов
	(подпись, дата)	(инициалы, фамилия)

РЕФЕРАТ

Объем работы равен 80 страницам. Работа содержит 25 иллюстраций, 11 таблиц, 20 библиографических источников и 8 листов графического материала. Количество приложений – 2. Графический материал представлен в приложении А. Фрагменты исходного кода представлены в приложении Б.

Перечень ключевых слов: система, организация, компьютер, IT- администратор, сервер, клиент, соединение, TCP, команда, ответ, сообщение.

Объектом разработки является приложение удаленного администрирования компьютеров организации.

Целью выпускной квалификационной работы является повышения эффективности работы IT-администраторов организации с помощью приложения удаленного администрирования компьютеров.

В процессе создания приложения были выделены основные сущности путем создания информационных блоков, использованы классы и методы модулей, обеспечивающие работу с сущностями предметной области, разработаны разделы, содержащие информацию о компании.

При разработке приложения использовалась система транспортировки данных TCP/IP.

Разработанное приложение было успешно внедрено в компанию.

ABSTRACT

The volume of work is 80 pages. The work contains 25 illustrations, 11 tables, 20 bibliographic sources and 8 sheets of graphic material. The number of applications is 2. The graphic material is presented in annex A. The fragment of the source code is provided in annex B.

The list of keywords: system, organization, computer, IT administrator, server, client, connection, TCP, command, response, message.

The object of development is an application for remote administration of the organization's computers.

The purpose of the final qualification work is to increase the efficiency of the organization's IT administrators using a remote computer administration application.

In the process of creating the application, the main entities were identified by creating information blocks, classes and methods of modules were used to work with the entities of the subject area, sections containing information about the company were developed.

During the development of the application, a TCP/IP data transport system was used.

The developed application was successfully implemented in the company.

СОДЕРЖАНИЕ

ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

TCP – протокол передачи данных.

ПК – персональный компьютер.

РП – рабочий проект.

ТЗ – техническое задание.

ТП – технический проект.

INI – файл конфигурации.

ВВЕДЕНИЕ

В современном мире информационных технологий, где компьютерные сети проникают во все сферы бизнеса и жизни, эффективное администрирование становится неотъемлемой частью успешного функционирования организаций. Однако, с увеличением масштабов бизнеса и географического разнообразия его структур, возникают новые трудности работы IT-специалистов. В этом контексте, удаленное администрирование компьютеров становится не просто дополнительной опцией, а необходимостью.

Удаленное администрирование позволяет IT-администраторам, без физического присутствия, эффективно управлять компьютерами на больших расстояниях. Данная функция становится крайне важной и необходимой в условиях удаленной работы, где сотрудники могут находиться в разных частях города, страны или даже континента.

Также стоит отметить что время становится одним из самых ценных ресурсов, наравне с техническими и человеческими. В ситуации, когда необходимо провести оперативные действия с компьютером, даже краткая задержка может оказать существенное воздействие на процессы бизнеса. Взаимодействие с данными или настройка системы, осуществляемые IT-администратором, порой вынуждают обычного пользователя останавливать свою работу. Тем самым создавая диссонанс в отточенный механизм корпоративных процессов. В свете этих обстоятельств, удаленное администрирование выступает важным инструментом, призванным не просто обеспечивать функциональность, но и минимизировать воздействие на оперативную деятельность. Воплощаясь в максимально незаметных процессах, оно позволяет пользователям беспрепятственно продолжать свои деловые задачи, не отвлекаясь на технические моменты.

Таким образом разработка приложения удаленного администрирования компьютеров становится актуальной задачей для разработчиков программного обеспечения. Такое приложение должно обеспечивать не только простоту и удобство в использовании, но и надежность, безопасность и

высокую производительность. Кроме того, оно должно быть гибким и масштабируемым, способным адаптироваться к изменяющимся потребностям и условиям бизнеса.

Цель настоящей работы – разработка приложение удаленного администрирования компьютеров организации для улучшения эффективности работы IT-администраторов. Для достижения поставленной цели необходимо решить *следующие задачи:*

- провести анализ предметной области;
- разработать концептуальную модель приложения;
- спроектировать приложение;
- реализовать приложение.

Структура и объем работы. Отчет состоит из введения, 4 разделов основной части, заключения, списка использованных источников, 2 приложений. Текст выпускной квалификационной работы равен 80 страницам.

Во введении сформулирована цель работы, поставлены задачи разработки, описана структура работы, приведено краткое содержание каждого из разделов.

В первом разделе на стадии описания технической характеристики предметной области приводится сбор информации о потребностях IT-администраторов и данные о компьютерах в организации.

Во втором разделе на стадии технического задания приводятся требования к разрабатываемому приложению.

В третьем разделе на стадии технического проектирования представлены проектные решения для приложения.

В четвертом разделе приводится список классов и их методов, использованных при разработке приложения, производится тестирование разработанного приложения.

В заключении излагаются основные результаты работы, полученные в ходе разработки.

В приложении А представлен графический материал. В приложении Б представлены фрагменты исходного кода.

1 Анализ предметной области

1.1 Описание предметной области

Удаленное администрирование представляет собой метод управления информационными системами, компьютерными сетями и серверами без необходимости физического присутствия администратора на месте. Этот подход становится все более востребованным в современном мире, где компьютеры компании расположенны на большом удалении друг от друга, а технологии позволяют обеспечить надежное удаленное управление.

Основной целью удаленного администрирования является обеспечение надежности, доступности и безопасности работы информационных систем и сетей. В современных условиях, когда организации все более активно используют облачные решения и развивают глобальные сети, удаленное администрирование становится необходимым компонентом эффективного функционирования бизнеса.

Удаленное администрирование позволяет оперативно реагировать на возникающие проблемы и инциденты, минимизируя время простоя систем и сокращая потери бизнеса. Благодаря возможности удаленного мониторинга и управления, администраторы могут непрерывно следить за состоянием систем, выявлять угрозы безопасности и проводить профилактические мероприятия, что способствует повышению общей надёжности и стабильности работы информационной инфраструктуры.

Кроме того, удаленное администрирование снижает операционные расходы организации за счёт уменьшения необходимости в дорогостоящем обслуживании и обновлении аппаратного обеспечения, а также сокращения затрат на поездки и проживание администраторов на объектах.

С учётом растущей цифровизации бизнеса и повсеместного использования удаленных рабочих мест, удаленное администрирование становится неотъемлемой составляющей успешного функционирования компаний в современном информационном пространстве.

1.2 Основные принципы удаленного администрирования

В основе удаленного администрирования лежат фундаментальные принципы, на которых строится эффективное управление информационными системами и сетями в удаленном режиме. Они определяют основные правила работы и подходы, которые необходимо учитывать при реализации и использовании удаленных административных решений. Соблюдение данных принципов является важным условием для успешного и безопасного функционирования систем удаленного администрирования. Ниже перечислены основные из них:

Безопасность: Одним из важнейших принципов удаленного администрирования является обеспечение безопасности системы. Это включает в себя использование шифрование данных во время передачи, контроль доступа к ресурсам, а также мониторинг и реагирование на потенциальные угрозы и атаки.

Доступность: Основным принципом удаленного администрирования является обеспечение постоянного доступа к информационным ресурсам независимо от времени и местоположения. Это достигается за счёт высокой надёжности сетевых соединений, резервирования каналов связи и резервного копирования данных.

Производительность: Для эффективного удаленного администрирования необходима высокая производительность средств управления и мониторинга. Это включает в себя оптимизацию сетевых протоколов, использование высокопроизводительного оборудования и программного обеспечения, а также оптимизацию алгоритмов и процессов администрирования.

Автоматизация: Одним из ключевых принципов удаленного администрирования является автоматизация рутиных операций и процессов. Это позволяет сократить время на выполнение задач, уменьшить вероятность ошибок и повысить эффективность работы администраторов.

Гибкость: Принцип гибкости предполагает, что системы удаленного администрирования должны быть гибкими и адаптивными к различным

условиям и требованиям. Это позволяет эффективно реагировать на изменения в бизнес-процессах и технологической инфраструктуре.

1.3 Задачи и функции удаленного администратора

Удаленный администратор играет ключевую роль в обеспечении надёжной и работы систем компании. Его задачи и функции включают в себя:

Управление службами: Администратор может использовать консоль для управления службами, такими как запуск, остановка, приостановка или изменение конфигурации служб.

Управление процессами: Администратор, при необходимости, может просматривать запущенные процессы на компьютере, завершать нежелательные процессы или управлять приоритетами процессов.

Управление пользователями и группами: Администратор может осуществлять управление пользователями и группами, включая создание, удаление, изменение учётных записей и назначение прав доступа.

Настройка и диагностика сети: Администратор выполняет различные сетевые задачи, такие как проверка состояния сетевого подключения, настройка параметров сети, выполнение утилит диагностики сети и много другое.

Управление файловой системой: Также администратор осуществляет работу с файлами и папками, такую как создание, копирование, редактирование, замена, перемещение и удаление файлов.

Мониторинг ресурсов: Администратору необходим доступ к различным утилитам и командам для мониторинга использования ресурсов компьютера, таких как процессор, память, дисковое пространство.

Управление политиками безопасности: Администратор может осуществлять настройку различных политик безопасности, включая управление правами доступа, настройку брандмауэра, а также аудит и журналирование событий безопасности.

1.4 Особенности администрирования в корпоративной среде

Администрирование в корпоративной среде имеет свои уникальные особенности и требования, которые отличаются от администрирования в других типах организаций. Рассмотрим основные из них:

Масштабность: В корпоративной среде часто существует большое количество компьютерных систем, серверов и сетевых устройств, что создает необходимость в масштабируемых решениях для управления всей инфраструктурой.

Системы управления конфигурацией: В корпоративной среде широко используются системы управления конфигурацией (Configuration Management Systems), которые позволяют централизованно управлять конфигурацией компьютерных систем.

Политики безопасности: В корпоративной среде обычно существуют строгие политики безопасности, которые требуют соблюдения определенных стандартов и процедур для обеспечения защиты информации и сетевой инфраструктуры.

Автоматизация и оркестрация: Для эффективного управления корпоративной средой часто применяются инструменты автоматизации и оркестрации, которые позволяют автоматизировать рутинные задачи, координировать работу между различными системами и обеспечивать высокую степень автоматизации.

Централизованный мониторинг: Важным аспектом администрирования в корпоративной среде является централизованный мониторинг состояния систем и сетей, который позволяет оперативно обнаруживать и реагировать на проблемы, а также проводить анализ производительности и использования ресурсов.

Управление доступом и аутентификация: В корпоративной среде особое внимание уделяется управлению доступом и аутентификации пользователей, что включает в себя настройку политик доступа, использование механизмов одноразовых паролей и многофакторной аутентификации.

Бизнес-процессы и требования: При администрировании в корпоративной среде необходимо учитывать специфические бизнес-процессы и требования заказчиков, что требует гибкости и адаптивности в реализации технических решений.

Все эти особенности делают администрирование в корпоративной среде сложным и ответственным процессом, требующим высокой квалификации и профессионализма со стороны администраторов. Однако, правильное управление корпоративной информационной инфраструктурой способствует повышению эффективности бизнеса, обеспечивает защиту от угроз и сбоев, а также снижает операционные риски.

2 Техническое задание

2.1 Основание для разработки

Основанием для разработки программно-информационной системы для удаленного администрирования компьютеров организации является задание на выпускную квалификационную работу приказ ректора ЮЗГУ от « » 2024 года № 0000-0 «Об утверждении тем выпускных квалификационных работ и руководителей выпускных квалификационных работ».

2.2 Цель и назначение разработки

Целью является разработка приложения удаленного администрирования компьютеров организации для улучшения эффективности работы IT-администраторов.

Функциональное назначение разрабатываемого приложения заключается в предоставлении IT-администраторам организации удаленного доступа к консоли и файловой системе компьютеров сотрудников компании.

Предполагается, что данной программой будут пользоваться работники IT-отдела организации для облегчения выполнения своих обязанностей и сокращения затрат человека-часов на решения возникших проблем. На рисунке 2.1 представлена диаграмма архитектуры локальной сети организации. На ней изображен пример того, как выглядит компьютерная сеть для которой необходимо данное приложение.

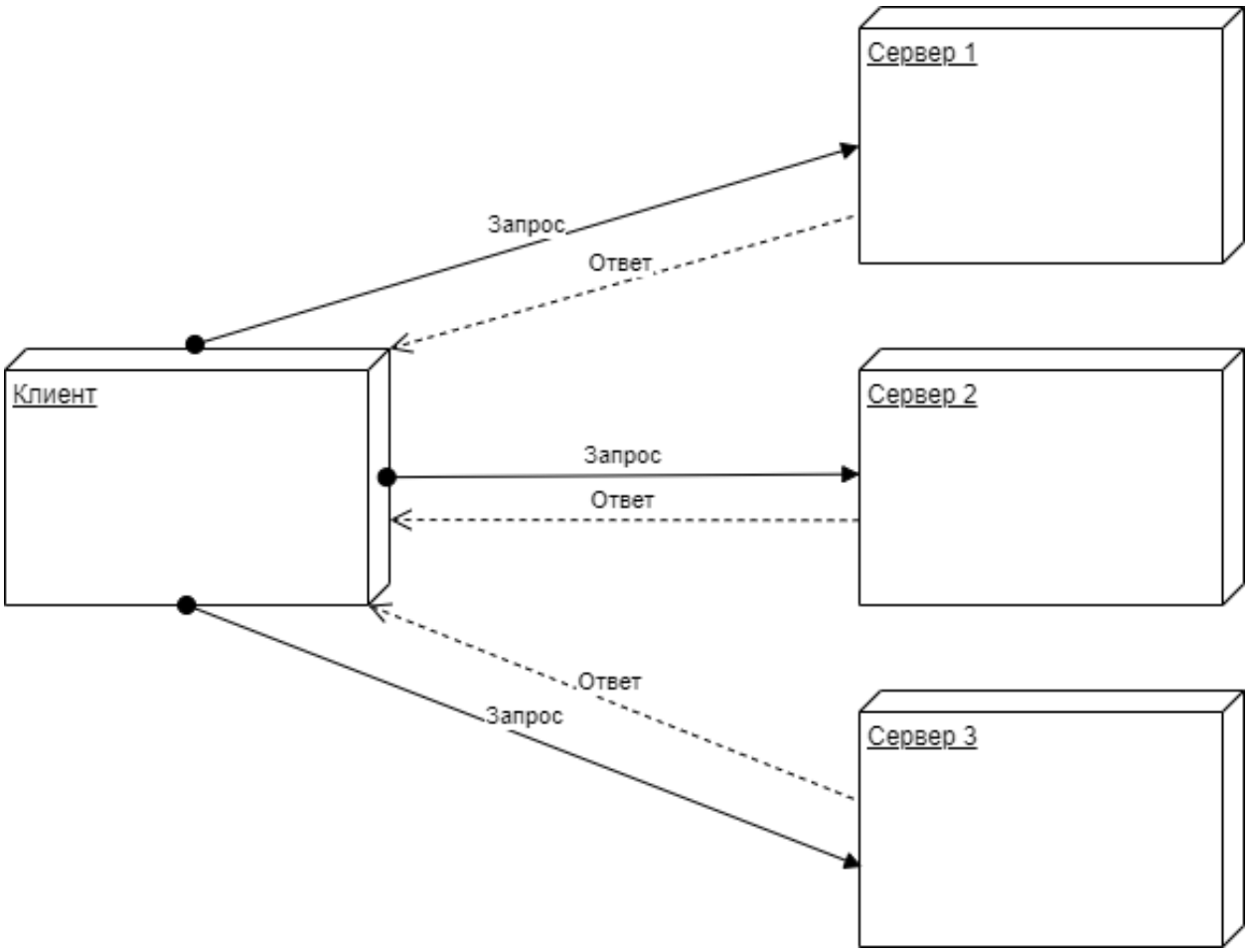


Рисунок 2.1 – Диаграмма архитектуры локальной сети

Задачами разработки данного приложения являются:

- создание клиентской части приложения;
- создание серверной части приложения;
- реализация клиент-серверного соединения;
- реализация обмена данными.

На рисунке 2.2 представлена UML диаграмма логики действия приложения. На ней видно как будет проходить процесс передачи пакетов между клиентом и сервером.

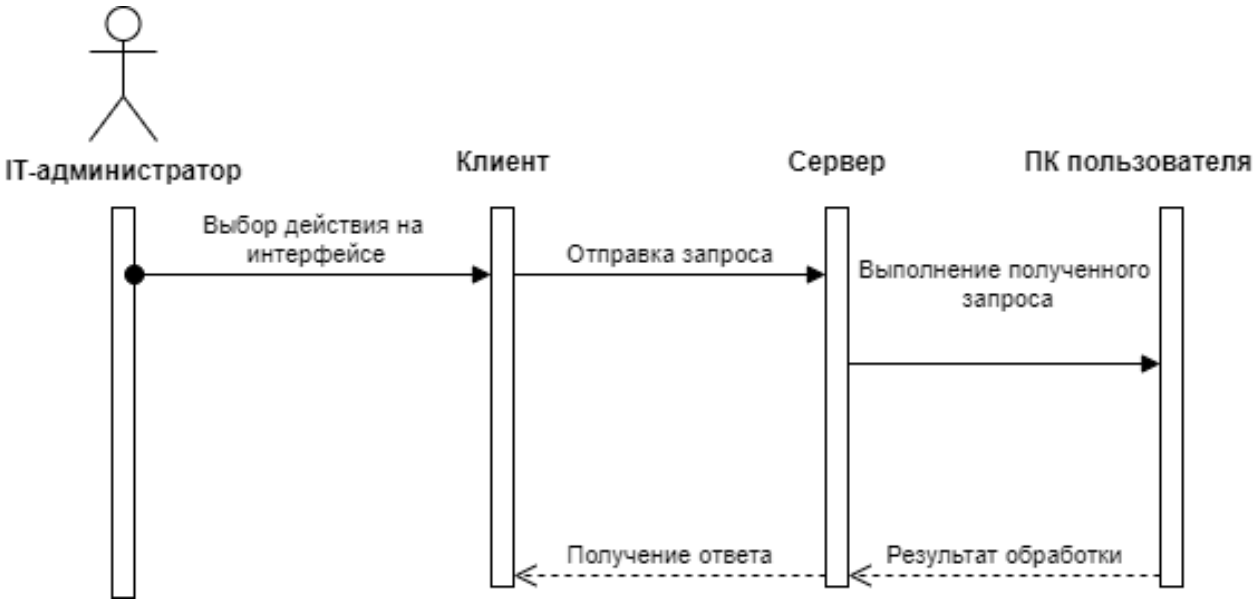


Рисунок 2.2 – UML диаграмма логики действия приложения

2.3 Требования пользователя к программному продукту

2.3.1 Требования к данным программного продукта

Входными данными для приложения являются:

- ip-адрес компьютера пользователя;
- команда для выполнения cmd;
- пути к папкам для отправки/получения файла;
- пути к файлам для отправки/получения.

Выходными данными для приложения являются:

- файловая структура клиента;
- файловая структура сервера;
- результат выполнения команды в cmd.

2.3.2 Функциональные требования к программному продукту

На основании анализа предметной области в разрабатываемой программно-информационной системе удаленного администрирования компьютеров организации должны быть реализованы следующие функции: Для клиентской части программы:

- добавление ip-адрес сервера;

- удаление ip-адрес сервера;
- подключение к серверу;
- открытие файловых систем компьютеров;
- получение и просмотр файловой системы клиента;
- получение и просмотр файловой системы сервера;
- отправка файла на сервер;
- получение файла от сервера;
- открытие консольного управления сервером;
- отправка команды на сервер;
- получение и просмотр ответа от сервера;
- отключение от сервера.

Для серверной части программы:

- ожидание подключения клиента;
- ожидание запроса от клиента;
- выполнение запроса клиента;
- отправка ответа клиенту;

На рисунке 2.3 представлены функциональные требования к системе в виде диаграммы прецедентов.

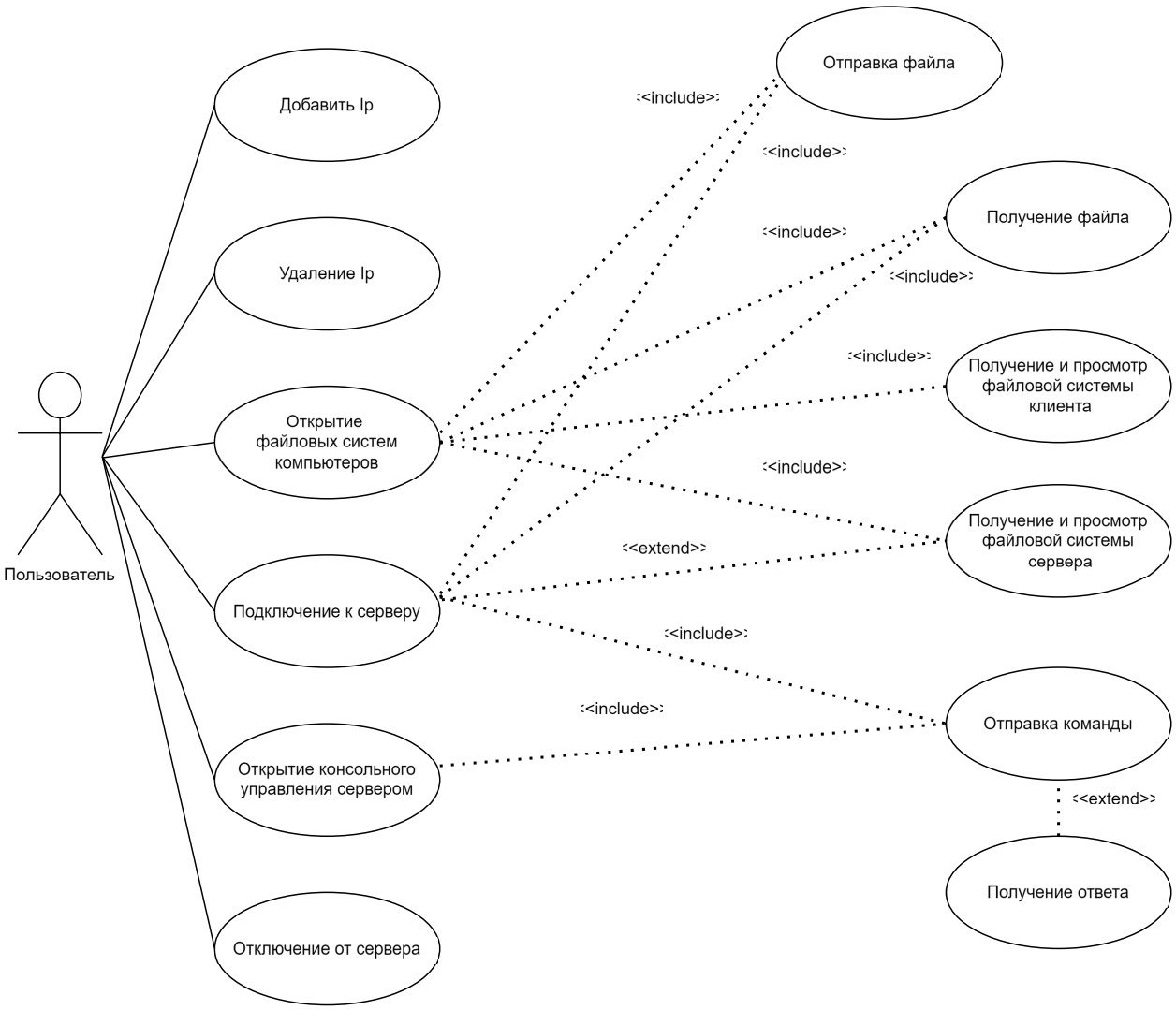


Рисунок 2.3 – Диаграмма прецедентов

2.3.3 Требования пользователя к интерфейсу в программном продукте

Интерфейс программного продукта должен соответствовать следующим требованиям: Для клиентской части программы должны быть реализованы следующие графические элементы:

- Кнопка добавления ip-адрес;
- Кнопка удаления ip-адрес;
- Поле ввода ip-адрес;
- Выпадающий список доступных ip-адрес;
- Кнопка открытие файловых систем компьютеров;
- Поле просмотра файловой системы клиента;

- Поле просмотра файловой системы сервера;
- Кнопка отправки файла на сервер;
- Кнопка получения файла от сервера;
- Кнопка открытие консольного управления сервером;
- Поле ввода команды для отправки на сервер;
- Поле просмотра ответа от сервера;
- Кнопка кнопка подключения к серверу;
- Кнопка отключение от сервера.

Серверная часть программы должна представлять из себя службу.

На рисунках 2.4 - 2.5 представлены функциональные требования к системе в виде диаграммы прецедентов.

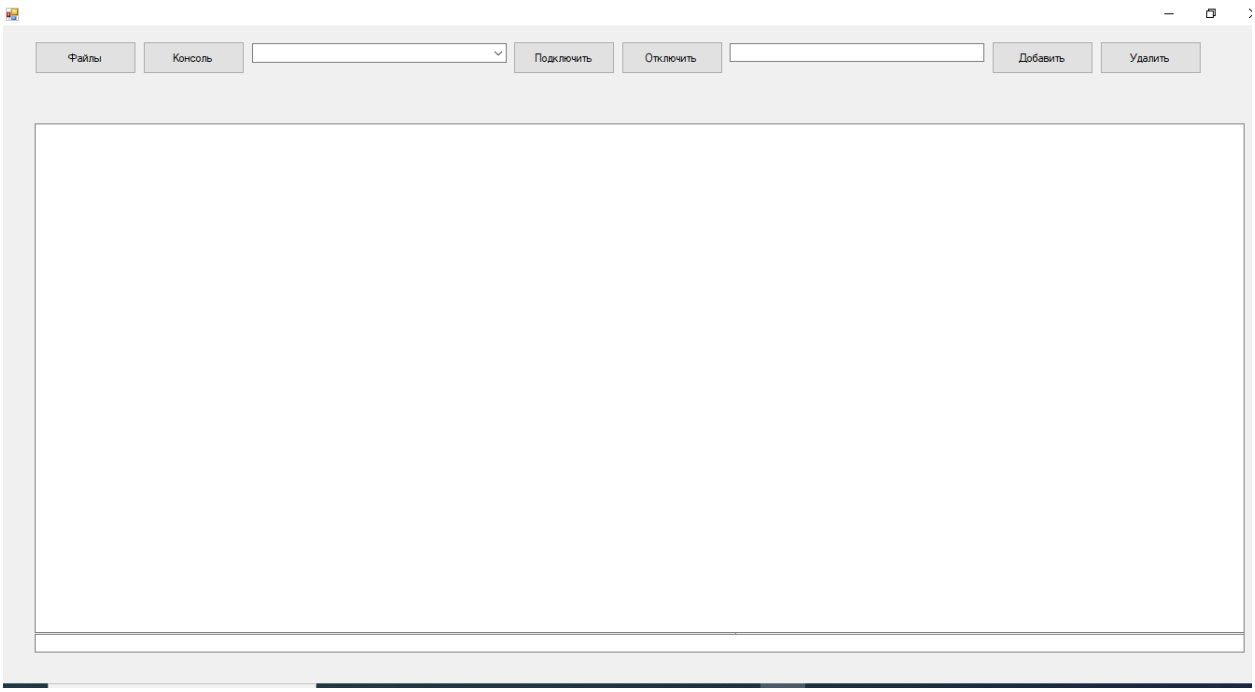


Рисунок 2.4 – Макет окна с консолью

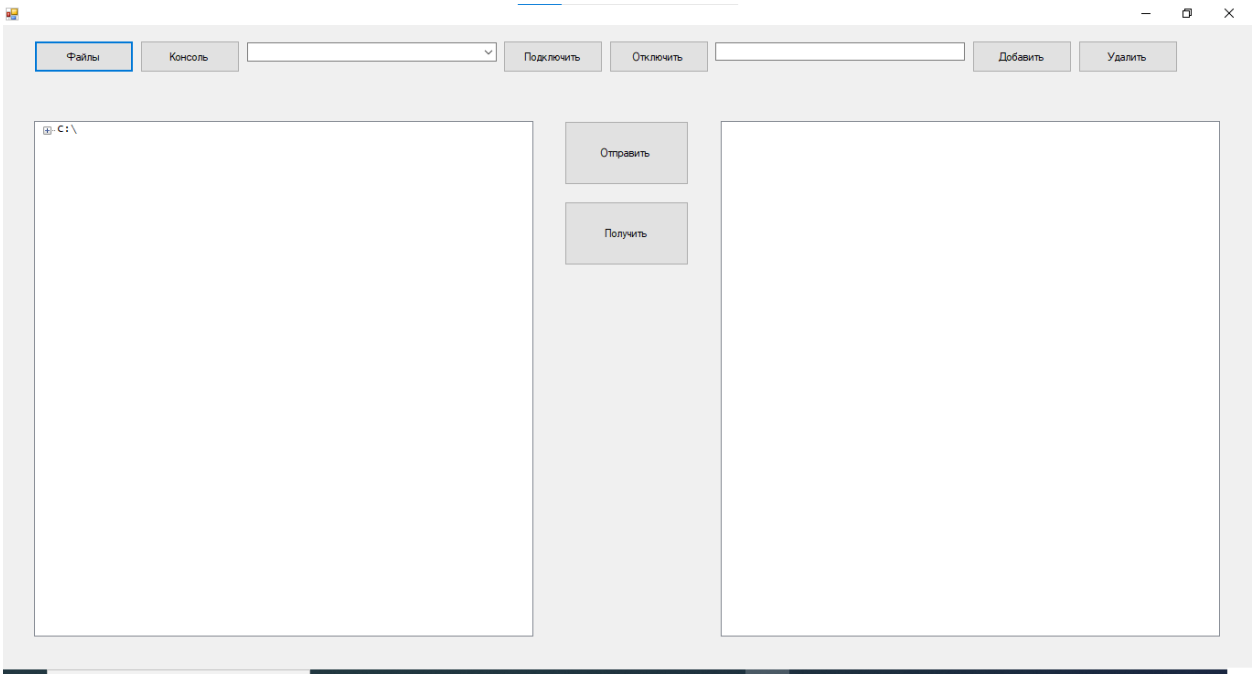


Рисунок 2.5 – Макет окна с файловой системой

2.3.4 Варианты использования программного продукта

2.3.4.1 Добавление ip-адрес

Заинтересованные лица и их требования: IT-администратор, который хочет добавить ip-адрес для дальнейшей возможности подключения к серверу. Предусловие: Пользователь запускает клиентскую часть приложения. Постусловие: Пользователь добавляет ip-адрес. Основной успешный сценарий:

1. Пользователь вводит ip-адрес в строку.
2. Пользователь нажимает кнопку ”Добавить”.
3. ip-адрес появляется в выпадающем списке доступных ip-адрес.

2.3.4.2 Удаление ip-адреса

Заинтересованные лица и их требования: IT-администратор, который хочет удалить ip-адрес. Предусловие: Пользователь запускает клиентскую часть приложения, ip-адрес был ранее добавлен. Постусловие: Пользователь удаляет ip-адрес. Основной успешный сценарий:

1. Пользователь вводит ip-адрес в строку.

2. Пользователь нажимает кнопку ”Удалить”.
3. ip-адрес пропадает из выпадающего списка доступных ip-адрес.

2.3.4.3 Подключение к серверу

Заинтересованные лица и их требования: IT-администратор, который хочет подключиться к серверу. Предусловие: Пользователь запускает клиентскую часть приложения, ip-адрес был ранее добавлен, на сервере запущена серверная часть программы. Постусловие: Пользователь подключается. Основной успешный сценарий:

1. Пользователь выбирает ip-адрес из выпадающего списка.
2. Пользователь нажимает кнопку ”Подключить”.
3. Система выводит сообщение об успешном подключении.

2.3.4.4 Открытие файловых систем компьютеров

Заинтересованные лица и их требования: IT-администратор, который хочет открыть файловую систему компьютеров. Предусловие: Пользователь запускает клиентскую часть приложения, было осуществлено подключение серверу. Постусловие: Система отображает поля для просмотра и взаимодействия с файловой системой компьютеров. Основной успешный сценарий:

1. Пользователь нажимает кнопку ”Файлы”.
2. Система отображает поля для просмотра и взаимодействия с файловой системой компьютеров.

2.3.4.5 Получение и просмотр файловой системы клиента

Заинтересованные лица и их требования: IT-администратор, который хочет посмотреть свою файловую систему. Предусловие: Пользователь запускает клиентскую часть приложения, была нажата кнопка ”Файлы”. Постусловие: Система отображает файловую систему клиента. Основной успешный сценарий:

1. Система отображает файловую систему клиента.
2. Пользователь может с ней взаимодействовать.

2.3.4.6 Получение и просмотр файловой системы сервера

Заинтересованные лица и их требования: IT-администратор, который хочет посмотреть файловую систему сервера. Предусловие: Пользователь запускает клиентскую часть приложения, было осуществлено подключение серверу, была нажата кнопка ”Файлы”. Постусловие: Система отображает файловую систему клиента. Основной успешный сценарий:

1. Система делает запрос на получение файловой системы сервера.
2. Система получает ответ и выводит его в поле.
3. Пользователь может с ней взаимодействовать.

2.3.4.7 Отправка файла

Заинтересованные лица и их требования: IT-администратор, который хочет отправить файл на сервера. Предусловие: Пользователь запускает клиентскую часть приложения, было осуществлено подключение серверу, была нажата кнопка ”Файлы”. Постусловие: Система отображает файловую систему клиента. Основной успешный сценарий:

1. Пользователь выбирает файл для отправки.
2. Пользователь выбирает путь для получения.
3. Пользователь нажимает кнопку ”Отправить”.
4. Система отправляет файл.

2.3.4.8 Получение файла

Заинтересованные лица и их требования: IT-администратор, который хочет получить файл от сервера. Предусловие: Пользователь запускает клиентскую часть приложения, было осуществлено подключение серверу, была нажата кнопка ”Файлы”. Постусловие: Система отображает файловую систему клиента. Основной успешный сценарий:

1. Пользователь выбирает файл для получения.
2. Пользователь выбирает путь для сохранения.
3. Пользователь нажимает кнопку ”Получить”.

4. Система отправляет запрос на получение файла.
5. Система получает файл.

2.3.4.9 Открытие консольного управления сервером

Заинтересованные лица и их требования: IT-администратор, который хочет открыть консольное управление сервером. Предусловие: Пользователь запускает клиентскую часть приложения. Постусловие: Система отображает поля для консольного управления сервером. Основной успешный сценарий:

1. Пользователь нажимает кнопку ”Консоль”.
2. Система отображает поля для консольного управления сервером.

2.3.4.10 Отправка команды

Заинтересованные лица и их требования: IT-администратор, который хочет отправить команду на сервера. Предусловие: Пользователь запускает клиентскую часть приложения, было осуществлено подключение серверу, была нажата кнопка ”Консоль”. Постусловие: Система отправляет команду на сервер. Основной успешный сценарий:

1. Пользователь вводит для отправки.
2. Пользователь нажимает Enter.
3. Система отправляет команду.

2.3.4.11 Получение ответа

Заинтересованные лица и их требования: IT-администратор, который хочет отправить команду на сервера. Предусловие: Пользователь запускает клиентскую часть приложения, было осуществлено подключение серверу, была нажата кнопка ”Консоль была отправлена команда. Постусловие: Система отправляет команду на сервер. Основной успешный сценарий:

1. Серверная часть приложения получает команду.
2. Серверная часть выполняет команду в cmd.
3. Серверная часть отправляет результат выполнения команды.
4. Система получает и выводит ответ от сервера.

2.3.4.12 Отключение от сервера

Заинтересованные лица и их требования: IT-администратор, который хочет отключиться от сервера серверу. Предусловие: Пользователь запускает клиентскую часть приложения, было осуществлено подключение серверу. Постусловие: Пользователь отключается. Основной успешный сценарий:

1. Пользователь нажимает кнопку ”Отключить”.
2. Система выводит сообщение об успешном отключении.

2.4 Нефункциональные требования к программному продукту

2.4.1 Требования к надежности

В приложении не должно возникать критических ошибок, приводящих к экстренному завершению работы.

2.4.2 Требования к программному обеспечению

Для реализации программной системы должен быть использован язык программирования высокого уровня C# для разработки поведения компонентов программы.

2.4.3 Требования к оформлению документации

Разработка программной документации и программного изделия должна производиться согласно ГОСТ 19.102-77 и ГОСТ 34.601-90. Единая система программной документации.

3 Технический проект

3.1 Общие сведения о программной систем

Необходимо спроектировать и разработать программно-информационную систему для удаленного администрирования компьютеров организации.

Разрабатываемая программная система предназначена для использования IT-администраторами организации с целью повышения эффективности и качества их работы. А также оптимизации рабочего время пользователей и уменьшения затрат человеко-часов на решение возникших проблем.

Основной принцип работы системы заключается в установке устойчивого соединения между компьютерами IT-администратора и пользователя с последующей возможностью передачи данных.

Целью разработки приложения удаленного администрирования компьютеров организации является создание эффективного инструмента для повешения уровня обслуживания IT-администраторами компьютера пользователя.

3.2 Обоснование выбора технологии проектирования

Для разработки был выбран проект WinForms так как он предоставляет широкий спектр необходимых условий для успешного создания программного продукта в соответствии с требованиями пользователя:

1. Простота и быстрота разработки: WinForms предоставляет интуитивно понятный инструментарий для создания пользовательского интерфейса. Благодаря этому, разработка приложения может быть выполнена быстро и эффективно, что особенно важно в условиях ограниченного времени на проект.

2. Широкая поддержка в среде разработки: Технология WinForms широко поддерживается в среде разработки .NET, что обеспечивает доступ к богатому набору инструментов и ресурсов для разработчиков. Это включает

в себя возможность использования Visual Studio, богатый выбор компонентов пользовательского интерфейса и библиотеки классов для работы с сетью и другими аспектами системы.

3. Стабильность и надежность: WinForms является проверенной и стабильной технологией, которая широко используется в промышленных приложениях на протяжении многих лет. Это обеспечивает надежность работы приложения и минимизирует риск возникновения ошибок или проблем в процессе эксплуатации.

4. Привычный пользовательский интерфейс: WinForms предлагает привычный пользовательский интерфейс для пользователей Windows, что может уменьшить время обучения и повысить удобство использования приложения.

5. Поддержка Windows-ориентированных функций: WinForms хорошо поддерживает различные функции операционной системы Windows. Это позволяет легко интегрировать приложение с окружением Windows и использовать его функциональные возможности.

3.3 Описание используемых технологий и языков программирования

В рамках разработки приложения удаленного администрирования компьютеров организации было принято решение использовать язык программирования C# в сочетании с протоколом TCP/IP для обеспечения сетевого взаимодействия. Этот выбор обусловлен рядом факторов, которые обеспечивают эффективность, надежность и удобство в разработке и использовании приложения.

3.3.1 Язык программирования C#

C# - это объектно-ориентированный язык программирования, разработанный корпорацией Microsoft. C# является частью платформы .NET и широко используется для создания различных типов приложений, включая веб-

приложения, мобильные приложения, приложения для настольных компьютеров и многое другое.

Особенности языка C#, подходящие для данного проекта:

1. Богатый набор стандартных библиотек: C# предоставляет доступ к обширному набору стандартных библиотек .NET Framework, включая классы и компоненты для работы с сетью, управления потоками, обработки исключений и многого другого. Это позволяет разработчикам эффективно использовать готовые решения для реализации функциональности удаленного администрирования.

2. Удобство работы с GUI: C# обладает удобным синтаксисом и широкими возможностями для создания графического пользовательского интерфейса (GUI). Это особенно важно для разработки приложения с удобным и интуитивно понятным интерфейсом, который позволит администраторам эффективно управлять компьютерами организации.

3. Поддержка многопоточности: C# имеет встроенную поддержку многопоточности, что позволяет разработчикам создавать многопоточные приложения для эффективной работы с параллельными задачами, такими как обработка запросов от нескольких компьютеров одновременно.

4. Высокая производительность: C# компилируется в нативный машинный код с использованием JIT (Just-In-Time) компиляции, что обеспечивает высокую производительность выполнения кода. Это особенно важно для приложения удаленного администрирования, где требуется быстрый отклик на запросы администраторов.

5. Поддержка современных парадигм программирования: C# поддерживает современные парадигмы программирования, такие как асинхронное программирование, что позволяет разработчикам создавать эффективные и гибкие приложения с использованием современных подходов к разработке.

3.3.2 Протокол TCP/IP

Протокол TCP/IP (Transmission Control Protocol/Internet Protocol) является основным протоколом сетевого взаимодействия в интернете и локальных сетях. Он обеспечивает надёжную и универсальную передачу данных между компьютерами и сетевыми устройствами.

Особенности протокола TCP/IP, подходящие для данного проекта:

1. Надежность передачи данных: TCP обеспечивает гарантированную доставку данных, контролируя потери, дублирование и порядок получения пакетов. Это особенно важно для приложения удаленного администрирования, где надежность передачи данных имеет высокий приоритет.

2. Установка соединения: TCP устанавливает соединение между отправителем и получателем перед началом передачи данных, что обеспечивает надежность и порядок доставки. Это позволяет обеспечить безопасность и целостность передаваемой информации.

3. Поддержка потоковой передачи данных: TCP обеспечивает передачу данных в виде потока байтов, что удобно для приложений, требующих непрерывного потока данных. В случае приложения удаленного администрирования, где требуется передача команд и ответов от удаленных компьютеров, потоковая передача данных является эффективным подходом.

4. Универсальность и широкое распространение: Протокол TCP/IP является стандартом сетевого взаимодействия и широко поддерживается на различных платформах и устройствах. Это обеспечивает совместимость и возможность взаимодействия с различными компьютерами и сетевыми устройствами в организации.

Использование протокола TCP/IP для сетевого взаимодействия в приложении удаленного администрирования компьютеров организации обеспечивает надежность, безопасность и эффективность передачи данных между администратором и пользователем.

3.3.3 Дополнительные протоколы

Исходя из информации полученной из предыдущего пункта мы видим, что протокол TCP имеет массу преимуществ, которое подходят нам для создания необходимой нам программной системы. Однако, несмотря на все его преимущества, в нашей конкретной задаче мы сталкиваемся с некоторыми трудностями, для решения которой нам нужны другие специфические протоколы.

Наша система предназначена для передачи файлов, выполнения команд и обработки запросов. В этом контексте TCP обеспечивает надежную и упорядоченную передачу данных, что критически важно для корректной доставки файлов и выполнения команд. Однако есть несколько аспектов, в которых TCP может не полностью удовлетворять наши требования.

Во-первых, при передаче файлов, особенно больших объемов данных, важно обеспечить безопасность и целостность передаваемой информации. Для этого мы будем использовать протокол FilePath.

Во-вторых, выполнение команд требует быстрого и надежного обмена сообщениями. TCP отлично подходит для этого, так как гарантирует доставку данных. Однако для гармоничного взаимодействия с другими протоколами необходим новый, функцию которого выполняет протокол Command.

В-третьих, обработка запросов требует гибкости и возможности работы с различными типами данных и форматами сообщений. Для реализации этой функциональности мы будем использовать протоколы DirectoryDrive, DirectoryFolder, ReceiveFile.

3.3.3.1 Протокол FilePath

Протокол отправки файла на сервер предназначен для передачи файлов между клиентом ик сервером с использованием TCP-соединения. Механизм работы протокола осуществляется с помощью функций SendFileAsync и ReceiveFileAsync(для отправки и получения файла соответственно).

Метод SendFileAsync предназначен для отправки файла между клиентом и сервером. Он асинхронно передает указанный файл.

Параметрами функции выступают:

- filePathClient: путь, который указывает расположение файла для отправки или папки для сохранения на стороне клиента.

- filePathServer: путь, который указывает расположение файла для отправки или папки для сохранения на стороне сервера.

Возвращаемое значение:

- Метод не возвращает значения. Выполняется асинхронно.

Описание процесса:

1. Константа с именем протокола преобразуется в массив байт и отправляется первой. Это означает что сейчас будет получен файл.

2. Вычисляется путь для сохранения файла, преобразуется в массив байт и отправляется вторым.

3. Вычисляется длинна файла, преобразуется в массив байт и отправляется третьей.

4. Последним по указанному пути открывается файл и отправляется блоками массивов байт.

Так как первые 3 пункта представляют строку, буфер для них представляет 4 байта. Для файла буфер(он же блок) составляет 32 Кбайт.

Метод ReceiveFileAsync предназначен для получения файла. Он сохраняет его по указанному пути.

Функция не имеет параметров и выходных данных.

Описание процесса:

1. Считывает массив байт и конвертирует его в строку, по которой нужно сохранить файл.

2. Считывает массив байт и преобразует его в длину файла и запускает цикл для считывания файла.

3. Считывает блоками файл.

3.3.3.2 Протокол Command

Протокол отправки команды на сервер предназначен для передачи сообщений cmd между клиентом и сервером с использованием TCP-соединения.

На стороне клиента механизм отправки команды представлен следующим образом:

1. Константа с именем протокола преобразуется в массив байт и отправляется первой. Это означает что сейчас будет получена команда.
2. Команда преобразуется в массив байт и отправляется на сервер
3. Полученный ответ в режиме реального времени выводится на экран.

На стороне сервера при получении при получении команды вызывается и используются методы ExecuteCommandAsync(входными данными для него служит строка команды, выходных данных нету) и SendCommandAsync(входными данными являются строка и поток данных).

Описание процесса:

1. Отправляет полученную команду в cmd на выполнение.
2. Константа с именем протокола преобразуется в массив байт и отправляется первой. Это означает что сейчас будет получена команда.
3. Полученный ответ преобразуется в массив байт и отправляется обратно клиенту.

Для отправки каждого блока этого протокола используется буфер в 4 байта.

3.3.3.3 Протокол DirectoryDrive

Протокол отправки запроса списка дисков на сервере предназначен для получения клиентом списка дисков сервера с использованием TCP-соединения.

На стороне клиента механизм отправки команды представлен следующим образом:

1. Константа с именем протокола преобразуется в массив байт и отправляется на сервер.

2. Полученный ответ в режиме реального времени выводится на экран.

Когда сервер получает данный запрос запрос запускается следующий процесс:

1. Выполняется функция получения списка дисков ПК.

2. Константа с именем протокола преобразуется в массив байт и отправляется первой. Это означает что сейчас будет получен список дисков.

3. Полученный список преобразуется в массив байт и отправляется обратно клиенту.

Для отправки каждого блока этого протокола используется буфер в 4 байта.

3.3.3.4 Протокол DirectoryFolder

Протокол отправки запроса списка подпапок в конкретной директории на сервере предназначен для получения клиентом списка дисков сервера с использованием TCP-соединения.

На стороне клиента механизм отправки команды представлен следующим образом:

1. Константа с именем протокола преобразуется в массив байт и отправляется на сервер.

2. Строка с указанием пути преобразуется в массив байт и отправляется на сервер.

3. Полученный ответ в режиме реального времени выводится на экран.

Когда сервер получает данный запрос запрос запускается следующий процесс:

1. Выполняется функция получения списка подпапок в полученной директории ПК.

2. Константа с именем протокола преобразуется в массив байт и отправляется первой. Это означает что сейчас будет получен список дисков.

3. Полученный список преобразуется в массив байт и отправляется обратно клиенту.

Для отправки каждого блока этого протокола используется буфер в 4 байта.

3.3.3.5 Протокол ReceiveFile

Протокол отправки запроса списка подпапок в конкретной директории на сервере предназначен для получения клиентом списка дисков сервера с использованием TCP-соединения.

На стороне клиента механизм отправки команды представлен следующим образом:

1. Константа с именем протокола преобразуется в массив байт и отправляется на сервер.

2. Строка с указанием путей для указания файла и сохранения преобразуется в массив байт и отправляется на сервер.

3. Полученный ответ обрабатывается как в протоколе FilePath.

Когда сервер получает данный запрос запрос запускаются методы описанные в протоколе FilePath.

3.4 Проектирование архитектуры программной системы

3.4.1 Архитектура программной системы

3.4.1.1 Описание клиента

Клиентская часть программы представляет собой проект WinForms. Эта часть программной системы непосредственно взаимодействует с пользователем, который представлен IT-администратором. Поэтому при разработке классов необходимо учитывать не только функциональные возможности, но и удобства при работе с интерфейсом.На основании этих требований была разработана диаграмма классов клиентской части, которая представлена на рисунке 3.1.

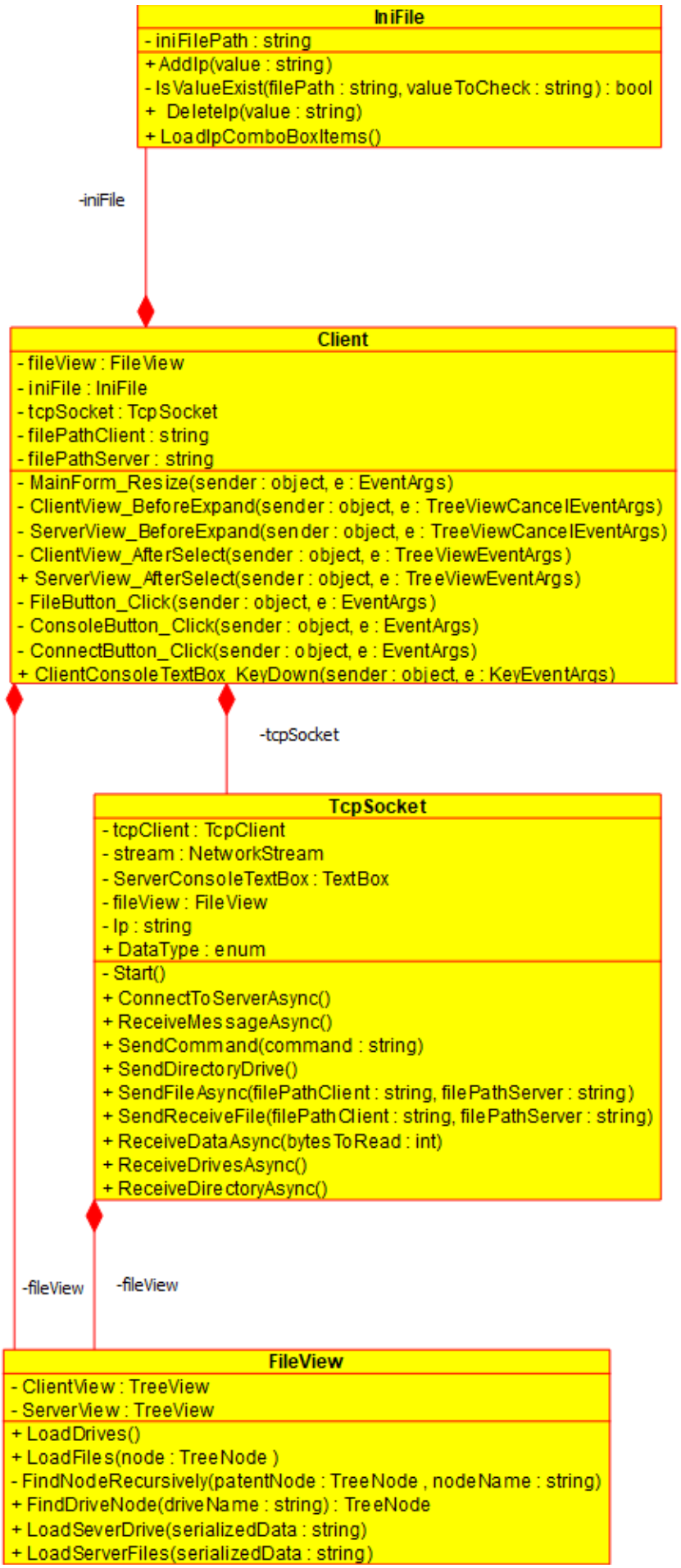


Рисунок 3.1 – Диаграмма классов клиента

В клиентской части программы находятся следующие классы:

- IniFile: для подключения к компьютеру пользователя используется ip-адрес. Так как компьютеров в организации много и вводить адрес каждый раз неудобно, было принято решение о добавлении в проект INI файла для хранения ip-адресов. Также для взаимодействия с этим файлом был создан этот класс. В нем содержатся функции которые позволяют пользователю сохранять и удалять ip-адреса ПК к которым нужно подключиться. Также все записи из файла автоматически выводятся в список для создания подключения.

- Client: этот класс создает окно с интерфейсом для работы администратора. Помимо добавления компонентов в нем также осуществляется их настройка. То есть при изменении размера формы компоненты также будут менять свой размер пропорционально этому. Помимо этот класс напрямую взаимодействует с пользователем. В нем обрабатываются все события, которые вызываются формой: нажатие кнопок, заполнение полей, работа с компонентами treeView.

- TcpSocket: этот класс отвечает за выполнения всех механик связанных с удаленным управлением. В нем осуществляется подключение к серверу, после которого осуществляются следующие действия:

1. Отправка запроса - в данном классе имеется набор функций направленных на реализацию механики отправки запроса по соответствующему протоколу.

2. Отправка команды - в данном классе имеется набор функций направленных на реализацию механики отправки команды по соответствующему протоколу.

3. Отправка файла - в данном классе имеется набор функций направленных на реализацию механики считывания и отправки файла по соответствующему протоколу.

4. Получение файла - в данном классе имеется набор функций, с помощью которого реализуется получение и сохранение файла от сервера по полученному пути.

5. Получение ответа - в данном классе имеется набор функций, с помощью которого осуществляется получение ответа от сервера.

6. Отключение от сервера - данный класс при необходимости обрывает текущее соединение с сервером.

- FileView: этот класс отвечает за обработку в форме файловой системы как ПК клиента, так и ПК сервера. Через него производится раскрытие и закрытие директории, просмотр её содержимого и выбор пути к необходимому файлу или папке.

3.4.1.2 Описание сервера

Сервер представляет из себя службу, которая запускает на всех необходимых IT-администратору ПК. На рисунке 3.2 представлена диаграмма классов сервера.

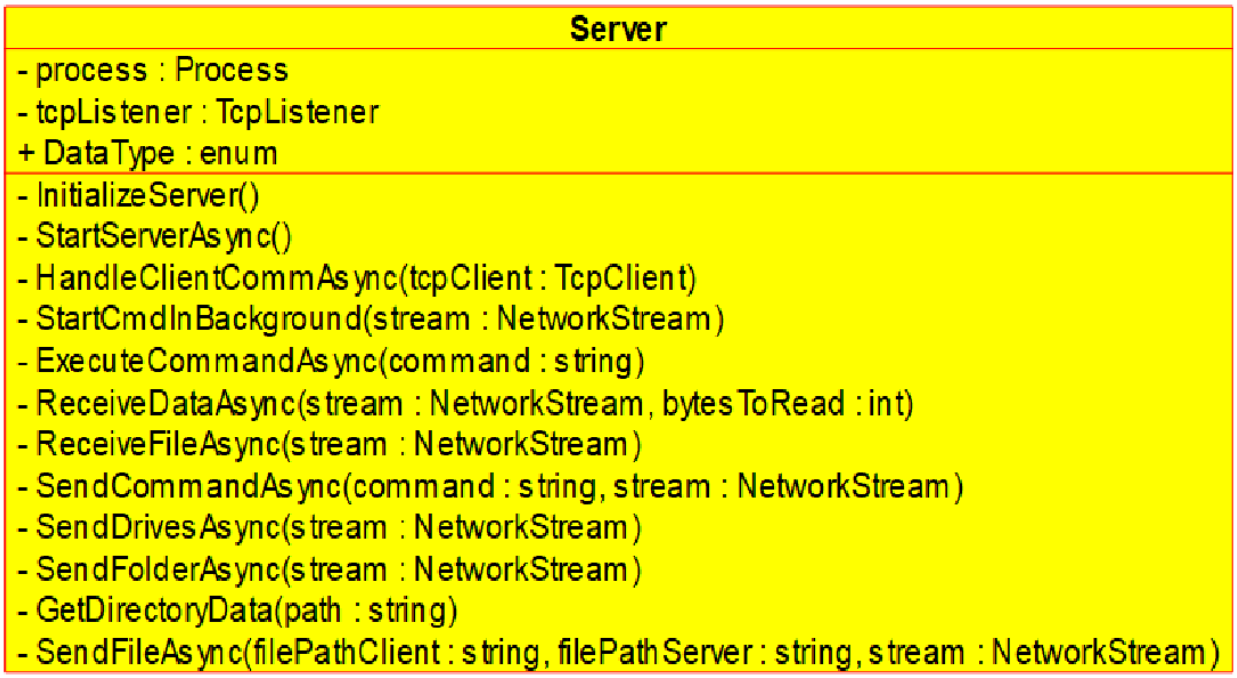


Рисунок 3.2 – Диаграмма классов сервера

В серверной части программы находятся один класс Server, который обрабатывает и выполняет все входяще запросы в соответствии с протоколом в фоновом режиме.

При включении службы запускается функция по ожиданию подключения. После установки соединения осуществляются следующие действия:

1. Получение запроса - в данном классе имеется набор функций направленных на реализацию механики получения запроса по соответствующему протоколу и его выполнение.

2. Выполнение команды - в данном классе имеется набор функций направленных на реализацию механики выполнения команды в cmd и отправки данных обратно клиенту.

3. Отправка файла - в данном классе имеется набор функций направленных на реализацию механики считывания и отправки файла по соответствующему протоколу.

4. Получение файла - в данном классе имеется набор функций, с помощью которого реализуется получение и сохранение файла от клиента по полученному пути.

5. Отправка списка дисков - в данном классе имеется набор функций, с помощью которого осуществляется считывание дисков на ПК и отправка их клиенту.

6. Отправка содержимого папки или диска - в данном классе имеется набор функций, с помощью которого осуществляется считывание содержимого по полученному пути.

3.5 Проектирование пользовательского интерфейса

На основе требований к пользовательскому интерфейсу, представленных в пункте 2.3.3 технического задания, был разработан интерфейс управления программной системой. Для разработки пользовательского интерфейса было выбрано использование компонентов формы в Winform.

Числами на рисунках обозначены номера объектов интерфейса.

На рисунках 3.3 - 3.4 представлен интерфейс программной системы.

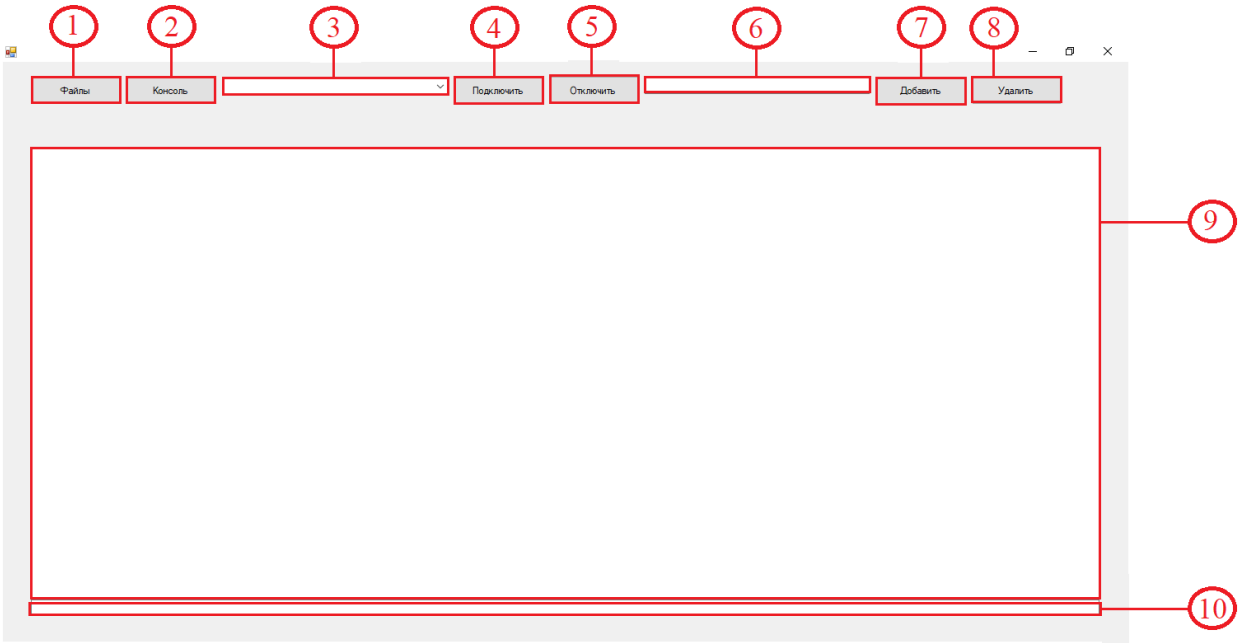


Рисунок 3.3 – Окно с отображением консоли

Элементами данного окна являются:

1. Кнопка открытие файловых систем компьютеров;
2. Кнопка открытие консольного управления сервером;
3. Выпадающий список доступных ip-адрес;
4. Кнопка подключения к серверу;
5. Кнопка отключения от сервера;
6. Поле ввода ip-адрес;
7. Кнопка добавления ip-адрес;
8. Кнопка удаления ip-адрес;
9. Поле просмотра ответа от сервера;
10. Поле ввода команды для отправки на сервер.

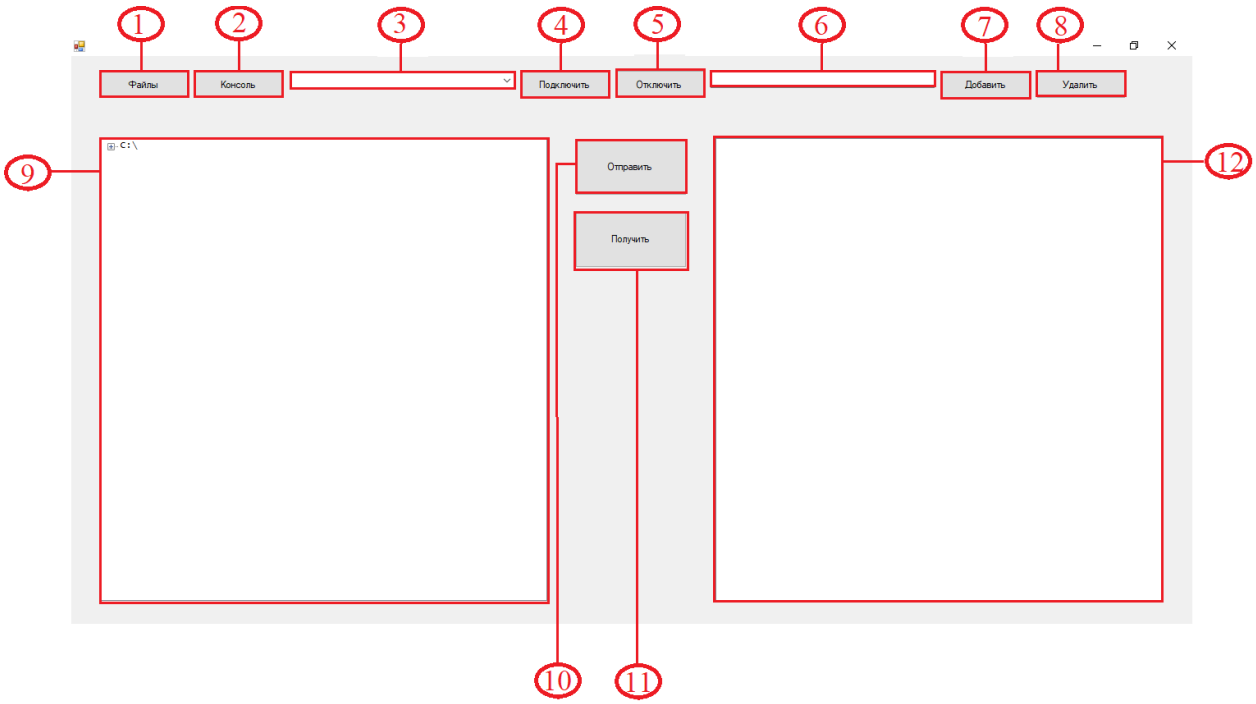


Рисунок 3.4 – Окно с отображением файловой системы

Элементами данного окна являются:

1. Кнопка открытие файловых систем компьютеров;
2. Кнопка открытие консольного управления сервером;
3. Выпадающий список доступных ip-адрес;
4. Кнопка подключения к серверу;
5. Кнопка отключения от сервера;
6. Поле ввода ip-адрес;
7. Кнопка добавления ip-адрес;
8. Кнопка удаления ip-адрес;
9. Поле просмотра файловой системы клиента;
10. Кнопка отправки файла на сервер;
11. Кнопка получения файла от сервера;
12. Поле просмотра файловой системы сервера.

4 Рабочий проект

4.1 Спецификация компонентов и классов программной системы

Для клиентской и серверной части приведено описание основных классов, их полей и методов. Фрагменты исходного кода приведены в приложении Б.

4.1.1 Спецификация классов клиента

Как уже говорилось в пункте 3.4.1.1 технического проекта клиентская часть программы содержит несколько классов. В таблицах 4.1 - 4.4 приведено описание полей данных классов.

Таблица 4.1 – Описание полей класса IniFile

Название поля	Метод доступа	Тип данных	Описание
1	2	3	4
iniFilePath	private	string	Содержит полный путь и имя до INI файла
IpComboBox	private	ComboBox	Содержитэкземпляр элемента ComboBox в который записываются доступные ip-адреса для вывода на форму

Таблица 4.2 – Описание полей класса Client

Название поля	Метод доступа	Тип данных	Описание
1	2	3	4
Ip	private	string	Строка в которую записывается текущий ip-адрес по которому будет осуществлено подключение к серверу
fileView	private	FileView	Экземпляр класса FileView
iniFile	private	IniFile	Экземпляр класса IniFile
tcpSocket	private	TcpSocket	Экземпляр класса TcpSocket
filePathClient	private	string	Строка которая используется для текущего выбранного пути в файловой системе на стороне клиента
filePathServer	private	string	Строка которая используется для текущего выбранного пути в файловой системе на стороне сервера

Таблица 4.3 – Описание полей класса TcpSocket

Название поля	Метод доступа	Тип данных	Описание
1	2	3	4
client	private	TcpClient	Экземпляр класса TcpClient из пространства имен System.Net.Sockets, который построен поверх сокетов и использует сокет для отправки и получения данных, но при этом упрощает написание некоторых вещей
stream	private	NetworkStream	Экземпляр класса NetworkStream из пространства имен System.Net.Sockets, который используется для получения и отправки данных
ServerConsole-TextBox	private	TextBox	Экземпляр класса TextBox, элемент которого используется для вывода сообщений cmd полученных от сервера
fileView	private	FileView	Экземпляр класса FileView

Продолжение таблицы 4.3

1	2	3	4
Ip	private	string	Строка в которую записывается текущий ip-адрес по которому будет осуществлено подключение к серверу

Таблица 4.4 – Описание полей класса FileView

Название поля	Метод доступа	Тип данных	Описание
1	2	3	4
ClientView	private	TreeView	Экземпляр класса TreeView, элемент которого используется для вывода файловой системы клиента
ServerView	private	TreeView	Экземпляр класса TreeView, элемент которого используется для вывода файловой системы сервера

В таблицах 4.5 - 4.8 приведено описание методов данных классов.

Таблица 4.5 – Описание методов класса IniFile

Название метода	Метод доступа	Описание
1	2	3

Продолжение таблицы 4.5

1	2	3
AddId	public	Метод который добавляет введенный пользователем ip-адрес. Результат: если ip-адреса не существует в INI файле, то он добавляется в файл, иначе выводит сообщение об ошибке.
IsValueExist	public	Метод проверки введенного пользователем ip-адреса на соответствие с имеющимися ip-адресами. Результат: возвращает true если совпадение есть, иначе false.
GenerateUniqueKey	private	Метод генерации уникального ключа для сохранения ip-адреса в INI файл. Результат: генерируется случайный ключ для сохранения ip-адреса
DeleteIp	public	Метод удаления введенного пользователем ip-адреса для удаления из INI файла.
LoadIpComboBoxItems	public	Метод считывания всех имеющихся в INI файле ip-адресов, а также вывода их в ComboBox. Результат: в ComboBox отображаются все имеющиеся ip-адреса.

Таблица 4.6 – Описание методов класса Client

Название метода	Метод доступа	Описание
1	2	3

Продолжение таблицы 4.6

1	2	3
MainForm_Resize	private	Метод обработки события изменения элемента управления. Используется для настройки элементов на форме интерфейса. Результат: при изменении формы также пропорционально ей меняются и все элементы.
ClientView _BeforeExpand	private	Метод обработки события разворачивания узла на элементе TreeView. Используется для вывода содержимого выбранной папки в файловой системе клиента. Результат: выводит список всех файлов и папок в выбранной ветке.
ServerView _BeforeExpand	private	Метод обработки события разворачивания узла на элементе TreeView. Используется для вывода содержимого выбранной папки в файловой системе сервера. Результат: выводит список всех файлов и папок в выбранной ветке.
ClientView _AfterSelect	private	Метод обработки события изменения узла на элементе TreeView. Используется для отображения пути выбранного пользователем на стороне клиента. Результат: выводит выбранный пользователем путь.

Продолжение таблицы 4.6

1	2	3
ServerView _AfterSelect	private	Метод обработки события изменения узла на элементе TreeView. Используется для отображения пути выбранного пользователем на стороне сервера. Результат: выводит выбранный пользователем путь.
FileButton_Click	private	Метод обработки нажатия кнопки FileButton. Используется для перехода в режим работы с файловыми системами. Результат: скрываются все лишние элементы и открываются необходимые для работы.
ConsoleButton _Click	private	Метод обработки нажатия кнопки ConsoleButton. Используется для перехода в режим работы с консолью. Результат: скрываются все лишние элементы и открываются необходимые для работы.
ConnectButton _Click	private	Метод обработки нажатия кнопки ConnectButton. Используется для создания подключения к серверу по выбранному ip-адресу. Результат: считывает ip-адрес и соответствующего TextBox и запускает функцию установки соединения.

Продолжение таблицы 4.6

1	2	3
ClientConsoleText-Box_KeyDown	private	Метод обработки нажатия кнопки Enter. Используется для отправки введенной команды на сервер. Результат: считывает команду и соответствующего TextBox и запускает функцию отправки команды.
DisconnectButton _Click	private	Метод обработки нажатия кнопки DisconnectButton. Используется для разрыва текущего соединения с сервером. Результат: соединение с сервером разрывается.
AddButton_Click	private	Метод обработки нажатия кнопки AddButton. Используется для сохранения ip-адреса. Результат: Считывает ip-адрес и соответствующего TextBox и запускает функцию сохранения его в INI файла.
SendButton_Click	private	Метод обработки нажатия кнопки SendButton. Используется для отправки файла на сервер. Результат: считывает файл по выбранному пути и путь по которому сохранить его на сервере и запускает функцию отправки файла на сервер.

Продолжение таблицы 4.6

1	2	3
ReceiveButton_Click	private	Метод обработки нажатия кнопки ReceiveButton. Используется для получения файла от сервера. Результат: считывает путь к файлу, путь сохранения файла и запускает функцию получения файла от сервер.
DeleteButton_Click	private	Метод обработки нажатия кнопки DeleteButton. Используется для удаления ip-адреса. Результат: Считывает ip-адрес и соответствующего TextBox и запускает функцию удаление его из INI файла.

Таблица 4.7 – Описание методов класса TcpSocket

Название метода	Метод доступа	Описание
1	2	3
Start	private	Метод асинхронного запуска поиска сервера. Результат: запуск функции ConnectToServerAsync
ConnectToServer-Async	private	Метод асинхронного подключения к указанному серверу на порте 8888. Результат: если предварительно сервер запущен, то осуществляется подключение, иначе выводится сообщение о соответствующей ошибке.

Продолжение таблицы 4.7

1	2	3
ReceiveMessage-Async	private	Метод получения и обработки полученных данных от сервера. Результат: в зависимости от типа сообщения запускает необходимые функции обработки входных данных.
SendCommand	public	Метод отправки команды на сервер. Результат: команда конвертируется в массив байт и по соответствующему протоколу отправляется на сервер.
SendDirectoryDrive	public	Метод отправки запроса на получения списка дисков на сервер. Результат: запрос конвертируется в массив байт и по соответствующему протоколу отправляется на сервер.
SendDirectoryFolder	public	Метод отправки запроса содержимого выбранной папки на сервер. Результат: запрос конвертируется в массив байт и по соответствующему протоколу отправляется на сервер.
SendFileAsync	public	Метод отправки файла на сервер. Результат: запрос конвертируется в массив байт и по соответствующему протоколу отправляется на сервер.
SendReceiveFile	public	Метод запроса на получение файла на сервер. Результат: запрос конвертируется в массив байт и по соответствующему протоколу отправляется на сервер.

Продолжение таблицы 4.7

1	2	3
ReceiveDataAsync	private	Метод вычитки определенного количества байт из потока. Используется для считывания полученных данных в соответствии с протоколами. Результат: из потока вычитывается указанное количество байт
ReceiveDrivesAsync	private	Метод получения списка запрошенных дисков ПК сервера. Результат: полученный список дисков сервера конвертируется в строку для вывода.
ReceiveDirectory-Async	private	Метод получения списка содержимого запрошенной директории сервера. Результат: полученный список содержимого директории сервера конвертируется в строку для вывода.
ReceiveFileAsync	private	Метод получения файла от сервера. Результат: полученный файл сохраняется по указанному пути.

Таблица 4.8 – Описание методов класса FileView

Название метода	Метод доступа	Описание
1	2	3
LoadDrives	private	Метод загрузки списка дисков ПК клиента. Результат: в TreeView выводятся все диски на ПК клиента

Продолжение таблицы 4.8

1	2	
LoadFiles	public	Метод вывода списка содержимого выбранной папки на ПК клиента. Результат: в TreeView выводятся содержимое выбранной папки на ПК клиента
FindNodeRecursively	private	Метод поиска ранее открытых узлов TreeView. Результат: возвращает полный путь узла или null
FindDriveNode	public	Метод поиска ранее открытых дисков и подпапок. Результат: возвращает имя диска или подпапки или null
LoadSeverDrive	public	Метод вывода списка дисков ПК сервера. Результат: в TreeView выводятся все диски на ПК сервера
LoadServerFiles	public	Метод вывода списка содержимого выбранной папки ПК сервера. Результат: в TreeView выводятся содержимое выбранной папки ПК сервера

4.1.2 Спецификация классов сервера

Как уже говорилось в пункте 3.4.1.2 технического проекта серверная часть программы содержит один класс. В таблицах 4.9 - 4.10 приведены описание полей и методов данного класса.

Таблица 4.9 – Описание полей класса Server

Название поля	Метод доступа	Тип данных	Описание
1	2	3	4
tcpListener	private	TcpListener	Экземпляр класса TcpListener из пространства имен System.Net.Sockets, который построен поверх сокетов и позволяет упростить написание некоторых вещей, по сравнению с использованием чистых сокетов.
process	private	Process	Экземпляр класса Process из пространства имен System.Diagnostics. Этот класс позволяет управлять уже запущенными процессами, а также запускать новые.

Таблица 4.10 – Описание методов класса Server

Название метода	Метод доступа	Описание
1	2	3

Продолжение таблицы 4.10

1	2	3
InitializeServer	private	Метод асинхронного запуска сервера. Результат: вызов метода StartServerAsync.
StartServerAsync	private	Метод запуска сервера на порте 8888 и ожидания подключения клиента. Результат: ожидает подключение клиента. Если оно есть вызывается HandleClientCommAsync.
HandleClient-CommAsync	private	Метод ожидания и считывания полученных данных от клиента. Результат: вызывает метод StartCmdInBackground и в зависимости от полученного сообщения функцию обработки.
StartCmdIn-Background	private	Метод запуска cmd на ПК сервера. Результат: cmd запускается в фоновом режиме от имени администратора.
ExecuteCommand-Async	private	Метод выполнения полученной команды. Результат: отправляет команду в cmd на выполнение.
ReceiveDataAsync	private	Метод вычитки определенного количества байт из потока. Используется для считывания полученных данных в соответствии с протоколами. Результат: из потока вычитывается указанное количество байт

Продолжение таблицы 4.10

1	2	3
ReceiveFileAsync	private	Метод получения файла от клиента. Результат: полученный файл сохраняется по указанному пути.
SendCommandAsync	private	Метод отправки результата выполнения команды клиенту. Результат: команда в режиме реального времени транслируется клиенту.
SendDrivesAsync	private	Метод отправки списка дисков клиенту. Результат: команда конвертируется в массив байт и по соответствующему протоколу отправляется клиенту.
SendFolderAsync	private	Метод отправки списка содержимого папки клиенту. Результат: список конвертируется в массив байт и по соответствующему протоколу отправляется клиенту.
GetDirectoryData	private	Метод считывания списка содержимого папки. Результат: возвращает список содержимого для отправки.
SendFileAsync	private	Метод отправки файла клиенту. Результат: файл конвертируется в массив байт и по соответствующему протоколу отправляется клиенту.

4.1.3 Спецификация enum которые используются в программной системе

Для того, чтобы в программе посылаемые сообщения не путались между собой были придуманы так называемые флаги. Они представляют из себя имена по которым клиент и сервер способны распознать какое именно сообщение приходит по потоку на данный момент. Реализована с помощью типа enum. В таблице 4.11 приведены названия и описания этих флагов.

Таблица 4.11 – Описание флагов в enum

Название	Описание
1	2
FilePath	Флаг который означает, что текущее сообщения передает файл и путь по которому его необходимо сохранить
Command	Флаг который означает, что текущее сообщение передает команду. её следует выполнить в cmd и отправить ответ клиенту
DirectoryDrive	Флаг который означает, что текущее сообщение является запросом на отправку списка дисков сервера клиенту
DirectoryFolder	Флаг который означает, что текущее сообщение является запросом на отправку содержимого папки клиенту по указанному пути
ReceiveFile	Флаг который означает, что текущее сообщение является запросом на отправку файла клиенту

4.2 Системное тестирование разработанного приложения

На рисунке 4.1 представлен интерфейс программы на стороне клиента. При запуске воспроизводится вариант использования консоли сервера.

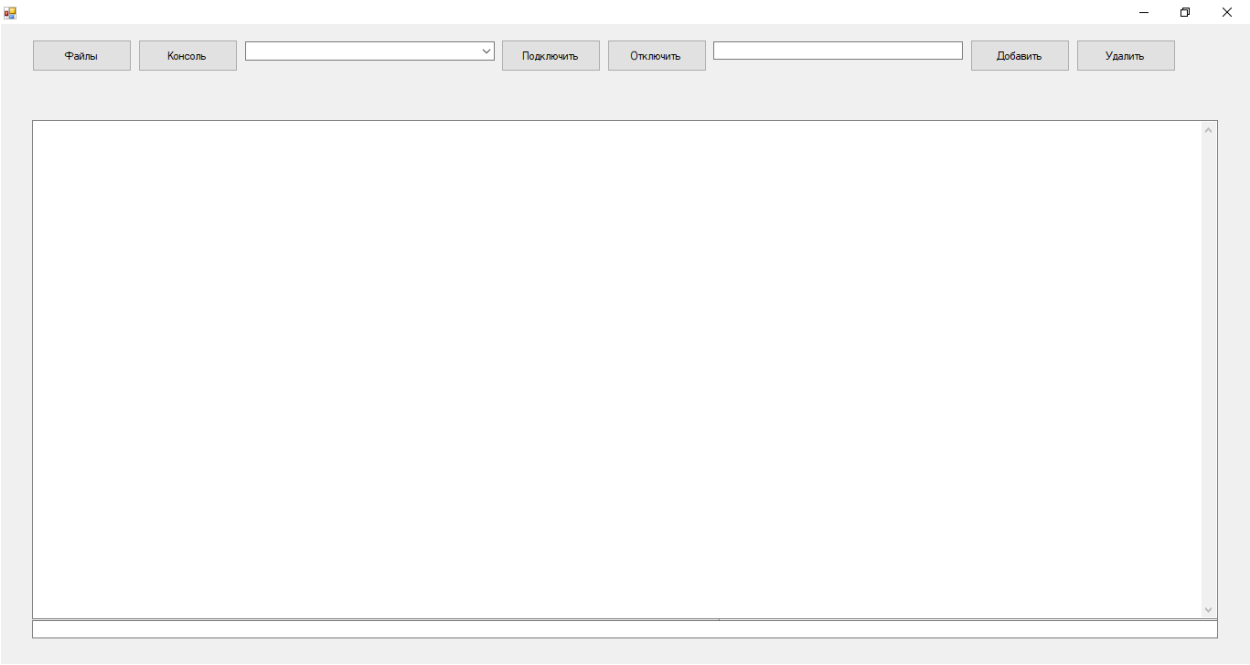


Рисунок 4.1 – Окно при запуске программы

На рисунке 4.2 представлено динамичное изменение интерфейса при деформации окна программы.

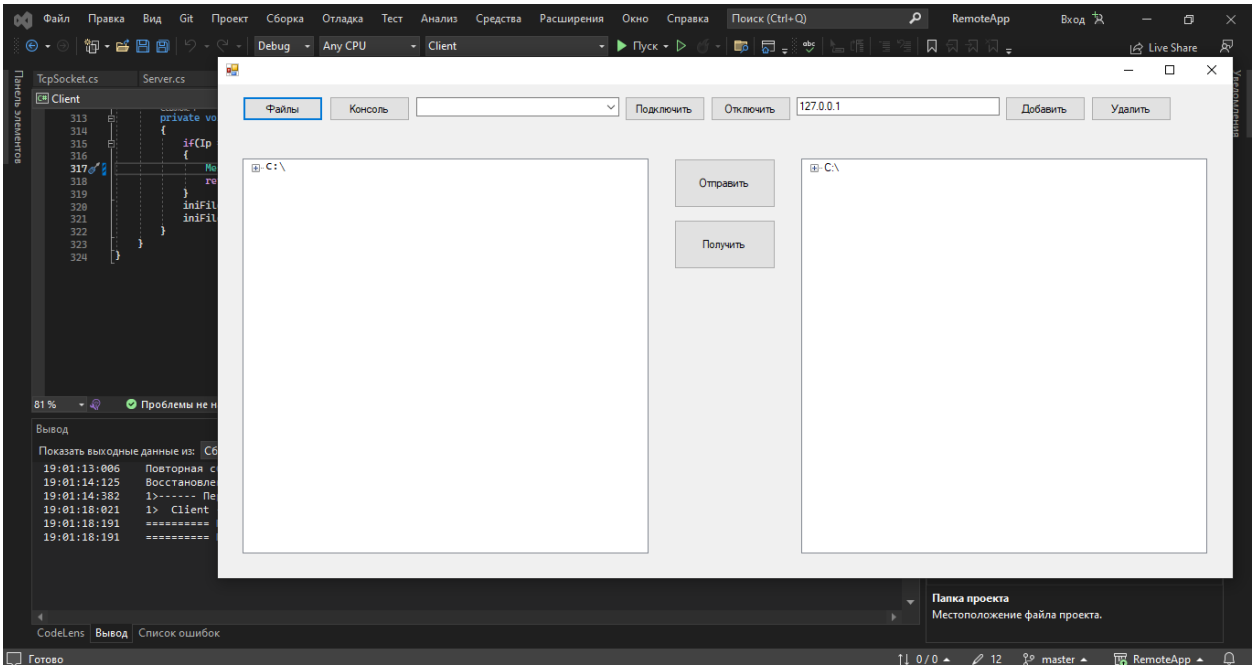


Рисунок 4.2 – Изменение интерфейса при деформации окна программы

На рисунке 4.3 представлено добавление нового ip-адреса в INI файл программы.

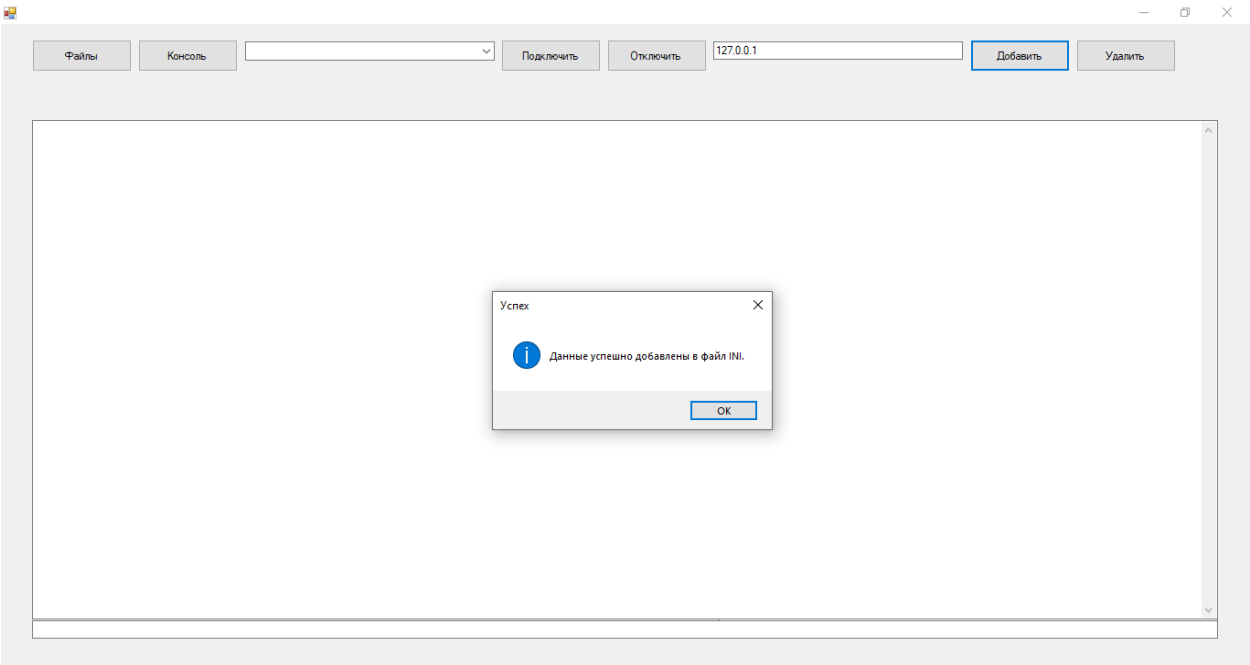


Рисунок 4.3 – Добавление нового ip-адреса

На рисунке 4.4 представлен запуск серверной части программной системы в виде службы.

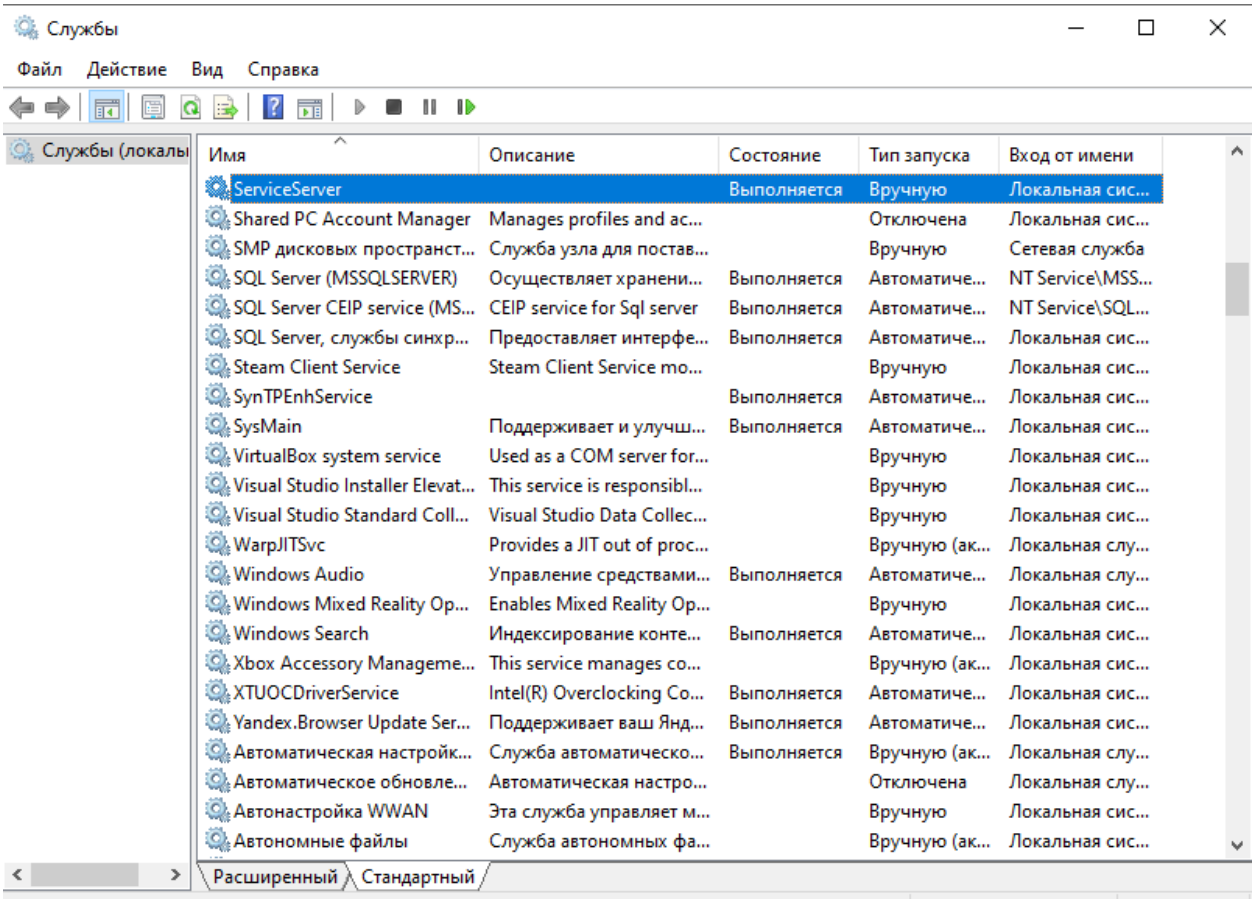


Рисунок 4.4 – Запуск серверной части системы в виде службы

На рисунке 4.5 представлен процесс успешного подключения к серверу по ранее добавленному ip-адресу.

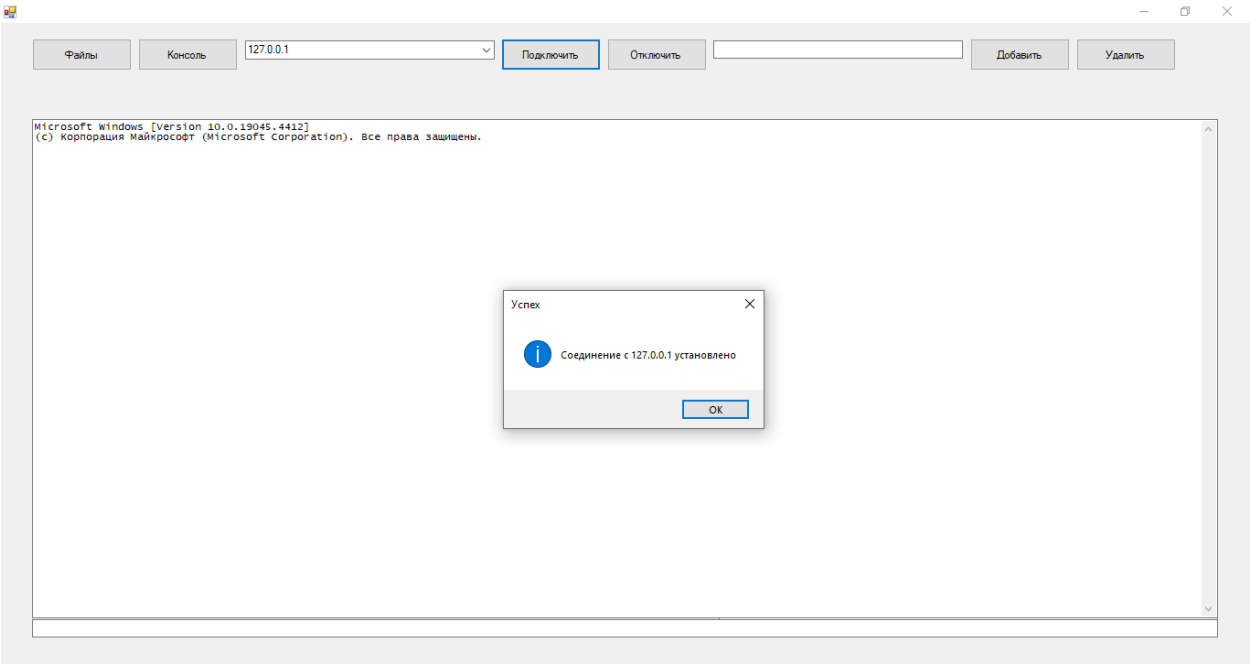


Рисунок 4.5 – Подключение к серверу

На рисунке 4.6 представлен процесс отправки команды в cmd сервера и получение результата её выполнения.

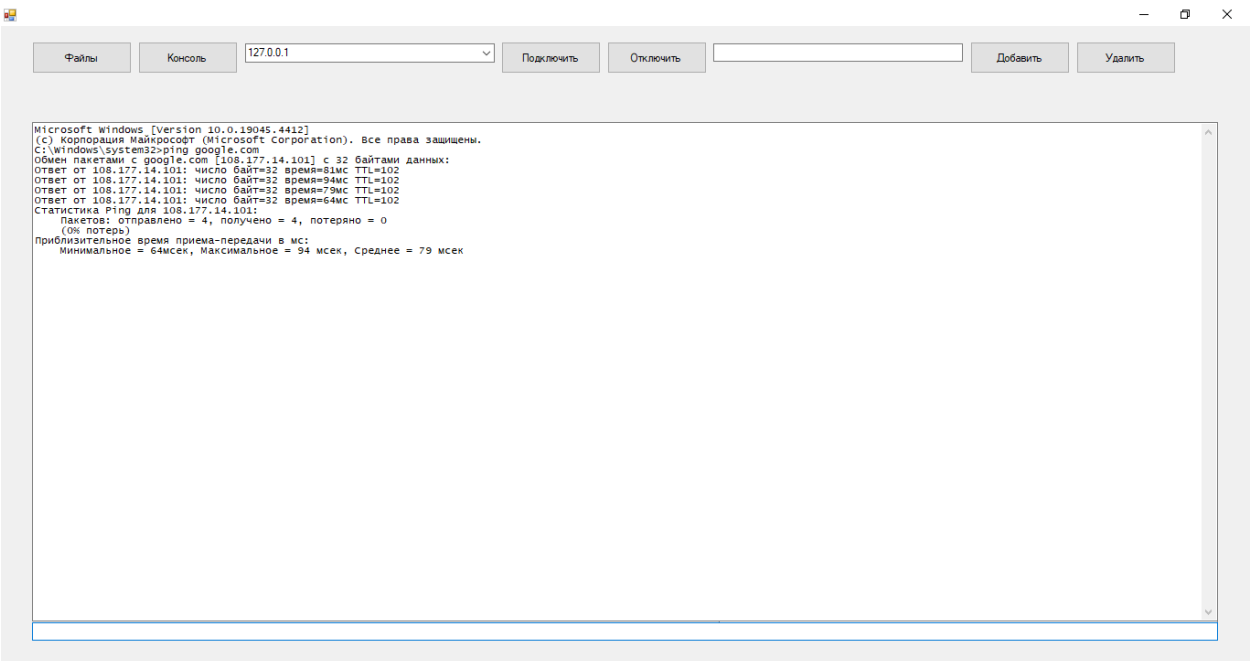


Рисунок 4.6 – Отправка команды и получения результата её выполнения

На рисунке 4.7 представлено переключения в режим работы файловой системы ПК.

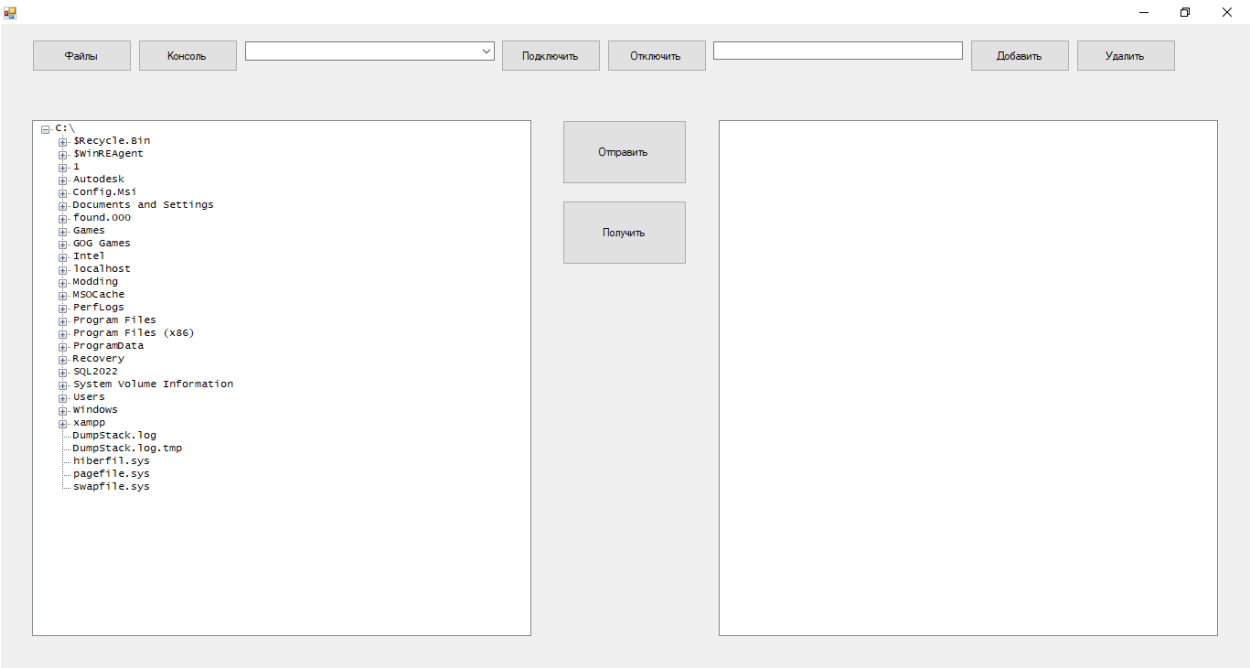


Рисунок 4.7 – Переключение в режим работы с файловой системой

На рисунке 4.8 представлено переключения в режим работы файловой системы при существующем активном соединении.

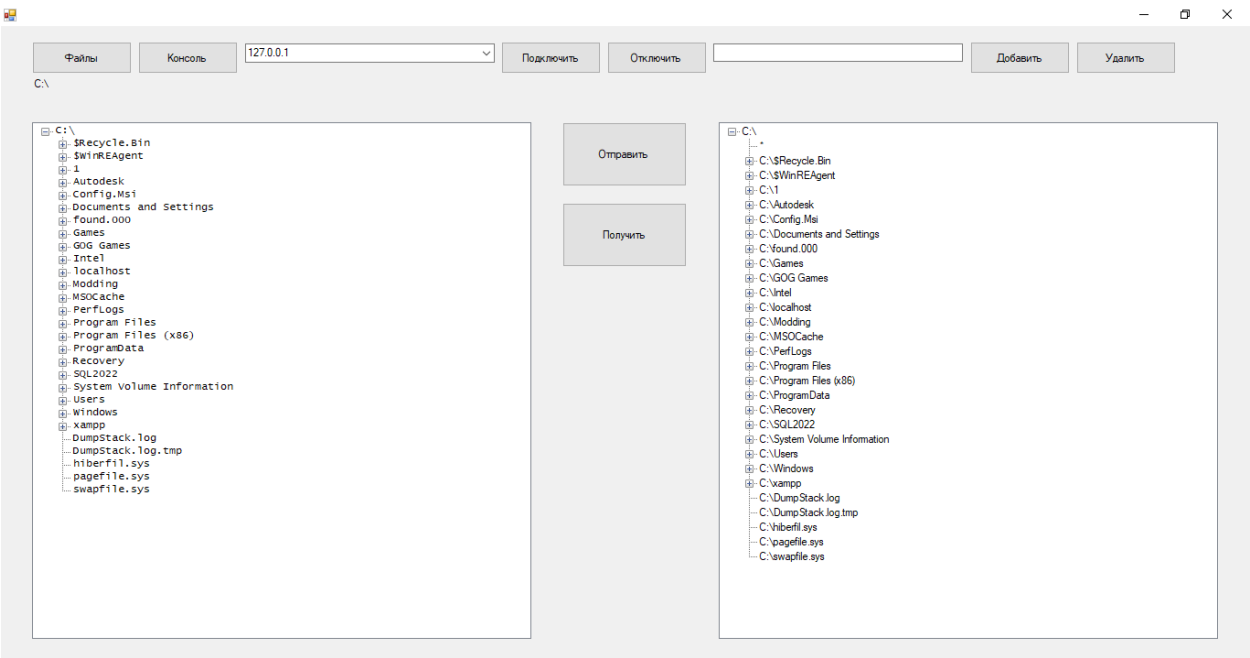


Рисунок 4.8 – Переключение в режим работы с файловой системой при существующем соединении

На рисунке 4.9 представлен процесс просмотра содержимого дисков и папок обоих ПК. А также процесс выбора пути для получения или отправления файлов.

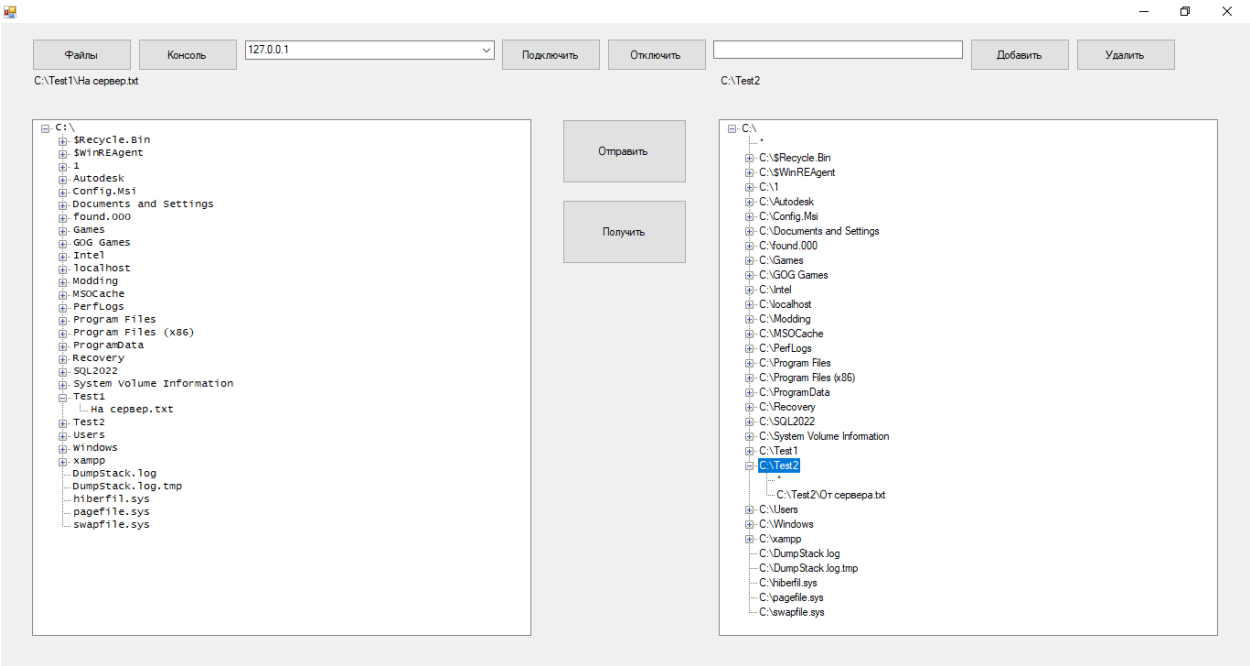


Рисунок 4.9 – Процесс просмотра содержимого дисков и папок

На рисунке 4.10 представлен процесс отправки файла на сервер.

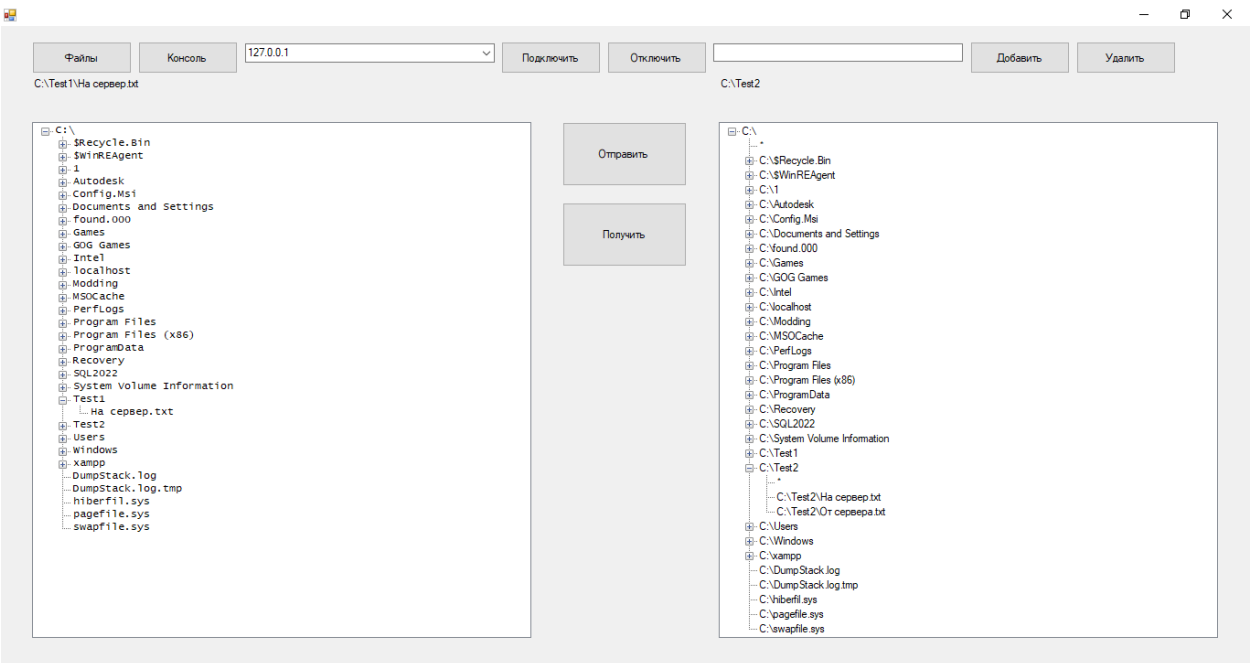


Рисунок 4.10 – Отправка файла на сервер

На рисунке 4.11 представлен процесс получения файла от сервера

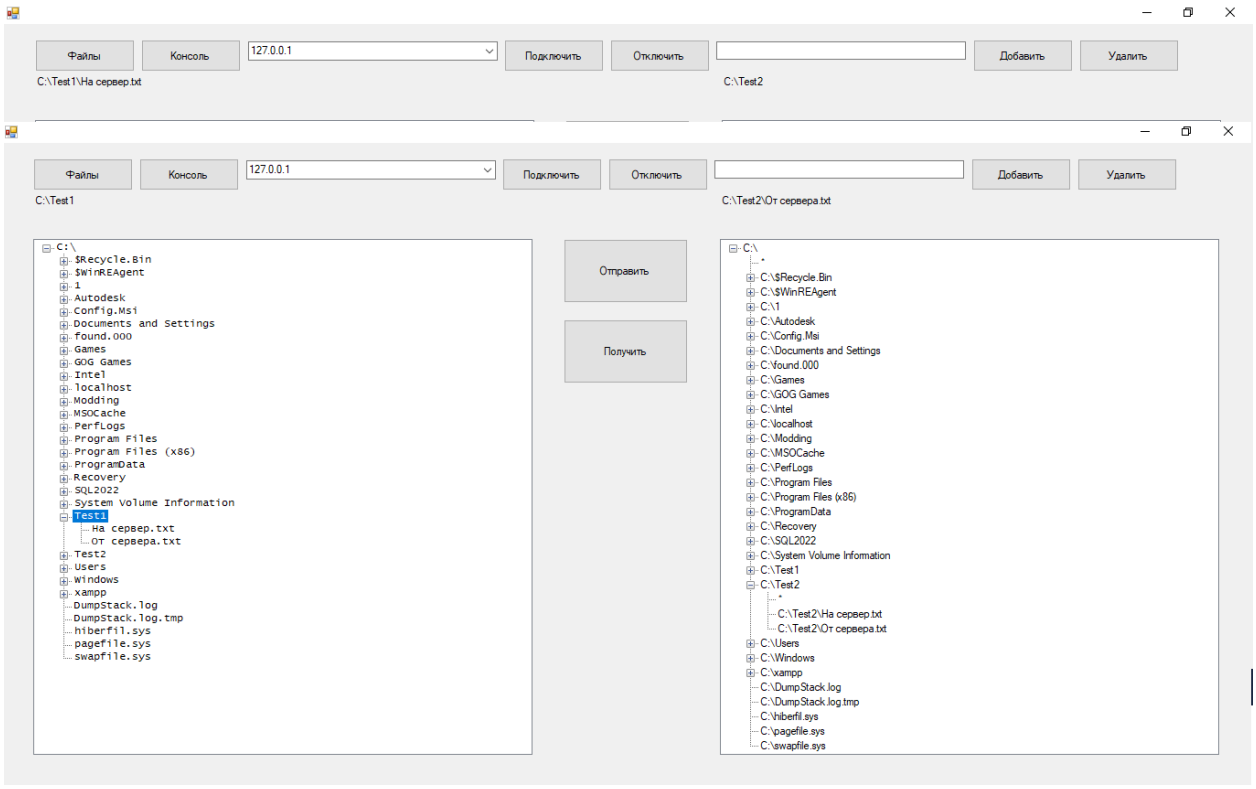


Рисунок 4.11 – Получение файла от сервера

На рисунке 4.12 представлен процесс отключения клиента от сервера.

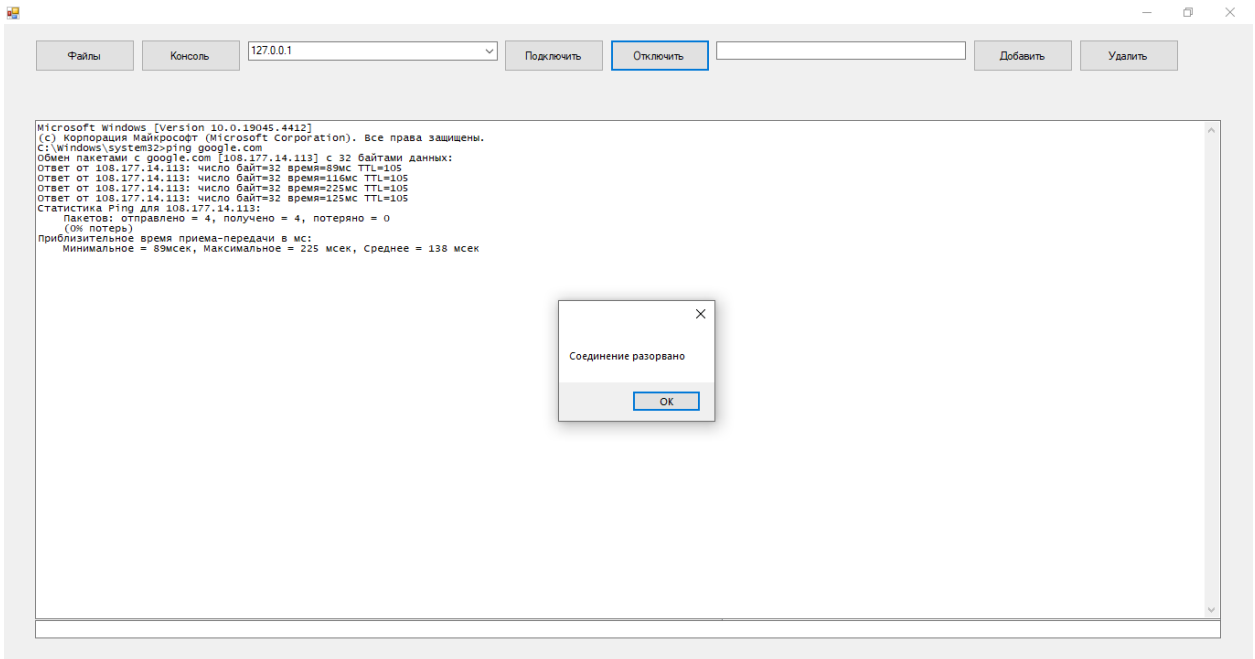


Рисунок 4.12 – Отключение клиента от сервера

На рисунке 4.13 представлен процесс удаления ip-адреса из INI файла.

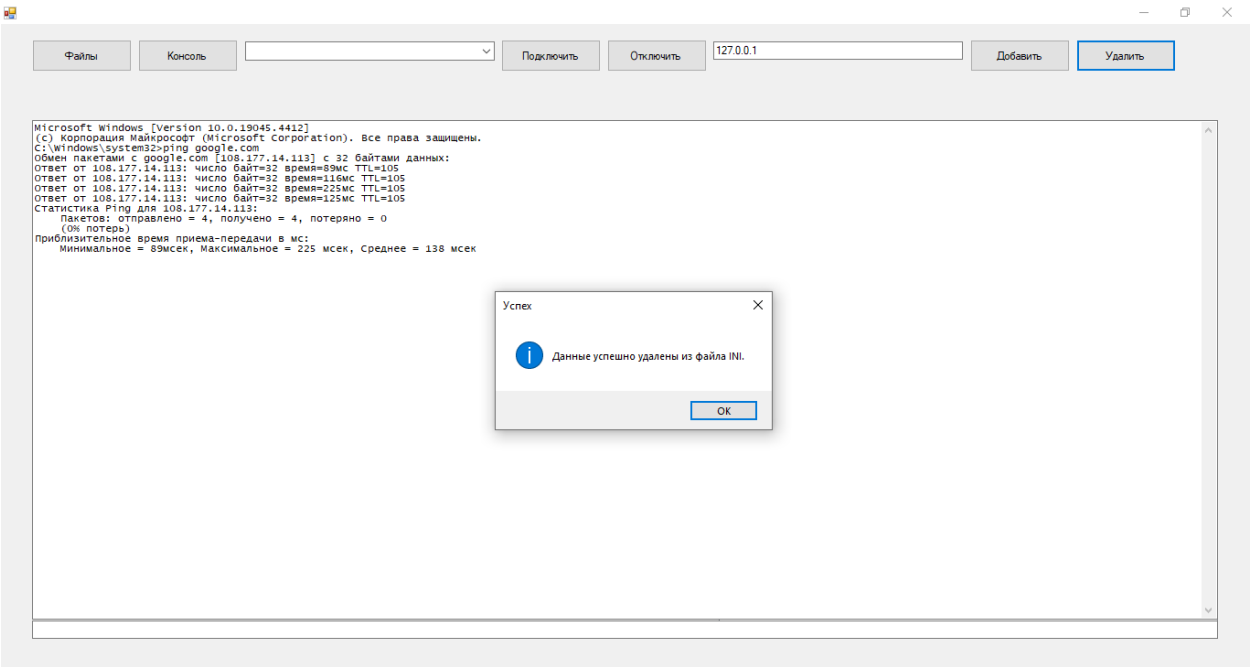


Рисунок 4.13 – Удаления ip-адреса из INI файла

На рисунке 4.14 представлен результат попытки подключения к не работающему или выключенному серверу.

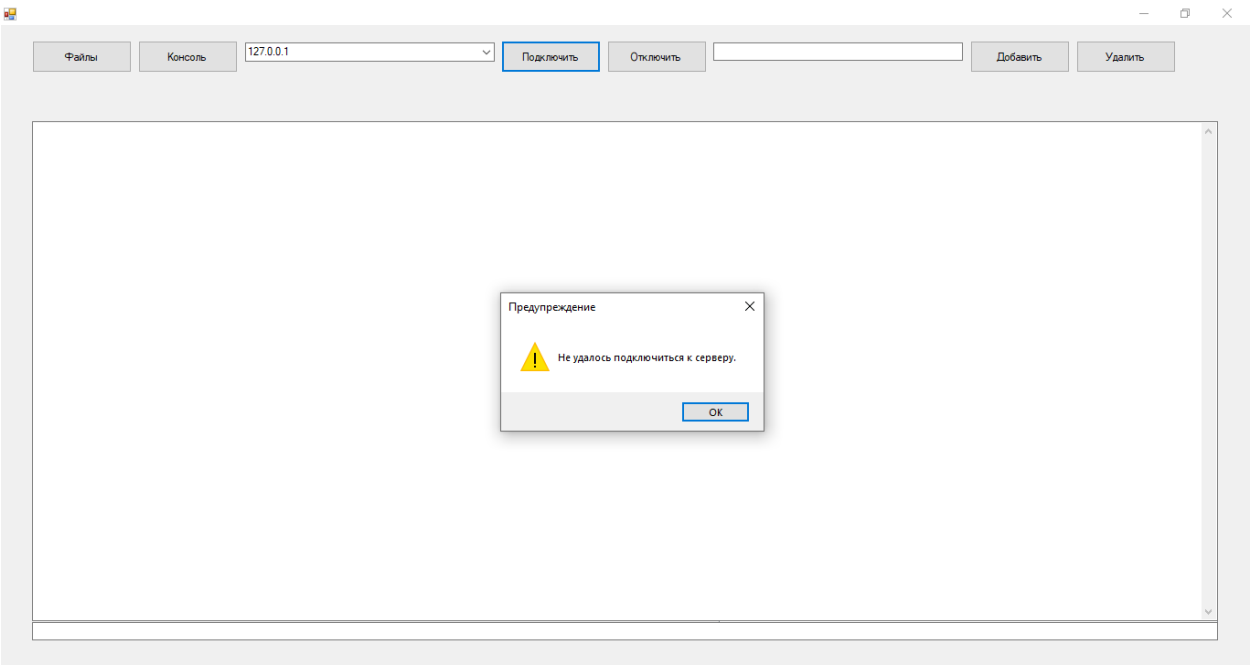


Рисунок 4.14 – Ошибка при подключении к серверу

На рисунке 4.15 представлен результат попытки удалить ip-адрес во время существования активного подключения.

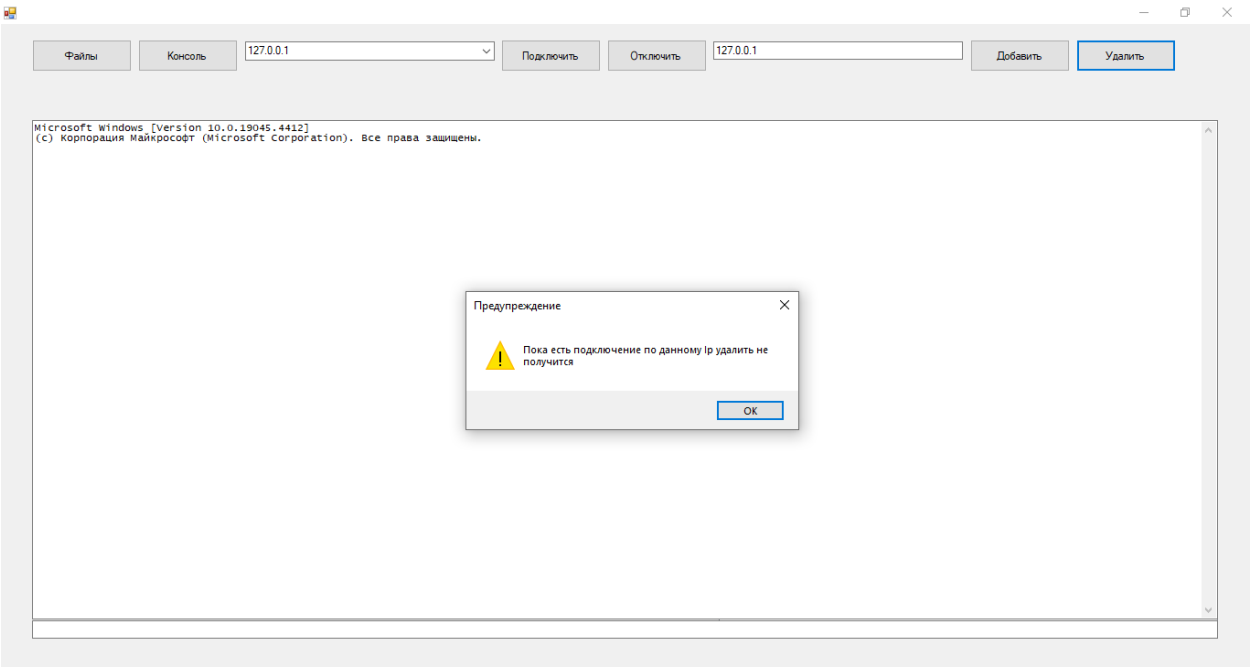


Рисунок 4.15 – Ошибка при удалении ip-адреса

На рисунке 4.16 представлен результат попытки добавления повторяющегося ip-адреса.

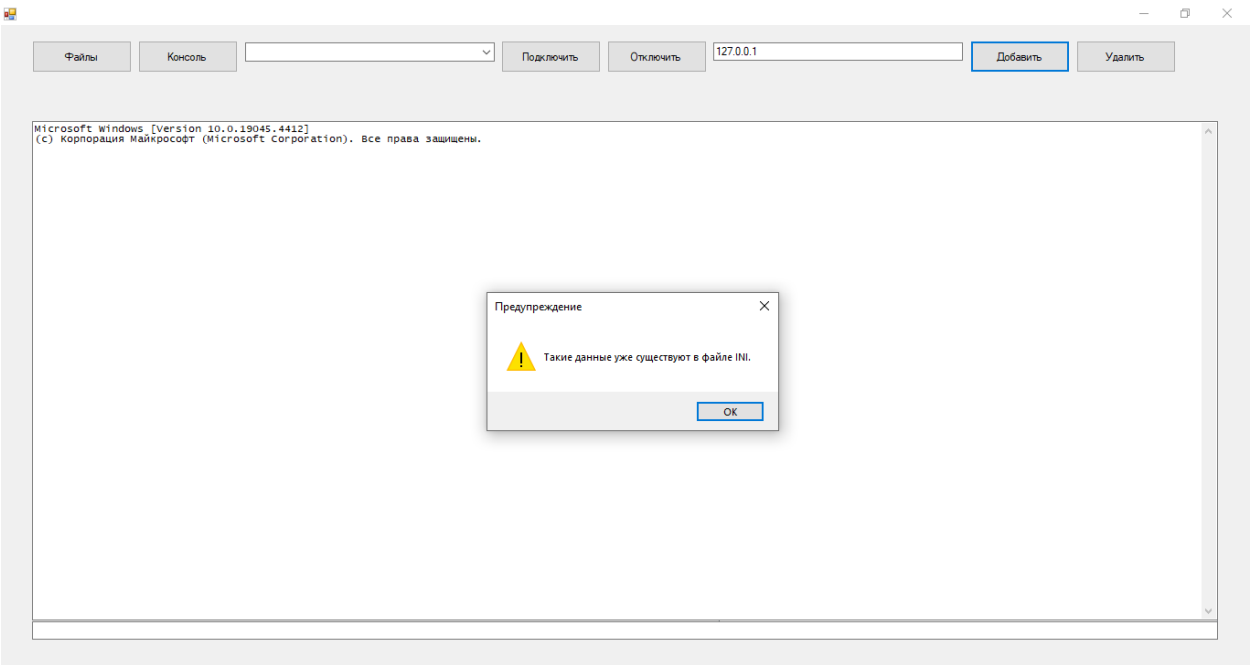


Рисунок 4.16 – Ошибка добавления ip-адреса

4.3 Сборка программной системы

Для компиляции и сборки всех компонентов, входящих в состав программно-информационной системы использовалась Visual Studio 2022.

Программную систему можно запустить в редакторе Visual Studio. Программный продукт можно запустить на операционной системе Windows.

ЗАКЛЮЧЕНИЕ

Преимущества технологий удаленного администрирования заключается в возможности без физического присутствия эффективно управлять компьютерами на больших расстояниях.

Компании, видя, как развиваются информационные технологии, пытаются использовать их выгодно для своего бизнеса, разрабатывая свои программные продукты для того, чтобы оптимизировать процесс устранения неполадок, а также минимизировать затраты времени на устранения возникших проблем. Для повышения эффективности работы IT-администраторов была разработан программно-информационная система для удаленного администрирования компьютеров организации.

Основные результаты работы:

1. Проведен анализ предметной области. Выявлена необходимость использования TCP/IP.

2. Разработана концептуальная модель приложения. Разработана модель протоколов. Определены требования к системе.

3. Осуществлено проектирование приложения. Разработана архитектура серверной части. Разработана архитектура клиентской части.Разработан пользовательский интерфейс клиентской части.

4. Реализована и протестирована программная система. Проведено системное тестирование.

Все требования, объявленные в техническом задании, были полностью реализованы, все задачи, поставленные в начале разработки проекта, были также решены.

Готовый рабочий проект представлен программно-информационной системой для удаленного администрирования компьютеров организации. Программный продукт используется в организации.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Биллиг, В. А. Основы программирования на C# / В.А. Биллиг. - М.: Интернет-университет информационных технологий, Бином. Лаборатория знаний, 2021. - 488 с.

2. Вагнер, Билл C# Эффективное программирование / Билл Вагнер. - М.: ЛОРИ, 2021. - 320 с.

3. Дейтел, П. Как программировать на Visual C# 2012 / П. Дейтел. - М.: Питер, 2018. - 2180 с.

4. Культин, Н. C# в задачах и примерах / Н. Культин. - М.: БХВ-Петербург, 2020. - 1293 с.

5. Фленов, Михаил Библия C# / Михаил Фленов. - М.: БХВ-Петербург, 2021. - 560 с.

6. Албахари Дж. C# 6.0. Справочник. Полное описание языка [Текст] / Дж . Албахари, Б. Албахари — 6-е изд. — Москва: Вильямс, 2016. — 1040 с.

7. Зыков, С.В. Введение в теорию программирования. Объектно-ориентированный подход / С.В. Зыков. - 2-е изд., испр. - М. : Национальный Открытый Университет «ИНТУИТ», 2016. - 189 с. : схем. - (Основы информационных технологий). - Библиогр. в кн.

8. Кариев Ч.А. Разработка Windows-приложений на основе Visual C# [Электронный ресурс] : учебное пособие / Ч.А. Кариев. — Электрон. текстовые данные. — Москва, Саратов: Интернет-Университет Информационных Технологий (ИНТУИТ), Вузовское образование, 2017. — 768 с.

9. Microsoft TCP/IP. Учебный курс. - М.: Microsoft Press. Русская Редакция; Издание 3-е, испр., 2021. - 400 с.

10. Microsoft TCP/IP: Учебный курс. - М.: Издательский отдел Русская редакция’ ТОО ’Channel Trading Ltd., 2019. - 392 с.

11. Фейт, С. TCP / IP. Архитектура. Протоколы. Реализация / С. Фейт. - Москва: ИЛ, 2019. - 424 с.

12. Хант, К. TCP/IP. Сетевое администрирование / К. Хант. - М.: СПб: Символ-Плюс; Издание 3-е, 2018. - 816 с.

13. Дэвис, Джозеф Microsoft Windows Server 2003. Протоколы и службы TCP/IP. Техническое руководство / Джозеф Дэвис , Томас Ли. - М.: Эком, 2019. - 752 с.

14. Паркер TCP/IP. Для профессионалов / Паркер, Сиян Тим; , К.. - М.: СПб: Питер; Издание 3-е, 2021. - 859 с.

15. Маклафлин, Б. Объектно-ориентированный анализ и проектирование / Б. Маклафлин, Г. Поллайс, Д. Уэст. - М.: Питер, 2017. - 857 с.

16. Приемы объектно-ориентированного проектирования: Паттерны проектирования / Э. Гамма и др. - М.: Addison Wesley Longman, Inc., 2019. - 368 с.

17. Стиллмен, Э. Изучаем C# / Э. Стиллмен, Дж. Грин. - М.: Питер, 2017. - 688 с.

18. Грекул, В. И. Методические основы управления ИТ-проектами / В.И. Грекул, Н.Л. Коровкина, Ю.В. Куприянов. - М.: Интернет-университет информационных технологий, Бином. Лаборатория знаний, 2021. - 392 с.

19. Бабаев С.И., Компьютерные сети. Часть 3. Стандарты и протоколы : учебник / С.И. Бабаев, Б.В. Костров, М.Б. Никифоров. — М.: КУРС, 2018. — 176 с.

20. Сергеев А.Н. Основы локальных компьютерных сетей: учебное пособие для СПО/ А.Н. Сергеев. – 3-е изд., стер. – Санкт-Петербург: Лань, 2023 . – 184 с.

ПРИЛОЖЕНИЕ А

Представление графического материала

Графический материал, выполненный на отдельных листах, изображен на рисунках А.1–А.8.

Сведения о ВКРБ

Минобрнауки России

Юго-Западный государственный университет

Кафедра программной инженерии

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ПО ПРОГРАММЕ БАКАЛАВРИАТА

«Веб-платформа для анализа и визуализации статистических данных киберспортивной игры Counter-Strike 2»

Руководитель ВКРБ
к.т.н, доцент
Аникина Елена Игоревна

Автор ВКРБ
студент группы ПО-01б
Авилов Иван Александрович

Рисунок А.1 – Сведения о ВКРБ

Цели и задачи разработки

Цель настоящей работы – разработка веб-платформы для анализа и визуализации статистических данных киберспортивной игры Counter-Strike 2.

Данная платформа должна предоставить киберспортивным командам, тренерам и аналитикам мощный инструмент для глубокого анализа игровых метрик и поддержки принятия решений на основе данных. Для достижения поставленной цели необходимо решить следующие задачи:

– провести анализ предметной области;

– разработать концептуальную модель веб-платформы;

– создать базу данных всех киберспортивных матчей;

– создать нейронную сеть для прогнозирования матчей;

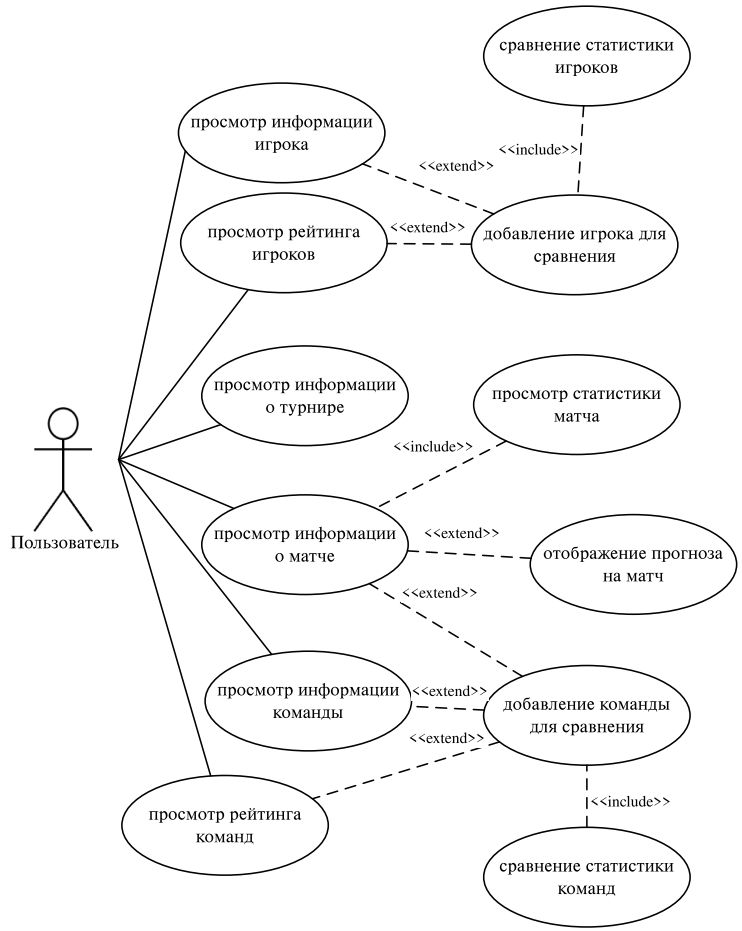
– реализовать вывода статистики на веб-платформу;

– реализовать анализа метрик из .dem записей игр.

				ВКРБ 2068443.09.03.04.24.001		
	Фамилия И. О.	Подпись	Дата		Лит.	Масса	Масштаб
Автор работы	Абилов Н.А.			Цели и задачи разработки			
Руководитель	Ахмадеев Е.И.						
Нормоконтроль	Чугунова А.А.				Лист 2	Листов 8	
				Выпускная квалификационная работа бакалавра	ЮЗГУ ПО-01б		

Рисунок А.2 – Цель и задачи разработки

Диаграмма прецедентов



				ВКРБ 2068443.09.03.04.24.001	
	Фамилия И. О.	Подпись	Дата		
Автор работы	Абилов Н.А.			Диаграмма прецедентов	Лит. / Масса / Масштаб
Руководитель	Аникин Е.И.				
Нормоконтроль	Чапыгин А.А.				Лист 3 / Листов 8
				Выпускная квалификационная работа бакалавра	ПЭГУ ПО-016

Рисунок А.3 – Диаграмма прецедентов

Концептуальные классы веб-платформы

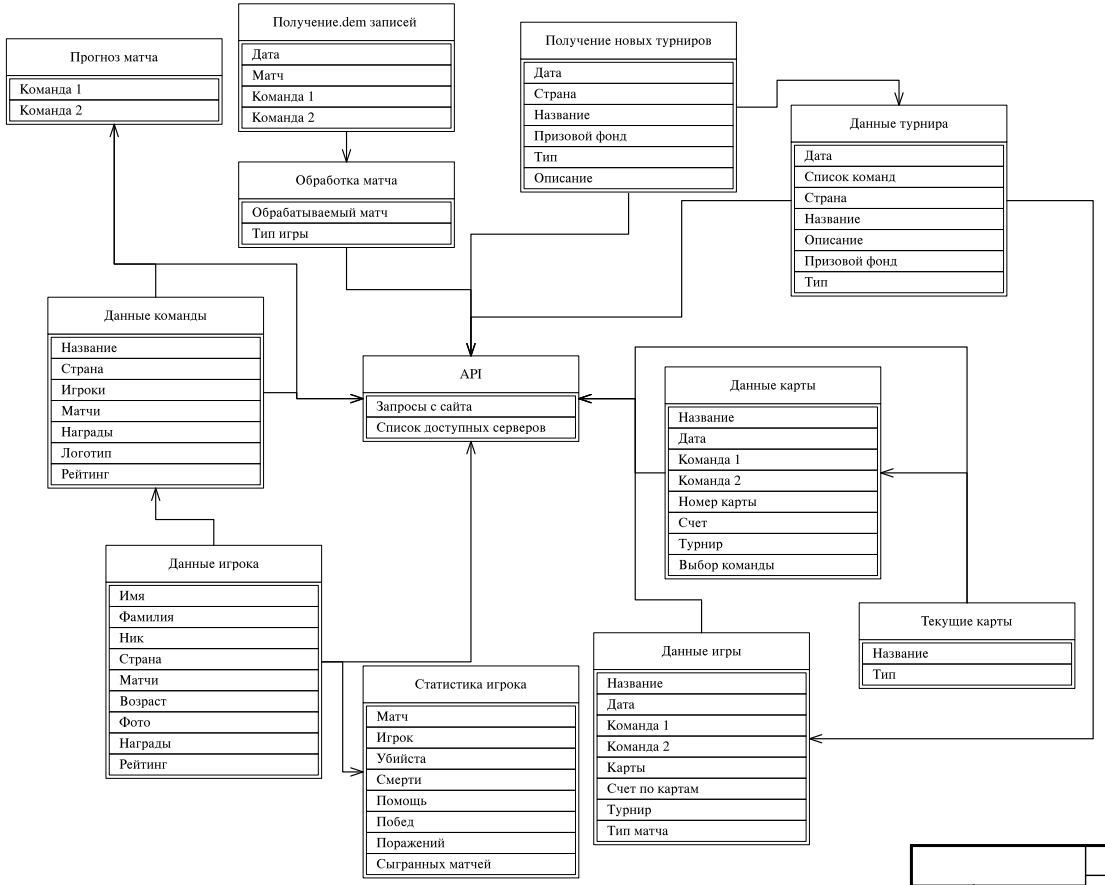
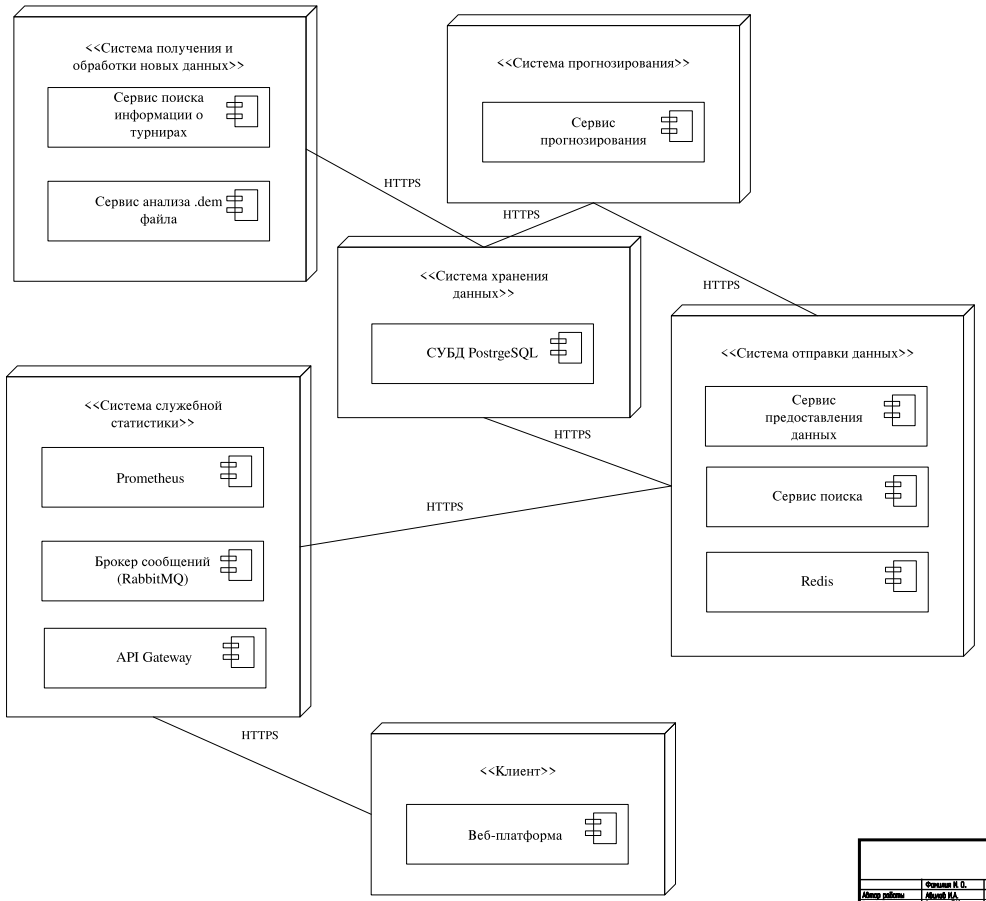


Рисунок А.4 – Концептуальные классы веб-платформы

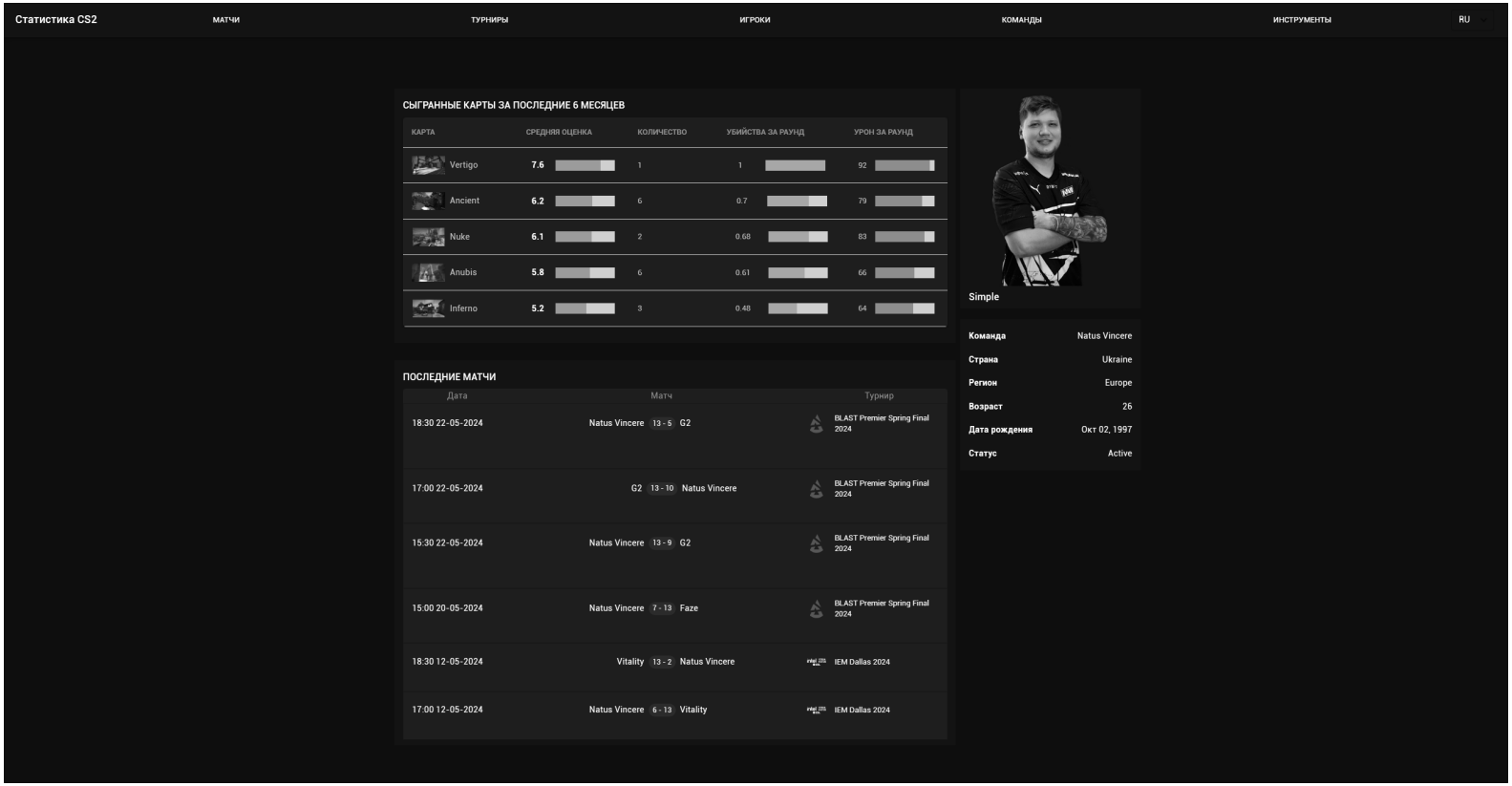
Архитектура программной системы



				ВКРБ 2068443.09.03.04.24.001	
	Фамилия И. О.	Подпись	Дата		
Автор работы	Абакий Н.А.			Архитектура программной системы	Лит. / Масса / Масштаб
Руководитель	Анашкин Е.И.				
Нормоконтроль	Чапыгин А.А.				Лист 5 / Листов 8
				Выпускная квалификационная работа бакалавра	ЮЗГУ ПО-01б

Рисунок А.5 – Архитектура программной системы

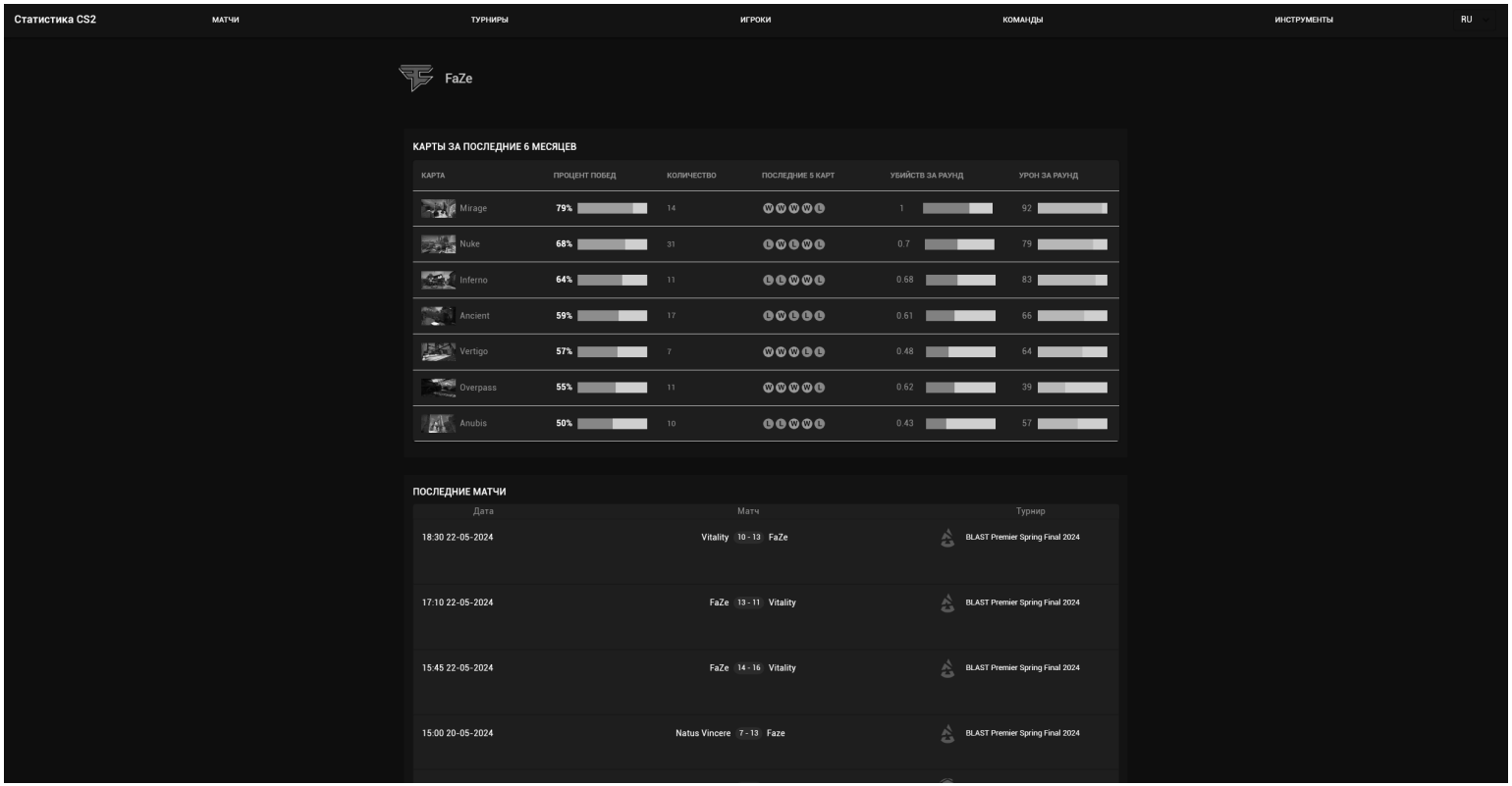
Страница игрока



				ВКРБ 2068443.09.03.04.24.001	
				Страница игрока	Лит. / Масса / Масштаб
Автор работы	Фамилия И. О.	Подпись	Дата		
Руководитель	Абакий И.А.				
Нормоконтроль	Аникин Е.И.				Лист 6 / Листов 8
	Чапыгин А.А.			Выпускная квалификационная работа бакалавра	ЮЗГУ ПО-01б

Рисунок А.6 – Страница игрока

Страница команды



				ВКРБ 2068443.09.03.04.24.001		
	Фамилия И. О.	Подпись	Дата		Лит.	Масса	Масштаб
Автор работы	Абакий Н.А.			Страница команды			
Руководитель	Аникин Е.И.				Лист 7	Листов 8	
Нормоконтроль	Чапыгин А.А.			Выпускная квалификационная работа бакалавра	ЮЗГУ ПО-01б		

Рисунок А.7 – Страница команды

Заключение

В данной выпускной квалификационной работе был разработан и внедрен комплексный инструмент для анализа и визуализации статистических данных киберспортивной игры Counter-Strike 2 (CS2). Данный проект направлен на улучшение игровых навыков и стратегий киберспортивных команд путем предоставления глубокого анализа игровых данных и прогнозов.

Основные результаты работы:

1. Проведен анализ предметной области киберспорта и выявлены ключевые метрики, необходимые для анализа и прогнозирования игровых результатов.

2. Разработана и реализована база данных для хранения информации о матчах, игроках, командах и турнирной статистике.

3. Создана нейронная сеть для прогнозирования исходов матчей на основе исторических данных и текущих игровых метрик.

4. Реализованы удобные интерфейсы для отображения статистики и прогнозов на веб-платформе, обеспечивая интуитивно понятный доступ к информации.

5. Осуществлен анализ метрик из .dem файлов игр для получения детальных данных о каждом матче.

6. Интегрированы внешние API для автоматического обновления данных о турнирах, матчах и игроках, что обеспечивает актуальность и точность информации.

7. Использованы технологии контейнеризации и оркестрации с Docker и Kubernetes для обеспечения высокой производительности и масштабируемости системы.

8. Внедрен брокер сообщений RabbitMQ для управления потоками данных и коммуникацией между микросервисами.

9. Проведены интеграционные и нагрузочные тестирования, которые подтвердили стабильную работу платформы под высокой нагрузкой.

Рисунок А.8 – Заключение