

Programsko inženjerstvo

Ak. god. 2022./2023.

Djeca za djecu

Dokumentacija, Rev. 2

Grupa: *Jamesove Obveznice*

Voditelj: *Dario Kiramarios*

Datum predaje: 13. 01. 2023.

Mentor/ica: *Martina Novak*

Sadržaj

1 Dnevnik promjena dokumentacije	3
2 Opis projektnog zadatka	6
3 Specifikacija programske potpore	12
3.1 Funkcionalni zahtjevi	12
3.1.1 Obrasci uporabe	15
3.1.2 Sekvencijski dijagrami	36
3.2 Ostali zahtjevi	40
4 Arhitektura i dizajn sustava	41
4.1 Baza podataka	44
4.1.1 Opis tablica	45
4.1.2 Dijagram baze podataka	50
4.2 Dijagram razreda	52
4.3 Dijagram stanja	61
4.4 Dijagram aktivnosti	63
4.5 Dijagram komponenti	66
5 Implementacija i korisničko sučelje	68
5.1 Korištene tehnologije i alati	68
5.2 Ispitivanje programskog rješenja	70
5.2.1 Ispitivanje komponenti	70
5.2.2 Ispitivanje sustava	74
5.3 Dijagram razmještaja	78
5.4 Upute za puštanje u pogon	80
5.4.1 Pristupanje Renderu	80
5.4.2 Backend deploy na Render	80
5.4.3 Frontend deploy na Render	84
6 Zaključak i budući rad	86

Popis literature	88
Indeks slika i dijagrama	90
Dodatak: Prikaz aktivnosti grupe	91

1. Dnevnik promjena dokumentacije

Rev.	Opis promjene/dodataka	Autori	Datum
0.1	Predložak prilagođen projektu	Dominik Jambrović	22.10.2022.
0.2	Dodan prvi dio opisa obrazaca uporabe, dodan dijagram obrazaca uporabe	Karla Kijac, Dominik Jambrović	22.10.2022.
0.3	Prepravljen dijagram obrazaca uporabe, dodani aktori, funkcionalni zahtjevi i ostali zahtjevi, dodani obrasci uporabe	Dominik Jambrović	26.10.2022.
0.4	Dodane sheme baze podataka i tablice baze podataka	Dominik Jambrović	31.10.2022.
0.5	Dodan opis projektnog zadatka	Karla Kijac	10.11.2022.
0.6	Izmjena opisa projektnog zadatka i obrazaca uporabe	Dominik Jambrović	10.11.2022.
0.7	Dodan opis tablica baza podataka	Dominik Jambrović	10.11.2022.
0.8	Dodani dijagrami razreda	Krešo Orešković, Dominik Jambrović	10.11.2022.

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

Rev.	Opis promjene/dodataka	Autori	Datum
0.9	Dodan opis arhitekture sustava	Dominik Jambrović	10.11.2022.
0.10	Dodani sekvencijski dijagrami	Karla Kijac, Dario Kiramarios	11.11.2022.
0.11	Izmjena obrazaca uporabe i funkcionalnih zahtjeva	Dominik Jambrović	14.11.2022.
0.12	Izmjena sekvencijskih dijagrama	Karla Kijac	14.11.2022.
0.13	Izmjena dijagrama razreda	Krešo Orešković	16.11.2022.
1.0	Verzija samo s bitnim dijelovima za 1. ciklus	Karla Kijac, Dominik Jambrović	16.11.2022.
1.1	Ispravak arhitekture i opisa dijagrama razreda	Dominik Jambrović	11.12.2022.
1.2	Dodavanje dijagrama stanja i opisa dijagrama stanja	Dominik Jambrović	5.1.2023.
1.3	Dodavanje dijagrama razmještaja i opisa dijagrama razmještaja	Dominik Jambrović	6.1.2023.
1.4	Dodatak implementacijskih dijagrama razreda	Dominik Jambrović	6.1.2023.
1.5	Dodatak opisa korištenih alata i tehnologija	Dominik Jambrović	6.1.2023.

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

Rev.	Opis promjene/dodatak	Autori	Datum
1.6	Dodatak uputa za deploy baze podataka i backenda	Dominik Jambrović	6.1.2023.
1.7	Dodatak uputa za deploy frontenda	Dominik Jambrović	7.1.2023
1.8	Dodatak druge verzije dijagrama stanja i opisa dijagrama stanja	Dominik Jambrović	9.1.2023
1.9	Dodatak opisa za testiranje komponenti	Dominik Jambrović	9.1.2023
1.10	Dodatak dijagrama aktivnosti i opisa dijagrama aktivnosti	Karla Kijac	9.1.2023
1.11	Dodatak opisa za testiranje sustava, manji prepravci	Dominik Jambrović	12.1.2023
1.12	Dodatak dijagrama komponenti i opisa dijagrama komponenti	Karla Kijac	12.1.2023
1.13	Dodatak zaključka, manji prepravci	Karla Kijac	12.1.2023
2.0	Finalna verzija dokumentacije	Karla Kijac, Dominik Jambrović	13.1.2023

2. Opis projektnog zadatka

Cilj projekta „Djeca za djecu“ je razviti web aplikaciju koja će omogućiti roditeljima, i onim budućim, da lakše doniraju i pronalaze donacije za svoju djecu.

U doba kada su već i osnovne namirnice preskupe i globalno zagađenje raste, „recikliranje“ predmeta postaje još važnije. Zašto bacati dobro očuvane stvari koje još mogu poslužiti nekome i spasiti njegov džep, a nemamo ih gdje čuvati dok ne zatrebaju kolegi/prijateljima/rodbini...

Ova aplikacija brzo i efikasno spaja donatora i primatelja donacije bez da korisnici moraju pretraživati bespuća interneta. Oglasi u aplikaciji prilagođavaju se profilu korisnika. Nakon što korisnik jednom upiše svoje podatke i podatke o svojoj djeci, automatski mu se prikazuju oglasi od interesa. Dodatno se mogu namještati filteri kategorija kod pregleda oglasa.

Doniranje, također, nikada nije bilo lakše. Nakon što korisnik objavi oglas za predmet, više se ne mora brinuti o njemu dok aplikacija sama ne pronađe savršenog primatelja. Tek kada predmet pronađe svog potencijalnog novog vlasnika zamišljeno je da primatelj kontaktira donatora izvan aplikacije te se dogovori za primopredaju. Nakon izvršene donacije donator treba samo potvrditi da je donirao korisniku upisivanjem mail adrese primatelja.

Sigurnost i kredibilitet aplikacije i oglasa važni su za iskustvo korisnika. Stoga je aplikacija zatvorena na registrirane korisnike što omogućava praćenje aktivnosti unutar aplikacije. Provode se provjere na razini korisnika i oglasa. Korisnici (ADMIN van aplikacije) se provjeravaju na temelju prijašnjih donacija ili postojećih zapisa o njima kako bi eliminirali moguće prevare ostalih korisnika aplikacije. Oglasi, pak, moraju biti opisani u skladu sa standardom aplikacije i predmeti ne stariji od preporučene starosti za određen predmet.

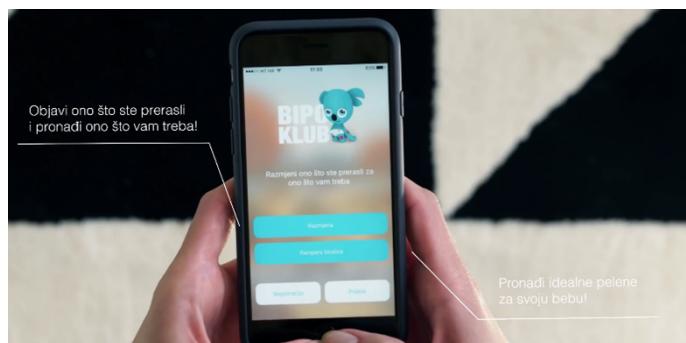
Istraživanjem hrvatskog „tržišta“ za sličnim aplikacijama pokazalo se da ne postoje aktivne aplikacije za doniranje i razmjenu dječjih stvari. U 2017. godini govorilo se o BIPO CLUB aplikaciji (poznatog BIPO branda za djecu, trgovačkog lanca BIPA) za razmjenu dječjih stvari koja trenutno više nije dostupna. Kako su odgovorili iz BIPO kluba, aplikacija se više ne koristi te su gašenjem iste spojili BIPO i BIPA kartice pa se više ne može aktivirati korisnički račun u aplikaciji.



Slika 2.1: Oglas za mobilnu aplikaciju BIPO CLUB

Za razliku od web aplikacije, vidimo da se ovdje radi o mobilnoj aplikaciji kompatibilnoj s IOS i Android uređajima, no slično našoj, postoje dvije funkcionalnosti dostupne korisniku - traženje i doniranje predmeta.

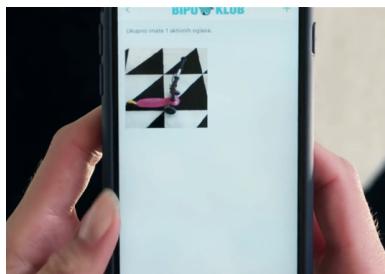
Registrirani korisnik može pregledavati oglase i primati donacije, ali i kreirati vlastite oglase i tim putem donirati dječje stvari koje mu više ne trebaju.



Slika 2.2: "Landing page" aplikacije BIPO CLUB

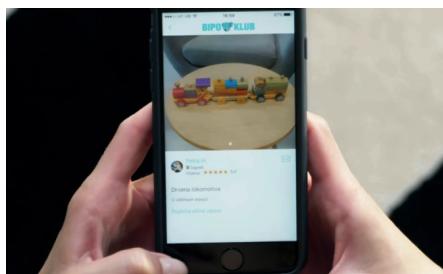
Slično aplikaciji koja će nastati u ovom projektu, aplikacija je zatvorena na registrirane korisnike, a na početnom zaslonu ima prijavu/registraciju korisnika.

U aplikaciji koju razvijamo neregistrirani korisnik će se jedino moći registrirati u sustav na početnom zaslonu, a registrirani prijaviti. Nakon registracije korisniku stiže mail potvrde te se ima pravo prijaviti sa svojim korisničkim računom. Ulaskom u aplikaciju korisnik dalje može birati hoće li uređivati/pregledavati svoj profil i/ili dodavati podatke o djeci koji se koriste za filtriranje preporučenog sadržaja. Primjerice, korisnik ima muško dijete u dobi od 3 godine - ako na svom profilu doda podatke o dobi i spolu svog djeteta, među preporučenim oglasima bit će samo predmeti za trogodišnjake.



Slika 2.3: Pregled vlastitih oglasa u aplikaciji BIPO CLUB

Aplikacija korisniku omogućava pregled vlastitih oglasa - ova funkcionalnost bit će implementirana i u našoj web aplikaciji. Naime, ako korisnik ima predmet koji želi donirati, može to učiniti kreiranjem oglasa. Oglas se prije objave „šalje“ administratoru na pregled i ukoliko dobije zeleno svjetlo, oglas se objavljuje. Ako oglas nije valjan(nedostaju podaci ili su neispravni), administrator ga vraća korisniku na doradu, a ima i mogućnost odbiti oglas.

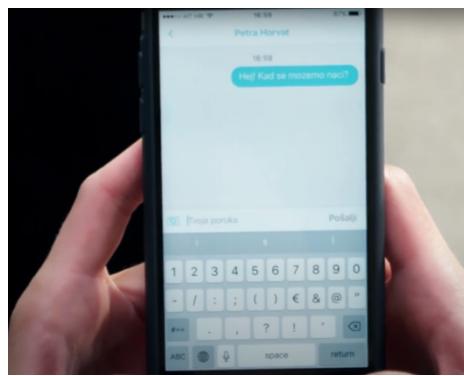


Slika 2.4: Pregled aktivnih oglasa u aplikaciji BIPO CLUB

Aplikacija omogućava pregled aktivnih oglasa- ova funkcionalnost bit će imple-

mentirana i u našoj web aplikaciji.

Na temelju podataka koje je korisnik unio o djeci, prikazuju mu se oglasi od najnovijih(do 3 dana starosti) prema najstarijima. Osim automatskog filtriranja koje rezultira preporučenim oglasima, korisnik će moći i ručno postaviti dodatne filtre za oglase (odabrati kategorije ili potkategorije). Uzmimo već navedeni primjer - osoba ima muško dijete u dobi od 3 godine i zanimaju je igračke za sina. U tom slučaju osoba će dodatno postaviti filter - odabrat će kategoriju dječjih igračaka, a iste će ponovo biti filtrirane prema dobi i spolu ako su oni definirani.

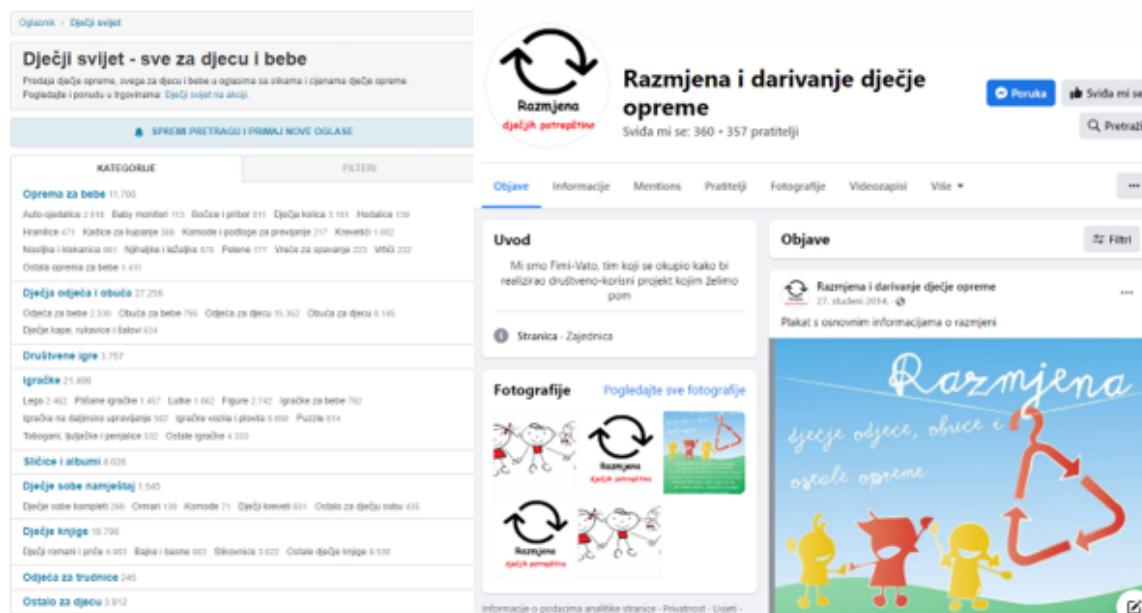


Slika 2.5: Chat room s donatorom u aplikaciji BIPO CLUB

Aplikacija nudi direktno javljanje korisniku koji je postavio oglas kroz aplikaciju. Web aplikacija koja se razvija neće imati tu opciju te će se korisnici međusobno morati dogovarati van aplikacije.

Po primitku donacije počinje odbrojavanje do trenutka kada će aplikacija ponuditi primatelju da isti predmet proslijedi dalje. Odbrojavanje se temelji na istraženim podacima o vremenu korištenja određenih predmeta. Primjerice, znamo da se kolica 3u1 koriste 1,5 godina i po isteku tog vremena korisniku koji je primio kolica u obliku donacije iskočit će prozor s upitom želi li možda proslijediti kolica dalje (donirati kreiranjem oglasa) ako su ista još u dobrom stanju.

Osim BIPO CLUB mobilne aplikacije, postoje razne grupe na Facebook-u za razmjenu i prodaju dječjih stvari, kao i odjeljak na Njuškalu koji nudi razmjenu i prodaju dječjih predmeta.



Slika 2.6: Njuškalo (lijevo) i Facebook grupa (desno)

Na kraju ovog projekta cilj je imati funkcionalnu web aplikaciju za doniranje i primanje predmeta putem oglasa. Kao što smo do sada vidjeli, takve aplikacije trenutno ne postoje i naša su očekivanja da bi aplikacija brzo postala popularna.

Cijeli proces izrade počeo je analizom zahtjeva naručitelja, razradom funkcionalnih zahtjeva i izradom UML dijagrama (Use case dijagrama i sekvenčnih dijagrama), kao i ER modela baze podataka koja će čuvati sve podatke o korisnicima i pratiti oglase i razna dopuštenja. Rad se nastavlja programiranjem backenda i frontenda te obuhvaća pisanje popratne dokumentacije kao i testiranje same aplikacije u više faza i puštanje u pogon. Na kraju nas još očekuje prezentiranje same aplikacije i demonstracija implementiranih funkcionalnosti.

Uvijek postoji mjesta za poboljšanje... Cijeli cilj ovog projekta je pomaganje zajednici tako da bi i nadogradnja aplikacije vodila k stvaranju sigurnije i ravno-pravne zajednice korisnika.

Za veće zadovoljstvo korisnika, ali i kako bi maksimizirali uspješnost doniranja i spajanja donatora s korisnikom koji je prima, sljedeća nadogradnja aplikacije bila bi usmjerenja prema sustavu ocjenjivanja donatora i primatelja donacija (dvo-smjerno ocjenjivanje), kao i sustavu za prijavljivanje korisnika koji krše opća pravila aplikacije i/ili sabotiraju uspješno doniranje predmeta. Nažalost, uvjek ima ljudi koji žele iskoristiti druge ljude ili tuđu dobrotu. Kako bi smanjili mogućnost da jedan korisnik primi velik broj donacija u jako kratkom vremenu ili nelogičnu kombinaciju donacija (prema podacima o djeci) te potom dobivene predmete pre-proda, uložili bi u razvijanje sustava za praćenje primljenih donacija.

Cilj je imati što sigurniju zajednicu donatora i primatelja donacija, kao i ulagati u sustav za poticanje kruženja predmeta i stalni priljev korisnika.

Aplikacija bi omogućavala nadogradnje, kao i prilagodbu na mobilne uređaje i tablete.

3. Specifikacija programske potpore

3.1 Funkcionalni zahtjevi

Dionici:

1. Vlasnik (naručitelj)
2. Administrator
3. Korisnici
4. Donatori
5. Razvojni tim

Aktori i njihovi funkcionalni zahtjevi:

1. Korisnik (inicijator) može:
 - (a) stvoriti novi korisnički račun za koji su mu potrebni ime, prezime, lokacija, mail adresa i lozinka
 - (b) prijaviti se u sustav koristeći mail adresu i lozinku
 - (c) odjaviti se iz sustava
 - (d) pregledavati sve aktivne oglase
 - (e) pregledavati preporučene oglase
 - (f) pregledavati detalje oglasa
 - (g) filtrirati sve aktivne ili preporučene oglase po kategorijama i potkategorijama
 - (h) pregledavati osobne podatke
 - (i) uređivati osobne podatke
 - (j) izbrisati račun
 - (k) pregledavati podatke o svojoj djeci i njihovim praćenim kategorijama
 - (l) uređivati podatke o svojoj djeci i njihovim praćenim kategorijama
 - (m) dodavati podatke o svojoj djeci i njihovim praćenim kategorijama
 - (n) brisati podatke o svojoj djeci

2. Donator (inicijator) može:

- (a) stvoriti novi korisnički račun za koji su mu potrebni ime, prezime, lokacija, mail adresa i lozinka
- (b) prijaviti se u sustav koristeći mail adresu i lozinku
- (c) odjaviti se iz sustava
- (d) pregledavati sve aktivne oglase
- (e) pregledavati preporučene oglase
- (f) pregledavati detalje oglasa
- (g) filtrirati sve aktivne ili preporučene oglase po kategorijama i potkategorijama
- (h) pregledavati osobne podatke
- (i) uređivati osobne podatke
- (j) izbrisati račun
- (k) pregledavati podatke o svojoj djeci i njihovim praćenim kategorijama
- (l) uređivati podatke o svojoj djeci i njihovim praćenim kategorijama
- (m) dodavati podatke o svojoj djeci i njihovim praćenim kategorijama
- (n) brisati podatke o svojoj djeci
- (o) prihvati ili odbiti ponovno doniranje primljene donacije
- (p) kreirati oglase za donacije
- (q) uređivati svoje oglase za donacije
- (r) zatvarati svoje oglase za donacije
- (s) pregledavati svoje oglase

3. Administrator (inicijator) može:

- (a) prijaviti se u sustav koristeći mail adresu i lozinku
- (b) odjaviti se iz sustava
- (c) pregledavati sve aktivne oglase
- (d) pregledavati detalje oglasa
- (e) pregledavati korisnike
- (f) dodijeliti korisnicima ovlast za kreiranje oglasa
- (g) oduzeti korisnicima ovlast za kreiranje oglasa
- (h) pregledavati neobjavljene oglase
- (i) prihvati i objaviti novostvorene oglase
- (j) poslati novostvorene oglase na doradu
- (k) odbiti novostvorene oglase
- (l) kreirati nove kategorije i potkategorije

4. Baza podataka (sudionik):

- (a) pohranjuje podatke o računima korisnika
- (b) pohranjuje podatke o djeci korisnika
- (c) pohranjuje sve podatke o donacijama

3.1.1 Obrasci uporabe

Opis obrazaca uporabe

UC1 - Registracija

- **Glavni sudionik:** Korisnik
- **Cilj:** Stvaranje računa
- **Sudionici:** Baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
 1. Korisnik je na početnoj stranici odabrao opciju za registraciju
 2. Sustav prikazuje polja za unos
 3. Korisnik unosi potrebne podatke
 4. Sustav validira unesene podatke
 5. Korisnik potvrđuje unos podataka
 6. Sustav ažurira podatke u bazi i na mail adresu korisnika šalje potvrdu o uspješnoj registraciji
- **Opis mogućih odstupanja:**
 - 1.a Korisnik ne odabire navedenu akciju - kraj scenarija
 - 3.a Korisnik odustaje od unosa u polja - kraj scenarija
 - 4.a Korisnik je unio podatke u neispravnom formatu
 1. Sustav upozorava korisika o neispravnom formatu podataka - scenarij nastavlja korakom 3
 - 6.a Sustav utvrđuje da unešeni podaci nisu ispravni
 1. Sustav upozorava korisnika o pogrešnom unosu podataka - scenarij nastavlja korakom 3
 - 6.b Sustav utvrđuje da već postoji korisnik sa istom mail adresom
 1. Sustav upozorava korisnika o zauzetoj mail adresi - scenarij nastavlja korakom 3
 - 6.c Sustav ne uspije ažurirati podatke u bazi
 1. Sustav upozorava korisnika o neuspješnoj pohrani podataka - scenarij nastavlja korakom 3

UC2 - Prijava u sustav

- **Glavni sudionik:** Korisnik ili administrator
- **Cilj:** Pristup početnoj stranici
- **Sudionici:** Baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
 1. Korisnik je na početnoj stranici odabrao opciju za prijavu
 2. Sustav prikazuje polja za unos
 3. Korisnik unosi potrebne podatke
 4. Sustav validira unesene podatke
 5. Korisnik potvrđuje unos podataka
 6. Sustav provjerava podatke u bazi i preusmjerava korisnika na početnu stranicu
- **Opis mogućih odstupanja:**
 - 1.a Korisnik ne odabire navedenu akciju - kraj scenarija
 - 3.a Korisnik odustaje od unosa u polja - kraj scenarija
 - 4.a Korisnik je unio podatake u neispravnom formatu
 1. Sustav upozorava korisnika o neispravnom formatu podataka - scenarij nastavlja korakom 3
 - 6.a Sustav utvrđuje da unešeni podaci nisu ispravni
 1. Sustav upozorava korisnika o pogrešnom unosu podataka - scenarij nastavlja korakom 3
 - 6.b Sustav ne uspije ažurirati podatke u bazi
 1. Sustav upozorava korisnika o neuspješnoj provjeri podataka - scenarij nastavlja korakom 3

UC3 - Odjava iz sustava

- **Glavni sudionik:** Korisnik ili administrator
- **Cilj:** Odjava korisnika
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav
- **Opis osnovnog tijeka:**
 1. Korisnik je odabrao opciju za odjavu
 2. Sustav pita korisnika želi li se sigurno odjaviti
 3. Korisnik potvrđuje da se želi odjaviti
 4. Sustav odjavljuje korisnika

- **Opis mogućih odstupanja:**
 - 1.a Korisnik ne odabire navedenu akciju - kraj scenarija
 - 3.a Korisnik odustaje od odjave - kraj scenarija
 - 4.a Sustav ne uspije odjaviti korisnika
 1. Sustav upozorava korisnika o neuspješnoj odjavi
 2. (a) Scenarij ponovno započinje
 - (b) Korisnik odustaje od odjave - kraj scenarija

UC4 - Pregled osobnih podataka

- **Glavni sudionik:** Korisnik
- **Cilj:** Pregled osobnih podataka unešenih u sustav
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav
- **Opis osnovnog tijeka:**
 1. Korisnik je odabrao opciju za prikaz osobnih podataka
 2. Sustav prikazuje korisnikove osobne podatke
 3. Korisnik pregledava osobne podatke
- **Opis mogućih odstupanja:**
 - 1.a Korisnik ne odabire navedenu akciju - kraj scenarija
 - 2.a Baza ne uspije poslati podatke o korisniku
 1. Korisnik dobiva obavijest o neuspješnom dohvatu podataka
 2. (a) Korisnik osvježava stranicu - scenarij nastavlja korakom 2
 - (b) Korisnik odustaje od pregleda osobnih podataka - kraj scenarija

UC5 - Uređivanje osobnih podataka

- **Glavni sudionik:** Korisnik
- **Cilj:** Promjena osobnih podataka unešenih u sustav
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav
- **Opis osnovnog tijeka:**
 1. Korisnik je odabrao opciju za uređivanje osobnih podataka
 2. Sustav prikazuje korisnikove osobne podatke
 3. Korisnik mijenja osobne podatke
 4. Sustav validira unesene podatke
 5. Korisnik potvrđuje promjene
 6. Sustav ažurira podatke u bazi i preusmjerava korisnika na početnu stranicu

- **Opis mogućih odstupanja:**

- 1.a Korisnik ne odabire navedenu akciju - kraj scenarija
- 3.a Korisnik odustaje od unosa promjena - kraj scenarija
- 4.a Korisnik je unio podatke u neispravnom formatu
 1. Sustav upozorava korisnika o neispravnom formatu podataka - scenarij nastavlja korakom 3
- 6.a Sustav utvrđuje da unešeni podaci nisu ispravni
 1. Sustav upozorava korisnika o pogrešnom unosu podataka - scenarij nastavlja korakom 3
- 6.b Sustav ne uspije ažurirati podatke u bazi
 1. Sustav upozorava korisnika o neuspješnoj pohrani podataka - scenarij nastavlja korakom 3

UC6 - Brisanje korisničkog računa

- **Glavni sudionik:** Korisnik

- **Cilj:** Brisanje računa

- **Sudionici:** Baza podataka

- **Preduvjet:** Korisnik je prijavljen u sustav

- **Opis osnovnog tijeka:**

1. Korisnik je odabrao opciju za brisanje računa
2. Sustav pita korisnika želi li sigurno obrisati svoj račun
3. Korisnik potvrđuje da želi obrisati račun
4. Sustav briše korisnikov račun

- **Opis mogućih odstupanja:**

- 1.a Korisnik ne odabire navedenu akciju - kraj scenarija
- 3.a Korisnik odustaje od brisanja računa - kraj scenarija
- 4.a Sustav ne uspije pobrisati korisnikov račun
 1. Sustav upozorava korisnika o neuspješnom brisanju
 2. (a) Scenarij ponovno započinje
 - (b) Korisnik odustaje od brisanja računa - kraj scenarija

UC7 - Pregled podataka o djeci

- **Glavni sudionik:** Korisnik

- **Cilj:** Pregled podataka o djetetu korisnika

- **Sudionici:** Baza podataka

- **Preduvjet:** Korisnik je prijavljen u sustav i u sustavu postoje podaci o barem jednom korisnikovom djetetu

- **Opis osnovnog tijeka:**
 1. Korisnik je odabrao opciju za prikaz podataka o jednom od svoje djece
 2. Sustav prikazuje podatke o odabranom korisnikovom djetetu
 3. Korisnik pregledava podatke o djetetu
- **Opis mogućih odstupanja:**
 - 1.a Korisnik ne odabire navedenu akciju - kraj scenarija
 - 2.a Baza ne uspije poslati podatke o djetetu
 1. Korisnik dobiva obavijest o neuspješnom dohvatu podataka
 2. (a) Korisnik osvježava stranicu - scenarij nastavlja korakom 2
(b) Korisnik odustaje od pregleda podataka o odabranom djetetu - kraj scenarija

UC8 - Uređivanje podataka o djeci

- **Glavni sudionik:** Korisnik
- **Cilj:** Promjena podataka o djetetu korisnika
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav i u sustavu postoje podaci o barem jednom korisnikovom djetetu
- **Opis osnovnog tijeka:**
 1. Korisnik je odabrao opciju za uređivanje podataka o jednom od svoje djece
 2. Sustav prikazuje podatke o odabranom korisnikovom djetetu
 3. Korisnik mijenja podatke o djetetu
 4. Sustav validira unesene podatke
 5. Korisnik potvrđuje promjene
 6. Sustav ažurira podatke u bazi i preusmjerava korisnika na početnu stranicu

- **Opis mogućih odstupanja:**

- 1.a Korisnik ne odabire navedenu akciju - kraj scenarija
- 3.a Korisnik odustaje od unosa promjena - kraj scenarija
- 4.a Korisnik je unio podatake u neispravnom formatu
 1. Sustav upozorava korisnika o neispravnom formatu podataka - scenarij nastavlja korakom 3
- 6.a Sustav utvrđuje da unešeni podaci nisu ispravni
 1. Sustav upozorava korisnika o pogrešnom unosu podataka - scenarij nastavlja korakom 3
- 6.b Sustav ne uspije ažurirati podatke u bazi
 1. Sustav upozorava korisnika o neuspješnoj pohrani podataka - scenarij nastavlja korakom 3

UC9 - Dodavanje podataka o djeci

- **Glavni sudionik:** Korisnik

- **Cilj:** Dodavanje podataka o djetetu korisnika

- **Sudionici:** Baza podataka

- **Preduvjet:** Korisnik je prijavljen u sustav

- **Opis osnovnog tijeka:**

1. Korisnik je odabrao opciju za dodavanje podataka o jednom od svoje djece
2. Sustav prikazuje polja za unos
3. Korisnik unosi potrebne podatke
4. Sustav validira unesene podatke
5. Korisnik potvrđuje unos podataka
6. Sustav ažurira podatke u bazi i preusmjerava korisnika na početnu stranicu

- **Opis mogućih odstupanja:**

- 1.a Korisnik ne odabire navedenu akciju - kraj scenarija
- 3.a Korisnik odustaje od unosa u polja - kraj scenarija
- 4.a Korisnik je unio podatke u neispravnom formatu
 1. Sustav upozorava korisnika o neispravnom formatu podataka - scenarij nastavlja korakom 3
- 6.a Sustav utvrđuje da unešeni podaci nisu ispravni
 1. Sustav upozorava korisnika o pogrešnom unosu podataka - scenarij nastavlja korakom 3
- 6.b Sustav ne uspije ažurirati podatke u bazi
 1. Sustav upozorava korisnika o neuspješnoj pohrani podataka - scenarij nastavlja korakom 3

UC10 - Brisanje podataka o djeci

- **Glavni sudionik:** Korisnik

- **Cilj:** Brisanje podataka o djetetu korisnika

- **Sudionici:** Baza podataka

- **Preduvjet:** Korisnik je prijavljen u sustav i u sustavu postoje podaci o barem jednom korisnikovom djetetu

- **Opis osnovnog tijeka:**

1. Korisnik je odabrao opciju za brisanje podataka o jednom od svoje djece
2. Sustav pita korisnika želi li sigurno obrisati podatke o odabranome djetu
3. Korisnik potvrđuje da želi obrisati podatke o odabranome djetu
4. Sustav briše podatke o odabranome djetu

- **Opis mogućih odstupanja:**

- 1.a Korisnik ne odabire navedenu akciju - kraj scenarija
- 3.a Korisnik odustaje od brisanja podataka o odabranome djetu - kraj scenarija
- 4.a Sustav ne uspije obrisati podatke o odabranome djetu
 1. Sustav upozorava korisnika o neuspješnom brisanju
 2. (a) Scenarij ponovno započinje
 - (b) Korisnik odustaje od brisanja podataka - kraj scenarija

UC11 - Pregled aktivnih oglasa

- **Glavni sudionik:** Korisnik ili administrator
- **Cilj:** Pregled svih aktivnih oglasa
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav
- **Opis osnovnog tijeka:**
 1. Korisnik je otišao na početnu stranicu
 2. Sustav prikazuje aktivne oglase
 3. Korisnik pregledava oglase
- **Opis mogućih odstupanja:**
 - 1.a Korisnik ne odlazi na početnu stranicu - kraj scenarija
 - 2.a Baza ne uspije poslati podatke o aktivnim oglasima
 1. Korisnik dobiva obavijest o neuspješnom dohvatu podataka
 2. (a) Korisnik osvježava stranicu - scenarij nastavlja korakom 2
(b) Korisnik odustaje od pregleda aktivnih oglasa - kraj scenarija
 - 2.b Ne postoji nijedan aktivan oglas
 1. Korisnik dobiva obavijest o nepostojanju aktivnih oglasa
 2. Korisnik odustaje od pregleda aktivnih oglasa - kraj scenarija

UC12 - Pregled preporučenih oglasa

- **Glavni sudionik:** Korisnik
- **Cilj:** Pregled oglasa personaliziranih korisniku
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav
- **Opis osnovnog tijeka:**
 1. Korisnik je odabrao opciju za prikaz preporučenih oglasa
 2. Sustav prikazuje preporučene oglase
 3. Korisnik pregledava oglase

- **Opis mogućih odstupanja:**

- 1.a Korisnik ne odabire navedenu akciju - kraj scenarija
- 2.a Baza ne uspije poslati podatke o preporučenim oglasima
 1. Korisnik dobiva obavijest o neuspješnom dohvatu podataka
 2. (a) Korisnik osvježava stranicu - scenarij nastavlja korakom 2
(b) Korisnik odustaje od pregleda preporučenih oglasa - kraj scenarija
- 2.b Ne postoje oglasi koji odgovaraju korisnikovim željama
 1. Korisnik dobiva obavijest o manjku odgovarajućih oglasa
 2. Korisnik odustaje od pregleda preporučenih oglasa - kraj scenarija

UC13 - Filtriranje oglasa

- **Glavni sudionik:** Korisnik

- **Cilj:** Pregled oglasa iz određene kategorije/potkategorije

- **Sudionici:** Baza podataka

- **Preduvjet:** Korisnik je prijavljen u sustav

- **Opis osnovnog tijeka:**

1. Korisnik je odabrao opciju za filtriranje aktivnih ili preporučenih oglasa po određenom parametru
2. Sustav prikazuje oglase koji odgovaraju odabranim kategorijama i/ili potkategorijama
3. Korisnik pregledava oglase

- **Opis mogućih odstupanja:**

- 1.a Korisnik ne odabire navedenu akciju - kraj scenarija
- 2.a Baza ne uspije poslati podatke o odgovarajućim oglasima
 1. Korisnik dobiva obavijest o neuspješnom dohvatu podataka
 2. (a) Scenarij ponovno započinje
(b) Korisnik odustaje od filtriranja - kraj scenarija
- 2.b Ne postoje oglasi koji odgovaraju zadanim opcijama filtriranja
 1. Korisnik dobiva obavijest o manjku odgovarajućih oglasa
 2. (a) Korisnik mijenja opcije filtriranja - scenarij nastavlja korakom 2
(b) Korisnik odustaje od filtriranja - kraj scenarija

UC14 - Pregled detalja o oglasu

- **Glavni sudionik:** Korisnik ili administrator
- **Cilj:** Pregled detalja određene donacije
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav i u sustavu postoje podaci o barem jednom oglasu
- **Opis osnovnog tijeka:**
 1. Korisnik je odabrao opciju za prikaz detalja o određenoj donaciji
 2. Sustav prikazuje detalje o izabranoj donaciji
 3. Korisnik pregledava detaljne informacije o donaciji
- **Opis mogućih odstupanja:**
 - 1.a Korisnik ne odabire navedenu akciju - kraj scenarija
 - 2.a Baza ne uspije poslati podatke o izabranoj donaciji
 1. Korisnik dobiva obavijest o neuspješnom dohvatu podataka
 2. (a) Korisnik osvježava stranicu - scenarij nastavlja korakom 2
(b) Korisnik odustaje od pregleda detalja o oglasu - kraj scenarija

UC15 - Poticanje donacija

- **Glavni sudionik:** Donator
- **Cilj:** Podsjećanje donatora o mogućnosti doniranja primljenog predmeta
- **Sudionici:** Baza podataka
- **Preduvjet:** Donator je prijavljen u sustav i ima primljene donacije kojima je isteklo standardno vrijeme korištenja
- **Opis osnovnog tijeka:**
 1. Sustav donatoru šalje obavijest o mogućnosti doniranja primljenog predmeta
 2. Donator prihvata mogućnost doniranja predmeta i biva preusmjeren na kreiranje oglasa
- **Opis mogućih odstupanja:**
 - 2.a Donator odbija mogućnost doniranja predmeta
 1. Obavijest o mogućnosti doniranja se više ne prikazuje - kraj scenarija

UC16 - Kreiranje oglasa

- **Glavni sudionik:** Donator
- **Cilj:** Stvaranje nove donacije
- **Sudionici:** Baza podataka
- **Preduvjet:** Donator je prijavljen u sustav
- **Opis osnovnog tijeka:**
 1. Donator je odabrao opciju za stvaranje nove donacije
 2. Sustav prikazuje polja za unos
 3. Donator unosi potrebne podatke
 4. Sustav validira unesene podatke
 5. Donator potvrđuje unos podataka
 6. Sustav ažurira podatke u bazi i preusmjerava donatora na početnu stranicu
- **Opis mogućih odstupanja:**
 - 1.a Donator ne odabire navedenu akciju - kraj scenarija
 - 3.a Donator odustaje od unosa u polja - kraj scenarija
 - 4.a Donator je unio podatke u neispravnom formatu
 1. Sustav upozorava donatora o neispravnom formatu podataka - scenarij nastavlja korakom 3
 - 6.a Sustav utvrđuje da unešeni podaci nisu ispravni
 1. Sustav upozorava donatora o pogrešnom unosu podataka - scenarij nastavlja korakom 3
 - 6.b Sustav ne uspije ažurirati podatke u bazi
 1. Sustav upozorava donatora o neuspješnoj pohrani podataka - scenarij nastavlja korakom 3

UC17 - Uređivanje oglasa

- **Glavni sudionik:** Donator
- **Cilj:** Izmjena podataka o donaciji
- **Sudionici:** Baza podataka
- **Preduvjet:** Donator je prijavljen u sustav i u sustavu postoje podaci o barem jednom oglasu koji korisnik treba doraditi

- **Opis osnovnog tijeka:**

1. Donator odabire oglas koji želi uređivati
2. Sustav prikazuje oglas
3. Donator odabere opciju za uređivanje oglasa
4. Sustav prikazuje podatke o donaciji
5. Donator mijenja podatke o donaciji
6. Sustav validira unesene podatke
7. Donator potvrđuje promjene
8. Sustav ažurira podatke u bazi i preusmjerava donatora na početnu stranicu

- **Opis mogućih odstupanja:**

- 1.a Donator ne odabire navedenu akciju - kraj scenarija
- 2.a Baza ne uspije poslati podatke o izabranom oglasu
 1. Donator dobiva obavijest o neuspješnom dohvatu podataka
 2. (a) Donator osvježava stranicu - scenarij nastavlja korakom 2
(b) Donator odustaje od uređivanja oglasa - kraj scenarija
- 5.a Donator odustaje od unosa promjena - kraj scenarija
- 6.a Donator je unio podatake u neispravnom formatu
 1. Sustav upozorava donatora o neispravnom formatu podataka - scenarij nastavlja korakom 5
- 8.a Sustav utvrđuje da unešeni podaci nisu ispravni
 1. Sustav upozorava donatora o pogrešnom unosu podataka - scenarij nastavlja korakom 5
- 8.b Sustav ne uspije ažurirati podatke u bazi
 1. Sustav upozorava donatora o neuspješnoj pohrani podataka - scenarij nastavlja korakom 5

UC18 - Zatvaranje oglasa

- **Glavni sudionik:** Donator
- **Cilj:** Zatvaranje trenutno aktivnog oglasa
- **Sudionici:** Baza podataka
- **Preduvjet:** Donator je prijavljen u sustav i u sustavu postoje podaci o barem jednom aktivnom oglasu korisnika

- **Opis osnovnog tijeka:**

1. Donator odabire oglas koji želi zatvoriti
2. Sustav prikazuje oglas
3. Donator odabere opciju za zatvaranje oglasa
4. Sustav pita donatora želi li sigurno zatvoriti odabrani oglas
5. Donator potvrđuje da želi zatvoriti oglas
6. Sustav ažurira podatke u bazi i preusmjerava donatora na početnu stranicu

- **Opis mogućih odstupanja:**

- 1.a Donator ne odabire navedenu akciju - kraj scenarija
- 2.a Baza ne uspije poslati podatke o izabranom oglasu
 1. Donator dobiva obavijest o neuspješnom dohvatu podataka
 2. (a) Donator osvježava stranicu - scenarij nastavlja korakom 2
(b) Donator odustaje od zatvaranja oglasa - kraj scenarija
- 3.a Donator odustaje od zatvaranja odabranog oglasa
- 4.a Sustav ne uspije ažurirati podatke u bazi
 1. Sustav upozorava donatora o neuspješnoj pohrani podataka
 2. (a) Scenarij ponovno započinje
(b) Donator odustaje od zatvaranja oglasa - kraj scenarija

UC19 - Dodjela dozvole

- **Glavni sudionik:** Administrator
- **Cilj:** Dodjela dozvole za doniranje odabranome korisniku
- **Sudionici:** Baza podataka
- **Preduvjet:** Administrator je prijavljen u sustav i postoji barem jedan korisnik kojemu se može dodijeliti dozvola
- **Opis osnovnog tijeka:**
 1. Administrator je odabrao opciju za dodjelu dozvole za doniranje odabranome korisniku
 2. Sustav pita administratora želi li sigurno dodijeliti dozvolu za doniranje odabranome korisniku
 3. Administrator potvrđuje da želi dodijeliti dozvolu
 4. Sustav ažurira podatke u bazi i preusmjerava administratora na početnu stranicu

- **Opis mogućih odstupanja:**

- 1.a Administrator ne odabire navedenu akciju - kraj scenarija
- 3.a Administrator odustaje od dodjele dozvole
- 4.a Sustav ne uspije ažurirati podatke u bazi
 1. Sustav upozorava administratora o neuspješnoj pohrani podataka
 2. (a) Scenarij ponovno započinje
 - (b) Administrator odustaje od dodjele dozvole odabranome korisniku - kraj scenarija

UC20 - Uklanjanje dozvole

- **Glavni sudionik:** Administrator

- **Cilj:** Uklanjanje dozvole za doniranje odabranome korisniku

- **Sudionici:** Baza podataka

- **Preduvjet:** Administrator je prijavljen u sustav i postoji barem jedan korisnik kojemu se može ukloniti dozvola

- **Opis osnovnog tijeka:**

1. Administrator je odabrao opciju za uklanjanje dozvole za doniranje odabranome korisniku
2. Sustav pita administratora želi li sigurno ukloniti dozvolu za doniranje odabranome korisniku
3. Administrator potvrđuje da želi ukloniti dozvolu
4. Sustav ažurira podatke u bazi i preusmjerava administratora na početnu stranicu

- **Opis mogućih odstupanja:**

- 1.a Administrator ne odabire navedenu akciju - kraj scenarija
- 3.a Administrator odustaje od uklanjanja dozvole
- 4.a Sustav ne uspije ažurirati podatke u bazi
 1. Sustav upozorava administratora o neuspješnoj pohrani podataka
 2. (a) Scenarij ponovno započinje
 - (b) Administrator odustaje od uklanjanja dozvole odabranome korisniku - kraj scenarija

UC21 - Pregled neobjavljenih oglasa

- **Glavni sudionik:** Administrator
- **Cilj:** Pregled svih neobjavljenih oglasa
- **Sudionici:** Baza podataka
- **Preduvjet:** Administrator je prijavljen u sustav
- **Opis osnovnog tijeka:**
 1. Administrator je odabrao opciju za prikaz neobjavljenih oglasa
 2. Sustav prikazuje neobjavljene oglase
 3. Administrator pregledava oglase
- **Opis mogućih odstupanja:**
 - 1.a Administrator ne odabire navedenu akciju - kraj scenarija
 - 2.a Baza ne uspije poslati podatke o neobjavljenim oglasima
 1. Administrator dobiva obavijest o neuspješnom dohvatu podataka
 2. (a) Administrator osvježava stranicu - scenarij nastavlja korakom 2
(b) Administrator odustaje od pregleda neobjavljenih oglasa - kraj scenarija
 - 2.b Ne postoji nijedan neobjavljeni oglas
 1. Administrator dobiva obavijest o nepostojanju neobjavljenih oglasa
 2. Administrator odustaje od pregleda neobjavljenih oglasa - kraj scenarija

UC22 - Odobravanje oglasa

- **Glavni sudionik:** Administrator
- **Cilj:** Odobravanje i objava donacije
- **Sudionici:** Baza podataka
- **Preduvjet:** Administrator je prijavljen u sustav i barem jedan oglas čeka odobrenje
- **Opis osnovnog tijeka:**
 1. Administrator odabire oglas koji čeka odobrenje
 2. Sustav prikazuje oglas
 3. Administrator provjerava oglas i odabire opciju za odobravanje oglasa
 4. Sustav pita administratora želi li sigurno odobriti navedeni oglas
 5. Administrator potvrđuje da želi odobriti oglas
 6. Sustav ažurira podatke u bazi, objavljuje oglas i preusmjerava administratora na početnu stranicu

- **Opis mogućih odstupanja:**

- 2.a Baza ne uspije poslati podatke o izabranom oglasu
 1. Administrator dobiva obavijest o neuspješnom dohvatu podataka
 2. (a) Administrator osvježava stranicu - scenarij nastavlja korakom 2
(b) Administrator odustaje od odobravanja oglasa - kraj scenarija
- 5.a Administrator odustaje od odobravanja navedenog oglasa
- 6.a Sustav ne uspije ažurirati podatke u bazi
 1. Sustav upozorava administratora o neuspješnoj pohrani podataka
 2. (a) Scenarij ponovno započinje
(b) Administrator odustaje od odobravanja oglasa - kraj scenarija

UC23 - Slanje oglasa na doradu

- **Glavni sudionik:** Administrator
- **Cilj:** Slanje oglasa natrag donatoru na doradu
- **Sudionici:** Baza podataka
- **Preduvjet:** Administrator je prijavljen u sustav i barem jedan oglas čeka odo- brenje
- **Opis osnovnog tijeka:**
 1. Administrator odabire oglas koji čeka odobrenje
 2. Sustav prikazuje oglas
 3. Administrator provjerava oglas i odabire opciju za slanje oglasa na do- radu
 4. Sustav pita administratora želi li sigurno poslati navedeni oglas na do- radu
 5. Administrator potvrđuje da želi poslati oglas na doradu
 6. Sustav ažurira podatke u bazi i preusmjerava administratora na početnu stranicu

- **Opis mogućih odstupanja:**

- 2.a Baza ne uspije poslati podatke o izabranom oglasu
 1. Administrator dobiva obavijest o neuspješnom dohvatu podataka
 2. (a) Administrator osvježava stranicu - scenarij nastavlja korakom 2
(b) Administrator odustaje od slanja oglasa na doradu - kraj scenerija
- 5.a Administrator odustaje od slanja navedenog oglasa na doradu
- 6.a Sustav ne uspije ažurirati podatke u bazi
 1. Sustav upozorava administratora o neuspješnoj pohrani podataka
 2. (a) Scenarij ponovno započinje
(b) Administrator odustaje od slanja oglasa na doradu - kraj scenerija

UC24 - Odbijanje oglasa

- **Glavni sudionik:** Administrator
- **Cilj:** Odbijanje oglasa
- **Sudionici:** Baza podataka
- **Preduvjet:** Administrator je prijavljen u sustav i barem jedan oglas čeka odobrenje
- **Opis osnovnog tijeka:**
 1. Administrator odabire oglas koji čeka odobrenje
 2. Sustav prikazuje oglas
 3. Administrator provjerava oglas i odabire opciju za odbijanje oglasa
 4. Sustav pita administratora želi li sigurno odbiti navedeni oglas
 5. Administrator potvrđuje da želi odbiti navedeni oglas
 6. Sustav ažurira podatke u bazi i preusmjerava administratora na početnu stranicu

- **Opis mogućih odstupanja:**

- 2.a Baza ne uspije poslati podatke o izabranom oglasu
 1. Administrator dobiva obavijest o neuspješnom dohvatu podataka
 2. (a) Administrator osvježava stranicu - scenarij nastavlja korakom 2
(b) Administrator odustaje od odbijanja oglasa - kraj scenarija
- 5.a Administrator odustaje od odbijanja navedenog oglasa
- 6.a Sustav ne uspije ažurirati podatke u bazi
 1. Sustav upozorava administratora o neuspješnoj pohrani podataka
 2. (a) Scenarij ponovno započinje
(b) Administrator odustaje od odbijanja oglasa - kraj scenarija

UC25 - Dodavanje kategorija

- **Glavni sudionik:** Administrator

- **Cilj:** Dodavanje nove kategorije u sustav

- **Sudionici:** Baza podataka

- **Preduvjet:** Administrator je prijavljen u sustav

- **Opis osnovnog tijeka:**

1. Administrator je odabrao opciju za dodavanje kategorija
2. Sustav prikazuje polja za unos
3. Administrator unosi potrebne podatke
4. Sustav validira unesene podatke
5. Administrator potvrđuje unos podataka
6. Sustav administratora preusmjeri na početnu stranicu

- **Opis mogućih odstupanja:**

- 1.a Administrator ne odabire navedenu akciju - kraj scenarija
- 3.a Administrator odustaje od unosa u polja - kraj scenarija
- 4.a Administrator je unio podatke u neispravnom formatu
 1. Sustav upozorava administratora o neispravnom formatu podataka - scenarij nastavlja korakom 3
- 6.a Sustav utvrđuje da unešeni podaci nisu ispravni
 1. Sustav upozorava administratora o pogrešnom unosu podataka - scenarij nastavlja korakom 3
- 6.b Sustav ne uspije ažurirati podatke u bazi
 1. Sustav upozorava administratora o neuspješnoj pohrani podataka - scenarij nastavlja korakom 3

UC26 - Dodavanje potkategorija

- **Glavni sudionik:** Administrator
- **Cilj:** Dodavanje nove potkategorije u sustav
- **Sudionici:** Baza podataka
- **Preduvjet:** Administrator je prijavljen u sustav
- **Opis osnovnog tijeka:**
 1. Administrator je odabrao opciju za dodavanje potkategorija
 2. Sustav prikazuje polja za unos
 3. Administrator unosi potrebne podatke
 4. Sustav validira unesene podatke
 5. Administrator potvrđuje unos podataka
 6. Sustav administratora preusmjeri na početnu stranicu
- **Opis mogućih odstupanja:**
 - 1.a Administrator ne odabire navedenu akciju - kraj scenarija
 - 3.a Administrator odustaje od unosa u polja - kraj scenarija
 - 4.a Administrator je unio podatke u neispravnom formatu
 1. Sustav upozorava administratora o neispravnom formatu podataka - scenarij nastavlja korakom 3
 - 6.a Sustav utvrđuje da unešeni podaci nisu ispravni
 1. Sustav upozorava administratora o pogrešnom unosu podataka - scenarij nastavlja korakom 3
 - 6.b Sustav ne uspije ažurirati podatke u bazi
 1. Sustav upozorava administratora o neuspješnoj pohrani podataka - scenarij nastavlja korakom 3

UC27 - Pregled korisnika

- **Glavni sudionik:** Administrator
- **Cilj:** Pregled svih korisnika
- **Sudionici:** Baza podataka
- **Preduvjet:** Administrator je prijavljen u sustav
- **Opis osnovnog tijeka:**
 1. Administrator je odabrao opciju za prikaz svih korisnika
 2. Sustav prikazuje sve korisnike
 3. Administrator pregledava korisnike

- **Opis mogućih odstupanja:**

- 1.a Administrator ne odabire navedenu akciju - kraj scenarija
- 2.a Baza ne uspije poslati podatke o korisnicima
 1. Administrator dobiva obavijest o neuspješnom dohvatu podataka o korisnicima
 2. (a) Administrator osvježava stranicu - scenarij nastavlja korakom 2
(b) Administrator odustaje od pregleda korisnika - kraj scenarija
- 2.b Ne postoji nijedan korisnik
 1. Administrator dobiva obavijest o nepostojanju korisnika
 2. Administrator odustaje od pregleda korisnika - kraj scenarija

UC28 - Pregled vlastitih oglasa

- **Glavni sudionik:** Donator

- **Cilj:** Pregled vlastitih oglasa

- **Sudionici:** Baza podataka

- **Preduvjet:** Donator je prijavljen u sustav

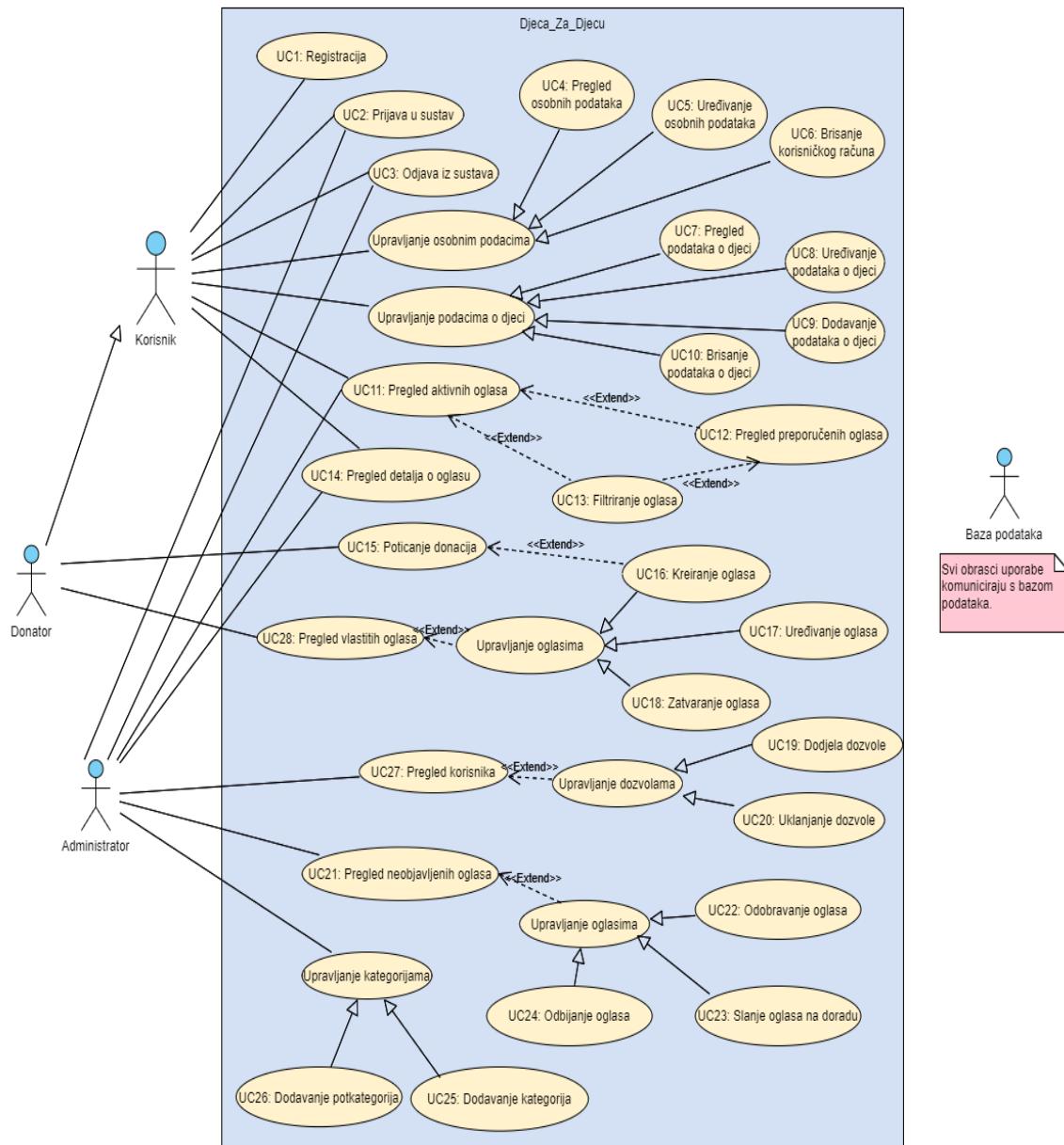
- **Opis osnovnog tijeka:**

1. Donator je odabrao opciju za prikaz svojih oglasa
2. Sustav prikazuje sve donatorove oglase
3. Donator pregledava oglase

- **Opis mogućih odstupanja:**

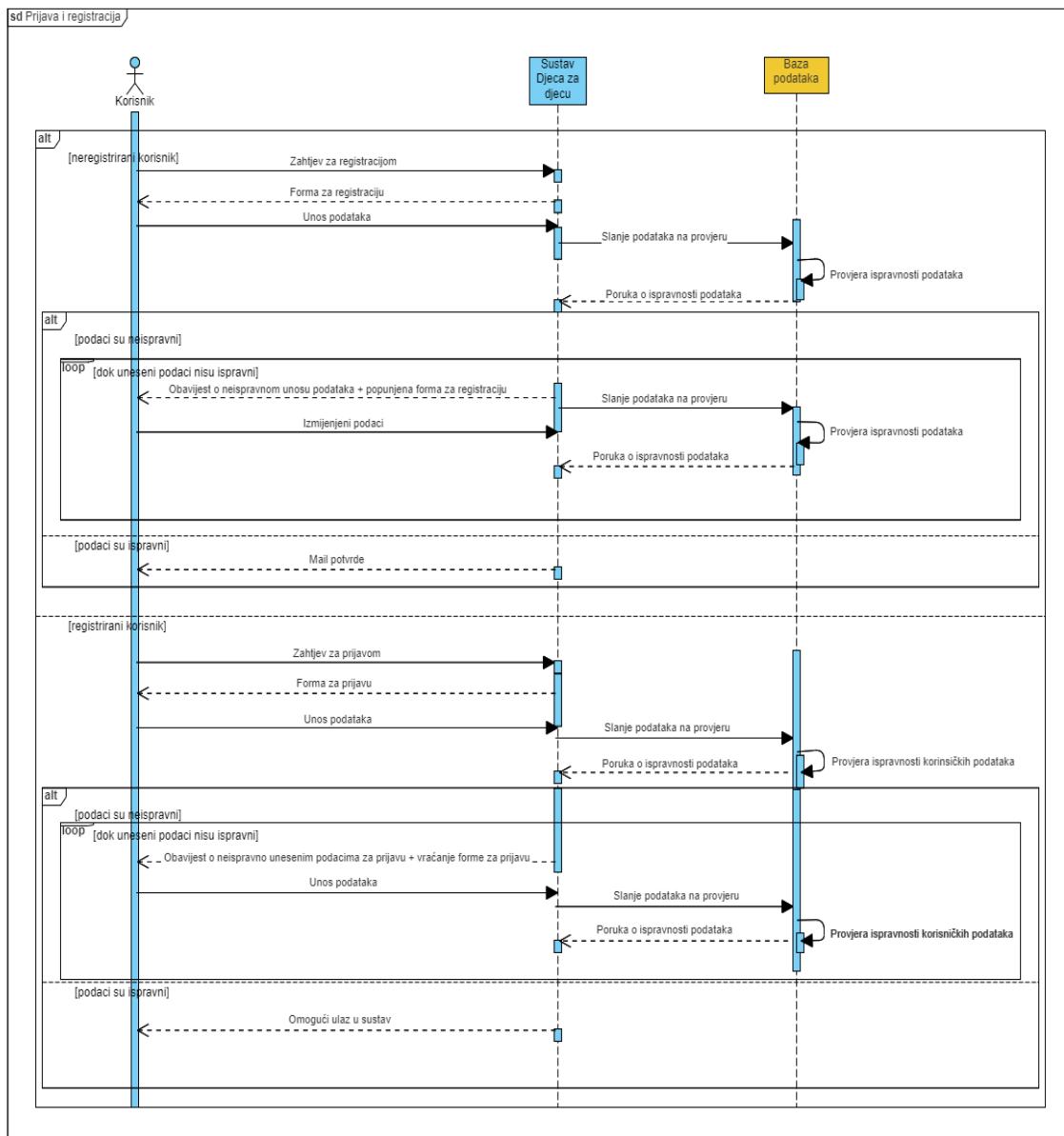
- 1.a Donator ne odabire navedenu akciju - kraj scenarija
- 2.a Baza ne uspije poslati podatke o oglasima
 1. Donator dobiva obavijest o neuspješnom dohvatu podataka o oglasima
 2. (a) Donator osvježava stranicu - scenarij nastavlja korakom 2
(b) Donator odustaje od pregleda korisnika - kraj scenarija
- 2.b Ne postoji nijedan donatorov oglas
 1. Donator dobiva obavijest o nepostojanju vlastitih oglasa
 2. Donator odustaje od pregleda vlastitih oglasa - kraj scenarija

Dijagram obrazaca uporabe



Slika 3.1: Dijagram obrazaca uporabe - aplikacija Djeca za djecu

3.1.2 Sekvencijski dijagrami

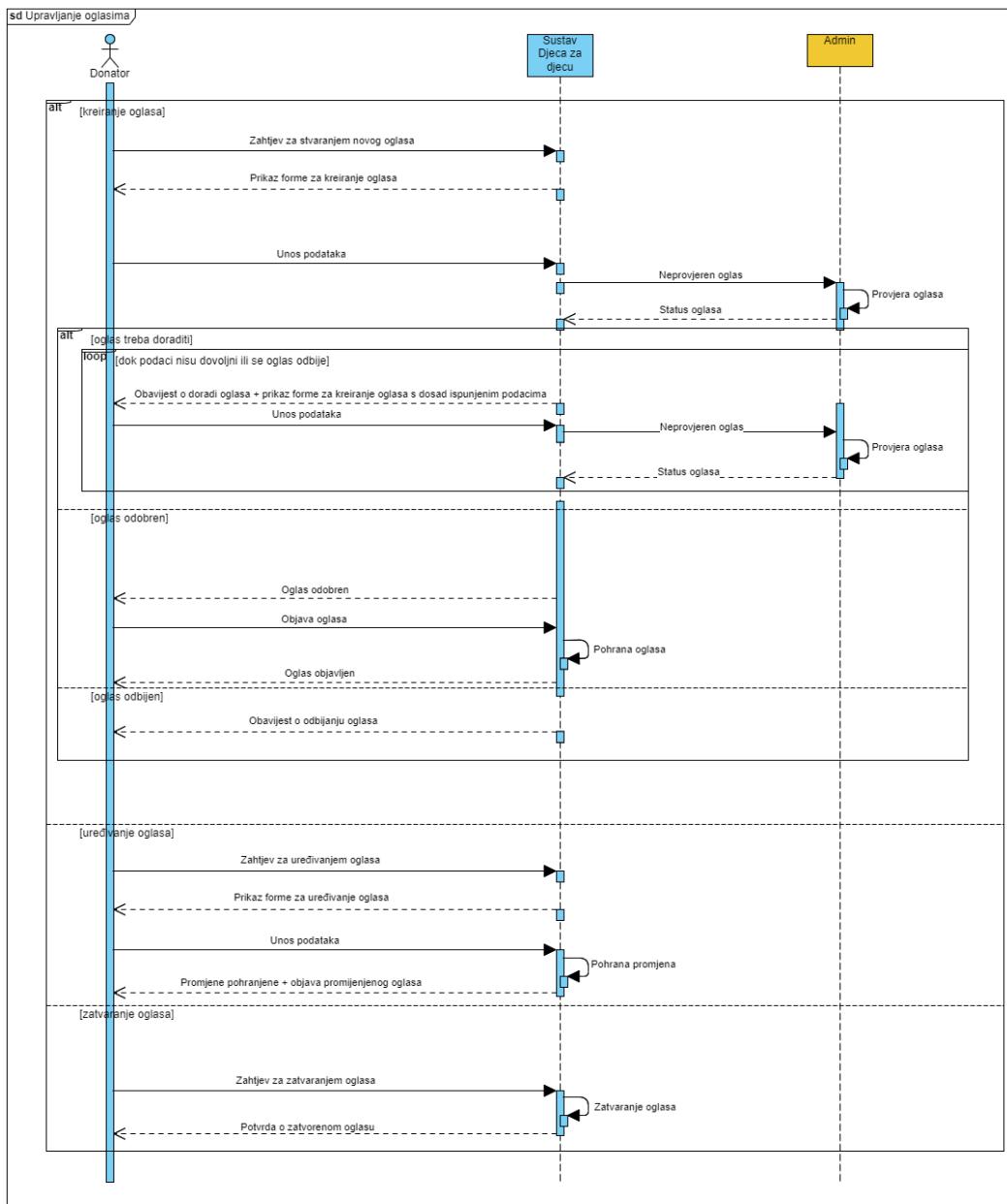


Slika 3.2: Sekvencijski dijagram - prijava i registracija

Dijagram prikazuje početni zaslon tj. opcije početnog zaslona kada korisnik pristupi aplikaciji. Korisnik ima dvije opcije, ako je neregistriran mora otiti na registraciju (klikom miša) i tada mu se prikazuje forma za registraciju koju treba popuniti i poslati sustavu klikom na gumb potvrde. Sustav provjerava ispravnost unesenih podataka slanjem u bazu podataka koja sustavu šalje odgovor o ispravnosti. Postoji mogućnost da su

uneseni podaci neispravni. U tom slučaju sustav vraća formu korisniku na popravljanje i korisnik izmjenjene podatke treba ponovo poslati sustavu koji će ih provjeriti (na isti način već opisan gore). Taj slučaj se može ponoviti onoliko puta dok uneseni podaci nisu ispravni. Ispravno uneseni podaci znače da sustav šalje mail potvrde, sada, registrirnom korisniku.

Registrirani korisnik može klikom na gumb za prijavu poslati sustavu zahtjev za formu za prijavu koju treba ispuniti kad mu se prikaže(sustav je pošalje). Uneseni podaci se provjeravaju u bazi korisnika u sustavu. Ako su ovdje uneseni podaci krivi korisniku se šalje forma za prijavu kao i poruka o neispravno unesenim podacima. Dok podaci nisu ispravni ovaj korak se ponavlja. Kada su podaci ispravno uneseni sustav omogućava ulazak u sustav(preusmjerava na stranicu s oglasima)



Slika 3.3: Sekvencijski dijagram - upravljanje oglasima

Dijagram prikazuje upravljanje oglasima koje je omogućeno donatorima (korisnici koji su od Administratora dobili dozvolu za doniranjem). Donator ima tri opcije koje može odabrat. Ukoliko odabere opciju kreiranja oglasa sustav mu šalje formu za kreiranje oglasa. Korisnik unosi podatke o predmetu koji oglašava toliko dugo dok nisu potvrđeni. Unešene podatke sustav šalje Adminu na provjeru koji potom sustavu javlja treba li ih poslati korisniku na doradu, javiti korisniku da je oglas odobren ili čak odbiti. U slučaju dorade donatoru sustav šalje njegovu formu s podacima o oglasu i gornji ko-

raci se provode ponovo. Ako je oglas odobren donator može objaviti oglas u sustavu koji taj oglas pohranjuje i šalje potvrdu donatoru o objavi oglasa. Kad je oglas odbijen on se odbacuje i korisnik dobiva obavijest o odbijanju oglasa.

Kod uređivanja oglasa korisnik sustavu šalje zahtjev za uređivanjem određenog oglasa i sustav odgovara slanjem forme za uređivanje oglasa. Donator unosi podatke i oni se pohranjuju u sustav te se ponovno objavljuje oglas.

Opcija zatvaranja oglasa podrazumijeva slanje zahtjeva za zatvaranjem oglasa. Po primjeku zahtjeva sustav zatvara oglaš i vraća korisniku potvrdu o zatvorenom oglasu.

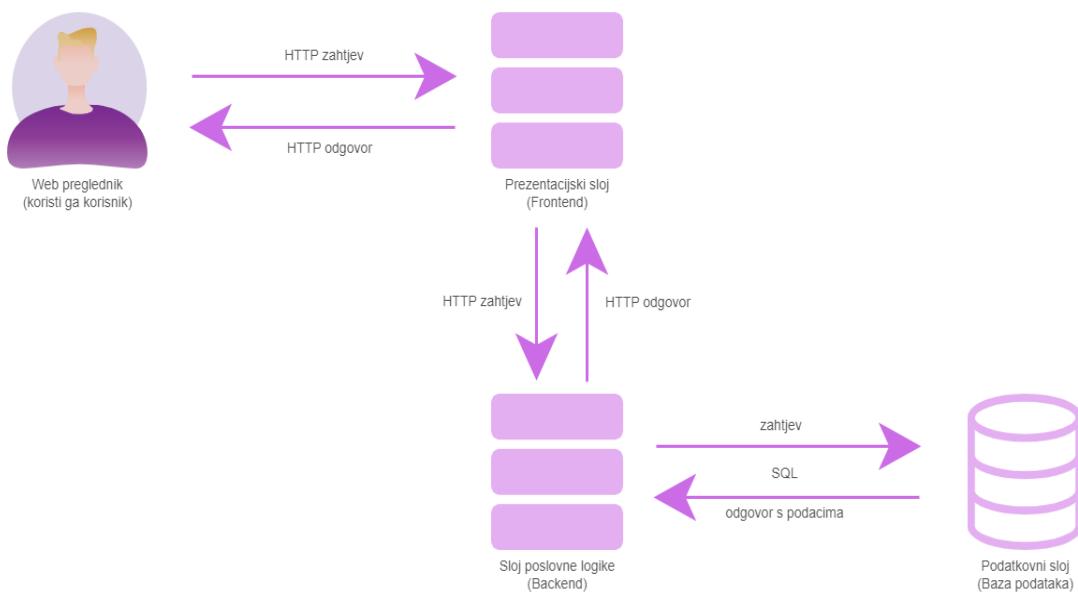
3.2 Ostali zahtjevi

- Sustav treba podržavati istovremenu aktivnost više korisnika
- Aplikacija treba biti responzivna tj. prilagođena uređajima svih rezolucija (tableti, mobiteli)
- Aplikacija treba biti jednostavna za korištenje i imati intuitivno sučelje
- Aplikacija treba biti implementirana u arhitekturi klijent-poslužitelj
- Poslužiteljska strana aplikacije mora biti implementirana koristeći jezik Java i radni okvir Spring Boot
- Podaci trebaju biti spremani u bazu podataka koristeći Java Persistence API (JPA)
- Funkcionalnosti poslužiteljske strane aplikacije trebaju biti izložene kroz REST Web servise
- Klijentska strana aplikacije mora biti implementirana koristeći tehnologije Angular ili React

4. Arhitektura i dizajn sustava

Arhitektura naše aplikacije zajedno s vanjskim resursima (bazom) na najvišoj razni apstrakcije odgovara troslojnoj klijent-poslužitelj arhitekturi sa sljedećim slojevima:

- Prezentacijski sloj (Frontend)
- Sloj poslovne logike (Backend)
- Podatkovni sloj (Baza podataka)



Slika 4.1: Arhitektura sustava

Općenito govoreći, prezentacijski sloj tj. frontend zadužen je za prikaz stranice korisniku i omogućavanje rada s istom. S druge strane, sloj poslovne logike tj. backend zadužena je za razne izračune i interakciju s bazom. U ovom pogledu, sloj poslovne logike služi kao spona između prezentacijskog sloja i podatkovnog sloja koja pritom obavlja sve akcije neprikladne za izvođenje na klijentovom računalu.

Kada korisnik u web preglednik unese URL naše stranice, web preglednik pristupa našoj aplikaciji slanjem HTTP GET zahtjeva prezentacijskom sloju. Prezentacijski sloj u odgovoru šalje sve potrebne podatke za prikaz korisničkog sučelja - naše stranice. Kada korisniku treba poslužiti neki sadržaj koji nije pohranjen na prezentacijskom sloju (npr. aktivne donacije, podatci o korisniku i oglasima), prezentacijski sloj sloju poslovne logike šalje HTTP zahtjev - npr. GET zahtjev s ciljem dohvata podataka o djeci korisnika. Komunikacija između prezentacijskog sloja i sloja poslovne logike odvija se po REST načelima.

Sloj poslovne logike potreban je za komplikiranije izračune, dohvat i izmjenu podataka te odgovaranje na zahtjeve pristigle od strane prezentacijskog sloja. Podatkovni sloj tj. baza podataka služi za pohranu svih potrebnih podataka (npr. podataka o registriranim korisnicima, oglasima i predmetima). Pri radu naše aplikacije, s bazom podataka komunicira isključivo sloj poslovne logike tj. backend - ako prezentacijskom sloju trebaju određeni podaci, poslat će zahtjev sloju poslovne logike te u idealnom slučaju u odgovoru dobiti tražene podatke. Pritom, JPA služi za objektno/relacijsko mapiranje (ORM) zahtjeva backenda u SQL upite pomoću kojih je moguće komunicirati s bazom.

Dodatno, i prezentacijski sloj i sloj poslovne logike interno imaju vlastitu podjelu - sloj poslovne logike možemo podijeliti na 3 glavne komponente: kontroleri, servisi i rezervorij (sloj pristupa podacima). Sličnu podjelu moguće je uočiti i na prezentacijskom sloju. Baza podataka vanjski je resurs koji naša aplikacija koristi.

Prezentacijski sloj tj. frontend izrađen je koristeći programski jezik Javascript zajedno s "markup" jezikom HTML, CSS-om i radnim okvirom React. Zbog korištenja radnog okvira React koji omogućava provođenje brojnih izračuna i funkcionalnosti na klijentovom računalu, možemo reći da je u pitanju "debeli" klijent. Sloj poslovne logike tj. backend izrađen je koristeći programski jezik Java zajedno s radnim okvirom Java Spring Boot. Za prevodenje zahtjeva backenda u ispravne SQL upite korišten je JPA - Java Persistence API. Shema baze podataka izrađena je koristeći alat ERDplus, a sama baza podataka izrađena je koristeći sustav za upravljanje bazama PostgreSQL.

4.1 Baza podataka

Za naš zadatak odabrali smo koristiti relacijsku bazu podataka kako bismo što učinkovitije i realističnije prikazali potreban sustav, ali isto tako i zato što s relacijskim bazama podataka imamo najviše iskustva, primarno stečenog na kolegiju Baze podataka. Baza podataka sastoji se od nekolicine relacija tj. tablica definiranih imenom relacije i skupom atributa relacije. Među atributima relacije obvezno se nalazi primarni ključ, a ponekad i strani ključevi koji povezuju tu relaciju s ostalima. Bazu koristimo za pohranu, izmjenu i dohvata potrebnih podataka.

Baza podataka naše aplikacije sastoji se od sljedećih relacija:

- User
- Child
- Category
- Subcategory
- isInCategory
- isInSubcategory
- Item
- Donation
- donatedToUser

4.1.1 Opis tablica

Primarni ključevi relacija označeni su zelenom bojom, a strani ključevi plavom bojom.

Relacija **User** služi za pohranu podataka o registriranome korisniku aplikacije. Sadrži atribute: email, userName, userSurname, password, userLocation, isAccountVerified i canDonate. Relacija je povezana s relacijom Child, relacijom Donation i relacijom donatedToUser.

User		
email	VARCHAR	Email adresa korisnika
userName	VARCHAR	Ime korisnika
userSurname	VARCHAR	Prezime korisnika
password	VARCHAR	Lozinka korisnika
userLocation	VARCHAR	Lokacija korisnika
isAccountVerified	BOOLEAN	Oznaka je li korisnik verificiran
canDonate	BOOLEAN	Oznaka može li korisnik donirati

Relacija **Child** služi za pohranu podataka o djetetu registriranog korisnika aplikacije. Sadrži atribute: email, childId, childName, childSex, childAge i predictedBirthDate. Relacija je povezana s relacijom User, relacijom isInCategory i relacijom isInSubcategory.

Child		
childId	INT	Generirani id djeteta
email	VARCHAR	Email adresa roditelja - također strani ključ
childName	VARCHAR	Ime djeteta
childSex	VARCHAR	Spol djeteta
childAge	INT	Dob djeteta
predictedBirthDate	DATE	Predviđeni datum rođenja djeteta

Relacija **Category** služi za pohranu podataka o određenoj kategoriji. Sadrži atribut: categoryName. Relacija je povezana s relacijom isInCategory, relacijom Subcategory i relacijom Item.

Category		
categoryName	VARCHAR	Ime kategorije

Relacija **Subcategory** služi za pohranu podataka o određenoj potkategoriji. Sadrži atribute: subcategoryName, itemDuration i categoryName. Relacija je povezana s relacijom isInSubcategory, relacijom Category i relacijom Item.

Subcategory		
subcategoryName	VARCHAR	Ime potkategorije
itemDuration	INT	Predviđeni rok uporabe predmeta iz potkategorije u godinama
categoryName	VARCHAR	Ime roditeljske kategorije

Relacija **isInCategory** nastala je iz veze N:N između relacija Child i Category i služi za povezivanje istih. Sadrži atribute: childId, email i categoryName. Relacija je povezana s relacijom Child i relacijom Category.

isInCategory		
childId	INT	Generirani id djeteta - također dio prvog stranog ključa
email	VARCHAR	Email adresa roditelja - također dio prvog stranog ključa
categoryName	VARCHAR	Ime kategorije označene za dijete - također drugi strani ključ

Relacija **isInSubcategory** nastala je iz veze N:N između relacija Child i Subcategory i služi za povezivanje istih. Sadrži atribute: childId, email i subcategoryName. Relacija je povezana s relacijom Child i relacijom Subcategory.

isInSubcategory		
childId	INT	Generirani id djeteta - također dio prvog stranog ključa
email	VARCHAR	Email adresa roditelja - također dio prvog stranog ključa
subcategoryName	VARCHAR	Ime potkategorije označene za dijete - također drugi strani ključ

Relacija **Item** služi za pohranu podataka o predmetu za donaciju. Sadrži attribute: id, productName, itemState, productionYear, productionBrand, forAge, forSex, categoryName i subcategoryName. Relacija je povezana s relacijom Category, relacijom Subcategory i relacijom Donation.

Item		
id	INT	Generirani id predmeta
productName	VARCHAR	Ime proizvoda
itemState	VARCHAR	Stanje predmeta
productionYear	DATE	Godina proizvodnje predmeta
productionBrand	VARCHAR	Marka predmeta
forAge	INT	Predviđena dob za korištenje predmeta
forSex	VARCHAR	Predviđen spol za korištenje predmeta
categoryName	VARCHAR	Ime kategorije predmeta
subcategoryName	VARCHAR	Ime potkategorije predmeta

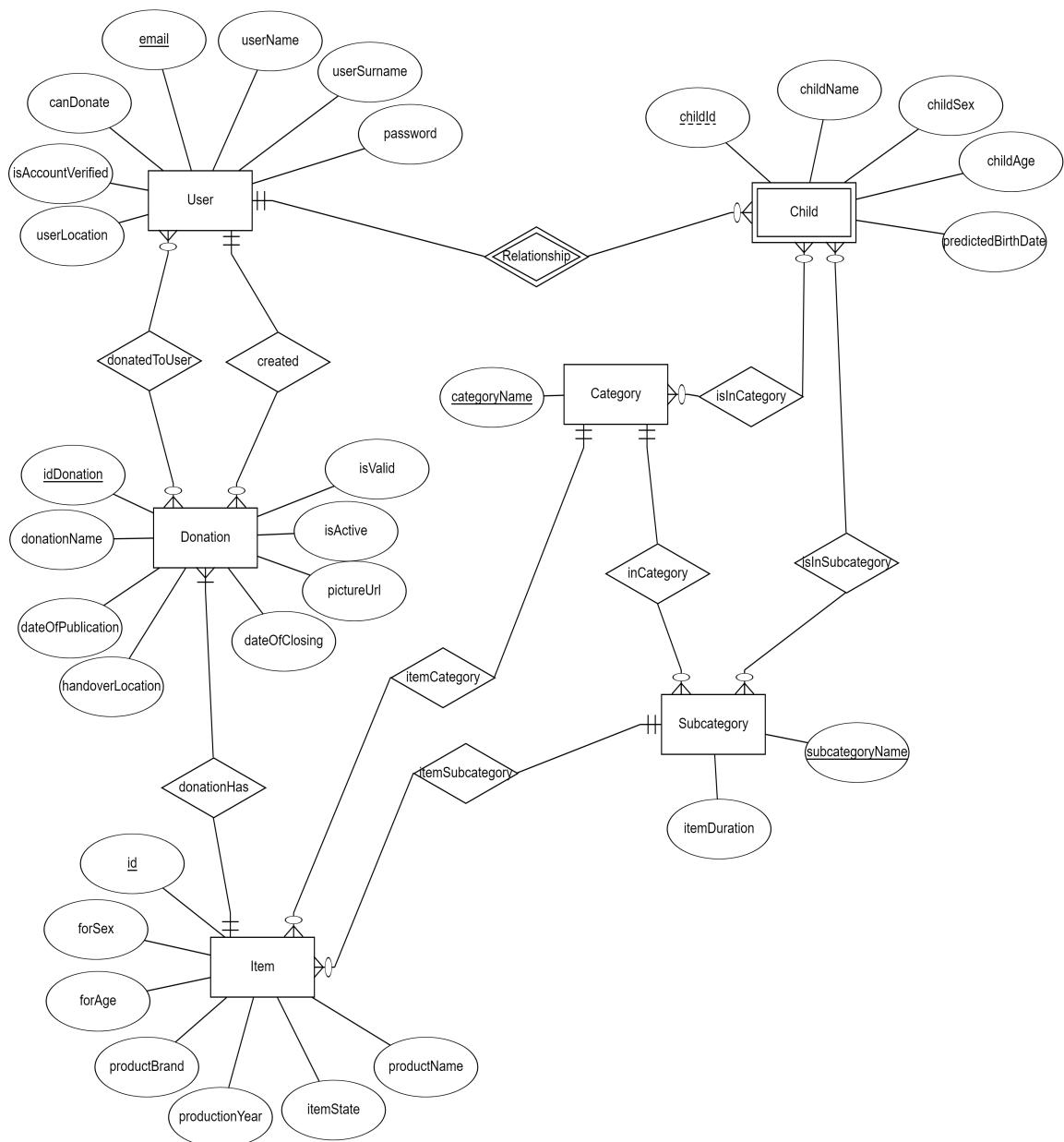
Relacija **Donation** služi za pohranu podataka o donaciji. Sadrži atribute: idDonation, donationName, dateOfPublication, dateOfClosing, isValid, isActive, pictureUrl, handoverLocation, email i id. Relacija je povezana s relacijom Item, relacijom User i relacijom donatedToUser.

Donation		
idDonation	INT	Generirani id donacije
donationName	VARCHAR	Ime donacije
dateOfPublication	DATE	Datum objave donacije
dateOfClosing	DATE	Datum zatvaranja donacije
isValid	BOOLEAN	Oznaka je li donacija prihvaćena od strane administratora
isActive	BOOLEAN	Oznaka je li donacija trenutno aktivna
pictureUrl	VARCHAR	Url slike donacije
handoverLocation	VARCAHR	Lokacija preuzimanja donacije
email	VARCHAR	Email adresa donatora
id	INT	Generirani id pripadajućeg predmeta

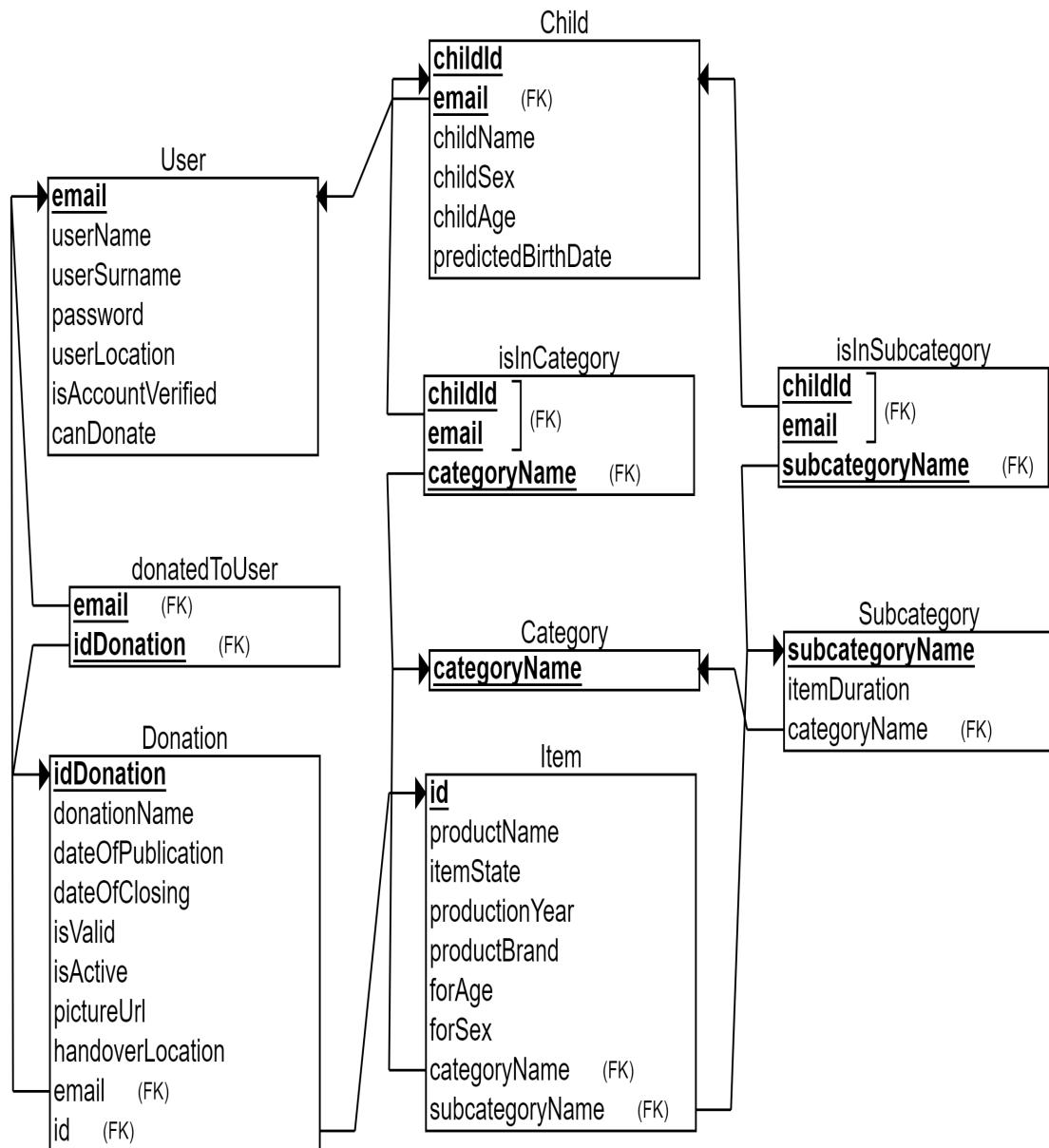
Relacija **donatedToUser** nastala je iz veze N:N između relacija User i Donation i služi za povezivanje istih. Sadrži atribute: email i idDonation. Relacija je povezana s relacijom User i relacijom Donation.

donatedToUser		
email	VARCHAR	Email adresa korisnika koji je prihvatio donaciju - također prvi strani ključ
idDonation	INT	Id prihvaćene donacije - također drugi strani ključ

4.1.2 Dijagram baze podataka



Slika 4.2: ER dijagram - aplikacija Djeca za djecu

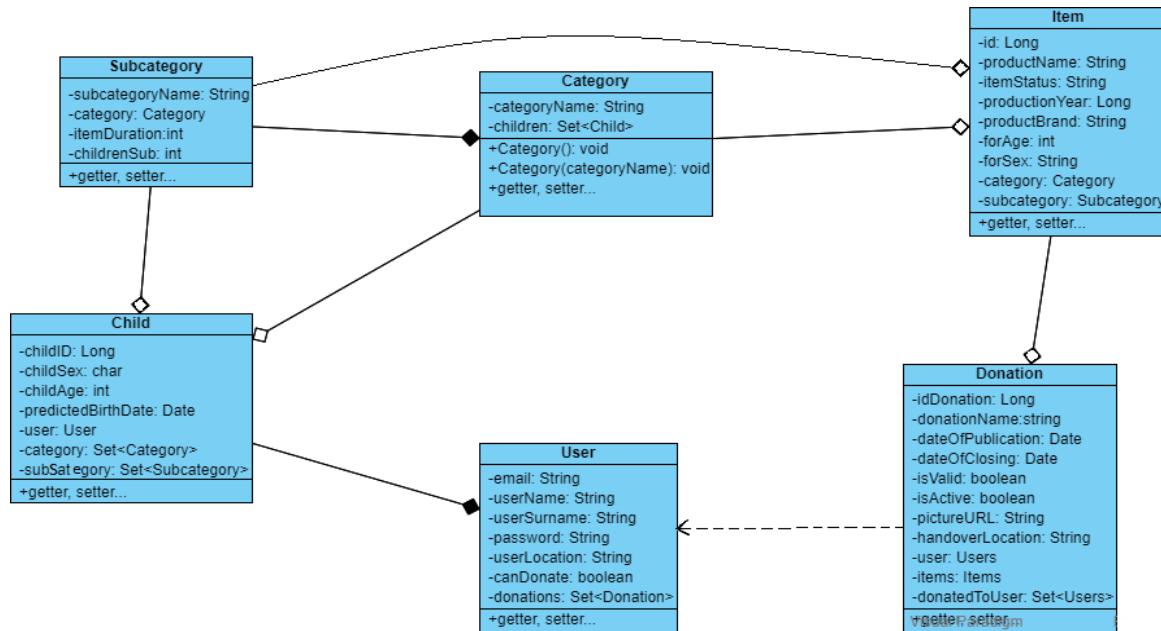


Slika 4.3: Relacijski dijagram - aplikacija Djeca za djecu

4.2 Dijagram razreda

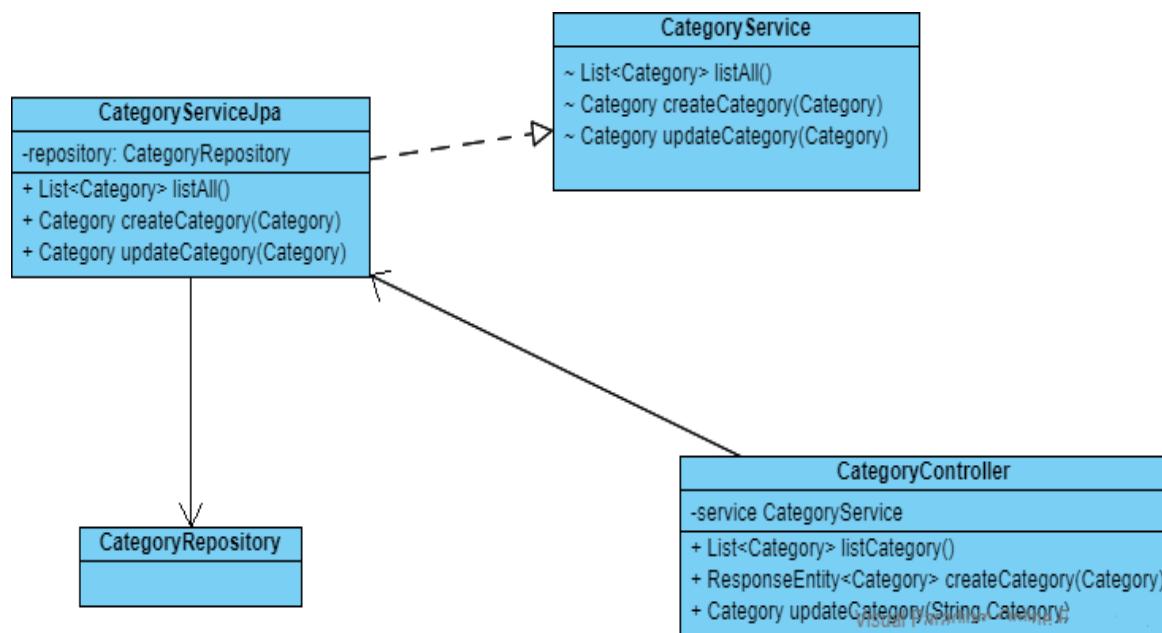
Na slikama 4.4 do 4.7 prikazani su razredi koji pripadaju backend dijelu našeg sustava. U pitanju su specifikacijski dijagrami razreda - implementacijski dijagrami prikazani su nakon opisa specifikacijskih dijagrama.

Slika 4.4 prikazuje razrede koji predstavljaju modele - preslikane relacije baze podataka. Razred User predstavlja registriranog korisnika sustava. Razred Child predstavlja dijete registriranog korisnika sustava. Razred Donation predstavlja jednu donaciju zabilježenu u sustavu. Razred Item predstavlja jedan predmet vezan za neke od donacija. Razred Category predstavlja podatke o jednoj od kategorija u koje mogu biti svrstani predmeti. Razred Subcategory predstavlja podatke o jednoj od potkategorija u koje mogu biti svrstani predmeti.



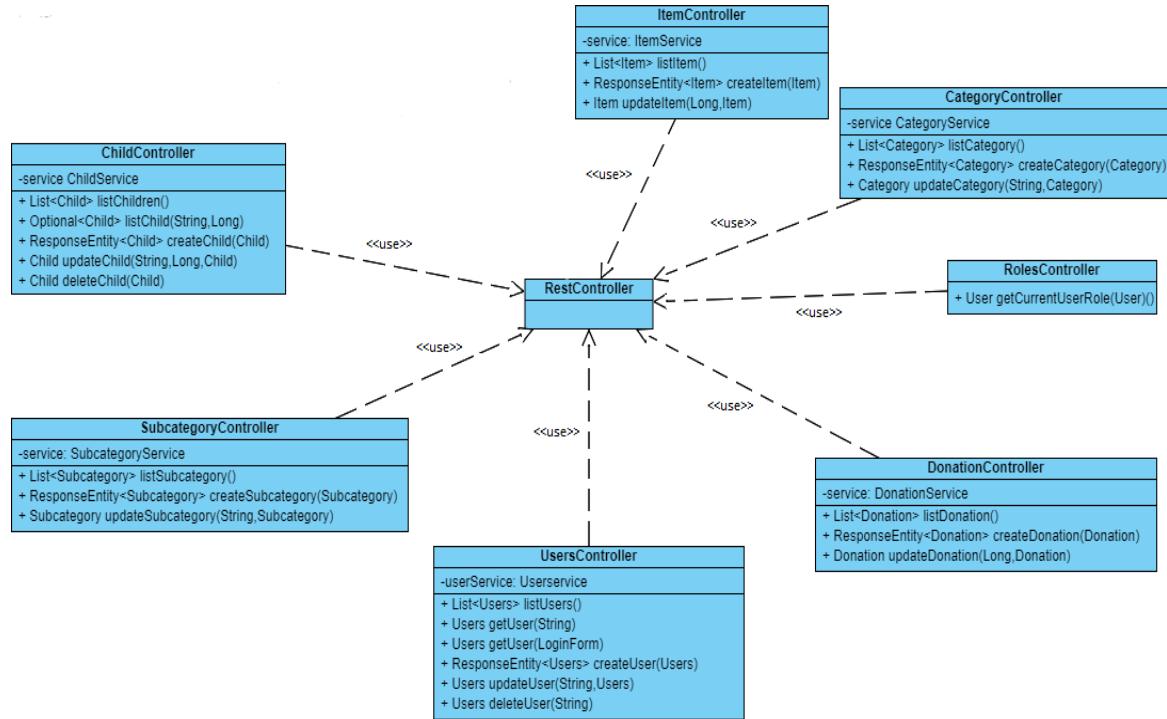
Slika 4.4: Dijagram razreda - dio Models

Slika 4.5 prikazuje odnos komponenti na primjeru razreda Category. Razred CategoryController prihvata i odgovara na HTTP zahtjeve poslane od korisnika tj. konkretnije frontenda. Kako bi na njih ispravno odgovorio, uspostavlja komunikaciju s razredom CategoryServiceJpa. Razred CategoryServiceJpa implementira sučelje CategoryService kako bi obavljao potrebne izračune i osigurao uspješno provođenje poslovne logike. Uz to, razred CategoryServiceJpa uspostavlja komunikaciju s razredom CategoryRepository koji služi kao sučelje za pohranu i dohvat podataka iz baze. Kako bi prikaz bio što sažetiji, odnos komponenti prikazan je samo na jednom razredu. U stvarnoj implementaciji, ovakav odnos komponenti tj. razreda postoji za svaki razred iz dijela Models.



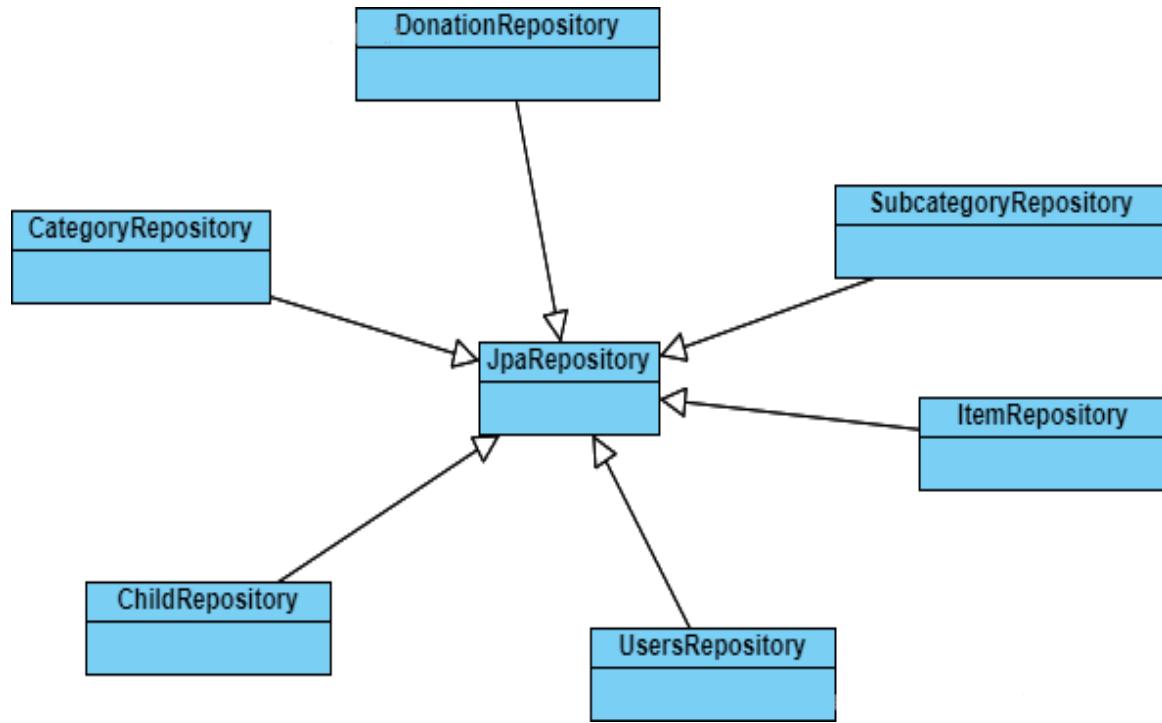
Slika 4.5: Dijagram razreda - odnos komponenti

Slika 4.6 prikazuje razrede izvedene iz razreda RestController. Svaki od razreda izvedenih iz RestController služi za prihvatanje i odgovor na HTTP zahtjeve poslane od strane korisnika tj. konkretnije frontenda.



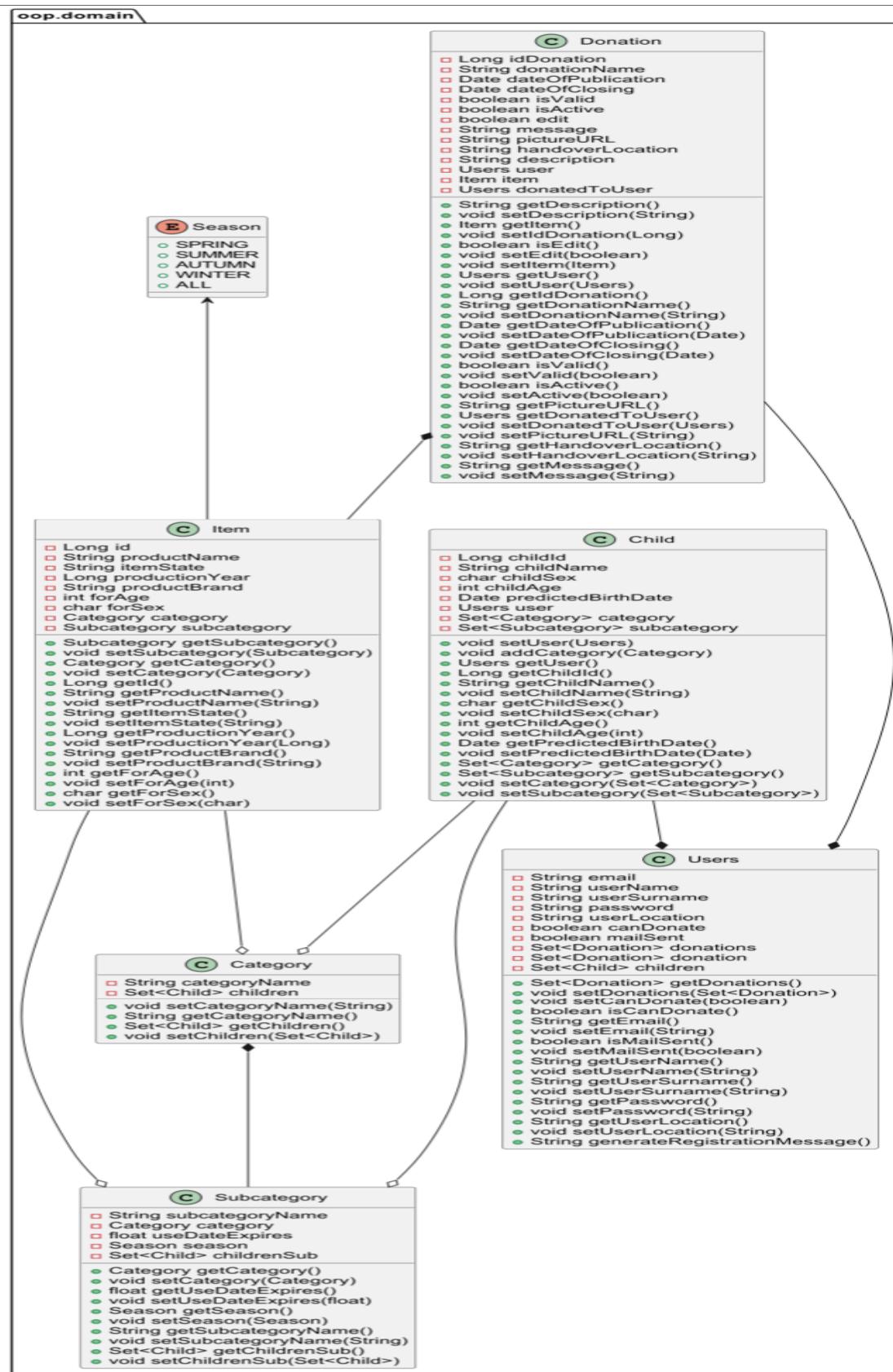
Slika 4.6: Dijagram razreda - dio Controllers

Slika 4.7 prikazuje razrede izvedene iz razreda JpaRepository. Svaki od prikazanih razreda služi kao sučelje za pohranu i dohvatanje podataka iz baze podataka za svoju pripadajuću relaciju.

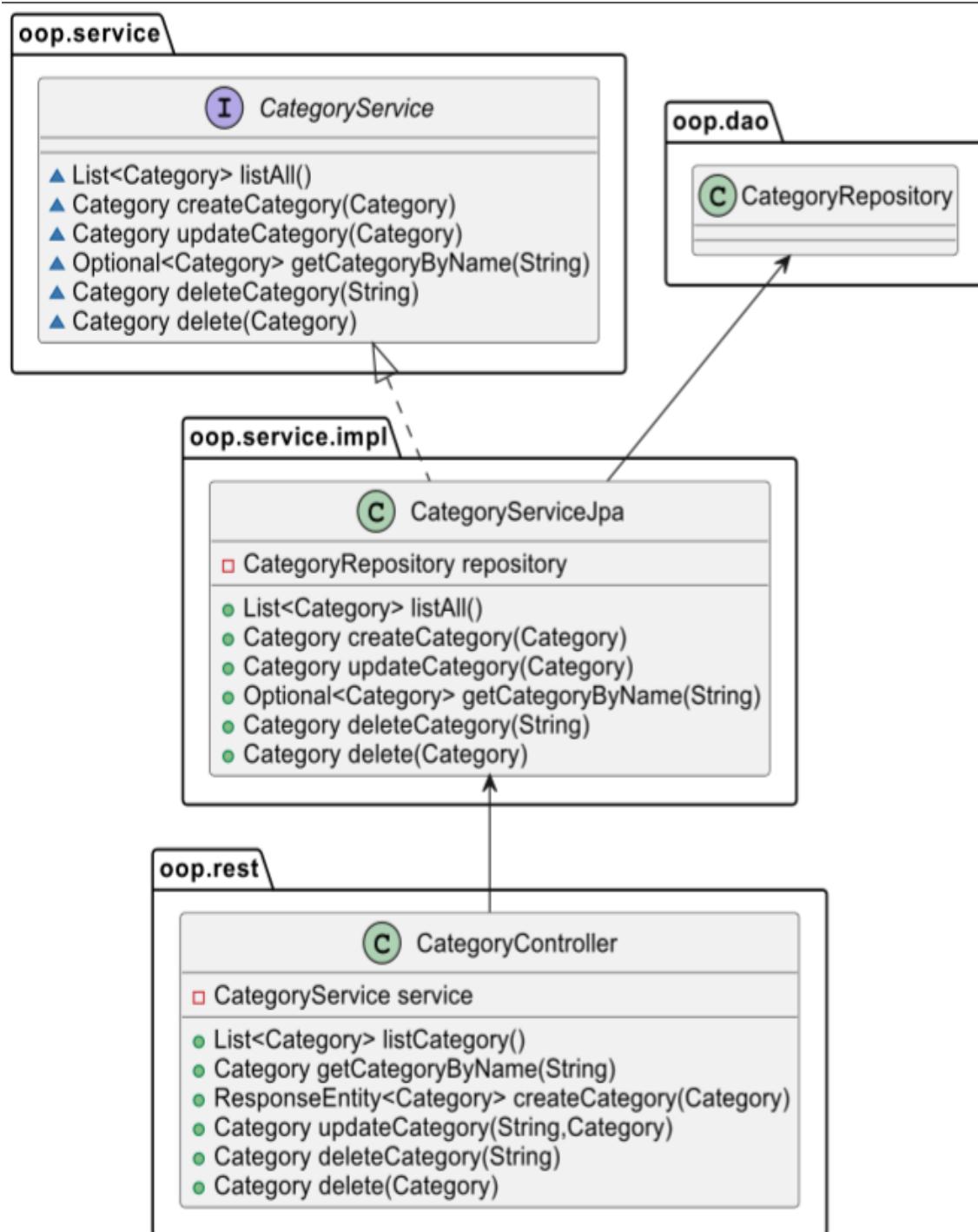


Slika 4.7: Dijagram razreda - dio Repositories

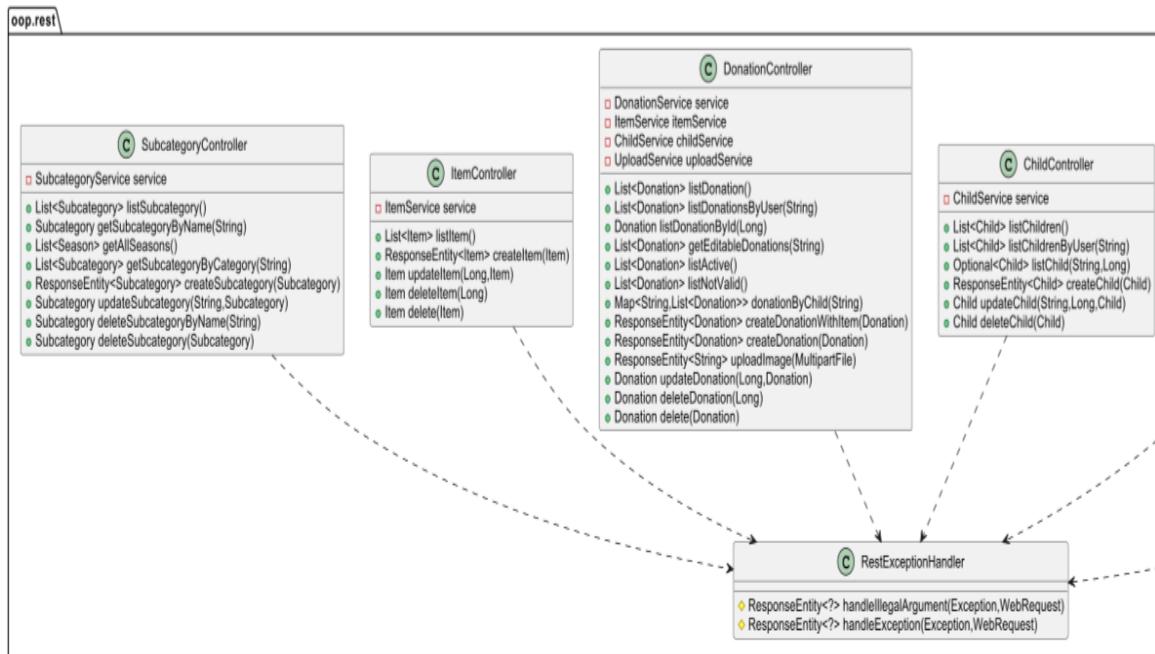
Na slikama 4.8 - 4.15 prikazani su implementacijski dijagrami razreda generirani direktno iz koda. Uz dijagrame razreda za koje su već opisani specifikacijski dijagrami (Models, odnos komponenti, Controllers, Repositories) dodatno su uključeni i dijagram razreda koji prikazuje odnos između `UsersController` klase i `EmailSenderService` klase koja se koristi za slanje potvrde o registraciji korisnika na mail, kao i dijagram razreda koji prikazuje odnos klase za ostvarenje sigurnosti, autentifikacije i autorizacije korisnika. Zbog preglednosti dijagrami razreda za dio Controllers i dio Repositories podijeljeni su u dva dijela.



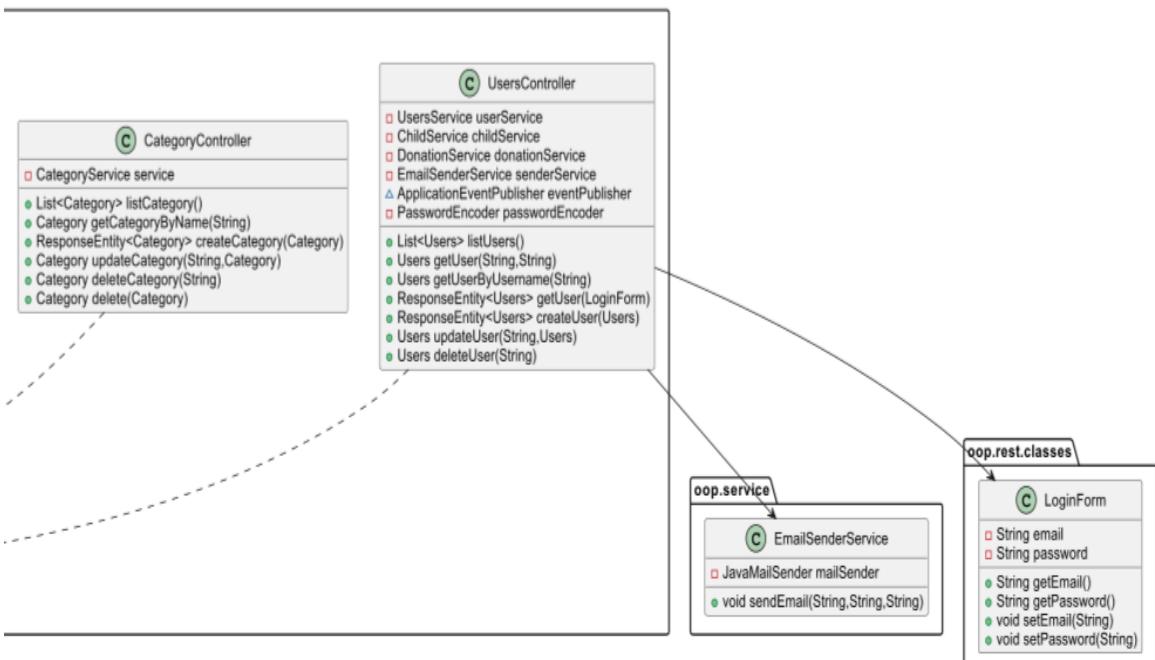
Slika 4.8: Dijagram razreda - dio Models (finalna verzija)



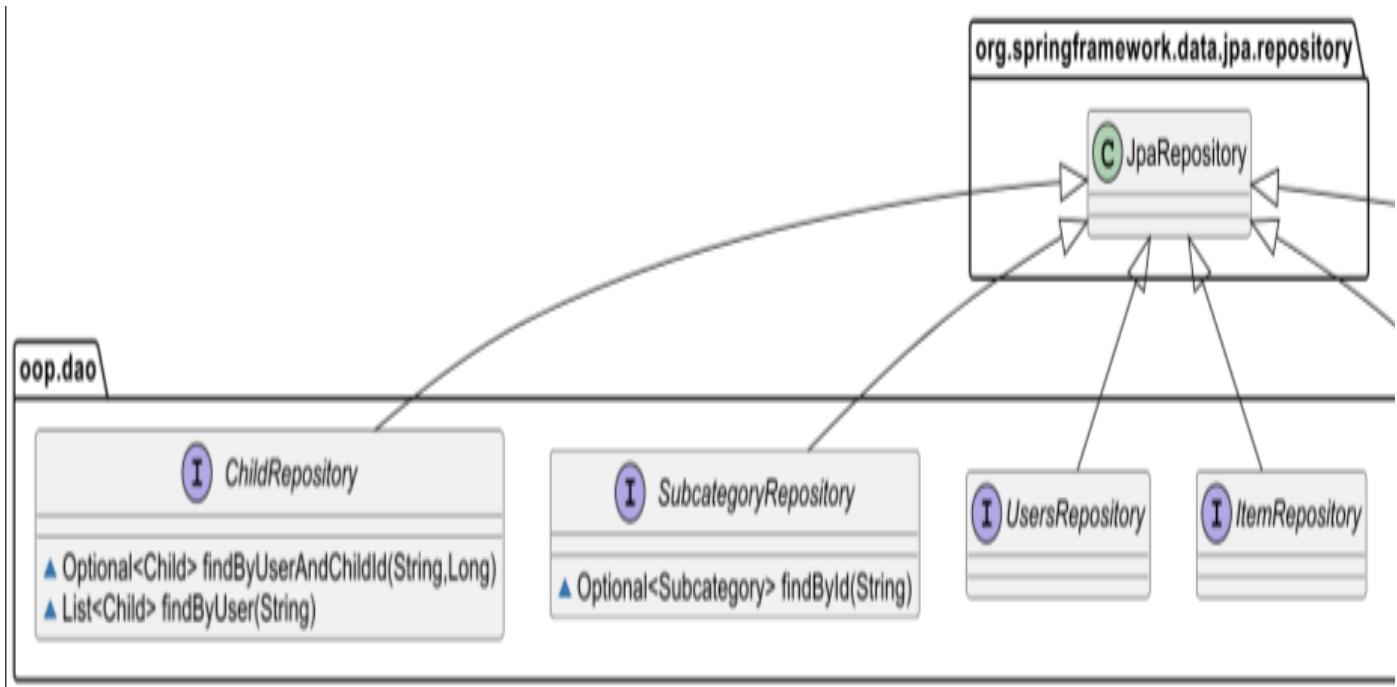
Slika 4.9: Dijagram razreda - odnos komponenti (finalna verzija)



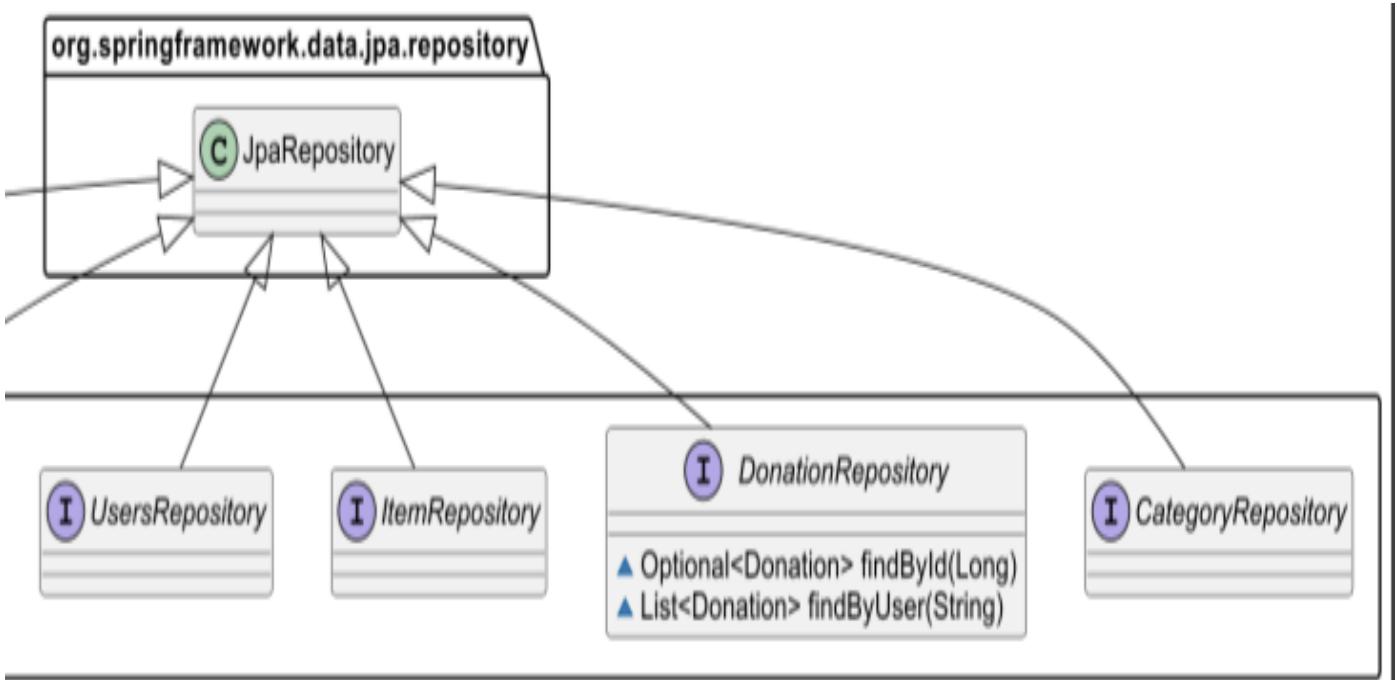
Slika 4.10: Dijagram razreda - dio Controllers, prvi dio (finalna verzija)



Slika 4.11: Dijagram razreda - dio Controllers, drugi dio (finalna verzija)



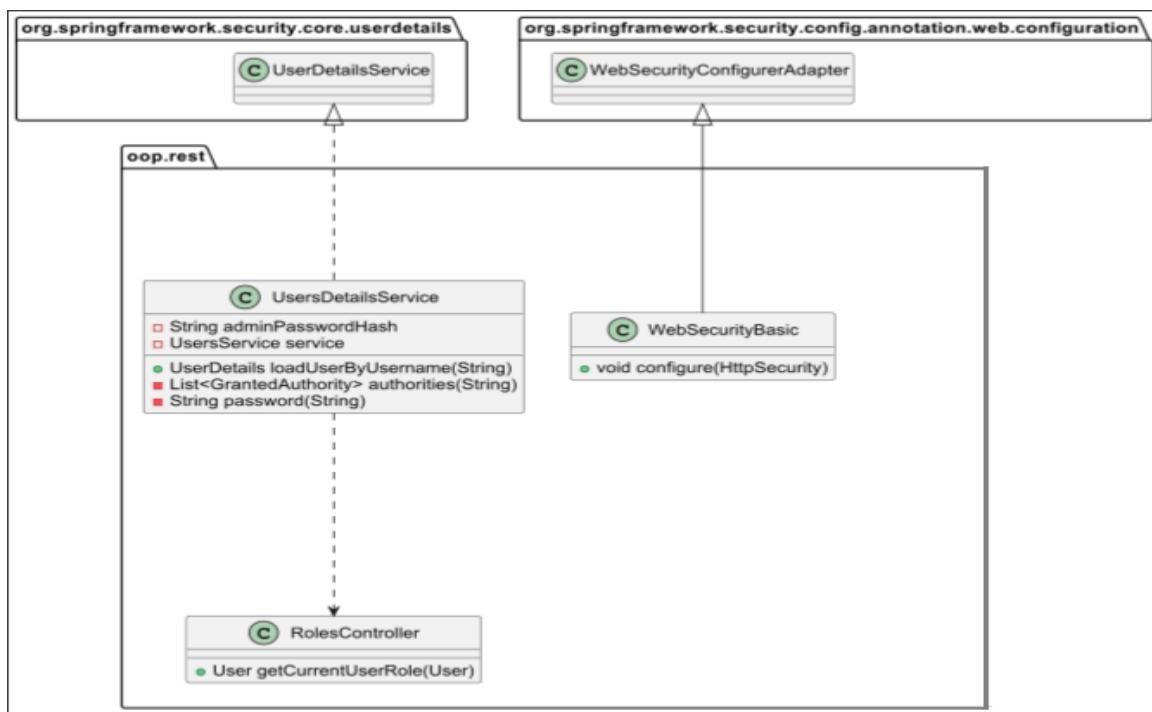
Slika 4.12: Dijagram razreda - dio Repositories, prvi dio (finalna verzija)



Slika 4.13: Dijagram razreda - dio Repositories, drugi dio (finalna verzija)



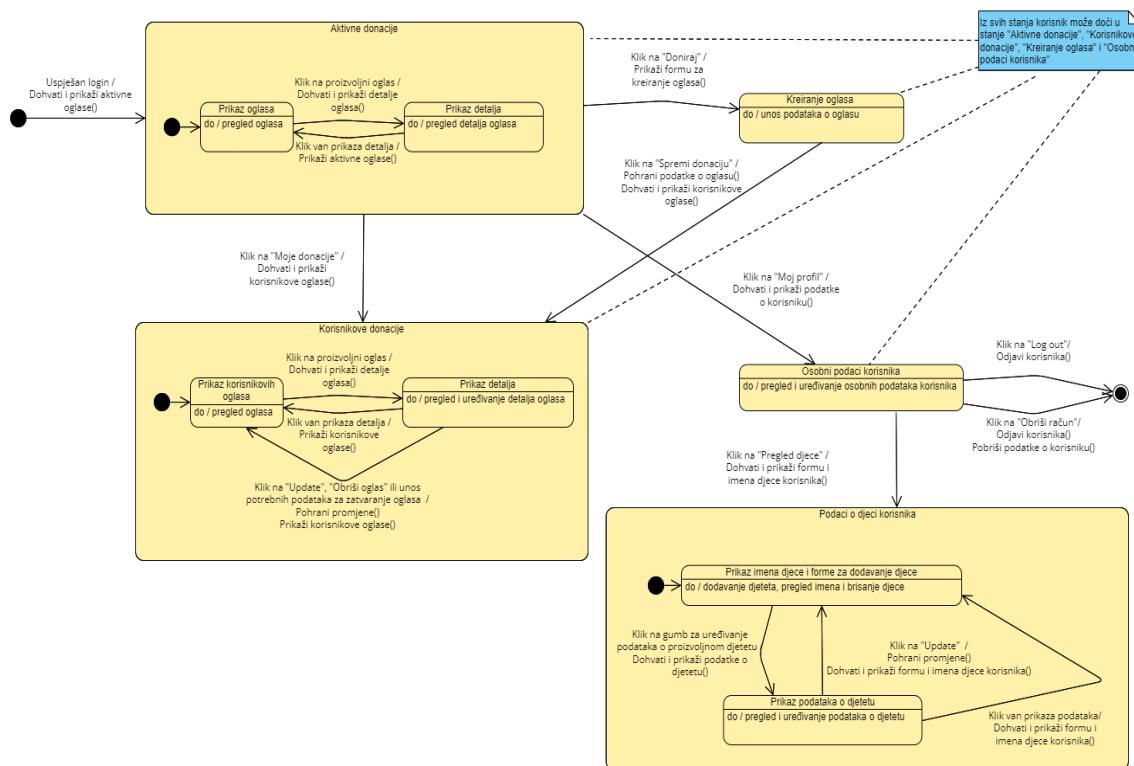
Slika 4.14: Dijagram razreda - dio Email Service



Slika 4.15: Dijagram razreda - dio Security

4.3 Dijagram stanja

Dijagram stanja prikazan na slici 4.16 prikazuje stanja u kojima se može naći sustav (konkretno, u pitanju je korisničko sučelje - frontend), kao i prijelaze pomoću kojih sustav dolazi u druga stanja. Svaki prijelaz označen je događajem (npr. "Klik na proizvoljni oglas"), kao i akcijom (npr. "Prikaži aktivne oglase()") koja se izvršava "tijekom prijelaza" - nakon što se dogodio događaj koji je potaknuo prijelaz u drugo stanje. Većina akcija označenih na prijelazima isto tako mogla je biti označena kao entry odnosno exit akcija samoga stanja, ali je zbog preglednosti i konzistentnosti odabran prikaz takvih akcija na prijelazima. Dijagram na slici 4.16 prikazuje stanja u kojima se sustav može naći kada je u sustav ulogiran registrirani korisnik za kojeg je administrator stavio dopuštenje da može donirati - da korisnik nema dopuštenje, sustav se pri radu s takvim korisnikom ne bi mogao naći u stanju "Kreiranje oglasa", kao ni u stanju "Korisnikove donacije" jer ono služi za prikaz korisnikovih prijašnjih, kao i trenutnih donacija.



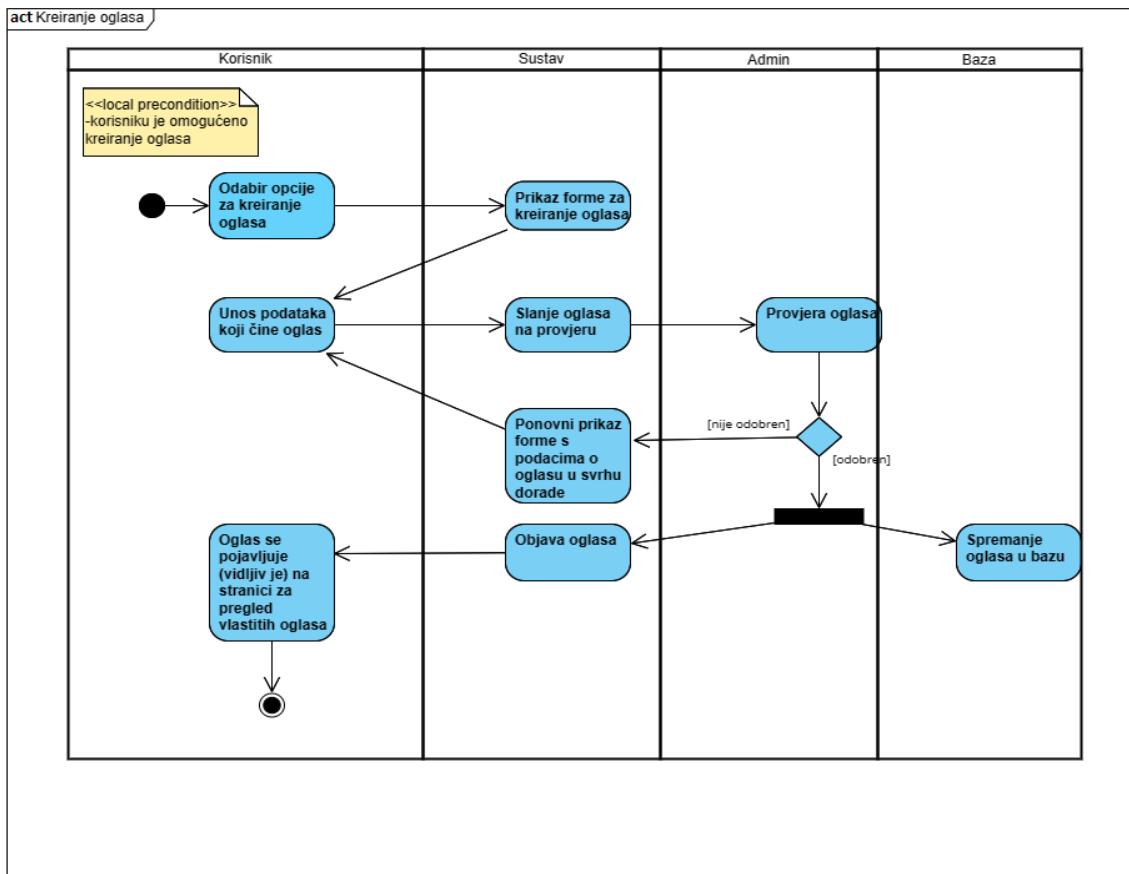
Slika 4.16: Dijagram stanja

Kako bi napravili razliku između akcija koje izvršava sustav i akcija koje radi korisnik u interakciji sa sustavom, sve akcije koje izvršava sustav na kraju imaju oznaku () (slično pozivu funkcije u kodu). Radi preglednosti, neke od strelica nisu prikazane na dijagramu - korisnik iz bilo kojeg stanja može preći u stanje "Aktivne donacije", "Korisnikove donacije", "Kreiranje oglasa" i "Osobno podaci korisnika" jer je prijelaz u ta stanja ostvaren klikom na odgovarajući dio zaglavlja stranice. Akcije koje radi korisnik u interakciji u sustavu naznačene su kao do akcije unutar stanja ili kao događaji koji potiču prijelaze u druga stanja.

Nakon uspješnog logina, registrirani korisnik preusmjerjen je na stranicu Aktivne donacije na kojoj može pregledavati sve oglase, ali isto tako i pregledavati detalje proizvoljnih oglasa. Pomoću zaglavlja stranice korisnik može doći do stranice Aktivne donacije, stranice za kreiranje oglasa, stranice za prikaz i uređivanje osobnih podataka te stranice za prikaz vlastitih donacija. Na stranici za kreiranje oglasa korisnik unosi podatke i nakon unosa klikom na gumb "Spremi donaciju" biva preusmjerjen na stranicu za prikaz vlastitih donacija. Na stranici za prikaz vlastitih donacija korisnik može pregledavati sve oglase koje je sam kreirao ili za-primio, kao i detalje o istima. Na stranici za prikaz i uređivanje osobnih podataka korisnik se također može i odjaviti, obrisati svoj račun i otići na stranicu za dodavanje, brisanje i prikaz podataka o djeci. Rad s aplikacijom završava kada korisnik odabere opciju "Log out".

4.4 Dijagram aktivnosti

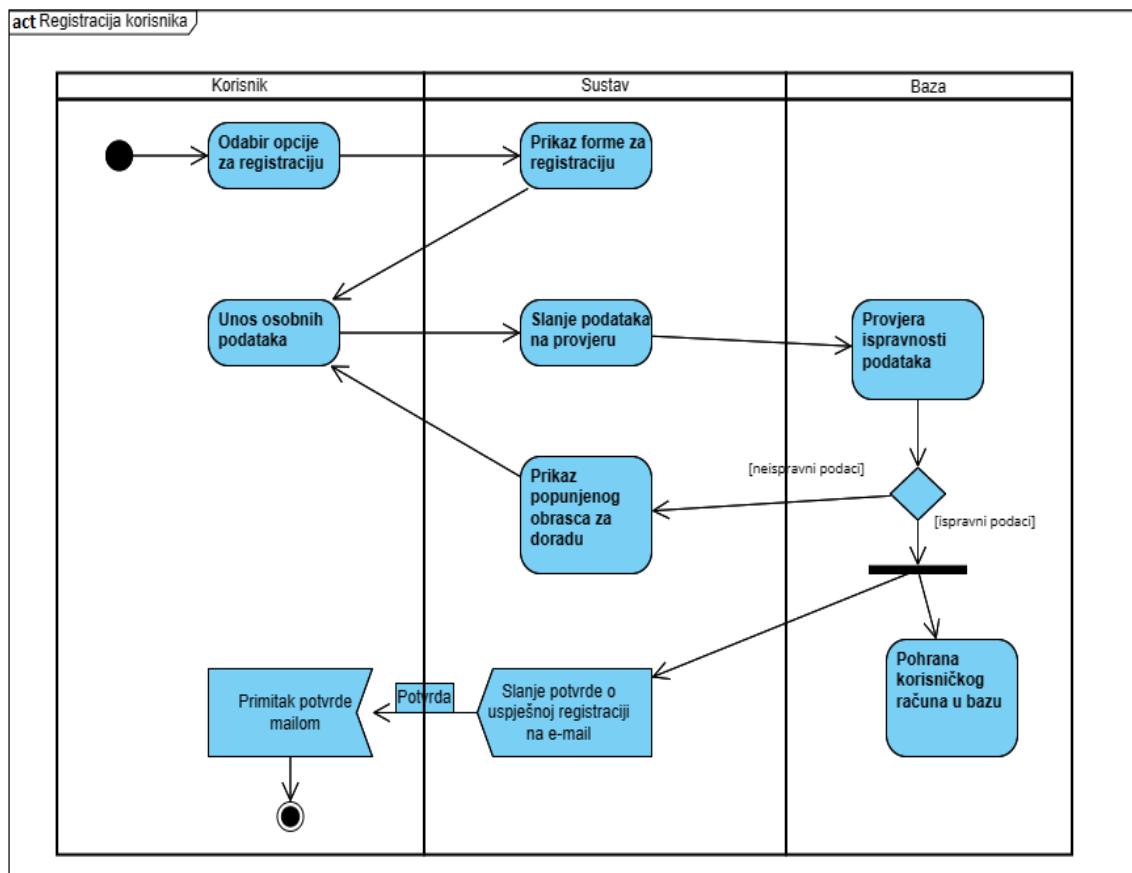
Dijagram aktivnosti na jednostavan način prikazuje opis modela toka upravljanja. Na slikama 4.17 i 4.18 su prikazani dijagrami aktivnosti za odabране procese: proces kreiranja oglasa i proces kreiranja korisničkog računa.



Slika 4.17: Dijagram aktivnosti - proces kreiranja oglasa

Na dijagramu jasno možemo razabrati aktore prema participijama i razlučiti njihove aktivnosti. Vidljivo je da započinjemo s korisnikom koji odabire opciju za kreiranje oglasa. Uvjet da bi uopće mogao izabrati tu opciju jest da mu je objavljenje/kreiranje oglasa dozvoljeno. Nakon što korisnik izabere opciju slijedi akcija od sustava koji će prikazati formu za kreiranje novog oglasa. Potom korisnik unosi podatke i tako kreira oglas koji zatim sustav šalje na provjeru. Oglase provjerava administrator i može odlučiti sadrži li oglas sve potrebne informacije ili ne i na temelju toga ih odobrava ili ne. Ukoliko oglasi nisu odobreni sustav će ih poslati korisniku na doradu. Odobreni oglasi se, pak objavljuju i spremaju u bazu. Nakon

objavljuvanja korisniku se njegov oglas pojavljuje na stranici za pregled vlastitih oglasa i time završava proces kreiranja oglasa.

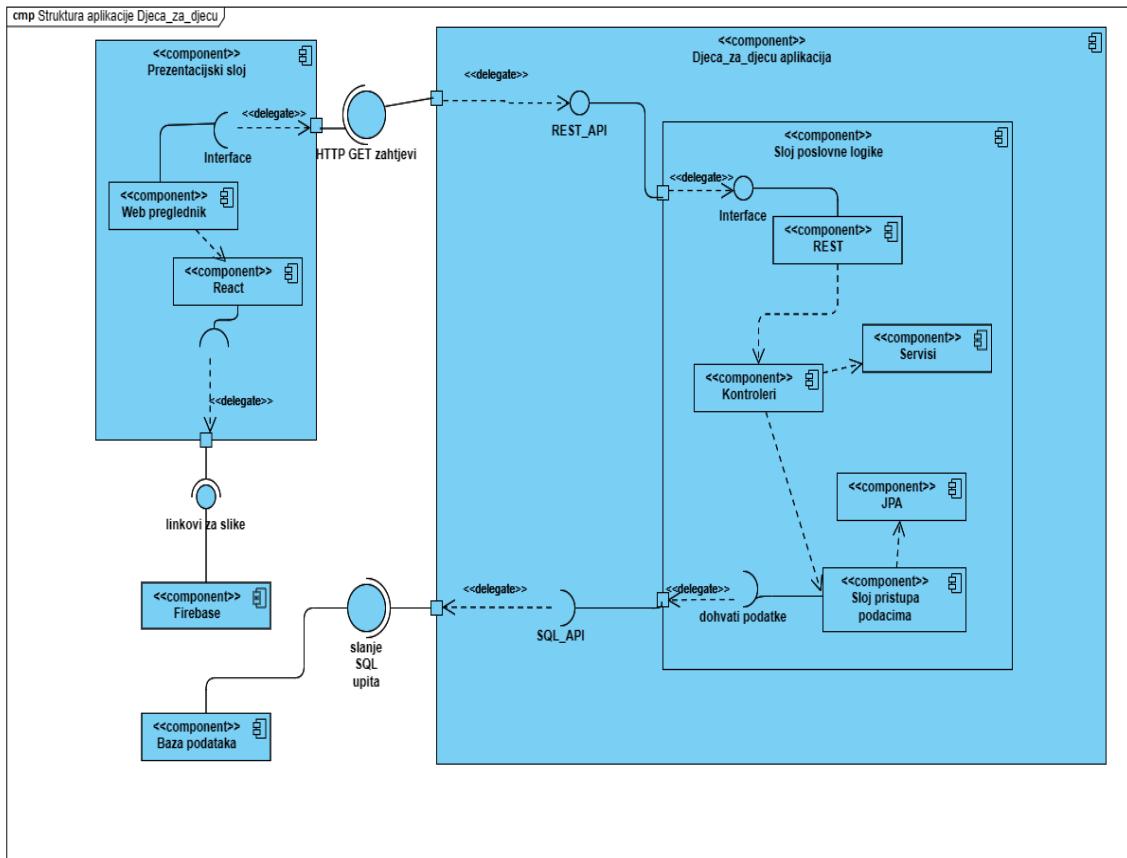


Slika 4.18: Dijagram aktivnosti - registracija korisnika

Na ovom dijagramu prikazan je proces registracije novog korisnika. U procesu sudjeluju korisnik, sustav i baza podataka. Iz participija se ponovo jasno vidi koje aktivnosti pripadaju kojim aktorima, a iz samog dijagrama jasan je i slijed tih akcija. Započinjemo s korisnikom koji odabire opciju za registraciju. Sustav mu potom prikazuje formu za registraciju u koju će korisnik unositi podatke. Nakon što su podaci uneseni sustav ih šalje bazi na provjeru ispravnosti. Baza odlučuje jesu li podaci ispravno popunjeni ili ne i o tome ovisi sljedeći korak. Ukoliko podaci nisu bili ispravni sustav će ponovo prikazati formu i korisnik će ispraviti, tj. ponovo unijeti svoje podatke. Kod ispravnih podataka u bazu će se spremiti novi korisnik i sustav će korisniku poslati mail potvrde uspješne registracije. Po primitku potvrde od strane korisnika proces registracije je završen.

4.5 Dijagram komponenti

U dijagramu komponenti možemo vidjeti međuvisnosti između implementacijskih komponenti i zaviriti u internu strukturu te odnos programske potpore prema okolini. Slika 4.19 nam prikazuje organizaciju komponenti strukture cijele aplikacije.



Slika 4.19: Dijagram komponenti - struktura aplikacije

Na prezentacijskom sloju (frontendu) smještena je komponenta Web preglednik koja uz pomoć komponente React dohvata i šalje stvari prema sustavu. Komponenta React koristi sučelje Firebase koje kreira linkove za slike koje možemo vidjeti u oglasima. Sustav također komunicira s Bazom podataka na kojoj su pohranjeni svi korisnički računi i oglasi. Na sloju poslovne logike nalaze se komponente koje vrše komplikiranije izračune, dohvati i izmjenu podataka iz baze i odgovaranje na zahtjeve pristigne s prezentacijskog sloja. Komunikacija između prezentacijskog i sloja poslovne logike odvija se prema REST načelima. Za dohvatanje podataka

služi komponenta Sloj pristupa podacima preko koje se komunicira s Bazom podataka koja koristi JPA za prevodenje zahtjeva backenda u sintaksno ispravne SQL upite. Upiti se potom usmjeravaju prema bazi. Podaci zatim "putuju" natrag do frontenda i prikazuju se u pregledniku.

5. Implementacija i korisničko sučelje

5.1 Korištene tehnologije i alati

Kako bi što više ubrzali i olakšali komunikaciju među članovima tima, koristili smo dvije aplikacije: Whatsapp¹ i Discord². Kako bismo se dodatno upoznali s tehnologijom koja se često koristi u praksi, koristili smo i platformu Jira³ za organizaciju i podjelu zadataka na članove tima putem zadataka (engl. task) na Kanban ploči. Kao distribuirani sustav za upravljanje izvornim kodom koristili smo Git⁴, a sami repozitorij projekta dostupan je na platformi Gitlab⁵

Za razvoj korisničkog sučelja tj. frontenda općenito, korišteni su markup jezik HTML⁶, stilski jezik CSS⁷, programski jezik Javascript⁸ i React.js⁹. React.js knjižnica je pisana u jeziku Javascript koja omogućava jednostavan i relativno brz razvoj komponenti koje je lako ponovno koristiti, a razvija ga primarno tvrtka Meta. Uz React.js, za razvoj robusnih i kvalitetnih komponenti, korišten je Material UI¹⁰ - open-source knjižnica React komponenti. Za pisanje koda u navedenim jezicima korišten je Visual Studio Code¹¹ - veoma popularan IDE (integrirana razvojna okolina) s velikim brojem opcionalnih ekstenzija koje služe kao pomoć pri razvoju. VSC razvija tvrtka Microsoft.

¹<https://www.whatsapp.com/>

²<https://discord.com/>

³<https://www.atlassian.com/software/jira>

⁴<https://git-scm.com/>

⁵<https://gitlab.com/>

⁶<https://en.wikipedia.org/wiki/HTML>

⁷<https://en.wikipedia.org/wiki/CSS>

⁸<https://www.javascript.com/>

⁹<https://reactjs.org/>

¹⁰<https://mui.com/>

¹¹<https://code.visualstudio.com/>

Za razvoj backenda korišten je programski jezik Java¹² - objektno orijentirani jezik korišten u brojnim aspektima razvoja programske potpore. Jezik Java razvija tvrtka Oracle. Uz to, korišten je Java Spring Boot¹³ - specijalizacija radnog okvira Java Spring¹⁴ razvijena s ciljem što bržeg, lakšeg i učinkovitijeg razvoja web aplikacija. Za pisanje koda u jeziku Java korišten je IntelliJ IDEA¹⁵ - integrirana razvojna okolina često korištена za razvoj programske potpore u jezicima Java i Kotlin. IntelliJ IDEA razvija tvrtka JetBrains.

Za razvoj baze podataka korišten je sustav za upravljanje relacijskim bazama podataka PostgreSQL¹⁶. Uz to, za lokalni razvoj i testiranje baze podataka korištена je aplikacija pgAdmin¹⁷.

Za kreiranje potrebnih UML dijagrama korišten je alat Visual Paradigm Online¹⁸ koji omogućava zajednički rad na dijagramima, time olakšavajući organizaciju i dijeljenje istih s članovima tima. Za razvoj dokumentacije korišten je markup jezik LaTeX¹⁹, već navedena integrirana razvojna okolina Visual Studio Code i alat Overleaf²⁰ koji omogućava online pisanje LaTeX koda.

Frontend, backend i baza podataka pušteni su u pogon na platformi Render²¹ koja omogućava besplatan i relativno jednostavan deploy aplikacija. Za uspješno puštanje u pogon backend koda, potrebno je pripremiti i Docker²² Container kako bi na istom mjestu bilo dostupno sve potrebno za uspješno puštanje backend-a u pogon.

¹²<https://www.java.com/en/>

¹³<https://spring.io/projects/spring-boot>

¹⁴<https://spring.io/>

¹⁵<https://www.jetbrains.com/idea/>

¹⁶<https://www.postgresql.org/>

¹⁷<https://www.pgadmin.org/>

¹⁸<https://online.visual-paradigm.com/>

¹⁹<https://www.latex-project.org/>

²⁰<https://www.overleaf.com/>

²¹<https://render.com/>

²²<https://www.docker.com/>

5.2 Ispitivanje programskog rješenja

Kako bismo što učinkovitije testirali naš sustav, testirali smo ispravnost rada pojedinih komponenti koristeći unit testove, ali i ispravnost rada čitavog sustava koristeći alat Selenium.

5.2.1 Ispitivanje komponenti

Za testiranje rada komponenti, napisano je i pokrenuto 9 unit testova. Testovi su pisani u programskom jeziku Java koristeći okvir za testiranje JUnit. Na slikama se nalazi kod samih testova, kao i dokaz da testovi prolaze (redak Tests passed). Uz to, vidljivo je i trajanje izvođenja testova.

Test 1

Prvim testom provjerava se valjanost zapisa o kategoriji pri dohvaćanju potkategorije - ako je dohvaćena potkategorija "Lutke", u dohvaćenom podatku trebao bi postojati i zapis o pripadnoj kategoriji - "Igračke". Test prolazi.

The screenshot shows an IDE interface with two main sections. The top section displays Java code for a unit test named `testSubcategoryDatabase`. The code uses the `MockMvcRequestBuilders` class to perform a GET request to a URL template `/subcategory/{subcategoryName}` with `subcategoryName` set to `Lutke`. It then checks the response status and the JSON path value of the category name. The bottom section shows a test results summary for the file `SubcategoryControllerTest`. It indicates 1 test passed in 547 ms. The test listed is `testSubcategoryDatabase`, which also took 547 ms to run.

```
26
27     @Test
28     public void testSubcategoryDatabase() throws Exception{
29
30         mvc.perform( MockMvcRequestBuilders
31                         .get( urlTemplate: "/subcategory/{subcategoryName}", ...uriVariables: "Lutke")
32                         .characterEncoding("utf-8")
33                         .contentType(MediaType.APPLICATION_JSON)
34                         .accept(MediaType.APPLICATION_JSON))
35                         .andExpect(MockMvcResultMatchers.status().isOk())
36                         .andExpect(jsonPath( expression: "$.category.categoryName").value( expectedValue: "Igračke"));
37
38     }
```

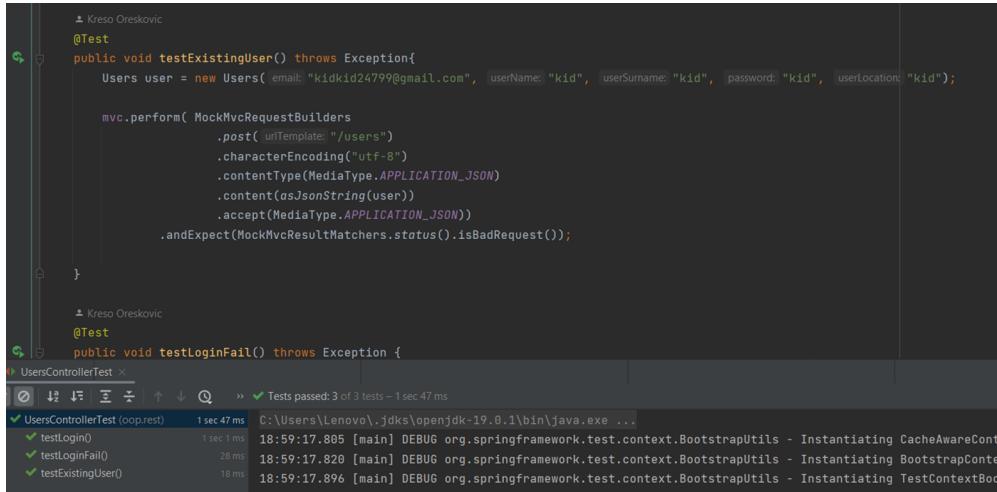
SubcategoryControllerTest.testSubcategoryDatabase x
Tests passed: 1 of 1 test – 547 ms

SubcategoryControllerTest (oop.rest) 547 ms
testSubcategoryDatabase() 547 ms

Slika 5.1: Kod prvog unit testa

Test 2

Drugim testom provjerava se dolazi li do pogreške ako pri registraciji korisnik unese mail adresu koja je već unešena u bazi - očekivan je status "400 Bad Request". Test prolazi.



```

    @Test
    public void testExistingUser() throws Exception{
        Users user = new Users( email:"kidkid24799@gmail.com", userName: "kid", userSurname: "kid", password: "kid", userLocation: "kid");

        mvc.perform( MockMvcRequestBuilders
                .post( "/users")
                .characterEncoding("utf-8")
                .contentType(MediaType.APPLICATION_JSON)
                .content(asJsonString(user))
                .accept(MediaType.APPLICATION_JSON))
                .andExpect(MockMvcResultMatchers.status().isBadRequest());
    }

    @Test
    public void testLoginFail() throws Exception {
    }

```

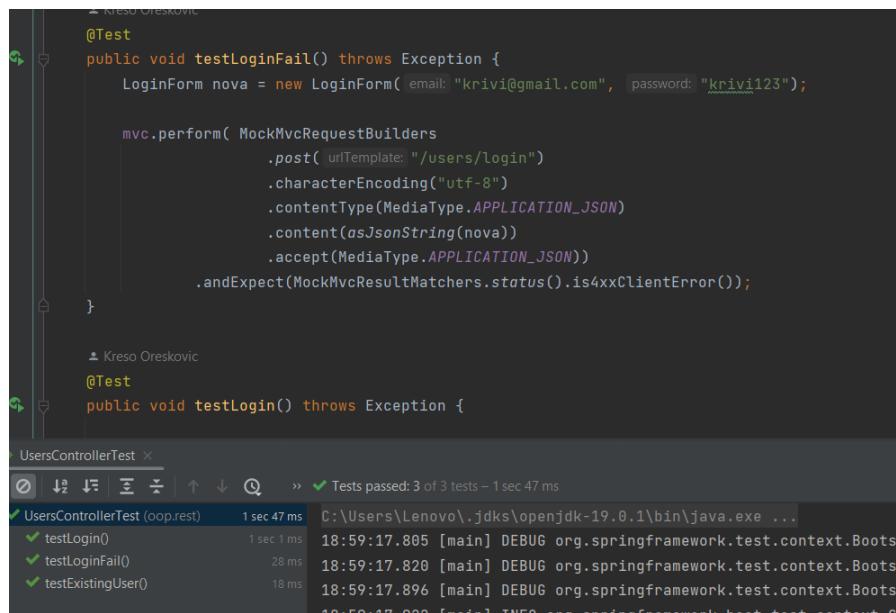
UsersControllerTest (oop.rest) 1 sec 47 ms C:\Users\Lenovo\.jdks\openjdk-19.0.1\bin\java.exe ...

- ✓ testLogin() 1 sec 1 ms 18:59:17.805 [main] DEBUG org.springframework.test.context.BootstrapUtils - Instantiating CacheAwareCont...
- ✓ testLoginFail() 28 ms 18:59:17.820 [main] DEBUG org.springframework.test.context.BootstrapUtils - Instantiating BootstrapCont...
- ✓ testExistingUser() 18 ms 18:59:17.896 [main] DEBUG org.springframework.test.context.BootstrapUtils - Instantiating TestContextBoo...

Slika 5.2: Kod drugog unit testa

Test 3

Trećim testom provjerava se dolazi li do pogreške ako pri loginu korisnik unese mail adresu za koju u bazi ne postoji zapis tj. podaci o korisniku. Test prolazi.



```

    @Test
    public void testLoginFail() throws Exception {
        LoginForm nova = new LoginForm( email: "krivi@gmail.com", password: "krivi123");

        mvc.perform( MockMvcRequestBuilders
                .post( "/users/login")
                .characterEncoding("utf-8")
                .contentType(MediaType.APPLICATION_JSON)
                .content(asJsonString(nova))
                .accept(MediaType.APPLICATION_JSON))
                .andExpect(MockMvcResultMatchers.status().is4xxClientError());
    }

    @Test
    public void testLogin() throws Exception {
    }

```

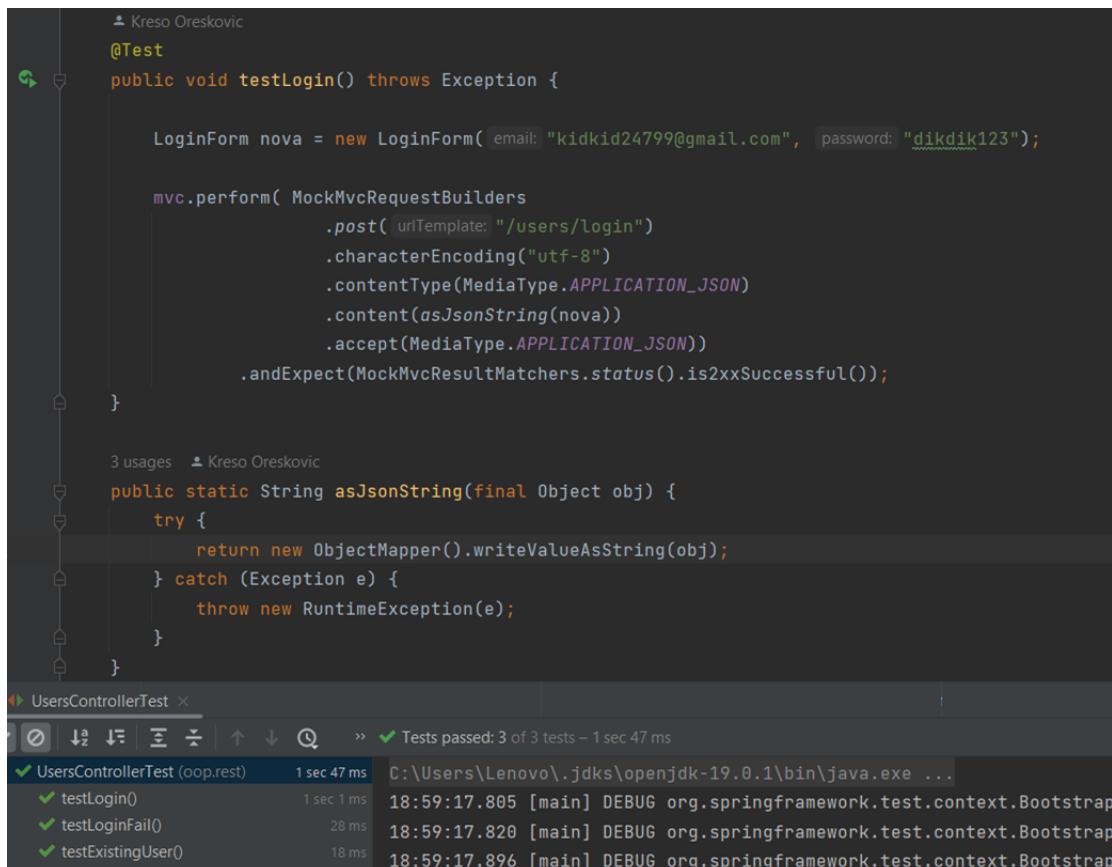
UsersControllerTest (oop.rest) 1 sec 47 ms C:\Users\Lenovo\.jdks\openjdk-19.0.1\bin\java.exe ...

- ✓ testLogin() 1 sec 1 ms 18:59:17.805 [main] DEBUG org.springframework.test.context.BootstrapUtils - Instantiating CacheAwareCont...
- ✓ testLoginFail() 28 ms 18:59:17.820 [main] DEBUG org.springframework.test.context.BootstrapUtils - Instantiating BootstrapCont...
- ✓ testExistingUser() 18 ms 18:59:17.896 [main] DEBUG org.springframework.test.context.BootstrapUtils - Instantiating TestContextBoo...

Slika 5.3: Kod trećeg unit testa

Test 4

Četvrtim testom provjerava se uspješnost logina korisnika za kojeg u bazi postoje podaci. Test prolazi.



```

    package Kreso Oreskovic;
    import org.junit.*;
    import static org.junit.Assert.*;
    import static org.springframework.test.web.servlet.request.MockMvcRequestBuilders.*;
    import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.*;

    public void testLogin() throws Exception {
        LoginForm nova = new LoginForm("kidkid24799@gmail.com", "dikdik123");

        mvc.perform(MockMvcRequestBuilders
                .post("/users/login")
                .characterEncoding("utf-8")
                .contentType(MediaType.APPLICATION_JSON)
                .content(asJsonString(nova))
                .accept(MediaType.APPLICATION_JSON))
                .andExpect(MockMvcResultMatchers.status().is2xxSuccessful());
    }

    public static String asJsonString(final Object obj) {
        try {
            return new ObjectMapper().writeValueAsString(obj);
        } catch (Exception e) {
            throw new RuntimeException(e);
        }
    }
}

```

UsersControllerTest x

Tests passed: 3 of 3 tests – 1 sec 47 ms

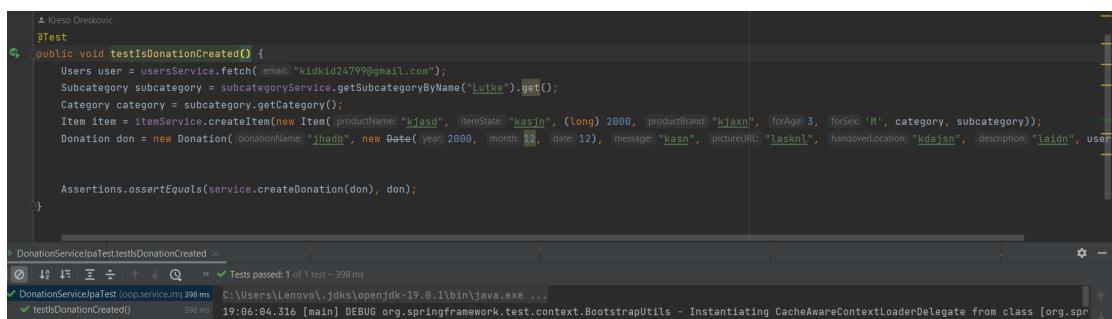
UsersControllerTest (oop.rest)	1 sec 47 ms
✓ testLogin()	1 sec 1 ms
✓ testLoginFail()	28 ms
✓ testExistingUser()	18 ms

C:\Users\Lenovo\.jdks\openjdk-19.0.1\bin\java.exe ...
18:59:17.805 [main] DEBUG org.springframework.test.context.Bootstrap
18:59:17.820 [main] DEBUG org.springframework.test.context.Bootstrap
18:59:17.896 [main] DEBUG org.springframework.test.context.Bootstrap

Slika 5.4: Kod četvrtoog unit testa

Test 5

Petim testom provjerava se uspješnost stvaranja donacije. Test prolazi.



```

    package Kreso Oreskovic;
    import org.junit.*;
    import static org.junit.Assert.*;
    import static org.springframework.test.web.servlet.request.MockMvcRequestBuilders.*;
    import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.*;

    public void testIsDonationCreated() {
        Users user = usersService.fetch("kidkid24799@gmail.com");
        Subcategory subcategory = subcategoryService.getSubcategoryByName("Lutke").get();
        Category category = subcategory.getCategory();

        Item item = itemService.createItem(new Item("Kjasa", "kasja", (long) 2000, "kjaxn", "M", category, subcategory));
        Donation don = new Donation("jhadi", new Date(2000, 12, 12), "kasa", "laskn", "kdajsn", "laidn", user);

        Assertions.assertEquals(service.createDonation(don), don);
    }
}

```

DonationServiceIpaTest x

Tests passed: 1 of 1 test – 398 ms

DonationServiceIpaTest (oop.service)	398 ms
✓ testIsDonationCreated()	398 ms

C:\Users\Lenovo\.jdks\openjdk-19.0.1\bin\java.exe ...
19:06:04.316 [main] DEBUG org.springframework.test.context.BootstrapUtils - Instantiating CacheAwareContextLoaderDelegate from class [org.spr

Slika 5.5: Kod petog unit testa

Testovi 6, 7

Šestim testom provjerava se uspješnost ažuriranja podataka o predmetu. Test prolazi. Sedmim testom provjerava se uspješnost brisanja predmeta iz baze podataka, kao i dolazi li do "custom" pogreške "EntityMissingException" pri pokušaju dohvaćanja izbrisaniog predmeta. Test prolazi.

```

    @Test
    public void testIsItemUpdated() {
        Subcategory subcategory = subcategoryService.getSubcategoryByName("Lutke").get();
        Category category = subcategory.getCategory();
        Item createdItem = itemService.createItem(new Item(productName: "jadbh", itemState: "kadna", (long) 200, productBrand: "akdn", forAge: 2, forSex: 'M', category, subcategory));
        createdItem.setForSex('F');
        Assertions.assertEquals(expected: 'F', itemService.updateItem(createdItem).getForSex());
    }

    ▲ Kreso Oreskovic
    @Test
    public void testIsItemDeleted() {
        Subcategory subcategory = subcategoryService.getSubcategoryByName("Lutke").get();
        Category category = subcategory.getCategory();
        Item createdItem = itemService.createItem(new Item(productName: "jadbh", itemState: "kadna", (long) 200, productBrand: "akdn", forAge: 2, forSex: 'M', category, subcategory));
        Long id = createdItem.getId();
        itemService.delete(createdItem);
        Throwable exception = Assertions.assertThrows(EntityMissingException.class, () -> itemService.getItemIdById(id));
        Assertions.assertEquals(exception.getMessage(), String.format("Entity with reference %d of class oop.domain.Item not found.", id));
    }
}

```

ItemServiceJpaTest x

Tests passed: 2 of 2 tests – 495 ms

itemServiceJpaTest (oop.service.impl) 495 ms C:\Users\Lenovo\jdk\openjdk-19.0.1\bin\java.exe ...
 ✓ testIsItemUpdated() 462 ms 19:08:36.632 [main] DEBUG org.springframework.test.context.BootstrapUtils - Instantiating CacheAwareContextLoaderDelegate from class [org.springfram...
 ✓ testIsItemDeleted() 33 ms 19:08:36.653 [main] DEBUG org.springframework.test.context.BootstrapUtils - Instantiating BootstrapContext using constructor [public org.springfram...
 19:08:36.742 [main] DEBUG org.springframework.test.context.TestContextBootstrapper - Instantiating TestContextBootstrapper for test class [oop.domain.ItemServiceJpaTest]

Slika 5.6: Kod šestog i sedmog unit testa

Testovi 8, 9

Osmim testom provjerava se postoji li admin račun u bazi podataka. Test prolazi. Devetim testom provjerava se dolazi li do "custom" pogreške "EntityMissingException" pri pokušaju dohvaćanja nepostojećeg korisnika. Test prolazi.

```

    @Test
    public void TestIsAdminInDatabase() {
        Users users = usersService.fetch(email: "admin");
        Assertions.assertEquals(passwordEncoder.matches(rawPassword: "pass", users.getPassword()), actual: true);
    }

    ▲ Kreso Oreskovic
    @Test
    public void TestEntityMissingException() {
        Throwable exception = Assertions.assertThrows(EntityMissingException.class, () -> usersService.fetch(email: "ksj@gmail.com"));
        Assertions.assertEquals(exception.getMessage(), actual: "Entity with reference ksj@gmail.com of class oop.domain.Users not found.");
    }
}

```

ItemServiceJpaTest x

Tests passed: 2 of 2 tests – 428 ms

itemServiceJpaTest (oop.service.impl) 428 ms C:\Users\Lenovo\jdk\openjdk-19.0.1\bin\java.exe ...
 ✓ testIsItemUpdated() 399 ms 19:11:55.028 [main] DEBUG org.springframework.test.context.BootstrapUtils - Instantiating CacheAwareContextLoaderDelegate from class [org.springfram...
 ✓ testIsItemDeleted() 29 ms 19:11:55.044 [main] DEBUG org.springframework.test.context.BootstrapUtils - Instantiating BootstrapContext using constructor [public org.springfram...
 19:11:55.098 [main] DEBUG org.springframework.test.context.TestContextBootstrapper - Instantiating TestContextBootstrapper for te

Slika 5.7: Kod osmog i devetog unit testa

5.2.2 Ispitivanje sustava

Za testiranje rada cjelovitog sustava, korišteni su alati Selenium Webdriver i Selenium IDE. Selenium Webdriver omogućava pisanje testova cjelovitog sustava koristeći proizvoljan programski jezik, kao i integraciju tih testova s unit testovima napisanim za testiranje rada pojedinačnih komponenti. S druge strane, Selenium IDE korisnicima omogućava generiranje testova cjelovitog sustava snimanjem svojih akcija i ponavljanjem tih akcija.

U našem testiranju, testirali smo funkcionalnosti aplikacije koja je već puštena u pogon (engl. deployana). Pritom smo za jednostavnije funkcionalnosti ručno pisali testove u programskom jeziku Java, dok smo za neke od složenijih funkcionalnosti snimili akcije koristeći alat Selenium IDE. Na slikama vidljiv je kod testova pisanih koristeći Selenium Webdriver, kao i akcije snimljene alatom Selenium IDE. Testovi pisani u programskom jeziku Java izvršeni su zajedno, a rezultat njihovog izvođenja prikazan je na zasebnoj slici.

Test 1

Prvim testom provjerava se registracija koristeći već unesenu mail adresu tj. adresu za koju u bazi već postoji zapis o korisniku. Očekivani rezultat izvođenja testa je da ne dolazi do preusmjeravanja na stranicu s prikazom aktivnih oglasa, već je i dalje prikazana stranica za registraciju (s porukom o neuspješnoj registraciji). Test prolazi.

```
@Test
public void testRegistrationAlreadyExisting() {
    WebDriver driver = new ChromeDriver();
    System.setProperty(key: "webdriver.chrome.driver", value: "C:\\\\Program Files (x86)\\\\Chrome Driver\\\\chromedriver.exe");
    driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(seconds: 2));
    driver.get(url: "https://frontend-service.onrender.com/");

    driver.findElement(By.cssSelector(cssSelector: ".svijetliji")).click();
    driver.findElement(By.id(id: "ime")).sendKeys(...keysToSend: "Ivana");
    driver.findElement(By.id(id: "prezime")).sendKeys(...keysToSend: "Horvatic");
    driver.findElement(By.id(id: "mjesto")).sendKeys(...keysToSend: "Ulica Horvatića");
    driver.findElement(By.id(id: "email")).sendKeys(...keysToSend: "ivanahorvatic@gmail.com");
    driver.findElement(By.id(id: "pass")).sendKeys(...keysToSend: "ivanahorvatic");
    driver.findElement(By.cssSelector(cssSelector: ".tamniji")).click();

    boolean unsuccessfulRegister = driver.findElements(By.linkText(linkText: "Aktivne donacije")).size() == 0;

    assertEquals(unsuccessfulRegister, actual: true);

    driver.quit();
}
```

Slika 5.8: Kod prvog Selenium testa

Test 2

Drugim testom provjerava se registracija koristeći ispravne podatke tj. podatke za koje u bazi do sada ne postoji zapis o korisniku. Očekivani rezultat izvođenja testa je da dolazi do preusmjeravanja na stranicu s prikazom aktivnih oglasa koja u zaglavlju ima poveznice na ostale dijelove stranice. Test prolazi.

```
@Test
public void testRegistrationSuccessful() {
    WebDriver driver = new ChromeDriver();
    System.setProperty(key: "webdriver.chrome.driver", value: "C:\\Program Files (x86)\\Chrome Driver\\chromedriver.exe");
    driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(seconds: 4));
    driver.get(url: "https://frontend-service.onrender.com/");

    driver.findElement(By.cssSelector(cssSelector: ".svijetlij").click());
    driver.findElement(By.id(id: "ime")).sendKeys(...keysToSend: "Ivica");
    driver.findElement(By.id(id: "prezime")).sendKeys(...keysToSend: "Horvatic");
    driver.findElement(By.id(id: "mjesto")).sendKeys(...keysToSend: "Ulica Horvatića");
    driver.findElement(By.id(id: "email")).sendKeys(...keysToSend: "ivicahorvatic@gmail.com");
    driver.findElement(By.id(id: "pass")).sendKeys(...keysToSend: "ivicahorvatic");
    driver.findElement(By.cssSelector(cssSelector: ".tamniji").click());

    boolean registeredSuccessfully = driver.findElements(By.linkText(linkText: "Aktivne donacije")).size() == 1;

    assertEquals(registeredSuccessfully, actual: true);
    driver.quit();
}
```

Slika 5.9: Kod drugog Selenium testa

Test 3

Trećim testom provjerava se login korisnika koristeći neispravne podatke (kombinacija ispravna mail adresa, neispravna lozinka korisnika). Očekivani rezultat izvođenja testa je da ne dolazi do preusmjeravanja na stranicu s prikazom aktivnih oglasa, već je i dalje prikazana stranica za login (s porukom o neuspješnoj prijavi). Test prolazi.

```
@Test
public void testLoginBadCreds() {
    WebDriver driver = new ChromeDriver();
    System.setProperty(key: "webdriver.chrome.driver", value: "C:\\Program Files (x86)\\Chrome Driver\\chromedriver.exe");
    driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(seconds: 2));
    driver.get(url: "https://frontend-service.onrender.com/");

    driver.findElement(By.cssSelector(cssSelector: ".tamniji").click();

    driver.findElement(By.id(id: "email")).sendKeys(...keysToSend: "ivanhorvat@gmail.com");
    driver.findElement(By.id(id: "pass")).sendKeys(...keysToSend: "krivipass");

    driver.findElement(By.cssSelector(cssSelector: ".tamniji").click();

    boolean unsuccessfulLogin = driver.findElements(By.linkText(linkText: "Aktivne donacije")).size() == 0;

    assertEquals(unsuccessfulLogin, actual: true);
    driver.quit();
}
```

Slika 5.10: Kod trećeg Selenium testa

Test 4

Četvrtim testom provjerava se login korisnika koristeći ispravne podatke (ispravna mail adresa, ispravna lozinka korisnika). Očekivani rezultat izvođenja testa je da dolazi do preusmjeravanja na stranicu s prikazom aktivnih oglasa koja u zaglavlju ima poveznice na ostale dijelove stranice. Test prolazi.

```
@Test
public void testLoginGoodCreds() {
    WebDriver driver = new ChromeDriver();
    System.setProperty(key: "webdriver.chrome.driver", value: "C:\\Program Files (x86)\\Chrome Driver\\chromedriver.exe");
    driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(seconds: 2));
    driver.get(url: "https://frontend-service.onrender.com/");

    driver.findElement(By.cssSelector(cssSelector: ".tammiji")).click();

    driver.findElement(id: "email").sendKeys(...keysToSend: "ivanhorvat@gmail.com");
    driver.findElement(id: "pass").sendKeys(...keysToSend: "ivanhorvat");

    driver.findElement(By.cssSelector(cssSelector: ".tammiji")).click();

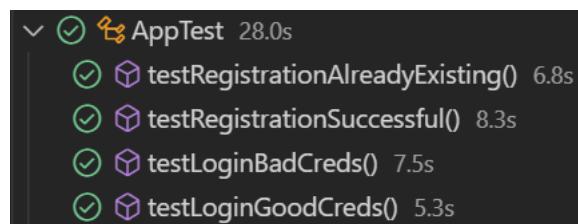
    boolean sucessfulLogin = driver.findElements(By.linkText(linkText: "Aktivne donacije")).size() == 1;
    assertEquals(sucessfulLogin, actual: true);

    driver.quit();
}
```

Slika 5.11: Kod četvrtog Selenium testa

Rezultati izvođenja Selenium testova

Na slici 5.12 prikazani su rezultati izvođenja do sada opisanih testova. Sva četiri testa uspješno prolaze, a na slici je prikazano i trajanje pojedinih testova.



Slika 5.12: Rezultati izvođenja Selenium testova

Test 5

Na slici 5.13 prikazane su akcije snimljene koristeći alat Selenium IDE. Alat Selenium IDE korišten je kako bi se preglednije prikazale potrebne akcije - kod za testiranje složenih funkcionalnosti dugačak je i nepregledniji od ovakvog prikaza. Testirano je dodavanje nove kategorije. Kako bi se dodala nova kategorija, korisnik se mora prijaviti koristeći admin podatke. Tijekom provođenja testa, ti su podaci već bili pohranjeni u pregledniku pa je bilo dovoljno kliknuti na gumb za prijavu. Nakon uspješne prijave, dolazimo do stranice za dodavanje kategorija i potkategorija te pokušavamo stvoriti novu kategoriju proizvoljnog imena. Test prolazi.

✓ DodavanjeKategorije*	Command	Target	Value
1	✓ open	/	
2	✓ set window size	1132x900	
3	✓ click	css=.jamniji	
4	✓ click	linkText=Moj profil	
5	✓ click	css=a:nth-child(4) > .gumbic	
6	✓ click	id=imeKategorije	
7	✓ mouse over	css=.MuiButtonBase-root:nth-child(3)	
8	✓ type	id=imeKategorije	Pomagala za prehranu
9	✓ click	css=.MuiButtonBase-root:nth-child(3)	
10	✓ mouse out	css=.MuiButtonBase-root:nth-child(3)	

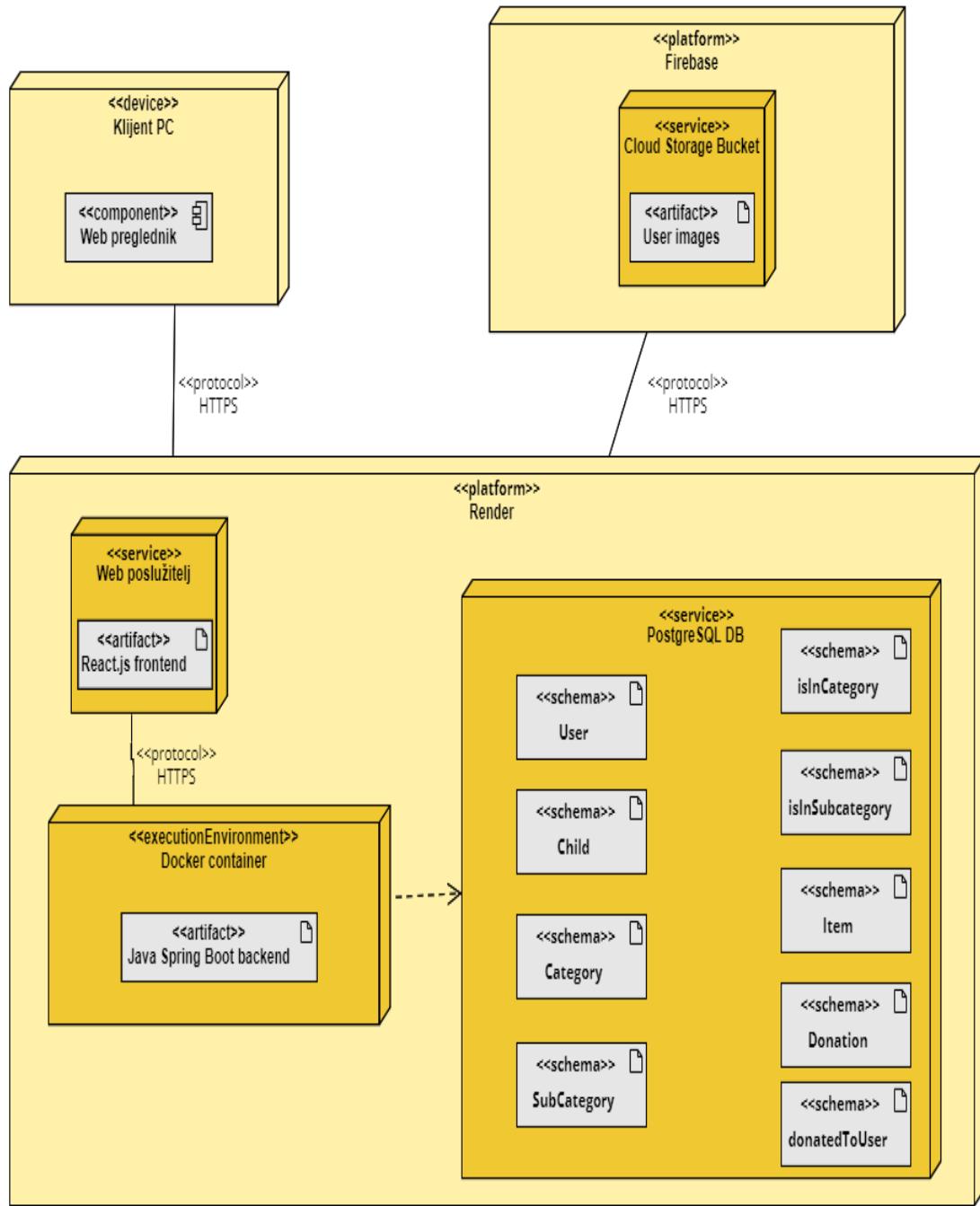
Slika 5.13: Akcije petog Selenium testa

Uz pet do sada prikazanih Selenium testova, dodatno su testirane i funkcionalnosti stvaranja oglasa, dodavanja potkategorije i uređivanja podataka, ali zbog preglednosti i sažetosti ti testovi nisu uključeni u dokumentaciju.

5.3 Dijagram razmještaja

Dijagram razmještaja opisuje konkretno ostvarenje odabrane arhitekture aplikacije - na slici 5.14 moguće je vidjeti raspodjelu programske potpore na konkretno sklopolje. Kod za prikaz korisničkog sučelja i komunikaciju s korisnikom pisan u React.js pušten je u pogon na platformi Render. Kako bi aplikacija bila funkcionalna, u pogon su zasebno pušteni (engl. deployani) baza podataka ostvarena koristeći PostgreSQL te kod za poslovnu logiku i komunikaciju s bazom (backend) pisan u Java Spring Boot-u. Kako bi kod za backend bio uspješno deployan, za njega postoji i Docker container koji osigurava da je u njemu ispravno "spakiran" sav kod, kao i zavisnosti potrebne za uspješan rad istoga. I backend i baza podataka pušteni su u pogon koristeći platformu Render.

Uz ovo, na platformi Firebase napravljen je Cloud Storage Bucket koji služi za jednostavnu pohranu slika potrebnih za uspješan prikaz oglasa. Komunikacija između zasebnih uređaja i platformi osigurana je protokolom HTTPS (komunikacija između klijentovog računala tj. web preglednika na istome i web poslužitelja deployanog na platformi Render, kao i komunikacija između web poslužitelja i Cloud Storage Bucket-a deployanog na platformi Firebase). Kako bi web poslužitelj korisniku mogao poslužiti sav potreban sadržaj, on komunicira s backendom koji po potrebi komunicira s bazom podataka.



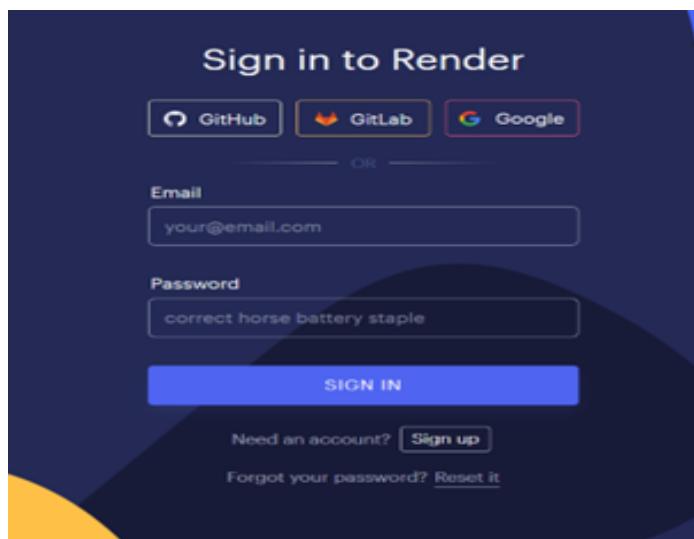
Slika 5.14: Dijagram razmještaja

5.4 Upute za puštanje u pogon

Frontend, backend i baza podatka puštene su u pogon na platformi Render. Upute za pristupanje platformi Render, kao i upute za puštanje u pogon (engl. deploy) slijede u sljedećim sekcijama.

5.4.1 Pristupanje Renderu

Kako bi pristupili Renderu, prilikom prijave u web aplikaciju koristimo GitLab račune koji se direktno mogu povezati s njime. Povezivanjem Gitlab računa s platformom Render dobivamo lakši pristup kodu koji će se pustiti u pogon.



Slika 5.15: Render sign in

5.4.2 Backend deploy na Render

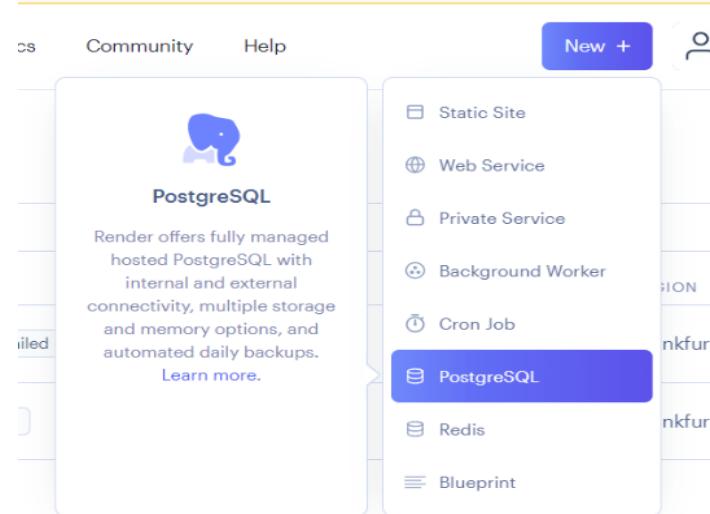
1. Priprema

Prije puštanja aplikacije u pogon na Renderu potrebno je u izvorni kod za backend dodati Dockerfile za Maven projekt te u datoteci application.properties postaviti property server.servlet.context-path na /api da bude prefiks svim zahtjevima na backend.

2. Kreiranje baze podataka

U render dashboard potrebno je:

- Pritisnuti tipku New te onda odabratи opciju PostgreSQL za kreiranje baze



Slika 5.16: Kreiranje nove PostgreSQL baze

- Postaviti ime baze
- Database i User ostaviti prazno jer se automatski generira pri kreiranju baze
- Postaviti Region na Frankfurt
- Pritisnuti Create Database

New PostgreSQL

Name	<input type="text" value="db"/>
Database	<input type="text" value="randomly generated unless specified"/>
User	<input type="text" value="randomly generated unless specified"/>
Region	<input type="text" value="Frankfurt (EU Central)"/>
PostgreSQL Version	<input type="text" value="15"/>
Datadog API Key	<input type="text"/>

Slika 5.17: Ispunjavanje podataka o bazi

Prije puštanja backend-a u pogon, u datoteci application.properties moramo uvrstiti username, password za pristup bazi, database, hostname i port za url prema bazi na koju se treba spojiti.

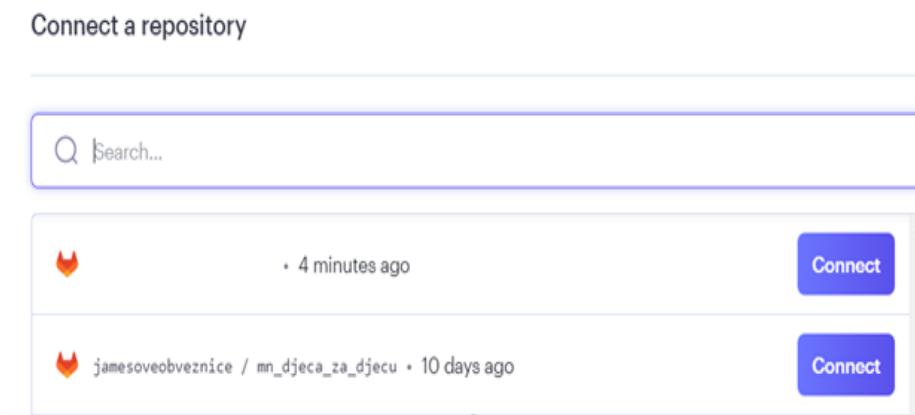


Slika 5.18: Ispunjavanje podataka u datoteci application.properties

3. Kreiranje backenda

U Render dashboard potrebno je:

- Pritisnuti tipku New te onda odabratи opciju Web Service za kreiranje web usluge
- Povezati GitLab račun - nakon povezivanja su za odabir dostupni svi projekti na koje imate prava pristupa
- Pritisnuti Connect pokraj odgovarajućeg projekta



Slika 5.19: Povezivanje repozitorija

- Postaviti ime za servis (ime servisa na kraju će biti dio web adrese)
- Root directory postaviti na IzvorniKod/backend
- Environment postaviti na Docker
- Region postaviti na Frankfurt

You are deploying a web service for [jamesoveobveznice/mn_djeca_za_djecu](#).

The screenshot shows the deployment configuration for a web service named "backend-service" in the "IzvorniKod" repository. The service is deployed to the "Frankfurt (EU Central)" region and uses the "main" branch. The "Root Directory" is set to "IzvorniKod/backend", and the "Environment" is set to "Docker". The "Name" field is also visible.

Slika 5.20: Podaci za puštanje web servisa u pogon

- Na dnu proširiti *advanced*
- Postaviti putanju za Dockerfile ovisno koji se package manager koristi (u ovom slučaju putanja je `./docker/maven/Dockerfile`)



Slika 5.21: Putanja Dockerfile-a

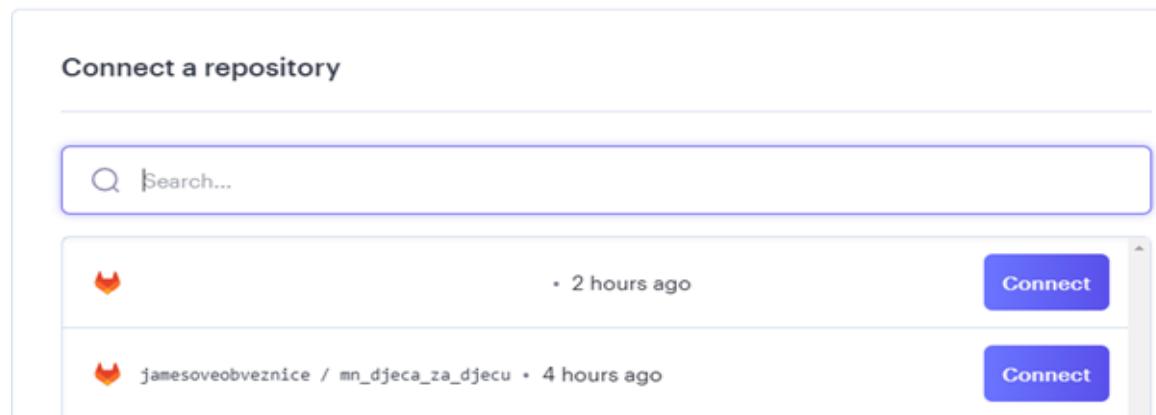
- Pritisnuti Create Web Service

Nakon izvođenja svih ovih koraka, u pogon bi se trebala uspješno pustiti (eng. deployati) backend aplikacija. Nakon puštanja backend aplikacije u pogon, isto ćemo napraviti i za frontend aplikaciju.

5.4.3 Frontend deploy na Render

Kako bi se frontend uspješno pustio u pogon, u Render dashboard potrebno je:

- Pritisnuti tipku New i odabratи opciju Web Service za kreiranje web usluge
- Povezati GitLab račun - nakon povezivanja su za odabir dostupni svi projekti na koje imate prava pristupa
- Pritisnuti Connect pokraj odgovarajućeg projekta



Slika 5.22: Povezivanje repozitorija za frontend

- Postaviti ime za servis (ime servisa na kraju će biti dio web adrese)
- Root directory postaviti na IzvorniKod/Frontend/dzd-app/
- Environment postaviti na Node

Name	Frontend
Region	Frankfurt (EU Central)
Branch	develop
Root Directory	IzvorniKod/Frontend/dzd-app/
Environment	Node

Slika 5.23: Ispunjavanje podataka o frontendu

- Region postaviti na Frankfurt
- Odabratи branch iz kojeg se uzimaju datoteke za frontend
- Build Command postaviti na yarn build
- Start Command postaviti na yarn start-prod

The screenshot shows the Render dashboard interface. It displays two command configurations:

- Build Command:** This command runs in the root directory of your repository when a new version of your code is pushed, or when you deploy manually. It is typically a script that installs libraries, runs migrations, or compiles resources needed by your app. The command is set to '\$ yarn build'.
- Start Command:** This command runs in the root directory of your app and is responsible for starting its processes. It is typically used to start a webserver for your app. It can access environment variables defined by you in Render. The command is set to '\$ yarn start-prod'.

Slika 5.24: Unošenje ispravnih naredbi

- Na dnu pritisnuti Advanced
- Dodati potrebne Environment varijable - API_BASE_URL postaviti na adresu deployanog backenda aplikacije dostupnu na Render dashboardu
- Isključiti Auto-Deploy kako se projekt ne bi ponovno deployao za svaki novi commit
- Pritisnuti Create Web Service

Nakon izvođenja svih ovih koraka, u pogon bi se trebala uspješno pustiti (eng. deployati) frontend aplikacija. Stranici pristupamo preko URL-a za frontend²³.

²³<https://frontend-service-3f78.onrender.com/>

6. Zaključak i budući rad

Zadatak naše grupe bio je razvoj web aplikacije za razmjenu dječjih predmeta i stvari. Cilj aplikacije je bio omogućiti lakše doniranje dječjih predmeta, raznih kategorija od igračaka do kolica i robe. Nakon 11 tjedana intenzivnog rada na projektu uspjeli smo ostvariti sve zahtijevane funkcionalnosti i poneku više. Projekt je bio organiziran u dvije velike faze, koje su, jasno, bile podijeljene na manje faze radi boljeg održavanja tempa i jasnijih ciljeva.

Prva faza započela je formiranjem tima od sedmoro studenata za koji će u narednih 11 tjedana razvijati aplikaciju, učiti odabrane alate, pisati dokumentaciju i naučiti održavati sastanke. Dodjela projektnog zadatka i upoznavanje asistentice uslijedila je ubrzo nakon. Bilo je važno dobro analizirati zadatak i izvući sve zahtjeve iz opisa zadatka. U prvoj fazi bio je veći naglasak na izradi dokumentacije koja je predstavljala ključ priprema za implementiranje aplikacije. Iz pažljivo prepoznatih funkcionalnih zahtjeva napravljeni su dijagrami obrazaca uporabe i sekvencijalski dijagrami koji su bili referentni "dokument" za implementaciju. Promišljenim modelom baze podataka također je olakšana i ubrzana implementacija baze. Jednom kada je promišljanje o arhitekturi i izgledu aplikacije bilo gotovo moglo se krenuti u drugu fazu u kojoj nije bilo vremena za gubiti i nismo ga izgubili jer smo imali dobru pripremu.

Druga faza je trajala kraće, ali je bila nešto intenzivnija od prve što se tiče samostalnog učenja tehnologija i kodiranja. Dokumentacija se razvijala paralelno s aplikacijom i u nju je trebalo dodati preostale UML dijagrame vezane za arhitekturu sustava. Sastanci tima postali su češći kako bi bili sigurni da se sve odvija po planu. Izrađen je i dio dokumentacije koji će omogućiti budućim korisnicima lakše snalaženje i korištenje aplikacije kao i eventualnu nadogradnju sustava.

Zadovoljni smo završnom verzijom aplikacije i suradnjom koja je trajala ovih 11 tjedana. Imamo ideje za budući rad u vidu vizualnih poboljšanja, ali i tehničkih/funkcionalnih poboljšanja. Glavna ideja bila bi dodati kartu u oglas kako bi se

odmah prikazivala adresu na mapi i mogla otvoriti karta u Google Maps aplikaciji. Također bi time olakšali posao adminu koji ne bi morao provjeravati postoje li adresa jer se u kreiranju oglasa ne bi mogla dodati adresa koju Google Maps ne prepoznaće. Druga ideja bila bi sustav za prijavljanje korisnika koji bi adminu olakšao dodijeljivanje prava na kreiranje oglasa kao i povećao zadovoljstvo korisnika. Kako bi aplikacija bila kompletna ideja je još dodati "chat" unutar aplikacije između osobe koja donira i osobe koja želi preuzeti donaciju. Imamo još puno sličnih ideja, ali ove su glavne pa smo njih odlučili spomenuti.

Kroz ovaj projekt smo naučili da je dobar tim i komunikacija u timu izrazito važna. Ispravna (kako se pokazalo kasnije) podjela u podtimove i procjena količine posla učinile su da tim funkcionira jako dobro. Komunicirali smo često putem WhatsApp grupe, a sastanke smo, osim uživo, imali preko Discorda. Motivacija tima je bila visoka i nije bilo problema između članova što je značajno za dobar napredak projekta.

Rad na ovakovom projektu nam je pokazao koliko je važno znati organizirati vrijeme i izvršavati svoje zadatke na vrijeme, ali i koliko znači imati prave ljude u timu i dobro vodstvo. Kontrole s mentoricom su nam puno značile i njezini komentari koji su nas usmjeravali i motivirali da budemo što bolji kao tim i razvijemo što kvalitetniju aplikaciju.

Popis literature

1. Programsko inženjerstvo, FER ZEMRIS, <http://www.fer.hr/predmet/proinzh>
2. I. Sommerville, "Software engineering", 8th ed, Addison Wesley, 2007.
3. T.C.Lethbridge, R.Langaniere, "Object-Oriented Software Engineering", 2nd ed. McGraw-Hill, 2005.
4. I. Marsic, Software engineering book“, Department of Electrical and Computer Engineering, Rutgers University, <http://www.ece.rutgers.edu/~marsic/books/SE>
5. The Unified Modeling Language, <https://www.uml-diagrams.org/>
6. Visual Paradigm, <https://www.visual-paradigm.com/>

Indeks slika i dijagrama

2.1	Oglas za mobilnu aplikaciju BIPO CLUB	7
2.2	"Landing page" aplikacije BIPO CLUB	7
2.3	Pregled vlastitih oglasa u aplikaciji BIPO CLUB	8
2.4	Pregled aktivnih oglasa u aplikaciji BIPO CLUB	8
2.5	Chat room s donatorom u aplikaciji BIPO CLUB	9
2.6	Njuškalo (lijevo) i Facebook grupa (desno)	10
3.1	Dijagram obrazaca uporabe - aplikacija Djeca za djecu	35
3.2	Sekvencijski dijagram - prijava i registracija	36
3.3	Sekvencijski dijagram - upravljanje oglasima	38
4.1	Arhitektura sustava	41
4.2	ER dijagram - aplikacija Djeca za djecu	50
4.3	Relacijski dijagram - aplikacija Djeca za djecu	51
4.4	Dijagram razreda - dio Models	52
4.5	Dijagram razreda - odnos komponenti	53
4.6	Dijagram razreda - dio Controllers	54
4.7	Dijagram razreda - dio Repositories	55
4.8	Dijagram razreda - dio Models (finalna verzija)	56
4.9	Dijagram razreda - odnos komponenti (finalna verzija)	57
4.10	Dijagram razreda - dio Controllers, prvi dio (finalna verzija)	58
4.11	Dijagram razreda - dio Controllers, drugi dio (finalna verzija)	58
4.12	Dijagram razreda - dio Repositories, prvi dio (finalna verzija)	59
4.13	Dijagram razreda - dio Repositories, drugi dio (finalna verzija)	59
4.14	Dijagram razreda - dio Email Service	60
4.15	Dijagram razreda - dio Security	60
4.16	Dijagram stanja	61
4.17	Dijagram aktivnosti - proces kreiranja oglasa	63
4.18	Dijagram aktivnosti - registracija korisnika	65
4.19	Dijagram komponenti - struktura aplikacije	66

5.1 Kod prvog unit testa	70
5.2 Kod drugog unit testa	71
5.3 Kod trećeg unit testa	71
5.4 Kod četvrtog unit testa	72
5.5 Kod petog unit testa	72
5.6 Kod šestog i sedmog unit testa	73
5.7 Kod osmog i devetog unit testa	73
5.8 Kod prvog Selenium testa	74
5.9 Kod drugog Selenium testa	75
5.10 Kod trećeg Selenium testa	75
5.11 Kod četvrtog Selenium testa	76
5.12 Rezultati izvođenja Selenium testova	76
5.13 Akcije petog Selenium testa	77
5.14 Dijagram razmještaja	79
5.15 Render sign in	80
5.16 Kreiranje nove PostgreSQL baze	81
5.17 Ispunjavanje podataka o bazi	81
5.18 Ispunjavanje podataka u datoteci application.properties	82
5.19 Povezivanje repozitorija	82
5.20 Podaci za puštanje web servisa u pogon	83
5.21 Putanja Dockerfile-a	83
5.22 Povezivanje repozitorija za frontend	84
5.23 Ispunjavanje podataka o frontendu	84
5.24 Unošenje ispravnih naredbi	85
6.1 Prikaz commitova, prvi dio	98
6.2 Prikaz commitova, drugi dio	99

Dodatak: Prikaz aktivnosti grupe

Dnevnik sastajanja

1. sastanak

- Datum: 14. listopada 2022
- Prisustvovali: D.Kiramarios, D.Kovačević, K.Tomičić, D.Jambrović, K.Orešković, K.Kijac, S.Barac
- Teme sastanka:
 - raspodjela uloga (podjela poslova na frontend, backend i razvoj dokumentacije)
 - rasprava o zadanoj temi
 - sakupljanje pitanja za prvi sastanak sa asistenticom

2. sastanak

- Datum: 20. listopada 2022
- Prisustvovali: D.Kiramarios, D.Kovačević, K.Tomičić, D.Jambrović, K.Orešković, K.Kijac, S.Barac
- Teme sastanka:
 - osnovni podaci o izvođenju projekta
 - postavljanje pitanja vezanih za implementaciju projekta asistentici tijekom prvih pola sata sastanka i sljedećih 20 min

3. sastanak

- Datum: 20. listopada 2022
- Prisustvovali: D.Kiramarios, D.Kovačević, K.Tomičić, D.Jambrović, K.Orešković, K.Kijac, S.Barac
- Teme sastanka:
 - detaljna analiza zadatka
 - planiranje raspodjele poslova tijekom sljedećeg tjedna vezanih za inicijalizaciju backenda, frontenda i dokumentacije
 - dogovor o korištenju Jire za bolje praćenje raspodjele poslova

4. sastanak

- Datum: 23. listopada 2022
- Prisustvovali: D.Kiramarios, D.Kovačević, K.Tomičić, D.Jambrović, K.Orešković, K.Kijac, S.Barac
- Teme sastanka:
 - pregled predložene skice klijentske aplikacije (frontend)
 - pregled predložene sheme za bazu podataka (backend)
 - pregled obrazaca uporabe i UC dijagrama (dokumentacija)
 - zaključak: male promjene sheme baze i slanje na pregled asistentici

5. sastanak

- Datum: 24. listopada 2022
- Prisustvovali: D.Kiramarios, K.Tomičić, S.Barac
- Teme sastanka:
 - detaljna analiza prijedloga dizajna za klijentsku aplikaciju
 - male promjene dizajna za klijentsku aplikaciju
 - podjela posla na klijentskoj aplikaciji među prisutnima

6. sastanak

- Datum: 31. listopada 2022
- Prisustvovali: D.Kiramarios, K.Tomičić, S.Barac
- Teme sastanka:
 - podjela posla na izradu routinga, izgleda, određenih komponenti i spajanja s backendom

7. sastanak

- Datum: 3. studenoga 2022
- Prisustvovali: D.Kiramarios, D.Kovačević, K.Tomičić, D.Jambrović, K.Orešković, K.Kijac, S.Barac
- Teme sastanka:
 - pregled napravljene dokumentacije s asistenticom
 - savjeti oko potrebnih promjena dokumentacije
 - razgovor o implementaciji konkretnih svari na frontendu i backendu

8. sastanak

- Datum: 5. studenoga 2022
- Prisustvovali: K.Kijac, D.Jambrović
- Teme sastanka:
 - pregled prve verzije dokumentacije
 - podjela posla oko finalnih promjena prve verzije dokumentacije

9. sastanak

- Datum: 8. studenoga 2022
- Prisustvovali: D.Kiramarios, D.Kovačević, K.Orešković, S.Barac
- Teme sastanka:
 - spajanje frontenda i backenda (integracija API poziva)

10. sastanak

- Datum: 9. studenoga 2022
- Prisustvovali: D.Kiramarios, K.Tomičić, S.Barac
- Teme sastanka:
 - daljnji planovi oko podjele posla na frontendu
 - priprema za deploy prve inačice

11. sastanak

- Datum: 10. studenoga 2022
- Prisustvovali: K.Orešković, D.Jambrović
- Teme sastanka:
 - pregled i diskusija dijagrama razreda

12. sastanak

- Datum: 10. studenoga 2022
- Prisustvovali: D.Kiramarios, K. Kijac
- Teme sastanka:
 - provjera sekveničkih dijagrama

13. sastanak

- Datum: 5. prosinca 2022
- Prisustvovali: D.Kiramarios, D.Kovačević, K.Tomičić, D.Jambrović, K.Orešković, K.Kijac, S.Barac
- Teme sastanka:
 - revizija do sad odrađenog posla

14. sastanak

- Datum: 6. prosinca 2022
- Prisustvovali: D.Kovačević, K.Orešković
- Teme sastanka:
 - pregled i diskusija funkcionalnosti administratora

15. sastanak

- Datum: 6. prosinca 2022
- Prisustvovali: D.Kiramarios, K.Tomičić, S.Barac
- Teme sastanka:
 - daljnja podjela posla oko rada na frontendu

16. sastanak

- Datum: 10. prosinca 2022
- Prisustvovali: D.Kiramarios, K.Tomičić, D.Kovačević
- Teme sastanka:
 - povezivanje frontenda i backenda
 - rasprava o potrebnim funkcionalnostima

17. sastanak

- Datum: 11. prosinca 2022
- Prisustvovali: D.Kiramarios, K.Tomičić, S.Barac, D.Jambrović
- Teme sastanka:
 - pregled do sad napravljenoga
 - savjeti oko sljedećih koraka razvoja frontenda
 - daljnja podjela posla oko rada na frontendu

18. sastanak

- Datum: 15. prosinca 2022
- Prisustvovali: D.Kiramarios, K.Tomičić, S.Barac, D.Jambrović
- Teme sastanka:
 - pregled do sad napravljenoga
 - savjeti oko sljedećih koraka razvoja frontenda
 - daljnja podjela posla oko rada na frontendu

19. sastanak

- Datum: 3. siječnja 2023
- Prisustvovali: D.Jambrović, K.Kijac
- Teme sastanka:
 - podjela posla oko druge revizije dokumentacije

20. sastanak

- Datum: 5. siječnja 2023
- Prisustvovali: D.Kiramarios, D.Kovačević, K.Tomičić, D.Jambrović, K.Orešković, K.Kijac, S.Barac
- Teme sastanka:
 - pregled do sad napravljenoga
 - podjela posla oko rada na frontendu i backendu
 - dogovor vremena testiranja

21. sastanak

- Datum: 11. siječnja 2023
- Prisustvovali: D.Kiramarios, D.Kovačević, K.Tomičić, D.Jambrović, K.Orešković, K.Kijac, S.Barac
- Teme sastanka:
 - pregled implementacije aplikacije
 - provođenje ručnog testiranja aplikacije

Tablica aktivnosti

Napomena: Doprinose u aktivnostima treba navesti u satima po članovima grupe po aktivnosti.

	Dario Kiramarios	David Kovačević	Krunoslav Tomičić	Dominik Jambrović	Krešo Orešković	Karla Kijac	Sven Barac
Upravljanje projektom	11	8	10	10	8	10	10
Koordinacija tima	22	21	21	22	21	21	21
Opis projektnog zadatka				2		9	
Funkcionalni zahtjevi				4		2	
Opis pojedinih obrazaca				15			
Dijagram obrazaca				10		6	
Sekvencijski dijagrami	1					15	
Opis ostalih zahtjeva				2			
Arhitektura i dizajn sustava				5			
Baza podataka				4	8		
Dijagram razreda		2		6	8		
Dijagram stanja				12			
Dijagram aktivnosti						7	
Dijagram komponenti						8	
Korištene tehnologije i alati	6	6	6	6	10	6	6
Ispitivanje programskog rješenja		30	3	10	25	6	4
Dijagram razmještaja				5			

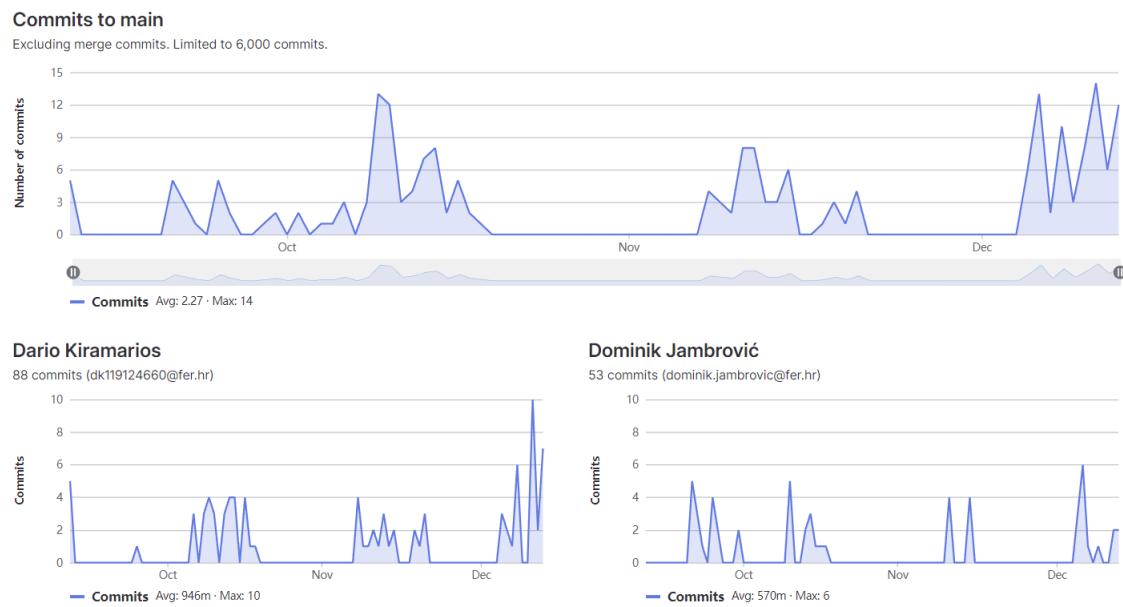
Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

	Dario Kiramarios	David Kovačević	Krunoslav Tomičić	Dominik Jambrović	Krešo Orešković	Karla Kijac	Sven Barac
Upute za puštanje u pogon	7		6	7			
Dnevnik sastajanja			7		1		
Zaključak i budući rad					7		
Popis literature			4		1		
<i>Front end</i>	80		77	7		8	73
<i>Izrada baze podataka</i>		8			11		
<i>Spajanje s bazom podataka</i>	2	8	2		7		2
<i>Back end</i>		44			23		
<i>Deploy</i>	11	10	1		2		4
<i>Popravak dokumentacije</i>				18	5	14	

Dijagrami pregleda promjena

Slike 6.1 i 6.2 prikazuju statistiku commitova preuzetu sa stranice Gitlab.



Slika 6.1: Prikaz commitova, prvi dio



Slika 6.2: Prikaz commitova, drugi dio