**Server.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Net;

using System.Net.Sockets;

using System.Text;

using System.Threading;

namespace ChatServer

{

class Program

{

static ServerObject server; // сервер

static Thread listenThread; // потока для прослушивания

static void Main(string[] args)

{

try

{

server = new ServerObject();

listenThread = new Thread(new ThreadStart(server.Listen));

listenThread.Start(); //старт потока

}

catch (Exception ex)

{

server.Disconnect();

Console.WriteLine(ex.Message);

}

}

}

public class ServerObject

{

static TcpListener tcpListener; // сервер для прослушивания

List<ClientObject> clients = new List<ClientObject>(); // все подключения

protected internal void AddConnection(ClientObject clientObject)

{

clients.Add(clientObject);

}

protected internal void RemoveConnection(string id)

{

// получаем по id закрытое подключение

ClientObject client = clients.FirstOrDefault(c => c.Id == id);

// и удаляем его из списка подключений

if (client != null)

clients.Remove(client);

}

// прослушивание входящих подключений

protected internal void Listen()

{

try

{

tcpListener = new TcpListener(IPAddress.Any, 8888);

tcpListener.Start();

Console.WriteLine("Сервер запущен. Ожидание подключений...");

while (true)

{

TcpClient tcpClient = tcpListener.AcceptTcpClient();

Console.WriteLine("новый придурок");

ClientObject clientObject = new ClientObject(tcpClient, this);

Thread clientThread = new Thread(new ThreadStart(clientObject.Process));

clientThread.Start();

}

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

Disconnect();

}

}

// трансляция сообщения подключенным клиентам

protected internal void BroadcastMessage(string message, string id)

{

byte[] data = Encoding.Unicode.GetBytes(message);

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

{

if (clients[i].Id != id) // если id клиента не равно id отправляющего

{

clients[i].Stream.Write(data, 0, data.Length); //передача данных

}

}

}

// отключение всех клиентов

protected internal void Disconnect()

{

tcpListener.Stop(); //остановка сервера

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

{

clients[i].Close(); //отключение клиента

}

Environment.Exit(0); //завершение процесса

}

}

public class ClientObject

{

protected internal string Id { get; private set; }

protected internal NetworkStream Stream { get; private set; }

string userName;

TcpClient client;

ServerObject server; // объект сервера

public ClientObject(TcpClient tcpClient, ServerObject serverObject)

{

Id = Guid.NewGuid().ToString();

client = tcpClient;

server = serverObject;

serverObject.AddConnection(this);

}

public void Process()

{

try

{

Stream = client.GetStream();

// получаем имя пользователя

string message = GetMessage();

userName = message;

message = userName + " вошел в чат";

// посылаем сообщение о входе в чат всем подключенным пользователям

server.BroadcastMessage(message, this.Id);

Console.WriteLine(message);

// в бесконечном цикле получаем сообщения от клиента

while (true)

{

try

{

message = GetMessage();

message = String.Format("{0}: {1}", userName, message);

Console.WriteLine(message);

server.BroadcastMessage(message, this.Id);

}

catch

{

message = String.Format("{0}: покинул чат", userName);

Console.WriteLine(message);

server.BroadcastMessage(message, this.Id);

break;

}

}

}

catch (Exception e)

{

Console.WriteLine(e.Message);

}

finally

{

// в случае выхода из цикла закрываем ресурсы

server.RemoveConnection(this.Id);

Close();

}

}

// чтение входящего сообщения и преобразование в строку

private string GetMessage()

{

byte[] data = new byte[64]; // буфер для получаемых данных

StringBuilder builder = new StringBuilder();

int bytes = 0;

do

{

bytes = Stream.Read(data, 0, data.Length);

builder.Append(Encoding.Unicode.GetString(data, 0, bytes));

}

while (Stream.DataAvailable);

return builder.ToString();

}

// закрытие подключения

protected internal void Close()

{

if (Stream != null)

Stream.Close();

if (client != null)

client.Close();

}

}

}