

SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
V A R A Ž D I N

Goran Drmenčić

Danijel Filipović

Matija Jurman

Danijel Sokač

ChatUP aplikacija

TEHNIČKA DOKUMENTACIJA

Github repozitij:

- <https://github.com/DanijelFilipovic/Chat-aplikacija>
- <https://github.com/DanijelFilipovic/ChatUp-Dodatno> (Web servisi)

Varaždin, 2015.

SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
V A R A Ž D I N

Oznaka tima: T18

Članovi tima:

- Goran Drmenčić, 0246018410
- Danijel Filipović, 0016090066
- Matija Jurman, 0016084901
- Danijel Sokač, 0016090066

ChatUP aplikacija
TEHNIČKA DOKUMENTACIJA

Github repozitorij:

- <https://github.com/DanijelFilipovic/Chat-aplikacija>
- <https://github.com/DanijelFilipovic/ChatUp-Dodatno> (Web servisi)

Mentor

:

Ivan Švogor, mag. inf.

Varaždin, studeni 2015.

Sadržaj

1. Uvod	1
2. Use-Case Dijagram.....	2
2.1. Korisnički zahtjevi.....	4
2.1.1. Perspektiva proizvoda i ciljano tržište.....	4
2.1.2. Korisničke priče.....	4
2.2. Prototip aplikacije.....	6
2.2.1. Slika prototipa aplikacije.....	6
3. Objašnjenje svakog pojedinog Mockup-a	1
4. Konfiguracija servera i baze podataka.....	6
5. Arhitektura sustava	10
5.1. Objašnjenje simbola	11
6. Web servisi	13
6.1. server.js.....	13
6.2. register.js	13
6.3. log_in.js	13
6.4. log_out.js	13
6.5. registeredUsers.js.....	13
6.6. get_messages.js	14
6.7. createConversation.js.....	14
6.8. sendMessage.js	14
6.9. getNewMessages.js	14
6.10. save_profile_pic.js	14
6.11. addFriends.js	14
6.12. editProfile.js	15
6.13. getUserDataEditProfile.js	15
6.14. registeredUsers2.js	15
6.15. forgotPassword.js.....	16
7. Dijagram klasa.....	17
7.1. app modul	17
7.2. core modul	19
7.3. webservice modul.....	20

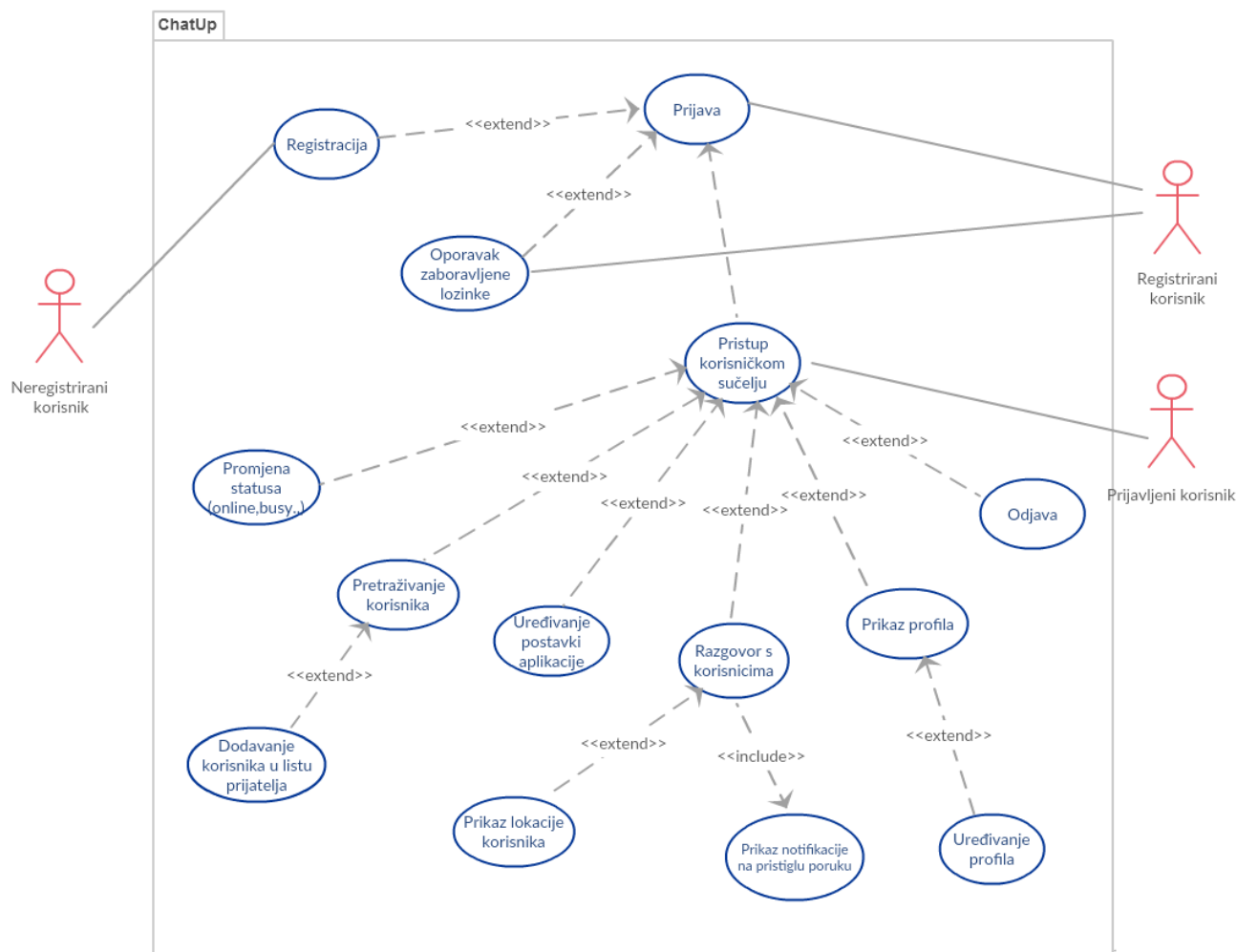
1. Uvod

Ovo je tehnička dokumentacija za izradu android aplikacije ChatUp kao projekt za kolegij Analiza i razvoj programa. U ovom dokumentu se mogu pronaći svi tehnički artefakti koji su bili potrebni za nastanak aplikacije.

Pod tehničke artefakte ubrajamo: izvorni kod aplikacije (ukoliko bude potrebno objasniti neke detalje), dijagrami (koji objašnjavaju aplikaciju) i tehnologija koja se koristila pri izradi aplikacije.

2. Use-Case Dijagram

Slučajevi korištenja aplikacije s tehničke razine biti će prikazani Use-Case dijagramom.



Slika 1: Use-Case dijagram ChatUp aplikacije

Iz tog je dijagrama vidljivo sljedeće. Imamo tri tipa korisnika aplikacije: neregistrirani, registrirani i prijavljeni korisnik. Neregistrirani korisnik može se samo registrirati. Registrirani korisnik može se prijaviti ili zatražiti oporavak izgubljene lozinke. Treći tip korisnika je prijavljeni korisnik. Riječ je o korisniku koji to postaje nako uspješne prijave u aplikaciju. Prijavljeni korisnik automatski pristupa korisničkom sučelju, a opcionalno, može izvoditi sljedeće akcije:

- Odjaviti se iz aplikacije
- Prikazati i urediti svoj profil

- Promjeniti vlastiti status
- Pretraživati druge korisnike aplikacije
- Dodati drugog prijatelja u svoju listu prijatelja, što uključuje dobivanje notifikacije prilikom dobivanja poruke od drugog korisnika
- Urediti vlastite postavke aplikacije
- Razgovarati s drugim korisnicima, što uključuje dobivanje notifikacije o primljenoj poruci kada mu drugi korisnik pošalje poruku
- Prikazati vlastitu lokaciju, ili vidjeti lokaciju od drugog prijatelja koji šalje poruku [opcionalno]

2.1. Korisnički zahtjevi

2.1.1. Perspektiva proizvoda i ciljano tržište

Kako je u današnje vrijeme zabilježen strahovit porast tzv. online ili virtualnih poznanstava, virtualne interakcije i virtualne komunikacije ovo programsko rješenje biti će namijenjeno prvenstveno mlađoj populaciji, kojoj je u današnje vrijeme gotovo nemoguće zamisliti život bez Facebook Messengera, Skype-a i sličnih programskih rješenja za online komunikaciju. No, aplikacija nije namijenjena samo njima, već i svim dobnim skupinama koje se brzo i lako prilagođavaju novim tehnologijama te žele držati korak sa modernom tehnologijom.

2.1.2. Korisničke priče

U nastavku slijede zahtjevi korisnika prikazani u obliku korisničkih priča, zajedno sa skraćenim nazivom svake priče koji će se koristiti kasnije u Sprint i Product Backlogu, te pripadnim prioritetom svakog zahtjeva korisnika odnosno svake korisničke priče. Prioritet će biti korišten tijekom razvoja aplikacije sa svrhom dobivanja uvida u prioritete implementacije određenih funkcionalnosti konačnog programskog rješenja. Ovisno o tim prioritetima definirati će se terminski plan projekta koji će biti prikazan u projektnoj dokumentaciji *Gantogramom*.

Kraći naziv	Korisničke priče	Prioritet
Izrada baze i konfiguracija servera	Ja kao korisnik aplikacije želim da se podaci koje koristi aplikacija ne spremaju lokalno na mobilni uređaj.	Visoki (1)
Registracija	Ja kao neregistrirani korisnik aplikacije želim imati mogućnost pristupa samo početnom ekranu za prijavu ili ekranu za registraciju na kojem se mogu registrirati sa podacima koje unesem u validnom formatu.	Visoki (2)
Prijava	Ja kao registrirani korisnik aplikacije želim imati mogućnost prijave u aplikaciju te pristup svim mogućnostima koje će nuditi aplikacija. Također želim imati mogućnost oporavka zaboravljene lozinke na način da unesem svoj e-mail s kojim sam registriran u aplikaciju i na taj e-mail dobijem lozinku za taj račun.	Visoki (3)
Odjava	Ja kao prijavljeni korisnik želim imati mogućnost odjave iz aplikacije nakon koje se otvara početni ekran za prijavu u aplikaciju.	Visoki (4)
Profilna stranica	Ja kao prijavljeni korisnik želim imati mogućnost pregleda vlastite profilne stranice zajedno s podacima o mojem korisničkom imenu, statusu, prijateljima i njihovim statusima.	Visoki (5)
Pretraživanje i dodavanje drugih korisnika	Ja kao prijavljeni korisnik želim imati mogućnost pretraživanja drugih registriranih korisnika, te mogućnost njihovih dodavanja u moju listu prijatelja na početnu profilnu stranicu.	Visoki (6)
Razgovor s drugim korisnicima	Ja kao prijavljeni korisnik želim imati mogućnost razgovora s mojim prijateljima koji se nalaze u mojoj listi prijatelja na početnoj profilnoj stranici.	Visoki (7)
Izbornik	Ja kao prijavljeni korisnik želim imati mogućnost odabira jedne od sljedećih stavki izbornika: uređivanje profila, skok na početnu profilnu stranicu, pregled podataka o razvojnom timu aplikacije, odjava iz aplikacije i zatvaranje aplikacije.	Srednji (8)
Grupni razgovori	Ja kao prijavljeni korisnik želim imati mogućnost razgovora sa dva ili više sugovornika u jednom razgovoru.	Srednji (9)

Povijest razgovora	Ja kao prijavljeni korisnik želim imati mogućnost pregleda povijesti svih razgovora u kojima sam sudjelovao, te mogućnost otvaranja nekog od tih razgovora jednostavnim odabirom tog razgovora.	Srednji (10)
Notifikacija kod dodavanja prijatelja	Ja kao prijavljeni korisnik želim imati mogućnost primitka notifikacije z trenutku kada me neki korisnik aplikacije doda u svoju listu prijatelja.	Srednji (11)
Notifikacija na primljenu poruku	Ja kao prijavljeni korisnik želim imati mogućnost primitka notifikacije kada mi neki korisnik aplikacije napiše novu poruku.	Srednji (12)
Promjena slike profila	Ja kao prijavljeni korisnik želim imati mogućnost promjene profilne slike. Također želim imati mogućnost rezanja te slike ukoliko ne budem zadovoljan sa dimenzijama slike	Srednji (13)
Uređivanje profila	Ja kao prijavljeni korisnik želim imati mogućnost uređivanja vlastitog profila. To podrazumijeva promjenu korisničkog imena, spola i lozinke	Niski (14)
Promjena statusa	Ja kao prijavljeni korisnik želim imati mogućnost pregleda vlastitog statusa i promjene istog. Također želim vidjeti koji status ima svaki pojedini prijatelj iz vlastite liste prijatelja na početnoj profilnoj stranici.	Niski (15)
Dodatna notifikacija	Ja kao prijavljeni korisnik želim imati mogućnost neke dodatne notifikacije u aplikaciji ovisno o želji razvojnog tima.	Niski (16)

2.2. Prototip aplikacije

Kako bi se objasnio prototip aplikacije najprije je potrebno prikazati sliku kompletnog prototipa, a kasnije detaljno objasniti svaki ekran na prototipu zasebno. Svaki ekran prikazan je uz pomoć Mockup-a i izrađen u alatu Ninjamock.

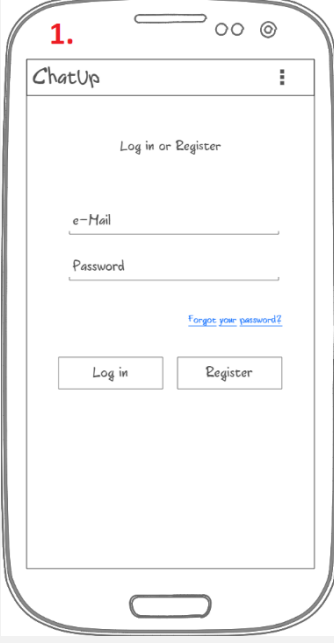

2.2.1. Slika prototipa aplikacije

Prototip aplikacije izgleda kao što je to prikazano na slici 2.1.



Slika 2.1 Prototip aplikacije ChatUp

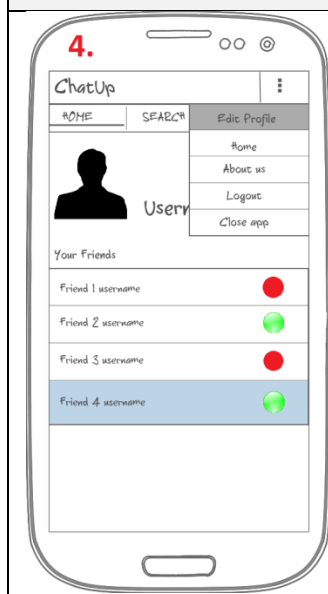
3. Objašnjenje svakog pojedinog Mockup-a

Izgled pojedinog ekrana (mockup)	Objašnjenje	Korisnička priča (kraći naziv)
	<p>Ovako izgleda prvi ekran koji se otvara prilikom otvaranja aplikacije. Registriranom korisniku se nudi mogućnost prijave sa već postojećom e-mail adresom i lozinkom, te također mogućnost oporavka zaboravljene lozinke. Neregistriranom korisniku se nudi mogućnost odabira gumba „Register“ kojim će se registrirati i tako postati potencijalni korisnik aplikacije.</p>	<p>Prijava (prioritet 3)</p>
	<p>Neregistrirani korisnik unosi redom sljedeće podatke u registracijski obrazac: e-mail adresa (u validnom formatu), korisničko ime, lozinku, potvrdu lozinke, spol i datum rođenja. Validnim unosom svih ranije nabrojanih podataka i klikom na gumb „Register“ korisnik se registrira za korištenje aplikacije. Uneseni podaci spremaju se u CouchDB bazu podataka na serveru. Ukoliko uneseni podaci nisu u validnom formatu, korisnik dobiva poruku o krivom unosu.</p>	<p>Registracija (prioritet 2)</p>



Ukoliko korisnik zaboravi lozinku, omogućen mu je oporavak zaboravljene lozinke. Korisnik unosi svoju e-mail adresu, i na nju – ukoliko je registriran s tom e-mail adresom – dobiva podatke o lozinki s kojom se prijavljuje u aplikaciju.

Prijava
(prioritet
3)



Nakon prijave otvara se glavnom sučelje zajedno sa izbornikom u gornjem desnom uglu. U izborniku se korisniku nudi na odabir 5 opcija: mogućnost editiranja profila, mogućnost povratka na naslovnu stranicu, mogućnost pogleda informacija o razvojnom timu, mogućnost odjave iz aplikacije i mogućnost zatvaranja aplikacije. Glavno sučelje biti će objašnjeno u Mockapu broj 7.

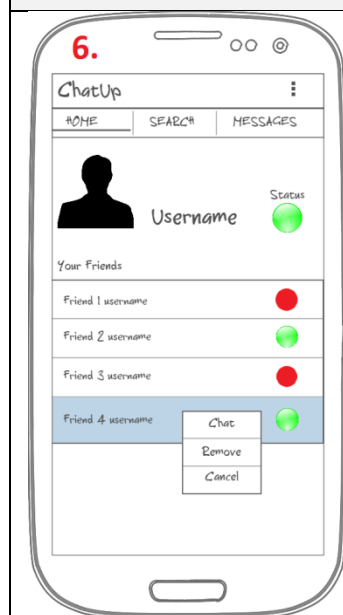
Izbornik
(prioritet
8)

Odjava
(prioritet
4)



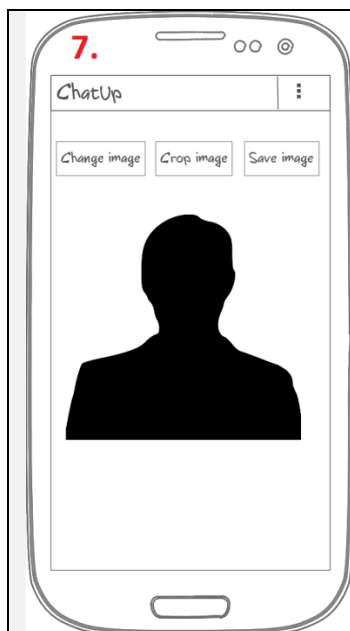
Odabirom „Edit profile“ u izborniku otvara se novi prozor koji korisniku nudi mogućnost editiranja vlastitih korisničkih postavki. Korisnik kao "hint" vidi svoje trenutne postavke. Promjenom određene i klikom na gumb „Save“ spremaju se uređene postavke korisničkog profila. Moguće je izmijeniti sljedeće: korisničko ime, spol i lozinku. Kod izmjene lozinke potrebno je dva put unijeti novu lozinku radi provjere korektnosti unosa.

Uređivanje profila (prioritet 13)



U glavnom prozoru aplikacije (Home) vidljiva je slika profila prijavljenog korisnika aplikacije, njegov status i njegovi prijatelji koje ima u listi prijatelja zajedno sa pripadnim statusom svakog prijatelja (online, busy, away..). Držanjem pritiska na nekog od prijatelja isti se selektira i otvara se izbornik sa tri mogućnosti. Otvaranje razgovora sa tim korisnikom (*Chat*), brisanje korisnika sa liste prijatelja (*Remove*) i odustajanje od akcije (*Cancel*).

Profilna stranica (prioritet 5)

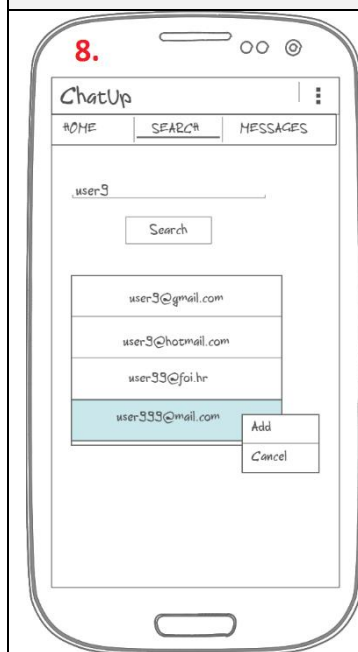


Nakon dugog klika na sliku iz glavnog prozora aplikacije (Home) otvara se sljedeći prozor sa sljedećim mogućnostima:

- Promjena trenutne slike
- Rezanje učitane slika
- Spremanje nove slike

Nakon što je korisnik odabrao novu sliku, i po potrebi je izrezao, dobiva mogućnost pohranjivanja iste, gdje se nova verzija ažurira u Home izborniku aplikacije.

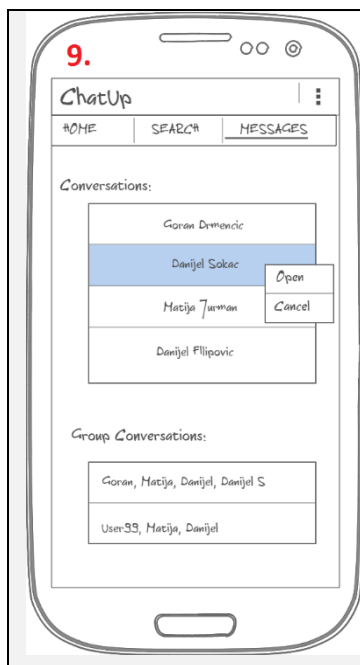
Promjena slike profila (prioritet 5)



Odabirom druge kartice (*Search*) korisniku je omogućeno pretraživanje svih registriranih korisnika. Upisom određene riječi i pritiskom na gumb „*Search*“ otvara se lista svih korisnika čija e-mail adresa započinje sa ranije navedenom riječi. Nakon što se otvori lista, može se dugim pritiskom na neku od e-mail adresa registriranih korisnika odabrati jedna od dvije opcije:

- *Add* – dodavanje korisnika u listu prijatelja
- *Cancel* – odustajanje od akcije

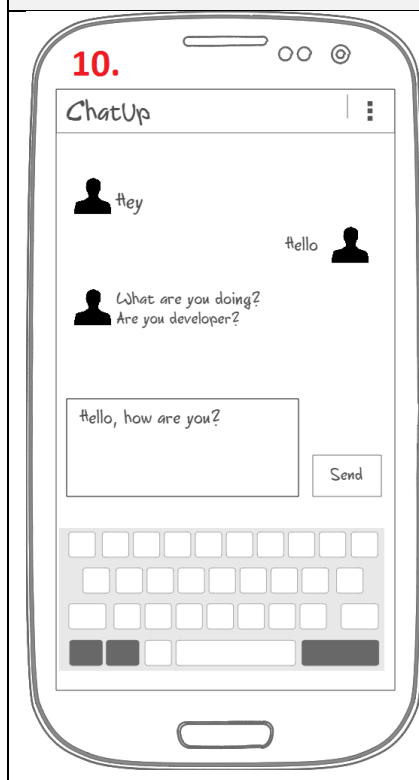
Pretraživanje i dodavanje drugih korisnika (prioritet 6)



Odabirom treće kartice (*Messages*) korisniku je omogućeno pregledati sve povijesti razgovora sa svim sudionicima s kojima je vodio razgovor. Također omogućen mu je i uvid u povijest razgovora grupnih poruka kod kojih je u razgovoru sudjelovalo tri ili više osoba. Dugim pritiskom na pojedini razgovor otvara se mini izbornik za navedeni selektirani razgovor sa dvije opcije:

- *Open* – otvaranje razgovora
- *Cancel* – odustajanje od akcije

Povijest
razgovora
(prioritet
10)



Odabirom opcije „*Open*“ u ranije objašnjenom izborniku otvara se odabrani razgovor i moguće je započeti razgovor s tim korisnikom ili korisnicima (ukoliko se radi o grupnome razgovoru). Korisniku je omogućeno slanje i primanje poruka. Uz svaku poslanu ili primljenu poruku prikazana je slika korisnika koji je tu poruku poslao. Dakle, unosom rečenice i pritiskom na tipku „*Send*“ poruka se šalje sugovorniku ili sugovornicima razgovora, ovisno o tome radi li se o dvočlanom ili grupnom razgovoru.

Razgovori
s drugim
korisnicima
(prioritet
7)

Grupni
razgovori
(prioritet
9)

4. Konfiguracija servera i baze podataka

Za bazu podataka odlučili smo koristiti CouchDB¹. CouchDB baza podataka sprema podatke u obliku JSON-a (engl. JavaScript Object Notation) i spada u NoSQL baze podataka. Pristup podacima i upiti mogu se odvijati preko web preglednika putem HTTP protokola. Indeksiranje, spajanje i transformiranje dokumenata vrši se s JavaScript programskim jezikom. CouchDB odlično radi s modernim web i mobilnim aplikacijama. Također podržava inkrementalnu replikaciju za distribuciju podataka ili aplikaciju putem mreže.

CouchDB dokument je JSON objekt koji se sastoji od imenovanih polja. Vrijednosti polja mogu biti tekstualni, numerički, datumski pa čak i posložena lista ili mapa. Za rješavanje problema dodavanja strukture natrag u polustrukturirane podatke, CouchDB integrira model pogleda koristeći JavaScript za opis. Pogledi (engl. Views) su načini agregiranja i izvještavanja o dokumentima unutar baze.

Da bi baza bila dostupna svima, potrebno je CouchDB smjestiti na server. U ovom projektu koristio se jedan od servera na Digital Ocean² stranici. Nakon izrade droplet-a dobije se javna IP adresa preko koje se može dohvaćati baza.

Slika 1 prikazuje korisničko sučelje na stranici DigitalOcean-a.

Img	Name	IP Address	Memory	Disk	Region
	ubuntuNew	104.236.58.50	512 MB	20 GB	NYC3

Slika 1 Podaci droplet-a korištenog na projektu

Nakon instalacije baze potrebno je kreirati bazu i njezine dokumente. Od dokumenata u projektu koristit će se dvije vrste dokumenata za pohranu podataka. Prvi dokument sadrži podatke o pojedinom korisniku dok drugi dokument bi sadržavao pohranu razgovora između dva ili više korisnika.

¹ Apache CouchDB - <http://couchdb.apache.org/>

² DigitalOcean, Simple Cloud Hosting - <https://www.digitalocean.com/>

Izgled dokumenta koji sadrži podatke o korisniku izgleda ovako:

```
{
  "_id": "mirko@mail.hr",
  "_rev": "2-d765a93614eeeca829c0c859f3c47482",
  "mail": "mirko@mail.hr",
  "username": "mirko",
  "password": "mirko123",
  "status": "offline",
  "gender": "m",
  "dateOfBirth": "2011-02-02",
  "friends": ["darko"],
  "type": "user",
  "profilePicture": "base64Kod"
}
```

Spremanje razgovora izgleda ovako:

```
{
  "_id": "7e7d8b9abd1348a8375f0c429e00547b",
  "_rev": "4-9e917920d5de94e950d0d0a6a65f10c3",
  "chat": [
    {
      "sender": "dfilipov",
      "text": "Ovaj razgovor je kreiran preko aplikacije.",
      "timeSend": "1450895076900",
      "location": "",
      "type": "text"
    },
    {
      "sender": "mjurman",
      "text": "odlicno",
      "timeSend": "1450949719208",
      "location": "",
      "type": "text"
    },
    {
      "sender": "mjurman",
      "text": "tu je ok",
      "timeSend": "1450979747175",
      "location": "",
      "type": "text"
    }
  ],
  "participants": "dfilipov@foi.hr,mjurman@foi.hr",
  "type": "message"
}
```

Također baza sadrži dosta pogleda koje koriste web servisi.

Da bi CouchDB znao da je dokument ustvari pogled na bazu, njegov *_id* mora započeti sa *_design/*, u nastavku je prikazan dio programskog kôda našeg dokumenta za pogled:

```
{
  "_id": "_design/view",
  "_rev": "15-962d2830e710e1b0b93caf9336738b4a",
  "views": {
    "getUserIDs": {
```

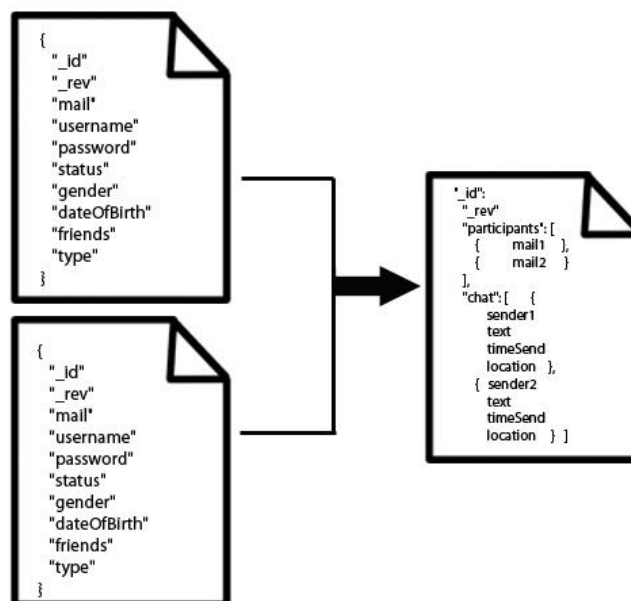
```

    "map": "function(doc) { if (doc.type && doc.type == 'user') emit(doc._id); }"
  },
  "getMailAndPasswords": {
    "map": "function(doc) { if(doc.mail && doc.password) { emit(doc._id, { Mail:
doc.mail, Lozinka: doc.password}) } } "
  },
  "getOnlineUsers": {
    "map": "function(doc) { if (doc.type && doc.type == 'user' && doc.status ==
'online') emit(doc); }"
  },
  "getOfflineUsers": {
    "map": "function(doc) { if (doc.type && doc.type == 'user') emit(doc); }"
  },
  "getAllMessages": {
    "map": "function(doc) { if (doc.participants.toLowerCase().indexOf('darko@mail.hr')
>= 0) { if (doc.type && doc.type == 'message' && doc.participants) emit(doc._id, doc.chat);}
}"
  },
  "getRegisteredUsers": {
    "map": "function(doc){if(doc.type == 'user' || doc.type == 'admin') emit(doc);}"
  },
  "getUserData": {
    "map": "function(doc) { if (doc.type && doc.type == 'user') { var user = new
Object(); user.username = doc.username; user.mail = doc.mail; user.gender = doc.gender;
user.dateOfBirth = doc.dateOfBirth; user.status = doc.status; user.friends = doc.friends;
emit(null, user); } }"
  }
}
}

```

Vidimo neke od pogleda iz baze (getUserIDs, getMailAndPasswords, getOnlineUsers, getOfflineUsers, ...). Sami broj pogleda u bazi daleko je veći.

Slika 2 grafički prikazuje sadržaj glavnih dokumenata i njihovu vezu.



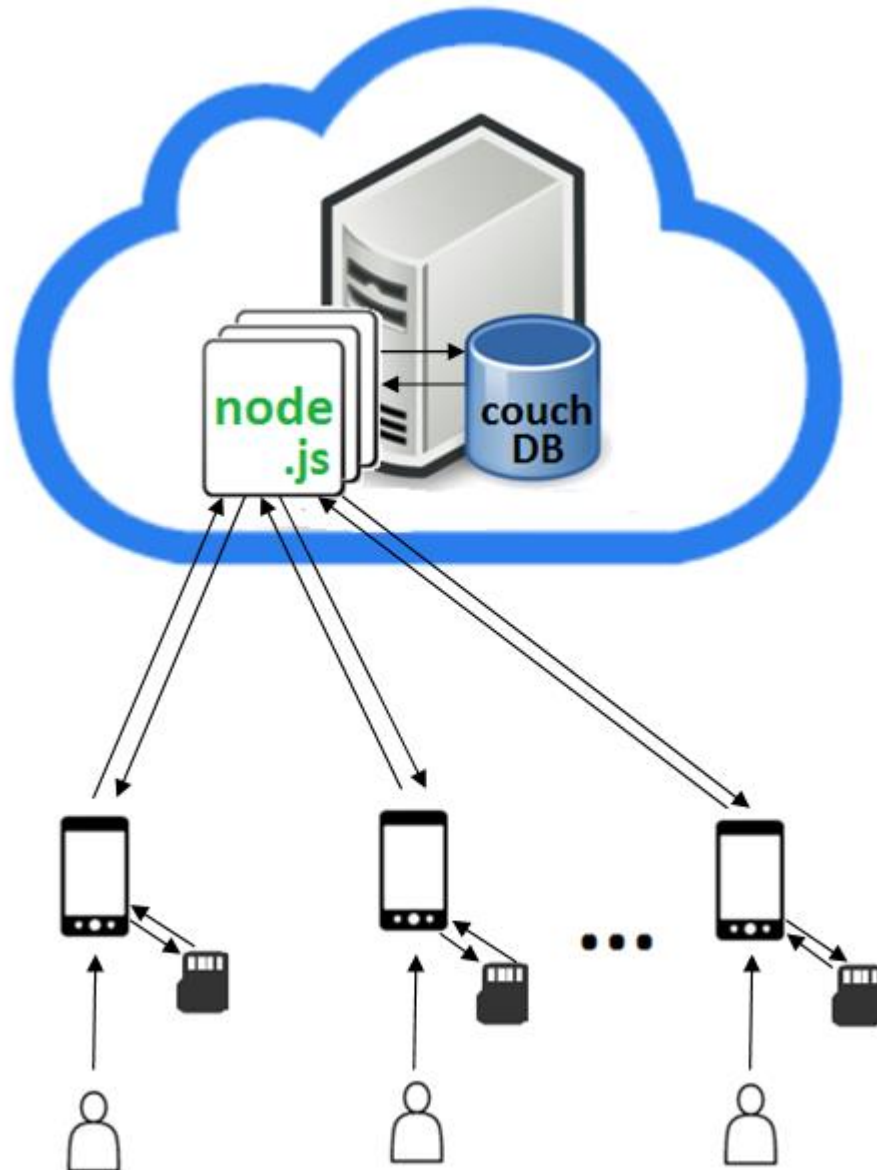
Slika 2 Grafički prikaz dokumenata i njihovih veza

Dva lijeva dokumenta predstavljaju pojedinog registriranog korisnika. Kada započnu razgovor kreira se poseban dokument za njihov razgovor u koji se spremaju sve poruke poslane između tih korisnika. Razgovor može sadržavati dva ili više sudionika.

Da bi baza i aplikacija na mobitelu mogli međusobno komunicirati potrebno je izraditi web servise. Oni će detaljno biti razrađeni u sljedećem poglavlju.

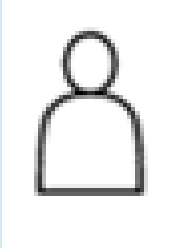
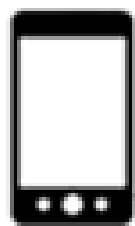


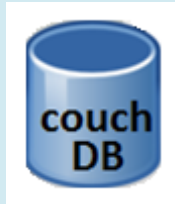
5. Arhitektura sustava

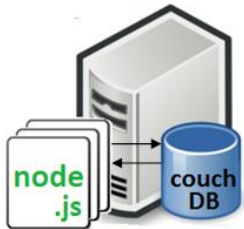
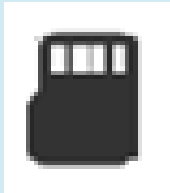
Na sljedećoj slici biti će prikazana arhitektura sustava naše aplikacije te će biti objašnjeni pojedini simbole i veze između njih.



Slika 3. Arhitektura sustava aplikacije

5.1. Objašnjenje simbola

Izgled simbola	Naziv	Opis
	Korisnik aplikacije	Podrazumijeva pojedinog korisnika aplikacije. Korisnik aplikacije može biti klijent koji je naručio aplikaciju ali i svi korisnici koji su aplikaciju skinuli sa Google Play-a.
	Aplikacija	Podrazumijeva dio programskog proizvoda kojim može rukovati korisnik. To uključuje sve aktivnosti unutar same aplikacije kojima može navigirati korisnik uključujući različite slike zaslona, obrasce za unos, pretraživanje, mijenjanje, odabir i sl.
	"Cloud"	Podrazumijeva koncept koji nudi pristup (u našem slučaju podacima u bazi i web servisima) uz karakteristike kao što su centralizacija, stalna dostupnost, kontroliran korisnički pristup, sigurnost itd.
	Node.js web servisi	Predstavlja web servise smještene na "Cloud" serveru gdje su konstantno pokrenuti. Napisani su u obliku node.js skripti. Služe kao posrednici između aplikacije i baze podataka, a svaki ima točno određenu funkciju odnosno zadaću.
	CouchDB baza podataka	CouchDB baza podataka smještena na "Cloud" serveru. U njoj su zapisani svi podaci potrebni za nesmetan rad aplikacije, koji se konstantno mijenjaju ovisno o korisnikovoj interakciji s aplikacijom. Također, baza podataka sadrži i poglede koje koriste različiti node.js web servisi.

	<p>Server na kojem se nalaze web servisi i baza podataka</p>	<p>Server sa stranice "<i>DigitalOcean</i>". IP adresa mu je 104.236.58.50. Na serveru se nalazi CouchDB baza podataka i node.js web servisi koji su pokrenuti u node modulu "<i>forever</i>". Od ostalih node modula instalirani su: <i>body-parser</i>, <i>express</i>, <i>nano</i> i <i>nodemailer</i>.</p>
	<p>Unutarnja memorija mobilnog uređaja</p>	<p>Predstavlja unutarnju memoriju mobilnog uređaja svakog pojedinog korisnika. U unutarnju memoriju zapisuje se profilna slika korisnika u .jpeg formatu.</p>

Za rad aplikacije koristi se *Digital Ocean* server koji je konstantno "*online*" i na kojemu se nalazi CouchDB baza podataka te svi node.js web servisi koje koristi aplikacija. Svi web servisi pokrenuti su u node modulu *forever*, što znači da se konstantno "vrte" na serveru. Korisnik prilikom pokretanja aplikacije dobiva mogućnost prijave ili registriranja u sustav. Nakon unosa podataka i odabira željenog gumba, ovisno o odabiru, node.js web servis čita podatke iz baze (prijava) ili ih zapisuje i kreira novi dokument u bazi (registracija). Nakon uspješne prijave, aplikacija funkcioniра na način da se svaka njena funkcionalnost realizira korištenjem različitih node.js web servisa koji se nalaze na serveru i konstantno imaju interakciju sa aplikacijom i podacima iz couchDB baze podataka. Za pohranu slike korisnika, aplikacija ima interakciju sa unutarnjom memorijom mobilnog uređaja svakog pojedinog korisnika.

6. Web servisi

Web servisi služe kao posrednici između baze podataka i android aplikacije. Web servisi su realizirani pomoću Node.js tehnologije za izradu *backend* Web aplikacija.

6.1. server.js

Ovaj Web servis je glavni Web servis koji će se vrtiti na poslužitelju. Svi ostali Web servisi su zapisani u zasebnim JavaScript datotekama kao funkcije. U ovoj datoteci se te funkcije uključuju i pridružuju se odgovarajućoj ruti preko koje se određeni Web servis može zatražiti.

6.2. register.js

Web servis za registraciju novog korisnika. Prima POST parametre koji sadrže sljedeće informacije o korisniku: korisničko ime, e-mail adresu, spol, datum rođenja i lozinka. Web servis će zatražiti pogled iz CouchDB baze podataka koji će izlistati popis svih registriranih korisnika. Ako ne postoji korisnik sa istim korisničkim imenom koji je poslan Web servisu, tada se novi korisnik zapisuje u bazu podataka.

6.3. log_in.js

Web servis za prijavu korisnika služi za prijavljivanje korisnika na aplikaciju. Parametri koje prima su e-mail korisnika te njegova lozinka. Nakon toga se pokreće pogled iz baze koji vraća sve neprijavljene korisnike i provjerava odgovaraju li parametri i je li korisnik već prijavljen ili nije. Ako je sve uredu onda promijeni status korisnika na „online“ i tako je korisnik prijavljen.

6.4. log_out.js

Web servis za odjavu korisnika vrlo je jednostavan. U njemu se provjera da li je u bazi korisnik koji se želi odjaviti zapravo online, te ako je, web servis mijenja polje *status* logiranog korisnika u offline. Web servis kao parametar prima ID logiranog korisnika.

6.5. registeredUsers.js

Web servis za dohvat svih registriranih korisnika. Koristi se pogled na bazu te se svi korisnici spremaju u odgovor koji je JSON objekt koji se sastoji od *status* i *message*. U

message dijelu je sadržana lista svih korisnika koji su upisani u bazi, a ukoliko dođe do greške u tom dijelu je ispisana poruka greške.

6.6. get_messages.js

Web servis za dohvat razgovora između sudionika. Prima e-mail korisnika kao parametar te na temelju toga filtrira razgovore. Vraća listu razgovora koji se sastoje od poruka unutar tog razgovora i osnovne informacije svakog sudionika razgovora.

6.7. createConversation.js

Web servis za kreiranje razgovora između dva korisnika. Prima e-mail adrese dvaju korisnika kao parametre.

6.8. sendMessage.js

Web servis koji sprema poruku u odgovarajući razgovor. Kao parametre prima ID razgovora (na temelju kojeg se filtrira razgovor), korisničko ime pošiljaoca, tekst poruke i tip poruke. Web servis također toj poruci dodaje vremensku oznaku prije pohrane u bazu podataka.

6.9. getNewMessages.js

Web servis koji dohvaća nove poruke iz baze podataka na temelju vremenske oznake zadnje poruke koja je primljena u aplikaciji. Kao parametre prima ID razgovora i vremensku oznaku zadnje poruke iz aplikacije.

6.10. save_profile_pic.js

Web servis sprema profilnu sliku korisnika u bazu kao string koji predstavlja base64 format same slike. Aplikacija na mobitelu kodira i dekodira sliku u/iz base64 format. Sve slike i dodatni dokumenti (engl. *attachments*) u CouchDB bazi podataka se spremaju u base64 formatu.

6.11. addFriends.js

Ovaj web servis služi za dodavanje korisnika u listu prijatelja. Web servis provjerava postoje li korisnik koji je logiran i korisnik koji je odabran unutar aplikacije u bazi podataka.

Postojanje korisnika provjerava se u pogledu *"getRegisteredUsers"*. Ukoliko oba korisnika postoje, logiranom se korisniku ažurira polje *friendssa* ID-em odabranog korisnika. Također odabranom korisniku se ažurira polje *friendssa* ID-em logiranog korisnika. Prije nego li se promjene u bazi podataka izvrše, web servis provjerava ne postoji li već korisnik sa tim ID-em u listi prijatelja, te se time postiže provjera jedinstvenosti u listi prijatelja. Web servis kao parametre prima ID logiranog korisnika i ID odabranog korisnika.

6.12. editProfile.js

Ovaj web servis kao parametre prima ID logiranog korisnika, odnosno njegov e-mail, njegovo korisničko ime, spol i lozinku. Web servis najprije provjerava postoji li ID logiranog korisnika u bazi podataka, nakon čega vrši promjene u određenim poljima dokumenta tog korisnika u bazi podataka. Ovisno o korisnikovom odabiru u aplikaciji, web servis može promijeniti njegovo korisničko ime, njegov spol ili lozinku. Ovaj web servis koristi *"getRegisteredUsers"* pogled na bazu te se svi korisnici spremaju u odgovor koji je JSON objekt koji se sastoji od *"status"* i *"message"*.

6.13. getUserDataEditProfile.js

Ovaj web servis kao parametar prima ID logiranog korisnika, odnosno njegov e-mail. Za pretragu korisnika u bazi podataka koristi pogled na bazu *"getRegisteredUsers"*. Ukoliko je korisnik pronađen u polje *"usersDetails"* spremaju se podaci o trenutnom korisnikovom ID-u (njegovoj e-mail adresi), o njegovom korisničkom imenu, spolu i lozinki. Kao odgovor, web servis vraća aplikaciji objekt koji se sastoji od *"data"*, *"status"* i *"message"* dijela. U *"data"* dijelu zapisani su ranije opisani podaci o korisniku. U *"message"* dijelu zapisana je poruka o pogrešci ili poruka o uspješnom čitanju podataka. *"Status"* može biti 1 ili 0. Ukoliko je 0, web servis je uspješno dohvatio podatke, inače je došlo do pogreške.

6.14. registeredUsers2.js

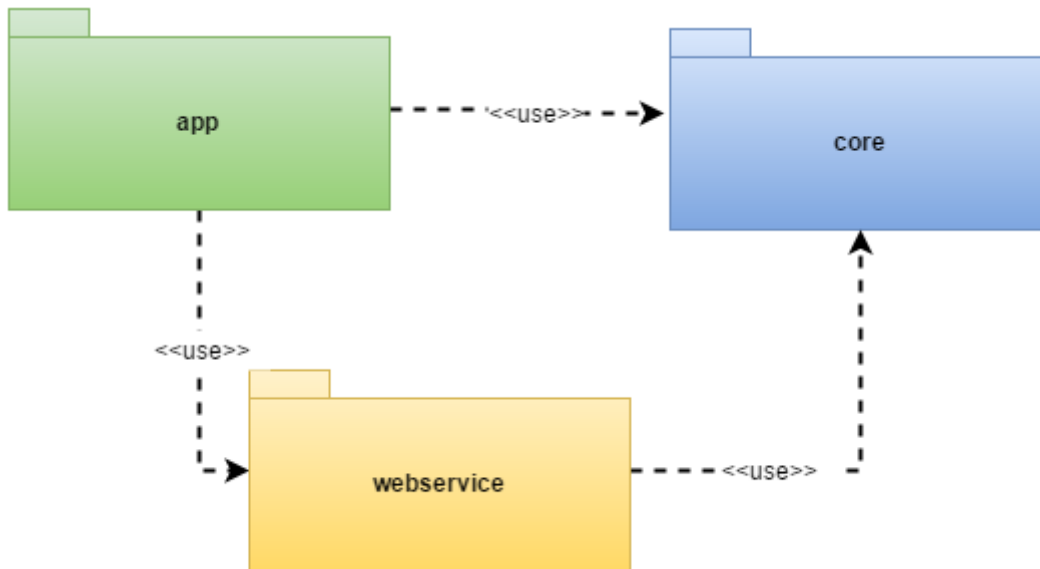
Ovaj web servis iz pogleda *"getRegisteredUsers"* čita podatke o svim registriranim korisnicima u bazi podataka, te za svakog korisnika pojedinačno u polje *"userDetails"* sprema podatke o njegovom ID-u, korisničkom imenu i statusu. Kao odgovor, web servis vraća aplikaciji objekt koji se sastoji od *"data"*, *"status"* i *"message"* dijela. U *"data"* dijelu zapisani su svi podaci koje će koristiti aplikacija u fragmentu/tab-u "Search". Web servis služi

za prikaz svih registriranih korisnika, kako bi korisnik kasnije u aplikaciji mogao pretraživati ili odabrati svakog pojedinačno i njime baratati (dodati u listu prijatelja i sl.).

6.15. forgotPassword.js

Ovaj web servis služi za oporavak zaobravljene lozinke. Kao parametar prima ID (odnosno e-mail) korisnika koji je zaboravio svoju lozinku. Ukoliko je ID korisnika pronađen u pogledu "*getRegisteredUsers*", što ujedno znači da korisnik postoji u bazi podataka i da je registriran, na taj isti mail web servis prosljeđuje podatak o njegovoj lozinki.

7. Dijagram klasa

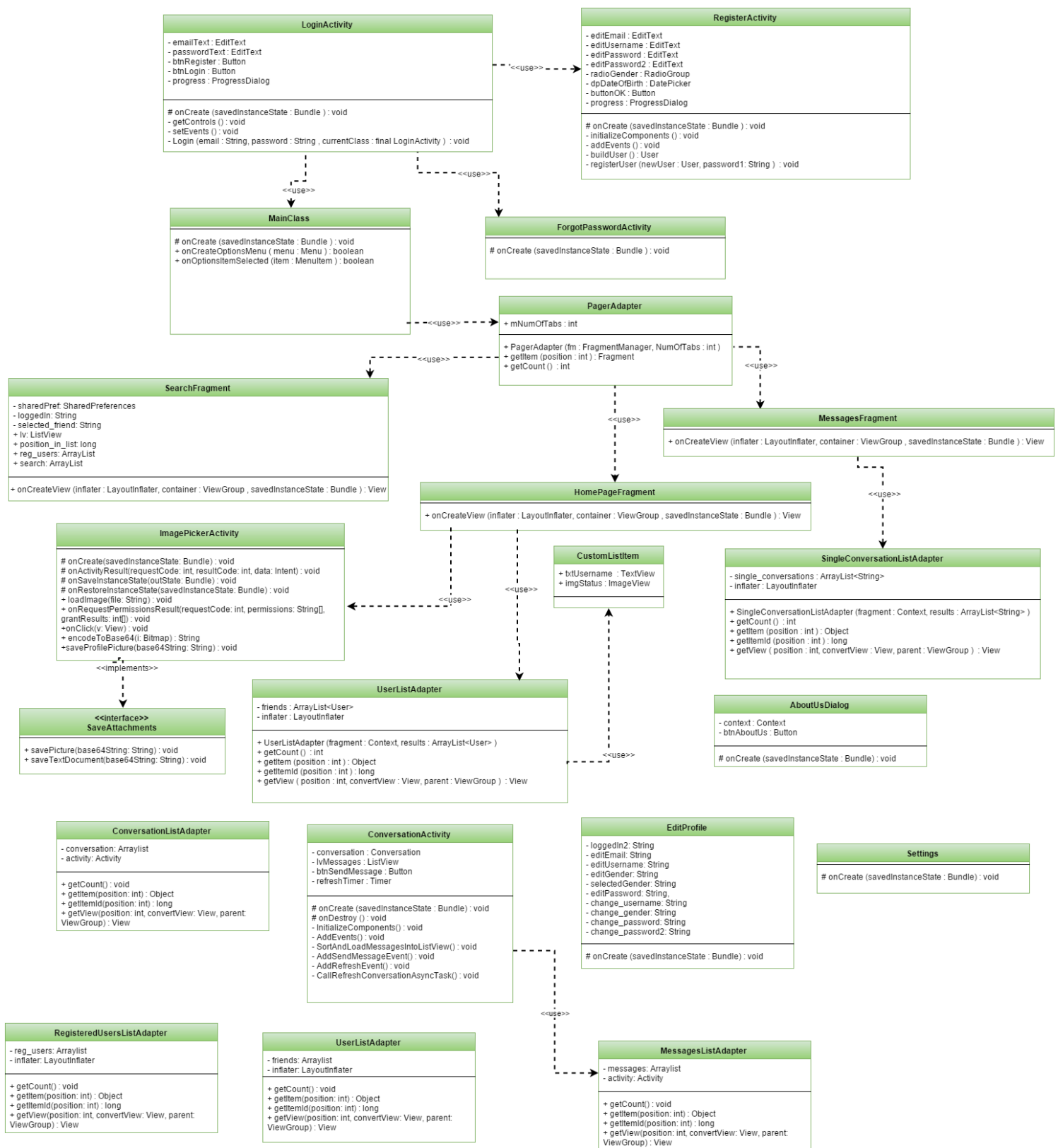


Slika 5.1. Package diagram – najviša razina dijagrama klasa

Na slici 5.1. prikazani su odnosi između modula aplikacije. Ovaj prikaz je korišten zbog lakšeg i preglednijeg prikaza dijagrama klasa aplikacije. Strelice između modula predstavljaju veze između klasa u detaljnijem prikazu. Strelica na slici 5.1. može predstavljati jednu vezu ili više veza između modula koje povezuje.

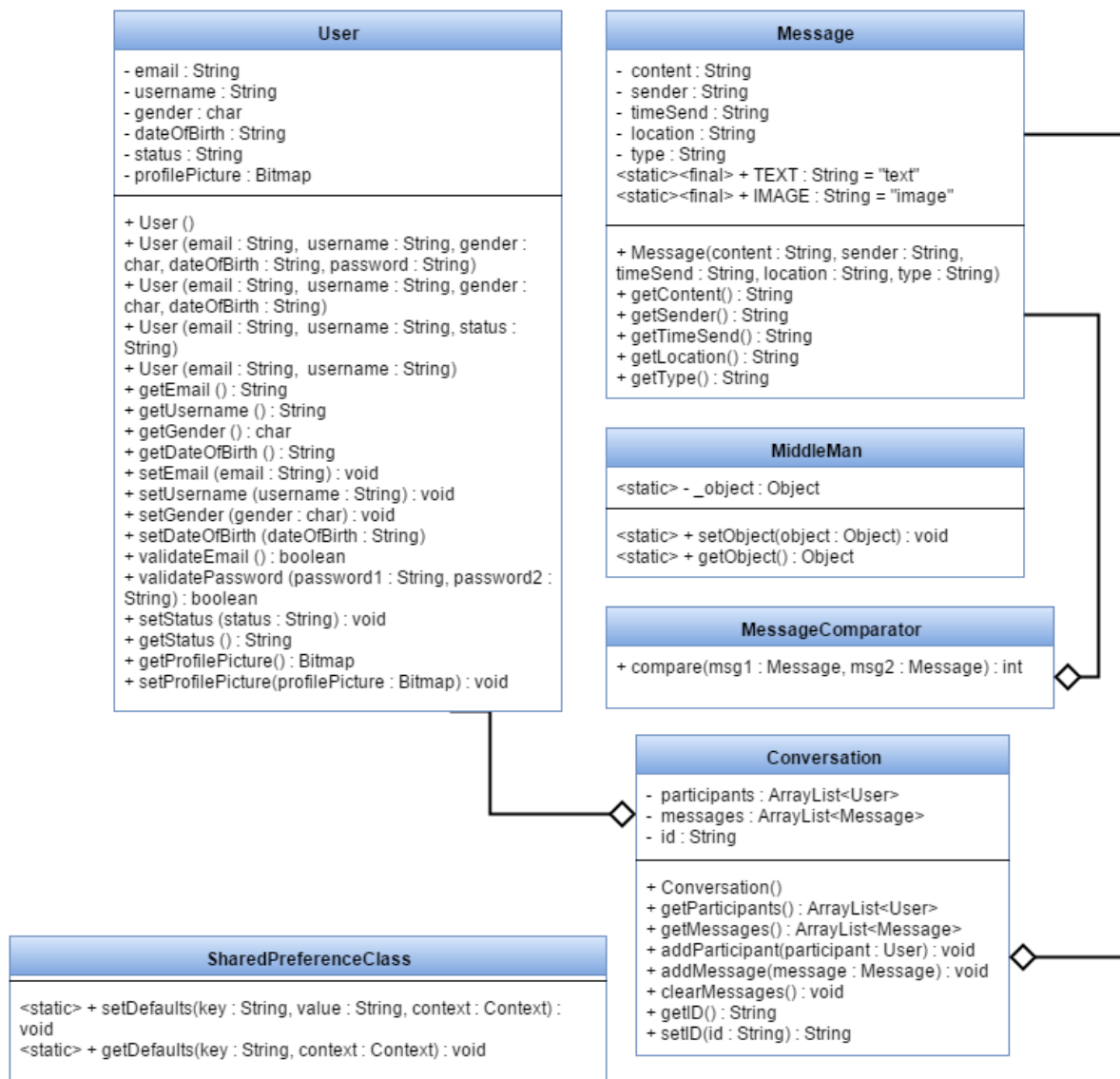
7.1. app modul

Na slici 5.2., na slijedećoj stranici, se vide sve klase koje se nalaze u app modulu. Klase su većinom *activity*-ji odnosno fragmenti. Iz dijagrama se primijeti da je prijava početni prikaz kada se aplikacija upali pa se onda inicijaliziraju ostali kako ih poziva zaslon za prijavu.



Slika 5.2. Dijagram klasa *app* modula

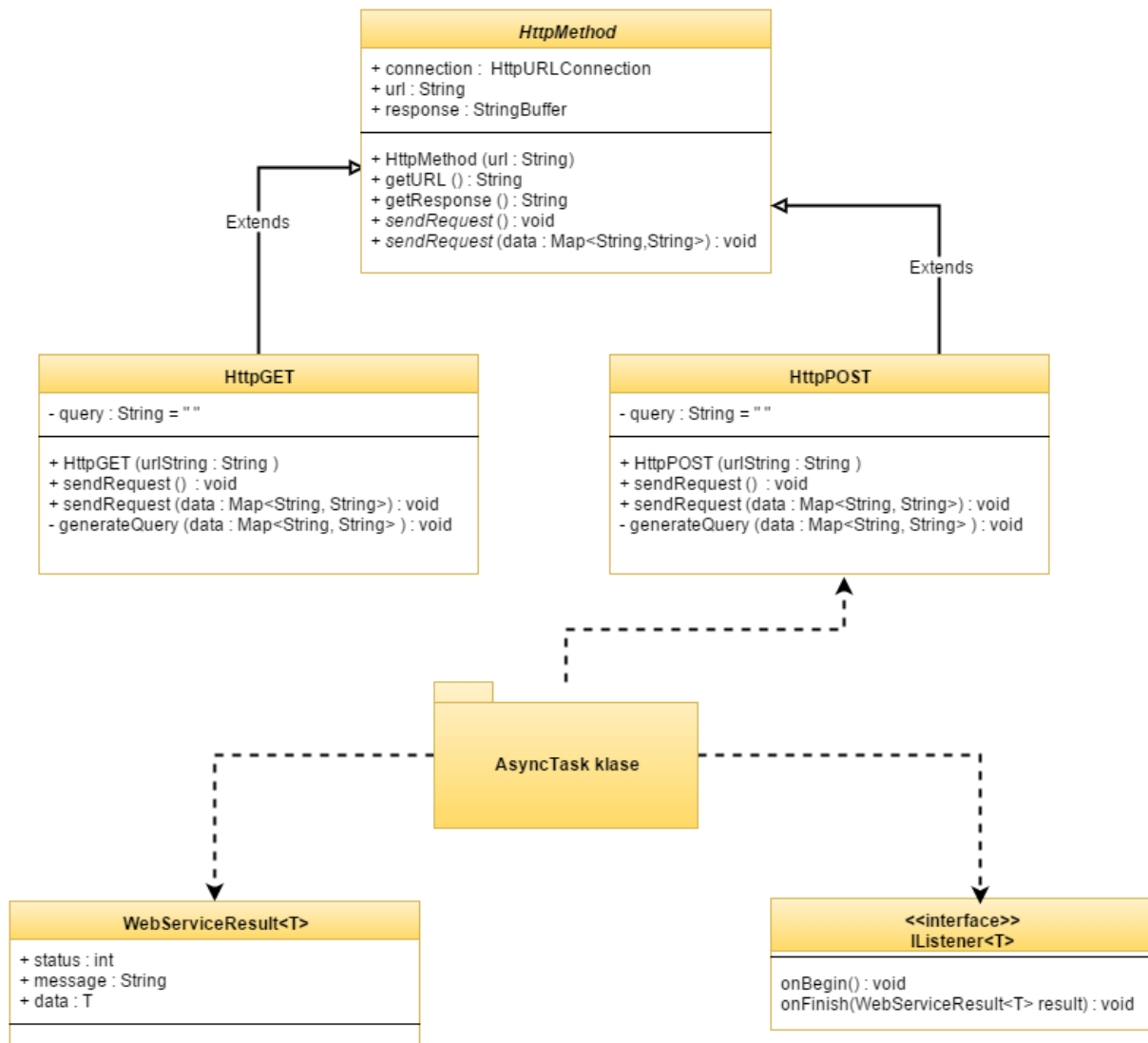
7.2. core modul



Slika 5.3. Dijagram klasa *core* modula

User klasa se prvenstveno koristi za prikaz korisnika u aplikaciji, ali i koristi za slanje parametara na web servis. Uočava se da ima dvije samostalne klase a to su *SharedPreferencesClass* koja služi za spremanje i dohvaćanje *SharedPreferences*-a, a druga klasa je *MiddleMan* koja služi za slanje podataka među fragmentima i aktivnostima. Klase *Message* i *Conversation* su klase koje omogućuju funkcioniranje samog dopisivanja pri čemu klasa *Message* vodi brigu o porukama, a klasa *Conversation* vodi brigu o razgovorima. Sve je to prikazano na slici 5.3.

7.3. webservice modul



Slika 5.4. Dijagram klasa *webservice* modula

Webservice modul se sastoji od klasa koje služe za komunikaciju sa web servisima na serveru koje su prikazane paketom *AsyncTask klase* kako bi prikaz dijagrama bio jasniji. Sve klase iz *AsyncTask klase* paketa će biti prikazane u nastavku. **Bitno je napomenuti da svaka klasa iz navedenog paketa ima iste veze kao što su prikazane sa paketom na slici 5.4.** Glavna zadaća im je dobiti i zatražiti podatke ovisno o zadaći koju oni rade. Svaki naziv klase ujedno i objašnjava njezinu funkciju. Na dijagramu klasa se vidi da se sučelje implementira u trima klasama sa sličnom strukturom.

SendMessageAsync - conversation : Conversation - message : Message - listener : IListener<Message> + SendMessageAsync(conversation : Conversation, message : Message, listener : IListener<Message>) # onPreExecute() : void # doInBackground(params : void) : String # onPostExecute(result : String) : void	LogoutAsync - email : String - password : String - listener : IListener<Void> + LogoutAsync(email : String, listener : IListener<Void>) # onPreExecute() : void # doInBackground(params : void) : String # onPostExecute(result : String) : void	LoginAsync - email : String - password : String - listener : IListener<Void> + LoginAsync(email : String, password : String, listener : IListener<Void>) # onPreExecute() : void # doInBackground(params : void) : String # onPostExecute(result : String) : void	RegisterAsync - newUser : User - password : String - listener : IListener<Void> + RegisterAsync(newUser : User, password : String, listener : IListener<Void>) # onPreExecute() : void # doInBackground(params : void) : String # onPostExecute(result : String) : void	RefreshConversationAsync - chatID : String - timestamp : String - listener : IListener<ArrayList<Message>> + RefreshConversationAsync(chatID : String, timestamp : String, listener : IListener<ArrayList<Message>>) # onPreExecute() : void # doInBackground(params : void) : String # onPostExecute(result : String) : void
AddFriendAsync - email_prij : String - email_odab : String - listener : IListener<Void> + AddFriendAsync(email_prij : String, email_odab : String, listener : IListener<Void>) # onPreExecute() : void # doInBackground(params : void) : String # onPostExecute(result : String) : void	CreateConversationAsync - email1 : String - email2 : String - listener : IListener<Void> + CreateConversationAsync(email1 : String, email2 : String, listener : IListener<Void>) # onPreExecute() : void # doInBackground(params : void) : String # onPostExecute(result : String) : void	EditProfileAsync - email : String - username : String - gender : String - password : String - listener : IListener<Void> + EditProfileAsync(email : String, username : String, gender : String, password : String, listener : IListener<Void>) # onPreExecute() : void # doInBackground(params : void) : String # onPostExecute(result : String) : void	FetchMessagesAsync - user : User - listener : IListener<ArrayList<Conversation>> + FetchMessagesAsync(user : User, listener : IListener<ArrayList<Conversation>>) # onPreExecute() : void # doInBackground(params : void) : String # onPostExecute(result : String) : void	
FetchUserDataAsync - email : String - listener : IListener<User> + FetchUserDataAsync(email : String, listener : IListener<User>) # onPreExecute() : void # doInBackground(params : void) : String # onPostExecute(result : String) : void	ForgotPasswordAsync - email : String - listener : IListener + ForgotPasswordAsync(email : String, listener : IListener) # onPreExecute() : void # doInBackground(params : void) : String # onPostExecute(result : String) : void	FriendAsync - email : String - listener : IListener + FriendAsync(email : String, listener : IListener) # onPreExecute() : void # doInBackground(params : void) : String # onPostExecute(result : String) : void	GetDataEditProfileAsync - email : String - listener : IListener + FriendAsync(email : String, listener : IListener) # onPreExecute() : void # doInBackground(params : void) : String # onPostExecute(result : String) : void	SaveImageAsync - imageEncoded : String - mail : String - listener : IListener<Void> + SaveImageAsync(mail : String, imageEncoded : String, listener : IListener<Void>) # onPreExecute() : void # doInBackground(params : void) : String # onPostExecute(result : String) : void

Slika 5.5. Prikaz paketa *AsyncTask* klase

Slika 5.5. prikazuje sve klase koje se nalaze u paketu *AsyncTask klase* . Bitno je ponoviti da svaka ta klasa ima veze iste kao i paket na slici 5.4. Svaka klasa ima naziv koji ujedno dočarava njezinu funkciju. Svaka klasa koristi web servis kao posrednika između baze podataka i aplikacije. Web servis dohvaća podatke iz baze podataka i obrađuje ih te vraća klasi rezultat, a ona onda prikaže taj rezultat u aplikaciji.