

Chat Room

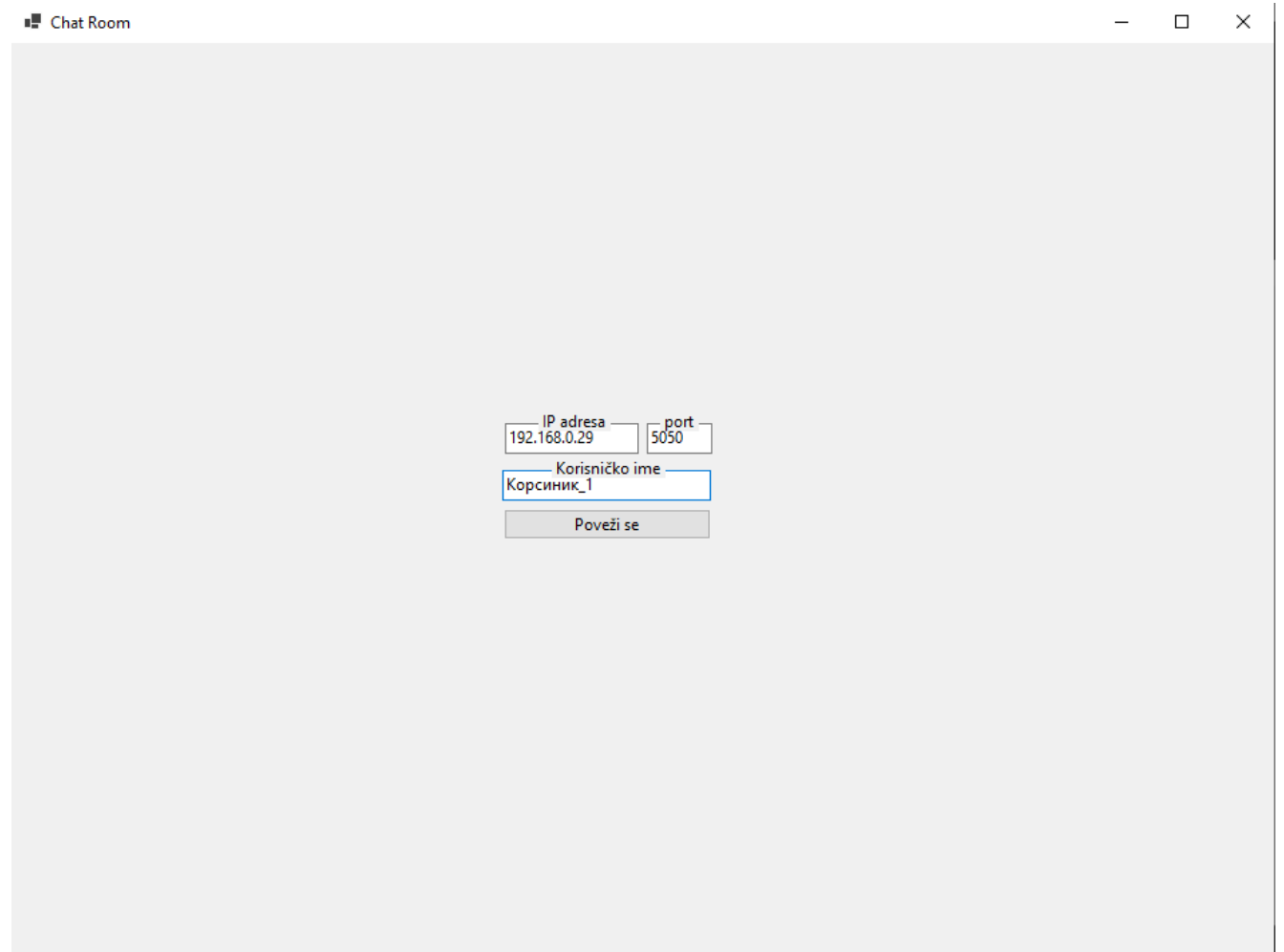
Вук Вукашиновић

УВОД

- Као тему овог рада направио сам апликацију која омогућава комуникацију између корисника. Реализовао сам је унутар С# окружења .NET, без употребе других библиотека.

Опис апликације

- Када покренемо апликацију, од нас се тражи да унесемо IP адресу и PORT сервера. Након тога чекамо да се повежемо са сервером.



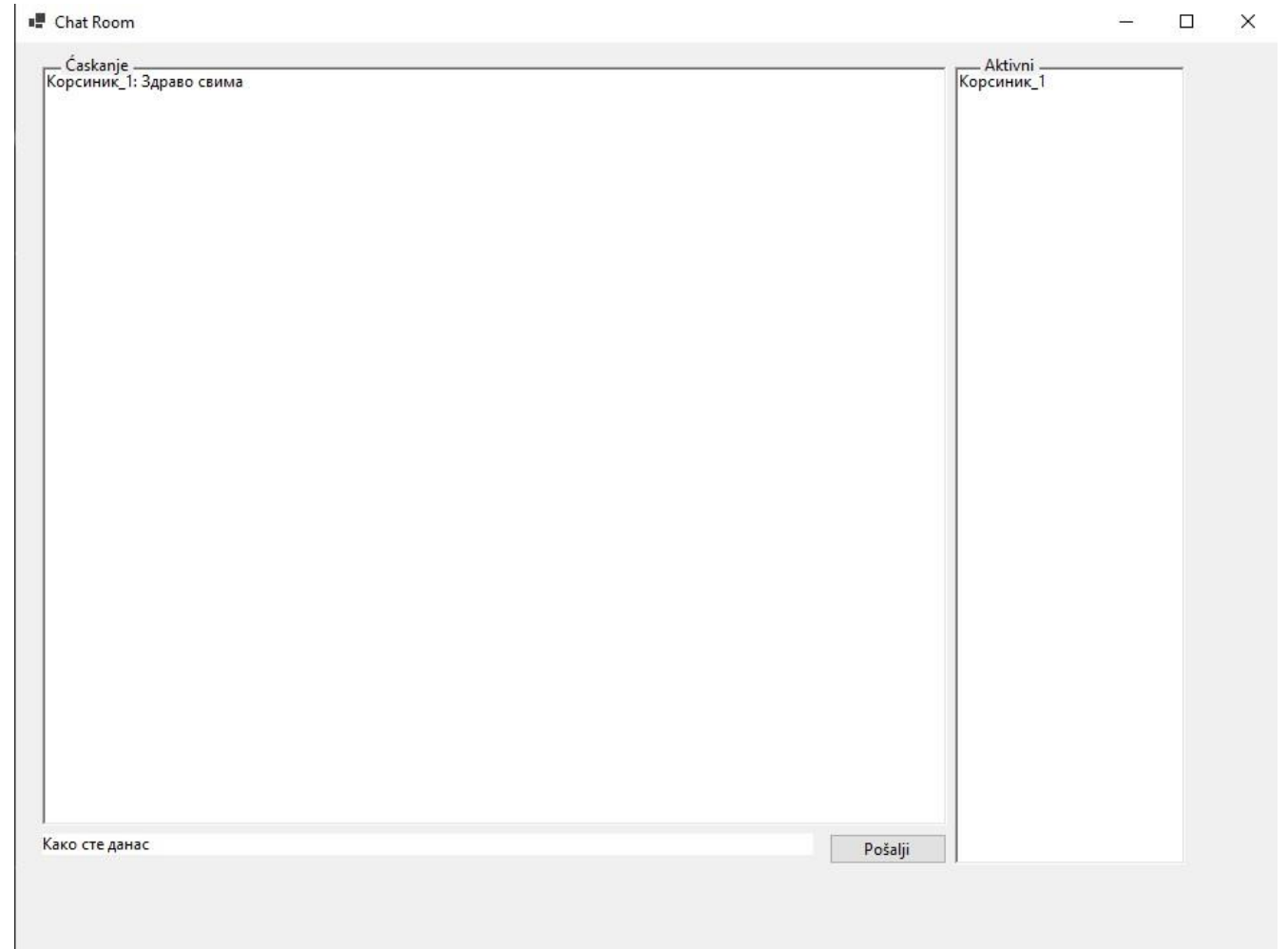
The screenshot shows a window titled "Chat Room" with a light gray background. In the bottom right corner, there is a form for connecting to a server. It consists of three input fields and a button. The first field is labeled "IP adresa" and contains the text "192.168.0.29". The second field is labeled "port" and contains the text "5050". The third field is labeled "Korisničko ime" and contains the text "Корсиник_1". Below these fields is a button labeled "Повежи се".

Field Label	Value
IP adresa	192.168.0.29
port	5050
Korisničko ime	Корсиник_1

Повежи се

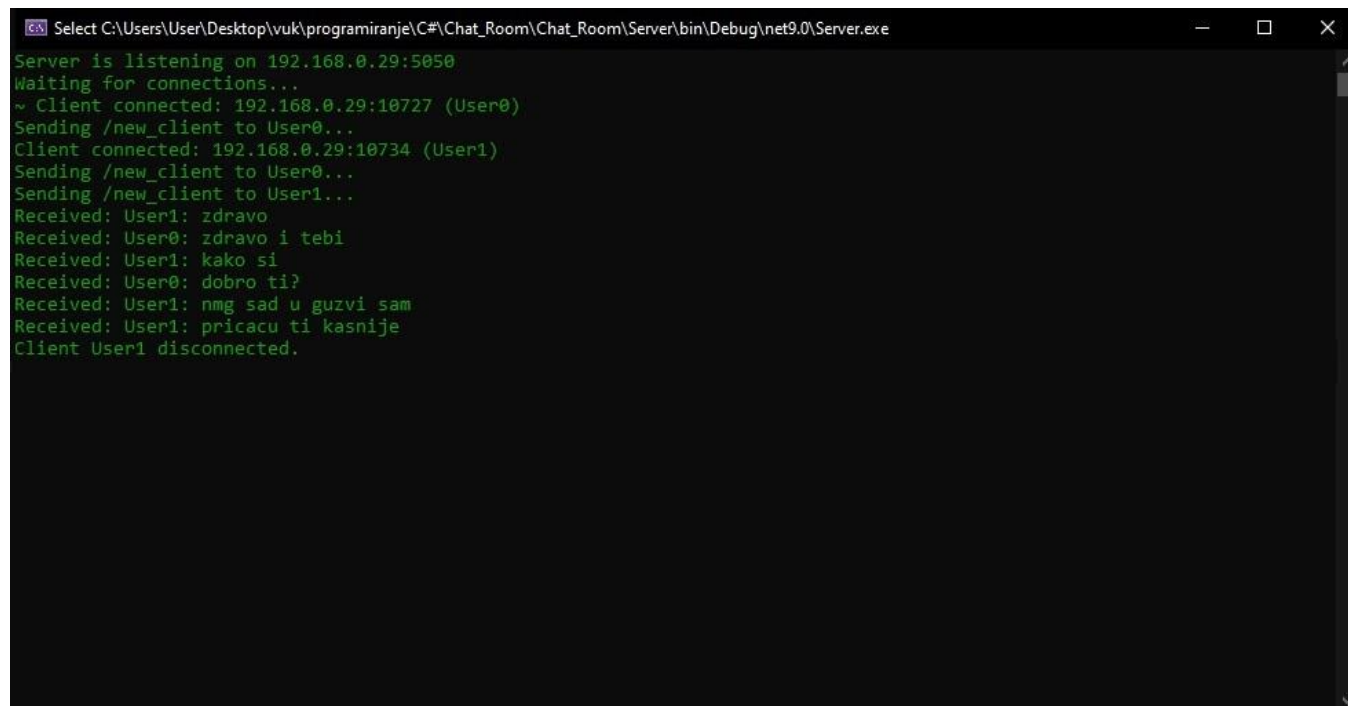
Опис апликације

- Ово је део апликације у коме се врши дописивање међу тренутно активним клијентима.



Како све то ради?

- Постоје две класе “Сервер” и “Клијент”. Свака одговорна за свој део.
- Сервер је задужен за обраду нових клијената и њихових порука.
- Клијен је одговоран за успостављање везе са сервером и обраду корсничког уноса.



```
Select C:\Users\User\Desktop\vuk\programiranje\C#\Chat_Room\Chat_Room\Server\bin\Debug\net9.0\Server.exe

Server is listening on 192.168.0.29:5050
Waiting for connections...
~ Client connected: 192.168.0.29:10727 (User0)
Sending /new_client to User0...
Client connected: 192.168.0.29:10734 (User1)
Sending /new_client to User0...
Sending /new_client to User1...
Received: User1: zdravo
Received: User0: zdravo i tebi
Received: User1: kako si
Received: User0: dobro ti?
Received: User1: nmg sad u guzvi sam
Received: User1: pricacu ti kasnije
Client User1 disconnected.
```

Како све то ради?

Сервер

- Када се програм покрене сервер исписује локалну IP адресу и PORT.
- Покреће бесконачну петљу која чека нове клијенте
- Када се нови клијент повеже позива се функција *HandleClient* унутар нове нити
- Прво се чека да клијент пошаље своје име и потом се то исто име шаље назад као потврда
- Након се обавештавају сви корисници о новом клијенту и новом клијенту се шаље списак активних клијената
- Унутар те функције се налази бесконачна петља која чека нове поруке од корисника и прослеђује их даље

```
15 // Класа која представља корисникове податке
16 private class Client
17 {
18     private static uint curect_id = 0; // Глобални тренутни ID који ће
19     public TcpClient TcpClient;        бити постављен кориснику, такође
20     public NetworkStream stream;        у случају да нема username биће
21     public string username;            му исти као ID. Пошто је static биће
22     public uint ID;                    исти за све кориснике
23
24     Invoke Windsurf | 1 reference
25     public Client(TcpClient tcpClient)
26     {
27         TcpClient = tcpClient;
28         stream = TcpClient.GetStream();
29         ID = curect_id++;
30         username = "Guest" + ID;
31     }
32
33     Invoke Windsurf | 11 references
34     private static void HandleClient(Client client)
35     {
36         // Obaveštavanje novog klijenta da se povezao i slanje liste postojećih korisnika.
37         Send_to_All(client, $"new_client {client.username}", false); // Novi klijent se obaveštava da se povezao.
38
39         /*
40         foreach (Client c in ActiveClient)
41         {
42             Console.WriteLine($"Sending /new_client to {c.username}...");
43             Send_to(client, $"new_client {c.username}", default_encoder);
44             Thread.Sleep(25);
45         }
46
47         /*
48         Send_to_All(client, $"[SERVER]: {client.username} joined the chat.");
49
50         /*
51         while ((bytesRead = stream.Read(buffer, 0, buffer.Length)) > 0)
52         {
53             if (bytesRead == 0) break;
54
55             string receivedMessage = Recv_from(buffer, bytesRead, default_encoder);
56
57             if (HandleCommand(client, receivedMessage) == "OK")
58             {
59                 continue;
60             }
61
62             Console.WriteLine($"Received: {client.username}: {receivedMessage.Trim()}");
63             Send_to_All(client, receivedMessage);
64         }
65
66         /*
67         Invoke Windsurf | 0 references
68         static void Main(string[] args)
69         {
70             IPAddress ipAddress = IPAddress.Any;
71             int ipPort = 5050;
72             TcpListener server = new TcpListener(ipAddress, ipPort);
73             server.Start();
74
75             // Dobijanje lokalne IP adrese klijenta, u slucaju da je rezultat null postavljamo Unknown IP
76             string localIP = Dns.GetHostEntry(Dns.GetHostName())
77                 .AddressList
78                 .FirstOrDefault(ip => ip.AddressFamily == AddressFamily.InterNetwork)
79                 ?.ToString() // ? poziva samo ako vrednost nije null
80                 ?? "Unknown IP"; // ?? postavlja vrednost samo ako je null
81
82             Console.WriteLine($"Server is listening on {localIP}:{ipPort}");
83             Console.WriteLine("Waiting for connections...");
84
85             Task.Run(() => HandleAdminInput(server));
86
87             while (serverRunning)
88             {
89                 try
90                 {
91                     Client client = new Client(server.AcceptTcpClient());
92                     lock (clientListLock)
93                     {
94                         ActiveClient.Add(client);
95                     }
96                     Task.Run(() => HandleClient(client));
97                 }
98                 catch (SocketException)
99                 {
100                     Console.WriteLine("Server has stopped accepting new connections.");
101                     break;
102                 }
103             }
104
105             Console.WriteLine("Server shut down.");
106         }
107     }
108 }
```

Како све то ради?

Клијент

- Када се класа клијента иницијализује, покушава да се повеже са сервером.
- Ако успе онда се врти у бесконачној петљи и чека нове поруке.
- Уколико порука намењена као команда онда се не приказује кориснику, у супротном приказује се

```
Invoke Windsurf | 2 references
public Client_Network(IPAddress ip, int port, string username)
{
    try
    {
        TcpClient client = new TcpClient();
        client.Connect(new IPEndPoint(ip, port));

        server_stream = client.GetStream();
        isActive = true;

        // Pošalji username serveru
        Send_to($"{username} set {username}");

        // Pročitaj potvrdu od servera
        byte[] buffer = new byte[1024];
        int bytesRead = server_stream.Read(buffer, 0, buffer.Length);
        this.username = username;

        Console.WriteLine($"Connected as {this.username}");

        // Pokreni procese za slanje i primanje poruka
        Task.Run(() => HandleReceiveMessages());
    }
    catch (Exception ex)
    {
        Console.WriteLine($"Connection failed: {ex.Message}");
        isActive = false;
    }
}

// Eventi
public delegate void Primljena_Poruka(string message);
public delegate void Novi_Korisnik(string message);
public delegate void Uklonjen_Korisnik(string message);

public event Primljena_Poruka OnNewMessage = delegate { };
public event Novi_Korisnik OnNewUser = delegate { };
public event Uklonjen_Korisnik OnExitUser = delegate { };

Invoke Windsurf | 1 reference
private void HandleReceiveMessages()
{
    byte[] buffer = new byte[1024];

    try
    {
        while (isActive)
        {
            int bytesRead = server_stream.Read(buffer, 0, buffer.Length);
            if (bytesRead == 0) break;

            string message = Encoding.UTF8.GetString(buffer, 0, bytesRead).Trim();

            if (message.StartsWith("/new_client "))
            {
                // Obradujemo samo poruke o novom klijentu
                OnNewUser?.Invoke(message.Substring(12));
            }
            else if (message.StartsWith("/exit_client "))
            {
                // Obrada izlaska klijenta
                OnExitUser?.Invoke(message.Substring(13));
            }
            else
            {
                // Obradujemo ostale poruke
                OnNewMessage?.Invoke(message);
            }
        }
    }
    catch
    {
        Console.WriteLine("Disconnected from server.");
    }

    isActive = false;
}
```

/*
Логика која решава пријем порука
са сервера. У линијама 17 - 23 се
декларису Event који омогућавају
једноставније повезивање са UI

Функција HandleReceiveMessages
је покренута у сопственој нити тако
да има while петљу унутар које се
чекају нове поруке са сервера
*/

Како све то ради?

User interface

- UI класа није неопходна за рад користи се само да би апликација била једноставнија за употребу. И тако се и користи, главна логика је у Клијент класи.
- Како бих имао више менија користио сам два панела један преко другог. На почетку се сцене додају у *Dict* (речник) који чува сцене. Мењају се тако што се све сем изабране сакрију.
- Када се кликне дугме *Повежи Се* читају се унети подаци и позива се конструктор *Клијент* класе

```
Invoke Windsurf | 1 reference
public Form1()
{
    InitializeComponent();

    scene.Add("MainMenu", MainMenu); // Додаје панеле у речник (dict)
    scene.Add("Chat", Caskanje);

    // На основу датаог имена сцене мења која се види а која не
}
Invoke Windsurf | 1 reference
private void Promeni_Scenu(string scena)
{
    foreach (var entry in scene)
    {
        entry.Value.Visible = (entry.Key == scena);
    }
}
Invoke Windsurf | 1 reference
private void Povezi_Se_Click(object sender, EventArgs e)
{
    try // Прикупља унос поља (IP, PORT, username)
    {
        ip_adress = IPAddress.Parse(IP_textbox.Text);

        string port_txt;

        if (Port_textbox.Text == "")
        {
            port_txt = Port_textbox.PlaceholderText;
        }
        else
        {
            port_txt = Port_textbox.Text;
        }

        port = int.Parse(port_txt);
        username = Username_textbox.Text;

        if (username == "")
        {
            username = Username_textbox.PlaceholderText;
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
        return;
    }

    try // Прави нову Клијент класу и предплаћује се на event
    {
        client = new Client_Network(ip_adress, port, username);

        client.OnNewMessage += Handle_New_Message;
        client.OnNewUser += Handle_New_Client;
        client.OnExitUser += Handle_Exit_Client;
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
        return;
    }

    Promeni_Scenu("Chat");
}
}
```

Document Outline - Form1

- Form1 Form
 - MainMenu Panel
 - Povezi_Se Button
 - label5 Label
 - label4 Label
 - label3 Label
 - Username_textbox TextBox
 - IP_textbox TextBox
 - Port_textbox TextBox
 - Caskanje Panel
 - label2 Label
 - label1 Label
 - Posalji Button
 - Chat_History RichTextBox
 - Active_Users_textbox RichTextBox
 - Poruka_textbox TextBox

Закључак

- Овај пројекат ми је пружио прилику да научим доста о *socket* програмирању и TCP протоколу, такође и изради графичко коричког интерфејса.
- Постоје разна поља и смерови која би могла да се усаврше:
 1. Додавање крипције података који се шаљу
 2. Да постоји могућност да постоје више соба на једном серверу. Или један главни сервер који би преусмеравао на друге сервере
 3. Могућност да се “*Enter*” дугметом пошаље порука
- Волео бих да напоменем да слике кодова су модификоване тако да прикажу само интересантне делове. Уколико хоћете да погледате цео код, детаљнији опис и да пробате као ради можете то учинити на GitHub страници
https://github.com/Vukile2801w/Chat_Room