

# СОБА ЗА ЋАСКАЊЕ CHAT ROOM

Вук Вукашиновић

*VIII, ОШ 'Илија Вирчанин', Регионални центар за таленте Београд 1 – Земун*

## **РЕЗИМЕ**

Овај рад представља апликацију 'Чат соба' - комуникациону апликацију развијену у чистом C# која омогућава реалновременску размену порука између више корисника. Апликација је дизајнирана тако да омогући једноставно повезивање клијент-сервер мрежног окружења, и са својим модуларним приступом лако се надограђује. Рад објашњава функције апликације, начин рада, кориснички интерфејс, као и структуру и значајне линије кода.

## **КЉУЧНЕ РЕЧИ:**

чат соба, комуникација, C#, клијент-сервер архитектура, апликација

## **ABSTRACT:**

This paper introduces the “Chat Room” application – a communication tool developed in pure C# that enables real-time message exchange among multiple users. The application is designed to facilitate simple connection between client and server modules and can easily be upgraded with its modular approach. The paper explains the functionalities, user interface, core code structure, and key components of the application.

## **KEYWORDS:**

chat room, communication, C#, client-server architecture, application

## Увод

Апликација која је предмет овог рада омогућава комуникацију више корисника у реалном времену преко интернета. Апликација је развијена унутар C# окружења уз помоћ .NET библиотека. Ово омогућава брзу и стабилну конекцију. Апликација је приближена корисницима израдом корисничког интерфејса унутар *Windows Forms*. Овај рад је примена наученог знања са часова програмирања и самосталног истраживања.

## Настанак идеје

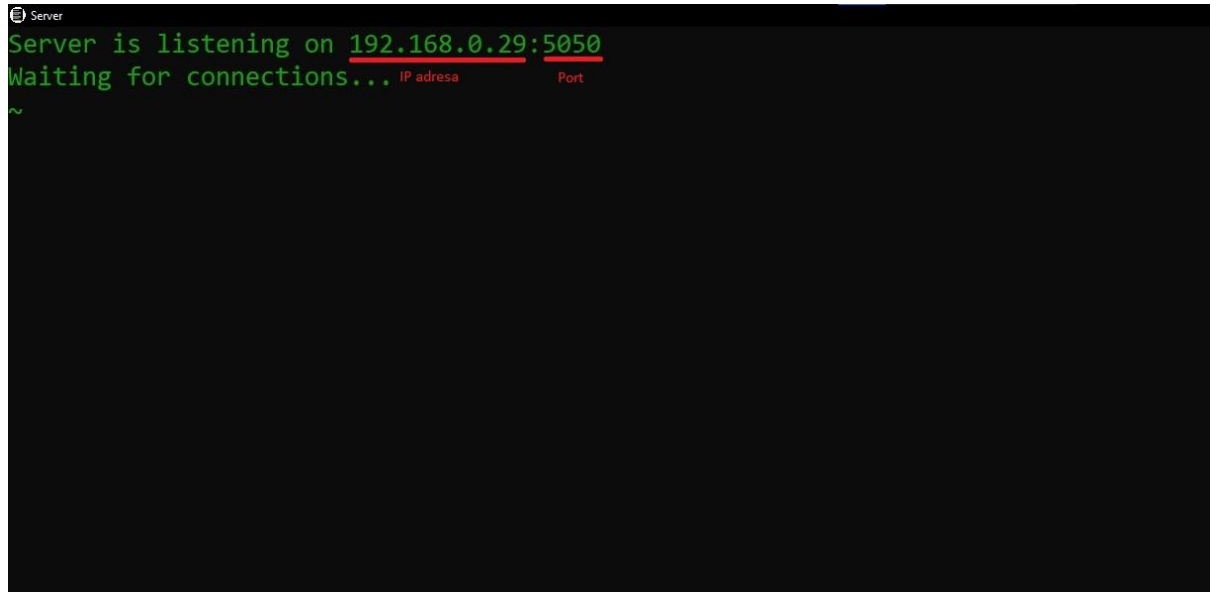
Један дан дописивао сам се са другом, запитао сам се како све то функционише. Како се врши размена порука, фајлова, видео и аудио разговора. Кренуо сам да се распитујем по интернету и да откривам како све то функционише. У мени се развила жеља да покушам да то све рекреирам.

Почео сам тиме што сам направио 'Сервер' и 'Клијент' код који су могли у минималном смислу да комуницирају један између другог. То је све било лимитирано на конзолу што као што можете да замислите није најприлагођеније корисницима. Тако да је на све то морала да се угради UI (*коринички интерфејс*). То је знатно олакшало интеракцију са апликацијом

## Рад са апликацијом

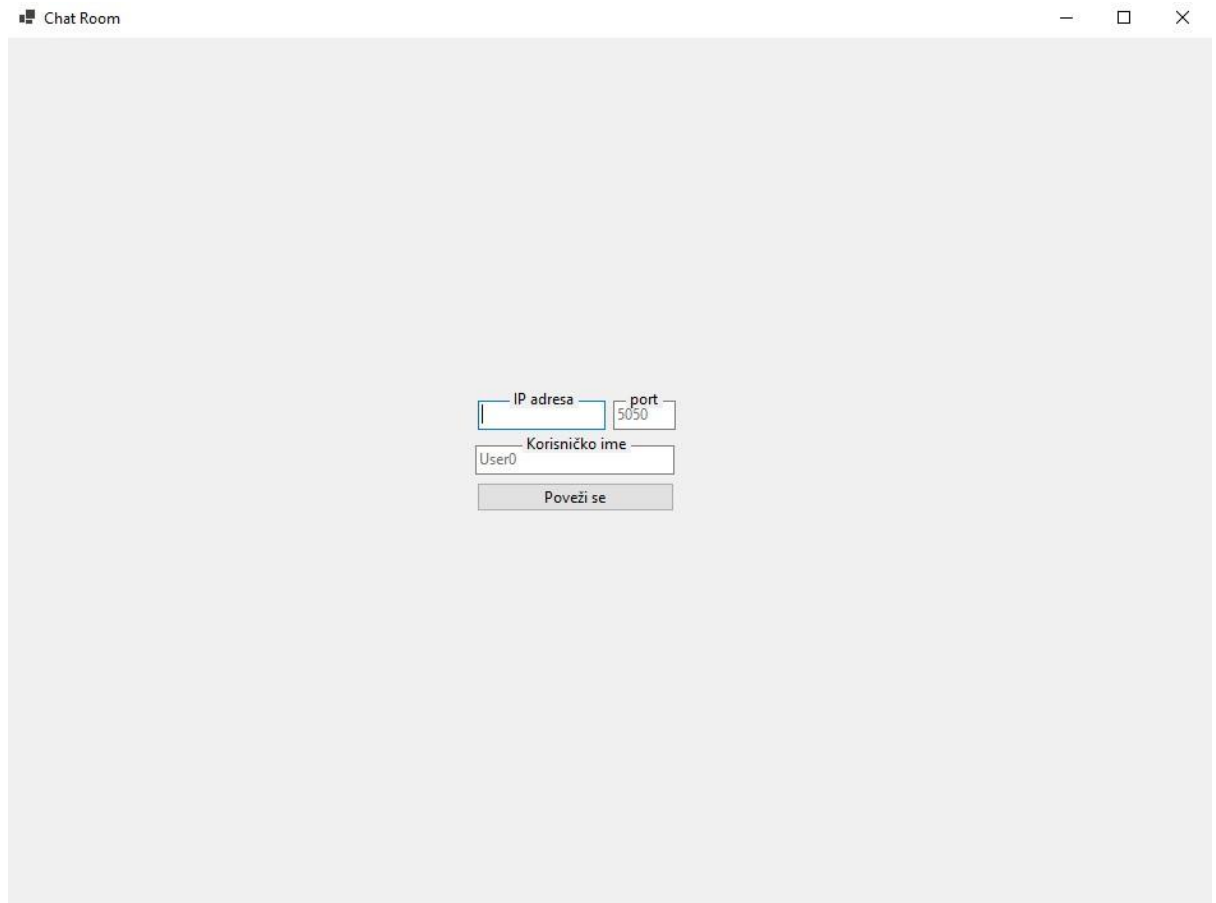
Ово поглавље представља упутство за кориснике

1. За почетак морамо да покренемо апликацију сервера преко које ће сви разговори да пролазе. Када се покрене имаћемо прозор (слика 1.1), исписаће локалну IP адресу и PORT на којој сервер ради, ово ће нам бити потребно да бисмо се повезали. Напомена уколико се у подешавањима рутера не подеси *port forwarding* конекције ће бити могуће унутар једне интернет везе, уколико је *port forwarding* подешен уместо исписане IP адресе мораће да се користи јавна адреса. Један од начина да се она сазна је да посетите [What is my ip address](#).



Слика 1.1 Конзола сервера

2. Након тога покрећемо Chat Room апликацију и у њој попуњавамо поља са слике (слика 2.1), IP адреса I PORT су адреса и порт нашег сервера, а у претходном кораку смо видели како да их сазнамо. Корисничко име само говори, уносите име под којим ће вас други људи у ћаскању видети, и кликнемо дугме “Poveži se”.

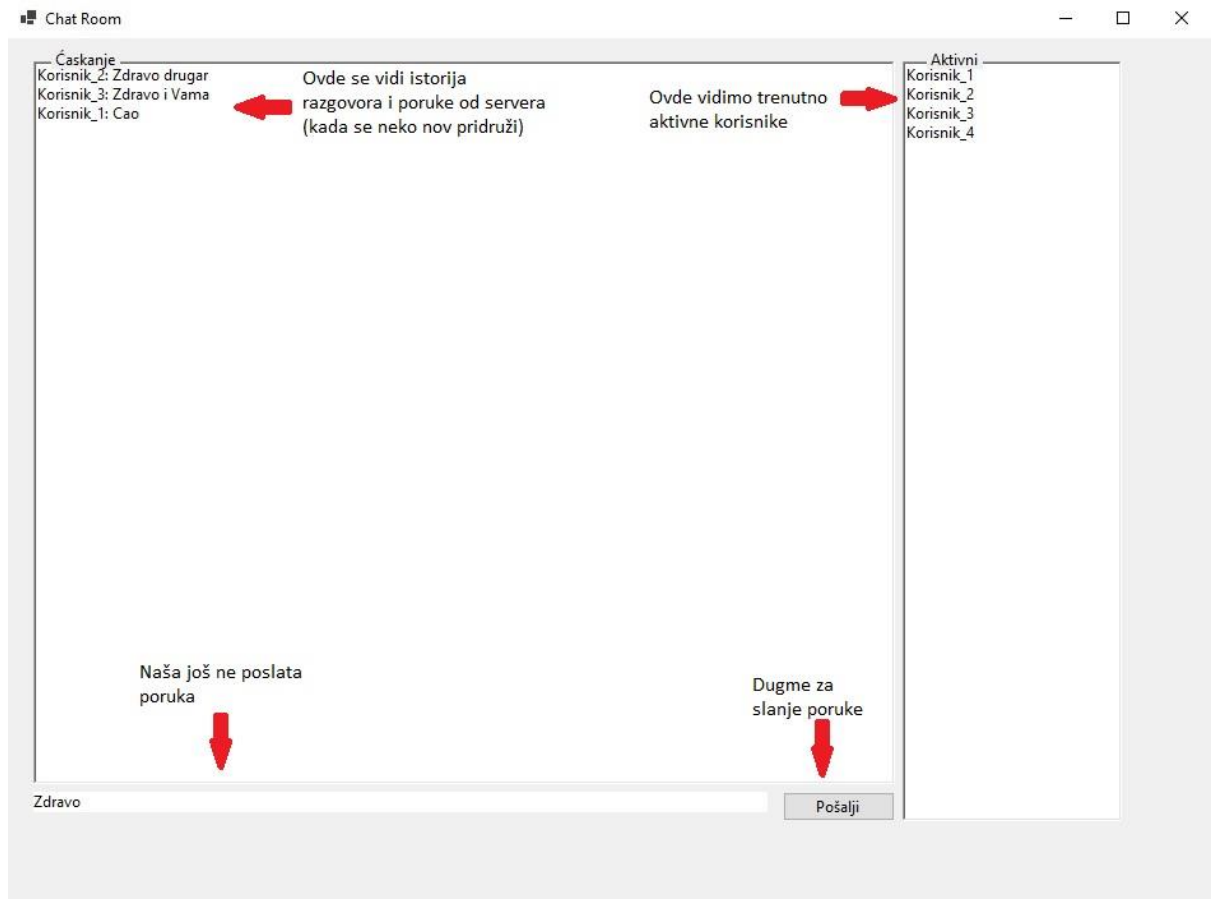


The screenshot shows a window titled "Chat Room" with a light gray background. In the center, there is a login form with the following elements:

- An "IP адреса" label above a text input field.
- A "port" label above a text input field containing the value "5050".
- A "Korisničko ime" label above a text input field containing the value "User0".
- A "Poveži se" button below the input fields.

Слика 2.1 Изглед почетног менија

3. Након повезивања имамо следећи мени (слика 2.2) и уједно и главни мени у коме се врши дописивање. У њему можемо видети разговор, тренутно активне кориснике, и шаљемо поруке



Слика 2.2 Изглед главног менија

## Опис Апликације

Апликација је развијена у чистом С# уз коришћењ .NET оквира. Примарни фокус је био на дизајнирању сервер-клијент конекције који је накнадно надограђен да користи UI. Апликацију би поделио у два главна дела, сервер и клијент класе. Конекција је базирана на *socket* и TCP протоколом. Постоје разне библиотеке које су ово имплементирале тако да је лакше да се користи и на бољи начин али ја сам изабрао да радим овако јер се на овај начин много више може научити рад са *socket-има*.

- Клијент: Ова класа има за задатак да успостави и одржава конекцију са сервером

```
16 // Event1
17 public delegate void Prijava_Poruka(string message);
18 public delegate void Novi_Korisnik(string message);
19 public delegate void Uklonjen_Korisnik(string message);
20
21 public event Prijava_Poruka OnNewMessage = delegate { };
22 public event Novi_Korisnik OnNewUser = delegate { };
23 public event Uklonjen_Korisnik OnExitUser = delegate { };
24
25 // Invoke Window1 reference
26 private void HandleReceiveMessages()
27 {
28     byte[] buffer = new byte[1024];
29
30     try
31     {
32         while (isActive)
33         {
34             int bytesRead = server_stream.Read(buffer, 0, buffer.Length);
35             if (bytesRead == 0) break;
36
37             string message = Encoding.UTF8.GetString(buffer, 0, bytesRead).Trim();
38
39             if (message.StartsWith("/new_client "))
40             {
41                 // Obradujemo samo poruke o novom klijentu
42                 OnNewUser?.Invoke(message.Substring(12));
43             }
44             else if (message.StartsWith("/exit_client "))
45             {
46                 // Obrada izlaska klijenta
47                 OnExitUser?.Invoke(message.Substring(13));
48             }
49             else
50             {
51                 // Obradujemo ostale poruke
52                 OnNewMessage?.Invoke(message);
53             }
54         }
55     }
56     catch
57     {
58         Console.WriteLine("Disconnected from server.");
59     }
60
61     isActive = false;
62 }
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

Слика 3.1 Структура Клијента

- Сервер: Задужен за обраду клијената прослеђивање порука.  
на слици 4.1 можемо видети класу која је представља клијенте на серверу (15-31),  
и функцију која обрађује везу са корисницима (107-171). На слици 4.2 можемо видети  
функцију која обрађује кориснички унос у конзоли, и видимо *Main* функцију која има

задатак да покрене сервер.

```
15 private class Client // Класа која представља корисникове податке
16 {
17     private static uint currect_id = 0; // Глобални тренутни ID који ће
18     // бити постављен кориснику, такође
19     // у случају да нема username биће
20     // му исти као ID. Пошто је static биће
21     // исти за све кориснике
22     public TcpClient tcpClient;
23     public NetworkStream stream;
24     public string username;
25     public uint ID;
26
27     Invoke Windsor! 1 reference
28     public Client(TcpClient tcpClient)
29     {
30         tcpClient = tcpClient;
31         stream = tcpClient.GetStream();
32         ID = currect_id++;
33         username = "Guest" + ID;
34     }
35
36     private static void HandleClient(Client client)
37     {
38         // Obaveštavanje novog klijenta da se povezao i slanje liste postojećih korisnika.
39         Send_to_All(client, $"/new_client {client.username}", false); // Novi klijent se obaveštava da se povezao.
40
41         /*
42         foreach (Client c in ActiveClient)
43         {
44             Console.WriteLine($"Sending /new_client to {c.username}...");
45             Send_to(client, $"/new_client {c.username}", default_encoder);
46             Thread.Sleep(25);
47         }
48         */
49
50         Send_to_All(client, $"[SERVER]: {client.username} joined the chat.");
51
52         /*
53         while ((bytesRead = stream.Read(buffer, 0, buffer.Length)) > 0)
54         {
55             if (bytesRead == 0) break;
56             string receivedMessage = Recv_from(buffer, bytesRead, default_encoder);
57
58             if (HandleCommand(client, receivedMessage) == "OK")
59             {
60                 continue;
61             }
62
63             /*
64             Console.WriteLine($"Received: {client.username}: {receivedMessage.Trim()}");
65             Send_to_All(client, receivedMessage);
66             */
67         }
68     }
69 }
```

Слика 4.1 Структура класе која чува податке клијената и обрада клијената

```

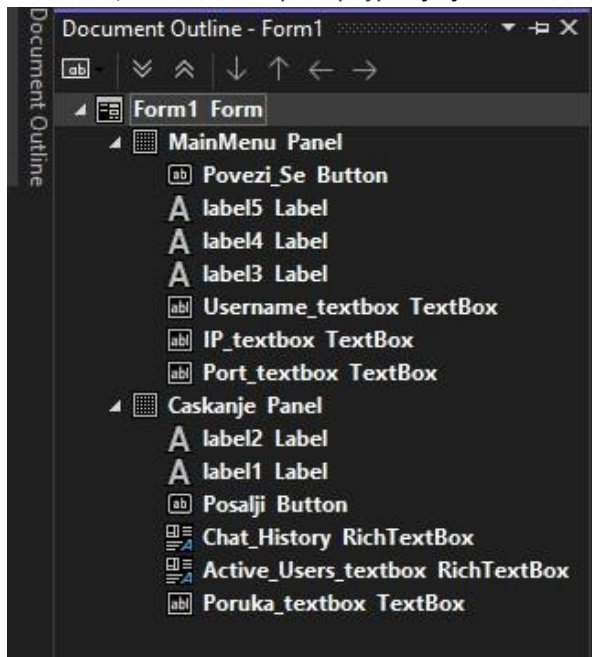
184 private static void HandleAdminInput(TcpListener server)
185 {
186     while (serverRunning)
187     {
188         Console.WriteLine(" ");
189         string adminInput = Console.ReadLine().ToLower(); // Чита унос из конзоле и ако се
190                                                         // погађа са речима за гашење
191                                                         // (exit, quit) сервер се гаси
192         if (EXIT_WORDS.Contains(adminInput))
193         {
194             Console.WriteLine("Shutting down server...");
195             serverRunning = false;
196             lock (clientListLock) // Потом иде кроз листу
197             { // активних клијената и
198                 foreach (Client client in ActiveClient) // disconnect-ује их
199                 {
200                     Send_to(client, "[SERVER] Server is shutting down. Disconnected.", default_encoder);
201                     client.TcpClient.Close();
202                 }
203                 ActiveClient.Clear();
204             }
205             server.Stop(); // И на крају гаси socket и
206             break;         // престаје да очекује нове
207                             // клијенте
208         }
209     }
210 }
211
212 static void Main(string[] args)
213 {
214     IPAddress ipAddress = IPAddress.Any;
215     int ipPort = 5050;
216     TcpListener server = new TcpListener(ipAddress, ipPort);
217     server.Start();
218
219     // Dobijanje lokalne IP adrese klijenta, u slucaju da je rezultat null postavljamo Unknown IP
220     string localIP = Dns.GetHostEntry(Dns.GetHostName())
221         .AddressList
222         .FirstOrDefault(ip => ip.AddressFamily == AddressFamily.InterNetwork)
223         ?.ToString() // ? poziva samo ako vrednost nije null
224         ?? "Unknown IP"; // ?? postavlja vrednost samo ako je null
225
226     Console.WriteLine($"Server is listening on {localIP}:{ipPort}");
227     Console.WriteLine("Waiting for connections...");
228
229     Task.Run(() => HandleAdminInput(server));
230
231     while (serverRunning)
232     {
233         try
234         {
235             Client client = new Client(server.AcceptTcpClient()); /*
236             lock (clientListLock)                                   Прихвата нове клијенте,
237             {                                                       додаје их у листу активних и
238                 ActiveClient.Add(client);                           покреће нову нит у којој се
239             }                                                       клијент обрађује
240             Task.Run(() => HandleClient(client));                    */
241         }
242         catch (SocketException)
243         {
244             Console.WriteLine("Server has stopped accepting new connections.");
245             break;
246         }
247     }
248     Console.WriteLine("Server shut down.");
249 }
250
251 }

```

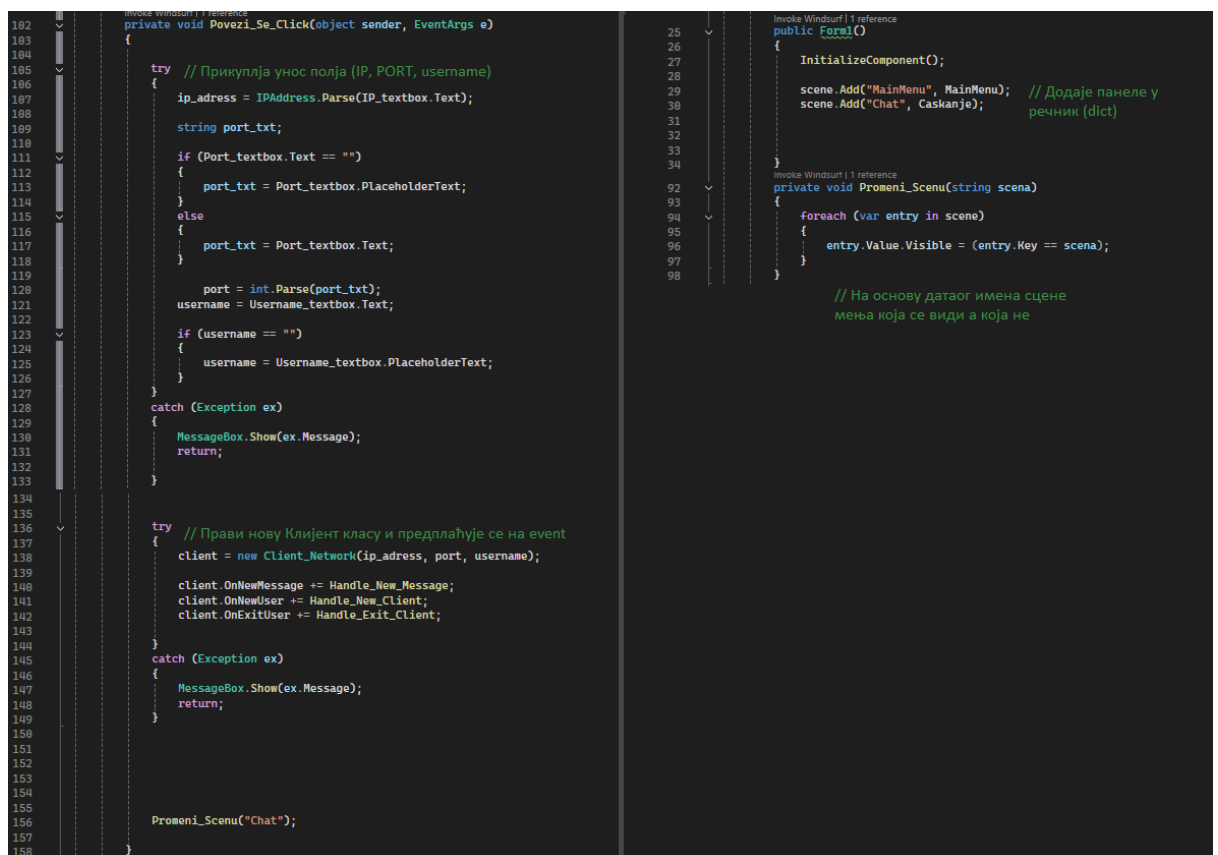
Слика 4.2 Обрада корисничког уноса и функција за покретање сервера



- UI: То је само спајање Windows Forms и класе Client\_Network (слике 5.2). Како бих лако могао да мењам 'сцене' додао сам два панела, један је, видљив други није када се мења 'сцена' само се ротирају који је панел видљив, а који није (слика 5.1).



Слика 5.1 Приказ WF дизајна



## ЗАКЉУЧАК

Апликација 'Чат соба' је развијена са циљем истраживања методологије за комуникацију у реалном времену преко мреже. Пројекат је пружио практично искуство у раду са мрежним протоколима, сокет програмирању, и изради графичко корисничког интерфејса.

Апликација је врло забавна и функционална, нуди разна поља за усавршавање, попут оптимизација и безбедности.

Уколико сте заинтересовани за цео код, и да пробате апликацију можете то урадити на GitHub страници: [Chat Room](#)

## Захвалница

Захваљујем се мојим наставницима Релји Ћурчин и Дијани Стефановић који су ми објаснили било које недоумице.

## Литература

1. Званична документација за .NET: [docs.microsoft.com](https://docs.microsoft.com)

