

Programsko inženjerstvo

Ak. god. 2022./2023.

# Hotelsko poslovanje

Dokumentacija, Rev. 2.

Grupa: *ProgiBatina*

Voditelj: *Marijan Lovrić*

Datum predaje: *13. siječnja 2023.*

Nastavnik: *Miljenko Krhen*

# Sadržaj

<b>1 Dnevnik promjena dokumentacije</b>	<b>3</b>
<b>2 Opis projektnog zadatka</b>	<b>5</b>
<b>3 Specifikacija programske potpore</b>	<b>7</b>
3.1 Funkcionalni zahtjevi . . . . .	7
3.1.1 Obrasci uporabe . . . . .	9
3.1.2 Sekvencijski dijagrami . . . . .	19
3.2 Ostali zahtjevi . . . . .	22
<b>4 Arhitektura i dizajn sustava</b>	<b>23</b>
4.1 Baza podataka . . . . .	24
4.1.1 Opis tablica . . . . .	24
4.1.2 Dijagram baze podataka . . . . .	29
4.2 Dijagram razreda . . . . .	30
4.3 Dijagram stanja . . . . .	35
4.4 Dijagram aktivnosti . . . . .	36
4.5 Dijagram komponenti . . . . .	38
<b>5 Implementacija i korisničko sučelje</b>	<b>40</b>
5.1 Korištene tehnologije i alati . . . . .	40
5.2 Ispitivanje programskog rješenja . . . . .	41
5.2.1 Ispitivanje komponenti . . . . .	41
5.2.2 Ispitivanje sustava . . . . .	46
5.3 Dijagram razmještaja . . . . .	48
5.4 Upute za puštanje u pogon . . . . .	49
<b>6 Zaključak i budući rad</b>	<b>52</b>
<b>Popis literature</b>	<b>53</b>
<b>Indeks slika i dijagonama</b>	<b>54</b>



# 1. Dnevnik promjena dokumentacije

*Kontinuirano osvježavanje*

Rev.	Opis promjene/dodatka	Autori	Datum
0.1	Napravljen predložak.	*	15.10.2022.
0.2	Opis projektnog zadatka.	*	21.10.2022.
0.3	Dodani funkcionalni zahtjevi	*	25.10.2022.
0.4	Prvi dio dijagrama uporabe	*	28.10.2022.
0.5	Drugi dio dijagrama uporabe	*	31.10.2022.
0.6	Dovršeni dijagrami uporabe i dodani sekvencijski dijagrami	*	3.11.2022.
0.7	Napisana arhitektura sustava	*	08.11.2022.
0.8	Nadopunjena baza podataka i dijagram razreda	*	15.11.2022.
0.9	Provjeren pravopis i dovršena dokumentacija	*	17.11.2022.
<b>1.0</b>	Verzija samo s bitnim dijelovima za 1. ciklus	*	18.11.2022.
1.1	Popravljena baza i dijagrami.	*	7.12.2022.
1.2	Dodan dijagram stanja	*	12.12.2022.
1.3	Dodan dijagram aktivnosti	*	20.12.2022.
1.4	Dovršeni obrasci uporabe	*	2.1.2023.
1.5	Dodan dijagram komponenti	*	8.1.2023.
1.6	Napisane korištene tehnologije i alati	*	11.1.2023.

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

<b>Rev.</b>	<b>Opis promjene/dodataka</b>	<b>Autori</b>	<b>Datum</b>
1.7	Dodano ispitivanje programskog rješenja	*	12.1.2023.
1.8	Dodana tablica aktivnosti, dijagram razmještaja	*	13.1.2023.
1.9	Napisan zaključak i provjeren pravopis	*	13.1.2023.
<b>2.0</b>	Finalna verzija dokumentacije	*	13.1.2023.

## 2. Opis projektnog zadatka

Cilj ovog projekta je razviti programsku podršku za poduzeće „Kod nas je najljepše“ koja će služiti za rezerviranje soba, pregled gostiju, smještajnih jedinica, itd. U ponudi hotela su sobe i apartmani, ali i dvije dvorane za posebne potrebe kao što su radni sastanci, kongresi, vjenčanja, proslave i sl.

U ponudi smještaja je 15 jednokrevetnih soba, 45 dvokrevetnih, 10 obiteljskih soba i 10 obiteljskih apartmana. Standardno se nudi usluga noćenja s doručkom, a kao posebnu uslugu moguće je dobiti ručak i/ili večeru. Prijava u hotel moguća je nakon 14:00 sati, a odjava je obavezna do 11:00 sati. Najmanje trajanje rezervacije i boravka u hotelu je 3 dana.

Sustav omogućuje praćenje rezervacija, grupnih rezervacija i zauzeća smještajnih jedinica, kao i potreba za posebnim uslugama po pojedinom danu. Također, sustav prati stanje oboljelih i trend o broju oboljelih od bolesti Covid u državama iz koje dolaze pojedini gosti.

Proces registracije gosta započinje rezervacijom jednog od smještajnih kapaciteta putem telefona, elektroničkom poštom ili direktnim kontaktom. Rezervaciju je moguće napraviti pojedinačno ili grupno. Prvi korak predviđa generičku rezervaciju smještajnog kapaciteta, bez osobnih podataka. Drugi korak predviđa unos stvarnih podataka o gostu (ime i prezime, adresa, država iz koje dolazi, OIB, broj telefona, zahtjev za posebnom uslugom), i to u trenutku kada se gost stvarno prijava na recepciji hotela.

Prilikom registracije gost dobiva korisnički račun za sustav preko kojeg može vidjeti u kojoj je smještajnoj jedinici smješten, i što je sve uključeno u njegovu rezervaciju.

Primjer sličnih rješenja su aplikacije za rezervaciju smještaja poput Booking, Airbnb i sl. Prednost vlastite aplikacije je ta što ne treba plaćati dio novca tim aplikacijama te što je ova aplikacija rađena specifično za ovaj hotel i nudi neke opcije specifične za njega.

Korisnici koji bi mogli biti zainteresirani za ostvareno rješenje

Aplikacija bi bila najkorisnija recepcionarima, administratorima(poslovođama) i sobaricama jer bi mogli jednostavnije međusobno razmjenjivati podatke i ažurnije obavljati zahtjeve gostiju. Također, gostima bi bilo jednostavnije vidjeti ponuđene usluge i vlasnik bi mogao lakše pratiti sveukupno.

- **Vlasnik hotela** - ima mogućnost unosa i ažuriranja osnovnih podataka o hotelu, broju, vrsti i kapacitetu smještajnih jedinica. Omogućen mu je pregled zauzeća smještajnih jedinica te pregled ostalih podataka na zahtjev. Također, vlasnik određuje glavnog administratora sustava.
- **Administrator** - određuje ga vlasnik prilikom unosa podataka o hotelu ili kasnije po potrebi. Administrator unosi i ažurira podatke o djelatnicima koji imaju pristup sustavu (djelatnici na recepciji i sobarice) i dodjeljuje im razine ovlasti koje odgovaraju potrebama za njihovu poziciju. Po potrebi može i upisivati podatke o rezervacijama smještaja.
- **Djelatnici na recepciji** - nakon dodjele ovlasti od strane administratora imaju mogućnost upisa rezervacije smještajnih jedinica kao i ažuriranje/uklanjanja istih zbog otkazivanja. Također, prilikom registracije u sustav upisuju podatke o gostima hotela i zapisuje sve zahtjeve nakon dolaska. Sustav im omogućuje i pregled podatka o broju zaraženih u državi iz koje dolazi gost.
- **Sobarice** - nakon dodjele ovlasti od strane administratora vode evidenciju o pospremljenim sobama i potrošenim elementima iz mini bara za pojedinu sobu. Nakon pospremanja sobe bilježe je li nešto u sobi pokvareno, razbijeno i sl. te nakon završnog spremanja sobe po odlasku gosta, postavljaju posebnu obavijest prema recepciji da je određena soba spremna za prihvatanje novog gosta.
- **Gosti** - prilikom prijave na recepciju hotela, nakon davanja osobnih podataka i rezervacije smještaja, gost dobiva svoj korisnički račun za sustav preko kojeg može vidjeti podatke o svom smještaju. Omogućen mu je pregled u kojoj se jedinici nalazi i što je sve uključeno u njegovu rezervaciju. Također, sustav mu omogućava unijeti dodatne zahtjeve za uslugama van onoga što je unaprijed rezervirano.

## 3. Specifikacija programske potpore

### 3.1 Funkcionalni zahtjevi

Dionici:

1. Naručitelj sustava (vlasnik hotela).
2. Osoblje hotela (sobarice, recepcionisti).
3. Administrator sustava.
4. Razvojni tim.
5. Gosti hotela.
6. Zakupci rezervacije dvorana.

Aktori i njihovi funkcionalni zahtjevi:

1. Vlasnik (inicijator) može:
  - (a) unos podataka o hotelu i smještajnim jedinicama.
  - (b) određivanje administratora sustava.
  - (c) pregled zauzeća.
2. Administrator sustava (inicijator) može:
  - (a) unos podataka o djelatnicima.
  - (b) dodjela razina ovlasti zaposlenicima.
  - (c) unos podataka o rezervacijama.

**3. Djelatnik na recepciji (inicijator) može:**

- (a) unos rezervacija.
  - i. unos gostiju.
  - ii. unos zahtjeva.
  - iii. pregled podataka gosta koji stvara rezervaciju.
  - iv. pregled podataka o broju oboljelih od COVID-a u zemlji podrijetla gosta.
- (b) izdavanje računa.
  - i. ispis računa.
  - ii. slanje računa e-poštom.
- (c) unos rezervacija dvorana.
  - i. rezervacija pojedinačnih dvorana.
  - ii. spajanje dvije dvorane.

**4. Sobarica (inicijator) može:**

- (a) podnošenje izvještaja o pospremanju sobe.
- (b) unos količine potrošnog materijala u minibaru.
- (c) obavijest o spremnosti sobe za prihvrat sljedećeg gosta.

**5. Sustav za praćenje broja oboljelih od COVID-a(sudionik) može:**

- (a) pružati informacije o broju oboljelih od COVID-a u državi podrijetla gosta koji radi rezervaciju.

**6. sustav koji pruža informaciju o vremenskoj prognozi(sudionik) može:**

- (a) pružati informacije o vremenskim prilikama na lokaciji hotela.

**7. Gost hotela(inicijator) može:**

- (a) podnositi zahtjeve za posebnim uslugama.
- (b) pregledati podataka o vremenskim prilikama na lokaciji hotela.
- (c) otkazati zahtjeve za uslugama.

### 3.1.1 Obrasci uporabe

#### UC1 - Pregled početne stranice

- **Glavni sudionik:** Korisnik
- **Cilj:** Pregled
- **Sudionici:** -
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
  1. Korisnik otvara stranicu početnu stranicu i dobiva kratki pregled s općom "Prijava"

#### UC2 - Prijava korisnika

- **Glavni sudionik:** Korisnik
- **Cilj:** Prijaviti se u sustav
- **Sudionici:** Baza podataka
- **Preduvjet:** Registriran korisnik
- **Opis osnovnog tijeka:**
  1. Unos korisničkog imena i lozinke
  2. Potvrda o ispravnosti unesenih podataka
  3. Pristup korisničkim funkcijama
- **Opis mogućih odstupanja:**
  - 2.a Neispravno korisničko ime/lozinka
    1. Sustav obaveštava korisnika o neuspjelom upisu

#### UC3 - Izvještaj

- **Glavni sudionik:** Sobarica/Spremačica
- **Cilj:** Obavit Izvještaj
- **Sudionici:** Baza podataka
- **Preduvjet:** Soba koja se čisti je zauzeta
- **Opis osnovnog tijeka:**
  1. Sobarica/Spremačica u aplikaciji odabire sobu
  2. Odabire opciju "Soba pospremljena"
  3. Upisuje količinu iskorištenih artikala mini-bar-a
  4. U slučaju zadnjeg dana boravka, dojavljuje recepciji da je soba spremna za prihvat novog gosta

### UC4 - Generička rezervacija

- **Glavni sudionik:** Djelatnik na recepciji
- **Cilj:** Rezervacija smještaja
- **Sudionici:** Baza podataka i gost
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
  1. Gost putem telefona, e-maila, direktnim kontaktom ili preko turističke agencije započinje rezervaciju.
  2. Sustav ponudi slobodne sobe za željeni termin
  3. Djelatnik na recepciji unosi generičku rezervaciju bez osobnih podataka
- **Opis mogućih odstupanja:**
  - 2.a Ne postoje slobodni kapaciteti
    1. Promjena datuma rezervacije

### UC5 - Registracija

- **Glavni sudionik:** Djelatnik na recepciji
- **Cilj:** Registracija gosta
- **Sudionici:** Baza podataka i gost
- **Preduvjet:** Rezervacija
- **Opis osnovnog tijeka:**
  1. Djelatnik na recepciji unosi stvarne podatke o gostu u trenutku dolaska
  2. Djelatnik na recepciji dobiva podatak o broju oboljelih od Covida
  3. Dodjela korisničkog računa gostu
- **Opis mogućih odstupanja:**

Nedolazak gosta

  1. Djelatnik briše rezervaciju

### UC6 - Rezervacija dvorane

- **Glavni sudionik:** Djelatnik na recepciji
- **Cilj:** Rezervacija dvorane
- **Sudionici:** Baza podataka i neprijavljen korisnik
- **Preduvjet:** Slobodan termin
- **Opis osnovnog tijeka:**
  1. Djelatnik na recepciji odabire dvoranu
  2. Djelatnik odabire datum rezervacije
  3. Prikaže se stranica sa slobodnim terminima
  4. Odabire termin i potvrđuje rezervaciju
- **Opis mogućih odstupanja:**
  - 3.a Nema slobodnih termina
    1. Sustav nudi opciju mijenjanja datuma

### UC7 - Pregled zauzeća smještajnih jedinica

- **Glavni sudionik:** Djelatnik na recepciji
- **Cilj:** Pregled zauzeća
- **Sudionici:** Baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
  1. Djelatnik odabire opciju pregleda u smještajnih jedinicama
  2. Sustav dohvaća podatke o stvarnom zauzeću, rezervacijama i otkazanim rezervacijama

### UC8 - Pregled liste gostiju

- **Glavni sudionik:** Djelatnik na recepciji
- **Cilj:** Pregled gostiju
- **Sudionici:** Baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
  1. Djelatnik odabire opciju pregleda liste gostiju
  2. Djelatnik na recepciji odabire vrijeme pregleda
  3. Sustav dohvaća podatke o gostima i rezervacijama za odabranou vrijeme

### UC9 - Izdavanje računa

- **Glavni sudionik:** Djelatnik na recepciji
- **Cilj:** Izdavanje računa
- **Sudionici:** Baza podataka
- **Preduvjet:** Postojeći račun
- **Opis osnovnog tijeka:**
  1. Djelatnik na recepciji odabire sobu
  2. Djelatnik odabire opciju "Izdaj račun"
  3. Ponuditi opciju "Pošalji račun putem elektroničke pošte" ili "Ispiši račun"

### UC10 - Pregled rezervacije

- **Glavni sudionik:** Gost
- **Cilj:** Pregledati rezervaciju
- **Sudionici:** Baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
  1. Gost odabire opciju "Moje rezervacija"
  2. Na zaslonu se ispisuju podaci o rezervaciji

### UC11 - Zahtjev za dodatnim uslugama

- **Glavni sudionik:** Gost
- **Cilj:** Realizirati dodatnu uslugu
- **Sudionici:** Baza podataka
- **Preduvjet:** Gost je prijavljen
- **Opis osnovnog tijeka:**
  1. Gost odabire vrstu usluge
  2. Gost odabire broj osoba za koje se realizira usluga
  3. Sustav obavještava gosta o realizaciji usluge
- **Opis mogućih odstupanja:**
  - 1.a Nedostatak usluge
    1. Sustav obavještava gosta o nedostatku usluge
  - 2.a Prevelik broj sudionika usluge
    1. Sustav obavještava gosta o prevelikom broju prijavljenih
    2. Nudi mu opciju ponovnog odabira broja sudionika

### UC12 - Brisanje usluge

- **Glavni sudionik:** Gost
- **Cilj:** Otkazivanje usluge
- **Sudionici:** Baza podataka
- **Preduvjet:** Usluga aktivirana
- **Opis osnovnog tijeka:**
  1. Gost isključuje uslugu
  2. Provjerava se je li uslugu moguće isključiti
  3. Usluga se isključuje
  4. Sustav javlja korisniku da je usluga isključena
- **Opis mogućih odstupanja:**
  - 2.a Uslugu nije moguće isključiti jer nije obavljeno pravovremeno
    1. Sustav javlja korisniku da uslugu nije moguće isključiti

### UC13 - Unos podataka o hotelu

- **Glavni sudionik:** Vlasnik
- **Cilj:** Unositi podatke o hotelu
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik s ulogom Vlasnik
- **Opis osnovnog tijeka:**
  1. Unos podataka o hotelu

### UC14 - Unos podataka o smještajnim jedinicama

- **Glavni sudionik:** Vlasnik
- **Cilj:** Uređivati podatke o hotelu
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik s ulogom Vlasnik
- **Opis osnovnog tijeka:**
  1. Unos podataka o smještajnim jedinicama

**UC15 - Određivanje administratora**

- **Glavni sudionik:** Vlasnik
- **Cilj:** Postaviti admina
- **Sudionici:** Baza podataka
- **Preduvjet:** Uloga vlasnika
- **Opis osnovnog tijeka:**
  1. Vlasnik odabire zaposlenika
  2. Vlasnik mijenja ulogu zaposlenika

**UC16 - Pregled liste računa**

- **Glavni sudionik:** Vlasnik
- **Cilj:** Pregled
- **Sudionici:** Baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
  1. Vlasnik odabire opciju pregleda liste računa
  2. Sustav dohvaća račune iz baze podataka

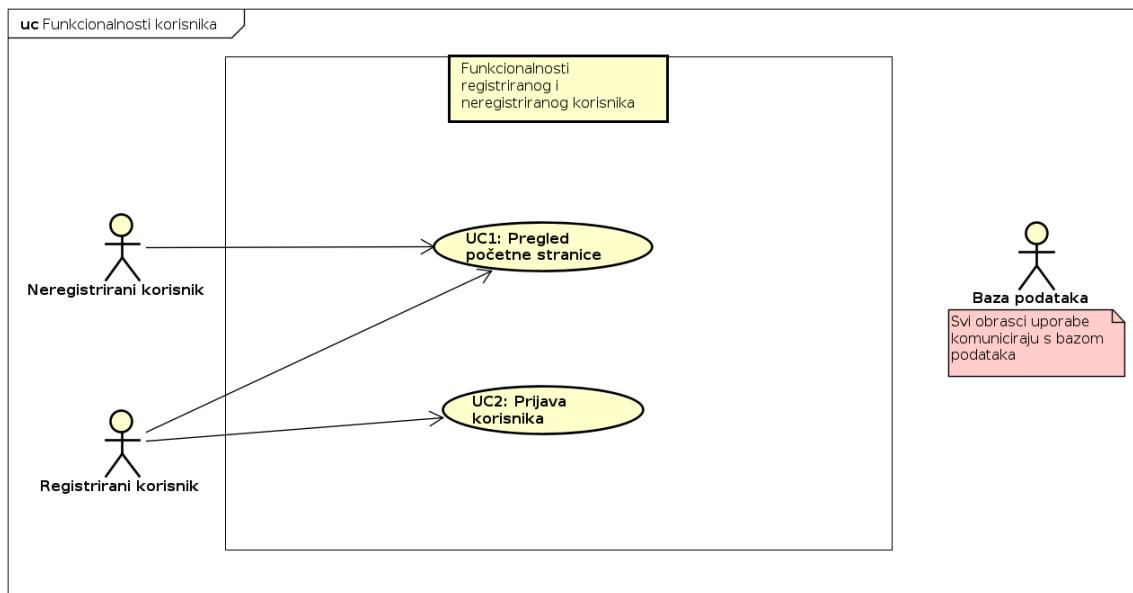
**UC17 - Izvještaj analize smještajnih jedinica**

- **Glavni sudionik:** Vlasnik
- **Cilj:** Pregled
- **Sudionici:** Baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
  1. Vlasnik odabire opciju analiza smještajnih jedinica
  2. Sustav dohvaća statistiku uporabe jedinica

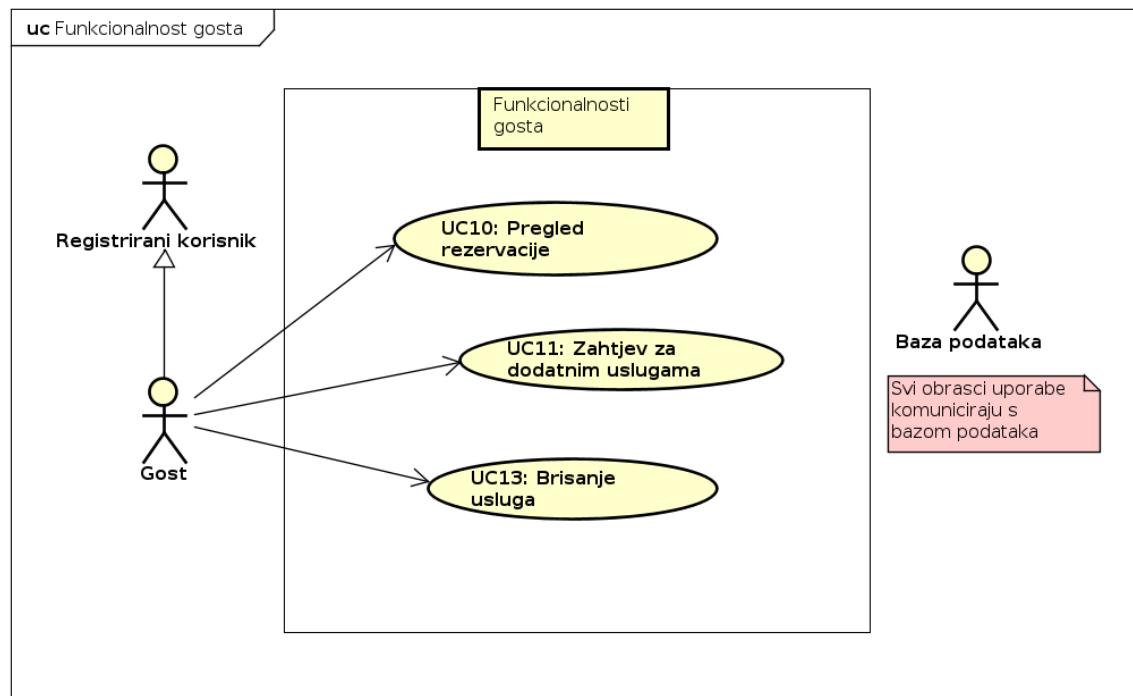
### UC18 - Unos podataka radnika

- **Glavni sudionik:** Administrator sustava
- **Cilj:** Unijeti podatke o novom radniku
- **Sudionici:** Baza podataka
- **Preduvjet:** Uloga administratora
- **Opis osnovnog tijeka:**
  1. Odabire unos podataka o radniku
  2. Unos podataka o radniku
  3. Provjera ispravnosti podataka
  4. Spremanje podataka u bazu
- **Opis mogućih odstupanja:**
  - 3.a Nisu uneseni svi potrebni podaci ili podaci ne odgovaraju uvjetima za određeni podatak
    1. Sustav obavještava korisnika o neuspjelom dodavanju radnika u bazu i ispisuje koji je problem nastao prilikom unosa

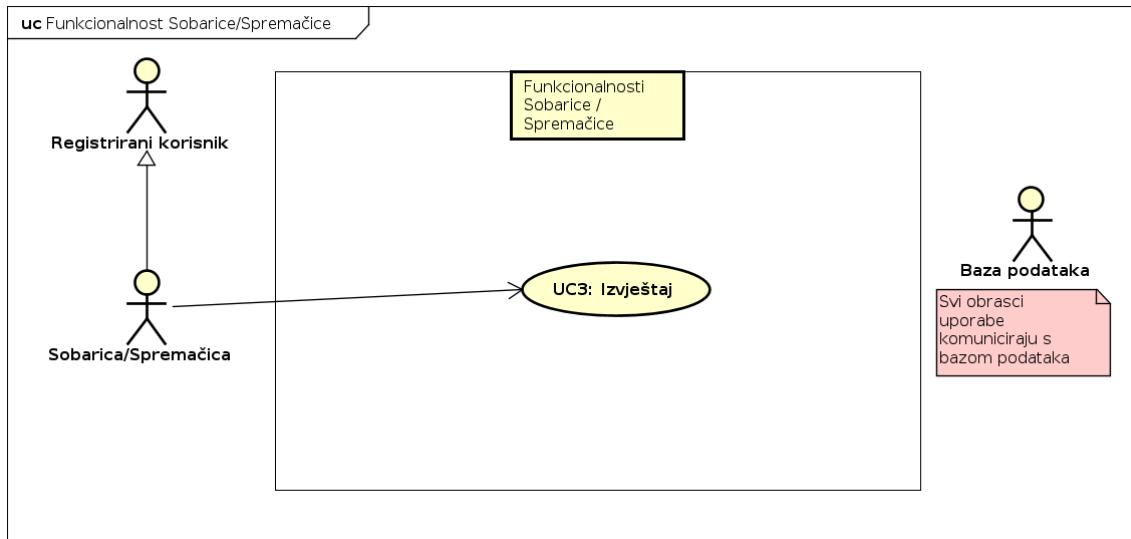
## Dijagrami obrazaca uporabe



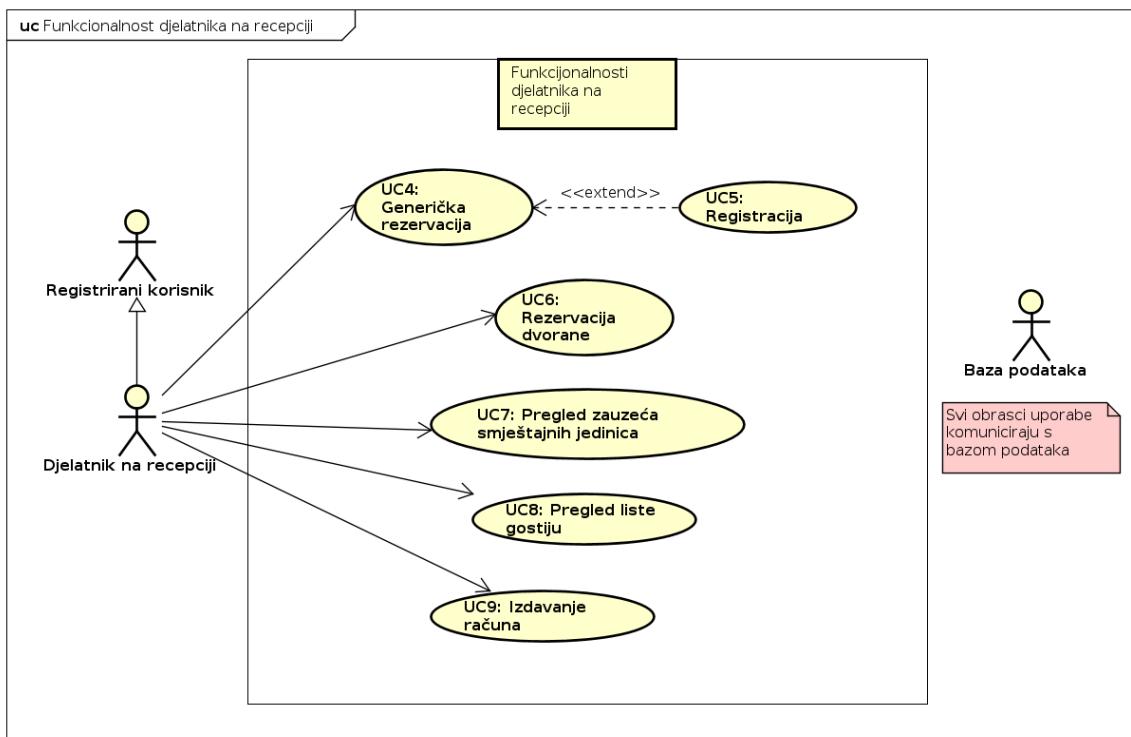
Slika 3.1: Korisnik



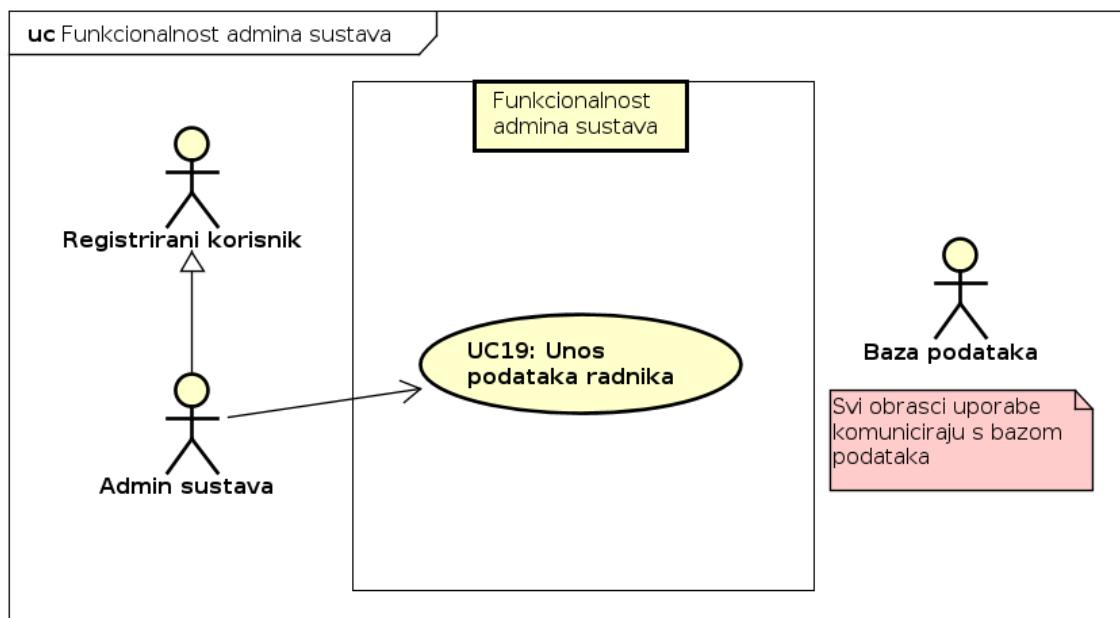
Slika 3.2: Gost



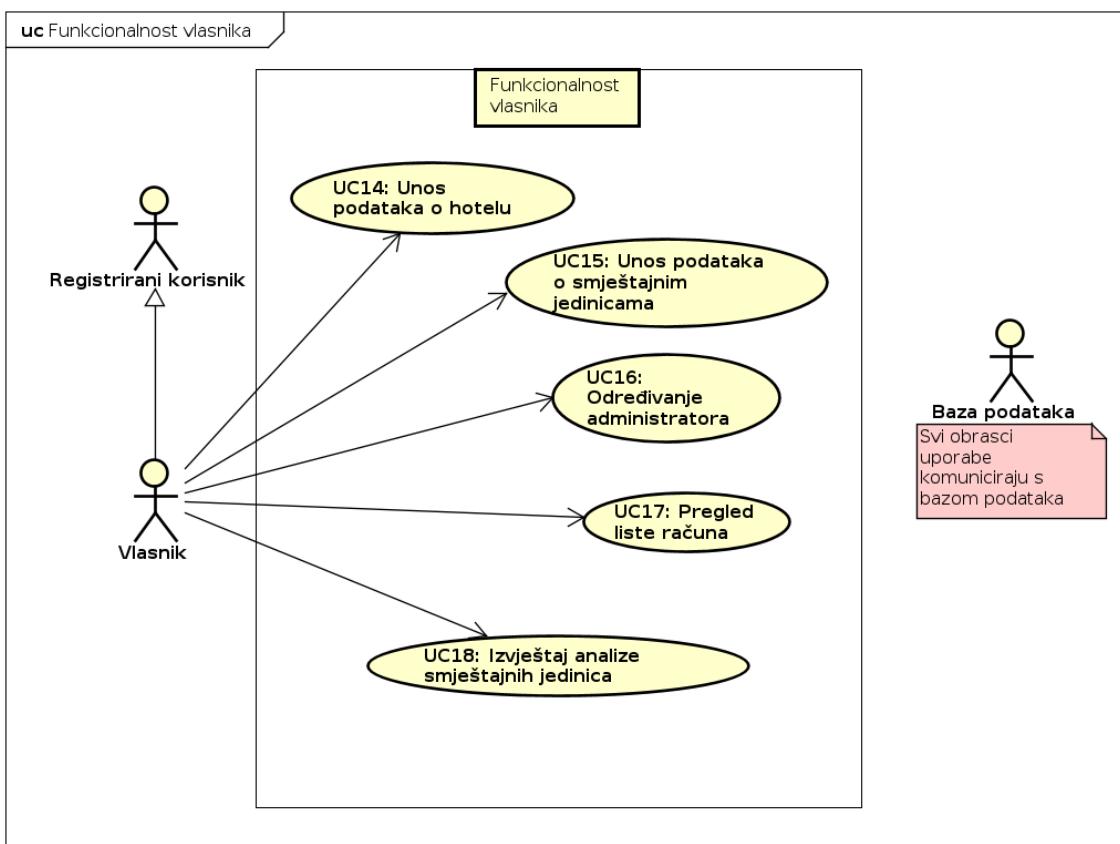
Slika 3.3: Sobarica/Spremačica



Slika 3.4: Djelatnik na recepciji



Slika 3.5: Administrator

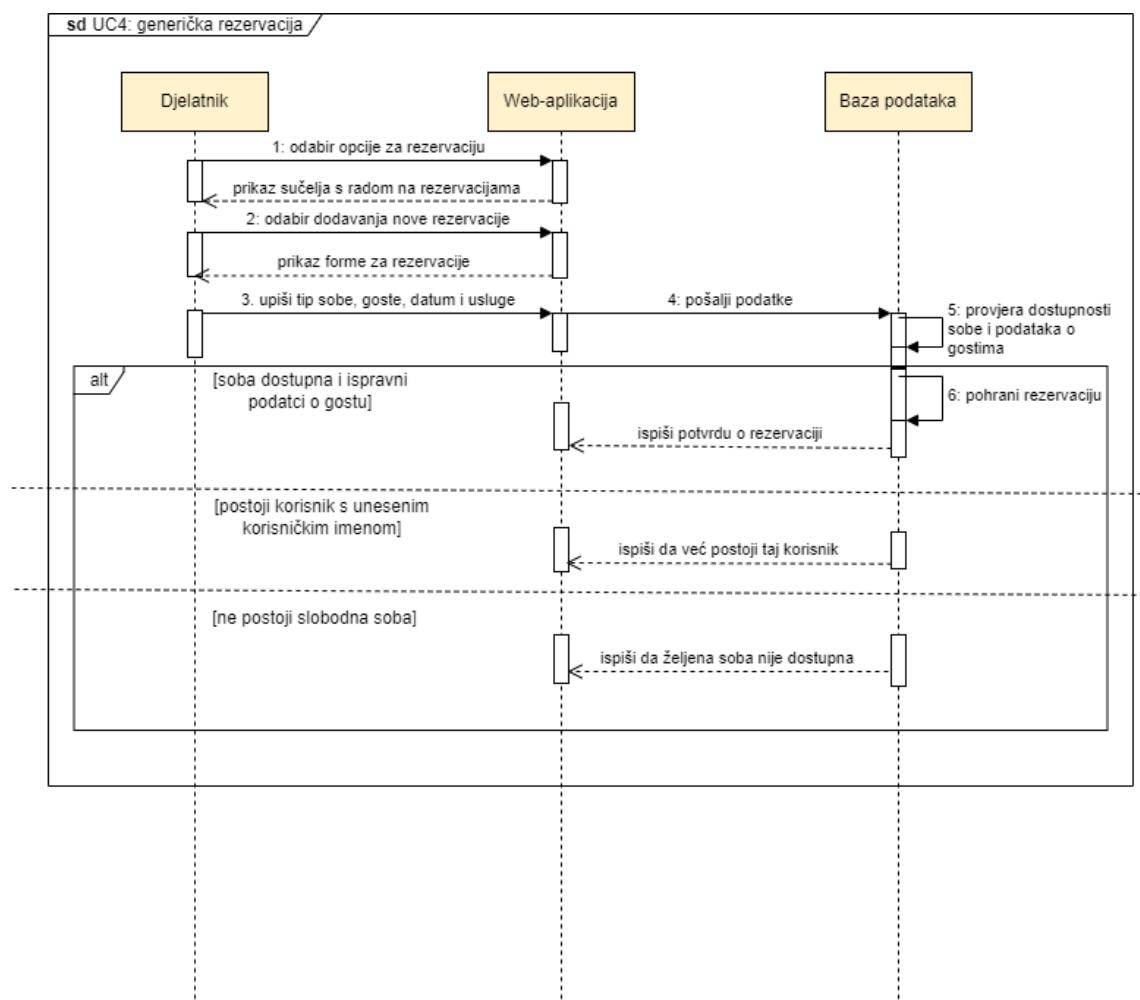


Slika 3.6: Vlasnik

### 3.1.2 Sekvencijski dijagrami

#### Obrazac uporabe - UC4 generička rezervacija

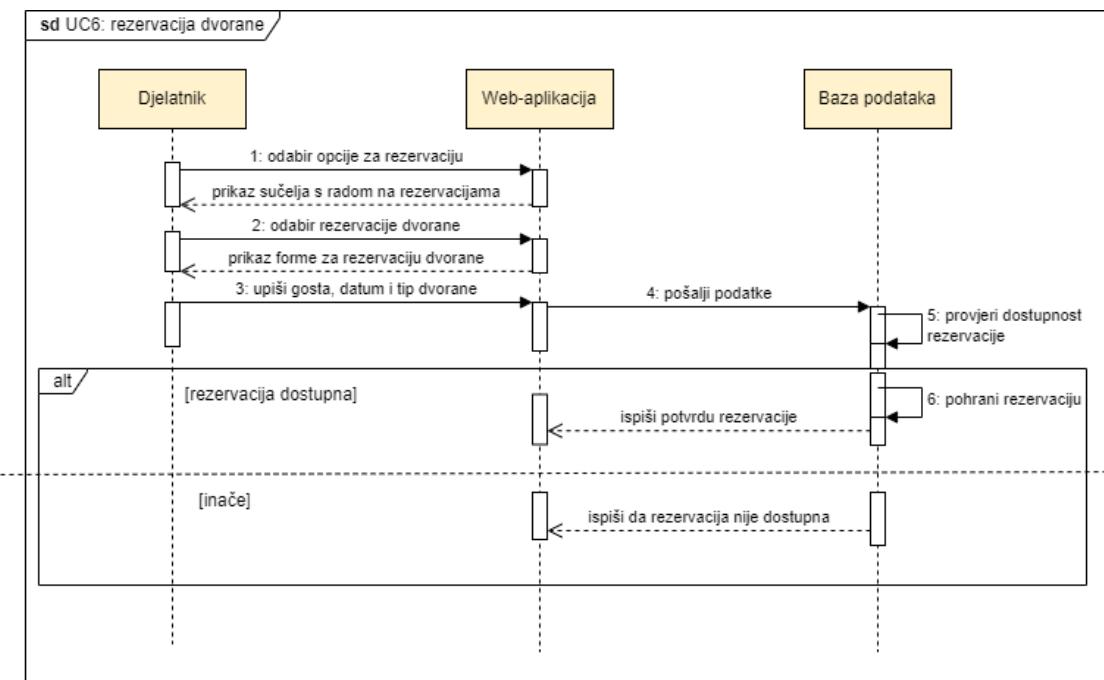
Djelatnik na recepciji odabire tip sobe koji želi rezervirati i šalje upit na poslužitelj. Poslužitelj provjerava da li postoji slobodna soba u bazi podataka. Ako postoji, automatski se rezervira soba, zauzima se u bazi podataka i vraća djelatniku da je rezervacija uspješna i broj sobe. Ako ne postoji slobodna soba, vraća poruku da su sve rezervirane.



Slika 3.7: Sekvencijski dijagram za UC4

### Obrazac uporabe - UC6 rezervacija dvorane

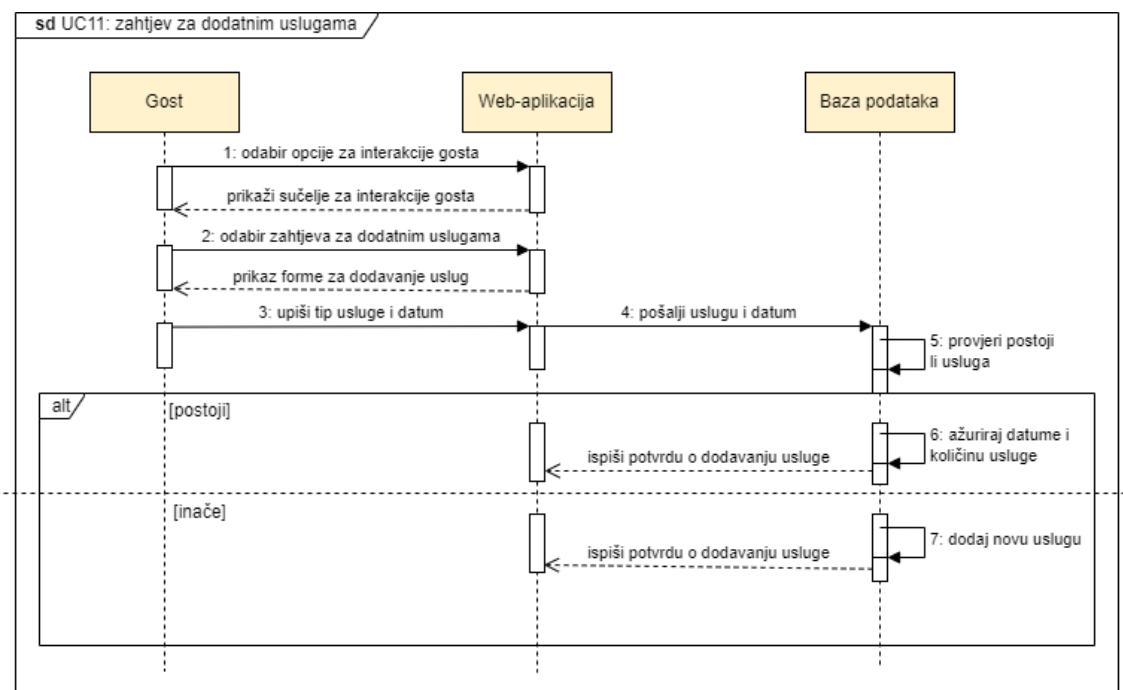
Djelatnik na recepciji odabire datum i šalje upit na poslužitelj. Poslužitelj provjera postoji li slobodni termini za dani datum. Ako postoji vraća ih te djelatnik potom odabire termin, poslužitelj pohranjuje promjene u bazi podataka i vraća potvrdu o rezervaciji. Ako nema slobodnih termina ispisuje se poruka da su svi termini zauzeti i nudi se opcija odabira drugog datuma.



Slika 3.8: Sekvencijski dijagram za UC6

### Obrazac uporabe - UC11 zahtjev za dodatnim uslugama

Gost odabire vrstu usluge koji želi i pošalje zahtjev na poslužitelj. Poslužitelj provjerava da li je dana usluga dostupna i ako je vraća potvrdu o postojanju usluge, ako ne postoji vraća poruku da tražena usluga nije dostupna. Korisnik bira za koji broj ljudi želi realizirati uslugu i šalje zahtjev. Poslužitelj provjeri je li usluga dostupna za traženi broj ljudi, ako je pohranjuje uslugu u bazi i vraća potvrdu da je usluga realizirana. Ako nije vraća mu poruku da nije dostupna za taj broj ljudi i nudi mu upis drugog broja ljudi.



Slika 3.9: Sekvencijski dijagram za UC11

### 3.2 Ostali zahtjevi

- Sustav treba omogućiti rad više korisnika u stvarnom vremenu
- Sustav treba koristiti hrvatski jezik te podržava hrvatsku abecedu
- Zadaci koji pristupaju bazi podataka se trebaju izvršiti u kratkom vremenu od najviše nekoliko sekundi
- Sustav treba biti izrađen kao web aplikacija koristeći objektno-orientirane jezike
- Sustavu se pristupa iz javne mreže pomoću protokola HTTPS
- Neispravno korištenje ne smije poremetiti rad sustava
- Sustav mora biti jednostavan za korištenje tako da ga je moguće koristiti bez uputa
- Rad na sustavu ne smije narušavati funkcionalnosti sustava
- Svi privatni podaci na sustavu su zaštićeni

## 4. Arhitektura i dizajn sustava

Arhitektura sustava se sastoji od tri glavna podsustava, a to su web preglednik, web poslužitelj i baza podataka.

**Web preglednik** je program koji korisniku omogućuje pregled web-stranica i multimedijalnih sadržaja vezanih uz njih. Svaki internetski preglednik je prevoditelj. Dakle, stranica je pisana u kodu koji preglednik nakon toga interpretira kao nešto svakome razumljivo. Korisnik putem web preglednika šalje zahtjev web poslužitelju.

**Web poslužitelj** osnova je rada web aplikacije. Njegova primarna zadaća je komunikacija klijenta s aplikacijom. Komunikacija se odvija preko HTTP protokola. Poslužitelj je onaj koji pokreće web aplikaciju te joj prosljeđuje zahtjev.

Korisnik koristi **web aplikaciju** za obrađivanje željenih zahtijeva. Web aplikacija obrađuje zahtjev te ovisno o zahtjevu, pristupa bazi podataka nakon čega preko poslužitelja vraća korisniku odgovor u obliku HTML dokumenta vidljivog u web pregledniku.

Programski jezik kojeg smo odabrali za izradu web aplikacije je Java Spring za backend i Angular za frontend. Odabранo razvojno okruženje je IntelliJ. Arhitektura sustava temeljiti će se na MVC (Model-View-Controller) konceptu. MVC koncept podržan je od strane Java Spring okvira i kao takav ima gotove predloške koji nam olakšavaju razvoj web aplikacije.

Karakteristika MVC koncepta je nezavisan razvoj pojedinih dijelova aplikacije što za posljedicu ima jednostavnije ispitivanje kao i jednostavno razvijanje i dodavanje novih svojstava u sustav.

MVC koncept sastoji se od:

- **Model** sadrži razrede čiji objekti se obrađuju.
- **View** (hrv. pogled) sadrži razrede čiji objekti služe za prikaz podataka.
- **Controller** (hrv. nadglednik) sadrži razrede koji upravljaju i rukuju korisničkom interakcijom s pogledom i modelom.

## 4.1 Baza podataka

Za potrebe našeg sustava koristit ćemo relacijsku bazu podataka koja svojom strukturom olakšava modeliranje stvarnog svijeta. Gradivna jedinka baze je relacija, odnosno tablica koja je definirana svojim imenom i skupom atributa. Zadaća baze podataka je brza i jednostavna pohrana, izmjena i dohvata podataka za daljnju obradu. Baza podataka ove aplikacije sastoje se od sljedećih entiteta: gost, hotel, rezervacija, usluga, radnik i smještajna jedinica

### 4.1.1 Opis tablica

**Hotel** Ovaj entitet sadržava sve informacije o hotelu. Sadrži atribute: ID, ime hotela i adresu hotela.

<b>Hotel</b>		
ID	INT	Jedinstveni ID hotela
HotelName	VARCHAR	Naziv hotela
Adress	VARCHAR	Adresa lokacije hotela
Email	VARCHAR	Mail adresa hotela
Fax	VARCHAR	Fax adresa hotela
OIB	INT	Jedinstveni OIB hotela

**MiniBar** Ovaj entitet sadržava sve informacije o različitim minibar opcijama u hotelu. Svaka opcija je opisana sa jedinstvenim IDem, veličinom pakiranja, imenom i cijenom.

<b>MiniBar</b>		
ID	INT	Jedinstveni ID minibar opcije
Amount	FLOAT	Količina proizvoda
Name	VARCHAR	Ime proizvoda
Price	VARCHAR	Cijena artikla

**Service** Ovaj entitet sadržava sve informacije o prodanim uslugama u hotelu. Svaka usluga opisana je jedinstvenim IDem, cijenom, tipom usluge te rezervacijom preko koje je naručena.

Service		
ID	INT	Jedinstveni ID usluge
Amount	FLOAT	Cijena usluge
reservationId	INT	ID rezervacije
productId	INT	ID produkta
guestReservationId	INT	ID gost-rezervacije

**Reservation** Ovaj entitet sadržava sve informacije o rezervaciji. Povezan je više-na-jedan s tablicama ReservationGuests, ReservationAccomodatingUnits te jedan-na-jedan sa ReservationServices

Reservation		
ID	INT	Jedinstveni ID rezervacije
startDate	DATE	Datum pocetka rezervacije
spentForMinibar	FLOAT	Ukupna cijena minibar usluga
endDate	DATE	Datum kraja rezervacije
guestId	INT	ID gosta

**AccommodatingUnit** Ovaj entitet sadržava sve informacije o smještajnim jedinicama hotela.

AccommodatingUnit		
ID	INT	ID smještajne jedinice
Name	VARCHAR	Jedinstveno ime smještajne jedinice
Capacity	INT	Kapacitet smještajne jedinice
Type	VARCHAR	Tip smještajne jedinice

**ReservationGuests** Ovaj entitet sadržava sve informacije o gostima na nekoj rezervaciji. Povezan je više-na-jedan s tablicama Reservation te Guest.

ReservationGuests		
ID	INT	ID rezervacije gosta
reservationId	INT	ID rezervacije
guestsId	INT	ID gostiju
accomodatingUnitId	INT	Id smjestajne jedinice

**Guest** Ovaj entitet sadržava informacije o gostu koje su potrebne za covid izvješće(Nadovezuje se na HotelUsera u kojem su opisani svi ostali potrebni atributi). Povezan je više-na-jedan s tablicom HotelUser te ReservationGuests

Guest		
guestId	INT	ID gosta
country	varchar	Drzava iz koje gost dolazi
accomodatingUnitId	INT	ID smjestajne jedinice

**HotelUser** Ovaj entitet sadržava sve informacije o gostu hotela. Sadrži atribute: Korisničko ime, ID, ID smještajne jedinice, ime, prezime, OIB, adresu prebivališta, državu porijekla, broj mobitela i lozinku. Ovaj entitet je u vezi vise-na-vise s entitetom Usluge preko ID-a gosta i ID-a usluge, u vezi vise-na-jedan s entitetom Smještajna jedinica preko ID-a smještajne jedinice.

HotelUser		
ID	INT	ID gosta
userName	VARCHAR	Jedinstveni naziv računa gosta
Name	VARCHAR	Ime gosta
Surname	VARCHAR	Prezime gosta
OIB	VARCHAR	OIB gosta
address	VARCHAR	Adresa prebivališta gosta
phoneNumber	VARCHAR	Jedinstveni telefonski broj gosta
password	VARCHAR	Hash lozinke

**Worker** Ovaj entitet sadržava sve informacije o radniku hotela.

Worker		
workerId	INT	ID radnik
dateOfHire	date	Datum zaposlenja radnika

**Bill** Ovaj entitet sadržava informacije o računu prikazanom na kraju boravka u hotelu.

Bill		
brojRacuna	INT	ID računa
iznos	FLOAT	Ukupni iznos računa

**UnitType** Ovaj entitet sadržava informacije o različitim tipovima smještaja u hotelu

UnitType		
type	VARCHAR	Tip smještajne jedinice
iznos	FLOAT	Cijena noćenja u smj. jedinici

**Hall** Ovaj entitet sadržava informacije o različitim tipovima smještaja u hotelu

Hall		
Id	INT	Id dvorane
capacity	INT	Broj sjedećih mesta dvorane
name	VARCHAR	Naziv dvorane

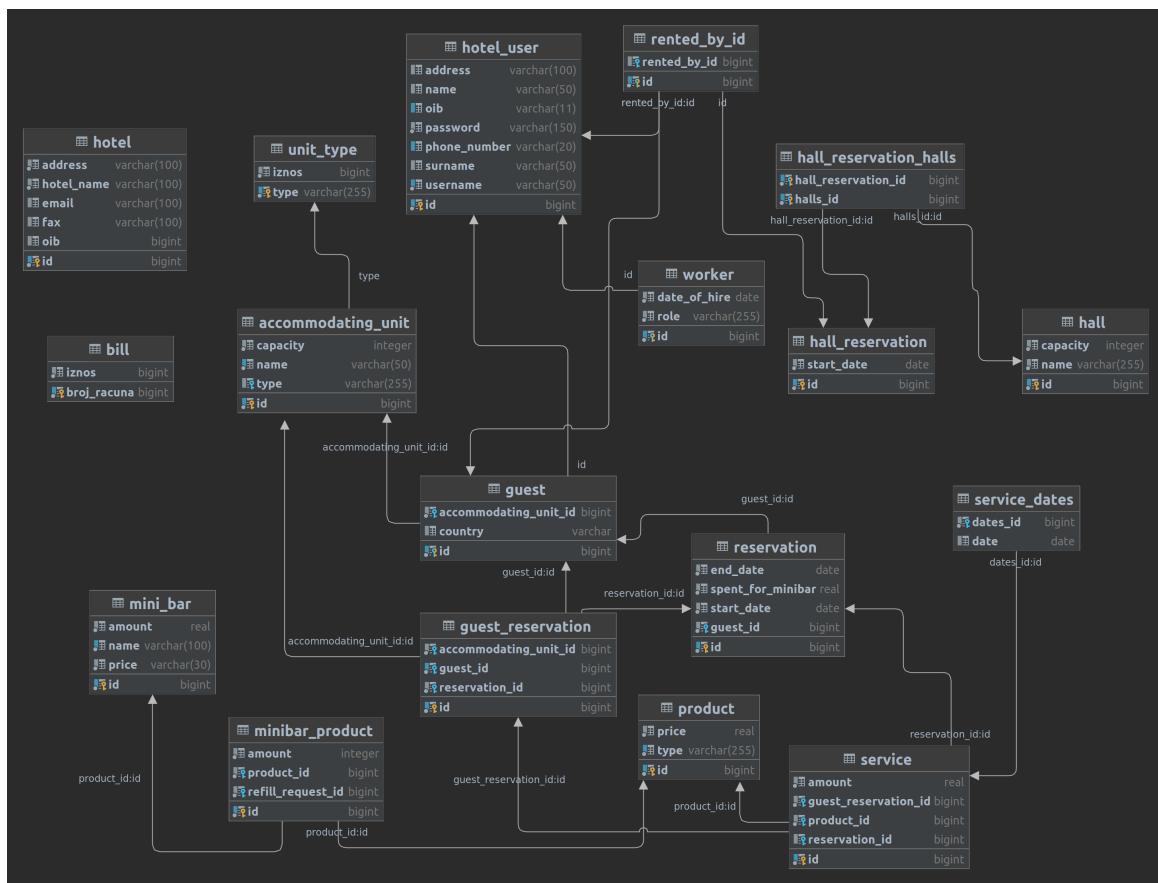
**HallReservation** Ovaj entitet sadržava informacije o rezervaciji dvorane

HallReservation		
Id	INT	Id rezervacije dvorane
rentedById	INT	Id gosta koji je rezervirao dvoranu
startDate	DATE	Datum rezervacije dvorane
hallId	INT	Id dvorane

**ServiceProduct** Ovaj entitet sadržava sve informacije o različitim tipovima usluga u hotelu.

ServiceProduct		
ID	INT	Jedinstveni ID usluge
price	FLOAT	Cijena produkta
type	INT	Tip produkta

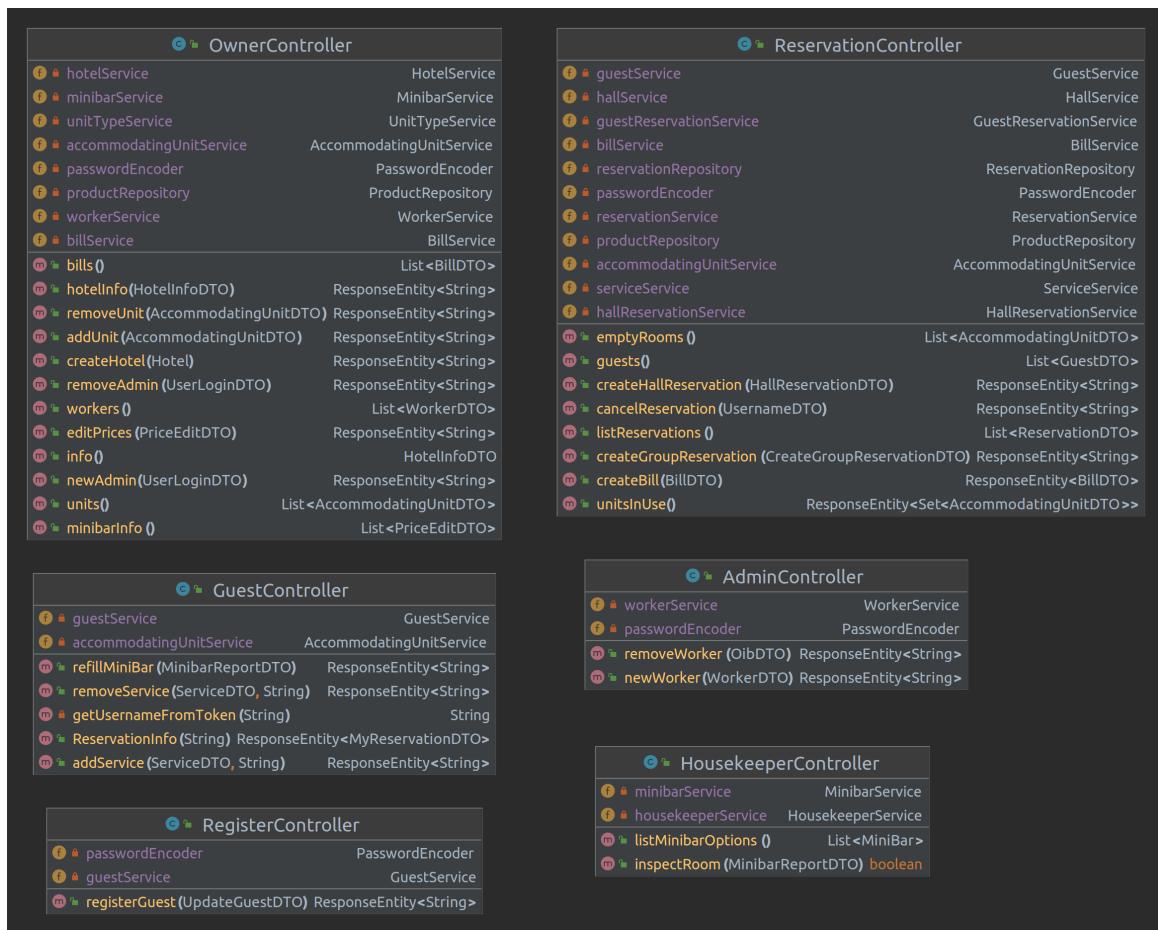
#### 4.1.2 Dijagram baze podataka



Slika 4.1: ER dijagram baze podataka

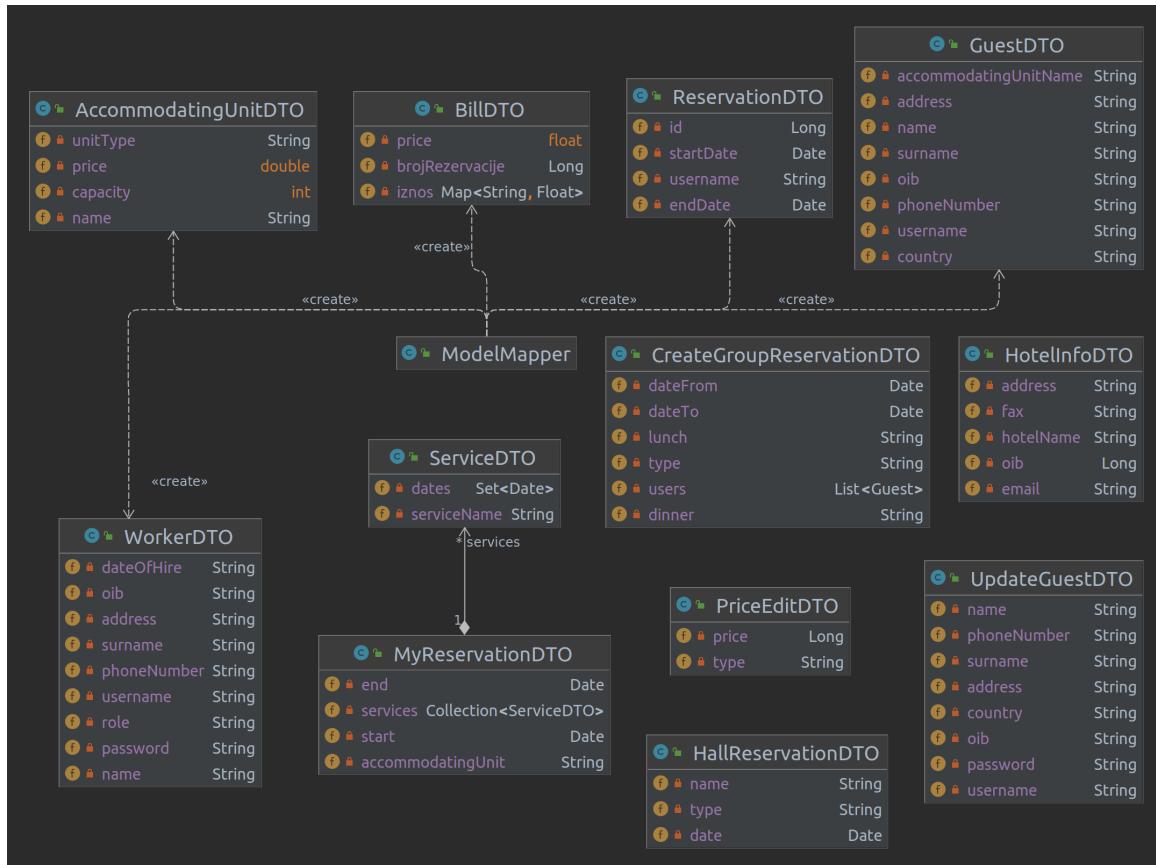
## 4.2 Dijagram razreda

Na slikama 4.2, 4.3 i 4.4 su prikazani razredi koji pripadaju backend dijelu MVC arhitekture. Razredi prikazani na slici 4.2 su kontroleri koji primaju http zahjeve. Metode implementirane u tim razredima manipuliraju s DTO (Data transfer object). Metode implementirane u Controller razredima vraćaju http odgovore s html statusnim kodom i podacima u JSON formatu. Na slici se mogu vidjeti sve ponuđene mogućnosti dohvata podataka s backend-a.



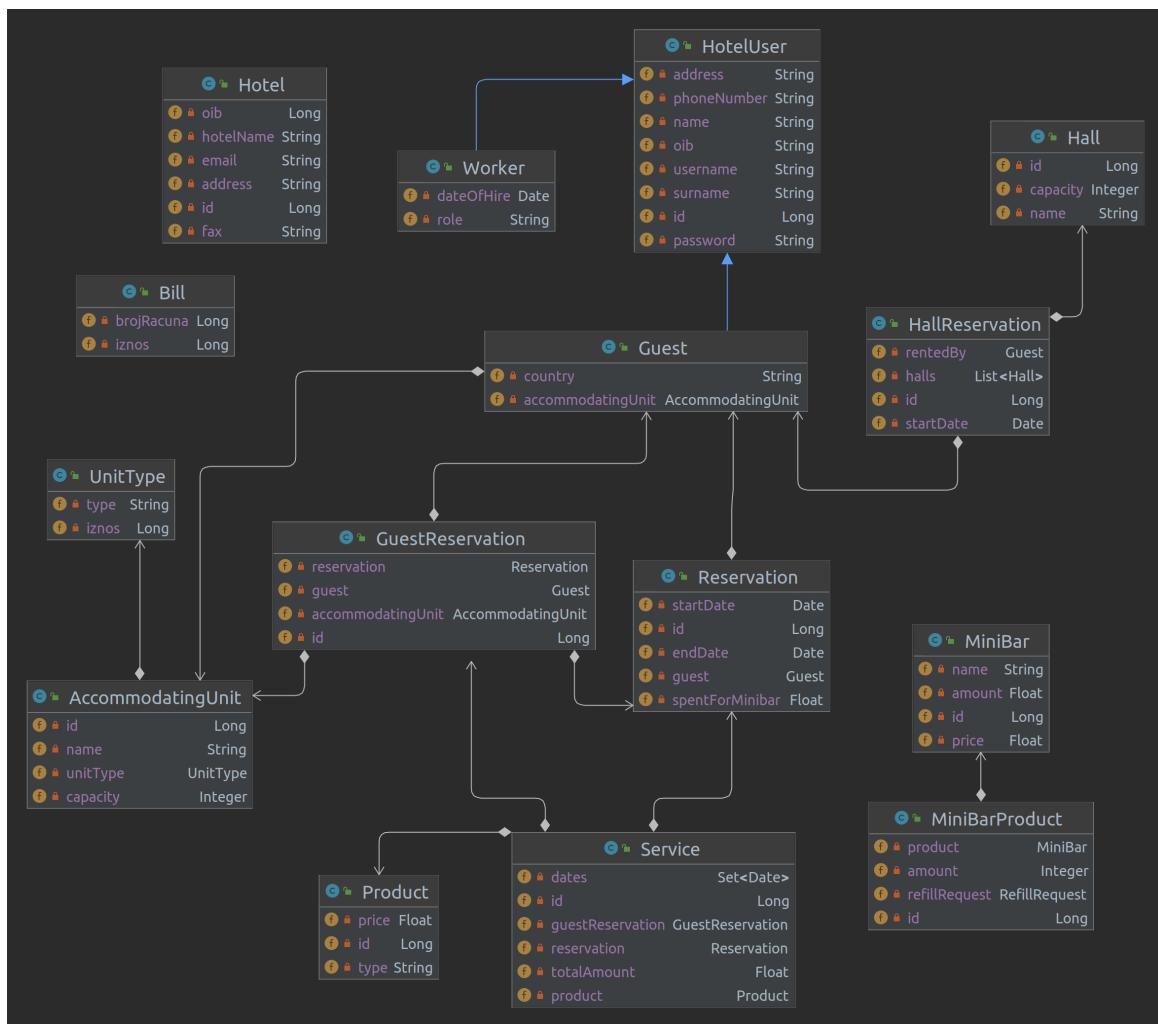
Slika 4.2: Dijagram razreda - Kontroleri

Razredi prikazani na slici 4.3 predstavljaju oblik DTO (Data transfer object). Ti razredi se koriste kako bi se, u slučaju izmjena na bazi podataka, izmjene koda što lakše napravile. Odnosno, to su razredi u koje se preslikavaju entiteti iz koda/baze kad ih je potrebno vratiti u JSON (u našem slučaju samo JSON) obliku.

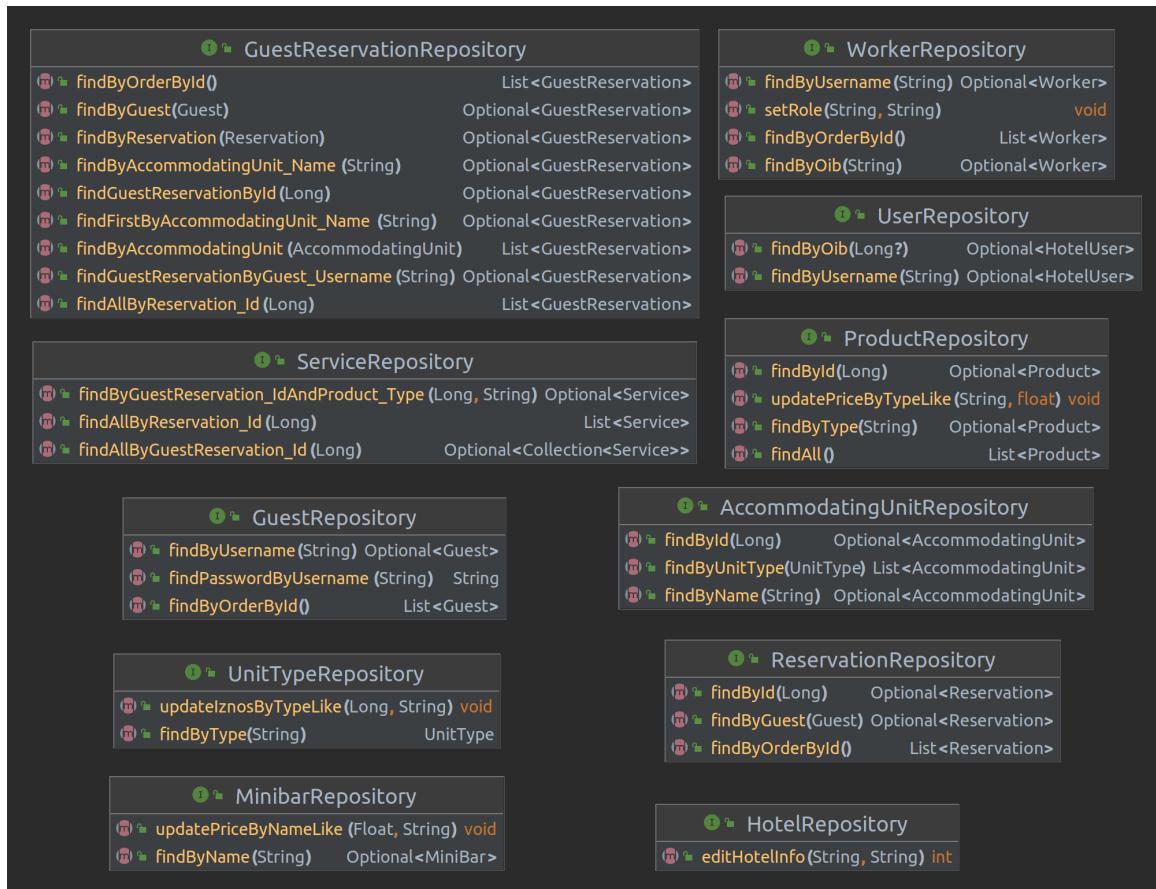


Slika 4.3: Dijagram razreda - DTO modeli

Na slici 4.4 prikazani su razredi koji preslikavaju strukture baze podataka u aplikaciju (kod). Ti se razredi koriste kao modeli koji se mogu direktno spremiti u bazu podataka. Za spremanje u bazu podataka koriste se sučelja repository (repository oblikovni obrazac) koja nasljeđuju JpaRepository i prikazana su na slici 4.5. Ista sučelja se koriste i za dohvata podataka. U centru domene su rezervacija i korisnik hotela (reservation i hotelUser). Rezervaciju u sebi sadrže rezervirane jedinice, goste hotela i ostalo potrebne informacije. Korisnici hotela su osoblje hotela, goste zakupci dvorana.

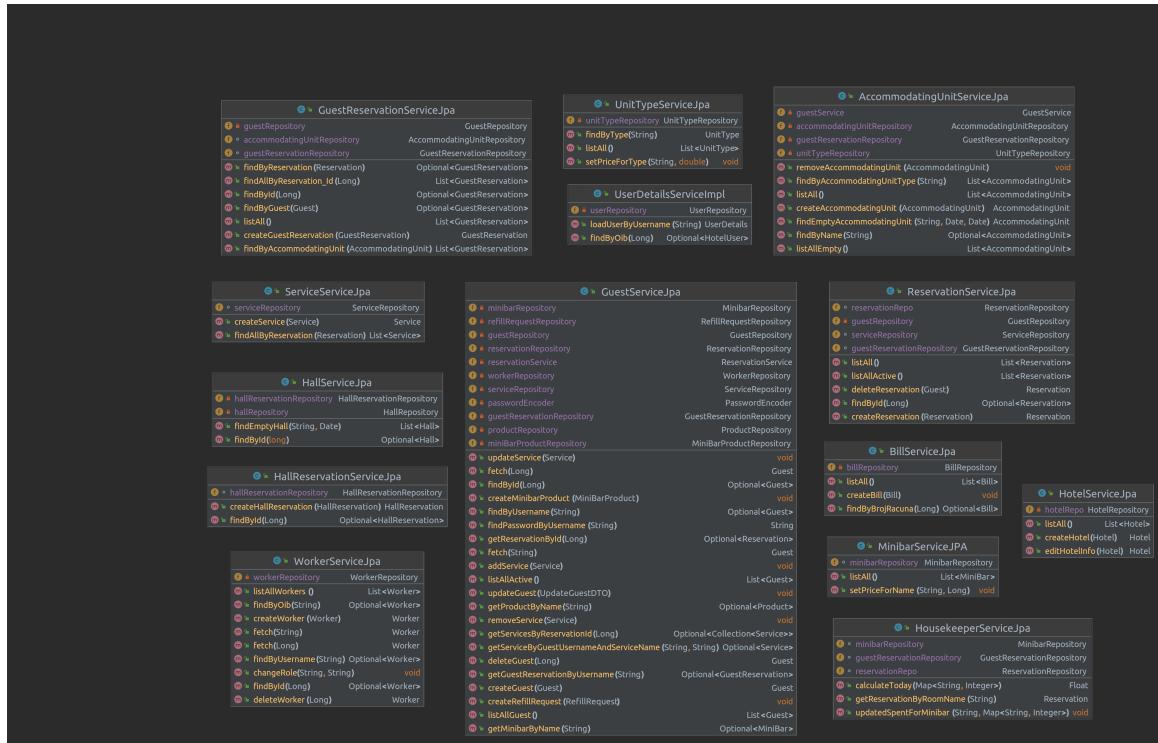


Slika 4.4: Dijagram razreda - Domenski modeli



Slika 4.5: Dijagram razreda - Repository sloj

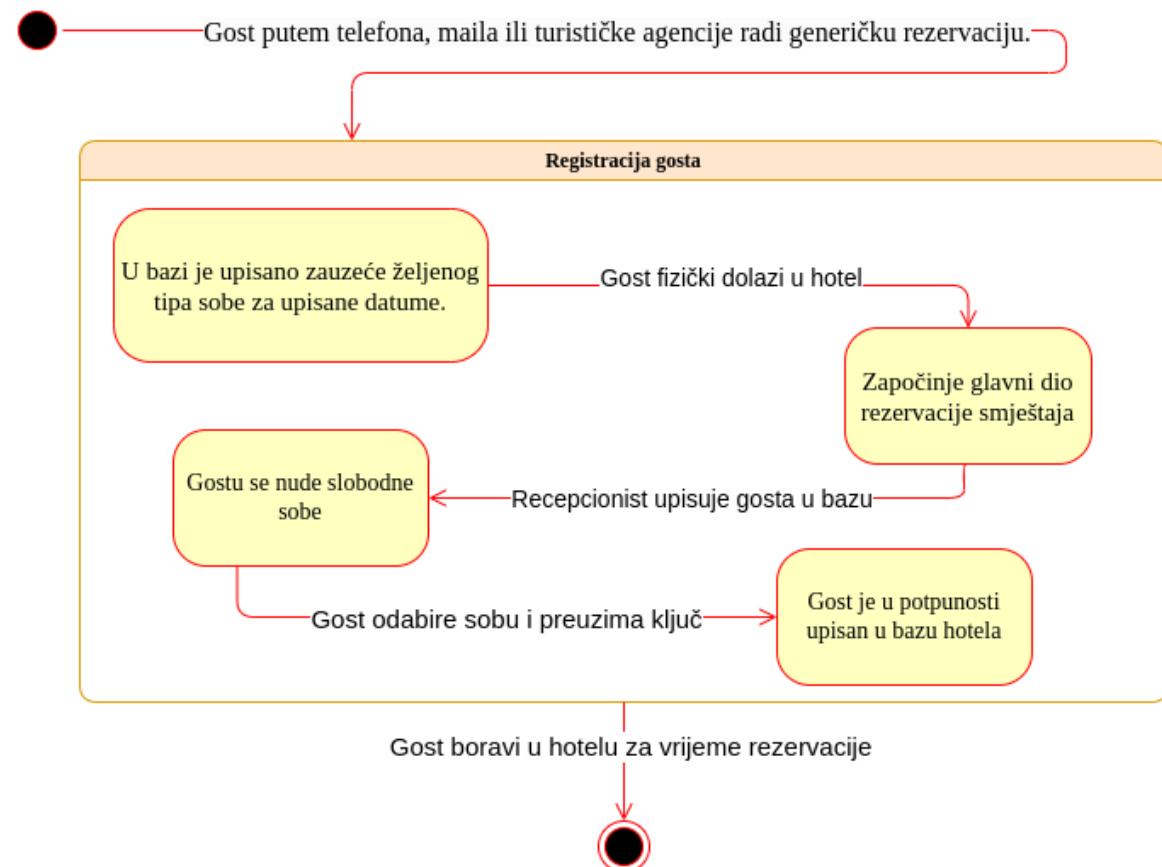
Slika 4.6 prikazuje Service sloj backend-a. Service sloj je povezan s repository slojem i s kontrolerima koji su prikazani na slici 4.2. U service sloju se nalazi poslovna logika aplikacije.



Slika 4.6: Dijagram razreda - Service sloj

### 4.3 Dijagram stanja

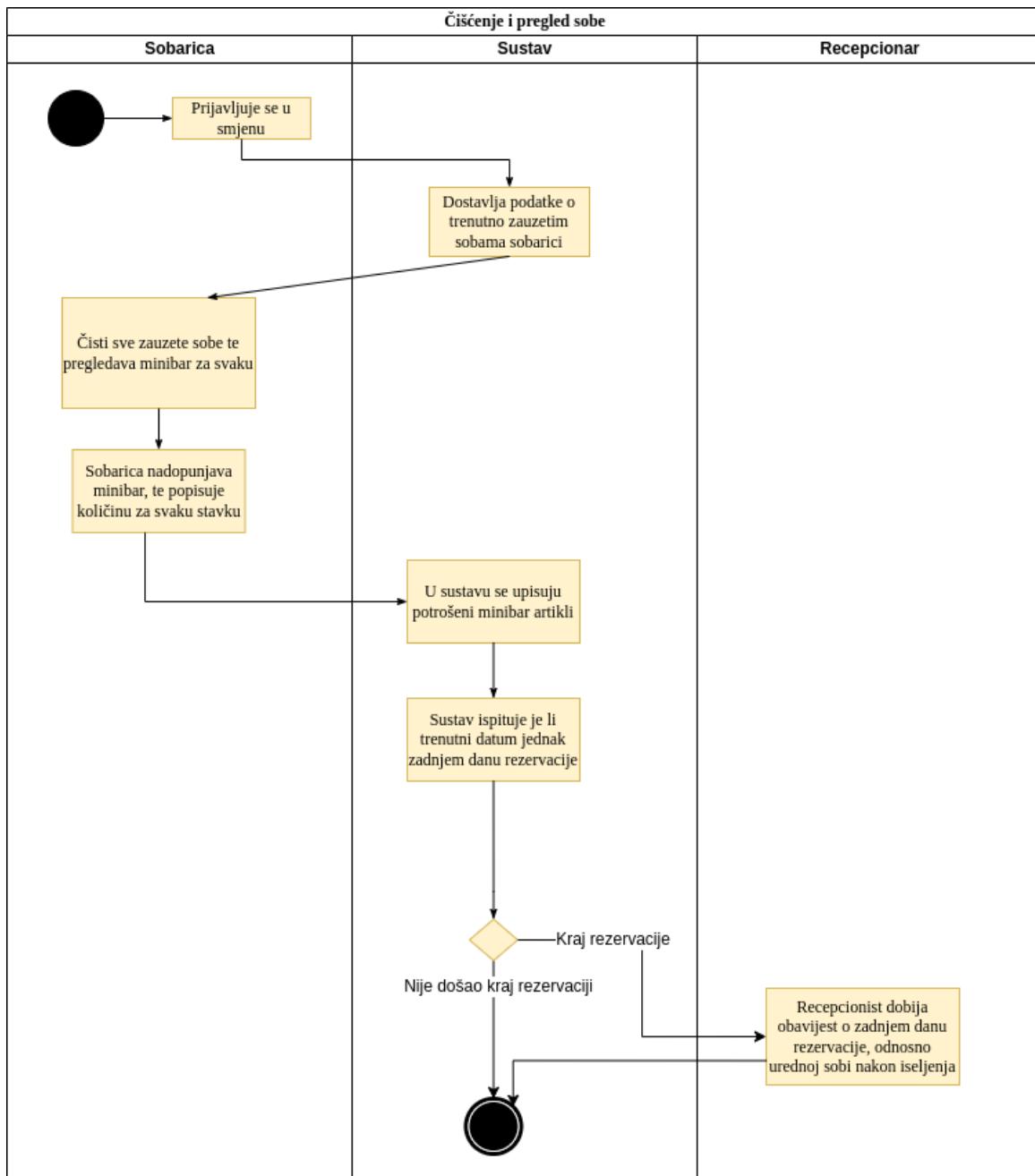
Na slici 4.7 prikazan je postupak rezervacije te useljenja gosta u hotel. Sam postupak rezervacije započinje kada gost telefonom, mailom ili preko turističke agencije napravi generičku rezervaciju u kojoj on za željeni datum predrezervira tip sobe u kojoj želi biti. Potom, kada gost dođe fizički u hotel započinje se drugi dio registracije tijekom kojeg gost daje recepcionistu svoje osobne podatke koje onda taj recepcionist unosi u aplikaciju. Nakon uspješne registracije, recepcionist od aplikacije dobije popis slobodnih soba tipa kojeg je gost unio prilikom generičke rezervacije, te potom gost odabire jednu i preuzima njen ključ, čime se registracija gosta završava.



Slika 4.7: Registracija gosta

## 4.4 Dijagram aktivnosti

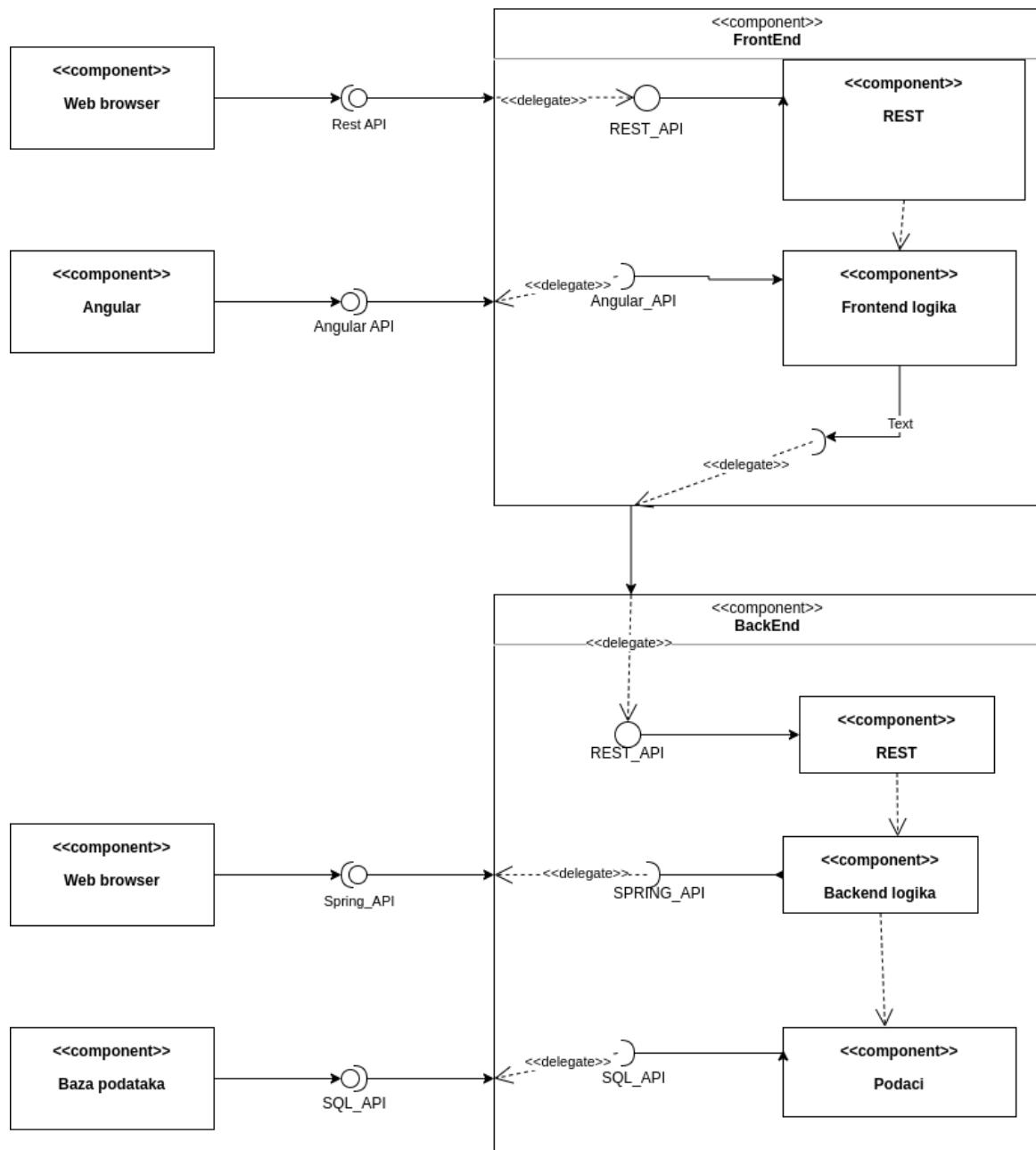
Na slici 4.8 prikazan je postupak čišćenja, pregleda i izvještaja sobe od strane soberice hotela. Sobarica nakon što se prijavi u smjenu, kreće redom čistiti sobe u kojima borave gosti. Prilikom čišćenja, soberica nadopunjava minibar do vrha, te u izvještaj zapisuje koliko je kojeg artikla iz minibara gost potrošio. Nakon završenog čišćenja, soberica podnosi izvještaj u aplikaciji, koja onda na ukupnu potrošnju rezervacije zbraja dnevnu potrošnju minibara, te provjerava ako je li trenutni datum jednak zadnjem danu rezervacije, te ako jest, šalje recepcionaru obavijest o pregleđanoj i urednoj sobi, koja je uvjet za iseljenje.



Slika 4.8: Pregled sobe

## 4.5 Dijagram komponenti

Na slici 4.9 prikazan je dijagram komponenti. Korisnik pristupa korisničkom sučelju preko svog web preglednika koji je preko REST API servisa komunicira s poslužiteljskom stranom aplikacije. Poslužiteljska se strana aplikacije sastoji od baze podataka i modula poslovne logike(backend logika). Korisničko sučelje dizajnirano je u Angularu, a poslužiteljska strana u Spring Boot-u.



Slika 4.9: Dijagram komponenti

# 5. Implementacija i korisničko sučelje

## 5.1 Korištene tehnologije i alati

Prilikom generiranja ove dokumentacije, koristili smo **OverLeaf** kao glavni alat za organiziranje i zapis dokumentacijskog teksta. Za neke dijagrame smo koristili alate **DrawIO**(<https://app.diagrams.net/>) čija je prednost laka izrada proizvoljnih dijagrama uporabom grafičkog sučelja, dok smo **AstahUML**(<https://astah.net/products/astah-uml/>) koristili pri izradi sekvencijskih te dijagrama obrazaca uporabe. Komunikacija u timu je ostvarena korištenjem aplikacije **Discord**(<https://discord.com/>). Za upravljanje izvornim kodom korišten je alat **Git**(<https://git-scm.com/>). Udaljeni repozitorij projekta nalazi se na platformi **Gitlab**(<https://gitlab.com/>). Za upravljanje zadacima korištena je aplikacija **Trello**(<https://trello.com/>).

Za razvojno okruženje korišten je **IntelliJ**(<https://www.jetbrains.com/idea/>). Aplikaciju smo razvijali u dvjema tehnologijama, za korisničku stranu aplikacije bio je to **Angular**, dok smo za poslužiteljsku stranu koristili **Java Spring Boot**.

**Angular** je TypeScript okvir dizajniran za razvoj korisničkog dijela web aplikacija odnosno korisničkog sučelja kojeg održava jedan od timova u Google-u.

**Spring Boot** je Java okvir koji je namijenjen kreiranju poslužiteljske strane aplikacije, odnosno obradi podataka iz baze podataka i njihova prenamjena prema korisničkom sučelju

## 5.2 Ispitivanje programskog rješenja

### 5.2.1 Ispitivanje komponenti

Ispitivanje komponenti ostvareno je u Spring Boot-u pomoću JUnit i Mockito alata za ispitivanje. U nastavku su opisani provedeni testovi i priloženi su njihovi kodovi. Test 1 testira service koji se koristi za dodavanje i uklanjanje radnika iz sustava. Provjerava se slučaj kad pokušamo dodati radnika s username-om koji već postoji u sustavu i brisanje nakon toga.

---

```
@Test
public void createWorkerTest(){
    Assertions.assertThrows(NullPointerException.class, () ->
        workerService.createWorker(null));

    Worker w1 = new Worker("Test1", "Test1", "10201020101", "Test1",
        "098123456", "TEST", "TEST", "ROLE_RECEPTIONIST",
        new Date(System.currentTimeMillis()));

    workerService.createWorker(w1);
    Assertions.assertThrows(IllegalArgumentException.class, () ->
        workerService.createWorker(w1));

    workerService.deleteWorker(workerService.findByUsername("TEST").get().getId());
    Assertions.assertTrue(workerService.findByUsername("TEST").isEmpty());
}
```

---

Test 2 testira kontroler za kreiranje računa pomoću MOCKMVC-a. Cilj testa je provjeriti da se JSON objekt koji se vraća dobro generira. To jest da ima dobre elemente i da su vrijenosti elemenata dobro izračunate.

---

```
@Test
public void billCreationTest() throws Exception {
    Date start = new Date(System.currentTimeMillis());
    Date end = new Date(System.currentTimeMillis() + 240000000L);
    Reservation reservation = new Reservation(start, end, 142F, new
        Guest("Kalamarko", "12345"));
    reservation.setId(2L);
```

---

```
given(reservationService.findById(reservation.getId())).willReturn(Optional.of(reservation));
given(serviceService.findAllByReservation(reservation)).willReturn(new
    LinkedList<>());
doNothing().when(billService).createBill(any());

List<GuestReservation> guestReservations = new ArrayList<>();

guestReservations.add(new GuestReservation(reservation, new
    Guest("Kalamarko", "12345")
    , new AccommodatingUnit("101", 1, new UnitType("JEDNOKREVETNA",
        20L)));
guestReservations.add(new GuestReservation(reservation, new
    Guest("Patrik", "12345")
    , new AccommodatingUnit("202", 1, new UnitType("DVOKREVETNA",
        50L)));
guestReservations.add(new GuestReservation(reservation, new
    Guest("Spuzvabob", "12345")
    , new AccommodatingUnit("202", 1, new UnitType("DVOKREVETNA",
        50L))));

given(guestReservationService.findAllByReservation_Id(reservation.getId())).willReturn(
    guestReservations);

BillDTO bill = new BillDTO();
bill.setBrojRezervacija(2L);

mvc.perform(post("/reservations/bill")
    .header("Authorization",
        token).contentType(MediaType.APPLICATION_JSON).accept(MediaType.APPLICATION_JSON)
    .content(new ObjectMapper().writeValueAsString(bill)))
.andExpect(status().isOk())
.andExpect(jsonPath("iznos.Minibar").value(142))
.andExpect(jsonPath("iznos.Noenje").value(240))
.andExpect(jsonPath("iznos.[‘Dodatne usluge’]").value(0))
.andExpect(jsonPath("price").value(382));
}
```

Test 3 testira kontroler za dohvata podataka o rezervaciji određenog korisnika. Cilj testa je provjeriti točnost podataka koje kontroler vraća. Kod testa je u nastavku:

```

@Test
public void reservationInfoTest() throws Exception {
    Date start = new Date(System.currentTimeMillis());
    Date end = new Date(System.currentTimeMillis() + 24000000);
    GuestReservation guestResrvation = new GuestReservation(
        new Reservation(start, end, 100F, new Guest("Kalamarko",
            "12345"))
        , new Guest("Kalamarko", "12345")
        , new AccommodatingUnit("101", 1, new UnitType("JEDNOKREVETNA",
            100L)));
    given(guestService.getGuestReservationByUsername("Kalamarko")).willReturn(Optional.of(guestResrvation));
    mvc.perform(get("/myreservation").header("Authorization",
        token).contentType(MediaType.APPLICATION_JSON).accept(MediaType.APPLICATION_JSON))
        .andExpect(status().isOk())
        .andExpect(jsonPath("start").value(start.toString()))
        .andExpect(jsonPath("end").value(end.toString()))
        .andExpect(jsonPath("accommodatingUnit").value("101"))
        .andExpect(jsonPath("services", hasSize(0)));
}

```

Testovi 4-6 testiraju kontroler koji se koristi za funkcionalnosti vlasnika. Test 4 provjera točnost vraćenih podataka o hotelu.

```

@Test
public void hotelInfoTest() throws Exception {
    List<Hotel> hotelInfo = new ArrayList<>();
    hotelInfo.add(new Hotel("TEST", "TEST", "TEST", 12345678911L, "TEST"));
    given(hotelService.listAll()).willReturn(hotelInfo);

    mvc.perform(get("/hotel/info").header("Authorization",
        token).contentType(MediaType.APPLICATION_JSON).accept(MediaType.APPLICATION_JSON))
        .andExpect(status().isOk())
        .andExpect(jsonPath("hotelName").value("TEST"))
        .andExpect(jsonPath("oib").value("12345678911"));
}

```

Cilj testa 5 je provjeriti kontroler koji vraća podatke o svim radnicima.

---

```
@Test
public void allWorkersTest() throws Exception {
    List<Worker> workers = new ArrayList<>();
    workers.add(new Worker("TEST", "TEST", "12345678911", "TEST",
        "0981564256", "TEST", "1234", "ROLE_RECEPTIONIST", new
        Date(System.currentTimeMillis())));
    workers.add(new Worker("TEST", "TEST", "12345678912", "TEST",
        "0981564256", "TEST", "1234", "ROLE_RECEPTIONIST", new
        Date(System.currentTimeMillis())));
    workers.add(new Worker("TEST", "TEST", "12345678913", "TEST",
        "0981564256", "TEST", "1234", "ROLE_HOUSEKEEPER", new
        Date(System.currentTimeMillis())));
    workers.add(new Worker("TEST", "TEST", "12345678914", "TEST",
        "0981564256", "TEST", "1234", "ROLE_OWNER", new
        Date(System.currentTimeMillis())));
    given(workerService.listAllWorkers()).willReturn(workers);

    mvc.perform(get("/hotel/info/workers").header("Authorization",
        token).contentType(MediaType.APPLICATION_JSON).accept(MediaType.APPLICATION_JSON))
        .andExpect(status().isOk())
        .andExpect(jsonPath("$.size()", hasSize(3)))
        .andExpect(jsonPath("$[0].oib").value("12345678911"))
        .andExpect(jsonPath("$[1].oib").value("12345678912"))
        .andExpect(jsonPath("$[2].oib").value("12345678913"));
}
```

---

Test 6 testira vraćanje podataka o svim smještajnim jedinicama u hotelu.

---

```
@Test
public void allUnitsTest() throws Exception {
    List<AccommodatingUnit> aUnits = new ArrayList<>();
    aUnits.add(new AccommodatingUnit("101", 4, new UnitType("OBITELJSKA",
        100L)));
    aUnits.add(new AccommodatingUnit("102", 1, new
        UnitType("JEDNOKREVETNA", 20L)));
    aUnits.add(new AccommodatingUnit("201", 2, new UnitType("DVOKREVETNA",
        45L)));

    given(accommodatingUnitService.listAll()).willReturn(aUnits);

    mvc.perform(get("/hotel/info/units").header("Authorization",
        token).contentType(MediaType.APPLICATION_JSON).accept(MediaType.APPLICATION_JSON))
        .andExpect(status().isOk())
        .andExpect(jsonPath("$.name", hasSize(3)))
        .andExpect(jsonPath("$[0].name").value("101"))
        .andExpect(jsonPath("$[1].capacity").value("1"))
        .andExpect(jsonPath("$[2].name").value("201"));
}
```

---

### 5.2.2 Ispitivanje sustava

Ispitivanje sustava smo proveli u alatu **Karma**. Kreirali smo 37 ispitnih slučajeva koji testiraju kreiranje komponenti, te validaciju podataka.

1. ispitni slučaj testira omogućavanje gumba nakon ispravno unesenog ulaza u formularu. U polje za unos se postavlja ispravna vrijednost, te sustav očekuje omogućenost gumba.

---

```
it('should be disabled', () => {
    component.make.controls["username"].setValue("Mate");
    expect(el.nativeElement.querySelector('button').disabled).toBeFalsy();
})
```

---

2. ispitni slučaj testira omogućenost podnošenja formulara kojem nisu ispunjena sva nužna polja za unos. U polje "tip sobe" postavlja se prazan znakovni niz i test očekuje neispravan formular.

---

```
it('should be invalid', async () =>{
    component.roomRes.controls["type"].setValue("");
    expect(component.roomRes.valid).toBeFalsy();
})
```

---

3. slučaj provjerava validaciju duljine unesenog niza znakova u polje za unos vrijednosti OIB-a. Unosi se znakovni niz duljine veće od 11 znakova i test očekuje da formular neće biti ispravan.

---

```
it('oib should be invalid', async () =>{
    component.regForm.controls['oib'].setValue("12345678912345");
    expect(component.regForm.valid).toBeFalsy()
})
```

---

4. slučaj provjerava stvara li se servis za dohvaćanje rezervacija ispravno stvara.

---

```
describe('ReservationService', () => {
  let service: ReservationService;

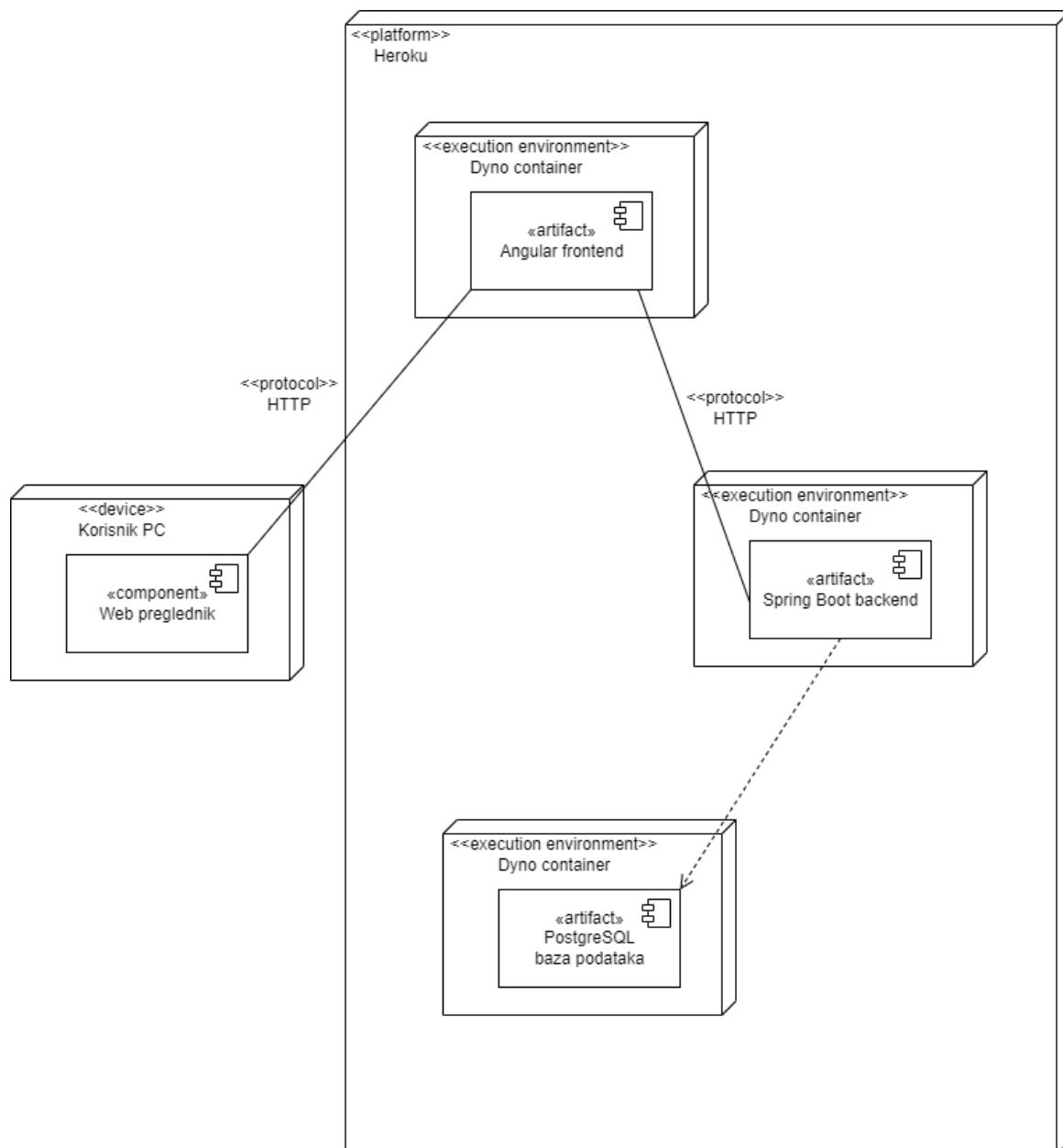
  beforeEach(() => {
    TestBed.configureTestingModule({imports: [HttpClientTestingModule]});
    service = TestBed.inject(ReservationService);
  });

  it('should be created', () => {
    expect(service).toBeTruthy();
  });
});
```

---

### 5.3 Dijagram razmještaja

Dijagrami razmještaja opisuju topologiju sklopolja i programsku potporu koja se koristi u implementaciji sustava u njegovom radnom okruženju. Na platformi Heroku se nalaze poslužitelji za frontend, backend i bazu podataka. Klijenti pristupaju aplikaciji korištenjem web preglednika. Sustav se temelji na arhitekturi "klijent-poslužitelj", a komunikacija između računala korisnika (gosti, zaposlenici, administrator, vlasnik) i poslužitelja se odvija preko HTTP veze.



Slika 5.1: Dijagram razmještaja

## 5.4 Upute za puštanje u pogon

Potrebno je napraviti korisnički račun na Heroku i zatim odabirom opcije "New app" stvoriti dvije aplikacije, jednu za frontend i jednu za backend. S obzirom na to da je puštanje u pogon značajno jednostavnije s Githuba nego s Gitlaba, potrebno je neki, npr. osobni Github račun, povezati s Heroku računom i potom klonirati projekt dva puta. U prvom kloniranom projektu je potrebno u root direktorij premjestiti sadržaj poddirektorija *backend*, a ostatak sadržaja projekta obrisati. U drugom je projektu potrebno učiniti istu stvar za sadržaj poddirektorija *frontend*. Zatim je u svakom projektu na main grani potrebno odabrati opciju "Deploy to Heroku".

Deploy s gitlab-a može se napraviti s danom skriptom iz direktorija na gitlabu pomoću HEROKU CLI-a Upute za heroku CLI su dane na linku:

<https://devcenter.heroku.com/articles/heroku-cli>. Postupak je isti kao za deploy s githuba osim što nije potrebno raditi dva projekta nego samo dvije aplikacije. Nakon sto su obje aplikacije napravljene na heroku, potrebno je izmijeniti skriptu da odgovara imenima tih aplikacija. Predložak je dan u nastavku:

---

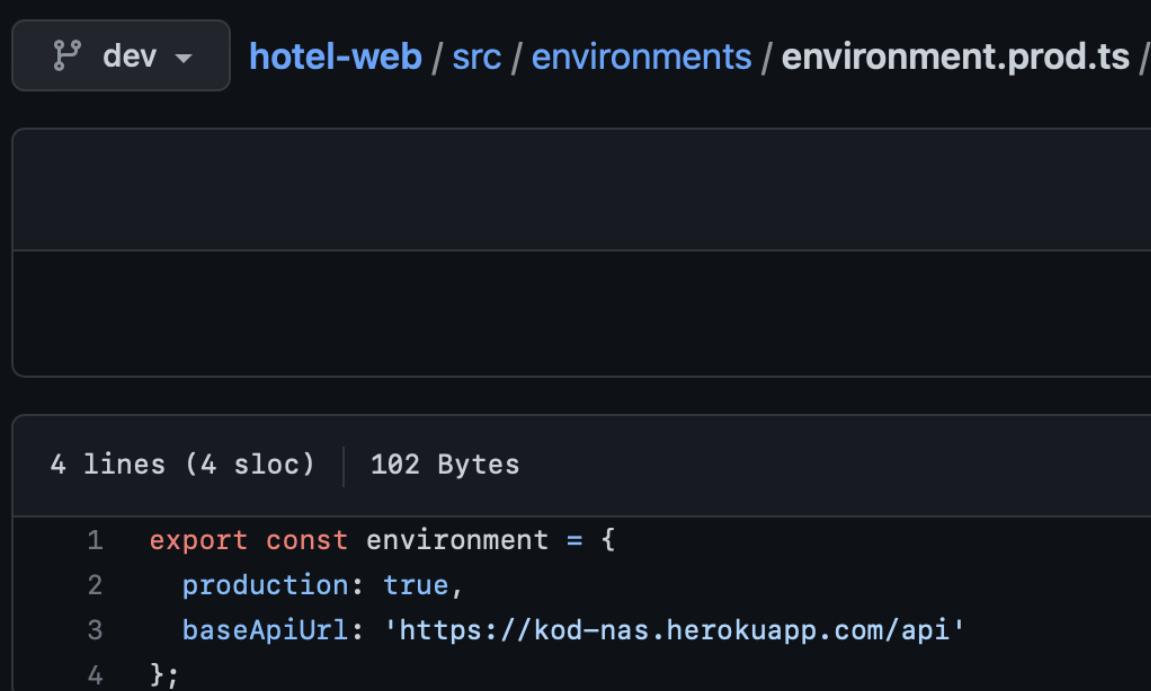
```
heroku git:remote -a [ime heroku aplikacije za frontend] &&
git subtree push --prefix frontend/ heroku [ime grane]:master &&
heroku git:remote -a [ime heroku aplikacije za backend] &&
git subtree push --prefix backend/ heroku [ime grane]:master
```

---

Nakon pokretanja skripte ili deployanja s Githuba postupak je dalje isti. Na Heroku Vas dočeka stranica poput one na slici 5.2. Kada ste u pogon pustili frontend i backend, potrebno je u datoteku /src/environments/environment.prod.ts na frontend projektu napisati točan URL na kojem se nalazi backend. Primjer je prikazan na slici 5.3. Pri puštanju u pogon backenda, automatski se stvori i baza podataka. Na slici 5.2 vidljiva je poveznica "Resources". Potrebno je nju odabrati i pod "add-ons" odabrati "heroku postgres" i dočekaju Vas podaci slični onima na slici 5.4., nakon prijave na bazu podataka u inicijalnom pokretanju potrebno je postaviti vlasnika hotela. Nakon toga vlasnik ima daljnje ovlasti dodavanja ostalog osoblja u hotel.

The screenshot shows the Heroku application settings page for 'kod-nas-je-najljepse'. At the top, there's a navigation bar with 'Personal' and 'GitHub' buttons, and tabs for 'Overview', 'Resources', 'Deploy' (which is selected), 'Metrics', 'Activity', 'Access', and 'Settings'. On the left, under 'Deployment method', it says 'App connected to GitHub' and lists 'Code diffs, manual and auto deploys are available for this app.' In the center, there's a section about pipelines: 'Add this app to a pipeline' and 'Add this app to a stage in a pipeline to enable additional features'. It shows icons for connecting multiple apps and for GitHub Pipelines. Below this is a dropdown menu 'Choose a pipeline'. To the right, there are sections for 'Heroku Git' (using Heroku CLI), 'GitHub Connected' (with a green checkmark), and 'Container Registry' (using Heroku CLI). Under 'Connected to', it says 'Connected to' and shows 'Releases in the activity feed link to GitHub to view commit diffs' and 'Automatically deploys from dev'. A 'Disconnect...' button is also present. In the bottom right corner of this section, there's a note: 'You can now change your main deploy branch from "master" to "main" for both manual and automatic deploys, please follow the instructions here.' Below this, under 'Automatic deploys', it says 'Enables a chosen branch to be automatically deployed to this app.' and has a checkbox for 'Automatic deploys from dev are enabled'. There's also a note about pushing to 'dev' triggering automatic deploys. A 'Disable Automatic Deploys' button is available. At the bottom, under 'Manual deploy', it says 'Deploy a GitHub branch' and 'Choose a branch to deploy' with a dropdown set to 'dev' and a 'Deploy Branch' button.

Slika 5.2: Prikaz aplikacije na poslužitelju Heroku



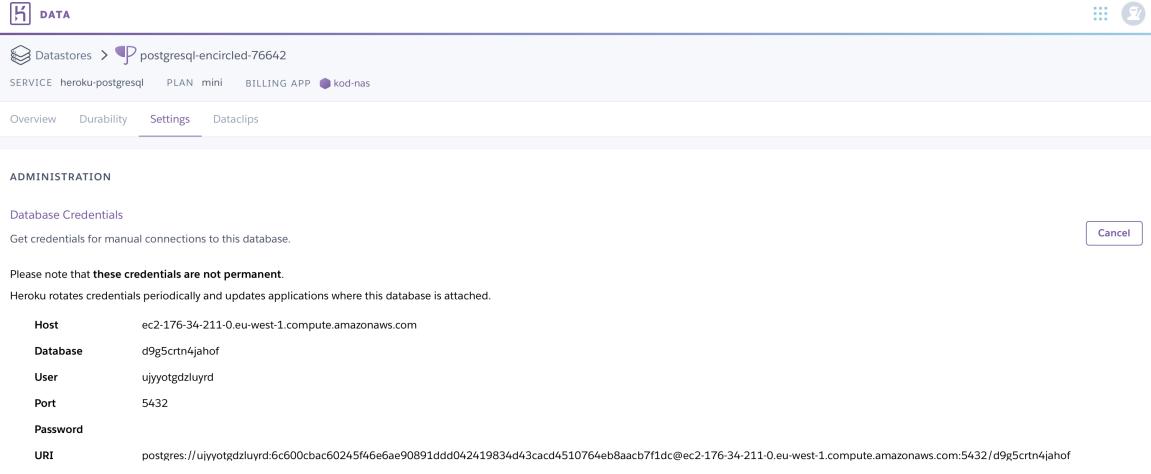
The screenshot shows a code editor interface with a dark theme. At the top, there is a navigation bar with a dropdown menu set to 'dev' and a path 'hotel-web / src / environments / environment.prod.ts /'. Below the navigation bar, there is a large empty workspace area. In the bottom left corner of the workspace, there is a status message: '4 lines (4 sloc) | 102 Bytes'. The main content of the editor is a single file named 'environment.prod.ts' containing the following code:

```

1  export const environment = {
2    production: true,
3    baseApiUrl: 'https://kod-nas.herokuapp.com/api'
4  };

```

Slika 5.3: Povezivanje frontenda i backenda



The screenshot shows the Heroku Datastore settings page for a PostgreSQL service. The top navigation bar includes icons for Datastores, heroku-postgresql, PLAN mini, BILLING APP, and kod-nas. Below the navigation, there are tabs for Overview, Durability, Settings (which is selected), and Dataclips. The main section is titled 'ADMINISTRATION' and contains a 'Database Credentials' section. It instructs users to get credentials for manual connections and notes that credentials are not permanent. It also states that Heroku rotates credentials periodically. The 'Host' is listed as ec2-176-34-211-0.eu-west-1.compute.amazonaws.com, 'Database' as d9g5crt4jaho, 'User' as ujyyotgdzluryd, 'Port' as 5432, and 'URI' as postgres://ujyyotgdzluryd:6c600cbac6024546e6ae90891ddd042419834d43cacd4510764eb8aacb7f1dc@ec2-176-34-211-0.eu-west-1.compute.amazonaws.com:5432/d9g5crt4jaho. A 'Cancel' button is visible in the top right corner.

Slika 5.4: Podaci za pristup bazi

## 6. Zaključak i budući rad

Zadatak naše grupe je bio razvoj web aplikacije za upravljanje radom hotela uz mogućnost upravljanja uslugama iznajmljivanja hotelskog prostora i dvije velike dvorane za posebne potrebe, cijenom najma smještajnih jedinica, prikazom zauzeća smještajnih jedinica i potreba za posebnim uslugama po pojedinom danu. Provedba projekta tekla je kroz dva ciklusa:

Prvi ciklus projekta uključivao je okupljanje tima za razvoj aplikacije i dodjelu projektnog zadatka. Intenzivno smo radili na pronalaženju i dokumentiranju zahjeva sustava i izlučivanju svih obrazaca uporabe, te izradi **UML** dijagrama istih.

U drugom ciklusu, samostalni rad na projektu je bio puno intenzivniji i puno veći naglasak je bio na samostalnom učenju korištenja radnih okvira i sustava za verzioniranje softvera, pošto je se većina članova grupe prvi put susrela s njima. Radilo se i na ispravljanju grešaka u prvotnoj verziji dokumentacije, te dodavanju novih **UML** dijagrama u dokumentaciju.

Komunikacija tima se održavala putem više platformi. Dodjela zadataka članovima tima je bila održivana kroz **Trello**, a većina komunikacije se održavala putem **Discorda** i **WhatsApp Messenger**. Sudjelovanje u projektu je dopridonijelo vrijedno iskustvo svakom od članova tima na području grupnog rada na istom projektu. To iskustvo bi nam sigurno bilo od koristi prije početka rada na projektu, te bismo sigurno kroz izradu projekta plovili brže, sigurnije i organiziranije. U budućnosti bi se na aplikaciji moglo poraditi na organizaciji menija u podmenije i općenito na estetskoj nadogradnji aplikacije.

# Popis literature

## *Kontinuirano osvježavanje*

1. Projektni zadatak [https://www.fer.unizg.hr/\\_download/repository/MK3\\_-\\_Hotelsko\\_poslovanje.pdf](https://www.fer.unizg.hr/_download/repository/MK3_-_Hotelsko_poslovanje.pdf)
2. Programsko inženjerstvo, FER ZEMRIS, <http://www.fer.hr/predmet/proinzh>
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. I. Marsic, Software engineering book“, Department of Electrical and Computer Engineering, Rutgers University, <http://www.ece.rutgers.edu/~marsic/books/SE>
6. The Unified Modeling Language, <https://www.uml-diagrams.org/>
7. Astah Community, <http://astah.net/editions/uml-new>
8. Spring Boot tutorials <https://www.baeldung.com>

# Indeks slika i dijagrama

3.1	Korisnik . . . . .	16
3.2	Gost . . . . .	16
3.3	Sobarica/Spremačica . . . . .	17
3.4	Djelatnik na recepciji . . . . .	17
3.5	Administrator . . . . .	18
3.6	Vlasnik . . . . .	18
3.7	Sekvencijski dijagram za UC4 . . . . .	19
3.8	Sekvencijski dijagram za UC6 . . . . .	20
3.9	Sekvencijski dijagram za UC11 . . . . .	21
4.1	ER dijagram baze podataka . . . . .	29
4.2	Dijagram razreda - Kontroleri . . . . .	30
4.3	Dijagram razreda - DTO modeli . . . . .	31
4.4	Dijagram razreda - Domenski modeli . . . . .	32
4.5	Dijagram razreda - Repository sloj . . . . .	33
4.6	Dijagram razreda - Service sloj . . . . .	34
4.7	Registracija gosta . . . . .	35
4.8	Pregled sobe . . . . .	37
4.9	Dijagram komponenti . . . . .	39
5.1	Dijagram razmještaja . . . . .	48
5.2	Prikaz aplikacije na poslužitelju Heroku . . . . .	50
5.3	Povezivanje frontenda i backenda . . . . .	51
5.4	Podaci za pristup bazi . . . . .	51
6.1	Dijagram Promjena . . . . .	59

# Dodatak: Prikaz aktivnosti grupe

## Dnevnik sastajanja

### *Kontinuirano osvježavanje*

#### 1. sastanak

- Datum: 15.10.2022.
- Prisustvovali: Svi
- Teme sastanka:
  - Rasprava o temi projekta
  - Raspisivanje projektnog zadatka

#### 2. sastanak

- Datum: 20.10.2022.
- Prisustvovali: Svi
- Teme sastanka:
  - Raspored u timove
  - Raspored dužnosti svakog člana tima

#### 3. sastanak

- Datum: 16.11.2022.
- Prisustvovali: Svi
- Teme sastanka:
  - Komentar generičkih funkcija
  - Rasprava o dalnjem napretku aplikacije

#### 4. sastanak

- Datum: 5.12.2022.
- Prisustvovali: Svi
- Teme sastanka:
  - Zaduženja do predaje Alpha inačice
  - Retrospektiva na efikasnost tima do sastanka
  - Grupno testiranje aplikacije

5. sastanak

- Datum: 20.12.2022.
- Prisustvovali: Svi
- Teme sastanka:
  - Komentiranje Alpha verzije aplikacije
  - Rasprava o dodatnim mogućnostima aplikacije

6. sastanak

- Datum: 3.1.2023.
- Prisustvovali: Svi
- Teme sastanka:
  - Rasprava o završnoj fazi projekta
  - Raspoređivanje dužnosti

7. sastanak

- Datum: 11.1.2023.
- Prisustvovali: Svi
- Teme sastanka:
  - Komentiranje cjelokupnog projekta
  - Dogovori oko završnih akcija projekta

## Tablica aktivnosti

### *Kontinuirano osvježavanje*

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

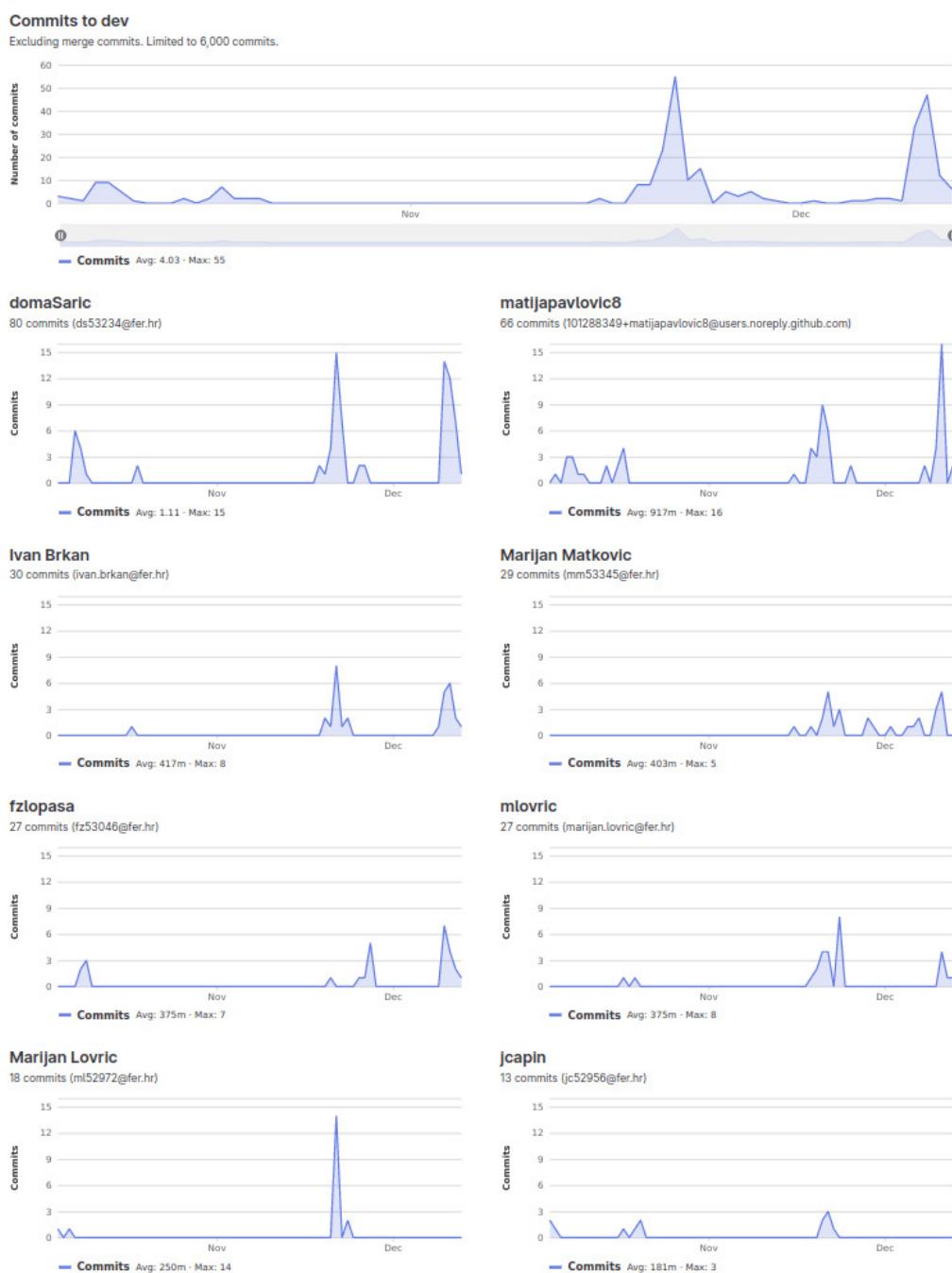
	Marijan Lovrić	Ivan Brkan	Dominik Šarić	Filip Zlopaša	Matija Pavlović	Marijan Matković	Jurica Čapin
Upravljanje projektom	10	2	2	1	5	1	7
Opis projektnog zadatka	1	2	5	2	1	1	5
Funkcionalni zahtjevi	2	1	6	5	6	5	4
Opis pojedinih obrazaca	2	1	20	20	10	2	2
Dijagram obrazaca	0	0	20	20	0	0	0
Sekvencijski dijagrami	0	0	1	0	0	5	0
Opis ostalih zahtjeva	2	0	2	2	1	2	2
Arhitektura i dizajn sustava	15	7	0	15	1	5	5
Baza podataka	10	10	0	0	0	0	10
Dijagram razreda	4	0	0	0	0	0	0
Dijagram stanja	0	0	0	0	0	0	2
Dijagram aktivnosti	0	0	0	0	0	0	3
Dijagram komponenti	0	0	0	0	0	0	2
Korištene tehnologije i alati	0	5	6	2	2	4	2
Ispitivanje programskog rješenja	7	10	8	10	10	2	0
Dijagram razmještaja	0	0	0	0	0	7	0
Upute za puštanje u pogon	2	2	0	0	0	0	0

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

	Marijan Lovrić	Ivan Brkan	Dominik Šarić	Filip Zlopša	Matija Pavlović	Marijan Matković	Jurica Čapin
Dnevnik sastajanja	3	1	1	1	0	0	2
Zaključak i budući rad	1	1	1	5	2	2	2
Popis literature	1	1	1	0	0	0	1
Izrada baze podataka	10	10	0	0	0	10	5
Deploy	20	20	0	0	0	0	50
Back end	160	180	0	0	0	120	100
Front end	4	6	120	120	180	1	5

# Dijagrami pregleda promjena



Slika 6.1: Dijagram Promjena