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 slika

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 [Slika 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 *član*.

1. Dodatno dodaje, briše i menja 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

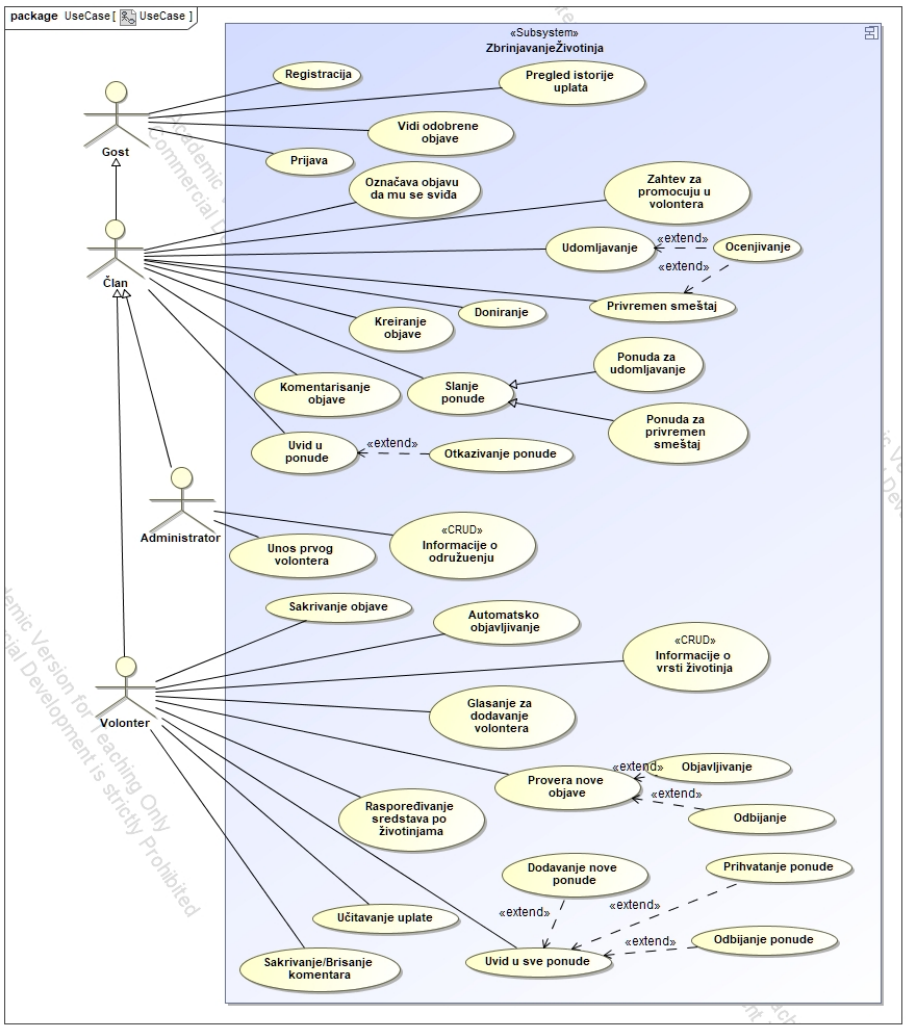
Sigurnost podrazumeva da su podaci osigurani i zaštićeni na takav način da ih je nemoguće neželjeno izmeniti, ukloniti ili zauvek uništiti. U cilju zadovoljavanja datog zahteva, potrebno je vršiti validaciju kako na klijenskoj, tako na serverskoj strani, da bi sugurnost celog sistema bila obezbeđena.

3.4 Pouzdanost, dostupnost i održivost

3.5 Robusnost

4 Dijagram slučajeva korišćenja

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



Slika 1: Dijagram slučajeva korišćenja

4.1 Specifikacija slučajeva korišćenja

U nastavku se nalaze specifikacija korišćenja prikazani na dijagramu slučajeva korišćenja [Slika 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: GI5
Naziv: Doniranje
Učesnik: Član
Opis: Prikaz informacija za donacije
Preduslovi: Član je prijavljen na sistem i dugme za donacije je pritisnuto
Posledice: A: 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: MI1
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 unosi razlog 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: MI2
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: MI3
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: MI4
Naziv: Uvid u ponude
Učesnik: Član
Opis: Član ima uvid u sve njehgve ponude. EP1 Član otkazuje ponudu
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] 3. Kraj scenarija.

Slučaj korišćenja 9: Uvid u ponude

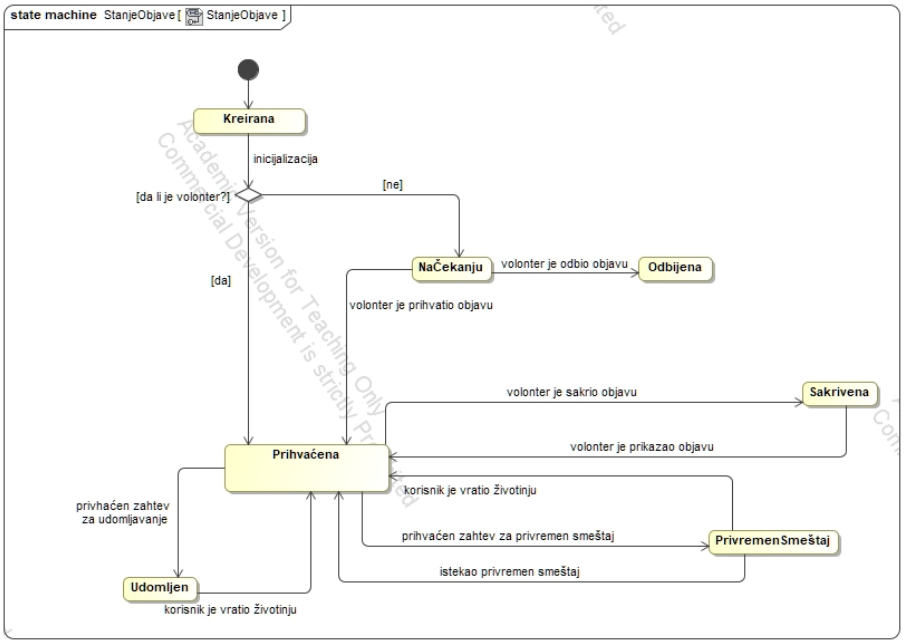
Identifikator: EP1
Naziv: Otkazivanje ponude
Učesnik: Član
Opis: Otkazivanje članove ponude
Preduslovi: Otvoren je prikaz članovih ponuda
Posledice: Članova odluka se trajno čuva
Osnovni tok izvršavanja 1. Član aktivira brisanje ponude. 2. Sistem obaveštava člana o posledicama otkazivanja. 3. Član pritiska 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

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.

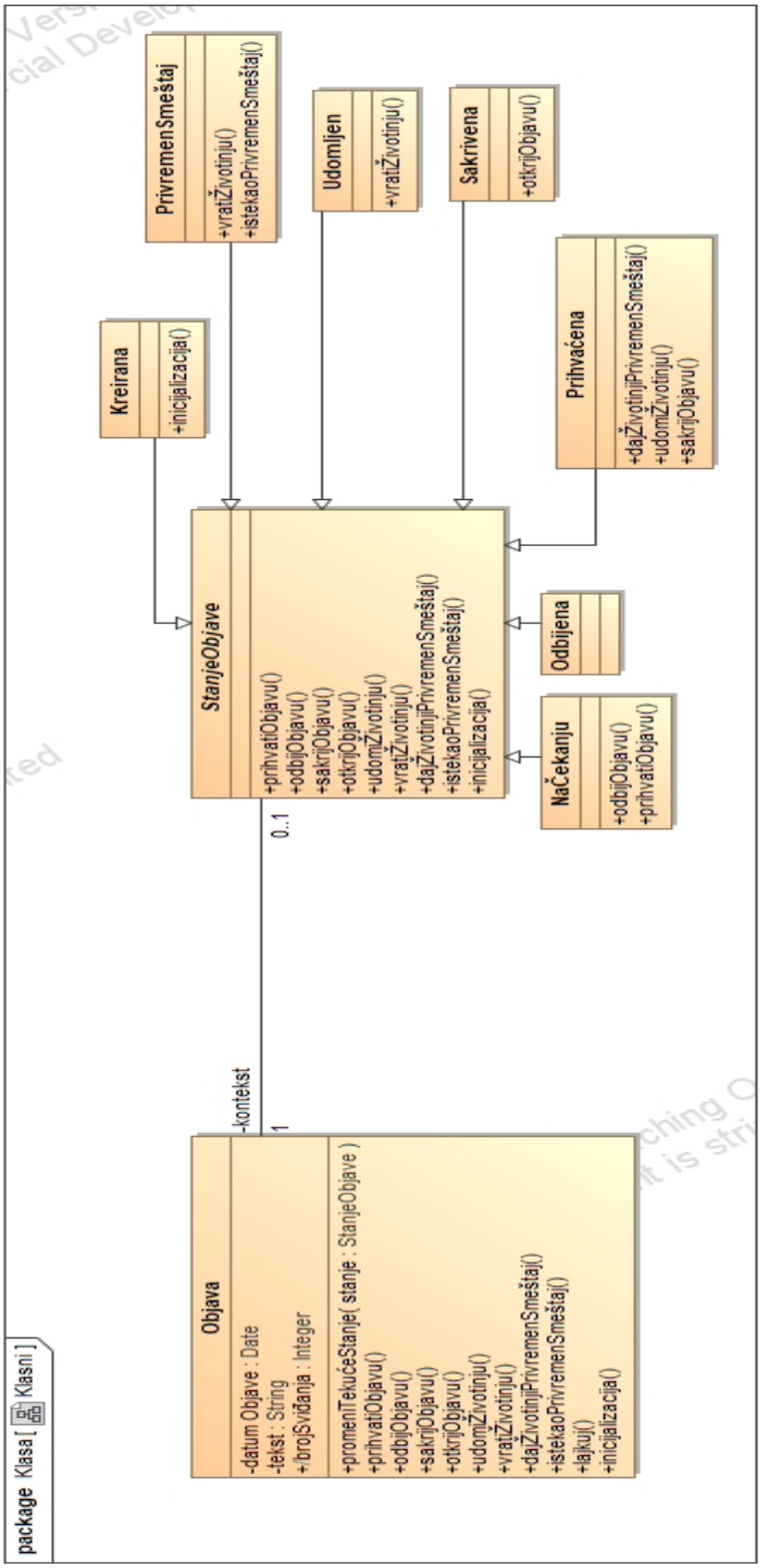
U nastavku je navedeno na koji način objava dolazi do određenog stanja, kao i dijagram prelaza stanja [Slika 2].



Slika 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** koji se nalazi na slici 3.



Slika 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:

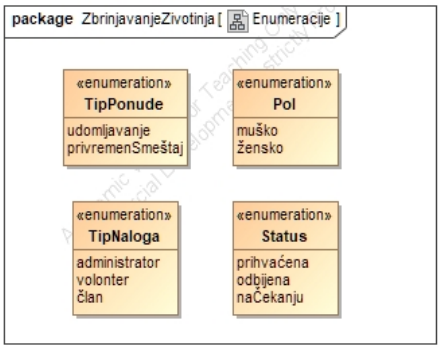
- Enumeracije
- Korisnici
- Zahtevi
- Donacija
- Objave
- Prevod dijagrama prelaza stanja, prethodno na slici 3

Čitav dijagram se može naći na slici 9.

6.1 Enumeracije

Enumeracije se koriste kada imamo tačan skup povezanih konstanti. Na klasnom dijagramu [Slika 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*.

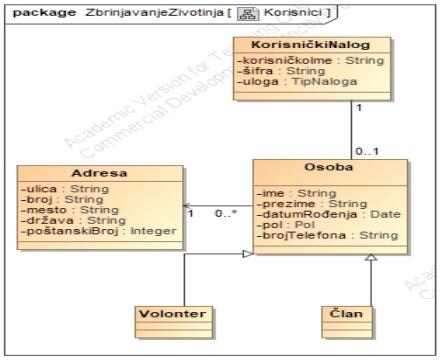


Slika 4: Enumeracije u klasnom dijagramu

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 slici 5.

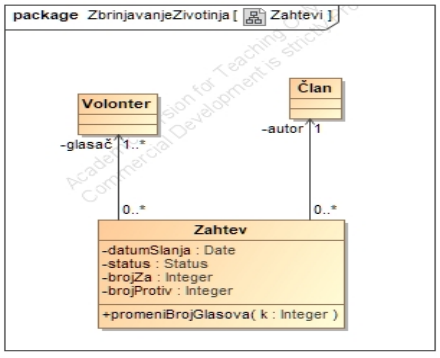


Slika 5: Korisnici u klasnom dijagramu

6.3 Zahtevi

Zahteve za promociju može samo da podnese *član*, a da odobri *volonter*. Ova interakcija je detaljnije prikazana na dijagramu sekvence [Slika 12]. Svi zahtevi se čuvaju u klasi *PetCentar*.

Zahtevi su prikazani na slici 6.

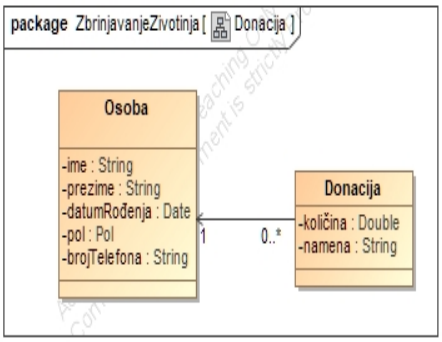


Slika 6: Zahtevi u klasnom dijagramu

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 slici 7.



Slika 7: Donacije u klasnom dijagramu

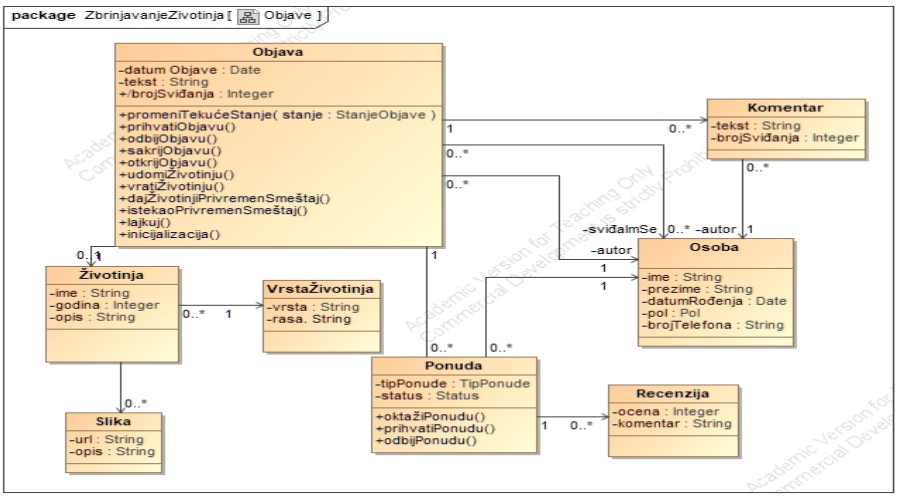
6.5 Objave

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.

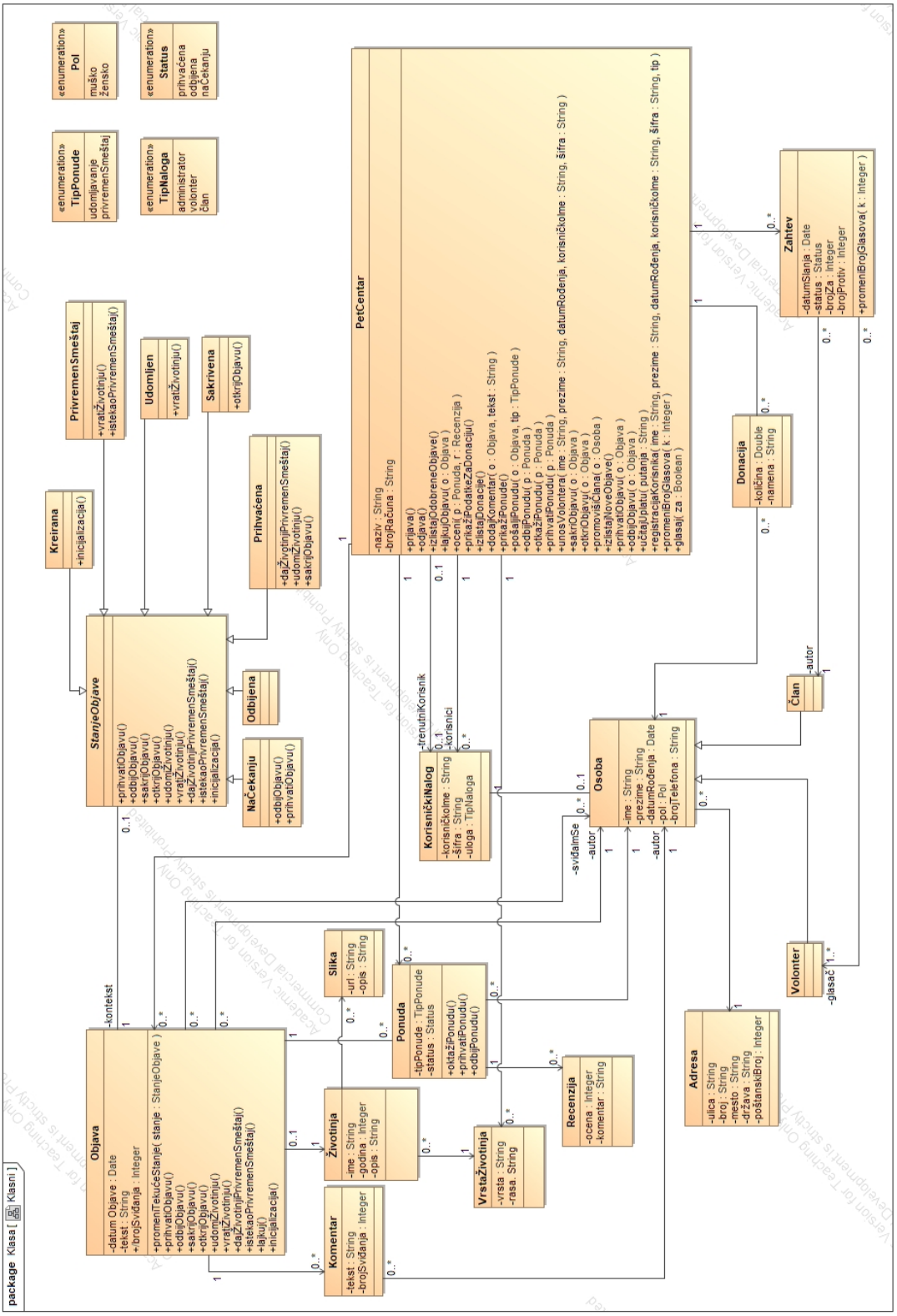
Ponude mogu da imaju *recenziju*.

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

Objave su prikazane na slici 8.



Slika 8: Objave u klasnom dijagramu



Slika 9: Klasni dijagram

7 Dijagrami aktivnosti

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

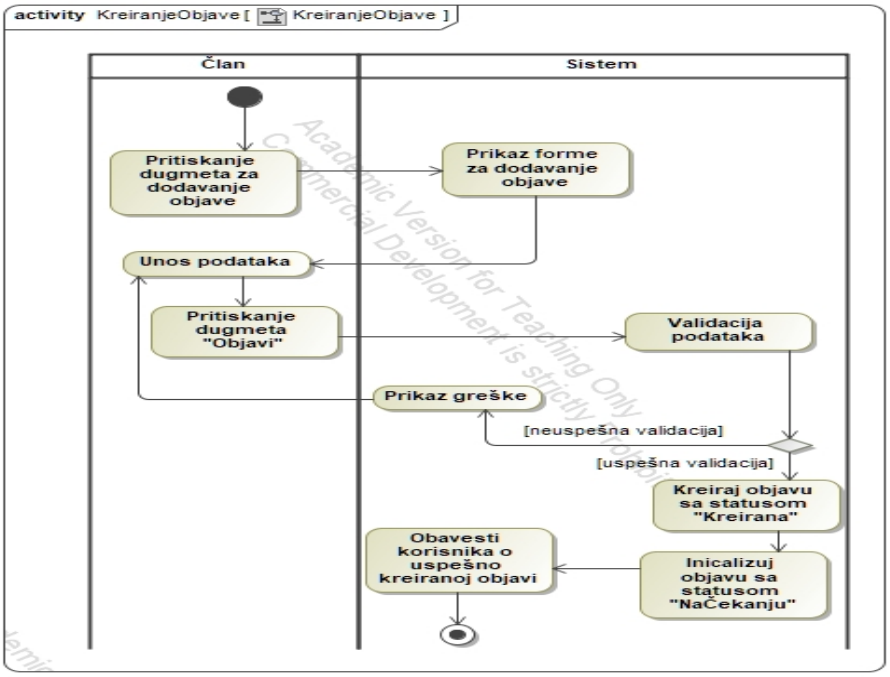
1. **Aktivnost objave**
2. todo
3. todo
4. todo

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 [Slika 10] i pregled objave [Slika 11].

Aktivnost kreiranja objave

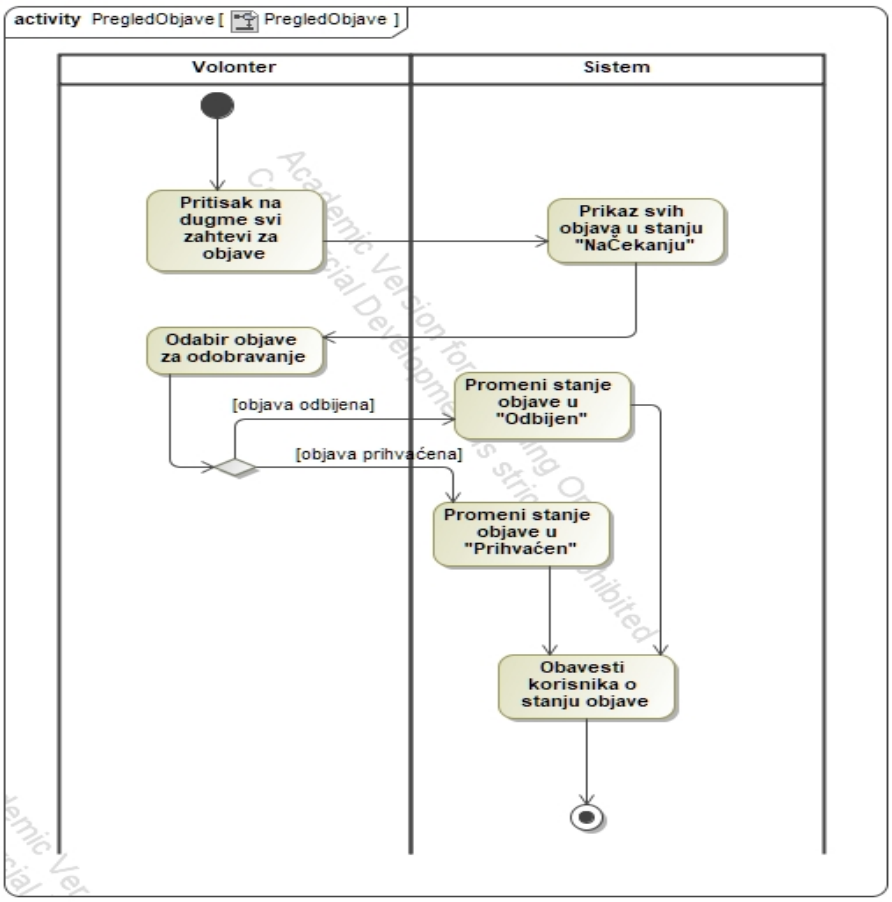
Nakon prijave, članu se nudi mogućnost da otvori formu za objavu. U formu se unose potrebne informacije o životinji u vidu teksta, slike ili videa. Sva polja moraju biti popunjena, u suprotnom se članu prikazuje poruka greške. Nakon uspešne provere unesenih podataka, korsnik se obaveštava da je objava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



Slika 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 slici 11.



Slika 11: Dijagram aktivnosti za pregled objave

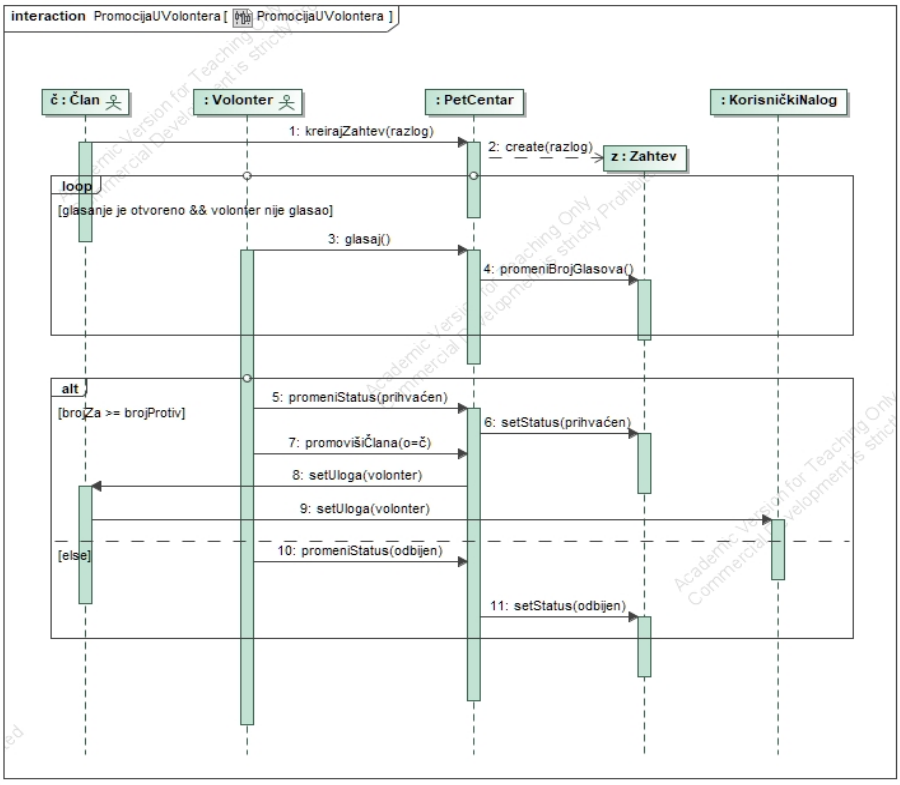
8 Dijagrami sekvence

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

- **Promocija u volontera**
- todo
- todo
- todo

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 slici 12.



Slika 12: Promocija u volontera

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 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 služe za komunikaciju sa podacima i poslovnom logikom.

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 2.

```
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 2: ICrud<T> interfejs

View

ViewModel

ViewModel je posrednički sloj između korisničkog interfejsa i domenskog sloja.

Za svaki entitet je napravljen **omotač** (eng. wrapper) [Listing 3], 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 3: 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 4] 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 4: 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 5: 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 6 ili preko anotacija [Listing 7].

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

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

Listing 6: Primer postavljanja unique ograničenja

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

Listing 7: Primer postavljanja anotacija

Konekcija

Da bi smo se uspešno povezali na bazu, moramo uneti parametre konekcije [Listing 8]. 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 8: Parametri konekcije

Primer *user-secret*-a se nalazi na listingu 9.

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

Listing 9: Primer *user-secret*

Konekciju uspostavljamo **UseNpgsql** metodom:

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

Listing 10: Uspostavljanje konekcije

Validacija

Validacija na nivou baze podataka je pisana u **SQL**-u. Najčešće korišćen vid validacije je **BEFORE INSERT/UPDATE TRIGGER** [Listing 11], 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 11: Primer validacije pomoću trigera

Na listingu 11 proveravamo da li je kreator zahteva *član* (vrednost 0).

10.3 Orgnizacija foldera