SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

**Palaić Leon**

**Turek Tomislav**

**Vračan Maja**

Aplikacija Team UP

TEHNIČKA dokumentacija iz kolegija analiza i razvoj programa

**Varaždin, 2015.**

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Oznaka tima: T19

Članovi tima:

Palaić Leon

Turek Tomislav

Vračan Maja

GitHub repozitorij: <https://github.com/mvracan/team-up>

Aplikacija team up

TEHNIČKA dokumentacija iz kolegija analiza i razvoj programa

**Mentor:**

Doc.dr.sc. Zlatko Stapić

**Varaždin, 2015.**

SADRŽAJ

SADRŽAJ i

1. Team Up – specifikacija zahtjeva 1

1.1 Korisnički zahtjevi 1

1.2 Opis funkcionalnosti 2

1.2.1. Prijava u sustav 2

1.2.2. Registracija 2

1.2.3. Izmjena profila 3

1.2.4. Kreiranje grupa 3

1.2.5. Dodavanje/prijava u grupu 4

1.2.6. Pregled lokacija ljudi iz grupe 4

1.2.7. Obavijesti vođi grupe 5

1.2.8. Panic button 5

1.2.9. Poništavanje grupe 6

2. Team Up – specifikacija arhitekture 6

3.1 Arhitektura aplikacije 6

Baza podataka 7

Web servis 8

Mobilna aplikacija 11

Komunikacija sa servisom 11

Specifikacija modula 12

# Team Up – specifikacija zahtjeva

## Korisnički zahtjevi

U tablici 1 prikazani su zahtjevi definirani od strane korisnika zajedno sa njihovim prioritetima i vrijednosti koje one predstavljaju korisniku.

Prioriteti su izraženi tako da najvažniji zahtjev ima iznos jedan, a ostalim zahtjevima sa smanjenjem važnosti im se povećava iznos prioriteta. S druge strane, vrijednosti zahtjeva izražene su tako da veća vrijednost ima veći iznos u tablici, gdje iznosi mogu biti u rasponu od jedan do devet.

|  |  |  |
| --- | --- | --- |
| **Zahtjev** | **Vrijednost** | **Prioritet** |
| Izrada profila | 8 | 1 |
| Izmjena profila | 2 | 2 |
| Kreiranje grupe | 9 | 3 |
| Brisanje grupe | 3 | 4 |
| Dodavanje ljudi u grupu preko korisničkog imena | 4 | 5 |
| Pregled lokacija ljudi iz grupe | 7 | 6 |
| Obavijest vođi o udaljavanju korisnika | 6 | 7 |
| Slanje obavijesti vođi od strane korisnika da se izgubio | 5 | 8 |
| Dodavanje u grupu samostalno unosom koda | 7 | 9 |
| Dodavanje u grupu uz pomoć NFC-a | 4 | 10 |
| Brojač ljudi | 1 | 11 |

Tablica 1: Popis funkcionalnosti

Prema opisu aplikacije i korisničkih zahtjeva mogu se uočiti njene osnovne funkcionalnosti:

* Prijava u sustav
* Registracija
* Izmjena profila
* Kreiranje grupa
* Dodavanje ljudi u grupe/prijava u grupu
* Pregled lokacija ljudi iz grupe
* Obavijest vođi grupe ukoliko se pojedinac odvoji od dozvoljenog radijusa grupe
* Panic button – pojedinac šalje informaciju da se izgubio
* Poništavanje grupe
* Brojač ljudi

## Opis funkcionalnosti

### Prijava u sustav

#### Opis

Svaki korisnik dužan je prijaviti se u aplikaciju sa registriranim korisničkim imenom i lozinkom kako bi ga se moglo jedinstveno prepoznati prilikom grupiranja, ali i locirati ukoliko nešto pođe po zlu. Kada se korisnik prijavi u sustav, svi njegovi podaci zajedno sa podacima pridruženih grupa, preuzeti će se i prikazati u aplikaciji.

#### Slijed zahtjeva

Konkretno, korisnik unosi korisničko ime i lozinku, a servis odgovara podacima osobe registriranom pod tim korisničkim imenom (ime i prezime) te listom grupa u kojima je navedena osoba (podaci grupe su naziv i opis).

#### Funkcijski zahtjevi

Prijava u aplikaciju dostupna je svim korisnicima i nužna je za njeno daljnje korištenje. Pri unosu potrebno je izvršiti provjeru korisničkog unosa gdje je za korisničko ime i lozinku potrebno unesti minimalno 5 znakova. Nakon uspješnog unosa, korisničko ime i lozinka poslati će se na web servis u obliku objekta, a pri zaprimanju odgovora, propustiti korisnika dalje u aplikaciju ovisno o konkretnom HTTP odgovoru.

### Registracija

#### Opis

Kako bi korisnik mogao koristiti aplikaciju, potrebno se registrirati u sustavu. Registracija će biti omogućena direktno putem mobilne aplikacije gdje će se uzeti samo najbitniji podaci za korištenje aplikacije.

#### Slijed zahtjeva

Prilikom registracije, korisnik unosi ime, prezime te željeno korisničko ime i lozinku. Navedeni podaci šalju se na web servis gdje web servis vraća status registracije navedenog korisnika (uspješan ili neuspješan).

#### Funkcijski zahtjevi

Prije slanja podataka, potrebno je izvršiti provjeru korisničkog unosa. Korisnik unosi vlastito ime, prezime te korisničko ime i lozinku. Konkretno, za polja ime, prezime te korisničko ime potrebno je provjeriti duljinu koja mora biti barem duljine pet znakova, a maksimalno četrdeset i pet te dozvoliti samo unos malih i velikih slova. Što se tiče unosa lozinke, lozinka može sadržavati sve znakove no mora biti dulja od pet a manja od četrdeset i pet znakova. Kako bi bili sigurni da je korisnik točno unio vlastitu lozinku, morati će istu lozinku unijeti dva put kako bi ju potvrdio. Nakon što je izvršena registracija korisnika, korisnik se prijavljuje odmah u aplikaciju prema specifikaciji navedenoj u prijavi.

### Izmjena profila

#### Opis

Svakom korisniku dodana je mogućnost izmjene vlastitog profila po njegovoj želji. Izmjena profila obuhvaća izmjenu podataka za navedenog korisnika što obuhvaća korisničko ime, lozinku te njegovo ime i prezime.

#### Slijed zahtjeva

Prilikom izmjene profila, korisnik sa mobilnog uređaja šalje nove podatke o korisniku, a web servis vraća HTTP odgovor o statusu izvršenja izmjene podataka (uspješno ili neuspješno).

#### Funkcijski zahtjevi

Izmjenu profila moguće je vršiti u bilo kojem trenutku, no potrebno je paziti postoji li već isto korisničko ime u bazi prilikom izmjene (korisničko ime mora biti jedinstveno). Također, potrebno je izvršiti provjeru korisničkog unosa koji je definiran pri registraciji korisnika.

### Kreiranje grupa

#### Opis

Osnovna funkcionalnost ove aplikacije jest kreiranje grupa. Bilo koji korisnik može kreirati grupu putem forme u koju unosi ime grupe. Ti podaci šalju se na web servis koji kreira novu grupu te za zadanu grupu kreira jedinstveni kod za tu grupu putem koje se ostali korisnici mogu pridružiti. Nakon slanja podataka o grupi na web servis. Web servis odgovara statusom kreirane grupe te u slučaju uspješno kreirane grupe vraća podatke koji su generirani prilikom kreiranja (primjerice jedinstveni kod).

#### Slijed zahtjeva

Korisnik unosi ime grupe, a nakon toga šalje zahtjev prema web servisu. Web servis kreira ostale podatke na temelju imena i pošiljatelja zahtjeva. Odgovor od strane web servisa sadržavati će sve podatke koji su generirane za tu grupu gdje će biti naveden i konkretan kod za pristupanje grupi.

#### Funkcijski zahtjevi

Kod kreiranja grupe potrebno je kreirati objekt u koji će postojati samo naziv grupe, a nakon toga će web servis popuniti ostale podatke vezano uz tu grupu. Po primitku svih podataka za tu grupu, omogućiti će se dodavanje novih ljudi u tu grupu. Također, potrebno je pripaziti da generirani kod za grupu bude jedinstven kako ne bi došlo do kolizija prilikom pridruživanja grupi.

### Dodavanje/prijava u grupu

#### Opis

Korisnici mogu ući u grupu putem jedinstvenog koda koji je generiran za svaku novu grupu. Korisnik ima dva načina kako bi se pridružio grupi: utipkavanjem koda u predviđeni tekstualni okvir ili prijenosom putem NFC čipa na pozadini uređaja (NFC beam).

#### Slijed zahtjeva

Implementacija unosa koda jednaka je za oba načina, razlikuje se samo način unosa. Unošenjem koda šalje se poziv na web servis koji prima zahtjev za pridruživanje, obrađuje ga te vraća informacije o grupi u koju se korisnik prijavio. Također, u aplikaciji ostaviti će se mogućnost kreatoru grupe da doda korisnike unosom njihovih korisničkih imena.

#### Funkcijski zahtjevi

Prilikom pridruživanja grupi potrebno je provjeriti da li navedena grupa uopće postoji. Kada smo utvrdili da grupa postoji, dovoljno je pridružiti osobu sa tom grupom te prikazati informacije o ostalim osobama koji su u navedenoj grupi.

### Pregled lokacija ljudi iz grupe

#### Opis

Radi se o jednostavnoj funkcionalnosti gdje će korisnici putem mape moći pratiti lokaciju drugih korisnika unutar grupe. Prilikom slanja zahtjeva za pregled lokacija, web servis vraća pozicije ostalih korisnika koji se pritom prikazuju na karti.

#### Slijed zahtjeva

Korisnik odabirom na kartu automatski šalje zahtjev web servisu koji čita njihove lokacije. Lokacije svih osoba iz te grupe šalju se mobilnoj aplikaciji koja iscrtava lokacije na kartu gdje će svaki marker jedinstveno identificirati o kojoj se osobi točno radi.

#### Funkcijski zahtjevi

U trenutku slanja zahtjeva prema web servisu, web servis će povući podatke o lokaciji svih korisnika unutar grupe te poslati ih korisniku u JSON obliku. Navedene lokacije koristiti će se za iscrtavanje markera na mapu gdje će svaki marker biti označen drugačijom bojom, a uz svaku boju biti će navedeno i ime korisnika kako bi ga se moglo identificirati na karti.

### Obavijesti vođi grupe

#### Opis

Ukoliko dođe do udaljavanja pojedinca van dozvoljenog radijusa grupe, kreator grupe dobiti će obavijest da je došlo do udaljavanja pojedinca, zajedno s njegovim informacijama i lokacijom. Ovo se može shvatiti kao svojevrsni alarm kreatoru da postoji velika mogućnost gubitka pojedinca iz grupe.

#### Slijed zahtjeva

Implementacijski, svaki korisnik odašiljati će informaciju o svojoj lokaciji koja će se uspoređivati sa dozvoljenim radijusom definiranim od strane kreatora. Ukoliko prilikom dobivanja informacija o lokaciji web servis ustanovi da je netko izašao iz radijusa, slati će obavijest korisniku koji je definiran kao kreator te grupe.

#### Funkcijski zahtjevi

Kako bi se pravovremeno poslala obavijest vođi grupe, potrebno je konstantno nadzirati lokacije ostalih korisnika. Prilikom utvrđivanja lokacije potrebno je računati udaljenost od vođe grupe te ukoliko izračunata udaljenost bude veća od definiranog radijusa, potrebno je poslati alarm vođi na mobilni uređaj. Trenutno alarm ne bi smio prestati dok se korisnik ne vrati unutar radijusa, no razmatra se mogućnost da se omogući vođi da privremeno ugasi alarm.

### Panic button

#### Opis

Također je potrebno uzeti u obzir da postoje i drugi faktori gubitka pojedinca iz grupe osim udaljenosti. Primjerice, u velikim gužvama, udaljenost ne mora biti glavni faktor za gubitak pojedinca već je dovoljno izgubiti pogled na grupu ili doticaj s grupom. Budući da se takve situacije mogu dešavati u stvarnosti, dodana je funkcionalnost gumba panike koji će na pritisak odmah alarmirati kreatora grupe da sa jednim od pojedinaca nešto nije u redu te je potrebno poduzeti određene mjere.

#### Slijed zahtjeva

Implementacijski, gumb će slati zahtjev na web servis, a web servis će kontaktirati kreatora i poslati mu obavijest sa podacima te osobe koji će se prikazati na ekranu.

#### Funkcijski zahtjevi

Kada korisnik pritisne na panic gumb, slati će se jednostavan POST zahtjev bez parametara koji će inicirati slanje podataka o toj osobi do vođe te grupe. Kada aplikacija primi podatke, podaci će se ispisati na ekran, zajedno sa porukom da se korisnik izgubio.

### Poništavanje grupe

#### Opis

Prije nego se grupa raziđe, preporuča se njeno poništavanje kako bi kreator prestao dobivati obavijesti.

#### Slijed zahtjeva

Kreator će prije napuštanja grupe poslati zahtjev poništenja na web servis koji će grupu raspustiti i poslati obavijest o uspjehu raspuštanja grupe.

#### Funkcijski zahtjevi

Kada kreator odluči raspustiti grupu, potrebno ga je zatražiti potvrdu da stvarno želi inicirati raspuštanje. Ukoliko potvrdi raspuštanje, slati će se HTTP zahtjev na web servis bez parametara gdje će web servis obrisati zapise vezano uz tu grupu te poslati HTTP odgovor (uspješno ili neuspješno raspuštanje).

# . Korisničke priče

# 1. Kao korisnik aplikacije želim moći kreirati tim kako bih omogućio sigurnost članovima svog tima.

# Kriterij prihvatljivosti:

# Pod pretpostavkom da je korisnik autentificiran u trenutku kreiranja grupe te da nije učlanjen u niti jednu drugu grupu omogući kreiranje nove grupe I preusmjeri ga na zaslon o timu

# 2. Kao korisnik aplikacije želim moći ući u tim putem timu dodijeljenog koda.

# Kriterij prihvatljivosti:

# Pod pretpostavkom da je korisnik autentificiran I da mu je poznat kod tima u trenutku učlanjenja omogući učlanjenje u novu grupu te ga preusmjeri na zaslon o ranije navedenom timu.

# 3. Kao korisnik aplikacije želim moći ući u tim putem NFC komunikacije sa dijeljenjem podataka o timu s drugim članom tima.

# Kriterij prihvatljivosti :

# Pod pretpostavkom da su oba korisnika autentificirana , da oba imaju omogućenu infrastrukturu za NFC komunikaciju te da je jedan član komunikacije unutar tima, omogući učlanjenje u tim te prikaži zaslon tima.

# 4. Kao vođa(kreator) tima želim obavijest ukoliko član tima izađe iz dozvoljenog radiusa kako bih mogao pravovremeno reagirati na zadanu situaciju.

# Kriterij prihvatljvosti:

# Pod predpostavkom da je vođa autentificiran I da korisnik izađe iz predefiniranog radiusa grupe pošalji obavijest vođi kako se određeni član tima izgubio.

# 5. Kao vođa tima želim vidjeti na karti lokacije članova tima izvan radiusa kako bih mogao pravovremeno reagirati I vratiti ih ostatku grupe.

# Kriterij prihvatljivosti :

# Pod pretpostavkom da je vođa autentificiran te da je član tima izvan radiusa u trenutku zahtjeva za lokacijom omogući vođi otvaranje mape I prikaz lokacije člana tima.

# 6. Kao vođa tima želim mogućnost smirivanja člana tima kako bih potvrdio da sam članu tima osigurao sigurnost.

# Kriterij prihvatljivosti:

# Pod pretpostavkom da je vođa tima autentificiran te da je član tima u danom trenutku u panici omogući prozor za smirivanje člana tima

# 7. Kao član tima želim poslati obavijest vođi tima kako sam se izgubio.

# Kriterij prihvatljivosti:

# Pod pretpostavkom da je član tima autentificiran I da se nalazi unutar aktivnog tima omogući slanje obavijesti vođi grupe.

# 

# 3. Team Up – specifikacija arhitekture

## Arhitektura aplikacije

TeamUp aplikacija zamišljena je kao mobilna aplikacija koja koristi web servis kako bi dohvatila, izmjenila ili dodala podatke u bazu podataka. Prema tim zahtjevima gradi se arhitektura koja sadrži jedan web poslužitelj koji ima bazu podataka sa podacima važnim za funkcioniranje aplikacije te web servis koji vrši obradu podataka unutar te baze podataka. Na centralni web poslužitelj spajaju se klijenti putem mobilne aplikacije te određenim radnjama unutar mobilne aplikacije vrše pozive prema web servisu, a web servis vrši promjenu podataka unutar baze.

### Baza podataka

### D:\Dokumenti\AndroidStudioProjects\team-up\dokumentacija\db\ERAmodelTeamUp.png

Slika 1: ERA model

Kako bi implementirali navedene funkcionalnosti potrebana nam je baza podataka koja će implementirati naš podatkovni model. Za implementaciju podatkovnog modela koristit ćemo MySQL bazu podataka.

Naš podatkovni model sastoji se od tri tablice : user, team i teammember. Tablice user i team u našem modelu su jaki entiteti koji služe za pohranjivanje korisnika i timova. Svaki tim ima svog kreatora koji je ujedno ii voditelj tima te je povezana sa tablicom user putem vanjskog ključa. U ovom slučaju se radi o relaciji jedan naprama više. Jedan korisnik može biti kreator nula ili više timova dok tim ima samo jednog kreatora. Također svaki korisnik može biti član više timova te tim ima jednog ili više članova. Realcija više naprama više je implementirana slabim entitetom teammembers. Detalji tablice user prikazani su u tablici 1.

|  |  |
| --- | --- |
| Atribut | Opis |
| id\_user | Primarni ključ tablice user |
| name | Atribut za bilježenje kornisnikovog imena |
| surname | Atribut za bilježenje kornisnikovog prezimena |
| username | Atribut za bilježenje korinikovog korisničkog imena |
| password | Atribut za bilježenje korinikove lozinke |
| latitude | Atribut za bilježenje korisnikove lokacije |
| longitude | Atribut za bilježenje korisnikove lokacije |

Tablica 2: tablica user

Detalji tablice team prikazani su u tablici 2.

|  |  |
| --- | --- |
| Atribut | Opis |
| id\_team | Primarni ključ tablice |
| name | Atribut za bilježenje imena grupe |
| desc | Atribut za bilježenje opisa grupe |
| password | Atribut za bilježenje lozinke grupe, kako bi se korisnici sami mogli dodati u grupu |
| nfc\_code | Atribut za bilježenje nfc koda, kako bi se korisnici mogli dodavati u grupu putem nfc tehnologije |
| radius | Atribut za bilježenje radiusa grupe |
| id\_creator | Vanjski ključ na tablicu user |

Tablica 3: tablica team

Detalji tablice teammember prikazani su u tablici 3.

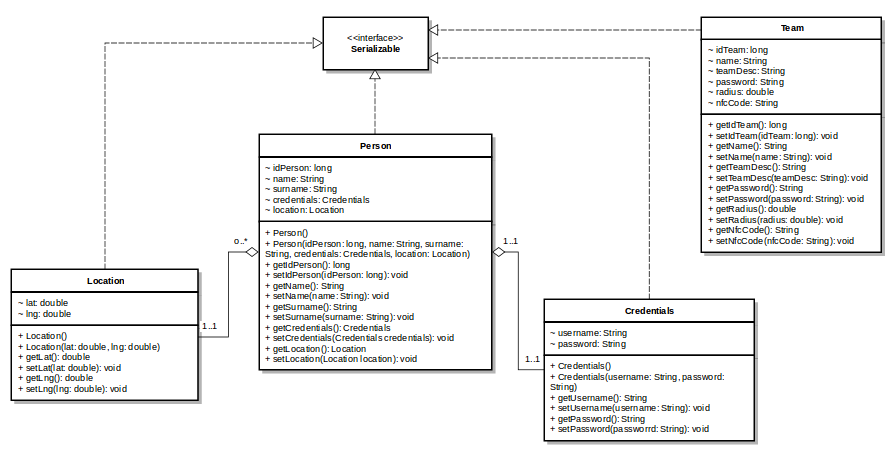
|  |  |
| --- | --- |
| Atribut | Opis |
| id\_user | Primarni ključ, vanjski ključ na tablicu user |
| id\_team | Primarni ključ,vanjski ključ na tablicu team |

Tablica 4: tablica teammember

### Web servis

Web servis leži na poslužitelju te se pokreće kao samostalna aplikacija. Prilikom pokretanja web servisa aplikacija podiže svoj interni (ugrađeni) web poslužitelj putem kojeg prima pozive od klijenta te vraća odgovore. Ugrađeni web poslužitelj pokreće se tako da radi paralelno sa web poslužiteljem na kojem je smješten osluškivajući port 8080.

Kako bismo mogli shvatiti logiku web servisa potrebno je prethodno pojasniti klase koje su napravljene pomoću kojih će klijent komunicirati sa web servisom. U nastavku se nalazi dijagram klasa koji je implementiran unutar web servisa i klijentske aplikacije.



Slika 2: Dijagram klasa modula model

Možemo vidjeti da model aplikacije prati tablice iz ERA dijagrama te atribute te tablice preslikava u članove klasa. Važno je napomenuti da web servis koristi Hibernate ORM koji koristi kako bi mapirao klase sa pripadajućim klasama i omogućio brži razvoj. Također, te iste klase predstavljaju temelj za slanje podataka na web servis (klasa i njeni podaci pretvaraju se u JSON format).

Cijela logika web servisa napisana je u RESTful paradigmi gdje se putem zadanog URL-a i HTTP metode (POST, GET, PUT, DELETE) vrše pozivi na web servis, što okida akcije za obradu podataka ovisno o URL-u, metodi te podacima koji su poslani. Svi podaci šalju se u tijelu zahtjeva (request body) u JSON formatu (application/json) što servis čita i koristi pri obradi. Primjerice, pri registraciji novih korisnika koristiti ćemo URL oblika [https://url\_web\_mjesta/person/signup](https://IP_ADRESA:8080/person/signup) te HTTP metodu POST gdje će u tijelu zahtjeva biti Person objekt popunjen podacima i iskazan u JSON formatu. Detaljnija specifikacija putanja web servisa opisana je u tablici u nastavku.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Br. | Operacija | URL | HTTP metoda | Tijelo zahtjeva |
| 1 | Ispis svih korisnika | /person/ | GET | - |
| 2 | Registracija korisnika | /person/signup | POST | Person objekt |
| 3 | Login korisnika | /person/login | POST | Credentials objekt |
| 4 | Ispis određenog korisnika prema id-u | /person/{id} | GET | - |
| 5 | Izmjena korisnika | /person/{id} | PUT | Person objekt |
| 6 | Brisanje korisnika | /person/{id} | DELETE | - |
| 7 | Ispis timova | /team/ | GET | - |
| 8 | Kreiranje novog tima | /team/create | POST | Team objekt |
| 9 | Brisanje tima | /team/{id} | DELETE | - |

Tablica 5: Popis putanja web servisa

Prilikom zaprimanja zahtjeva, web servis radi određene obrade te vraća zadane HTTP odgovore koji ovise o rezultate same obrade. Svi mogući odgovori od strane web servisa definirani su u tablici u nastavku prema putanji.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| URL | HTTP zahtjev | Mogući odgovor | | Komentar |
| HTTP kod | JSON odgovor |  |
| /person/ | GET | 200 OK | lista korisnika | vraća listu svih korisnika iz baze |
| /person/login | POST | 200 OK | podaci korisnika | uspješna prijava |
| 404 NOT FOUND | - | vraćeno ukoliko ne postoji navedeni korisnik ili lozinka nije valjana |
| /person/signup | POST | 200 OK | podaci korisnika | uspješna registracija |
| 400 BAD REQUEST | - | vraćeno ukoliko je došlo do pogreške prilikom spremanja (primjerice isto korisničko ime kao i neki drugi korisnik) |
| /person/{id} | GET | 200 OK | podaci korisnika | podaci za osobu sa id-em {id} |
| 404 NOT FOUND | - | ukoliko nije nađen korisnik sa id-em {id} |
| PUT | 200 OK | - | uspješna izmjena podataka za korisnika s id-em {id} |
| 404 NOT FOUND | - | ne postoji korisnik sa id-em {id} |
| /team/ | GET | 200 OK | lista grupa | vraća listu svih grupa iz baze |

Tablica 6: Popis mogućih odgovora web servisa

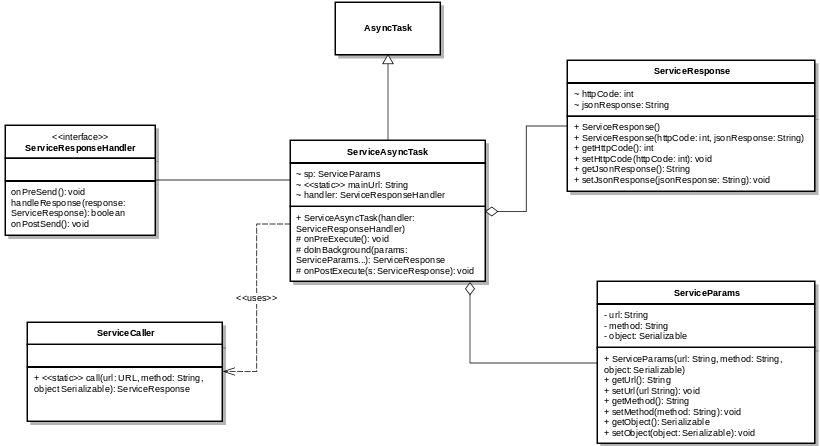
Naravno, važno je napomenuti da svaka putanja može vratiti HTTP 500 (INTERNAL SERVER ERROR) ukoliko postoji se dogodi unutarnja pogreška prilikom obrade, u tablici su navedeni samo odgovori nastali prilikom pogreške na klijentskoj strani ili uspješne obrade.

### Mobilna aplikacija

Cilj i naglasak u izradi ove aplikacije jest na modularnosti. Arhitektura mobilne aplikacije zamišljena je da bude sastavljena od "kockica" koje se međusobno spoje i čine cjelinu, ali isto vremeno ih je moguće nadograditi te ponovno iskoristiti da se svaka od tih "kockica" može primjeniti i u kasnijim projektima i programskim rješenjima. Kako bi se postigli navedeni ciljevi potrebno je pažljivo kreirati module tako da imaju minimalnu ovisnost o drugim modulima unutar aplikacije, ali i da su izrađene kao da će biti korištene i kasnije, a ne samo za ovaj projekt. Unutar aplikacije izrađeni su sljedeći moduli: app, access, core, model, nfc, prompts, webservice, stomp i websocket. Websocket je open source implementacija koja omogućava korištenje websocketa u Androidu.

### Komunikacija sa servisom

Klijent koristi mobilnu aplikaciju kako bi kontaktirao web servis i napravio izmjene unutar aplikacije. Kako bi se uspostavio kontakt sa serverom, korištena je standardna HttpURLConnection klasa. U nastavku možemo vidjeti dijagram klasa webservice modula.



Slika 3: Dijagram klasa modula webservice

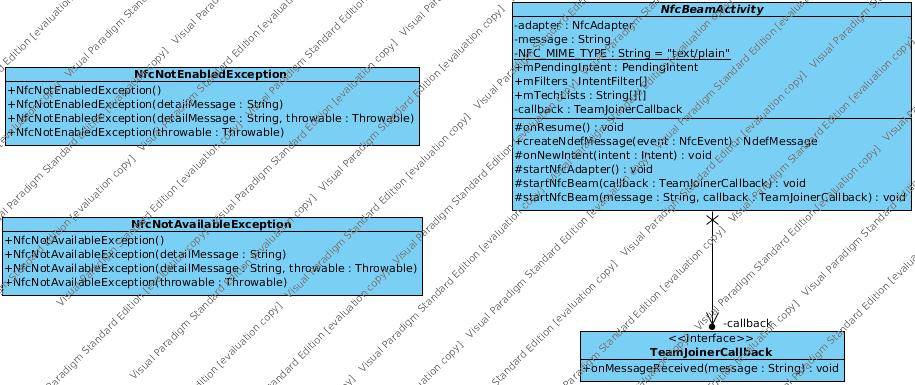
Unutar modula definirali smo sučelje koje je potrebno implementirati kako bismo mogli obraditi odgovor od strane našeg web servisa. U tom sučelju potrebno je implementirati što će se izvršiti prije slanja zahtjeva (onPreSend), konkretna obrada zahtjeva (handleResponse) te što će se izvršiti nakon slanja zahtjeva (onPostSend). Kako bismo izvršili poziv prema web servisu, implementirali smo metodu call u klasi ServiceCaller, koja se poziva iz ServiceAsyncTask klase koja objedinjuje pozive prema web servisu te pozive prema sučelju za obradu zahtjeva. Također, kako bismo mogli slati i primati podatke, definiramo klase ServiceResponse i ServiceParams. ServiceParams klasa sadrži putanju prema kojoj se šalju podaci, HTTP metodu koja će se koristiti pri slanju te konkretni podaci. Primjetimo da objekt mora implementirati Serializable sučelje kako bi mogao poslati podatke. ServiceResponse klasa predstavlja podatke zaprimljene od strane web servisa, a to je konkretno HTTP kod odgovora te konkretan JSON odgovor.

### Specifikacija modula

Moduli app, access i model svojstveni su za ovu aplikaciju te sadrže implementaciju koje su specifične za projekt i nemaju visok stupanj ponovne iskoristivosti. Modul app sadrži konkretne aktivnosti, fragmente, adaptere te implementaciju koja se izvršava nakon poziva prema serveru. Modul access je modul koji je naveden kao zahtjev kod izrade ovog projekta te mora implementirati ulaz u tim (\*), a modul model su obične domenske klase koje definiraju tipove unutar aplikacije.

S druge strane, imamo module koji su izrađeni kako bi se omogućila njihova ponovna iskoristivost, a to su konkretno modul webservice, nfc, prompts, stomp i core.

**Webservice modul**

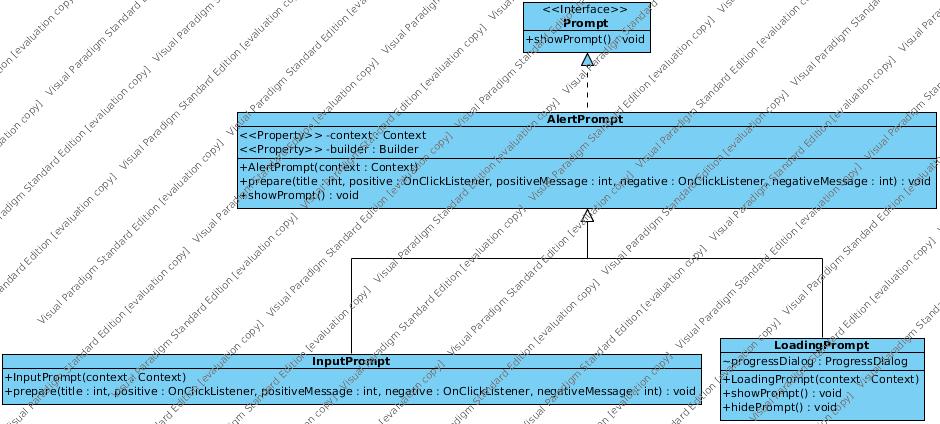
Modul webservice sadrži implementaciju vezanu uz kontaktiranje sa serverom. Konkretno, implementiran je AsyncTask koji poziva metodu u klasi ServiceCaller koja vrši slanje i primanje podataka. Ulazni parametri ovog modula su objekt tipa ServiceParams te objekt koji implementira sučelje ServiceResponseHandler ili SimpleResponseHandler, dok je izlaz objekt ServiceResponse koji je prosljeđen objektu koji implementira prethodno navedena sučelja. Klasa ServiceParams definira putanju (URL) na koju se šalju podatci zatim HTTP metoda (POST, GET, PUT, DELETE) te konkretni podaci koji moraju realizirati Serializable sučelje. Klasa ServiceResponse će na zahtjev odgovoriti sa HTTP kodom te konkretnim JSON-om. Klase ServiceParams i ServiceResponse kao i mogući odgovori poslužitelja navedeni su u tablici ranije u ovom dokumentu.

Slika 4: Dijagram klasa modula nfc

**Nfc modul**

Modul nfc brine se za prenošenje podataka NFC-om. Trenutno ovaj modul sadrži samo tzv. NFC beam operacije jer nam ostale nisu potrebne (pisanje na čip i čitanje sa čipa), ali prostor za nadogradnju ovog modula postoji. U ovom modulu definirana je apstraktna klasa koja nasljeđuje Activity klasu te implementira CreateNdefMessageCallback sučelje. Ova apstraktna klasa implementira dvije metode startNfcBeam i startNfcAdapter. Uz apstraktnu klasu definirane su dvije iznimke NfcNotAvailableException koja je bačena ukoliko uređaj ne podržava NFC te NfcNotEnabledException koja je bačena ukoliko na uređaju postoji NFC, ali on nije prethodno aktiviran. Ideja je da sve aktivnosti koje će koristiti NFC beam operaciju naslijede ovu klasu te pozovu startNfcAdapter i startNfcBeam metode. Implementacija koja se izvršava nakon operacije daje programer implementiranjem sučelja i njegovim prosljeđivanjem u metodu startNfcBeam.

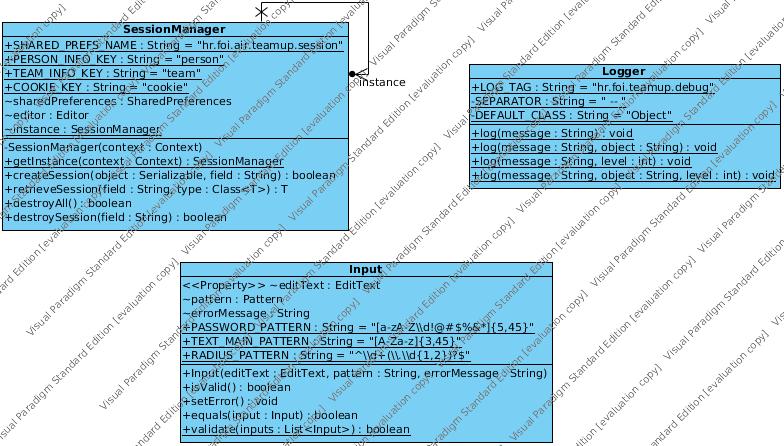
**Prompts modul**



Slika 5: Dijagram klasa modula prompts

Modul prompts je jednostavan modul u kojem je izdvojena implementacija za dijaloge koji mogu iskočiti korisniku na zaslon. Tako definiramo sučelje Prompt i klase LoadingPrompt koja prikazuje dijalog za učitavanje, InputPrompt koja očekuje korisnički unos te AlertPrompt koja prikazuje upozorenje korisniku.

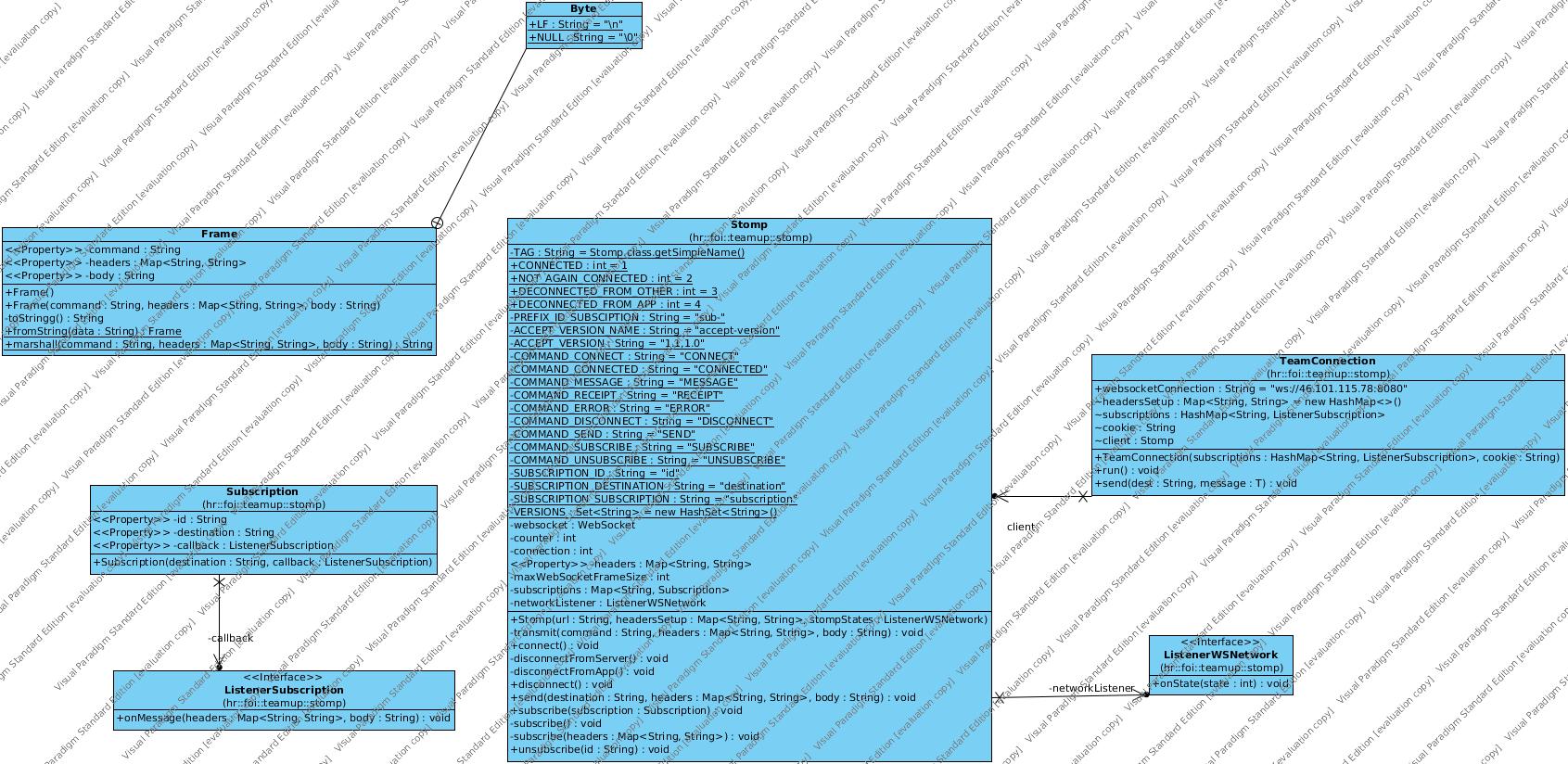
**Core modul**



Slika 6: Dijagram klasa modula core

Modul core sadrži sve one klase koje olakšavaju (helper klase) implementaciju određenih elemenata aplikacije te nema smisla svrstavati ih u zaseban modul jer se radi o manjoj količini koda. Ovdje se radi o klasama Input, Logger i SessionManager. Logger klasa je vlastita klasa za logiranje u zapisnik (logcat), Input klasa je klasa omotač za Androidov EditText koja nam olakšava validaciju korisničkog unosa, a SessionManager je klasa koja upravlja sa SharedPreferences koja nam je idealna za čuvanje određenih podataka u uređaju kako ne bismo morali svaki put dohvaćati relevantne podatke putem weba.

**Stomp modul**



Slika 7: Dijagram klasa modula stomp

Kako bi korisnici mogli sa serverom komunicirati putem STOMP protokola napravljen je modul za STOMP komunikaciju. Detalji o STOMP protokolu mogu se naći na sljedećem linku <https://stomp.github.io/>. Navedeni modul se sastoji od sljedećih klasa a to su redom Frame, Stomp, Subscription i TeamConnection te sučelja: ListenerSubscription te ListenerWSNetwork. Kako bi mogli komunicirati putem Stomp protokola potrebno je bilo defnirati Stomp okvire koji će se slati na webservis. Stomp okvir je definiran u klasi Frame koja sadrži Stomp naredbu, zaglavlje poruke te samo tijelo poruke koja se šalje. Više o klasi se nalazi u java docsima. Stomp radi na principu pub/subscribe modela stoga klasa Subscription sadrži detalje o pojedinoj pretplati. Unutar klase Subscription definirana je putanja pretplate, id pretplate te slušać pretplate( ListenerSubscription). Klasa Stomp nam služi kao klijent za spajanje na webservis putem websocketa i Stomp protokola. Sučeljem ListenerWSNetwork definirali smo što se treba izvršiti na određeno stanje websocket veze te sučeljem ListenerSubscription što će se izvršiti u trenutku kada se primi poruka od strane webservisa. TeamConnection klasa koja objedinjuje pozive prema web servisu te pozive prema sučelju za obradu poruka od webservisa.