Projektni zadatak

Hotel Booking System

**Predmet:** SE211 – Konstruisanje softvera

**Student:** Polina Korepanova

**Indeks:** 5859

Contents

[Naslov i opis problema 4](#_Toc172767288)

[Specifikacija zahteva korisnika 4](#_Toc172767289)

[Registracija i prijava korisnika 4](#_Toc172767290)

[Pretraga i rezervacija hotela 4](#_Toc172767291)

[Sistem za ocenjivanje 5](#_Toc172767292)

[Upravljanje rezervacijama 5](#_Toc172767293)

[Pregled informacija o hotelima 5](#_Toc172767294)

[Opis arhitekture 6](#_Toc172767295)

[Procesni model softvera 7](#_Toc172767296)

[Klasni dijagram aplikacije 8](#_Toc172767297)

[Podela sistema na module 9](#_Toc172767298)

[Korisnički Modul 9](#_Toc172767299)

[Hotel Modul 10](#_Toc172767300)

[Rezervacioni Modul 11](#_Toc172767301)

[Recenzioni Modul 12](#_Toc172767302)

[Sekvencijalni dijagram aplikacije 13](#_Toc172767303)

[Glavni scenariji upotrebe 13](#_Toc172767304)

[Prikaz testiranja 14](#_Toc172767305)

[Integracioni testovi 15](#_Toc172767306)

[Implementacija u Javi 16](#_Toc172767307)

[Upotreba alata za otklanjanje grešaka 17](#_Toc172767308)

[Rezultate Inspekcije Koda 17](#_Toc172767309)

[Upozorenja u HTML datotekama 17](#_Toc172767310)

[Neprijavljene Reference Baze Podataka u Anotacijama 18](#_Toc172767311)

[Neiskorišćeni Izrazi 18](#_Toc172767312)

[Nepronađena Lokalno Skladištena Biblioteka za HTTP Link 18](#_Toc172767313)

[Integraciono Testiranje Realizovanog Modula 19](#_Toc172767314)

[Unit Testiranje Realizovanog Modula 20](#_Toc172767315)

# Naslov i opis problema

Postoji potreba za jednostavnim i efikasnim sistemom koji omogućava korisnicima da rezervišu sobe u hotelima, kao i da ocenjuju svoje iskustvo nakon boravka. Hoteli žele da imaju povratne informacije od svojih gostiju kako bi mogli da poboljšaju svoje usluge. Korisnici žele da imaju mogućnost pregleda i poređenja ocena hotela pre nego što donesu odluku o rezervaciji.

# Specifikacija zahteva korisnika

Sistem za rezervaciju hotela sa ocenjivanjem korisnika treba da zadovolji sledeće korisničke zahteve kako bi omogućio efikasno i jednostavno korišćenje za sve učesnike:

## Registracija i prijava korisnika

* Korisnici mogu da kreiraju naloge sa osnovnim informacijama (ime, email, lozinka).
* Korisnici mogu da verifikuju svoje naloge putem email-a.
* Korisnici mogu da se prijave u sistem koristeći svoje kredencijale.
* Korisnici mogu da resetuju zaboravljenu lozinku putem email-a.
* Administrator može da upravlja korisnicima (dodavanje, ažuriranje, brisanje).
* Administrator može da dodeljuje i menja prava pristupa korisnicima.

## Pretraga i rezervacija hotela

* Korisnici mogu da pretražuju hotele po lokaciji, datumu dolaska i odlaska.
* Korisnici mogu da filtriraju hotele po ceni, broju zvezdica, udaljenosti od centra grada.
* Korisnici mogu da pregledaju dostupne sobe za odabrani period.
* Korisnici mogu da pregledaju detalje o sobama (broj kreveta, veličina sobe, dostupni sadržaji).
* Korisnici mogu da rezervišu sobu online i dobiju potvrdu rezervacije putem email-a.
* Korisnici mogu da vide ukupnu cenu boravka pre potvrde rezervacije.
* Korisnici mogu da izaberu dodatne usluge (doručak, prevoz od/do aerodroma) tokom rezervacije.

## Sistem za ocenjivanje

* Korisnici mogu da ocenjuju hotele i ostavljaju komentare nakon boravka.
* Administratori mogu da pregledaju i upravljaju ocenama i komentarima.

## Upravljanje rezervacijama

* Korisnici mogu da pregledaju i otkazuju svoje rezervacije.
* Hoteli mogu da upravljaju dostupnošću soba i pregledaju rezervacije.

## Pregled informacija o hotelima

* Korisnici mogu da pregledaju detaljne informacije o hotelima (adresa, kontakt informacije, opis).
* Korisnici mogu da vide galeriju slika hotela i soba.
* Korisnici mogu da pregledaju dostupne sadržaje u hotelu (bazen, teretana, restoran).
* Korisnici mogu da dobiju informacije o lokalnim atrakcijama i mestima od interesa u blizini hotela.
* Korisnici mogu da vide pravila hotela (politika otkazivanja, prijava/odjava, kućni ljubimci).

# Opis arhitekture

Arhitektura sistema za rezervaciju hotela sa ocenjivanjem korisnika će biti bazirana na višeslojnom modelu sa sledećim komponentama:

A diagram of a network

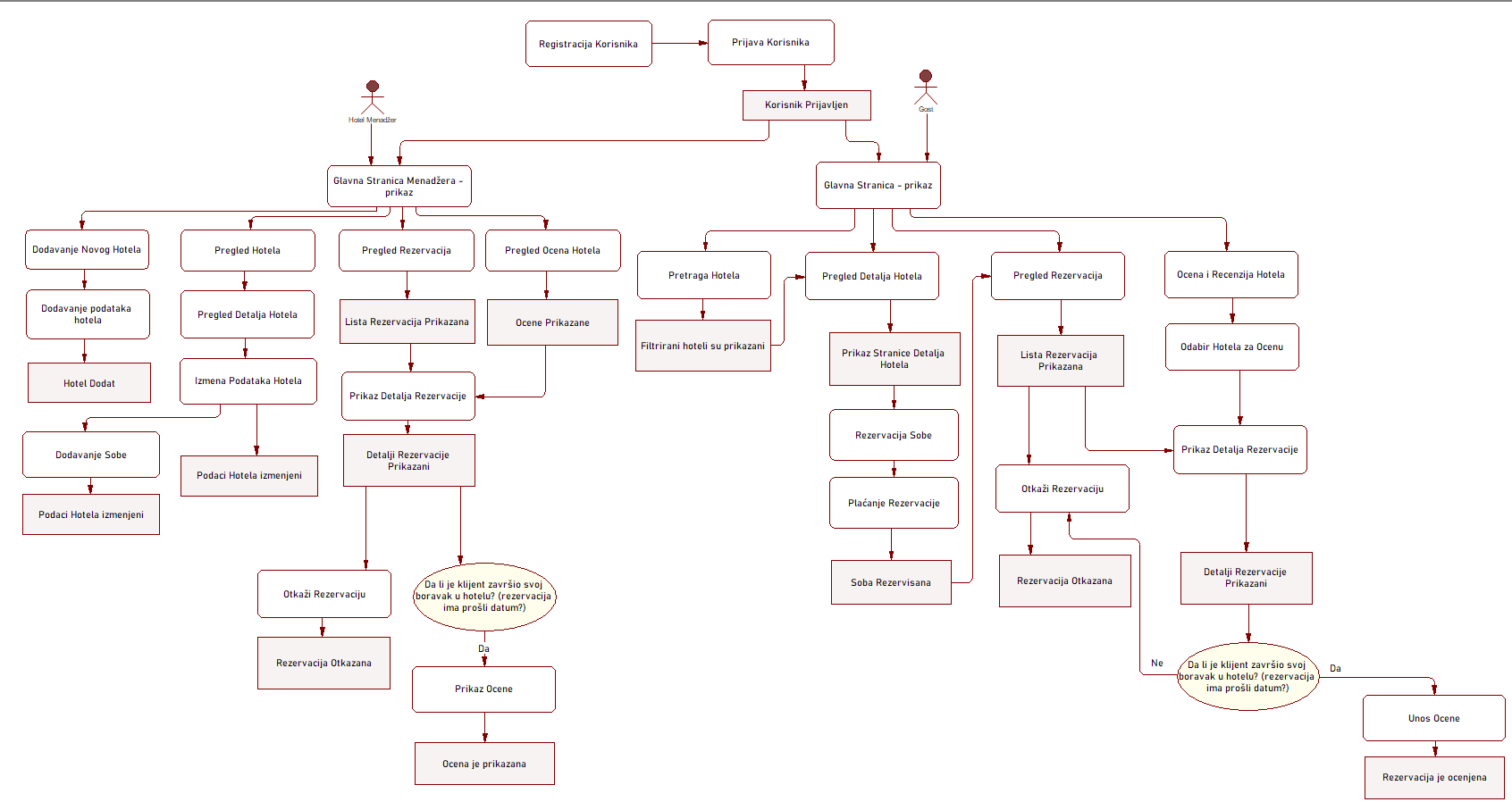
Description automatically generatedLink: <https://lucid.app/lucidchart/f4c530f1-5425-4d6e-8105-8db174c59a18/view>

* Klijent Aplikacija: Korisnici pristupaju aplikaciji preko interneta koristeći browser na svojim uređajima.
* Router/Firewall: Upravljajući mrežnim saobraćajem, ovaj komponent obezbeđuje sigurnost i distribuciju zahteva od klijenata do servera.
* Switch: Povezuje različite servere u mreži, omogućavajući efikasan prenos podataka između njih.
* Primarni Server: Ovaj server obrađuje sve poslovne logike aplikacije, uključujući registraciju korisnika, pretragu i rezervaciju hotela, upravljanje rezervacijama i ocene hotela.
* Admin Konzola: Interfejs za administratore gde mogu upravljati korisnicima, hotelima, rezervacijama i recenzijama.
* Baza Podataka: Skladišti sve podatke vezane za korisnike, hotele, rezervacije, ocene i recenzije. Baza podataka je dizajnirana da podržava brzi pristup i sigurnost podataka.

# Procesni model softvera

Detaljan procesni model za aplikaciju za rezervaciju hotela je dat ispod, kao i u projektnim fajlovima nazvanim ProcessDiagram.oom.

Procesni model uključuje razne akcije koje obavljaju hotel menadžeri i gosti, od registracije i prijave do rezervacije i ocenjivanja usluga. Ovaj sveobuhvatni model obuhvata sve neophodne korake i interakcije, pružajući jasan pregled celokupnog toka rada aplikacije.



Struktura dijagrama:

* Rezultati predstavljeni su tamnijim pravougaonicima.
* Procesi imaju zaobljene uglove.
* Tipovi korisnika označeni su malim figurama.
* Uslovi za odluku označeni su žutim elipsama.
* Tok procesa označen je strelicama koje povezuju aktivnosti, rezultate i uslove za odluku.

# Klasni dijagram aplikacije

Dijagram klasa za sistem rezervacije hotela služi kao nacrt za objektno-orijentisani dizajn sistema, olakšavajući razumevanje i implementaciju komponenti sistema i njihovih interakcija. Dijagram je takođe dostupan u projektnim datotekama pod nazivom ClassDiagram.oom.

# Podela sistema na module

## Korisnički Modul

Ovaj modul upravlja korisnicima sistema, uključujući goste, menadžere i administratore. Omogućava registraciju, prijavu i upravljanje korisnicima.

## Hotel Modul

Ovaj modul upravlja hotelima i njihovim informacijama, uključujući sobe, recenzije i ocene.



## Rezervacioni Modul

Ovaj modul upravlja rezervacijama soba, uključujući kreiranje, ažuriranje i otkazivanje rezervacija.



## Recenzioni Modul

Ovaj modul upravlja recenzijama hotela koje korisnici ostavljaju nakon boravka.



# Sekvencijalni dijagram aplikacije

## Glavni scenariji upotrebe

Glavni scenarij upotrebe za Korisnika se sastoji od ovih koraka:

1. Registracija korisnika
2. Prijava korisnika
3. Pretraga hotela
4. Rezervacija sobe
5. Plaćanje rezervacije
6. Ocenjivanje hotela

Glavni scenarij upotrebe za Hotel Managera

1. Prijava menadžera
2. Pregled podataka o hotelu
3. Dodavanje i ažuriranje informacija o sobama
4. Pregled rezervacija
5. Kontaktiranje gostiju
6. Pregled recenzija

# Prikaz testiranja

U ovom projektnom zadatku koristiće se deterministički način testiranja. Pet test slučajeva su predstavljeni u tabeli ispod.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test №** | **Naziv Testa** | **Ulazni Podaci** | **Koraci** | **Očekivani Rezultat** |
| 1 | Registracija Korisnika | Username: test\_user Password: "password" | 1. Otvoriti stranicu za registraciju. 2. Uneti username i lozinku. 3. Kliknuti na dugme "Register". | Kreiran novi korisnički nalog, prikazana poruka o uspehu. |
| 2 | Prijava Korisnika | Email: "test@test.com", Lozinka: "password123" | 1. Otvoriti stranicu za prijavu. 2. Uneti email i lozinku. 3. Kliknuti na dugme "Login". | Korisnik uspešno prijavljen, preusmeravanje na glavnu stranu. |
| 3 | Pretraga Hotela | Lokacija: "Beograd" | 1. Otvoriti stranicu za pretragu hotela. 2. Uneti lokaciju. 3. Kliknuti na dugme "Search". | Prikazana lista dostupnih hotela. |
| 4 | Rezervacija Sobe | Hotel ID: 1, Soba ID: 2, Korisnički ID: 3, Datum dolaska: "01.08.2024", Datum odlaska: "08.08.2024" | 1. Korisnik bira hotel sa ID 1. 2. Korisnik bira sobu sa ID 2. 3. Korisnik unosi podatke za rezervaciju. 4. Korisnik potvrđuje rezervaciju. | Kreirana nova rezervacija, prikazana poruka o uspešnoj rezervaciji. |
| 5 | Ocenjivanje Hotela | Hotel ID: 1, Korisnički ID: 3, Ocena: 5, Komentar: "TEST REVIEW", Datum: "2024-08-11" | 1. Korisnik bira hotel za ocenjivanje. 2. Korisnik unosi ocenu i komentar. 3. Kliknuti na dugme "Rate". | Dodata nova ocena, prikazana poruka o uspehu. |

# Integracioni testovi

|  |  |  |  |
| --- | --- | --- | --- |
| Test № | Korak | Akcija | Očekivani rezultat |
| 1 | 1 | Pozvati metodu createHotel(Hotel hotel) u HotelService | Hotel se uspešno kreira i čuva u bazi podataka |
| 2 | Pozvati metodu save(Hotel hotel) u HotelRepository | Hotel se uspešno čuva u bazi podataka |
| 2 | 1 | Pozvati metodu getHotelById(int id) u HotelService | Vraća se odgovarajući hotel za dati ID |
| 2 | Pozvati metodu findById(int id) u HotelRepository | Vraća se hotel sa traženim ID-om iz baze podataka |
| 3 | 1 | Pozvati metodu deleteHotel(int id) u HotelService | Hotel se uspešno briše iz sistema |
| 2 | Pozvati metodu deleteById(int id) u HotelRepository | Hotel se briše iz baze podataka |
| 4 | 1 | Pozvati metodu updateRoom(Room room) u RoomService | Podaci o sobi se ažuriraju u servisu |
| 2 | Pozvati metodu save(Room room) u RoomRepository | Ažurirani podaci o sobi se čuvaju u bazi podataka |
| 3 | Pozvati metodu findById(Long id) u RoomRepository | Vraća se ažurirana soba sa traženim ID-om iz baze podataka |
| 5 | 1 | Pozvati metodu getRoomsByHotelId(int hotelId) u RoomService | Vraća se lista soba za dati hotel |
| 2 | Pozvati metodu findAll() u RoomRepository | Vraća se lista svih soba iz baze podataka |

# Implementacija u Javi

Za implementaciju modula rezervacije, bilo je potrebno napraviti nekoliko dodatnih klasa kako bi aplikacija radila ispravno, iako je inicijalni zadatak zahtevao samo implementaciju jednog modula. Modul rezervacije obuhvata entitete, repozitorijume, servise i kontrolere koji omogućavaju kreiranje i upravljanje rezervacijama u sistemu.

Entiteti predstavljaju osnovne podatkovne strukture u aplikaciji. U modulu rezervacije, dva ključna entiteta su Reservation i Payment.

Repozitorijumi su odgovorni za interakciju sa bazom podataka. U modulu rezervacije, korišćeni su repozitorijumi ReservationRepository i PaymentRepository.

Servisi sadrže poslovnu logiku aplikacije. U modulu rezervacije, ključni servisi su ReservationService i PaymentService.

Kontroler upravlja HTTP zahtevima i odgovara klijentima. U modulu rezervacije, ključni kontroler je ReservationController, ali sam takođe dodala još jedan kontroler za prosleđivanje van modula – MainController.

Pored toga, korišćena je H2 baza podataka zajedno sa inicijalizatorom podataka za popunjavanje baze – DatabaseInitializer.

DatabaseInitializer.java:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42 | @Component  public class DatabaseInitializer implements CommandLineRunner {  @Autowired  private HotelRepository hotelRepository;  @Autowired  private RoomRepository roomRepository;  @Autowired  private UserRepository userRepository;  @Autowired  private ReservationRepository reservationRepository;  @Override  @Transactional  public void run(String... args) {  if (hotelRepository.count() == 0) {  Hotel hotel1 = new Hotel("Grand Budapest Hotel", "Zubrowka", 4.8f);  hotelRepository.save(hotel1);  Room room201 = new Room(hotel1, "201", "double", 120.0f);  roomRepository.save(new Room(hotel1, "101", "single", 105.0f));  roomRepository.save(new Room(hotel1, "102", "double", 110.0f));  roomRepository.save(room201);  roomRepository.save(new Room(hotel1, "202", "double", 200.0f));  Guest guest1 = new Guest("M. Gustave", "mgustave@grandbudapest.com", "1234", 60);  userRepository.save(guest1);  Reservation reservation = new Reservation();  reservation.setRoom(room201);  reservation.setGuest(guest1);  reservation.setCheckInDate(LocalDateTime.of(2024, 7, 25, 14, 0));  reservation.setCheckOutDate(LocalDateTime.of(2024, 7, 30, 11, 0));  reservationRepository.save(reservation);  }  }  } |

# Upotreba alata za otklanjanje grešaka

U ovom projektu koristim IntelliJ IDEA. IntelliJ IDEA nudi funkciju Code Inspection (Inspekcija Koda), dostupnu putem Code -> Inspect Code. Ovaj alat pomaže u analizi koda, identifikovanju potencijalnih problema i osiguravanju pridržavanja najboljih praksi. Korišćenjem ove funkcije mogu efikasno otkriti i rešiti probleme, poboljšavajući kvalitet koda i održavajući robustan softverski proizvod.

## Rezultate Inspekcije Koda

### Upozorenja u HTML datotekama

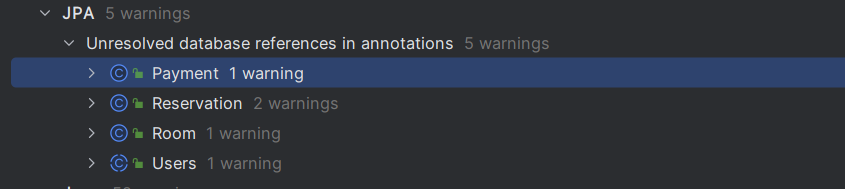
Inspekcija koda u IntelliJ IDEA identifikovala je nekoliko upozorenja u HTML datotekama u okviru projekta. Svi fajlovi imaju upozorenje vezano za pristupačnost koje kaže da nedostaje atribut lang.

A screenshot of a computer

Description automatically generated

Dodala sam atribut lang u sve HTML datoteke kako bih rešila upozorenja vezana za pristupačnost.

### Neprijavljene Reference Baze Podataka u Anotacijama



Inspekcija koda u IntelliJ IDEA identifikovala je nekoliko upozorenja u vezi sa neprijavljenim referencama baze podataka u anotacijama. Nakon detaljnog pregleda koda i testiranja aplikacije, utvrđeno je da baza podataka funkcioniše ispravno i da svi entiteti ispravno komuniciraju sa bazom podataka. Iz tog razloga, odlučeno je da se ova upozorenja ignorišu jer ne utiču na funkcionalnost aplikacije.

### Neiskorišćeni Izrazi

Tokom inspekcije, identifikovani su neiskorišćeni izrazi u nekim delovima koda.

U kontekstu kompletne aplikacije, ovi izrazi bi bili iskorišćeni kao deo sveobuhvatnog sistema. Budući da je trenutno implementiran samo jedan modul, ova upozorenja su zanemarena. Kada se svi moduli aplikacije integrišu i završe, ovi izrazi će postati aktivni i korisni, čime će ova upozorenja biti automatski rešena.

### Nepronađena Lokalno Skladištena Biblioteka za HTTP Link

Tokom analize koda, uočeno je nekoliko upozorenja vezanih za propuštenu lokalno skladištenu biblioteku za HTTP link u više HTML fajlova (kao što su edit-reservation-form.html, hotel-details.html, payment-form.html, i drugi).

A screen shot of a computer

Description automatically generated

Kako bih rešila ovaj problem, preuzela sam i lokalno skladištila ove biblioteke unutar projekta. Ovaj korak je omogućio pravilno funkcionisanje aplikacije čak i bez stalne internet konekcije.

# Integraciono Testiranje Realizovanog Modula

U PaymentRepositoryTest klasi, testiranje je fokusirano na metode PaymentRepository koristeći anotacije @DataJpaTest i @AutoConfigureTestDatabase. Ovi testovi proveravaju ispravnost metoda za pristup podacima tako što se kreiraju i čuvaju primerci Room, Guest, Reservation, i Payment klasa. Nakon što se ovi primerci sačuvaju u bazi podataka, proverava se da li se metoda findByReservationId uspešno vraća ispravan rezultat.

U ReservationRepositoryTest klasi, testirana je metoda findByGuest u ReservationRepository. Korišćenjem sličnog pristupa kao kod prethodnog testa, kreirani su i sačuvani primerci Room, Guest i Reservation. Nakon čuvanja, metoda findByGuest je testirana kako bi se proverilo da li uspešno vraća listu rezervacija za određenog gosta.

U klasi PaymentServiceTest, testovi su obuhvatili metode servisa PaymentService. Testirane su metode processPayment, calculateAmount, i findByReservationId. Testovi su proveravali stvarne operacije interakcijom sa bazom podataka i povezanim entitetima kao što su Payment, Reservation, Room, Guest, i Hotel. Korišćenjem stvarnih objekata, a ne mock objekata, testovi su osigurali da sve komponente sistema rade zajedno kako je predviđeno.

U klasi ReservationServiceTest, testirane su različite metode servisa ReservationService. Testovi su uključivali metode findRoomById, findGuestById, createReservation, findReservationsByGuestId, i cancel. Svaki test je koristio stvarne objekte za povezane repozitorijume kao što su RoomRepository, UserRepository, PaymentService, ReservationRepository, i HotelRepository, te proveravao ispravnost poslovne logike implementirane u servisu u kontekstu stvarne baze podataka.

Za ReservationControllerTest, korišćen je MockMvc okvir za simulaciju HTTP zahteva i odgovora. Ovi testovi proveravaju različite krajnje tačke kontrolera ReservationController, kao što su prikazivanje forme za rezervaciju, kreiranje nove rezervacije, prikaz svih rezervacija za gosta, uređivanje postojeće rezervacije, otkazivanje rezervacije i plaćanje rezervacije. Svaka test metoda koristi MockMvc za izvršavanje zahteva i proveru očekivanih rezultata, uključujući proveru HTTP status kodova, preusmeravanja i model atributa.

# Unit Testiranje Realizovanog Modula

Za jedinično (Unit) testiranje realizovanog modula, koristila sam JUnit, popularan alat za testiranje u Java okruženju. Fokusirala sam se na testiranje ključnih funkcionalnosti modula za rezervaciju kako bih osigurala da sve funkcije rade ispravno.

Repozitorijumske klase (repositories) u Spring Data JPA su jednostavne, interfejs bazirane komponente koje omogućavaju pristup podacima u bazi. One nasleđuju osnovne CRUD operacije iz JpaRepository interfejsa, čime se automatski generišu metode za rad sa podacima bez potrebe za dodatnim implementacijama. Iz tog razloga, testiranje ovih klasa jediničnim testovima nije neophodno.

Za klasu ReservationController, korišćeni su @Mock anotacije da bi se simulirale zavisnosti kao što su RoomRepository, ReservationService, UserRepository, ReservationRepository, PaymentService, Model, i RedirectAttributes. Ove simulacije omogućavaju izolovano testiranje logike kontrolera bez potrebe za stvarnim implementacijama ovih zavisnosti.

U klasi PaymentServiceUnitTest, testirane su metode servisa PaymentService korišćenjem mock objekata. Testovi su obuhvatili metode processPayment, calculateAmount, i findByReservationId. Korišćenjem mock objekata za PaymentRepository i ReservationRepository, testovi su simulirali stvarne operacije i proveravali očekivane rezultate. Na primer, metoda testProcessPayment simulirala je proces plaćanja i proverila da li je status rezervacije promenjen na "Paid" i da li su odgovarajući zapisi u bazi sačuvani.

U klasi ReservationServiceUnitTest, testirane su različite metode servisa ReservationService. Testovi su uključivali metode findRoomById, findGuestById, createReservation, findReservationsByGuestId, i cancel. Svaki test je koristio mock objekte za povezane repozitorijume kao što su RoomRepository, UserRepository, PaymentService, i ReservationRepository. Na primer, metoda testCreateReservation simulirala je kreiranje rezervacije i proverila da li su svi atributi rezervacije ispravno postavljeni i sačuvani.