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

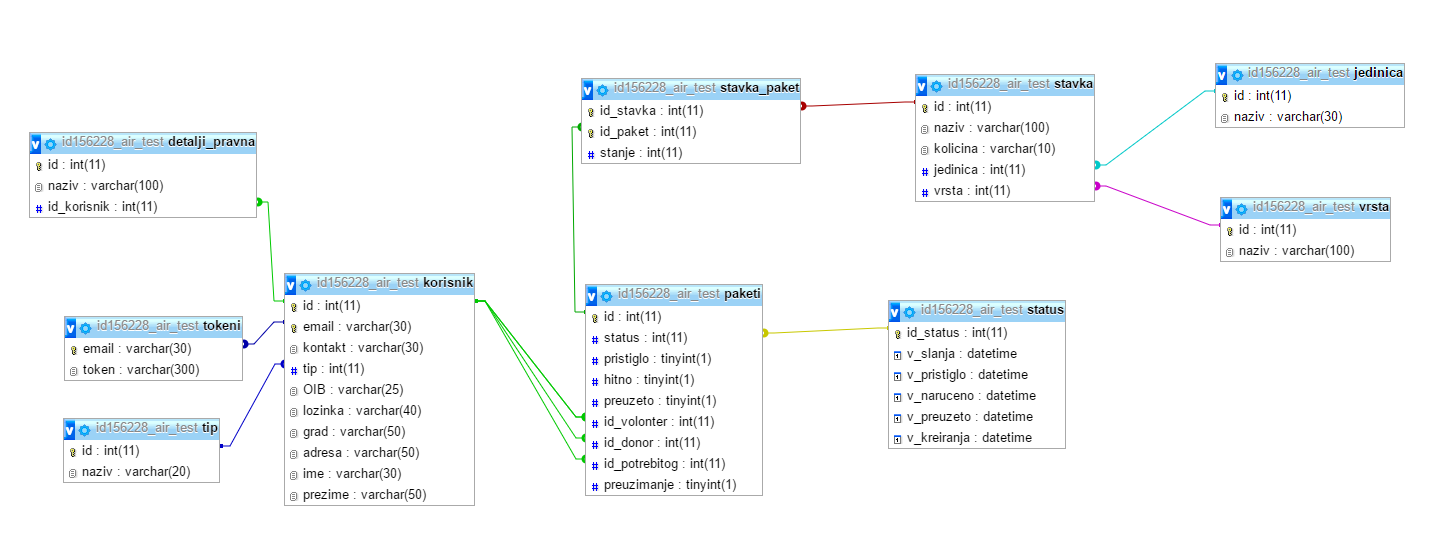
[Baza podataka 1](#_Toc471210737)

[Web servis 2](#_Toc471210738)

[Struktura aplikacije 9](#_Toc471210739)

# 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))

# 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}/ |
| 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}/ |

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 možemo vidjeti 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* i tablici 11. *Struktura odgovora pri dohvaćanju notifikacija za konfigurbilan prikaz*.

|  |  |  |
| --- | --- | --- |
| 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' |

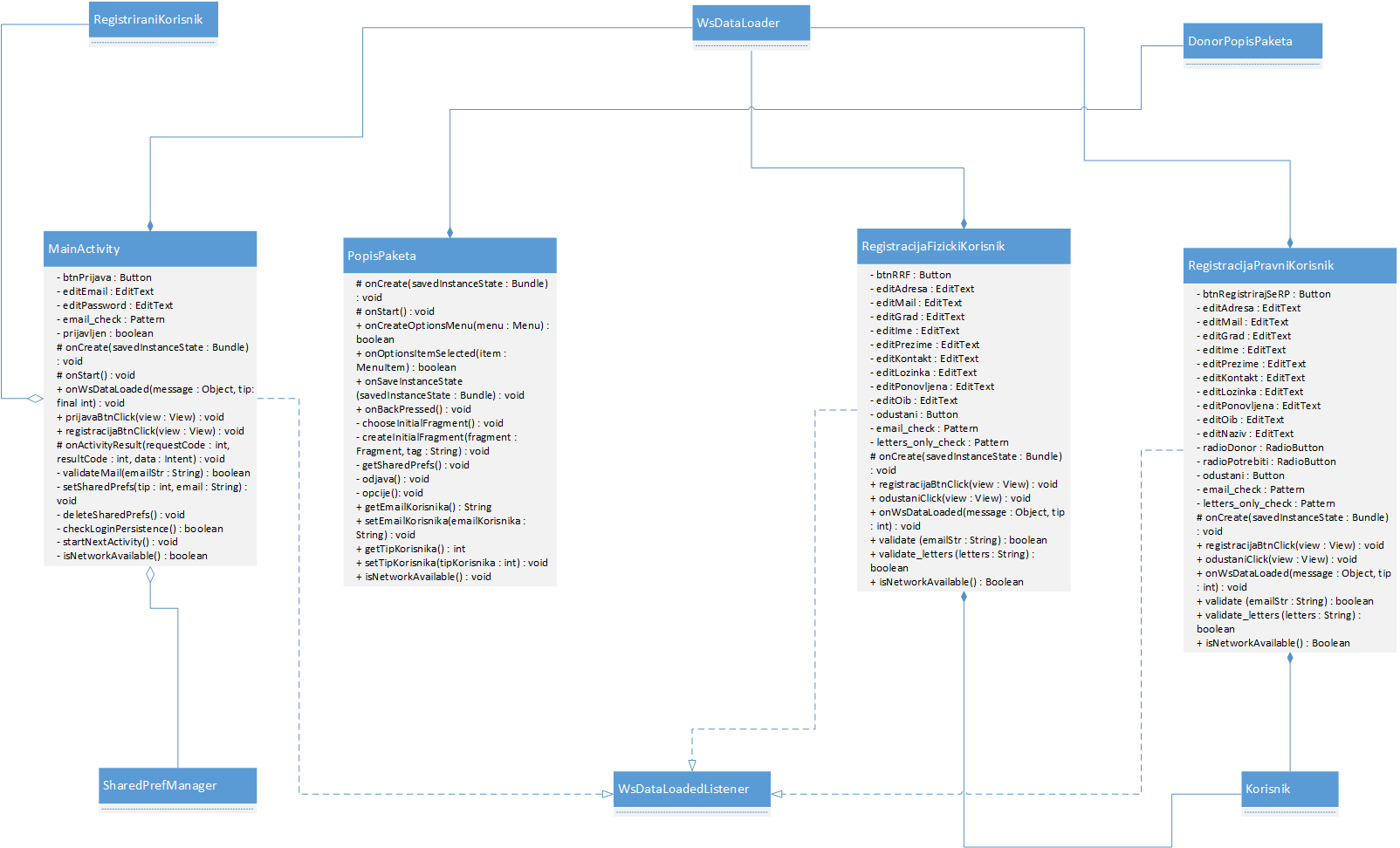
Tablica 10. Struktura odgovora pri slanju notifikacija na firebase servis

|  |  |  |
| --- | --- | --- |
| 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.

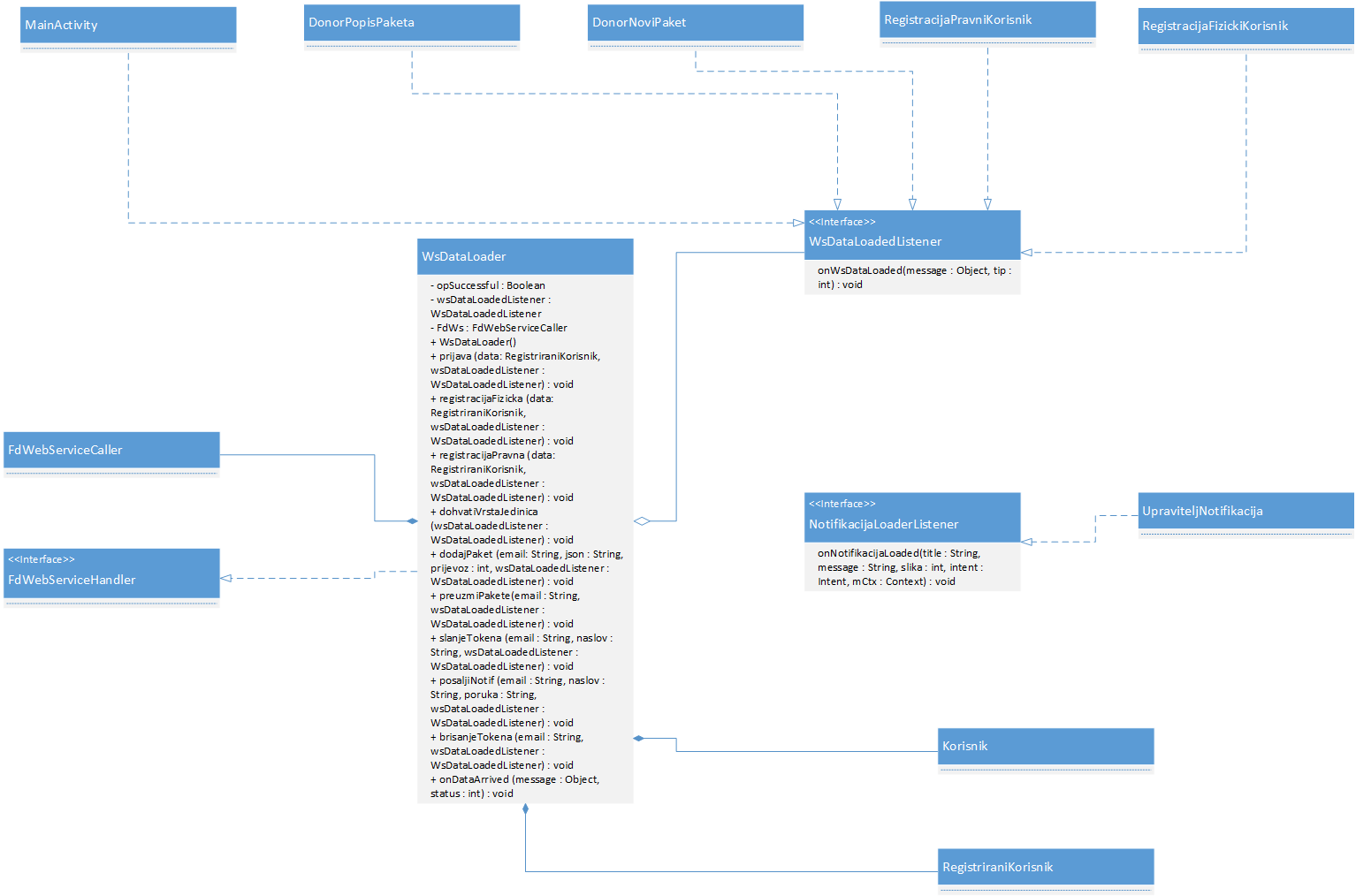
# Struktura aplikacije

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.



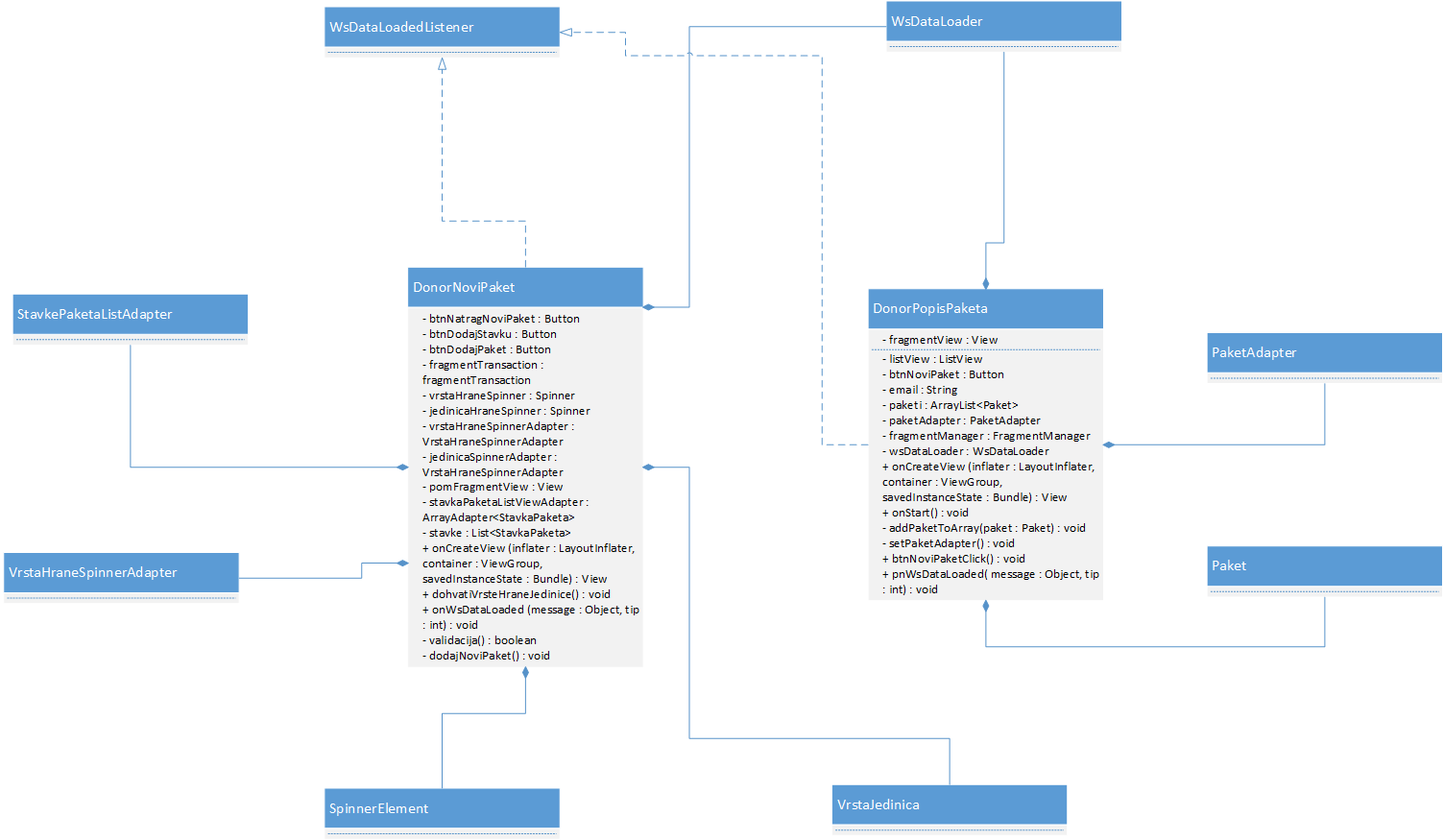
Slika 2: Dijagram klasa za modul app

* 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.



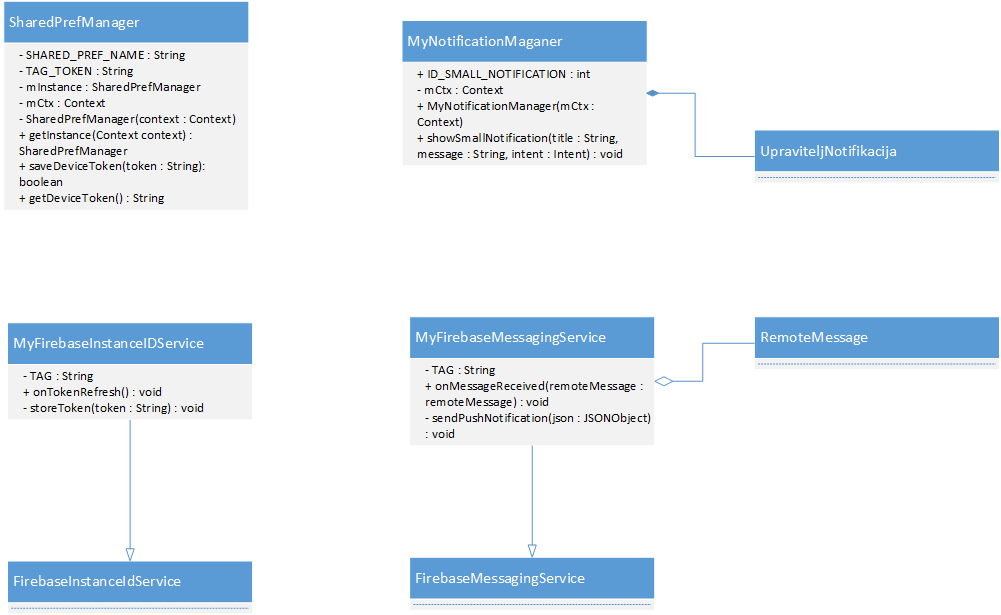
Slika 3: Dijagram klasa za modul app, paket loaders

* 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.



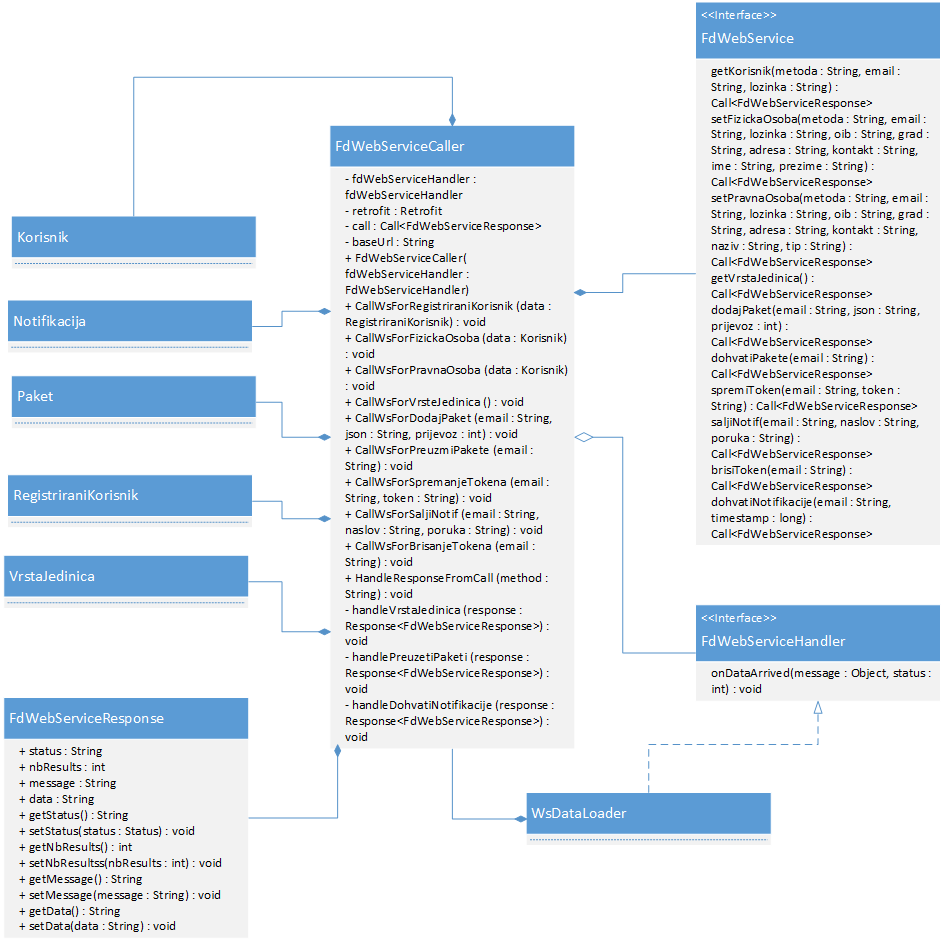
Slika 4: Dijagram klasa za modul app, paket fragments

* 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.



Slika 6: Dijagram klasa za modul webservice

* 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).