UNIVERZITET U NOVOM SADU
FAKULTET TEHNIČKIH NAUKA

UNIVERZITET U NOVOM SADU
FAKULTET TEHNIČKIH NAUKA
NOVI SAD

Aplikacija za praćenje zbrinjavanja životinja

Kandidati:	Lazar Nagulov	SV61/2022
	Filip Tot	SV14/2022
	Ilija Jordanovski	SV73/2022
	Vuk Vićentić	SV45/2022

Predmet: Specifikacija i modeliranje softvera

Novi Sad, jul, 2024.

Sadržaj

Spisak dijagrama

Spisak slučajeva korišćenja

1 Uvod

1.1 Svrha i opis dokumenta

Dokument obuhvata deo informacionog sistema koji se bavi praćenjem zbrinjavanja životinja. Cilj je da omogućimo olakšano pronalaženje novog vlasnika za životinje, ili pronalaženje privremenog smeštaja.

Pored uvodnog poglavlja, dokument sadrži funkcionalne i nefunkcionalne zahteve (poglavlja 2 i 3), opis UML dijagrama koji su nam pomogli u samoj implementaciji (poglavlja 4, 5, 6, 7, 8) i tehnologije koje smo koristili (poglavlje 9). Na kraju se nalazi kratak opis samog rešenja (poglavlje 10).

1.2 Korišćene konvencije

U cilju boljeg razumevanja date specifikacije, termini koji se odnose na procese i poslove aplikacije će biti ukošeni, dok će ključni segmenti poglavlja biti podebljani kako bi naglasili njihovu važnost.

1.3 Reference

2 Funkcionalni zahtevi

Na osnovu dijagrama slučajeva korišćenja [Dijagram 1] primećujemo da postoje četiri uloge u aplikaciji: neregistrovani korisnik (gost), član, volonter i administrator. U nastavku se nalazi opis funkcionalnosti za svaku od uloga.

2.1 Neregistrovani korisnik (gost)

1. Mogućnost registracije unošenjem imena, prezimena, adrese (ulica, broj, grad, država), pola, broja telefona, korisničkog imena i šifre.
2. Prijavljivanje na sistem unosom korisničkog imena i šifre.
3. Pregled odobrenih objava. Gost ne može da interaguje sa objavama dokle god se ne prijavi.

2.2 Član

Poseduje sve mogućnosti kao i *neregistrovani korisnik (gost)*.

1. Dodatno može da kreira objavu (koja mora biti prihvaćena od strane *volontera*) birajući već postojeći tip životinje.
2. Može da komentariše i označi da mu se sviđa odobrena objave kao i da pošalje zahtev za udomljavanje ili privremeni smeštaj. On uvek ima uvid u sve svoje zahteve, gde po mogućnosti može da odustane od njih.
3. Ima mogućnost slanja zahteva za promociju u *volontera*.
4. U svakom trenutku nakon udomljavanja ili obezbeđenog privremenog smeštaja, *član* može da oceni životinju sa brojem i komentarom, kao i da je vrati, ukoliko nije zadovoljan. U slučaju isteka privremenog smeštaja, objava se automatski vraća u stanje *prihvaćena*.
5. Mogućnost doniranja novca za pomoć udruženju. Sve donacije su javno dostupne.

2.3 Volonter

Poseduje većinu mogućnosti kao i *član* - ne može slati zahtev za promociju. Njegova objava je automatski prihvaćena. Da bi *član* postao *volonter*, on mora da bude izglasan od strane drugih *volontera* - mora da ima bar 50% glasnova. Prvog volontera dodaje *administrator*.

1. Zadužen je za prihvatanje (ili odbijanje) objava od strane članova i sakrivanje već prihvaćenih objava.
2. Dodatno je zadužen za unos uplate od strane člana i raspoređivanje sredstava po životinjama.
3. Ima pristup svim zahtevima za udomljavanje i privremen smeštaj, gde ih može odobriti ili odbiti. Takođe može da dodaje nove ponude.
4. Može da obriše nepoželjne komentare.

5. Mogućnost glasanja za promovisanje novog *člana* u *volontera*.
6. Može da dodaje, briše i menja podatke o vrsti životinja.

2.4 Administrator

Poseduje sve mogućnosti kao i *gost*.

1. Ažurira informacije o udruženju.
2. Dodaje prvog *volontera*.

3 Nefunkcionalni zahtevi

3.1 Performanse

3.2 Bezbednost

Treba obezbediti zaštitu svih podataka unutar datog sistema od strane neovlašćenog pristupa. Kako bi se obezbedio ovaj zahtev, neophodno je svakoj klasi korisnika dati određena ovlašćenja. Korisnici ne mogu da vide sve podatke drugih korisnika, ali volonteri mogu. Na objavama se pokazuju samo određeni podaci, jer ih i neregistrovani korisnici mogu videti. Šifre korisnika se ne upisuje direktno u bazu.

3.3 Sigurnost

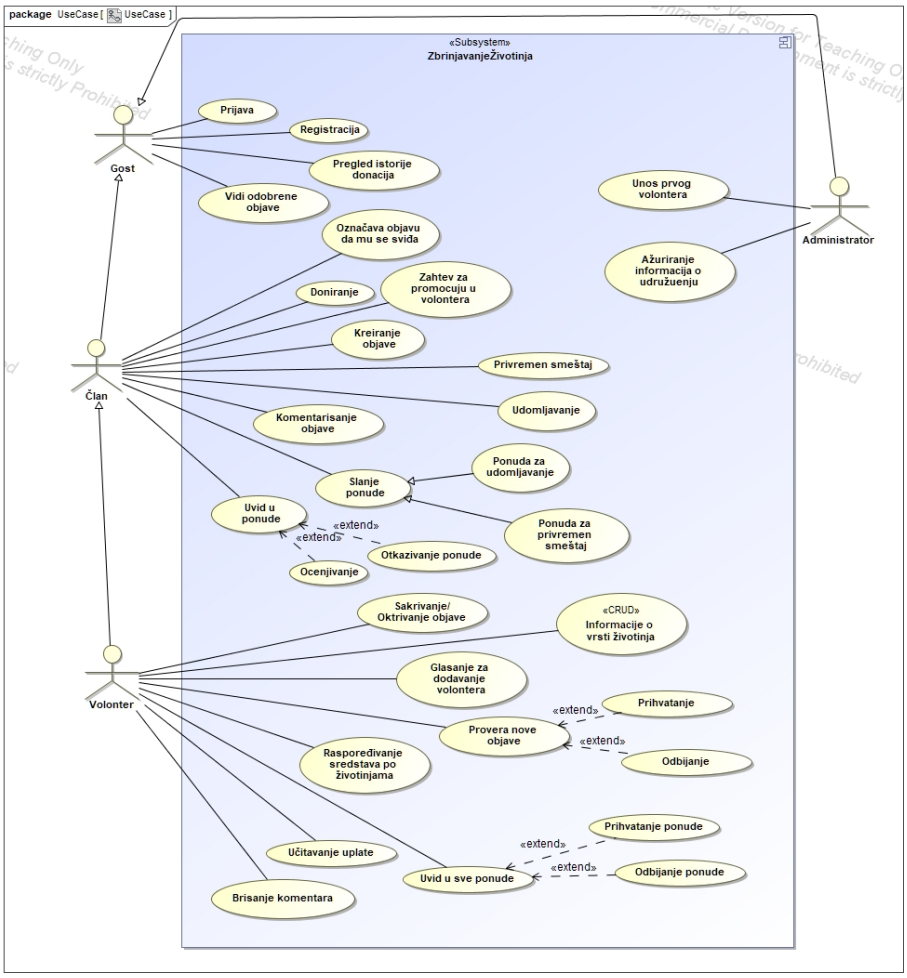
Sigurnost podrazumeva da su podaci osigurani i zaštićeni na takav način da ih je nemoguće neželjeno izmeniti, ukloniti ili zauvek uništiti. U cilju zadovoljavanja datog zahteva, potrebno je vršiti validaciju kako na klijenskoj, tako na serverskoj strani, da bi sugurnost celog sistema bila obezbeđena.

3.4 Pouzdanost, dostupnost i održivost

3.5 Robusnost

4 Dijagram slučajeva korišćenja

U nastavku se nalazi dijagram slučajeva korišćenja [Dijagram 1] na kome su prikazane pravila poslovanja.



Dijagram 1: Dijagram slučajeva korišćenja

4.1 Specifikacija slučajeva korišćenja

U nastavku se nalaze specifikacija korišćenja prikazani na dijagramu slučajeva korišćenja [Dijagram 1].

Identifikator: GI1
Naziv: Registracija
Učesnik: Gost
Opis: Gost se registruje na sistem
Preduslovi: Otvorena registraciona forma
Posledice: Gost ima mogućnost da se prijavi na sistem
Osnovni tok izvršavanja 1. Gost unosi ime, prezime, adresu, datum rođenja, pol i broj telefona. 2. Sistem proverava validnost podataka. 3. Sistem obaveštava gosta da je registracija prošla uspešno. [Alternativni tok A] 4. Kraj scenarija.
Alternativni tok A: Neispravno uneti podaci 1. Sistem obaveštava gosta o greškama u formi. 2. Prikazuje se forma za registraciju. 3. Kraj scenarija.

Slučaj korišćenja 1: Registracija

Identifikator: GI2
Naziv: Vidi odobrene objave
Učesnik: Gost
Opis: Prikaz odobrenih objava gostu
Preduslovi: Nema preduslova
Posledice: Član može da vidi odobrene objave
Osnovni tok izvršavanja 1. Sistem učitava sve odobrene objave. 2. Kraj scenarija.

Slučaj korišćenja 2: Vidi odobrene objave

Identifikator: GI3
Naziv: Pregled istorije donacija
Učesnik: Gost
Opis: Prikaz svih prethodnih donacija
Preduslovi: Pritisnuto je dugme za prikaz istorije donacija
Posledice: član može da vidi sve prethodne donacije
Osnovni tok izvršavanja 1. Sistem učitava istoriju donacija. 2. Kraj scenarija

Slučaj korišćenja 3: Pregled istorije donacija

Identifikator: GI4
Naziv: Prijava
Učesnik: Gost
Opis: Prijava člana na sistem unosom korisničkog imena i šifre
Preduslovi: Otvorena je forma za prijavu
Posledice: Član može da koristi funkcionalnosti sistema
Osnovni tok izvršavanja 1. Gost unosi korisničko ime i šifru. 2. Sistem proverava da li su korisničko ime i šifra ispravni. 3. Sistem otvara odgovarajući meni [Alternativni tok A] 4. Kraj scenarija.
Alternativni tok A: Neispravno korisničko ime ili šifra 1. Sistem obaveštava gosta da su korisničko ime ili šifra neispravni. 2. Gost potvrđuje da je video obaveštenje. 3. Prikazuje se forma za prijavu. 4. Kraj scenarija.

Slučaj korišćenja 4: Prijava

Identifikator: MI1
Naziv: Doniranje
Učesnik: Član
Opis: Prikaz informacija za donacije
Preduslovi: Član je prijavljen na sistem i dugme za donacije je pritisnuto
Posledice: Član ima informacije potrebne za slanje donacije
Osnovni tok izvršavanja 1. Sistem prikazuje informacije o bankovnom računu 2. Kraj scenarija

Slučaj korišćenja 5: Doniranje

Identifikator: MI2
Naziv: Zahtev za promociju u volontera
Učesnik: Član
Opis: Član šalje zahtev za promociju u volontera
Preduslovi: Član je prijavljen na sitem
Posledice: Zahtev je zabeležen u sistemu
Osnovni tok izvršavanja 1. Član podnosi zahtev za promociju. 2. Sistem obaveštava člana da je zahtev uspešno poslat. 3. Sistem trajno čuva zahtev. 4. Kraj scenarija.

Slučaj korišćenja 6: Zahtev za promociju u volontera

Identifikator: MI3
Naziv: Označava objavu da mu se sviđa
Učesnik: Član
Opis: Član označava objavu da mu se sviđa
Preduslovi: Član je prijavljen na sistem
Posledice: Promenjen broj sviđanja na objavi
Osnovni tok izvršavanja 1. Sistem vizuelno obaveštava člana da mu se objava sviđa. 2. Sistem dodaje člana u listu sviđanja za objavu.[Alternativni tok A] 3. Kraj scenarija.
Alternativni tok A: Član je već označio da mu se objava sviđa 1. Sistem uklanja člana iz liste sviđanja za objavu. 2. Kraj scenarija.

Slučaj korišćenja 7: Označava objavu da mu se sviđa

Identifikator: MI4
Naziv: Komentarisanje objave
Učesnik: Član
Opis: Član komentariše objavu
Preduslovi: Član je prijavljen na sistem
Posledice: Dodat novi komentar na objavu
Osnovni tok izvršavanja 1. Član unosi komentar. 2. Sistem dodaje komentar na objavu. 3. Kraj scenarija.

Slučaj korišćenja 8: Komentarisanje objave

Identifikator: MI5
Naziv: Uvid u ponude
Učesnik: Član
Opis: Član ima uvid u sve njehgve ponude. EP1 Član otkazuje ponudu. EP2 Član ostavlja ocenu
Preduslovi: Član je prijavljen na sistem. Pritisnuto je dugme za prikaz ponuda
Posledice: Članova odluka se trajno čuva
Osnovni tok izvršavanja 1. Sistem prikazuje sve njegove ponude. 2. [Tačka proširenja: EP1 Otkazivanje ponude] [Tačka proširenja: EP2 Ocenjivanje] 3. Kraj scenarija.

Slučaj korišćenja 9: Uvid u ponude

Identifikator: EP1
Naziv: Otkazivanje ponude
Učesnik: Član
Opis: Otkazivanje članove ponude
Preduslovi: Otvoren je prikaz članovih ponuda
Posledice: Članova odluka se trajno čuva
Osnovni tok izvršavanja 1. Član aktivira brisanje ponude. 2. Sistem obaveštava člana o posledicama otkazivanja. 3. Član pritiska dugme otkaži. [Alternativni tok A] 4. Sistem briše ponudu iz sistema i osvežava meni sa ponudama. 5. Kraj scenarija.
Alternativni tok A: Član odustaje od brisanja 1. Sistem osvežava meni sa ponudama. 2. Kraj scenarija.

Slučaj korišćenja 10: Otkazivanje ponude

Identifikator: EP2
Naziv: Ocenjivanje
Učesnik: Član
Opis: Ocenjivanje ponude
Preduslovi: Otvoren je prikaz članovih ponuda
Posledice: Članova odluka se trajno čuva
Osnovni tok izvršavanja 1. Član aktivira ocenjivanje ponude. 2. Član unosi ocenu i komentar. 3. Sistem čuva recenziju i osvežava meni sa ponudama. 4. Kraj scenarija.

Slučaj korišćenja 11: Ocenjivanje

Identifikator: MI6
Naziv: Kreiranje objave
Učesnik: Član
Opis: Član kreira ponudu
Preduslovi: Član je prijavljen na sistem
Posledice: Nova objava je dodata u sistem
Osnovni tok izvršavanja 1. Član aktivira kreiranje objave. 2. Član popunjava formu odgovarajućim podacima. 3. Sistem kreira objavu. 4. Sistem inicijalizuje objavu. 5. Kraj scenarija.

Slučaj korišćenja 12: Kreiranje objave

Identifikator: MI7
Naziv: Privremen smeštaj
Učesnik: Član
Opis: Član pruža privremen smeštaj
Preduslovi: Član je prijavljen na sistem i poslao je ponudu za privremen smeštaj koja je prihvaćena
Posledice: Životinja je dobila privremen smeštaj
Osnovni tok izvršavanja 1. Član pregleda obaveštenje da mu je ponuda za privremen smeštaj prihvaćena. 2. Član pruža životinji privremen smeštaj. 3. Kraj scenarija.

Slučaj korišćenja 13: Privremen smeštaj

Identifikator: MI8
Naziv: Udomljavanje
Učesnik: Član
Opis: Član udomljava životinju
Preduslovi: Član je prijavljen na sistem i poslao je ponudu za udomljavanje koja je prihvaćena
Posledice: Životinja je udomljena
Osnovni tok izvršavanja 1. Član pregleda obaveštenje da mu je ponuda za udomljavanje prihvaćena. 2. Član udomljava životinju. 3. Kraj scenarija.

Slučaj korišćenja 14: Udomljavanje

Identifikator: MI9
Naziv: Slanje ponude
Učesnik: Član
Opis: Član šalje ponudu. EP3 Član šalje ponudu za privremen smeštaj. EP4 Član šalje ponudu za udomljavanje
Preduslovi: Član je prijavljen na sistem i otvoren je prikaz objava
Posledice: Poslata ponuda je zabeležena u sistemu
Osnovni tok izvršavanja 1. [Tačka proširenja EP3 Ponuda za privremen smeštaj. EP4 Ponuda za udomljavanje] 2. Kraj scenarija.

Slučaj korišćenja 15: Slanje ponude

Identifikator: EP3
Naziv: Ponuda za privremen smeštaj
Učesnik: Član
Opis: Član šalje ponudu za privremen smeštaj
Preduslovi: Član je prijavljen na sistem i otvoren je prikaz objava
Posledice: Poslata ponuda je zabeležena u sistemu
Osnovni tok izvršavanja 1. Član bira slanje ponude za privremen smeštaj. 2. Sistem kreira ponudu. 3. Sistem obaveštava korisnika o uspešno poslatoj ponudu. 4. Kraj scenarija.

Slučaj korišćenja 16: Ponuda za privremen smeštaj

Identifikator: EP4
Naziv: Ponuda za udomljavanje
Učesnik: Član
Opis: Član šalje ponudu za udomljavanje
Preduslovi: Član je prijavljen na sistem i otvoren je prikaz objava
Posledice: Poslata ponuda je zabeležena u sistemu
Osnovni tok izvršavanja 1. Član bira slanje ponude za udomljavanje. 2. Sistem kreira ponudu. 3. Sistem obaveštava korisnika o uspešno poslatoj ponudu. 4. Kraj scenarija.

Slučaj korišćenja 17: Ponuda za udomljavanje

Identifikator: VI1
Naziv: Sakrivanje/Otkrivanje objave
Učesnik: Volonter
Opis: Volonter sakriva ili otkriva objavu
Preduslovi: Volonter je prijavljen na sistem i otvoren je prikaz objava
Posledice: Stanje objave je promenjeno
Osnovni tok izvršavanja 1. Volonter pritiska dugme za sakrivanje/otkrivanje objave. 2. Sistem menja stanje objave na Sakrivena. [Alternativni tok A] 3. Sistem menja prikaz objave. 4. Kraj scenarija.
Alternativni tok A: Objava već ima stanje Sakrivena 1. Sistem menja stanje objave na Prihvaćena. 2. Sistem menja prikaz objave. 3. Kraj scenarija.

Slučaj korišćenja 18: Sakrivanje/Otkrivanje objave

Identifikator: VI2
Naziv: Glasanje za dodavanje volontera
Učesnik: Volonter
Opis: Volonter glasa za dodavanje volontera
Preduslovi: Volonter je prijavljen na sistem, otvoren je prikaz zahteva na čekanju i korisnik još uvek nije glasao na dati zahtev
Posledice: Volonterov glas je zabeležen
Osnovni tok izvršavanja 1. Volonter pritiska dugme za glasanje za ili protiv. 2. Sistem beleži volonterov glas i ažurira zahtev. 3. Sistem osvežava listu zahteva. 4. Kraj scenarija.

Slučaj korišćenja 19: Glasanje za dodavanje volontera

Identifikator: VI3
Naziv: Provera nove objave
Učesnik: Volonter
Opis: Volonter proverava nove objave
Preduslovi: Volonter je prijavljen na sistem
Posledice: Stanje objave je promenjeno
Osnovni tok izvršavanja 1. Sistem učitava sve nove objave. 2. [Tačka proširenja EP5 Prihvatanje] [Tačka proširenja EP6 Odbijanje] 3. Kraj scenarija.

Slučaj korišćenja 20: Provera nove objave

Identifikator: EP5	
Naziv: Prihvatanje	
Učesnik: Volonter	
Opis: Volonter prihvata novu objavu	
Preduslovi:	Volonter je prijavljen na sistem i odabrana je objava sa stanjem NaČekanju
Posledice: Stanje objave je promenjeno	
Osnovni tok izvršavanja 1. Volonter pritiska dugme prihvatanja objave. 2. Sistem menja stanje objave na Prihvaćena. 3. Sistem menja prikaz objave. 4. Kraj scenarija.	

Slučaj korišćenja 21: Prihvatanje

Identifikator: EP6
Naziv: Odbijanje
Učesnik: Volonter
Opis: Volonter odbija novu objavu
Preduslovi: Volonter je prijavljen na sistem i odabrana je objava sa stanjem NaČekanju
Posledice: Stanje objave je promenjeno
Osnovni tok izvršavanja 1. Volonter pritiska dugme odbijanja objave. 2. Sistem menja stanje objave na Odbijena. 3. Sistem menja prikaz objave. 4. Kraj scenarija.

Slučaj korišćenja 22: Odbijanje

Identifikator: VI4
Naziv: Raspoređivanje sredstava po životinjama
Učesnik: Volonter
Opis: Volonter raspoređuje sredstva udruženja
Preduslovi: Volonter je prijavljen na sistem
Posledice: Raspored sredstava je ažuriran
Osnovni tok izvršavanja 1. Sistem učitava listu životinja. 2. Volonter menja vrednosti dodeljenjih sredstava za životnju. 3. Sistem čuva promenu rasporeda sredstava. 4. Kraj scenarija.

Slučaj korišćenja 23: Raspoređivanje sredstava po životinjama

Identifikator: VI5
Naziv: Učitavanje uplate
Učesnik: Volonter
Opis: Volonter učitava uplatu
Preduslovi: Volonter je prijavljen na sistem
Posledice: Lista donacije je ažurirana
Osnovni tok izvršavanja 1. Volonter pritiska dugme za učitavanje uplate. 2. Volonter bira datoteku sa sadržajem uplate. 3. Sistem ažurira listu donacija. [Alternativni tok A] 4. Kraj scenarija.
Alternativni tok A: Uplata je neispravna 1. Lista donacija ostaje nepromenjena. 2. Kraj scenarija.

Slučaj korišćenja 24: Učitavanje uplate

Identifikator: VI6
Naziv: Brisanje komentara
Učesnik: Volonter
Opis: Volonter briše komentar
Preduslovi: Volonter je prijavljen na sistem i otvoren je prikaz objava
Posledice: Komentar je obrisan
Osnovni tok izvršavanja 1. Volonter pritiska dugme za brisanje komentara. 2. Sistem briše komentar. 3. Sistem ažurira objavu. 4. Sistem osvežava prikaz objava. 5. Kraj scenarija.

Slučaj korišćenja 25: Brisanje komentara

Identifikator: VI7
Naziv: Uvid u sve ponude
Učesnik: Volonter
Opis: Volonter dobija uvid u sve ponude
Preduslovi: Volonter je prijavljen na sistem
Posledice: Ponude su ažurirane
Osnovni tok izvršavanja 1. Sistem učitava listu ponuda. 2. [Tačka proširenja EP7 Prihvatanje ponude] [Tačka proširenja EP8 Odbijanje ponude] 3. Kraj scenarija.

Slučaj korišćenja 26: Uvid u sve ponude

Identifikator: EP7
Naziv: Prihvatanje ponude
Učesnik: Volonter
Opis: Volonter prihvata ponudu
Preduslovi: Volonter je prijavljen na sistem i otvoren je prikaz ponuda
Posledice: Promenjeno je stanje objave
Osnovni tok izvršavanja 1. Volonter pritiska dugme za prihvatanje ponude. 2. Sistem šalje pošaljiocu ponude obaveštenje o uspehu i broj autora objave. 3. Sistem šalje autoru objave obaveštenje o uspehu i broj pošaljioca ponude. 4. Sistem menja stanje objave. 5. Sistem briše ponudu. 6. Kraj scenarija.

Slučaj korišćenja 27: Prihvatanje ponude

Identifikator: EP8
Naziv: Odbijanje ponude
Učesnik: Volonter
Opis: Volonter odbija ponudu
Preduslovi: Volonter je prijavljen na sistem i otvoren je prikaz ponuda
Posledice: Promenjeno je stanje objave
Osnovni tok izvršavanja 1. Volonter pritiska dugme za odbijanje ponude. 2. Sistem šalje pošaljiocu ponude obaveštenje o neuspehu. 3. Sistem briše ponudu. 4. Kraj scenarija.

Slučaj korišćenja 28: Odbijanje ponude

Identifikator: AI1
Naziv: Unos prvog volontera
Učesnik: Administrator
Opis: Administrator unosi prvog volontera
Preduslovi: Administrator je prijavljen na sistem
Posledice: Volonter je dodat
Osnovni tok izvršavanja 1. Administrator popunjava formu za dodavanje volontera. 2. Sistem dodaje volontera u sistem. 3. Kraj scenarija.

Slučaj korišćenja 29: Unos prvog volontera

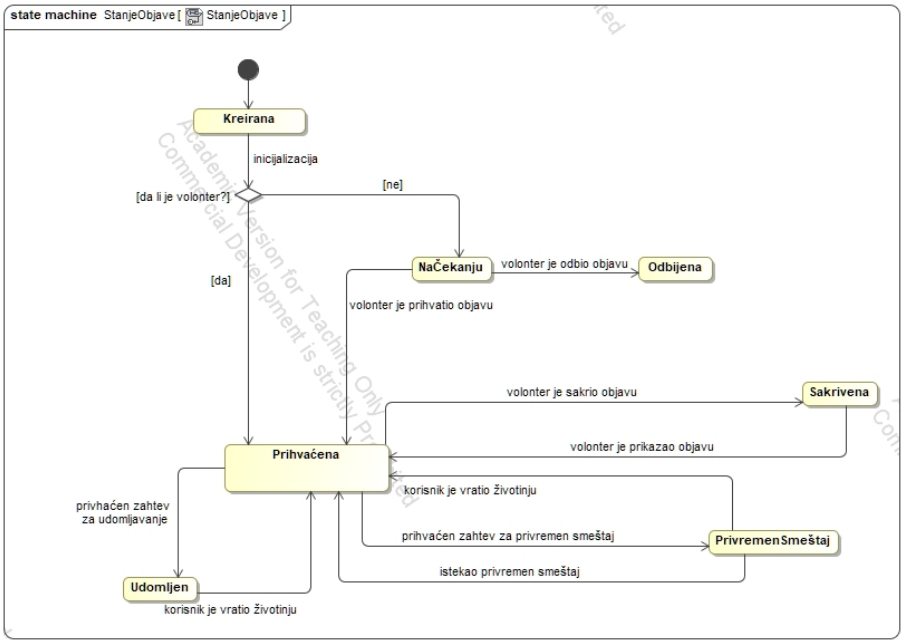
Identifikator: AI2
Naziv: Ažuriranje informacija o udruženju
Učesnik: Administrator
Opis: Administrator ažurira informacije o udruženju
Preduslovi: Administrator je prijavljen na sistem
Posledice: Informacije o udruženju su ažurirane
Osnovni tok izvršavanja 1. Sistem učitava informacije u udruženju. 2. Administrator menja željene informacije o udruženju. 3. Sistem beleži promene. 4. Kraj scenarija.

Slučaj korišćenja 30: Ažuriranje informacija o udruženju

5 Dijagram prelaza stanja za objave

Svaka objava može da se nalazi u 7 različitih stanja: *kreirana*, *na čekanju*, *odbijena*, *prihvaćena*, *sakrivena*, *udomljen* i *privremen smeštaj*. Stanja *prihvaćena*, *udomljena* i *privremen smeštaj* su vidljivi za sve korisnike, dok *volonter* može da vidi objave u stanju *na čekanju* i *sakrivena*, koja su posebno obeležena.

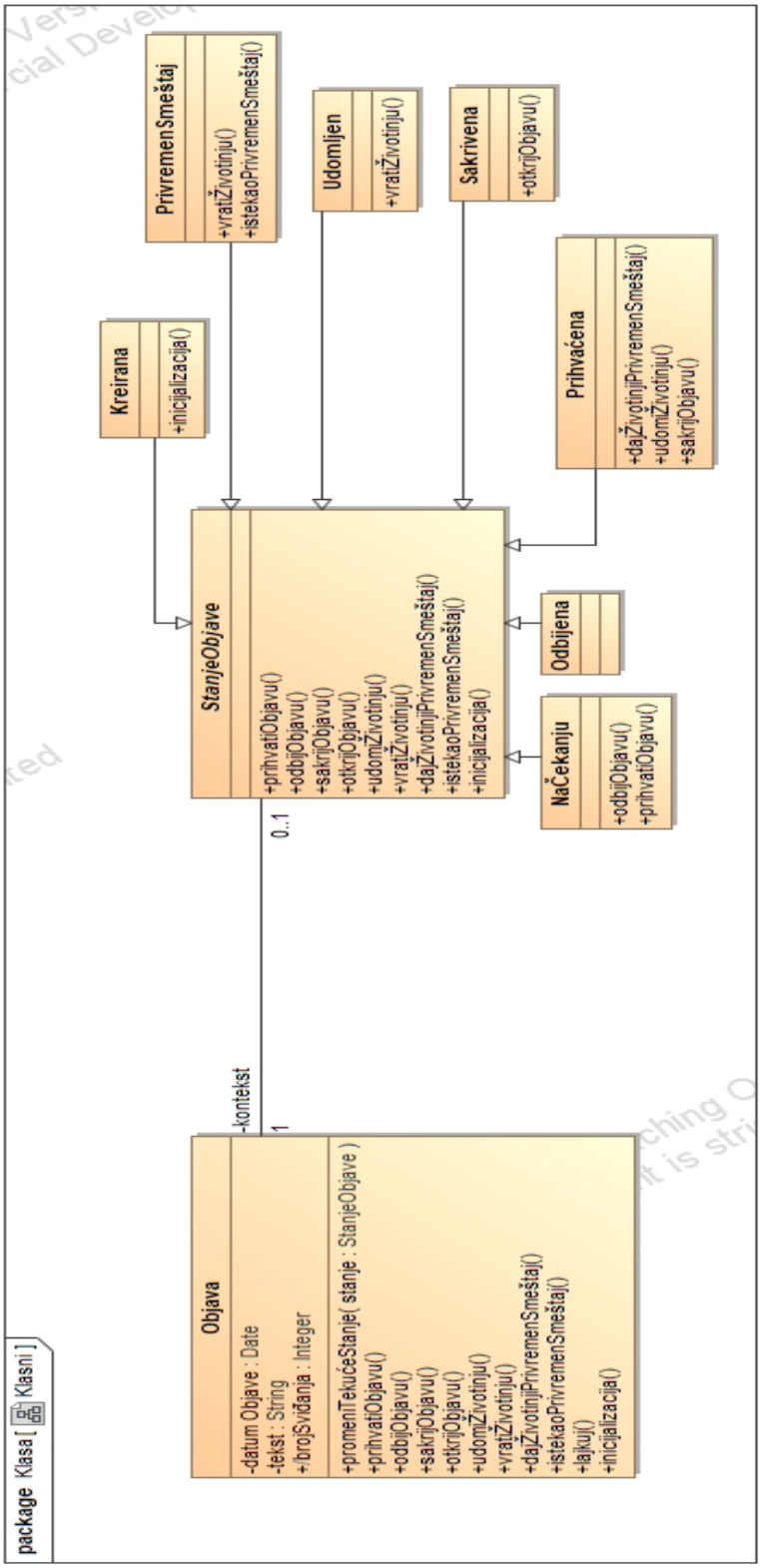
U nastavku je navedeno na koji način objava dolazi do određenog stanja, kao i dijagram prelaza stanja [Dijagram 2].



Dijagram 2: Dijagram prelaza stanja za objave

1. **Kreirana**: inicijalno stanje, kada se kreira objekat *Objava*.
2. **Na čekanju**: početno stanje ako je objava kreirana od strane *člana*.
3. **Odbijena**: *volonter* odbio objavu kreiranu od strane člana
4. **Prihvaćena**: početno stanje ako je objava kreirana od strane *volontera*; *volonter* je prihvatio objavu kreiranu od strane člana; član koji je udomio životinju ili joj pružio privremen smeštaj je odlučio da je vrati; privremen smeštaj je istekao; *volonter* je otkrio prethodno sakrivenu objavu
5. **Sakrivena**: *volonter* sakriva objavu
6. **Udomljen**: prihvaćen je zahtev za udomljavanje
7. **Privremen smeštaj**: prihvaćen je zahtev za privremen smeštaj

Dijagram prelaza stanja sa slike 2 se može prevesti u **klasni dijagram** [Dijagram 3].



Dijagram 3: Preveden dijagram prelaza stanja

6 Klasni dijagram

Klasni dijagram je najvažniji dijagram za implementaciju. Zbog njegova važnosti, svaki deo klasnog dijagrama je opisan u nastavku. Radi boljeg razumevanja, dijagram je podeljen na nekoliko segmenata:

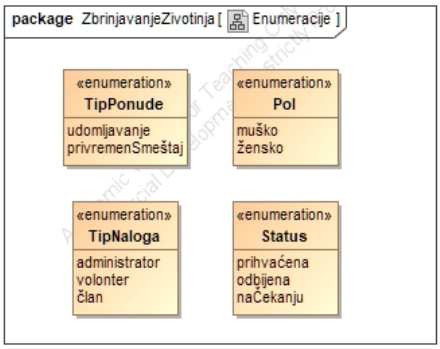
- Enumeracije
- Korisnici
- Zahtevi
- Donacija
- Objave
- Prevod dijagrama prelaza stanja, prethodno na dijagramu 3

Čitav dijagram se može naći na dijagramu 9.

6.1 Enumeracije

Enumeracije se koriste kada imamo tačan skup povezanih konstanti. Na klasnom dijagramu [Dijagram 4] se nalaze 4 enumeracije:

- TipPonude - koristi su klasi *Ponuda* i ima vrednosti *udomljavanje* i *privremenSmeštaj*.
- TipNaloga - koristi se klasi *KorisničkiNalog* i ima vrednosti *administrator*, *član* i *volonter*.
- Pol - koristi se u klasi *KorisničkiNalog* i ima vrednosti *muško* i *žensko*.
- Status - koristi se u klasama *Ponuda* i *Zahtev* i ima vrednosti *prihvaćena*, *odbijena* i *naČekanju*.

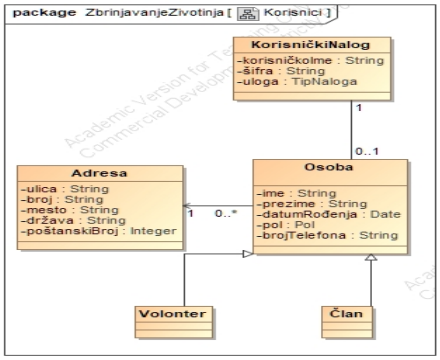


Dijagram 4: Enumeracije u klasnom dijagramu

6.2 Korisnici

Svaka *osoba* ima *KorisničkiNalog* i *Adresa*. Postoje dve naslednice klase *Osoba - Član* i *Volonter*, koje nemaju dodatne atribute, ali zbog njihovih interakcija sa drugim klasama, oni moraju da postoje u sistemu. Primetimo da ne postoje klase Gost i Administrator, jer oni nemaju dodatne interakcije koje se razlikuju od prethodnih klasa. Svi *koirnisčki nalozi* se pamte u klasi *PetCentar*, kao i *korisnički nalog* trenutno ulogovane *osobe*.

Korisnici u sistemu su prikazani na dijagramu 5.

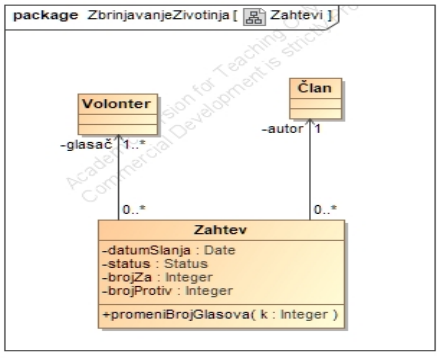


Dijagram 5: Korisnici u klasnom dijagramu

6.3 Zahtevi

Zahteve za promociju može samo da podnese *član*, a da odobri *volonter*. Ova interakcija je detaljnije prikazana na dijagramu sekvence [Dijagram 14]. Svi zahtevi se čuvaju u klasi *PetCentar*.

Zahtevi su prikazani na dijagramu 6.

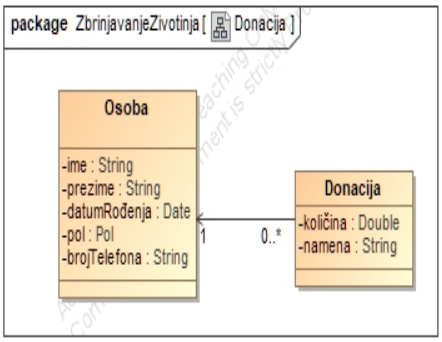


Dijagram 6: Zahtevi u klasnom dijagramu

6.4 Donacije

Prijavljena *osoba* može da donira. Pamti se ko je donirao kao i datum, dok se sve donacije pamte u klasi *PetCentar*.

Donacije su prikazane na dijagramu 7.



Dijagram 7: Donacije u klasnom dijagramu

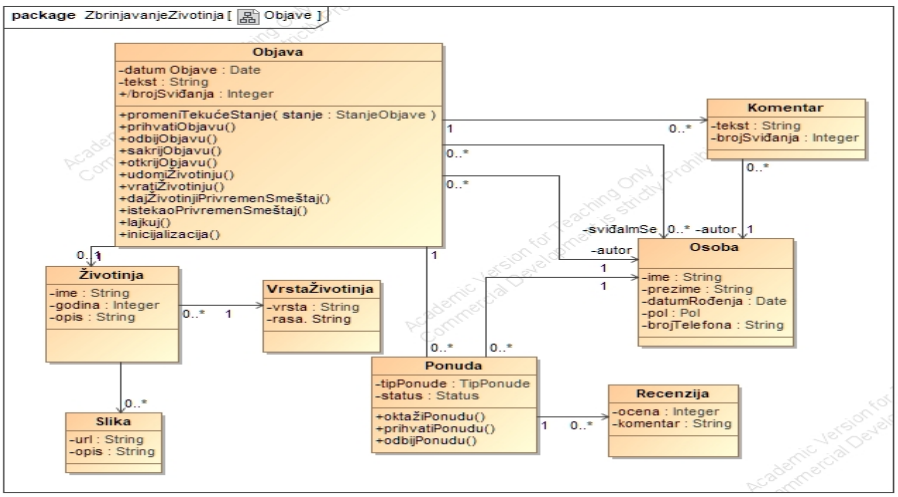
6.5 Objave

Najbitniji deo klasnog dijagrama. Većina interakcija u sistemu su vezani za objave. Svaka objava sadrži *komentare*, *autora*, listu svih *osoba* koji su označili objavu da im se sviđa, *ponude*, *životinju* za koju je vezana.

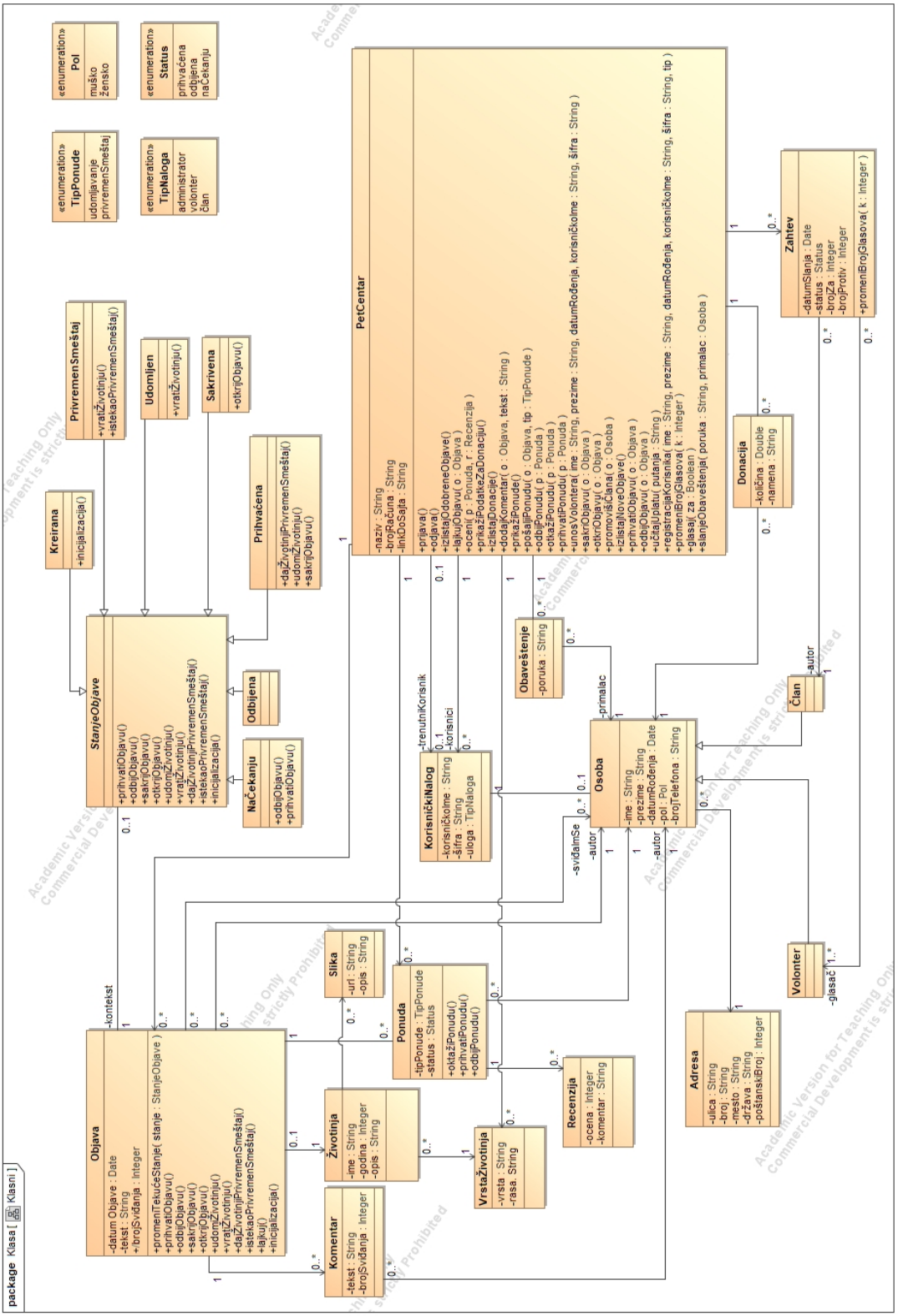
Ponude mogu da imaju *recenziju*.

Pored osnovnih informacija, *životinja* sadrži i *vrstu životinja*. Vrste životinja dodaje *volonter*.

Objave su prikazane na dijagramu 8.



Dijagram 8: Objave u klasnom dijagramu



Dijagram 9: Klasni dijagram

7 Dijagrami aktivnosti

U nastavku se nalaze dijagrami aktivnosti korišćeni za analizu zahteva i specifikaciju dizajna. Opisana su 4 dijagrama aktivnosti:

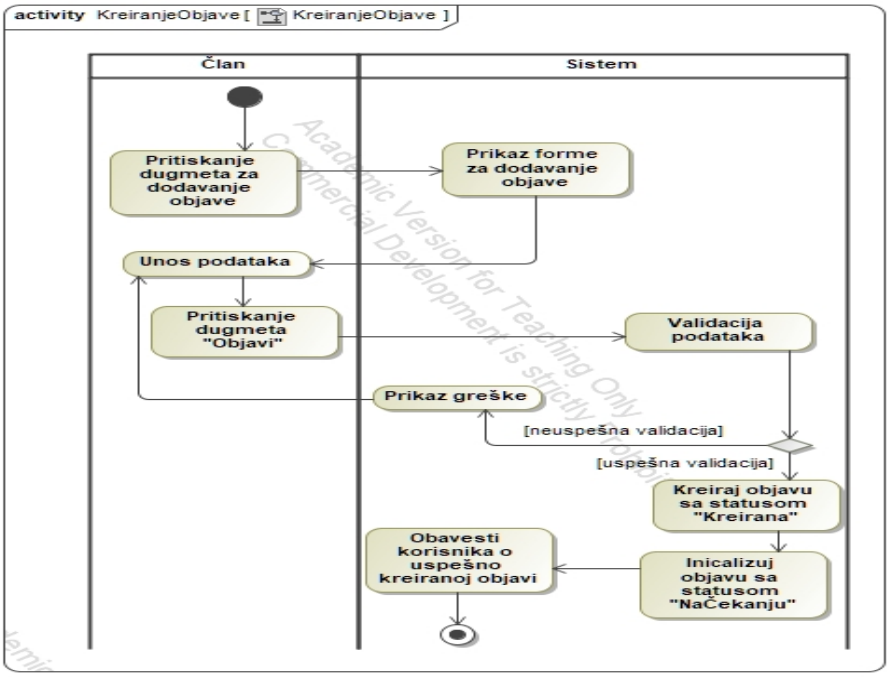
1. **Aktivnost objave**
2. **Slanje ponude**
3. **Učitavanje uplate**
4. **Prijava**

7.1 Aktivnost objave

Dijagram aktivnosti objave predstavlja proces kojim *član* kreira objavu, a *volonter* je prihvata ili odbija. Prikazana je pomoću dva dijagrama aktivnosti: kreiranje objave [Dijagram 10] i pregled objave [Dijagram 11].

Aktivnost kreiranja objave

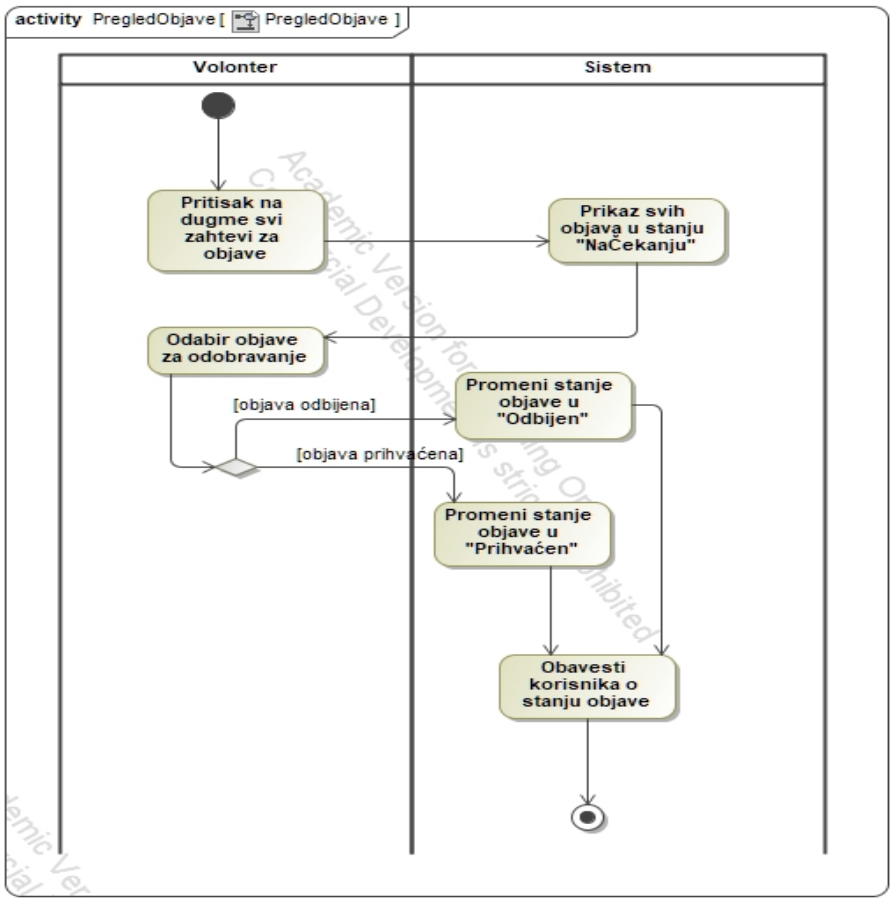
Nakon prijave, članu se nudi mogućnost da otvori formu za objavu. U formu se unose potrebne informacije o životinji u vidu teksta, slike ili videa. Sva polja moraju biti popunjena, u suprotnom se članu prikazuje poruka greške. Nakon uspešne provere unesenih podataka, korsnik se obaveštava da je objava uspešno kreirana. Ovo je prikazano na dijagramu 10.



Dijagram 10: Dijagram aktivnosti za kreiranje objave

Aktivnost pregleda objave

Nakon prijave, *volonter* može da otvori listu svih objava koje još uvek nisu prihvaćene ili odbijene (u stanju *na čekanju*). Pored pregleda, data je mogućnost prihvatanja i odbijanja objave Autor objave se obaveštava o odluci *volontera*, i stanje objave se menja u *prihvaćena*, odnosno *odbijena*. Ovo je prikazano na dijagramu 11.

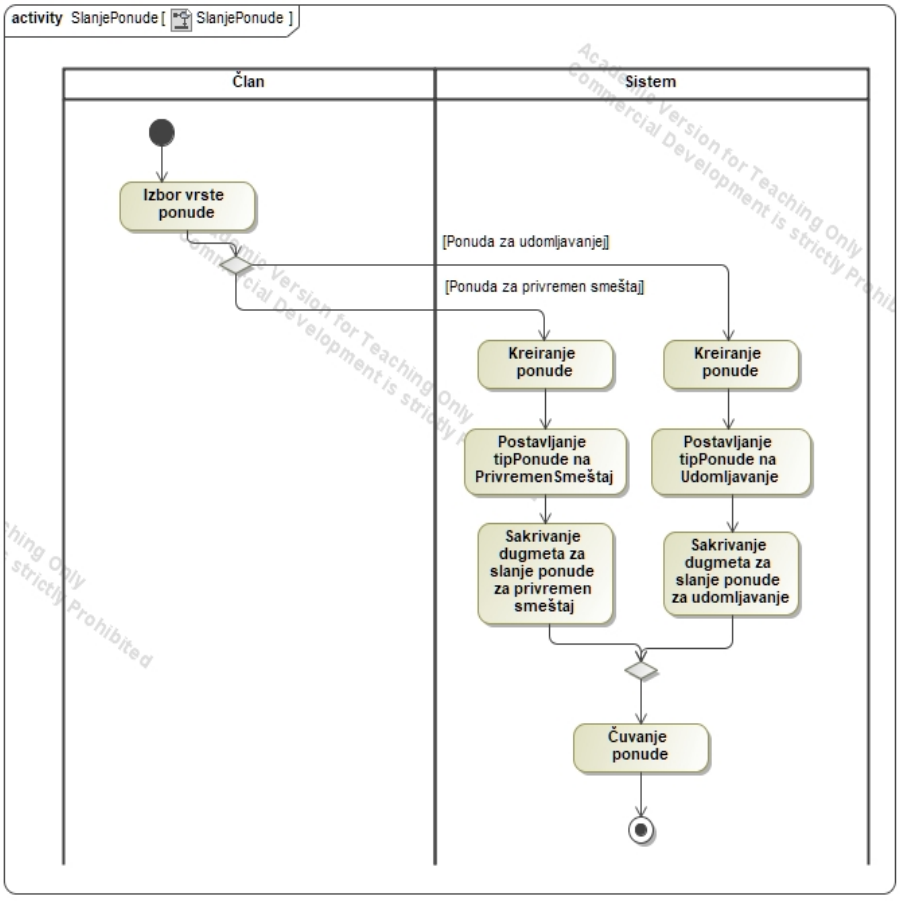


Dijagram 11: Dijagram aktivnosti za pregled objave

7.2 Slanje ponude

Dijagram slanja ponude predstavlja proces slanja ponude za odomljavanje ili privremen smeštaj. *Član* mora da napravi *ponudu*, koju pregleda *volonter*.

Nakon što *član* popuni formu za kreiranje ponude, od zavisnosti od tipa ponude, sistem kreira i trajno čuva ponude. Takođe onemogućava ponovno slanje iste ponude za *člana*. Ovo je prikazano na dijagramu 12.

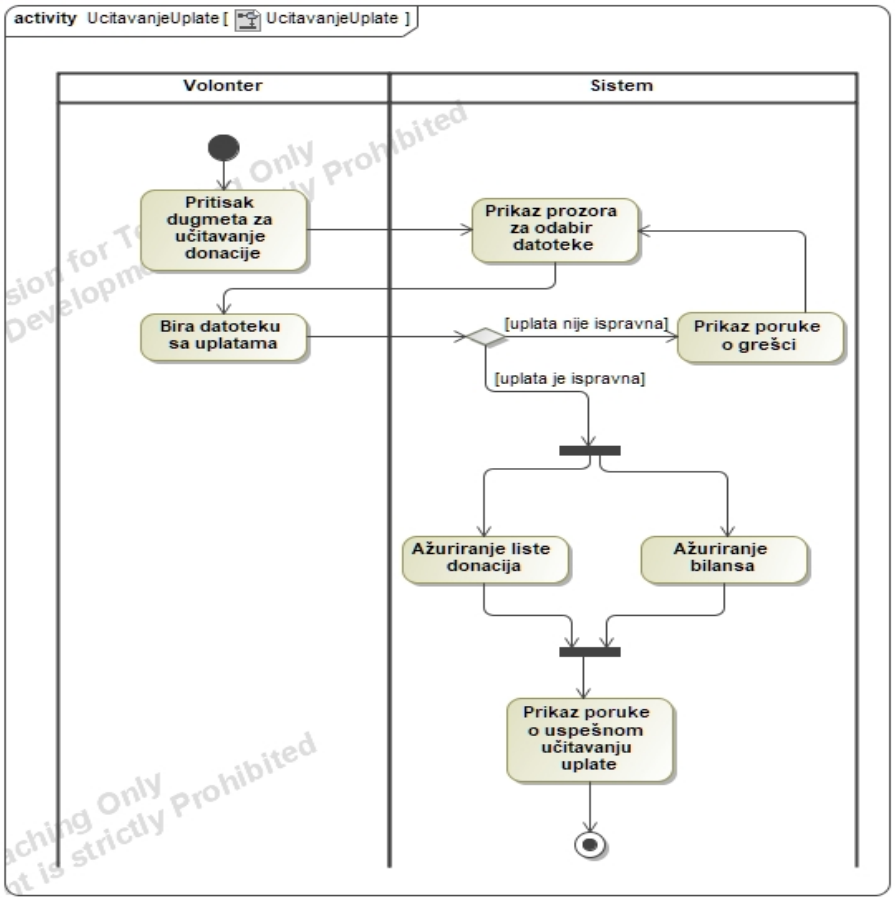


Dijagram 12: Dijagram aktivnosti za slanje ponude

7.3 Učitavanje uplate

Dijagram učitavanja uplate predstavlja proces unosa uplate u sistem. Donacije se vrše preko banke, a *volonter* učitava bankovni dokument za uplatu. Korisnici sistema u svakom trenutku mogu da vide račun na koji se uplaćuje donacija.

Volonter unosi uplatu tako što učita nalog iz banke. Ukoliko dođe do greške prilikom učitavanja, *volonter* se obaveštava i nudi se mogućnost učitavanja drugog fajla. U slučaju uspešnog učitavanja, sistem ažurira listu donacija i bilans. Na kraju, obaveštava volontera o uspešno učitanoj donaciji. Ovo je prikazano na dijagramu 13.



Dijagram 13: Dijagram aktivnosti za učitavanje donacija

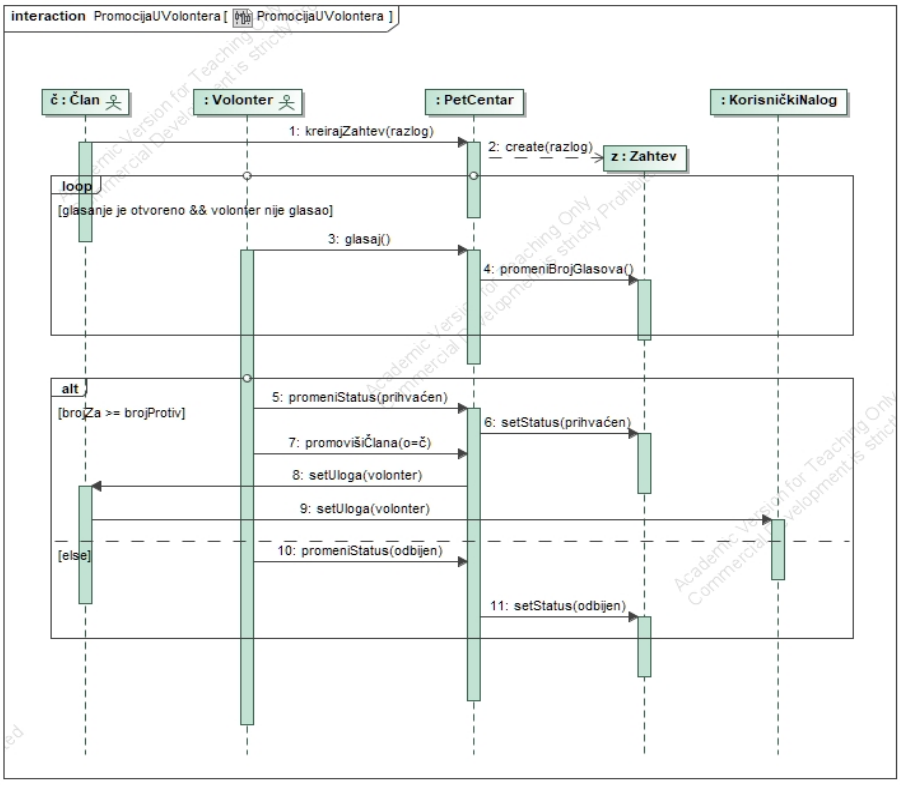
8 Dijagrami sekvence

U nastavku se nalaze dijagrami sekvence korišćeni za analizu interakcija u sistemu. Opisana su 4 dijagrama sekvence:

- **Promocija u volontera**
- **Pregled novih objava**
- **Sakrivanje/ostrkivanje objave**
- **Udomljavanje**

8.1 Promocija u volontera

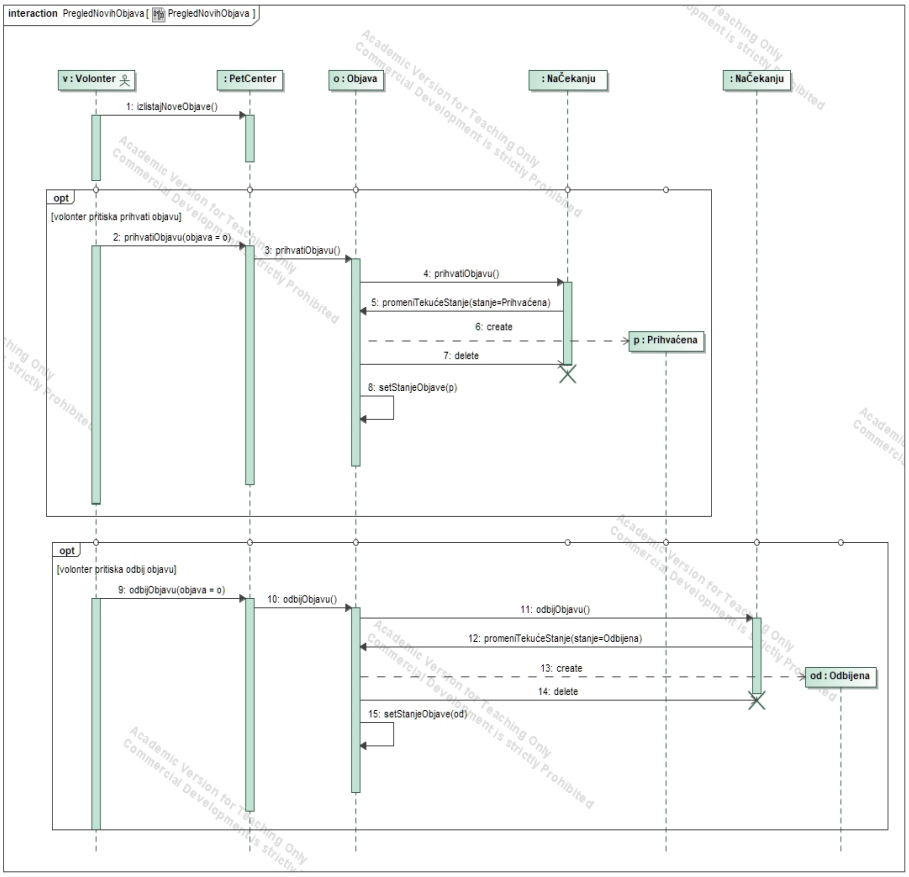
Član može da postane *volonter* tako što pošalje zahtev za volonterstvo. Svaki *volonter* može da glasa samo jednom dokle god traje glasanje. Na kraju glasanja, status zahteva se menja u zavisnosti od broja glasova. Da bi zahtev bio prihvaćen, potrebno je da 50% glasača budu za promociju. Ukoliko je zahtev prihvaćen članu se menja status. Ovo je prikazano na dijagramu 14.



Dijagram 14: Dijagram sekvence za promociju u volontera

8.2 Pregled novih objava

Volonter može da vidi sve objave koje su na čekanju. Nakon prikaza, on može da prihvati ili odbije ponudu. U zavisnosti od odluke, menja se stanje objave i briše prethodno stanje (*NaČekanju*). Ovo je prikazano na dijagramu 15.



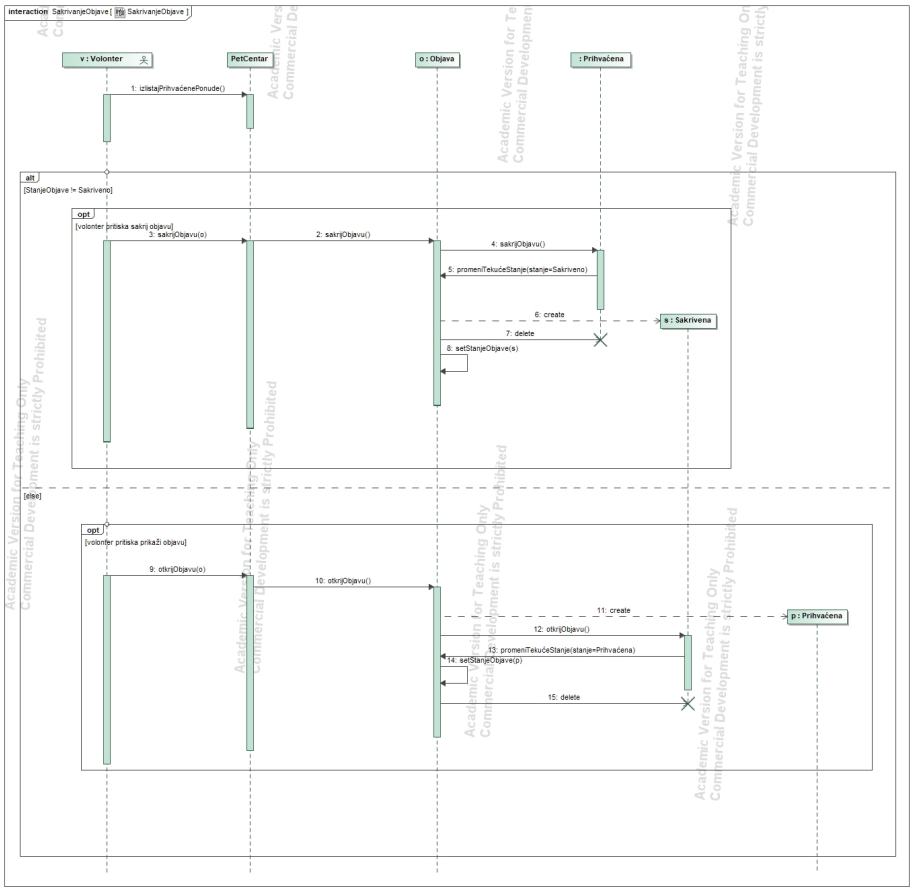
Dijagram 15: Dijagram sekvence za pregled novih objava

8.3 Sakrivanje/ostrkivanje objave

Volonter ima mogućnost sakrivanja objava pritiskom na dugme sakrij. Objava prelazi u stanje *sakrivena* i *članovi* je ne mogu videti, ali je mogu videti drugi *volonteri*. U svakom trenutno, neko od volontera može da otrkije objavu. Ovo je prikazano na dijagramu 16

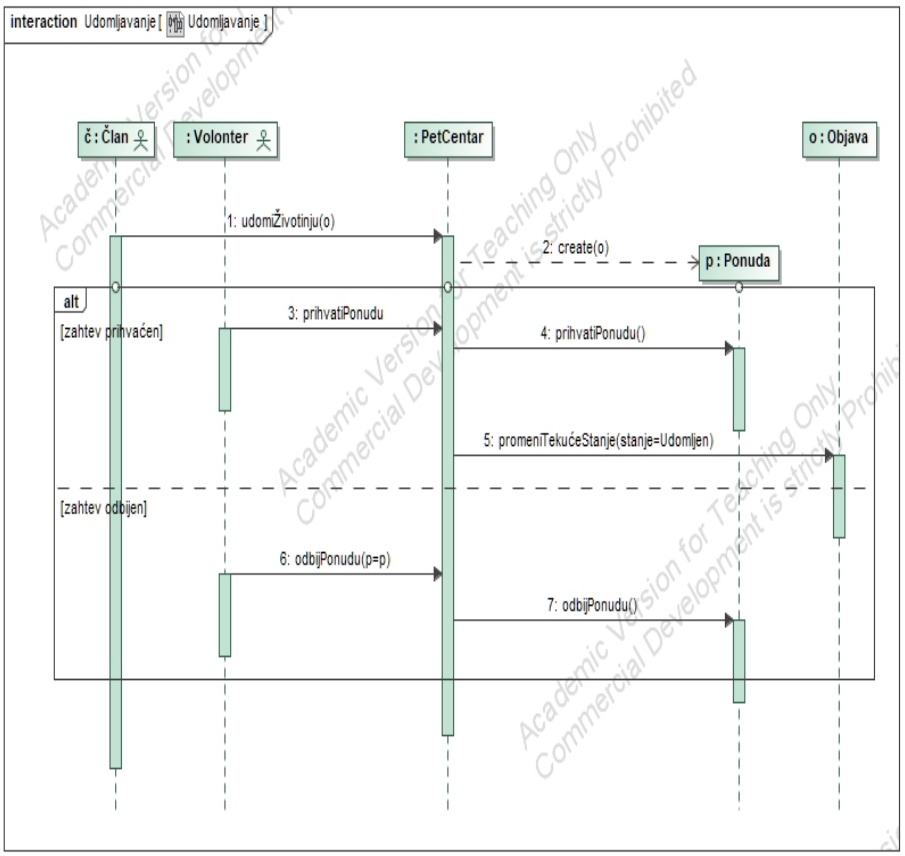
8.4 Udomljavanje

Član može da udomi životinju tako što kreira ponudu [Dijagram 12]. *Volonter* ima uvid u sve ponude i može da prihvati i odbije. Ukoliko je ponude prihvaćena,



Dijagram 16: Dijagram sekvence za sakrivanje/otkrivanje objave

stanje objave se menja u *udomljena*. U suprotnom, stanje objave ostaje isto, ali se menja u ponudi na *odbijena*.



Dijagram 17: Dijagram sekvence za udomljavanje

9 Korišćene tehnologije

U nastavku se nalaze tehnologije korišćena za izradu aplikacije.

9.1 MagicDraw 17.0.3

Korišćen za kreiranje dijagrama. Svi dijagrami se nalaze u *doc/diagrams/diagrams.mdzip* projektu, dok se sve slike dijagrama mogu pronaći u *doc/specification/img*.

9.2 C# .NET 8.0 + WPF

Korišćen za implementaciju aplikacije. Kod se nalazi u *PetCenter* folderu.

9.3 PostgreSQL

Korišćen za bazu podataka, zajedno sa **Entity** radnim okvirom. DDL i DML skripte za kreiranje i popunjavanje baze podataka sa test podacima se nalaze u *scripts* folderu.

10 Implementacija

Implementacija je javno dostupna na githubu:

https://github.com/lazarnagulov/zbrinjavanje-zivotinja

10.1 Primena MVVM šablona

Za razvoj je korišćen **MVVM** (Model-View-ViewModel) arhitektonski obrazac.

Model

Model čine **entiteti**, **servisi** i **repozitorijumi** (domenski sloj).

Entiteti

Entiteti su strukture podataka koje, pored samih podataka, sadrže pomoćne funkcije za upravljanje sa konteinerima.

Na primeru [Listing 1], klasa *Animal* sadrži listu fotografija i pomoćne funkcije za dodavanje, brisanje i dobavljanje readonly liste.

```
public class Animal(AnimalType type, string name, int
    ↪ age, string description)
{
    public AnimalType Type { get; set; } = type;
    public string Name { get; set; } = name;
    public int Age { get; set; } = age;
    public string Description { get; set; } =
        ↪ description;
    private readonly List<Photo> _photos = [];
    public IReadOnlyCollection<Photo> Photos
        => _photos;

    public void AddPhoto(Photo photo)
        => _photos.Add(photo);
    public void RemovePhoto(Photo photo)
        => _photos.Remove(photo);
}
```

Listing 1: Primer entiteta

Servisi

Servisi služe za komunikaciju sa podacima i poslovnom logikom. U njima se nalazi komplikovanija poslovna logika koja može da uzima u obzir više objekata. Koristi interfejse repozitorijuma za pristup podacima u bazi.

```
public class NotificationService(
    ↪ INotificationRepository notificationRepository)
{
    public bool Insert(Notification notification)
```

```
        => notificationRepository.Insert(notification);
    public bool Delete(Notification notification)
        => notificationRepository.Delete(notification);
    public Notification? GetById(Guid id)
        => notificationRepository.GetById(id);
    public List<Notification> GetAll()
        => notificationRepository.GetAll();
    public bool SendNotification(Person recipient,
        ↪ string message)
    {
        var notification = new Notification(recipient,
            ↪  message);
        return Insert(notification);
    }
    public bool DeleteById(Guid id)
    {
        var notification = GetById(id);
        return notification is not null &&
            ↪ notificationRepository.Delete(
            ↪ notification);
    }

    public List<Notification> GetAllIncluded()
        => notificationRepository.GetAllIncluded();

    public List<Notification> GetAllForPerson(Person
        ↪ person)
        => notificationRepository
            .GetAllIncluded()
            .Where(n => n.Recipient.Id == person.Id)
            .ToList();
}
```

Listing 2: Primer servisa

Primer servisa se nalazi na listingu 2. On je zadužen za dobavljanje, kreiranje i brisanje svi obaveštenja, dobavljanje svih objekata koji su u relaciji sa obaveštenjem, kao i dobavljanje svakog obaveštenja za određenu *osobu*.

Repozitorijumi

Repozitorijumi služe za pristup podacima. Najosnovnije operacije nad *entitetima* u bazi podataka su **CRUD** (create, read, update, delete) operacije, koje se nalaze u *ICrud<T>* intefejsu, prikazanom na listingu 3.

```
public interface ICrud<T>
    where T : class
{
    List<T> GetAll();
    T? GetById(Guid id);
    bool Insert(T entity);
    bool Delete(T entity);
```

```
    bool Update(T entity)
}
```

Listing 3: ICrud<T> interfejs

View

View je korisnički interfejs aplikacije. Odgovoran je za prikaz podataka i interakcijom sa korisnikom, ali ne sadrži poslovnu logiku.

Obično se implementira pomoću **XAML**. Podatke dobavlja sa *viewmodela* povezivanjem podataka (eng. data binding).

Aplikacije sadrži 5 glavnih prozora - za uloge u sistemu i početni. Sa početnog prozora se pristupa drugim prozorima prijavom na sistem. U svakom prozoru se smenjuju više *view*-ova. To se postiže **navigacijom**.

Svaki *view* je povezan sa jedniom *viewmodel* klasom, koja sadrži kontekst - podatke koje se koriste na samom *view*-u.

ViewModel

ViewModel je posrednički sloj između korisničkog interfejsa i domenskog sloja.

Za svaki entitet je napravljen **omotač** (eng. wrapper) [Listing 4], kako bi se prilagodilo korisničkom interfejsu. Svaki od svojstava ovih klasa se može povezati (eng. binding) za *xmal* dokument koji predstavlja *view*.

```
public class ViewModelBase : INotifyPropertyChanged
{/*...*/}
public class PhotoViewModel(Photo photo) :
    ↪ ViewModelBase
{
    public Guid Id
    {
        get => _id;
        set => SetField(ref _id, value);
    }

    public string Description
    {
        get => _description;
        set => SetField(ref _description, value);
    }

    public string Url
    {
        get => _url;
        set => SetField(ref _url, value);
    }

    private Guid _id = photo.Id;
    private string _description = photo.Description;
    private string _url = photo.Url;
```

```
}
```

Listing 4: Primer ViewModel omotača

Ključan element u svemu ovome je **INotifyPropertyChanged** interfejs jer omogućava obaveštavanje korisničkog interfejsa o promenama na *viewmodel*-u - **posmatrač** (eng. Observer).

Komande

Bitnu ulogu u odvajanju logike korisničkih akcija od korisničkog interfejsa igra **ICommand** interfejs. Ovim se povećava modularnost i testiranje u aplikaciji. Postoje dve implementacije komandi: **Relay** komande [Listing 5] i **Klasne** komande.

```
public PostListingViewModel(/* ... */)
{
    /* ... */
    LikePostCommand
       = new RelayCommand<PostViewModel>(LikeCommand);
    AddCommentCommand
       = new RelayCommand<PostViewModel>(
          ↪ CommentCommand, CanComment);
}

private static bool CanComment(PostViewModel arg)
    => !string.IsNullOrEmpty(arg.NewComment);

private void CommentCommand(PostViewModel obj)
{
    var comment = new Comment(_authenticationStore.
       ↪ LoggedUser!, obj.NewComment);
    obj.Comments.Add(new CommentViewModel(comment));
    _postService.AddComment(obj.Id, comment);
}
```

Listing 5: Primer *relay* komande za dodavanje komentara

10.2 Perzistencija podataka

Za perzistenciju podataka je korišćena PostgresSQL relaciona baza podataka. U nastavku je opisan način na koji smo kreirali bazu podataka (migracijom) i kreirali validaciju. Na kraju je dat primer funkcionisanja manjeg dela sistema za perzistenciju podataka.

Migracija

Migracija je urađena pomoću **Entity** radnog okvira komandama:

```
dotnet ef migrations add Initial
dotnet ef database update
```

Listing 6: Kreiranje migracije

Alat automatski generiše C# kod koji predstavlja promene u modelu i kako te promene treba primeniti na bazu podataka. Neke detalje je moguće podesi preko koda, poput *unique* ograničenja, prikazanom na listingu 7 ili preko anotacija [Listing 8].

```
modelBuilder.Entity<Account>()
    .HasIndex(account => account.Email)
    .IsUnique();

modelBuilder.Entity<Account>()
    .HasIndex(account => account.Username)
    .IsUnique();
```

Listing 7: Primer postavljanja unique ograničenja

```
[Column("username")]
[Required]
[MaxLength(30)]
public string Username { get; set; }
```

Listing 8: Primer postavljanja anotacija

Konekcija

Da bi smo se uspešno povezali na bazu, moramo uneti parametre konekcije [Listing 9]. Parametri se učitavaju iz **user-secrets** json datoteke, a pristupa im se pomoću klase *Configuration*.

```
public string Host { get; }
public int Port { get; }
public string Username { get; }
public string Password { get; }
public string Database { get; }
```

Listing 9: Parametri konekcije

Primer *user-secret*-a se nalazi na listingu 10.

```
{
    "Database": {
        "Host": "localhost",
        "Port": 5432,
        "DatabaseName": "PetCenter",
        "Username": "postgres",
        "Password": "1234"
    }
}
```

Listing 10: Primer *user-secret*

Konekciju uspostavljamo **UseNpgsql** metodom:

```
optionsBuilder.UseNpgsql(databaseCredentials.
    ↪ ConnectionString);
```

Listing 11: Uspostavljanje konekcije

Validacija

Validacija na nivou baze podataka je pisana u **SQL**-u. Najčešće korišćen vid validacije je **BEFORE INSERT/UPDATE TRIGGER** [Listing 12], koji nam omogućava da, pre samog upisa podataka, proverimo njihovu validnost.

```
CREATE OR REPLACE FUNCTION fn_validate_request()
RETURNS TRIGGER AS $$
DECLARE
        account_type INTEGER;
BEGIN
        SELECT acc_type
        INTO account_type
        FROM request r
        JOIN account a ON r.person_req_author = a.
           ↪ person_id_person
        WHERE r.person_req_author = NEW.
           ↪ person_req_author;

        IF account_type <> 0 THEN
                RAISE EXCEPTION 'Invalid account type
                   ↪ for author, expected 0 but got:
                   ↪ %', account_type;
        END IF;

        RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER trg_validate_request
BEFORE INSERT OR UPDATE ON request
FOR EACH ROW
EXECUTE FUNCTION fn_validate_request();
```

Listing 12: Primer validacije pomoću trigera

Na listingu 12 proveravamo da li je kreator zahteva *član* (vrednost 0).

10.3 Zaštita šifre

Za zaštitu šifre koristimo **SHA256** [Listing 13].

```
public static string Encode(string password)
{
    var sha = SHA256.Create();
    var asBytes = Encoding.UTF8.GetBytes(password);
    var hashed = sha.ComputeHash(asBytes);

    return Convert.ToBase64String(hashed);
}
```

Listing 13: Primer funkcije za zaštitu šifre

10.4 Organizacija foldera

Za organizaciju foldera je korišćen **package by layer** obrazac. Koristimo 6 glavnih foldera:

1. **Core** - aplikativni sloj, sadrži servise, klase koje čuvaju stanja sistema (Stores) i pomoćne klase.
2. **Domain** - domenski sloj, sarži enumeracije, modele i interfejse za repozitorijum.
3. **HostBuilder** - zadužen za **dependency injection**.
4. **Migrations** - sadrži migracije za bazu podataka.
5. **Repository** - sadrži sve repozitorijume i **data context**.
6. **WPF** - sadrži kod za korisnički interfejs - **view** i **viewmodel**.