

Programsko inženjerstvo

Ak. god. 2020./2021.

Stop waste

Dokumentacija, Rev. 2

Grupa: *IvicaMarica*

Voditelj: *Katarina Mišura*

Datum predaje: *14. 01. 2021.*

Nastavnik: *Ana Maria Jakoubek, mag. educ. math.*

Sadržaj

1 Dnevnik promjena dokumentacije	3
2 Opis projektnog zadatka	4
3 Specifikacija programske potpore	8
3.1 Funkcionalni zahtjevi	8
3.1.1 Obrasci uporabe	10
3.1.2 Sekvencijski dijagrami	19
3.2 Ostali zahtjevi	21
4 Arhitektura i dizajn sustava	22
4.1 Baza podataka	23
4.1.1 Opis tablica	24
4.1.2 Dijagram baze podataka	28
4.2 Dijagram razreda	30
4.3 Dijagram stanja	34
4.4 Dijagram aktivnosti	35
4.5 Dijagram komponenti	36
5 Implementacija i korisničko sučelje	37
5.1 Korištene tehnologije i alati	37
5.2 Ispitivanje programskog rješenja	40
5.2.1 Ispitivanje komponenti	40
5.2.2 Ispitivanje sustava	44
5.3 Dijagram razmještaja	50
5.4 Upute za puštanje u pogon	51
6 Zaključak i budući rad	53
Popis literature	54
Dodatak: Prikaz aktivnosti grupe	56

Indeks slika i dijagrama

56

1. Dnevnik promjena dokumentacije

Rev.	Opis promjene/dodatka	Autori	Datum
0.1	Napravljen predložak.	Fićković	06.11.2020.
0.2	Započeo pisati specifikacije	Aradski	6.11.2020.
0.3	Arhitektura,dodatak - dnevnik sastajanja i tablica aktivnosti.	Mišura	08.11.2020.
0.4	Nastavio pisati korisničke slučajeve.	Aradski	8.11.2020.
0.5	Dodano potpoglavlje baza podataka	Fićković	09.11.2020.
0.6	Funkcionalni zahtjevi	Rakocija	10.11.2020.
0.8	Napisao ostale zahtjeve za specifikaciju	Aradski	11.11.2020.
0.9	Opis projektnog zadatka	Lukačević	11.11.2020.
0.10	Dijagrami razreda	Brala	11.11.2020.
0.11	Dijagrami obrazca i sekvencijski dijagrami	Kolarec	12.11.2020.
1.0	Verzija samo s bitnim dijelovima za 1. ciklus	Mišura	12.11.2020.
1.1	Dodan dijagram stanja te upute za puštanje u pogon	Fićković	11.01.2021.
1.2	Dodani dijagrami razreda	Fićković	11.01.2021.
1.3	Dodan dijagram aktivnosti, dnevnik sastajanja i korištene tehnologije i alati	Mišura	12.01.2021.
1.4	Dodan dijagram komponenti	Brala	13.01.2021.
1.5	Dodano ispitivanje programske potpore	Lukačević	13.01.2021.
1.6	Dodan dijagram razmještaja i zaključak	Rakocija	14.01.2021.
1.7	Izmjena dijagrama aktivnosti, korištene tehnologije i alati i dnevnika aktivnosti	Mišura	14.01.2021.

2. Opis projektnog zadatka

Cilj ovog projekta je razviti programsku podršku za stvaranje web aplikacije „Stop waste“ koja će pridonijeti smanjivanju otpada od hrane. Sve se više podiže svijest o zaštiti okoliša na Zemlji te se postavlja pitanje o problemu otpada hrane. Otprilike se jedna trećina ukupne količine hrane proizvedene u svijetu baci. Kako bismo to spriječili ovaj projekt će omogućiti izgradnju aplikacije da hrana ne završi u otpadu, a istovremeno da je kupimo po nižoj cijeni. Ponuđači sa viškom hrane kao što su supermarketi, OPG-ovi i restorani objavljuju oglase s hranom koju prodaju ili doniraju.

Potencijalni korisnik aplikacije može biti bilo tko. Cilj aplikacije je povezati proizvođače hrane čiji proizvodi ne zadovoljavaju određene standarde, restorane s viškom pripremljene hrane, supermarkete s robom blizu isteka roka valjanosti ili robom oštećenom u prijevozu s ljudima koji su ekološki osviješteni i ne žele dopustiti propadanje još uvijek jestive hrane. Aplikacija bi također bila usmjerena prema dobrotvornim udrugama koje bi donirale hranu i potpomogle smanjenju problema gladi u svijetu.

Prilikom pokretanja sustava prikazuju se oglasi na početnoj stranici. Svaki oglas sadrži naziv, fotografiju, cijenu, opis, popust te vremenski period do isteka oglasa. Uz svaku objavu vežu se podaci o lokaciji koji se utvrđuju automatski iz lokacije ili preglednika, ali se može zadati i pomoću karte. Na početnoj stranici prvo su prikazani najbliži i najstariji aktivni oglasi, tj. oglasi koji će prvi isteći. Oglase je moguće pretraživati po nazivu ili filtrirati po određenim kriterijima.

Neregistrirani korisnici mogu koristiti aplikaciju samo za pretraživanje oglasa i pregledavanje statistike prodanih oglasa i količine hrane koja nije završila u otpadu, ali nemaju mogućnost kupnje proizvoda, tj. kontaktiranja ponuđača.

Za kreiranje novog računa potrebi su sljedeći podaci:

- korisničko ime
- lozinka
- email adresa
- lokacija
- kategorija

- preferirani popust
- opcija kupac

Registrirani korisnici bi ulaskom na svoj profil imali pregled osnovnih podataka koje mogu izmijeniti, pregled poruka koje su izmjenjivali sa ponuđačem te pregled kupljenih ili objavljenih oglasa.

Odabirom željenog oglasa otvara se nova stranica koja opisuje odabrani oglas kao na početnoj stranici, ali i daje mogućnost rezervacije i kontaktiranja sa ponuđačem. Kupac koji rezervira oglas dobiva povratnu informaciju o trajanju rezervacije, tj. vremenskom periodu u kojem mora obaviti kupovinu i preuzeti hranu. Putem privatnih poruka kupac i ponuđač dogovore plaćanje i preuzimanje hrane. Prodani oglas se automatski prikazuje kao dio statistike s podacima o količini i iznosu prodane hrane, a uklanja se iz tražilice. U slučaju da kupac nije preuzeo hranu u vremenskom periodu, rezervacija se ukida i oglas se vraća na prikaz aktivnih oglasa.

Korisnici aplikacije su :

- kupac
- ponuđač
- administrator

Kupac prilikom registracije odabire opciju kupac. Oni mogu samo pretraživati i kupovati proizvode, ali nemaju mogućnost objavljivanja proizvoda. Ulaskom u odjeljak „Moji oglasi“ imaju pregled nad oglasima koji su rezervirani odnosno kupljeni. Kupci mogu postaviti i izmjenjivati omiljene kategorije proizvoda za koje su zainteresirani, cjenovni razred proizvoda te lokaciju u čijoj blizini želi da mu se prikazuju aktivni oglasi. Postojala bi i mogućnost pretplate na obavijesti koje bi stizale kupcima svaki puta kada bi se postavio oglas koji zadovoljava neke od postavljenih kriterija ili pretplata na proizvode nekog konkretnog prodavača, registriranog korisnika aplikacije. Prilikom registracije kupac unosi svoju adresu, na primjer kućnu adresu, ali prilikom korištenja aplikacije uvijek može odabrati opciju prikaza aktivnih oglasa u blizini njegove trenutne lokacije te tako na primjer kada završava sa poslom može na brz i jednostavan način kupiti gotov obrok po nižoj cijeni na putu prema kući.

Ponuđač kod registracije odabire opciju ponuđač. Ponuđači predstavljaju trgovine, supermarkete, OPG-ove, restorane itd. Imaju mogućnost objavljivanja oglasa

s hranom koju prodaju ili doniraju, također mogu kupiti hranu koji su drugi objavili. Ulaskom u odjeljak „Moji oglasi“ imaju pregled nad oglasima koji su predani, rezervirani, prodani i kupljeni. Prodani oglasi sadrže statistiku s podacima o količini i iznosu prodane hrane. Ponuđač na svojoj početnoj stranici ima opciju „Predaj oglas“ koja mu omogućuje da objavi novi oglas sa hranom. Klikom na „Predaj oglas“ otvara se nova stranica na kojoj treba popuniti podatke o novom oglasu.

Za kreiranje novog oglasa potrebi su sljedeći podaci:

- naslov
- kratki opis
- lokacija
- cijena
- popust
- rok
- fotografija
- kontakt telefon
- e-mail adresa

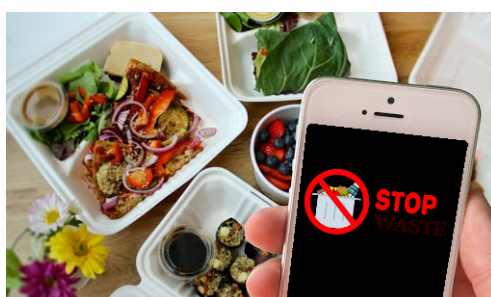
Naslov oglasa bi bio ime pripremljenog jela, pekarskog proizvoda, voća, povrća ili nekog drugog proizvoda koji je prodaje. U kratkom opisu oglasa, uz poruku prodavača potencijalnim kupcima, bile bi ponuđene i oznake kategorija kao što su na primjer „gotov obrok“, „svježe voće“, „pekarski proizvod“, „mliječni proizvodi“ i slično. Prodavač bi također bio obvezan upisati stvarnu cijenu proizvoda i sniženu cijenu proizvoda te bi se na osnovu toga automatski izračunavao popust. Svaki oglas ima svoj rok aktivnosti do čijeg isteka proizvodi moraju biti kupljeni i po mogućnosti preuzeti.

Administrator vodi računa o tome kakav je objavljeni sadržaj. Administratori imaju ovlasti brisanja sadržaja, privremenog ili trajnog blokiranja korisnika.

Neki od mogućih primjer korištenja aplikacije bi bili:

- restoran jedan sat pred zatvaranje objavljuje oglas o pripremljenom jelu koje nitko nije naručio, fotografira jelo i uz opis pripremljenog jela ističe popust po kojem ga prodaje. Kupci taj oglas mogu vidjeti na početnoj stranici ili ako su pretplaćeni na oglase njihovog omiljenog restorana ili tu vrstu jela. Kupac rezervira navedeni oglas te kontaktira prodavača za dogovor oko detalja preuzimanja obroka i o načinu plaćanja

- lokalna dobrotvorna organizacija objavljuje oglas kako će na navedeni dan u svojim prostorijama provoditi donacije hrane. U tom slučaju popust iznosi 100%, a ponuđač uklanja aktivan oglas nakon što se sva hrana donira
- obiteljsko poljoprivredno gospodarstvo s uzgojem jabuka je pogodila vremenska nepogoda te plod jabuke, iako i dalje vrlo ukusan, više ne zadovoljava estetske standarde po kojima veliki trgovački lanci otkupljuju njihove proizvode. Kako urod jabuka ne bi u potpunosti propao objavljuju oglas u „Stop waste“ aplikaciji i pronalaze kupce kojima je pri kupovini zdrave hrane bitnije poznato porijeklo hrane nego njen savršeni oblik



Slika 2.1: Ilustracija rada aplikacije, izvor[5]

Aplikacija „Stop waste“ je specifična upravo po njenoj brzini i jednostavnosti procesa prodaje i kupnje proizvoda. Ponudom prikaza aktivnih oglasa u blizini vlastite lokacije i mogućnošću rezervacije oglasa bitno se smanjuje vrijeme koje je potrebno za kupovinu i preuzimanje željenog proizvoda ili obroka.

„Stop waste“ bi smanjila količinu proizvedene hrane koja se baca i pridonijela smanjenju emisije stakleničkih plinova koji nastaju od hrane koja propada brinući tako o očuvanju okoliša, a ujedno i omogućila onima s manjim budžetom veću kupovnu moć i približila dobrotvorna organizacije onima u potrebi.

3. Specifikacija programske potpore

3.1 Funkcionalni zahtjevi

Dionici:

1. Ponuđač
2. Kupac
3. Administrator

Aktori i njihovi funkcionalni zahtjevi:

1. Neregistrirani/neprijavljeni korisnik (inicijator) može:

- (a) pregledavati i pretraživati oglase
- (b) pregledavati statistike oglasa

2. Registrirani kupac (inicijator) može:

- (a) pregledavati i pretraživati oglase
- (b) kontaktirati ponuđača vezano za oglas
- (c) rezervirati oglas
- (d) otkazati rezervaciju oglasa
- (e) pregledavati i uređivati osnovne podatke
- (f) pregledavati i slati poruke
- (g) pregledavati kupljene oglase

3. Registrirani ponuđač (inicijator) može:

- (a) pregledavati i pretraživati oglase
- (b) kontaktirati ponuđača vezano za oglas
- (c) rezervirati oglas
- (d) otkazati rezervaciju oglasa
- (e) pregledavati i uređivati osnovne podatke
- (f) pregledavati i slati poruke
- (g) pregledavati kupljene oglase

- (h) objavljivati oglase
- (i) brisati i uređivati vlastite oglase
- (j) potvrditi preuzimanje/prodaju

4. Administrator (inicijator) može:

- (a) pregledavati i pretraživati oglase
- (b) kontaktirati ponuđača vezano za oglas
- (c) rezervirati oglas
- (d) otkazati rezervaciju oglasa
- (e) pregledavati i uređivati osnovne podatke
- (f) pregledavati i slati poruke svim korisnicima
- (g) pregledavati vlastite kupljene oglase
- (h) vidjeti popis svih registriranih korisnika i njihovih podataka
- (i) brisati oglase
- (j) privremeno ili trajno blokirati ostale korisnike

5. Baza podataka (sudionik):

- (a) pohranjuje sve podatke o korisnicima i njihovim ovlastima
- (b) pohranjuje sve podatke o oglasima

3.1.1 Obrasci uporabe

Opis obrazaca uporabe

UC1 - Pregled oglasa

- **Glavni sudionik:** korisnik, kupac
- **Cilj:** Pregledati oglase
- **Sudionici:** Baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
 1. Karta je prikazana prilikom učitavanja aplikacije
 2. Korisnik dopušta pristup lokaciji i potvrđuje njezinu ispravnost
 3. Prikazuju se oglasi u blizini korisnika
 4. Odabirom oglasa se prikazuju informacije o oglasu
- **Opis mogućih odstupanja:**
 - 2.a Korisnik ne dopušta pristup lokaciji
 1. Korisniku ponuditi mogućnost ručnog unosa lokacije
 2. Korisniku omogućiti pregledavanje oglasa bez unosa lokacije
 - 2.b Automatski određena lokacija je neprecizna
 1. Korisniku ponuditi mogućnost ručnog unosa lokacije

UC2 - Registracija

- **Glavni sudionik:** Korisnik
- **Cilj:** Stvoriti korisnički račun za pristup sustavu
- **Sudionici:** Baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
 1. Korisnik odabire opciju za registraciju
 2. Korisnik unosi korisničke podatke
 3. Korisnik prima obavijest o uspješnoj registraciji
- **Opis mogućih odstupanja:**
 - 2.a Odabir već zauzetog korisničkog imena i/ili e-maila, unos korisničkog podatka u nedozvoljenom formatu ili pružanje neispravnog e-maila
 1. Sustav obavještava korisnika o neuspjehom upisu i vraća ga na stranicu za registraciju
 2. Korisnik mijenja potrebne podatke te završava unos ili odustaje od registracije

UC3 - Prijava u sustav

- **Glavni sudionik:** Kupac
- **Cilj:** Dobiti pristup dodtanim mogućnostima registriranog kupca
- **Sudionici:** Baza podataka
- **Preduvjet:** Registracija
- **Opis osnovnog tijeka:**
 1. Unos korisničkog imena i lozinke
 2. Potvrda o ispravnosti unesenih podataka
 3. Pristup korisničkim funkcijama
- **Opis mogućih odstupanja:**
 - 2.a Neispravno korisničko ime/lozinka
 1. Sustav obavještava korisnika o neuspjelom upisu, savjetuje da provjeri podatke i ponovno pokuša ili ako nema račun ga navodi na registraciju

UC4 - Pregled osobnih podataka

- **Glavni sudionik:** Kupac/ponuđač
- **Cilj:** Pregledati osobne podatke
- **Sudionici:** Baza podataka
- **Preduvjet:** Prijava
- **Opis osnovnog tijeka:**
 1. Korisnik odabire opciju "Osobni podatci"
 2. Aplikacija prkazuje osobne podatke korisnika

UC5 - Promjena osobnih podataka

- **Glavni sudionik:** Kupac
- **Cilj:** Promijeniti osobne podatke
- **Sudionici:** Baza podataka
- **Preduvjet:** Prijava
- **Opis osnovnog tijeka:**
 1. Korisnik odabire opciju uređivanja određenog osobnog podatka
 2. Korisnik mijenja osobni podatak
 3. Korisnik sprema promjene
 4. Baza podataka se ažurira
 5. Korisnik prima potvrdu promjene osobnih podataka

- **Opis mogućih odstupanja:**

- 2.a Korisnik promijeni svoje osobne podatke, ali ne odabere opciju "Spremi promjenu"

- 1. Sustav obavještava korisnika da nije spremio podatke prije izlaska iz prozora

- 3.a Odabir već zauzetog korisničkog imena i/ili e-maila, unos korisničkog podatka u nedozvoljenom formatu ili pružanje neispravnoga e-maila

- 1. Sustav obavještava korisnika o neuspjelom spremanju i ističe problematične podatke

UC6 - Rezervacija oglasa

- **Glavni sudionik:** Kupac

- **Cilj:** Rezervirati jedan ili više oglasa

- **Sudionici:** Baza podataka

- **Preduvjet:** Prijava

- **Opis osnovnog tijeka:**

- 1. Korisnik odabire oglas koji želi rezervirati

- 2. Korisnik odabire opciju "Rezerviraj oglas"

- 3. Korisnik dobije povratno u kojem vremenskom periodu mora obaviti kupnju i preuzimanje hrane

- 4. U slučaju da Korisnik nije preuzeo hranu u vremenskom periodu kojem je trebao, rezervacija se ukida.

- **Opis mogućih odstupanja:**

- 2.a Oglas koji je korisnik pokušao rezervirati je već rezerviran

- 1. Sustav obavještava korisnika da je oglas već rezerviran

UC7 - Brisanje rezervacije

- **Glavni sudionik:** Kupac

- **Cilj:** Obrisati rezervaciju

- **Sudionici:** Baza podataka

- **Preduvjet:** Prijava i obavljena barem jedna rezervacija

- **Opis osnovnog tijeka:**

- 1. Klijent odabire opciju "Moje rezervacije"

- 2. Prikaze se lista rezervacija klijenta

- 3. Klijent odabire opciju "Obrisi rezervaciju"

- 4. Rezervacija se briše iz baze podataka

UC8 - Pregled aktivne narudžbe

- **Glavni sudionik:** Ponuđač
- **Cilj:** Pogledati trenutno aktivne narudžbe
- **Sudionici:** Baza podataka
- **Preduvjet:** Narudžba je zaprimljena i plaćena
- **Opis osnovnog tijeka:**
 1. Zaposlenik odabere opciju "Aktivne narudžbe"
 2. Prikaze se lista trenutno aktivnih narudžbi

UC9 - Označavanje oglasa gotovim

- **Glavni sudionik:** Ponuđač
- **Cilj:** Oznaciti narudžbu kao gotovu
- **Sudionici:** Baza podataka, klijent
- **Preduvjet:** Zaposlenik je prijavljen, narudžba je zaprimljena i plaćena
- **Opis osnovnog tijeka:**
 1. Prodani oglas se automatski prikazuje kao dio statistike s podacima o količini i iznosu prodane hrane
 2. Zaposlenik označava odabranu narudžbu gotovom

UC10 - Pregledavanje poruka

- **Glavni sudionik:** Kupac/ponuđač
- **Cilj:** Pregledati poruke
- **Sudionici:** Baza podataka
- **Preduvjet:** Prijava
- **Opis osnovnog tijeka:**
 1. Korisnik odabire opciju "Moje poruke"
 2. Prikazu se sve poruke koje je korisnik izmjenio

UC11 - Slanje poruke

- **Glavni sudionik:** Kupac/ponuđač
- **Cilj:** Dogovor oko preuzimanja i plaćanja
- **Sudionici:** Baza podataka
- **Preduvjet:** Prijava
- **Opis osnovnog tijeka:**
 1. Korisnik odabire opciju "Moje poruke"

2. Prikažu se sve poruke koje je korisnik izmjenio
3. Korisnik odabire opciju "Pošalji novu poruku"
4. Prikaže se prozor sa prostorom za upis poruke i osobe kojoj se poruka isporuča
5. Korisnik odabere opciju "Pošalji"
- **Opis mogućih odstupanja:**
 - 2.a Korisnik napiše poruku, ali ne odabere opciju "Pošalji"
 1. Sustav obavještava korisnika da nije poslao poruku prije izlaska iz prozora

UC12 - Pregled kupljenih ili objavljenih oglasa

- **Glavni sudionik:** Kupac/ponuđač
- **Cilj:** Pregledati kupljene ili objavljene oglase
- **Sudionici:** Baza podataka
- **Preduvjet:** Prijava
- **Opis osnovnog tijeka:**
 1. Korisnik odabire opciju "Povijest mojih oglasa"
 2. Prikažu se svi oglasi koje je korisnik ili objavio ili kupio

UC13 - Objavljivanje oglasa

- **Glavni sudionik:** Ponuđač
- **Cilj:** Objava oglasa
- **Sudionici:** Baza podataka
- **Preduvjet:** Prijava
- **Opis osnovnog tijeka:**
 1. Ponuđač odabere opciju "Dodaj novi oglas"
 2. Ponuđač određuju vremenski period u kojemu se hrana treba kupiti
 3. Ponuđač dodaje naziv, opis i fotografiju hrane, prijašnju cijenu i sadašnji popust
 4. Ponuđač odabere opciju "Objavi"
 5. Promjena se dodaje u bazu podataka
- **Opis mogućih odstupanja:**
 - 2.a Korisnik napiše oglas, ali ne odabere opciju "Objavi"
 1. Sustav obavještava korisnika da nije objavio oglas prije izlaska iz prozora

UC14 - Uređivanje vlastitih oglasa

- **Glavni sudionik:** Ponuđač
- **Cilj:** urediti oglas
- **Sudionici:** Baza podataka
- **Preduvjet:** Prijava
- **Opis osnovnog tijeka:**
 1. Ponuđač odabere opciju "Uredi oglas"
 2. Nakon uređivanja potvrđuje izmjenu
 3. Promjene se upisuju u bazu podataka

UC15 - Brisanje vlastitih oglasa

- **Glavni sudionik:** Ponuđač
- **Cilj:** obrisati oglas
- **Sudionici:** Baza podataka
- **Preduvjet:** Prijava
- **Opis osnovnog tijeka:**
 1. Ponuđač odabere opciju "Obriši oglas"
 2. Potvrđuje izmjenu
 3. Promjene se upisuju u bazu podataka

UC16 - Pregled korisnika

- **Glavni sudionik:** Administrator
- **Cilj:** pregledati registrirane korisnike
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je registriran i dodijeljena su mu prava administratora
- **Opis osnovnog tijeka:**
 1. Administrator odabire opciju pregledavanja korisnika
 2. Prikaze se lista svih ispravno registriranih korisnika s osobnim podacima

UC17 - Brisanje sadržaja

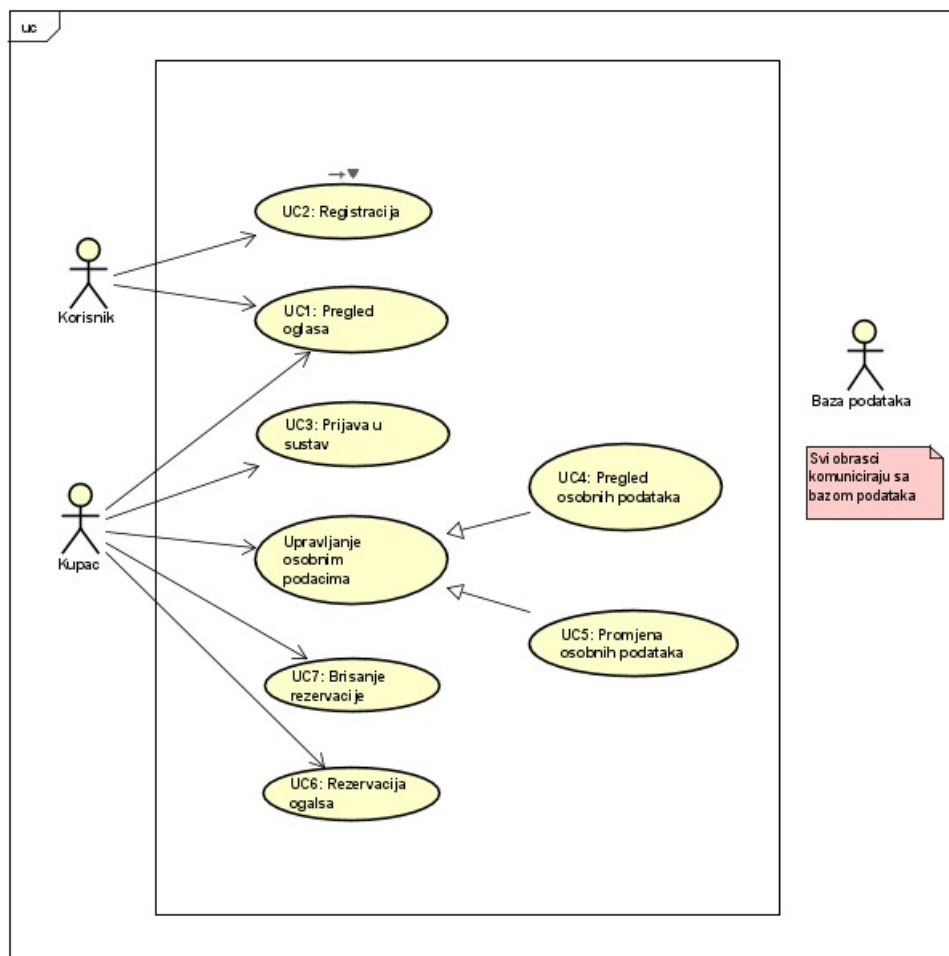
- **Glavni sudionik:** Administrator
- **Cilj:** obrisati sadržaj
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je registriran i dodijeljena su mu prava administratora
- **Opis osnovnog tijeka:**
 1. Administrator odabire sadržaj koji želi obrisati

2. Odabere opciju "Obriši"
3. Promjene se upisuju u bazu podataka

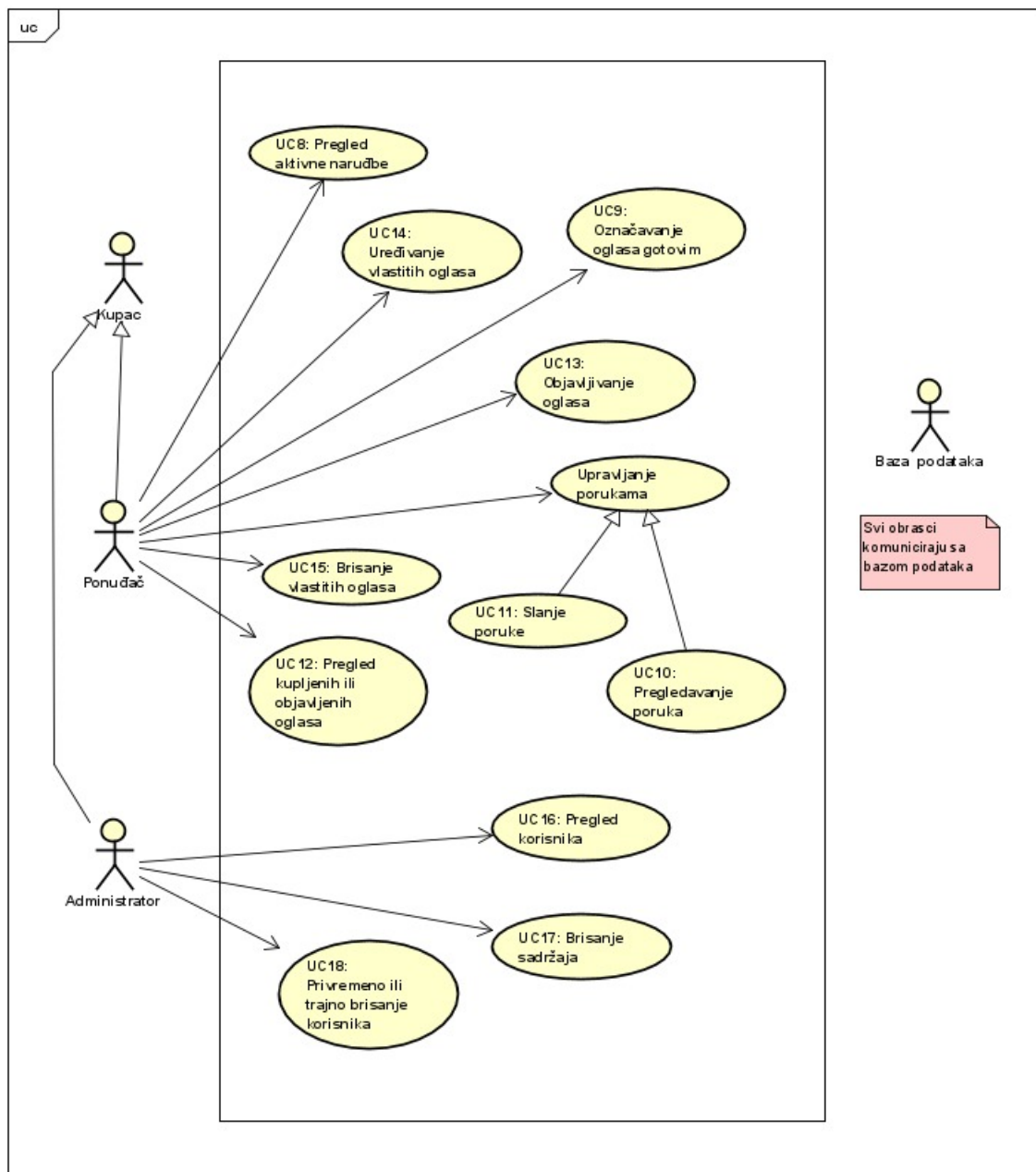
UC18 - Privremeno ili trajno blokiranje korisnika

- **Glavni sudionik:** Administrator
- **Cilj:** blokirati korisnika
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je registriran i dodijeljena su mu prava administratora
- **Opis osnovnog tijeka:**
 1. Administrator odabire opciju blokiranja korisnika
 2. Administrator pronalazi željenog korisnika
 3. Administrator odabere jednu od opcija: "Privremeno blokiraj" ili "Trajno blokiraj" ovisno o namjeni
 4. Ako odabere opciju privremeno blokiraj zadaje vremenski period u kojem želi blokirati korisnika, a ako ga trajno blokira uklanja njega i njegove podatke iz baze podataka

Dijagrami obrazaca uporabe



Slika 3.1: Dijagram obrasca uporabe, funkcionalnosti korisnika i kupca

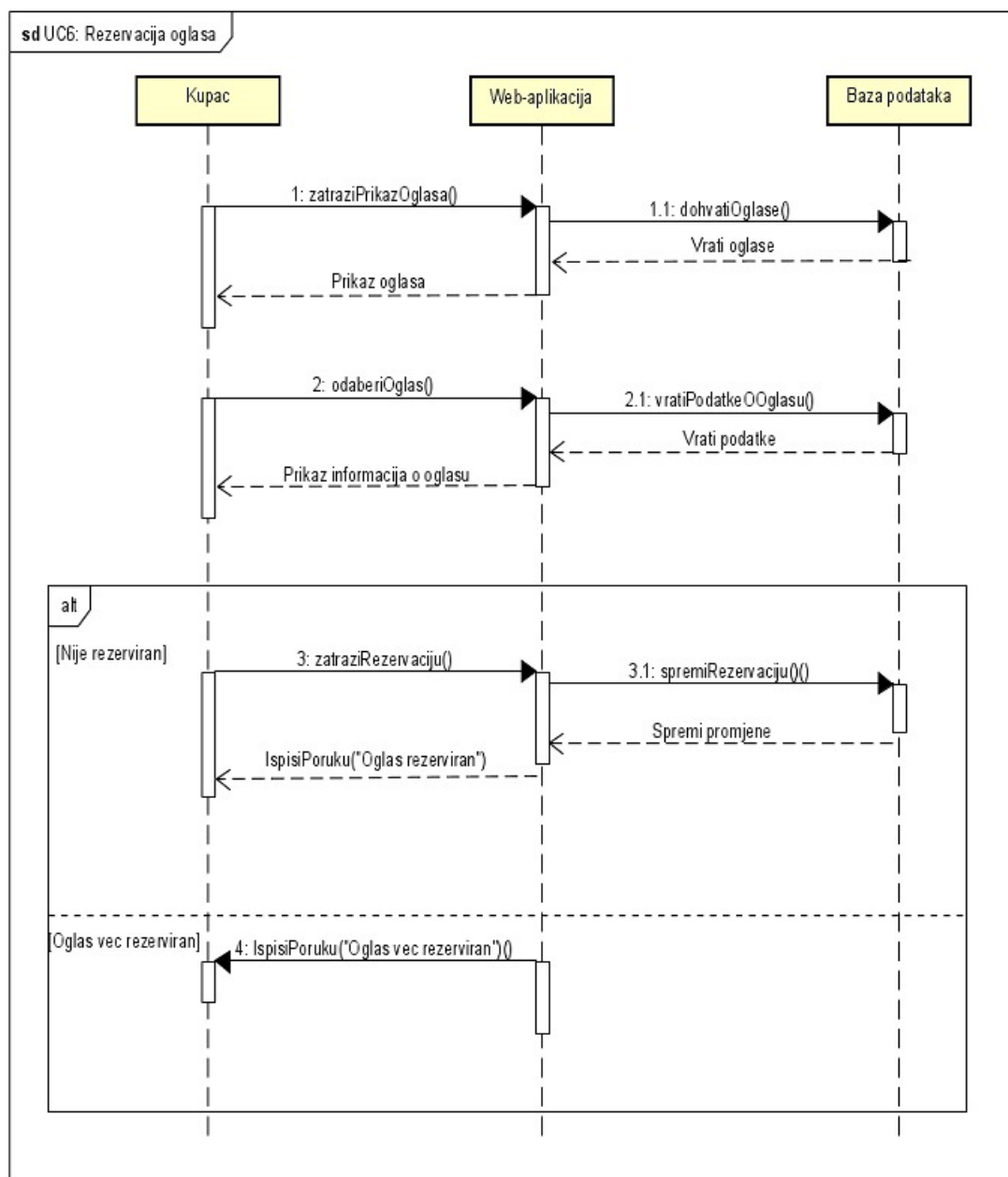


Slika 3.2: Dijagram obrasca uporabe, funkcionalnosti ponuđača i administratora

3.1.2 Sekvencijski dijagrami

UC6 - Rezervacija oglasa

Klijent salje zahtjev za prikaz svih oglasa. Poslužitelj dohvaca obližnje oglase i prikazuje ih. Odabirom oglasa, poslužitelj iz baze podataka dohvaca osnovne podatke o oglasu i prikazuje ih korisniku. Da bi započeo rezervaciju, klijent salje zahtjev za rezervaciju. Poslužitelj u bazi podataka provjerava dostupnost oglasa. Ukoliko je oglas već rezerviran, sustav obavještava klijenta o tome uz poruku. Ako oglas nije rezerviran, poslužitelj informaciju o rezervaciji prosljeđuje bazi koja sprema promjenu.



Slika 3.3: Sekvencijski dijagram za UC6

3.2 Ostali zahtjevi

- Sustav treba omogućiti rad više korisnika u stvarnom vremenu
- Korisničko sučelje i sustav moraju podržavati hrvatsku abecedu (dijakritičke znakove) pri unosu i prikazu tekstualnog sadržaja
- Izvršavanje dijela programa u kojem se pristupa bazi podataka ne smije trajati duže od nekoliko sekundi
- Sustav treba biti implementiran kao web aplikacija koristeći objektno-orijentirane jezike
- Neispravno korištenje korisničkog sučelja ne smije narušiti funkcionalnost i rad sustava
- Sustav treba biti jednostavan za korištenje, korisnici se moraju znati koristiti sučeljem bez opširnih uputa
- Nadogradnja sustava ne smije narušavati postojeće funkcionalnosti sustava
- Sustav kao valutu koristi HRK
- Veza s bazom podataka mora biti kvalitetno zaštićena, brza i otporna na vanjske greške
- Pristup sustavu mora biti omogućen iz javne mreže pomoću HTTPS.

4. Arhitektura i dizajn sustava

Stil višeslojne arhitekture odjeljuje cjelovite komponente i te princip oblikovanja podijeli pa vladaj može preuzeti već postojeću podjelu. Troslojni stil arhitekture je prilagođen za jednostavne web aplikacije i mi ćemo ga koristiti za izradu naše aplikacije.

Arhitektura koju smo izabrali temelji se na principu oblikovanja podijeli pa vladaj. Tim principom smo uspjeli postići da odvojeni manji timovi unutar našeg tima rade na manjim problemima čime smo omogućili specijalizaciju. Takva podjela omogućuje efikasniji rad tima.

Arhitektura se sastoji od tri sloja:

1. podatkovni sloj
2. sloj poslovne logike
3. korisnički ili prezentacijski sloj

Podatkovni sloj je sloj za pohranu podataka u bazu ili datotečni sustav. U našoj aplikaciji koristit će se relacijska baza podataka, a odabrani jezik je SQL te razvojno okruženje PostgreSQL, pgAdmin . Također u počecima izrade aplikacije drugi slojevi će moći koristiti privremenu H2 bazu podataka pisanu u programskom jeziku Java u razvojem okruženju IntelliJIDEA.

Sloj poslovne logike je sloj s implementacijom poslovnih procesa i izračuna. Kod web aplikacija ovaj sloj je podržan od strane web poslužitelja ili aplikacijskog poslužitelja. Ovaj sloj obrađuje i generira dinamički sadržaj koji prosljeđuje sljedećem sloju.

Korisnički ili prezentacijski sloj je sloj s korisničkim sučeljem. Ovaj sloj kod web aplikacija se sastoji od web poslužitelja koji isporučuje statički i dinamički sadržaj, a sadržaj web stranice prikazuje web preglednik. Web poslužitelj to izvršava pomoću mrežnih protokola, a to su HTTP protokoli. Korisnik pomoću ovog sloja šalje upite i naredbe sloju poslovne logike koji ih obrađuje, te također pregledava sadržaj koji mu pruža sloj poslovne logike.

Za razvoj naše aplikacije koristimo radni okvir Java Spring Boot. Pri korištenju tog radnog okvira potrebno je dodati još dodatnih slojeva u arhitekturu pri čemu

se klijenstka i poslužiteljska strana mogu organizirati u više slojeva, ali ne utječu na organizaciju tima po glavna tri sloja.

Višeslojna se arhitektura u tom slučaju dijeli na šest slojeva:

1. sloj korisničke strane
2. sloj nadglednika
3. sloj usluge
4. sloj domene
5. sloj za pristup podacima
6. sloj baze podataka

Sloj korisničke strane implementiran je u JavaScriptu, koristi React koji omogućuje prikaz korisničkog sučelja, a korištena platforma je Microsoft visual studio.

Sloj nadglednika povezuje korisničku stranu s poslužiteljskom stranom. Implementiran je u jeziku Java na platformi IntelliJIDEA.

Sloj usluge obavlja svu poslovnu logiku i potrebne izračune. Implementiran je u jeziku Java na platformi IntelliJIDEA.

Sloj domene ima razrađeni model podataka domene. Implementiran je u jeziku Java na platformi IntelliJIDEA.

Sloj za pristup podacima je sloj koji omogućuje spremanje i dohvat podataka iz određene baze podataka te razmjenu tih podataka sa slojem domene. Implementiran je u jeziku Java na platformi IntelliJIDEA.

Sloj baze podataka omogućuje pohranu podataka u relacijsku bazu ili H2 bazu. Za relacijsku bazu koriste se PostgreSQL i pgAdmin, a za H2 bazu koristi se Java na platformi IntelliJIDEA.

4.1 Baza podataka

Naš sustav zahtijeva pohranu puno podataka, stoga je bilo potrebno izraditi relacijsku bazu podataka koju bi mogli koristiti za te potrebe. Baza nam je potrebna također za izmjenu i dohvat podataka za daljnju obradu te nam omogućava trajno čuvanje podataka. U ovoj ranijoj fazi izgradnje programske potpore za naš sustav odlučili smo se za jednostavniju verziju baze podataka H2 koja pohranjuje podatke u memoriju računala na kojemu je program pokrenut. U kasnijim fazama razvoja potpore ćemo se povezati na trajnu bazu podataka. Baza podataka koja nam je potrebna za naš sustav sastoji se od sljedećih entiteta:

- user
- address
- city
- role
- category
- ad
- condition
- message
- usersPrefersCategory
- adInCategory

4.1.1 Opis tablica

User Ovaj entitet modelira korisnika naše aplikacije. Entitet sadrži sljedeće atribute: *idUser*, *userName*, *email*, *password*, *name*, *surname*, *idAdress* i *idRole*. Atribut *surname* je opcionalan. Ovaj entitet je u vezi Many-to-One s entitetom *Adress* preko entiteta *idAddress* te u vezi Many-to-One s entitetom *Role* preko atributa *idRole*.

User		
idUser	INT	Identifikacijski broj korisnika
<i>userName</i>	VARCHAR	Korisničko ime
<i>email</i>	VARCHAR	Mail adresa korisnika
<i>password</i>	VARCHAR	Lozinka korisnika
<i>name</i>	VARCHAR	Ime korisnika
<i>surname</i>	VARCHAR	Prezime korisnika
<i>idAdress</i>	INT	Identifikacijski broj adrese korisnika
<i>idRole</i>	INT	Identifikacijski broj uloge korisnika

Address Ovaj entitet predstavlja adresu našega korisnika. Sadrži atribute: *idAddress*, *street*, *number*, *latitude*, *longitude* i *postalCode*. Ovaj entitet u vezi je Many-to-One s entitetom *City* preko atributa *postalCode*.

Address		
idAddress	INT	Identifikacijski broj adrese
street	VARCHAR	Ulica lokacije
number	INT	Kućni broj lokacije
latitude	VARCHAR	Geografska širina lokacije
longitude	VARCHAR	Geografska duljina lokacije
<i>postalCode</i>	INT	Poštanski broj mjesta u kojemu se nalazi lokacija

City Ovaj entitet predstavlja grad unutar kojega živi naš korisnik. Sadrži attribute postalCode i cityName.

City		
postalCode	INT	Poštanski broj mjesta u kojemu se nalazi lokacija
cityName	VARCHAR	Naziv grada u kojemu se nalazi lokacija

Role Ovaj entitet predstavlja ulogu koju naš korisnik može imati. Sadrži attribute idRole i roleName.

Role		
idRole	INT	Identifikacijski broj uloge koju korisnik može imati u sustavu
roleName	VARCHAR	Naziv uloge

Category Ovaj entitet predstavlja kategoriju koju naš oglas može imati. Sadrži attribute idCategory te categoryName.

Category		
idCategory	INT	Identifikacijski broj kategorije
categoryName	VARCHAR	Naziv kategorije

Ad Ovaj entitet modelira oglas koji povezuje one koji žele prodati proizvod i one koji žele kupiti proizvod. Entitet sadrži attribute: idAd, caption, image, description, price, discount, timeOfAddition, timeOfExpiration, IdUserSeller, IdUserBuyer i idCondition. Ovaj entitet u vezi je One-To-One s entitetom Condition preko idCondition, u vezi Many-to-One sa entitetom User preko atributa IdUserSeller te

i preko atributa *IdUserBuyer*.

Ad		
idAd	INT	Identifikacijski broj oglasa
caption	VARCHAR	Naslov oglasa
image	LONGBLOB	Slika oglasa
description	VARCHAR	Opis oglasa
price	DECIMAL	Cijena oglasa
discount	INT	Popust koji se ostvaruje
timeOfAddition	DATETIME	Vrijeme postavljanja oglasa
timeOfExpiration	DATETIME	Vrijeme trajanja oglasa
<i>IdUserSeller</i>	INT	Identifikacijski broj korisnika koji je objavio oglas
<i>IdUserBuyer</i>	INT	Identifikacijski broj korisnika koji je kupio preko oglas
<i>idCondition</i>	INT	Identifikacijski broj stanja oglasa

Condition Ovaj entitet predstavlja stanje u kojemu se nalazi oglas. Entitet sadrži attribute: *idCondition* i *conditionName*.

Condition		
idCondition	INT	Identifikacijski broj stanja
conditionName	VARCHAR	Naziv stanja

Message Ovaj entitet modelira poruke koje izmjenjuju kupac i prodavač. Entitet sadrži attribute: *idMessage*, *text*, *time*, *idUserRecieved* i *idUserSent*. Entitet je u vezi Many-to-One sa entitetom User preko atributa *idUserSent* i atributa *idUserRecieved*.

Message		
idMessage	INT	Identifikacijski broj poruke
text	VARCHAR	Tekst poruke
time	DATETIME	Vrijeme poruke
<i>idUserRecieved</i>	INT	Identifikacijski broj korisnika koji je dobio poruku
<i>idUserSent</i>	INT	Identifikacijski broj korisnika koji je poslao poruku

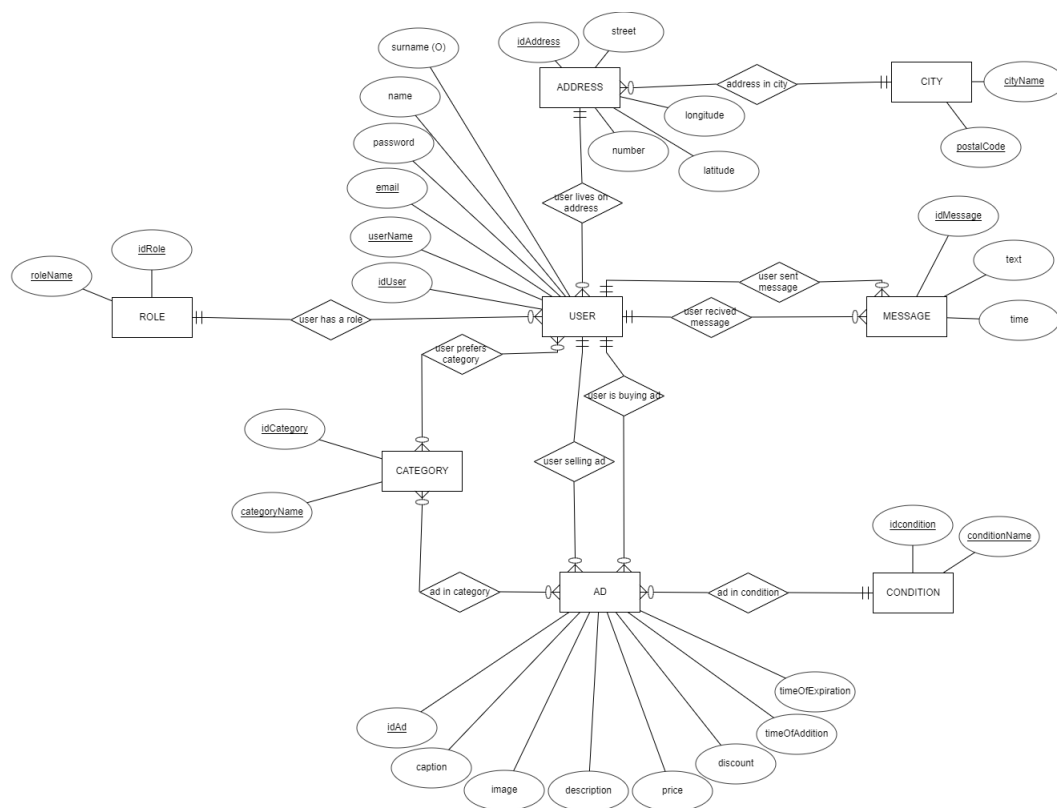
UsersPrefersCategory Ovaj entitet sadrži podatke koji nam govore koje su korisniku najdraže kategorije oglasa. Entitet sadrži attribute: idUser i idCategory. Entitet je u vezi Many-to-Many sa entitetom User preko atributa idUser te je u vezi Many-to-Many sa entitetom Category preko atributa idCategory.

UsersPrefersCategory		
idUser	INT	Identifikacijski broj korisnika
<i>idCategory</i>	INT	Identifikacijski broj kategorija koje korisnik preferira

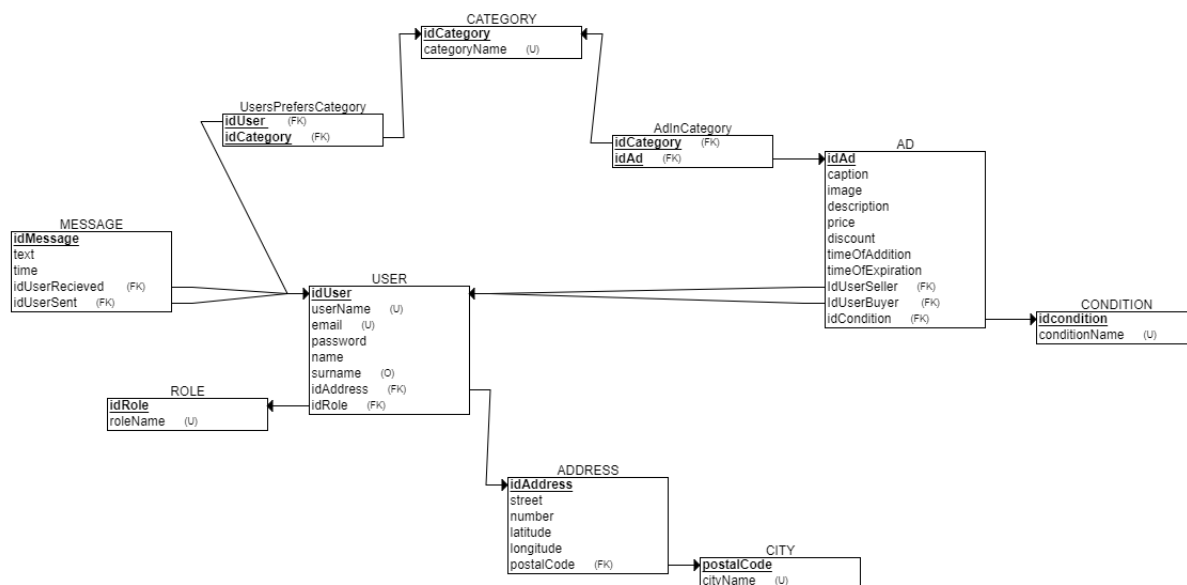
AdInCategory Ovaj entitet sadrži podatke koji nam govore koje sve kategorije mogu biti oglasi. Entitet sadrži attribute: idAd i idCategory. Entitet je u vezi Many-to-Many sa entitetom Ad preko atributa idAd te je u vezi Many-to-Many sa entitetom Category preko atributa idCategory.

AdInCategory		
idAd	INT	Identifikacijski broj oglasa
<i>idCategory</i>	INT	Identifikacijski broj kategorije oglasa

4.1.2 Dijagram baze podataka



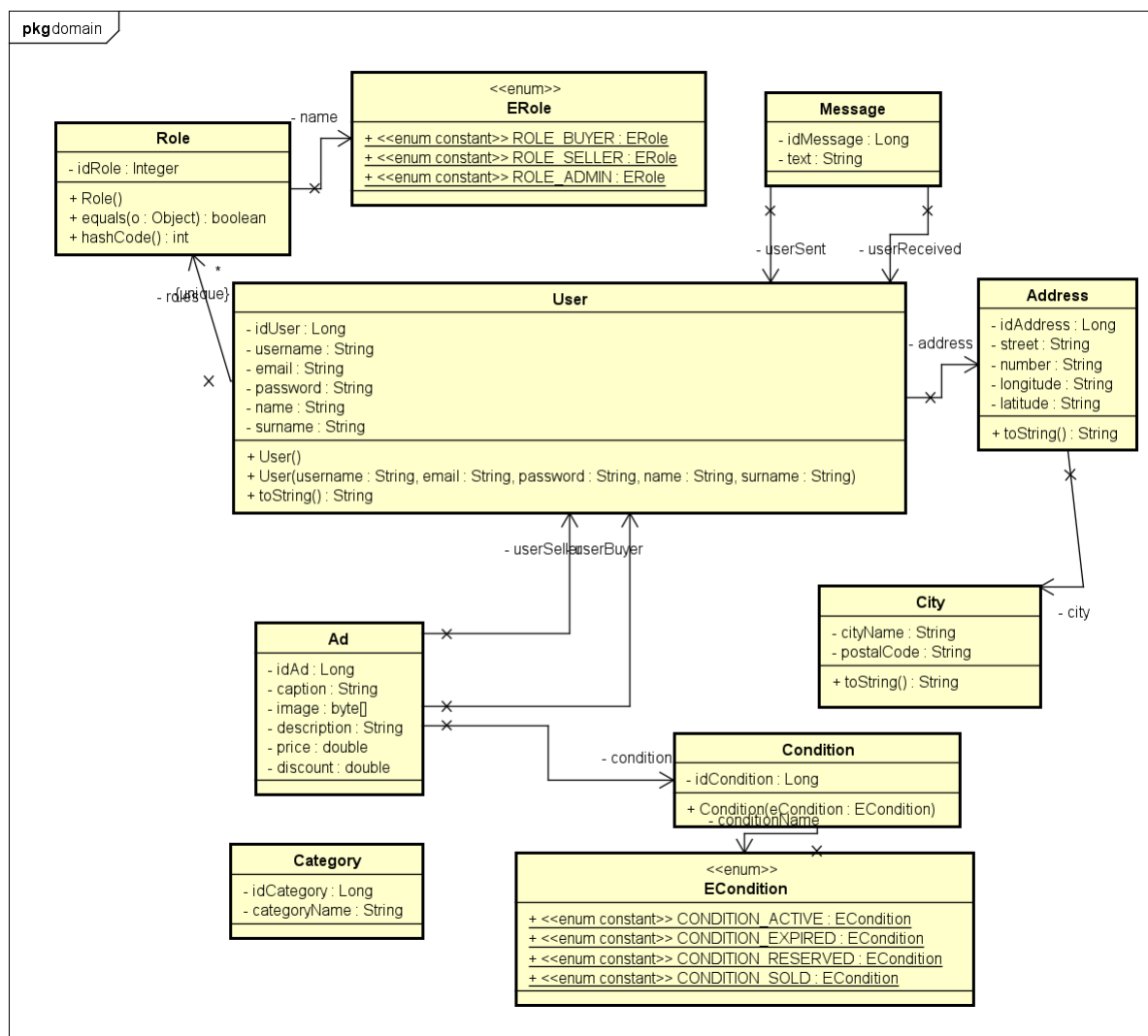
Slika 4.1: E-R dijagram baze podataka



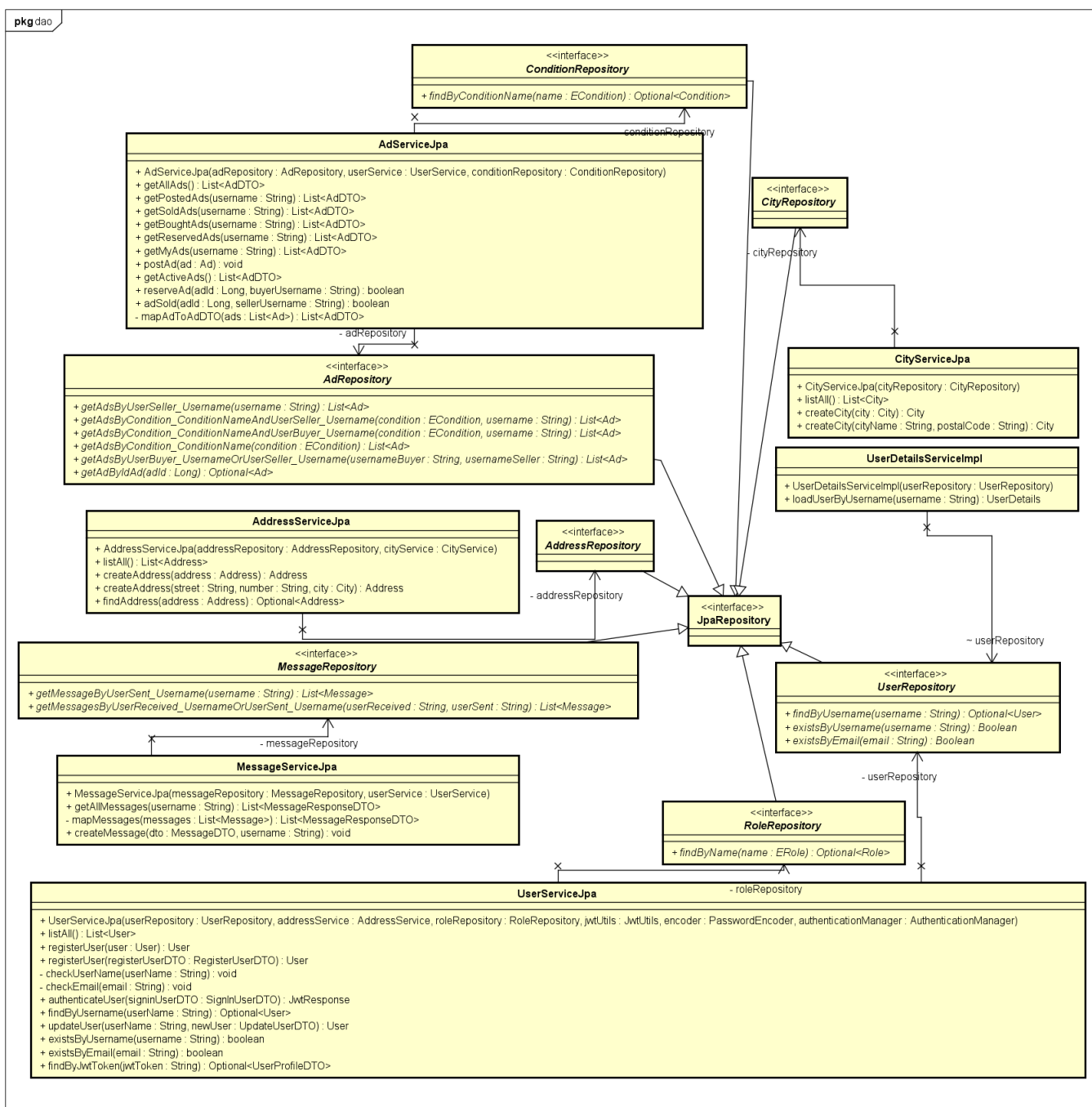
Slika 4.2: Relacijski dijagram baze podataka

4.2 Dijagram razreda

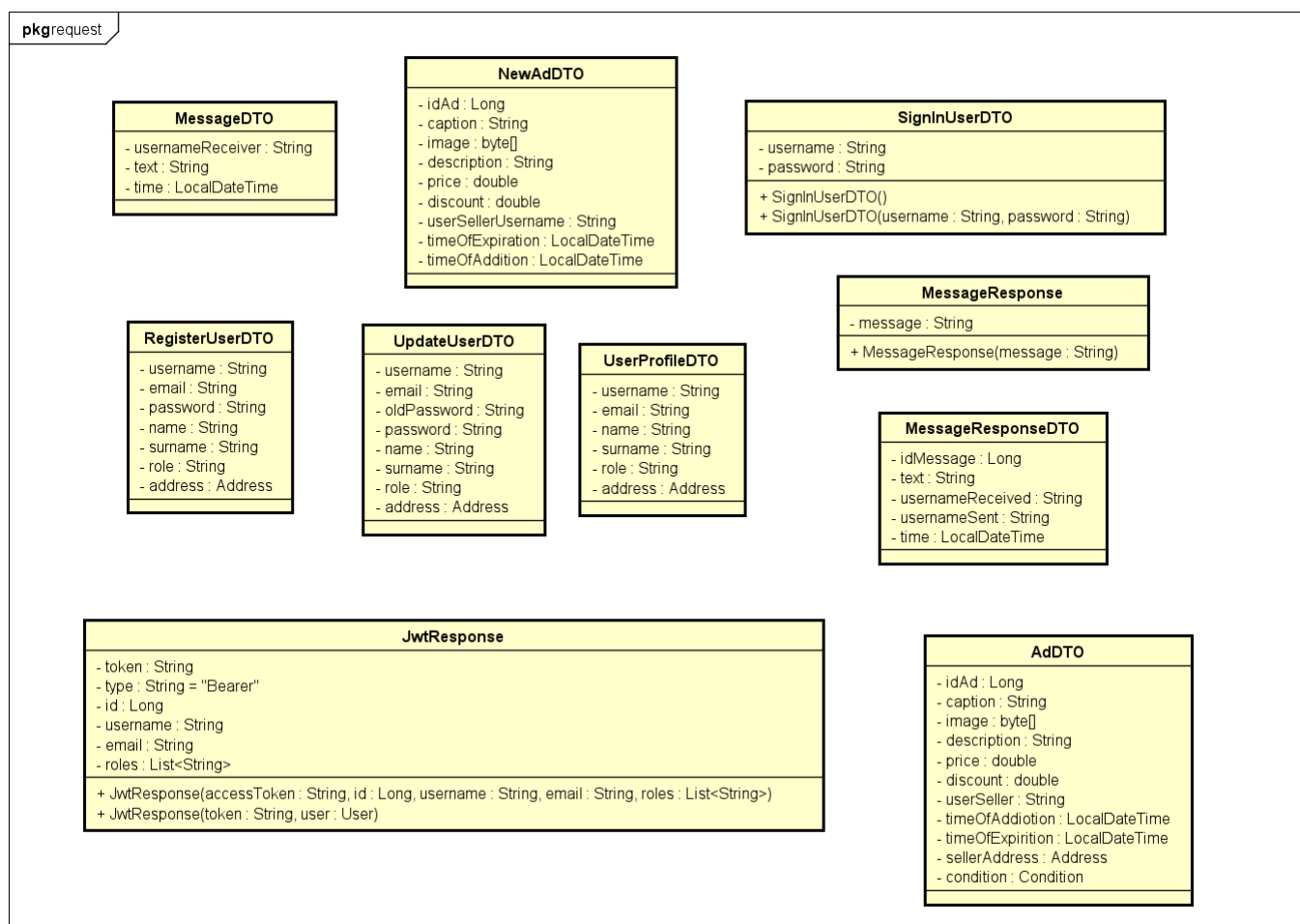
Na slikama su prikazani razredi koji pripadaju backend dijelu arhitekture. Sve su slike izgenerirane pomoću alata Astah za programski jezik Java. Dijagrami su grupini po paketu unutar kojega se nalaze u projektu.



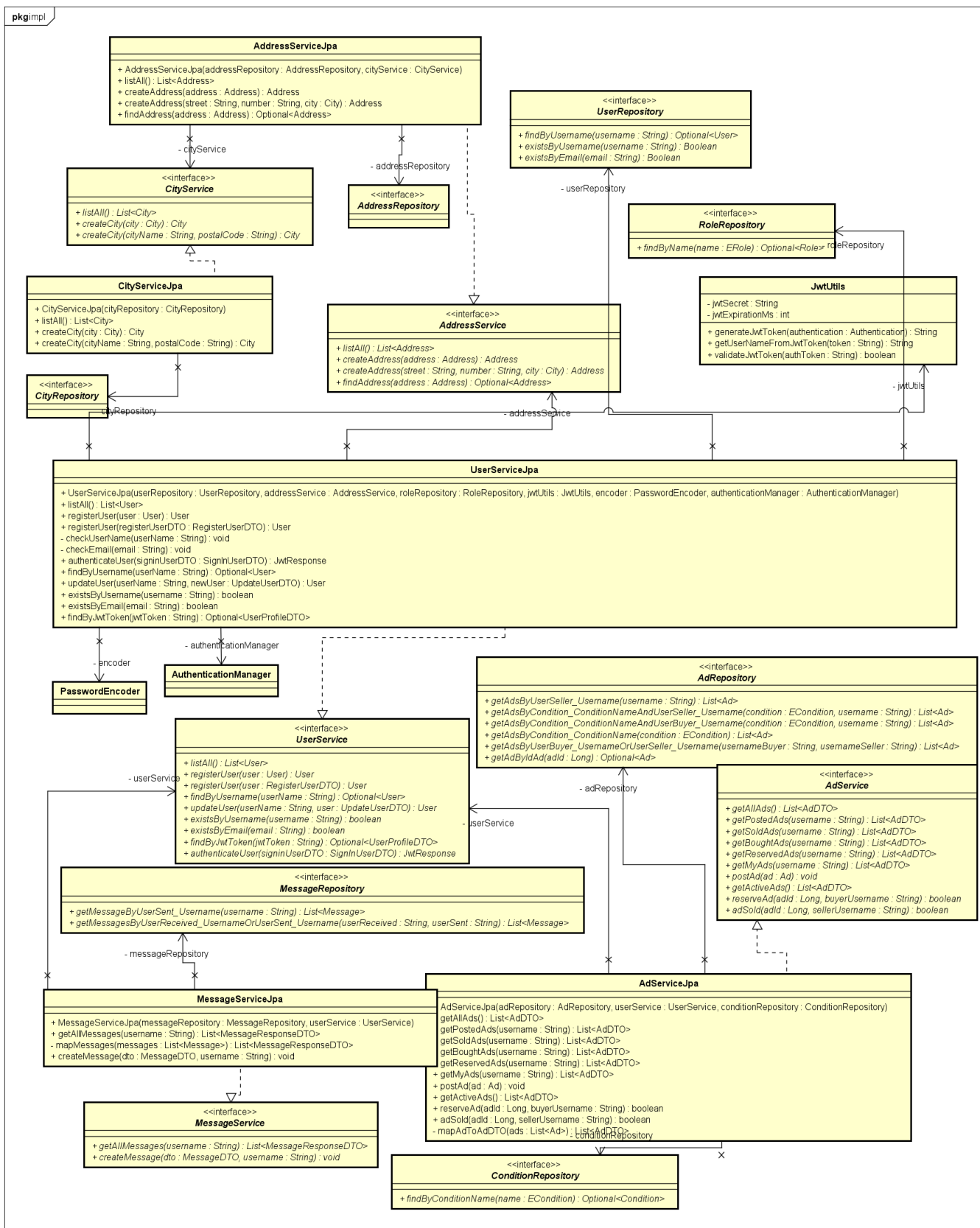
Slika 4.3: Dijagram razreda - domain



Slika 4.4: Dijagram razreda - dao



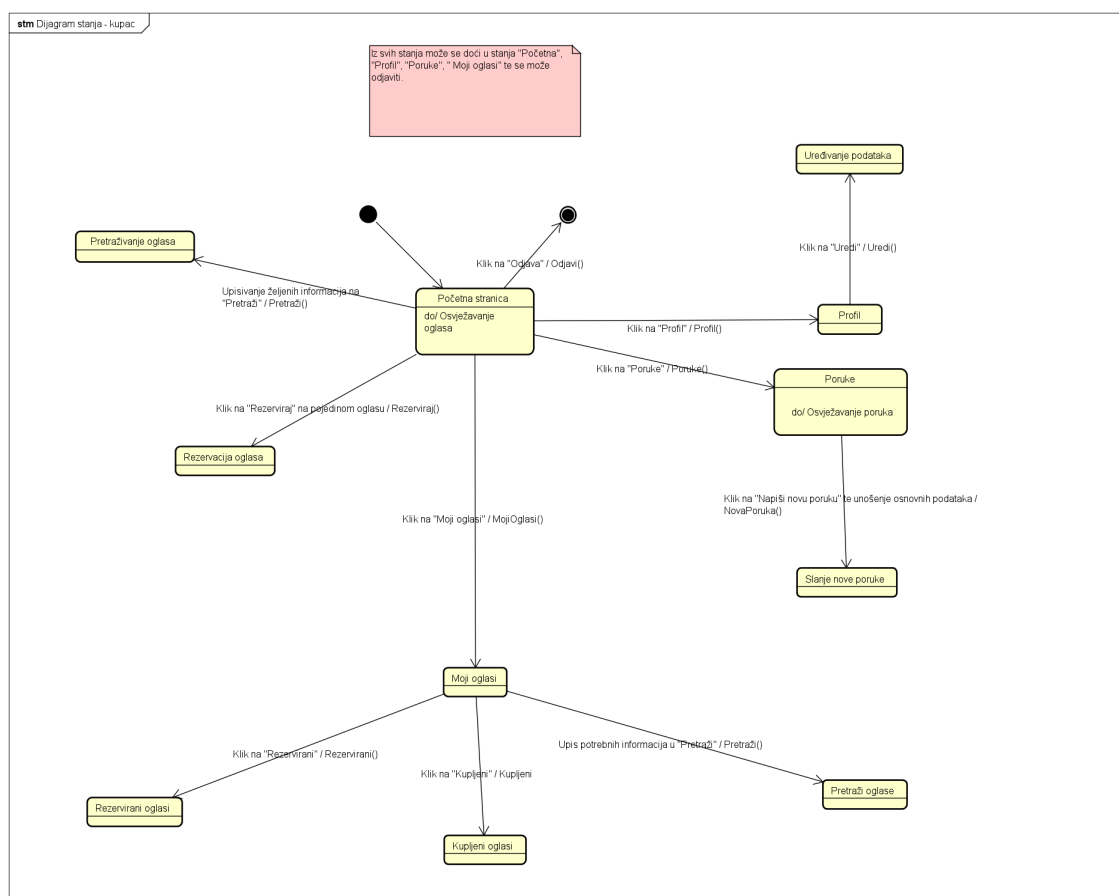
Slika 4.5: Dijagram razreda - Dto



Slika 4.6: Dijagram razreda - Servisi

4.3 Dijagram stanja

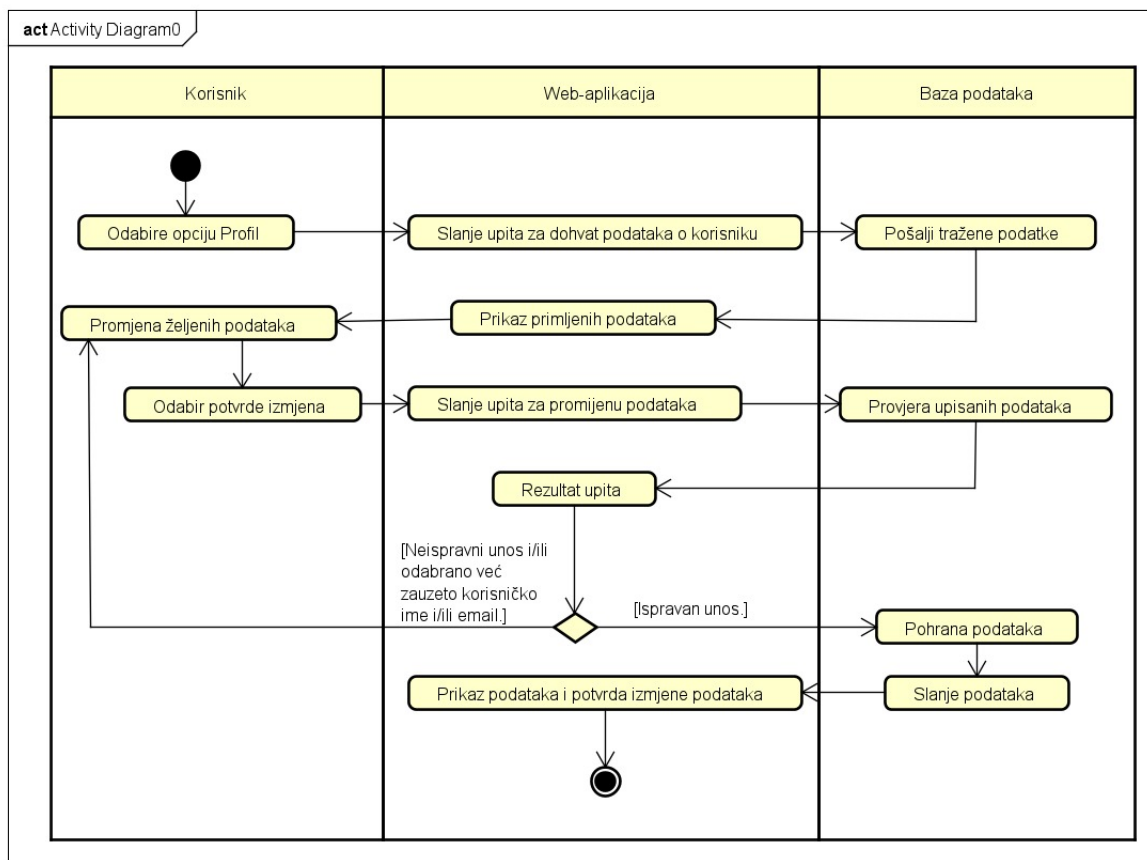
Dijagram stanja prikazuje stanja objekta te prijelaze iz jednog stanja u drugo temeljene na događajima. Na slici ispod prikazan je dijagram stanja za prijavljenog korisnika koji samo kupuje ono što drugi korisnici objave. Kada se kupac prijavljuje, otvara mu se početna strana na kojoj vidi trenutne aktivne oglase. Također tu može pretražiti oglase po svome izboru te može rezervirati željene oglase. Ukoliko klikne na "Profil" otvaraju mu se svi trenutni podatci te ih može urediti. Ukoliko se odluči za "Poruke" tamo može pregledati sve primljene i poslane poruke. Ukoliko klikne na gumb "Moji oglasi" tamo može pregledati i pretražiti rezervirane i kupljene oglase. Konačno ako pritisne gumb "Odjavi", korisnik se odjavljuje te mu se prikazuje stranica za prijavu.



Slika 4.7: Dijagram stanja

4.4 Dijagram aktivnosti

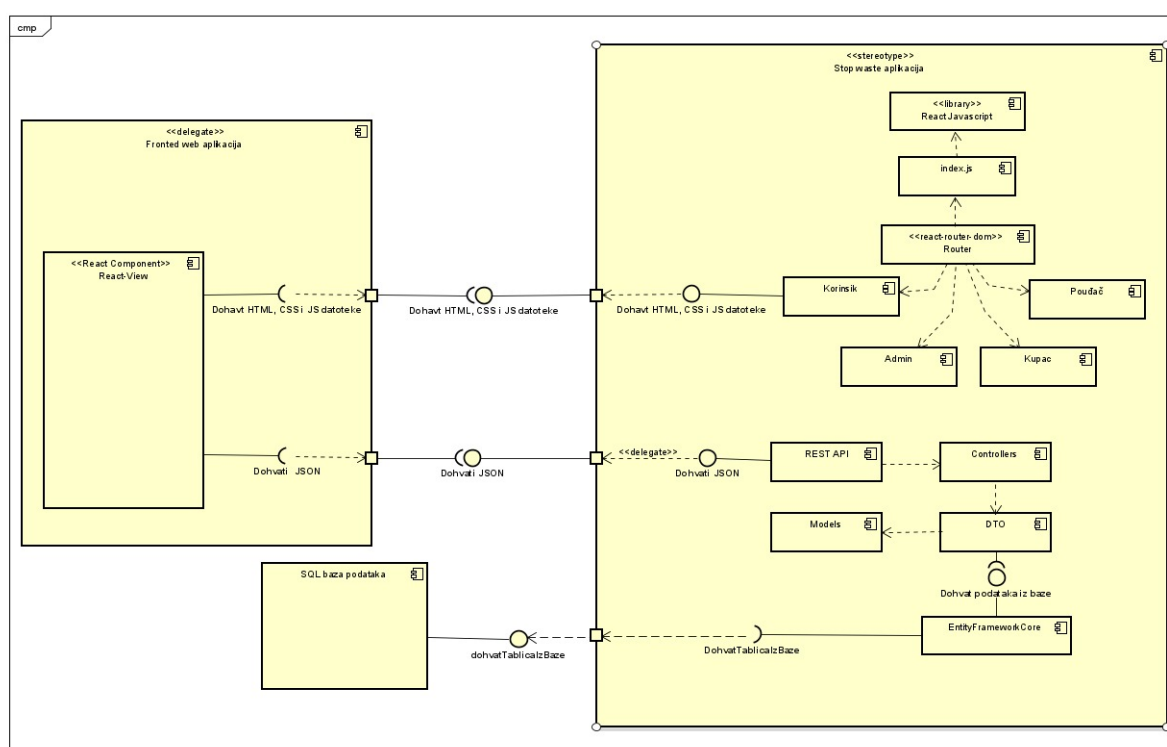
Dijagram aktivnosti primjenjuje se za opis modela toka upravljanja ili toka podataka. U modeliranju toka upravljanja svaki novi korak poduzima se nakon završenog prethodnog. Na dijagramu aktivnosti 4.8 prikazan je proces promjene osobnih podataka registriranog korisnika. Nakon što se korisnik prijavi u sustav, odabire Profil gdje mu se prikazuju njegovi trenutni osobi podaci koje je unio prilikom registracije. Korisnik zatim izmijeni željene podatke i odabere opciju Uredi. Baza podataka se ažurira s novim podacima. Korisniku se prikaže profil s izmijenjenim podacima uz potvrdu o uspješnosti izmjene.



Slika 4.8: Dijagram aktivnosti

4.5 Dijagram komponenti

Dijagram komponenti prikazan na slici 4.9 opisuje organizaciju i zavisnost programskih komponenti, interne strukture i odnose prema okolini. Sustavu se pristupa preko dva različita sučelja. Preko sučelja za dohvat HTML, CSS i JS datoteka poslužuju se datoteke koje pripadaju frontend dijelu aplikacije. Router je komponenta koja na upit s url određuje koja datoteka će se poslužiti na sučelje. Frontend dio se sastoji od niza JavaScript datoteka koje su raspoređene u logičke cjeline nazvane po tipovima aktera koji im pristupaju. Sve JavaScript datoteke ovise o React biblioteci iz koje dohvaćaju gotove komponente kao što su gumbi, forme i slično. Preko sučelja za dohvat JSON podataka pristupa se REST API komponenti. REST API poslužuje podatke koji pripadaju backend dijelu aplikacije. EntityFramework-Core je zadužen za dohvaćanje tablica iz baze podataka pomoću SQL upita. Podaci koji su pristigli iz baze se šalju dalje MVC arhitekturi u obliku DTO(Data transfer object). React-view komponenta preko dostupnih sučelja komunicira sa Stop waste aplikacijom te ovisno o korisnikovim akcijama osvježava prikaz i dohvaća nove podatke ili datoteke.



Slika 4.9: Dijagram komponenti

5. Implementacija i korisničko sučelje

5.1 Korištene tehnologije i alati

Komunikacija unutar tima:

WhatsApp

Komunikacija unutar tima ostvarena je korištenjem aplikacije WhatsApp. WhatsApp je aplikacija za komunikaciju korisnika. Omogućava komunikaciju u stvarnom vremenu putem tekstualnih poruka, glasovnih poruka ili videopoziva. Podržava mogućnost dijeljenja slika, dokumenata i videa.

Službena stranica: <https://www.whatsapp.com/>

Microsoft Teams

Video sastanci tima, zajedničko uređivanje i diejljenje dokumenata održavani su putem aplikacije Microsoft Teams. Microsoft Teams je aplikacija koja nudi radni prostor timovima u obliku videokonferencija, poruka, pohrane datoteka i zajedničkog uređivanja dokumenata u stvarnom vremenu.

Službena stranica: <https://www.microsoft.com/hr-hr/microsoft-365/microsoft-teams/group-chat-software>

Dokumentacija:

Astah UML

Za izradu svih UML dijagrama korišten je Astah UML. Astah UML je alat za stvaranje UML dijagrama. Omogućava stvaranje dijagrama obrazaca uporabe, dijagrama razreda, dijagrama stanja, dijagrama aktivnosti, sekvencijskih dijagrama, dijagrama komponenata, dijagrama komunikacije, složene strukture, razmještaja i umnih mapa.

Službena stranica: <https://astah.net>

Texmaker

Za pisanje dokumentacije koristili smo Texmaker. Texmaker je besplatni višeplatformski

LaTeX uređivač. Uključuje mnoge alate potrebne za razvoj dokumenata s LaTeX-om, podršku za unicode, provjeru pravopisa, automatsko dovršavanje i ugrađeni pdf preglednik.

Službena stranica: <https://www.xmlmath.net/texmaker/>

Rad na aplikaciji:

Git i GitLab

Git je korišten kao sustav za upravljanje izvornim kodom, dok je udaljeni re-ozitirij projekta dostupan na web platformi GitLab. Git je sustav otvorenog koda namijenjen upravljanju izvornim kodom. Pruža podršku za razvoj, stvaranje i spajanje grana. GitLab je web platforma koja pruža upravljanje Git repozitotijem i kontinuiranu integraciju uz praćenje verzija.

Službene stranice: <https://git-scm.com/>
<https://gitlab.com/>

H2

Kao početna baza podataka koristio se H2 sustav. H2 je sustav za upravljanje relacijskom bazom podataka napisanom u programskom jeziku Java. Omogućava čuvanje svih podataka u memoriji i podaci se osvježavaju pri svakom pokretanju aplikacije.

Službena stranica: <https://www.h2database.com/html/main.html>

PostgreSQL i pgAdmin

Stalna baza u aplikaciji izrađena je pomoću PostgreSQL u pgAdminu. PostgreSQL je bespaltn sustav otvorenog koda za upravljanje relacijskom bazom podataka. Podržava SQL upite i provjerava sigurnost. PgAdmin je bespaltni administrator alat s grafičkim korisničkim sučeljem za upravljanje PostgreSQL-om.

Službena stranica: <https://www.pgadmin.org/>

Intellij IDEA

Za pisanje koda korišten je Intellij IDEA Intellij IDEA je integrirana razvojna okolina za razvoj programske potpore. Sadrži integrirani sustav za upravljanje inačicama. Najpopularnija razvojna okolina za pisanje programske potpore u programskom jeziku Java.

Službena stranica: <https://www.jetbrains.com/idea/>

HTML

Za prikaz web aplikacije koristi se HTML. HTML je standardni označni jezik i u njemu se pišu dokumenti koji se prikazuju u web pregledniku. Opisuje semantičku strukturu web stranice.

CSS

Stilska prezentacija web stranice izvedena je pomoću CSS-a. CSS je stilski jezik za opis prezentacije dokumenta napisanog u označnom jeziku i omogućava odvajanje sadržaja od prezentacije što omogućava veću fleksibilnost pri izradi.

Za izradu fronteda koristili smo React i JavaScript.

JavaScript

Dinamičko kreiranje web stranice izvedeno je pomoću JavaScript-a. JavaScript je skriptni programski jezik koji se izvršava u internet pregledniku na strani korisnika.

Službena stranica: <https://www.javascript.com/>

React

React je biblioteka u JavaScriptu za izgradnju korisničkih sučelja. Održavana je od strane Facebooka. React se najčešće koristi kao osnova u razvoju web ili mobilnih aplikacija.

Službena stranica: <https://reactjs.org/>

SpringBoot

Za izradu pozadinske aplikacije (engl. back-end) koristili smo SpringBoot. Pozadinska aplikacija pisana je u razvojnom okruženju IntelliJ, koje je omogućilo korištenje okvira Spring Boot koji je baziran na programskom jeziku Java. Spring Boot je projekt koji se oslanja na Spring Framework i koji omogućuje puno učinkovitiji i brži pristup izgradnji Spring aplikacija.

Službena stranica: <https://spring.io/projects/spring-boot>

5.2 Ispitivanje programskog rješenja

Provedeno je ispitivanje programskog rješenja u svrhu pronalaska grešaka i nepredviđenog ponašanja sustava na korisnikove akcije. Testiranje se sastoji od dva dijela, testiranja komponenti sustava te testiranje sustava u cjelini. U svrhu testiranja komponenti programskog rješenja korišten je alat JUnit. Ponašanje sustava je testirano pomoću alata Selenium WebDriver za Google Chrome

5.2.1 Ispitivanje komponenti

Ispitivanjem komponenti testirana je funkcionalnost razreda koji implementiraju temeljne funkcionalnosti sustava.

Ispitni slučaj 1: Testiranje dohvata podataka o korisniku iz baze podataka premu username-u

Testira se dohvata li metoda `findByUsername(String username)` doista korisnika čije smo podatke zapisali u bazu podataka.

```
@DataJpaTest
class UserRepositoryTest {

    ...

    @Autowired
    private TestEntityManager entityManager;

    @Autowired
    private UserRepository userRepository;

    @Test
    public void whenFindByUsername_thenReturnUser() {
        User user = new User("test", "test@test.hr", "testing", "Test", "Test");
        entityManager.persist(user);
        entityManager.flush();

        Optional<User> found = userRepository.findByUsername(user.getUsername());

        assert found.isPresent();
        assertEquals(user.getUsername(), found.get().getUsername());
    }

    ...
}
```

Slika 5.1: Testiranje dohvata korisnika po username-ui

Ispitni slučaj 2: Testiranje postoji li korisnik u bazi podataka po username-u

Testira se funkcionalnost metode `findByUsername(String username)` tako da se pomoću metode `existsByUsername(String username)` provjerava postoji li korisnik u bazi podataka ako postoji da će metoda `findByUsername(String username)`

također pronaći zapis u korisniku u bazi podataka, u suprotnom nijedna od navedenih metoda neće pronaći podatke o korisniku.

```
@DataJpaTest
class UserRepositoryTest {
    ...

    @Autowired
    private TestEntityManager entityManager;

    @Autowired
    private UserRepository userRepository;

    @Test
    public void whenExistsByUsername_thenReturnUser() {
        User user = new User("test", "test@test.hr", "testing", "Test", "Test");
        entityManager.persist(user);
        entityManager.flush();

        if (userRepository.existsByUsername("test")) {
            assert userRepository.findByUsername("test").isPresent();
        } else {
            assert userRepository.findByUsername("test").isEmpty();
        }
        if (userRepository.existsByUsername("test1")) {
            assert userRepository.findByUsername("test1").isPresent();
        } else {
            assert userRepository.findByUsername("test1").isEmpty();
        }
    }
    ...
}
```

Slika 5.2: Testiranje postoji li korisnik u bazi podataka po username-u

Ispitni slučaj 3: Testiranje postoji li korisnik u bazi podataka po email-u

Testira se funkcionalnost metode `findByEmail(String email)` tako da se pomoću metode `existsByEmail(String email)` provjerava postoji li korisnik u bazi podataka te ako postoji da će metoda `findByEmail(String email)` također pronaći zapis o korisniku u bazi podatak, u suprotnom nijedna od navedenih metoda neće pronaći podatke o korisniku.

```

@DataJpaTest
class UserRepositoryTest {
...

@Autowired
private TestEntityManager entityManager;

@Autowired
private UserRepository userRepository;

@Test
public void whenExistsByEmail_thenReturnUser() {
    User user = new User("test", "test@test.hr", "testing", "Test",
"Test");
    entityManager.persist(user);
    entityManager.flush();

    if (userRepository.existsByEmail("test@test.hr")) {
assertTrue(userRepository.findByEmail("test@test.hr").isPresent());
    } else {
        assertTrue(userRepository.findByEmail("test@test.hr").isEmpty());
    }
    if (userRepository.existsByEmail("test1@test.hr")) {
assertTrue(userRepository.findByEmail("test1@test.hr").isPresent());
    } else {
        assertTrue(userRepository.findByEmail("test1@test.hr").isEmpty());
    }
    }
...
}

```

Slika 5.3: Testiranje postoji li korisnik u bazi podataka po email-u

U svrhu sljedećih ispitivanja koriste se dvojnici odnosno Mock objekti kojima možemo simulirati radi stvarnih objekata

```

@Mock
UserRepository userRepository = mock(UserRepository.class);
@Mock
AddressService addressService;
@Mock
RoleRepository roleRepository = mock(RoleRepository.class);
@Mock
JwtUtils jwtUtils;
@Mock
CategoryService categoryService;
@Mock
PasswordEncoder passwordEncoder = mock(PasswordEncoder.class);
@Mock
AuthenticationManager authenticationManager;
@Spy
@InjectMocks
UserServiceJpa userService = new UserServiceJpa(userRepository,
addressService, roleRepository, jwtUtils, categoryService,
passwordEncoder, authenticationManager);

```

Slika 5.4: Mock objekti

Ispitni slučaj 4: Testiranje registracije Testiramo dobivamo li za povratnu vrijednost metode za registraciju korisnika doista korisnika kojeg smo htjeli registrirati.

```
@RunWith(SpringJUnit4ClassRunner.class)
class UserServiceJpaTest {
    ...

    @Test
    public void registerUserReturnsUser() {
        User testUser = new User("test", "test@test.hr", "newPassword", null, null);
        when(passwordEncoder.encode(any(String.class))).thenReturn("newPassword");
        when(userRepository.save(any(User.class))).thenReturn(testUser);
        when(roleRepository.findByName(any(ERole.class))).thenReturn(Optional.of(new
        Role(ERole.ROLE_SELLER)));
        RegisterUserDTO registerUserDTO = new RegisterUserDTO("test",
        "test@test.hr", "testing");

        User user = userService.registerUser(registerUserDTO);
        assertEquals(registerUserDTO.getUsername(), user.getUsername());
    }
    ...
}
```

Slika 5.5: Testiranje registracije korisnika

Ispitni slučaj 5 : Testiranje jedinstvenosti email-a

Testiramo mogu li se registrirati dva korisnika s identičnom email adresom, očekivano ponašanje je bacanje iznimke RequestDeniedException.

```
@RunWith(SpringJUnit4ClassRunner.class)
class UserServiceJpaTest {
    ...

    @Test
    public void registerUserWithSameEmailThrows() {
        User testUser = new User("test", "test@test.hr", "newPassword", null, null);
        when(passwordEncoder.encode(any(String.class))).thenReturn("newPassword");
        when(userRepository.save(any(User.class))).thenReturn(testUser);
        when(userRepository.existsByEmail("test@test.hr")).thenReturn(true);
        when(roleRepository.findByName(any(ERole.class))).thenReturn(Optional.of(new
        Role(ERole.ROLE_SELLER)));
        RegisterUserDTO registerUserDTO = new RegisterUserDTO("test", "test@test.hr",
        "testing");

        assertThrows(RequestDeniedException.class, () ->
        userService.registerUser(registerUserDTO));
    }
    ...
}
```

Slika 5.6: Testiranje jedinstvenosti email-a

Ispitni slučaj 6 : Testiranje jedinstvenosti username-a

Testiramo mogu li se registrirati dva korisnika s identičnim korisničkim imenom, očekivano ponašanje je bacanje iznimke RequestDeniedException.

```
@RunWith(SpringJUnit4ClassRunner.class)
class UserServiceJpaTest {
    ...

    @Test
    public void registerUserWithSameEmailThrows() {
        User testUser = new User("test", "test@test.hr", "newPassword", null, null);
        when(passwordEncoder.encode(any(String.class))).thenReturn("newPassword");
        when(userRepository.save(any(User.class))).thenReturn(testUser);
        when(userRepository.existsByEmail("test@test.hr")).thenReturn(true);
        when(roleRepository.findByName(any(ERole.class))).thenReturn(Optional.of(new
        Role(ERole.ROLE_SELLER)));
        RegisterUserDTO registerUserDTO = new RegisterUserDTO("test", "test@test.hr",
        "testing");

        assertThrows(RequestDeniedException.class, () ->
        userService.registerUser(registerUserDTO));
    }
    ...
}
```

Slika 5.7: Testiranje jedinstvenosti username-a

5.2.2 Ispitivanje sustava

Ispitivanje sustava je provedeno pomoću alata Selenium WebDriver za web preglednik Google Chrome. Testirani su slučajevi registracije i prijave korisnika te objava oglasa, rezervacija oglasa i slanje poruka.

Ispitni slučaj 1: Ispitivanje registracije

Pomoću konfiguriranog Chrome drivrera dohvaća se stranica za registraciju i šalju se potrebni podatci za registraciju korisnika, prvim izvođenjem testa email korisnika i korisničko ime su jedinstveni te je očekivani rezultat uspješna registracija.

```
@Test
public void testRegistrationSuccessful() {
    driver.get(BASE_URL);
    WebElement registraiija = driver.findElement(By.linkText("Registracija"));
    registraiija.click();
    assertEquals(BASE_URL + "/registracija", driver.getCurrentUrl());
    driver.findElement(By.name("username")).sendKeys("test user");
    driver.findElement(By.name("email")).sendKeys("user");
    Actions actions = new Actions(driver);
    actions.keyDown(Keys.LEFT_ALT).keyDown(Keys.LEFT_CONTROL);
    actions.sendKeys("V");
    actions.keyUp(Keys.LEFT_ALT).keyUp(Keys.LEFT_CONTROL);
    actions.build().perform();
    driver.findElement(By.name("email")).sendKeys("gmail.com");
    driver.findElement(By.name("password")).sendKeys("123456");
    driver.findElement(By.name("name")).sendKeys("user");
    driver.findElement(By.name("surname")).sendKeys("123456");
    driver.findElement(By.name("street")).sendKeys("unska");
    driver.findElement(By.name("streetNumber")).sendKeys("3");
    driver.findElement(By.name("postalCode")).sendKeys("10000");
    driver.findElement(By.name("city")).sendKeys("Zagreb");
    Select select = new Select(driver.findElement(By.name("role")));
    select.getOptions().get(1).click();
    driver.findElement(By.className("gumb1")).click();
    assertEquals("Korisnik uspješno registriran!",
    driver.findElement(By.xpath("/html/body/div/div/div/div[2]/form/div/div")).getText());
}
```

Slika 5.8: Testiranje registracije

Ispitni slučaj 2: Testiranje prijave postojećeg korisnika

Testiramo prijavu u aplikaciju pomoću unaprijed registriranog korisnika, očekivano ponašanje je uspješna prijava što dokazujemo ispisivanje korisničkog imena prijavljenog korisnika u gornjem desnom kutu stranice.

```
public class SeleniumTests {  
    ...  
    @Test  
    public void testLoginSuccessful() {  
        driver.get(BASE_URL);  
        WebElement prijava = driver.findElement(new By.LinkText("Prijava"));  
        prijava.click();  
        assertEquals(BASE_URL + "/prijava", driver.getCurrentUrl());  
        SignInUserDTO user = new SignInUserDTO("Josip", "123456");  
        WebElement username = driver.findElement(By.name("username"));  
        username.sendKeys(user.getUsername());  
        WebElement password = driver.findElement(By.name("password"));  
        password.sendKeys(user.getPassword());  
        WebElement loginButton = driver.findElement(By.className("gumb1"));  
        loginButton.click();  
        WebElement usernameElement = driver.findElement(By.className("korisnickoime"));  
        assertEquals("korisničko ime: " + user.getUsername(), usernameElement.getText());  
    }  
    ...  
}
```

Slika 5.9: Testiranje prijave postojećeg korisnika

Ispitni slučaj 3: Testiranje prijave nepostojećeg korisnika

Testiramo možemo li se prijaviti u aplikaciju s nepostojećim korisničkim imenom, očekivano ponašanje je odbijanje prijave u sustav te ispis prigodne poruke korisniku.

```
public class SeleniumTests {  
    ...  
    @Test  
    public void testLoginNotSuccessful() {  
        driver.get(BASE_URL);  
        WebElement prijava = driver.findElement(new By.LinkText("Prijava"));  
        prijava.click();  
        assertEquals(BASE_URL + "/prijava", driver.getCurrentUrl());  
        SignInUserDTO user = new SignInUserDTO("NoNameUser", "123456");  
        WebElement username = driver.findElement(By.name("username"));  
        username.sendKeys(user.getUsername());  
        WebElement password = driver.findElement(By.name("password"));  
        password.sendKeys(user.getPassword());  
        WebElement loginButton = driver.findElement(By.className("gumb1"));  
        loginButton.click();  
        WebElement alert = driver.findElement(By.className("alert-danger"));  
        assertEquals("Prijava nije uspjela. Provjerite unesene podatke. Ako nemate račun, prvo se registrirajte.",  
            alert.getText());  
    }  
    ...  
}
```

Slika 5.10: Testiranje prijave nepostojećeg korisnika

Ispitni slučaj 4: Testiranje slanja poruka Testiramo slanje poruka između dvaju postojećih korisnika. Prijavljujemo se na jedan korisnički račun (metoda login(String username, String password)), zatim šaljemo poruku korisniku i odjavljujemo se kako bismo se mogli prijaviti na drugi korisnički račun. Nakon prijave korisnika koji je trebao zaprimiti poruku i provjeramo postoji li poruka koju smo poslali.


```
@Test
public void sendMessage() {
    SignInUserDTO user1 = new SignInUserDTO("Josip", "123456");
    SignInUserDTO user2 = new SignInUserDTO("Josip2", "123456");
    login(user1.getUsername(), user1.getPassword());
    driver.findElement(By.linkText("Poruke")).click();
    driver.findElement(By.className("gumb1")).click();
    assertEquals(BASE_URL + "/novaporuka", driver.getCurrentUrl());
    driver.findElement(By.name("receiver")).sendKeys("Josip2");
    String textMessage = "Hello! I'm interested in your ad";
    driver.findElement(By.name("message")).sendKeys(textMessage);
    driver.findElement(By.className("gumb1")).click();
    driver.findElement(By.linkText("Odjava")).click();
    login(user2.getUsername(), user2.getPassword());
    driver.findElement(By.linkText("Poruke")).click();
    boolean messageReceived = false;
    List<WebElement> messages = driver.findElements(By.className("card-oglas"));
    for (WebElement message : messages) {
        WebElement sender = message.findElements(By.tagName("b")).get(0);
        if (!sender.getText().equals("Od: " + user1.getUsername()))
            continue;
        WebElement receiver = driver.findElements(By.tagName("b")).get(1);
        if (!receiver.getText().equals("Za: " + user2.getUsername()))
            continue;
        WebElement textMessageElement = message.findElements(By.tagName("p")).get(2);
        if (textMessageElement.getText().equals(textMessage))
            messageReceived = true;
    }
    assertTrue(messageReceived);
}
```

Slika 5.11: Testiranje slanja poruke

Ispitni slučaj 5: Testiranje objave oglasa

Testiramo objavu oglasa tako s početne stranice te nakon toga provjeravamo je li se upravo objavljeni oglas pojavio u korisnikovoj listi njegovih objavljenih oglasa.

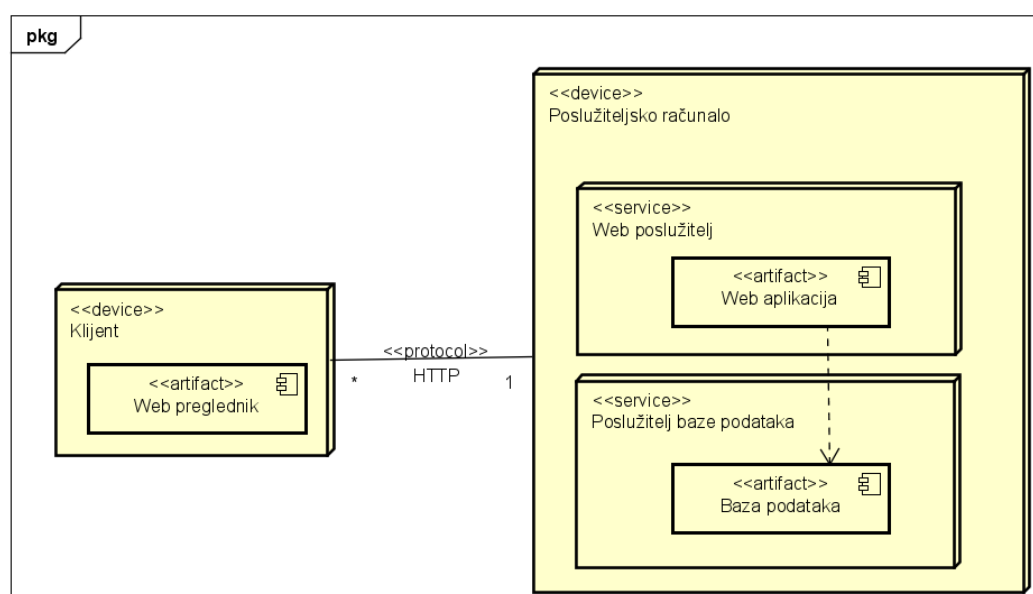
```
public class SeleniumTests { ...

@Test
public void postAd() {
    SignInUserDTO user = new SignInUserDTO("Josip", "123456");
    login(user.getUsername(), user.getPassword());
    driver.findElement(By.linkText("Dodaj oglas")).click();
    assertEquals(BASE_URL + "/novioglas", driver.getCurrentUrl());
    String title = "Kava", description = "Mljevena, pržena", location = "Požega", price = "50", discount = "30";
    driver.findElement(By.name("title")).sendKeys(title);
    driver.findElements(By.name("password")).get(0).sendKeys(description);
    driver.findElements(By.name("password")).get(1).sendKeys(description);
    driver.findElement(By.name("price")).sendKeys(price);
    driver.findElement(By.name("discount")).sendKeys(discount);
    driver.findElement(By.name("deadline")).sendKeys("22012021", Keys.TAB, "1200");
    driver.findElement(By.name("pictureSource")).sendKeys("C:/Users/Korisnik/OneDrive/Slike/1588787534-489506-
edgarangry.png");
    driver.findElement(By.xpath("/html/body/div/div/div/div[2]/form/div[8]/button")).click();//predaj oglas
    assertEquals("Oglas objavljen!",
    driver.findElement(By.xpath("/html/body/div/div/div/div[2]/form/div[9]/div")).getText());
    driver.findElement(By.xpath("/html/body/div/div/div/div[1]/div[2]/a[4]")).click();//moji oglasi
    driver.findElement(By.xpath("/html/body/div/div/div/div[2]/div[1]/a[1]/button")).click();//predani oglasi
    List<WebElement> postedAds = driver.findElements(By.className("card-oglas"));
    boolean adPosted = false;
    for (int i = 0; i < postedAds.size(); i++) {
        WebElement ad = postedAds.get(i);
        WebElement adTitleAndDescription = ad.findElement(By.className("NasloviOpis"));
        String adTitle = adTitleAndDescription.findElement(By.tagName("h2")).getText();
        if (!adTitle.equals(title))
            continue;
        String adDescription = adTitleAndDescription.findElement(By.className("opis")).getText();
        if (!adDescription.equals(description))
            continue;
        WebElement priceAndDiscount = ad.findElement(By.xpath("//*[@id=\"root\"]/div/div/div[2]/div[" + (i+2) +
        "]/div[3]/p[1]"));
        if (priceAndDiscount.getText().contains(price + "kn" + ", " + discount + "%")) adPosted = true; }
    assertTrue(adPosted);
} ...}
```

Slika 5.12: Testiranje objave oglasa

5.3 Dijagram razmještaja

Dijagrami razmještaja opisuju topologiju sklopovlja i programsku potporu koja se koristi u implementaciji sustava u njegovom radnom okruženju. Na poslužiteljskom računalu se nalaze web poslužitelj i poslužitelj baze podataka. Klijenti koriste web preglednik kako bi pristupili web aplikaciji. Sustav je baziran na arhitekturi "klijent – poslužitelj", a komunikacija između računala korisnika (kupac, ponuđač, administrator) i poslužitelja odvija se preko HTTP veze.



Slika 5.13: Dijagram razmještaja

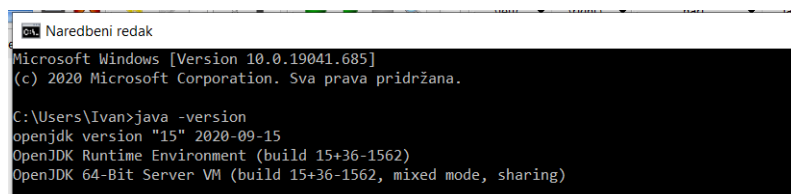
5.4 Upute za puštanje u pogon

Instalacija potrebnih aplikacija

Za pokretanje ove aplikacije kao poslužitelja potrebno je imati instalirano nekoliko aplikacija koje omogućavaju pokretanje i sam rad aplikacije.

Za bazu je potrebno instalirati pgAdmin koji omogućava povezivanje na PostgreSQL bazu podataka. Poveznica za preuzimanje je: <https://www.pgadmin.org/download/> ali se i u popisu literature nalazi poveznica na kojemu je moguće preuzeti pgAdmin te postoje detaljne upute kako se može postaviti sve potrebno za rad baze.

Na računalu je svakako potrebno imati podršku za prevođenje i pokretanje aplikacija napisanih u programskom jeziku Java. Također poveznica za preuzimanje potrebne podrške: <http://jdk.java.net/11/> te je dodana u popisu literature te se lako mogu pronaći upute koje će vas voditi kroz postavljanje svega potrebnog za uspostavu podrške za rad s programskim jezikom Java. Nakon instalacije je potrebno provjeriti radi li sve uspješno, a to je najjednostavnije provjeriti u naredbenom retku unosom naredbe "java -version". U danom primjeru je instalirana verzija 15, no za izvođenje programa je sasvim dovoljna verzija 11.

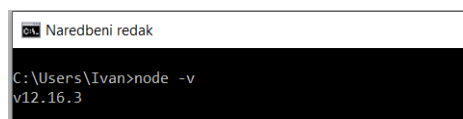


```
Naredbeni redak
Microsoft Windows [Version 10.0.19041.685]
(c) 2020 Microsoft Corporation. Sva prava pridržana.

C:\Users\Ivan>java -version
openjdk version "15" 2020-09-15
OpenJDK Runtime Environment (build 15+36-1562)
OpenJDK 64-Bit Server VM (build 15+36-1562, mixed mode, sharing)
```

Slika 5.14: Provjera instalacije Java

Kako se uz Javu koristi i programski jezik JavaScript potrebno je instalirati i njegovu podršku odnosno potrebno je instalirati Node.js. Poveznica za instalaciju Node.js-a: <https://nodejs.org/en/download/> te je dana u popisu literature te se jednostavno uz pomoć uputa instalira. Također je potrebno provjeriti uspješnost instaliranja Node.js-a u naredbenom retku naredbom: "node -v".



```
Naredbeni redak

C:\Users\Ivan>node -v
v12.16.3
```

Slika 5.15: Provjera instalacije Node.js-a

Kada je sve to spremno potrebno je instalirati razvojnu okolinu koja nam pomaže pri izradi same aplikacije i njenog pokretanja. Primjer jedne takve okoline je IntelliJ IDE koji je za studente FER-a besplatan, no moguće je koristiti i druge razvojne okoline poput Eclipsea. Poveznica za preuzimanje ove razvojne okoline: <https://www.jetbrains.com/idea/download/#section=windows> također se nalazi i u popisu literature.

Uz to za preuzimanje projekta možete se poslužiti sa Git GUI-em kojega je moguće preuzeti sa slijedeće poveznice: <https://git-scm.com/downloads>. Međutim možete samo skinuti projekt sa <https://gitlab.com/ivicamarica/ivicamarica> te ga pohraniti na željeno mjesto.

Za kraj morate na svome računalu imati i program za upravljanje projektima, u ovom slučaju je to Gradle. <https://gradle.org/install/> je poveznica za preuzimanje.

Kada ste sve to pripremili konačno možete pokrenuti aplikaciju. Za uspješno pokretanje aplikacije potrebno je prvo instalirati npm pakete koji se koriste. To se treba učiniti pokretanjem naredbe `npm install` unutar direktorija `src/main/js`. Spring Boot aplikaciju je moguće pokrenuti unutar IDE-a ili naredbom `gradle bootRun` iz naredbenog retka. Frontend se pokreće naredbom `npm start` unutar direktorija `src/main/js` nakon čega se aplikacija može pregledavati u pregledniku na adresi <http://localhost:3000>. Nakon što ste ovako pokrenuli aplikaciju možete ju koristiti jer će se ona sama pobrinuti za sve potrebne komponente poput stvaranja baze i ostaloga.

Ukoliko samo želite pogledati i isprobati aplikaciju, aplikacija je dostupna na poveznici <https://progi-stop-waste.herokuapp.com/>.

6. Zaključak i budući rad

Zadatak naše grupe bio je razvoj web aplikacije koja pridonosi smanjuje bacanje hrane, uz mogućnosti upravljanja rezervacijom, oglasima i porukama. Ostvarili smo zadani cilj nakon 15 tjedana rada u timu. Sama provedba projekta podjeljena je u dvije faze.

Prva faza projekta uključivala je okupljanje tima za razvoj aplikacije, dodjelu projektnog zadatka i intenzivan rad na dokumentiranju zahtjeva. Kvalitetna provedba prve faze uvelike je olakšala daljnji rad pri realizaciji osmišljenog sustava. Izrađeni obrasci i dijagrami (obraci uporabe, sekvencijski dijagrami, model baze podataka, dijagram razreda) bili su od pomoći podtimovima zaduženima za razvoj backenda i frontenda. Izrada vizualnih prikaza idejnih rješenja problemskih zadataka uštedjela je mnogo vremena u drugom ciklusu kada su članovi tima nailazili na nedoumice oko implementacije rješenja.

Druga faza ukazala je na manjak iskustva članova u izradi sličnih implementacijskih rješenja što je primorilo članove na samostalno učenje odabranih alata i programskih jezika. Osim realizacije rješenja, u drugoj fazi je bilo potrebno dokumentirati ostale UML dijagrame (dijagram stanja, dijagram aktivnosti, komponentni dijagram i dijagram razmještaja) i izraditi popratnu dokumentaciju kako bi budući korisnici mogli lakše koristiti ili vršiti preinake na sustavu. Komunikacija među članovima tima bila je putem WhatsAppa i Microsoft Teamsa čime smo postigli informiranost svih članova grupe o napretku projekta.

OVDJE TREBA DODATI JOŠ FUNKCIONALNOSTI KOJA NISU IMPLEMENTIRANE U APLIKACIJI. Moguće proširenje postojeće inačice sustava je izrada mobilne aplikacije čime bi se cilj projektnog zadatka bio ostvaren u većoj mjeri no s web aplikacijom.

Sudjelovanje na ovakvom projektu bilo je vrijedno iskustvo svim članovima tima jer smo iskusili rad u timu i stekli nova znanja u odabranim alatima i programskim jezicima. Također, osjetili smo važnost dobre vremenske organiziranosti i koordiniranosti između članova tima. Svakako bi s više iskustava članova tima otvorenje projekta bilo brže i kvalitetnije no unatoč tome zadovoljni smo s potignutignućem te ćemo u narednom periodu nastojat poboljšati funkcionalnosti aplikacije.

Popis literature

1. Programsko inženjerstvo, FER ZEMRIS, <http://www.fer.hr/predmet/proinz>
2. The Unified Modeling Language, <https://www.uml-diagrams.org/>
3. Astah Community, <http://astah.net/editions/uml-new>
4. ERDPlus, <https://erdplus.com/>
5. Slika StopWaste (pristupano 11.11.2020.), <https://www.google.com/url?sa=i&url=http%3A%2F%2Fwww.kllexportmalaysia.com%2F2017%2F02%2F21%2F5186%2F&psig=A0vVaw2oATDkNqo4mue1uyCEpcFt&ust=1605218845441000&source=images&cd=vfe&ved=0CAIQjRxqFwoTCLj9wrDA--wCFQAAAAAdAAAAABAI>
6. pgAdmin, <https://www.pgadmin.org/download/>
7. Java jdk 11, <http://jdk.java.net/11/>
8. Node.js, <https://nodejs.org/en/download/>
9. IntelliJ IDE, <https://www.jetbrains.com/idea/download/#section=windows>
10. Git, <https://git-scm.com/downloads>
11. Gradle, <https://gradle.org/install/>

Indeks slika i dijagrama

2.1	Ilustracija rada aplikacije, izvor[5]	7
3.1	Dijagram obrasca uporabe, funkcionalnosti korisnika i kupca	17
3.2	Dijagram obrasca uporabe, funkcionalnosti ponuđača i administratora	18
3.3	Sekvencijski dijagram za UC6	20
4.1	E-R dijagram baze podataka	28
4.2	Relacijski dijagram baze podataka	29
4.3	Dijagram razreda - domain	30
4.4	Dijagram razreda - dao	31
4.5	Dijagram razreda - Dto	32
4.6	Dijagram razreda - Servisi	33
4.7	Dijagram stanja	34
4.8	Dijagram aktivnosti	35
4.9	Dijagram komponenti	36
5.1	Testiranje dohvata korisnika po username-ui	40
5.2	Testiranje postoji li korisnik u bazi podataka po username-u	41
5.3	Testiranje postoji li korisnik u bazi podataka po email-u	42
5.4	Mock objekti	42
5.5	Testiranje registracije korisnika	43
5.6	Testiranje jedinstvenosti email-a	43
5.7	Testiranje jedinstvenosti username-a	44
5.8	Testiranje registracije	45
5.9	Testiranje prijave postojećeg korisnika	46
5.10	Testiranje prijave nepostojećeg korisnika	47
5.11	Testiranje slanja poruke	48
5.12	Testiranje objave oglasa	49
5.13	Dijagram razmještaja	50
5.14	Provjera instalacije Jave	51
5.15	Provjera instalacije Node.js-a	51

Dodatak: Prikaz aktivnosti grupe

Dnevnik sastajanja

1. sastanak

- Datum: 29. rujna 2020.
- Prisustvovali: K.Mišura, I.Aradski, P.Brala, I.Fičković, J.Kolarec, J.Lukačević, A.Rakocija
- Teme sastanka:
 - odabran voditelj tima i ime tima
 - odabrane pojedinosti za prijavu tima

2. sastanak

- Datum: 7. listopada 2020.
- Prisustvovali: K.Mišura, I.Aradski, P.Brala, J.Kolarec, J.Lukačević, A.Rakocija
- Teme sastanka:
 - sastanak s asistenticom i demosom
 - raščišćavanje dilema funkcionalnosti
 - analiza zadatka

3. sastanak

- Datum: 8. listopada 2020.
- Prisustvovali: K.Mišura, I.Aradski, P.Brala, I.Fičković, J.Kolarec, J.Lukačević, A.Rakocija
- Teme sastanka:
 - procjena trajanja projekta
 - procjena znanja i sposobnosti članova

4. sastanak

- Datum: 15. listopada 2020.
- Prisustvovali: K.Mišura, I.Aradski, P.Brala, I.Fičković, J.Kolarec, J.Lukačević, A.Rakocija
- Teme sastanka:
 - postavljanje gitlab repozitorija

- konačan odabir alata i tehnologija

5. sastanak

- Datum: 19. listopada 2020.
- Prisustvovali: K.Mišura, I.Aradski, P.Brala, I.Fićković, J.Kolarec, J.Lukačević, A.Rakocija
- Teme sastanka:
 - pregled modela baze
 - pregled zamišljenog izgleda aplikacije

6. sastanak

- Datum: 22. listopada 2020.
- Prisustvovali: K.Mišura, I.Aradski, P.Brala, I.Fićković, J.Kolarec, J.Lukačević, A.Rakocija
- Teme sastanka:
 - ispravljanje pogrešaka u modelu baze podataka

7. sastanak

- Datum: 28. listopada 2020.
- Prisustvovali: K.Mišura, I.Aradski, P.Brala, I.Fićković, J.Kolarec, J.Lukačević, A.Rakocija
- Teme sastanka:
 - podjela posla među članovima tima

8. sastanak

- Datum: 2. studenoga 2020.
- Prisustvovali: K.Mišura, I.Aradski, P.Brala, I.Fićković, J.Kolarec, J.Lukačević, A.Rakocija
- Teme sastanka:
 - postavljanje prvih uradaka na gitLab repozitorij

9. sastanak

- Datum: 3. studenoga 2020.
- Prisustvovali: K.Mišura, I.Aradski, P.Brala, I.Fićković, J.Kolarec, J.Lukačević, A.Rakocija
- Teme sastanka:
 - podjela uloga za rad na dokumentaciji

10. sastanak

- Datum: 9. prosinca 2020.

- Prisustvovali: K.Mišura, I.Aradski, P.Brala, I.Fićković, J.Kolarec, J.Lukačević, A.Rakocija
- Teme sastanka:
 - analiza projektnog zadatka
 - podjela zadataka među članovima

11. sastanak

- Datum: 20. prosinca 2020.
- Prisustvovali: K.Mišura, I.Aradski, P.Brala, I.Fićković, J.Kolarec, J.Lukačević, A.Rakocija
- Teme sastanka:
 - analiza projektnog zadatka
 - pregled trenutno napravljenog
 - podjela zadataka među članovima

12. sastanak

- Datum: 28. prosinca 2020.
- Prisustvovali: K.Mišura, I.Aradski, P.Brala, I.Fićković, J.Kolarec, J.Lukačević, A.Rakocija
- Teme sastanka:
 - analiza projektnog zadatka
 - pregled trenutno napravljenog
 - podjela zadataka među članovima
 - dogovor oko izrade baze

13. sastanak

- Datum: 29. prosinca 2020.
- Prisustvovali: K.Mišura, I.Aradski, P.Brala, I.Fićković, J.Kolarec, J.Lukačević, A.Rakocija
- Teme sastanka:
 - pregled izrađene baze

14. sastanak

- Datum: 4. siječnja 2021.
- Prisustvovali: K.Mišura, I.Aradski, P.Brala, I.Fićković, J.Kolarec, J.Lukačević, A.Rakocija
- Teme sastanka:
 - analiza projektnog zadatka
 - pregled trenutno napravljenog

- pregled dokumentacije
- podjela zadataka među članovima
- podjela zadataka za izradu dokumentacije među članovima

15. sastanak

- Datum: 8. siječnja 2020.
- Prisustvovali: K.Mišura, I.Aradski, P.Brala, I.Fićković, J.Kolarec, J.Lukačević, A.Rakocija
- Teme sastanka:
 - demonstracija alfa verzije

Tablica aktivnosti

	Katarina Mišura	Igor Aradski	Petra Brala	Ivan Ficković	Josip Kolarec	Josip Lukačević	Andrea Rakocija
Upravljanje projektom	6	1		3			
Opis projektnog zadatka			2			2	
Funkcionalni zahtjevi		2					
Opis pojedinih obrazaca		1					6
Dijagram obrazaca					2		
Sekvencijski dijagrami					2		
Opis ostalih zahtjeva							
Arhitektura i dizajn sustava	6	10		1			
Baza podataka	8	1		7			
Dijagram razreda			2	4			
Dijagram stanja				2			
Dijagram aktivnosti	2						
Dijagram komponenti							
Korištene tehnologije i alati	1						
Ispitivanje programskog rješenja	3	1		3		1	
Dijagram razmještaja							
Upute za puštanje u pogon		1		4			
Dnevnik sastajanja	6			1			
Zaključak i budući rad							
Popis literature				1			
Izrada početne stranice	1	10	3		3		20
Izrada baze podataka		1		4			
Spajanje s bazom podataka	1	5		3			
Back end	1	35	4	1		6	