Семинарски рад из Пројектовања софтвера

**Софтверски систем библиотеке**

**Студент**: **Ментор**:

Јелена Марјановић 91/14 др Синиша Влајић

Садржај

1. Вербални опис система 4

2. Случајеви коришћења 5

2.1 СК – Унос новог члана библиотеке 6

2.2 СК – Претраживање чланова библиотеке 7

2.3 СК – Измена члана библиотеке 8

2.4 СК – Брисање члана библиотеке 9

2.5 СК – Унос нове књиге 10

2.6 СК – Претраживање књига 11

2.7 СК – Измена књиге 12

2.8 СК – Брисање књиге 13

2.9 СК – Задуживање 14

2.10 СК – Раздуживање 15

3. Анализа 16

3.1 Понашање софтверског система: Системски дијаграм секвенци 16

3.1.1 ДС – Унос новог члана библиотеке 16

3.1.2 ДС – Претраживање чланова библиотеке 18

3.1.3 ДС – Измена члана библиотеке 20

3.1.4 ДС – Брисање члана библиотеке 23

3.1.5 ДС – Унос нове књиге 26

3.1.6 ДС – Претраживање књига 27

3.1.7 ДС – Измена књиге 29

3.1.8 ДС – Брисање књиге 32

3.1.9 ДС – Задуживање 35

3.1.10 ДС – Раздуживање 37

3.1.11 Списак системских операција 40

3.2 Понашање софтверског система: Дефинисање уговора о системским операцијама 41

3.2.1 Уговор УГ1: KreirajClana(Clan) 41

3.2.2 Уговор УГ2: PretrazivanjeClanova(kriterijum, List<Clan>) 41

3.2.3 Уговор УГ3: PronadjiClana(Clan, List<Clan>) 41

3.2.4 Уговор УГ4: PromeniClana(Clan) 42

3.2.5 Уговор УГ5: ObrisiClana(Clan) 42

3.2.6 Уговор УГ6: KreirajKnjigu(Knjiga) 42

3.2.7 Уговор УГ7: PretrazivanjeKnjiga(kriterijum, List<Knjiga>) 42

3.2.8 Уговор УГ8: PronadjiKnjigu(Knjiga, List<Knjiga>) 43

3.2.9 Уговор УГ9: PromeniKnjigu(Knjiga) 43

3.2.10 Уговор УГ10: ObrisiKnjigu(Knjiga) 43

3.2.11 Уговор УГ11: UcitajKnjige() 43

3.2.12 Уговор УГ12: UcitajClanoveBiblioteke() 44

3.2.13 Уговор УГ13: KreirajZaduzenje(Zaduzenje) 44

3.2.14 Уговор УГ14: PretrazivanjeZaduzenja(kriterijum, List<Zaduzenje>) 44

3.2.15 Уговор УГ15: PronadjiZaduzenje(Zaduzenje, List<Zaduzenje>) 44

3.2.16 Уговор УГ16: PromeniZaduzenje(Zaduzenje) 45

3.3 Структура софтверског система: Концептуални модел 46

3.4 Структура софтверског система: Релациони модел 48

3.4.1 Табеле ограничења 48

4. Пројектовање 51

4.1 Пројектовање корисничког интерфејса 52

4.1.1 Пројектовање екранске форме 52

Пројектовање сценарија СК 54

4.1.2 Пројектовање контролера корисничког интерфејса 88

4.2 Пројектовање апликационе логике 97

4.2.1 Контролер апликационе логике 97

4.2.2 Пословна логика 97

Пројектовање понашања (системске операције) 97

Пројектовање структуре софтверског система (доменске класе) 118

4.3 Пројектовање складишта података 121

5. Имплементација 127

6. Закључак 130

7. Литература 131

1. Вербални опис система

Софтверски систем библиотеке користи библиотекар како би управљао члановима и књигама које се налазе у систему, као и да бележи и прати задужења чланова библиотеке. Предуслов за коришћење система јесте да је библиотекар улогован.

У систему библиотеке се налазе подаци о постојећим члановима, које је могуће претраживати, мењати и брисати, уколико појединац не жели више да буде учлањен, а такође, могуће је додавати нове чланове. Осим чланова, систем чува и податке о књигама. Води се евиденција о свим задуженим и расположивим књигама, све их је могуће мењати, претраживати и брисати, а у случају проширења понуде књига, нове је потребно додати у систем.

За сваког члана се води евиденција и о узетим књигама, и када је исте вратио у библиотеку.

2. Случајеви коришћења

У овом поглављу наводим све случајеве коришћења моје апликације. Библитекар користи систем како би управљао члановима библиотеке и књигама које се налазе у систему.



*Слика 1 - Случајеви коришћења*

# 2.1 СК – Унос новог члана библиотеке

**Назив СК**

Креирање члана

**Актори СК**

Библиотекар

**Учесници СК**

Библиотекар и систем (програм)

**Предуслов**: Систем је укључен и библиотекар je улогован под својом шифром. Систем приказује форму за рад са чланом.

**Основни сценарио СК**

1. Библиотекар **уноси** податке о члану. (АПУСО)
2. Библиотекар **контролише** да ли је коректно унео податке о члану. (АНСО)
3. Библиотекар **позива** систем да креира и запамти податке о члану. (АПСО)
4. Систем **памти** податке о члану. (СО)
5. Систем **приказује** библиотекару запамћеног члана и поруку: “Систем је креирао члана“. (ИА)

Алтернативна сценарија

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

# 2.2 СК – Претраживање чланова библиотеке

**Назив СК**

Претраживање члана

**Актори СК**

Библиотекар

**Учесници СК**

Библиотекар и систем (програм)

**Предуслов**: Систем је укључен и библиотекар je улогован под својом шифром. Систем приказује форму за рад са чланом.

**Основни сценарио СК**

1. Библиотекар **уноси** вредностпо којој претражује чланове. (АПУСО)
2. Библиотекар **позива** систем да нађе чланове по задатој вредности. (АПСО)
3. Систем **тражи** чланове по задатој вредности. (СО)
4. Систем приказује библиотекару податке о члановима и поруку: “Систем је нашао чланове по задатој вредности”. (ИА)
5. Библиотекар **бира** члана. (АПУСО)
6. Библиотекар **позива** систем да нађе податке о одабраном члану(АПСО)
7. Систем **тражи** члана по задатој вредности. (СО)
8. Систем приказује библиотекару податке о члану и поруку: “Систем је нашао члана по задатој вредности”. (ИА)

Алтернативна сценарија

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

8.1 Уколико систем не може да нађе члана он приказује библиотекару поруку: “Систем не може да нађе члана по задатој вредности”. (ИА)

# 2.3 СК – Измена члана библиотеке

**Назив СК**

Промена члана

**Актори СК**

Библиотекар

**Учесници СК**

Библиотекар и систем (програм)

**Предуслов**: Систем је укључен и библиотекар je улогован под својом шифром. Систем приказује форму за рад са чланом.

**Основни сценарио СК**

1. Библиотекар **уноси** вредностпо којој претражује чланове. (АПУСО)
2. Библиотекар **позива** систем да нађе чланове по задатој вредности. (АПСО)
3. Систем **тражи** чланове по задатој вредности. (СО)
4. Систем приказује библиотекару чланове и поруку: “Систем је нашао чланове по задатој вредности”. (ИА)
5. Библиотекар **бира** члана. (АПУСО)
6. Библиотекар **позива** систем да нађе податке о одабраном члану(АПСО)
7. Систем **тражи** члана по задатој вредности. (СО)
8. Систем приказује библиотекару податке о члану и поруку: “Систем је нашао члана по задатој вредности”. (ИА)
9. Библиотекар **уноси** **(мења)** податке о члану. (АПУСО)
10. Библиотекар **контролише** да ли је коректно унео податке о члану. (АНСО)
11. Библиотекар **позива** систем да промени податке о члану. (АПСО)
12. Систем **мења** податке о члану. (СО)
13. Систем **приказује** библиотекару промењеног члана и поруку: “Систем је успешно променио члана.” (ИА)

Алтернативна сценарија

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

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

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

# 2.4 СК – Брисање члана библиотеке

**Назив СК**

Брисање члана

**Актори СК**

Библиотекар

**Учесници СК**

Библиотекар и систем (програм)

**Предуслов**: Систем је укључен и библиотекар je улогован под својом шифром. Систем приказује форму за рад са чланом.

**Основни сценарио СК**

1. Библиотекар **уноси** вредностпо којој претражује чланове. (АПУСО)
2. Библиотекар **позива** систем да нађе чланове по задатој вредности. (АПСО)
3. Систем **тражи** чланове по задатој вредности. (СО)
4. Систем приказује библиотекару чланове и поруку: “Систем је нашао чланове по задатој вредности”. (ИА)
5. Библиотекар **бира** члана. (АПУСО)
6. Библиотекар **позива** систем да нађе податке о одабраном члану(АПСО)
7. Систем **тражи** члана по задатој вредности. (СО)
8. Систем приказује библиотекару податке о члану и поруку: “Систем је нашао члана по задатој вредности”. (ИА)
9. Библиотекар **позива** систем да обрише члана. (АПСО)
10. Систем **брише** члана. (СО)
11. Систем **приказује** библиотекару поруку: “Систем је обрисао члана.” (ИА)

Алтернативна сценарија

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

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

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

# 2.5 СК – Унос нове књиге

**Назив СК**

Креирање књиге

**Актори СК**

Библиотекар

**Учесници СК**

Библиотекар и систем (програм)

**Предуслов**: Систем је укључен и библиотекар je улогован под својом шифром. Систем приказује форму за рад са књигом.

**Основни сценарио СК**

1. Библиотекар **уноси** податке о књизи. (АПУСО)
2. Библиотекар **контролише** да ли је коректно унео податке о књизи. (АНСО)
3. Библиотекар **позива** систем да креира и запамти податке о књизи. (АПСО)
4. Систем **памти** податке о књизи. (СО)
5. Систем **приказује** библиотекару запамћену књигу и поруку: “Систем је креирао књигу“. (ИА)

Алтернативна сценарија

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

# 2.6 СК – Претраживање књига

**Назив СК**

Претраживање књига

**Актори СК**

Библиотекар

**Учесници СК**

Библиотекар и систем (програм)

**Предуслов**: Систем је укључен и библиотекар je улогован под својом шифром. Систем приказује форму за рад са књигом.

**Основни сценарио СК**

1. Библиотекар **уноси** вредностпо којој претражује књиге. (АПУСО)
2. Библиотекар **позива** систем да нађе књиге по задатој вредности. (АПСО)
3. Систем **тражи** књиге по задатој вредности. (СО)
4. Систем приказује библиотекару податке о књигама и поруку: “Систем је нашао књиге по задатој вредности”. (ИА)
5. Библиотекар **бира** књигу. (АПУСО)
6. Библиотекар **позива** систем да нађе податке о одабраној књизи(АПСО)
7. Систем **тражи** књигу по задатој вредности. (СО)
8. Систем приказује библиотекару податке о књизи и поруку: “Систем је нашао књигу по задатој вредности”. (ИА)

Алтернативна сценарија

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

8.1 Уколико систем не може да нађе књигу он приказује библиотекару поруку: “Систем не може да нађе књигу по задатој вредности”. (ИА)

# 2.7 СК – Измена књиге

**Назив СК**

Промена књиге

**Актори СК**

Библиотекар

**Учесници СК**

Библиотекар и систем (програм)

**Предуслов**: Систем је укључен и библиотекар je улогован под својом шифром. Систем приказује форму за рад са књигом.

**Основни сценарио СК**

1. Библиотекар **уноси** вредностпо којој претражује књиге. (АПУСО)
2. Библиотекар **позива** систем да нађе књиге по задатој вредности. (АПСО)
3. Систем **тражи** књиге по задатој вредности. (СО)
4. Систем приказује библиотекару податке о књигама и поруку: “Систем је нашао књиге по задатој вредности”. (ИА)
5. Библиотекар **бира** књигу. (АПУСО)
6. Библиотекар **позива** систем да нађе податке о одабраној књизи(АПСО)
7. Систем **тражи** књигу по задатој вредности. (СО)
8. Систем приказује библиотекару податке о књизи и поруку: “Систем је нашао књигу по задатој вредности”. (ИА)
9. Библиотекар **уноси** **(мења)** податке о књизи. (АПУСО)
10. Библиотекар **контролише** да ли је коректно унео податке о књизи. (АНСО)
11. Библиотекар **позива** систем да промени податке о књизи. (АПСО)
12. Систем **мења** податке о књизи. (СО)
13. Систем **приказује** библиотекару промењену књигу и поруку: “Систем је успешно променио књигу.” (ИА)

Алтернативна сценарија

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

8.1 Уколико систем не може да нађе књигу он приказује библиотекару поруку: “Систем не може да нађе књигу по задатој вредности”. Прекида се извршење сценарија. (ИА)

13.1 Уколико систем не може да запамти податке о књизи он приказује библиотекару поруку “Систем не може да промени књигу”. (ИА)

# 2.8 СК – Брисање књиге

**Назив СК**

Брисање књиге

**Актори СК**

Библиотекар

**Учесници СК**

Библиотекар и систем (програм)

**Предуслов**: Систем је укључен и библиотекар je улогован под својом шифром. Систем приказује форму за рад са књигом.

**Основни сценарио СК**

1. Библиотекар **уноси** вредностпо којој претражује књиге. (АПУСО)
2. Библиотекар **позива** систем да нађе књиге по задатој вредности. (АПСО)
3. Систем **тражи** књиге по задатој вредности. (СО)
4. Систем приказује библиотекару податке о књигама и поруку: “Систем је нашао књиге по задатој вредности”. (ИА)
5. Библиотекар **бира** књигу. (АПУСО)
6. Библиотекар **позива** систем да нађе податке о одабраној књизи(АПСО)
7. Систем **тражи** књигу по задатој вредности. (СО)
8. Систем приказује библиотекару податке о књизи и поруку: “Систем је нашао књигу по задатој вредности”. (ИА)
9. Библиотекар **позива** систем да обрише књигу. (АПСО)
10. Систем **брише** књигу. (СО)
11. Систем **приказује** библиотекару поруку: “Систем је обрисао књигу.” (ИА)

Алтернативна сценарија

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

8.1 Уколико систем не може да нађе књигу он приказује библиотекару поруку: “Систем не може да нађе књигу по задатој вредности”. Прекида се извршење сценарија. (ИА)

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

# 2.9 СК – Задуживање

**Назив СК**

Креирање задужења

**Актори СК**

Библиотекар

**Учесници СК**

Библиотекар и систем (програм)

**Предуслов**: Систем је укључен и библиотекар je улогован под својом шифром. Систем приказује форму за рад са задужењем. Учитана је листа чланова и књига.

**Основни сценарио СК**

1. Библиотекар **уноси** податке о задужењу. (АПУСО)
2. Библиотекар **контролише** да ли је коректно унео податке о задужењу. (АНСО)
3. Библиотекар **позива** систем да креира и запамти податке о задужењу. (АПСО)
4. Систем **памти** податке о задужењу. (СО)
5. Систем **приказује** библиотекару запамћено задужење и поруку: “Систем је креирао задужење“. (ИА)

Алтернативна сценарија

5.1 Уколико систем не може да запамти податке о задужењу он приказује библиотекару поруку “Систем не може да креира задужење”. (ИА)

# 2.10 СК – Раздуживање

**Назив СК**

Измена задужења

**Актори СК**

Библиотекар

**Учесници СК**

Библиотекар и систем (програм)

**Предуслов**: Систем је укључен и библиотекар je улогован под својом шифром. Систем приказује форму за рад са задужењем.

**Основни сценарио СК**

1. Библиотекар **уноси** вредностпо којој претражује задужења. (АПУСО)
2. Библиотекар **позива** систем да нађе задужења по задатој вредности. (АПСО)
3. Систем **тражи** задужења по задатој вредности. (СО)
4. Систем приказује библиотекару задужења и поруку: “Систем је нашао задужења по задатој вредности”. (ИА)
5. Библиотекар **бира** задужење. (АПУСО)
6. Библиотекар **позива** систем да нађе податке о одабраном задужењу (АПСО)
7. Систем **тражи** задужење по задатој вредности. (СО)
8. Систем приказује библиотекару податке о задужењу и поруку: “Систем је нашао задужење по задатој вредности”. (ИА)
9. Библиотекар **уноси** **(мења)** податке о задужењу. (АПУСО)
10. Библиотекар **контролише** да ли је коректно унео податке о задужењу. (АНСО)
11. Библиотекар **позива** систем да промени податке о задужењу. (АПСО)
12. Систем **мења** податке о задужењу. (СО)
13. Систем **приказује** библиотекару промењено задужење и поруку: “Систем је променио задужење.” (ИА)

Алтернативна сценарија

4.1 Уколико систем не може да нађе ниједно задужење он приказује библиотекару поруку: “Систем не може да нађе ниједно задужење по задатој вредности”. Прекида се извршење сценарија. (ИА)

8.1 Уколико систем не може да нађе задужење он приказује библиотекару поруку: “Систем не може да нађе задужење по задатој вредности”. Прекида се извршење сценарија. (ИА)

13.1 Уколико систем не може да промени податке о задужење он приказује библиотекару поруку “Систем не може да промени задужење”. (ИА)

3. Анализа

У фази анализе, описујемо логичку структуру и понашање софтверског система, односно пословну логику система.

Понашање описујемо помоћу дијаграма секвенци и уговора о системским операцијама, које добијамо на основу дијаграма секвенци. Структуру система описујемо помоћу концептуалног (Дијаграм класа) и релационог модела.

# 3.1 Понашање софтверског система: Системски дијаграм секвенци

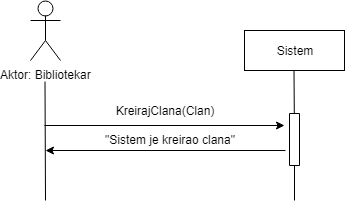
Системски дијаграм секвенци приказује редослед догађаја за конкретан случај коришћења, и њих правимо на основу претходно описаних СК (само АПСО и ИА).

Актор позива системску операцију посредством примаоца догађаја.

## 3.1.1 ДС – Унос новог члана библиотеке

**Основни сценарио**

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



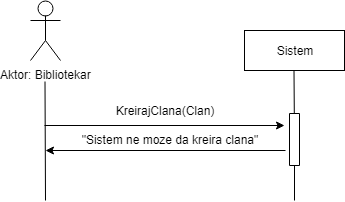
*Слика 2 - ДС Унос новог члана библиотеке*

Идентификоване системске операције:

1. **KreirajClana**(Clan)

**Алтернативни сценарио 1**

2.1 Уколико систем не може да креира и запамти члана он приказује библиотекару поруку “Систем не може да кеира члана”. (ИА)

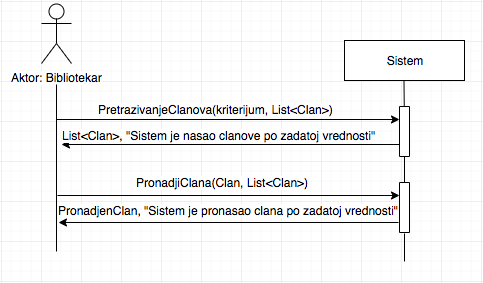
****

*Слика 4 - ДС Систем не може да запамти члана*

## 3.1.2 ДС – Претраживање чланова библиотеке

**Основни сценарио**

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



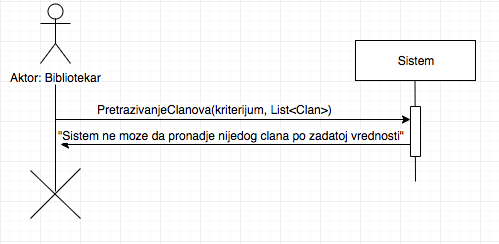
*Слика 5 - ДС Претраживање чланова библиотеке*

Идентификоване системске операције:

1. **PretrazivanjeClanova**(kriterijum, List<Clan>)
2. **PronadjiClana**(Clan, List<Clan>)

**Алтернативни сценарио 1**

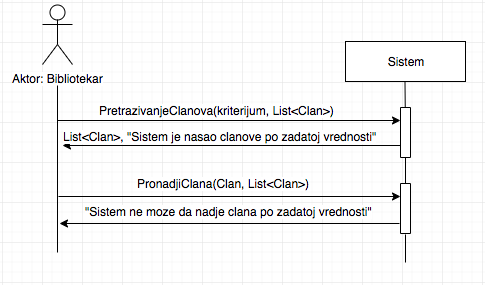
2.1 Уколико систем не може да нађе ниједног члана он приказује библиотекару поруку: “Систем не може да нађе ниједног члана по задатој вредности”. Прекида се извршење сценарија. (ИА)



*Слика 6 - ДС Систем не може да нађе ниједног члана*

**Алтернативни сценарио 2**

4.1 Уколико систем не може да нађе члана он приказује библиотекару поруку: “Систем не може да нађе члана по задатој вредности”. (ИА)

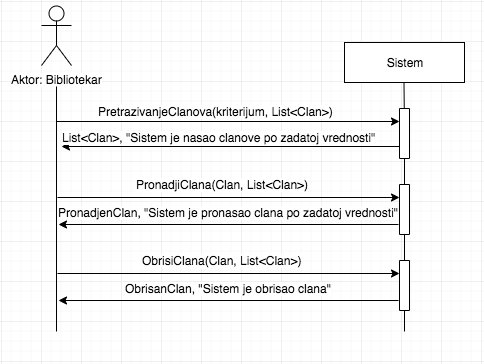


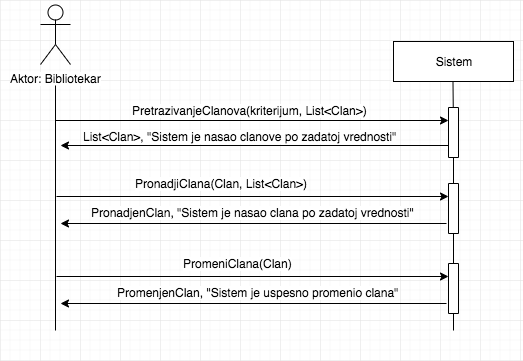
*Слика 7 – ДС Систем не може да нађе одабраног члана*

## 3.1.3 ДС – Измена члана библиотеке

**Основни сценарио**

1. Библиотекар **позива** систем да нађе чланове по задатој вредности. (АПСО)
2. Систем приказује библиотекару чланове и поруку: “Систем је нашао чланове по задатој вредности”. (ИА)
3. Библиотекар **позива** систем да нађе податке о одабраном члану(АПСО)
4. Систем приказује библиотекару податке о члану и поруку: “Систем је нашао члана по задатој вредности”. (ИА)
5. Библиотекар **позива** систем да промени податке о члану. (АПСО)
6. Систем **приказује** библиотекару промењеног члана и поруку: “Систем је успешно променио члана.” (ИА)





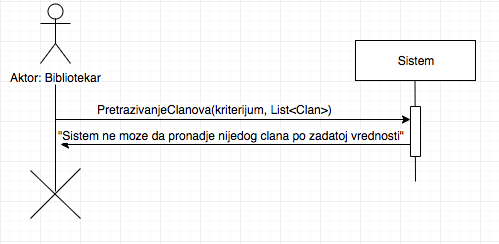
*Слика 8 - ДС Измена члана библиотеке*

Идентификоване системске операције:

1. **PretrazivanjeClanova**(kriterijum, List<Clan>)
2. **PronadjiClana**(Clan, List<Clan>)
3. **PromeniClana**(Clan)

**Алтернативни сценарио 1**

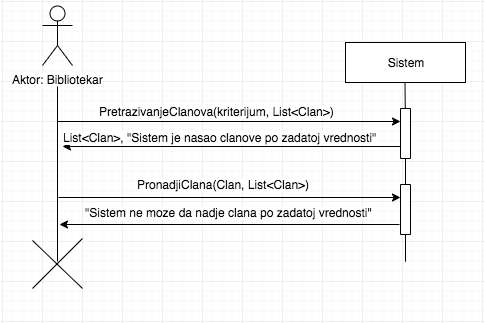
4.1 Уколико систем не може да нађе ниједног члана он приказује библиотекару поруку: “Систем не може да нађе ниједног члана по задатој вредности”. Прекида се извршење сценарија. (ИА)



*Слика 9 - ДС Систем не може да нађе ниједног члана*

**Алтернативни сценарио 2**

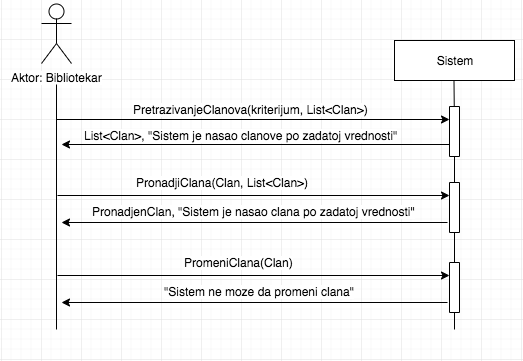
8.1 Уколико систем не може да нађе члана он приказује библиотекару поруку: “Систем не може да нађе члана по задатој вредности”. Прекида се извршење сценарија. (ИА)



*Слика 10 – ДС Систем не може да нађе одабраног члана*

**Алтернативни сценарио 3**

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

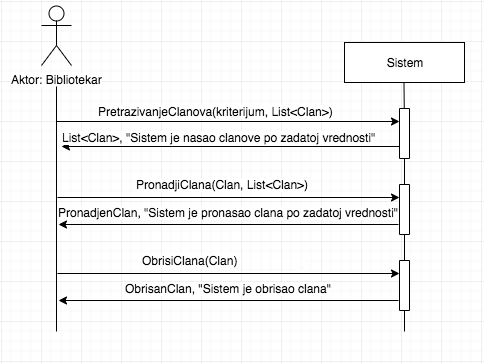


*Слика 11 –ДС Систем не може да измени члана*

## 3.1.4 ДС – Брисање члана библиотеке

**Основни сценарио**

1. Библиотекар **позива** систем да нађе чланове по задатој вредности. (АПСО)
2. Систем приказује библиотекару чланове и поруку: “Систем је нашао чланове по задатој вредности”. (ИА)
3. Библиотекар **позива** систем да нађе податке о одабраном члану(АПСО)
4. Систем приказује библиотекару податке о члану и поруку: “Систем је нашао члана по задатој вредности”. (ИА)
5. Библиотекар **позива** систем да обрише члана. (АПСО)
6. Систем **приказује** библиотекару поруку: “Систем је обрисао члана.” (ИА)



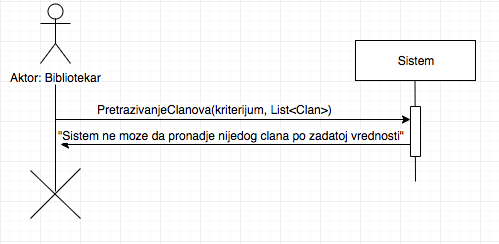
*Слика 12 - ДС Брисање члана библиотеке*

Идентификоване системске операције:

1. **PretrazivanjeClanova**(kriterijum, List<Clan>)
2. **PronadjiClana**(Clan, List<Clan>)
3. **ObrisiClana**(Clan)

**Алтернативни сценарио 1**

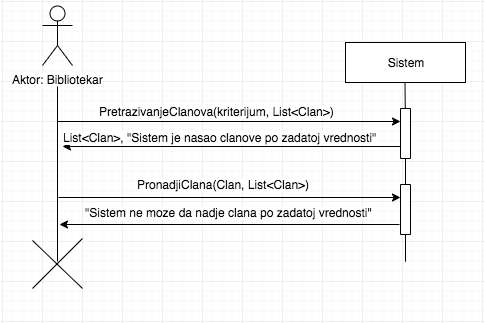
2.1 Уколико систем не може да нађе ниједног члана он приказује библиотекару поруку: “Систем не може да нађе ниједног члана по задатој вредности”. Прекида се извршење сценарија. (ИА)



*Слика 13 - ДС Систем не може да нађе ниједног члана*

**Алтернативни сценарио 2**

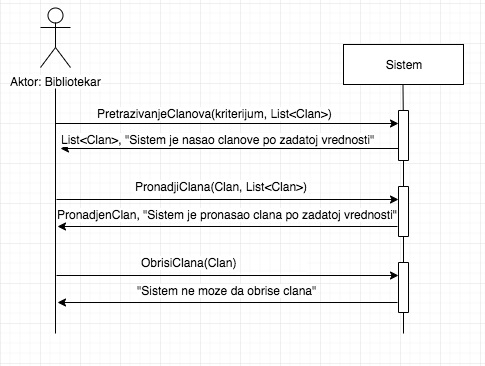
4.1 Уколико систем не може да нађе члана он приказује библиотекару поруку: “Систем не може да нађе члана по задатој вредности”. Прекида се извршење сценарија. (ИА)



*Слика 14 - ДС Систем не може да нађе одабраног члана*

**Алтернативни сценарио 3**

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

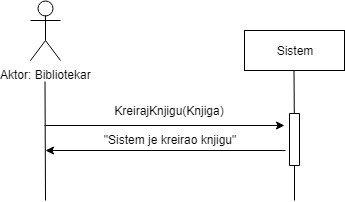


*Слика 15 - ДС Систем не може да обрише члана*

## 3.1.5 ДС – Унос нове књиге

**Основни сценарио**

1. Библиотекар **позива** систем да креира и запамти податке о књизи. (АПСО)
2. Систем **приказује** библиотекару запамћену књигу и поруку: “Систем је креирао књигу“. (ИА)



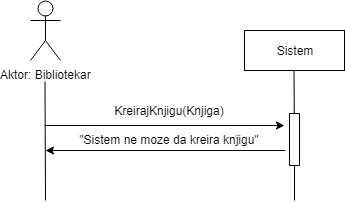
*Слика 16 - ДС Унос нове књиге*

Идентификоване системске операције:

1. **KreirajKnjigu**(Knjiga)

**Алтернативни сценарио 1**

2.1 Уколико систем не може да креира и запамти књигу он приказује библиотекару поруку: “Систем не може да креира књигу”. (ИА)

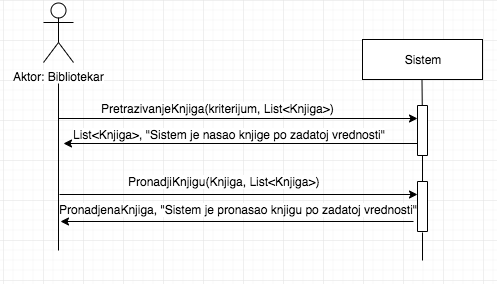


*Слика 17 - ДС Систем не може да креира књигу*

## 3.1.6 ДС – Претраживање књига

**Основни сценарио**

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



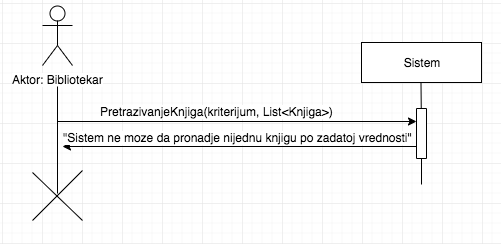
*Слика 19 – ДС Претраживање књига*

Идентификоване системске операције:

1. **PretrazivanjeKnjiga**(kriterijum, List<Knjiga>)
2. **PronadjiKnjigu**(Knjiga, List<Knjiga>)

**Алтернативни сценарио 1**

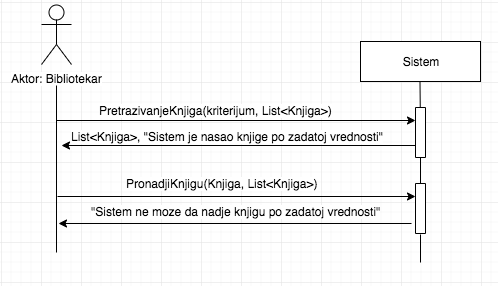
2.1 Уколико систем не може да нађе ниједну књигу он приказује библиотекару поруку: “Систем не може да нађе ниједну књигу по задатој вредности”. Прекида се извршење сценарија. (ИА)



*Слика 20 - ДС Систем не може да нађе ниједну књигу*

**Алтернативни сценарио 2**

4.1 Уколико систем не може да нађе књигу он приказује библиотекару поруку: “Систем не може да нађе књигу по задатој вредности”. (ИА)

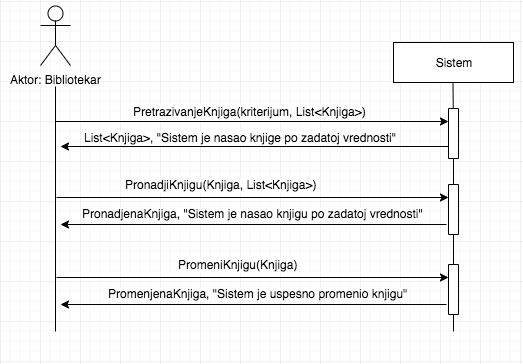
****

*Слика 21 - ДС Систем не може да нађе одабрану књигу*

## 3.1.7 ДС – Измена књиге

**Основни сценарио**

1. Библиотекар **позива** систем да нађе књиге по задатој вредности. (АПСО)
2. Систем приказује библиотекару податке о књигама и поруку: “Систем је нашао књиге по задатој вредности”. (ИА)
3. Библиотекар **позива** систем да нађе податке о одабраној књизи(АПСО)
4. Систем приказује библиотекару податке о књизи и поруку: “Систем је нашао књигу по задатој вредности”. (ИА)
5. Библиотекар **позива** систем да промени податке о књизи. (АПСО)
6. Систем **приказује** библиотекару промењену књигу и поруку: “Систем је успешно променио књигу.” (ИА)



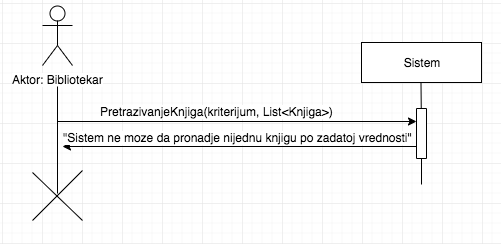
*Слика 22 - ДС Измена књиге*

Идентификоване системске операције:

1. **PretrazivanjeKnjiga**(kriterijum, List<Knjiga>)
2. **PronadjiKnjigu**(Knjiga, List<Knjiga>)
3. **PromeniKnjigu**(Knjiga)

**Алтернативни сценарио 1**

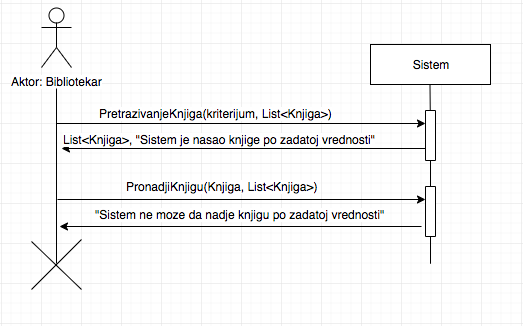
2.1 Уколико систем не може да нађе ниједну књигу он приказује библиотекару поруку: “Систем не може да нађе ниједну књигу по задатој вредности”. Прекида се извршење сценарија. (ИА)



*Слика 23 - ДС Систем не може да нађе ниједну књигу*

**Алтернативни сценарио 2**

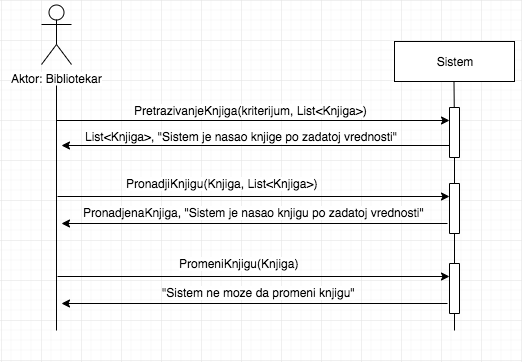
4.1 Уколико систем не може да нађе књигу он приказује библиотекару поруку: “Систем не може да нађе књигу по задатој вредности”. Прекида се извршење сценарија. (ИА)



*Слика 24 - ДС Систем не може да нађе одабрану књигу*

**Алтернативни сценарио 3**

6.1 Уколико систем не може да промени податке о књизи он приказује библиотекару поруку “Систем не може да промени књигу”. (ИА)



*Слика 25 - ДС Систем не може да измени књигу*

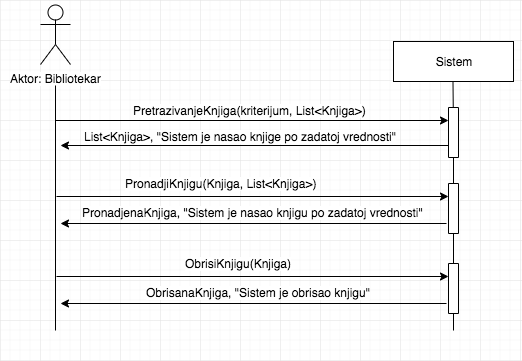
Идентификоване системске операције:

1. **PretrazivanjeKnjiga**(kriterijum, List<Knjiga>)
2. **PronadjiKnjigu**(Knjiga, List<Knjiga>)
3. **PromeniKnjigu**(Knjiga)

## 3.1.8 ДС – Брисање књиге

**Основни сценарио**

1. Библиотекар **позива** систем да нађе књиге по задатој вредности. (АПСО)
2. Систем приказује библиотекару податке о књигама и поруку: “Систем је нашао књиге по задатој вредности”. (ИА)
3. Библиотекар **позива** систем да нађе податке о одабраној књизи(АПСО)
4. Систем приказује библиотекару податке о књизи и поруку: “Систем је нашао књигу по задатој вредности”. (ИА)
5. Библиотекар **позива** систем да обрише књигу. (АПСО)
6. Систем **приказује** библиотекару поруку: “Систем је обрисао књигу.” (ИА)



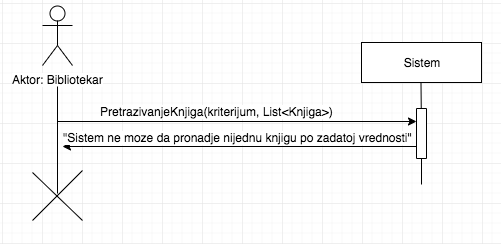
*Слика 26 - ДС Брисање књиге*

Идентификоване системске операције:

1. **PretrazivanjeKnjiga**(kriterijum, List<Knjiga>)
2. **PronadjiKnjigu**(Knjiga, List<Knjiga>)
3. **ObrisiKnjigu**(Knjiga)

**Алтернативни сценарио 1**

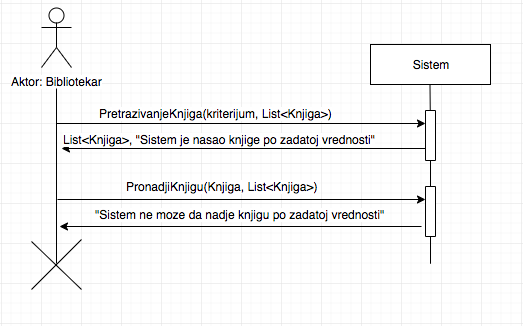
2.1 Уколико систем не може да нађе ниједну књигу он приказује библиотекару поруку: “Систем не може да нађе ниједну књигу по задатој вредности”. Прекида се извршење сценарија. (ИА)



*Слика 27 - ДС Систем не може да пронађе ниједну књигу*

**Алтернативни сценарио 2**

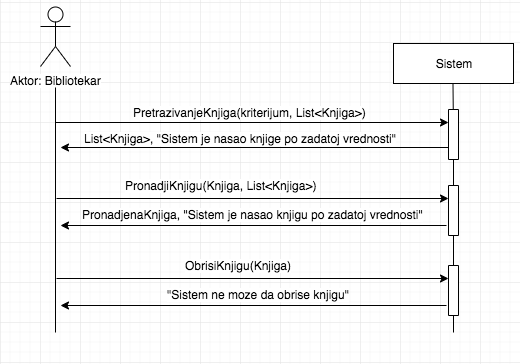
4.1 Уколико систем не може да нађе књигу он приказује библиотекару поруку: “Систем не може да нађе књигу по задатој вредности”. Прекида се извршење сценарија. (ИА)



*Слика 28 - ДС Систем не може да пронађе одабрану књигу*

**Алтернативни сценарио 3**

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

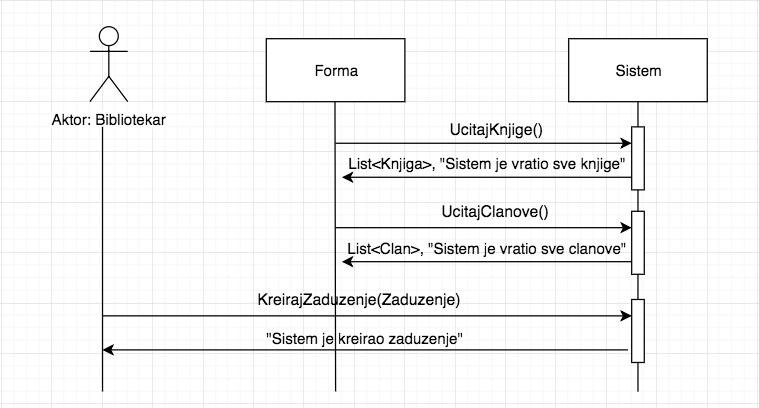


*Слика 29 - ДС Систем не може да обрише књигу*

## 3.1.9 ДС – Задуживање

**Основни сценарио**

1. Библиотекар **позива** систем да креира задужење. (АПСО)
2. Систем **приказује** библиотекару задужење и поруку: “Систем је креирао задужење“. (ИА)



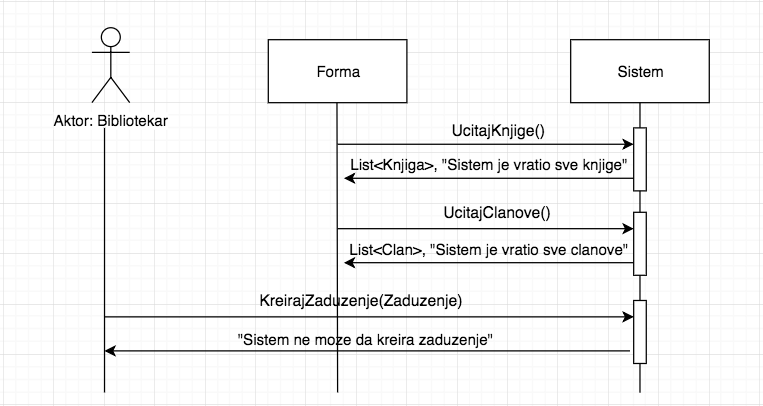
*Слика 30 - ДС Креирање задужења*

Идентификоване системске операције:

1. **UcitajKnjige()**
2. **UcitajClanove()**
3. **KreirajZaduzenje**(Zaduzenje)

**Алтернативни сценарио 1**

2.1 Уколико систем не може да креира задужење он приказује библиотекару поруку: “Систем не може да креира задужење”. (ИА)

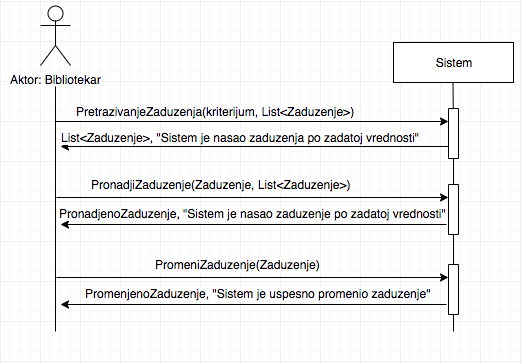


*Слика 31 - ДС Систем не може да креира задужење*

## 3.1.10 ДС – Раздуживање

**Основни сценарио**

1. Библиотекар **позива** систем да нађе задужења по задатој вредности. (АПСО)
2. Систем приказује библиотекару задужења и поруку: “Систем је нашао задужења по задатој вредности”. (ИА)
3. Библиотекар **позива** систем да нађе податке о одабраном задужењу (АПСО)
4. Систем приказује библиотекару податке о задужењу и поруку: “Систем је нашао задужење по задатој вредности”. (ИА)
5. Библиотекар **позива** систем да промени податке о задужењу. (АПСО)
6. Систем **приказује** библиотекару промењено задужење и поруку: “Систем је успешно променио задужење.” (ИА)

****

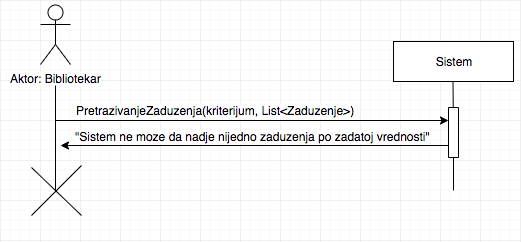
*Слика 33 - ДС Раздуживање*

Идентификоване системске операције:

1. **PretrazivanjeZaduzenja**(kriterijum, List<Zaduzenje>)
2. **PronadjiZaduzenje**(Zaduzenje, List<Zaduzenje>)
3. **PromeniZaduzenje**(Zaduzenje)

**Алтернативни сценарио 1**

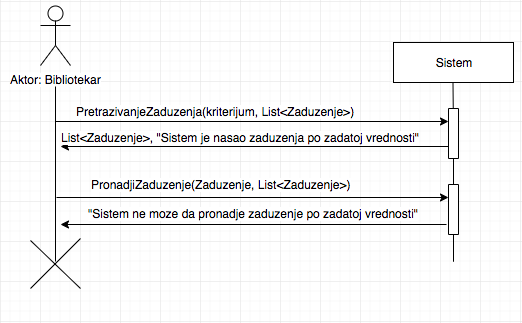
4.1 Уколико систем не може да нађе ниједно задужење он приказује библиотекару поруку: “Систем не може да нађе ниједно задужење по задатој вредности”. Прекида се извршење сценарија. (ИА)

****

*Слика 34 - ДС Систем не може да нађе ниједно задужење*

**Алтернативни сценарио 2**

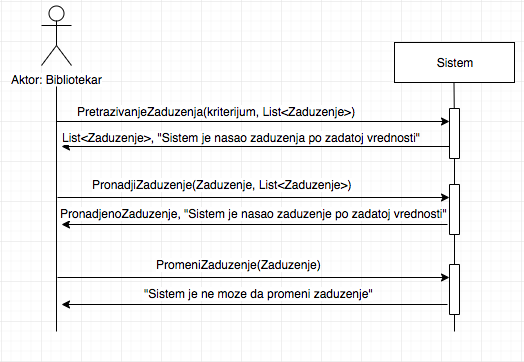
8.1 Уколико систем не може да нађе задужење он приказује библиотекару поруку: “Систем не може да нађе задужење по задатој вредности”. Прекида се извршење сценарија. (ИА)

****

*Слика 35 - ДС Систем не може да нађе одабрано задужење*

**Алтернативни сценарио 3**

13.1 Уколико систем не може да промени податке о задужење он приказује библиотекару поруку “Систем не може да промени задужење”. (ИА)

****

*Слика 36 - ДС Систем не може да промени задужење*

## 3.1.11 Списак системских операција

Као резултат анализе сценарија добијено је **16** системских операција које треба пројектовати:

1. signal: **KreirajClana(Clan)**
2. signal: **PretrazivanjeClanova(kriterijum, List<Clan>)**
3. signal: **PronadjiClana(Clan, List<Clan>)**
4. signal: **PromeniClana(Clan)**
5. signal: **ObrisiClana(Clan)**
6. signal: **KreirajKnjigu(Knjiga)**
7. signal: **PretrazivanjeKnjiga(kriterijum, List<Knjiga>)**
8. signal: **PronadjiKnjigu(Knjiga, List<Knjiga>)**
9. signal: **PromeniKnjigu(Knjiga)**
10. signal: **ObrisiKnjigu(Knjiga)**
11. signal: **UcitajKnjige()**
12. signal: **UcitajClanove()**
13. signal: **KreirajZaduzenje(Zaduzenje)**
14. signal: **PretrazivanjeZaduzenja(kriterijum, List<Zaduzenje>)**
15. signal: **PronadjiZaduzenje(Zaduzenje, List<Zaduzenje>)**
16. signal: **PromeniZaduzenje(Zaduzenje)**

# 3.2 Понашање софтверског система: Дефинисање уговора о системским операцијама

Системска операција представља понашање софтверског система. За сваку од уочених системских операција се праве *уговори,* који описују понашање саме операције, односно оно што она треба да одради, али не описују начин на који ће се то одрадити. Један уговор се везује за једну системску операцију.

## 3.2.1 Уговор УГ1: KreirajClana(Clan)

**Операција:** KreirajClana(Clan): signal;

**Веза са СК:** СК1

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

**Постуслови:** Креиран је нови члан.

## 3.2.2 Уговор УГ2: PretrazivanjeClanova(kriterijum, List<Clan>)

**Операција:** PretrazivanjeClanova(kriterijum*,* List<Clan>): signal;

**Веза са СК:** СК2, СК3, СК4

**Предуслови:** /

**Постуслови:** /

## 3.2.3 Уговор УГ3: PronadjiClana(Clan, List<Clan>)

**Операција:** PronadjiClana(Clan*,* List<Clan>): signal;

**Веза са СК:** СК2, СК3, СК4

**Предуслови:** /

**Постуслови:** /

## 3.2.4 Уговор УГ4: PromeniClana(Clan)

**Операција:** PromeniClana(Clan): signal;

**Веза са СК:** СК3

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

**Постуслови:** Унети подаци о члану су промењени.

## 3.2.5 Уговор УГ5: ObrisiClana(Clan)

**Операција:** ObrisiClana(Clan): signal;

**Веза са СК:** СК4

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

**Постуслови:** Члан је обрисан.

## 3.2.6 Уговор УГ6: KreirajKnjigu(Knjiga)

**Операција:** KreirajKnjigu(Knjiga): signal;

**Веза са СК:** СК5

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

**Постуслови:** Креирана је нова књига.

## 3.2.7 Уговор УГ7: PretrazivanjeKnjiga(kriterijum, List<Knjiga>)

**Операција:** PretrazivanjeKnjiga(kriterijum*,* List<Knjiga>): signal;

**Веза са СК:** СК6, СК7, СК8

**Предуслови:** /

**Постуслови:** /

## 3.2.8 Уговор УГ8: PronadjiKnjigu(Knjiga, List<Knjiga>)

**Операција:** PronadjiKnjigu(Knjiga*,* List<Knjiga>): signal;

**Веза са СК:** СК6, СК7, СК8

**Предуслови:** /

**Постуслови:** /

## 3.2.9 Уговор УГ9: PromeniKnjigu(Knjiga)

**Операција:** PromeniKnjigu(Knjiga): signal;

**Веза са СК:** СК7

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

**Постуслови:** Унети подаци о књизи су промењени.

## 3.2.10 Уговор УГ10: ObrisiKnjigu(Knjiga)

**Операција:** ObrisiKnjigu(Knjiga): signal;

**Веза са СК:** СК8

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

**Постуслови:** Књига је обрисана.

## 3.2.11 Уговор УГ11: UcitajKnjige()

**Операција:** UcitajKnjige(): signal;

**Веза са СК:** СК9

**Предуслови:** /

**Постуслови:** /

## 3.2.12 Уговор УГ12: UcitajClanoveBiblioteke()

**Операција:** UcitajClanoveBiblioteke(): signal;

**Веза са СК:** СК9

**Предуслови:** /

**Постуслови:** /

## 3.2.13 Уговор УГ13: KreirajZaduzenje(Zaduzenje)

**Операција:** KreirajZaduzenje(Zaduzenje): signal;

**Веза са СК:** СК9

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

**Постуслови:** Креиранo је новo задужење.

## 3.2.14 Уговор УГ14: PretrazivanjeZaduzenja(kriterijum, List<Zaduzenje>)

**Операција:** PretrazivanjeZaduzenja(kriterijum*,* List<Zaduzenje>): signal;

**Веза са СК:** СК10

**Предуслови:** /

**Постуслови:** /

## 3.2.15 Уговор УГ15: PronadjiZaduzenje(Zaduzenje, List<Zaduzenje>)

**Операција:** PronadjiZaduzenje(Zaduzenje*,* List<Zaduzenje>): signal;

**Веза са СК:** СК10

**Предуслови:** /

**Постуслови:** /

## 3.2.16 Уговор УГ16: PromeniZaduzenje(Zaduzenje)

**Операција:** PromeniZaduzenje(Zaduzenje): signal;

**Веза са СК:** СК10

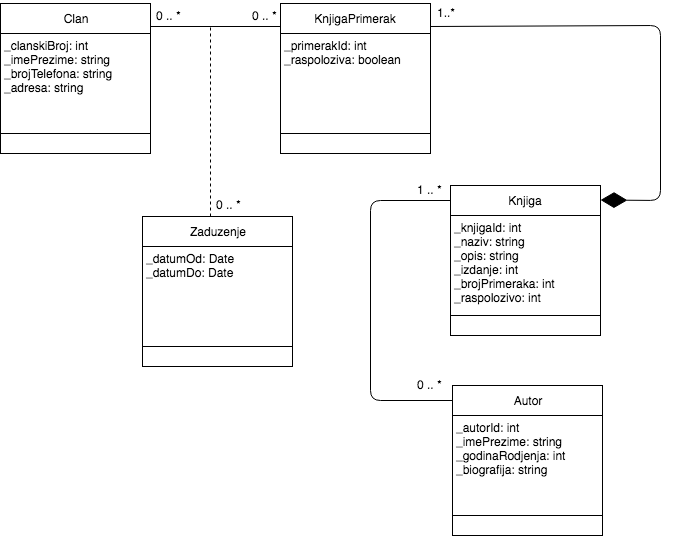
**Предуслови:** Вредносна и структурна ограничења над објектом *Zaduzenje* морају бити задовољена.

**Постуслови:** Унети подаци о задужењу су промењени.

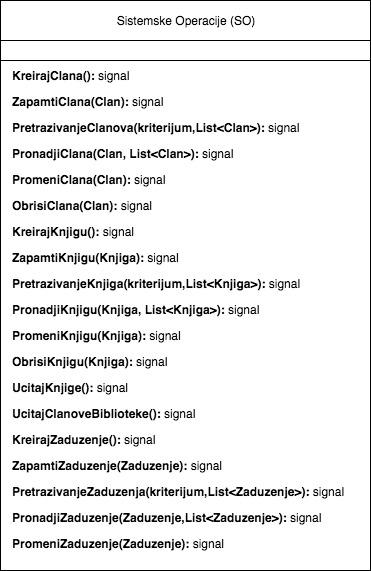
# 3.3 Структура софтверског система: Концептуални модел

Структура софтверског система се описује помоћу *концептуалног модела*. Концептуални модел садржи концептуалне класе (доменске објекте) и асоцијације између концептуалних класа. Често се још и називају *доменским моделима* или *моделима објектне анализе.*

Као резултат анализе сценарија СК и направљеног концептуалног модела добија се *логичка структура и понашање софтверског система.*



*Слика 37 - Концептуални дијаграм класа (Логичка структура софтверског система)*



*Слика 38 - Системске операције (Понашање софтверског система)*

# 3.4 Структура софтверског система: Релациони модел

На основу концептуалног модела, креирамо релациони. Релациони модел представља основу за пројектовање релационе базе података.

Идентификовали смо следеће класе: **Clan, Knjiga, KnjigaPrimerak, Autor** и **Zaduzenje**.

Релациони модел:

* **Clan**(ClanskiBroj, ImePrezime, BrojTelefona, Adresa)
* **Knjiga**(KnjigaID, Naziv, Opis, Izdanje, BrojPrimeraka, Raspolozivo)
* **KnjigaAutor**(KnjigaID, AutorID)
* **Autor**(AutorID, ImePrezime, GodinaRodjenja, Biografija)
* **KnjigaPrimerak**(PrimerakID, KnjigaID, Raspoloziva)
* **Zaduzenje**(ClanskiBroj, KnjigaID, PrimerakID, DatumOd, DatumDo)

## 3.4.1 Табеле ограничења

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Табела *Clan*** | | **Просто вредносно ограничење** | | **Сложено вредносно ограничење** | | **Структурно ограничење** |
| **Атрибути** | **Име** | **Тип** **атрибута** | **Вредност атрибута** | **Међузависност атрибута једне табеле** | **Међузависност атрибута више табела** | INSERT /  UPDATE CASCADES *Zaduzenje*  DELETE RESTRICTED *Zaduzenje* |
| ClanskiBroj | int | not null |  |  |
| ImePrezime | string |  |  |  |
| BrojTelefona | string |  |  |  |
| Adresa | string |  |  |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Табела *Knjiga*** | | **Просто вредносно ограничење** | | **Сложено вредносно ограничење** | | **Структурно ограничење** | |
| **Атрибути** | **Име** | **Тип** **атрибута** | **Вредност атрибута** | **Међузависност атрибута једне табеле** | **Међузависност атрибута више табела** | INSERT /  UPDATE CASCADES *KnjigaPrimerak, KnjigaAutor*  DELETE RESTRICTED *KnjigaPrimerak, KnjigaAutor* |
| KnjigaID | int | not null |  |  |
| Naziv | string | not null |  |  |
| Opis | string |  |  |  |
| Izdanje | int |  |  |  |
| BrojPrimeraka | int | (default:0) |  | BrojPrimeraka = COUNT(KnjigaPrimerak.PrimerakID) |
| Raspolozivo | int | Autor |  | Raspolozivo = COUNT(KnjigaPrimerak.Raspoloziva = true) |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Табела *KnjigaAutor*** | | **Просто вредносно ограничење** | | **Сложено вредносно ограничење** | | **Структурно ограничење** | |
| **Атрибути** | **Име** | **Тип** **атрибута** | **Вредност атрибута** | **Међузависност атрибута једне табеле** | **Међузависност атрибута више табела** | INSERT RESTRICTED *Knjiga, Autor*  UPDATE RESTRICTED *Knjiga, Autor*  DELETE / |
| KnjigaID | int | not null |  |  |
| AutorID | int | not null |  |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Табела *Autor*** | | **Просто вредносно ограничење** | | **Сложено вредносно ограничење** | | **Структурно ограничење** | |
| **Атрибути** | **Име** | **Тип** **атрибута** | **Вредност атрибута** | **Међузависност атрибута једне табеле** | **Међузависност атрибута више табела** | INSERT /  UPDATE CASCADES *KnjigaAutor*  DELETE CASCADES *KnjigaAutor* |
| AutorID | int | not null |  |  |
| ImePrezime | string |  |  |  |
| GodinaRodjenja | int |  |  |  |
| Biografija | string |  |  |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Табела *KnjigaPrimerak*** | | **Просто вредносно ограничење** | | **Сложено вредносно ограничење** | | **Структурно ограничење** | |
| **Атрибути** | **Име** | **Тип** **атрибута** | **Вредност атрибута** | **Међузависност атрибута једне табеле** | **Међузависност атрибута више табела** | INSERT RESTRICTED *Knjiga*  UPDATE CASCADES *Zaduzenje* RESTRICTED *Knjiga*  DELETE RESTRICTED *Zaduzenje* |
| PrimerakID | int | not null |  |  |
| KnjigaID | string | not null |  |  |
| Raspoloziva | string | (default: true) |  |  |

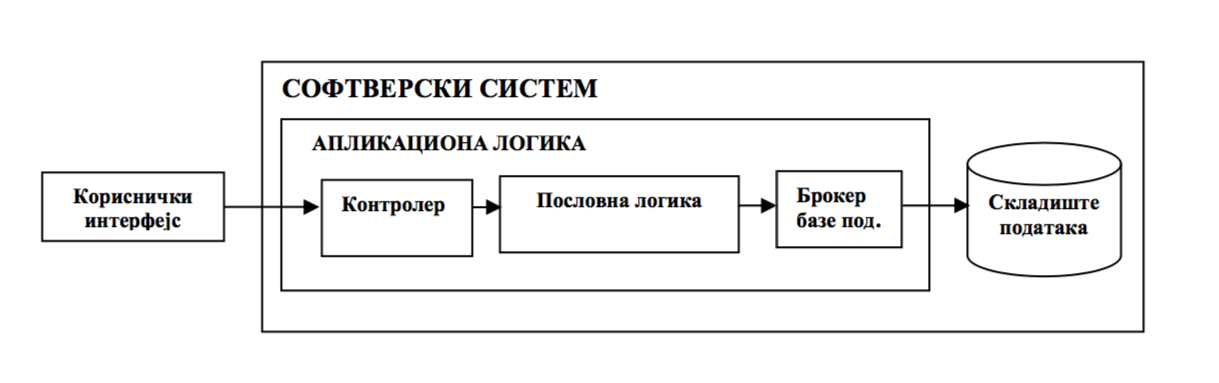
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Табела *Zaduzenje*** | | **Просто вредносно ограничење** | | **Сложено вредносно ограничење** | | **Структурно ограничење** | |
| **Атрибути** | **Име** | **Тип** **атрибута** | **Вредност атрибута** | **Међузависност атрибута једне табеле** | **Међузависност атрибута више табела** | INSERT RESTRICTED *Clan, KnjigaPrimerak*  UPDATE RESTRICTED *Clan, KnjigaPrimerak*  DELETE / |
| ClanskiBroj | int | not null |  |  |
| KnjigaID | int | not null |  |  |
| PrimerakID | int | not null |  |  |
| DatumOd | Date | not null |  |  |
| DatumDo | Date |  |  |  |

4. Пројектовање

Фаза пројектовања описује физичку структуру и понашање софтвeрског система (архитектуру софтверског система). Пројектовање архитектуре софтверског система обухвата пројектовање корисничког интерфејса, апликационе логике и складишта података. Пројектовање **корисничког** **интерфејса** обухвата пројектовање екранских форми и контролера корисничког интерфејса. У оквиру **апликационе** **логике** се пројектују контролер апликационе логике, пословна логика и брокер базе података. Пројектовање **пословне** **логика** обухвата пројектовање логичке структуре и понашања софтверског система.

Тронивојска апхитектура софтверског система се састоји од:

1. Корисничког интерфејса – улазно-излазна репрезентација софтверског система
2. Апликационе логике – описује стуктуру и понашање софтверског система
3. Складишта података – чува стање атрибута софтверског система

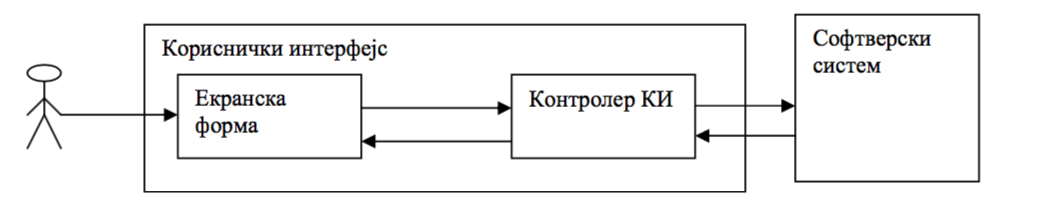


Слика 38 - Тронивојска архитектура

# 4.1 Пројектовање корисничког интерфејса

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

1. Екранске форме
2. Контролера корисничког интерфејса

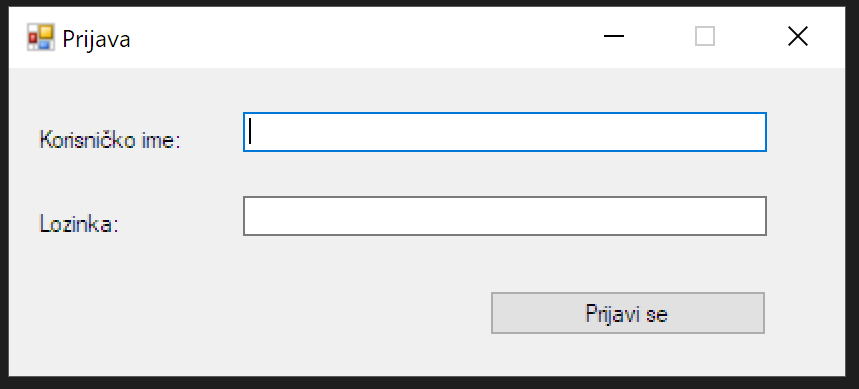


Слика 39 - Пројектовање корисничког интерфејса

## 4.1.1 Пројектовање екранске форме

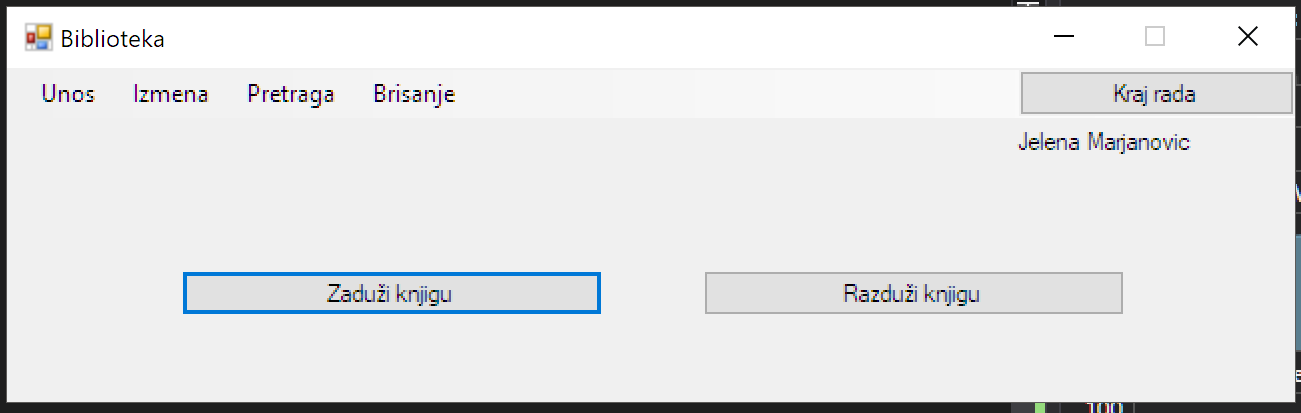
Постоје два аспекта пројектовања екранске форме: пројектовање сценарија СК који се изводе преко екранске форме и пројектовање метода екранске форме.

Након покретања система, библиотекару се отвара форма за пријаву на систем.



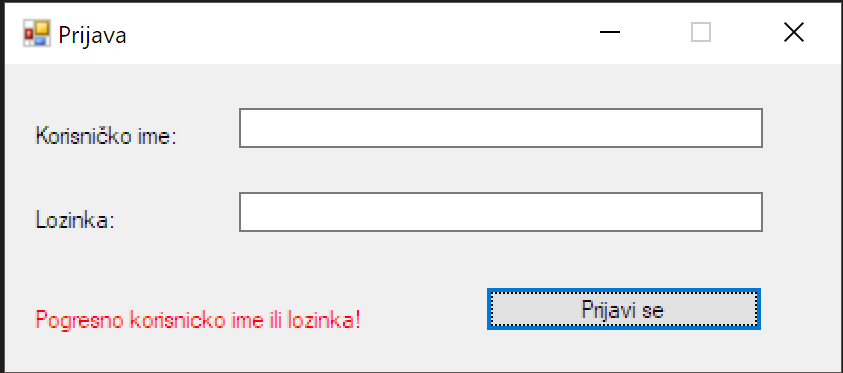
Слика 40 - Пријава на систем: Непопуњена форма

Уколико се успешно пријави, отвара му се главна форма, са које бира жељену операцију.

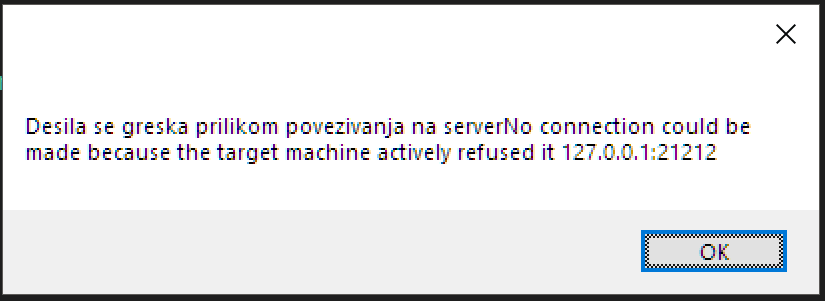
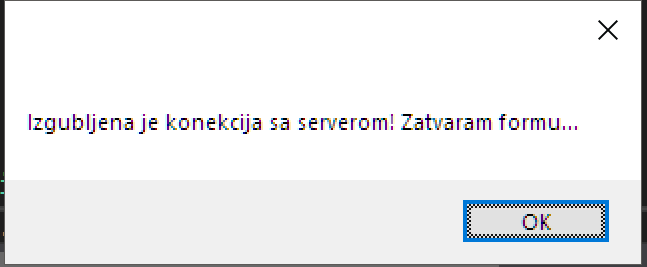


Слика 41 - Главна форма

Уколико дође до грешке при пријави, појавиће му се порука о грешци, а уколико је немогуће пријавити се због конекције са сервером, неће бити могуће коришћење система и апликација ће се угасити.



Слика 42 - Грешка: Погрешно корисничко име или лозинка

Слика 43 - Грешка: Није могуће повезати се на сервер Слика 44 - Грешка: Дисконектован са сервера

### Пројектовање сценарија СК

**СК1 – Унос новог члана библиотеке**

**Назив СК**

Креирање члана

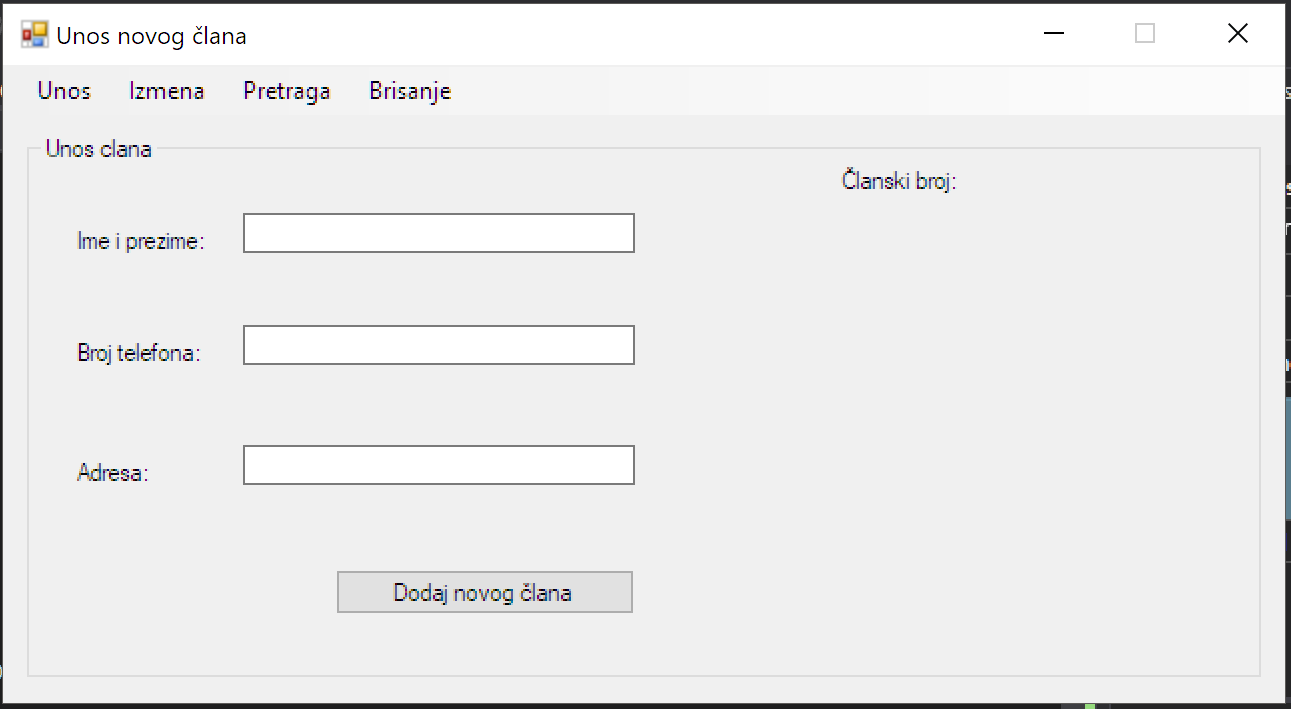
**Актори СК**

Библиотекар

**Учесници СК**

Библиотекар и систем (програм)

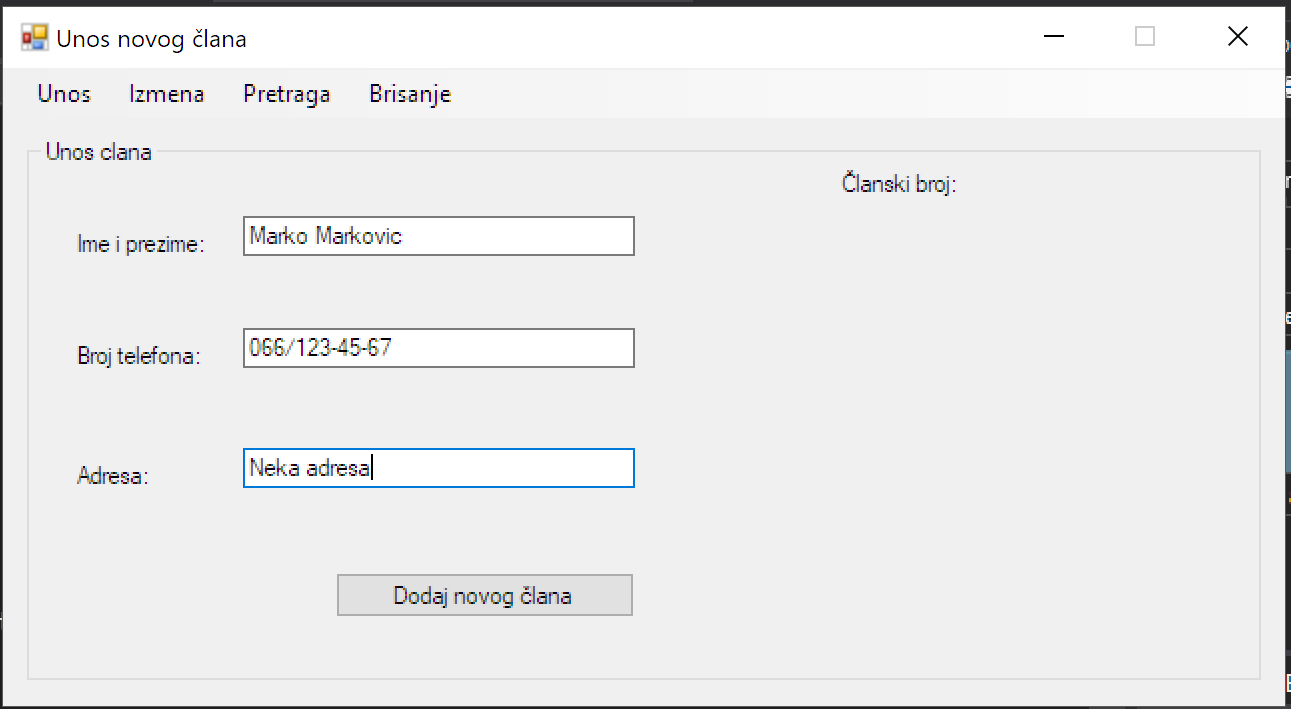
**Предуслов**: Систем је укључен и библиотекар je улогован под својом шифром. Систем приказује форму за рад са чланом.



Слика 4 - Унос новог члана: Непопуњена форма

**Основни сценарио СК**

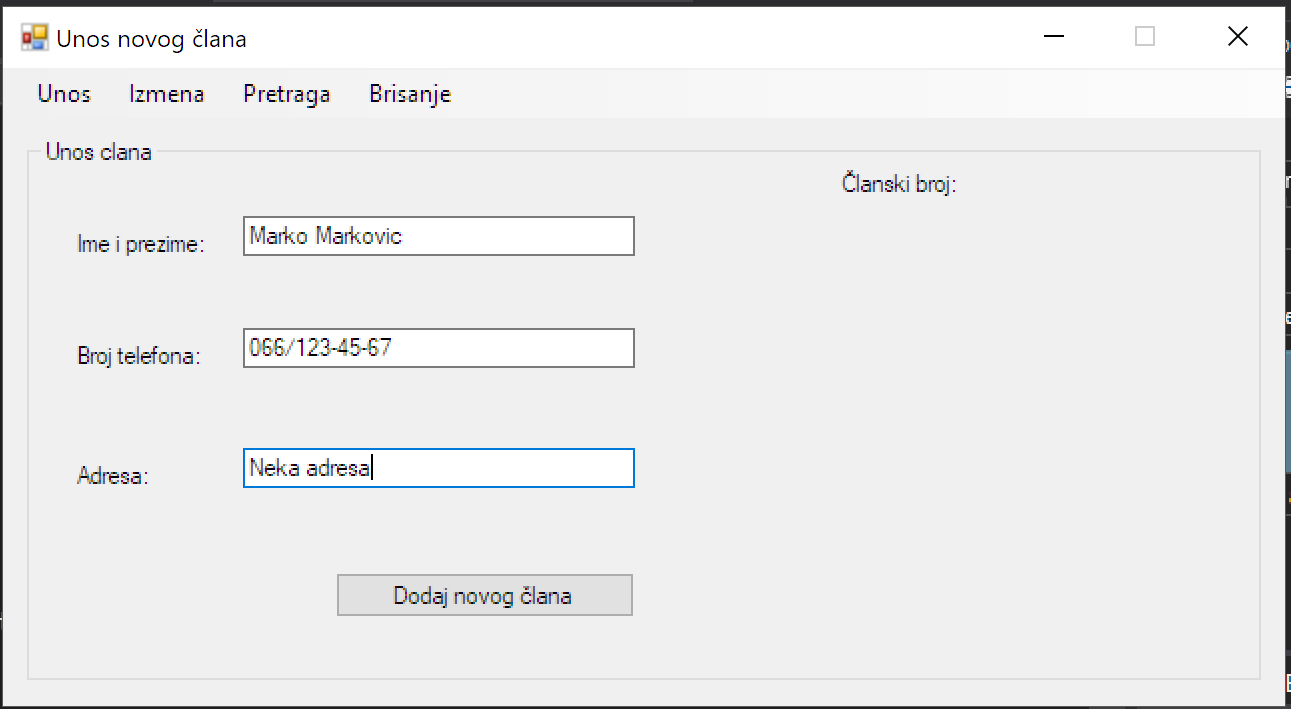
1. Библиотекар **уноси** податке о члану. (АПУСО)



Слика 4 - Унос новог члана: Попуњена форма

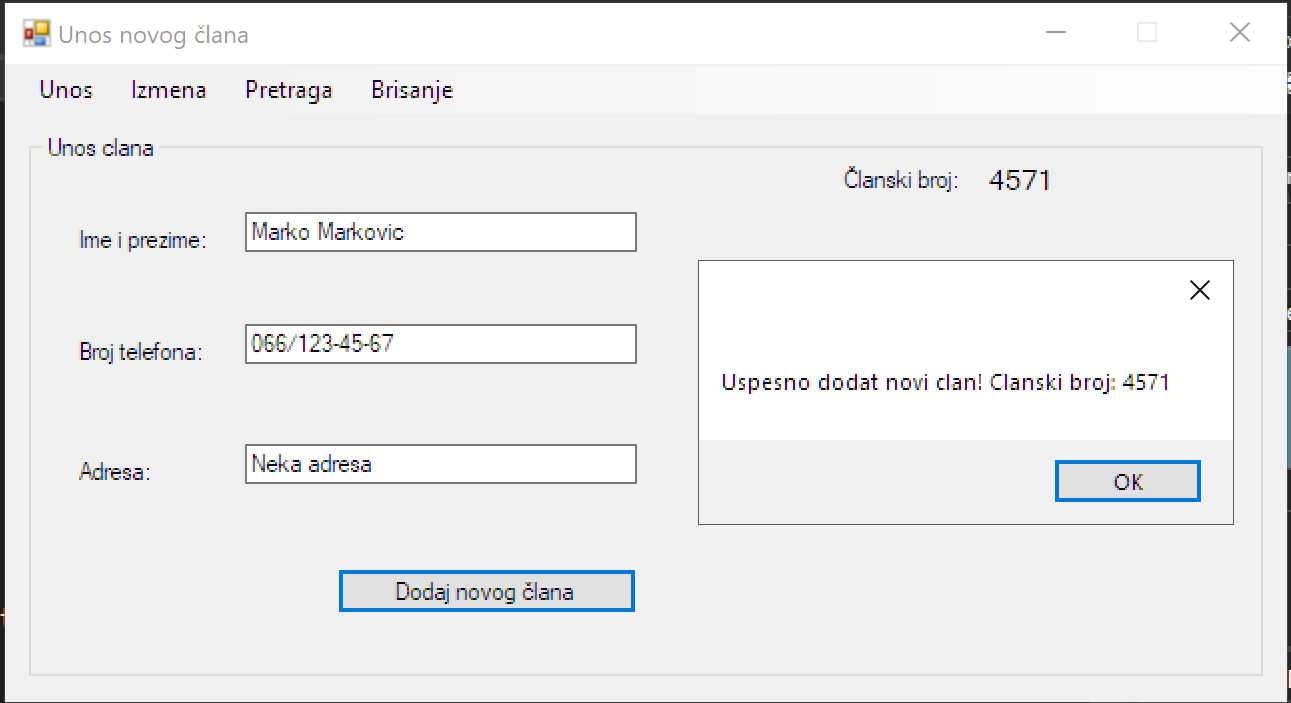
1. Библиотекар **контролише** да ли је коректно унео податке о члану. (АНСО)
2. Библиотекар **позива** систем да креира и запамти податке о члану. (АПСО)

***Опис акције:*** *Библиотекар кликом на дугме ”Додај новог члана” позива системску операцију* ***UbaciClanaSO*** *која креира и памти новог члана библиотеке.*



Слика 4 - Унос новог члана: Позивање система да изврши операцију

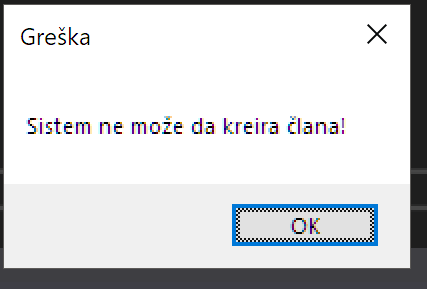
1. Систем **памти** податке о члану. (СО)
2. Систем **приказује** библиотекару запамћеног члана и поруку: “Систем је креирао члана“. (ИА)



Слика 4 - Унос новог члана: Успешно

**Алтернативна сценарија**

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



Слика 4 - Грешка: Систем не може да креира члана

**СК2 – Претраживање чланова библиотеке**

**Назив СК**

Претраживање члана

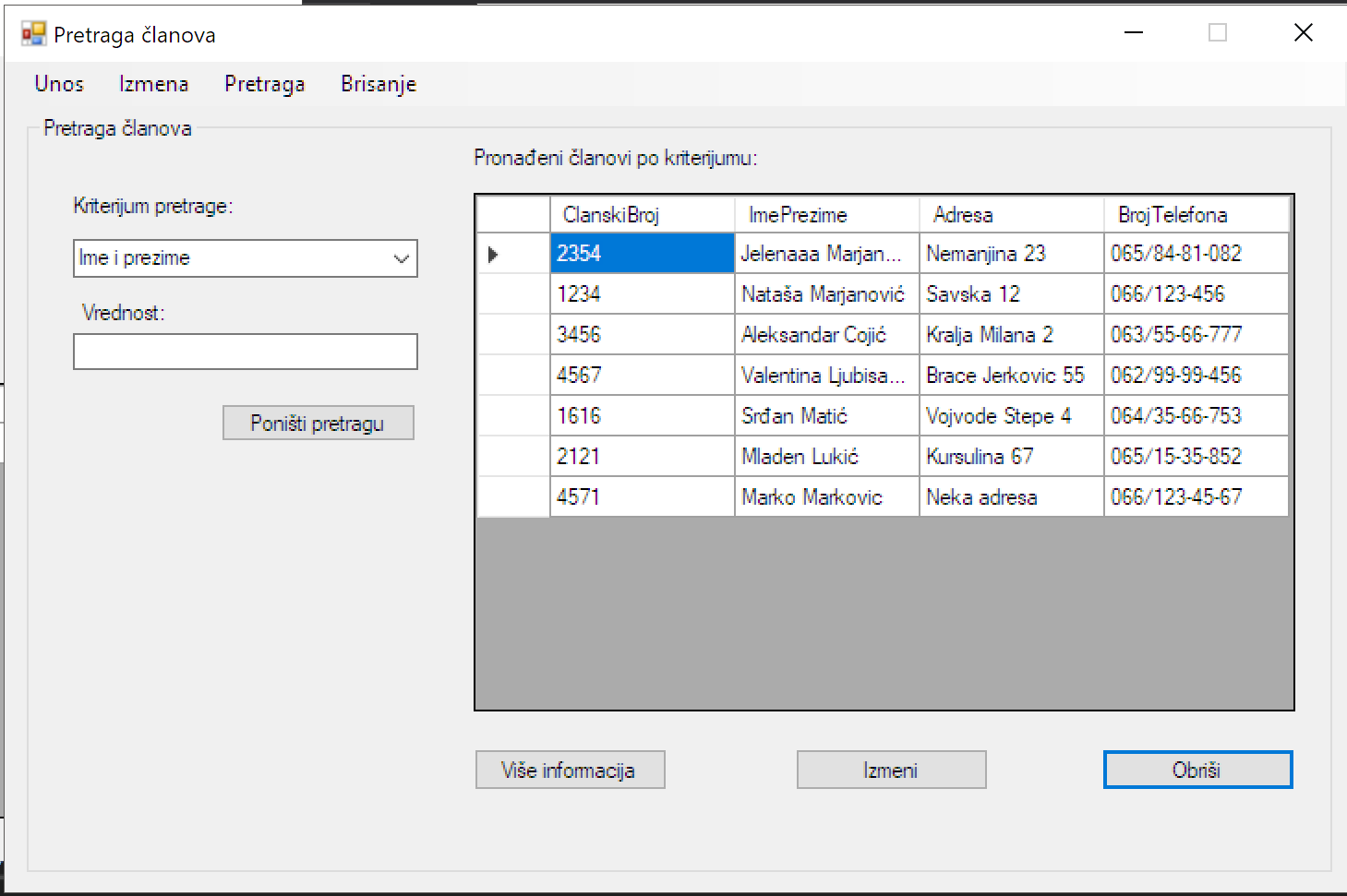
**Актори СК**

Библиотекар

**Учесници СК**

Библиотекар и систем (програм)

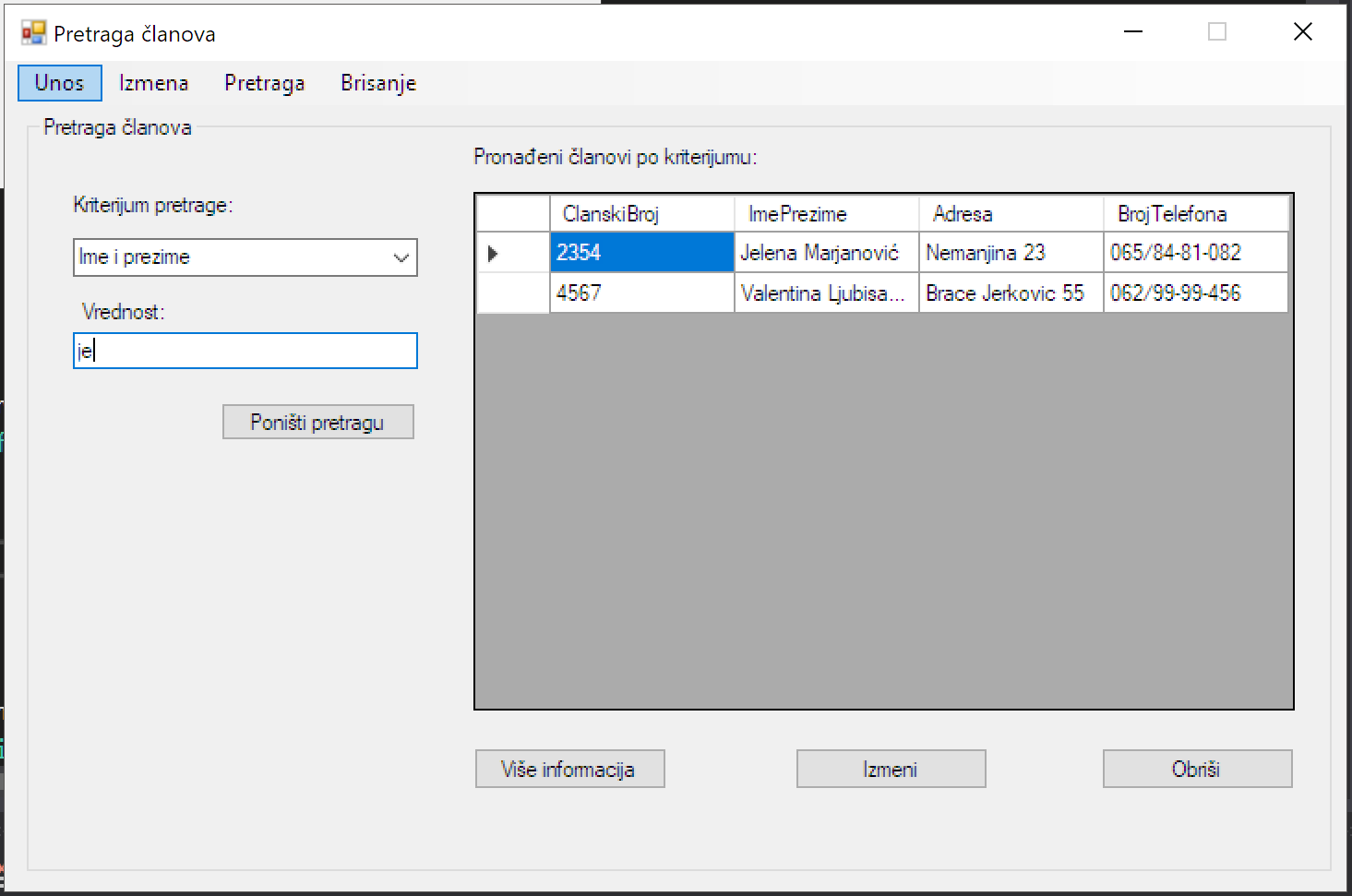
**Предуслов**: Систем је укључен и библиотекар je улогован под својом шифром. Систем приказује форму за рад са чланом.



Слика 46 - Претрага чланова: Непопуњена форма

**Основни сценарио СК**

1. Библиотекар **уноси** вредностпо којој претражује чланове. (АПУСО)



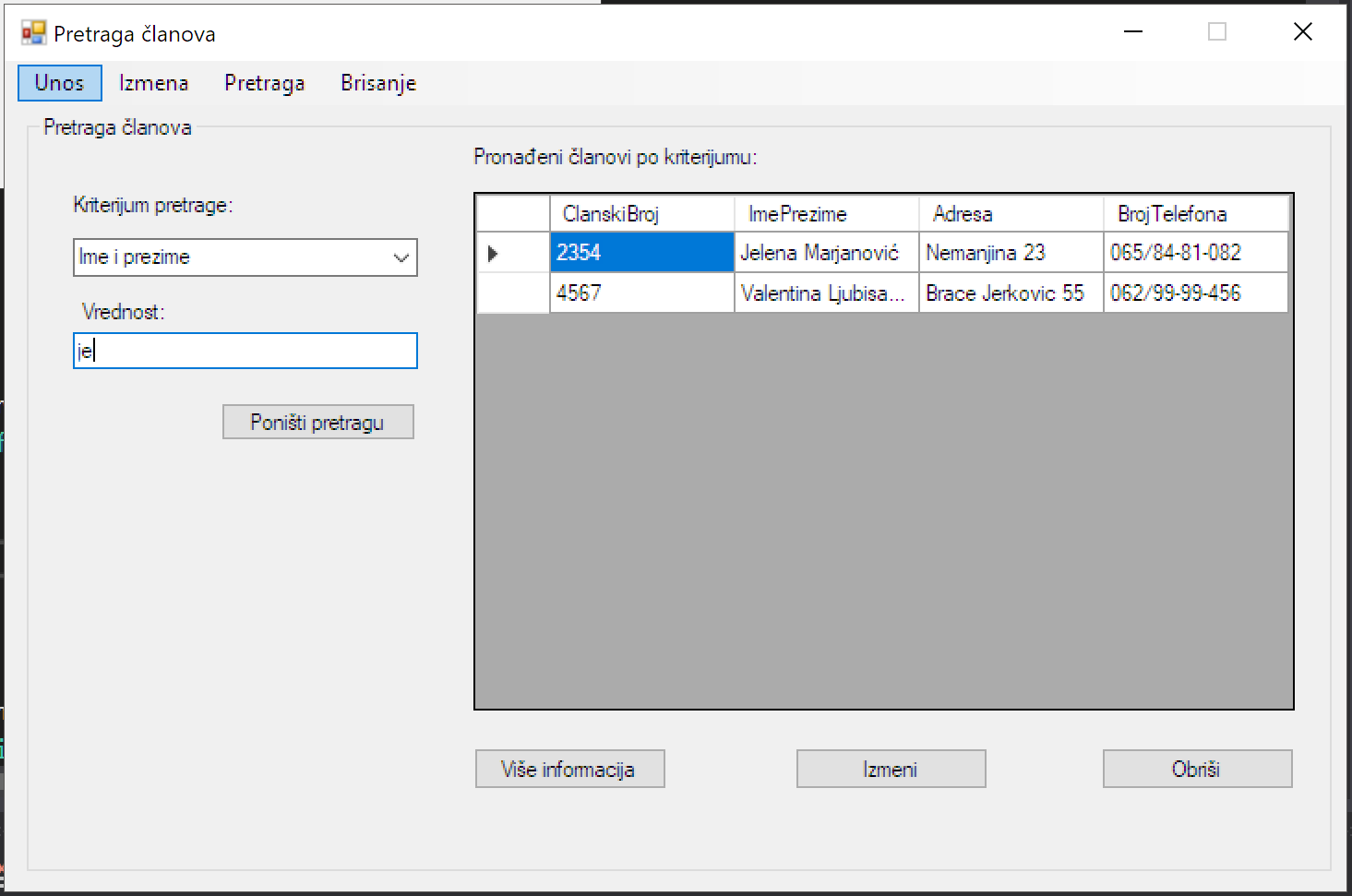
Слика 47 - Претрага чланова: Попуњена форма

1. Библиотекар **позива** систем да нађе чланове по задатој вредности. (АПСО)

***Опис акције:*** *Библиотекар уношењем вредности у поље које је за то предвиђено, позива системску операцију* ***PretraziClanoveSO,*** *која враћа све чланове који задовољавају услов.*

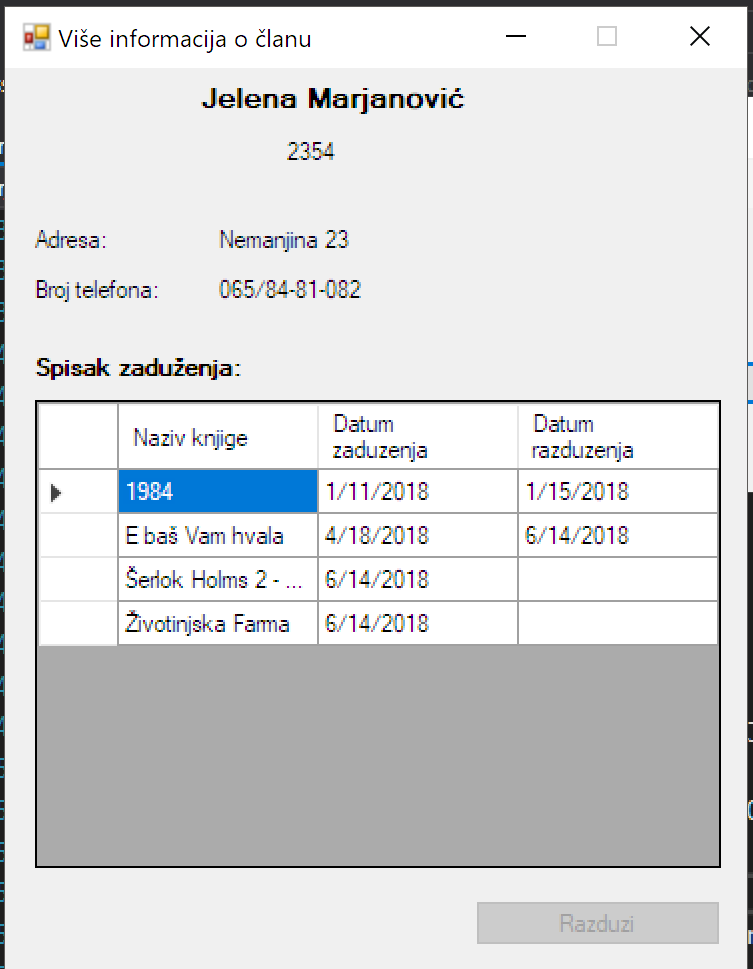
1. Систем **тражи** чланове по задатој вредности. (СО)
2. Систем приказује библиотекару податке о члановима и поруку: “Систем је нашао чланове по задатој вредности”. (ИА)
3. Библиотекар **бира** члана. (АПУСО)
4. Библиотекар **позива** систем да нађе податке о одабраном члану(АПСО)

***Опис акције:*** *Библиотекар кликом на дугме ”Бише информација” позива системску операцију* ***NadjiClanaSO*** *која налази све информације о одабраном члану.*



Слика 48 - Претрага чланова: Позивање система да нађе одређеног члана

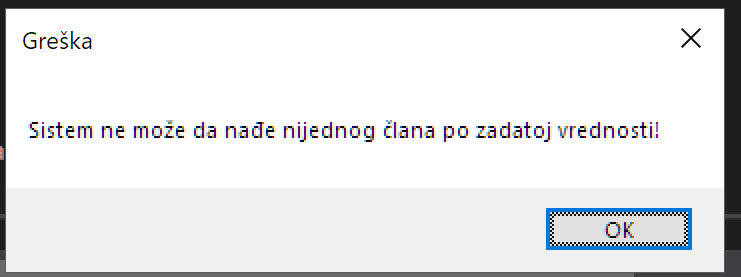
1. Систем **тражи** члана по задатој вредности. (СО)
2. Систем приказује библиотекару податке о члану и поруку: “Систем је нашао члана по задатој вредности”. (ИА)



Слика 49 - Претрага чланова: Више информација о члану

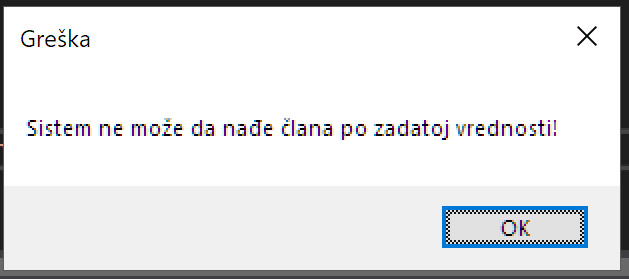
**Алтернативна сценарија**

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



Слика 50 - Грешка: Систем не може да нађе ниједног члана

8.1 Уколико систем не може да нађе члана он приказује библиотекару поруку: “Систем не може да нађе члана по задатој вредности”. (ИА)



Слика 51 - Грешка: Систем не може да нађе члана

**СК3 – Измена члана библиотеке**

**Назив СК**

Промена члана

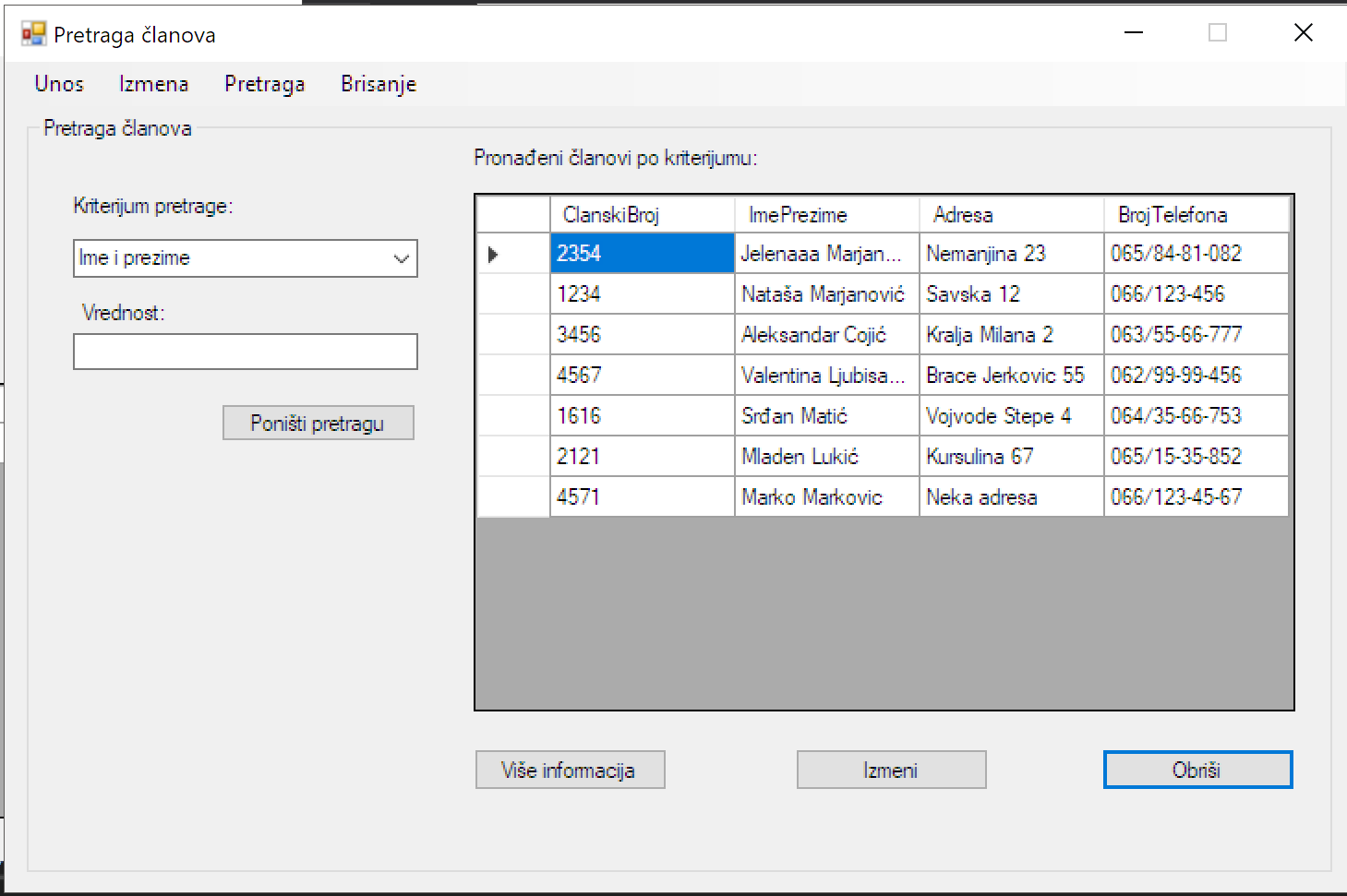
**Актори СК**

Библиотекар

**Учесници СК**

Библиотекар и систем (програм)

**Предуслов**: Систем је укључен и библиотекар je улогован под својом шифром. Систем приказује форму за рад са чланом.



Слика 52 - Форма за рад са чланом

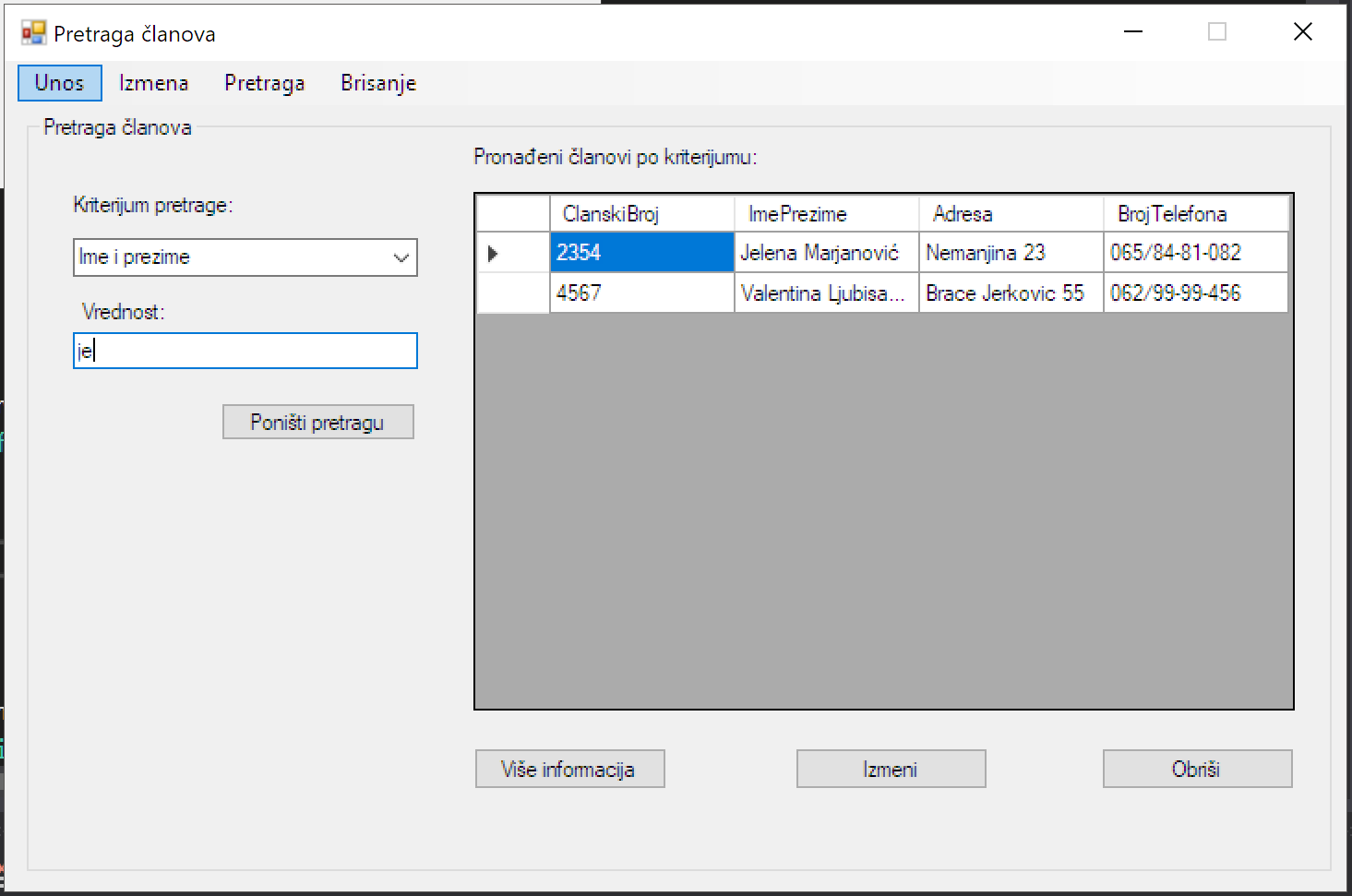
**Основни сценарио СК**

1. Библиотекар **уноси** вредностпо којој претражује чланове. (АПУСО)
2. Библиотекар **позива** систем да нађе чланове по задатој вредности. (АПСО)

***Опис акције:*** *Библиотекар уношењем вредности у поље које је за то предвиђено, позива системску операцију* ***PretraziClanoveSO****, која враћа све чланове који задовољавају услов.*

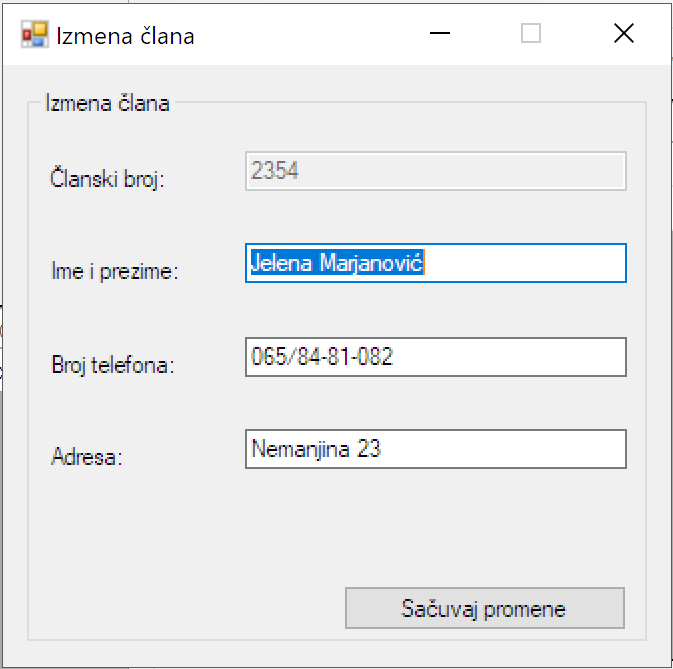
1. Систем **тражи** чланове по задатој вредности. (СО)
2. Систем приказује библиотекару чланове и поруку: “Систем је нашао чланове по задатој вредности”. (ИА)
3. Библиотекар **бира** члана. (АПУСО)
4. Библиотекар **позива** систем да нађе податке о одабраном члану(АПСО)

***Опис акције:*** *Библиотекар кликом на дугме ”Измени” позива системску операцију* ***NadjiClanaSO*** *која налази све информације о одабраном члану.*



Слика 53 - Измена одабраног члана

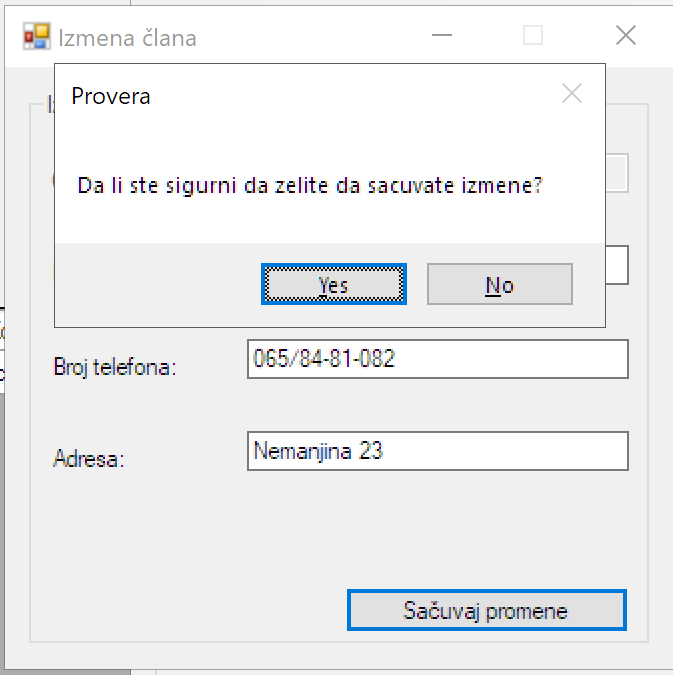
1. Систем **тражи** члана по задатој вредности. (СО)
2. Систем приказује библиотекару податке о члану и поруку: “Систем је нашао члана по задатој вредности”. (ИА)



Слика 54 - Измена члана: Информације о члану

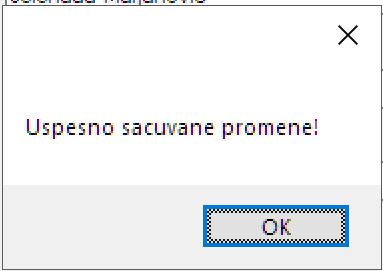
1. Библиотекар **уноси** **(мења)** податке о члану. (АПУСО)
2. Библиотекар **контролише** да ли је коректно унео податке о члану. (АНСО)
3. Библиотекар **позива** систем да промени податке о члану. (АПСО)

***Опис акције:*** *Библиотекар кликом на дугме ”Сачувај промене”, и затим ”Да” код провере, позива системску операцију* ***SacuvajIzmeneClanSO,*** *која мења информације о одабраном члану.*



Слика 55 - Измена члана: Позив система да изврши системску операцију

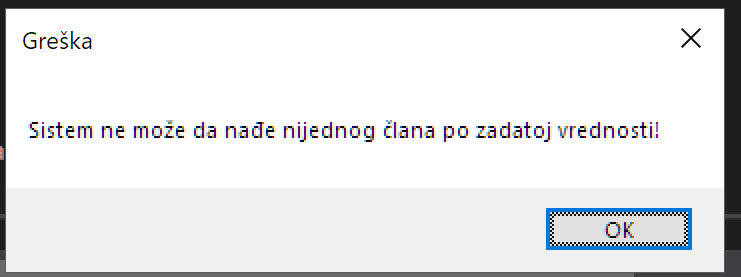
1. Систем **мења** податке о члану. (СО)
2. Систем **приказује** библиотекару промењеног члана и поруку: “Систем је успешно променио члана.” (ИА)



Слика 56 - Измена члана: Успешно сачувано

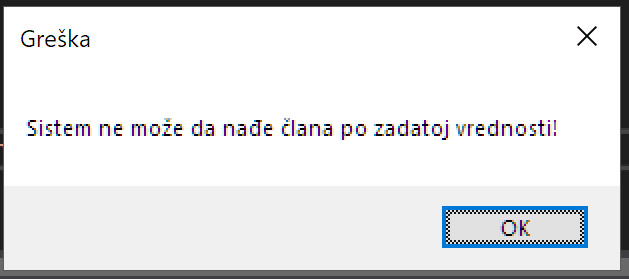
**Алтернативна сценарија**

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



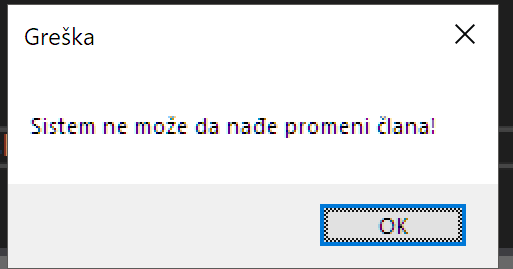
Слика 57 - Грешка: Систем не може да нађе ниједног члана

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



Слика 58 - Грешка: Систем не може да нађе члана

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



Слика 59 - Грешка: Систем не може да промени члана

**СК4 – Брисање члана библиотеке**

**Назив СК**

Брисање члана

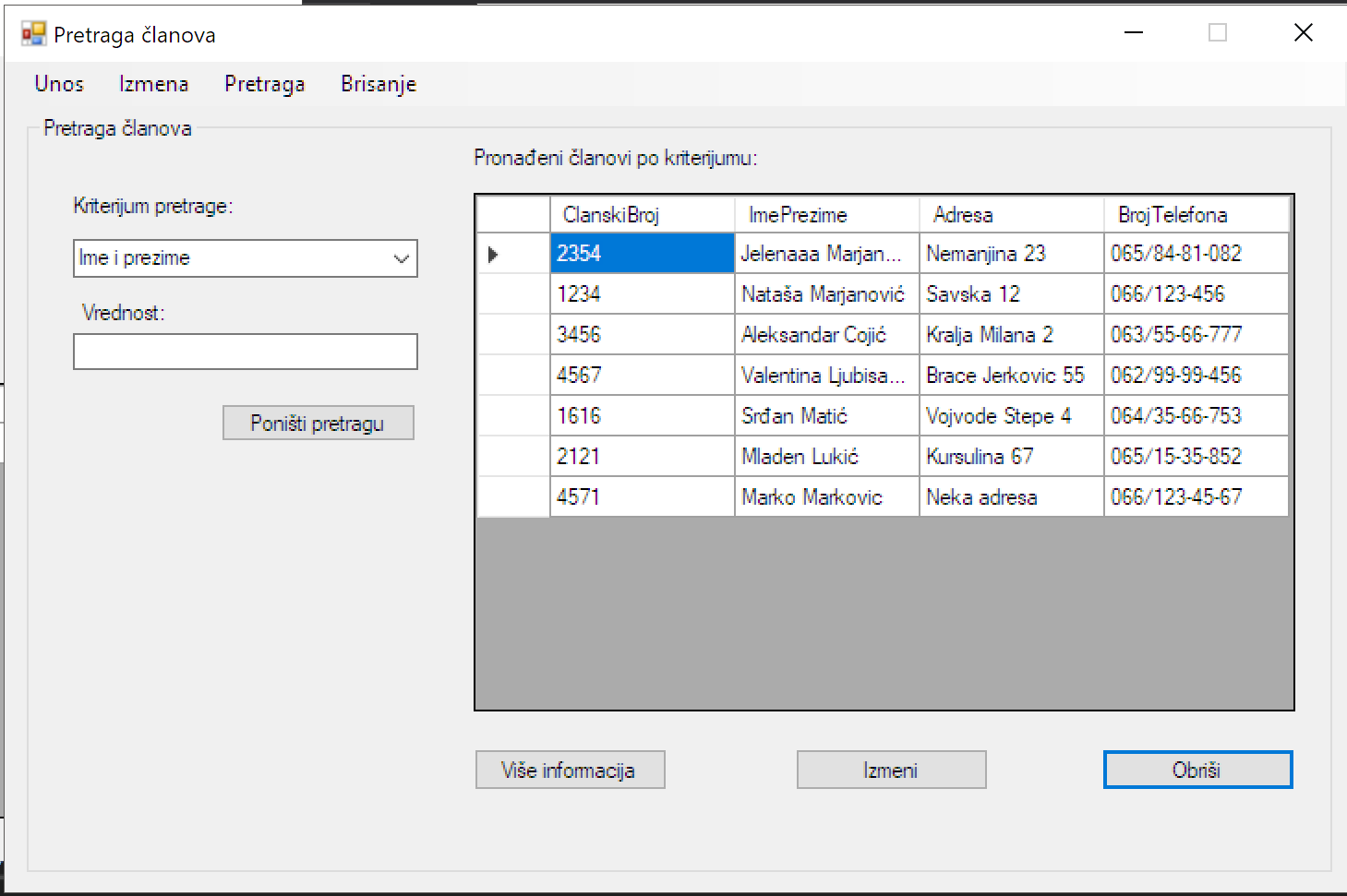
**Актори СК**

Библиотекар

**Учесници СК**

Библиотекар и систем (програм)

**Предуслов**: Систем је укључен и библиотекар je улогован под својом шифром. Систем приказује форму за рад са чланом.



Слика 60 - Форма за рад са чланом

**Основни сценарио СК**

1. Библиотекар **уноси** вредностпо којој претражује чланове. (АПУСО)

2. Библиотекар **позива** систем да нађе чланове по задатој вредности. (АПСО)

***Опис акције:*** *Библиотекар уношењем вредности у поље које је за то предвиђено, позива системску операцију* ***PretraziClanoveSO****, која враћа све чланове који задовољавају услов.*

3. Систем **тражи** чланове по задатој вредности. (СО)

4. Систем приказује библиотекару чланове и поруку: “Систем је нашао чланове по задатој вредности”. (ИА)

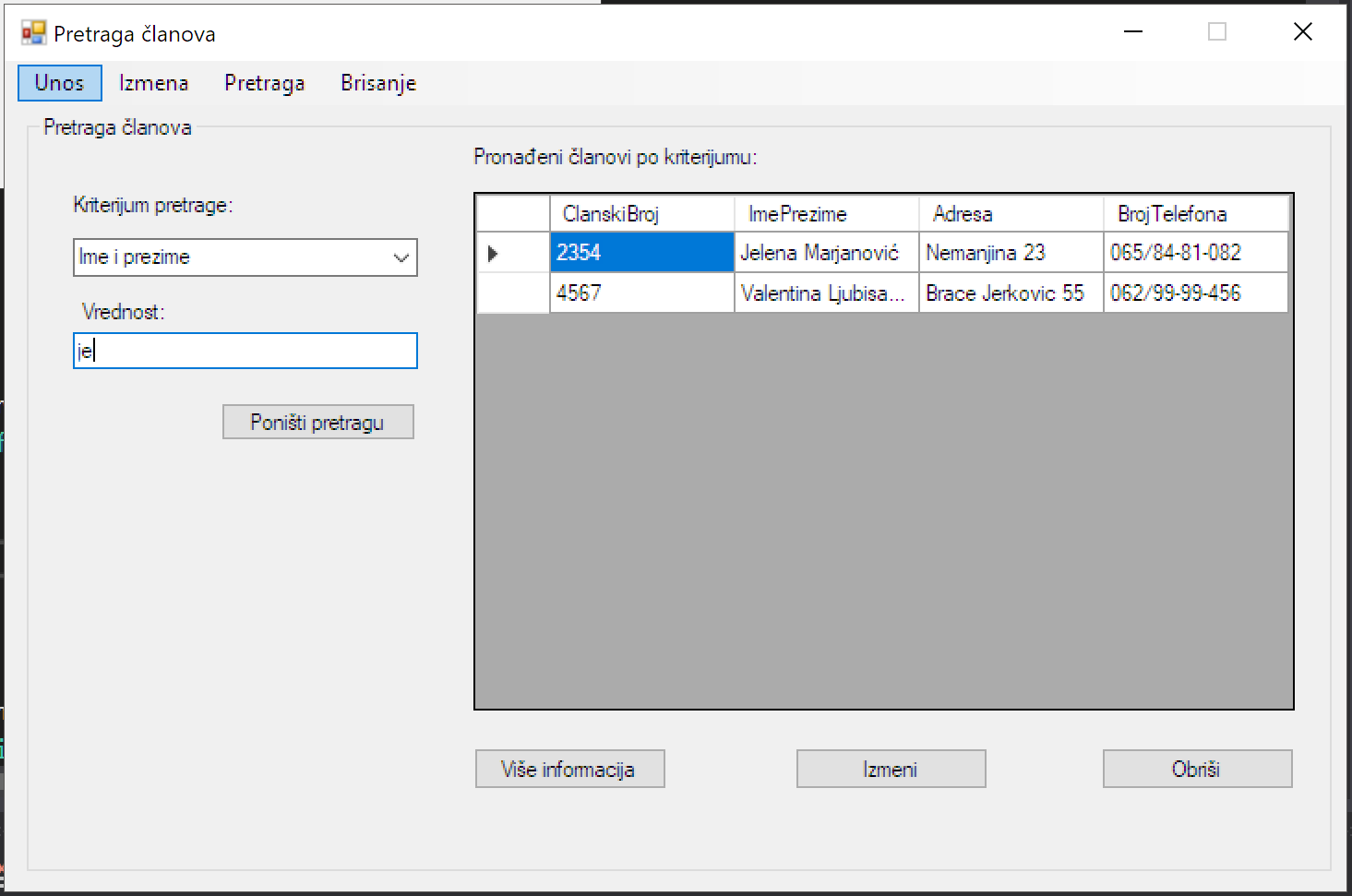
5. Библиотекар **бира** члана. (АПУСО)

6. Библиотекар **позива** систем да нађе податке о одабраном члану(АПСО)

***Опис акције:*** *Библиотекар кликом на дугме ”Измени” позива системску операцију* ***NadjiClanaSO*** *која налази све информације о одабраном члану.*

7. Систем **тражи** члана по задатој вредности. (СО)

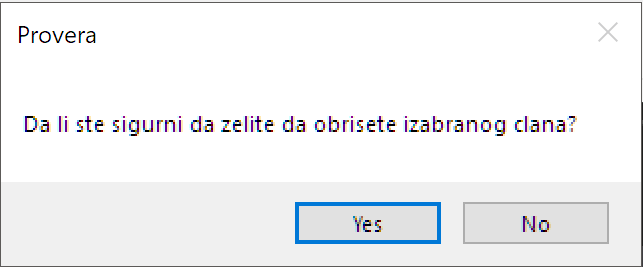
8. Систем приказује библиотекару податке о члану и поруку: “Систем је нашао члана по задатој вредности”. (ИА)



Слика 61 - Брисање члана: Одабран члан

9. Библиотекар **позива** систем да обрише члана. (АПСО)

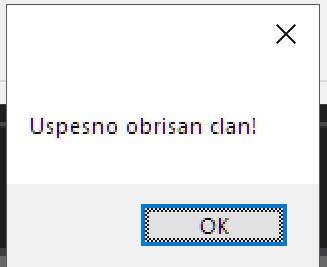
***Опис акције:*** *Библиотекар кликом на дугме ”Обриши”, а затим ”Да” код провере, позива системску операцију* ***ObrisiClanaSO*** *која брише одабраног члана.*



Слика 62 - Брисање члана: Провера

10. Систем **брише** члана. (СО)

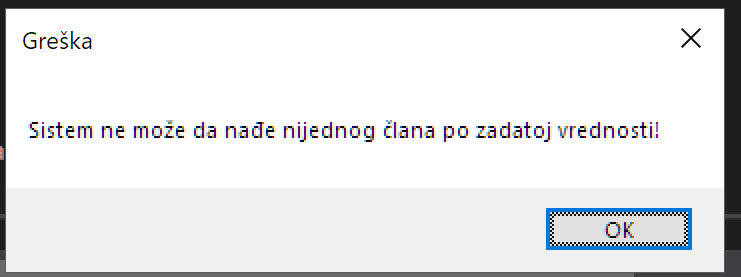
11. Систем **приказује** библиотекару поруку: “Систем је обрисао члана.” (ИА)



Слика 63 - Брисање члана: Успешно обрисан члан

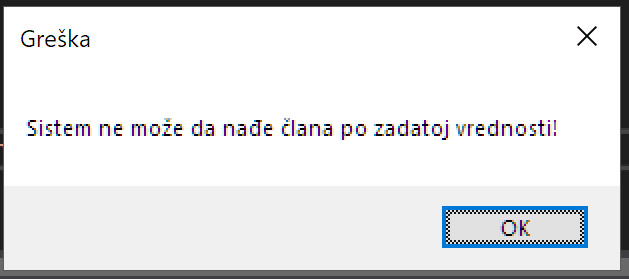
**Алтернативна сценарија**

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



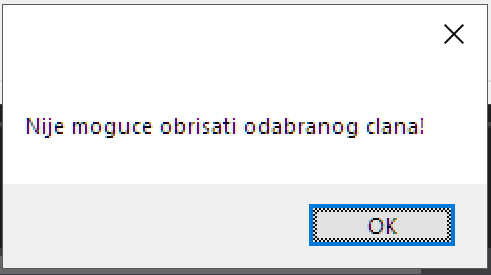
Слика 64 - Грешка: Систем не може да пронађе ниједног члана

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



Слика 65 - Грешка: Систем не може да пронађе члана

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



Слика 66 – Грешка: Систем не може да обрише члана

**СК5 – Унос нове књиге**

**Назив СК**

Креирање књиге

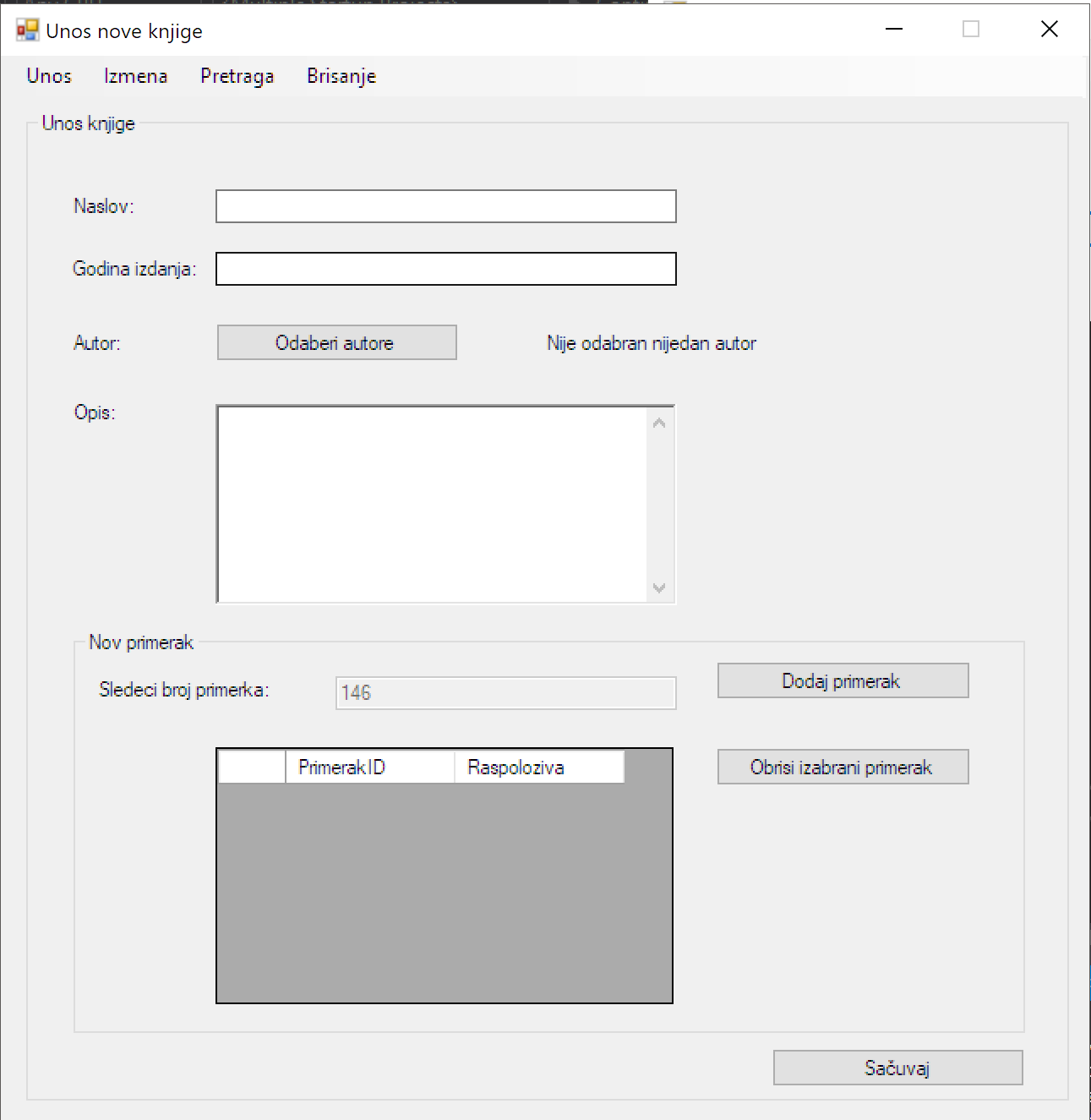
**Актори СК**

Библиотекар

**Учесници СК**

Библиотекар и систем (програм)

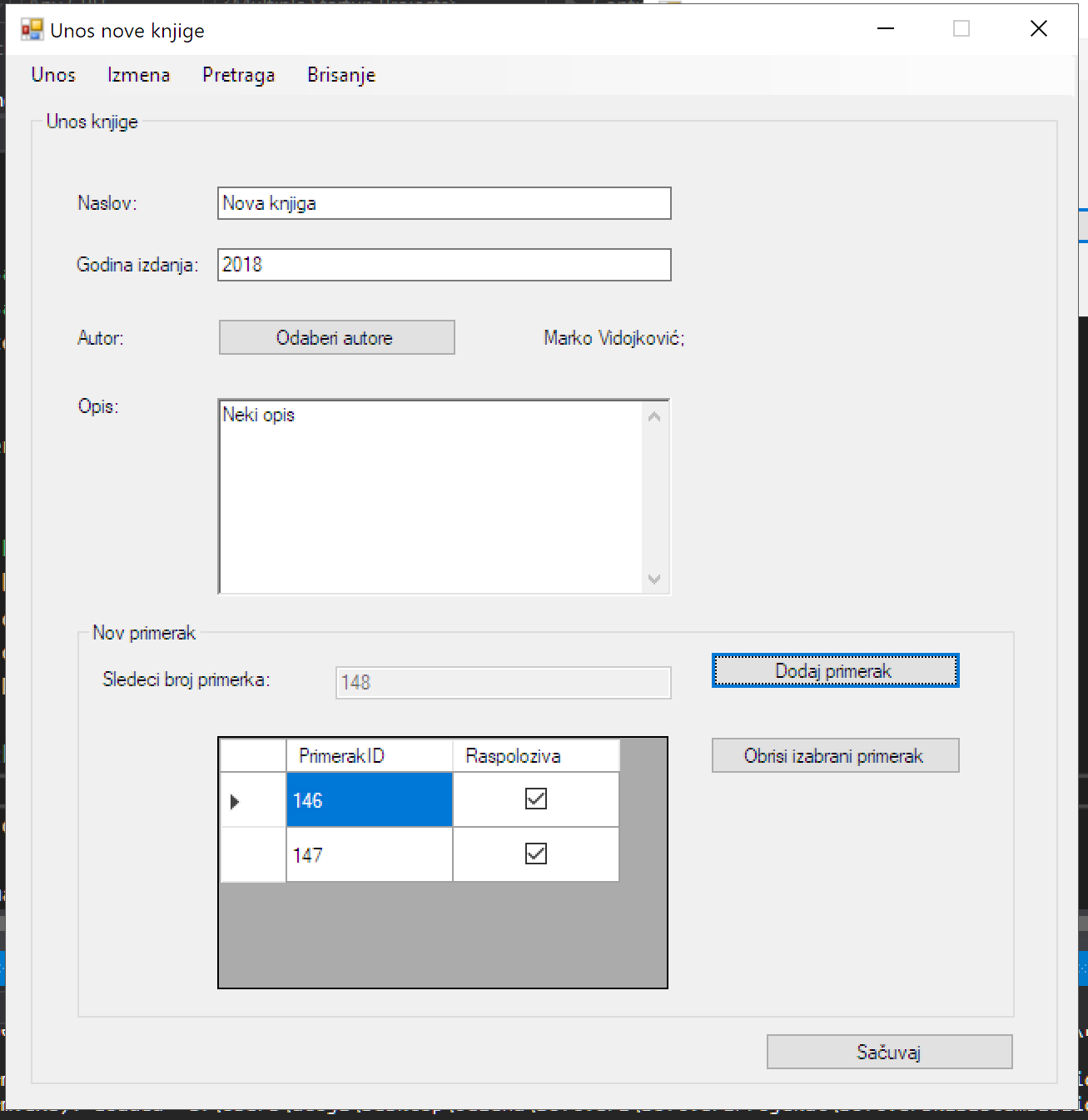
**Предуслов**: Систем је укључен и библиотекар je улогован под својом шифром. Систем приказује форму за рад са књигом.



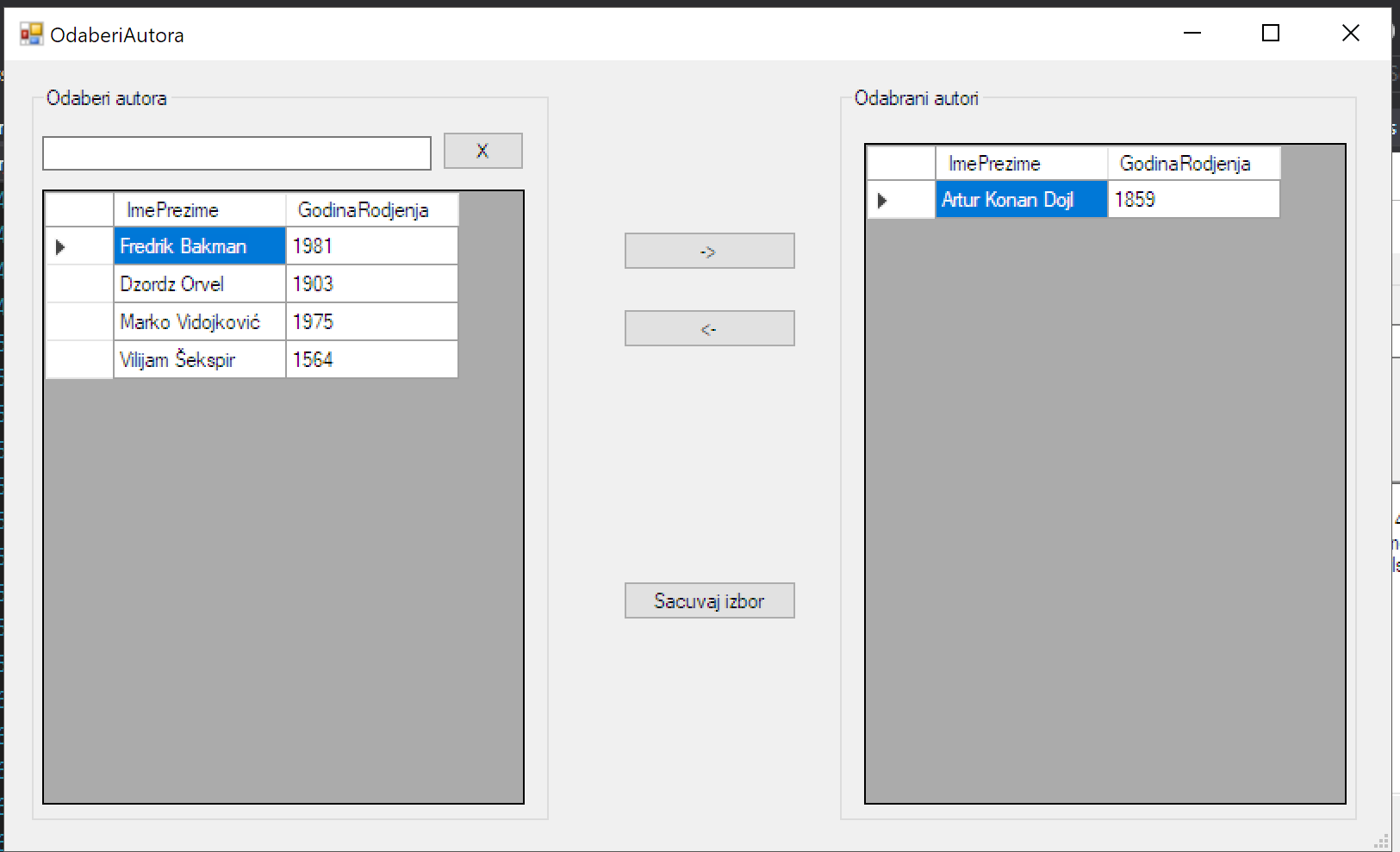
Слика 67 - Унос књиге: Непопуњена форма

**Основни сценарио СК**

1. Библиотекар **уноси** податке о књизи. (АПУСО)



Слика 68 - Унос књиге: Попуњена форма

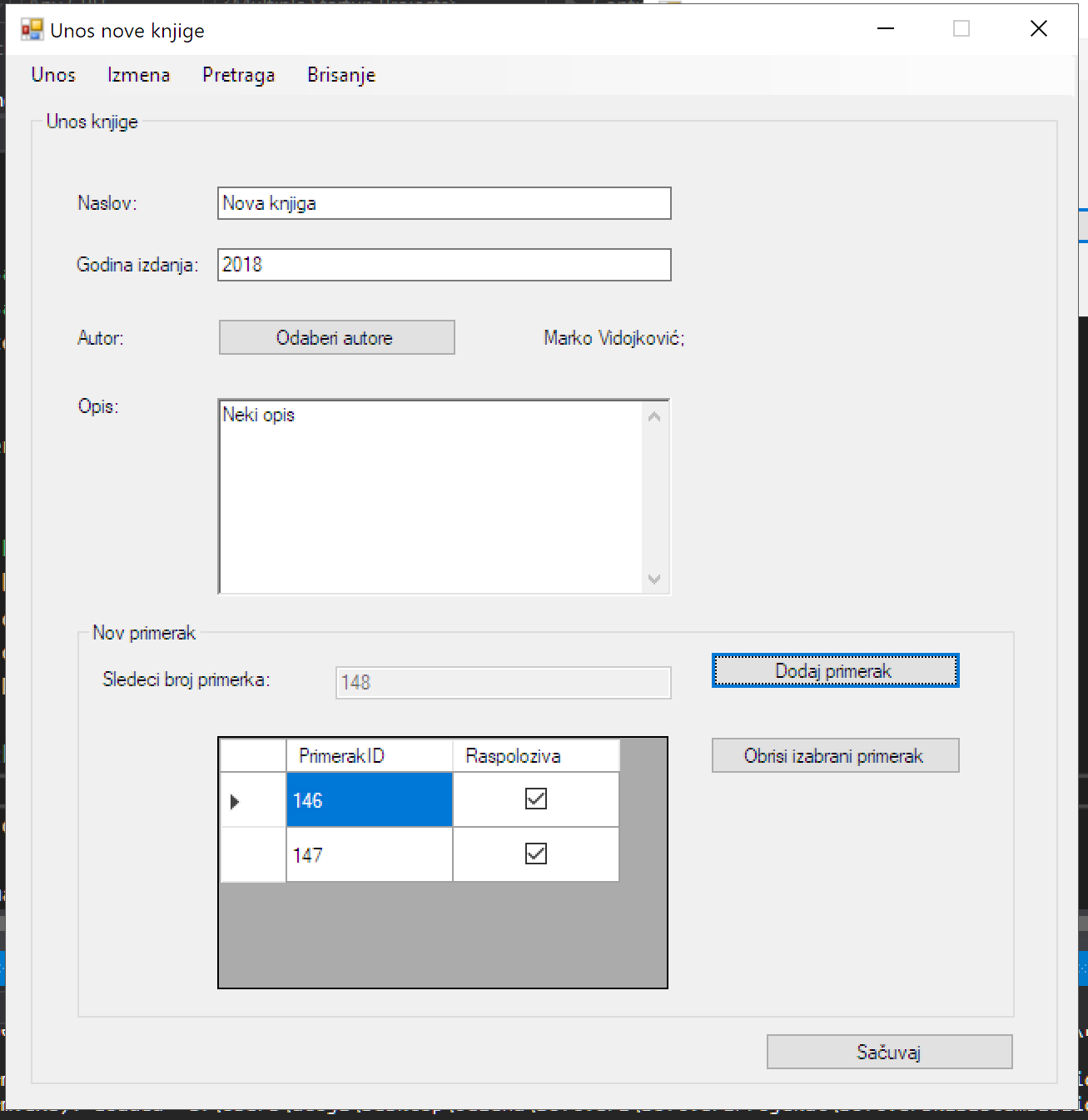
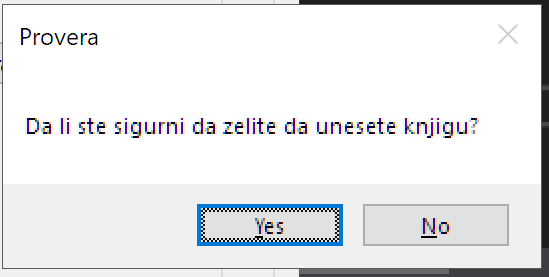


Слика 69 - Унос књиге: Бирање аутора

2. Библиотекар **контролише** да ли је коректно унео податке о књизи. (АНСО)

3. Библиотекар **позива** систем да креира и запамти податке о књизи. (АПСО)

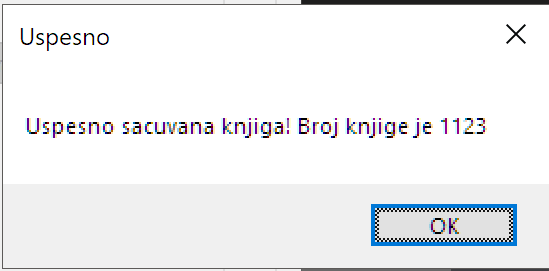
***Опис акције:*** *Библиотекар кликом на дугме ”Сачувај”, а затим ”Да”, позива системску операцију* ***UbaciKnjiguSO,*** *која налази креира и памти нову књигу.*



Слика 70 - Унос књиге: Провера

4. Систем **памти** податке о књизи. (СО)

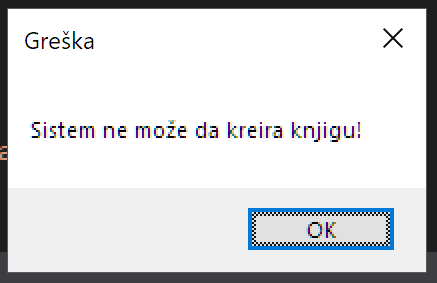
5. Систем **приказује** библиотекару запамћену књигу и поруку: “Систем је креирао књигу“. (ИА)



Слика 71 - Унос књиге: Успешно креирана књига

**Алтернативна сценарија**

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



Слика 72 - Грешка: Систем не може да креира књигу

**СК6 – Претраживање књига**

**Назив СК**

Претраживање књига

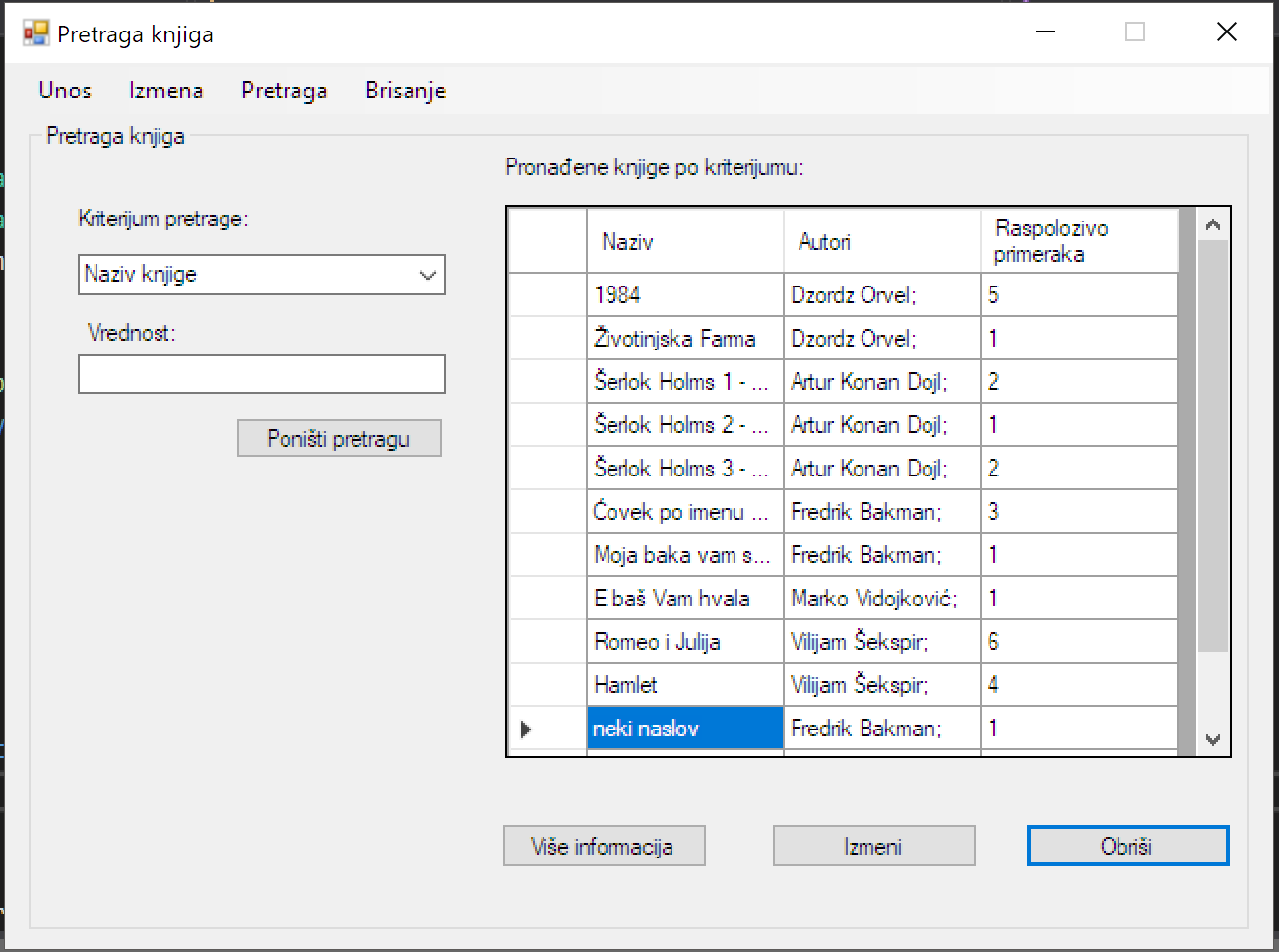
**Актори СК**

Библиотекар

**Учесници СК**

Библиотекар и систем (програм)

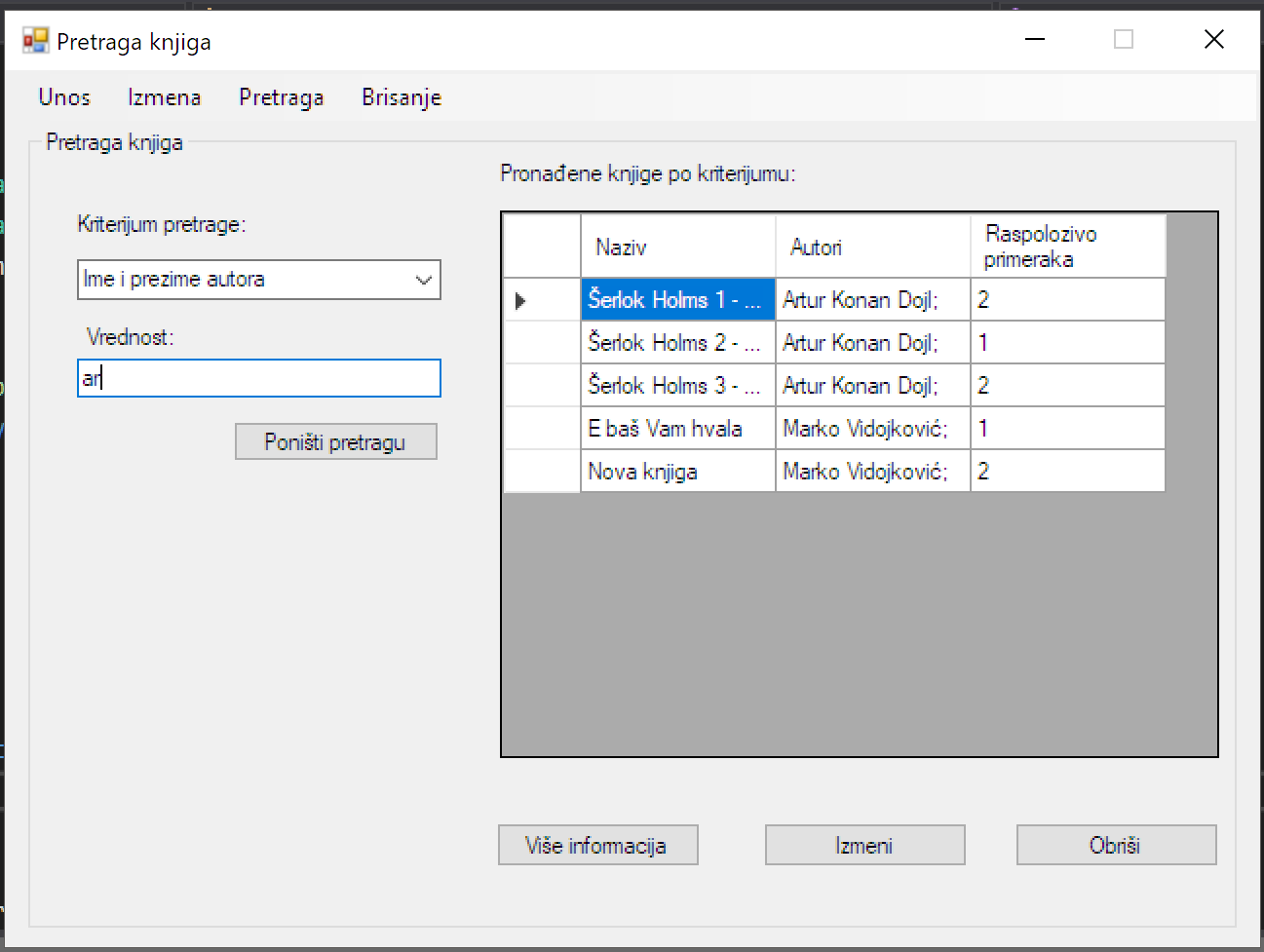
**Предуслов**: Систем је укључен и библиотекар je улогован под својом шифром. Систем приказује форму за рад са књигом.



Слика 73 - Форма за претрагу књига

**Основни сценарио СК**

1. Библиотекар **уноси** вредностпо којој претражује књиге. (АПУСО)



Слика 74 - Претрага књига: Попуњена форма

2. Библиотекар **позива** систем да нађе књиге по задатој вредности. (АПСО)

***Опис акције:*** *Библиотекар уношењем вредности у поље које је за то предвиђено, позива системску операцију* ***PretraziKnjigeSO****, која враћа све књиге који задовољавају услов.*

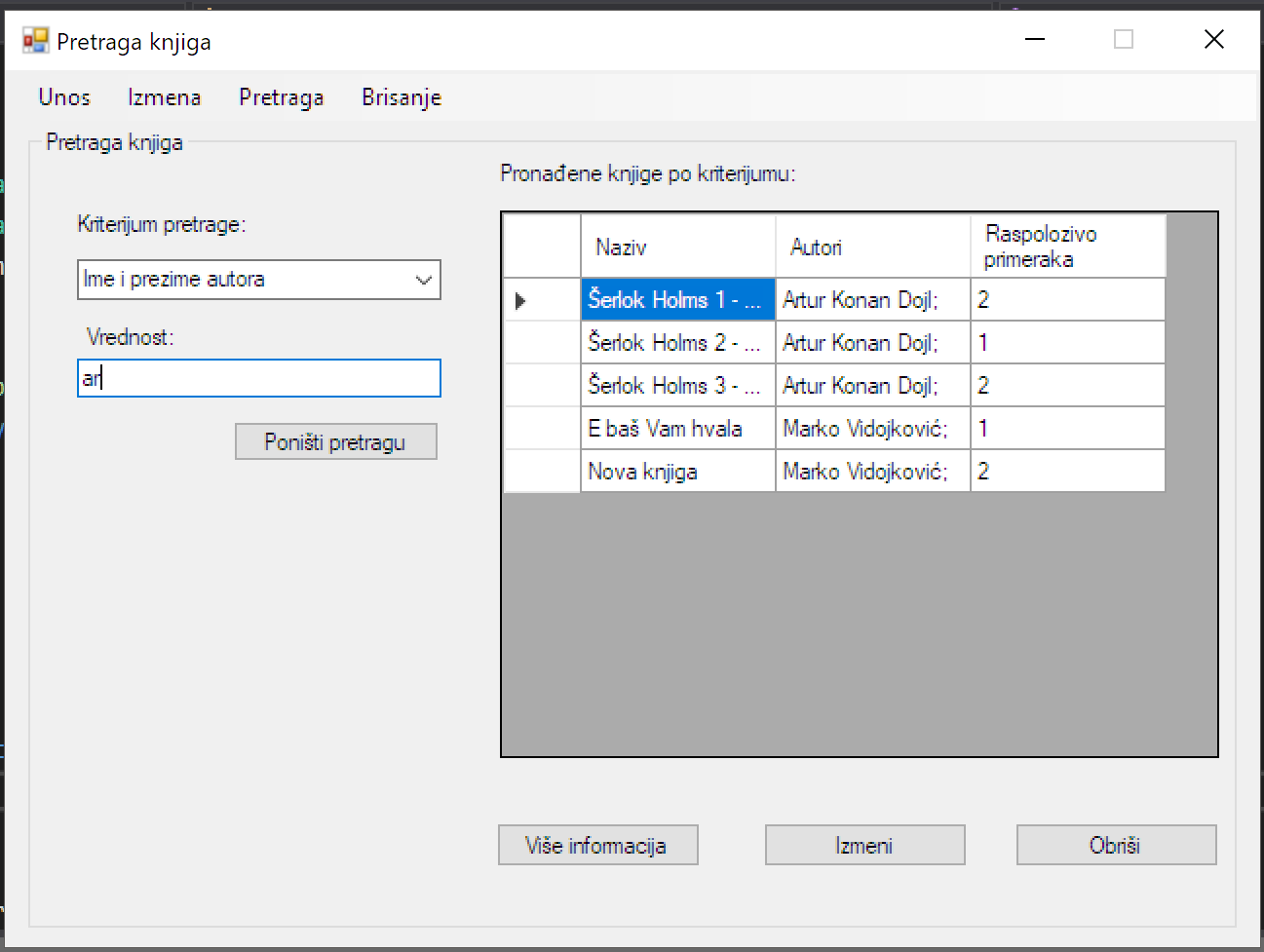
3. Систем **тражи** књиге по задатој вредности. (СО)

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

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

6. Библиотекар **позива** систем да нађе податке о одабраној књизи(АПСО)

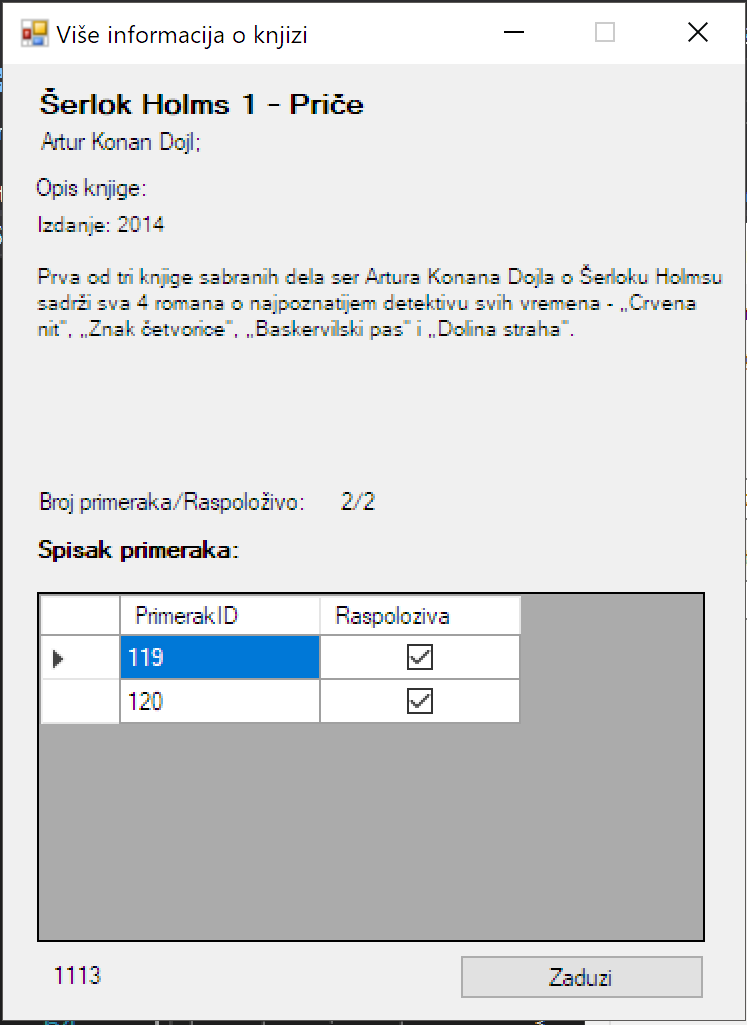
***Опис акције:*** *Библиотекар кликом на дугме ”Више информација” позива системску операцију* ***NadjiKnjiguSO,*** *која налази све информације о одабраној књизи.*



Слика 75 - Претрага књига: Позивање система да нађе одређену књигу

7. Систем **тражи** књигу по задатој вредности. (СО)

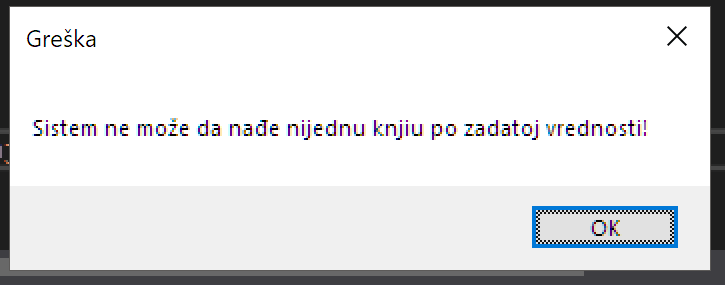
8. Систем приказује библиотекару податке о књизи и поруку: “Систем је нашао књигу по задатој вредности”. (ИА)



Слика 76 - Претрага књига: Информације о одабраној књизи

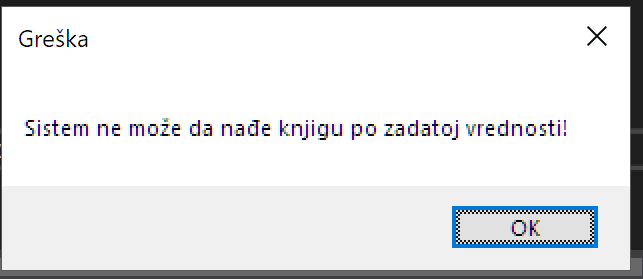
**Алтернативна сценарија**

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



Слика 77 - Грешка: Систем не може да нађе ниједну књигу

8.1 Уколико систем не може да нађе књигу он приказује библиотекару поруку: “Систем не може да нађе књигу по задатој вредности”. (ИА)



Слика 78 - Грешка: Систем не може да нађе књигу

**СК7 – Измена књиге**

**Назив СК**

Промена књиге

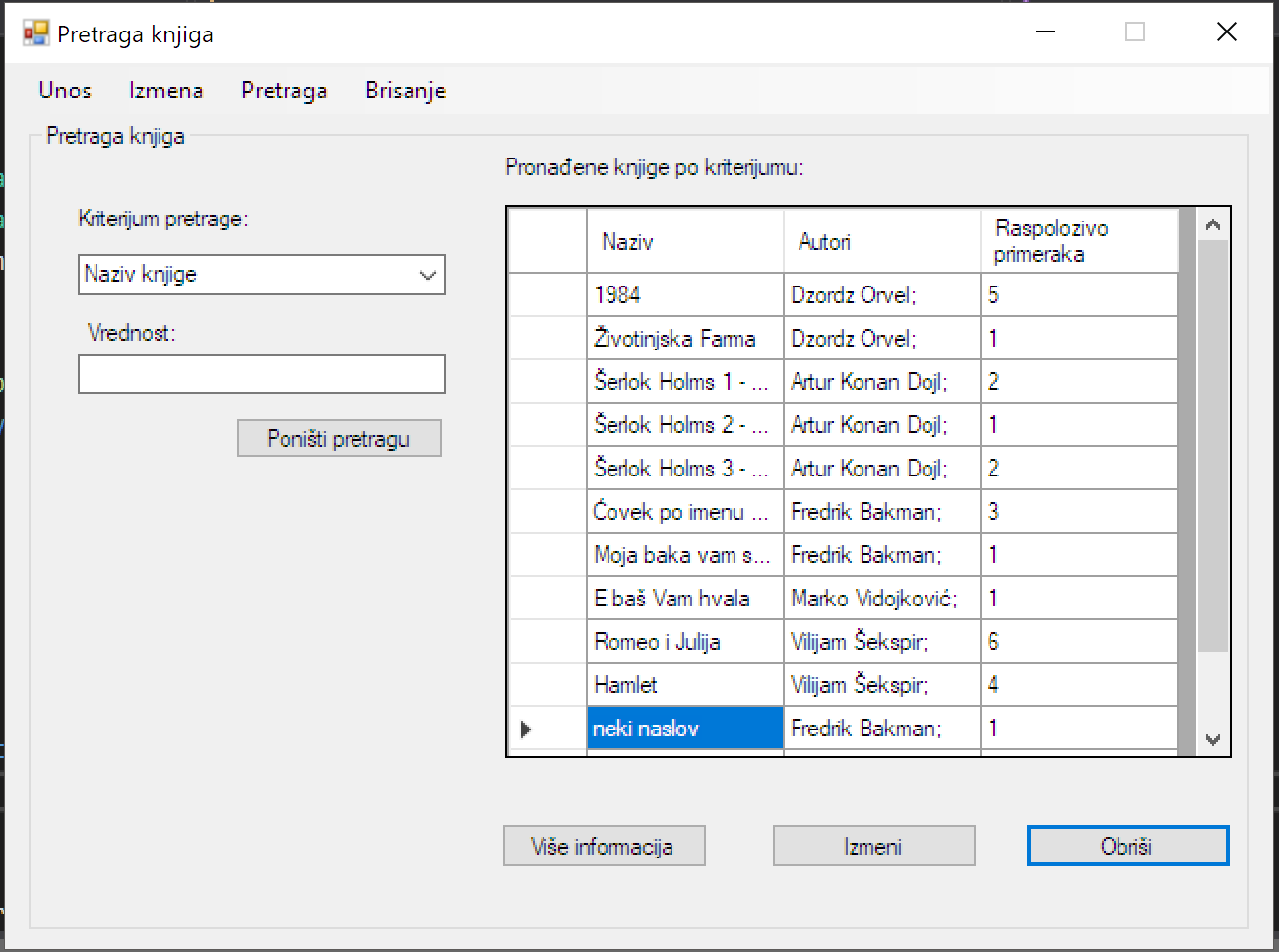
**Актори СК**

Библиотекар

**Учесници СК**

Библиотекар и систем (програм)

**Предуслов**: Систем је укључен и библиотекар je улогован под својом шифром. Систем приказује форму за рад са књигом.



Слика 79 - Форма за рад са књигом

**Основни сценарио СК**

1. Библиотекар **уноси** вредностпо којој претражује књиге. (АПУСО)

2. Библиотекар **позива** систем да нађе књиге по задатој вредности. (АПСО)

***Опис акције:*** *Библиотекар уношењем вредности у поље које је за то предвиђено, позива системску операцију* ***PretraziKnjigeSO****, која враћа све књиге који задовољавају услов.*

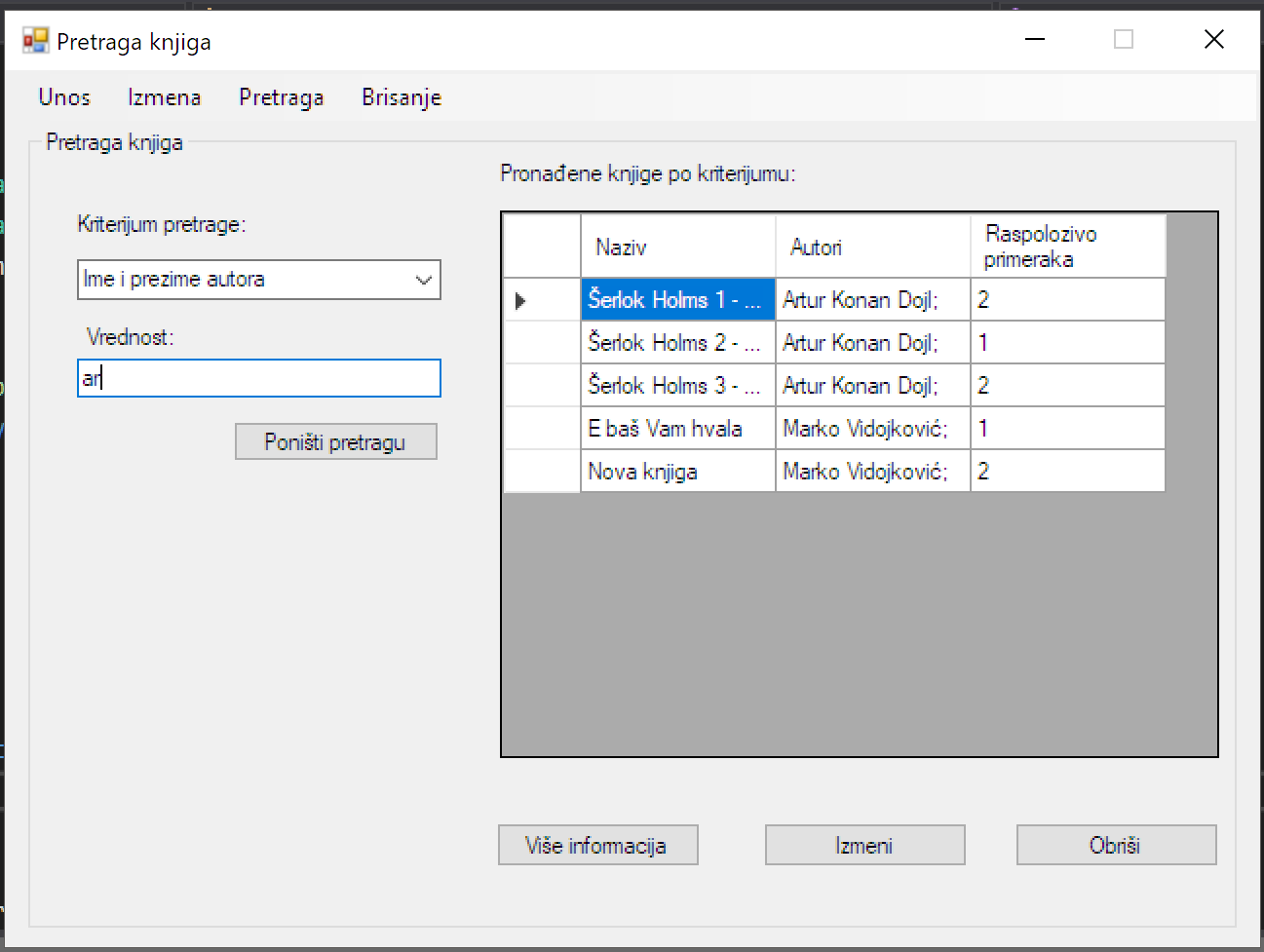
3. Систем **тражи** књиге по задатој вредности. (СО)

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

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

6. Библиотекар **позива** систем да нађе податке о одабраној књизи(АПСО)

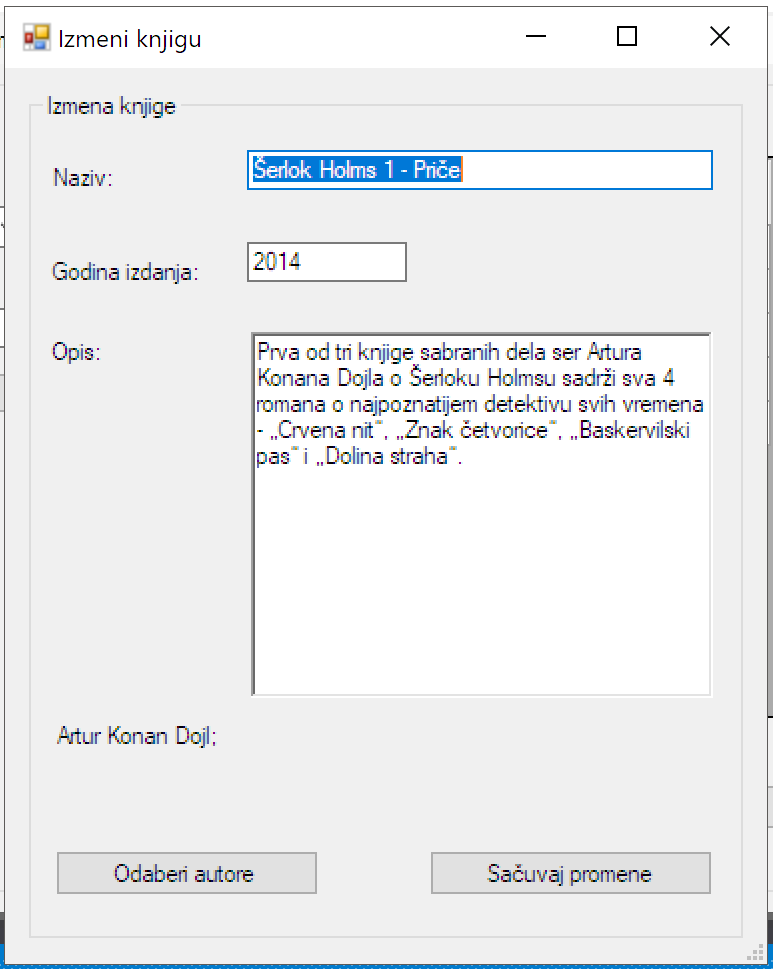
***Опис акције:*** *Библиотекар кликом на дугме ”Измени” позива системску операцију* ***NadjiKnjiguSO,*** *која налази све информације о одабраној књизи*



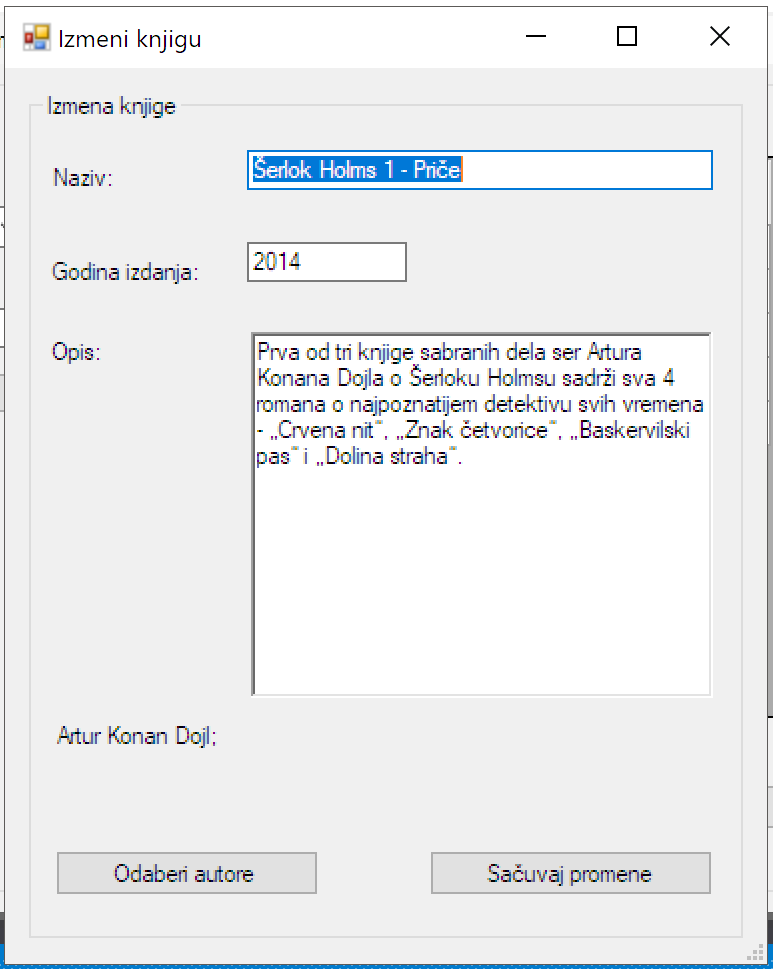
Слика 80 - Измена књиге: Одабир књиге

7. Систем **тражи** књигу по задатој вредности. (СО)

8. Систем приказује библиотекару податке о књизи и поруку: “Систем је нашао књигу по задатој вредности”. (ИА)



9. Библиотекар **уноси** **(мења)** податке о књизи. (АПУСО)

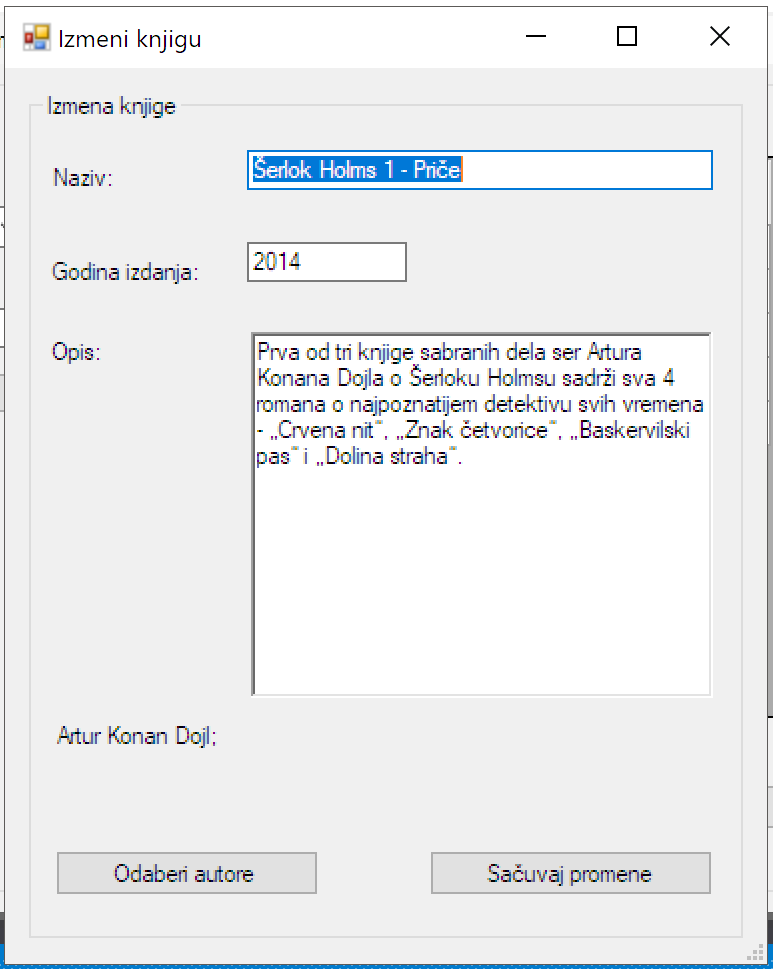
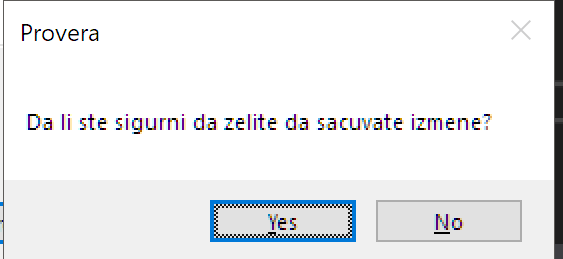


Слика 81 - Измена књиге

10. Библиотекар **контролише** да ли је коректно унео податке о књизи. (АНСО)

11. Библиотекар **позива** систем да промени податке о књизи. (АПСО)

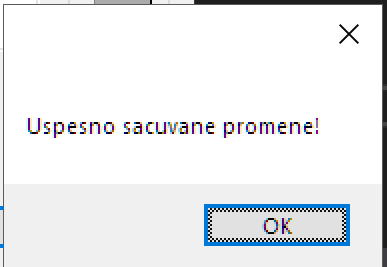
***Опис акције:*** *Библиотекар кликом на дугме ”Сачувај промене”, и затим ”Да” код провере, позива системску операцију* ***SacuvajIzmeneKnjigeSO,*** *која мења информације о одабраној књизи.*



Слика 82 - Измена књиге: Позив система да измени одабрану књигу

12. Систем **мења** податке о књизи. (СО)

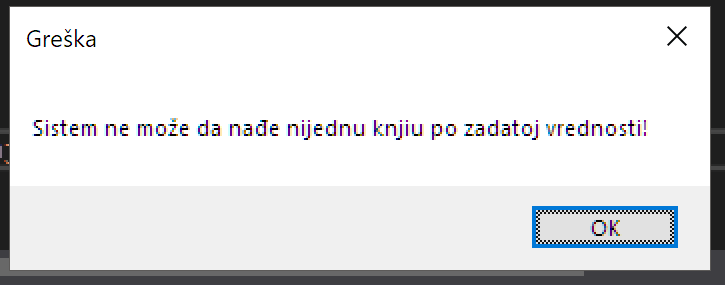
13. Систем **приказује** библиотекару промењену књигу и поруку: “Систем је успешно променио књигу.” (ИА)



Слика 83 - Измена књиге: Успешно сачуване промене

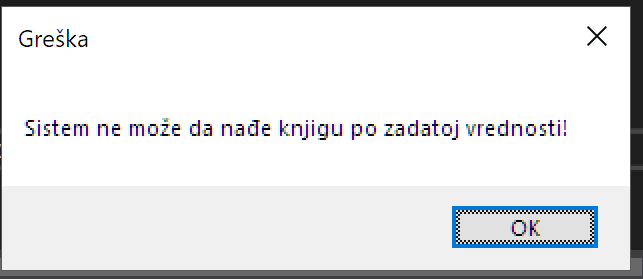
**Алтернативна сценарија**

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



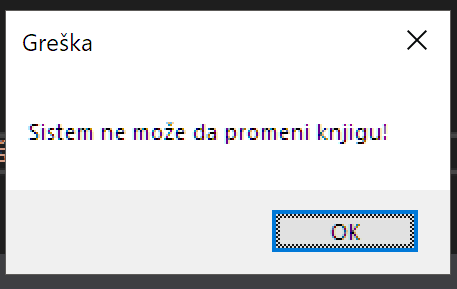
Слика 84 - Грешка: Систем не може да нађе ниједну књигу

8.1 Уколико систем не може да нађе књигу он приказује библиотекару поруку: “Систем не може да нађе књигу по задатој вредности”. Прекида се извршење сценарија. (ИА)



Слика 85 - Грешка: Систем не може да нађе књигу

13.1 Уколико систем не може да запамти податке о књизи он приказује библиотекару поруку “Систем не може да промени књигу”. (ИА)



Слика 86 - Грешка: Систем не може да промени књигу

**СК8 – Брисање књиге**

**Назив СК**

Брисање књиге

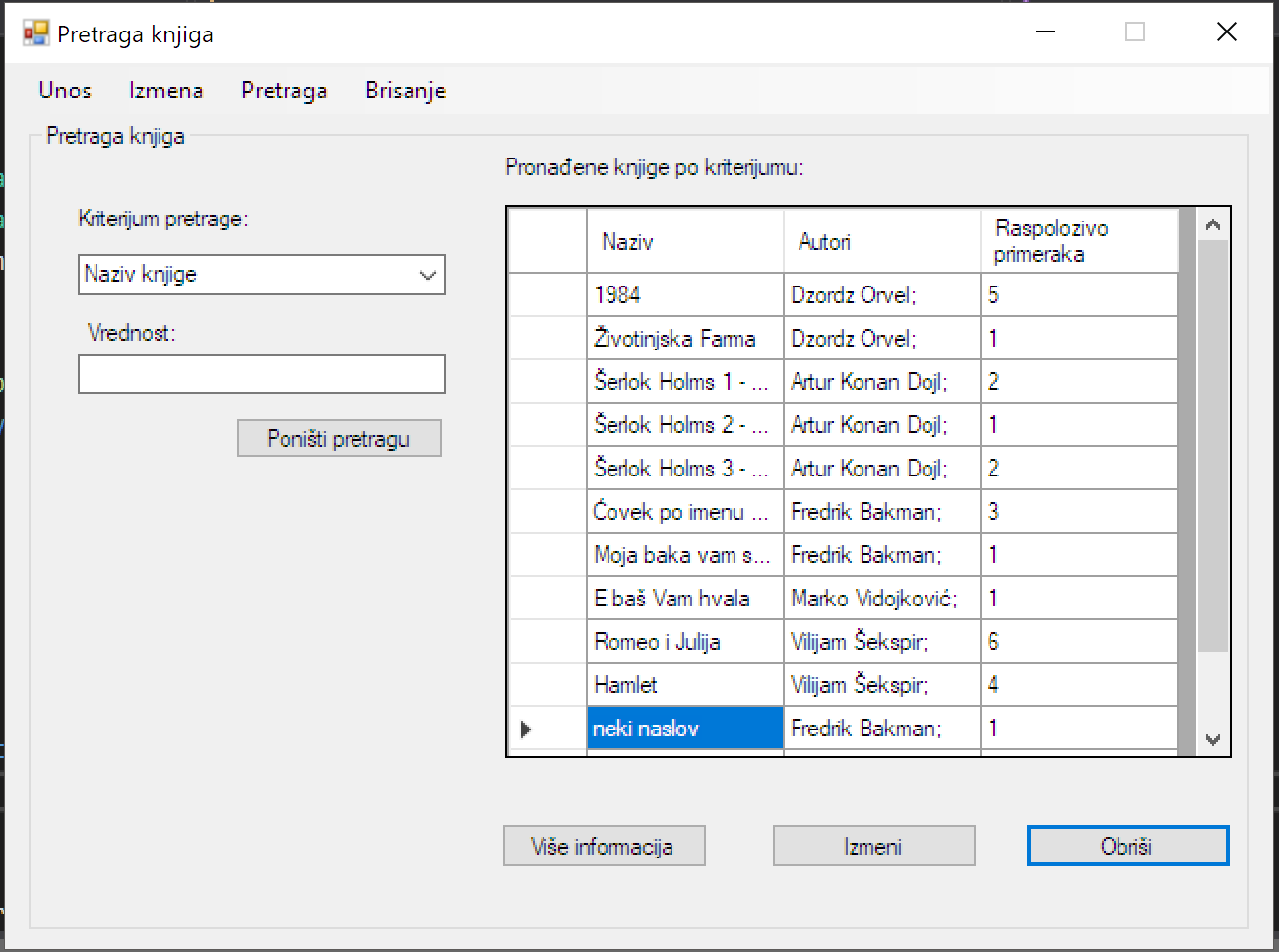
**Актори СК**

Библиотекар

**Учесници СК**

Библиотекар и систем (програм)

**Предуслов**: Систем је укључен и библиотекар je улогован под својом шифром. Систем приказује форму за рад са књигом.



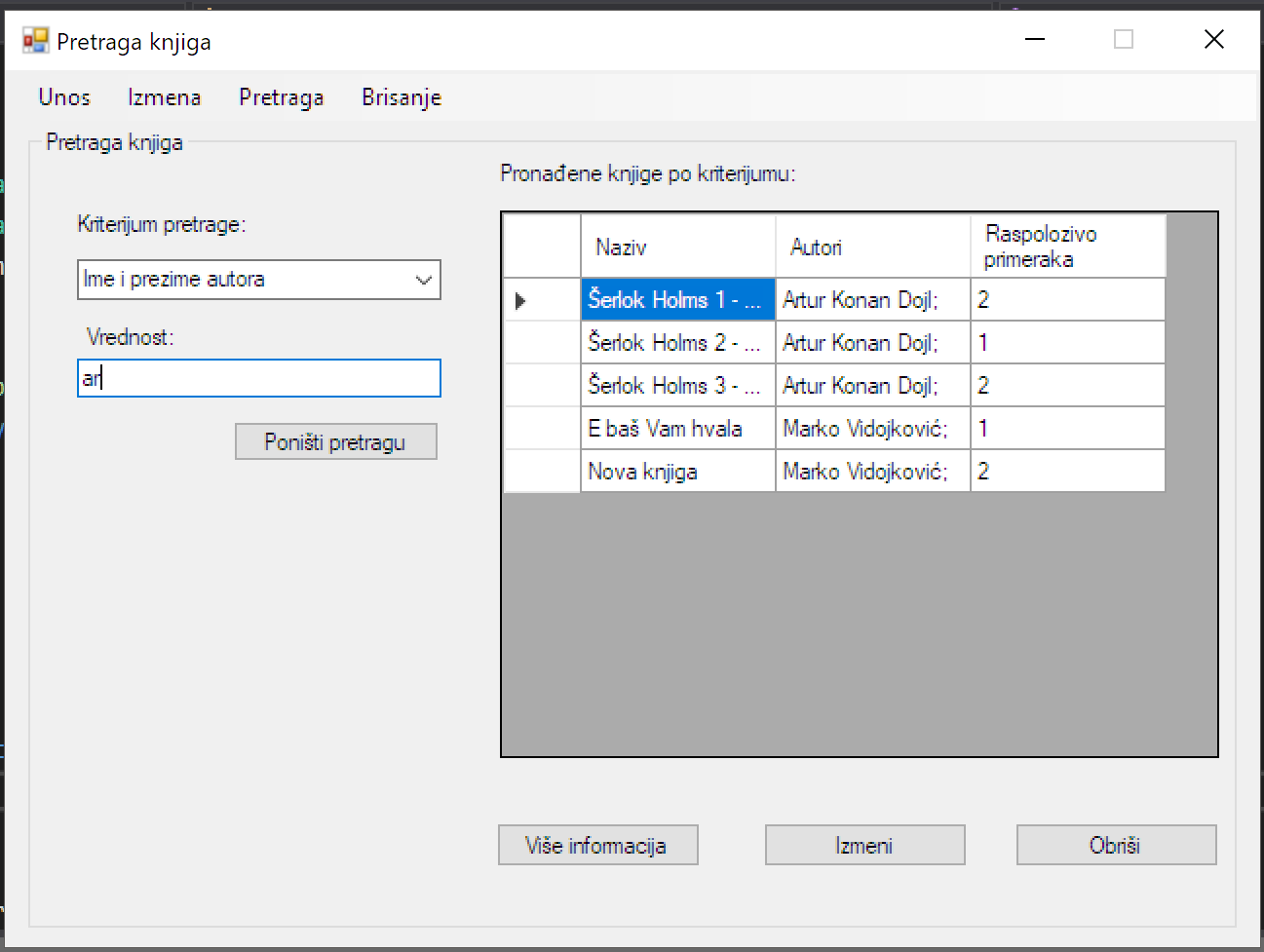
Слика 87 - Форма за рад са књигом

**Основни сценарио СК**

1. Библиотекар **уноси** вредностпо којој претражује књиге. (АПУСО)
2. Библиотекар **позива** систем да нађе књиге по задатој вредности. (АПСО)

***Опис акције:*** *Библиотекар уношењем вредности у поље које је за то предвиђено, позива системску операцију* ***PretraziKnjigeSO,***  *која враћа све књиге који задовољавају услов.*

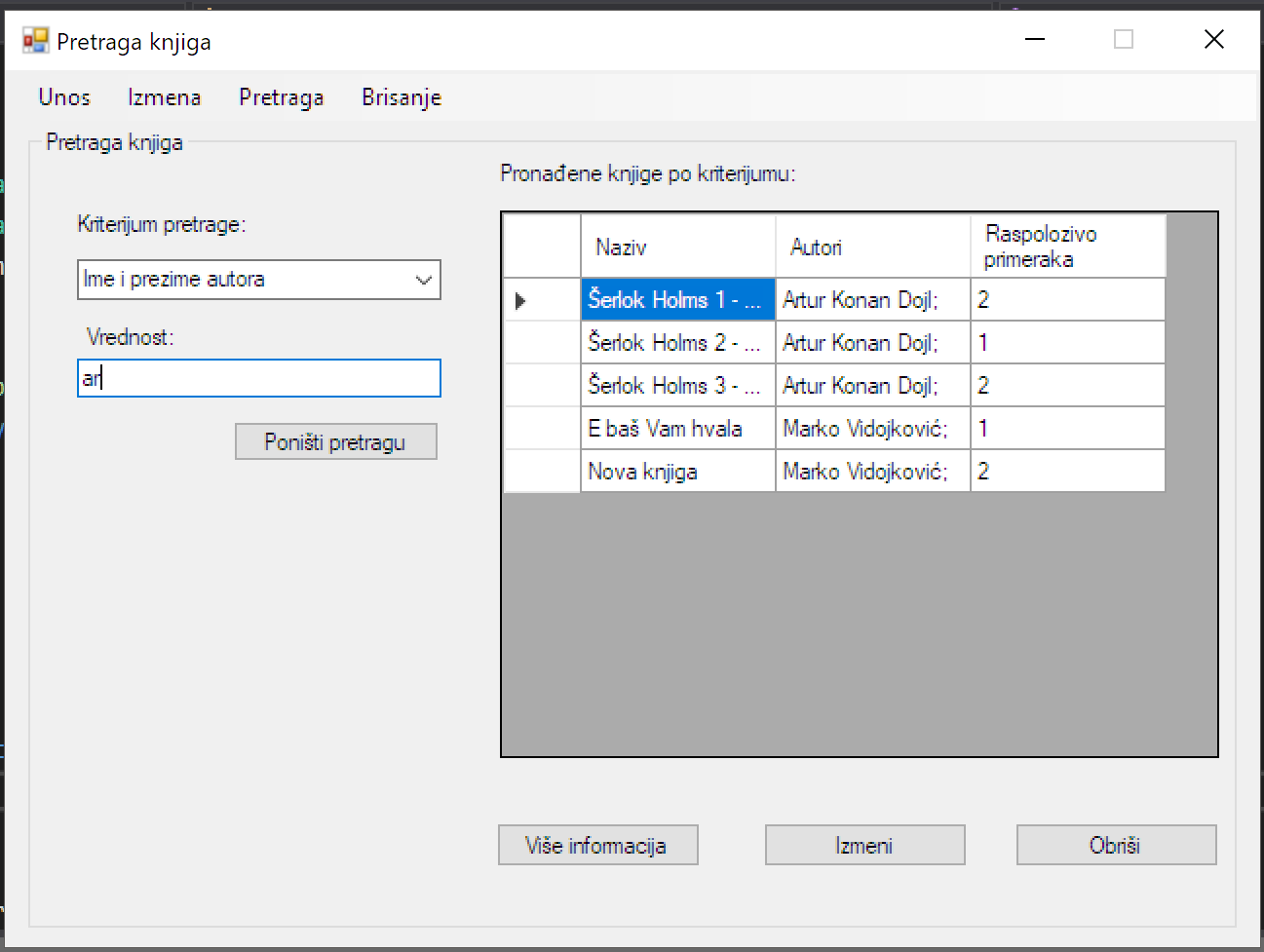
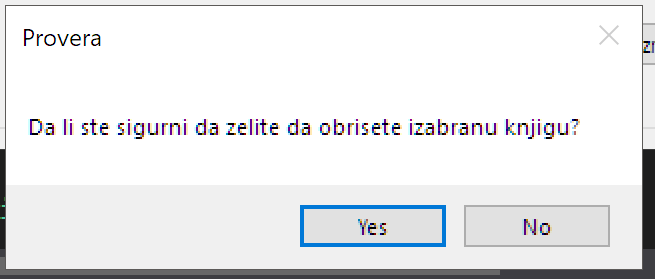
1. Систем **тражи** књиге по задатој вредности. (СО)
2. Систем приказује библиотекару податке о књигама и поруку: “Систем је нашао књиге по задатој вредности”. (ИА)
3. Библиотекар **бира** књигу. (АПУСО)
4. Библиотекар **позива** систем да нађе податке о одабраној књизи(АПСО)
5. Систем **тражи** књигу по задатој вредности. (СО)
6. Систем приказује библиотекару податке о књизи и поруку: “Систем је нашао књигу по задатој вредности”. (ИА)



Слика 88 - Брисање књиге: Позив система да обрише одабрану књигу

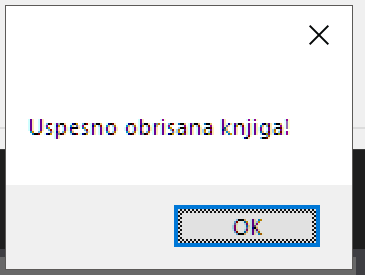
1. Библиотекар **позива** систем да обрише књигу. (АПСО)

***Опис акције:*** *Библиотекар кликом на дугме ”Обриши”, и затим ”Да” код провере, позива системску операцију* ***ObrisiKnjiguSO,***  *која брише одабрану књигу.*



Слика 89 - Брисање књиге: Позивање система да обрише књигу

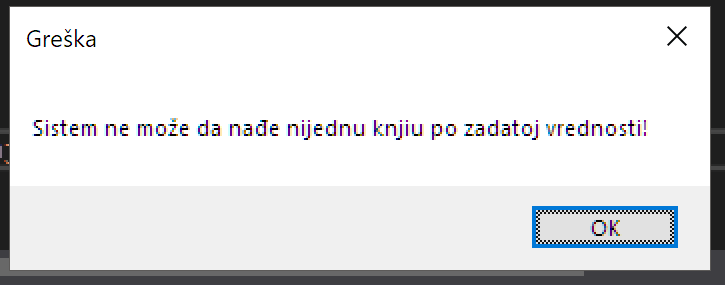
1. Систем **брише** књигу. (СО)
2. Систем **приказује** библиотекару поруку: “Систем је обрисао књигу.” (ИА)



Слика 90 - Брисање књиге: Успешно обрисана књига

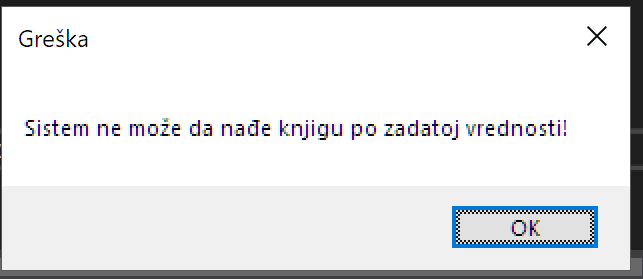
**Алтернативна сценарија**

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



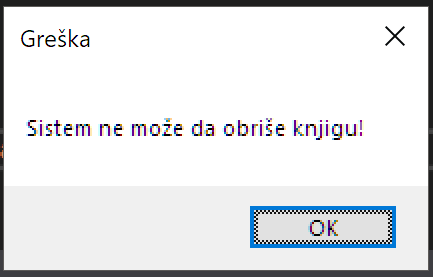
Слика 91 - Грешка: Систем не може да нађе ниједну књигу

8.1 Уколико систем не може да нађе књигу он приказује библиотекару поруку: “Систем не може да нађе књигу по задатој вредности”. Прекида се извршење сценарија. (ИА)



Слика 92 - Грешка: Систем не може да нађе књигу

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



Слика 93 - Грешка: Систем не може да обрише књигу

**СК9 – Задуживање**

**Назив СК**

Креирање задужења

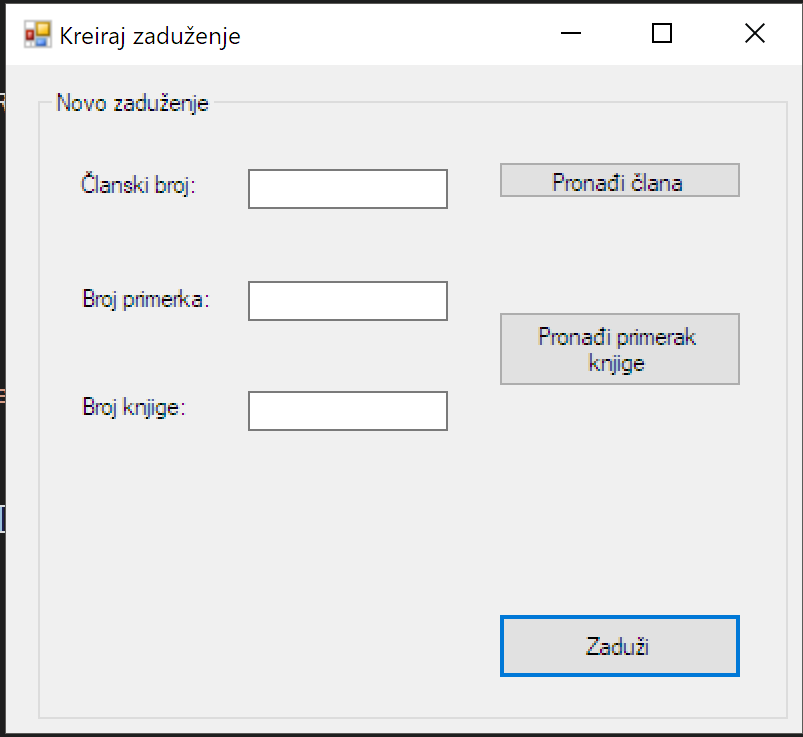
**Актори СК**

Библиотекар

**Учесници СК**

Библиотекар и систем (програм)

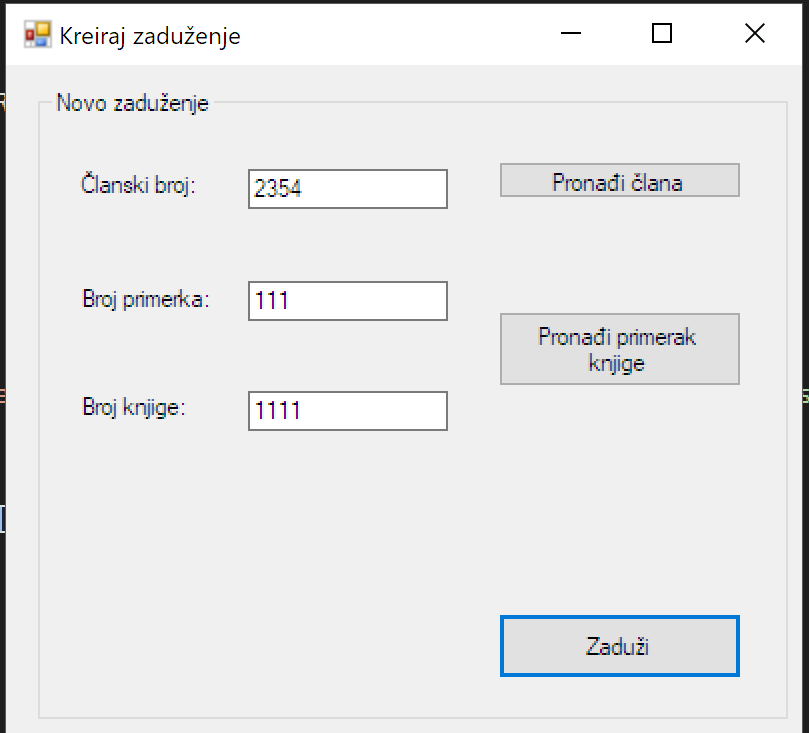
**Предуслов**: Систем је укључен и библиотекар je улогован под својом шифром. Систем приказује форму за рад са задужењем. Учитана је листа чланова и књига.



Слика 94 - Задужи: Непопуњена форма

**Основни сценарио СК**

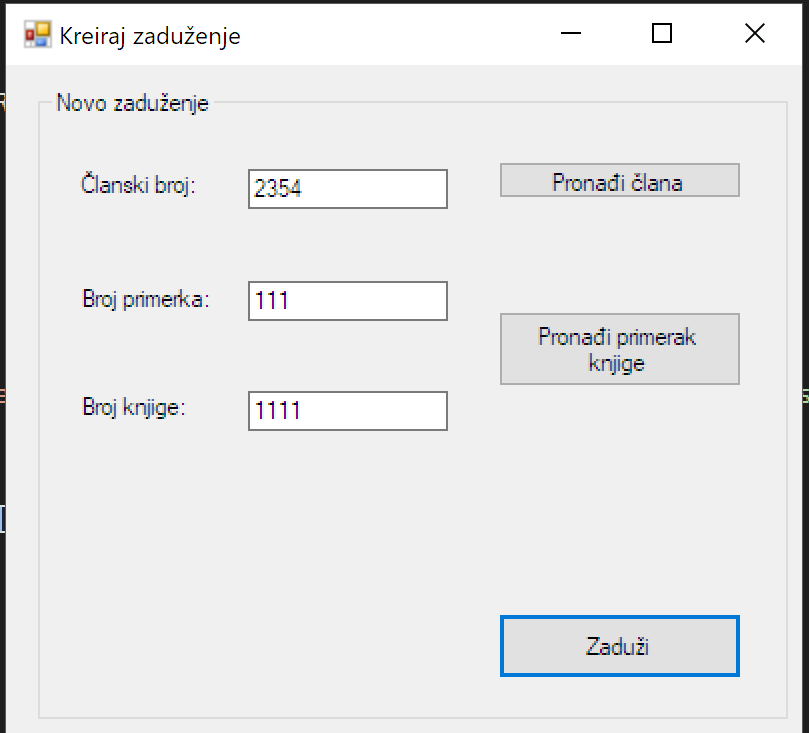
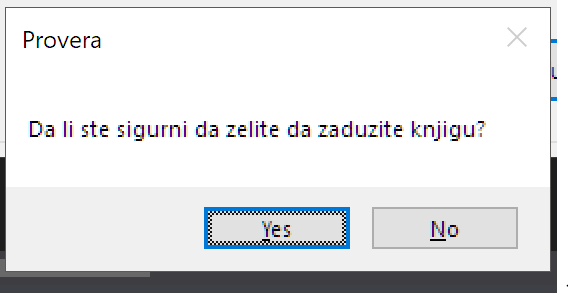
1. Библиотекар **уноси** податке о задужењу. (АПУСО)



Слика 95 - Задужи: Попуњена форма

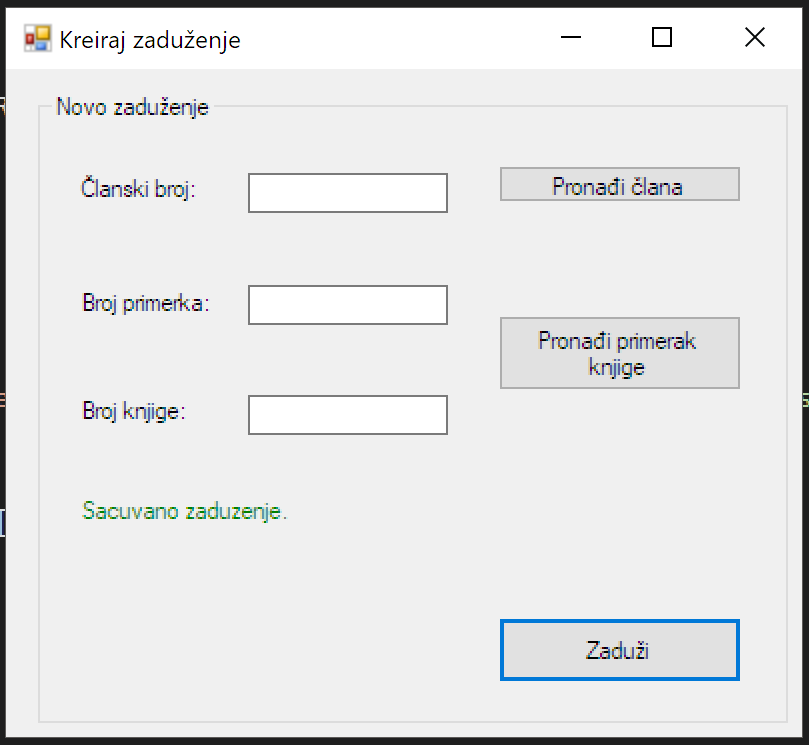
1. Библиотекар **контролише** да ли је коректно унео податке о задужењу. (АНСО)
2. Библиотекар **позива** систем да креира и запамти податке о задужењу. (АПСО)

***Опис акције:*** *Библиотекар кликом на дугме ”Задужи”, а затим ”Да”, позива системску операцију* ***ZaduziSO,*** *која креира и памти ново задужење.*



Слика 96 - Задужи: Позив система да направи задужење

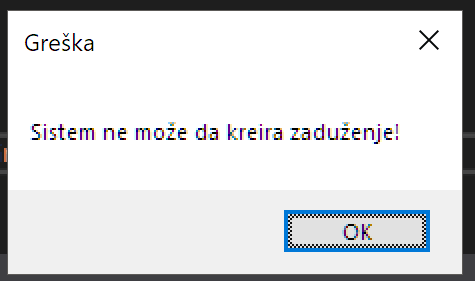
1. Систем **памти** податке о задужењу. (СО)
2. Систем **приказује** библиотекару запамћено задужење и поруку: “Систем је креирао задужење“. (ИА)



Слика 97 - Задужи: Успешно креирано задужење

Алтернативна сценарија

5.1 Уколико систем не може да запамти податке о задужењу он приказује библиотекару поруку “Систем не може да креира задужење”. (ИА)



Слика 98 - Грешка: Систем не може да креира задужење

**СК10 – Раздуживање**

**Назив СК**

Измена задужења

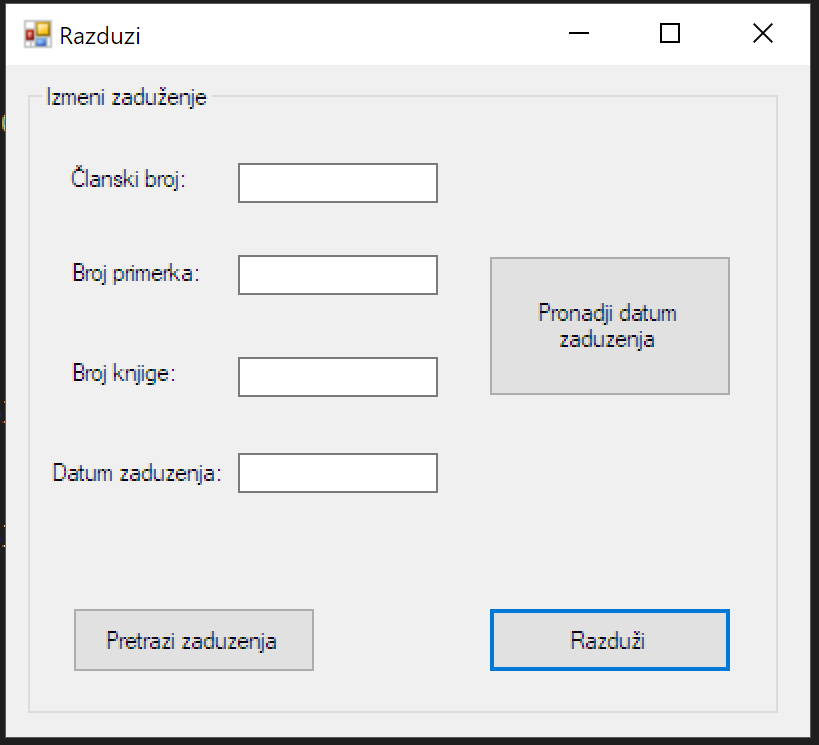
**Актори СК**

Библиотекар

**Учесници СК**

Библиотекар и систем (програм)

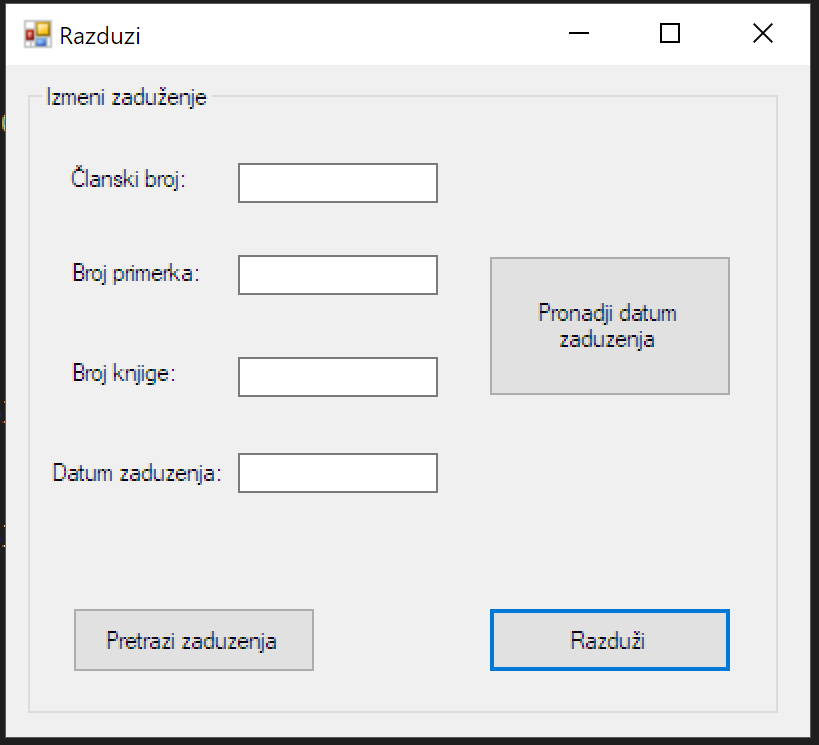
**Предуслов**: Систем је укључен и библиотекар je улогован под својом шифром. Систем приказује форму за рад са задужењем.

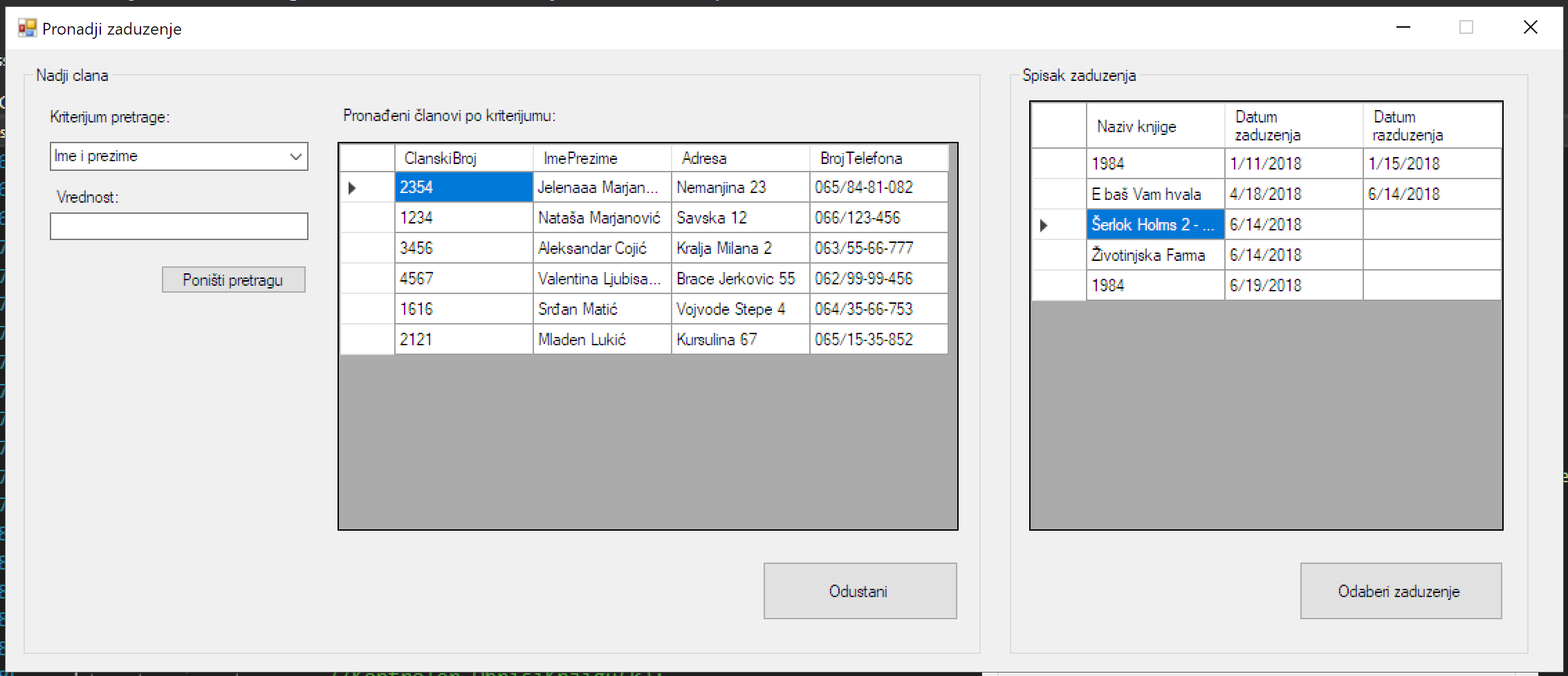


Слика 99 - Раздужи: Непопуњена форма

**Основни сценарио СК**

1. Библиотекар **уноси** вредностпо којој претражује задужења. (АПУСО)





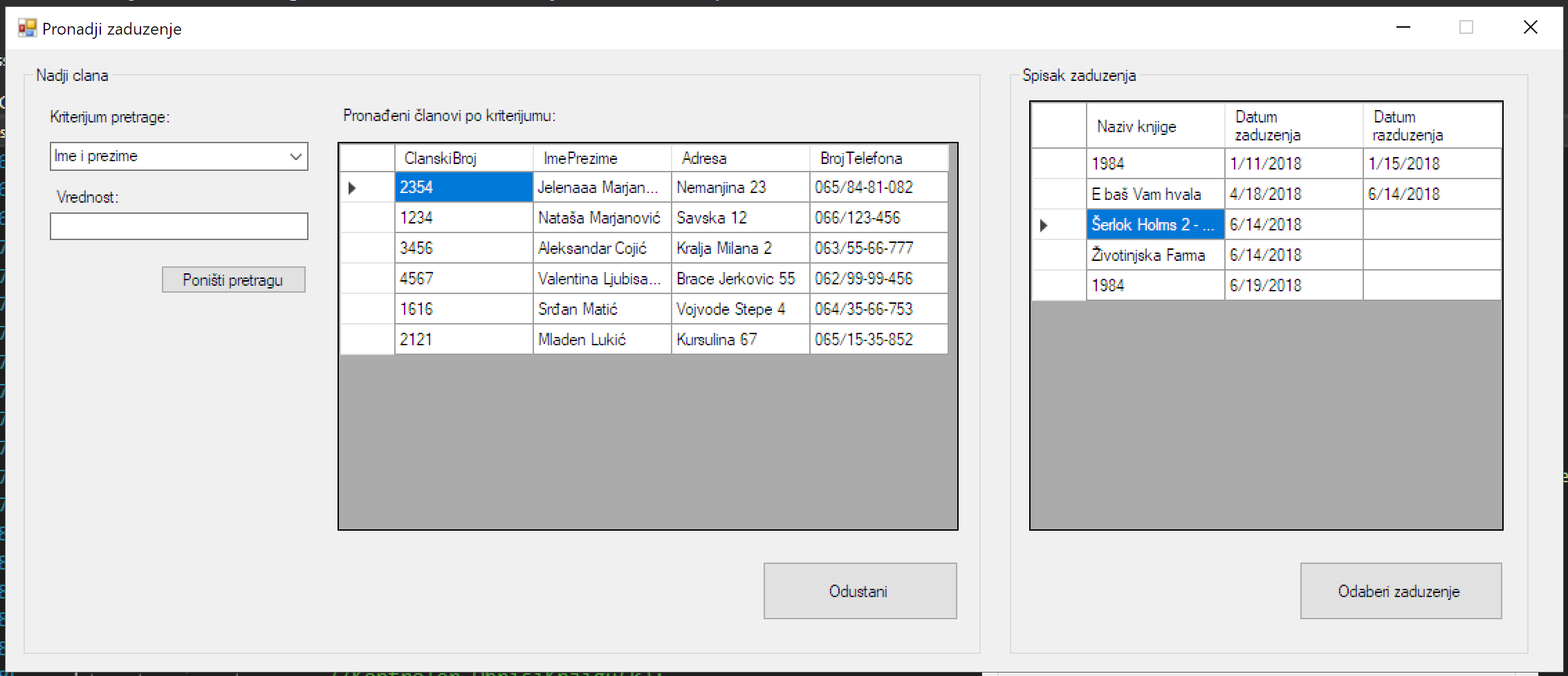
Слика 100 - Раздужи: Претрага задужења

1. Библиотекар **позива** систем да нађе задужења по задатој вредности. (АПСО)

***Опис акције:*** *Библиотекар кликом на дугме ”Претражи задужења” отвара нову форму где му се налазе сви чланови и сва њихова задужења. Уношењем вредности у поље које је за то предвиђено, позива системску операцију* ***PretraziClanoveSO,*** *која враћа све чланове и њихова задужења који задовољавају услов.*

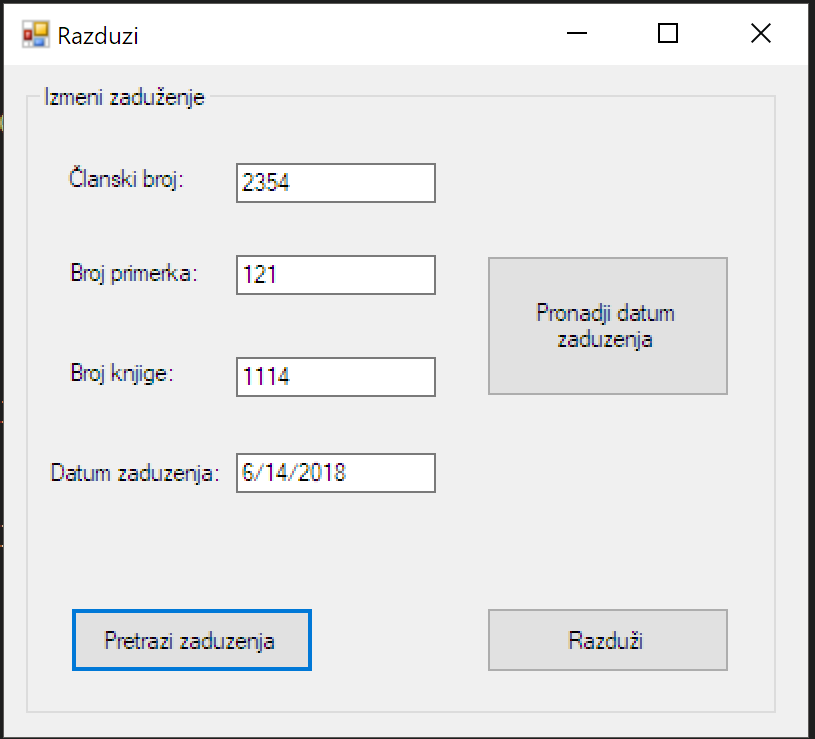
1. Систем **тражи** задужења по задатој вредности. (СО)
2. Систем приказује библиотекару задужења и поруку: “Систем је нашао задужења по задатој вредности”. (ИА)
3. Библиотекар **бира** задужење. (АПУСО)
4. Библиотекар **позива** систем да нађе податке о одабраном задужењу (АПСО)

***Опис акције:*** *Библиотекар кликом на дугме ”Одабери задужење”, позива системску операцију* ***NadjiZaduzenjeSO,*** *која враћа информације о одабраном задужењу.*



Слика 101 - Раздужи: Одабир задужења

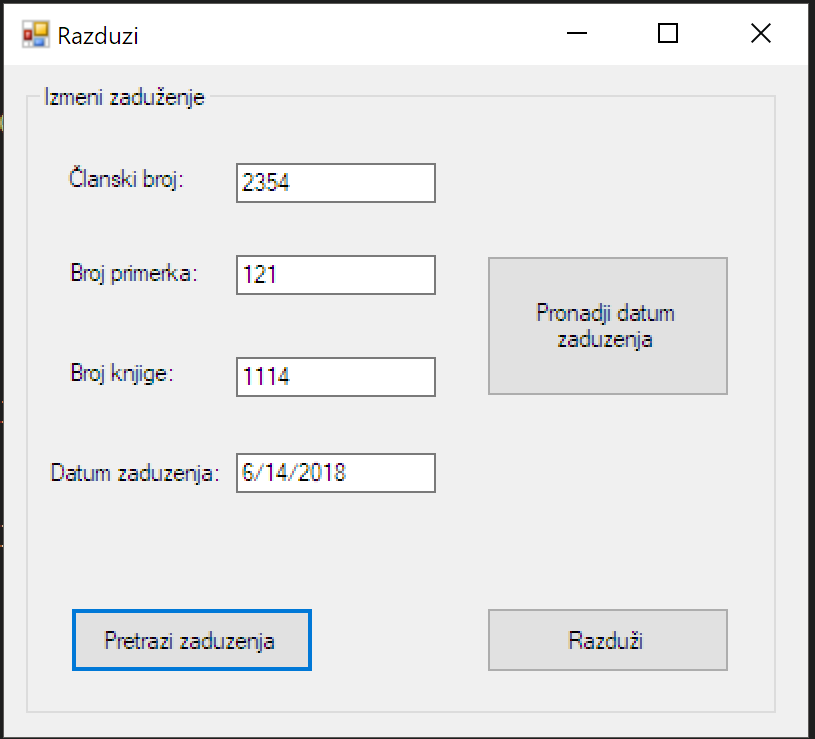
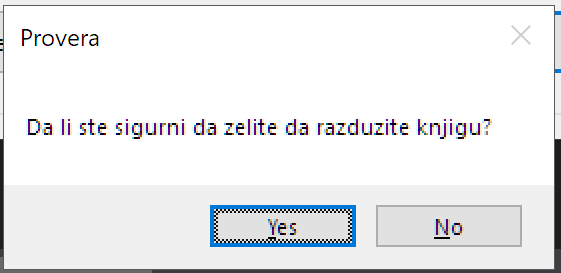
1. Систем **тражи** задужење по задатој вредности. (СО)
2. Систем приказује библиотекару податке о задужењу и поруку: “Систем је нашао задужење по задатој вредности”. (ИА)



Слика 102 - Раздужи: Попуњена форма

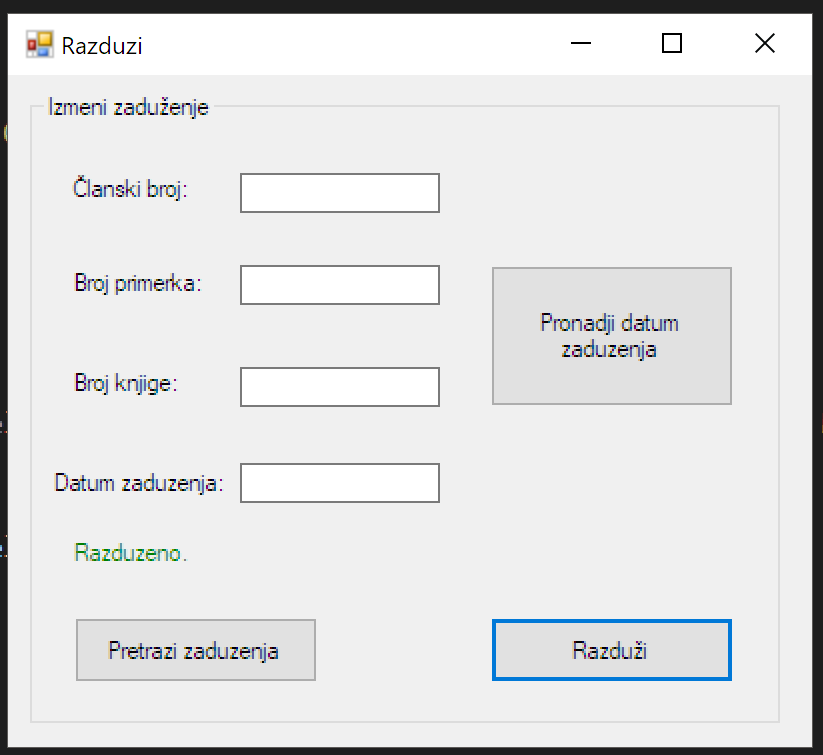
1. Библиотекар **уноси** **(мења)** податке о задужењу. (АПУСО)
2. Библиотекар **контролише** да ли је коректно унео податке о задужењу. (АНСО)
3. Библиотекар **позива** систем да промени податке о задужењу. (АПСО)

***Опис акције:*** *Библиотекар кликом на дугме ”Раздужи”, а затим ”Да”, позива системску операцију* ***RazduziSO,*** *која мења одабрано задужење (поставља атрибут ДатумДо на текући датум).*



Слика 103 - Раздужи: Позив система да раздужи књигу

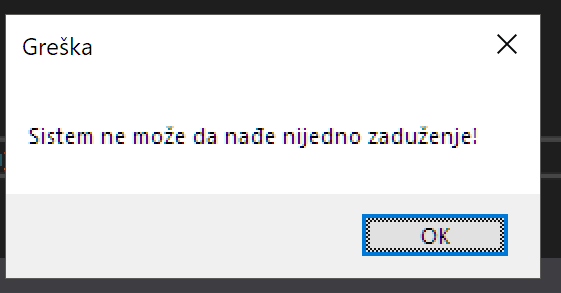
1. Систем **мења** податке о задужењу. (СО)
2. Систем **приказује** библиотекару промењено задужење и поруку: “Систем је променио задужење.” (ИА)



Слика 104 - Раздужи: Успешно раздужена књига

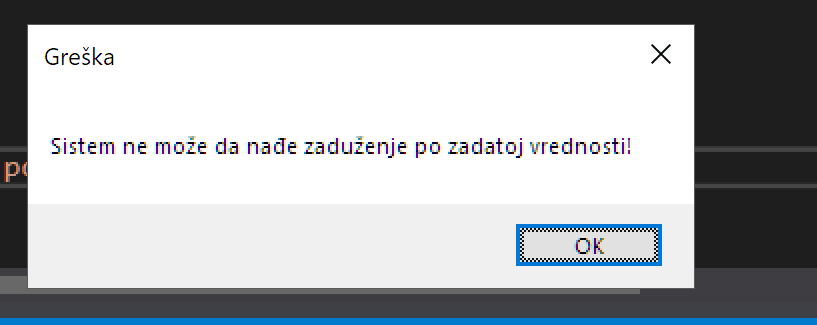
**Алтернативна сценарија**

4.1 Уколико систем не може да нађе ниједно задужење он приказује библиотекару поруку: “Систем не може да нађе ниједно задужење по задатој вредности”. Прекида се извршење сценарија. (ИА)



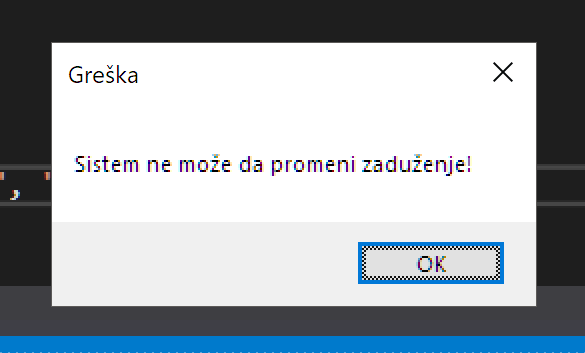
Слика 105 - Грешка: Систем не може да нађе ниједно задужење

8.1 Уколико систем не може да нађе задужење он приказује библиотекару поруку: “Систем не може да нађе задужење по задатој вредности”. Прекида се извршење сценарија. (ИА)



Слика 106 - Грешка: Систем не може да нађе задужење

13.1 Уколико систем не може да промени податке о задужење он приказује библиотекару поруку “Систем не може да промени задужење”. (ИА)



Слика 107 - Грешка: Систем не може да промени задужење

## 4.1.2 Пројектовање контролера корисничког интерфејса

Kонтролер корисничког интерфејса је одговоран да:

а) прихвати податке које шаље екранска форма,

б) конвертује податке (који се налазе у графичким елементима) у објекат који представља улазни аргумент СО која ће бити позвана,

ц) шаље захтев за извршење СО до апликационог сервера (софтверског система),

д) прихвата објекат (излаз) софтверског система који настаје као резултат извршења СО и

е) конвертује објекат у податке графичких елемената.

Контролер корисничког интерфејса (класа Комуникација) изгледа овако:

public class Komunikacija  
    {  
        private static Komunikacija instance;  
  
        TcpClient klijent;  
        NetworkStream tok;  
        BinaryFormatter formater = new BinaryFormatter();  
        MainForm glavnaForma;  
  
        private Komunikacija()  
        {  
  
        }  
  
        public static Komunikacija Instance  
        {  
            get  
            {  
                if (instance == null)  
                {  
                    instance = new Komunikacija();  
                }  
                return instance;  
            }  
        }  
  
        public void PoveziSe(MainForm mf)  
        {  
            try  
            {  
                glavnaForma = mf;  
                this.klijent = new TcpClient("127.0.0.1", 21212);  
                this.tok = klijent.GetStream();  
                glavnaForma.Povezan = true;  
            }  
            catch (Exception ex)  
            {  
                throw ex;  
            }  
        }  
  
        public Bibliotekar PrijaviSe(string korisnickoIme, string lozinka)  
        {  
            try  
            {  
                TransferKlasa zahtev = new TransferKlasa()  
                {  
                    TransferObjekat = new Bibliotekar()  
                    {  
                        KorisnickoIme = korisnickoIme,  
                        Lozinka = lozinka  
                    },  
                    Operacija = Operacija.Login  
                };  
                formater.Serialize(tok, zahtev);  
                TransferKlasa odgovor = formater.Deserialize(tok) as TransferKlasa;  
                if (odgovor.Signal)  
                {  
                    return odgovor.TransferObjekat as Bibliotekar;  
                }  
                else  
                {  
                    return null;  
                }  
            }  
            catch (Exception e)  
            {  
                MessageBox.Show("Izgubljena je konekcija sa serverom! Zatvaram formu...");  
                klijent.Close();  
                glavnaForma.Povezan = false;  
                glavnaForma.Zatvori();  
                return null;  
            }  
        }  
  
        public void Kraj()  
        {  
            try  
            {  
                TransferKlasa transfer = new TransferKlasa  
                {  
                    Operacija = Operacija.Kraj,  
                };  
                formater.Serialize(tok, transfer);  
                klijent.Close();  
            }  
            catch (Exception)  
            {  
                MessageBox.Show("Izgubljena je konekcija sa serverom! Zatvaram formu...");  
                klijent.Close();  
                glavnaForma.Povezan = false;  
                glavnaForma.Zatvori();  
            }  
        }  
  
        public bool Razduzi(Zaduzenje z)  
        {  
            try  
            {  
                TransferKlasa zahtev = new TransferKlasa()  
                {  
                    TransferObjekat = z,  
                    Operacija = Operacija.Razduzi  
                };  
                formater.Serialize(tok, zahtev);  
                TransferKlasa odgovor = formater.Deserialize(tok) as TransferKlasa;  
  
                return odgovor.Signal;  
            }  
            catch (Exception)  
            {  
                MessageBox.Show("Izgubljena je konekcija sa serverom! Zatvaram formu...");  
                klijent.Close();  
                glavnaForma.Zatvori();  
                return false;  
            }  
        }  
  
        public Clan NadjiClana(int clanskiBroj)  
        {  
            try  
            {  
                TransferKlasa zahtev = new TransferKlasa()  
                {  
                    Operacija = Operacija.NadjiClana,  
                    TransferObjekat = clanskiBroj  
                };  
                formater.Serialize(tok, zahtev);  
                TransferKlasa odgovor = formater.Deserialize(tok) as TransferKlasa;  
  
                if (odgovor.Signal)  
                    return odgovor.TransferObjekat as Clan;  
                else return null;  
            }  
            catch (Exception)  
            {  
                MessageBox.Show("Izgubljena je konekcija sa serverom! Zatvaram formu...");  
                klijent.Close();  
                glavnaForma.Povezan = false;  
                glavnaForma.Zatvori();  
                return null;  
            }  
        }  
  
        public bool SacuvajIzmeneClan(Clan c)  
        {  
            try  
            {  
                TransferKlasa zahtev = new TransferKlasa()  
                {  
                    Operacija = Operacija.SacuvajIzmeneClan,  
                    TransferObjekat = c  
                };  
                formater.Serialize(tok, zahtev);  
                TransferKlasa odgovor = formater.Deserialize(tok) as TransferKlasa;  
  
                return odgovor.Signal;  
            }  
            catch (Exception)  
            {  
                MessageBox.Show("Izgubljena je konekcija sa serverom! Zatvaram formu...");  
                klijent.Close();  
                glavnaForma.Povezan = false;  
                glavnaForma.Zatvori();  
                return false;  
            }  
        }  
  
        public List<Clan> VratiSveClanove()  
        {  
            try  
            {  
                List<Clan> listaClanova = new List<Clan>();  
                TransferKlasa zahtev = new TransferKlasa()  
                {  
                    Operacija = Operacija.VratiSveClanove  
                };  
                formater.Serialize(tok, zahtev);  
                TransferKlasa odgovor = formater.Deserialize(tok) as TransferKlasa;  
  
                if (odgovor.Signal)  
                {  
                    List<IOpstiDomenskiObjekat> lista = odgovor.TransferObjekat as List<IOpstiDomenskiObjekat>;  
                    foreach (IOpstiDomenskiObjekat odo in lista)  
                    {  
                        listaClanova.Add((Clan)odo);  
                    }  
                    return listaClanova;  
                }  
                else  
                {  
                    return null;  
                }  
            }  
            catch (Exception)  
            {  
                MessageBox.Show("Izgubljena je konekcija sa serverom! Zatvaram formu...");  
                klijent.Close();  
                glavnaForma.Povezan = false;  
                glavnaForma.Zatvori();  
                return null;  
            }  
        }  
  
        public List<Clan> PretraziClanove(string vrednost, KriterijumPretrage kriterijum)  
        {  
            try  
            {  
                List<Clan> listaPronadjenihClanova = new List<Clan>();  
                TransferKlasa zahtev = new TransferKlasa()  
                {  
                    Operacija = Operacija.PretraziClanove,  
                    TransferObjekat = new Pretraga()  
                    {  
                        KriterijumPretrage = kriterijum,  
                        Vrednost = vrednost  
                    }  
                };  
                formater.Serialize(tok, zahtev);  
                TransferKlasa odgovor = formater.Deserialize(tok) as TransferKlasa;  
  
                if (odgovor.Signal)  
                {  
                    List<IOpstiDomenskiObjekat> lista = odgovor.TransferObjekat as List<IOpstiDomenskiObjekat>;  
                    foreach (IOpstiDomenskiObjekat odo in lista)  
                    {  
                        listaPronadjenihClanova.Add((Clan)odo);  
                    }  
                    return listaPronadjenihClanova;  
                }  
                else return null;  
            }  
            catch (Exception)  
            {  
                MessageBox.Show("Izgubljena je konekcija sa serverom! Zatvaram formu...");  
                klijent.Close();  
                glavnaForma.Povezan = false;  
                glavnaForma.Zatvori();  
                return null;  
            }  
        }  
  
        public bool ObrisiClana(Clan c)  
        {  
            try  
            {  
                TransferKlasa zahtev = new TransferKlasa()  
                {  
                    Operacija = Operacija.ObrisiClana,  
                    TransferObjekat = c  
                };  
                formater.Serialize(tok, zahtev);  
                TransferKlasa odgovor = formater.Deserialize(tok) as TransferKlasa;  
  
                return odgovor.Signal;  
            }  
            catch (Exception)  
            {  
                MessageBox.Show("Izgubljena je konekcija sa serverom! Zatvaram formu...");  
                klijent.Close();  
                glavnaForma.Povezan = false;  
                glavnaForma.Zatvori();  
                return false;  
            }  
        }  
  
        public int UbaciClana(Clan c)  
        {  
            try  
            {  
                TransferKlasa zahtev = new TransferKlasa()  
                {  
                    Operacija = Operacija.UbaciClana,  
                    TransferObjekat = c  
                };  
                formater.Serialize(tok, zahtev);  
                TransferKlasa odgovor = formater.Deserialize(tok) as TransferKlasa;  
  
                if (odgovor.Signal)  
                    return (int)odgovor.TransferObjekat;  
                else  
                    return -1;  
            }  
            catch (Exception)  
            {  
                MessageBox.Show("Izgubljena je konekcija sa serverom! Zatvaram formu...");  
                klijent.Close();  
                glavnaForma.Povezan = false;  
                glavnaForma.Zatvori();  
                return -1;  
            }  
        }  
  
        public bool SacuvajIzmeneKnjiga(Knjiga k)  
        {  
            try  
            {  
  
                TransferKlasa zahtev = new TransferKlasa()  
                {  
                    Operacija = Operacija.SacuvajIzmeneKnjiga,  
                    TransferObjekat = k  
                };  
                formater.Serialize(tok, zahtev);  
                TransferKlasa odgovor = formater.Deserialize(tok) as TransferKlasa;  
  
                return odgovor.Signal;  
            }  
            catch (Exception)  
            {  
                MessageBox.Show("Izgubljena je konekcija sa serverom! Zatvaram formu...");  
                klijent.Close();  
                glavnaForma.Povezan = false;  
                glavnaForma.Zatvori();  
                return false;  
            }  
        }  
  
        public List<Autor> VratiSveAutore()  
        {  
            try  
            {  
                List<Autor> listaAutora = new List<Autor>();  
                TransferKlasa zahtev = new TransferKlasa()  
                {  
                    Operacija = Operacija.VratiSveAutore  
                };  
                formater.Serialize(tok, zahtev);  
                TransferKlasa odgovor = formater.Deserialize(tok) as TransferKlasa;  
  
                if (odgovor.Signal)  
                {  
                    List<IOpstiDomenskiObjekat> lista = odgovor.TransferObjekat as List<IOpstiDomenskiObjekat>;  
                    foreach (IOpstiDomenskiObjekat odo in lista)  
                    {  
                        listaAutora.Add((Autor)odo);  
                    }  
                    return listaAutora;  
                }  
                else return null;  
            }  
            catch (Exception)  
            {  
                MessageBox.Show("Izgubljena je konekcija sa serverom! Zatvaram formu...");  
                klijent.Close();  
                glavnaForma.Povezan = false;  
                glavnaForma.Zatvori();  
                return null;  
            }  
        }  
  
        public List<Knjiga> VratiSveKnjige()  
        {  
            try  
            {  
                List<Knjiga> listaKnjiga = new List<Knjiga>();  
                TransferKlasa zahtev = new TransferKlasa()  
                {  
                    Operacija = Operacija.VratiSveKnjige  
                };  
                formater.Serialize(tok, zahtev);  
                TransferKlasa odgovor = formater.Deserialize(tok) as TransferKlasa;  
  
                if (odgovor.Signal)  
                {  
                    List<IOpstiDomenskiObjekat> lista = odgovor.TransferObjekat as List<IOpstiDomenskiObjekat>;  
                    foreach (IOpstiDomenskiObjekat odo in lista)  
                    {  
                        listaKnjiga.Add(odo as Knjiga);  
                    }  
                    return listaKnjiga;  
                }  
                else return null;  
            }  
            catch (Exception)  
            {  
                MessageBox.Show("Izgubljena je konekcija sa serverom! Zatvaram formu...");  
                klijent.Close();  
                glavnaForma.Povezan = false;  
                glavnaForma.Zatvori();  
                return null;  
            }  
        }  
  
        public bool ObrisiKnjigu(Knjiga k)  
        {  
            try  
            {  
                TransferKlasa zahtev = new TransferKlasa()  
                {  
                    Operacija = Operacija.ObrisiClana,  
                    TransferObjekat = k  
                };  
                formater.Serialize(tok, zahtev);  
                TransferKlasa odgovor = formater.Deserialize(tok) as TransferKlasa;  
  
                return odgovor.Signal;  
            }  
            catch (Exception)  
            {  
                MessageBox.Show("Izgubljena je konekcija sa serverom! Zatvaram formu...");  
                klijent.Close();  
                glavnaForma.Povezan = false;  
                glavnaForma.Zatvori();  
                return false;  
            }  
        }  
  
        public List<Knjiga> PretraziKnjige(string vrednost, KriterijumPretrage kriterijum)  
        {  
            try  
            {  
                List<Knjiga> listaPronadjenihKnjiga = new List<Knjiga>();  
                TransferKlasa zahtev = new TransferKlasa()  
                {  
                    Operacija = Operacija.PretraziKnjige,  
                    TransferObjekat = new Pretraga()  
                    {  
                        KriterijumPretrage = kriterijum,  
                        Vrednost = vrednost  
                    }  
                };  
                formater.Serialize(tok, zahtev);  
                TransferKlasa odgovor = formater.Deserialize(tok) as TransferKlasa;  
  
                if (odgovor.Signal)  
                {  
                    List<IOpstiDomenskiObjekat> lista = odgovor.TransferObjekat as List<IOpstiDomenskiObjekat>;  
                    foreach (IOpstiDomenskiObjekat odo in lista)  
                    {  
                        listaPronadjenihKnjiga.Add((Knjiga)odo);  
                    }  
                    return listaPronadjenihKnjiga;  
                }  
                else return null;  
            }  
            catch (Exception)  
            {  
                MessageBox.Show("Izgubljena je konekcija sa serverom! Zatvaram formu...");  
                klijent.Close();  
                glavnaForma.Povezan = false;  
                glavnaForma.Zatvori();  
                return null;  
            }  
        }  
  
        public int DajPrimerakID()  
        {  
            try  
            {  
                TransferKlasa zahtev = new TransferKlasa()  
                {  
                    Operacija = Operacija.DajPrimerakID  
                };  
                formater.Serialize(tok, zahtev);  
                TransferKlasa odgovor = formater.Deserialize(tok) as TransferKlasa;  
  
                if (odgovor.Signal)  
                    return (int)odgovor.TransferObjekat;  
                else return -1;  
            }  
            catch (Exception)  
            {  
                MessageBox.Show("Izgubljena je konekcija sa serverom! Zatvaram formu...");  
                klijent.Close();  
                glavnaForma.Povezan = false;  
                glavnaForma.Zatvori();  
                return -1;  
            }  
        }  
  
        public int UbaciKnjigu(Knjiga k)  
        {  
            try  
            {  
                TransferKlasa zahtev = new TransferKlasa()  
                {  
                    Operacija = Operacija.UbaciKnjigu,  
                    TransferObjekat = k  
                };  
                formater.Serialize(tok, zahtev);  
                TransferKlasa odgovor = formater.Deserialize(tok) as TransferKlasa;  
  
                if (odgovor.Signal)  
                    return Convert.ToInt32(odgovor.TransferObjekat);  
                else  
                    return -1;  
            }  
            catch (Exception)  
            {  
                MessageBox.Show("Izgubljena je konekcija sa serverom! Zatvaram formu...");  
                klijent.Close();  
                glavnaForma.Povezan = false;  
                glavnaForma.Zatvori();  
                return -1;  
            }  
        }  
  
        public Knjiga NadjiKnjigu(int knjigaID)  
        {  
            try  
            {  
                TransferKlasa zahtev = new TransferKlasa()  
                {  
                    Operacija = Operacija.NadjiKnjigu,  
                    TransferObjekat = knjigaID  
                };  
                formater.Serialize(tok, zahtev);  
                TransferKlasa odgovor = formater.Deserialize(tok) as TransferKlasa;  
  
                if (odgovor.Signal)  
                    return odgovor.TransferObjekat as Knjiga;  
                else return null;  
            }  
            catch (Exception)  
            {  
                MessageBox.Show("Izgubljena je konekcija sa serverom! Zatvaram formu...");  
                klijent.Close();  
                glavnaForma.Povezan = false;  
                glavnaForma.Zatvori();  
                return null;  
            }  
        }  
  
        public Zaduzenje NadjiZaduzenje(Clan c, KnjigaPrimerak kp)  
        {  
            try  
            {  
                Zaduzenje z = new Zaduzenje()  
                {  
                    Clan = c,  
                    KnjigaPrimerak = kp  
                };  
                TransferKlasa zahtev = new TransferKlasa()  
                {  
                    Operacija = Operacija.NadjiZaduzenje,  
                    TransferObjekat = z  
                };  
                formater.Serialize(tok, zahtev);  
                TransferKlasa odgovor = formater.Deserialize(tok) as TransferKlasa;  
  
                if (odgovor.Signal)  
                    return odgovor.TransferObjekat as Zaduzenje;  
                else return null;  
            }  
            catch (Exception)  
            {  
                MessageBox.Show("Izgubljena je konekcija sa serverom! Zatvaram formu...");  
                klijent.Close();  
                glavnaForma.Povezan = false;  
                glavnaForma.Zatvori();  
                return null;  
            }  
        }  
  
        public bool Zaduzi(Zaduzenje z)  
        {  
            try  
            {  
                TransferKlasa zahtev = new TransferKlasa()  
                {  
                    Operacija = Operacija.Zaduzi,  
                    TransferObjekat = z  
                };  
                formater.Serialize(tok, zahtev);  
                TransferKlasa odgovor = formater.Deserialize(tok) as TransferKlasa;  
  
                return odgovor.Signal;  
            }  
            catch (Exception)  
            {  
                MessageBox.Show("Izgubljena je konekcija sa serverom! Zatvaram formu...");  
                klijent.Close();  
                glavnaForma.Povezan = false;  
                glavnaForma.Zatvori();  
                return false;  
            }}  
    }

# 4.2 Пројектовање апликационе логике

Апликациона логика описује структуру и понашање система. Сваки апликациони сервер се састоји од три дела:

1. Контролер апликационе логике – служи за комуникацију са клијентом и одговоран је да прихвати захтев за извршење системске операције од клијента и проследи га до пословне логике која је одговорна за извршење СО
2. Брокер базе података – служи за комуникацију између пословне логике и базе података
3. Пословна логика – описана је структуром (доменске класе) и понашањем (системске операције)

## 4.2.1 Контролер апликационе логике

Софтверски систем реализован је као клијент-сервер апликација. Слање и примање података од клијента се остварује преко сокета. Клијент шаље захтев за извршење неке од СО до одговарајуће нити (коју смо назвали “КлијентНит”), која је повезана са тим клијентом. “КлијентНит” прима захтев и даље га преусмерава до класа које су одговорне за извршење СО. Након извршења СО резултат се враћа до апликационе логике, односно до “КлијентНит”, која тај резултат шаље назад до клијента.

## 4.2.2 Пословна логика

### Пројектовање понашања (системске операције)

За сваки од претходно дефинисаних уговора, правимо системску операцију. Дефинишемо апстрактну класу Општа системска операција коју ће свака од конкретних СО да наследи и имплементира потребне методе.

public abstract class OpstaSistemskaOperacija  
    {  
        public object Rezultat { get; set; }  
  
        public bool IzvrsiSO(IOpstiDomenskiObjekat odo)  
        {  
            bool rezultat = false;  
  
            GenerickiBroker.Instanca.OtvoriKonekciju();  
            GenerickiBroker.Instanca.PokreniTransakciju();  
            rezultat = Izvrsi(odo);  
            if (rezultat)  
            {  
                GenerickiBroker.Instanca.Commit();  
            }  
            else  
            {  
                GenerickiBroker.Instanca.Rollback();  
            }  
            GenerickiBroker.Instanca.ZatvoriKonekciju();  
            return rezultat;  
        }  
  
        protected abstract bool Izvrsi(IOpstiDomenskiObjekat odo);  
  
    }

**Уговор УГ1: KreirajClana(Clan)**

**Операција:** KreirajClana(Clan): signal;

**Веза са СК:** СК1

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

**Постуслови:** Креиран је нови члан.

public class UbaciClanaSO : OpstaSistemskaOperacija  
    {  
        protected override bool Izvrsi(IOpstiDomenskiObjekat odo)  
        {  
            try  
            {  
                Clan c = odo as Clan;  
                //c.ClanskiBroj = GenerickiBroker.Instanca.DajClanskiBroj();  
                c.ClanskiBroj = GenerickiBroker.Instanca.SelectMax(c);  
  
                int rez = GenerickiBroker.Instanca.Insert(c);  
  
                if (rez > 0)  
                {  
                    return true;  
                }  
                else  
                {  
                    return false;  
                }  
  
            }  
            catch (Exception e)  
            {  
                return false;  
            }  
        }  
    }



Слика 108 - Дијаграм секвенцe ”UbaciClanaSO”

Уговор УГ2: PretrazivanjeClanova(kriterijum, List<Clan>)

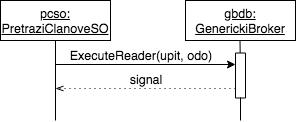
**Операција:** PretrazivanjeClanova(kriterijum*,* List<Clan>): signal;

**Веза са СК:** СК2, СК3, СК4

**Предуслови:** /

**Постуслови:** /

public class PretraziClanoveSO : OpstaSistemskaOperacija  
    {  
        public Pretraga Pretraga { get; set; }  
  
        public PretraziClanoveSO(Pretraga p)  
        {  
            this.Pretraga = p;  
        }  
  
        protected override bool Izvrsi(IOpstiDomenskiObjekat odo)  
        {  
            try  
            {  
                string upit = "";  
  
                switch (Pretraga.KriterijumPretrage)  
                {  
                    case KriterijumPretrage.ImePrezimeClan:  
                        upit = $"Select \* from Clan where ImePrezime like '%{Pretraga.Vrednost}%' order by ImePrezime";  
                        break;  
                    case KriterijumPretrage.ClanskiBroj:  
                        upit = $"Select \* from Clan where ClanskiBroj = {Convert.ToInt32(Pretraga.Vrednost)} order by ImePrezime";  
                        break;  
                }  
  
                List<IOpstiDomenskiObjekat> listaPronadjeniClanovi = GenerickiBroker.Instanca.ExecuteReader(upit, odo);  
  
                foreach (IOpstiDomenskiObjekat o in listaPronadjeniClanovi)  
                {  
                    Clan c = o as Clan;  
                    upit = $"Select \* from Zaduzenje where ClanskiBroj = {c.ClanskiBroj}";  
  
                    List<IOpstiDomenskiObjekat> listaZaduzenja = GenerickiBroker.Instanca.ExecuteReader(upit, new Zaduzenje());  
  
                    foreach (IOpstiDomenskiObjekat zaduzenje in listaZaduzenja)  
                    {  
                        Zaduzenje z = zaduzenje as Zaduzenje;  
                        z.Clan = c;  
  
                        upit = $"Select \* from KnjigaPrimerak where PrimerakID = {z.KnjigaPrimerak.PrimerakID} and KnjigaID = {z.KnjigaPrimerak.Knjiga.KnjigaID}";  
                        List<IOpstiDomenskiObjekat> listaPrimeraka = GenerickiBroker.Instanca.ExecuteReader(upit, new KnjigaPrimerak());  
  
                        foreach (IOpstiDomenskiObjekat prim in listaPrimeraka)  
                        {  
                            KnjigaPrimerak kp = prim as KnjigaPrimerak;  
  
                            upit = $"Select \* from Knjiga where KnjigaID = {kp.Knjiga.KnjigaID}";  
                            List<IOpstiDomenskiObjekat> knjiga = GenerickiBroker.Instanca.ExecuteReader(upit, kp.Knjiga);  
                            kp.Knjiga = knjiga[0] as Knjiga;  
  
                            upit = $"Select \* from KnjigaPrimerak where KnjigaID = {kp.Knjiga.KnjigaID}";  
                            List<IOpstiDomenskiObjekat> listaPrimerakaKnjiga = GenerickiBroker.Instanca.ExecuteReader(upit, new KnjigaPrimerak());  
  
                            foreach (IOpstiDomenskiObjekat lp in listaPrimerakaKnjiga)  
                            {  
                                kp.Knjiga.SpisakPrimeraka.Add(lp as KnjigaPrimerak);  
                            }  
  
                            upit = $"Select \* from KnjigaAutor where KnjigaID = {kp.Knjiga.KnjigaID}";  
                            List<IOpstiDomenskiObjekat> listaKnjigaAutor = GenerickiBroker.Instanca.ExecuteReader(upit, new KnjigaAutor());  
  
                            foreach (IOpstiDomenskiObjekat ka in listaKnjigaAutor)  
                            {  
                                KnjigaAutor kaa = ka as KnjigaAutor;  
  
                                upit = $"Select \* from Autor where AutorID = {kaa.AutorID}";  
  
                                List<IOpstiDomenskiObjekat> autor = GenerickiBroker.Instanca.ExecuteReader(upit, new Autor());  
                                kp.Knjiga.ListaAutora.Add(autor[0] as Autor);  
                            }  
  
                            z.KnjigaPrimerak = kp;  
                        }  
  
                        c.ListaZaduzenja.Add(z);  
                    }  
  
                }  
  
                Rezultat = listaPronadjeniClanovi;  
  
                return true;  
            }  
            catch (Exception)  
            {  
                return false;  
            }  
        }  
    }



Слика 109 - Дијаграм секвенцe ”PretraziClanoveSO”

Уговор УГ3: PronadjiClana(Clan, List<Clan>)

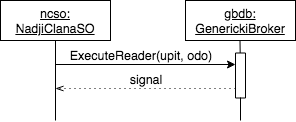
**Операција:** PronadjiClana(Clan*,* List<Clan>): signal;

**Веза са СК:** СК2, СК3, СК4

**Предуслови:** /

**Постуслови:** /

public class NadjiClanaSO : OpstaSistemskaOperacija  
    {  
        protected override bool Izvrsi(IOpstiDomenskiObjekat odo)  
        {  
            try  
            {  
                Clan clan = odo as Clan;  
                string upit = $"Select \* from Clan where ClanskiBroj = {clan.ClanskiBroj}";  
  
                List<IOpstiDomenskiObjekat> clanovi = GenerickiBroker.Instanca.ExecuteReader(upit, odo);  
  
                if (clanovi.Count == 0) return false;  
                 
                Clan c = clanovi[0] as Clan;  
                upit = $"Select \* from Zaduzenje where ClanskiBroj = {c.ClanskiBroj}";  
  
                List<IOpstiDomenskiObjekat> listaZaduzenja = GenerickiBroker.Instanca.ExecuteReader(upit, new Zaduzenje());  
  
                foreach (IOpstiDomenskiObjekat zaduzenje in listaZaduzenja)  
                {  
                    Zaduzenje z = zaduzenje as Zaduzenje;  
                    z.Clan = c;  
  
                    upit = $"Select \* from KnjigaPrimerak where PrimerakID = {z.KnjigaPrimerak.PrimerakID} and KnjigaID = {z.KnjigaPrimerak.Knjiga.KnjigaID}";  
                    List<IOpstiDomenskiObjekat> listaPrimeraka = GenerickiBroker.Instanca.ExecuteReader(upit, new KnjigaPrimerak());  
  
                    foreach (IOpstiDomenskiObjekat prim in listaPrimeraka)  
                    {  
                        KnjigaPrimerak kp = prim as KnjigaPrimerak;  
  
                        upit = $"Select \* from Knjiga where KnjigaID = {kp.Knjiga.KnjigaID}";  
                        List<IOpstiDomenskiObjekat> knjiga = GenerickiBroker.Instanca.ExecuteReader(upit, kp.Knjiga);  
                        kp.Knjiga = knjiga[0] as Knjiga;  
  
                        upit = $"Select \* from KnjigaPrimerak where KnjigaID = {kp.Knjiga.KnjigaID}";  
                        List<IOpstiDomenskiObjekat> listaPrimerakaKnjiga = GenerickiBroker.Instanca.ExecuteReader(upit, new KnjigaPrimerak());  
  
                        foreach (IOpstiDomenskiObjekat lp in listaPrimerakaKnjiga)  
                        {  
                            kp.Knjiga.SpisakPrimeraka.Add(lp as KnjigaPrimerak);  
                        }  
  
                        upit = $"Select \* from KnjigaAutor where KnjigaID = {kp.Knjiga.KnjigaID}";  
                        List<IOpstiDomenskiObjekat> listaKnjigaAutor = GenerickiBroker.Instanca.ExecuteReader(upit, new KnjigaAutor());  
  
                        foreach (IOpstiDomenskiObjekat ka in listaKnjigaAutor)  
                        {  
                            KnjigaAutor kaa = ka as KnjigaAutor;  
  
                            upit = $"Select \* from Autor where AutorID = {kaa.AutorID}";  
  
                            List<IOpstiDomenskiObjekat> autor = GenerickiBroker.Instanca.ExecuteReader(upit, new Autor());  
                            kp.Knjiga.ListaAutora.Add(autor[0] as Autor);  
                        }  
  
                        z.KnjigaPrimerak = kp;  
                    }  
  
                    c.ListaZaduzenja.Add(z);  
                }  
  
                Rezultat = c;  
  
                return true;  
            }  
            catch (Exception e)  
            {  
                return false;  
            }  
        }  
    }



Слика 110 - Дијаграм секвенцe ”NadjiClanaSO”

Уговор УГ4: PromeniClana(Clan)

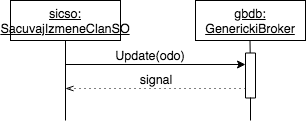
**Операција:** PromeniClana(Clan): signal;

**Веза са СК:** СК3

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

**Постуслови:** Унети подаци о члану су промењени.

public class SacuvajIzmeneClanSO : OpstaSistemskaOperacija  
    {  
        protected override bool Izvrsi(IOpstiDomenskiObjekat odo)  
        {  
            try  
            {  
                GenerickiBroker.Instanca.Update(odo);  
  
                return true;  
            }  
            catch (Exception e)  
            {  
                Console.WriteLine(e.Message);  
                return false;  
            }  
        }  
    }



Слика 111 - Дијаграм секвенцe ”SacuvajIzmeneClanSO”

Уговор УГ5: ObrisiClana(Clan)

**Операција:** ObrisiClana(Clan): signal;

**Веза са СК:** СК4

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

Постуслови: Члан је обрисан.

public class ObrisiClanaSO : OpstaSistemskaOperacija  
    {  
        protected override bool Izvrsi(IOpstiDomenskiObjekat odo)  
        {  
            try  
            {  
                GenerickiBroker.Instanca.Delete(odo);  
  
                return true;  
            }  
            catch (Exception)  
            {  
                return false;  
            }  
        }  
    }



Слика 112 - Дијаграм секвенцe ”ObrisiClanaSO”

Уговор УГ6: KreirajKnjigu(Knjiga)

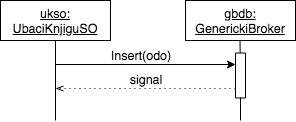
**Операција:** KreirajKnjigu(Knjiga): signal;

**Веза са СК:** СК5

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

**Постуслови:** Креирана је нова књига.

public class UbaciKnjiguSO : OpstaSistemskaOperacija  
    {  
        protected override bool Izvrsi(IOpstiDomenskiObjekat odo)  
        {  
            try  
            {  
                Knjiga k = odo as Knjiga;  
                //k.KnjigaID = GenerickiBroker.Instanca.DajKnjigaID();  
                k.KnjigaID = GenerickiBroker.Instanca.SelectMax(k);  
  
                int rez = GenerickiBroker.Instanca.Insert(k);  
  
                if (rez > 0)  
                {  
                    foreach (KnjigaPrimerak kp in k.SpisakPrimeraka)  
                    {  
                        kp.Knjiga = k;  
                        GenerickiBroker.Instanca.Insert(kp);  
                    }  
  
                    foreach(Autor a in k.ListaAutora)  
                    {  
                        KnjigaAutor ka = new KnjigaAutor()  
                        {  
                            AutorID = a.AutorID,  
                            KnjigaID = k.KnjigaID  
                        };  
                        GenerickiBroker.Instanca.Insert(ka);  
                    }  
                }  
                else return false;  
  
                return true;  
            }  
            catch (Exception)  
            {  
                return false;  
            }  
        }  
    }



Слика 113 - Дијаграм секвенцe ”UbaciKnjiguSO”

Уговор УГ7: PretrazivanjeKnjiga(kriterijum, List<Knjiga>)

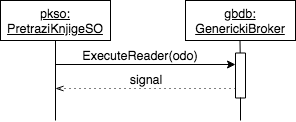
**Операција:** PretrazivanjeKnjiga(kriterijum*,* List<Knjiga>): signal;

**Веза са СК:** СК6, СК7, СК8

**Предуслови:** /

**Постуслови:** /

public class PretraziKnjigeSO : OpstaSistemskaOperacija  
    {  
        public Pretraga Pretraga { get; set; }  
  
        public PretraziKnjigeSO(Pretraga p)  
        {  
            this.Pretraga = p;  
        }  
  
        protected override bool Izvrsi(IOpstiDomenskiObjekat odo)  
        {  
            try  
            {  
                string upit = "";  
                List<IOpstiDomenskiObjekat> listaKnjiga = new List<IOpstiDomenskiObjekat>();  
  
                switch (Pretraga.KriterijumPretrage)  
                {  
                    case KriterijumPretrage.NazivKnjige:  
                        upit = $"Select \* from Knjiga where Naziv like '%{Pretraga.Vrednost}%' order by Naziv ASC";  
                        break;  
                    case KriterijumPretrage.ImePrezimeAutor:  
                        return PretraziPoAutoru();  
                    case KriterijumPretrage.BrojKnjige:  
                        upit = $"Select \* from Knjiga where KnjigaID = {Convert.ToInt32(Pretraga.Vrednost)} order by Naziv ASC";  
                        break;  
                }  
  
                listaKnjiga = GenerickiBroker.Instanca.ExecuteReader(upit, odo);  
  
                foreach(IOpstiDomenskiObjekat o in listaKnjiga)  
                {  
                    Knjiga k = o as Knjiga;  
  
                    upit = $"Select \* from KnjigaPrimerak where KnjigaPrimerak.KnjigaID = {k.KnjigaID}";  
                    List<IOpstiDomenskiObjekat> listaPrimeraka = GenerickiBroker.Instanca.ExecuteReader(upit, new KnjigaPrimerak());  
  
                    foreach(IOpstiDomenskiObjekat kp in listaPrimeraka)  
                    {  
                        KnjigaPrimerak knjP = kp as KnjigaPrimerak;  
                        knjP.Knjiga = k;  
  
                        k.SpisakPrimeraka.Add(knjP);  
                    }  
  
                    upit = $"Select \* from KnjigaAutor where KnjigaID = {k.KnjigaID}";  
                    List<IOpstiDomenskiObjekat> listaKnjigaAutor = GenerickiBroker.Instanca.ExecuteReader(upit, new KnjigaAutor());  
  
                    foreach(IOpstiDomenskiObjekat ka in listaKnjigaAutor)  
                    {  
                        KnjigaAutor kaa = ka as KnjigaAutor;  
  
                        upit = $"Select \* from Autor where AutorID = {kaa.AutorID}";  
  
                        List<IOpstiDomenskiObjekat> autor = GenerickiBroker.Instanca.ExecuteReader(upit, new Autor());  
                        k.ListaAutora.Add(autor[0] as Autor);  
                    }  
                }  
  
                Rezultat = listaKnjiga;  
                return true;  
            }  
            catch (Exception e)  
            {  
                Console.WriteLine(e.Message);  
                return false;  
            }  
        }  
          
        private bool PretraziPoAutoru()  
        {  
            try  
            {  
                List<IOpstiDomenskiObjekat> listaPronadjenihKnjiga = new List<IOpstiDomenskiObjekat>();  
  
                List<IOpstiDomenskiObjekat> listaSvihKnjiga = GenerickiBroker.Instanca.SelectAll(new Knjiga());  
                  
                foreach(IOpstiDomenskiObjekat odo in listaSvihKnjiga)  
                {  
                    Knjiga k = odo as Knjiga;  
                      
                    string upit = $"Select \* from KnjigaAutor where KnjigaID = {k.KnjigaID}";  
                    List<IOpstiDomenskiObjekat> listaKnjigaAutor = GenerickiBroker.Instanca.ExecuteReader(upit, new KnjigaAutor());  
  
                    foreach (IOpstiDomenskiObjekat ka in listaKnjigaAutor)  
                    {  
                        KnjigaAutor kaa = ka as KnjigaAutor;  
  
                        upit = $"Select \* from Autor where AutorID = {kaa.AutorID}";  
  
                        List<IOpstiDomenskiObjekat> autor = GenerickiBroker.Instanca.ExecuteReader(upit, new Autor());  
                        k.ListaAutora.Add(autor[0] as Autor);  
                    }  
  
                    upit = $"Select \* from KnjigaPrimerak where KnjigaPrimerak.KnjigaID = {k.KnjigaID}";  
                    List<IOpstiDomenskiObjekat> listaPrimeraka = GenerickiBroker.Instanca.ExecuteReader(upit, new KnjigaPrimerak());  
  
                    foreach (IOpstiDomenskiObjekat lp in listaPrimeraka)  
                    {  
                        k.SpisakPrimeraka.Add(lp as KnjigaPrimerak);  
                    }  
  
                    foreach (Autor a in k.ListaAutora)  
                    {  
                        if (a.ImePrezime.ToLower().Contains(Pretraga.Vrednost.ToLower()))  
                        {  
                            listaPronadjenihKnjiga.Add(k);  
                            break;  
                        }  
                    }  
                }  
                Rezultat = listaPronadjenihKnjiga;  
                return true;  
            }  
            catch (Exception e)  
            {  
                return false;  
            }  
        }  
    }



Слика 114 - Дијаграм секвенцe ”PretraziKnjigeSO”

Уговор УГ8: PronadjiKnjigu(Knjiga, List<Knjiga>)

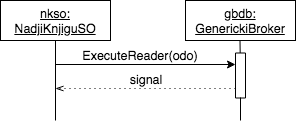
**Операција:** PronadjiKnjigu(Knjiga*,* List<Knjiga>): signal;

**Веза са СК:** СК6, СК7, СК8

**Предуслови:** /

**Постуслови:** /

public class NadjiKnjiguSO : OpstaSistemskaOperacija  
    {  
        protected override bool Izvrsi(IOpstiDomenskiObjekat odo)  
        {  
            try  
            {  
                Knjiga knjiga = odo as Knjiga;  
  
                string upit = $"Select \* from Knjiga where KnjigaID = {knjiga.KnjigaID} order by Naziv ASC";  
                List<IOpstiDomenskiObjekat> listaKnjiga = GenerickiBroker.Instanca.ExecuteReader(upit, odo);  
  
                foreach(IOpstiDomenskiObjekat o in listaKnjiga)  
                {  
                    Knjiga k = o as Knjiga;  
                    upit = $"Select \* from KnjigaPrimerak where KnjigaPrimerak.KnjigaID = {k.KnjigaID}";  
  
                    List<IOpstiDomenskiObjekat> listaPrimeraka = GenerickiBroker.Instanca.ExecuteReader(upit, new KnjigaPrimerak());  
                    foreach (IOpstiDomenskiObjekat lp in listaPrimeraka)  
                    {  
                        k.SpisakPrimeraka.Add(lp as KnjigaPrimerak);  
                    }  
  
                    upit = $"Select \* from KnjigaAutor where KnjigaID = {k.KnjigaID}";  
                    List<IOpstiDomenskiObjekat> listaKnjigaAutor = GenerickiBroker.Instanca.ExecuteReader(upit, new KnjigaAutor());  
  
                    foreach (IOpstiDomenskiObjekat ka in listaKnjigaAutor)  
                    {  
                        KnjigaAutor kaa = ka as KnjigaAutor;  
  
                        upit = $"Select \* from Autor where AutorID = {kaa.AutorID}";  
  
                        List<IOpstiDomenskiObjekat> autor = GenerickiBroker.Instanca.ExecuteReader(upit, new Autor());  
                        k.ListaAutora.Add(autor[0] as Autor);  
                    }  
                }  
  
                if (listaKnjiga.Count == 0) return false;  
  
                Rezultat = listaKnjiga[0];  
  
                return true;  
            }  
            catch (Exception)  
            {  
                return false;  
            }  
        }}



Слика 115 - Дијаграм секвенцe ”NadjiKnjiguSO”

Уговор УГ9: PromeniKnjigu(Knjiga)

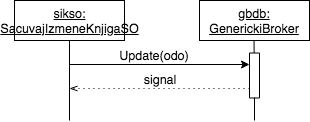
**Операција:** PromeniKnjigu(Knjiga): signal;

**Веза са СК:** СК7

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

**Постуслови:** Унети подаци о књизи су промењени.

public class SacuvajIzmeneKnjigaSO : OpstaSistemskaOperacija  
    {  
        protected override bool Izvrsi(IOpstiDomenskiObjekat odo)  
        {  
            try  
            {  
                Knjiga novo = odo as Knjiga;  
                GenerickiBroker.Instanca.Update(novo);  
  
                string upit = $"Delete from KnjigaAutor where KnjigaID = {novo.KnjigaID}";  
                GenerickiBroker.Instanca.ExecuteNonQuery(upit);  
  
                foreach(Autor a in novo.ListaAutora)  
                {  
                    upit = $"Insert into KnjigaAutor values ({novo.KnjigaID}, {a.AutorID})";  
                    GenerickiBroker.Instanca.ExecuteNonQuery(upit);  
                }  
  
                return true;  
            }  
            catch (Exception)  
            {  
  
                return false;  
            }  
        }  
    }



Слика 116 - Дијаграм секвенцe ”SacuvajIzmeneKnjigaSO”

Уговор УГ10: ObrisiKnjigu(Knjiga)

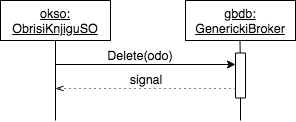
**Операција:** ObrisiKnjigu(Knjiga): signal;

**Веза са СК:** СК8

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

**Постуслови:** Књига је обрисана.

public class ObrisiKnjiguSO : OpstaSistemskaOperacija  
    {  
        protected override bool Izvrsi(IOpstiDomenskiObjekat odo)  
        {  
            try  
            {  
                GenerickiBroker.Instanca.Delete(odo);  
  
                return true;  
            }  
            catch (Exception)  
            {  
                return false;  
            }  
        }  
    }



Слика 117 - Дијаграм секвенцe ”ObrisiKnjiguSO”

Уговор УГ11: UcitajKnjige()

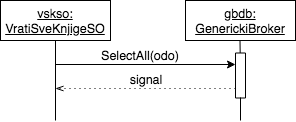
**Операција:** UcitajKnjige(): signal;

**Веза са СК:** СК9

**Предуслови:** /

**Постуслови:** /

public class VratiSveKnjigeSO : OpstaSistemskaOperacija  
    {  
        protected override bool Izvrsi(IOpstiDomenskiObjekat odo)  
        {  
            try  
            {  
                List<IOpstiDomenskiObjekat> listaKnjiga = GenerickiBroker.Instanca.SelectAll(odo);  
                string upit;  
  
                foreach(IOpstiDomenskiObjekat knjiga in listaKnjiga)  
                {  
                    Knjiga k = knjiga as Knjiga;  
  
                    upit = $"Select \* from KnjigaPrimerak where KnjigaPrimerak.KnjigaID = {k.KnjigaID}";  
                    List<IOpstiDomenskiObjekat> listaPrimeraka = GenerickiBroker.Instanca.ExecuteReader(upit, new KnjigaPrimerak());  
  
                    foreach(IOpstiDomenskiObjekat lp in listaPrimeraka)  
                    {  
                        k.SpisakPrimeraka.Add(lp as KnjigaPrimerak);  
                    }  
  
                    upit = $"Select \* from KnjigaAutor where KnjigaID = {k.KnjigaID}";  
                    List<IOpstiDomenskiObjekat> listaKnjigaAutor = GenerickiBroker.Instanca.ExecuteReader(upit, new KnjigaAutor());  
  
                    foreach (IOpstiDomenskiObjekat ka in listaKnjigaAutor)  
                    {  
                        KnjigaAutor kaa = ka as KnjigaAutor;  
  
                        upit = $"Select \* from Autor where AutorID = {kaa.AutorID}";  
  
                        List<IOpstiDomenskiObjekat> autor = GenerickiBroker.Instanca.ExecuteReader(upit, new Autor());  
                        k.ListaAutora.Add(autor[0] as Autor);  
                    }  
  
                }  
                Rezultat = listaKnjiga;  
  
                return true;  
            }  
            catch (Exception e)  
            {  
                Console.WriteLine(e.Message);  
                return false;  
            }  
        }  
    }



Слика 118 - Дијаграм секвенцe ”VratiSveKnjigeSO”

Уговор УГ12: UcitajClanoveBiblioteke()

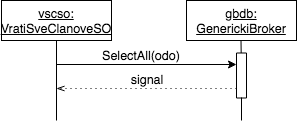
**Операција:** UcitajClanoveBiblioteke(): signal;

**Веза са СК:** СК9

**Предуслови:** /

**Постуслови:** /

public class VratiSveClanoveSO : OpstaSistemskaOperacija  
    {  
        protected override bool Izvrsi(IOpstiDomenskiObjekat odo)  
        {  
            try  
            {  
                List<IOpstiDomenskiObjekat> listaClanova = GenerickiBroker.Instanca.SelectAll(odo);  
                foreach (IOpstiDomenskiObjekat o in listaClanova)  
                {  
                    Clan c = o as Clan;  
                    string upit = $"Select \* from Zaduzenje where ClanskiBroj = {c.ClanskiBroj}";  
  
                    List<IOpstiDomenskiObjekat> listaZaduzenja = GenerickiBroker.Instanca.ExecuteReader(upit, new Zaduzenje());  
  
                    foreach (IOpstiDomenskiObjekat zaduzenje in listaZaduzenja)  
                    {  
                        Zaduzenje z = zaduzenje as Zaduzenje;  
                        z.Clan = c;  
  
                        upit = $"Select \* from KnjigaPrimerak where PrimerakID = {z.KnjigaPrimerak.PrimerakID} and KnjigaID = {z.KnjigaPrimerak.Knjiga.KnjigaID}";  
                        List<IOpstiDomenskiObjekat> listaPrimeraka = GenerickiBroker.Instanca.ExecuteReader(upit, new KnjigaPrimerak());  
  
                        foreach (IOpstiDomenskiObjekat prim in listaPrimeraka)  
                        {  
                            KnjigaPrimerak kp = prim as KnjigaPrimerak;  
  
                            upit = $"Select \* from Knjiga where KnjigaID = {kp.Knjiga.KnjigaID}";  
                            List<IOpstiDomenskiObjekat> knjiga = GenerickiBroker.Instanca.ExecuteReader(upit, kp.Knjiga);  
                            kp.Knjiga = knjiga[0] as Knjiga;  
  
                            upit = $"Select \* from KnjigaPrimerak where KnjigaID = {kp.Knjiga.KnjigaID}";  
                            List<IOpstiDomenskiObjekat> listaPrimerakaKnjiga = GenerickiBroker.Instanca.ExecuteReader(upit, new KnjigaPrimerak());  
  
                            foreach (IOpstiDomenskiObjekat lp in listaPrimerakaKnjiga)  
                            {  
                                kp.Knjiga.SpisakPrimeraka.Add(lp as KnjigaPrimerak);  
                            }  
  
                            upit = $"Select \* from KnjigaAutor where KnjigaID = {kp.Knjiga.KnjigaID}";  
                            List<IOpstiDomenskiObjekat> listaKnjigaAutor = GenerickiBroker.Instanca.ExecuteReader(upit, new KnjigaAutor());  
  
                            foreach (IOpstiDomenskiObjekat ka in listaKnjigaAutor)  
                            {  
                                KnjigaAutor kaa = ka as KnjigaAutor;  
  
                                upit = $"Select \* from Autor where AutorID = {kaa.AutorID}";  
  
                                List<IOpstiDomenskiObjekat> autor = GenerickiBroker.Instanca.ExecuteReader(upit, new Autor());  
                                kp.Knjiga.ListaAutora.Add(autor[0] as Autor);  
                            }  
                              
                            z.KnjigaPrimerak = kp;  
                        }  
  
                        c.ListaZaduzenja.Add(z);  
                    }  
  
                }  
                Rezultat = listaClanova;  
  
                return true;  
            }  
            catch (Exception e)  
            {  
                Console.WriteLine(e.Message);  
                return false;  
            }  
        }  
    }



Слика 119 - Дијаграм секвенцe ”VratiSveClanoveSO”

Уговор УГ13: KreirajZaduzenje(Zaduzenje)

**Операција:** KreirajZaduzenje(Zaduzenje): signal;

**Веза са СК:** СК9

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

**Постуслови:** Креиранo је новo задужење.

public class ZaduziSO : OpstaSistemskaOperacija  
    {  
        protected override bool Izvrsi(IOpstiDomenskiObjekat odo)  
        {  
            try  
            {  
                Zaduzenje z = odo as Zaduzenje;  
                GenerickiBroker.Instanca.Insert(z);  
  
                string upit = $"update KnjigaPrimerak set Raspoloziva = false where KnjigaID = {z.KnjigaPrimerak.Knjiga.KnjigaID} and PrimerakID = {z.KnjigaPrimerak.PrimerakID}";  
                GenerickiBroker.Instanca.ExecuteNonQuery(upit);  
  
                upit = $"Update Knjiga set Raspolozivo = {z.KnjigaPrimerak.Knjiga.Raspolozivo - 1} where KnjigaID = {z.KnjigaPrimerak.Knjiga.KnjigaID}";  
                GenerickiBroker.Instanca.ExecuteNonQuery(upit);  
  
                return true;  
            }  
            catch (Exception)  
            {  
                return false;  
            }  
        }  
    }



Слика 120 - Дијаграм секвенцe ”ZaduziSO”

Уговор УГ15: PronadjiZaduzenje(Zaduzenje, List<Zaduzenje>)

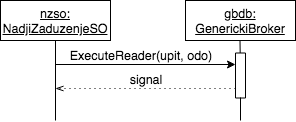
**Операција:** PronadjiZaduzenje(Zaduzenje*,* List<Zaduzenje>): signal;

**Веза са СК:** СК10

**Предуслови:** /

**Постуслови:** /

public class NadjiZaduzenjeSO : OpstaSistemskaOperacija  
    {  
        protected override bool Izvrsi(IOpstiDomenskiObjekat odo)  
        {  
            try  
            {  
                Zaduzenje zaduzenje = odo as Zaduzenje;  
  
                string upit = $"Select \* from Zaduzenje where ClanskiBroj = {zaduzenje.Clan.ClanskiBroj} and" +  
                $" PrimerakID = {zaduzenje.KnjigaPrimerak.PrimerakID} and KnjigaID = {zaduzenje.KnjigaPrimerak.Knjiga.KnjigaID}" +  
                $" and DatumDo is NULL";  
                List<IOpstiDomenskiObjekat> listaZaduzenja = GenerickiBroker.Instanca.ExecuteReader(upit, zaduzenje);  
  
                if (listaZaduzenja.Count == 0) return false;  
                Zaduzenje z = listaZaduzenja[0] as Zaduzenje;  
  
                upit = $"Select \* from Clan where ClanskiBroj = {z.Clan.ClanskiBroj}";  
                List<IOpstiDomenskiObjekat> listaClanova = GenerickiBroker.Instanca.ExecuteReader(upit, new Clan());  
                if (listaClanova.Count == 0) return false;  
                z.Clan = listaClanova[0] as Clan;  
  
                upit = $"Select \* from KnjigaPrimerak where PrimerakID = {z.KnjigaPrimerak.PrimerakID} and KnjigaID = {z.KnjigaPrimerak.Knjiga.KnjigaID}";  
                List<IOpstiDomenskiObjekat> listaPrimeraka = GenerickiBroker.Instanca.ExecuteReader(upit, new KnjigaPrimerak());  
                if (listaPrimeraka.Count == 0) return false;  
                z.KnjigaPrimerak = listaPrimeraka[0] as KnjigaPrimerak;  
                  
                Rezultat = z;  
  
                return true;  
            }  
            catch (Exception)  
            {  
                return false;  
            }  
        }  
    }



Слика 121 - Дијаграм секвенцe ”NadjiZaduzenjeSO”

Уговор УГ16: PromeniZaduzenje(Zaduzenje)

**Операција:** PromeniZaduzenje(Zaduzenje): signal;

**Веза са СК:** СК10

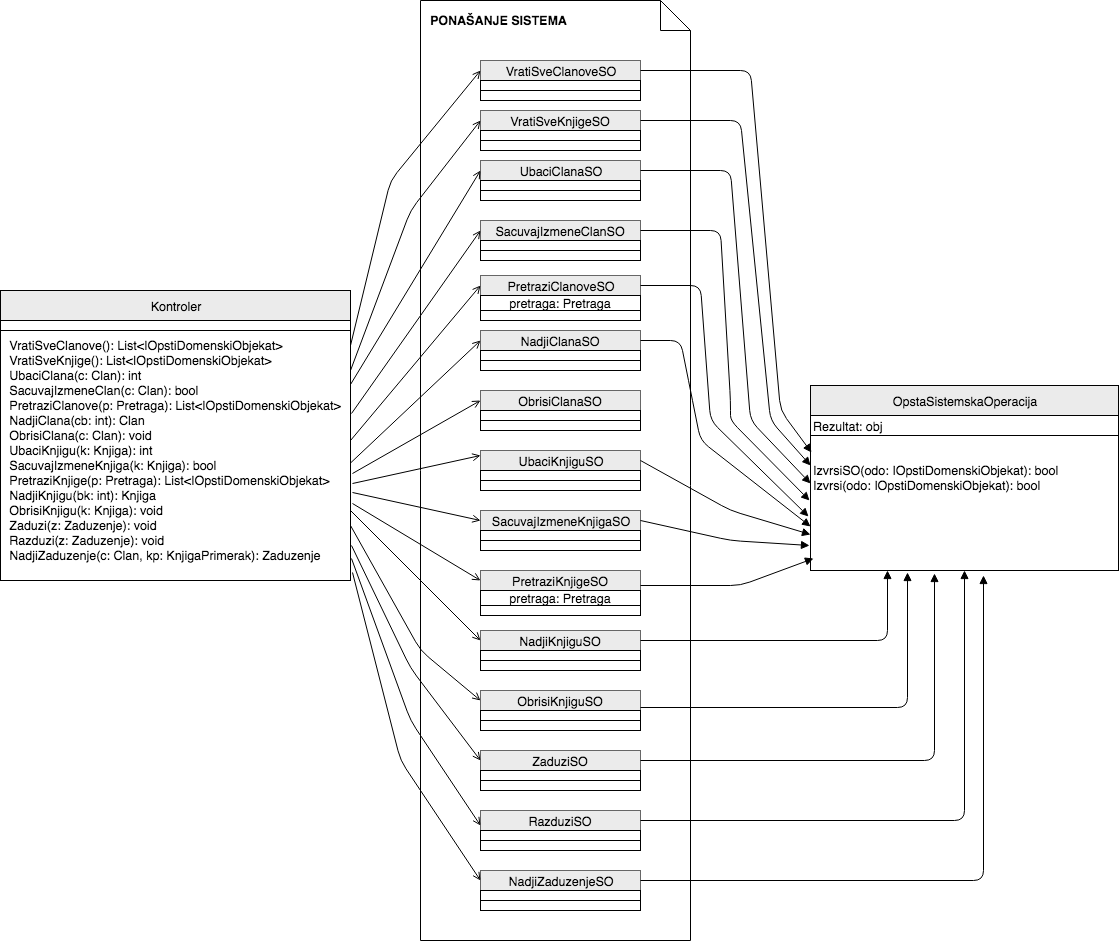
**Предуслови:** Вредносна и структурна ограничења над објектом *Zaduzenje* морају бити задовољена.

**Постуслови:** Унети подаци о задужењу су промењени.

public class RazduziSO : OpstaSistemskaOperacija  
    {  
        protected override bool Izvrsi(IOpstiDomenskiObjekat odo)  
        {  
            try  
            {  
                Zaduzenje z = odo as Zaduzenje;  
  
                string upit = $"Update Zaduzenje set DatumDo = '{DateTime.Now.ToString("dd-MMM-yyyy")}' where ClanskiBroj = {z.Clan.ClanskiBroj} and " +  
                $"PrimerakID = {z.KnjigaPrimerak.PrimerakID} and KnjigaID = {z.KnjigaPrimerak.Knjiga.KnjigaID} and DatumOd = #{z.DatumOd.ToString("dd-MMM-yyyy")}#";  
                GenerickiBroker.Instanca.ExecuteNonQuery(upit);  
  
                upit = $"Update KnjigaPrimerak set Raspoloziva = true where PrimerakID = {z.KnjigaPrimerak.PrimerakID}";  
                GenerickiBroker.Instanca.ExecuteNonQuery(upit);  
  
                upit = $"Update Knjiga set Raspolozivo = {z.KnjigaPrimerak.Knjiga.Raspolozivo + 1} where KnjigaID = {z.KnjigaPrimerak.Knjiga.KnjigaID}";  
                GenerickiBroker.Instanca.ExecuteNonQuery(upit);  
  
                return true;  
            }  
            catch (Exception)  
            {  
                return false;  
            }  
        }  
    }



Слика 122 - Дијаграм секвенцe ”RazduziSO”



Слика 123 - Пројектовање системских операција (понашање система)

### Пројектовање структуре софтверског система (доменске класе)

На основу претходно дефинисаног модела, идентификоване су следеће класе:

* Autor
* Bibliotekar
* Clan
* Knjiga
* KnjigaAutor
* KnjigaPrimerak
* Zaduzenje

Поред ових класа, додате су и класе:

* IOpstiDomenskiObjekat – интерфејс који наслеђују све горе наведене класе
* Pretraga – у оквиру ове класе дефинисани су критеријум и вредност претраге
* TransferKlasa – класа која се шаље од клијента на сервер и обрнуто

Дефиниције ових класа су приказане испод.

public interface **IOpstiDomenskiObjekat**  
    {  
        string VratiImeTabele();  
        string VratiKljucIUslov();  
        string VratiVrednostiZaInsert();  
        string VratiVrednostZaUpdate();  
        string VratiKljucZaMax();  
        List<IOpstiDomenskiObjekat> VratiListu(OleDbDataReader citac);  
  
    }

[Serializable]  
    public class **Autor**: IOpstiDomenskiObjekat  
    {  
        private int autorID;  
        private string imePrezime;  
        private int godinaRodjenja;  
        private string biografija;

    }

[Serializable]  
    public class **Bibliotekar**  
    {  
        public string Ime { get; set; }  
        public string Prezime { get; set; }  
        public string KorisnickoIme { get; set; }  
        [Browsable(false)]  
        public string Lozinka { get; set; }

    }

[Serializable]  
    public class **Clan**: IOpstiDomenskiObjekat  
    {  
        private int clanskiBroj;  
        private string imePrezime;  
        private string brojTelefona;  
        private string adresa;  
  
        private List<Zaduzenje> listaZaduzenja;  
}

[Serializable]  
    public class **Knjiga**: IOpstiDomenskiObjekat  
    {  
        private int knjigaID;  
        private string naziv;  
        private string opis;  
        private int godinaIzdanja;  
        private int brojPrimeraka;  
        private int raspolozivo;  
  
        private List<Autor> listaAutora;  
        private List<KnjigaPrimerak> spisakPrimeraka;  
  
        public Knjiga()  
        {  
            listaAutora = new List<Autor>();  
            spisakPrimeraka = new List<KnjigaPrimerak>();  
            brojPrimeraka = 0;  
            raspolozivo = 0;  
        }

    }

public class **KnjigaAutor**:IOpstiDomenskiObjekat  
    {  
        public int KnjigaID { get; set; }  
        public int AutorID { get; set; }

    }

[Serializable]  
    public class **KnjigaPrimerak**: IOpstiDomenskiObjekat  
    {  
        private int primerakID;  
        private bool raspoloziva;  
        private Knjiga knjiga;

    }

public enum KriterijumPretrage  
    {  
        ImePrezimeClan,  
        ClanskiBroj,  
        ImePrezimeAutor,  
        BrojKnjige,  
        NazivKnjige  
    }  
    [Serializable]  
    public class **Pretraga**  
    {  
        public string Vrednost { get; set; }  
        public KriterijumPretrage KriterijumPretrage { get; set; }  
    }

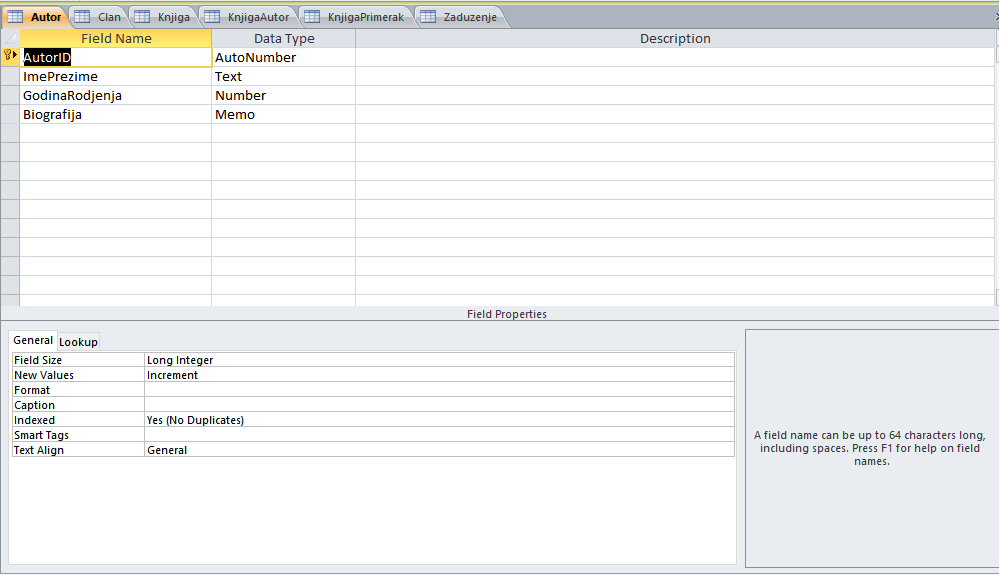
public enum Operacija  
    {  
        Login,  
        VratiSveClanove,  
        VratiSveKnjige,  
        VratiSveAutore,  
        SacuvajIzmeneKnjiga,  
        DajPrimerakID,  
        UbaciKnjigu,  
        PretraziKnjige,  
        NadjiKnjigu,  
        ObrisiKnjigu,  
        SacuvajIzmeneClan,  
        UbaciClana,  
        PretraziClanove,  
        NadjiClana,  
        ObrisiClana,  
        Zaduzi,  
        Razduzi,  
        NadjiZaduzenje,  
        Kraj  
    }  
  
    [Serializable]  
    public class **TransferKlasa**  
    {  
        public Operacija Operacija { get; set; }  
        public object TransferObjekat { get; set; }  
        public bool Signal { get; set; }  
    }

[Serializable]  
    public class **Zaduzenje**: IOpstiDomenskiObjekat  
    {  
        private Clan clan;  
        private KnjigaPrimerak knjigaPrimerak;  
        private DateTime datumOd;  
        private DateTime? datumDo;  
  
        public Zaduzenje()  
        {  
            datumDo = null;  
        }

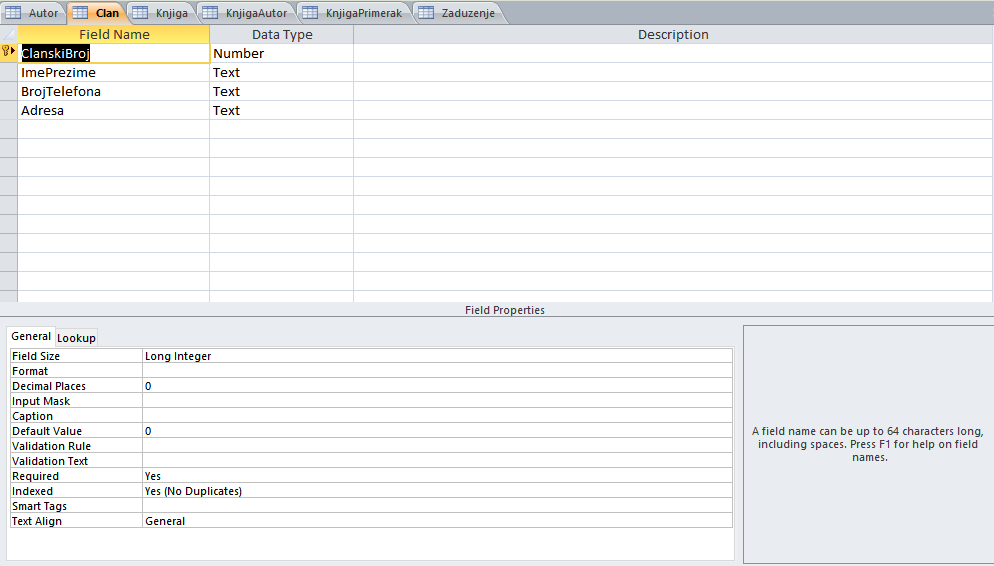
    }

# 4.3 Пројектовање складишта података

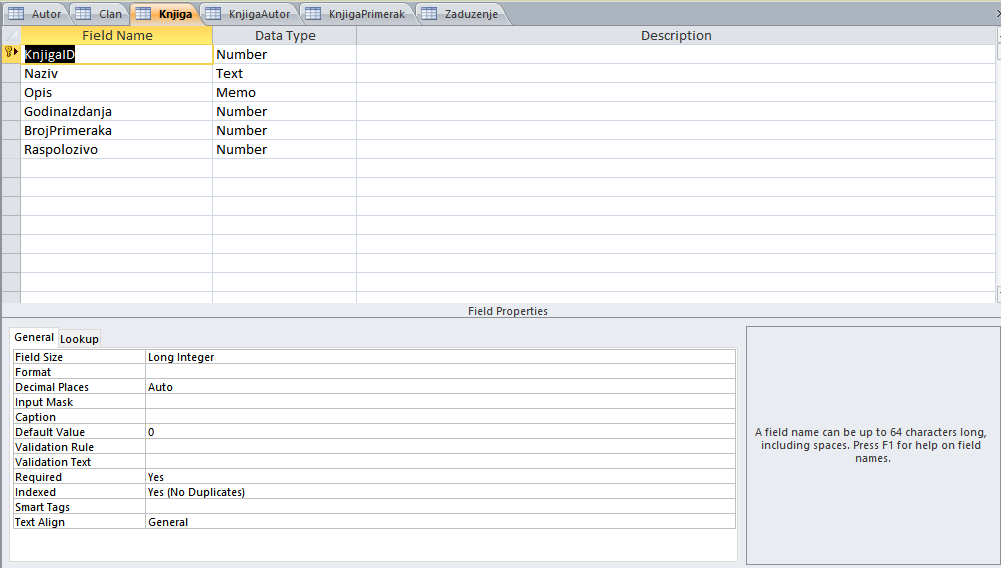
На основу релационог модела и ограничења, пројектоване су табеле базе података коју користи софтверски систем.



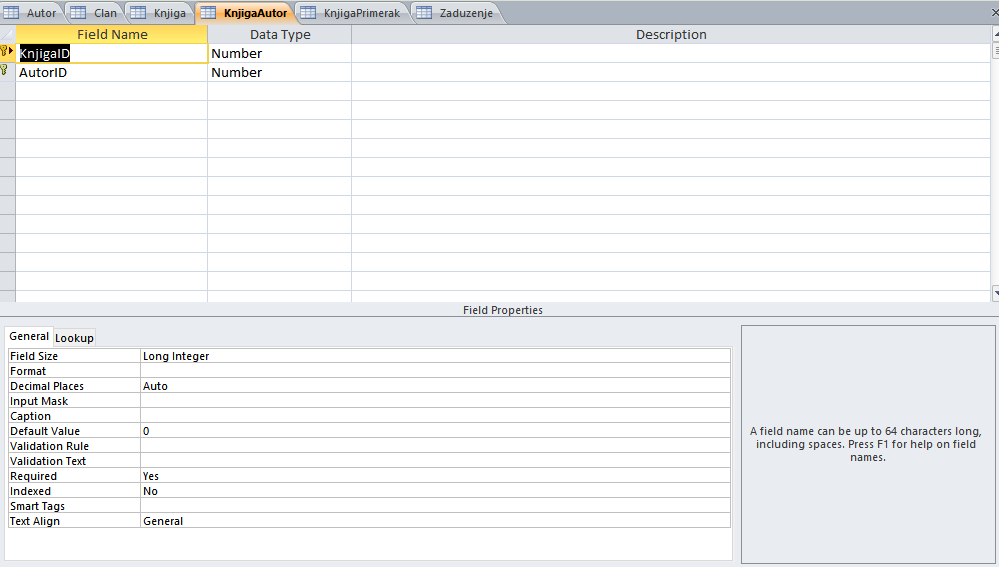
Слика 124 - Табела Аутор



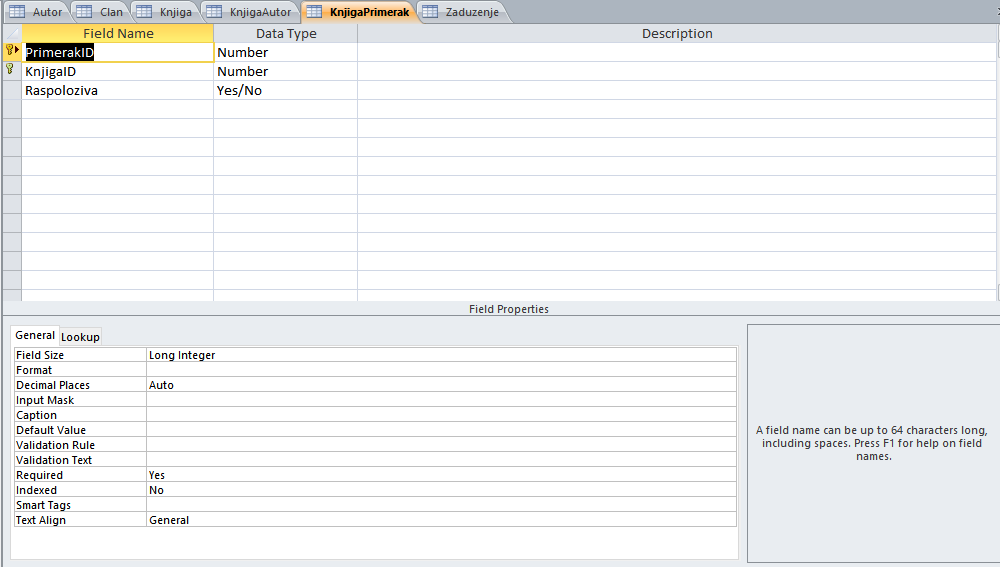
Слика 125 - Табела Члан



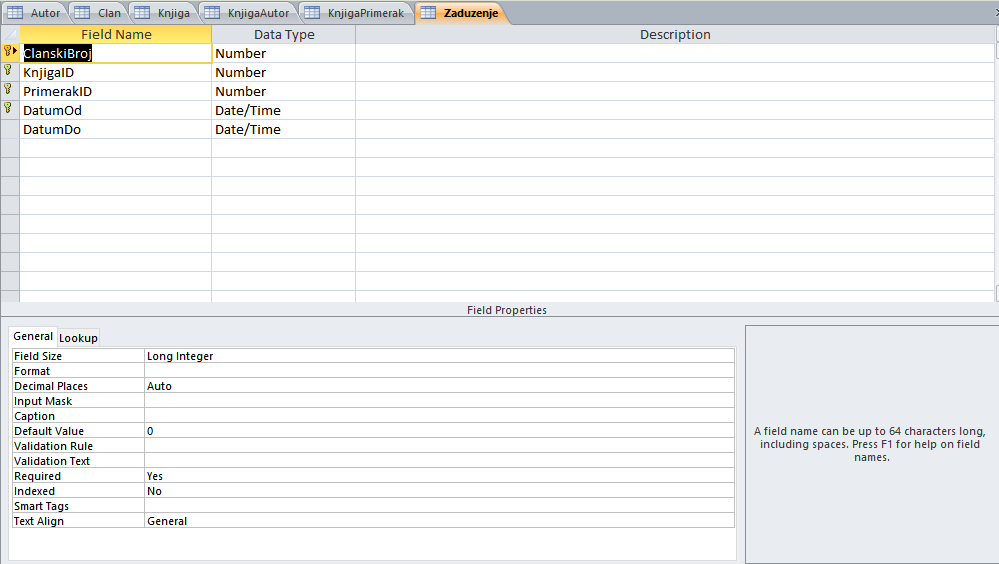
Слика 126 - Табела Књига



Слика 127 - Табела КњигаАутор



Слика 128 - Табела КњигаПримерак

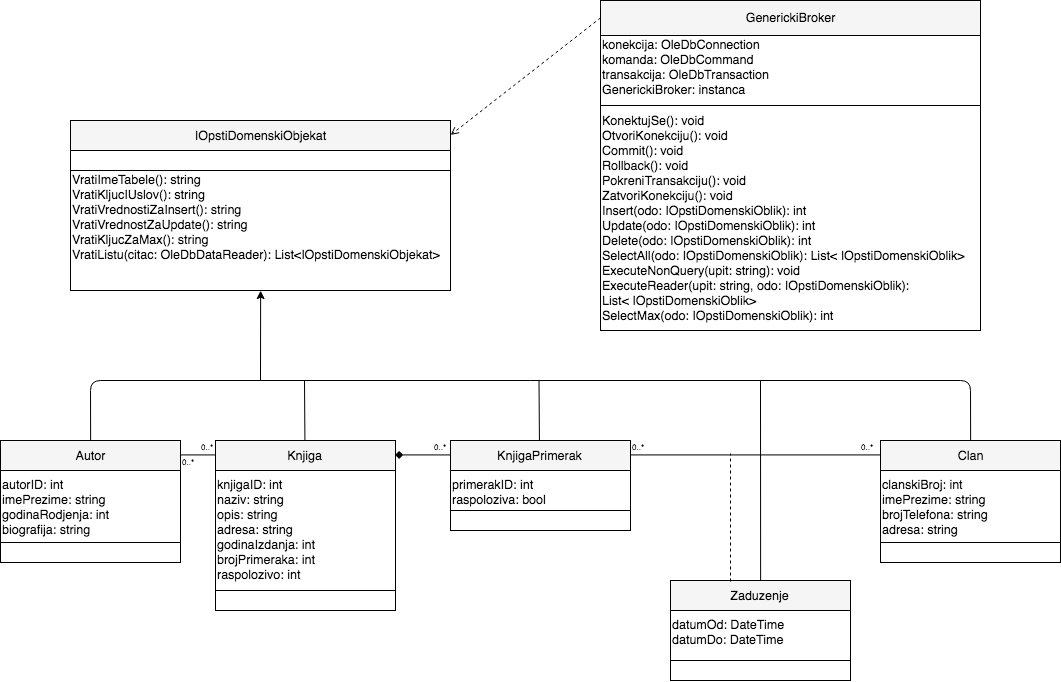


Слика 129 - Табела Задужење

За комуникацију са базом података правимо брокера базе података. Имплементиран је брокер базе података патерн који је одговоран за материјализацију, дематеријализацију и кеширање објеката у меморији. Он се често назива и **Database Mapper** патерн.

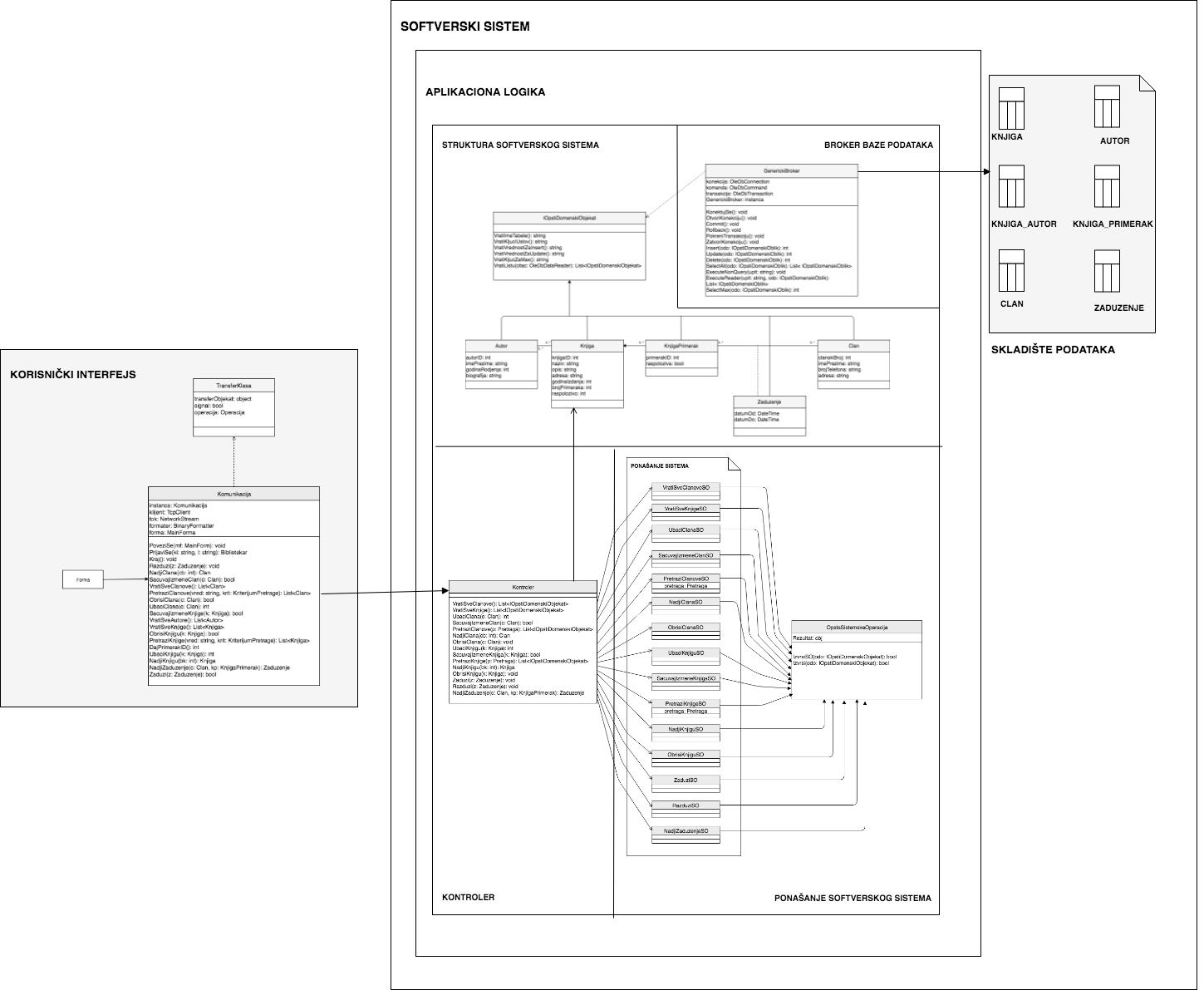
public class **GenerickiBroker**  
    {  
        private OleDbConnection konekcija;  
        private string connectionString = @"Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\Users\acoji\Desktop\Jelena\Softveri\SoftveriProjekat\Baza.accdb";  
        private OleDbCommand komanda;  
        private OleDbTransaction transakcija;  
  
        #region Singleton  
        private static GenerickiBroker instanca;  
          
        private GenerickiBroker()  
        {  
            KonektujSe();  
        }  
  
        public static GenerickiBroker Instanca  
        {  
            get  
            {  
                if (instanca == null)  
                    instanca = new GenerickiBroker();  
                return instanca;  
            }  
        }  
  
        void KonektujSe()  
        {  
            konekcija = new OleDbConnection(connectionString);  
            komanda = konekcija.CreateCommand();  
        }  
        #endregion  
  
        #region OsnovneFje  
        public void PokreniTransakciju()  
        {  
            transakcija = konekcija.BeginTransaction();  
            komanda.Transaction = transakcija;  
        }  
  
        public void Commit()  
        {  
            transakcija.Commit();  
        }  
  
        public void Rollback()  
        {  
            transakcija.Rollback();  
        }  
  
        public void OtvoriKonekciju()  
        {  
            try  
            {  
                konekcija.Open();  
            }  
            catch (Exception)  
            {  
                  
            }  
        }  
  
        public void ZatvoriKonekciju()  
        {  
            if (konekcija != null)  
                konekcija.Close();  
        }  
        #endregion  
  
        #region ProsteGenericke  
        public int Insert(IOpstiDomenskiObjekat odo)  
        {  
            komanda.CommandText = $"Insert into {odo.VratiImeTabele()} values ({odo.VratiVrednostiZaInsert()})";  
            komanda.CommandType = System.Data.CommandType.Text;  
            return komanda.ExecuteNonQuery();  
        }  
          
        public void Update(IOpstiDomenskiObjekat odo)  
        {  
            komanda.CommandText = $"Update {odo.VratiImeTabele()} set {odo.VratiVrednostZaUpdate()} where {odo.VratiKljucIUslov()}";  
            komanda.CommandType = System.Data.CommandType.Text;  
            komanda.ExecuteNonQuery();  
        }  
  
        public void Delete(IOpstiDomenskiObjekat odo)  
        {  
            komanda.CommandText = $"Delete from {odo.VratiImeTabele()} where {odo.VratiKljucIUslov()}";  
            komanda.CommandType = System.Data.CommandType.Text;  
            komanda.ExecuteNonQuery();  
        }  
  
        public List<IOpstiDomenskiObjekat> SelectAll(IOpstiDomenskiObjekat odo)  
        {  
            komanda.CommandText = $"Select \* from {odo.VratiImeTabele()}";  
            komanda.CommandType = System.Data.CommandType.Text;  
            OleDbDataReader citac = komanda.ExecuteReader();  
            List<IOpstiDomenskiObjekat> lista =  odo.VratiListu(citac);  
            citac.Close();  
            return lista;  
        }  
        #endregion  
  
        
        public void ExecuteNonQuery(string upit)  
        {  
            komanda.CommandText = upit;  
            komanda.CommandType = System.Data.CommandType.Text;  
            komanda.ExecuteNonQuery();  
        }  
  
        public List<IOpstiDomenskiObjekat> ExecuteReader(string upit, IOpstiDomenskiObjekat odo)  
        {  
            komanda.CommandText = upit;  
            komanda.CommandType = System.Data.CommandType.Text;  
            OleDbDataReader citac = komanda.ExecuteReader();  
            List<IOpstiDomenskiObjekat> lista = odo.VratiListu(citac);  
            citac.Close();  
            return lista;  
        }  
  
        public int SelectMax(IOpstiDomenskiObjekat odo)  
        {  
            komanda.CommandText = $"Select max({odo.VratiKljucZaMax()}) from {odo.VratiImeTabele()}";  
            komanda.CommandType = System.Data.CommandType.Text;  
            var res = komanda.ExecuteScalar();  
            if(res != DBNull.Value)  
            {  
                return Convert.ToInt32(res) + 1;  
            }  
            else  
            {  
                return 11;  
            }  
        }}

Као реултат пројектовања брокера базе података и доменских класа, добијени су дијаграми класа:



Слика 130 - Дијаграм класа након пројектовања брокера базе података

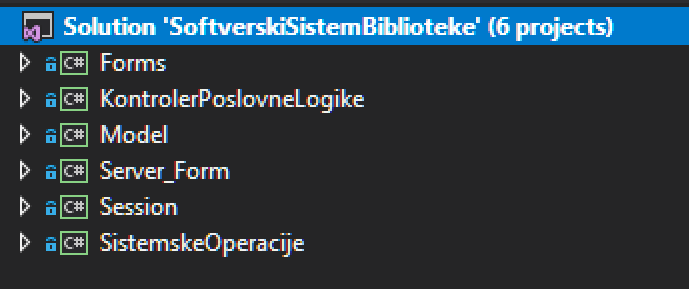
Архитектура софтверског система након пројектовања базе података изгледа овако:



Слика 131 - Архитектура софтверског система након пројектовања складишта података

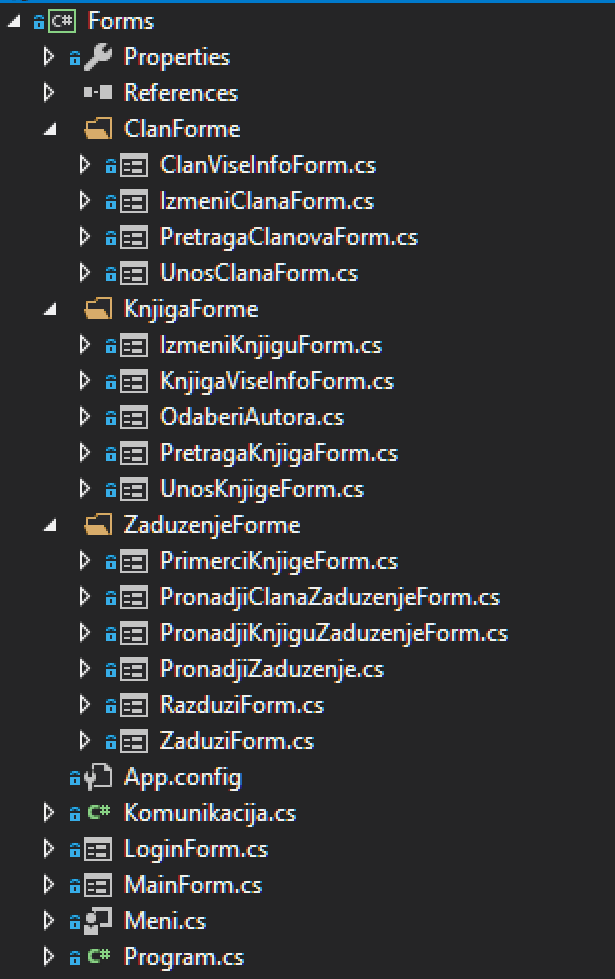
5. Имплементација

Имплементација овог софтверског система рађена је у С# програмском језику. Организација пројеката приказана је на слици испод.



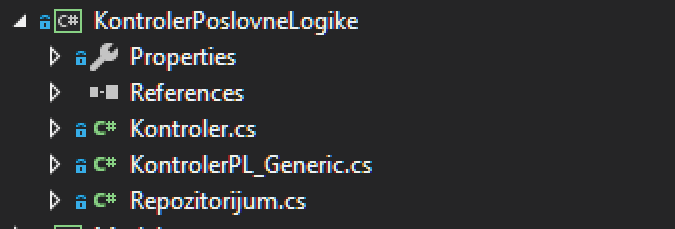
Слика 132 - Садржај пројекта

Први пројекат садржи клијентску страну апликације. У њему су креиране све форме као и контролер корисничког интергејса, који служи за комуникацију између клијентске и серверске апликације (одатле и назив класе).



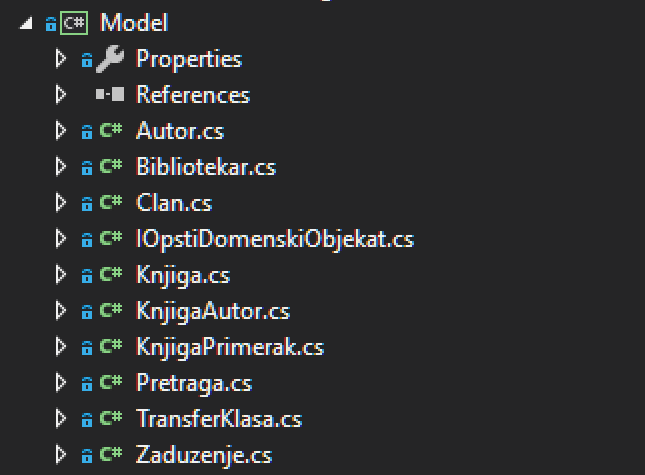
Слика 133 - Клијентски пројекат

Следећи пројекат, како што његов назив каже, садржи контролера пословне логике.



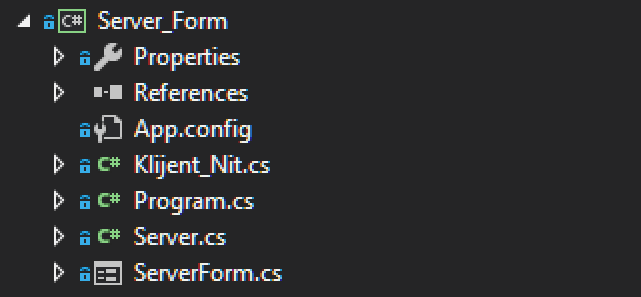
Слика 134 - Контролер пословне логике пројекат

Даље, имамо пројекат који садржи све доменске класе, као и класе коришћене за размену података између клијентске и серверске апликације. Класа Претрага садржи критеријум и вредност по којој се врше претраге, док је Трансфер класа је класа која се шаље од клијента ка серверу и обрнуто.



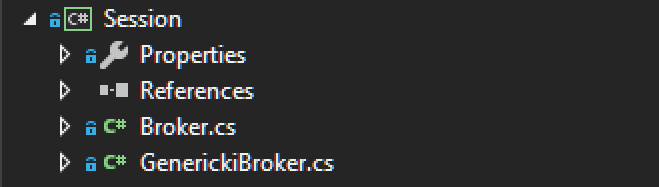
Слика 135 - Модел пројекат

Следећи пројекат је сервер. У оквиру овог пројекта налази се серверска форма и класе потребне за функсионисање сервера.



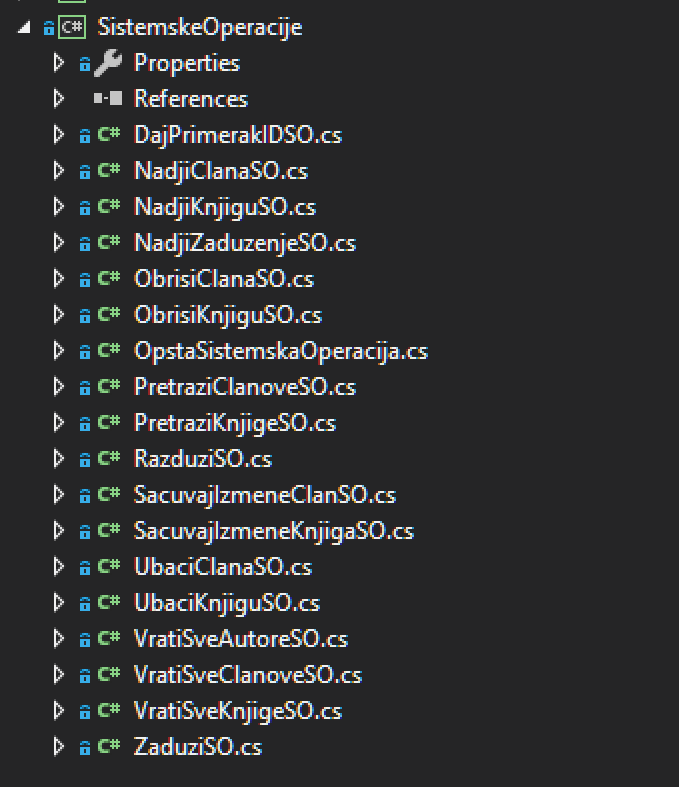
Слика 136 - Сервер пројекат

У оквиру пројекта Сесија, дефинисали смо брокера базе података. Методе имплементиране у овој класи су генеричке, чиме смо обезбедили једноставно проширивање пројекта (уколико бисмо додали нове класе у модел, брокер се не би мењао).



Слика 137 - Сесија пројекат

Последњи пројекат садржи све коришћене системске операције.



Слика 138 - Системске операције пројекат

6. Закључак

Пратећи фазе развоја софтверког система, као крајњи резултат добијен је систем који се може користити у библиотеци. Ово је само један од могућих решења, а такође се и овај систем може додатно проширити и унапредити.

7. Литература

* ”Projektovanje softvera - skripta”, Siniša Vlajić, 2015 (преузето са: <https://www.researchgate.net/publication/303858135_Projektovanje_softvera_-_skripta?enrichId=rgreq-43bc76d9065bff75557b5ab9b01ee2d2-XXX&enrichSource=Y292ZXJQYWdlOzMwMzg1ODEzNTtBUzozNzA4NDg4NzI1ODMxNjhAMTQ2NTQyODY2OTAyNQ%3D%3D&el=1_x_2&_esc=publicationCoverPdf>)