

Министерство образования Республики Беларусь

Учреждение образования  
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА  
к курсовому проекту  
на тему

Многопоточный FTP-сервер с возможностью запроса каталогов и передачи  
их в архивированном виде.

по дисциплине

«Системное программное обеспечение вычислительных машин»

КП БГУИР 1-40 02 01.310 ПЗ

Выполнил:  
Студент гр. 050503  
Казак И. А.

Руководитель:  
Глоба А. А.

Минск 2022

Министерство образования Республики Беларусь  
Учреждение образования  
«Белорусский государственный университет  
информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

Дисциплина: Системное программное обеспечение вычислительных машин

УТВЕРЖДАЮ  
Заведующий кафедрой ЭВМ  
\_\_\_\_\_ Б.В. Никульшин  
(подпись)  
«\_\_\_\_ » \_\_\_\_\_ 2022 г.

**ЗАДАНИЕ**  
по курсовой работе студента  
Казак Ивана Александровича

1. Тема работы: «Многопоточный FTP-сервер с возможностью запроса каталогов и передачи их в архивированном виде»
2. Срок сдачи студентом законченной работы: 28.05.2022
3. Исходные данные к работе:
  - 3.1. Программа должна иметь удобный пользовательский интерфейс с необходимыми пунктами меню. В режиме реального времени контролировать подключение и отключение клиентов. По запросу предоставлять подробную информацию о файлах на сервере. Программа должна иметь возможность вернуть каталог в архивном виде.
4. Содержание пояснительной записи (перечень подлежащих разработке вопросов):
  - 4.1. Введение. 1. Обзор методов и алгоритмов решения поставленной задачи. 2. Обоснование выбранных методов и алгоритмов. 3. Описание программы для программиста. 4. Описание алгоритмов решения задачи. 5. Руководство пользователя. Заключение. Список литературы.
5. Перечень графического материала:  
Диаграмма классов.  
Структурная схема.

## КАЛЕНДАРНЫЙ ПЛАН

Наименование этапов курсовой работы	Объём этапа, %	Срок выполнения этапа	Примечания
Разработка структурной схемы программы	10	07.03–17.03	С выполнением чертежей
Разработка диаграммы классов	20	18.03–08.04	С выполнением чертежей
Разработка основного функционала приложения с выполнением блок-схем	50	09.04–10.05	С выполнением чертежа
Разработка интерфейса	75	11.05–25.05	
Завершение оформления пояснительной записи	100	26.05–10.06	

Дата выдачи задания: 20.02.2022 г.

РУКОВОДИТЕЛЬ

\_\_\_\_\_

A.A. Глоба

(подпись)

Задание принял к исполнению

\_\_\_\_\_

I.A. Казак

(дата и подпись студента)

## СОДЕРЖАНИЕ

<b>ВВЕДЕНИЕ .....</b>	<b>5</b>
<b>1 ОБЗОР ЛИТЕРАТУРЫ.....</b>	<b>6</b>
1.1 Обзор используемой литературы.....	6
1.2 Анализ существующих аналогов.....	6
1.2.1 FileZilla Server.....	6
1.2.2 FTP-сервер Xlight.....	7
1.2.3 Complete FTP.....	8
1.3 Постановка задачи.....	9
<b>2 СИСТЕМНОЕ ПРОЕКТИРОВАНИЕ.....</b>	<b>11</b>
2.1 Модуль пользовательского интерфейса.....	11
2.3 Модуль чтения и записи данных.....	11
2.4 Модуль преобразования данных.....	12
2.5 Модуль сети.....	12
<b>3 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ.....</b>	<b>13</b>
3.1 MainWindow.....	13
3.2 File.....	13
3.3 Dir.....	13
3.3 User.....	14
3.3 Server.....	14
<b>4 РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ.....</b>	<b>16</b>
4.1 Выделение ключевых процедур.....	16
4.1.1 Конструктор класса Sever.....	16
4.1.1 Ожидание соединения с сокетом start_server21.....	16
4.1.1 Ожидание соединения с сокетом new_connection21.....	16
4.1.1 Обработки команд пользователя menu.....	17
4.1.1 Получения файлов в текущем каталоге пользователя current_dir.....	18
<b>5 ПРОГРАММА И МЕТОДИКА ИСПЫТАНИЙ.....</b>	<b>20</b>
5.1 Тестирования работы приложения со стороны сервера.....	20
5.1 Тестирования работы приложения со стороны клиента.....	21
<b>6 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ.....</b>	<b>24</b>
<b>ЗАКЛЮЧЕНИЕ.....</b>	<b>29</b>
<b>СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ.....</b>	<b>30</b>
<b>ПРИЛОЖЕНИЕ А.....</b>	<b>31</b>
<b>ПРИЛОЖЕНИЕ Б.....</b>	<b>32</b>
<b>ПРИЛОЖЕНИЕ В.....</b>	<b>33</b>

## **ВВЕДЕНИЕ**

Курсовое проектирование является обязательным элементом подготовки специалиста с высшим образованием и является одной из форм текущей аттестации студента по учебной дисциплине.

Данная курсовая работа посвящена разработке многопоточного FTP-сервера с возможностью запроса каталогов и передачи их в архивированном виде.

Для реализации поставленной задачи был выбран язык программирования C++, а также, для разработки удобного пользовательского интерфейса, Фреймворк Qt. Как один из базовых и самых распространенных языков программирования C++ включает в себя процедурное программирование, позволяет создать интерактивный интерфейс, что значительно расширяет круг его применения и упрощает написание проектов высокого уровня сложности. Кроме того, этот язык наиболее приближен к системе и системным функциям среди высокоуровневых языков, таким образом являясь наиболее подходящим для решения поставленной задачи.

Данная разработка может быть использована в качестве удобного хранилища файлов и управления доступом к этим файлам. У данного проекта крайне широкая целевая аудитория так как он позволяет не имея серьезных навыков делиться и хранить свои файлы.

# **1 ОБЗОР ЛИТЕРАТУРЫ**

## **1.1 Обзор используемой литературы**

Тема курсового проекта была выбрана в первую очередь для углубления знаний по языку C++ и в области объектно-ориентированного программирования, а также получения знаний в проектировании пользовательских приложений, поэтому моей целью не является разработать конкурентоспособный продукт. Тем не менее, чтобы создать корректно работающее приложение, нужно иметь представление о существующих аналогах, об их недостатках, преимуществах и реализованных внутри функциях.

FTP расшифровывается как File Transfer Protocol — протокол передачи файлов. Он отличается от других протоколов тем, что если в процессе передачи возникает какая-то ошибка, то процесс останавливается и выводится сообщение для пользователя. Если ошибок не было, значит, пользователь получил именно тот файл, который нужен, в целости и без недостающих элементов[5].

Протоколом называется набор правил, задающих форматы сообщений и процедуры, которые позволяют компьютерам и прикладным программам обмениваться информацией. Эти правила соблюдаются каждым компьютером в сети, в результате чего любой хост-получатель может понять отправленное ему сообщение[4].

Для работы по FTP нужны двое: FTP-сервер и FTP-клиент. Что делает сервер:

- обеспечивает доступ по логину и паролю к нужным файлам;
- показывает пользователю только те файлы и папки, которые он может просматривать или загружать в них;
- следит за качеством передачи и смотрит, чтобы не было ошибок;
- управляет параметрами соединения в пассивном режиме;

FTP-клиент – это файловый менеджер, который осуществляет подключение к удаленному серверу для передачи данных. Для подключения клиента к удаленному серверу нужны следующие данные:

- логин;
- пароль;
- хост (имя сервера);
- номер порта (по умолчанию 21 для FTP-соединения);[5]

### **5.1. Анализ существующих аналогов**

Для качественного выполнения курсового проекта следует рассмотреть аналоги данной утилиты.

## 5.1.1. FileZilla Server

FileZilla Server — это серверное приложение с открытым исходным кодом для Windows. Он может администрировать как локальный сервер, так и удаленный FTP-сервер. Поддерживает обмен данными по протоколу FTP. Интерфейс FileZilla Server изображен на рисунке 1.1:

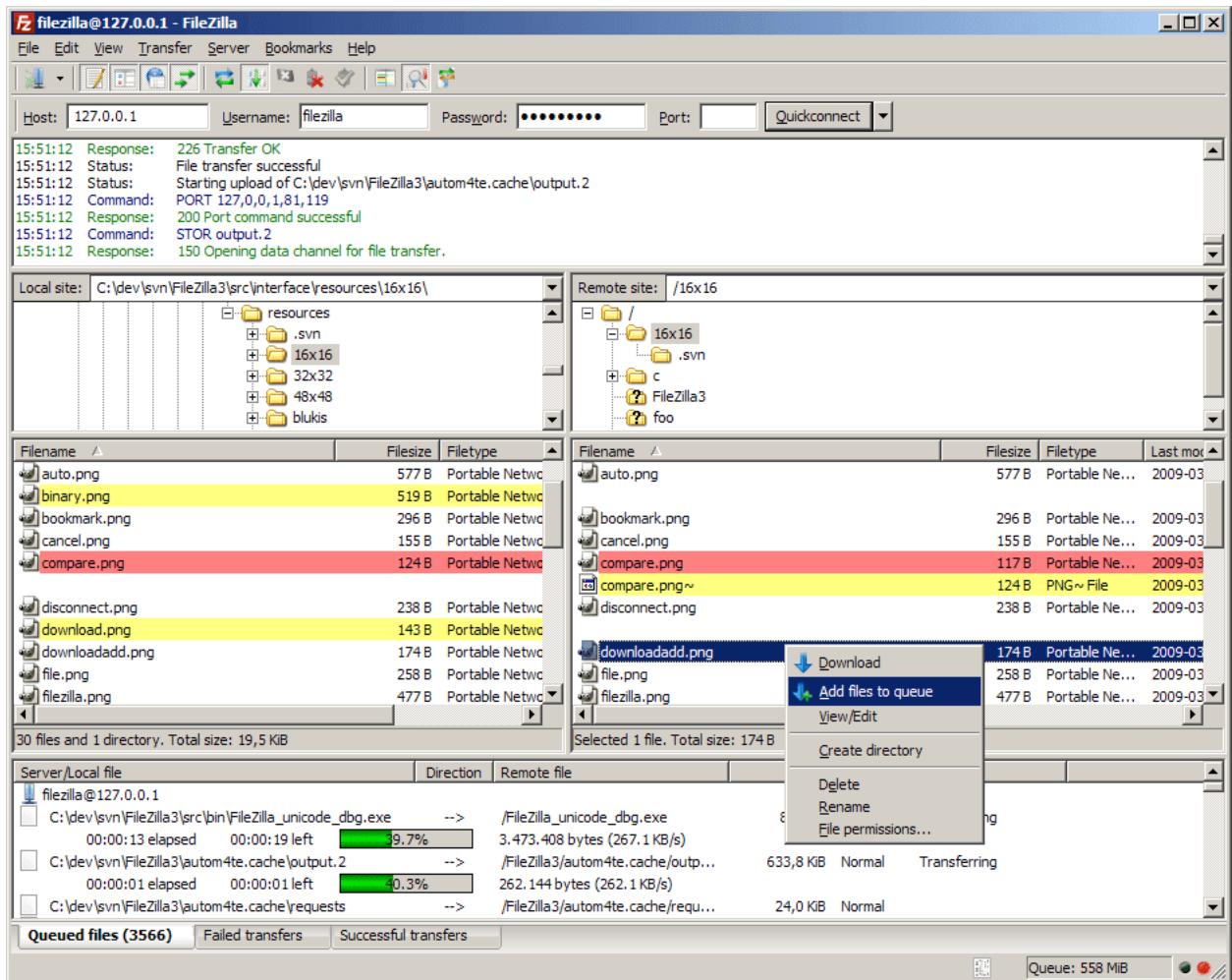


Рисунок 1.1 — Скриншот программы FileZilla Server

Достоинства FileZilla Server:

- Возможность передачи файлов одновременно;
- Поддерживает безопасную передачу файлов;
- Закладки для быстрого подключения;

Недостатки FileZilla Server:

- Не удается редактировать файлы из приложения;
- Отсутствие автоматического обновления видов папок;

### 5.1.2. FTP-сервер Xlight

Xlight — это бесплатный FTP-сервер, который выглядит намного современнее, чем FileZilla, а также содержит множество настроек.

FTP-сервер Xlight может использовать SSL и может требовать от клиентов использования сертификата. Он также поддерживает ODBC, Active Directory и аутентификацию LDAP. Интерфейс FTP-сервер Xlight изображен на рисунке 1.2:

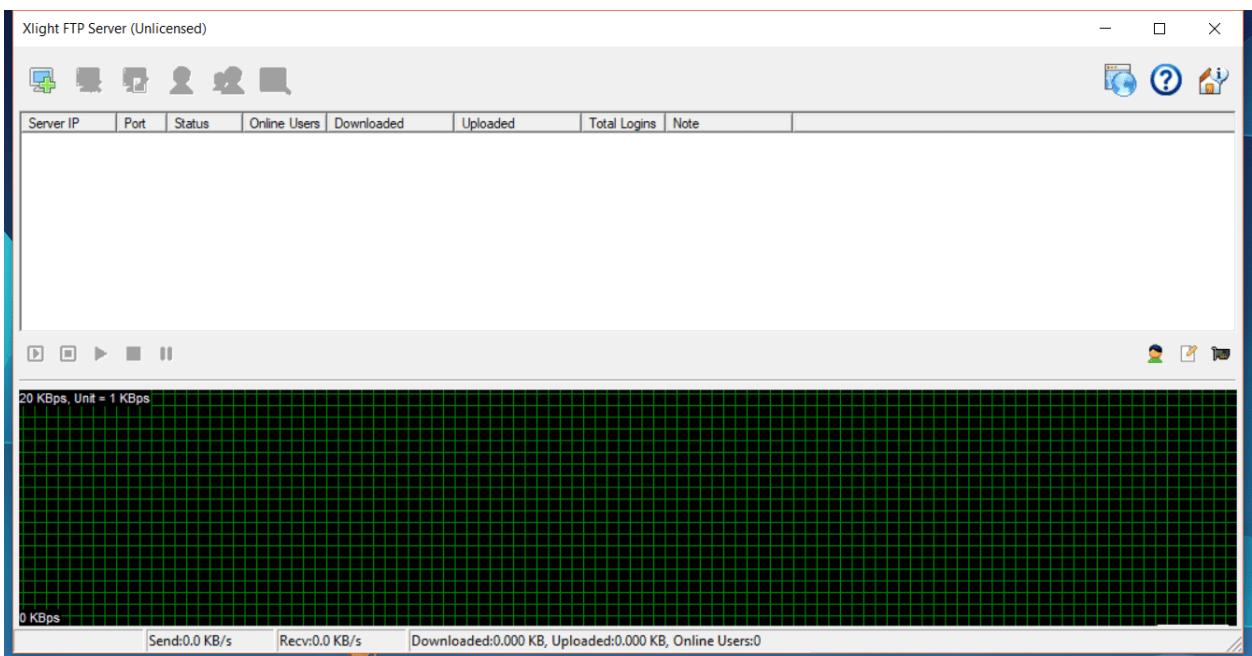


Рисунок 1.2 — Скриншот программы FTP-сервер Xlight

Достоинства FTP-сервер Xlight:

- Поддерживает безопасную передачу файлов;
- Функция удаленного администрирования;
- Поддерживает несколько подключений одновременно;

Недостатки FTP-сервер Xlight:

- Сложнее использовать для новичков FTP;
- Может быть сложным в настройке;

### 5.1.3. Complete FTP

Complete FTP — еще один бесплатный Windows FTP-сервер, который поддерживает FTP и FTPS. Эта программа имеет полный графический интерфейс пользователя и очень проста в использовании. Сам интерфейс до-

вольно прост, но все настройки скрыты в боковом меню и просты для доступа. Интерфейс Complete FTP изображен на рисунке 1.3:

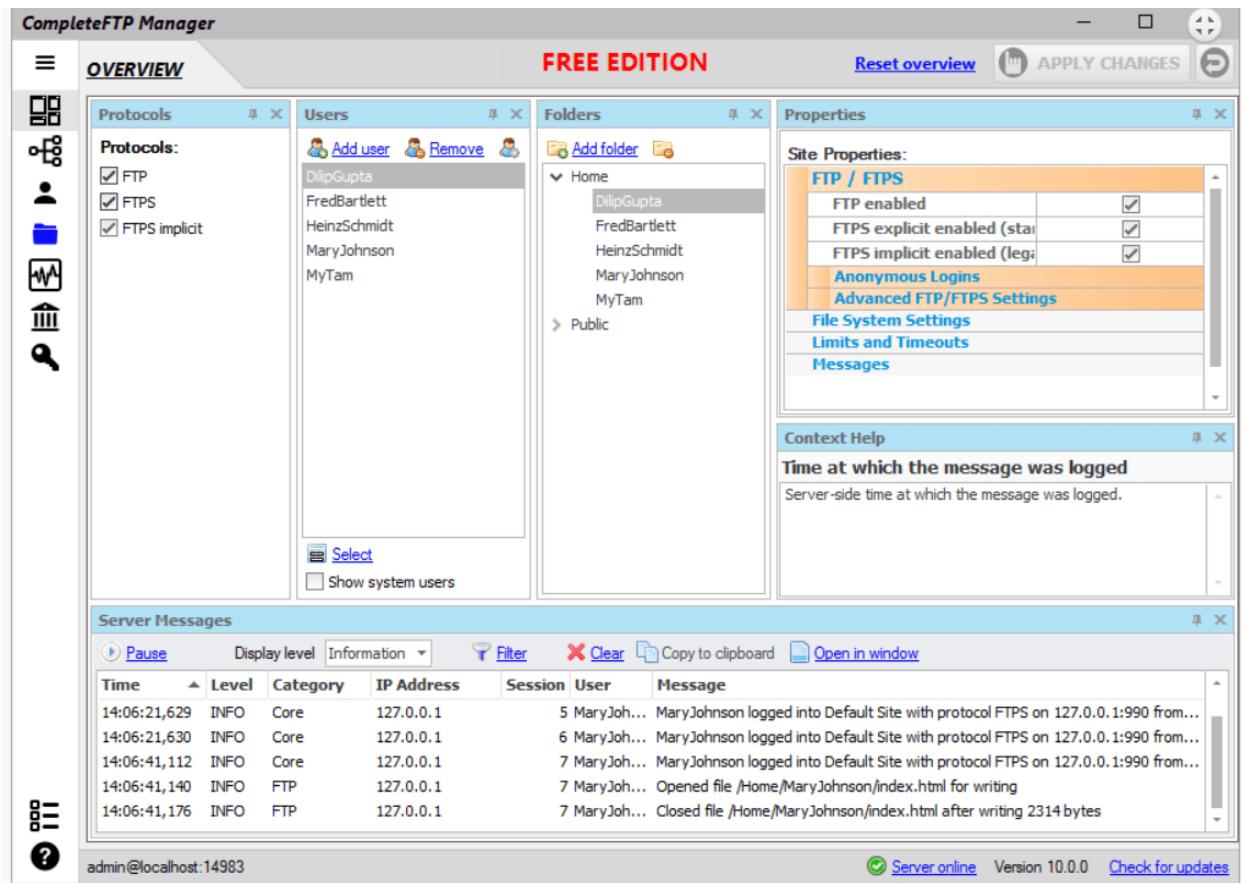


Рисунок 1.3 — Скриншот программы Complete FTP

### Достоинства Complete FTP:

- Поддерживает зашифрованные передачи файлов;
- Много вариантов настройки;

### Недостатки Complete FTP:

- Полное меню скрыто по умолчанию;
- Иногда возникают проблемы с производительностью;

## 5.2. Постановка задачи

После рассмотрения аналогов можно сказать, что все они обладают большим количеством функций, которые невозможно реализовать в курсовом проекте за данный период времени. Поэтому были выбраны несколько ключевых возможностей, которые будут выполнены в рамках одного семестра:

- Программа должна иметь удобный пользовательский интерфейс с необходимыми пунктами меню;
- В программе должна быть предусмотрена возможность хранения информации;

- Программа должна поддерживать обмен файлами;
- В программе должна быть реализована работа с протоколом FTP;
- Должна быть реализована возможность авторизации пользователей;

В качестве языка программирования выбран C++, по причине быстрой производительности, требуемой для обработки большого количества объектов и наличия опыта в использования данного языка. В качестве реализация графического интерфейса используется фреймворк Qt, основанный на языке C++. Qt обладает большим количеством базовых классов, которые позволяют создавать собственные классы для реализации графического интерфейса, удобной системой общения между виджетами приложения с помощью системы сигналов и слотов и хорошей документацией, позволяющей в быстрые сроки разбираться в устройстве Qt.

Данный список средств позволяет реализовать все задачи, выбранные для курсового проекта.

## **2 СИСТЕМНОЕ ПРОГРАММИРОВАНИЕ**

После определения требований к функционалу разрабатываемого приложения его следует разбить на функциональные блоки. Такой подход упростит понимание проекта, позволит устраниТЬ проблемы в архитектуре, обеспечит гибкость и масштабируемость программного продукта в будущем путем добавления новых блоков.

### **2.1 Модуль работы с приложением**

Модуль работы с приложением необходим для взаимодействия пользователя с приложением и является неотъемлемой его частью. Тут осуществляется весь необходимый функционал и возможности:

- Установление соединения с сервером;
- Отправка управляющих команд;
- Получение ответов;
- Отправка файлов и создание каталогов;
- Получение файлов и каталогов;
- Просмотр каталогов на сервере;

### **2.2 Модуль пользовательского интерфейса**

Модуль пользовательского интерфейса предназначен для взаимодействия пользователя с приложением, основываясь на представлении и визуализации данных.

Для разрабатываемого проекта создается простейший пользовательский интерфейс с помощью фреймворка Qt. Данный пользовательский интерфейс представляет собой набор пунктов меню, с которыми может взаимодействовать пользователь.

### **2.3 Модуль чтения и записи данных**

Модуль чтения и записи данных необходим для взаимодействия с данными, которые хранят приложение, что очень важно при работе с проектом.

Данный модуль должен отвечать за считывание информации от пользователя при авторизации и оправки запросов, её записи в используемую область памяти для дальнейшей работы, а также считывание и запись всей другой необходимой информации по мере работы с приложением.

### **2.4 Модуль преобразования данных**

Модуль преобразования данных необходим для преобразования данных, считываемых от пользователя или из приложения для корректной рабо-

ты с ними.

Очень часто одни и те же данные используются для разных целей, например, считанное время, может сравниваться с другими данными в формате QTime или же выводиться на экран в формате QString, поэтому для корректной работы и отображения этих данных необходимо их преобразовать к нужному типу.

## **2.5 Модуль сети**

Модуль сети предназначен для подключения пользователя к серверу и необходим для их взаимодействия в реальном времени.

Данный модуль должен отвечать за передачу файлов по протоколу FTP между пользователям и сервером, отслеживать активность клиента в приложении.

## **3 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ**

В данном разделе описывается функционирование и структура разрабатываемого приложения.

### **Описание классов приложения**

#### **3.1 MainWindow**

Данный класс является графическим представлением программы, он представляет собой окно, в котором происходит работа в приложении.

Поля:

- User \*user - пользователь работающий в приложении;
- Server server - объект сервера для его инициализации;
- make\_dir \*make - объект для создания нового каталога;
- void clicked\_button - метод который обрабатывает сигнал с кнопок перехода по папкам;
- void on\_add\_file\_clicked - срабатывает при нажатии на кнопку добавления файла;
- void on\_back\_clicked - метод срабатывает при нажатии на клавишу назад;
- void on\_delete\_file\_clicked - метод отвечает за удаления файла и обработку нажатия клавиши удаления;
- void on\_create\_folder\_clicked - метод отвечающий за создание папки;
- void on\_action\_ip\_triggered - метод отвечающий за установку ip адреса сервера;

#### **3.2 file**

Структура содержащая информацию о файле.

Поля:

- QString name - содержит имя файла;
- QString type - содержит тип файла (директория, ссылка, регулярный файл);

#### **3.3 Dir**

Данный класс отвечает за хранение и получение информации о директории.

Поля:

- std::vector<struct file> all\_files - хранит информацию о всех файлах в текущем каталоге;

- int num - количество файлов в текущем каталоге;
- void dirWalk - принимает char\* path - путь к директории. Отвечает за рекурсивный обход директории;
  - void current\_dir - принимает char\* path - путь к директории. Отвечает за обход директории;
  - int get\_num - возвращает количество файлов в текущем каталоге;
  - void set\_empty() - обнуляет счетчик количества файлов в директории и отчищает массив файлов;
  - std::vector<struct file> get\_all\_files - возвращает массив содержащий информацию о все файлах;

### 3.4 User

Данный класс отвечает за хранение о работающем пользователе.

Поля:

- QString current\_dir - строка хранящая путь к текущей директории;
- int id - хранит id пользователя;
- Dir\* all\_files - содержит информацию о всех файлах в текущем каталоге;
- int get\_id - возвращает id пользователя;
- void set\_id - принимает int id - id пользователя. Устанавливает id пользователя;
- QString get\_current\_dir - возвращает путь к текущему каталогу.
- Dir\* get\_all\_files - возвращает объект содержащий информацию о всех файлах в текущем каталоге;
- void set\_current\_dir - принимает QString current\_dir - путь к каталогу. Устанавливает текущую директорию;

### 3.5 Server

Данный класс инициализирует сервер для управления входящими соединениями, а так же обрабатывает входящие команды.

Поля:

- int s20 - содержит значения сокета для 20 порта;
- int s21 - содержит значения сокета для 21 порта;
- int num\_of\_users21 - содержит количество пользователей 21 порта;
- int sa20 - файловый дескриптор для 20 порта;
- int sa21[10] - массив файловых дескрипторов для 21 порта;
- char buffer[10][1024] - буфер для чтения и отправки данных;
- struct sockaddr\_in channel - структура содержащая настройки сокета;
- pthread\_t tid - id потока;
- int len\_of\_path - количество подкаталогов в пути к корню сервера;

- `QStack<int>indexes` - стек который хранит доступные id для пользователей;
- `QString dir` - содержит корневую директорию сервера;
- `QString ip` - содержит ip вида \* , \* , \* , \* для команды перехода в пасивный режим;
- `User *user` - объект хранящий информацию о пользователе;
- `static void* start_server21` - принимает `void *s` - указатель на объект `Server`. Функция ожидания запроса на 21 порту;
- `static void* new_connection21` - принимает `void *s` - указатель на объект `Server`. Функция ожидает входных данных от клиента на 21 порту;
- `int menu` - функция обрабатывает команды пришедшие от клиента;
- `void set_ip` - принимает `QString ip` - ip адрес сервера. Устанавливает ip адрес сервера;
- `void set_length_of_path` - принимает `int len`. Устанавливает количество подкаталогов в пути к корню сервера;

## **4 РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ**

### **4.1 Выделение ключевых процедур**

Одна из ключевых возможностей сервера заключается в установлении соединения с пользователем для которой требуются следующие процедуры и конструкторы: конструктор класса `Server`, процедура ожидания подключения, и процедура ожидания управляющей команды. Так же ключевыми процедурами утилиты многопоточный FTP-сервер можно назвать процедуры обработки запросов клиента, получения файлов в текущем каталоге пользователя.

#### **4.1.1 Конструктор класса Server**

Конструктор принимает один параметр - `path` корневую директорию пользователя. Для сервера устанавливается корневая директория. После в стек `index` заносятся все возможные `id` пользователей. Далее устанавливается параметры для 20 порта(порт передачи данных): `ip` адрес, порт, и семейство протоколов. Потом создается сокет с семейством протоколов `AF_INET`, режим передачи - поток бит и протоколом `TCP`. Для этого сокета устанавливаются настройки повторного использования порта и `ip` адреса. Это делается для избежания ошибки которая может возникнуть если на 20 порту уже есть работающая программа. После настройки сокета он привязывается к `ip` адресу и порту и далее устанавливается прослушивания порта. Те же действия происходят и для 21 порта(порт управляющих команд). После создания и настройки сокета на 20 порту создается новый поток в котором вызывается процедура ожидания соединения с сокетом `start_server21`.

#### **4.1.2 Ожидание соединения с сокетом start\_server21**

Процедура принимает объект класса `Server` для доступа к его полям. Запускается бесконечный цикл. В цикле с помощью функции `accept` ожидается подключения. После установления соединения в массив дескрипторов по индексу взятым из стека `index` объекта `Server` заносится дескриптор который вернула функция `accept` после установления соединения. Далее на этот дескриптор отправляется ответ со статусом 200 который говорит об успешном соединении с сервером. В конце создается новый поток который вызывает процедуру ожидания управляющей команды `new_connection21`.

#### **4.1.3 Ожидание соединения с сокетом new\_connection21**

Процедура принимает объект класса `Server` для доступа к его полям. После создается объект класса `User` - новый пользователь. Этому объекту присваивается корневая директория сервера и `id` который был получений из стека `index`. Запускается бесконечный цикл. В цикле происходит чтения из

дескриптора который берется из массива дескрипторов по индексу который равен id пользователя. Если количество прочитанных байтов больше нуля, то есть пришла команда, вызывается функция обработки команд пользователя menu в которую передается полученная команда.

#### 4.1.4 Обработки команд пользователя menu

Процедура принимает один параметр - управляющую команду. В зависимости от управляющей команды происходят определенные действия.

Команда USER в ответ отправляет ответ со статус кодом 230 сигнализирующий об успешном входе.

Команда SYST в ответ отправляет ответ который содержит информацию о системе вместе с кодом 215 который говорит об успешной обработке команды SYST.

Команда FEAT отправляет в ответе команды которые сервер может обработать и статус код 211 который говорит об успешной обработке команды FEAT.

Команда PWD формирует ответ из текущего пути пользователя на сервере и статус код 257 который говорит об успешной обработке команды PWD. Алгоритм формирования заключается в разделении текущего каталога, который включает в себя так же системные папки кроме папок доступных на сервере, по символу '/'. Если длина разделенного пути равна количеству системных папок, то в ответе отправляется символ '/' который говорит о том что пользователь находится в корневом каталоге сервера. Иначе формируется путь исключающий системные папки и отправляется в ответе.

Команда TYPE отправляет в ответ 230 статус код.

Команда PASV отправляет в структуре ответа ip адрес и порт передачи данных в виде \*\*\*,\*\*\*,\*\*\*,\*\*\*,port,port и статус код 227.

Команда LIST ожидает подключения к 20 порту после чего вызывает функции обхода текущей директории, который является методом класса User. Отправляется ответ со статус кодом 150, который говорит о начале передачи данных, в порт управляющих команд. Далее запускается цикл который формирует информацию о файле. Информация о файле включает тип файла, дата последнего изменения, права доступа, владелец файла. Вся информация о файле передается в дескриптор который был получен при подключении пользователя к 20 порту. После в порт управляющих команд отправляется ответ включающий количество найденных файлов и статус код 226 который говорит об успешной передачи файлов.

Команда MKD получает текущий путь пользователя на сервере и в этой директории создается папка с именем переданным вместе с командой. В ответ отправляется путь к созданной папке начиная с корня сервера и статус код 257 сигнализирующий об успешной обработке команды.

Команда CWD получает текущий путь пользователя на сервере, проверяет папку которая была запрошена. Если запрошенная папка была «..», то

вызывается функция menu с командой «CDUP», иначе происходит проверка существует ли такая директория и если существует, формируется путь к этой директории и отправляется в качестве ответа со статус кодом 227, иначе отправляется статус код 550 и сообщения о том что данная папка не была найдена.

Команда CDUP проверяет находится ли пользователь в корневой директории и если находится, отправляется ответ со статусом 550 и сообщениям о отсутствии папки. Если пользователь не находится в корневой директории то функции формирует путь исключая последнюю директорию в пути пользователя на сервере и отправляет статус код 250 который говорит об успешной обработке команды

Команда RETR проверяет является ли запрошенный файл директорией, в случае если является, с помощью встроенной утилиты «tar» создается архив и открывается в бинарном режиме для чтения. Если файл не является папкой, то файл открывается так же в бинарном режиме для чтения, если функция fopen отвечающая за открытие файла возвращает NULL значит не существует запрошенной папки или файла и отправляется статус код 450 с сообщением об отсутствии файла или папки, иначе происходит ожидания подключения к 20 порту после чего отправляется ответ со статусом 150 и запускается цикл который читает файл и отправляет его в порт данных, после окончания чтения отправляется ответ со статусом 226 и сообщением об успешной отправке файла.

Команда STOR получает текущий путь пользователя на сервер, после создает файл и с именем переданным вместе с командой и ожидает подключения к 20 порту. Как только соединение с 20 портом было установлено отправляется сообщение о начале передачи данных со статусом 150 и функция читает данные из дескриптора 20 порта и записывает прочитанные байты в созданный файл. После завершения чтения отправляется сообщение об успешном получении файла со статусом 226.

Команда QUIT отправляет ответ об успешном закрытии соединения со статусом 221 и функция завершается с кодом -1 который сигнализирует о требовании на закрытие соединения.

Если получения не соответствует ни одной из перечисленных выше команд то отправляется сообщения о том что указанная команда не поддерживается со статусом 502.

#### **4.1.5 Получения файлов в текущем каталоге пользователя current\_dir**

Метод класса `Dir` на вход получает путь к директории. Создается объект структуры `DIR` и объект структуры `dirrent`. Далее с помощью функции `opendir` открывается текущая директория, функция возвращает `DIR`, после в цикле происходит чтение директории, в цикле пропускаются директории «..», «..», «.DS\_Store», формируется путь к прочитанному файлу и в вектор зано-

ситься структура `file` содержащая путь к папке и тип файла. В конце итерации цикла переменная отвечающая за количество файлов директории увеличивается на единицу. В конце функция закрывает директорию и завершается.

## 5 ПРОГРАММА И МЕТОДИКА ИСПЫТАНИЙ

Тестирование — это наблюдение за работой приложения в разных искусственно созданных ситуациях. Данные, полученные в ходе тестирования, важны при планировании последующей стратегии развития приложения. Это своего рода диагностика, которая влияет на многие дальнейшие действия.

### 5.1 Тестирование работы приложения со стороны сервера.

Данное приложение работает по клиент-серверной модели, со стороны сервера программа может добавлять, удалять и создавать различные папки.

Пример добавления файла на сервер представлен на рисунках 5.1.1, 5.1.2:

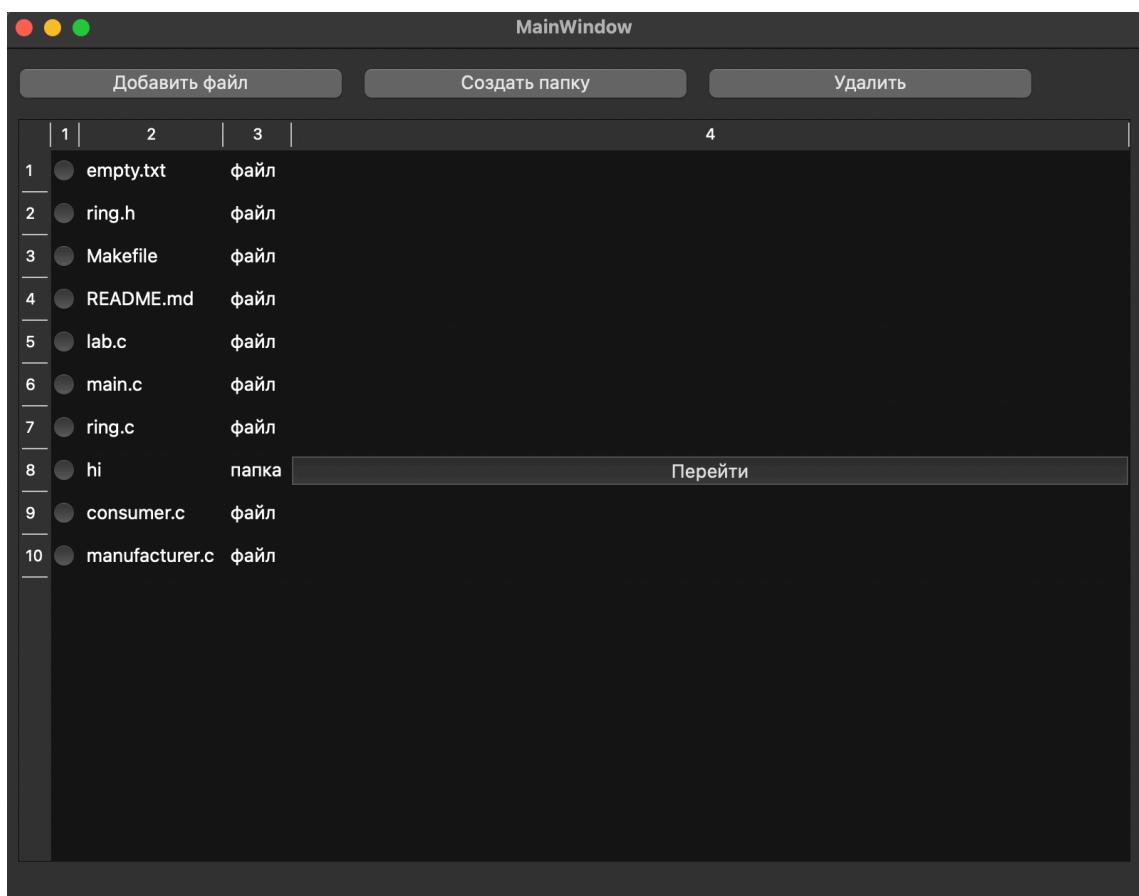


Рисунок 5.1.1 — Добавление файла на сервер

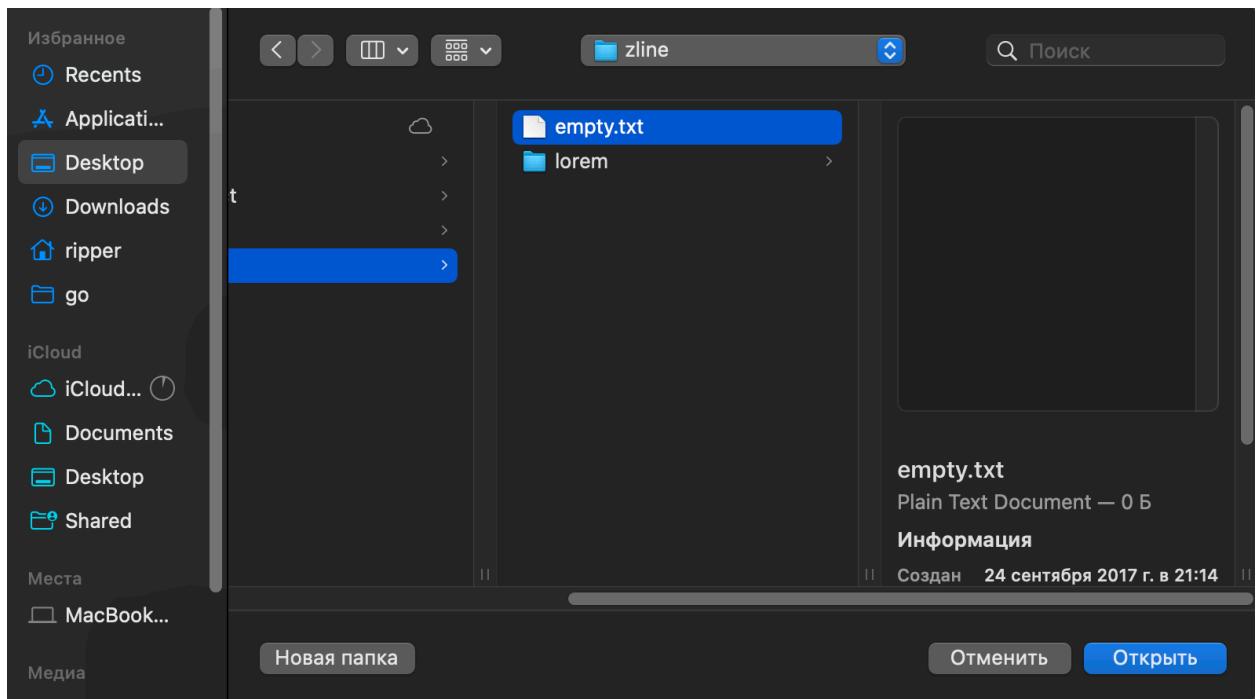


Рисунок 5.1.2 — Отображение добавленного файла на сервере

Создание папки представлено на рисунке 5.1.3:

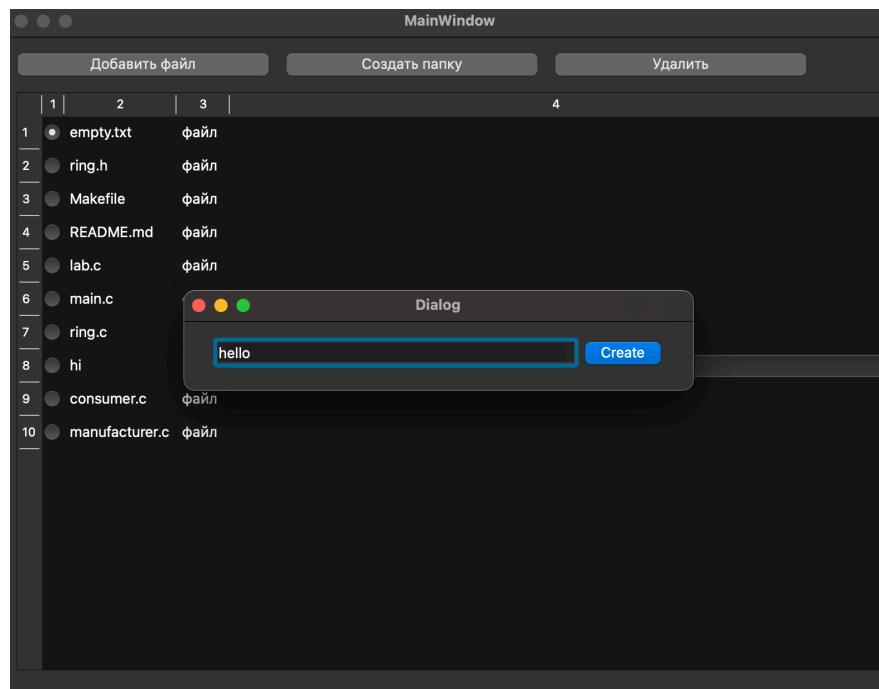


Рисунок 5.1.3 — Создание папки на сервере

## 5.2 Тестирование приложения со стороны клиента.

Пример подключения к серверу и просмотра каталога представлен на рисунке 5.2.1:

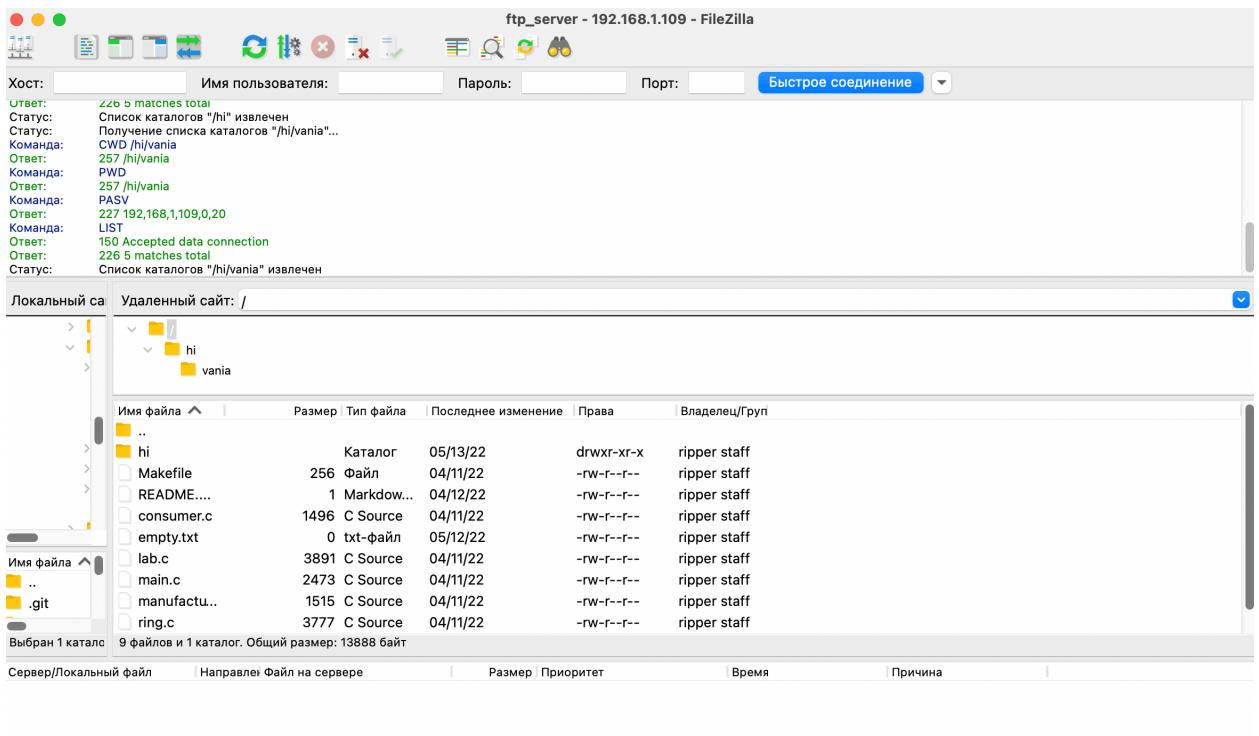


Рисунок 5.2.1 — Подключение и просмотр файлов на сервере

Скачивание несуществующего файла с сервера представлено на рисунке 5.2.2:

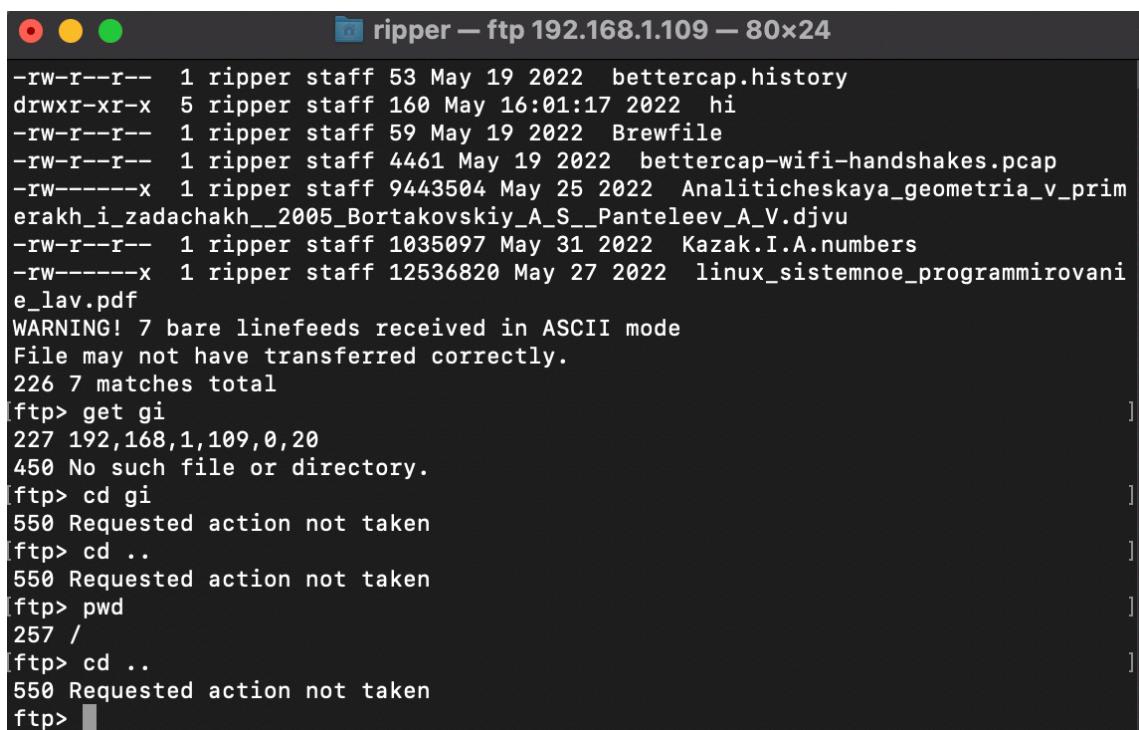
```
ripper — ftp 192.168.1.109 — 80x24
230 Ok
ftp> passive
Passive mode on.
ftp> binary
230 Ok
ftp> ls
227 192,168,1,109,0,20
150 Accepted data connection
-rw-r--r-- 1 ripper staff 53 May 19 2022 bettercap.history
drwxr-xr-x 5 ripper staff 160 May 16:01:17 2022 hi
-rw-r--r-- 1 ripper staff 59 May 19 2022 Brewfile
-rw-r--r-- 1 ripper staff 4461 May 19 2022 bettercap-wifi-handshakes.pcap
-rw-----x 1 ripper staff 9443504 May 25 2022 Analiticheskaya_geometriya_v_primenenii_i_zadachakh__2005_Bortakovskiy_A_S__Panteleev_A_V.djvu
-rw-r--r-- 1 ripper staff 1035097 May 31 2022 Kazak.I.A.numbers
-rw-----x 1 ripper staff 12536820 May 27 2022 linux_sistemnoe_programmirovaniye_lav.pdf
WARNING! 7 bare linefeeds received in ASCII mode
File may not have transferred correctly.
226 7 matches total
ftp> get gi
227 192,168,1,109,0,20
450 No such file or directory.
ftp>
```

Рисунок 5.2.2 — Скачивание несуществующего файла с сервера

Переход в несуществующий каталог представлен на рисунках 5.2.3, 5.2.4:

```
Passive mode on.  
[ftp> binary  
230 Ok  
[ftp> ls  
227 192,168,1,109,0,20  
150 Accepted data connection  
-rw-r--r-- 1 ripper staff 53 May 19 2022 bettercap.history  
drwxr-xr-x 5 ripper staff 160 May 16:01:17 2022 hi  
-rw-r--r-- 1 ripper staff 59 May 19 2022 Brewfile  
-rw-r--r-- 1 ripper staff 4461 May 19 2022 bettercap-wifi-handshakes.pcap  
-rw-----x 1 ripper staff 9443504 May 25 2022 Analiticheskaya_geometria_v_prim  
erakh_i_zadachakh__2005_Bortakovskiy_A_S__Panteleev_A_V.djvu  
-rw-r--r-- 1 ripper staff 1035097 May 31 2022 Kazak.I.A.numbers  
-rw-----x 1 ripper staff 12536820 May 27 2022 linux_sistemnoe_programmirovani  
e_lav.pdf  
WARNING! 7 bare linefeeds received in ASCII mode  
File may not have transferred correctly.  
226 7 matches total  
[ftp> get gi  
227 192,168,1,109,0,20  
450 No such file or directory.  
[ftp> cd gi  
550 Requested action not taken  
ftp> ]
```

Рисунок 5.2.3 — Переход в несуществующий каталог



```
● ● ● ripper — ftp 192.168.1.109 — 80x24  
-rw-r--r-- 1 ripper staff 53 May 19 2022 bettercap.history  
drwxr-xr-x 5 ripper staff 160 May 16:01:17 2022 hi  
-rw-r--r-- 1 ripper staff 59 May 19 2022 Brewfile  
-rw-r--r-- 1 ripper staff 4461 May 19 2022 bettercap-wifi-handshakes.pcap  
-rw-----x 1 ripper staff 9443504 May 25 2022 Analiticheskaya_geometria_v_prim  
erakh_i_zadachakh__2005_Bortakovskiy_A_S__Panteleev_A_V.djvu  
-rw-r--r-- 1 ripper staff 1035097 May 31 2022 Kazak.I.A.numbers  
-rw-----x 1 ripper staff 12536820 May 27 2022 linux_sistemnoe_programmirovani  
e_lav.pdf  
WARNING! 7 bare linefeeds received in ASCII mode  
File may not have transferred correctly.  
226 7 matches total  
[ftp> get gi  
227 192,168,1,109,0,20  
450 No such file or directory.  
[ftp> cd gi  
550 Requested action not taken  
[ftp> cd ..  
550 Requested action not taken  
[ftp> pwd  
257 /  
[ftp> cd ..  
550 Requested action not taken  
ftp> ]
```

Рисунок 5.2.4 — Переход на каталог назад из корня

## 6 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

Для запуска программы необходимо открыть файл «ftp\_server.app». После этого откроется окно программы (рисунок 6.1).

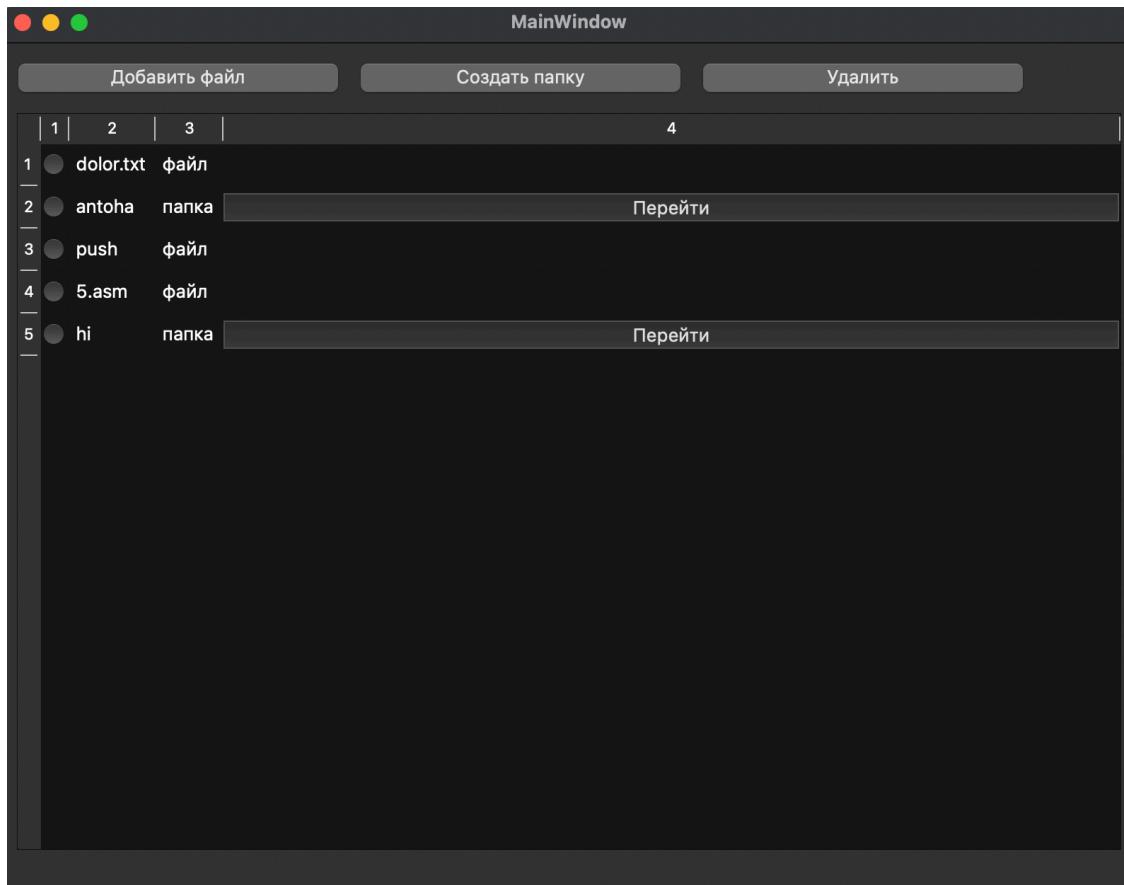


Рисунок 6.1 — окно программы

Для того, чтобы начать пользоваться приложением, необходимо ввести ip адрес сервера, для этого следует зайти в «Настройки» -> «Задать ip» (рисунок 6.2).

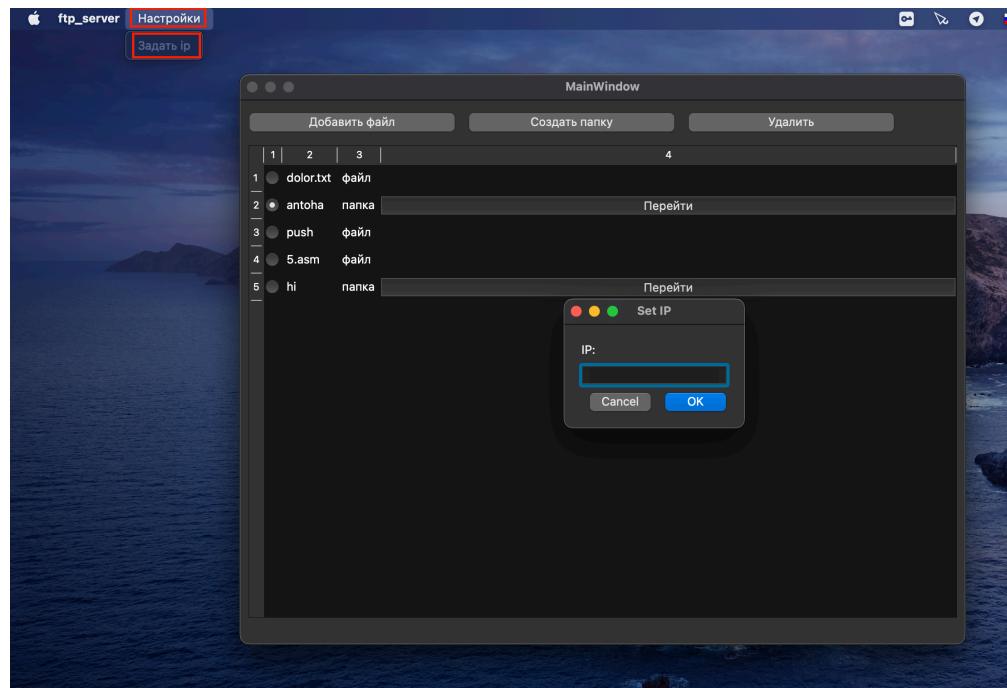


Рисунок 6.2 — настройка ip адреса

Для добавления файла на сервер нажмите кнопку «Добавить файл» и выберете файлы которые хотите добавить (рисунок 6.3):

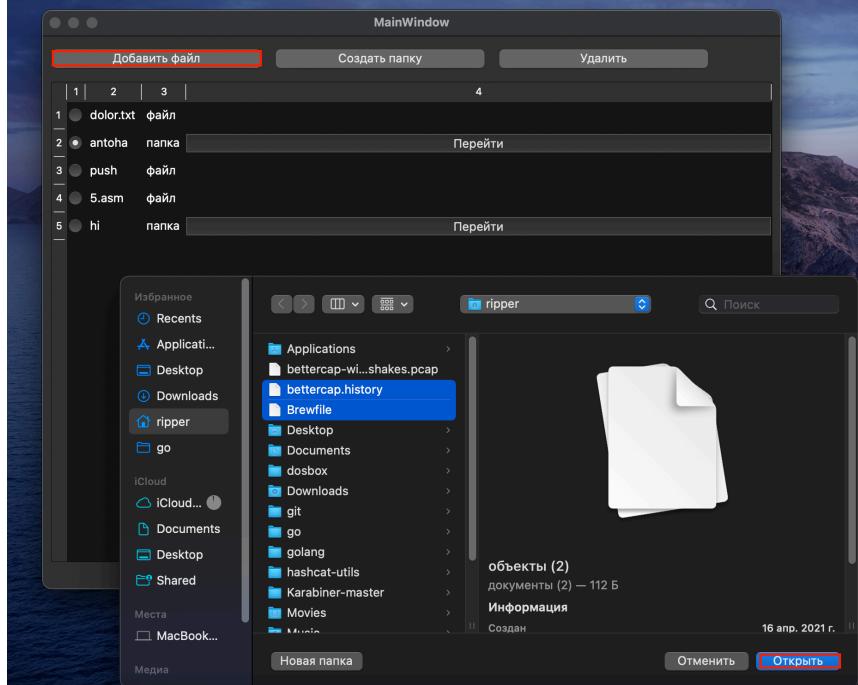


Рисунок 6.3 — Добавление файла на сервер

Для создания папки нажмите кнопку «Создать папку» и задайте имя папки (рисунок 6.4):

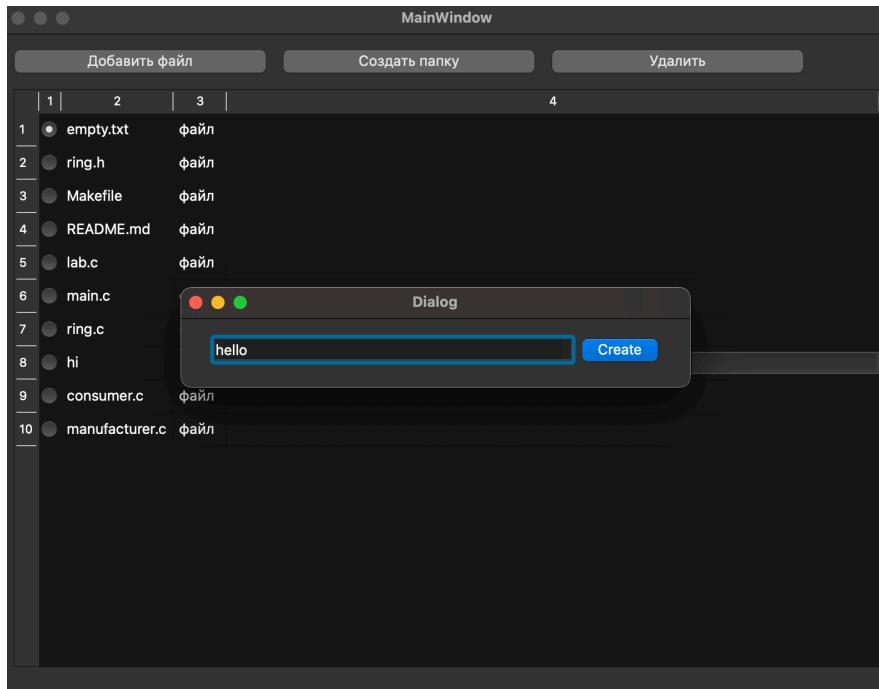


Рисунок 6.4 — Создание папки на сервере

Для удаления файла либо папки выберете файл/папку с помощью кнопки выбора находящуюся в слева от имени файла/папки и нажмите кнопку «Удалить» (рисунок 6.5):

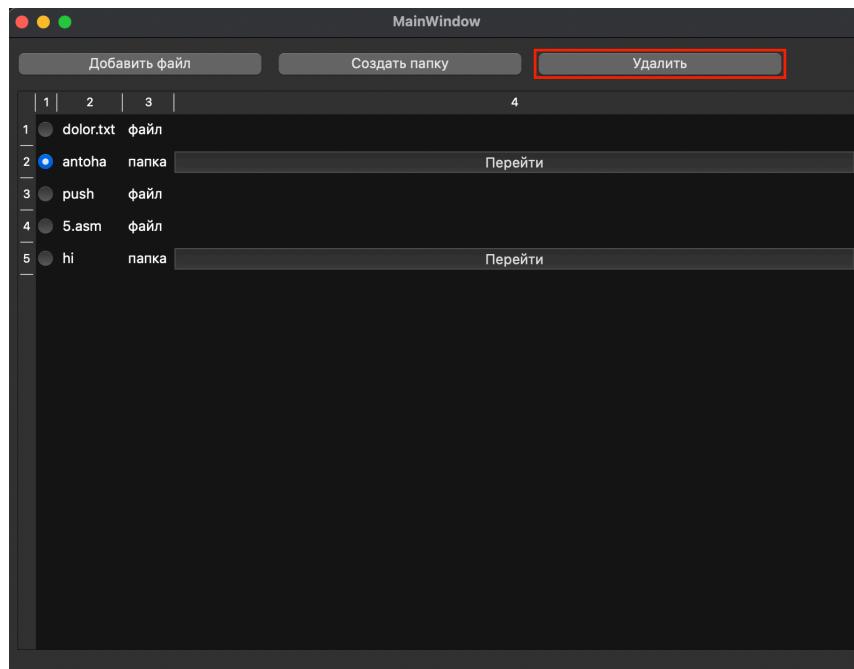


Рисунок 6.5 — Удаление файла/папки с сервера

Для перехода в папку нажмите на кнопку «перейти» справа от имени папки (рисунок 6.6):

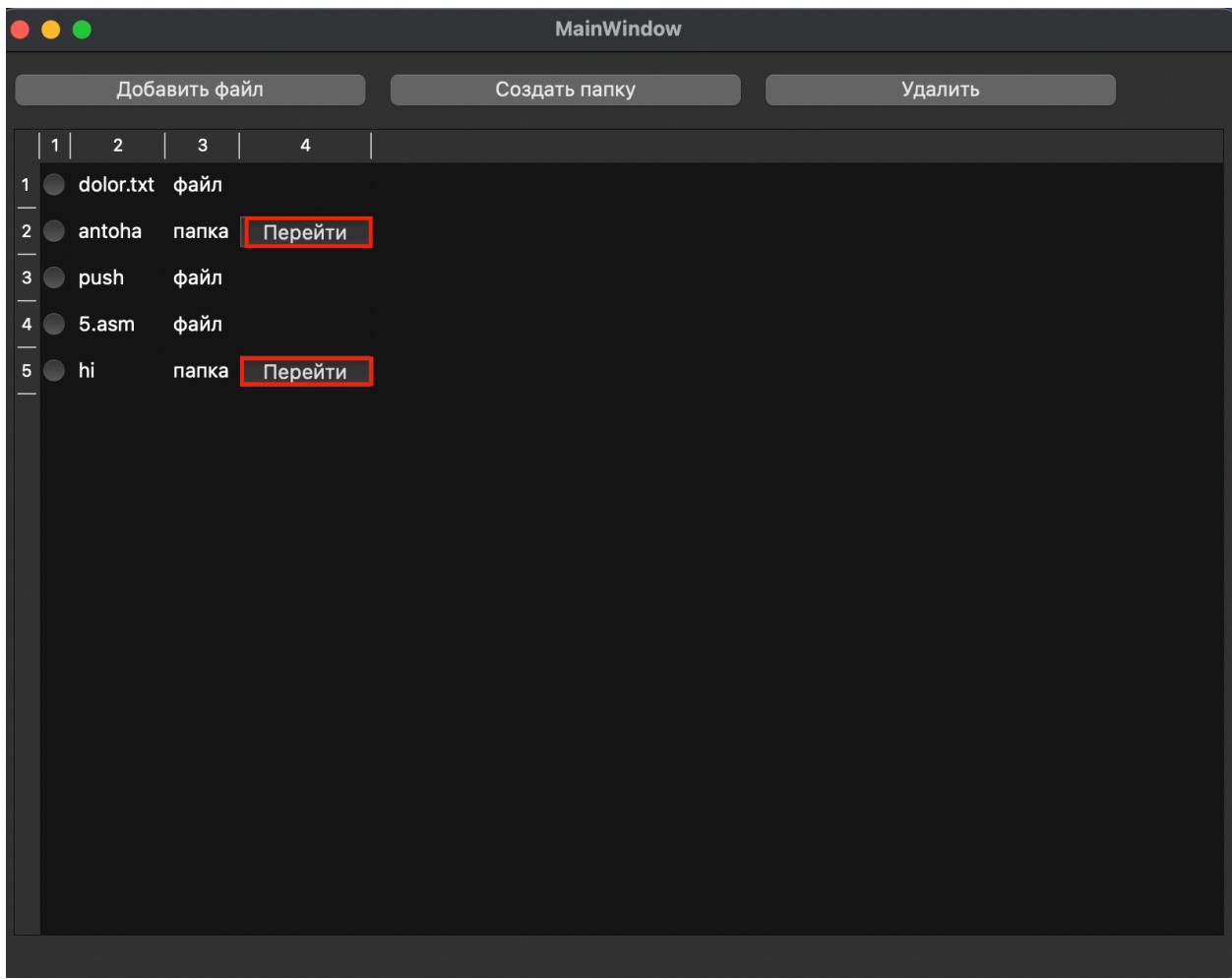


Рисунок 6.6 — Переход в каталог

Для перехода в предыдущую директорию нажмите кнопку «Назад». Кнопки «Назад» в корне сервера нету (рисунок 6.7):

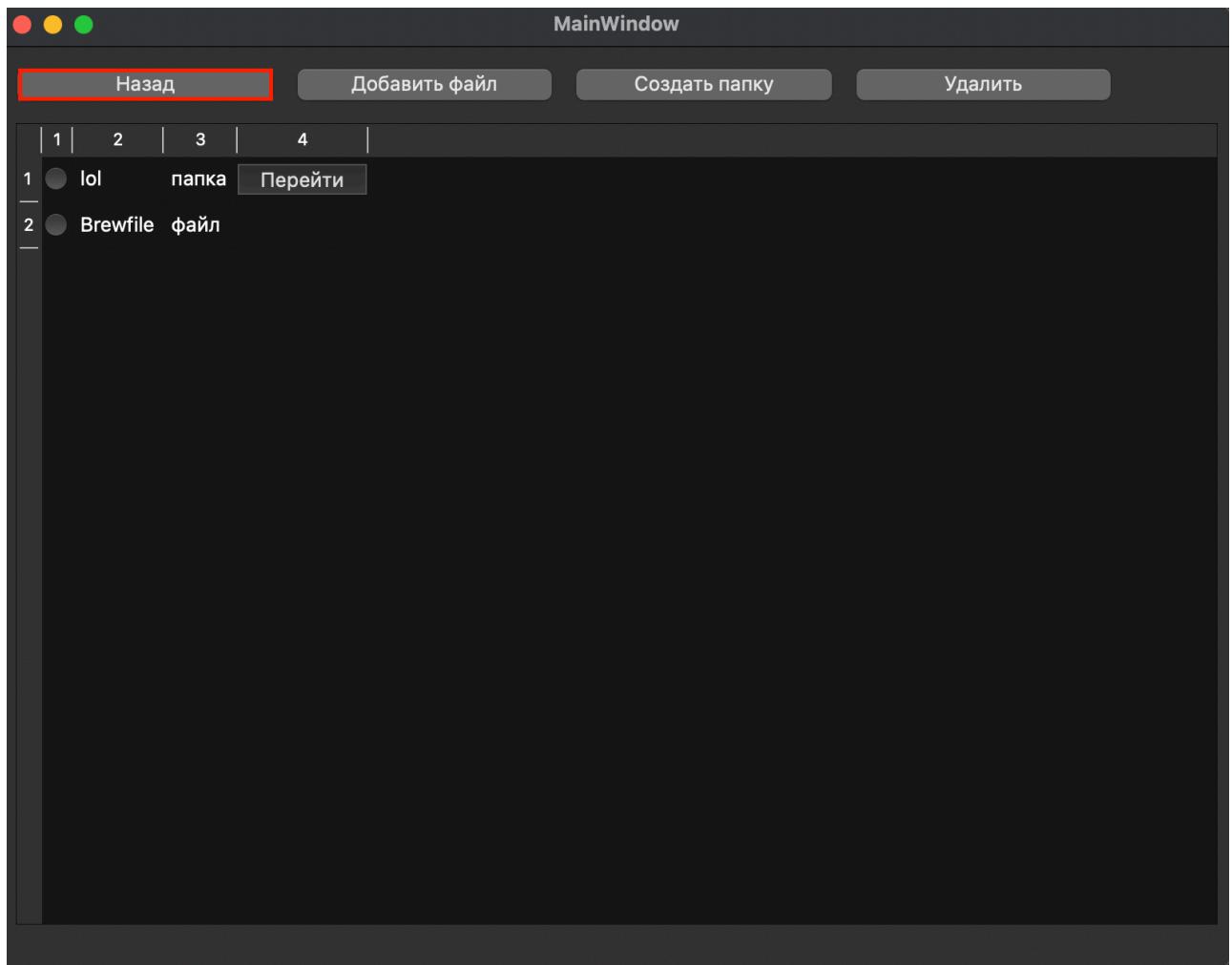


Рисунок 6.7 — Выход из каталога

## **ЗАКЛЮЧЕНИЕ**

В результате работы над данным курсовым проектом было разработано работоспособное приложение со своим набором функций и графическим интерфейсом. Данный курсовой проект был разработан в соответствии с поставленными задачами, весь функционал был реализован в полном объеме.

В ходе разработки были углублены знания языка программирования C++ и в области объектно-ориентированного программирования, а также получен опыт работы с фреймворком Qt и с графическим интерфейсом.

Работа была разделена на такие этапы, как анализ существующих аналогов, литературных источников, постановка требований к проектируемому программному продукту, системное и функциональное проектирование, конструирование программного продукта, разработка программных модулей и тестирование проекта. После последовательного выполнения вышеперечисленных этапов разработки было получено исправно работающее приложение.

В дальнейшем планируется усовершенствование текущего функционала приложения, путем улучшения графического интерфейса, добавления новых функций и модулей, а также добавления возможности работы под различными системами.

## **СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ**

- [1] Кернigan Б. Язык программирования С/ 4-е издание М.:Питер, 2004.  
– 923 с.
- [2] Лав Р. Системное программирование на Linux/ 2-е издание 2014.
- [3] Таненбаум Э. Компьютерные сети 5-е издание 2019.
- [4] IBM [Электронный ресурс] Режим доступа: <https://www.ibm.com/docs/ru/aix/7.1?topic=protocol-tcpip-protocols>
- [5] ВЭБ-ШПАРГАЛКА [Электронный ресурс] Режим доступа:  
<https://web-shpargalka.ru/ftp-server-chto-jeto.php>

**ПРИЛОЖЕНИЕ А**

(обязательное)

**Схема структурная**

**ПРИЛОЖЕНИЕ Б**  
(обязательное)

**Диаграмма классов**

**ПРИЛОЖЕНИЕ В**

(обязательное)

**Ведомость документов**