**Лабораторна робота №4**

**Тема:** Розробка сервіс-орієнтованих додатків у WCF.

**Мета роботи:** навчитися працювати з Windows Communication Foundation.

**Виділений час:** 8 годин (4 години лабораторних робіт та 4 години самостійної роботи).

**Завдання:**

За допомогою Windows Communication Foundation розробити мережевий чат, забезпечивши виконання наступних вимог:

- додаток має складатися з трьох проектів - бібліотеки класів,

клієнтського додатку, серверного додатку;

- при підключенні користувач вводить своє ім’я (нікнейм);

- у вікні чату відображається список користувачів, які знаходяться

онлайн;

- є можливість відправляти повідомлення у груповий чат або особисті

повідомлення певному користувачу.

**Лістинг Library:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.ServiceModel;

namespace ChatLibrary

{

[ServiceContract(CallbackContract = typeof(IChatServiceCallback))]

public interface IChatService

{

[OperationContract]

int Connect(string username);

[OperationContract(IsOneWay = true)]

void Disconnect(int id);

[OperationContract(IsOneWay = true)]

void SendMessage(string message, int id, int systBool, string name);

[OperationContract(IsOneWay = true)]

void UpdateUserList();

}

[ServiceContract]

public interface IChatServiceCallback

{

[OperationContract(IsOneWay = true)]

void SendMessageToClient(string message);

[OperationContract(IsOneWay = true)]

void UpdateUserListToClient(List<string> users);

}

public class ChatUser

{

public int Id { get; set; }

public string Name { get; set; }

public OperationContext Context { get; set; }

}

[ServiceBehavior(InstanceContextMode = InstanceContextMode.Single)]

public class ChatService : IChatService

{

List<ChatUser> usersList = new List<ChatUser>();

int nextUserId = 1;

public int Connect(string username)

{

ChatUser user = new ChatUser()

{

Id = nextUserId++,

Name = username,

Context = OperationContext.Current,

};

SendMessage($"---{user.Name}--- joined to the chat!", 0, 0);

usersList.Add(user);

return user.Id;

}

public void Disconnect(int id)

{

var user = usersList.FirstOrDefault(x => x.Id == id);

if (user != null)

{

usersList.Remove(user);

SendMessage($"---{user.Name}--- Disconnected", 0, 0);

}

}

public void SendMessage(string message, int id, int Sysbool, string nameSender = "")

{

string time = System.DateTime.Now.ToShortTimeString();

string answer;

if (id == 0)

{

foreach (ChatUser user in usersList)

{

if (Sysbool == 0)

{

answer = message;

user.Context.GetCallbackChannel<IChatServiceCallback>().SendMessageToClient(answer);

}

else

{

answer = $"{time}| {nameSender} => {message}";

user.Context.GetCallbackChannel<IChatServiceCallback>().SendMessageToClient(answer);

}

}

}

else

{

var userFrom = usersList.FirstOrDefault(x => x.Name == nameSender);

var userTo = usersList.FirstOrDefault(x => x.Id == id);

answer = $"{time}| Personal messages from {userFrom.Name} to {userTo.Name}:\n" +

$"{message}";

userTo.Context.GetCallbackChannel<IChatServiceCallback>().SendMessageToClient(answer);

userFrom.Context.GetCallbackChannel<IChatServiceCallback>().SendMessageToClient(answer);

}

}

public void UpdateUserList()

{

List<string> list = new List<string>();

foreach(var user in usersList)

{

list.Add(user.Name);

}

foreach (ChatUser user in usersList)

{

user.Context.GetCallbackChannel<IChatServiceCallback>().UpdateUserListToClient(list);

}

}

}

}

**Лістинг ServerConnector.cs:**

using Client.ChatServiceReference;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Client

{

public class ChatServerConnector : IChatServiceCallback

{

private static ChatServerConnector Instance = null;

private ChatServiceClient Client;

private int UserId;

private string Name;

private ChatServerConnector()

{

Client = new ChatServiceClient(new System.ServiceModel.InstanceContext(this));

}

static ChatServerConnector()

{

if (Instance == null)

Instance = new ChatServerConnector();

}

public static ChatServerConnector GetInstance()

{

if (Instance == null)

Instance = new ChatServerConnector();

return Instance;

}

public static void Disconnect()

{

GetInstance().Client.Disconnect(Instance.UserId);

}

internal static string Connect(string text)

{

if (text.Length < 4)

{

return ErrorMessageUserName();

}

Instance.UserId = Instance.Client.Connect(text);

Instance.Name = text;

GetInstance().Client.UpdateUserList();

return "OK";

}

public void SendMessageToServer(string message, int id)

{

GetInstance().Client.SendMessage(message, id, 1, Instance.Name);

}

public int SelectedUser()

{

return (Application.OpenForms["Form1"] as Form1).listBoxUsers.SelectedIndex;

}

public void SendMessageToClient(string message)

{

(Application.OpenForms["Form1"] as Form1).listBoxChat.Items.Add(message);

}

public void UpdateUserListToClient(string[] users)

{

(Application.OpenForms["Form1"] as Form1).listBoxUsers.Items.Clear();

(Application.OpenForms["Form1"] as Form1).listBoxUsers.Items.Add("GroupChat---");

(Application.OpenForms["Form1"] as Form1).listBoxUsers.SelectedIndex = 0;

foreach (var user in users)

{

(Application.OpenForms["Form1"] as Form1).listBoxUsers.Items.Add(user);

}

}

private static string ErrorMessageUserName()

{

return "UserName can't be less than 4";

}

}

}

**Лістинг Form:**

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void Form1\_Load(object sender, EventArgs e)

{

listBoxChat.Items.Clear();

}

private void button1\_Click(object sender, EventArgs e)

{

string msg = ChatServerConnector.Connect(textBox1.Text);

if (msg != "OK")

{

MessageBox.Show(msg);

return;

}

textBox1.ReadOnly = true;

button3.Visible = true;

}

private void button2\_Click(object sender, EventArgs e)

{

SendMessage();

}

private void SendMessage()

{

ChatServerConnector.GetInstance().SendMessageToServer(textBox2.Text, ChatServerConnector.GetInstance().SelectedUser());

}

private void button3\_Click(object sender, EventArgs e)

{

ChatServerConnector.Disconnect();

Application.Exit();

}

private void textBox2\_KeyDown(object sender, KeyEventArgs e)

{

if (e.KeyCode == Keys.Enter)

{

SendMessage();

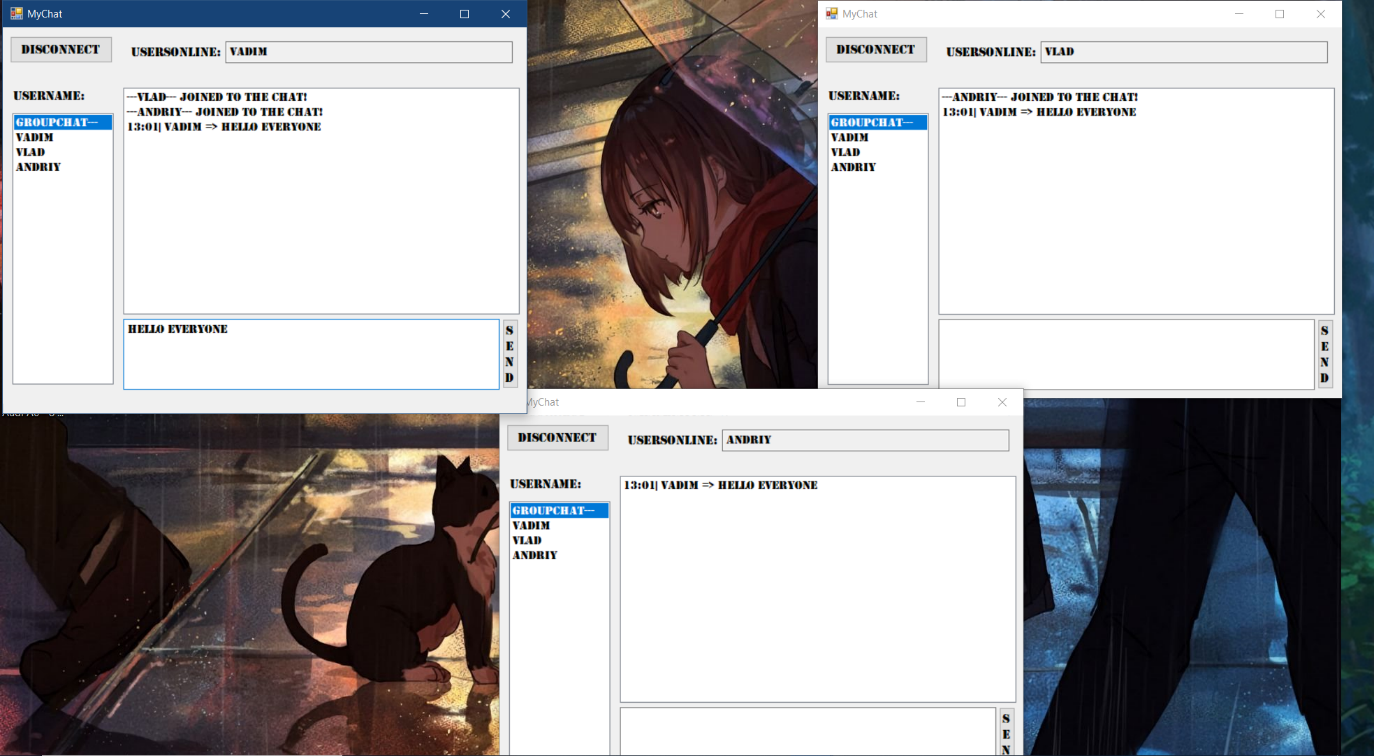
}

}

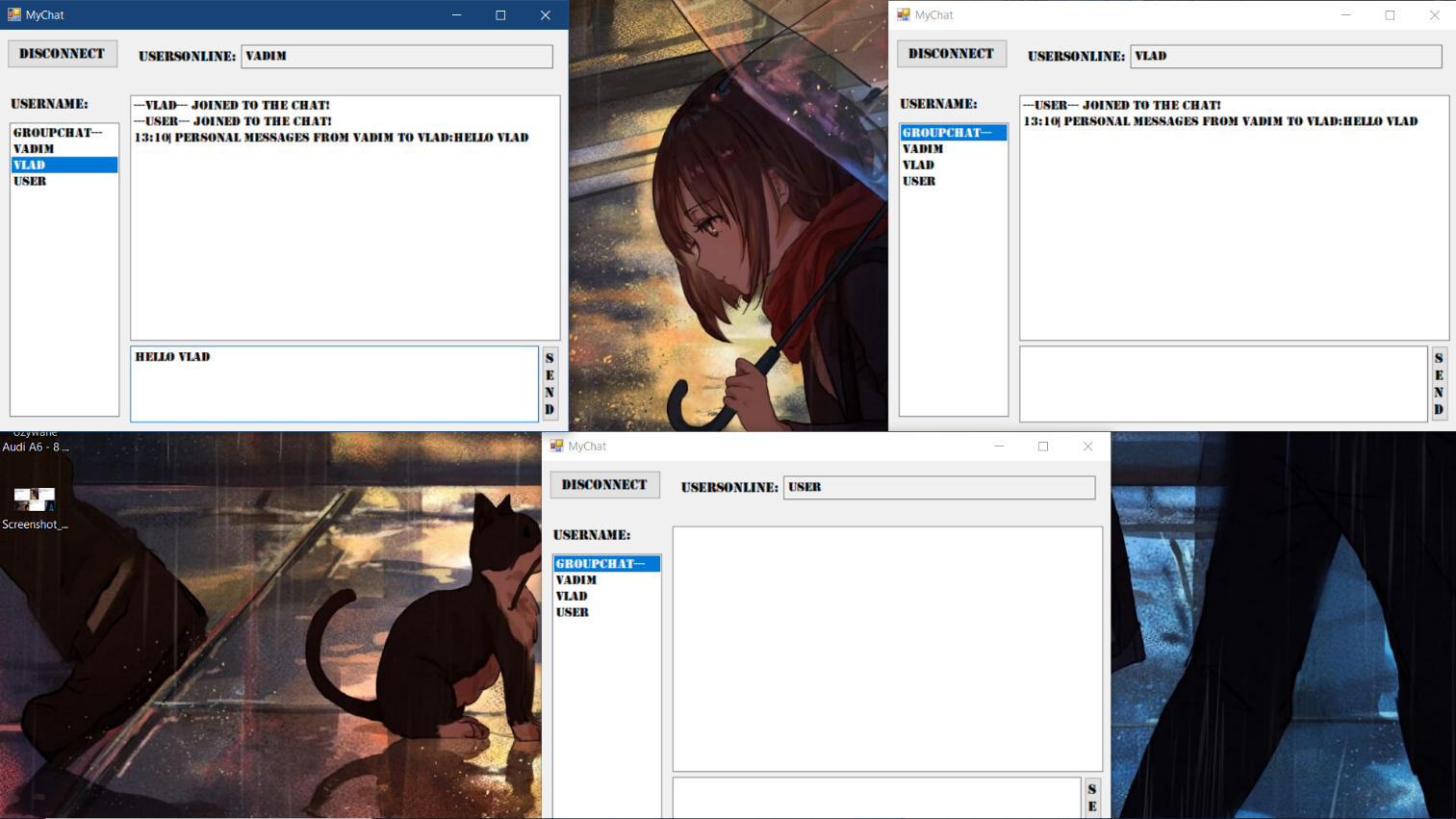
}

**Результат:**

**--Надсилання повідомлень усім юзерам:**

****

**--Надсилання повідомлень 1 юзеру:**

****

**Висновок:** Я навчився працювати з Windows Communication Foundation

**Посилання на git:**

**https://gitlab.com/ipz212\_bvm/dotnetlab3**