

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 (КСиС)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к курсовой работе
на тему

FTP-КЛИЕНТ НА ПЛАТФОРМЕ .NET

БГУИР КР 1-40 01 01 619 ПЗ

Студент гр. 851006
Руководитель

Мискевич П.Л.
Шимко И.В.

Минск 2020

Учреждение образования

«Белорусский государственный университет информатики и
радиоэлектроники»

Факультет компьютерных систем и сетей

УТВЕРЖДАЮ

Заведующий кафедрой ПОИТ
Лапицкая Н.В.

(подпись)

2020 г.

ЗАДАНИЕ

по курсовому проектированию

Студенту Мискевичу Павлу Леонидовичу, 851006

1. Тема работы FTP-клиент на платформе .NET
2. Срок сдачи студентом законченной работы 05.06.2020 г.
3. Исходные данные к работе Язык программирования C#
4. Содержание расчётно-пояснительной записки (перечень вопросов, которые подлежат разработке)

Введение

1. Анализ прототипов, литературных источников и формирование требований к проектируемому программному средству;
2. Моделирование предметной области и разработка функциональных требований;
3. Проектирование программного средства;
4. Тестирование, проверка работоспособности и анализ полученных результатов;
5. Руководство по установке и использованию;

Заключение

Список используемой литературы

5. Перечень графического материала (с точным обозначением обязательных чертежей и графиков)

1. Программное средство "FTP-клиент на платформе .NET", A1, схема программы, чертеж.

6. Консультант по курсовой работе

Шимко И.В.

7. Дата выдачи задания 10.02.2020

8. Календарный график работы над проектом на весь период проектирования (с обозначением сроков выполнения и процентом от общего объема работы):

раздел 1, 2 к 24.02.2020 – 15 % готовности работы;

разделы 3, 4 к 16.03.2020 – 30 % готовности работы;

разделы 5, 6 к 06.04.2020 – 60 % готовности работы;

раздел 7, 8, 9 к 04.05.2020 – 90 % готовности работы;

оформление пояснительной записки и графического материала к 05.06.2020 – 100 % готовности работы.

РУКОВОДИТЕЛЬ _____ Шимко И.В.
(подпись)

Задание принял к исполнению Мискевич П.Л. _____
(дата и подпись студента)

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	5
1 Анализ прототипов, литературных источников и формирование требований к проектируемому программному средству	6
1.1 Анализ существующих аналогов	6
1.2 Постановка задачи	8
2 Анализ требований к программному средству и разработка функциональных требований.....	10
2.1 Основные сведения об используемом протоколе	10
3 Проектирование программного средства	14
3.1 Интерфейс программного средства	14
3.2 Проектирование функционала	15
3.3 Проектирование установщика.....	17
4 Тестирование, проверка работоспособности и анализ полученных результатов.....	18
4.1 Тестирование функционала программы	18
4.2 Вывод из прохождения тестирования	21
5 Руководство по установке и использованию	22
Заключение	26
Список литературы	27
Приложение А	28
Приложение Б	29

ВВЕДЕНИЕ

В настоящее время компьютерные сети получили очень широкое распространение. Компьютерные сети занимают особое место в нашей повседневной жизни, в нашей производственной деятельности и в других областях. Когда интернет только зарождался, но уже были компьютерные сети, возникла потребность передавать файлы от одного компьютера к другому. В 1971 году каналы передачи данных были не такие надёжные и не такие быстрые, как сейчас, поэтому нужен был инструмент, который поможет обмениваться документами друг с другом на расстоянии. Основные требования были такие: простота работы и надёжность при отправке и получении. Таким инструментом стал FTP-протокол.

Для работы по FTP нужны FTP-сервер и FTP-клиент.

FTP-сервер обеспечивает доступ к нужным файлам, предоставляет пользователю файлы и папки, следит за качеством передачи, смотрит, чтобы не было ошибок и управляет параметрами соединения в пассивном режиме.

Чтобы подключиться к серверу, нужна специальная программа, её ещё называют FTP-клиентом.

Целью данного курсового проекта является разработка уникального FTP-клиента для соединения с FTP-сервером.

Данный FTP-клиент не претендует на звание самого полного по функциональности, а лишь отражает взгляд на то, каким должен быть простой, но вместе с тем функциональный FTP-клиент.

В этой пояснительной записке отображены следующие этапы написания курсовой работы:

- 1 Анализ прототипов, литературных источников и формирование требований к проектируемому программному средству.
- 2 Анализ требований к программному средству и разработка функциональных требований.
- 3 Проектирование программного средства.
- 4 Тестирование, проверка работоспособности и анализ полученных результатов.
- 5 Руководство по установке и использованию.

1 АНАЛИЗ ПРОТОТИПОВ, ЛИТЕРАТУРНЫХ ИСТОЧНИКОВ И ФОРМИРОВАНИЕ ТРЕБОВАНИЙ К ПРОЕКТИРУЕМОМУ ПРОГРАММНОМУ СРЕДСТВУ

1.1 Анализ существующих аналогов

На сегодняшний день существует многочисленное количество FTP-клиентов. При поиске и изучении аналогов было замечено, что пользователи часто пытаются найти программное средство, которое не только отвечает требованиям пользователя, но и имеет весьма приятный интерфейс.

1.1.1 FileZilla – очень популярный и простой FTP-клиент, где удобный интерфейс сочетается с многофункциональностью [1].

В интерфейсе FTP-клиента нет ничего лишнего, перед Вами только нужные панели. Верхнее окно используется для журнала сообщений, нижнее – для создания заданий. Файлы могут быть представлены в виде древовидного или локального списка, это позволяет легко управлять и распределять их между двумя колонками.

Подключиться на хост можно через панель с быстрым соединением, но доступны к просмотру и передающие FTP-протоколы. В программе есть специальные настройки, которые допускают изменение протокола (FTP/SFTP), ввод требуемых учетных данных, назначение локального и удаленного каталогов, отрывающихся в момент подключения, изменение способа файловой отсылки.

Клиент поддерживает синхронизацию просмотра, сравнивает каталоги, фильтрует, кэширует и занимается удалённым поиском стандартных, а также необходимых для навигации функций. У клиента не поддерживается командная строка.

Регулярное обновление программы лучшим образом свидетельствует об активном ее обслуживании и дальнейшей разработке.

Внешний вид данного аналога представлен на рисунке 1.1.

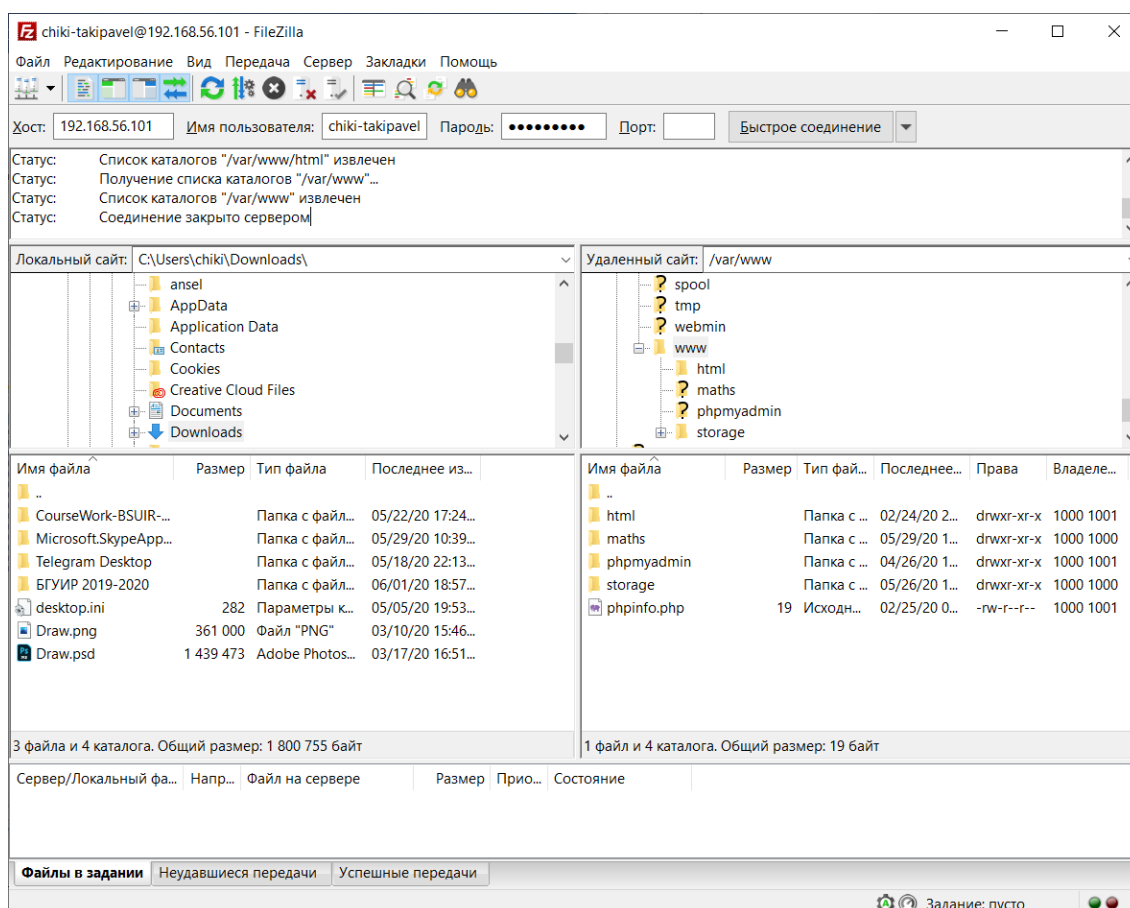


Рисунок 1.1 – Интерфейс FTP-клиента FileZilla

1.1.2 WinSCP – бесплатная FTP-программа, которая в отличие от FileZilla поддерживает командную строку [2].

У данного аналога есть открытый исходный код и поддержка протоколов FTP, SFTP, SCP и FTPS. FTP-клиент поддерживает сессии, вкладки, синхронизирует каталоги и выполняет ряд других функций. WinSCP работает с графикой, но программа создана не для этого. Она не привлекательна для малоопытных пользователей, так как большинство опций доступно лишь из командного режима.

Данная программа неплохо автоматизирована, имеет консольное управление и гибкие настройки, а вот русификация требует доработки.

Интерфейс FTP-программы представлен на рисунке 1.2.

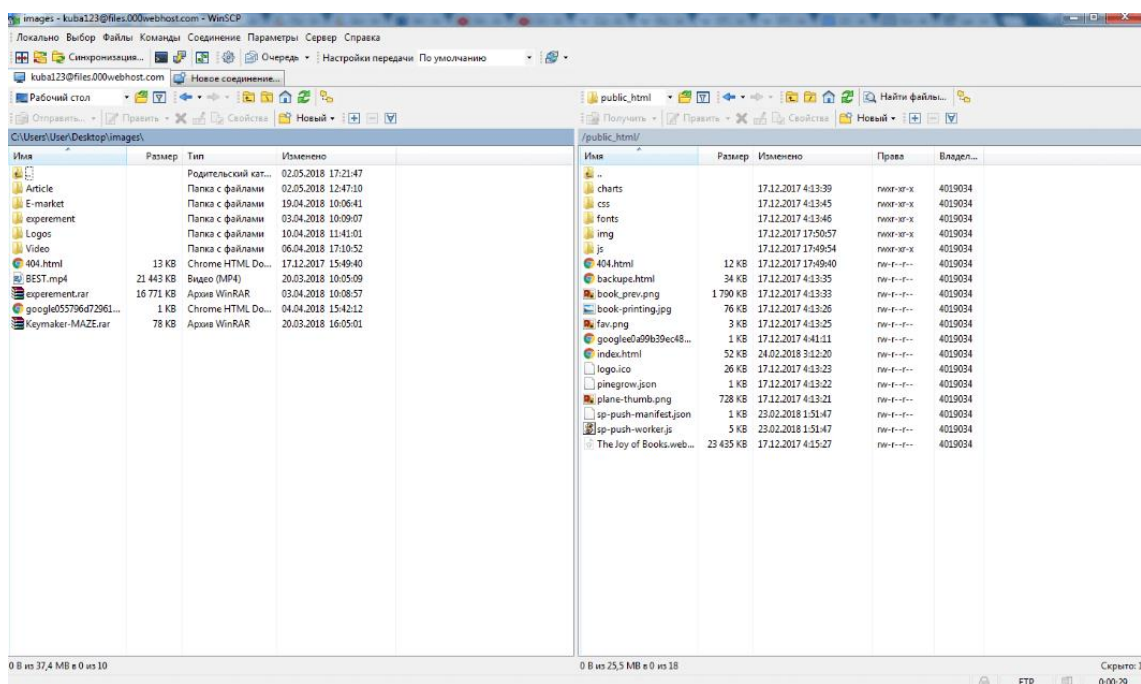


Рисунок 1.2 – Интерфейс FTP-клиента WinSCP

1.1.3 CuteFTP – условно бесплатный FTP-клиент для Microsoft Windows и Mac OS X [3].

Данный клиент поддерживает протоколы FTP, FTPS, HTTP, HTTPS и SSH. CuteFTP поддерживает автоматическую загрузку файлов в установленное время, надёжную и безопасную работу с несколькими удалёнными узлами в одно время, безопасное резервное копирование, многофункциональную настройку и работу с прокси, копирование файлов с одного FTP-сервера на другой, OpenPGP шифрование, одноразовый пароль аутентификации и менеджер паролей.

Внешний вид данного аналога представлен на рисунке 1.3.

1.2 Постановка задачи

В задачу курсового проекта входит создание оконного приложения FTP-клиент на языке C# с использованием технологии WPF [4] в интегрированной среде разработки Microsoft Visual Studio 2019.

Для того, чтобы программное средство можно было считать FTP-клиентом, на основе анализа популярных клиентов, в конечном приложении необходимо наличие следующих возможностей:

- доступ авторизованным клиентам к FTP-серверу;
- просмотр файлов и директорий;

- удаление файлов и директорий;
- загрузка файлов с сервера на компьютер пользователя;
- создание каталогов на сервере;
- загрузка файлов на сервер.

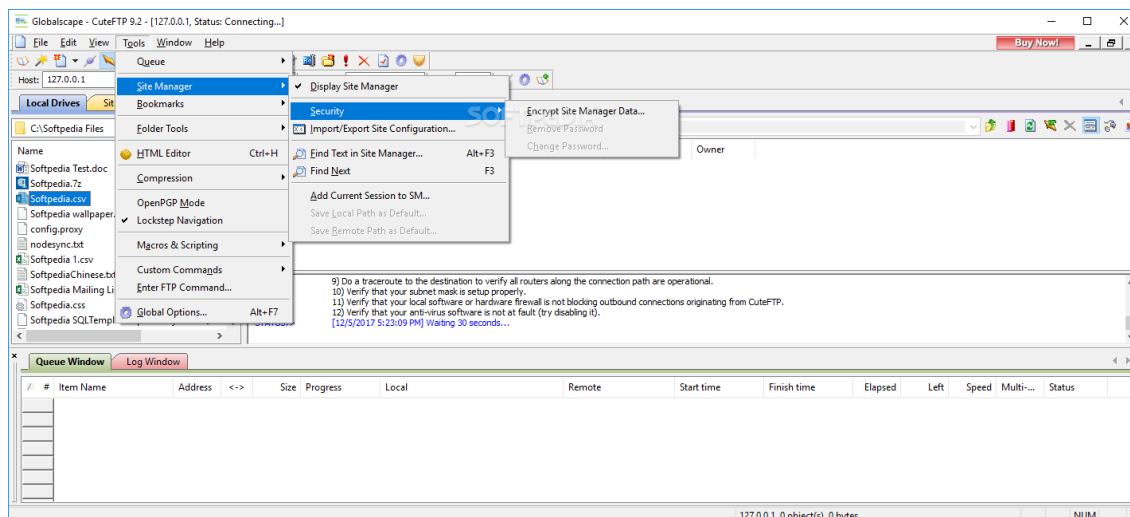


Рисунок 1.3 – Интерфейс FTP-клиента CuteFTP

2 МОДЕЛИРОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ И РАЗРАБОТКА ФУНКЦИОНАЛЬНЫХ ТРЕБОВАНИЙ

2.1 Основные сведения об используемом протоколе

2.1.1 Общие сведения и история протокола

FTP (RFC-959) [5] обеспечивает файловый обмен между удаленными пользователями. Протокол FTP формировался многие годы. Первые реализации в МТИ относятся к 1971. (RFC 114 и 141). RFC 172 рассматривает протокол, ориентированный на пользователя, и предназначенный для передачи файлов между ЭВМ. Позднее в документах RFC 265 и RFC 281 протокол был усовершенствован. Заметной переделке протокол подвергся в 1973, и окончательный вид он обрел в 1985 году. Таким образом, данный протокол является одним из старейших. В стеке TCP/IP протокол находится на прикладном уровне (см. рисунок 2.1).

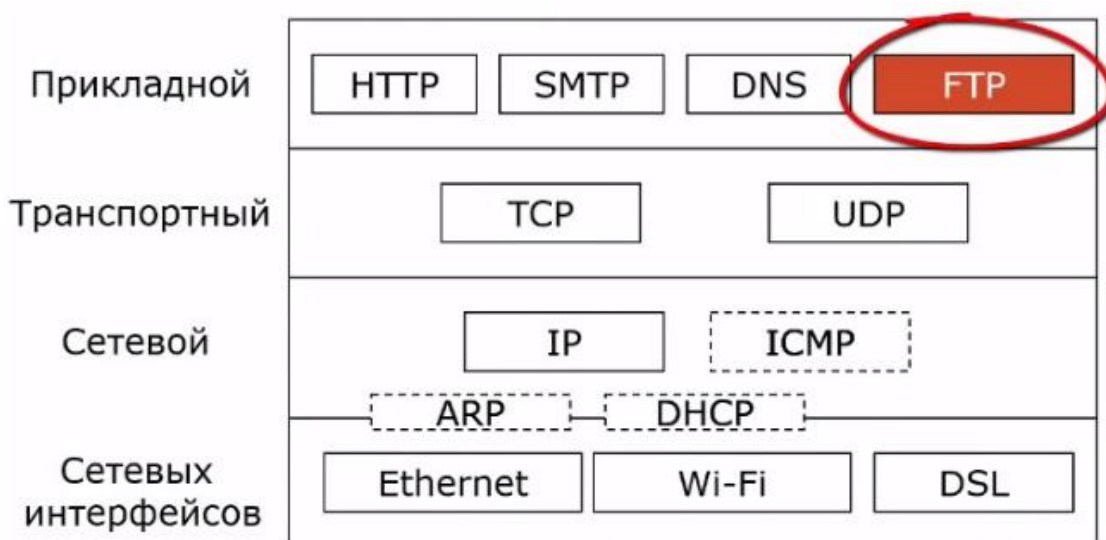


Рисунок 2.1 – Стек TCP/IP

Работа FTP на пользовательском уровне содержит несколько этапов:

- идентификация (ввод имени-идентификатора и пароля);
- выбор каталога;
- определение режима обмена (поблочный, поточный, ASCII или двоичный);
- выполнение команд обмена (get, mget, dir, mdel, mput или put);
- завершение процедуры (quit или close).

2.1.2 Принцип работы протокола FTP

Протокол FTP работает в режиме клиент-сервер. На сервере есть файловая система, это структура каталогов, в которой находятся файлы. Клиент по протоколу FTP подключается к серверу и может работать с файловой системой, просматривать каталоги, переходить между ними, загружать и записывать файлы сервера, перемещать их между разными каталогами и выполнять другие операции, которые можно делать с файловой системой [6].

В отличие от других протоколов прикладного уровня, протокол FTP использует два отдельных соединения (см. рисунок 2.2). Первое соединение управляющее, второе соединение для передачи данных.



Рисунок 2.2 – Соединения между сервером и клиентом

Использовать соединение данных можно тремя способами:

- отправка файлов от клиента к серверу;
- отправка файлов от сервера к клиенту;
- отправка списка файлов или директорий от сервера к клиенту.

2.1.3 Аутентификация в FTP

Протокол FTP требует, чтобы пользователь прошел аутентификацию. Для этого необходимо ввести идентификатор и пароль. В зависимости от идентификатора пользователя ему может быть предоставлено больше или меньше прав для доступа к файловой системе сервера.

Специальный тип пользователя FTP – анонимный пользователь. Для анонимного пользователя в качестве логина используется ftp, или anonymous. Как правило анонимный пользователь имеет ограниченные права, обычно он может только скачивать файлы и записывать файлы в один определенный каталог.

2.1.4 Команды протокола FTP

Протокол FTP, как и многие протоколы прикладного уровня работает в текстовом режиме. Большинство из команд состоят из 4-х символов [6].

Некоторые команды FTP приведены в таблице 2.1.

Таблица 2.1 – Команды протокола FTP

Команда	Описание
ABOR	Прервать предыдущую команду FTP и любую передачу данных.
LIST	Список файлов или директорий.
PASS	Пароль.
PORT	Войти в активный режим.
QUIT	Отключиться.
RETR	Скачать файл.
STOR	Загрузить файл.
DELE	Удалить файл.
RMD	Удалить каталог.
MKD	Создать каталог.

Команды и отклики передаются по управляющему соединению между клиентом и сервером в формате NVT ASCII. В конце каждой строки команды или отклика присутствует пара CR, LF. Команды состоят из 3 или 4 байт, а именно из заглавных ASCII символов, некоторые с необязательными аргументами. Клиент может отправить серверу более чем 30 различных FTP команд.

2.1.5 Коды ответов FTP

Отклики состоят из 3-цифрных значений в формате ASCII, и необязательных сообщений, которые следуют за числами. Подобное представление откликов объясняется тем, что программному обеспечению необходимо посмотреть только цифровые значения, чтобы понять, что ответил процесс, а дополнительную строку может прочитать человек. Поэтому пользователю достаточно просто прочитать сообщение (причем нет необходимости запоминать все цифровые коды откликов). Группы откликов представлены в таблице 2.2.

Таблица 2.2 – Коды ответов FTP

Код ответа	Описание
1yz	Положительный предварительный отклик. Действие началось, однако необходимо дождаться еще одного отклика перед отправкой следующей команды.
2yz	Положительный отклик о завершении. Может быть отправлена новая команда.
3yz	Положительный промежуточный отклик. Команда принята, однако необходимо отправить еще одну команду.
4yz	Временный отрицательный отклик о завершении. Требуемое действие не произошло, однако ошибка временная, поэтому команду необходимо повторить позже.
5yz	Постоянный отрицательный отклик о завершении. Команда не была воспринята и повторять ее не стоит.
x0z	Синтаксическая ошибка.
x1z	Информация.
x2z	Соединения. Отклики имеют отношение либо к управляющему, либо к соединению данных.
x3z	Аутентификация и бюджет. Отклик имеет отношение к логированию или командам, связанным с бюджетом.
x4z	Не определено.
x5z	Состояние файловой системы.

Третья цифра дает дополнительное объяснение сообщению об ошибке.

3 ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА

3.1 Интерфейс программного средства

Интерфейс обеспечивает комфортное взаимодействие между пользователем и приложением. При его разработке следует учитывать эргономику и устройства, с какими будет взаимодействовать приложение. Комфорт и удобство пользователя должны быть главным критерием в построении интерфейса и создании дизайна.

Поэтому для удобства был использован стиль Material Design с помощью подключения библиотеки Material Design In XAML Toolkit. Material Design – стиль графического дизайна интерфейсов программного обеспечения и приложений, разработанный компанией Google. Стиль расширяет идею «карточек», более широким применением строгих макетов, анимаций и переходов, отступов и эффектов глубины (света и тени). По идее графических дизайнеров, у приложений не должно быть острых углов, карточки должны переключаться между собой плавно и практически незаметно.

Пример интерфейса FTP-клиента продемонстрирован на рисунке 3.1.

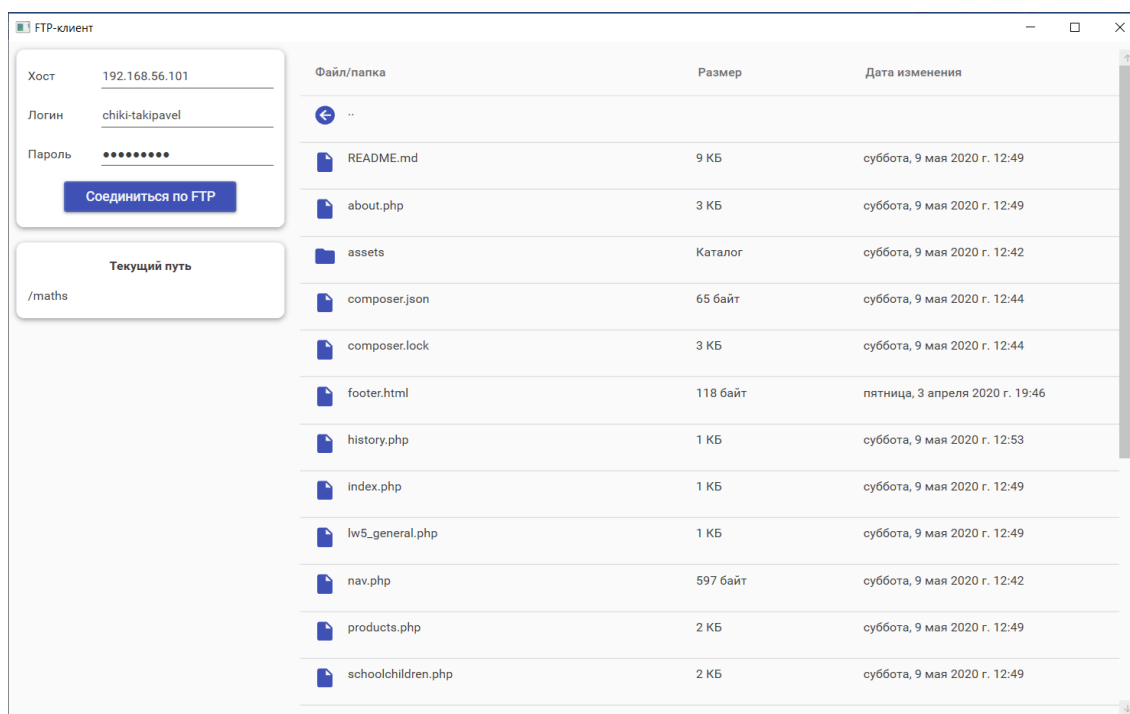


Рисунок 3.1 – Интерфейс FTP-клиента

Для комфортного использования программного средства пользователем ввод адреса, логина, пароля и кнопка подключения по FTP располагаются в левой части окна. А отображение файлов, директорий и информации о них располагаются в правой части окна.

Для наглядности отображения файлов, директорий и возвращения к предыдущей директории были использованы иконки из IconPack библиотеки Material Design In XAML Toolkit (см. рисунок 3.2).

Также реализована функция Drag-and-drop, позволяющая загрузить файлы на сервер с помощью перетаскивания файла с компьютера пользователя в программу мышью.



Рисунок 3.2 – Используемые иконки

3.2 Проектирование функционала

Диаграмма классов изображена на рисунке 3.3. Исходный код программы представлен в приложении Б.

3.2.1 Класс FtpClient

Функционал для взаимодействия с сервером был выделен в отдельный класс FtpClient. Для отправления запросов к FTP-серверу и получения ответов используются классы FtpWebRequest и FtpWebResponse.

FtpWebRequest позволяет отправить запрос к серверу. Для настройки запроса используются следующие его свойства:

- Credentials – задаёт аутентификационные данные пользователя;
- EnableSsl – указывает, надо ли использовать ssl-соединение;
- Method: задает команду протокола FTP, которая будет использоваться в запросе;
- UsePassive – устанавливает пассивный режим запроса к серверу
- UseBinary – указывает тип данных, которые будут использоваться в запросе.

Для вывода списка файлов и каталогов используется метод ListDirectory. В его основе лежит FTP-метод LIST. В качестве параметра принимается путь директории.

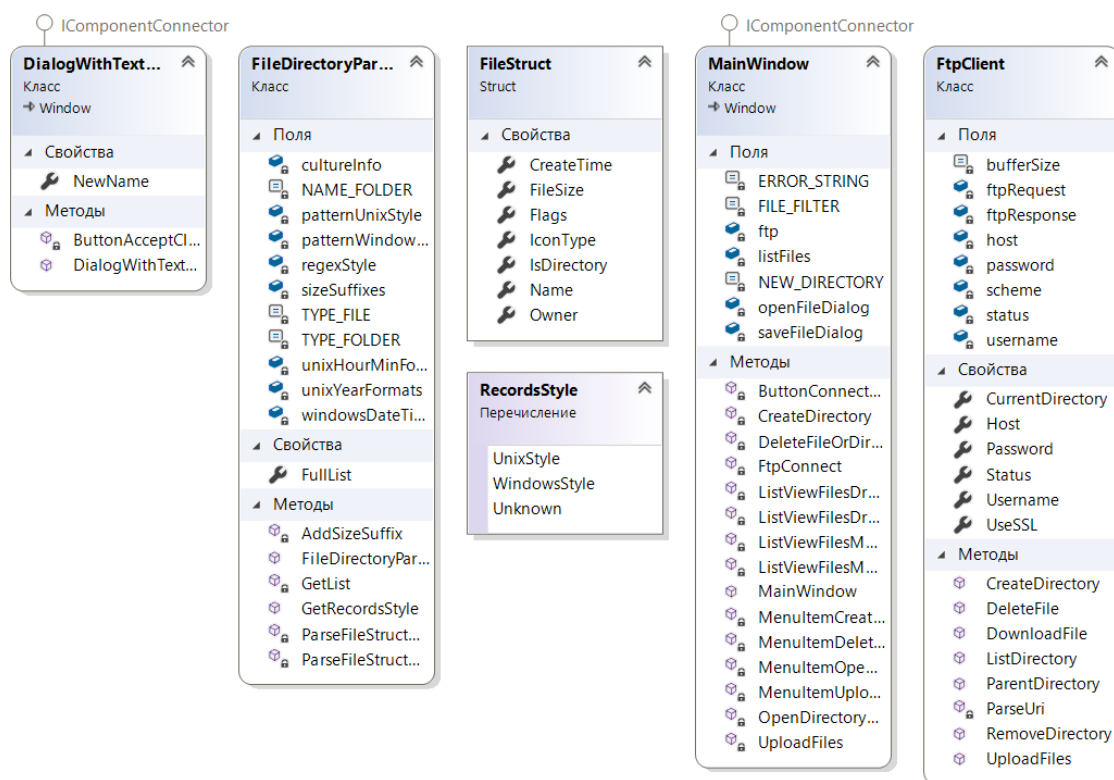


Рисунок 3.3 – Диаграмма классов

В основе метода DownloadFile используется FTP-метод RETR. В качестве параметров используются: путь директории, имя файла и путь, куда будет сохранён файл.

Для удаления файлов и каталогов используются DeleteFile и RemoveDirectory соответственно. Соответствующие им FTP-методы: DELE и RMD. Параметры: путь директории и имя файла или каталога.

Для загрузки файлов используется метод UploadFiles, который в качестве параметров получает путь директории и массив имён файлов для загрузки.

Для получения родительского каталога из пути используется метод GetParentDirectory.

3.2.2 Класс FileDirectoryInfo

Для анализа FTP-ответов в методе ListDirectory (см. пункт 3.2.1) и разбиения на структуры был введён вспомогательный класс FileDirectoryInfo. Данный класс содержит такие методы, как:

- 1 GetList. Данный метод позволяет получить из ответа сервера список файлов и каталогов.

2 **GetRecordsStyle**. В этом методе анализируются записи для определения операционной системы, используемой на FTP-сервере. На выходе есть три возможных значения: **WindowsStyle**, **UnixStyle**, **Unknown**.

3 **ParseFileStructWindowsStyle**. Анализ записей, если FTP-сервер находится под управлением операционной системы Windows.

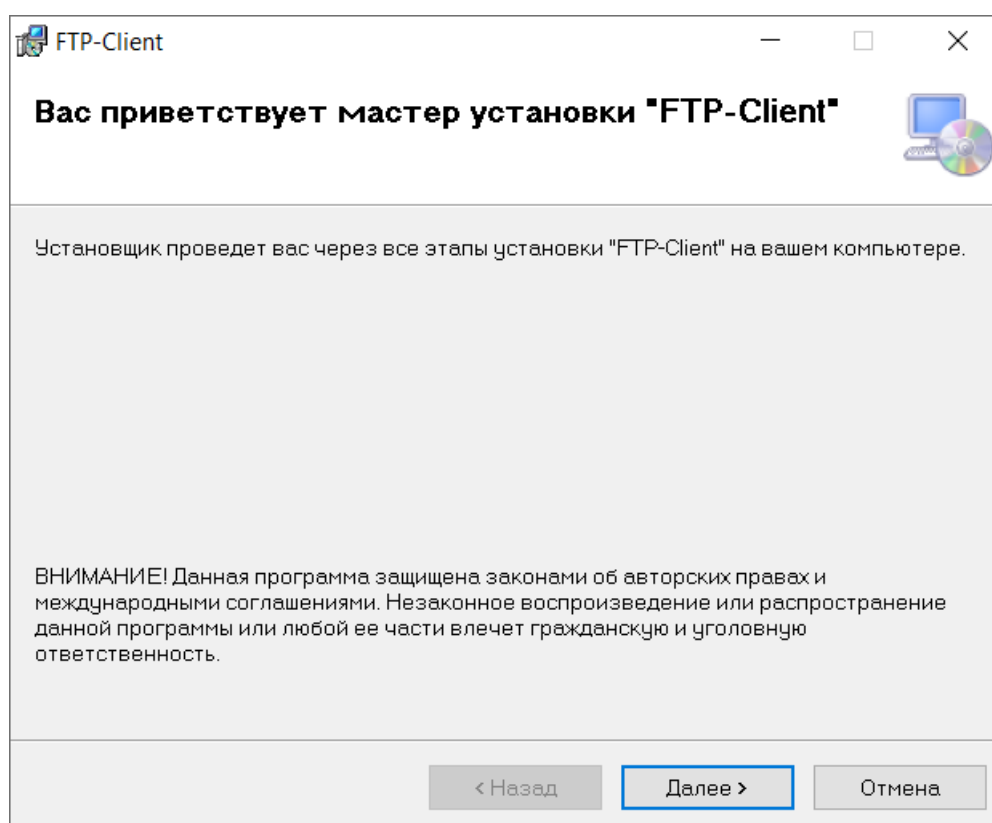
4 **ParseFileStructUnixStyle**. Анализ записей, если FTP-сервер находится под управлением Unix-подобной операционной системы.

5 **AddSizeSuffix**. Добавляет суффикс к размеру файла (байт, КБ, МБ и др.).

Схема алгоритма составления списка файлов и каталогов отображена в приложении А.

3.3 Проектирование установщика

С помощью расширения Microsoft Visual Studio Installer Project был разработан установщик программы. Окно приветствия изображено на рисунке 3.4.



4 ТЕСТИРОВАНИЕ, ПРОВЕРКА РАБОТОСПОСОБНОСТИ И АНАЛИЗ ПОЛУЧЕННЫХ РЕЗУЛЬТАТОВ

Проведено тестирование программного средства. Тестирование программного средства производилась на персональном компьютере с установленной операционной системой Windows 10.

4.1 Тестирование функционала программы

Таблица 4.1 – Тестирование функционала программы

Номер теста	Тестируемая функциональность	Последовательность действий	Ожидаемый и полученный результаты
1	Подключение к удалённому серверу.	– оставить поле «Хост» пустым; – нажать кнопку «Подключиться по FTP».	Выводится диалоговое окно с ошибкой о невозможности оставить поле «Хост» пустым.
2	Подключение к удалённому серверу.	– ввести корректные данные в поля авторизации; – нажать кнопку «Подключиться по FTP».	Отображается содержимое текущего каталога и код ответа с описанием.
3	Удаление файла.	– нажать правую кнопку мыши на файле из списка; – в контекстном меню выбрать пункт «Удалить».	Файл удалён. Отображается содержимое текущего каталога без удалённого файла.
4	Удаление каталога.	– нажать правую кнопку мыши на каталоге из списка; – в контекстном меню выбрать пункт «Удалить».	Каталог удалён. Отображается содержимое текущего каталога без удалённого каталога.

Продолжение таблицы 4.1

Номер теста	Тестируемая функциональность	Последовательность действий	Ожидаемый и полученный результаты
5	Создание нового каталога.	<ul style="list-style-type: none"> – нажать правую кнопку мыши вне элементов списка; – в контекстном меню выбрать пункт «Создать каталог»; – оставить поле ввода пустым; – нажать кнопку «ОК». 	Каталог не создан.
6	Создание нового каталога.	<ul style="list-style-type: none"> – нажать правую кнопку мыши вне элементов списка; – в контекстном меню выбрать пункт «Создать каталог»; – ввести в поле ввода название каталога; – нажать кнопку «ОК». 	Создан каталог. Отображается содержимое текущего каталога с новым каталогом.
7	Загрузка файла с сервера.	<ul style="list-style-type: none"> – нажать два раза левой кнопкой мыши по файлу; – в диалоговом окне сохранения выбрать каталог; – нажать кнопку «Сохранить». 	Файл сохранён на компьютере в выбранном каталоге.
8	Открытие каталога.	<ul style="list-style-type: none"> – нажать два раза левой кнопкой мыши на каталоге. 	Отображается содержимое выбранного каталога. Изменяется текущий путь.

Продолжение таблицы 4.1

Номер теста	Тестируемая функциональность	Последовательность действий	Ожидаемый и полученный результаты
9	Загрузка файлов на сервер с помощью Drag-and-drop.	<ul style="list-style-type: none"> – выбрать файл на компьютере; – удерживая файл левой кнопкой мыши, перетащить его в список программы; – отпустить левую кнопку мыши. 	Файлы загружены. Отображается содержимое текущего каталога с новыми файлами.
10	Загрузка файлов на сервер.	<ul style="list-style-type: none"> – нажать правую кнопку мыши вне элементов списка; – в контекстном меню выбрать пункт «Загрузить»; – в диалоговом окне открытия выбрать файлы; – нажать кнопку «Открыть». 	Файлы загружены. Отображается содержимое текущего каталога с новыми файлами.
11	Загрузка файлов на сервер.	<ul style="list-style-type: none"> – нажать правую кнопку мыши вне элементов списка; – в контекстном меню выбрать пункт «Загрузить»; – нажать кнопку «Отмена». 	Файлы не загружены.
12	Скачивание файла на компьютер пользователя.	<ul style="list-style-type: none"> – нажать два раза левой кнопкой мыши по файлу; – нажать кнопку «Отмена». 	Файлы не сохранены на компьютере.

Продолжение таблицы 4.1

Номер теста	Тестируемая функциональность	Последовательность действий	Ожидаемый и полученный результаты
13	Подключение к удалённому серверу с указанным каталогом в адресе сервера.	– ввести корректные данные в поля авторизации, в адресе хоста ввести каталог, существующий на сервере; – нажать кнопку «Подключиться по FTP».	Отображается содержимое каталога, указанного в адресе хоста, и код ответа с описанием.
14	Подключение к удалённому серверу с неверно указанным логином или паролем.	– ввести некорректные данные в поля авторизации; – нажать кнопку «Подключиться по FTP».	Вывод ошибки о невозможности подключения к удалённому хосту.

4.2 Вывод из прохождения тестирования

Программа успешно прошла все тесты, что показывает корректность работы программы и соответствие функциональным требованиям.

5 РУКОВОДСТВО ПО УСТАНОВКЕ И ИСПОЛЬЗОВАНИЮ

Для того, чтобы приступить непосредственно к использованию FTP-клиента, необходимо установить приложение. Для этого требуется запустить Setup.exe и пройти шаги установки следуя инструкциям. После установки необходимо запустить программу. Начальное окно программы изображено на рисунке 5.1.

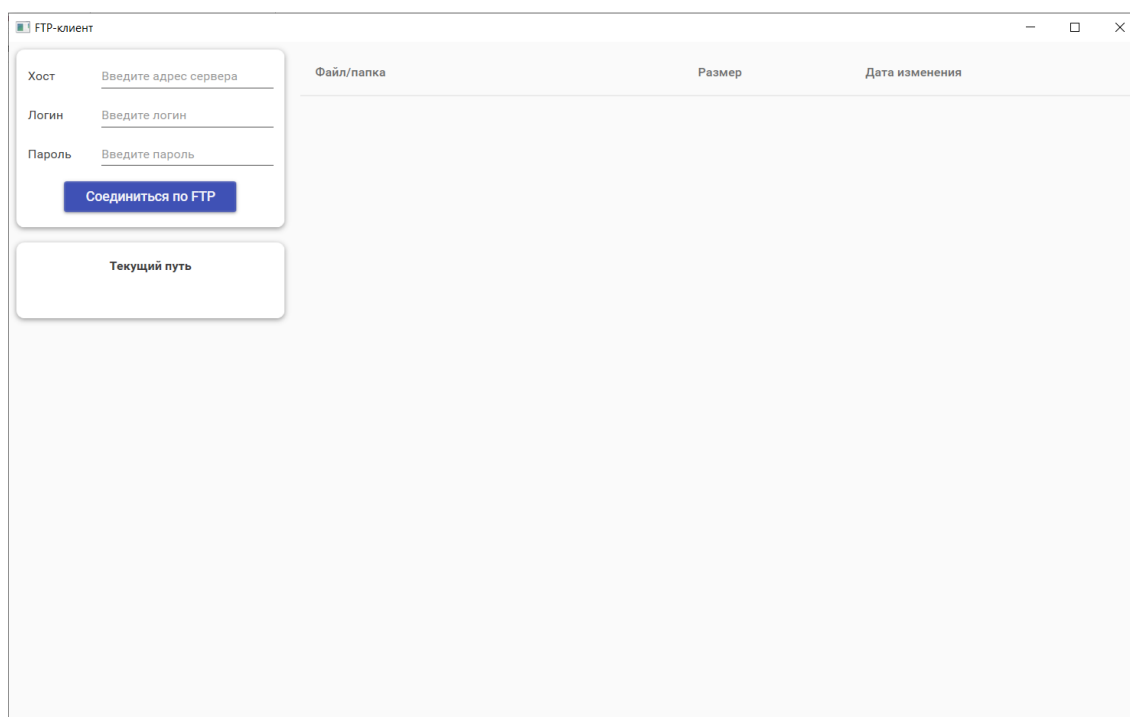
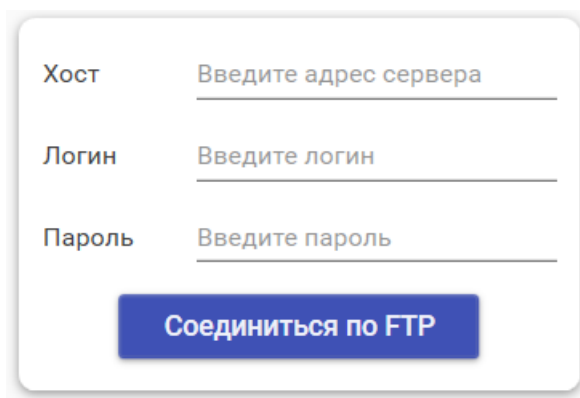


Рисунок 5.1 – Начальное окно программы

Для того, чтобы иметь доступ к удалённым данным, необходимо сначала подключиться к FTP-серверу. Для этого необходимо заполнить поля авторизации (см. рисунок 5.2). В первом поле вводится адрес удалённого сервера. Он может начинаться с идентификатора протокола, если идентификатор не введён, то используется стандартный. Затем обязательной частью идет имя сервера. Также можно указать путь к директории. Во втором и третьем полях пользователь вводит логин и пароль соответственно и получает доступ к файлам и папкам в соответствии с его правами доступа. (см. рисунок 5.3)



Хост

Логин

Пароль

Соединиться по FTP

Рисунок 5.2 – Поля ввода для авторизации

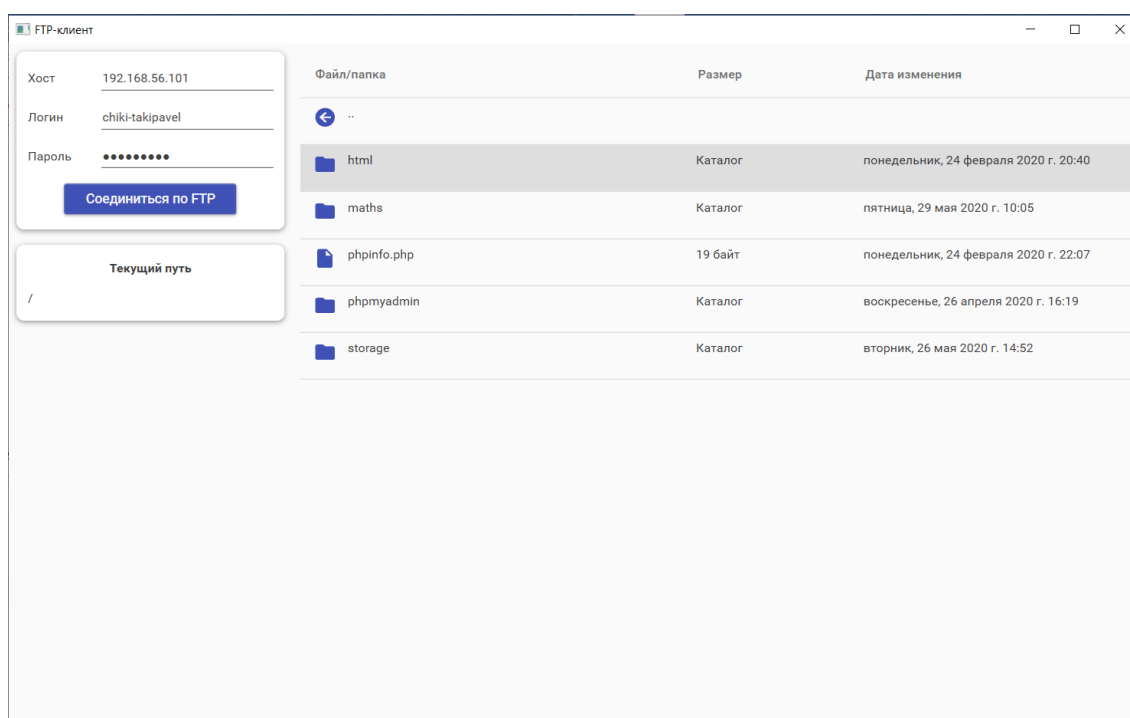


Рисунок 5.3 – Окно программы после авторизации

Загрузка выбранного файла или открытие директории производится двойным щелчком мыши. Также это действие можно выполнить с помощью вызова контекстного меню нажатием правой кнопкой мыши по выделенному элементу списка (см. рисунок 5.4). Удаление файла или директории также выполняется с помощью вывода контекстного меню для выбранного элемента списка. Если же вызвать контекстное меню вне списка, то будут отображены функции: создание каталога и загрузка файлов (см. рисунок 5.5).

Файл/папка	Размер	Дата изменения
← ..		
html	Каталог	понедельник, 24 февраля 2020 г. 20:40
maths	Каталог	пятница, 29 мая 2020 г. 10:05
phpinfo.php	19 байт	понедельник, 24 февраля 2020 г. 22:07
phpmyadmin	Каталог	воскресенье, 26 апреля 2020 г. 16:19
storage	Каталог	вторник, 26 мая 2020 г. 14:52

Открыть
Удалить

Рисунок 5.4 – Контекстное меню для элемента списка

Файл/папка	Размер	Дата изменения
← ..		
html	Каталог	понедельник, 24 февраля 2020 г. 20:40
maths	Каталог	пятница, 29 мая 2020 г. 10:05
phpinfo.php	19 байт	понедельник, 24 февраля 2020 г. 22:07
phpmyadmin	Каталог	воскресенье, 26 апреля 2020 г. 16:19
storage	Каталог	вторник, 26 мая 2020 г. 14:52

Создать каталог
Загрузить

Рисунок 5.5 – Контекстное меню вне элементов списка

При загрузке файлов выводится диалоговое окно с полем ввода, где необходимо ввести название нового каталога. Пример изображён на рисунке 5.6.

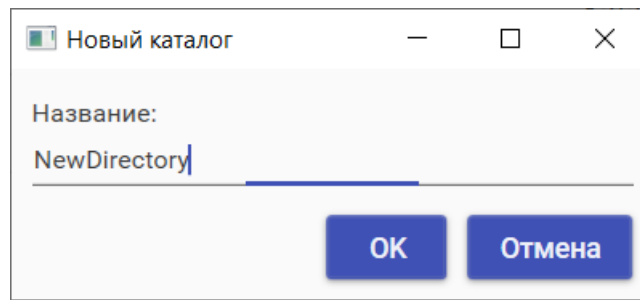


Рисунок 5.6 – Диалоговое окно создания нового каталога

Можно загрузить файлы на сервер с помощью перетаскивания файла с компьютера пользователя в список файлов и каталогов программы.

ЗАКЛЮЧЕНИЕ

В рамках данного курсового проекта был произведен анализ предметной области и реализовано программное и информационное обеспечение FTP-клиент. Согласно поставленным задачам, в данном приложении были реализованы такие возможности, как вывод списка файлов и каталогов, удаление файлов и каталогов, создание каталогов, загрузка файлов на сервер и скачивание файлов с сервера.

Для успешного выполнения всех поставленных задач потребовалось подробно разобраться с сетевым протоколом FTP, изучить основы C#, ознакомиться с возможностями среды Microsoft Visual Studio.

При тестировании и отладке не было выявлено случаев некорректной работы программы, нестабильной работы, появления сторонних ошибок и т.д.

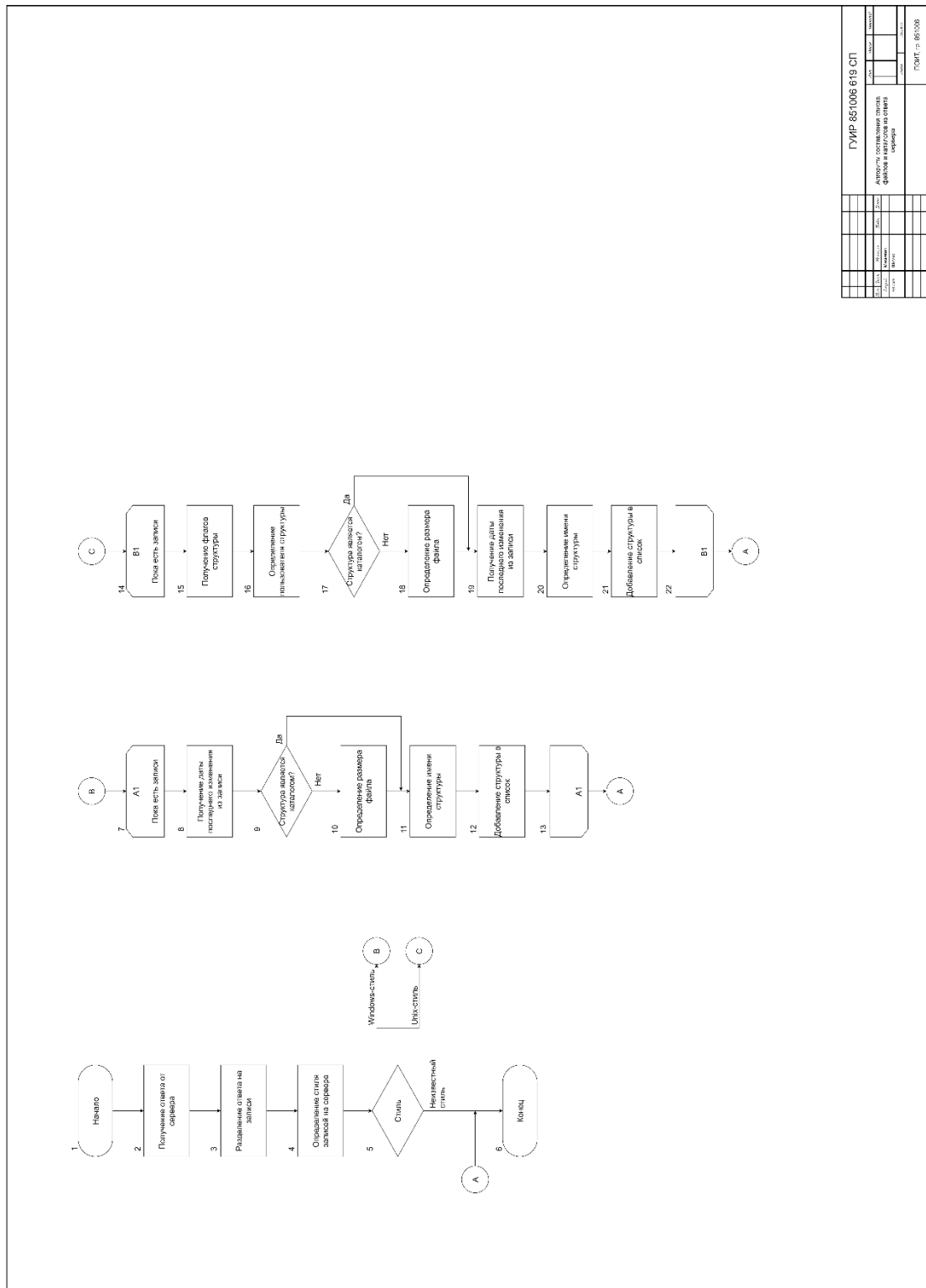
Написанный код легко модифицируется для добавления новых функций. В дальнейшем возможны улучшения и доработки, вносящие новый функционал в программу.

Спроектированная система имеет интуитивно понятный интерфейс для пользователя, проста в использовании, не требует серьезных аппаратных затрат. Разработанное программное средство можно применять в личных целях.

СПИСОК ЛИТЕРАТУРЫ

- [1] FileZilla – Wikipedia [Электронный ресурс] Режим доступа: <https://ru.wikipedia.org/wiki/FileZilla> — Дата доступа: 29.02.2020.
- [2] WinScp – Wikipedia [Электронный ресурс] Режим доступа: <https://ru.wikipedia.org/wiki/WinSCP> — Дата доступа: 29.02.2020.
- [3] CuteFtp – Wikipedia [Электронный ресурс] Режим доступа: <https://ru.wikipedia.org/wiki/CuteFTP> — Дата доступа: 01.03.2020.
- [4] Центр разработки для Windows [Электронный ресурс] Режим доступа: <https://docs.microsoft.com> — Дата доступа: 12.03.2020.
- [5] FTP – Wikipedia [Электронный ресурс] Режим доступа: <https://ru.wikipedia.org/wiki/FTP> — Дата доступа: 27.03.2020.
- [6] Компьютерные сети – Сайт Андрея Созыкина [Электронный ресурс] Режим доступа: https://www.asozykin.ru/courses/networks_online — Дата доступа: 27.03.2020.
- [7] Таненбаум Э., Уэзеролл Д. Компьютерные сети. 5-е изд. / Э. Таненбаум. – СПб.: Питер, 2012. – 960 с.
- [8] В. Олифер, Н. Олифер, «Компьютерные сети. Принципы, технологии, протоколы» (4-е издание) – СПб.: Питер, 2010. — 944 с.

Схема алгоритма составления списка файлов и каталогов из ответа сервера



ПРИЛОЖЕНИЕ Б

Исходный код программы

Файл FtpClient.cs

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Net;
using System.Text;

namespace CourseWork_CSaN
{
    class FtpClient
    {
        const int bufferSize = 1024;

        private FtpWebRequest ftpRequest;
        private FtpWebResponse ftpResponse;
        private string scheme;
        private string host;
        private string username;
        private string password;
        private string status;

        public string Host { get => host; set => host = ParseUri(value.Trim()); }
        public string Username { get => username; set => username = value.Trim(); }
        public string Password { get => password; set => password = value.Trim(); }
        public string Status { get => status; set => status = value.Trim(); }
        public string CurrentDirectory { get; set; }
        public bool UseSSL { get; set; } = false;

        /// <summary>
        /// Метод LIST для получения подробного списка файлов и каталогов на FTP-сервере
        /// </summary>
        /// <param name="path">Путь директории</param>
        /// <returns>Список файлов и каталогов</returns>
        public List<FileStruct> ListDirectory(string path)
        {
            try
            {
                path = string.IsNullOrEmpty(path) ? "/" : path;
            }
        }
    }
}
```

```

        ftpRequest = (FtpWebRequest)WebRequest.Create(Uri.EscapeUriString(scheme +
Host + path));
        ftpRequest.Credentials = new NetworkCredential(Username, Password);
        ftpRequest.Method = WebRequestMethods.Ftp.ListDirectoryDetails;
        ftpRequest.EnableSsl = UseSSL;
        ftpRequest.UseBinary = true;
        ftpRequest.UsePassive = true;
        ftpRequest.KeepAlive = true;

        string response;
        using (ftpResponse = (FtpWebResponse)ftpRequest.GetResponse())
        {
            using StreamReader stream = new
StreamReader(ftpResponse.GetResponseStream(), Encoding.ASCII);
            response = stream.ReadToEnd();
            Status = ftpResponse.StatusDescription;
        }

        FileDirectoryParser parser = new FileDirectoryParser(response); //Парсим
полученные данные
        return parser.FullList;
    }
    catch
    {
        throw new Exception("Невозможно подключиться к серверу.");
    }
}

/// <summary>
/// Метод RETR для загрузки файла с FTP-сервера
/// </summary>
/// <param name="path">Путь директории</param>
/// <param name="fileName">Имя файла</param>
/// <param name="downloadPath">Путь, где будет сохранён файл</param>
public void DownloadFile(string path, string fileName, string downloadPath)
{
    try
    {
        ftpRequest = (FtpWebRequest)WebRequest.Create(Uri.EscapeUriString(scheme +
Host + path + "/" + fileName));
        ftpRequest.Credentials = new NetworkCredential(Username, Password);
        ftpRequest.Method = WebRequestMethods.Ftp.DownloadFile;
        ftpRequest.EnableSsl = UseSSL;
        ftpRequest.UseBinary = true;
    }
}

```

```

        ftpRequest.UsePassive = true;
        ftpRequest.KeepAlive = true;

        using FileStream downloadedFile = new FileStream(downloadPath, FileMode.Create,
FileAccess.ReadWrite);
        using (ftpResponse = (FtpWebResponse)ftpRequest.GetResponse())
        {
            using Stream responseStream = ftpResponse.GetResponseStream();
            byte[] buffer = new byte[1024];
            int size;
            while ((size = responseStream.Read(buffer, 0, bufferSize)) > 0)
            {
                downloadedFile.Write(buffer, 0, size);
            }
            Status = ftpResponse.StatusDescription;
        }
    }
    catch
    {
        throw new Exception("Невозможно скачать файл с сервера.");
    }
}

/// <summary>
/// Метод STOR для загрузки файла на FTP-сервер
/// </summary>
/// <param name="path">Путь директории</param>
/// <param name="fileName">Имя файла</param>
public void UploadFiles(string path, string[] fileNames)
{
    try
    {
        byte[] fileContents;
        foreach (string fileName in fileNames)
        {
            ftpRequest = (FtpWebRequest)WebRequest.Create(Uri.EscapeUriString(scheme +
Host + path + "/" + Path.GetFileName(fileName)));
            ftpRequest.Credentials = new NetworkCredential(Username, Password);
            ftpRequest.Method = WebRequestMethods.Ftp.UploadFile;
            ftpRequest.EnableSsl = UseSSL;
            ftpRequest.UseBinary = true;
            ftpRequest.UsePassive = true;
            ftpRequest.KeepAlive = true;

```

```

        using (StreamReader uploadedFile = new StreamReader(fileName))
        {
            fileContents = Encoding.UTF8.GetBytes(uploadedFile.ReadToEnd());
        }
        using (Stream requestStream = ftpRequest.GetRequestStream())
        {
            requestStream.Write(fileContents, 0, fileContents.Length);
        }
        using (ftpResponse = (FtpWebResponse)ftpRequest.GetResponse())
        {
            Status = ftpResponse.StatusDescription;
        }
    }
}
catch
{
    throw new Exception("Невозможно загрузить файлы на сервер.");
}
}

/// <summary>
/// Метод DELE для удаления файла с FTP-сервера
/// </summary>
/// <param name="path"></param>
/// <param name="fileName"></param>
public void DeleteFile(string path, string fileName)
{
    try
    {
        ftpRequest = (FtpWebRequest)WebRequest.Create(Uri.EscapeUriString(scheme +
Host + path + "/" + fileName));
        ftpRequest.Credentials = new NetworkCredential(Username, Password);
        ftpRequest.Method = WebRequestMethods.Ftp.DeleteFile;
        ftpRequest.EnableSsl = UseSSL;
        ftpRequest.UseBinary = true;
        ftpRequest.UsePassive = true;
        ftpRequest.KeepAlive = true;

        using (ftpResponse = (FtpWebResponse)ftpRequest.GetResponse())
        {
            Status = ftpResponse.StatusDescription;
        }
    }
}
catch

```



```

    {
        throw new Exception("Невозможно удалить файл на сервере.");
    }
}

/// <summary>
/// Метод MKD для создания каталога на FTP-сервере
/// </summary>
/// <param name="path">Путь директории</param>
/// <param name="folderName">Имя создаваемого каталога</param>
public void CreateDirectory(string path, string folderName)
{
    try
    {
        ftpRequest = (FtpWebRequest)WebRequest.Create(Uri.EscapeUriString(scheme +
Host + path + "/" + folderName));
        ftpRequest.Credentials = new NetworkCredential(Username, Password);
        ftpRequest.Method = WebRequestMethods.Ftp.MakeDirectory;
        ftpRequest.EnableSsl = UseSSL;
        ftpRequest.UseBinary = true;
        ftpRequest.UsePassive = true;
        ftpRequest.KeepAlive = true;

        using (ftpResponse = (FtpWebResponse)ftpRequest.GetResponse())
        {
            Status = ftpResponse.StatusDescription;
        }
    }
    catch
    {
        throw new Exception("Невозможно создать каталог на сервере.");
    }
}

/// <summary>
/// Метод RMD для удаления каталога с FTP-сервера
/// </summary>
/// <param name="path">Путь директории</param>
/// <param name="folderName">Имя каталога для удаления</param>
public void RemoveDirectory(string path, string folderName)
{
    try
    {

```

```

        ftpRequest = (FtpWebRequest)WebRequest.Create(Uri.EscapeUriString(scheme +
Host + path + "/" + folderName));
        ftpRequest.Credentials = new NetworkCredential(Username, Password);
        ftpRequest.Method = WebRequestMethods.Ftp.RemoveDirectory;
        ftpRequest.EnableSsl = UseSSL;
        ftpRequest.UseBinary = true;
        ftpRequest.UsePassive = true;
        ftpRequest.KeepAlive = true;

        using (ftpResponse = (FtpWebResponse)ftpRequest.GetResponse())
        {
            Status = ftpResponse.StatusDescription;
        }
    }
    catch
    {
        throw new Exception("Невозможно скачать файл с сервера.");
    }
}

/// <summary>
/// Получение схемы, адреса хоста и текущего пути из URI
/// </summary>
/// <param name="uriString">URI</param>
/// <returns>Адрес хоста</returns>
private string ParseUri(string uriString)
{
    if (Uri.TryCreate(host, UriKind.Absolute, out Uri uri))
    {
        scheme = (uri.Scheme == Uri.UriSchemeFtp) ? uri.Scheme : string.Empty;
        CurrentDirectory = uri.AbsolutePath;
        return uri.GetComponents(UriComponents.HostAndPort, UriFormat.UriEscaped);
    }
    else
    {
        scheme = Uri.UriSchemeFtp + Uri.SchemeDelimiter;
        int pathPos = uriString.IndexOf('/');
        if (pathPos >= 0)
        {
            CurrentDirectory = uriString.Substring(pathPos);
            uriString = uriString.Remove(pathPos);
        }
        else
        {

```

```

        CurrentDirectory = string.Empty;
    }
    if (Uri.CheckHostName(uriString) == UriHostNameType.Unknown)
        uriString = string.Empty;
    return uriString;
}
}

/// <summary>
/// Возвращает путь родительского каталога
/// </summary>
/// <param name="path">Путь каталога</param>
/// <returns>Путь родительского каталога</returns>
public string GetParentDirectory(string path)
{
    path.TrimEnd('/');
    if (path.LastIndexOf('/') >= 0)
    {
        path = path.Remove(path.LastIndexOf('/'));
    }
    else
    {
        path = string.Empty;
    }

    return path;
}
}
}

```

Файл **FileDirectoryInfo.cs**

```

using System;
using System.Collections.Generic;
using System.Globalization;
using System.IO;
using System.Linq;
using System.Text.RegularExpressions;

namespace CourseWork_CSaN
{
    /// <summary>
    /// Структура для хранения информации о файле или каталоге
    /// </summary>

```

```

public struct FileStruct
{
    public string Flags { get; set; }
    public bool IsDirectory { get; set; }
    public string Owner { get; set; }
    public string FileSize { get; set; }
    public string CreateTime { get; set; }
    public string Name { get; set; }
    public string IconType { get; set; }
}

public enum RecordsStyle
{
    UnixStyle,
    WindowsStyle,
    Unknown
}

public class FileDirectoryParser
{
    const string NAME_FOLDER = "Каталог";
    const string TYPE_FOLDER = "Folder";
    const string TYPE_FILE = "File";

    readonly string[] sizeSuffixes = { "байт", "КБ", "МБ", "ГБ", "ТБ", "ПБ", "ЭБ", "ЗБ", "ЙБ"
};
    readonly string patternWindowsStyle = @"^(?<datetime>\d+-\d+-\d+)\s+\d+:\d+(?:AM|PM))\s+(?<sizeordir><DIR>\|(\d+)\s+(?<name>.+)$";
    readonly string windowsDateTimeFormat = "MM-dd-yy hh:mmtt";
    readonly string patternUnixStyle = @"^(?<flags>[\w-]+)\s+(?<inode>\d+)\s+(?<owner>\w+)\s+(?<group>\w+)\s+" +
    @"(?<size>\d+)\s+(?<datetime>\w+\s+\d+\s+\d+|\w+\s+\d+\s+\d+:\d+)\s+(?<name>.+)$";
    readonly string[] unixHourMinFormats = { "MMM dd HH:mm", "MMM dd H:mm", "MMM d HH:mm", "MMM d H:mm" };
    readonly string[] unixYearFormats = { "MMM dd yyyy", "MMM d yyyy" };
    readonly IFormatProvider cultureInfo = CultureInfo.GetCultureInfo("en-us");
    Regex regexStyle;

    public List<FileStruct> FullList { get; }

    public FileDirectoryParser(string response)
    {
        FullList = GetList(response);
    }

```

```

}

/// <summary>
/// Получает из ответа список файлов и каталогов
/// </summary>
/// <param name="data">Данные ответа</param>
/// <returns>Список файлов и каталогов</returns>
private List<FileStruct> GetList(string data)
{
    List<FileStruct> listResult = new List<FileStruct>
    {
        new FileStruct() { Name = "..", IconType = "ArrowBackCircle" }
    };
    string[] records = data.Split('\n');
    RecordsStyle style = GetRecordsStyle(records); //Получаем стиль записей на сервере
    switch (style)
    {
        case RecordsStyle.UnixStyle:
            regexStyle = new Regex(patternUnixStyle);
            break;
        case RecordsStyle.WindowsStyle:
            regexStyle = new Regex(patternWindowsStyle);
            break;
        case RecordsStyle.Unknown:
            return listResult;
    }
    foreach (string record in records)
    {
        if (!string.IsNullOrEmpty(record))
        {
            FileStruct fileStruct;
            if (style == RecordsStyle.UnixStyle)
            {
                fileStruct = ParseFileStructUnixStyle(record);
            }
            else
            {
                fileStruct = ParseFileStructWindowsStyle(record);
            }
            if (fileStruct.Name != "" && fileStruct.Name != "." && fileStruct.Name != "..")
            {
                listResult.Add(fileStruct);
            }
        }
    }
}

```

```

    }
    return listResult;
}

/// <summary>
/// Определяет операционную систему, на которой работает FTP-сервер
/// </summary>
/// <param name="records">Записи</param>
/// <returns>Стиль записей</returns>
public RecordsStyle GetRecordsStyle(string[] records)
{
    foreach (string record in records)
    {
        if (record.Length > 10 && Regex.IsMatch(record.Substring(0, 10), "(-|d)((-|r)(-|w)(-|x)){3}"))
        {
            return RecordsStyle.UnixStyle;
        }
        else if (record.Length > 8 && Regex.IsMatch(record.Substring(0, 8), "[0-9]{2}-[0-9]{2}-[0-9]{2}"))
        {
            return RecordsStyle.WindowsStyle;
        }
    }
    return RecordsStyle.Unknown;
}

/// <summary>
/// Парсинг записи, если FTP-сервер работает на Windows (WindowsStyle)
/// </summary>
/// <param name="record">Запись</param>
/// <returns>Структура файла</returns>
private FileStruct ParseFileStructWindowsStyle(string record)
{
    FileStruct fileStruct = new FileStruct();
    Match match = regexStyle.Match(record);
    fileStruct.CreateTime = DateTime.ParseExact(match.Groups["datetime"].Value,
windowsDateFormat,
    cultureInfo, DateTimeStyles.None).ToString("f");
    if (match.Groups["sizeordir"].Value == "<DIR>")
    {
        fileStruct.IsDirectory = true;
        fileStruct.FileSize = NAME_FOLDER;
        fileStruct.IconType = TYPE_FOLDER;
    }
}

```

```

    }
    else
    {
        fileStruct.IsDirectory = false;
        fileStruct.FileSize = AddSizeSuffix(long.Parse(match.Groups["sizeordir"].Value));
        fileStruct.IconType = TYPE_FILE;
    }
    fileStruct.Name = match.Groups["name"].Value.Trim();

    return fileStruct;
}

/// <summary>
/// Парсинг записи, если FTP-сервер работает на Unix (UnixStyle)
/// </summary>
/// <param name="record">Запись</param>
/// <returns>Структура файла</returns>
private FileStruct ParseFileStructUnixStyle(string record)
{
    FileStruct fileStruct = new FileStruct();
    Match match = regexStyle.Match(record);
    fileStruct.Flags = match.Groups["flags"].Value;
    fileStruct.IsDirectory = fileStruct.Flags[0] == 'd';
    fileStruct.Owner = match.Groups["owner"].Value;
    if (!fileStruct.IsDirectory)
    {
        fileStruct.FileSize = AddSizeSuffix(long.Parse(match.Groups["size"].Value,
cultureInfo));
        fileStruct.IconType = TYPE_FILE;
    }
    else
    {
        fileStruct.FileSize = NAME_FOLDER;
        fileStruct.IconType = TYPE_FOLDER;
    }
    string tempDateTime = Regex.Replace(match.Groups["datetime"].Value, @"\s+", " ");
    if (tempDateTime.IndexOf('.') >= 0)
    {
        fileStruct.CreateTime = DateTime.ParseExact(tempDateTime, unixHourMinFormats,
cultureInfo, DateTimeStyles.None).ToString("f");
    }
    else
    {

```

```

        fileStruct.CreateTime = DateTime.ParseExact(tempDateTime, unixYearFormats,
cultureInfo, DateTimeStyles.None).ToString("f");
    }
    fileStruct.Name = match.Groups["name"].Value.Trim();

    return fileStruct;
}

/// <summary>
/// Переводит количество байт в КБ, МБ, ГБ и т.д.
/// </summary>
/// <param name="value">Размер в байтах</param>
/// <returns>Размер с суффиксом</returns>
private string AddSizeSuffix(long value)
{
    if (value < 0) { return "-" + AddSizeSuffix(-value); }
    if (value == 0) { return string.Format("{0:n0} {1}", 0, sizeSuffixes[0]); }

    int mag = (int)Math.Log(value, 1024);
    decimal adjustedSize = (decimal)value / (1L << (mag * 10));
    if (Math.Round(adjustedSize) >= 1000)
    {
        mag += 1;
        adjustedSize /= 1024;
    }

    return string.Format("{0:n0} {1}", adjustedSize, sizeSuffixes[mag]);
}
}
}

```


Обозначение					Наименование					Дополнительные сведения				
					<u>Текстовые документы</u>									
БГУИР КР 1–40 01 01 619 ПЗ					Пояснительная записка					40 с.				
					<u>Графические документы</u>									
ГУИР 851006 619 СА					"Алгоритм составления списка файлов и каталогов из ответа сервера", А1, схема программы, чертеж					Формат А1				