Višekorisnička chat aplikacija u C#-u

Antonio Kovačić
Janko Marohnić
Lana Arzon

26. lipnja 2014.

Sadržaj

1 Opis problema

Potrebno je napraviti aplikaciju za gdje se ljudi mogu međusobno dopisivati u realnom vremenu. Dvije ili više osoba se treba moći "naći na jednom mjestu", i komunicirati zajedno putem tekstualnih poruka.

Kada se osoba prijavi u chat, treba upisati svoje korisničko ime, kojim će se ta osoba identificirati. Ako je postoji druga osoba u istoj "sobi" koja već ima željeno korisničko ime, osoba bi trebala izabrati drugo korisničko ime. Nakon što je upisala korisničko ime, osoba mora moći vidjeti koji su sve drugi korisnici trenutno prijavljeni u istu "sobu".

Kada osoba pošalje poruku u grupni chat (napiše teskt i stisne "enter"), svaka druga osoba mora primiti tu istu poruku (i treba pisati od koga dolazi). Na taj način svatko zna što su svi ostali rekli. I poruke moraju stizati redom kojim su poslane.

Kada se osoba odjavi iz chata, ostalim osobama treba doći poruka da se ta osoba odjavila iz chata.

2 Implementacija

2.1 Server

2.1.1 Kreiranje

Windows servis je program koji zapravo radi u pozadini dok je korisnik prijavljen na korisnički račun unutar Windows operativnog sustava. Podešavanjem konfiguracija u `Command Promptu` Windows servis postaje aktivan. Samo kreiranje Windows servisa (detaljnim opisom što radi) se vrši kodiranjem u *C#*. Najprije kreiramo novi projekt u *C#*, odabirući opciju Windows Service. Zatim rekonstruiramo metodu *OnStart* - dodajemo kôd za otvaranje (instanciranje) kanala, registraciju objekata (dio tog koda *C#* doda automatski), itd. Zatim se instalacija vrši na jednostavan način preko Visual Studia, odnosno `Command Prompta`.

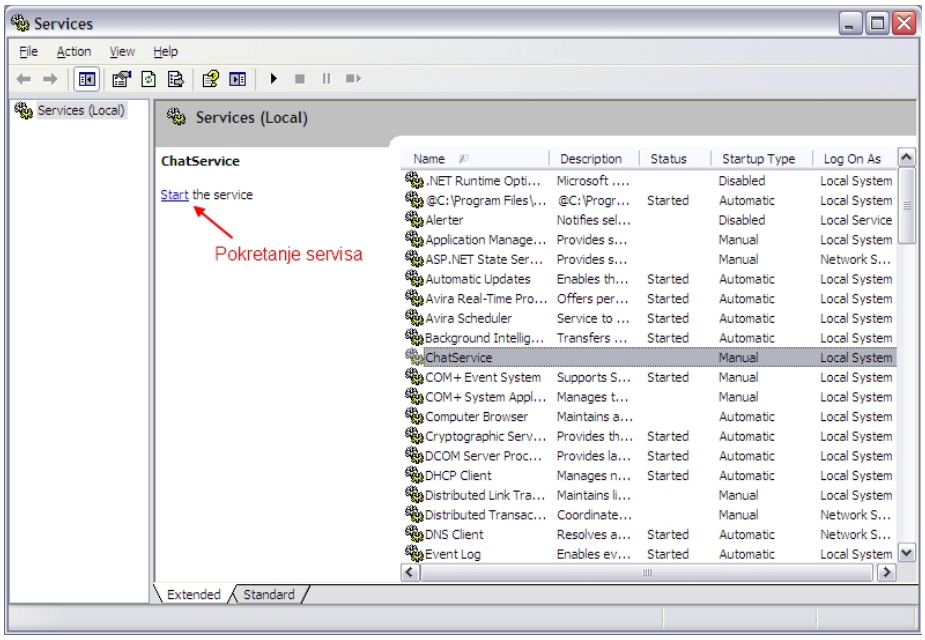
```
1 protected override void OnStart(string[] args) {
2       BinaryServerFormatterSinkProvider provider = new
            BinaryServerFormatterSinkProvider();
3       provider.TypeFilterLevel = System.Runtime.
            Serialization.Formatters.TypeFilterLevel.Full;
4       IDictionary props = new Hashtable(); props["port"]
            = 12345; //server ce slushati a portu 12345
5       TcpChannel channel = new TcpChannel(props, null,
            provider);
6       ChannelServices.RegisterChannel(channel); //
            registriranje na kanal
7       RemotingConfiguration.RegisterWellKnownServiceType(
            typeof(ChatServer), "Chat.ChatServer",
            WellKnownObjectMode.Singleton);//ime servera je
            "Chat.Chatserver"
8   }
```

2.1.2 Instaliranje

Za postavljanje servisa smo napomenuli da ga moramo konfigurirati najprije preko `Command Prompta`. Nakon prevođenja koda Windows servisa, dobivamo binarnu (izvršnu) datoteku Chat.exe. Najprije u `Command Promptu` trebamo otići u direktorij .NET Framework v. 4 naredbom:
`cd "C:\Windows\Microsoft.NET\Framework64\v4.0.30319"`
Potom naredbom:
`installutil "C:\Ciljni_ direktorij\Chat\bin\Debug\Chat.exe"`
instaliramo dani windows servis.

2.1.3 Pokretanje

Pokretanje servisa se vrši na sljedeći način: pokrenemo `Administrative tools` preko `Control Panela` (ili preko `Search programs and files` na startnoj liniji Windowsa. Zatim odaberemo opciju `Services` u kojoj nađemo `ChatService` (tako smo nazvali naš servis). Lijevim odabirom na opciju `Start` pokrećemo dani servis.

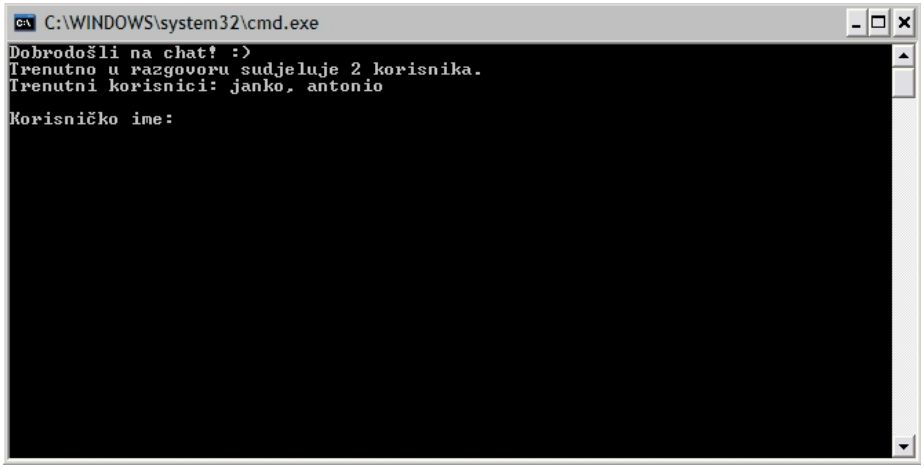


Slika 1: Pokretanje servisa ChatService

2.2 Klijent

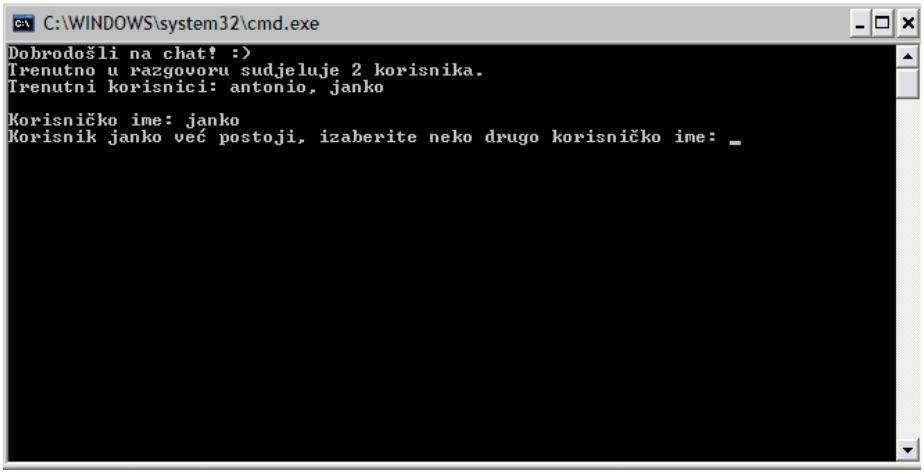
U klijentu nalazimo TCP kanal servisa, i spojimo se na server. Ako servis nije upaljen, javljamo grešku.

Zatim ispisujemo poruku dobrodošlice, i vadimo sa servisa sve trenutno prijavljene korisnike, i ispisujemo njihova korisnička imena novopridošlom korisniku.



Slika 2: Poruka dobrodošlice, broj prijavljenih korisnika i ispis njihovih korisničkih imena

Zatim ga tražimo da upiše svoje korisničko ime, s time da server provjerava postoji li već korisničko ime, i u tom slučaju traži korisnika da upiše drugo korisničko ime.

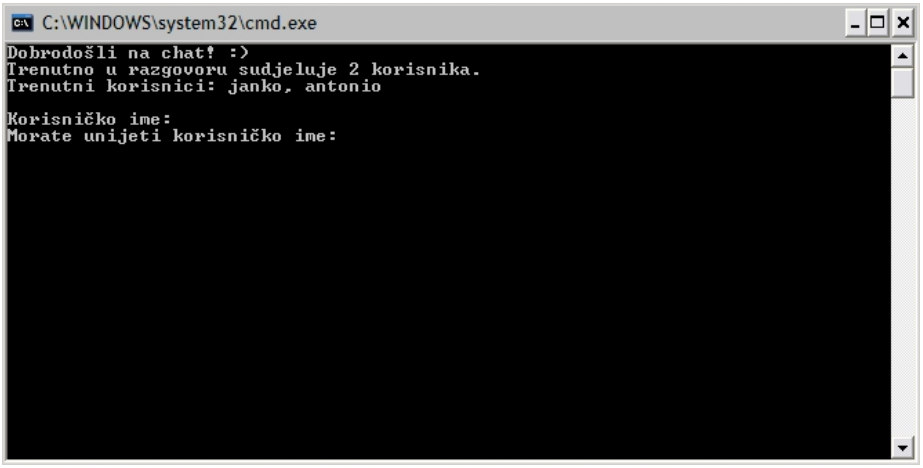


Slika 3: Nemogućnost unosa već postojećeg korisničkog imena

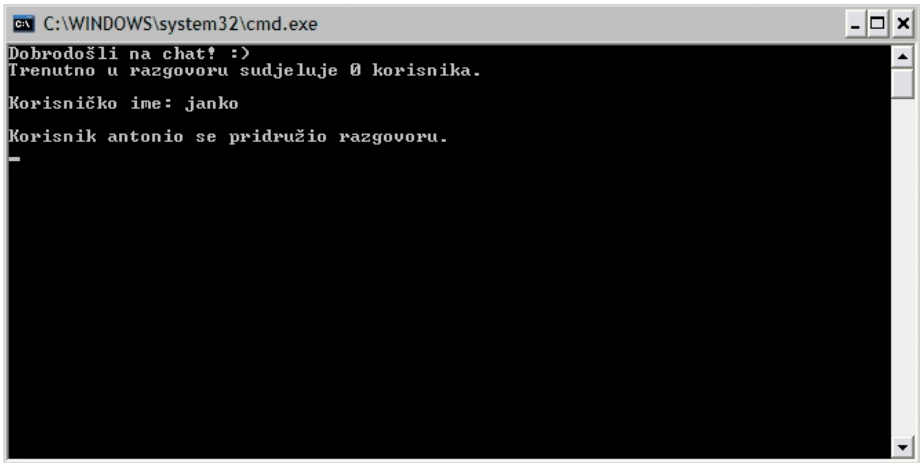
Također, unos korisničkog imena je obavezan pa ako ga nismo unijeli, dobivamo upozorenje da to moramo učiniti.

Nakon prijave se svim korisnicima šalje obavijest da se novi korisnik prijavio u chat, i korisnik se dodaje u servis.

Komunikacija se odvija na način da korisnik unese poruku koju zatim server prosljeđuje svim ostalim korisnicima koji su prijavljeni na chat.



Slika 4: Unos korisničkog imena je obavezan



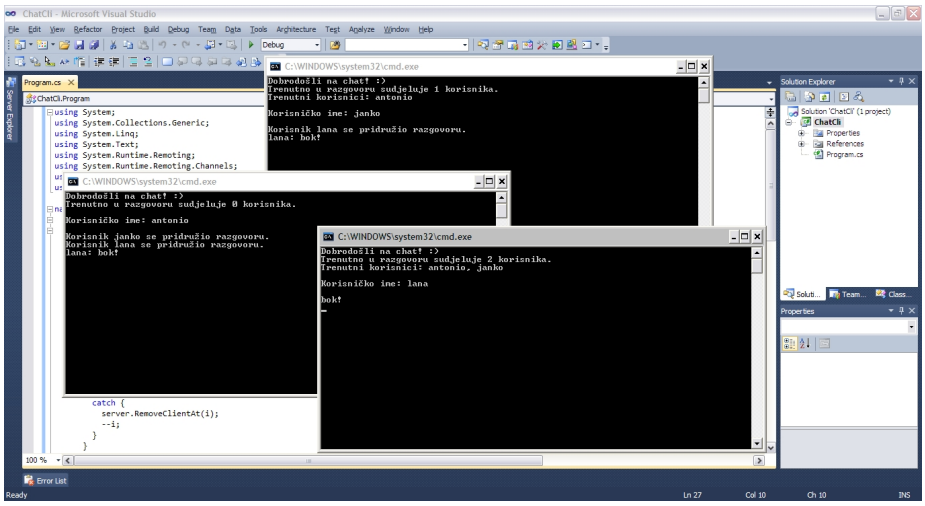
Slika 5: Korisnik se pridružje razgovoru

Korisnik može u bilo kojem trenutku napustiti razgovor te nakon što to učini, svi ostali korisnici dobivaju obavijest o tome.

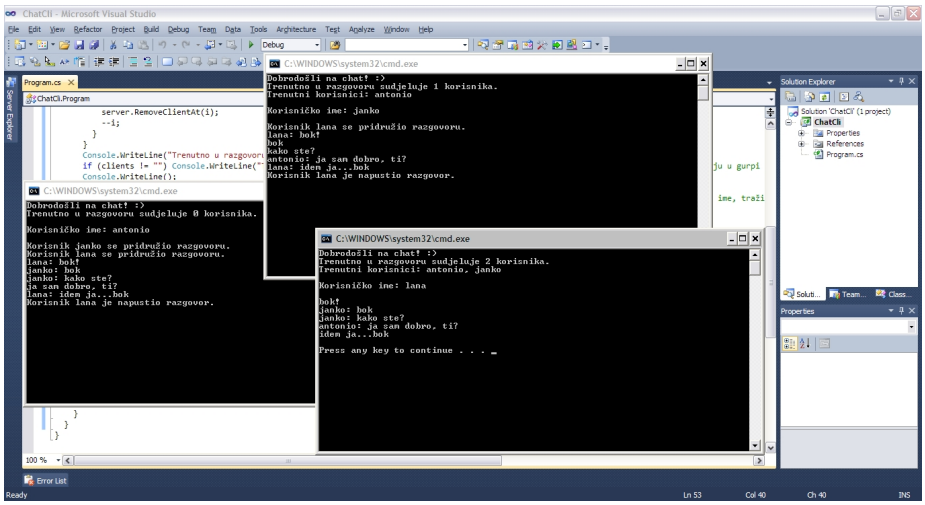
3 Rasprava

Ova implementacija ima nekoliko prednosti. Jedna je da smo koristili Windowsove servise za server, umjesto da smo implementirali svoj vlastiti. Na taj način možemo biti sigurni da je taj server optimiziran i za veći broj korisnika. Druga je što danas programeri uglavnom provode vrijeme u komandnoj liniji, pa činjenica da je naša chat aplikacija napisana kao komandno-linijska aplikacija pruža takvim korisnicima već poznato okruženje. Treća prednost je to što prestankom rada računala, prestane raditi i servis, pa nemamo problema sa eventualnim cachiranjem koje bi druge implementacije mogle napraviti.

Neke mane naše implementacije su da, ukoliko korisnik zatvori prozor, naš



Slika 6: Primjer jednostavne komumnikacije



Slika 7: Korisnik napušta razgovor

servis ne registrira da se korisnik odjavio (jer je zapravo došlo do iznimke, a naš program trenutno ne obrađuje tu iznimku). Druga mana je da kada korisnik piše poruku, a u isto mu vrijeme stigne poruka od nekog drugog korisnika, zbog ograničenja komandne linije će mu pisanje vizualno prekinuti novopristigla poruka. Korisnik još uvijek može dovršiti svoju poruku, i normalno je poslati, samo mu se može učiniti da je došlo do greške.