

Specifikacija projekta

1. Osnovne informacije o sistemu

Naziv teme: VetNet System

Logo:



Naziv tima: Quattor Homines

Nastavna grupa: Grupa 5 - utorak 15:00

Link na repozitorij tima: <https://github.com/OOAD-2023-2024/VetNet-System>

Članovi tima:

1. Mirza Mahmutović 19320
2. Ismar Muslić 19304
3. Nedim Banda 19460
4. Haris Mališević 19328

Namjena sistema:

Opisati sistem i njegovu namjenu sa maksimalno sedam rečenica. U okviru ovog polja potrebno je objasniti šta sistem treba raditi na apstraktnom nivou, bez detaljnog objašnjavanja pojedinačnih funkcionalnosti i načina razlikovanja aktera sistema (što je predmet daljih poglavljja).

VetNet sistem zamišljen je kao centralizovan način za evidentiranje vlasništva nad ljubimcima, zdravstvene njege i izdavanja lijekova. Ovaj sistem ima za cilj objediniti rad lokalnih veterinarskih stanica i apoteka kako bi se stvorio bolji uvid u stanje na terenu. Detaljan uvid na

*Univerzitet u Sarajevu
Elektrotehnički Fakultet
Objektno Orijentisana Analiza i Dizajn*

širem geografskom prostoru nudi bolju prevenciju zaraza, broj neudomljenih ljubimaca, ističe problematične pojave i poboljšava efikasnost liječenja.



2. Funkcionalnosti (poslovni procesi) sistema

Opisati 6 do 8 najznačajnijih funkcionalnosti sistema (u zavisnosti od broja članova u timu). Funkcionalnosti sistema predstavljaju usluge koje sistem pruža korisnicima. Sve funkcionalnosti pripadaju nekoj od različitih vrsta: u svrhu ostvarivanja krajne usluge sistema, persistencija podataka (CRUD operacije), operacije koje koriste principe asinhronne obrade zahtjeva, operacije koje koriste specifične algoritme obrade podataka i operacije u kojima se vrši korištenje vanjskih uređaja. Neophodno je navesti barem po jednu funkcionalnost svake od različitih vrsta.

- 1) **Naziv funkcionalnosti:** Registracija vlasnika

Vrsta funkcionalnosti: Perzistencija podataka (CRUD operacija) ▾

Opis funkcionalnosti:

Opisati način ostvarivanja funkcionalnosti sa maksimalno pet rečenica.

Za vlasnike ljubimaca bez korisničkog računa, moguće je kreirati račun kod veterinara prilikom posjete veterinarskoj službi. Čuvaju se osnovni lični podaci o vlasniku. Korisniku se izdaje ID broj i elektronska kartica.

- 2) **Naziv funkcionalnosti:** Registracija ljubimca

Vrsta funkcionalnosti: Perzistencija podataka (CRUD operacija) ▾

Opis funkcionalnosti:

Opisati način ostvarivanja funkcionalnosti sa maksimalno pet rečenica.

Prilikom prve posjete veterinaru, ljubimac se registruje u sistemu i čipuje. Čip nosi osnovne podatke o ljubimcu i njegovom vlasniku.

- 3) **Naziv funkcionalnosti:** Bilježenje pregleda

Vrsta funkcionalnosti: Usluga sistema ▾

Opis funkcionalnosti:

Opisati način ostvarivanja funkcionalnosti sa maksimalno pet rečenica.

Prilikom pregleda, veterinar unosi relevantne podatke za taj pregled (datum, vrijeme, tip zahvata i prateće bilješke) i po potrebi propisuje lijekove.

4) **Naziv funkcionalnosti:** Prikaz korisničkih podataka čitanjem elektronske kartice

Vrsta funkcionalnosti: Korištenje vanjskog uređaja ▾

Opis funkcionalnosti:

Opisati način ostvarivanja funkcionalnosti sa maksimalno pet rečenica.

Apoteke i veterinarske službe će imati odgovarajući čitač za fizičke kartice. Pomoću ovog uređaja, čitanjem kartice se daje uvid u osnovne korisničke podatke vlasnika kartice.

5) **Naziv funkcionalnosti:** Prikaz podataka ljubimca čitanjem elektronskog čipa

Vrsta funkcionalnosti: Korištenje vanjskog uređaja ▾

Opis funkcionalnosti:

Opisati način ostvarivanja funkcionalnosti sa maksimalno pet rečenica.

Veterinarske službe će imati odgovarajući čitač za čipove. Pomoću ovog uređaja, čitanjem čipa se daje uvid u osnovne podatke o ljubimcu i njegovom vlasniku.

6) **Naziv funkcionalnosti:** Evidencija i regulisanje izdavanja lijekova na recept

Vrsta funkcionalnosti: Usluga sistema ▾

Opis funkcionalnosti:

Opisati način ostvarivanja funkcionalnosti sa maksimalno pet rečenica.

Strogo regulisani lijekovi se izdaju isključivo na recept od veterinara. Apoteka čitanjem kartice korisnika dobija uvid u lijekove i doze lijekova koje taj korisnik može da kupi. Prilikom izdavanja lijeka, evidentira se količina izdatog lijeka i ažurira količina koja može da se izda. Ako je izdata maksimalna dozvoljena količina, recept se poništava.

7) **Naziv funkcionalnosti:** Poništavanje recepata koji su istekli

Vrsta funkcionalnosti: Asinhrona operacija ▾

Opis funkcionalnosti:

Opisati način ostvarivanja funkcionalnosti sa maksimalno pet rečenica.



Recepti koji nisu ispunjeni u predviđenom periodu, poništavaju se. Ova provjera provodi se svakog dana.

- 8) **Naziv funkcionalnosti:** Slanje relevantnih obavještenja korisnicima

Vrsta funkcionalnosti: Operacija sa specifičnim algoritmom obrade ▾

Opis funkcionalnosti:

Opisati način ostvarivanja funkcionalnosti sa maksimalno pet rečenica.

Ovisno od sitaucije, korisnicima se mogu poslati odgovarajuća obavještenja, npr. na osnovu lokacije korisnika, slanje obavijesti o planskoj vakcinaciji ljubimaca i sl.

- 9) **Naziv funkcionalnosti:** Analitika i izvještavanje

Vrsta funkcionalnosti: Usluga sistema ▾

Opis funkcionalnosti:

Opisati način ostvarivanja funkcionalnosti sa maksimalno pet rečenica.

Sistem periodično ili po potrebi generiše analitički izvještaj o zdravstvenom stanju ljubimaca, broju pregleda i/ili količini izdatih lijekova na nekom geografskom prostoru u datom periodu vremena.

3. Akteri sistema

Potrebno je navesti najmanje tri aktera sistema. Neophodno je navesti barem po jednog aktera za svaku od različitih vrsta.

Korisnici usluga sistema

- a) **Naziv aktera:** Vlasnik ljubimca

Vrsta aktera: Korisnik usluge ▾

Funkcionalnosti u kojima akter učestvuje:

Funkcionalnost sistema	Način učešća
1 - Registracija vlasnika	Mogućnost uređivanja ▾
2 - Registracija ljubimca	Mogućnost uređivanja ▾
6 - Evidencija i regulisanje izdavanja lijekova na recept	Mogućnost pregleda ▾
8 - Slanje relevantnih obavještenja korisnicima	Mogućnost pregleda ▾

- b) **Naziv aktera:** Veterinar

Vrsta aktera: Zaposlenik sistema ▾

Funkcionalnosti u kojima akter učestvuje:

Funkcionalnost sistema	Način učešća
1 - Registracija vlasnika	Mogućnost uređivanja ▾
2 - Registracija ljubimca	Mogućnost uređivanja ▾
3 - Bilježenje pregleda	Mogućnost uređivanja ▾
4 - Prikaz korisničkih podataka čitanjem elektronske kartice	Mogućnost pregleda ▾
5 - Prikaz podataka ljubimca čitanjem elektronskog čipa	Mogućnost pregleda ▾

c) **Naziv aktera:** Apotekar

Vrsta aktera: Zaposlenik sistema

Funkcionalnosti u kojima akter učestvuje:

Funkcionalnost sistema	Način učešća
4 - Prikaz korisničkih podataka čitanjem elektronske kartice	Mogućnost pregleda
6 - Evidencija i regulisanje izdavanja lijekova na recept	Mogućnost uređivanja
8 - Slanje relevantnih obavještenja korisnicima	Mogućnost pregleda

d) **Naziv aktera:** Administrator sistema

Vrsta aktera: Administrator

Funkcionalnosti u kojima akter učestvuje:

Funkcionalnost sistema	Način učešća
4 - Prikaz korisničkih podataka čitanjem elektronske kartice	Mogućnost uređivanja
5 - Prikaz podataka ljubimca čitanjem elektronskog čipa	Mogućnost uređivanja
7 - Poništavanje recepata koji su istekli	Mogućnost uređivanja
8 - Slanje relevantnih obavještenja korisnicima	Mogućnost uređivanja
9 - Analitika i izvještavanje	Mogućnost uređivanja



4. Nefunkcionalni zahtjevi sistema

Opisati najmanje tri najznačajnija nefunkcionalna zahtjeva sistema. Nefunkcionalni zahtjevi predstavljaju ograničenja koja sistem mora zadovoljiti kako bi mogao ispravno obavljati svoje funkcionalnosti. Validacije polja za unos vrijednosti ne predstavljaju nefunkcionalne zahtjeve.

- 1) **Naziv nefunkcionalnog zahtjeva:** Sistem dostupan za upotrebu 24/7

Opis:

Opisati ograničenje sistema i način na koje se ono ispoljava.

Zbog domena u kojem ovaj sistem djeluje, neophodno je obezbijediti pristup podacima u svakom trenutku. Iz tog razloga je neometan i kontinuiran rad sistema izrazito bitan za njegovo ispravno funkcionisanje.

- 2) **Naziv nefunkcionalnog zahtjeva:** Čuvanje historije recepata i pregleda

Opis:

Opisati ograničenje sistema i način na koje se ono ispoljava.

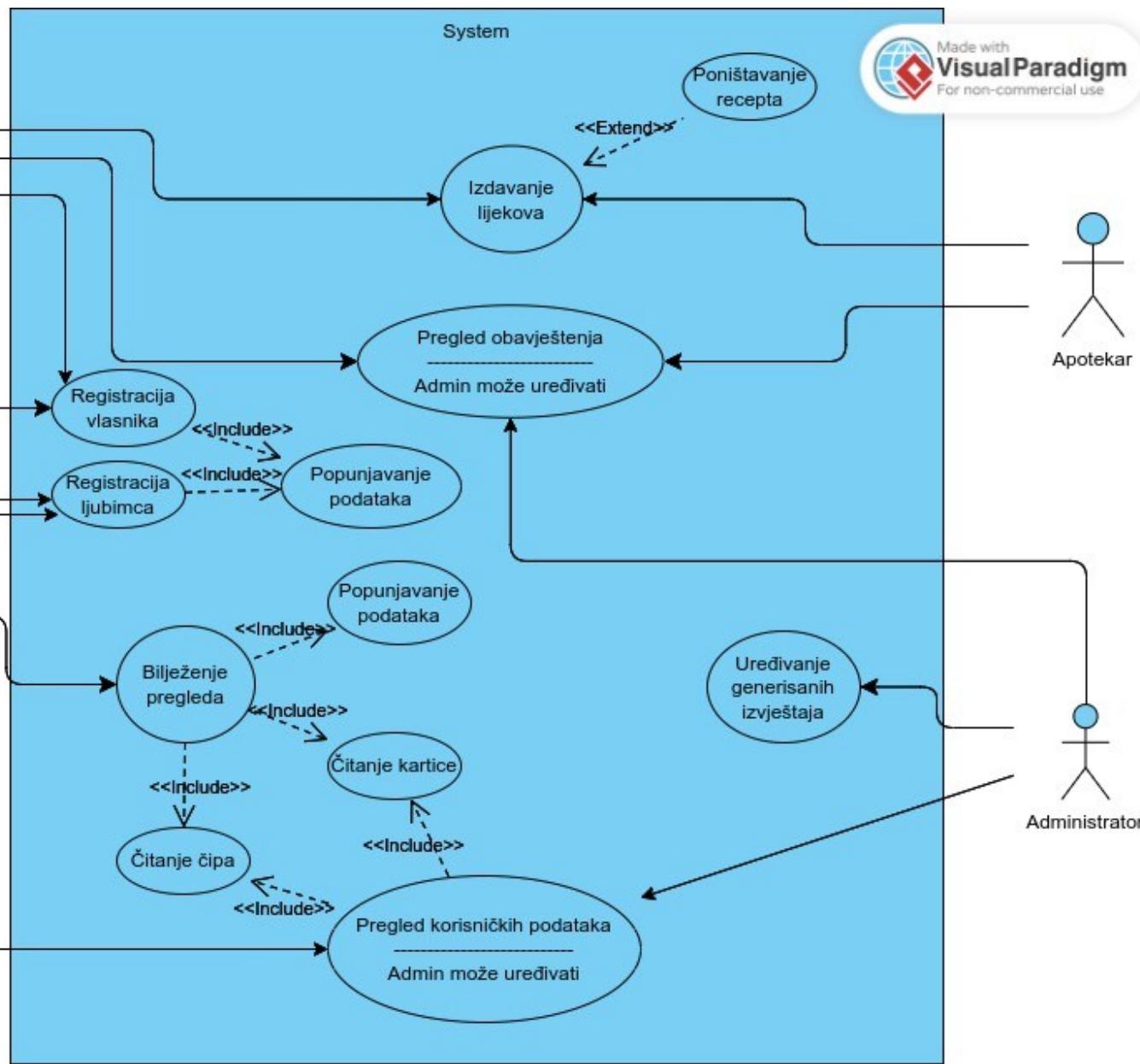
Recepti, čak i nakon što se ponište, ostaju sačuvani u bazi podataka. Izvještaji ranijih pregleda se čuvaju izvjesno vrijeme. Ovo daje uvid u zdravstvenu historiju neke životinje, nivou njege koju pružaju veterinar i vlasnik.

- 3) **Naziv nefunkcionalnog zahtjeva:** Ograničen pristup podacima

Opis:

Opisati ograničenje sistema i način na koje se ono ispoljava.

S obzirom na osjetljivost korisničkih podataka i odsustvo potrebe da pojedinačni vlasnici čitaju podatke, istima je pristup nemoguć. Pristup korisničkim podacima ili podacima ljubimca moguć je samo u veterinarskoj službi ili apoteci uz posjedovanje elektronske kartice ili čipa.



Use Case Dijagram

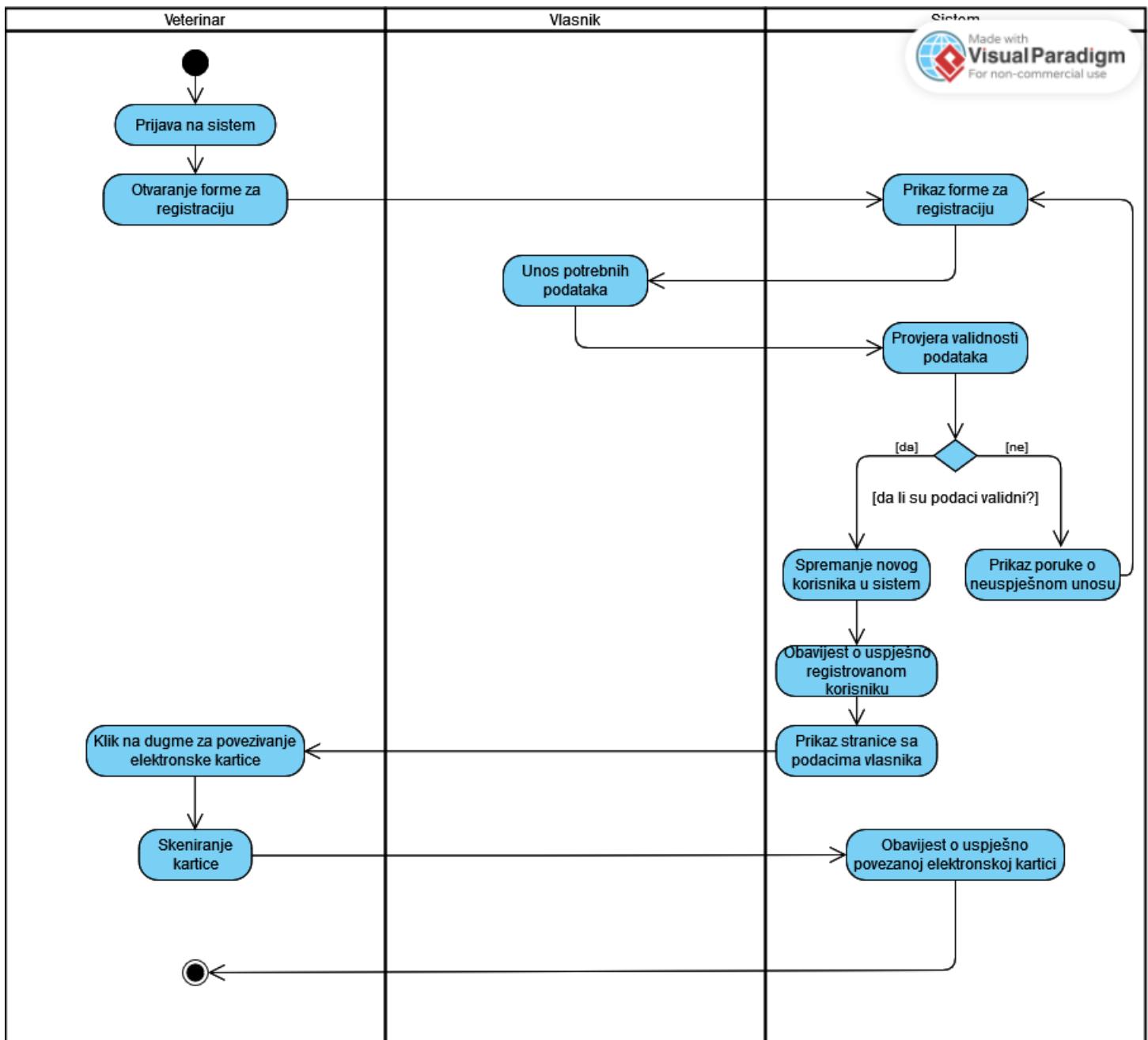
NAZIV SLUČAJA UPOTREBE	Registracija vlasnika
OPIS SLUČAJA UPOTREBE	Vlasnik zajedno sa veterinarom popunjava podatke i dobija elektronsku karticu
VEZANI ZAHTJEVI	Popunjavanje podataka
PREDUSLOVI	Veterinar se uspješno prijavio na sistem i otvorio formu za registraciju novog vlasnika
POSLJEDICE-USPJEŠAN ZAVRŠETAK	Uspješno registrovan novi vlasnik
POSLJEDICE-NEUSPJEŠAN ZAVRŠETAK	Nesupješna registracija vlasnika
PRIMARNI AKTERI	Vlasnik i veterinar
OSTALI AKTERI	/
GLAVNI TOK	Veterinar otvara formu za registraciju u koju zajedno sa vlasnikom unosi potrebne podatke i kreira novi račun. Veterinar za ovaj korisnički račun veže elektronsku karticu koju izdaje vlasniku.
ALTERNATIVE/PROŠIRENJA	/

USPJEŠAN ZAVRŠETAK – Registracija vlasnika

KORISNIK	SISTEM
1. Uspješna prijava veterinara u sistem	
2. Otvaranje forme za registraciju	
	3. Prikaz forme za registraciju
4. Unos potrebnih podataka	
	5. Provjera validnosti podataka
	6. Spremanje novog korisnika u sistem
	7. Obavijest o uspješno registrovanom korisniku
	8. Prikaz stranice sa podacima vlasnika i dugmeta za povezivanje elektronske kartice za taj račun
9. Klik na dugme za povezivanje elektronske kartice	
	10. Čekanje na skeniranje kartice
11. Skeniranje kartice	
	12. Obavijest o uspješno povezanoj elektronskoj kartici

NEUSPJEŠAN ZAVRŠETAK – Registracija vlasnika

KORISNIK	SISTEM
1. Uspješna prijava doktora u sistem	
2. Otvaranje forme za registraciju	
	3. Prikaz forme za registraciju
4. Unos potrebnih podataka	
	5. Provjera validnosti podataka
	6. Prikaz poruke o neuspješnom unosu



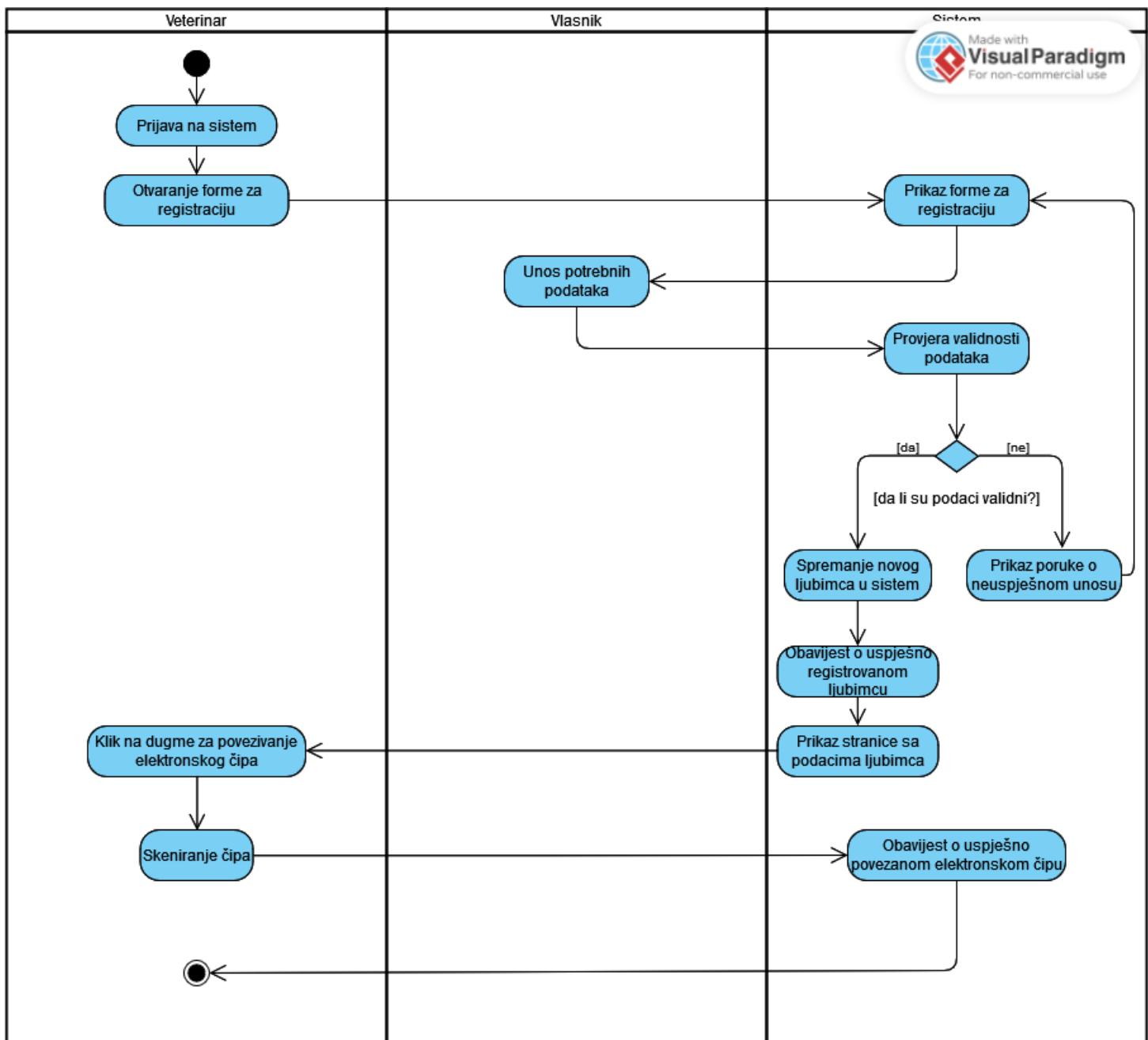
NAZIV SLUČAJA UPOTREBE	Registracija ljubimca
OPIS SLUČAJA UPOTREBE	Vlasnik zajedno sa veterinarom popunjava podatke za ljubimca koji se pored toga i čipuje.
VEZANI ZAHTJEVI	Popunjavanje podataka
PREDUSLOVI	Vlasnik ima već kreiran korisnički račun.
POSLJEDICE-USPJEŠAN ZAVRŠETAK	Uspješno registrovan novi ljubimac
POSLJEDICE-NEUSPJEŠAN ZAVRŠETAK	Nesupješna registracija ljubimca
PRIMARNI AKTERI	Vlasnik i veterinar
OSTALI AKTERI	/
GLAVNI TOK	Veterinar otvara formu za registraciju u koju zajedno sa vlasnikom unosi potrebne podatke i kreira novi račun. Veterinar za ovaj račun veže elektronski čip.
ALTERNATIVE/PROŠIRENJA	/

USPJEŠAN ZAVRŠETAK – Registracija ljubimca

KORISNIK	SISTEM
1. Uspješna prijava veterinara u sistem	
2. Otvaranje forme za registraciju	
	3. Prikaz forme za registraciju
4. Unos potrebnih podataka	
	5. Provjera validnosti podataka
	6. Spremanje novog ljubimca u sistem
	7. Obavijest o uspješno registrovanom ljubimcu
	8. Prikaz stranice sa podacima ljubimca i dugmeta za povezivanje čipa za taj račun
9. Klik na dugme za povezivanje elektronskog čipa	
	10. Čekanje na povezivanje čipa
11. Povezivanje čipa	
	12. Obavijest o uspješno povezanom čipu

NEUSPJEŠAN ZAVRŠETAK – Registracija vlasnika

KORISNIK	SISTEM
1. Uspješna prijava doktora u sistem	
2. Otvaranje forme za registraciju	
	3. Prikaz forme za registraciju
4. Unos potrebnih podataka	
	5. Provjera validnosti podataka
	6. Prikaz poruke o neuspješnom unosu



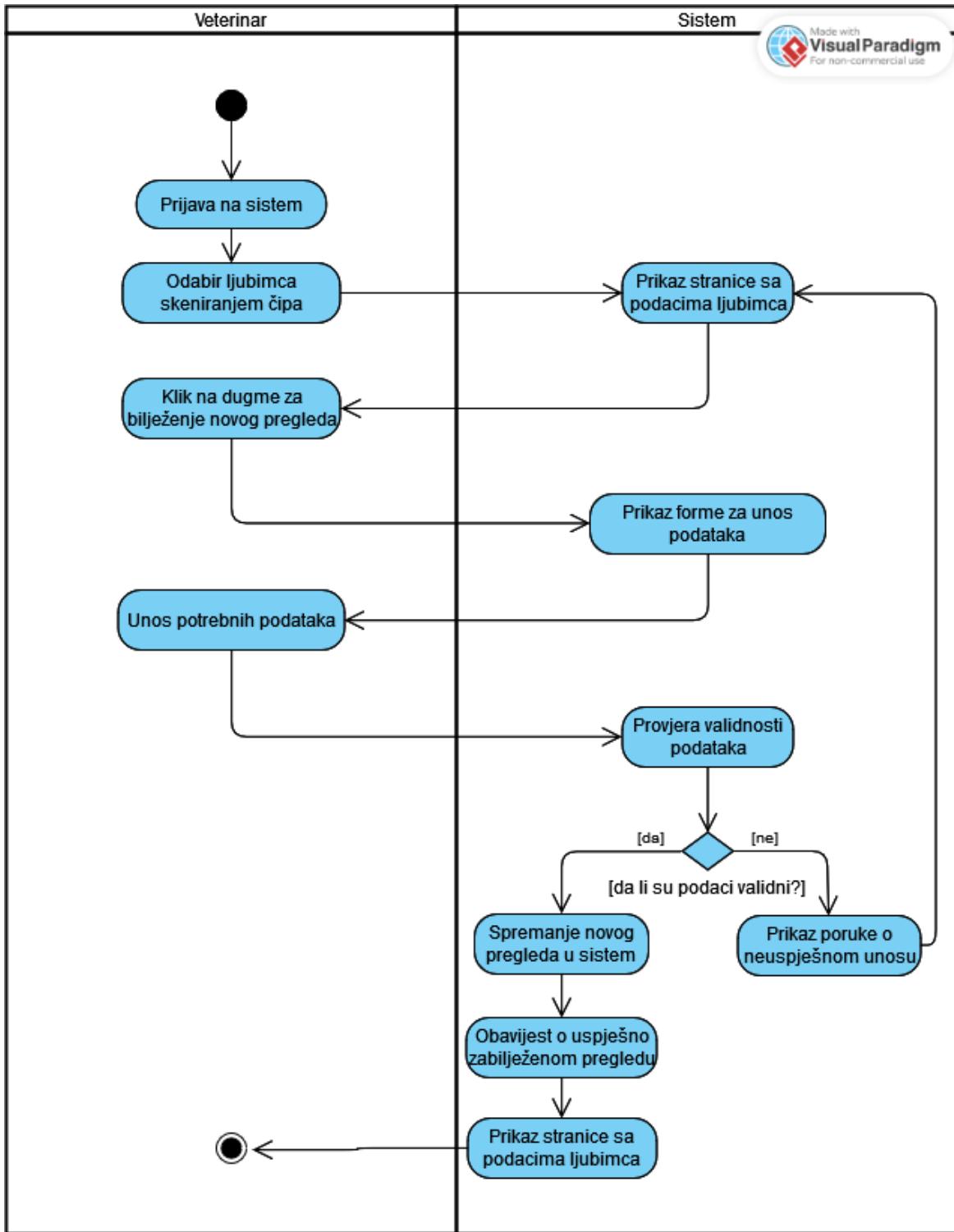
NAZIV SLUČAJA UPOTREBE	Bilježenje pregleda
OPIS SLUČAJA UPOTREBE	Veterinar popunjava obrazac i dodaje ga odabranom ljubimcu.
VEZANI ZAHTJEVI	Popunjavanje podataka, čitanje čipa ili kartice
PREDUSLOVI	Uspješna prijava veterinara u sistem i postojanje računa ljubimca u sistemu.
POSLJEDICE-USPJEŠAN ZAVRŠETAK	Uspješno zabilježen novi pregled.
POSLJEDICE-NEUSPJEŠAN ZAVRŠETAK	Nesupješno zabilježen novi pregled.
PRIMARNI AKTERI	Veterinar
OSTALI AKTERI	/
GLAVNI TOK	Veterinar se prijavljuje na sistem, skenira čip ljubimca, bira dodavanje novog pregleda, te popunjava fomu.
ALTERNATIVE/PROŠIRENJA	/

USPJEŠAN ZAVRŠETAK – Bilježenje pregleda

KORISNIK	SISTEM
1. Uspješna prijava veterinara u sistem	
2. Odabir ljubimca	
	3. Prikaz stranice sa podacima ljubimca i dugmetom za bilježenje novog pregleda
4. Klik na dugme za bilježenje novog pregleda	
	5. Prikaz forme za unos podataka
6. Popunjavanje potrebnih podataka	
	7. Provjera validnosti podataka
	8. Spremanje novog pregleda u sistemu
	9. Obavijest o uspješno zabilježenom pregledu
	10. Vraćanje na stranicu sa podacima ljubimca

NEUSPJEŠAN ZAVRŠETAK – Bilježenje pregleda

KORISNIK	SISTEM
1. Uspješna prijava veterinara u sistem	
2. Odabir ljubimca	
	3. Prikaz stranice sa podacima ljubimca i dugmetom za bilježenje novog pregleda
4. Klik na dugme za bilježenje novog pregleda	
	5. Prikaz forme za unos podataka
6. Popunjavanje potrebnih podataka	
	7. Provjera validnosti podataka
	8. Prikaz poruke o neuspješnom unosu



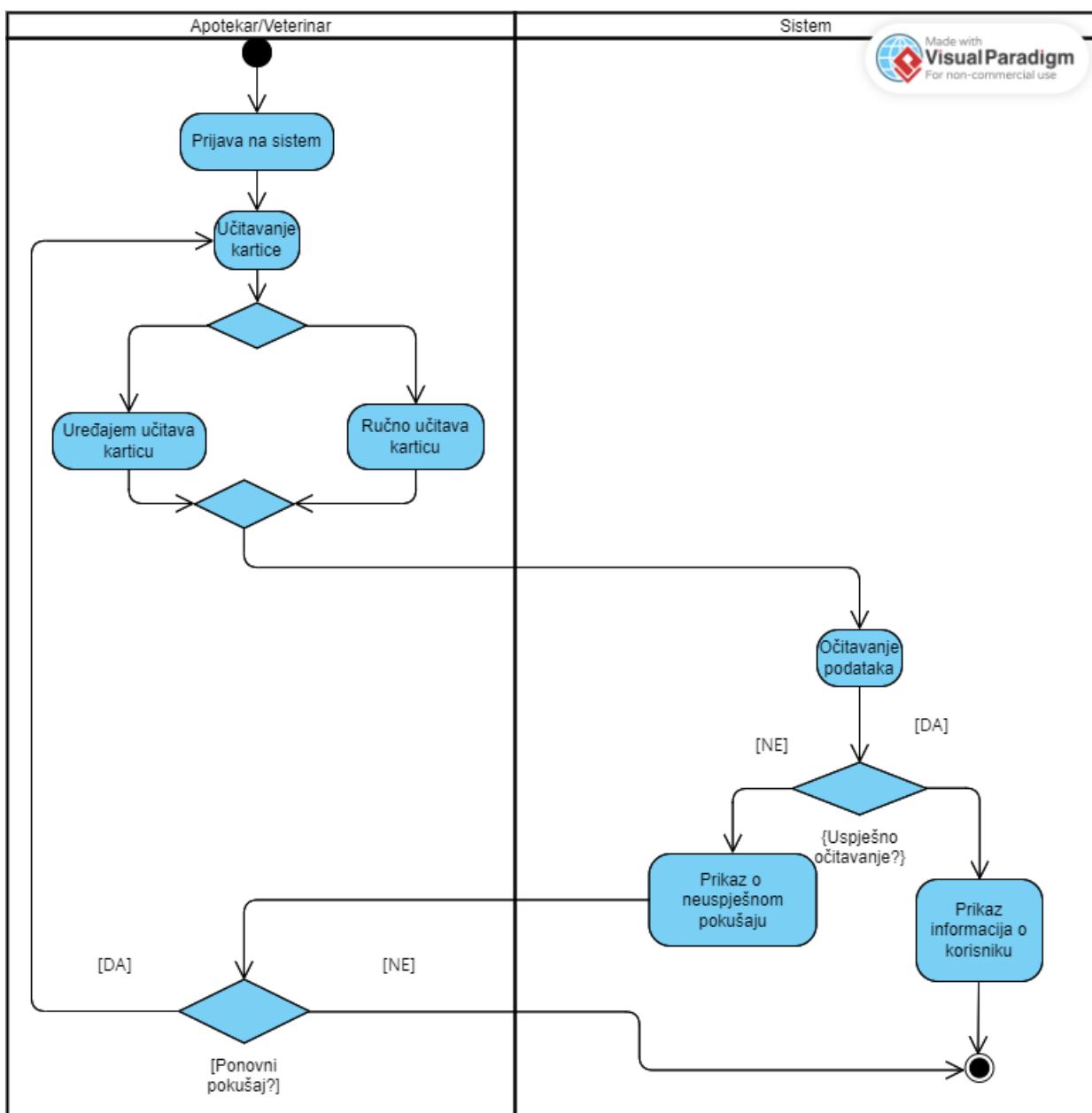
NAZIV SLUČAJA UPOTREBE	Prikaz korisničkih podataka čitanjem elektronske kartice
OPIS SLUČAJA UPOTREBE	Veterinar /apotekar dobija prikaz
VEZANI ZAHTJEVI	Prikaz podataka
PREDUSLOVI	Postojanje korisničkog računa, kartice
POSLJEDICE-USPJEŠAN ZAVRŠETAK	Uspješno očitana kartica i prikaz informacija
POSLJEDICE-NEUSPJEŠAN ZAVRŠETAK	Neuspješno očitana kartica
PRIMARNI AKTERI	Veterinar/apotekar
OSTALI AKTERI	/
GLAVNI TOK	Očitanje kartice. Nakon toga se prikazuju informacije o korisniku i informacije bitne za veterinara/apotekara
ALTERNATIVE/PROŠIRENJA	/

USPJEŠAN ZAVRŠETAK - Prikaz informacija

KORISNIK	SISTEM
1. Veterinar/apotekar očitava karticu korisnika	
	2. Sistem uspješno učitava
	3. Veterinar/apotekar učitava karticu korisnika

NEUSPJEŠAN ZAVRŠETAK - Neučitana kartica

KORISNIK	SISTEM
1. Veterinar/apotekar učitava karticu korisnika	
	2. Sistem neuspješno učitava
	3. Sistem otvara stranicu i ispisuje grešku. Odabir da se ponovo pokuša ili ručno unese kod korisnika



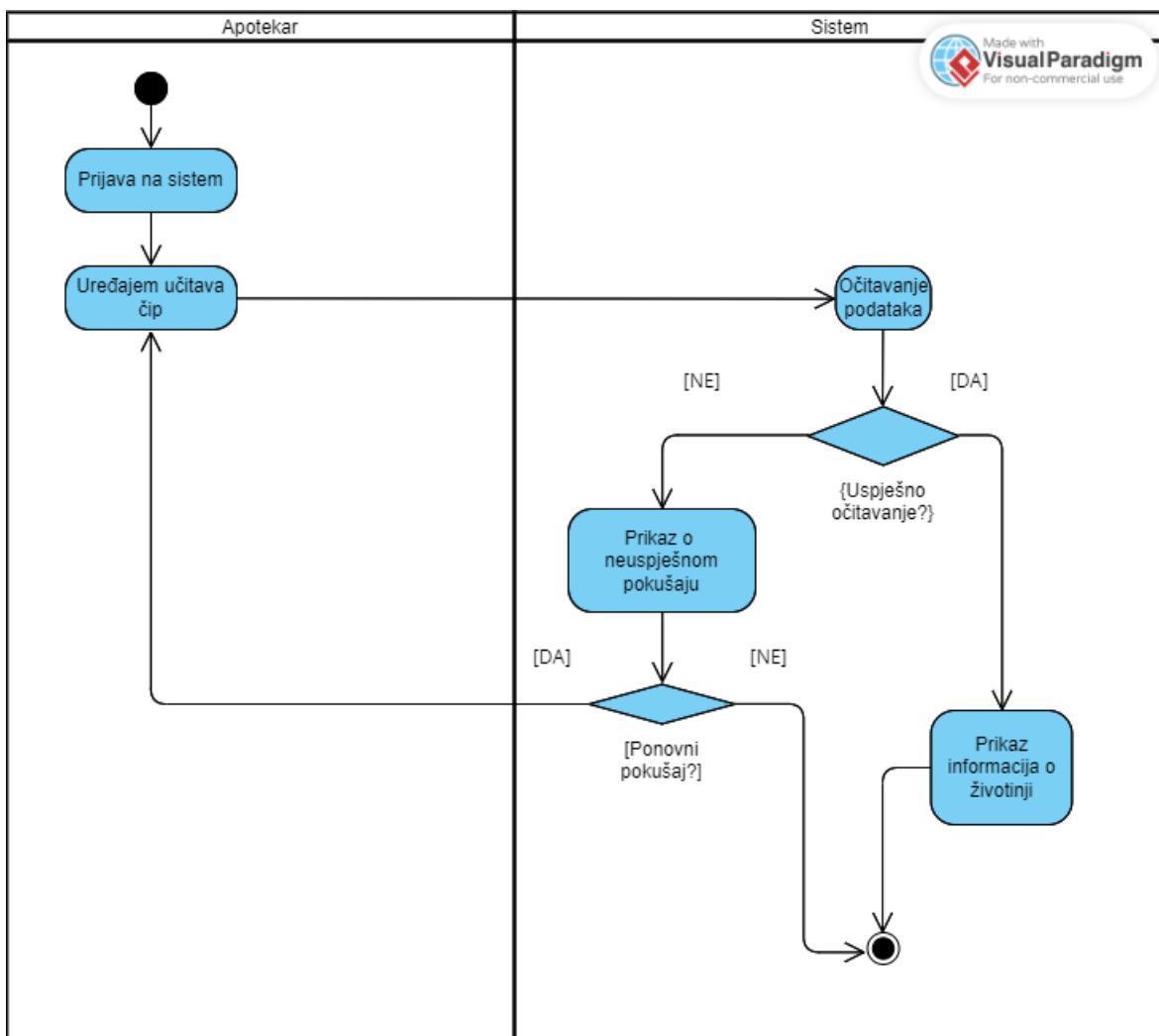
NAZIV SLUČAJA UPOTREBE	Prikaz podataka ljubimca čitanjem elektronskog čipa
OPIS SLUČAJA UPOTREBE	Uređajem očitava čip sa životinje u sistem
VEZANI ZAHTJEVI	Čitanje podataka
PREDUSLOVI	Postojeći čip koji se nalazi u bazi podataka
POSLJEDICE-USPJEŠAN ZAVRŠETAK	Prikazani podaci o životinji
POSLJEDICE-NEUSPJEŠAN ZAVRŠETAK	Ispisivanje greške kod čitanja, prijedlog za ponovni pokušaj
PRIMARNI AKTERI	Veterinar
OSTALI AKTERI	/
GLAVNI TOK	Veterinar uređajem očitava čip, sistem prikazuje podatke o životinji i opcije koje veterinar može uraditi
ALTERNATIVE/PROŠIRENJA	/

USPJEŠAN ZAVRŠETAK - Čitanje podataka

KORISNIK	SISTEM
1. Veterinar čita čip	2. Uspješno čitanje čipa
	3. Prikazuje podatke o životinji

NEUSPJEŠAN ZAVRŠETAK- Neuspješno čitanje

KORISNIK	SISTEM
1. Veterinar čita čip	
	2. Neuspješno učitavanje čipa
	3. Prikazuje se stranica koja ispisuje grešku, daje alternative učitavanje



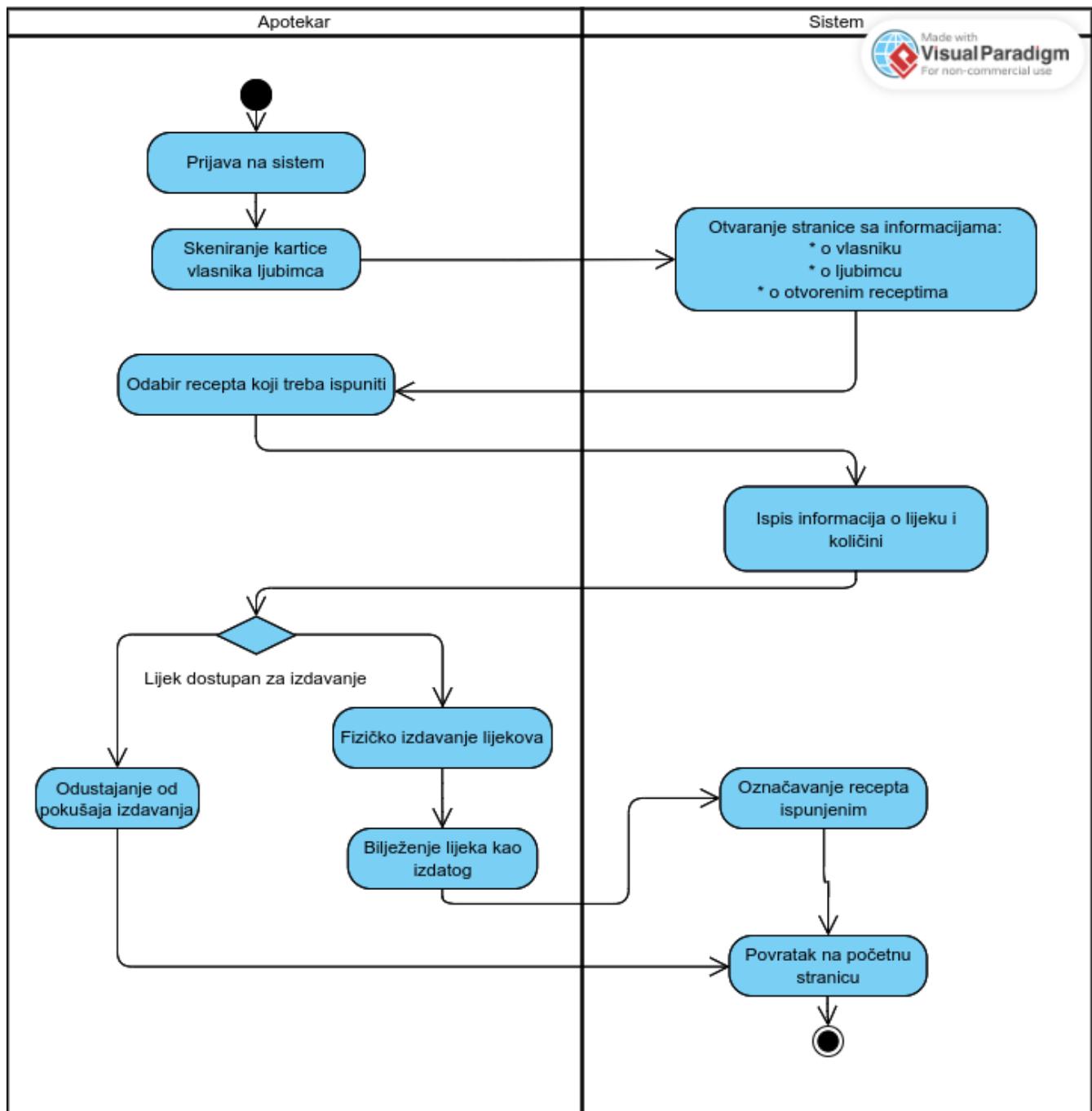
NAZIV SLUČAJA UPOTREBE	Evidencija i regulisanje izdavanja lijekova na recept
OPIS SLUČAJA UPOTREBE	Vlasnik preuzima lijek u apoteci, nakon što je propisan od veterinara
VEZANI ZAHTJEVI	Usluga sistema
PREDUSLOVI	Uspješna prijava u sistem i postojanje recepta u sistemu
POSLJEDICE-USPJEŠAN ZAVRŠETAK	Recept označen kao ispunjen
POSLJEDICE-NEUSPJEŠAN ZAVRŠETAK	Recept ostaje označen kao neispunjeno
PRIMARNI AKTERI	Veterinar
OSTALI AKTERI	Vlasnik ljubimca
GLAVNI TOK	Nakon propisanog lijeka, generisani recept se veže za račun vlasnika i ljubimca. Vlasnik od apotekara traži lijek na osnovu recepta. Ako je lijek dostupan, izdaje se u adekvatnoj količini i označi kao ispunjen.
ALTERNATIVE/PROŠIRENJA	Ako lijek nije dostupan u dатој apoteci, recept ostaje otvoren.

USPJEŠAN ZAVRŠETAK - Evidentiranje ispunjenog recepta

KORISNIK	SISTEM
1. Uspješna prijava apotekara u sistem	
2. Skeniranje kartice vlasnika ljubimca	
	3. Sistem otvara stranicu sa podacima o vlasniku, ljubimcu i daje listu otvorenih recepata
4. Odabir recepta koji treba ispuniti	
	5. Sistem ispisuje informacije o lijeku i količini koja se izdaje
6. Fizičko izdavanje lijekova	
7. Bilježenje lijeka kao izdatog	
	8. Označavanje recepta ispunjenim
	9. Vraćanje na početnu stranicu

NEUSPJEŠAN ZAVRŠETAK - Evidentiranje ispunjenog recepta

KORISNIK	SISTEM
1. Uspješna prijava apotekara u sistem	
2. Skeniranje kartice vlasnika ljubimca	
	3. Sistem otvara stranicu sa podacima o vlasniku, ljubimcu i daje listu otvorenih recepata
4. Odabir recepta koji treba ispuniti	
	5. Sistem ispisuje informacije o lijeku i količini koja se izdaje
6. Lijek nije dostupan	
7. Odustajanje od pokušaja izdavanja	
	8. Recept ostaje označen kao neispunjen



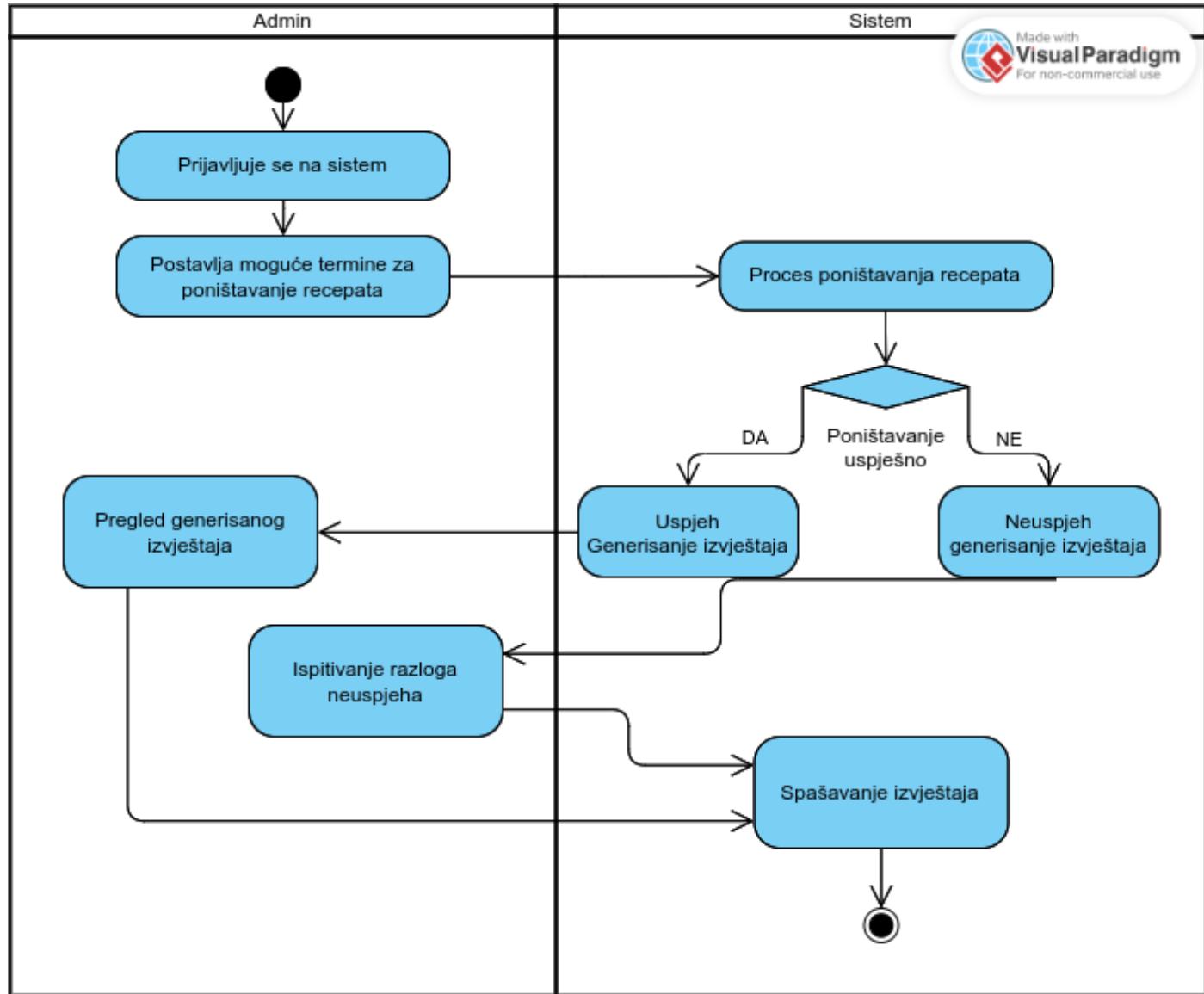
NAZIV SLUČAJA UPOTREBE	Poništavanje recepata koji su istekli
OPIS SLUČAJA UPOTREBE	Sistem asinhrono, kada je minimalno opterećen, poništava recepte koji su istekli
VEZANI ZAHTJEVI	Asinhrona operacija
PREDUSLOVI	Uspješno poništeni recepti koji su istekli
POSLJEDICE-USPJEŠAN ZAVRŠETAK	Recepti označeni kao neispunjeni i kao istekli
POSLJEDICE-NEUSPJEŠAN ZAVRŠETAK	Recept ostaje označen kao otvoren
PRIMARNI AKTERI	Administrator
OSTALI AKTERI	/
GLAVNI TOK	Sistem asinhrono pokreće funkciju baze podataka koja prolazi kroz tabelu otvorenih recepata i one koji su istekli označava kao poništene
ALTERNATIVE/PROŠIRENJA	Ako sistem nije u mogućnosti, poništavanje se neće provesti

USPJEŠAN ZAVRŠETAK - Poništavanje recepata koji su istekli

KORISNIK	SISTEM
1. Administrator se uspješno prijavljuje	
2. Administrator postavlja moguće termine za poništavanje recepata	
	3. Pokreće se proces poništavanja recepata
	4. Generiše se izvještaj o poništenim receptima
5. Administrator pregleda generisani izvještaj	
	6. Generisani izvještaj se spašava

NEUSPJEŠAN ZAVRŠETAK- Poništavanje recepata koji su istekli

KORISNIK	SISTEM
1. Administrator se uspješno prijavljuje	
2. Administrator postavlja moguće termine za poništavanje recepata	
	3. Pokreće se proces poništavanja recepata
	4. Poništavanje nije uspješno
	5. Generiše se izvještaj o neuspjehu
6. Administrator pregleda generisani izvještaj	
7. Administrator ispituje razlog neuspjeha	
	8. Generisani izvještaj se spašava



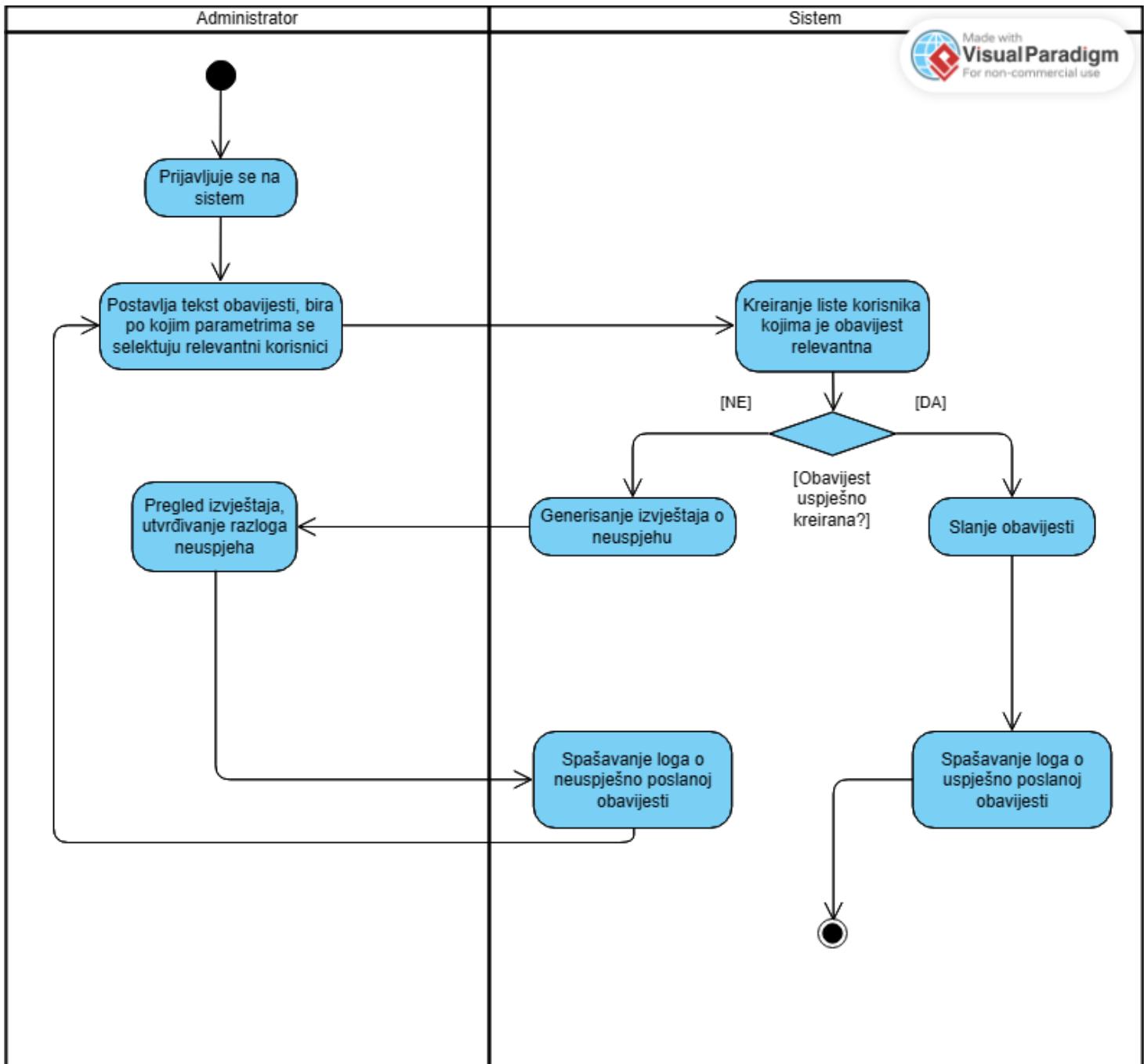
NAZIV SLUČAJA UPOTREBE	Slanje obavijesti svim ili određenim korisnicima od strane sistema
OPIS SLUČAJA UPOTREBE	Na osnovu zadanih parametara, sistem korisnicima šalje relevantne obavijesti
VEZANI ZAHTJEVI	/
PREDUSLOVI	Korisnik registrovan u sistemu
POSLJEDICE-USPJEŠAN ZAVRŠETAK	Korisnici dobijaju obavijesti na vrijeme
POSLJEDICE-NEUSPJEŠAN ZAVRŠETAK	Korisnici dobijaju obavijesti kasno ili ih ne dobiju nikako
PRIMARNI AKTERI	Administrator
OSTALI AKTERI	/
GLAVNI TOK	Administrator postavlja tekst obavijesti, te bira parametar po kom su korisnici slični. Obavještenje se šalje svim korisnicima iz slične/iste grupe
ALTERNATIVE/PROŠIRENJA	/

USPJEŠAN ZAVRŠETAK - Poništavanje recepata koji su istekli

KORISNIK	SISTEM
1. Administrator se prijava na sistem	
2. Administrator postavlja tekst obavijesti i parametre po kojima se obavijest šalje	
	3. Sistem iz baze podataka kreira listu korisnika na osnovu parametra kojeg je zadao administrator (adresa stanovanja, rasa ljubimca i sl.)
	4. Obavijest se šalje svim korisnicima iz kreirane liste, administrator dobija potvrdu o uspješnom slanju
	5. Spašava se log o uspješno posланом obavještenju

NEUSPJEŠAN ZAVRŠETAK- Poništavanje recepata koji su istekli

KORISNIK	SISTEM
1. Administrator se prijavljuje na sistem	
2. Administrator postavlja tekst obavijesti i parametre po kojima se obavijest šalje	
	3. Sistem iz baze podataka kreira listu korisnika na osnovu parametra kojeg je zadao administrator (adresa stanovanja, rasa ljubimca i sl.)
	4. U listi ne postoji niti jedan korisnik, ili obavijest ne uspijeva biti poslana iz drugog tehničkog razloga
	5. Generiše se izvještaj o neuspjehu
6. Administrator pregleda generisani izvještaj i utvrđuje razlog neuspjeha	
	7. Spašava se log o neuspjelom slanju obavijesti



NAZIV SLUČAJA UPOTREBE	Analitika i izvještavanje
OPIS SLUČAJA UPOTREBE	Sistem po zahtjevu, ili automatski, kreira izvještaj o zdravstvenom stanju ljubimaca, količini izdatih lijekova, broju obavljenih pregleda i slično za određeno geografsko područje.
VEZANI ZAHTJEVI	Redovno bilježenje pregleda i izdatih lijekova
PREDUSLOVI	Postojanje urednih zapisa o izdanim lijekovima i pregledima
POSLJEDICE-USPJEŠAN ZAVRŠETAK	Uspješno generisan izvještaj
POSLJEDICE-NEUSPJEŠAN ZAVRŠETAK	Izvještaj nije generisan
PRIMARNI AKTERI	Administrator
OSTALI AKTERI	/
GLAVNI TOK	Sistem po zahtjevu, ili automatski, prolazi kroz sve zabilješke o izdanim lijekovima kao i obavljenim pregledima te daje analitički izvještaj o istima.
ALTERNATIVE/PROŠIRENJA	Ako određeni lijek nije izdan, neće se ni prikazati u izvještaju.

USPJEŠAN ZAVRŠETAK - Poništavanje recepata koji su istekli

KORISNIK	SISTEM
1. Administrator se uspješno prijavljuje na sistem	
2. Administrator šalje zahtjev za generisanjem izvještaja	
	3. Pokreće se proces kreiranja izvještaja
	4. Generiše se izvještaj na osnovu zabilježenih pregleda i izdanih lijekova, te se prikazuje administratoru
5. Administrator pregleda generisani izvještaj	
	6. Generisani izvještaj se spašava

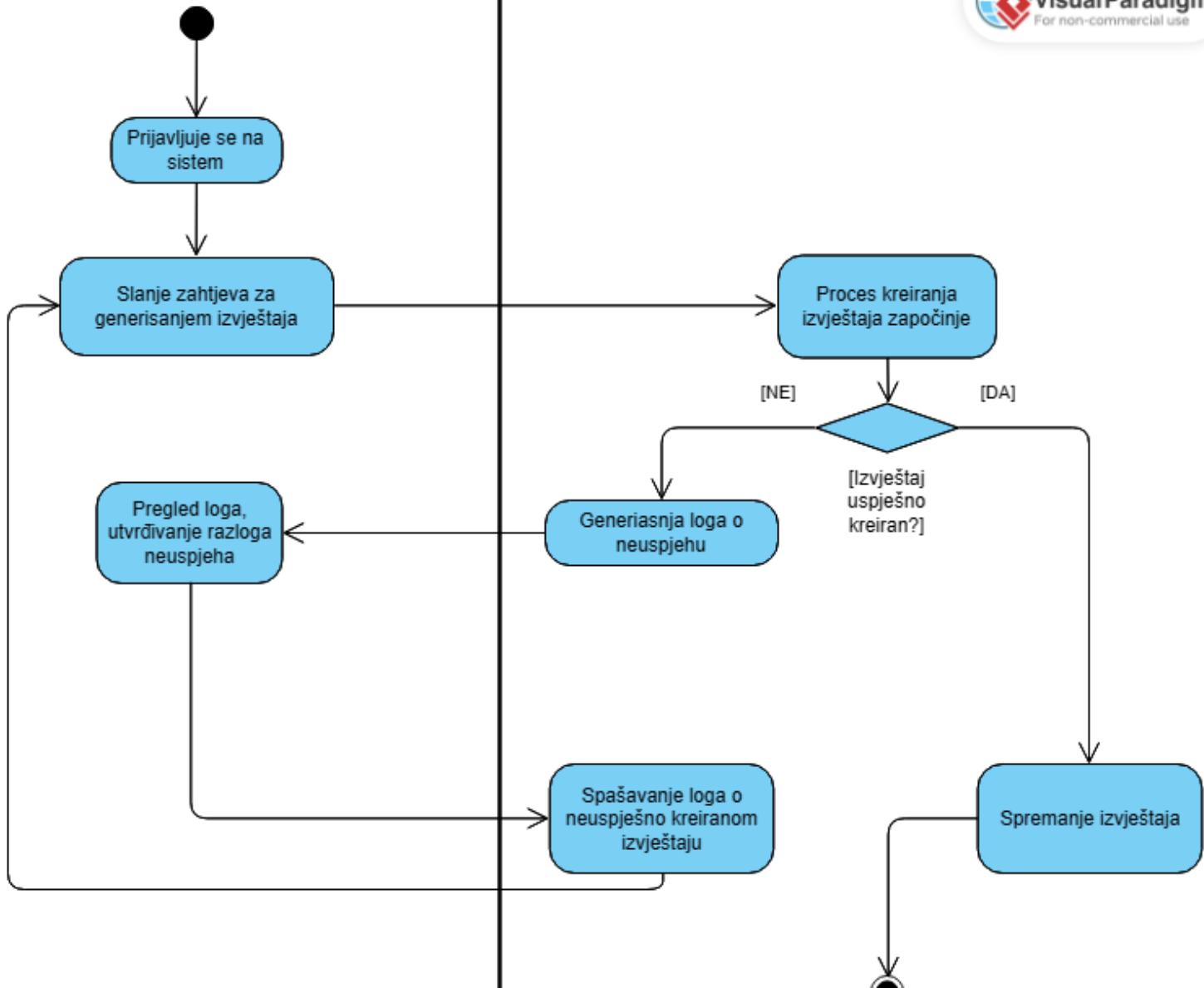
NEUSPJEŠAN ZAVRŠETAK- Poništavanje recepata koji su istekli

KORISNIK	SISTEM
1. Administrator se uspješno prijavljuje na sistem	
2. Administrator šalje zahtjev za generisanjem izvještaja	
	3. Pokreće se proces kreiranja izvještaja
	4. Kreiranje izvještaja je neuspješno
	5. Generiše se log o neuspjehu kreiranja izvještaja
6. Administrator utvrđuje razlog neuspjeha te eventualno pokušava pokrenuti novi proces generisanja izvještaja	
	7. Spašava se log o neuspjehu kreiranja izvještaja u sistemu

Administrator

Sistem

Made with
Visual Paradigm
For non-commercial use



Analiza i dizajn sistema

U nastavku je potrebno definisati sve potencijalne klase koje će se koristiti u sistemu. Za određivanje klasa koje će biti neophodne za rad sistema potrebno je koristiti specifikaciju sistema i prethodno kreirane dijagrame.

Template za jednu klasu potrebno je iskopirati onoliko puta koliko je neophodno da bi se definisale sve klase u sistemu.

Definicija klasa u sistemu

Naziv klase: Korisnik

Funkcionalni zahtjevi u kojima klasa učestvuje:

Svim funkcionalnim zahtjevima

Atributi koje klasa posjeduje:

Naziv atributa	Tip varijable	Dodatne napomene
Ime	String	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je enumeration
Prezime	String	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je enumeration
Username	String	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je enumeration
Password	String	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je enumeration
Tip	TipKorisnika	<input type="checkbox"/> Atribut je statički <input checked="" type="checkbox"/> Atribut je enumeration
DatumRodjenja	Date	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je enumeration
Spol	Spol	<input type="checkbox"/> Atribut je statički <input checked="" type="checkbox"/> Atribut je enumeration
Adresa	String	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je enumeration
BrojTelefona	String	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je enumeration

Naziv klase: Vlasnik (nasljeđuje klasu Korisnik)

Funkcionalni zahtjevi u kojima klasa učestvuje:

- FZ br. 1: Registracija vlasnika
- FZ br. 2: Registracija ljubimca
- FZ br. 4: Prikaz korisničkih podataka čitanjem elektronske kartice
- FZ br. 6: Evidencija i regulisanje izdavanje lijekova na recept
- FZ br. 8: Slanje relevantnih obavještenja korisnicima

Atributi koje klasa posjeduje:

Naziv atributa	Tip varijable	Dodatne napomene
Ljubimci	ArrayList<Ljubimac>	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
Recepti	ArrayList<Recept>	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>

Naziv klase: Apotekar (nasljeđuje klasu Korisnik)

Funkcionalni zahtjevi u kojima klasa učestvuje:

FZ br. 4: Prikaz korisničkih podataka čitanjem elektronske kartice

FZ br. 5: Prikaz podataka ljubimca čitanjem elektronskog čipa

FZ br. 6: Evidencija i regulisanje izdavanje lijekova na recept

Atributi koje klasa posjeduje:

Naziv atributa	Tip varijable	Dodatne napomene
Poslovnica	Poslovnica	<input type="checkbox"/> Atribut je statički <input checked="" type="checkbox"/> Atribut je <i>enumeration</i>

Naziv klase: Veterinar (nasljeđuje klasu Korisnik)

Funkcionalni zahtjevi u kojima klasa učestvuje:

FZ br. 3: Bilježenje pregleda

FZ br. 4: Prikaz korisničkih podataka čitanjem elektronske kartice

FZ br. 5: Prikaz podataka ljubimca čitanjem elektronskog čipa

FZ br. 6: Evidencija i regulisanje izdavanje lijekova na recept

Atributi koje klasa posjeduje:

Naziv atributa	Tip varijable	Dodatne napomene
VeterinarskaSluzba	Sluzba	<input type="checkbox"/> Atribut je statički <input checked="" type="checkbox"/> Atribut je <i>enumeration</i>
TipSpecijalizacije	Specijalizacija	<input type="checkbox"/> Atribut je statički <input checked="" type="checkbox"/> Atribut je <i>enumeration</i>

Naziv klase: Ljubimac

Funkcionalni zahtjevi u kojima klasa učestvuje:

FZ br. 2: Registracija ljubimca

FZ br. 3: Bilježenje pregleda

FZ br. 5: Prikaz podataka ljubimca čitanjem elektronskog čipa

FZ br. 6: Evidencija i regulisanje izdavanje lijekova na recept

Atributi koje klasa posjeduje:

Naziv atributa	Tip varijable	Dodatne napomene
Ime	String	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
Vlasnik	Vlasnik	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
Spol	Spol	<input type="checkbox"/> Atribut je statički <input checked="" type="checkbox"/> Atribut je <i>enumeration</i>
Vrsta	Vrsta	<input type="checkbox"/> Atribut je statički <input checked="" type="checkbox"/> Atribut je <i>enumeration</i>
Rasa	Rasa	<input type="checkbox"/> Atribut je statički <input checked="" type="checkbox"/> Atribut je <i>enumeration</i>
DatumRodjenja	Date	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
Slika	String	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
Pregledi	ArrayList<Pregled>	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
QRCode	QR Code	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>

Naziv klase: Recept

Funkcionalni zahtjevi u kojima klasa učestvuje:

FZ br. 3: Bilježenje pregleda

FZ br. 5: Prikaz korisničkih podataka čitanjem elektronske kartice

FZ br. 6: Evidencija i regulisanje izdavanje lijekova na recept

Atributi koje klasa posjeduje:

Naziv atributa	Tip varijable	Dodatne napomene
DatumVrijeme	DateTime	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
Ljubimac	Ljubimac	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
Lijek	String	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
Doza	Int	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
Veterinar	Veterinar	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
Napomena	String	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>

Naziv klase: Pregled

Funkcionalni zahtjevi u kojima klasa učestvuje:

FZ br. 3: Bilježenje pregleda

FZ br. 5: Prikaz podataka ljubimca čitanjem elektronskog čipa

Atributi koje klasa posjeduje:

Naziv atributa	Tip varijable	Dodatne napomene
DatumVrijeme	DateTime	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
Ljubimac	Ljubimac	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
Razlog	String	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
Postpuak	String	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
Dijagnoza	String	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
Terapija	Boolean	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
Veterinar	Veterinar	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
Napomena	String	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>

Naziv klase: Izvjestaj

Funkcionalni zahtjevi u kojima klasa učestvuje:

FZ br. 9: Analitika i izvještavanje

Atributi koje klasa posjeduje:

Naziv atributa	Tip varijable	Dodatne napomene
DatumVrijeme	DateTime	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
Sadrzaj	String	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>

Naziv klase: Obavjestenje

Funkcionalni zahtjevi u kojima klasa učestvuje:

FZ br. 8: Slanje relevantnih obavještenja korisnicima

Atributi koje klasa posjeduje:

Naziv atributa	Tip varijable	Dodatne napomene
DatumVrijeme	DateTime	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
Sadrzaj	String	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>

Prototipovi korisničkog sučelja



Korisničko ime: _____

Lozinka: _____

Prijava greške [Zaboravljena lozinka?](#) Login



Novi pregled

Datum i vrijeme: _____

Ljubimac: _____

Razlog: _____

Postupak: _____

Dijagnoza: _____

Terapija: _____

Veterinar: _____

Napomena: _____



Novi recept

Datum i vrijeme:

Ljubimac:

Lijek:

Doza:

Napomena:



Flaki

Vlasnik:

Spol:

Rasa:

Datum rođenja:



Prikaži indeks zabilježenih pregleda

Podaci o vakcinaciji

Zabilježi novi pregled

Novi ljekarski recept



Mujo Mujić

Registrovani ljubimci:

Datum rođenja: _____

Spol: _____

Adresa: _____

Broj telefona: _____

[Prikaži listu izdatih recepata](#)

FLAKI

Pomeranac



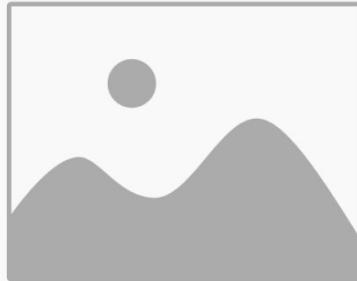
Svi korisnici/ljubimci...

Ime i prezime korisnika	Adresa	Three	Four
Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed	161651561	425614896	A
At vero eos et accusam et justo duo dolores et ea	243125151	15265247	B
Stet clita kasd gubergren, no sea takimata sanctus est Lorem	34587654	12531467	C
Eirmod tempor invidunt ut labore et dolore magna aliquyam	87643452	9556342	D
Stet clita kasd gubergren, no sea takimata sanctus	95463425	82346315	E

Skeniranje



**Prinesite QR
kod kameri**



Poslovnica

Adresa:

Broj telefona:

e-mail:



Novo obavještenje

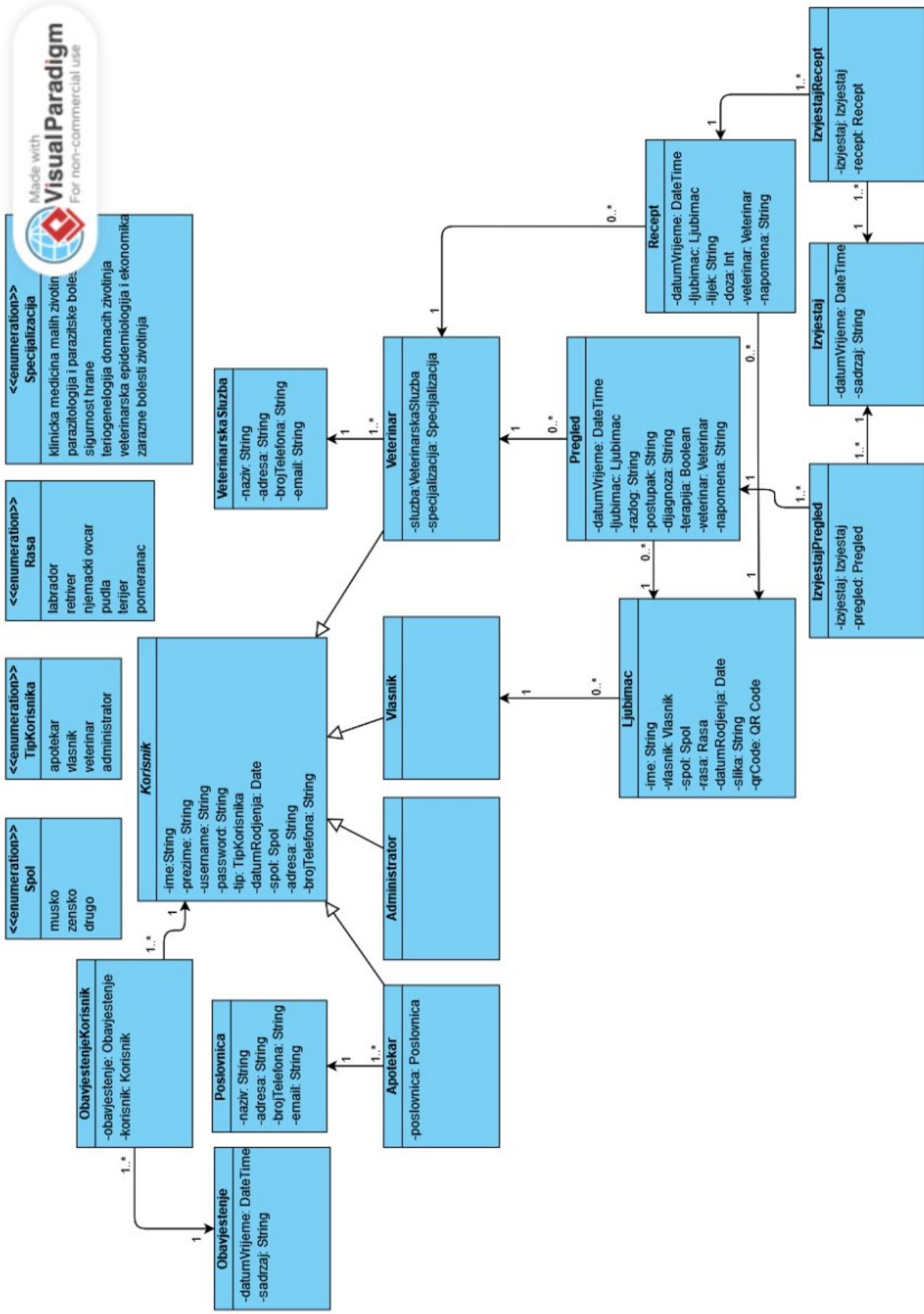
Filtriranje po: ▾

Adresa: _____

Ljubimac: _____

Tekst obavještenja:

Dijagram klasa



SOLID principi

Tim 19 – VetNet System

1. Single responsibility principle – princip pojedinačne odgovornosti

Ovaj princip nam govori da bi svaka klasa trebala imati samo jedan razlog za promjenu. Odnosno, svaka klasa treba da zna samo o jednoj stvari i da ima samo jednu odgovornost. U našem sistemu svaka klasa ima svoju odgovornost čime je ovaj zahtjev ispunjen.

2. Open closed principle – otvoreno zatvoren princip

U sistemu, klase su povezane na način da jedna klasa kao atribut sadrži objekat tipa druge klase. Samim tim, mijenjanje jedne klase neće uzrokovati mijenjanje druge klase. Npr. klasa *Pregled* kao svoje attribute sadrži objekte tipa *Ljubimac* i *Veterinar*. Mijenjanje ovih klasa neće narušiti funkcionalnost klase *Pregled*.

3. Liskov substitution principle – Liskov princip zamjene

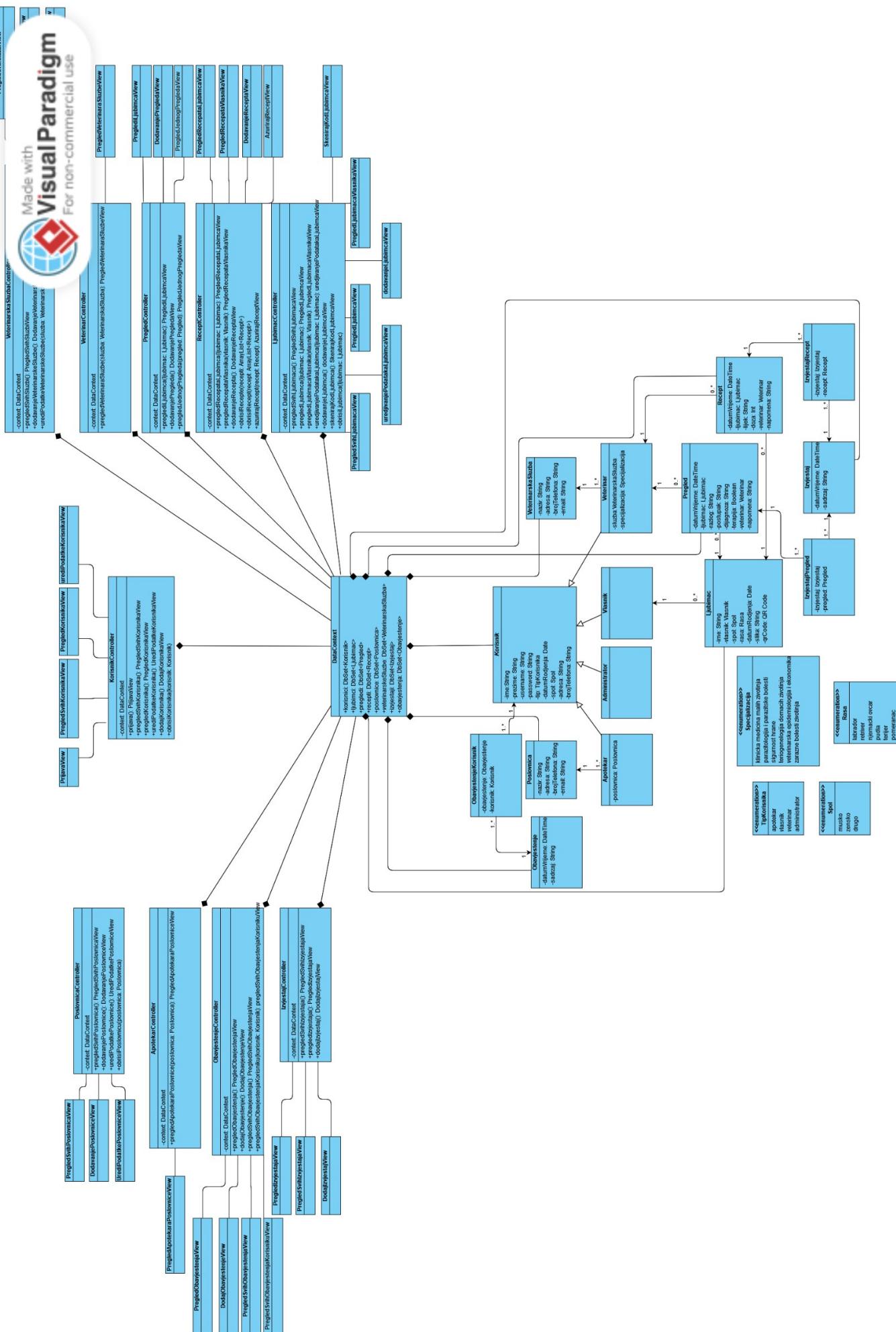
Dijagram klase sadrži apstraktну klasu *Korisnik*. Iz nje su izvedene klase *Vlasnik*, *Apotekar* i *Veterinar*. Prilikom dodavanja novog korisnika u sistem, umjesto objekta tipa klase *Korisnik* mi dodajemo objekat tipa *Vlasnik*, *Apotekar* ili *Veterinar*. Kako svaki od ovih objekata smisleno zadržava sve attribute klase *Korisnik*, tako je ispunjen i ovaj princip.

4. Interface segregation principle – princip izoliranja interfejsa

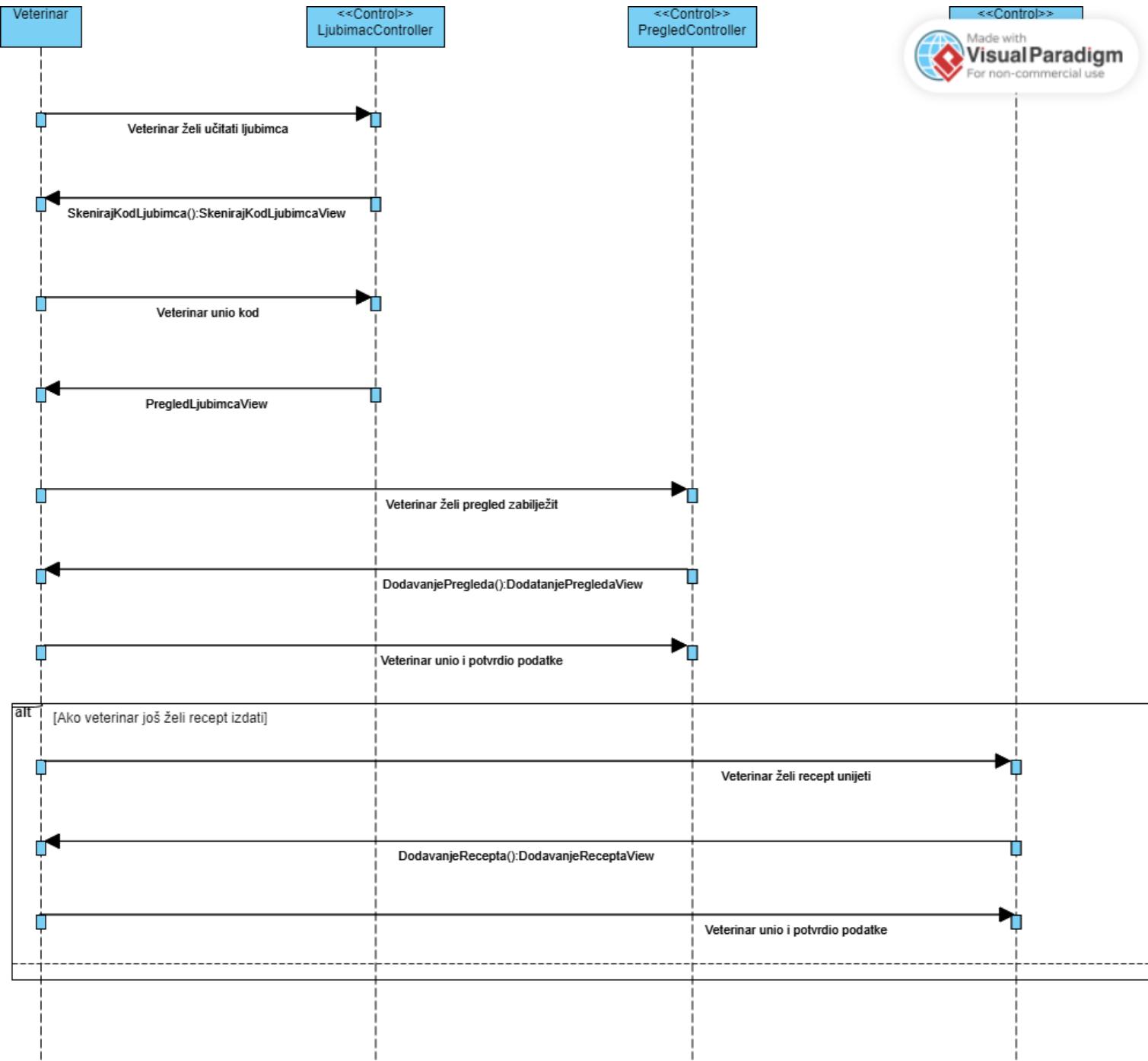
Klijenti ne treba da ovise o metodama koje neće upotrebljavati. Ovaj zathjev je ispunjen činjenicom da sve klase u sistemu obavljaju samo one aktivnosti koje su klijentu potrebne.

5. Dependency inversion principle – princip inverzije ovisnosti

Ovaj princip nam govori da ne treba ovisiti od konkretnih klasa, odnosno da prilikom nasljeđivanja bazna klasa bude apstraktna. U našem sistemu imamo jednu baznu klasu *Korisnik*, iz koje su izvedene klase *Apotekar*, *Veterinar* i *Vlasnik*. Kako je ova bazna klasa apstraktna tako je ispunjen i posljednji princip.



MVC Diagram



STRUKTURALNI PATERNI

Adapter pattern:

Da bi vlasnik mogao pratiti recepte na jednostavan način, trebao bi moći dobiti sortirane ili odabrane po ljubimcu koji ga zanima. Tu bi mogli koristiti adapter pattern da obavlja tu funkciju da ne bi mijenjali glavnu klasu Recept

Bridge pattern:

S obzirom da u našem slučaju kod prijava imamo više načina da se prijavimo u sistem, od skeniranja QR koda, ručne prijave, a i budućih mogućih načina na koje bi se prijavljivali, mogli bismo imati apstraktnu klasu PrijavaKorisnika koja će imati dalje mogućnosti kao što su QRLogin, RucniLogin, itd.

Facade pattern:

Korisnik koji je prijavljen kao veterinar/ka treba imati pristup i ljubimcima, pregledima, receptima. Svaku od tih klase možemo posmatrati kao poseban podsistem onoga što veterinar/ka može da radi. Tako bismo u klasi Facade mogli imati svaki od tih 3 podsistema, a svaki od tih podsistema ima svoje vlastite operacije. Sve one operacije koje veterinar/ka može da izvodi možemo da stavimo u klasu Facade jer u njoj možemo držati operacije sastavljene od različitih dijelova podsistema. Na taj način sakrivamo kompleksnost sistema i pružamo korisniku (veterinaru/ki) interfejs kojim on može da pristupa svakom od tih podsistema.

Proxy patern:

Proxy patern bi mogao biti realizovan na klasi Administrator, zato što toj klasi ne bi trebalo moći direktno pristupiti, a ona sama ima poprilično velike efekte na čitav sistem.

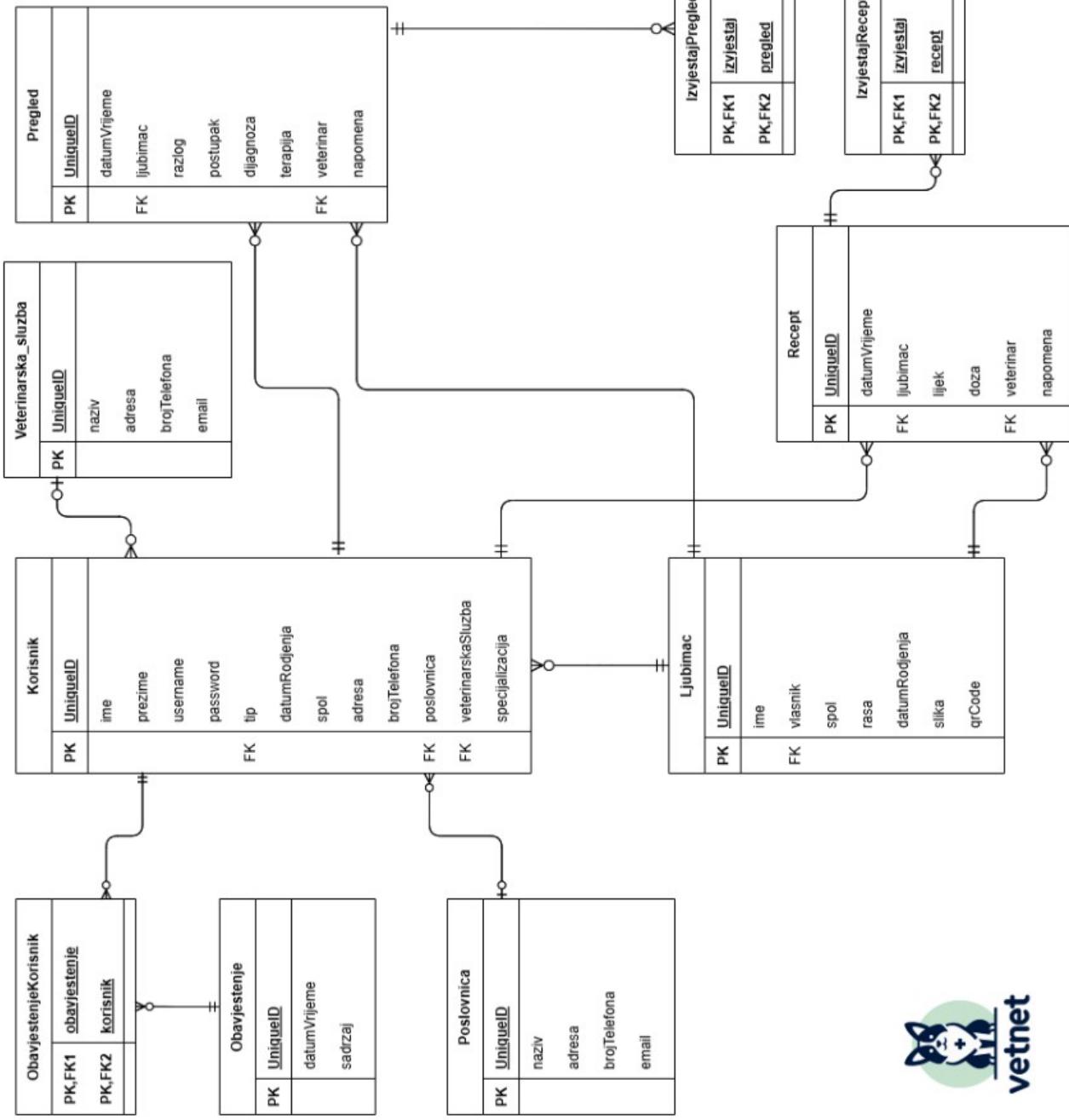
Decorator patern:

Pošto kod klase ljubimac dodajemo sliku, mogli bismo dodati interfejs ISlikaLjubimca koja bi imala metode kao što su postaviSliku(), izrežiSliku() i sl. Uglavnom da se slika može malo editovati prije nego što se postavi u klasi ljubimac.

Flyweight patern i Composite paterne paterne nemamo nigdje primijeniti.



Made with
Visual Paradigm
For non-commercial use



Entity Relationship Dijagram



Kreacijski paterni

Za ovaj sistem, implementirat će se *Singleton* i *Prototype* pattern. U nastavku data je primjena patterna ili ideja gdje bi se mogli implementirati.

1. Singleton

Ovaj pattern pogodno je koristiti kod držanja i obrade informacija trenutno prijavljenog korisnika. Kako samo jedan korisnik može biti prijavljen na uređaju u jednom trenutku, prirodno se nameće upotreba ovog patterna. Sve interakcije sistema sa informacijama o trenutno prijavljenom korisniku mogu se vršiti preko jedne instance klase *Korisnik*.

U slučaju da se korisnik promijeni, dovoljno je prepisati sadržaj objekta *trenutniKorisnik* koji može biti je tipa *Korisnik*.

2. Prototype

Prototype pattern podrazumijeva kloniranje - kreiranje kopije, nekog objekta. Prilikom izdvanja recepta, navodi se datum, ljubimac i veterinar. U slučaju da se izdaje više lijekova, svi recepti će imati iste vrijednosti pomenutih atributa. Pogodno je klonirati recept te prepisati samo preostala bolja. To bi značajno uštedilo na ručnom pisanju ovih stavki.

3. Builder

Builder pattern koristimo za sklapanje objekata složenijih klasa. Kako u planu za naš sistem nema složenijih klasa, nema potrebe za upotrebom ovog patterna.

Eventualno, kada bi sistem podržavao velik broj vrsta životinja sa velikim brojem njihovih rasa i podvrsta gdje bi postojala potreba za specijalizacijom veterinara. Pod tim okolnostima imalo bi smisla koristiti Builder pattern i to kod registracije veterinara. Prvi korak bi bio unošenje generalnih informacija dok u narednijim koracima se grade detaljne mogućnosti tog veterinare poput vrsta životinja koje liječi, koje zahvate radi nad kojim životnjama, koja prava ima da izdaje specifične lijekove i slično.

4. Factory Method

Factory Method pattern omogućuje instanciranje objekata neke super-klase ali dozvoljava podklasama da prave izmjene.

U našem slučaju, jedan način gdje se može iskoristiti je kod kreiranja nalaza obzirom da mogu postojati različiti nalazi kao što su redovni nalazi, urgrentni nalazi, nalazi prije putovanja i tako dalje.

5. Abstract Factory

Ovajim patternom se definišu koraci za kreiranje porodice sličnih objekata ali uz specifične razlike. Potencijalno se mogu raditi specijalizirane klinike ili veterinarske stanice koje se razlikuju po vrsti životine kojom se bave ili uslugama za tu životinju koju pružaju.

PATERNI PONAŠANJA

1. Strategy pattern

Strategy pattern koristimo onda kada želimo izdvojiti određene algoritme iz njihovih matičnih klasa u posebne klase. Pogodan je za korištenje kada postoje različiti algoritmi koji se mogu primijeniti na isti problem. On omogućava klijentu izbor jednog od algoritama iz familije dostupnih algoritama, dok su implementacije istih neovisne od klijenata koji ih koriste.

U našem slučaju, Strategy pattern bi se mogao iskoristiti kod implementacije pretrage korisnika kojima je potrebno poslati relevantno obavještenje putem dostupnih notification servisa. Ukoliko je to potrebno, moguće je vlasnike ljubimaca filtrirati po adresi stanovanja (geografski lokalizirana selekcija), rasi ljubimca (rasna selekcija), datumu zadnje vakcinacije (selekcija na osnovu recentnosti vakcinisanja) i slično.

2. State pattern

State pattern podrazumijeva promjenjiv način ponašanja objekta na osnovu njegovog trenutnog stanja. Pogodan je za korištenje ukoliko objekat može imati više stanja, te ukoliko često dolazi do promjene istih.

objektima tog tipa. Ljekarski recept može biti u stanju popunjavanja, izdat i poništen. Prema tome, stanja koja bi objekat ove klase mogao imati bi bila „pending“, „processed“ i „discarded“ ili tome slično. U ovisnosti od toga u kojem je stanju, recept može čekati na slanje, biti poslan (kada se može i ručno povući u slučaju greške, ili isprintati u fizičkom obliku), i poništen (kada vlasnik preuzme lijekove od apotekara).

3. Template Method pattern

Template method pattern omogućuje izdvajanje određenih koraka nekog algoritma u odvojene podklase. Ova praksa ne utiče na strukturu algoritma, već samo omogućuje implementaciju pojedinih njegovih dijelova na različite načine.

Ovaj pattern bi se u našem sistemu mogao koristiti kod implementacije Pregleda, gdje bi se za različite tipove pregleda tražili različiti podaci. Recimo, ukoliko je u pitanju obični pregled bez intervencije, mogli bi se tražiti samo osnovni podaci o ljubimcu, razlogu posjete, te napomeni ljekara. Ukoliko je pregled tipa operativnog zahvata, vakcinacije ili sličnog tretmana, trebaju se zatražiti podaci o tipu operativnog zahvata, vrsti vaccine koju je ljubimac primio i sl.

4. Observer pattern

Ovaj pattern karakteriše uspostava relacija između objekata tako da kada jedan objekat promijeni stanje, drugi povezani objekti se o tome obavještavaju.

Konkretno u našem sistemu je implementiran sistem slanja obavijesti koji funkcionira na način da ciljane grupe korisnika (vlasnika ljubimaca) dobijaju obavijesti o aktuelnim događajima, kako u samom sistemu, tako i aktuelnih veterinarskih događaja koji su relevantni datom korisniku. Recimo, tehničke informacije o dostupnosti servise bi se mogle slati e-mailom, kao i podaci o organiziranoj planskoj vakcinaciji ljubimaca na određenoj lokaciji ili dostupnosti / nedostupnosti lijekova koji su na aktivnom receptu za ljubimca nekog vlasnika.

5. Iterator pattern

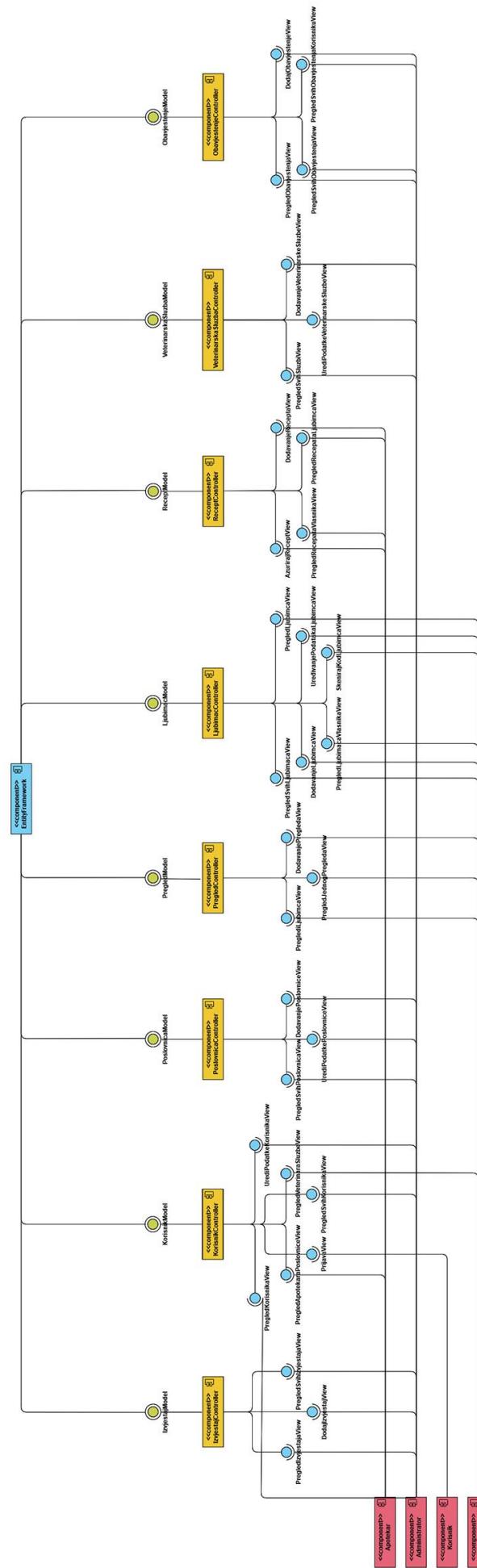
Princip iterator patterna se ogleda u omogućivanju pristupa elementima bez poznavanja kako je sama kolekcija istih struktuirana. Osnovna ideja iterator patterna je ekstrakcija navigacije kroz kolekciju u odvojene objekte nazvane iteratorima. Iteratori implementiraju različite algoritme navigacije kroz zajednički interface bez znanja o njihovoj temeljnoj implementaciji.

U našem sistemu se iterator pattern može koristiti prilikom pretrage korisnika kojima je potrebno poslati obavijest. Ukoliko se šalje obavijest za geografski lokaliziranu grupu korisnika potrebno je izdvojiti sve korisnike koji su tipa vlasnik, a zatim iteriranjem kroz dobijenu kolekciju izdvojiti one sa relevantnom geografskom lokacijom (recimo kod planske vakcinacije).

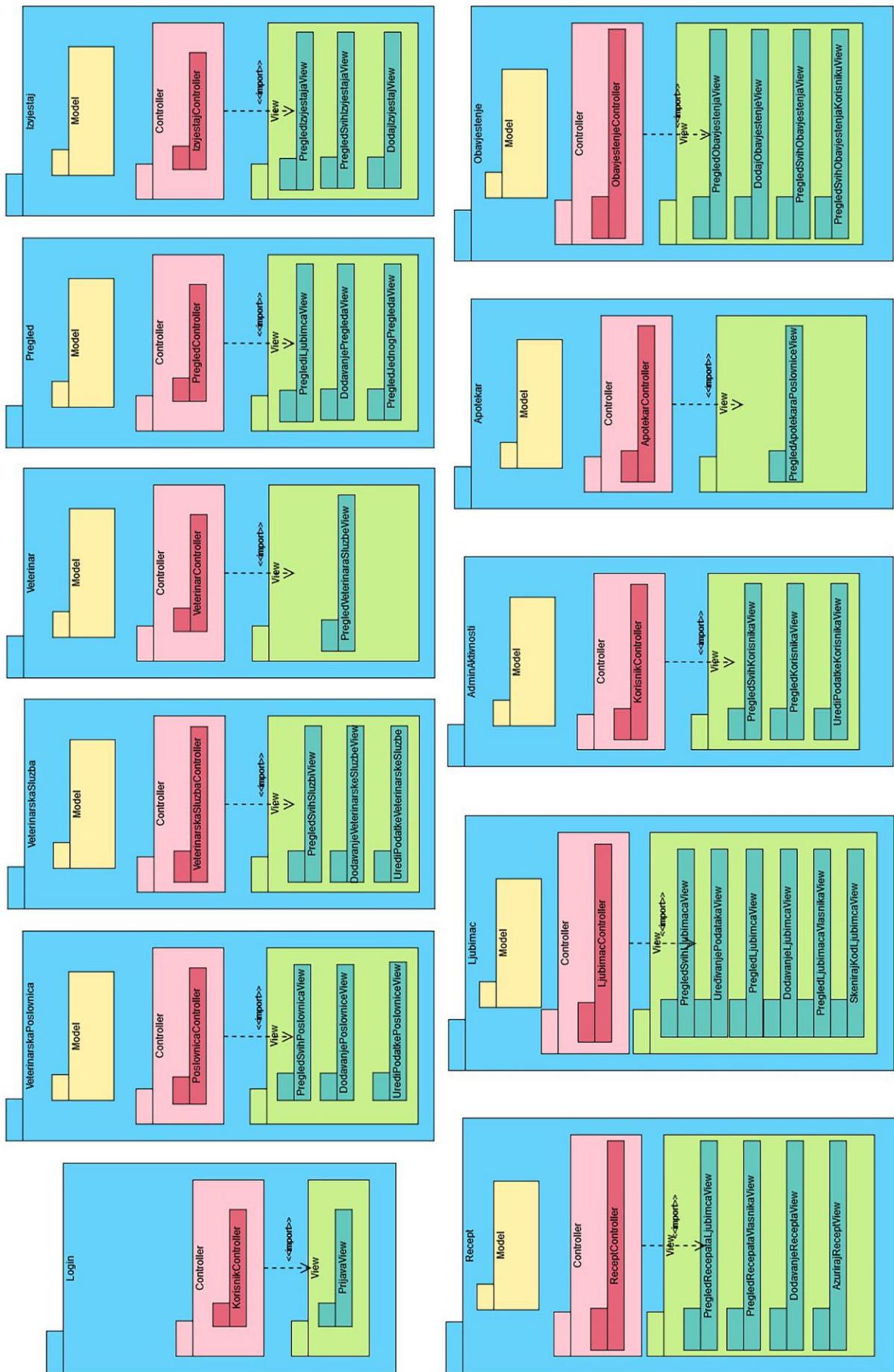
6. Command pattern

Command pattern se koristi za enkapsulaciju svih informacija potrebnih za odgođeno izvođenje akcije ili pokretanje događaja. Informacije uključuju naziv metode, objekat koji tu metodu posjeduje, kao i vrijednosti parametara metode.

U našem sistemu postoji više različitih načina da se podaci o vlasniku ili njegovom ljubimcu prikažu u sistemu. U ovisnosti od toga koji je način upotrebljen (skeniranje QR koda s kartice, RFID čipa ili unos ID-a ručnim putem), moguće je ovaj pattern iskoristiti za kreiranje zahtjeva za prikaz podataka, dok se informacije potrebne za prikaz korisnika dobavljaju ovisno o odabranom načinu unosa.

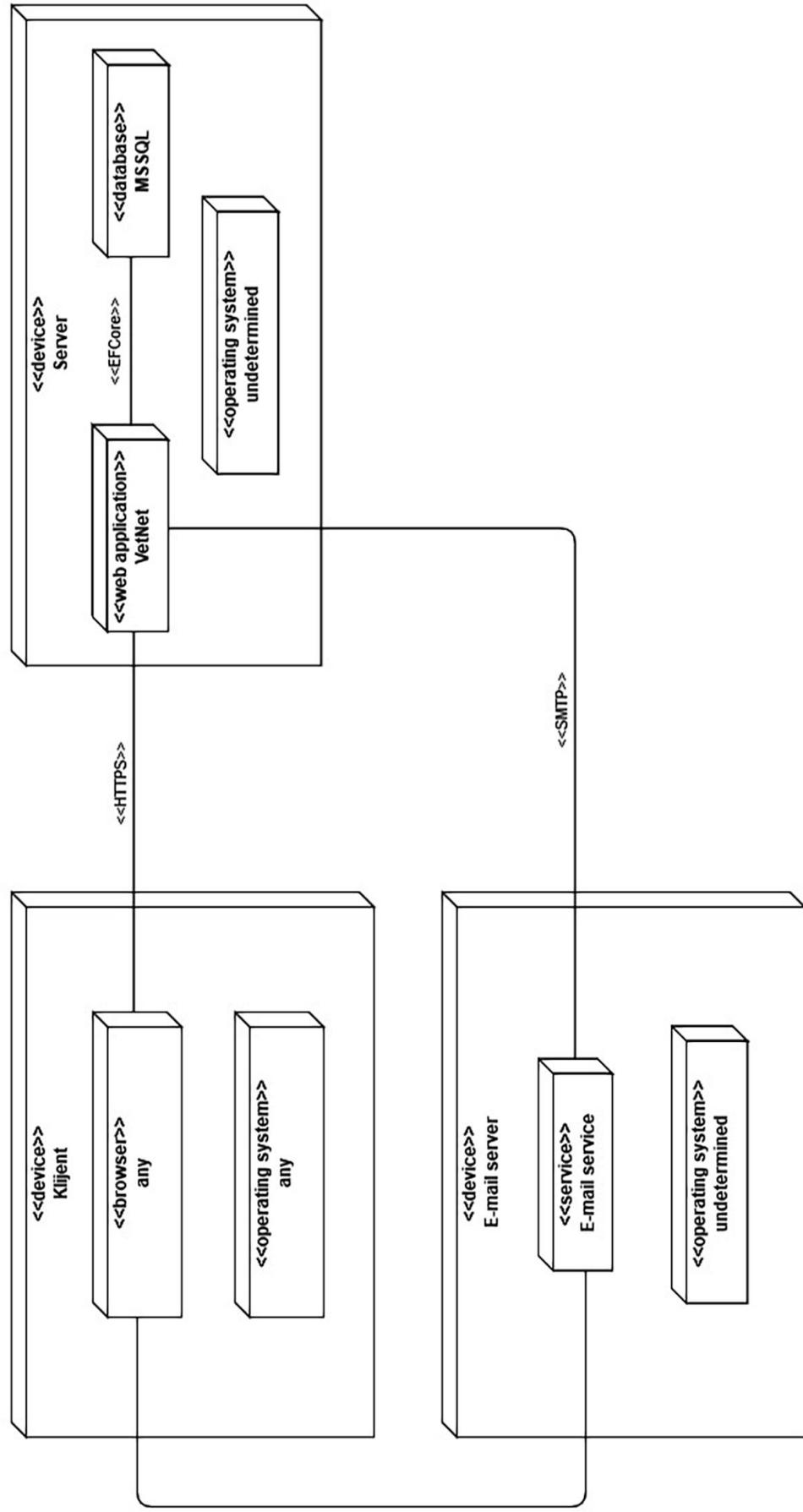


Dijagram komponenti





vetnet



Dijagram rasporedivanja