

Универзитет у Београду  
Факултет организационих наука  
Катедра за софтверско инжењерство



Семинарски рад из предмета  
**Пројектовање софтвера**

**Софтверски систем за управљање радом  
интернет кафића у Јава окружењу**

Ментор: проф. др Синиша Влајић

Студент: **Огњен Павловић**

Број индекса: **2021/0137**

Београд, 2024.

## Садржај

1. Увод .....	3
2. Прикупљање корисничких захтева .....	4
2.1 Вербални опис .....	4
2.2 Случајеви коришћења .....	6
3. Анализа .....	17
3.1 Понашање софтверског система - Одређивање системских операција на основу сценарија случаја коришћења .....	17
3.2 Понашање софтверског система - Секвенци дијаграми случаја коришћења .....	20
ДС1: Дијаграм секвенци случаја коришћења – Креирај рачун .....	20
ДС2: Дијаграм секвенци случаја коришћења – Претражи рачун .....	23
ДС3: Дијаграм секвенци случаја коришћења – Промени рачун .....	26
3.3 Понашање софтверског система - Дефинисање уговора о системским операцијама .....	32
3.4 Структура софтверског система – Концептуални (доменски) модел .....	34
3.5 Структура софтверског система – Релациони модел .....	35
3.6 Табела структурних и вредносних ограничења релационог модела .....	36
4. Пројектовање .....	39
4.1 Пројектовање корисничког интерфејса .....	39
4.2 Пројектовање апликационе логике .....	69
4.3 Пројектовање складишта података .....	85
5. Имплементација .....	88
6. Тестирање .....	91
LoginMusterijaSOTest .....	92
LogoutMusterijaSOTest .....	94
UbaciRacunSOTest .....	96
VratiSveSOTest .....	98
7. Закључак .....	100
Литература .....	101

## 1. Увод

У овом семинарском раду обрађена је израда клијентско-серверске апликације намењене управљању пословањем **интернет кафића и гејминг играоница**. Главни циљ апликације јесте да омогући једноставно и интуитивно коришћење, при чему је посебан акценат стављен на „*user-friendly*“ концепт; систем је осмишљен тако да свако може лако да га користи, без обзира на техничко предзнање. Као посебна карактеристика издваја се функционалност оверлеја који се налази на корисничким рачунарима, а који обезбеђује да клијенти могу да користе услуге само у оквиру времена које имају на свом налогу. На тај начин омогућава се транспарентна и поуздана контрола времена сесија, чиме се директно подржава ефикасно пословање.

Са техничке стране, архитектура апликације ослања се на **апстраховану логику** која је развијена тако да систем буде **лако одржив и једноставан за проширивање**. Овако дефинисана структура омогућава да се решење, уз минималне измене, примени и у другим сродним типовима пословања који захтевају управљање корисничким налозима, услугама и ограниченим временом коришћења. Овим приступом постиже се висок степен флексибилности и поузданости, што ову апликацију чини практичним и одрживим решењем за сваку организацију која тежи модернизацији и оптимизацији рада.

Поред овог уводног поглавља, дата су и следећа поглавља која се надовезују:

2. **Прикупљање корисничких захтева** које подразумева вербални опис и извођење случајева коришћења.
3. **Анализа софтверског система**, која даје целовит увид у то шта систем представља, која треба да буде његова улога, и како он ту улогу треба да врши.
4. **Пројектовање информационог система**, које говори о томе на који начин је пројектована структура и изглед корисничког интерфејса, апликационе логике и складишта података.
5. **Имплементација**, која говори о конкретним начинима и методама којима су имплементирани концепти о којима су говорила претходна поглавља.
6. **Тестирање** које обухвата објашњење методологије спровођења тестова помоћу оквира `jUnit`.
7. **Закључак** који сумира све до тада наведено, обухвата најважније концепте и даје финалну поруку о томе како је сам рад написан, уз осврт аутора на изазове са којима се сусрео током писања, али и на нова знања која је стекао.

На самом крају побројан је списак литературе која је била употребљивана приликом израде семинарског рада.

## 2. Прикупљање корисничких захтева

### 2.1 Вербални опис

Потребно је направити **Софтверски систем за управљање радом интернет кафића / гејминг играонице у Јава окружењу**. Софтверски систем, односно његова пословна логика (у општем смислу), састоји се из следећих апстрактних концепата:

а) **пружалац услуге**

б) **прималац услуге**

ц) **документ** који описује процес пружања услуге

д) **шифарници** у којима се налазе подаци о конкретним концептима који се користе у процесу пружања услуге, а који нису пружалац услуге, прималац услуге или документ који описује процес пружања услуге.

У наведеном софтверском систему **пружалац услуге** је **Продавац**, **прималац услуге** је **Муштерија**, **документ** који описује процес пружања услуге је **Рачун**. **Шифарници** су **Категорија муштерије**, **ТерминДежурства**, **Услуга**.

**Конкретни концепти** су између себе повезани на следећи начин:

**Преко рачуна** је могуће продати више ставки. **Рачун** је повезан са једним продавцем и једном муштеријом. **Муштерија** је везан за једну категорију муштерије. **Продавац** може бити везан за више термина дежурстава, док **термин дежурства** може бити везан за више продаваца.

При пријављивању на софтверски систем потребно је обезбедити **аутентификацију** (преко корисничког имена и шифре) корисника софтверског система.

Потребно је обезбедити следеће функционалности за наведене конкретне концепте:

Редни број концепта	Концепт	Функционалности
1.	Рачун	Убаци, Претражи, Промени
2.	Муштерија	Креирај, Претражи, Промени, Обриши, Пријави
3.	Продавац	Пријави, Креирај, Претражи, Промени, Обриши
4.	Услуга	Креирај, Претражи, Промени, Обриши
5.	Категорија муштерије	Креирај, Претражи, Промени, Обриши
6.	ТерминДежурства	Убаци, Претражи, Промени, Обриши
7.	Ставка рачуна	Креирај, Обриши, Претражи, Промени, Убаци
8.	Запослени-Термин дежурства	

Табела 1: Повезивање концепата и функционалности

Функционалностима софтверског система се приступа преко главног менија који има следећу структуру:

1. Документи
  - 1.1 Рачун
2. Пружалац услуге
  - 2.1 Продавац
3. Прималац услуге
  - 3.1 Муштерија
4. Шифарници
  - 4.1 Категорија муштерије,
  - 4.2 ТерминДежурства,
  - 4.3 Услуга
5. Подешавања софтверског система
6. О програму

## 2.2 Случајеви коришћења

На основу наведених функционалности концепата уочени су следећи случајеви коришћења:

Шифра случаја коришћења	Назив случаја коришћења	Предуслови С.К. (листе)	Критеријуми претраживања се односе на:
SK1	СК1- Убази рачун	Учитане су листе: а) Продавац б) Муштерија с) Услуга	
SK2	СК2- Претражи рачун		а) Рачун б) Продавац с) Муштерија д) Услуга
SK3	СК3- Промени рачун	Учитане су листе: а) Продавац б) Муштерија с) Услуга	а) Рачун б) Продавац с) Муштерија д) Услуга
SK4	СК4- Креирај муштерија	Учитане су листе: а) Категорија муштерије	
SK5	СК5- Претражи муштерија		а) Муштерија б) Категорија муштерије
SK6	СК6- Промени муштерија	Учитане су листе: а) Категорија муштерије	а) Муштерија б) Категорија муштерије
SK7	СК7- Обриши муштерија		а) Муштерија б) Категорија муштерије
SK8	СК8- Пријави муштерија		
SK9	СК9- Пријави продавац		
SK10	СК10- Креирај продавац	Учитане су листе: а) ТерминДежурства	
SK11	СК11- Претражи продавац		а) Продавац б) ТерминДежурства
SK12	СК12- Промени продавац	Учитане су листе: а) ТерминДежурства	а) Продавац б) ТерминДежурства
SK13	СК13- Обриши продавац		а) Продавац б) ТерминДежурства
SK14	СК14- Креирај услуга		
SK15	СК15- Претражи услуга		а) Услуга
SK16	СК16- Промени услуга		а) Услуга
SK17	СК17- Обриши услуга		а) Услуга
SK18	СК18- Креирај категорија муштерије		
SK19	СК19- Претражи категорија муштерије		а) Категорија муштерије
SK20	СК20- Промени категорија муштерије		а) Категорија муштерије
SK21	СК21- Обриши категорија муштерије		а) Категорија муштерије
SK22	СК22- Убази терминдежурства		
SK23	СК23- Претражи терминдежурства		а) ТерминДежурства
SK24	СК24- Промени терминдежурства		а) ТерминДежурства
SK25	СК25- Обриши терминдежурства		а) ТерминДежурства
SK26	СК26- Креирај ставка		

	рачуна		
SK27	СК27- Обриши ставка рачуна		
SK28	СК28- Претражи ставка рачуна		
SK29	СК29- Промени ставка рачуна		
SK30	СК30- Убази ставка рачуна		

Табела 2: Случајеви коришћења софтверског система

За следеће случајеве коришћења ћемо дати детаљан опис:

СК1- Убази рачун
СК2- Претражи рачун
СК3- Промени рачун
СК4- Убази муштерија
СК6- Промени муштерија
СК7- Обриши муштерија
СК8- Пријави муштерија
СК9- Пријави продавац
СК22- Убази термин дежурства

Табела 3: Случајеви коришћења за које ће бити дат детаљан опис

## СК1- Убацн рачун

### Назив СК

Убацн рачун

### Акторн СК

Продавац

### Учесннкн СК

Продавац, корисннкн интерфејс (кнјентскн програм) и систем (серверскн програм)

**Предуслови:** Корисннкн интерфејс (кнјентскн програм) и систем (серверскн програм) су покренути. Продавац је пријављен под својом шифром. Систем приказује форму за рад са рачуном. *Учитане су листе: а) Ставка рачуна, б) Услуга*

### Основнн сценарно СК:

1. Продавац уноси податке о рачуну. (АПУСО)
2. Продавац контролише да ли је коректно унео податке о рачуну. (АНСО)
3. Продавац позива систем да запамти податке о рачуну. (АПСО)
4. Систем памти податке о рачуну. (СО)
5. Систем приказује продавцу рачун и поруку: “Систем је запамтио рачун.” (ИА)

### Алтернативна сценарија:

5.1 Уколико систем не може да запамти податке о рачуну он приказује продавцу поруку: “Систем не може да запамти рачун”. (ИА)



## СК2- Претражи рачун

### Назив СК

Претражи рачун

### Актори СК

Продавац

### Учесници СК

Продавац, кориснички интерфејс (клијентски програм) и систем (серверски програм)

**Предуслови:** Кориснички интерфејс (клијентски програм) и систем (серверски програм) су покренути. Продавац је пријављен под својом шифром. Кориснички интерфејс приказује форму за рад са рачуном. На наведеној екранској форми су дефинисани критеријуми, који се односе на: а) Рачун б) Продавац с) Муштерија д) Услуга, који ће да врате листу рачуна.

### Основни сценарио СК:

1. Продавац бира критеријуме на основу којих претражује рачуне. (АПУСО)
2. Продавац позива систем да нађе рачуне по задатим критеријумима. (АПСО)
3. Систем тражи рачуне по задатим критеријумима. (СО)
4. Систем приказује продавцу рачуне и поруку: "Систем је нашао рачуне по задатим критеријумима". (ИА)
5. Продавац бира рачун. (АПУСО)
6. Продавац позива систем да нађе рачун. (АПСО)
7. Систем тражи рачун. (СО)
8. Систем приказује продавцу рачун и поруку: "Систем је нашао рачун". (ИА)

### Алтернативна сценарија:

4.1 Уколико систем не може да нађе рачуне он приказује продавцу празну табелу. Прекида се извршење сценарија. (ИА)

8.1 Уколико систем не може да нађе рачун он приказује продавцу поруку: "Изаберите неки рачун".(ИА)

### СКЗ- Промени рачун

#### Назив СК

Промени рачун

#### Актори СК

Продавац

#### Учесници СК

Продавац, кориснички интерфејс (клијентски програм) и систем (серверски програм)

**Предуслови:** Кориснички интерфејс (клијентски програм) и систем (серверски програм) су покренути. Продавац је пријављен под својом шифром. Кориснички интерфејс приказује форму за рад са рачуном. На наведеној екранској форми су дефинисани критеријуми, који се односе на: а) Рачун б) Продавац с) Муштерија д) Услуга, који ће да врате листу рачуна. Учитане су листе: а) Продавац б) Муштерија с) Услуга

#### Основни сценарио СК:

1. Продавац бира критеријуме на основу којих претражује рачуне. (АПУСО)
2. Продавац позива систем да нађе рачуне по задатим критеријумима. (АПСО)
3. Систем тражи рачуне по задатим критеријумима. (СО)
4. Систем приказује продавцу рачуне. (ИА)
5. Продавац бира рачун. (АПУСО)
6. Продавац позива систем да нађе рачун. (АПСО)
7. Систем тражи рачун. (СО)
8. Систем приказује продавцу рачун. (ИА)
9. Продавац уноси (мења) податке о рачуну. (АПУСО)
10. Продавац контролише да ли је коректно унео податке о рачуну. (АНСО)
11. Продавац позива систем да запамти податке о рачуну. (АПСО)
12. Систем памти податке о рачуну. (СО)
13. Систем приказује продавцу рачун и поруку: "Систем је запамтио рачун." (ИА)

#### Алтернативна сценарија:

- 4.1 Уколико систем не може да нађе рачуне он приказује продавцу празну табелу. Прекида се извршење сценарија. (ИА)
- 8.1 Уколико систем не може да нађе рачун он приказује продавцу поруку: "Изаберите неки рачун".(ИА)
- 13.1 Уколико систем не може да запамти податке о рачуну он приказује продавцу поруку: "Систем не може да запамти рачун". (ИА)

## СК4- Убаци муштерија

### Назив СК

Убаци муштерија

### Актори СК

Муштерија

### Учесници СК

Муштерија, кориснички интерфејс (клијентски програм) и систем (серверски програм)

**Предуслови:** Кориснички интерфејс (клијентски програм) и систем (серверски програм) су покренути. Систем приказује форму за рад са муштеријом. *Учитане су листе: а) Муштерија.*

### Основни сценарио СК:

1. Муштерија уноси податке о муштерији. (АПУСО)
2. Муштерија контролише да ли је коректно унео податке о муштерији. (АНСО)
3. Муштерија позива систем да запамти податке о муштерији. (АПСО)
4. Систем памти податке о муштерији. (СО)
5. Систем приказује муштерији поруку: "Систем је запамтио муштерију." (ИА)

### Алтернативна сценарија:

5.1 Уколико систем не може да запамти податке о муштерији он приказује муштерији поруку о детаљнијем разлогу неуспешног убацивања. (ИА)

## СК6- Промени муштерија

### Назив СК

Промени муштерија

### Актори СК

Продавац

### Учесници СК

Продавац, кориснички интерфејс (клијентски програм) и систем (серверски програм)

**Предуслови:** Кориснички интерфејс (клијентски програм) и систем (серверски програм) су покренути. Продавац је пријављен под својом шифром. Кориснички интерфејс приказује форму за рад са муштеријом. На наведеној екранској форми су дефинисани критеријуми, који се односе на: а) Муштерија б) Категорија муштерије, који ће да врате листу муштерија. Учитане су листе: а) Категорија муштерије

### Основни сценарио СК:

1. Продавац бира муштерију. (АПУСО)
2. Продавац позива систем да нађе муштерију. (АПСО)
3. Систем тражи муштерију. (СО)
4. Систем приказује продавцу муштерију. (ИА)
5. Продавац уноси (мења) податке о муштерији. (АПУСО)
6. Продавац контролише да ли је коректно унео податке о муштерији. (АНСО)
7. Продавац позива систем да запамти податке о муштерији. (АПСО)
8. Систем памти податке о муштерији. (СО)
9. Систем приказује продавцу муштерију и поруку: "Систем је запамтио муштерију." (ИА)

### Алтернативна сценарија:

- 4.1 Уколико систем не може да нађе муштерију он приказује продавцу поруку: "Изаберите муштерију". Прекида се извршење сценарија. (ИА)
- 9.1 Уколико систем не може да запамти податке о муштерији он приказује продавцу поруку: "Систем не може да запамти муштерију". (ИА)

## СК7- Обриши муштерија

### Назив СК

Обриши муштерија

### Актори СК

Продавац

### Учесници СК

Продавац, кориснички интерфејс (клијентски програм) и систем (серверски програм)

**Предуслови:** Кориснички интерфејс (клијентски програм) и систем (серверски програм) су покренути. Продавац је пријављен под својом шифром. Кориснички интерфејс приказује форму за рад са муштеријом. На наведеној екранској форми су дефинисани критеријуми, који се односе на: а) Муштерија б) Категорија муштерије, који ће да врате листу муштерија.

### Основни сценарио СК:

1. Продавац бира муштерију. (АПУСО)
2. Продавац контролише да ли је изабрао коректног муштерију. (АНСО)
3. Продавац позива систем да обрише муштерију. (АПСО)
4. Систем брише муштерију. (СО)
5. Систем приказује продавцу поруку: "Систем је обрисао муштерију." (ИА)

### Алтернативна сценарија:

- 5.1 Уколико систем не може да обрише муштерију јер је и даље пријављен он приказује продавцу поруку: "Систем не може да обрише муштерију која је пријављена. Прво одјавите муштерију". (ИА)
- 5.2 Уколико систем не може да обрише муштерију он приказује продавцу поруку: "Систем не може да обрише муштерију". (ИА)

## СК8- Пријави муштерија

### Назив СК

Пријави муштерија

### Актори СК

Муштерија

### Учесници СК

Муштерија, кориснички интерфејс (клијентски програм) и систем (серверски програм)

**Предуслови:** Кориснички интерфејс (клијентски програм) и систем (серверски програм) су покренути. Кориснички интерфејс приказује форму за **пријављивање**.

### Основни сценарио СК:

1. Муштерија **уноси** корисничко име и шифру. (АПУСО)
2. Муштерија **контролише** да ли је коректно унео корисничко име и шифру. (АНСО)
3. Муштерија **позива** систем да провери корисничко име и шифру. (АПСО)
4. Систем **проверава** корисничко име и шифру. (СО)
5. Систем **приказује муштерији** поруку: “Успешно сте се пријавили.” (ИА)
6. Кориснички интерфејс **позива** главни форму и мени. (КИПГФМ)

### Алтернативна сценарија:

- 5.1 Уколико систем провером установи да корисничка шифра и/или шифра нису исправни он **приказује муштерији** поруку: “Корисничко име и шифра нису исправни”. (ИА)
- 5.2 Уколико систем провером установи да је корисник са тим креденцијалима већ пријављен он **приказује муштерији** поруку: “Корисник је већ пријављен на овом налогу”. (ИА)
- 6.1 Уколико кориснички интерфејс не може да отвори главну форму и мени **приказује продавцу** поруку: “Не може да се отвори главна форма и мени”. (НПГФМ)

**Постуслови:** Отворена главна форма и мени.

## СК9- Пријави продавац

### Назив СК

Пријави продавац

### Актори СК

Продавац

### Учесници СК

Продавац, кориснички интерфејс (клијентски програм) и систем (серверски програм)

**Предуслови:** Кориснички интерфејс (клијентски програм) и систем (серверски програм) су покренути. Кориснички интерфејс приказује форму за **пријављивање**.

### Основни сценарио СК:

1. Продавац **уноси** корисничко име и шифру. (АПУСО)
2. Продавац **контролише** да ли је коректно унео корисничко име и шифру. (АНСО)
3. Продавац **позива систем** да провери корисничко име и шифру. (АПСО)
4. Систем **проверава** корисничко име и шифру. (СО)
5. Систем **приказује продавцу** поруку: “ Успешно сте се пријавили.” (ИА)
6. Кориснички интерфејс **позива** главни форму и мени. (КИПГФМ)

### Алтернативна сценарија:

5.1 Уколико систем провером установи да корисничка шифра и/или шифра нису исправни он **приказује продавцу** поруку: “Корисничко име и шифра нису исправни ”. (ИА)

6.1 Уколико кориснички интерфејс не може да отвори главну форму и мени **приказује продавцу** поруку: “Не може да се отвори главна форма и мени”. (НПГФМ)

**Постуслови:** Отворена главна форма и мени.

## СК22- Убацн термн дежурства

### Назв СК

Убацн термн дежурства

### Акторн СК

Продавац

### Учесннкн СК

Продавац, корисннкн интерфејс (кнјентскн програм) н снстем (серверскн програм)

**Предусловн:** Корисннкн интерфејс (кнјентскн програм) н снстем (серверскн програм) су покренутн. Продавац је прнјављен под својом шнфром. Снстем прнказује форму за рад са термном дежурства.

### Основнн сценарно СК:

1. Продавац уносн податке о термну дежурства. (АПУСО)
2. Продавац контролише да лн је коректно унео податке о термну дежурства. (АНСО)
3. Продавац познва снстем да запамтн податке о термну дежурства. (АПСО)
4. Снстем памтн податке о термну дежурства. (СО)
5. Снстем прнказује продавцу главну форму н поруку: “Снстем је успешно започео термн дежурства” (ИА)

### Алтернативна сценарнја:

5.1 Уколно снстем не може да запамтн податке о термну дежурства он прнказује продавцу поруку: “Овај термн дежурства је већ започет, снстем вас преусмерава на главну форму”. (ИА)



### 3. Анализа

#### 3.1 Понашање софтверског система - Одређивање системских операција на основу сценарија случаја коришћења

На основу наведених случајева коришћења уочене су следеће системске операције:

Шифра случаја коришћења	Назив случаја коришћења	Системске операције
SK1	СК1- Креирај рачун	1. signal KreirajRacun(Racun) 2. signal PromeniRacun(Racun) 3. signal vratiListuSviProdavac(Lista<Prodavac>) 4. signal vratiListuSviMusterija(Lista<Musterija>) 5. signal vratiListuSviUsluga(Lista<Usluga>)
SK2	СК2- Претражи рачун	1. signal PretraziRacun(Racun) 2. signal vratiListuRacun(kriterijumRacun, Lista<Racun>) 3. signal vratiListuRacun(kriterijumProdavac, Lista<Racun>) 4. signal vratiListuRacun(kriterijumMusterija, Lista<Racun>) 5. signal vratiListuRacun(kriterijumUsluga, Lista<Racun>)
SK3	СК3- Промени рачун	1. signal PromeniRacun(Racun) 2. signal PretraziRacun(Racun) 3. signal vratiListuSviProdavac(Lista<Prodavac>) 4. signal vratiListuSviMusterija(Lista<Musterija>) 5. signal vratiListuSviUsluga(Lista<Usluga>) 6. signal vratiListuRacun(kriterijumRacun, Lista<Racun>) 7. signal vratiListuRacun(kriterijumProdavac, Lista<Racun>) 8. signal vratiListuRacun(kriterijumMusterija, Lista<Racun>)
SK4	СК4- Креирај муштерија	1. signal KreirajMusterija(Musterija) 2. signal PromeniMusterija(Musterija) 3. signal vratiListuSviKategorijaMusterije(Lista<KategorijaMusterije>)
SK5	СК5- Претражи муштерија	1. signal PretraziMusterija(Musterija) 2. signal vratiListuMusterija(kriterijumMusterija, Lista<Musterija>) 3. signal vratiListuMusterija(kriterijumKategorijaMusterije, Lista<Musterija>)
SK6	СК6- Промени муштерија	1. signal PromeniMusterija(Musterija) 2. signal PretraziMusterija(Musterija) 3. signal vratiListuSviKategorijaMusterije(Lista<KategorijaMusterije>) 4. signal vratiListuMusterija(kriterijumMusterija, Lista<Musterija>) 5. signal vratiListuMusterija(kriterijumKategorijaMusterije, Lista<Musterija>)
SK7	СК7- Обриши муштерија	1. signal ObrisiMusterija(Musterija) 2. signal vratiListuMusterija(kriterijumMusterija, Lista<Musterija>) 3. signal vratiListuMusterija(kriterijumKategorijaMusterije, Lista<Musterija>)
SK8	СК8- Пријави муштерија	1. signal PrijaviMusterija(korisnickolme, sifra)
SK9	СК9- Пријави продавац	1. signal PrijaviProdavac(korisnickolme, sifra)
SK10	СК10- Креирај продавац	1. signal KreirajProdavac(Prodavac) 2. signal PromeniProdavac(Prodavac) 3. signal vratiListuSviTerminDezurstva(Lista<TerminDezurstva>)
SK11	СК11- Претражи продавац	1. signal PretraziProdavac(Prodavac) 2. signal vratiListuProdavac(kriterijumProdavac, Lista<Prodavac>) 3. signal vratiListuProdavac(kriterijumTerminDezurstva, Lista<Prodavac>)
SK12	СК12- Промени продавац	1. signal PromeniProdavac(Prodavac) 2. signal PretraziProdavac(Prodavac) 3. signal vratiListuSviTerminDezurstva(Lista<TerminDezurstva>) 4. signal vratiListuProdavac(kriterijumProdavac, Lista<Prodavac>) 5. signal vratiListuProdavac(kriterijumTerminDezurstva, Lista<Prodavac>)
SK13	СК13- Обриши продавац	1. signal ObrisiProdavac(Prodavac) 2. signal vratiListuProdavac(kriterijumProdavac, Lista<Prodavac>) 3. signal vratiListuProdavac(kriterijumTerminDezurstva, Lista<Prodavac>)
SK14	СК14- Креирај услуга	1. signal KreirajUsluga(Usluga) 2. signal PromeniUsluga(Usluga)
SK15	СК15- Претражи услуга	1. signal PretraziUsluga(Usluga) 2. signal vratiListuUsluga(kriterijumUsluga, Lista<Usluga>)

SK16	СК16-Промени услуга	1. signal PromeniUsluga(Usluga) 2. signal PretraziUsluga(Usluga) 3. signal vratiListuUsluga(kriterijumUsluga, Lista<Usluga>)
SK17	СК17-Обриши услуга	1. signal ObrisiUsluga(Usluga) 2. signal vratiListuUsluga(kriterijumUsluga, Lista<Usluga>)
SK18	СК18-Креирај категорија муштерије	1. signal KreirajKategorijaMusterije(KategorijaMusterije) 2. signal PromeniKategorijaMusterije(KategorijaMusterije)
SK19	СК19-Претражи категорија муштерије	1. signal PretraziKategorijaMusterije(KategorijaMusterije) 2. signal vratiListuKategorijaMusterije(kriterijumKategorijaMusterije, Lista<KategorijaMusterije>)
SK20	СК20-Промени категорија муштерије	1. signal PromeniKategorijaMusterije(KategorijaMusterije) 2. signal PretraziKategorijaMusterije(KategorijaMusterije) 3. signal vratiListuKategorijaMusterije(kriterijumKategorijaMusterije, Lista<KategorijaMusterije>)
SK21	СК21-Обриши категорија муштерије	1. signal ObrisiKategorijaMusterije(KategorijaMusterije) 2. signal vratiListuKategorijaMusterije(kriterijumKategorijaMusterije, Lista<KategorijaMusterije>)
SK22	СК22-Убаци терминде журства	1. signal UbaciTerminDezurstva(TerminDezurstva) 2. signal PromeniTerminDezurstva(TerminDezurstva)
SK23	СК23-Претражи терминде журства	1. signal PretraziTerminDezurstva(TerminDezurstva) 2. signal vratiListuTerminDezurstva(kriterijumTerminDezurstva, Lista<TerminDezurstva>)
SK24	СК24-Промени терминде журства	1. signal PromeniTerminDezurstva(TerminDezurstva) 2. signal PretraziTerminDezurstva(TerminDezurstva) 3. signal vratiListuTerminDezurstva(kriterijumTerminDezurstva, Lista<TerminDezurstva>)
SK25	СК25-Обриши терминде журства	1. signal ObrisiTerminDezurstva(TerminDezurstva) 2. signal vratiListuTerminDezurstva(kriterijumTerminDezurstva, Lista<TerminDezurstva>)
SK26	СК26-Креирај ставка рачуна	1. signal KreirajStavkaRacuna(StavkaRacuna) 2. signal PromeniStavkaRacuna(StavkaRacuna)
SK27	СК27-Обриши ставка рачуна	1. signal ObrisiStavkaRacuna(StavkaRacuna)
SK28	СК28-Претражи ставка рачуна	1. signal PretraziStavkaRacuna(StavkaRacuna)
SK29	СК29-Промени ставка рачуна	1. signal PromeniStavkaRacuna(StavkaRacuna) 2. signal PretraziStavkaRacuna(StavkaRacuna)
SK30	СК30-Убаци ставка рачуна	1. signal UbaciStavkaRacuna(StavkaRacuna) 2. signal PromeniStavkaRacuna(StavkaRacuna)

Табела 4: Системске операције случаја коришћења

Наводимо све уочене системске операције:

1. signal KreirajKategorijaMusterije(KategorijaMusterije)
2. signal KreirajMusterija(Musterija)
3. signal KreirajProdavac(Prodavac)
4. signal KreirajRacun(Racun)
5. signal KreirajStavkaRacuna(StavkaRacuna)
6. signal KreirajUsluga(Usluga)
7. signal ObrisiKategorijaMusterije(KategorijaMusterije)
8. signal ObrisiMusterija(Musterija)
9. signal ObrisiProdavac(Prodavac)
10. signal ObrisiStavkaRacuna(StavkaRacuna)
11. signal ObrisiTerminDezurstva(TerminDezurstva)
12. signal ObrisiUsluga(Usluga)
13. signal PretraziKategorijaMusterije(KategorijaMusterije)
14. signal PretraziMusterija(Musterija)
15. signal PretraziProdavac(Prodavac)
16. signal PretraziRacun(Racun)
17. signal PretraziStavkaRacuna(StavkaRacuna)
18. signal PretraziTerminDezurstva(TerminDezurstva)
19. signal PretraziUsluga(Usluga)
20. signal PrijaviMusterija(korisnickolme, sifra)
21. signal PrijaviProdavac(korisnickolme, sifra)
22. signal PromeniKategorijaMusterije(KategorijaMusterije)
23. signal PromeniMusterija(Musterija)
24. signal PromeniProdavac(Prodavac)
25. signal PromeniRacun(Racun)
26. signal PromeniStavkaRacuna(StavkaRacuna)
27. signal PromeniTerminDezurstva(TerminDezurstva)
28. signal PromeniUsluga(Usluga)
29. signal UbaciStavkaRacuna(StavkaRacuna)
30. signal UbaciTerminDezurstva(TerminDezurstva)
31. signal vratiListuKategorijaMusterije(kriterijumKategorijaMusterije, Lista<KategorijaMusterije>)
32. signal vratiListuMusterija(kriterijumKategorijaMusterije, Lista<Musterija>)
33. signal vratiListuMusterija(kriterijumMusterija, Lista<Musterija>)
34. signal vratiListuProdavac(kriterijumProdavac, Lista<Prodavac>)
35. signal vratiListuProdavac(kriterijumTerminDezurstva, Lista<Prodavac>)
36. signal vratiListuRacun(kriterijumMusterija, Lista<Racun>)
37. signal vratiListuRacun(kriterijumProdavac, Lista<Racun>)
38. signal vratiListuRacun(kriterijumRacun, Lista<Racun>)
39. signal vratiListuRacun(kriterijumUsluga, Lista<Racun>)
40. signal vratiListuSviKategorijaMusterije(Lista<KategorijaMusterije>)
41. signal vratiListuSviMusterija(Lista<Musterija>)
42. signal vratiListuSviProdavac(Lista<Prodavac>)
43. signal vratiListuSviTerminDezurstva(Lista<TerminDezurstva>)
44. signal vratiListuSviUsluga(Lista<Usluga>)
45. signal vratiListuTerminDezurstva(kriterijumTerminDezurstva, Lista<TerminDezurstva>)
46. signal vratiListuUsluga(kriterijumUsluga, Lista<Usluga>)

Табела 5: Системске операције софтверског система

### 3.2 Понашање софтверског система - Секвенци дијаграми случаја коришћења

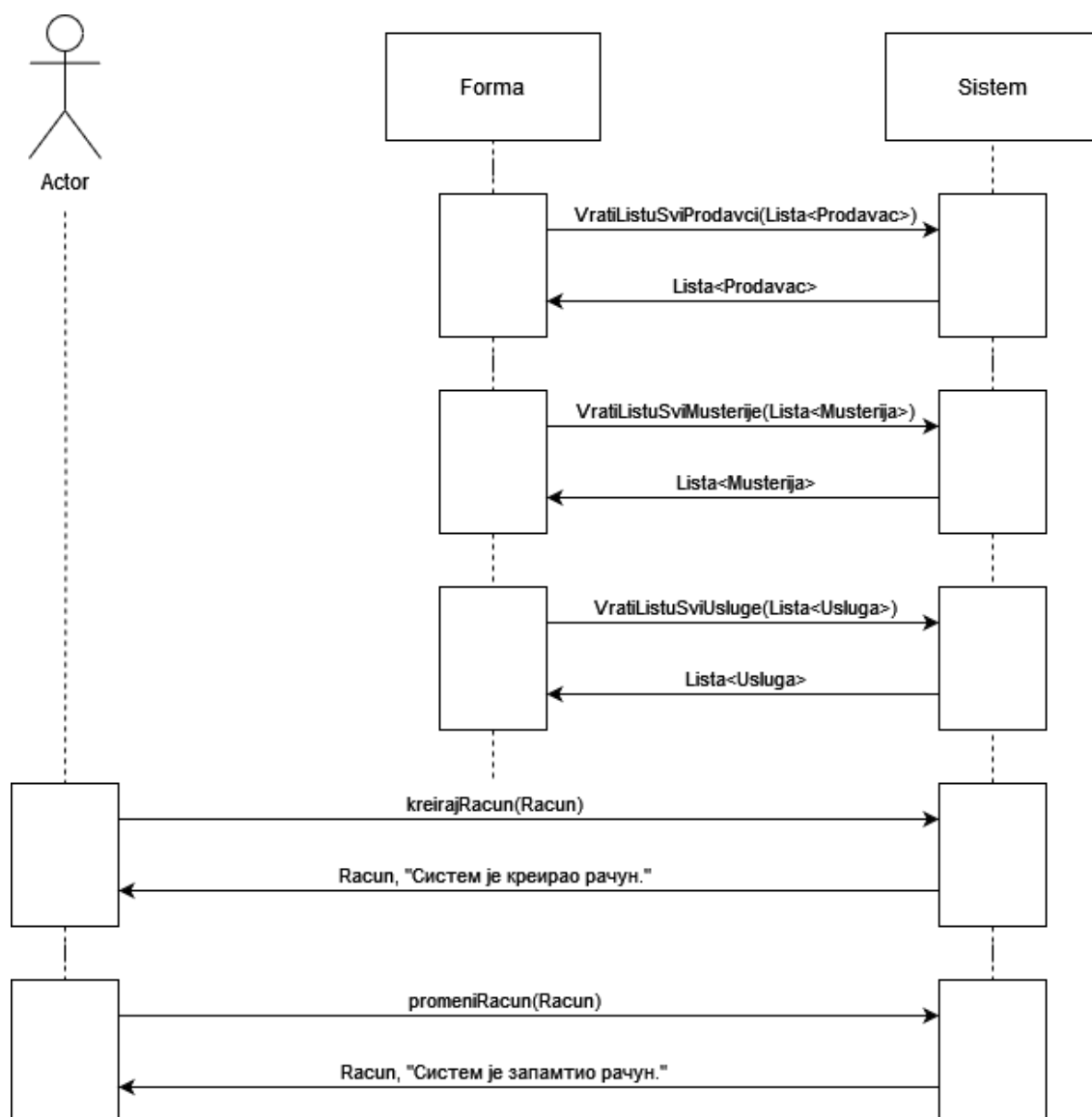
#### ДС1: Дијаграм секвенци случаја коришћења – Креирај рачун

Предуслови:

1. **Форма** позива **систем** да **врати** листу свих продаваца. (АПСО)
2. **Систем** враћа **форми** листу свих продаваца. (ИА)
3. **Форма** позива **систем** да **врати** листу свих муштерија. (АПСО)
4. **Систем** враћа **форми** листу свих муштерија. (ИА)
5. **Форма** позива **систем** да **врати** листу свих услуга. (АПСО)
6. **Систем** враћа **форми** листу свих услуга. (ИА)

Основни сценарио СК:

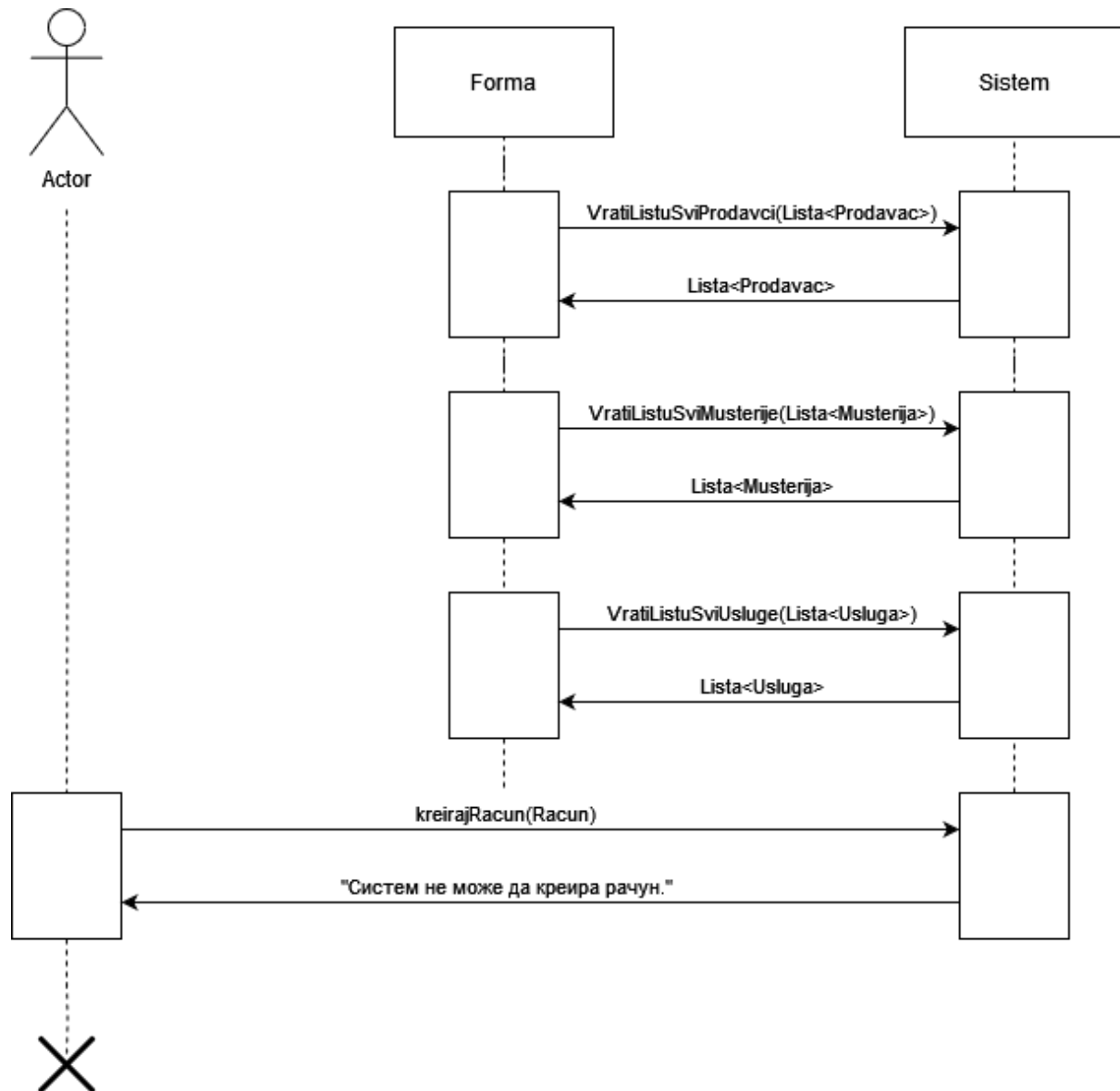
7. **Продавац** позива **систем** да **креира рачун**. (АПСО)
8. **Систем** приказује **продавцу рачун** и поруку: “**Систем** је креирао **рачун**”. (ИА)
9. **Продавац** позива **систем** да запамти податке о **рачуну**. (АПСО)
10. **Систем** приказује **продавцу рачун** и поруку: “**Систем** је запамтио **рачун**.” (ИА)



Слика 1 - Секвенци дијаграм креирања рачуна

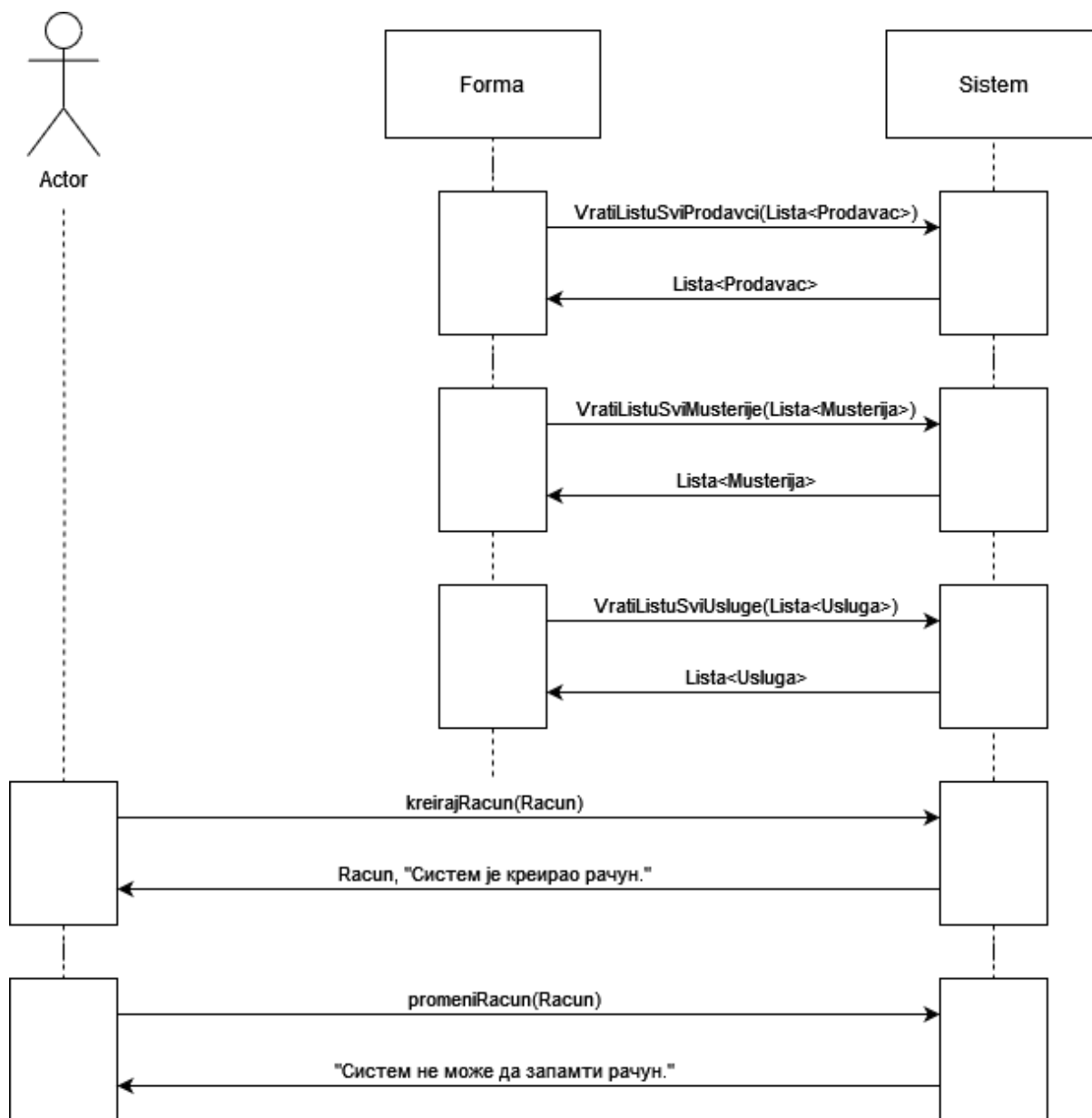
## Алтернативна сценарија:

8.1 Уколико **систем** не може да креира **рачун** он **приказује** **продавцу** поруку: “**Систем** не може да креира **рачун**”. Прекида се извршење сценарија. (IA)



Слика 2 - Секвенчни дијаграм креирања рачуна (алтернативни сценарио 1)

10.1 Уколико **систем** не може да запамти податке о **рачуну** он **приказује** **продавцу** поруку: “**Систем** не може да запамти **рачун**”. (ИА)



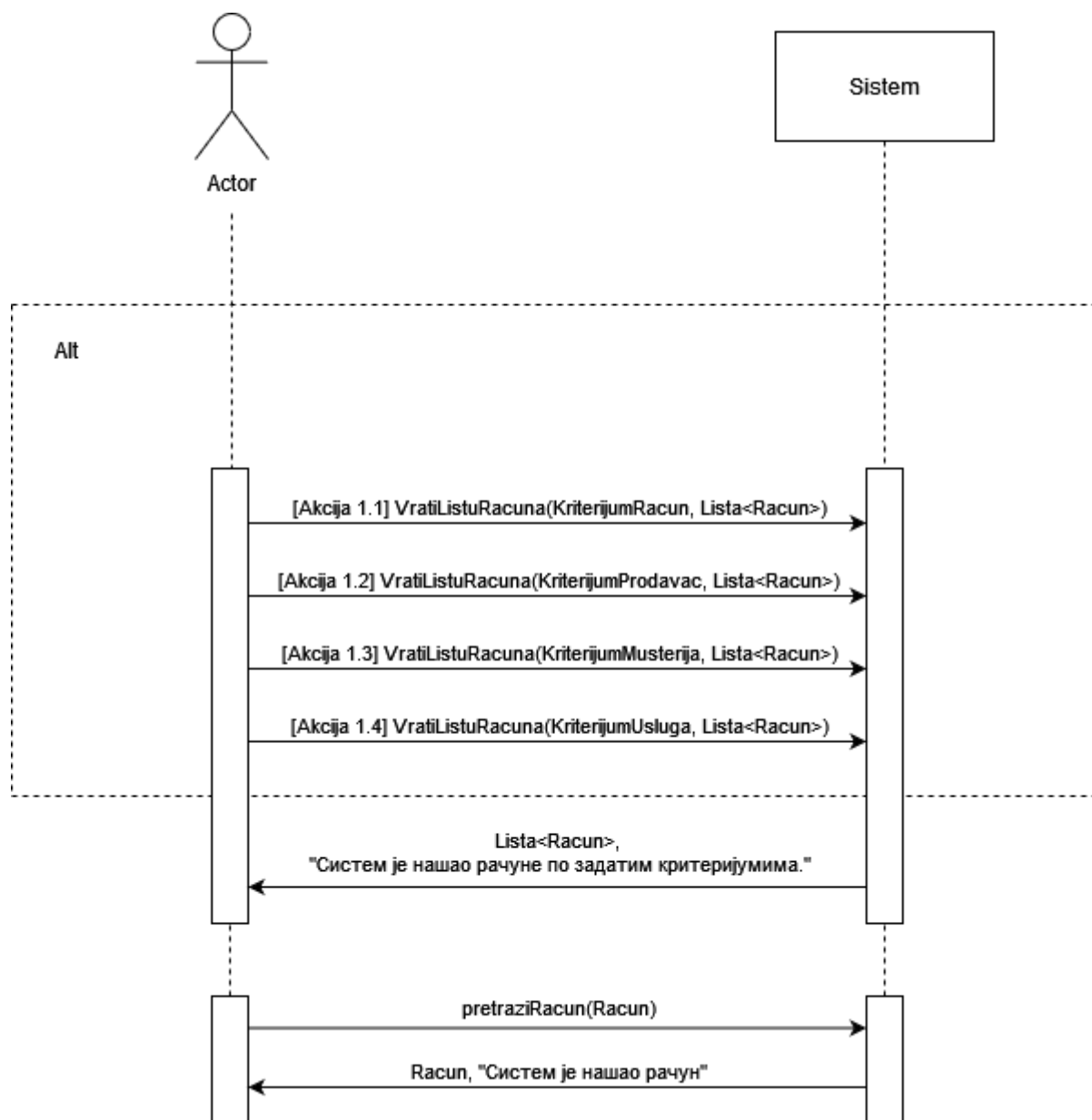
Слика 3 - Секвенцни дијаграм креирања рачуна (алтернативни сценарио 2)

1.	signal kreirajRacun(Racun)
2.	signal promeniRacun(Racun)
3.	signal VratiListuSviProdavci(Lista<Prodavac>)
4.	signal VratiListuSviMusterije(Lista<Musterija>)
5.	signal VratiListuSviUsluge(Lista<Usluga>)

## ДС2: Дијаграм секвенци случаја коришћења – Претражи рачун

### Основни сценарио СК:

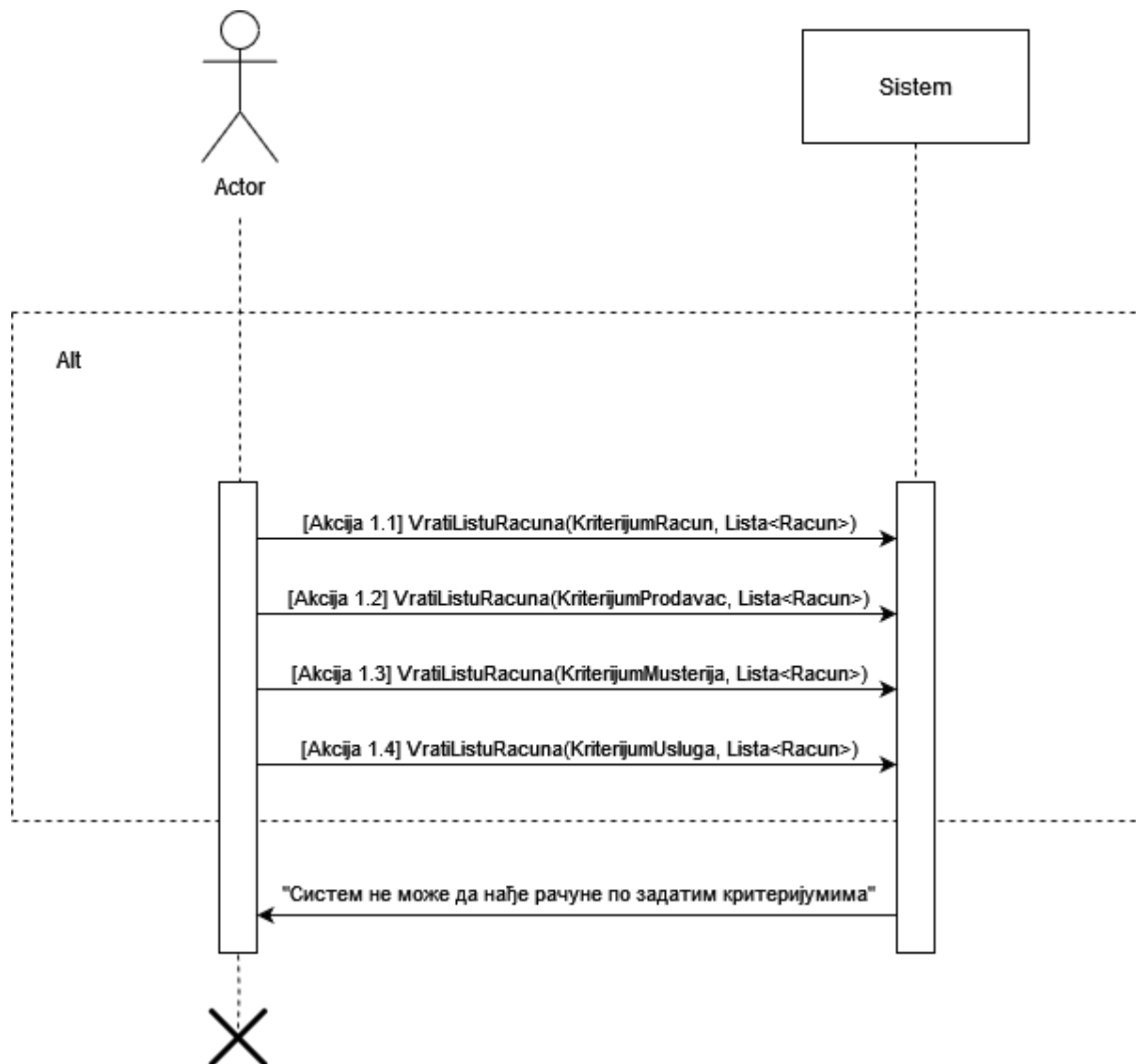
1. **Продавац** позива **систем** да нађе **рачуне** по задатим критеријумима. (АПСО)
2. **Систем** приказује **продавцу** **рачуне** и поруку: "**Систем** је нашао **рачуне** по задатим критеријумима". (ИА)
3. **Продавац** позива **систем** да нађе **рачун**. (АПСО)
4. **Систем** приказује **продавцу** **рачун** и поруку: "**Систем** је нашао **рачун**". (ИА)



Слика 4 - Секвенци дијаграм претраге рачуна

## Алтернативна сценарија:

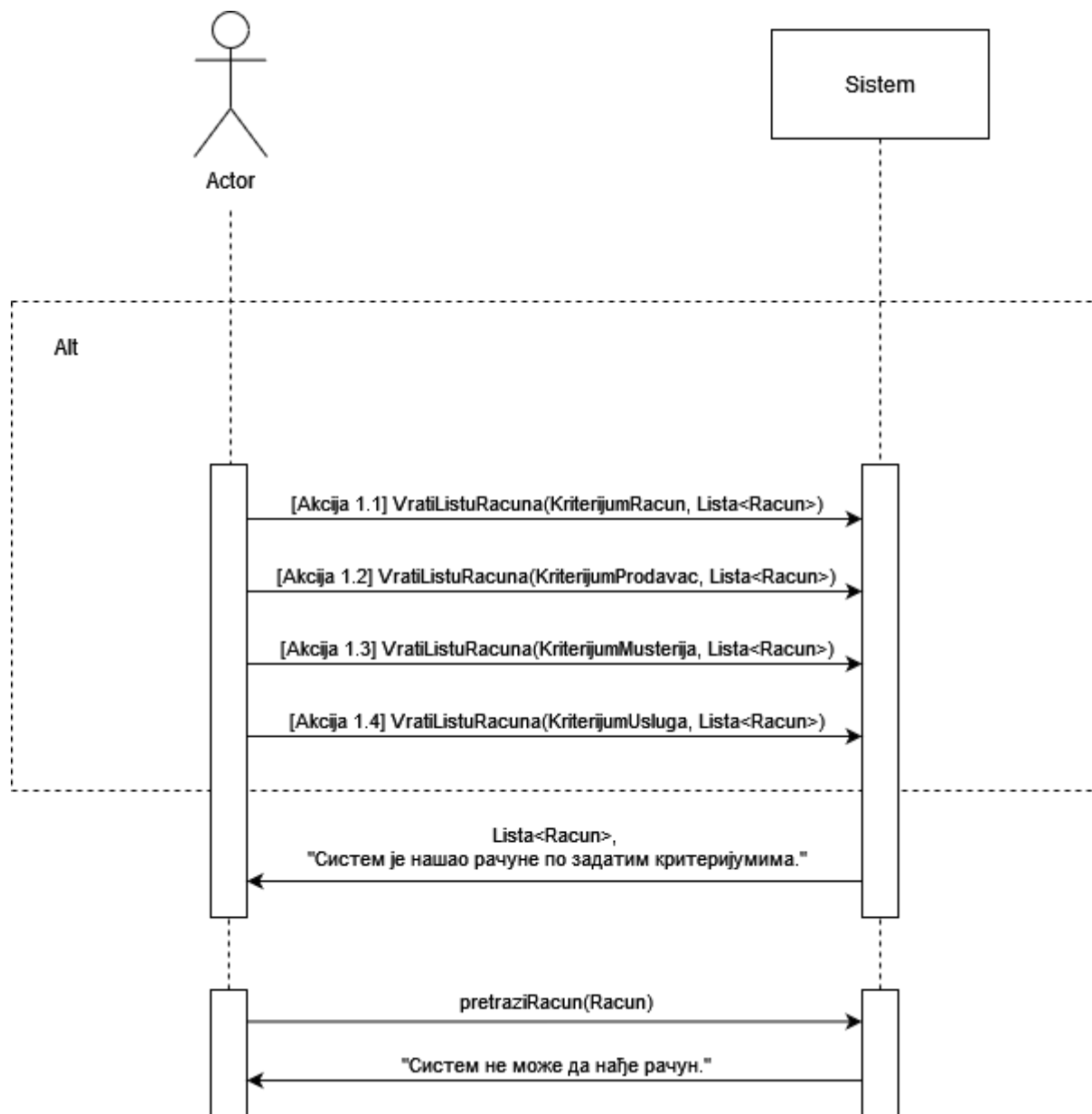
2.1 Уколико **систем** не може да нађе **рачуне** он **приказује продавцу** поруку: “**Систем** не може да нађе **рачуне** по задатим критеријумима”. Прекида се извршење сценарија. (ИА)



Слика 5 - Секвенци дијаграм претраге рачуна (алтернативни сценарио 1)



4.1 Уколико **систем** не може да нађе **рачун** он **приказује** **продавцу** поруку: “Систем не може да нађе **рачун**”.(ИА)



Слика 6 - Секвенци дијаграм претраге рачуна (алтернативни сценарио 2)

1.	signal pretraziRacun(Racun)
2.	signal VратиListuRacuna(KriterijumRacun, Lista<Racun>)
3.	signal VратиListuRacuna(KriterijumProdavac, Lista<Racun>)
4.	signal VратиListuRacuna(KriterijumMusterija, Lista<Racun>)
5.	signal VратиListuRacuna(KriterijumUsluga, Lista<Racun>)

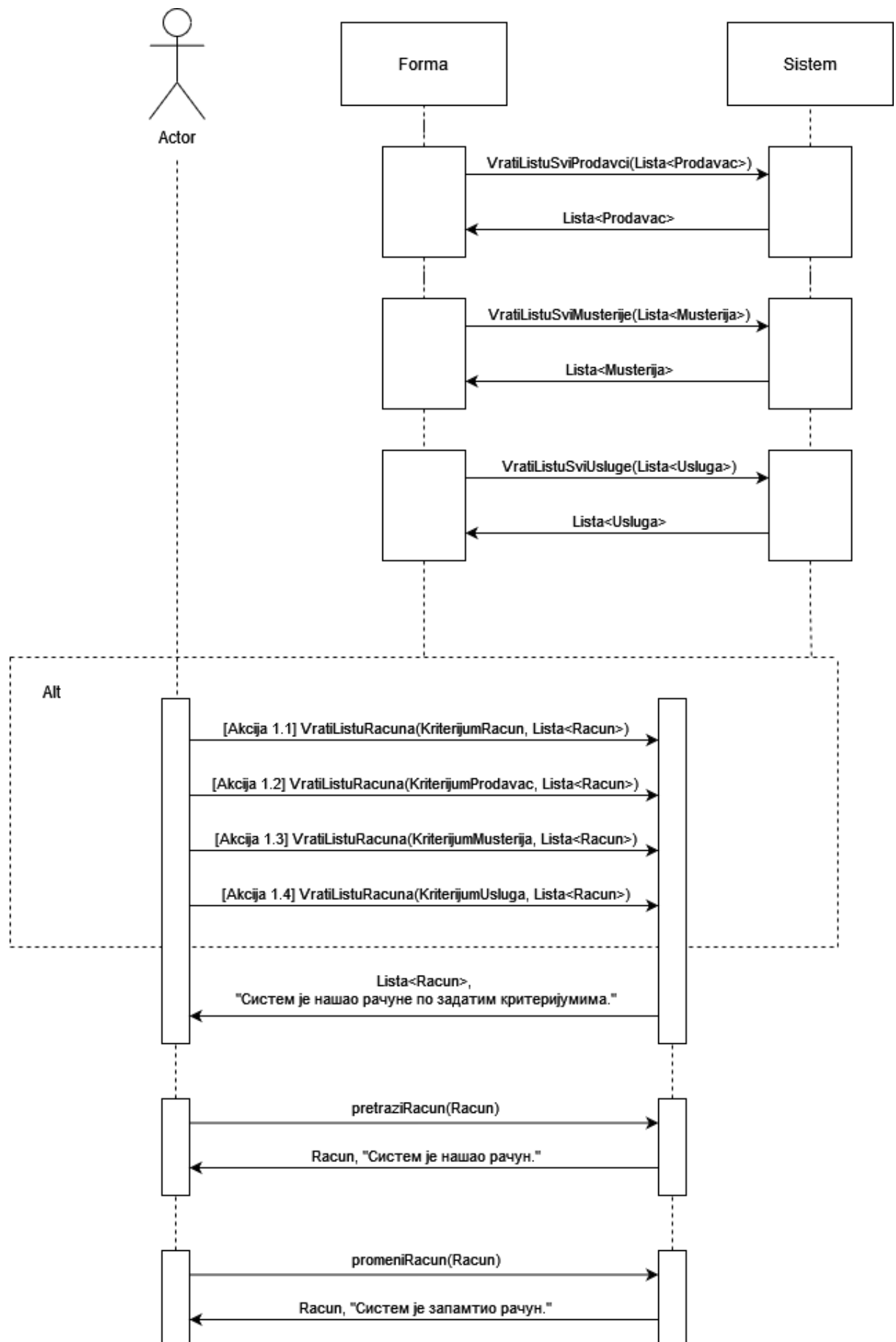
### **ДСЗ: Дијаграм секвенци случаја коришћења – Промени рачун**

#### **Предуслови:**

1. **Форма** **позива** **систем** да **врати** листу свих продаваца. (АПСО)
2. **Систем** **враћа** **форми** листу свих продаваца. (ИА)
3. **Форма** **позива** систем да **врати** листу свих муштерија. (АПСО)
4. **Систем** **враћа** **форми** листу свих муштерија. (ИА)
5. **Форма** **позива** систем да **врати** листу свих услуга. (АПСО)
6. **Систем** **враћа** **форми** листу свих услуга. (ИА)

#### **Основни сценарио СК:**

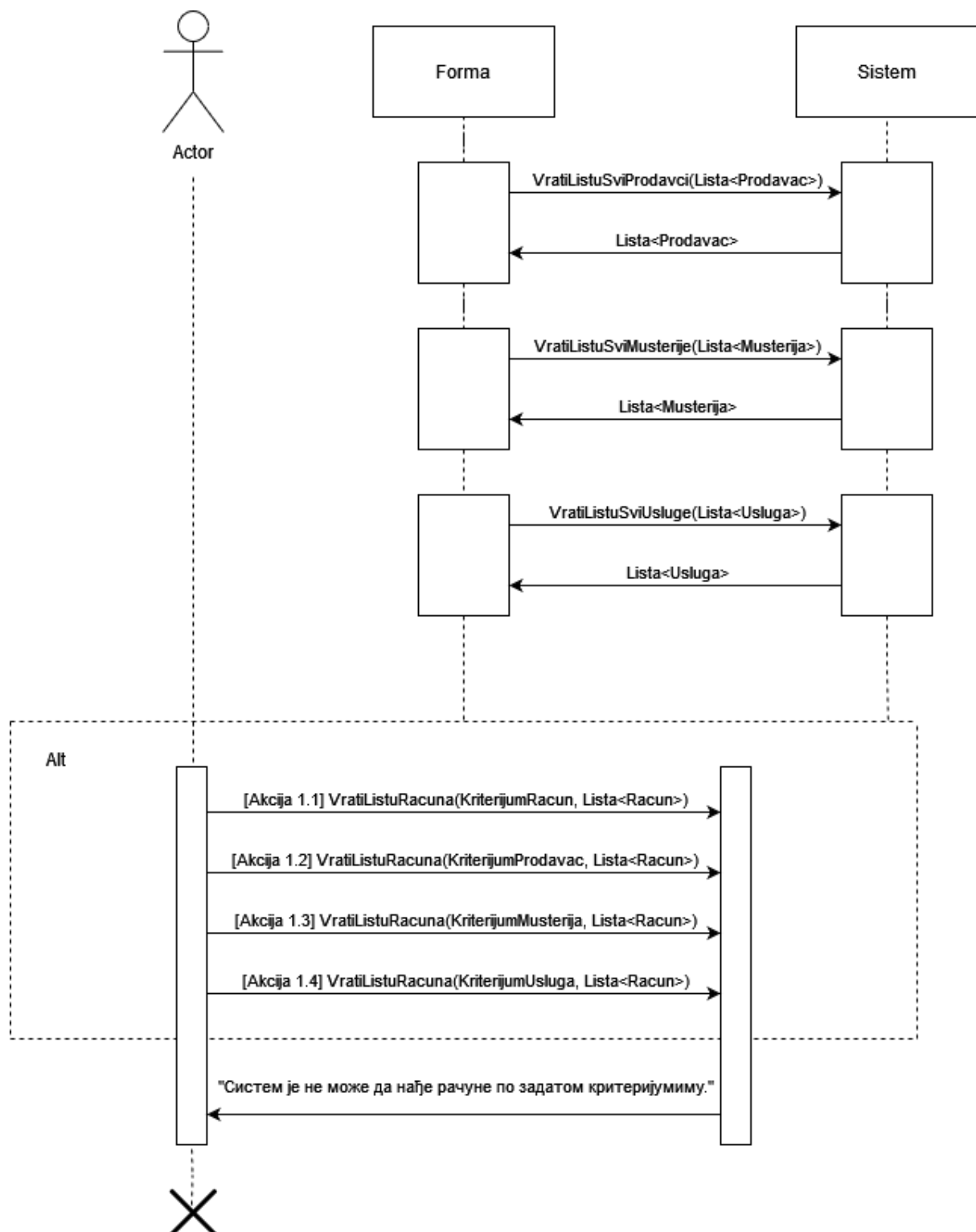
7. **Продавац** **позива** **систем** да нађе **рачуне** по задатим критеријумима. (АПСО)
8. **Систем** **приказује** **продавцу** **рачуне** и поруку: “**Систем** је нашао **рачуне** по задатим критеријумима”. (ИА)
9. **Продавац** **позива** **систем** да нађе **рачун**. (АПСО)
10. **Систем** **приказује** **продавцу** **рачун** и поруку: “**Систем** је нашао **рачун**”. (ИА)
11. **Продавац** **позива** **систем** да запамти податке о **рачуну**. (АПСО)
12. **Систем** **приказује** **продавцу** **рачун** и поруку: “**Систем** је запамтио **рачун**.” (ИА)



Слика 7 - Секвенци дијаграм промене рачуна

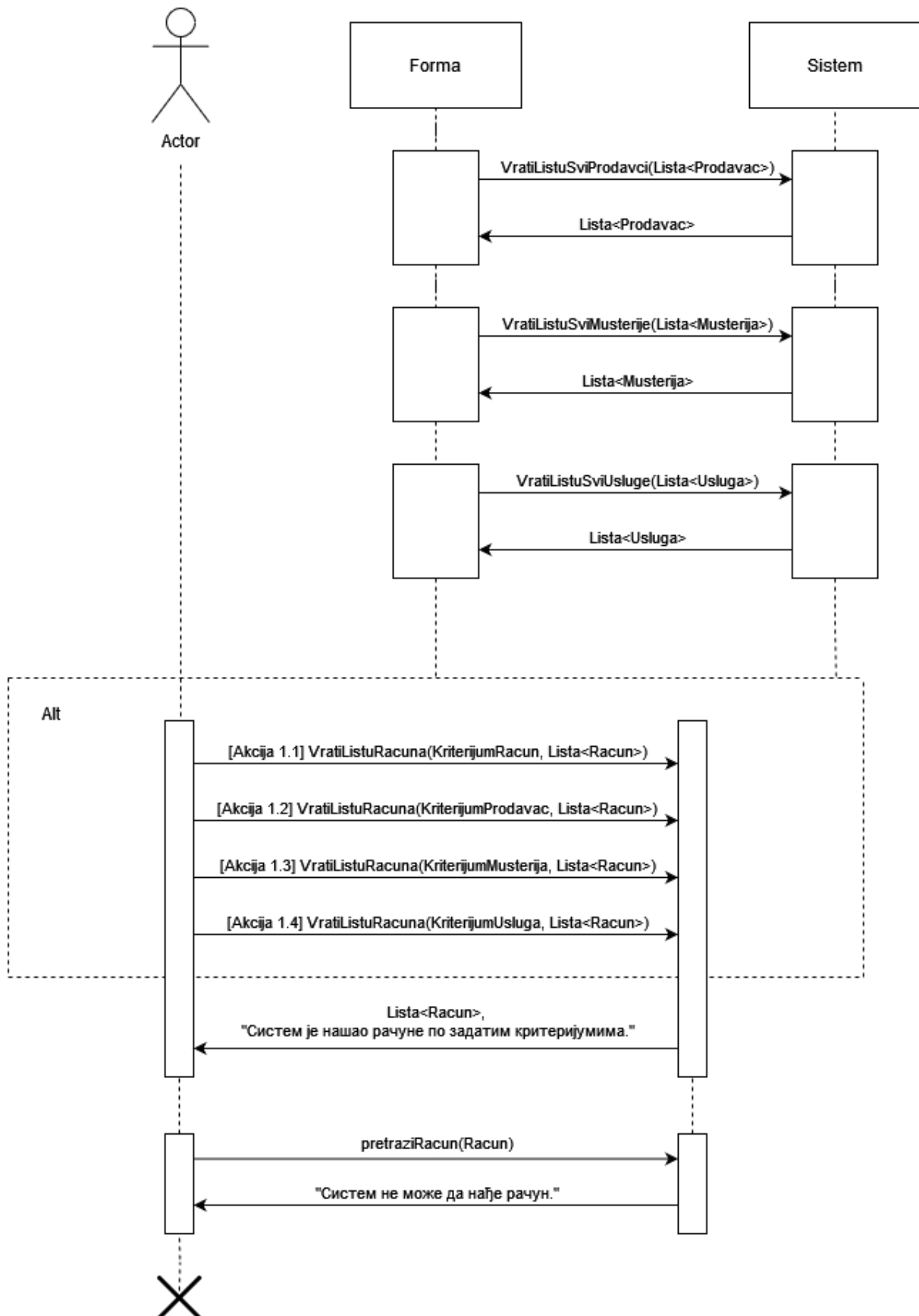
## Алтернативна сценарија:

8.1 Уколико **систем** не може да нађе **рачуне** он **приказује продавцу** поруку: “**Систем** не може да нађе **рачуне** по задатом критеријуму”. Прекида се извршење сценарија. (ИА)



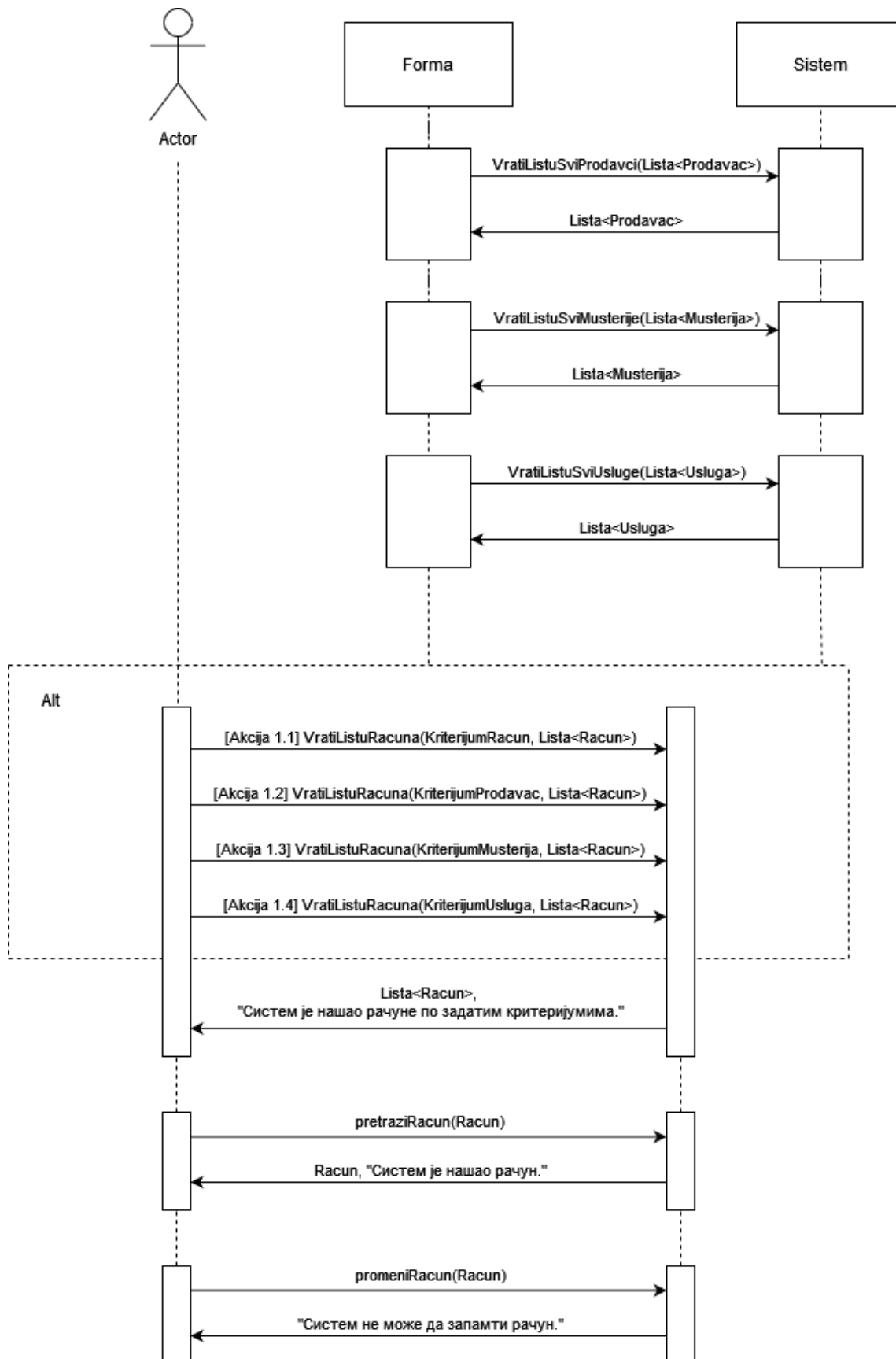
Слика 8 - Секвенчни дијаграм промене рачуна (алтернативни сценарио 1)

10.1 Уколико **систем** не може да нађе **рачун** он **приказује** **продавцу** поруку: “Систем не може да нађе **рачун**”. Прекида се извршење сценарија. (ИА)



Слика 9 - Секвенцни дијаграм промене рачуна (алтернативни сценарио 2)

12.1 Уколико **систем** не може да запамти податке о **рачу**ну он **приказује** **продавцу** поруку: “**Систем** не може да запамти **рачун**”. (ИА)



Слика 10 - Секвенци дијаграм промене рачуна (алтернативни сценарио 3)

1.	signal promeniRacun(Racun)
2.	signal pretraziRacun(Racun)
3.	signal VratilistuSviProdavci(Lista<Prodavac>)
4.	signal VratilistuSviMusterije(Lista<Musterija>)
5.	signal VratilistuSviUsluge(Lista<Usluga>)
6.	signal VratilistuRacuna(KriterijumRacun, Lista<Racun>)
7.	signal VratilistuRacuna(KriterijumProdavac, Lista<Racun>)
8.	signal VratilistuRacuna(KriterijumMusterija, Lista<Racun>)
9.	signal VratilistuRacuna(KriterijumUsluga, Lista<Racun>)

### **3.3 Понашање софтверског система - Дефинисање уговора о системским операцијама**

За системске операције се праве уговори. Овде ћемо навести осам различитих (типских) уговора за системске операције.

#### **1. Уговор UG1: *PrijavaProdavac(korisnickolme, Sifra)***

**Операција:** *PrijavaProdavac(korisnickolme, Sifra):signal;*

**Веза са СК:** СК9

**Предуслови:**

**Постуслови:** *Корисник је пријављен на систем.*

#### **2. Уговор UG2: *UbaciRacun(Racun, List<StavkaRacuna>)***

**Операција:** *UbaciRacun(Racun, List<StavkaRacuna>):signal;*

**Веза са СК:** СК1

**Предуслови:** *Структурна и вредносна ограничење над објектом класе Рачун морају бити задовољена.*

**Постуслови:** *Направљен је нови објекат класе Рачун.*

#### **3. Уговор UG3: *UbaciTerminDezurstva(TerminDezurstva)***

**Операција:** *UbaciTerminDezurstva(TerminDezurstva):signal;*

**Веза са СК:** СК22

**Предуслови:** *Структурна и вредносна ограничење над објектом класе ТерминДежурства морају бити задовољена.*

**Постуслови:** *Направљен је нови објекат класе ТерминДежурства.*

#### **4. Уговор UG4: *PromeniRacun(Racun)***

**Операција:** *PromeniRacun(Racun):signal;*

**Веза са СК:** СК2, СК3

**Предуслови:** *Структурна и вредносна ограничење над објектом класе Рачун морају бити задовољена.*

**Постуслови:** *Објекат класе Рачун је промењен.*



#### **5. Уговор UG5: *ObrisiMusterija(Musterija)***

**Операција:** *ObrisiMusterija(Musterija):signal;*

**Веза са СК:** СК7

**Предуслови** Структурна и вредносна ограничење над објектом класе *Муштерија* морају бити задовољена.

**Постуслови:** *Објекат класе Муштерија је обрисан.*

#### **6. Уговор UG6: *PretraziRacun(Racun)***

**Операција:** *PretražiRacun(Racun):signal;*

**Веза са СК:** СК2

**Предуслови:**

**Постуслови:** *Пронађен је тражени објекат класе Рачун.*

#### **7. Уговор UG7: *vratiListuRacuna(KriterijumRacun, Lista<Racun>)***

**Операција:** *vratiListuRacuna (KriterijumRacun,Lista<Racun> ):signal;*

**Веза са СК:** СК2

**Предуслови:**

**Постуслови:** *Пронађена је листа тражених објеката класе Рачун.*

#### **8. Уговор UG8: *vratiListuSviMusterije(Lista<Musterija>)***

**Операција:** *vratiListuSviMusterije (Lista<Musterija> ):signal;*

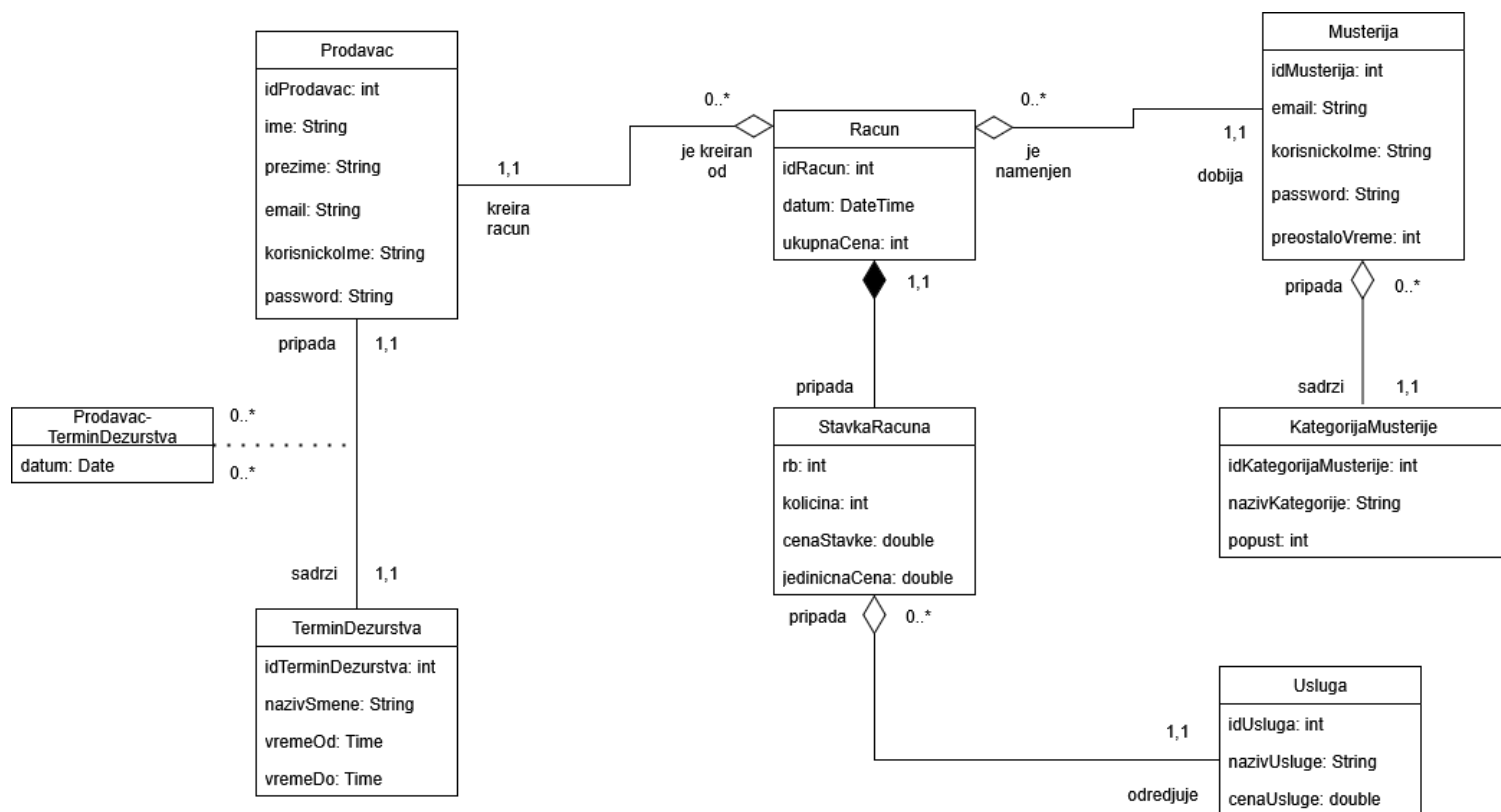
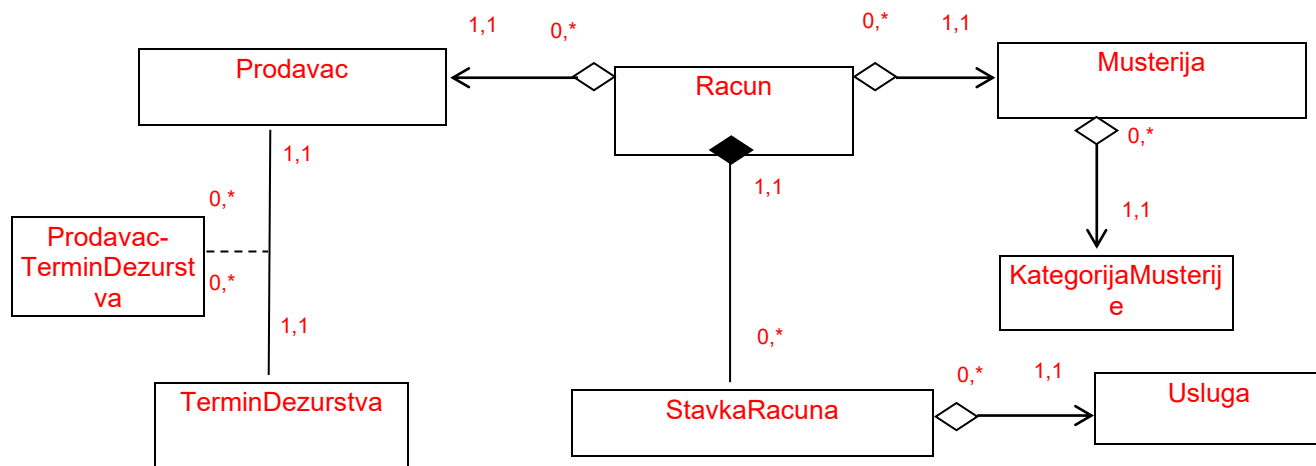
**Веза са СК:** СК5

**Предуслови:**

**Постуслови:** *Пронађена је листа свих објеката класе Муштерија.*

### 3.4 Структура софтверског система – Концептуални (доменски) модел

Тип концептуалног модела 1:



Слика 11 – Дијаграм концептуалног модела

### 3.5 Структура софтверског система – Релациони модел

На основу концептуалног модела се прави релациони модел .

Релациони модел добијен из **првог** типа концептуалног модела:

1. **Prodavac** (idProdavac, ime, prezime, email, korisnickolme, password)
2. **Usluga** (idUsluga, nazivUsluge, cenaUsluge)
3. **KategorijaMusterije** (idKategorijaMusterije, nazivKategorije, popust)
4. **TerminDezurstva** (idTerminDezurstva, nazivSmene, vremeOd, vremeDo)
5. **Musterija** (idMusterija, email, korisnickolme, password, preostaloVreme, *idKategorijaMusterije*)
6. **Racun** (idRacun, datum, ukupnaCena, *idProdavac*, *idMusterija*)
7. **StavkaRacuna** (idRacun, rb, kolicina, cenaStavke, jedinicaCena, *idUsluga*)
8. **Prodavac-TerminDezurstva** (idProdavac, idTerminDezurstva, datum)

### 3.6 Табела структурних и вредносних ограничења релационог модела

За сваку релацију се прави табела структурних и вредносних ограничења.

Табела структурних и вредносних ограничења релационог модела који је добијен из **првог** типа концептуалног модела:

1.Табела <b>Prodavac</b>		Просто вредносно ограничење		Сложено вредносно ограничење		Структурно ограничење
Атрибути	Назив	Тип атрибута	Вредност атрибута	Међузав. атрибута једне табеле	Међузав.атрибута више табела	INSERT /  UPDATE CASCADES <b>Racun</b> , <b>Prodavac-TerminDezurstva</b>  DELETE RESTRICTED <b>Racun</b> , <b>Prodavac-TerminDezurstva</b>
	<b>idProdavac</b>	integer	Not null and >0			
	ime	String	Not null			
	prezime	String	Not null			
	email	String	Not null and like '%@%'			
	korisnickolme	String	Not null			
	password	String	Not null and length(password) > 6	!=korisnickolme		

2.Табела <b>Usluga</b>		Просто вредносно ограничење		Сложено вредносно ограничење		Структурно ограничење
Атрибути	Назив	Тип атрибута	Вредност атрибута	Међузав. атрибута једне табеле	Међузав.атрибута више табела	INSERT /  UPDATE CASCADES <b>StavkaRacuna</b> ,  DELETE RESTRICTED <b>StavkaRacuna</b> ,
	<b>idUsluga</b>	integer	Not null and >0			
	nazivUsluge	String	Not null			
	cenaUsluge	double	Not null and >=0			

3.Табела <b>KategorijaMusterije</b>		Просто вредносно ограничење		Сложено вредносно ограничење		Структурно ограничење
Атрибути	Назив	Тип атрибута	Вредност атрибута	Међузав. атрибута једне табеле	Међузав.атрибута више табела	INSERT /  UPDATE CASCADES <b>Musterija</b> ,  DELETE RESTRICTED <b>Musterija</b> ,
	<b>idKategorijaMusterije</b>	integer	Not null and >0			
	nazivKategorije	String	Not null			
	popust	integer	Not null and between 0 and 100			

4.Табела TerminDezurstva		Просто вредносно ограничење		Сложено вредносно ограничење		Структурно ограничење
Атрибути	Назив	Тип атрибута	Вредност атрибута	Међузав. атрибута једне табеле	Међузав.атрибута више табела	INSERT /  UPDATE CASCADES Prodavac-TerminDezurstva ,  DELETE RESTRICTED Prodavac-TerminDezurstva ,
	<u>idTerminDezurstva</u>	integer	Not null and >0			
	nazivSmene	String	Not null			
	vremeOd	Time	Not null and between 00:00 and 23:59			
	vremeDo	Time	Not null and between 00:00 and 23:59			

5.Табела Musterija		Просто вредносно ограничење		Сложено вредносно ограничење		Структурно ограничење
Атрибути	Назив	Тип атрибута	Вредност атрибута	Међузав. атрибута једне табеле	Међузав.атрибута више табела	INSERT RESTRICTED KategorijaMusterije  UPDATE RESTRICTED KategorijaMusterije  UPDATE CASCADES Racun  DELETE RESTRICTED Racun
	<u>idMusterija</u>	integer	Not null and >0			
	email	String	Not null and like '%@%'			
	korisnickolme	String	Not null			
	password	String	Not null and length(password) > 6	!=korisnickolme		
	idKategorijaMusterije	integer	Not null and >0			
	preostaloVreme	integer	Not null and >=0			

6.Табела Racun		Просто вредносно ограничење		Сложено вредносно ограничење		Структурно ограничење
Атрибути	Назив	Тип атрибута	Вредност атрибута	Међузав. атрибута једне табеле	Међузав.атрибута више табела	INSERT RESTRICTED Musterija Prodavac  UPDATE RESTRICTED Musterija Prodavac  UPDATE CASCADES StavkaRacuna  DELETE RESTRICTED StavkaRacuna
	<u>idRacun</u>	integer	Not null and >0			
	datum	Date	Not null			
	ukupnaCena	double	Not null and >= 0		=SUM(StavkaRacuna.cenaStavke) * (100 – KategorijaMusterije.popust) / 100	
	idProdavac	integer	Not null and >0			
	idMusterija	integer	Not null and >0			

7.Табела StavkaRacuna		Просто вредносно ограничење		Сложено вредносно ограничење		Структурно ограничење
Атрибути	Назив	Тип атрибута	Вредност атрибута	Међузава. атрибута једне табеле	Међузава. атрибута више табела	INSERT RESTRICTED Racun Usluga  UPDATE RESTRICTED Racun Usluga  DELETE /
	<u>idRacun</u>	integer	Not null and >0			
	<u>rb</u>	integer	Not null and >0			
	kolicina	integer	Not null and >0			
	cenaStavke	double	Not null and >=0	=jedinicnaCena * kolicina		
	jedinicnaCena	double	Not null and >=0			
	idUsluga	integer	Not null and >0			

8.Табела Prodavac-TerminDezurstva		Просто вредносно ограничење		Сложено вредносно ограничење		Структурно ограничење
Атрибути	Назив	Тип атрибута	Вредност атрибута	Међузава. атрибута једне табеле	Међузава. атрибута више табела	INSERT RESTRICTED Prodavac TerminDezurstva  UPDATE RESTRICTED  Prodavac TerminDezurstva  DELETE /
	<u>idProdavac</u>	integer	Not null and >0			
	<u>idTerminDezurstva</u>	integer	Not null and >0			
	datum	Date	Not null			

## 4. Пројектовање

Фаза пројектовања обухвата дефинисање архитектуре софтверског система, односно његове структуре и понашања. У том процесу пројектују се апликациона логика, база података и кориснички интерфејс, који подразумева израду екранских форми и контролера. Посебна пажња посвећује се пројектовању контролера апликационе логике, приступа бази и пословној логици која дефинише функционисање система.

### 4.1 Пројектовање корисничког интерфејса

Кориснички интерфејс представља улазно-излазну реализацију софтверског система. Састоји се од:

1. Екранске форме
2. Контролера корисничког интерфејса



Слика 12 - Структура корисничког интерфејса

У наредним странама ће фокус бити на пројектовању екранских форми и приказу различитих случајева коришћења кроз графички корисничк интерфејс.

## СК1- Убацн рачун

### Назив СК

Убацн рачун

### Акторн СК

Продавац

### Учесннцн СК

**Продавац**, корисннчкн интерфејс (кнлнјентскн програм) н **снстем** (серверскн програм)

**Предусловн:** Корисннчкн интерфејс (кнлнјентскн програм) н **снстем** (серверскн програм) су покренутн. **Продавац** је прнјављен под својом шнфром. Снстем прнказује форму за рад са рачуном. *Учнтане су лнсте: а) Ставка рачуна, б) Услуга*

RB	Naziv	Kolicina	Jedinicna cena	Ukupna cena
1	Banana	15	20.0	300.0
2	Kafa	2	90.0	180.0
3	Sat vremena	1	100.0	100.0

Dodaj stavku

Izmeni stavku

Obrisi stavku

Ukupan iznos:

580.0 (popust je uracunat)

Finalizuj prodaju

Odustani

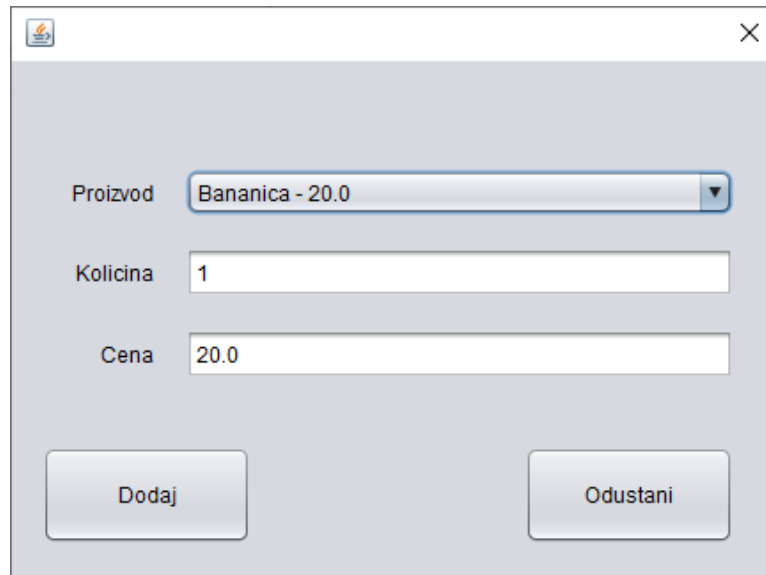
Снлака 13 - Интерфејс за креирање рачуна



### Основни сценарио СК:

1. **Продавац уноси** податке о **рачуну**. (АПУСО)

Опис акције: продавац у прозору за креирање рачуна, користећи се „Додај ставку“ дугметом уноси ставке и њихове количине у рачун. Количина ставки не сме бити мања или једнака нули. Са сваком унетом ставком укупан износ рачуна се аутоматски израчунава и на њега се додаје попуст (ако постоји) и онда се освежава у корисничком интерфејсу.

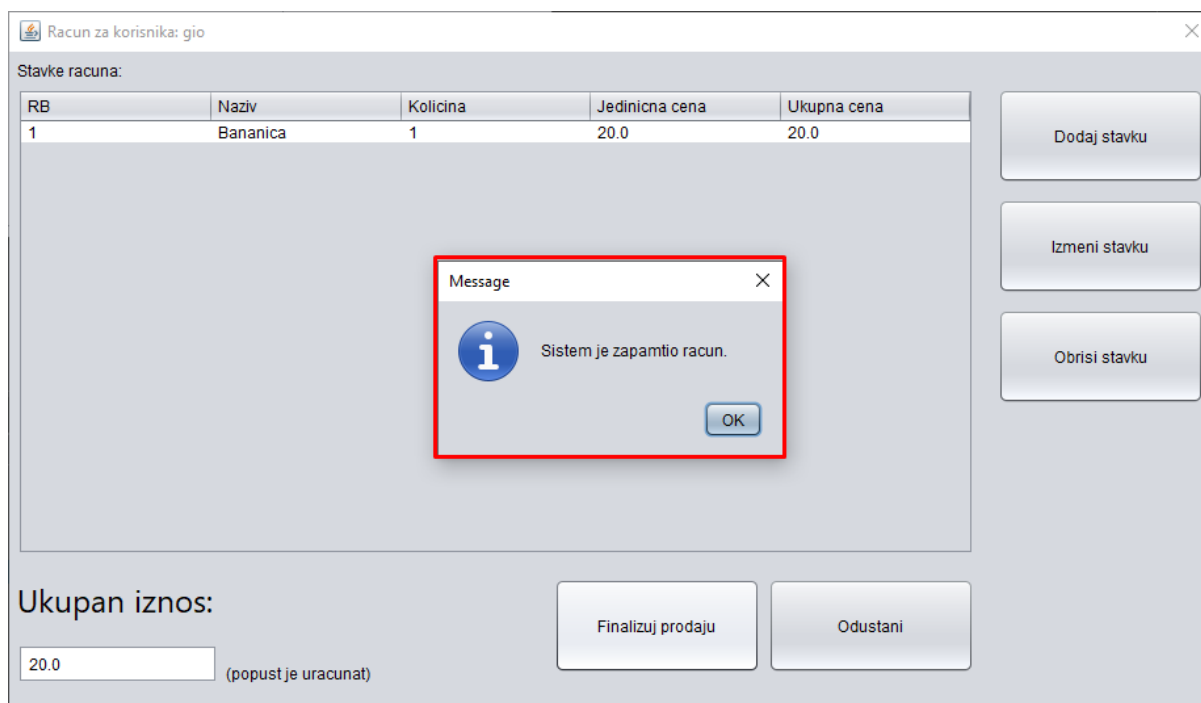


Слика 14 - Интерфејс за додавање ставке

2. **Продавац контролише** да ли је коректно унео податке о **рачуну**. (АНСО)
3. **Продавац позива систем** да запамти податке о **рачуну**. (АПСО)

Опис акције: притиском на дугме „Финализуј продају“ се рачун, заједно са ставкама, убацује у базу података и памти.

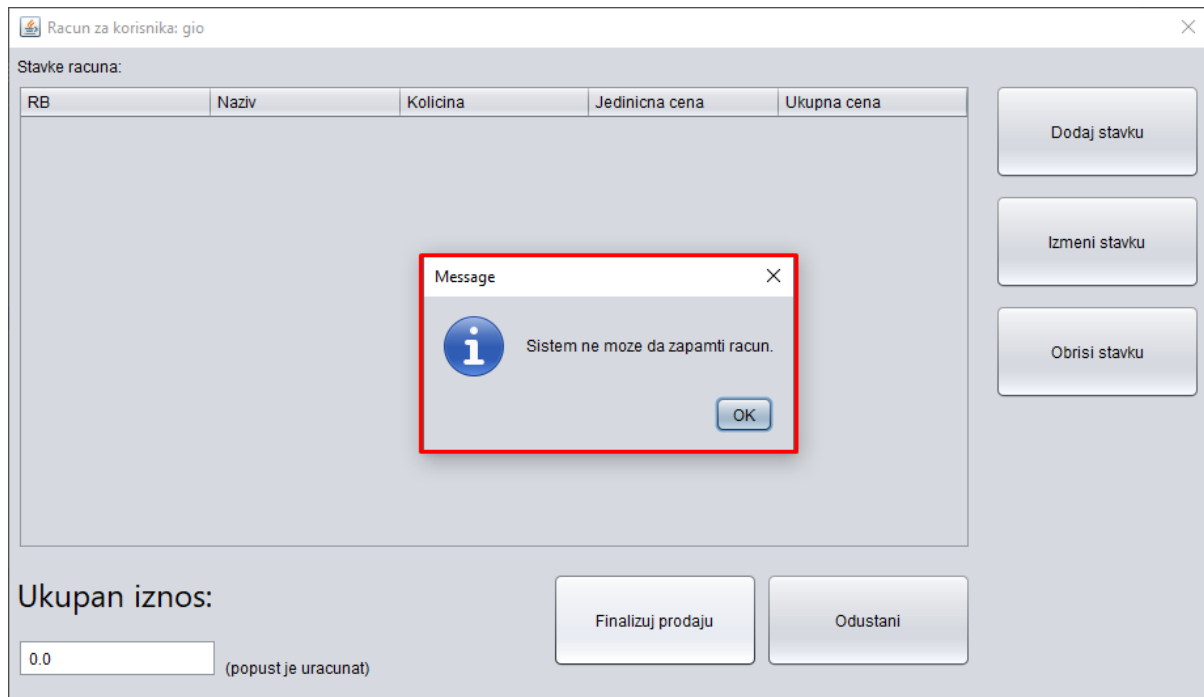
4. **Систем памти** податке о **рачуну**. (СО)
5. **Систем приказује** **продавцу рачун** и поруку: “Систем је запамтио **рачун**.” (ИА)



Слика 15 - Порука након успешног додавања рачуна

### Алтернативна сценарија:

5.1 Уколико **систем** не може да запамти податке о **рачуну** он **приказује** **продавцу** поруку: “**Систем** не може да запамти **рачун**”. (ИА)



Слика 16 - Порука након неуспешног додавања рачуна

## СК2- Претражи рачун

### Назив СК

Претражи рачун

### Актори СК

Продавац

### Учесници СК

Продавац, кориснички интерфејс (клијентски програм) и систем (серверски програм)

**Предуслови:** Кориснички интерфејс (клијентски програм) и систем (серверски програм) су покренути. Продавац је пријављен под својом шифром. Кориснички интерфејс приказује форму за рад са рачуном. На наведеној екранској форми су дефинисани критеријуми, који се односе на: а) Рачун б) Продавац с) Муштерија д) Услуга, који ће да врате листу рачуна.

Prodavac	Musterija	Ukupan iznos	Datum
Ognjen	gio	100.0 DIN	19.9.2025 - 13:11:26
Ognjen	gio	580.0 DIN	19.9.2025 - 19:36:28
Ognjen	gio	20.0 DIN	19.9.2025 - 19:55:49
Ognjen	gio	91.0 DIN	21.9.2025 - 1:42:40
Ognjen	gio	91.0 DIN	21.9.2025 - 13:12:14
Ognjen	gio	91.0 DIN	21.9.2025 - 13:12:17
Ognjen	gio	91.0 DIN	21.9.2025 - 13:12:19
Ognjen	gio	91.0 DIN	21.9.2025 - 13:12:21
Ognjen	gio	147.0 DIN	22.9.2025 - 17:2:42
Ognjen	igrac	14.0 DIN	22.9.2025 - 17:2:33
Ognjen	igrac	14.0 DIN	22.9.2025 - 17:3:28
Ognjen	TestKorisnik	90.0 DIN	22.9.2025 - 18:19:24
Ognjen	TestKorisnik	200.0 DIN	22.9.2025 - 18:55:57
Admin	gio	266.0 DIN	24.9.2025 - 12:38:25
Admin	igrac	189.0 DIN	24.9.2025 - 12:38:43
Admin	TestKorisnik	100.0 DIN	24.9.2025 - 12:38:51

Слика 17 - Почетни интерфејс за претрагу рачуна

### Основни сценарио СК:

1. **Продавац бира** критеријуме на основу којих претражује **рачуне**. (АПУСО)

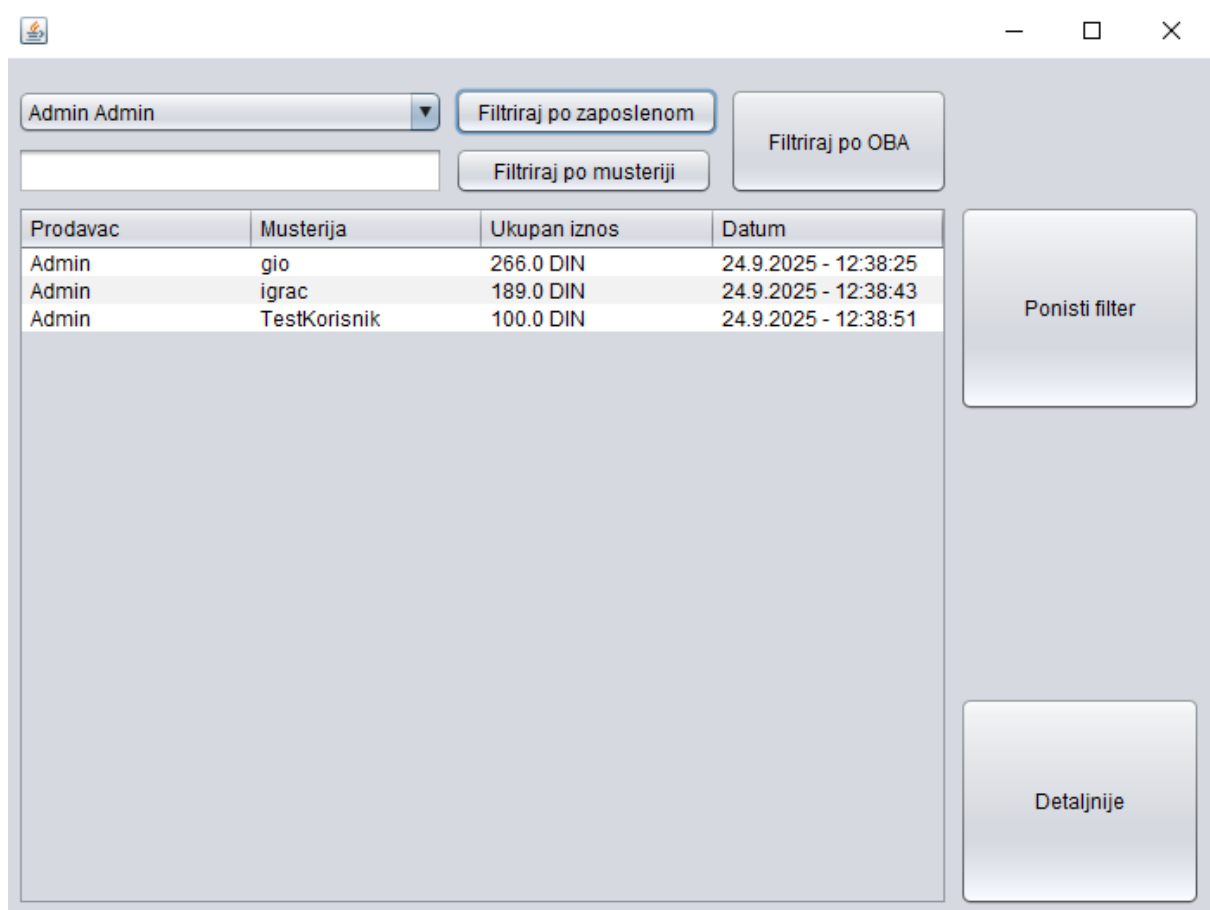
Опис акције: продавац у почетном прозору за претрагу има увид у све креиране рачуне и селекцијом било ког може да види детаљне информације о њему кликом на дугме Детаљније, поред тога постоје два критеријума на основу којих може да их филтрира а то су запослени и муштерија.

2. **Продавац позива систем** да нађе **рачуне** по задатим критеријумима. (АПСО)

Опис акције: продавац кликом на неко од дугмади филтрира рачуне по једном или оба критеријума.

3. **Систем тражи** **рачуне** по задатим критеријумима. (СО)

4. **Систем приказује** **продавцу рачуне**. (ИА)



Слика 18 - Филтрирани рачуни


5. **Продавац бира** **рачун**. (АПУСО)

6. **Продавац позива систем** да нађе **рачун**. (АПСО)

Опис акције: кликом на дугме „Детаљније“ се позива сервер да прикаже форму са детаљнијим информацијама о одабраном рачуну. За ову функцију мора бити одабран неки рачун.

7. **Систем тражи** **рачун**. (СО)

8. **Систем приказује** **продавцу рачун** и поруку: “Систем је нашао **рачун**”. (ИА)



×

Prodavac: Ognjen Pavlović

Kupac: gio - (0% popusta)

Datum i vreme: 19.9.2025 - 19:36:28

Ukupan iznos: 580.0 DIN

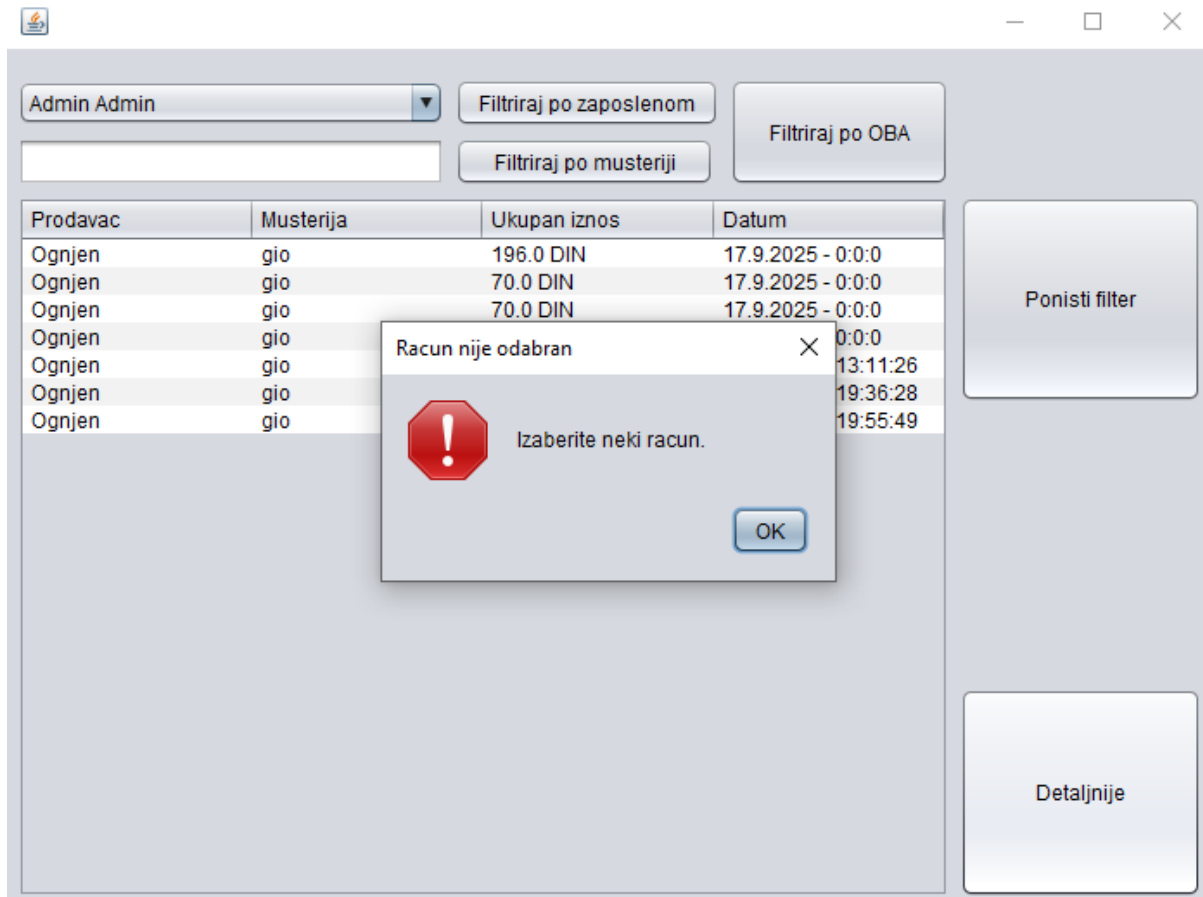
RB	Naziv	Kolicina	Jedinicna cena	Ukupna cena
1	Bananica	15	20.0	300.0
2	Kafa	2	90.0	180.0
3	Sat vremena	1	100.0	100.0

Слика 19 - Детаљне информације о одабраном рачуну

### Алтернативна сценарија:

4.1 Уколико **систем** не може да нађе **рачуне** он **приказује** **продавцу** празну табелу **рачуна**. Прекида се извршење сценарија. (ИА)

8.1 Уколико **систем** не може да нађе **рачун** он **приказује** **продавцу** поруку: “Изаберите неки **рачун**”. Прекида се извршење сценарија.(ИА)



Слика 20 - Порука грешке приликом детаљнијег приказа

### СКЗ- Промени рачун

#### Назив СК

Промени рачун

#### Актори СК

Продавац

#### Учесници СК

Продавац, кориснички интерфејс (клијентски програм) и систем (серверски програм)

**Предуслови:** Кориснички интерфејс (клијентски програм) и систем (серверски програм) су покренути. Продавац је пријављен под својом шифром. Кориснички интерфејс приказује форму за рад са рачуном. На наведеној екранској форми су дефинисани критеријуми, који се односе на: а) Рачун б) Продавац с) Муштерија д) Услуга, који ће да врате листу рачуна. Учитане су листе: а) Продавац б) Муштерија с) Услуга

Prodavac:	<input type="text" value="Ognjen Pavlovic"/>
Kupac:	<input type="text" value="TestKorisnik - (0% popusta)"/>
Datum i vreme:	<input type="text" value="22.9.2025 - 18:55:57"/>
Ukupan iznos:	<input type="text" value="180.0 DIN"/>

Naziv	Kolicina	Jedinicna cena	Ukupna cena
Kafa	2	90.0	180.0

Слика 21 - Почетни интерфејс за измену рачуна

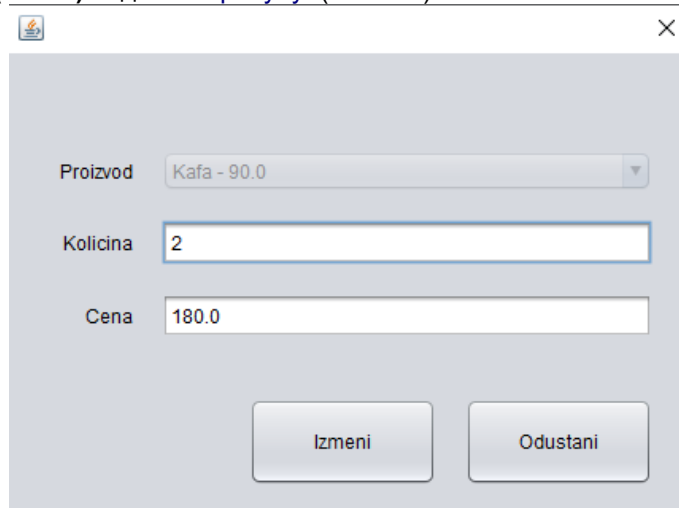


### Основни сценарио СК:

1. **Продавац бира** критеријуме на основу којих претражује **рачуне**. (АПУСО)
2. **Продавац позива систем** да нађе **рачуне** по задатим критеријумима. (АПСО)
3. **Систем тражи рачуне** по задатим критеријумима. (СО)
4. **Систем приказује продавцу рачуне**. (ИА)
5. **Продавац бира рачун**. (АПУСО)
6. **Продавац позива систем** да нађе **рачун**. (АПСО)
7. **Систем тражи рачун**. (СО)
8. **Систем приказује продавцу рачун**. (ИА)

Опис акције: систем продавцу показује све детаље о рачуну са могућности да продавац измени, дода и обрише ставке тог рачуна.

9. **Продавац уноси (мења)** податке о **рачуну**. (АПУСО)




Слика 22 - Поље за измену ставке

10. **Продавац контролише** да ли је коректно унео податке о **рачуну**. (АНСО)
11. **Продавац позива систем** да запамти податке о **рачуну**. (АПСО)
12. **Систем памти** податке о **рачуну**. (СО)
13. **Систем приказује продавцу рачун** и поруку: “Систем је запамтио рачун.” (ИА)

Naziv	Kolicina	Jedinicna cena	Ukupna cena
Banatica	10	20.0	200.0

Message

 Sistem je zapamtio racun.

OK

Dodaj stavku

Obrisi stavku

Izmeni stavku

POTVRDI

ODUSTANI

Слика 23 - Порука након успешне измене рачуна

### Алтернативна сценарија:

4.1 Уколико **систем** не може да нађе **рачуне** он **приказује** **продавцу** празну табелу. Прекида се извршење сценарија. (ИА)

8.1 Уколико **систем** не може да нађе **рачун** он **приказује** **продавцу** поруку: “Изаберите неки **рачун**”.(ИА)

13.1 Уколико **систем** не може да запамти податке о **рачуну** он **приказује** **продавцу** поруку: “**Систем** не може да запамти **рачун**”. (ИА)

Prodavac: Ognjen Pavlovic


Kupac: TestKorisnik - (0% popusta)

Datum i vreme: 22.9.2025 - 18:55:57

Ukupan iznos: 0.0

Naziv	Kolicina	Jedinicna cena	Ukupna cena
-------	----------	----------------	-------------

Message

 Sistem ne moze da zapamti racun.

OK

Dodaj stavku    Obrisi stavku    Izmeni stavku    POTVRDI    ODUSTANI

Слика 24 - Порука грешке при измени рачуна (рачун је празан)

## СК4- Убаци муштерија

### Назив СК

Убаци муштерија

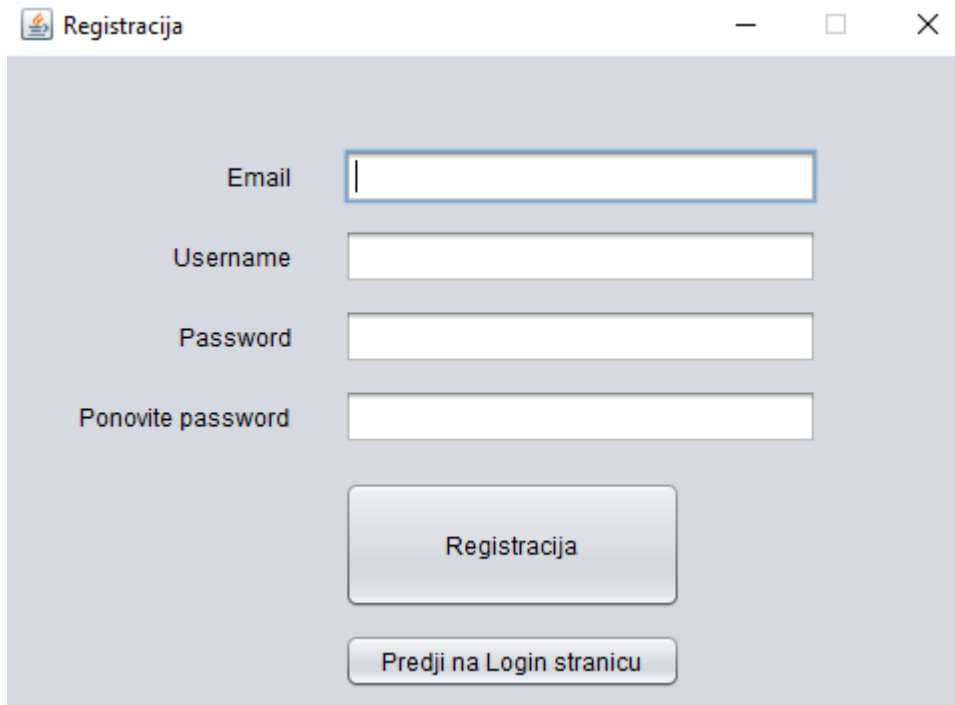
### Актори СК

Муштерија

### Учесници СК

Муштерија, кориснички интерфејс (клијентски програм) и систем (серверски програм)

**Предуслови:** Кориснички интерфејс (клијентски програм) и систем (серверски програм) су покренути. Систем приказује форму за рад са муштеријом. *Учитане су листе: а) Муштерија.*



The image shows a software window titled "Registracija". Inside the window, there is a registration form with the following elements:

- Four text input fields labeled "Email", "Username", "Password", and "Ponovite password".
- A button labeled "Registracija" located below the password fields.
- A button labeled "Predji na Login stranicu" located below the registration button.

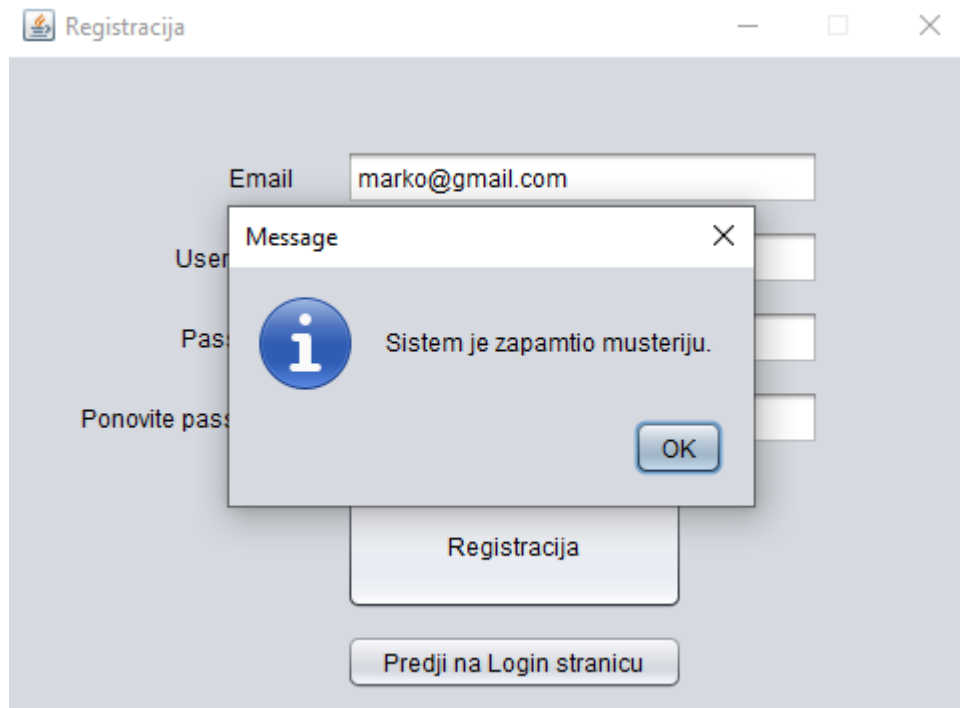
Слика 25 - Основни интерфејс за регистрацију муштерије

### Основни сценарио СК:

1. **Муштерија уноси** податке о **муштерији**. (АПУСО)

Опис акције: муштерија уноси тражене креденцијале који морају да задовоље предефинисане критеријуме као што су уникатност корисничког имена и е-поште или дужина шифре.

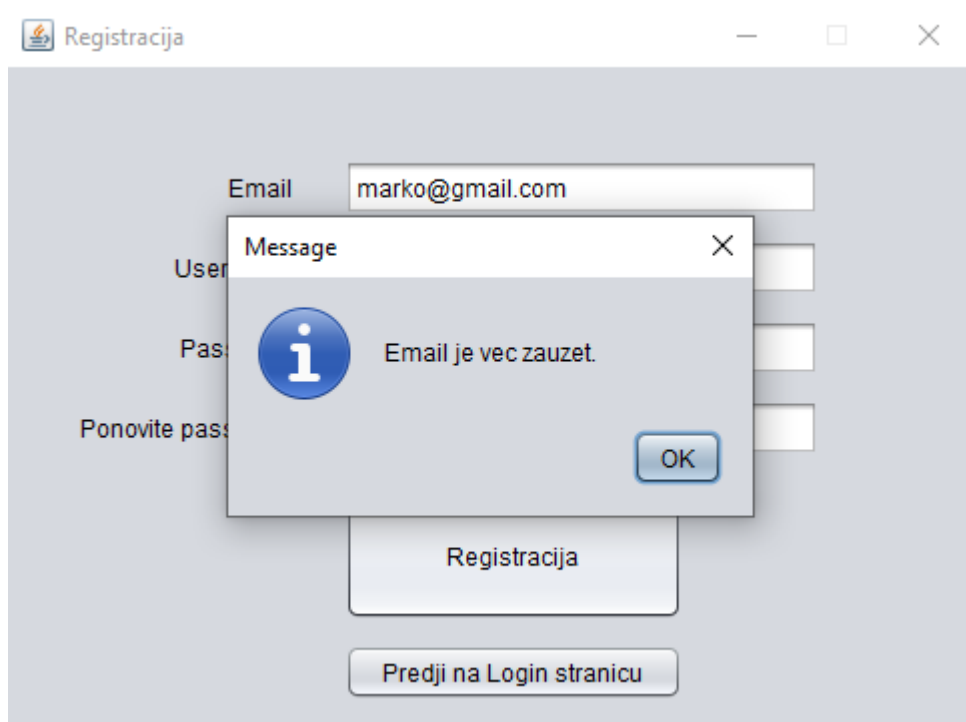
2. **Муштерија контролише** да ли је коректно унео податке о **муштерији**. (АНСО)
3. **Муштерија позива систем** да запамти податке о **муштерији**. (АПСО)
4. **Систем памти** податке о **муштерији**. (СО)
5. **Систем приказује муштерији** поруку: “Систем је запамтио муштерију.” (ИА)



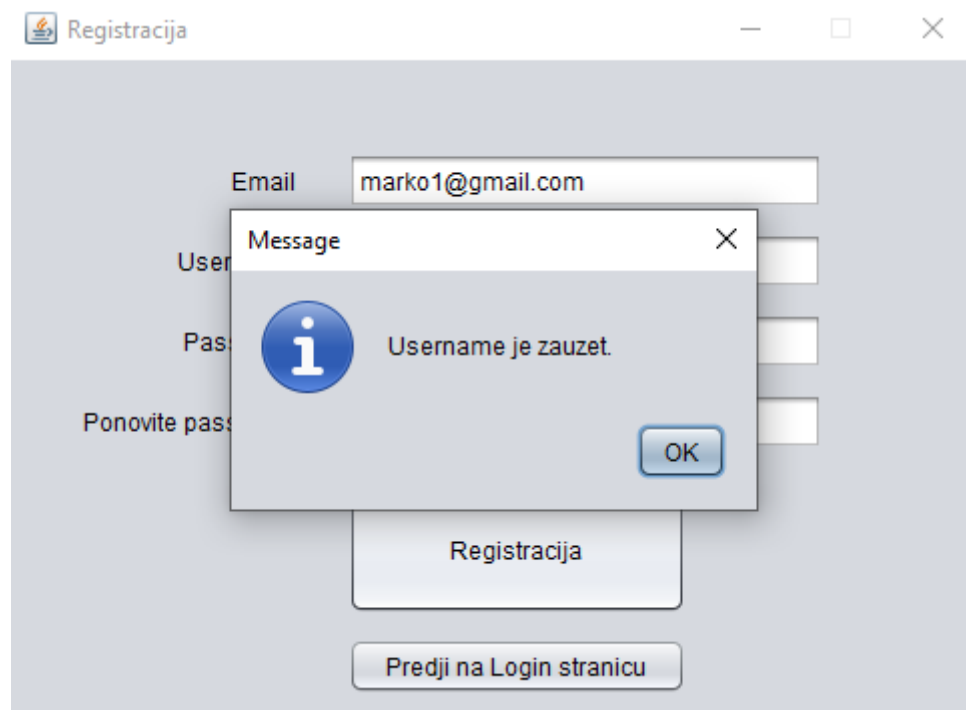
Слика 26 - Порука након успешне регистрације

### Алтернативна сценарија:

5.1 Уколико **систем** не може да запамти податке о **муштерији** он **приказује** **муштерији** поруку о детаљнијем разлогу неуспешног убацавања. (ИА)



Слика 27 - Порука грешке услед заузете е-поште



Слика 28 - Порука грешке услед заузетог корисничког имена

## СК6- Промени муштерија

### Назив СК

Промени муштерија

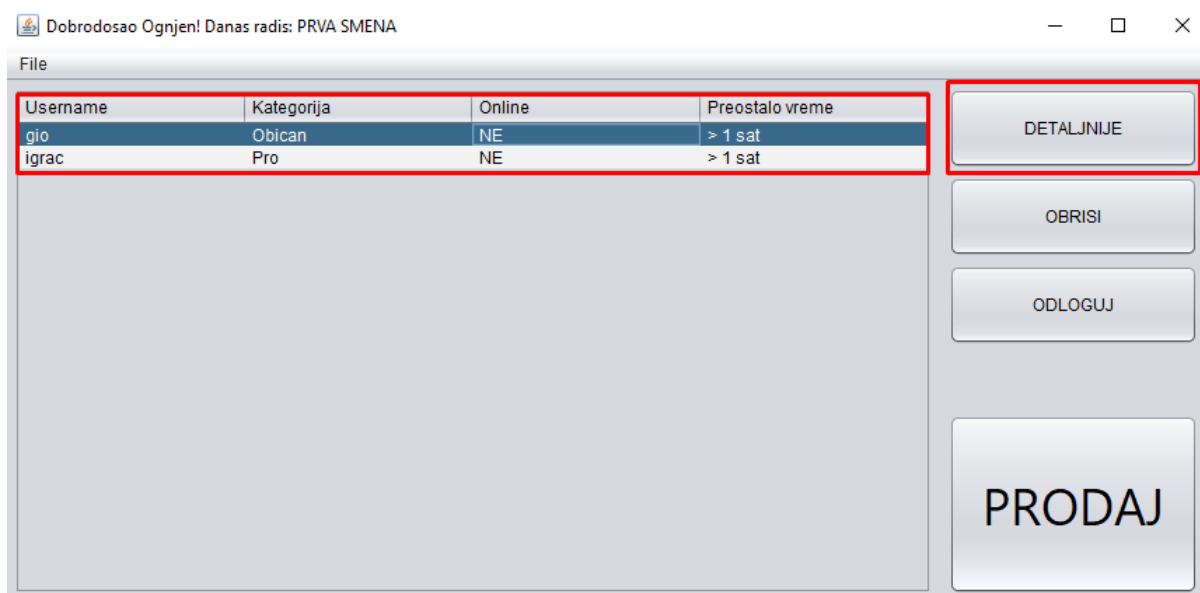
### Актери СК

Продавац

### Учесници СК

Продавац, кориснички интерфејс (клијентски програм) и систем (серверски програм)

**Предуслови:** Кориснички интерфејс (клијентски програм) и систем (серверски програм) су покренути. Продавац је пријављен под својом шифром. Кориснички интерфејс приказује форму за рад са муштеријом. На наведеној екранској форми су дефинисани критеријуми, који се односе на: а) Муштерија б) Категорија муштерије, који ће да врате листу муштерија. Учитане су листе: а) Категорија муштерије



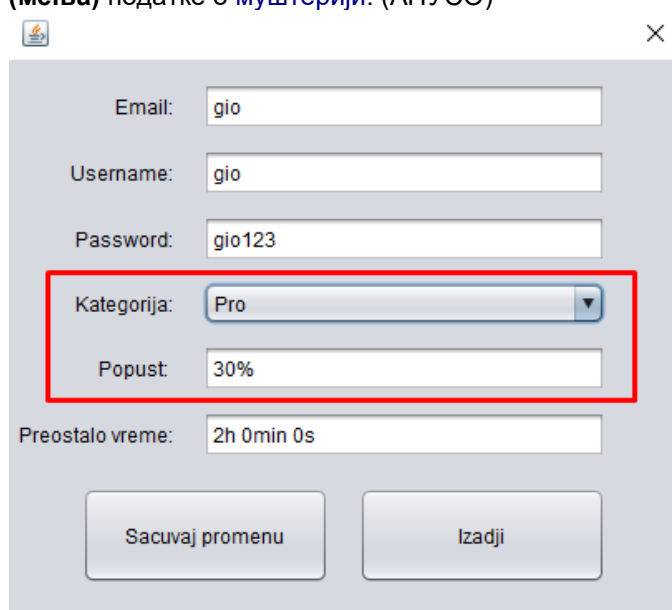
Слика 29 - Основни интерфејс за избор муштерије

### Основни сценарио СК:

1. **Продавац** бира муштерију. (АПУСО)
2. **Продавац** позива систем да нађе муштерију. (АПСО)
3. **Систем** тражи муштерију. (СО)
4. **Систем** приказује продавцу муштерију. (ИА)

Опис акције: продавац види информације о муштерији као и његову категорију (од које зависи попуст) коју може да измени и сачува у базу.

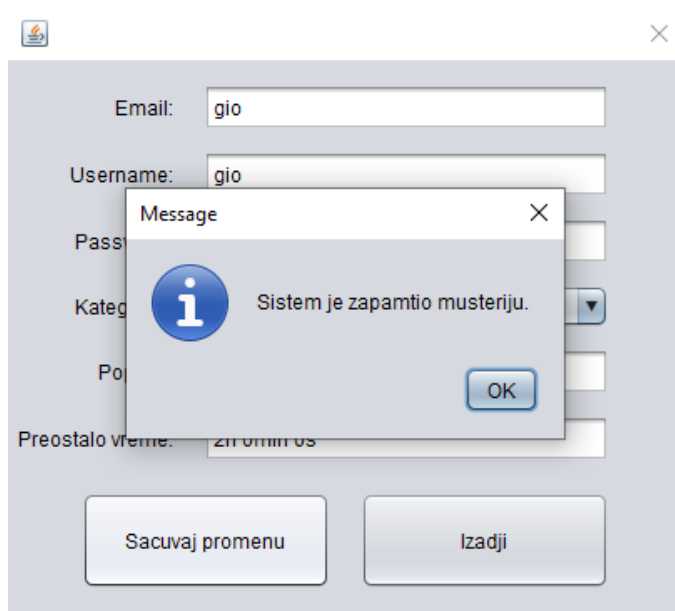
5. **Продавац** уноси (мења) податке о муштерији. (АПУСО)



The screenshot shows a web form for editing a customer's profile. The form has the following fields: Email (gio), Username (gio), Password (gio123), Kategorija (Pro), Popust (30%), and Preostalo vreme (2h 0min 0s). The 'Kategorija' and 'Popust' fields are highlighted with a red rectangle. At the bottom, there are two buttons: 'Sacuvaj promenu' and 'Izadji'.

Слика 30 - Форма за детаљнији приказ и промену муштерије

6. **Продавац** контролише да ли је коректно унео податке о муштерији. (АНСО)
7. **Продавац** позива систем да запамти податке о муштерији. (АПСО)
8. **Систем** памти податке о муштерији. (СО)
9. **Систем** приказује продавцу муштерију и поруку: "Систем је запамтио муштерију." (ИА)

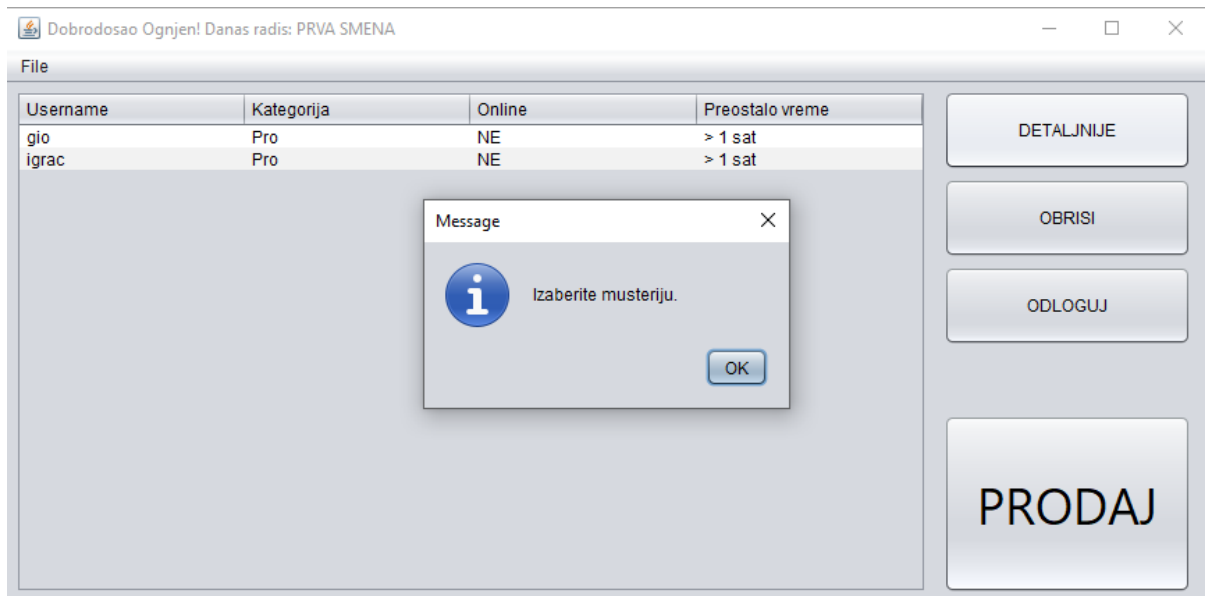


The screenshot shows the same user profile form as in Slika 30, but with a confirmation message overlay. The message box has a title 'Message' and contains an information icon and the text 'Sistem je zapamtio musteriju.' with an 'OK' button. The background form is partially obscured by the message box.

Слика 31 - Форма потврде

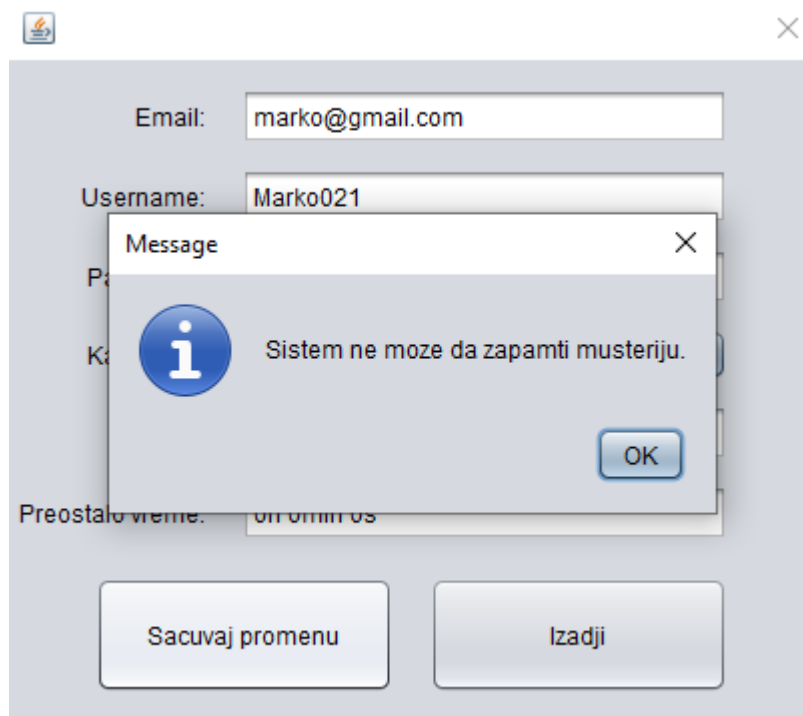
### Алтернативна сценарија:

4.1 Уколико **систем** не може да нађе **муштерију** он **приказује** **продавцу** поруку: “Изаберите **муштерију**”. Прекида се извршење сценарија. (ИА)



Слика 32 - Порука грешке уколико није одабран муштерија

9.1 Уколико систем не може да запамти податке о муштерији он **приказује** продавцу поруку: “Систем не може да запамти муштерију”. (ИА)



Слика 33 - Порука грешке уколико систем не може да запамти муштерију



## СК7- Обриши муштерија

### Назив СК

Обриши муштерија

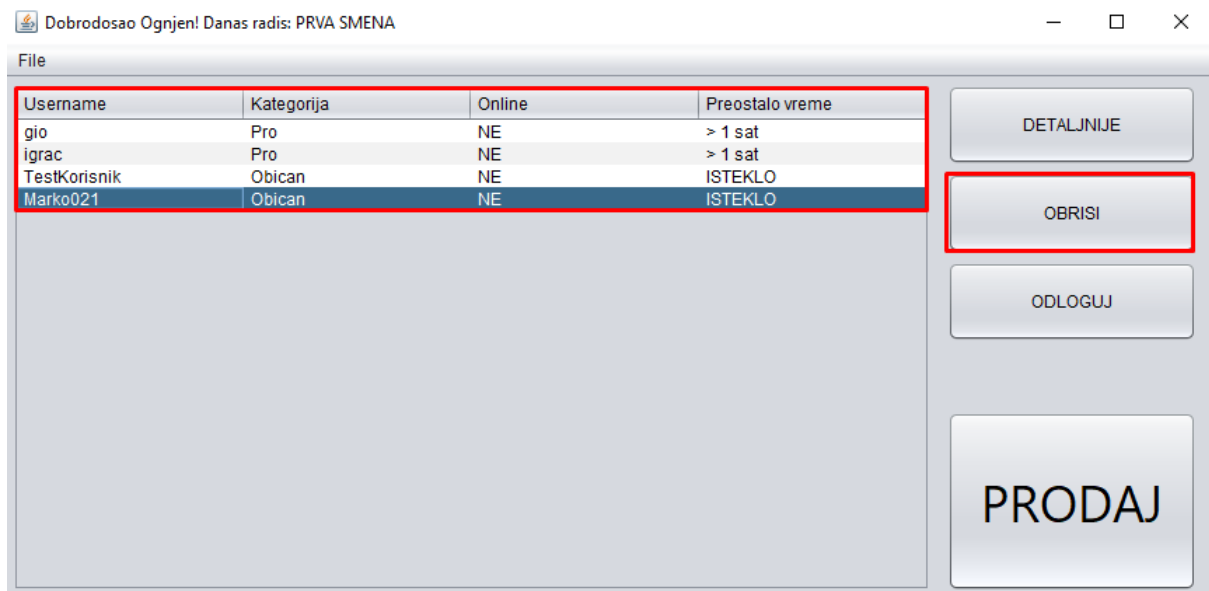
### Актори СК

Продавац

### Учесници СК

Продавац, кориснички интерфејс (клијентски програм) и систем (серверски програм)

**Предуслови:** Кориснички интерфејс (клијентски програм) и систем (серверски програм) су покренути. Продавац је пријављен под својом шифром. Кориснички интерфејс приказује форму за рад са муштеријом. На наведеној екранској форми су дефинисани критеријуми, који се односе на: а) Муштерија б) Категорија муштерије, који ће да врате листу муштерија.



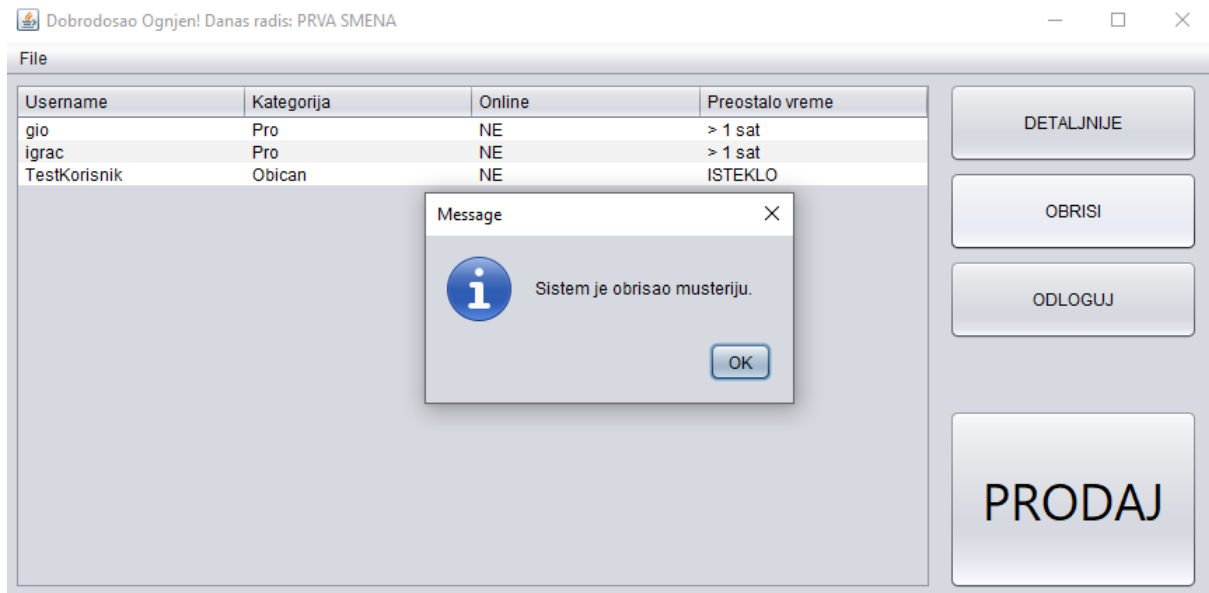
Слика 34 - Основни интерфејс за брисање муштерије

### Основни сценарио СК:

1. **Продавац бира муштерију.** (АПУСО)
2. **Продавац контролише** да ли је изабрао коректног **муштерију.** (АНСО)
3. **Продавац позива систем** да обрише **муштерију.** (АПСО)

Опис акције: изабрани муштерија се брише из базе уколико није пријављен у тренутку извршавања ове акције. За тај случај погледати 5.1 алтернативни сценарио.

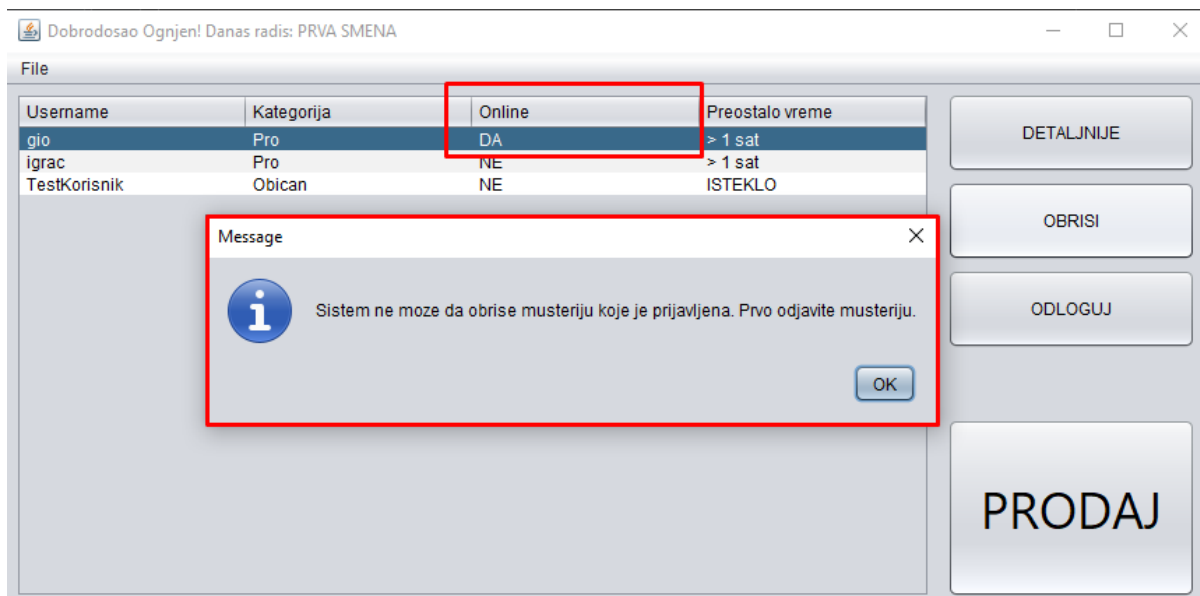
4. **Систем брише муштерију.** (СО)
5. **Систем приказује продавцу** поруку: “**Систем** је обрисао **муштерију.**” (ИА)



Слика 35 - Порука након успешног брисања одабраног муштерије

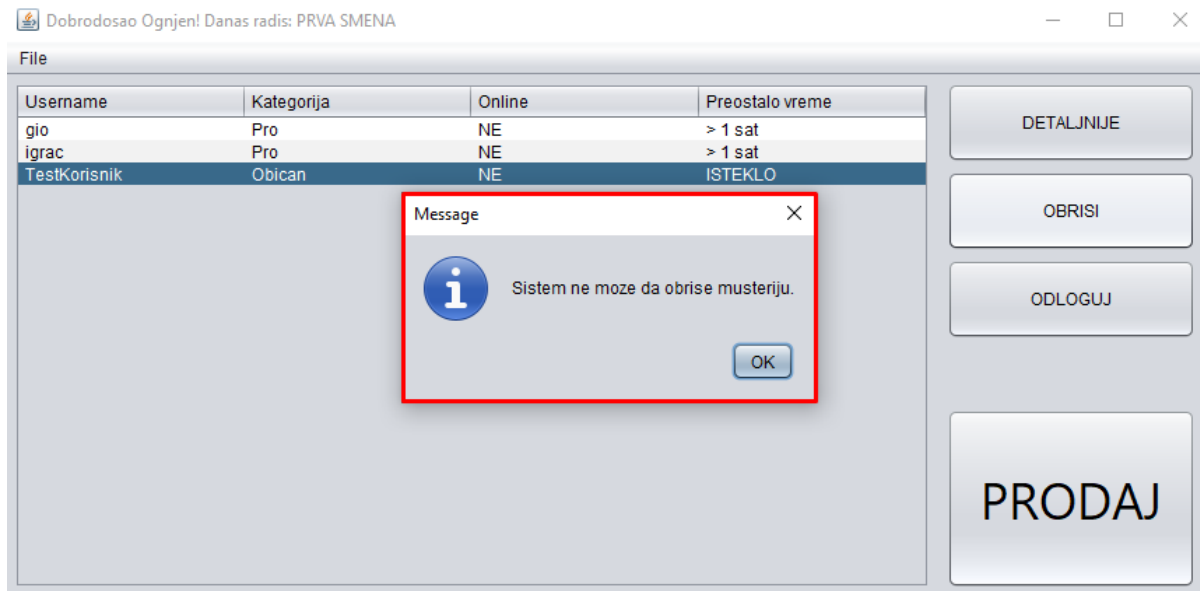
### Алтернативна сценарија:

5.1 Уколико **систем** не може да обрише **муштерију** јер је и даље пријављен он **приказује** **продавцу** поруку: “**Систем** не може да обрише **муштерију** која је пријављена. Прво одјавите **муштерију**”. (ИА)



Слика 36 - Порука након неуспешног брисања већ пријављене муштерије

5.2 Уколико **систем** не може да обрише **муштерију** он **приказује** **продавцу** поруку: “**Систем** не може да обрише **муштерију**”. (ИА)



Слика 37 - Порука након неуспешног брисања одабраног муштерије

## СК8- Пријави муштерија

### Назив СК

Пријави муштерија

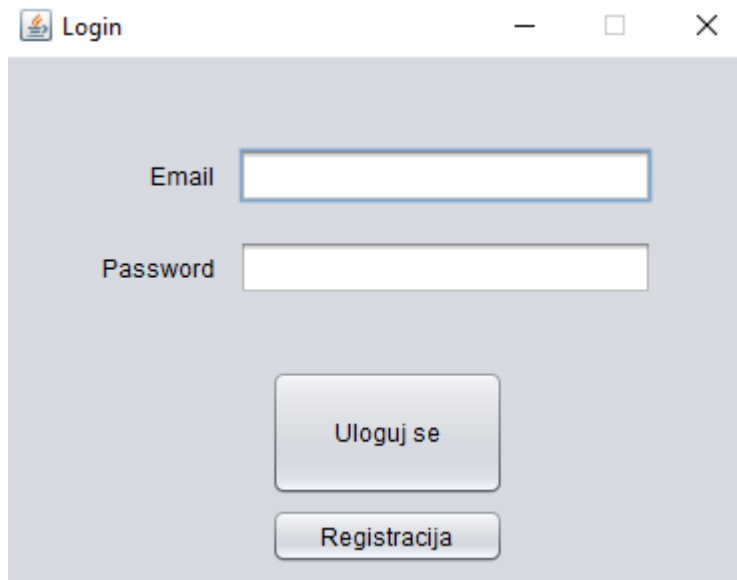
### Актори СК

Муштерија

### Учесници СК

Муштерија, кориснички интерфејс (клијентски програм) и систем (серверски програм)

**Предуслови:** Кориснички интерфејс (клијентски програм) и систем (серверски програм) су покренути. Кориснички интерфејс приказује форму за *пријављивање*.

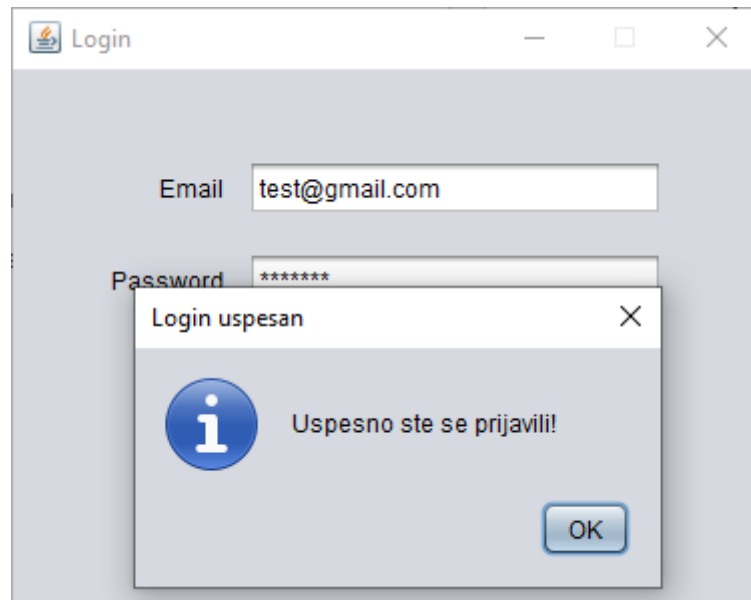


The image shows a window titled "Login" with a standard Windows-style title bar (minimize, maximize, close buttons). The window has a light gray background. It contains two text input fields: the first is labeled "Email" and the second is labeled "Password". Below the "Password" field, there are two buttons: "Uloguj se" (Login) and "Registracija" (Registration). The buttons are rectangular with a slight gradient and rounded corners.

Слика 38 - Основни интерфејс за пријаву муштерије

### Основни сценарио СК:

1. **Муштерија уноси** е-пошту и шифру. (АПУСО)
2. **Муштерија контролише** да ли је коректно унео е-пошту и шифру. (АНСО)
3. **Муштерија позива систем** да провери е-пошту и шифру. (АПСО)
4. **Систем проверава** корисничко име и шифру. (СО)
5. **Систем приказује муштерији** поруку: "Успешно сте се пријавили." (ИА)
6. **Кориснички интерфејс позива** главни форму и мени. (КИПГФМ)



Слика 39 - Порука након успешне пријаве на налог Муштерије

#### Алтернативна сценарија:

- 5.3 Уколико **систем** провером установи да корисничка шифра и/или шифра нису исправни он **приказује муштерији** поруку: “Корисничко име и шифра нису исправни”. (ИА)
- 5.4 Уколико **систем** провером установи да је корисник са тим креденцијалима већ пријављен он **приказује муштерији** поруку: “Корисник је већ пријављен на овом налогу”. (ИА)
- 6.1 Уколико **кориснички интерфејс** не може да отвори главну форму и мени **приказује продавцу** поруку: “Не може да се отвори главна форма и мени”. (НПГФМ)

Постуслови: *Отворена главна форма и мени.*

## СК9- Пријави продавац

### Назив СК

Пријави продавац

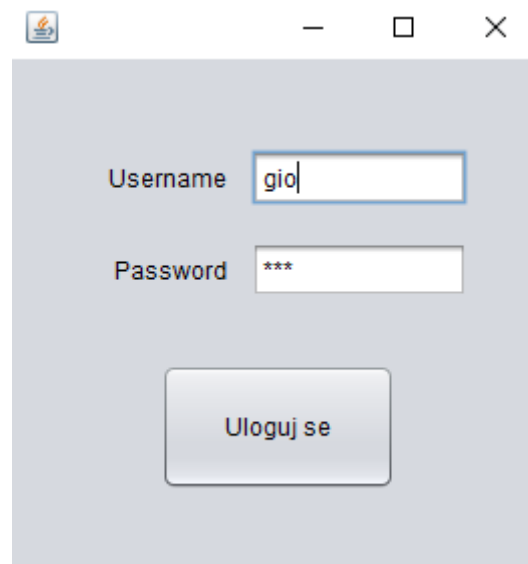
### Актери СК

Продавац

### Учесници СК

Продавац, кориснички интерфејс (клијентски програм) и систем (серверски програм)

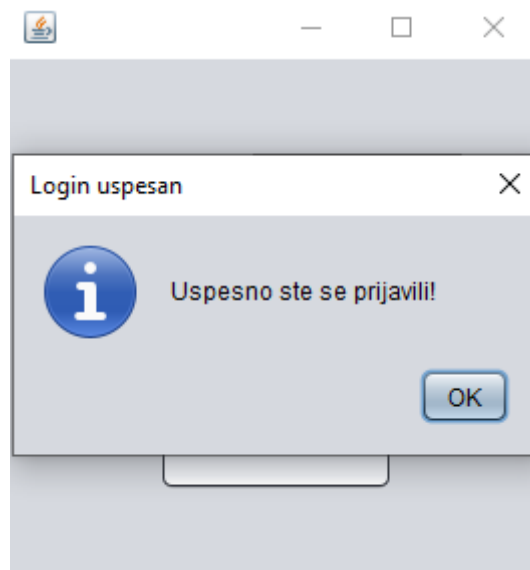
**Предуслови:** Кориснички интерфејс (клијентски програм) и систем (серверски програм) су покренути. Кориснички интерфејс приказује форму за *пријављивање*.



Слика 40 - Основни интерфејс за пријаву продавца

#### Основни сценарио СК:

1. **Продавац уноси** корисничко име и шифру. (АПУСО)
2. **Продавац контролише** да ли је коректно унео корисничко име и шифру. (АНСО)
3. **Продавац позива систем** да провери корисничко име и шифру. (АПСО)
4. **Систем проверава** корисничко име и шифру. (СО)
5. **Систем приказује продавцу** поруку: “ Успешно сте се пријавили.” (ИА)



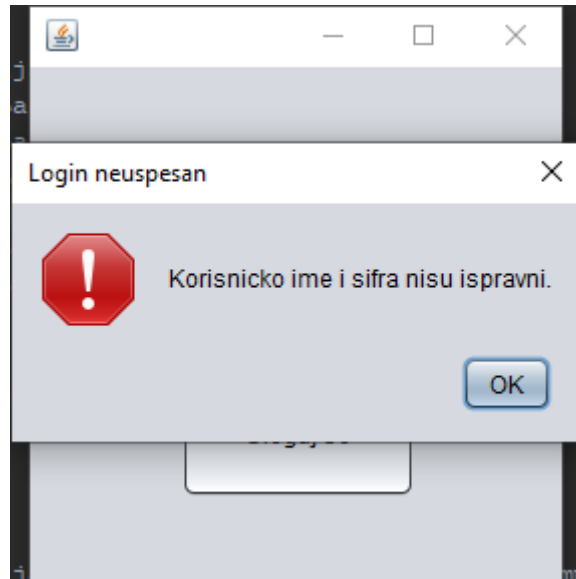
Слика 41 - Порука о успешном пријављивању продавца

6. **Кориснички интерфејс позива** главни форму и мени. (КИПГФМ)  
Опис акције: након успешне пријаве, пре отварања главне форме ће се Продавцу појавити форма о започињању новог термина дежурства. За тај сценарио погледати 22. случај коришћења (Убаца термина дежурства)



**Алтернативна сценарија:**

5.1 Уколико **СИСТЕМ** провером установи да корисничка шифра и/или шифра нису исправни он **приказује** **продавцу** поруку: “*Корисничко име и шифра нису исправни*”. (ИА)



Слика 42 - Порука о неуспешном пријављивању продавца

**Постуслови:** *Отворена главна форма и мени.*

## СК22- Убаџи термин дежурства

### Назив СК

Убаџи термин дежурства

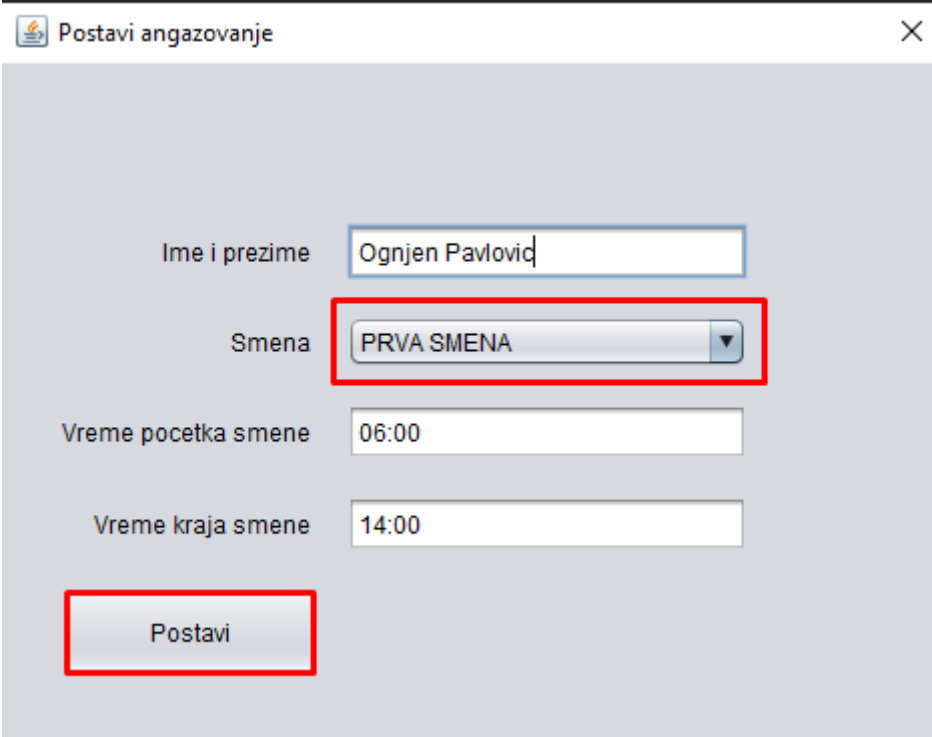
### Актори СК

Продавац

### Учесници СК

Продавац, кориснички интерфејс (клијентски програм) и систем (серверски програм)

**Предуслови:** Кориснички интерфејс (клијентски програм) и систем (серверски програм) су покренути. Продавац је пријављен под својом шифром. Систем приказује форму за рад са термином дежурства.



Postavi angazovanje

Ime i prezime: Ognjen Pavlović

Smena: PRVA SMENA

Vreme pocetka smene: 06:00

Vreme kraja smene: 14:00

Postavi

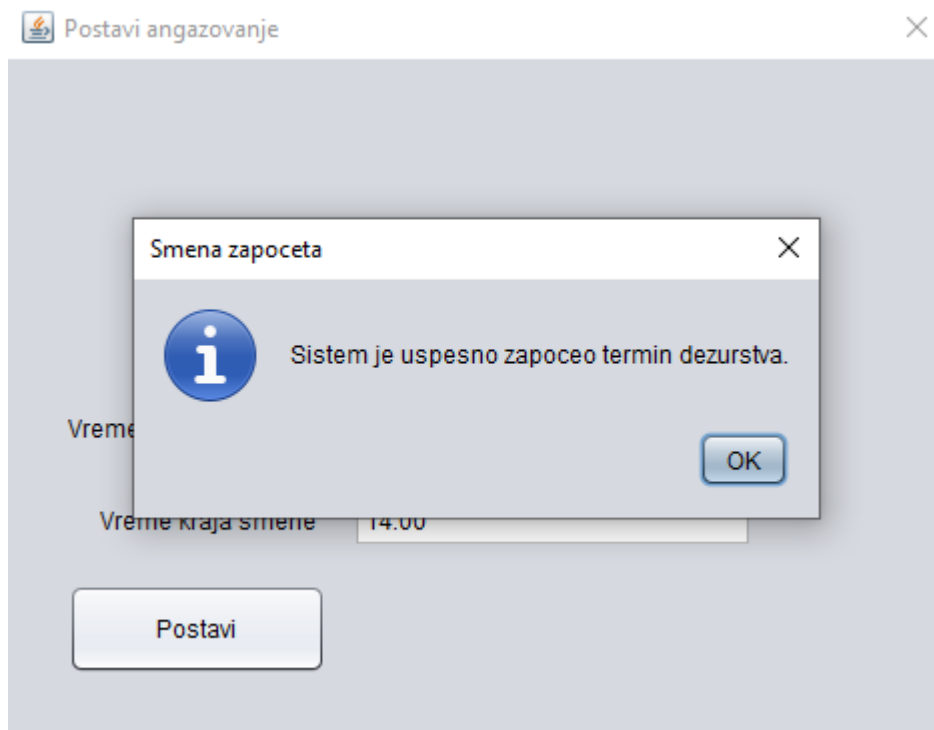
Слика 43 - Основни интерфејс за убаџивање термина дежурства

### Основни сценарио СК:

1. **Продавац уноси** податке о **термину дежурства**. (АПУСО)
2. **Продавац контролише** да ли је коректно унео податке о **термину дежурства**. (АНСО)
3. **Продавац позива систем** да запамти податке о **термину дежурства**. (АПСО)

Опис акције: корисник поставља смену свог термина дежурства на основу које се аутоматски ажурирају поља „Време почетка и време краја смене“.

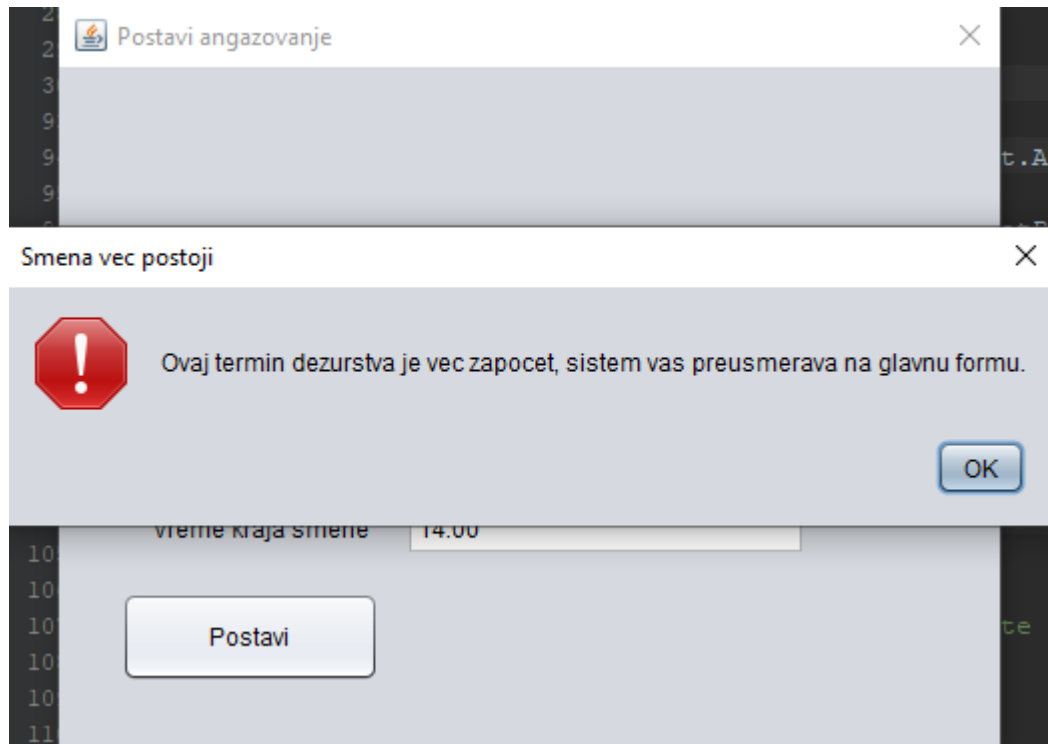
4. **Систем памти** податке о **термину дежурства**. (СО)
5. **Систем приказује продавцу** главну форму и поруку: “**Систем** је успешно започео **темин дежурства**” (ИА)



Слика 44 - Порука о успешном постављању термина дежурства

### Алтернативна сценарија:

5.1 Уколико **систем** не може да запамти податке о **термину дежурства** он **приказује** **продавцу** поруку: “Овај **термин дежурства** је већ започет, **систем** вас преусмерава на главну форму”. (ИА)  
Опис акције: овај алтернативни сценарио је применљив у неким граничним случајевима (нпр. нестанак струје или случајно гашење компјутера) у којима смена мора се наставити. Тада се не памте подаци о термину дежурства а продавцу се даје приступ главној форми.



Слика 45 - Порука о неуспешном бележењу термина дежурства и преусмеравању на главну форму

## 4.2 Пројектовање апликационе логике

За сваки уговор креирамо **системску операцију** која пројектује понашање софтверског система наслеђивањем апстрактне класе *AbstractSystemOperation*.

Ова класа садржи методу *Execute* која као параметар има *DomainObject*. Класе **DodajSO**, **IzmeniSO**, **ObrisiSO** и **VratiSveSo** наслеђују класу *AbstractSystemOperation* и имплементирају њену методу *Execute*. У зависности од тога да ли је потребно извршити операцију додај, измени, обриши или врати све позива се реимплементирана метода *Execute* једне од респективних класа за тип објекта који је прослеђен као параметар. Та метода поново респективно позива једну од метода *update, delete, insert* или *getAll*, генеричког брокера базе података који извршава тражену операцију и враћа сигнал о успешности извршења и објекте одговарајуће класе.

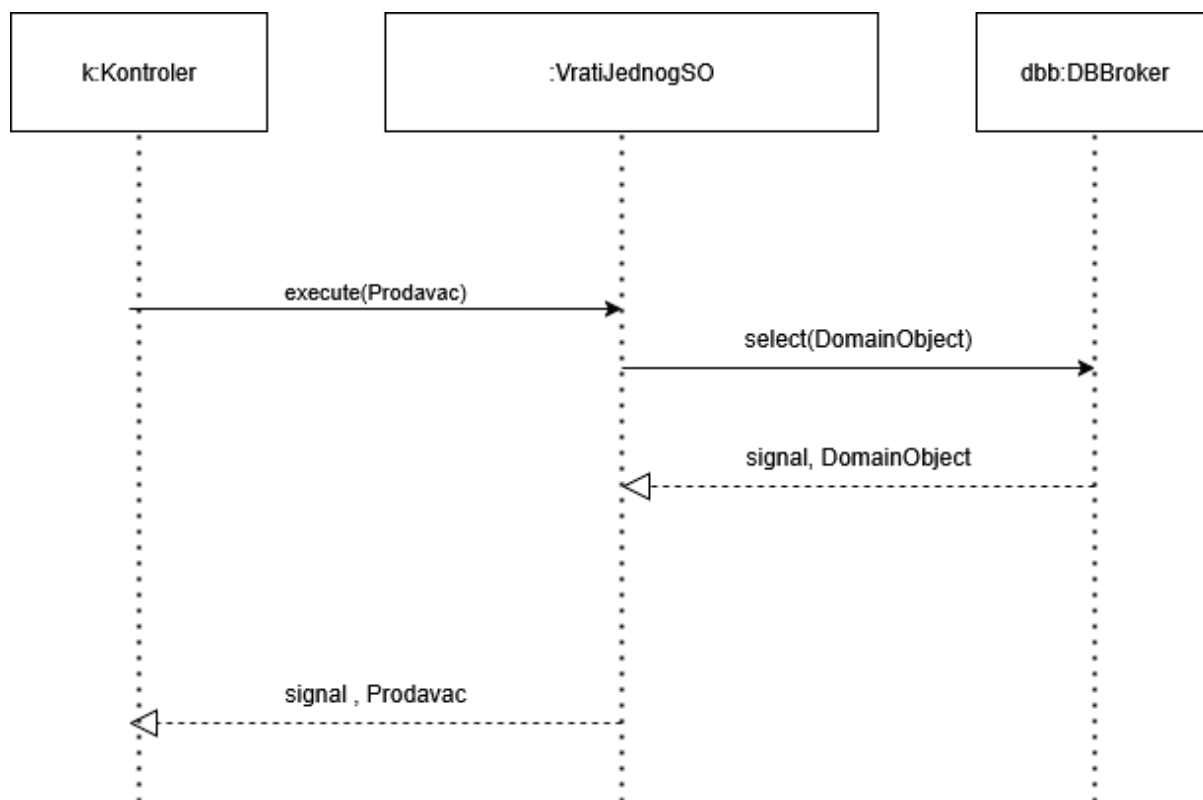
### 1. Уговор UG1: *PrijavaProdavac(korisnickolme, Sifra)*

Операција: *PrijavaProdavac(korisnickolme, Sifra):signal;*

Веза са СК: СК9

Предуслови:

Постуслови: *Корисник је пријављен на систем.*



Слика 46 - Дијаграм ПријаваПродавац

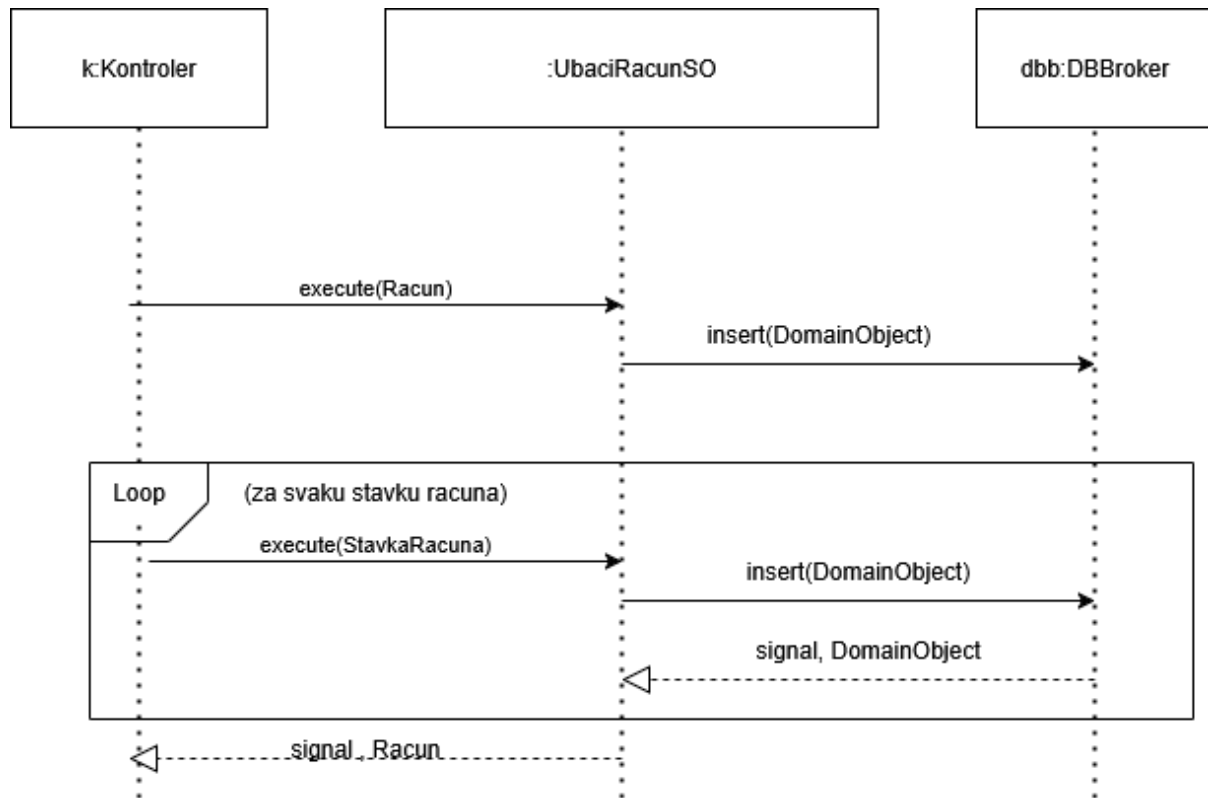
## 2. Уговор UG2: UbaciRacun(Racun, List<StavkaRacuna>)

Операција: UbaciRacun(Racun, List<StavkaRacuna>):signal;

Веза са СК: СК1

Предуслови: Структурна и вредносна ограничење над објектом класе Рачун морају бити задовољена.

Постуслови: Направљен је нови објекат класе Рачун.



Слика 47 - Дијаграм УбациРачун

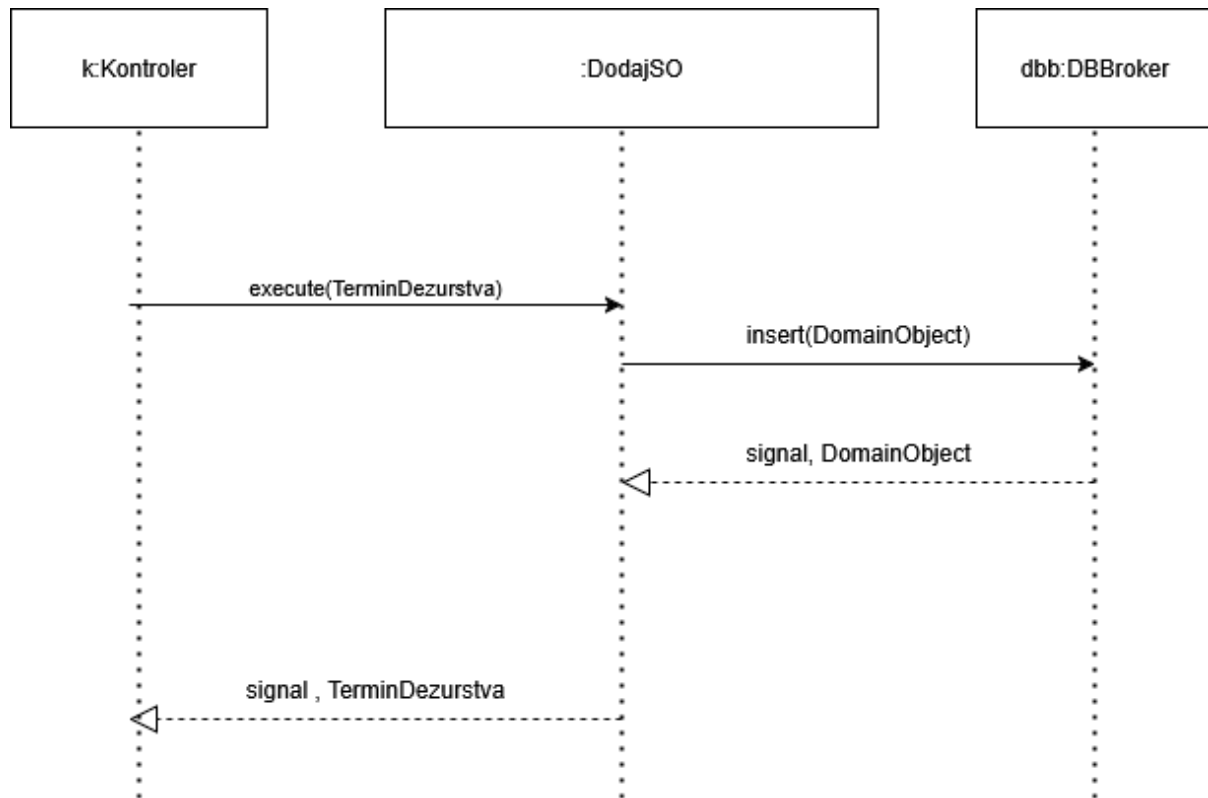
### 3. Уговор UG3: UbaciTerminDezurstva(TerminDezurstva)

Операција: UbaciTerminDezurstva(TerminDezurstva):signal;

Веза са СК: СК22

Предуслови: Структурна и вредносна ограничење над објектом класе ТерминДежурства морају бити задовољена.

Постуслови: Направљен је нови објект класе ТерминДежурства.



Слика 48 - Дијаграм УбациТерминДежурства

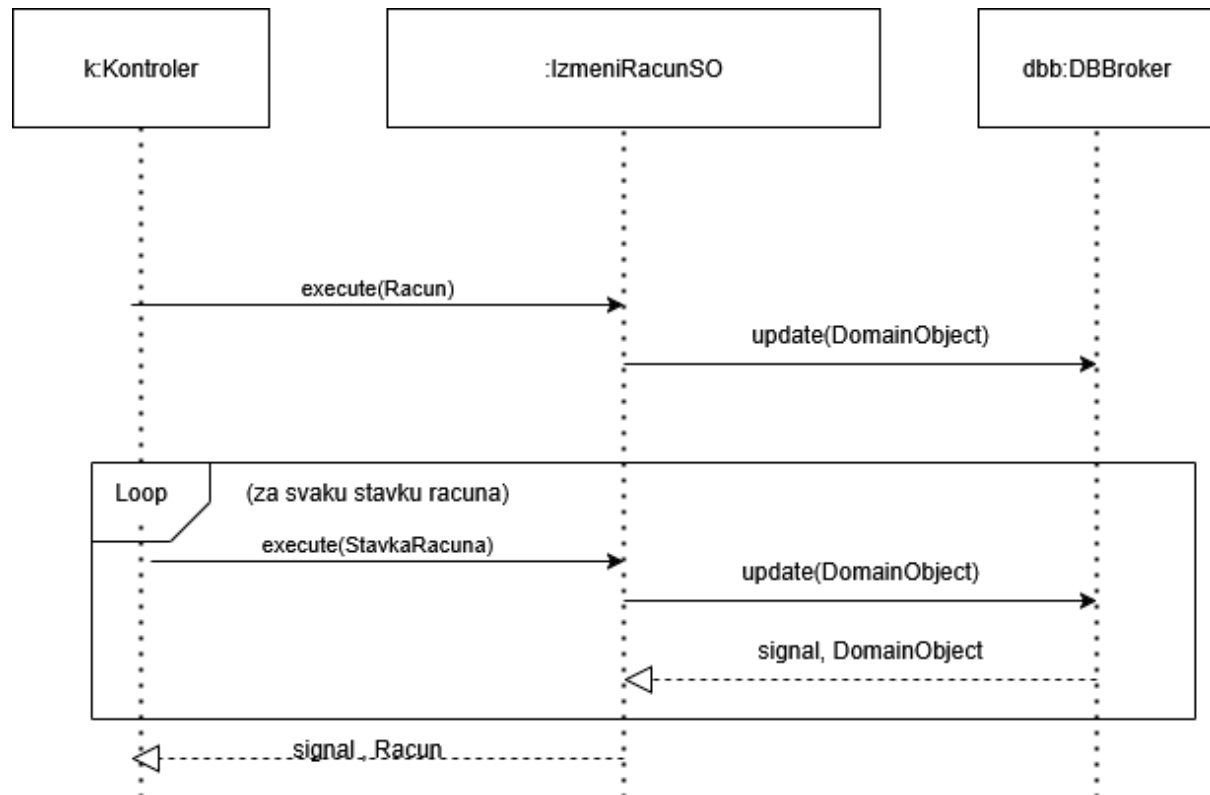
#### 4. Уговор UG4: PromeniRacun(Racun)

Операција: PromeniRacun(Racun):signal;

Веза са СК: СК2, СК3

Предуслови: Структурна и вредносна ограничење над објектом класе Рачун морају бити задовољена.

Постуслови: Објекат класе Рачун је промењен.



Слика 49 - Дијаграм ИзмениРачун



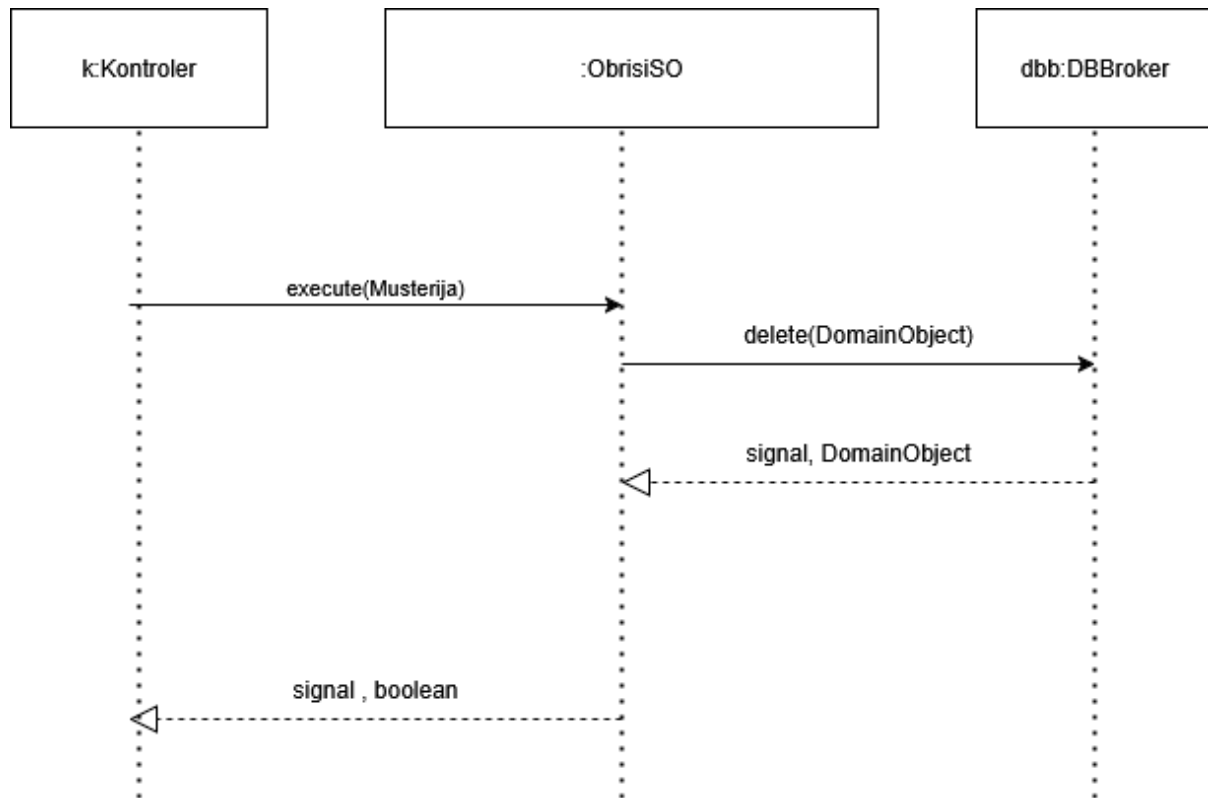
### 5. Уговор UG5: *ObrisiMusterija(Musterija)*

Операција: *ObrisiMusterija(Musterija):signal;*

Веза са СК: СК7

Предуслови Структурна и вредносна ограничење над објектом класе Муштерија морају бити задовољена.

Постуслови: Објекат класе Муштерија је обрисан.



Слика 50 - Дијаграм ОбришиМуштерија

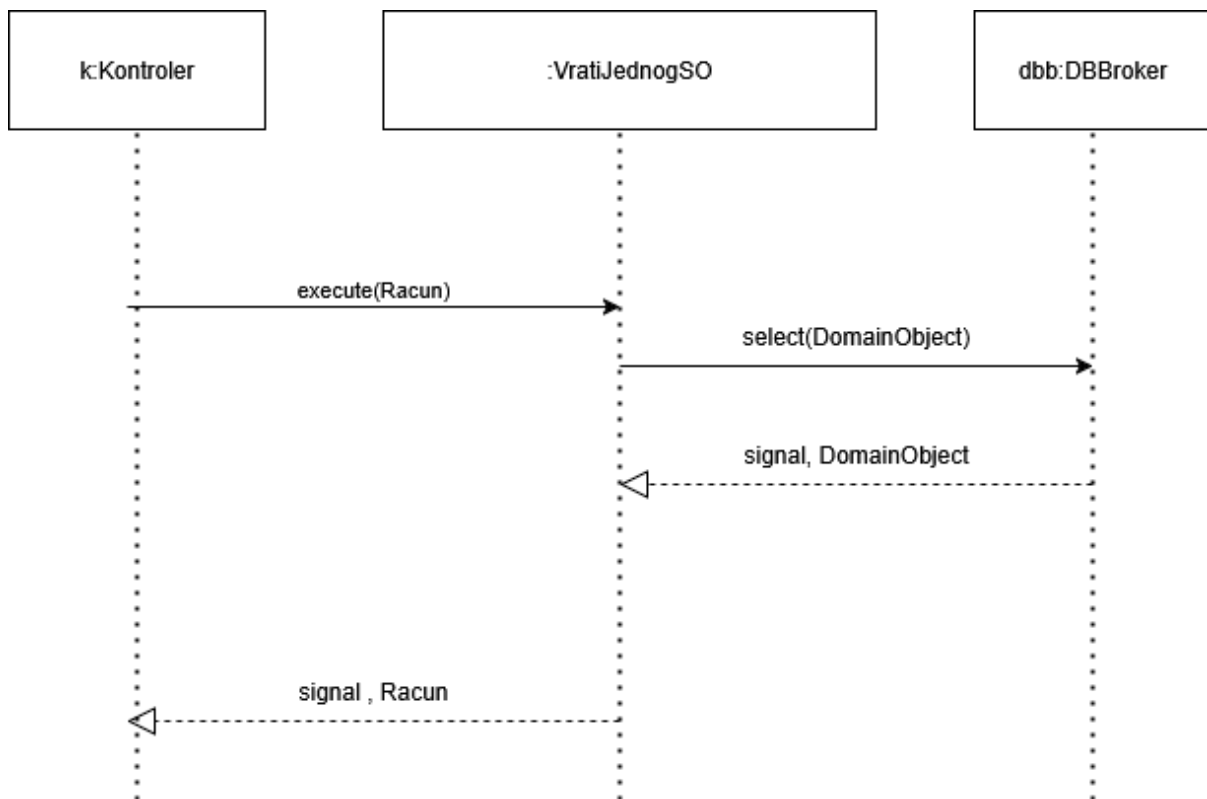
## 6. Уговор UG6: *PretraziRacun(Racun)*

Операција: *PretražiRacun(Racun):signal;*

Веза са СК: СК2

Предуслови:

Постуслови: *Пронађен је тражени објект класе Рачун.*



Слика 51 - Дијаграм ПретражиРачун

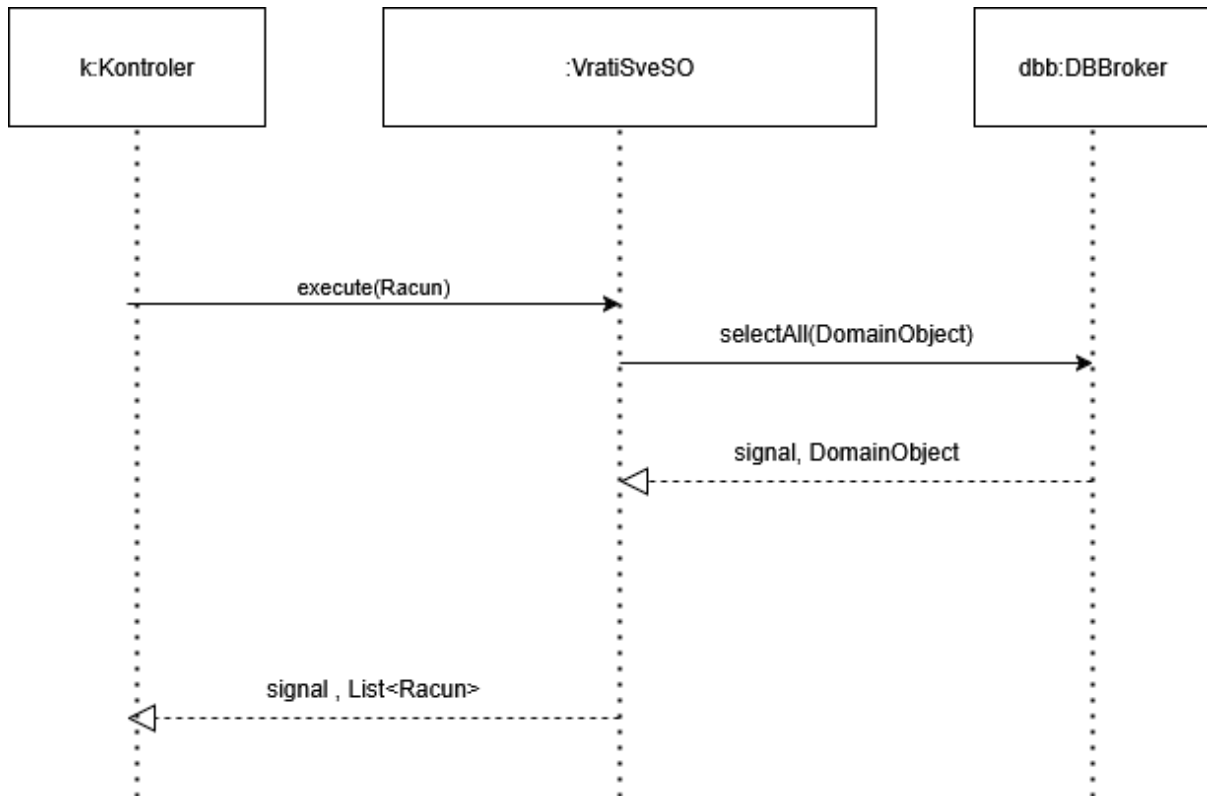
**7. Уговор UG7: vratiListuRacuna(KriterijumRacun, Lista<Racun>)**

**Операција:** vratiListuRacuna (KriterijumRacun, Lista<Racun> ):signal;

**Веза са СК:** СК2

**Предуслови:**

**Постуслови:** Пронађена је листа тражених објеката класе Рачун.



Слика 52 - Дијаграм ВратиЛистуРачуна

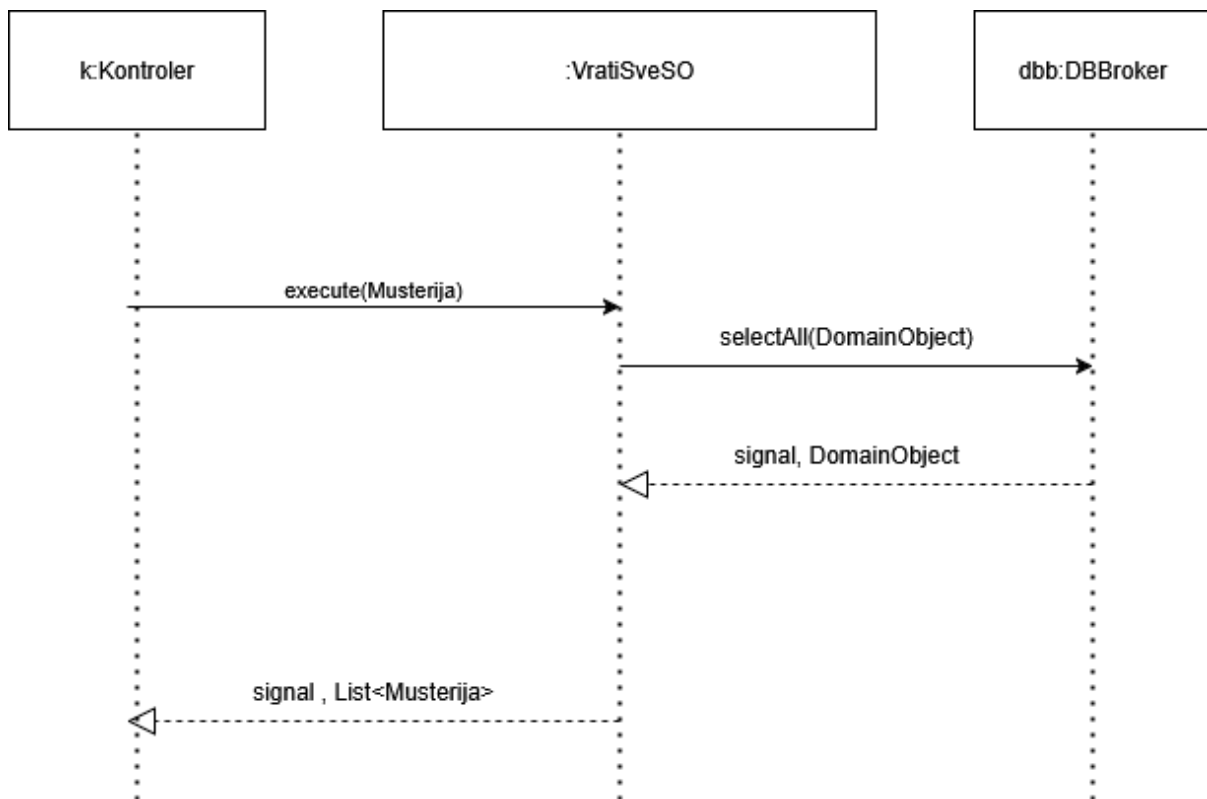
**8. Уговор UG8: vratiListuSviMusterije(Lista<Musterija>)**

**Операција:** vratiListuSviMusterije (Lista<Musterija> ):signal;

**Веза са СК:** СК5

**Предуслови:**

**Постуслови:** Пронађена је листа свих објеката класе Муштерија.



Слика 53 - Дијаграм ВратиЛистуМуштерија

## Проектовање структуре софтверског система (доменске класе)

Софтверске класе структуре:



```
public class Angazovanje implements Serializable, DomainObject<Angazovanje> {

    private Prodavac prodavac;
    private TerminDezurstva termin;
    private LocalDate datum;

    public Angazovanje() {
    }

    public Angazovanje(Prodavac prodavac, TerminDezurstva termin, LocalDate datum) {
        this.prodavac = prodavac;
        this.termin = termin;
        this.datum = datum;
    }
}
```

Слика 54 - Класа Ангажовање



```
public class KategorijaMusterije implements Serializable, DomainObject<KategorijaMusterije> {

    private int id;
    private String naziv;
    private int popust;

    public KategorijaMusterije() {
    }

    public KategorijaMusterije(int id, String naziv, int popust) {
        this.id = id;
        this.naziv = naziv;
        this.popust = popust;
    }
}
```

Слика 55 - Класа КатегоријаМуштерије

```

public class Musterija implements Serializable, DomainObject<Musterija> {

    private int id;
    private String email;
    private String username;
    private String password;
    private KategorijaMusterije kategorijaMusterije;
    private Duration preostaloVreme;

    public Musterija() {
    }

    public Musterija(int id, String email, String username, String password, KategorijaMusterije kategorijaMusterije, Duration preostaloVreme) {
        this.id = id;
        this.email = email;
        this.username = username;
        this.password = password;
        this.kategorijaMusterije = kategorijaMusterije;
        this.preostaloVreme = preostaloVreme;
    }
}

```

Слика 56 - Класа Муштерија

```

public class Prodavac implements Serializable, DomainObject<Prodavac> {


    private int id;
    private String ime;
    private String prezime;
    private String email;
    private String username;
    private String password;

    public Prodavac() {
    }

    public Prodavac(int id, String ime, String prezime, String email, String username, String password) {
        this.id = id;
        this.ime = ime;
        this.prezime = prezime;
        this.email = email;
        this.username = username;
        this.password = password;
    }
}

```

Слика 57 - Класа Продавац



```

public class Racun implements Serializable, DomainObject<Racun> {


    private int id;
    private LocalDateTime datum;
    private double ukupnaCena;
    private Prodavac prodavac;
    private Musterija musterija;

    public Racun() {
    }

    public Racun(int id, LocalDateTime datum, double ukupnaCena, Prodavac prodavac, Musterija musterija) {
        this.id = id;
        this.datum = datum;
        this.ukupnaCena = ukupnaCena;
        this.prodavac = prodavac;
        this.musterija = musterija;
    }
}

```

Слика 58 - Класа Рачун



```


public enum Smene implements Serializable {
    PRVA(LocalTime.of(6, 0), LocalTime.of(14, 0)),
    DRUGA(LocalTime.of(14, 0), LocalTime.of(22, 0)),
    TRECA(LocalTime.of(22, 0), LocalTime.of(6, 0));

    private final LocalTime vremeOd;
    private final LocalTime vremeDo;

    Smene(LocalTime vremeOd, LocalTime vremeDo) {
        this.vremeOd = vremeOd;
        this.vremeDo = vremeDo;
    }
}

```

Слика 59 - Енум Смене



```

public class StavkaRacuna implements Serializable, DomainObject<StavkaRacuna> {

    private int rb;
    private Racun racun;
    private int kolicina;
    private double cenaStavke;
    private double jedinicnaCena;
    private Usluga usluga;

    public StavkaRacuna() {
    }

    public StavkaRacuna(int rb, Racun racun, int kolicina, double cenaStavke, double jedinicnaCena, Usluga usluga) {
        this.rb = rb;
        this.racun = racun;
        this.kolicina = kolicina;
        this.cenaStavke = cenaStavke;
        this.jedinicnaCena = jedinicnaCena;
        this.usluga = usluga;
    }
}

```

Слика 60 - Класа СтавкаРачуна



```

public class TerminDezurstva implements Serializable, DomainObject<TerminDezurstva> {

    private int id;
    private Smene smena;
    private LocalTime vremeOd;
    private LocalTime vremeDo;

    public TerminDezurstva() {
    }

    public TerminDezurstva(int id, Smene smena, LocalTime vremeOd, LocalTime vremeDo) {
        this.id = id;
        this.smena = smena;
        this.vremeOd = vremeOd;
        this.vremeDo = vremeDo;
    }
}

```

Слика 61 - Класа ТерминДежурства





```
public class Usluga implements Serializable, DomainObject<Usluga> {  
  
    private int id;  
    private String naziv;  
    private double cena;  
  
    public Usluga() {  
    }  
  
    public Usluga(int id, String naziv, double cena) {  
        this.id = id;  
        this.naziv = naziv;  
        this.cena = cena;  
    }  
}
```

Слика 62 - Класа Услуга

Помоћне класе:



```
public enum Operacija implements Serializable {  
    LOGIN,  
    LOGOUT,  
    SERVER_LOGOUT,  
    REGISTER,  
    AZURIRANJE_PASSWORD,  
    AZURIRANJE_USERNAME;  
}
```

Слика 63 – Помоћни еnum Операција

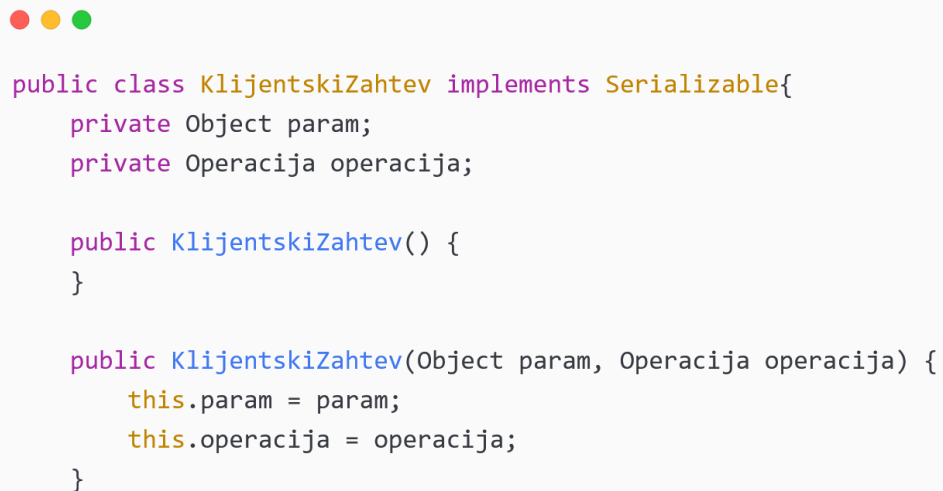


```
public class UlogovaniMusterija extends Musterija implements Serializable{  
  
    public UlogovaniMusterija(){  
  
    }  
  
    public UlogovaniMusterija(int id, String username) {  
        super(id, null, username, null, null, null); // samo id i username su mi u bazi  
    }  
}
```

Слика 64 - Помоћна класа УлогованиМуштерија

**КлијентскиЗахтев** - објект за слање података од клијента ка серверу. Састоји се од два атрибута:

1. *param* - представља објект над којим треба да се изврши захтевана операција
2. *operacija* - представља операцију која треба да се изврши над објектом



```
public class KlientскиЗахтев implements Serializable{
    private Object param;
    private Operacija operacija;

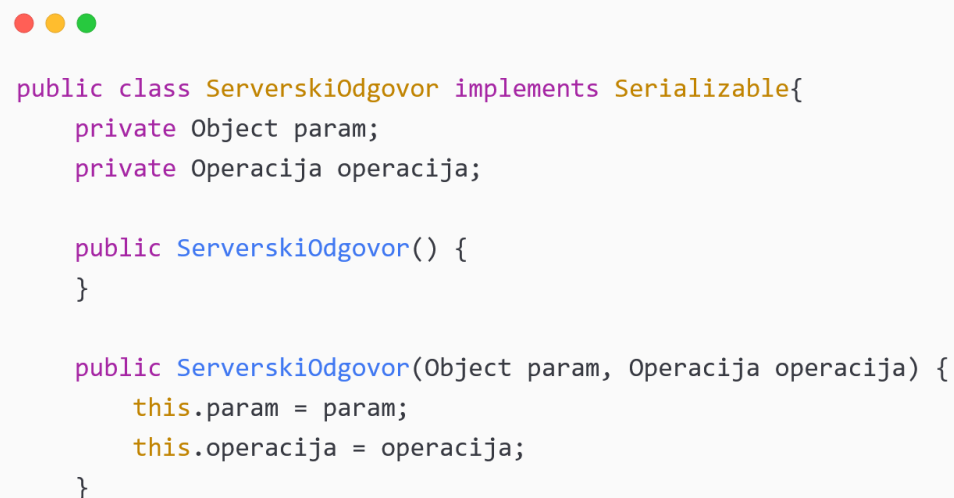
    public KlientскиЗахтев() {
    }

    public KlientскиЗахтев(Object param, Operacija operacija) {
        this.param = param;
        this.operacija = operacija;
    }
}
```

Слика 65 - Помоћна класа КлијентскиЗахтев

**СерверскиОдговор** - објект се користи за слање података од сервера ка клијенту. Садржи два атрибута:

1. *param* - представља објект који је резултат операције извршене на серверу.
2. *operacija* - представља операцију коју клијент треба да обради на својој страни



```
public class ServersкиОдговор implements Serializable{
    private Object param;
    private Operacija operacija;

    public ServersкиОдговор() {
    }

    public ServersкиОдговор(Object param, Operacija operacija) {
        this.param = param;
        this.operacija = operacija;
    }
}
```

Слика 66 - Помоћна класа СерверскиОдговор

**DomainObject** - интерфејс који све доменске класе наслеђују.



```
public interface DomainObject<T> {

    // Vraća SQL upit za INSERT operaciju
    String getInsertQuery();
    // Popunjava PreparedStatement za INSERT
    void fillInsertStatement(PreparedStatement ps) throws SQLException;

    // Vraća SQL upit za UPDATE operaciju
    String getUpdateQuery();
    // Popunjava PreparedStatement za UPDATE
    void fillUpdateStatement(PreparedStatement ps) throws SQLException;

    // Vraća SQL upit za DELETE operaciju
    String getDeleteQuery();
    // Popunjava PreparedStatement za DELETE
    void fillDeleteStatement(PreparedStatement ps) throws SQLException;

    // Vraća SQL upit za SELECT operaciju za jedan objekat
    String getSelectQuery();
    // Popunjava PreparedStatement za SELECT
    void fillSelectStatement(PreparedStatement ps) throws SQLException;

    // Kreira objekat iz ResultSet-a
    T createFromResultSet(ResultSet rs) throws SQLException;

    // Vraća SQL upit za SELECT operaciju za sve redove
    String getSelectAllQuery();
    // Popunjava PreparedStatement za SELECT
    void fillSelectAllStatement(PreparedStatement ps) throws SQLException;

    List<T> createListFromResultSet(ResultSet rs) throws SQLException;
}
```

Слика 67 - Помоћни интерфејс ДоменскиОбјекат

## 4.3 Пројектовање складишта података

На основу релационог модела и ограничења пројектоване су табеле базе података које користи наш софтверски систем:

Column Name	Data Type	Length	Default	PK?	Not Null?	Unsigned?	Auto Incr?	Zerofill?	On Update	Comment
prodavacId	bigint	20		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
terminId	bigint	20		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
datum	date			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Слика 68 - Табела Ангажовање

Column Name	Data Type	Length	Default	PK?	Not Null?	Unsigned?	Auto Incr?	Zerofill?	On Update	Comment
id	bigint	20		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
naziv	varchar	50		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
popust	bigint	20		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Слика 69 - Табела КатегоријаМуштерије

Column Name	Data Type	Length	Default	PK?	Not Null?	Unsigned?	Auto Incr?	Zerofill?	On Update	Comment
id	bigint	20		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
email	varchar	50		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
username	varchar	50		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
password	varchar	50		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
kategorijaId	bigint	20		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
preostaloVreme	bigint	20		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Слика 70 - Табела Муштерија

Column Name	Data Type	Length	Default	PK?	Not Null?	Unsigned?	Auto Incr?	Zerofill?	On Update	Comment
id	bigint	20		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
username	varchar	50		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Слика 71 - Табела МуштеријаУлоговани

Column Name	Data Type	Length	Default	PK?	Not Null?	Unsigned?	Auto Incr?	Zerofill?	On Update	Comment
id	bigint	20		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
ime	varchar	50		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
prezime	varchar	50		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
email	varchar	50		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
username	varchar	50		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
password	varchar	50		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Слика 72 - Табела Продавац

Column Name	Data Type	Length	Default	PK?	Not Null?	Unsigned?	Auto Incr?	Zerofill?	On Update	Comment
id	bigint	20		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
datum	datetime			<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
ukupnaCena	decimal	10,2		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
prodavacId	bigint	20		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
musterijaId	bigint	20		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Слика 73 - Табела Рачун

1 Columns 2 Indexes 3 Foreign Keys 4 Check Constraint 5 Advanced 6 SQL Preview										
<input type="checkbox"/> Column Name	Data Type	Length	Default	PK?	Not Null?	Unsigned?	Auto Incr?	Zerofill?	On Update	Comment
<input type="checkbox"/> racunId	bigint	20		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/> rb	bigint	20		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/> kolicina	bigint	20		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/> jedinicnaCena	decimal	10,2		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/> uslugaId	bigint	20		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/> cenaStavke	decimal	10,2		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

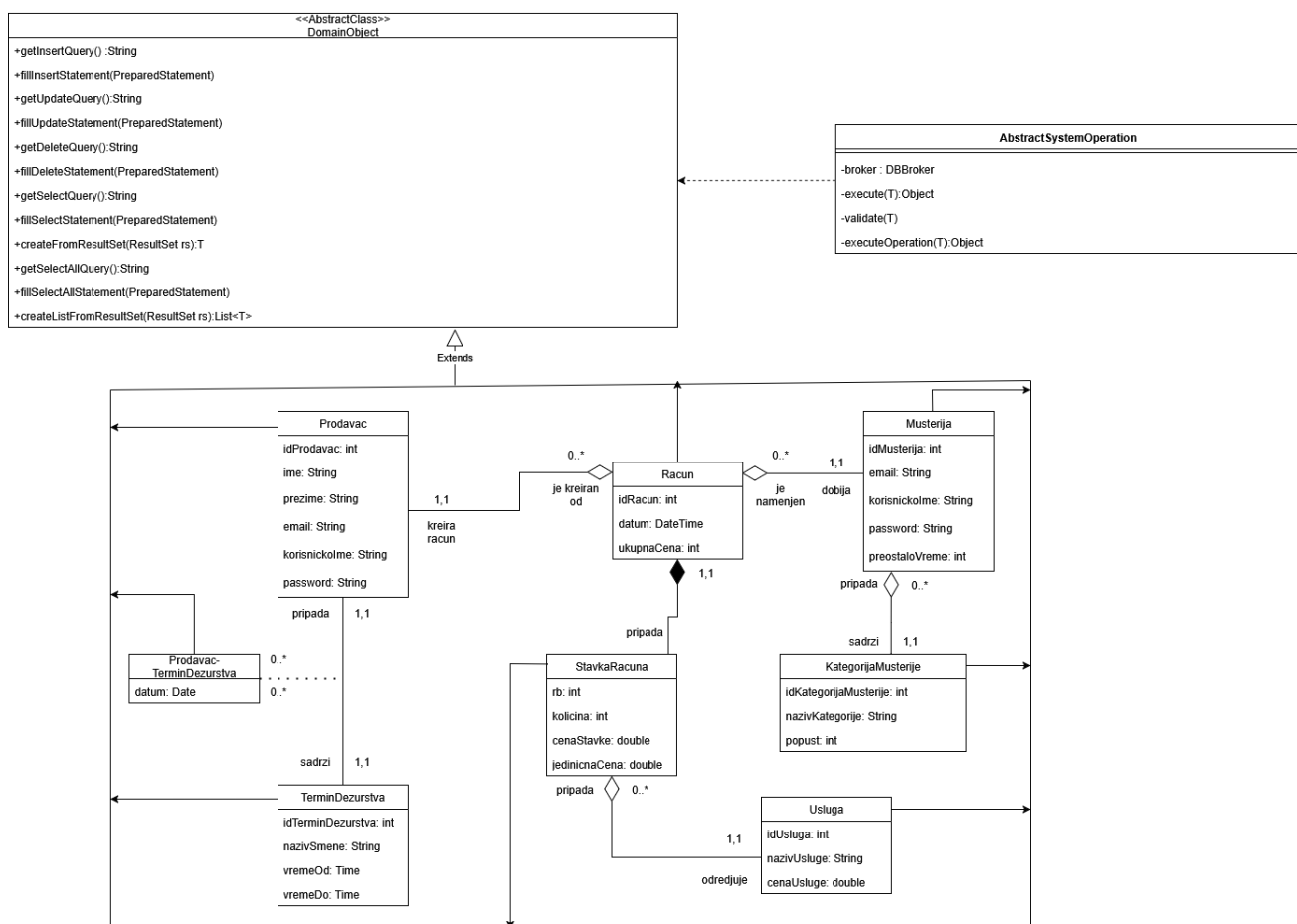
Слика 74 - Табела СтавкаРачуна

1 Columns 2 Indexes 3 Foreign Keys 4 Check Constraint 5 Advanced 6 SQL Preview										
<input type="checkbox"/> Column Name	Data Type	Length	Default	PK?	Not Null?	Unsigned?	Auto Incr?	Zerofill?	On Update	Comment
<input type="checkbox"/> id	bigint	20		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/> smena	varchar	20		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/> vremeOd	time			<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/> vremeDo	time			<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

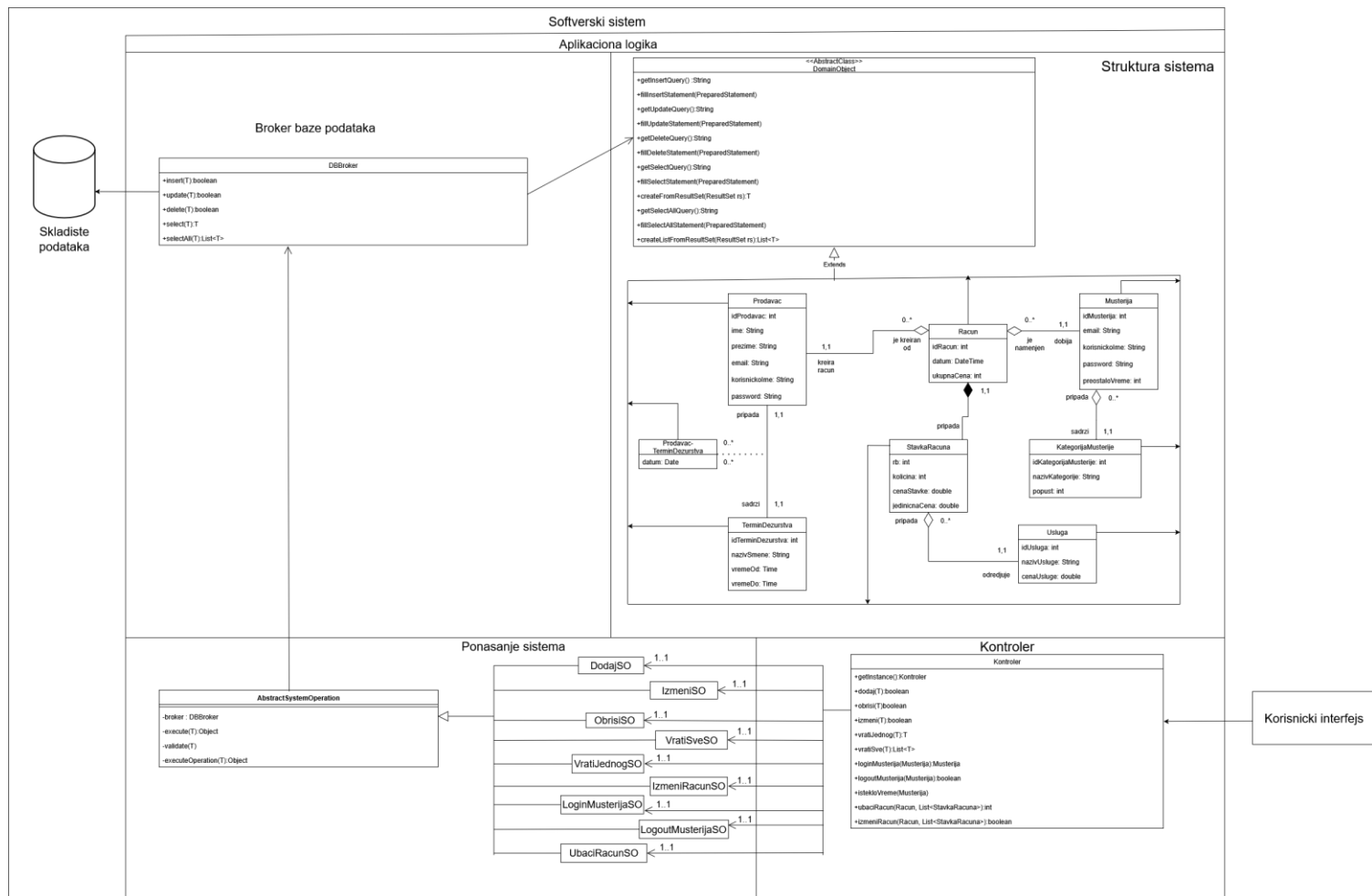
Слика 75 - Табела ТерминДежурства

1 Columns 2 Indexes 3 Foreign Keys 4 Check Constraint 5 Advanced 6 SQL Preview										
<input type="checkbox"/> Column Name	Data Type	Length	Default	PK?	Not Null?	Unsigned?	Auto Incr?	Zerofill?	On Update	Comment
<input type="checkbox"/> id	bigint	20		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/> naziv	varchar	50		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/> cena	decimal	10,2		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Слика 76 - Табела Услуга



Слика 77 - Дијаграм класа добијен након пројектовања доменских класа и класе ОпштиДоменскиОбјект



Слика 78 - Архитектура софтверског система



## 5. Имплементација

Софтверски систем је развијен у програмском језику Java и пројектован као клијент-

сервер апликација. За управљање базом података коришћен је **mysql Ver 15.1 Distrib 10.4.32-MariaDB**, док је као развојно окружење коришћен **NetBeans IDE 24**. На основу архитектуре система дефинисане су следеће софтверске класе:

### **iCaffe-Zajednicki:**

domen/DomainObject.java  
model/pomocne/UlogovaniMusterija.java  
model/pomocne/Angazovanje.java  
model/pomocne/KategorijaMusterije.java  
model/pomocne/Musterija.java  
model/pomocne/Prodavac.java  
model/pomocne/Racun.java  
model/pomocne/Smene.java  
model/pomocne/StavkaRacuna.java  
model/pomocne/TerminDezurstva.java  
model/pomocne/Usluga.java  
model/operacije/Operacija.java  
transfer/KlijentskiZahtev.java  
transfer/ServerskiOdgovor.java

### **iCaffe-Klijent:**

komunikacija/Komunikacija.java  
kontroler/Kontroler.java  
niti/MusterijaTimerNit.java  
view/IzmenaKlijentaForma.java  
view/KlijentskaForma.java  
view/LoginForma.java  
view/RegistracijaForma.java

## **iCaffe-Server:**

baza/DBBroker.java  
baza/Konekcija.java  
kontroler/Kontroler.java  
modeli/ModelTabeleAngazovanja.java  
modeli/ModelTabeleKategorijaMusterija.java  
modeli/ModelTabeleMusterija.java  
modeli/ModelTabeleRacuna.java  
modeli/ModelTabeleStavki.java  
modeli/ModelTabeleUsluga.java  
niti/MusterijaDetaljnoNit.java  
niti/MusterijaTimerNit.java  
operacije/AbstractSystemOperation.java  
operacije/DodajSO.java  
operacije/IzmeniSO.java  
operacije/LoginMusterijaSO.java  
operacije/LogoutMusterijaSO.java  
operacije/ObrisiSO.java  
operacije/UbaciracunSO.java  
operacije/VratiJednogSO.java  
operacije/VratiSveSO.java  
server/ObradiKlijentskiZahtev.java  
server/PokreniServer.java  
view/angazovanje/AngazovanjeForma.java  
view/angazovanje/PostaviAngazovanjeForma.java  
view/musterija/KategorijeMusterijaForma.java  
view/musterija/MusterijaDetaljnijeForma.java  
view/musterija/PostaviKategorijuMusterijeForma.java  
view/prodaja/PostaviStavkuForma.java  
view/prodaja/RacunForma.java  
view/usluga/LoginForma.java  
view/usluga/PretraziRacuneForma.java  
view/usluga/RacunDetaljnijeForma.java

## 6. Тестирање

Тестирање представља кључну фазу у процесу развоја софтвера, јер обезбеђује да системске операције раде исправно и да се грешке открију на време. У оквиру овог пројекта коришћен је **JUnit** оквир за аутоматизовано тестирање, што омогућава да се појединачне методе и функционалности системских операција провере независно од остатка система. На овај начин тестови обезбеђују стабилност, лакше одржавање кода и брже уочавање проблема, у поређењу са класичним ручним тестирањем.

**JUnit тестови** организовани су коришћењем посебних анотација које омогућавају јасну структуру и контролу извршавања. Анотације `@BeforeClass` и `@AfterClass` користе се за иницијализацију и ослобађање ресурса који важе за све тестове у оквиру једне класе, док `@Before` и `@After` служе за припрему и чишћење окружења пре и после сваког појединачног теста. Кључна анотација је `@Test`, којом се означава метода која садржи **конкретан сценарио провере**. На овај начин сваки тест је независан, поновљив и лако читљив, што значајно олакшава одржавање и проширивање тестног кода.

За потребе овог пројекта тестиране су системске операције као што су **LoginMusterijaSO**, **LogoutMusterijaSO**, **UbaciRacunSO** и **VratiSveSO**. Помоћу JUnit тестова проверавано је да ли ове операције враћају очекиване резултате у различитим сценаријима, као и да ли се у случају грешке систем понаша у складу са предвиђеним механизмима. На овај начин обезбеђена је поузданост основне логике апликације, а цео процес тестирања је аутоматизован тако да се може поновити више пута без додатног ручног рада.

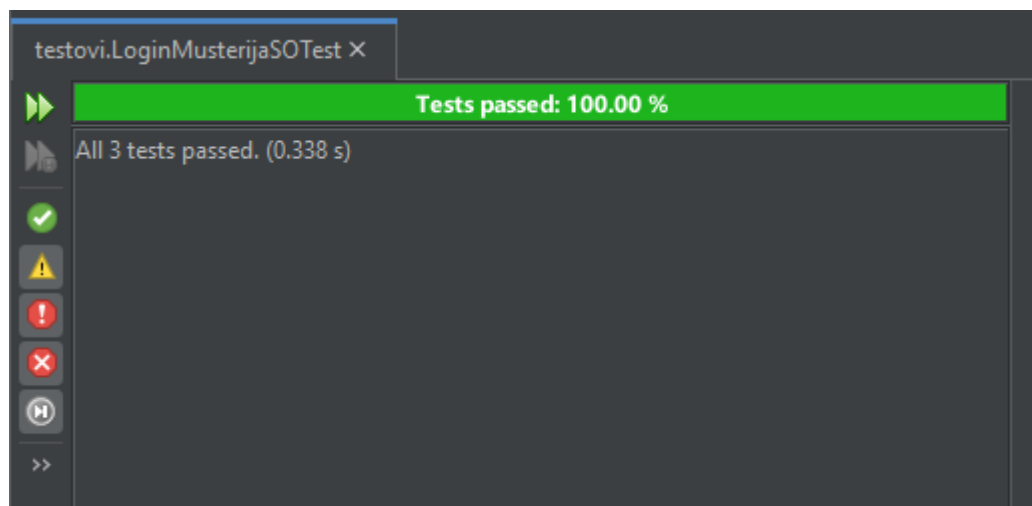
## LoginMusterijaSOTest

У оквиру тестирања системске операције **LoginMusterijaSO** провераване су различите ситуације при пријави корисника у систем. Коришћени су JUnit тестови који омогућавају аутоматизовану проверу појединачних сценарија.

Конкретно, тестирано је следеће:

1. **Успешна пријава** – овде се тестира пријава корисника који постоји у бази са исправним подацима. Тест проверава да ли систем враћа објекат са позитивним ID-јем, чиме се потврђује да је логика пријаве исправна и да корисник може нормално да приступи систему.
2. **Пријава непостојеће муштерије** – у овом тесту покушава се пријава корисника који није регистрован у систему. Очекује се да систем врати објекат са ID-јем -1, чиме се симболички означава да корисник не постоји. Овим се проверава да операција правилно обрађује неважеће податке.
3. **Пријава већ улогованог корисника** – овај тест симулира покушај пријаве корисника који је већ пријављен. Очекивани резултат је ID -2, што сигнализира да систем правилно детектује да је корисник већ у систему и да не дозвољава дупле пријаве.

Тестови укључују и чишћење стања у табели *муштерија\_улоговани* након сваког теста како би се обезбедила независност извршења следећих тестова. На овај начин осигурана је поузданост и стабилност функције пријаве, а аутоматизовани JUnit тестови омогућавају да се процес понавља више пута без додатног ручног рада.



Слика 79 - Сва три теста везана за пријаву су успешна

```

public class LoginMusterijaSOTest {

    private static LoginMusterijaSO loginSO;
    private Musterija rezultat;

    @BeforeClass
    public static void setUpClass() {
        // Ovo se pokreće jednom pre svih testova
        loginSO = new LoginMusterijaSO();
    }

    @AfterClass
    public static void tearDownClass() {
        // Oslobađanje resursa nakon svih testova, ako je potrebno
        loginSO = null;
    }

    @After
    public void ocistiBazu() throws Exception {
        if (rezultat != null) {
            Kontroler.getInstance().logoutMusterija(rezultat);
        }
    }

    @Test
    public void testLoginNePostojiMusterija() throws Exception {

        Musterija nepostojeci = new Musterija();
        nepostojeci.setEmail("fake@email.com");
        nepostojeci.setPassword("pogresno");

        rezultat = (Musterija) loginSO.execute(nepostojeci);

        assertNotNull("Rezultat ne sme biti null", rezultat);
        assertEquals("Ako musterija ne postoji, ID treba da bude -1", -1, rezultat.getId());
    }

    @Test
    public void testLoginUspesan() throws Exception {
        Musterija m = new Musterija();
        m.setEmail("test@test.com"); // mora postojati u bazi
        m.setPassword("123456"); // odgovara password-u

        rezultat = (Musterija) loginSO.execute(m);

        assertNotNull("Rezultat ne sme biti null", rezultat);
        assertTrue("Ako login uspe, ID mora biti pozitivan", rezultat.getId() > 0);
    }

    @Test
    public void testLoginVecUlogovan() throws Exception {
        Musterija m = new Musterija();
        m.setEmail("test@test.com"); // ista musterija kao u prethodnom testu
        m.setPassword("123456");

        // Prvi Login - uspešan
        rezultat = (Musterija) loginSO.execute(m);
        // Drugi Login - već uLogovan
        Musterija drugiRezultat = (Musterija) loginSO.execute(m);

        assertEquals("Ako se isti korisnik ponovo loginuje, ID treba da bude -2", -2, drugiRezultat.getId());
    }
}

```

Слика 80 - Код теста за пријаву корисника

## LogoutMusterijaSOTest

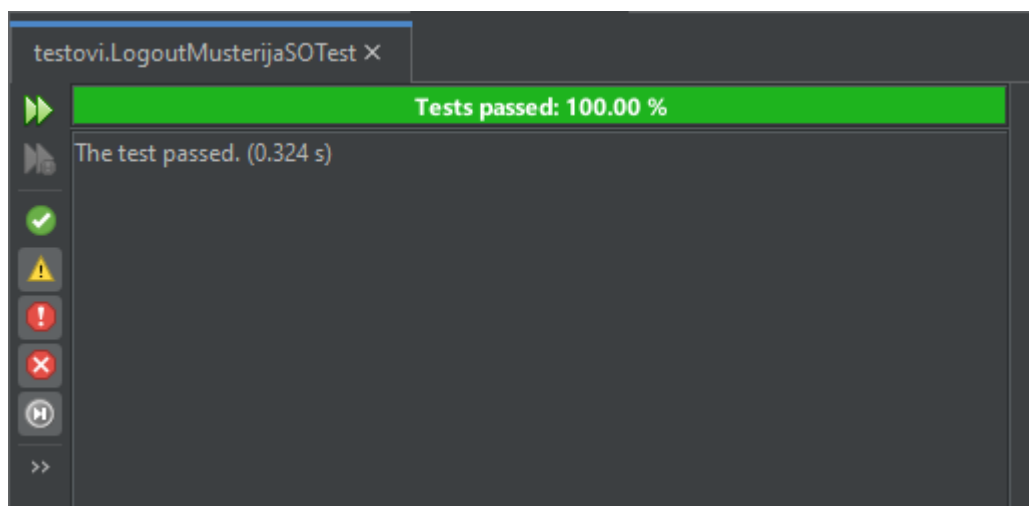
У оквиру тестирања системске операције **LogoutMusterijaSO** проверена је функционалност одјаве корисника из система. Тестови су осмишљени тако да симулирају стварну ситуацију у којој је корисник већ пријављен и потребно је да се без проблема одјави.

Пре покретања теста, корисник се аутоматски пријављује у систем како би се омогућила провера одјаве.

Конкретно, тестирано је следеће:

1. **Успешна одјава корисника** – Тест позива операцију одјаве и проверава да ли је резултат *true*, што означава да је корисник успешно излогован. Овим се потврђује да систем правилно управља стањем улогованих корисника и да одјава функционише без грешака.

Током тестирања обезбеђено је и чишћење стања у табели *муштерија\_улоговани* након сваког покушаја одјаве, како би се спречиле потенцијалне последице по друге тестове или по базу података. На овај начин је осигурана поузданост и стабилност основне логике одјаве корисника, а аутоматизовани JUnit приступ омогућава поновљивост теста без додатног ручног рада.



Слика 81 - Тест везан за одјаву корисника је успешан



```
public class LogoutMusterijaSOTest {

    private static Musterija musterija;

    @BeforeClass
    public static void setUpClass() throws Exception {
        musterija = new Musterija();
        musterija.setId(1);
        musterija = (Musterija) Kontroler.getInstance().vratiJednog(musterija);

        // Mora prvo da se uloguje da bi ga test izlogovao
        Kontroler.getInstance().loginMusterija(musterija);
    }

    @AfterClass
    public static void tearDownClass() throws Exception {
        // Ovo je da sigurno ukloni iz musterija_ulogovani ako tokom testa nesto ne bude ok
        Kontroler.getInstance().logoutMusterija(musterija);
    }

    @Test
    public void testLogoutUspesan() throws Exception {
        boolean rezultat = Kontroler.getInstance().logoutMusterija(musterija);

        assertTrue("Musterija treba da bude uspešno odlogovana", rezultat);
    }
}
```

Слика 82 - Код теста за одјаву корисника

## UbaciRacunSOTest

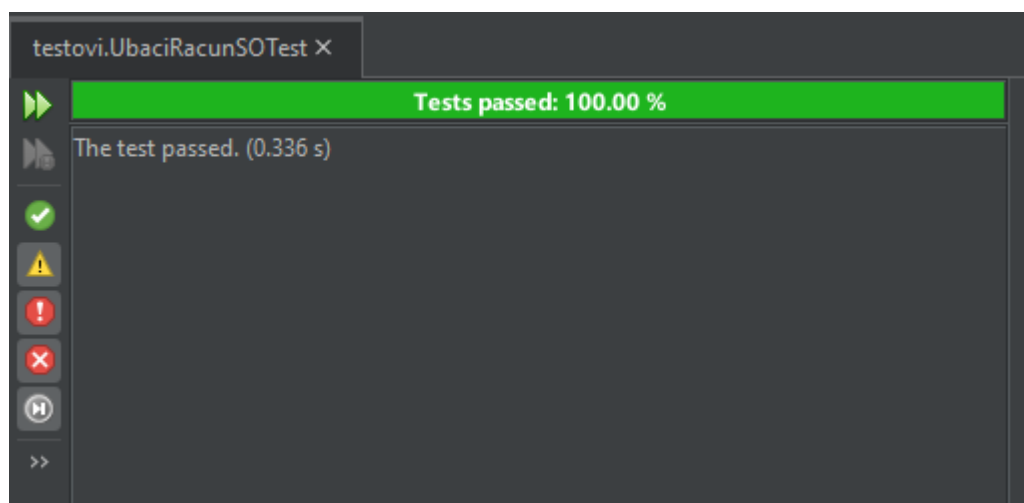
У оквиру тестирања системске операције **UbaciRacunSO** проверавана је исправност процеса креирања и чувања рачуна у систему. Тестови су осмишљени да симулирају стварне сценарије у којима корисник купује услуге и систем мора правилно да израчуна укупну цену, обради попусте и повезује све ставке са рачуном.

Пре извршења теста припремају се објекти продавца, муштерије и услуга, који већ постоје у бази. Затим се креирају ставке рачуна са дефинисаним количинама и ценама.

Конкретно, тестирано је следеће:

1. **Израчунавање укупне цене** – за сваки рачун тест проверава да ли систем правилно израчунава цену ставки и укупну цену рачуна након примене попушта, који зависи од категорије муштерије. Овим се осигурава тачност финансијских података.
2. **Повезивање ставки са рачуном** – свака ставка се правилно повезује са рачуном и добија своју израчунавану цену. Тиме се гарантује интегритет података у систему.
3. **Извршење системске операције и провера резултата** – операција UbaciRacunSO се извршава и проверава да ли је враћени резултат идентификатор рачуна (Integer) и да ли је позитиван. На тај начин се потврђује да је рачун успешно сачуван у бази.

Током тестирања обезбеђено је и чишћење базе након сваког покушаја, како би се избегле последице по друге тестове или постојеће податке. Овим је осигурана стабилност и поузданост логике за креирање и чување рачуна у систему, а аутоматизовани JUnit тестови омогућавају понављање процеса без додатног ручног рада.



Слика 83 - Тест везан за додавање ставки и рачуна је успешан



```

public class UbaciRacunSOTest {

    private List<StavkaRacuna> stavke = new ArrayList<>();
    private Racun racun = new Racun();

    @After
    public void ocistiBazu() throws Exception {
        if (stavke != null) {
            for (StavkaRacuna s : stavke) {
                Kontroler.getInstance().obrisi(s);
            }
        }
        if (racun != null) {
            Kontroler.getInstance().obrisi(racun);
        }
    }

    @Test
    public void testUbaciRacun() throws Exception {

        // Priprema prodavca i musterije
        Prodavac prodavac = new Prodavac();
        prodavac.setId(1);
        prodavac = Kontroler.getInstance().vratiJednog(prodavac);

        Musterija musterija = new Musterija();
        musterija.setId(1);
        musterija = Kontroler.getInstance().vratiJednog(musterija);

        // Stavke racuna
        Usluga usluga1 = new Usluga(1, null, 0.0);
        Usluga usluga2 = new Usluga(2, null, 0.0);

        usluga1 = Kontroler.getInstance().vratiJednog(usluga1);
        usluga2 = Kontroler.getInstance().vratiJednog(usluga2);

        StavkaRacuna s1 = new StavkaRacuna(1, null, 2, 0, 20, usluga1); // 2 x 20
        StavkaRacuna s2 = new StavkaRacuna(2, null, 1, 0, 90, usluga2); // 1 x 90

        stavke.add(s1);
        stavke.add(s2);

        // Racun
        racun.setDatum(LocalDateTime.now());
        racun.setProdavac(prodavac);
        racun.setMusterija(musterija);

        // Izračunavanje ukupne cene
        double suma = 0;
        for (StavkaRacuna s : stavke) {
            suma += s.getJedinicnaCena() * s.getKolicina();
        }
        double ukupnaCena = suma * (1 - musterija.getKategorijaMusterije().getPopust() / 100.0);
        racun.setUkupnaCena(ukupnaCena);

        // Povezivanje stavki sa racunom
        for (StavkaRacuna s : stavke) {
            s.setRacun(racun);
            s.setCenaStavke(s.getJedinicnaCena() * s.getKolicina());
        }

        // Poziv SO
        UbaciRacunSO so = new UbaciRacunSO(stavke);
        Object rezultat = so.execute(racun);

        assertTrue(rezultat instanceof Integer);
        int racunId = (Integer) rezultat;
        assertTrue(racunId > 0);

        // Provera ukupne cene
        assertEquals(91.0, racun.getUkupnaCena(), 0.001); // 2*20 + 1*90 = 130, minus 30% = 91 i ovo trece 0.001 je tolerancija
    }
}

```

Слика 84 - Код теста за убацивање ставки и рачуна

## VratiSveSOTest

У оквиру тестирања системске операције **VratiSveSO** проверавана је функционалност враћања свих улогованих корисника у систему. Циљ теста је да се потврди да систем правилно води евиденцију о активним сесијама корисника и да је могуће добити ажурну листу улогованих муштерија.

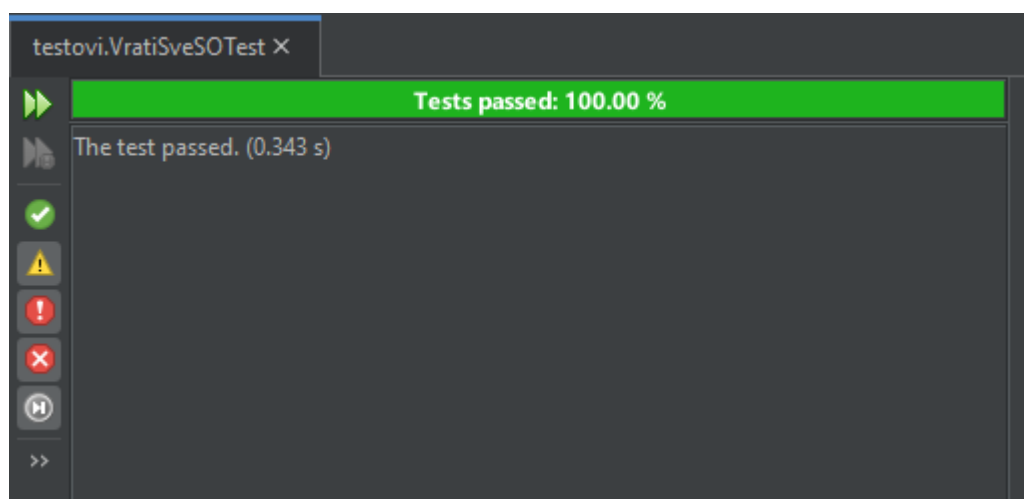
Пре самог теста, корисници са ID-јевима 1 и 2 се учитавају из базе и пријављују у систем. Ово омогућава да операција која враћа све улоговане кориснике има реалне податке за проверу.

Конкретно, тестирано је следеће:

1. **Враћање листе улогованих корисника** – тест позива операцију `vratiSve` и проверава да резултат није `null` и да садржи све кориснике који су пријављени у систему. Овим се осигурава тачност и интегритет информација о активним сесијама.

На крају теста корисници се одјављују како би се очистила табела улогованих муштерија, што омогућава независност следећих тестова и спречава утицај на базе података.

Тиме је потврђена поузданост логике за праћење улогованих корисника, а аутоматизовани JUnit тестови омогућавају поновљивост процеса без потребе за ручним проверама.



Слика 85 - Тест везан за враћање свих улогованих корисника је успешан



```

public class VratiSveSOTest {

    private static Musterija m1;
    private static Musterija m2;

    @BeforeClass
    public static void setUpClass() throws Exception {
        // Pre samog testa musterije sa id 1 i 2 se fetchuju iz baze
        m1 = new Musterija();
        m1.setId(1);
        m1 = (Musterija) Kontroler.getInstance().vratiJednog(m1);

        m2 = new Musterija();
        m2.setId(2);
        m2 = (Musterija) Kontroler.getInstance().vratiJednog(m2);

        // Musterije se loguju kako bi mogla da se vrati lista svih ulogovanih
        Kontroler.getInstance().loginMusterija(m1);
        Kontroler.getInstance().loginMusterija(m2);
    }

    @AfterClass
    public static void tearDownClass() throws Exception {
        // Ovo se poziva na kraju da bi se ocistila tabela musterija_ulogovani
        Kontroler.getInstance().logoutMusterija(m1);
        Kontroler.getInstance().logoutMusterija(m2);
    }

    @Test
    public void testVratiSveUlogovaneMusterije() throws Exception {
        List<Musterija> ulogovani = Kontroler.getInstance().vratiSve(new UlogovaniMusterija());

        assertNotNull("Lista ne sme biti null", ulogovani);
        assertTrue("Lista mora da sadrži m1", ulogovani.contains(m1));
        assertTrue("Lista mora da sadrži m2", ulogovani.contains(m2));
    }
}

```

Слика 86 - Код теста за враћање свих улогованих корисника

## 7. Закључак

У оквиру овог семинарског рада реализована је израда **клијентско-серверске апликације намењене управљању пословањем интернет кафића / гејминг играонице**. Апликација омогућава продавцима да се улогују у систем, започну своју смену и имају потпун увид у активности муштерија у реалном времену. Кориснички интерфејс пружа могућност евидентирања свих услуга и производа које кафић нуди, као и креирања рачуна за сваког појединачног клијента. Овај систем омогућава лакшу контролу и праћење рада, смањује ризик од грешака у евиденцији и побољшава укупну организацију пословања.

Једна од битних функционалности апликације јесте **праћење преосталог времена** које муштерије имају на својим сесијама. Уколико муштеријама време истекне, оне се аутоматски одјављују из система, чиме се обезбеђује да корисници не могу да користе услуге ван дозвољеног периода. Овај механизам омогућава прецизну контролу рада и доприноси транспарентном пословању кафића.

Рад на овом пројекту је омогућио боље разумевање клијентско-серверске архитектуре и примену шаблонског обрасца (Template Pattern). **Template Pattern** је коришћен за дефинисање опште структуре системских операција које су покривале основне CRUD операције, док су конкретне операције као што су пријава, одјава и креирање рачуна имплементирани у засебним системским операцијама. Ово омогућава генеричке функције које обезбеђују исти редослед корака у свим операцијама, поједностављују код и олакшавају одржавање система.

Највећи изазов на овом раду ми је представљала правилна имплементација Template Pattern-а који је за мене био нешто потпуно ново. У почетку нисам у потпуности схватао његов значај и потребу али кроз рад на пројекту сам приметио да у значајној мери доприноси „чистоћи“ кода као и једноставнијем одржавању система. Такође, оно што ми је доста значило приликом израде пројекта је основно предзнање SQL програмског језика и мислим да сам га кроз рад на пројекту још више продубио.

## Литература

1. Бојан Томић, Јелена Јовановић, Никола Миликић, Зоран Шеварац, Драган Ђурић, *Принципи програмирања*, Београд 2018.
2. Проф.др. Синиша Влајић, *Пројектовање софтвера(скрипта)*, Београд 2020.