Aplikacija za razmjenu tekstualnih poruka unutar tematskih skupina

Antonio, Glešić

SVEUČILIŠTE U ZAGREBU
FAKULTET ORANIZACIJE I INFORMATIKE
VARAŽDIN

Antonio Glešić

Aplikacija za razmjenu tekstualnih poruka unutar tematskih skupina

ZAVRŠNI RAD

Varaždin, 2018.

SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
VARAŽDIN

Antonio Glešić
Matični broj: 43124/14-R
Studij: Poslovni sustavi

Aplikacija za razmjenu tekstualnih poruka unutar tematskih skupina

ZAVRŠNI RAD

Mentor:

Doc. dr. sc. Ivković Nikola

Varaždin, 2018.

SAŽETAK

Cilj ovoga rada je bio izraditi aplikativno rješenje pomoću kojega se omogućuje korisniku aplikacije razmjena tekstualnih poruka putem internetske mreže. Prilikom izrade aplikacija za mrežnu komunikaciju govorimo o mrežnom programiranju. Kako bismo ostvarili uspješnu komunikaciju, potrebna su znanja o mrežama računala, slojevima komunikacije putem internetskih mreža i paketima s podacima koji se šalju i primaju. Za izradu ovoga rada i aplikacije je korišten protokol MQTT s pomoću kojega se komunikacija vrši unutar tematskih skupina ili grupa. Protokol je specifičan i ograničen samo na tekstualne poruke, ali zbog široke primjene moguće ga je implementirati u raznim oblicima, bile to inačice za stolna računala, za preglednike ili za mobilne uređaje, ukratko za sve uređaje s pristupom internetu. Stoga su u ovom slučaju implementirane dvije inačice aplikativnog rješenja, a to su inačica za stolna računala te inačica za web preglednike, koja se ujedno može koristiti i putem mobilnih uređaja.

Ključne riječi: mrežno programiranje, protokol MQTT, tematske skupine, tekstualne poruke, komunikacija

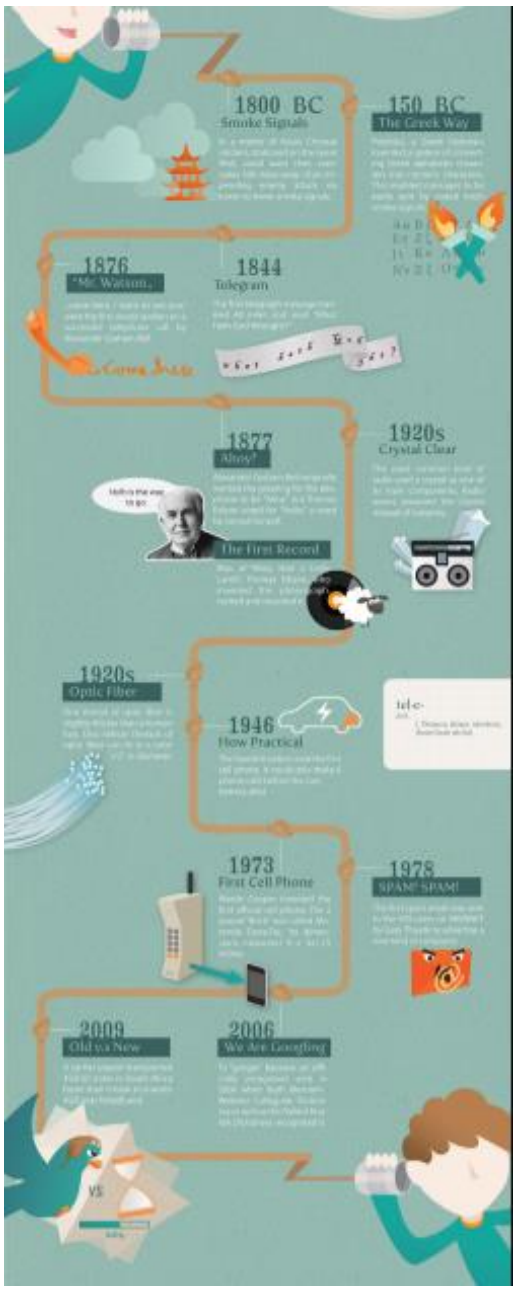
SADRŽAJ

1. UVOD

Tehnologije i načini komunikacije, odnosno razmjene poruka se konstantno razvijaju kroz povijest, a njihov razvoj možemo i vrlo jednostavno prikazati pomoću vremenske crte. Jedan od najranijih i najprimitivnijih načina komunikacije su bili dimni signali koje su kineski vojnici koristili čak 1800 godina prije nove ere. Zatim su se pojavili grčki alfabet i prve pisane riječi koje su se razmjenjivale pomoću golubova pismonoša, no to je i dalje veoma primitivno bilo sve dok se 1844. godine nije razvio telegram. Nakon izuma telegrama, razvoj je znatno ubrzan, što možemo vidjeti i izumom prvog telefona pa sve do najsuvremenijih načina komunikacije kao što su elektroničke pošte, ali sve više se koriste tehnologije za razmjenu poruka u realnom vremenu kao što su to aplikacije za razmjene poruka (tzv. *chat* aplikacije) koje se najčešće mogu pronaći na pametnim uređajima (eng. *smartphone*). [1]

Slijedno nazivu teme ovoga rada možemo zaključiti kako će se isti usredotočiti na aplikaciju čija će svrha biti razmjena tekstualnih poruka između tematskih grupa. Za izradu ove aplikacije je dogovoreno korištenje protokola MQTT, a kako je isti veoma rasprostranjen, moguće je bilo ostvariti programsko rješenje u obliku *desktop* aplikacije te isto tako i web aplikacije za korištenje pomoću web preglednika.

Iduća će se poglavlja dotaknuti tehnologija korištenih pri izradi rješenja ove aplikacije te će se detaljnije razmotriti sam protokol MQTT i razlog odabira upravo tog protokola.



Slika 1. *Vremenska crta razvoja tehnologija komunikacije (2017)*

2. TEHNOLOGIJE

Kako je programsko rješenje ovoga rada aplikacija za razmjenu tekstualnih poruka, to znači da je mrežna aplikacija, odnosno radi se o mrežnom programiranju. Mrežno programiranje označava izradu aplikacija koje šalju i/ili primaju podatke preko internetske mreže. Kao jednu od najbitnijih tehnologija mrežnog programiranja možemo istaknuti Sockete.

Što se tiče Socket programiranja, oni omogućavaju aplikacijama mogućnost komunikacije koristeći mreže. Jednostavnije rečeno, oni su svojevrsne točke ili poveznice koje primaju i/ili šalju podatke sa istog ili drugog računala/uređaja. [2]

Postoji više vrsta socketa, a najpoznatiji su socketi za protokole transportnog sloja TCP i UDP. Za izradu ove aplikacije korišteni su socketi za protokol MQTT. Iako se poruke MQTT-a mogu prenositi tako da programer sam generira zaglavlja poruka i šalje ih socketima za TCP ili TLS/SSL uporaba gotovih socketa za MQTT bitno olakšava posao programera.

2.1. PROTOKOL MQTT

MQTT znači MQ Telemetry Transport. On je protokol koji se koristi sa poslužiteljske jednako kao i sa klijentske strane. U principu je vrlo jednostavan objavi/pretplati transport protokol za razmjenu poruka. Lagan je, otvoren i veoma jednostavan, dizajniran tako da ga se s lakoćom može na raznim mjestima implementirati. Te karakteristike pogoduju njegovoj širokoj primjeni, kako u komunikaciji između raznih uređaja i strojeva (M2M – Machine to Machine), tako i u IoT (Internet of Things) kontekstu. [3]

Protokol MQTT se koristi TCP/IP protokolima. Veoma je jednostavan za shvatiti i koristiti. On ima nekoliko glavnih značajki:

- Mogućnost objave neke tekstualne poruke na određenu temu
- Mogućnost pretplate na određenu temu i primanje poruka za istu
- Razine kvalitete usluge (QoS – Quality of Service)

2.2. PONAŠANJE PROTOKOLA MQTT

Kako je već ranije utvrđeno da je protokol MQTT relativno jednostavan za shvatiti, prije samog objašnjenja aplikacije, valja se detaljnije dotaknuti glavnih značajki kako funkcionira korišteni protokol.

Za potrebe aplikacije koja služi za razmjenu tekstualnih poruka, protokolu je potreban način za slanje, ali i primanje poruka. Ovdje dolazi specifičnost protokola MQTT. Naveli smo da su mu glavne značajke slanje poruka na određenu temu i primanje poruka, odnosno pretplata na neku temu.

Ovakav rad s temama, odnosno tematskim grupama omogućuje razmjenu tekstualnih poruka između većeg broja korisnika vrlo jednostavno i brzo. Svaki korisnik kako bi poslao poruku, mora odrediti temu tj. tematsku skupinu kojoj želi poslati. Nakon slanja poruke, ona ne odlazi direktno krajnjim korisnicima, nego se sadržaj (payload) šalje udaljenom poslužitelju (remote server), koji zatim prosljeđuje sadržaj poruke svim krajnjim korisnicima koji su pretplaćeni na temu kojoj prvotni korisnik šalje poruku.

Ujedno je i objašnjena značajka za primanje poruka, odnosno pretplatu na neku temu. Svaki korisnik ima mogućnost pretplaćivanja na teme, odnosno tematske grupe za koje želi primati poruke. Jedan korisnik može poslati poruku istovremeno samo na jednu temu, ali može biti pretplaćen na više tema i primati poruke od više njih. Samo primanje poruka je već prethodno spomenuto kako korisnik nikad ne prima sadržaj poruke direktno od drugog korisnika, nego se poruke primaju sa udaljenih poslužitelja na koje korisnici prije samog korištenja aplikacije moraju uspostaviti vezu.

3. O APLIKACIJI

3.1. SVRHA APLIKACIJE

Njena svrha je razmjena tekstualnih poruka između korisnika unutar određenih tematskih skupina, što bi značilo da je veoma slična većini ostalih aplikacija namijenjenih za razmjenu tekstualnih poruka, ali kako se koristi protokol MQTT, donosi malo drugačije iskustvo putem tematskih skupina koje budu detaljnije promatrane kasnije tokom rada. Aplikacija je relativno jednostavne prirode, nema korisnika različitih uloga i prava, svi su korisnici jednaki i imaju iste funkcionalnosti. Nastoji se omogućiti korisnicima brz i efikasan način komunikacije odnosno razmjene tekstualnih poruka između sebe.

3.2. NAČIN IZRADE APLIKACIJE

Zbog široke mogućnosti primjene protokola MQTT i jednostavnosti njegove implementacije, odluka o vrsti aplikacije je bila istovremeno veoma lagana, ali i teška. Stoga su kao aplikativna rješenja ovoga rada priložene *desktop* inačica aplikacije zajedno sa inačicom za web preglednike, kako bi istovremeno korisnici računala, koji mogu koristiti obje inačice, dakle *desktop* aplikaciju i web aplikaciju, mogli komunicirati zajedno sa korisnicima mobilnih uređaja kojima je dostupna inačica putem preglednika.

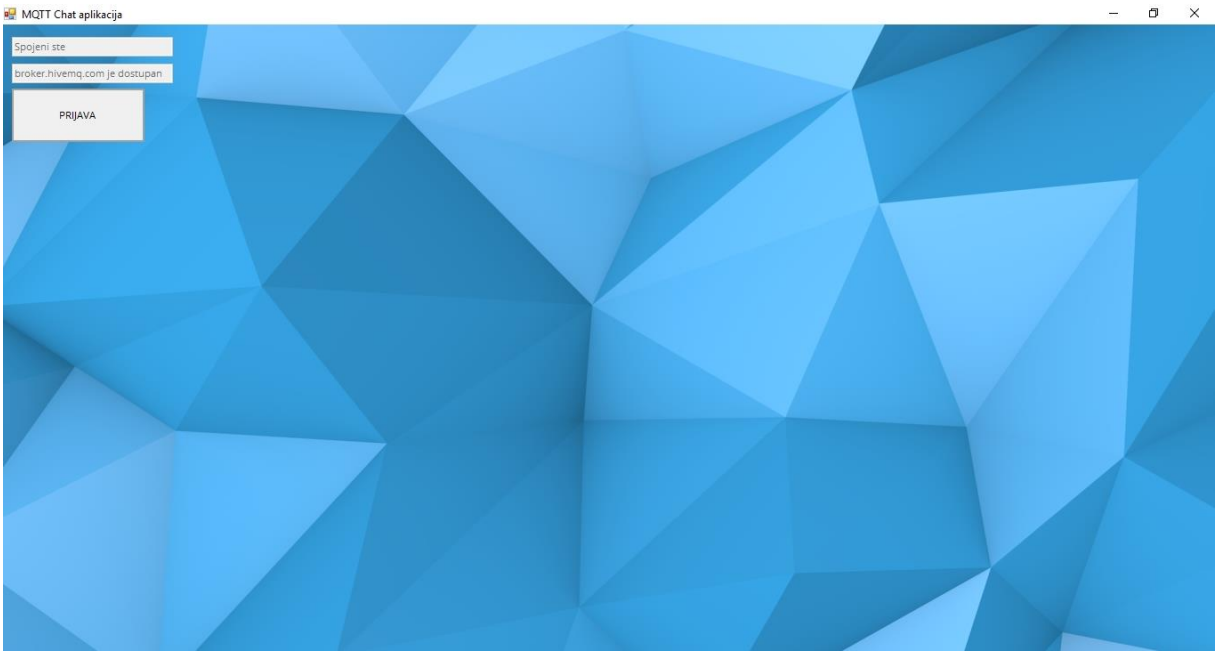
Za samu izradu aplikacija korišteni su dva različita IDE okružja. *Desktop* inačica je rađena u *Microsoftovom Visual Studio 2017* alatu zbog jednostavnosti izrade sučelja i implementacije biblioteke protokola MQTT, dok je inačica za web preglednik izrađena u *Netbeans IDE-u*.

Desktop aplikacija je kompletno implementirana u C# programskom jeziku, čije ćemo dijelove koda u kasnijim poglavljima detaljnije razmotriti. Za razliku od *desktop* inačice, web aplikacija je implementirana pomoću HTML-a i CSS-a, a kako je biblioteka protokola MQTT za web aplikacije pisana u JAVASCRIPT programskom jeziku, pozadinski (*backend*) dio web aplikacije je rađen isto tako JAVASCRIPT-om. U kasnijem dijelu ovoga rada ćemo moći i usporediti obje inačice te razmotriti sličnosti i razlike.

3.3. POKRETANJE DESKTOP INAČICE

Protokol MQTT nema načina za registraciju korisnika, ali ima mogućnost prilikom uspostave veze klijenta sa poslužiteljem da zabilježi korisnika, odnosno njegovo korisničko ime. Stoga u aplikaciji nije implementirana registracija korisnika, ali prilikom pokretanja aplikacije, korisnik se mora prijaviti sa određenim korisničkim imenom koje se zatim šalje poslužitelju i sadržaj poruka je u pozadini implementiran tako da se prije samog sadržaja poruke nadoda i korisničko ime kako bi se znalo koji korisnik šalje točno koju poruku.

Prilikom samog pokretanja *desktop* aplikacije, korisniku se otvara početna forma koja se sastoji od dva okvira sa tekstom i dugmetom za prijavu. Aplikacija prilikom pokretanja početne forme koristi testne podatke za korisničko ime i pokušava uspostaviti vezu sa unaprijed određenim poslužiteljem te se u tekstualnim okvirima prikazuje status povezivanja i adresa poslužitelja kako bi korisnik, ukoliko ne zna adresu nijednog drugog poslužitelja, mogao koristiti upravo taj unaprijed određeni poslužitelj.



Slika 2. *Početna forma desktop aplikacije*

Kao što se na slici može vidjeti, aplikacija je uspješno pokrenuta i uspješno je uspostavljena veza sa poslužiteljem „*broker.hivemq.com*“. Pritiskom na dugme PRIJAVA korisnika se preusmjerava na formu za prijavu, gdje unosi korisničko ime te ima mogućnost povezivanja na poslužitelj i biranje tematskih skupina.

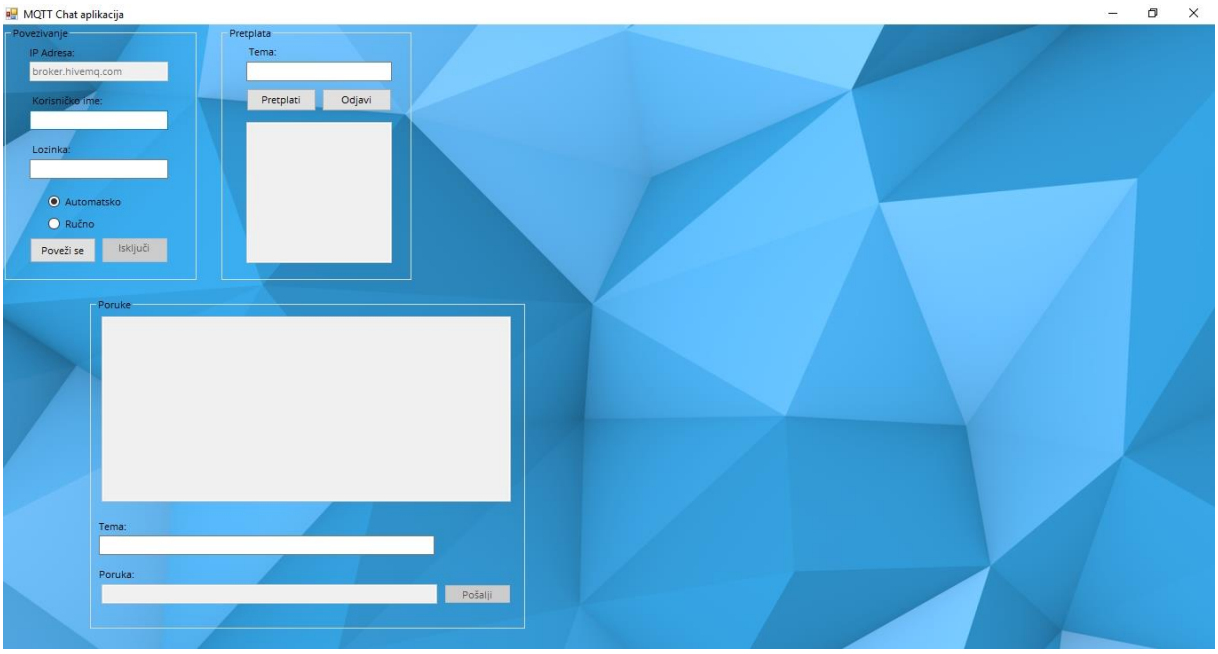
3.4. PROZOR KORISNIKA DESKTOP INAČICE

Nakon što korisnik pritisne dugme PRIJAVA na početnoj formi *desktop* aplikacije, ona ga preusmjerava na iduću formu aplikacije, gdje prije nego ima mogućnost korištenja ostalih funkcionalnosti, korisnik mora unijeti ispravno korisničko ime i lozinku te ima mogućnost automatskog povezivanja na unaprijed određeni poslužitelj ili ručno povezivanje, gdje unosi IP adresu i port poslužitelja na koji se želi povezati.

Nakon što se korisnik uspješno poveže na poslužitelj pritiskom na dugme Poveži se, omogućene su mu funkcionalnosti pretplate na tematske skupine i slanje, odnosno primanje poruka od istih.

Okvir za pretplaćivanje na tematske skupine se sastoji od tekstualnog okvira za unos naziva teme odnosno tematske skupine, dugmeta za pretplatu na temu, okvira sa popisom svih pretplaćenih tema te dugmeta za odjavu pretplate na odabranu tematsku skupinu u okviru popisa pretplaćenih tema.

Okvir za slanje i primanje poruka se nalazi ispod prijašnja dva okvira te se sastoji od tri tekstualna okvira i dugmeta. Najveći tekstualni okvir je okvir u kojemu se prikazuju pristigle poruke, dok su dva manja tekstualna okvira zaslužna za slanje poruka. Korisnik nužno mora prije slanja sadržaja poruke u tekstualni okvir za temu napisati naziv tematske skupine u koju šalje poruku, tek tada će se uspješno poslati poruka upravo na tu tematsku skupinu.

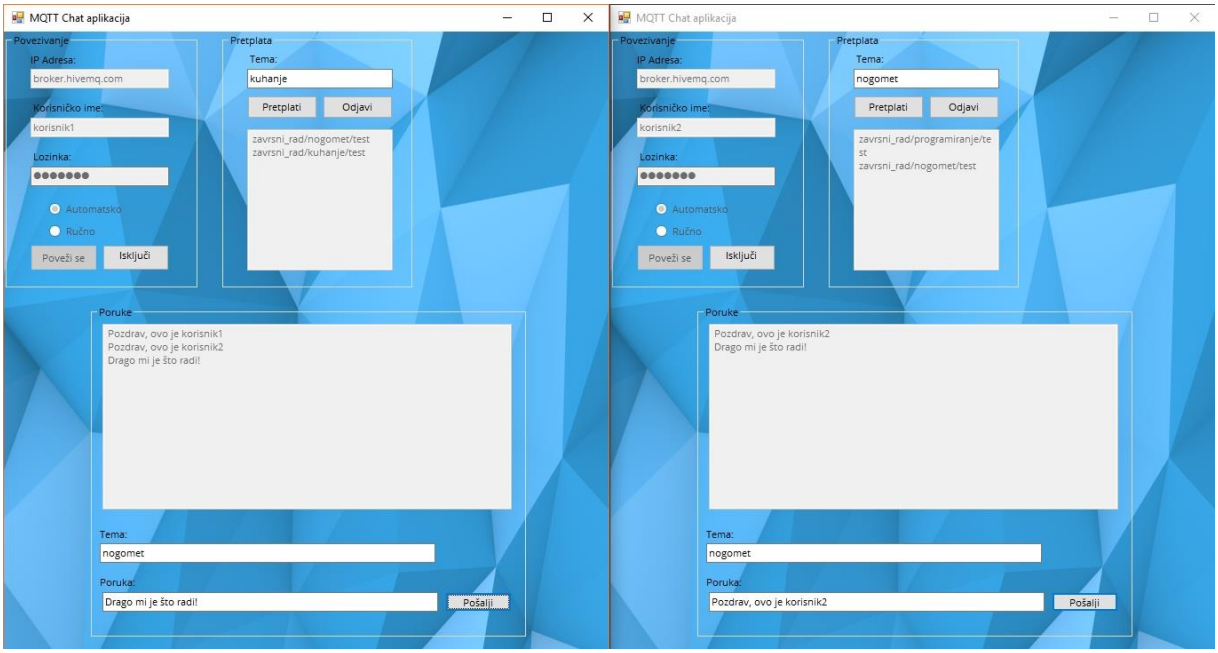


Slika 3. *Forma funkcionalnosti desktop aplikacije*

3.5. RAZMJENA PORUKA IZMEĐU DVA KORISNIKA DESKTOP INAČICE

Kako je sama svrha ove aplikacije razmjena poruka između više korisnika, u svrhu isprobavanja su pokrenute dvije instance aplikacije na računalu te su prijavljena dva različita korisnika sa korisničkim imenima „korisnik1“ i „korisnik2“ jer nije bilo potrebe za kompliciranjem.

Prilikom testiranja, instanca korisnik1 se prvo pretplatila na tematsku skupinu naziva „nogomet“ i nakon toga na tematsku skupinu „kuhanje“, dok se nakon toga instanca korisnik2 pretplatila na tematsku skupinu „programiranje“, a nakon toga tek na zajedničku temu naziva „nogomet“.



Slika 4. *Razmjena poruka između dva korisnika desktop inačice*

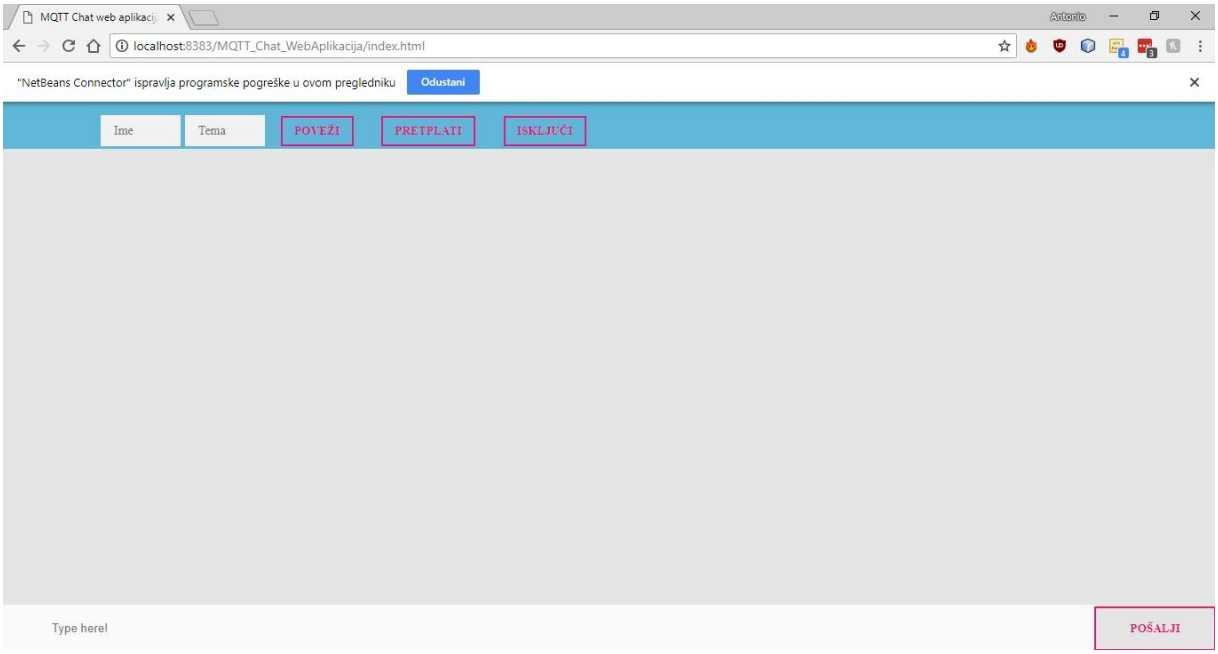
Kako se na slici broj 4 može vidjeti, korisnik1 je poslao prvo poruku sadržaja „Pozdrav, ovo je korisnik1“, ali ona nije stigla kod korisnika2. Razlog je vrlo jednostavan jer je korisnik1 poslao poruku u tematsku skupinu „nogomet“ prije nego je korisnik2 bio pretplaćen na tu istu tematsku skupinu. Tako možemo utvrditi da aplikacija u potpunosti ispravno radi koristeći protokol MQTT i njegove funkcionalnosti te nema propusta.

3.6. POKRETANJE WEB INAČICE

Kako je ranije već napomenuto, uz *Windows Forms* inačicu, izrađena je i inačica za web preglednik. Ona je ponešto jednostavnija od *desktop* inačice, odnosno nema mogućnosti biranja poslužitelja i odjavljivanja pretplate na određene tematske skupine.

Web aplikacija je dizajnom nadahnuta od strane IRC tehnologije. IRC (*Internet Relay Chat*) je tehnologija za komunikaciju putem interneta nastala 1988. godine te se koristi još danas. IRC tehnologija se bazira na tome da korisnici biraju poslužitelje i kanale na poslužitelju, slično protokolu MQTT, gdje međusobno razmjenjuju tekstualne poruke putem TCP ili SSL protokola. [4]

Prilikom samog pokretanja aplikacije u web pregledniku, sučelje je kao što je naglašeno veoma slično IRC tehnologiji, odnosno većini ostalih svojevrsnih *chat* aplikacija. Pri vrhu ekrana se nalazi traka sa dva tekstualna okvira od kojih jedan služi za ime korisnika, a drugi za temu na koju se pretplaćuje i šalje poruku. Uz tekstualne okvire, nalaze se tri dugmeta, jedan za uspostavljanje veze sa poslužiteljem, drugi za pretplatu na tematsku skupinu, a treći za uništavanje veze sa poslužiteljem. Za svaku izvršenu akciju se dobiva povratna informacija u okviru predviđenom za poruke. U samoj sredini ekrana se nalazi praznina predviđena za primljene poruke. Na dnu ekrana se nalazi tekstualni okvir za pisanje poruke koju korisnik želi poslati i dugme za slanje.

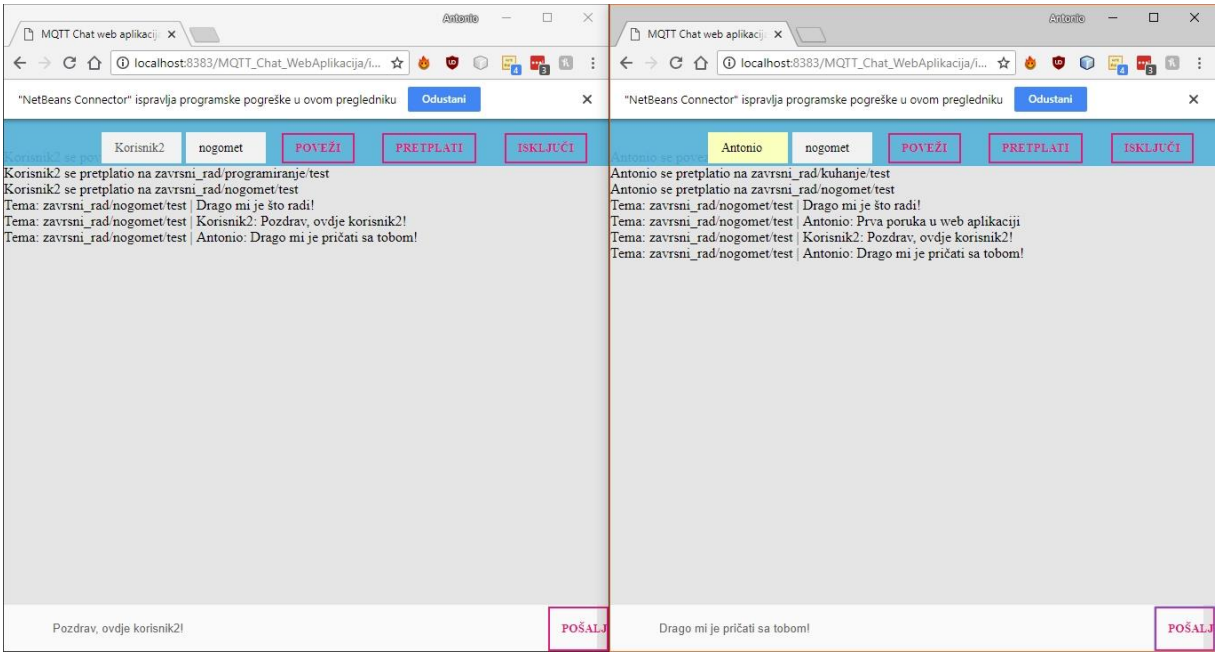


Slika 5. *Pokrenuta web aplikacija za razmjenu poruka*

Ranije je već navedeno kako su za izradu web aplikacije korišteni HTML, CSS i JAVASCRIPT programski jezici. U kasnijim poglavljima bude prikazan i kod aplikacije, no valja napomenuti kako je HTML korišten za temelj odnosno kostur aplikacije, a CSS za oblikovanje i dizajn. Sve pozadinske procese kao uspostavljanje veze, pretplata na tematske skupine, slanje i primanje poruka zajedno sa uništavanjem veze na poslužitelja su pisane pomoću JAVASCRIPT-a.

3.7. RAZMJENA PORUKA IZMEĐU DVA KORISNIKA WEB APLIKACIJE

Slično kao i kod inačice za *desktop*, inačica za web preglednike je testirana prvo sa dva korisnika, jedan pod nazivom „Antonio“, a drugi nazivom „korisnik2“. Korisnici pri pokretanju web aplikacije prvo upisuju svoje ime, zatim pritiskom na prvo dugme moraju uspostaviti vezu s poslužiteljem koji je unaprijed definiran, a zatim upisuju naziv tematske skupine na koju se žele pretplatiti. Nakon upisivanja naziva tematske skupine, pritiskom na dugme za pretplatu se pretplaćuju te od tog trenutka zaprimaju sve poruke vezane uz nju. Postoji još i dugme za uništavanje veze s poslužiteljem. Veza s poslužiteljem se, kao što se može i predvidjeti, uništi prilikom zatvaranja kartice unutar preglednika i/ili gašenjem samog preglednika od strane korisnika, no uz taj način, korisnik može uništiti vezu putem dugmeta te tako ostati unutar aplikacije, a razlog tome je veoma jednostavan. Kako ne postoji direktan način za odjavu pretplate na neku tematsku skupinu i ukoliko korisnik želi promijeniti svoje ime unutar aplikacije, pritiskom na dugme za uništenje veze omogućava se izmjena imena korisnika i ponovno uspostavljanje veze sa novim imenom. Paralelno s time, korisniku je omogućeno pretplaćivanje na tematske skupine koristeći svoj novi naziv pod kojim je uspostavljena nova veza na poslužitelj, tako korisnik ne mora bespotrebno osvježavati preglednik i ponovno pokretati aplikaciju. Kako bi korisnik poslao tekstualnu poruku, mora je upisati u tekstualni okvir koji se nalazi pri dnu ekrana, odnosno preglednika i pritiskom na dugme za slanje poruka se šalje poslužitelju koji zatim jednako kao i kod *desktop* aplikacije šalje poruku svim korisnicima koji su ujedno i pretplaćeni na tu tematsku grupu.



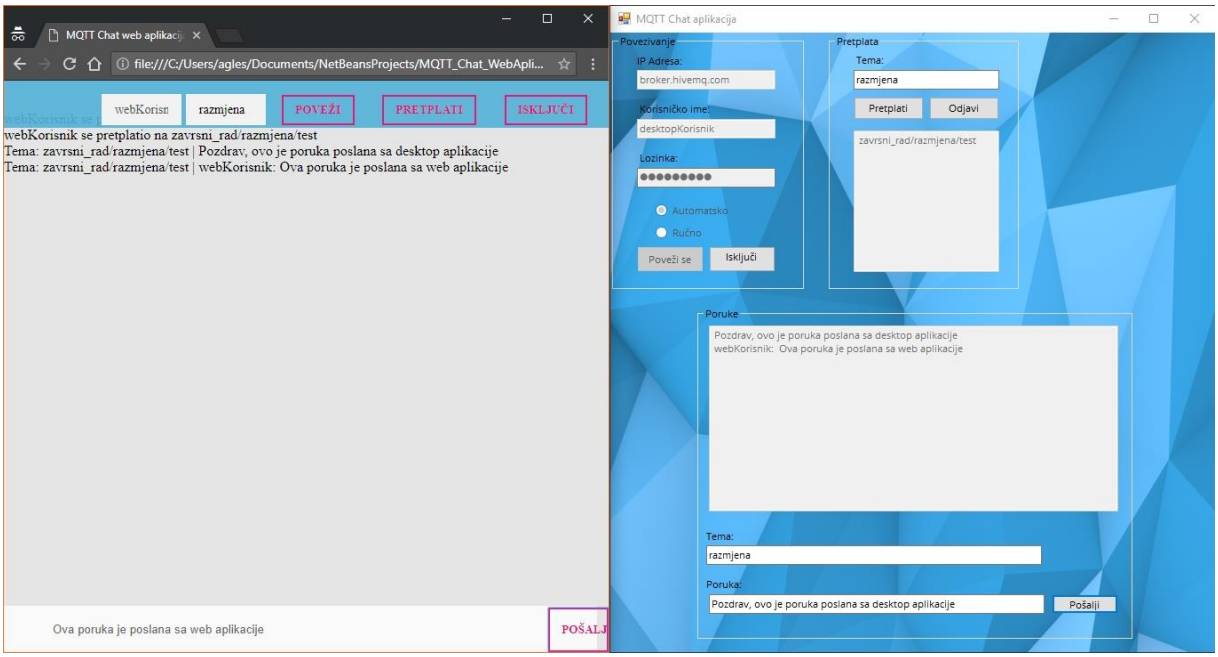
Slika 6. *Razmjena poruka između dva korisnika web aplikacije*

Kao što se može vidjeti na slici broj 6, korišteni su slični podaci kao i za *desktop* inačicu aplikacije, sa manjim izmjenama. Jedan korisnik je koristio ime „Antonio“, dok drugi je samo „Korisnik2“. Slično kao i kod testiranja *desktop* inačice, jedan je korisnik pretplaćen na tematske skupine „kuhanje“ i „nogomet“, dok je drugi korisnik pretplaćen na „programiranje“ i naravno kao zajedničku tematsku skupinu isto koristi „nogomet“. Kao specifičnost web aplikacije u usporedbi sa *desktop* inačicom uočljivo je kako prije samog sadržaja tekstualne poruke bude prikazan naziv teme na koju je objavljena poruka, zajedno sa nazivom korisnika koji je objavio poruku. Zbog jednostavnosti korištenja protokola MQTT pomoću JAVASCRIPT jezika nam je to omogućeno. Zatim valja uočiti kako naizmjenično korisnici objavljuju poruke koje im se naravno prikazuju u dijelu predviđenom za prikaz razmjene poruka. Stoga može jednako kao i za *desktop* inačicu biti rečeno da i web aplikacija radi besprijekorno i da nisu uočeni problemi, odnosno nije došlo do nekih komplikacija pri ostvarenju programskog rješenja.

3.9. RAZMJENA PORUKA IZMEĐU KORISNIKA RAZLIČITIH INAČICA

Dakle, uspješno su testirane obje inačice aplikacije, *desktop* i web inačice. Zatim je bio objašnjen princip prema kojemu funkcioniraju i dokazano je da komunikacija između dva ili više korisnika u potpunosti ispravno radi bez ikakvih komplikacija. Postoji još jedan moguć scenarij prilikom korištenja aplikacije, a to je komunikacija između dva korisnika dviju različitih inačica.

Kako bi ova komunikacija uspješno funkcionirala, postoji samo jedan uvjet koji mora biti postignut. Neovisno o korisničkim imenima te o poslužitelju s kojim su korisnici uspostavili vezu, samo je jedan uvjet nužan, a to je da oba korisnika budu pretplaćena, odnosno da objavljuju na istu temu tj. tematsku skupinu. Stoga ćemo pokrenuti *desktop* inačicu istovremeno sa inačicom za web preglednike te ćemo sa obje instance aplikacije se pretplatiti na istu tematsku skupinu i objaviti nekoliko tekstualnih poruka na tu zajedničku tematsku skupinu. Ukoliko je komunikacija između dviju različitih inačica uspješno uspostavljena pomoću protokola MQTT, biti će prikazane iste poruke na obje inačice aplikacije.



Slika 7. *Razmjena tekstualnih poruka između korisnika obiju inačica*

Sa slike broj 7 je vidljivo kako su opet korišteni testni podaci za testiranje komunikacije između različitih inačica aplikacije. Korisnik *desktop* inačice aplikacije se nazvao „desktopKorisnik“, a slično tome se korisnik inačice za web preglednike nazvao „webKorisnik“.

Kako je još ranije ustanovljeno provođenjem testova, obje inačice aplikacije funkcioniraju u potpunosti bez ikakvih komplikacija, stoga za ovaj posljednji test nisu provođena prijašnja testiranja. Oba korisnika, tj. obje inačice aplikacije su uspostavile vezu sa već unaprijed određenim poslužiteljem (*broker.hivemq.com*).

Kako bi ova vrsta komunikacije između dviju različitih inačica bila moguća, odnosno ostvariva, obje inačice moraju biti pretplaćene na istu tematsku skupinu, što je vidljivo i na slici. Oba korisnika su pretplaćena na tematsku skupinu „razmjena“ te objavljuju tekstualne poruke na istu. Poruke se tako uspješno šalju poslužitelju i uspješno zaprimaju od istog.

Dosad još nisu bili prikazani načini implementacije, odnosno isječci programskog koda, no nakon prikazanih funkcionalnosti i njihovog rada, u idućem dijelu će biti prikazan pozadinski dio istih, odnosno aplikativno rješenje u obliku programskog koda.

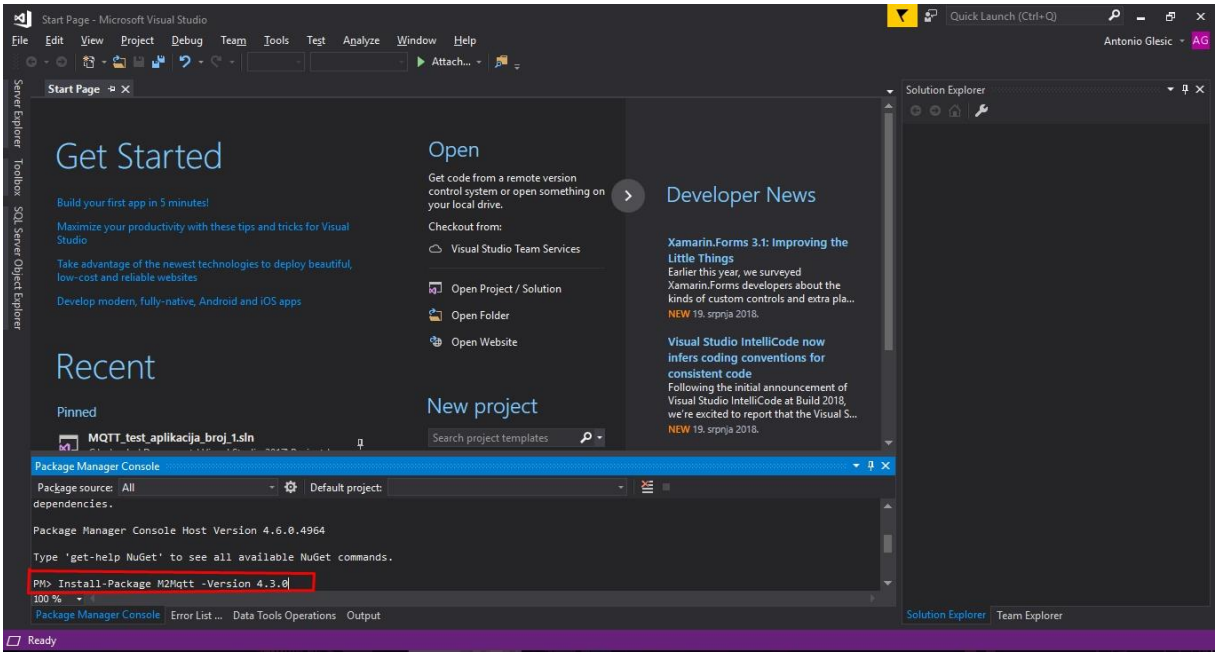
4. IMPLEMENTACIJA

U prethodnim poglavljima je bio prikazan izgled, odnosno sučelje aplikacije za razmjenu tekstualnih poruka unutar tematskih skupina i to za obje inačice, a to su inačica za windows okružje i inačica za web preglednike. Uz samo sučelje aplikacija, prikazane su i sve njihove funkcionalnosti zajedno sa testnim primjerima pokrenutih aplikacija i njihovo korištenje. Pomoću primjera i testiranja je lako uočljivo kako je komunikacija, odnosno razmjena tekstualnih poruka unutar tematskih skupina ostvarena i ne samo zasebno za svaku inačicu, nego i komunikaciju između korisnika različitih inačica aplikacije.

Dosad još nije prikazan dio s programskim kodom kako je to zapravo u pozadini sve ostvareno, no to će biti prikazano u sljedećem dijelu ponešto detaljnije, naravno neće biti prikazana svaka linija koda, nego samo važniji i kritični dijelovi za najvažnije dijelove funkcionalnosti aplikacija.

4.1. UVOĐENJE BIBLIOTEKE MQTT U DESKTOP APLIKACIJU

Kako protokol MQTT nije protokol za mrežno programiranje koji dolazi u standardnim paketima, odnosno odmah uz alat ili IDE koji se koristio za ostvarenje ove aplikacije. Stoga bude prikazano za oba programska rješenja kako su uvedene biblioteke za protokol MQTT.



Slika 8. *Uvođenje biblioteke MQTT za .NET framework pomoću NuGet-a*

Korisnik prije kreiranja samog projekta u *.NET frameworku* treba prvo dohvatiti biblioteku MQTT. Sama biblioteka je paket naziva „M2Mqtt“ (*machine-2-machine*), a moguće ju je dohvatiti pomoću *NuGet package* menadžer alata unutar *Microsoft Visual Studio 2017* IDE-a koji je korišten. Prilikom dohvaćanja biblioteke, otvara se konzola NuGet menadžera te se upisuje naredba za instalaciju paketa, naziv paketa i njegova verzija koju želimo, po mogućnosti posljednje izdana, odnosno najnovija verzija. Nakon dohvaćanja i instalacije paketa, on je na raspolaganju korisniku te prilikom idućeg kreiranja novog projekta u *Visual Studio* alatu može dodati paket ili u ovom slučaju biblioteku u svoj projekt. [5]

```
pocetnaFrm.cs | prijavaFrm.cs [Design]
MQTT_test_aplikacija_broj_1 | MQTT_test_aplikacija_broj_1.pocetnaFrm | btnPrijava_Click(object sender, EventArgs e)
1   using System;
2   using System.Collections.Generic;
3   using System.ComponentModel;
4   using System.Data;
5   using System.Drawing;
6   using System.Linq;
7   using System.Net; //Ukljucena biblioteka za web programiranje
8   using System.Text;
9   using System.Threading.Tasks;
10  using System.Windows.Forms;
11  using uPLibrary.Networking.M2Mqtt; //Ukljucena biblioteka za koristenje MQTT protokola
12  using uPLibrary.Networking.M2Mqtt.Messages; //Ukljucena biblioteka za slanje/primanje poruka preko MQTT protokola
13
```

Slika 9. *Uključivanje biblioteke u .NET projektu*

Prilikom kreiranja početne forme projekta, u pozadini forme se prvenstveno mora uključiti biblioteka za web programiranje, a zatim sama protokol MQTT biblioteka (M2Mqtt). Nakon toga je sve spremno i može se krenuti s aktivnim korištenjem protokola MQTT za svrhu ove aplikacije.

4.2. UVOĐENJE BIBLIOTEKE MQTT U WEB INAČICU

Uvođenje odnosno uključivanje biblioteke protokola MQTT za web aplikaciju je nešto jednostavnije nego kod *desktop* inačice. Kako je biblioteka za protokol MQTT koja je korištena ovdje zapravo biblioteka za Javascript, ona se jednako kao i Jquery biblioteka uključuje u zaglavlju HTML datoteke.

```
<!DOCTYPE html>
<html>
    <head>
        <title>MQTT Chat web aplikacija</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <link href='//fonts.googleapis.com/css?family=Lato:300,400' rel='stylesheet' type='text/css'>
        <link href='//fonts.googleapis.com/css?family=Lekton:400,700' rel='stylesheet' type='text/css'><
        <link href="//netdna.bootstrapcdn.com/font-awesome/4.0.3/css/font-awesome.css" rel="stylesheet">
        <link rel="stylesheet" type="text/css" href="css/antglesic.css">
        <script src="http://ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.min.js"></script>
        <script src="http://www.hivemq.com/demos/websocket-client/js/mqttws31.js"></script>
    </head>
```

Slika 10. *Uključivanje Javascript biblioteke za protokol MQTT*

Na slici je vidljivo kako se u samom zaglavlju HTML datoteke dodaje biblioteka za protokol MQTT u oznakama za skripte. U samom tekstu između oznaka za skripte možemo vidjeti link sa kojeg se preuzima biblioteka, naziv same datoteke i njena ekstenzija koja je „js“. Stoga je vrlo lako za prepoznati kako je to biblioteka Javascript programski jezik, a pomoću kojega je ujedno i ostvarena web aplikacija za razmjenu tekstualnih poruka unutar tematskih skupina. [8]

4.3. USPOSTAVLJANJE VEZE S POSLUŽITELJEM .NET INAČICE

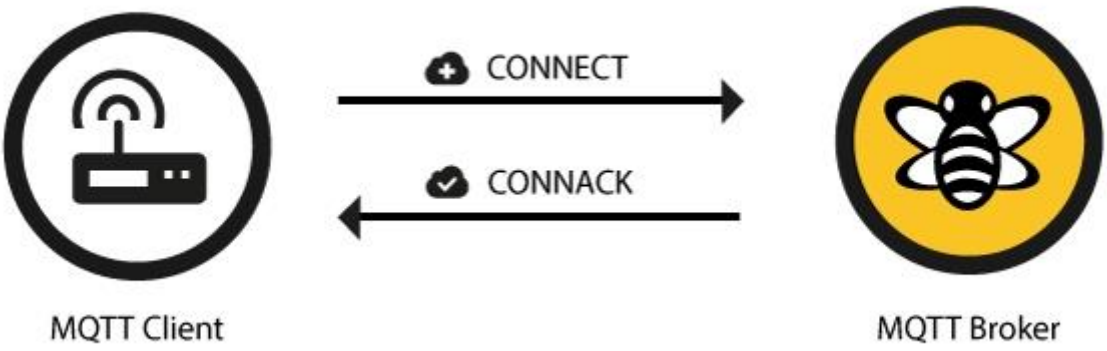
Prilikom samog pokretanja aplikacije, prije nego su korisniku omogućene sve funkcionalnosti, on prvo mora uspostaviti uspješnu vezu sa poslužiteljem preko kojega želi uspostaviti komunikaciju, odnosno razmjenu tekstualnih poruka unutar određenih tematskih skupina.

```
public partial class pocetnaFrm : Form
{
    //Prilikom pokretanja pocetne forme instanciramo MQTT klasu i povezujemo se na brokera
    MqttClient Client = new MqttClient("broker.hivemq.com");

    public pocetnaFrm()
    {
        InitializeComponent();
    }

    private void pocetnaForma_Load(object sender, EventArgs e)
    {
        //Rucno podesavanje najnovije verzije protokola
        //Napomena: verziju protokola treba podesiti prije pozivanja Connect() metode
        Client.ProtocolVersion = MqttProtocolVersion.Version_3_1_1;
        //Uspostavljanje veze pomocu korisnickog imena i lozinke
        byte code = Client.Connect(Guid.NewGuid().ToString(), "korisnikTest", "lozinkaTest");

        if(code == 0) //Ukoliko je uspješno uspostavljena veza
        {
            outputSpajanje.Text = "Spojeni ste";
            outputPretplata.Text = "broker.hivemq.com je dostupan";
        }
```

Slika 11. *Programski kod za uspostavljanje veze s poslužiteljem za .NET inačicu*

Kako je programski jezik C# kompletno objektno orijentirano okružje, tako se u *.NET frameworku* biblioteka MQTT ponaša tako da se na klijentskoj strani aplikacije radi pomoću instance klase odnosno objekta. Na slici broj 11 je uočljivo kako se prilikom samog učitavanja forme kreira objekt *MqttClient*, čiji konstruktor prima argument adrese poslužitelja na koji se povezujemo. Zatim valja napomenuti kako se protokolu MQTT vrlo jednostavno može odrediti verzija koja će se koristiti u aplikaciji, ali kao što je navedeno u napomeni, verziju protokola treba odrediti prije povezivanja. Kako kasnije budu razmotrene sve moguće vrste poruka koje korisnik dobiva kao povratnu informaciju od poslužitelja prilikom povezivanja ili pokušaja uspostave veze, zasad je dovoljno znati da se ta povratna informacija pohranjuje u varijablu tipa *byte*. [6]

Kada se govori o poslužitelju za komunikaciju putem protokola MQTT i uspostavljanjem veze sa istim, valja napomenuti kako se poslužitelje naziva brokerima, a komunikacija između korisnika i poslužitelja se naziva klijent-broker komunikacija. Njihova komunikacija započinje od korisnika koji šalje poruku sa zahtjevom povezivanja poslužitelju (brokeru) te ovisno o uspješnosti uspostave veze, korisnik dobiva povratnu informaciju (*connack*). [7]



Slika 12. *Komunikacija klijent-broker prilikom uspostavljanja veze*

Nakon što korisnik pošalje pomoću *Connect* metode poruku zahtjeva za povezivanjem s brokerom, ovaj korisniku ili klijentu šalje povratnu informaciju, a prije smo mogli vidjeti kako se povratna informacija pohranjuje u varijabli tipa *byte*, a razlog tomu je veoma jednostavan. Naime povratna informacija se sastoji od brojčane znamenke koja se kreće u rasponu od 0 pa do 5, a jedan bajt se sastoji od 8 bitova, stoga je savršeno za korištenje varijablu tipa *byte* jer ne zauzima suvišnu memoriju, a u potpunosti je dovoljna za svrhu koju služi.

Return Code	Return Code Response
0	Connection accepted
1	Connection refused, unacceptable protocol version
2	Connection refused, identifier rejected
3	Connection refused, server unavailable
4	Connection refused, bad user name or password
5	Connection refused, not authorized

Slika 13. *Sve moguće povratne informacije zaprimljene od brokera*

4.4. PRETPLAĆIVANJE NA TEMATSKU SKUPINU ZA .NET INAČICU

Nakon što korisnik uspješno uspostavi vezu s poslužiteljem, odnosno brokerom, korisniku su na raspolaganju sve ostale funkcionalnosti kao što je primanje poruka, slanje poruka i pretplaćivanje na tematske skupine. Primanje poruka nakon samog uspostavljanja veze nije još uvijek moguće dok se korisnik ne pretplati na neku temu, stoga ćemo prvo razmotriti kako je ostvarena funkcionalnost pretplaćivanja.

```
private void btnPretplati_Click(object sender, EventArgs e)
{
    string tema = "zavrsni_rad/" + inputTema.Text.ToString() + "/test";

    pretplata = this.korisnik.dodajTemu(tema);
    if(pretplata == 1)
    {
        Client.Subscribe(new string[] { tema }, new byte[] { 2 });
        pretplacenaTema();
        chat();
    }
    else
    {
        MessageBox.Show("Već ste pretplaćeni na tu temu!");
        chat();
    }
}

public void pretplacenaTema()
{
    outputPretplate.Clear();
    foreach(var item in this.korisnik.vratiPopis())
    {
        outputPretplate.AppendText(item + "\n");
    }
    outputPretplate.Update();
}
```

Slika 13. *Pretplaćivanje na tematsku skupinu .NET inačice aplikacije*

Sama metoda koja se koristi za pretplaćivanje na temu je zapravo voditelj događaja (*event handler*) za pritisak na dugme pretplaćivanja na tematsku skupinu. Dosad se moglo uočiti kako se prilikom pretplaćivanja na tematske skupine uvijek na naziv teme koju korisnik želi nadodao prefiks „zavrsni_rad/“ i sufiks „/test“ zbog sigurnosnih razloga kako ne bi došlo do neželjenih sudionika u komunikaciji, odnosno kako ne bi netko „slučajno“ se pretplatio na tematsku skupinu koju koristimo za svrhe testiranja rada aplikacije. Uz to, valja nadodati kako se prilikom gašenja aplikacije gubi veza klijenta sa brokerom, a ujedno i sve tematske skupine na koje je klijent bio pretplaćen te zbog jednostavnosti se podaci ne pohranjuju u neku bazu podataka, nego se pohranjuju unutar aplikacije pomoću klase Korisnik. Pomoću klase Korisnik se pohranjuju podaci poput *Id*-a korisnika, korisničkog imena, lozinke, popisa svih tematskih

skupina na koje je korisnik pretplaćen i svih poruka koje je objavio. Uz te podatke, klasa Korisnik sadrži metode za dodavanje nove tematske skupine na popis prilikom pretplaćivanja, dodavanje nove poruke prilikom objavljivanja, vraćanje popisa svih tematskih skupina, prebrojavanje istih, itd. Sa slike broj 13 je uočljivo kako se prvo poziva metoda za dodavanje nove teme pomoću objekta korisnika te ukoliko korisnik nije pretplaćen na tu tematsku skupinu i metoda vrati zadovoljavajuću povratnu informaciju, onda se pomoću objekta klijenta MQTT poziva metoda *Subscribe.* Nakon pretplaćivanja na tematsku skupinu se poziva metoda koja je ispod prikazana, a služi za dodavanje nove tematske skupine i ažuriranje prikaza svih pretplaćenih tema unutar okvira sa popisom tema.

4.5. ODJAVLJIVANJE PRETPLATE ZA .NET INAČICU

Veoma slično kao i kod pretplaćivanja na tematsku skupinu, isto tako se vrši i odjava tematske skupine. Korisnik odabire sa okvira sa popisom pretplaćenih tematskih skupina onu koju želi odjaviti te pritiskom na dugme se poziva metoda za odjavu pretplate.

```
public void odjavljeneTeme()
{
    outputPretplate.Clear();
}

private void btnOdjava_Click(object sender, EventArgs e)
{
    string tema = "zavrsni_rad/" + inputTema.Text.ToString() + "/test";

    pretplata = korisnik.makniTemu(tema);
    if (pretplata == 1)
    {
        Client.Unsubscribe(new string[] { tema });
        odjavljeneTeme();
        pretplacenaTema();
        chat();
    }
    else
    {
        MessageBox.Show("Nema postojeće teme!");
        chat();
    }
}
```

Slika 14. *Odjavljivanje tematske skupine u .NET inačici*

Kao i kod pretplate na tematsku skupinu, sve se događa pritiskom na dugme, zatim se preko klase Korisnik i metode za odjavu teme odabrana tema izbacuje sa popisa pretplaćenih tema. Ukoliko je uspješno odstranjena tema sa popisa pretplaćenih tema, onda se preko objekta klijenta MQTT poziva metoda *Unsubscribe* i sve ostale metode za osvježavanje prikazanih tematskih skupina.

4.6. OBJAVLJIVANJE PORUKA S POMOĆU .NET INAČICE

Razmotrene su sve funkcionalnosti, osim dvije glavne kojima se postiže glavna svrha ove aplikacije i ovoga rada. Te dvije glavne funkcionalnosti su objavljivanje i zaprimanje tekstualnih poruka jer ipak je glavna svrha ove aplikacije uspješna razmjena tekstualnih poruka.

Tako dolazimo do funkcionalnosti za objavljivanje tekstualnih poruka unutar tematskih skupina. Putem slika je bilo uočljivo kako je ostvarena ta funkcionalnost preko sučelja, a to je tako da korisnik upisuje naziv tematske skupine na koju želi objaviti poruku te nakon toga upisuje sam sadržaj poruke koju namjerava objaviti.

```
private void btnObjavi_Click(object sender, EventArgs e)
{
    //Provjera je li unesena poruka ili tema
    if(inputPoruka.Text == "" || inputTemaObjavi.Text == "")
    {
        if(inputPoruka.Text == "")
        {
            MessageBox.Show("Ne možete poslati praznu poruku!");
        }
        if(inputTemaObjavi.Text == "")
        {
            MessageBox.Show("Ne možete objaviti na praznu temu!");
        }
    }
    else
    {
        //Tema na koju objavljujemo poruku
        string tema = "zavrsni_rad/" + inputTemaObjavi.Text.ToString() + "/test";

        Client.MqttMsgPublishReceived += client_MqttMsgPublishReceived;

        //Objava poruke sa QoS 2
        Client.Publish(tema, Encoding.UTF8.GetBytes(inputPoruka.Text.ToString()), MqttMsgBase.QOS_LEVEL_EXACTLY_ONCE, true);
    }
}
```

Slika 15. *Objavljivanje tekstualne poruke u .NET inačici*

Kao i dosad, zbog jednostavnosti aplikacije, sve se vrši preko voditelja događaja, (eng. *event handler*) koji se pokreće prilikom pritiska na dugme za objavljivanje poruke. Dohvaćaju se tekstovi iz tekstualnih okvira za naziv tematske skupine i za sadržaj poruke te se provjerava jesu li upisane potrebne informacije. Ukoliko je sve uneseno ispravno, nazivu tematske skupine se dodaju prefiks i sufiks vezan uz ovaj rad zbog sigurnosnih razloga. Zatim se pomoću objekta klijenta MQTT poziva metoda za objavljivanje, odnosno *Publish* metoda koja poprima nekoliko parametara. Prvi parametar je sam naziv tematske skupine na koju se nastoji poslati poruka, drugi je sadržaj poruke, treći je razina kvalitete usluge koja je u ranijim poglavljima bila objašnjena, no za svrhu ove aplikacije je unaprijed određena na samo jedno objavljivanje poruke.

4.7. ZAPRIMANJE PORUKA POMOĆU .NET INAČICE

Kada govorimo o *desktop*, odnosno .NET inačici aplikacije, ovo će biti posljednja funkcionalnost koje ćemo se dotaknuti, a to je zaprimanje tekstualnih poruka unutar tematskih skupina. Jedina funkcionalnost koja nije ostvarena pomoću pritiska na dugme, nego ima drugačiju vrstu voditelja događaja tj. *event handlera*, a on se poziva prilikom zaprimanja poruke.

```
private void client_MqttMsgPublishReceived(object sender, MqttMsgPublishEventArgs e)
{
    string zaprimljenaPoruka = Encoding.UTF8.GetString(e.Message);

    //Debug.WriteLine("Received = " + Encoding.UTF8.GetString(e.Message) + " on topic " + e.Topic);

    output = "";

    if(this.korisnik.vratiBrojPoruka() != 0)
    {
        if (this.korisnik.posljednjaPoruka() != zaprimljenaPoruka.ToString())
        {
            this.korisnik.novaPoruka(zaprimljenaPoruka);
            foreach (var elem in this.korisnik.vratiPoruke())
            {
                output += elem + "\n";
            }
            SetText(output.ToString());
        }
    }
    else
    {
        this.korisnik.novaPoruka(zaprimljenaPoruka);
        foreach (var elem in this.korisnik.vratiPoruke())
        {
            output += elem + "\n";
        }
        SetText(output.ToString());
    }
}
```

Slika 16. *Zaprimanje tekstualnih poruka unutar tematskih skupina za .NET inačicu*

Ovo je ujedno i jedina funkcionalnost kod koje je došlo do sitnih komplikacija. Naime ukoliko je korisnik pretplaćen na tematsku skupinu u koju objavljuje poruku, iz nepoznatih razloga dolazi do slučajeva kada bi se ista ta poruka zaprimila više puta nego jednom. Stoga je unutar klase Korisnik napravljena metoda koja prilikom zaprimanja poruke provjerava objavljene poruke te ukoliko se posljednja objavljena poruka podudara sa već zaprimljenom porukom, nju se onda preskače kako se ne bi prikazivala više puta u području predviđenom za sve zaprimljene poruke.

4.8. USPOSTAVLJANJE VEZE S POSLUŽITELJEM WEB APLIKACIJE

Nakon što je detaljnije razmotren programski kod za aplikativno rješenje *desktop*, odnosno .NET inačice ove aplikacije jer je ponešto kompleksnije od inačice za web preglednike, ali svejedno valja pogledati kako se iste funkcionalnosti prenose iz C# programskog jezika u Javascript.

Kao i za .NET inačicu, započinje se sa uspostavljanjem veze klijenta s poslužiteljem, odnosno brokerom. Korisnik prilikom pokretanja web aplikacije u web pregledniku dobiva cijelu aplikaciju u jednom prozoru, ali funkcionalnosti nisu omogućene sve dok ne uspostavi vezu s brokerom, a kako bi to učinio, mora upisati svoje ime po kojem će biti prepoznatljiv tokom razmjene poruka.

```
// Kreira se instanca klijenta koja je zapravo NULL
client = null;
connected = false;

// Funkcija koja se poziva pritiskom na dugme za povezivanje
function connect(){
    var hostname = 'broker.mqttdashboard.com'; //Unaprijed određeni poslužitelj (broker)
    var port = '8000'; //port
    var clientId = document.getElementById("ime").value; //Dohvaćanje imena korisnika

    client = new Paho.MQTT.Client(hostname, Number(port), clientId); //Kreiranje instance MQTT klijenta

    console.info('Connecting to Server: Hostname: ', hostname, '. Port: ', port, '. Client ID: ', clientId);

    // Postavljanje metoda koje će se koristiti kao event handleri za određene događaje
    client.onConnectionLost = onConnectionLost;
    client.onMessageArrived = onMessageArrived;

    // Parametri koje funkcija za uspostavljanje veze poprima spremljeni u jednu varijablu
    var options = {
      invocationContext: {host : hostname, port: port, clientId: clientId},
      onSuccess: onConnect,
      onFailure: onFail
    };

    // Povezivanje klijenta
    client.connect(options);

}
```

Slika 17. *Povezivanje klijenta s poslužiteljem web aplikacije*

Veoma slično *desktop* inačici, tako se i ovdje prilikom pritiska na dugme za povezivanje unutar web aplikacije preko *onClick* događaja poziva Javascript metoda za povezivanje koja je vidljiva na slici iznad. Za razliku od .NET inačice, ovdje objekt klijenta MQTT poprima i parametar port-a za povezivanje s poslužiteljem tj. brokerom. Uz to, moraju se unaprijed odrediti metode odnosno upravitelji događaja za određene događaje koji se mogu dogoditi prilikom povezivanja, kao što su to gubitak veze, uspješno povezivanje ili čak neuspješno povezivanje. Svi ti parametri se stave u jednu varijablu koju se onda na kraju proslijedi metodi *connect*.

4.9. PRETPLAĆIVANJE NA TEMATSKU SKUPINU ZA WEB APLIKACIJU

Jednako kao i kod .NET inačice aplikacije, pomoću web aplikacije korisnik se može pretplatiti na više tematskih skupina, ali postoji jedna razlika, a to je da nema popisa tema na koje se korisnik pretplatio. Stoga se prilikom svake pretplate, u dijelu predviđenom za poruke, prikaže poruka o korisniku i tematskoj skupini na koju se pretplaćuje.

Kako bi se korisnik uspješno pretplatio, prvo mora upisati korisničko ime koje želi koristiti te povezati se s poslužiteljem, a nakon toga upisuje naziv tematske skupine na koju se pretplaćuje te pritiskom na dugme za pretplatu se poziva *subscribe* metoda pomoću koje se to onda i ostvaruje.

```
// Metoda za pretplaćivanje na tematsku skupinu
function subscribe(){
    var topic = document.getElementById("tema").value; //dohvaćanje naziva teme
    var topic1 = 'zavrsni_rad/' + topic + '/test'; //dodavanje prefiksa i sufiksa
    var qos = 2; //određivanje qos
    console.info('Subscribing to: Topic: ', topic1, '. QoS: ', qos);
    var poruka = document.createElement('span');
    var korisnik = document.getElementById("ime").value; //dohvaćanje imena korisnika
    poruka.innerHTML = korisnik + ' se pretplatio na ' + topic1 + '</span><br/>';
    var messages = document.getElementById("messages");
    messages.appendChild(poruka); //prikazivanje poruke o pretplaćivanju
    client.subscribe(topic1, {qos: Number(qos)}); //pretplaćivanje na tematsku skupinu
}
```

Slika 18. *Pretplaćivanje na tematsku skupinu za web aplikaciju*

4.10. OBJAVLJIVANJE PORUKA S POMOĆU WEB APLIKACIJE

```
// Metoda za objavljivanje poruke
function publish(){
    var topic = document.getElementById("tema").value; //dohvaćanje naziva teme
    var topic1 = 'zavrsni_rad/' + topic + '/test'; //dodavanje prefiksa i sufiksa
    var korisnik = document.getElementById("ime").value; //dohvaćanje imena korisnika
    var qos = 2; //razina qos
    var message = document.getElementById("ulaz").value; //sadržaj poruke
    var poruka = korisnik + ':  ' + message; //poruka pripremljena za objavljivanje
    console.info('Publishing Message: Topic: ', topic1, '. QoS: ' + qos + '. Message: ', poruka);
    message = new Paho.MQTT.Message(poruka); //mqtt objekt poruke
    message.destinationName = topic1; //objektu dodana destinacija
    message.qos = Number(qos); //objektu određen qos
    client.send(message); //objavljivanje poruke
}
```

Slika 19. *Objavljivanje tekstualnih poruka unutar tematskih skupina web aplikacije*

Objavljivanje sadržaja tekstualnih poruka na određene tematske skupine je veoma intuitivno i jednostavno, nakon što korisnik upiše naziv tematske skupine na koju želi objaviti poruku, upisuje sadržaj poruke te pritiskom na dugme za slanje se poziva metoda sa slike iznad.

Prvo se dohvaća tema na koju korisnik objavljuje poruku te joj se dodaju prefiks i sufiks kao i kod *desktop* inačice. Nakon toga se dohvaća naziv korisnika i formira se poruka MQTT sa sadržajem, određuje joj se destinacija, odnosno tematska skupina i naposljetku se pomoću *send* metode šalje poruka poslužitelju tj. brokeru koji nakon toga prosljeđuje sadržaj te poruke svim korisnicima koji su unutar te tematske skupine.

4.11. ZAPRIMANJE PORUKA UNUTAR TEMATSKIH SKUPINA WEB APLIKACIJE

Jednako kao i kod *desktop* inačice, pri samome kraju će biti detaljnije razmotrena funkcionalnost zaprimanja tekstualnih poruka unutar tematskih skupina kod inačice za web preglednike.

Korisnik prvenstveno mora biti pretplaćen na barem jednu temu kako bi mogao zaprimati poruke, a nakon što se pretplati na neku određenu temu, prilikom svakog zaprimanja poruka se poziva voditelj događaja (*event handler*) za zaprimanje poruka. Kako se u inačici za web preglednik mora prikazati poruka korisniku prilikom zaprimanja, mora se kreirati HTML element. Tom elementu onda se dodaje tekst poput naziva tematske skupine, imena korisnika koji objavljuje poruku te sam sadržaj (*payload*) poruke. Da bi taj element bio prikazan u dijelu predviđenom za poruke, dohvaća se HTML element predviđen za prikazivanje poruka te mu se na kraj dodaje (*appenda*) element koji sadrži tekstualnu poruku.

```
// Event handler za pristiglu poruku
function onMessageArrived(message) {
  console.log('Message Recieved: Topic: ', message.destinationName, '. Payload: ', message.payloadString, '. QoS:
  console.log(message);
  var messageTime = new Date().toISOString(); //timestamp primitka pristigle poruke
  var poruka = document.createElement('span'); //kreiranje html elementa sa sadržajem poruke
  poruka.innerHTML = 'Tema: ' + message.destinationName + '  |  ' + message.payloadString + '</span><br/>';
  var messages = document.getElementById("messages"); //dohvaćanje html elementa za prikazivanje poruka
  messages.appendChild(poruka); //dodavanje nove poruke html elementu predviđenom za prikaz pristiglih poruka
}
```

Slika 20. *Zaprimanje tekstualnih poruka za web aplikaciju*

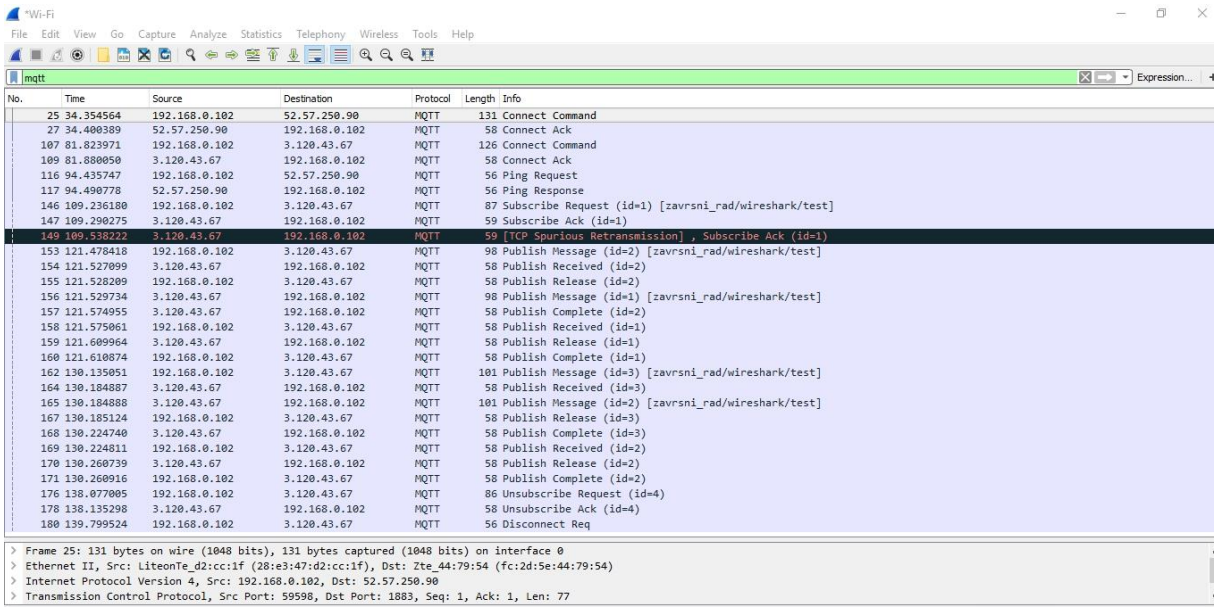
Već ranije je spomenuto kako je inačica za web preglednike ponešto jednostavnija i lakše ostvarena u usporedbi s .NET inačicom, tako bi valjalo i napomenuti da nije došlo do nikakvih problema ili sličnih komplikacija, pogotovo za funkcionalnost zaprimanja poruka kao što je to bio slučaj kod *desktop* inačice aplikacije.

5. WIRESHARK PROMET

U ranijim dijelovima je već ustanovljeno kako kod mrežnog programiranja jedna od glavnih značajki je komunikacija koja se svodi na razmjenu, odnosno slanje i primanje paketa s informacijama putem internetske mreže. Sukladno tome možemo reći kako je od iznimne važnosti, kod aplikacija za razmjenu podataka putem interneta, provjera s kim one komuniciraju, kakve pakete šalju, odnosno primaju te naravno koje su veličine ti paketi.

Za potrebu toga se koriste alati za prisluškivanje mrežne komunikacije, kojih ima podosta, a ujedno su i veoma rasprostranjeni te dostupni uglavnom svima. Zbog već ranijim susretanjem sa istim alatom, za ovu svrhu je odabran alat Wireshark.

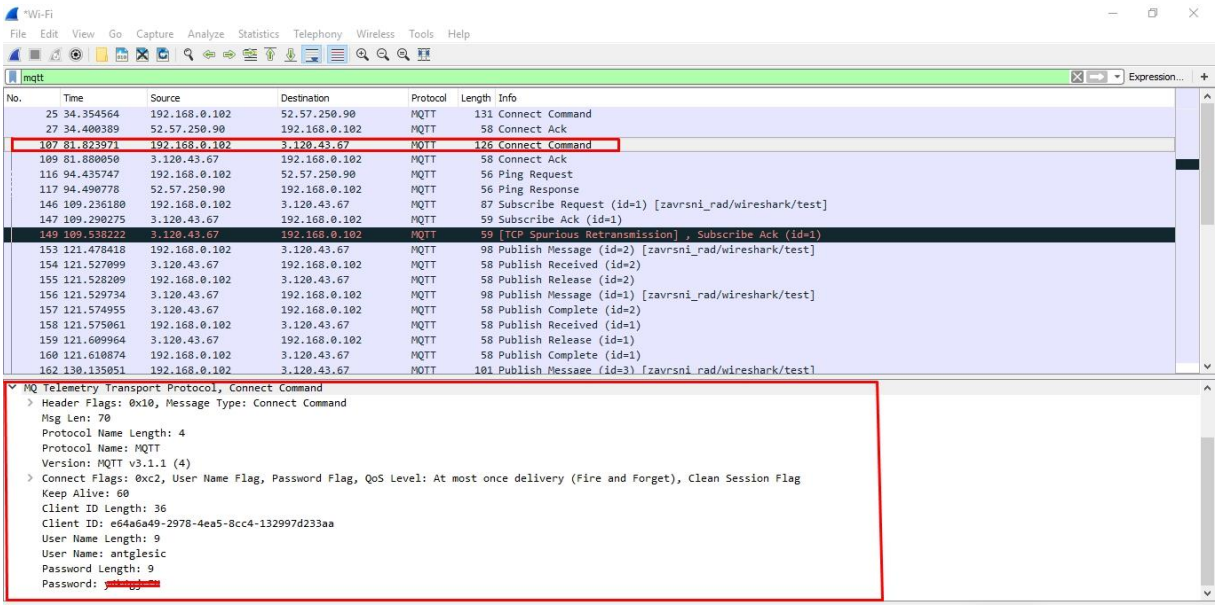
Samo sučelje alata Wireshark je veoma intuitivno, stoga nema potrebe za objašnjenjem istoga. Valja samo spomenuti kako prilikom osluškivanja mrežnog prometa se svaki zaprimljeni i/ili poslani paket prikazuje u tabličnom prikazu sa najvažnijim informacijama prema kojima se ujedno paketi mogu i sortirati. Podaci od značaja u ovom slučaju će nam biti izvor (*Source*) dolaska paketa, destinacija (*Destination*), kako aplikacija koristi protokol MQTT, valja napomenuti da se u alatu Wireshark mogu filtrirati paketi prema protokolu koji je korišten te se na priloženoj slici može vidjeti kako su prikazani samo paketi protokola MQTT. Kao još jednu od važnih informacija o paketu valja promatrati dio sa osnovnim informacijama o paketu (*Info*) gdje se može pročitati svrha paketa koju on služi. [9]



Slika 21. *Slika mrežnog prometa snimljena Wireshark alatom*

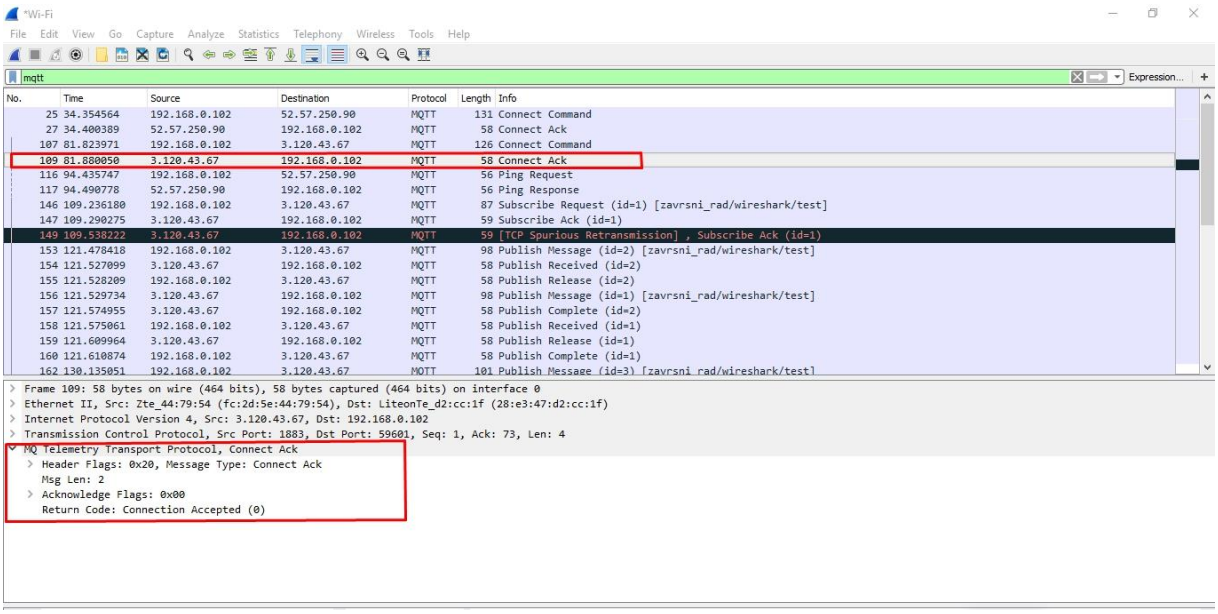
5.1. ZAHTJEV ZA POVEZIVANJE

Prilikom korištenja aplikacije za razmjenu tekstualnih poruka unutar tematskih skupina se šalju i primaju razni paketi. Zbog relativne sličnosti i važnosti bit će samo prikazani neki paketi poput zahtjeva za povezivanje i paketa prilikom objavljivanja (*publish*) poruke na neku temu, odnosno tematsku skupinu.



Slika 22. *Slanje paketa za zahtjev povezivanja s poslužiteljem*

Sa slike broj 22 prvo valja uočiti prilikom slanja paketa sa zahtjevom za povezivanje kome se šalje paket. To je lako uočljivo iz dijela sa IP adresama, gdje kao izvorišnu adresu korisnik vidi svoju trenutnu IP adresu, a kao adresu destinacije vidi IP adresu poslužitelja s kojim se nastoji povezati. U ovom slučaju adresa „*broker.hivemq.com*“ izgleda ovako „3.120.43.67“, a informacija paketa se navodi „*Connect Command*“, što označava kako korisnik nastoji uspostaviti vezu sa poslužiteljem kojemu se na prethodno navedenu IP adresu šalje zahtjev sa određenim podacima. Paketi u mrežnoj komunikaciji se učahure putem raznih slojeva te svaki sloj ima svojevrsno zaglavlje sa određenim njemu potrebnim podacima, a protokol MQTT nije iznimka tome. Od svih paketa protokola MQTT, može se uočiti kako je paket sa zahtjevom za povezivanje najveći paket po sadržaju. Kao neke od podataka koji se šalju poslužitelju prilikom pokušaja povezivanja vrijedi spomenuti da se šalje naziv korištenog protokola, njegova verzija, razne zastavice za povezivanje, vrijeme koliko paket ostaje relevantan, oznaka korisnika, njegovo korisničko ime i lozinka, itd.

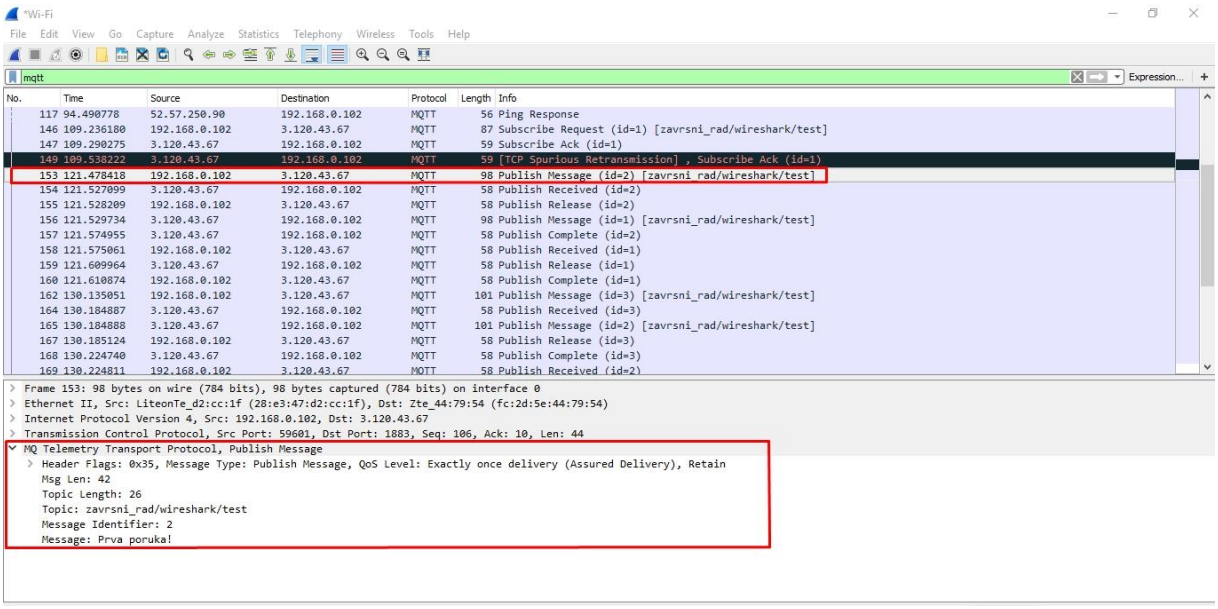


Slika 23. *Zaprimanje paketa sa potvrdom o uspješnom povezivanju*

Sukladno već ranije napomenutim vrstama poruka koje korisnik dobiva kao povratnu informaciju od poslužitelja, odnosno brokera prilikom pokušaja uspostavljanja veze sa istim, tako se može primijetiti da nakon slanja paketa sa zahtjevom za povezivanje, korisnik zaprima paket „*Connect Ack*“ od poslužitelja pomoću kojega dobiva povratnu informaciju o uspješnosti uspostavljanja veze. U slučaju na slici broj 23 se može vidjeti kako je povratna informacija nula (0), što znači da je bez greške uspostavljena veza između korisnika i poslužitelja te se može nastaviti komunikacija istih.

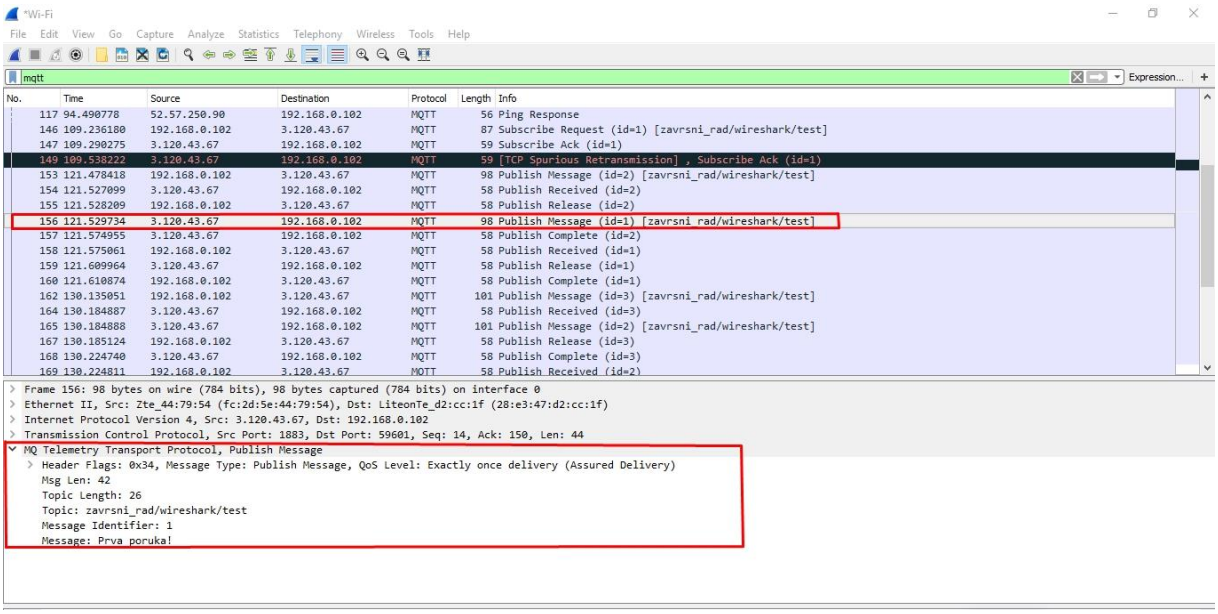
5.2. SLANJE I ZAPRIMANJE PORUKE

Nakon što korisnik uspješno uspostavi vezu sa poslužiteljem, odnosno brokerom, može nastaviti sa komunikacijom. Naravno glavna svrha aplikacije je slanje i/ili primanje poruka, a svaka poruka se ne šalje direktno korisnicima, nego poslužitelju koji zatim prosljeđuje korisnicima koji su pretplaćeni na tu tematsku skupinu poruku.



Slika 24. *Slanje paketa prilikom objave poruke*

Slično kao i kod slanja paketa sa zahtjevom za povezivanje, korisnik ukoliko želi objaviti poruku na tematsku skupinu, poruku šalje prvo poslužitelju unutar određenog paketa. IP adresa je jednaka kao i kod zahtjeva za povezivanje, znači da je komunikacija sa istim poslužiteljem, ali veličina paketa i njegova glavna informacija je drugačija. U dijelu sa informacijama o paketu se može vidjeti kako se radi o paketu za objavljivanje poruke koji ima svoju određenu oznaku te se u zagradi navodi čak i pun naziv tematske skupine na koju korisnik želi objaviti poruku. Putem paketa se šalju potrebni podaci kao što su zastavice zaglavlja, podaci o poruci kao što je njena duljina i sam sadržaj poruke te naziv tematske skupine, itd.



Slika 25. *Zaprimanje paketa o zaprimljenoj poruci*

Zaprimanje poruke je ponešto drugačije nego očekivano jer poslužitelj nakon što mu korisnik pošalje poruku prosljeđuje istu svim korisnicima koji su pretplaćeni na tu tematsku skupinu. Može se reći kako poslužitelj isto kao korisnik objavljuje poruku, stoga korisnik prilikom zaprimanja poruke dobiva paket o tome kako poslužitelj objavljuje poruku na određenu temu.

Kako ne bi bilo nesporazuma o tome, vrijedi spomenuti kako nakon objavljivanja poruke, korisnik zaprima paket od poslužitelja kao povratnu informaciju da je poslužitelj zaprimio objavljenu poruku. Uz to, paketi vezani za određene poruke imaju svoje identifikatore, odnosno oznake kako ne bi došlo do zabune, no paket prilikom zaprimanja poruke je veoma sličan paketu za objavljivanje poruke.

6. ZAKLJUČAK

Kako se načini komunikacije konstantno razvijaju od davnina, a u posljednjem stoljeću je sve brži napredak u području telekomunikacije. Sve je veća potražnja za što bržim načinom prijenosa poruke od točke A do točke B bez gubitka sadržaja poruke.

Danas je najpoznatiji način komunikacije pomoću aplikacija koje prenose poruke u realnom vremenu, kao što su to *Facebook Messenger*, *Whatsapp Messenger*, *Viber* i mnoge druge aplikacije, odakle je i došla ideja za temom ovoga rada, odnosno aplikacijom za razmjenu tekstualnih poruka.

Odabran je protokol MQTT zbog svoje lake primjenjivosti i malih zahtjeva. On se koristi za razne sitne uređaje i naprave kao što su senzori i slično, gdje nastoje prenositi podatke prema tematskim skupinama tako da im mogu razni drugi uređaji i korisnici pristupiti, a ne opterećuje bespotrebno uređaj koji šalje podatke jer samo jednom šalje sadržaj poruke poslužitelju i ne mora slati podatke svakom zasebnom korisniku.

Protokol MQTT je relativno jednostavan za shvatiti i koristiti, ali ta jednostavnost može biti i svojevrstan nedostatak jer kako korisnici se pretplaćuju na tematske skupine, ne postoji neki određen način zaštite prijenosa podataka od neželjenih sudionika, odnosno ukoliko netko uspije „pogoditi“ naziv tematske skupine na koju objavljujemo, može prihvaćati poruke koje nisu nužno namijenjene tom korisniku.

Stoga možemo zaključiti kako u suvremenu svijetu veliku ulogu igraju aplikacije za brzu razmjenu tekstualnih poruka između korisnika, a protokol MQTT upravo to omogućuje korisnicima, no uvijek ima prostora za napredak pa tako i kod protokola MQTT.

7. LITERATURA

[1] Povijest razvoja komunikacijske tehnologije (2017), https://www.conferencecallsunlimited.com/history-of-communication-technology/ dostupno (14.07.2018.)

[2] Što je to socket (2018), https://www.tutorialspoint.com/unix_sockets/what_is_socket.htm dostupno (15.07.2018.)

[3] Protokol MQTT (2015), http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html dostupno (15.07.2018.)

[4] IRC (2013), https://hr.wikipedia.org/wiki/IRC dostupno (19.07.2018.)

[5] NuGet Gallery | M2Mqtt (2015), https://www.nuget.org/packages/M2Mqtt/ dostupno (24.07.2018.)

[6] Using MqttClient (2015), https://m2mqtt.wordpress.com/using-mqttclient/ dostupno (25.07.2018.)

[7] MQTT Essentials: Client, Broker (2018), https://www.hivemq.com/blog/mqtt-essentials-part-3-client-broker-connection-establishment dostupno (25.07.2018.)

[8] MQTT Websocket Client, http://www.hivemq.com/demos/websocket-client/ dostupno (26.07.2018.)

[9] Wireshark MQTT display filter (2015), https://www.wireshark.org/docs/dfref/m/mqtt.html dostupno (28.08.2018.)