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)
* <http://danijelfilipovic.github.io/> (JavaDocs)

Varaždin, 2016.

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)
* <http://danijelfilipovic.github.io/> (JavaDocs)

Mentor

Ivan Švogor, mag. inf.

Varaždin, siječanj 2016.

**Sadržaj**

[1. Uvod 1](#_Toc441782658)

[2. Use-Case Dijagram 2](#_Toc441782659)

[2.1. Korisnički zahtjevi 4](#_Toc441782660)

[2.1.1. Perspektiva proizvoda i ciljano tržište 4](#_Toc441782661)

[2.1.2. Korisničke priče 4](#_Toc441782662)

[2.2. Prototip aplikacije 6](#_Toc441782663)

[2.2.1. Slika prototipa aplikacije 6](#_Toc441782664)

[3. Objašnjenje svakog pojedinog Mockup-a 8](#_Toc441782665)

[4. Konfiguracija servera i baze podataka 13](#_Toc441782666)

[5. Arhitektura sustava 17](#_Toc441782667)

[5.1. Objašnjenje simbola 18](#_Toc441782668)

[6. Vanjske biblioteke korištene u projektu 20](#_Toc441782669)

[6.1. ImageFileSelector 20](#_Toc441782670)

[6.2. javamail-android 20](#_Toc441782671)

[6.3. Socket.io Client 20](#_Toc441782672)

[7. Web servisi 21](#_Toc441782673)

[7.1. server.js 21](#_Toc441782674)

[7.2. register.js 21](#_Toc441782675)

[7.3. log\_in.js 21](#_Toc441782676)

[7.4. log\_out.js 21](#_Toc441782677)

[7.5. registeredUsers.js 21](#_Toc441782678)

[7.6. get\_messages.js 22](#_Toc441782679)

[7.7. createConversation.js 22](#_Toc441782680)

[7.8. sendMessage.js 22](#_Toc441782681)

[7.9. getNewMessages.js 22](#_Toc441782682)

[7.10. save\_profile\_pic.js 22](#_Toc441782683)

[7.11. addFriends.js 22](#_Toc441782684)

[7.12. editProfile.js 23](#_Toc441782685)

[7.13. getUserDataEditProfile.js 23](#_Toc441782686)

[7.14. registeredUsers2.js 23](#_Toc441782687)

[7.15. forgotPassword.js 23](#_Toc441782688)

[7.16. addParticipantsToConversation.js 24](#_Toc441782689)

[7.17. get\_userData.js 24](#_Toc441782690)

[7.18. getFriends.js 24](#_Toc441782691)

[8. Dijagram klasa 25](#_Toc441782692)

[8.1. app modul 25](#_Toc441782693)

[8.2. core modul 27](#_Toc441782694)

[8.3. webservice modul 28](#_Toc441782695)

[9. Testovi 31](#_Toc441782696)

[9.1. Testovi prihvatljivosti 31](#_Toc441782697)

[9.1.1. Test prihvatljivosti za kreiranje razgovora 31](#_Toc441782698)

[9.1.2. Test prihvatljivosti za uređivanje profila 32](#_Toc441782699)

[9.1.3. Test prihvatljivosti za slanje poruke 33](#_Toc441782700)

[9.1.4. Test prihvatljivosti za funkcionalnost reklama 34](#_Toc441782701)

[9.2. Robotium testovi 35](#_Toc441782702)

[9.2.1. EditProfileRobotium 35](#_Toc441782703)

[9.2.2. SettingsRobotium 35](#_Toc441782704)

[9.2.3. HomePageFragmentRobotium 36](#_Toc441782705)

[9.2.4. LogoutRobotium 36](#_Toc441782706)

[9.2.5. AboutUsRobotium 37](#_Toc441782707)

[9.2.6. SearchFragmentRobotium 37](#_Toc441782708)

[10. Modularnost u aplikaciji 38](#_Toc441782709)

[10.1. Modularnost sa *FragmentBuffer* 38](#_Toc441782710)

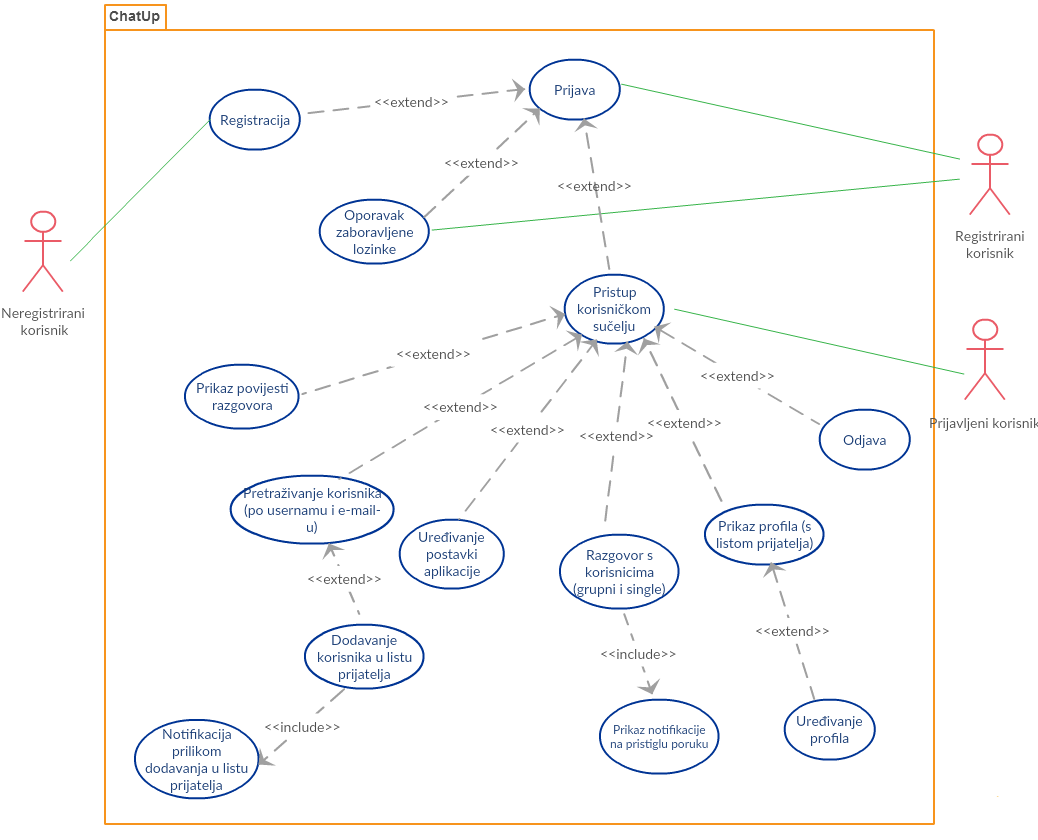
1. Uvod

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
* Vidjeti listu i status svojih prijatelja
* Pretraživati druge korisnike aplikacije (prema korisničkom imenu i e-mailu)
* Dodati drugog prijatelja u svoju listu prijatelja, što uključuje slanje notifikacije (korisnik koji je dodan dobiva notifikaciju o dodavanju)
* Urediti vlastite postavke aplikacije
* Razgovarati s drugim korisnicima. Razgovor može biti:
  + S jednim korisnikom
  + S više korisnika u istom razgovoru
* Vidjeti povijest vođenih razgovora
  1. Korisnički zahtjevi
     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.

* + 1. 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) |
| Pregled statusa | Ja kao prijavljeni korisnik želim imati mogućnost pregleda vlastitog statusa. Također želim vidjeti koji status ima svaki pojedini prijatelj iz vlastite liste prijatelja na početnoj profilnoj stranici. | Niski (15) |
| Zarada | Ja kao korisnik želim imati implementirane reklame unutar aplikacije. | Niski (15) |
| Testiranje | Ja kao korisnik želim da se aplikacija ponaša stabilno, robusno i pouzdano, te da nema bug-ova unutar nje. | Niski (15) |

* 1. 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.

* + 1. Slika prototipa aplikacije

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

**C:\Users\JurmanLap\Desktop\AIR predaja2\prototype.tif**

Slika 2.1 Prototip aplikacije ChatUp

1. Objašnjenje svakog pojedinog Mockup-a

|  |  |  |
| --- | --- | --- |
| Izgled pojedinog ekrana (mockup) | Objašnjenje | Korisnička priča (kraći naziv) |
| C:\Users\JurmanLap\Desktop\CHATUP FAZA 1\SVI MOCKAPI\editiranje\1 Log in.png | 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. | Prijava  (prioritet 3) |
| C:\Users\JurmanLap\Desktop\CHATUP FAZA 1\SVI MOCKAPI\editiranje\2 Register.png | Neregistrirani korisnik unosi redom sljedeće podatke u registracijski obrazac: e-mail adresa (u validnom formatu), korisničko ime, lozinku, potrvrdu 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. | Registracija  (prioritet 2) |
| C:\Users\JurmanLap\Desktop\CHATUP FAZA 1\SVI MOCKAPI\editiranje\3 Forgot password.png | 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) |
| C:\Users\JurmanLap\AndroidStudioProjects\docf2v29\Dokumentacija\Mockup\5 Menu - Copy.png | 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) |
| C:\Users\JurmanLap\AndroidStudioProjects\docf2v29\Dokumentacija\Mockup\6 Edit profile - Copy.png | 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. | Uređiva-nje profila  (prioritet 13) |
| C:\Users\JurmanLap\AndroidStudioProjects\docf2v29\Dokumentacija\Mockup\7 Home page ( Tab 1) - Copy.png | 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) |
| C:\Users\JurmanLap\AndroidStudioProjects\docf2v29\Dokumentacija\Mockup\12 Change image - Copy.png | 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) |
| C:\Users\JurmanLap\Desktop\CHATUP FAZA 1\SVI MOCKAPI\editiranje\8 Search (Tab2).png | 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 | Pretraži-vanje i dodava-nje drugih korisni-ka  (prirotet 6) |
| C:\Users\JurmanLap\Desktop\CHATUP FAZA 1\SVI MOCKAPI\editiranje\9 Messages (Tab3).png | 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) |
| C:\Users\JurmanLap\Desktop\CHATUP FAZA 1\SVI MOCKAPI\editiranje\10 Conversation.png | 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 korisinicima (prioritet7)  Grupni razgovori (prioritet9) |

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.



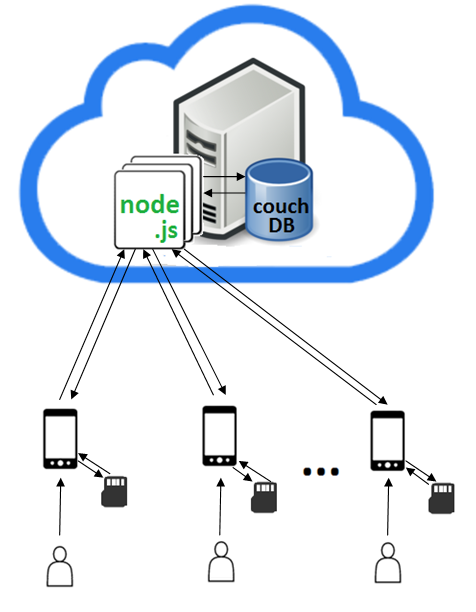
Slika 2 Grafički prikaz dokumenata i njihovih veza

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 jednom od sljedećih poglavlja.

1. 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**

* 1. Objašnjenje simbola

|  |  |  |
| --- | --- | --- |
| Izgled simbola | Naziv | Opis |
| C:\Users\JurmanLap\Desktop\7.png | 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. |
| C:\Users\JurmanLap\Desktop\5.png | 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. |
| C:\Users\JurmanLap\Desktop\1.png | "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. |
| C:\Users\JurmanLap\Desktop\2.png | 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. |
| C:\Users\JurmanLap\Desktop\3.png | 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. |
| C:\Users\JurmanLap\Desktop\4.png | Server na kojem se nalaze web servisi i baza podataka | Server sa stranice *"DigitalOcean".* 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 *"forever"*. Od ostalih node modula instalirani su*: body-parser*, *express*, *nano* i *nodemailer.* |
| C:\Users\JurmanLap\Desktop\6.png | Unutarnja memorija mobilnog uređaja | Predastavlja unutarnju memoriju mobilnog uređaja svakog pojedinog korisnika. U unutarnju memoriju zapisuje se profilna slika korisnika u .jpeg formatu. |

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 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. Vanjske biblioteke korištene u projektu

Ovdje će biti naglašene samo biblioteke koje nisu bile obrađivane na laboratorijskim vježbama. Od biblioteka koje su spomenute i korištene na laboratorijskim vježbama koristili smo Google Mobile Ads SDK za prikaz reklama i JUnit za testiranje aplikacije.

* 1. ImageFileSelector

U ovom projektu korištena je jedna vanjska biblioteka ImageFileSelector. To je jednostavna za koristiti biblioteka za odabir i obrezivanje slika. ImageFileSelector korišten je kod odabira profilne slike tako da korisnik može odabrati željenu sliku te nakon odabira obrezati ju da bude u dobrom formatu i rezoluciji za prikaz u aplikaciji.

Više o samoj biblioteci ImageFileSelector može se vidjeti na stranicama:

The Android Arsenal - <http://android-arsenal.com/details/1/2775>

GitHub - <https://github.com/sw926/ImageFileSelector>

* 1. javamail-android

Ukoliko bi korisnik zaboravio svoju lozinku može ju obnoviti tako da mu se pošalje mail sa novom lozinkom. To se riješilo na strani Android-a preko library-a javamail-android. javamail-android sastoji se od tri .jar datoteke: activation.jar, additional.jar i mail.jar. Zajedno, služe za slanje, primanje i obradu/kreiranje elektroničke pošte. U ovom projektu koristio se za slanje kreiranje maila prilikom resetiranja lozinke korisnika.

Više o samom javamail-android može se vidjeti na stranici:

javamail-android - <https://code.google.com/archive/p/javamail-android/wikis>

* 1. Socket.io Client

Kompletan Socket.IO klijent za Java-u. Kompatibilan je sa Socket.IO v1.0 i kasnije. U

projektu služi za komuniciranje sa našim websocket serverom.

Više informacija na linku - <https://github.com/socketio/socket.io-client-java>

1. Web servisi

Web servisi služe kao posrednici između baze podataka i android aplikacije. Web servisi su realiziranipomoću 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. addParticipantsToConversation.js

Ovaj web servis služi za dodavanje dodatnih korisnika u razgovor. Kao parametar prima ID razgovora i listu e-mail adresa koji, u slučaju ove aplikacije, su također i ID-evi tih korisnika u bazi podataka. Web servis će na temelju ID-a razgovora dohvatiti odgovarajući razgovor i u listu „*participants“* dodati e-mail adrese.

* 1. get\_userData.js

Ovaj web servis služi za dohvat podataka o jednom korisniku. Kao parametar prima e-mail korisnika koji, u slučaju ove aplikacije, predstavlja ID tog korisnika. Web servis će iz baze podataka dovatiti sve korisnike i izvući jednog korisnika na temelju e-mail adrese.

* 1. getFriends.js

Ovaj web servis koristi se za dohvat informacija korisnikovih prijatelja. Dohvaća sve registrirane korisnike te ih filtrira i zadržava samo one koji su prijatelji korisnika. Sprema prijateljev ID, ime i trenutni status te šalje natrag klijentu odnosno korisniku.

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**

*User* klasa se prvenstveno koristi za prikaz korisnika u aplikaciji, ali i koristi za slanje parametara na web servis. Uočava se da postoji jedna samostalna klasa, a to je *ChatUpPreferences* koja služi za spremanje i dohvaćanje *SharedPreferences*-a. 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.

* 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. U slici 5.4. te klase su, prema njihovoj funkciji, svrstane u pakete: *ConversationAsync, LoginAsync, MainAsync* i *MenuAsync*. Sve klase iz navedenih paketa će biti prikazane u daljnjim slikama. **Bitno je napomenuti da svaka klasa iz navedenih paketa ima iste veze kao što su prikazane s paketima na slici 5.4.** Glavna zadaća im je dobaviti i zatražiti podatke ovisno o zadaći koju oni rade. Svaki naziv klase ujedno i objašnjava njezinu funkciju.

*IListener<T>* je tipizirano sučelje čije funkcije koriste kao jedna vrsta *callback* funkcija, odnosno omogućava programeru da definira ponašanje tih metoda izvan granica same klase. U spomenutom sučelju najvažnija je metoda *onFinish()* jer nam ona omogućava pristup rezultatu koje je vratilo neki web servis.

Klasa *HttpPOST* služi kao omotač (engl. *wrapper*) nizu poziva metoda koji omogućavaju uspostavljanje komunikacije s web servisima i slanje parametara POST metodom. Parametri se uglavnom prosljeđuju kao instance *Map* klase, no zbog ograničenja te klase (jedan ključ se veže za jednu vrijednost) potrebno je bilo definirati klasu *Pair<T1,T2>* da se mogu prosljeđivati više vrijednosti pod „istim ključem“ (kada se pošalju više vrijednosti pod istim ključem onda HTTP interpretira to kao polje).

Klasa *WebServiceStrings* se sastoji samo od konstantnih *string*-ova. Sadrži *string* za IP adresu servera te velik broj *string*-ova koji predstavljaju rute prema web servisima.



**Slika 5.5. Klase u paketu *MainAsync***



**Slika 5.6. Klase u paketu *MenuAsync***

****

**Slika 5.7. Klase u paketu *LoginAsync***



**5.8. Klase u paketu *ConversationAsync***

Slike 5.5., 5.6., 5.7. i 5.8. prikazuju sve klase koje se nalaze u paketima *MainAsync, MenuAsync, LoginAsync* i *ConversationAsync*. Bitno je ponoviti da svaka ta klasa ima veze iste kao i paketi 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.

1. Testovi
   1. Testovi prihvatljivosti

Napravljen je niz testova prihvatljivosti kako bi se utvrdilo koje funkcionalnosti u aplikaciji su dobro realizirane, a koje nisu, te kako bi se pronašli i ispravili mogući bug-ovi i nelogičnosti u aplikaciji.

* + 1. Test prihvatljivosti za kreiranje razgovora

U nastavku je prikazan test prihvatljivosti za kreiranje razgovora.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test prihvatljivosti za kreiranje razgovora**  **Napisan:** 2.1.2015.  **Pokrenuti:** 4.1.2015.  **Prošao/Pao: Prošao**  **Izveo ga:** Danijel Sokač  **Kreiranje razgovora**  Korisnik mora imati mogućnost pregleda svih prijatelja koji su mu u listi prijatelja. Također sa svakim prijateljem mora imati mogućnost kreiranja razgovora ili odustajanja od kreiranja na način da jednu od tih opcija odabere u meniju koji se pojavi nakon dugog klika na jednog od prijatelja iz liste.  **Kontekst**  Aplikacija pokrenuta na uređaju, prikazan Home Page tab, u kojem postoji lista prijatelja.  **Korisnička akcija**   1. Korisnik drži dugi klik na određenog prijatelja iz liste   **Očekivani izlaz**  Otvara se meni sa nazivom “Friend menu” u kojem može odabrati “Start Conversation” ili “Cancel”  **Korisnička akcija**  2.1. Korisnik klikne na “Start conversation” ili  2.2. Korisnik klikne na “Cancel”  **Očekivani izlaz**  Korisnička akcija 2.1   |  |  |  | | --- | --- | --- | | Slučaj 1 | Razgovor sa tim korisnikom već postoji | Ispisuje se poruka: “Conversation creation suspend. Conversation alredi exists” | | Slučaj 2: | Razgovor s tim korisnikom još ne postoji | Ispisuje se poruka: “Conversation created”. U tabu messages se kreira novi razgovor u listi Single Conversations |   Korisnička akcija 2.2 Zatvara se otvoreni kontekstualni meni **Bilješke:**   |  | | --- | | Test je uspješno prošao. U oba slučaja dogodio se izlaz jednak očekivanom. Testiranje je obavljeno na uzorku od 10 prijatelja iz liste i u svim slučajevima izlaz je bio jednak očekivanom. | |

* + 1. Test prihvatljivosti za uređivanje profila

U nastavku je prikazan test prihvatljivosti uređivanje profila korisnika.

|  |  |
| --- | --- |
| **Test prihvatljivosti za uređivanje profila**  **Napisan:** 3.1.2015.  **Pokrenuti:** 6.1.2015.  **Prošao/Pao:** Prošao  **Izveo ga:** Goran Drmenčić  **Uređivanje profila**  Korisnik mora imati mogućnost uređivanja vlastitog profila. To podrazumijeva promjenu korisničkog imena, spola i lozinke.  **Kontekst**  Aplikacija pokrenuta na uređaju, odabrana stavka “Edit profile” iz izbornika. Kao hintovi su postavljeni trenutni podaci o korisničkom imenu, spolu i lozinci.  **Korisnička akcija**   1. Korisnik unese korisničko ime i klikne na “Save changes”   **Očekivani izlaz**  Ispisuje se poruka “Update profile successfully”. Hint u korisničkom imenu ažurira se na način da umjesto starog korisničkog imena piše novo uneseno.  **Korisnička akcija**   1. Korisnik unese korisničko ime, promijeni vrijednost spola i klikne na “Save changes”   **Očekivani izlaz**  Ispisuje se poruka “Update profile successfully”. Ažurira se korisničko ime i spol.  **Korisnička akcija**   1. Korisnik unese vrijednost u polje s hintom trenutnog passworda i drugačiju vrijednost u polje s hintom “Confirm your password”.   **Očekivani izlaz**  Ispisuje se poruka “Password don’t match”. Ništa se ne ažurira.  **Korisnička akcija**   1. Korisnik unese iste vrijednosti u polje s hintom trenutnog passworda i polje s hintom “Confirm new password” i promijeni vrijednost spola u radiobutton-u.   **Očekivani izlaz**  Ispisuje se poruka “Update profile successfully”. Ažurira se lozinka i spol.  **Bilješke:**   |  | | --- | | Test je uspješno prošao. U sve 4 korisničke akcije dogodio se izlaz jednak očekivanom. Testiranje je obavljeno sa 8 ponavljanja za svaku od korisničkih akcija sa 4 različita logirana korisnika i u svim slučajevima izlaz je bio jednak očekivanom. | |

* + 1. Test prihvatljivosti za slanje poruke

U nastavku slijedi test prihvatljivosti za slanje poruke.

|  |  |
| --- | --- |
| **Test prihvatljivosti za slanje poruke**  **Napisan:** 6.1.2015.  **Pokrenuti:** 8.1.2015.  **Prošao/Pao: Prošao**  **Izveo ga:** Matija Jurman  **Slanje poruke**  Korisnik mora imati mogućnost slanja poruke korisniku kojeg je odabrao iz liste kreiranih razgovora u tab-u “Message”  **Kontekst**  Aplikacija pokrenuta na uređaju, otvoren je jedan od razgovora iz liste “Single conversations” koja se nalazi u tab-u “Message”.  **Korisnička akcija**   1. Korisnik klikne na područje za pisanje tekst-a na dnu ekrana.   **Očekivani izlaz**  Otvara se tipkovnica i pomiče se prozor aktivnosti prema gore, kako tipkovnica ne bi prekrila mjesto za unos teksta.  **Korisnička akcija**  2. Korisnik upiše tekst u područje za pisanje teksta i pritisne tipku “Send”    **Očekivani izlaz**  Tekst je vidljiv na zaslonu kao poslana poruka zajedno sa imenom pišiljatelja i datumom slanja poruke. Upisani tekst vidljiv je i na uređaju korisnika s kojim se vodio razgovor.  **Korisnička akcija**  3. Pritisne se tipka za zatvaranje tipkovnice    **Očekivani izlaz**  Prozor aktivnosti pomiče se natrag prema dolje kao što je bio prije klika na područje za pisanje teksta.  **Korisnička akcija**  4. Pritisne se tipka “undo” na mobilnom uređaju.  **Očekivani izlaz**  Aktivnost razgovora se zatvara. Otvoreni je tab “Message” gdje je vidljiva lista svih kreiranih razgovora za prijavljenog korisnika.  **Bilješke:**   |  | | --- | | Test je uspješno prošao. U sva četiri slučaja dogodio se izlaz jednak očekivanom. Testiranje je obavljeno na uzorku od 7 razgovora, i svi su uspješno prošli sa jednakim očekivanim izlazom. | |

* + 1. Test prihvatljivosti za funkcionalnost reklama

U nastavku slijedi test prihvatljivosti za funkcionalnost reklama.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test prihvatljivosti za funkcionalnost reklama**  **Napisan:** 6.1.2015.  **Pokrenuti:** 10.1.2015.  **Prošao/Pao: Prošao**  **Izveo ga:** Danijel Filipović  **Funkcionalnost reklama**  U aplikaciji mora biti realizirana mogućnost otvaranja reklame prilikom otvaranja razgovora sa nekim od korisnika.  **Kontekst**  Aplikacija pokrenuta na uređaju, korisnik pozicioniran u tabu “Message”. Kliknuto zaglavlje “Single conversations” tako da je prikazana lista svih “Single” razgovora za prijavljenog korisnika.  **Korisnička akcija**   1. Korisnik klikne na jedan od kreiranih razgovora iz liste.   **Očekivani izlaz**  Otvara se reklama preko punog zaslona.  **Korisnička akcija**     |  |  | | --- | --- | | Akcija 1 | Korisnik klikne na sadržaj reklame | | Akcija 2 | Korisnik klikne na “X” u gornjem lijevom kutu reklame | | Akcija 3 | Korisnik klikne tipku “Back” na mobilnom uređaju. |     **Očekivani izlaz**     |  |  | | --- | --- | | Očekivani izlaz za akciju 1 | Otvara se oglas sa reklamom | | Očekivani izlaz za akciju 2 | Zatvara se reklama i prikazuje aktivnost razgovora sa odabranim korisnikom | | Očekivani izlaz za akciju 3 | Zatvara se reklama i prikazuje aktivnost razgovora sa odabranim korisnikom |   **Bilješke:**   |  | | --- | | Test je uspješno prošao. U oba slučaja dogodio se izlaz jednak očekivanom. Testiranje je obavljeno na uzorku od 6 razgovora, i svi su uspješno prošli sa jednakim očekivanim izlazom. | |

* 1. Robotium testovi

U nastavku slijedi prikaz odrađenih Robotium testova u aplikaciji. Testirane su neke osnovne funkcionalnost iz glavnog izbornika aplikacije kao i funkcionalnost iz taba HomePage.

* + 1. EditProfileRobotium

Navedenim testom testirana je funkcionalnost „Edit profile“ iz glavnog izbornika aplikacije. Napisana je automatizacijska skripta korištenjem Robotium framework-a kojim je testiran sljedeći niz koraka:

1. Logiranje korisnika
2. Otvaranje Edit profile stavke glavnog izbornika
3. Promjena korisničkog imena
4. Spremanje postavki
5. Promjena vrijednosti spola
6. Spremanje postavki

Test je uspješno prošao i aplikacija se ponašala u skladu s očekivanjima.

* + 1. SettingsRobotium

Navedenim testom testirana je funkcionalnost „Settings“ iz glavnog izbornika aplikacije. Napisana je automatizacijska skripta korištenjem Robotium framework-a kojim je testiran sljedeći niz koraka:

1. Logiranje korisnika
2. Otvaranje „Settings“ stavke glavnog izbornika
3. Odabir prve opcije i spremanje promjena
4. Odabir druge opcije i spremanje promjena
5. Odabir treće opcije i spremanje promjena

Test je uspješno prošao i aplikacija se ponašala u skladu s očekivanjima.

* + 1. HomePageFragmentRobotium

Navedenim testom testirana je funkcionalnost započinjanja razgovora s jednim od korisnika iz liste prijatelja u „Home page“ tab-u. Napisana je automatizacijska skripta korištenjem Robotium framework-a kojim je testiran sljedeći niz koraka:

1. Logiranje korisnika
2. Dugi klik na jednog od korisnika
3. Dugi klik na drugog korisnika
4. Odabir opcije „Start conversation“ iz kontekstnog izbornika
5. Odabir opcije „Cancel“ iz kontekstng izbornika

Test je uspješno prošao i aplikacija se ponašala u skladu s očekivanjima.

* + 1. LogoutRobotium

Navedenim testom testirana je funkcionalnost „Logout“ iz glavnog izbornika aplikacije. Napisana je automatizacijska skripta korištenjem Robotium framework-a kojim je testiran sljedeći niz koraka:

1. Logiranje korisnika
2. Otvaranje Logout stavke glavnog izbornika
3. Ponovno logiranje korisnika

Test je uspješno prošao i aplikacija se ponašala u skladu s očekivanjima.

* + 1. AboutUsRobotium

Navedenim testom testirana je funkcionalnost „About us“ iz glavnog izbornika aplikacije. Napisana je automatizacijska skripta korištenjem Robotium framework-a kojim je testiran sljedeći niz koraka:

1. Logiranje korisnika
2. Otvaranje „About us“ stavke glavnog izbornika
3. Zatvaranje otvorenog dijaloškog okvira

Test je uspješno prošao i aplikacija se ponašala u skladu s očekivanjima.

* + 1. SearchFragmentRobotium

Navedenim testom testirana je funkcionalnost pretraživanja korisnika. Napisana je automatizacijska skripta korištenjem Robotium framework-a kojim je testiran sljedeći niz koraka:

1. Logiranje korisnika
2. Pozicioniranje na tab „Search“
3. Upis kriterija pretraživanja
4. Klik na gumb za pretraživanje
5. Dugi klik na jednog od pronađenih korisnika i odabir stavke „Cancel“

Test je uspješno prošao i aplikacija se ponašala u skladu s očekivanjima.

1. Modularnost u aplikaciji
   1. Modularnost sa *FragmentBuffer*

Jedna od modularnosti realizirana je pomoću klase *FragmentBuffer*. Navedena klasa u sebi sprema objekte klase *Fragment*, odnosno objekte čije klase nasljeđuju klasu *Fragment*. Može se reći da je modularnost realizirana na temelju klase *Fragment.*

Klasa *FragmentBuffer* se u ovoj aplikaciji koristi za pohranu fragmenata koji će se poslije učitati u kartice (engl. *tabs*). U klasi su spremljeni sljedeći fragmenti: *HomePageFragment*, *SearchFragment* i *Message.*

*FragmentBuffer* je definiran da za svaki unos novog fragmenta može se tom fragmentu dodijeliti neka oznaka (engl *tag*). U ovom slučaju svakom fragmentu je dodijeljen naziv kartice kao oznaka. Nazivi kartica su po redu: *Home Page*, *Search* i *Message*.

Na temelju fragmenata koji su pohranjeni u instanci *FragmentBuffer* klase dinamički se kreiraju kartice. Ako programer želi dodati neku novu karrticu jednostavno mora kreirat novu klasu koja nasljeđuje *Fragment* i u kodu (*MainClass.java* > *createTabs()*) dodati naredbu:

fragmentBuffer.add(new NoviFragment(), “oznaka“);

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