

# Programsko inženjerstvo ak.god 2025./2026

---

Sveučilište u Zagrebu

Fakultet elektrotehnike i računarstva

Naziv projekta: Julius Meinl

---

Tim: <TG03.3>

Julius Meinl

Nastavnik: Vlado Sruk

## Uvod

---

Današnji hoteli zahtijevaju učinkovit informacijski sustav. Cilj je gostima omogućiti jednostavno i brzo rezerviranje soba, ali i dodatnog sadržaja koji hotel nudi. S druge strane, hotelu su neophodne informacije o poslovanju, mogućnost komunikacije s gostima, upravljanje smještajem te optimizacija raspodjele soba.

Digitalizacija hotelskog poslovanja ključna je za opstanak na modernom tržištu. Popularni načini rezervacija preko posrednika stavlju naglasak na količinu različitih opcija i nemoguće je ostvariti izravan odnos s gostom i prikazati cijelokupnu ponudu.

Cilj ove aplikacije je pružiti personalizirani pristup koji objedinjuje sve funkcionalnosti hotelskog poslovanja u jedinstveni sustav. Gostima omogućuje jednostavno rezerviranje smještaja i dodatnih usluga, dok vlasnicima i osoblju pruža mogućnost upravljanja kapacitetima, praćenje statistike i donošenje poslovnih odluka na temelju podataka. Hotel dobiva dimenziju neovisnosti, smanjuju se troškovi za provizije posrednicima te se izgrađuju dugoročni odnosi s korisnicima.

## Slične stranice

---

[Villa Dubrovnik](#)



# Popis smještaja

[ALL](#)[ROOMS](#)[SUITES](#)[RESIDENCES](#)

## Superior Sea View Room

Uživajte u zadivljujućem pogledu na Jadransko more s panoramskog balkona sofisticirane dizajnerske sobe.

[Rezervirajte](#)[Saznajte više](#)

## Deluxe Sea View Room

Uživajte u pogledu iz prvog reda na bezvremenSKI šarm Dubrovnika iz dizajnerske sobe.

[Rezervirajte](#)[Saznajte više](#)

Web stranica hotela Villa Dubrovnik nudi mogućnost rezervacija i biranja konkretnе vrste sobe. Na primjer "Superior Sea View Room" ili "Beleca Suite". Nema prostora za iskazivanje preferencija gosta, odnosno kakvu sobu želi i što mu je bitno da soba ima pa da na osnovu toga sustav odluci koja soba je optimalna za njega te skuplja informacije na osnovu kojih bi hotel mogao promijeniti ponudu s obzirom na potražnju. Stranica ima bogatu galeriju. Fokusira se na ponudu Dubrovnika i njegovu povijest, ali pruža i informacije o vlastitim uslugama. Ne postoji izravan način komunikacije ni FAQ. Za stupanje u kontakt, gost mora poslati mail ili zvati na broj.

[\*\*Bluesun Hotel Elaphusa\*\*](#)



Sličan princip rezervacija kao i hotel Villa Dubrovnik. Ne postoji opcija filtriranja smještaja stoga biranje željene sobe zahtijeva vrijeme i proučavanje svih ponuđenih vrsta soba, kojih je nemali broj. Prikazuje dio s posebnom ponudom gdje gosti mogu iskoristiti popuste na razne aktivnosti, večere, smještaj... Postoji dio s FAQ, kao i dio gdje se izravno može prijaviti na natječaj za rad ili poslati otvorena molba za rad u hotelu.

## Korisnici i koristi

Potencijalna korist aplikacije je donošenje promjena u trenutni princip hotelskog poslovanja. Ovaj projekt integrira sve bitne funkcionalnosti u jedinstvenu aplikaciju. Hoteli mogu bolje predstaviti što nude potencijalnim gostima u vlastitoj aplikaciji nasuprot trenutnim popularnim rješenjima gdje su samo jedan od više hotela u ponudi te je količina informacija o njima ograničena. Također, aplikacija daje novu dimenziju poslovanju jer proučavanjem statističkih podataka koje skuplja o gostima i njihovim preferencijama, vlasnici mogu donositi bolje odluke za budućnost poslovanja hotela.

Korisnici koji bi mogli biti zainteresirani za aplikaciju

- Gosti

Preferiraju aplikaciju koja je fokusirana na jedan hotel i pruža sve potrebne informacije kako bi mogli biti sigurni kakav smještaj su odabrali. Žele mogućnost biranja smještaja i dodatnog sadržaja prema sklonostima. U slučaju problema mogu kontaktirati podršku pomoći chatbota ili sličnog alata.

- Vlasnici

Ne žele plaćati proviziju posrednicima u rezervaciji smještaja. Trebaju aplikaciju koja je jedinstvena za njihov hotel i pruža

uvid u statistiku zauzeća, sezonske trendove i strukturu gostiju prema zemlji i/ili gradu dolaska. Identifikacija najtraženijih dodatnih usluga i mogućnost prilagodbe ponude. Mogućnost prilagodbe sustava specifičnim poslovnim potrebama i strategijama hotela.

## Opseg projektnog zadatka

Projekt obuhvaća razvoj web aplikacije koja spaja sve ključne funkcionalnosti hotelskog poslovanja u jedinstveni informacijski sustav. Sustav mora omogućiti učinkovito upravljanje rezervacijama, smještajem, korisnicima i statistikom poslovanja, uz podršku za višekorisnički rad i responzivan dizajn prilagođen različitim uređajima.

Prijava i registracija korisnika OAuth2 servisom. Upravljanje korisničkim računima te različite razine pristupa za goste, osoblje, administratore i vlasnike hotela.

Odabir i rezervacija sobe prema preferencijama i trenutnoj dostupnosti. Rezervacija dodatnog sadržaja u ponudi hotela (*nije nužna rezervacija sobe da bi se koristio dodatni sadržaj hotela*). Sam proces dodjelivanja konkretnе sobe mora biti optimiziran tako da broj praznih dana u hotelu bude minimalan. Potrebna je mogućnost plaćanja internet bankarstvom i potvrda rezervacije.

Aplikacija pruža integraciju s Google Maps servisom radi prikaza točne lokacije hotela i olakšanog planiranja dolaska gostiju. Responzivan dizajn prilagođen različitim uređajima te pregled izgleda hotela i soba putem prikaza slika i galerija.

Djelatnici hotela s potrebnim privilegijama mogu pregledavati i unositi rezervacije u ime gostiju. Upravljanje dostupnošću soba i dodatnih sadržaja trenutno ili trajno. Automatsko ažuriranje dostupnosti u stvarnom vremenu.

Pregled statistike o zauzeću smještaja, strukturi gostiju i korištenju dodatnih usluga te izvoz podataka u pdf, xml i xlsx formatu, kao i izvoz podataka statistike u pdf ili xlsx formatu. Pristup analitičkim podacima za vlasnika radi donošenja poslovnih odluka.

Korisnici imaju mogućnost kontaktiranja korisničke podrške putem integriranog chat sustava ili chatbot alata kao i mogućnost ostavljanja recenzije nakon boravka u hotelu.

## Moguće nadogradnje i prilagodbe

Aplikacija je osmišljena kao platforma za digitalno upravljanje hotelskim poslovanjem koja se može unaprjeđivati, nadograđivati i prilagođavati potražnji i potrebama korisnika, novim tehnologijama ili drugim promjenama unutar hotela ili izvan hotela koja zahtijevaju i određenu promjenu aplikacije.

- Pružanje prikaza aplikacije na više jezika kako bi pridobili više gostiju.
- Sustav nagrada za stare goste. Na primjer princip posebne ponude kao hotel Bluesun hotel Elaphusa koji nudi dio s posebnim pogodnostima i akcijama. Slična ideja bi se mogla primjeniti kao sustav nagrade za vjerne goste kako bi ih potaknuli da nastave birati hotel iznova.
- Integriranje dodatnih funkcionalnosti u aplikaciju poput digitalnog ključa za sobu, parking ili slično.
- Mogućnost izravne komunikacije s osobljem tokom radnog vremena funkcijama aplikacije za narudžbu hrane, pića, higijenskih potrepština te mogućnost prijave kvara ili za dodatna pitanja i informacije.

## Funkcionalni zahtjevi

ID zahtjeva	Opis	Prioritet	Izvor	Kriterij prihvatanja
-------------	------	-----------	-------	----------------------

ID zahtjeva	Opis	Prioritet	Izvor	Kriterij prihvatanja
F-001	Sustav omogućuje korisnicima prijavu u sustav pomoću OAuth 2.0 standarda.	Visok	Zahtjev dionika	Korisnik se može prijaviti putem Google računa.
F-002	Sustav omogućuje odabir i rezervaciju dostupnih soba.	Visok	Zahtjev dionika	Korisnik može odabrati i rezervirati sobu ovisno o dostupnosti.
F-003	Sustav omogućuje rezervaciju dodatnog sadržaja koji je dostupan.	Srednji	Zahtjev dionika	Korisnik može rezervirati dodatni sadržaj samostalno ili zajedno s rezervacijom sobe, a sustav šalje potvrdu rezervacije.
F-004	Sustav omogućuje plaćanje rezervacije sobe putem interneta.	Srednji	Zahtjev dionika	Korisnik može odabrati plaćanje putem online bankarstva ili pri prijavi u hotel.
F-005	Sustav omogućuje pregled lokacije i izgleda hotela.	Visok	Postojeći sustav	Korisnik može vidjeti lokaciju hotela putem integrirane karte i pregledavati slike hotela i soba u aplikaciji.
F-006	Sustav omogućuje kontaktiranje korisničke podrške.	Srednji	Zahtjev dionika	Korisnik može kontaktirati korisničku podršku putem chatбота ili drugog sličnog alata.
F-007	Sustav omogućuje pregled i rezervacije soba.	Visok	Zahtjev dionika	Korisnik s privilegijama može rezervirati sobu i pregledavati sve rezervacije i podatke o gostima.
F-008	Sustav omogućuje pregled statistike.	Visok	Zahtjev dionika	Korisnik s privilegijama može pristupiti i pregledavati sve statističke podatke hotela.
F-009	Sustav omogućuje izvoz statistike i podataka.	Visok	Zahtjev dionika	Korisnik može izvesti sve potrebne podatke i statistike hotela u obliku PDF-a, CSV-a, XML-a i XLSX-a.
F-010	Sustav omogućuje upravljanje dostupnošću soba i dodatnog sadržaja.	Visok	Zahtjev dionika	Korisnik s privilegijama može postaviti dostupnost soba i dodatnog sadržaja za određeni period ili neodređeno vrijeme.
F-011	Sustav omogućuje upravljanje korisničkim računima.	Srednji	Zahtjev dionika	Korisnik s privilegijama može upravljati korisničkim računima, uključujući brisanje računa i pregled povijesti rezervacija.
F-012	Sustav omogućuje ostavljanje recenzije hotela.	Srednji	Zahtjev dionika	Korisnik može ostaviti recenziju nakon završetka boravka u hotelu ili korištenja dodatne usluge.
F-013	Sustav omogućuje pregled osobnih podataka i profila.	Srednji	Zahtjev dionika	Korisnik može pregledati svoje podatke (ime, kontakt, rezervacije).
F-014	Sustav omogućuje uređivanje osobnih podataka i profila.	Srednji	Zahtjev dionika	Korisnik može urediti svoje podatke, a promjene se uspješno spremaju i prikazuju.

## Ostali zahtjevi

### Zahtjevi za održavanje

ID zahtjeva	Opis	Prioritet
M-001	Sustav treba biti dizajniran tako da omogućuje jednostavno održavanje. Svaka izmjena u kodu treba biti dokumentirana, a sam kod treba imati jasne komentare.	Srednji
M-002	Sustav treba imati dokumentirane funkcionalne i nefunkcionalne zahtjeve, te arhitekturu pomoću dijagrama: obrasci uporabe, sekvencijski dijagrami, dijagrami razreda, dijagrami komponenti, aktivnosti i razmještaja.	Visok

ID zahtjeva	Opis	Prioritet
M-003	Sustav treba biti popraćen Wiki dokumentacijom koja sadrži sve informacije vezane za sustav i način uporabe.	Visok

## Nefunkcionalni zahtjevi

ID zahtjeva	Opis	Prioritet
NF-001	Sustav mora ograničiti korisnike na pristup samo resursima za koje imaju ovlasti.	Visok
NF-002	Sustav treba podržavati autentifikaciju putem OAuth2 sustava.	Visok
NF-003	Sustav treba ograničiti dostupnost usluga i funkcija ovisno o njihovoj stvarnoj dostupnosti.	Visok
NF-004	Sustav treba prikazivati raspoloživost u stvarnom vremenu.	Visok
NF-005	Sustav treba omogućiti nesmetani rad više korisnika istovremeno.	Visok
NF-006	Sustav treba biti dostupan najmanje 99% vremena.	Visok
NF-007	Sustav treba raditi na svim većim web preglednicima (Mozilla Firefox, Chrome, Edge).	Srednji
NF-008	Sustav treba imati responzivan dizajn za različite veličine ekrana.	Visok

## Dionici

### 1. Korisnici sustava (D-1)

- Gost
- Korisnik (registrirani korisnik)

### 2. Osoblje (D-2)

- Repcionist
- Upravitelj

### 3. Administratori sustava (D-3)

- Administrator

### 4. Razvojni tim (D-4)

- Razvojni tim

### 5. Naručitelji sustava (D-5)

- Naručitelji

## Aktori

### A-1 Gost (Inicijator)

- Kreirati račun preko OAuth2 sustava (F-001)
- Pregled lokacije i izgleda hotela (F-005)

## A-2 Korisnik (Inicijator)

- Login u račun preko OAuth2 sustava (F-001)
- Odabir i rezervacija soba (F-002)
- Rezervacija dodatnog sadržaja (F-003)
- Plaćanje rezervacije preko internet bankarstva (F-004)
- Pregled lokacije i izgleda hotela (F-005)
- Kontaktiranje korisničke podrške (F-006)
- Ostavljanje recenzije hotela (F-012)

## A-3 Receppcionist (Inicijator)

- Pregled i rezervacija soba (F-007)

## A-4 Upravitelj (Inicijator)

- Pregled statistike (F-008)
- Izvoz statistike i podataka (F-009)
- Upravljanje dostupnošću soba i dodatnog sadržaja (F-010)

## A-5 Map servis (Sudionik)

- Posluživanje interaktivne karte (F-005)

## A-6 Servis za login (Sudionik)

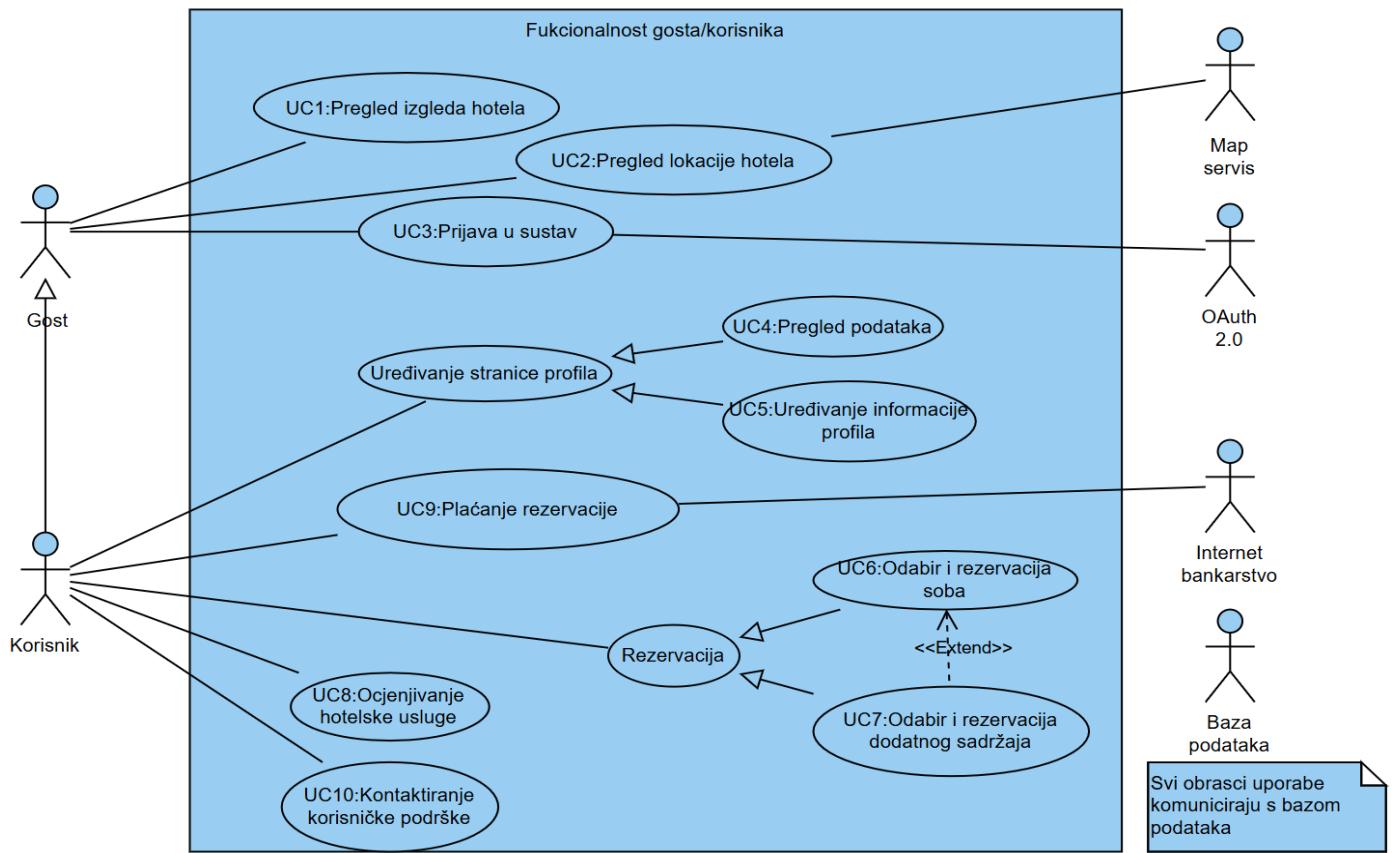
- Omogućavanje prijave i registracije korisnika (F-001)

## A-7 Baza podataka (Sudionik)

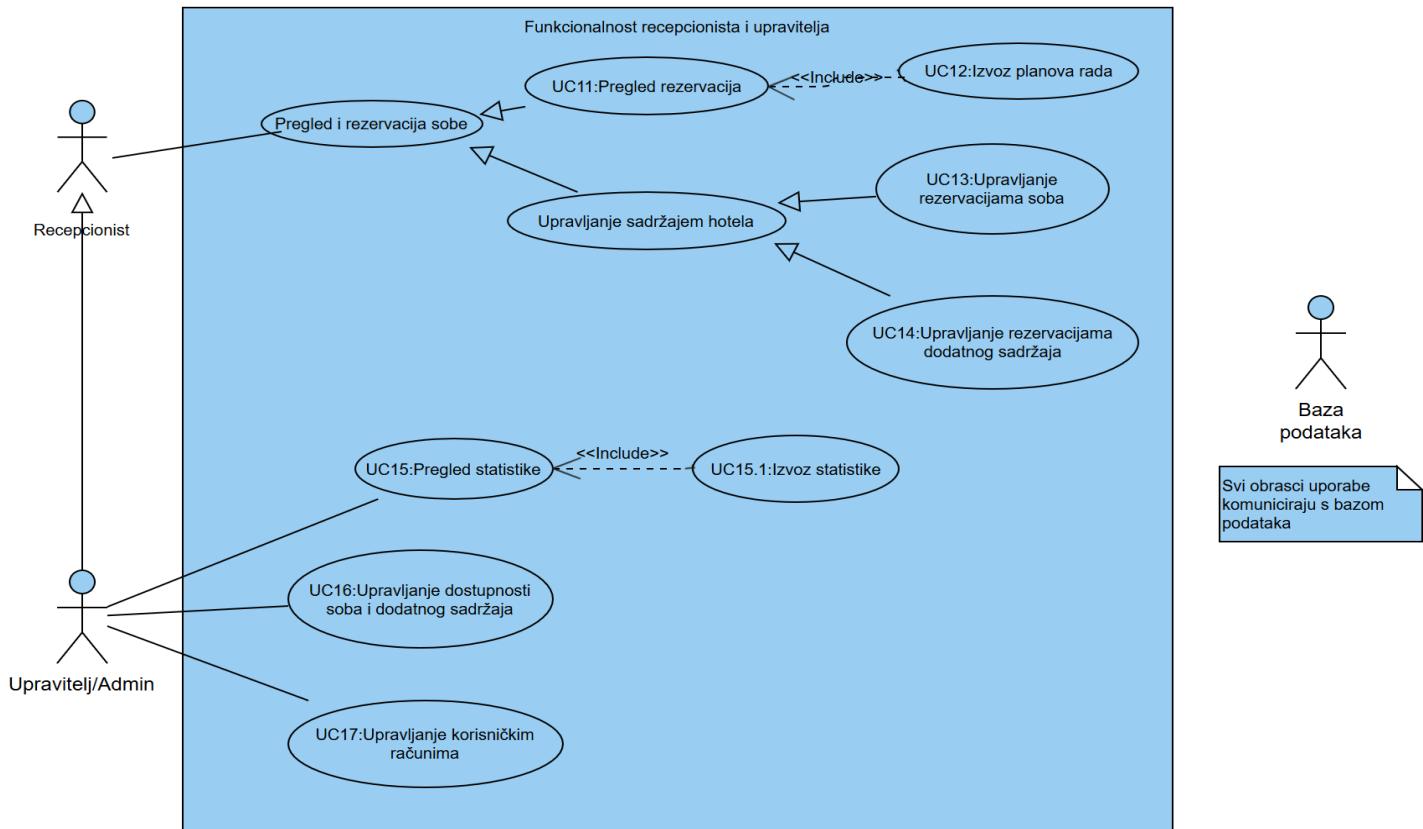
- Sudjeluje u svim obrascima upotrebe pohranjivanjem ili dohvatom relevantnih podataka.

# Dijagram obrazaca uporabe

---



Slika 3.1. Funkcionalnost gosta i korisnika



Slika 3.2. Funkcionalnost recepcionista i upravitelja

## Obrasci uporabe

## **UC1 - Pregled izgleda hotela**

**Namjena:** Omogućiti gostu pregled fotografija i opisa hotela

**Glavni aktor:** Gost

**Preduvjeti:** Gost ima pristup web stranici hotela

**Opis osnovnog tijeka:**

1. Gost pristupa web stranici
2. Odabire opciju "Pregled izgleda hotela"
3. Sustav prikazuje galeriju slika i videa hotela te osnovne informacije o objektu

**Opis mogućih odstupanja:**

Ako slike ili video nisu dostupni, sustav prikazuje obavijest o nedostupnosti sadržaja

## **UC2 - Pregled lokacije hotela**

**Namjena:** Omogućiti gostu pregled lokacije hotela putem povezanog map servisa

**Glavni aktor:** Gost

**Preduvjeti:** Nema posebnih preduvjeta

**Opis osnovnog tijeka:**

1. Gost odabire opciju "Pregled lokacije hotela"
2. Sustav se povezuje s map servisom
3. Na karti se prikazuje lokacija hotela

**Opis mogućih odstupanja:** -

## **UC3 - Prijava u sustav**

**Namjena:** Omogućiti korisniku autentifikaciju putem OAuth 2.0 sustava

**Glavni aktor:** Gost

**Preduvjeti:** -

**Opis osnovnog tijeka:**

1. Korisnik odabire opciju "Prijava"
2. Unosi korisničke podatke (e-mail i lozinku)
3. Sustav šalje zahtjev UAuth 2.0 servisu
4. UAuth 2.0 potvrđuje identitet korisnika
5. Sustav omogućuje pristup korisničkom profilu

**Opis mogućih odstupanja:**

- Pogrešni podaci za prijavu → prikaz poruke o pogrešci
- Neuspješna komunikacija s UAuth 2.0 → obavijest o tehničkom problemu

## UC4 - Pregled podataka

**Namjena:** Omogućiti korisniku pregled osobnih i rezervacijskih podataka

**Glavni aktor:** Korisnik

**Preduvjeti:** Korisnik je prijavljen u sustav

**Opis osnovnog tijeka:**

1. Korisnik pristupa svom profilu
2. Sustav prikazuje korisničke i rezervacijske podatke

**Opis mogućih odstupanja:** Ako podaci nisu dostupni u bazi, sustav prikazuje poruku o nedostupnosti

## UC5 - Uređivanje informacije profila

**Namjena:** Omogućiti korisniku izmjenu osobnih podataka na profilu

**Glavni aktor:** Korisnik

**Preduvjeti:** Korisnik je prijavljen u sustav

**Opis osnovnog tijeka:**

1. Korisnik odabire "Uredi profil"
2. Mijenja željene informacije (ime, kontakt, lozinku itd.)
3. Sustav ažurira podatke u bazi

**Opis mogućih odstupanja:** Nema predviđenih odstupanja

## UC6 - Odabir i rezervacija soba

**Namjena:** Omogućiti korisniku odabir i rezervaciju soba

**Glavni aktor:** Korisnik

**Preduvjeti:** Korisnik je prijavljen i pregledao ponudu soba

**Opis osnovnog tijeka:**

1. Korisnik odabire željeni tip sobe
2. Odabire datume dolaska i odlaska
3. Sustav provjerava dostupnost sobe
4. Potvrđuje rezervaciju
5. Sustav pohranjuje podatke o rezervaciji

**Opis mogućih odstupanja:**

- Odabrana soba nije dostupna u traženom terminu → sustav prikazuje obavijest i nudi alternativne opcije.
- Greška u bazi ili komunikaciji sa sustavom → prikazuje se poruka o tehničkom problemu.
- Korisnik prekida rezervaciju prije potvrde → rezervacija se ne spremi.

## UC7 - Odabir i rezervacija dodatnog sadržaja

**Namjena:** Omogućiti korisniku odabir dodatnih usluga (bazen, restoran, teren itd.)

**Glavni aktor:** Korisnik

**Preduvjeti:** Korisnik je prijavljen u sustav

**Opis osnovnog tijeka:**

1. Korisnik odabire dodatne sadržaje
2. Sustav provjerava dostupnost sadržaja
3. Sustav dodaje odabrane stavke rezervaciji
4. Korisnik potvrđuje izbor.

**Opis mogućih odstupanja:**

- Odabrani sadržaj nije dostupan → sustav prikazuje obavijest i nudi alternativne opcije.
- Greška u sustavu ili bazi → prikazuje se poruka o tehničkom problemu.
- Korisnik prekida izbor prije potvrde → odabrani sadržaji se ne pohranjuju.

## UC8 - Ocjenjivanje hotelske usluge

**Namjena:** Omogućiti korisniku ostavljanje recenzije i ocjene

**Glavni aktor:** Korisnik

**Preduvjeti:** Korisnik ima realiziranu rezervaciju

**Opis osnovnog tijeka:**

1. Korisnik otvara sekciju "Ocijeni hotel"
2. Unosi ocjenu i komentar
3. Sustav pohranjuje ocjenu u bazu

**Opis mogućih odstupanja:**

## UC9 - Plaćanje rezervacije

**Namjena:** Omogućiti korisniku plaćanje putem internet bankarstva

**Glavni aktor:** Korisnik

**Preduvjeti:** Korisnik ima potvrđenu rezervaciju i pristup internet bankarstvu

**Opis osnovnog tijeka:**

1. Korisnik odabire opciju "Plati rezervaciju"

2. Sustav se povezuje s internet bankarstvom

3. Korisnik potvrđuje plaćanje

4. Sustav bilježi status plaćanja

#### **Opis mogućih odstupanja:**

- Plaćanje odbijeno → korisniku se prikazuje poruka i nudi ponovno plaćanje
- Greška u komunikaciji s bankom → obavijest o tehničkom problemu

## **UC10 - Kontaktiranje korisničke podrške**

**Namjena:** Omogućiti korisniku slanje upita ili prijavu problema podršci

**Glavni aktor:** Korisnik

**Preduvjeti:** Korisnik je prijavljen ili ima pristup kontakt formi

#### **Opis osnovnog tijeka:**

1. Korisnik odabire opciju "Kontaktiraj podršku"

2. Unosi upit i šalje ga

3. Sustav proslijeđuje poruku timu podrške

4. Korisnik dobiva potvrdu o uspješnom slanju

#### **Opis mogućih odstupanja:**

## **UC11 - Pregled rezervacija**

**Namjena:** Omogućiti recepcionistu pregled svih postojećih rezervacija

**Glavni aktor:** Recepcionist

**Preduvjeti:** Korisnik je prijavljen u sustav

#### **Opis osnovnog tijeka:**

1. Recepcionist odabire opciju "Pregled rezervacija"

2. Sustav dohvaća podatke o svim aktivnim, otkazanim i nadolazećim rezervacijama iz baze podataka

3. Prikazuje se lista rezervacija s osnovnim detaljima (gost, datum dolaska/odlaska, status)

**Opis mogućih odstupanja:** Ako nema rezervacija, prikazuje se obavijest "Nema dostupnih rezervacija"

## **UC12 - Izvoz planova rada**

**Namjena:** Omogućiti recepcionistu izvoz rasporeda rada i zauzetosti soba u datoteku

**Glavni aktor:** Recepcionist

**Preduvjeti:** Recepcionist ima pristup pregledima rezervacija (UC11)

#### **Opis osnovnog tijeka:**

1. Repcionist otvara pregled rezervacija
2. Odabire opciju "Izvezi plan rada"
3. Sustav generira i nudi preuzimanje datoteke s planom zauzetosti

**Opis mogućih odstupanja:** Ako dođe do greške prilikom generiranja datoteke, prikazuje se poruka o pogrešci

## UC13 – Upravljanje rezervacijama soba

**Namjena:** Omogućiti recepcionistu dodavanje, izmjenu ili otkazivanje rezervacija soba

**Glavni aktor:** Repcionist

**Preduvjeti:**

**Opis osnovnog tijeka:**

1. Repcionist otvara modul "Upravljanje rezervacijama soba"
2. Pregledava postojeće rezervacije ili kreira novu
3. Sustav ažurira podatke u bazi i potvrđuje uspješnu promjenu

**Opis mogućih odstupanja:** Ako soba nije dostupna u traženom terminu, sustav prikazuje upozorenje

## UC14 – Upravljanje rezervacijama dodatnog sadržaja

**Namjena:** Omogućiti recepcionistu ili upravitelju da upravljaju rezervacijama dodatnih usluga

**Glavni aktor:** Repcionist/Upravitelj

**Preduvjeti:** Korisnik je prijavljen u sustav

**Opis osnovnog tijeka:**

1. Repcionist odabire rezervaciju sobe
2. Otvara sekciju "Dodatni sadržaji"
3. Dodaje, mijenja ili briše rezervacije dodatnih usluga
4. Sustav pohranjuje promjene u bazu

**Opis mogućih odstupanja:** Ako odabrani sadržaj nije dostupan, sustav prikazuje upozorenje

## UC15 – Pregled statistike

**Namjena:** Omogućiti upravitelju pregled poslovnih i operativnih statistika (popunjeno, prihodi, broj gostiju itd.)

**Glavni aktor:** Upravitelj

**Preduvjeti:** Korisnik je prijavljen u sustav

**Opis osnovnog tijeka:**

1. Upravitelj otvara modul "Pregled statistike"
2. Sustav dohvata i prikazuje relevantne podatke iz baze

3. Korisnik može filtrirati statistike prema datumu, tipu sobe ili periodu

**Opis mogućih odstupanja:** Ako podaci nisu dostupni, prikazuje se obavijest o nedostupnosti

## UC15.1 – Izvoz statistike

**Namjena:** Omogućiti upravitelju izvoz statističkih podataka u datoteku

**Glavni aktor:** Upravitelj

**Preduvjeti:** Pregled statistike (UC15) je uspješno izvršen

**Opis osnovnog tijeka:**

1. Upravitelj odabire opciju "Izvezi statistiku"
2. Sustav generira datoteku (PDF, Excel) sa statistikom
3. Upravitelj preuzima datoteku

**Opis mogućih odstupanja:** Ako nema dostupnih podataka za izvoz, prikazuje se poruka o grešci

## UC16 – Upravljanje dostupnošću soba i dodatnog sadržaja

**Namjena:** Omogućiti upravitelju upravljanje dostupnošću soba i dodatnih hotelskih sadržaja

**Glavni aktor:** Upravitelj

**Preduvjeti:** Upravitelj je prijavljen

**Opis osnovnog tijeka:**

1. Upravitelj pristupa modulu "Dostupnost soba i sadržaja"
2. Označava koje su sobe i usluge dostupne u određenom periodu
3. Sustav ažurira stanje u bazi podataka

**Opis mogućih odstupanja:** Ako sustav ne može ažurirati podatke, prikazuje se obavijest

## UC17 – Upravljanje korisničkim računima

**Namjena:** Omogućiti upravitelju dodavanje, izmjenu i brisanje korisničkih računa (receppcionista i korisnika)

**Glavni aktor:** Upravitelj

**Preduvjeti:** -

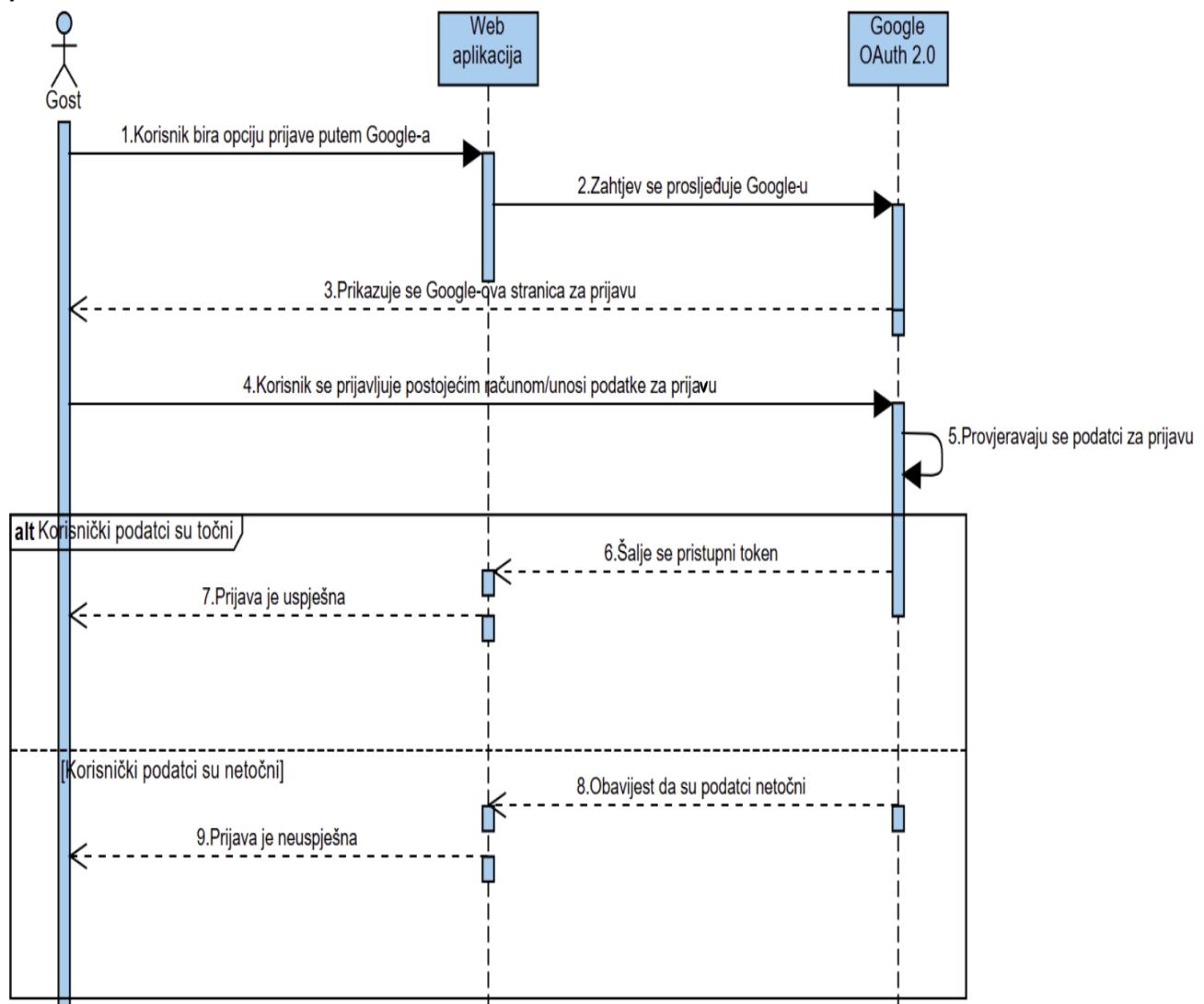
**Opis osnovnog tijeka:**

1. Upravitelj otvara modul "Korisnički računi"
2. Pregledava postojeće korisnike
3. Dodaje novog, mijenja podatke ili briše račun
4. Sustav pohranjuje promjene u bazu podataka

**Opis mogućih odstupanja:** Ako pokušava obrisati račun koji je aktivan, sustav prikazuje upozorenje

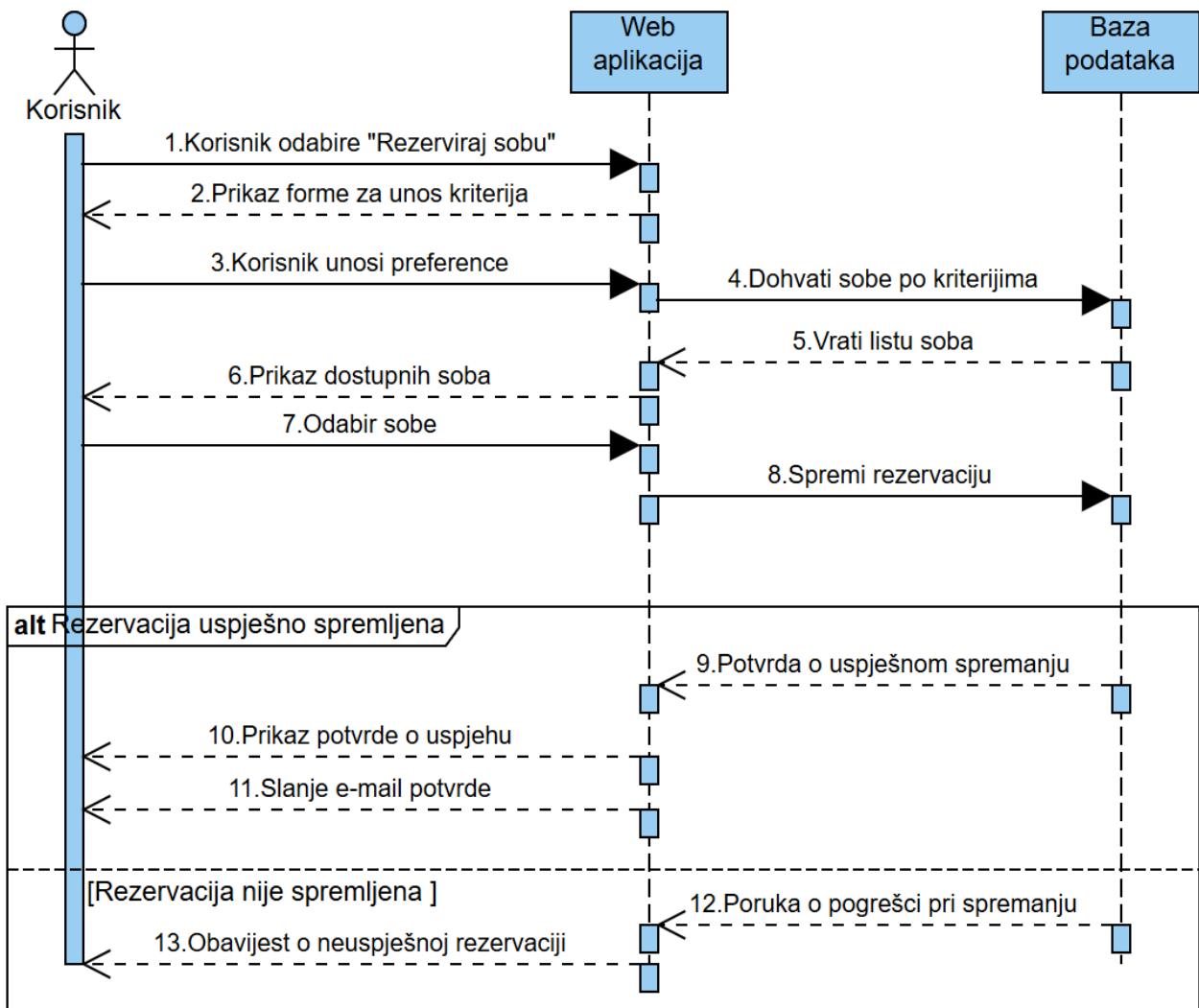
# Sekvencijski dijagrami

UC3: Prijava u sustav



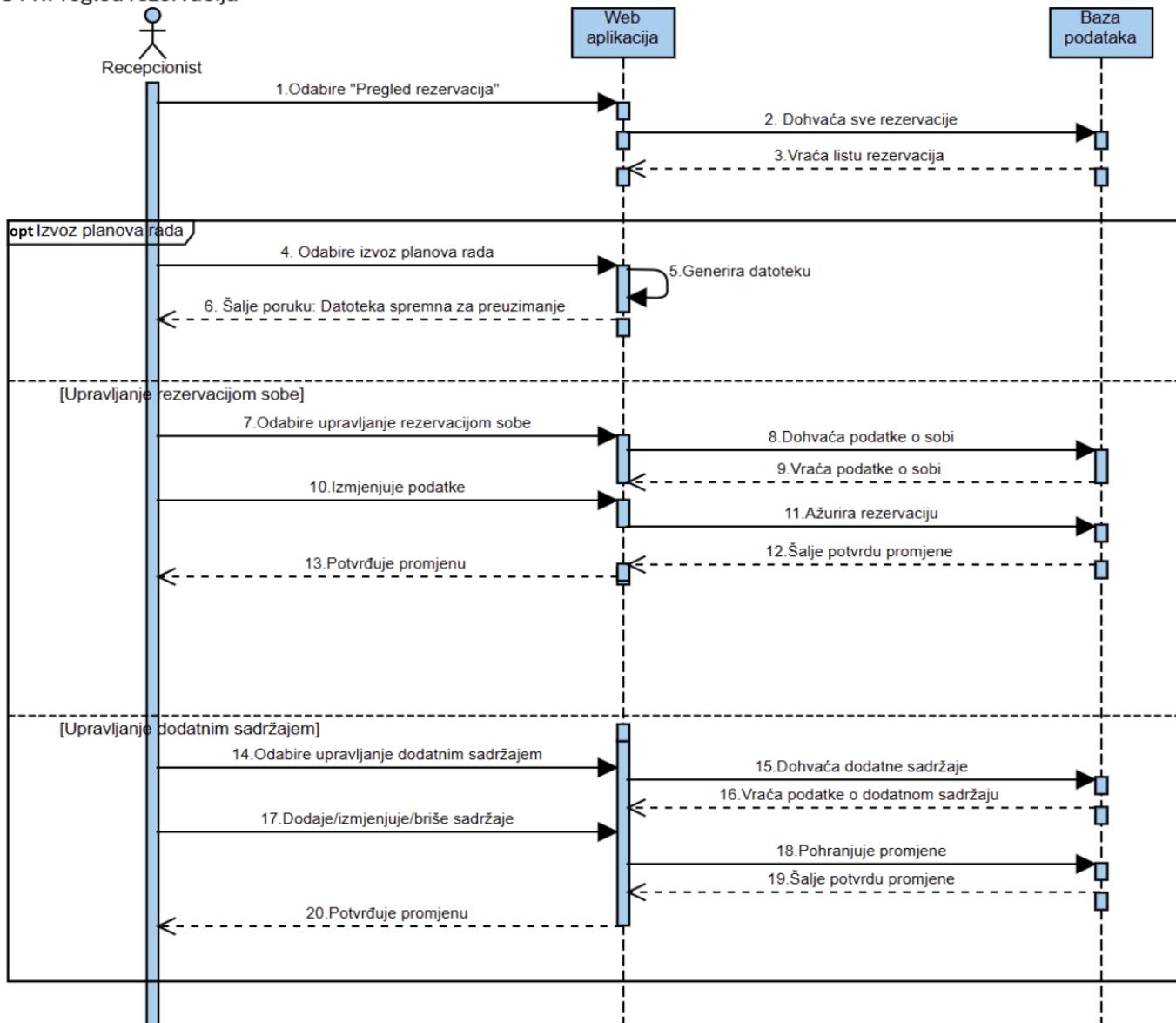
Slika 3.3. Prijava u sustav

## UC6:Odabir i rezervacija sobe

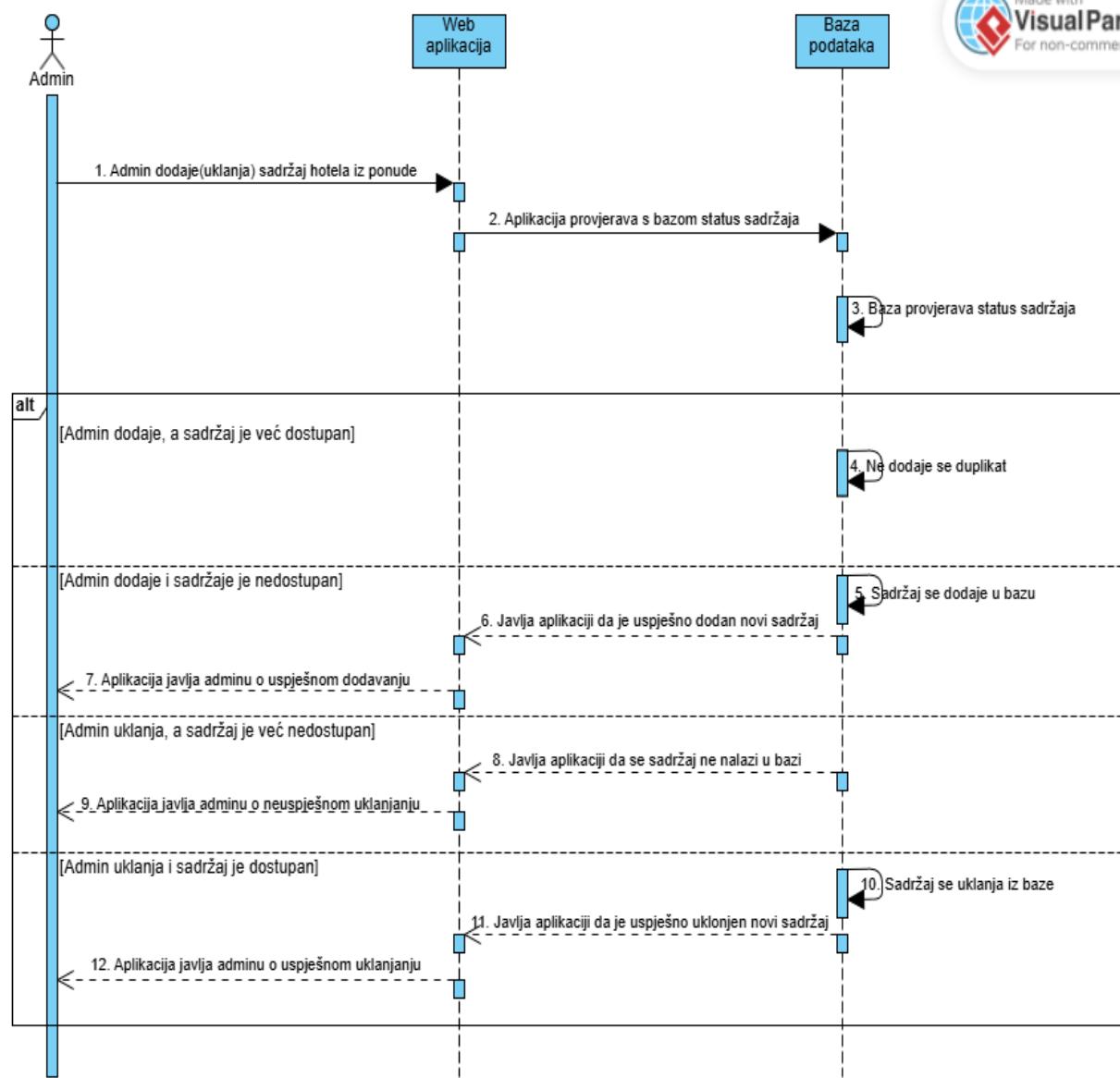


Slika 3.4. Odabir i rezervacija sobe

### UC11:Pregled rezervacija



Slika 3.5. Pregled rezervacija



Slika 3.6. Upravljanje dostupnosti soba i dodatnog sadržaja

## Provjera uključenosti ključnih funkcionalnosti u obrasce uporabe

Redni broj	Naziv obrasca uporabe	Funkcionalni zahtjevi
UC1	Pregled izgleda hotela	F-005
UC2	Pregled lokacije hotela	F-005
UC3	Prijava u sustav	F-001
UC4	Pregled podataka	F-013
UC5	Uređivanje informacija profila	F-014
UC6	Odabir i rezervacija soba	F-002

Redni broj	Naziv obrasca uporabe	Funkcionalni zahtjevi
UC7	Odabir i rezervacija dodatnog sadržaja	F-003
UC8	Ocenjivanje hotelske usluge	F-012
UC9	Plaćanje rezervacije	F-004
UC10	Kontaktiranje korisničke podrške	F-006
UC11	Pregled rezervacija	F-007
UC12	Izvoz planova rada	F-007
UC13	Upravljanje rezervacijama soba	F-007
UC14	Upravljanje rezervacijama dodatnog sadržaja	F-010
UC15	Pregled statistike	F-008
UC15.1	Izvoz statistike	F-009
UC16	Upravljanje dostupnošću soba i dodatnog sadržaja	F-010
UC17	Upravljanje korisničkim računima	F-011

## Arhitektura sustava

### Opis arhitekture

Arhitektura sustava koristi klijent-poslužitelj model organiziran u tri glavna sloja:

- Web preglednik (klijent):** Web preglednik služi kao korisničko sučelje za interakciju s aplikacijom. Putem web preglednika korisnik pokreće zahtjeve i šalje podatke poslužitelju te prima i obrađuje odgovore za prikaz traženih informacija. Web preglednik također obrađuje HTML, CSS i JavaScript sadržaj te osigurava ispravno prikazivanje elemenata korisničkog sučelja.
- Web poslužitelj (backend):** Web poslužitelj djeluje posrednik između klijenta i baze podataka te je odgovoran za obradu dolaznih zahtjeva i slanje odgovora natrag klijentu. Komunikacija se odvija putem RESTful API-ja koristeći HTTP protokol. Ovisno o vrsti zahtjeva web poslužitelj ili dohvaca podatke iz baze podataka ili pohranjuje podatke koje je poslao klijent.
- Baza podataka:** Baza podataka je glavni sloj za pohranu svih podataka aplikacije. U njoj se čuvaju trajni podaci poput korisničkih informacija te sadržaja i postavki aplikacije. Skoro svi zahtjevi kojima upravlja web poslužitelj uključuju komunikaciju s bazom podataka za čitanje ili pohranjivanje informacija.

### Obrazloženje odabira arhitekture

Stil klijent-poslužitelj s RESTful API-jem odabran je zbog sljedećih prednosti:

- Razdvajanje problema:** Klijentska i poslužiteljska strana razvijaju se i održavaju razdvojeno i neovisno jedna o drugoj, što omogućuje učinkovito upravljanje resursima.
- Učinkovito rukovanje podacima:** Poslužiteljska strana je pružatelj podataka, dok klijentska strana brine o API-jevima

### Organizacija sustava na visokoj razini

Sustav je organiziran u tri funkcionalno odvojena dijela koji međusobno komuniciraju putem API-ja:

- Klijent (React aplikacija):** šalje zahtjeve prema poslužitelju
- Backend (Spring Boot aplikacija):** obrađuje zahtjeve s klijenta i komunicira s bazom
- Baza podataka:** pohranjuje i vraća podatke koje koristi poslužitelj

## Organizacija aplikacije

- Poslužiteljski dio:** Ostvaren programskim jezikom Java pomoću alata Spring-Boot radnog okvira. Razvija se u razvojnom okruženju IntelliJ IDEA, koje podržava Spring-Boot alate.
- Klijentski dio:** Ostvaren programskim jezikom JavaScript pomoću React radnog okvira. Razvija se u razvojnom okruženju WebStorm.
- Dizajn korisničkog sučelja:** Ostvaren pomoću alata Figma.

Sva razvojna okruženja razvijena su od strane JETBrains-a radi konzistentnosti.

## Baza podataka

### Opis tablica

#### Drzava

Predstavlja zemlju porijekla mjesta. Atributi: drzava\_id (PK), nazivDrzave (U).

Atribut	Tip podatka	Opis varijable
drzava_id	SERIAL	Jedinstveni identifikator svake države
nazivDrzave	VARCHAR(50)	Jedinstveni naziv države

#### Mjesto

Predstavlja mjesto iz kojeg je korisnik aplikacije. Tablica je povezana: Many-to-One s Drzava preko drzava\_id. Atributi: postBr (PK), nazMjesto, drzava\_id (FK).

Atribut	Tip podatka	Opis varijable
id	SERIAL	jedinstveni identifikator svakog mjeseta
postBr	VARCHAR(20)	Jedinstveni poštanski broj mjeseta
nazMjesto	VARCHAR(50)	Naziv mjeseta
drzava_id	INTEGER	Identifikator države

#### Korisnik

Predstavlja korisnika aplikacije, uključujući goste i zaposlenike. Sadrži osobne podatke poput imena, email-a, telefonskog broja te njihove uloge u aplikaciji. Tablica je povezana: Many-to-One s Mjesto preko postBr te One-to-Many s Upit, Rezervacija i

Recenzija preko korisnik\_id. Atributi: korisnik\_id (PK), imeKorisnik, prezimeKorisnik, emailKorisnik (U), telefonKorisnik (U), ovlastKorisnik, postBR (FK).

Atribut	Tip podatka	Opis varijable
korisnik_id	SERIAL	Jedinstveni identifikator svakog korisnika
imeKorisnik	VARCHAR(50)	Ime korisnika
prezimeKorisnik	VARCHAR(50)	Prezime korisnika
emailKorisnik	VARCHAR(50)	Jedinstveni email korisnika
telefonKorisnik	VARCHAR(20)	Jedinstveni telefonski broj korisnika
ovlastKorisnik	VARCHAR(20)	Ovlast korisnika u aplikaciji
mjesto_id	INTEGER	Identifikator mesta odakle je korisnik

## Soba

Predstavlja sobu hotela. Sadrži podatke poput vrste sobe, cijene i dostupnosti. Tablica je povezana: One-to-Many s RezervirajSobu preko broj\_sobe. Atributi: broj\_sobe (PK), vrsta, cijena, status.

Atribut	Tip podatka	Opis varijable
soba_id	SERIAL	Jedinstveni identifikator svake sobe
broj_sobe	VARCHAR(5)	Jedinstveni broj svake sobe
vrsta	VARCHAR(20)	Vrsta sobe
cijena	INTEGER	Cijena sobe
status	VARCHAR(20)	Status sobe, je li soba slobodna ili zauzeta
kapacitet	INTEGER	Ne null element koji označava kapacitet sobe
balkon	BOOLEAN	opisuje sadrzi li soba balkon
pogled_na_more	BOOLEAN	opisuje sadri li soba pogled_na_more

## Rezervacija

Predstavlja rezervaciju. Tablica je povezana: Many-to-One s Korisnik preko korisnik\_id te One-to-Many s Plaćanje, RezervirajSobu i RezervirajSadrzaj preko rezervacija\_id. Atributi: rezervacija\_id (PK), datumRezerviranja, placeno?, korisnik\_id (FK).

Atribut	Tip podatka	Opis varijable
rezervacija_id	SERIAL	Jedinstveni identifikator svake rezervacije
datumRezerviranja	DATE	Datum rezervacije
placeno	BOOLEAN	Je li rezervacija plaćena ili nije
iznos_rezervacije	NUMERIC(10,2)	cijena rezervacije
korisnik_id	INTEGER	Identifikator korisnika koji je napravio rezervaciju

## DodatniSadrzaj

Predstavlja dodatni sadržaj hotela koji se može rezervirati. Sadrži podatke poput: vrsta, cijena, dostupnost i kapacitet dodatnog sadržaja. Tablica je povezana: One-to-Many s RezervirajSadrzaj preko dodatniSadrzaj\_id. Atributi: dodatniSadrzaj\_id(PK), vrstaDodatniSadrzaj, statusDodatniSadrzaj, cijenaDOdatniSadrzaj, kapacitetDodatniSadrzaj.

Atribut	Tip podatka	Opis varijable
dodatniSadrzaj_id	SERIAL	Jedinstveni identifikator svakog dodatnog sadržaja
vrstaDodatniSadrzaj	VARCHAR(20)	Vrsta dodatnog sadržaja
cijena_sadrzaj	NUMERIC(10,2)	Cijena sobe
statusDodatniSadrzaj	VARCHAR(20)	Status dodatnog sadržaja, je li dodatan sadržaj dostupan ili nije

## Recenzija

Predstavlja recenziju koju gost holela može ostaviti. Tablica je povezana: Many-to-One s Korisnik preko korisnik\_id. Atributi: recenzija\_id(PK), ocjenaRecenzija, komentarRecenzija, datumRecenzija, korisnik\_id(FK).

Atribut	Tip podatka	Opis varijable
recenzija_id	SERIAL	Jedinstveni identifikator svake recenzije
ocjenaRecenzija	INTEGER	Ocjena dana u recenziji
komentarRecenzija	TEXT	Komentar dan u recenziji
datumRecenzija	DATE	Datum recenzije
korisnik_id	INTEGER	Identifikator korisnika koji je ostavio recenziju

## RezervirajSadrzaj

Predstavlja rezervaciju dodatnog sadržaja hotela. Tablica je povezana: Many-to-One s Rezervacija preko rezervacija\_id i One-to-Many s DodatniSadrzaj preko dodatniSadrzaj\_id. Atributi: rezervacija\_id(FK), dodatniSadrzaj\_id(FK), datumOdSadrzaj, datumDoSadrzaj. Ključ tablice je par (rezervacija\_id, dodatniSadrzaj\_id).

Atribut	Tip podatka	Opis varijable
rezervacija_id	INTEGER	Identifikator rezervacije
dodatniSadrzaj_id	INTEGER	Identifikator dodatnog sadržaja koji će se rezervirati
datum_sadrzaj	DATE	Datum rezervacije dodatnog sadržaja

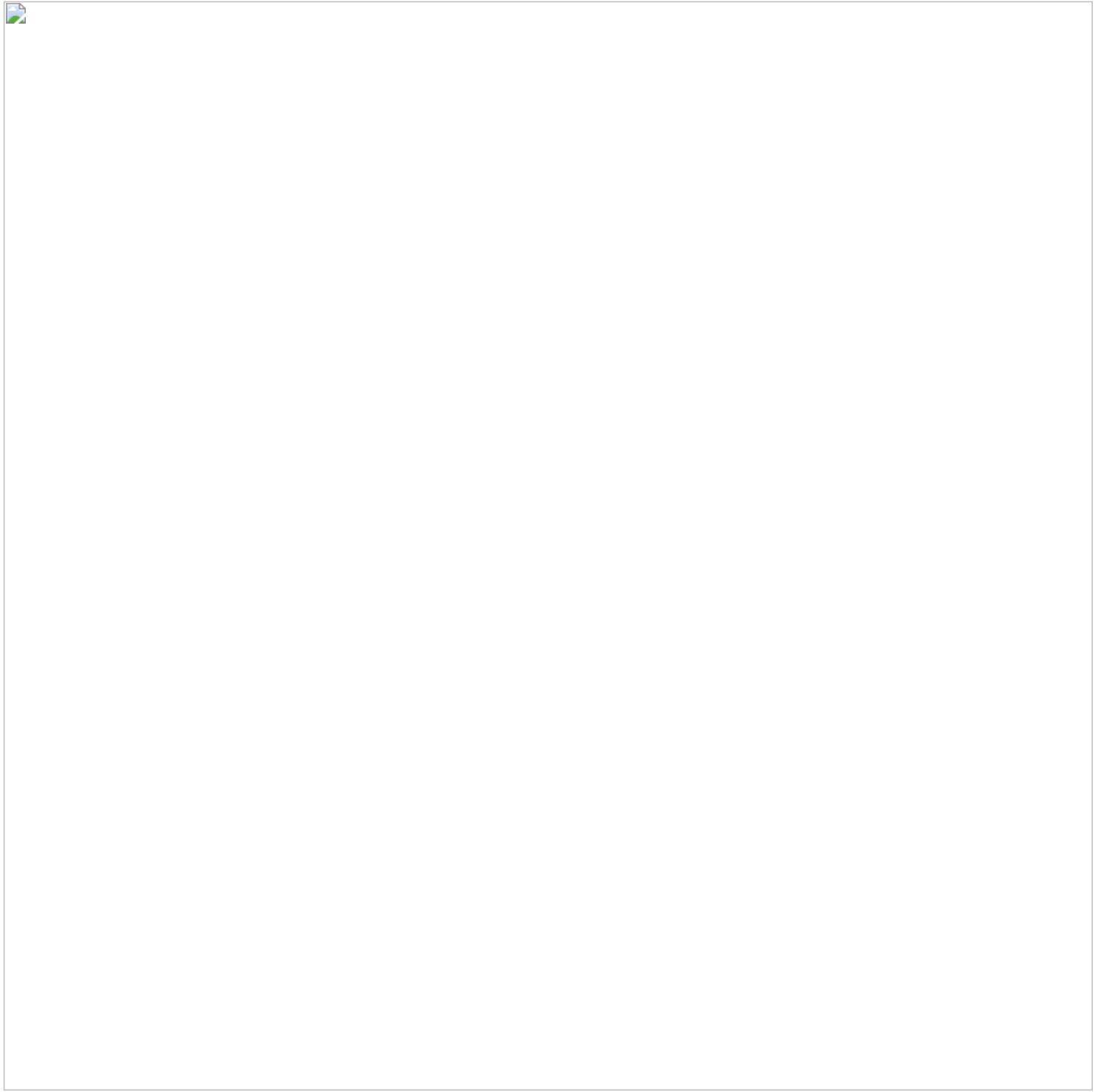
## RezervirajSobu

Predstavlja rezervaciju sobe hotela. Tablica je povezana: Many-to-One s Rezervacija preko rezervacija\_id i One-to-Many s Soba preko broj\_sobe. Atributi: rezervacija\_id(FK), broj\_sobe(FK), datumOdSoba, datumDoSoba. Ključ tablice je par (rezervacija\_id, broj\_sobe).

Atribut	Tip podatka	Opis varijable
rezervacija_id	INTEGER	Identifikator rezervacije
soba_id	INTEGER	Identifikator sobe
datumOdSoba	DATE	Datum početka rezervacije sobe

Atribut	Tip podatka	Opis varijable
datumDoSoba	DATE	Datum završetka rezervacije sobe

## Dijagram baze podataka



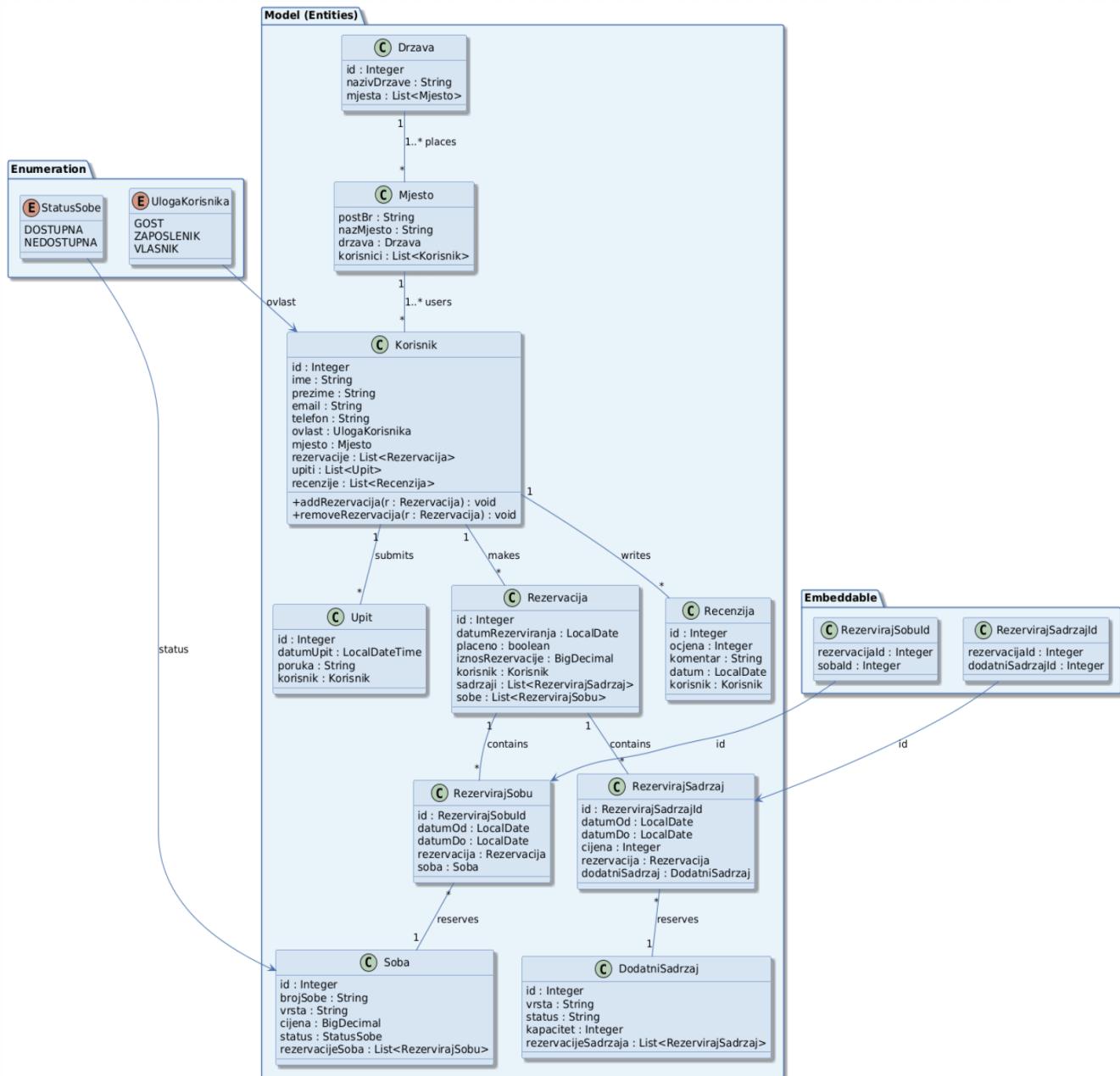
## Dijagram razreda

Na slikama 4.1 – 4.4 prikazani su dijagrami razreda razdvojeni po paketima unutar backend (Spring Java) i frontend (React) dijela aplikacije. Radi preglednosti, dijagrami su razlomljeni po slojevima, a zatim je prikazan integrirani dijagram koji povezuje frontend i backend slojeve.

## Model / Entities (Backend)

Slika 4.1 prikazuje entitete koji predstavljaju tablice baze podataka. Klase unutar **model paketa** sadrže atribute koji se mapiraju na stupce baze i prikazuju relacije između entiteta.

- Klase sadrže osnovne metode za dodavanje i uklanjanje povezanih objekata(npr. `Korisnik.addRezervacija()` ).
- Embeddable klase( `RezervirajSobuId` , `RezervirajSadrzajId` )koriste se za složene primarne ključeve.
- Enumeracije( `UlogaKorisnika` , `StatusSobe` )definiraju fiksni skup vrijednosti za atribute korisnika i soba.
- Relacije između entiteta prikazane su strelicama s opisom(npr. `makes` , `reserves` ).



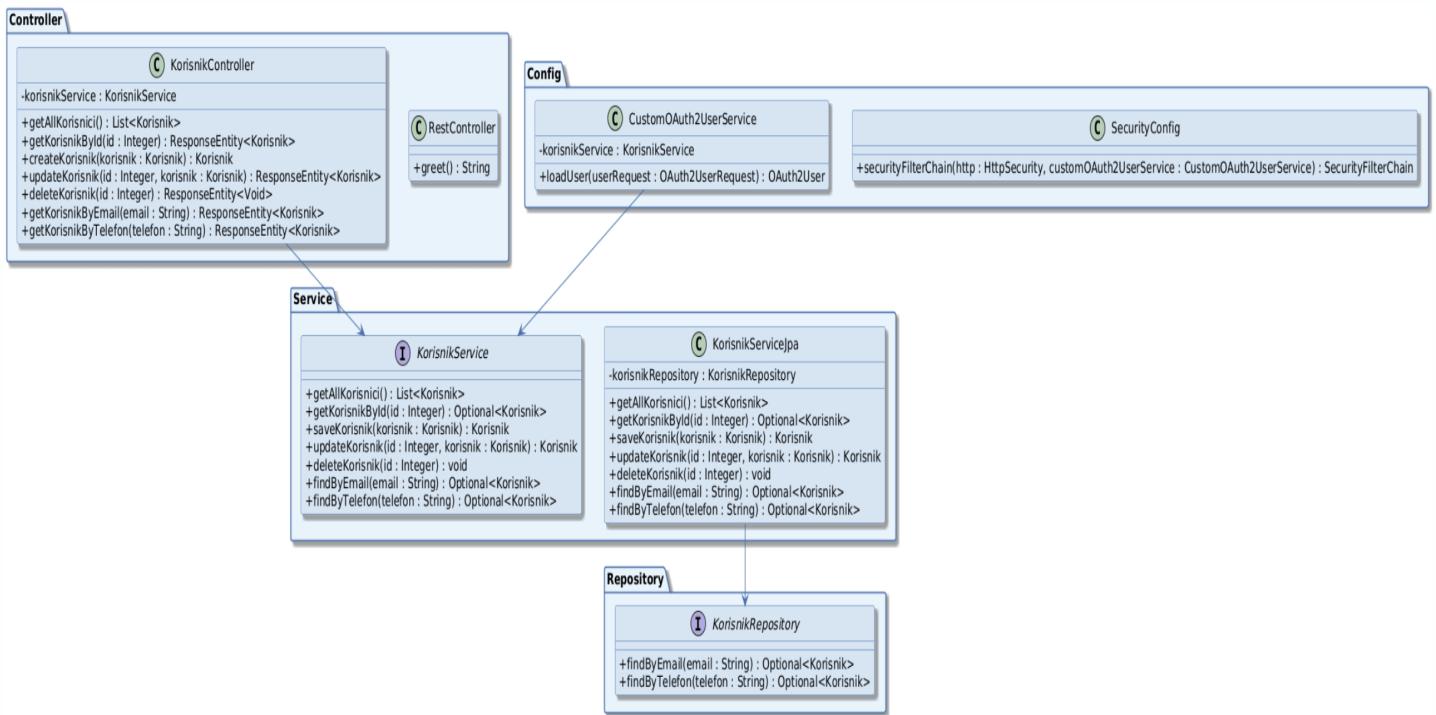
Slika 4.1 – Dijagram entiteta (Model / Entities)

## Backend slojevi (Repository / Service / Controller / Config)

Slika 4.2 prikazuje klase unutar **repository**, **service** i **controller** paketa.

- **Repository paketi** sadrže sučelja( `KorisnikRepository` ) koja olakšavaju CRUD operacije nad bazom.
- **Service paketi** implementiraju poslovnu logiku, npr. `KorisnikService` i `KorisnikServiceJpa` upravljaju pozivima repozitorija i mapiranjem podataka.

- **Controller paketi** (`KorisnikController`) definiraju REST krajne točke i obrađuju HTTP zahteve.
- **Config paketi** sadrže sigurnosne konfiguracije i servise za OAuth2 servise  
(`CustomOAuth2UserService`, `SecurityConfig`).

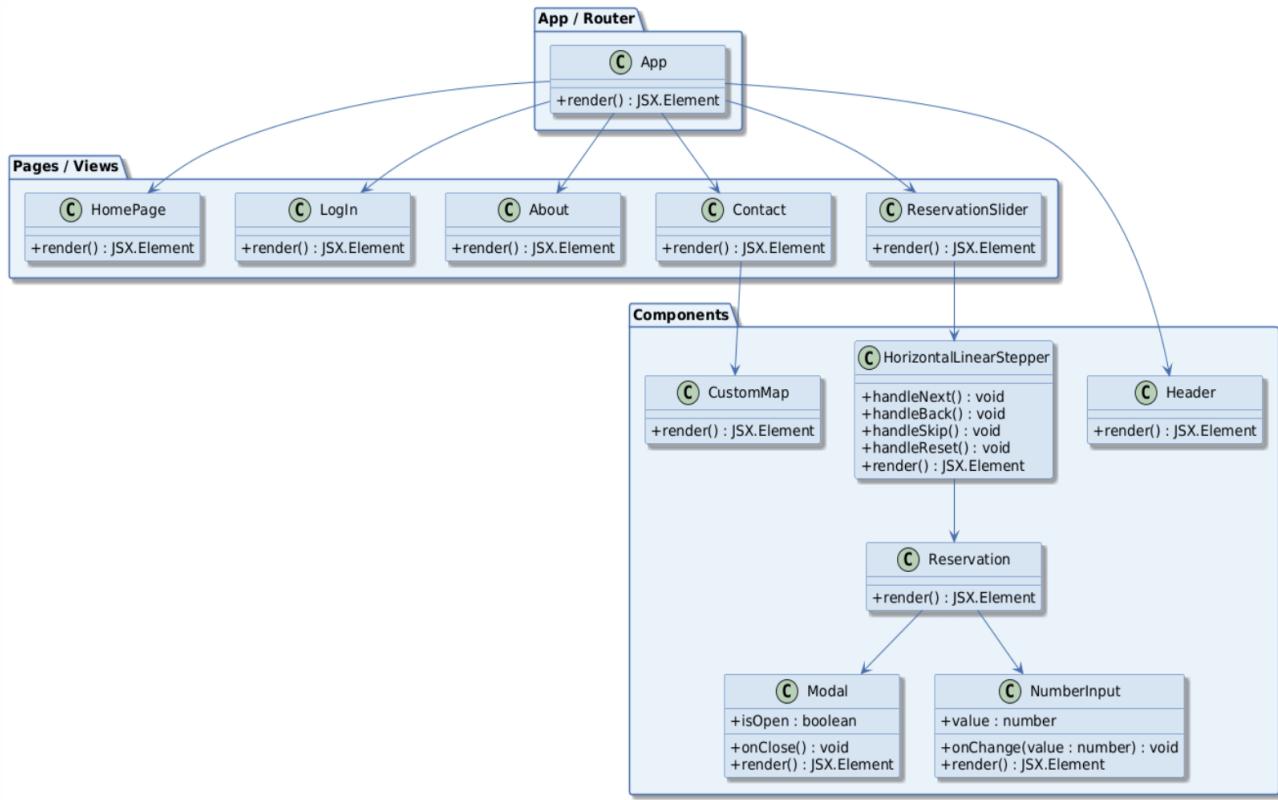


Slika 4.2 – Backend slojevi (Repository / Service / Controller / Config)

## Frontend (React)

Slika 4.3 prikazuje glavne stranice i komponente React aplikacije.

- **Pages / Views:** `HomePage`, `LogIn`, `About`, `Contact`, `ReservationSlider` – stranice koje prikazuju sadržaj korisniku.
- **Components:** `Header`, `CustomMap`, `Reservation`, `HorizontalLinearStepper`, `Modal`, `NumberInput` – komponente koje se višekratno koriste unutar stranica.
- **App / Router:** `App` komponenta koja koristi React Router za navigaciju između stranica.
- Strelice prikazuju hijerarhiju komponenti i tok navigacije korisnika.

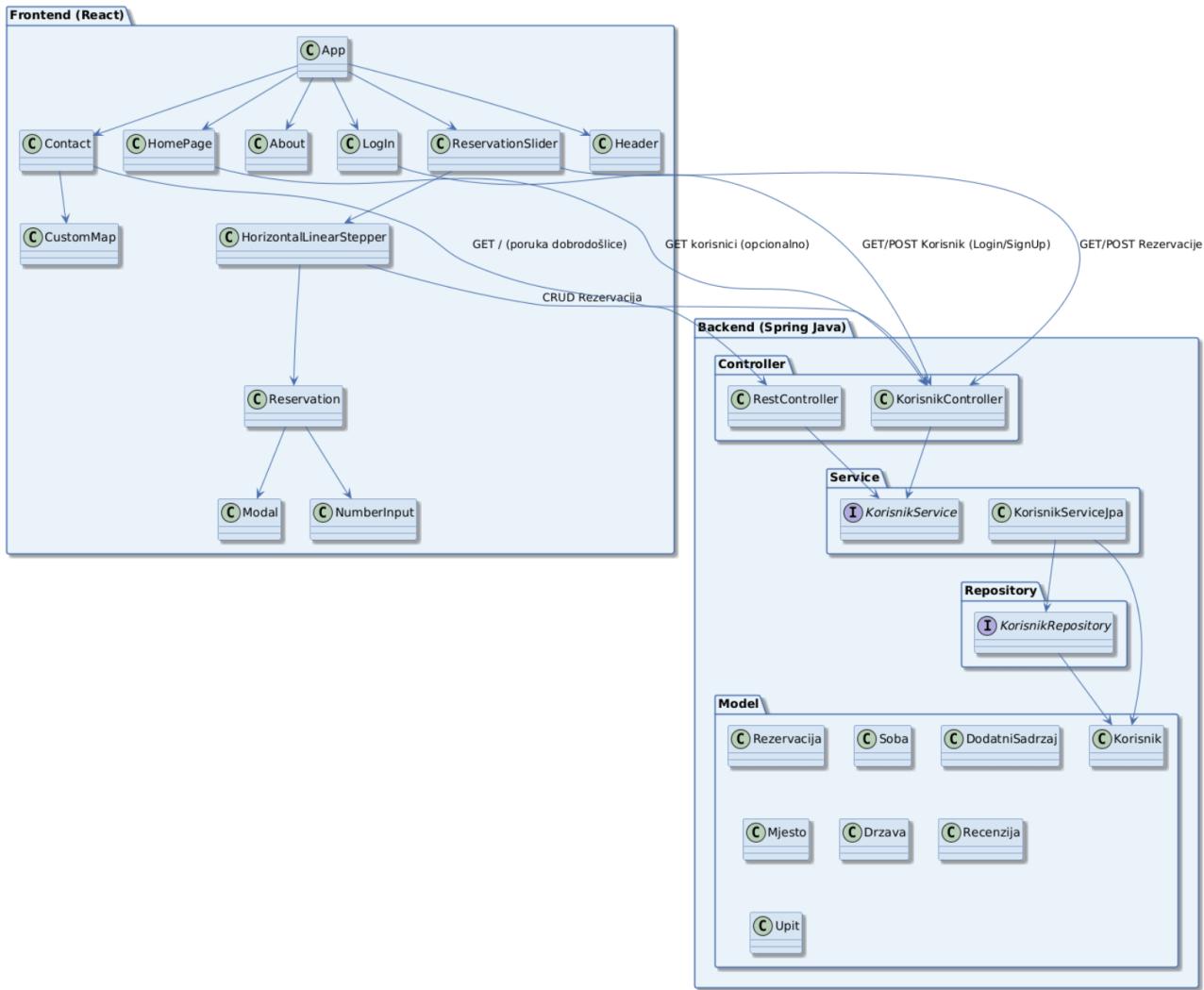


Slika 4.3 – Frontend React komponente

## Integrirani dijagram slojeva (Frontend → Backend)

Slika 4.4 prikazuje **integrirani dijagram koji povezuje frontend i backend slojeve**.

- Frontend komponente i stranice prikazane su u gornjem dijelu, backend paketi u donjem.
- Strelice označavaju REST pozive i tok podataka:
  - `ReservationSlider` i `HorizontalLinearStepper` komuniciraju s `KorisnikController` za CRUD operacije rezervacija.
  - `LogIn` komponenta poziva backend za autentifikaciju korisnika.
  - `Contact` komponenta šalje zahtjeve prema `RestController` za prikaz poruke dobrodošlice.

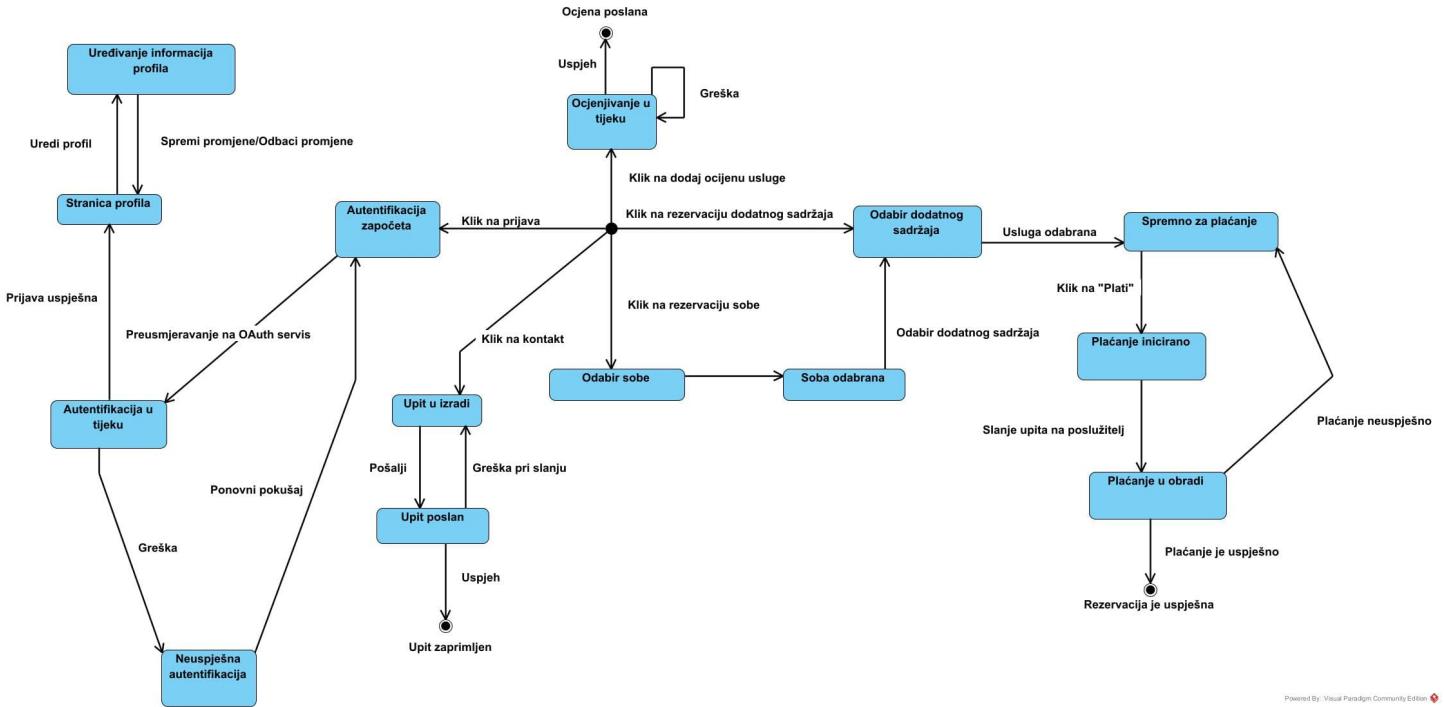


Slika 4.4 – Integrirani dijagram (Frontend → Backend)

## Dinamičko ponašanje aplikacije

### UML dijagrami stanja

Dijagram stanja prikazuje dinamičko ponašanje aplikacije iz perspektive korisnika, odnosno prijelaze između različitih stanja sustava tijekom tipičnih korisničkih aktivnosti kao što su autentifikacija, uređivanje profila, slanje upita, rezervacija sobe i dodatnog sadržaja, plaćanje te ostavljanje recenzije.



Slika 4.5 - Dijagram stanja

Proces započinje početnim stanjem u kojem korisnik može pokrenuti prijavu u sustav. Tijekom autentifikacije sustav može prijeći u stanje uspješne ili neuspješne autentifikacije, pri čemu se u slučaju pogreške korisniku omogućuje ponovni pokušaj prijave. Nakon uspješne prijave korisnik dolazi na stranicu profila, gdje može pregledavati i uređivati svoje podatke.

Iz glavnog stanja aplikacije korisnik može odabratи više funkcionalnosti:

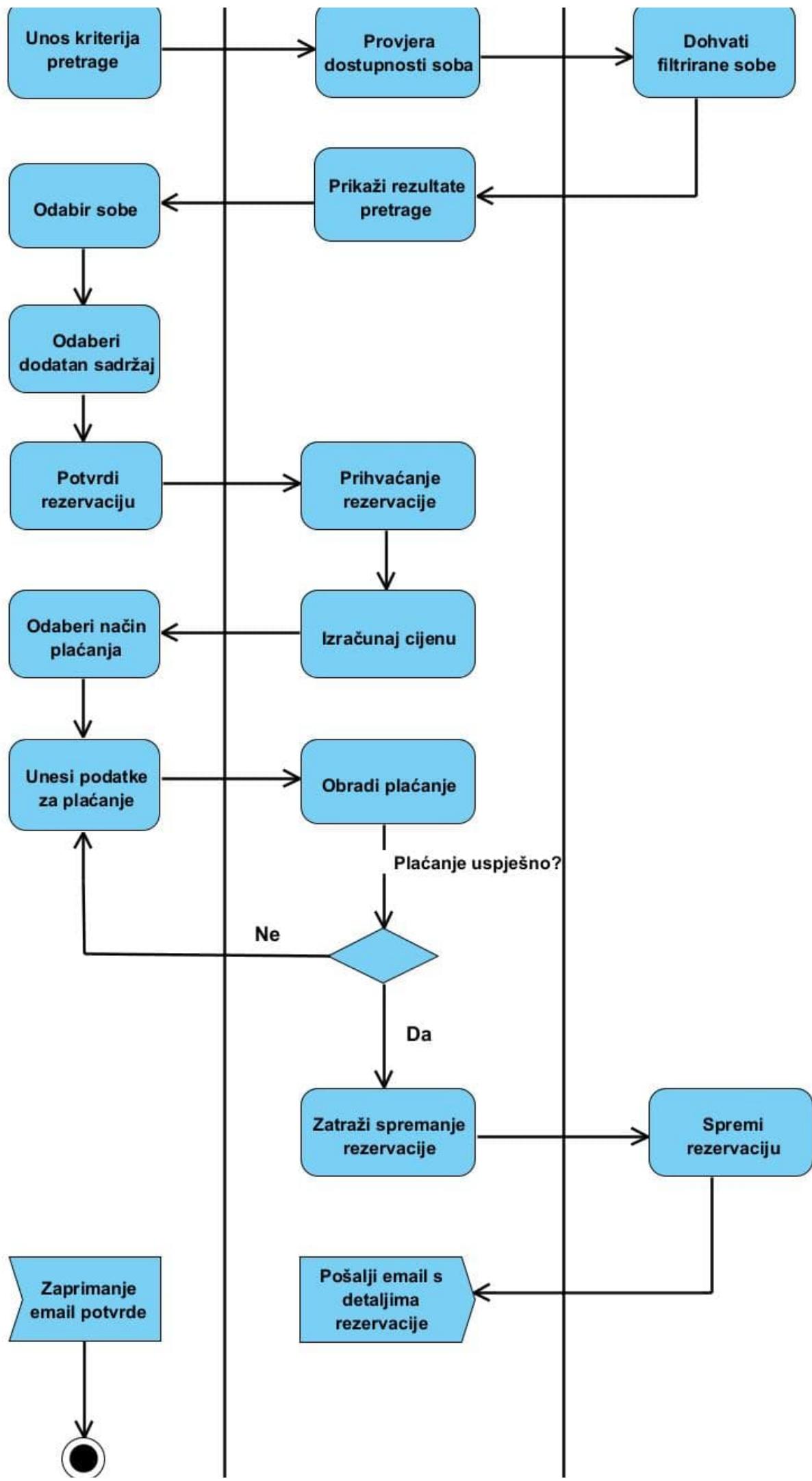
- Slanje upita putem kontakt forme, gdje sustav obrađuje upit i vraća informaciju o uspješnom ili neuspješnom slanju.
- Rezervaciju sobe, koja uključuje odabir sobe i prelazak u stanje odabrane sobe.
- Rezervaciju dodatnog sadržaja, gdje korisnik bira dodatne usluge koje se vežu uz rezervaciju.
- Plaćanje rezervacije, koje prolazi kroz stanja iniciranja i obrade plaćanja te završava uspješnom ili neuspješnom transakcijom.
- Ocjenjivanje usluge, gdje korisnik ostavlja recenziju, a sustav potvrđuje uspješno zaprimanje ocjene ili javlja pogrešku.

Ovim dijagramom obuhvaćeni su ključni poslovni procesi aplikacije te je vizualno prikazan način na koji sustav reagira na korisničke akcije i upravlja prijelazima između stanja tijekom rada aplikacije.

## UML dijagrami aktivnosti

Dijagram aktivnosti prikazuje proces rezervacije hotelske sobe s odabirom dodatnog sadržaja i provedbom plaćanja. Aktori uključeni u proces su korisnik, web aplikacija i baza podataka.



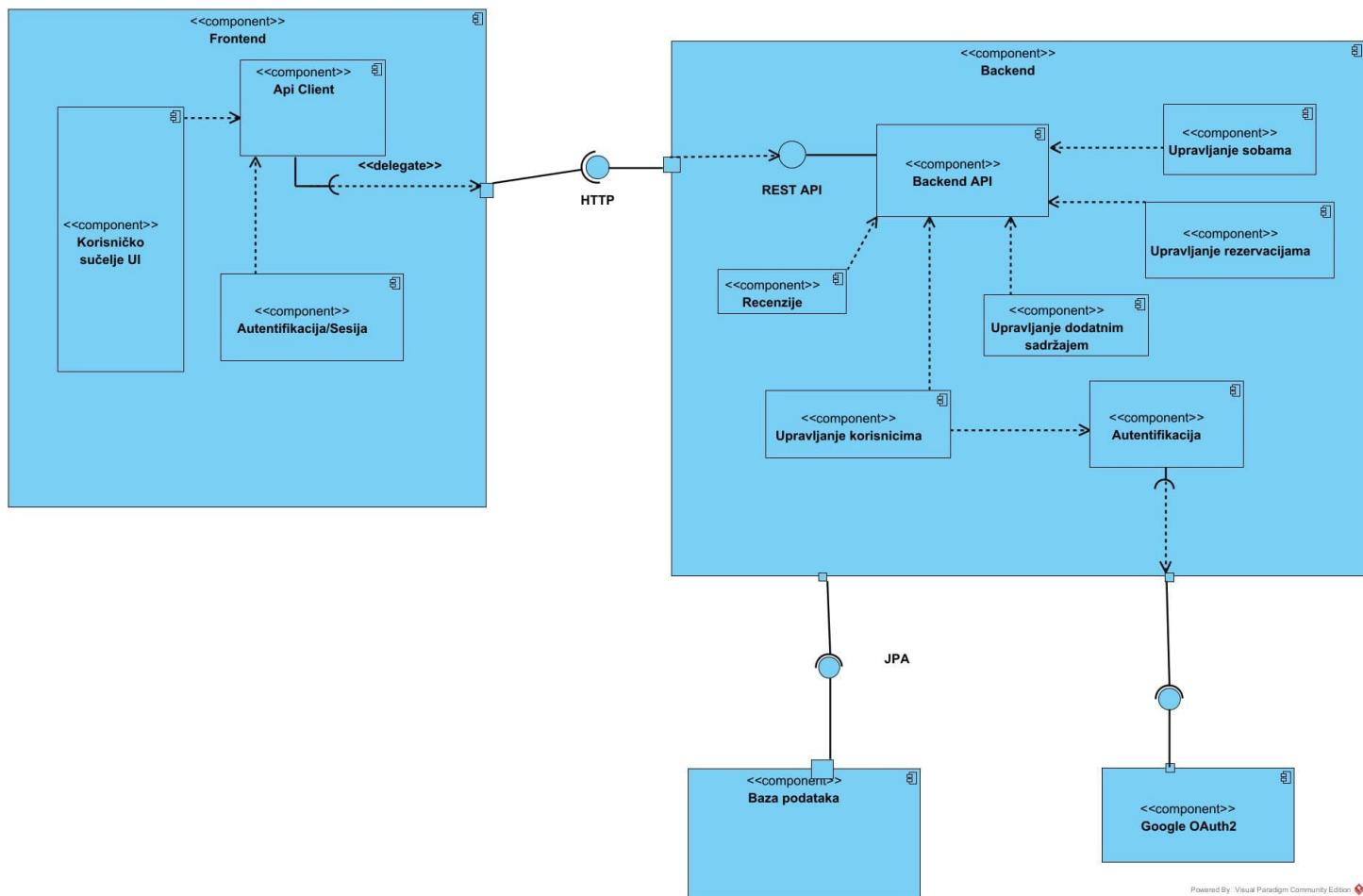


Slika 4.6 - Dijagram aktivnosti

Interakcija između sudionika odvija se kroz sljedeće korake:

1. Korisnik pokreće pretragu soba te u web aplikaciji unosi kriterije pretrage (datum boravka, broj gostiju i broj soba).
2. Web aplikacija provjerava dostupnost soba te dohvata filtrirane podatke iz baze podataka, nakon čega korisniku prikazuje rezultate pretrage.
3. Korisnik odabire željenu sobu te po potrebi dodaje dodatni sadržaj (bazen, restoran, teretana).
4. Nakon potvrde rezervacije, web aplikacija izračunava ukupnu cijenu na temelju odabrane sobe i dodatnog sadržaja.
5. Korisnik odabire način plaćanja i unosi potrebne podatke za plaćanje, dok web aplikacija obrađuje plaćanje.
6. U slučaju neuspješnog plaćanja, korisniku se omogućuje ponovni unos podataka za plaćanje.
7. U slučaju uspješnog plaćanja, web aplikacija zahtijeva spremanje rezervacije, a baza podataka pohranjuje podatke o rezervaciji.
8. Nakon uspješnog spremanja rezervacije, web aplikacija korisniku šalje e-mail s detaljima rezervacije, koji korisnik zaprima kao potvrdu uspješne rezervacije.

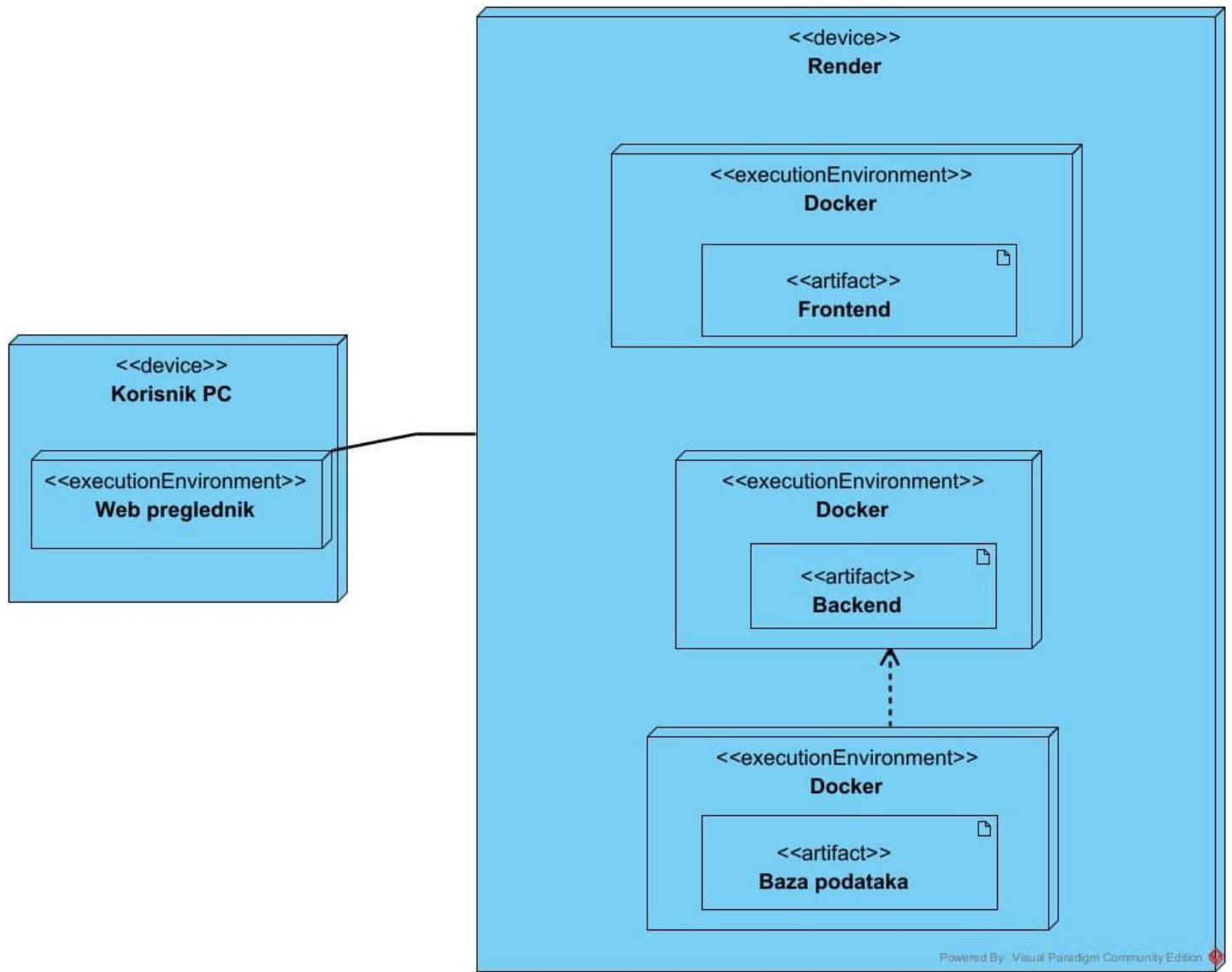
## Dijagram komponenata



Slika 5.1

Dijagram prikazuje arhitekturu sustava podijeljenu na frontend i backend dio. Frontend se sastoji od korisničkog sučelja, API klijenta te komponente za autentifikaciju i upravljanje sesijom, koje zajedno omogućuju komunikaciju s backendom putem REST sučelja. Backend dio obuhvaća središnji Backend API koji koordinira rad funkcionalnih komponenti za upravljanje korisnicima, sobama, rezervacijama, dodatnim sadržajem i recenzijama, od kojih svaka ima svoj kontroler, servisni dio i repozitorni dio. Autentifikacija korisnika realizirana je putem interne autentifikacijske komponente uz integraciju vanjskog servisa Google OAuth2. Backend komunicira s bazom podataka radi pohrane i dohvatanja podataka.

## Dijagram razmještaja



Slika 5.2

Dijagram prikazuje način implementacije i izvođenja sustava. Korisnik pristupa aplikaciji putem web preglednika koji se izvršava na korisničkom računalu. Frontend i backend aplikacije implementirane su kao zasebni artefakti koji se izvršavaju unutar Docker okruženja na poslužitelju Render. Backend aplikacija komunicira s bazom podataka, koja je također pokrenuta unutar zasebnog Docker okruženja. Ovakva arhitektura omogućuje izolaciju komponenti, lakše upravljanje ovisnostima te jednostavnije skaliranje i održavanje sustava.

## Ispitivanje komponenti

# KorisnikServiceTests

## 1. Test: Spremanje korisnika s postojećim mjestom i automatskom dodjelom administratorske uloge

**1. Funkcionalnost koju testiramo:** Ovim testom ispituje se funkcionalnost metode spremiKorisnika u servisnoj klasi KorisnikService. Test provjerava ispravno spremanje novog korisnika u sustav kada korisnik ima administratorski e-mail i kada mjesto prebivališta već postoji u bazi podataka. Cilj je osigurati da se poslovna logika pravilno izvršava: korisniku se dodjeljuje odgovarajuća uloga (VLASNIK), koristi se postojeće mjesto iz baze (bez duplicitanja), te se naziv mjesta spremi u standardiziranom formatu (lowercase), dok poštanski broj ostaje nepromijenjen. Test također implicitno provjerava integritet veza između korisnika i mjesta.

### 2. Ispitni slučaj

#### Ulazni podaci:

- Email korisnika : [admin@hotel.com](mailto:admin@hotel.com)
- adminEmail: [admin@hotel.com](mailto:admin@hotel.com)
- Naziv mjesta: Cres
- Poštanski broj: 51557
- Država: Hrvatska

#### Očekivani rezultati:

- Korisniku se dodjeljuje uloga VLASNIK
- Naziv mjesta se spremi kao lowercase("cres")
- Poštanski broj ostaje nepromijenjen("51557")
- Koristi se postojeće mjesto iz baze podataka

#### Dobiveni rezultati:

- Dodijeljena uloga korisnika: VLASNIK
- Naziv mjesta: "cres"
- Poštanski broj: "51557"

#### 3. Postupak ispitivanja:

Prije pokretanja ispitivanja mockiraju se repozitoriji KorisnikRepository, MjestoRepository i DrzavaRepository kako bi se simuliralo ponašanje baze podataka bez korištenja stvarne baze. Zatim se ručno postavlja konfiguracijska vrijednost adminEmail potrebna za pravilno izvođenje poslovne logike u servisnoj klasi. Nakon toga pripremaju se testni objekti Korisnik, Mjesto i Drzava s unaprijed definiranim vrijednostima koje predstavljaju ulazne podatke ispitnog slučaja. Simulira se stanje baze podataka u kojem mjesto već postoji, tako da repozitorij za mjesta vraća postojeći zapis. Potom se poziva metoda spremiKorisnika nad servisom KorisnikService. Na kraju se dobiveni rezultati provjeravaju pomoću JUnit assercija te se na temelju usporedbe očekivanih i dobivenih rezultata utvrđuje ishod ispitivanja.

```

@Test & LeniBosnic
void spremiKorisnika_MjestoExistsAndAdminEmail(){
    ReflectionTestUtils.setField(korisnikService, "name", "adminEmail", "admin@hotel.com"); // treba da bi se test mogao izraditi

    Drzava drzava = new Drzava();
    drzava.setNazivDrzave("Hrvatska");

    Mjesto mjesto = new Mjesto();
    mjesto.setNazMjesto("Cres"); // trebalo bi se staviti u lowercase to usput provjeravamo dolje
    mjesto.setPostBr("51557");
    mjesto.setDrzava(drzava);

    // simuliramo da je cres vec u bazi podataka, dakle "punimo" bazu podataka prije testiranja mjestom cres
    Mjesto bazaMjesto = new Mjesto();
    bazaMjesto.setNazMjesto("cres");
    bazaMjesto.setPostBr("51557");

    Korisnik korisnik = new Korisnik();
    korisnik.setEmail("admin@hotel.com"); // isti kao adminEmail, provjeravamo hoce li se uloga korisnika automatski postaviti na VLASNIK
    korisnik.setMjesto(mjesto);

    Mockito.when(drzavaRepository.findByNazivDrzave("Hrvatska")).thenReturn(drzava);
    Mockito.when(mjestoRepository.findByPostBrAndNazMjesto(postBr: "51557", nazMjesto: "cres")).thenReturn(Optional.of(bazaMjesto));
    Mockito.when(korisnikRepository.save(any(Korisnik.class))).thenAnswer(invocationOnMock -> invocationOnMock.getArguments()[0]); // prvo dodano u bazu

    Korisnik rez= korisnikService.spremiKorisnika(korisnik);
    assertEquals(VLASKA, rez.getUloga());
    assertEquals("cres", rez.getMjesto().getNazMjesto());
    assertEquals("51557", rez.getMjesto().getPostBr());
}

```

Slika 6.1.1

## 2. Test: Spremanje korisnika s novim mjestom

**1. Funkcionalnost koju testiramo:** Ovim testom ispituje se funkcionalnost spremanja novog korisnika u sustav u slučaju kada korisnik nema administratorsku e-mail adresu te kada mjesto prebivališta ne postoji u bazi podataka. Cilj testa je provjeriti ispravnost poslovne logike unutar servisne klase KorisnikService, uključujući dodjelu odgovarajuće uloge korisniku i ispravno spremanje novog mjesta.

### 2. Ispitni slučaj

#### Ulazni podaci:

- E-mail korisnika: [test@gmail.com](mailto:test@gmail.com)
- Konfiguracijska vrijednost adminEmail: [admin@hotel.com](mailto:admin@hotel.com)
- Naziv mjesta: Cres
- Poštanski broj: 51557
- Država: Hrvatska

#### Očekivani rezultati:

- Korisniku se dodjeljuje uloga REGISTRIRAN
- Naziv mjesta spremi se u malim slovima ("cres")
- Poštanski broj ostaje nepromijenjen ("51557")
- Kreira se i spremi novo mjesto u bazu podataka

#### Dobiveni rezultati:

- Dodijeljena uloga korisnika: REGISTRIRAN
- Naziv mjesta: "cres"
- Poštanski broj: "51557"

**3. Postupak ispitivanja:** Prije pokretanja ispitivanja mockiraju se repozitoriji KorisnikRepository, MjestoRepository i DrzavaRepository kako bi se simuliralo ponašanje baze podataka. Zatim se ručno postavlja konfiguracijska vrijednost adminEmail, te se pripremaju testni objekti Korisnik, Mjesto i Drzava s unaprijed definiranim vrijednostima. Simulira se stanje baze podataka u kojem traženo mjesto ne postoji, pri čemu repozitorij za mjesta vraća prazan rezultat. Na kraju se dobiveni rezultati provjeravaju pomoću JUnit assercija te se na temelju usporedbe očekivanih i dobivenih rezultata utvrđuje uspješnost ispitivanja.

```
@Test & LeniBosnic
void spremiKorisnika_NewMjesto(){
    ReflectionTestUtils.setField(korisnikService, name: "adminEmail", value: "admin@hotel.com"); // treba da bi se test mogao odrediti

    Drzava drzava = new Drzava();
    drzava.setNazivDrzave("Hrvatska");

    Mjesto mjesto = new Mjesto();
    mjesto.setNazMjesta("Cres");
    mjesto.setPostBr("51557");
    mjesto.setDrzava(drzava);

    Korisnik korisnik = new Korisnik();
    korisnik.setEmail("test@gmail.com");
    korisnik.setMjesto(mjesto);

    Mockito.when(drzavaRepository.findByNazivDrzave("Hrvatska")).thenReturn(drzava);
    Mockito.when(mjestoRepository.findByPostBrAndNazMjesta(postBr: "51557", nazMjesta: "cres")).thenReturn(Optional.empty());
    Mockito.when(korisnikRepository.save(any(Korisnik.class))).thenAnswer(invocationOnMock -> invocationOnMock.getArguments()[0]); // prvo dodano u bazu

    Korisnik rez= korisnikService.spremiKorisnika(korisnik);
    assertEquals(UlogaKorisnika.REGISTRIRAN, rez.getUloga());
    assertEquals(expected: "cres", rez.getMjesto().getNazMjesta());
    assertEquals(expected: "51557", rez.getMjesto().getPostBr());
}
}
```

Slika 6.1.2

### 3. Test: Promjena uloge korisnika s nepostojećim ID-jem

**1. Funkcionalnost koju testiramo:** Ovim testom ispituje se funkcionalnost promjene uloge korisnika u sustavu u slučaju kada korisnik s navedenim identifikatorom ne postoji u bazi podataka. Cilj testa je provjeriti ispravno rukovanje greškama unutar metode updateRole u servisnoj klasi KorisnikService.

#### 2. Ispitni slučaj

##### Ulazni podaci:

- ID korisnika: 1
- Nova uloga: "VLASNIK"

##### Očekivani rezultati:

- Metoda updateRole baca RuntimeException
- Poruka iznimke glasi: "Korisnik s ID 1 ne postoji"

##### Dobiveni rezultati:

- Bačena je RuntimeException
- Poruka iznimke: "Korisnik s ID 1 ne postoji"

**3. Postupak ispitivanja:** Prije provođenja ispitivanja mockira se repozitorij KorisnikRepository kako bi se simuliralo stanje baze podataka u kojem korisnik s navedenim identifikatorom ne postoji. Repozitorij je podešen da za traženi ID vrati prazan rezultat. Zatim se poziva metoda updateRole nad servisom KorisnikService s definiranim ulaznim parametrima. Tijekom izvođenja

metode očekuje se bacanje iznimke, što se provjerava pomoću JUnit mehanizma assertThrows. Na kraju se provjerava poruka iznimke te se na temelju dobivenih rezultata utvrđuje uspješnost ispitivanja.

```
@Test & LeniBosnic
void updateRole_UserIdNotFound(){
    Integer korisnikId = 1;
    String novaUloga = "VLASNIK";

    Mockito.when(korisnikRepository.findById(korisnikId)).thenReturn( t: Optional.empty());

    RuntimeException exception = assertThrows(RuntimeException.class, () -> {
        korisnikService.updateRole(korisnikId, novaUloga);
    });

    assertEquals( expected: "Korisnik s ID " + korisnikId + " ne postoji", exception.getMessage());
}
```

Slika 6.1.3

## RecenzijaServiceTests

### 1. Test: Uspješno spremanje recenzije

**1. Funkcionalnost koju testiramo:** Ovim testom ispituje se funkcionalnost spremanja nove recenzije u sustav putem metode spremiRecenziju u servisnoj klasi RecenzijaService. Test provjerava ispravno povezivanje recenzije s postojećim korisnikom te pravilno spremanje ocjene i komentara.

#### 2. Ispitni slučaj

##### Ulagni podaci:

- E-mail korisnika: [test@gmail.com](mailto:test@gmail.com)
- Ocjena recenzije: 5
- Komentar recenzije: "Odlično"

##### Očekivani rezultati:

- Recenzija se uspješno sprema u sustav
- Ocjena recenzije ima vrijednost 5
- Komentar recenzije odgovara unesenom tekstu
- Recenzija je povezana s ispravnim korisnikom

##### Dobiveni rezultati:

- Ocjena recenzije: 5
- Komentar recenzije: "Odlično"
- Recenzija je povezana s korisnikom [test@gmail.com](mailto:test@gmail.com)

**3. Postupak ispitivanja:** Prije pokretanja ispitivanja mockiraju se repozitoriji RecenzijaRepository i KorisnikRepository kako bi se simuliralo ponašanje baze podataka bez korištenja stvarne baze. Repozitorij korisnika podešen je tako da za zadalu e-mail adresu vrati postojećeg korisnika. Nakon toga pripremaju se ulazni podaci u obliku objekta RecenzijaRequestDTO, koji sadrži ocjenu i komentar recenzije. Zatim se poziva metoda spremiRecenziju nad servisom RecenzijaService. Na kraju se dobiveni rezultat provjerava pomoću JUnit assercija te se na temelju usporedbe očekivanih i dobivenih rezultata utvrđuje uspješnost ispitivanja.

```

@Test & LeniBosnic
void spremiRecenziju_Successful(){
    String korisnikEmail = "test@gmail.com";
    Korisnik korisnik = new Korisnik();
    korisnik.setEmail(korisnikEmail);

    RecenzijaRequestDTO recReqDTO = new RecenzijaRequestDTO();
    recReqDTO.setValue(5);
    recReqDTO.setKomentar("Očito");

    Mockito.when(korisnikRepository.findByEmail(korisnikEmail)).thenReturn(Optional.of(korisnik));
    Mockito.when(recenzijaRepository.save(any(Recenzija.class))).thenAnswer(i -> i.getArguments()[0]); // vracamo prvi spremljeni argument, u nasem slučaju jedini mockani

    Recenzija recenzija = recenzijaService.spremiRecenziju(korisnikEmail, recReqDTO);
    assertEquals(expected: 5, recenzija.getOcjena());
    assertEquals(expected: "Očito", recenzija.getKomentar());
    assertEquals(korisnik, recenzija.getKorisnik());
}

```

Slika 6.1.4

## 2. Test: Spremanje recenzije s nepostojećim korisnikom

**1. Funkcionalnost koju testiramo:** Ovim testom ispituje se ponašanje metode spremiRecenziju u servisnoj klasi RecenzijaService u slučaju kada korisnik s navedenom e-mail adresom ne postoji u sustavu. Cilj testa je provjeriti ispravno rukovanje greškama i vraćanje odgovarajućeg HTTP statusa.

### 2. Ispitni slučaj

#### Ulazni podaci:

- E-mail korisnika: [test@gmail.com](mailto:test@gmail.com)
- Podaci recenzije: prazan RecenzijaRequestDTO

#### Očekivani rezultati:

- Metoda spremiRecenziju baca ResponseStatusException
- HTTP status iznimke je 404 NOT FOUND

#### Dobiveni rezultati:

- Baćena je ResponseStatusException
- HTTP status iznimke: 404 NOT FOUND

**3. Postupak ispitivanja:** Prije provođenja ispitivanja mockira se repozitorij KorisnikRepository kako bi se simuliralo stanje baze podataka u kojem korisnik s navedenom e-mail adresom ne postoji. Repozitorij je podešen da za traženu e-mail adresu vrati prazan rezultat. Zatim se poziva metoda spremiRecenziju nad servisom RecenzijaService s definiranim ulaznim podacima. Tijekom izvođenja metode očekuje se bacanje iznimke tipa ResponseStatusException, što se provjerava pomoću JUnit assercije. Na kraju se provjerava HTTP status iznimke te se na temelju dobivenih rezultata utvrđuje uspješnost ispitivanja.

```

@Test & LeniBosnic
void spremiRecenziju_EmailNotFound(){
    String korisnikEmail = "test@gmail.com";
    RecenzijaRequestDTO recReqDTO = new RecenzijaRequestDTO();

    Mockito.when(korisnikRepository.findByEmail(korisnikEmail)).thenReturn(Optional.empty());

    assertEquals(HttpStatus.NOT_FOUND, assertThrows(ResponseStatusException.class, () -> {
        recenzijaService.spremiRecenziju(korisnikEmail, recReqDTO);
    }).getStatusCode());
}

```

# RezervacijaServiceTests

## 1. Test: Uspješno kreiranje rezervacije

**1. Funkcionalnost koju testiramo:** Ovim testom ispituje se funkcionalnost kreiranja nove rezervacije u sustavu putem metode kreirajRezervaciju u servisnoj klasi RezervacijaService. Test provjerava ispravno povezivanje rezervacije s postojećim korisnikom te uspješno spremanje rezervacije u sustav.

### 2. Ispitni slučaj

#### Ulazni podaci:

- ID korisnika: 1
- Postojeći korisnik u sustavu

#### Očekivani rezultati:

- Rezervacija se uspješno kreira i sprema u sustav
- Metoda kreirajRezervaciju vraća ID novokreirane rezervacije
- Vraćeni ID rezervacije ima vrijednost 2

#### Dobiveni rezultati:

- Vraćeni ID rezervacije: 2

**3. Postupak ispitivanja:** Prije provođenja ispitivanja mockiraju se repozitoriji RezervacijaRepository i KorisnikRepository kako bi se simuliralo ponašanje baze podataka bez korištenja stvarne baze. Repozitorij korisnika podešen je tako da za zadani identifikator korisnika vrati postojeći zapis. Repozitorij rezervacija konfiguriran je da prilikom spremanja vrati objekt rezervacije s definiranim identifikatorom. Nakon toga poziva se metoda kreirajRezervaciju nad servisom RezervacijaService. Na kraju se dobiveni rezultat provjerava pomoću JUnit assercije te se na temelju dobivenog rezultata utvrđuje uspješnost ispitivanja.

```
@Test @LeniBosnic
void kreirajRezervaciju_Successful(){
    Integer korisnikId = 1;
    Korisnik korisnik = new Korisnik();
    korisnik.setId(korisnikId);

    Integer rezervacijaId = 2;
    Rezervacija rezervacija=new Rezervacija();
    rezervacija.setId(rezervacijaId);

    Mockito.when(korisnikRepository.findById(korisnikId)).thenReturn(Optional.of(korisnik));
    Mockito.when(rezervacijaRepository.save(any(Rezervacija.class))).thenReturn(rezervacija);

    assertEquals( expected: 2, rezervacijaService.kreirajRezervaciju(korisnikId));
}
```

## 2. Test: Ažuriranje rezervacije s nepostojećim ID-jem

**1. Funkcionalnost koju testiramo:** Ovim testom ispituje se ponašanje metode azurirajRezervaciju u servisnoj klasi RezervacijaService u slučaju kada rezervacija s navedenim identifikatorom ne postoji u sustavu. Cilj testa je provjeriti ispravno rukovanje greškama i vraćanje odgovarajuće poruke iznimke.

## 2. Ispitni slučaj

### Ulazni podaci:

- ID rezervacije: 1
- Objekt rezervacije: prazan Rezervacija

### Očekivani rezultati:

- Metoda azurirajRezervaciju baca IllegalArgumentException
- Poruka iznimke glasi: "Rezervacija ne postoji"

### Dobiveni rezultati:

- Bačena je IllegalArgumentException
- Poruka iznimke: "Rezervacija ne postoji"

**3. Postupak ispitivanja:** Prije provođenja ispitivanja mockira se repozitorij RezervacijaRepository kako bi se simuliralo stanje baze podataka u kojem rezervacija s navedenim identifikatorom ne postoji. Repozitorij je podešen da za bilo koji ID vrati prazan rezultat (Optional.empty()). Zatim se poziva metoda azurirajRezervaciju nad servisom RezervacijaService s definiranim ulaznim podacima. Tijekom izvođenja metode očekuje se bacanje iznimke tipa IllegalArgumentException, što se provjerava pomoću JUnit assercije. Na kraju se provjerava poruka iznimke te se na temelju dobivenih rezultata utvrđuje uspješnost ispitivanja.

```
@Test
void azurirajRezervaciju_RezervacijaIDNotFound(){
    Integer rezervacijaId = 1;
    Rezervacija rezervacija = new Rezervacija();

    Mockito.when(rezervacijaRepository.findById(any())).thenReturn(Optional.empty());

    InvalidArgumentException exception = assertThrows(InvalidArgumentException.class, () -> {
        rezervacijaService.azurirajRezervaciju(rezervacijaId, rezervacija);
    });

    assertEquals("Rezervacija ne postoji", exception.getMessage());
}
```

Slika 6.1.7

## SobaServiceTests

### 1. Test: Dohvat sobe po nepostojećem ID-ju

**1. Funkcionalnost koju testiramo:** Ovim testom ispituje se ponašanje metode getSobaByld u servisnoj klasi SobaService u slučaju kada soba s navedenim identifikatorom ne postoji u sustavu. Cilj testa je provjeriti ispravno rukovanje greškama i vraćanje odgovarajuće poruke iznimke.

## 2. Ispitni slučaj

### Ulazni podaci:

- ID sobe: 1

#### Očekivani rezultati:

- Metoda getSobaById baca RuntimeException
- Poruka iznimke glasi: "Soba s ID 1 nije pronađena."

#### Dobiveni rezultati:

- Bačena je RuntimeException
- Poruka iznimke: "Soba s ID 1 nije pronađena."

**3. Postupak ispitivanja:** Prije provođenja ispitivanja mockira se repozitorij SobaRepository kako bi se simuliralo stanje baze podataka u kojem soba s navedenim identifikatorom ne postoji. Repozitorij je podešen da za traženi ID vrati prazan rezultat (Optional.empty()). Zatim se poziva metoda getSobaById nad servisom SobaService s definiranim ID-jem sobe. Tijekom izvođenja metode očekuje se bacanje iznimke tipa RuntimeException, što se provjerava pomoću JUnit assercije. Na kraju se provjerava poruka iznimke te se na temelju dobivenih rezultata utvrđuje uspješnost ispitivanja.

```
@Test @LeniBosnic
void getSobaById_SobaIdNotFound(){
    Integer sobaId = 1;

    Mockito.when(sobaRepository.findById(sobaId)).thenReturn( Optional.empty());

    RuntimeException exception = assertThrows(RuntimeException.class, () -> {
        sobaService.getSobaById(sobaId);
    });

    assertEquals( expected: "Soba s ID " + sobaId + " nije pronađena.", exception.getMessage());
}
```

Slika 6.1.8

## 2. Test: Dohvat sobe kada nema dostupnih soba

**1. Funkcionalnost koju testiramo:** Ovim testom ispituje se ponašanje metode dohvatiSobe u servisnoj klasi SobaService kada za traženi period i kriterije nema dostupnih soba. Cilj testa je provjeriti ispravno rukovanje situacijom u kojoj nije moguće rezervirati sobu te vraćanje odgovarajućeg HTTP statusa.

#### 2. Ispitni slučaj

##### Ulazni podaci:

- Period rezervacije: od danas do 5 dana u budućnosti
- Vrsta sobe: DVOKREVETNA\_TWIN
- Balkon: false
- Pogled na more: true

##### Očekivani rezultati:

- Metoda dohvatiSobe baca ResponseStatusException
- HTTP status iznimke je 409 CONFLICT

##### Dobiveni rezultati:

- Bačena je ResponseStatusException
- HTTP status iznimke: 409 CONFLICT

**3. Postupak ispitivanja:** Prije provođenja ispitivanja mockiraju se rezervirajSobuRepository i SobaRepository. Repozitorij rezervacija soba vraća listu nedostupnih soba za zadani period, dok repozitorij soba simulira da nema dostupnih soba koje odgovaraju traženim kriterijima. Zatim se pripremaju ulazni podaci u obliku RezervacijaRequestDTO i SobaRequestDTO s definiranim periodom i željenim karakteristikama sobe. Nakon toga se poziva metoda dohvatiSobe nad servisom SobaService. Tijekom izvođenja metode očekuje se bacanje iznimke tipa ResponseStatusException, što se provjerava pomoću JUnit assercije. Na kraju se provjerava HTTP status iznimke te se na temelju dobivenih rezultata utvrđuje uspješnost ispitivanja.

```

@Test @LeniBosnic
void dohvatiSobu_NemaDostupnih(){
    LocalDate from = LocalDate.now();
    LocalDate to = LocalDate.now().plusDays( daysToAdd: 5);

    RezervacijaRequestDTO rezReqDTO = new RezervacijaRequestDTO();
    rezReqDTO.setDatumOd(from);
    rezReqDTO.setDatumDo(to);

    SobaRequestDTO sobaReqDTO = new SobaRequestDTO();
    sobaReqDTO.setVrsta(VrstaSobe.DVOKREVETNA_TWIN);
    sobaReqDTO.setBalkon(false);
    sobaReqDTO.setPogledNaMore(true);
    rezReqDTO.setSobe(List.of(sobaReqDTO));

    // da ne pukne kod
    Mockito.when(rezervirajSobuRepository.findNedostupneSobeById(from, to)).thenReturn(List.of(1, 2));
    // vraca prazno za dostupne
    Mockito.when(sobaRepository.findDostupneSobeIds(any(), anyBoolean(), anyBoolean(), any()))
        .thenReturn(Collections.emptyList());

    // nije dovoljno jer test prolazi i ako baci neki drugi HttpStatus npr. 404
    // assertThrows(ResponseStatusException.class, () -> {
    //     sobaService.dohvatiSobe(rezReqDTO);
    // });

    assertEquals(HttpStatus.CONFLICT, assertThrows(ResponseStatusException.class, () -> {
        sobaService.dohvatiSobe(rezReqDTO);
    }).getStatusCode());
}

```

Slika 6.1.9

## Prikaz svih izvršenih testova

✓ ✓ <default package>	1 sec 199 ms
✓ ✓ KorisnikServiceTests	978 ms
✓ updateRole_UserIdNotFound()	918 ms
✓ spremiKorisnika_NewMjesto()	55 ms
✓ spremiKorisnika_MjestoExistsAndAdminEmail()	5 ms
✓ ✓ SobaServiceTests	123 ms
✓ dohvatiSobu_NemaDostupnih()	116 ms
✓ getSobaById_SobaidNotFound()	7 ms
✓ ✓ RezervacijaServiceTests	60 ms
✓ azurirajRezervaciju_RezervacijaIDNotFound()	42 ms
✓ kreirajRezervaciju_Successful()	18 ms
✓ ✓ RecenzijaServiceTests	38 ms
✓ spremiRecenziju_Successful()	35 ms
✓ spremiRecenziju_EmailNotFound()	3 ms

Slika 6.1.10

## Ispitivanje sustava

### 1. Test: Brisanje sobe

#### 1. Ulazi:

- Postojeća soba u sustavu
- Korisnik je prijavljen i ima ovlasti za brisanje sobe

#### 2. Koraci ispitivanja:

- Otvoriti web aplikaciju i postaviti veličinu prozora
- Navigirati do popisa soba
- Postaviti pokazivač miša na gumb s dodatnim opcijama za sobu
- Kliknuti na opciju Brisanje sobe
- Potvrditi dijaloški prozor za brisanje (OK)

Command	Target	Value
1 ✓ open	/	
2 ✓ set window size	785x799	
3 ✓ run script	window.scrollTo(0,0)	
4 ✓ mouse over	css=div:nth-child(3) > .MuiButtonBase-root	
5 ✓ mouse out	css=div:nth-child(3) > .MuiButtonBase-root	
6 ✓ click	css=div:nth-child(3) > .MuiButtonBase-root	
7 ✓ mouse over	css=.MuiButtonBase-root:nth-child(4)	
8 ✓ mouse over	css=.MuiButton-contained:nth-child(3)	
9 ✓ mouse over	css=.MuiButtonBase-root:nth-child(5)	
10 ✓ mouse down at	css=body	744,184
11 ✓ mouse move at	css=body	744,184
12 ✓ mouse up at	css=body	744,184
13 ✓ choose ok on next confirmation		
14 ✓ click	css=div:nth-child(6) .MuiButton-contained	

Slika 6.2.1

### 3. Očekivani izlaz:

- Sustav prikazuje dijalog za potvrdu brisanja sobe
- Nakon potvrde, odabrana soba se briše iz sustava
- Soba se više ne prikazuje na popisu soba

### 4. Dobiveni izlaz:

Log	Reference	
Running 'Brisanje sobe'		14:50:17
1. open on / OK		14:50:17
2. setWindowSize on 785x799 OK		14:50:17
3. runScript on window.scrollTo(0,0) OK		14:50:17
4. mouseOver on css=div:nth-child(3) > .MuiButtonBase-root OK		14:50:20
5. mouseOut on css=div:nth-child(3) > .MuiButtonBase-root OK		14:50:20
6. click on css=div:nth-child(3) > .MuiButtonBase-root OK		14:50:20
7. mouseOver on css=.MuiButtonBase-root:nth-child(4) OK		14:50:20
8. mouseOver on css=.MuiButton-contained:nth-child(3) OK		14:50:21
9. mouseOver on css=.MuiButtonBase-root:nth-child(5) OK		14:50:21
10. mouseDownAt on css=body with value 744,184 OK		14:50:21
11. mouseMoveAt on css=body with value 744,184 OK		14:50:22
12. mouseUpAt on css=body with value 744,184 OK		14:50:22
13. chooseOkOnNextConfirmation OK		14:50:22
14. click on css=div:nth-child(8) .MuiButton-contained OK		14:50:23
15. assertConfirmation on Jeste li sigurni da želite obrisati ovu sobu? OK		14:50:23
16. webdriverChooseOkOnVisibleConfirmation OK		14:50:24
17. mouseOver on css=div:nth-child(7) .MuiButton-outlined OK		14:50:24
18. mouseOver on css=div:nth-child(8) .MuiButton-contained OK		14:50:24
19. click on css=.nova_soba OK		14:50:24
'Brisanje sobe' completed successfully		14:50:24

Slika 6.2.2

## 2. Test: Dodavanje sobe

### 1. Ulazi:

- Korisnik je prijavljen i ima ovlasti za dodavanje sobe
- Postavke sobe (npr. dvokrevetna soba)

### 2. Koraci ispitivanja:

- Otvoriti web aplikaciju i postaviti veličinu prozora
- Navigirati do modula za upravljanje sobama
- Kliknuti na gumb Nova soba
- Pričekati da se obrazac za unos prikaže
- Unijeti broj sobe u polje za unos
- Označiti odgovarajuće opcije (npr. dvokrevetna soba)
- Kliknuti gumb za spremanje

Command	Target	Value
✓ mouse over	css=.nova_soba_btn	
✓ click	css=.nova_soba_btn	
✓ wait for element visible	css=[data-testid="room-number-input"]	3000
✓ click	css=[data-testid="room-number-input"] input	
✓ execute script	const input = document.querySelector('[data-testid="room-number-input"] input'); input.value = '185'; input.dispatchEvent(new Event('input', { bubbles: true }));	
✓ click	id=_r_7o_	
✓ click	css=.MuiFormControlLabel-root:nth-child(1) > .MuiCheckbox-root > .PrivateSwitchBase-input	
✓ click	css=.css-1cfak8w-MuiButtonBase-root-MuiButton-root	
✓ mouse down at	css=body	701,667
✓ mouse move at	css=body	701,667
✓ mouse up at	css=body	701,667
✓ mouse over	css=.MuiButton-text:nth-child(3)	

Slika 6.2.3

### 3. Očekivani izlaz:

- Sustav prikazuje obrazac za dodavanje nove sobe
- Uneseni podaci se prihvaćaju bez greške
- Nova soba se uspješno sprema u sustav
- Nova soba je vidljiva na popisu soba

### 4. Dobiveni izlaz:

Log	Reference	
1. open on / OK		14:59:02
2. setWindowSize on 785x799 OK		14:59:02
3. runScript on window.scrollTo(0,0) OK		14:59:02
4. mouseOver on css=div:nth-child(3) > .MuiButtonBase-root OK		14:59:05
5. mouseOut on css=div:nth-child(3) > .MuiButtonBase-root OK		14:59:05
6. mouseOver on linkText=Modriš OK		14:59:05
7. mouseOut on linkText=Modriš OK		14:59:05
8. click on css=div:nth-child(3) > .MuiButtonBase-root OK		14:59:05
9. mouseOver on css=.MuiButtonBase-root:nth-child(5) OK		14:59:06
10. mouseOver on css=.nova_soba_btn OK		14:59:06
11. click on css=.nova_soba_btn OK		14:59:06
12. waitForElementVisible on css=[data-testid="room-number-input"] with value 3000 OK		14:59:06
13. click on css=[data-testid="room-number-input"] input OK		14:59:07

Slika 6.2.4

## 3. Test: Filtriranje rezervacija

### 1. Ulazi:

- Početni datum filtriranja: 2026-01-22
- Završni datum filtriranja: 2026-01-28

### 2. Koraci ispitivanja:

- Otvoriti web-aplikaciju za upravljanje rezervacijama
- Kliknuti na gumb za otvaranje filtera rezervacija
- U polje za početni datum unijeti 2026-01-22

- U polje za završni datum unijeti 2026-01-28
- Kliknuti na gumb za primjenu filtera

Command	Target	Value
✓ setWindowSize	css=body	
✓ run script	window.scrollTo(0,0)	
✓ mouse down at	css=body	703,271
✓ mouse move at	css=body	703,271
✓ mouse up at	css=body	703,271
✓ mouse over	css=div:nth-child(3) > .MuiButtonBase-root	
✓ click	css=div:nth-child(3) > .MuiButtonBase-root	
✓ click	css=.MuiButtonBase-root:nth-child(4)	
✓ click	css=div:nth-child(2) > input	
✓ type	css=div:nth-child(2) > input	2026-01-22
✓ mouse over	css=.MuiButton-text:nth-child(3)	
✓ mouse out	css=.MuiButton-text:nth-child(3)	
✓ click	css=div:nth-child(3) > input	
✓ type	css=div:nth-child(3) > input	2026-01-28
✓ click	css=tr:nth-child(2) > td:nth-child(7)	
✓ mouse over	css=.MuiButtonBase-root:nth-child(5)	

Slika 6.2.5

### 3. Očekivani izlaz:

- Sustav prihvata unesene datume bez greške
- Prikazuju se samo rezervacije čiji se datum nalazi unutar zadatog raspona (22.01.2026 – 28.01.2026)
- Rezervacije izvan tog raspona nisu prikazane

### 4. Dobiveni izlaz:

Running "Filtriranje rezervacija"	15:00:03
1. open on / OK	15:00:03
2. setWindowSize on 785x799 OK	15:00:04
3. runScript on window.scrollTo(0,0) OK	15:00:04
4. mouseDownAt on css=body with value 703,271 OK	15:00:07
5. mouseMoveAt on css=body with value 703,271 OK	15:00:07
6. mouseUpAt on css=body with value 703,271 OK	15:00:07
7. mouseOver on css=div:nth-child(3) > .MuiButtonBase-root OK	15:00:07
8. click on css=div:nth-child(3) > .MuiButtonBase-root OK	15:00:07
9. click on css=.MuiButtonBase-root:nth-child(4) OK	15:00:08
10. click on css=div:nth-child(2) > input OK	15:00:08
11. type on css=div:nth-child(2) > input with value 2026-01-22 OK	15:00:08
12. mouseOver on css=.MuiButton-text:nth-child(3) OK	15:00:09
13. mouseOut on css=.MuiButton-text:nth-child(3) OK	15:00:09
14. click on css=div:nth-child(3) > input OK	15:00:09
15. type on css=div:nth-child(3) > input with value 2026-01-28 OK	15:00:09
16. click on css=tr:nth-child(2) > td:nth-child(7) OK	15:00:09
17. mouseOver on css=.MuiButtonBase-root:nth-child(5) OK	15:00:10
'Filtriranje rezervacija' completed successfully	15:00:10

Slika 6.2.6

## 4. Test: Unos recenzije

### 1. Ulazi:

- Dostupna navigacijska stavka Recenzije
- Vidljivo polje za unos recenzije

### 2. Koraci ispitivanja:

- Otvoriti web aplikaciju u pregledniku.
- Podesiti veličinu prozora preglednika.
- Kliknuti na navigacijsku stavku Recenzije.
- Kliknuti u polje za unos recenzije.
- Unijeti tekst „Sve je u redu.“ u polje za recenziju.
- Kliknuti na gumb za slanje recenzije.

Command	Target	Value
✓ open	/	
✓ set window size	785x692	
✓ click	css=path	
✓ mouse over	linkText=Rezervacija soba	
✓ mouse out	linkText=Rezervacija soba	
✓ mouse over	linkText=Rezervacija dodatnog sadržaja	
✓ mouse out	linkText=Rezervacija dodatnog sadržaja	
✓ mouse over	linkText=Recenzije	
✓ click	linkText=Recenzije	
✓ run script	window.scrollTo(0,0)	
✓ click	id=_r_f_	
✓ mouse over	css=.MuiButton-contained	
✓ wait for element visible	id=_r_f_	30000
✓ execute script	id=_r_f_	var el = document.getElementById('_r_f_');\nel.value = 'Sve je u redu.';\nel.dispatchEvent(new Event('input', { bubble: true }));
✓ click	css=.MuiButton-contained	

Slika 6.2.7

### 3. Očekivani izlaz:

- Forma za unos recenzije je vidljiva i aktivna.
- Tekst recenzije je uspješno unesen u polje.
- Nakon klika na gumb za slanje, recenzija se prihvata i obrađuje bez greške.
- Sustav ne prikazuje poruke o pogrešci.

### 4. Dobiveni izlaz:

Running 'Unos recenzija'	15:01:11
1. open on / OK	15:01:12
2. setWindowSize on 785x692 OK	15:01:12
3. click on css=path OK	15:01:12
4. mouseOver on linkText=Rezervacija soba OK	15:01:15
5. mouseOut on linkText=Rezervacija soba OK	15:01:15
6. mouseOver on linkText=Rezervacija dodatnog sadržaja OK	15:01:15
7. mouseOut on linkText=Rezervacija dodatnog sadržaja OK	15:01:15
8. mouseOver on linkText=Recenzije OK	15:01:15
9. click on linkText=Recenzije OK	15:01:15
10. runScript on window.scrollTo(0,0) OK	15:01:16

Slika 6.2.8

## 5. Test:

### 1. Ulazi:

- Prijavljen korisnik s administratorskim ovlastima
- Dostupna opcija Uređivanje dodatnog sadržaja

### 2. Koraci ispitivanja:

- Otvoriti web aplikaciju u pregledniku.
- Podesiti veličinu prozora preglednika.
- Otvoriti sekciju za uređivanje dodatnog sadržaja.
- Kliknuti na odgovarajući gumb za uređivanje.
- Privremeno ugasiti dodatan sadržaj
- Spremiti promjene

	Command	Target	Value
6	✓ mouse out	css=div:nth-child(3) > .MuiButton-textInherit	
7	✓ mouse over	css=.MuiButtonBase-root:nth-child(5)	
8	✓ click	css=.MuiButtonBase-root:nth-child(5)	
9	✓ mouse out	css=.active	
10	✓ mouse down at	css=body	748,202
11	✓ mouse move at	css=body	748,202
12	✓ mouse up at	css=body	748,202
13	✓ click	css= base--focused > .base-NumberInput-incrementButton	
14	✓ click	css= base--focused > .base-NumberInput-incrementButton	
15	✓ double click	css= base--focused > .base-NumberInput-incrementButton	

Slika 6.2.9

### 3. Očekivani izlaz:

- Sekcija za uređivanje dodatnog sadržaja se ispravno otvara.
- Promjene se uspješno spremaju bez greške.
- Sustav ne prikazuje poruke o pogrešci.

### 4. Dobiveni izlaz:

```

Running "Uređivanje dodatnog sadržaja"
1. open on / OK
2. setWindowSize on 765x799 OK
3. runScript on window.scrollTo(0,0) OK
4. mouseOver on css=div:nth-child(3) > .MuiButtonBase-root OK
5. click on css=div:nth-child(3) > .MuiButtonBase-root OK
6. mouseOut on css=div:nth-child(3) > .MuiButton-textInherit OK
7. mouseOver on css=.MuiButtonBase-root:nth-child(5) OK
8. click on css=.MuiButtonBase-root:nth-child(5) OK
9. mouseOut on css=.active OK
10. mouseDownAt on css=body with value 748,202 OK
11. mouseMoveAt on css=body with value 748,202 OK
12. mouseUpAt on css=body with value 748,202 OK
13. Trying to find css=.base--focused > .base-NumberInput-incrementButton... OK
Warning Element found with secondary locator xpath=/div[@id='root']/div/div[2]/div/div[2]/div/div/div/button[2]. To use it by default, update the test step to use it as the primary locator.
14. click on css=.base--focused > .base-NumberInput-incrementButton OK
15. doubleClick on css=.base--focused > .base-NumberInput-incrementButton OK
16. click on css=.base--focused > .base-NumberInput-incrementButton OK
17. click on css=body OK
18. click on css=.adminStartContent > div OK
19. mouseOver on css=.MuiButton-contained:nth-child(1) OK
20. mouseOver on css=.MuiButtonBase-root:nth-child(5) OK
21. click on css=.MuiButtonBase-root:nth-child(5) OK
22. mouseOut on css=.active OK
23. runScript on window.scrollTo(0,0) OK
'Uređivanje dodatnog sadržaja' completed successfully

```

Slika 6.2.10

## Korištene tehnologije i alati

---

### 1. Programski jezici

- **JavaScript** – koristi se za razvoj interaktivnog frontend dijela aplikacije. Omogućuje dinamičko prikazivanje sadržaja i povezivanje sa serverom, a React framework se oslanja na njega.
- **Java (verzija 23)** – koristi se za backend razvoj aplikacije. Omogućuje kreiranje logike poslovanja, obradu podataka, povezivanje s bazom podataka i sigurnu komunikaciju između korisnika i sustava.
- **HTML** – definira strukturu web stranica, pruža semantičko označavanje elemenata i omogućuje preglednicima interpretaciju sadržaja.
- **CSS** – stilizacija elemenata i vizualni izgled aplikacije, uključujući fontove, boje, raspored i responzivni dizajn za različite uređaje.

### 2. Radni okviri i biblioteke

- **React (Vite)** – JavaScript framework za frontend. Omogućuje razvoj komponentnog i responzivnog korisničkog sučelja, olakšava ponovno korištenje koda i ažuriranje elemenata sučelja.
- **Spring Boot** – Java backend framework koji pojednostavljuje razvoj web servisa i API-ja, ubrzava konfiguraciju i omogućuje stabilan rad aplikacije.

### 3. Baza podataka

- **PostgreSQL (verzija 17)** – relacijska baza podataka za pohranu korisnika, rezervacija i dodatnih usluga. Podržava složene upite, transakcije i osigurava integritet podataka.

### 4. Razvojni alati

- **Visual Studio Code (VSC)** – glavni alat za frontend razvoj, s podrškom za React, JavaScript i web tehnologije.
- **IntelliJ IDEA** – IDE specijaliziran za backend razvoj u Javi i Spring Boot frameworku.
- **WebStorm** – IDE optimiziran za frontend razvoj i debugging, posebno za React i Vite projekte.

## 5. Alati za ispitivanje

- **JUnit** – Java framework za testiranje. Omogućuje pisanje i izvršavanje automatiziranih testova u Java aplikacijama, posebno u Spring Boot okruženju.
- **Selenium** – alat za automatizirano testiranje web-aplikacija koji omogućuje programima da kontroliraju web-preglednike i simuliraju korisničke radnje.

## 6. Okruženje za hosting

- **Render** – koristi se kao hosting platforma na kojoj se nalazi čitava aplikacija: frontend, backend i baza podataka. Omogućuje automatsko deployanje, skaliranje i stabilno izvođenje servisa, dok istovremeno povezuje sve dijelove sustava u jednu funkcionalnu cjelinu.

## 6. Suradnja i verzioniranje

- **Git** – lokalno praćenje promjena u kodu, omogućuje grananje, spajanje i verzioniranje projektnog koda.
- **GitHub** – online platforma za pohranu koda i timsku suradnju, olakšava push/pull promjene i koordinaciju tima.
- **Microsoft Teams, Discord, WhatsApp** – alati za internu komunikaciju i koordinaciju članova tima tijekom razvoja.

## 7. Dijagrami i vizualizacija

- **Visual Paradigm Online** – korišten za sekvensijske dijagrame, dijagrame aktivnosti i stanja, omogućuje brzo i pregledno crtanje dijagrama online.

## 1. Instalacija

### Potrebni softveri i verzije:

Docker (Verzija 28.5.2)

Node.js (Verzija 11.4.2)

Java (Verzija 22.0.2)

Maven (Verzija ^3.9.0)

Postgresql (17.4)

Git (Verzija 2.45.1)

### Razvojni alati

IntelliJ IDEA (Verzija 25)

Webstorm (Verzija 25)

## Preuzimanje izvornog koda

Izvorni kod mozete skinuti preko command line-a pomocu sljedecih komandi:

```
git clone https://github.com/rokoKnin/ProgiG03.3_JuliusMeinl.git  
cd ./ProgiG03.3_JuliusMeinl/
```

Zatim instalirani projekt mozete otvoriti i pokretati iz IntelliJja i WebStorma

## Preuzimanje dependencia i postavljanje projekta:

Ovdje cu se fokusirati na IntelliJ razvojno okruzenje, ali slicno se sve postavlja u svim ostalim orkuzenjima.

Ovo su upute za lokalno pokretanje:

1. Kad otvorite projekt u IntelliJ-u izaci ce vam pop-up da je prepoznao maven build file, pritisnite build.
2. IntelliJ ce potencijalno i prepoznati da vam fale dependencies iz package.json-a u frontendu pa kliknite na gumb za npm install ili se pozicionirajte u ProgiG03.3\_JuliusMeinl\frontend i pokrenite npm install.

## 2. Postavke

upute za namjestanje kofiguracijskih datoteka

- frontend .env mora sadrzavati ovo:

```
VITE_API_URL = VAS_BACKEND_URL (za lokalno runnanje ce to biti http://localhost:8080)
```

- backend .env mora sadrzavati ovo:

```
DB_URL= VAS_URL_ZA_POSTGRES_BAZU
```

```
DB_USERNAME= POSTGRES_USERNAME
```

```
DB_PASSWORD= SIFRA_ZA_POSTGRES_USERNAME
```

```
JULIUS_FRONTEND_URL= VAS_FRONTEND_URL (za lokalno runnanje ce to biti http://localhost:5173)
```

```
MAIL_PASSWORD= SIFRA_ZA_SMTP_MAIL
```

```
GOOGLE_CLIENT_ID= VAS_GOOGLE_CLIENT_ID
```

```
GOOGLE_CLIENT_SECRET= SIFRA_VASEG_GOOGLE_CLIENT_ID
```

- Postavljanje Baze Podataka:

bazu se postavlja iz filea baza.sql koji se nalazi u folderu bazaPodataka

bazu je moguce i napuniti defaultnim sobama koje se nalaze u sobe\_insert.sql

## 3. Pokretanje aplikacije

- Razvojno okruzenje:

Pokretanje frontnda:



SLika 8.1: Pokretanje frontend-a u razvojnom okruzenju

pokretanje backenda je nalakše pomoću IntelliJ-a pokretanjem direktno iz `BackendApplication.java` (Vidi Sliku 8.2). IntelliJ također omogućuje pokretanje u debug načinu što daje mogućnost detaljnijeg pregleda funkciranja aplikacije. No može se i iz terminala pomoću mavena s funkcijom: `mvn spring-boot:run`.



SLika 8.2: Pokretanje backenda u razvojnom okruzenju

- Producjsko okruzenje:

Buildanje frontnda:



SLika 8.3: Priprema frontend-a za produksijsko okruzenje

Buildanje Docker kontejnera za backend:



SLika 8.4: Priprema backenda za produksijsko okruzenje

## 4. Upute za administratore

Smjernice za administratore aplikacije nakon puštanja u pogon:

- Pristup administratorskom sučelju:
  - [AdminPanel URL](#)
  - Podatci za prijavu (preko OAuth2):
    - email: [juliusmeint3.3@gmail.com](mailto:juliusmeint3.3@gmail.com)
    - sifra: progiT3.3
- Redovito održavanje:
  - *Pregled logova:*
    - Logovi aplikacije dostupni su putem Render sučelja, gdje se mogu pregledati za dijagnosticiranje grešaka i praćenje aktivnosti.
  - *Ažuriranje aplikacije:*
    - Render preuzima direktno s DockerHuba docker kontejner s tagom latest, tako da se prilikom izmjena vrlo lako može deployati ažurirana verzija.

## 5. Primjer za Render platformu (Cloud Deploy)

Render je popularna cloud platforma za jednostavno smještanje aplikacija.

Prvi korak postavljanja aplikacije na Render je kreiranje svog racuna i workspacea

- Postavljanje na Render:
  - Baza Podataka:
    - Napravi se novi service 'Postgres'

- pomocu SQL Shella se spoji sa svog računala na bazu na Renderu i pokrenu se potrebne skripte za učitavanje podataka
- Frontend:
  - Napravi se novi service: 'Static site'
  - poveže ga se s githubom
  - postave se enviromental varijable
- Backend:
  - Napravi se novi service: 'Web Service'
  - Poveže ga se ili sa githubom ili sa dockerhubom
  - postave se sve enviromental varijable
    - potrebno je izmijeniti link na bazu podataka da bude oblika: jdbc:postgresql://dpg-#####:PORT/NAZIV\_BAZE\_####

## Opis prisutpa aplikaciji na javnom poslužitelju

---

- Ograničenja:
  - Aplikacija je deployana na besplatnoj verziji Rendera. To povlači da je relativno spora, ne podržava veliki broj istovremenih korisnika.
  - Baza Podataka traje mjesec dana prije nego što ju Render obriše.
- Korisnici pristupaju aplikaciji preko [URL-a](#)
- Repcionist pristupa ili preko istog URL-a pa klikom na gumb Receptionist ili direktno preko [Repcionist URL-a](#)
- Administratori pristupaju ili preko istog URL-a pa klikom na gumb Admin ili direktno preko [Admin URL-a](#)

## Zaključak

---

Zadatak naše grupe bio je osmisлити и razviti modernu hotelsku web aplikaciju koja omogуује jednostavnu rezervaciju soba i dodatnih sadržaja, uz sigurnu prijavu korisnika putem Google OAuth2.0 sustava. Kroz nekoliko mjeseci rada prošli smo sve faze razvoja aplikacije, od početne ideje i planiranja do konačne implementacije i testiranja funkcionalnosti.

U prvom ciklusu nastave definirali smo koncept aplikacije, ključne funkcionalnosti i strukturu sustava. Nakon inicijalnog dogovora, tim se podijelio prema područjima rada, pri čemu je svaki dio tima imao jasno definirane zadatke. Postavljeni su temelji aplikacije, uključujući bazu podataka, sustav za autentifikaciju korisnika, osnovne funkcionalnosti prikaza soba te početni izgled stranica.

Drugi ciklus bio je usmjeren na nadogradnju i dovršavanje sustava. Implementirane su mogućnosti poput rezervacije soba i dodatnih hotelskih sadržaja, poboljšan je korisnički doživljaj te su izvršene prilagodbe dizajna kako bi aplikacija bila intuitivna i pregledna. Posebna pažnja posvećena je testiranju i ispravljanju uočenih nedostataka.

Tijekom razvoja susreli smo se s nizom izazova, ponajviše vezanih uz integraciju Google login sustava, upravljanje korisničkim sesijama i deploy aplikacije. Nedostatak prethodnog iskustva u timskom radu i korištenju određenih tehnologija dodatno je produljio vrijeme izrade, no svi su problemi uspješno riješeni uz dobru međusobnu komunikaciju te dodatni trud članova tima.

Ovaj projekt svim je članovima tima pružio vrijedno iskustvo rada na stvarnoj web aplikaciji, upoznavanje s modernim alatima i tehnologijama te, što je najvažnije, razvoj timskih i komunikacijskih vještina. Stečena znanja i iskustva predstavljaju kvalitetnu pripremu za budući rad u struci, a zaključak projekta potvrđuje kako su dobra organizacija, suradnja i komunikacija ključni za uspješnu realizaciju složenih softverskih rješenja.

1. Programsko inženjerstvo, FER ZEMRIS, <http://www.fer.hr/predmet/proinz>
2. I. Sommerville, "Software engineering", 8th ed, Addison Wesley, 2007.
3. T.C.Lethbridge, R.Langaniere, "Object-Oriented Software Engineering", 2nd ed. McGraw-Hill, 2005.
4. I. Marsic, Software engineering book", Department of Electrical and Computer Engineering, Rutgers University, <http://www.ece.rutgers.edu/~marsic/> books/SE
5. The Unified Modeling Language, <https://www.uml-diagrams.org/>
6. Visual Paradigm online, <https://www.visual-paradigm.com/>  
[draw.io] (<https://app.diagrams.net/>) alat za crtanje dijagrama
7. YouTube (<https://www.youtube.com/>)
8. Wiki profesora Sruka <https://github.com/VladoSruk/Programsko-inzenjerstvo/wiki>
9. Spring boot dokumentacija (<https://docs.spring.io/spring-framework/reference/index.html>)
10. Vite dokumentacija (<https://vite.dev/guide/>)

Rev.	Opis promjene/dodataka	Autori	Datum
0.1	Napravio strukturu wiki projekta	Roko Tojčić	20.10.2025.
0.2	Napravljena 1. Wiki Stranica	Nino Ljubas	21.10.2025.
0.3	Napravljena 2. Wiki Stranica	Roko Tojčić	21.10.2025
0.4	Napravljeni UML dijagrami 3. Wiki Stranice	Andrija Rebić	21.10.2025
0.5	Raspisani UML dijagrami 3. Wiki Stranice	Luka Omrčen	22.10.2025.
0.6	Popravci 2. i 3. Wiki Stranice	Leona Križanac	24.10.2025.
0.7	Početak 4. Wiki Stranice	Leni Bosnić	26.10.2025.
0.8	Raspis baze podataka u 4. Wiki Stranici	Leni Bosnić	29.10.2025.
0.9	Raspisani dijagrami razreda	Leona Križanac	10.11.2025.
1.0	Raspisane A, B i C Stranica za prvu predaju	Roko Tojčić	14.11.2025.
1.1	Napisani dijagrami aktivnosti i dijagram stanja	Luka Omrčen	7.1.2026
1.2	Napravljena 8. Wiki Stranica	Roko Tojčić	19.1.2026.
1.3	Napravljena 5. Wiki Stranica	Luka Omrčen	19.1.2026.
1.4	Napravljena 7. Wiki Stranica	Leona Križanac, Luka Omrčen	20.1.2026.
1.5	Napravljena 6. Wiki Stranica	Luka Omrčen	22.1.2026.
1.6	Napravljena 9. Wiki Stranica	Luka Omrčen	22.1.2026.
1.7	Ispravci dokumentacije i formatiranja	Roko Tojčić	23.1.2026.
1.8	Završene stranice A, B i C za finalnu predaju	Roko Tojčić	23.1.2026.

## Dnevnik sastajanja

---

## 1. Sastanak

- Datum: 8.10.2025.
- Prisustovali: Roko Tojčić, Nino Ljubas, Luka Omrčen, Andrija Rebić, Gracia Hlobik, Leni Bosnić
- Teme Sastanka:

- Dogovoren koristenje Whatsapp grupe, Discord Projectsa
- Prvotna raspodijela na ekipe
- Dogovoren koristenje SpringBoota i Reacta

## 2. Sastanak

- Datum: 16.10.2025.
- Prisustovali: Roko Tojčić, Nino Ljubas, Luka Omrčen, Andrija Rebić, Gracia Hlobik
- Teme Sastanka:

- Nova unutarnja podjela na poslove
- Dogovaranje Funkcijskih Zahtjeva

## 3. Sastanak

- Datum: 21.10.2025.
  - Prisustovali: Roko Tojčić, Nino Ljubas, Luka Omrčen, Andrija Rebić, Gracia Hlobik, Leona Križanac, Leni Bosnić
  - Teme Sastanka:
- komentiranje Trenutacnog raspisa prve tri stranice wiki-a
  - Finalna raspodjela poslova među članovima :)
  - Dogovorena zaduzenja do sljedećeg tjedna

## 4. Sastanak

- Datum: 27.10.2025.
  - Prisustovali: Roko Tojčić, Nino Ljubas, Luka Omrčen, Andrija Rebić, Gracia Hlobik, Leona Križanac, Leni Bosnić
  - Teme Sastanka:
- Komentiranje potrebnih popravaka UML dijagrama
  - Komentiranje potrebnih popravaka Baze podataka
  - Dogovorena zaduzenja do sljedećeg tjedna

## 5. Sastanak

- Datum: 3.11.2025.
  - Prisustovali: Roko Tojčić, Nino Ljubas, Luka Omrčen, Andrija Rebić, Gracia Hlobik, Leona Križanac, Leni Bosnić
  - Teme Sastanka:
- Dogovorene specifikacije Hotela za potrebe prve funkcionalnosti
  - Komentirano stanje dokumentacije, frontenda i backenda
  - Rasporeden posao za rad na drugoj funkcionalnosti

## 6. Sastanak

- Datum: 10.11.2025.
- Prisustovali: Roko Tojčić, Nino Ljubas, Luka Omrčen, Andrija Rebić, Gracia Hlobik, Leona Križanac, Leni Bosnić
- Teme Sastanka:
  - Podsjetnici na obaveze prije predaje dokumentacije
  - Finalni poslovi za dokumentaciju
  - Zadnji preostali poslovi prije labosa u srijedu
  - Dogovor oko podizanja aplikacije na web

## 7. Sastanak

- Datum: 18.12.2025.
- Prisustovali: Roko Tojčić, Luka Omrčen, Andrija Rebić, Gracia Hlobik
- Teme Sastanka:
  - prošli kroz sto još trebamo raditi
  - izmijenili pozicije interno

## 8. Sastanak

- Datum: 8.1.2026.
- Prisustovali: Roko Tojčić, Nino Ljubas, Luka Omrčen, Andrija Rebić, Gracia Hlobik, Leona Križanac,
- Teme Sastanka:
  - riješavanje deploja
  - prošli kroz sve preostale funkcionalnosti i raspodijelili se za sljedeći tjedan

## 9. Sastanak

- Datum: 15.1.2026.
- Prisustovali Roko Tojčić, Nino Ljubas, Luka Omrčen, Andrija Rebić, Gracia Hlobik
- Teme Sastanka:
  - Raspodjela zadnjih presotalih poslova pred alpha predaju
  - Dogovoren način na koji će se odraditi testiranje

## Plan rada

- *Prvi ciklus:*



|-----|

| Slika C.1 Prikaz plana rada za prvi ciklus |

- *Drugi ciklus:*



## Tablica aktivnosti

Aktivnost	Roko Tojčić	Nino Ljubas	Luka Omrčen	Andrija Rebić	Gracia Hlobik	Leona Križanac	Leni Bosnić
Upravljanje projektom	17						
Opis projektnog zadatka		5					
Funkcionalni zahtjevi	6				1	3	
Opis pojedinih obrazaca				6		1	
Dijagram obrazaca						3	
Sekvenički dijagrami						6	
Opis ostalih zahtjeva							
Arhitektura i dizajn sustava							7
Baza podataka	6	19					
Dijagram razreda			1			6	
Dijagram stanja			4				
Dijagram aktivnosti			4				
Dijagram komponenti			4				
Arhitektura komponenata i razmjestaja			1				
Korištene tehnologije i alati			1				
Ispitivanje programskog rješenja			3	6	1		5
Upute za puštanje u pogon	2						
Dnevnik sastajanja	1						
Zaključak i budući rad			1				
Popis Literature	1						
Ucenje	8			5	5		
Potpore za OAuth2		11	4	14			
Frontend				34	41		
Backend	2	32	13			17	8
Deployment	32	5				14	2
Povezivanje Backenda i Frontenda	3	6	4	6	9		
Autorizacija	8						
Ukupno:	86	78	40	71	57	50	22

# Dijagram pregleda promjena



Slika C.3 Prikaz ukupnih commitova na GitHubu



Slika C.4 Prikaz pojedinačnih commitova na GitHubu

## Ključni izazovi i rješenja

- Problem autentifikacije preko Googleovog OAuth2 -> Nismo na ispravan način pristupili upravljanjem cookiesa i sesija.
  - Riješili detaljnijim proučavanjem Spring Securitya i načina na koji ti mehanizmi funkcioniraju
- Problem ne pridržavanja istih standarda pisanja
  - Na sastancima se trudilo svima ukazati standarda pisanja koji koristimo da bi sve bilo unificirano

- **Manjak iskustva članova, susretanje s ovakvim projektom prvi puta kao i timskim radom u ovakovom stilu**
  - problemi medu članovima se riješavali kroz komunikaciju, a nedostatke znanja kroz učenje s izvora ili međusobnim pokazivanjem