**Міністерство освіти і науки, молоді та спорту України  
Національний технічний університет України**

**"Київський політехнічний інститут"**

**Кафедра обчислювальної техніки**

## ПОЯСНЮВАЛЬНА ЗАПИСКА

з курсової роботи з навчальної дисципліни "Системне програмування"

Тема. FTP-клієнт

# **Захищено з оцінкою**

# **Керівник роботи**

\_\_\_\_\_ Стіренко Сергій Григорович

“\_\_” грудня 2011 року

# **Допущено до захисту**

# **Керівник роботи**

\_\_\_\_\_ Стіренко Сергій Григорович

“\_\_” грудня 2011 року

|  |
| --- |
| ***Роботу виконала***  Студентка групи ІО-91  \_\_\_\_\_Косейкіна Ганна Сергіївна  “\_\_” грудня 2011 року |

Київ 2011 р.

Зміст

Завдання на курсову роботу…………………………………………………….……….3

Вступ………………………………………………………………………………….…..4

1. Блок-схеми виконання операцій з описанням алгоритму……………………….….6

2. Інтерфейс програми…………………………………………………………………...9

3. Інструкція до програмного проекту…………………………………………….……9

4. Перевірка програмного проекту та контрольні приклади………...………………12

5. Висновки………………………………………………………………………….…..15

6. Список використаної літератури…………………………………………………....16

Додаток………………………………………………………………………………….17

**Національний технічний університет України „КПІ”**

**Факультет інформатики і обчислювальної техніки**

**Кафедра обчислювальної техніки**

## ЗАВДАННЯ

на курсову роботу з навчальної дисципліни "Системне програмування"

Тема роботи. FTP-клієнт

Функціональні вимоги до програми: Можливість завантажувати та скачувати файл з FTP-серверу, продивлятися вміст серверу, створення та видалення тек, перейменування файлу. Програма має працювати з сервером напряму, в активному режимі, без використання бібліотек. Програма має забезпечувати додатковий захист для FTP-серверу та зменшувати навантаження на сервер.

Вимоги до форматів вхідних і вихідних даних програми: Файли будь-якого формату, теки.

Вимоги до програмних та апаратних інструментів: Використання мови C#, вбудовані бібліотеки мови, робота з сервером напряму.

Рекомендована література: Герберт Шилдт C# 4.0: полное руководство = C# 4.0 The Complete Reference — М.: [«Вильямс»](http://ru.wikipedia.org/w/index.php?title=%D0%92%D0%B8%D0%BB%D1%8C%D1%8F%D0%BC%D1%81_(%D0%B8%D0%B7%D0%B4%D0%B0%D1%82%D0%B5%D0%BB%D1%8C%D1%81%D1%82%D0%B2%D0%BE)&action=edit&redlink=1), 2010. — С. 1056. — [ISBN 978-5-8459-1684-6](http://ru.wikipedia.org/wiki/%D0%A1%D0%BB%D1%83%D0%B6%D0%B5%D0%B1%D0%BD%D0%B0%D1%8F:BookSources/9785845916846);

Джон Скит C#: программирование для профессионалов, 2-е издание = C# in Depth, 2nd Edition — М.: [«Вильямс»](http://ru.wikipedia.org/w/index.php?title=%D0%92%D0%B8%D0%BB%D1%8C%D1%8F%D0%BC%D1%81_(%D0%B8%D0%B7%D0%B4%D0%B0%D1%82%D0%B5%D0%BB%D1%8C%D1%81%D1%82%D0%B2%D0%BE)&action=edit&redlink=1), 2011. — 544 с. — [ISBN 978-5-8459-1555-9](http://ru.wikipedia.org/wiki/%D0%A1%D0%BB%D1%83%D0%B6%D0%B5%D0%B1%D0%BD%D0%B0%D1%8F:BookSources/9785845915559);

http://msdn.microsoft.com/ru-ru/library/67ef8sbd.aspx

**Завдання на роботу видав Завдання на роботу прийняли**

**Керівник роботи Студенти групи ІО-91**

\_\_\_\_ Стіренко Сергій Григорович \_\_\_\_\_ Косейкіна Ганна Сергіївна

“\_\_”грудня 2011р. ” \_\_” грудня 2011 р.

**Вступ**

**FTP-клієнт** - [програма](http://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%B0) для спрощення доступу до [FTP](http://uk.wikipedia.org/wiki/FTP) [серверу](http://uk.wikipedia.org/wiki/%D0%A1%D0%B5%D1%80%D0%B2%D0%B5%D1%80). Залежно від призначення може або надавати користувачеві простий доступ до віддаленого FTP-сервера в режимі текстової [консолі](http://uk.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D1%81%D0%BE%D0%BB%D1%8C), беручи на себе тільки роботу з пересилання команд користувача і [файлів](http://uk.wikipedia.org/wiki/%D0%A4%D0%B0%D0%B9%D0%BB), або відображати файли на віддаленому сервері так, щоб вони були частиною файлової системи [комп'ютера](http://uk.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%27%D1%8E%D1%82%D0%B5%D1%80) користувача, або і те й інше. В останніх двох випадках FTP-клієнт бере на себе завдання інтерпретації дій користувача у команди протоколу FTP, тим самим даючи можливість використовувати протокол передачі файлів.

Приватними прикладами використання FTP-клієнта можуть бути:

* Публікація сторінок сайту на Інтернет-сервері Веб-розробником
* Завантаження музики, програм і будь-яких інших файлів даних звичайних користувачів [Інтернету](http://uk.wikipedia.org/wiki/%D0%86%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82).

У найпростішому для користувача (але при цьому найбільш комплексному) випадку FTP-клієнт являє собою [емулятор](http://uk.wikipedia.org/w/index.php?title=%D0%95%D0%BC%D1%83%D0%BB%D1%8F%D1%82%D0%BE%D1%80&action=edit&redlink=1) файлової системи, яка просто знаходиться на іншому комп'ютері. З цією файловою системою можна здійснювати всі звичні користувачеві дії: копіювати файли з серверу і на сервер, видаляти файли, створювати нові файли. В окремих випадках можливе також відкриття файлів - для перегляду, запуску програм, редагування. Необхідно враховувати, що відкриття файлу передбачає його попереднє скачування на комп'ютер користувача. Прикладами таких FTP-клієнтів можуть служити:

1. Інтернет-браузери (часто працюють в режимі «лише читання», тобто не дозволяють додавати файли на сервер)
2. Файлові менеджери, наприклад: Windows Explorer (Провідник), [Total Commander](http://uk.wikipedia.org/wiki/Total_Commander), [FAR](http://uk.wikipedia.org/w/index.php?title=FAR&action=edit&redlink=1), [Midnight Commander](http://uk.wikipedia.org/wiki/Midnight_Commander), [Krusader](http://uk.wikipedia.org/wiki/Krusader)
3. Спеціалізовані програми, наприклад: [FileZilla](http://uk.wikipedia.org/wiki/FileZilla)

Завдяки поширеності протоколу FTP, прості (з точки зору реалізації) FTP-клієнти є практично в кожній операційній системі. Однак використання цих клієнтів вимагає навичок використання консолі, а також знання команд [протоколу](http://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB) для спілкування з сервером.

Було обрано мову C#, оскільки вона розроблена корпорацією Microsoft спеціально для використання у операційних системах сімейства Windows для CLR - Common Language Runtime — «загального середовища виконання мов» — компоненту пакету [Microsoft](/wiki/Microsoft)[.NET Framework](/wiki/.NET_Framework), віртуальної машини, на якій виконуються всі [мови](/wiki/%D0%9C%D0%BE%D0%B2%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F) платформи .NET Framework. Тому ця мова має можливості для коректного виконання програм на Windows.

**1. Блок-схеми алгоритмів виконання операцій з поясненням**

1. Створення нової теки. Операція створення нової теки створює нову теку на FTP-сервері. Спочатку перевіряється чи є вірними адреса та ім’я створюваної теки. Якщо є якісь помилки, то програма видає вікно з помилкою, після чого можна ввести нову адресу та ім’я. Після коректного вводу програма створює нову теку.



Рисунок 1.1.Блок-схема алгоритму створення теки.

2. Видалення теки. Операція видалення теки видаляє обрану теку. Програма перевіряє чи обрано теку, після чого якщо обрано видаляє, якщо ні - видає помилку, після чого можна обрати теку. Також можна видаляти файли, алгоритм такий самий.



Рисунок 1.2.Блок-схема алгоритму видалення файлу або теки.

3. Перейменування файлу. Операція перейменування файлу перейменовує файл. Програма перевіряє чи коректно написані адреса та нове ім’я файлу, чи обрано файл. Після цього або перейменовує, або видає помилку.



Рисунок 1.3.Блок-схема алгоритму перейменовування файлу.

4. Скачування файлу. Операція скачування файлу скачує файл в обрану теку на жорсткому диску комп’ютера. Програма перевіряє чи правильно введені адреси та імена вхідного та вихідного файлів, після чого або скачує файл, або видає помилку.



Рисунок 1.4.Блок-схема алгоритму скачування файлу

5. Закачування файлу на сервер. Операція закачування файлу закачує файл на сервер. Програма перевіряє чи правильно введені адреси та імена файлів, після чого або закачує файл, або видає помилку.



Рисунок 1.5. Блок-схема закачування файлу

**2. Інтерфейс програми**

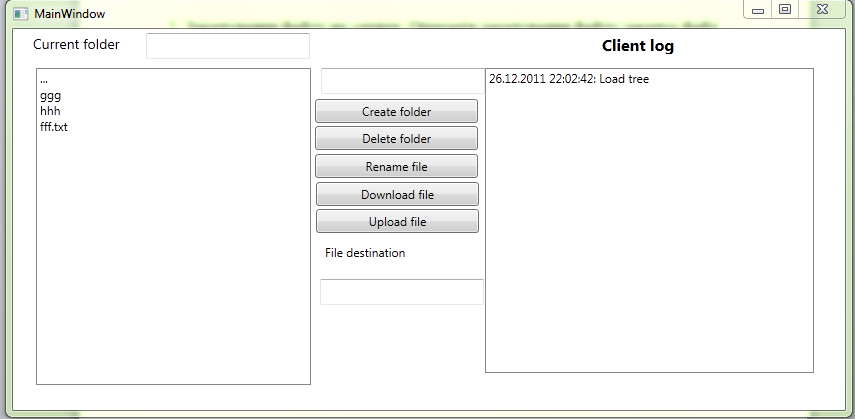


Рисунок 2.1. Інтерфейс програми

**3. Інструкція до програмного проекту**

Якщо ви бажаєте запустити програму для підтримки FTP-серверу, то необхідно запустити FTPProxyServer.exe. Цей сервер приєднується до FTP-серверу на тому ж комп’ютері, або на іншому. Після запуску з’являється вікно програми.

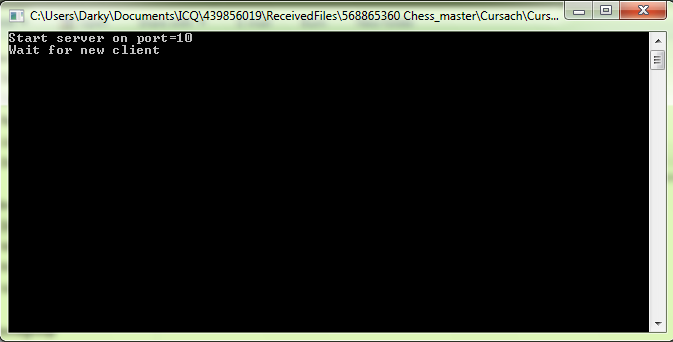


Рисунок 3.1.Вікно програми FTPProxyServer.

До нього приєднуються клієнти. Якщо ви бажаєте запустити програму-клієнт, то необхідно запустити ClientGUI.exe. Після цього необхідно ввести адресу комп’ютеру, на якому запущена програма-сервер та адресу самого FTP-серверу. Також потрібно ввести ім’я користувача та пароль.

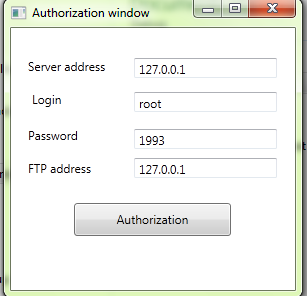


Рисунок 3.2.Вікно авторизації у ClientGUI.

Після коректних імені користувача та паролю з’являється вікно програми-клієнту.

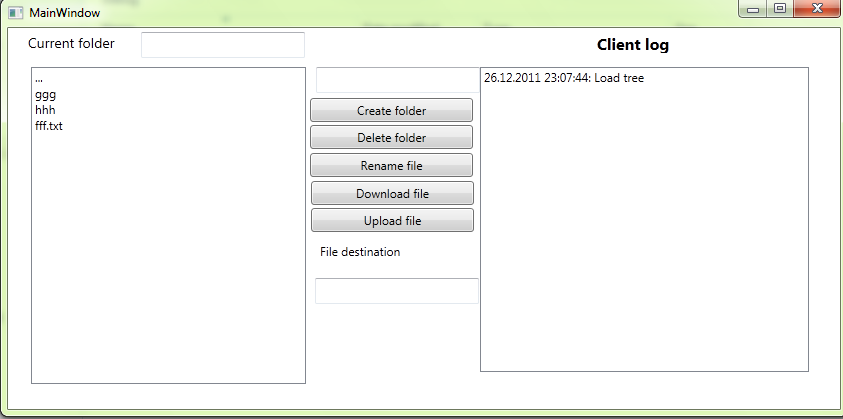


Рисунок 3.3.Вікно програми ClientGUI.

**Навігація на сервері**

В лівій половині вікна можна орієнтуватися на FTP-сервері. Файли пишуться з розширенням, а теки без. Щоб увійти у теку потрібно два рази натиснути лівою клавішею на імені теки. Щоб виділити теку або файл потрібно натиснути на імені теки або файлу один раз.

**Виконання операцій над теками**

Над теками можна виконувати дві операції – створення та видалення.

Для створення теки необхідно в верхньому полі вводу ввести адресу та ім’я нової теки та натиснути на кнопку Create folder.

Для видалення теки необхідно виділити теку та натиснути Delete folder.

**Виконання операцій над файлами**

Над теками можна виконувати чотири операції – видалення, перейменування, скачування та закачування.

Для видалення файлу необхідного виділити файл та натиснути Delete folder.

Для перейменування файлу необхідно виділити файл та ввести у верхньому полі вводу нове ім’я файлу з адресою, після цього натиснути Rename file.

Для скачування файлу з серверу потрібно виділити потрібний файл в лівій половині вікна, у верхньому полі вводу ввести адресу на комп’ютері та ім’я, під яким буде збережено файл. Після цього треба натиснути Download file.

Для закачування файлу на сервер потрібно в верхньому полі вводу ввести адресу та ім’я файлу на комп’ютері, в нижньому полі вводу ввести адресу та ім’я файлу, під яким файл буде збережено на сервері. Після цього натиснути Upload file.

**4. Перевірка програмного проекту та контрольні приклади**

Для перевірки програмного проекту було виконано декілька операцій. Результати перевірялись провідником Windows.

**Вигляд провіднику до виконання операцій.**

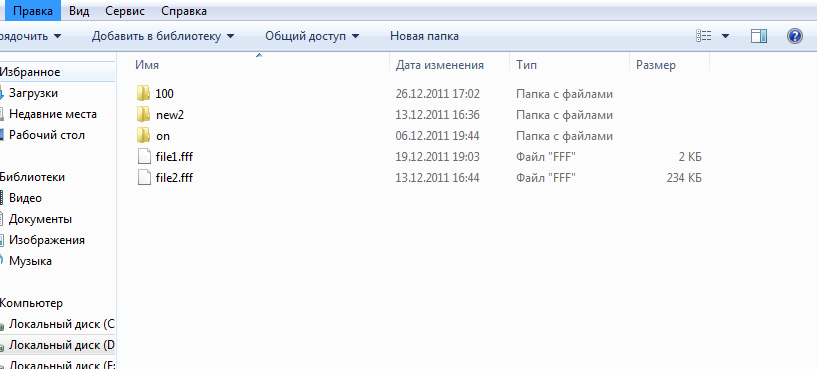


Рисунок 4.1.Вигляд провіднику до виконання операцій.

**Створення теки.**

Вигляд програми до створення теки та після

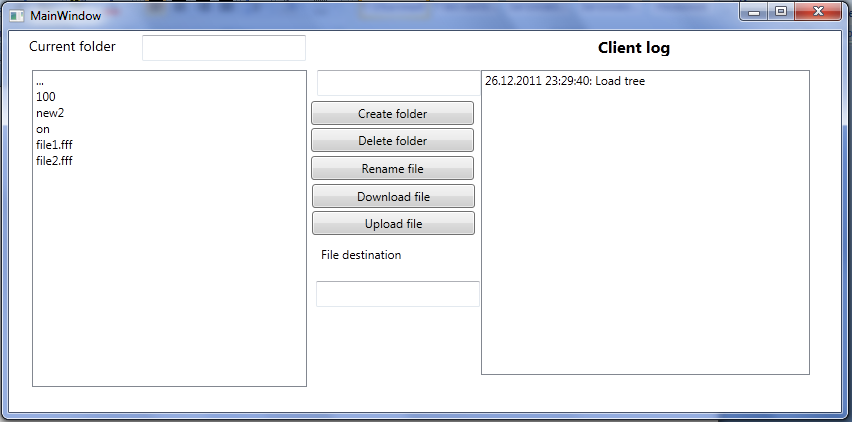


Рисунок 4.2.Вигляд програми ClientGUI до створення теки

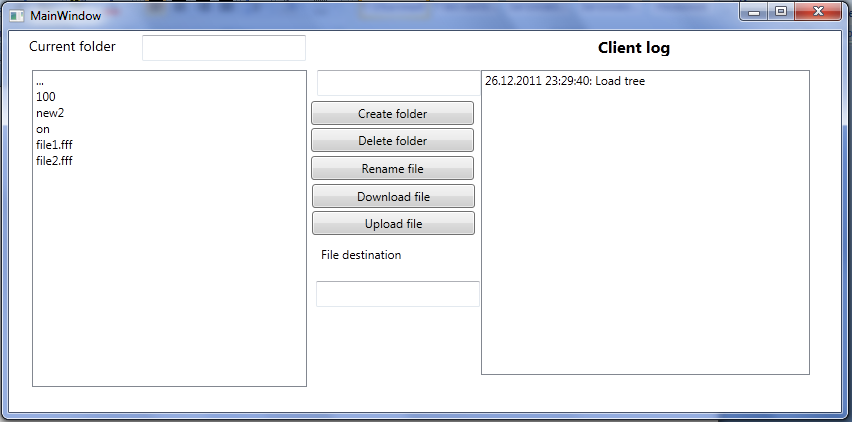


Рисунок 4.3.Вигляд програми після створення теки.

**Перейменування файлу та видалення теки**

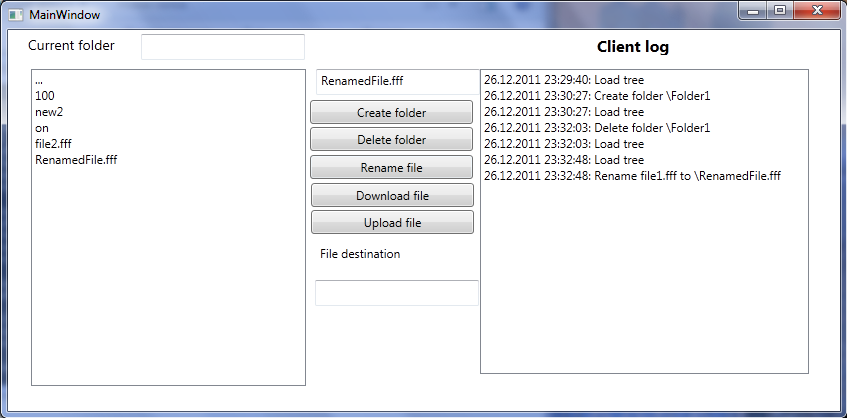


Рисунок 4.4.Вигляд програми після перейменування файлу та видалення теки.

**Видалення файлу**

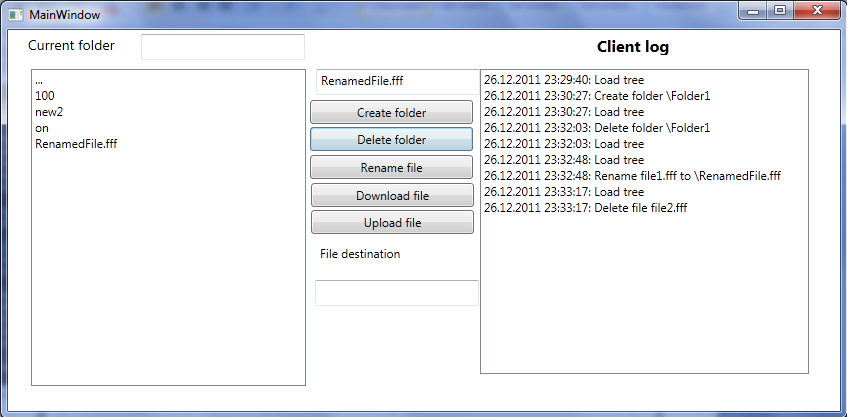


Рисунок 4.5. Вигляд програми після видалення файлу.

**Закачування файлу**

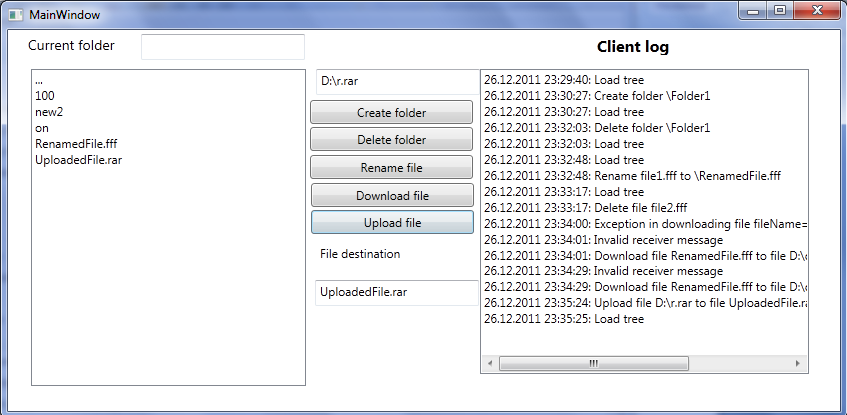


Рисунок 4.6.Вигляд програми після закачування файлу

**Вигляд провіднику після виконання цих операцій**

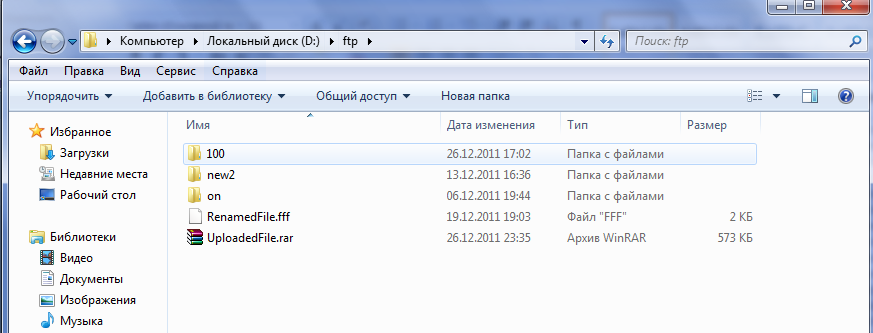


Рисунок 4.7.Вигляд провіднику після виконання операцій

**5. Висновки:**

Результатом виконаної роботи є програмний проект, що дозволяє працювати з FTP-сервером в активному режимі. Отримано дві програми – програма-сервер, яка приєднується до FTP-серверу для зменшення кількості клієнтів серверу та команда-клієнт, яка приєднується до програми-серверу для роботи з FTP-сервером. Програма-клієнт приєднується до програми-серверу за протоколом TCP, між ними відкриваються два сокети на різні порти, один для дерева, інший для команд. Програма-сервер підключається до FTP-серверу також за протоколом TCP та через два порти.

В процесі виконання роботи було отримано навички роботи з протоколом TCP, з виконанням операцій з FTP-сервером

**6. Список використаної літератури**

1. Герберт Шилдт C# 4.0: полное руководство = C# 4.0 The Complete Reference — М.: [«Вильямс»](http://ru.wikipedia.org/w/index.php?title=%D0%92%D0%B8%D0%BB%D1%8C%D1%8F%D0%BC%D1%81_(%D0%B8%D0%B7%D0%B4%D0%B0%D1%82%D0%B5%D0%BB%D1%8C%D1%81%D1%82%D0%B2%D0%BE)&action=edit&redlink=1), 2010. — С. 1056. — [ISBN 978-5-8459-1684-6](http://ru.wikipedia.org/wiki/%D0%A1%D0%BB%D1%83%D0%B6%D0%B5%D0%B1%D0%BD%D0%B0%D1%8F:BookSources/9785845916846);

2. Джон Скит C#: программирование для профессионалов, 2-е издание = C# in Depth, 2nd Edition — М.: [«Вильямс»](http://ru.wikipedia.org/w/index.php?title=%D0%92%D0%B8%D0%BB%D1%8C%D1%8F%D0%BC%D1%81_(%D0%B8%D0%B7%D0%B4%D0%B0%D1%82%D0%B5%D0%BB%D1%8C%D1%81%D1%82%D0%B2%D0%BE)&action=edit&redlink=1), 2011. — 544 с. — [ISBN 978-5-8459-1555-9](http://ru.wikipedia.org/wiki/%D0%A1%D0%BB%D1%83%D0%B6%D0%B5%D0%B1%D0%BD%D0%B0%D1%8F:BookSources/9785845915559);

3. http://msdn.microsoft.com/ru-ru/library/67ef8sbd.aspx

4. http://ru.wikipedia.org/wiki/FTP-%EA%EB%E8%E5%ED%F2

**Додаток**

**Лістинг програми (C#)**

**Лістинг проекту FTPProxyServer**

**Client.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Net;

using System.Net.Sockets;

using System.IO;

namespace FTPProxyServer

{

public class Client

{

public const String resultString = "Succesfull";//

public const String createFolderString = "CreateFolder";//delimiter folderName

public const String deleteFolderString = "DeleteFolder";//delimiter folderName

public const String deleteFileString = "DeleteFile";//delimiter fileName

public const String downloadFileString = "DownloadFile";//delimiter fileName

public const String uploadFileString = "UploadFile";//delimiter fileName

public const String renameFileString = "RenameFile";//delimiter oldName delimiter newName

public const String connectString = "Connect";//delimiter serverAddress delimiter login delimiter password

public const String delimiter = "@#@";

public const int buffLength = 1024;

public FTPOperations operations;

public TcpClient clientTcp;

public TcpClient clientTree;

public int tcpPort;

public int treePort;

public String clientIp;

public bool isConnected;

public TCPServerToFTP tcpServerToFTP;

public bool isConnectToFTP;

public Client(int port, int treePort1, String clientIP1, TCPServerToFTP tcpServerToFTP1)

{

if (clientIP1 == null)

{

throw new ConnectionException(ConnectionException.INVALID\_CLIENT\_ADDRESS\_OR\_PORT\_EXCEPTION);

}

tcpPort = port;

treePort = treePort1;

isConnected = false;

clientIp = clientIP1;

tcpServerToFTP = tcpServerToFTP1;

isConnectToFTP = false;

}

public void waitForConnection()

{

TcpListener listener = new TcpListener(new IPEndPoint(IPAddress.Any, tcpPort));

listener.Start();

clientTcp = listener.AcceptTcpClient();

isConnected = true;

receive();

}

public void callSendTree()

{

lock (tcpServerToFTP.clients)

{

for (int i = 0; i < tcpServerToFTP.clients.Count; i++)

{

System.Console.WriteLine("Start spoke to send tree="+i);

tcpServerToFTP.clients.ElementAt(i).sendTree();

}

}

}

public void sendResultOfOperation(String line)

{

try

{

System.Text.ASCIIEncoding encoding = new System.Text.ASCIIEncoding();

byte[] array = encoding.GetBytes(line);

clientTcp.GetStream().Write(array, 0, array.Length);

}

catch (Exception ex)

{

System.Console.WriteLine("Exception in sending result");

}

}

public void renameFile(String informationPart)

{

String oldName = informationPart.Substring(0, informationPart.IndexOf(delimiter));

informationPart = informationPart.Substring(informationPart.IndexOf(delimiter) + delimiter.Length, informationPart.Length - informationPart.IndexOf(delimiter) - delimiter.Length);

operations.renameFile(oldName, informationPart);

System.Console.WriteLine("Rename succesfull");

}

public void downloadingFile(String informationPart)

{

MemoryStream file = operations.downloadFile(informationPart);

StreamOperation.writeBytes(clientTcp.GetStream(), file);

file.Close();

System.Console.WriteLine("Download succesfull");

}

public void uploadingFile(String informationPart)

{

MemoryStream ms = new MemoryStream();

ms = StreamOperation.readBytes(clientTcp.GetStream());

operations.uploadFile(ms, informationPart);

ms.Close();

System.Console.WriteLine("Uploads succesfull");

}

public void connect(String informationPart)

{

// System.Console.WriteLine("Start connect client");

String ftpAddr = informationPart.Substring(0, informationPart.IndexOf(delimiter));

informationPart = informationPart.Substring(informationPart.IndexOf(delimiter) + delimiter.Length, informationPart.Length - informationPart.IndexOf(delimiter) - delimiter.Length);

String login = informationPart.Substring(0, informationPart.IndexOf(delimiter));

informationPart = informationPart.Substring(informationPart.IndexOf(delimiter) + delimiter.Length, informationPart.Length - informationPart.IndexOf(delimiter) - delimiter.Length);

String password = informationPart;

// System.Console.WriteLine("Start connect to ftp");

operations = new FTPOperations(ftpAddr, login, password);

// System.Console.WriteLine("End connect to ftp");

isConnectToFTP = true;

sendResultOfOperation(resultString);

// System.Console.WriteLine("Start send tree");

sendTree();

System.Console.WriteLine("Connection succesfull");

}

public void operate(String operatePart, String informationPart)

{

if (operatePart == null)

{

return;

}

if (!isConnectToFTP && operatePart.Equals(connectString))

{

connect(informationPart);

return;

}

try

{

if (operations == null)

{

sendResultOfOperation(FTPExceptions.invalid\_FTP\_connection\_message);

return;

}

switch (operatePart)

{

case createFolderString: { operations.createFolder(informationPart); System.Console.WriteLine("Create folder succesfull"); callSendTree(); break; };

case renameFileString: { renameFile(informationPart); System.Console.WriteLine("Rename file succesfull"); callSendTree(); break; };

case deleteFolderString: { operations.deleteFolder(informationPart); System.Console.WriteLine("Delete folder succesfull"); callSendTree(); break; };

case deleteFileString: { operations.deleteFile(informationPart); System.Console.WriteLine("Delete file succesfull"); callSendTree(); break; };

case downloadFileString: { downloadingFile(informationPart); break; };

case uploadFileString: { uploadingFile(informationPart); callSendTree(); break; };

}

sendResultOfOperation(resultString);

}

catch (FTPExceptions ex)

{

sendResultOfOperation(ex.getMessage());

}

catch (Exception e)

{

sendResultOfOperation(FTPExceptions.undefined\_exception);

}

}

public void receive()

{

while (true)

{

try

{

if (!clientTcp.Connected)

{

//continue;

System.Console.WriteLine("Break1");

break;

}

string str = StreamOperation.readString(clientTcp.GetStream());

if (str.Length == 0)

{

System.Console.WriteLine("Break2");

break;

}

System.Console.WriteLine("Receive string=" + str+" from client="+clientIp);

if (str.Equals(""))

{

continue;

}

if (str.Length == 0)

{

continue;

}

String operatePart = str.Substring(0, str.IndexOf(delimiter));

String informationPart = str.Substring(str.IndexOf(delimiter) + delimiter.Length, str.Length - str.IndexOf(delimiter) - delimiter.Length);

operate(operatePart, informationPart);

}

catch (Exception e)

{

sendResultOfOperation(ConnectionException.INVALID\_CLIENT\_MESSAGE\_MESSAGE);

}

}

}

public void sendTree()

{

//System.Console.WriteLine("Start method send tree");

if (clientTree == null)

{

// System.Console.WriteLine("Create tree connect on port="+treePort);

clientTree = new TcpClient();

clientTree.Connect(new IPEndPoint(IPAddress.Parse(clientIp), treePort));

//System.Console.WriteLine("End create connect");

}

MemoryStream ms = new MemoryStream();

//System.Console.WriteLine("Start build tree");

ms = operations.buildTree();

//System.Console.WriteLine("Have build tree");

StreamOperation.writeBytes(clientTree.GetStream(), ms);

System.Console.WriteLine("Have send tree to client" + clientIp + " on port=" + treePort);

}

}

}

**ConnectionException.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace FTPProxyServer

{

class ConnectionException:Exception

{

public int value;

public const int INVALID\_CLIENT\_MESSAGE\_EXCEPTION = 1;

public const int INVALID\_CLIENT\_ADDRESS\_OR\_PORT\_EXCEPTION = 2;

public const String INVALID\_CLIENT\_MESSAGE\_MESSAGE = "Invalid client message";

public const String INVALID\_CLIENT\_ADDREES\_OR\_PORT\_MESSAGE = "Invalid client ip address or port";

public const String undefined\_exception = "Undefined exceptin";

public ConnectionException(int value1)

{

value = value1;

}

public String getMessage()

{

switch (value)

{

case (INVALID\_CLIENT\_MESSAGE\_EXCEPTION): { return INVALID\_CLIENT\_MESSAGE\_MESSAGE; };

case (INVALID\_CLIENT\_ADDRESS\_OR\_PORT\_EXCEPTION): { return INVALID\_CLIENT\_ADDREES\_OR\_PORT\_MESSAGE; };

}

return undefined\_exception;

}

}

}

**FTPExceptions.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace FTPProxyServer

{

class FTPExceptions: Exception

{

public int value;

public const int INVALID\_FILE\_LIST\_EXCEPTION=1;

public const int INVALID\_BUILDING\_AND\_SERIALIZING\_TREE\_EXCEPTION = 2;

public const int INVALID\_DOWNLOADING\_FILE\_EXCEPTION = 3;

public const int INVALID\_UPLOADING\_FILE\_EXCEPTION = 4;

public const int INVALID\_REMOVING\_FILE\_EXCEPTION = 5;

public const int INVALID\_MOVING\_FILE\_EXCEPTION = 6;

public const int INVALID\_CREATING\_FOLER\_EXCEPTION = 7;

public const int INVALID\_REMOVING\_FOLER\_EXCEPTION = 8;

public const int INVALID\_RENAMING\_FILE\_EXCEPTION = 9;

public const int INVALID\_FTP\_CONNECTION\_EXCEPTION = 10;

public static String invalid\_file\_list\_message = "Invalid file list from ftp server";

public static String invalid\_building\_and\_serializing\_tree\_message = "Invalid building and serizalizing tree";

public static String invalid\_downloading\_file\_message = "Invalid downloading file";

public static String invalid\_uploading\_file\_message = "Invalid uploading file";

public static String invalid\_removing\_file\_message = "Invalid removing file";

public static String invalid\_moving\_file\_message = "Invalid moving file";

public static String invalid\_removing\_folder\_message = "Invalid removing folder";

public static String invalid\_creating\_folder\_message = "Invalid creating folder";

public static String invalid\_renaming\_file\_message = "Invalid renaming file";

public static String undefined\_exception = "Undefined exceptin";

public static String invalid\_FTP\_connection\_message = "Invalid connection to ftp server";

public FTPExceptions(int value1)

{

value = value1;

}

public String getMessage()

{

switch (value)

{

case INVALID\_FILE\_LIST\_EXCEPTION: { return invalid\_file\_list\_message; };

case INVALID\_BUILDING\_AND\_SERIALIZING\_TREE\_EXCEPTION: { return invalid\_building\_and\_serializing\_tree\_message; };

case INVALID\_DOWNLOADING\_FILE\_EXCEPTION: { return invalid\_downloading\_file\_message; };

case INVALID\_UPLOADING\_FILE\_EXCEPTION: { return invalid\_uploading\_file\_message; };

case INVALID\_MOVING\_FILE\_EXCEPTION: { return invalid\_moving\_file\_message; };

case INVALID\_REMOVING\_FILE\_EXCEPTION: { return invalid\_removing\_file\_message; };

case INVALID\_REMOVING\_FOLER\_EXCEPTION: { return invalid\_removing\_folder\_message; };

case INVALID\_CREATING\_FOLER\_EXCEPTION: { return invalid\_creating\_folder\_message; };

case INVALID\_RENAMING\_FILE\_EXCEPTION: { return invalid\_renaming\_file\_message; };

case INVALID\_FTP\_CONNECTION\_EXCEPTION: { return invalid\_FTP\_connection\_message; };

}

return undefined\_exception;

}

}

}

**FTPOperations.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Net;

using System.IO;

using System.Net.Sockets;

using System.Threading;

namespace FTPProxyServer

{

public class FTPOperations

{

private String endCMD = "\r\n";

private String userCMD = "USER ";

private String passCMD = "PASS ";

private String pasvCMD = "PASV ";

private String portCMD = "PORT ";

private String listCMD = "LIST ";

private String mkdCMD = "MKD ";

private String rmdCMD = "RMD ";

private String deleCMD = "DELE ";

private String rnfrCMD = "RNFR ";

private String rntoCMD = "RNTO ";

private String retrCMD = "RETR ";

private String storCMD = "STOR ";

private int commandPort=21;

private String ftpServerIP;

private String user;

private String password;

private TcpClient commandForFTP;

private TcpClient dataForFTP;

private int p = 1;

private int p1 = 1;

private TcpListener listener;

public FTPOperations(String ftpServerIP1, String user1, String password1)

{

ftpServerIP = ftpServerIP1;

user = user1;

password = password1;

connectToServer();

p1 = Program.portNumberForFTP;

Program.portNumberForFTP += 10;

}

public bool connectToServer()

{

Tuple<bool, String> error;

commandForFTP = new TcpClient();

commandForFTP.Connect(new IPEndPoint(IPAddress.Parse(ftpServerIP), commandPort));

String answ = StreamOperation.readString(commandForFTP.GetStream());

while (answ.IndexOf("http") < 0)

{

answ = StreamOperation.readString(commandForFTP.GetStream());

}

error = isError(answ);

if (error.Item1)

{

System.Console.WriteLine("Error " + error.Item2);

return false;

}

StreamOperation.writeString(commandForFTP.GetStream(), userCMD + user + endCMD);

answ = StreamOperation.readString(commandForFTP.GetStream());

error = isError(answ);

if (error.Item1)

{

return false;

}

StreamOperation.writeString(commandForFTP.GetStream(), passCMD + password + endCMD);

answ = StreamOperation.readString(commandForFTP.GetStream());

error = isError(answ);

if (error.Item1)

{

System.Console.WriteLine("Error " + error.Item2);

return false;

}

return true;

}

public void sendPasv()

{

StreamOperation.writeString(commandForFTP.GetStream(), portCMD + ftpServerIP+"."+p1.ToString()+"."+p.ToString() + endCMD);

String answ = StreamOperation.readString(commandForFTP.GetStream());

Tuple<bool,String> error = isError(answ);

if (error.Item1)

{

System.Console.WriteLine("Error " + error.Item2);

return;

}

}

public void connectToData()

{

if (listener != null)

{

listener.Stop();

dataForFTP.Close();

}

listener = new TcpListener(IPAddress.Any, 256\*p1+p);

p++;

//System.Console.WriteLine("Port for FTP=" + (256 \* p1 + p));

if (p > 99)

{

p = 0;

p1++;

}

listener.Start();

dataForFTP = listener.AcceptTcpClient();

}

public List<Tuple<String,bool>> getFileList(String folder)

{

sendPasv();

StreamOperation.writeString(commandForFTP.GetStream(), listCMD + folder + endCMD);

connectToData();

String filesLine= StreamOperation.readString(dataForFTP.GetStream());

String answ = StreamOperation.readString(commandForFTP.GetStream());

Tuple<bool,String> error = isError(answ);

if (error.Item1)

{

System.Console.WriteLine("Error " + error.Item2);

throw new FTPExceptions(FTPExceptions.INVALID\_FILE\_LIST\_EXCEPTION);

}

if (filesLine == null)

{

return null;

}

if (filesLine.Length==0)

{

return null;

}

List<Tuple<String,bool>> files = new List<Tuple<String,bool>>();

int endP;

Tuple<String, bool> ct;

while (true)

{

endP=filesLine.IndexOf(endCMD);

if(endP<=0)

{

break;

}

String fileName = filesLine.Substring(0, endP);

bool isDirectory=false;

if (fileName.ElementAt(0) == 'd')

{

isDirectory = true;

}

files.Add(new Tuple<String,bool>(fileName.Substring(fileName.LastIndexOf(' ')+1,fileName.Length-fileName.LastIndexOf(' ')-1),isDirectory));

filesLine = filesLine.Substring(endP+endCMD.Length, filesLine.Length - endP-endCMD.Length);

}

return files;

}

public bool createFolder(String folderName)

{

StreamOperation.writeString(commandForFTP.GetStream(), mkdCMD+folderName+ endCMD);

String answ = StreamOperation.readString(commandForFTP.GetStream());

Tuple<bool,String> error = isError(answ);

if (error.Item1)

{

System.Console.WriteLine("Error " + error.Item2);

return false;

}

return true;

}

public bool deleteFolder(String folderName)

{

StreamOperation.writeString(commandForFTP.GetStream(), rmdCMD + folderName + endCMD);

String answ = StreamOperation.readString(commandForFTP.GetStream());

Tuple<bool, String> error = isError(answ);

if (error.Item1)

{

System.Console.WriteLine("Error " + error.Item2);

return false;

}

return true;

}

public bool deleteFile(String fileName)

{

StreamOperation.writeString(commandForFTP.GetStream(), deleCMD + fileName + endCMD);

String answ = StreamOperation.readString(commandForFTP.GetStream());

Tuple<bool, String> error = isError(answ);

if (error.Item1)

{

System.Console.WriteLine("Error " + error.Item2);

return false;

}

return true;

}

public bool renameFile(String fileOldName,String fileNewName)

{

StreamOperation.writeString(commandForFTP.GetStream(), rnfrCMD + fileOldName + endCMD);

String answ = StreamOperation.readString(commandForFTP.GetStream());

Tuple<bool, String> error = isError(answ);

if (error.Item1)

{

System.Console.WriteLine("Error " + error.Item2);

return false;

}

StreamOperation.writeString(commandForFTP.GetStream(), rntoCMD + fileNewName + endCMD);

answ = StreamOperation.readString(commandForFTP.GetStream());

error = isError(answ);

if (error.Item1)

{

System.Console.WriteLine("Error " + error.Item2);

return false;

}

return true;

}

public Tuple<bool, String> isError(String answer)

{

try{

int number = int.Parse(answer.Substring(0, 1));

if (number!= 4 || number!=5)

{

return new Tuple<bool, string>(false, null);

}

else

{

return new Tuple<bool, string>(true, answer.Substring(3, answer.Length - 3));

}

}catch(Exception e){

return new Tuple<bool, string>(true, "unidentifier error");

}

}

public MemoryStream downloadFile(String fileName)

{

sendPasv();

StreamOperation.writeString(commandForFTP.GetStream(), retrCMD + fileName + endCMD);

connectToData();

String answ = StreamOperation.readString(commandForFTP.GetStream());

Tuple<bool, String> error = isError(answ);

if (error.Item1)

{

System.Console.WriteLine("Error " + error.Item2);

return null;

}

MemoryStream ms=StreamOperation.readBytes(dataForFTP.GetStream());

return ms;

}

public bool uploadFile(MemoryStream ms, String fileName)

{

sendPasv();

StreamOperation.writeString(commandForFTP.GetStream(), storCMD + fileName + endCMD);

connectToData();

StreamOperation.writeBytes(dataForFTP.GetStream(), ms);

String answ = StreamOperation.readString(commandForFTP.GetStream());

Tuple<bool, String> error = isError(answ);

if (error.Item1)

{

System.Console.WriteLine("Error " + error.Item2);

return false;

}

return true;

}

public Node buildNodeFromFolder(String folderName)

{

if (folderName == null)

{

throw new ArgumentNullException();

}

Node node = new Node();

node.name = folderName;

List<Tuple<String,bool>> currentFolder = getFileList(folderName);

if (currentFolder == null)

{

return node;

}

Tuple<String, bool> currTup;

for (int i = 0; i < currentFolder.Count; i++)

{

currTup = currentFolder.ElementAt(i);

if (currTup.Item2)

{

node.addFolder(buildNodeFromFolder(folderName+"\\"+currTup.Item1));

}

else

{

node.addFile(currTup.Item1);

}

}

return node;

}

public MemoryStream buildTree()

{

MemoryStream ms=null;

try

{

Node head = buildNodeFromFolder("");

List<Node> nodes = new List<Node>();

nodes = head.getList();

nodes.Add(head);

System.Console.WriteLine("Have build, start serialize");

System.Xml.Serialization.XmlSerializer serializer = new System.Xml.Serialization.XmlSerializer(nodes.GetType());

ms = new MemoryStream();

serializer.Serialize(ms, nodes);

System.Console.WriteLine("Have serizalize");

}

catch (Exception ex)

{

System.Console.WriteLine("Exception e"+ex.ToString());

throw new FTPExceptions(FTPExceptions.INVALID\_BUILDING\_AND\_SERIALIZING\_TREE\_EXCEPTION);

}

return ms;

}

}

}

**Node.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Xml.Serialization;

using System.Runtime.Serialization;

namespace FTPProxyServer

{

[Serializable]

public class Node

{

[XmlAttribute("File")]

public List<String> files

{ get; set; }

[XmlAttribute("Folders")]

public List<String> foldersNames

{ get; set; }

[XmlIgnore]

public List<Node> folders

{ get; set; }

[XmlAttribute("FolderName")]

public String name

{ get; set; }

public Node()

{

files = new List<String>();

folders = new List<Node>();

foldersNames = new List<String>();

}

public void addFile(String fileName)

{

files.Add(fileName);

}

public void addFolder(Node folderName)

{

folders.Add(folderName);

}

public List<Node> getList()

{

List<Node> res = new List<Node>();

for (int i = 0; i < folders.Count; i++)

{

foldersNames.Add(folders.ElementAt(i).name);

List<Node> nodes = folders.ElementAt(i).getList();

res.AddRange(nodes);

res.Add(folders.ElementAt(i));

}

return res;

}

}

}

**Program.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.IO;

namespace FTPProxyServer

{

class Program

{

public static int portNumberForFTP=1;

static void Main(string[] args)

{

//FTPOperations op = new FTPOperations("127.0.0.1", "root", "1993");

/\*for (int i = 0; i < 11; i++)

{

op.getFileList("\\100");

}\*/

//op.buildTree();

TCPServerToFTP tcpServer = new TCPServerToFTP(10);

tcpServer.takeConnection();

String a;

a = System.Console.ReadLine();

}

}

}

**StreamOperation.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Net.Sockets;

using System.IO;

namespace FTPProxyServer

{

class StreamOperation

{

public static MemoryStream readBytes(NetworkStream stream)

{

if (stream == null)

{

throw new ConnectionException(ConnectionException.INVALID\_CLIENT\_ADDRESS\_OR\_PORT\_EXCEPTION);

}

System.Text.ASCIIEncoding encoding = new System.Text.ASCIIEncoding();

int buffLength = 1024;

byte[] array = new byte[buffLength];

int length;

MemoryStream ms = new MemoryStream();

while (true)

{

length = stream.Read(array, 0, buffLength);

ms.Write(array, 0, length);

if (length < buffLength)

{

break;

}

}

return ms;

}

public static void writeBytes(NetworkStream stream, MemoryStream ms)

{

if (stream == null || ms == null)

{

throw new ConnectionException(ConnectionException.INVALID\_CLIENT\_ADDRESS\_OR\_PORT\_EXCEPTION);

}

int buffLength=1024;

byte[] array=new byte[buffLength];

int length;

ms.Seek(0, SeekOrigin.Begin);

while (true)

{

length=ms.Read(array, 0, buffLength);

stream.Write(array, 0, length);

if (length < buffLength)

{

break;

}

}

}

public static void writeString(NetworkStream stream,String line)

{

if (stream == null || line==null)

{

throw new ConnectionException(ConnectionException.INVALID\_CLIENT\_ADDRESS\_OR\_PORT\_EXCEPTION);

}

System.Text.ASCIIEncoding encoding = new System.Text.ASCIIEncoding();

byte[] array = encoding.GetBytes(line);

stream.Write(array, 0, array.Length);

}

public static String readString(NetworkStream stream){

if (stream == null)

{

throw new ConnectionException(ConnectionException.INVALID\_CLIENT\_ADDRESS\_OR\_PORT\_EXCEPTION);

}

int buffLength = 1024;

byte[] buff=new byte[buffLength];

int length=-1;

int sumLength=0;

List<byte[]> list = new List<byte[]>();

List<int> sizes = new List<int>();

while (true)

{

length = stream.Read(buff, 0, buffLength);

list.Add(buff);

sumLength += length;

sizes.Add(length);

if (length < buffLength)

{

break;

}

}

byte[] resArr = new byte[sumLength];

int n = 0;

for (int i = 0; i < list.Count; i++)

{

for (int j = 0; j < sizes.ElementAt(i); j++)

{

resArr[n] = list.ElementAt(i)[j];

n++;

}

}

string str = System.Text.Encoding.ASCII.GetString(resArr, 0, resArr.Length);

return str;

}

}

}

**TCPServerToFTP.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Net;

using System.Net.Sockets;

using System.Collections;

using System.Threading;

using System.IO;

namespace FTPProxyServer

{

public class TCPServerToFTP

{

public int portTcp;

public List<Client> clients;

public TcpListener tcpListener;

public List<int> busyPorts;

public TCPServerToFTP(int portTcp1)

{

if (portTcp1 < 0)

{

throw new ConnectionException(ConnectionException.INVALID\_CLIENT\_ADDRESS\_OR\_PORT\_EXCEPTION);

}

portTcp = portTcp1;

clients = new List<Client>();

busyPorts = new List<int>();

busyPorts.Add(portTcp);

}

public int generateFreePort()

{

if (busyPorts.Count > 65535)

{

return -1;

}

while (true)

{

Random r = new Random();

int p = r.Next(65534);

if (p % 2 == 1)

{

continue;

}

for (int i = 0; i < busyPorts.Count; i++)

{

if (p == busyPorts.ElementAt(i))

{

continue;

}

}

if (p < 500)

{

continue;

}

return p;

}

}

public void takeConnection()

{

System.Console.WriteLine("Start server on port=" + portTcp);

tcpListener = new TcpListener(IPAddress.Any, portTcp);

tcpListener.Start();

while (true)

{

try

{

System.Console.WriteLine("Wait for new client");

TcpClient clientTcp = tcpListener.AcceptTcpClient();

// System.Console.WriteLine("Accept new client");

int newPort = generateFreePort();

if (newPort == -1)

{

clientTcp.Close();

continue;

}

StreamOperation.writeString(clientTcp.GetStream(), newPort.ToString());

// System.Console.WriteLine("Send string to client");

String val = clientTcp.Client.RemoteEndPoint.ToString();

String ip = val.Substring(0, val.IndexOf(':'));

String port = val.Substring(val.IndexOf(':') + 1, val.Length - val.IndexOf(':') - 1);

System.Console.WriteLine("Connect new client ip=" + ip + " port=" + newPort);

Client client = new Client(newPort, newPort + 1, ip, this);

clients.Add(client);

Thread threadForClient = new Thread(new ThreadStart(client.waitForConnection));

threadForClient.Start();

}

catch (Exception ex)

{

System.Console.WriteLine("Exception " + ex.ToString());

}

}

}

}

}

**Лістинг ClientGUI**

**App.xaml.cs**

using System;

using System.Collections.Generic;

using System.Configuration;

using System.Data;

using System.Linq;

using System.Windows;

namespace ClientGUI

{

/// <summary>

/// Interaction logic for App.xaml

/// </summary>

public partial class App : Application

{

}

}

**App.xaml**

<Application x:Class="ClientGUI.App"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

StartupUri="AuthorizationWnd.xaml">

<Application.Resources>

</Application.Resources>

</Application>

**AuthorizationWnd.xaml.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

using System.Threading;

namespace ClientGUI

{

/// <summary>

/// Interaction logic for Window1.xaml

/// </summary>

public partial class AuthorizationWnd : Window

{

private int singleton = 0;

public MainWindow mainWindow;

public AuthorizationWnd()

{

InitializeComponent();

}

private void authorizationBtn\_Click(object sender, RoutedEventArgs e)

{

if (singleton > 0)

{

return;

}

singleton++;

if (loginTxt.Text.Length == 0 || passwordTxt.Text.Length == 0 || ftpServerTxt.Text.Length == 0)

{

string messageBoxText = "You must write all parametrs of connection";

string caption = "Connection";

MessageBoxButton button = MessageBoxButton.OK;

MessageBoxImage icon = MessageBoxImage.Error;

MessageBox.Show(messageBoxText, caption, button, icon);

return;

}

Logger logger = new Logger();

mainWindow = new MainWindow(logger);

logger.window = mainWindow;

TCPClientToFTP client = new TCPClientToFTP(ftpServerTxt.Text, 10, ftpAddressTxt.Text,loginTxt.Text,passwordTxt.Text,logger);

client.connect();

{

mainWindow.setClient(client);

//mainWindow.setLogger(logger);

client.mainWindow = mainWindow;

//Thread receiveThread = new Thread(new ThreadStart(client.receiveTree));

//receiveThread.Start();

mainWindow.Show();

this.Close();

}

}

}

}

**Authorization.xaml**

<Window x:Class="ClientGUI.AuthorizationWnd"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

Title="Authorization window" Height="300" Width="300">

<Grid>

<Button Content="Authorization" Height="33" HorizontalAlignment="Left" Margin="63,175,0,0" Name="authorizationBtn" VerticalAlignment="Top" Width="157" Click="authorizationBtn\_Click" />

<Label Content="Server address" Height="25" HorizontalAlignment="Left" Margin="12,25,0,0" Name="label1" VerticalAlignment="Top" Width="96" />

<Label Content="Login" Height="26" HorizontalAlignment="Left" Margin="16,58,0,0" Name="label2" VerticalAlignment="Top" Width="78" />

<Label Content="Password" Height="27" HorizontalAlignment="Left" Margin="12,94,0,0" Name="label3" VerticalAlignment="Top" Width="78" />

<TextBox Height="20" HorizontalAlignment="Left" Margin="123,30,0,0" Name="ftpServerTxt" VerticalAlignment="Top" Width="143" Text="127.0.0.1" />

<TextBox Height="20" HorizontalAlignment="Left" Margin="123,64,0,0" Name="loginTxt" VerticalAlignment="Top" Width="143" Text="root" />

<TextBox Height="20" HorizontalAlignment="Left" Margin="123,101,0,0" Name="passwordTxt" VerticalAlignment="Top" Width="143" Text="1993" />

<Label Content="FTP address" Height="27" HorizontalAlignment="Left" Margin="12,127,0,0" Name="label4" VerticalAlignment="Top" Width="78" />

<TextBox Height="20" HorizontalAlignment="Left" Margin="123,130,0,0" Name="ftpAddressTxt" VerticalAlignment="Top" Width="143" Text="127.0.0.1" />

</Grid>

</Window>

**ClientException.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace ClientGUI

{

class ClientException:Exception

{

public int value;

public const int INVALID\_SERVER\_ADDRESS\_OR\_PORT\_EXCEPTION=1;

public const String invalid\_server\_address\_or\_port\_message = "Invalid server address or port";

public const String invalid\_received\_message = "Invalid receiver message";

public const String undefined\_exception = "Undefined exception";

public ClientException(int value1)

{

value = value1;

}

public String getMessage()

{

switch (value)

{

case (INVALID\_SERVER\_ADDRESS\_OR\_PORT\_EXCEPTION): { return invalid\_server\_address\_or\_port\_message; };

}

return undefined\_exception;

}

}

}

**ConnectionException.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace ClientGUI

{

class ConnectionException:Exception

{

public int value;

public const int INVALID\_CLIENT\_MESSAGE\_EXCEPTION = 1;

public const int INVALID\_CLIENT\_ADDRESS\_OR\_PORT\_EXCEPTION = 2;

public const String INVALID\_CLIENT\_MESSAGE\_MESSAGE = "Invalid client message";

public const String INVALID\_CLIENT\_ADDREES\_OR\_PORT\_MESSAGE = "Invalid client ip address or port";

public const String undefined\_exception = "Undefined exceptin";

public ConnectionException(int value1)

{

value = value1;

}

public String getMessage()

{

switch (value)

{

case (INVALID\_CLIENT\_MESSAGE\_EXCEPTION): { return INVALID\_CLIENT\_MESSAGE\_MESSAGE; };

case (INVALID\_CLIENT\_ADDRESS\_OR\_PORT\_EXCEPTION): { return INVALID\_CLIENT\_ADDREES\_OR\_PORT\_MESSAGE; };

}

return undefined\_exception;

}

}

}

**Logger.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace ClientGUI

{

public class Logger

{

public bool isUpdate = true;

public MainWindow window;

public List<String> messages;

public Logger()

{

messages = new List<String>();

}

public void addMessage(String message)

{

DateTime dt = DateTime.Now;

String messageTime = dt.ToString();

String fullMes = messageTime + ": " + message;

messages.Add(fullMes);

isUpdate = true;

}

}

}

**MainWindow.xaml.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using System.Threading;

using System.IO;

using System.Windows.Threading;

namespace ClientGUI

{

/// <summary>

/// Interaction logic for MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

private bool isSetNewTree;

private List<Node> previousFolder;

private List<String> folders;

public Logger logger;

public Node currentNodeFolder;

public Node treeHead;

public TCPClientToFTP client;

public Timer timer;

static private readonly object theLock = new Object();

public MainWindow(Logger logger1)

{

InitializeComponent();

previousFolder = new List<Node>();

isSetNewTree = false;

logger = logger1;

DispatcherTimer dispatcherTimer = new DispatcherTimer();

dispatcherTimer.Tick += new EventHandler(loadTree);

dispatcherTimer.Interval = new TimeSpan(10000000);

dispatcherTimer.Start();

}

private String takeName(String name)

{

int l=name.LastIndexOf('\\');

if(l<0){

l=0;

}else{

l++;

}

String res=name.Substring(l,name.Length-l);

return res;

}

public void setClient(TCPClientToFTP client1)

{

client = client1;

}

public void setLogger(Logger logger1)

{

logger = logger1;

}

public void setTree(Node treeHead1)

{

lock (this)

{

logger.addMessage("Load tree");

treeHead = treeHead1;

isSetNewTree = true;

}

}

private void loadTree(object sender, EventArgs e)

{

lock (logger)

{

if (logger.isUpdate)

{

logList.Items.Clear();

for (int i = 0; i < logger.messages.Count; i++)

{

logList.Items.Add(logger.messages.ElementAt(i));

}

logger.isUpdate = false;

}

}

if (!isSetNewTree)

{

return;

}

lock (this)

{

if (treeHead != null)

{

currentNodeFolder = treeHead;

drawCurrentDirectory();

}

else

{

Console.WriteLine("TreeHad=null");

}

}

}

private void drawCurrentDirectory()

{

folders = new List<String>();

folders.Add("...");

for (int i = 0; i < currentNodeFolder.folders.Count; i++)

{

folders.Add(takeName(currentNodeFolder.folders.ElementAt(i).name));

}

for (int i = 0; i < currentNodeFolder.files.Count; i++)

{

folders.Add(takeName(currentNodeFolder.files.ElementAt(i)));

}

this.foldersList.Items.Clear();

for (int i = 0; i < folders.Count; i++)

{

this.foldersList.Items.Add(folders.ElementAt(i));

}

isSetNewTree = false;

}

private void Window\_Loaded(object sender, RoutedEventArgs e)

{

}

private void foldersList\_SelectionChanged(object sender, MouseButtonEventArgs e)

{

if (treeHead == null || currentNodeFolder==null)

{

return;

}

int si = foldersList.SelectedIndex;

String sname ="\\"+ (String)foldersList.SelectedItem;

System.Console.WriteLine("-----------Selected item=" + si + " sname=" + sname);

Node prevF = currentNodeFolder;

if (si == 0)

{

if (previousFolder.Count == 0)

{

return;

}

currentNodeFolder = previousFolder.ElementAt(previousFolder.Count-1);

previousFolder.Remove(currentNodeFolder);

currentFolder.Text = currentNodeFolder.name;

drawCurrentDirectory();

return;

}

for (int i = 0; i < currentNodeFolder.folders.Count; i++)

{

if (currentNodeFolder.folders.ElementAt(i).name.Equals(currentFolder.Text+sname))

{

currentNodeFolder = currentNodeFolder.folders.ElementAt(i);

currentFolder.Text = currentNodeFolder.name;

previousFolder.Add(prevF);

drawCurrentDirectory();

}

}

}

private void deleteBtn\_Click(object sender, RoutedEventArgs e)

{

int si = foldersList.SelectedIndex;

if (si <= 0)

{

showMessageDialog("Select file", "Deleting file");

return;

}

String fname = currentFolder.Text + "\\" + (String)foldersList.SelectedItem;

int n = currentNodeFolder.isFolder(fname);

if (n >= 0)

{

client.deleteFolder(fname);

logger.addMessage("Delete folder " + fname);

return;

}

String sname=null;

if (currentFolder.Text.Length != 0)

{

sname = currentFolder.Text + "\\" + (String)foldersList.SelectedItem;

}

else

{

sname = (String)foldersList.SelectedItem;

}

n = currentNodeFolder.isFile(sname);

if (n >= 0)

{

client.deleteFile(sname);

logger.addMessage("Delete file " + sname);

return;

}

logger.addMessage("Cann`t find entity "+(String)foldersList.SelectedItem);

}

private void showMessageDialog(String message,String caption)

{

if (message == null || caption == null)

{

return;

}

MessageBoxButton button = MessageBoxButton.OK;

MessageBoxImage icon = MessageBoxImage.Error;

MessageBox.Show(message, caption, button, icon);

}

private void createFolderBtn\_Click(object sender, RoutedEventArgs e)

{

if (currentFolder==null && additionFile.Text.Length==0)

{

showMessageDialog("Write folder name", "Creating folder");

return;

}

client.createFolder(currentFolder.Text + "\\" + additionFile.Text);

logger.addMessage("Create folder " + currentFolder.Text + "\\" + additionFile.Text);

}

private void RenameBtn\_Click(object sender, RoutedEventArgs e)

{

int si = foldersList.SelectedIndex;

if (si <= 0 || additionFile.Text.Length==0)

{

showMessageDialog("Write new name of file", "Renaming file");

return;

}

client.renameFile((String)foldersList.SelectedItem, currentFolder.Text + "\\" + additionFile.Text);

logger.addMessage("Rename " + (String)foldersList.SelectedItem + " to " + currentFolder.Text + "\\" + additionFile.Text);

}

private void DownloadBtn\_Click(object sender, RoutedEventArgs e)

{

int si = foldersList.SelectedIndex;

if (si <= 0 || additionFile.Text.Length == 0)

{

showMessageDialog("Write destination file", "Downloading file");

return;

}

client.downloadFile((String)foldersList.SelectedItem, additionFile.Text);

logger.addMessage("Download file " + (String)foldersList.SelectedItem + " to file " + additionFile.Text);

}

private void UploadBtn\_Click(object sender, RoutedEventArgs e)

{

int si = foldersList.SelectedIndex;

if (destinationFile.Text.Length==0 || additionFile.Text.Length == 0)

{

showMessageDialog("Write destination file", "Uploading file");

return;

}

client.uploadFile((String)additionFile.Text, destinationFile.Text);

logger.addMessage("Upload file " + (String)additionFile.Text + " to file " + destinationFile.Text);

}

}

}

**MainWindow.xaml**

<Window x:Class="ClientGUI.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

Title="MainWindow" Height="419" Width="848" Loaded="Window\_Loaded">

<Grid Height="398">

<Grid.RowDefinitions>

<RowDefinition Height="312\*" />

<RowDefinition Height="86\*" />

</Grid.RowDefinitions>

<ListBox Height="317" HorizontalAlignment="Left" Margin="23,39,0,0" Name="foldersList" VerticalAlignment="Top" Width="275" ItemsSource="{Binding}" MouseDoubleClick="foldersList\_SelectionChanged" Grid.RowSpan="2" />

<Button Content="Delete folder" Height="24" HorizontalAlignment="Left" Margin="302,97,0,0" Name="deleteBtn" VerticalAlignment="Top" Width="163" Click="deleteBtn\_Click" />

<ListBox Height="305" HorizontalAlignment="Left" Margin="472,39,0,0" Name="logList" VerticalAlignment="Top" Width="329" Grid.RowSpan="2" IsEnabled="True" />

<Label Content="Client log" Height="30" HorizontalAlignment="Left" Margin="584,0,0,0" Name="label1" VerticalAlignment="Top" Width="107" FontWeight="Bold" FontStretch="SemiCondensed" FontSize="16" />

<Label Content="Current folder" Height="31" HorizontalAlignment="Left" Margin="15,0,0,0" Name="label2" VerticalAlignment="Top" Width="102" FontSize="14" />

<TextBox Height="26" HorizontalAlignment="Right" Margin="0,4,535,0" Name="currentFolder" VerticalAlignment="Top" Width="164" IsReadOnly="True" />

<Button Content="Create folder" Height="24" HorizontalAlignment="Left" Margin="302,70,0,0" Name="createFolderBtn" VerticalAlignment="Top" Width="163" Click="createFolderBtn\_Click" />

<TextBox Height="26" HorizontalAlignment="Right" Margin="0,39,360,0" Name="additionFile" VerticalAlignment="Top" Width="164" Text="" />

<Button Content="Rename file" Height="24" HorizontalAlignment="Left" Margin="302,125,0,0" Name="RenameBtn" VerticalAlignment="Top" Width="163" Click="RenameBtn\_Click" />

<Button Content="Download file" Height="24" HorizontalAlignment="Left" Margin="303,153,0,0" Name="DownloadBtn" VerticalAlignment="Top" Width="163" Click="DownloadBtn\_Click" />

<Button Content="Upload file" Height="24" HorizontalAlignment="Left" Margin="303,180,0,0" Name="UploadBtn" VerticalAlignment="Top" Width="163" Click="UploadBtn\_Click" />

<TextBox Height="26" HorizontalAlignment="Right" Margin="0,250,361,0" Name="destinationFile" Text="" VerticalAlignment="Top" Width="164" />

<Label Content="File destination" Height="34" HorizontalAlignment="Left" Margin="307,210,0,0" Name="label3" VerticalAlignment="Top" Width="158" />

</Grid>

</Window>

**Node.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Xml.Serialization;

namespace ClientGUI

{

[Serializable]

public class Node

{

[XmlAttribute("File")]

public List<String> files

{ get; set; }

[XmlAttribute("Folders")]

public List<String> foldersNames

{ get; set; }

[XmlIgnore]

public List<Node> folders

{ get; set; }

[XmlAttribute("FolderName")]

public String name

{ get; set; }

public Node()

{

files = new List<String>();

folders = new List<Node>();

foldersNames = new List<String>();

}

public void addFile(String fileName)

{

files.Add(fileName);

}

public void addFolder(Node folderName)

{

folders.Add(folderName);

}

public List<Node> getList()

{

List<Node> res = new List<Node>();

for (int i = 0; i < folders.Count; i++)

{

List<Node> nodes = folders.ElementAt(i).getList();

for (int j = 0; j < nodes.Count; j++)

{

foldersNames.Add(nodes.ElementAt(j).name);

res.Add(nodes.ElementAt(j));

}

//System.Console.WriteLine("have add node=" + folders.ElementAt(i).name);

foldersNames.Add(folders.ElementAt(i).name);

res.Add(folders.ElementAt(i));

}

return res;

}

public void buildChild(List<Node> list)

{

this.folders = new List<Node>();

//System.Console.WriteLine("folder for=" + name);

for (int i = 0; i < this.foldersNames.Count; i++)

{

//System.Console.WriteLine("folder parent=" + this.name + " child=" + foldersNames.ElementAt(i));

Node n = searchNodeByName(foldersNames.ElementAt(i), list);

n.buildChild(list);

folders.Add(n);

}

}

public Node searchNodeByName(String name, List<Node> list)

{

for (int i = 0; i < list.Count; i++)

{

if (list.ElementAt(i).name.Equals(name))

{

return list.ElementAt(i);

}

}

return null;

}

public void writeTree()

{

System.Console.WriteLine("Current folder=" + name);

for (int i = 0; i < files.Count; i++)

{

System.Console.WriteLine("file=" + files.ElementAt(i));

}

for (int i = 0; i < folders.Count; i++)

{

System.Console.WriteLine("folder=" + folders.ElementAt(i).name);

folders.ElementAt(i).writeTree();

}

}

public int isFolder(String folderName)

{

for (int i = 0; i < folders.Count; i++)

{

if (folders.ElementAt(i).name.Equals(folderName))

{

return i;

}

}

return -1;

}

public int isFile(String fileName)

{

for (int i = 0; i < files.Count; i++)

{

if (files.ElementAt(i).Equals(fileName))

{

return i;

}

}

return -1;

}

}

}

**StreamOperations.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.IO;

using System.Net.Sockets;

namespace ClientGUI

{

class StreamOperation

{

public static MemoryStream readBytes(NetworkStream stream)

{

System.Console.WriteLine("Start read");

if (stream == null)

{

throw new ConnectionException(ConnectionException.INVALID\_CLIENT\_ADDRESS\_OR\_PORT\_EXCEPTION);

}

System.Console.WriteLine("Continue read");

int buffLength = 1024;

byte[] array = new byte[buffLength];

int length;

MemoryStream ms = new MemoryStream();

while (true)

{

length = stream.Read(array, 0, buffLength);

System.Console.WriteLine("Read=" + length);

ms.Write(array, 0, length);

if (length < buffLength)

{

break;

}

}

return ms;

}

public static void writeBytes(NetworkStream stream, MemoryStream ms)

{

if (stream == null || ms == null)

{

throw new ConnectionException(ConnectionException.INVALID\_CLIENT\_ADDRESS\_OR\_PORT\_EXCEPTION);

}

System.Text.ASCIIEncoding encoding = new System.Text.ASCIIEncoding();

int buffLength = 1024;

byte[] array = new byte[buffLength];

int length;

ms.Seek(0, SeekOrigin.Begin);

while (true)

{

length = ms.Read(array, 0, buffLength);

stream.Write(array, 0, length);

if (length < buffLength)

{

break;

}

}

}

public static void writeString(NetworkStream stream, String line)

{

if (stream == null || line == null)

{

throw new ConnectionException(ConnectionException.INVALID\_CLIENT\_ADDRESS\_OR\_PORT\_EXCEPTION);

}

System.Text.ASCIIEncoding encoding = new System.Text.ASCIIEncoding();

byte[] array = encoding.GetBytes(line);

stream.Write(array, 0, array.Length);

}

public static String readString(NetworkStream stream)

{

if (stream == null)

{

throw new ConnectionException(ConnectionException.INVALID\_CLIENT\_ADDRESS\_OR\_PORT\_EXCEPTION);

}

int buffLength = 1024;

byte[] buff = new byte[buffLength];

int length = -1;

int sumLength = 0;

List<byte[]> list = new List<byte[]>();

List<int> sizes = new List<int>();

while (true)

{

length = stream.Read(buff, 0, buffLength);

list.Add(buff);

sumLength += length;

sizes.Add(length);

if (length < buffLength)

{

break;

}

}

byte[] resArr = new byte[sumLength];

int n = 0;

for (int i = 0; i < list.Count; i++)

{

for (int j = 0; j < sizes.ElementAt(i); j++)

{

resArr[n] = list.ElementAt(i)[j];

n++;

}

}

string str = System.Text.Encoding.ASCII.GetString(resArr, 0, resArr.Length);

return str;

}

}

}

**TCPClientToFTP.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Net.Sockets;

using System.Net;

using System.IO;

using System.Xml.Serialization;

using System.Xml;

using System.Threading;

namespace ClientGUI

{

public class TCPClientToFTP

{

public const String resultString = "Succesfull";//

public static String registerString = "RegisterMe";//

public static String createFolderString = "CreateFolder";//delimiter folderName

public static String deleteFolderString = "DeleteFolder";//delimiter folderName

public static String deleteFileString = "DeleteFile";//delimiter fileName

public static String renameFileString = "RenameFile";//delimiter oldName delimiter newName

public static String downloadFileString = "DownloadFile";//delimiter fileName

public static String uploadFileString = "UploadFile";//delimiter fileName

public static String connectString = "Connect";//delimiter serverAddress delimiter login delimiter password

public static String delimiter = "@#@";

public static String pingQuery\_str = "Ping query";

public static String getTree\_str = "Get tree";

public static String pingAnswer\_str = "Ping answer";

public String ftpAddress;

public String login;

public String password;

public const int bufferLength = 1024;

public String ipAddress;

public int portTcp;

public int portTree;

public TcpClient tcpClient;

public TcpClient tcpClientTree;

public Logger logger;

public MainWindow mainWindow;

public TCPClientToFTP(String ipAddress1, int portTcp1,String ftpAddress1,String login1,String password1,Logger logger1)

{

if (ftpAddress1==null|| ipAddress1 == null || portTcp1 < 0 || login1==null || password1==null)

{

throw new ClientException(ClientException.INVALID\_SERVER\_ADDRESS\_OR\_PORT\_EXCEPTION);

}

ftpAddress = ftpAddress1;

login = login1;

password = password1;

logger = logger1;

ipAddress = ipAddress1;

portTcp = portTcp1;

}

public void connect()

{

System.Console.WriteLine("Start connection on port=" + portTcp);

tcpClient = new TcpClient();

tcpClient.Connect(new IPEndPoint(IPAddress.Parse(ipAddress), portTcp));

System.Console.WriteLine("Have connect");

String answ = StreamOperation.readString(tcpClient.GetStream());

System.Console.WriteLine("New port=" + answ);

portTcp = int.Parse(answ);

tcpClient.Close();

tcpClient = new TcpClient();

tcpClient.Connect(new IPEndPoint(IPAddress.Parse(ipAddress), portTcp));

Thread treeReceiveThread = new Thread(new ThreadStart(this.receiveTree));

treeReceiveThread.Start();

connectToFtp();

}

public void sendLine(String command)

{

try

{

System.Console.WriteLine("Send message=" + command);

System.Text.ASCIIEncoding encoding = new System.Text.ASCIIEncoding();

byte[] array = encoding.GetBytes(command);

tcpClient.GetStream().Write(array, 0, array.Length);

}

catch (Exception ex)

{

System.Console.WriteLine(ex);

logger.addMessage(ClientException.invalid\_server\_address\_or\_port\_message);

}

}

public bool receiveAnswer()

{

try

{

String answer = StreamOperation.readString(tcpClient.GetStream());

System.Console.WriteLine("answer=" + answer);

if (answer.Equals(resultString))

{

//logger.addMessage("Receive answer from server=" + answer);

return true;

}

else

{

throw new Exception();

}

}

catch (Exception e)

{

System.Console.WriteLine(e);

logger.addMessage(ClientException.invalid\_received\_message);

return false;

}

}

public bool sendAnswerOnPing()

{

sendLine(pingAnswer\_str);

// logger.addMessage("Send answer on ping to server");

return receiveAnswer();

}

public bool createFolder(String folderName)

{

sendLine(createFolderString + delimiter + folderName);

//logger.addMessage("Create folder folderName="+folderName);

return receiveAnswer();

}

public bool deleteFolder(String folderName)

{

sendLine(deleteFolderString + delimiter + folderName);

return receiveAnswer();

}

public bool deleteFile(String fileName)

{

sendLine(deleteFileString + delimiter + fileName);

//logger.addMessage("Delete file fileName=" + fileName);

return receiveAnswer();

}

public bool renameFile(String oldName, String newName)

{

sendLine(renameFileString + delimiter + oldName + delimiter + newName);

//logger.addMessage("Rename file lastFileName=" + oldName+" next fileName="+newName);

return receiveAnswer();

}

public List<byte[]> downloadFile(String filename, String fileDestination)

{

sendLine(downloadFileString + delimiter + filename);

List<byte[]> byteList = new List<byte[]>();

int buffLength = 1024;

int length;

try

{

FileStream fileOut = new FileStream(fileDestination, FileMode.Create, FileAccess.Write);

while (true)

{

byte[] buff = new byte[buffLength];

length = tcpClient.GetStream().Read(buff, 0, buffLength);

byteList.Add(buff);

fileOut.Write(buff, 0, length);

if (length < buffLength)

{

break;

}

}

fileOut.Close();

//logger.addMessage("Download file fileName=" + fileDestination);

}

catch (Exception ex)

{

System.Console.WriteLine(ex);

logger.addMessage("Exception in downloading file fileName=" + fileDestination);

}

receiveAnswer();

return byteList;

}

public void uploadFile(String filename, String fileDestination)

{

sendLine(uploadFileString + delimiter + fileDestination);

System.Console.WriteLine("Start send file");

try

{

FileStream fileIn = new FileStream(filename, FileMode.Open, FileAccess.Read);

int length;

int buffLength = 1024;

byte[] buff = new byte[buffLength];

int n = 0;

while (true)

{

System.Console.WriteLine("next buff=" + n);

n++;

length = fileIn.Read(buff, 0, buffLength);

if (length != 0)

{

tcpClient.GetStream().Write(buff, 0, length);

}

else

{

break;

}

}

fileIn.Close();

}

catch (Exception ex)

{

System.Console.WriteLine("Exception in uploading");

System.Console.WriteLine(ex);

}

receiveAnswer();

}

public bool connectToFtp()

{

sendLine(connectString + delimiter + ftpAddress + delimiter + login + delimiter + password);

//logger.addMessage("Connect to ftp with address="+ftpAddress+" login="+login+" password="+password);

return receiveAnswer();

}

public void receiveTree()

{

if (tcpClientTree == null)

{

System.Console.WriteLine("Port for tree=" + (portTcp + 1));

TcpListener listener = new TcpListener(new IPEndPoint(IPAddress.Parse(ipAddress), portTcp + 1));

listener.Start();

tcpClientTree = listener.AcceptTcpClient();

}

try

{

while (true)

{

System.Console.WriteLine("Receive tree from port=" + portTcp);

MemoryStream ms = StreamOperation.readBytes(tcpClientTree.GetStream());

ms.Seek(0, SeekOrigin.Begin);

System.Console.WriteLine("have receive tree");

List<Node> nodeList = new List<Node>();

XmlSerializer serializer = new XmlSerializer(nodeList.GetType());

XmlReader reader = new XmlTextReader(ms);

nodeList = (List<Node>)serializer.Deserialize(reader);

for (int i = 0; i < nodeList.Count - 1; i++)

{

for (int j = 0; j < nodeList.ElementAt(i).foldersNames.Count; j++)

{

if (nodeList.ElementAt(i).foldersNames.ElementAt(j).Length == 0)

{

nodeList.ElementAt(i).foldersNames.Remove("");

}

}

for (int j = 0; j < nodeList.ElementAt(i).files.Count; j++)

{

if (nodeList.ElementAt(i).files.ElementAt(j).Length == 0)

{

nodeList.ElementAt(i).files.Remove("");

}

}

}

System.Console.WriteLine("Have build tree");

Node head = nodeList.ElementAt(nodeList.Count - 1);

head.buildChild(nodeList);

mainWindow.setTree(head);

System.Console.WriteLine("Have set tree");

}

}

catch (Exception e)

{

System.Console.WriteLine(e);

logger.addMessage("Exception in receiving tree");

}

}

}

}