|  |
| --- |
| Пермский филиал федерального государственного автономного образовательного учреждения высшего образования  «Национальный исследовательский университет  «Высшая школа экономики»      [*Факультет социально-экономических и компьютерных наук*](https://perm.hse.ru/scs/) |
| Кирьянов Сергей Вячеславович Платонов Владислав Алексеевич      **Лабораторная работа № 4**      студента образовательной программы «Программная инженерия» по направлению подготовки *09.03.04 Программная инженерия*      Проверил(а)  Преподаватель кафедры информационных технологий в бизнесе    Е. Б. Замятина |

**2024**

**Задание 3**

Изучите разработанное приложение. Запустите исполняемый файл сервера и клиента на одной машине, затем перенесите один из компонентов распределенного приложения на другой вычислительный узел (лучше на ноутбук, т.к. администратор ЛВС вуза может запретить обмен сообщения между различными узлами ЛВС) и попробуйте его запустить. Объясните, почему приложение перестало работать? Запустите несколько клиентов на одной машине с сервером и попробуйте отправить сообщения серверу. В процессе изучения (тестирования) приложения просмотрите очереди сообщений компьютера и убедитесь, что сообщения действительно приходят в очередь.

Первой возможной причиной недоступности работы как клиентской, так и серверной частей на разных вычислительных узлах - выключенная служба "сервер очереди сообщений Майкрософт (MSMQ)”. На всех вычислительных узлах, участвующих в обмене сообщениями, должна быть включена вышеописанная служба.

Второй причиной недоступности клиентского или серверного приложений на разных вычислительных узлах могут послужить сетевые настройки, которые не позволяют обмен сообщениями между различными узлами.

Проконтролировать создание очередей на компьютере можно перейдя в каталог по фиксированному пути “C:\Windows\System32\msmq\storage\lqs”. В данном каталоге, помимо созданных в ходе работы с клиентским и серверным приложениями, содержатся очереди сообщений, используемые системой.

Перечислим название очередей, хранящихся в вышеописанном каталоге:

1. QueueName=\private$\admin\_queue$
2. QueueName=\private$\order\_queue$
3. QueueName=\private$\triggers\_queue$
4. QueueName=\private$\serverqueue
5. QueueName=\private$\windows10pc
6. QueueName=\private$\win10pc2

Примечание: очереди в списке выше под номерами 4, 5 и 6 - созданные очереди в ходе работы клиентских и серверных приложений.

**Задание 4**

Запустите несколько серверов на одной машине. Отправьте им сообщения от нескольких клиентов. Объясните, почему приложение перестало корректно работать.

Серверное приложение при инициализации берет имя для создания или открытия очереди из переменной “path”, значение которой зафиксировано в коде программы и никак не меняется. Поэтому каждый вновь запущенный сервер попытается работать с файлом очереди, который уже находиться в работе у первого по счету запущенного серверного приложения.

**Задание 5**

Модифицируйте приложение так, чтобы существовала возможность на сервере идентифицировать клиентов не по имени вычислительного узла, а по нику/логину пользователя.

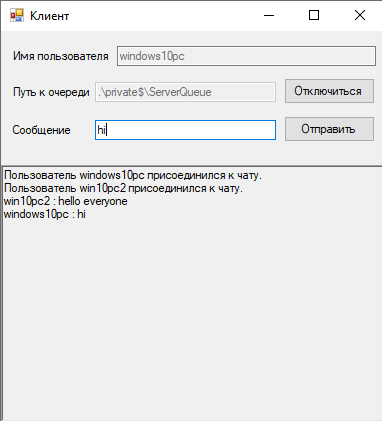
Для добавления возможности идентификации клиента по нику/логину пользователя для клиентского приложения добавлено:

1. Поле для ввода имени пользователя.
2. Проверка на заполнение имени пользователя перед подключением.

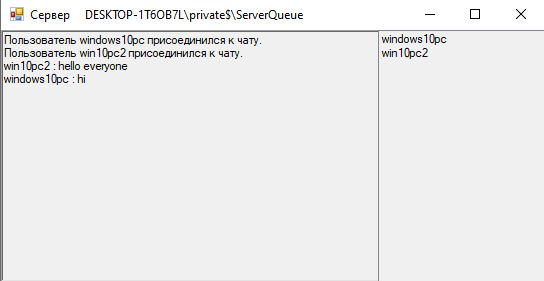
Для добавления возможности идентификации клиента по нику/логину пользователя для серверного приложения добавлено:

1. Список в правой части приложения для просмотра подключенных пользователей
2. Вывод сообщения в формате: имя пользователя : сообщение.

Изображения с обновленным интерфейсом представлены на рисунке 1 и рисунке 2.



*Рисунок 1 - “Клиентское приложение 5”*



*Рисунок 2 - “Серверное приложение 5”*

**Задание 6**

Модифицируйте приложение так, чтобы получился полноценный чат. Клиент может отправлять сообщения всем клиентам, участвующим в беседе. Для этого каждый клиент должен иметь возможность просмотра всех сообщений от всех клиентов, а сервер должен содержать список клиентов, которые хотят участвовать в беседе, чтобы каждый раз выполнять им рассылку сообщений.

Модифицируем клиентское и серверное приложения для рассылки сервером сообщений подключенным клиентам и приема и обработки клиентом сообщений сервера.

Внесем следующие изменения в клиентское приложение:

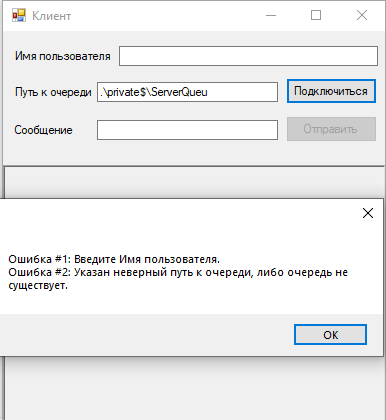
1. Добавим поток, отвечающий за работу с очередью сообщений.
2. Клиентскую очередь сообщений, в которую будет производиться запись со стороны сервера.
3. Текстовое поле, для отображения полученных сообщений.

Внесем следующие изменения в серверное приложение:

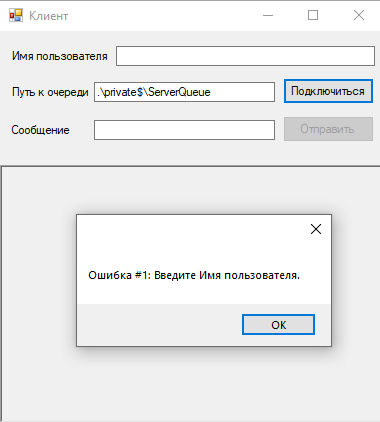
1. Словарь для хранения информации user\_name клиента и pc\_name клиента.
2. Словарь для хранения информации user\_name клиента и ссылка на элемент ListView
3. Цикл для перебора словаря и отправки клиентам сообщений в клиентскую очередь при получении нового сообщения в очередь сервера.

Для удобства создадим и подключим библиотеку “MsgJsonLibrary”для удобной работы с сообщениями в формате “json”.

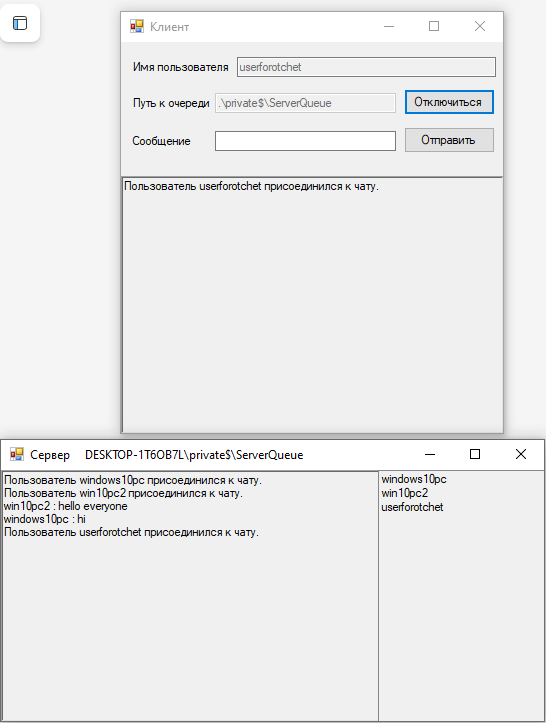
Изображения работы с программой представлены ниже.



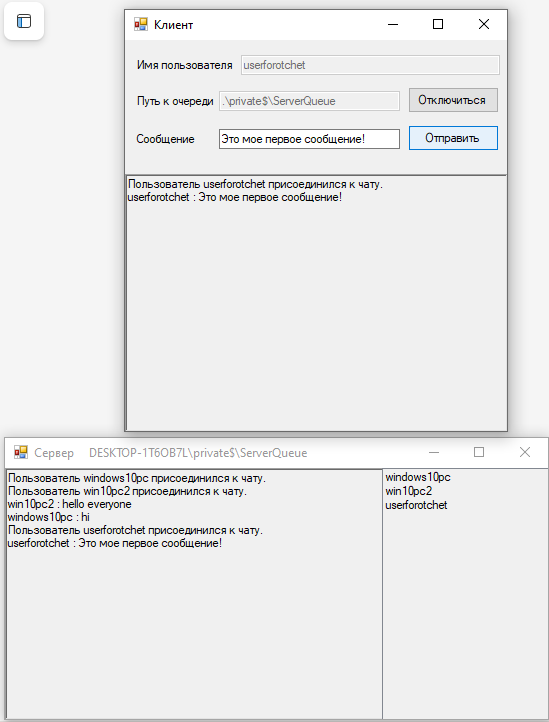
*Рисунок 3 - “Ошибка #1 и Ошибка #2”*



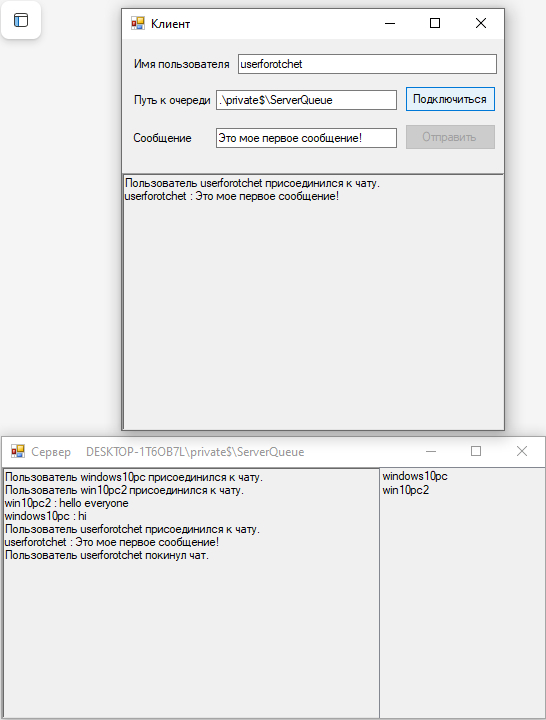
*Рисунок 4 - “Ошибка #1”*



*Рисунок 5 - “Присоединение нового клиента”*



*Рисунок 6 - “Сообщение нового клиента”*



*Рисунок 7 - “Отключение нового клиента”*

Приложение разрабатывалось на платформе WinForms. Версия .NETFramework = 4.8.1.

Репозиторий с файлами программ - <https://github.com/vl4xd/hse_tiavv_labs/tree/main/lab_4>.

Код, измененных и добавленных для работы программы файлов, представлен ниже.

**Код программы**[**Client.cs**](https://github.com/vl4xd/hse_tiavv_labs/blob/main/lab_4/MSMQClient/MSMQClient/Client.cs)

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

using System.Net;

using System.Net.Sockets;

using System.IO;

using System.Messaging;

using MsgJsonLibrary;

using System.Runtime.InteropServices;

using System.Net.NetworkInformation;

using System.Threading;

namespace MSMQ

{

public partial class frmMain : Form

{

private MessageQueue q\_client = null; // очередь сообщений

private Thread t = null; // поток, отвечающий за работу с очередью сообщений

private MessageQueue q\_server = null; // очередь сообщений, в которую будет производиться запись сообщений

bool user\_connected = false; // поле статуса подлкючения пользователя к серверу (флаг, указывающий продолжается ли работа с мэйлслотом)

// конструктор формы

public frmMain()

{

InitializeComponent();

}

private void ChangeInterface()

{

user\_connected = !user\_connected;

tbPath.Enabled = !user\_connected;

tbUserName.Enabled = !user\_connected;

btnSend.Enabled = user\_connected;

btnConnectDisconnet.Text = !user\_connected ? "Подключиться" : "Отключиться";

}

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

{

if (user\_connected) // если пользователь подключен к серверу

{

ChangeInterface();

CloseClientQueue();

}

else

{

int count\_error = 0;

string message\_error = "";

if (tbUserName.Text.Length == 0)

message\_error += $"Ошибка #{++count\_error}: Введите Имя пользователя.\n";

if (MessageQueue.Exists(tbPath.Text))

{

// если очередь, путь к которой указан в поле tbPath существует, то открываем ее

q\_server = new MessageQueue(tbPath.Text);

}

else

message\_error += $"Ошибка #{++count\_error}: Указан неверный путь к очереди, либо очередь не существует.\n";

if (message\_error.Length != 0)

MessageBox.Show(message\_error);

else

{

ChangeInterface();

OpenClientQueue();

}

}

}

private void OpenClientQueue()

{

string path = Dns.GetHostName() + $"\\private$\\{tbUserName.Text}"; // путь к очереди сообщений, Dns.GetHostName() - метод, возвращающий имя текущей машины

// если очередь сообщений с указанным путем существует, то открываем ее, иначе создаем новую

if (MessageQueue.Exists(path))

q\_client = new MessageQueue(path);

else

q\_client = MessageQueue.Create(path);

// задаем форматтер сообщений в очереди

q\_client.Formatter = new XmlMessageFormatter(new Type[] { typeof(String) });

// создание потока, отвечающего за работу с очередью сообщений

Thread t = new Thread(ReceiveMessage);

t.Start();

//

MsgJsonMSMQ info = new MsgJsonMSMQ(user\_connected, !user\_connected, Dns.GetHostName(), tbUserName.Text, "Text");

string info\_json = MsgJsonMSMQ.MsgJsonSerialize(info);

q\_server.Send("", info\_json);

}

private void CloseClientQueue()

{

//

MsgJsonMSMQ info = new MsgJsonMSMQ(user\_connected, !user\_connected, Dns.GetHostName(), tbUserName.Text, "Text");

string info\_json = MsgJsonMSMQ.MsgJsonSerialize(info);

if (MessageQueue.Exists(tbPath.Text))

q\_server.Send("", info\_json);

if (q\_client != null)

{

//MessageQueue.Delete(q.Path); // в случае необходимости удаляем очередь сообщений

}

if (t != null)

{

t.Abort(); // завершаем поток

}

}

private void ReceiveMessage()

{

if (q\_client == null)

return;

System.Messaging.Message msg = null;

// входим в бесконечный цикл работы с очередью сообщений

while (user\_connected)

{

if (q\_client.Peek() != null) // если в очереди есть сообщение, выполняем его чтение, интервал до следующей попытки чтения равен 10 секундам

msg = q\_client.Receive(TimeSpan.FromSeconds(10.0));

rtbMessages.Invoke((MethodInvoker)delegate

{

if (msg == null) return;

rtbMessages.Text += $"{msg.Body}\n";

});

Thread.Sleep(500); // приостанавливаем работу потока перед тем, как приcтупить к обслуживанию очередного клиента

}

}

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

{

MsgJsonMSMQ info = new MsgJsonMSMQ(false, false, Dns.GetHostName(), tbUserName.Text, "Text");

string info\_json = MsgJsonMSMQ.MsgJsonSerialize(info);

// выполняем отправку сообщения в очередь

q\_server.Send(tbMessage.Text, info\_json);

}

}

}

**Код программы**[**Client.designer.cs**](https://github.com/vl4xd/hse_tiavv_labs/blob/main/lab_4/MSMQClient/MSMQClient/Client.designer.cs)

namespace MSMQ

{

partial class frmMain

{

/// <summary>

/// Требуется переменная конструктора.

/// </summary>

private System.ComponentModel.IContainer components = null;

/// <summary>

/// Освободить все используемые ресурсы.

/// </summary>

/// <param name="disposing">истинно, если управляемый ресурс должен быть удален; иначе ложно.</param>

protected override void Dispose(bool disposing)

{

if (disposing && (components != null))

{

components.Dispose();

}

base.Dispose(disposing);

}

#region Код, автоматически созданный конструктором форм Windows

/// <summary>

/// Обязательный метод для поддержки конструктора - не изменяйте

/// содержимое данного метода при помощи редактора кода.

/// </summary>

private void InitializeComponent()

{

this.btnSend = new System.Windows.Forms.Button();

this.tbMessage = new System.Windows.Forms.TextBox();

this.lblIP = new System.Windows.Forms.Label();

this.btnConnectDisconnet = new System.Windows.Forms.Button();

this.lblMessage = new System.Windows.Forms.Label();

this.tbPath = new System.Windows.Forms.TextBox();

this.tbUserName = new System.Windows.Forms.TextBox();

this.label1 = new System.Windows.Forms.Label();

this.rtbMessages = new System.Windows.Forms.RichTextBox();

this.SuspendLayout();

//

// btnSend

//

this.btnSend.Enabled = false;

this.btnSend.Location = new System.Drawing.Point(425, 131);

this.btnSend.Margin = new System.Windows.Forms.Padding(4, 5, 4, 5);

this.btnSend.Name = "btnSend";

this.btnSend.Size = new System.Drawing.Size(136, 40);

this.btnSend.TabIndex = 3;

this.btnSend.Text = "Отправить";

this.btnSend.UseVisualStyleBackColor = true;

this.btnSend.Click += new System.EventHandler(this.btnSend\_Click);

//

// tbMessage

//

this.tbMessage.Location = new System.Drawing.Point(141, 137);

this.tbMessage.Margin = new System.Windows.Forms.Padding(4, 5, 4, 5);

this.tbMessage.Name = "tbMessage";

this.tbMessage.Size = new System.Drawing.Size(270, 26);

this.tbMessage.TabIndex = 1;

//

// lblIP

//

this.lblIP.AutoSize = true;

this.lblIP.Location = new System.Drawing.Point(13, 83);

this.lblIP.Margin = new System.Windows.Forms.Padding(4, 0, 4, 0);

this.lblIP.Name = "lblIP";

this.lblIP.Size = new System.Drawing.Size(127, 20);

this.lblIP.TabIndex = 2;

this.lblIP.Text = "Путь к очереди";

//

// btnConnectDisconnet

//

this.btnConnectDisconnet.Location = new System.Drawing.Point(425, 72);

this.btnConnectDisconnet.Margin = new System.Windows.Forms.Padding(4, 5, 4, 5);

this.btnConnectDisconnet.Name = "btnConnectDisconnet";

this.btnConnectDisconnet.Size = new System.Drawing.Size(136, 40);

this.btnConnectDisconnet.TabIndex = 2;

this.btnConnectDisconnet.Text = "Подключиться";

this.btnConnectDisconnet.UseVisualStyleBackColor = true;

this.btnConnectDisconnet.Click += new System.EventHandler(this.btnConnectDisconnect\_Click);

//

// lblMessage

//

this.lblMessage.AutoSize = true;

this.lblMessage.Location = new System.Drawing.Point(13, 141);

this.lblMessage.Margin = new System.Windows.Forms.Padding(4, 0, 4, 0);

this.lblMessage.Name = "lblMessage";

this.lblMessage.Size = new System.Drawing.Size(96, 20);

this.lblMessage.TabIndex = 2;

this.lblMessage.Text = "Сообщение";

//

// tbPath

//

this.tbPath.Location = new System.Drawing.Point(141, 78);

this.tbPath.Margin = new System.Windows.Forms.Padding(4, 5, 4, 5);

this.tbPath.Name = "tbPath";

this.tbPath.Size = new System.Drawing.Size(270, 26);

this.tbPath.TabIndex = 0;

this.tbPath.Text = ".\\private$\\ServerQueue";

//

// tbUserName

//

this.tbUserName.Location = new System.Drawing.Point(174, 23);

this.tbUserName.Margin = new System.Windows.Forms.Padding(4, 5, 4, 5);

this.tbUserName.Name = "tbUserName";

this.tbUserName.Size = new System.Drawing.Size(386, 26);

this.tbUserName.TabIndex = 4;

//

// label1

//

this.label1.AutoSize = true;

this.label1.Location = new System.Drawing.Point(13, 28);

this.label1.Margin = new System.Windows.Forms.Padding(4, 0, 4, 0);

this.label1.Name = "label1";

this.label1.Size = new System.Drawing.Size(153, 20);

this.label1.TabIndex = 5;

this.label1.Text = "Имя пользователя";

//

// rtbMessages

//

this.rtbMessages.Anchor = ((System.Windows.Forms.AnchorStyles)((((System.Windows.Forms.AnchorStyles.Top | System.Windows.Forms.AnchorStyles.Bottom)

| System.Windows.Forms.AnchorStyles.Left)

| System.Windows.Forms.AnchorStyles.Right)));

this.rtbMessages.Location = new System.Drawing.Point(0, 206);

this.rtbMessages.Margin = new System.Windows.Forms.Padding(4, 5, 4, 5);

this.rtbMessages.Name = "rtbMessages";

this.rtbMessages.ReadOnly = true;

this.rtbMessages.Size = new System.Drawing.Size(573, 395);

this.rtbMessages.TabIndex = 6;

this.rtbMessages.Text = "";

//

// frmMain

//

this.AutoScaleDimensions = new System.Drawing.SizeF(9F, 20F);

this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;

this.ClientSize = new System.Drawing.Size(573, 602);

this.Controls.Add(this.rtbMessages);

this.Controls.Add(this.tbUserName);

this.Controls.Add(this.label1);

this.Controls.Add(this.tbPath);

this.Controls.Add(this.lblMessage);

this.Controls.Add(this.lblIP);

this.Controls.Add(this.tbMessage);

this.Controls.Add(this.btnConnectDisconnet);

this.Controls.Add(this.btnSend);

this.Margin = new System.Windows.Forms.Padding(4, 5, 4, 5);

this.Name = "frmMain";

this.StartPosition = System.Windows.Forms.FormStartPosition.CenterScreen;

this.Text = "Клиент";

this.ResumeLayout(false);

this.PerformLayout();

}

#endregion

private System.Windows.Forms.Button btnSend;

private System.Windows.Forms.TextBox tbMessage;

private System.Windows.Forms.Label lblIP;

private System.Windows.Forms.Button btnConnectDisconnet;

private System.Windows.Forms.Label lblMessage;

private System.Windows.Forms.TextBox tbPath;

private System.Windows.Forms.TextBox tbUserName;

private System.Windows.Forms.Label label1;

private System.Windows.Forms.RichTextBox rtbMessages;

}

}

**Код программы**[**Server.cs**](https://github.com/vl4xd/hse_tiavv_labs/blob/main/lab_4/MSMQServer/MSMQServer/Server.cs)

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

using System.Net;

using System.Net.Sockets;

using System.Threading;

using System.Messaging;

using MsgJsonLibrary;

using System.IO;

namespace MSMQ

{

public partial class frmMain : Form

{

private MessageQueue q = null; // очередь сообщений

private Thread t = null; // поток, отвечающий за работу с очередью сообщений

private bool \_continue = true; // флаг, указывающий продолжается ли работа с мэйлслотом

// словарь для хранения информации user\_name клиента и pc\_name клиента

private Dictionary<string, string> clients = new Dictionary<string, string>();

// словарь для хранения информации user\_name клиента и ссылка на элемент ListView

private Dictionary<string, ListViewItem> lvClients\_link = new Dictionary<string, ListViewItem>();

// конструктор формы

public frmMain()

{

InitializeComponent();

string path = Dns.GetHostName() + "\\private$\\ServerQueue"; // путь к очереди сообщений, Dns.GetHostName() - метод, возвращающий имя текущей машины

// если очередь сообщений с указанным путем существует, то открываем ее, иначе создаем новую

if (MessageQueue.Exists(path))

q = new MessageQueue(path);

else

q = MessageQueue.Create(path);

// задаем форматтер сообщений в очереди

q.Formatter = new XmlMessageFormatter(new Type[] { typeof(String) });

// вывод пути к очереди сообщений в заголовок формы, чтобы можно было его использовать для ввода имени в форме клиента, запущенного на другом вычислительном узле

this.Text += " " + q.Path;

// создание потока, отвечающего за работу с очередью сообщений

Thread t = new Thread(ReceiveMessage);

t.Start();

}

// получение сообщения

private void ReceiveMessage()

{

if (q == null)

return;

System.Messaging.Message msg = null;

// входим в бесконечный цикл работы с очередью сообщений

while (\_continue)

{

if (q.Peek() != null) // если в очереди есть сообщение, выполняем его чтение, интервал до следующей попытки чтения равен 10 секундам

msg = q.Receive(TimeSpan.FromSeconds(10.0));

string send\_message = "";

rtbMessages.Invoke((MethodInvoker)delegate

{

if (msg == null) return;

MsgJsonMSMQ info = MsgJsonMSMQ.MsgJsonDeserialize(msg.Label);

if (info.Is\_connection)

{

send\_message = $"Пользователь {info.User\_name} присоединился к чату.";

rtbMessages.Text += send\_message;

ListViewItem new\_client = new ListViewItem(info.User\_name);

clients.Add(info.User\_name, info.User\_pc\_name);

lvClients\_link.Add(info.User\_name, new\_client);

lvClients.Invoke((MethodInvoker)delegate

{

lvClients.Items.Add(new\_client);

});

}

else if (info.Is\_disconnection)

{

send\_message = $"Пользователь {info.User\_name} покинул чат.";

rtbMessages.Text += send\_message;

clients.Remove(info.User\_name);

lvClients.Invoke((MethodInvoker)delegate

{

lvClients.Items.Remove(lvClients\_link[info.User\_name]);

});

lvClients\_link.Remove(info.User\_name);

}

else

{

send\_message = $"{info.User\_name} : {msg.Body}";

rtbMessages.Text += send\_message;

}

rtbMessages.Text += "\n";

});

List<string> remove\_clients = new List<string>();

foreach (var client in clients)

{

string path\_client = $"{client.Value}\\private$\\{client.Key}";

if (MessageQueue.Exists(path\_client))

{

// если очередь, путь существует, то открываем ее

MessageQueue q\_client = new MessageQueue(path\_client);

q\_client.Send(send\_message);

}

else

{

send\_message = $"Пользователь {client.Key} покинул чат.";

rtbMessages.Invoke((MethodInvoker)delegate

{

rtbMessages.Text += send\_message;

});

remove\_clients.Add(client.Key);

lvClients.Invoke((MethodInvoker)delegate

{

lvClients.Items.Remove(lvClients\_link[client.Key]);

});

lvClients\_link.Remove(client.Key);

}

}

foreach (var client\_name in remove\_clients)

{

clients.Remove(client\_name);

}

Thread.Sleep(500); // приостанавливаем работу потока перед тем, как приcтупить к обслуживанию очередного клиента

}

}

private void frmMain\_FormClosing(object sender, FormClosingEventArgs e)

{

\_continue = false; // сообщаем, что работа с очередью сообщений завершена

if (q != null)

{

//MessageQueue.Delete(q.Path); // в случае необходимости удаляем очередь сообщений

}

if (t != null)

{

t.Abort(); // завершаем поток

}

}

}

}

**Код программы**[**Server.designer.cs**](https://github.com/vl4xd/hse_tiavv_labs/blob/main/lab_4/MSMQServer/MSMQServer/Server.designer.cs)

namespace MSMQ

{

partial class frmMain

{

/// <summary>

/// Требуется переменная конструктора.

/// </summary>

private System.ComponentModel.IContainer components = null;

/// <summary>

/// Освободить все используемые ресурсы.

/// </summary>

/// <param name="disposing">истинно, если управляемый ресурс должен быть удален; иначе ложно.</param>

protected override void Dispose(bool disposing)

{

if (disposing && (components != null))

{

components.Dispose();

}

base.Dispose(disposing);

}

#region Код, автоматически созданный конструктором форм Windows

/// <summary>

/// Обязательный метод для поддержки конструктора - не изменяйте

/// содержимое данного метода при помощи редактора кода.

/// </summary>

private void InitializeComponent()

{

this.rtbMessages = new System.Windows.Forms.RichTextBox();

this.lvClients = new System.Windows.Forms.ListView();

this.SuspendLayout();

//

// rtbMessages

//

this.rtbMessages.Anchor = ((System.Windows.Forms.AnchorStyles)((((System.Windows.Forms.AnchorStyles.Top | System.Windows.Forms.AnchorStyles.Bottom)

| System.Windows.Forms.AnchorStyles.Left)

| System.Windows.Forms.AnchorStyles.Right)));

this.rtbMessages.Location = new System.Drawing.Point(0, 0);

this.rtbMessages.Margin = new System.Windows.Forms.Padding(4, 5, 4, 5);

this.rtbMessages.Name = "rtbMessages";

this.rtbMessages.ReadOnly = true;

this.rtbMessages.Size = new System.Drawing.Size(567, 315);

this.rtbMessages.TabIndex = 0;

this.rtbMessages.Text = "";

//

// lvClients

//

this.lvClients.Anchor = ((System.Windows.Forms.AnchorStyles)((((System.Windows.Forms.AnchorStyles.Top | System.Windows.Forms.AnchorStyles.Bottom)

| System.Windows.Forms.AnchorStyles.Left)

| System.Windows.Forms.AnchorStyles.Right)));

this.lvClients.BackColor = System.Drawing.SystemColors.Control;

this.lvClients.HideSelection = false;

this.lvClients.Location = new System.Drawing.Point(566, 0);

this.lvClients.Name = "lvClients";

this.lvClients.Size = new System.Drawing.Size(249, 315);

this.lvClients.TabIndex = 1;

this.lvClients.UseCompatibleStateImageBehavior = false;

this.lvClients.View = System.Windows.Forms.View.List;

//

// frmMain

//

this.AutoScaleDimensions = new System.Drawing.SizeF(9F, 20F);

this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;

this.ClientSize = new System.Drawing.Size(815, 315);

this.Controls.Add(this.lvClients);

this.Controls.Add(this.rtbMessages);

this.Margin = new System.Windows.Forms.Padding(4, 5, 4, 5);

this.Name = "frmMain";

this.StartPosition = System.Windows.Forms.FormStartPosition.CenterScreen;

this.Text = "Сервер";

this.FormClosing += new System.Windows.Forms.FormClosingEventHandler(this.frmMain\_FormClosing);

this.ResumeLayout(false);

}

#endregion

private System.Windows.Forms.RichTextBox rtbMessages;

private System.Windows.Forms.ListView lvClients;

}

}

**Код программы**[**MsgJsonMSMQ.cs**](https://github.com/vl4xd/hse_tiavv_labs/blob/main/lab_4/MsgJsonLibrary/MsgJsonMSMQ.cs)

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Text.Json;

namespace MsgJsonLibrary

{

public class MsgJsonMSMQ

{

private bool \_is\_connection;

private bool \_is\_disconnection;

private string \_user\_pc\_name;

private string \_user\_name;

private string \_user\_message;

public bool Is\_connection

{

get => \_is\_connection;

set => \_is\_connection = value;

}

public bool Is\_disconnection

{

get => \_is\_disconnection;

set => \_is\_disconnection = value;

}

public string User\_pc\_name

{

get => \_user\_pc\_name;

set => \_user\_pc\_name = value;

}

public string User\_name

{

get => \_user\_name;

set => \_user\_name = value;

}

public string User\_message

{

get => \_user\_message;

set => \_user\_message = value;

}

public MsgJsonMSMQ() { }

public MsgJsonMSMQ(bool is\_connection, bool is\_disconnection, string user\_pc\_name, string user\_name, string user\_message)

{

Is\_connection = is\_connection;

Is\_disconnection = is\_disconnection;

User\_pc\_name = user\_pc\_name;

User\_name = user\_name;

User\_message = user\_message;

}

/// <summary>

/// Десериализация строки json формата через перегруженный конструктор класса MsgJsonMSMQ

/// </summary>

/// <param name="json\_string">строка json формата с полями класса MsgJsonMSMQ</param>

public MsgJsonMSMQ(string json\_string)

{

MsgJsonMSMQ temp = JsonSerializer.Deserialize<MsgJsonMSMQ>(json\_string);

Is\_connection = temp.Is\_connection;

Is\_disconnection = temp.Is\_disconnection;

User\_pc\_name = temp.User\_pc\_name;

User\_name = temp.User\_name;

User\_message = temp.User\_message;

}

/// <summary>

/// Сериализация класса MsgJsonMSMQ в строку json формата

/// </summary>

/// <returns>строка json формата с полями класса MsgJsonMSMQ</returns>

public static string MsgJsonSerialize(MsgJsonMSMQ obj\_MsgJsonMSMQ)

{

return JsonSerializer.Serialize(obj\_MsgJsonMSMQ);

}

public static MsgJsonMSMQ MsgJsonDeserialize(string json\_string)

{

// https://metanit.com/sharp/tutorial/6.5.php

return JsonSerializer.Deserialize<MsgJsonMSMQ>(json\_string);

}

}

}