

Универзитет у Београду

**Факултет организационих наука**

Лабораторија за софтверско инжењерство

Предмет: Пројектовање софтвера

**СЕМИНАРСКИ РАД**

Тема: Развој софтверског система за праћење рада  
библиотеке у Јава окружењу

Студент:

Ђорђије Радовић 162/2019

Ментор:

Татјана Стојановић

*Београд, 2023.*

## Садржај

Прикупљање корисничких захтева.....	1
1.1 Вербални опис .....	1
1.2 Случајеви коришћења.....	1
СК1: Случај коришћења – Креирање књиге.....	2
СК2: Случај коришћења – Претраживање књиге .....	3
СК3: Случај коришћења – Промена података о књизи.....	4
СК4: Случај коришћења – Брисање књиге .....	5
СК5: Случај коришћења – Креирање члана .....	6
СК6: Случај коришћења – Претраживање члана.....	7
СК7: Случај коришћења – Промена података о члану .....	8
СК8: Случај коришћења – Брисање члана.....	9
СК9: Случај коришћења – Креирање потврде о изнајмљивању књига (сложен СК) .....	10
СК10: Случај коришћења – Промена потврде о изнајмљивању књига (сложен СК) .....	11
Анализа .....	13
2.1 Системски дијаграми секвенци .....	13
ДС1: Дијаграм секвенце случаја коришћења – Креирање књиге .....	13
ДС2: Дијаграм секвенце случаја коришћења – Претраживање књиге .....	14
ДС3: Дијаграм секвенце случаја коришћења – Промена података о књизи.....	16
ДС4: Дијаграм секвенце случаја коришћења – Брисање књиге .....	19
ДС5: Дијаграм секвенце случаја коришћења – Креирање члана .....	21
ДС6: Дијаграм секвенце случаја коришћења – Претраживање члана.....	22
ДС7: Дијаграм секвенце случаја коришћења – Промена података о члану.....	24
ДС8: Дијаграм секвенце случаја коришћења – Брисање члана .....	27
ДС9: Дијаграм секвенце случаја коришћења – Креирање потврде о изнајмљивању књига (сложен СК).....	29
ДС10: Дијаграм секвенце случаја коришћења – Промена потврде о изнајмљивању књига (сложен СК).....	31
2.2 Понашање софтверског система – Дефинисање уговора о системским операцијама .....	35
2.3 Структура софтверског система – Концептуални (доменски) модел .....	37
2.4 Struktura softverskog sistema – relacioni model .....	37
Пројектовање .....	42
3.1 Архитектура софтверског система .....	42
3.2 Пројектовање корисничког интерфејса.....	42
3.2.1 Пројектовање екранских форми.....	43
СК1: Случај коришћења – Креирање књиге – пројектовање екранске форме .	45
СК2: Случај коришћења – Претраживање књиге – пројектовање екранске форме .....	47
СК3: Случај коришћења – Промена података о књизи – пројектовање екранске форме .....	50

СК4: Случај коришћења – Брисање књиге – пројектовање екранске форме ....	54
СК5: Случај коришћења – Креирање члана – пројектовање екранске форме ..	58
СК6: Случај коришћења – Претраживање члана – пројектовање екранске форме .....	60
СК7: Случај коришћења – Промена података о члану – пројектовање екранске форме .....	63
СК8: Случај коришћења – Брисање члана – пројектовање екранске форме ....	66
СК9: Случај коришћења – Креирање потврде о изнајмљивању књига (сложен СК) – пројектовање екранске форме .....	69
3.2.2 Пројектовање контролера корисничког интерфејса .....	74
3.3 Пројектовање апликационе логике .....	74
3.3.1 Контролер апликационе логике .....	75
3.3.2 Пословна логика .....	76
3.4 Пројектовање структуре софтверског система .....	82
3.4.1 Брокер базе података .....	93
3.4.2 Пројектовање складишта података .....	97
3.5 Коначна архитектура софтверског система .....	99
Тестирање .....	103
Литература .....	104

# Прикупљање корисничких захтева

## 1.1 Вербални опис

Апликација омогућава вођење евиденције о изнајмљивању књига у оквиру система библиотеке. Корисник (запослени) може креирањем књига да уноси у систем нове књиге, да претражује већ постојеће, као и да измењује податке о њима и да их брише из система. Исто тако, могу се креирати, претраживати, измењивати и брисати чланови библиотеке. Креирањем потврде о изнајмљивању књига, запослени бележи у систем које књиге је који члан библиотеке у којој згради библиотеке када узео. Променом потврде о изнајмљивању књига омогућава се бележење враћања изнајмљених књига.

## 1.2 Случајеви коришћења

У овој апликацији идентификовано је десет случајева коришћења:

1. Креирање књиге
2. Претраживање књиге
3. Промена података о књизи
4. Брисање књиге
5. Креирање члана
6. Претраживање члана
7. Промена података о члану
8. Брисање члана
9. Креирање потврде о изнајмљивању књига (сложен СК)
10. Промена потврде о изнајмљивању књига (сложен СК)



Слика 1. Модел случајева коришћења

## СК1: Случај коришћења – Креирање књиге

### Назив СК

Креирање књиге

### Актори СК

Запослени

### Учесници СК

Запослени и систем (програм)

**Предуслов:** Систем је укључен и Запослени је пријављен под својом шифром. Систем приказује форму за рад са књигом. Учитана је листа аутора.

### Основни сценарио СК

1. Запослени уноси податке за креирање књиге. (АПУСО)
2. Запослени контролише претходно унете податке за креирање књиге. (АНСО)
3. Запослени позива систем да запамти податке о књизи. (АПСО)
4. Систем памти податке о књизи. (СО)
5. Систем приказује запосленом запамћену књигу и поруку: “Систем је запамтио књигу”. (ИА)

### Алтернативна сценарија

- 5.1 Уколико систем не може да запамти податке о књизи он приказује запосленом поруку “Систем не може да запамти књигу”. (ИА)

## СК2: Случај коришћења – Претраживање књиге

### Назив СК

Претраживање књиге

### Актори СК

Запослени

### Учесници СК

Запослени и систем (програм)

**Предуслов:** Систем је укључен и **запослени** је пријављен под својом шифром. Систем приказује форму за рад са књигом.

### Основни сценарио СК

1. **Запослени** уноси вредност по којој претражује књиге. (АПУСО)
2. **Запослени** позива систем да нађе књиге по задатој вредности. (АПСО)
3. Систем тражи књиге по задатој вредности. (СО)
4. Систем приказује **запосленом** податке о књигама и поруку: “Систем је нашао књиге по задатој вредности”. (ИА)
5. **Запослени** бира књигу. (АПУСО)
6. **Запослени** позива систем да прочита књигу. (АПСО)
7. Систем учитава књигу. (СО)
8. Систем приказује **запосленом** податке о књизи и поруку: “Систем је успешно прочитао књигу.”. (ИА)

### Алтернативна сценарија

- 4.1 Уколико систем не може да нађе књиге он приказује **запосленом** поруку: “Систем не може да нађе ниједну књигу по задатој вредности”. Прекида се извршење сценарија. (ИА)
- 8.1 Уколико систем не може да прочита књигу он приказује **запосленом** поруку: “Систем не може да прочита књигу”. (ИА)

## СКЗ: Случај коришћења – Промена података о књизи

### Назив СК

Промена података о књизи

### Актори СК

Запослени

### Учесници СК

Запослени и систем (програм)

**Предуслов:** Систем је укључен и запослени је пријављен под својом шифром. Систем приказује форму за рад са књигом. Учитана је листа аутора.

### Основни сценарио СК

1. Запослени уноси вредност по којој претражује књиге. (АПУСО)
2. Запослени позива систем да нађе књиге по задатој вредности. (АПСО)
3. Систем тражи књиге по задатој вредности. (СО)
4. Систем приказује запосленом податке о књигама и поруку: “Систем је нашао књиге по задатој вредности”. (ИА)
5. Запослени бира књигу. (АПУСО)
6. Запослени позива систем да прочита књигу. (АПСО)
7. Систем учитава књигу. (СО)
8. Систем приказује запосленом податке о књизи и поруку: “Систем је успешно прочитао књигу”. (ИА)
9. Запослени уноси (мења) податке о књизи. (АПУСО)
10. Запослени контролише да ли је коректно унео податке о књизи. (АНСО)
11. Запослени позива систем да измени податке о књизи. (АПСО)
12. Систем памти податке о књизи. (СО)
13. Систем приказује запосленом измењену књигу и поруку: “Систем је изменио књигу.” (ИА)

### Алтернативна сценарија

- 4.1 Уколико систем не може да нађе књиге он приказује запосленом поруку: “Систем не може да нађе ниједну књигу по задатој вредности”. Прекида се извршење сценарија. (ИА)
- 8.1 Уколико систем не може да прочита књигу он приказује запосленом поруку: “Систем не може да прочита књигу”. Прекида се извршење сценарија. (ИА)
- 13.1 Уколико систем не може да измени податке о књизи он приказује запосленом поруку “Систем не може да измени књигу”. (ИА)

## СК4: Случај коришћења – Брисање књиге

### Назив СК

Брисање књиге

### Актори СК

Запослени

### Учесници СК

Запослени и систем (програм)

**Предуслов:** Систем је укључен и запослени је пријављен под својом шифром. Систем приказује форму за рад са књигом.

### Основни сценарио СК

1. Запослени уноси вредност по којој претражује књиге. (АПУСО)
2. Запослени позива систем да нађе књиге по задатој вредности. (АПСО)
3. Систем тражи књиге по задатој вредности. (СО)
4. Систем приказује запосленом податке о књигама и поруку: “Систем је нашао књиге по задатој вредности”. (ИА)
5. Запослени бира књигу. (АПУСО)
6. Запослени позива систем да учита књигу. (АПСО)
7. Систем учитава књигу. (СО)
8. Систем приказује запосленом књигу и поруку: “Систем је успешно учитао књигу” (ИА)
9. Запослени позива систем да обрише књигу. (АПСО)
10. Систем брише књигу. (СО)
11. Систем приказује запосленом поруку: “Систем је обрисао књигу.” (ИА)

### Алтернативна сценарија

- 4.1 Уколико систем не може да нађе књиге он приказује запосленом поруку: “Систем не може да нађе ниједну књигу по задатој вредности”. Прекида се извршење сценарија. (ИА)
- 8.1 Уколико систем не може да учита књигу он приказује запосленом поруку: “Систем не може да учита књигу”. Прекида се извршење сценарија. (ИА)
- 11.1 Уколико систем не може да обрише књигу он приказује запосленом поруку “Систем не може да обрише књигу”. (ИА)



## СК5: Случај коришћења – Креирање члана

### Назив СК

Креирање члана

### Актери СК

Запослени

### Учесници СК

Запослени и систем (програм)

**Предуслов:** Систем је укључен и **запослени** је пријављен под својом шифром. Систем приказује форму за рад са **чланом**.

### Основни сценарио СК

1. **Запослени** уноси податке за креирање **члана**. (АПУСО)
2. **Запослени** контролише претходно унете податке за креирање **члана**. (АНСО)
3. **Запослени** позива **систем** да запамти податке о **члану**. (АПСО)
4. **Систем** памти податке о **члану**. (СО)
5. **Систем** приказује **запосленом** запамћеног **члана** и поруку: “**Систем** је запамтио **члана**”. (ИА)

### Алтернативна сценарија

- 5.1 Уколико **систем** не може да запамти податке о **члану** он приказује **запосленом** поруку “**Систем** не може да запамти **члана**”. (ИА)

## СК6: Случај коришћења – Претраживање члана

### Назив СК

Претраживање члана

### Актери СК

Запослени

### Учесници СК

Запослени и систем (програм)

**Предуслов:** Систем је укључен и **запослени** је пријављен под својом шифром. Систем приказује форму за рад са **чланом**.

### Основни сценарио СК

1. **Запослени** уноси вредност по којој претражује **чланове**. (АПУСО)
2. **Запослени** позива **систем** да нађе **чланове** по задатој вредности. (АПСО)
3. **Систем** тражи **чланове** по задатој вредности. (СО)
4. **Систем** приказује **запосленом** податке о **члановима** и поруку: “**Систем** је нашао **чланове** по задатој вредности”. (ИА)
5. **Запослени** бира **члана**. (АПУСО)
6. **Запослени** позива **систем** да учита **члана**. (АПСО)
7. **Систем** учитава **члана**. (СО)
8. **Систем** приказује **запосленом** податке о **члану** и поруку: “**Систем** је успешно учитао **члана**”. (ИА)

### Алтернативна сценарија

- 4.1 Уколико **систем** не може да нађе **чланове** он приказује **запосленом** поруку: “**Систем** не може да нађе ниједног **члана** по задатој вредности”. Прекида се извршење сценарија. (ИА)
- 8.1 Уколико **систем** не може да учита **члана** он приказује **запосленом** поруку: “**Систем** не може да учита **члана**”. (ИА)

## СК7: Случај коришћења – Промена података о члану

### Назив СК

Промена података о члану

### Актори СК

Запослени

### Учесници СК

Запослени и систем (програм)

**Предуслов:** Систем је укључен и запослени је пријављен под својом шифром. Систем приказује форму за рад са чланом.

### Основни сценарио СК

1. Запослени уноси вредност по којој претражује чланове. (АПУСО)
2. Запослени позива систем да нађе чланове по задатој вредности. (АПСО)
3. Систем тражи чланове по задатој вредности. (СО)
4. Систем приказује запосленом податке о члановима и поруку: “Систем је нашао чланове по задатој вредности”. (ИА)
5. Запослени бира члана. (АПУСО)
6. Запослени позива систем да учита члана. (АПСО)
7. Систем учитава члана. (СО)
8. Систем приказује запосленом податке о члану и поруку: “Систем је успешно учитао члана.”. (ИА)
9. Запослени уноси (мења) податке о члану. (АПУСО)
10. Запослени контролише да ли је коректно унео податке о члану. (АНСО)
11. Запослени позива систем да измени податке о члану. (АПСО)
12. Систем памти податке о члану. (СО)
13. Систем приказује запосленом измењеног члана и поруку: “Систем је изменио члана.” (ИА)

### Алтернативна сценарија

- 4.1 Уколико систем не може да нађе чланове он приказује запосленом поруку: “Систем не може да нађе ниједног члана по задатој вредности”. Прекида се извршење сценарија. (ИА)
- 8.1 Уколико систем не може да учита члана он приказује запосленом поруку: “Систем не може да учита члана”. Прекида се извршење сценарија. (ИА)
- 13.1 Уколико систем не може да измени податке о члану он приказује запосленом поруку “Систем не може да измени члана”. (ИА)

## СК8: Случај коришћења – Брисање члана

### Назив СК

Брисање члана

### Актери СК

Запослени

### Учесници СК

Запослени и систем (програм)

**Предуслов:** Систем је укључен и запослени је пријављен под својом шифром. Систем приказује форму за рад са чланом.

### Основни сценарио СК

1. Запослени уноси вредност по којој претражује чланове. (АПУСО)
2. Запослени позива систем да нађе чланове по задатој вредности. (АПСО)
3. Систем тражи чланове по задатој вредности. (СО)
4. Систем приказује запосленом податке о члановима и поруку: “Систем је нашао чланове по задатој вредности”. (ИА)
5. Запослени бира члана. (АПУСО)
6. Запослени позива систем да учита члана. (АПСО)
7. Систем учитава члана. (СО)
8. Систем приказује запосленом члана и поруку: “Систем је успешно учитао члана” (ИА)
9. Запослени позива систем да обрише члана. (АПСО)
10. Систем брише члана. (СО)
11. Систем приказује запосленом поруку: “Систем је обрисао члана.” (ИА)

### Алтернативна сценарија

- 4.1 Уколико систем не може да нађе чланове он приказује запосленом поруку: “Систем не може да нађе ниједног члана по задатој вредности”. Прекида се извршење сценарија. (ИА)
- 8.1 Уколико систем не може да учита члана он приказује запосленом поруку: “Систем не може да учита члана”. Прекида се извршење сценарија. (ИА)
- 11.1 Уколико систем не може да обрише члана он приказује запосленом поруку “Систем не може да обрише члана”. (ИА)

## СК9: Случај коришћења – Креирање потврде о изнајмљивању књига (сложен СК)

### Назив СК

Креирање потврде о изнајмљивању књига

### Актери СК

Запослени

### Учесници СК

Запослени и систем (програм)

**Предуслов:** Систем је укључен и **запослени** је пријављен под својом шифром. Систем приказује форму за рад са **потврдом о изнајмљивању књига**. Учитане су листа свих чланова и листа свих књига у **систему**.

### Основни сценарио СК

1. **Запослени** уноси податке у **потврду о изнајмљивању књига**. (АПУСО)
2. **Запослени** контролише да ли је коректно унео податке у **потврду о изнајмљивању књига**. (АНСО)
3. **Запослени** позива **систем** да запамти податке о **потврди о изнајмљивању књига**. (АПСО)
4. **Систем** памти податке о **потврди о изнајмљивању књига**. (СО)
5. **Систем** приказује **запосленом** запамћену **потврду о изнајмљивању књига** и поруку: “**Систем** је запамтио **потврду о изнајмљивању књига**”. (ИА)

### Алтернативна сценарија

5.1 Уколико **систем** не може да запамти податке о **потврди о изнајмљивању књига** он приказује **запосленом** поруку “**Систем** не може да запамти **потврду о изнајмљивању књига**”. (ИА)

## СК10: Случај коришћења – Промена потврде о изнајмљивању књига (сложен СК)

### Назив СК

Промена потврде о изнајмљивању књига

### Актери СК

Запослени

### Учесници СК

Запослени и систем (програм)

**Предуслов:** Систем је укључен и запослени је пријављен под својом шифром. Систем приказује форму за рад са потврдом о изнајмљивању књига. Учитане су листа свих чланова и листа свих књига у систему.

### Основни сценарио СК

1. Запослени уноси вредност по којој претражује изнајмљивања књига. (АПУСО)
2. Запослени позива систем да нађе изнајмљивања књига по задатој вредности. (АПСО)
3. Систем тражи изнајмљивања књига по задатој вредности. (СО)
4. Систем приказује запосленом изнајмљивања књига и поруку: “Систем је нашао изнајмљивања књига по задатој вредности”. (ИА)
5. Запослени бира изнајмљивања књига. (АПУСО)
6. Запослени позива систем да прочита изнајмљивања књига. (АПСО)
7. Систем читава изнајмљивања књига. (СО)
8. Систем приказује запосленом податке о изнајмљивањима књига и поруку: “Систем је успешно прочитао изнајмљивања књига”. (ИА)
9. Запослени уноси (мења) податке о изнајмљивањима књига. (АПУСО)
10. Запослени контролише да ли је коректно унео податке о изнајмљивањима књига. (АНСО)
11. Запослени позива систем да измени податке о изнајмљивањима књига. (АПСО)
12. Систем памти податке о изнајмљивањима књига. (СО)
13. Систем приказује запосленом измењена изнајмљивањима књига и поруку: “Систем је изменио изнајмљивањима књига.” (ИА)

## Алтернативна сценарија

4.1 Уколико **систем** не може да нађе **изнајмљивања књига** он приказује **запосленом** поруку: “**Систем** не може да нађе ниједно **изнајмљивање књига** по задатој вредности”. Прекида се извршење сценарија. (ИА)

8.1 Уколико **систем** не може да учита **изнајмљивања књига** он приказује **запосленом** поруку: “**Систем** не може да учита **изнајмљивања књига**”. Прекида се извршење сценарија. (ИА)

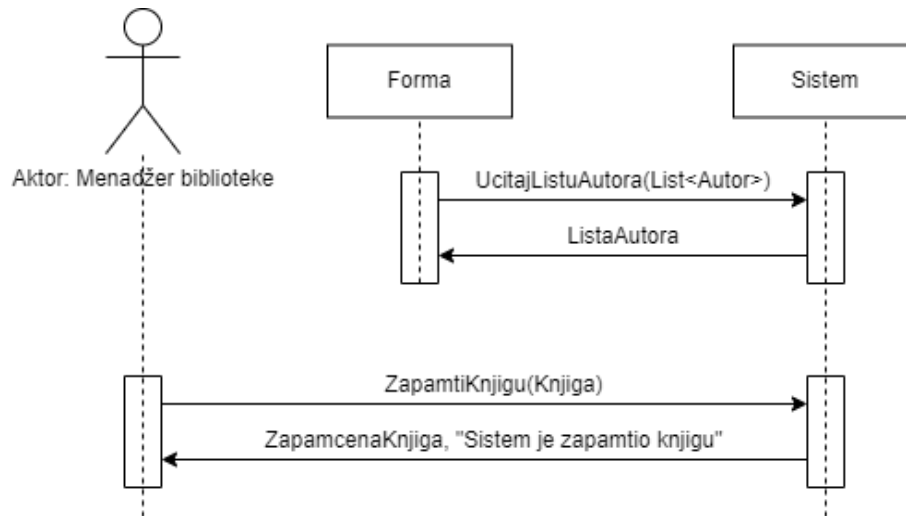
13.1 Уколико **систем** не може да измени податке о **изнајмљивањима књига** он приказује **запосленом** поруку “**Систем** не може да измени **изнајмљивања књига**”. (ИА)

# Анализа

## 2.1 Системски дијаграми секвенци

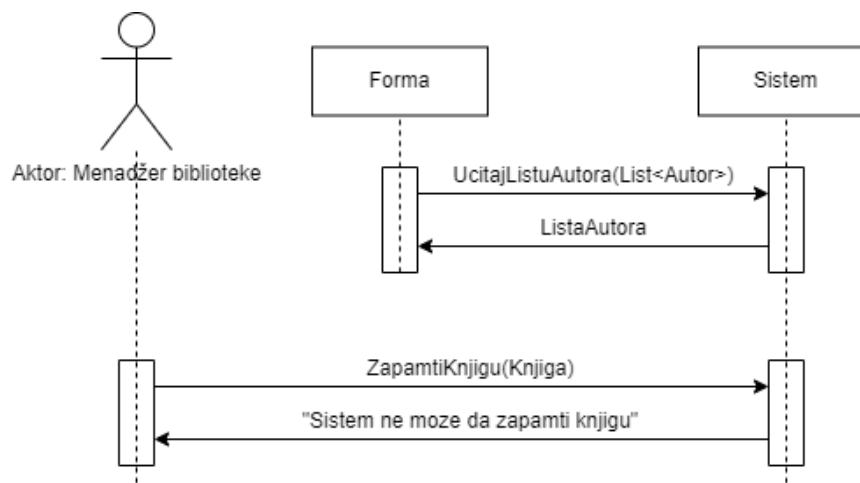
### ДС1: Дијаграм секвенце случаја коришћења – Креирање књиге

1. **Форма** **позива** **систем** да учита листу аутора. (АПСО)
2. **Систем** **враћа** **форми** листу аутора. (ИА)
3. **Запослени** **позива** **систем** да запамти податке о **књизи**. (АПСО)
4. **Систем** **приказује** **запосленом** запамћену **књигу** и поруку: “**Систем** је запамтио **књигу**”. (ИА)



### Алтернативна сценарија

- 4.1 Уколико **систем** не може да запамти податке о **књизи** он приказује **запосленом** поруку “**Систем** не може да запамти **књигу**”. (ИА)



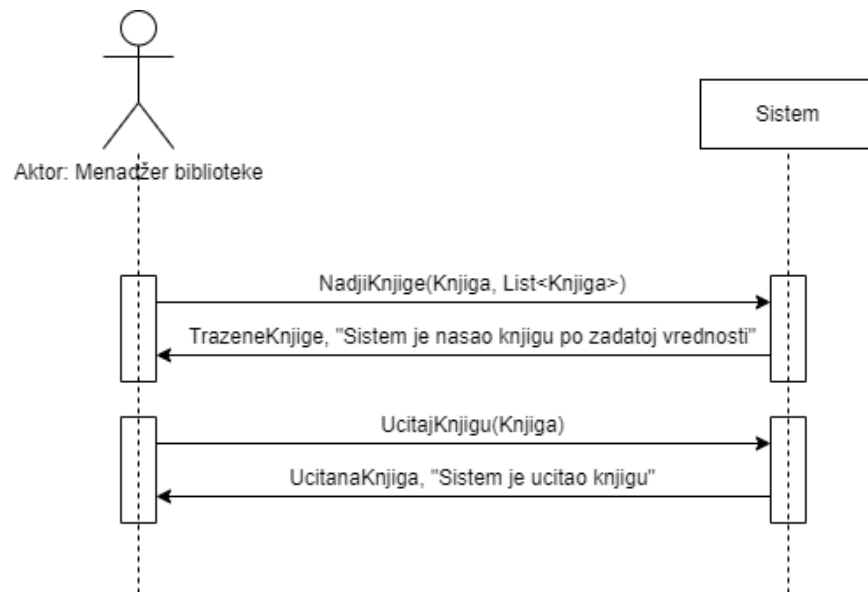
Са наведених секвенцих дијаграма уочавају се 2 системске операције:

1. Signal **UcitajListuAutora(List<Autor>);**
2. Signal **ZapamtiKnjigu(Knjiga).**



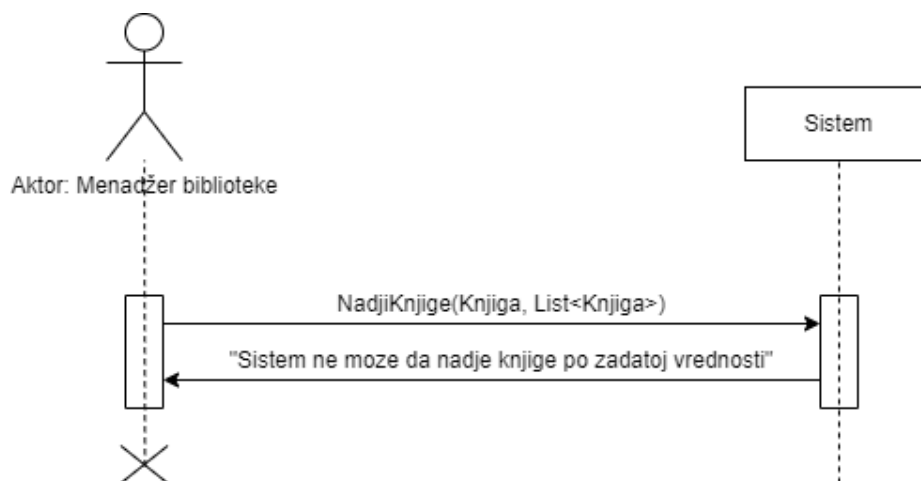
## ДС2: Дијаграм секвенце случаја коришћења – Претраживање књиге

1. **Запослени** **позива** **систем** да нађе **књиге** по задатој вредности. (АПСО)
2. **Систем** **приказује** **запосленом** податке о **књигама** и поруку: “**Систем** је нашао **књиге** по задатој вредности”. (ИА)
3. **Запослени** **позива** **систем** да учита **књигу**. (АПСО)
4. **Систем** **приказује** **запосленом** податке о **књизи** и поруку: “**Систем** је успешно учитао **књигу**.”. (ИА)

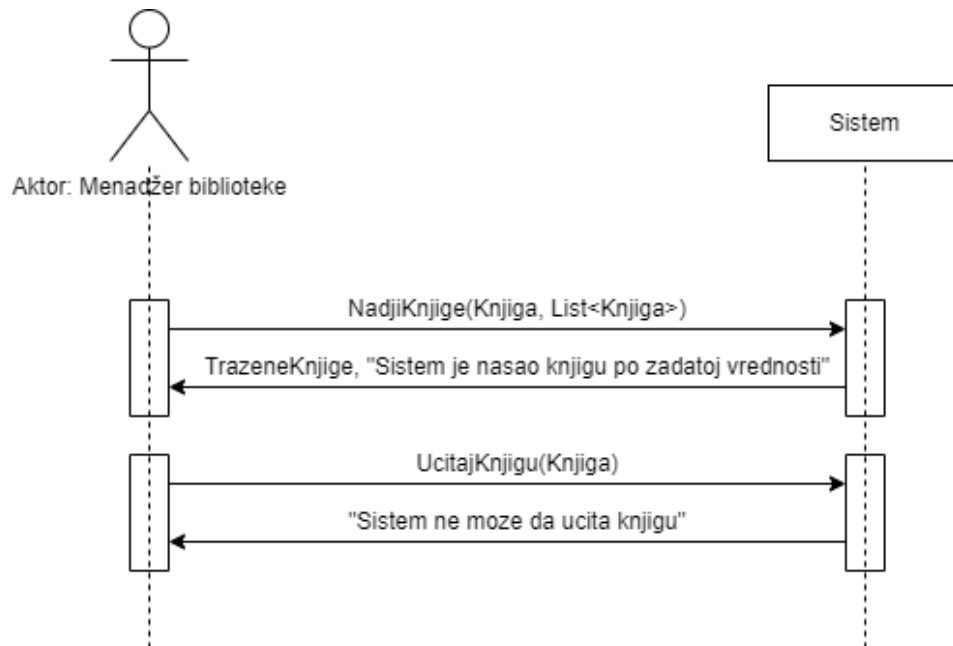


### Алтернативна сценарија

2.1 Уколико **систем** не може да нађе **књиге** он приказује **запосленом** поруку: “**Систем** не може да нађе ниједну **књигу** по задатој вредности”. Прекида се извршење сценарија. (ИА)



4.1 Уколико **систем** не може да учита **књигу** он приказује **запосленом** поруку:  
“**Систем** не може да учита **књигу**”. (ИА)

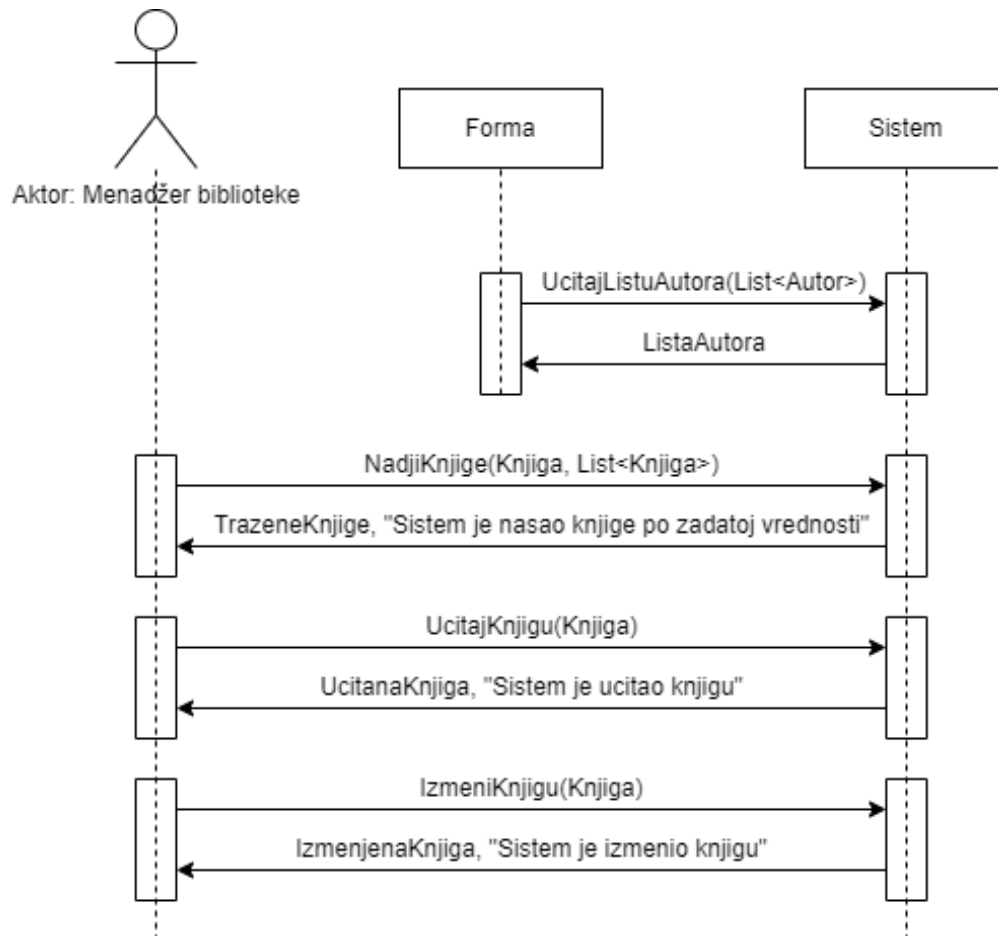


Са наведених секвенцих дијаграма уочавају се 2 системске операције:

1. Signal **NadjiKnjige(Knjiga, List<Knjiga>);**
2. Signal **UcitajKnjigu(Knjiga).**

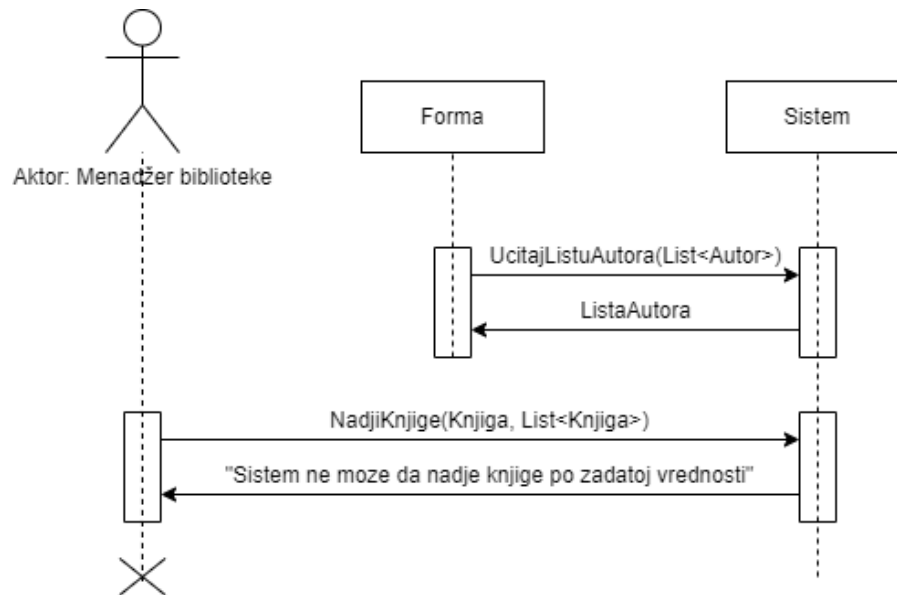
### ДСЗ: Дијаграм секвенце случаја коришћења – Промена података о књизи

1. **Форма** **позива** **систем** да учита листу аутора. (АПСО)
2. **Систем** **враћа** **форми** листу аутора. (ИА)
3. **Запослени** **позива** **систем** да нађе **књиге** по задатој вредности. (АПСО)
4. **Систем** **приказује** **запосленом** податке о **књигама** и поруку: “**Систем** је нашао **књиге** по задатој вредности”. (ИА)
5. **Запослени** **позива** **систем** да учита **књигу**. (АПСО)
6. **Систем** **приказује** **запосленом** податке о **књизи** и поруку: “**Систем** је успешно учитао **књигу**”. (ИА)
7. **Запослени** **позива** **систем** да измени податке о **књизи**. (АПСО)
8. **Систем** **приказује** **запосленом** измењену **књигу** и поруку: “**Систем** је изменио **књигу**.” (ИА)

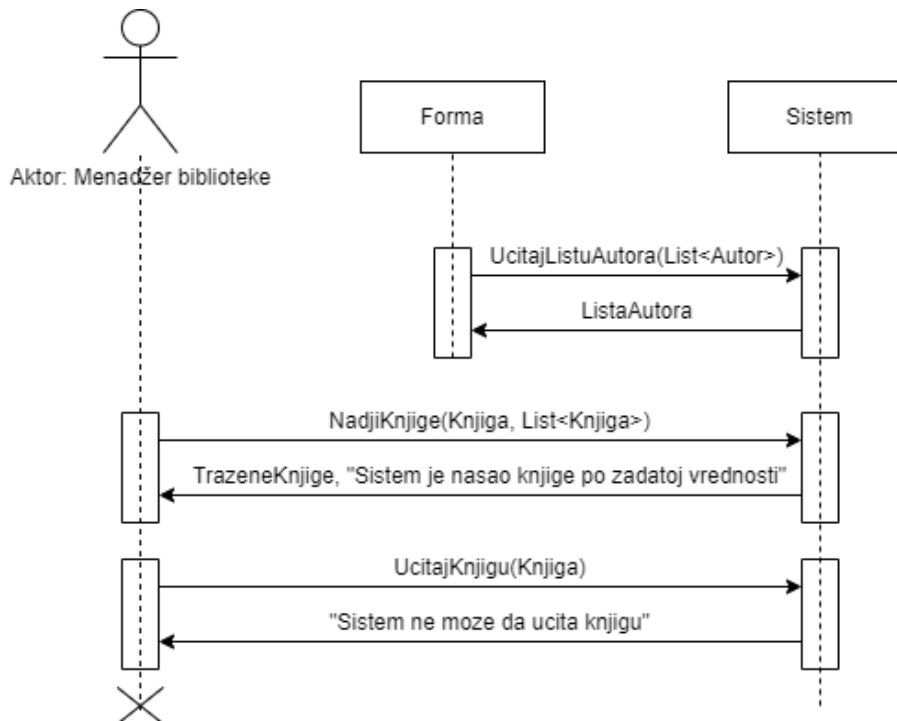


## Алтернативна сценарија

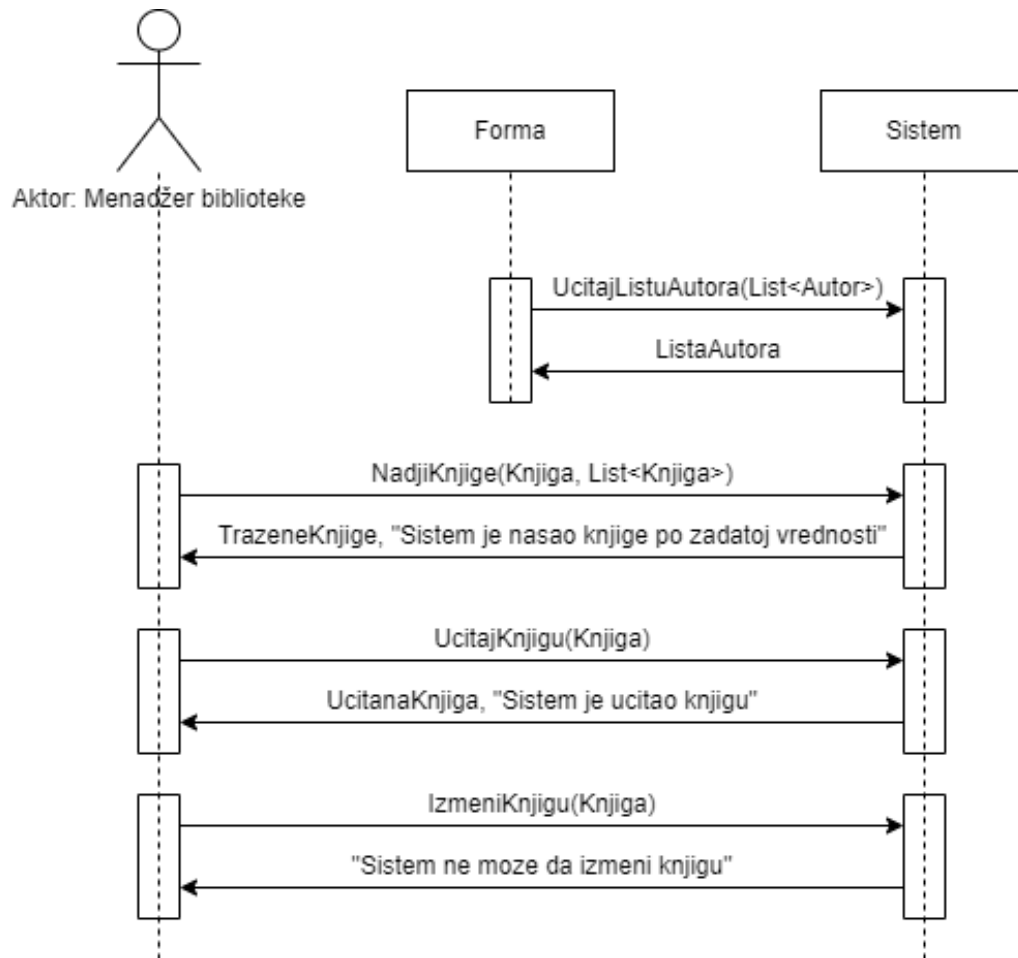
4.1 Уколико **систем** не може да нађе **књиге** он приказује **запосленом** поруку: “**Систем** не може да нађе ниједну **књигу** по задатој вредности”. Прекида се извршење сценарија. (ИА)



6.1 Уколико **систем** не може да учита **књигу** он приказује **запосленом** поруку: “**Систем** не може да учита **књигу**”. Прекида се извршење сценарија. (ИА)



8.1 Уколико **систем** не може да запамти податке о **књизи** он приказује **запосленом** поруку “**Систем** не може да запамти **књигу**”. (ИА)

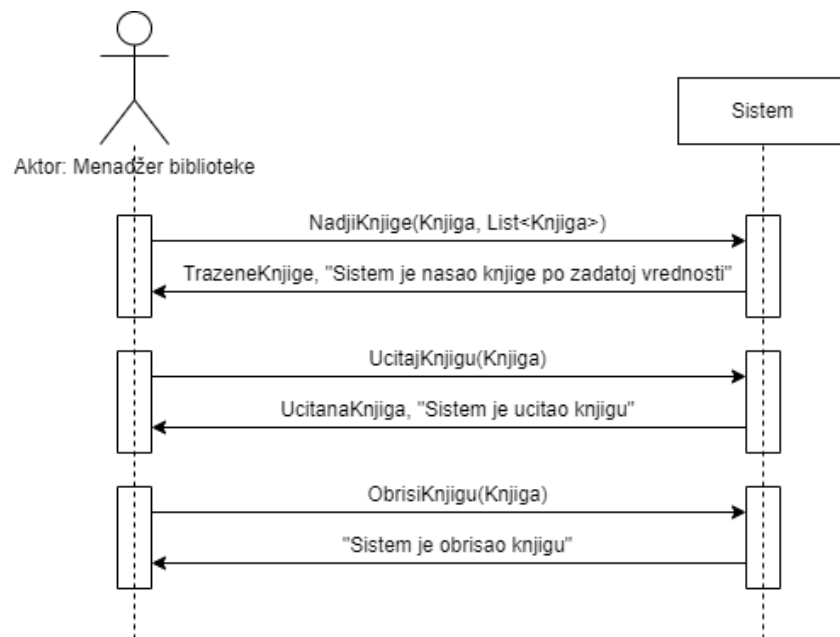


Са наведених секвенчних дијаграма уочавају се 4 системске операције:

1. Signal **UcitajListuAutora(List<Autor>);**
2. Signal **UcitajListuZgradaBiblioteke(List<ZgradaBiblioteke>);**
3. Signal **NadjiKnjige(Knjiga, List<Knjiga>);**
4. Signal **UcitajKnjigu(Knjiga);**
5. Signal **IzmeniKnjigu(Knjiga).**

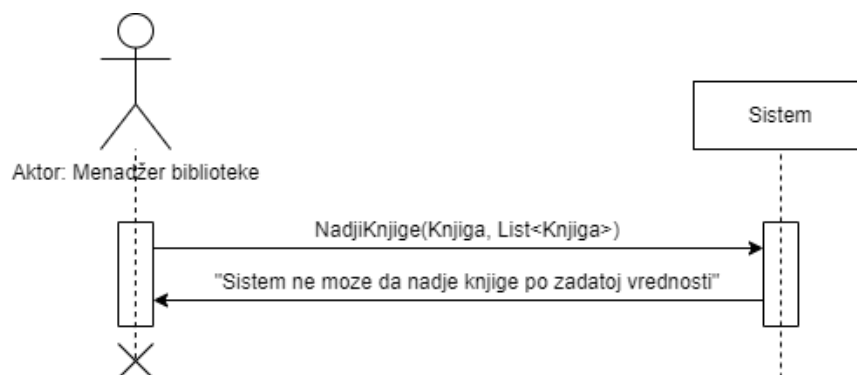
#### ДС4: Дијаграм секвенце случаја коришћења – Брисање књиге

1. **Запослени** **позива** **систем** да нађе **књиге** по задатој вредности. (АПСО)
2. **Систем** **приказује** **запосленом** податке о **књигама** и поруку: “**Систем** је нашао **књиге** по задатој вредности”. (ИА)
3. **Запослени** **позива** **систем** да учита **књигу**. (АПСО)
4. **Систем** **приказује** **запосленом** **књигу** и поруку: “**Систем** је успешно учитао **књигу**” (ИА)
5. **Запослени** **позива** **систем** да обрише **књигу**. (АПСО)
6. **Систем** **приказује** **запосленом** поруку: “**Систем** је обрисао **књигу**.” (ИА)

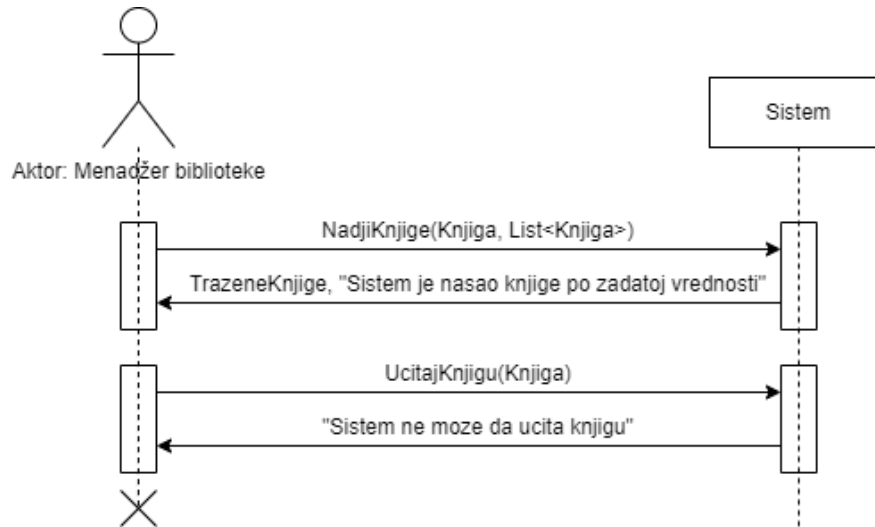


#### Алтернативна сценарија

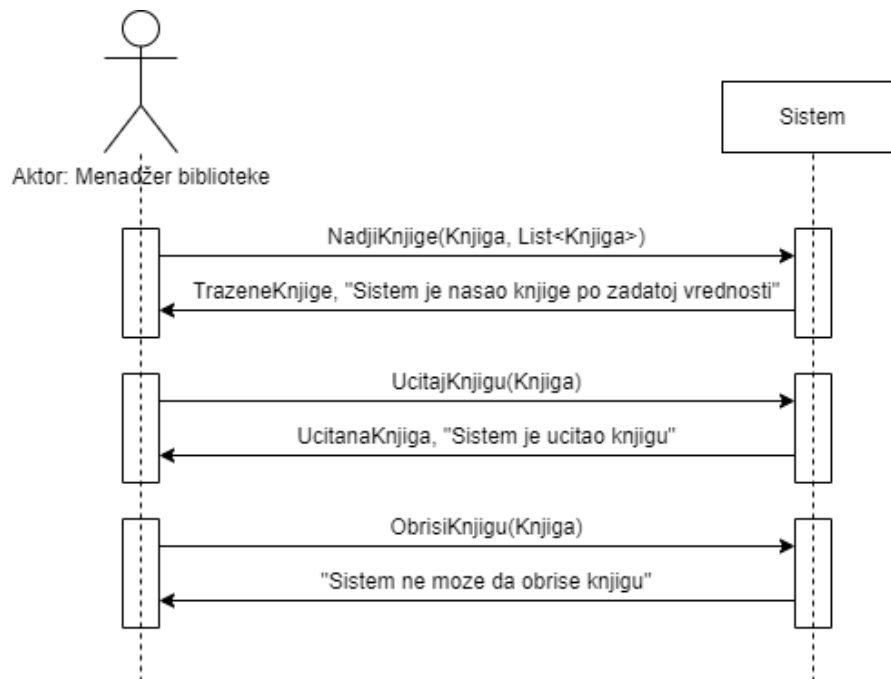
2.1 Уколико **систем** не може да нађе **књиге** он приказује **запосленом** поруку: “**Систем** не може да нађе ниједну **књигу** по задатој вредности”. Прекида се извршење сценарија. (ИА)



4.1 Уколико **систем** не може да учита **књигу** он приказује **запосленом** поруку: “**Систем** не може да учита **књигу**”. Прекида се извршење сценарија. (ИА)



6.1 Уколико **систем** не може да обрише **књигу** он приказује **запосленом** поруку: “**Систем** не може да обрише **књигу**”. (ИА)

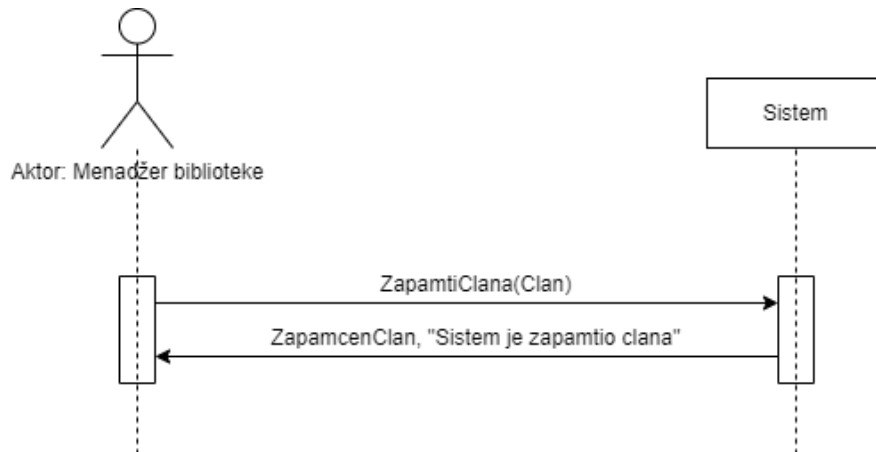


Са наведених секвенцих дијаграма уочавају се 3 системске операције:

1. Signal **NadjiKnjige(Knjiga, List<Knjiga>);**
2. Signal **UcitajKnjigu(Knjiga);**
3. Signal **ObrisiKnjigu(Knjiga).**

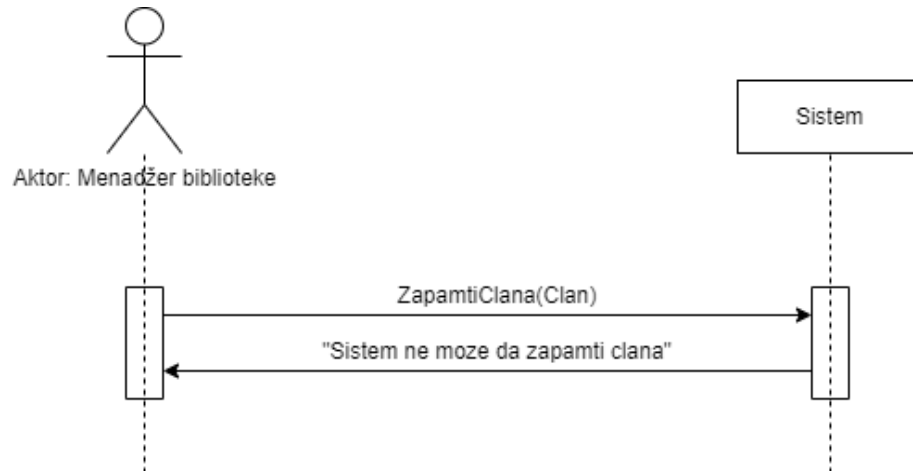
## ДС5: Дијаграм секвенце случаја коришћења – Креирање члана

1. **Запослени** **позива** **систем** да запамти податке о **члану**. (АПСО)
2. **Систем** **приказује** **запосленом** запамћеног **члана** и поруку: “**Систем** је запамтио **члана**”. (ИА)



### Алтернативна сценарија

- 2.1 Уколико **систем** не може да запамти податке о **члану** он приказује **запосленом** поруку “**Систем** не може да запамти **члана**”. (ИА)



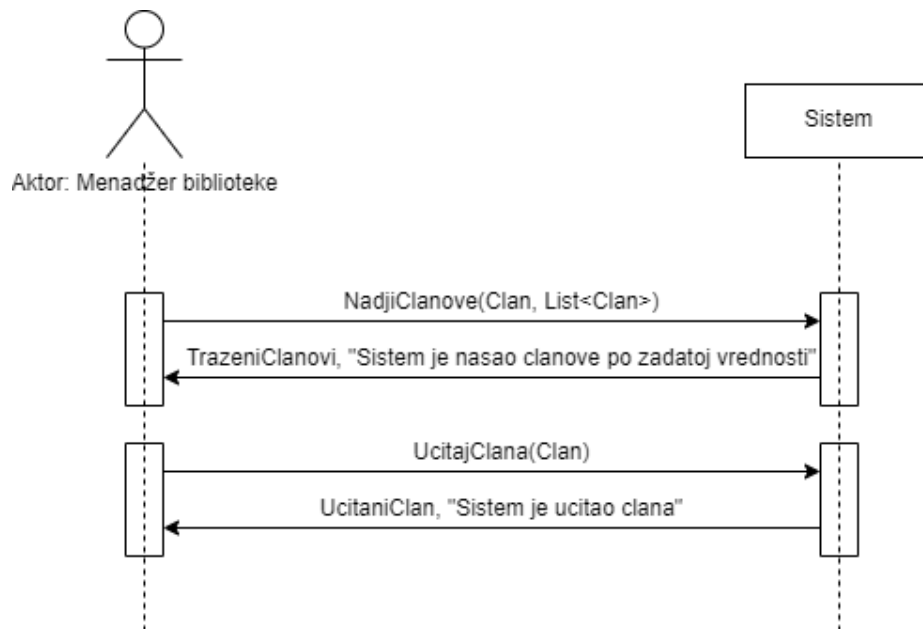
Са наведених секвенцих дијаграма уочава се 1 системска операција:

1. Signal **ZapamtiClana(Clan)**.



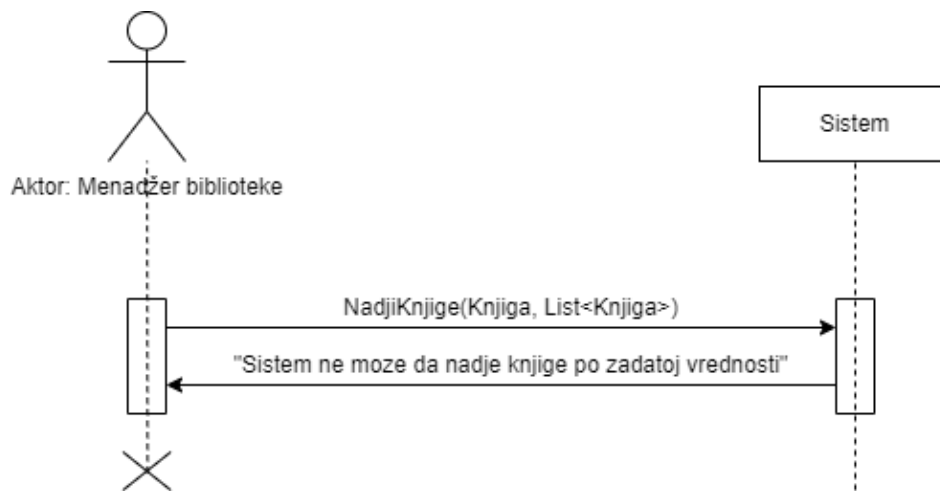
## ДС6: Дијаграм секвенце случаја коришћења – Претраживање члана

1. **Запослени** **позива** **систем** да нађе **чланове** по задатој вредности. (АПСО)
2. **Систем** **приказује** **запосленом** податке о **члановима** и поруку: “**Систем** је нашао **чланове** по задатој вредности”. (ИА)
3. **Запослени** **позива** **систем** да учита **члана**. (АПСО)
4. **Систем** **приказује** **запосленом** податке о **члану** и поруку: “**Систем** је успешно учитао **члана**.”. (ИА)

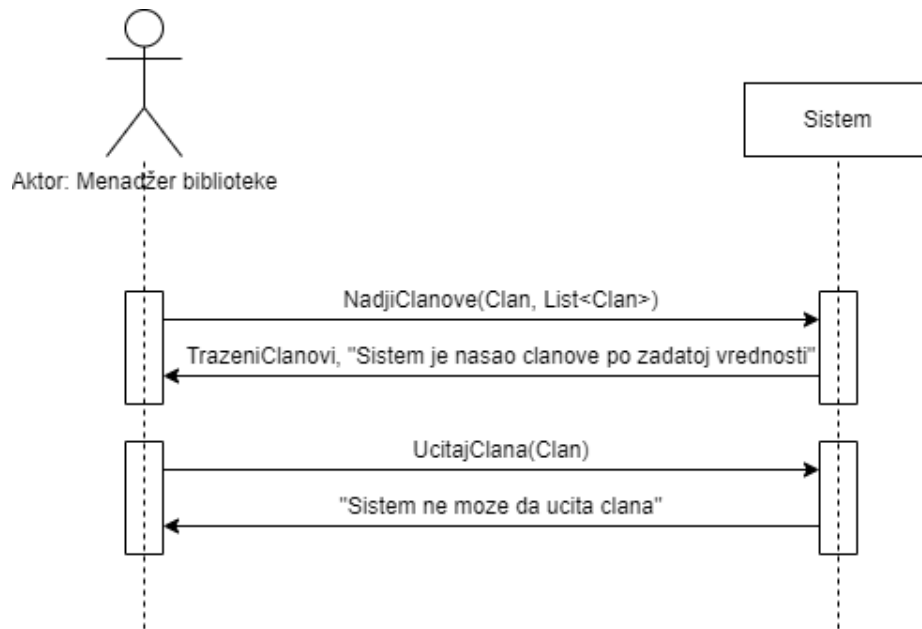


### Алтернативна сценарија

2.1 Уколико **систем** не може да нађе **чланове** он приказује **запосленом** поруку: “**Систем** не може да нађе ниједног **члана** по задатој вредности”. Прекида се извршење сценарија. (ИА)



4.1 Уколико **систем** не може да учита **члана** он приказује **запосленом** поруку:  
“**Систем** не може да учита **члана**”. (ИА)

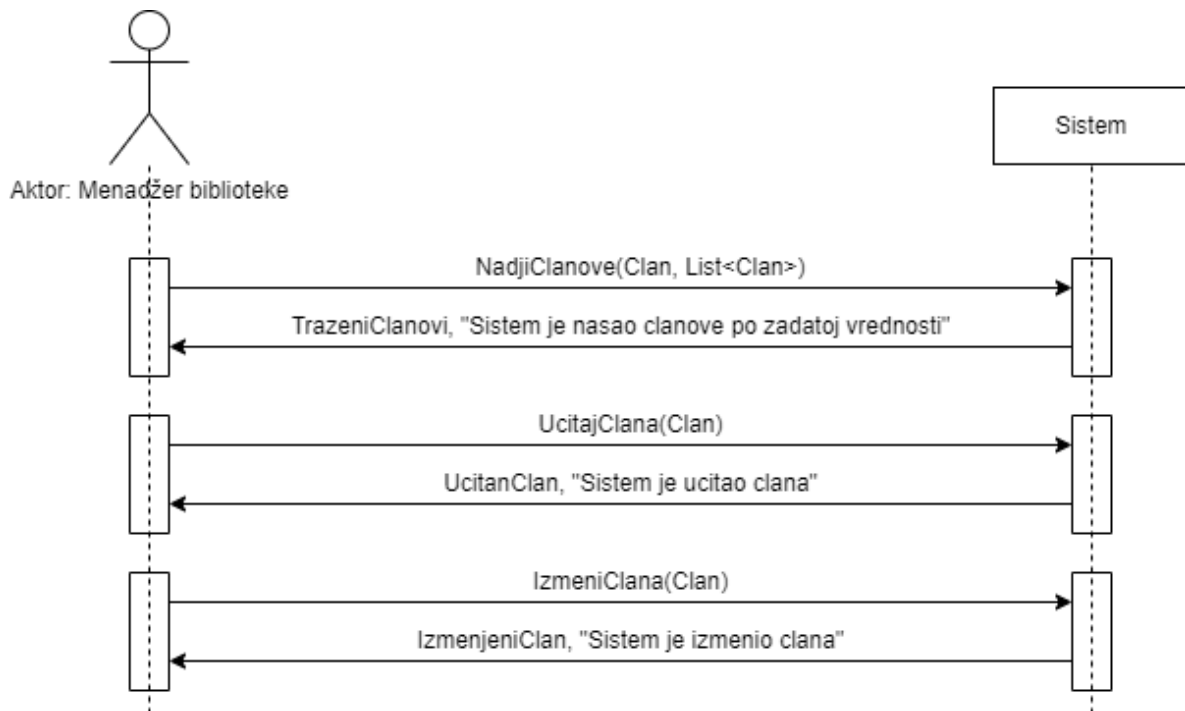


Са наведених секвенцих дијаграма уочавају се 2 системске операције:

1. Signal **NadjiClanove(Clan, List<Clan>);**
2. Signal **UcitajClana(Clan).**

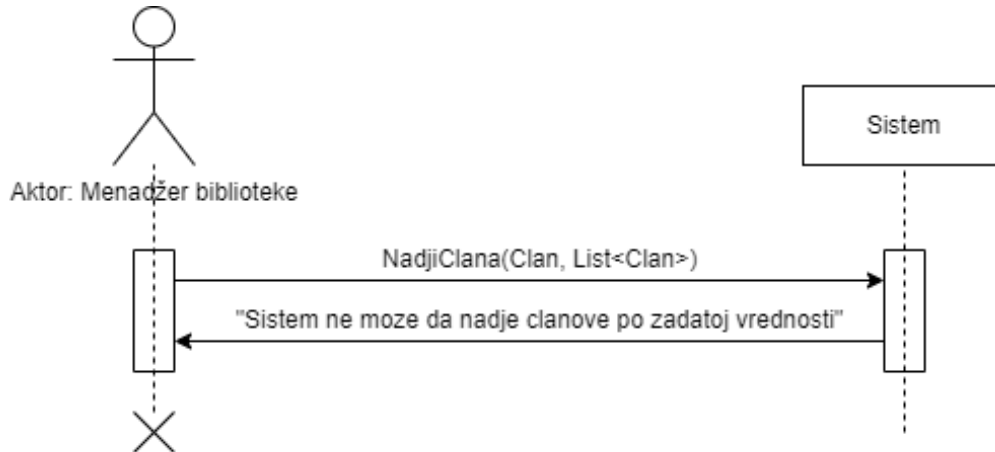
## ДС7: Дијаграм секвенце случаја коришћења – Промена података о члану

1. **Запослени** **позива** **систем** да нађе **чланове** по задатој вредности. (АПСО)
2. **Систем** **приказује** **запосленом** податке о **члановима** и поруку: “**Систем** је нашао **чланове** по задатој вредности”. (ИА)
3. **Запослени** **позива** **систем** да учита **члана**. (АПСО)
4. **Систем** **приказује** **запосленом** податке о **члану** и поруку: “**Систем** је успешно учитао **члана**.”. (ИА)
5. **Запослени** **позива** **систем** да измени податке о **члану**. (АПСО)
6. **Систем** **приказује** **запосленом** измењеног **члана** и поруку: “**Систем** је изменио **члана**.” (ИА)

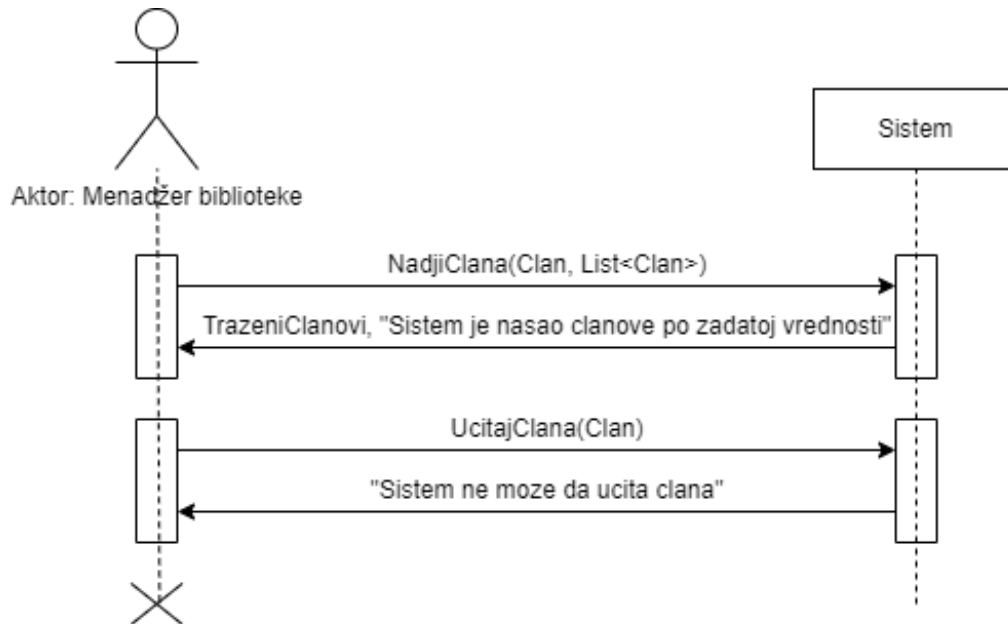


## Алтернативна сценарија

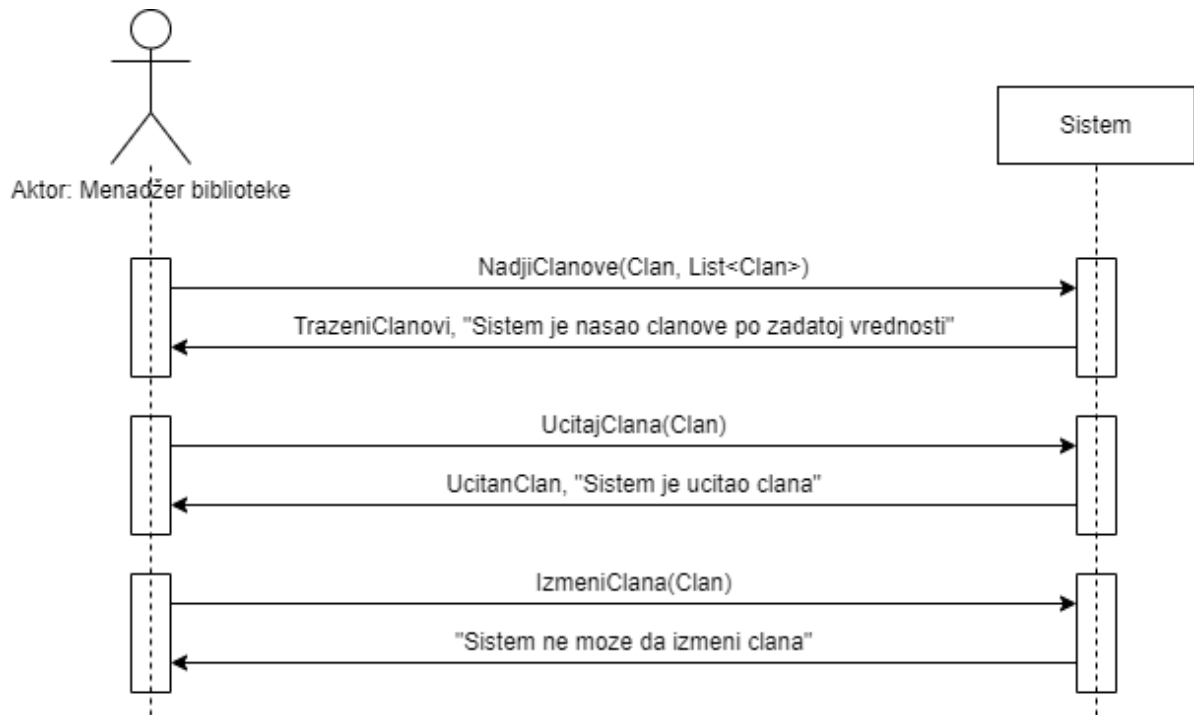
2.1 Уколико **систем** не може да нађе **чланове** он приказује **запосленом** поруку: “**Систем** не може да нађе ниједног **члана** по задатој вредности”. Прекида се извршење сценарија. (ИА)



4.1 Уколико **систем** не може да учита **члана** он приказује **запосленом** поруку: “**Систем** не може да учита **члана**”. Прекида се извршење сценарија. (ИА)



6.1 Уколико **систем** не може да измени податке о **члану** он приказује **запосленом** поруку “**Систем** не може да измени **члана**”. (ИА)

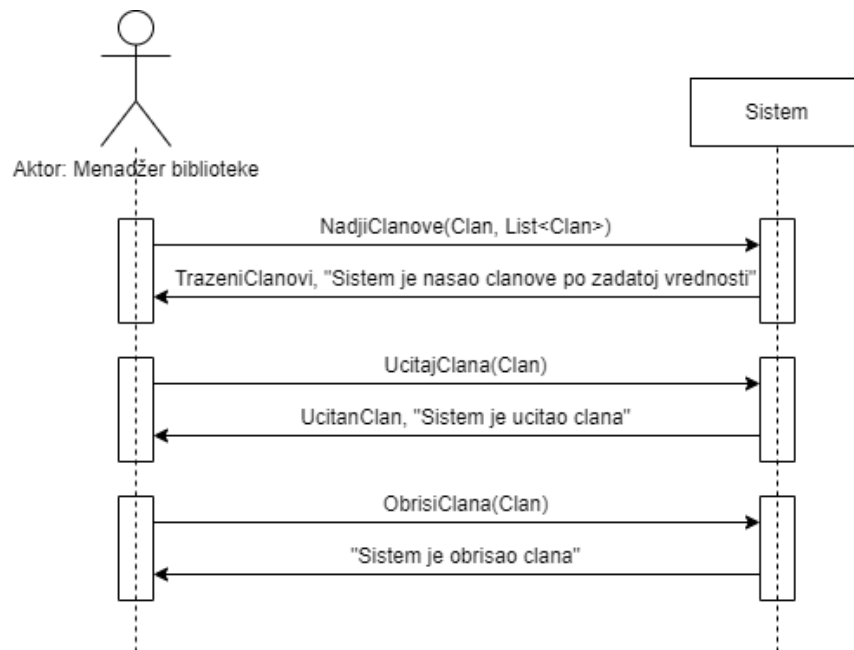


Са наведених секвенцих дијаграма уочавају се 3 системске операције:

1. Signal **NadjiClanove(Clan, List<Clan>);**
2. Signal **UcitajClana(Clan);**
3. Signal **IzmeniClana(Clan).**

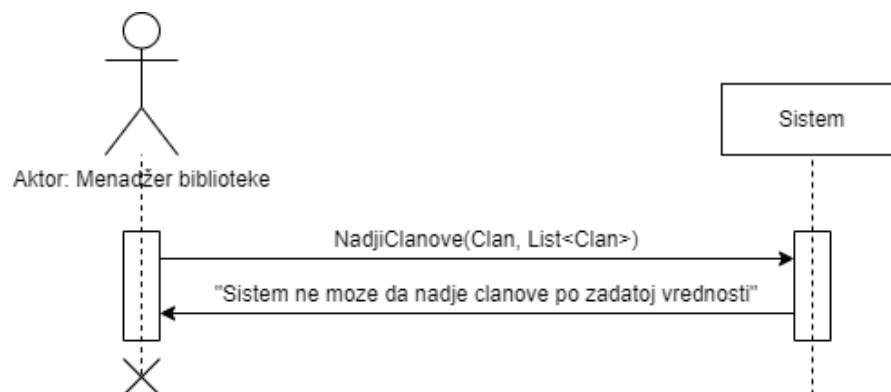
## ДС8: Дијаграм секвенце случаја коришћења – Брисање члана

1. **Запослени** **позива** **систем** да нађе **чланове** по задатој вредности. (АПСО)
2. **Систем** **приказује** **запосленом** податке о **члановима** и поруку: “**Систем** је нашао **чланове** по задатој вредности”. (ИА)
3. **Запослени** **позива** **систем** да учита **члана**. (АПСО)
4. **Систем** **приказује** **запосленом** **члана** и поруку: “**Систем** је успешно учитао **члана**” (ИА)
5. **Запослени** **позива** **систем** да обрише **члана**. (АПСО)
6. **Систем** **приказује** **запосленом** поруку: “**Систем** је обрисао **члана**.” (ИА)

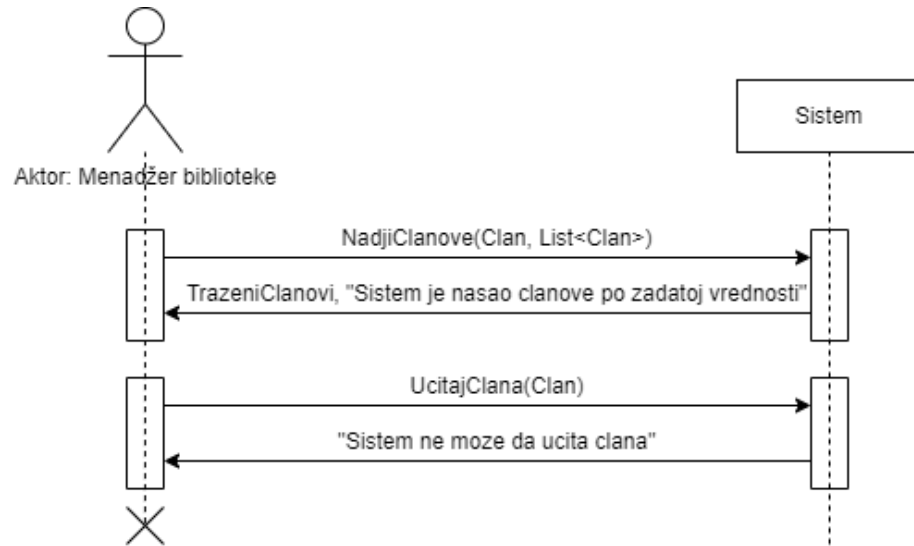


### Алтернативна сценарија

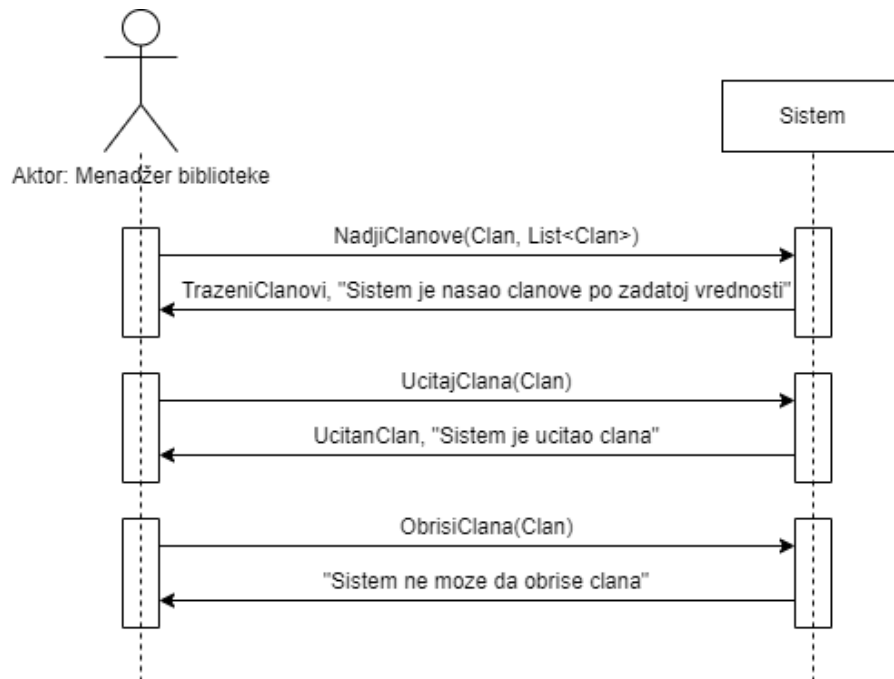
2.1 Уколико **систем** не може да нађе **чланове** он приказује **запосленом** поруку: “**Систем** не може да нађе ниједног **члана** по задатој вредности”. Прекида се извршење сценарија. (ИА)



4.1 Уколико **систем** не може да учита **члана** он приказује **запосленом** поруку: “**Систем** не може да учита **члана**”. Прекида се извршење сценарија. (ИА)



6.1 Уколико **систем** не може да обрише **члана** он приказује **запосленом** поруку: “**Систем** не може да обрише **члана**”. (ИА)

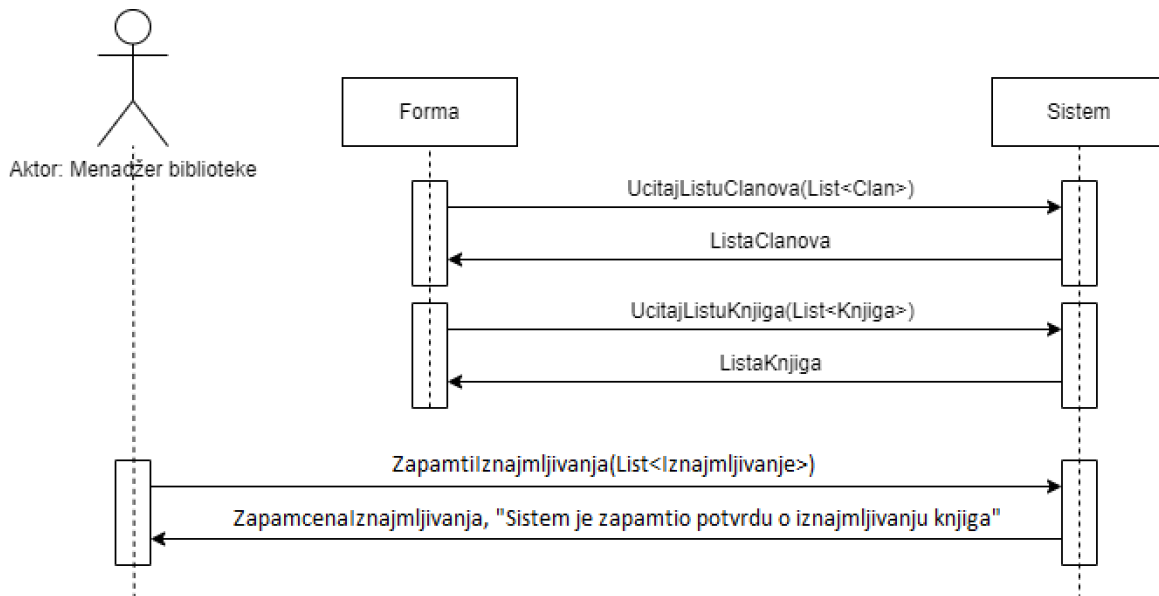


Са наведених секвенцих дијаграма уочавају се 3 системске операције:

1. Signal **NadjiClanove(Clan, List<Clan>);**
2. Signal **UcitajClana(Clan);**
3. Signal **ObrisiClana(Clan).**

### ДС9: Дијаграм секвенце случаја коришћења – Креирање потврде о изнајмљивању књига (сложен СК)

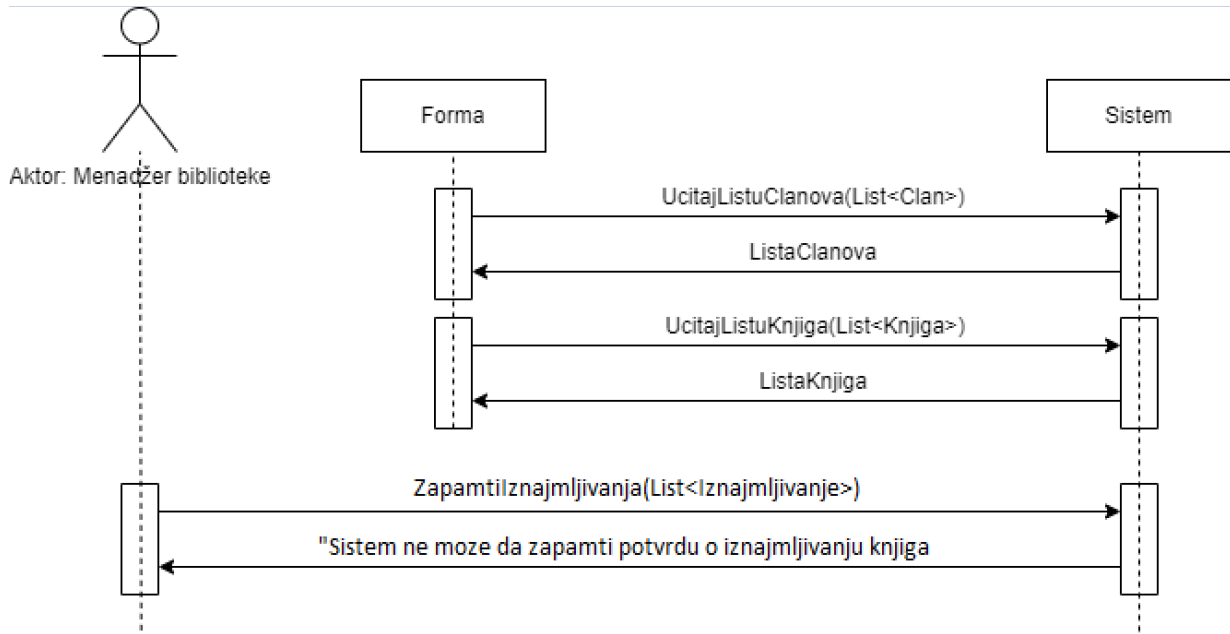
1. **Форма** **позива** **систем** да учита листу чланова. (АПСО)
2. **Систем** **враћа** **форми** листу чланова. (ИА)
3. **Форма** **позива** **систем** да учита листу књига. (АПСО)
4. **Систем** **враћа** **форми** листу књига. (ИА)
5. **Запослени** **позива** **систем** да запамти податке о **потврди о изнајмљивању књига**. (АПСО)
6. **Систем** **приказује** **запосленом** запамћену **потврду о изнајмљивању књига** и поруку: **"Систем је запамтио потврду о изнајмљивању књига"**. (ИА)





## Алтернативна сценарија

6.1 Уколико **систем** не може да запамти податке о **потврди о изнајмљивању књига** он приказује **запосленом** поруку “**Систем** не може да запамти **потврду о изнајмљивању књига**”. (ИА)

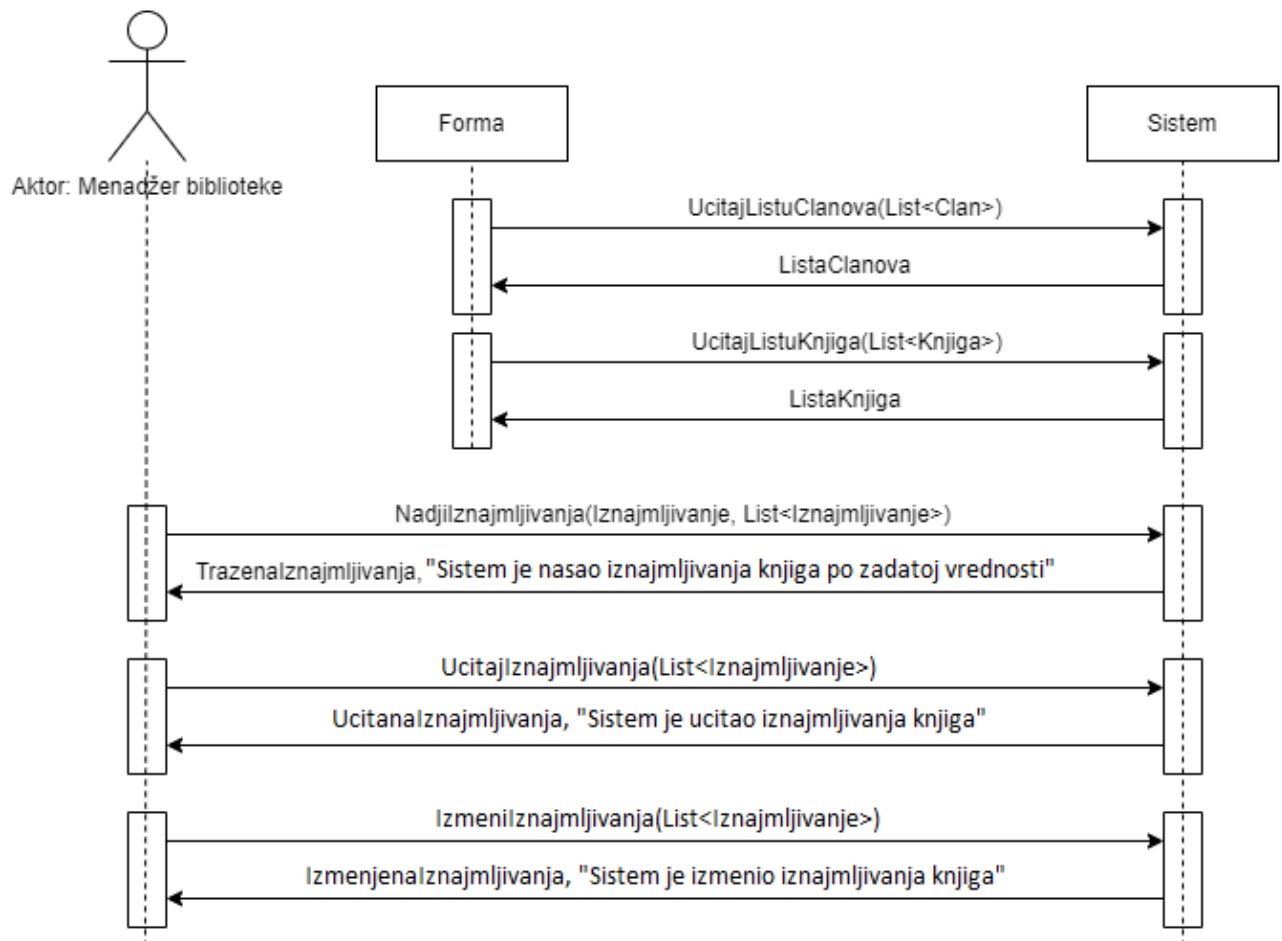


Са наведених секвенчних дијаграма уочавају се 4 системске операције:

1. Signal **UcitajListuClanova(List<Clan>);**
2. Signal **UcitajListuZgradaBiblioteke(List<ZgradaBiblioteke>);**
3. Signal **UcitajListuKnjiga(List<Knjiga>);**
4. Signal **Zapamtilznajmljivanja(List<Iznajmljivanje>).**

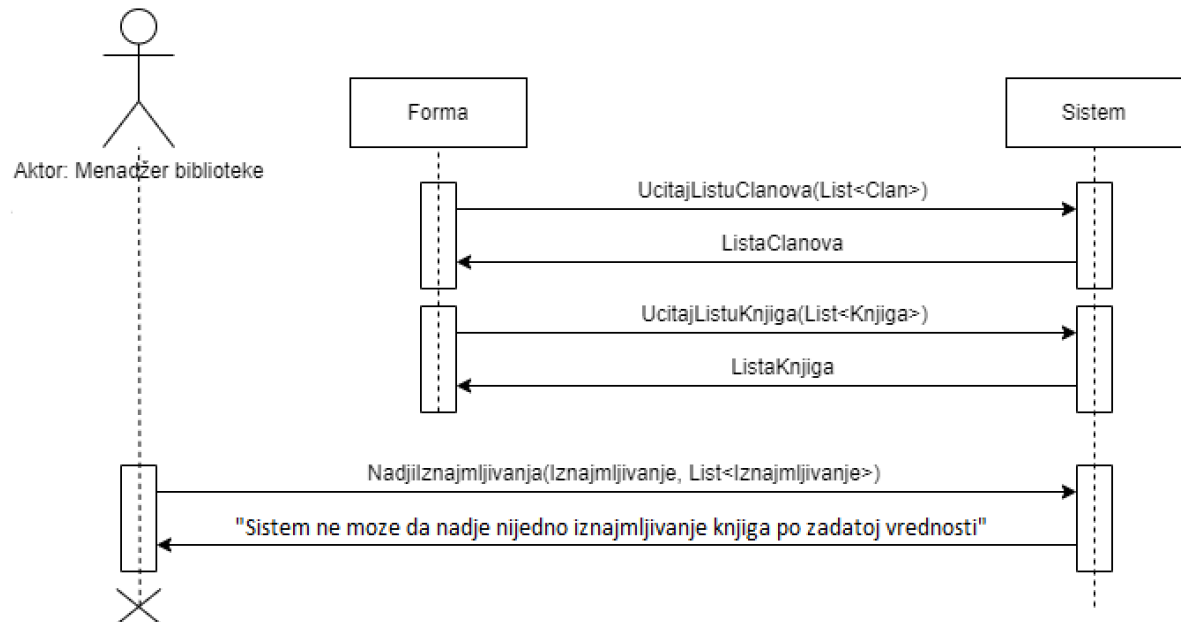
## ДС10: Дијаграм секвенце случаја коришћења – Промена потврде о изнајмљивању књига (сложен СК)

1. **Форма** **позива** **систем** да учита листу чланова. (АПСО)
2. **Систем** **враћа форми** листу чланова. (ИА)
3. **Форма** **позива** **систем** да учита листу књига. (АПСО)
4. **Систем** **враћа форми** листу књига. (ИА)
5. **Запослени** **позива** **систем** да нађе **изнајмљивања књига** по задатој вредности. (АПСО)
6. **Систем** **приказује** **запосленом** **изнајмљивања књига** и поруку: “**Систем** је нашао **изнајмљивања књига** по задатој вредности”. (ИА)
7. **Запослени** **позива** **систем** да учита **изнајмљивања књига**. (АПСО)
8. **Систем** **приказује** **запосленом** податке о **изнајмљивањима књига** и поруку: “**Систем** је успешно прочитао **изнајмљивања књига**”. (ИА)
9. **Запослени** **позива** **систем** да измени податке о **изнајмљивању књига**. (АПСО)
10. **Систем** **приказује** **запосленом** измењена **изнајмљивања књига** и поруку: “**Систем** је изменио **изнајмљивања књига**.” (ИА)

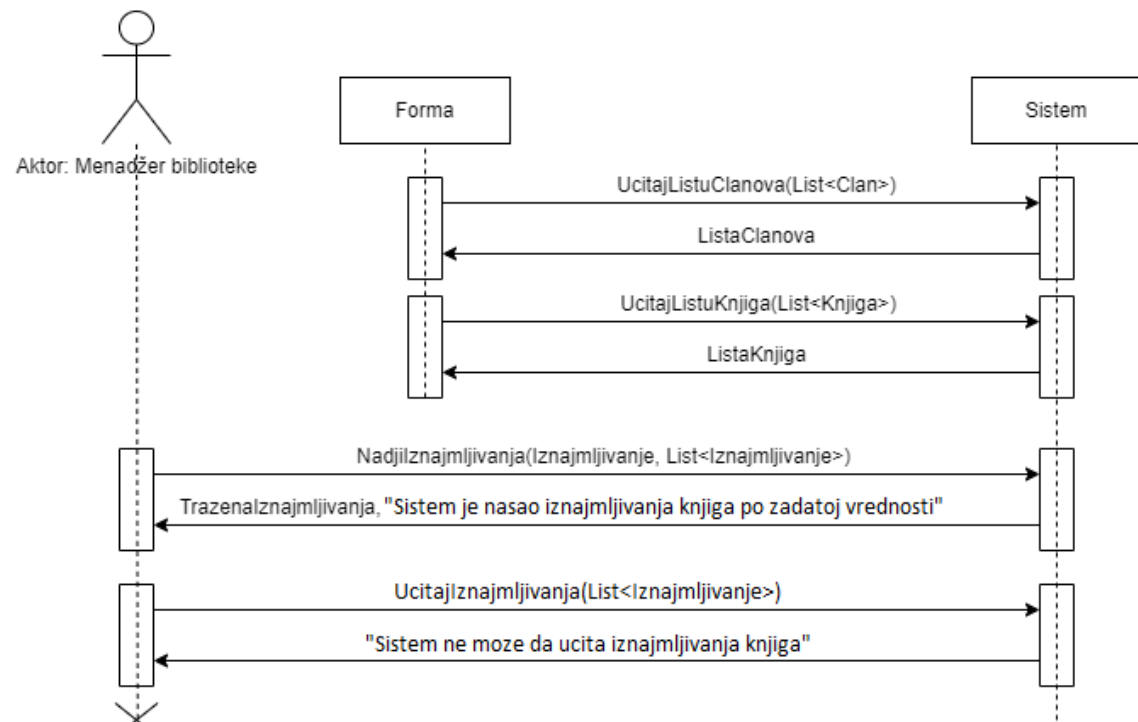


Алтернативна сценарија

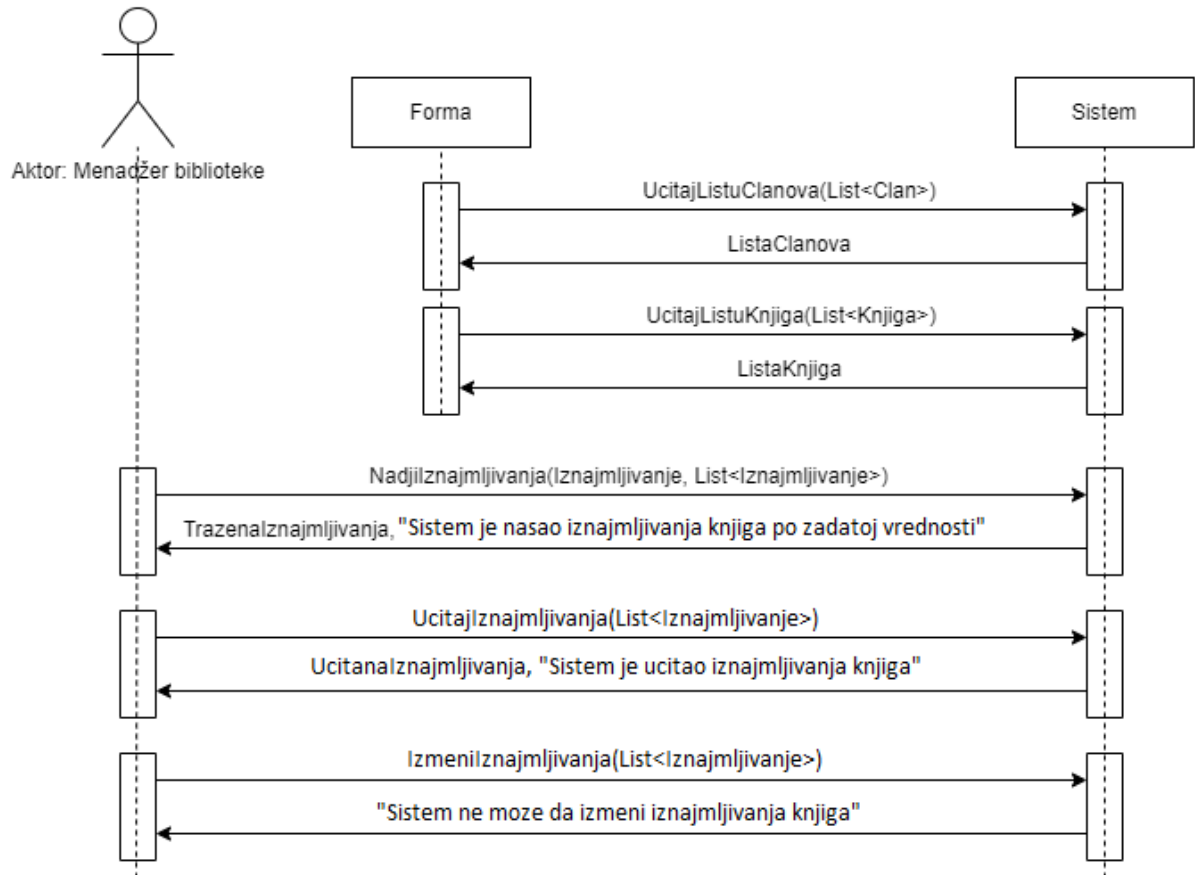
6.1 Уколико **систем** не може да нађе **изнајмљивања књига** он приказује **запосленом** поруку: “**Систем** не може да нађе ниједно **изнајмљивање књига** по задатој вредности”. Прекида се извршење сценарија. (ИА)



8.1 Уколико **систем** не може да учита **изнајмљивања књига** он приказује **запосленом** поруку: “**Систем** не може да учита **изнајмљивања књига**”. Прекида се извршење сценарија. (ИА)



10.1 Уколико **систем** не може да измени податке о **изнајмљивањима књига** он приказује **запосленом** поруку “**Систем** не може да измени **изнајмљивања књига**”. (ИА)



Са наведених секвенцих дијаграма уочавају се 4 системске операције:

1. Signal **UcitajListuClanova(List<Clan>)**;
2. Signal **UcitajListuKnjiga(List<Knjiga>)**;
3. Signal **Nadjilznajmljivanja(Iznajmljivanje, List<Iznajmljivanje>)**;
4. Signal **UcitajIznajmljivanja(List<Iznajmljivanje>)**;
5. Signal **IzmeniIznajmljivanja(List<Iznajmljivanje>)**.

На основу анализе сценарија добијено је 17 системских операција:

1. Signal **ZapamtiKnjigu**(Knjiga);
2. Signal **ZapamtiClana**(Clan);
3. Signal **NadjiKnjige**(Knjiga, List<Knjiga>);
4. Signal **UcitajKnjigu**(Knjiga);
5. Signal **IzmeniKnjigu**(Knjiga);
6. Signal **NadjiClanove**(Clan, List<Clan>);
7. Signal **UcitajClana**(Clan);
8. Signal **IzmeniClana**(Clan);
9. Signal **ObrisiKnjigu**(Knjiga);
10. Signal **ObrisiClana**(Clan);
11. Signal **UcitajListuKnjiga**(List<Knjiga>);
12. Signal **UcitajListuClanova**(List<Clan>);
13. Signal **UcitajListuAutora**(List<Autor>);
14. Signal **Zapamtilznajmljivanja**(List<Iznajmljivanje>);
15. Signal **Nadjilznajmljivanja**(Iznajmljivanje, List<Iznajmljivanje>);
16. Signal **UcitajIznajmljivanja**(List<Iznajmljivanje>);
17. Signal **Izmenilznajmljivanja**(List<Iznajmljivanje>).

## 2.2 Понашање софтверског система – Дефинисање уговора о системским операцијама

Уговор УГ1: ZapamtiKnjigu(Knjiga) Signal

Веза са СК: СК1

Предуслови: Вредносна и структурна ограничења над објектом **Knjiga** морају бити задовољена.

Постуслови: Подаци о књизи су запамћени.

Уговор УГ2: ZapamtiClana(Clan) Signal

Веза са СК: СК5

Предуслови: Вредносна и структурна ограничења над објектом **Clan** морају бити задовољена.

Постуслови: Подаци о члану су запамћени.

Уговор УГ3: NadjiKnjige(Knjiga, List<Knjiga>) Signal

Веза са СК: СК2, СК3, СК4

Предуслови:

Постуслови:

Уговор УГ4: UcitajKnjigu(Knjiga) Signal

Веза са СК: СК2, СК3, СК4

Предуслови:

Постуслови:

Уговор УГ5: IzmeniKnjigu(Knjiga) Signal

Веза са СК: СК3

Предуслови: Вредносна и структурна ограничења над објектом **Knjiga** морају бити задовољена.

Постуслови: Подаци о књизи су измењени.

Уговор УГ6: NadjiClanove(Clan, List<Clan>) Signal

Веза са СК: СК6, СК7, СК8

Предуслови:

Постуслови:

Уговор УГ7: UcitajClana(Clan) Signal

Веза са СК: СК6, СК7, СК8

Предуслови:

Постуслови:

Уговор УГ8: IzmeniClana(Clan) Signal

Веза са СК: СК7

Предуслови: Вредносна и структурна ограничења над објектом **Clan** морају бити задовољена.

Постуслови: Подаци о члану су измењени.

Уговор УГ9: ObrisiKnjigu(Knjiga) Signal

Веза са СК: СК4

Предуслови: Структурна ограничења над објектом **Knjiga** морају бити задовољена.

Постуслови: Књига је обрисана.

Уговор УГ10: ObrisiClana(Clan) Signal

Веза са СК: СК8

Предуслови: Структурна ограничења над објектом **Clan** морају бити задовољена.

Постуслови: Члан је обрисан.

Уговор УГ11: UcitajListuKnjiga(List<Knjiga>) Signal

Веза са СК: СК9, СК10

Предуслови:

Постуслови:

Уговор УГ12: UcitajListuClanova(List<Clan>) Signal

Веза са СК: СК9, СК10

Предуслови:

Постуслови:

Уговор УГ13: UcitajListuAutora(List<Autor>) Signal

Веза са СК: СК1, СК3

Предуслови:

Постуслови:

Уговор УГ14: Zapamtilznajmljivanja(List<Iznajmljivanje>) Signal

Веза са СК: СК9

Предуслови: Вредносна и структурна ограничења над објектом **Iznajmljivanje** морају бити задовољена.

Постуслови: Подаци о изнајмљивању су запамћени.

Уговор УГ15: Nadjilznajmljivanja(Iznajmljivanje, List<Iznajmljivanje>) Signal

Веза са СК: СК10

Предуслови:

Постуслови:

Уговор УГ16: UcitajIznajmljivanje(Iznajmljivanje) Signal

Веза са СК: СК10

Предуслови:

Постуслови:

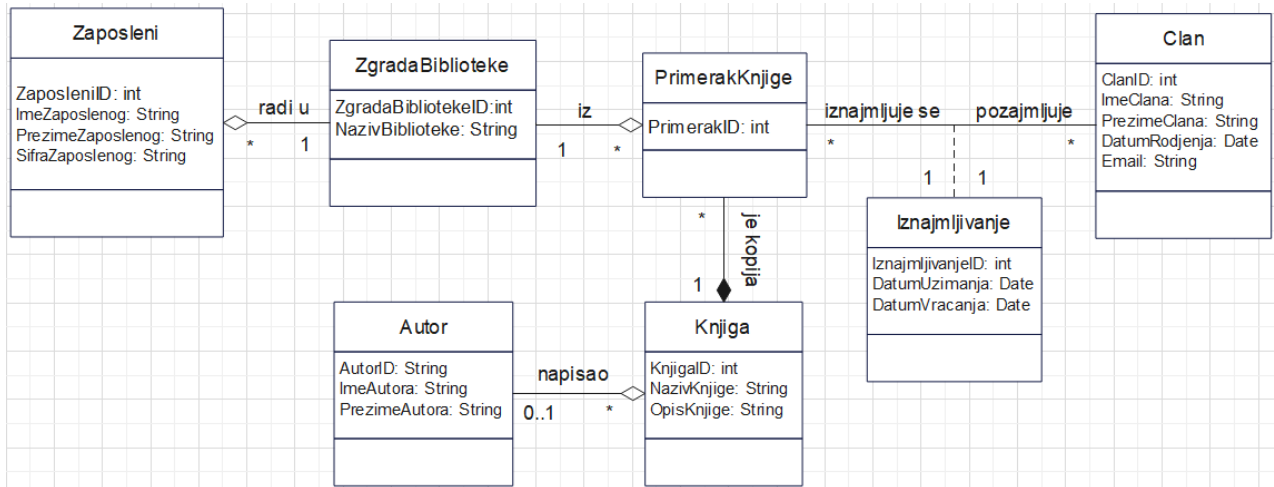
Уговор УГ17: Izmenilznajmljivanja(List<Iznajmljivanje>) Signal

Веза са СК: СК10

Предуслови: Вредносна и структурна ограничења над објектом **Iznajmljivanje** морају бити задовољена.

Постуслови: Подаци о изнајмљивању су измењени.

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



## 2.4 Struktura softverskog sistema – relacioni model

Autor(AutorID, ImeAutora, PrezimeAutora)  
 Clan(ClanID, ImeClana, PrezimeClana, DatumRodjenja, Email)  
 ZgradaBiblioteke(ZgradaBibliotekeID, NazivBiblioteke)  
 Zaposleni(ZaposleniID, ImeZaposlenog, PrezimeZaposlenog, SifraZaposlenog, ZgradaBibliotekeID)  
 Knjiga(KnjigaID, NazivKnjige, OpisKnjige, AutorID)  
 PrimerakKnjige(KnjigaID, PrimerakID, ZgradaBibliotekeID)  
 Iznajmljivanje(IznajmljivanjeID, KnjigaID, PrimerakID, ClanID, DatumUzimanja, DatumVracanja)



Tabela Autor		Prosto vrednosno ograničenje		Složeno vrednosno ograničenje		Strukturno ograničenje
Atributi	Ime	Tip atributa	Vrednost atributa	Međuzavisnost atributa jedne tabele	Međuzavisnost atributa više tabela	INSERT /  UPDATE CASCADE Knjiga  DELETE RESTRICTED Knjiga
	AutorID	int	not null and > 0			
	ImeAutora	String				
	PrezimeAutora	String	not null			

Tabela Clan		Prosto vrednosno ograničenje		Složeno vrednosno ograničenje		Strukturno ograničenje
Atributi	Ime	Tip atributa	Vrednost atributa	Međuzavisnost atributa jedne tabele	Međuzavisnost atributa više tabela	INSERT /  UPDATE CASCADE Iznajmljivanje  DELETE RESTRICTED Iznajmljivanje
	ClanID	int	not null and > 0			
	ImeClana	String	not null			
	PrezimeClana	String	not null			
	DatumRodjenja	Date	not null			
	Email	String				

Tabela ZgradaBiblioteke		Prosto vrednosno ograničenje		Složeno vrednosno ograničenje		Strukturno ograničenje
Atributi	Ime	Tip atributa	Vrednost atributa	Međuzavisnost atributa jedne tabele	Međuzavisnost atributa više tabela	INSERT /  UPDATE CASCADE PrimerakKnjige, Zaposleni  DELETE RESTRICTED PrimerakKnjige, Zaposleni
	ZgradaBibliotekeID	int	not null and > 0			
	NazivBiblioteke	String	not null			

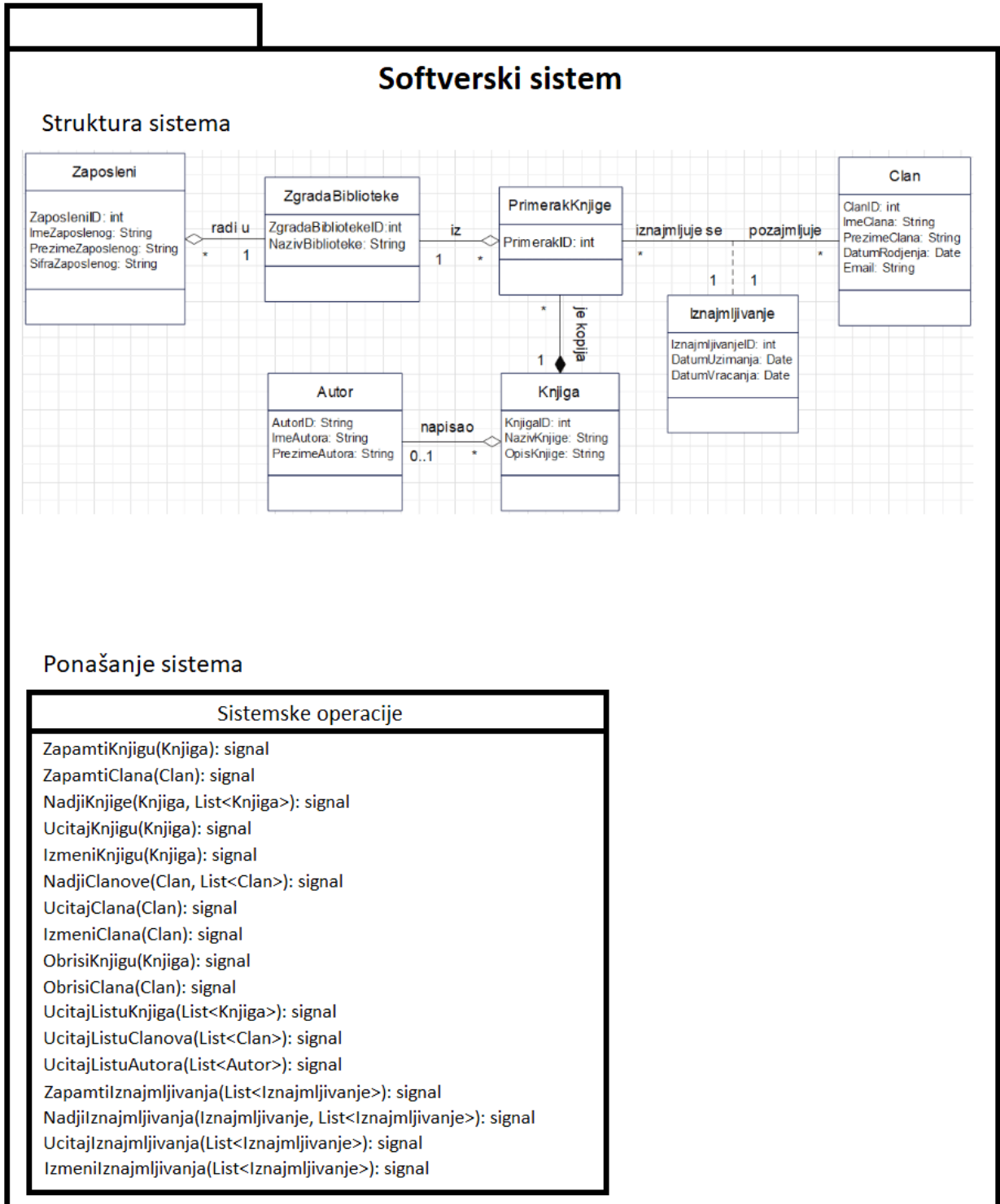
Tabela Zaposleni		Prosto vrednosno ograničenje		Složeno vrednosno ograničenje		Strukturno ograničenje
Atributi	Ime	Tip atributa	Vrednost atributa	Međuzavisnost atributa jedne tabele	Međuzavisnost atributa više tabela	INSERT RESTRICTED ZgradaBiblioteke
	ZaposleniID	int	not null and > 0			UPDATE RESTRICTED ZgradaBiblioteke
	ImeZaposlenog	String	not null			DELETE /
	PrezimeZaposlenog	String	not null			
	SifraZaposlenog	String	not null			
	ZgradaBibliotekeID	int	not null			

Tabela Knjiga		Prosto vrednosno ograničenje		Složeno vrednosno ograničenje		Strukturno ograničenje
Atributi	Ime	Tip atributa	Vrednost atributa	Međuzavisnost atributa jedne tabele	Međuzavisnost atributa više tabela	INSERT RESTRICTED Autor
	KnjigaID	int	not null and > 0			UPDATE RESTRICTED Autor CASCADE
	NazivKnjige	String	not null			PrimerakKnjige
	OpisKnjige	String				DELETE RESTRICTED PrimerakKnjige
	AutorID	int				

Tabela PrimerakKnjige		Prosto vrednosno ograničenje		Složeno vrednosno ograničenje		Strukturno ograničenje
Atributi	Ime	Tip atributa	Vrednost atributa	Međuzavisnost atributa jedne tabele	Međuzavisnost atributa više tabela	INSERT RESTRICTED ZgradaBiblioteke, Knjiga
	KnjigaID	int	not null and > 0			UPDATE RESTRICTED ZgradaBiblioteke, Knjiga CASCADE
	PrimerakID	int	not null and > 0			Iznajmljivanje
	ZgradaBibliotekeID	int	> 0			DELETE RESTRICTED Iznajmljivanje

Tabela Iznajmljivanje		Prosto vrednosno ograničenje		Složeno vrednosno ograničenje		Strukturno ograničenje
Atributi	Ime	Tip atributa	Vrednost atributa	Međuzavisnost atributa jedne tabele	Međuzavisnost atributa više tabela	INSERT RESTRICTED PrimerakKnjige, Clan  UPDATE RESTRICTED PrimerakKnjige, Clan  DELETE /
	IznajmljivanjeID	int	not null and > 0			
	KnjigaID	int	not null and > 0			
	PrimerakID	int	not null and > 0			
	ClanID	int	not null and > 0			
	DatumUzimanja	Date	not null			
	DatumVracanja	Date				

Kao rezultat analize scenarija SK i pravљења konceptualnog modela dobiја se logicka struktura i ponašaње softverskog sistema:



## Пројектовање

Фаза пројектовања има улогу у описивању физичке структуре и понашања софтверског система.

### 3.1 Архитектура софтверског система

Архитектура софтверског система је тринивојска и састоји се од следећих нивоа:

- Кориснички интерфејс
- Апликациона логика
  - Контролер апликационе логике
  - Пословна логика – структура и понашање
  - Брокер базе података
- Складиште података

Кориснички интерфејс се налази на корисничкој страни, док су апликациона логика и складиште података нивои серверске стране апликације.



### 3.2 Пројектовање корисничког интерфејса

Кориснички интерфејс представља реализацију улаза и/или излаза софтверског система. Његову структуру чине екранска форма и контролер корисничког интерфејса.



### 3.2.1 Пројектовање екранских форми

Кориснички интерфејс је дефинисан преко скупа екранских форми.

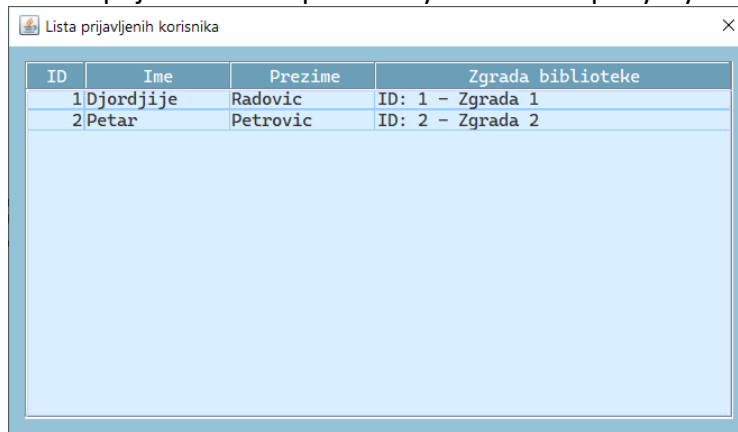
На серверској страни, при покретању апликације отвара се екранска форма:



Након покретања сервера, кликом на одговарајуће дугме, екранска форма приказује кориснику поруку:

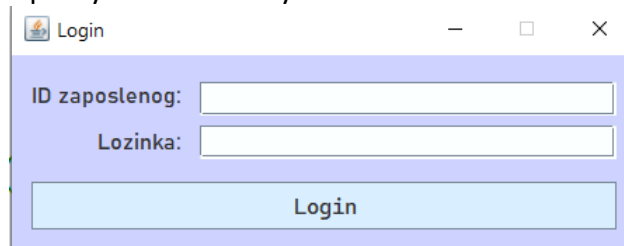


Као што се види у горњем левом углу форме, корисник има могућности да управља конфигурацијама апликације везаних за базу података. Такође, постоји опција прегледа листе свих пријављених корисника у било ком тренутку:



ID	Ime	Prezime	Zgrada biblioteke
1	Djordjije	Radovic	ID: 1 - Zgrada 1
2	Petar	Petrovic	ID: 2 - Zgrada 2

На клијентској страни, приликом покретања апликације, кориснику се отвара екранска форма за приступ свом налогу:



Login

ID zaposlenog:

Lozinka:

Login

Након пријављивања у систем, почетна екранска форма апликације за корисника изгледа овако:



Knjige Clanovi Iznajmljivanje Log out

Zgrada biblioteke: ID: 1 - Zgrada 1

Prijavljeni zaposleni: Djordjije Radovic



На форми су приказане информације о пријављеном запосленом и згради библиотеке у којој ради.

## СК1: Случај коришћења – Креирање књиге – пројектовање екранске форме

### Назив СК

Креирање књиге

### Актери СК

Запослени

### Учесници СК

Запослени и систем (програм)

**Предуслов:** Систем је укључен и **Запослени** је пријављен под својом шифром. Систем приказује форму за рад са књигом. Учитана је листа аутора.

Knjiga

Naslov:

Opis:

Autor:

Trenutno na stanju:

Broj primeraka knjige koji se dodaje:

SACUVAJ

### Основни сценарио СК

1. **Запослени** уноси податке за креирање књиге. (АПУСО)

Knjiga

Naslov:

Opis:

Autor:

Trenutno na stanju:

Broj primeraka knjige koji se dodaje:

SACUVAJ

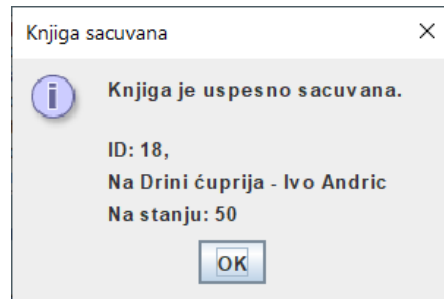


2. **Запослени контролише** претходно унете податке за креирање **књиге**. (АНСО)
3. **Запослени позива систем** да запамти податке о **књизи**. (АПСО)

*Опис акције:* Запослени кликом на дугме „SACUVAJ“ позива системску операцију

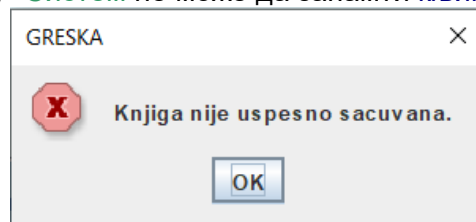
**ZapamtiKnjigu(Knjiga).**

4. **Систем памти** податке о **књизи**. (СО)
5. **Систем приказује запосленом** запамћену **књигу** и поруку: “Систем је запамтио **књигу**”. (ИА)



#### Алтернативна сценарија

- 5.1 Уколико **систем** не може да запамти податке о **књизи** он приказује **запосленом** поруку “Систем не може да запамти **књигу**”. (ИА)



## СК2: Случај коришћења – Претраживање књиге – пројектовање екранске форме

### Назив СК

Претраживање књиге

### Актери СК

Запослени

### Учесници СК

Запослени и систем (програм)

**Предуслов:** Систем је укључен и **запослени** је пријављен под својом шифром. Систем приказује форму за рад са књигом.

Pretraga knjiga

Naslov:

ID	Naslov	Autor
1	Sidarta	Hermann Hesse
2	Covek koji se smeje	Victor Hugo
4	1984	George Orwell
5	Biblija	
9	Seobe	Miloš Crnjanski
10	Besnilo	Borislav Pekić
11	Prokleta avlija	Ivo Andrić
15	Stepski vuk	Hermann Hesse
16	Nepodnošljiva lakoća postojanja	
18	Na Drini ćuprija	Ivo Andrić

### Основни сценарио СК

1. **Запослени уноси** вредност по којој претражује књиге. (АПУСО)

Pretraga knjiga

Naslov:

ID	Naslov	Autor
1	Sidarta	Hermann Hesse
2	Covek koji se smeje	Victor Hugo
4	1984	George Orwell
5	Biblija	
9	Seobe	Miloš Crnjanski
10	Besnilo	Borislav Pekić
11	Prokleta avlija	Ivo Andrić
15	Stepski vuk	Hermann Hesse
16	Nepodnošljiva lakoća postojanja	
18	Na Drini ćuprija	Ivo Andrić

2. **Запослени** **позива** **систем** да нађе **књиге** по задатој вредности. (АПСО)  
*Опис акције:* Запослени кликом на дугме „PRETRAZI“ позива системску операцију NadjiKnjige(Knjiga,List<Knjiga>)
3. **Систем** **тражи** **књиге** по задатој вредности. (СО)
4. **Систем** **приказује** **запосленом** податке о **књигама** и поруку: “**Систем** је нашао **књиге** по задатој вредности”. (ИА)

Pretraga knjiga

Naslov: s PRETRAZI

ID	Naslov	Autor
1	Sidarta	Hermann Hesse
9	Seobe	Miloš Crnjanski
15	Stepski vuk	Hermann Hesse

OTVORI FORMU ZA ODBRANU KNJIGU DODAJ KNJIGU

5. **Запослени** **бира** **књигу**. (АПУСО)
6. **Запослени** **позива** **систем** да учита **књигу**. (АПСО)  
*Опис акције:* Запослени кликом на дугме „OTVORI FORMU ZA ODABRANU KNJIGU“ позива системску операцију UcitajKnjigu(Knjiga)
7. **Систем** **учитава** **књигу**. (СО)
8. **Систем** **приказује** **запосленом** податке о **књизи** и поруку: “**Систем** је успешно учитао **књигу**”. (ИА)

Knjiga

Naslov: Sidarta

Opis: Sidarta, prelijepi sin bramana, nezadovoljan duhovnom ispunjenoscu koju mu pruza tradicionalna vjera krece na put traganja za krajnjom istinom i konacnim prosvjetljenjem, ne odustajuci od svog zivotnog cilja i autenticnosti ni po koju cijenu. U Heseovoj kritici savremene civilizacije i gradjanskog drustva, u njegovom

Autor: Hermann Hesse

Trenutno na stanju: 17

Broj primeraka knjige koji se dodaje: 0

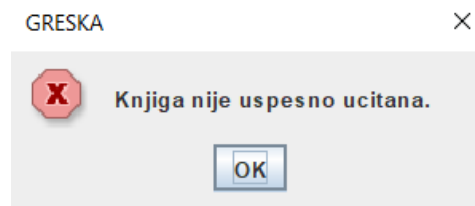
OBRISI SACUVAJ

## Алтернативна сценарија

4.1 Уколико **систем** не може да нађе **књиге** он приказује **запосленом** поруку:  
“**Систем** не може да нађе ниједну **књигу** по задатој вредности”. Прекида се извршење сценарија. (ИА)



8.1 Уколико **систем** не може да учита **књигу** он приказује **запосленом** поруку:  
“**Систем** не може да учита **књигу**”. (ИА)



### СК3: Случај коришћења – Промена података о књизи – пројектовање екранске форме

#### Назив СК

Промена података о књизи

#### Актери СК

Запослени

#### Учесници СК

Запослени и систем (програм)

**Предуслов:** Систем је укључен и запослени је пријављен под својом шифром. Систем приказује форму за рад са књигом. Учитана је листа аутора.

Pretraga knjiga

Naslov:

ID	Naslov	Autor
1	Sidarta	Hermann Hesse
2	Covek koji se smeje	Victor Hugo
4	1984	George Orwell
5	Biblija	
9	Seobe	Miloš Crnjanski
10	Besnilo	Borislav Pekić
11	Prokleta avlija	Ivo Andric
15	Stepski vuk	Hermann Hesse
16	Nepodnošljiva lakoća postojanja	
18	Na Drini ćuprija	Ivo Andric

#### Основни сценарио СК

1. **Запослени** уноси вредност по којој претражује књиге. (АПУСО)

Pretraga knjiga

Naslov: Stepski vuk PRETRAZI

ID	Naslov	Autor
1	Sidarta	Hermann Hesse
2	Covek koji se smeje	Victor Hugo
4	1984	George Orwell
5	Biblija	
9	Seobe	Miloš Crnjanski
10	Besnilo	Borislav Pekić
11	Prokleta avlija	Ivo Andrić
15	Stepski vuk	Hermann Hesse
16	Nepodnošljiva lakoća postojanja	
18	Na Drini ćuprija	Ivo Andrić

OTVORI FORMU ZA ODBRANU KNJIGU DODAJ KNJIGU

2. **Запослени** **позива** **систем** да нађе **књиге** по задатој вредности. (АПСО)
3. **Систем** **тражи** **књиге** по задатој вредности. (СО)
4. **Систем** **приказује** **запосленом** податке о **књигама** и поруку: “**Систем** је нашао **књиге** по задатој вредности”. (ИА)

Pretraga knjiga

Naslov: Stepski vuk PRETRAZI

ID	Naslov	Autor
15	Stepski vuk	Hermann Hesse

OTVORI FORMU ZA ODBRANU KNJIGU DODAJ KNJIGU

5. **Запослени** **бира** **књигу**. (АПУСО)

Pretraga knjiga

Naslov: Stepski vuk PRETRAZI

ID	Naslov	Autor
15	Stepski vuk	Hermann Hesse

OTVORI FORMU ZA ODBRANU KNJIGU DODAJ KNJIGU

6. **Запослени** **позива** **систем** да учита **књигу**. (АПСО)
7. **Систем** **учитава** **књигу**. (СО)

8. Систем приказује запосленом податке о књизи и поруку: “Систем је успешно учитао књигу”. (ИА)

Knjiga

Naslov: Stepski vuk

Opis:

Autor: Hermann Hesse

Trenutno na stanju: 0

Broj primeraka knjige koji se dodaje: 0

OBRISI SACUVAJ

9. Запослени уноси (мења) податке о књизи. (АПУСО)

Knjiga

Naslov: Stepski vuk

Opis: maste ali ne u smislu proizvoljnog izmisljanja , vec kao pokusaj da se duboko doživljavani dusevni tokovi prikazu kao vidljiva zbivanja. Tako je Heseovo kazivanje, s jedne strane, vezano za stvarnost koja je formirana u vremenu, a s druge, za nevidljivu i neoformljenu stvarnost koja ne zna za vreme...

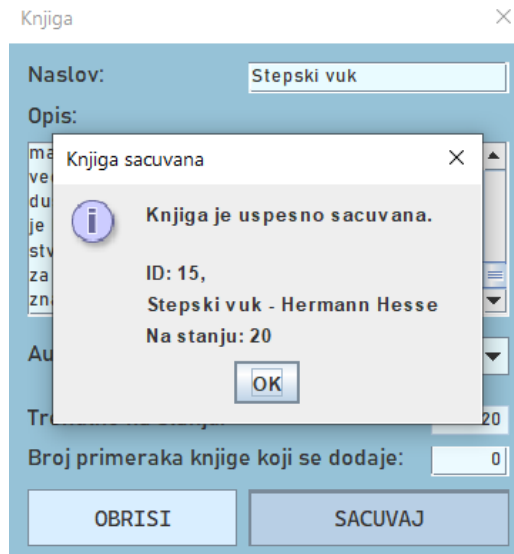
Autor: Hermann Hesse

Trenutno na stanju: 0

Broj primeraka knjige koji se dodaje: 20

OBRISI SACUVAJ

10. Запослени контролише да ли је коректно унео податке о књизи. (АНСО)
11. Запослени позива систем да измени податке о књизи. (АПСО)
12. Систем памти податке о књизи. (СО)
13. Систем приказује запосленом измењену књигу и поруку: “Систем је изменио књигу.” (ИА)

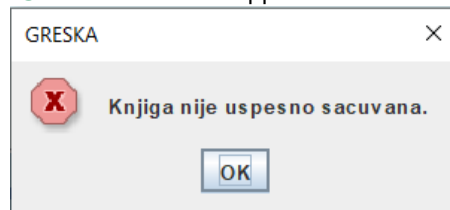


### Алтернативна сценарија

4.1 Уколико **систем** не може да нађе **књиге** он приказује **запосленом** поруку: “**Систем** не може да нађе ниједну **књигу** по задатој вредности”. Прекида се извршење сценарија. (ИА)



13.1 Уколико **систем** не може да измени податке о **књизи** он приказује **запосленом** поруку “**Систем** не може да измени **књигу**”. (ИА)





## СК4: Случај коришћења – Брисање књиге – пројектовање екранске форме

### Назив СК

Брисање књиге

### Актери СК

Запослени

### Учесници СК

Запослени и систем (програм)

**Предуслов:** Систем је укључен и запослени је пријављен под својом шифром. Систем приказује форму за рад са књигом.

Pretraga knjiga

Naslov:  PRETRAZI

ID	Naslov	Autor
1	Sidarta	Hermann Hesse
2	Covek koji se smeje	Victor Hugo
4	1984	George Orwell
5	Biblija	
9	Seobe	Miloš Crnjanski
10	Besnilo	Borislav Pekić
11	Prokleta avlija	Ivo Andrić
15	Stepski vuk	Hermann Hesse
16	Nepodnošljiva lakoća postojanja	
18	Na Drini ćuprija	Ivo Andrić

OTVORI FORMU ZA ODBRANU KNJIGU DODAJ KNJIGU

### Основни сценарио СК

1. Запослени уноси вредност по којој претражује књиге. (АПУСО)

Pretraga knjiga

Naslov:  PRETRAZI

ID	Naslov	Autor
1	Sidarta	Hermann Hesse
2	Covek koji se smeje	Victor Hugo
4	1984	George Orwell
5	Biblija	
9	Seobe	Miloš Crnjanski
10	Besnilo	Borislav Pekić
11	Prokleta avlija	Ivo Andrić
15	Stepski vuk	Hermann Hesse
16	Nepodnošljiva lakoća postojanja	
18	Na Drini ćuprija	Ivo Andrić

OTVORI FORMU ZA ODBRANU KNJIGU DODAJ KNJIGU

2. Запослени позива систем да нађе књиге по задатој вредности. (АПСО)
3. Систем тражи књиге по задатој вредности. (СО)

4. **Систем приказује запосленом** податке о **књигама** и поруку: “**Систем** је нашао **књиге** по задатој вредности”. (ИА)

Pretraga knjiga

Naslov: Stepski vuk PRETRAZI

ID	Naslov	Autor
15	Stepski vuk	Hermann Hesse

OTVORI FORMU ZA ODBRANU KNJIGU DODAJ KNJIGU

5. **Запослени бира књигу**. (АПУСО)

Pretraga knjiga

Naslov: Stepski vuk PRETRAZI

ID	Naslov	Autor
15	Stepski vuk	Hermann Hesse

OTVORI FORMU ZA ODBRANU KNJIGU DODAJ KNJIGU

6. **Запослени позива систем** да учита **књигу**. (АПСО)
7. **Систем учитава књигу**. (СО)
8. **Систем приказује запосленом књигу** и поруку: “**Систем** је успешно учитао **књигу**” (ИА)

Knjiga

Naslov: Stepski vuk

Opis:  
maste ali ne u smislu proizvoljnog izmisljanja ,  
vec kao pokusaj da se duboko doživljavani  
dusevni tokovi prikazu kao vidljiva zbivanja. Tako  
je Heseovo kazivanje, s jedne strane, vezano za  
stvarnost koja je formirana u vremenu, a s druge,  
za nevidljivu i neoformljenu stvarnost koja ne  
zna za vreme...

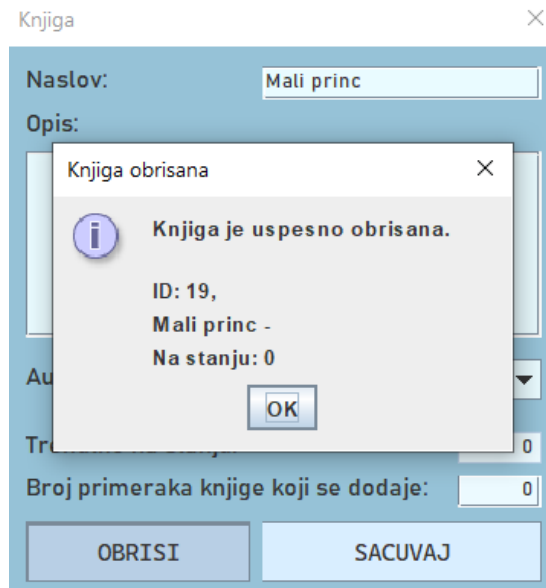
Autor: Hermann Hesse

Trenutno na stanju: 20

Broj primeraka knjige koji se dodaje: 0

OBRISI SACUVAJ

9. **Запослени** **позива** **систем** да обрише **књигу**. (АПСО)
10. **Систем** **брише** **књигу**. (СО)
11. **Систем** **приказује** **запосленом** поруку: “**Систем** је обрисао **књигу**.” (ИА)

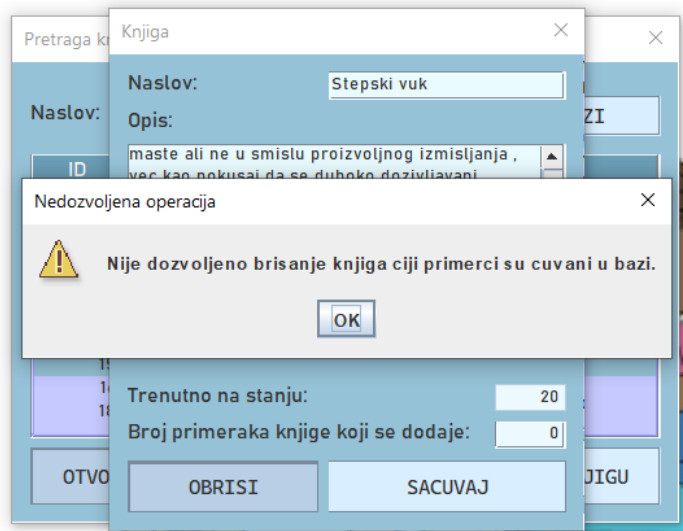


## Алтернативна сценарија

4.1 Уколико **систем** не може да нађе **књиге** он приказује **запосленом** поруку:  
“**Систем** не може да нађе ниједну **књигу** по задатој вредности”. Прекида се извршење сценарија. (ИА)



11.1 Уколико **систем** не може да обрише **књигу** он приказује **запосленом** поруку:  
“**Систем** не може да обрише **књигу**”. (ИА)



## СК5: Случај коришћења – Креирање члана – пројектовање екранске форме

### Назив СК

Креирање члана

### Актори СК

Запослени

### Учесници СК

Запослени и систем (програм)

**Предуслов:** Систем је укључен и **запослени** је пријављен под својом шифром. Систем приказује форму за рад са **чланом**.

Clan X

Ime:

Prezime:

Datum rođenja:

Dan:

Mesec:

Godina:

E-mail:

SACUVAJ

### Основни сценарио СК

1. **Запослени** уноси податке за креирање члана. (АПУСО)

Clan X

Ime:

Prezime:

Datum rođenja:

Dan:

Mesec:

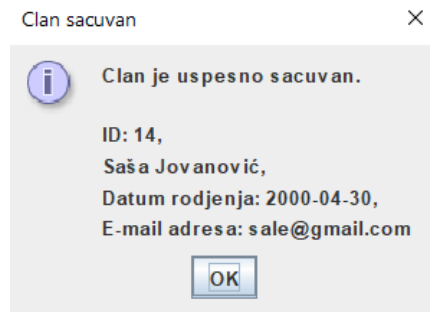
Godina:

E-mail:

OBRISI SACUVAJ

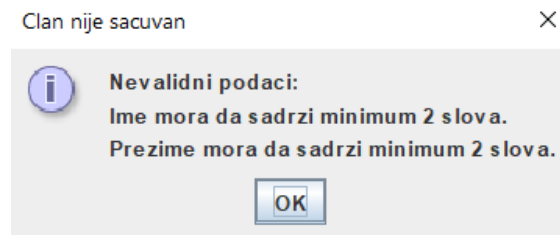
2. **Запослени** контролише претходно унете податке за креирање члана. (АНСО)
3. **Запослени** позива **систем** да запамти податке о члану. (АПСО)
4. **Систем** памти податке о члану. (СО)

5. Систем приказује запосленом запамћеног члана и поруку: “Систем је запамтио члана”. (ИА)



#### Алтернативна сценарија

- 5.1 Уколико систем не може да запамти податке о члану он приказује запосленом поруку “Систем не може да запамти члана”. (ИА)



## СК6: Случај коришћења – Претраживање члана – пројектовање екранске форме

### Назив СК

Претраживање члана

### Актори СК

Запослени

### Учесници СК

Запослени и систем (програм)

**Предуслов:** Систем је укључен и **запослени** је пријављен под својом шифром. Систем приказује форму за рад са **чланом**.

Clanovi biblioteke

Ime:

Prezime:

PRETRAZI

ID	Ime	Prezime	Datum rođenja	E-mail
1	Đorđije	Radovic	2001-02-07	djordjijeradovic24@gmail.com
2	Milica	Pantić	2000-08-10	milicapantic@gmail.com
3	Stefan	Radović	1997-09-25	stefanlalov@gmail.com
5	Milos	Klebecko	2001-01-14	klebec@gmail.com
8	Mario	Muscatello	2000-09-18	dox@gmail.com
9	Uros	Radovic	1967-03-25	lalos@gmail.com
13	Marko	Antić	2000-06-07	
14	Saša	Jovanović	2000-04-30	sale@gmail.com

OTVORI FORMU ZA ODBRANOG CLANA

DODAJ CLANA

### Основни сценарио СК

1. **Запослени** уноси вредност по којој претражује **чланове**. (АПУСО)

Clanovi biblioteke

Ime:

Prezime:

PRETRAZI

ID	Ime	Prezime	Datum rođenja	E-mail
1	Đorđije	Radovic	2001-02-07	djordjijeradovic24@gmail.com
2	Milica	Pantić	2000-08-10	milicapantic@gmail.com
3	Stefan	Radović	1997-09-25	stefanlalov@gmail.com
5	Milos	Klebecko	2001-01-14	klebec@gmail.com
8	Mario	Muscatello	2000-09-18	dox@gmail.com
9	Uros	Radovic	1967-03-25	lalos@gmail.com
13	Marko	Antić	2000-06-07	
14	Saša	Jovanović	2000-04-30	sale@gmail.com

OTVORI FORMU ZA ODBRANOG CLANA

DODAJ CLANA

2. **Запослени** **позива** **систем** да нађе **чланове** по задатој вредности. (АПСО)
3. **Систем** **тражи** **чланове** по задатој вредности. (СО)
4. **Систем** **приказује** **запосленом** податке о **члановима** и поруку: “**Систем** је нашао **чланове** по задатој вредности”. (ИА)

Clanovi biblioteke

Ime:  PRETRAZI

Prezime:

ID	Ime	Prezime	Datum rođenja	E-mail
2	Milica	Pantic	2000-08-10	milicapantic@gmail.com

OTVORI FORMU ZA ODBRANOG CLANA DODAJ CLANA

5. **Запослени** **бира** **члана**. (АПУСО)

Clanovi biblioteke

Ime:  PRETRAZI

Prezime:

ID	Ime	Prezime	Datum rođenja	E-mail
2	Milica	Pantic	2000-08-10	milicapantic@gmail.com

OTVORI FORMU ZA ODBRANOG CLANA DODAJ CLANA

6. **Запослени** **позива** **систем** да учита **члана**. (АПСО)
7. **Систем** **учитава** **члана**. (СО)
8. **Систем** **приказује** **запосленом** податке о **члану** и поруку: “**Систем** је успешно учитао **члана**.”. (ИА)

Clan

Ime:

Prezime:

Datum rođenja:

Dan:

Mesec:

Godina:

E-mail:

OBRISI SACUVAJ



## Алтернативна сценарија

4.1 Уколико **систем** не може да нађе **чланове** он приказује **запосленом** поруку:  
“**Систем** не може да нађе ниједног **члана** по задатој вредности”. Прекида се извршење сценарија. (ИА)

Clanovi biblioteke ×


Ime:

Prezime:

ID	Ime	Prezime	Datum rođenja	E-mail
----	-----	---------	---------------	--------

8.1 Уколико **систем** не може да учита **члана** он приказује **запосленом** поруку:  
“**Систем** не може да учита **члана**”. (ИА)

Odaberite clana ×

 Nije odabran nijedan clan.

## СК7: Случај коришћења – Промена података о члану – пројектовање екранске форме

### Назив СК

Промена података о члану

### Актори СК

Запослени

### Учесници СК

Запослени и систем (програм)

**Предуслов:** Систем је укључен и запослени је пријављен под својом шифром. Систем приказује форму за рад са чланом.

Clanovi biblioteke

Ime:

Prezime:

PRETRAZI

ID	Ime	Prezime	Datum rođenja	E-mail
1	Đorđije	Radović	2001-02-07	djordjijeradovic24@gmail.com
2	Milica	Pantić	2000-08-10	milicapantic@gmail.com
3	Stefan	Radović	1997-09-25	stefanalov@gmail.com
5	Milos	Klebecko	2001-01-14	klebec@gmail.com
8	Mario	Muscattello	2000-09-18	dox@gmail.com
9	Uros	Radovic	1967-03-25	lalos@gmail.com
13	Marko	Antić	2000-06-07	
14	Saša	Jovanović	2000-04-30	sale@gmail.com

OTVORI FORMU ZA ODBRANOГ CLANA DODAJ CLANA

### Основни сценарио СК

1. Запослени уноси вредност по којој претражује чланове. (АПУСО)

Clanovi biblioteke

Ime:

Prezime:

PRETRAZI

ID	Ime	Prezime	Datum rođenja	E-mail
1	Đorđije	Radović	2001-02-07	djordjijeradovic24@gmail.com
2	Milica	Pantić	2000-08-10	milicapantic@gmail.com
3	Stefan	Radović	1997-09-25	stefanalov@gmail.com
5	Milos	Klebecko	2001-01-14	klebec@gmail.com
8	Mario	Muscattello	2000-09-18	dox@gmail.com
9	Uros	Radovic	1967-03-25	lalos@gmail.com
13	Marko	Antić	2000-06-07	
14	Saša	Jovanović	2000-04-30	sale@gmail.com

OTVORI FORMU ZA ODBRANOГ CLANA DODAJ CLANA

2. Запослени позива систем да нађе чланове по задатој вредности. (АПСО)
3. Систем тражи чланове по задатој вредности. (СО)

4. Систем приказује запосленом податке о члановима и поруку: “Систем је нашао чланове по задатој вредности”. (ИА)

Clanovi biblioteke

Ime:  PRETRAZI

Prezime:

ID	Ime	Prezime	Datum rođenja	E-mail
2	Milica	Pantic	2000-08-10	milicapantic@gmail.com

OTVORI FORMU ZA ODBRANOG CLANA DODAJ CLANA

5. Запослени бира члана. (АПУСО)

Clanovi biblioteke

Ime:  PRETRAZI

Prezime:

ID	Ime	Prezime	Datum rođenja	E-mail
2	Milica	Pantic	2000-08-10	milicapantic@gmail.com

OTVORI FORMU ZA ODBRANOG CLANA DODAJ CLANA

6. Запослени позива систем да учита члана. (АПСО)
7. Систем учитава члана. (СО)
8. Систем приказује запосленом податке о члану и поруку: “Систем је успешно учитао члана.”. (ИА)

Clan

Ime:

Prezime:

Datum rođenja:

Dan:

Mesec:

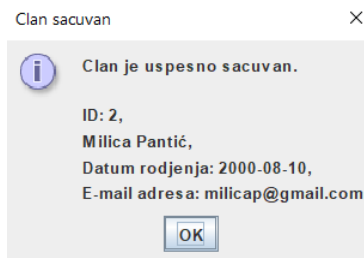
Godina:

E-mail:

OBRISI SACUVAJ

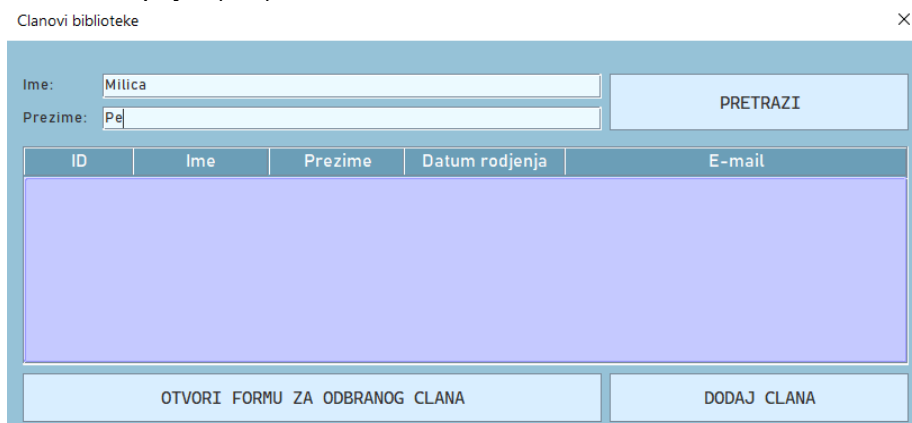
9. Запослени уноси (мења) податке о члану. (АПУСО)
10. Запослени контролише да ли је коректно унео податке о члану. (АНСО)
11. Запослени позива систем да измени податке о члану. (АПСО)
12. Систем памти податке о члану. (СО)

13. Систем приказује запосленом измењеног члана и поруку: “Систем је изменио члана.” (ИА)

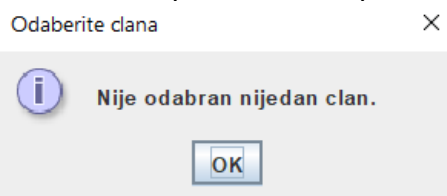


### Алтернативна сценарија

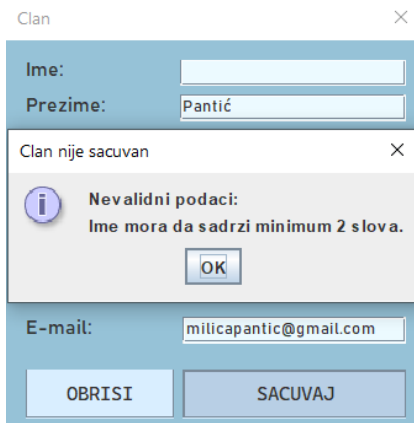
- 4.1 Уколико систем не може да нађе чланове он приказује запосленом поруку: “Систем не може да нађе ниједног члана по задатој вредности”. Прекида се извршење сценарија. (ИА)



- 8.1 Уколико систем не може да учита члана он приказује запосленом поруку: “Систем не може да учита члана”. Прекида се извршење сценарија. (ИА)



- 13.1 Уколико систем не може да измени податке о члану он приказује запосленом поруку “Систем не може да измени члана”. (ИА)



## СК8: Случај коришћења – Брисање члана – пројектовање екранске форме

Назив СК

Брисање члана

Актори СК

Запослени

Учесници СК

Запослени и систем (програм)

**Предуслов:** Систем је укључен и запослени је пријављен под својом шифром. Систем приказује форму за рад са чланом.

Clanovi biblioteke

Ime:  PRETRAZI

Prezime:

ID	Ime	Prezime	Datum rođenja	E-mail
1	Đorđije	Radovic	2001-02-07	djordjijeradovic24@gmail.com
2	Milica	Pantić	2000-08-10	milicapantic@gmail.com
3	Stefan	Radović	1997-09-25	stefanlalov@gmail.com
5	Milos	Klebecko	2001-01-14	klebec@gmail.com
8	Mario	Muscatello	2000-09-18	dox@gmail.com
9	Uros	Radovic	1967-03-25	lalos@gmail.com
13	Marko	Antić	2000-06-07	
14	Saša	Jovanović	2000-04-30	sale@gmail.com

OTVORI FORMU ZA ODBRANOG CLANA DODAJ CLANA

### Основни сценарио СК

1. **Запослени** уноси вредност по којој претражује чланове. (АПУСО)

Clanovi biblioteke

Ime:  PRETRAZI

Prezime:

ID	Ime	Prezime	Datum rođenja	E-mail
1	Đorđije	Radovic	2001-02-07	djordjijeradovic24@gmail.com
2	Milica	Pantić	2000-08-10	milicapantic@gmail.com
3	Stefan	Radović	1997-09-25	stefanlalov@gmail.com
5	Milos	Klebecko	2001-01-14	klebec@gmail.com
8	Mario	Muscatello	2000-09-18	dox@gmail.com
9	Uros	Radovic	1967-03-25	lalos@gmail.com
13	Marko	Antić	2000-06-07	
14	Saša	Jovanović	2000-04-30	sale@gmail.com

OTVORI FORMU ZA ODBRANOG CLANA DODAJ CLANA

2. **Запослени** **позива** **систем** да нађе **чланове** по задатој вредности. (АПСО)
3. **Систем** **тражи** **чланове** по задатој вредности. (СО)
4. **Систем** **приказује** **запосленом** податке о **члановима** и поруку: “**Систем** је нашао **чланове** по задатој вредности”. (ИА)

Clanovi biblioteke

Ime:  PRETRAZI

Prezime:

ID	Ime	Prezime	Datum rođenja	E-mail
2	Milica	Pantic	2000-08-10	milicapantic@gmail.com

OTVORI FORMU ZA ODBRANOG CLANA DODAJ CLANA

5. **Запослени** **бира** **члана**. (АПУСО)

Clanovi biblioteke

Ime:  PRETRAZI

Prezime:

ID	Ime	Prezime	Datum rođenja	E-mail
2	Milica	Pantic	2000-08-10	milicapantic@gmail.com

OTVORI FORMU ZA ODBRANOG CLANA DODAJ CLANA

6. **Запослени** **позива** **систем** да учита **члана**. (АПСО)
7. **Систем** **учитава** **члана**. (СО)
8. **Систем** **приказује** **запосленом** **члана** и поруку: “**Систем** је успешно учитао **члана**” (ИА)

Clan

Ime:

Prezime:

Datum rođenja:

Dan:

Mesec:

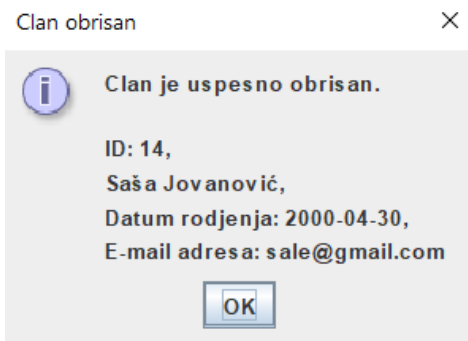
Godina:

E-mail:

OBRISI SACUVAJ

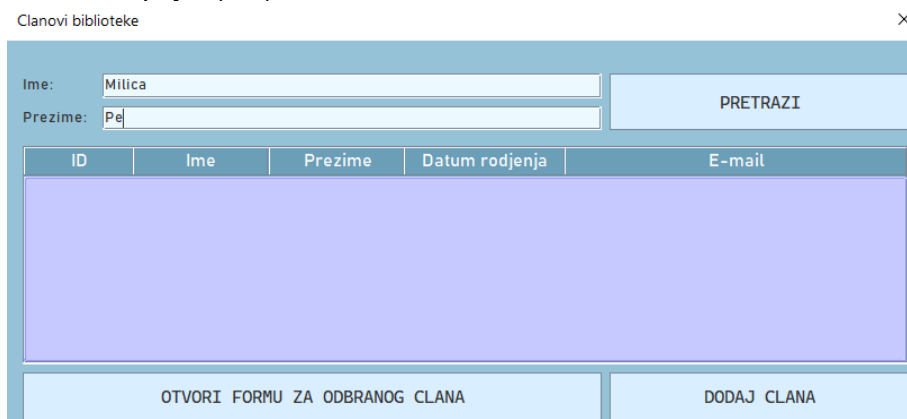
9. **Запослени** **позива** **систем** да обрише **члана**. (АПСО)
10. **Систем** **брише** **члана**. (СО)

11. Систем приказује запосленом поруку: “Систем је обрисао члана.” (ИА)

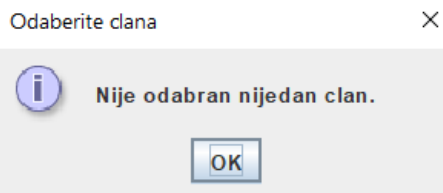


### Алтернативна сценарија

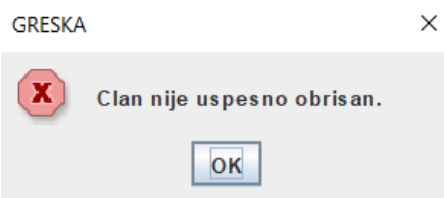
4.1 Уколико систем не може да нађе чланове он приказује запосленом поруку: “Систем не може да нађе ниједног члана по задатој вредности”. Прекида се извршење сценарија. (ИА)



8.1 Уколико систем не може да учита члана он приказује запосленом поруку: “Систем не може да учита члана”. Прекида се извршење сценарија. (ИА)



11.1 Уколико систем не може да обрише члана он приказује запосленом поруку “Систем не може да обрише члана”. (ИА)



## СК9: Случај коришћења – Креирање потврде о изнајмљивању књига (сложен СК) – пројектовање екранске форме

### Назив СК

Креирање потврде о изнајмљивању књига

### Актори СК

Запослени

### Учесници СК

Запослени и систем (програм)

**Предуслов:** Систем је укључен и **запослени** је пријављен под својом шифром. Систем приказује форму за рад са **потврдом о изнајмљивању књига**. Учитане су листа свих чланова и листа свих књига у **систему**.

The screenshot shows a web form titled 'Iznajmljivanje knjiga'. It has several input fields and buttons. On the left, there are fields for 'Knjiga' (Title, ID primerka), 'Clan biblioteke' (Name, Surname), and buttons 'PRETRAZI KNJIGU' and 'PRETRAZI CLANA'. In the center, there is a table with columns 'ID', 'Naslov', and 'Autor'. On the right, there is a dropdown menu 'ODABERI CLANA BIBLIOTEKE', a large empty text area, and buttons 'VRATI ODABRANI PRIMERAK' and 'POTVRDI'.

### Основни сценарио СК

1. **Запослени** уноси податке у **потврду о изнајмљивању књига**. (АПУСО)

The screenshot shows the same form as before, but with data entered. The 'Naslov' field contains '19'. The 'Clan biblioteke' fields contain 'Marko'. The table now has data rows. The dropdown menu is set to 'ID: 13, Marko Antić'. The large text area contains the text: 'ID: 13, Marko Antić', 'ID: 168, Sidarta - Hermann Hesse', and 'ID: 238, 1984 - George Orwell'.

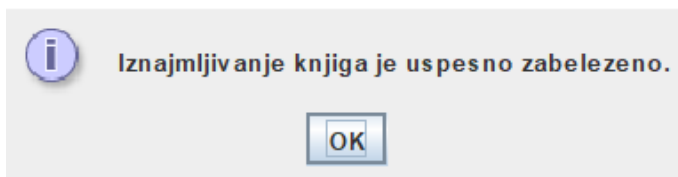
ID	Naslov	Autor
235 1984		George Orwell
236 1984		George Orwell
237 1984		George Orwell
238 1984		George Orwell
239 1984		George Orwell
240 1984		George Orwell
241 1984		George Orwell
242 1984		George Orwell
243 1984		George Orwell
244 1984		George Orwell

2. **Запослени** контролише да ли је коректно унео податке у **потврду о изнајмљивању књига**. (АНСО)
3. **Запослени** позива **систем** да запамти податке о **потврди о изнајмљивању књига**. (АПСО)
4. **Систем** памти податке о **потврди о изнајмљивању књига**. (СО)



5. Систем приказује запосленом запамћену потврду о изнајмљивању књига и поруку: “Систем је запамтио потврду о изнајмљивању књига”. (ИА)

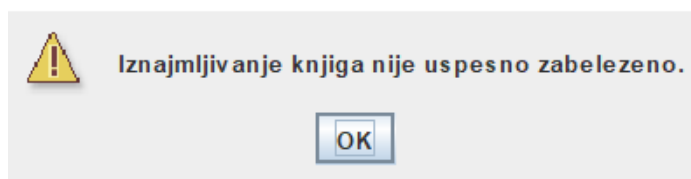
Knjige su iznajmljene



#### Алтернативна сценарија

- 5.1 Уколико систем не може да запамти податке о потврди о изнајмљивању књига он приказује запосленом поруку “Систем не може да запамти потврду о изнајмљивању књига”. (ИА)

GRESKA



СК10: Случај коришћења – Промена потврде о изнајмљивању књига (сложен СК) – пројектовање екранске форме

### Назив СК

Промена **потврде о изнајмљивању књига**

### Актори СК

**Запослени**

### Учесници СК

**Запослени** и **систем** (програм)

**Предуслов:** **Систем** је укључен и **запослени** је пријављен под својом шифром. Систем приказује форму за рад са **потврдом о изнајмљивању књига**. Учитане су листа свих чланова и листа свих књига у **систему**.

Vracanje iznajmljenih knjiga

Clan biblioteke

Ime:

Prezime:

PRETRAZI CLANA

ODABERI CLANA BIBLIOTEKE

ID	Naslov	ID primerka	Datum iznajmljivanja
----	--------	-------------	----------------------

DODAJ ODABRANI PRIMERAK

Iznajmljeni primerci koje clan biblioteke vraca:

VRATI ODABRANI PRIMERAK

POTVRDI

### Основни сценарио СК

1. **Запослени** уноси вредност по којој претражује **изнајмљивања књига**. (АПУСО)

Vracanje iznajmljenih knjiga

Clan biblioteke

Ime:

Prezime:

PRETRAZI CLANA

ODABERI CLANA BIBLIOTEKE

ID: 2, Milica Pantić

ID	Naslov	ID primerka	Datum iznajmljivanja
----	--------	-------------	----------------------

DODAJ ODABRANI PRIMERAK

Iznajmljeni primerci koje clan biblioteke vraca:

VRATI ODABRANI PRIMERAK

POTVRDI

2. **Запослени** позива **систем** да нађе **изнајмљивања књига** по задатој вредности. (АПСО)

3. **Систем** тражи **изнајмљивања књига** по задатој вредности. (СО)

4. **Систем** приказује **запосленом** **изнајмљивања књига** и поруку: “**Систем** је нашао **изнајмљивања књига** по задатој вредности”. (ИА)

Vracanje iznajmljenih knjiga

Clan biblioteke ID: 2, Milica Pantić

Ime: Milica  
Prezime:

PRETRAZI CLANA

ODABERI CLANA BIBLIOTEKE  
ID: 2, Milica Pantić

ID	Naslov	ID primerka	Datum iznajmljivanja
9	Covek koji se smeje	25	2022-12-27
27	Biblija	270	2023-01-01

DODAJ ODABRANI PRIMERAK

Iznajmljeni primerci koje clan biblioteke vraca:

VRATI ODABRANI PRIMERAK

POTVRDI

##### 5. **Запослени** бира **изнајмљивања** књига. (АПУСО)

Vracanje iznajmljenih knjiga

Clan biblioteke ID: 2, Milica Pantić

Ime: Milica  
Prezime:

PRETRAZI CLANA

ODABERI CLANA BIBLIOTEKE  
ID: 2, Milica Pantić

ID	Naslov	ID primerka	Datum iznajmljivanja
9	Covek koji se smeje	25	2022-12-27
27	Biblija	270	2023-01-01

DODAJ ODABRANI PRIMERAK

Iznajmljeni primerci koje clan biblioteke vraca:

VRATI ODABRANI PRIMERAK

POTVRDI

##### 6. **Запослени** **позива систем** да учита **изнајмљивања** књига. (АПСО)

##### 7. **Систем** **учитава** **изнајмљивања** књига. (СО)

##### 8. **Систем приказује** **запосленом** податке о **изнајмљивањима** књига и поруку: “Систем је успешно учитао **изнајмљивања** књига”. (ИА)

Vracanje iznajmljenih knjiga

Clan biblioteke ID: 2, Milica Pantić

Ime: Milica  
Prezime:

PRETRAZI CLANA

ODABERI CLANA BIBLIOTEKE  
ID: 2, Milica Pantić

ID	Naslov	ID primerka	Datum iznajmljivanja
9	Covek koji se smeje	25	2022-12-27
27	Biblija	270	2023-01-01

DODAJ ODABRANI PRIMERAK

Iznajmljeni primerci koje clan biblioteke vraca:

ID: 9; 2022-12-27; 25 - Covek koji se smeje; Milica P  
ID: 27; 2023-01-01; 270 - Biblija; Milica Pantić

VRATI ODABRANI PRIMERAK

POTVRDI

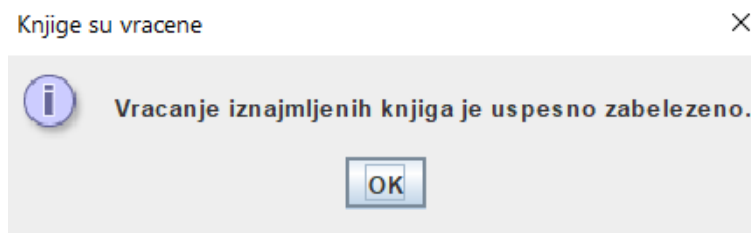
##### 9. **Запослени** **уноси (мења)** податке о **изнајмљивањима** књига. (АПУСО)

##### 10. **Запослени** **контролише** да ли је коректно унео податке о **изнајмљивањима** књига. (АНСО)

##### 11. **Запослени** **позива систем** да измени податке о **изнајмљивањима** књига. (АПСО)

##### 12. **Систем** **памти** податке о **изнајмљивањима** књига. (СО)

13. Систем приказује запосленом измењена изнајмљивањима књига и поруку: “Систем је изменио изнајмљивањима књига.” (ИА)

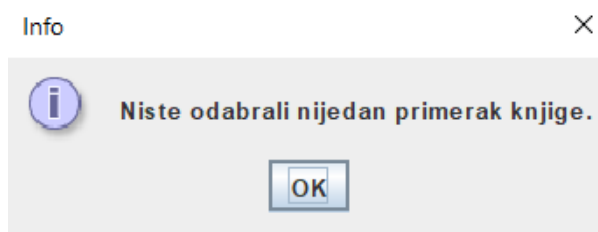


### Алтернативна сценарија

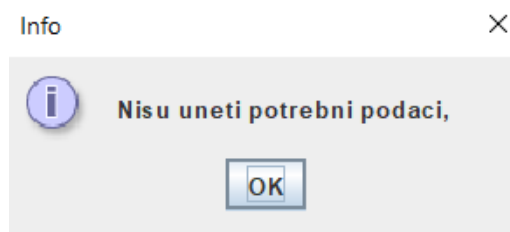
- 4.1 Уколико систем не може да нађе изнајмљивања књига он приказује запосленом поруку: “Систем не може да нађе ниједно изнајмљивање књига по задатој вредности”. Прекида се извршење сценарија. (ИА)



- 8.1 Уколико систем не може да учита изнајмљивања књига он приказује запосленом поруку: “Систем не може да учита изнајмљивања књига”. Прекида се извршење сценарија. (ИА)



- 13.1 Уколико систем не може да измени податке о изнајмљивањима књига он приказује запосленом поруку “Систем не може да измени изнајмљивања књига”. (ИА)



### 3.2.2 Пројектовање контролера корисничког интерфејса

Контролер корисничког интерфејса је одговоран за:

- Прихватање графичких објеката од екранске форме;
- Конвертовање података који се налазе у графичким објектима у доменске објекте који ће бити прослеђени преко мреже до апликационог сервера;
- Конвертовање доменских објеката у графичке објекте и прослеђује их до екранске форме

### 3.3 Пројектовање апликационе логике

Апликациони сервери су одговорни да обезбеде сервисе који ће да омогуће реализацију апликационе логике софтверског система. Пројектовани апликациони сервер садржи:

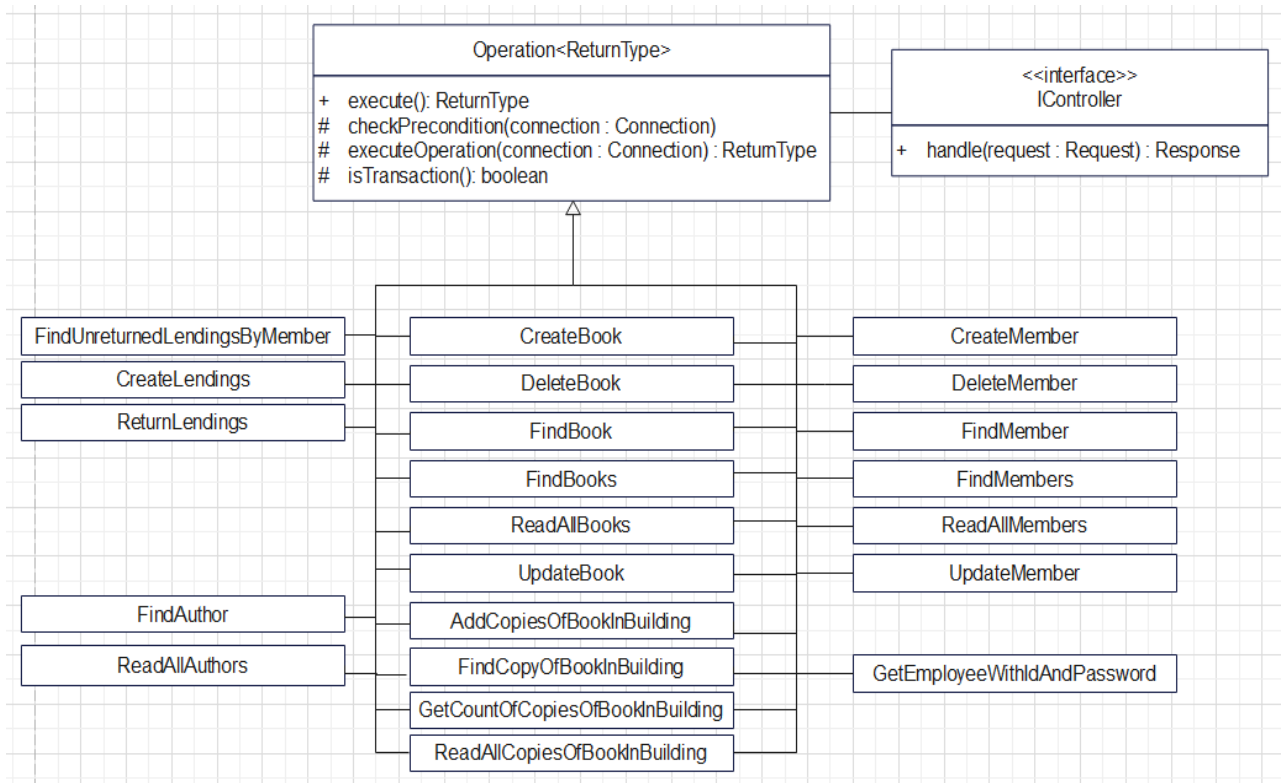
- Део за комуникацију са клијентима,
- Контролер апликационе логике,
- Део који садржи пословну логику,
- Део за комуникацију са складиштем података (брокер базе података).

Део за комуникацију подиже серверски сокет који ослушкује мрежу. Када клијентски сокет успостави конекцију са серверским сокетом, тада сервер генерише нит која ће успоставити двосмерну комуникацију са клијентом.

Слање и примање података од клијента се обавља разменом објеката класе Request i Response и остварује се преко сокета. Клијент шаље захтев за извршење неке од системских операција до одговарајуће нити која је повезана са тим клијентом. Та нит прихвата захтев и прослеђује га до контролера апликационе логике. Након извршења системске операције, резултат се преко контролера апликационе логике враћа до нити клијента која тај резултат шаље назад до клијента.

### 3.3.1 Контролер апликационе логике

Контролер апликационе логике прихвата захтев за извршење системске операције од нити клијента и даље га преумерава до класа које су одговорне за извршење системских операција. Након извршења системске операције контролер апликационе логике прихвата резултат и прослеђује га позиваоцу (нити клијента).



### 3.3.2 Пословна логика

Пројектовање понашања софтверског система – системске операције

За сваку системску операцију треба направити концептуална решења која су директно повезана са логиком проблема. За сваки уговор пројектује се концептуално решење.

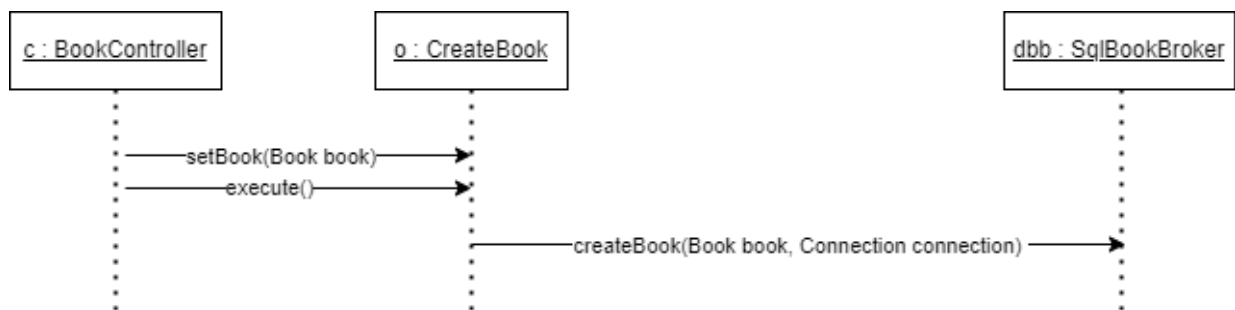
- **Уговор УГ1: ZapamtiKnjigu(Knjiga) Signal**

Операција: CreateBook(Book, Connection) Signal

Веза са СК: СК1

Предуслови: Вредносна и структурна ограничења над објектом **Book** морају бити задовољена.

Постуслови: Подаци о књизи су запамћени.



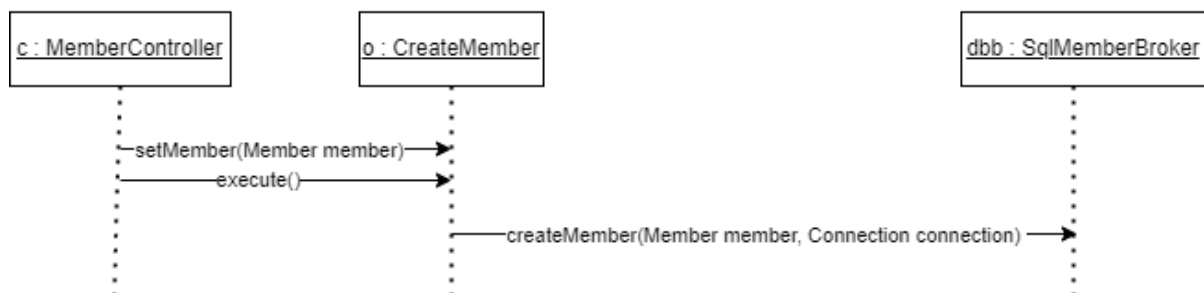
- **Уговор УГ2: ZapamtiClana(Clan) Signal**

Операција: CreateMember(Member, Connection) Signal

Веза са СК: СК5

Предуслови: Вредносна и структурна ограничења над објектом **Clan** морају бити задовољена.

Постуслови: Подаци о члану су запамћени.



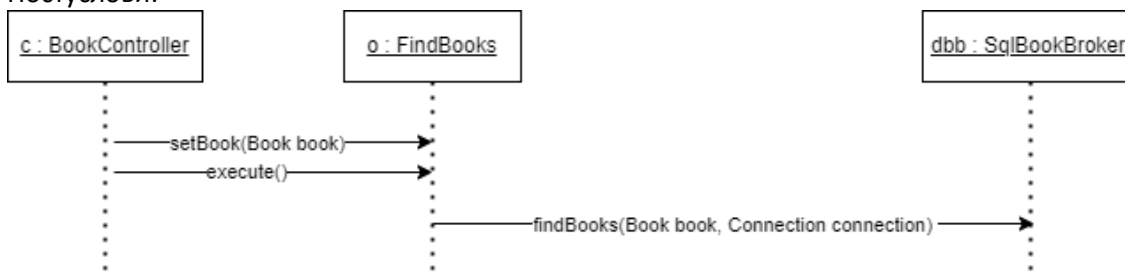
- **Уговор УГ3: NadjiKnjige(Knjiga, List<Knjiga>) Signal**

Операција: FindBooks(Book, Connection) Signal

Веза са СК: СК2, СК3, СК4

Предуслови:

Постуслови:



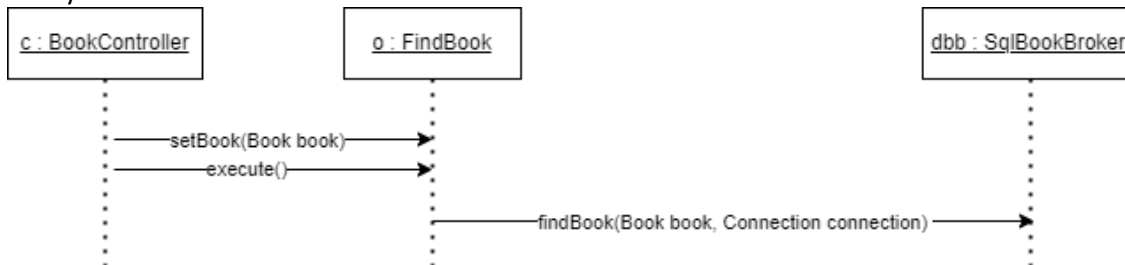
- **Уговор УГ4: UcitajKnjigu(Knjiga) Signal**

Операција: FindBook(Book, Connection) Signal

Веза са СК: СК2, СК3, СК4

Предуслови:

Постуслови:



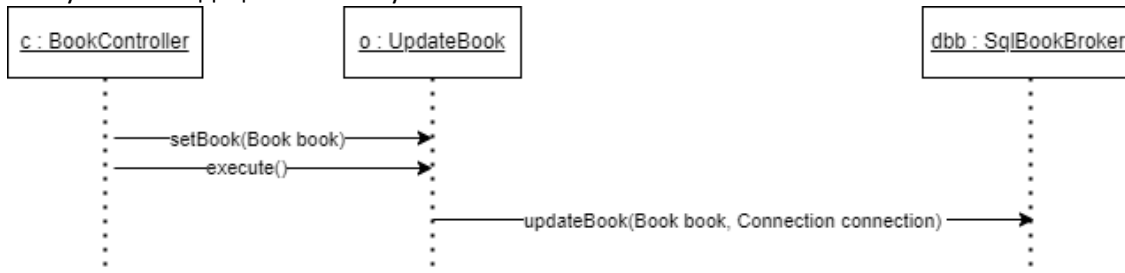
- **Уговор УГ5: IzmeniKnjigu(Knjiga) Signal**

Операција: UpdateBook(Book, Connection) Signal

Веза са СК: СК3

Предуслови: Вредносна и структурна ограничења над објектом **Knjiga** морају бити задовољена.

Постуслови: Подаци о књизи су измењени.





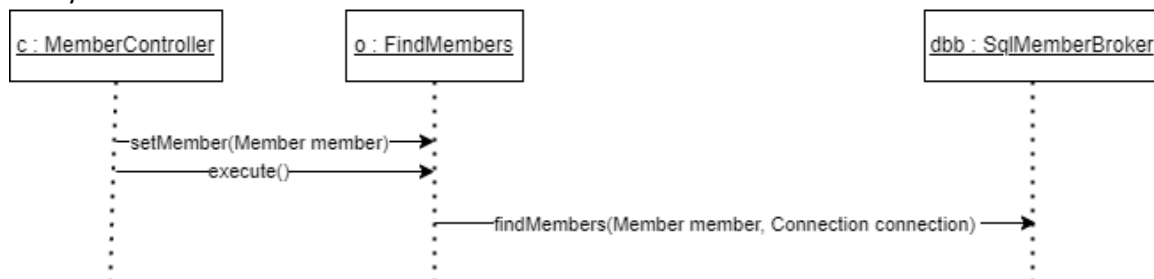
- Уговор УГ6: NadjiClanove(Clan, List<Clan>) Signal

Операција: FindMembers(Member, Connection) Signal

Веза са СК: СК6, СК7, СК8

Предуслови:

Постуслови:



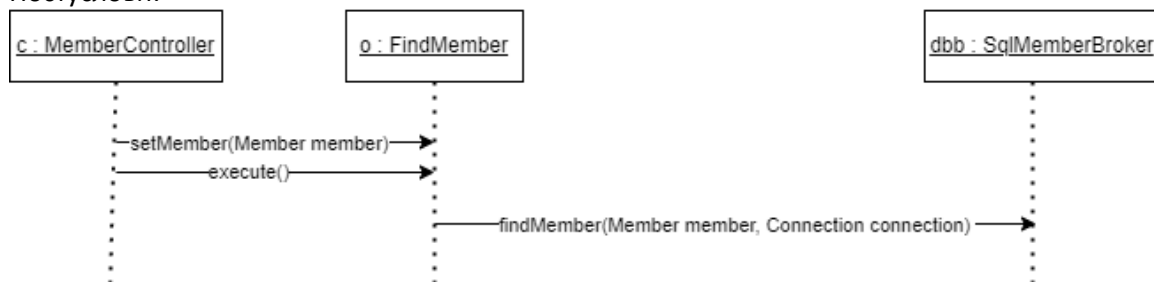
- Уговор УГ7: UcitajClana(Clan) Signal

Операција: FindMember(Member, Connection) Signal

Веза са СК: СК6, СК7, СК8

Предуслови:

Постуслови:



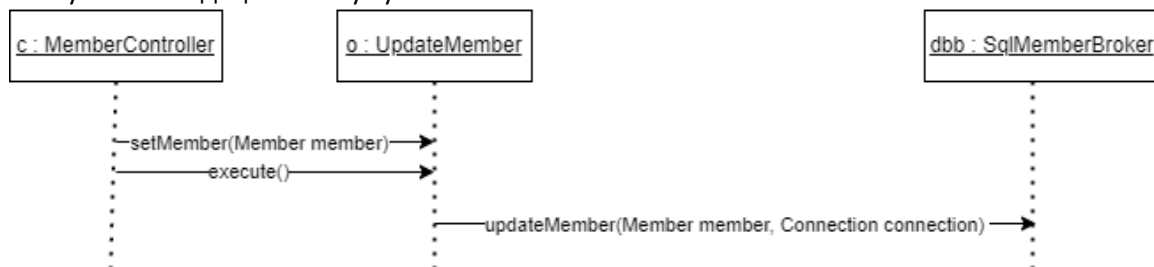
- Уговор УГ8: IzmeniClana(Clan) Signal

Операција: UpdateMember(Member, Connection) Signal

Веза са СК: СК7

Предуслови: Вредносна и структурна ограничења над објектом **Clan** морају бити задовољена.

Постуслови: Подаци о члану су измењени.



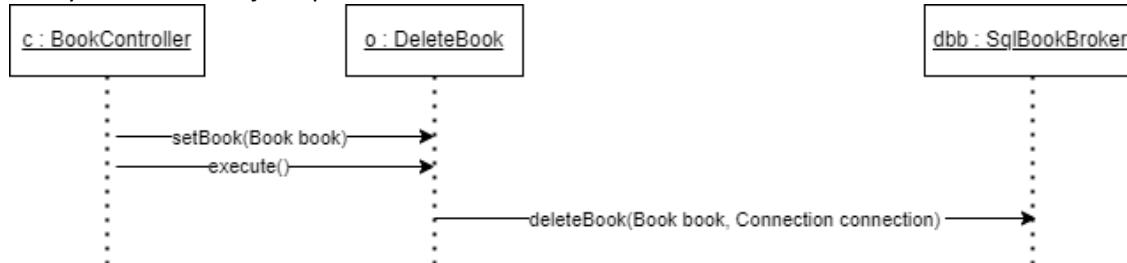
- **Уговор УГ9: ObrisiKnjigu(Knjiga) Signal**

Операција: DeleteBook(Book, Connection) Signal

Веза са СК: СК4

Предуслови: Структурна ограничења над објектом **Knjiga** морају бити задовољена.

Постуслови: Књига је обрисана.



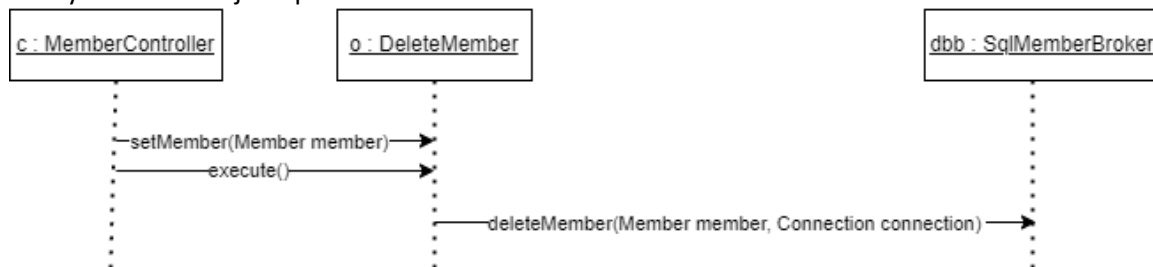
- **Уговор УГ10: ObrisiClana(Clan) Signal**

Операција: DeleteMember(Member, Connection) Signal

Веза са СК: СК8

Предуслови: Структурна ограничења над објектом **Clan** морају бити задовољена.

Постуслови: Члан је обрисан.



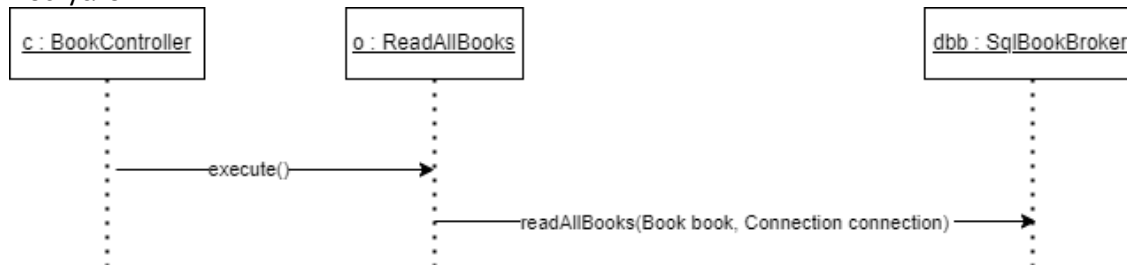
- **Уговор УГ11: UcitajListuKnjiga(List<Knjiga>) Signal**

Операција: ReadAllBooks(Book, Connection) Signal

Веза са СК: СК9, СК10

Предуслови:

Постуслови:



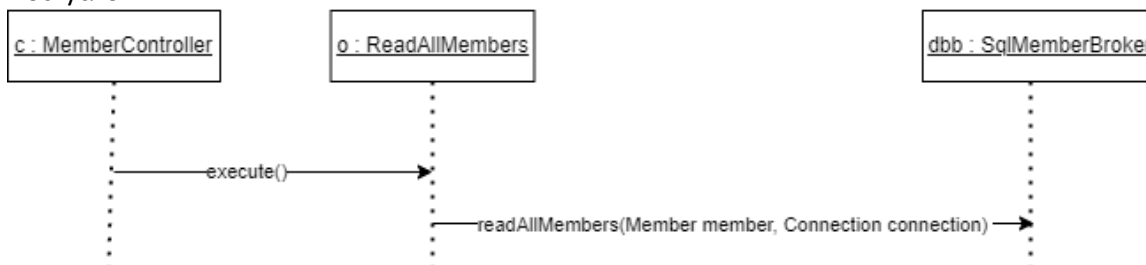
- **Уговор УГ12: UcitajListuClanova(List<Clan>) Signal**

Операција: ReadAllMembers(Member, Connection) Signal

Веза са СК: СК9, СК10

Предуслови:

Постуслови:



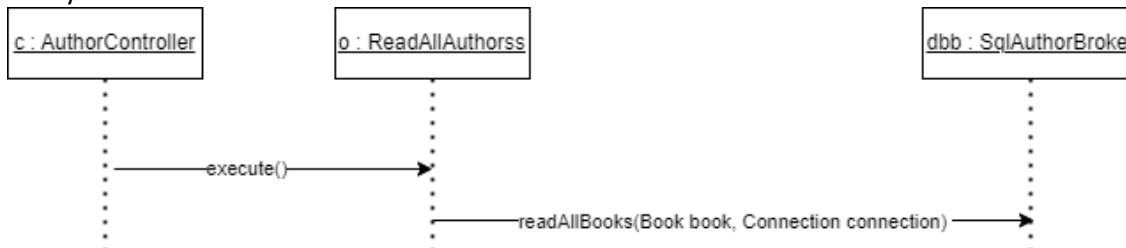
- **Уговор УГ13: UcitajListuAutora(List<Autor>) Signal**

Операција: ReadAllAuthors(Author, Connection) Signal

Веза са СК: СК1, СК3

Предуслови:

Постуслови:



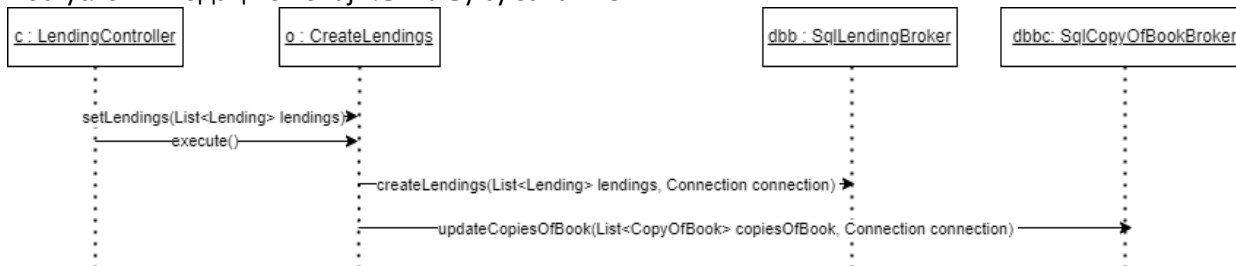
- **Уговор УГ14: Zapamtilznajmljivanja(List<Iznajmljivanje>) Signal**

Операција: CreateLendings(List<Lending>, Connection) Signal

Веза са СК: СК9

Предуслови: Вредносна и структурна ограничења над објектом **Iznajmljivanje** морају бити задовољена.

Постуслови: Подаци о изнајмљивању су запамћени.



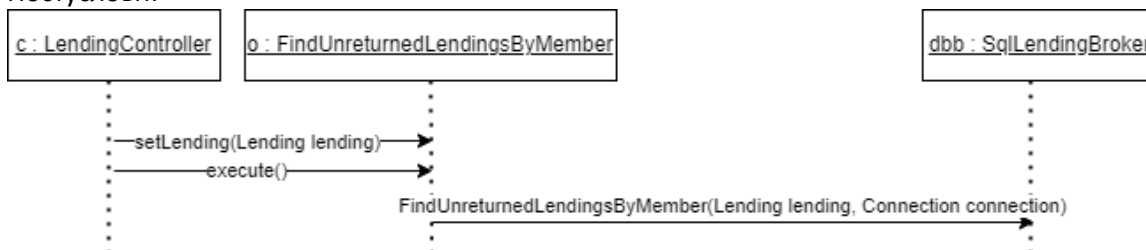
- **Уговор УГ15: Nadjilznajmljivanja(Iznajmljivanje, List<Iznajmljivanje>) Signal**

Операција: FindUnreturnedLendingsByMember(Lending, Connection) Signal

Веза са СК: СК10

Предуслови:

Постуслови:



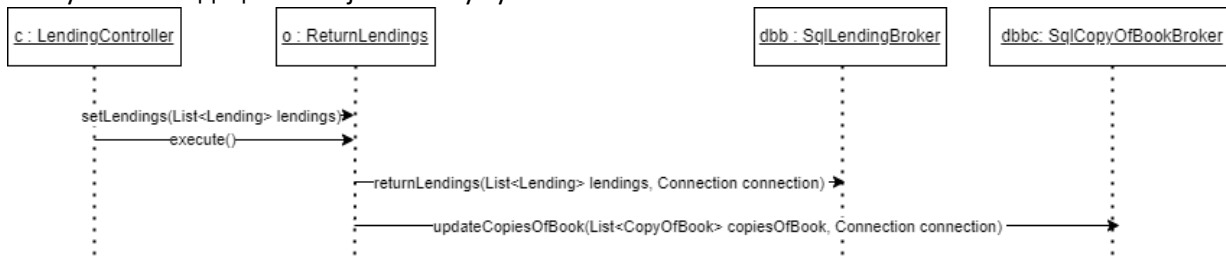
- **Уговор УГ17: Izmenilznajmljivanja(List<Iznajmljivanje>) Signal**

Операција: ReturnLendings(List<Lending>, Connection) Signal

Веза са СК: СК10

Предуслови: Вредносна и структурна ограничења над објектом **Iznajmljivanje** морају бити задовољена.

Постуслови: Подаци о изнајмљивању су измењени.



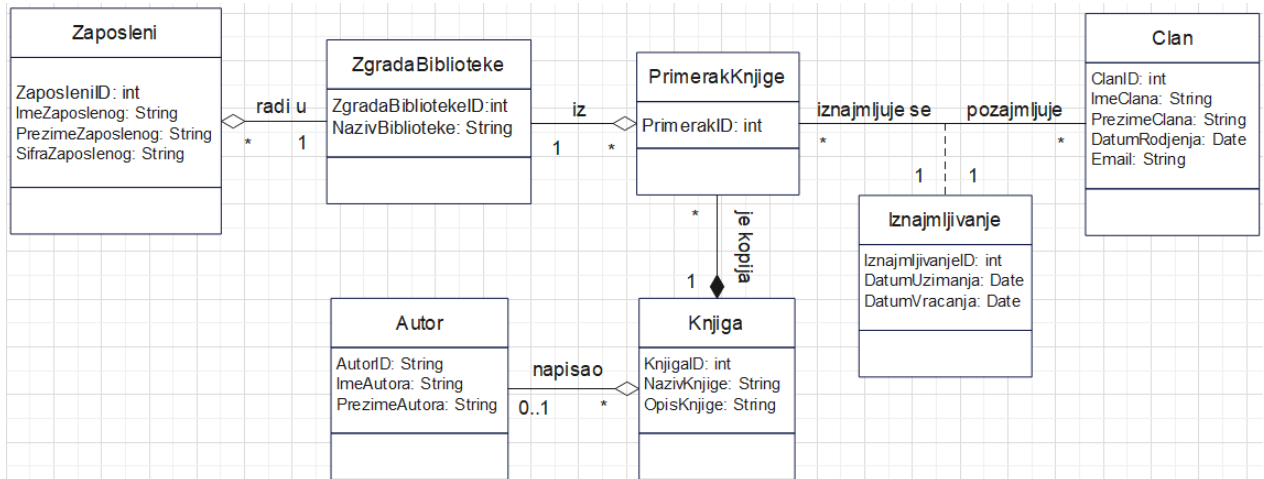
Класе које су одговорне за извршење системских операција наслеђују класу `Operation<ReturnType>`. Та класа представља апстрактну класу чија јавна метода (`execute()`) у себи садржи отварање конекције са базом, валидацију и проверу предуслова, извршење операције, потврду у бази уколико је извршење операције успешно, поништавање уколико извршење операције није било успешно, и затварање конекције. Редослед обављања тих функционалности је генеричан па је уведен `Template Method` патерн како би се одвојили специфичан део кода од генеричног.

### 3.4 Пројектовање структуре софтверског система

На основу концептуалних класа праве се софтверске класе структуре.

Свака доменска класа имплементира интерфејс IEntity да би уз помоћ њега и апстрактне генеричне класе SqlEntity<T extends IEntity> било омогућено коришћење особине полиморфизма и прављење генеричког брокера базе SqlEntityBroker<T extends IEntity> који садржи све основне функционалности заједничке за све доменске класе.

Концептуалне класе:



Софтверске класе структуре:

- Author

```
public class Author implements IEntity, Serializable {
    private Long id;
    private String firstname;
    private String lastname;

    public Author(Long id, String firstname, String lastname) {
        this.id = id;
        this.firstname = firstname;
        this.lastname = lastname;
    }
    public Author() {
    }

    @Override
    public ModelElement getModelElement() {
        return ModelElement.AUTHOR;
    }
    @Override
    public Long getId() {
        return id;
    }
    @Override
    public void setId(Long id) {
        this.id = id;
    }
    public String getFirstname() {
        return firstname;
    }
    public void setFirstname(String firstname) {
        this.firstname = firstname;
    }
    public String getLastname() {
        return lastname;
    }
    public void setLastname(String lastname) {
        this.lastname = lastname;
    }
    @Override
    public String toString() {
        String firstnameToPrint = this.firstname != null ? this.firstname : "";
        String lastnameToPrint = this.lastname != null ? this.lastname : "";
        return firstnameToPrint + " " + lastnameToPrint;
    }
    @Override
    public boolean equals(Object obj) {
        if(obj != null && obj instanceof Author author){
            return Objects.equals(this.id, author.id);
        } else return false;
    }
}
```

- Book

```
public class Book implements IEntity, Serializable {
    private Long id;
    private String title;
    private String description;
    private Author author;

    public Book(){
    }
    public Book(Long id){
        this.id = id;
    }
    public Book(Long id, String title, String description, Author author){
        this(id);
        this.title = title;
        this.description = description;
        this.author = author;
    }
    @Override
    public ModelElement getModelElement() {
        return ModelElement.BOOK;
    }
    @Override
    public void setId(Long id) {
        this.id = id;
    }
    @Override
    public Long getId() {
        return id;
    }
    public String getTitle() {
        return title;
    }
    public void setTitle(String title) {
        this.title = title;
    }
    public String getDescription() {
        return description;
    }
    public void setDescription(String description) {
        this.description = description;
    }
    public Author getAuthor() {
        return author;
    }
    public void setAuthor(Author author) {
        this.author = author;
    }
    public String singlePrint(){
        return "ID: " + id + ",\n" + title + ",\nAuthor: " + (author == null ? "" : author);
    }
    @Override
    public String toString() {
        return title + (author == null ? "" : " - " + author);
    }
}
```

- Building

```
public class Building implements IEntity, Serializable {
    private Long id;
    private String name;

    public Building(Long id, String name) {
        this(id);
        this.name = name;
    }
    public Building(Long id){
        this.id = id;
    }
    public Building() {
    }

    @Override
    public ModelElement getModelElement() {
        return ModelElement.BUILDING;
    }
    @Override
    public void setId(Long id) {
        this.id = id;
    }
    @Override
    public Long getId(){
        return id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }

    @Override
    public String toString() {
        return "ID: " + id + " - " + name;
    }
}
```



- CopyOfBook

```
public class CopyOfBook implements IEntity, Serializable {
    private Long id;
    private Book book;
    private Long buildingId;

    public CopyOfBook(){}
    public CopyOfBook(Long id) {
        this.id = id;
    }
    public CopyOfBook(Long id, Long buildingId){
        this(id);
        this.buildingId = buildingId;
    }
    public CopyOfBook(Long id, Book book){
        this(id);
        this.book = book;
    }
    public CopyOfBook(Book book){
        this.book = book;
    }
    public CopyOfBook(Book book, Long buildingId){
        this.book = book;
        this.buildingId = buildingId;
    }
    public CopyOfBook(Long id, Book book, Long buildingId){
        this(id, buildingId);
        this.book = book;
    }
    @Override
    public ModelElement getModelElement() {
        return ModelElement.COPYOFBOOK;
    }
    @Override
    public void setId(Long id) {
        this.id = id;
    }
}
```

```

@Override
public Long getId(){
    return id;
}
public Book getBook(){
    return book;
}
public void setBook(Book book){
    this.book = book;
}
public Long getBuildingId() {
    return buildingId;
}
public void setBuildingId(Long buildingId) {
    this.buildingId = buildingId;
}
@Override
public String toString() {
    return "ID: " + id + ", " + book;
}

@Override
public boolean equals(Object obj) {
    if(obj instanceof CopyOfBook copyOfBook){
        return Objects.equals(this.id, copyOfBook.getId());
    } else return false;
}

@Override
public int hashCode() {
    int hash = 3;
    hash = 79 * hash + Objects.hashCode(this.id);
    return hash;
}
}

```

- Employee

```
public class Employee implements IEntity, Serializable {
    private Long id;
    private String firstname;
    private String lastname;
    private String password;
    private Building building;

    public Employee(){}
    public Employee(Long id, String password){
        this.id = id;
        this.password = password;
    }
    public Employee(Long id, String firstname, String lastname, String password, Building building) {
        this.id = id;
        this.firstname = firstname;
        this.lastname = lastname;
        this.password = password;
        this.building = building;
    }

    @Override
    public ModelElement getModelElement() {
        return ModelElement.EMPLOYEE;
    }
    @Override
    public void setId(Long id) {
        this.id = id;
    }
    @Override
    public Long getId() {
        return id;
    }
    public String getFirstname(){
        return firstname;
    }
    public String getLastname(){
        return lastname;
    }
    public String getPassword(){
        return password;
    }
    public Building getBuilding(){
        return building;
    }
    @Override
    public String toString() {
        return firstname + " " + lastname;
    }
}
```

- Lending

```
public class Lending implements IEntity, Serializable {
    private Long id;
    private CopyOfBook copyOfBook;
    private Member member;
    private LocalDate lendingDate;
    private LocalDate returnDate;

    public Lending(){}
    public Lending(CopyOfBook copyOfBook, Member member, LocalDate lendingDate){
        this.copyOfBook = copyOfBook;
        this.member = member;
        this.lendingDate = lendingDate;
    }
    public Lending(Long id, CopyOfBook copyOfBook, Member member, LocalDate lendingDate){
        this(copyOfBook, member, lendingDate);
        this.id = id;
    }

    @Override
    public ModelElement getModelElement() {
        return ModelElement.LENDING;
    }
    @Override
    public void setId(Long id) {
        this.id = id;
    }
    @Override
    public Long getId() {
        return id;
    }

    public CopyOfBook getCopyOfBook() {
        return copyOfBook;
    }

    public void setCopyOfBook(CopyOfBook copyOfBook) {
        this.copyOfBook = copyOfBook;
    }

    public Member getMember() {
        return member;
    }

    public void setMember(Member member) {
        this.member = member;
    }

    public LocalDate getLendingDate() {
        return lendingDate;
    }
}
```

```

public void setLendingDate(LocalDate lendingDate) {
    this.lendingDate = lendingDate;
}

public LocalDate getReturnDate() {
    return returnDate;
}

public void setReturnDate(LocalDate returnDate) {
    this.returnDate = returnDate;
}

public String singlePrint(){
    return "ID: " + id + "\nKnjiga: " + copyOfBook + "\nClan biblioteke: " + member + "\nDatum: " + lendingDate;
}

@Override
public String toString() {
    return "ID: " + id + "; " + lendingDate + "; " + copyOfBook.getId() + " - " + copyOfBook.getBook().getTitle() + "; " + member.getFirstname() + " " + member.getLastname();
}

@Override
public boolean equals(Object obj) {
    if(obj instanceof Lending lending){
        return Objects.equals(this.id, lending.getId());
    } else return false;
}

@Override
public int hashCode() {
    int hash = 7;
    hash = 53 * hash + Objects.hashCode(this.id);
    return hash;
}
}

```

- Member

```
public class Member implements IEntity, Serializable {
    private Long id;
    private String firstname;
    private String lastname;
    private LocalDate birthday;
    private String email;

    public Member(Long id, String firstname, String lastname, LocalDate birthday, String email) {
        this.id = id;
        this.firstname = firstname;
        this.lastname = lastname;
        this.birthday = birthday;
        this.email = email;
    }
    public Member(String firstname, String lastname, LocalDate birthday, String email) {
        this.firstname = firstname;
        this.lastname = lastname;
        this.birthday = birthday;
        this.email = email;
    }
    public Member(Long id){
        this.id = id;
    }
    public Member(){}

    @Override
    public ModelElement getModelElement() {
        return ModelElement.MEMBER;
    }
    @Override
    public void setId(Long id) {
        this.id = id;
    }
    @Override
    public Long getId() {
        return id;
    }
    public String getFirstname() {
        return firstname;
    }
    public void setFirstname(String firstname) {
        this.firstname = firstname;
    }
}
```

```

public String getLastname() {
    return lastname;
}
public void setLastname(String lastname) {
    this.lastname = lastname;
}
public LocalDate getBirthday() {
    return birthday;
}
public void setBirthday(LocalDate birthday) {
    this.birthday = birthday;
}
public String getEmail() {
    return email;
}
public void setEmail(String email) {
    this.email = email;
}

public String singlePrint(){
    return "ID: " + id + ",\n" + firstname + " " + lastname + ",\nDatum rodjenja: " + birthday + ",\nE-mail adresa: " + email;
}

@Override
public String toString() {
    return "ID: " + id + ", " + firstname + " " + lastname;
}
}

```

### 3.4.1 Брокер базе података

Апстрактна класа `SqlEntityBroker<T extends IEntity>` представља оквир за све конкретне специфичне брокере доменских класа. Она садржи следеће методе:

```
protected synchronized T create(SqlEntity<T> sqlEntity, Connection connection) throws Exception{
    T entity = sqlEntity.getEntity();
    PreparedStatement preparedStatement = connection.prepareStatement(sqlEntity.getPreparedStatementInsertQuery(), Statement.RETURN_GENERATED_KEYS);
    sqlEntity.setUpPreparedStatementInsert(preparedStatement);
    preparedStatement.executeUpdate();
    ResultSet result = preparedStatement.getGeneratedKeys();
    if(result.next())
        entity.setId(result.getLong(1));
    result.close();
    preparedStatement.close();
    return entity;
}

protected synchronized List<T> createList(SqlEntity<T> sqlEntity, Connection connection) throws Exception{
    List<T> listOfEntities = sqlEntity.getListOfEntities();
    PreparedStatement preparedStatement = connection.prepareStatement(sqlEntity.getPreparedStatementInsertQuery(), Statement.RETURN_GENERATED_KEYS);
    for (T entity : listOfEntities) {
        sqlEntity.setEntity(entity);
        sqlEntity.setUpPreparedStatementInsert(preparedStatement);
        preparedStatement.executeUpdate();
        ResultSet result = preparedStatement.getGeneratedKeys();
        //check if this works or you have to try with for(int i = 0...)
        if(result.next())
            entity.setId(result.getLong(1));
        result.close();
    }
    preparedStatement.close();
    return listOfEntities;
}

protected synchronized T find(SqlEntity<T> sqlEntity, Connection connection) throws Exception{
    Statement statement = connection.createStatement();
    ResultSet resultSet = statement.executeQuery(sqlEntity.getStatementSelectByIdQuery());
    T instance = null;
    if(resultSet.next()){
        instance = sqlEntity.getEntityFromResultSet(resultSet);
    }
    resultSet.close();
    statement.close();
    return instance;
}
```



```

protected synchronized List<T> findEntities(SqlEntity<T> sqlEntity, Connection connection) throws Exception{
    List<T> entities = new ArrayList<>();
    Statement statement = connection.createStatement();
    ResultSet resultSet = statement.executeQuery(sqlEntity.getStatementSelectWithConditionQuery());
    while(resultSet.next()){
        entities.add(sqlEntity.getEntityFromResultSet(resultSet));
    }
    resultSet.close();
    statement.close();
    return entities;
}

protected synchronized List<T> findEntitiesWithCondition(SqlEntity<T> sqlEntity, Connection connection, List<String> conditions) throws Exception{
    List<T> entities = new ArrayList<>();
    Statement statement = connection.createStatement();
    ResultSet resultSet = statement.executeQuery(sqlEntity.constructSelectWithConditionsQuery(conditions));
    while(resultSet.next()){
        entities.add(sqlEntity.getEntityFromResultSet(resultSet));
    }
    resultSet.close();
    statement.close();
    return entities;
}

protected synchronized List<T> readAll(SqlEntity<T> sqlEntity, Connection connection) throws Exception{
    List<T> entities = new ArrayList<>();
    Statement statement = connection.createStatement();
    ResultSet resultSet = statement.executeQuery(sqlEntity.getStatementSelectAllQuery());
    while(resultSet.next()){
        entities.add(sqlEntity.getEntityFromResultSet(resultSet));
    }
    resultSet.close();
    statement.close();
    return entities;
}

protected synchronized T update(SqlEntity<T> sqlEntity, Connection connection) throws Exception{
    PreparedStatement preparedStatement = connection.prepareStatement(sqlEntity.getPreparedStatementUpdateQuery());
    sqlEntity.setUpPreparedStatementUpdate(preparedStatement);
    preparedStatement.executeUpdate();
    preparedStatement.close();
    return sqlEntity.getEntity();
}

```

```

protected synchronized List<T> updateList(SqlEntity<T> sqlEntity, Connection connection) throws Exception{
    List<T> listOfEntities = sqlEntity.getListOfEntities();
    PreparedStatement preparedStatement = connection.prepareStatement(sqlEntity.getPreparedStatementUpdateQuery());
    for (T listEntity : listOfEntities) {
        sqlEntity.setEntity(listEntity);
        sqlEntity.setUpPreparedStatementUpdate(preparedStatement);
        preparedStatement.executeUpdate();
    }
    preparedStatement.close();
    return listOfEntities;
}

protected synchronized T delete(SqlEntity<T> sqlEntity, Connection connection) throws Exception{
    PreparedStatement preparedStatement = connection.prepareStatement(sqlEntity.getPreparedStatementDeleteByIdQuery());
    sqlEntity.setUpPreparedStatementDeleteById(preparedStatement);
    preparedStatement.executeUpdate();
    preparedStatement.close();
    return sqlEntity.getEntity();
}

protected synchronized boolean checkIfExists(SqlEntity<T> sqlEntity, Connection connection) throws Exception{
    Statement statement = connection.createStatement();
    ResultSet resultSet = statement.executeQuery("SELECT COUNT(1) FROM " + sqlEntity.getTableNames() + " WHERE ID = " + sqlEntity.getEntity().getId());
    int numberOfEntities = 0;
    if(resultSet.next()){
        numberOfEntities = resultSet.getInt("COUNT(1)");
    }
    resultSet.close();
    statement.close();
    return numberOfEntities == 1;
}

```

Ове операције су генеричне и свим доменским класама су обезбеђене ове функционалности у њиховим конкретним брокерима помоћу наслеђивања апстрактног, генеричног брокера.

Помоћу класе SqlConnectionFactory са статичким методама имплементиран је “*connection pool*” у ком се чувају конекције са базом генерисане за покренути програм апликације. Приликом позивања системских операција, оне узимају слободну конекцију из “*connection pool*” и враћају је при завршетку обаваљања операције.

```

public class SqlConnectionFactory {
    private static List<Connection> connectionPool;
    private static List<Connection> usedConnections = new ArrayList<>();
    private static final int INITIAL_POOL_SIZE = 10;
    private static final int MAX_POOL_SIZE = 30;
    private static final int MAX_TIMEOUT = 10000;
    private static IConfigurationManager configManager;

    public static void initialize(IConfigurationManager configManager) throws Exception{
        SqlConnectionFactory.configManager = configManager;
        connectionPool = new ArrayList<>(INITIAL_POOL_SIZE);
        for (int i = 0; i < INITIAL_POOL_SIZE; i++) {
            connectionPool.add(createConnection());
        }
    }

    private static Connection createConnection() throws Exception{
        if(configManager == null)
            throw new Exception("Missing ConfigurationManager!");
        return DriverManager.getConnection(configManager.getConfigParam(ConfigParamKeys.URL), configManager.getConfigParam(ConfigParamKeys.USER),
            configManager.getConfigParam(ConfigParamKeys.PASSWORD));
    }

    public static Connection getConnection() throws Exception {
        if (connectionPool.isEmpty()) {
            if (usedConnections.size() < MAX_POOL_SIZE) {
                connectionPool.add(createConnection());
            } else {
                throw new RuntimeException("Maximum pool size reached, no available connections!");
            }
        }
        Connection connection = connectionPool.remove(connectionPool.size() - 1);
        if(connection == null || connection.isValid(MAX_TIMEOUT) == false){
            connection = createConnection();
        }
        usedConnections.add(connection);
        connection.setAutoCommit(false);
        return connection;
    }

    public static boolean releaseConnection(Connection connection) throws SQLException {
        if(connection == null || connection.isValid(MAX_TIMEOUT) == false)
            return false;
        if(usedConnections.remove(connection) == true)
            return connectionPool.add(connection);
        else return false;
    }

    public static void shutdown() throws SQLException {
        //usedConnections.forEach(this::releaseConnection);
        for (Connection usedConnection : usedConnections) {
            releaseConnection(usedConnection);
        }
        for (Connection connection : connectionPool) {
            if(connection != null && connection.isClosed() == false){
                connection.close();
            }
        }
        connectionPool.clear();
        configManager = null;
    }
}

```

### 3.4.2 Пројектовање складишта података

На основу доменских класа софтвера пројектоване су табеле (складишта података) релационог система за управљање базом података. Систем за управљање базом података који је коришћен у студијском примеру је MySQL.

Класе су креиране путем следећег SQL кода:

```
CREATE TABLE author(  
  ID BIGINT PRIMARY KEY NOT NULL AUTO_INCREMENT,  
  firstname VARCHAR(200),  
  lastname VARCHAR(200) NOT NULL  
);
```

```
CREATE TABLE MEMBER(  
  ID BIGINT PRIMARY KEY NOT NULL AUTO_INCREMENT,  
  firstname VARCHAR(200) NOT NULL,  
  lastname VARCHAR(200) NOT NULL,  
  birthday DATE NOT NULL,  
  email VARCHAR(255)  
);
```

```
CREATE TABLE building(  
  ID BIGINT PRIMARY KEY NOT NULL AUTO_INCREMENT,  
  NAME VARCHAR(100) NOT NULL  
);
```

```
CREATE TABLE employee(  
  ID BIGINT PRIMARY KEY NOT NULL,  
  firstname VARCHAR(200) NOT NULL,  
  lastname VARCHAR(200) NOT NULL,  
  PASSWORD VARCHAR(128) NOT NULL,  
  buildingID BIGINT NOT NULL,  
  CONSTRAINT employee_building_fk FOREIGN KEY (buildingID) REFERENCES building(ID) ON UPDATE CASCADE  
);
```

```

CREATE TABLE book(
ID BIGINT PRIMARY KEY NOT NULL AUTO_INCREMENT,
title VARCHAR(200) NOT NULL,
DESCRIPTION TEXT,
authorID BIGINT,
CONSTRAINT book_author_fk FOREIGN KEY (authorID) REFERENCES author(ID) ON UPDATE CASCADE
);

```

```

CREATE TABLE copyofbook(
ID BIGINT NOT NULL AUTO_INCREMENT,
bookID BIGINT NOT NULL,
buildingID BIGINT,
PRIMARY KEY(ID, bookID),
CONSTRAINT copyofbook_book_fk FOREIGN KEY (bookID) REFERENCES book(ID) ON UPDATE CASCADE,
CONSTRAINT copyofbook_building_fk FOREIGN KEY (buildingID) REFERENCES building(ID) ON UPDATE CASCADE
);

```

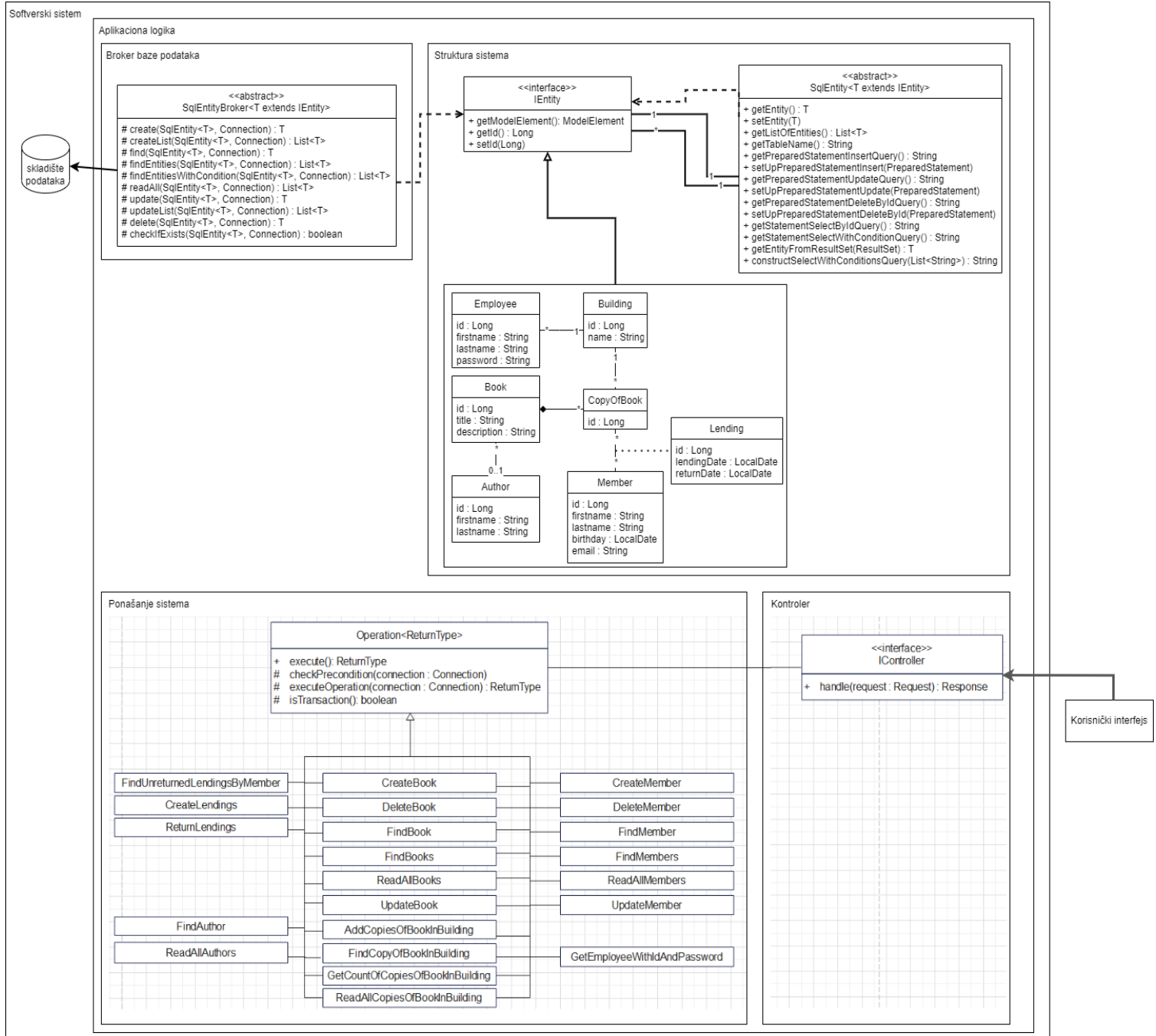
```

CREATE TABLE lending(
ID BIGINT NOT NULL AUTO_INCREMENT,
bookID BIGINT NOT NULL,
copyofbookID BIGINT NOT NULL,
memberID BIGINT NOT NULL,
lending_date DATE NOT NULL,
return_date DATE,
PRIMARY KEY(ID, bookID, copyofbookID, memberID),
CONSTRAINT lending_book_fk FOREIGN KEY (bookID) REFERENCES book(ID) ON UPDATE CASCADE,
CONSTRAINT lending_copyofbook_fk FOREIGN KEY (copyofbookID) REFERENCES copyofbook(ID) ON UPDATE CASCADE,
CONSTRAINT lending_member_fk FOREIGN KEY (memberID) REFERENCES MEMBER(ID) ON UPDATE CASCADE
);

```

### 3.5 Коначна архитектура софтверског система

На основу свих претходно дефинисаних целина, можемо стећи коначну слику целокупног софтверског система.



## Имплементација

Софтверски систем је развијен у програмском језику Java. Пројектован је као клијент-сервер апликација. Као систем за управљање базом података коришћен је MySQL, а као развојно окружење NetBeans IDE 14. На основу архитектуре софтверског система добијене су следеће софтверске класе:

- LibraryApp.Client
  - forms/LoginForm
  - forms/MainForm
  - forms/MainImagePanel
  - forms.book/BookForm
  - forms.book/ViewBooksForm
  - forms.book.table/BooksTableModel
  - forms.controllers/BookFormsController
  - forms.controllers/IClosable
  - forms.controllers/LendingFormsController
  - forms.controllers/LoginFormsController
  - forms.controllers/MainFormsController
  - forms.controllers/MemberFormsController
  - forms.controllers/ReturnLendingFormsController
  - forms.lending/LendingForm
  - forms.lending/ReturnLendingForm
  - forms.lending.table/CopiesOfBookTableModel
  - forms.lending.table/LendingsTableModel
  - forms.member/MemberForm
  - forms.member/ViewMembersForm
  - forms.member.table/MembersTableModel
  - main/Client
  - services/AuthService
  - services/BookService
  - services/CopyOfBookService
  - services/EntityService
  - services/LendingService
  - services/MemberService
  - session/Session
  - tcp/TcpClient
  - validations/BookDtoValidator
  - validations/EmployeeValidator
  - validations/MemberValidator
  - validations/Validator
  - validations.exceptions/ValidationException

- LibraryApp.Domain
  - message/Method
  - message/ModelElement
  - message/Request
  - message/Response
  - models/Author
  - models/Book
  - models/Building
  - models/CopyOfBook
  - models/Employee
  - models/IEntity
  - models/Lending
  - models/Member
  - models.dto/BookDto
  - tcp/TcpCommunicator
- LibraryApp.Server
  - controllers.impl/AuthorController
  - controllers.impl/BookController
  - controllers.impl/CopyOfBookController
  - controllers.impl/EmployeeController
  - controllers.impl/LendingController
  - controllers.impl/MemberController
  - controllers.interfaces/IController
  - database.configurations/ConfigFilePaths
  - database.configurations/ConfigParamKeys
  - database.configurations/IConfigurationManager
  - database.configurations/SqlJsonFileConfigurationManager
  - database.sql.brokers.impl/SqlAuthorBroker
  - database.sql.brokers.impl/SqlBookBroker
  - database.sql.brokers.impl/SqlBuildingBroker
  - database.sql.brokers.impl/SqlCopyOfBookBroker
  - database.sql.brokers.impl/SqlEmployeeBroker
  - database.sql.brokers.impl/SqlEntityBroker
  - database.sql.brokers.impl/SqlLendingBroker
  - database.sql.brokers.impl/SqlMemberBroker
  - database.sql.brokers.interfaces/IAuthorBroker
  - database.sql.brokers.interfaces/IBookBroker
  - database.sql.brokers.interfaces/IBuildingBroker
  - database.sql.brokers.interfaces/ICopyOfBookBroker
  - database.sql.brokers.interfaces/IEmployeeBroker
  - database.sql.brokers.interfaces/ILendingBroker
  - database.sql.brokers.interfaces/IMemberBroker
  - database.sql.connection/SqlConnectionFactory
  - database.sql.migrations/ddl



- database.sql.sqlmodels/SqlAuthor
- database.sql.sqlmodels/SqlBook
- database.sql.sqlmodels/SqlBuilding
- database.sql.sqlmodels/SqlCopyOfBook
- database.sql.sqlmodels/SqlEmployee
- database.sql.sqlmodels/SqlEntity
- database.sql.sqlmodels/SqlLending
- database.sql.sqlmodels/SqlMember
- forms/EmployeesTableModel
- forms/ServerEmployeesForm
- forms/ServerForm
- forms/ServerSettingsForm
- helper/EntitiesConverter
- helper/HashPassword
- helper/RandomID
- logics.impl/AuthorLogic
- logics.impl/BookLogic
- logics.impl/EmployeeLogic
- logics.impl/LendingLogic
- logics.impl/MemberLogic
- logics.interfaces/IAuthorLogic
- logics.interfaces/IBookLogic
- logics.interfaces/IEmployeeLogic
- logics.interfaces/ILendingLogic
- logics.interfaces/IMemberLogic
- logics.operations/Operation
- logics.operations.author/FindAuthor
- logics.operations.author/ReadAllAuthors
- logics.operations.book/AddCopiesOfBookInBuilding
- logics.operations.book/CreateBook
- logics.operations.book/DeleteBook
- logics.operations.book/FindBook
- logics.operations.book/FindBooks
- logics.operations.book/FindCopyOfBookInBuilding
- logics.operations.book/GetCountOfCopiesOfBookInBuilding
- logics.operations.book/ReadAllBooks
- logics.operations.book/ReadAllCopiesOfBookInBuilding
- logics.operations.book/UpdateBook
- logics.operations.employee/GetEmployeeWithIdAndPassword
- logics.operations.lending/CreateLendings
- logics.operations.lending/FindUnreturnedLendingsByMember
- logics.operations.lending/ReturnLendings
- logics.operations.member/CreateMember
- logics.operations.member/DeleteMember

- logics.operations.member/FindMember
- logics.operations.member/FindMembers
- logics.operations.member/ReadAllMembers
- logics.operations.member/UpdateMember
- main/ClientHandler
- main/Server
- tcp/TcpServer

## Тестирање

Сваки од имплементираних случајева коришћења је тестиран. Приликом тестирања сваког случаја коришћења, поред унетих правилних података, уношени су и неправилни подаци да би се утврдило какав ће бити резултат извршења. Након фазе тестирања, софтвер је спреман за коришћење од стране крајњег корисника.

## Литература

1. Проф.др. Синиша Влајић, Пројектовање софтвера(скрипта), Београд 2020.