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 repozirotij:

* <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 repozirotij:

* <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](#_Toc439208645)

[2. Use-Case Dijagram 2](#_Toc439208646)

[3. Konfiguracija servera i baze podataka 4](#_Toc439208647)

[4. Arhitektura sustava 8](#_Toc439208648)

[5. Web servisi 10](#_Toc439208649)

[5.1. server.js 10](#_Toc439208650)

[5.2. register.js 10](#_Toc439208651)

[5.3. log\_in.js 10](#_Toc439208652)

[5.4. log\_out.js 10](#_Toc439208653)

[5.5. registeredUsers.js 10](#_Toc439208654)

[5.6. get\_messages.js 11](#_Toc439208655)

[5.7. createConversation.js 11](#_Toc439208656)

[5.8. sendMessage.js 11](#_Toc439208657)

[5.9. getNewMessages.js 11](#_Toc439208658)

[5.10. save\_profile\_pic.js 11](#_Toc439208659)

[5.11. addFriends.js 11](#_Toc439208660)

[5.12. editProfile.js 12](#_Toc439208661)

[5.13. getUserDataEditProfile.js 12](#_Toc439208662)

[5.14. registeredUsers2.js 12](#_Toc439208663)

[5.15. forgotPassword.js 13](#_Toc439208664)

[6. Dijagram klasa 14](#_Toc439208665)

[6.1. app modul 14](#_Toc439208666)

[6.2. core modul 16](#_Toc439208667)

[6.3. webservice modul 17](#_Toc439208668)

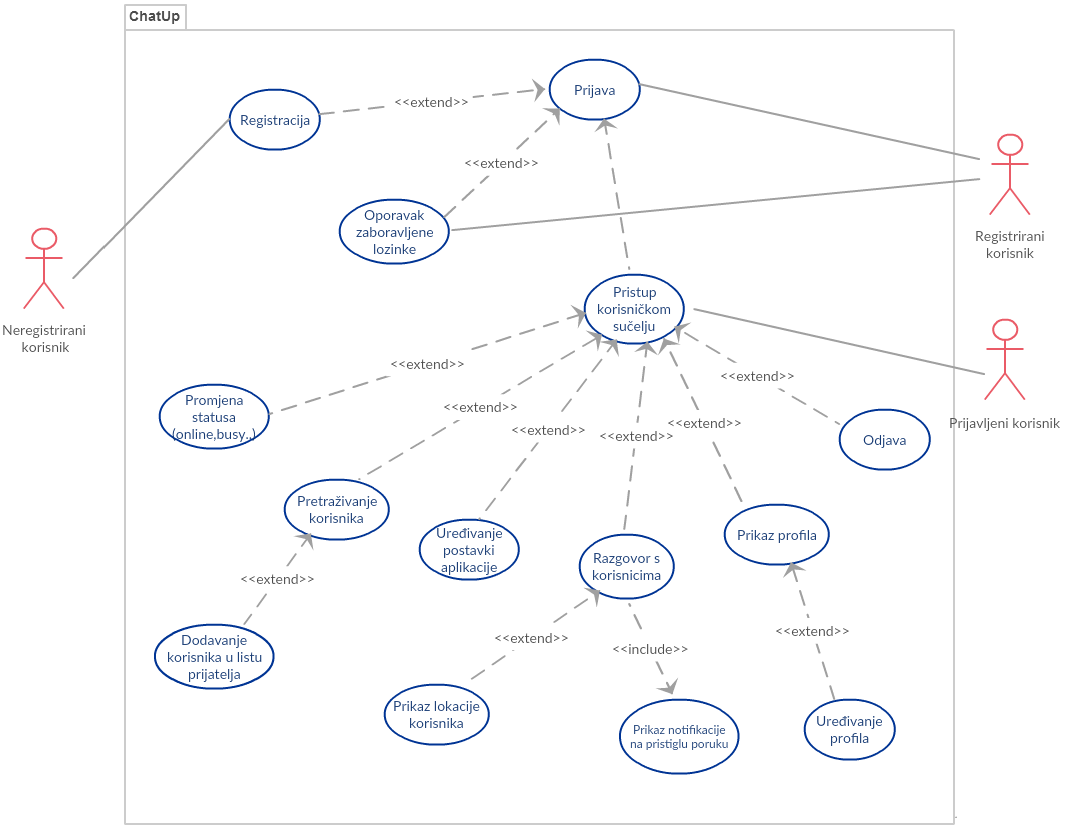
1. Uvod

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.

1. 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]

1. Konfiguracija servera i baze podataka

Za bazu podataka odlučili smo koristiti CouchDB[[1]](#footnote-1). 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 Ocean[[2]](#footnote-2) 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.



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.

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 poglede:

**{**

**"\_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.



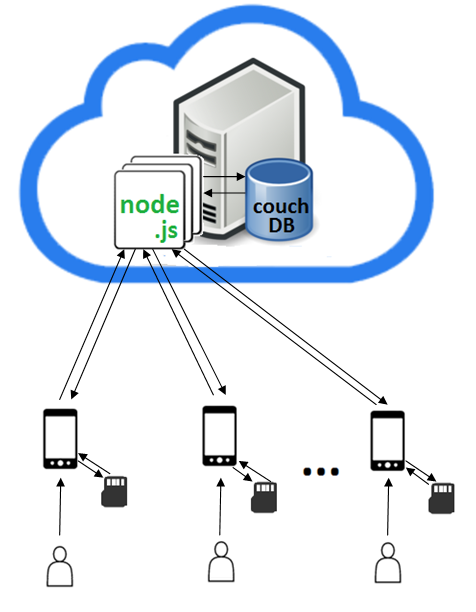
Slika 2 Grafički prikaz dokumenata i njihovih veza

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.

1. Arhitektura sustava

Na sljedećoj slici prikazat ćemo arhitekturu sustava naše aplikacije te objasniti pojedine simbole i veze između njih.



**Slika 4. Arhitektura sustava aplikacije**

Za funkcioniranje aplikacije korišten je *Digital Ocean* server koji je konstantno "*online*" i na kojemu se nalazi CouchDB baza podataka te svi node.js web servisi koje koristi aplikacija. 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 funkcionira 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.

1. 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.

* 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.

* 1. register.js

Web servis za registraciju novog korisnika. Prima POST parametre koji sadrže sljedeće informacije o korsniku: 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.

* 1. 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.

* 1. 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 korisnikau offline. Web servis kao parametar prima ID logiranog korisnika.

* 1. 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.

* 1. 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.

* 1. createConversation.js

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

* 1. sendMessage.js

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

* 1. 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.

* 1. 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.

* 1. 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 *friends* sa ID-em odabranog korisnika. Također odabranom korisniku se ažurira polje *friends* sa 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.

* 1. 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"*.

* 1. 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.

* 1. 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.).

* 1. 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.

1. 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.

* 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**

* 1. core modul



**Slika 5.3. Dijagram klasa*core* modula**

U *core* modulu se vidi da imamo dvije odvojene cjeline: jedna je *HttpMethod* i sve vezano uz HTTP, a druga se odnosi na objekt *User*. *User* klasa se prvenstveno koristi za prikaz korisnika u aplikaciji, ali i koristi za slanje parametara na web servis.

* 1. webservice modul



**Slika 5.4. Dijagram klasa*webservice* modula**

*Webservice* modul se sastoji od klasa koje služe za komunikaciju sa web servisima na serveru. Glavna zadaća im je dobaviti i zatražiti podatke ovisno o zadaći koju oni rade. Na dijagramu klasa se vidi da se sučelje implementira u trima klasama sa sličnom strukturom.

1. Apache CouchDB - http://couchdb.apache.org/ [↑](#footnote-ref-1)
2. DigitalOcean, Simple Cloud Hosting - https://www.digitalocean.com/ [↑](#footnote-ref-2)