Oblikovanje programske potpore

Ak. god. 2017./2018.

Sustav za upravljanje rada pizzerie

Dokumentacija, Rev. *1*

Grupa *Vrata*

Voditelj: *Josip Torić*

Datum predaje: *17.11.2017.*

Nastavnik: *Alan Jović*

Sadržaj

Sadržaj bi se trebao automatski osvježavati prema tekstu (desni klik, „Update Field“) ako se bude držalo zadanih formata poglavlja.

1. Dnevnik promjena dokumentacije

Rev.	Opis promjene/dodatka	Autor(i)	Datum
0.1	Napravljen predložak.	Mrđen	16.10.2017.
0.2	Dodani funkcionalni zahtjevi.	Frković, Grgić	19.10.2017.
0.4	Dodani use case dijagrami	Mrđen, Frković, Grgić	28.10.2017
0.5	Dodani sekvencijski dijagrami	Mrđen, Torić, Frković, Grgić, Crnomarković, Kudra	29.10.2017
0.55	Dodan opis projekta	Torić	29.10.2017.
0.6	Dodani ostali zahtjevi	Mrđen, Kudra	29.10.2017.
0.65	Dodan dijagram razreda	Mrđen,Kudra, Crnomarković	15.11.2017.
0.7	Dodan dijagram objekata	Mrđen, Grgić, Frković	15.11.2017.
0.8	Dodana arhitektura sustava	Torić, Mrđen	17.11.2017.
0.85	Dodan ER dijagram baze podataka	Mrđen	17.11.2017
0.95	Dodani opisi dijagrama razreda i objekata	Svi	17.11.2017.
1.0	Dodani podaci oko sastanaka i aktivnosti	Mrđen	17.11.2017.
1.1.	Dovršena alpha verzija implementacije	Mrđen, Torić	8.1.2018.
1.15	Dodani dijagrami stanja i aktivnosti	Frković, Grgić, Crnomarković	10.1.2018.
1.30	Dodani komunikacijski dijagrami	Kudra	10.1.2018.
1.4	Dodani komponentni dijagram i dijagram razmještaja	Mrđen	15.1.2018.
1.5	Dodane upute za instalaciju	Mrđen	15.1.2018.
1.6	Dodane korištene tehnologije	Mrđen	15.1.2018.
1.8	Dovršena implementacija aplikacije	Mrđen, Torić, Crnomarković	18.1.2018.
1.9.	Dodane korisničke upute i testovi	Svi	18.1.2018.
2.0.	Dodana literatura i prikaz commitova	Mrđen	18.1.2018.

2. Opis projektnog zadatka

Cilj ovog projekta je razviti sustav za upravljanje rada pizzeriom, preciznije govoreći upravljanje online naručivanje pizza iz pizzerie. Sustav je namijenjen pizzeriama koje imaju dostavu pizze te žele unaprijediti i osuvremeniti način naručivanja pizze. Potencijalna korist ovog projekta je olakšavanje naručivanja pizze ljudima, a pizzeriama olakšavanje distribucije pizze. Olakšavanje naručivanje i distribucije pizze dovodi do veće potražnje pizze, a samim time i do povećanja kvalitete same pizze.

Slika 2.1. Logo pizzerije

Sustav je osmišljen kao web aplikacija koja se sastoji od tri cjeline, dio za korisnike, dio za djelatnike i dio za administratora.

Neprijavljeni korisnici mogu pogledati kompletnu ponudu pizzerie, vidjeti galeriju slika te sve opće informacije o pizzeriji.

Ulaskom na glavnu stranicu pizzerie korisnici bi dobili prikaz glavnih usluga koje ona nudi s ponudama različitih pizza. Na stranici bi se također nalazili i linkovi na galeriju slika, on-line naručivanje, sekciju za kontakt i informacije o pizzeriji.

Ponuda pizzerie se sastoji od prikaza svih pizza s njihovom ocjenom. Klikom na pojedini pizzu, otvara se stranica s detaljima pizze. Detalji pizze se sastoje od opisa pizze, njenih sastojaka, nutritivne vrijednosti i cijene. Također svaka pizza ima svoju mini galeriju gdje se vidi kako izgleda gotova pizza.

Klikom na galeriju slika s naslovne stranice može se vidjeti ambijent same pizzerie, proces izrade pizze i nasmijano osoblje.

Opće informacije pizzerie se sastoji od adrese pizzerie, lokacije na google karti, kontakt brojeva te opisa pizzerie.

Na sučelju neprijavljenih korisnika se u svakom momentu treba nuditi da se registriraju. Nuđenje registracije treba biti suptilno koliko je to moguće. Za registraciju je potrebno unijeti ime, prezime, e-mail, lozinku i broj telefona. Nakon uspješne registracije korisnik se može prijaviti u sustav.

Neprijavljenim korisnicima je mogućnost online naručivanja nije dostupna, samo prijavljeni korisnici mogu naručiti pizzu.

Prijavljeni korisnici mogu i naručiti pizzu i vidjeti sve svoje prethodne narudžbe. Nakon svake dostavljene pizze, kupce se potiče da ocjenu pizzu. Kupci vide program vjernosti pizzerie. Uvjete programa vjernosti pizzerie postavlja administrator, a u principu funkcionira da nakon određenog broja dostavljenih pizza, kupci imaju pravo na besplatne pizze. Program vjernosti ima u sebi track bar koji se polagano puni kako su kupci bliže nekom od pragova programa vjernosti, naravno u cilju da naručuju što više i više pizza. Prelaskom praga od 5 naručenih pizza dobila bi se mogućnost izabiranja pizze besplatno. Klikom na gumb za naručivanje, zaposleniku u pizzeriji bi se poslala poruka da je korisnik naručio pizzu.

Zaposlenici pizzerie imaju vlastito sučelje na kojem administriraju primljene narudžbe.

Kada narudžba stigne u sustav, ona je u statusu "NARUČENA". Tu narudžbu vide svi zaposlenici, a kada je neki zaposlenik prihvati, ona je u statusu "U PRIPRAVLJANJU". Narudžba se također onda sprema na njegovo ime, tj. pamti se u sustavu da ju je taj zaposlenik prihvatio. Kada se narudžba spremi i narudžba krene, tada bude u statusu "U DOSTAVI". Kada se vrati dostavljač s novcima, narudžba dođe u završni status "PLAĆENA".

No, ako je došlo do nekih poteškoća s kupcem i narudžba nije dostavljena ili nije plaćena, tada narudžba bude u statusu "NIJE
PLAĆENA".

Kada narudžba stigne u roku od 3 minute zaposlenik je dužan obavijestiti korisnika za koliko će mu vremena stići pizza. Također kada dostavljač krene na korisnikovu lokaciju, zaposlenik je dužan obavijestiti korisnika da mu je pizza krenula. Nadalje, kada se dostavljač vrati s novcima korisnika, zaposlenik je dužan u sustav unijeti da je pizza plaćena, no za ovu naredbu ne treba posebno obavijestiti korisnika.

Zaposlenik sustava također može privremeno onemogućiti nove narudžbe za slučaj kada gužva u pizzeriji postane prevelika. Korisniku bi se prikazala poruka da su narudžbe trenutno onemogućene te da pričeka neko vrijeme.

Zaposlenik ima sučelje na kojem vidi sve prethodne današnje narudžbe na njegovo ime i sve aktualne narudžbe.

Administrator sustava može uređivati ponudu pizza, administrirati korisnike te vidjeti izvješća za pizzeriu.

Administrator može dodavati nove pizze, uređivati postojeće pizze i brisati pizze iz ponude. Svaka pizza (govoreći kao entitet) se sastoji od imena pizze, opisa pizze, sastojaka u pizzi, nutritivne vrijednosti (kao deskripciju) i cijene, te naravno slika same pizze.

Administrator ima uvid u sve korisnike. On može dodavati nove korisnike u sustav, uređivati postojeće korisnike i brisati korisnike. Ove operacije se prvenstveno odnose na dodavanje zaposlenika u pizzeriu i brisanje istih, iako administrator ima mogućnost za brisanje i dodavanje svih korisnika. Brisanje ljudi koji su korisnici pizzerije treba biti omogućeno zbog ljudi koji pokušaju prevariti pizzeriu.

Administrator sustava ne može vidjeti trenutnu lozinku korisnika, iako može na zahtjev korisnika im postaviti novu lozinku za koji će im savjetovati da ga što prije promjene.

Klikom na zaposlenika otvara se stranica s detaljima zaposlenika gdje se mogu uređivati svi njegovi ostali podaci. Prikaz svih zaposlenika je u tablici u kojoj su stupci ime, prezime, datum rođenja, dan zaposlenja te broj izdanih pizza. Prikaz svih kupaca je također u tablici u kojoj se vidi ime, prezime, datum rođenja i broj naručenih pizza. Klikom na kupca, također se otvara nova stranica gdje on može uređivati osobne podatke korisnika.

Izvješća za pizzeriu trebaju biti tjedna, mjesečna, kvartarna, polugodišnja i godišnja. Uz prikaz na web stranici, izvještaji se mogu preuzeti u .xlsx formatu ili .pdf formatu. Svako izvješće sastoji se od ukupnog broja naručenih pizza, broj naručenih pizza po vrsti pizze te broj naručenih pizza po svakom satu radnog vremena. Nadalje u izvješću treba biti prosječno vrijeme dostave pizze, prosječno vrijeme dostave po svakom satu, prosječna ocjena pizza te prosječna ocjena po svakoj pizzi.

Nakon završetka projekta dobro bi bilo razmotriti mogućnosti razvijanja mobilne aplikacije tako da bi ljudi mogli naručiti pizzu od bilo kud, a ne samo iz svog vlastitog doma. Mobilna aplikacija bi bila samo za korisnike, djelatnici pizze i administrator su puno produktivniji s računalom nego s mobitelom.

3. Pojmovnik

- **Angular** – Framework u JavaScriptu koji se koristi za razvoj Web aplikacija
- **Api** – Sučelje koja jedna aplikacije nudi prema drugoj, preko njega dvi aplikacije međusobno komuniciraju
- **Backend** – Server dio aplikacije, služi za logiku sustava i rad s bazom podataka
- **Bootstrap** – Framework u CSS koji sadrži velik broj predizajniranih stvari poput tablica, dugmadi, itd.
- **CSS** – Programski jezik koji služi za oblikovanje sadržaja u pregledniku
- **Framework** – Slojevita struktura pisana u nekom programskog jeziku koja rješava neke probleme i postavlja način kako se neke stvari rješavaju. Framework može sadržavati gotova rješenja nekih problema ili samo upute kako napraviti vlastita rješenja
- **Frontend** – Klijentski dio aplikacije, služi za prezentaciju sustava
- **HTML** - Programski jezik koji služi za prikaz sadržaja u pregledniku
- **HTTP** – Protokol za razmjenjivanje sadržaja na internetu
- **Java** – Objektno orijentirani programski jezik široke primjene
- **JavaScript** – Programski jezik koji služi za logiku sadržaja u pregledniku
- **Klijent – Server** : Klijent-server model je model aplikacija koji razdvaja zadatke prikaza podataka i same logike i pohranjivanja podataka. Koriste se dvije razdvojene aplikacije, jedna za klijent, druga za server koje komuniciraju preko REST api-a, dok jedna o drugoj ne znaju ama baš ništa.
- **PostreSQL** – Implementacija SQL baze podataka
- **REST** – Api sučelje u kojem dvije aplikacije komuniciraju putem HTTP metoda kao što su *GET, POST, PUT i DELETE*
- **Spring** – Framework za Java programski jezik koji olakšava razvoj web aplikacija
- **SQL** – Relacijska baza podataka
- **TypeScript** – Jezik nastao iz JavaScripta koji dodaje brojne funkcionalnosti u JavaScript

4. Funkcionalni zahtjevi

Dionici:

- neprijavljeni korisnici
- prijavljeni korisnici
- zaposlenici
- administrator
- baza podataka

Aktori i njihovi funkcionalni zahtjevi:

- Neprijavljeni korisnik,inicijator:
  - Može pregledavati ponudu pizzerie
  - Može vidjeti galeriju slika
  - Može vidjeti opće informacije o pizzeriji
  - Može se registrirati
  - Može se prijaviti
- Prijavljeni korisnik,inicijator
  - Može naručiti pizzu online
  - Može vidjeti svoje prethodne narudžbe
  - Može ocijeniti prethodno naručene pizze
  - Može vidjeti program vjernosti pizzerie
  - Može uređivati svoj profil
- Zaposlenik, inicijator
  - Može primati narudžbe i obrađivati ih
  - Može vidjeti narudžbe koje je isporučio tog dana
  - Može blokirati naručivanje pizza u slučaju gužve
- Administrator, inicijator
  - Može uređivati ponudu pizza
  - Može administrirati korisnike
  - Može gledati izvješća
  - Može dodavati i brisati zaposlenike
- Baza podataka, sudionik
  - Sadrži podatke o korisnicima sustava
  - Sadrži podatke o narudžbama
  - Sadrži podatke o pizzama
  - Sadrži podatke o izvješćima

Opis obrazaca uporabe:

UC1 – Registriraj se

- **Glavni sudionik:** Neprijavljeni korisnik
- **Cilj:** Izrada novog korisničkog profila
- **Sudionici:** Baza podataka
- **Rezultat:** Korisnik je registriran (izrađen je novi profil u sustavu).
- **Željeni scenarij:**
    1. Korisnik odabire opciju registracije.
    2. Korisnik unosi sve potrebne podatke u web obrascu i potvrđuje svoj unos.
    3. Aplikacija provjerava valjanost unesenih podataka, te provjerava da li korisnik već postoji u bazi podataka.
    4. Ako korisnik ne postoji u bazi podataka, aplikacija upisuje u bazu podataka podatke o novom korisniku, te šalje korisniku potvrdu o registraciji.
- **Mogući drugi scenariji:**
    4. Aplikacija ne može upisati korisnika u bazu podataka jer korisnik već postoji

UC2 – Prijavi se

- **Glavni sudionik**: Neprijavljeni korisnik
- **Cilj:** Prijava u sustav.
- **Sudionici:** Baza podataka.
- **Rezultat:** Korisnik je prijavljen u sustav .
- **Željeni scenarij:**
    1. Korisnik odabire opciju prijave u sustav.
    2. Korisnik unosi sve potrebne podatke u web obrascu i potvrđuje svoj unos.
    3. Provjerava se valjanost unesenih podataka i podatak postoji li korisnik u bazi podataka.
    4. Ako korisnik postoji u bazi podataka, prijavljuje se u sustav.
- **Mogući drugi scenariji:**
    4. Korisnik nije unio ispravne podatke potrebne za prijavu. Sustav ga upozorava i nudi opciju ponovne prijave.

UC3 – Uredi profil

- **Glavni sudionik:** Prijavljeni korisnik
- **Cilj:** Urediti korisničke podatke na profilu.
- **Sudionici:** Baza podataka.
- **Preduvjeti:** Korisnik je prijavljen u sustav.
- **Rezultat:** Unesene promjene na korisnikovom profilu .
- **Željeni scenarij:**
    1. Korisnik odabire opciju pregleda profila.
    2. Korisnik odabire uređivanje osobnih podataka.
    3. Korisnik potvrđuje unesene promjene.
    4. Promjene se upisuju u bazu podataka.

UC4 – Pogledaj informacije

- **Glavni sudionik:** Neprijavljeni korisnik i prijavljeni korisnik
- **Cilj:** Pregled informacija.
- **Sudionici:**
- **Rezultat:** Korisnik je pronašao sve željene informacije .
- **Željeni scenarij:**
    1. Korisnik odabire opciju pregleda informacija o pizzeriji ili opciju pregleda ponude pizza.
    2. Sustav ispisuje tražene informacije

UC5 – Naruči pizzu

- **Glavni sudionik:** Prijavljeni korisnik.
- **Cilj:** Naručiti pizzu.
- **Sudionici:** Baza podataka.
- **Preduvjeti:** Korisnik treba biti registriran i opcija naručivanja treba biti omogućena.
- **Rezultat:** Prikaz narudžbe zaposleniku.
- **Željeni scenarij:**
    1. Ako korisnik nije siguran koju pizzu želi naručiti može pogledati ponudu pizza i detalje o pizzama koje ga zanimaju.
    2. Korisnik odabire opiju naručivanja te naručuje pizzu.
    3. Sustav provjerava je li narudžba ispravna te u bazu podataka zapisuje narudžbu.
    4. Sustav šalje poruku o narudžbi zaposlenicima.

UC6 – Ocijeni pizzu

- **Glavni sudionik:** Prijavljeni korisnik.
- **Cilj:** Ocijeniti pizzu.
- **Sudionici:** Baza podataka.
- **Preduvjeti:** Korisnik treba biti registriran i narudžba treba biti isporučena i plaćena.
- **Rezultat:** Pohrana ocijene u bazu podataka.
- **Željeni scenarij:**
    1. Korisnik odabire opciju ocjenjivanja narudžbe.
    2. Sustav nudi korisniku da ocijeni posljednju plaćenu narudžbu.
    3. Korisnik ocjenjuje narudžbu s ocjenama u intervalu od 1-5.
    4. Ocjena se sprema u bazu podataka te se ispisuje poruka u kojoj se zahvaljuje korisniku na ocjeni.
- **Mogući drugi scenarij:**
    3. Korisnik ocjenjuje narudžbu s ocjenama izvan intervala od 1-5.
    4. Ocjena se ne može upisati u bazu podataka jer nije u ispravnom intervalu,pa se ispisuje prikladna poruku korisniku.

UC7 – Pogledaj program vjernosti

- **Glavni sudionik:** Prijavljeni korisnik.
- **Cilj:** Pogledati program vjernosti.
- **Sudionici:** Baza podataka.
- **Preduvjeti:** Korisnik treba biti registriran.
- **Rezultat:** Prikaz programa vjernosti korisniku.
- **Željeni scenarij:**
    1. Korisnik odabire opciju pregleda programa vjernosti.
    2. Aplikacija pronalazi u bazi podataka narudžbe koje je korisnik platio do tog trenutka te ih ispisuje.
    3. Korisnik ima uvid u broj narudžbi koje je do tada ostvario i upoznat je s brojem narudžbi koje treba ostvariti da bi dobio besplatnu pizzu..

UC8 – Pregledaj aktualne narudžbe

- **Glavni sudionik:** zaposlenik
- **Cilj:** pregledati sve narudžbe koje još nisu obrađene
- **Sudionici:** Korisnik, baza podataka
- **Preduvjeti:** prijava u sustav kao zaposlenik
- **Rezultat:** zaposlenik dobije prikaz aktualnih narudžbi
- **Željeni scenarij:**
    1. Zaposlenik na sučelju za zaposlenike odabire opciju za pregled aktualnih narudžbi
    2. Pregledava status i podatke o narudžbi koji su u bazi podataka temeljem kojih može dalje djelovati prema korisniku (obavijestiti ga o statusu narudžbe)
    3. Mijenja status narudžbe ovisno o stanju narudžbe

UC9 – Pregledaj prethodne narudžbe

- **Glavni sudionik:** zaposlenik
- **Cilj:** pregledati sve narudžbe koje su obrađene
- **Sudionici:** Baza podataka
- **Preduvjeti:** prijava u sustav kao zaposlenik
- **Rezultat:** zaposlenik dobije prikaz svih obrađenih naredbi tog dana
- **Željeni scenarij:**
    1. Zaposlenik na sučelju za zaposlenike odabire opciju za pregled prethodnih narudžbi
    2. Pregledava status i podatke o narudžbi koji su u bazi podataka

UC10 – Onemogući naručivanje

- **Glavni sudionik:** zaposlenik
- **Cilj:** u slučaju velike gužve privremeno onemogućiti naručivanje pizza
- **Sudionici:** Korisnik
- **Preduvjeti:** prijava u sustav kao zaposlenik, broj aktualnih narudžbi prelazi maksimum
- **Rezultat:** privremeno u sustav ne stižu narudžbe
- **Željeni scenarij:**
    1. Zaposlenik odabire onemogućavanje naručivanja
    2. Korisniku se prikazuje odgovarajuća poruka

UC11 – Prihvati narudžbu
- **Glavni sudionik:** zaposlenik
- **Cilj:** u sustavu je narudžba spremljena na ime zaposlenika te ju on obrađuje
- **Sudionici:** Korisnik, baza podataka
- **Preduvjeti:** zaposlenik je prijavljen u sustav, naručivanje je omogućeno, nije još prihvaćena od strane drugih zaposlenika
- **Rezultat:** podaci o narudžbi kod zaposlenika se nalaze u bazi podataka
- **Željeni scenarij:**
    1. Kada narudžba stigne u sustav, zaposleniku se prikaže mogućnost prihvaćanja narudžbe
    2. Nakon prihvaćanja, može ju vidjeti među aktualnim narudžbama

UC12 – Obavijesti korisnika
- **Glavni sudionik:** zaposlenik
- **Cilj:** obavijestiti korisnika o statusu narudžbe
- **Sudionici:** Korisnik
- **Preduvjeti:** zaposlenik je prijavljen u sustav, zaposlenik je prihvatio narudžbu od korisnika i ona je aktualna
- **Rezultat:** korisnik je obaviješten o svojoj narudžbi
- **Željeni scenarij:**
    1. Zaposlenik pregleda podatke o narudžbi korisnika
    2. Ovisno o statusu, zaposlenik upisuje tekst poruke i šalje obavijest korisniku

UC13 – Dodaj zaposlenika u sustav
- **Glavni sudionik:** administrator
- **Cilj:** dodati zaposlenika u sustav
- **Sudionici:** Baza podataka
- **Preduvjet:** Zaposlenik ne postoji već u bazi podataka
- **Rezultat:** U bazi podataka se nalaze podaci o novom zaposleniku
- **Željeni scenarij:**
    1. Upišu se ispravni podaci
    2. Podaci se unesu u bazu podataka
    3. Stvori se zaposlenikov račun
- **Mogući drugi scenarij:**
    1. Zaposlenikovi podaci nisu ispravni, odgovarajuća poruka

UC14 – Uredi postojeće podatke korisnika ili zaposlenika

- **Glavni sudionik:** administrator
- **Cilj:** urediti i izmijeniti postojeće podatke korisnika
- **Sudionici:** Baza podataka
- **Preduvjet:** korisnik je prijavljen u sustav kao administrator
- **Rezultat:** u bazi podataka se nalaze izmijenjeni podaci o korisniku/zaposleniku
- **Željeni scenarij:**
    1. Klikom na korisnika administrator dobiva uvid u njegove podatke
    2. Uređuje podatke

UC15 – Obriši korisnika

- **Glavni sudionik:** administrator
- **Cilj:** brisanje korisnika ili zaposlenika i svih njegovih podataka iz baze podataka
- **Sudionici:** Baza podataka
- **Preduvjeti:** korisnik kojeg se briše je registriran u sustavu i nije dugo imao nikakve narudžbe
- **Rezultat:** korisnik više nije u bazi podataka
- **Željeni scenarij:**
    1. Klikom na korisnika administrator dobiva uvid u njegove podatke
    2. Odabire opciju za brisanje korisnika

UC16 – Dodaj pizzu u ponudu

- **Glavni sudionik:** administrator
- **Cilj:** dodavanje nove pizze u ponudu pizzerie i svih njenih podataka u bazu
- **Sudionici:** baza podataka
- **Preduvjeti:** ispravni podaci
- **Rezultat:** u bazi podataka se nalaze podaci o novoj pizzi
- **Željeni scenarij:**
    1. Administrator klikne na pregled ponude pizzerie
    2. Odabire opciju za dodavanje novih pizza
    3. Ispuni sve podatke i doda sliku pizze
    4. Dodaje pizzu u ponudu
- **Mogući drugi scenarij:**
    1. Podaci su neispravni ili pizza već postoji u ponudi

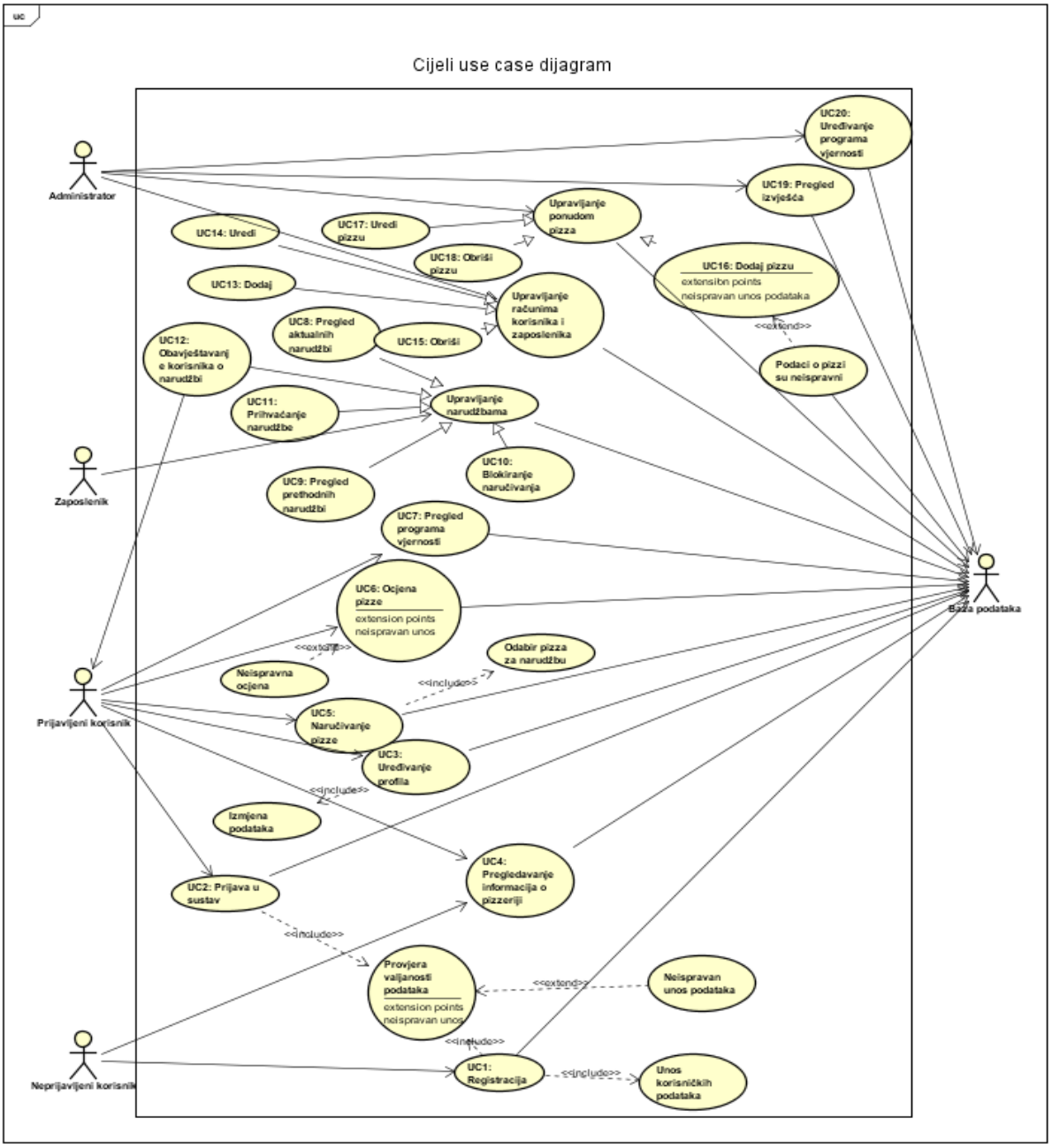
UC17 – Uredi ponudu pizza
- **Glavni sudionik:** administrator
- **Cilj:** izmjena postojećih podataka o pizzama u ponudi pizzerie u bazi podataka
- **Sudionici:** baza podataka
- **Preduvjeti:** korisnik je prijavljen u sustav kao administrator
- **Rezultat:** podaci o pizzi su izmijenjeni u bazi podataka
- **Željeni scenarij:**
    1. Klikne na pregled ponude pizzerie
    2. Odabire opciju za uređivanje ponude
    3. Odabire pizzu i izmijeni podatke
- **Mogući drugi scenarij:**
    1. Podaci su neispravni

UC18 – Obriši pizzu iz ponude
- **Glavni sudionik:** administrator
- **Cilj:** obrisati pizzu iz ponude pizzerie i baze podataka
- **Sudionici:** baza podataka
- **Preduvjeti:** pizza postoji u ponudi, tj. njezini podaci su u bazi podataka
- **Rezultat:** podaci o pizzi više nisu u bazi podataka
- **Željeni scenarij:**
    1. Klikne na pregled ponude pizzerie
    2. Odabere pizzu i opciju za brisanje

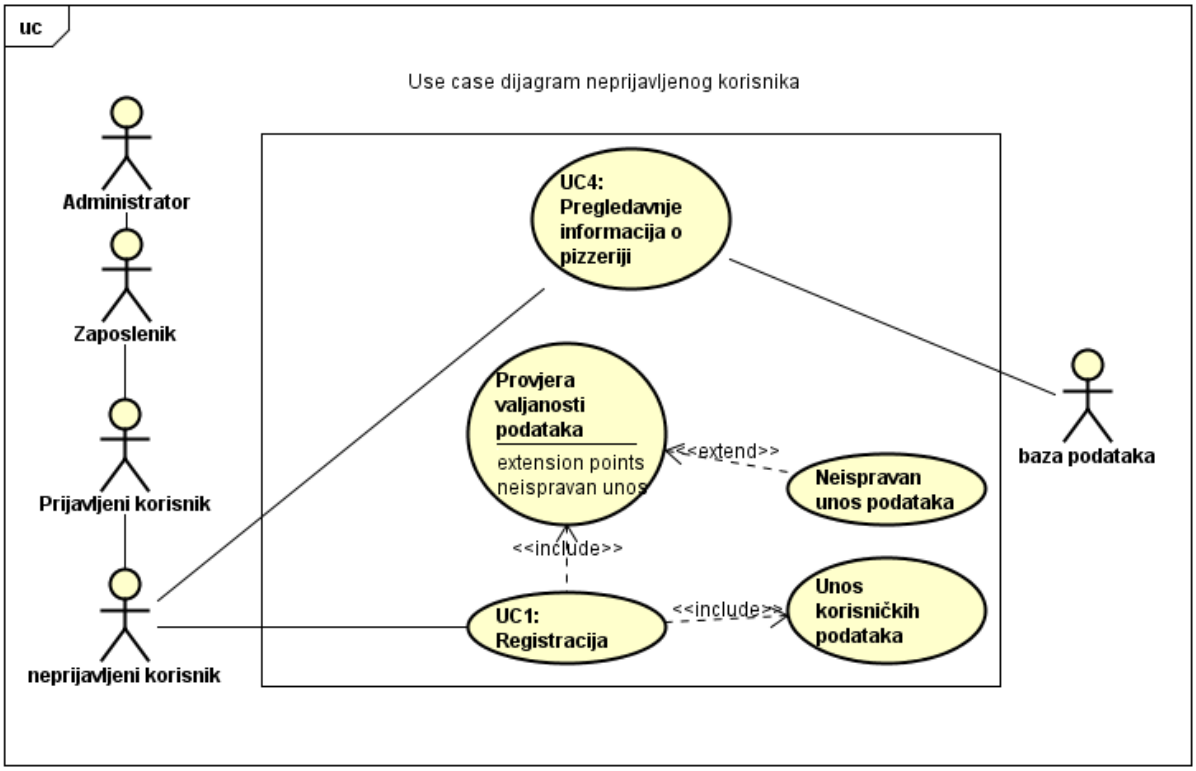
UC19 – Pregledaj izvješće
- **Glavni sudionik:** administrator
- **Cilj:** odabrati izvješće, pregledati ga
- **Sudionici:** Baza podataka
- **Preduvjeti:** korisnik je prijavljen u sustav kao administrator
- **Rezultat:** pregled izvješća na web stranici ili nakon što je preuzeto u nekom od mogućih formata
- **Željeni scenarij:**
    1. Klikne na pregled svih izvješća
    2. Klikom na određeno izvješće vidi podatke, koje zatim može i preuzeti

UC20 – Uredi program vjernosti

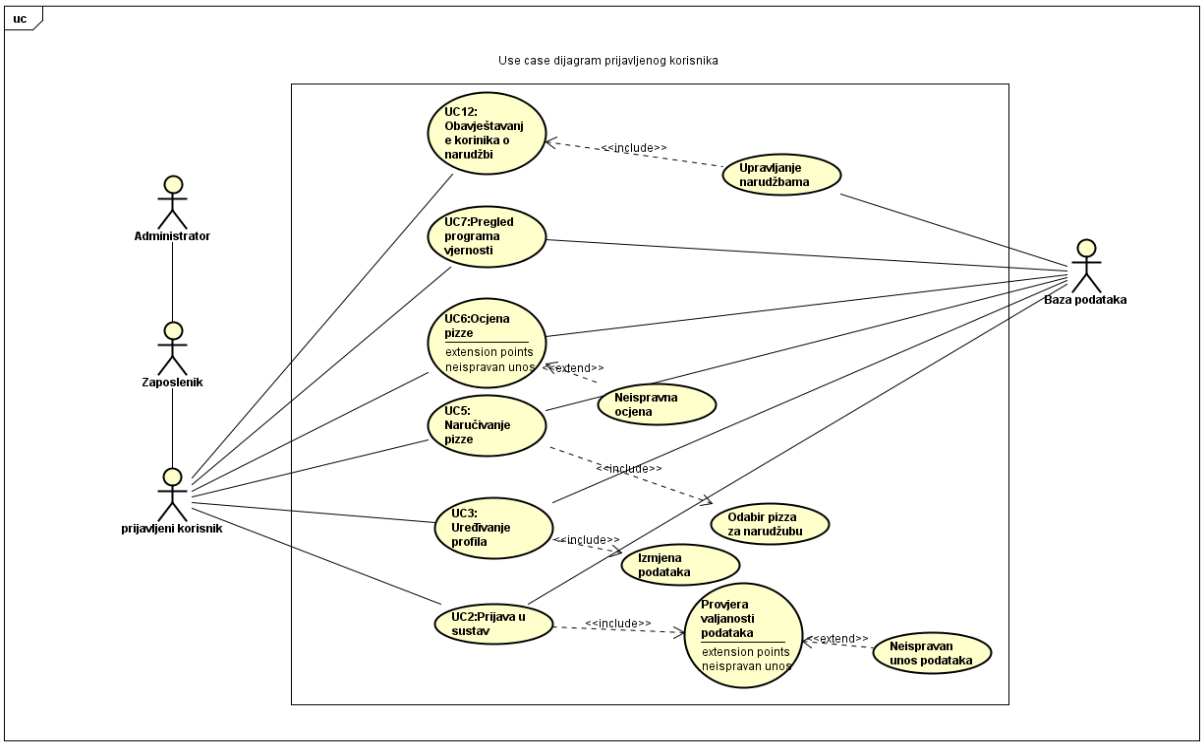
- **Glavni sudionik:** administrator
- **Cilj:** izmjena programa vjernosti
- **Sudionici:** Baza podataka
- **Preduvjeti:** korisnik je prijavljen u sustav kao administrator
- **Rezultat:** Novi uvjeti programa vjernosti
- **Željeni scenarij:**
    1. Klikne na pregled programa vjernosti
    2. Odabere opciju za uređivanje programa vjernosti
    3. Uredi podatke i spremi ih u bazu



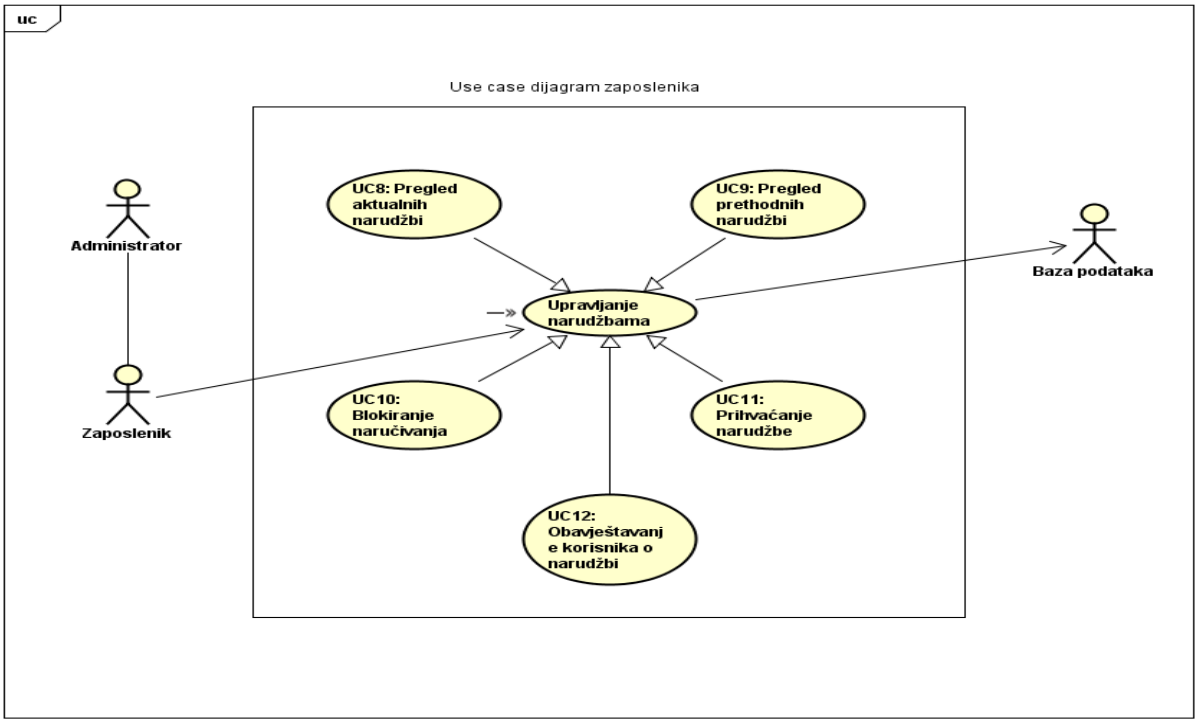
Slika 4.0.1. Dijagram obrasca uporabe, cjeloviti pregled, UC 1-UC 19



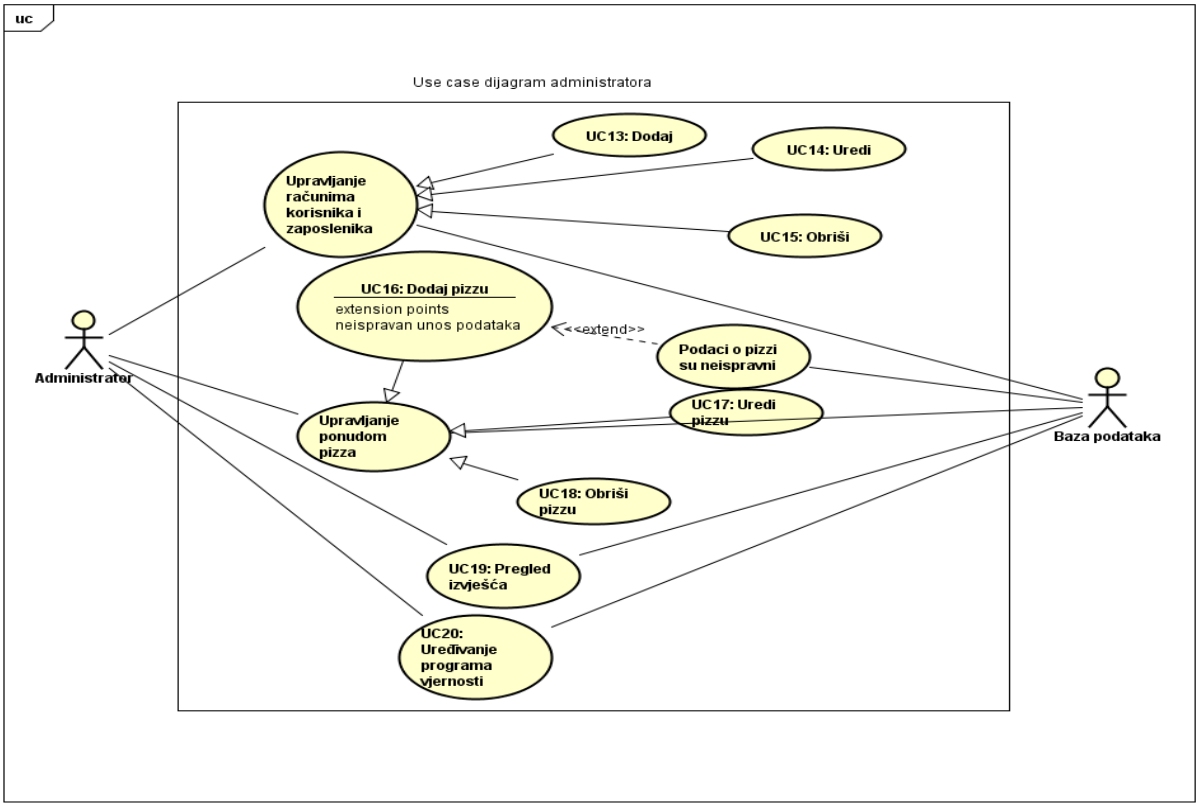
Slika 4.0.2. Dijagram obrasca uporabe, ponašanje neprijavljenog korisnika



Slika 4.0.3. Dijagram obrasca uporabe, ponašanje prijavljenog korisnika



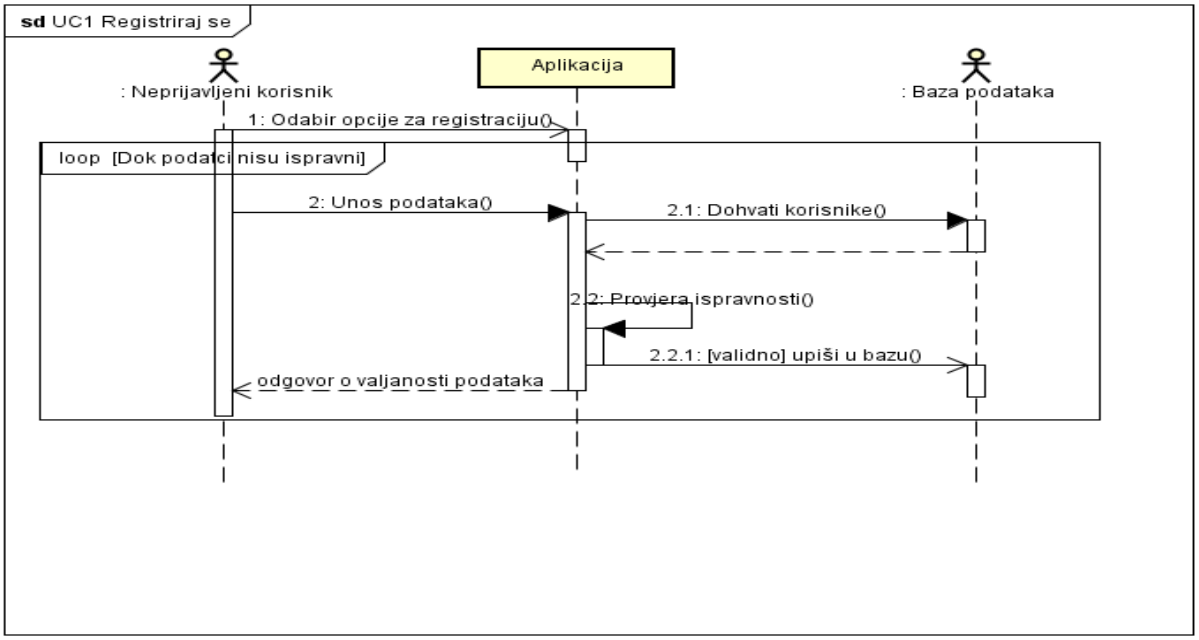
Slika 4.0.4. Dijagram obrasca uporabe, ponašanje zaposlenika



Slika 4.0.5. Dijagram obrasca uporabe, ponašanje administratora

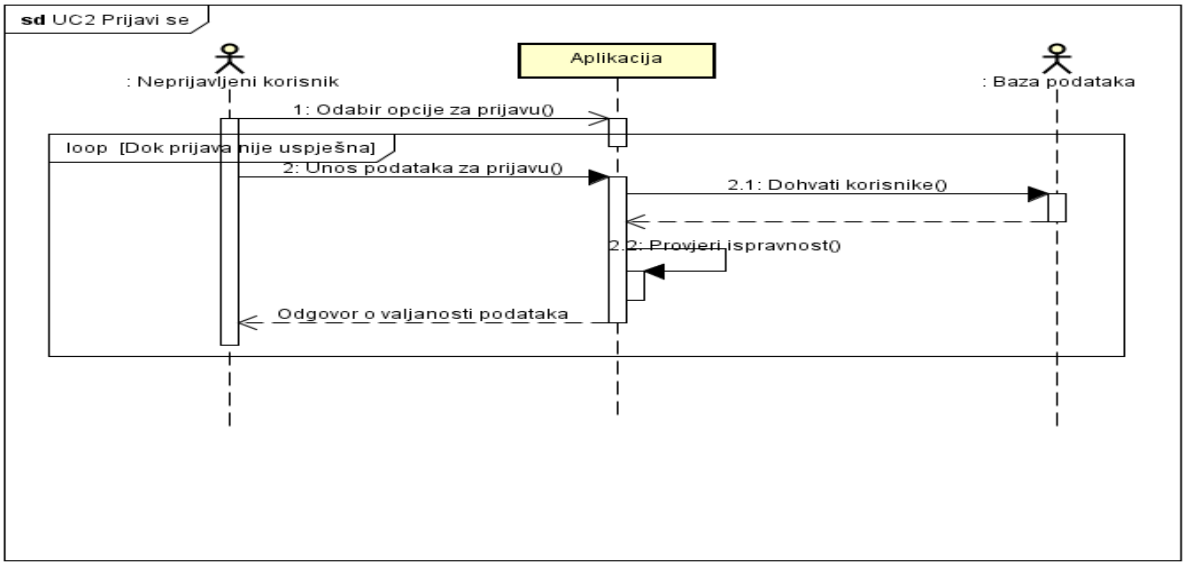
Sekvencijski dijagrami:

Obrazac uporabe UC1 (Registriraj se):
Neprijavljeni korisnik odabire opciju za registraciju te unosi svoje podatke u polja za registraciju. Sustav u komunikaciji s bazom podataka vidi postoji li takav korisnik u bazi ili su podatci neispravni te ako je registracija uspjela, upisuje ga u bazu. U protivnom, ponavlja se postupak registracije.



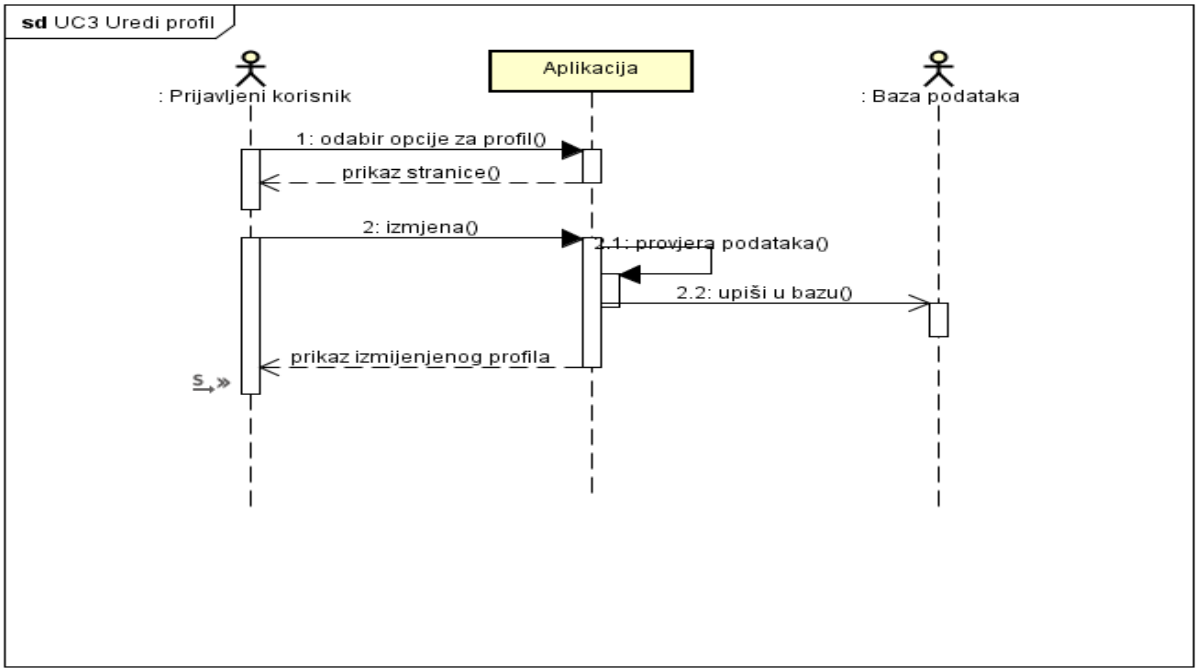
Slika 4.1.1. Sekvencijski dijagram 1 (Registriraj se)

Obrazac uporabe UC2 (Prijavi se):
Neprijavljeni korisnik može kliknuti opciju za prijavu gdje upisuje svoje podatke u polja za prijavu. Sustav je dužan u komunikaciji s bazom podataka provjeriti validnost unesenih podataka i korisnika prijaviti u sustav ili omogućiti mu ponovnu prijavu.



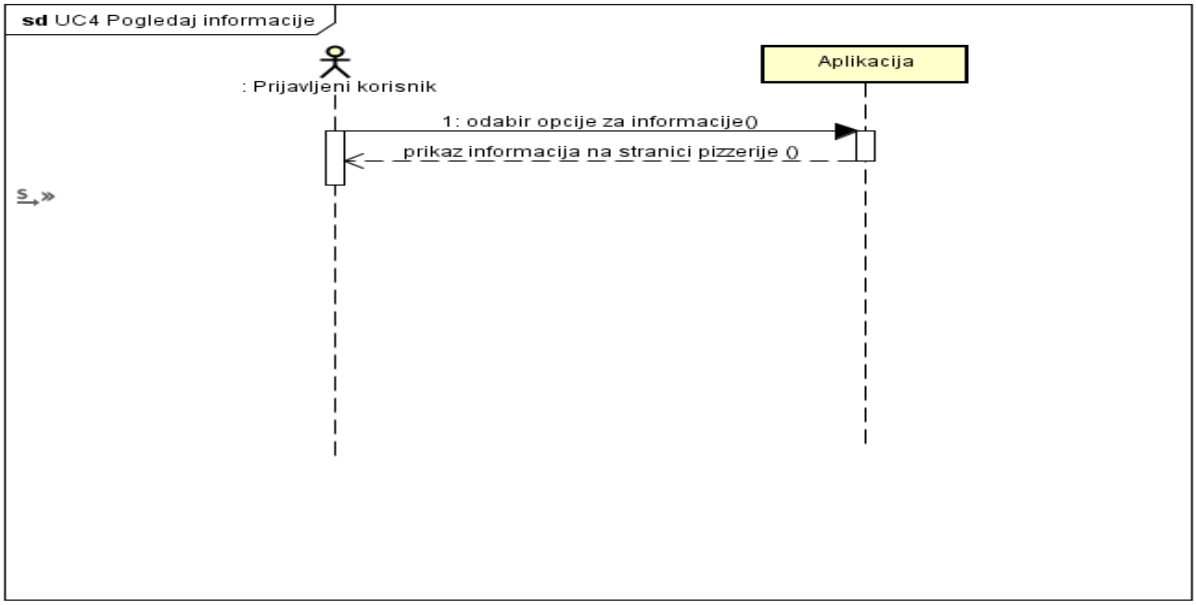
Slika 4.1.2. Sekvencijski dijagram 2 (Prijavi se)

Obrazac uporabe UC3 (Uredi profil):
Korisnik odabire opciju za uređivanje profila. Osim pregleda profila može i izmijeniti svoj profil nakon kojeg Sustav provjerava jesu li podatci ispravni te upisuje podatke u bazu.



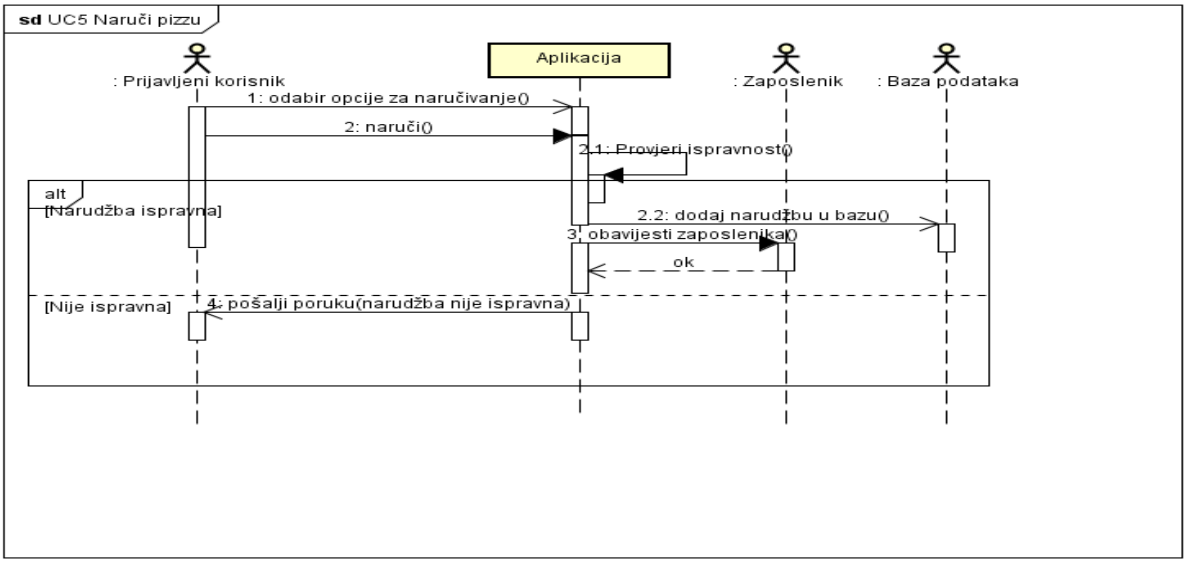
Slika 4.1.3. Sekvencijski dijagram 3 (Uredi profil)

Obrazac uporabe UC4 (Pogledaj informacije):
Prijavljeni i neprijavljeni korisnici mogu kliknuti gumb za pregledavanje informacija nakon čega se otvara stranica s informacijama o pizzeriji



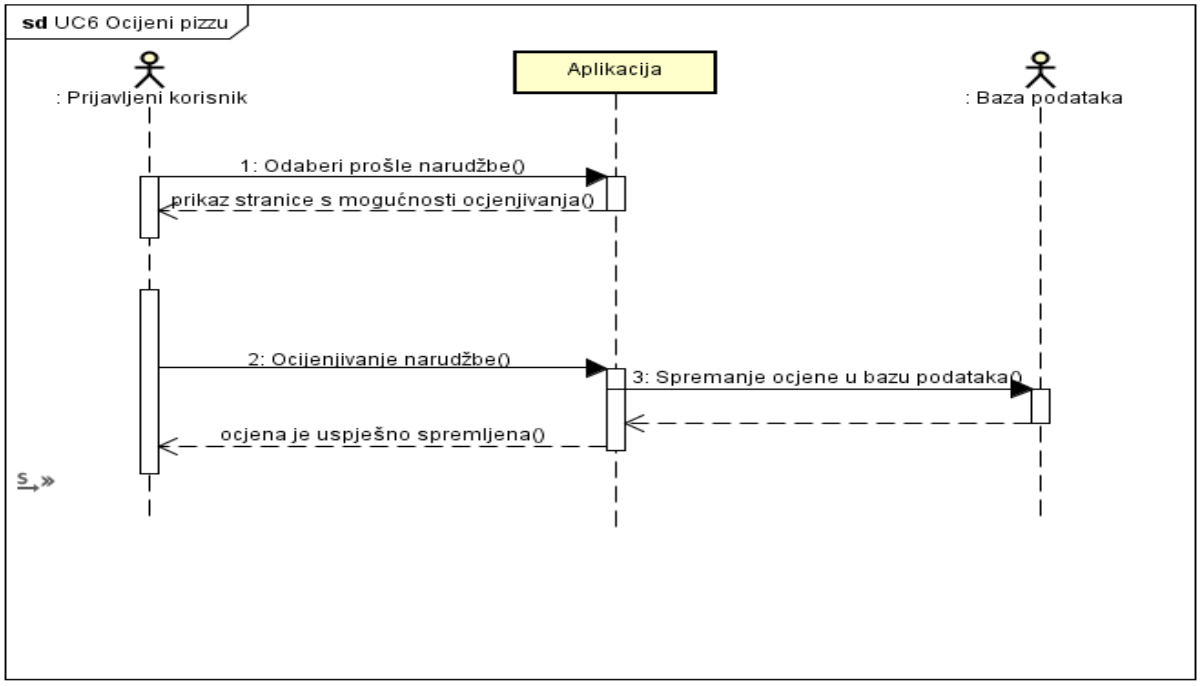
Slika 4.1.4. Sekvencijski dijagram 4 (Pogledaj informacije)

Obrazac uporabe UC5 (Naruči pizzu):
Korisnik odabire pizze koje želi naručiti iz ponude. Sustav provjerava ispravnost narudžbe te doda narudžbu u bazu te obavijesti zaposlenika o tome da mu je narudžba došla.



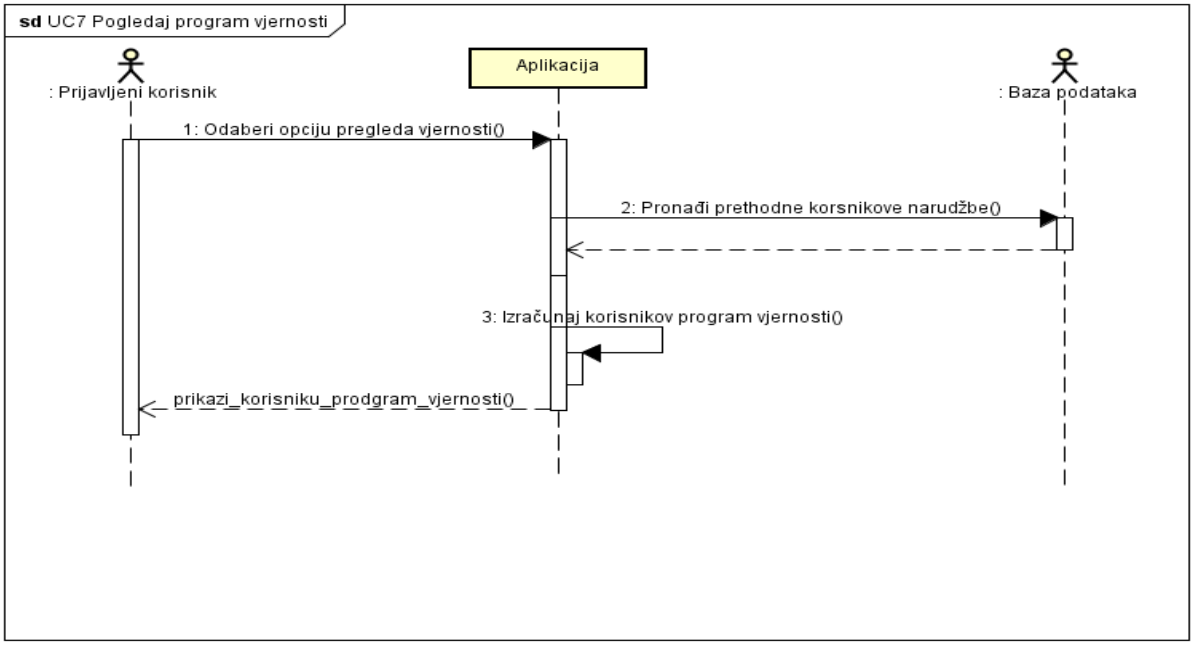
Slika 4.1.5. Sekvencijski dijagram 5 (Naruči pizzu)

Obrazac uporabe UC6 (Ocijeni pizzu):
Korisnik odabire opciju ocjenjivanja narudžbe. Sustav nudi korisniku da ocijeni svoju zadnju narudžbu u intervalu od 1-5 te nakon toga sustav sprema u bazu podataka ocjenu i zahvaljuje se korisniku na ocjeni.



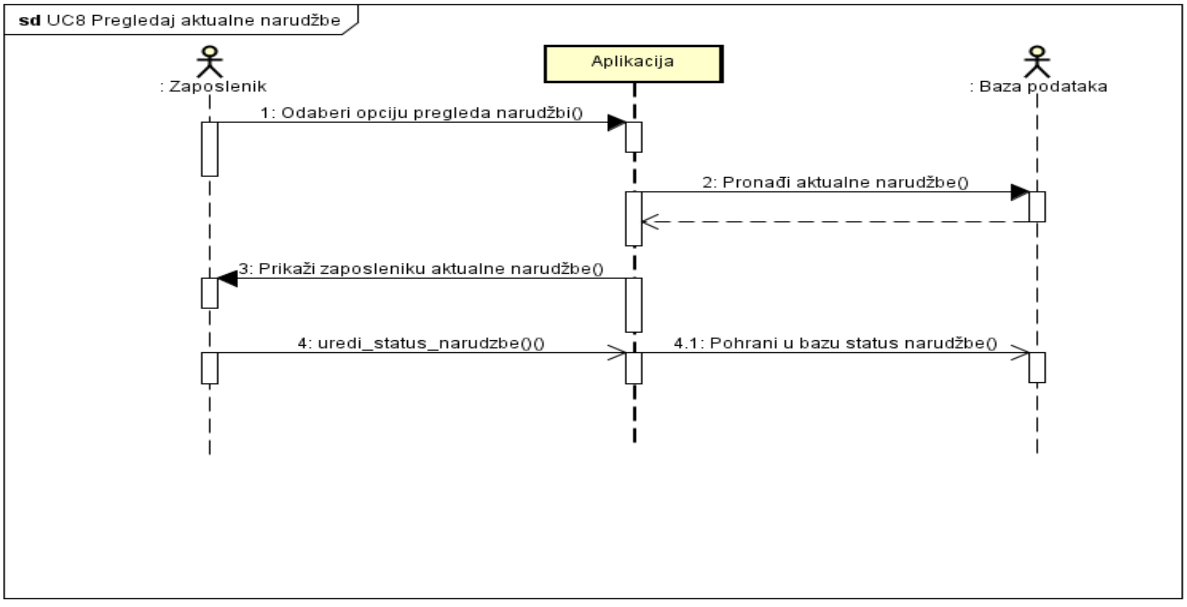
Slika 4.1.6. Sekvencijski dijagram 6 (Ocijeni pizzu)

Obrazac uporabe UC7 (Pregledaj program vjernost):
Korisnik odabire opciju pregleda programa vjernosti. Poslužitelj pronalazi u bazi podataka narudžbe koje je korisnik platio do tog trenutka te ih ispisuje. Korisnik ima uvid u broj narudžbi koje je do tada ostvario i upoznat je s brojem narudžbi koje treba ostvariti da bi dobio besplatnu pizzu.



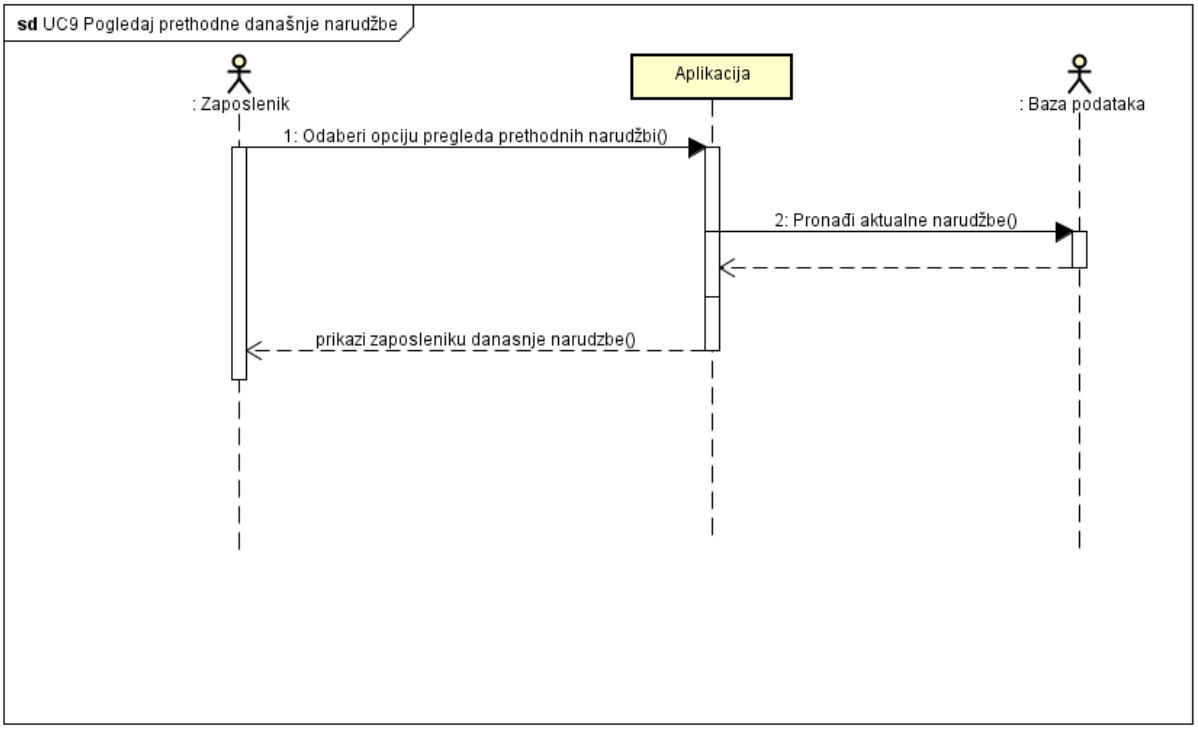
Slika 4.1.7. Sekvencijski dijagram 7 (Pregledaj program vjernosti)

Obrazac uporabe UC8 (Pregledaj aktualne narudžbe):
Zaposlenik na sučelju za zaposlenike odabire opciju za pregled aktualnih narudžbi. pregledava status i podatke o narudžbi koji su u bazi podataka temeljem kojih može dalje djelovati prema korisniku (obavijestiti ga o statusu narudžbe) te mijenja status narudžbe ovisno o stanju narudžbe.



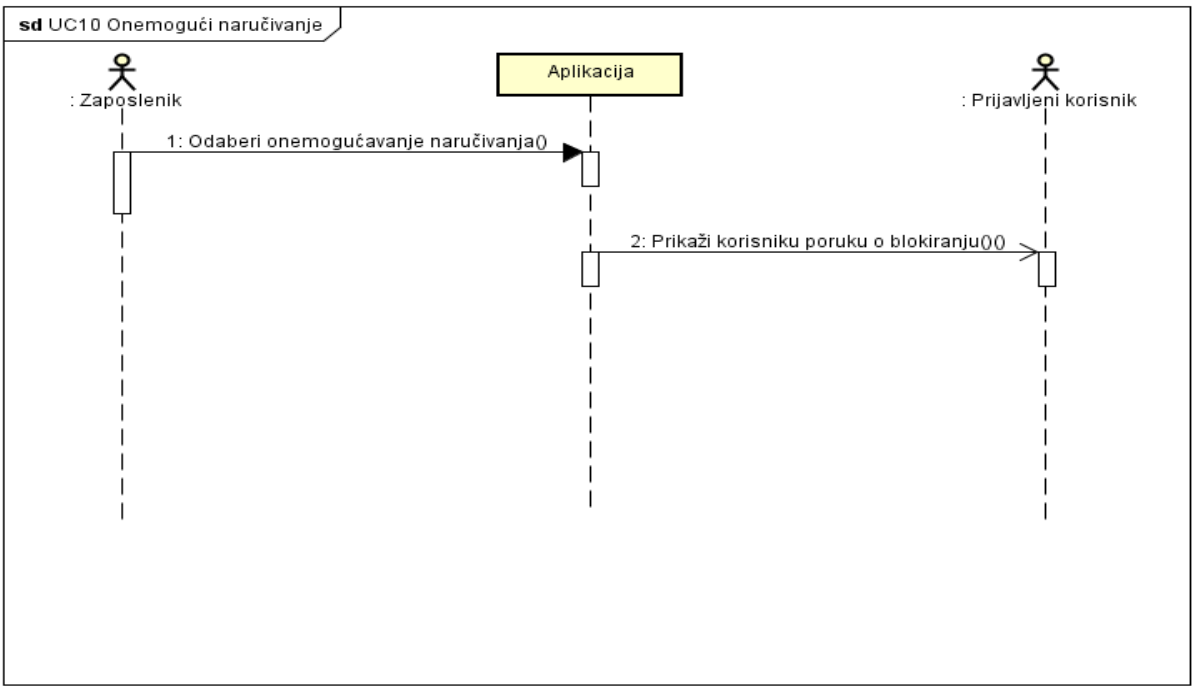
Slika 4.1.8. Sekvencijski dijagram 8 (Pregledaj aktualne narudžbe)

Obrazac uporabe UC9 (Pregledaj prethodne narudžbe):
Zaposlenik na sučelju za zaposlenike odabire opciju za pregled prethodnih narudžbi Pregledava status i podatke o narudžbi koji su u bazi podataka.



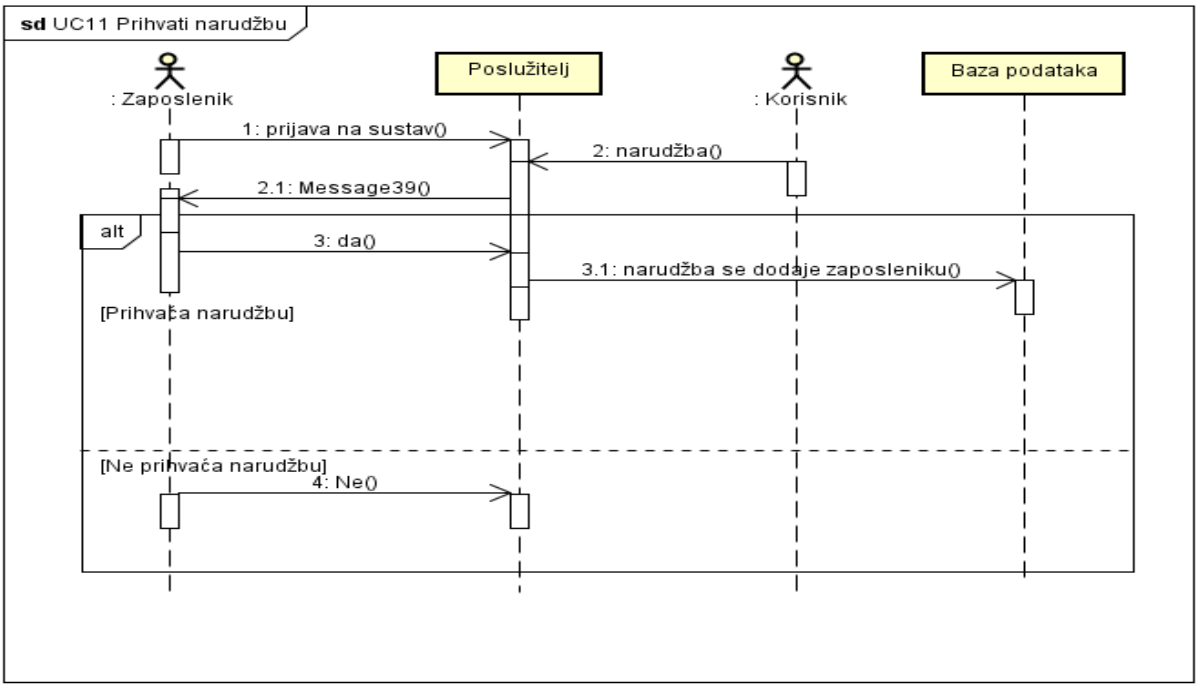
Slika 4.1.9. Sekvencijski dijagram 9 (Pregledaj prethodne narudžbe)

Obrazac uporabe UC10 (Onemogući naručivanje):
Zaposlenik odabire opciju za onemogućavanje naručivanja. Korisniku se prikazuje odgovarajuća poruka



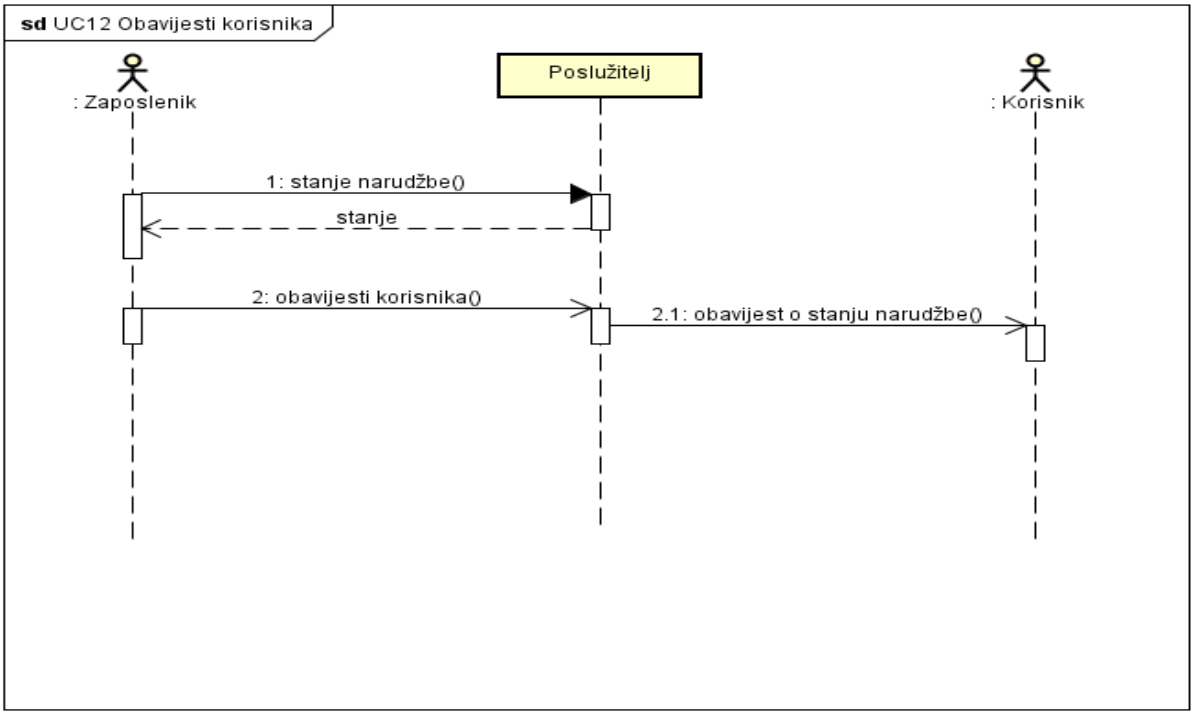
Slika 4.1.10. Sekvencijski dijagram 10 (Onemogući naručivanje)

Obrazac uporabe UC11 (Prihvati narudžbu):
Prijavljeni zaposlenik dobiva obavijest o novoj narudžbi te odlučuje hoće li obraditi.



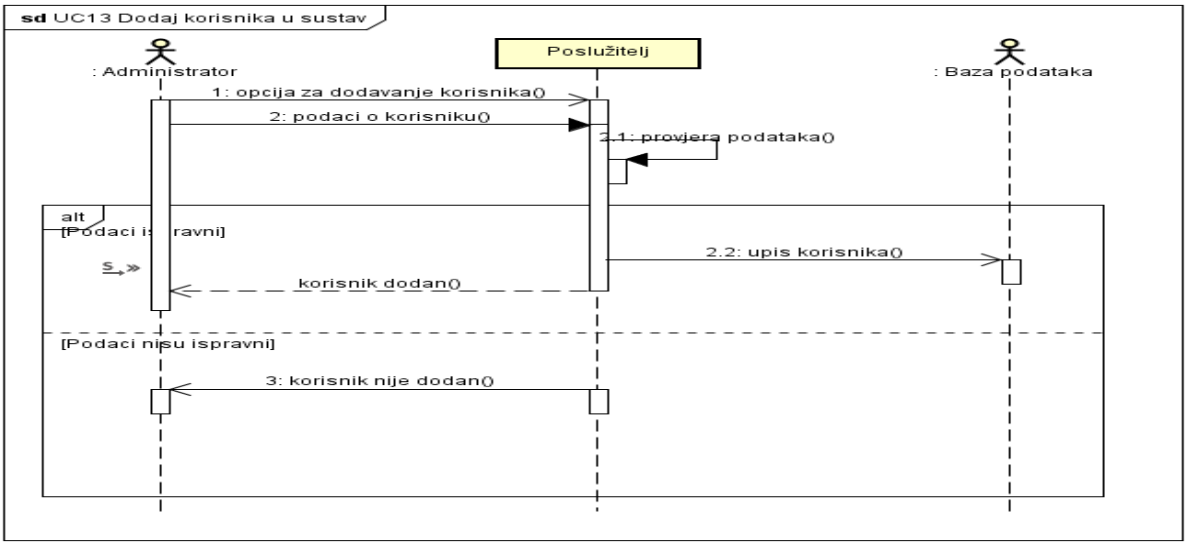
Slika 4.1.11. Sekvencijski dijagram 11 (Prihvati narudžbu)

Obrazac uporabe UC12 (Obavijesti korisnika):
Zaposlenik provjerava stanje narudžbe te obavještava korisnika o stanju narudžbe.



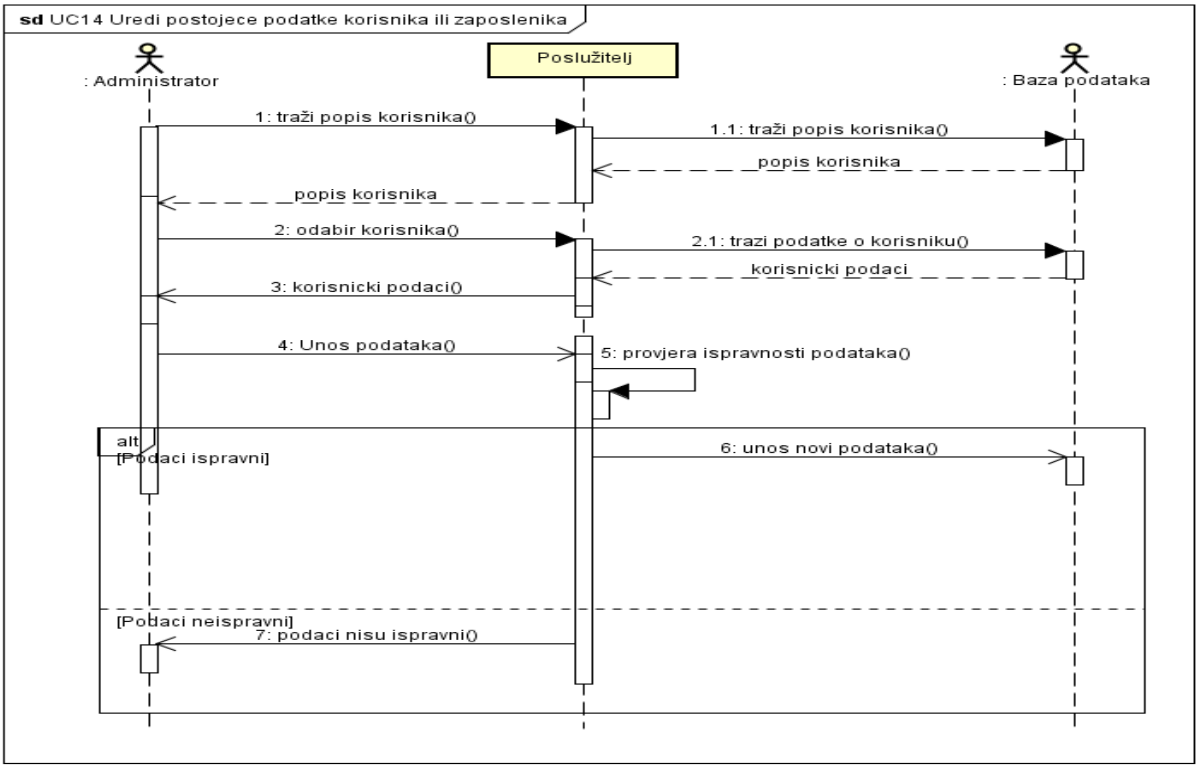
Slika 4.1.12. Sekvencijski dijagram 12 (Obavijesti korisnika)

Obrazac uporabe UC13 (Dodaj korisnika u sustav):
Administrator odabire opciju za dodavanje korisnika te unosi korisnikove podatke u sustav. Ako su podaci ispravni korisnik će biti dodan u sustav inače administrator dobiva poruku da podaci nisu ispravni.



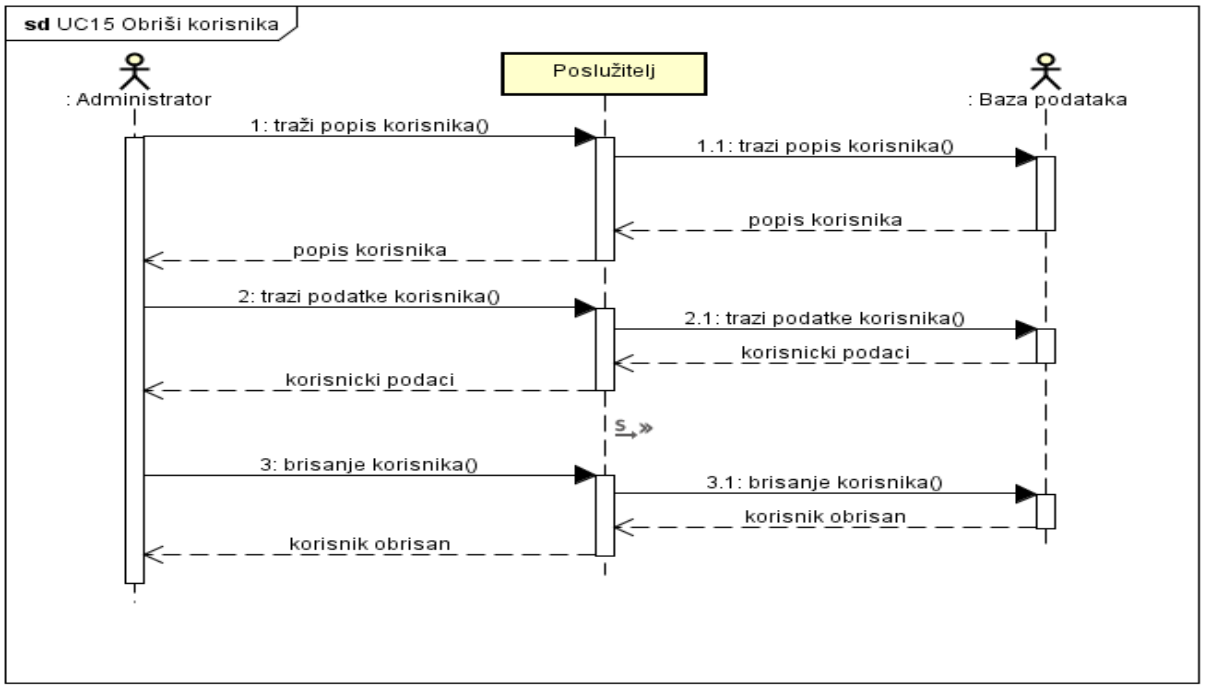
Slika 4.1.13. Sekvencijski dijagram 13 (Dodaj korisnika u sustav)

Obrazac uporabe UC14 (Uredi postojeće podatke korisnika ili zaposlenika):
Administrator traži od sustava popis korisnika iz popisa odabire korisnika kojem želi promijeniti podatke te unosi nove podatke. Ako su podaci ispravni sustav unosi nove podatke inače ispisuje poruku da podaci nisu ispravni.



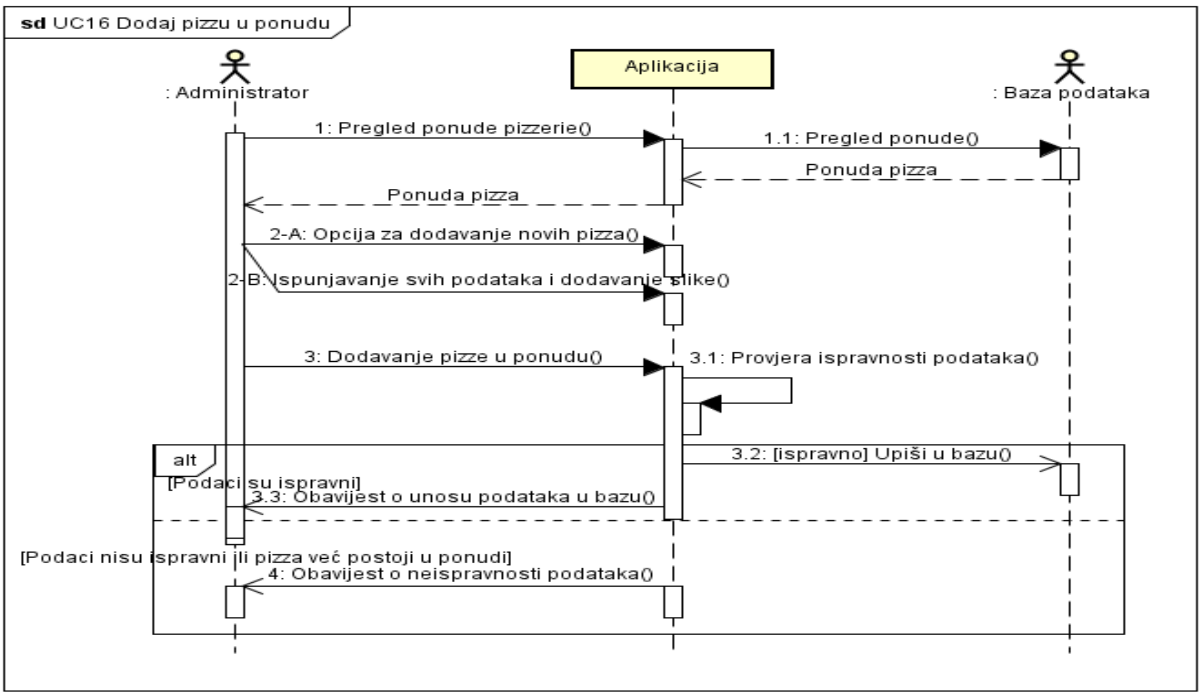
Slika 4.1.14. Sekvencijski dijagram 14 (Uredi postojeće podatke korisnika ili zaposlenika)

Obrazac uporabe UC15 (Obriši korisnika):
Administrator od sustava traži popis korisnika te odabire iz popisa kojeg će korisnika obrisati.



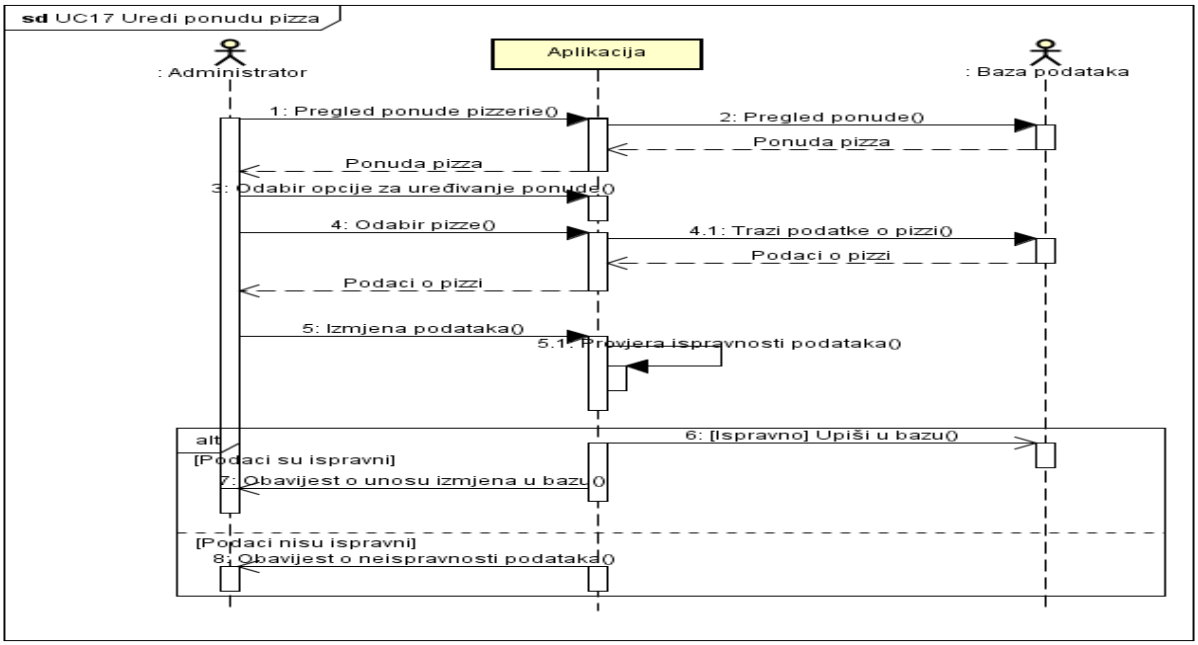
Slika 4.1.15. Sekvencijski dijagram 15 (Obriši korisnika)

Obrazac uporabe UC16 (Dodaj pizzu u ponudu):
Administrator pregledava ponudu pizza i dodaje novu pizzu u ponudu. Sustav provjerava ispravnost podataka te ako su ispravni upisuje podatke u bazu.



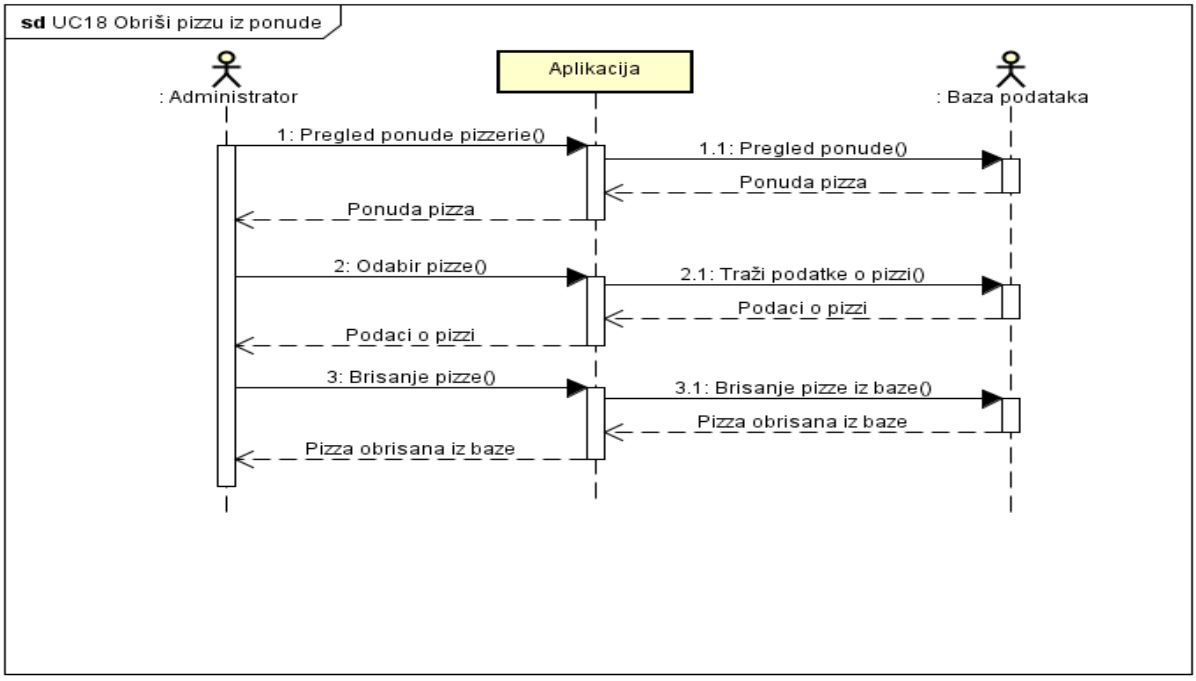
Slika 4.1.16. Sekvencijski dijagram 16 (Dodaj pizzu u ponudu)

Obrazac uporabe UC17 (Uredi ponudu pizza):
Administrator pregledava ponudu pizza te odabire opciju za uređivanje ponude. Zatim odabire pizzu te mijenja njene podatke. Sustav provjerava ispravnost podataka te unosi izmjene u bazu ako su podaci ispravni.



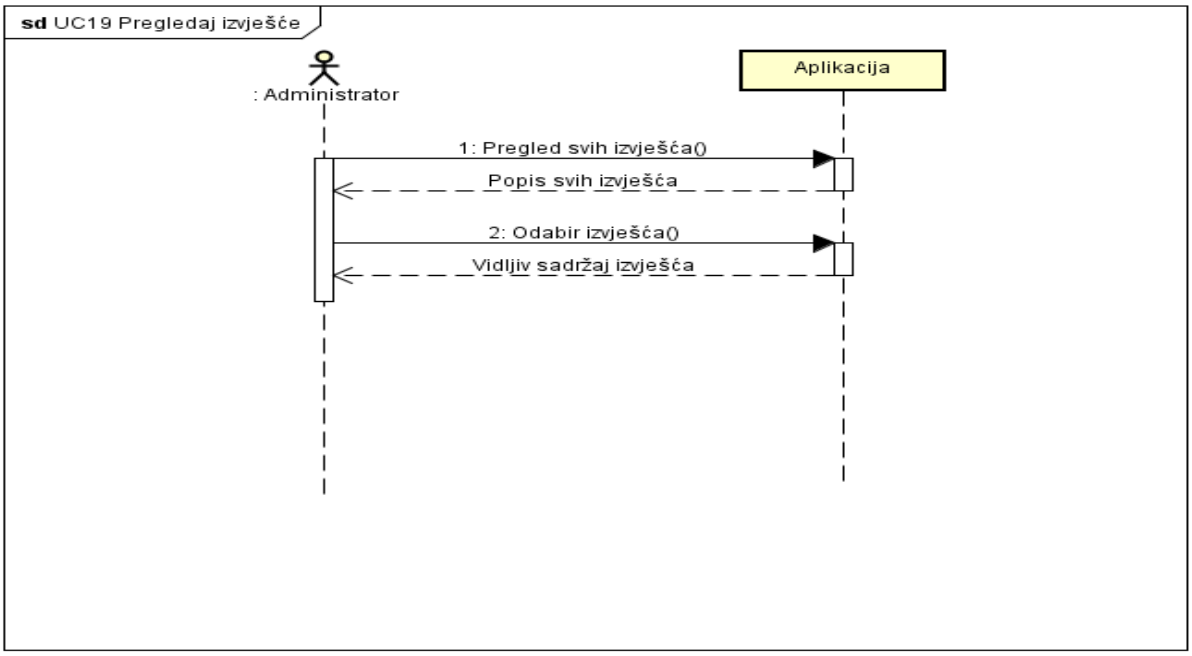
Slika 4.1.17. Sekvencijski dijagram 17 (Uredi ponudu pizza)

Obrazac uporabe UC18 (Obriši pizzu iz ponude):
Administrator pregledava ponudu pizza te odabire pizzu za brisanje. Sustav briše pizzu iz baze podataka te o tome obavještava administratora.



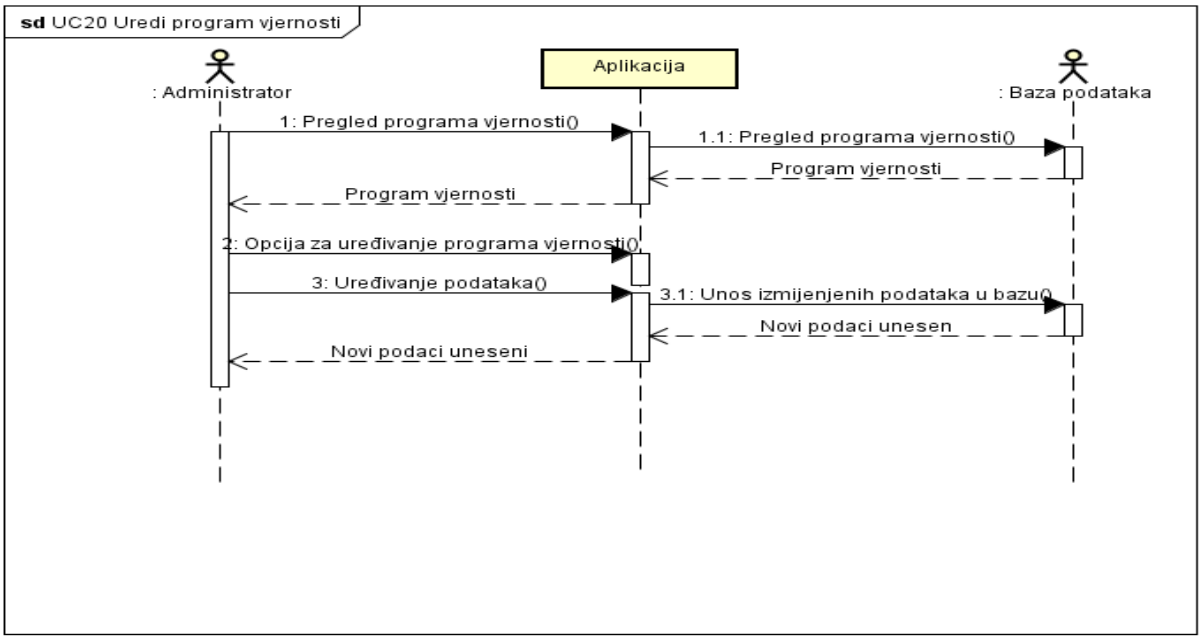
Slika 4.1.18. Sekvencijski dijagram 18 (Obriši pizzu iz ponude)

Obrazac uporabe UC19 (Pregledaj izvješće):
Administrator klikne na pregled svih izvješća te odabire pojedino izvješće čiji sadržaj mu se prikazuje.



Slika 4.1.19. Sekvencijski dijagram 19 (Pregledaj izvješće)

Obrazac uporabe UC20 (Uredi program vjernosti):
Administrator klikne na pregled postojećih programa vjernosti. Potom odabire opciju za uređivanje i unosi promjene u program vjernosti koje se zapisuju u bazi podataka.



Slika 4.1.20. Sekvencijski dijagram 20 (Uredi program vjernosti)

5. Ostali zahtjevi

- Sustav mora podržavati korištenje od strane više korisnika u stvarnom vremenu
- Aplikacija mora podržavati hrvatske dijakritike
- Izvršavanje programa s uključenim pristupom bazi podataka za svaki pogled ne smije trajati duže od nekoliko sekundi
- Administrator sustava ne smije vidjeti trenutnu lozinku korisnika
- Korisnički podaci pojedinog korisnika ne smiju biti vidljivi ostalim korisnicima
- Unutar 3 minute od trenutka narudžbe zaposlenik je dužan obavijestiti naručitelja o vremenu dolaska narudžbe
- Vođenje evidencije o broju naručenih pizza i ocjeni
- Osigurati da prije registracije korisnik mora unijeti ime, prezime, e-mail, lozinku i broj telefona

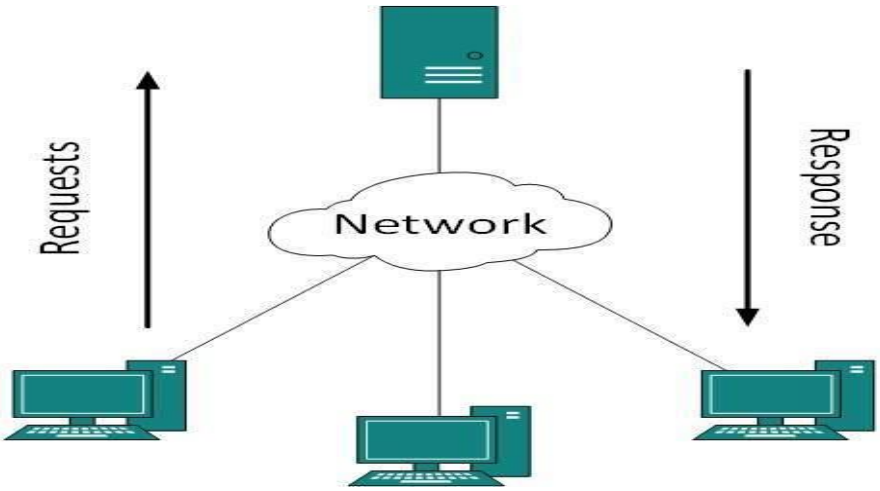
6. Arhitektura i dizajn sustava

6.1. Svrha, opći prioriteti i skica sustava

Prilikom izrade plana za razvoj projekta, važna odluka je bila odabir programskih jezika u kojima ćemo razvijati aplikaciju. Kako imamo u planu izradu web aplikaciju koja će koristiti bazu podataka, odabir je pao na korištenje Klijent – Server modela aplikacije koji će međusobno komunicirati putem REST api-a.

Klijent (frontend) i server(backend) se ponašaju kao potpuno odvojene aplikacije te rad jednog nema veze s drugim. Server ne pohranjuje nikakve podatke o klijentu, već samo odgovara na upite koje mu pošalje klijent.

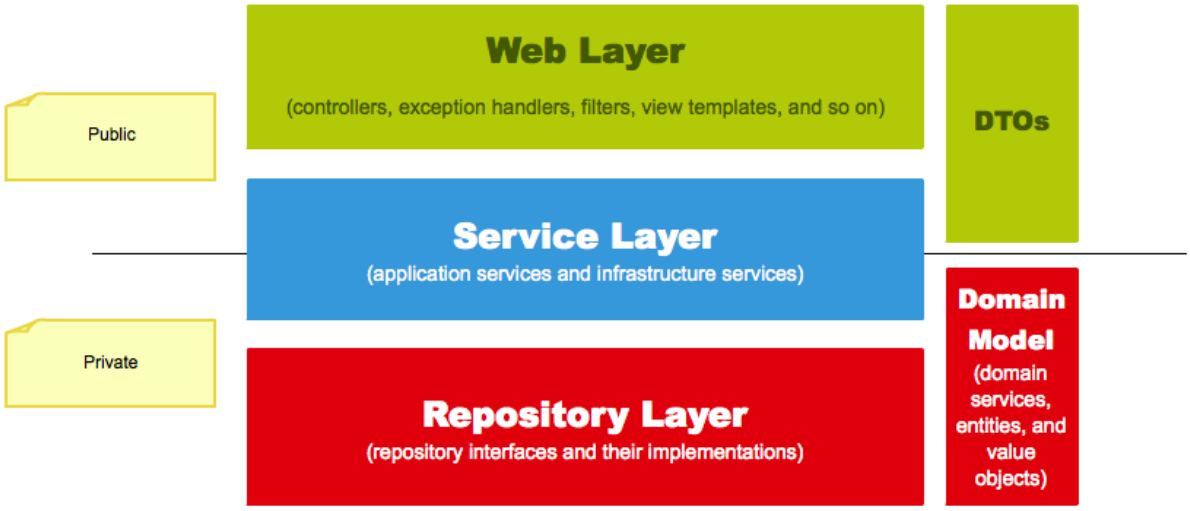
Sukladno tome, server se isto ponaša bez obzira odakle mu došao upit, bilo to sa smartphone-a, PC, bez obzira u kojem programskog jeziku pisan klijent itd. Ista stvar vrijedi za klijenta.



Slika 6.1.1. Klijent – Server model

Backend dio aplikacije

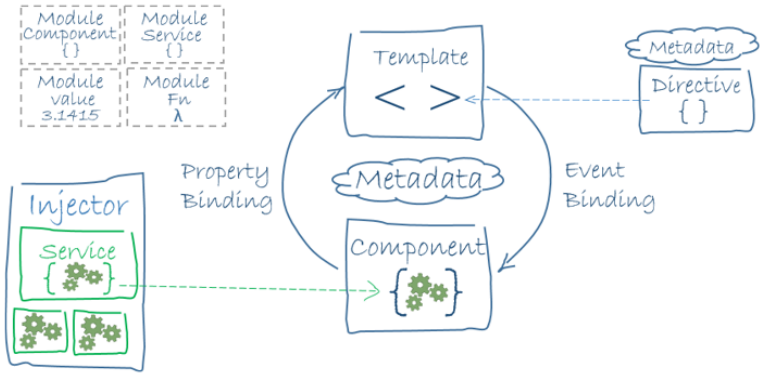
Backend dio aplikacije će biti realiziran uporabom programskog jezika Java i Spring frameworka za Javu. Spring framework aplikaciju dijeli na tri dijela, web dio, servise(core/business) logic dio te dio za pristup bazi podataka. Prema vanjskom svijetu je otvoren samo web dio, vanjska aplikacija koja će koristiti našu aplikaciju neće moći direktno pristupiti bazi podataka već će njen upit morati proći kroz sva tri sloja aplikacije. Spring framework nam omogućuje jednostavno rješavanje povezivanja s bazom podataka te jednostavno uspostavljanje REST api-a. Uz sve to, versatilnost programskog jezika Jave nam omogućuje da lako izvedemo bilo kakav logički problem koji se može pojaviti prilikom izrade sustava.



Slika 6.1.2. Spring arhitektura

Frontend dio aplikacije

Za frontend dio aplikacije koristit ćemo Angular tehnologiju. Angular tehnologija omogućava jednostavan razvoj web sustava različite veličine. Angular tehnologija omogućava razbijanje frontend aplikacije u komponente koje se kasnije mogu ponovno iskoristiti. Komponenta predstavlja bilo što, od liste, čitave stranice, forme za unos, itd. Logika komponente je izvedena uporabom TypeScript programskog jezika koji se prije izvođenja u browser-u prevodi u jezik JavaScript koji je može izvoditi u svim modernim browserima. Prikaz komponente je realiziran uporabom HTML jezika, a izgled komponente je realiziran uporabom CSS jezika. Također, veoma je lagano uspostaviti komunikaciju s backend-om putem REST api-a. Za povezivanje s bazom, spremanje bilo kakvih podataka vezanih za rad same aplikacije se sprema u Angular servise koji su izvedeni uporabom TypeScript programskog jezika. Angular za mnogo stvari ima gotova rješenja koja su modularizirana te se mogu skinuti s interneta. Za dizajn ćemo iskoristiti Bootstrap modul koji sadrži niz već gotovih dizajniranih komponenti koje izgledaju veoma lijepo i oku ugodno.



Slika 6.1.3. Angular arhitektura

Baza podataka

Za potrebe našeg sustava koristit ćemo SQL relacijsku bazu podataka. Relacijska baza podataka nam najviše odgovara jer nam olakšava modeliranje događaja i entiteta iz stvarnog svijeta. Sve relacije u bazi su svedene na 3. normalnu formu tako da nemamo redundantnih podataka. Prilikom izrade baze podataka služili smo se PostreSQL-om.

U nastavku je prikazan ER model baze podataka po kojem će biti napravljena implementacija u Spring frameworku. Umjesto nasljeđivanja razreda kako će biti opisano u dijagramu razreda, dodan je entitet Role koji se slaže sa Spring MVC i Spring Security protokolima za lakšu izradu aplikacija.

Relacija **User** definira korisnika sustava za upravljanje radom pizzerije, a to može biti *kupac*, *zaposlenik* ili *administrator*. Ono što će ih razlikovati bit će uloga koja će im biti dodijeljena a to je modelirano entitetom **Role.** Moramo napomenuti da 1 korisnik može imati samo jednu ulogu. Svaka uloga sadrži svoje *ime*, dok User ima korisničko *ime, lozinku, ime, prezime, spol, adresu i broj telefona.*

Svi parametri osim spola kod relacije **User** moraju biti popunjeni za ubacivanje korisnika u bazu zbog sigurnosnih razloga (npr. ako korisnik ne želi platiti pizzu).

User je povezan sa svojim **LoyaltyProgram**-om vezom 1 na 1, te se u toj relaciji specificira koliko je još pizza ostalo kupcu da ostvari popust (*number_of_pizzas_until_free_one*).

Relacija **Order** sadrži *datum narudžbe, ID kupca, ID zaposlenika*, te je povezana relacijom many-to-many s pizzama koje kupac naruči. Jedna narudžba može sadržavati više **Pizza**-a dok jedna pizza može biti odabrana u više narudžbi.

Relacija **Pizza** je povezana many-to-many vezom s relacijom **Ingredient** koja predstavlja sastojak. Sastojak može biti sadržan u više pizza, dok jedna pizza sadrži niz sastojaka.

Pizza može biti ocijenjena nekom ocjenom, ali u svojoj relaciji ne može imati prosječnu ocjenu jer se ne bi ona onda mogla ažurirati. Zbog toga je uvedena nova relacija **Grade**, koja povezuje ID kupca, ID **Pizza**-e te ocjenu koju je kupac dodijelio pizzi. Na temelju toga se osnovnim SQL upitom može dobiti prosječna ocjena **Pizza**-e, a također je prednost što će sve ocjene biti pohranjene.

Na kraju, relacija **Report** služi za zapisivanje statističkih podataka o radu pizzerije te je povezana s relacijom **ReportTimeSpan** koja ima u sebi cikluse tijekom kojih se mogu izvještaji stvarati. Ciklusi koji će biti dopušteni bit će: *WEEK, MONTH, QUARTER-YEAR, HALF-YEAR* te *YEAR.*

Svi atributi relacija detaljno su opisani u tablicama, a također je dan i ER dijagram baze.

User

User_id	Primarni ključ za korisnika	INT	PK
Username	Korisničko ime korisnika, alternativni ključ budući da je jedinstven	VARCHAR(255)	AK
Email	E-mail korisnika	VARCHAR(255)	
Password	Korisnikova lozinka (hash)	VARCHAR(255)	
First_name	Ime korisnika	VARCHAR(255)	
Last_name	Prezime korisnika	VARCHAR(255)	
Phone_number	Broj korisnika	VARCHAR(255)	
Gender	Spol	CHAR	
Address	Adresa korisnika	VARCHAR(255)	
Role_id	Uloga	VARCHAR(25)	FK

Role

Role_id	Primarni ključ za ulogu	INT	PK
Name	Ime uloge, može biti alternativni ključ jer će biti svaka različita	VARCHAR(255)	AK

LoyaltyProgram

Loyalty_program_id	Primarni ključ za program vjernosti	INT	PK
Number_of_pizzas_until_free_one	Koliko je pizza ostalo korisniku do popusta	INT	
User_id	ID kupca	INT	FK

Order

Order_id	Primarni ključ za narudžbu	INT	PK
Date_created_at	Datum izvršavanja narudžbe	DATE	
User_id	ID kupca	INT	FK
Employee_id	ID zaposlenika	INT	FK
Order_status_id	ID statusa narudžbe	VARCHAR(255)	FK

OrderStatus

Order_status_id	Primarni ključ za status narudžbe	INT	PK
Name	Naziv statusa narudžbe, jedinstven	VARCHAR(255)	AK

Order_Pizza

Order_id	ID narudžbe	INT	FK
Pizza_id	ID pizze	INT	FK

Pizza

Pizza_id	Primarni ključ za pizzu	INT	PK
Name	Ime pizze	VARCHAR(255)	
Price	Cijena pizze	NUMERIC(3, 2)	
Description	Opis pizze	VARCHAR(255)	
Picture_url	Put do datoteke sa slikom pizze	VARCHAR(255)	

Pizza_Ingredient

Pizza_id	ID pizze	INT	PK
Ingredient_id	ID sastojka	INT	PK

Ingredient

Ingredient_id	Primarni ključ za sastojak	INT	PK
Name	Naziv sastojka	VARCHAR(255)	

Grade

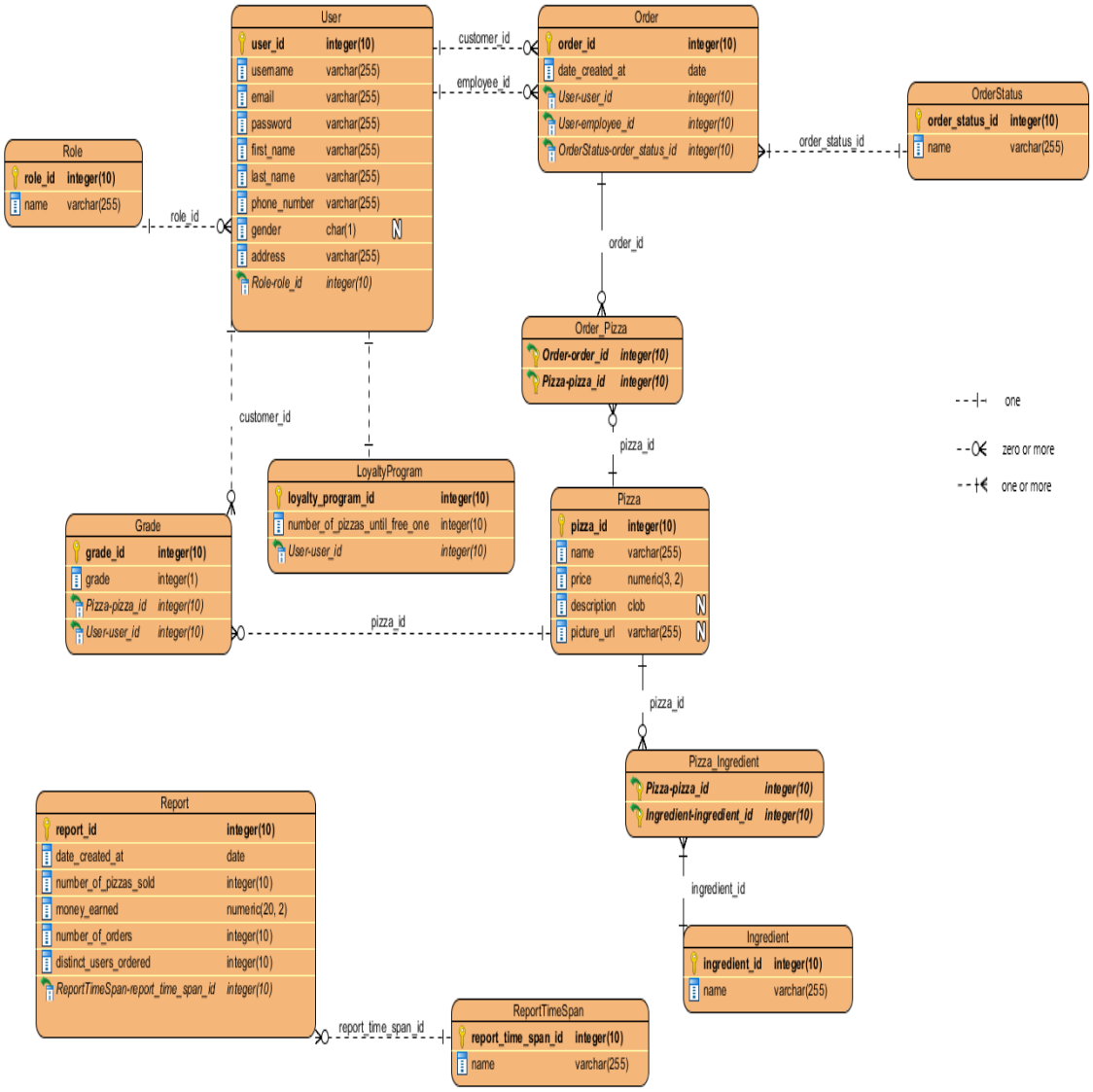
Grade_id	Primarni ključ za ocjenu	INT	PK
Grade	Ocjena	INT (CHECK CONSTRAINT BETWEEN 1, 5)	
Pizza_id	ID pizze	INT	FK
User_id	ID kupca	INT	FK

Report

Report_id	Primarni ključ za izvještaj	INT	PK
Date_created_at	Datum stvaranja izvještaja	DATE	
Number_of_pizzas_sold	Broj prodanih pizza	INT	
Money_earned	Zarađeni novac	NUMERIC(20, 2)	
Number_of_orders	Broj narudžbi	INT	
Distinct_users_ordered	Broj različitih korisnika koji su naručili	INT	
Report_time_span_id	ID ciklusa	INT	FK

ReportTimeSpan

Report_time_span_id	Primarni ključ za ciklus	INT	PK
Name	Naziv ciklusa, jedinstven	VARCHAR(255)	AK



Slika 6.1.4. ER dijagram baze podataka

6.2. Dijagram razreda s opisom

U ovom poglavlju sadržan je konceptualni dijagram razreda sa svim potrebnim klasama za rad sustava za upravljanje pizzerijom. Naglasak na tome je da je postoji i konceptualni i stvarni dijagram razreda te konceptualni služi više kao predodžba i uvid u to kako su stvari logički organizirane, dok se promjene mogu očekivati budući da je implementacija samog rješenja bila rađena u Spring frameworku. Tako možemo uočiti neke razlike između ER modela i samog dijagrama razreda, dok je potpuna kompatibilnost ER modela i dijagrama razreda bila nadopunjena konačnim dijagramom razreda nakon same implementacije.
Konceptualna ideja bazirana je na funkcionalnom aspektu sustava te smo prema tome konstruirali dijagram razreda. Svi atributi opremljeni su getterima i setterima dok su parametri settera izostavljeni s pretpostavkom da se podrazumijevaju.

Ključni aktori modelirani su razredom ***Person***, koji je apstraktan te predstavlja osobu definiranu sljedećim podacima: *korisničko ime*, *email adresa*, *lozinka*, *ime*, *prezime*, *broj telefona*, *spol* i *adresa*. On se može granati na sljedeće podrazrede:

- **Customer** – predstavlja korisnika (kupca) koji je registriran te ima sve potrebne metode za naručivanje pizze, pogled na program vjernosti za dobivanje besplatnih pizza, te uvid u svoje izvršene narudžbe
- **Employee** – predstavlja zaposlenika u pizzeriji koji može vidjeti svoje trenutne i izvršene narudžbe. Također, može potvrditi narudžbu, ali i blokirati ako je red čekanja za narudžbe velik
- **Administrator** – osoba koja nadgleda web stranicu te po potrebi može upravljati pizzama te korisnicima. Nude mu se mogućnosti poput dodavanja, uređivanja i brisanja korisnika, zaposlenika, pizza i programa vjernosti

Program vjernosti modeliran je razredom **CustomerLoyaltyProgram** koji ima statičku varijablu koja predstavlja definiranu vrijednost broja pizza koju korisnik treba naručiti da bi dobio gratis pizze (*numberOfPizzasUntilFreeOne*). Sa svakom naručenom pizzom bi se tako polje *pizzasLeft* trebalo smanjiti za jednu jedinicu čime bi se korisnik približio traženom popustu.

Svaki **Customer** ima metodu *orderPizza()* s kojom počinje njegovo odabiranje koje pizze želi staviti u svoju narudžbu.
To je uvelo potrebu za razredom **Order** koji tako ima polja *dateCreatedAt* koji predstavlja datum narudžbe, referencu na **Customer**-a, **Employee**-a kako bi se znalo tko je naručio a tko primio narudžbu, te listu samih naručenih pizza.

Da bi se pratilo stanje narudžbe, poslužit će nam enumeracija **OrderStatus** u kojem imamo sljedeća stanja: *ORDERED* (kupac je naručio), *ACCEPTED* (zaposlenik je prihvatio narudžbu), *DELIVERING* (narudžba je u dostavi), *PAID* (narudžba je plaćena), NOT-*PAID* (narudžba nije plaćena).

Razred **Pizza** ima polja *name, price, description, pictureUrl* (put do slike), *pictureFile* (sliku), te listu sastojaka od koje se ona sastoji.

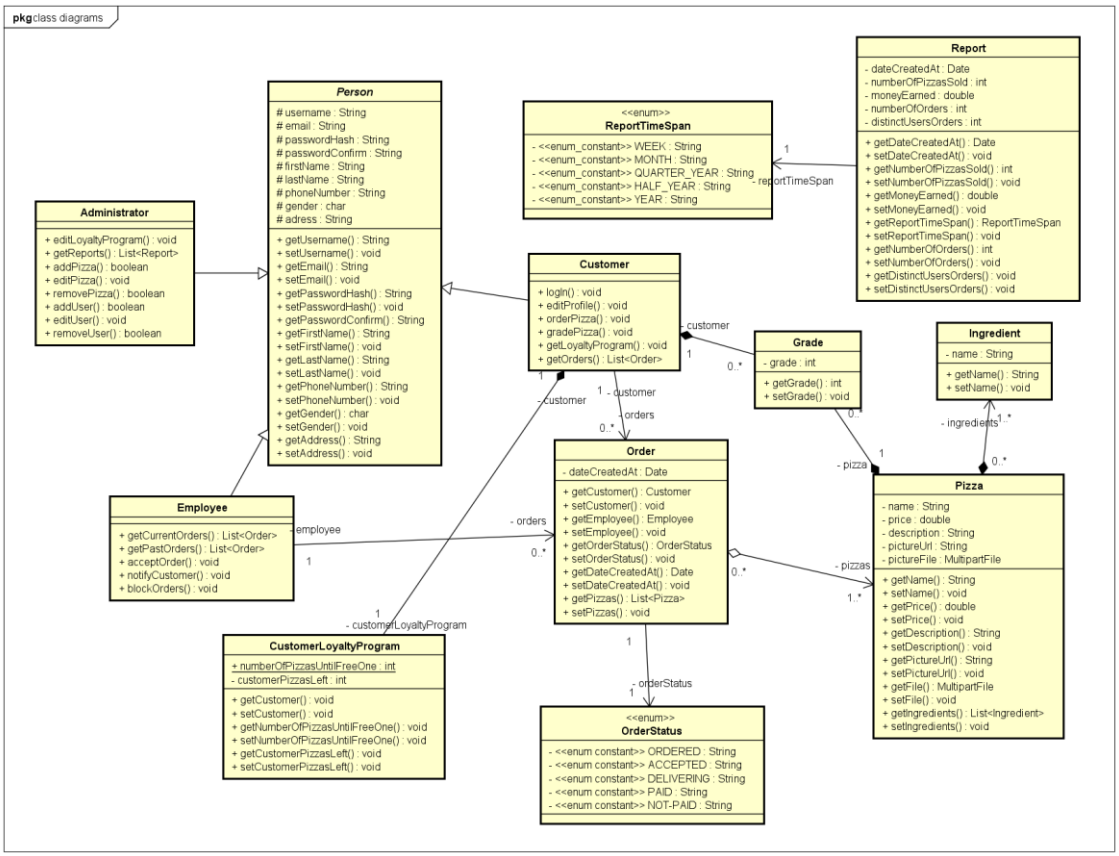
Sastojci su modelirani razredom **Ingredient** s poljem *name* kao ime sastojka.

Pojedina ocjena **Pizza**-e sadržana je u razredu **Grade** koje ima ocjenu te referencu na **Pizza**-u i **Customer**-a koji je ocijenio tu istu pizzu.

Na kraju svakog određenog ciklusa, stvara se **Report** koji predstavlja statistiku rada pizzerije tijekom tog ciklusa. **Report** će biti stvoren svaki put nakon što je otkucao ciklus opisan u enumeraciji **ReportTimeSpan** (*WEEK* – tjedno, *MONTH* – mjesečno, *QUARTER_YEAR* – kvartarno, *HALF-YEAR* – polugodišnje, *YEAR* – godišnje).

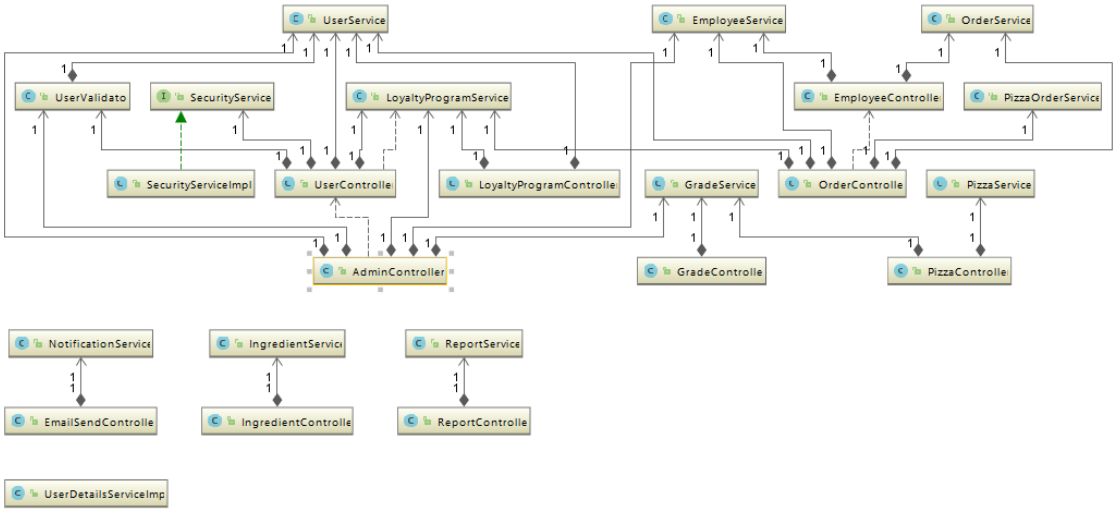
Osim ciklusa koji obilježava u vremenski raspon u kojem je stvoren, **Report** ima i polje *dateCreatedAt,* zatim broj prodanih pizza u tom ciklusu (*numberOfPizzasSold*), zarađeni novac (*moneyEarned*), broj narudžbi (*numberOfOrders*), te broj **Customer**-a koji su naručili pizzu (*distincUsersOders*).

U nastavku je prikazana i slika konceptualnog dijagrama razreda dok će potpuni dijagram razreda biti stvoren naknadno nakon implementacije.

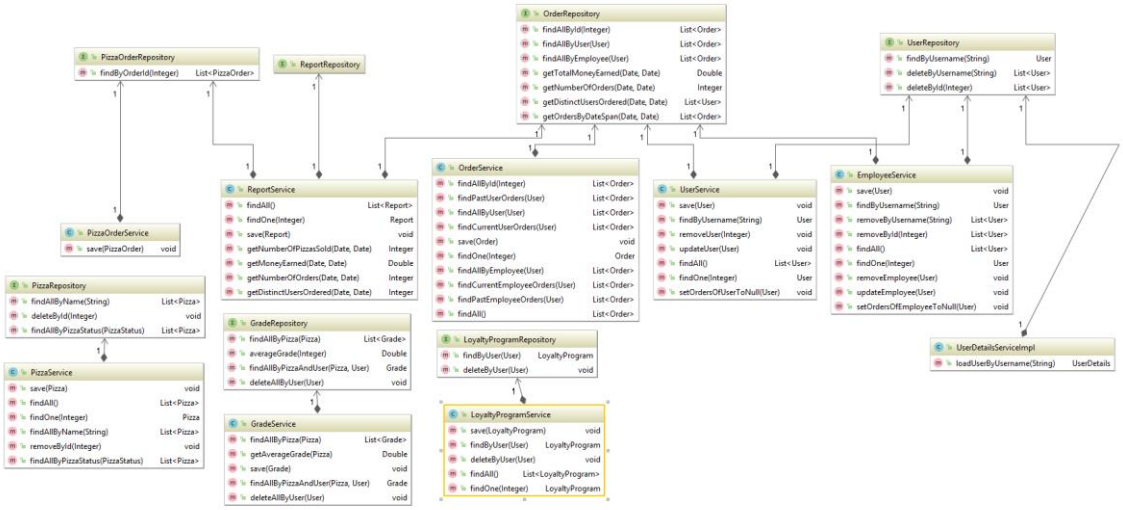


Slika 6.2.1. Konceptualni dijagram razreda za sustav za upravljanje radom pizzerije

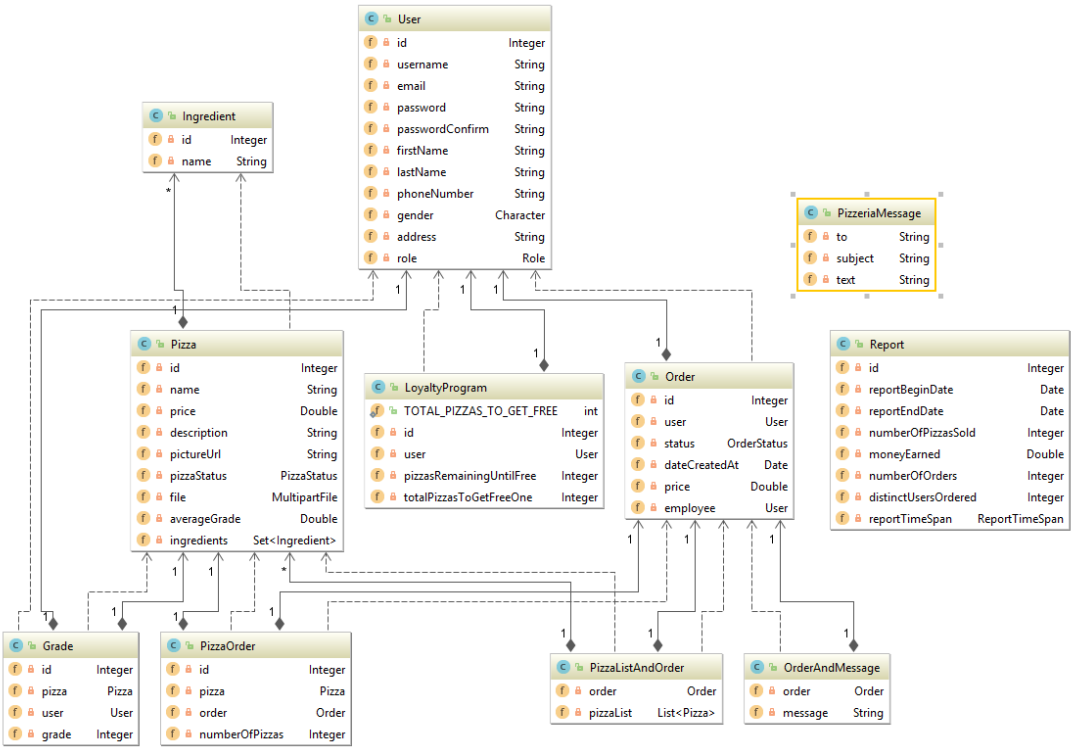
Osim konceptualnog dijagrama, ovako izgleda i stvarna implementacija u Spring Frameworku. Projekt bi mogli izdvojiti u 4 dijela a to su modeli (implementacije entiteta u bazi), repozitoriji (klase koje imaju direktan pristup bazi), service sloj (koji je spojen na repozitorij i radi finiju obradu podataka i dodatno uređivanje) te REST sloj (controlleri koji imaju na sebe spojen service sloj, a osim toga rješavaju pitanja dohvaćanja resursa i HTTP zahtjeva).



Slika 6.2.2. Slika REST i service(business) sloja



Slika 6.2.3. Service i repository sloj



Slika 6.2.4. Modeli

6.3. Dijagram objekata

U nastavku je prikazan dijagram objekata za konceptualni dijagram razreda kako bi se prikazala realna situacija u sustavu za upravljanje radom pizzerije.

Možemo vidjeti iz svakog razreda po jednu ili dvije instance. **Customer** *customer1* ima u svojoj listi narudžbi dvije narudžbe (**Order**, u članskoj varijabli List<**Order**> *orders*), te je u svakoj narudžbi naručio po jednu **Pizza**-u, u ovom slučaju to su bile *margarita* i *mijesana.*

Iz svake pizze možemo vidjeti njene sastojke koji su u listi *ingredients* tipa **Ingredient**. **Pizza** *margarita* ima ocjenu *grade1* te ju je *customer1* ocijenio s ocjenom 5, dok *mijesana* nema još nikakvih ocjena unesenih za nju.

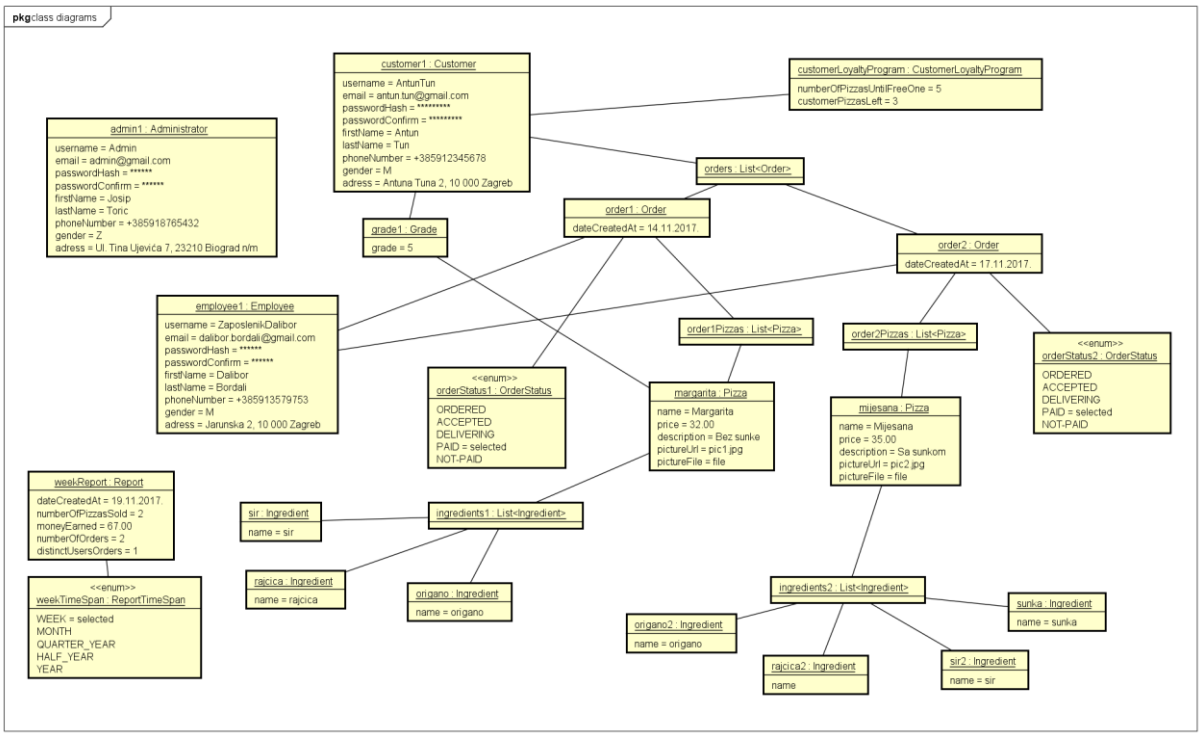
U narudžbama možemo vidjeti također da su obje plaćene (**OrderStatus**, *PAID* = selected), te su povezane s **Employee** objektom *employee1* koji predstavlja zaposlenika Dalibora koji je prihvatio narudžbu.

Customer1 koji se zove AntunTun ima i svoj *customerLoyaltyProgram* odnosno program vjernosti. Trenutno mu je još ostalo 3 **Pizza**-e do popusta.

Admin1 je također kreiran i on nije povezan ni s čim, ali ima metode navedene u dijagramu razreda te može uređivati cijelu ponudu i objekte u bazi.

Administrator može vidjeti *weekReport*, a to je instanca izvještaja (**Report**) koji je bio kreiran ovaj tjedan (*WEEKLY* = selected u enumeraciji **ReportTimeSpan** instanciranog sa *weekTimeSpan*. Možemo vidjeti statistiku izvještaja, npr. koliko je **Pizza**-a prodano taj tjedan, koliko je novca zarađeno itd.

U nastavku slijedi slika dijagrama svih instanciranih objekata u nekom trenutku.



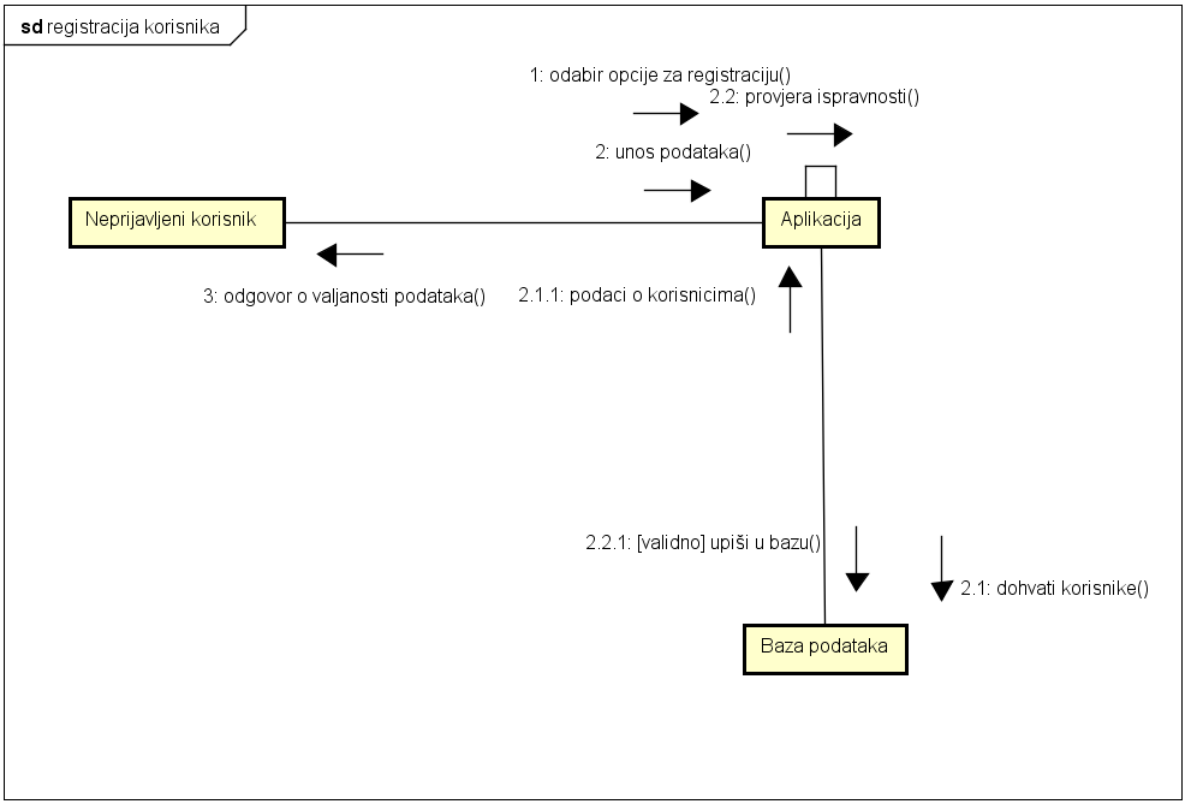
Slika 6.3.1. Priloženi dijagram objekata po uzoru na konceptualni dijagram razreda

6.4. *Ostali UML dijagrami*

Komunikacijski dijagrami

Komunikacijski dijagram – registracija korisnika:

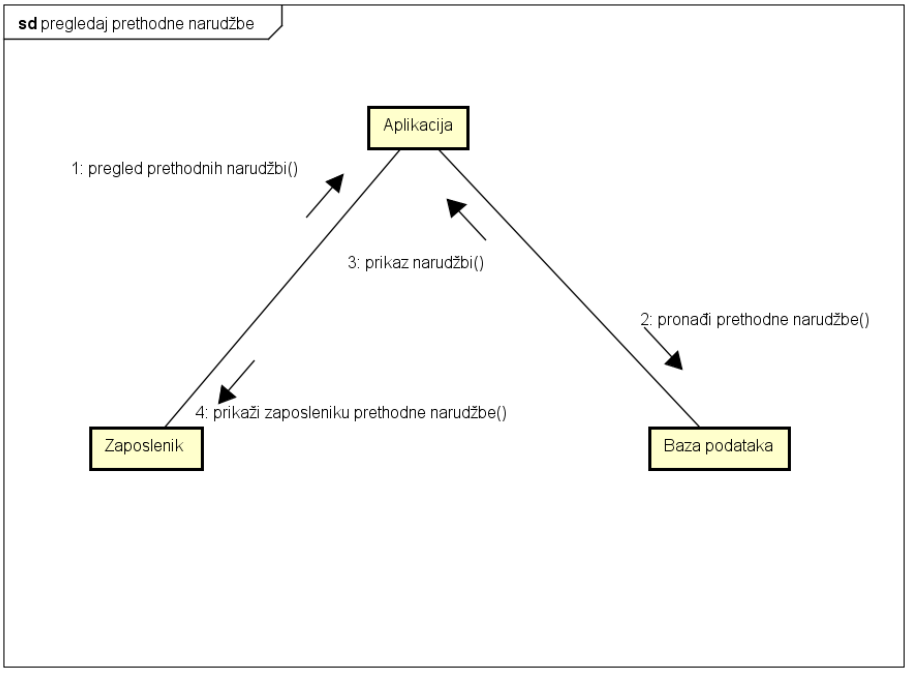
Neprijavljeni korisnik odabire opciju za registraciju te unosi svoje podatke u polja za registraciju. Sustav u komunikaciji s bazom podataka vidi postoji li takav korisnik u bazi ili su podatci neispravni te ako je registracija uspjela, upisuje ga u bazu.



Slika 6.4.1. Komunikacijski dijagram za registraciju korisnika

Komunikacijski dijagram – pregledaj prethodne narudžbe:

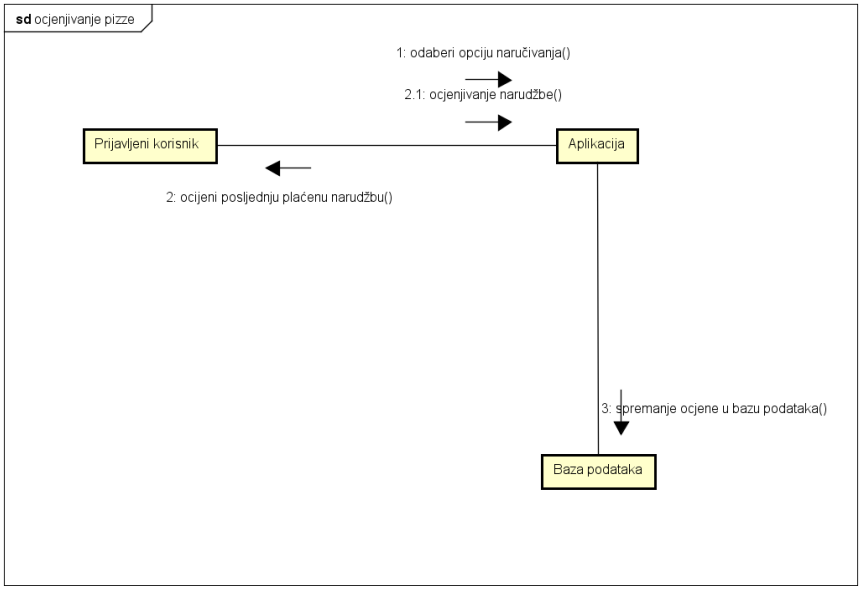
Zaposlenik na sučelju za zaposlenike odabire opciju za pregled prethodnih narudžbi Pregledava status i podatke o narudžbi koji su u bazi podataka te mu sustav ispisuje podatke o prethodnim narudžbama.



Slika 6.4.2. Komunikacijski dijagram za pregledavanje prethodnih narudžbi

Komunikacijski dijagram – ocjenjivanje pizze:

Korisnik odabire opciju ocjenjivanja narudžbe. Sustav nudi korisniku da ocijeni svoju zadnju narudžbu ocjenom od 1 do 5. Nakon toga sustav sprema ocjenu u bazu podataka.



Slika 6.4.3. Komunikacijski dijagram - ocjenjivanje pizze

Dijagrami stanja

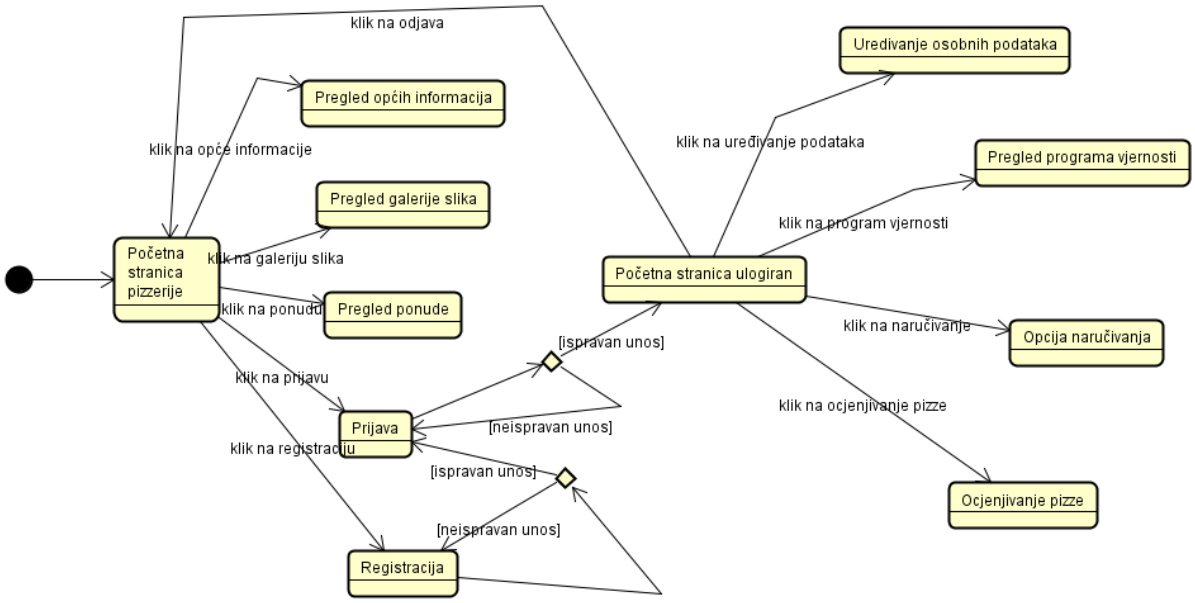
Dijagrami stanja su ponašajni i dinamički dijagrami, a razlika u odnosu na ostale UML-dijagrame je to što opisuju diskretna stanja sustava i prijelaze između tih stanja.

Na prvoj slici (Slika 6.4.4.) je prikazan dijagram stanja za prijavljenog korisnika. Korisnik dolazi na početnu stranicu i ima mogućnost jednim klikom otići u različita stanja kao što su pregled ponude, pregled općih informacija o pizzeriji, pregled galerije slika, prijava i registracija.

Ako korisnik uđe u stanje registracija unosom ispravnih podataka odlazi u stanje prijava, a unosom neispravnih podataka ponovo se vraća u stanje registracija. Kada se korisnik nađe u stanju prijava unosom ispravnih podataka ulazi u stanje početna stranica ulogiran i tu mu se ponovno nudi mogućnost da samo jednim klikom ode u nova stanja, a unosom neispravnih podataka zadržava se u stanju prijava.

Stanje uređivanje osobnih podataka služi za izmjenu podataka unesenih za vrijeme registracije, u stanju pregled programa vjernosti može provjeriti trenutno stanje programa vjernosti.

Pizzu može naručiti ako se nađe u stanju opcija naručivanja, a ocijeniti je može ako ode u stanje ocjenjivanje pizze . Prijavljenom korisniku je stalno omogućen prijelaz u stanje početna stranica pizzerije klikom na odjavu.



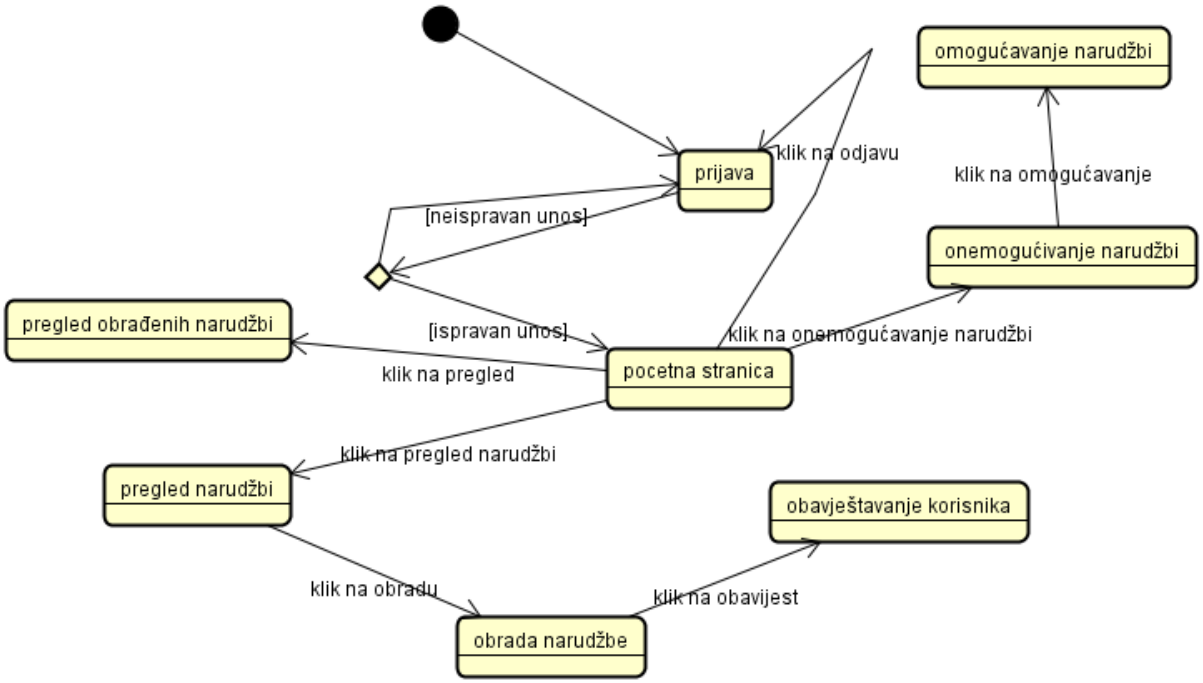
Slika 6.4.4. Dijagram stanja za prijavljenog korisnika

Na sljedećoj slici (Slika 6.4.5.) se nalazi dijagram stanja zaposlenika.

Zaposlenik se uvijek mora prijaviti da bi mogao ići u druga stanja. Ako u stanju prijava unese neispravne podatke zadržava se u tom stanju, a s ispravnim podatcima dolazi u stanje početna stranica.

Korisnik može iz stanja početna stranica otići u stanje pregled narudžbi i vidjeti trenutne narudže te klikom odabrati narudžbu koju želi obraditi i tako prijeći u stanje obrada narudžbi.

Također može otići u stanje pregled obrađenih narudžbi gdje može vidjeti sve narudžbe koje je obradio,u slučaju velike gužve može otići u stanje onemogućavanje narudžbi i privremeno onemogućiti korisnicima narudžbe. Iz tog stanja može preći u stanje omogućavanje narudžbi kada se ponovno stvore uvjeti za obradu narudžbi.



Slika 6.4.5. Dijagram stanja za zaposlenika

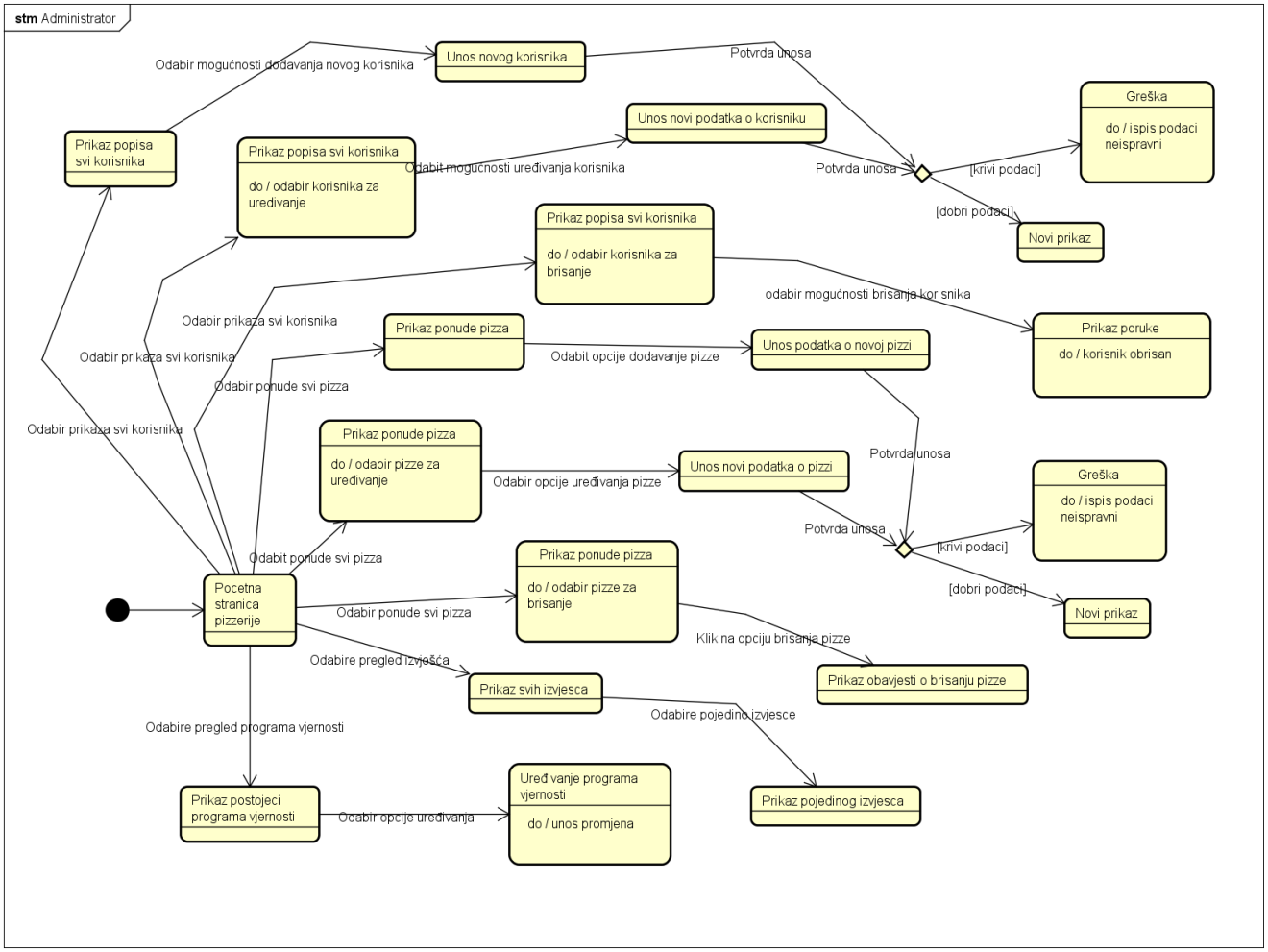
Sljedeća slika (Slika 6.4.6.) prikazuje dijagram stanja za administratora.

Administrator s početne stranice pizzerije može vidjeti popis svih korisnika, ponudu pizza, sva izvješća te prikaz postojećih programa vjernosti. Ukoliko se odluči za prikaz svih korisnika, ima mogućnost unijeti novog korisnika (zahtjeva potvrdu unosa). U slučaju krivo unesenih podataka događa se greška uz ispis da su podaci neispravni, a ako su podaci ispravni, pojavljuje se novi prikaz uz izmijenjene podatke. Ista stvar događa se i kada administrator pristupa promjeni korisničkih podataka korisnika. Administrator može odabrati i brisanje korisnika s popisa te se u tom slučaju pojavljuje poruka da je korisnik obrisan.

Ako administrator odabere opciju pregleda ponude pizza, ima mogućnost dodati novu pizzu u ponudu ili izmijeniti podatke o postojećoj pizzi (obje radnje zahtijevaju potvrdu unosa). Ukoliko su podaci bili ispravni pokazuje se novi prikaz ponude s unesenim podacima, a u suprotnom se pojavljuje greška uz ispis da su uneseni podaci neispravni. Administrator može i obrisati pizzu iz ponude te mu se u tom slučaju prikazuje obavijest da je pizza obrisana.

Također, administrator može vidjeti i prikaz svih izvješća ili odabrati prikaz pojedinog izvješća.

Naposljetku, administrator može odabrati i prikaz postojećih programa vjernosti te unijeti promjene u njima.



Slika 6.4.6. Dijagram stanja za administratora

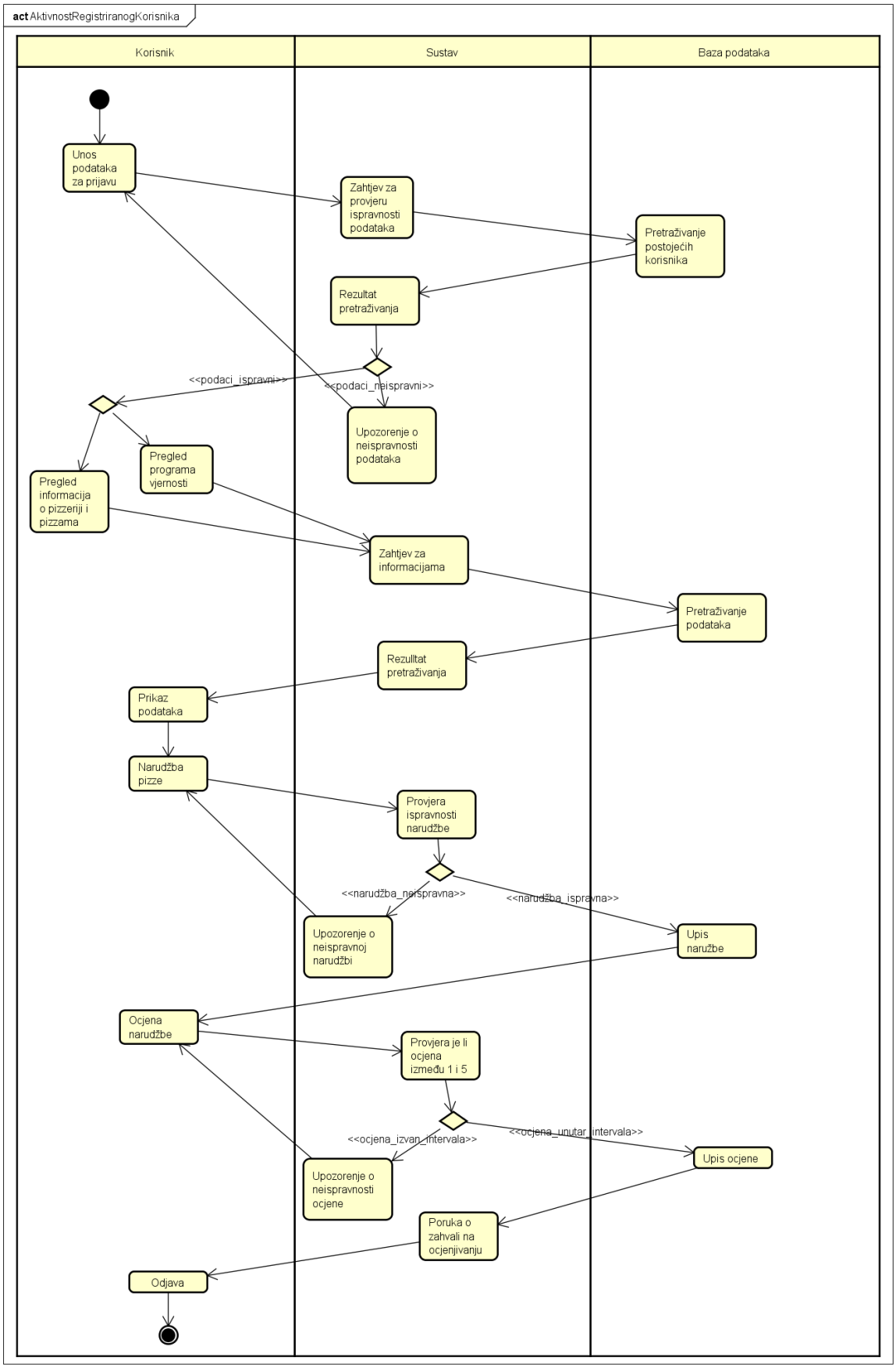
Dijagram aktivnosti

Dijagram aktivnosti je ponašajni dijagram i dinamički dijagram. Prikazuje radni tok aktivnosti koje se obavljaju u sustavu korak po korak. Na slici (Slika 6.4.6.) je prikazan dijagram aktivnosti registriranog korisnika koji želi naručiti pizzu.

Korisnik odabire opciju za prijavu te upisuje svoje podatke. Nakon toga sustav provjerava postoji li korisnik u bazi podataka, ako postoji, korisnik je prijavljen, inače ga sustav upozorava i nudi mogućnost ponovne prijave.

Zatim korisnik pregledava informacije na web stranici tj. odabire informacije o pizzeriji ili pregled ponude pizza ili pregledava program vjernosti, a sustav ispisuje tražene informacije. Nakon toga odabire opciju naručivanja pizze te naručuje željenu pizzu. Sustav provjerava je li narudžba ispravna. Ako je ispravna, zapisuje ju u bazu podataka, inače šalje upozorenje.

Nakon nekog vremena korisnik odabire opciju ocjenjivanja narudžbe te ju ocjenjuje ocjenama 1-5. Ako je interval ispravan, ocjena se sprema u bazu podataka, inače sustav upozorava korisnika. Nakon uspješnog ocjenjivanja, sustav šalje poruku korisniku u kojoj se zahvaljuje na ocjenjivanju.



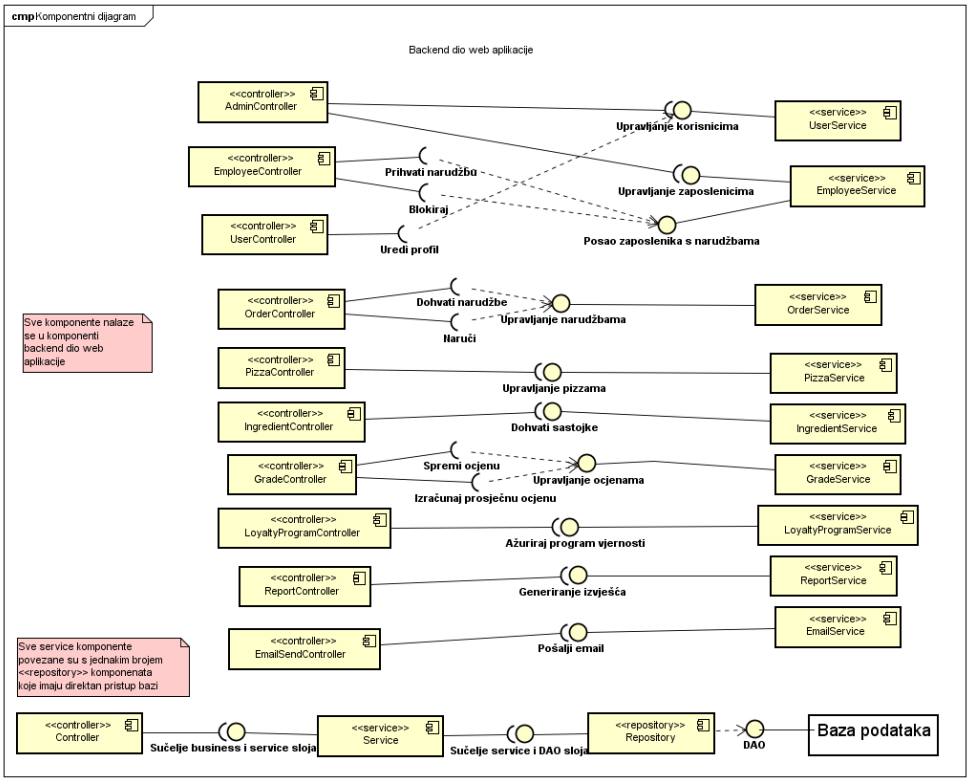
Slika 6.4.7. Dijagram aktivnosti za prijavljenog korisnika

Komponentni dijagram

Dijagram komponenti sastoji se od više slojeva istovrsnih komponenti koje oblikuju strukturu sustava. Iz razloga što je implementacija samog projekta bila rađena u Spring Frameworku te Angular Frameworku, najjednostavnije je iz toga izdvojiti 2 dijagrama koja opisuju backend dio i frontend dio.

Korisnik unosom URL-a u svom browseru ili pritiskom ne neku hipervezu na stranici web aplikacije pokreće korespondirajući Controller mapiran na taj URL. Osim same hiperveze, određen je i HTTP zahtjev koji također određuje metodu koju će pojedini Controller pokrenuti. Controller komponente u REST sloju sloju komuniciraju sa Service komponentama u REST sloju, a zadaća Service komponenti jest usklađivanje i „uljepšavanje“ podataka iz baze (SQL upiti) s podacima iz Controllera (forme modela).

Konačno, same upite izvršava DAO/Repository sloj modeliran repository klasama koje izvršavaju SQL upite i dohvaćaju, brišu, unos i ažuriraju podatke u bazi. Sam komponentni dijagram backend dijela web aplikacije prikazan je na slici dolje (Slika 6.4.7.).



Slika 6.4.8. Komponentni dijagram

Za bolje snalaženje u ovom komponentnom dijagramu, naveden je opis funkcionalnosti pojedinog Controllera:

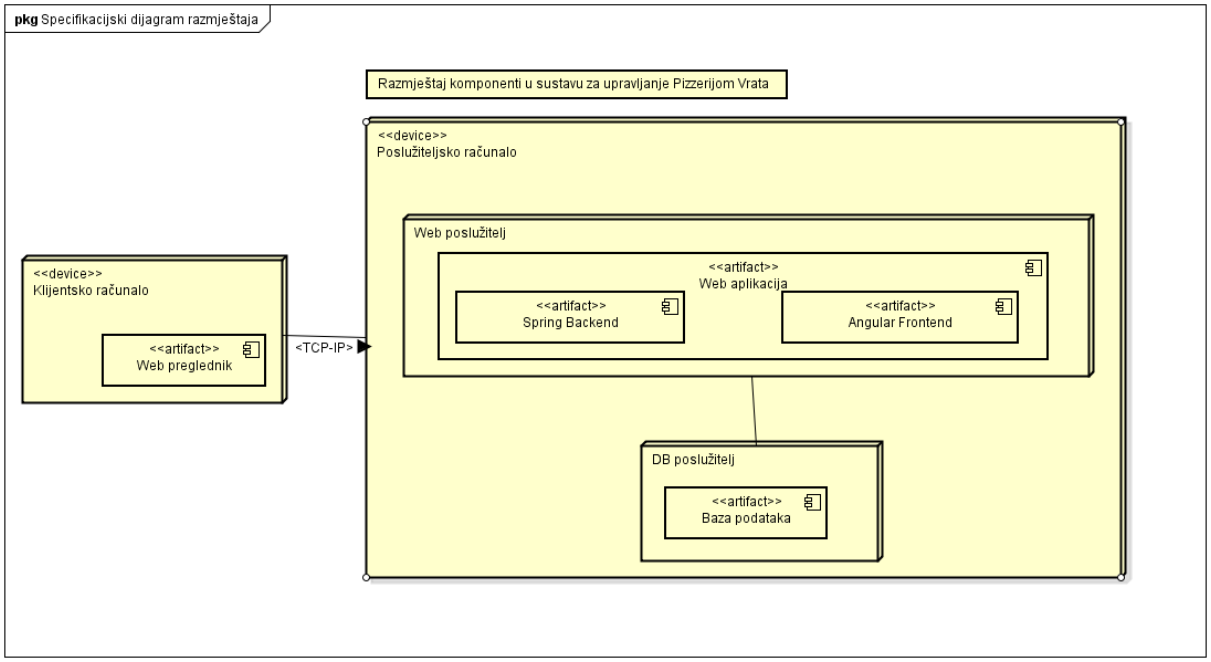
- ***AdminController*** – upravljanje korisnicima i zaposlenicima (dodavanje/brisanje/uređivanje)
- ***EmployeeController*** – uređivanje profila, upravljanje/prihvaćanje/blokiranje narudžbi
- ***UserController*** – uređivanje profila
- ***OrderController*** – zadužen za naručivanje, pregled i dohvaćanje pojedinih narudžbi
- ***PizzaController*** – upravljanje pizzama (dohvat/brisanje/uređivanje)
- ***IngredientController*** – dohvat sastojaka (koristi se obično uz PizzaController)
- ***GradeController*** – ocjenjivanje pizze te spremanje u bazu, računanje prosječne ocjene pizze (potpomognuto GradeServiceom)
- ***LoyaltyProgramController*** **-** kreiranje loyalty programa, ažuriranje
- ***ReportController*** – kreiranje izvješća i prikaz
- ***EmailSendController*** – slanje informativnih e-mailova

7. Implementacija i korisničko sučelje

7.1. Dijagram razmještaja

Dijagram razmještaja

Slika (Slika 7.1.) prikazuje specifikacijski dijagram razmještaja u sustavu za upravljanje pizzerijom. Klijentsko računalo se povezuje s poslužiteljem putem Web preglednika te kod HTTP zahtjeva stvara TCP-IP vezu u mreži. Samo poslužiteljsko računalo sastoji se od više dijelova, a mogu se izdojiti 3 osnovne i najbitnije komponente u Web aplikaciji: backend dio, frontend dio te baza podataka.



Slika 7.1. Dijagram razmještaja

7.2. Korištene tehnologije i alati

Pri izradi web aplikacije korištene su mnoge tehnologije koje su pomogle u izradi backend i frontend dijela aplikacije. Kao razvojna okolina za backend dio korišten je IntelliJ Idea Ultimate dok je sama backend logika rađena u programskom jeziku Javi i Spring Frameworku, točnije ***Spring Boot*** uz ***Rest API*** za obradu HTTP zahtjeva poslužiteljskog dijela.

Za rad s bazom podataka koristio se ***PostgreSQL*** sustav za upravljanje bazom podataka te je ***Hibernate ORM*** poslužio za mapiranje objekata u relacije baze.

Dodatno, kao Web poslužitelj uzet je ***Apache Tomcat*** koji je direktno ugrađen u Spring Boot Framework.

Frontend dio koji je služio za izgled same Web aplikacije rađen je u Angular 4 Frameworku a odabrana je razvojna okolina ***WebStorm***.
Kao pomoć za crtanje UML dijagrama poslužio je alat Astah, a za Version Control dokumentacije i implementacije projekta korištena je Gitlab usluga.

IntelliJ Idea Ultimate
IntelliJ IDEA je razvojna okolina koju je razvila tvrtka JetBrains, a služi za pisanje i razvijanje računalnog softvera (desktop i web aplikacija) u programskom jeziku Javi.

Spring Boot Framework
Spring Boot je projekt koji se naslanja na Spring Framework te omogućava lakšu izradu Spring aplikacija, pa tako i Spring web aplikacija. Pruža široki spektar opcija za izradu aplikacija te olakšava posao programeru koji se ne treba brinuti o mnogim implementacijskim detaljima. Neke od funkcionalnosti koje Spring Boot sam rješava su pokretanje Jettyja ili Apache Tomcat poslužitelja te skrivanje velikog dijela koda zbog ugrađene funkcionalnosti koju je moguće lako izmijeniti s konfiguracijskim datotekama.

Rest API

Rest API je temeljni dio razvoja Web aplikacija baziran oko HTTP zahtjeva. Definira set funkcija s kojima programeri mogu napraviti HTTP zahtjeve i dobivati odgovore, a tehnologija je zamišljena tako da HTTP pozivi odgovaraju onome što se obavi u aplikaciji (npr. GET – dohvat, POST – upis, PUT – ažuriranje, DELETE – brisanje…).

PostgreSQL

PostgreSQL je objektno relacijskiSUBP (sustav za upravljanje bazom podataka) koji podržava velik dio SQL standarda poput kreiranja i upravljanja bazom podataka, očuvanje integriteta, kompleksne upite, trigere itd.

Hibernate ORM

Hibernate ORM je radni okvir za mapiranje Javinih objekata s relacijskim modelom u bazi podataka (u našem slučaju PostgreSQL). Osim toga, nudi brojne anotacije koje olakšavaju rad s entitetima, upitima te generiranje tablica u bazi podataka.

Apache Tomcat

Apache Tomcat je open source Web poslužitelj koji implementira nekoliko Java EE specifikacija kao što su Java Servlet, JSP (Java Servlet Pages), Java EL (Java Expression Language). Posebnost Apache Tomcata je što omogućava izvođenje koda napisanog u Javi na svom Web poslužitelju.

Angular 4 Framework

WebStorm

7.3. Isječak programskog koda vezan za temeljnu funkcionalnost sustava

U ovom odlomku biti će izdvojeni bitni dijelovi programskog koda važni za razumijevanje implementacije sustava za rad pizzerije.

Overview

Iako je kod na slici dolje (Slika 7.3.1.) izrazito malen, on je jako bitan jer počinje Spring Boot aplikaciju. Omogućava mnoge funkcionalnosti kao što su pokretanje Apache Tomcat Servera, instanciranje Dispatcher Servleta i kreiranje Beanova odnosno primjeraka odabranih klasa za rad bez potrebe korisnikovo manualnog rada. Uz to, mapira i potrebne URL-ove na za to predviđene servlete te po potrebi mapira objekte u relacije i kreira relacije uz pomoć Hibernate-a .

```
@SpringBootApplication
public class PizzeriaSystemApplication extends SpringBootServletInitializer{

    @Override
    protected SpringApplicationBuilder configure(SpringApplicationBuilder application) {
        return application.sources(PizzeriaSystemApplication.class);
    }

    public static void main(String[] args) {
        SpringApplication.run(PizzeriaSystemApplication.class, args);
    }
```

Slika 7.3.1. Pokretanje Spring aplikacije

REST sloj

REST sloj predstavlja servlete koji su zaduženi za obavljanje zahtjeva. Kada korisnik unese URL ili klikne na mjesto koje pokreće određeni HTTP zahtjev, mapirani Controller za taj URL i specifični zahtjev (npr. GET, POST, PUT, DELETE…) se okine i pokreće izvođenje metode. GET zahtjevi se obično koriste za dohvaćanje resursa, POST za spremanje u bazu, PUT za ažuriranje te DELETE za brisanje.

Na slici (Slika 7.3.2.) prikazan je PizzaController (anotacija @Controller) koji je zadužen za aktivnosti vezane uz Pizza object i relaciju.

Anotacijom @{httpZahtjev}Mapping specificira se koji je zahtjev u pitanju a pridružuje mu se određeni URL. Povratna vrijednost ResponseEntity se koristi za vraćanje objekata i HTTP statusnog koda da dojavi uspjeh ili neuspjeh u izvođenju.

Svaki Controller može imati jednu ili više service/business komponenti koje su objašnjene ispod.

```java
@RestController
public class PizzaController {

    @Autowired
    private PizzaService pizzaService;

    @Autowired
    private GradeService gradeService;

    @Autowired
    private SecurityService securityService;

    //sve pizze
    @GetMapping("/pizza")
    public ResponseEntity<List<Pizza>> getPizzas() {
        List<Pizza> pizzas = pizzaService.findAll();
        assignPizzaGrades(pizzas);
        return new ResponseEntity<>(pizzas,HttpStatus.OK);
    }

    @GetMapping("/offer/pizza")
    public ResponseEntity<List<Pizza>> getOfferedPizzas() {
        List<Pizza> pizzas = pizzaService.findAllByPizzaStatus(PizzaStatus.IN_OFFER);
        assignPizzaGrades(pizzas);
        return new ResponseEntity<>(pizzas,HttpStatus.OK);
    }

    //pizza po id-u
    @GetMapping("/pizza/{id}")
    public ResponseEntity<Pizza> getPizza(@PathVariable Integer id) {
        Pizza pizza = pizzaService.findOne(id);

        if (pizza == null) {
            return new ResponseEntity<>(HttpStatus.NOT_FOUND);
        }

        return new ResponseEntity<>(pizza, HttpStatus.OK);
    }


    @PostMapping(value = "/admin/pizza")
    public ResponseEntity<Pizza> addPizza(@RequestBody Pizza pizzaForm) {
        User user = securityService.loggedIn(pizzaForm.getToken());

        if (user==null || user.getRole()!= Role.ADMIN){
            return new ResponseEntity<>(HttpStatus.FORBIDDEN);
        }
        pizzaForm.setPizzaStatus(PizzaStatus.IN_OFFER);
        pizzaService.save(pizzaForm);
        return new ResponseEntity<>(pizzaForm,HttpStatus.OK);
    }
```

Slika 7.3.2. Primjer REST Controllera

Business sloj

Business sloj je spona između REST i DAO sloja te omogućava izvođenje logike aplikacije. Pretpostavlja se da ima potrebne DAO objekte koji mu daju podatke iz baze na što on radi finiju doradu i uređiva podatke.

Podatke je moguće ponovo dati DAO sloju na spremanje u bazu ili vratiti nazad REST sloju za prikaz na Web stranici. Business sloj označen je anotacijom @Service.

Anotacija @Transactional osigurava izvođenje cijele logike na razini transakcije da ukloni mogućnost gubitka podataka te istodobnog pristupa.

```
@Service
public class OrderService {

    @Autowired
    private OrderRepository orderRepository;

    @Transactional
    public List<Order> findAllById(Integer id) { return orderRepository.findAllById(id); }

    @Transactional
    public List<Order> findPastUserOrders(User user) {
        List<Order> orders = orderRepository.findAllByUser(user);
        if (orders == null) orders = new ArrayList<>();

        return orders.stream().filter(OrderStatus::isPast).collect(Collectors.toList());
    }

    @Transactional
    public List<Order> findPastEmployeeOrders(User employee) {
        List<Order> orders = orderRepository.findAllByEmployee(employee);
        if (orders == null) orders = new ArrayList<>();

        return orders.stream().filter(OrderStatus::isPast).collect(Collectors.toList());
    }

    @Transactional
    public List<Order> findCurrentEmployeeOrders(User employee) {
        List<Order> orders = orderRepository.findAllByEmployee(employee);
        if (orders == null) orders = new ArrayList<>();

        return orders.stream().filter(e -> !OrderStatus.isPast(e)).collect(Collectors.toList());
    }

    @Transactional
    public List<Order> findAllByUser(User user) { return orderRepository.findAllByUser(user); }

    @Transactional
    public List<Order> findCurrentUserOrders(User user) {
        List<Order> orders = orderRepository.findAllByUser(user);
        if (orders == null) orders = new ArrayList<>();
        return orders.stream().filter(e -> !OrderStatus.isPast(e)).collect(Collectors.toList());
    }

    @Transactional
    public void save(Order order) { orderRepository.save(order); }

    @Transactional
    public Order findOne(Integer id) { return orderRepository.findOne(id); }

    @Transactional
    public List<Order> findAllByEmployee(User employee) { return orderRepository.findAllByEmployee(employee); }
```

Slika 7.3.3. Primjer service usluge koja je posrednik između REST i DAO sloja

DAO sloj

DAO sloj predstavlja objekte koji mogu komunicirati s bazom i raditi SQL upite. Označeni su anotacijom @Repository, a funkcionalnost Spring Frameworka omogućava i izvođenje upita bez njihovog pisanja u Standard Query Languageu (SQL) ili pisanja metoda koje rade isto.

Za složenije upite koje Spring ne može razlučiti, koristi se anotacija @Query kojom korisnik može direktno napisati složeni SQL upit. Na slici (Slika 7.3.4.) prikazan je OrderRepository.

Prve 3 metode je Spring odmah prepoznao te ponudio svoju implementaciju, pa programmer mora samo unijeti deklaraciju metoda da one postanu dostupne.

U nastavku su prikazane složenije naredbe za generiranje izvješća u pizzeriji te je za njih bilo potrebno napisati SQL upit. Usprkos tome, nije bilo potrebno pisati implementaciju same metode.

```
@Repository
public interface OrderRepository extends JpaRepository<Order, Integer> {

    List<Order> findAllById(Integer id);
    List<Order> findAllByUser(User user);
    List<Order> findAllByEmployee(User employee);

    @Query(value = "SELECT sum(o.price) FROM Order o WHERE o.dateCreatedAt BETWEEN ?1 AND ?2")
    Double getTotalMoneyEarned(Date beginDate, Date endDate);

    @Query(value = "SELECT COUNT(o.id) FROM Order o WHERE o.dateCreatedAt BETWEEN ?1 AND ?2")
    Integer getNumberOfOrders(Date beginDate, Date endDate);

    @Query(value = "SELECT DISTINCT(o.user) FROM Order o WHERE o.dateCreatedAt BETWEEN ?1 AND ?2")
    List<User> getDistinctUsersOrdered(Date beginDate, Date endDate);

    @Query(value = "SELECT * FROM orders WHERE date_created_at BETWEEN ?1 AND ?2", nativeQuery = true)
    List<Order> getOrdersByDateSpan(Date beginDate, Date endDate);

    @Query(value = "SELECT max(o.price) FROM Order o WHERE o.dateCreatedAt BETWEEN ?1 AND ?2")
    Double getMaxOrderPrice(Date beginDate, Date endDate);

    @Query(value = "SELECT min(o.price) FROM Order o WHERE o.dateCreatedAt BETWEEN ?1 AND ?2")
    Double getMinOrderPrice(Date beginDate, Date endDate);
}
```

Slika 7.3.4. Primjer DAO objekta za pristup bazi

Modeli

Modeli su označeni anotacijom @Entity, čime se oni prikazuji kao relacije u bazi podataka. To su jednostavni objekti s privatnim svojstvima i javnim getterima i setterima za lakšu manipulaciju nad njima.

Anotacijom @Column specificira se da je članska varijabla objekta stupac relacije podataka, a kao što možemo vidjeti na slici (Slika 7.3.5.) Moguće je i specificirati njihova imena, ime relacije, mogućnost stavljanja NULL vrijednosti u polje u bazi itd.

```java
@Entity
@Table(name = "users")
public class User {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "user_id")
    private Integer id;

    @Column(name = "username", nullable = false)
    private String username;

    @Column(name = "email", nullable = false)
    private String email;

    @Column(name = "password", nullable = false)
    private String password;

    @Column(name = "first_name", nullable = false)
    private String firstName;

    @Column(name = "last_name", nullable = false)
    private String lastName;

    @Column(name = "phone_number", nullable = false)
    private String phoneNumber;

    @Column(name = "gender", nullable = false)
    private String gender;

    @Column(name = "address", nullable = false)
    private String address;

    @Column(name = "role", nullable = false)
    @Enumerated(EnumType.STRING)
    private Role role;
```

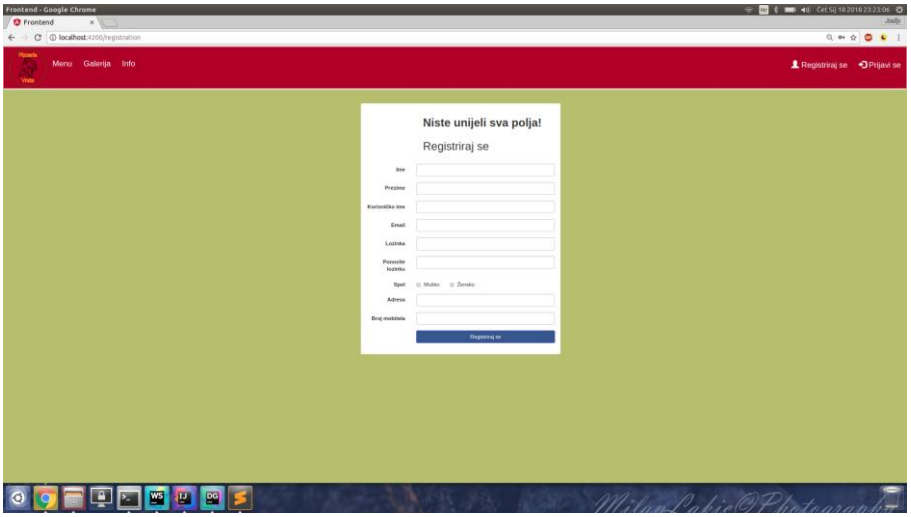
Slika 7.3.5. Primjer modela korisnika

7.4. *Ispitivanje programskog rješenja*

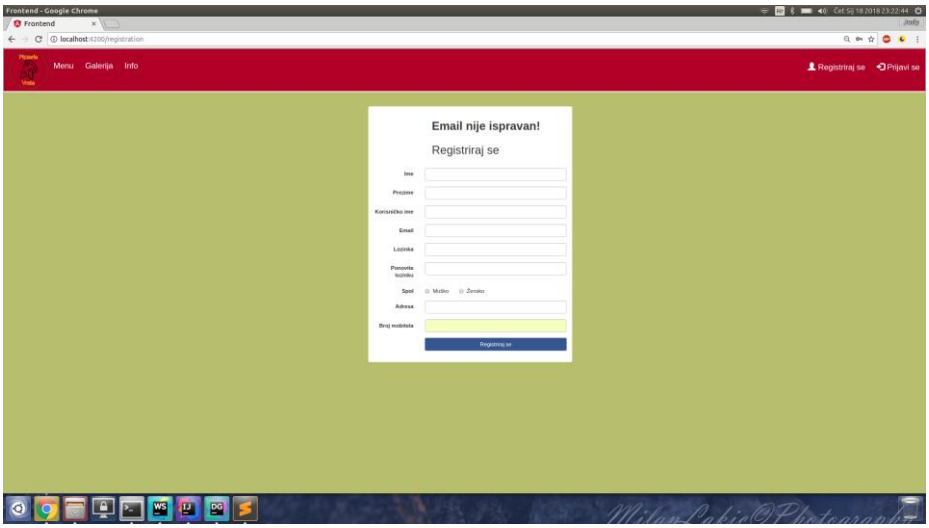
7.4.1 Test 1: Registracija korisnika

Očekivano: prijavljeni korisnik klikom na gumb „Registriraj se“ dobiva formu kojom može unijeti svoje podatke i registrirati se. Forma se sastoji od unosa imena, prezimena, adrese, broja telefona, emaila, korisničkog imena i lozinke. Ukoliko su sva polja ispravna, korisnik će se uspješno registrirati u bazu, u protivnom, sustav će dojaviti grešku.

Rezultat: Korisnik je uspješno registriran u bazu, ili je sustav dojavio grešku.



Slika 7.4.1.1. Test za neunesena polja u formi

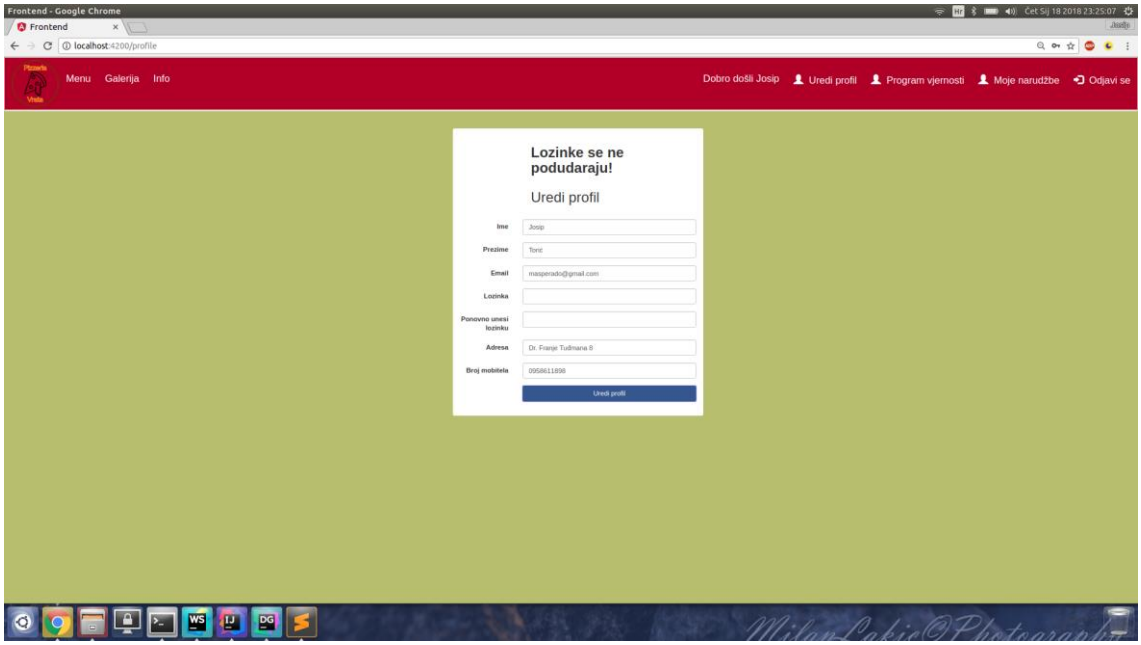


Slika 7.4.1.2. Test za neispravan email

7.4.2 Test 2: Izmjena profila korisnika

Očekivano: Prijavom u sustav, korisniku se prikazuje njegov profil koji je moguće izmijeniti. Forma za izmjenu korisnika slična je kao i ona kod registracije, ali nije moguće promijeniti korisničko ime. Ako su svi podaci ispravno uneseni, korisnik će biti ispravno ažuriran. U protivnom, sustav će dojaviti grešku

Rezultat: Korisnik je uspješno izmijenio svoje podatke, ili je sustav dojavio grešku.



Slika 7.4.2.1. Test za nepodudaranje lozinki

7.4.3 Test 3: Prelazak praga programa vjernosti za popust u narudžbi

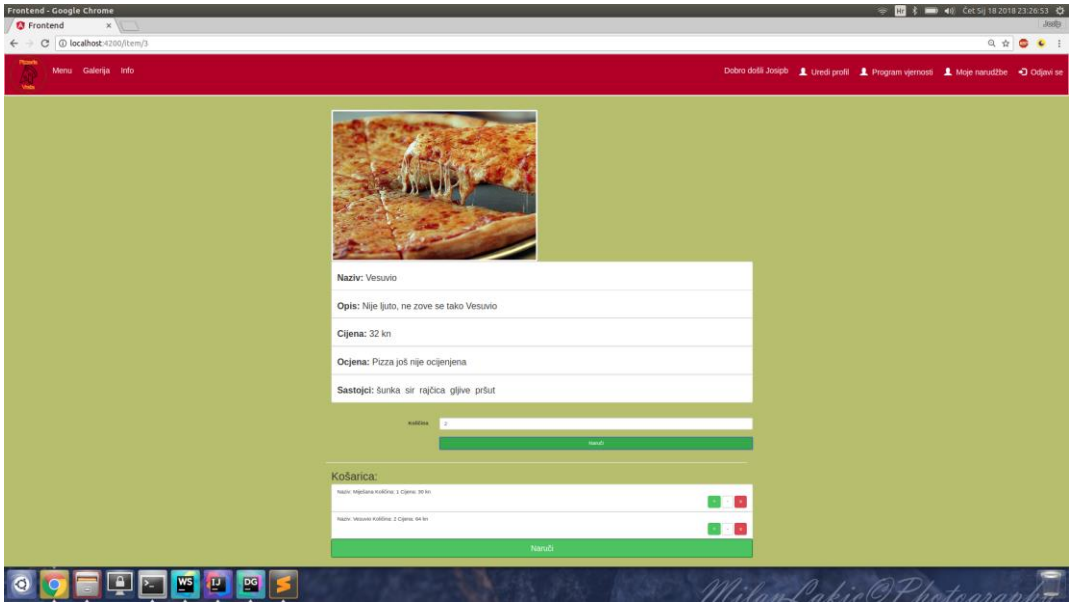
Očekivano: Za svakog korisnika veže se njegov program vjernosti s kojim može ostvariti popust na narudžbu. Kad prijeđe limit od određenog broja pizza, popust bi se trebao uračunati u narudžbu te bi se pojeftinjena cijena trebala prikazati korisniku. Time bi korisnik bio obaviješten da je prešao svoj prag vjernosti, a stari se istom prilikom mora ažurirati na početno stanje.

Rezultat: Korisnik je uspješno ostvario popust na narudžbu jer je prešao svoj prag vjernosti.

7.4.4 Test 4: Unos negativnog broja pizza u narudžbu

Očekivano: Kod naručivanja pizza, u formu za naručivanje moguće je unijeti bilo što, pa tako i text ili negativan broj pizza za unos. Kod ovakvog ponašanja, sustav bi trebao ignorirati takve unose i ostaviti stanje kako je bilo prije invalidnog unosa.

Rezultat: Sustav je onemogućio dodavanje negativnog broja pizza u narudžbu.



Slika 7.4.4. 1. Test za unos negativnog broja narudžbi ignorira se, a za neocijenjenu pizzu ispisuje se poruka da nije ocijenjena

7.4.5. Test 5: Prikaz neocijenjene pizze u narudžbi

Očekivano: Kako ne bi došlo do greške kod prikaza pizze koja nije ocijenjena, morala bi se ispisati prikladna poruka. Samo ocijenjene pizze mogu imati svoju prosječnu ocjenu od 1 do 5. U protivnom, u polju ocjena prikaže se „Pizza još nije ocijenjena“

Rezultat: Sustav je onemogućio dodavanje negativnog broja pizza u narudžbu.

7.5. *Upute za instalaciju*

U nastavku su prikazane detaljne upute za instalaciju projekta na operacijskim sustavima **Windows** i **LINUX**. Za instalaciju su potrebni Java JDK, PostgreSQL baza podataka, Apache Tomcat poslužitelj, Angular Framework te vaš omiljeni IDE. IDE koji se može koristiti jest IntelliJ IDEA (preferirano, https://www.jetbrains.com/idea/download), ili Eclipse (https://www.eclipse.org/downloads/). Za Angular Framework poželjno je koristiti IDE WebStorm (https://www.jetbrains.com/webstorm/download). Međutim, razvojne okoline potrebne su samo za daljnji rad na projektu a ne za produkciju.

Java

Windows: potrebno je skinuti odgovarajući JDK, a može se pronaći na stranici (http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html),

Linux:

sudo add-apt-repository ppa:webupd8team/java

sudo apt update;sudo apt install oracle-java8-installer

sudo apt install oracle-java8-set-default

Maven

Windows: (https://maven.apache.org/install.html)

Linux: sudo apt install maven

Docker

Docker je pomoćni alat za lakše instaliranje PostgreSQL baze podataka.

Windows: (https://www.docker.com/docker-windows)

Linux: (https://docs.docker.com/engine/installation/linux/docker-ce/ubuntu/)

PostgreSQL

Upute su iste za Windows i za Linux uz pomoć Dockera.

Iz Dockera se upiše naredba: docker run –name posgres955 -p 5432:5432 -e POSGRES_PASSWORD=postgres -d postgres:9.5.5.

Time se kreira Docker container imena postgress 955 izložen na portu 5432 i svaki put kad ga se želi pokrenuti ili zaustaviti obave se naredbe:

docker start postgres955

docker stop postgres955

pgAdmin

Windows: (https://www.pgdmin.org/download/pgadmin-4-windows/)

Linux: sudo apt install pgadmin3

NodeJS

Windows: (https://nodejs.org/en/download)

Linux:

curl -sL https://deb.nodesource.com/setup_8.x | suo -E bash –

sudo apt-get install -y nodejs

NPM

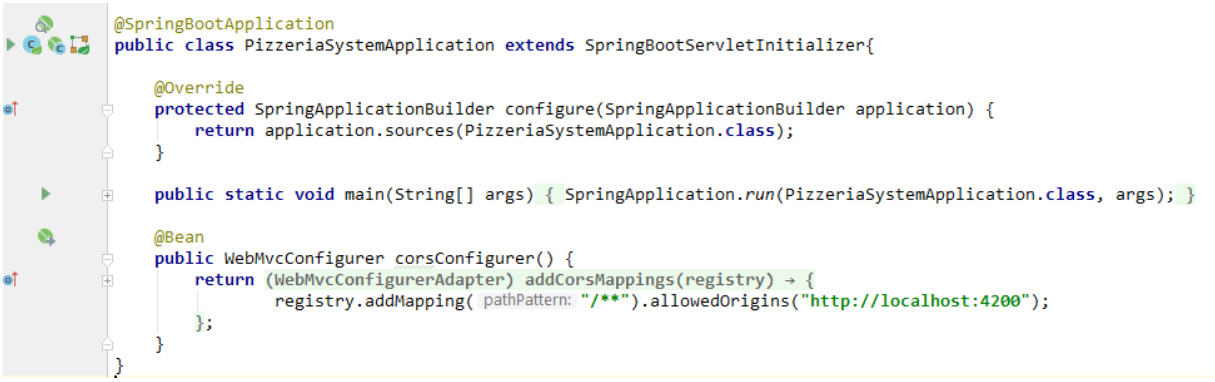
Trebao bi biti instaliran zajedno s NodeJS-om.

Angular CLI

U konzoli se treba izvršiti naredba npm install -g @angular/cli. Time se instalira Angular Framework.

IntelliJ/Eclipse IDE (pokretanje backenda)

Nakon što se vaš IDE otvori, potrebno je otvoriti backend kao projekt. Za pokretanje, potrebno je u klasi Pizzeria Application stisnuti zeleni trokut **RunApplication** (Slika 7.5.1.).



Slika 7.5.1.Pokretanje iz klase PizzeriaSystemApplication

Ako je sve uspješno prošlo, u konzoli bi se trebala ispisati krajnja poruka da je Tomcat Server pokrenut.

Slika 7.5.2. Pokretanje Tomcat poslužitelja

Pokretanje frontenda

Za oba operacijska sustava, pozicionirajte se u projekt te upišite nardbu:

ng serve

Za dodatnu konfiguraciju porta možete upisati izmijenjenu naredbu:

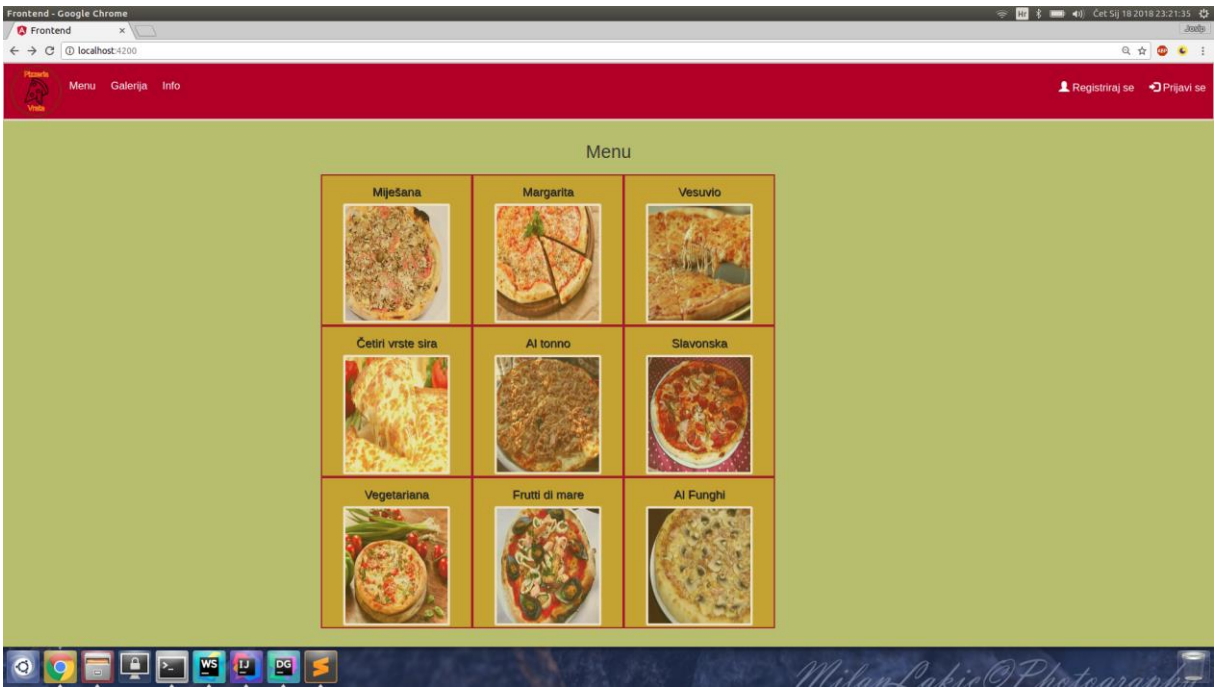
ng serve –host 0.0.0.0 –port 4201

7.6. Korisničke upute

Sustavu za upravljanje radom pizzerije mogu pristupiti neprijavljeni korisnici, prijavljeni korisnici/kupci, zaposlenici pizzerije te administratori. U nastavku slijede detaljne upute za svaku vrstu korisnika koji ulazi na Web stranice aplikacije

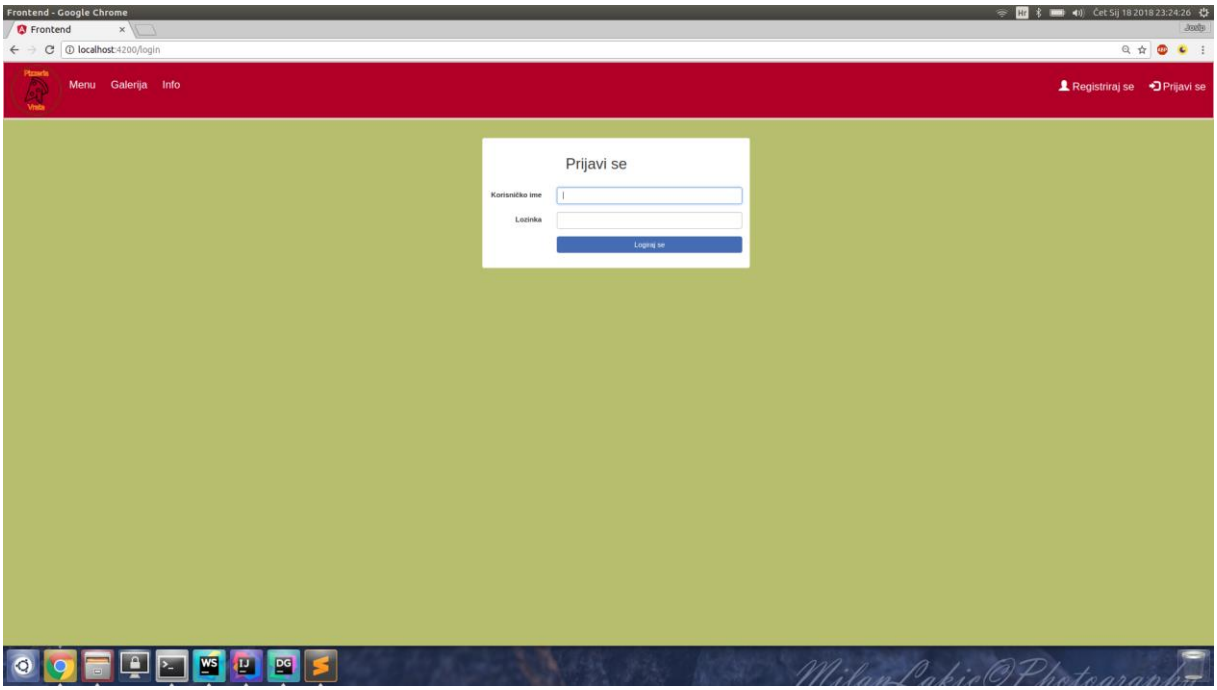
Neprijavljeni korisnici

Neprijavljeni korisnici mogu pristupati ograničenom dijelu Web aplikacije budući da nisu prijavljeni u sustav. Međutim, to ih ne sprječava od pogleda na pizze koje se nalaze u pizzeriji. Klikom na pojedinu pizzu moguće je vidjeti detalje same pizze. Međutim, to je sve što se tiče početne stranice.



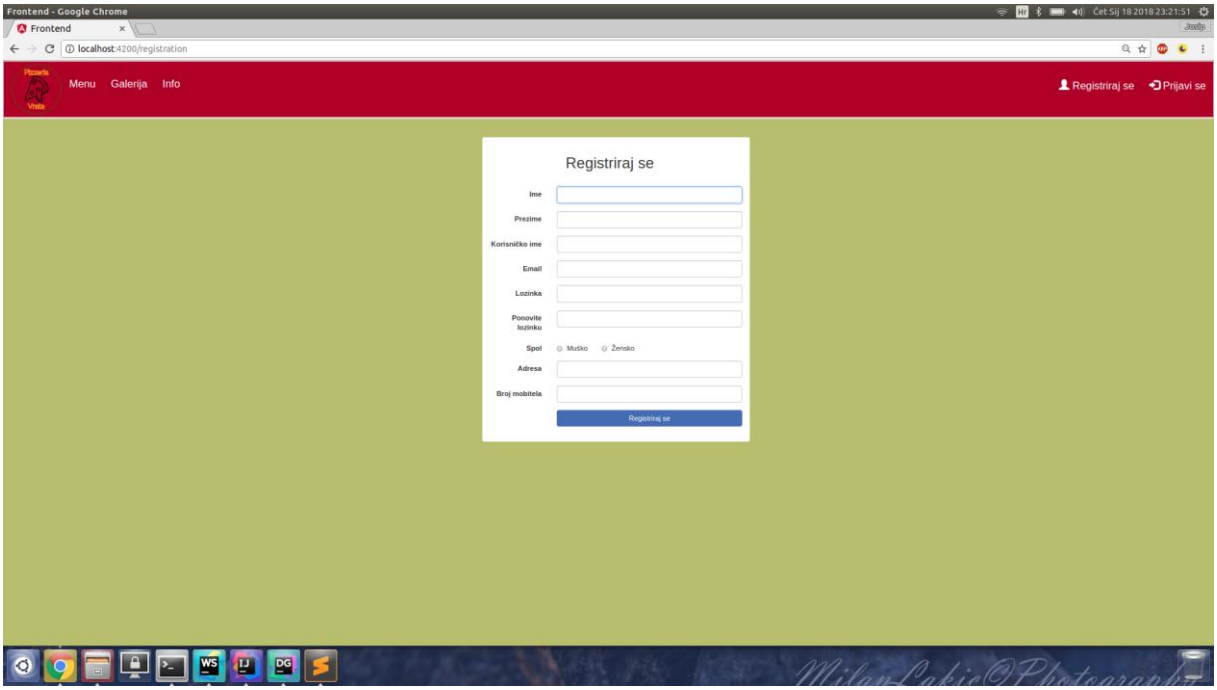
Slika 7.6.1. Prikaz pizza u meniju

U headeru postoje polja „Prijavi se“ i „Registriraj se“ koja omogućavaju anonimnim korisnicima da se prijave u sustav. Za prijavljivanje u sustav dovoljno je unijeti korisničko ime i lozinku. Ukoliko su ta polja kriva (bilo da ne postoje ili su podaci netočni), sustav će dojaviti poruku o grešci.



Slika 7.6.2. Prikaz forme za prijavu

Za registraciju korisnika potrebno je više podataka: ime, prezime, adresa, broj telefona, korisničko ime i lozinka. Također tu postoji mehanizam za provjeru emaila, telefona itd. Uspješnom prijavom ili registracijom korisnici će biti prijavljeni u sustav te će moći pristupati stranicama pizzerije kao prijavljeni korisnici.

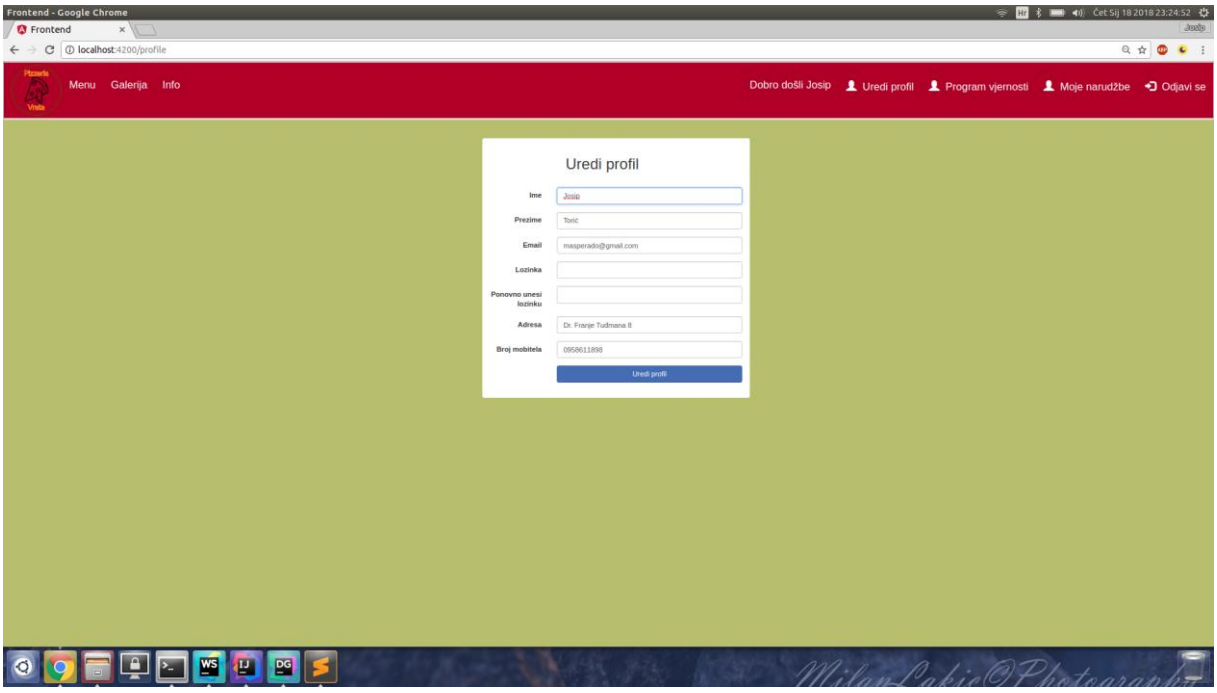


Slika 7.6.3. Prikaz forme za registraciju

Osim toga, za neprijavljene korisnike postoji i dio za galeriju slika te informacije o pizzeriji gdje mogu pogledati druge dodatne sadržaje.

Prijavljeni korisnici/kupci

Kad se kupac prijavi u sustav, dobiva dodatne mogućnosti za istraživanje sadržaja aplikacije. Čim se prijavi u sustav, na korisničkoj stranici dobiva se prikaz profila čije je podatke moguće izmijeniti. Validacija podataka funkcionira isto kao kod registracije te neće biti ponovo objašnjavana. Ukoliko su promjene ispravne, izmijenjeni podaci bit će prikazani na stanici. Kada se klikne na pojedinu pizzu, klikom na gumb „Dodaj u košaricu“ moguće je dodati pojedinu pizzu u košaricu. Također, korisnik može dodati više pizza iste vrste u košaricu. Uz svaku pizzu u košarici postoje zeleni gumb za dodavanje po jedne pizze, bijeli za uklanjanje jedne pizze te crveni za brisanje pizze iz košarice.

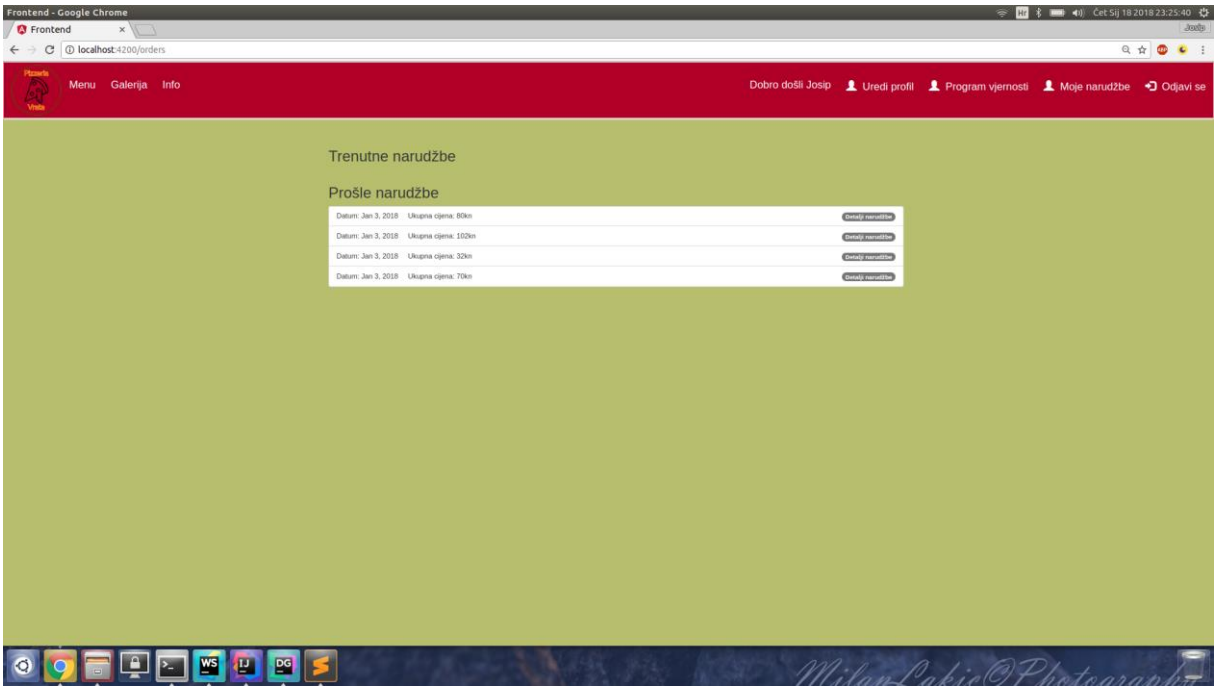


Slika 7.6.4. Uređivanje profila korisnika

Klikom na gumb „Naruči“ narudžba se šalje zaposleniku te se čeka na njegov odgovor da se prihvati.

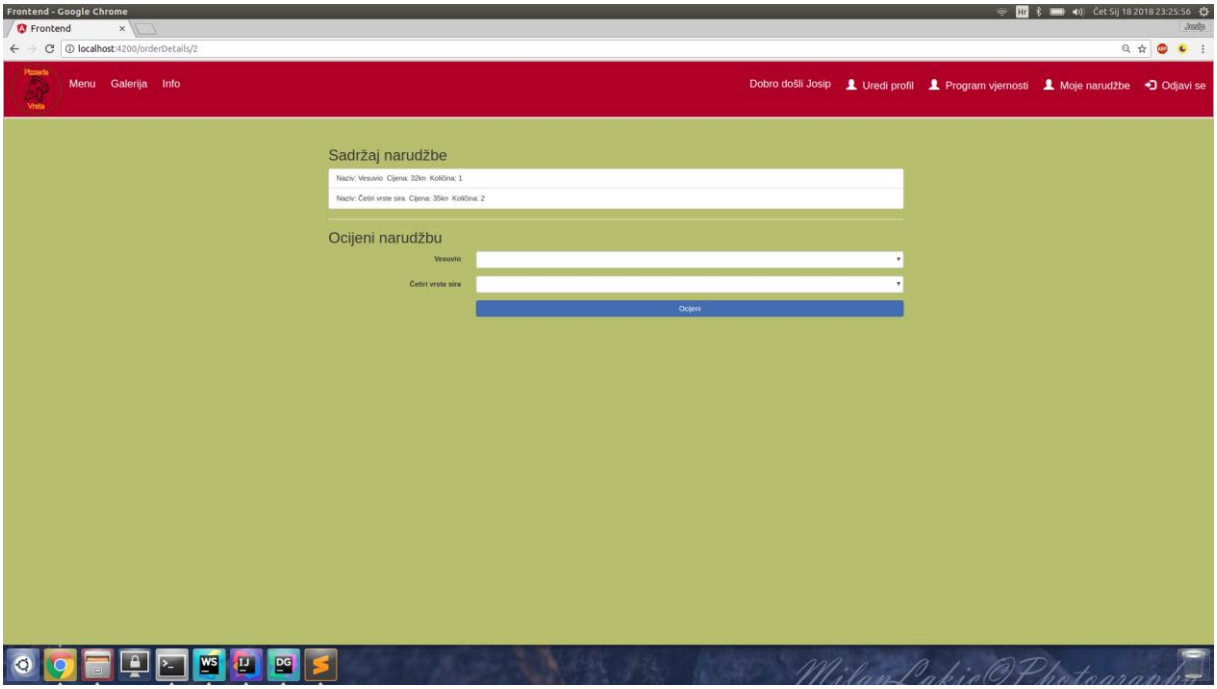
U headeru aplikacije se za prijavljenog korisnika nalaze polja za prikaz programa vjernosti te svih korisničkih narudžbi. Klikom na program vjernosti dobivaju se potrebne informacije o broju pizza koje je potrebno kupiti da se ostvari popust na narudžbu te koliko je korisniku preostalo pizza da dostigne prag.

Klikom na gumb za narudžbe korisnik može vidjeti svoje trenutne i prošle narudžbe. Klikom na pojedinu naružbu moguće je vidjeti koje su pizze naručene i koliko je komada pizze naručeno. Također, status narudžbe je isto tako prikazan.



Slika 7.6.5. Prikaz narudžbi

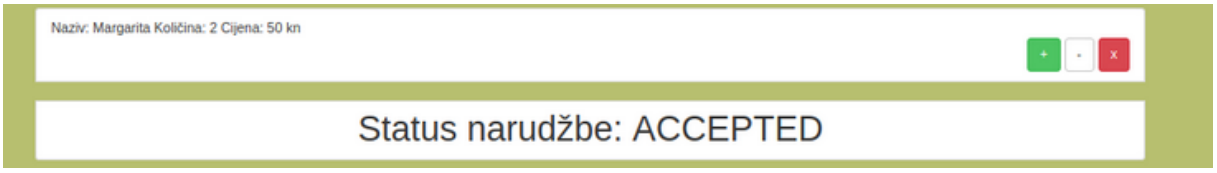
U prikazu pojedine narudžbe moguće je unijeti ocjenu pojedine pizze. Klikom na gumb „Ocijeni“, ocjene za ocijenjene pizze se upisuju u bazu te su ocjene zatvorene za tu narudžbu. Ocjene koje je moguće unijeti su u rasponu od 1 do 5.



Slika 7.6.6. Mogućnost ocjenjivanja narudžbe

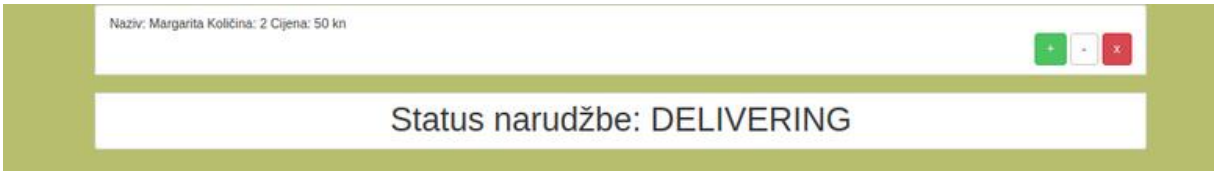
Zaposlenici

Zaposlenici upravljaju sustavom narudžbi. Kada se zaposlenik prijavi u sustav, na početnoj stranici moguće je vidjeti sve narudžbe za koje je taj zaposlenik zadužen. Kada narudžba stigne u sustav ona prelazi u status „Ordered“. Čim zaposlenik prihvati narudžbu, narudžba ulazi u status „Accepted“, što znači da je zaposlenik uspješno zaprimio narudžbu.



Slika 7.6.7. Promjena statusa narudžbe u "Accepted"

Kad je pizza pripremljena, zaposlenik može stisnuti gumb da narudžba prijeđe u status „Delivering“, čime je označio da je narudžba u isporuci.



Slika 7.6.8. Promjena statusa narudžbe u "Delivering"

Konačno, zaposlenik može označiti je li narudžba plaćena ili ne čime narudžba može prijeći u status „Paid“ (ako se dostavljač vrati s novcima kupca) i „Not paid“ (u slučaju poteškoća s kupcem).

Ako ima previše posla, klikom na gumb „Blokiraj“ zaposlenik blokira narudžbe te je naručivanje za korisnike zatvoreno, isto tako klikom na gumb „Otvori narudžbe“ mogućnost naručivanja se omogućava.

Administratori

Administratori imaju punu kontrolu nad korisnicima, zaposlenicima i pizzama.

Na svom sučelju, kad se prijavi u sustav, administrator ima uvid u sva izvješća pizzerije. Izvješća se generiraju periodično te administrator ima uvid u tjedna, mjesečna, kvartarna, polugodišnja i godišnja izvješća. Sva izvješća dostupna su u PDF i XLS formatu. U headeru administratora postoje sljedeće opcije: pizze, korisnici i zaposlenici.

Klikom na polje „Pizze“, administratoru se otvara popis svih pizza, s gumbom „Dodaj“ pizzu gdje se otvara forma za dodavanje nove pizze. Dodavanje pizze sastoji se od unosa naziva, url-a pizze, cijene, opisa te potrebnih sastojaka. Ukoliko su sva polja ispravna, pizza će biti uspješno dodana.

Klikom na pojedinu pizzu, moguće ju je izmijeniti, ili klikom na gumb „Povuci iz ponude“ moguće ju je povući iz ponude. Pizza se unosi ponovo u ponudu klikom na „Dodaj u ponudu“. U opcijama „Zaposlenici“ i „Korisnici“, moguće je na isti način dodavati, izmjenjivati i brisati korisnike te zaposlenike. Razlika je samo što će kod brisanja zaposlenika i korisnika oni biti trajno izbrisani iz baze.

8. Zaključak i budući rad

Zadatak projekta je razviti web aplikaciju za upravljanje rada pizzerijom, preciznije govoreći upravljanje online naručivanje pizza iz pizzerije. Realizacija projekta je podijeljena u dvije faze.

U prvoj fazi obavljene su sve formalnosti poput formiranja i upoznavanja tima, iznošenja ideja te podjele uloga pojedinim članovima tima,napisana je većina dokumentacije. Općenito, prva faza je priprema za konačno ostvarenje računalnog sustava. Mogli bismo reći da je u njoj napravljen temelj cijelog sustava. Razrađeni su funkcionalni i nefunkcionalni zahtjevi,te su dodani razni obrasci, dijagrami i modeli poput obrazaca uporabe, sekvencijskih dijagrama, dijagrama razreda, dijagram objekata, itd. Nadamo se da će nam ova prva faza biti od velike pomoći kada krenemo s implementacijom. Također se mogu očekivati i neke izmjene oko dijagrama razreda i objekata budući da će projekt biti rađen u Spring Frameworku s mnogim Controllerima, modelima, service i repository klasama i dr. Puno posla je još pred nama,moramo napraviti implementaciju,testirati sustav,napisati upute,dodati još neke obrasce i sve to uz ostale obaveze na fakultetu,ali smo optimistični i vjerujemo da ćemo sve stići u zadanim rokovima.

9. Popis literature

Popisati sve reference i literaturu koja je pomogla pri ostvarivanju projekta.

[1] Oblikovanje programske potpore, FER ZEMRIS, http://www.fer.hr/predmet/opp

[2] Oblikovanje programske potpore, FER ZEMRIS, http://www.zemris.fer.hr/predmeti/opp

[3] I. Sommerville, „Software engineering“, 8th ed, Addison Wesley, 2007.

[4] T.C.Lethbridge, R.Langaniere, „Object-Oriented Software Engineering“, 2nd ed. McGraw-Hill, 2005.

[5] Software engineering ,Rutgers University, http://www.ece.rutgers.edu/~marsic/Teaching/SE

[6] I. Marsic, „Software engineering book“, Department of Electrical and Computer Engineering, Rutgers University, http://www.ece.rutgers.edu/~marsic/books/SE

[7] Concepts: Requirements, http://www.upedu.org/upedu/process/gcncpt/co_req.htm

[8] UML 2 Class Diagram Guidelines, http://www.agilemodeling.com/style/classDiagram.htm

[9] Domain Class Diagram Modeling Standards and Guidelines, http://www.bced.gov.bc.ca/imb/downloads/classdiagramstandards.pdf

[10] Astah Community, http://astah.net/editions/community/

Dodatak A: Indeks (slika, dijagrama, tablica, ispisa kôda)

Dodatak B: Dnevnik sastajanja

SASTANAK 1: 20.10.2017

Prisutni: Silvija Grgić, Marija Frković, Josip Mrđen, Josip Torić, Luka Kudra, Ivan Crnomarković

Sažetak: Međusobno upoznavanje članova tima i početna analiza zadatka.

SASTANAK 2: 27.10.2017

Prisutni: Silvija Grgić, Marija Frković, Josip Mrđen, Josip Torić, Luka Kudra, Ivan Crnomarković

Sažetak: Podjela izrade dokumentacije projekta među članovima.

SASTANAK 3: 3.11.2017

Prisutni: Silvija Grgić, Marija Frković, Josip Mrđen, Josip Torić, Luka Kudra, Ivan Crnomarković

Sažetak: Dodatna analiza zadatka te podjela tima na dijelove koji će izrađivati backend i frontend.

SASTANAK 4: 10.11.2017

Prisutni: Silvija Grgić, Marija Frković, Josip Mrđen, Josip Torić, Luka Kudra, Ivan Crnomarković

Sažetak: Analiza dosad napravljene dokumentacije. Rasprava o optimalnom modelu za backend projekta.

SASTANAK 5: 8.1.2018.

Prisutni: Silvija Grgić, Marija Frković, Josip Mrđen, Josip Torić, Luka Kudra, Ivan Crnomarković

Sažetak: Raspodjela preostalog dijela posla za projekt, checkpoint za daljnje implementiranje frontenda budući da je backend već gotov.

SASTANAK 6: 14.1.2018.

Prisutni: Silvija Grgić, Marija Frković, Josip Mrđen, Josip Torić, Luka Kudra, Ivan Crnomarković

Sažetak: Dorada dokumentacije i daljnja izgradnja frontenda.

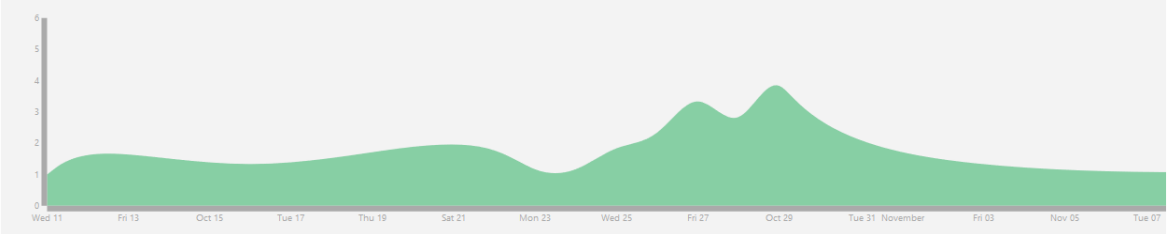
Dodatak C: Prikaz aktivnosti grupe

Popis aktivnosti	Josip Torić	Josip Mrđen	Marija Frković	Silvija Grgić	Ivan Crnomarković	Luka Kudra
Upravljanje projektom	50%	50%				
Opis projektnog zadatka	100%					
Rječnik pojmova	100%					
Opis funkcionalnih zahtjeva		10%	25%	25%	20%	20%
Opis ostalih zahtjeva					50%	50%
Arhitektura i dizajn sustava	80%	20%				
Svrha, opći prioriteti i skica sustava	50%	50%				
Dijagram razreda s opisom	5%	65%	5%	5%	10%	10%
Dijagram objekata		50%			25%	25%
Ostali UML dijagrami			25%	25%	25%	25%
Implementacija i korisničko sučelje	50%	50%				
Dijagram razmještaja	50%	50%				
Korištene tehnologije i alati	50%	50%				
Isječak programskog kôda	50%	50%				
Ispitivanje programskog rješenja	50%	50%				
Upute za instalaciju	50%	50%				
Korisničke upute			25%	25%	25%	25%
Plan rada			50%	50%		
Pregled rada i stanje			50%	50%		

ostvarenja						
Zaključak i budući rad			50%	50%		
Popis literature						100%
Dodaci	10%	10%	20%	20%	20%	20%
Indeks					100%	
Dnevnik sastajanja					50%	50%

Napomena: Doprinose u aktivnostima treba navesti u postocima po članovima grupe. Zbroj postotaka u svakom retku treba biti 100%.

Pregled pohrana kroz vrijeme trajanja projekta:



Dodatak D: Plan rada / Pregled rada i stanje ostvarenja

Za Rev. 2 planiramo nastaviti tempom kojim smo i započeli te dovršiti sve UML dijagrame. Čeka nas dosta posla budući da će trebati i implementirati dokumentirano ali mislimo da nećemo imati problema oko toga. Osim uspjeha na projektnom zadatku, plan je i da svi ponešto naučimo o svakom dijelu dokumentiranja i implementiranja koda a to je možda i najteža zadaća budući da nas je šestero u timu. Pokušavamo se konzultirati što više da svi budemo u toku oko svih faza projekta.