

*Prolećni semestar, 2022/23*

*PREDMET: SE211 Konstruisanje softvera*

Projektni zadatak

Ime i prezime: **Uros Mirkovic**

Broj indeksa: **5078**

Datum izrade: **16.06.2023.**

Sadržaj

[Apstrakt 3](#_heading=h.gjdgxs)

[Opis aplikacije 4](#_heading=h.30j0zll)

[1.1. Opis 5](#_heading=)

[1.2. Funkcionalni zahtevi 6](#_heading=)

[**2. Procesni model 8**](#_heading=)

[2.1. Kompletni klasni dijagram 9](#_heading=)

[2.2. Klasni moduli 10](#_heading=)

[2.3. Modelovanje klasnih metoda pseudokodom (PPP) 10](#_heading=h.1t3h5sf)

[2.3.1. Prva metoda 11](#_heading=)

[2.3.2. Druga metoda 11](#_heading=)

[2.3.3. Treća metoda 12](#_heading=)

[2.3.4. Četvrta metoda 12](#_heading=)

[2.3.5. Peta metoda 12](#_heading=)

[2.3.6. Šesta metoda 13](#_heading=)

[2.3.7. Sedma metoda 13](#_heading=)

[Sekvencijalni dijagram 14](#_heading=h.1ksv4uv)

[2.4. Primena alata FindBugs 18](#_heading=)

[2.5. Debagovanje aplikacije 18](#_heading=)

[Testiranje 20](#_heading=h.3j2qqm3)

[2.6. JUnit testiranje- jedinično testiranje 21](#_heading=)

[2.7. Integraciono testiranje 21](#_heading=h.4i7ojhp)

# Apstrakt

Cilj ovog projekta je kreiranje sistema online biblioteke. Sistem će imati korisnika, zaposlenog I admina. Korisnik će moći da pregleda sve knjige, da vidi odabranu knjigu, da se prijavi kao redovan član I da kupi knjigu koju je odabrao. Zaposleni će imati mogućnost dodavanja novih knjiga I uvid u sve knjige. Admin će moći da briše, dodaje, I vidi sve zaposlene na sistemu kao I pregled transakcija korisnika. Platforma za koju bi se ova aplikacija razvila je web, najviše zbog njegove velike učestalosti na tržištu. Izrada ove apliakcije poštovala bi i sledila sve zadatke zahteve koji moraju da se ispune u sklopu ovog projektnog zadatka.

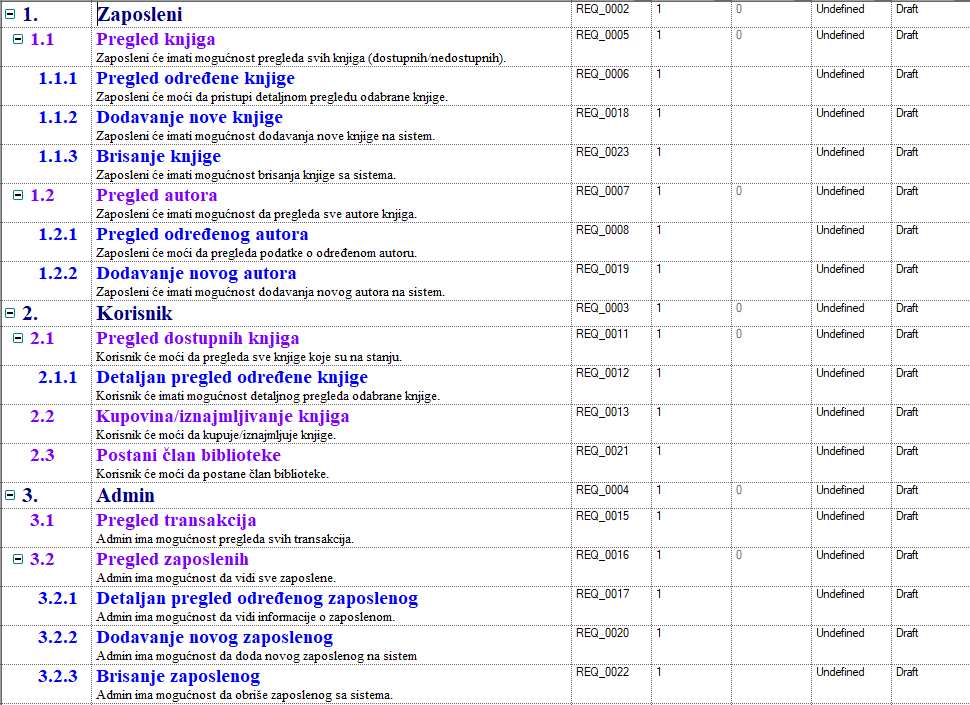
# Opis aplikacije

## ***Opis***

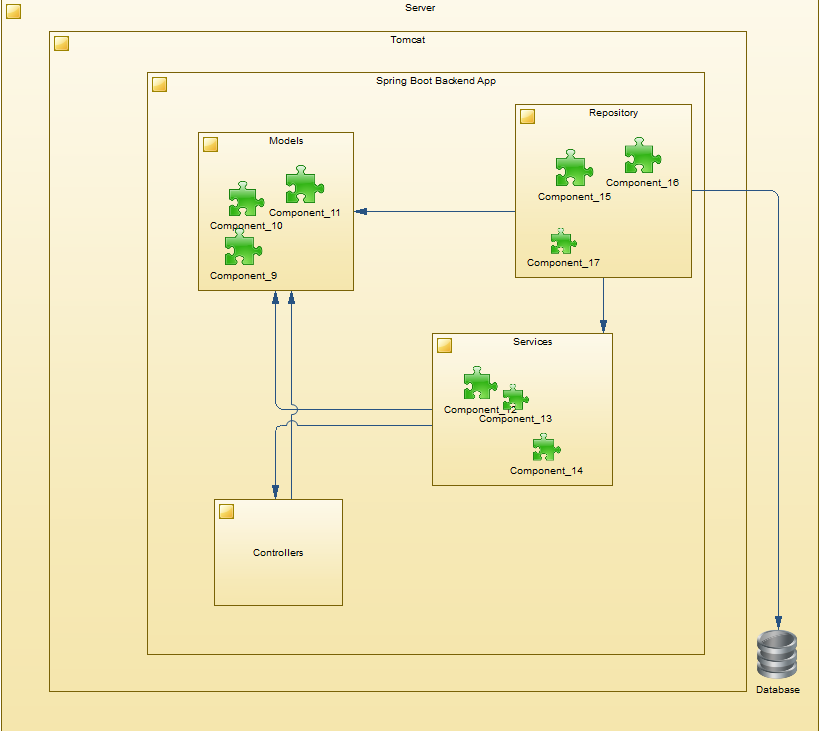
Naziv aplikacije je “OnlineLibrary”. Omogućila bi vam kupovinu velikog izbora knjiga uz mogućnost učlanjenja u klub čitalaca s kojim biste imali određene pogodnosti. U mogućnosti ste da pretražujete, sortirate knjige po raznim parametrima I da odaberete one koje volite.

Korisnički zahtevi

## ***Funkcionalni zahtevi***

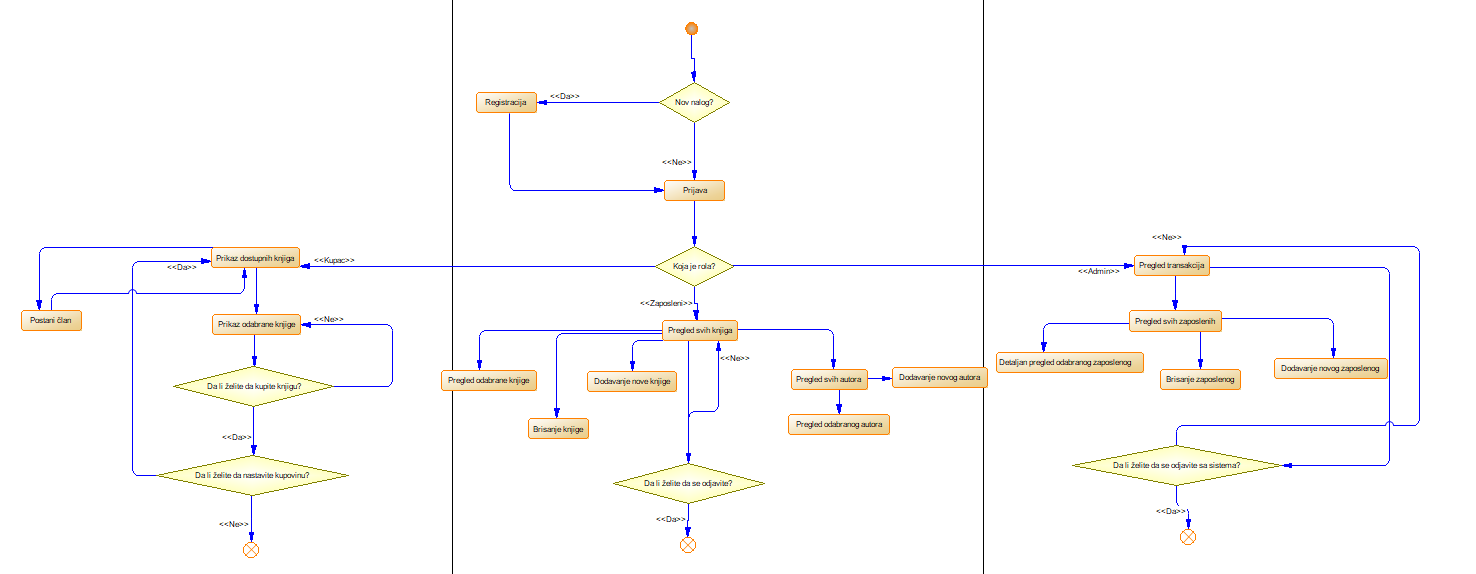


**Slika 1. Prikaz funkcionalnih zahteva**Arhitektura

****

**Slika 3. Prikaz arhitekture sistema**

# Procesni model

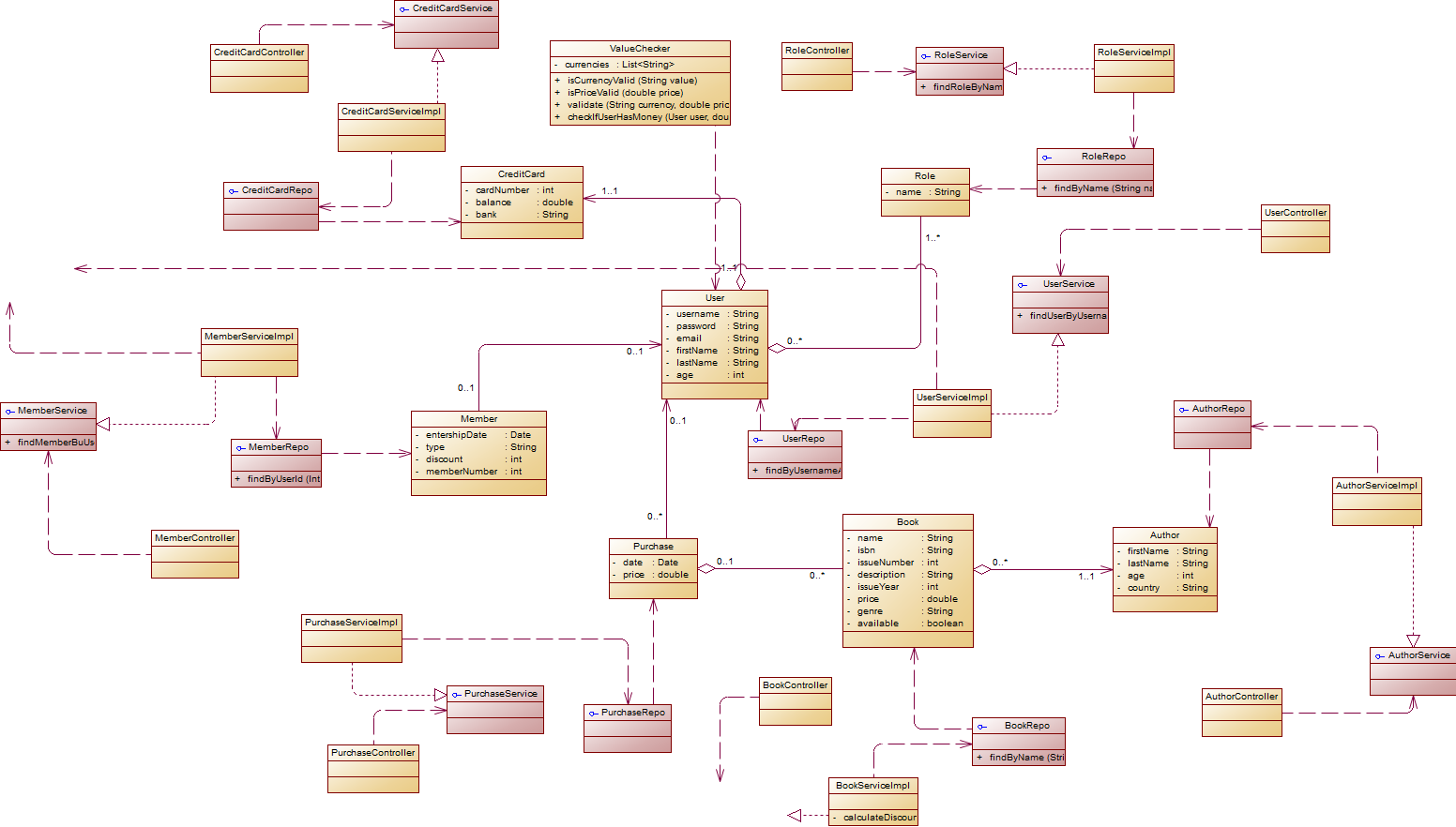


**Slika 4. Prikaz procesnog modela**

Kao prva stavka procesnog modela je odabir korisnika da li želi da se uloguje (ukoliko već poseduje nalog) ili da se registruje ukoliko ne poseduje nalog. Zatim na osnovu role koju korisnik ima sistem mu daje različite permisije.

1. Ukoliko je rola KORISNIK mogućnosti se baziraju na kupovini knjiga i pregledu određene knjige ili svih knjiga. Postoji mogućnost i učlanjena u klub čitalaca koji imaju određene povoljnosti prilikom kupovine.
2. Ukoliko je rola ZAPOSLENI zaposleni ima mogućnost pregleda autora knjiga, dodavanje i brisanje autora takođe i dodavanje i brisanje knjiga kao i njihov pregled.
3. Ukoliko je rola ADMIN adminu se nudi pregled transakcija kao i rukovođenje zaposlenima (brisanje, dodavanje, pregled).Klasni dijagram

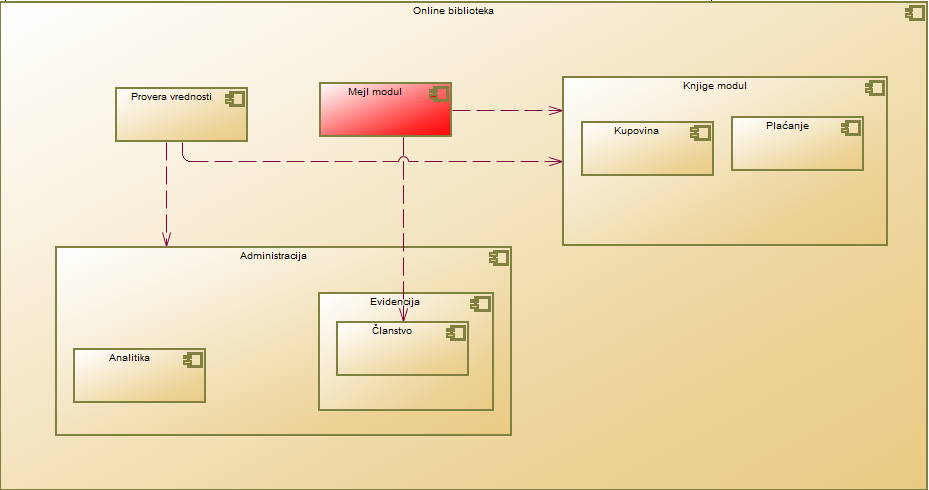
## ***Kompletni klasni dijagram***



**Slika 5. Klasni dijagram**

Klasni dijagram koji predstavlja potpunu strukturu aplikacije. Na ovom dijagramu može se videti dizajn svih neophodnih klasa kao i dizajn metoda koju svaka od klasa poseduje.

## ***Klasni moduli***



Aplikacija je podeljena na 4 modula od koga u 2 modula imamo podmodule. Moduli su Mejl, Provera vrednosti, Administracija(Analitika, Evidencija(Članstvo)), Knjiga(Plaćanje, Kupovina).

## ***Modelovanje klasnih metoda pseudokodom (PPP)***

### **Prva metoda**

*Funkcija: isCurrencyValid(value)*

*Ulaz: String value*

*Izlaz: Boolean*

*BEGIN*

*IF value = null*

*Throw exception*

*ENDIF*

*IF currencies not contains value*

*Throw exception*

*ENDIF*

*RETURN true;*

*END*

### **Druga metoda**

Funkcija: isPriceValid(price)

Ulaz: Double price

Izlaz: Boolean

*BEGIN*

*IF price = null*

*Throw exception*

*ENDIF*

*IF price < 0*

*Throw exception*

*ENDIF*

*RETURN true;*

*END*

### **Treća metoda**

Funkcija: validate(currency, price)

Ulaz: String currency, Double price

Izlaz: Boolean

BEGIN

RETURN isPriceValid(price) (2.3.2 Druga metoda) AND isCurrencyValid(currency) (2.3.1 Prva metoda)

END

### **Četvrta metoda**

Funkcija: checkIfHasMoney(user, money)

Ulaz: User user, Double money

Izlaz Boolean

BEGIN

IF user.getCreditCard() = null

Throw exception

ENDIF

IF user.getCreditCard().getBalance() < 0

Throw exception

ENDIF

RETURN user.getCreditCard().getBalance() >= money;

END

### **Peta metoda**

Funkcija: findAlllAvailableBooks()

Ulaz:

Izlaz: List<Book

BEGIN

RETURN findAllByAvailableIsTrue();

END

### **Šesta metoda**

Funkcija: findByUserId(userId)

Ulaz: Integer userId

Izlaz: Member

BEGIN

RETURN findByUserId(userId);

END

### **Sedma metoda**

Funkcija: calculateDiscount(price, percent)

Ulaz: Double price, Double percent

Izlaz: Double – popust

BEGIN

DISCOUNT := price \* (percent / 100);

RETURN price – discount;

END

* + 1. **Osma metoda**

Funkcija: findUserByUsernameAndPassword(username, password)

Ulaz: STRING username, STRING password

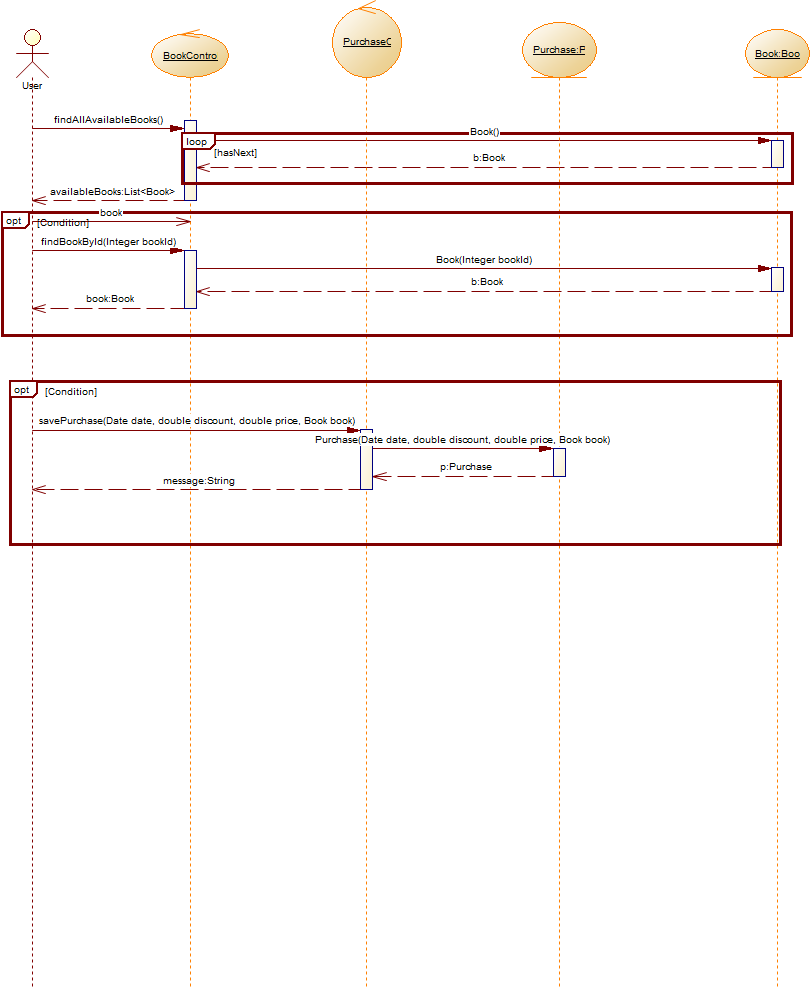
Izlaz: USER user

BEGIN

RETURN findByUsernameAndPassword(username, password);

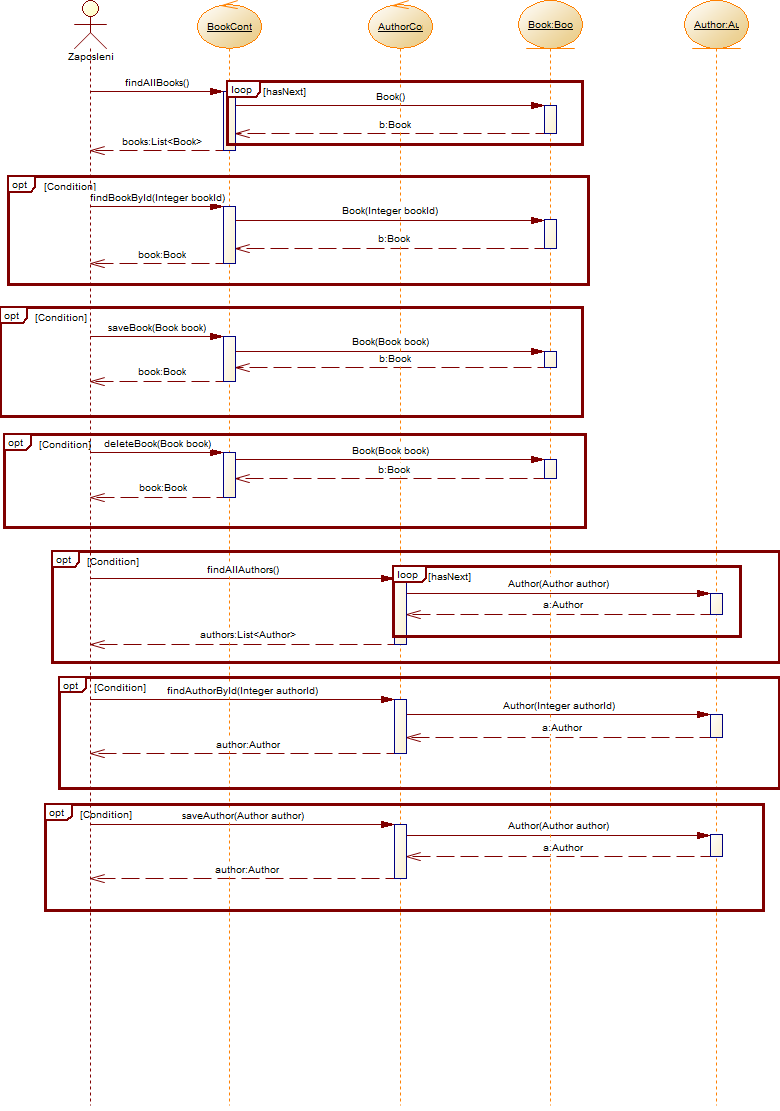
END

# Sekvencijalni dijagram



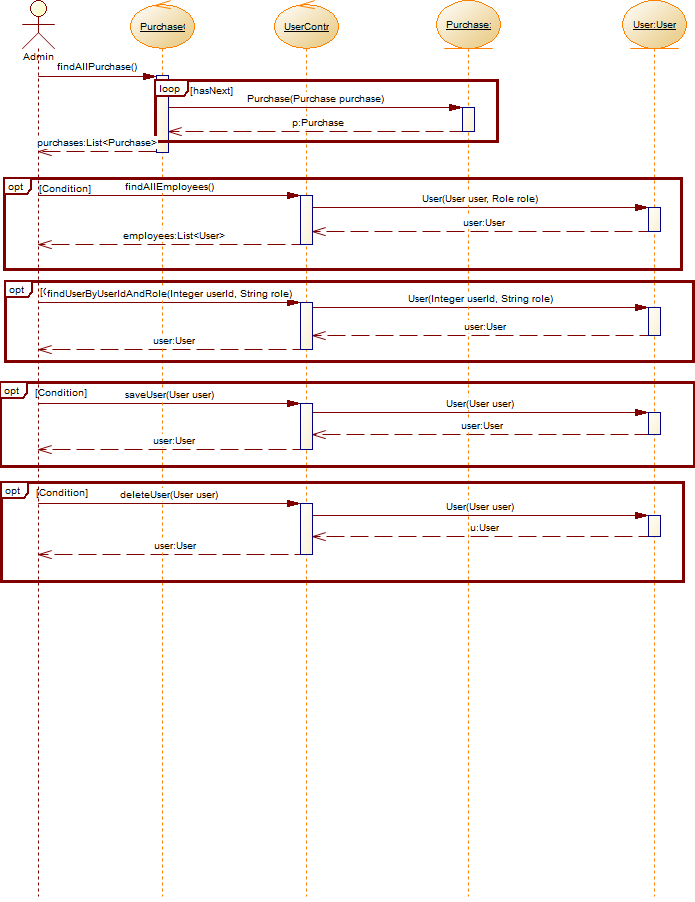
**Slika 11. Prikaz sekvencijalnog dijagrama - USER**

Ovaj sekvencijalni dijagram predstavlja pozivanje metoda za prikaz knjiga I opcionalno user može da odabere prikaz određene knjige ili da obavi kupovinu.



**Slika 12. Prikaz sekvencijalnog dijagrama - ZAPOSLENI**

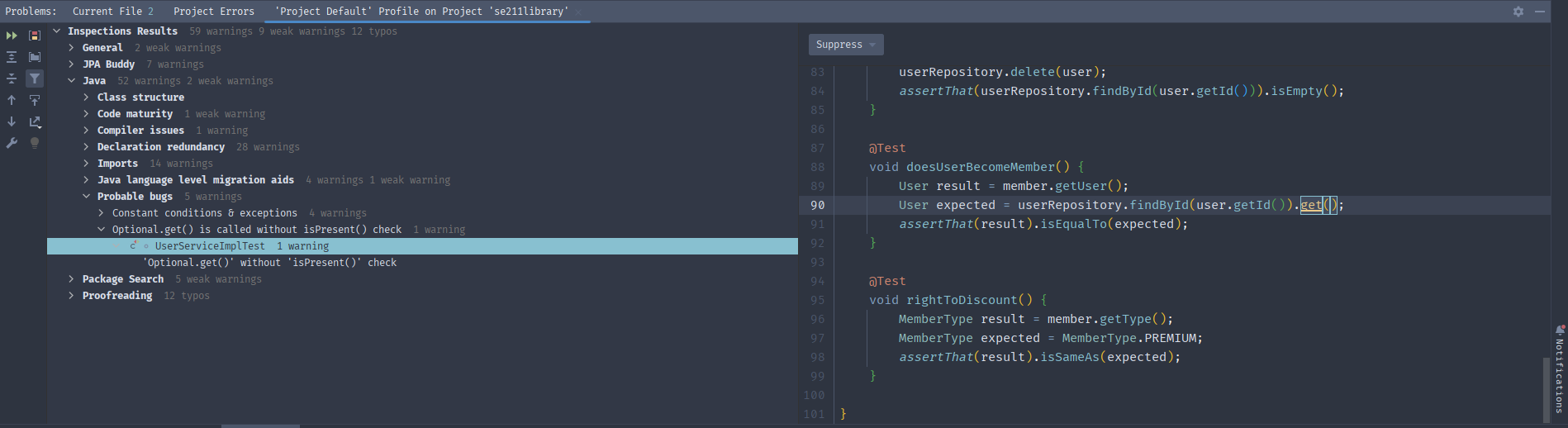
Ovaj sekvencijalni dijagram predstavlja pozivanje metoda za prikaz, dodavanje, brisanje I pregleda određenih knjiga. Takođe opcionalno moguće su sve te akcije I sa autorima knjiga.



**Slika 13. Prikaz sekvencijalnog dijagrama - ADMIN**

Ovaj sekvencijalni dijagram predstavlja pozivanje metoda za pregled transakcija I rukovanje zaposlenima.

### **Primena alata FindBugs**

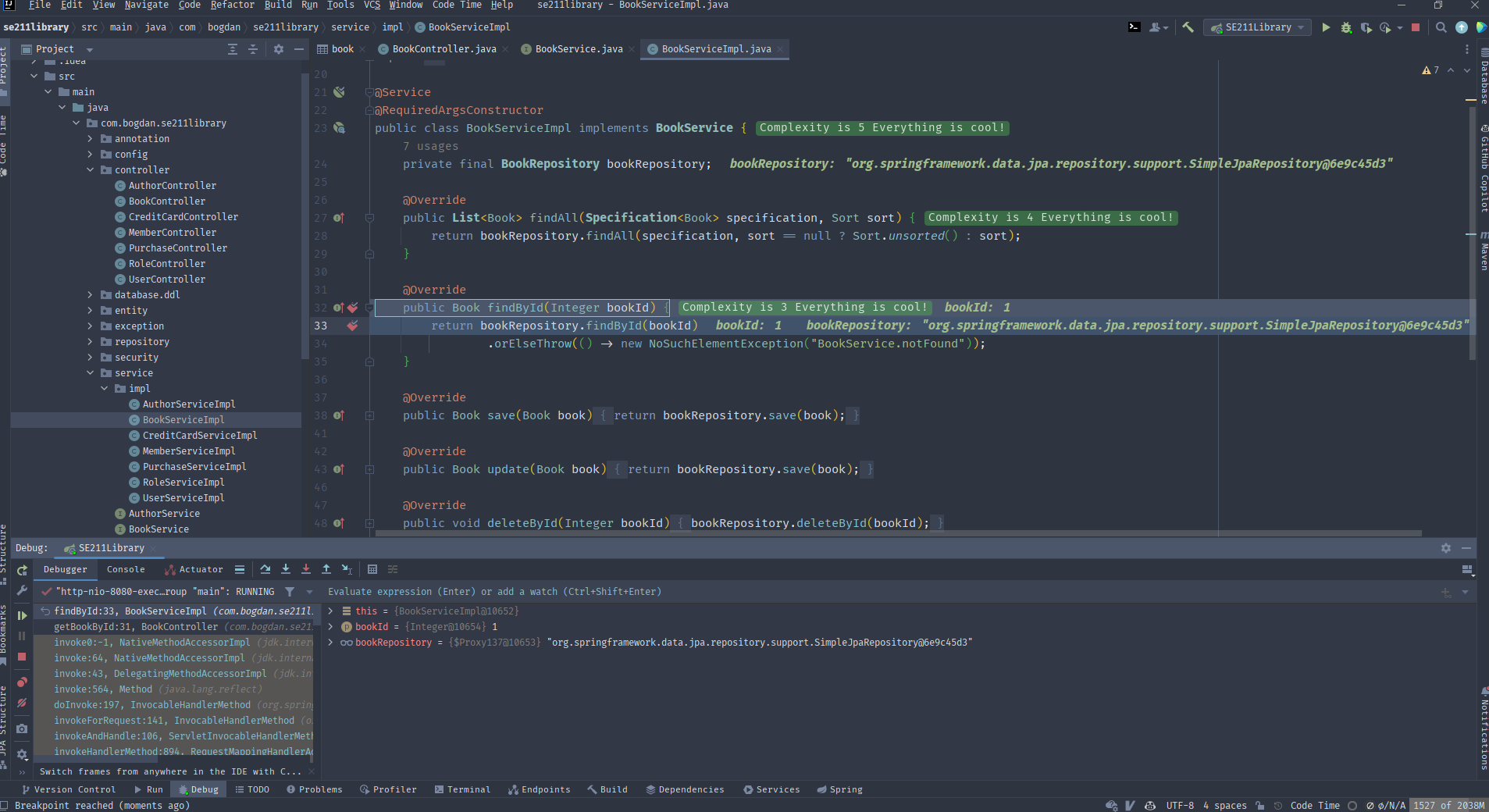
****

**Slika 22. Prikaz rezultata FindBugs alata**

Na slici možemo videti rezultate otklananja grešaka u programskom kodu putem FindBugs alata.

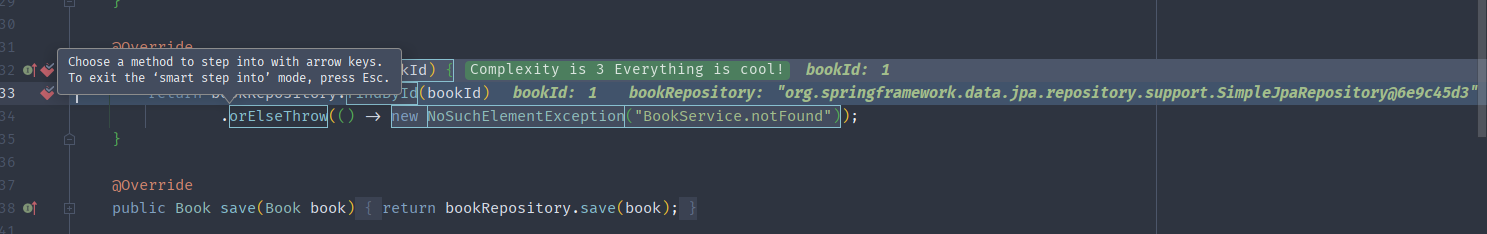
## ***Debagovanje aplikacije***

Debagovanje će biti izvršeno nad klasi Book, odnosno na metodi za pronalaženje knjige po ID-u.

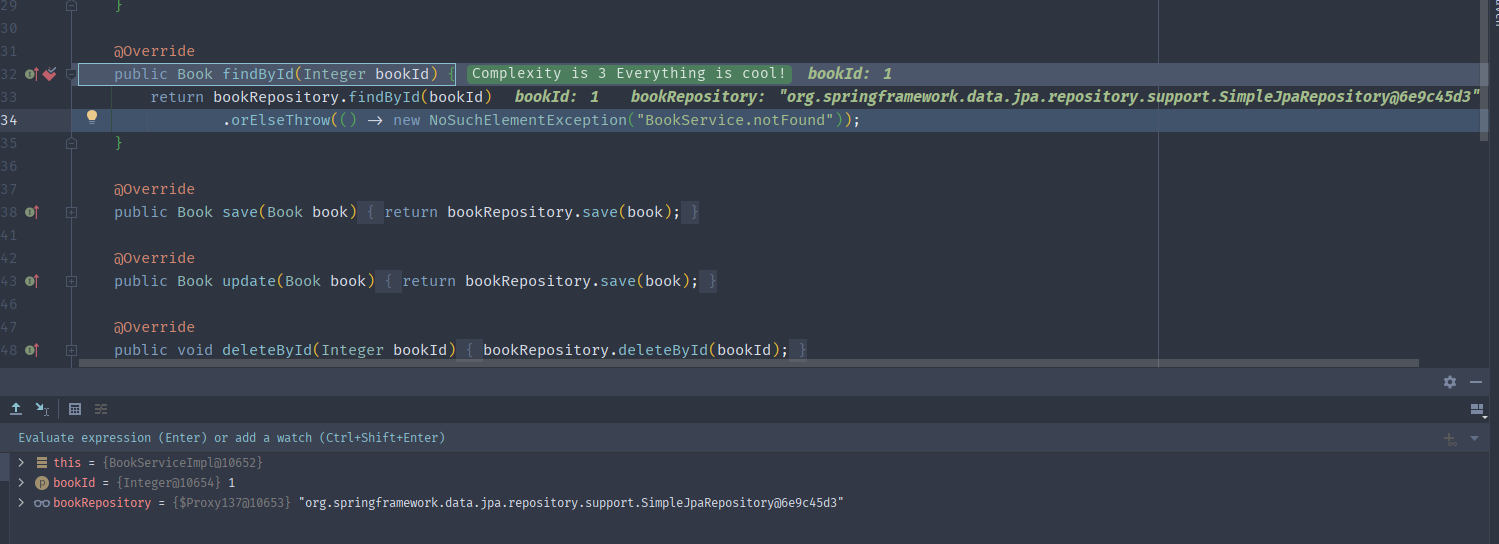


**Slika 23. Prikaz postavljanja Breakpoint-a**

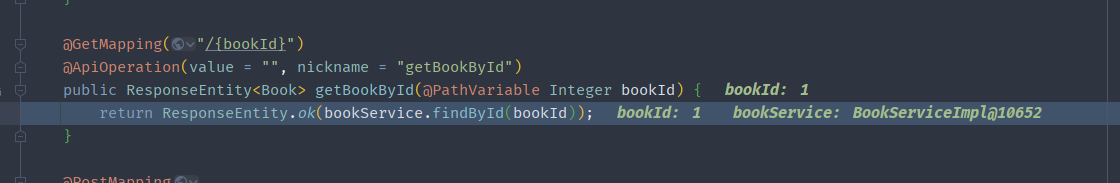
Na ovoj slici možemo videti sam početak debagovanja, odnosno postavljanja Breakpoint-a, tj. mesto odakle želimo da počnemo debagovanje.



**Slika 24. Prikaz pokretanja debagovanja**



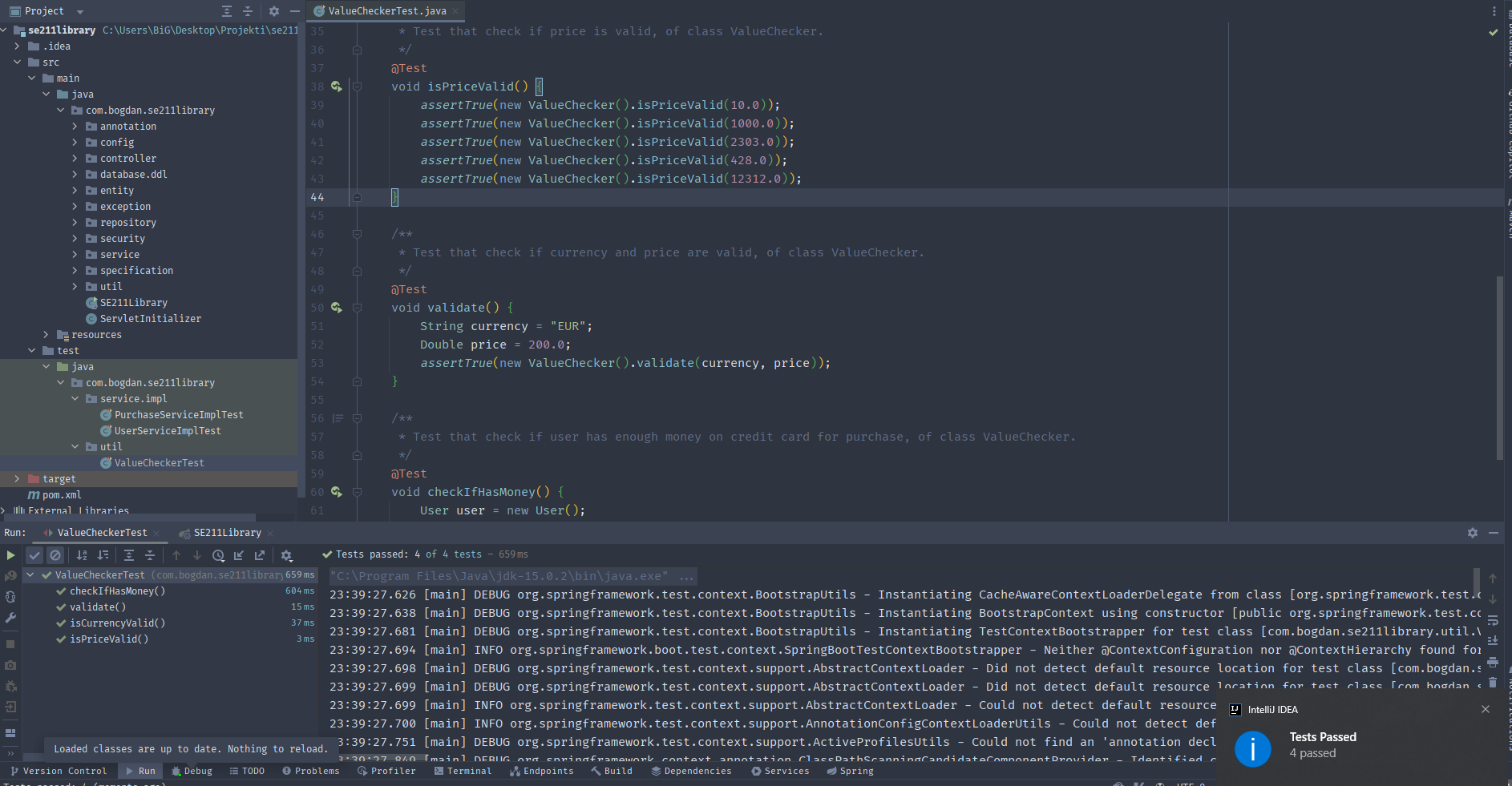
**Slika 25. Prikaz nakon pokretanja debagovanja**



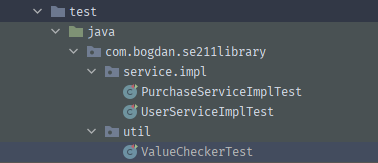
**Slika 26. Prikaz nakon opcije „Step over“**

# Testiranje

## ***JUnit testiranje- jedinično testiranje***



**Slika 29. Prikaz testova**

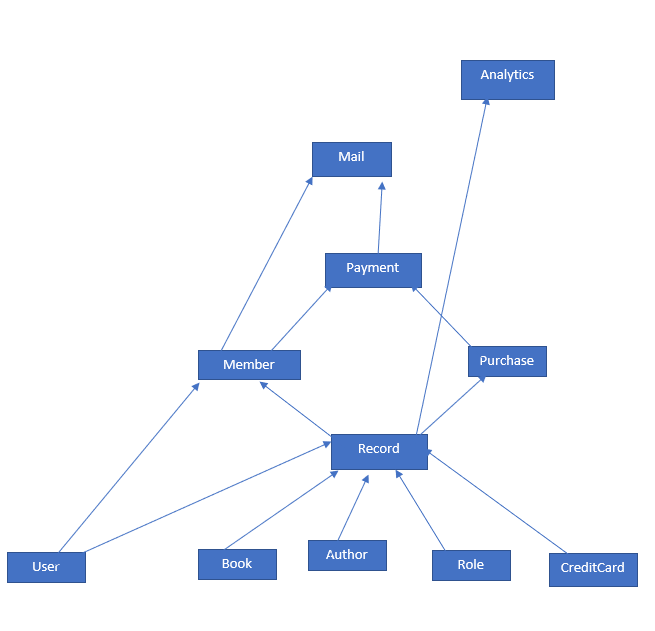


**Slika 30. Prikaz kreiranja testova**

U ovim testovima porede se dobijeni rezultat sa očekivanim, i kao što možemo videti metoda je uspešno završila testove za sve moguće vrednosti i samim tim smo ustanovili da je metoda u potpunosti ispravna.

## ***Integraciono testiranje***

Tehnika za integraciono korišćenje koje je korišćena je „Bottom-Up“ .



Ovaj pristup prvo testira najniže delove softvera (jedinice i module) i kreće se ka vrhu integracijom manjih celina u veće. Takođe se naziva i klasični pristup zato što je prirodno krenuti od manjih ka većim celinama. Za najmanje celine je potrebno pisati i “stub”-ove i “driver”-e. Teorijski ovo je dobar pristup, ali se praksi u teško implementira. Dobra strana ovog pristupa je ta sto se testira svaka jedinica ili komponenta i samo ako je ispravna može učestvovati u integraciji.