|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | **Министерство науки и высшего образования Российской Федерации**  Калужский филиал  федерального государственного бюджетного  образовательного учреждения высшего образования  ***«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»***  ***(КФ МГТУ им. Н.Э. Баумана)*** |

**ФАКУЛЬТЕТ** \_ ***ИУ-КФ «Информатика и управление»\_\_\_\_\_\_\_\_\_***

**КАФЕДРА** \_\_ ***ИУ5-КФ «Системы обработки информации»***

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**к курсовому проекту на тему:**

**FTP клиент**

по дисциплине ***Сетевые технологии в АСОИУ***

Студент гр. ИУК5.Б-61 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ( Доровских Д. Е. )

(подпись) (Ф.И.О.)

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_( Вершинин Е.В )

(подпись) (Ф.И.О.)

Оценка руководителя \_\_\_\_\_ баллов \_\_\_\_\_\_\_\_\_\_\_

30-50 (дата)

Оценка защиты \_\_\_\_\_ баллов \_\_\_\_\_\_\_\_\_\_\_

30-50 (дата)

Оценка проекта \_\_\_\_\_ баллов \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(оценка по пятибалльной шкале)

Комиссия: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

(подпись) (Ф.И.О.)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

(подпись) (Ф.И.О.)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

(подпись) (Ф.И.О.)

Калуга, 2021

**Министерство науки и высшего образования Российской Федерации**

Калужский филиал  
 федерального государственного бюджетного   
образовательного учреждения высшего образования

***«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»***

***(КФ МГТУ им. Н.Э. Баумана)***

УТВЕРЖДАЮ

Заведующий кафедрой **\_\_ИУ5-КФ\_\_\_**

\_\_\_\_\_\_\_\_\_\_\_(Е.В.Вершинин)

«\_ 06 \_»\_\_\_\_февраля\_\_\_\_2021г.

**ЗАДАНИЕ**

**на выполнение курсового проекта**

по дисциплине \_\_\_***Сетевые технологии в АСОИУ\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

Студент Доровских Д.Е. ИУК5-61Б

(фамилия, инициалы, индекс группы)

Руководитель Вершинин Е.В.\_ \_\_\_

График выполнения проекта: 25% к\_4\_нед., 50% к\_7\_нед., 75% к\_10\_нед., 100% к\_14\_нед.

1. Тема курсового проекта FTP клиент\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2. Техническое задание

*Выполнить исследование и описание предметной области. Разработать и реализовать FTP клиент..*

#### **Оформление курсовой работы**

* 1. Расчетно-пояснительная записка на 32 листах формата А4.
  2. Перечень графического материала КР (плакаты, схемы, чертежи и т.п.)

1. UML-диаграмма
2. Схема структурная

Дата выдачи задания « 06 » февраля 2021г.

Руководитель курсового проекта \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/\_\_\_\_\_Е.В. Вершинин\_\_/

Задание получил\_\_\_\_\_\_\_\_\_\_\_\_\_\_/\_\_\_\_\_Доровских Д,Е\_\_\_\_\_\_\_/ «\_\_\_\_\_»\_\_февраля\_2021г.

Примечание:

Задание оформляется в двух экземплярах: один выдается студенту, второй хранится на кафедре.

**ОГЛАВЛЕНИЕ**

[ВВЕДЕНИЕ 5](#_Toc73202618)

[1. ТЕХНИЧЕСКОЕ ЗАДАНИЕ 6](#_Toc73202619)

[1.1 Общие сведения 6](#_Toc73202620)

[1.1.1 Наименование 6](#_Toc73202621)

[1.1.2 Исполнитель 6](#_Toc73202622)

[1.1.3 Перечень нормативно-технических документов, методических материалов, использованных при разработке ТЗ: 6](#_Toc73202623)

[1.1.4 Плановые сроки по созданию приложения 6](#_Toc73202624)

[1.2 Назначение и цели создания системы 6](#_Toc73202625)

[1.2.1 Назначение системы 6](#_Toc73202626)

[1.2.2 Цели создания системы 6](#_Toc73202627)

[1.3 Требования к системе 6](#_Toc73202628)

[1.3.1 Требования к архитектуре АСОИ 6](#_Toc73202629)

[1.4.2 Требования к надежности 6](#_Toc73202630)

[1.4.3 Требования к функциям, выполняемым системой 6](#_Toc73202631)

[1.4.6 Требования к методическому обеспечению 7](#_Toc73202632)

[1.5 Состав и содержание работ по созданию(развитию) системы 7](#_Toc73202633)

[1.6 Порядок контроля и приемки системы 8](#_Toc73202634)

[1.6.1 Методы испытания системы 8](#_Toc73202635)

[1.6.2 Общие требования к приемке работ по стадиям 8](#_Toc73202636)

[1.6.3 Статус приемочной комиссии 8](#_Toc73202637)

[1.7 Требования к документированию 8](#_Toc73202638)

[2. НАУЧНО-ИССЛЕДОВАТЕЛЬСКАЯ ЧАСТЬ 9](#_Toc73202639)

[2.1 Постановка задачи 9](#_Toc73202640)

[2.2 Описание предметной области 9](#_Toc73202641)

[2.2.1 Общие сведения 9](#_Toc73202642)

[2.3 Анализ аналогов 10](#_Toc73202643)

[2.4 Перечень реализуемых функций 12](#_Toc73202644)

[2.5 Обоснование выбора платформы для разработки 12](#_Toc73202645)

[3. ПРОЕКТНО-КОНСТРУКТОРСКАЯ ЧАСТЬ 15](#_Toc73202646)

[3.1 Разработка структуры приложения 15](#_Toc73202647)

[3.2 Разработка архитектуры приложения 16](#_Toc73202648)

[3.3 Разработка систем передачи информации 17](#_Toc73202649)

[3.4 Разработка интерфейса взаимодействия пользователя с системой …………………………………………………………………………….20](#_Toc73202650)

[4. ПРОЕКТНО-ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ 23](#_Toc73202651)

[4.1 Тестирование и отладка макета рабочей программы 23](#_Toc73202652)

[4.2 Разработка руководства пользователя 25](#_Toc73202653)

[4.3 Экспериментальные данные тестирования процедур и функциональных задач 28](#_Toc73202654)

[4.4 Разработка спецификации на оснащение подразделений, эксплуатирующих разработанное приложение, компьютерным и сетевым оборудованием 30](#_Toc73202655)

[ЗАКЛЮЧЕНИЕ 31](#_Toc73202656)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 32](#_Toc73202657)

# **ВВЕДЕНИЕ**

В данной работе разрабатывается FTP клиент, для соединения с сервером, который способен копировать файлы с удаленного сервера и на удаленный сервер.

В техническом задании описаны основные требования к разрабатываемому приложению.

В исследовательской части выполнен анализ предметной области, проанализированы аналоги, обоснован выбор среды и средств разработки.

В конструкторской части приведен интерфейс приложения. Дано краткое описание использованных структур данных и архитектуры приложения.

В технологической части объяснены основные принципы взаимодействия с разработанным приложением. Выполнено тестирование.

В заключении сформированы выводы о проделанной работе.

В списке использованных источников перечислены ссылки на адреса использованных Интернет-ресурсов.

1. **ТЕХНИЧЕСКОЕ ЗАДАНИЕ**

## **1.1 Общие сведения**

### **1.1.1 Наименование**

FTP клиент

### **1.1.2 Исполнитель**

Студент КФ МГТУ Им. Н.Э. Баумана Доровских Д.Е. гр. ИУК5-61Б

### **1.1.3 Перечень нормативно-технических документов, методических материалов, использованных при разработке ТЗ:**

-Методические указания по выполнению курсовой работы по дисциплине “Сетевые технологии в АСОИУ”, автором которых является Е.В. Вершинин, оформленная в соответствии с ГОСТ 34.602-89.

### **1.1.4 Плановые сроки по созданию приложения**

Начало работ – 6 февраля 2021 года, окончание работ -1 мая 2021 года.

## **1.2 Назначение и цели создания системы**

### **1.2.1 Назначение системы**

FTP-клиент

### **1.2.2 Цели создания системы**

Клиент должен соединятся с сервером, а также совершать различные операции с данными, располагающим на нем.

## **1.3 Требования к системе**

### **1.3.1 Требования к архитектуре АСОИ**

Следует обеспечить комфортный и максимально упрощенный доступ пользователей к сервисам.

### **1.4.2 Требования к надежности**

В случае, если любой пользователь FTP клиента совершает ошибку при его использовании, он должен быть уведомлен об этом, причем клиент должен оставаться работоспособным.

### **1.4.3 Требования к функциям, выполняемым системой**

Клиент должен обеспечивать реализацию следующих функций:

1. Просмотр файлов с сервера.
2. Загрузка файлов на сервер.
3. Выгрузка файла с сервера.

**1.1.4 Требования к входным данным**

К входным данным FTP клиента относятся:

1. Логин пользователя
2. Пароль пользователя
3. Адрес сервера

**1.4.5 Требования к выходным данным**

К входным данным FTP клиента относятся:

1. Список файлов и каталогов на сервере

2) Скачанный файл

### **1.4.6 Требования к методическому обеспечению**

Разработку системы требуется сопровождать расчетно-пояснительной запиской и графической частью. Расчетно-пояснительная записка состоит из следующих частей:

1. Техническое задание

2. Исследовательская часть.

3.Проектно-конструкторская часть.

4. Проектно-технологическая часть.

## **1.5 Состав и содержание работ по созданию(развитию) системы**

Первый этап – изучение предметной области, определение состава и структуры данных. В результате прохождения первого этапа определяются: логическая структура данных, ТЗ, описание объекта автоматизации и целей разработки.

Второй этап – анализ пользовательских сценариев работы. В результате создается пользовательский интерфейс.

Третий этап – анализ взаимодействия клиента с серверной компонентой. В результате определяются: архитектура приложения и структура классов.

## **1.6 Порядок контроля и приемки системы**

### **1.6.1 Методы испытания системы**

Испытания проектируемой системы проводятся заказчиком.

### **1.6.2 Общие требования к приемке работ по стадиям**

Место проведения приемки работ КФ МГТУ им. Н.Э.Баумана.

Сроки проведения приёмки работ согласовывается с заказчиком и исполнителем

### **1.6.3 Статус приемочной комиссии**

Прием работы на всех этапах осуществляется непосредственным руководителем курсовой работы. По окончанию разработки работоспособное приложение демонстрируется на комиссии.

## **1.7 Требования к документированию**

-ТЗ по проектируемой системе

-Расчетно-пояснительная записка

-Презентация Microsoft PowerPoint

1. **НАУЧНО-ИССЛЕДОВАТЕЛЬСКАЯ ЧАСТЬ**

## **2.1 Постановка задачи**

Задачей проектирования курсовой работы является реализация FTP клиента, способного подключаться к FTP серверу, с возможностью последующей загрузки файлов.

## **2.2 Описание предметной области**

### **2.2.1 Общие сведения**

FTP – протокол передачи файлов, использующийся для подключения к серверу и последующей передачи файлов по сети. Для подключения пользователю необходим адрес FTP сервера, а также данные для входа.

Сервер обеспечивает доступ к файлам по паролю, показывает файлы и папки, пользователь видит только те файлы и папки, которые ему доступны. Сервер также следит за статусом передачи и в случае ошибки, останавливает процесс.

FTP клиент – программа, способная упростить доступ к серверу. Способная предоставить пользователю доступ к папкам и файлам, находящимся на сервере.

Протокол базируется на архитектуре “клиент-сервер”, т. е для работы по FTP необходим FTP сервер и FTP клиент.

Принцип взаимодействия FTP сервера и FTP клиента:

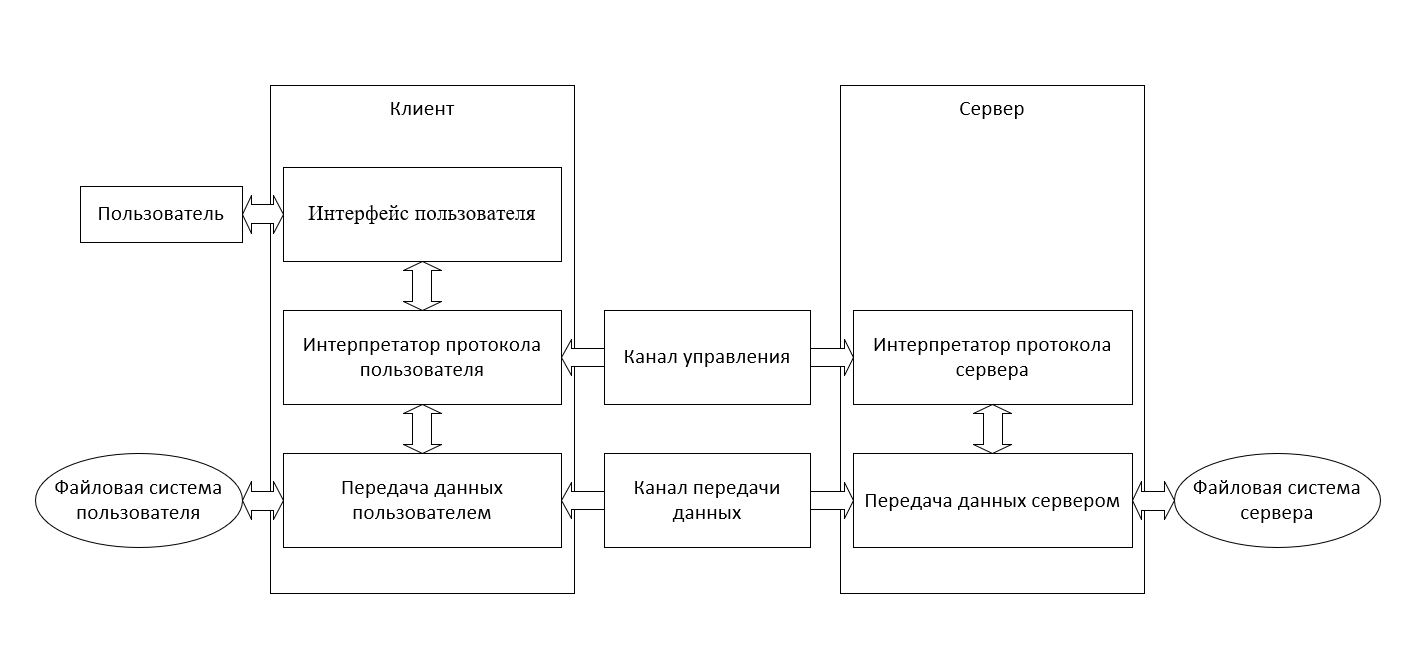


Рисунок 1 – принцип взаимодействия сервера и клиента

1. Пользователь соединяется с сервером, используя адрес сервера, логин и пароль.
2. Устанавливается соединение интерпретаторами протокола как со стороны клиента, так и со стороны сервера.
3. Пользователь с помощью клиента передает команды серверу, определяющий действия пользователя и параметра FTP соединения.
4. Один из участников, будь то клиент или сервер, ожидает открытия порта FTP для передачи информации.
5. Другой же участник, отличный от предыдущего, открывает соединение и начинает передачу данных.
6. После завершения передачи соединение закрывается, но канал между интерпретаторами остается открытым.

## **2.3 Анализ аналогов**

1) FileZilla – популярный и бесплатный FTP клиент с открытым исходным кодом, работающий на Windows, Linux и MacOS. Данный клиент находится на рынке более 20 лет. Он имеет синхронизацию каталогов, поддерживает многопоточную загрузку файлов и способен возобновить докачку файлов после разрыва соединения.

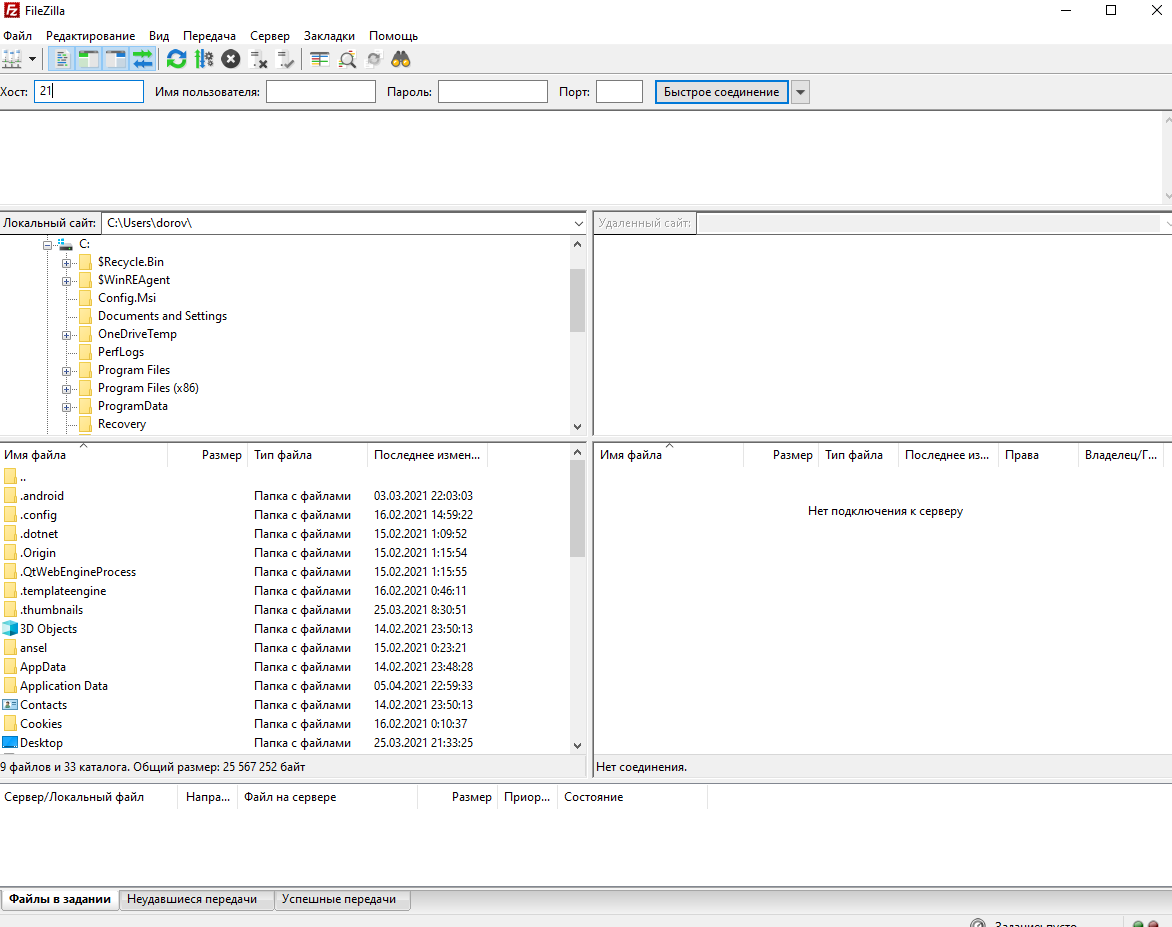


Рисунок 2 - интерфейс FileZilla

2) Cyberduck – FTP клиент с простым интерфейсом, способным упростить выполнение всех задач, связанных с FTP. Он способен работать не только с FTP протоколами, но и с SFTP, WebDAV, Cloud Files, Google Docs, и Amazon S3. Клиент является бесплатным и работает на MacOS, так и на Windows.

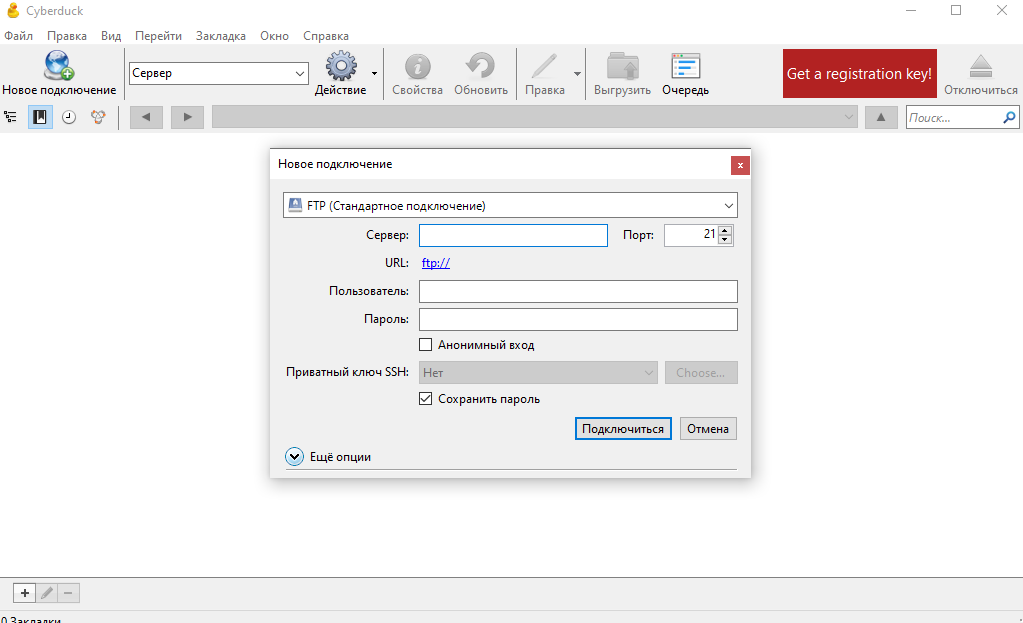


Рисунок 3 – интерфейс Cyberduck

## **2.4 Перечень реализуемых функций**

- Подключение к серверу  
 - Просмотр файлов и каталогов хранящихся на сервере

- Возможность скачивать, загружать и удалять файлы

## **2.5 Обоснование выбора платформы для разработки**

Основная задача заключается в разработке оконного приложения. Так как приложение является оконным и будет разрабатываться для платформы OS Windows. Причиной выбора данной платформы является ее распространённость.

Среди современных инструментов для разработки оконных приложений можно рассмотреть такие популярные варианты IDE, как: MS Visual Studio, Pelles C, Net Beans, Code Block. Eclipce, QT creator и IntelliJ IDE.

Выбор сред разработки пал на MS Visual Studio, так как в ней имеются бесплатные Community версии, совместимость с OS Windows и понятный интерфейс.

Среди языков программирования для разработки оконных приложений можно рассмотреть следующие языки:

* С++
* С#
* Python
* Java
* С

Среди всех вышеперечисленных языков программирования под цель разработки подходит C#. Он является объектно-ориентированным языком, имеет типовую защищенность, может “собирать мусор”, поддерживает совместимость версий и многое другое. C# разрабатывался с нуля изначально, что позволило не переносить все неудачные особенности других языков, смог включить в себя все лучшее из таких языков программирования как C, C++, Java, Python и другие.

С# – один из языков программирования высокого уровня, который может использоваться для создания приложений на платформе .NET. Платформа .NET — многоязыковая среда, открытая для свободного включения новых языков, создаваемых не только Microsoft, но и другими фирмами. Все языки, включаемые в платформу .NET, должны опираться на единый каркас, роль которого играет .NET Framework. Главным достоинством языка С# можно назвать его согласованность с возможностями .NET Framework и вытекающую отсюда компонентную ориентированность. Компоненты позволяют решать проблему модульного построения приложений на новом уровне.

Для работы с FTP протоколом была использована библиотека FluetFTP, раннее известная как System.Net.FtpClient. Эта библиотека позволяет полностью управлять FTP и FTPS для .NET. Она предоставляет обширные команды FTP, такие как, выгрузку/загрузку файлов, защищенные соединения по протоколу SSL/TLS. Данная библиотека полностью написана на C#, выпускается под лицензией MIT, поэтому его можно использовать как в проприетарных приложениях, так и в бесплатных.

1. **ПРОЕКТНО-КОНСТРУКТОРСКАЯ ЧАСТЬ**

## **3.1 Разработка структуры приложения**

После исследование предметной области было принято решение о создании FTP-клиента, с помощью которого можно подключиться к серверу, просмотреть файлы на нем, выгрузить на сервер и загрузить с сервера.

Далее будет описано взаимодействие клиента с системой, ожидаемой функциональности системы и понятий предметной области, которая относится к описанию функциональности системы, была составлена UML диаграмма вариантов использования. Диаграмма отражает функциональность и поведение системы и является составной частью модели, позволяющей описать систему на концептуальном уровне.

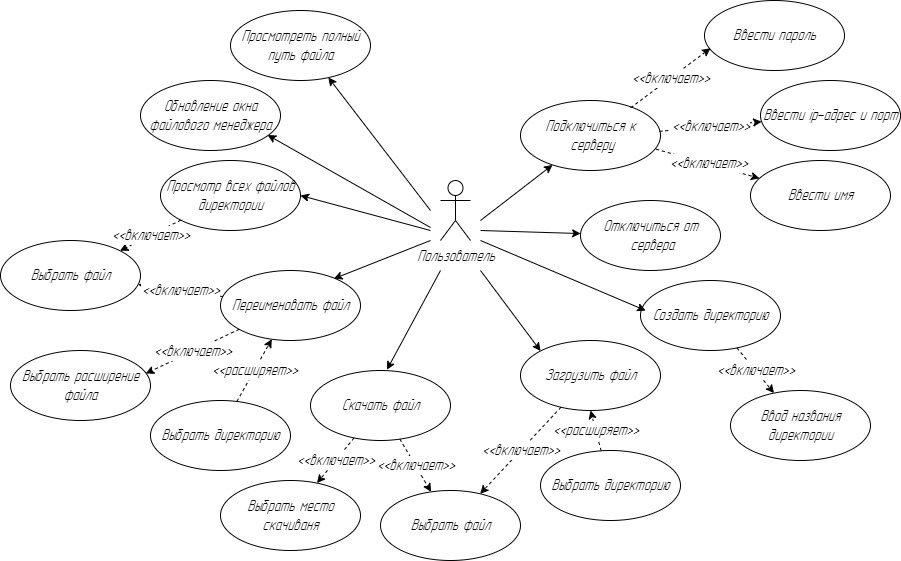


Рисунок 4 – UML диаграмма вариантов использования

В приложении будет одно действующее лицо, которым является пользователь. Со стороны пользователя будет реализована возможность подключения к серверу, ввод данных пользователя, отключение от сервера, загрузки файлов, скачивания файлов, переименование и просмотр всех файлов директорий.

## **3.2 Разработка архитектуры приложения**

Для данного приложения будем использовать архитектуру клиент-серверное приложение, в котором в качестве клиента выступает наша программа, а сервером является настроенный локальный FTP сервер.

Клиент-сервер описывает отношение взаимодействующих программ в приложении. Сервер предоставляет услуги одному или нескольким клиентам, которые отправляют запросы на услуги.

Клиенту не нужно беспокоиться о состоянии сервера при выполнении запроса и доставке ответа. Клиенту необходимо только понять ответ, основанный на протоколе приложения, то есть содержание и форматирование данных для запрашиваемой услуги.

Клиент и сервер обмениваются командами в шаблоне запрос-ответ. Клиент отправляет запрос, а сервер должен предоставить ответ. Данный обмен сообщениями является межпроцессным взаимодействием (IPC) и может быть запущен как на одном и том же компьютере, так и на разных, соединенных сетью. Клиент и сервер должны иметь между собой общий язык и знали, чего стоит ожидать друг от друга. Этот язык определен в протоколе связи. Все протоколы клиент-серверной модели работают на уровне приложений.

Система разбивается на две части – клиентскую и серверную. Пользователь взаимодействует с клиенткой частью, которая обеспечивает надсетевой интерфейс для работы с сервером. Клиентская часть системы при необходимости обращается по сети к серверной части.

В нашей программе серверную часть создавать не нужно, она уже должна быть развернута, следовательно, необходимо реализовать только клиентскую часть. Так как шифрование пакетов с данными не обязательно, то по умолчанию сервер будет использовать 21 порт для принятия запросов и отправки ответов. Для передачи данных будет открыт 20 порт. Сервер должен иметь параметры входа для пользователя и в нем содержатся логин и пароль пользователя, либо сервер должен иметь возможность подключать анонимных пользователей.

Программа клиент. В этом случае пользователь должен подключиться к серверу путем заполнения полей хост, порт, а также данные пользователя для получения доступа к серверу. После этого пользователь получает возможность просмотра файлов и директорий на сервере, загрузить, выгрузить и удалить файлы, а также переименовывать файлы/директории на сервере.

* 1. **Разработка систем передачи информации**

Для взаимодействия между сервером и клиентами будет использоваться сокеты.

Сокет – это один конец двустороннего канала связи между двумя программами, работающими в сети. Соединяя вместе два сокета, можно передавать данные между разными процессами. Реализация сокетов обеспечивает инкапсуляцию протоколов сетевого и транспортного уровней.

Существует два типа основных сокетов – потоковые сокеты и дейтаграммные.

Потоковый сокет – это сокет с установленным соединением, который состоит из потока байтов, который может быть двунаправленным, то есть через эту конечную точку приложение может и передавать, и получать данные. Этот сокет подходит для передачи больших объемов данных. Потоковые сокеты используют протокол Transmission Control Protocol (TCP), что обеспечивает поступление данных на другую сторону в нужной последовательности.

Дейтаграммные сокеты – сокеты без организации соединений, то есть никакого явного соединения между ними не устанавливается – сообщение отравляется указанному сокету и, соответственно, может получаться от указанного сокета.

Так как нам необходимо двунаправленное соединение, то будет использоваться потоковый сокет. Сокет состоит из IP-адреса и порта, используемого приложением TCP, поскольку IP-адрес является уникальным в интернете, а номера портов уникальны на отдельной машине, номера сокетов также уникальны во всем интернете, что позволяет процессу общаться через сеть с другим процессом исключительно на основе номера сокета. За определенными службами номера портов зарезервированы, например порт 21, который мы будем использовать, является портом для FTP. В C# для работы с сокетами будем использовать встроенную библиотеку System.Net.Sockets.

Функции, которые мы будем использовать в нашем приложении для работы с сокетами показаны в таблице 1.

|  |  |
| --- | --- |
| Используемая функция | Назначение функции |
| Socket.Accept() | Создает новый объект Socket для заново созданного подключения. |
| Socket.Send(byte[],SocketFlags) | Передает данные в подключенный объект Socket, используя заданный объект SocketFlags. |
| Socket.Recive(byte[], socketFlags) | Получает данные из связанного объекта Socket в приемный буфер, используя объект SocketFlags. |
| Socket.bind(new IPEndPoint(address, port)) | Привязывает сокет к адресу, в котором инициализируется IP-адрес и порт. Сокет не должен быть привязан до этого. |
| Socket.Listen(int backlog) | Устанавливает объект Socket в состояние прослушивания. Параметр Backlog – количество соединений, которые будет принимать сервер. |
| Socket.close() | Закрывает подключение Socket и освобождает все связанные ресурсы |
| Socket.Connect() | Устанавливает подключение к удаленному узлу. |
| Socket.ProtocolType.tcp | Получает тип протокола объекта Socket. В данном случае указывает тип протокола как TCP. |
| Socket.SocketType.stream | Указывает тип сокета, являющегося экземпляром класса Socket. Поле Stream поддерживает надежные двусторонние байтовые потоки в режиме с установлением подключения без дублирования данных и без сохранения границ данных. Stream использует протокол TCP. |
| Socket.AdressFamily | Задает схему адресации, которую может использовать экземпляр класса Socket. |
| Socket.Poll(int32, SelectMode) | Определяет состояние объекта Socket. |

Сервер создает слушающий сокет и ожидает соединения от клиента. Когда клиент выполняет подключение к серверу, сервер вызывает функцию accept(), для принятия подключения. Далее клиент вызывает connect(), для установки соединения с сервером путем трехстороннего рукопожатия. На следующем шаге расположена секция двустороннего обмена, в которой данные обмениваются между клиентом и сервером, путем вызова функций Send() и Recive(). Функция Close() закрывает сокеты и разрывает подключение клиента с сервером.

* 1. **Разработка интерфейса взаимодействия пользователя с системой**

Разработка графического интерфейса необходимо для простого взаимодействия пользователя с системой.

Программа-сервер представляет из себя настроенный сервер с использованием стандартных средств Windows. Для его настройки необходимо включить службу для работы с FTP-сервером в Internet Information Service (IIS). Далее в диспетчере информационных служб интернета (IIS) добавить FTP-сайт и указать ему IP адрес, порт, а также путь расположения. В настройках брандмуэра Windows нужно включить входящий и исходящий трафик для FTP сервера. Далее создаем пользователя или группу пользователей и добавляем их в авторизацию FTP-сайта, при этом нужно дать им права доступа к локальной папке сервера. При создании прав можно добавить анонимный доступ к серверу, это означает что абсолютно любой пользователь может подключиться к данному серверу. Если после этих действий сервер не запустился, необходимо в настройках маршрутизатора создать удаленное подключение и зарезервировать 21 порт и локальный адрес компьютера.

Программа-клиент представляет из себя GUI-программу, которая реализуется с помощью стандартной функции для построения клиентских приложений под названием Windows Presentation Foundation (WPF), которая входит в подсистему .NET Framework.

Для правильной разработки интерфейса необходимо учитывать потребности всех пользователей, следовательно, интерфейс должен быть прост, удобен и понятен каждому. Для этого необходимо поставить себя на место пользователя и расписать User Story.

1. Я, как пользователь хочу иметь понятное поле для подключения к серверу.
2. Я, как пользователь хочу иметь поле, в котором расположены все файлы и папки, находящиеся на сервере, с возможностью навигации и упорядочивании по имени или типу файлов/директорий.
3. Я, как пользователь хочу иметь поле, в котором собраны все необходимые кнопки для работы с сервером
4. Я, как пользователь хочу иметь поле, в котором будет выводится состояние сервера

На основе US был разработан прототип, представленный на рисунке 5.

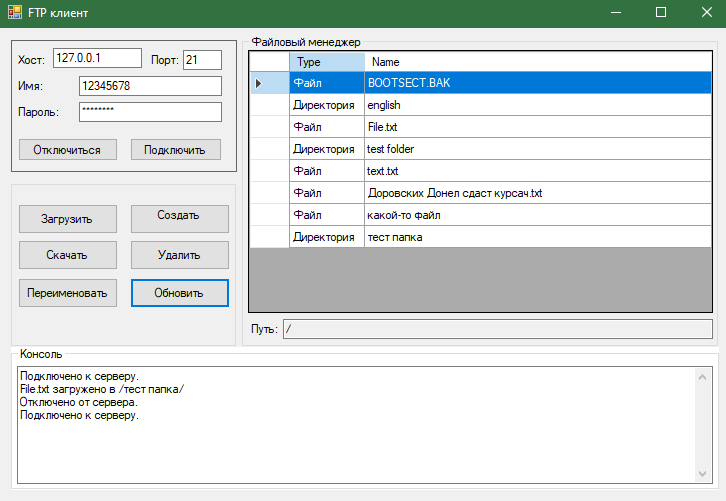


Рисунок 5 – Прототип приложения

# **4. ПРОЕКТНО-ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ**

## **4.1 Тестирование и отладка макета рабочей программы**

При завершении проектирования приложения, необходимо оформить требования к обработке ошибок и к обработке некорректных данных разрабатываемой системы. Эти требования нужно оформить в виде тест-кейсов для визуального и наглядного понимания.

Тест-кейс №1. Проверка подключения к серверу

Шаги:

1. В поле “хост” записываем значения, отличающиеся от адреса сервера
2. В поле “порт” записываем значения, отличающиеся от порта сервера
3. Нажать кнопку “Подключить”

Таблица 2 – Ожидаемый результат для тест-кейса

|  |  |
| --- | --- |
| Вводимое значение | Ожидаемый результат |
| Вводим в поле “хост” значения, отличающиеся от любого существующего FTP-сервера | В консоли запишется “Не удалось отобразить файлы с сервера.” и “Не удалось подключиться к серверу.” |
| Оставляем поле “хост” пустым | В консоли запишется “Не удалось отобразить файлы с сервера.” и “Не удалось подключиться к серверу.” |
| Вводим в полу “хост” адрес локального сервера (127.0.0.1) или localhost | Выполнится подключение к локальному серверу, если он существует, иначе выдаст сообщение в консоли |
| Вводим в поле “порт” значение отличающиеся от порта сервера | В консоли запишется “Не удалось отобразить файлы с сервера.” и “Не удалось подключиться к серверу.” |
| Оставляем поле “порт” пустым | Клиент попробует подключиться по 21 порту, если он отличается, то запишет в консоль ошибку |

Тест-кейс №2 Проверка авторизации пользователя

Шаги:

1. В поле “имя” вводим имя пользователя
2. В поле “пароль” вводим пароль
3. Нажимаем кнопку “Подключить”

Таблица 3 – Ожидаемый результат для тест-кейса №2

|  |  |
| --- | --- |
| Вводимое значение | Ожидаемый результат |
| Вводим правильное имя пользователя и неправильный пароль | В консоли запишется ошибка “530 неверное имя или пароль” и “Не удалось подключиться к серверу.” |
| Вводим неправильное имя пользователя и неправильный пароль | В консоли запишется ошибка “530 неверное имя или пароль” и “Не удалось подключиться к серверу.” |
| Оставляем поле “Имя” или “пароль” пустым | В консоли запишется ошибка “530 неверное имя или пароль” и “Не удалось подключиться к серверу.” |
| Вводим в поле “имя” значение anonymous, а поле “пароль” оставляем пустым или записываем любое значение | Будет выполнено подключение к серверу, если сервер имеет возможность подключение анонимных пользователей (параметры доступа могут быть изменены на сервере) |
| Вводим в поле “имя” и “пароль” значения, которые являются верными | В консоли запишется “Подключено к серверу.” Будет выполнено подключение к серверу и выведется список файлов и директорий |

После написания тест-кейсов была отлажена и протестирована работа программы при возникновении ошибок.

* 1. **Разработка руководства пользователя**

Для того, чтобы обычный пользователь начал пользоваться FTP-клиентом, ему необходимо запустить файл ClientFTP.exe, с помощью двойного клика по файлу.

Далее в открывшемся диалоговом окне необходимо ввести IP адрес хоста и порт. После этого нужно ввести данные пользователя и нажать кнопку “Подключить”. Если все сделано правильно, то в консоли будет написано “Подключено к серверу.”

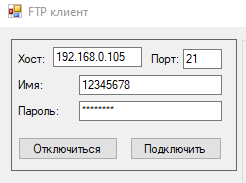


Рисунок 6 – параметры подключения

После подключения к серверу в файловом менеджере появится список файлов и директорий. В нем можно выбрать нужную нам директорию и двойным кликом раскрыть ее, чтобы посмотреть список файлов, хранящиеся в ней. Что бы вернуться в корневую папку необходимо двойным кликом нажать на строчку “../”. Также есть возможность вручную прописать путь к необходимой директории.

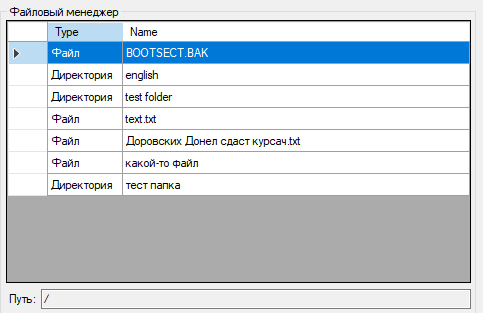


Рисунок 7 – файловый менеджер

Если пользователь хочет скачать файл или папку, ему необходимо один раз кликнуть на нужную ему строку и нажать кнопку скачать. После этого откроется окно “обзор папок”, где нужно выбрать место, куда скачать файл и если нужно, можно создать папку, в которую его поместить. Далее необходимо нажать кнопку “Ок” и начнется загрузка файла, в консоли будет показан результат загрузки.

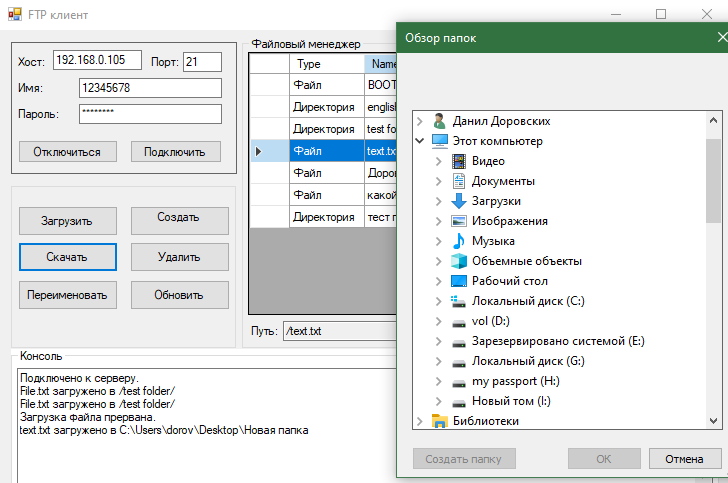


Рисунок 8 – скачивание файла и выбор места для скачивания

Для загрузки файла на сервер необходимо в файловом менеджере выбрать директорию, в которую загрузится файл, нажать кнопку “Загрузить” и выбрать файл, который нужно загрузить. Выглядит это почти также, как и скачивание. Если после загрузки файла на сервер, он не появился в списке, то нужно нажать кнопку “Обновить”.

Для удаления папки или директории, нужно выбрать ее из списка и нажать кнопку “Удалить”, после которой файл/директория сразу пропадет с сервера.

Если пользователю необходимо переименовать файл или директорию, нужно выбрать из списка и нажать кнопку “Переименовать”. Для смены названия файла нужно в конце через точку написать расширение файла, в противном же случае в консоль выдаст ошибку. В всплывающем окне нужно ввести новое название и нажать кнопку “Ок”.

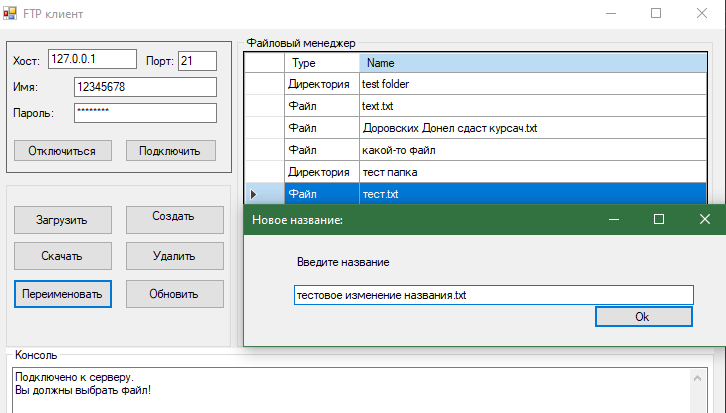


Рисунок 9 – Переименование файлов или директорий

Для создания директории на сервере нужно нажать кнопку “Создать” и в появившейся форме ввести название директории (форма такая же как при переименовании см. Рис 9).

Если пользователю нужно отключиться от сервера, то нужно нажать кнопку “Отключиться”, но если нужно переподключиться к другому серверу, можно просто ввести новые данные и нажать кнопку “подключить”, клиент сам отключится от сервера и переподключится к другому серверу.

* 1. **Экспериментальные данные тестирования процедур и функциональных задач**

Для тестирования функций приложения и анализа и нагрузки, используемых ресурсов.

Сначала была протестирована нагрузка на сеть. В тесте был отправлен 1 файл, через клиент на сервер был отправлен огромный файл с весом 100 000 000 000 байт (93,1ГБ) рис.10.

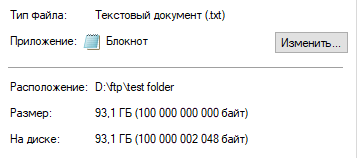


Рисунок 10 – размер файла передаваемого для теста

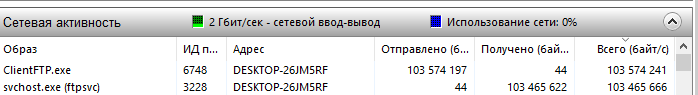


Рисунок 11 – Сетевые ресурсы, затраченные при отправке файла.

В этом тесте клиент отправил на сервер файл со скоростью около 105 000 000 байт/c (100МБ/С). Сервер получил данные с такой же скоростью.

В следующем тесте был скачан с сервера этот же файл.

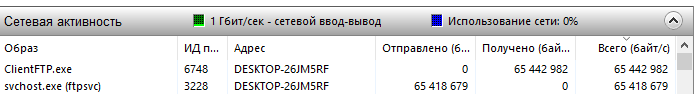


Рисунок 12 – Сетевые ресурсы, затраченные при скачивании файла

В данном тесте клиент получал данные сервера со скоростью 65 000 000 байт/c (62МБ/C).

Во время приема файла была протестирована нагрузка на оперативную память (Рисунок 13).



В данном тесте был запущен 1 клиент и было выяснено, что во время работы клиент занимает 14.8 МБ оперативной памяти.

* 1. **Разработка спецификации на оснащение подразделений, эксплуатирующих разработанное приложение, компьютерным и сетевым оборудованием**

Рекомендуемые операционные системы:

Windows 7

Windows 8

Windows 8.1

Windows 10

Windows Server 2008 R2

Рекомендуемая аппаратная конфигурация для клиента:

• Процессор Pentium 4 или более новый

• 200 МБ RAM

• 200 МБ RAM для 64-битной сборки

• 40 KБ дискового пространства

Требования для подключения к сети: для соединения с сервером по локальной сети подключение к интернету не обязательно, но если подключаться к серверу находящимся вне локальной сети рекомендуется использовать интернет-соединение со скоростью от 1Мб/с.

# **ЗАКЛЮЧЕНИЕ**

При выполнении курсовой работы на тему “FTP-клиент” была исследована и описана предметная область, проведен анализ аналогов данной системы, а также выбор инструментов и платформы разработки.

В процессе работы были получены навыки разработки приложений с применение сетевых технологий на примере использования потокового сокета на протоколе TCP. Также был проведен анализ автоматизации и разработаны методы решения технических задач, была разработана и реализована структура и архитектура системы, структура системы передачи информации и интерфейса взаимодействия пользователя с системой. Также проведено тестирование и отладка приложения.

Результатом курсовой работы является рабочее приложение FTP-клиент, который осуществляет подключение к FTP серверу с возможностью загрузки, выгрузки файлов, а также просмотр файлов, находящихся на сервере.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Е.В. Смирнова, А.В. Пролетарский, Е.А. Ромашкина и др. Технологии коммутации и маршрутизации в локальных компьютерных сетях. Учебное пособие. М.: МГТУ им. Н.Э. Баумана, 2013.
2. Сидоров В.Н., Сломинская Е.Н., Полникова Т.В., Макарова О.Ю. Оформление графической части выпускной квалификационной работы. Учебное пособие. М.: МГТУ им. Н.Э. Баумана, 2016
3. Э.Таненбаум, Д.Уэзеролл. Компьютерные сети. 5-е изд. СПб.: Питер, 2012.
4. Сокеты в C# и .Net. [Электронный ресурс] – <https://metanit.com/sharp/net/3.1.php>
5. Сокеты в C# и .Net. [Электронный ресурс] – <https://docs.microsoft.com/ru-ru/dotnet/api/system.net.sockets.socket?view=net-5.0>
6. Фленов М.Е. Библия C# 4 издание изд. БХВ-Петербург 2020г