Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: Компьютерные системы и сети

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

на тему

FTP-КЛИЕНТ

БГУИР КП 1-40 01 01 017 ПЗ

Студент Виктор В.В.

Руководитель Шамына А.Ю.

Минск 2025

**СОДЕРЖАНИЕ**

|  |  |
| --- | --- |
| Содержание…..……………………………………………………………… | 2 |
| Введение…………………………………………………………………….. | 3 |
| 1 Анализ требований к программному средству FTP-клиент…..…...…... | 4 |
| 1.1 Обзор существующих аналогов…………………………………...… | 4 |
| 1.2 Назначение разработки………………………………………………. | 4 |
| 1.3 Состав выполняемых функций……………………………………… | 5 |
| 2 Анализ требований к программе и разработка функциональных требований…………………………………………………………………... | 5 |
| 2.1 Описание функциональности системы……………………………... | 5 |
| 2.2 Спецификация функциональных требований……………………… | 6 |
| 3 Проектирование программного средства…………..…………………… | 9 |
| 3.1 Структура проекта…………………………………………………… | 9 |
| 3.2 Методы программного средства………………..…………………… | 10 |
| 4 Тестирование, проверка работоспособности и анализ полученных результатов…………………………………………………………………... | 13 |
| 4.1 Тестирование программного средства……………………………… | 13 |
| 4.2 Анализ полученных результатов……………………………............. | 16 |
| 5 Руководство по эксплуатации…………………….……………………… | 17 |
| 5.1 Эксплуатация…………………………………………………………. | 17 |
| Заключение………………………………………………………………….. | 24 |
| Приложение А………………………………………………………............. | 25 |

**ВВЕДЕНИЕ**

Современные информационные системы всё чаще требуют надёжного и удобного способа удалённого обмена файлами между клиентами и серверами. Одним из распространённых решений является использование протокола FTP (File Transfer Protocol), который обеспечивает передачу файлов по сети. FTP активно применяется в системах резервного копирования, обновления программного обеспечения, управления веб-сайтами и других сферах, где необходимо централизованное хранение и обработка данных.

Целью данного курсового проекта является разработка FTP-клиента на языке программирования C#, обеспечивающего базовую функциональность взаимодействия с FTP-сервером. Программа позволяет устанавливать соединение с сервером, отображать структуру каталогов и список файлов, хранящихся на сервере, а также выполнять основные операции: просмотр информации о файле, скачивание и удаление файла, загрузку новых файлов, изменение имени и перемещение файла на сервере, создание папок.

Разработка FTP-клиента способствует более глубокому пониманию сетевого взаимодействия и является хорошей основой для изучения клиент-серверных архитектур и создания прикладного программного обеспечения для обмена данными в распределённых системах.

**1** **АНАЛИЗ ТРЕБОВАНИЙ К ПРОГРАММНОМУ СРЕДСТВУ FTP-КЛИЕНТ**

* 1. **Обзор существующих аналогов**

В настоящее время существует множество FTP-клиентов, различающихся по функциональности, удобству использования и технической реализации. Наиболее известные из них — это FileZilla, WinSCP и Total Commander, каждый из которых обладает своими особенностями.

FileZilla — один из самых популярных кроссплатформенных FTP-клиентов с открытым исходным кодом. Он поддерживает протоколы FTP, SFTP, FTPS, обеспечивает стабильную работу при передаче больших объёмов данных, имеет удобный интерфейс с возможностью управления несколькими соединениями. Основными достоинствами являются высокая производительность, поддержка возобновления передачи файлов и встроенный файловый менеджер.

WinSCP — это FTP-, SFTP- и SCP-клиент для Windows, отличающийся простым и понятным интерфейсом. Он ориентирован на безопасную передачу данных и хорошо интегрируется с Windows. Программа поддерживает автоматизацию через скрипты и командную строку, что делает её удобной для администраторов. Минусом может быть отсутствие кроссплатформенности.

Total Commander — файловый менеджер с поддержкой FTP как одной из дополнительных функций. Пользователи могут подключаться к FTP-серверам, просматривать и передавать файлы. Однако интерфейс ориентирован на более опытных пользователей, и многие функции скрыты в контекстных меню или требуют предварительной настройки.

* 1. **Назначение разработки**

Назначением данной разработки является создание простого и удобного в использовании FTP-клиента, предназначенного для установления соединения с удалённым FTP-сервером и выполнения основных операций с файлами. На основе проведённого анализа требуется реализовать следующие функции:

Разработка ориентирована на пользователей, которым необходимо безопасно и эффективно обмениваться файлами с сервером, без использования стороннего программного обеспечения.

**1.3 Состав выполняемых функций**

* подключение к FTP-серверу путем ввода параметров для соединения;
* отображение структуры каталогов и файлов на сервере для удобной навигации по содержимому сервера;
* загрузка файлов с сервера на локальное устройство;
* добавление файлов на сервер с локального устройства в выбранную папку назначения;
* возможность создания папок на сервере;
* возможность изменения имени для файла или папки;
* перемещение файлов на сервере;
* удаление файлов и папок с сервера;
* обработка ошибок и уведомления об успешных и неудачных операциях.

**2** **АНАЛИЗ ТРЕБОВАНИЙ К ПРОГРАММЕ И** **РАЗРАБОТКА ФУНКЦИОНАЛЬНЫХ ТРЕБОВАНИЙ**

**2.1 Описание функциональности системы**

Функционал разрабатываемой системы должен предоставлять пользователю удобную и интуитивно понятную работу с удалёнными файлами на FTP-сервере. Программа ориентирована на пользователей, которым требуется просматривать содержимое FTP-сервера, а также выполнять базовые операции с файлами.

При запуске программы пользователь вводит параметры подключения к FTP-серверу: адрес сервера, порт, логин и пароль. После успешной аутентификации система устанавливает соединение и отображает содержимое сервера в виде дерева каталогов с возможностью навигации.

Пользователь может просматривать структуру файлов и папок на сервере в наглядной форме. Доступно скачивание файлов с сервера на локальный компьютер и удаление файлов с сервера. Пользователь может загружать локальные файлы на сервер в выбранную папку. При необходимости можно обновить список файлов и каталогов. При выборе файла/папки пользователь видит информацию о нем: имя, размер, последнее изменение. Также пользователь может изменить имя файла/папки, удалить файл/папку.

Интерфейс приложения обеспечивает простую навигацию по папкам сервера, лаконичное и понятное оформление, наглядные кнопки для выполнения основных действий (удалить, обновить, переименовать, переместить, добавить папку).

Выход из программы осуществляется через кнопку "крестик". При завершении работы соединение с сервером корректно завершается.

**2.2 Спецификация функциональных требований**

Основные функции программы должны быть реализованы и оформлены в соответствии со следующими требованиями.

Функция подключения к FTP-серверу:

* пользователь должен иметь возможность ввести адрес сервера, порт, имя пользователя и пароль;
* при нажатии кнопки «Подключиться» программа должна установить соединение с сервером и отобразить сообщение об успешном подключении;
* в случае ошибки подключения должно выводиться понятное сообщение с причиной сбоя.

Схема алгоритма представлена на рисунке 2.2.1.

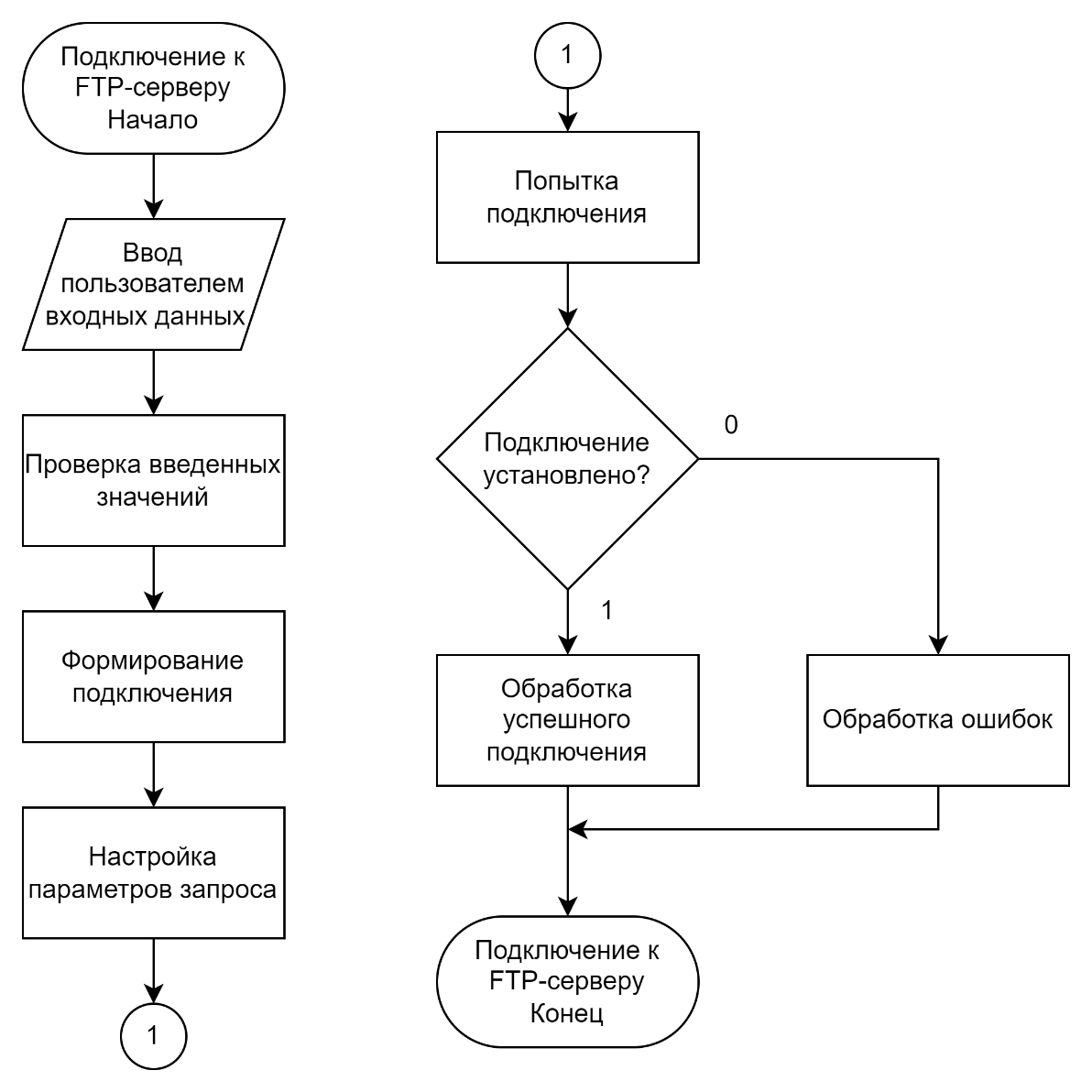


Рисунок 2.2.1 – Подключение к FTP-серверу

Функция скачивания файлов:

* пользователь должен иметь возможность выбрать файл на сервере и место для сохранения на локальном компьютере;
* после завершения загрузки программа уведомляет пользователя об успешной операции или возникшей ошибке.

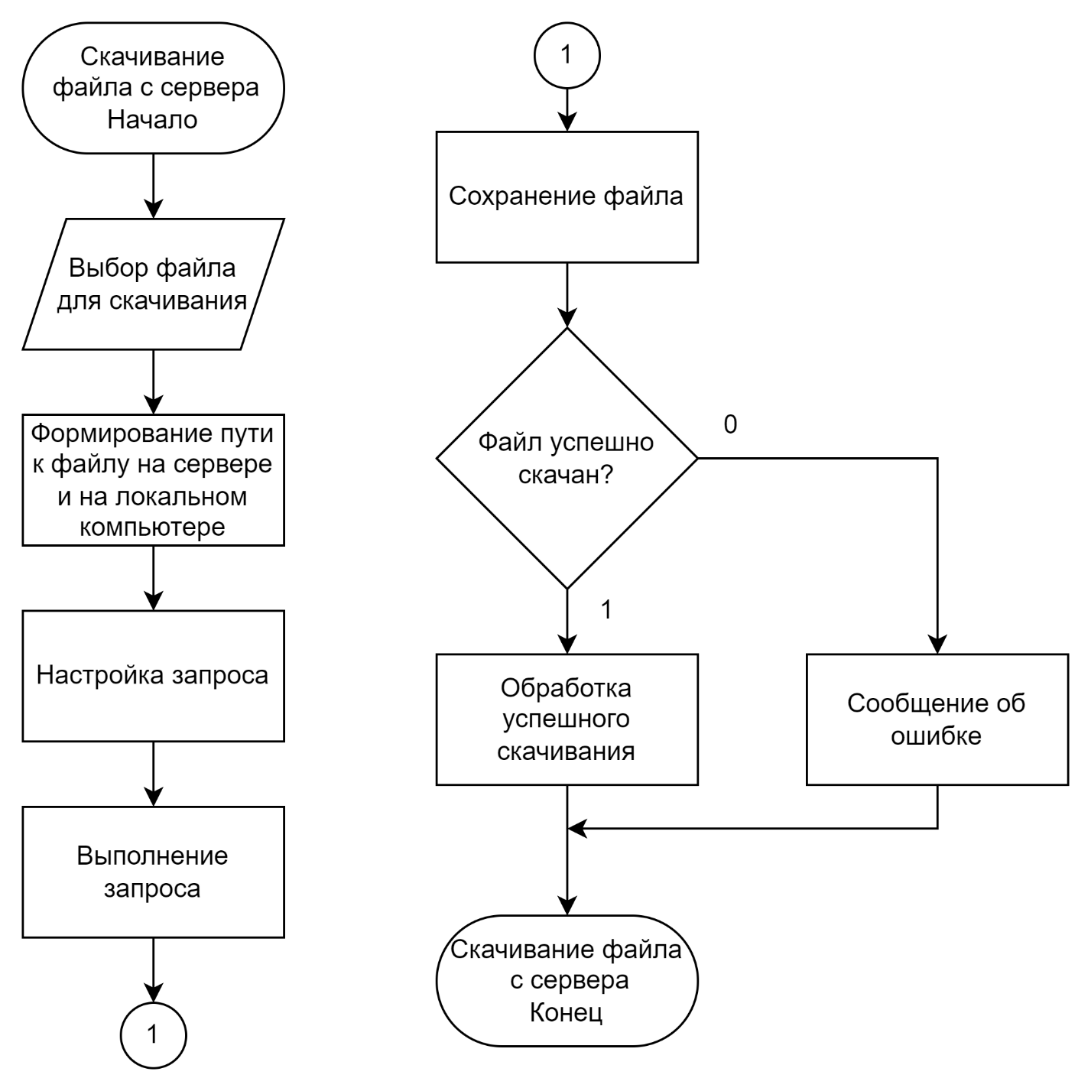
Схема алгоритма представлена на рисунке 2.2.2.

Рисунок 2.2.2 – Скачивание файла с сервера

Функция загрузки файлов на сервер:

* пользователь должен иметь возможность выбрать файл на локальном компьютере и загрузить в определенную папку на сервере;
* при успешной передаче файл должен появиться в списке содержимого сервера без необходимости ручного обновления.

Схема алгоритма представлена на рисунке 2.2.3.

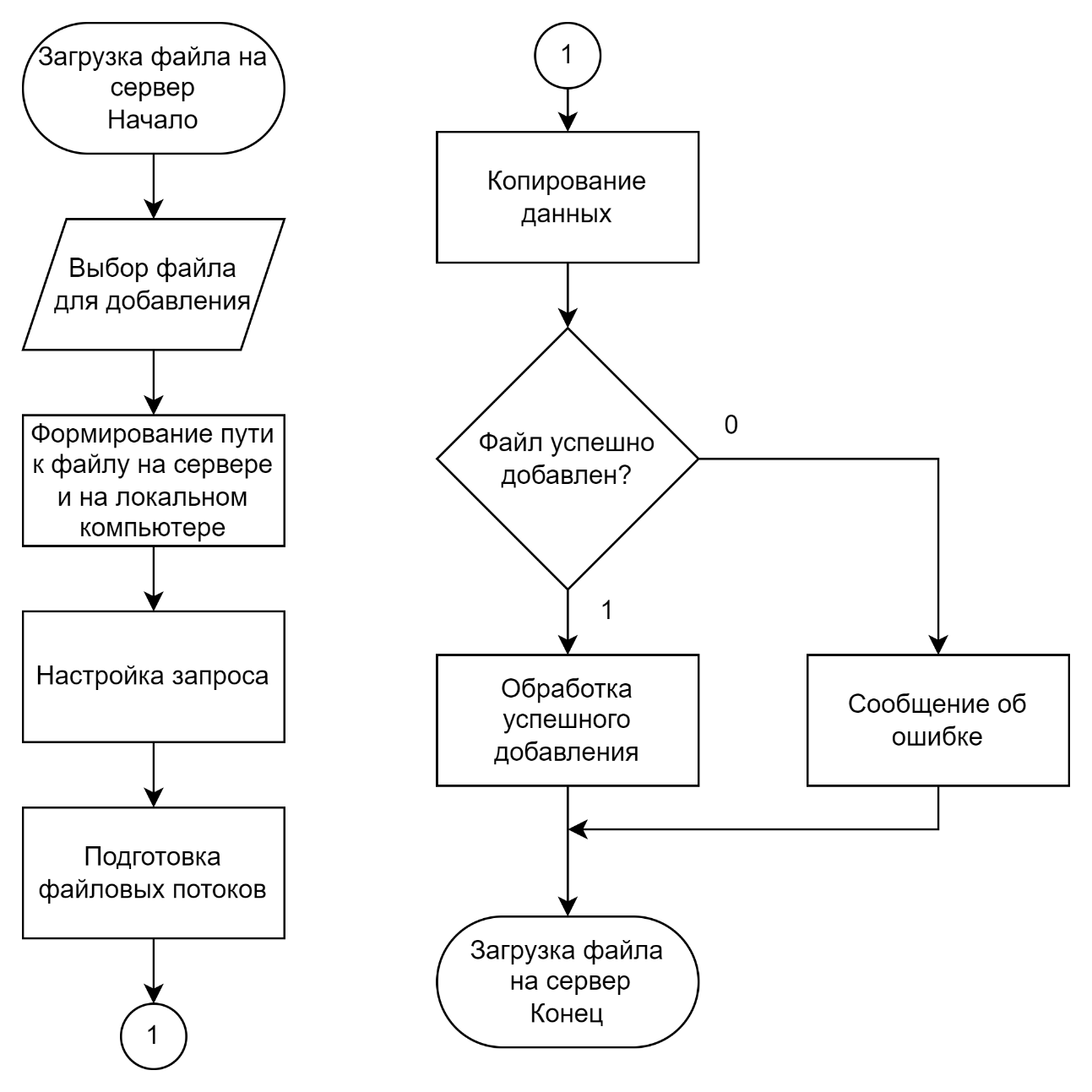


Рисунок 2.2.3 – Загрузка файла на сервер

Функция удаления файлов и папок:

* пользователь должен иметь возможность выбрать файл или папку на сервере и удалить;
* перед удалением должно всплывать окно с подтверждением;
* в случае успешного удаления содержимое каталога обновляется автоматически.

Схема алгоритма представлена на рисунке 2.2.4.

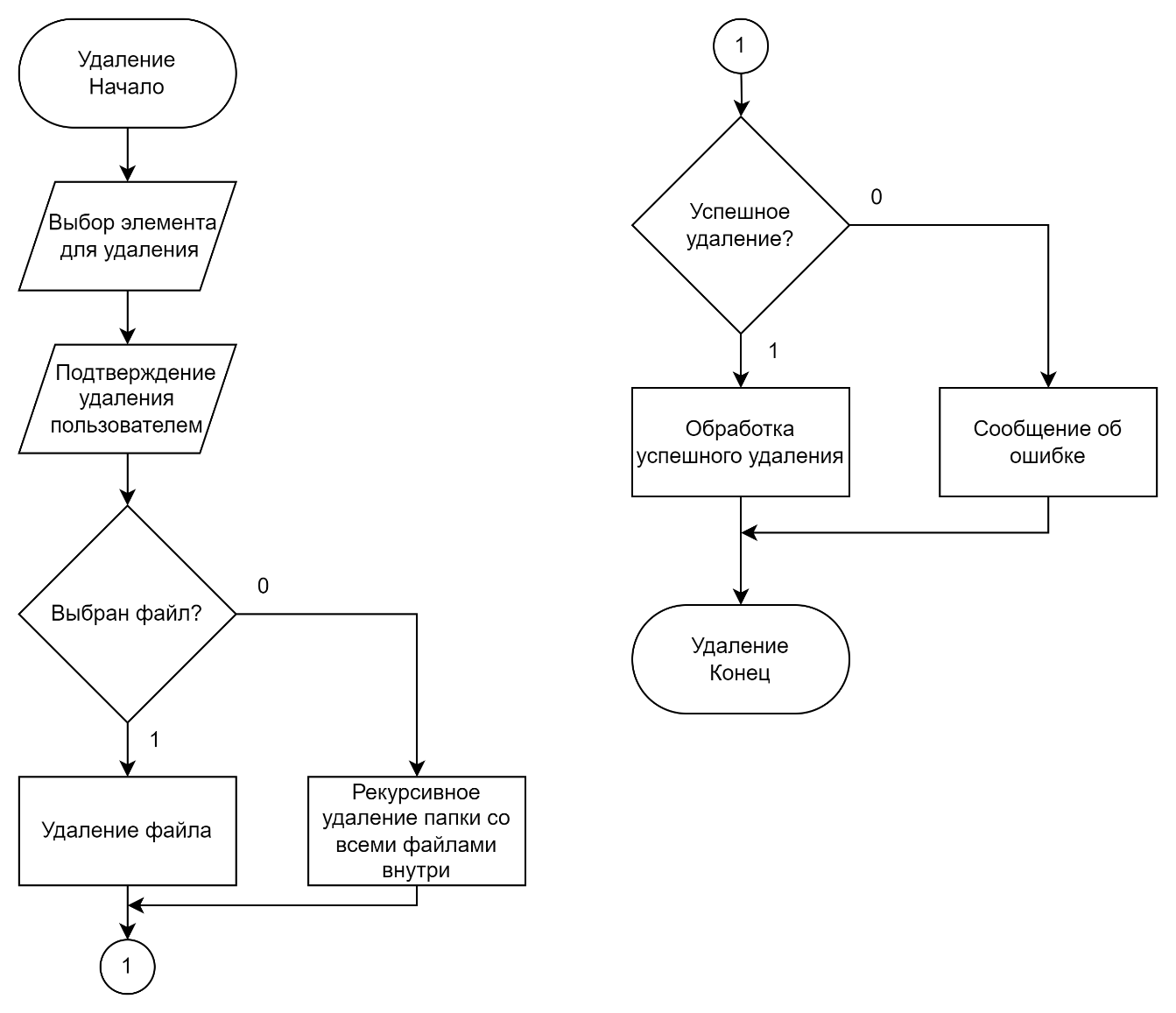


Рисунок 2.2.4 - Удаление

1. **ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА**

**3.1 Структура проекта**

На рисунке 3.1.1 представлена файловая организация проекта.

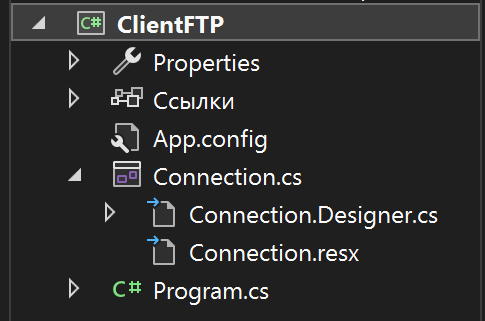


Рисунок 3.1.1 – Структура проекта

Проект ClientFTP состоит из следующих компонентов, организованных в виде файлов и форм Windows Forms. Структура проекта включает визуальную форму, вспомогательные ресурсы и точку входа в приложение.

Основные компоненты проекта:

Файл Connection.cs - главная форма приложения, отвечающая за подключение к FTP-серверу и взаимодействие с файлами и каталогами на сервере. Здесь реализован ввод пользователем необходимых данных: адреса сервера, порта, логина и пароля, размещена кнопка для подключения. Также файл реализует основную логику клиента: управление соединением, работа с FTP-командами и отображение данных.

Connection.Designer.cs — автоматически сгенерированный файл, содержащий код инициализации графических компонентов формы (кнопок, TreeView для отображения файловой структуры и др.). Изменения в этот файл вносятся только через редактор WinForms.

Connection.resx содержит ресурсы формы, такие как строки, изображения или параметры локализации.

Файл App.config - конфигурационный файл проекта.

Файл Program.cs - точка входа в приложение. Содержит метод Main(), который запускает форму Connection.

**3.2 Методы программного средства**

Таблица 3.2.1 – Описание методов

|  |  |  |
| --- | --- | --- |
| Имя | Назначение и характеристика | Параметры |
| private void btnConnect\_Click(object sender, EventArgs e) | Подключение к FTP-серверу | Объект, который вызвал событие; базовый класс для всех аргументов событий |
| private async void LoadFtpInformation() | Загрузка файлов с FTP-сервера | - |
| private void LoadLocalInformation() | Создание каталогов локального компьютера | - |
| private void trvLocal\_BeforeExpand(object sender, TreeViewCancelEventArgs e) | Загрузка файлов локального компьтера | Объект, который вызвал событие; узел - каталог, который необходимо раскрыть |

Продолжение таблицы 3.2.1

|  |  |  |
| --- | --- | --- |
| private async Task LoadFilesRecursive(string currentPath, TreeNode parentNode) | Рекурсивный обход всех файлов на сервере | Текущий путь; корневой узел |
| private string ExtractName(string line) | Извлечение имени файла | Информация о файле |
| private bool IsDirectory(string line) | Определение, является ли текущий узел директорией | Информация об узле |
| private void trvLocal\_ItemDrag(object sender, ItemDragEventArgs e) | Инициирует операцию перетаскивания из локального дерева | Объект, который вызвал событие; объект, который перетаскивают |
| private void trvLocal\_DragEnter(object sender, DragEventArgs e) | Проверка перетаскиваемых данных на локальное дерево | Объект, который вызвал событие; аргумент событий |
| private async void trvLocal\_DragDrop(object sender, DragEventArgs e) | Обработка сброса перетаскиваемых данных на локальное дерево | Объект, который вызвал событие; аргумент событий |
| private void trvServer\_ItemDrag(object sender, ItemDragEventArgs e) | Обработка перетаскивания из серверного дерева | Объект, который вызвал событие; объект, который перетаскивают |
| private void trvServer\_DragEnter(object sender, DragEventArgs e) | Проверка перетаскиваемых данных на серверное дерево | Объект, который вызвал событие; аргумент событий |
| private async void trvServer\_DragDrop(object sender, DragEventArgs e) | Обработка сброса перетаскиваемых данных на серверное дерево | Объект, который вызвал событие; аргумент событий |
| private string GetFullRemotePath(TreeNode node) | Получение полного пути на сервере | Узел дерева |

Продолжение таблицы 3.2.1

|  |  |  |
| --- | --- | --- |
| private string GetLocalTargetPath(TreeNode targetNode) | Получение пути для сохранения на локальном дереве | Узел дерева |
| private async Task UploadFileToFtp(string localPath, string remotePath) | Загрузка файла на сервер | Путь к файлу на локальном компьютере, путь к файлу на сервере |
| private async Task DownloadFileFromFtp(string remotePath, string localPath) | Скачивание файла с сервера | Текущий путь к файлу, путь к файлу на локальном дереве |
| private async void treeView1\_AfterSelect(object sender, TreeViewEventArgs e) | Обработка выбора узла дерева | Объект, который вызвал событие; аргумент событий |
| private async Task<FileInfo> GetFileInformation(string filePath) | Получение информации о выбранном файле | Путь к выбранному файлу |
| private string FormatFileSize(long sizeInBytes) | Преобразование размера файла в байтах в более читаемый вид | Размер файла в байтах |
| private async Task<FileInformation> GetDirectoryInformation(string path) | Получение информации о выбранной директории | Путь к выбранной директории |
| private async Task<long> CalculateDirectorySize(string path) | Вычисление размера директории | Путь к выбранной директории |
| private async void btnDelete\_Click(object sender, EventArgs e) | Обработка нажатия на кнопку «Удалить» | Объект, который вызвал событие; базовый класс для всех аргументов событий |
| private async Task DeleteDirectoryRecursive(string path) | Удаление директории с сервера | Путь к удаляемой директории |

Продолжение таблицы 3.2.1

|  |  |  |
| --- | --- | --- |
| private async Task DeleteFile(string path) | Удаление файла с сервера | Путь к выбранному файлу |
| private async void btnAddFolder\_Click(object sender, EventArgs e) | Обработка добавления папки на сервер | Объект, который вызвал событие; базовый класс для всех аргументов событий |
| private async Task CreateRemoteDirectoryRecursive(string remotePath) | Асинхронное добавление папки на сервер | Путь для новой папки на сервере |
| private async void btnRename\_Click(object sender, EventArgs e) | Переименовывание файла на сервере | Объект, который вызвал событие; базовый класс для всех аргументов событий |
| private void btnMove\_Click(object sender, EventArgs e) | Определение пути для перемещения файла | Объект, который вызвал событие; базовый класс для всех аргументов событий |
| private async Task MoveFileAsync(FileInfo srcFile, string remoteTargetDir) | Перемещение файла на сервере | Объект с информацией о файле; новый путь для перемещения файла |
| private void btnUpdate\_Click(object sender, EventArgs e) | Обновление списка файлов сервера | Объект, который вызвал событие; базовый класс для всех аргументов событий |

**4 ТЕСТИРОВАНИЕ, ПРОВЕРКА РАБОТОСПОСОБНОСТИ И АНАЛИЗ ПОЛУЧЕННЫХ РЕЗУЛЬТАТОВ**

**4.1 Тестирование программного средства**

Для проверки работоспособности разработанного FTP-клиента был использован готовый Docker-образ FTP-сервера stilliard/pure-ftpd. Такой подход позволил быстро развернуть тестовую среду без необходимости настройки отдельного физического сервера.

Запуск FTP-сервера в Docker-контейнере со следующими параметрами:

docker run -d --name ftp\_server -p 21:21 -p 30000-30009:30000-30009 \

-e "PUBLICHOST=localhost" -e "FTP\_USER\_NAME=myuser" -e "FTP\_USER\_PASS=mypass" \

-e "FTP\_USER\_HOME=/home/myuser" -e "PASSIVE\_PORTS=30000:30009" \

-v C:\Users\user\data:/home/myuser stilliard/pure-ftpd

Таблица 4.1.1 – Тестирование программного средства

|  |  |  |  |
| --- | --- | --- | --- |
| Специфика тестирования | Вводимые данные | Ожидаемый результат | Полученный результат |
| Запуск программы | - | Отображение формы с возможностью подключения к серверу, отображение локальных данных | Тест пройден |
| Подключение к несуществующему серверу | Ввод данных для несуществующего сервера | Ошибка подключения | Тест пройден |
| Использование некорректного номера порта | Ввод в поле «Порт» некорректного номера | Сообщение «Введите корректный порт» | Тест пройден |
| Подключение без ввода IP-адреса | Ввод порта | Сообщение «Введите корректный IP-адрес» | Тест пройден |
| Подключение к существующему публичному серверу | Ввод IP-адреса, корректного порта, логина, пароля | Сообщение «Успешное подключение», появление структуры с данными сервера | Тест пройден |
| Выбор любого файла сервера | Нажатие на любой файл сервера | Информация о файле (размер, дата последнего изменения) появляется, кнопки для работы с файлом становятся доступны | Тест пройден |
| Выбор любой папки сервера | Нажатие на любую папку сервера | Информация о папке (размер, дата последнего изменения) появляется, кнопки для работы с папкой становятся доступны | Тест пройден |

Продолжение таблицы 4.1.1

|  |  |  |  |
| --- | --- | --- | --- |
| Попытка изменить имя файла | Нажатие на кнопку «Переименовать» | Появление поля для ввода с кнопками «Ок» и «Назад» | Тест пройден |
| Изменение имени файла | Ввод нового имени файла в специальное поле, нажатие на кнопку «Ок» | Сообщение «Имя файла успешно изменено», изменение имени файла на сервере | Тест пройден |
| Попытка изменить имя папки | Нажатие на кнопку «Переименовать» | Появление поля для ввода с кнопками «Ок» и «Назад» | Тест пройден |
| Изменение имени папки | Ввод нового имени для папки в специальное поле, нажатие на кнопку «Ок» | Сообщение «Имя папки успешно изменено», изменение имени папки на сервере | Тест пройден |
| Отмена изменения имени файла/папки, возвращение назад | Ввод нового имени файла/папки в специальное поле, нажатие на кнопку «Назад» | Имя файла/папки не изменено | Тест пройден |
| Попытка добавить новый файл | Выбор файла на локальном дереве и перенос в выбранное место на сервере | Сообщение «Файл успешно загружен на сервер» | Тест пройден |
| Попытка скачать файл на локальный компьютер | Выбор файла на сервере и перенос в выбранное место на локальном дереве | Сообщение «Файл успешно скачан» | Тест пройден |
| Попытка удалить файл с сервера | Нажатие на кнопку «Удалить» | Сообщение «Вы уверены, что хотите удалить файл {Имя файла}» | Тест пройден |
| Удаление файла | Подтверждение удаления файла | Сообщение «Файл был успешно удален» | Тест пройден |

Продолжение таблицы 4.1.1

|  |  |  |  |
| --- | --- | --- | --- |
| Попытка удалить папку с сервера | Нажатие на кнопку «Удалить» | Сообщение «Вы уверены, что хотите удалить папку {Имя папки}» | Тест пройден |
| Удаление папки с сервера | Подтверждение удаления папки | Сообщение «Папка была успешно удалена» | Тест пройден |
| Попытка обновить данные сервера | Нажатие на кнопку «Обновить» | Перезагрузка данных сервера, успешное обновление | Тест пройден |
| Попытка добавить папку на сервер | Нажатие на кнопку «Добавить папку» | Сообщение «Выберите папку на сервере для добавления новой папки» | Тест пройден |
| Добавление папки на сервер | Ввод имени для новой папки, добавление | Сообщение «Папка {Имя папки} успешно создана», появление папки на сервере | Тест пройден |
| Выход из программы | Нажатие на «Крестик» | Разрыв текущего соединения, закрытие программы | Тест пройден |

**4.2 Анализ полученных результатов**

Программное средство функционирует корректно. Форма отображается корректно и обеспечивают ожидаемое взаимодействие с пользователем. Подключение к FTP-серверу осуществляется успешно, структура каталогов загружается и отображается в виде дерева.

Переход по директориям и выбор файлов выполняются без ошибок. Скачивание и удаление файлов работают корректно — выбранные файлы скачиваются или удаляются с сервера по запросу пользователя. Добавление файлов на сервер также работает корректно. Переименование и перемещение файлов реализованы согласно заданной логике, операции выполняются корректно и изменения отображаются в интерфейсе. Все возможные операции с папками также работают корректно.

Программа работает стабильно, все элементы интерфейса функционируют в соответствии с ожидаемым поведением, ошибок во время выполнения не обнаружено.

**5 РУКОВОДСТВО ПО ЭКСПЛУАТАЦИИ**

**5.1 Эксплуатация**

Чтобы начать использовать программу необходимо запустить ее. Вы увидите окно, представленное на рисунке 5.1.1.

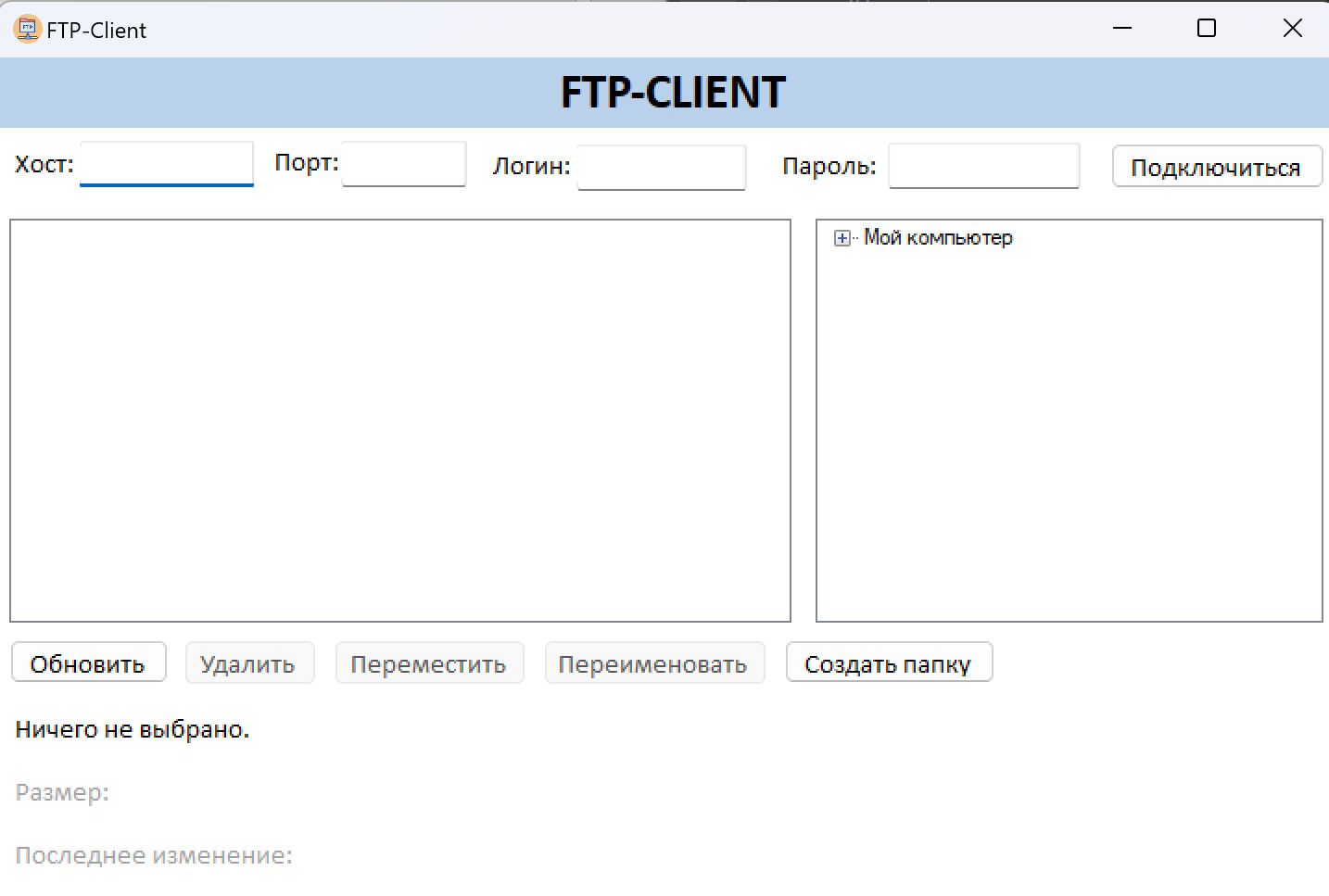
****

Рисунок 5.1.1 – Форма FTP-клиента

При вводе ошибочных данных для подключения к серверу и нажатие на кнопку «Подключиться» программа вернет сообщение, представленное на рисунке 5.1.2

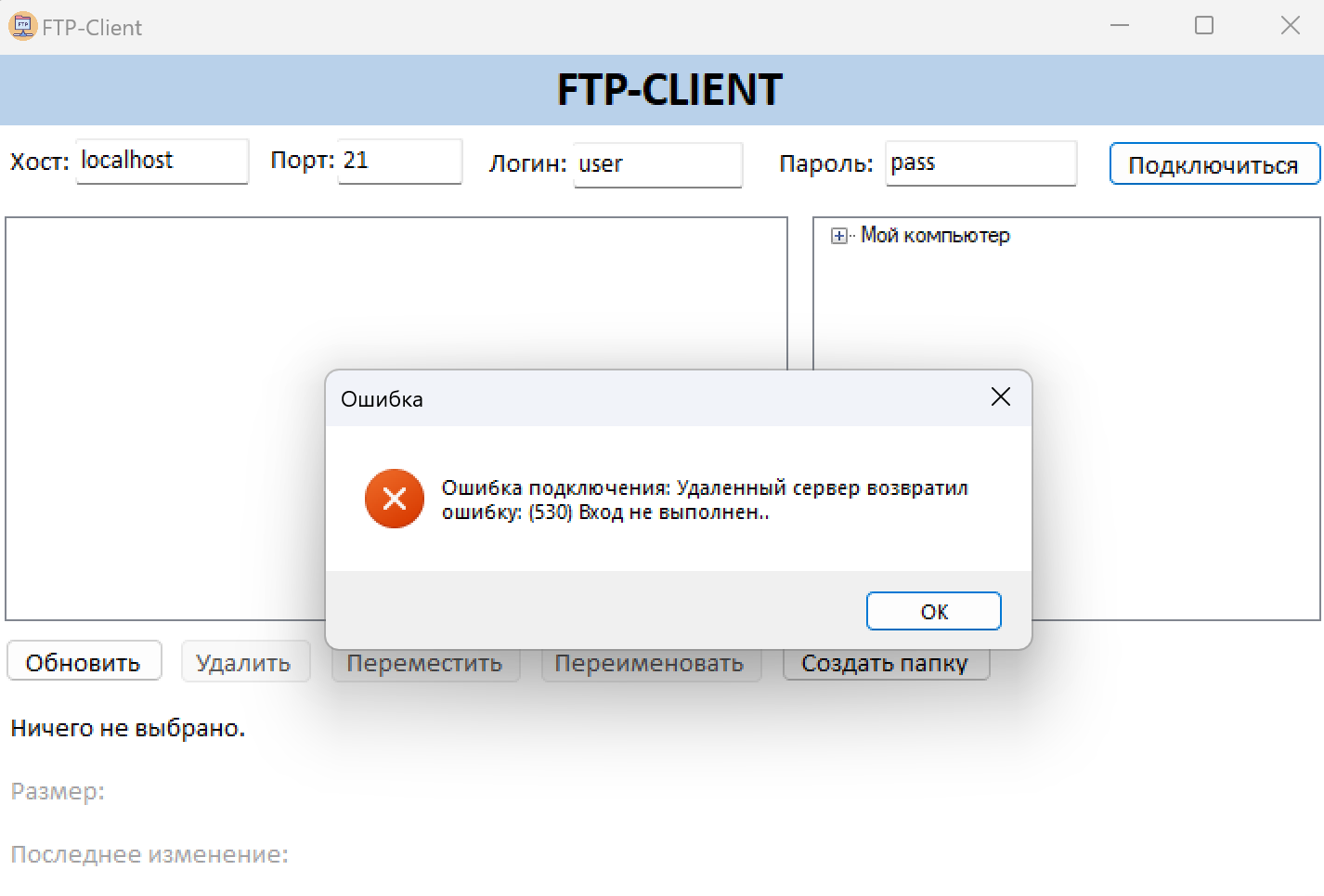


Рисунок 5.1.2 – Ошибка подключения

При вводе корректной информации и успешном подключении программа вернет сообщение, представленное на рисунке 5.1.3.

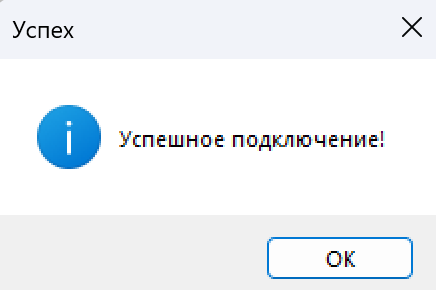
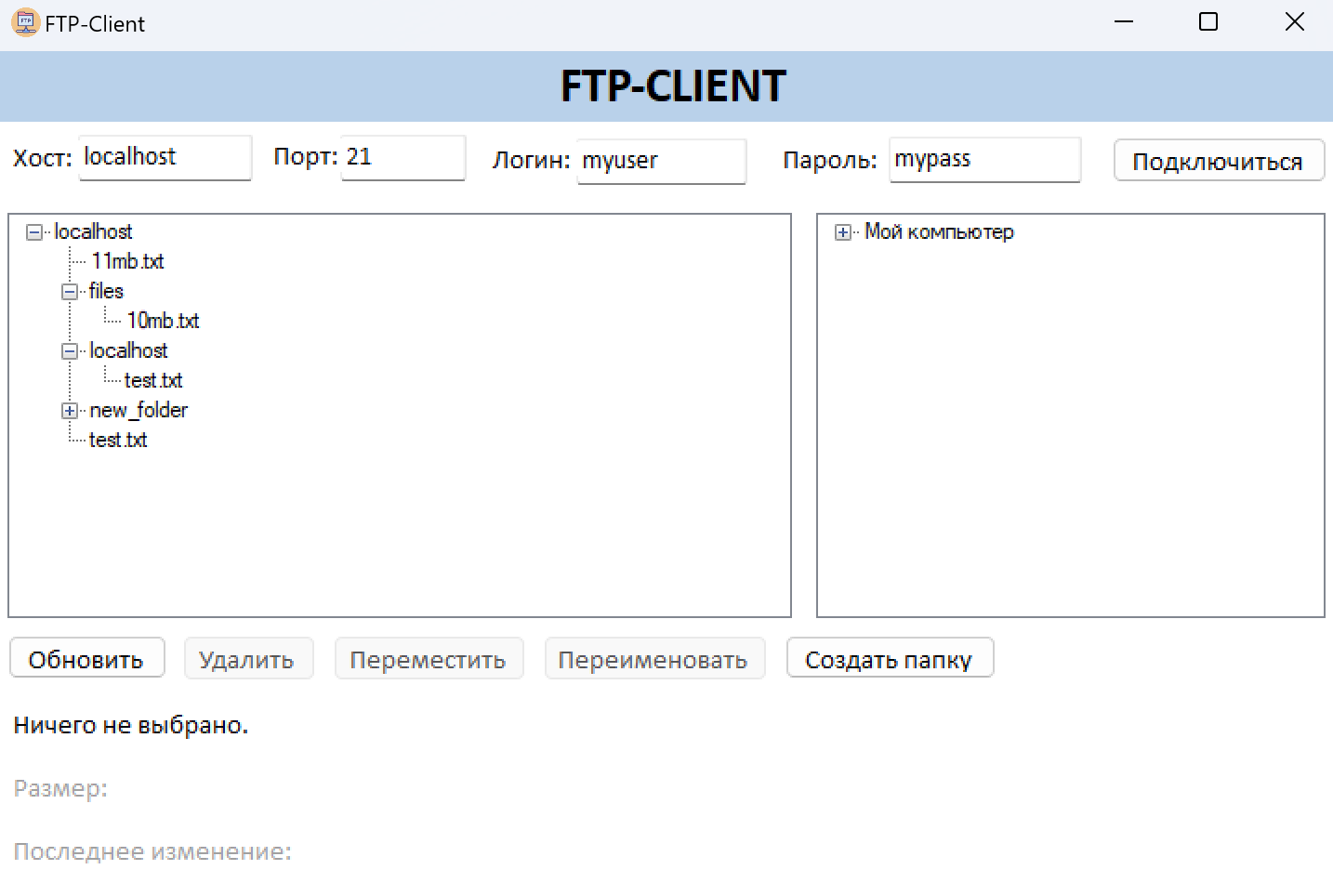
****

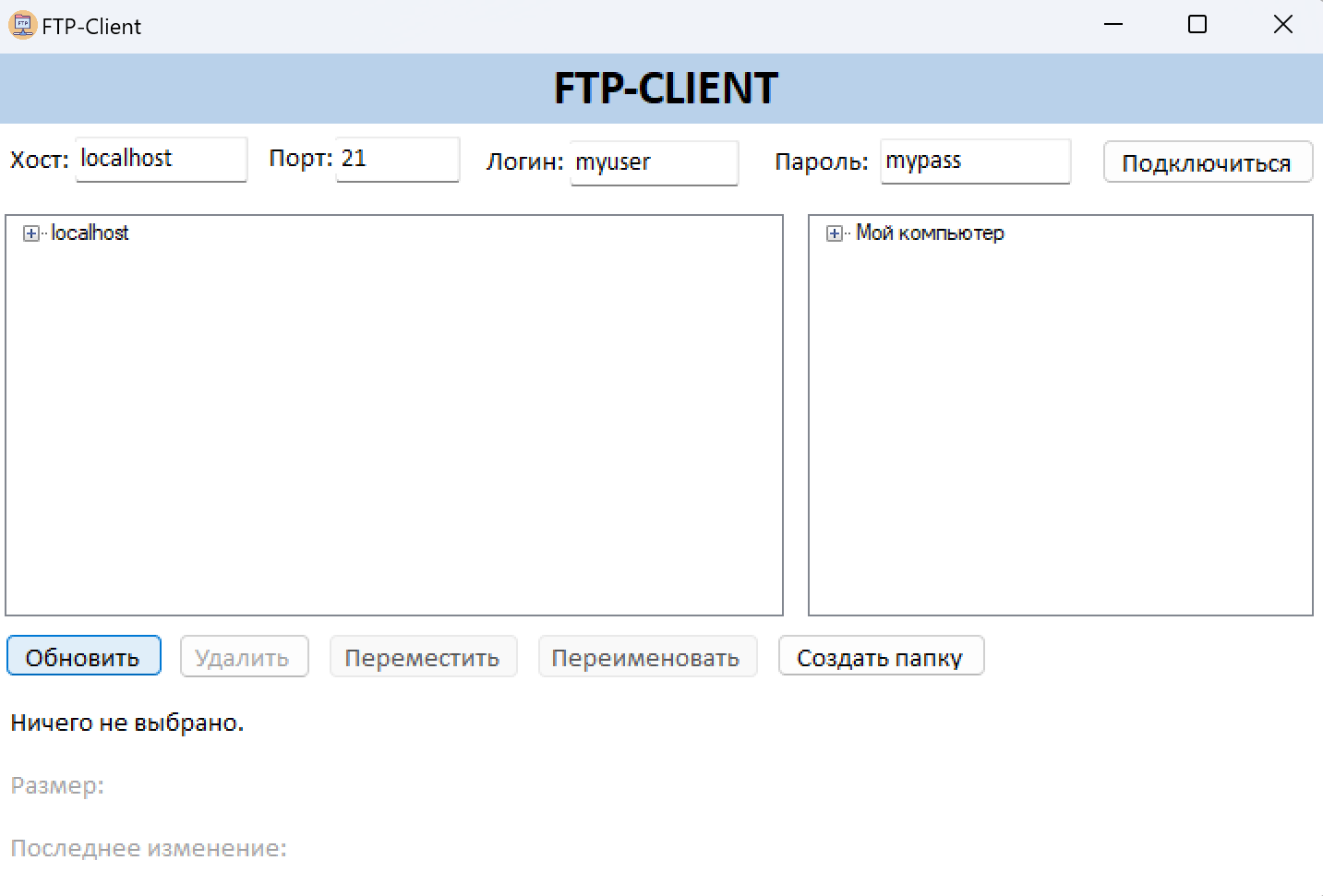
Рисунок 5.1.3 – Успешное подключение

После успешного подключения появится дерево со всеми каталогами сервера, с возможностью их раскрывать и просматривать хранящиеся в них файлы (рис. 5.1.4).

Рисунок 5.1.4 – Успешное подключение

При нажатии на кнопку «Обновить» дерево сервера обновляется (рис 5.1.5). При выборе файла/каталога появляется информация о выбранном элементе, становятся доступны кнопки для работы (рис. 5.1.6, 5.1.7). При переносе данных с локального дерева на серверное появляется сообщение, представленное на рисунке 5.1.8, дерево сервера обновляется. При переносе данных с серверного дерева на локальное появляется сообщение, представленное на рисунке 5.1.9, локальное дерево обновляется.

На картинке 5.1.10 показан загруженный на сервер файл 10mb.txt по пути localhost/localhost и скачанный на локальный компьютер файл 11mb.txt в выделенную папку.

Рисунок 5.1.5 – Обновление дерева сервера

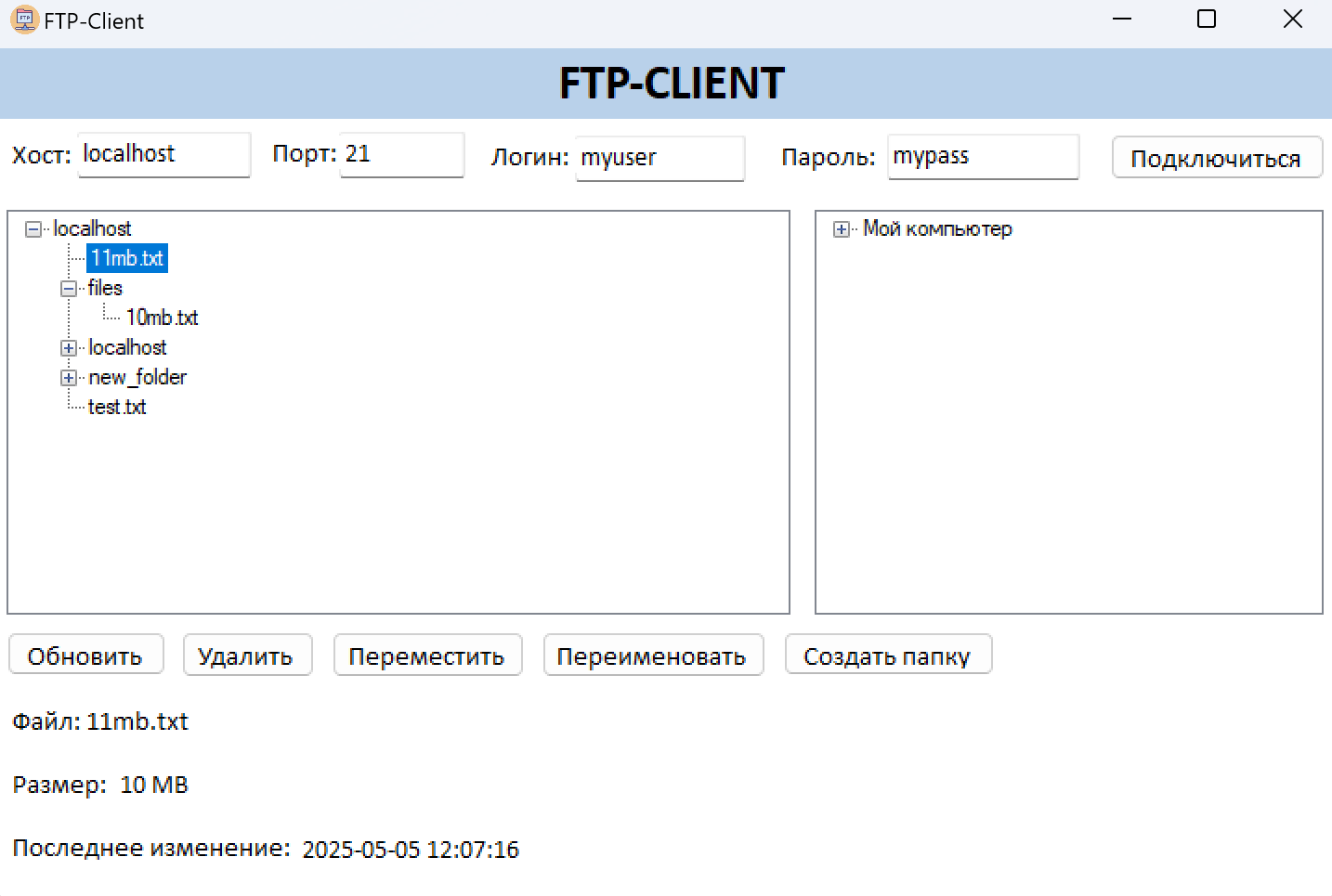
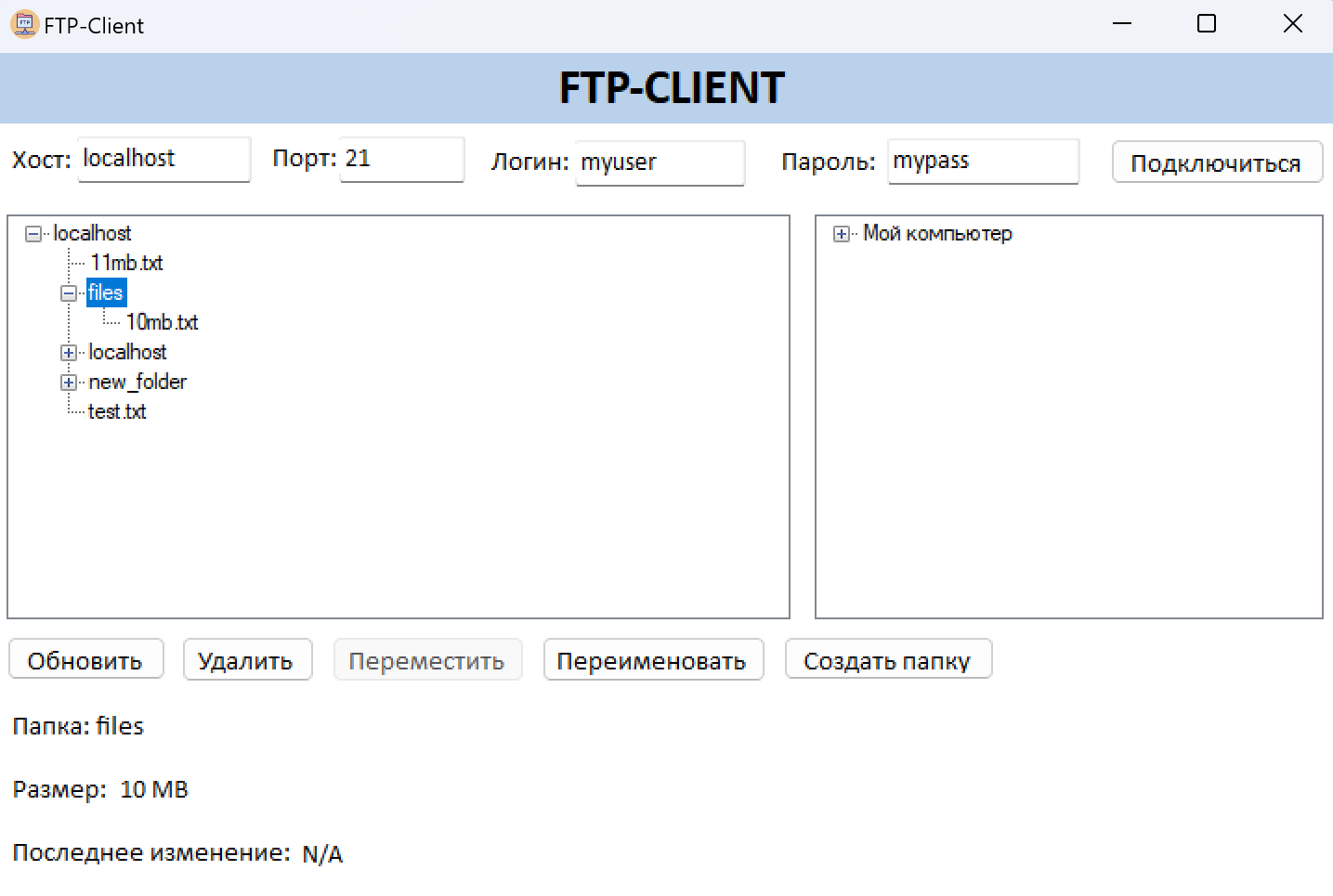


Рисунок 5.1.6 – Информация о выбранном файле

Рисунок 5.1.7 – Информация о выбранной папке

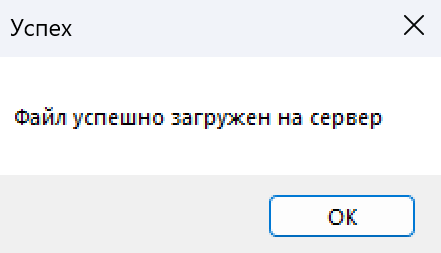


Рисунок 5.1.8 – Успешная загрузка файла на сервер

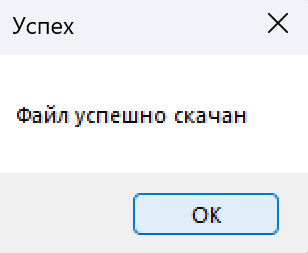
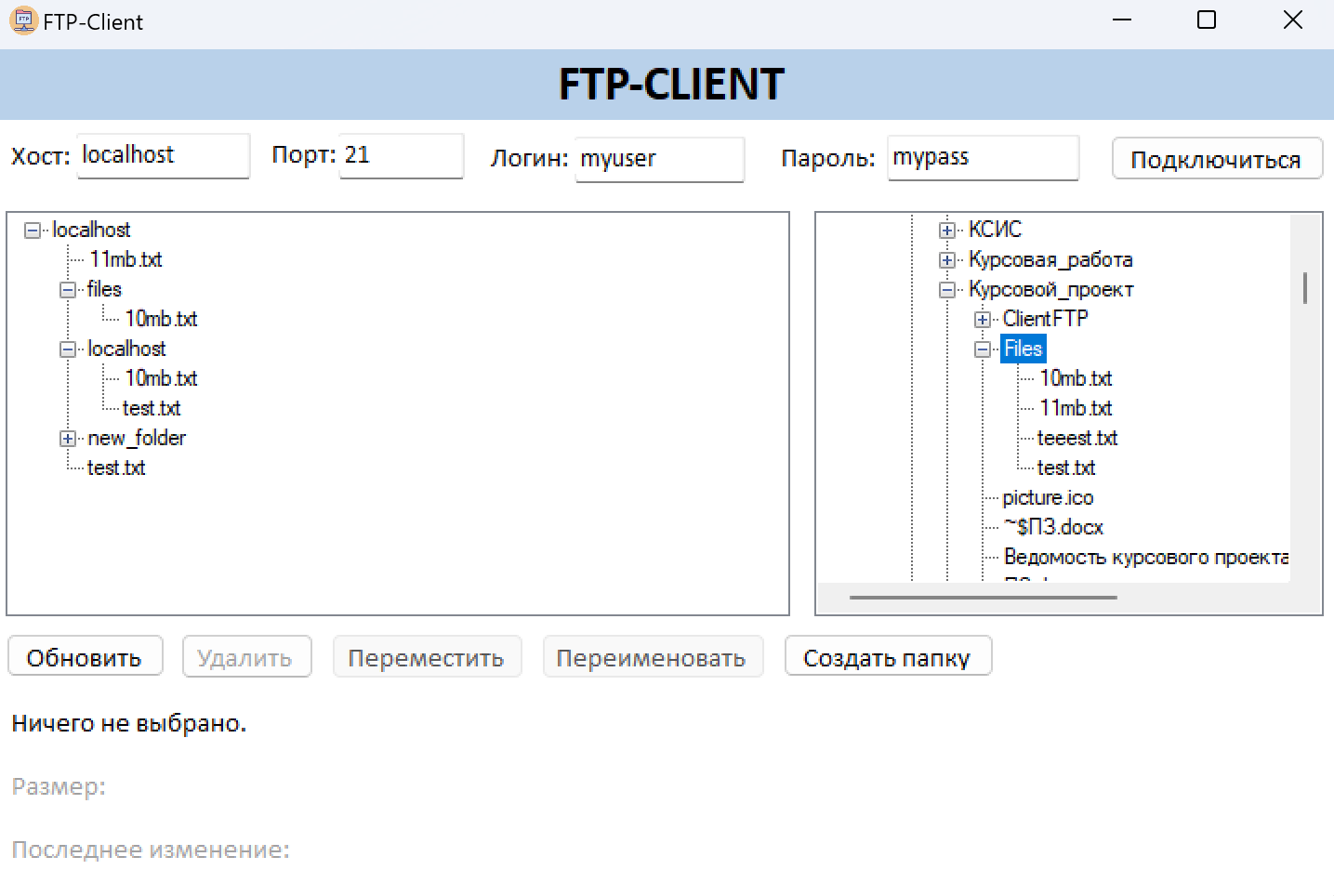


Рисунок 5.1.9 – Успешная загрузка файла на локальный компьютер

Рисунок 5.1.10 – Результат добавления и скачивания файлов

При выборе файла/папки и нажатии кнопки удалить появится сообщение для подтверждения удаления файла или папки. На рисунке 5.1.11 представлено это сообщение. Результат удаления файла test.txt по пути localhost\localhost представлен на рисунке 5.1.12.

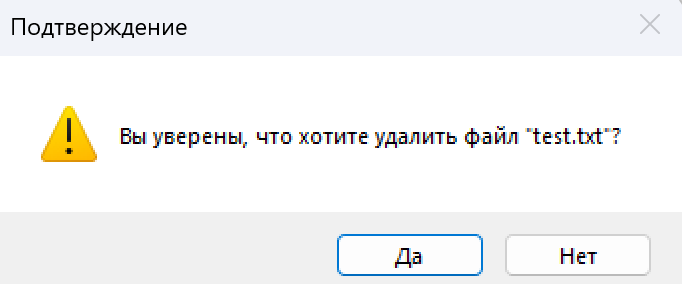


Рисунок 5.1.11 – Подтверждение удаления файла

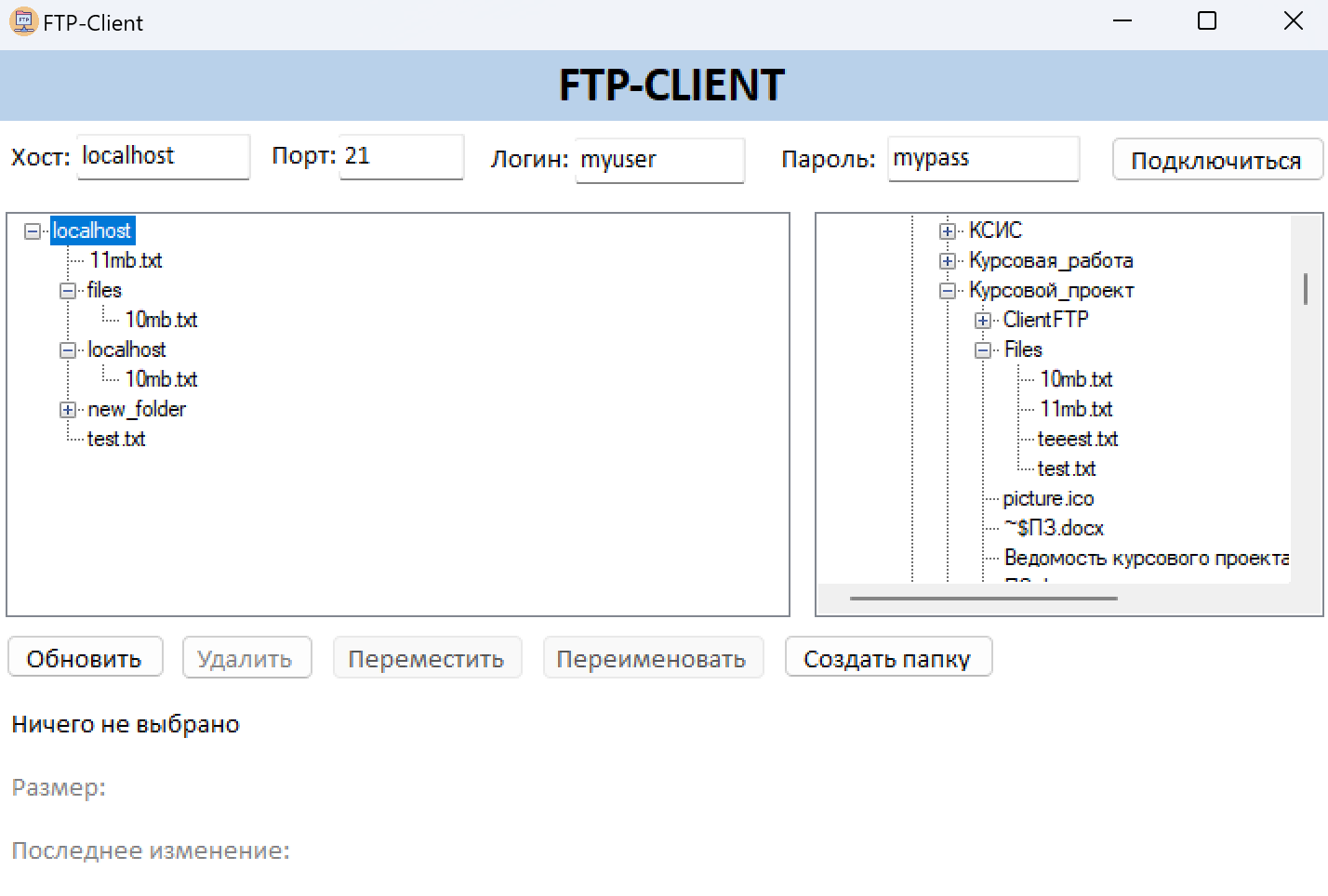
****

Рисунок 5.1.12 – Результат удаления файла

Для перемещения выбранного файла необходимо выбрать файл, нажать на кнопку "Переместить" и выбрать папку для перемещения. Результат перемещения файла localhost\test.txt в localhost\files\test.txt показан на рисунке 5.1.13.

Рисунок 5.1.13 – Перемещение файла

Для того, чтобы переименовать выбранный файл/папку необходимо выбрать элемент и нажать на кнопку «Переименовать». Появится поле для ввода нового имени файла/папки, представленное на рисунке 5.1.14. На рисунке 5.1.15 представлен результат изменения имени файла 11mb.txt на newname.txt.

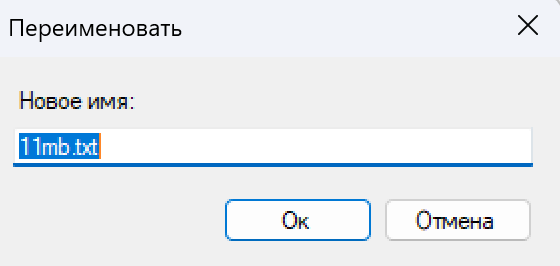


Рисунок 5.1.14 – Ввод нового имени для файла

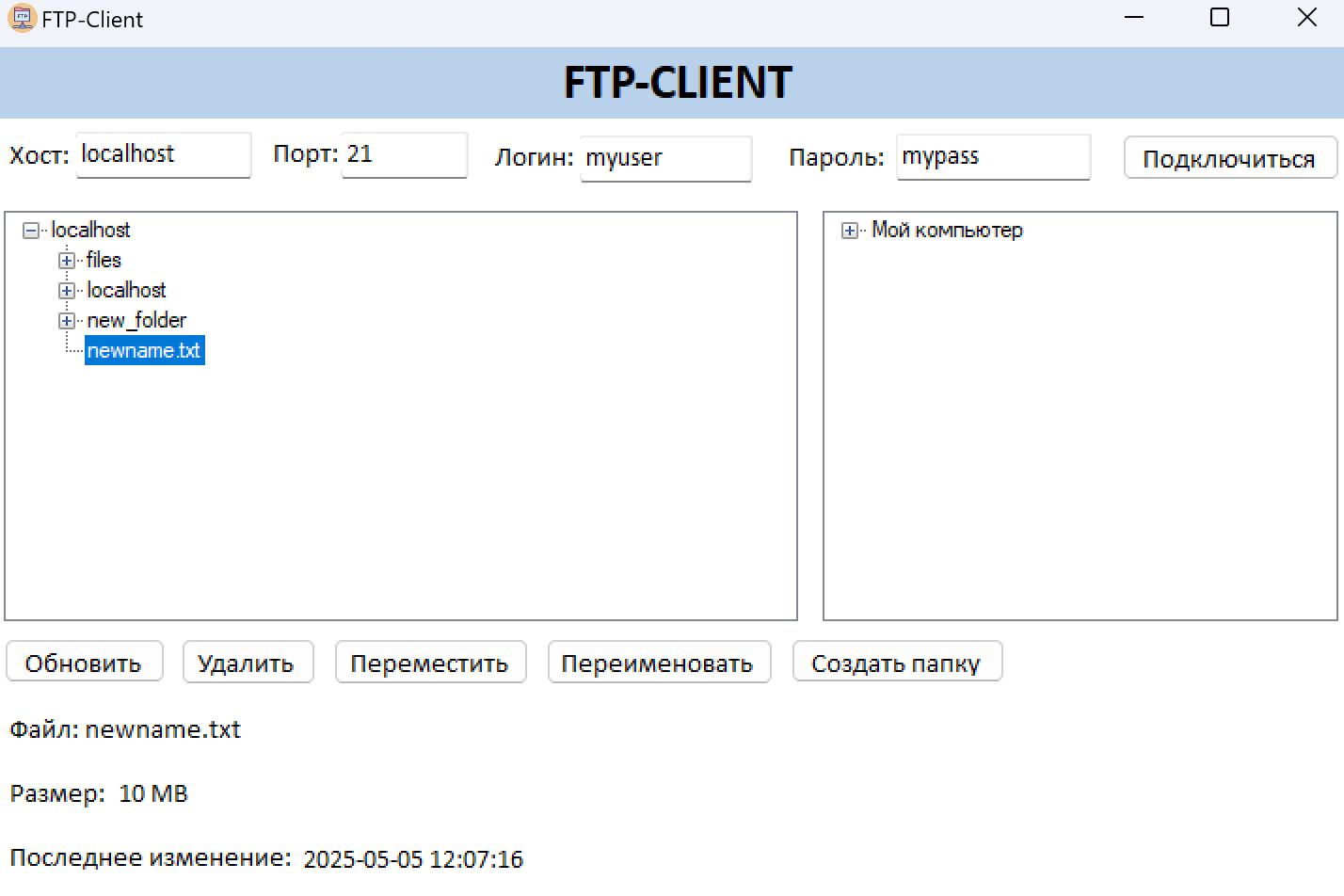


Рисунок 5.1.15 – Успешное изменение имени файла

**ЗАКЛЮЧЕНИЕ**

В ходе работы была разработана программа FTP-клиент с графическим интерфейсом, позволяющая удобно управлять файлами на FTP-сервере и локальной файловой системе.

Основные достижения программы включают в себя:

* подключение к FTP-серверу с аутентификацией;
* просмотр иерархии файлов и папок на сервере и локальном компьютере в виде дерева (TreeView);
* загрузка и скачивание файлов;
* создание, переименование и удаление файлов и папок;
* перемещение файлов между папками на сервере.

Программа имеет удобный интерфейс с интуитивно понятным отображением файловой структуры, поддержкой операций перетаскивания (Drag-and-drop). Реализованы асинхронные методы для избежания зависаний.

Программа представляет собой полноценный FTP-клиент с удобным графическим интерфейсом, отвечающий требованиям курсового проекта. Она демонстрирует применение современных технологий .NET (FTP-клиент, асинхронность, WinForms) и может быть доработана для использования в реальных условиях.

**ПРИЛОЖЕНИЕ А**

**Файл Connection.cs**

**Установка соединения с сервером**

private void btnConnect\_Click(object sender, EventArgs e)

{

if (!IPAddress.TryParse(txtIp.Text, out \_) && Uri.CheckHostName(txtIp.Text) == UriHostNameType.Unknown)

{

MessageBox.Show("Введите корректный IP-адрес или доменное имя!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

if (txtPort.Text != "")

{

if (!int.TryParse(txtPort.Text, out int port) || port < 1 || port > 65535)

{

MessageBox.Show("Введите корректный порт!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

}

try

{

string ip = txtIp.Text;

string port = txtPort.Text;

string userName = txtLogin.Text;

string password = txtPassword.Text;

string ftpUrl = $"ftp://{ip}:{port}/";

//создание запроса

FtpWebRequest request = (FtpWebRequest)WebRequest.Create(ftpUrl);

//запрос на получение списка файлов

request.Method = WebRequestMethods.Ftp.ListDirectory;

//указание логина и пароля, если требуется авторизация

request.Credentials = new NetworkCredential(userName, password);

//сервер сам открывает порт

request.UsePassive = true;

//передача файлов в бинарном виде

request.UseBinary = true;

//после выполнения команды соединение закрывается

request.KeepAlive = false;

//получение ответа от сервера

using (FtpWebResponse response = (FtpWebResponse)request.GetResponse())

{

MessageBox.Show("Успешное подключение!", "Успех", MessageBoxButtons.OK, MessageBoxIcon.Information);

serverInfo = new FtpServerInfo

{

Ip = txtIp.Text,

Port = txtPort.Text,

UserName = string.IsNullOrEmpty(txtLogin.Text) ? "anonymous" : txtLogin.Text,

Password = txtPassword.Text

};

LoadFtpInformation();

}

}

catch (WebException ex)

{

MessageBox.Show($"Ошибка подключения: {ex.Message}.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

catch (Exception ex)

{

MessageBox.Show($"Неизвестная ошибка: {ex.Message}.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

**Загрузка списка содержимого сервера**

private async void LoadFtpInformation()

{

trvServer.Nodes.Clear();

var rootNode = new TreeNode(serverInfo.Ip);

trvServer.Nodes.Add(rootNode);

await LoadFilesRecursive("/", rootNode);

}

private async Task LoadFilesRecursive(string currentPath, TreeNode parentNode)

{

string ftpUrl = $"ftp://{serverInfo.Ip}:{serverInfo.Port}{currentPath}";

try

{

var request = (FtpWebRequest)WebRequest.Create(ftpUrl);

request.Method = WebRequestMethods.Ftp.ListDirectoryDetails;

request.Credentials = new NetworkCredential(serverInfo.UserName, serverInfo.Password);

request.UsePassive = true;

request.UseBinary = true;

request.KeepAlive = false;

using (var response = (FtpWebResponse)await request.GetResponseAsync())

using (var stream = response.GetResponseStream())

using (var reader = new StreamReader(stream))

{

while (!reader.EndOfStream)

{

string line = await reader.ReadLineAsync();

string name = ExtractName(line);

bool isDirectory = IsDirectory(line);

var node = new TreeNode(name)

{

Tag = isDirectory

};

parentNode.Nodes.Add(node);

if (isDirectory)

{

//рекурсивный обход подкатологов

string nextPath = currentPath.EndsWith("/")

? currentPath + name

: currentPath + "/" + name;

await LoadFilesRecursive(nextPath + "/", node);

}

**}**

}

}

catch (Exception ex)

{

Console.WriteLine($"Error when receiving the contents of the directory: {ex.Message}", "Ошибка");

}

}

**Загрузка содержимого локального компьютера**

private void LoadLocalInformation()

{

trvLocal.Nodes.Clear();

TreeNode rootNode = new TreeNode("Мой компьютер");

trvLocal.Nodes.Add(rootNode);

DriveInfo[] drives = DriveInfo.GetDrives();

foreach (DriveInfo drive in drives)

{

TreeNode driveNode = new TreeNode(drive.Name)

{

Tag = drive.RootDirectory

};

driveNode.Nodes.Add("\*dummy\*");

rootNode.Nodes.Add(driveNode);

}

}

private void trvLocal\_BeforeExpand(object sender, TreeViewCancelEventArgs e)

{

if (e.Node.Nodes.Count == 1 && e.Node.Nodes[0].Text == "\*dummy\*")

{

e.Node.Nodes.Clear();

if (e.Node.Tag is DirectoryInfo dir)

{

try

{

foreach (DirectoryInfo subDir in dir.GetDirectories())

{

TreeNode dirNode = new TreeNode(subDir.Name)

{

Tag = subDir,

ImageKey = "folder",

SelectedImageKey = "folder"

};

dirNode.Nodes.Add("\*dummy\*");

e.Node.Nodes.Add(dirNode);

}

foreach (FileInfo file in dir.GetFiles())

{

TreeNode fileNode = new TreeNode(file.Name)

{

Tag = file,

ImageKey = "file",

SelectedImageKey = "file"

};

e.Node.Nodes.Add(fileNode);

}

}

catch (UnauthorizedAccessException)

{

e.Node.Nodes.Add("Нет доступа");

}

}

}

}

**Загрузка файла на сервер**

private async Task UploadFileToFtp(string localPath, string remotePath)

{

try

{

string fileName = Path.GetFileName(localPath);

string ftpUrl = $"ftp://{serverInfo.Ip}:{serverInfo.Port}/{remotePath}/{fileName}";

var request = (FtpWebRequest)WebRequest.Create(ftpUrl);

request.Method = WebRequestMethods.Ftp.UploadFile;

request.Credentials = new NetworkCredential(serverInfo.UserName, serverInfo.Password);

request.UsePassive = true;

request.UseBinary = true;

request.KeepAlive = false;

using (FileStream fileStream = File.OpenRead(localPath))

using (Stream requestStream = await request.GetRequestStreamAsync())

{

await fileStream.CopyToAsync(requestStream);

}

MessageBox.Show("Файл успешно загружен на сервер", "Успех");

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка загрузки файла: {ex.Message}", "Ошибка");

}

}

**Скачивание файла с сервера**

private async Task DownloadFileFromFtp(string remotePath, string localPath)

{

try

{

string fileName = Path.GetFileName(remotePath);

string fullLocalPath = Path.Combine(localPath, fileName);

string ftpUrl = $"ftp://{serverInfo.Ip}:{serverInfo.Port}/{remotePath}";

var request = (FtpWebRequest)WebRequest.Create(ftpUrl);

request.Method = WebRequestMethods.Ftp.DownloadFile;

request.Credentials = new NetworkCredential(serverInfo.UserName, serverInfo.Password);

request.UsePassive = true;

request.UseBinary = true;

request.KeepAlive = false;

using (var response = (FtpWebResponse)await request.GetResponseAsync())

using (var responseStream = response.GetResponseStream())

using (var fileStream = File.Create(fullLocalPath))

{

await responseStream.CopyToAsync(fileStream);

}

MessageBox.Show("Файл успешно скачан", "Успех");

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка скачивания файла: {ex.Message}", "Ошибка");

}

}

**Создание новой папки на сервере**

private async void btnAddFolder\_Click(object sender, EventArgs e)

{

if (selectedNode == null)

{

MessageBox.Show("Выберите папку на сервере, куда хотите добавить новую папку.", "Выбор");

return;

}

if (selectedNode?.Tag is bool isDirectory)

{

if (!isDirectory)

{

MessageBox.Show("Выберите папку на сервере, куда хотите добавить новую папку.", "Выбор");

return;

}

}

Form inputForm = new Form()

{

Width = 300,

Height = 150,

Text = "Новая папка",

FormBorderStyle = FormBorderStyle.FixedDialog,

StartPosition = FormStartPosition.CenterParent,

MaximizeBox = false,

MinimizeBox = false

};

Label label = new Label() { Left = 10, Top = 15, Text = "Имя папки:", AutoSize = true };

TextBox textBox = new TextBox() { Left = 10, Top = 35, Width = 260 };

Button okButton = new Button() { Text = "Создать", Left = 115, Width = 75, Top = 70, DialogResult = DialogResult.OK };

Button cancelButton = new Button() { Text = "Назад", Left = 195, Width = 75, Top = 70, DialogResult = DialogResult.Cancel };

inputForm.Controls.AddRange(new Control[] { label, textBox, okButton, cancelButton });

inputForm.AcceptButton = okButton;

inputForm.CancelButton = cancelButton;

if (inputForm.ShowDialog() == DialogResult.OK)

{

string folderName = textBox.Text.Trim();

if (string.IsNullOrWhiteSpace(folderName))

{

MessageBox.Show("Имя папки не может быть пустым.", "Предупреждение");

return;

}

try

{

string remotePath = GetFullRemotePath(selectedNode);

string newFolderPath = string.IsNullOrEmpty(remotePath)

? folderName

: $"{remotePath}/{folderName}";

await CreateRemoteDirectoryRecursive(newFolderPath);

MessageBox.Show($"Папка \"{folderName}\" успешно создана.", "Успех");

LoadFtpInformation();

}

catch (WebException ex)

{

if (ex.Response is FtpWebResponse ftpResp)

{

MessageBox.Show($"FTP ошибка: {ftpResp.StatusDescription.Trim()}", "Ошибка");

}

else

{

MessageBox.Show("Ошибка создания папки: " + ex.Message, "Ошибка");

}

}

catch (Exception ex)

{

MessageBox.Show("Ошибка: " + ex.Message, "Ошибка");

}

}

}

**Выбор элемента на дереве сервера**

private async void treeView1\_AfterSelect(object sender, TreeViewEventArgs e)

{

// Если мы в режиме перемещения и щёлкнули по папке

if (isMoveMode && e.Node.Tag is bool isDir && isDir)

{

string targetDir = e.Node.FullPath.Replace("\\", "/").TrimStart('/');

await MoveFileAsync(fileToMove, targetDir);

btnDelete.Enabled = false;

btnRename.Enabled = false;

btnMove.Enabled = false;

lblFile.Text = "Ничего не выбрано.";

lblSizet.Text = "";

lblModifiedt.Text = "";

// Сброс режима перемещения и обновление дерева

isMoveMode = false;

fileToMove = null;

LoadFtpInformation();

return;

}

selectedNode = e.Node;

if (selectedNode?.Tag is bool isDirectory)

{

if (!isDirectory)

{

string filePath = selectedNode.Text;

fileInfo = await GetFileInformation(filePath);

var elements = new Control[] { lblSize, lblModified, btnDelete, btnMove, btnRename };

foreach (var el in elements)

{

el.ForeColor = System.Drawing.Color.Black;

}

btnDelete.Enabled = true;

btnRename.Enabled = true;

btnMove.Enabled = true;

lblFile.Text = "Файл: " + fileInfo.FileName;

lblSizet.Text = fileInfo.FileSize.ToString();

lblModifiedt.Text = fileInfo.LastModified.ToString();

}

else

{

var dirInfo = await GetDirectoryInformation(GetFullRemotePath(selectedNode));

btnDelete.Enabled = true;

btnRename.Enabled = true; // Отключаем переименование для папок (можно включить если нужно)

btnMove.Enabled = false;

var elements = new Control[] { lblSize, lblModified, btnDelete, btnMove, btnRename };

foreach (var el in elements)

{

el.ForeColor = System.Drawing.Color.Black;

}

lblFile.Text = "Папка: " + selectedNode.Text;

lblSizet.Text = dirInfo.FileSize;

lblModifiedt.Text = dirInfo.LastModified;

}

}

else

{

btnDelete.Enabled = true;

}

}

**Получение информации о файле**

private async Task<FileInformation> GetFileInformation(string filePath)

{

string ftpUrl = $"ftp://{serverInfo.Ip}:{serverInfo.Port}/{filePath}";

string sizeFormatted = "N/A", lastModified = "N/A";

try

{

//асинхронный запроc на получение размера файла

var sizeRequest = (FtpWebRequest)WebRequest.Create(ftpUrl);

sizeRequest.Method = WebRequestMethods.Ftp.GetFileSize;

sizeRequest.Credentials = new NetworkCredential(serverInfo.UserName, serverInfo.Password);

sizeRequest.UsePassive = true;

sizeRequest.UseBinary = true;

sizeRequest.KeepAlive = false;

using (var sizeResponse = (FtpWebResponse)await sizeRequest.GetResponseAsync())

{

long size = sizeResponse.ContentLength;

sizeFormatted = FormatFileSize(size);

}

//асинхронный запрос на получение даты последнего изменения файла

var dateRequest = (FtpWebRequest)WebRequest.Create(ftpUrl);

dateRequest.Method = WebRequestMethods.Ftp.GetDateTimestamp;

dateRequest.Credentials = new NetworkCredential(serverInfo.UserName, serverInfo.Password);

dateRequest.UsePassive = true;

dateRequest.UseBinary = true;

dateRequest.KeepAlive = false;

using (var dateResponse = (FtpWebResponse)await dateRequest.GetResponseAsync())

{

lastModified = dateResponse.LastModified.ToString("yyyy-MM-dd HH:mm:ss");

}

}

catch (Exception ex)

{

Console.WriteLine($"Ошибка при загрузке данных файла: {ex.Message}", "Ошибка");

}

return new FileInformation

{

FileName = filePath,

FileSize = sizeFormatted,

LastModified = lastModified,

DirectoryPath = Path.GetDirectoryName(filePath).Replace("\\", "/")

};

}

**Получение информации о папке**

private async Task<FileInformation> GetDirectoryInformation(string path)

{

try

{

string ftpUrl = $"ftp://{serverInfo.Ip}:{serverInfo.Port}/{path}";

string lastModified = "N/A";

long size = 0;

// Получаем дату последнего изменения

try

{

var dateRequest = (FtpWebRequest)WebRequest.Create(ftpUrl);

dateRequest.Method = WebRequestMethods.Ftp.GetDateTimestamp;

dateRequest.Credentials = new NetworkCredential(serverInfo.UserName, serverInfo.Password);

using (var response = (FtpWebResponse)await dateRequest.GetResponseAsync())

{

lastModified = response.LastModified.ToString("yyyy-MM-dd HH:mm:ss");

}

}

catch

{

// Если не удалось получить дату, оставляем "N/A"

}

// Пытаемся получить размер (может быть долго для больших папок)

try

{

size = await CalculateDirectorySize(path);

}

catch

{

// Если не удалось вычислить размер, оставляем 0

}

return new FileInformation

{

FileName = Path.GetFileName(path),

FileSize = size > 0 ? FormatFileSize(size) : "N/A",

LastModified = lastModified,

DirectoryPath = Path.GetDirectoryName(path)?.Replace("\\", "/") ?? "/"

};

}

catch (Exception ex)

{

Console.WriteLine($"Ошибка при получении информации о папке: {ex.Message}");

return new FileInformation

{

FileName = Path.GetFileName(path),

FileSize = "N/A",

LastModified = "N/A",

DirectoryPath = Path.GetDirectoryName(path)?.Replace("\\", "/") ?? "/"

};

}

}

**Удаление файла или папки**

private async void btnDelete\_Click(object sender, EventArgs e)

{

selectedNode = trvServer?.SelectedNode;

// Если ничего не выбрано или нет подключения к серверу - выходим

if (selectedNode == null || serverInfo == null)

return;

if (selectedNode.Tag is bool isDirectory)

{

string itemName = selectedNode.Text;

string itemType = isDirectory ? "папку" : "файл";

string path = GetFullRemotePath(selectedNode);

var confirm = MessageBox.Show(

$"Вы уверены, что хотите удалить {itemType} \"{itemName}\"?",

"Подтверждение",

MessageBoxButtons.YesNo,

MessageBoxIcon.Warning);

if (confirm != DialogResult.Yes)

return;

try

{

if (isDirectory)

{

await DeleteDirectoryRecursive(path);

}

else

{

await DeleteFile(path);

}

MessageBox.Show(isDirectory ? "Папка успешно удалена." : "Файл успешно удален.", "Успех");

LoadFtpInformation();

fileInfo = null;

lblFile.Text = "Ничего не выбрано";

lblSizet.Text = "";

lblModifiedt.Text = "";

btnDelete.Enabled = false;

btnRename.Enabled = false;

btnMove.Enabled = false;

var elements = new Control[] { lblSize, lblModified, btnDelete, btnMove, btnRename };

foreach (var el in elements)

{

el.ForeColor = System.Drawing.Color.Gray;

}

}

catch (WebException ex) when (ex.Response is FtpWebResponse ftpResp)

{

if (isDirectory && ftpResp.StatusCode == FtpStatusCode.ActionNotTakenFileUnavailable)

{

MessageBox.Show("Не удалось удалить папку. Убедитесь, что она пуста.", "Ошибка");

}

else

{

MessageBox.Show($"Ошибка FTP: {ftpResp.StatusDescription.Trim()}", "Ошибка");

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка удаления: {ex.Message}", "Ошибка");

}

}

else

{

MessageBox.Show("Выберите файл или папку на FTP-сервере для удаления.", "Информация");

}

}

private async Task DeleteDirectoryRecursive(string path)

{

try

{

string ftpUrl = $"ftp://{serverInfo.Ip}:{serverInfo.Port}/{path}";

var listRequest = (FtpWebRequest)WebRequest.Create(ftpUrl);

listRequest.Method = WebRequestMethods.Ftp.ListDirectoryDetails;

listRequest.Credentials = new NetworkCredential(serverInfo.UserName, serverInfo.Password);

using (var response = (FtpWebResponse)await listRequest.GetResponseAsync())

using (var stream = response.GetResponseStream())

using (var reader = new StreamReader(stream))

{

while (!reader.EndOfStream)

{

string line = await reader.ReadLineAsync();

string name = ExtractName(line);

bool isDir = IsDirectory(line);

if (name == "." || name == "..") continue;

string itemPath = $"{path}/{name}";

if (isDir)

{

await DeleteDirectoryRecursive(itemPath);

}

else

{

await DeleteFile(itemPath);

}

}

}

var deleteRequest = (FtpWebRequest)WebRequest.Create(ftpUrl);

deleteRequest.Method = WebRequestMethods.Ftp.RemoveDirectory;

deleteRequest.Credentials = new NetworkCredential(serverInfo.UserName, serverInfo.Password);

using (var deleteResponse = (FtpWebResponse)await deleteRequest.GetResponseAsync())

{

}

}

catch (Exception ex)

{

throw new Exception($"Не удалось удалить папку {path}: {ex.Message}");

}

}

private async Task DeleteFile(string path)

{

string ftpUrl = $"ftp://{serverInfo.Ip}:{serverInfo.Port}/{path}";

var request = (FtpWebRequest)WebRequest.Create(ftpUrl);

request.Method = WebRequestMethods.Ftp.DeleteFile;

request.Credentials = new NetworkCredential(serverInfo.UserName, serverInfo.Password);

using (var response = (FtpWebResponse)await request.GetResponseAsync())

{

}

}

**Изменение имени файла или папки**

private async void btnRename\_Click(object sender, EventArgs e)

{

selectedNode = trvServer?.SelectedNode;

// Если ничего не выбрано

if (selectedNode == null)

{

MessageBox.Show("Выберите файл или папку для переименования.", "Предупреждение");

return;

}

string currentName = selectedNode.Text;

string currentPath = GetFullRemotePath(selectedNode);

bool isDirectory = selectedNode.Tag is bool dirFlag && dirFlag;

Form renameForm = new Form()

{

Width = 300,

Height = 150,

Text = "Переименовать",

FormBorderStyle = FormBorderStyle.FixedDialog,

StartPosition = FormStartPosition.CenterParent,

MaximizeBox = false,

MinimizeBox = false

};

Label label = new Label() { Left = 10, Top = 15, Text = "Новое имя:", AutoSize = true };

TextBox textBox = new TextBox() { Left = 10, Top = 35, Width = 260, Text = currentName };

Button okButton = new Button() { Text = "Oк", Left = 115, Width = 75, Top = 70, DialogResult = DialogResult.OK };

Button cancelButton = new Button() { Text = "Отмена", Left = 195, Width = 75, Top = 70, DialogResult = DialogResult.Cancel };

renameForm.Controls.AddRange(new Control[] { label, textBox, okButton, cancelButton });

renameForm.AcceptButton = okButton;

renameForm.CancelButton = cancelButton;

if (renameForm.ShowDialog() == DialogResult.OK)

{

string newName = textBox.Text.Trim();

if (string.IsNullOrWhiteSpace(newName))

{

MessageBox.Show("Имя не может быть пустым.", "Ошибка");

return;

}

try

{

string newPath = Path.GetDirectoryName(currentPath) + "/" + newName;

newPath = newPath.Replace("\\", "/").TrimStart('/');

string ftpUrl = $"ftp://{serverInfo.Ip}:{serverInfo.Port}/{currentPath}";

var request = (FtpWebRequest)WebRequest.Create(ftpUrl);

request.Method = WebRequestMethods.Ftp.Rename;

request.Credentials = new NetworkCredential(serverInfo.UserName, serverInfo.Password);

request.UseBinary = true;

request.UsePassive = true;

request.KeepAlive = false;

request.RenameTo = newName;

using (FtpWebResponse response = (FtpWebResponse)await request.GetResponseAsync())

{

MessageBox.Show($"{(isDirectory ? "Папка" : "Файл")} успешно переименован на {newName}.", "Успех");

}

// Обновляем информацию

LoadFtpInformation();

// Обновляем информацию о выбранном элементе

if (isDirectory)

{

var dirInfo = await GetDirectoryInformation(newPath);

lblFile.Text = "Папка: " + newName;

lblSizet.Text = dirInfo.FileSize;

lblModifiedt.Text = dirInfo.LastModified;

}

else

{

fileInfo = await GetFileInformation(newPath);

lblFile.Text = "Файл: " + newName;

lblSizet.Text = fileInfo.FileSize;

lblModifiedt.Text = fileInfo.LastModified;

}

}

catch (WebException ex) when (ex.Response is FtpWebResponse ftpResp)

{

MessageBox.Show($"FTP ошибка: {ftpResp.StatusDescription.Trim()}", "Ошибка");

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка переименования: {ex.Message}", "Ошибка");

}

}

}

**Перемещение файла**

private void btnMove\_Click(object sender, EventArgs e)

{

if (fileInfo == null)

{

MessageBox.Show("Выберите файл для перемещения.", "Выбор");

return;

}

fileToMove = fileInfo;

isMoveMode = true;

MessageBox.Show(

"Теперь выберите папку на сервере, куда хотите переместить файл.",

"Перемещение",

MessageBoxButtons.OK,

MessageBoxIcon.Information

);

}

private async Task MoveFileAsync(FileInformation srcFile, string remoteTargetDir)

{

try

{

string oldPath = $"{srcFile.DirectoryPath.Trim('/')}/{srcFile.FileName}";

string newPath = $"{remoteTargetDir.Trim('/')}/{srcFile.FileName}";

oldPath = oldPath.Replace("localhost/", "").Trim('/');

newPath = newPath.Replace("localhost/", "").Trim('/');

string ftpUrl = $"ftp://{serverInfo.Ip}:{serverInfo.Port}/{oldPath}";

var moveReq = (FtpWebRequest)WebRequest.Create(ftpUrl);

moveReq.Method = WebRequestMethods.Ftp.Rename;

moveReq.Credentials = new NetworkCredential(serverInfo.UserName, serverInfo.Password);

moveReq.UsePassive = true;

moveReq.UseBinary = true;

moveReq.KeepAlive = false;

// указываем относительный или абсолютный путь назначения

moveReq.RenameTo = newPath.StartsWith("/") ? newPath.Substring(1) : newPath;

using (var resp = (FtpWebResponse)await moveReq.GetResponseAsync())

{

MessageBox.Show($"Файл перемещен в /{newPath}", "Успех");

}

}

catch (WebException ex) when (ex.Response is FtpWebResponse ftpResp)

{

MessageBox.Show($"FTP ошибка: {ftpResp.StatusDescription.Trim()}", "Ошибка");

}

catch (Exception ex)

{

MessageBox.Show("Ошибка перемещения файла: " + ex.Message, "Ошибка");

}

}