Višeplatformska aplikacija za korisničko komuniciranje

Turković, Fran

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Fran Turković

Višeplatformska aplikacija za korisničko komuniciranje

ZAVRŠNI RAD

Varaždin, 2021.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Fran Turković

Matični broj: 46221/18–R

Studij: Informacijski sustavi

Višeplatformska aplikacija za korisničko komuniciranje

ZAVRŠNI RAD

Mentor:

Doc. dr. sc. Nikola Ivković

Varaždin, rujan 2021.

Fran Turković

Izjava o izvornosti

Izjavljujem da je moj završni/diplomski rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autor/Autorica potvrdio/potvrdila prihvaćanjem odredbi u sustavu FOI-radovi

Sažetak

U ovom završnom radu prikazan je način izrade višeplatformske mobilne aplikacije korisničko komuniciranje. Mobilna aplikacije je izrađena u okviru za izradu mobilnih aplikacija React Native u programskom jeziku JavaScript. U sustavu se još nalaze dva poslužitelja izrađena u Node.js tehnologiji, jedan koji služi kao REST API i servira sve potrebne podatke za rad aplikacije osim čavrljanja, npr. popis zadnjih razgovora pojedinog korisnika, osnovne informacije korisnika, zahtjeve za kontakte, podatke o grupama itd. On služi za dohvaćanje resursa aplikaciji iz baze podataka. Drugi koji koristi WebSocket tehnologiju i služi za upravljanje razmjenom poruka korisnika implementiran je pomoću biblioteke Socket.IO. Također se u sustavu nalazi i baza podataka MySQL kao primarno skladište podataka sustava. U radu se prvo kreće od specifikacije softverskih zahtjeva (engl. *software requirements specification*, SRS) koja pobliže objašnjava način rada cijelog sustava sa svim njegovim funkcionalnostima. Sljedeće je navedena specifikacija softverskog dizajna (engl. *software design specification*, SDS) u kojoj su još detaljnije objašnjene pojedine kompleksnije funkcionalnosti sustava pomoću raznih dijagrama. Na kraju se nalazi korisnička dokumentacija sa slikama iz aplikacije i objašnjenjima kako pravilno koristiti aplikaciju.

Ključne riječi: React Native, mobilna aplikacija, Socket.IO, softver, chat aplikacija

Sadržaj

1. Uvod

Sustav ovog završnog rada sastoji se od tri glavna dijela: *frontenda* koji je višeplatformska mobilna aplikacija napravljena u React Nativeu koji podržava Android i iOS uređaje, *middlewarea* koji se sastoji od dva poslužitelja Node.js napravljena pomoću Express okvira i od toga jedan služi za dohvaćanje resursa iz baze podataka i napravljen je prema REST arhitekturi, a drugi služi za upravljanje razmjenom poruka u stvarnom vremenu i napravljen je pomoću Socket.IO biblioteke te *backenda* koji je baza podataka MySQL koja služi kao primarno skladište podataka u sustavu.

Node.js je asinkrono JavaScript izvršno okruženje vođeno događajima. Velika prednost ovog okruženja je što nije blokirajuće. To upravo omogućava asinkronost sustava. Sustav se izvršava na jednoj dretvi te koristi petlju događaja koja izvršava sve funkcije koje joj se pošalju. Time se dobiva sustav u kojem se ne mora čekati izvršavanje jedne funkcije da bi se pozvala i krenula izvršavati druga funkcija nego se sve funkcije mogu pozvati u isto vrijeme i šalju se u petlju događaja koja ih izvršava. Hoće li petlja događaja uspjeti i izvršiti sve funkcije u isto vrijeme ovisi o kapacitetu. [1]

Express je okvir za izradu web aplikacija u Node.js okružju. On uvelike pojednostavljuje izradu web aplikacija. Koristi se za programiranje web poslužitelja. [2]

Mobilna aplikacija preko HTTP protokola ima komunikaciju s poslužiteljem za dohvaćanje resursa koji radi na portu 3000 i stalnu konekciju preko WebSocket tehnologije s poslužiteljem za upravljanje razmjenom poruka na portu 3001. U radu je fokus bio direktno na ovom sustavu, a ponešto manje su objašnjene tehnologije koje se koriste i kako one točno rade. Pošto je fokus rada na korisničkoj komunikaciji *frontend* dio je manje objašnjen, a najviše se objašnjava rad cjelokupnog sustava te poslužitelja za upravljanje razmjenom poruka. Stoga je u radu direktno objašnjen sustav implementiran za rad ove aplikacije.

2. Specifikacija softverskih zahtjeva

U ovom dijelu specifikacije detaljnije će se objasniti svrha, ciljana publika i opseg aplikacije. Također je i naveden rječnik s pojašnjenjima određenih važnijih pojmova.

Svrha ove specifikacije je objašnjenje rada višeplatformske aplikacije za korisničko komuniciranje. Ovaj dokument opisuje cijeli plan izrade aplikacije sa svim svojim funkcionalnostima te objašnjenjima pojedinih funkcionalnosti, cjeloviti opis aplikacije, zahtjeve vanjskih sučelja, funkcionalnosti same aplikacije, ostale nefunkcionalne značajke i ostale zahtjeve aplikacije.

Prvenstveno ciljana publika ove aplikacije su mobilni korisnici koji koriste Android i iOS mobilne operacijske sustave. Također, ciljana publika su svi korisnici koji komuniciraju mobilnim uređajima preko internet veze, neovisno o njihovoj dobi, spolu, rasi, radnom mjestu, školovanju itd.

Opseg ovog projekta vrlo je velik i sastoji se od svih današnjih standarda klasičnih aplikacija za korisničko komuniciranje. Sama aplikacija se sastoji od *frontend* dijela, odnosno mobilne višeplatformske aplikacije za Android i iOS, *middleware* dijela koji se sastoji od poslužitelja za čavrljanje Node.js i Node.js REST APIa te *backend* dijela koji je baza podataka MySQL. Sama aplikacija bi se trebala kasnije koristiti u cijelom svijetu, tj. trebala bi biti globalnih razmjera, no za početak fokus će biti na Hrvatskoj zato što će aplikacija u prvoj radnoj verziji podržavati samo hrvatski jezik. Aplikaciju izrađuje jedan *full stack developer*. Sljedeće je prikazan rječnik određenih pojmova u obliku tablice.

POJAM	DEFINICIJA
VIŠEPLATFORMSKA APLIKACIJA	Aplikacija koja radi i može se koristiti na više različitih platformi, npr. Android i iOS
FRONTEND	Odnosi se na grafičko korisničko sučelje, može biti mobilna aplikacija, web stranica itd.
MIDDLEWARE	Program koji omogućava komunikaciju između više različitih aplikacija, npr. program koji omogućuje komunikaciju između aplikacije i baze podataka
BACKEND	Odnosi se na pozadinski dio aplikacije u kojoj se šalju i obrađuju podatci, tj. sloj pristupa podatcima
FULL STACK DEVELOPER	Programer koji odrađuje i *frontend* i *backend* dio programa
SOCKET	Završna točka u dvosmjernoj komunikaciji dva programa u realnom vremenu, ovakav tip konekcije je stalan, tj. jednom kad se konekcija uspostavi ona traje sve dok ju jedan od sudionika ne prekine

Tablica 1. Rječnik (Izvor: Fran Turković, 2021.)

2.1. Opis

U ovom dijelu specifikacije pojašnjena je perspektiva aplikacije, funkcionalnosti, korisničke klase i karakteristike, radno okruženje, ograničenja dizajna i implementacije te pretpostavke i ovisnosti.

2.1.1. Perspektiva i funkcionalnosti

Ovaj projekt je samostalan projekt, tj. nije komponenta nekog većeg projekta. Poslužitelji i baza podataka koja su dio ovog projekta služit će samo za ovaj projekt. Ovaj projekt ne zamjenjuje postojeći projekt, nego je naprotiv nova implementacija aplikacije za korisničku komunikaciju.

Glavna funkcionalnost ovog projekta je komunikacija između dva ili više korisnika aplikacije. Komunikacija se razlaže na više dijelova od kojih je implementirana komunikacija putem tekstualnih poruka, komunikacija putem slika, videozapisa i audio zapisa. Ova aplikacija također ima funkcionalnost grupnog razgovora između korisnika pojedine grupe. Pod ovim funkcionalnostima se misli na slanje i primanje poruka u realnom vremenu, bilo to tekstualne poruke, videozapisi, slike ili audio zapisi. Također se prati status svake poruke u aplikaciji kao i status i datum i vrijeme zadnjeg korištenja aplikacije svakog korisnika. Svi statusi poruka se prate u realnom vremenu, a status korisnika i datum i vrijeme zadnjeg korištenja može imati mala kašnjenja.

2.1.2. Korisničke klase i karakteristike

Aplikacija se sastoji od jedne glavne korisničke klase koja se kasnije može promijeniti ovisno o korisnikovim akcijama. Tako je osoba koja kreira novu grupu automatski administrator te grupe. Time će se dobiti dvije korisničke klase samo za grupe: administrator i korisnik grupe. Svaki korisnik je obični korisnik, ali mu se mijenja status posebno za svaku grupu. Tako da jedan korisnik može biti administrator u jednog grupi, a obični korisnik u nekoj drugoj grupi. To znači da se uloge dijele jedino unutar grupa i da svaka grupa ima posebno određene uloge pojedinih korisnika. Administrator ima veće privilegije od korisnika te može dodavati nove korisnike, dodjeljivati korisnicima ulogu administratora te brisati korisnike iz grupe. Korisnik grupe može mijenjati sliku grupe, mijenjati opis grupe te svakako izaći iz grupe isto kao i administrator.

2.1.3. Radno okruženje

- SUBP: MySQL Workbench 8.0 CE
- Programiranje: Visual Studio Code 2021
- Dizajn modela: Visual Paradigm i Draw.io
- Dokumentacija: Office Word 2019
- *Backend* okruženje: Node.js
- *Frontend* okruženje: React Native

2.1.4. Ograničenja dizajna i implementacije

Frontend okruženje ima pojedinih ograničenja naspram klasičnih okruženja za izradu Android i iOS aplikacija, tj. pisanja koda aplikacije u prirodnom kodu (engl. native code), jer su okruženja poput Android studio napravljena da bi se u njima programirale Android aplikacije u jeziku Java ili Xcodeu za iOS aplikacije, a u ovom projektu je korišten React Native koji ne podržava sve funkcionalnosti koje podržavaju okruženja prirodnog koda. Nedostatak pisanja aplikacija za mobilne uređaje u prirodnom kodu je što se ista aplikacija mora napisati i za Android i za iOS. Ovo je ujedno i najveća prednost korištenja okvira poput React Nativea baš zbog toga što se piše jedan kod, a on se kasnije može izgraditi i u Android i u iOS aplikaciju [5]. Ima pojedinih ograničenja s kompatibilnošću s ostalim paketima koje se normalno integriraju u aplikacije prirodnog koda zato što je okvir relativno mlad. Također se ne može sve odraditi samo s jednim kodom pošto neke značajke ovise o platformi na kojoj su pokrenute tako da se svejedno mora imati dosta znanja i o Android i iOS aplikacijama s prirodnim kodom, ali dobra stvar je što se u React Native može dodatno ugraditi prirodni kod tako da nismo totalno ograničeni da je to nemoguće napraviti [5]. Također, baza podataka ima pojedina ograničenja kod skaliranja jer bi se ovakva vrsta aplikacije kod velikog broja korisnika definitivno morala horizontalno skalirati. To bi se najbolje riješilo takozvanom „master – slave“ arhitekturom u kojoj imamo jednu glavnu instancu baze i ostale pomoćne. Glavna instanca služi samo za pisanje u bazu, a ostale pomoćne samo za čitanje iz baze. Najbolje rješenje za ovaj problem aplikacije je da se koristi visoko skalabilna baza podataka „Cassandra“. Ona rješava dosta problema skaliranja, no u ovom projektu se koristi MySQL baza podataka jer sam puno bolje upoznat s njom, a i „Cassandra“ baza podataka ima svoj poseban jezik za programiranje baze CQL i ona je NoSQL baza. To znači da ona nije relacijska baza podataka, nego je strukturirana kao „ključ-vrijednost“ arhitektura.

2.1.5. Pretpostavke i ovisnosti

Za implementaciju aplikacije da radi u stvarnom vremenu koristit će se Socket.IO knjižnica koja omogućuje dvosmjernu komunikaciju u stvarnom vremenu temeljenu na događajima. Ova knjižnica pojednostavljuje implementiranje WebSocket tehnologije, no ima ograničenja da se samo klijent koji koristi također Socket.IO klijentsku verziju knjižnice može povezati na poslužitelj koji koristi Socket.IO, klasični poslužitelj ili klijent s običnom implementacijom WebSocket tehnologije ne može uspostaviti konekciju s klijentom ili poslužiteljem napravljenim pomoću Socket.IO knjižnice. [3]

2.2. Zahtjevi vanjskih sučelja

U ovom dijelu specifikacije pojašnjena su korisnička sučelja, hardverska sučelja, softverska sučelja te komunikacijska sučelja.

2.2.1. Korisnička sučelja

Korisnicima preko glavnih sučelja bit će omogućeno komuniciranje putem interneta s drugim korisnicima aplikacije. Početno korisničko sučelje sastojat će se od popisa razgovora s korisnicima ujedno i grupama te će biti kronološki poredano ovisno o razgovoru u kojem je zadnje pristigla poruka. Time će korisnik moći lakše i preglednije započinjati razgovore s osobama s kojima je češće u kontaktu ili trenutno komunicira s njima. Također će biti opcija stvaranja nove poruke te pregleda grupa, pregleda profila i dodavanja novih korisnika kao kontakte u aplikaciji. Korisnik će moći na početnoj stranici otići i na listu poziva. Korisnik može pretraživati razgovore po korisnicima koji sudjeluju u njima.

Nakon pritiska na neki od ponuđenih razgovora u listi razgovora, korisniku se otvara novo sučelje koje služi direktno za komunikaciju. Na dnu ekrana se nalazi element za unos teksta u kojem se sastavlja poruka te opcije za dodavanje pojedine slike ili videa. Korisnik može listati kroz poruke koje su kronološki poredane od zadnje pristigle poruke prema prvoj. Isto korisničko sučelje te sam način rada je napravljen i za opciju grupnog razgovora. Jedina razlika je što u ovom tipu razgovora imamo više korisnika koji međusobno komuniciraju. Također se može označiti pojedina poruka kako bi se stavila u listu označenih poruka i kasnije se lakše pronašla.

Korisnik može otvoriti korisničko sučelje svog profila na aplikaciji u kojem može ažurirati podatke o profilu isto kao i profilnu sliku. Korisnik je u mogućnosti promijeniti svoj naziv koji je vidljiv drugim korisnicima, svoje korisničko ime, svoju profilnu sliku, opis profila, lozinku te email koji mu je trenutno aktivan.

2.2.2. Hardverska i softverska sučelja

Aplikacija je namijenjena mobilnim operacijskim sustavima Android i iOS. Aplikacija će najmanje podržavati 11.0 verziju iOSa i 5.0 (API 21) verziju Android sustava i normalno svaku noviju verziju od tih dvaju operacijskih sustava.

Baza podataka je baza podataka MySQL koja je pokrenuta na Windows desktop operacijskom sustavu, a poslužitelji su također pokrenuti na istom sustavu. Mobilna aplikacija se mora instalirati na mobilni uređaj.

2.2.3. Komunikacijska sučelja

Sustav se sastoji od komunikacije između klijentske mobilne aplikacije i poslužitelja te poslužitelja i baze podataka. Prvo ćemo navesti komunikaciju između poslužitelja i baze podataka jer je onda jednostavnija. Ova komunikacija je direktna komunikacija preko TCP protokola na određeni port zadan u bazi podataka. Komunikacija se sastoji od dvije glavne faze: faza konekcije i naredbena faza. Poslužitelj prvo uspostavlja konekciju s bazom te dostavlja podatke potrebne za autentifikaciju, to je dio faze konekcije na bazu podataka MySQL u kojoj ona dopušta ili odbija konekciju ovisno o valjanosti podataka za ovjeru. Ako baza podataka dopusti konekciju slijedi naredbena faza u kojoj poslužitelj vrši određene naredbe u bazi podataka sve dok on sam ne prekine konekciju ili se konekcija prekine na neki nepredviđeni način (npr. prekine se Internet veza). Poslužitelj i baza podataka će raditi na istom računalu tako da će konekcija biti lokalna. Konekcija između mobilne aplikacije i poslužitelja vrši se preko HTTP protokola i WebSocketa ovisno o tome što korisnik želi učiniti. Za komunikaciju između korisnika, tj. funkcionalnosti razgovora dvoje korisnika ili razgovori unutar grupa se izvode isključivo preko WebSocket protokola. Sustav se sastoji od 2 instance poslužitelja, jedna služi kao WebSocket sučelje koja ujedno i rukuje svim WebSocket događajima te komuniciranjem između korisnika u stvarnom vremenu. Druga instanca poslužitelja služi kao REST API koji obavlja klasične CRUD operacije poput dohvaćanja podataka korisnika, dohvaćanja popisa razgovora, dohvaćanja poruka prilikom otvaranja postojećeg razgovora. Poslužitelj za upravljanje razmjenom poruka može obavljati funkcije slanja poruka u realnom vremenu, ažuriranja stanja poruka, slanja multimedije u realnom vremenu, praćenje statusa korisnika, praćenje je li drugi korisnik trenutno piše poruku.

2.3. Sistemske značajke

U ovom dijelu specifikacije pojašnjene su sistemske značajke sustava, tj. pojedine funkcionalnosti koji sustav uistinu ima.

2.3.1. Registracija korisnika

Registracija korisnika obavlja se na način da korisnik upisuje svoje ime, prezime, korisničko ime koje mora biti jedinstveno i direktno se provjerava iz baze podataka da li je već zauzeto, lozinku te potvrdu lozinke koja osigurava ispravnost prvog unosa lozinke korisnika. Nakon toga podatci prolaze kroz osnovnu validaciju i ako je validacija uspješna, korisnik je pravilno registriran u sustav i može se početi koristiti sustavom. Ukoliko je validacija neispravna, aplikacija će korisnika obavijestiti točno što je krivo uneseno. Kod registracije lozinka će se odmah na poseban način pribrojiti s automatski generiranom „soli“ (engl. *salt* – automatski generirani niz znakova koji se pribrojava lozinki prilikom sažimanja radi povećanja sigurnosti) posebnom za tog korisnika te će se tada napraviti „sažetak“ (engl. *hash*) tog pribrojenog znakovnog niza kako bismo osigurali veću sigurnost pri zaštiti lozinke korisnika. To znači da se u bazu podataka ne spremaju direktno tekstualno napisane lozinke, nego se gore opisanim načinom radi sažetak lozinke koji se sprema u bazu podataka. Ovim načinom direktno osiguravamo krađu lozinke prilikom neovlaštenog upada u bazu podataka i dohvaćanja podataka o korisniku.

2.3.2. Prijava korisnika

Prijava korisnika odvija se na više načina ovisno o tome da li je korisnik tek registrirani korisnik koji se prvi put prijavljuje u aplikaciju te ovisi o tome da li je korisnik omogućio automatsku prijavu u aplikaciju. Ako je korisnik tek registriran, onda on nema opciju automatske prijave već mora obaviti prijavu na sljedeći način: upisuje svoje korisničko ime i lozinku te može odabrati opciju da aplikacija radi na principu automatske prijave. Tek registrirani korisnik, kada se uspješno prijavi prvi put, mora unesti svoj naziv koji će drugi vidjeti u aplikaciji te opcionalno može unesti svoj opis. Nakon toga slobodno može koristiti aplikaciju. Ukoliko se prilikom prijave označi opcija automatske prijave, aplikacija će kod sljedećeg pokretanja aplikacije provjeriti u SecureStoreu da li je vrijednost ključa „zapamti“ jednaka istini. Ukoliko je to točno, aplikacija na REST API šalje spremljeni JWT token također iz SecureStorea. Ukoliko je token validan, poslužitelj iz tokena izvlači korisničko ime te vraća identifikator korisnika. Aplikacija dozvoljava pristup ukoliko je poslužitelj vratio uspješan odgovor, a ukoliko je vraćen neuspješan odgovor, prikazuje se obrazac za prijavu. Ovim

načinom dobivamo automatizaciju sistema prijave u kojoj korisnik u svakoj sljedećoj prijavi jednostavno pokrene aplikaciju na svom telefonu i bez ponovnog unošenja podataka o prijavi, poslužitelj obavlja autentifikaciju i aplikacija direktno dozvoljava pristup korisniku koji je uspješno autentificiran. Uspješnom prijavom u aplikaciju, zapisuje se JWT token u lokalni spremnik telefona tj. u takozvani „SecureStore“ kako bi korisnik mogao biti autentificiran pri svakom daljnjem korištenju aplikacije. Ukoliko korisnik odabere da želi spremiti svoje podatke za prijavu, JWT token ostaje u sigurnom spremištu aplikacije te će služiti kao sigurnost za sljedeću prijavu u aplikaciju.

2.3.3. Glavni izbornik

Glavni izbornik aplikacije nalazi se u zaglavlju aplikacije na početnoj stranici s desne strane. Ikona glavnog izbornika su tri vertikalne točkice. Pritiskom na ikonu otvara se izbornik s opcijama kreiranje grupe, napuštanja grupe, brisanja kontakta, označenih poruka te odjavom iz aplikacije. Pritiskom na ekran izvan glavnog izbornika kada je on aktivan događa se zatvaranje istog.

2.3.4. Dodavanje kontakata

Da bi korisnik uopće mogao koristiti glavnu djelatnost ove aplikacije, komuniciranje s drugim korisnicima, prvo mora obaviti pretraživanje i dodavanje drugih ljudi kao njegove kontakte. Dodavanja kontakata je jedan od tri glavna prozora u početnoj stranici aplikacije. Korisnik prvo mora pretražiti kontakte prema njihovom nazivu te mu se nakon toga prikazuje filtrirana lista svih korisnika aplikacije. Korisnik ne može odmah dodati drugu osobu kao kontakt, nego mu prvo može samo poslati zahtjev. Nakon uspješno poslanog zahtjeva, zahtijevani korisnik se više neće nalaziti u popisu filtriranih korisnika prilikom pretraživanja. Također, zahtijevani korisnik ne može poslati zahtjev prvobitnom korisniku jer je zahtjev između ta dva korisnika već aktivan. Zahtijevani korisnik tada dobiva zahtjeva za kontakt i može odbiti ili prihvatiti taj zahtjev. Odbijanjem zahtjeva korisnici neće postati kontakti i oboje ponovno mogu zatražiti zahtjev za kontaktom. Prihvaćanjem zahtjeva korisnici će postati kontakti i mogu započeti razgovor.

2.3.5. Brisanje kontakata

Brisanjem kontakata može se pristupiti putem glavnog izbornika aplikacije. Otvara se nova stranica s popisom svih korisnikovih kontakta i opcijom da se pojedini obriše. Brisanje kontakta povlači brisanje sesije s tim kontaktom što povlači brisanje svih dosadašnjih poruka između ta dva kontakta. Ponovnim dodavanjem korisnika u kontakte, kreira se nova sesije i ponovno se mora započeti razgovor.

2.3.6. Kreiranje grupe

Kreiranju grupe može se pristupiti putem glavnog izbornika. Otvara se nova stranica s popisom svih korisnikovih kontakata i opcijom da se pojedini kontakt označi kao kontakt koji ulazi u novokreiranu grupu. Da bi se moglo nastaviti s daljnjim kreiranje grupe mora se izabrati barem jedan kontakt. Ukoliko korisnik još nije dodao niti jednog korisnika kao kontakt, korisnik ne može napraviti grupu i umjesto popisa kontakata ispisano je da doda osobe kao kontakte kako bi mogao kreirati grupu. Nakon odabira kontakta, korisnik odlazi na sljedeću stranicu na kojoj postavlja sliku profila opcionalno, postavlja naziv grupe te opcionalno postavlja opis grupe. Potom se otvara stranica s osnovnim informacijama o grupi te unosom poruke pozdrava. Pravilnim unosom poruke pozdrava korisnik uspješno može kreirati novu grupu. Označeni korisnici će automatski biti dodani u novonastalu grupu, a samoj grupi se može pristupiti na početnoj stranici u kartici „Razgovori“ u popisu posljednjih razgovora.

2.3.7. Napuštanje grupe

Svaki korisnik može napustiti bilo koju grupu u kojoj se nalazi. Ovu radnju može obaviti na dva načina. Prvi način je da pristupi stranici za napuštanje grupa putem glavnog izbornika u kojoj se nalazi popis svih korisnikovih grupa te opcija da napusti grupu. Drugi način je da kroz grupni razgovor dođe do informacija o grupi u kojoj se nalazi gumb koji služi za napuštanje grupe.

2.3.8. Započinjanje razgovora

Započinjanje razgovora se odvija samo u slučaju komunikacije dva korisnika i ne može se primijeniti u slučaju grupnog razgovora. Započinjanje razgovora se nalazi na kartici „Razgovori“ na početnoj stranici na kojoj je gumb u donjem desnom kutu ekrana koji služi za kreiranje novog razgovora. Klikom na gumb otvara se nova stranica u kojoj vidimo listu naših kontakata. Odabirom pojedinog kontakta otvara se stranica razgovora u kojoj piše da slanjem poruke započinjemo razgovor s nekom osobom ukoliko do sada nije bilo poruka ili će biti prikazan dosadašnji razgovor s tom osobom ukoliko je bilo poruka. Grupni razgovor funkcionira

tako da se kod kreiranja grupe mora dodati poruka pozdrava i tada se grupni razgovor nalazi na početnoj stranici na kartici „Razgovori“ u listi posljednjih razgovora.

2.3.9. Komunikacija s korisnikom

Korisnik ima više opcija kod komunikacije s drugim korisnikom ili nekom od grupa. Može izabrati opciju započinjanja novog razgovora koja je detaljnije objašnjena u točki 2.4.4. ovog dokumenta ili može izabrati neki od postojećih razgovora u listi posljednjih razgovora na početnoj stranici. Ukoliko odabere bilo koju od opcija daljnje komuniciranje s drugim korisnikom/grupom se odvija na isti način i korisnik će imati potpuno jednake opcije. To jest, otvara se ista stranica u aplikaciji bilo to da pristupimo razgovoru preko popisa korisnika ili odabirom razgovora u listi posljednjih razgovora. Komunikacija se može odvijati putem pisanja tekstualne poruke, slanja slike, slanja videozapisa ili slanja zvučnog zapisa. Sve poruke u listi poruka će biti prikazane kronološki od posljednje poslane poruke u razgovoru neovisno o korisniku koji ju je poslao ni o sadržaju poruke. Svaka poruka ima svoja 3 statusa. Statusi poruke su sljedeći: poruka poslana (u aplikaciji se označava jednom kvačicom), poruka stigla do poslužitelja (u aplikaciji se označava s dvije kvačice) te poruka pročitana (u aplikaciji se označava s dvije kvačice zelene boje). Sva komunikacija između korisnika odvija se u realnom vremenu.

2.3.10. Komunikacija u grupnom razgovoru

Komunikacija u grupnom razgovoru identična je komunikaciji između dva korisnika. Jedina razlika između jedan na jedan razgovora i grupnog razgovora je što u grupnom razgovoru može sudjelovati više od dvije osobe do predefinirano dvadeset osoba u svakom grupnom razgovoru. Vrsta komunikacije između korisnika, odnosi se na tekstualne poruke, slike, videozapise i audio snimke, ostaje ista kao i kod razgovora jedan na jedan. U grupnom razgovoru postoje dva tipa korisnika: administrator i obični korisnik. Samo administratori mogu dodjeljivati i brisati ulogu administratora drugim korisnicima te samo oni mogu brisati druge osobe iz grupe. Svi korisnici grupnog razgovora mogu mijenjati naziv, opis te profilnu sliku. Ova opcija je onemogućena kod razgovora jedan na jedan jer bi tada korisnik mijenjao naziv, opis i sliku neke druge osobe.

2.3.11. Uređivanje profila

Uređivanje profila odnosi se na uređivanje imena, prezimena, naziva, profilne slike te opisa profila. Također korisnik ima opciju dodavanje nove profilne slike.

2.3.12. Označene poruke

Svaki korisnik može bilo koju poruku označiti kako bi imao brži pristup do nje. Mogućnost označavanja poruke ne ovisi o tip razgovora niti o pojedincima unutar tog razgovora. Svaki korisnik može pristupiti svojim označenim porukama na posebnoj stranici u aplikaciji i tako imati direktan pristup do te poruke bez da mora kronološki tražiti poruku u određenom razgovoru. Također tip poruke ne ovisi o tome da li poruka može biti označena. Do označenih poruka može se doći putem glavnog izbornika aplikacije koji se nalazi u zaglavlju aplikacije na desnoj strani.

2.3.13. Odjava korisnika

Korisnik se može odjaviti iz aplikacije tako da izabere opciju „Odjava“ iz glavnog izbornika. Klikom na odjavu korisniku se brišu podatci iz lokalnog sigurnog spremišta aplikacije i postavlja se njegov trenutni status na odjavljen u bazi podataka. Time se postavlja vrijednost ključa „zapamti“ na lažno u lokalnom sigurnom spremištu i prilikom sljedećeg pokretanja aplikacije neće biti moguće automatska prijava, već se korisnik morati ručno ponovno prijaviti u aplikaciju.

2.3.14. Status korisnika

Prilikom prijave u aplikaciju, korisnikov trenutni status se postavlja na aktivan. Status drugih korisnika može se vidjeti u stranici razgovora s tim korisnikom. Ukoliko je u zaglavlju stranice pored profilne slike korisnika zelena točkica, to znači da je korisnik aktivan. Ukoliko nema zelene točkice, to označava da je korisnik trenutno neaktivan. Status se također može vidjeti ako se u razgovoru s korisnikom ode na stranicu informacija o pojedinom korisniku u kojoj je na isti način prikazan status trenutne aktivnosti korisnika.

2.3.15. Zadnje viđen

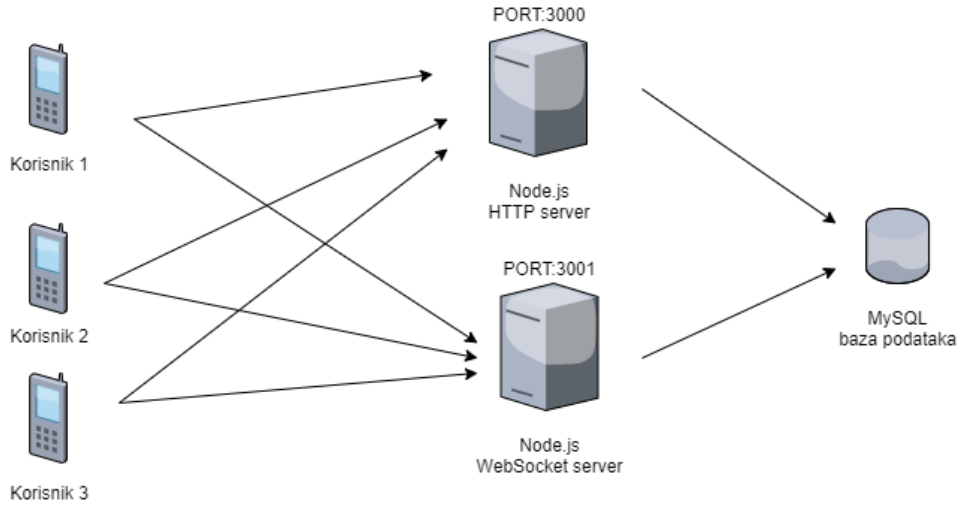
Sen [4] u svom videu pojašnjava kako je ovu značajku najbolje napraviti tako da se datum i vrijeme zadnjeg viđenja korisnika spremi u bazu podataka i da se onda to polje ažurira prilikom svakog zahtjeva prema poslužitelju pa je takva implementacija ostvarena i u ovom sustavu. Prilikom korištenja aplikacije, na svaki korisnikov zahtjev prema poslužitelju u bazi podataka se ažurira datum i vrijeme na trenutni datum i vrijeme. Time se prati datum i vrijeme kada je korisnik koristio aplikaciju. Datum i vrijeme zadnje aktivnosti pojedinog korisnika može se vidjeti na stranici razgovora s tim korisnikom. Ono se nalazi u zaglavlju stranice ispod naziva i profilne slike korisnika.

3. Specifikacija softverskog dizajna

Svrha ovog dokumenta je detaljno objasniti cijelu arhitekturu ovog sustava. Počevši od samog dizajna arhitekture sustava do pojedinih dijagrama koji će pobliže objasniti neke određene kompleksnije dijelove sustava.

3.1. Arhitektura cjelovitog sustava

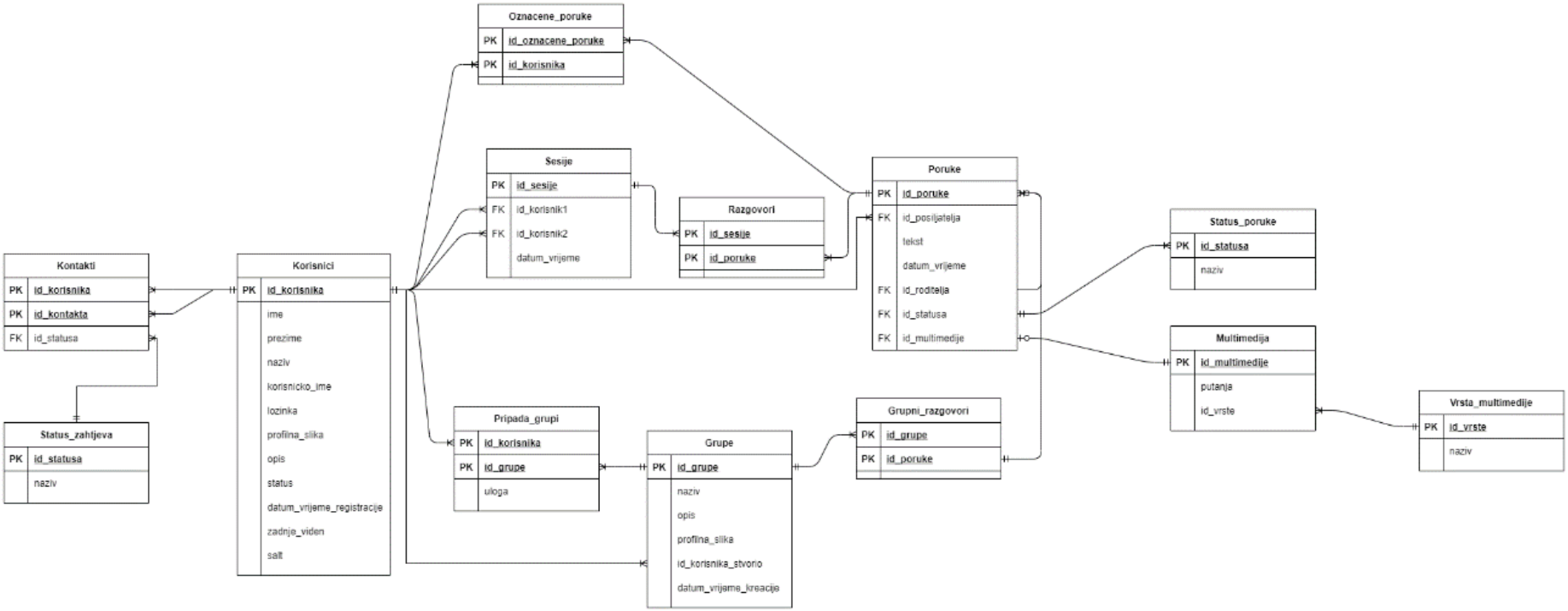
Arhitektura cjelovitog sustava sastoji se od tri glavna dijela: klijentskog dijela, poslužiteljskog dijela i baze podataka. Jain [6] u svom videu pojašnjava kako je najbolje podijeliti arhitekturu poslužitelja na dva dijela: jedan za dohvaćanje resursa, a drugi za upravljanje razmjenom poruka. U teoriji bi poslužitelj za upravljanje razmjenom poruka trebao obavljati samo taj posao jer se na njemu očekuje najveći promet i najviše konekcija. Klijentski dio sastoji se od mnogo instanci mobilne aplikacije instalirane na različite uređaje koji održavaju komunikaciju s poslužiteljskim dijelom. Poslužiteljski dio sastoji se od dvije instance poslužitelja Node.js. Jedna služi kao REST API za klasičnu HTTP razmjenu podataka, dok druga instanca služi kao poslužitelj za upravljanje razmjenom poruka. Baza podataka je baza podataka MySQL koja je u komunikaciji s poslužiteljskim dijelom.



Slika 1. Arhitektura cjelovitog sustava (Izvor: Fran Turković, 2021)

3.2. ERA model

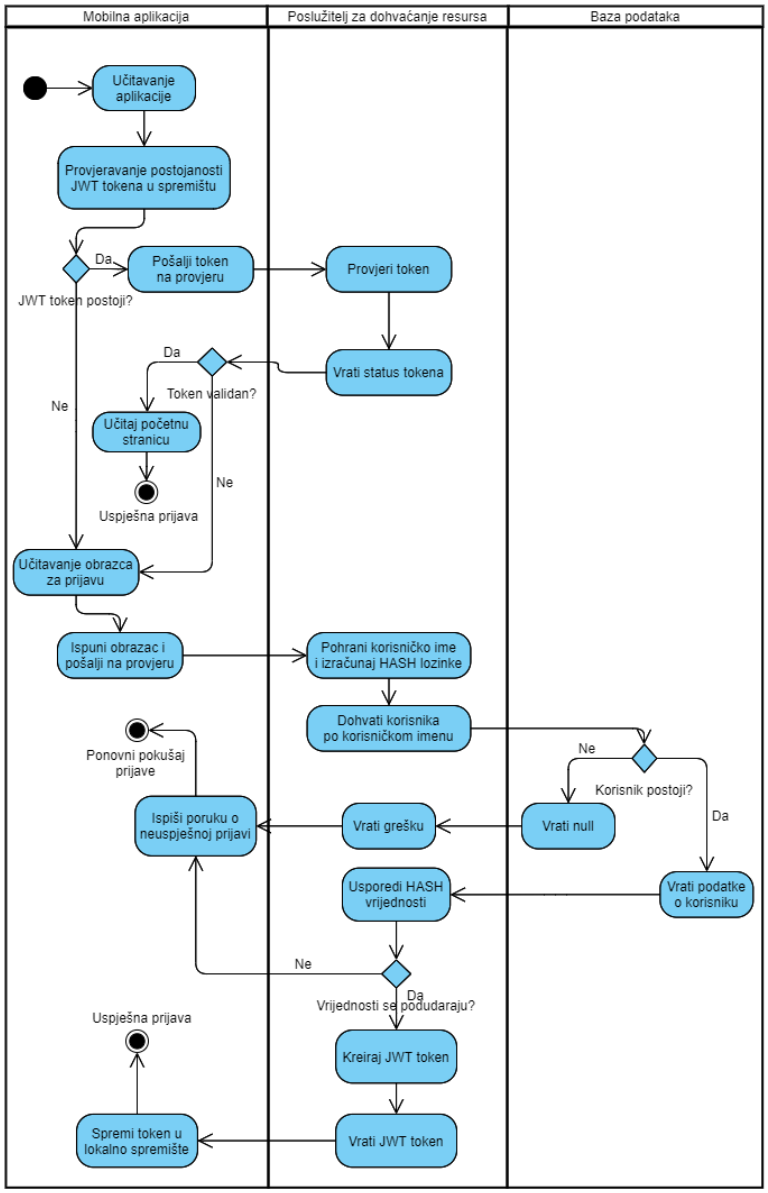
ERA model ovog sustava sastoji se od 13 entiteta i potrebnih veza između njih te atributa koji im logički pripadaju. Veze se odnose na korisnike, grupe, sesije i poruke. Iz prikazanog ERA modela napravljena je baza podataka. Posebna veza koju je potrebno istaknuti je vanjski ključ u tablici „Poruke“ nad atributima „id_poruke“ i „id_roditelja“. Atribut „id_roditelja“ se odnosi na „id_poruke“ u istoj tablici pomoću kojeg možemo dohvatiti prethodnika određene poruke. Takvim sistemom, dobivamo puno na brzini dohvaćanja poruka jer se one u bazi podataka mogu dohvatiti samo preko njihovih primarnih ključeva pomoću rekurzivnog CTE upita. Stari način bez ove unarne 1:1 veze je bio da se izaberu sve poruke te tada sortiraju uzlazno po datumu što je vrlo spora implementacija naspram sistema unarne 1:1 veze s rekurzivnim CTE upitom.



Slika 2: ERA model (Izvor: Fran Turković, 2021)

3.3. Dijagram aktivnosti prijave

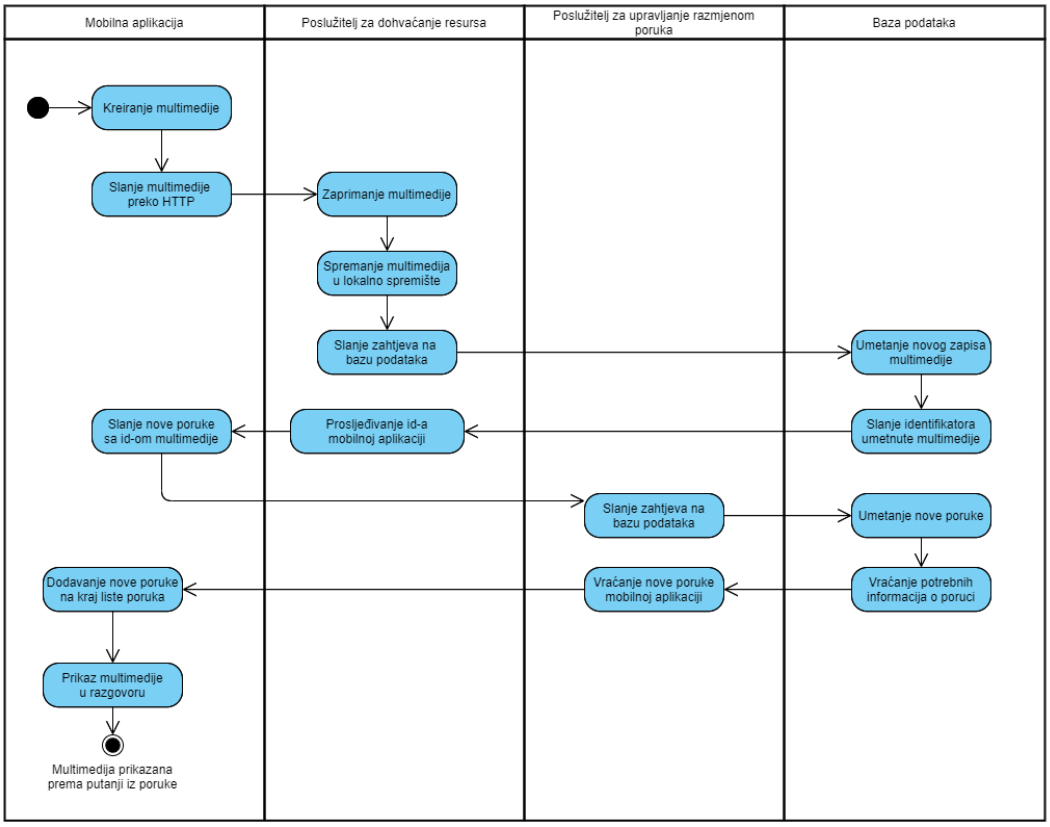
Ovaj dijagram je istaknut jer pokazuje dva glavna načina prijave: automatsku prijavu i ručnu prijavu. Automatska prijava radi tako da se uzme JWT token iz lokalnog spremišta telefona ukoliko postoji te napravi provjeru na poslužitelju da li je token validan. Ukoliko je validan, korisniku je dopušteno korištenje aplikacije i uspješno je ovjeren njegov identitet. Ručna prijava radi tako da korisnik na poslužitelj pošalje svoje korisničko ime i lozinku te poslužitelj odradi provjeru identiteta na taj način te korisniku vrati JWT token koji mu služi za daljnju provjeru identiteta.



Slika 3: Dijagram aktivnosti prijave (Izvor: Fran Turković, 2021.)

3.4. Dijagram aktivnosti slanja multimedije

Na sljedećoj slici nalazi se postupak slanje multimedije u poruci. Trik kod slanja multimedije u poruci je da ne možemo direktno poslati poruku, već prvo moramo multimediju pohraniti na poslužitelj za dohvaćanje resursa. Kada se multimedija pohrani na poslužitelj i zapiše se njezin identifikator u bazu podataka, poslužitelj te informacije vraća korisničkoj strani koja onda tek šalje novu poruku s identifikatorom multimedije. Poslužitelj za upravljanje razmjenom poruka tada pohranjuje novu poruku te prosljeđuje novu poruku svim aktivnim korisnicima u razgovoru. Poruka nema tekst, nego posjeduje putanju do multimedije na poslužitelju te vrstu multimedije. Tim načinom lako možemo razlikovati tip multimedije i koju multimediju je potrebno prikazati u popisu poruka u pojedinom razgovoru.



Slika 4. Dijagram aktivnosti slanja multimedije (Izvor: Fran Turković, 2021.)

4. Implementacija poslužitelja za upravljanje razmjenom poruka i opis rada komunikacije

Poslužitelj je implementiran u Node.js okruženju uz pomoć Express web aplikativnog okvira te Socket.IO JavaScript biblioteke za web aplikacije u stvarnom vremenu. Ove tehnologije su izabrane jer su vrlo jednostavne za korištenje, relativno brzo rade te je jako velika brzina implementacije baš zbog njihove jednostavnosti.

Socket.IO je knjižnica koja se sastoji od dva dijela: jedan za klijenta, a jedan za poslužitelja. Da bi uspostavili komunikaciju preko WebSocket tehnologije, moramo oba dijela knjižnice ispravno implementirati. Klijentski dio knjižnice je implementiran u mobilnoj aplikaciji, a poslužiteljski dio je implementiran na poslužiteljskoj strani sustava. Socket.IO je zapravo prilagođen transportni protokol nadograđen na postojeće protokole u realnom vremenu, poput npr. WebSocket tehnologije . Knjižnica nije direktna implementacija WebSocket tehnologije, te se samo klijent koji koristi Socket.IO može spojiti na Socket.IO poslužitelj i obrnuto, a to znači da se komunikacija može uspostaviti samo između Socket.IO klijenta i poslužitelja. Time dobivamo jednu razinu sigurnosti u aplikaciji. Knjižnica radi na principu slanja i slušanja događaja. I klijent i poslužitelj mogu slušati i slati događaje. Klijenti obično šalju pojedinačne događaje na poslužitelj, a poslužitelj onda može vratiti novi događaj kao odgovor na događaj klijenta te taj odgovor može poslati jednom, više klijenata ili više klijenata bez inicijalnog klijenta koji je poslao prvobitni događaj. [3]

U ovom poslužitelju samostalno je implementiran način praćenja trenutno spojenih klijenata. Sama knjižnica pruža jedinstveni identifikator svake nove konekcije, ali je problem što se taj identifikator razlikuje od primarnog ključa korisnika u bazi podataka. Problem je riješen *hash* tablicom u koju upisujemo primarni ključ korisnika iz baze kao ključ i identifikator konekcije kao vrijednost. Na svakoj konekciji klijent šalje svoj primarni ključ na poslužitelj. Time smo osigurali da u svakom trenutku znamo koji je korisnik aktivan u aplikaciji i kojem korisniku možemo direktno proslijediti novu poruku. Na prekinutoj konekciji korisnika, poslužitelj briše tog korisnika iz *hash* tablice. Poslužitelj se dalje sastoji od pojedinih slušača događaja: za novu poruku, kada se poruka označi ili odznači, kada korisnik pročita pojedine poruke, kada korisnik pročita novu poruku te kada korisnik krene pisati poruku. Pojedini slušači događaja s cijelim sistemom kako rade od klijenta do poslužitelja će biti daljnje objašnjeni.

4.1. Slanje poruka

Slanje poruka se sastoji od dva glavna dijela: slanja tekstualne poruke i slanja multimedije. Slanje tekstualne poruke započinje korisnikovim pisanjem poruke i slanjem nove poruke na poslužitelj. Poruka se odmah ne zapisuje u razgovoru u mobilnoj aplikaciji, već se čeka odgovor poslužitelja čak i za klijenta koji je poslao poruku. Na taj način smo sigurni da je poruka uspješno stigla na poslužitelj i da je uspješno upisana u bazu podataka. Kada poslužitelj dobije novu poruku, on je zapisuje u bazu podataka i ako je uspješno upisana, on je vraća svim korisnicima koji su u tom razgovoru samo ako su trenutno spojeni na poslužitelj, a to se prati pomoću već spomenute *hash* tablice. Eze [7] objašnjava da je ovaj pristup najbolji ako poruke spremamo u trajno skladište podataka, objašnjava kako je najbolje prvo spremiti poruke u bazu podataka, a potom je proslijediti svim korisnicima jer na taj način imamo sigurnost da je poruka zaista spremljena u bazu podataka. Za upisivanje nove poruke u bazu podataka na poslužitelju se samo poziva procedura iz baze podataka. Procedura razlikuje vrstu poruke te da li je grupni razgovor ili ne. Ukoliko je poruka uspješno upisana, procedura vraća na poslužitelj sve potrebne podatke o novoj poruci koju tada poslužitelj šalje korisnicima. U mobilnoj aplikaciji je slušač događaja koji očekuje novu poruku i ako nova poruka stigne on je trenutačno stavlja na dno razgovora.

```
socket.on("poruka", async (poruka) =>{
      connection.query({
            sql: "call UnesiPoruku(?,?,?,?,?);",
            values:[ id_kontakta, id_posiljatelja, tekst, sesijaBool,
                     id_multimedije]},
      (error, results, fields)=>{
                  if(error){
                      return console.log("Error: " + error.message);
                  }
                  else{
                      for(var item of poruka.users){
                          if(clients.has(item.id_korisnika.toString())){
                              var socketIdToSend =
clients.get(item.id_korisnika.toString());
                              io.sockets.to(socketIdToSend).emit("newMessage"
                        ,results[0])
                          }
                      }
                  }
        })
})
```

Slanje multimedije radi na isti princip kao i slanje tekstualne poruke, ali se prvo multimedija mora poslati na poslužitelj za dohvaćanje resursa. Slikanjem nove slike, snimanjem videa ili zvučnog zapisa direktno po završetku multimedija se šalje na poslužitelj za dohvaćanje resursa koji multimediju zapisuje na memoriju poslužitelja te putanju i jedinstveni identifikator u bazu podataka. Nakon toga poslužitelj za dohvaćanje resursa korisniku vraća taj jedinstveni identifikator multimedije i korisnik potom šalje novu poruku na poslužitelj za upravljanje razmjenom poruka, samo što je u njoj tekst prazan, a identifikator multimedije je zapisan. Proces je dalje isti kao i kod slanja tekstualne poruke pošto procedura koja se poziva iz baze podataka razlikuje vrstu poruke. Poslužitelj za upravljanje razmjenom poruka tada vraća poruku korisniku bez teksta, ali s putanjom iz memorije poslužitelja koja vodi do te multimedije. Kada korisnik dobije novu multimedijsku poruku, u mobilnoj aplikaciji je napravljeno posebno stvaranje nove stavke u listi poruka za svaku od vrsta poruka. Tako se za multimedijske poruke ne prikazuje tekst nego se po putanji koja je zapisana u objektu nove poruke i vrsti poruke može učitati slika, videozapis ili zvučni zapis. Ukoliko je poruka tekstualna, onda će se stavka liste u listi poruka u razgovoru prikazati kao oblačić s tekstom te poruke.

4.2. Označavanje poruka

Označavanje poruka je specifično za svakog korisnika, tj. svaki korisnik ima svoje označene poruke i drugi korisnici ne mogu znati koje poruke su označene kod drugih korisnika. Ovim sistemom poslužitelj za upravljanje razmjenom poruka mora znati samo koja je poruka označena ili odznačena te koji je korisnik poslao događaj. Ukoliko je korisnik označio neku od poruka, onda se ta poruka sprema u bazu podataka u tablicu označenih poruka, a ukoliko je odznačio neku poruku, onda se ona briše iz tablice. Uspješnom radnjom u bazi podataka poslužitelj vraća samo tom korisniku identifikator te poruke i trenutni status označenosti poruke. U mobilnoj aplikaciji je slušač događaja za odgovor na označavanje poruka i ovisno o statusu trenutačno označava poruku kao označenu ili odznačenu, ovisno o statusu označenosti koji je dobio s poslužitelja.

```
socket.on("oznacenaPoruka", async (oznacenaPoruka) =>{
          var sql = "";
          if(oznacenaPoruka.status==1){
              sql = "insert into oznacene_poruke values(?,?);";
          }
          else{
              sql = "delete from oznacene_poruke where id_oznacene_poruke=?
and id_korisnika=?;";
          }
          connection.query({
              sql: sql,
              values:[oznacenaPoruka.id_poruke, oznacenaPoruka.id_korisnika]
          },(error, results, fields)=>{
              if(error){
                  return console.log("Error: " + error.message);
              }
              if(results.affectedRows==1){
                  var socketIdToSend =
      clients.get(taggedMessage.userId.toString());
                  io.sockets.to(socketIdToSend).emit("novaOznacenaPoruka",
      {id_poruke: oznacenaPoruka.porukaId, status: oznacenaPoruka.status})
              }
          })
})
```

4.3. Čitanje poruka prilikom otvaranja razgovora

Kako bi označili poruke kao pročitane od strane drugog korisnika, a taj korisnik prilikom slanja tih poruka nije bio na vezi, napravljen je sistem kojim, prilikom svakog otvaranja razgovora ili grupe, mobilna aplikacije šalje listu svih nepročitanih poruka na poslužitelj za upravljanje razmjenom poruka. Tada se u bazi podataka ažuriraju sve poruke na status „pročitane“ i baza podataka vraća te poruke poslužitelju koji ih prosljeđuje nazad korisniku. U mobilnoj aplikaciji se status svih tih poruka tada trenutačno mijenja na „pročitane“.

```
socket.on("azurirajPoruke", async (zadnjaPoruka) =>{
	connection.query({
		sql: "call AzurirajPoruke(?,?);",
		values: [zadnjaPoruka.id_korisnika, zadnjaPoruka.id_poruke]
		}, (error, results, fields)=>{
		if(error){
			return console.log("Error: " + error.message);
		}
		if(results[0]){
			for(var item of zadnjaPoruka.users){
				if(clients.has(item.id_korisnika.toString())){
					var socketIdToSend =
	clients.get(item.id_korisnika.toString());

					io.sockets.to(socketIdToSend).emit(
	"azuriranePoruke",results[0])
				}
			}
		}
	})
})
```

4.4. Čitanje nove poruke

Kada na klijentskoj strani dođe nova poruka i korisnik koji je zaprimio poruku nije isti korisnik koji je poslao poruku, odmah se na poslužitelj za upravljanje razmjenom poruka šalje status da je ta trenutno pristigla poruka odmah i pročitana. Poslužitelj tada tu poruku ažurira u bazi podataka na status „pročitana“ i vraća identifikator te poruke svim korisnicima. U mobilnoj aplikaciji se tada status te poruke odmah stavlja na status „pročitana“.

```
socket.on("novaVidjenaPoruka",async (novaVidjenaPoruka)=>{
      connection.query({
            sql: "update poruke set id_statusa=3 where id_poruke=?",
                  values: [novaVidjenaPoruka.id_poruke]
                  },
            (error, results, fields)=>{
                  if(error){
                    return console.log("Error: " + error.message);
                  }
                  if(results.affectedRows==1){
                      for(var item of novaVidjenaPoruka.users){
                          if(clients.has(item.id_korisnika.toString())){
                              var socketIdToSend =
clients.get(item.id_korisnika.toString());

                              io.sockets.to(socketIdToSend).emit(
"vidjenaPoruka",{id_poruke: novaVidjenaPoruka.id_poruke })
                          }
                  }
            }
      })
})
```

4.5. Status tipkanja

Kada korisnik započne pisati novu poruku, na poslužitelj se šalje događaj da je korisnik počeo tipkati s podatcima o korisniku, statusu tipkanja i popisu svih korisnika u razgovoru. Također vrijedi i za prestanak pisanja, samo se tada šalje status da je korisnik prestao pisati. U ovom slučaju nema nikakvog zapisivanja podataka u bazu podataka nego se događaj jednostavno proslijedi svim ostalim korisnicima u razgovoru koji su aktivni. U mobilnoj aplikaciji se prikaže status koji korisnik trenutno piše poruku, a prilikom prestanka pisanja status nestaje.

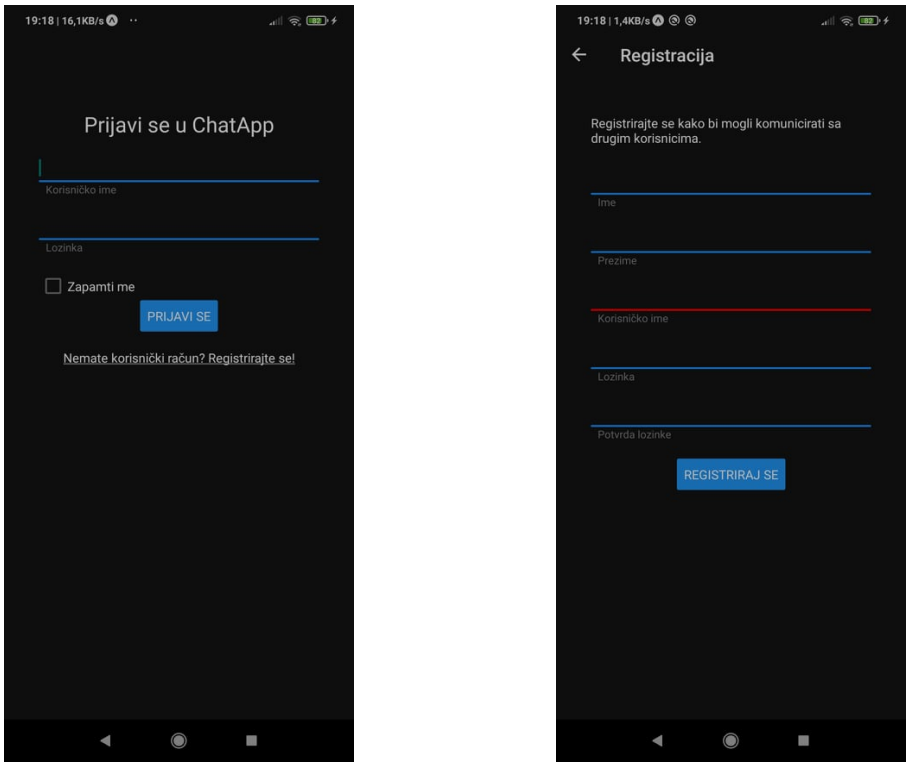
```
socket.on("typing",(obj)=>{
      for(var item of obj.users){
            if(clients.has(item.id_korisnika.toString())){
                  var socketIdToSend =
clients.get(item.id_korisnika.toString());
                  io.sockets.to(socketIdToSend).emit("typingBack", {typing:
obj.typing, userTyping: obj.currentUser, userTyping: obj.userTyping})
                  }
              }

    })
```

5. Korisnička dokumentacija

5.1. Prijava i registracija

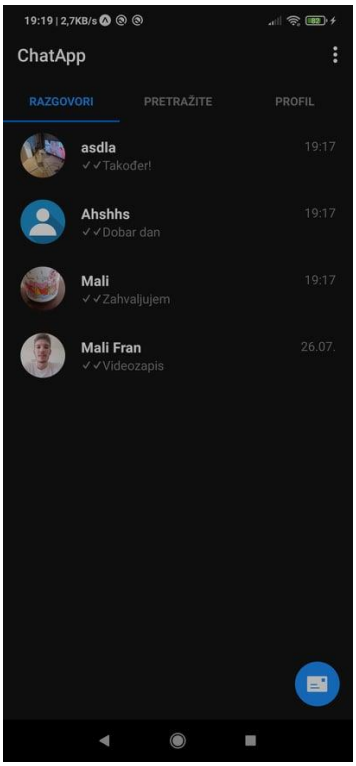
U sustav se prijavljuje unošenjem svog korisničkog imena i lozinke. Ako se izabere opcija „Zapamti me“, onda će sljedeća prijava u sustav biti automatska. Stranica prijave je s lijeve strane. Stranica registracije je s desne strane i ona se obavlja na način da se obavezno unese ime i prezime, potom korisničko ime koje se automatski provjerava da li je ispravno i da li već postoji. Nakon toga se moraju unesti lozinka i potvrda lozinke koje moraju biti iste.



Slika 5. Stranica prijave (lijevo) i stranica registracije (desno) (Izvor: Fran Turković, 2021.)

5.2. Razgovori

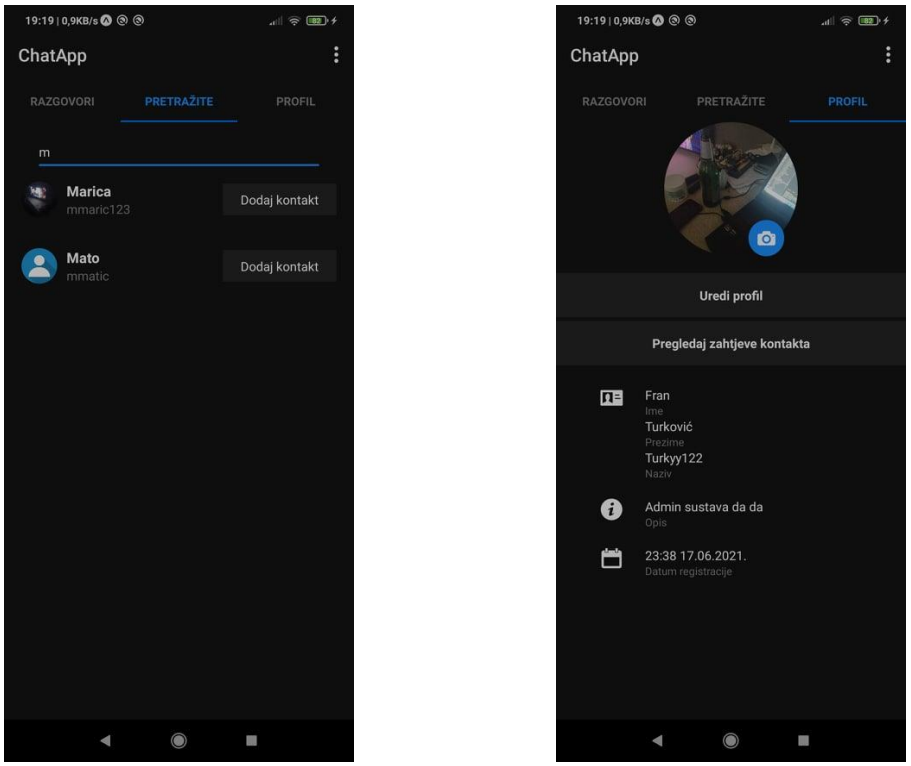
Početna stranica sadrži popis zadnjih razgovora i poruka u njima. Razgovori su kronološki poredani od posljednjeg. Klikom na pojedini razgovor otvara se stranica za razgovor s tom osobom ili grupom. Klikom na gumb u donjem desnom uglu otvara se stranica s kontaktima za kreiranje nove poruke. Klikom na tri točkice u zaglavlju aplikacije otvara se glavni izbornik aplikacije. Svaka stavka u popisu zadnjih razgovora se sastoji od profilne slike, naziva, poruke, statusa poruke i datuma kada je poruka dostavljena. Profilna slika i naziv se odnose na sliku i naziv korisničkog profila ako je razgovor s pojedinim korisnikom, a ukoliko je grupni razgovor, onda je to profilna slika grupe i naziv grupe. Ukoliko je poruka tekstualna poruka, onda će biti prikazan tekst poruke, a ako je bilo koji tip multimedije, onda će pisati koji je to tip multimedije.



Slika 7. Stranica razgovora (Izvor: Fran Turković, 2021.)

5.3. Pretraga korisnika i profil

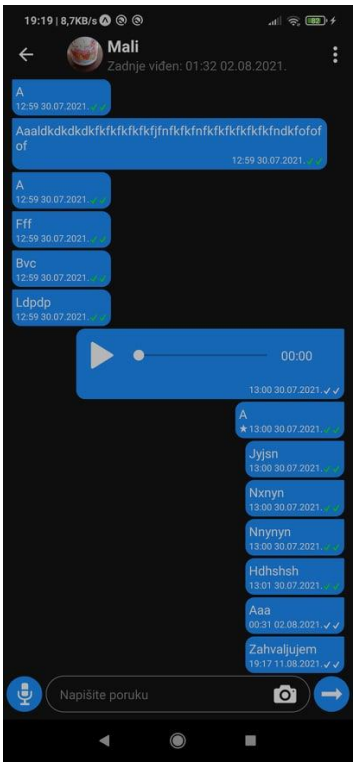
Na stranici pretrage korisnika, koja se nalazi na desnoj slici, možemo poslati zahtjev za kontaktom u listi dostupnih korisnika. Korisnici koji su dostupni u listi su korisnici kojima nismo već poslali zahtjev za kontaktom ili već od prije niste kontakti. Pretraga radi na način da pretražuje korisnike po njihovom korisničkom imenu. Kada pošaljemo zahtjev za kontakt, taj korisnik će dobiti zahtjev te dalje ovisi o njemu hoće li ga prihvatiti ili odbiti. Ukoliko korisnik prihvati zahtjev, možete započeti razgovor s tim korisnikom, a ukoliko odbije, moguće mu je ponovno poslati zahtjev. Stranica profila, na lijevoj slici, je stranica na kojoj možete mijenjati profilnu sliku, urediti osnovne informacije profila te pregledati zahtjeve za kontaktima i potvrditi ili odbiti određeni zahtjev. Na njoj se nalaze osnovne informacije o profilu poput: profilne slike, imena, prezimena, opisa profila te datuma registracije.



Slika 8. Stranica pretrage korisnika (lijevo) i stranica profila (desno) (Izvor: Fran Turković, 2021.)

5.4. Stranica čavrljanja

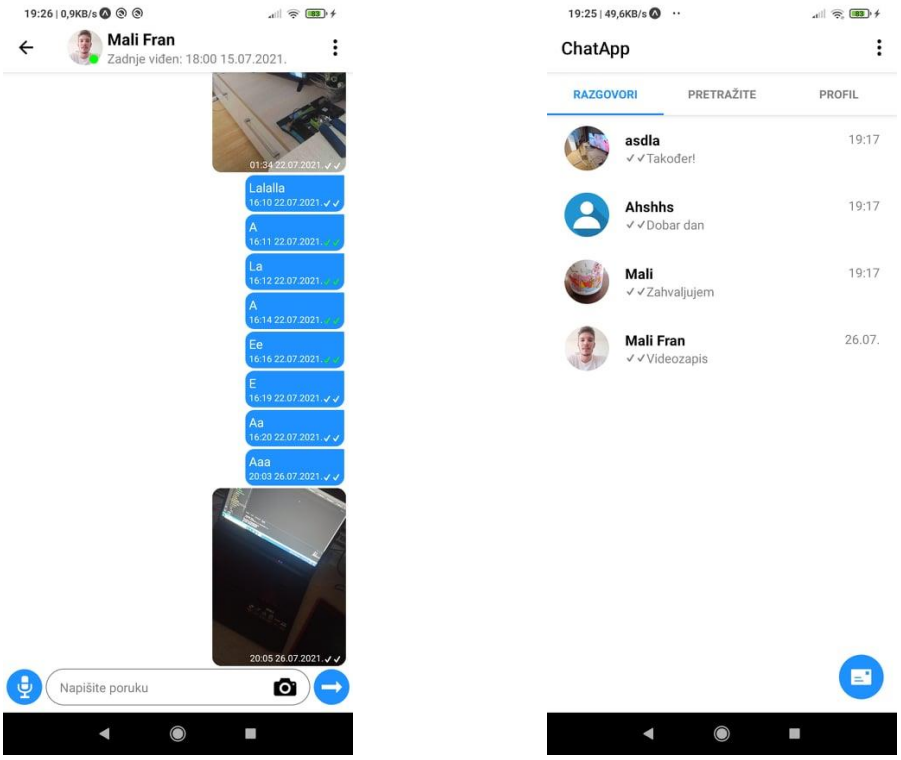
Stranica čavrljanja identična je za razgovore jedan na jedan i grupne razgovore. Na njoj se u zaglavlju prikazuje slika kontakta/grupe te naziv. Kod kontakata vidi se i kada je taj kontakt zadnje bio viđen u aplikaciji. Poruke su jednake za grupe i za razgovore jedan na jedan. U podnožju se nalazi gumb za snimanje zvuka koji snima zvuk kada ga se pritisne i drži se prst na njemu. Također se nalazi unos teksta u kojem unosimo tekstualnu poruku, gumb za otvaranje kamere u kojoj možemo poslati sliku ili video te gumb za slanje tekstualne poruka s desne strane podnožja. Pritiskom na tri točkice u zaglavlju stranice možemo vidjeti detaljnije informacije o grupi ili kontaktu te upravljati njima.



Slika 10. Stranica čavrljanja (Izvor: Fran Turković, 2021.)

5.5. Prikaz aplikacije u *light* načinu

Ovo je prikaz aplikacije u takozvanom *light* načinu. Aplikacije u potpunosti podržava *light* i *dark* način rada.



Slika 11. Aplikacija u *light* načinu (Izvor: Fran Turković, 2021.)

6. Zaključak

Mislim da je u ovom radu pogođena poanta, tj. da je napravljeno ono što je tema rada tražila, ali da se tu još može mnogo toga nadodati. Tema je zapravo vrlo široka i ako se želi napraviti aplikacija za korisničku komunikaciju poput WhatsAppa, to je zahtjevno vremenski, znanjem i poslom da odradi jedna osoba kao u ovom radu. Vidim daljnje nadogradnje aplikacije u vidu poziva putem telefonskog broja, video poziva, enkripcije i za grupnu i za komunikaciju jedan na jedan te također korektno napravljenu enkripciju za tekstualne i multimedijske poruke, poboljšanje cjelokupne sigurnosti aplikacije. Mislim da su sigurnost i integritet dvije vrlo važne osobine bilo koje aplikacije za korisničko komuniciranje. Također cijeli dio korisničke komunikacije u ovom sustavu je napravljen u realnom vremenu. U ovom radu je navedeno dosta kompleksnijih stvari koje su također istaknute, poput samog dizajna baze podataka za ovakav tip aplikacije kao i neke napredne tehnike u programskom jeziku SQL. Sam način kako odraditi sve funkcionalnosti u realnom vremenu, poput slanja multimedije, praćenja stanja aktivnih korisnika, praćenja korisnika koji trenutno tipka unutar razgovora su također dosta kompleksne, ali su detaljno objašnjene. *Frontend* dio aplikacije, tj. dio napravljen u React Nativeu, je nešto manje objašnjen pošto on nije direktno fokus rada, no i u tom dijelu ima dosta stvari koje bi se mogle istaknuti i detaljnije objasniti, poput transformacije liste poruka kako bi se lista uvijek pokazivala od kraja, tj. na zadnjoj poruci, kada otvorimo razgovor s nekim. Za korektnu izradu ovakvog sustava potrebno je dosta vremena i stručnjaka različitih znanosti, kao npr. definitivno nam je potrebna osoba koja je specijalizirana za enkripciju kako bi se taj dio odradio potpuno ispravno. Problem ovakvih sustava je što uvijek moraju biti na vrlo visokoj razini sigurnosti i ako se dogodi jedan neovlašten upad u sustav, cijeli sustav se proglašava nesigurnim i problem se mora riješiti u što kraćem vremenu.

Popis literature

[1] Node.js, 2021. [Na internetu]. Dostupno: https://nodejs.org/en/ [pristupano 15.08.2021.]

[2] Express, 2021. [Na internetu]. Dostupno: https://expressjs.com/ [pristupano 15.08.2021.]

[3] Socket.IO, 2021. [Na internetu]. Dostupno: https://socket.io/ [pristupano 15.08.2021.]

[4] G. Sen, (22.01.2019.) „Whatsapp System Design: Chat Messaging Systems for Interviews“, *Youtube* [Video datoteka.] Dostupno: https://www.youtube.com/watch?v=vvhC64hQZMk [pristupano 15.08.2021.]

[5] React Native, 2021. [Na internetu]. Dostupno: https://reactnative.dev/ [pristupano 15.08.2021.]

[6] V. Jain, (12.01.2021.) „Developing a real-time secure chat application like WhatsApp & Signal with end-to-end encryption.“, Youtube [Video datoteka]. Dostupno: https://www.youtube.com/watch?v=gNbdgIznjhU [pristupano 15.08.2021.]

[7] E. S. Eze, „How to Build a Real-time Chat App With NodeJS, Socket.IO, and MongoDB“, [Blog post] 2019. [Na internetu]. Dostupno: https://dev.to/rexeze/how-to-build-a-real-time-chat-app-with-nodejs-socketio-and-mongodb-2kho [pristupano 15.08.2021.]

Popis slika

Popis tablica

Prilozi

1. Izvorni kod s poslužitelja za upravljanje razmjenom poruka