CODIGO SERVIDOR

using System;

using System.Collections;

using System.Net;

using System.Net.NetworkInformation;

using System.Net.Sockets;

using System.Text;

using System.Threading;

using System.Windows.Forms;

namespace Servidor

{

/// <summary>

/// Servidor para administrar el chat.

/// </summary>

public partial class frmServidor : Form

{

#region [ Variables ]

/// <summary>

/// La clase TcpListener proporciona métodos sencillos para escuchar y

/// aceptar solicitudes de conexión entrantes en modo de bloqueo sincrónico.

/// </summary>

private TcpListener servidor;

/// <summary>

/// Almacenará el nombre de los clientes conectados

/// </summary>

private static Hashtable clientes\_conectados;

/// <summary>

/// Mensaje recibido del cliente o que se enviará al cliente

/// </summary>

private String mensajeCliente;

/// <summary>

/// Mensaje que se envía a la pantalla del Servidor.

/// </summary>

private String mensajeChat;

/// <summary>

/// Variable boolean que simula el patron singleton al pulsar el boton iniciar

/// </summary>

private Boolean btnActivar = true;

/// <summary>

/// Variable para el manejo del id del hijo que maneja el servidor

/// </summary>

private int idThread;

/// <summary>

/// Definimos la variable a utilizar en la creacion de los hilos

/// </summary>

Thread InitServer;

public static int contadorAmistad = 0;

public static int contadorAmor = 0;

public static int contadorCulinaria = 0;

public static int contadorConocimiento = 0;

public static Boolean online = false;

#endregion

#region [ Constructor ]

public frmServidor()

{

InitializeComponent();

}

#endregion

#region [ Eventos ]

/// <summary>

/// Inicia el servidor de chat

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void btnIniciar\_Click(object sender, EventArgs e)

{

if (btnActivar && !online)

{

online = true;

btnActivar = false;

btnIniciar.Enabled = false;

btnCerrar.Enabled = true;

InitServer = new Thread(iniciarServidor);

InitServer.Start();

}

}

/// <summary>

/// Detiene el servidor de chat

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void btnCerrar\_Click(object sender, EventArgs e)

{

if (servidor != null && !servidor.Server.Connected)

{

online = false;

btnActivar = true;

btnIniciar.Enabled = false;

btnCerrar.Enabled = true;

this.Hide();

}

}

#endregion

#region [ Métodos ]

/// <summary>

/// Difunde el mensaje de un usuario a todos los usuarios conectados a la sala.

/// </summary>

/// <param name="mensaje">Mensaje del Usuario</param>

/// <param name="nombre">Nombre del usuario</param>

/// <param name="flag">Bandera que se utiliza para determinar si se agrega el texto

/// "dice" al mensaje enviado por el usuario</param>

///

public void iniciarServidor()

{

try

{

//Inicalizamos nuestro hashtable

clientes\_conectados = new Hashtable();

//Inicializamos un nuevo servidor en la IP y puerto seleccionado.

servidor = new TcpListener(IPAddress.Parse(txtIP.Text), int.Parse(txtPuerto.Text));

//Iniciamos el servidor

servidor.Start();

//Indicamos que el servidor esta listo para aceptar peticiones de los clientes

mensajeChat = "Ha iniciado el Servidor";

Mensaje();

//definimos la variable de acepcacion de clientes

TcpClient cliente;

//Sólo aceptaremos mensajes de máximo 256 caracteres

Byte[] bytesCliente = new Byte[256];

NetworkStream streamCliente;

Chat chat;

while (true)

{

//Acepta la petición de un cliente

cliente = servidor.AcceptTcpClient();

//Leemos el mensaje del cliente

streamCliente = cliente.GetStream();

//bytesCliente sólo acepta mensajes de máximo 160 caracteres

streamCliente.Read(bytesCliente, 0, bytesCliente.Length);

//Traducimos el stream de bytes a un string en codificación ASCII

mensajeCliente = Encoding.ASCII.GetString(bytesCliente, 0, bytesCliente.Length);

mensajeCliente = mensajeCliente.Substring(0, mensajeCliente.IndexOf("$"));

//Verificamos si ya existe ese nombre de usuario.

while(clientes\_conectados.ContainsKey(mensajeCliente))

{

Byte[] bytes = Encoding.ASCII.GetBytes("/\*usuario duplicado\*/");

//transmitimos el mensaje

streamCliente.Write(bytes, 0, bytes.Length);

streamCliente.Flush();

mensajeCliente = mensajeCliente + "1";

}

//Realmente no hacemos ninguna validación posterior ni se manda un mensaje al

//usuario pero por funcionalidad agregaremos esta condición.

clientes\_conectados.Add(mensajeCliente, cliente);

try

{

switch (mensajeCliente.Substring(0, (mensajeCliente.IndexOf(" "))))

{

case "Amistad":

contadorAmistad++;

break;

case "Amor":

contadorAmor++;

break;

case "Culinaria":

contadorCulinaria++;

break;

case "Conocimiento":

contadorConocimiento++;

break;

}

}catch(Exception ex)

{}

mensajeChat = string.Format("{0} se ha unido al servidor", mensajeCliente);

Mensaje();

//Mandamos el mensaje a todos los clientes el mensaje

DifundirATodos(mensajeCliente, mensajeCliente, false);

//Ciclamos el proceso para que se quede el servidor en espera de nuevas conexiones o mensajes

chat = new Chat(cliente, mensajeCliente);

}

}

catch (Exception ex)

{

}

finally

{

if(servidor!=null)

servidor.Stop();

}

}

public static void DifundirATodos(string mensaje, string nombre, bool flag)

{

try

{

if (online)

{

if ("/\*Listar Usuarios\*/" == mensaje)

{

switch(nombre.Substring(0,(nombre.IndexOf(" "))))

{

case "Amistad":

mensaje = "La cantidad de usuarios conectados es: " + contadorAmistad;

break;

case"Culinaria":

mensaje = "La cantidad de usuarios conectados es: " + contadorCulinaria;

break;

case "Amor":

mensaje = "La cantidad de usuarios conectados es: " + contadorAmor;

break;

case "Conocimiento":

mensaje = "La cantidad de usuarios conectados es: " + contadorConocimiento;

break;

}

}

//Por cada cliente

foreach (DictionaryEntry Item in clientes\_conectados)

{

Byte[] bytes = null;

TcpClient cliente;

cliente = (TcpClient)Item.Value;

NetworkStream streamCliente = cliente.GetStream();

if (flag == true)

bytes = Encoding.ASCII.GetBytes(nombre + " Dice : " + mensaje);

else

bytes = Encoding.ASCII.GetBytes(nombre + "Ha iniciado sesión ");

//transmitimos el mensaje

streamCliente.Write(bytes, 0, bytes.Length);

streamCliente.Flush();

}

}

}

catch (Exception ex)

{

}

}

public void Mensaje()

{

if (this.InvokeRequired)

this.Invoke(new MethodInvoker(Mensaje));

else

txtChat.Text = txtChat.Text + Environment.NewLine + " -> " + mensajeChat;

}

#endregion

}

public class Chat

{

#region [ Variables ]

TcpClient clienteChat;

string nombreUsuario;

#endregion

public Chat(TcpClient cliente, string Usuario)

{

clienteChat = cliente;

nombreUsuario = Usuario;

//Iniciamos un nuevo proceso que cicle la espera de mensajes nuevos por parte de los clientes.

Thread ctThread = new Thread(doChat);

ctThread.Start();

}

/// <summary>

/// Cicla el proceso indefinidamente para que el servidor quede a la espera de nuevos mensajes

/// por parte de los clientes.

/// </summary>

private void doChat()

{

byte[] bytesFrom = new byte[160];

string mensajeCliente = null;

while (true)

{

try

{

NetworkStream networkStream = clienteChat.GetStream();

networkStream.Read(bytesFrom, 0, bytesFrom.Length);

mensajeCliente = System.Text.Encoding.ASCII.GetString(bytesFrom);

mensajeCliente = mensajeCliente.Substring(0, mensajeCliente.IndexOf("$"));

//Difundimos el mensaje a todos los clientes

frmServidor.DifundirATodos(mensajeCliente, nombreUsuario, true);

}

catch (Exception ex)

{

}

}

}

}

}