**SVEUČILIŠTE U ZAGREBU, FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA**

36. Chat

**WEB PRIKLJUČNICE I SIGURNA KOMUNIKACIJA**

Matija Šekrst

**Zagreb, prosinac 2016.**

# Uvod

Tema ovog izvješća je su Web priključnice (engl. *Web sockets*), sigurna komunikacija između više priključnica, te mogućnosti ostvarenja takve komunikacije korištenjem *Node.js* platforme.

*Node.js* je platforma koja omogućuje izradu skripti koje se izvode na poslužitelju. Sigurna komunikacija odnosi se na nemogućnost nekog napadača da sazna sadržaj poruka koje razmjenjuju klijent i poslužitelj. Pošto je cilj projekta izrada aplikacije koja će za komunikaciju koristiti gotovo Web priključnice treba se obratiti posebna pažnja sigurnosti web priključnica.

U prvom dijelu bit će objašnjena tehnologija Web priključnica, u drugom dijelu tema je sigurna komunikacija. Treći dio bavi se implementacijom sigurne komunikacija za Node.js platformu.

# Web priključnice

Web priključnice su tehnologija koja omogućuje interaktivno slanje poruka s klijenta na poslužitelj. Interaktivno ovdje znači da je s klijentske strane moguće poslati poruku na poslužitelj korištenjem priključnica, te nastaviti neki drugi posao. Isto tako je moguće da poslužitelj primi poruku, obradi je i da po potrebi pošalje odgovor klijentu koji onda taj odgovor obrađuje. Tehnologija web priključnica je dakle vid asinkrone komunikacije jer klijent ne čeka odgovor nego nakon slanja poruke nastavlja svoj posao. Važno je napomenuti da tehnologija Web priključnica omogućuje i da poslužitelj obavještava klijenta o promjenama. Upravo će se ova ideja upotrijebiti u izradi projekta. Naime klijent će se registrirati kod poslužitelja poruka koji će primati poruke, pohranjivati ih u bazu podataka, te obavještavati klijenta kojemu je namijenjena poruka o novoj poruci. Druga važna napomena je da će se cijela komunikacija odvijati u stvarnom vremenu (engl. *real time*), što znači da će klijent primati poruke i prikazivati ih bez potrebe za osvježavanjem stranice (engl. *refresh*) što donosi manji mrežni promet, ali i ugodnije korištenje korisnicima[[1]](#footnote-1).

Web priključnice se mogu izvesti korištenjem oba transportna protokola: UDP[[2]](#footnote-2) i TCP[[3]](#footnote-3). Korištenje oba protokola ima svojih prednosti i mana pa treba dobro procijeniti zahtjeve svakog projekta i na kraju donijeti ispravnu odluku u vezi odabira protokola.

TCP garantira isporuku poruka, tj. kada se nešto pošalje korištenjem TCP protokola garantira se da će poruka doći na odredište. To se postiže tako da se prvo uspostavlja veza između pošiljatelja i primatelja. Stoga se lako zaključuje da je prednost TCP-a sigurnost isporuke, a nedostatak u odnosu na TCP je potreba za otvaranjem veze i posljedično nešto sporija brzina rada, jer se više opterećuje mreža između primatelja i pošiljatelja slanjem dodatnih zahtjeva.

UDP pak ne garantira isporuku poruka. Kod UDP-a poruke se samo šalju mrežom i ovisi o mreži ako će one doći do primatelja. Ako se neka poruka izgubi pošiljatelj ne može znati da ta poruka nije došla do primatelja. Iz ovoga slijedi da je prednost UDP-a brzina slanja, a mana svakako ne garantirana isporuka. Naravno korištenjem UDP-a se može programski postići da se garantira isporuka, tako da pošiljatelj šalje poruke dok mu primatelj ne vrati potvrdu da je poruku primio te se tek tada šalje iduća poruka.

Prilikom izrade projekta koristit će se TCP komunikacijski protokol jer poruke koje razmjenjuju korisnici moraju garantirano pristići na svoje odredište.

# Sigurna komunikacija

Pojam sigurne komunikacije odnosi se na nemogućnost neke treće strane da sazna o čemu komuniciraju neki entiteti korištenjem Interneta. Laički rečeno to je komunikacija koja ima za zahtjev da se ne može prisluškivati od strane nekog tko u komunikaciji ne sudjeluje. Na Internetu se sigurna komunikacija najčešće postiže enkripcijom prometa između strana koje komuniciraju. Enkripcija znači da se svaka poruka prije slanja šifrira koristeći specijalan algoritam za kriptiranje[[4]](#footnote-4) koji iz originalne poruke i nekog ključa kreira šifriranu poruku. Nakon što primatelj primi takvu poruku on mora moći dekriptirati odnosno dešifrirati primljenu poruku. Osnovni problem kriptiranja je dakle mehanizam kojim samo pošiljatelj i primatelj mogu saznati originalnu poruku, dok ona mora biti nečitljiva drugim, tj. razmjena ključeva kojima se kriptira. Naime zahtjev kriptiranja je da se originalna poruka ne može dešifrirati iz šifrirane poruke niti ako se zna algoritam kojim se šifrira.

Postoje dvije vrste kriptografije: simetrična i asimetrična.

Simetrična kriptografija bazira se na jednom ključu koji imaju i primatelj i pošiljatelj. Simetrična je kriptografija brža i do 1000 puta od asimetrične, ali ima problem kako distribuirati zajednički ključ bez da se njegova tajnost prekrši.

Asimetrična kriptografija bazira se na arhitekturi dva ključa: javnog i privatnog. Privatni ključ posjeduje samo jedna osoba, vlasnik ključa, dok javni ključ mora biti javno dostupan svima. To se postiže najčešće korištenjem poslužitelja javnih ključeva. Kao što je već napomenuto mana asinkrone kriptografije je brzina, ali prednost je lagana razmjena ključeva.

Danas se sigurna komunikacija na Internetu osigurava korištenjem HTTPS[[5]](#footnote-5) protokola. HTTPS protokol zahtjeva da se sav promet između klijenta i poslužitelja kriptira korištenjem SSL[[6]](#footnote-6) ili TLS[[7]](#footnote-7) protokola.

Kod korištenja protokola HTTPS na poslužitelju se mora definirati privatni ključ za dešifriranje, te se treba postaviti i certifikat. Certifikat je potvrda kojim poslužitelj potvrđuje da je upravo onaj koji tvrdi da je, tj. potvrđuje svoju autentičnost. Certifikat koji se koristi dobiva se od specijalne agencije koja se bavi certificiranjem. Riječ je o dokumentu koji sadrži adresu poslužitelja, njegov javni ključ i ostale podatke. Certifikatu autentičnost daje potpis koji ostavlja izdavatelj certifikata i koji garantira da je certifikat valjan koristeći postupak digitalnog potpisa. Postupak izdavanja certifikata na za ovaj projekt bit će opisan u idućem poglavlju.

Kada klijent pristupa poslužitelju on preko poslužiteljeve certifikata zna da je poslužitelj autentičan i da mu može vjerovati. Certifikat automatski pregledava klijentski pretraživač Weba (engl. *browser*).

Protokol korištenja web priključnica dodaje dvije nove URI[[8]](#footnote-8) sheme: *ws* i *wss.* Korištenjem ws sheme zapravo se koristi nekriptirana komunikacija između priključnica (npr. ws://posluzitelj.com), dok se navođenjem wss-a zahtjeva kriptirana komunikacija (npr. wss://posluzitelj.com). Kriptiranje se postiže korištenjem TLS protokola, tj. slanjem poruka preko HTTPS-a. Tako da bi se postigla kriptirana komunikacija između Web priključnica treba također posjedovati certifikat koji će omogućiti kriptiranje i autentičnost poslužitelja.

# Implementacija sigurne komunikacije za Node.js platformu

Node.js platforma omogućuje jednostavan razvoj Web poslužitelja. Sigurna komunikacija se korištenjem Node.js platforme najčešće postiže na dva načina. Prvi način je arhitektura koja se sastoji od jednog poslužitelja koji poslužuje klijente putem HTTPS protokola (HTTPS poslužitelj), a druga arhitektura se sastoji od dva poslužitelja koji u paru osiguravaju sigurnu komunikaciju. Prvi koji služi kao *Proxy* za drugi poslužitelj. Zadaća *Proxyja* je da kriptira sav promet između sebe i klijenta. Ovo kriptiranje obuhvaća ulazni i izlazni promet. Drugi poslužitelj ne komunicira s klijentima nego samo s *Proxy* poslužiteljem koji mu prosljeđuje dekriptirane zahtjeve.

Obje arhitekture aplikacije zato zahtijevaju certifikate kako bi mogli podržati HTTPS protokol. Certifikati se mogu kupiti od tvrtke koja se bavi izdavanjem certifikata ili se mogu koristiti certifikati koji su generirani besplatno. Problem kod nekih besplatnih certifikata je kratko vrijeme trajanja što znači da se oni periodički moraju obnavljati. Drugi problem koji prati besplatne certifikate je to da ih ne priznaju svi preglednici pa često prilikom susreta s takvim certifikatom Web preglednici znaju korisnika upozoriti da je možda riječ o prevari, te tako odvratiti korisnika od korištenja željene aplikacije. Ovaj drugi slučaj se događa i ako je certifikat samostalno izdan.

Za ovaj projekt koristit će se certifikat generiran od strane *Let's Ecrypt* certifikacijskog tijela. Riječ je o besplatnom, automatiziranom i otvorenom certifikacijskom tijelu. *Let's Encrypt* omogućuje jednostavno kreiranje certifikata koje prepoznaje većina današnjih Web preglednika (nije podržan na Android operacijskom sustavu ispod inačice 2.3.6 i na Windows XP operacijskom sustavu). Tako generirani certifikati traju 90 dana, ali *Let's Encypt* podržava automatsko produljenje certifikata, koje omogućuje jednostavno obnavljanje certifikata.

Postupak izdavanja certifikata sastoji se od popunjavanja podataka vezanih uz poslužitelj koji se koristi (domena, operacijski sustav poslužitelja i sl.). Drugi dio je dokazivanje kontrole nad domenom. Kontrola se dokazuje korištenjem programa koji koriste ACME[[9]](#footnote-9) protokol. Takvi programi postaju tako klijeti za poslužitelj i šalju specijalne zahtjeve poslužitelju. Protokol definira kakav odgovor mora vratiti poslužitelj za svaki zahtjev koji primi. Ako poslužitelj dobro odgovori na sve zahtjeve (još se nazivaju engl. *challenges*) onda se izdaje važeći certifikat koji traje 90 dana od trenutka izdavanja. Za izdavanje certifikata će se koristiti *certbot[[10]](#footnote-10)* koji lako izdaje *Let's Encypt* certifikate.

Nakon što se generira certifikat može će prionuti razvoju aplikacije koja koristi sigurnu komunikaciju. U ovom projektu sigurna komunikacija će se postići korištenjem arhitekture s *Proxy* poslužiteljem. Razlog je taj da se takvim razvojem dobiva raspodijeljeni sustav koji je bolje rješenje od jednog centraliziranog poslužitelja.

Za izradu web priključnica koristi će se modul *Socket.io* koji je najpoznatiji modul za izradu priključnica korištenjem Node.js platforme. Prednosti modula su u činjenici da je Modul također sadrži podršku za kriptiranu razmjenu poruka (wss), podršku za razmjenu datoteka, kao i podršku za mobilne uređaje s operacijskim sustavima Android i iOS. Druga važna prednost *Socket.io* modula je svakako mogućnost da u slučaju preopterećenja prijeđe na komunikaciju razmjenama poruka kako bi oslobodio dio resursa na poslužitelju.

1. Bolji UX – ***U****ser* ***E****xperience*  [↑](#footnote-ref-1)
2. ***U****ser* ***D****atagram* ***P****rotocol* – nespojni transportni protokol [↑](#footnote-ref-2)
3. ***T****ransmission* ***C****ontrol* ***P****rotocol* – spojni transportni protokol [↑](#footnote-ref-3)
4. Primjerice: *DES, Triple DES, RSA, Blowfish, AES* [↑](#footnote-ref-4)
5. ***H****yper-* ***T****ext****T****ransfer****P****rotocol****S****ecure* – protokol koji signalizira da se koristi kriptirana komunikacija [↑](#footnote-ref-5)
6. ***S****ecure* ***S****ocket* ***L****ayer* – kriptografski protokol [↑](#footnote-ref-6)
7. ***T****ransfer* ***L****ayer* ***S****ecurity* – kriptografski protokol, nasljednik SSL-a [↑](#footnote-ref-7)
8. ***U****niform* ***R****esource* ***I****dentifier –* identifikator resursa [↑](#footnote-ref-8)
9. ***A****utomatic* ***C****ertificate* ***M****anagement* **E**nvironment – protokol korišten za izdavanje certifikata [↑](#footnote-ref-9)
10. Automatiziran klijent koji služi za dohvaćanje i instalaciju certifikata na poslužitelj [↑](#footnote-ref-10)