Mobilna aplikacija za dopisivanje s prijevodom

Brčina, Marijan

Sveučilište Jurja Dobrile u Puli

Fakultet informatike u Puli

Marijan Brčina

MOBILNA APLIKACIJA ZA DOPISIVANJE S PRIJEVODOM

Diplomski rad

Sveučilište Jurja Dobrile u Puli

Fakultet informatikeu Puli

Marijan Brčina

MOBILNA APLIKACIJA ZA DOPISIVANJE S PRIJEVODOM

Diplomski rad

JMBAG: 0303075733, redovni student

Studijski smjer: Informatika

Predmet: Mobilne aplikacije

Znanstveno područje: Društvene znanosti

Znanstveno polje: Informacijske i komunikacijske znanosti

Znanstvena grana: Informacijski sustavi i informatologija

Mentor: izv. prof. dr. sc. Siniša Sovilj

Pula, siječanj 2023.

IZJAVA O AKADEMSKOJ ČESTITOSTI

Ja, dolje potpisani **Marijan Brčina**, kandidat za **magistra informatike** ovime izjavljujem da je ovaj Diplomski rad rezultat isključivo mojega vlastitog rada, da se temelji na mojim istraživanjima te da se oslanja na objavljenu literaturu kao što to pokazuju korištene bilješke i bibliografija. Izjavljujem da niti jedan dio Završnog rada nije napisan na nedozvoljen način, odnosno da je prepisan iz kojega necitiranog rada, te da ikoji dio rada krši bilo čija autorska prava. Izjavljujem, također, da nijedan dio rada nije iskorišten za koji drugi rad pri bilo kojoj drugoj visokoškolskoj, znanstvenoj ili radnoj ustanovi.

Student

_____

U Puli,_____, 2023. godine

IZJAVA

o korištenju autorskog djela

Ja, **Marijan Brčina** dajem odobrenje Sveučilištu Jurja Dobrile u Puli, kao nositelju prava iskorištavanja, da moj završni rad pod nazivom **Mobilna aplikacija za dopisivanje s prijevodom** koristi na način da gore navedeno autorsko djelo, kao cjeloviti tekst trajno objavi u javnoj internetskoj bazi Sveučilišne knjižnice Sveučilišta Jurja Dobrile u Puli te kopira u javnu internetsku bazu završnih radova Nacionalne i sveučilišne knjižnice (stavljanje na raspolaganje javnosti), sve u skladu s Zakonom o autorskom pravu i drugim srodnim pravima i dobrom akademskom praksom, a radi promicanja otvorenoga, slobodnoga pristupa znanstvenim informacijama.
Za korištenje autorskog djela na gore navedeni način ne potražujem naknadu.

U Puli, _____ (datum)

Potpis

_____

SVEUČILIŠTE JURJA DOBRILE U PULI
FAKULTET INFORMATIKE

Pula, 27. ožujka 2022.

DIPLOMSKI ZADATAK

Pristupnik:	**Brčina Marijan (0303075733)**
Studij:	Sveučilišni diplomski studij Informatike
Naslov (hrv.):	**Mobilna aplikacija za dopisivanje s prijevodom**
Naslov (eng.):	Mobile messaging application with translation
Opis zadatka:	Zadatak diplomskog rada je izrada aplikacije za razmjenjivanje tekstualnih poruka s automatskim prijevodom poruka na odabrani jezik. Aplikacija se treba temeljiti na tehnologijama programskog jezika „Java/Kotlin“ koristeći alate „Android Studio“ te „Google Firebase“. Pri samom početku korištenja aplikacije bit će potrebna registracija korisnika, a za korisnike koji su već registrirani postojati će mogućnost klasične prijave u aplikaciju. Pri registraciji uz klasične podatke moći će se postaviti i slika profila korisnika. Sučelje aplikacije biti će slično ostalim aplikacija za razgovor u realnom vremenu kao što su Whatsapp, Viber i sl. uz postojanje opcije odabira jezika primljenih poruka. Korisnici će također imati uvid u status drugih korisnika s kojima komuniciraju. Istražiti način rada postojećih mobilnih aplikacija za razgovor te nakon toga uz izradu vlastite aplikacije pronaći način kako implementirati prevoditelj poruka u takvu vrstu mobilne aplikacije. Prednost ove aplikacije u odnosu na ostale slične aplikacije za razgovor je mogućnost automatskog prijevoda poruka na odabrani jezik što bi znatno olakšalo komunikaciju korisnika koji ne govore isti jezik.

Zadatak uručen pristupniku: 27. ožujka 2022.

Rok za predaju rada: 27. ožujka 2023.

Mentor:

Siniša Sovilj

doc.dr.sc. Siniša Sovilj

SADRŽAJ

1. UVOD 1

2. TEHNOLOGIJE KORIŠTENE PRI IZRADI MOBILNE APLIKACIJE 2

2.1 PROGRAMSKI JEZIK JAVA 2

2.2 ANDROID STUDIO 4

2.3 GOOGLE FIREBASE 7

3. MOTIVACIJA 11

3.1 SWOT ANALIZA APLIKACIJE 11

4. RAZRADA FUNKCIONALNOSTI 13

4.1 USE CASE DIAGRAM 13

4.2 USE CASE SEQUENCE DIAGRAM (DIJAGRAM TOKA) 15

4.3 USE CASE CLASS DIAGRAM (KLASNI DIJAGRAM) 18

4.3.1 PREGLED KLASA 19

4.3.2 VEZE U KLASNOM DIJAGRAMU 21

5. IMPLEMENTACIJA 23

5.1 POVEZIVANJE PROJEKTA SA GOOGLE FIREBASE BAZOM PODATAKA 25

5.2 REGISTRACIJA I PRIJAVA 27

5.2.1 BASE64 METODA KODIRANJA 31

5.3 POČETNO SUČELJE APLIKACIJE 32

5.4 PRIKAZ REGISTRIRANIH KORISNIKA 34

5.5 SUČELJE ZA RAZMJENJIVANJE PORUKA 37

5.6 STATUS AKTIVNOSTI KORISNIKA 39

5.7 PREVODITELJ TEKSTUALNIH PORUKA 42

6. KORISNIČKE UPUTE 47

6.1 PRIJAVA I REGISTRACIJA U APLIKACIJU 47

6.2 POČETNO SUČELJE APLIKACIJE 49

6.3 PRIKAZ REGISTRIRANIH KORISNIKA 50

6.4 SUČELJE ZA RAZGOVOR 51

7. ZAKLJUČAK 53

8. LITERATURA 54

9. POPIS SLIKA 55

10. SAŽETAK 57

1. UVOD

Opće poznato je da je komunikacija temelj međuljudskih odnosa, služi nam za razmjenu informacija pomoću dogovorenog sistema znakova, u današnjem svijetu svaki narod ima svoj sistem znakova, odnosno jezik. Kako bi ostvarili komunikaciju s nekom osobom potrebno je poznavanje istog govornog jezika. Od davnina su ljudi s različitih govornih područja imali problem s komunikacijom i sporazumijevanjem. Ljudska populacija je navedeni problem davno uvidjela te su isti pokušali riješiti tako što su djecu od malih nogu počeli učiti strane jezike. Pojavom računala i interneta dobili smo i različite alate koji nam pomažu da prevedemo i razumijemo strani jezik. Međutim možemo vidjeti da ništa od navedenog nije u potpunosti uklonilo ovaj problem te da ljudi i u 21. stoljeću imaju problem sa sporazumijevanjem ukoliko ne govore isti jezik. Svjedoci smo današnjeg vremena u kojem je cijeli svijet povezan, razni poslovi se ugovaraju i sklapaju isključivo preko interneta i na daljinu, u tom istom svijetu je vrlo bitno ostvariti jasnu komunikaciju sa svojim sugovornicima u realnom vremenu i bez puno odugovlačenja. Alati koji su trenutno dostupni a čija je svrha prijevod teksta su često komplicirani, neusklađeni i nepraktični za ljude kojima je pomoć u komunikaciji potrebna. Kada sagledamo cijelu situaciju možemo primijetiti da u današnje vrijeme postoji jako puno aplikacija i alata koji su komunikaciju na daljinu doveli skoro do savršenstva, međutim veliki nedostatak je što ni jedan od tih alata ne nudi brz i jednostavan prijevod poruke na željeni jezik. Svrha ovog diplomskog rada je stvaranje aplikacije koja će u realnom vremenu nuditi korisnicima sa različitim govornim jezicima nesmetanu i brzu komunikaciju. Smatram da bi jedna ovakva aplikacija uvelike olakšala razmjenu informacija, odnosno komunikaciju osobama koji imaju potrebu biti u kontaktu sa sugovornicima koji imaju različite govorne jezike.

2. TEHNOLOGIJE KORIŠTENE PRI IZRADI MOBILNE APLIKACIJE

Mobilna aplikacija za dopisivanje s prijevodom temelji se na tehnologijama programskog jezika „Java“ koristeći alate Android studio te Google Firebase.

2.1 PROGRAMSKI JEZIK JAVA

Programski jezik Java poznat je kao objektno orijentirani programski jezik, razlog tome je što je baziran na klasama i objektima. Razvijen je u tvrtki Sun Microsystems od strane Jamesa Goslinga, Patricka Naughtona i drugih inženjera koji su radili u spomenutoj kompaniji. Java je razvijena kao dio projekta Green, sam razvoj Jave je počeo 1991. godine a svoj konačni rad inženjeri tvrtke Sun Microsystems objavili su u studenom 1995. godine, kada je službeno programski jezik Java i ugledao svjetlo dana.

Prije samog razvoja programskog jezika Java inženjeri su postavili 5 jasnih ciljeva koje moraju ostvariti, a to su:

- Jezik mora biti jednostavan, objektno orijentiran i prepoznatljiv
- Jezik mora biti siguran i robustan
- Jezik mora konceptualno neutralan i prenosiv
- Jezik mora imati dobre performanse
- Jezik mora biti jednostavan za čitanje

Java je moderan programski jezik koji se koristi u praksi i jedan je od najpopularnijih jezika na svijetu. Prvotno je napravljen sa jednim ciljem, a to je da sa što manje dodataka, omogući programerima razvoj aplikacija bilo koje vrste. Java je programski jezik u kojem možete isprogramirati gotovo sve. Usudili bi se reći da je uz C# najkompletniji jezik na svijetu. I prije nego počnete spominjati Python, C, C++ i slične jezike, sjetite se da su oni ograničeni. Možda su brži od Jave, u određenim poljima su

svakako i lakši za korištenje, no Java ima najširi spektar mogućnosti i definitivno je programski jezik koji je programerima broj jedan, procjene i izvješća o broju korisnika kreću se od gotovo 7 do preko 10 milijuna. Od četiri programera na svijetu, jedan programira u ovom jeziku.

Programski jezik java se smatra jednim od najmodernijih i najpopularnijih programski jezika današnjeg vremena. Osnovna zamisao inženjera koji su stvarali Javu je bila stvaranje programskog jezika koji će uz što manje dodataka omogućiti programerima razvijanje aplikacija bilo koje vrste. Programski jezici kao što su Python, C, C++ i slični su često brži od Jave, također znaju biti i lakši za korištenje, međutim imaju mnogo manji spektar mogućnosti od Jave te je iz navedenog razloga Java programski jezik koji je programerima broj jedan. Procjenjuje se da programski jezik Java koristi od 7 do preko 10 milijuna ljudi širom svijeta, o popularnosti Jave nam govori i podatak da svaki četvrti programer koristi Java programski jezik za svoj rad.

Cilj same Jave je postati najkorištenija tehnologija na svijetu te omogućiti kompatibilnost sa svim mogućim uređajima. Sa sigurnošću možemo reći da je Java na pravom putu jer već sada više od milijardu uređaja na svijetu pokreće Java programski jezik, također je bitno naglasiti da ova brojka iz dana u dan raste velikom brzinom te je sukladno tome tvrtka Java sve bliža svome cilju.

Java ima izuzetno široku primjenu na trenutnom tržištu, konkretno prednjači tamo gdje je brzina razvoja programskog jezika bitnija od brzine rada samog sustava koji se programira. Programski jezik Java je zapravo inspiriran programskim jezikom C, međutim Java zahvaljujući VM-u i hermetički zatvorenom okolišu u kojem se svaki program izvršava, pruža veći stupanj sigurnosti i pouzdanosti. Također je bitno naglasiti da se u Javi brže razvija program te je prisutan manji broj grešaka.

Upravo je iz gore spomenutih razloga programski jezik Java izuzetno popularan za razvoj mobilnih aplikacija ali i za programe financijskih kompanija. Java se smatra osnovnim jezikom za razvoj Googleovog sustava Android.

Slika 1. Prikaz široke primjene programskog jezika Java

Izvor: https://www.edureka.co/blog/what-is-java/ (11.01.2023.)

2.2 ANDROID STUDIO

Android Studio je službeno integrirano razvojno okruženje za Googleov operativni sustav Android , temeljeno je na JetBrainsovom IntelliJ IDEA softveru i dizajnirano posebno za Android razvoj. Dostupno je za preuzimanje na operativnim sustavima Windows , macOS i Linux . Android Studio je došao kao zamjena za Eclipse Android Development Tools (E-ADT) koji je do tada bio primarno razvojno okruženje za razvoj Android aplikacija.

Android Studio je najavljen 16. svibnja 2013. na Google-ovoj konferenciji. Bio je u fazi pregleda ranog pristupa počevši od verzije 0.1 u svibnju 2013., zatim je ušao u beta fazu počevši od verzije 0.8 koja je objavljena u lipnju 2014. Prva stabilna verzija objavljena je u prosincu 2014. godine, počevši od verzije 1.0. Krajem 2015. Google je

ukinuo podršku za Eclipse ADT, čime je Android Studio postao jedino službeno podržano integrirano razvojno okruženje za Android razvoj.

Bitna razlika između Android Studia i Eclipse-a je ta što je Eclipse projekte organizirao u radni prostor koji se odredio pri pokretanju. Takav način rada je ograničavajući jer dopušta da se koriste samo projekti iz određenog radnog prostora. U slučaju potrebe za radom s projektima iz drugog radnog prostora, moramo ugasiti Eclipse i prilikom ponovnog pokretanja prijeći u drugi radni prostor. Android Studio taj problem rješava korištenjem modula. Moduli mogu biti svaki dio nekog projekta, na primjer jedan modul je aplikacija na kojoj se upravo radi, drugi modul je integrirani SDK. Svaki od modula može imati svoje Gradle bulid datoteke i zahtijevati svoje pakete za ovisnost.

Još jedna od prednosti Android Studia u odnosu na Eclipse se odnosi na performanse i stabilnost integriranog razvojnog okruženja. Eclipse je IDE koji je razvijen u svrhu razvoja aplikacija u Javi. Zahtijeva puno RAM-a i CPU-a kako bi radio bez greške. Kada se u Eclipseu razvijaju aplikacije za Android, često se dogodi da se IDE sruši jer nije predviđen za tu vrstu razvoja. Zna se dogoditi da nakon nekoliko sati neprekidnog rada u Eclipseu, IDE prestane raditi pa ga se treba ponovo pokrenuti. Za razliku od Eclipse-a, Android Studio je, kada se počeo koristiti, bio još u beta verziji razvoja, to je donijelo probleme sa stabilnošću razvojne okoline, u međuvremenu su navedeni problemi popravljeni ali još uvijek nisu u potpunosti uklonjeni.

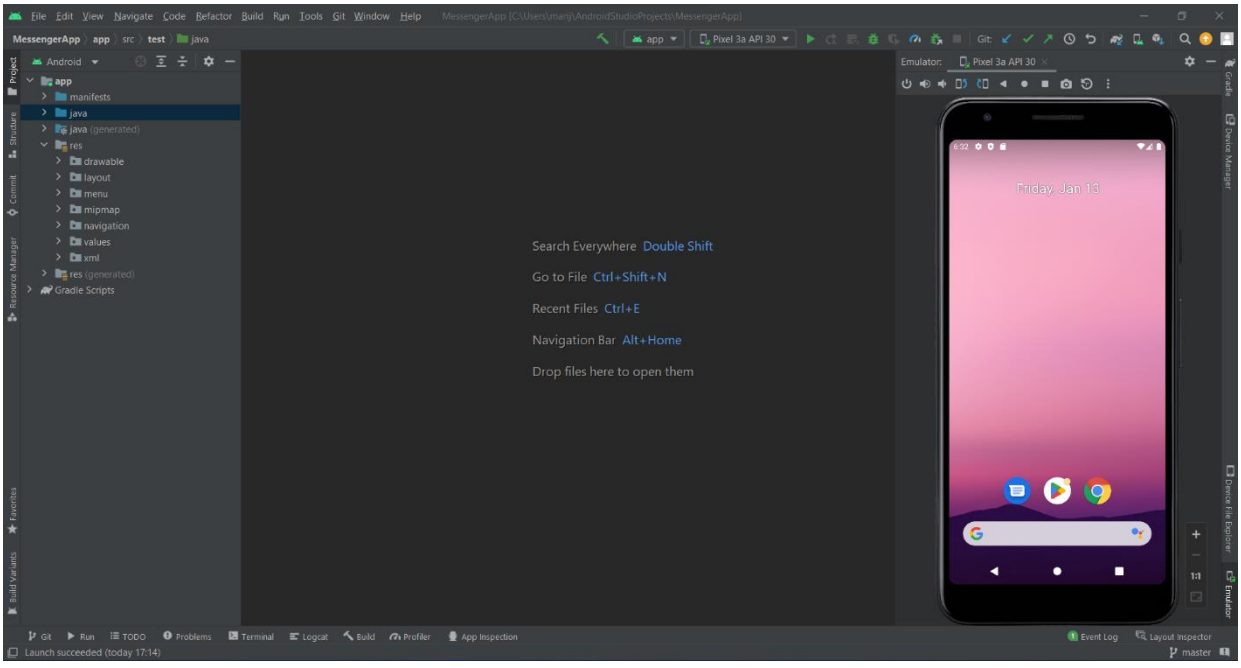
Android Studio stvoren je na način da nam uveliko pomaže i olakšava programiranje Android aplikacija. Jedna od glavnih prednosti ovog alata je njegova autocomplete opcija koja nam kada krenemo pisati ime funkcije ili nekog drugog elementa automatski ponudi sve mogućnosti koje bi mogle odgovarati traženom nazivu.

Još jedna od prednosti Android Studia je što ne moramo znati u kojem su paketu klase ili metode koje želimo, jednostavno kliknemo mišem na objekt koji javlja grešku i kombinacijom tipki alt+Enter dobijemo popis paketa koje bi trebali uvesti, dalje ih jednostavno uvezemo odabirom na željene pakete.

Značajke koje trenutno pruža Android studio:

- Podrška za izgradnju temeljena na Gradleu
- Refactoring i brzi popravci specifični za Android
- Lint alati za hvatanje performansi, upotrebljivosti, kompatibilnosti verzija i drugih problema
- ProGuard integracije i mogućnosti potpisivanja aplikacija
- Čarobnjaci temeljeni na predlošcima za stvaranje uobičajenih Android dizajna i komponenti
- Bogati uređivač izgleda koji korisnicima omogućuje povlačenje i ispuštanje komponenti korisničkog sučelja, mogućnost pregleda izgleda na više konfiguracija zaslona
- Podrška za izradu Android Wear aplikacija
- Ugrađena podrška za Google Cloud Platform, koja omogućuje integraciju s Firebase Cloud Messaging (ranije Google Cloud Messaging) i Google App Engine
- Android virtualni uređaj (emulator) za pokretanje i otklanjanje pogrešaka u aplikacijama u Android Studiu

Slika 2. Sučelje alata Android Studio



Izvor: autor

2.3 GOOGLE FIREBASE

Firebase se smatra skalabilnom bazom podataka koja radi u stvarnom vremenu (eng. Real time database). Firebase platforma je osnovana 2011. godine te je istu kupila tvrtka Google 2014. godine. Već u listopadu 2018. godine Firebase nudi 18 različitih usluga i proizvoda, Firebase u 2018. godini također broji i 1 500 000 aplikacija koje koriste usluge ove platforme. Firebase baza podataka se razlikuje od klasičnih baza podataka po tome što koristi relacijski model, odnosno Firebase je modeliran po sistemu pohranjivanja dokumenata (eng. Document store).

Firebase platforma je zapravo integrirana u oblaku. Svaki spremljeni podatak se pohranjuje kao JSON te se sinkronizira u realnom vremenu sa svakim povezanim klijentom. Klijenti mogu biti implementirani na više platformi, na primjer Android, IOS, Unity, JavaScript, C++. Svaki od spomenutih klijenata dijeli jednu instancu baze podataka što omogućava automatsko ažuriranje najnovijih podataka u stvarnom vremenu. Još jedna prednost Firebase baze podataka je ta što ne koristi klasične HTTP zahtjeve nego koristi sinkronizaciju podataka, dakle radi na principu da se pri svakoj promjeni pojedinog podataka šalje ažurirani podatak na svaki od povezanih uređaja, sve to se događa u nekoliko milisekundi i neovisno o jeziku kojim je implementiram povezani uređaj. Prednost Firebase platforme je i ta što aplikacijska rješenja ostaju u funkciji i kada uređaj nije povezan na internetsku mrežu. Ovaj slučaj se izvodi na način da se upisani podaci ili naredbe spremaju na lokalni disk samog uređaja te se nakon ponovnog uspostavljanja internetske veze klijentskom uređaju šalju svi podaci koji su ostali na disku dok je uređaj bio u izvanmrežnom načinu rada.

Firebase bazi podataka je moguće pristupiti pomoću Internet preglednika ili direktno s mobilnog uređaja, dakle nema potrebe za poslužiteljskim aplikacijama. Firebase također nudi i sigurnost i provjeru valjanosti podataka, isti su dostupni putem Firebase sigurnosnih pravila baze podataka. Spomenuta pravila se temelje na izrazima koji se izvršavaju tijekom čitanja ili zapisivanja podataka. Firebase je licenciran za komercijalnu upotrebu ali u svojoj ponudi ima i besplatan paket koji se naziva „Blaze“ (plamen), spomenuti paket može bez problema podržati većinu podatkovnih zahtjeva aplikacije.

Firebase baza podataka u svojoj ponudi ima i provjeru autentičnosti pomoću Firebase identifikacije korisnika. Firebase platforma ima mogućnost izgradnje bogatih kolaborativnih aplikacija omogućavajući siguran pristup podacima izravno iz koda na strani klijenta

Firebase baza podataka nudi fleksibilna sigurnosna pravila što omogućuje samim programerima da definiraju sigurnosna pravila po svojoj želji. Ova mogućnost uvelike pomaže programerima jer na taj način mogu definirati kada se podaci mogu čitati ili pisati. Programeri također imaju i mogućnost definiranja prava pristupa pojedinim podacima, navedenu opciju programerima omogućuje Firebase identifikacija korisnika.

Firebase baza podataka se smatra nerelacijskom bazom podataka (eng. NoSQL-Structured Query Language), iz tog razloga ima različite optimizacije i funkcionalnosti u odnosu na relacijeske baze podataka.

Firebase platforma je osmišljena i dizajnirana za implementaciju operacija čije izvršavanje traje kratko, odnosno operacija koje se mogu izvršiti brzo. Ovakav način rada daje mogućnost korištenja aplikacija velikom broju korisnika bez ugrožavanja odaziva. Iz tog razloga je veoma bitno kojim sve podacima može pristupiti određeni korisnik, također je bitno i kako korisnik može pristupiti tim podacima. Nakon što se riješe ova dva pitanja po tome treba strukturirati podatke u bazi podataka.

Firebase alati korišteni tijekom izrade aplikacije:

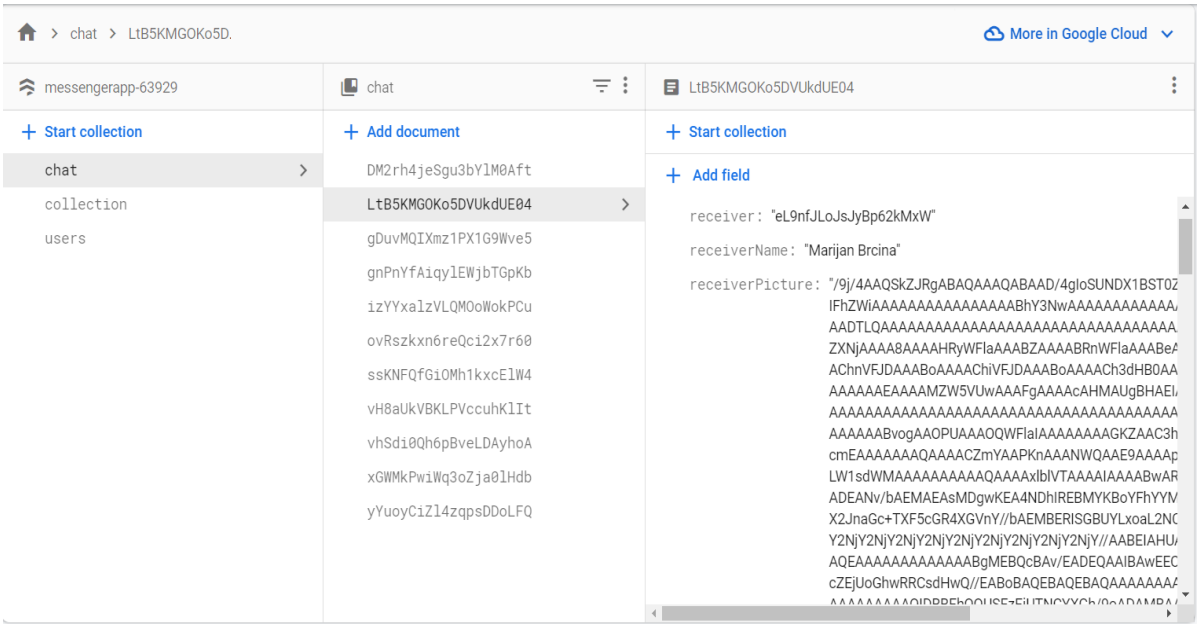
Realtime Database: Ovaj alat nam omogućuje spremanje, brisanje i ažuriranje podataka u stvarnom vremenu. Kao što sam ranije naveo, svaki podatak se sprema u JSON formatu u obliku objekta, svaki objekt sadržava kolekciju podataka u formatu parova ključ-vrijednost. Prednost spomenutog alata je što ne postoji potreba za nabavom i pokretanjem osobne poslužiteljske opreme. Realtime baza ima i svoje mane a one su upiti i sortiranja tijekom rada sa bazom.

Firebase Auth: omogućuje nam autentifikaciju korisnika koristeći kod samo na klijentskoj strani. Također nudi i metode autentifikacije preko društvenih mreža te upravljanje i autentifikaciju korisnika pomoću emaila i lozinke. Ovaj alat nam pomaže i pri provjeri valjanosti adrese korisnika te nam omogućuje ponovno postavljenje lozinke preko elektroničke pošte.

Firebase Storage: alat koji nam daje mogućnost sigurnog spremanja i preuzimanja raznih datoteka kao što su slike, video uradci, audio sadržaj i sl.

Firebase Cloud Messaging: višeplatformski alat koji nam daje mogućnost slanja poruka i obavijesti operacijskim sustavima Android, IOS te web aplikacijama. Sadrži API-e (eng. Application programming interface) kao što su slanje obavijesti, slanje grupnih obavijesti, uključivanje i isključivanje grupnih obavijesti.

Slika 3. Prikaz spremljenih podataka u Google Firebase bazi podataka



Izvor: autor

Slika 4. Podaci o upitima i zapisima prema Firebase bazi podataka

Izvor: autor

3. MOTIVACIJA

U današnjem svijetu gdje smo svi povezani i gdje u svakom trenutku možemo ostvariti komunikaciju sa svakom pojedinom osobom, vrlo je bitno da se možemo brzo i jednostavno sporazumjeti s tom istom osobom. Kako na primjer samo u Europi imamo 234 različita jezika postoji vrlo velika vjerojatnost da, ukoliko oba govornika ne poznaju engleski jezik, nećete moći ostvariti normalnu komunikaciju ni sa samim Europljanima. Upravo iz ovog razloga sam došao na ideju da izradim jednostavnu mobilnu aplikaciju koja će u stvarnom vremenu prevoditi poruke na odabrani jezik. Kao što je već spomenuto, ciljano tržište mobilne aplikacije za dopisivanje s prijevodom su korisnici koji žele ostvariti brzu i jednostavnu komunikaciju s osobama koje ne govore isti jezik.

3.1 SWOT ANALIZA APLIKACIJE

Prednosti

Kao što je već spomenuto, smatram da je osnovna prednost ove aplikacije brzina i jednostavnost korištenja. Aplikacija svakako nije na razini popularnih mobilnih aplikacija za razgovor što se tiče funkcionalnosti i samog dizajna, bitno je naglasiti da prvenstveni cilj ove aplikacije nije ni bio nadmašiti konkurentske aplikacije nego stvoriti nešto novo, nešto što takve aplikacije nemaju. Još jedna od prednosti za mnoge korisnike je jednostavnost korištenja aplikacije, aplikacija je fokusirana samo na razgovor te ne opterećuje korisnika s možda njemu nepotrebnim informacijama kao što su razne priče, statusi i slične stvari koje su moderne aplikacije za razgovor implementirale u svoj sustav.

Slabosti

Kao slabost aplikacije naveo bih nedovoljno stručno znanje pomoću kojeg bi nadmašio konkurentske aplikacije te samu aplikaciju gurnuo u sami vrh aplikacija za razgovor po popularnosti. Još jedna od slabosti je i nedovoljno vremena za kvalitetno razvijanje svake komponente aplikacije. Kao slabost aplikacije se može navesti i moje osobno nedovoljno znanje o marketingu i reklamiranju proizvoda, u ovom slučaju aplikacije, smatram da bi stručna osoba dosta bolja reklamirala i prodala navedeni proizvod. Nedostatak aplikacije je i taj što je sučelje trenutno samo na hrvatskom jeziku te ne postoji mogućnost promjene na neki drugi jezik.

Prilike

Na današnjem tržištu postoji vrlo malo aplikacija koje nudi automatski prijevod poruka, smatram da je ovaj način komuniciranja još uvijek inovativan i neistražen, samim time smatram i da na navedenom području postoji dosta prostora za napredak i poboljšanje ove vrste komunikacije. Kako bi aplikaciju učinili popularnijom bilo bi dobro proširiti aplikaciju i na druge platforme. U aplikaciju bi se također mogla implementirati i umjetna inteligencija, npr. slanje glasovne poruke koja bi se primatelju ispisala kao tekst na njemu odabranom jeziku i slične stvari.

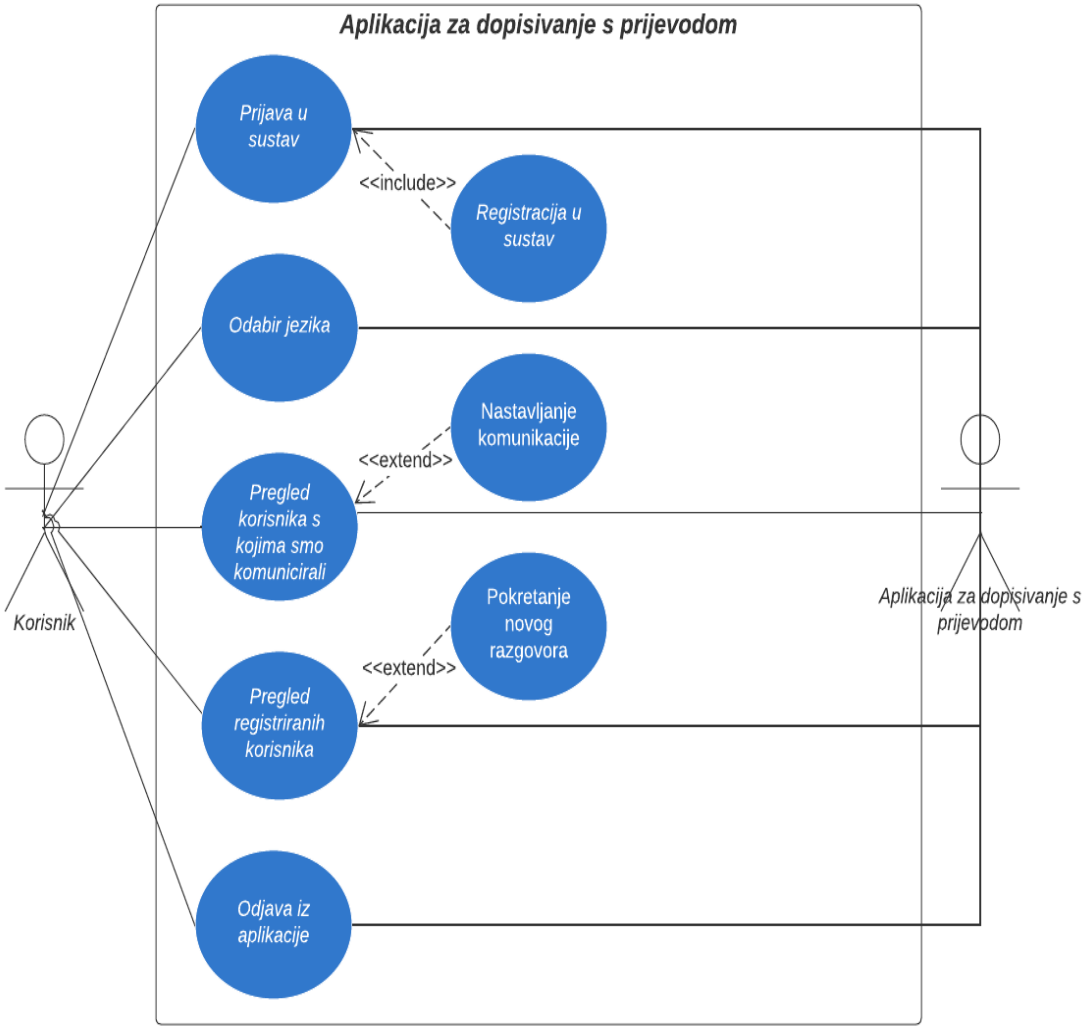
Prijetnje

Jedna od prijetnji ovoj aplikaciji je ta što bi se u budućnosti vrlo brzo na tržištu mogla pojaviti slična programska rješenja. Kao prijetnju možemo navesti i nedovoljno znanje u reklamiranju same aplikacije, pitanje je vremena kada će osvanuti neka ozbiljna konkurentska aplikacija iza koje će stajati tim dobro uigranih stručnjaka koji će pomoću marketinga, znanja i vještina vrlo lako nadmašiti jednostavne konkurentske aplikacije.

4. RAZRADA FUNKCIONALNOSTI

4.1 USE CASE DIAGRAM

Slika 5. Use Case dijagram



Izvor: autor

Pogledom na Use case dijagram možemo vidjeti da aplikacija nudi korisniku 5 osnovnih mogućnosti iz kojih proizlaze još neke pod mogućnosti od kojih smo neke primorani koristiti, točnije one koji su povezani „include“ vezom, dok pod mogućnosti koje su povezane „extend“ vezom možemo ali i ne moramo koristiti.

U gore prikazanom Use case dijagramu imamo dva sudionika, korisnika aplikacija te samu aplikaciju, odnosno nekog pružatelja usluga.

Prvi korak pri korištenju same aplikacije je korak prijave u aplikaciju, odnosno ukoliko niste registrirani obavezno je napraviti registraciju jer bez iste nećete biti u mogućnosti pristupiti samoj aplikaciji, to je ujedno i razlog zašto je registracija u sustav povezana „include“ vezom sa prijavom u sustav.

Druga aktivnost je „Odabir jezika“, u navedenoj aktivnost odabiremo jedan od 6 ponuđenih jezika na koji želimo da nam se prevode poruke koje nam šalje drugi sudionik razgovora. Ponuđeni jezici su Engleski, Njemački, Francuski, Talijanski, Ruski i Španjolski.

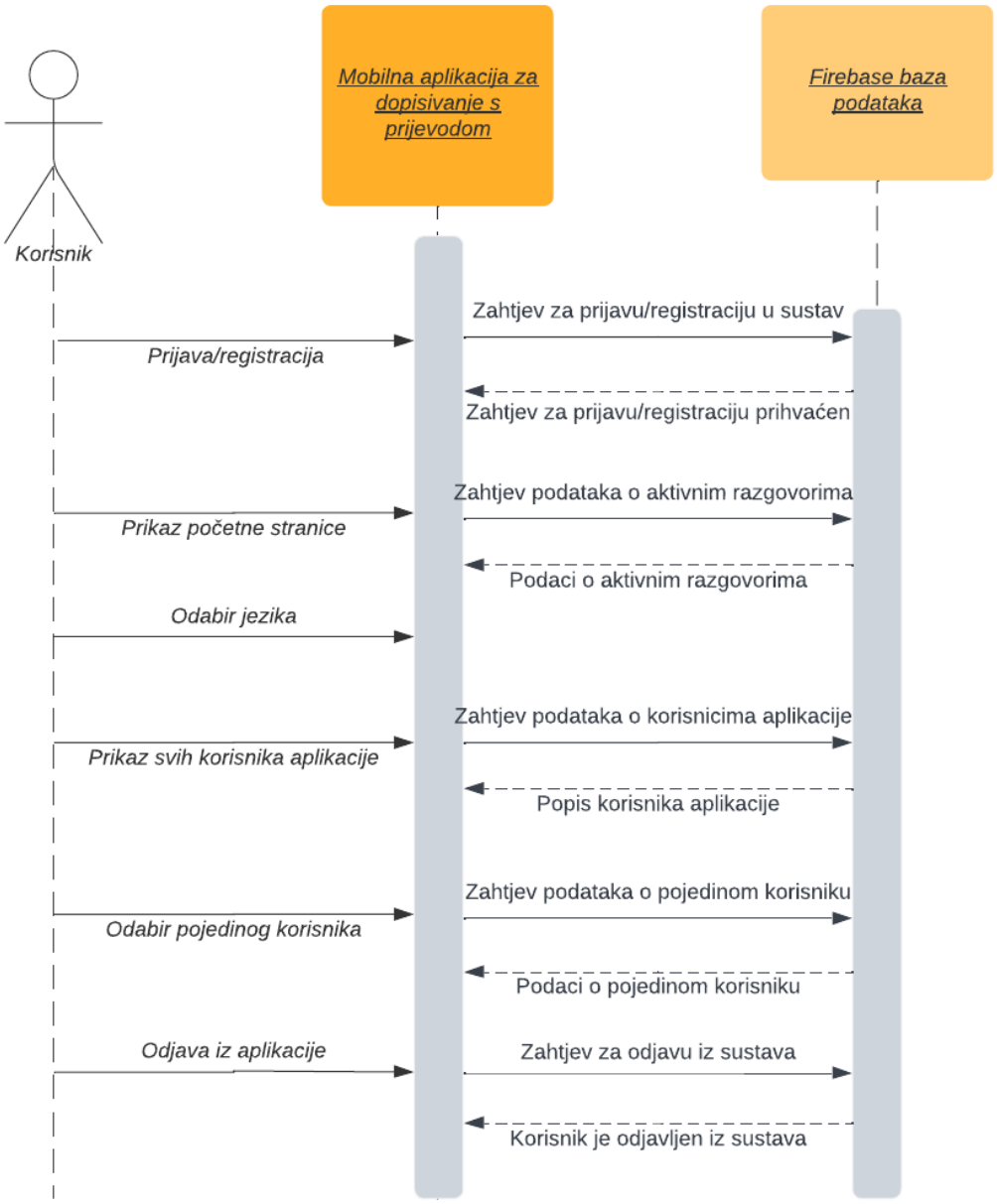
Treća aktivnost je „Pregled korisnika s kojima smo komunicirali“, to je aktivnost koja nam se prikaže nakon uspješne prijave ili registracije u aplikaciju. U Navedenoj aktivnosti aplikacija korisniku prikazuje sve korisnike s kojim je do tada vodio razgovor. Kao što možemo vidjeti i na prikazanom Use Case dijagramu, korisnik ima mogućnost nastaviti razgovor s nekim od ponuđenih sugovornika, to će učiniti tako što će kliknuti na željenog sugovornika.

Sljedeća aktivnost je „Pregled registriranih korisnika“, odabirom na gumb u donjem lijevom kutu početnog sučelja prikazati će se svi korisnici naše aplikacije. Korisnik ima mogućnost povratka na početno sučelje aplikacije ili pokretanje novog razgovora, novi razgovor će pokrenuti tako što će jednostavno kliknuti na jednog od ponuđenih korisnika s kojim želi razgovarati.

Posljednja aktivnost na našem Use Case dijagramu je „Odjava iz aplikacije“. Kako bi se odjavili iz aplikacije potrebno je odabrati gumb koji se nalazi u donjem desnom kutu početnog sučelja aplikacije. Nakon odjave iz aplikacije imamo mogućnost ponovo se prijaviti u aplikaciju ili registrirati novog korisnika.

4.2 USE CASE SEQUENCE DIAGRAM (DIJAGRAM TOKA)

Slika 6. Sequence diagram (Dijagram toka)



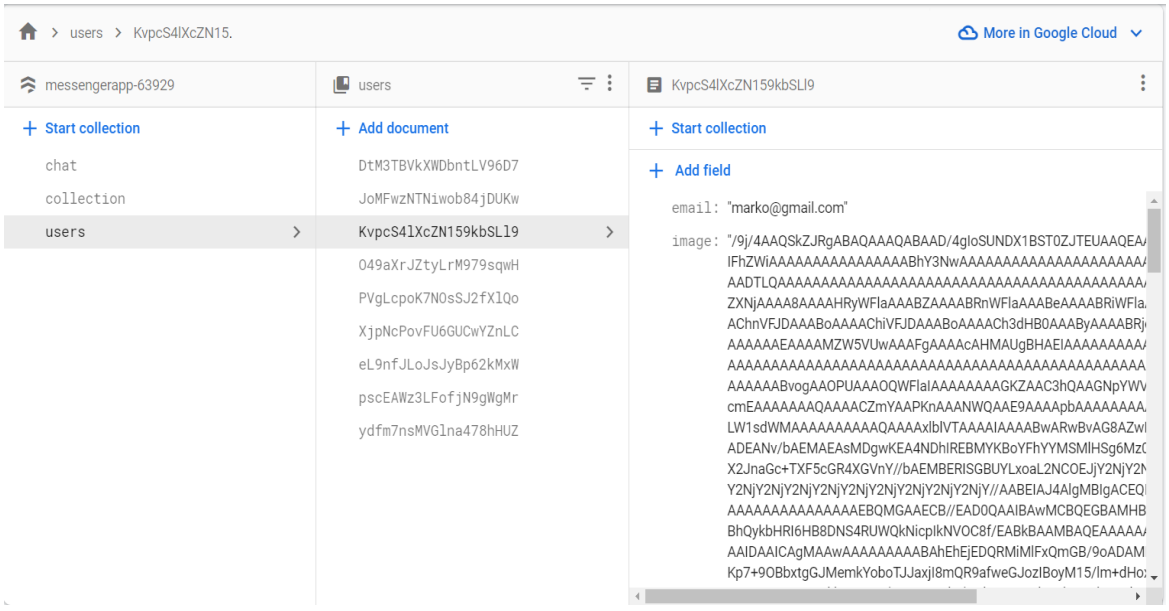
Izvor: autor

Na slici 6 možemo vidjeti sequence diagram ili dijagram toka, isti prikazuje uključene procese i slijed poruka koje se razmjenjuju između procesa potrebnih za provođenje funkcionalnosti.

U našem dijagramu slijeda imamo jednog sudionika, on je korisnik aplikacije. Također imamo dva objekta: jedan od njih je sama aplikacija dok je drugi Firebase baza podataka. Za svaki od navedenih objekata imamo i takozvani „lifeline“ ili liniju života, linija života nam prikazuje koliko se dugo prikazani objekt koristi kroz aplikaciju. Iz priložene slike možemo vidjeti da se linije života oba objekta protežu kroz čitavo vrijeme korištenja aplikacije, odnosno aplikacija ne može raditi bez prikazanih objekata.

Kada korisnik pokrene aplikaciju samim tim se pokreće i linija života same aplikacije. Sljedeći korak je prijava ili registracija u sustav, kada korisnik izvrši prijavu ili registraciju u sustav počinje linija života Firebase baze podataka. Prema bazi podataka se u tom trenutku šalje zahtjev za prijavu/registraciju u sustav. Ukoliko je sve ispravno prošlo baza podataka šalje povratnu informaciju da je korisnik uspješno prijavljen ili registriran te ga automatski prebacuje na početnu stranicu aplikacije. Nakon što se prijavimo nije moguće posjetiti stranicu za prijavu ili registraciju dok korisnik ne bude odjavljen iz aplikacije. Na slici 7 možemo vidjeti kako u bazi podataka izgledaju registrirani korisnici, odnosno možemo vidjeti pojedine podatke o korisnicima koji su registrirani u aplikaciju i koji zapravo imaju pravo pristupa aplikaciji bez registracije, odnosno sve što trebaju proći je prijava u sustav.

Slika 7. Registrirani korisnici



Izvor: autor

Zatim dolazimo do početne stranice aplikacije gdje nam aplikacija prikazuje sve korisnike s kojima smo do sada razmjenjivali poruke. Kako bi nam aplikacija prikazala naveden korisnike potrebno je poslati upit prema bazi podataka kako bi nam ista vratila tražene informacije. Nakon što baza podataka vrati informacije o korisnicima s kojima smo već razgovarali isti se prikazuju na početnom sučelju aplikacije

Sljedeća aktivnost koju korisnik treba odabrati je željeni jezik, isti će vrlo jednostavno odabrati pomoću padajućeg izbornika. Kao što možemo uočiti iz dijagrama toka kod ove aktivnosti aplikacija nema potrebu komunicirati s Firebase bazom podataka.

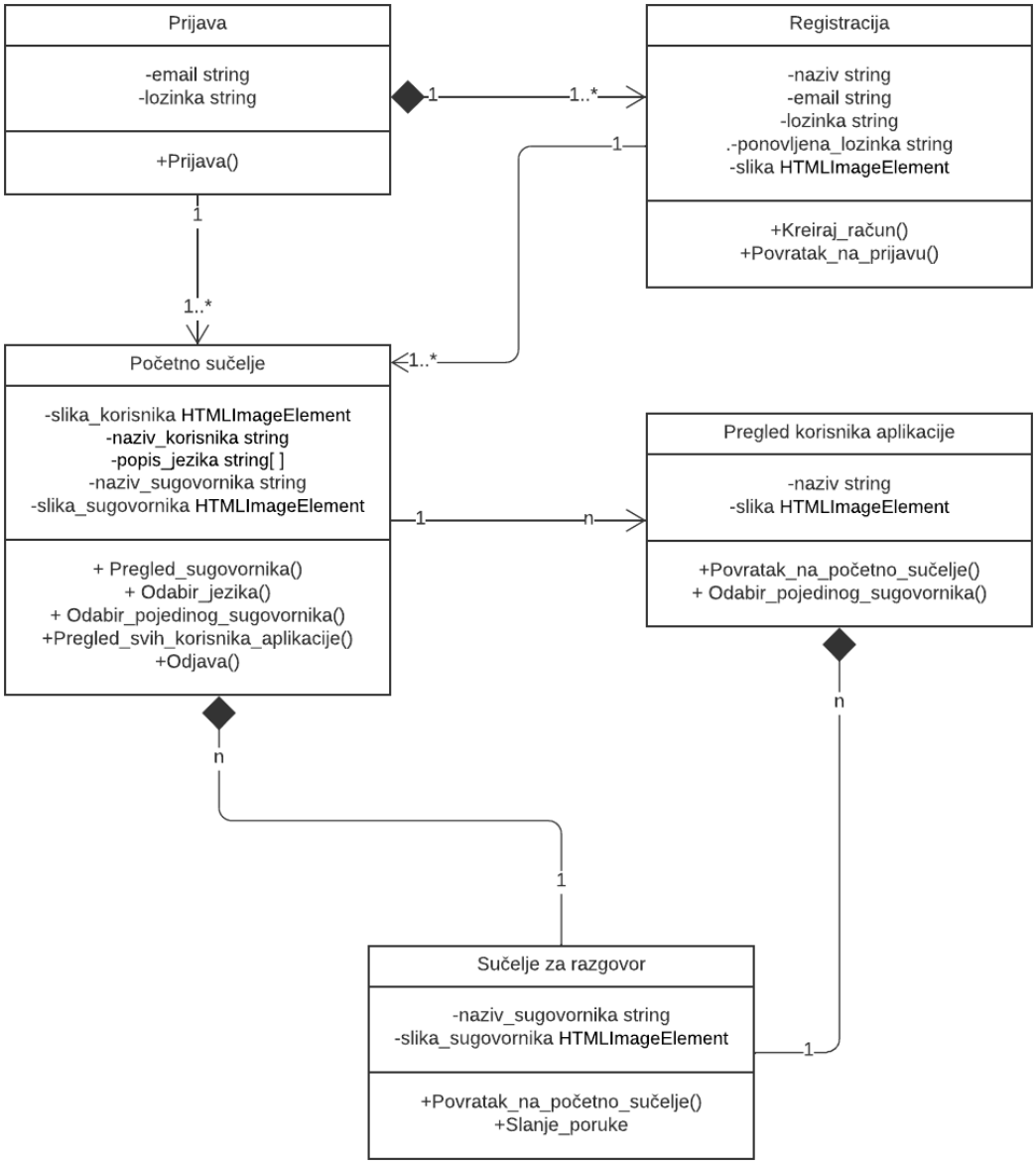
Nakon toga slijedi proces „Prikaz svih korisnika aplikacije“, nakon pokretanja spomenutog procesa aplikacija šalje bazi podataka zahtjev o slanju informacija koji su joj potrebni za prikaz registriranih korisnika aplikacije, zatim Firebase vraća povratnu informaciju u kojoj aplikaciji šalje podatke o registriranim korisnicima.

U sučelju gdje su prikazani registrirani korisnici korisnik ima mogućnost pokretanja novog razgovora s nekim od ponuđenih korisnika. Nakon što odabere jednog od korisnika aplikacija šalje bazi podataka zahtjev za slanje informacija o odabranom korisniku, zatim baza podataka vraća aplikaciji tražene podatke kako bi se moglo kreirati sučelje za razgovor.

Posljednji proces u našem Sequence dijagramu je odjava iz aplikacije gdje korisnik pritiskom na gumb koji se nalazi u donjem desnom kutu početnog sučelja aplikacije šalje zahtjev prema Firebase-u za odjavu korisnika iz baze podataka. Kao rezultat tog procesa dolazimo na stranicu za prijavu i registraciju te korisnik nije u mogućnosti pristupiti aplikaciji dok se ponovo ne prijavi u istu.

4.3 USE CASE CLASS DIAGRAM (KLASNI DIJAGRAM)

Slika 8. Klasni dijagram



Izvor: autor

Kao što možemo vidjeti, u našem klasnom dijagramu imamo 5 klasa, svaka od tih klasa ima svoje atribute te operacije koje se mogu izvršavati unutar te klase. Također možemo vidjeti da je svaka klasa povezana nekom vezom sa drugom klasom, te veze zapravo prikazuju odnose između pojedinih klasa.

4.3.1 PREGLED KLASA

Prva klasa je klasa **„Prijava“**, to je ujedno i sučelje koje nam se prvo prikaže pri pokretanju aplikacije, unutar navede klase imamo sljedeće atribute:

- -email string
- -lozinka string

Ispred svakog atributa stoji znak „-“ što označava da su privatni.

Unutar ove klase imamo operaciju:

- +Prijava()

Ispred svake operacije stoji znak „+“ što označava da su one javne.

Sljedeća klasa je klasa **„Registracija“**. U navedenoj klasi imamo sljedeće atribute:

- -naziv string
- -email string
- -lozinka string
- -ponovljena_lozinka string
- -slika HTMLImageElement

Operacije koje predstavljaju ovu klasu su:

- +Kreiraj_račun()
- +Povratak_na_prijavu()

Sljedeća klasa je klasa **„Početno sučelje“** koju čine sljedeći atributi:

- -slika_korisnika HTMLImageElement
- -naziv_korisnika string
- -popis_jezika string[ ]
- -naziv_sugovornika string
- -slika_sugovornika HTMLImageElement

Operacije koje ova klasa sadrži su:

- + Pregled_sugovornika()
- + Odabir_jezika()
- + Odabir_pojedinog_sugovornika()
- +Pregled_svih_korisnika_aplikacije()
- +Odjava()

Dalje imamo klasu **„Pregled korisnika aplikacije“,** njeni atributi su:

- -naziv string
- -slika HTMLImageElement

Operacije klase „Pregled korisnika aplikacije“ su:

- +Povratak_na_početno_sučelje()
- + Odabir_pojedinog_sugovornika()

Posljednja, peta klasa je klasa **„Sučelje za razgovor“,** ova klasa sadrži sljedeće atribute:

- -naziv_sugovornika string
- -slika_sugovornika HTMLImageElement

operacije unutar ove klase su:

- +Povratak_na_početno_sučelje()
- +Slanje_poruke

4.3.2 VEZE U KLASNOM DIJAGRAMU

Krenuti ćemo od klase „Prijava“, ona je sa klasom „Registracija“ povezana vezom kompozicije što znači da je klasa „Prijava“ ovisna o klasi „Registracija“ što bi značilo da se ne možemo prijaviti u sustav ukoliko prethodno nismo napravili registraciju. Također iz ove veze možemo pročitati da se jedan korisnik može više puta registrirati u aplikaciju ali sa istim emailom je to moguće napraviti samo jednom. Također vidimo da je klasa „Prijava“ povezana vezom asocijacije sa klasom „Početno sučelje“, iz iste možemo pročitati da je prijavu u sustav moguće izvršiti jednom ili više puta dok u isto vrijeme u sustavu može biti prijavljen samo jedan korisnik sa jedinstvenom email adresom.

Sljedeća klasa je klasa „Registracija“ , ona je povezana vezom kompozicije sa klasom „Prijava, spomenuta veza je već opisana u gornjem dijelu teksta. Klasa „Registracija“ je također povezana sa klasom „Početno sučelje“ vezom asocijacije kao i klasa „Prijava“ dakle registraciju u sustav je moguće izvršiti jednom ili više puta međutim s druge strane vidimo da je s jednom email adresom to moguće napraviti isključivo jednom dakle u isto vrijeme u sustavu može biti samo jedan korisnik sa jedinstvenom email adresom.

Dalje dolazimo do klase „Početno sučelje“, prvo ćemo opisati vezu ove klase sa klasom „Pregled korisnika aplikacije“. Spomenute klase su povezane vezom asocijacije gdje vidimo da imamo mogućnost pregledavanja n broja korisnika, dakle određenog broja korisnika koji je promjenjiv te je zato označen sa n.

Klasa „Početno sučelje“ je povezana i sa klasom „Sučelje za razgovor“, Ove dvije klase su povezane vezom kompozicije, što znači da klasa „Sučelje za razgovor“ ovisi o klasi „Početno sučelje“ jer ukoliko ne bi bilo ni jednog korisnika za odabrati u klasi „Početno sučelje“ samim tim ne bi imali s kim ni komunicirati te nam klasa „Sučelje za razgovor“ ne bi bila potrebna. Iz ove veze također možemo vidjeti da korisnik ima pregled n broja sugovornika koje može odabrati za razgovor ali u isto vrijeme može komunicirati samo s jednim sugovornikom.

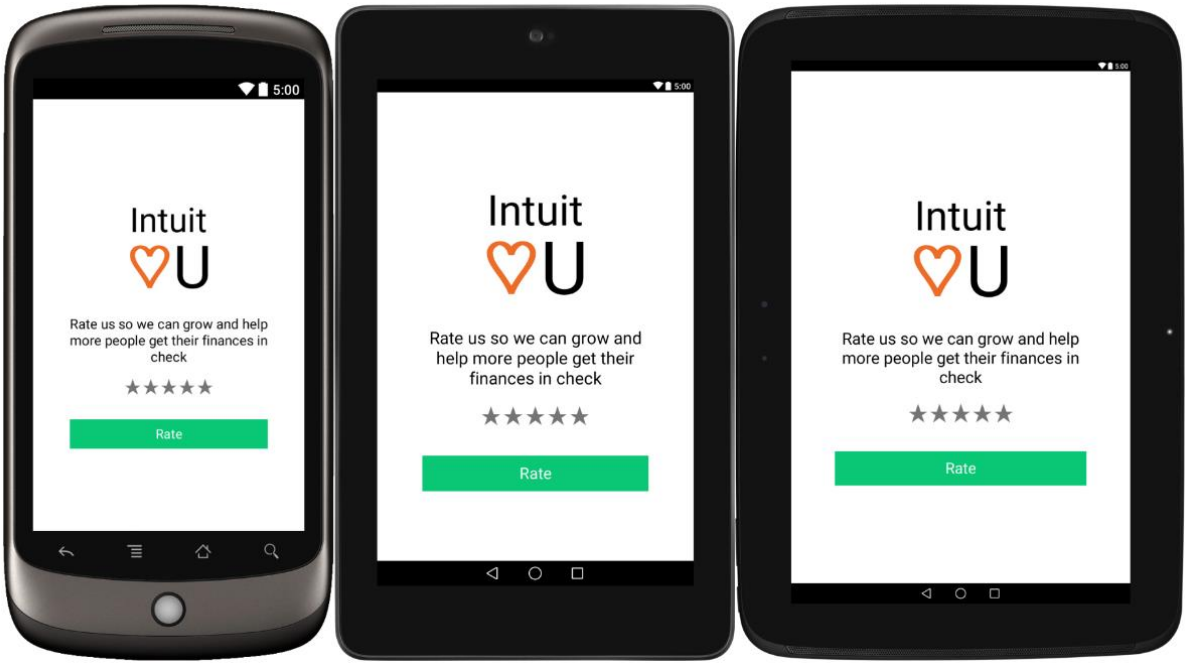
Posljednja veza u našem klasnom dijagramu je veza između klase „Pregled korisnika aplikacije“ i „Sučelje za razgovor“. Ove dvije klase su povezane identičnom vezom kao i prethodno dvije objašnjene klase. Dakle odnos među klasama je takav da klasa

„Sučelje za razgovor“ ovisi o klasi „Pregled korisnika aplikacije“ iz razloga što kada ne bi bilo korisnika u aplikaciji klasa „Sučelje za razgovor“ nam ne bi bila potrebna iz razloga što ne bi imali s kim komunicirati. Kao i kod prethodno objašnjenog slučaja, korisnik ima mogućnost odabira između n broja sugovornika ali u isto vrijeme se može dopisivati samo s jednim od tih sugovornika.

5. IMPLEMENTACIJA

Prije samog početka izrade aplikacije vrlo je važno pobliže se upoznati s alatima koje ćemo koristiti pri implementaciji. Prvi korak je bio postavljanje projekta, nakon kreiranja novog projekta i uklanjanja nepotrebnog dijela koda u projekt su uključeni SDP i SSP alati. SDP (a scalable size unit) je alat koji nam omogućuje da se aplikacija identično prikazuje na svakom android uređaju, neovisno o veličini zaslona uređaja. SSP (a scalable size unit for texts) je također alat sa sličnom svrhom samo što SSP prilagođava veličinu teksta zaslonu android uređaja. Na slici 9 možemo vidjeti kako zapravo alati SDP i SSP utječu na prikaz pojedinog sučelja na android uređajima koji imaju različite dimenzije zaslona.

Slika 9. Primjer upotrebe SDP i SSP alata



Izvor: https://www.uplabs.com/posts/sdp-a-scalable-size-unit (15.01.2023.)

Navedeni alati se vrlo jednostavno implementiraju u projekt, dovoljno je samo u datoteku „build.gradle (:app)“ pod „dependencies“ dodati dvije linije koda koje možemo vidjeti na slici 10.

Slika 10. Implementiranje alata SDP i SSP u projekt

```
//SDP, SSP
implementation 'com.intuit.sdp:sdp-android:1.0.6'
implementation 'com.intuit.ssp:ssp-android:1.0.6'
```

Izvor: autor

Prije početka izrade aplikacije u projekt je na isti način uključena i biblioteka „Rounded ImageView“ koja nam omogućuje promjene nad slikama, odnosno omogućuje nam da slike budu ovalne, s zaobljenim kutovima i sl. Navedena biblioteka je korištena kod prikazivanja slike profila korisnika i sugovornika. Kao što možemo vidjeti na slici 11 ovaj alat je također u projekt implementiran jednostavnim dodavanjem jedne linije koda u „dependencies“.

Slika 11. Implementiranje alata Rounded ImageView

```
//Rounded ImageView
implementation 'com.makeramen:roundedimageview:2.3.0'
```

Izvor: autor

Sljedeće na redu je uključivanje značajke „View Binding“, ova značajka nam omogućuje lakše pisanje koda koji je u interakciji s view-ovima. Jednom kada je povezivanje pogleda omogućeno u modulu, ono generira klasu za svaku XML datoteku prisutnu u tom modulu. Instanca klase sadrži izravne reference na sve view-ove koji imaju ID u odgovarajućem pogledu. Naredba za uključivanje značajke „View Binding“ je prikazana na slici 12.

Slika 12. Uključivanje značajke „View Binding“

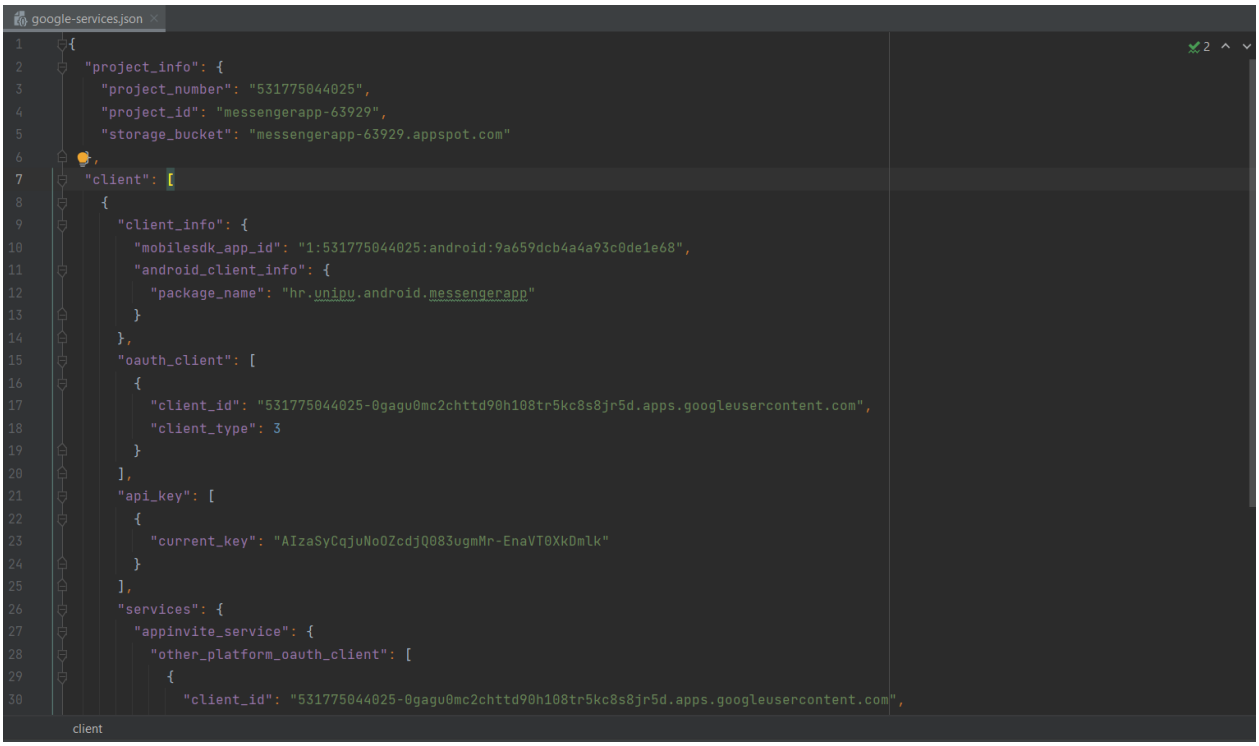
```
buildFeatures {
    viewBinding true
}
```

Izvor: autor

5.1 POVEZIVANJE PROJEKTA SA GOOGLE FIREBASE BAZOM PODATAKA

Prije samog početka korištenja Firebase-ove baze podataka potrebno se ulogirati ili registrirati s Google računom u Firebase te zatim kreirati novi projekt te ga povezati s aplikacijom. Google Firebase nam pri kreiranju novog projekta kreira google-services.json datoteku koju je potrebno umetnuti u projekt, ista je prikaza na slici 13.

Slika 13. google-services.json datoteka

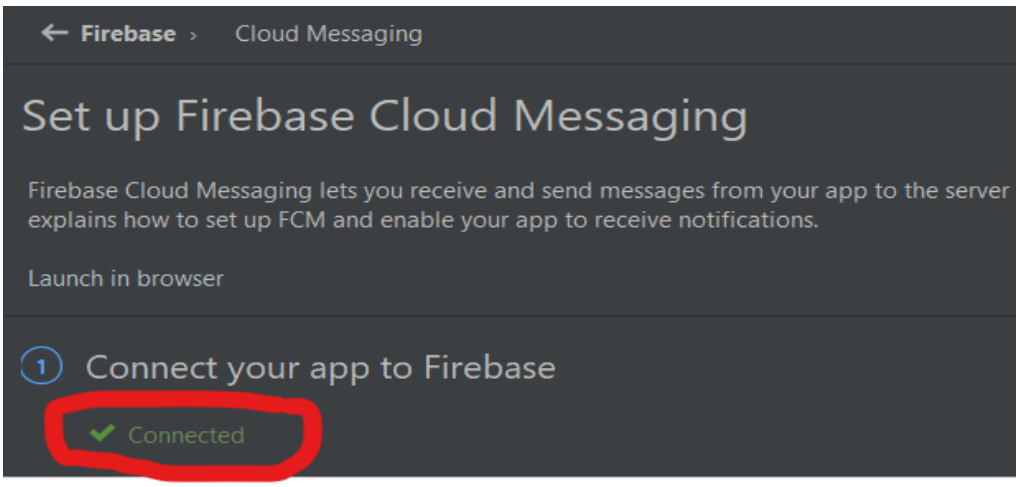


Izvor: autor

Preostalo je još implementirati „google-services“ u dependecies te u „plugins“. Svi ovi koraci su detaljno objašnjeni kroz kreiranje novog projekta putem Google Firebase-a. Nakon što smo obavili sve potrebne korake možemo u alatnoj traci kliknuti na „Tools“ te zatim na „Firebase“, dalje odabiremo „Cloud Messaging“ nakon toga je potrebno kliknuti na „Set up Firebase Cloud Messaging“.

Dolazimo do sučelja koji nam, ukoliko smo uspješno izvršili sve korake povezivanja aplikacije sa Firebase bazom podataka, prikazuje da je aplikacija povezana sa Firebase-om. Ako pogledamo sliku 14. možemo vidjeti kako treba izgledati opisano sučelje ako je aplikacija uspješno povezana.

Slika 14. Uspješno povezivanje aplikacije s alatom Google Firebase

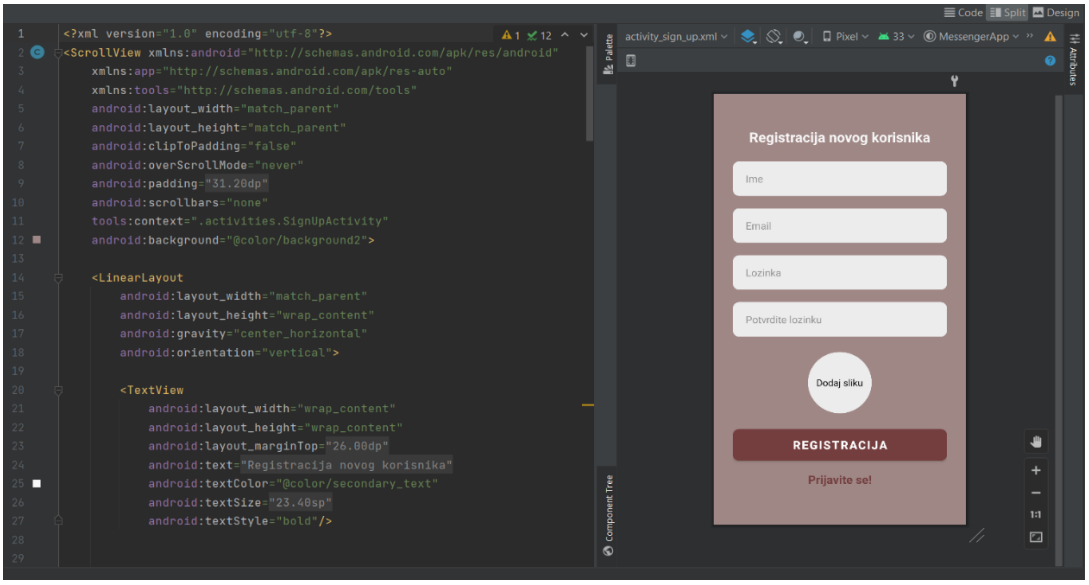


Izvor: autor

5.2 REGISTRACIJA I PRIJAVA

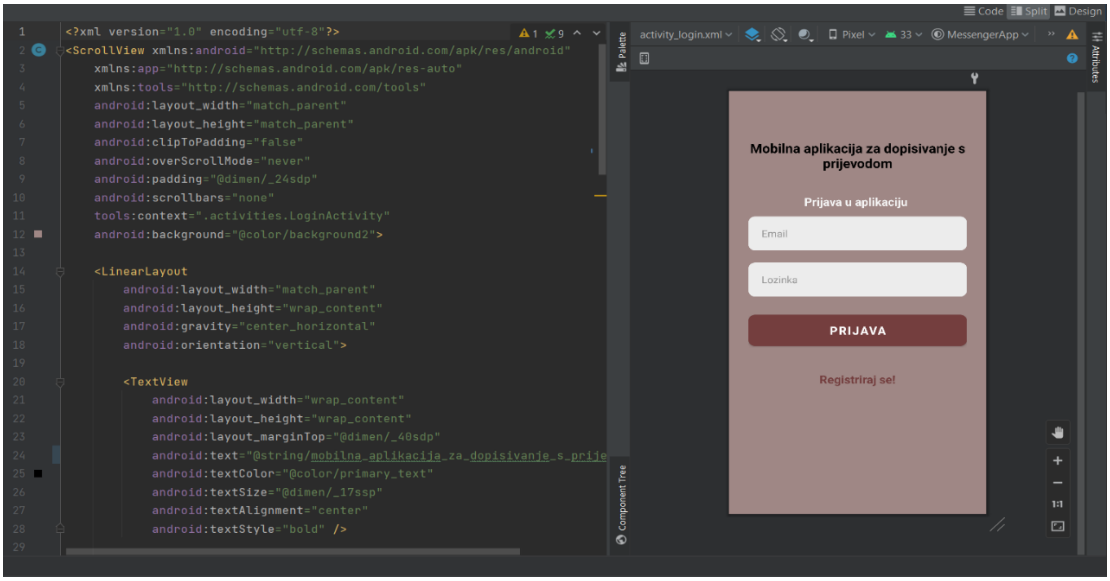
Na samom početku izrade aplikacije potrebno je osmisliti i implementirati način registracije i prijave u istu, ponajprije su napravljene xml datoteke koje predstavljaju izgled sučelja za prijavu i registraciju. Spomenute xml datoteke su prikazane na slikama 15. i 16.

Slika 15. Izgled xml datoteke za registraciju u aplikaciju



Izvor: autor

Slika 16. Izgled xml datoteke za prijavu u aplikaciju



Izvor: autor

Kako bi provjerili jesu li ispravno uneseni svi podaci pri registraciji novog korisnika napravljena je funkcija koja provjera ispravnost unesenih podataka, ukoliko neki od traženih podataka nije unesen neće biti moguće izvršiti registraciju te će se korisniku ispisati poruka za točno određeni podatak. Funkcija koja provjerava ispravnost podatka prikazana je na slici 17.

Slika 17. Provjera ispravnosti podataka pri registraciji u aplikaciju

```
private Boolean isValidSignup() {
    if (binding.inputName.getText().toString().trim().isEmpty()) {
        showToast("Unesite ime");
        return false;
    } else if (binding.inputEmail.getText().toString().trim().isEmpty()) {
        showToast("Unesite email");
        return false;
    } else if (!Patterns.EMAIL_ADDRESS.matcher(binding.inputEmail.getText().toString()).matches()) {
        showToast("Unijeli ste nevažeću email adresu");
        return false;
    } else if (binding.inputPassword.getText().toString().trim().isEmpty()) {
        showToast("Unesite lozinku");
        return false;
    } else if (binding.inputConfirmPassword.getText().toString().trim().isEmpty()) {
        showToast("Potvrdite svoju lozinku");
        return false;
    } else if (!binding.inputPassword.getText().toString().equals(binding.inputConfirmPassword.getText().toString())) {
        showToast("Lozinka i ponovljena lozinka se ne podudaraju");
        return false;
    } else if (encodedImage == null){
        showToast("Postavite sliku profila");
        return false;
    } else {
        return true;
    }
}
```

Izvor: autor

Za prijavu u aplikaciju je također potrebno provjeriti ispravnost podataka. Ispravnost podataka kod prijave u aplikaciju je također provjerena jednostavnom funkcijom koja je prikazana na slici 18.

Slika 18. Provjera unesenih podataka pri prijavi u aplikaciju

```
private Boolean validation(){
    if (binding.inputEmail.getText().toString().trim().isEmpty()) {
        showToast("Unesite email adresu");
        return false;
    } else if (!Patterns.EMAIL_ADDRESS.matcher(binding.inputEmail.getText().toString()).matches()) {
        showToast("Email adresa nije ispravna");
        return false;
    }else if (binding.inputPassword.getText().toString().trim().isEmpty()) {
        showToast("Unesite lozinku");
        return false;
    }else {
        return true;
    }
}
```

Izvor: autor

Na posljetku je preostalo implementirati funkcije za registraciju i prijavu u aplikaciju. Na slici 19 vidimo funkciju za registraciju koja unesene podatke sprema u Firebase bazu podataka pod kolekciju s nazivom „USERS“.

Slika 19. Implementacija registracije u aplikaciju

```
private void signUp(){
    loading( isLoading: true);
    FirebaseFirestore database = FirebaseFirestore.getInstance();
    HashMap<String, Object> user = new HashMap<>();
    user.put(Constants.NAME, binding.inputName.getText().toString());
    user.put(Constants.EMAIL, binding.inputEmail.getText().toString());
    user.put(Constants.PASSWORD, binding.inputPassword.getText().toString());
    user.put(Constants.IMAGE, encodedImage);
    database.collection(Constants.USERS) CollectionReference
            .add(user) Task<DocumentReference>
            .addOnSuccessListener(documentReference -> {
                loading( isLoading: false);
                preferenceManager.putBoolean(Constants.IS_SIGNED_IN, true);
                preferenceManager.putString(Constants.USER_ID, documentReference.getId());
                preferenceManager.putString(Constants.NAME, binding.inputName.getText().toString());
                preferenceManager.putString(Constants.IMAGE,encodedImage);
                Intent intent = new Intent(getApplicationContext(),MainActivity.class);
                intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK | Intent.FLAG_ACTIVITY_CLEAR_TASK);
                startActivity(intent);
            })
            .addOnFailureListener(exception -> {
                loading( isLoading: false);
                showToast(exception.getMessage());
            });
}
```

Izvor: autor

Funkcija za prijavu u aplikaciju, koju možemo vidjeti na slici 20, radi na način da čita kolekciju u bazi podataka gdje su spremljeni registrirani korisnici te pronalazi korisnika kod kojeg se podudaraju email i lozinka sa unesenim emailom i lozinkom pri prijavi u aplikaciju. Ukoliko funkcija ne pronađe slučaj gdje se email i lozinka ne podudaraju s nekim od istih podataka u bazi ispasti će poruku da nije moguće izvršiti prijavu.

Slika 20. Implementacija prijave u aplikaciju

```
private void login(){
    loading(true);
    FirebaseFirestore database = FirebaseFirestore.getInstance();
    database.collection(Constants.USERS) CollectionReference
            .whereEqualTo(Constants.EMAIL, binding.inputEmail.getText().toString()) Query
            .whereEqualTo(Constants.PASSWORD, binding.inputPassword.getText().toString())
            .get() Task<QuerySnapshot>
            .addOnCompleteListener(task -> {
                if (task.isSuccessful() && task.getResult() != null && task.getResult().getDocuments().size() >0) {
                    DocumentSnapshot documentSnapshot = task.getResult().getDocuments().get(0);
                    preferenceManager.putBoolean(Constants.IS_SIGNED_IN, true);
                    preferenceManager.putString(Constants.USER_ID, documentSnapshot.getId());
                    preferenceManager.putString(Constants.NAME, documentSnapshot.getString(Constants.NAME));
                    preferenceManager.putString(Constants.IMAGE, documentSnapshot.getString(Constants.IMAGE));
                    Intent intent = new Intent(getApplicationContext(), MainActivity.class);
                    intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK | Intent.FLAG_ACTIVITY_CLEAR_TASK);
                    startActivity(intent);
                } else {
                    loading(false);
                    showToast("Nije moguće izvršiti prijavu, provjerite točnost podataka");
                }
            });
}
```

Izvor: autor

5.2.1 BASE64 METODA KODIRANJA

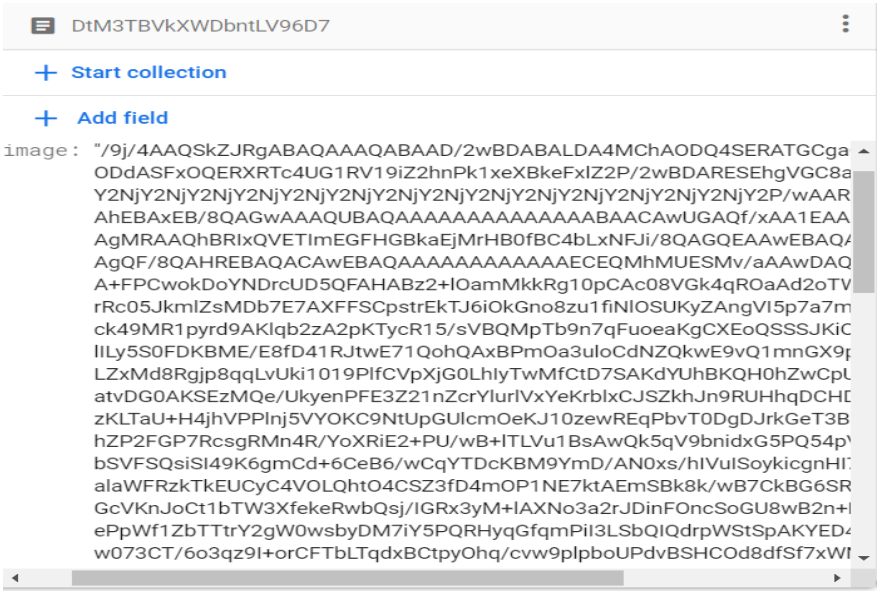
Pri spremanju slike korisnika u bazu podataka korištena je Base64 metoda kodiranja, razlog korištenja ove metode je taj što Firebase baza podataka podržava samo tekstualni sadržaj. Dakle pomoću Base64 metode kodiranja pretvaramo sliku u tekst i obrnuto. Točnije, u računalnom programiranju , Base64 je skupina shema kodiranja binarnog u tekst koje predstavljaju binarne podatke u nizu od 24 bita koji se mogu predstaviti s četiri 6-bitne Base64 znamenke. Primjer implementiranja Base64 metode kodiranja u aplikaciju je prikazan na slici 21. dok na slici 22. možemo vidjeti kako izgleda fotografija koja je pomoću Base64 metode kodiranja pretvorena u niz znakova te spremljena u Firebase bazu podataka.

Slika 21. Implementacija Base64 metode kodiranja

```
private String encodeImage(Bitmap bitmap){
    int previewWidth = 150;
    int previewHeight = bitmap.getHeight() * previewWidth / bitmap.getWidth();
    Bitmap previewBitmap = Bitmap.createScaledBitmap(bitmap, previewWidth, previewHeight, filter: false);
    ByteArrayOutputStream byteArrayOutputStream = new ByteArrayOutputStream();
    previewBitmap.compress(Bitmap.CompressFormat.JPEG, quality: 50, byteArrayOutputStream);
    byte[] bytes = byteArrayOutputStream.toByteArray();
    return Base64.encodeToString(bytes, Base64.DEFAULT);
}
```

Izvor: autor

Slika 22. Spremanje fotografije u Google Firebase bazu podataka

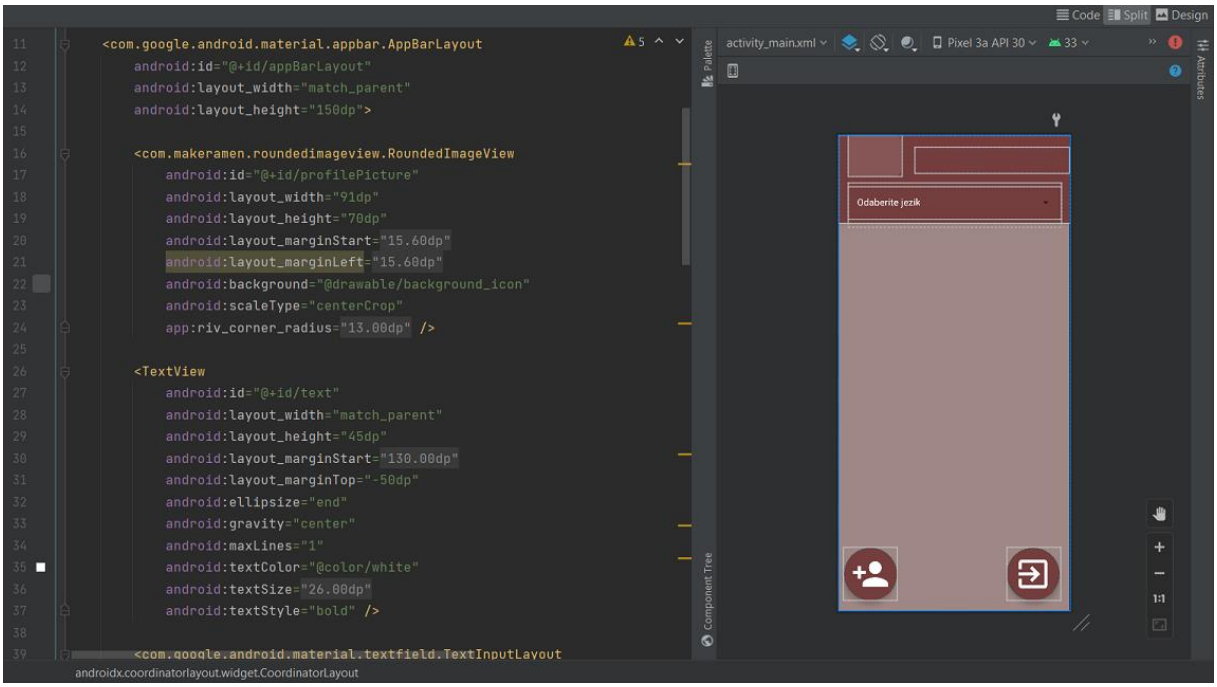


Izvor: autor

5.3 POČETNO SUČELJE APLIKACIJE

Početno sučelje aplikacije sastoji se od slike korisnika koji je ulogiran, korisnikovog imena, padajućeg izbornika u kojem biramo jezik na koji želimo da nam se poruke prevode, dalje u donjem desnom kutu imamo gumb koji nas vodi na sve korisnike aplikacije, isti ćemo kliknuti ukoliko želimo pokrenuti novi razgovor te nam preostaje još gumb koji se nalazi u donjem desnom kutu, isti ćemo kliknuti ukoliko se želimo odjaviti iz aplikacije. Sve navedene komponente možemo vidjeti na slici 23. koja se nalazi ispod teksta, na slici je zapravo prikazana xml datoteka početnog sučelja aplikacije.

Slika 23. Izgled xml datoteke početnog sučelja aplikacije



Izvor: autor

Polja koja su u xml datoteci prazna, odnosno polje gdje treba biti slika i ime korisnika, se popunjavaju preko funkcije „userDetails“ koja je prikaza na slici 24. Iz slike 24 možemo vidjeti da nam je opet Base64 metoda kodiranja bila potrebna kako bi uspješno prikazali sliku korisnika.

Slika 24. Funkcija koja prikazuje ime korisnika i sliku profila

```
private void userDetails(){
    binding.text.setText(preferenceManager.getString(Constants.NAME));
    byte[] bytes = Base64.decode(preferenceManager.getString(Constants.IMAGE), android.util.Base64.DEFAULT);
    Bitmap bitmap = BitmapFactory.decodeByteArray(bytes, offset: 0, bytes.length);
    binding.profilePicture.setImageBitmap(bitmap);
}
```

Izvor: autor

Na početnom sučelju aplikacije nam se još prikazuju i već pokrenuti razgovori, odnosno sugovornici s kojima smo već vodili tekstualne razgovore putem ove aplikacije. Kako bi prikazali spomenute sugovornike bilo je potrebno izraditi jednu jednostavnu xml datoteku koje će prikazivati profilnu sliku i ime sugovornika. Kako bi se sugovornici ispravno prikazivali na početnom sučelju aplikacije moramo imati i funkciju „chats“ koja dohvaća spomenute korisnike iz baze podataka. Navedena funkcija je prikazana na slici 25.

Slika 25. Dohvaćanje podataka o sugovornicima

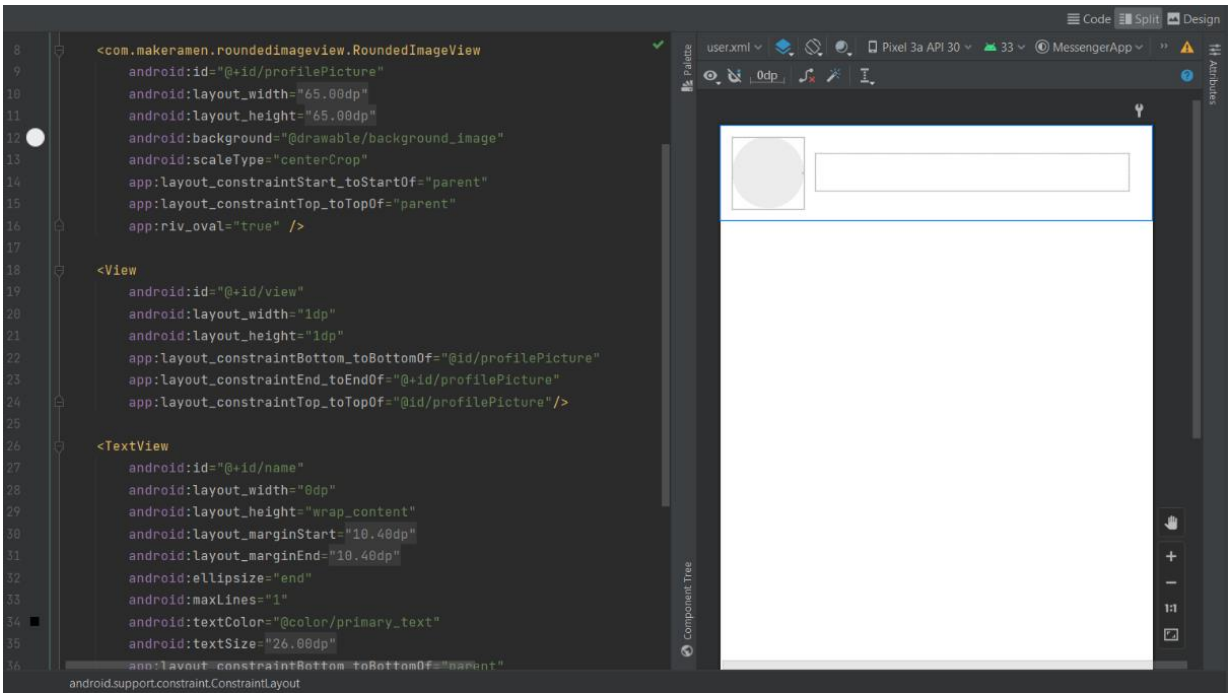
```
private void chats(){
    database.collection(Constants.COLLECTION_CHAT)
            .whereEqualTo(Constants.SENDER, preferenceManager.getString(Constants.USER_ID))
            .addSnapshotListener(eventListener);
    database.collection(Constants.COLLECTION_CHAT)
            .whereEqualTo(Constants.RECEIVER, preferenceManager.getString(Constants.USER_ID))
            .addSnapshotListener(eventListener);
}
```

Izvor: autor

5.4 PRIKAZ REGISTRIRANIH KORISNIKA

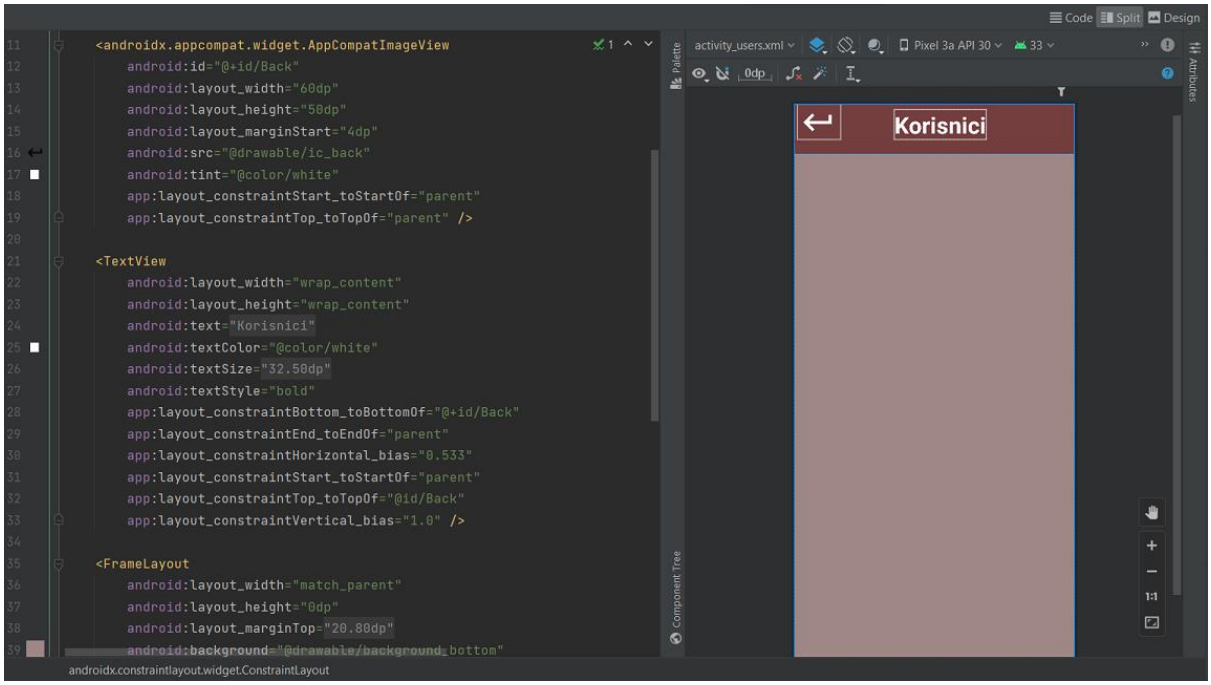
Kako bi korisnik imao uvid u osobe s kojima ima mogućnost komunicirati putem aplikacije, odnosno u sve registrirane korisnike isti su prikazani u sučelju koje je izrađeno u te svrhe. Kao i za svako drugo, tako su i za ovo sučelje prvotno izrađene xml datoteke koje predstavljaju izgled samog sučelja. Za ovo sučelje su nam bile potrebne dvije xml datoteke, jedna koja prikazuje informacije o pojedinom korisniku aplikacije, prikazana na slici 26. te druga u kojoj se nalazi lista koje prikazuje sve korisnike koji su trenutno registrirani u aplikaciju, slika 27.

Slika 26. Izgled xml datoteke koja predstavlja pojedinog korisnika



Izvor: autor

Slika 27. Izgled xml datoteke pomoću koje se prikazuju registrirani korisnici



Izvor: autor

Nakon što smo izradili xml datoteke koje nam određuju izgled sučelja potrebno je napraviti funkciju koja će iz baze čitati podatke koji su potrebni za ispravno prikazivanje spomenutog sučelja. Kao što možemo vidjeti na slici 28. u aplikaciju je implementirana funkcija koja iz kolekcije „USERS“ koja se nalazi u Firebase-u dohvaća sve registrirane korisnike, odnosno dohvaća ime te fotografiju svakog registriranog korisnika, te ih prikazuje u spomenutom sučelju aplikacije.

Slika 28. Funkcija za dohvaćanje podataka o registriranim korisnicima

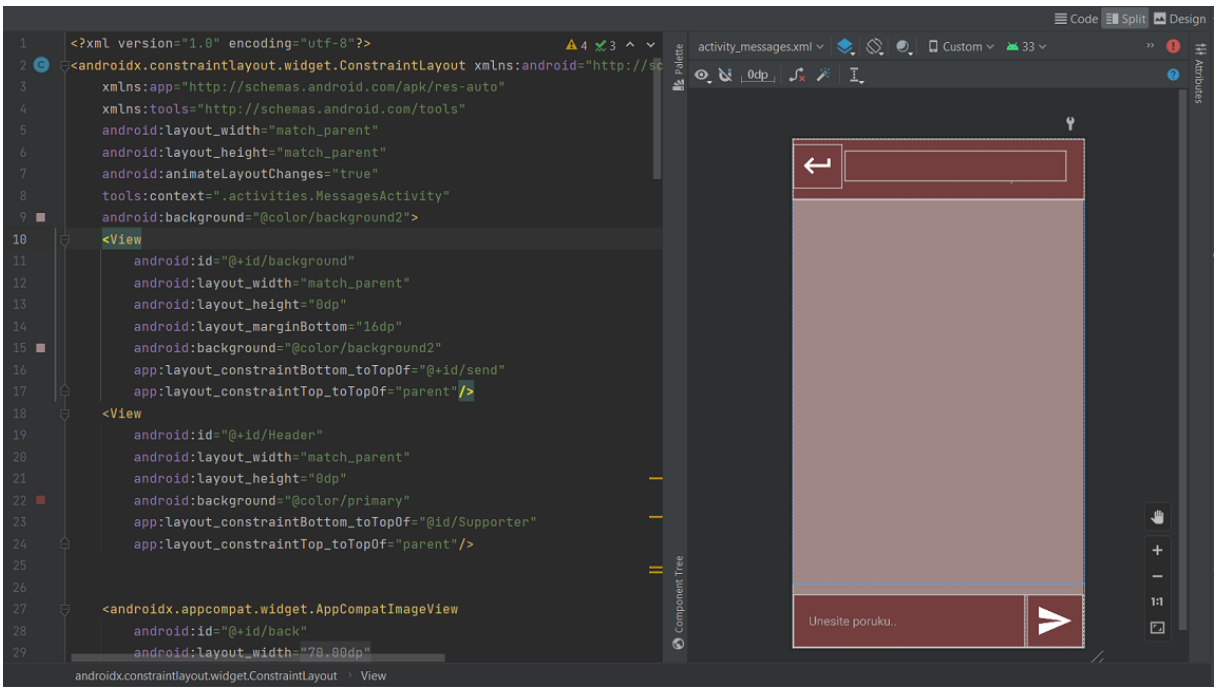
```
38  private void getUsers(){
39      FirebaseFirestore database = FirebaseFirestore.getInstance();
40      database.collection(Constants.USERS)
41              .get()
42              .addOnCompleteListener(task -> {
43
44                  String currentId = preferenceManager.getString(Constants.USER_ID);
45                  if (task.isSuccessful() && task.getResult() != null) {
46                      List<User> users = new ArrayList<>();
47                      for (QueryDocumentSnapshot queryDocumentSnapshot : task.getResult()){
48                          if (currentId.equals(queryDocumentSnapshot.getId())){
49                              continue;
50                          }
51                          User user = new User();
52                          user.name = queryDocumentSnapshot.getString(Constants.NAME);
53                          user.picture = queryDocumentSnapshot.getString(Constants.IMAGE);
54                          user.token = queryDocumentSnapshot.getString(Constants.FCM_TOKEN);
55                          user.id = queryDocumentSnapshot.getId();
56                          users.add(user);
57                      }
58                      if (users.size()>0) {
59                          UsersView usersView = new UsersView(users, listener: this);
60                          binding.userView.setAdapter(usersView);
61                          binding.userView.setVisibility(View.VISIBLE);
62                      }
63                  }
64              });
65
66  }
```

Izvor: autor

5.5 SUČELJE ZA RAZMJENJIVANJE PORUKA

Na red je došlo implementiranje sučelja u kojem će korisnik aplikacije razmjenjivati poruke sa svojim sugovornicima, odnosno s ostalim korisnicima aplikacije. Na početku je prvo potrebno implementirati xml datoteke koje određuju izgled navedenog sučelja. Kao što možemo vidjeti na slici 29. sučelje je vrlo jednostavno i slično sučeljima ostalih aplikacija za razmjenjivanje poruka, dakle gore nam je prikazano ime korisnika s kojim komuniciramo dok nam se na dnu ekrana prikazuje tekstualni okvir u koji pišemo poruku koju želimo poslati. Između ove dvije navedene komponente će se nalaziti poruke koje korisnici razmjenjuju. Na posljetku još imamo i dva gumba, jedan gumb pomoću kojeg ćemo poslati napisanu poruku, nalazi se pokraj tekstualnog okvira u donjem desnom kutu te gumb pomoću kojeg se vraćamo na početnu stranicu aplikacije, nalazi se u gornjem lijevom kutu.

Slika 29. Izgled xml datoteke sučelja za razmjenu poruka



Izvor: autor

Također još imamo i dvije xml datoteke od kojih jedna predstavlja poruke korisnika dok druga xml datoteka predstavlja poruke sugovornika. Ove dvije xml datoteke su vrlo jednostavne te zato nisu zasebno prikazane, razlikuju se po boji pozadine poruke te po tome što je uz poruku sugovornika prikazana i njegova slika.

Kako bi sučelje ispravno radilo za početak je potrebno implementirati funkciju koja dohvaća naziv korisnika s kojim komuniciramo. Kao što možemo vidjeti na slici 30. spomenuta funkcija je vrlo jednostavna i ne zahtijeva dodatno objašnjavanje.

Slika 30. Funkcija za dohvaćanje naziva sugovornika

```
private void loadDetails(){
    user1 = (User) getIntent().getSerializableExtra(Constants.USER);
    binding.name.setText(user1.name);
}
```

Izvor: autor

Dalje je implementirana funkcija za spremanje poslanih poruka u bazu podataka, na slici 31. možemo vidjeti kako zapravo izgleda spomenuta funkcija.

Slika 31. Funkcija za slanje tekstualnih poruka u bazu podataka

```
private void message(){
    HashMap<String, Object> message = new HashMap<>();
    message.put(Constants.SENDER, preferenceManager.getString(Constants.USER_ID));
    message.put(Constants.RECEIVER, user1.id);
    message.put(Constants.MESSAGE, binding.writeMessage.getText().toString());
    database.collection(Constants.COLLECTION).add(message);
    if (chatId != null) {
        updateChat(binding.writeMessage.getText().toString());
    } else {
        HashMap<String, Object> chat = new HashMap<>();
        chat.put(Constants.SENDER, preferenceManager.getString(Constants.USER_ID));
        chat.put(Constants.NAME_SENDER, preferenceManager.getString(Constants.NAME));
        chat.put(Constants.PICTURE_SENDER, preferenceManager.getString(Constants.IMAGE));
        chat.put(Constants.RECEIVER, user1.id);
        chat.put(Constants.NAME_RECEIVER, user1.name);
        chat.put(Constants.PICTURE_RECEIVER, user1.picture);
        addChat(chat);
    }
    binding.writeMessage.setText(null);
}
```

Izvor: autor

Prethodno spomenuta funkcija sprema sve poruke u Firebase bazu podataka, zatim je potrebna funkcija koja će dohvaćati spremljene poruke te ih prikazivati korisniku u aplikaciji. Za to je zadužena funkcija „Messages“ koju možemo vidjeti na slici 32.

Slika 32. Funkcija za dohvaćanje poruka iz baze podataka

```
private void Messages(){
    database.collection(Constants.COLLECTION) CollectionReference
            .whereEqualTo(Constants.SENDER, preferenceManager.getString(Constants.USER_ID)) Query
            .whereEqualTo(Constants.RECEIVER, user1.id)
            .addSnapshotListener(eventListener);
    database.collection(Constants.COLLECTION) CollectionReference
            .whereEqualTo(Constants.SENDER, user1.id) Query
            .whereEqualTo(Constants.RECEIVER, preferenceManager.getString(Constants.USER_ID))
            .addSnapshotListener(eventListener);
}
```

Izvor: autor

5.6 STATUS AKTIVNOSTI KORISNIKA

U aplikaciju je implementiran i status aktivnosti korisnika s kojim razgovaramo, dakle ako je sugovornik trenutno aktivan korisniku će se prikazati, kada uđe u razgovor s njim, tekst ispod njegovog naziva, odnosno pisati će da je na mreži. Ukoliko sugovornik trenutno nije aktivan ispod njegovog naziva neće pisati ništa. Potrebno je u xml datoteku „activity_messages“ dodati tekstualni okvir koji će nam služiti za prikazivanje aktivnosti korisnika u sučelju.

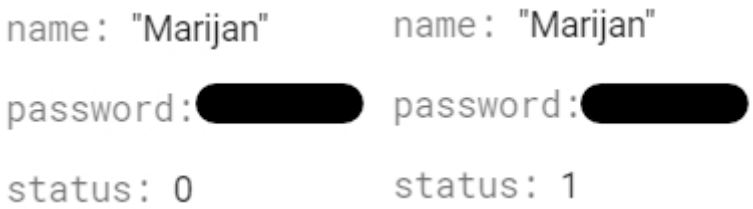
Na samom početku su implementirane dvije funkcije koje će u bazu podataka zapisivati broj 1 ukoliko je korisnik aktivan ili broj 0 ukoliko korisnik nije aktivan. Opisane funkcije su prikazane na slici 33.

Slika 33. Prikaz funkcija koje bilježe aktivnost korisnika

```
@Override
protected void onPause() {
    super.onPause();
    documentReference.update(Constants.USER_STATUS,  value: 0);
}

@Override
protected void onResume() {
    super.onResume();
    documentReference.update(Constants.USER_STATUS,  value: 1);
}
```

Izvor: autor

Slika 34. Aktivnost korisnika u bazi podataka



Izvor: autor

U narednom koraku je u projekt implementirana biblioteka „Retrofit“, Retrofit je jednostavna i brza biblioteka za dohvaćanje i učitavanje podataka putem web usluge temeljene na REST-u. Retrofit upravlja procesom primanja, slanja i stvaranja HTTP zahtjeva i odgovora. Rješava probleme prije slanja pogreške i rušenja aplikacije. Udružuje veze kako bi smanjio kašnjenje. Koristi se za spremanje odgovora u pred memoriju kako bi se izbjeglo slanje dvostrukih zahtjeva.

Dalje imamo funkciju koja provjera u bazi podataka je li korisnik aktivan ili ne te na osnovu toga ispisuje ili ne ispisuje tekst „Na mreži“ ispod naziva sugovornika. Spomenuta funkcija je prikazana na slici 35.

Slika 35. Funkcija za prikaz statusa aktivnosti korisnika

```
private void UserStatus (){
    database.collection(Constants.USERS).document(
            user1.id
    ).addSnapshotListener( activity: MessagesActivity.this, (value, error) -> {
        if (error != null){
            return;
        }
        if (value != null){
            if (value.getLong(Constants.USER_STATUS) != null) {
                int status = Objects.requireNonNull(
                        value.getLong(Constants.USER_STATUS)
                ).intValue();
                isOnline = status == 1;
            }
        }
        if (isOnline){
            binding.textStatus.setVisibility(View.VISIBLE);
        }else {
            binding.textStatus.setVisibility(View.GONE);
        }
    });
```

Izvor: autor

5.7 PREVODITELJ TEKSTUALNIH PORUKA

Za prijevod tekstualnih poruka korišten je Firebase prevoditelj. Na samom početku implementiranja prevoditelja tekstualnih poruka potrebno je uključiti Firebase prevoditelj u projekt, s Firebase-ove dokumentacije su jednostavno kopirane linije koda koje možete vidjeti na slici 36 te su dodane u datoteku „build.gradle“ pod „dependencies“.

Slika 36. Implementiranje Firebase prevoditelja

```
implementation 'com.google.firebase:firebase-ml-natural-language:22.0.1'
implementation 'com.google.firebase:firebase-ml-natural-language-language-id-model:20.0.8'
implementation 'com.google.firebase:firebase-ml-natural-language-translate-model:20.0.9'
implementation "com.google.firebase:firebase-iid:21.1.0"
```

Izvor: autor

Zatim u datoteci „MainActivity“ imamo funkciju koja nam služi za odabir jezika na koji želimo da se naše tekstualne poruke prevode. Kao što možemo vidjeti na slici 37. funkcija dohvaća jezik koji je odabran u padajućem izborniku na početnom sučelju aplikacije te isti šalje u „MessagesActivity“ gdje se dalje poruke prevode na odabrani jezik. Ukoliko korisnik nije odabrao niti jedan jezik poruke će ostati u originalnom formatu.

Slika 37. Odabir jezika

```
public void onClicked(User user) {
    Intent intent = new Intent(getApplicationContext(), MessagesActivity.class);
    intent.putExtra(Constants.USER, user);
    String jezik_odabran = String.valueOf(jezik_dropdown.getText());
    if(jezik_odabran.equals(null) | jezik_odabran.equals(""))
        intent.putExtra( name: "jezik",  value: "org");
    else
        intent.putExtra( name: "jezik", jezik_odabran);
    startActivity(intent);
}
```

Izvor: autor

Zatim prelazimo u „MessagesActivity“ gdje se zapravo odvija cijeli proces oko prevođenja poruka. Na samom početku imamo funkciju koja od prethodno objašnjene funkcije prima vrijednost o odabranom jeziku. Zatim prolazi kroz sve ponuđene jezike te postavlja prevoditelj na onaj jezik koji je korisnik odabrao. Na slici 38. možemo vidjeti kako zapravo izgleda ta funkcija, također možemo vidjeti da ukoliko ne odaberemo niti jedan jezik poruke će se prikazivati u originalnom formatu.

Slika 38. Izgled funkcije za postavljanje jezika

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    binding = ActivityMessagesBinding.inflate(getLayoutInflater());
    setContentView(binding.getRoot());
    jezik_odabran = getIntent().getExtras().getString( key: "jezik");
    if(jezik_odabran == null)
        a = 1;
    if(jezik_odabran == null)
        jezik_odabran = "org";
    if (jezik_odabran.equals("Francuski") )
        jezik_odabran = "FR";
    else if (jezik_odabran.equals("Engleski"))
        jezik_odabran = "EN";
    else if (jezik_odabran.equals("Njemački"))
        jezik_odabran = "DE";
    else if (jezik_odabran.equals("Španjolski"))
        jezik_odabran = "ES";
    else if (jezik_odabran.equals("Ruski"))
        jezik_odabran = "RU";
    else if (jezik_odabran.equals("Talijanski"))
        jezik_odabran = "IT";
    else
        jezik_odabran = "org";
```

Izvor: autor

Zatim imamo funkciju pomoću koje program prepoznaje originalni jezik s kojim je napisana poruka.

Ukoliko korisnik nije odabrao niti jedan jezik poruka će se jednostavno ispisati te aplikacija neće dalje ulaziti u petlju za prijevod poruke. Može se desiti i da program ne može prepoznati jezik, u tom slučaju će se također ispisati originalna poruka.

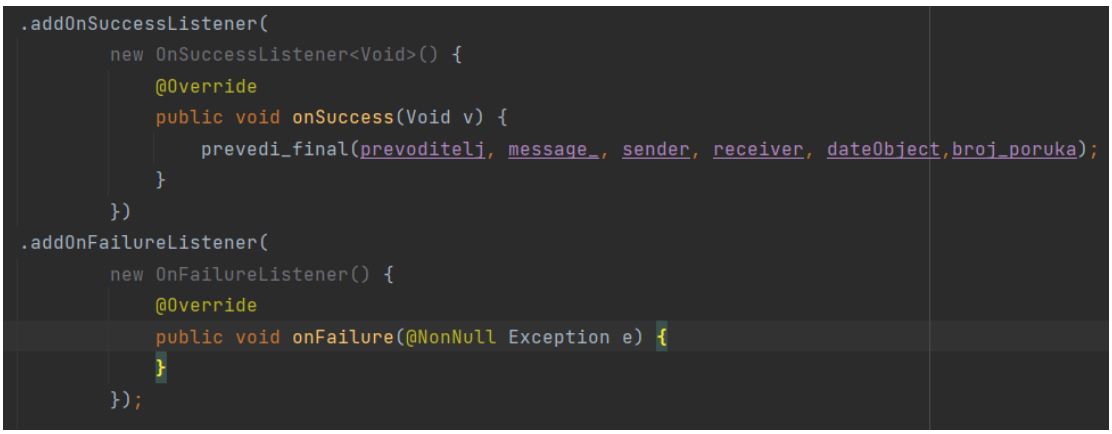
Slika 39. Provjera uvjeta za prijevod teksta

Izvor: autor

Na samom vrhu slike 39 možemo vidjeti naredbu koja provjerava je li odabran neki jezik koji je nedostupan za prevođenje, ukoliko je nedostupan postavit će se hrvatski kao zadani jezik. U nastavku koda se provjerava ima li Firebase alate za prijevod odabranog teksta, ukoliko ih nema pokušati će ih preuzeti.

Ukoliko su zadovoljeni svi uvjeti, odnosno ukoliko preuzimanje alata za prijevod teksta nije bilo potrebno ili je bilo potrebno te je uspješno izvršeno kod prelazi u „onSuccess“ u kojoj se dalje poziva funkcija za prijevod poruke. U slučaju da nije moguće izvršiti prijevod poruke te da nije moguće preuzeti alate za prijevod kod odlazi u funkciju „onFailure“. Opisane funkcije možemo vidjeti na slici 40.

Slika 40. Funkcije za slučaj uspješnog i neuspješnog pronalaženja jezika za prijevod



Izvor: autor

Zatim imamo glavnu funkciju za prijevod poruka, funkcija je prikazana na slici 41. Također imamo slučaj gdje ako je prijevod uspješan kod odlazi u „onSucess“ funkciju. Navedena funkcija kao parametar ima tekst koji se ukoliko je prijevod uspješan odmah prevodi i ažurira. Dalje imamo naredbu koja nam sortira tekstualne poruke po vremenu kada su poslane, ista se ponavlja više puta kroz kod.

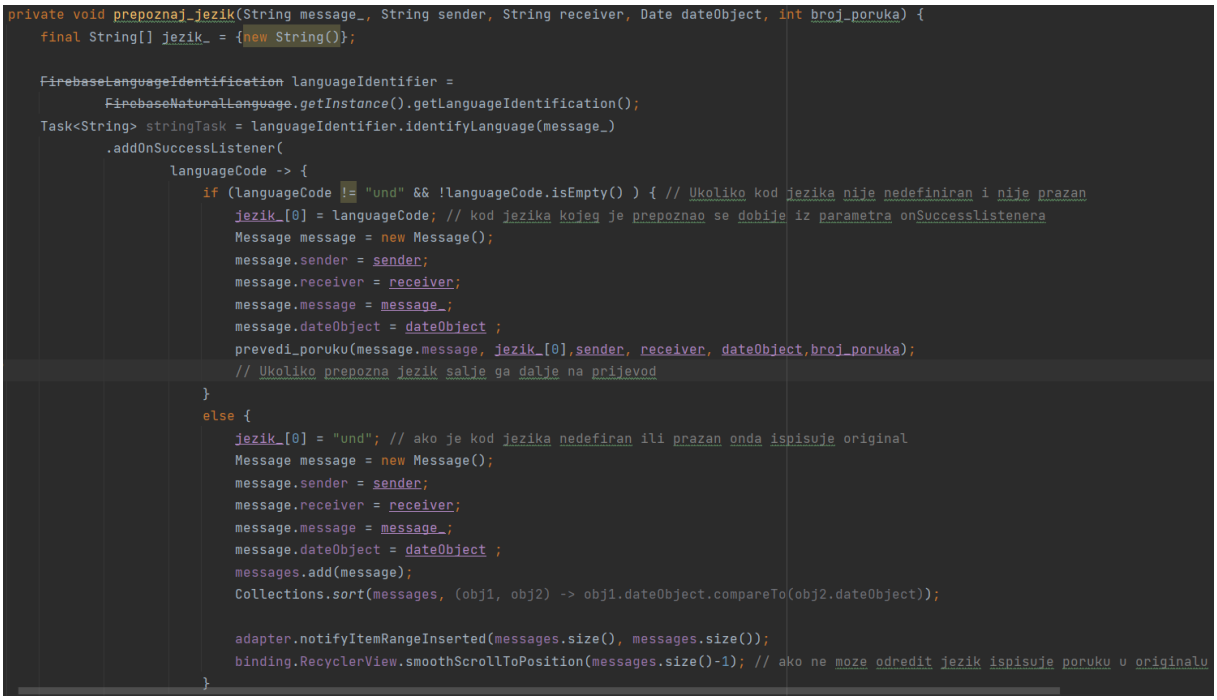
Slika 41. Funkcija za prijevod poruka

```
private void prevedi_final(FirebaseTranslator prevoditelj, String message_, String sender, String receiver, Date dateObject, int broj_poruka) {
    ++broj_poruka_temp;
    kraj_chata = false;
    prevoditelj.translate(message_)
            .addOnSuccessListener(
                    new OnSuccessListener<String>() {
                        @Override
                        public void onSuccess(@NonNull String translatedText) {

                            // Translation successful.
                            Message message = new Message();
                            message.sender = sender;
                            message.receiver = receiver;
                            message.message = translatedText;
                            message.dateObject = dateObject;
                            messages.add(message);
                            Collections.sort(messages, (obj1, obj2) -> obj1.dateObject.compareTo(obj2.dateObject)); //Sortiranje poruka

                            adapter.notifyItemRangeInserted(messages.size(), messages.size());
                            binding.RecyclerView.smoothScrollToPosition(messages.size()-1);

                        }
                    })
            .addOnFailureListener(
                    new OnFailureListener() {
                        @Override
                        public void onFailure(@NonNull Exception e) {
                        }
                    });}
```

Izvor: autor

Za prikazati je ostala još funkcija „prepoznaj_jezik“, ista prepoznaje jezik kojim je napisana originalna poruka kako bi bilo moguće izvršiti prijevod, spomenutu funkciju možemo vidjeti na slici 42. Ukoliko kod jezika nije nedefiniran i nije prazan pozvati će se funkcija za prijevod poruke, ukoliko je kod jezika nedefiniran ili prazan ispisati će se originalna poruka. Ako ne može odrediti jezik također će se ispisati poruka u originalnoj verziji.

Slika 42. Funkcija za prepoznavanje originalno napisanog jezika



```
private void prepoznaj_jezik(String message_, String sender, String receiver, Date dateObject, int broj_poruka) {
    final String[] jezik_ = {new String()};

    FirebaseLanguageIdentification languageIdentifier =
            FirebaseNaturalLanguage.getInstance().getLanguageIdentification();
    Task<String> stringTask = languageIdentifier.identifyLanguage(message_)
            .addOnSuccessListener(
                    languageCode -> {
                        if (languageCode != "und" && !languageCode.isEmpty() ) { // Ukoliko kod jezika nije nedefiniran i nije prazan
                            jezik_[0] = languageCode; // kod jezika kojeg je prepoznao se dobije iz parametra onSuccesslistenera
                            Message message = new Message();
                            message.sender = sender;
                            message.receiver = receiver;
                            message.message = message_;
                            message.dateObject = dateObject ;
                            prevedi_poruku(message.message, jezik_[0],sender, receiver, dateObject,broj_poruka);
                            // Ukoliko prepozna jezik salje ga dalje na prijevod
                        }
                        else {
                            jezik_[0] = "und"; // ako je kod jezika nedefiran ili prazan onda ispisuje original
                            Message message = new Message();
                            message.sender = sender;
                            message.receiver = receiver;
                            message.message = message_;
                            message.dateObject = dateObject ;
                            messages.add(message);
                            Collections.sort(messages, (obj1, obj2) -> obj1.dateObject.compareTo(obj2.dateObject));

                            adapter.notifyItemRangeInserted(messages.size(), messages.size());
                            binding.RecyclerView.smoothScrollToPosition(messages.size()-1); // ako ne moze odredit jezik ispisuje poruku u originalu
                        }
```

Izvor: autor

6. KORISNIČKE UPUTE

6.1 PRIJAVA I REGISTRACIJA U APLIKACIJU

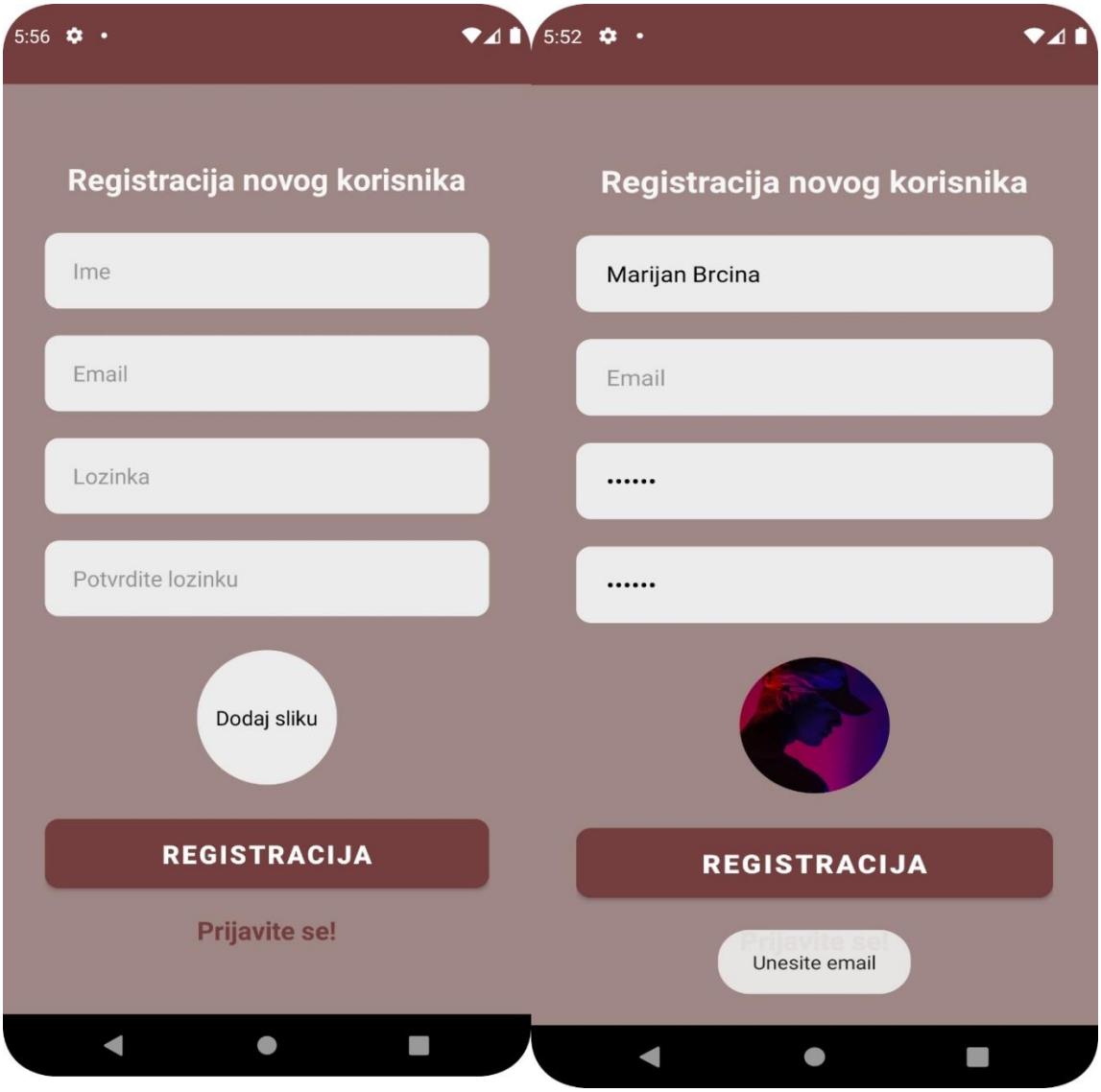
Pri svakom pokretanju aplikacije, ukoliko smo se prethodno odjavili, otvorit će nam se sučelje za prijavu u aplikaciju. Kao što možemo vidjeti na slici 43. prijava u aplikaciju je vrlo jednostavna odnosno dosta slična kao i prijava u svaku drugu aplikaciju. Za prijavu u aplikaciju nam je potrebna email adresa te lozinka koju smo postavili pri registraciji.

Slika 43. Prijava u aplikaciju

Izvor: autor

Ukoliko ste prvi put pokrenuli aplikaciju ili se želite registrirati s nekim drugim profilom potrebno je jednostavno kliknuti na tekst „Registriraj se“ koji se nalazi ispod gumba za prijavu te će vas aplikacija prebaciti na sučelje koje služi za registraciju u aplikaciju. Izgled spomenutog sučelja možemo vidjeti na slici 44. Dakle, za registraciju u aplikaciju potrebno je unijeti ime, email, lozinku zatim ponoviti unesenu lozinku te je na posljetku potrebno iz galerije učitati sliku koja će se koristiti kao profilna slika.

Slika 44. Registracija u aplikaciju



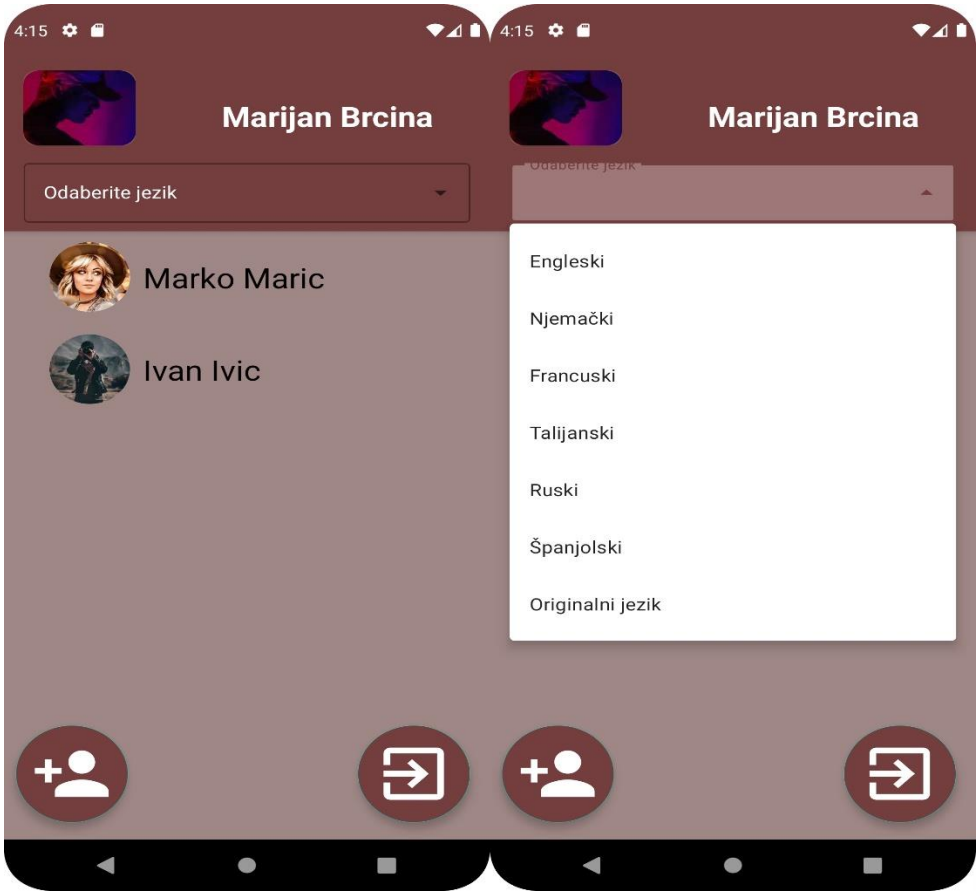
Izvor: autor

Na slici 44. također možemo vidjeti primjer poruke koja će nam se ispisati ukoliko izostavimo neki od podataka koji je potreban za registraciju u aplikaciju.

6.2 POČETNO SUČELJE APLIKACIJE

Nakon uspješne prijave ili registracije dolazimo na početno sučelje aplikacije. Kao što možemo vidjeti na slici 45. početno sučelje aplikacije nam nudi nekoliko mogućnosti. Na samom vrhu sučelja je prikaza slika profila i ime korisnika koji je trenutno prijavljen, dalje imamo padajući izbornik u kojem je potrebno izabrati jezik na kojem želimo da nam stižu tekstualne poruke koje nam šalju drugi korisnici. Nakon toga imamo dva korisnika s kojima smo već komunicirali kako bi mogli brže pristupiti osobama s kojima često komuniciramo. Na samom dnu početnog sučelja možemo vidjeti dva gumba, gumb koji se nalazi s lijeve strane služi nam kako bi se prebacili u sučelje koje nam prikazuje sve registrirane korisnike, odnosno korisnike s kojima imamo mogućnost komunicirati. U donjem desnom kutu možemo vidjeti gumb koji nam služi za odjavu iz aplikacije, klikom na taj gumb vraćamo se na sučelje za prijavu u aplikaciju. Na slici 45. prikazan je i izgled padajućeg izbornika, odnosno na spomenutoj slici možemo vidjeti koju su sve jezici za prijevod poruka trenutno dostupni u aplikaciji.

Slika 45. Početno sučelje aplikacije

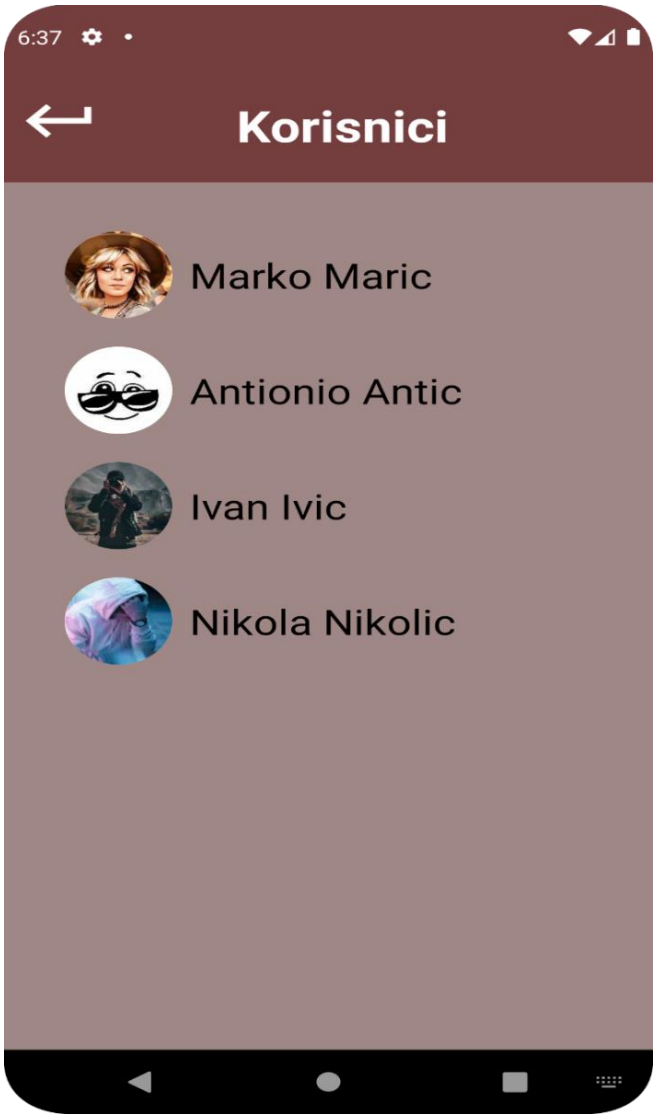


Izvor: autor

6.3 PRIKAZ REGISTRIRANIH KORISNIKA

Klikom na gumb u donjem lijevom kutu početnog sučelja aplikacija dolazimo na sučelje koje je prikazano na slici 46. odnosno sučelje koje nam prikazuje sve registrirane korisnike ove aplikacije, dakle prikazuje nam one korisnike s kojima imamo mogućnost komuniciranja putem aplikacije. Iz opisanog sučelja se klikom na strelicu koja se nalazi u gornjem lijevom kutu vraćamo na početno sučelje aplikacije dok klikom na pojedinog korisnika otvaramo sučelje za razgovor s odabranim korisnikom.

Slika 46. Registrirani korisnici aplikacije

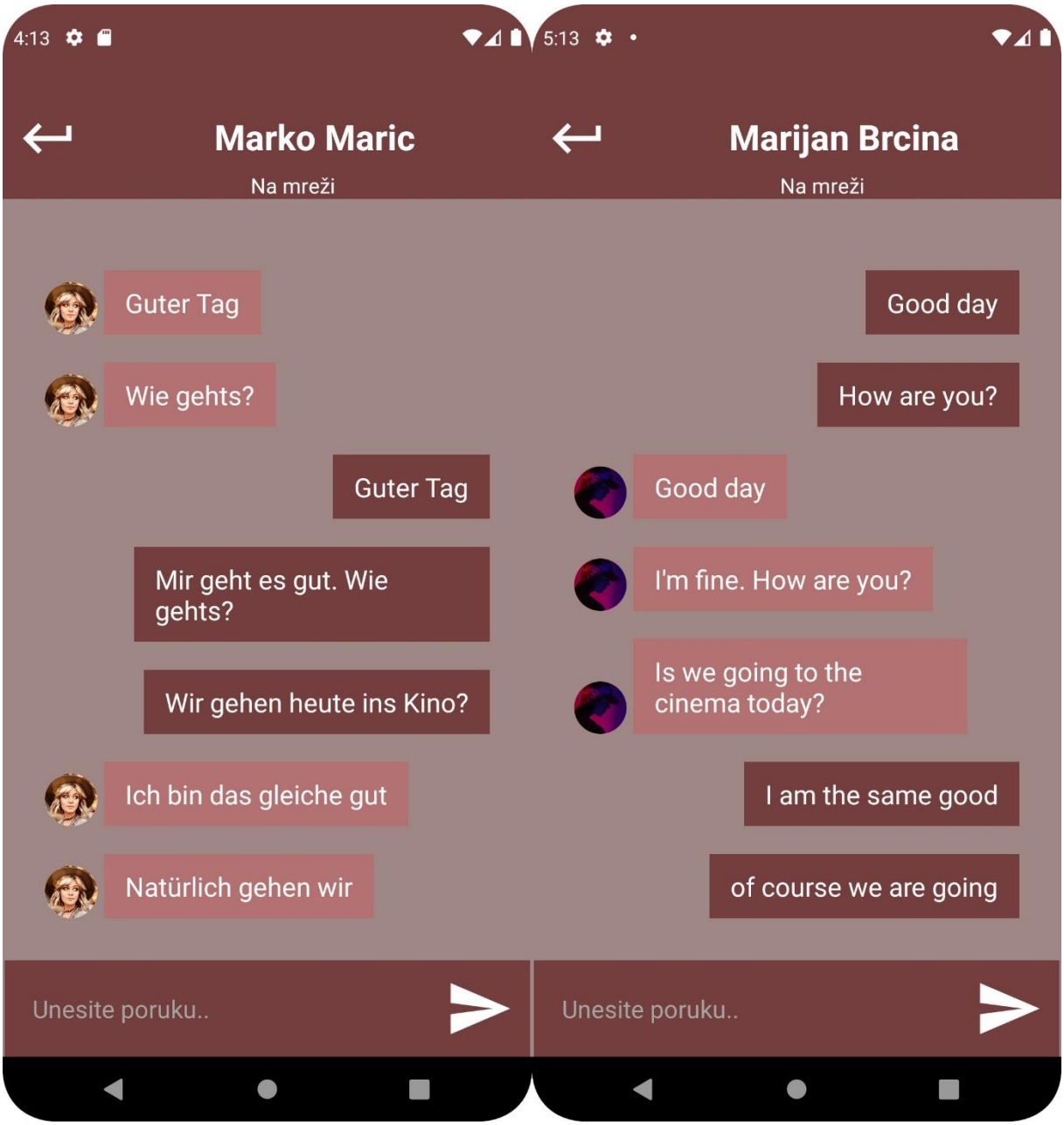


Izvor: autor

6.4 SUČELJE ZA RAZGOVOR

Klikom na pojedinog korisnika, bilo to u početnom sučelju ili sučelju koje nam prikazuje registrirane korisnike, dolazimo u sučelje za razgovor, odnosno sučelje gdje razmjenjujemo tekstualne poruke sa odabranim korisnikom. Pogledom na sliku 47. možemo vidjeti da se na samom vrhu sučelja prikazuje ime korisnika s kojim komuniciramo, u gornjem lijevom kutu možemo vidjeti strelicu pomoću koje se možemo vratiti na početno sučelje aplikacije. Na samom dnu sučelja za razgovor imamo tekstualni okvir u koji zapravo pišemo poruku, klikom na tekstualni okvir otvara nam se tipkovnica pomoću koje pišemo poruku. Kao i kod svake druge aplikacije, u središnjem dijelu sučelja prikazuju nam se naše poruke i poruke sugovornika, razlikovati ćemo ih po tome što se naše poruke prikazuju s desne strane dok se poruke sugovornika prikazuju s lijeve strane, možemo ih također razlikovati i po boji pozadine tekstualnog okvira, što možemo primijetiti i ako pogledamo sliku 47. Na slici 47. je također prikazana kratka konverzacija između dva korisnika, iz ovog primjera možemo vidjeti kako zapravo radi prevoditelj tekstualnih poruka, korisnik Marko je odabrao da se njegove poruke prikazuju na njemačkom jeziku dok je Marijan odabrao engleski jezik. Također možemo vidjeti da ispod naziva oba korisnika piše da su oni na mreži, odnosno da su trenutno aktivni što nam potvrđuje da i ta funkcionalnost radi savršeno.

Slika 47. Primjer razgovora između dva korisnika



Izvor: autor

7. ZAKLJUČAK

Svi smo dobro upoznati s tim da internet postaje sve potrebniji za današnje društvo, mnoge brojke nam govore o tome kako je sve teže opstati na tržištu ukoliko ne poslujete preko interneta. Sa sigurnošću mogu reći da online komunikacija ima veliki utjecaj u povezivanju ljudi, poduzeća odnosno općenito cijeloga svijeta. Udaljenost i vrijeme su postali skoro pa nebitan faktor u ljudskoj komunikaciji. Korona pandemija koja je vladala do nedavno je dobar pokazatelj koliko je bitna internet komunikacija a uz to i internet poslovanje.

Velika prepreka u današnjem svijetu je nerazumijevanje govornog jezika sugovornika, odnosno kada se na internetu susretnu dva sugovornika koji ne poznaju isti govorni jezik. U navedenom slučaju je komunikacija vrlo otežana te mnogo teže dolazi do sklapanja novih prijateljstava, poslova itd. Dobro smo upoznati s tim da u današnjem svijetu postoji jako puno kvalitetnih aplikacija za internet komunikaciju, međutim niti jedna od popularnih aplikacija ne nudi mogućnost prijevoda teksta koji nam naš sugovornik šalje.

Tijekom izrade ovog diplomskog rada fokusirao sam se upravo na prijevod tekstualnih poruka unutar internetskog razgovora. Smatram da je u današnje vrijeme vrlo bitno da sugovornici ostvare brzu i jasnu komunikaciju neovisno o tome odakle dolaze i koji jezik pričaju.

8. LITERATURA

Članci:

1) Neil Smyth, (2015.), Android Studio Development Essentials – Second Edition, Dostupno na: https://www.ebookfrenzy.com/pdf_previews/AndroidStudioEssentialsPreview.pdf [Pristupljeno: 11.01.2023.]

2) J. Paul Cardle, (2017.), Android App Development in Android Studio - Java + Android Edition for Beginners, Dostupno na: https://www.pdfdrive.com/android-app-development-in-android-studio-javaandroid-edition-for-beginners-e60596566.html [Pristupljeno: 14.01.2023.]

Internet izvori:

1) https://element.hr/wp-content/uploads/2020/06/unutra-13579.pdf [Pristupljeno: 11.01.2023.]
2) https://www.edureka.co/blog/what-is-java/ [Pristupljeno: 11.01.2023.]
3) https://repozitorij.unipu.hr/islandora/object/unipu%3A3649/datastream/PDF/view [Pristupljeno: 12.01.2023.]
4) https://repozitorij.oss.unist.hr/islandora/object/ossst%3A516/datastream/PDF/view [Pristupljeno: 12.01.2023.]
5) https://medium.com/firebase-developers/what-is-firebase-the-complete-story-abridged-bcc730c5f2c0 [Pristupljeno: 15.01.2023.]
6) https://www.uplabs.com/posts/sdp-a-scalable-size-unit [Pristupljeno: 17.01.2023.]
7) https://www.geeksforgeeks.org/how-to-create-language-translator-in-android-using-firebase-ml-kit/ [Pristupljeno: 25.01.2023.]

9. POPIS SLIKA

Slika 1. Prikaz široke primjene programskog jezika Java 4
Slika 2. Sučelje alata Android Studio 6
Slika 3. Prikaz spremljenih podataka u Google Firebase bazi podataka 9
Slika 4. Podaci o upitima i zapisima prema Firebase bazi podataka 10
Slika 5. Use Case dijagram 13
Slika 6. Sequence diagram (Dijagram toka) 15
Slika 7. Registrirani korisnici 16
Slika 8. Klasni dijagram 18
Slika 9. Primjer upotrebe SDP i SSP alata 23
Slika 10. Implementiranje alata SDP i SSP u projekt 24
Slika 11. Implementiranje alata Rounded ImageView 24
Slika 12. Uključivanje značajke „View Binding“ 24
Slika 13. google-services.json datoteka 25
Slika 14. Uspješno povezivanje aplikacije s alatom Google Firebase 26
Slika 15. Izgled xml datoteke za registraciju u aplikaciju 27
Slika 16. Izgled xml datoteke za prijavu u aplikaciju 27
Slika 17. Provjera ispravnosti podataka pri registraciji u aplikaciju 28
Slika 18. Provjera unesenih podataka pri prijavi u aplikaciju 29
Slika 19. Implementacija registracije u aplikaciju 29
Slika 20. Implementacija prijave u aplikaciju 30
Slika 21. Implementacija Base64 metode kodiranja 31
Slika 22. Spremanje fotografije u Google Firebase bazu podataka 31
Slika 23. Izgled xml datoteke početnog sučelja aplikacije 32
Slika 24. Funkcija koja prikazuje ime korisnika i sliku profila 33
Slika 25. Dohvaćanje podataka o sugovornicima 33
Slika 26. Izgled xml datoteke koja predstavlja pojedinog korisnika 34
Slika 27. Izgled xml datoteke pomoću koje se prikazuju registrirani korisnici 35
Slika 28. Funkcija za dohvaćanje podataka o registriranim korisnicima 36
Slika 29. Izgled xml datoteke sučelja za razmjenu poruka 37
Slika 30. Funkcija za dohvaćanje naziva sugovornika 38
Slika 31. Funkcija za slanje tekstualnih poruka u bazu podataka 38
Slika 32. Funkcija za dohvaćanje poruka iz baze podataka 39
Slika 33. Prikaz funkcija koje bilježe aktivnost korisnika 40
Slika 34. Aktivnost korisnika u bazi podataka 40
Slika 35. Funkcija za prikaz statusa aktivnosti korisnika 41
Slika 36. Implementiranje Firebase prevoditelja 42
Slika 37. Odabir jezika 42
Slika 38. Izgled funkcije za postavljanje jezika 43
Slika 39. Provjera uvjeta za prijevod teksta 44
Slika 40. Funkcije za slučaj uspješnog i neuspješnog pronalaženja jezika za prijevod 44
Slika 41. Funkcija za prijevod poruka 45
Slika 42. Funkcija za prepoznavanje originalno napisanog jezika 46
Slika 43. Prijava u aplikaciju 47
Slika 44. Registracija u aplikaciju 48
Slika 45. Početno sučelje aplikacije 49

Slika 46. Registrirani korisnici aplikacije 50
Slika 47. Primjer razgovora između dva korisnika..... 52

10. SAŽETAK

Struktura ovo diplomskog rada dijeli se na teorijski i praktični dio razvoja mobilne aplikacije za dopisivanje s prijevodom. U prvom dijelu diplomskog rada opisane su tehnologije i alati koji su korišteni pri izradi same aplikacije. Zatim imamo nekoliko dijagrama koji objašnjavaju način rada aplikacije. Teorijski dio ovog rada zaključen je swot analizom koja nam je omogućila da ispravno sagledamo unutrašnje snage i slabosti aplikacije te vanjske prilike i prijetnje s kojima se aplikacija susreće. U nastavku ovog rada prikazana je implementacija mobilne aplikacije, ona ujedno predstavlja i praktični dio diplomskog rada. Na posljetku dolazimo do korisničkih uputa koje su napisane u svrhu lakšeg snalaženja korisnika pri korištenju same aplikacije.

Ključne riječi: Diplomski rad, Mobilna aplikacija, razmjenjivanje tekstualnih poruka, prijevod poruka, Android Studio, Firebase, Java

SUMMARY

The structure of this thesis is divided into the theoretical and practical part of the development of mobile applications for correspondence with translation. The first part of the thesis describes the technologies and tools that were used in the creation of the application itself. Then we have a few diagrams that explain how the app works. The theoretical part of this work was concluded with a swot analysis that allowed us to correctly see the internal strengths and weaknesses of the application and the external opportunities and threats that the application faces. In the continuation of this paper, the implementation of the mobile application is shown, it also represents the practical part of the thesis. Finally, we come to the user instructions, which are written for the purpose of making it easier for users to find their way around using the application itself.

Key words: Graduation thesis, Mobile application, exchanging text messages, messages translation, Android Studio, Firebase, Java