SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

**Matija Čoklica, 45270/16-R**

**Valentina Krhlanko, 45268/16-R**

**Aleksandar Trajkov, 45280/16-R**

FoodDonor

PROJEKT IZ KOLEGIJA ANALIZA I RAZVOJ PROGRAMA

PROJEKTNA DOKUMENTACIJA

[Link na github repozitorij](https://github.com/valkrhlan/FoodDonor)

[Link na github repozitorij web servisa](https://github.com/valkrhlan/FoodDonorWebservice)

Varaždin, 2016.

Sadržaj

[1. Baza podataka 1](#_Toc473655942)

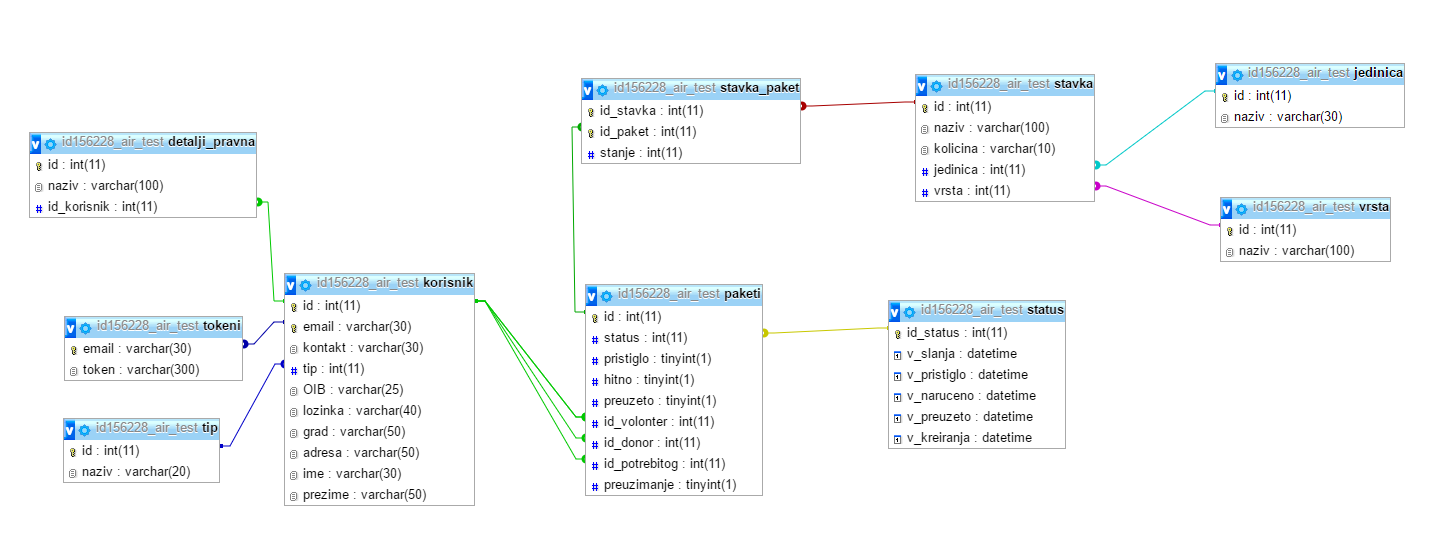
[2. Web servis 2](#_Toc473655943)

[3. Struktura aplikacije 11](#_Toc473655944)

[4. Dijagrami klasa 16](#_Toc473655945)

1. Baza podataka

Tablica korisnik sadrži zajednička obilježja svih tipova korisnika. Kao primarni ključ koristi se atribut id. Atribut email predstavlja oznaku korisničkog imena . Tablica tip sadrži moguće tipove korisnika koji kasnije služe za definiranje akcija kojima korisnik ima pristup. Trenutno su moguće vrijednosti : „volonter“,“ donor“ i „potrebiti“. Tablica detalji\_pravna je dodatna tablica koja pobliže opisuje korisnika (ako je pravna osoba mora imati i naziv poduzeća ili udruge), a popunjava se za korisnike koji su tipa „donor“ ili „potrebiti“.Tablica fizicka predstavlja dodatne informacije o tipu korisnika „. U tablici tokeni nalaze se trenutno važeći i pohranjeni tokeni za slanje push notifikacija pomoću firebase-a. Svaki korisnik ima određene interakcije vezane uz određeni paket, pa tako u bazi postoje tablice paket i status (sadrži bitne vremenske informacije o događajima vezanim uz kreiranje, naručivanje, preuzimanje i ostale akcije vezane uz pojedini paket). Svaki paket također može imati jednu ili više stavki pa u bazi postoje tablice stavka\_paket, stavka te jedinica i vrsta. Tablica stavka\_paketa služi kao posrednik u vezi n:n između tablica paketi i stavka jer jedan paket može imati više od jedne stavke. Svaka stavka ima određenu jedinicu koja predstavlja mjernu jedinicu količine i vrstu koje predstavlja skup najčešćih prehrambenih proizvoda. ERA model ove baze podataka prikazan je na slici 1. *ERA model*.



Slika 1: ERA model ([link na sliku](http://i.imgur.com/MSdUkAX.png))

1. Web servis

Kako aplikaciju istovremeno može koristiti više korisnika koji trebaju podatke iz zajedničke baze napravljen je PHP web servis. Pripadni web servis je RESTfull tipa te se za dohvaćanje ili manipulaciju podataka treba organizirati određena struktura URL-a. Link na github repozitorij web servisa je <https://github.com/valkrhlan/FoodDonorWebservice> Kao nepromjenjivi dio tj. baseUrl u aplikaciji korišten je url: <https://air-web-service.000webhostapp.com/webservice/>. Za prvu fazu implementirane su četiri metode, a njihov naziv, HTTP metodu i url prikazani su u tablici 1. Metode web *servisa*. Pristupni url se kreira tako da se *baseUrl*-u doda url metode.

|  |  |  |
| --- | --- | --- |
| WS METODA | HTTP METODA | URL |
| Dohvati sve gradove | GET | gradovi/all |
| Prijava | GET | prijava/{email}/{lozinka}/ |
| Registracija volontera | GET | registracija/{metoda}/{email}/{lozinka}/{oib}/{grad}/{adresa}/{kontakt}/{ime}/{prezime}/ |
| Registracija donora i potrebitih | GET | registracijaOstali/{metoda}/{email}/{lozinka}/{oib}/{grad}/{adresa}/{kontakt}/{naziv}/{tip}/ |
| Dohvaćanje svih vrsta hrane i jedinica količine | GET | vrstaJedinica/all/ |
| Pohrana novog paketa | GET | paket/novi/{email}/{json}/{prijevoz}/ |
| Dohvaćanje paketa za pojedinog korisika | GET | paket/dohvati/{email}/{odabrani}/{grad}/ |
| Pohrani firebase token u bazu | GET | registerDevice/{email}/{token}/ |
| Obriši firebase token iz baze | GET | obrisiToken/{email}/ |
| Pošalji notifikaciju na firebase servis | GET | sendNotifications/{email}/{title}/{message}/ |
| Dohvati nottifikacije za konfigurabilan prikaz obavijesti | GET | getNotifications/{email}/{timestamp}/ |
| Potrebiti je odabrao neki paket za preuzimanje i/ili poslao hitan signal | GET | odaberiPaketPotrebiti/{email}/{hitno}/{idPaketa}/ |
| Volonter je odabrao paket za prijevoz | GET | odaberiPaketVolonter/{email}/{idPaketa}/ |
| Evidentiraj dolazak volontera po paket | GET | evidentirajDolazak/{idPaket}/ |

Tablica 1: Metode web servisa

Svaka funkcija vraća odgovor JSON formata. Glavni atributi u odgovoru su status, nbResults, message i dana. Pri čemu status označava uspješno ili neuspješno dohvaćanje podataka sa web servisa, a prikazuju i ispravnost ili neispravnost pri izmjeni ili dodavanju podataka u bazu. Strukturu odgovora i moguće vrijednosti koje može poprimiti nalazi se u tablici *2. Struktura odgovora pri metodi za ispis svih gradova*, tablici *3. Struktura odgovora pri metodi za prijavu* ,tablici *4. Struktura odgovora pri metodama za registraciju* ,tablici 5. *Struktura odgovora pri metodi za dohvaćanje vrste hrane i mjernih jedinica, tablici 6. Struktura odgovora pri metodi za pohranu novog paketa*, tablici 7. *Struktura odgovora pri metodi za dohvaćanje paketa pojedinog korisnika*, tablici 8. *Struktura odgovora pri pohrani firebase tokena*, tablici 9. *Struktura odgovora pri brisanju tokena iz baze*, tablici 10. *Struktura odgovora pri slanju notifikacija na firebase servis*, tablici 11. *Struktura odgovora pri dohvaćanju notifikacija za konfigurbilan prikaz*, tablici 12. *Struktura odgovora pri odabiru paketa za preuzimanje i slanje hitnog signala od strane potrebitog*, tablici 13. *Struktura odgovora pri odabiru paketa za priejvoz od strane volontera* i tablici 14*. Struktura odgovora pri evidentiranju dolaska*.

|  |  |  |
| --- | --- | --- |
| RESPONSE | Nema gradova u bazi | Dohvaćeni su gradovi |
| status | OK | OK |
| nbResults | 0 | Broj dohvaćenih gradova |
| Message | „Nema ni jednog grada u bazi!“ | „Pronađeni su gradovi“ |
| data | Niz pbr = -1 i naziv =““ | Niz svih gradova sa vrijednostima pbr i naziv u JSON formatu |

Tablica 2: Struktura odgovora pri metodi za ispis svih gradova

Svi mogući odgovori prilikom metode za prijavu vidljivi su u tablici *3.* Za registraciju postoje dvije funkcije, jedna omogućava registraciju volontera, tj. fizičkih osoba, a druga omogućava registraciju donora i potrebitih koji su pravne osobe. Obje funkcije za registraciju vraćaju odgovor iste strukture,pa su zajedno obuhvaćene u tablici 4*. Struktura odgovora pri metodama za registraciju*

|  |  |  |  |
| --- | --- | --- | --- |
| RESPONSE | Uspješna prijava | Neuspješna prijava (krivi uneseni podatci) | Neuspješna prijava (nedostupna baza podataka) |
| status | OK | NOT OK | NOT OK |
| nbResults | Tip korisnika ( 1 – „donor“, 2 – „volonter“ i 3 –„ potrebiti“) | 0 | 0 |
| message | „Uspješna prijava“ | tekst greški | "Došlo je do problema na web servisu.“ |
| data | Niz prijava = „OK“ | Niz kojem je prijava = „error“ | Niz kojem je prijava = ''error'' |

Tablica 3:Struktura odgovora pri metodi za prijavu

|  |  |  |
| --- | --- | --- |
| RESPONSE | Uspješna registracija | Neuspješna registracija |
| status | OK | NOT OK |
| nbResults | 0 | 0 |
| Message | „Uspješna registracija“ | tekst greški |
| data | Niz reg = „OK“ | Niz kojem je reg= „error“ |

Tablica 4: Struktura odgovora pri metodi za registraciju

|  |  |  |
| --- | --- | --- |
| RESPONSE | Nema vrsta i hrane u bazi | Dohvaćeni su vrste i jedinice |
| status | OK | OK |
| nbResults | 0 | 0 |
| Message | „Nema zapisa u bazi!“ | „Pronađeni su rezultati“ |
| data | Atributi elemenata u vrsta i jedinice su id = -1 i  naziv =““ | Niz svih gradova sa vrijednostima vrsta i jedinica u JSON formatu. Vrsta i jedinice imaju atribute id i naziv |

Tablica 5. Struktura metode pri dohvaćanju svih vrsta hrane i njihovih mjernih jedinica

|  |  |  |
| --- | --- | --- |
| RESPONSE | Paket nije dodan | Pake uspješno dodan |
| status | NOT OK | OK |
| nbResults | 0 | 0 |
| Message | Tekst greški | „Uspješno dodano!“ |
| data | Niz s atributom dodavanje i njegovom vrijednosti error | Niz s nazivom atributa dodavanje i njegovom vrijednosti OK |

Tablica 6. Struktura odgovora pri metodi za pohranu novog paketa

|  |  |  |
| --- | --- | --- |
| RESPONSE | Nije dohvaćen ni jedan paket | Uspješno dohvaćeni paketi |
| status | NOT OK | OK |
| nbResults | 0 | Broj dohvaćenih paketa |
| Message | "Nema paketa za traženog korisnika" ili "Korisnik ne postoji u bazi" | "Uspješno dohvaćanje" |
| data | Niz s atributom paket (vrijednost mu je „error“) | Niz svih paketa sa atributima: id, preuzimanje, hitno,id\_volonter, naziv\_volonter, id\_donor, naziv\_donor, id\_potrebitog, naziv\_potrebitog, preuzimanje, v\_kreiranja, v\_naruceno, v\_preuzeto, v\_slanja, v\_pristiglo, stavke (niz s atributima: id, naziv, količina,id\_vrsa,vrsta,id\_jedinica,jedinica) |

Tablica 7. Struktura odgovora pri metodi za dohvaćanje paketa pojedinog korisnika

|  |  |
| --- | --- |
| RESPONSE | Dodan token |
| status | OK |
| nbResults | 0 |
| Message | „Dodan token“ ili „Izmijenjen token“ |
| data | Niz s atributom evidentiranje (vrijednost mu je „provedeno“) |

Tablica 8. Struktura odgovora pri pohrani firebase tokena

|  |  |
| --- | --- |
| RESPONSE | Obrisan token |
| status | OK |
| nbResults | 0 |
| Message | „Obrisan token“ |
| data | Niz : 'token'=>'obrisan' |

Tablica 9. Struktura odgovora pri brisanju tokena iz baze

|  |  |  |
| --- | --- | --- |
| RESPONSE | Neuspešno slanje/Nema ni jedne obavijesti | Poslane obavijesti |
| status | NOT OK | OK |
| nbResults | 0 | Broj poslanih obavijesti |
| Message | „Nije poslana ni jedna notifikacija“ | „Poslane notifikacije“ |
| data | 'notifikacija'=>'nema' | 'notifikacija'=>'ima' |

Tablica 10. Struktura odgovora pri slanju notifikacija na firebase servis

|  |  |  |
| --- | --- | --- |
| RESPONSE | Neuspješno dohvaćanje | Uspješno dohvaćanje |
| status | NOT OK | OK |
| nbResults | 0 | 0 (kad nema ni jedne nove obavijesti) ili broj novih obavijesti |
| Message | "Greška pri spajanju na bazu!" | „Nema ni jedne nove obavijesti!“ ili "Ima novih notifikacija!" |
| data | 'notifikacije' => 'error' | Niz notifikacija s atributima title i message |

Tablica 11. Struktura odgovora pri dohvaćanju notifikacija za konfigurbilan prikaz.

|  |  |  |
| --- | --- | --- |
| RESPONSE | Neuspješan odabir/slanje signala | Uspješan odabir/slane hitnog signala |
| Status | NOT OK | OK |
| nbResults | 0 | 1 |
| Message | „Nepostojeći korisnik.“ | „Paket odabran!“ ili „Poslan hitan signal!“ |
| data | Niz 'odabraniPaketi'=>“error“ | Niz 'odabraniPaketi'=>“OK“ |

Tablica 12. Struktura odgovora pri odabiru paketa za preuzimanje i slanje hitnog signala od strane potrebitog

|  |  |  |
| --- | --- | --- |
| RESPONSE | Neuspješan odabir | Uspješan odabir |
| Status | NOT OK. | OK |
| nbResults | 0 | 1 |
| Message | „Nepostojeći korisnik.“ i/ili „Nepostojeći paket.“ | "Paket odabran!" |
| data | Niz 'odabraniPaketi'=>“error“ | Niz 'odabraniPaketi'=>“OK“ |

Tablica 13. Struktura odgovora pri odabiru paketa za prijevoz od strane volontera

|  |  |  |
| --- | --- | --- |
| RESPONSE | Neuspješno evidentiran dolazak | Uspješno evidentiran dolazak |
| Status | NOT OK | OK |
| nbResults | 0 | 1 |
| Message | „Nepostojeći paket.“ | „Evidentirano!“ |
| data | Niz 'evidentiranDolazak' => "error" | Niz 'evidentiranDolazak' => "OK" |

Tablica 14. Struktura odgovora pri evidentiranju dolaska

1. Struktura aplikacije

Struktura se sastoji od tri modula (app, core i webservice). Unutar app modula nalaze se paketi i klase potrebni za sam prikaz podataka, obradu podataka, listeneri za web servis i nekoliko paketa. Unutar core modula nalaze se klase koje su potrebne app i core modulu. Nalaze se upravo tu jer se većina tih klasa koristi kod parsiranja podataka sa web servisa, a slijedno tome koriste se i za kreiranje, ažuriranje ili slanje podataka unutar app modula. Modul web service služi za dohvaćanje i slanje podataka na php web servis.

* 1. Modul webservice

U sklopu ovog modula imamo paket „responses“ unuta kojeg se nalazi klasa „FdWebServiceResponse“ (čini strukturu odgovora koji primamo od web servisa). Uz paket tu se nalaze klasa „FdWebService“ te sučelja „FdWebServiceCaller“ i „FdWebServiceHandler“. Interface „FdWebService“ sadrži retrofit anotacije te putanje sa parametrima za kreiranje HTTP zahtjeva.“FdWebServiceHandler“ je interface sa metodom za prosljeđivanje dobivenog rezultata od web servisa prema aplikaciji, tj. koristi se za proslijeđivanje poruke odgovora (Parametar message može sadržavati popis paketa, običan tekst koji govori o uspješnosti ili neuspješnosti obavljene akcije na web servisu koju smo pozvali i slično.). Klasa „FdWebServiceCaller“ sastoji se od metoda za prosljeđivanje parametara iz aplikacije prema HTTP zahtevu koji se generira, ali i za handlanje odgovora koji nam web servis vraća i njihovo prosljeđivanje listeneru, tj konkretno sučelju fdWebServiceHandlerer.

* 1. Modul Core

Služi kao posrednik za klase koje trebaju modulima app i webservice za komunikaciju. Sastoji se od 11 klasa. Pri čemu klase „VrstaJedinica“ i „SpinnerElement“ služe za prikaz vrstae i količinskih jednica hrane prilikom dodavanja paketa. „GooglaMapa“ sadržava kordinate za prikaz na mapi kako bi volonter lakše vidio mjesto preuzimanja i mjesto dostave paketa. Klase „Gradovi“ ,“Korisnik“, „Notifikacija“ i „RegistriraniKorisnik“ koriste se za prikaz u različitim popisima i parsiranje ili slanje podataka web servisu. Neintuitivna je jedino klasa „Paket\_Parcelabler“ koja služi za prijenos podataka između fragmenta, o potrebna je za poštivanje bundle strukture.

* 1. Modul app

Sastoji se od paketa „adapters“, “firebase“, “fragments“, “klase“, “konfigurabilno“ i „loaders“ te klasa koje predstavaljaju aktivnosti: „GlavnaAktivnost“, „MapaPaket“, „NotifikacijeOpcije“, „Prijava“, „RegistracijaFizickiKorisnik“ i „RegistracijaPravniKorisnik“.

* + 1. Paket „klase“

Paket klase sastoji se od nekih najbitnijih klasa za manipulaciju podatcima i/ili komunikaciju između modula. Posebno je važno istaknuti klase „NotifikacijaMoguceOpcije i „UpraviteljNotifikacija“. Klasa „NotifikacijeMoguceOpcije“ koristi se za povezivanje sa modulima, unutar nje se definira novi modul za slanje notifikacija sa ključnom riječi, intervalima i instancom listenera sa strukturom definiranom unutar klase „ItemNotifikacija.MoguceOpcije.

Sljdeća bitna klasa je „UpraviteljNotifikacija“ koji se koristi prilikom dinamičkog generiranja izbornika za odabir načina dobivanja notifikacija, ali i za modularnu komunikaciju s „firebase“ i „konfigurabilno“ paketom. Zajedno sa klasom „NotifikacijaLoadedListener“ (nalazi se u paketu „loaders“) čini listener za modularnu komunikaciju od modula.

Klasa „ContextProvider“ služi da dohvaćanje konteksta aplikacije, najviše se koristi za dohvaćanje konteksta kod prikaza notifikacije, jer neki moduli teško dolaze do konteksta, te se tako sprječava korištenje neprikladnog konteksta.

* + 1. Paket „adapters“

Paket „adapters“ sadrži klase koje služe kao adapteri za popunjavanje layouta na fragmentima.

* + 1. Paket „loaders“

Paket sa koji sadrži jedan listner (klasa „WsDataLoader“ i sučelje „WsDataLoadedListener“) te dva sučelja, „NotifikacijeLoadedListener“(koristi se kao dio listenera za komunikaciju od modula prema aplikaciji, tj klasi „Upravitelj notifikcija“) i “SlanjePodatakaModulimaListener“ (upotrebljava se kao dio listenera za modularnu komunikaciju prema firebasu i konfigurabilnom načinu prikazivanja notifikacija)

* + 1. Paket „fragments“

Paket koji se sastoji od fragmenata „DetaljiPaketa“, “DonorNoviPaket“, „GradoviFragment“, „PopisPaketa“ i „PotrebitiOdabraniPaketi“. Klasa „DetaljiPaketa“ sadrži logiku potrebnu za prikaz odabranog paketa te ovisno o gumbu na koji se klikne omogućuje prikaz karte, odabir paketa ili povratak na popis paketa. Klasa „DonorNoviPaket služi za dodavanje novih stavki te kreiranje novog paketa koji se šalje web servisu. U slučaju uspješnog dodavanja paketa na web servisu obavještavaju se potrebiti korisnici. Klasa „GradoviFragment“ koristi se za prikaz podataka kod volontera. Na temelju odabranog grada(mjesto odakle je donor) u fragmentu za prikaz popisa paketa prikažu se paketi donora koji se nalaze u odabranom gradu. Klasa „PopisPaketa“ služi za prikaz liste paketa. Prikazani podatci razlikuju ovisno o tipu prikazanog korisnika, npr. donor vidi samo svoje pakete, potrebiti vidi sve pakete koje može odabrati, ali također vidi i one pakete koje je već odabrao, a volonter slično kao i potrebiti svidi sve pakete koje može prevesti ili koje je odabrao za prijevoz. Klasa „PotrebitiOdabraniPaketi“ omogućuje prikaz odabranih paketa potrebitog korisnika i volontera na posebnom fragmentu.. Klikom na neki od paketa iz liste view-a prebacuje se na detalje kliknutog paketa.

* + 1. Paket „firebase“

Paketu kojem se odvija sve vezano uz modularno dohvaćanje notifikacija pomoću firebasea. Unutar paketa nalaze se četiri klase: „MyFirebasenstanceIDService“, “MyFirebaseMessagingService“, „MyNotificationManager“ i „SharedPrefManager“. Pri čemu klase „MyFirebaseInstanceIDService“ i „SharedPrefManager“ služe za pohranjivanje tokena. „MyFirebaseMessagingService“ klasa koja prima json objekt od Firebase servisa, parsira ga i proslijeđuje klasi MyNotificationManager. Klasa „MyNotificationManager“ dobiva notifikaciju sa naslovom te porukom te na temelju toga obavještava klasu „UpraviteljNotifikacija“ (upravlja modularnim paketima za prikaz notifikacija) da je notifikacija pristigla te da ju treba prikazati. Također šalje podatke listeneru za upravljanje notifikacija da treba neku metodu i implementira upravljanje odgovorom sa web servisa koji je proslijeđen. Pri pozivu web servisa traži dodavanje ili brisanje tokena ovisno o tome koja je opcija prethodno bila odabrana i koja je sad odabrana.

* + 1. Paket „kofigurabilno“

Paket koji sadrži klase potrebne za pokretanje alarma, točnije BroadcastReceivera koji se okida određeni interval i dohvaća promjene sa web servisa i šalje notifikacije upravitelju notifikacija za njihov prikaz. Sastoji se od klasa „Alarm“, “AutoStar“, “KonfigurabilnoListener“ i „MyAlarmService“. Klasa „MyAlarmService“ i „AutoStart“ omogućuje pokretanje alarma nakon reboota uređaja. Klasa „Alarm“ proširuje „BroadcastReceivera“ te se na određeni interval obavljaju određene akcije (npr. klasi „KonfigurabilnoListener se šalje zahtjev za povezivanjem na web servis koji ona dalje propagira do listenera prema tom modulu). Klasa „KonfigurabilnoListener“ služi za modularnu komunikaciju između upravitelja notifikacija i modularnog kofigurabilnog prikaza notifikacija. Implementira sučelje „SlanjePodatakaModulima“. Omogućuje modularan poziv potrebne metode sa web servisa, dostavljanje podataka preko sučelja za prikaz notifikacija i obradu promjene opcije u izborniku.

* + 1. Aktivnosti u sklopu app modula

Također u sklopu app modula postoje već ranije navedene klase koje predstavljaju aktivnosti. Popis aktivnosti s pojašnjnjem i ključnim metodama nalazi se u nastavku:

* + - 1. „GlavnaAktivnost“

Klasa koja proširuje „AppCompatActivity“, koristi se za prikaz i izmjenu različitih fragmenata -> popis paketa, odabrani paketi za pojedinog korisnika, fragment za dodavanje novog paketa te fragment za prikaz karte. Na njoj je postavljen i meni u kojem korisnik može odabrati mogućnost odjave(„Odjava“) ili postavke za notifikacije („Notifikacije“).

* + - 1. „MapaPaket“

Klasa koja implementira sučelja „OnMapReadyCallback“ i „WsDataLoadedListener“. Služi za prikaz destinacija donora i potrebitog za odabrani paket kako bi volonter znao od kuda do kuda mora prevesti paket.

* + - 1. „NotifikacijeOpcije“

Klasa koja služi za dinamičko generiranje mogućih opcija definiraniu u klasi „NotifikacijeMoguceOpcije“. Radio Group za opcije se kreira samo jednom i to prilikom kreiranja aktivnosti, a radio Group za intervale se dinamički generira ovisno o trenutno odabranoj opciji.

* + - 1. „Prijava“

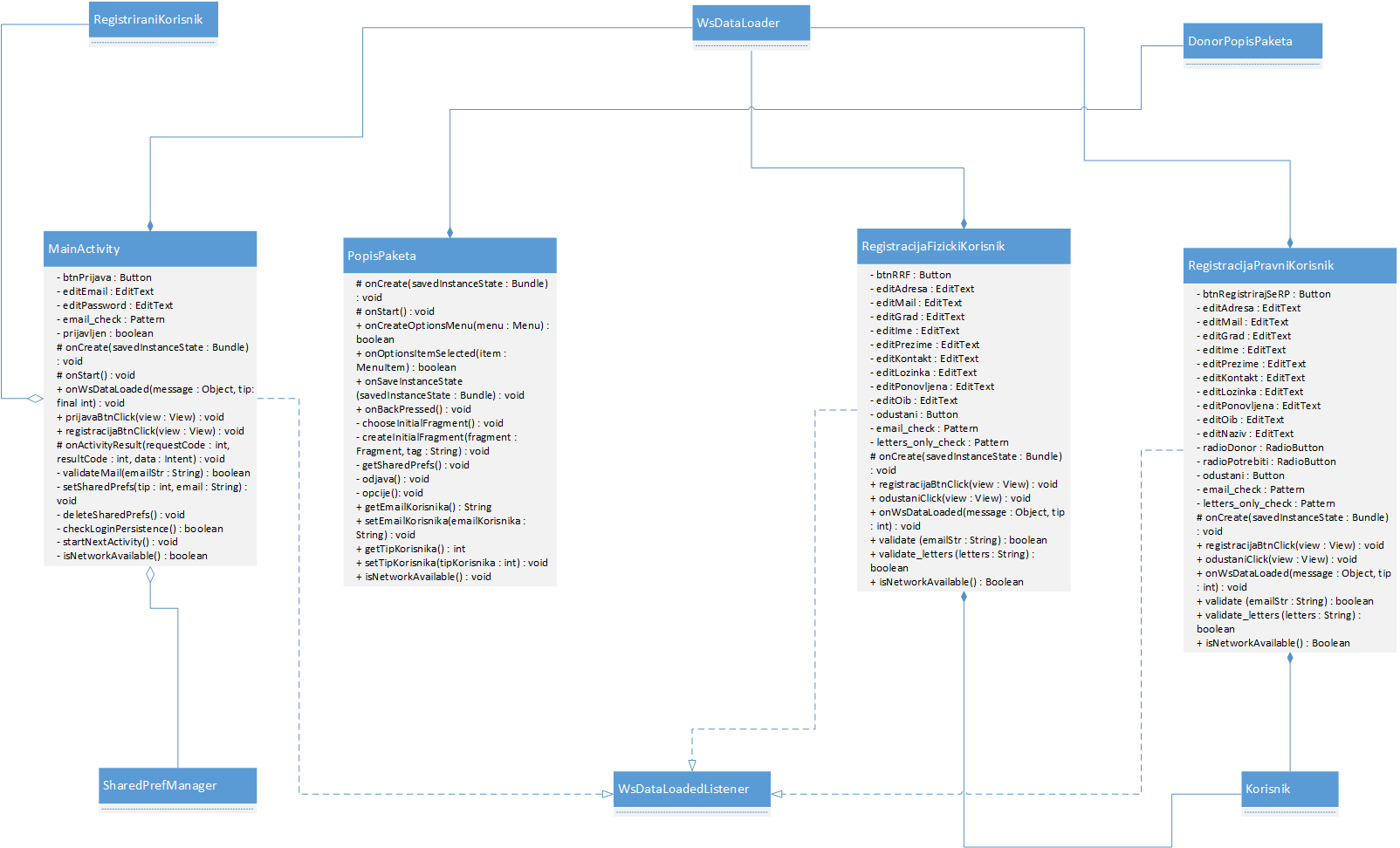
Klasa u kojoj se provjerava da li je Internet uljučen. Ako je onda se korisnik može prijaviti u aplikaciju, ako nije prijava u je omogućena jer se cijeli radi aplikacije temelji na komunikaciji sa web servisom. U sklopu klase se uz provjeru intereta omogućuje i porsljeđivanje na registraciju fizičke ili privatne osobe. U slučaju da je Internet dostupan, omogućuje i prijavu korisnika u aplikaciju, te pohranu firebase tokena u bazu na web servisu.

* + - 1. „RegistracijaFizickiKorisinik“ i „RegistracijaPravniKorisnik“

Klase u kojima se popunjavaju podatci za registraciju korisnika. Svaka sadrži metode za validaciju korisničkog unosa. Nakon ispravnog unosa podatci se šalju web servisu. Nakon uspješne validacije ili odustajanja od registracije korisnik se preusmjerava na aktivnost za prijavu.

1. Dijagrami klasa

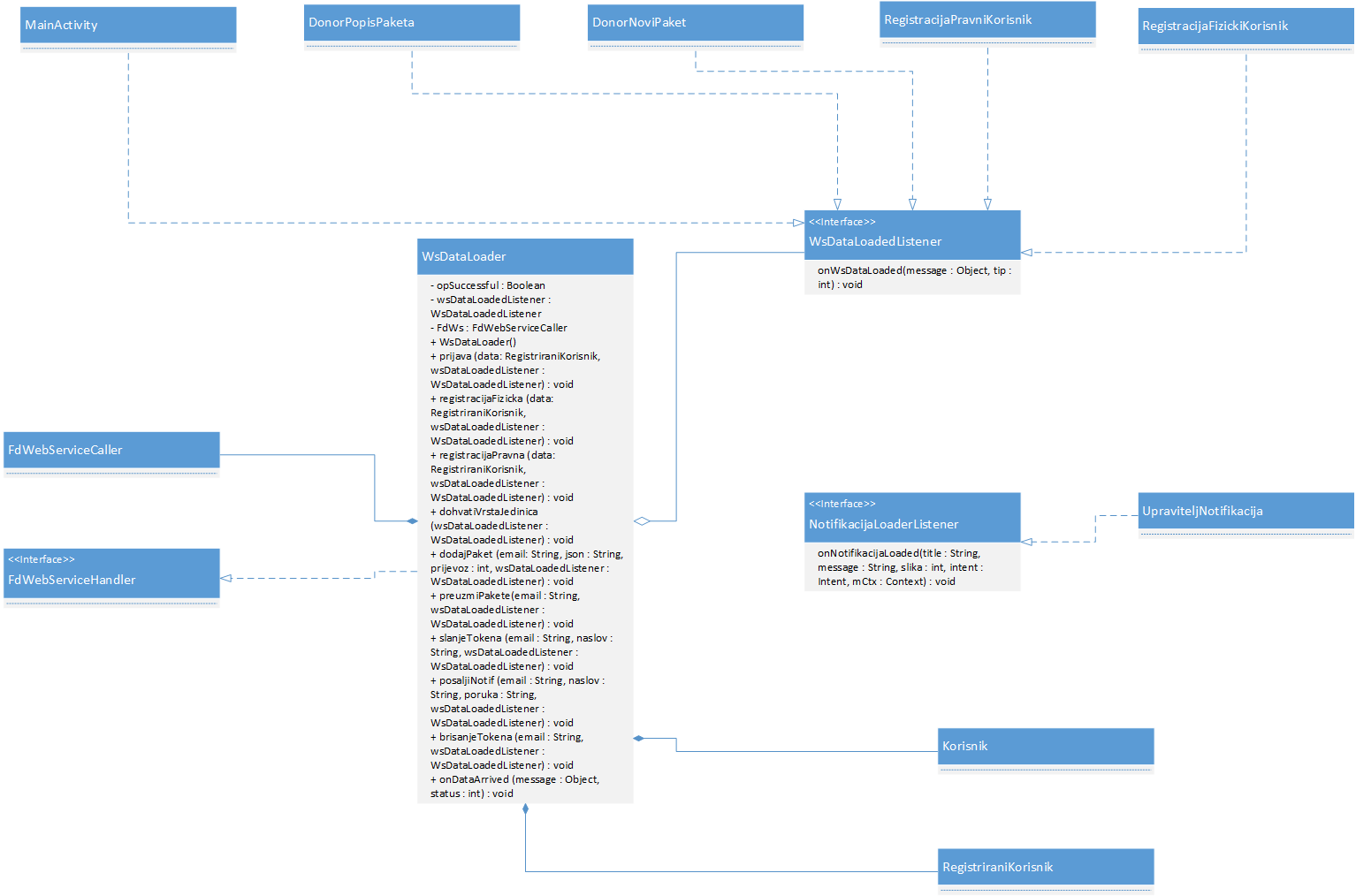
U aplikaciji se koriste tri modula – *app*, *core*, *webservice*. Početna aktivnost nalazi se u *app* modulu, zajedno sa svim ostalim aktivnostima, fragmentima, adapterima za listu, te klasama za upravljanje push notifikacijama. Modul *core* sadrži POJO klase koje opisuju entitete poput korisnika, paketa, notifikacije, te podataka za padajuće izbornike. Modul *webservice* brine se o slanju poslanih podataka iz aplikacije, te slanju dobivenih podataka iz web servisa natrag u aplikaciju (prosljeđuje ih preko sučelja prema aktivnostima i fragmentima). Sve do 4. sprinta, aplikacija *FoodDonor* broji 31 klasu i 4 sučelja. Od 31 klase, 5 je aktivnosti i 2 fragmenta.



Slika 2: Dijagram klasa za modul app

Na slici 2. nalazi se dijagram klasa za app modul, a detaljan opis klasa nalazi u nastavku:

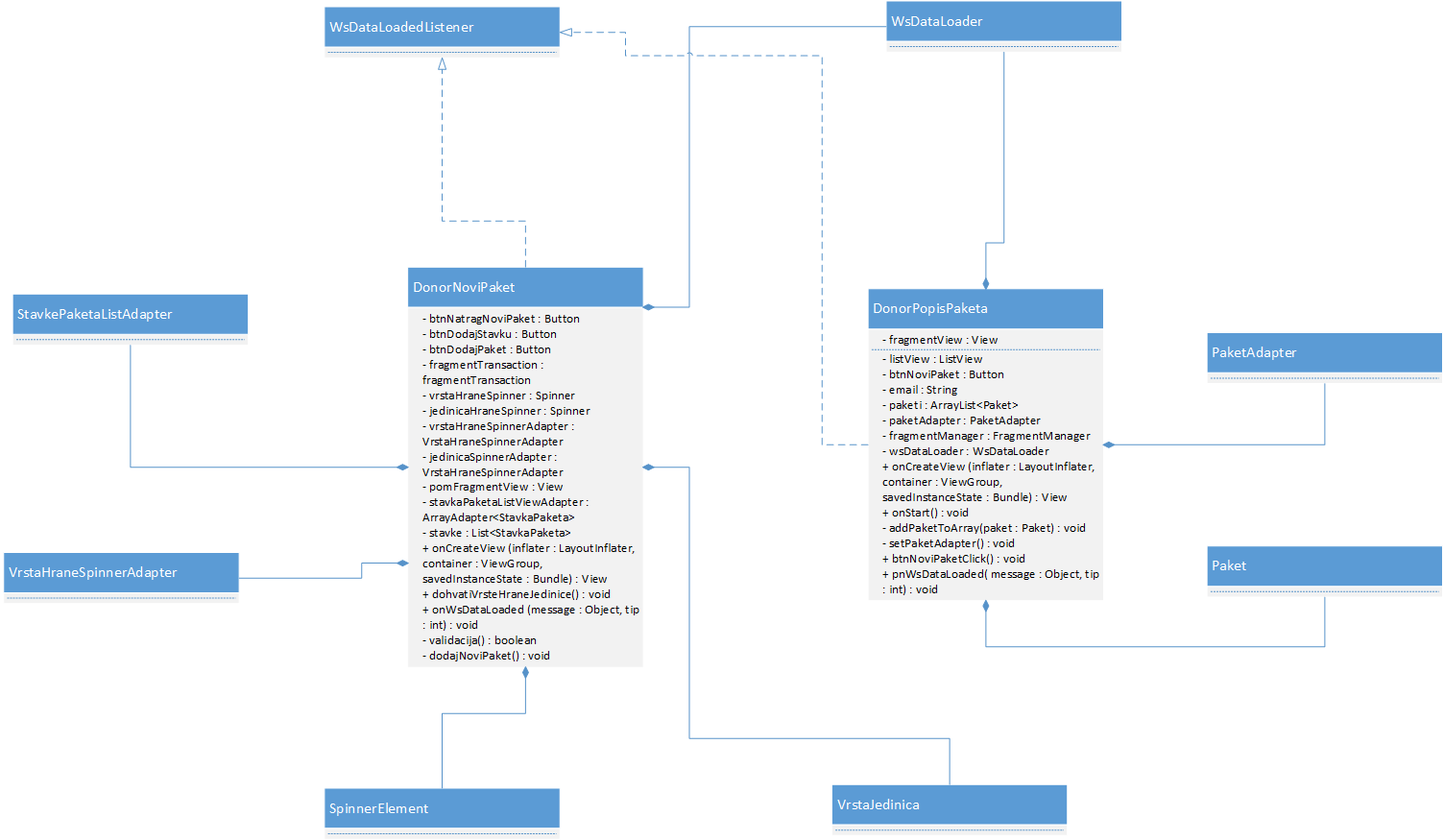
* MainActivity : Aktivnost u kojoj je implementirana prijava na aplikaciju, te poveznica (preko pomoćnog prozora s pitanjem) na aktivnosti koje implementiraju registraciju. Ako je prijava uspješna, podaci se spremaju u shared preferences.
* RegistracijaFizickiKorisnik : Aktivnost u kojoj je implementirana registracija fizičkog korisnika. Uspješnom registracijom, ili odabirom na odustani, korisnik se vraća na MainActivity aktivnost.
* RegistracijaPravniKorisnik : Aktivnost u kojoj je implementirana registracija fizičkog korisnika. Uspješnom registracijom, ili odabirom na odustani, korisnik se vraća na MainActivity aktivnost.
* PopisPaketa : Nakon uspješne prijave, prikazuje se ova aktivnost. U njoj je implementiran action bar, te „prostor“ koji služi za dodavanje fragmenata.



Slika 3: Dijagram klasa za modul app, paket loaders

U sklopu app modula nalazi se i paket loaders čiji dijagram klasa je prikazan na slici 3., pojedinosti klasa nalaze se u nastavku:

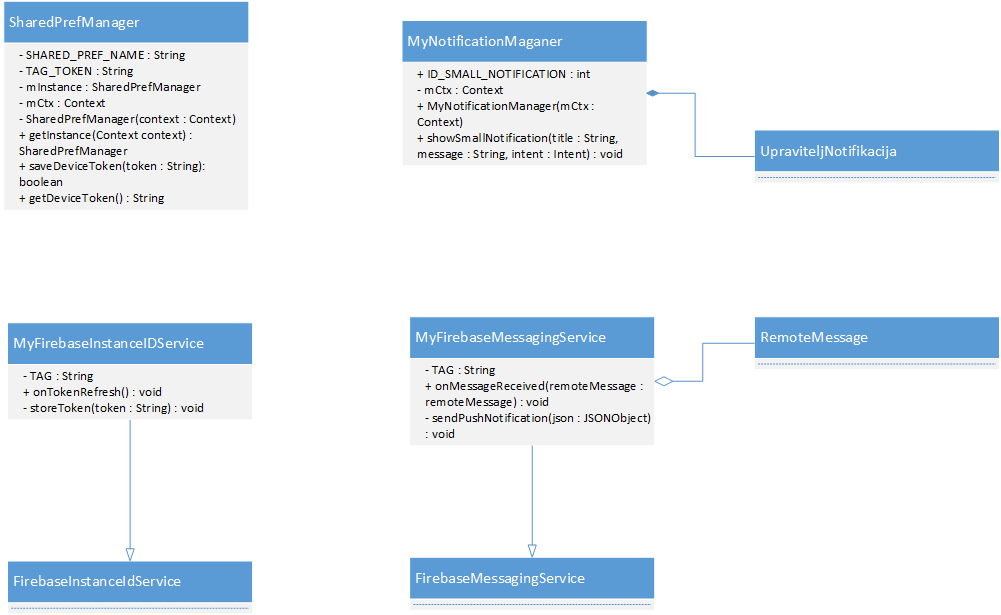
* WsDataLoader : Klasa koja posreduje komunikaciju između app i webservice modula. Prosljeđuje podatke između ta dva modula pomoću sučelja koje implementira (FdWebServiceHandler) i sučelja koje poziva (WsDataLoadedListener).
* WsDataLoadedListener : Sučelje koje moraju implementirati sve aktivnosti i fragmenti ako žele primiti podatke iz web servisa.



Slika 4: Dijagram klasa za modul app, paket fragments

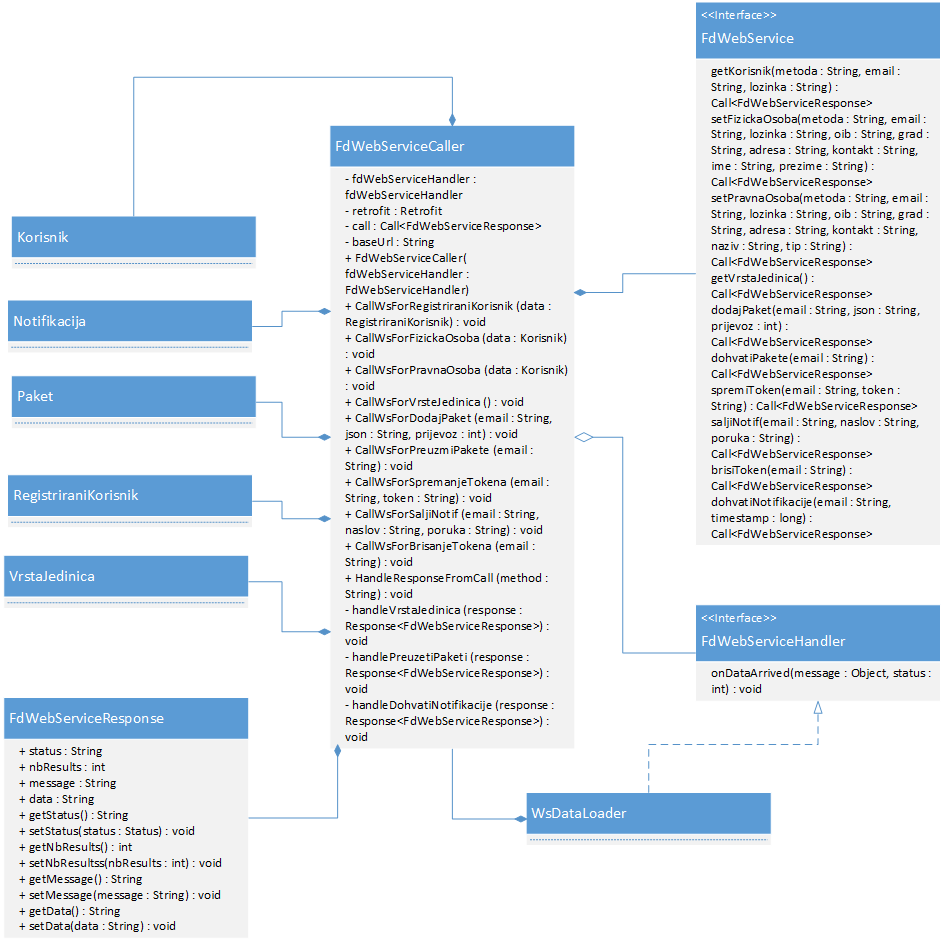
Dijagram klasa za paket fragments prikazan je na slici 4., a detalji klasa iz paketa navedeni su u nastavku:

* DonorNoviPaket : Fragment koji aktivira donor kada želi dodati novi paket. Unutar fragmenta moraju se popuniti prazna polja (sa podacima o nazivu hrane i količini), odabrati željena vrsta i mjerna jedinica, te odabir o prijevozu hrane. Implementiran je list adapter koji sprema napravljene stavke paketa. Nakon uspješno poslanog paketa, aplikacija se natrag prebacuje na fragment DonorPopisPaketa.
* DonorPopisPaketa : Fragment koji se inicijalno stvara u aktivnosti PopisPaketa, nakon uspješne prijave donora. Sadrži list adapter koji u sebi ima sve pakete (trenutno prijavljenog donora) koji još nisu došli do potrebitih korisnika, te gumb kojim se stvara novi paket.



Slika 5: Dijagram klasa za modul app, paket firebase

Paket firebase sadrži klase koje upravljaju registracijom korisničkog mobilnog uređaja, primanjem registracijskog tokena iz firebase-a, te primanjem notifikacija od strane firebase web notification servisa. Nakon prijave korisnika, obavlja se (asinkrono) registracija uređaja, te se tokom rada aplikacije (trenutno aktivnom radu ili radu u pozadini) sluša na pristignute notifikacije. Kada podaci za notifikaciju stignu, Notification Manager, predaje rad klasi UpraviteljNotifikacija, koja ujedno napravi notifikaciju sa poslanim podacima na korisnikovom mobilnom uređaju. Dijagram klasa za navedeni paket nalazi se na slici 5.



Slika 6: Dijagram klasa za modul webservice

Dijagram klasa za module webservice prikazan je na slici 6.,a detalji pojedine klase navedeni su u nastavku:

* FdWebServiceResponse : POJO klasa koji definira izgled odgovora od strane web servisa.
* FdWebServiceHandler : Sučelje koje javlja svim klasama koje je implementiraju da je web servis vratio odgovor. Ovu klasu implementira jedino WsDataLoader, koji je ujedno i zadužen za komunikaciju app modula i webservice modula.
* FdWebService : Sučelje koje korisni retrofit. U njemu su definirani svi API pozivi prema web servisu.
* FdWebServiceCaller : Klasa koja šalje podatke dobivene od WsDataLoadera, te dobiva podatke direktno od web servisa (te ih po potrebi filtrira i pošalje natrag WsDataLoader-u).