

Универзитет у Београду  
Факултет организационих наука  
Катедра за софтверско инжењерство



Семинарски рад из предмета  
**Пројектовање софтвера**

**Софтверски систем за праћење рада дома  
здравља у C# окружењу**

Ментор: **проф. др Синиша Влајић**

Студент: **Јован Петровић**

Број индекса: **2021/0013**

Београд, 2024.

# Садржај

1. Увод .....	3
2. Прикупљање корисничких захтева .....	4
2.1 Вербални опис .....	4
2.2 Случајеви коришћења .....	6
3. Анализа .....	17
3.1 Понашање софтверског система - Одређивање системских операција на основу сценарија случаја коришћења .....	17
3.2 Понашање софтверског система - Секвенцни дијаграми случаја коришћења .....	20
3.3 Понашање софтверског система - Дефинисање уговора о системским операцијама .....	32
3.4 Структура софтверског система – Концептуални (доменски) модел .....	34
3.5 Структура софтверског система – Релациони модел .....	36
3.6 Табела структурних и вредносних ограничења релационог модела .....	37
4. Пројектовање .....	40
4.1 Пројектовање корисничког интерфејса .....	41
Пројектовање екранских форми .....	41
Пројектовање контролера корисничког интерфејса .....	76
4.2 Пројектовање апликационе логике .....	77
Контролер апликационе логике .....	77
Пословна логика .....	78
Пројектовање понашања софтверског система - системске операције .....	79
Пројектовање структуре софтверског система - доменске класе .....	84
Пројектовање брокера базе података .....	95
4.3 Пројектовање складишта података .....	96
Преглед табела .....	96
Тригери .....	98
5. Имплементација .....	99
Структура пројекта .....	101
6. Тестирање .....	103
7. Закључак .....	104
Литература .....	105

## 1. Увод

У овом семинарском раду биће представљен развој софтверског система за праћење рада дома здравља, чији ће циљ бити ефикасно управљање и централизована обрада података о пацијентима, лекарима, здравственим картонима и другим релевантним медицинским информацијама. Систем ће бити реализован као трослојна десктоп клијент-сервер апликација.

Процес развоја ће се базирати на упрошћеној Лармановој методи, која подразумева следеће фазе:

- **Прикупљање корисничких захтева** – у овој фази дефинишу се својства и услови које софтверски систем мора да задовољи. Захтеви се деле на функционалне, који описују конкретне функције система, и нефункционалне, који обухватају атрибуте квалитета као што су употребљивост, поузданост и перформансе.
- **Анализа** – у овој фази описује се логичка структура и понашање софтверског система, односно пословна логика. Кључни делови анализе укључују концептуални (доменски) модел и релациони модел за структуру, као и секвенцијалне дијаграме и дефиницију системских операција за понашање система.
- **Пројектовање** – фаза у којој се детаљно дефинише архитектура софтверског система, која је реализована као тронивојска и састоји се од: корисничког интерфејса, апликационе логике и складишта података.
- **Имплементација** – овде се реализују имплементационе компоненте дефинисане у претходним фазама, коришћењем одговарајућих технологија као што су C# и .NET. Кодирају се функционалности према пројектованој архитектури, уз примену добрих пракси и дизајн патерна.
- **Тестирање** – у последњој фази се тестирањем проверава исправност имплементационих компоненти, понашање система у различитим сценаријима и усаглашеност са захтевима, како би се осигурало да систем ради поуздано

Свакој фази упрошћене Ларманове методе посвећено је једно поглавље овог семинарског рада, чиме је обезбеђена логична и јасна структура рада.

## 2. Прикупљање корисничких захтева

### 2.1 Вербални опис

Потребно је направити **Софтверски систем за праћење рада дома здравља у С# окружењу**. Софтверски систем, односно његова пословна логика (у општем смислу), састоји се из следећих *апстрактних концепата*:

а) **пружалац услуге**

б) **прималац услуге**

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

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

У наведеном софтверском систему пружалац услуге је Лекар, прималац услуге је Пацијент, документ који описује процес пружања услуге је **Здравствени картон**. **Шифарници** су **Дијагноза**, **Место**, **Сертификат**.

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

Преко здравственог картона је могуће забележити више дијагноза. Здравствени картон је повезан са једним лекаром и једним пацијентом. Пацијент је везан за једно место. Лекар може бити везан за више сертификата, док сертификат може бити везан за више лекара

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

Потребно је обезбедити следеће функционалности за наведене конкретне концепте:

Редни број концепта	Концепт	Функционалности
1.	Здравствени картон	Креирај, Претражи, Промени, Обриши
2.	Пацијент	Креирај, Претражи, Промени, Обриши
3.	Лекар	Пријави, Креирај, Претражи, Промени, Обриши
4.	Дијагноза	Креирај, Претражи, Промени, Обриши
5.	Место	Креирај, Претражи, Промени, Обриши
6	Сертификат	Убаци, Претражи, Промени, Обриши
7.	Ставка здравственог картона	
8.	ЛеС	

Табела 1: Повезивање концепата и функционалности

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

1. Документи
  - 1.1 Здравствени картон
2. Пружалац услуге
  - 2.1 Лекар
3. Прималац услуге
  - 3.1 Пацијент
4. Шифарници
  - 4.1 Дијагноза,
  - 4.2 Место,
  - 4.3 Сертификат,
5. Подешавања софтверског система
6. О програму

## 2.2 Случајеви коришћења

На основу наведених функционалности концепата уочени су следећи случајеви коришћења:

Шифра случаја коришћења	Назив случаја коришћења	Предуслови С.К. (листе)	Критеријуми претраживања се односе на:
SK1	СК1- Креирај здравствени картон	Учитане су листе: а) Лекар б) Пацијент с) Дијагноза	
SK2	СК2- Претражи здравствени картон		а) Здравствени картон б) Лекар с) Пацијент д) Дијагноза
SK3	СК3- Промени здравствени картон	Учитане су листе: а) Лекар б) Пацијент с) Дијагноза	а) Здравствени картон б) Лекар с) Пацијент д) Дијагноза
SK4	СК4- Обриши здравствени картон		а) Здравствени картон б) Лекар с) Пацијент д) Дијагноза
SK5	СК5- Креирај пацијент	Учитане су листе: а) Место	
SK6	СК6- Претражи пацијент		а) Пацијент б) Место
SK7	СК7- Промени пацијент	Учитане су листе: а) Место	а) Пацијент б) Место
SK8	СК8- Обриши пацијент		а) Пацијент б) Место
SK9	СК9- Пријави лекар		
SK10	СК10- Креирај лекар	Учитане су листе: а) Сертификат	
SK11	СК11- Претражи лекар		а) Лекар б) Сертификат
SK12	СК12- Промени лекар	Учитане су листе: а) Сертификат	а) Лекар б) Сертификат
SK13	СК13- Обриши лекар		а) Лекар б) Сертификат
SK14	СК14- Креирај дијагноза		
SK15	СК15- Претражи дијагноза		а) Дијагноза
SK16	СК16- Промени дијагноза		а) Дијагноза
SK17	СК17- Обриши дијагноза		а) Дијагноза
SK18	СК18- Креирај место		
SK19	СК19- Претражи место		а) Место
SK20	СК20- Промени место		а) Место
SK21	СК21- Обриши место		а) Место
SK22	СК22- Убацати сертификат		
SK23	СК23- Претражи сертификат		а) Сертификат
SK24	СК24- Промени сертификат		а) Сертификат
SK25	СК25- Обриши сертификат		а) Сертификат

Табела 2: Случајеви коришћења софтверског система

За следеће случајеве коришћења ћемо дати детаљан опис:

СК1- Креирај здравствени картон
СК2- Претражи здравствени картон
СК3- Промени здравствени картон
СК5- Креирај пацијент
СК6- Претражи пацијент
СК7- Промени пацијент
СК8- Обриши пацијент
СК9- Пријави лекар
СК22- Убацати сертификат

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

## СК1- Креирај здравствени картон

### Назив СК

Креирај здравствени картон

### Актори СК

Лекар

### Учесници СК

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

**Предуслови:** Кориснички интерфејс (клијентски програм) и систем (серверски програм) су покренути. Лекар је пријављен под својом шифром. Систем приказује форму за рад са здравственим картоном. *Учитане су листе: а) Лекар б) Пацијент с) Дијагноза*

### Основни сценарио СК:

1. Лекар позива систем да креира здравствени картон. (АПСО)
2. Систем креира здравствени картон. (СО)
3. Систем приказује лекару здравствени картон и поруку: "Систем је креирао здравствени картон". (ИА)
4. Лекар уноси податке о здравственом картону. (АПУСО)
5. Лекар контролише да ли је коректно унео податке о здравственом картону. (АНСО)
6. Лекар позива систем да запамти податке о здравственом картону. (АПСО)
7. Систем памти податке о здравственом картону. (СО)
8. Систем приказује лекару здравствени картон и поруку: "Систем је запамтио здравствени картон." (ИА)

### Алтернативна сценарија:

- 3.1 Уколико систем не може да креира здравствени картон он приказује лекару поруку: "Систем не може да креира здравствени картон". Прекида се извршење сценарија. (ИА)
- 8.1 Уколико систем не може да запамти податке о здравственом картону он приказује лекару поруку: "Систем не може да запамти здравствени картон". (ИА)



## СК2- Претражи здравствени картон

### Назив СК

Претражи здравствени картон

### Актори СК

Лекар

### Учесници СК

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

**Предуслови:** Кориснички интерфејс (клијентски програм) и систем (серверски програм) су покренути. Лекар је пријављен под својом шифром. Кориснички интерфејс приказује форму за рад са здравственим картоном. На наведеној екранској форми су дефинисани критеријуми, који се односе на: а) Здравствени картон б) Лекар с) Пацијент d) Дијагноза, који ће да врате листу здравствених картона.

### Основни сценарио СК:

1. Лекар бира критеријуме на основу којих претражује здравствене картоне. (АПУСО)
2. Лекар позива систем да нађе здравствене картоне по задатим критеријумима. (АПСО)
3. Систем тражи здравствене картоне по задатим критеријумима. (СО)
4. Систем приказује лекару здравствене картоне и поруку: "Систем је нашао здравствене картоне по задатим критеријумима". (ИА)
5. Лекар бира здравствени картон. (АПУСО)
6. Лекар позива систем да нађе здравствени картон. (АПСО)
7. Систем тражи здравствени картон. (СО)
8. Систем приказује лекару здравствени картон и поруку: "Систем је нашао здравствени картон". (ИА)

### Алтернативна сценарија:

- 4.1 Уколико систем не може да нађе здравствене картоне он приказује лекару поруку: "Систем не може да нађе здравствене картоне по задатим критеријумима". Прекида се извршење сценарија. (ИА)
- 8.1 Уколико систем не може да нађе здравствени картон он приказује лекару поруку: "Систем не може да нађе здравствени картон ".(ИА)

### СКЗ- Промени здравствени картон

#### Назив СК

Промени здравствени картон

#### Актери СК

Лекар

#### Учесници СК

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

**Предуслови:** Кориснички интерфејс (клијентски програм) и систем (серверски програм) су покренути. Лекар је пријављен под својом шифром. Кориснички интерфејс приказује форму за рад са здравственим картоном. На наведеној екранској форми су дефинисани критеријуми, који се односе на: а) Здравствени картон б) Лекар с) Пацијент d) Дијагноза, који ће да врате листу здравствених картона. Учитане су листе: а) Лекар б) Пацијент с) Дијагноза

#### Основни сценарио СК:

1. Лекар бира критеријуме на основу којих претражује здравствене картоне. (АПУСО)
2. Лекар позива систем да нађе здравствене картоне по задатим критеријумима. (АПСО)
3. Систем тражи здравствене картоне по задатим критеријумима. (СО)
4. Систем приказује лекару здравствене картоне и поруку: "Систем је нашао здравствене картоне по задатим критеријумима". (ИА)
5. Лекар бира здравствени картон. (АПУСО)
6. Лекар позива систем да нађе здравствени картон. (АПСО)
7. Систем тражи здравствени картон. (СО)
8. Систем приказује лекару здравствени картон и поруку: "Систем је нашао здравствени картон". (ИА)
9. Лекар уноси (мења) податке о здравственом картону. (АПУСО)
10. Лекар контролише да ли је коректно унео податке о здравственом картону. (АНСО)
11. Лекар позива систем да запамти податке о здравственом картону. (АПСО)
12. Систем памти податке о здравственом картону. (СО)
13. Систем приказује лекару здравствени картон и поруку: "Систем је запамтио здравствени картон." (ИА)

#### Алтернативна сценарија:

- 4.1 Уколико систем не може да нађе здравствене картоне он приказује лекару поруку: "Систем не може да нађе здравствене картоне по задатом критеријуму". Прекида се извршење сценарија. (ИА)
- 8.1 Уколико систем не може да нађе здравствени картон он приказује лекару поруку: "Систем не може да нађе здравствени картон ". Прекида се извршење сценарија. (ИА)
- 13.1 Уколико систем не може да запамти податке о здравственом картону он приказује лекару поруку: "Систем не може да запамти здравствени картон". (ИА)

## СК5- Креирај пацијент

### Назив СК

Креирај пацијент

### Актори СК

Лекар

### Учесници СК

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

**Предуслови:** Кориснички интерфејс (клијентски програм) и систем (серверски програм) су покренути. Лекар је пријављен под својом шифром. Систем приказује форму за рад са пацијентом. *Учитане су листе: а) Место*

### Основни сценарио СК:

1. Лекар позива систем да креира пацијента. (АПСО)
2. Систем креира пацијента. (СО)
3. Систем приказује лекару пацијента и поруку: "Систем је креирао пацијента". (ИА)
4. Лекар уноси податке о пацијенту. (АПУСО)
5. Лекар контролише да ли је коректно унео податке о пацијенту. (АНСО)
6. Лекар позива систем да запамти податке о пацијенту. (АПСО)
7. Систем памти податке о пацијенту. (СО)
8. Систем приказује лекару пацијента и поруку: "Систем је запамтио пацијента." (ИА)

### Алтернативна сценарија:

3.1 Уколико систем не може да креира пацијента он приказује лекару поруку: "Систем не може да креира пацијента". Прекида се извршење сценарија. (ИА)

8.1 Уколико систем не може да запамти податке о пацијенту он приказује лекару поруку: "Систем не може да запамти пацијента". (ИА)

## СК6- Претражи пацијент

### Назив СК

Претражи пацијент

### Актори СК

Лекар

### Учесници СК

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

**Предуслови:** Кориснички интерфејс (клијентски програм) и систем (серверски програм) су покренути. Лекар је пријављен под својом шифром. Кориснички интерфејс приказује форму за рад са пацијентом. На наведеној екранској форми су дефинисани критеријуми, који се односе на: а) Пацијент б) Место, који ће да врате листу пацијената.

### Основни сценарио СК:

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

### Алтернативна сценарија:

- 4.1 Уколико систем не може да нађе пацијенте он приказује лекару поруку: "Систем не може да нађе пацијенте по задатим критеријумима". Прекида се извршење сценарија. (ИА)
- 8.1 Уколико систем не може да нађе пацијента он приказује лекару поруку: "Систем не може да нађе пацијента ".(ИА)

## СК7- Промени пацијент

### Назив СК

Промени пацијент

### Актери СК

Лекар

### Учесници СК

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

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

### Основни сценарио СК:

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

### Алтернативна сценарија:

- 4.1 Уколико систем не може да нађе пацијенте он приказује лекару поруку: "Систем не може да нађе пацијенте по задатом критеријуму". Прекида се извршење сценарија. (ИА)
- 8.1 Уколико систем не може да нађе пацијента он приказује лекару поруку: "Систем не може да нађе пацијента". Прекида се извршење сценарија. (ИА)
- 13.1 Уколико систем не може да запамти податке о пацијенту он приказује лекару поруку: "Систем не може да запамти пацијента". (ИА)

## СК8- Обриши пацијент

### Назив СК

Обриши пацијент

### Актори СК

Лекар

### Учесници СК

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

**Предуслови:** Кориснички интерфејс (клијентски програм) и систем (серверски програм) су покренути. Лекар је пријављен под својом шифром. Кориснички интерфејс приказује форму за рад са пацијентом. На наведеној екранској форми су дефинисани критеријуми, који се односе на: а) Пацијент б) Место, који ће да врате листу пацијената.

### Основни сценарио СК:

1. Лекар бира критеријуме на основу којих претражује пацијенте. (АПУСО)
2. Лекар позива систем да нађе пацијенте по задатим критеријумима. (АПСО)
3. Систем тражи пацијенте по задатим критеријумима. (СО)
4. Систем приказује лекару пацијенте и поруку: "Систем је нашао пацијенте по задатим критеријумима". (ИА)
5. Лекар бира пацијента. (АПУСО)
6. Лекар позива систем да нађе пацијента. (АПСО)
7. Систем тражи пацијента. (СО)
8. Систем приказује лекару пацијента и поруку: "Систем је нашао пацијента". (ИА)
9. Лекар позива систем да обрише пацијента. (АПСО)
10. Систем брише пацијента. (СО)
11. Систем приказује лекару поруку: "Систем је обрисао пацијента." (ИА)

### Алтернативна сценарија:

- 4.1 Уколико систем не може да нађе пацијенте он приказује лекару поруку: "Систем не може да нађе пацијенте по задатим критеријумима". Прекида се извршење сценарија. (ИА)
- 8.1 Уколико систем не може да нађе пацијента он приказује лекару поруку: "Систем не може да нађе пацијента". Прекида се извршење сценарија. (ИА)
- 11.1 Уколико систем не може да обрише пацијента он приказује лекару поруку: "Систем не може да обрише пацијента". (ИА)

## СК9- Пријави лекар

### Назив СК

Пријави лекар

### Актори СК

Лекар

### Учесници СК

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

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

### Основни сценарио СК:

1. Лекар уноси корисничко име и шифру. (АПУСО)
2. Лекар контролише да ли је коректно унео корисничко име и шифру. (АНСО)
3. Лекар позива систем да провери корисничко име и шифру. (АПСО)
4. Систем проверава корисничко име и шифру. (СО)
5. Систем приказује лекару поруку: "Корисничко име и шифра су исправни." (ИА)
6. Кориснички интерфејс позива главни форму и мени. (КИПГФМ)

### Алтернативна сценарија:

5.1 Уколико систем провером установи да корисничка шифра и/или шифра нису исправни он приказује лекару поруку: "Корисничко име и шифра нису исправни ". (ИА)

6.1 Уколико кориснички интерфејс не може да отвори главну форму и мени приказује лекару поруку: "Не може да се отвори главна форма и мени". (НПГФМ)

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

## СК22- Убацати сертификат

### Назив СК

Убацати сертификат

### Актери СК

Лекар

### Учесници СК

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

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

### Основни сценарио СК:

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

### Алтернативна сценарија:

5.1 Уколико систем не може да запамти податке о сертификату он приказује лекару поруку: "Систем не може да запамти сертификат". (ИА)



### 3. Анализа

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

На основу наведених случајева коришћења уочене су следеће системске операције:

Шифра случаја коришћења	Назив случаја коришћења	Системске операције
SK1	СК1- Креирај здравствени картон	1. signal KreirajZdravstveni karton(Zdravstveni karton) 2. signal PromeniZdravstveni karton(Zdravstveni karton) 3. signal vratiListuSviLekar(Lista<Lekar>) 4. signal vratiListuSviPacijent(Lista<Pacijent>) 5. signal vratiListuSviDijagnoza(Lista<Dijagnoza>)
SK2	СК2- Претражи здравствени картон	1. signal PretraziZdravstveni karton(Zdravstveni karton) 2. signal vratiListuZdravstveni karton(kriterijumZdravstveni karton, Lista<Zdravstveni karton>) 3. signal vratiListuZdravstveni karton(kriterijumLekar, Lista<Zdravstveni karton>) 4. signal vratiListuZdravstveni karton(kriterijumPacijent, Lista<Zdravstveni karton>) 5. signal vratiListuZdravstveni karton(kriterijumDijagnoza, Lista<Zdravstveni karton>)
SK3	СК3- Промени здравствени картон	1. signal PromeniZdravstveni karton(Zdravstveni karton) 2. signal PretraziZdravstveni karton(Zdravstveni karton) 3. signal vratiListuSviLekar(Lista<Lekar>) 4. signal vratiListuSviPacijent(Lista<Pacijent>) SO6
SK4	СК4- Обриши здравствени картон	1. signal ObrisiZdravstveniKarton(Zdravstveni kraton) 2. signal vratiListuZdravstveni karton(kriterijumZdravstveni karton, Lista<Zdravstveni karton>) 3. signal vratiListuZdravstveni karton(kriterijumLekar, Lista<Zdravstveni karton>) 4. signal vratiListuZdravstveni karton(kriterijumPacijent, Lista<Zdravstveni karton>) 5. signal vratiListuZdravstveni karton(kriterijumDijagnoza, Lista<Zdravstveni karton>)
SK5	СК5- Креирај пацијент	1. signal KreirajPacijent(Pacijent) 2. signal PromeniPacijent(Pacijent) 3. signal vratiListuSviMesto(Lista<Mesto>)
SK6	СК6- Претражи пацијент	1. signal PretraziPacijent(Pacijent) 2. signal vratiListuPacijent(kriterijumPacijent, Lista<Pacijent>) 3. signal vratiListuPacijent(kriterijumMesto, Lista<Pacijent>)
SK7	СК7- Промени пацијент	1. signal PromeniPacijent(Pacijent) 2. signal PretraziPacijent(Pacijent) 3. signal vratiListuSviMesto(Lista<Mesto>) 4. signal vratiListuPacijent(kriterijumPacijent, Lista<Pacijent>) 5. signal vratiListuPacijent(kriterijumMesto, Lista<Pacijent>)
SK8	СК8- Обриши пацијент	1. signal ObrisiPacijent(Pacijent) 2. signal vratiListuPacijent(kriterijumPacijent, Lista<Pacijent>) 3. signal vratiListuPacijent(kriterijumMesto, Lista<Pacijent>)
SK9	СК9- Пријави лекар	1. signal PrijaviLekar(korisnickolme, sifra)
SK10	СК10- Креирај лекар	1. signal KreirajLekar(Lekar) 2. signal PromeniLekar(Lekar) 3. signal vratiListuSviSertifikat(Lista<Sertifikat>)
SK11	СК11- Претражи лекар	1. signal PretraziLekar(Lekar) 2. signal vratiListuLekar(kriterijumLekar, Lista<Lekar>) 3. signal vratiListuLekar(kriterijumSertifikat, Lista<Lekar>)
SK12	СК12- Промени лекар	1. signal PromeniLekar(Lekar) 2. signal PretraziLekar(Lekar) 3. signal vratiListuSviSertifikat(Lista<Sertifikat>) 4. signal vratiListuLekar(kriterijumLekar, Lista<Lekar>) 5. signal vratiListuLekar(kriterijumSertifikat, Lista<Lekar>)
SK13	СК13- Обриши лекар	1. signal ObrisiLekar(Lekar) 2. signal vratiListuLekar(kriterijumLekar, Lista<Lekar>) 3. signal vratiListuLekar(kriterijumSertifikat, Lista<Lekar>)
SK14	СК14- Креирај дијагноза	1. signal KreirajDijagnoza(Dijagnoza) 2. signal PromeniDijagnoza(Dijagnoza)

SK15	СК15-Претражи дијагноза	1. signal PretraziDijagnoza(Dijagnoza) 2. signal vratiListuDijagnoza(kriterijumDijagnoza, Lista<Dijagnoza>)
SK16	СК16-Промени дијагноза	1. signal PromeniDijagnoza(Dijagnoza) 2. signal PretraziDijagnoza(Dijagnoza) 3. signal vratiListuDijagnoza(kriterijumDijagnoza, Lista<Dijagnoza>)
SK17	СК17-Обриши дијагноза	1. signal ObrisiDijagnoza(Dijagnoza) 2. signal vratiListuDijagnoza(kriterijumDijagnoza, Lista<Dijagnoza>)
SK18	СК18-Креирај место	1. signal KreirajMesto(Mesto) 2. signal PromeniMesto(Mesto)
SK19	СК19-Претражи место	1. signal PretraziMesto(Mesto) 2. signal vratiListuMesto(kriterijumMesto, Lista<Mesto>)
SK20	СК20-Промени место	1. signal PromeniMesto(Mesto) 2. signal PretraziMesto(Mesto) 3. signal vratiListuMesto(kriterijumMesto, Lista<Mesto>)
SK21	СК21-Обриши место	1. signal ObrisiMesto(Mesto) 2. signal vratiListuMesto(kriterijumMesto, Lista<Mesto>)
SK22	СК22-Убаци сертификат	1. signal UbaciSertifikat(Sertifikat) 2. signal PromeniSertifikat(Sertifikat)
SK23	СК23-Претражи сертификат	1. signal PretraziSertifikat(Sertifikat) 2. signal vratiListuSertifikat(kriterijumSertifikat, Lista<Sertifikat>)
SK24	СК24-Промени сертификат	1. signal PromeniSertifikat(Sertifikat) 2. signal PretraziSertifikat(Sertifikat) 3. signal vratiListuSertifikat(kriterijumSertifikat, Lista<Sertifikat>)
SK25	СК25-Обриши сертификат	1. signal ObrisiSertifikat(Sertifikat) 2. signal vratiListuSertifikat(kriterijumSertifikat, Lista<Sertifikat>)

Табела 4: Системске операције случаја коришћења

Наводимо све уочене системске операције:

1. signal KreirajDijagnoza(Dijagnoza)
2. signal KreirajLekar(Lekar)
3. signal KreirajMesto(Mesto)
4. signal KreirajPacijent(Pacijent)
5. signal KreirajZdravstveni karton(Zdravstveni karton)
6. signal ObrisiDijagnoza(Dijagnoza)
7. signal ObrisiLekar(Lekar)
8. signal ObrisiMesto(Mesto)
9. signal ObrisiPacijent(Pacijent)
10. signal ObrisiSertifikat(Sertifikat)
11. signal ObrisiZdravstveniKarton(Zdravstveni karton)
12. signal PretraziDijagnoza(Dijagnoza)
13. signal PretraziLekar(Lekar)
14. signal PretraziMesto(Mesto)
15. signal PretraziPacijent(Pacijent)
16. signal PretraziSertifikat(Sertifikat)
17. signal PretraziZdravstveni karton(Zdravstveni karton)
18. signal PrijaviLekar(korisnickolme, sifra)
19. signal PromeniDijagnoza(Dijagnoza)
20. signal PromeniLekar(Lekar)
21. signal PromeniMesto(Mesto)
22. signal PromeniPacijent(Pacijent)
23. signal PromeniSertifikat(Sertifikat)
24. signal PromeniZdravstveni karton(Zdravstveni karton)
25. signal UbaciSertifikat(Sertifikat)
26. signal vratiListuDijagnoza(kriterijumDijagnoza, Lista<Dijagnoza>)
27. signal vratiListuLekar(kriterijumLekar, Lista<Lekar>)
28. signal vratiListuLekar(kriterijumSertifikat, Lista<Lekar>)
29. signal vratiListuMesto(kriterijumMesto, Lista<Mesto>)
30. signal vratiListuPacijent(kriterijumMesto, Lista<Pacijent>)
31. signal vratiListuPacijent(kriterijumPacijent, Lista<Pacijent>)
32. signal vratiListuSertifikat(kriterijumSertifikat, Lista<Sertifikat>)
33. signal vratiListuSviDijagnoza(Lista<Dijagnoza>)
34. signal vratiListuSviLekar(Lista<Lekar>)
35. signal vratiListuSviMesto(Lista<Mesto>)
36. signal vratiListuSviPacijent(Lista<Pacijent>)
37. signal vratiListuSviSertifikat(Lista<Sertifikat>)
38. signal vratiListuZdravstveni karton(kriterijumDijagnoza, Lista<Zdravstveni karton>)
39. signal vratiListuZdravstveni karton(kriterijumLekar, Lista<Zdravstveni karton>)
40. signal vratiListuZdravstveni karton(kriterijumPacijent, Lista<Zdravstveni karton>)
41. signal vratiListuZdravstveni karton(kriterijumZdravstveni karton, Lista<Zdravstveni karton>)

Табела 5: Системске операције софтверског система

## 3.2 Понашање софтверског система - Секвенцни дијаграми случаја коришћења

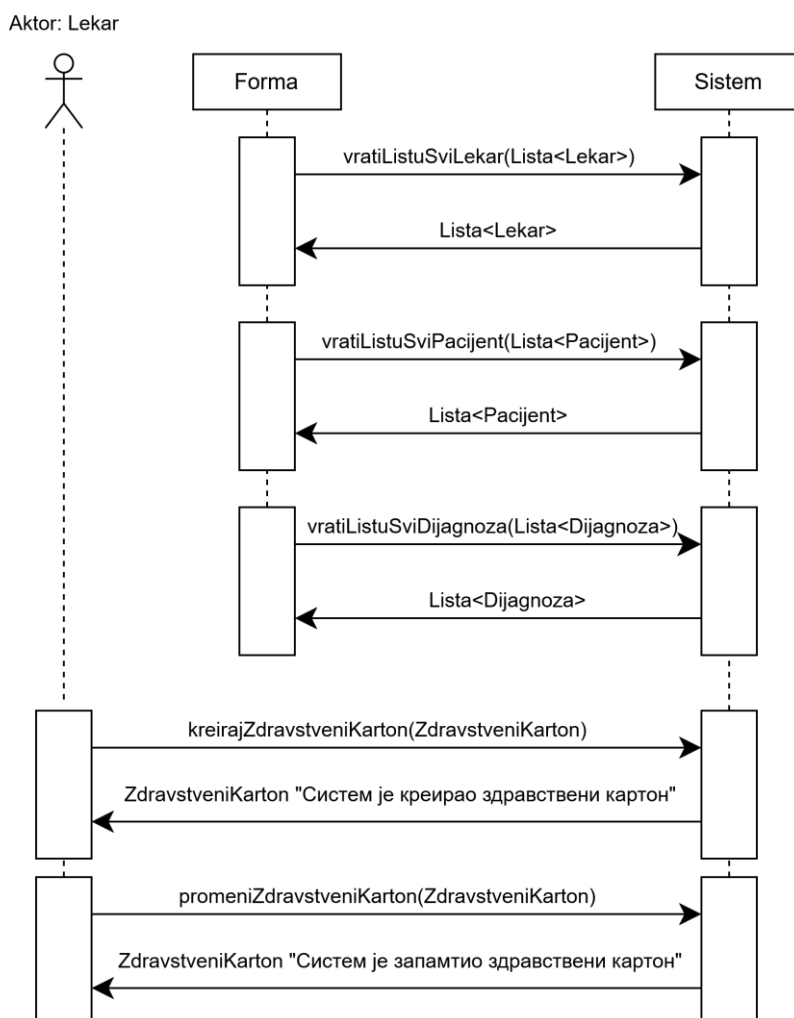
### ДС1: Дијаграми секвенци случаја коришћења – Креирај здравствени картон

#### Предуслови:

1. **Форма** позива систем да **врати** листу свих лекара. (АПСО)
2. **Систем** враћа **форми** листу свих лекара. (ИА)
3. **Форма** позива систем да **врати** листу свих пацијената. (АПСО)
4. **Систем** враћа **форми** листу свих пацијената. (ИА)
5. **Форма** позива систем да **врати** листу свих дијагноза. (АПСО)
6. **Систем** враћа **форми** листу свих дијагноза. (ИА)

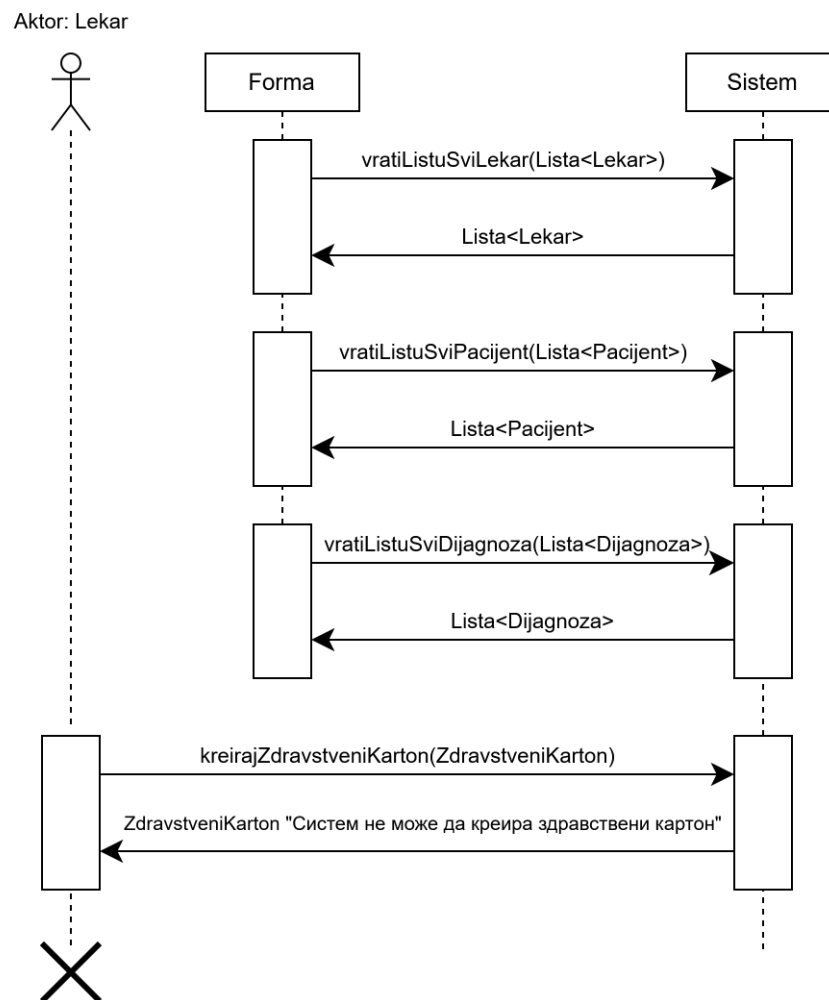
#### Основни сценарио СК:

7. **Лекар** позива **систем** да **креира** здравствени картон. (АПСО)
8. **Систем** приказује **лекару** здравствени картон и поруку: “**Систем** је креирао здравствени картон”. (ИА)
9. **Лекар** позива **систем** да **запамти** податке о **здравственом картону**. (АПСО)
10. **Систем** приказује **лекару** здравствени картон и поруку: “**Систем** је запамтио здравствени картон.” (ИА)

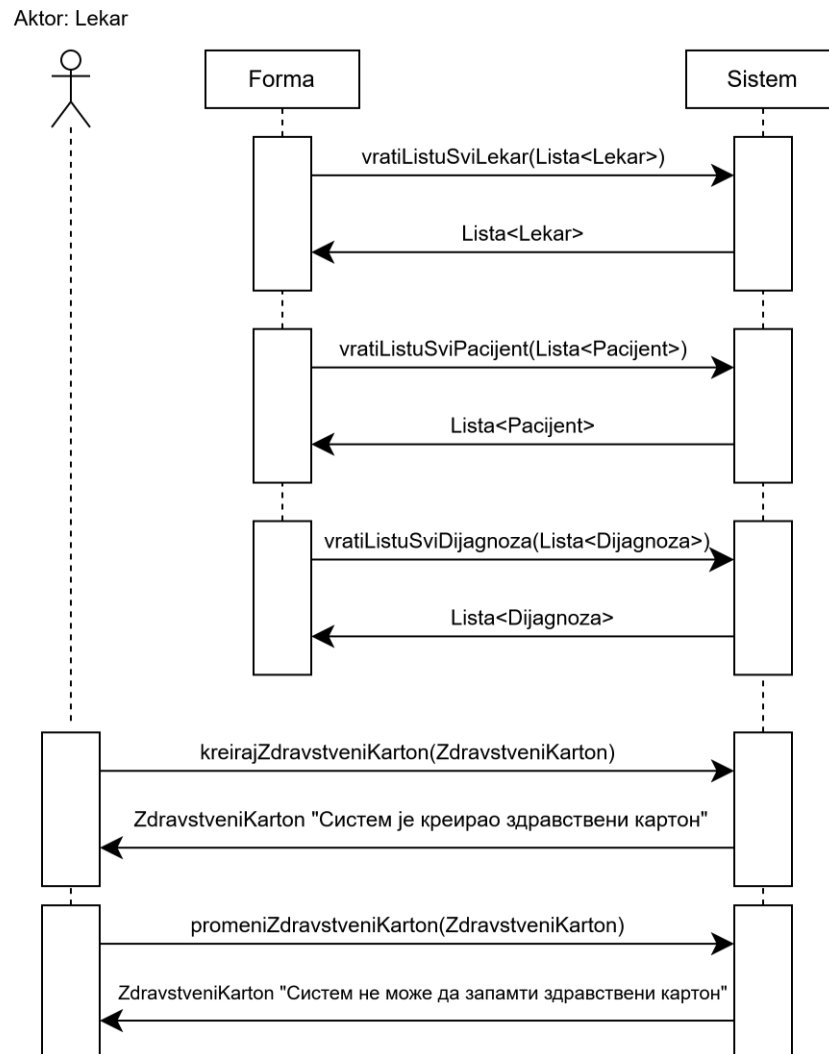


### Алтернативна сценарија:

8.1 Уколико **систем** не може да креира **здравствени картон** он **приказује** **лекару** поруку:  
“**Систем** не може да креира **здравствени картон**”. Прекида се извршење сценарија. (ИА)



10.1 Уколико **систем** не може да запамти податке о **здравственом картону** он **приказује** **лекару** поруку: “**Систем** не може да запамти **здравствени картон**”. (ИА)



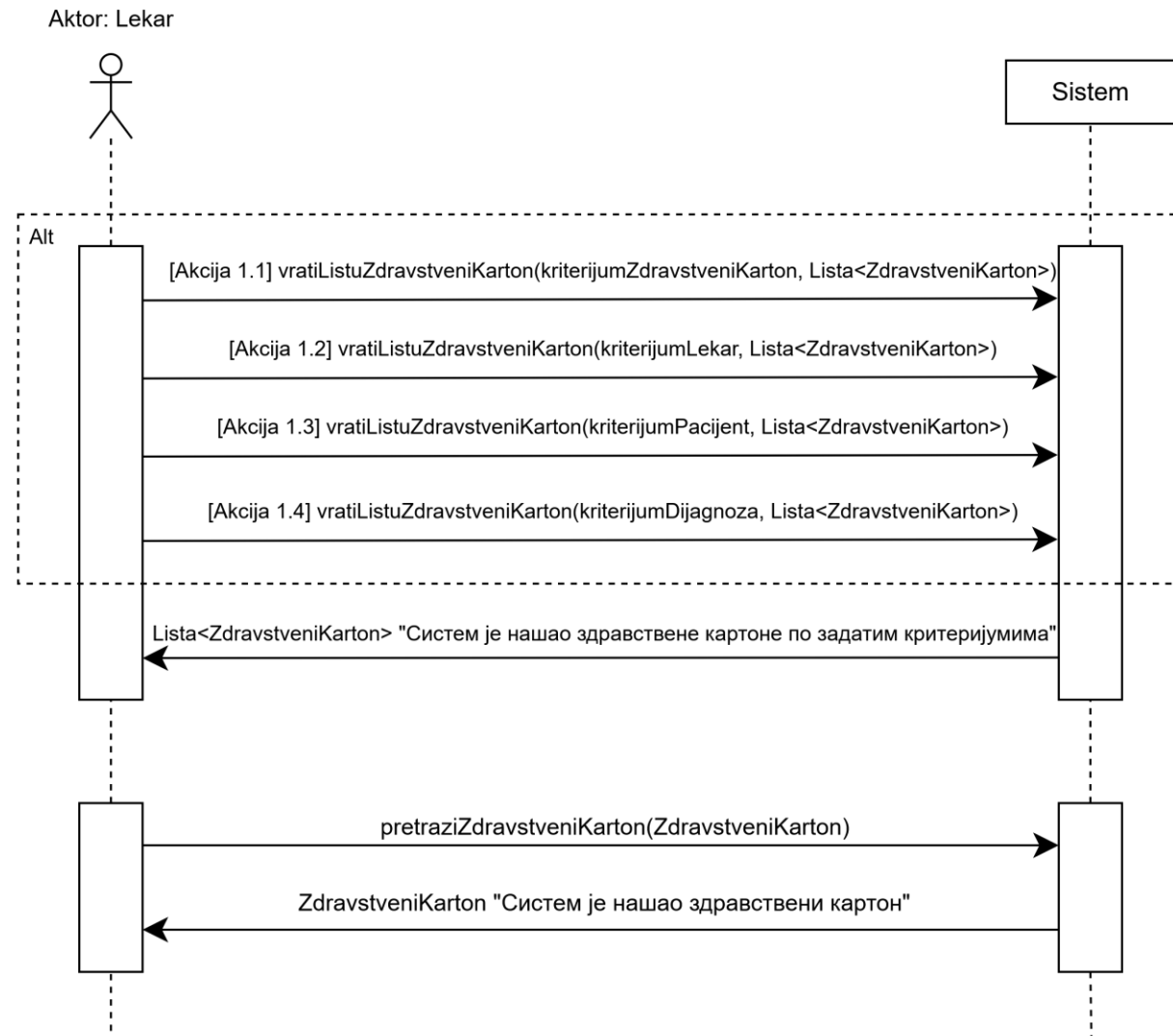
Са наведених секвенчних дијаграма уочавају се 5 системских операција које треба пројектовати:

1.	signal KreirajZdravstveniKarton(ZdravstveniKarton)
2.	signal PromeniZdravstveniKarton(ZdravstveniKarton)
3.	signal vratiListuSviLekar(Lista<Lekar>)
4.	signal vratiListuSviPacijent(Lista<Pacijent>)
5.	signal vratiListuSviDijagnoza(Lista<Dijagnoza>)

## ДС2: Дијаграми секвенци случаја коришћења – Претражи здравствени картон

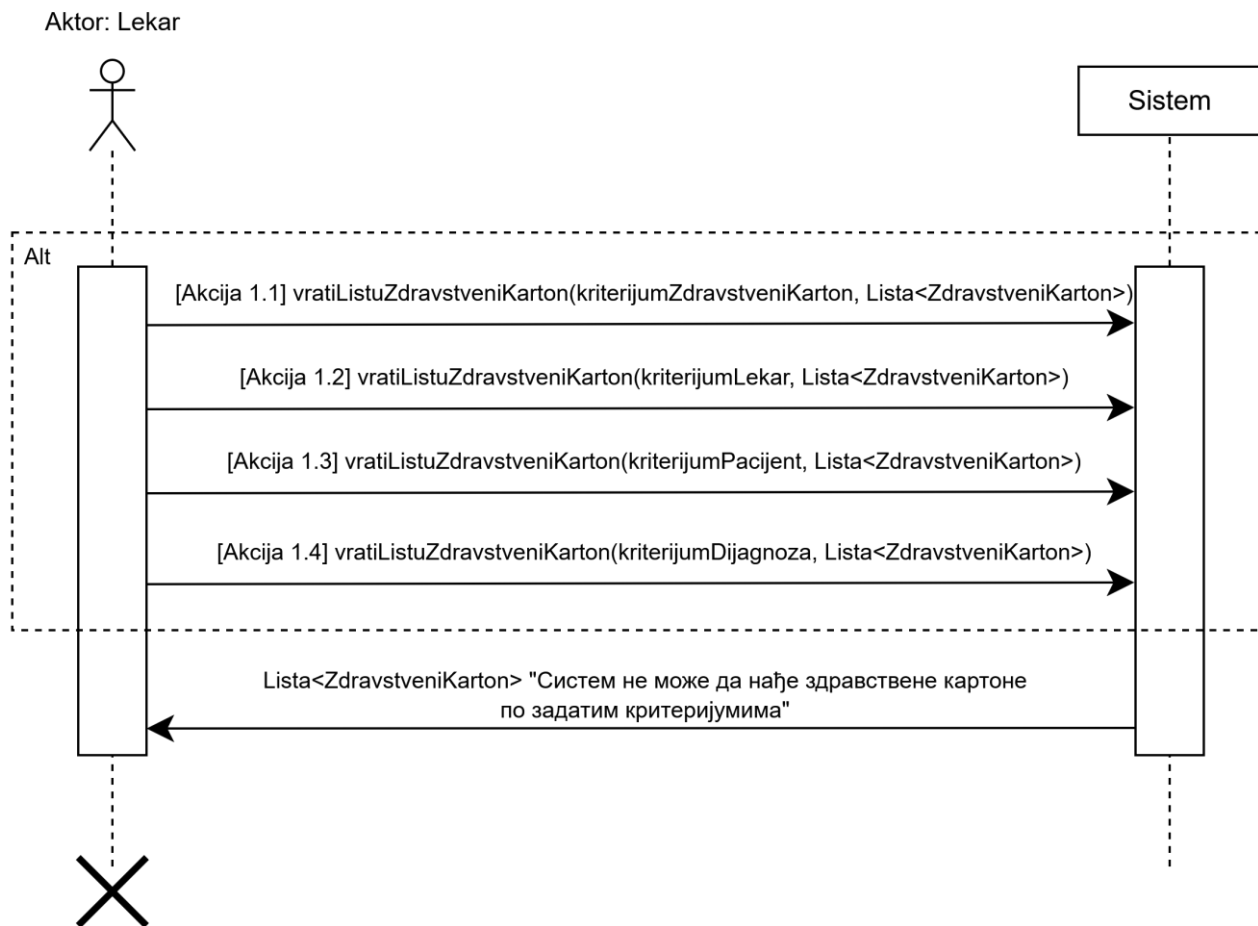
### Основни сценарио СК:

1. **Лекар** позива **систем** да нађе **здравствене картоне** по задатим критеријумима. (АПСО)
2. **Систем** приказује **лекару** **здравствене картоне** и поруку: "**Систем** је нашао **здравствене картоне** по задатим критеријумима". (ИА)
3. **Лекар** позива **систем** да нађе **здравствени картон**. (АПСО)
4. **Систем** приказује **лекару** **здравствени картон** и поруку: "**Систем** је нашао **здравствени картон**". (ИА)



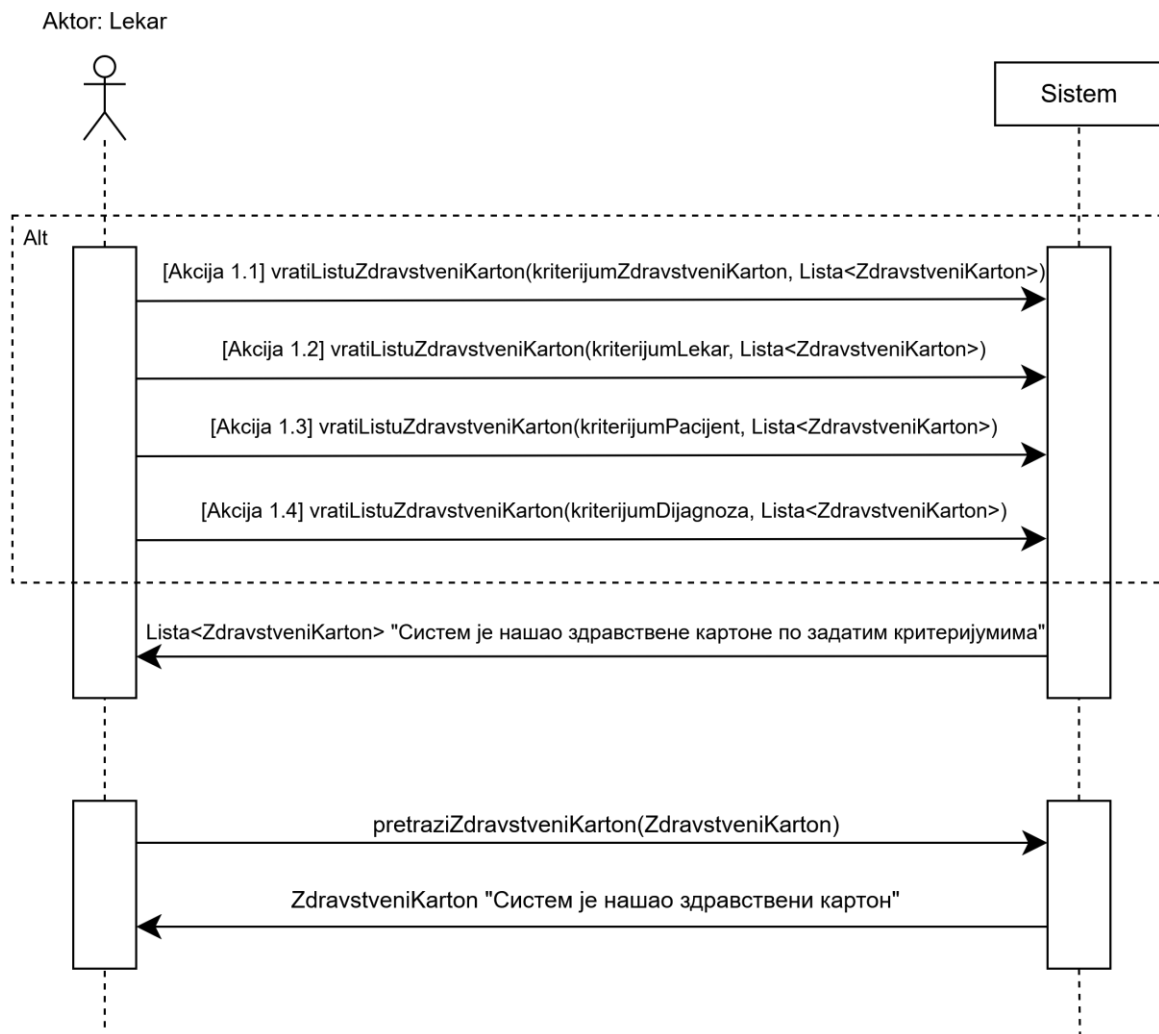
### Алтернативна сценарија:

2.1 Уколико **систем** не може да нађе **здравствени картон** он **приказује лекару** поруку: “**Систем** не може да нађе **здравствене картоне** по задатим критеријумима”. Прекида се извршење сценарија. (ИА)





4.1 Уколико **систем** не може да нађе **здравствени картон** он **приказује лекару** поруку: “**Систем** не може да нађе **здравствени картон**”.(ИА)



Са наведених секвенцих дијаграма уочавају се 5 системских операција које треба пројектовати:

1.	<b>signal PretraziZdravstveniKarton(ZdravstveniKarton)</b>
2.	<b>signal vratiListuZdravstveniKarton(kriterijumZdravstveniKarton, Lista&lt;ZdravstveniKarton&gt;)</b>
3.	<b>signal vratiListuZdravstveniKarton(kriterijumLekar, Lista&lt;ZdravstveniKarton&gt;)</b>
4.	<b>signal vratiListuZdravstveniKarton(kriterijumPacijent, Lista&lt;ZdravstveniKarton&gt;)</b>
5.	<b>signal vratiListuZdravstveniKarton(kriterijumDijagnoza, Lista&lt;ZdravstveniKarton&gt;)</b>

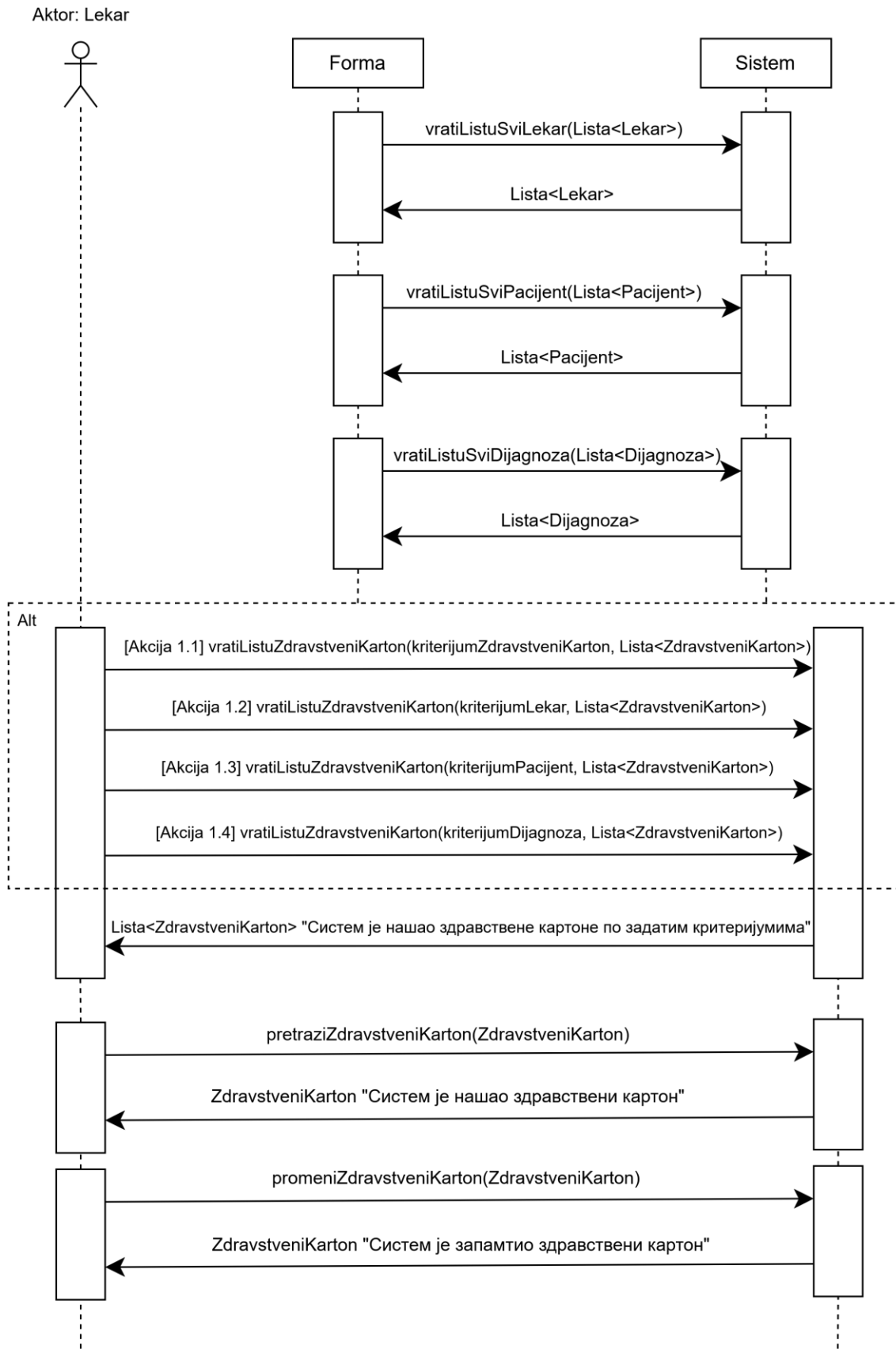
### ДСЗ: Дијаграми секвенци случаја коришћења – Промени здравствени картон

#### Предуслови:

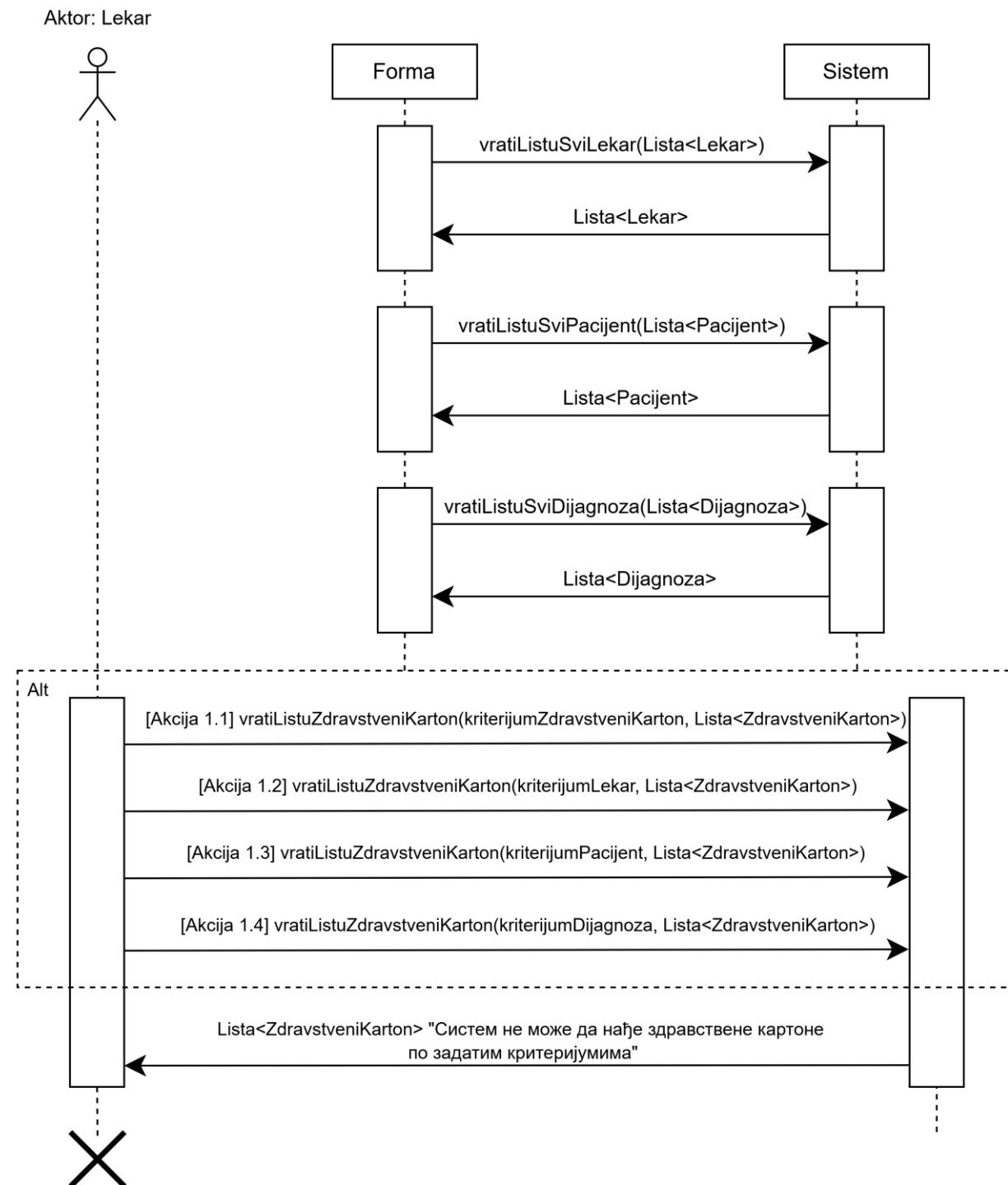
1. **Форма позива** систем да **врати** листу свих лекара. (АПСО)
2. **Систем враћа форми** листу свих лекара. (ИА)
3. **Форма позива** систем да **врати** листу свих пацијената. (АПСО)
4. **Систем враћа форми** листу свих пацијената. (ИА)
5. **Форма позива** систем да **врати** листу свих дијагноза. (АПСО)
6. **Систем враћа форми** листу свих дијагноза. (ИА)

#### Основни сценарио СК:

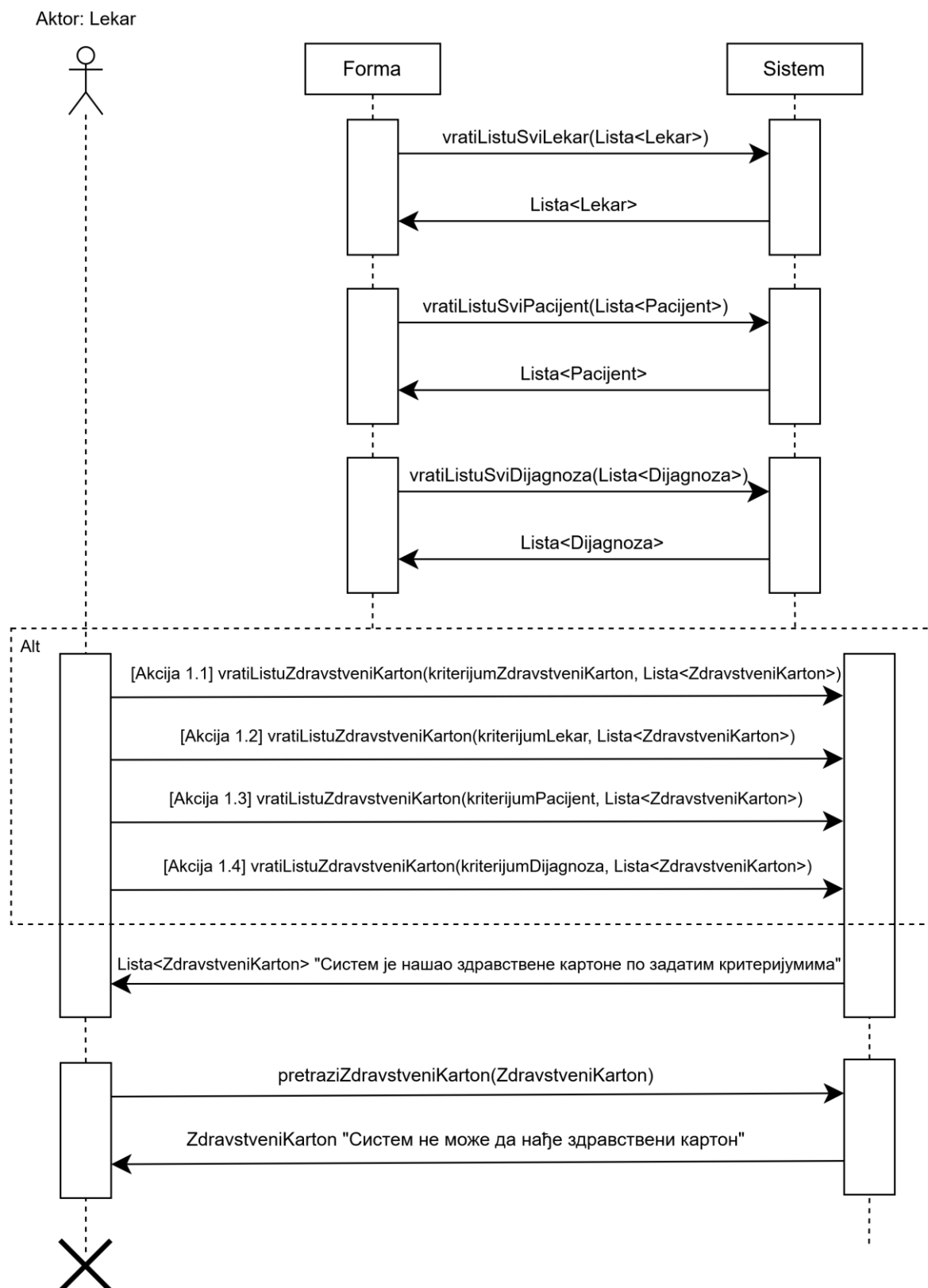
7. **Лекар позива систем** да нађе **здравствене картоне** по задатим критеријумима. (АПСО)
8. **Систем приказује лекару здравствене картоне** и поруку: “**Систем** је нашао **здравствене картоне** по задатим критеријумима”. (ИА)
9. **Лекар позива систем** да нађе **здравствени картон**. (АПСО)
10. **Систем приказује лекару здравствени картон** и поруку: “**Систем** је нашао **здравствени картон**”. (ИА)
11. **Лекар позива систем** да запамти податке о **здравственом картону**. (АПСО)
12. **Систем приказује лекару здравствени картон** и поруку: “**Систем** је запамтио **здравствени картон**.” (ИА)



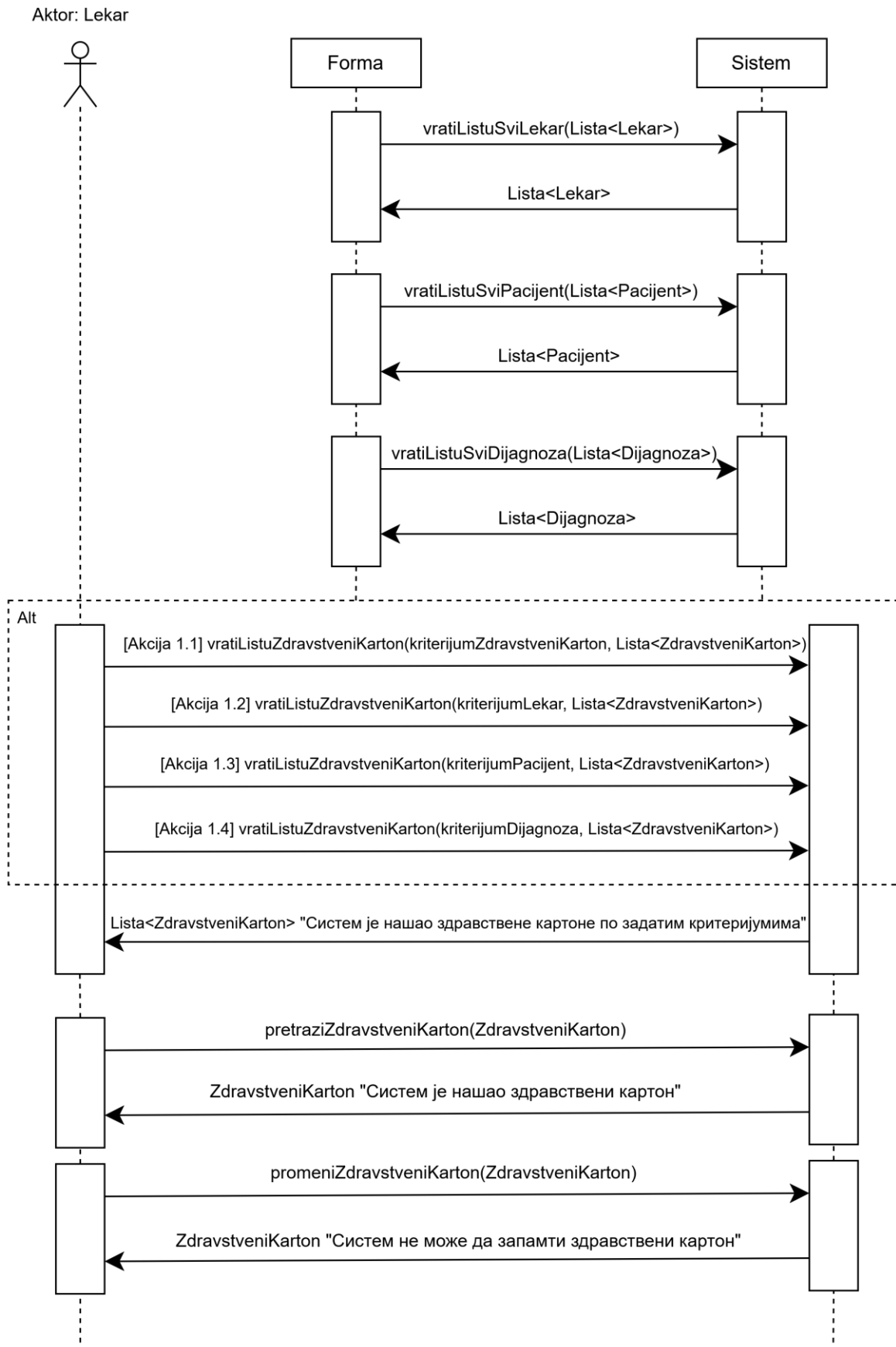
8.1 Уколико **систем** не може да нађе **здравствене картоне** он **приказује лекару** поруку: “**Систем** не може да нађе **здравствене картоне** по задатом критеријуму”. Прекида се извршење сценарија. (ИА)



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



12.1 Уколико **систем** не може да запамти податке о **здравственом картону** он **приказује** **лекару** поруку: “**Систем** не може да запамти **здравствени картон**”. (ИА)



Са наведених секвенчних дијаграма уочавају се 9 системских операција које треба пројектовати:

1.	signal PromeniZdravstveniKarton(ZdravstveniKarton)
2.	signal PretraziZdravstveniKarton(ZdravstveniKarton)
3.	signal vratiListuSviLekar(Lista)
4.	signal vratiListuSviPacijent(Lista)
5.	signal vratiListuSviDijagnoza(Lista)
6.	signal vratiListuZdravstveniKarton(kriterijumZdravstveniKarton, Lista)
7.	signal vratiListuZdravstveniKarton(kriterijumLekar, Lista)
8.	signal vratiListuZdravstveniKarton(kriterijumPacijent, Lista)
9.	signal vratiListuZdravstveniKarton(kriterijumDijagnoza, Lista)

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

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

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

**Операција:** PrijaviLekar (Korisnickolme, Sifra):signal;

**Веза са СК:** СК9

**Предуслови:**

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

#### **2. Уговор UG2: KreirajZdravstveniKarton(ZdravstveniKarton)**

**Операција:** KreirajZdravstveniKarton(ZdravstveniKarton):signal;

**Веза са СК:** СК1

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

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

#### **3. Уговор UG3: UbaciSertifikat(Sertifikat)**

**Операција:** UbaciSertifikat(Sertifikat):signal;

**Веза са СК:** СК22

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

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

#### **4. Уговор UG4: PromeniPacijent(Pacijent)**

**Операција:** PromeniPacijent(Pacijent):signal;

**Веза са СК:** СК5, СК7

**Предуслови:** Структурна и вредносна ограничење над објектом класе Пацијент морају бити задовољена.<sup>1</sup>

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

---

<sup>1</sup> Ако је објекат класе Пацијент сторниран не може се извршити системска операција. Ако је објекат класе Пацијент обрађен не може се извршити системска операција осим превођења објекта класе Пацијент у стање сторниран.



#### **5. Уговор UG5: ObrisiPacijent(Pacijent)**

**Операција:** ObrisiPacijent(*Pacijent*):signal;

**Веза са СК:** СК8

**Предуслови** Структурна и вредносна ограничење над објектом класе Пацијент морају бити задовољена.<sup>2</sup>

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

#### **6. Уговор UG6: PretraziLekar**

**Операција:** PretraziLekar(*X*):signal;

**Веза са СК:** СК11, СК12

**Предуслови:**

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

#### **7. Уговор UG7: vratiListuZdravstveniKarton**

**Операција:** vratiListuZdravstveniKarton

(*KriterijumZdravstveniKarton, Lista<ZdravstveniKarton>*):signal;

**Веза са СК:** СК2, СК4

**Предуслови:**

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

#### **8. Уговор UG8: vratiListuSviPacijent**

**Операција:** vratiListuSviPacijent (*Lista<Pacijent>*):signal;

**Веза са СК:** СК1, СК3

**Предуслови:**

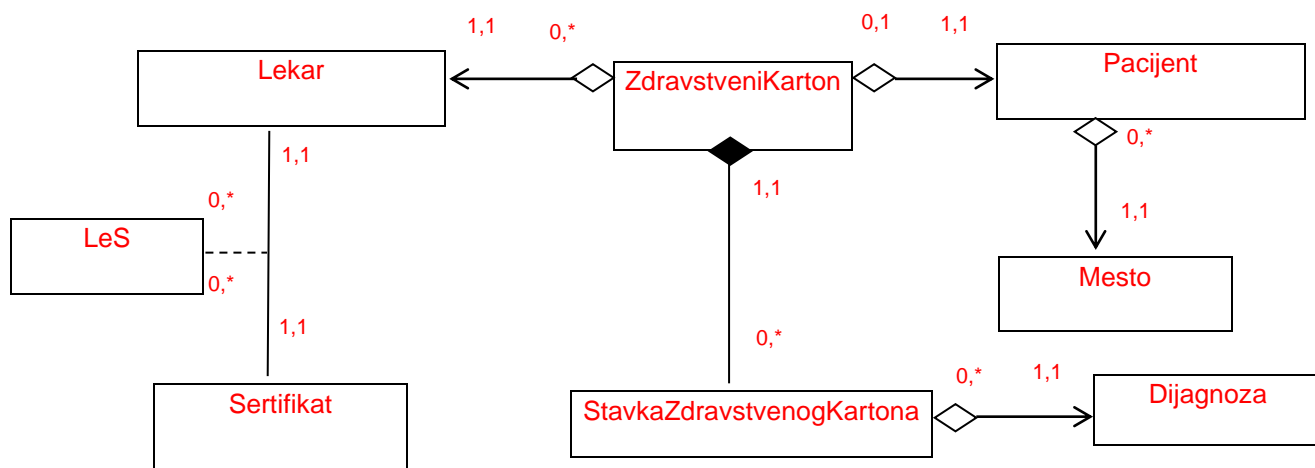
**Постуслови:** Пронађена је листа свих објеката класе Пацијент.

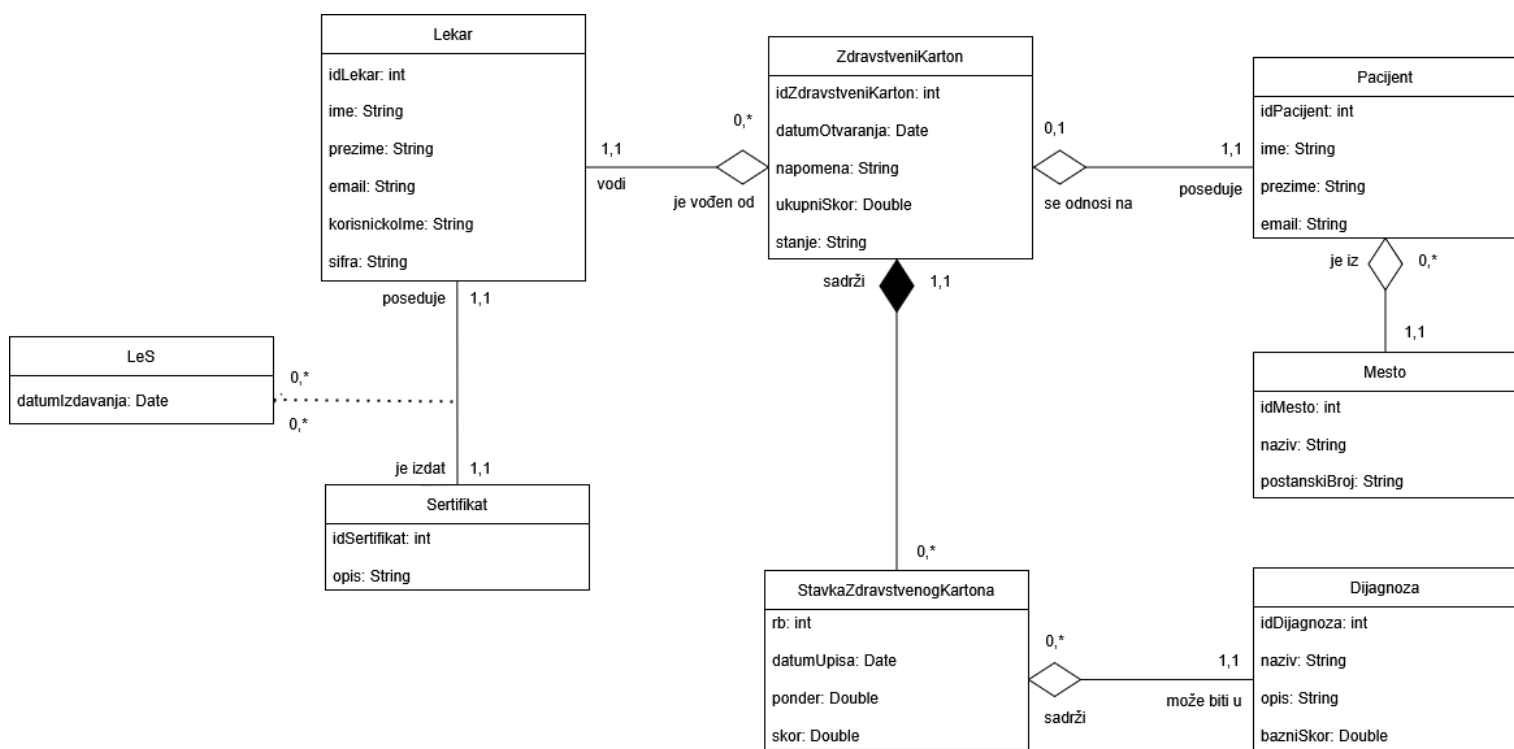
---

<sup>2</sup> Ако је објекат класе Пацијент обрађен или сторниран не може се извршити системска операција.

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

Тип концептуалног модела 1:





### 3.5 Структура софтверског система – Релациони модел

На основу концептуалног модела се прави релациони модел.

Релациони модел добијен из првог типа концептуалног модела:

1. **Lekar** (idLekar, ime, prezime, email, korisnickoIme, sifra)
2. **Dijagnoza** (idDijagnoza, naziv, opis, bazniSkor)
3. **Mesto** (idMesto, naziv, postanskiBroj)
4. **Sertifikat** (idSertifikat, opis)
5. **Pacijent** (idPacijent, ime, prezime, email, *idMesto*)
6. **ZdravstveniKarton** (idZdravstveniKarton, datumOtvaranja, napomena, ukupniSkor, stanje, *idLekar*, *idPacijent*)
7. **StavkaZdravstvenogKartona** (idZdravstveniKarton, rb, datumUpisa, ponder, skor, *idDijagnoza*)
8. **LeS** (idLekar, idSertifikat, datumIzdavanja)

### 3.6 Табела структурних и вредносних ограничења релационог модела

За сваку релацију се прави табела структурних и вредносних ограничења.

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

1. Табела <b>Lekar</b>		Просто вредносно ограничење		Сложено вредносно ограничење		Структурно ограничење
Атрибути	Назив	Тип атрибута	Вредност атрибута	Међузав. атрибута једне табеле	Међузав. атрибута више табела	INSERT /  UPDATE CASCADES Zdravstveni karton , LeS  DELETE RESTRICTED Zdravstveni karton , LeS
	<u>idLekar</u>	Integer	NOT NULL AND > 0			
	ime	String	NOT NULL			
	prezime	String	NOT NULL			
	email	String	NOT NULL AND LIKE ' % @ % . % '			
	korisnickoIme	String	NOT NULL			
	sifra	String	NOT NULL			

2. Табела <b>Dijagnoza</b>		Просто вредносно ограничење		Сложено вредносно ограничење		Структурно ограничење
Атрибути	Назив	Тип атрибута	Вредност атрибута	Међузав. атрибута једне табеле	Међузав. атрибута више табела	INSERT /  UPDATE CASCADES Stavka zdravstvenog kartona ,  DELETE RESTRICTED Stavka zdravstvenog kartona ,
	<u>idDijagnoza</u>	Integer	NOT NULL AND > 0			
	naziv	String	NOT NULL			
	opis	String	NOT NULL			
	bazniSkor	Double	NOT NULL AND > 0			

3. Табела <b>Mesto</b>		Просто вредносно ограничење		Сложено вредносно ограничење		Структурно ограничење
Атрибути	Назив	Тип атрибута	Вредност атрибута	Међузав. атрибута једне табеле	Међузав. атрибута више табела	INSERT /  UPDATE CASCADES Pacijent ,  DELETE RESTRICTED Pacijent ,
	<u>idMesto</u>	Integer	NOT NULL AND > 0			
	naziv	String	NOT NULL			
	postanskiBroj	String	NOT NULL AND LIKE '[0-9][0-9][0-9][0-9][0-9]'			

4. Табела <b>Sertifikat</b>		Просто вредносно ограничење		Сложено вредносно ограничење		Структурно ограничење
Атрибути	Назив	Тип атрибута	Вредност атрибута	Међузав. атрибута једне табеле	Међузав. атрибута више табела	INSERT /  UPDATE CASCADES <b>LeS</b> ,  DELETE RESTRICTED <b>LeS</b> ,
	<u>idSertifikat</u>	Integer	NOT NULL AND > 0			
	opis	String	NOT NULL			

5. Табела <b>Pacijent</b>		Просто вредносно ограничење		Сложено вредносно ограничење		Структурно ограничење
Атрибути	Назив	Тип атрибута	Вредност атрибута	Међузав. атрибута једне табеле	Међузав. атрибута више табела	INSERT RESTRICTED <b>Mesto</b>  UPDATE RESTRICTED <b>Mesto</b>  UPDATE CASCADES <b>Zdravstveni karton</b>  DELETE RESTRICTED <b>Zdravstveni karton</b>
	<u>idPacijent</u>	Integer	NOT NULL AND > 0			
	ime	String	NOT NULL			
	prezime	String	NOT NULL			
	email	String	NOT NULL AND LIKE '_%@_%.%'			
	idMesto	Integer	NOT NULL AND > 0			

6. Табела <b>ZdravstveniKarton</b>		Просто вредносно ограничење		Сложено вредносно ограничење		Структурно ограничење
Атрибути	Назив	Тип атрибута	Вредност атрибута	Међузав. атрибута једне табеле	Међузав. атрибута више табела	INSERT RESTRICTED <b>Pacijent Lekar</b>  UPDATE RESTRICTED <b>Pacijent Lekar</b>  UPDATE CASCADES <b>Stavka zdravstvenog kartona</b>  DELETE RESTRICTED <b>Stavka zdravstvenog kartona</b>
	<u>idZdravstveniKarton</u>	Integer	NOT NULL AND > 0			
	datumOtvaranja	Date	NOT NULL			
	napomena	String	NOT NULL			
	ukupniSkor	Double	NOT NULL (default:0)		ukupniSkor = SUM ( StavkaZdravstvenogKartona.skor )	
	stanje	String	NOT NULL AND IN ( 'Zdrav', 'Lakše bolestan', 'Teže bolestan' )	stanje = CASE WHEN ukupniSkor BETWEEN (0, 100) THEN 'Zdrav' WHEN ukupniSkor BETWEEN (100, 500) THEN 'Lakše bolestan' CASE WHEN ukupniSkor > 500 THEN 'Teže bolestan'		
	idLekar	Integer	NOT NULL AND > 0			
	idPacijent	Integer	NOT NULL AND > 0			

7.Табела StavkaZdravstvenog Kartona		Просто вредносно ограничење		Сложено вредносно ограничење		Структурно ограничење
Атрибу ти	Назив	Тип атрибуа	Вредност атрибуа	Међуав. атрибуа једне табеле	Међуав.атрибуа више табела	INSERT RESTRICTED Zdravstveni karton Dijagnoza  UPDATE RESTRICTED Zdravstveni karton Dijagnoza  DELETE /
	<u>idZdravstve niKarton</u>	Integer	NOT NULL AND > 0			
	<u>rb</u>	Integer	NOT NULL AND > 0			
	datumUpisa	Date	NOT NULL			
	ponder	Double	NOT NULL AND > 0 (default:1)			
	skor	Double	NOT NULL (default:0)		skor = Dijagnoza.bazniSkor * ponder	
	idDijagnoza	Integer	NOT NULL AND > 0			

8.Табела LeS		Просто вредносно ограничење		Сложено вредносно ограничење		Структурно ограничење
Атрибу ти	Назив	Тип атрибуа	Вредност атрибуа	Међуав. атрибуа једне табеле	Међуав.атрибуа више табела	INSERT RESTRICTED Lekar Sertifikat  UPDATE RESTRICTED  Lekar Sertifikat  DELETE /
	<u>idLekar</u>	Integer	NOT NULL AND > 0			
	<u>idSertifikat</u>	Integer	NOT NULL AND > 0			
	datumIzdavanja	Date	NOT NULL			

## 4. Пројектовање

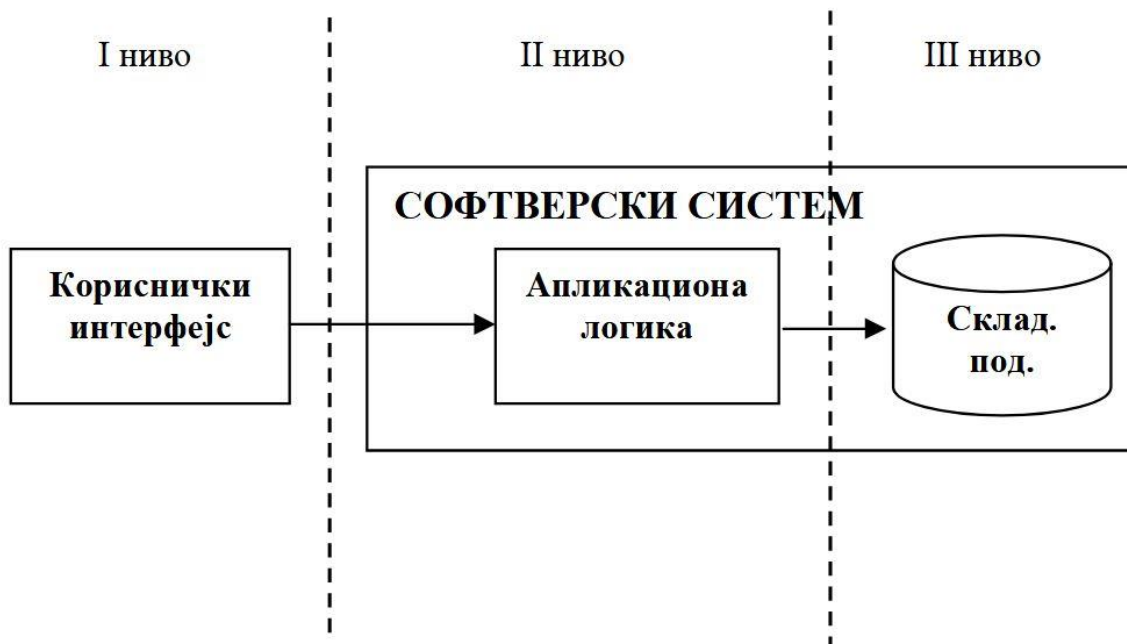
Фаза пројектовања описује физичку структуру и понашање софтверског система (архитектура софтверског система).

Пројектовање архитектуре софтверског система обухвата пројектовање корисничког интерфејса (пројектовање контролера корисничког интерфејса и екранских форми), апликационе логике (пројектовање контролера апликационе логике и пословне логике) и складишта података.

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

- Кориснички интерфејс
- Апликациона логика
- Складиште података

Ниво корисничког интерфејса је на страни корисника, а апликациона логика и складиште података на страни сервера.

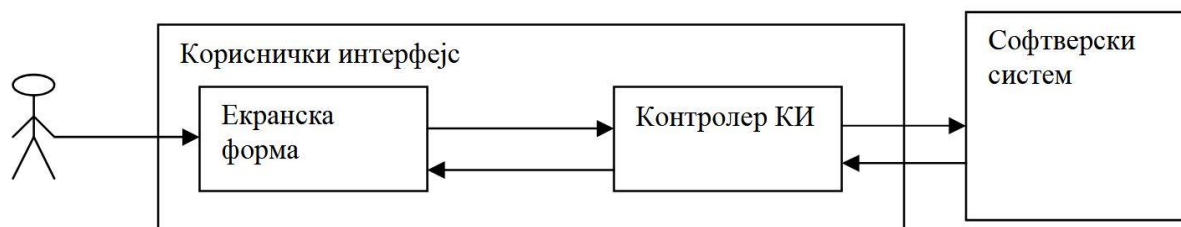




## 4.1 Пројектовање корисничког интерфејса

Кориснички интерфејс представља улазно-излазну реализацију софтверског система и састоји се од:

- Екранске форме и
- Контролера корисничког интерфејса



Корисник комуницира са екранском формом, помоћу графичких компоненти, која прихвата унете податке и догађаје корисника, позива контролера корисничког интерфејса и прослеђује му те информације. Контролер је одговоран за конверзију ових података у одговарајуће доменске класе и да захтева извршење системских операција. Такође, контролер прима податке од софтверског система, које затим преводи у елементе екранске форме.

### Пројектовање екранских форми

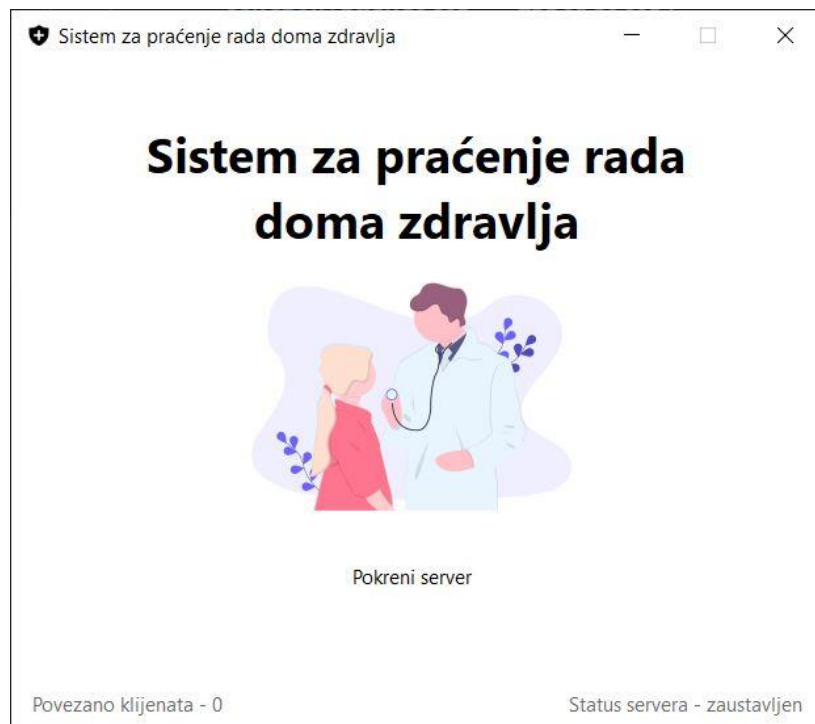
Кориснички интерфејс је дефинисан преко скупа екранских форми. Сценарио коришћења екранских форми је директно повезан са сценаријима случајева коришћења.

Постоје два аспекта пројектовања екранске форме:

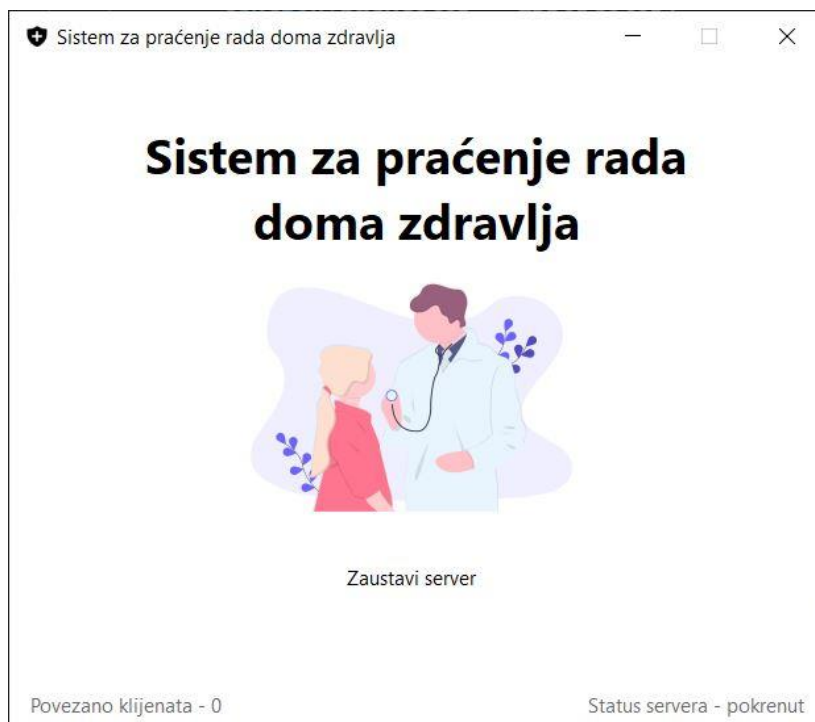
- Пројектовање сценарија СК који се изводе преко екранске форме
- Пројектовање метода екранске форме

Пројектовање екранске форме треба да постигне равнотежу између техничке апстракције и корисничке разумљивости. Добра форма не служи само као средство за унос и приказ података, већ као елегантан мост између корисника и сложености система. Сценарија случајева коришћења треба да буду интуитивно уткана у структуру форме, док методе које их подржавају морају бити логично организоване, ненаметљиве и поуздане. Циљ је да се кроз једноставност и јасноћу обезбеди ефективна интеракција, без нарушавања уредности и одрживости.

На серверској страни програма налази се форма која регулише стање сервера и приликом покретања програма, она изгледа овако:



Кликом на дугме "Pokreni server", покреће се сервер, серверски сокет ослушкује пристизање клијената, а главна серверска форма изгледа овако:



Кликом на дугме "Zaustavi server", сервер се зауставља и престаје да обрађује захтеве корисника.

На клијентској страни прво је потребно пријавити се како би се доспело у могућност коришћења апликације. Форма за пријаву изгледа овако:

The screenshot shows a web browser window titled 'Sistem za praćenje rada doma zdravlja | Prijava'. Below the title bar is a navigation bar with two links: 'Podešavanja softverskog sistema' and 'O programu'. The main content area contains a login form on the left with two input fields labeled 'Korisničko ime:' and 'Sifra:', followed by a 'Prijava se' button. On the right is a large illustration of two healthcare professionals in white coats standing next to a potted plant, with a large blue heart and a white ECG line in the background.

Након успешне пријаве на систем, клијенту се приказује главна клијентска екранска форма из које се може доћи до осталих форми за рад:

The screenshot shows the main dashboard of the application. The title bar is 'Sistem za praćenje rada doma zdravlja'. Below it is a navigation bar with four links: 'Dokumenti', 'Pružalac usluge', 'Primalac usluge', and 'Šifarnici'. The main content area features an illustration of two healthcare professionals on the left. On the right, there is a welcome message: 'Dobro došli, Jovan', followed by a paragraph: 'Želimo Vam uspešan i prijatan rad u sistemu. Vaši podaci su ažurirani, a pacijenti Vas očekuju.', and an 'Odjavi se' button at the bottom.

## СК1- Креирај здравствени картон

### Назив СК

Креирај здравствени картон

### Актери СК

Лекар

### Учесници СК

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

**Предуслови:** Кориснички интерфејс (клијентски програм) и систем (серверски програм) су покренути. Лекар је пријављен под својом шифром. Систем приказује форму за рад са здравственим картоном. Учитане су листе: а) Лекар б) Пацијент с) Дијагноза

Sistem za praćenje rada doma zdravlja | Zdravstveni karton

Kriterijumi pretrage

Otvoren nakon:

June 2025

Sun	Mon	Tue	Wed	Thu	Fri	Sat
25	26	27	28	29	30	31
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	1	2	3	4	5

Today: 6/20/2025

Ime lekara:

Ime pacijenta:

Sadrži dijagnozu:

-- Bez izbora --

Akcije

Pretraži

Kreiraj novi

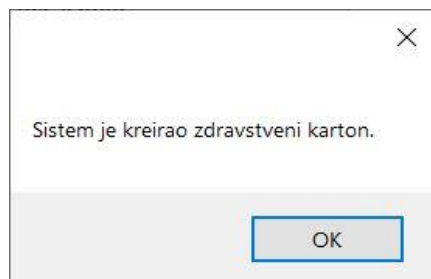
Detalji

### Основни сценарио СК:

1. Лекар позива систем да креира здравствени картон. (АПСО)

**Опис акције:** Лекар кликом на дугме “Kreiraj novi” позива системску операцију KreirajZdravstveniKarton(Zdravstveni karton).

2. Систем креира здравствени картон. (СО)
3. Систем приказује лекару здравствени картон и поруку: “Систем је креирао здравствени картон”. (ИА)



4. **Лекар** уноси податке о **здравственом картону**. (АПУСО)

Sistem za praćenje rada doma zdravlja | Zdravstveni karton

Lekar:  
Jovan Petrovic

Pacijent:  
Martin Jovanović

Napomena:  
Pacijent ima izražene simptome astme praćene težim oblikom bronhitisa.

Dijagnoze:  
Hronični bronhitis  
Ponder: 4  
Astma | 2.5  
Hronični bronhitis | 4

Zapamti
Obrisi
Odustani

Unos novog zdravstvenog kartona

5. **Лекар** контролише да ли је коректно унео податке о **здравственом картону**. (АНСО)

6. **Лекар** позива **систем** да запамти податке о **здравственом картону**. (АПСО)

Опис акције: **Лекар** кликом на дугме “Zapamti” позива системску операцију **PromeniZdravstveniKarton(Zdravstveni karton)**.

7. **Систем** памти податке о **здравственом картону**. (СО)

8. **Систем** приказује **лекару** **здравствени картон** и поруку: “**Систем** је запамтио **здравствени картон**.” (ИА)

Sistem je zapamtio zdravstveni karton.

OK

Sistem za praćenje rada doma zdravlja | Zdravstveni karton

Kriterijumi pretrage  
Otvoren nakon:  
June 2025  
Sun Mon Tue Wed Thu Fri Sat  
25 26 27 28 29 30 31  
1 2 3 4 5 6 7  
8 9 10 11 12 13 14  
15 16 17 18 19 20 21  
22 23 24 25 26 27 28  
29 30 1 2 3 4 5  
Today: 6/20/2025

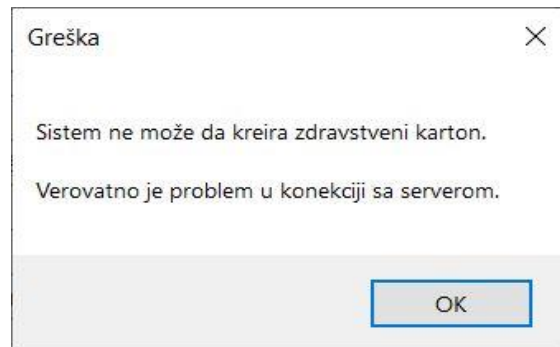
Ime lekara:  
Ime pacijenta:  
Sadrži dijagnozu:  
Astma

Akcije  
Pretraži  
Kreiraj novi  
Detalji

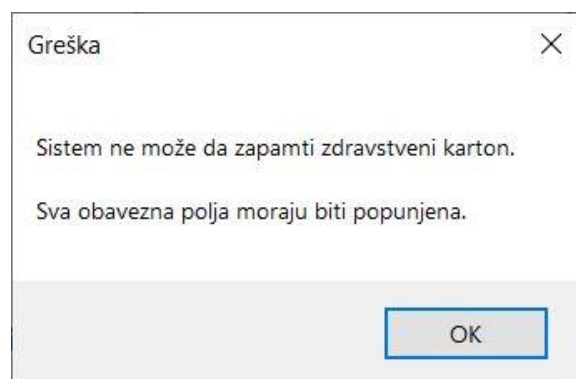
	Id	DatumOtvaranja	Napomena	Stanje	Uku
▶	3	6/20/2025	Periodični problemi sa disanjem, preporučena dodatna pulmološka dijagnostika.	Lakše bolestan	345
	7	6/20/2025	Pacijent ima izražene simptome astme praćene težim oblikom bronhitisa.	Teže bolestan	965

### Алтернативна сценарија:

3.1 Уколико **систем** не може да креира **здравствени картон** он **приказује лекару** поруку: “**Систем** не може да креира **здравствени картон**”. Прекида се извршење сценарија. (ИА)



8.1 Уколико **систем** не може да запамти податке о **здравственом картону** он **приказује лекару** поруку: “**Систем** не може да запамти **здравствени картон**”. (ИА)



## СК2- Претражи здравствени картон

### Назив СК

Претражи здравствени картон

### Актери СК

Лекар

### Учесници СК

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

**Предуслови:** Кориснички интерфејс (клијентски програм) и систем (серверски програм) су покренути. Лекар је пријављен под својом шифром. Кориснички интерфејс приказује форму за рад са здравственим картоном. На наведеној екранској форми су дефинисани критеријуми, који се односе на: а) Здравствени картон б) Лекар с) Пацијент д) Дијагноза, који ће да врате листу здравствених картона.

Sistem za praćenje rada doma zdravlja | Zdravstveni karton

Kriterijumi pretrage

Otvoren nakon:

June 2025

Sun	Mon	Tue	Wed	Thu	Fri	Sat
25	26	27	28	29	30	31
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	1	2	3	4	5

Today: 6/20/2025

Ime lekara:

Ime pacijenta:

Sadrži dijagnozu:

-- Bez izbora --

Akcije

Pretraži

Kreiraj novi

Detalji

### Основни сценарио СК:

1. Лекар бира критеријуме на основу којих претражује здравствене картоне. (АПУСО)

Sistem za praćenje rada doma zdravlja | Zdravstveni karton

Kriterijumi pretrage

Otvoren nakon:

June 2025

Sun	Mon	Tue	Wed	Thu	Fri	Sat
25	26	27	28	29	30	31
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	1	2	3	4	5

Today: 6/20/2025

Ime lekara:

Ime pacijenta:

Sadrži dijagnozu:

Asthma

Akcije

Pretraži

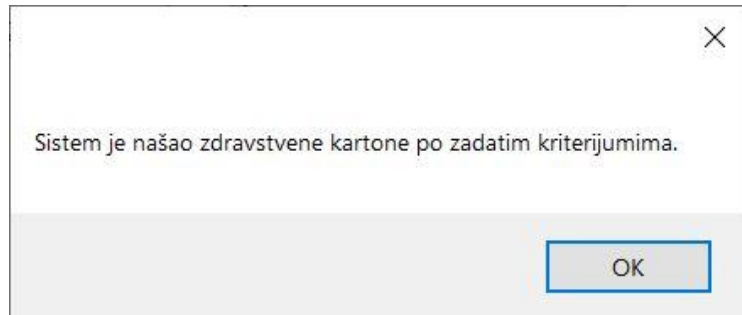
Kreiraj novi

Detalji

2. **Лекар** позива **систем** да нађе **здравствене картоне** по задатим критеријумима. (АПСО)

**Опис акције:** **Лекар** кликом на дугме “Pretraži” позива системске операције за враћање листе **здравствених картона**.

3. **Систем** тражи **здравствене картоне** по задатим критеријумима. (СО)  
4. **Систем** приказује **лекару** **здравствене картоне** и поруку: “**Систем** је нашао **здравствене картоне** по задатим критеријумима”. (ИА)



Sistem za praćenje rada doma zdravlja | Zdravstveni karton

Kriterijumi pretrage

Otvoren nakon:

June 2025

Sun	Mon	Tue	Wed	Thu	Fri	Sat
25	26	27	28	29	30	31
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	1	2	3	4	5

Today: 6/20/2025

Ime lekara:

Ime pacijenta:

Sadrži dijagnozu:

Astma

Akcije

Pretraži

Kreiraj novi

Detalji

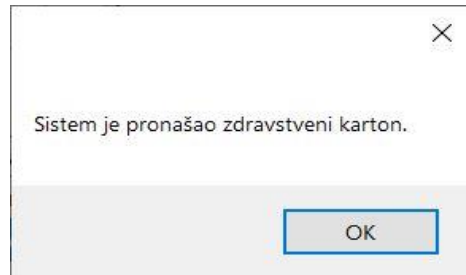
	Id	DatumOtvaranja	Napomena	Stanje	Uku.
▶	3	6/20/2025	Periodični problemi sa disanjem, preporučena dodatna pulmološka dijagnostika.	Lakše bolestan	345
	7	6/20/2025	Pacijent ima izražene simptome astme praćene težim oblikom bronhitisa.	Teže bolestan	965

5. **Лекар** бира **здравствени картон**. (АПУСО)  
6. **Лекар** позива **систем** да нађе **здравствени картон**. (АПСО)

**Опис акције:** **Лекар** кликом на дугме “Detalji” позива системску операцију **PretraziZdravstveniKarton(Zdravstveni karton)**.



7. Систем тражи здравствени картон. (CO)
8. Систем приказује лекару здравствени картон и поруку: “Систем је нашао здравствени картон”. (IA)



Sistem za praćenje rada doma zdravlja | Zdravstveni karton

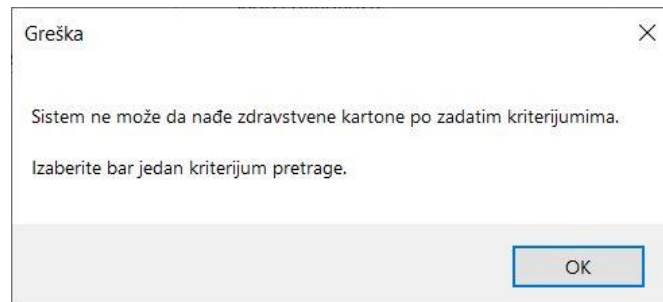
Lekar:	Pacijent:
Jovan Petrovic	Martin Jovanović
Napomena:	Dijagnoze:
Pacijent ima izražene simptome astme praćene težim oblikom bronhitisa.	-- Bez izbora --
	Ponder: <input type="text"/> +
	Astma   2.5 -
	Hronični bronhitis   4

Zapamti      Obriši      Odustani

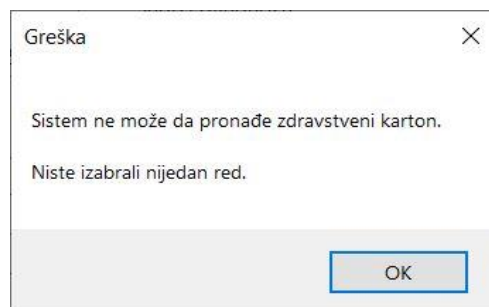
Zdravstveni karton ID: 7

### Алтернативна сценарија:

4.1 Уколико **систем** не може да нађе **здравствене картоне** он **приказује лекару** поруку: “**Систем** не може да нађе **здравствене картоне** по задатим критеријумима”. Прекида се извршење сценарија. (ИА)



8.1 Уколико **систем** не може да нађе **здравствени картон** он **приказује лекару** поруку: “**Систем** не може да нађе **здравствени картон**”. (ИА)



## СКЗ- Промени здравствени картон

### Назив СК

Промени здравствени картон

### Актори СК

Лекар

### Учесници СК

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

**Предуслови:** Кориснички интерфејс (клијентски програм) и систем (серверски програм) су покренути. Лекар је пријављен под својом шифром. Кориснички интерфејс приказује форму за рад са здравственим картоном. На наведеној екранској форми су дефинисани критеријуми, који се односе на: а) Здравствени картон б) Лекар с) Пацијент д) Дијагноза, који ће да врате листу здравствених картона. Учитане су листе: а) Лекар б) Пацијент с) Дијагноза

Sistem za praćenje rada doma zdravlja | Zdravstveni karton

Kriterijumi pretrage

Otvoren nakon:

Sun	Mon	Tue	Wed	Thu	Fri	Sat
25	26	27	28	29	30	31
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	1	2	3	4	5

Today: 6/20/2025

Ime lekara:

Ime pacijenta:

Sadrži dijagnozu:

-- Bez izbora --

Akcije

Pretraži

Kreiraj novi

Detalji

### Основни сценарио СК:

1. Лекар бира критеријуме на основу којих претражује здравствене картоне. (АПУСО)

Sistem za praćenje rada doma zdravlja | Zdravstveni karton

Kriterijumi pretrage

Otvoren nakon:

Sun	Mon	Tue	Wed	Thu	Fri	Sat
25	26	27	28	29	30	31
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	1	2	3	4	5

Today: 6/20/2025

Ime lekara:

Ime pacijenta:

Sadrži dijagnozu:

Asthma

Akcije

Pretraži

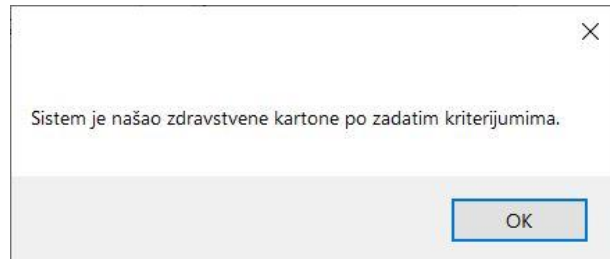
Kreiraj novi

Detalji

2. **Лекар** **позива систем** да нађе **здравствене картоне** по задатим критеријумима. (АПСО)

**Опис акције:** **Лекар** кликом на дугме “Pretraži” позива системске операције за враћање листе **здравствених картона**.

3. **Систем тражи** **здравствене картоне** по задатим критеријумима. (СО)
4. **Систем приказује лекару** **здравствене картоне** и поруку: “**Систем** је нашао **здравствене картоне** по задатим критеријумима”. (ИА)



The application window is titled "Sistem za praćenje rada doma zdravlja | Zdravstveni karton". It contains several sections:

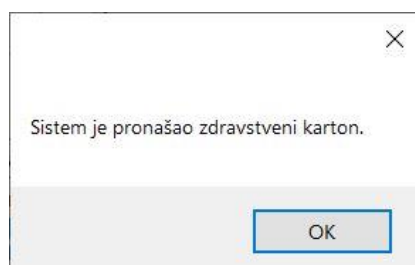
- Kriterijumi pretrage:** Includes a calendar for "June 2025" with "Today: 6/20/2025" selected. There are input fields for "Ime lekara:", "Ime pacijenta:", and a dropdown for "Sadrži dijagnozu:" with "Asthma" selected.
- Akcije:** A vertical list of buttons: "Pretraži", "Kreiraj novi", and "Detalji".
- Table:** A table with columns: Id, DatumOtvaranja, Napomena, Stanje, and Uku. It contains two rows of data.

	Id	DatumOtvaranja	Napomena	Stanje	Uku.
▶	3	6/20/2025	Periodični problemi sa disanjem, preporučena dodatna pulmološka dijagnostika.	Lakše bolestan	345
	7	6/20/2025	Pacijent ima izražene simptome astme praćene težim oblikom bronhitisa.	Teže bolestan	965

5. **Лекар бира** **здравствени картон**. (АПУСО)
6. **Лекар позива систем** да нађе **здравствени картон**. (АПСО)

**Опис акције:** **Лекар** кликом на дугме “Detalji” позива системску операцију **PretraziZdravstveniKarton(Zdravstveni karton)**.

7. **Систем тражи** **здравствени картон**. (СО)
8. **Систем приказује лекару** **здравствени картон** и поруку: “**Систем** је нашао **здравствени картон**”. (ИА)



Sistem za praćenje rada doma zdravlja | Zdravstveni karton

Lekar:  
Jovan Petrovic

Pacijent:  
Martin Jovanović

Napomena:  
Pacijent ima izražene simptome astme praćene težim oblikom bronhitisa.

Dijagnoze:  
-- Bez izbora --  
Ponder:  +  
Asthma | 2.5  
Hronični bronhitis | 4

Zapamti
Obriši
Odustani

Zdravstveni karton ID: 7

9. **Лекар** уноси (мења) податке о **здравственом картону**. (АПУСО)
10. **Лекар** контролише да ли је коректно унео податке о **здравственом картону**. (АНСО)
11. **Лекар** позива **систем** да запамти податке о **здравственом картону**. (АПСО)

Опис акције: **Лекар** кликом на дугме “Zapamti” позива системску операцију PromeniZdravstveniKarton(Zdravstveni karton).

12. **Систем** памти податке о **здравственом картону**. (СО)
13. **Систем** приказује **лекару** **здравствени картон** и поруку: “Систем је запамтио **здравствени картон**.” (ИА)

Sistem je zapamtio zdravstveni karton.

OK

Sistem za praćenje rada doma zdravlja | Zdravstveni karton

Kriterijumi pretrage:  
Otvoren nakon:  
June 2025  
Sun Mon Tue Wed Thu Fri Sat  
25 26 27 28 29 30 31  
1 2 3 4 5 6 7  
8 9 10 11 12 13 14  
15 16 17 18 19 20 21  
22 23 24 25 26 27 28  
29 30 1 2 3 4 5  
Today: 6/20/2025

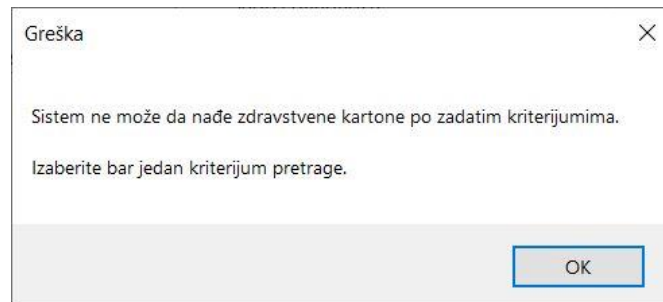
Ime lekara:  
Ime pacijenta:  
Sadrži dijagnozu:  
Asthma

Акције:  
Pretraži  
Kreiraj novi  
Detalji

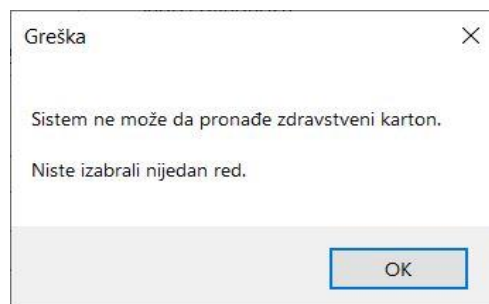
	Id	DatumOtvaranja	Napomena	Stanje	Uku.
▶	3	6/20/2025	Periodični problemi sa disanjem, preporučena dodatna pulmološka dijagnostika.	Lakše bolestan	345
	7	6/20/2025	Pacijent se bolje oseća nakon 2 nedelje terapije. Bronhitis je nestao.	Lakše bolestan	221

### Алтернативна сценарија:

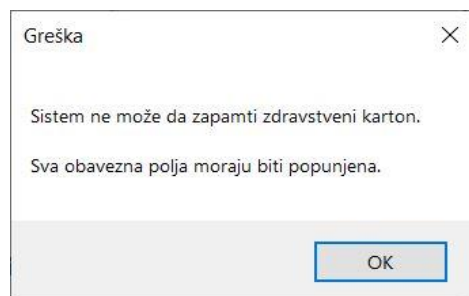
4.1 Уколико **систем** не може да нађе **здравствене картоне** он **приказује лекару** поруку: “**Систем** не може да нађе **здравствене картоне** по задатом критеријуму”. Прекида се извршење сценарија. (ИА)



8.1 Уколико **систем** не може да нађе **здравствени картон** он **приказује лекару** поруку: “**Систем** не може да нађе **здравствени картон**”. Прекида се извршење сценарија. (ИА)



13.1 Уколико **систем** не може да запамти податке о **здравственом картону** он **приказује лекару** поруку: “**Систем** не може да запамти **здравствени картон**”. (ИА)



## СК5- Креирај пацијент

### Назив СК

Креирај пацијент

### Актори СК

Лекар

### Учесници СК

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

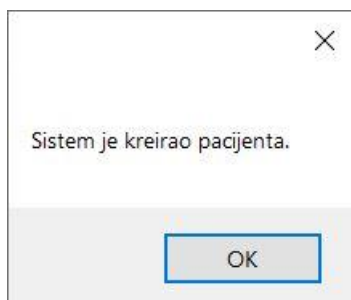
**Предуслови:** Кориснички интерфејс (клијентски програм) и систем (серверски програм) су покренути. Лекар је пријављен под својом шифром. Систем приказује форму за рад са пацијентом. *Учитане су листе: а) Место*

### Основни сценарио СК:

1. Лекар позива систем да креира пацијента. (АПСО)

**Опис акције:** Лекар кликом на дугме “Kreiraj novog” позива системску операцију KreirajPacijent(Pacijent).

2. Систем креира пацијента. (СО)
3. Систем приказује лекару пацијента и поруку: “Систем је креирао пацијента”. (ИА)



4. **Лекар** уноси податке о **пацијенту**. (АПУСО)

Sistem za praćenje rada doma zdravlja | Pacijent

Ime:  Zapamti

Prezime:  Obriši

Email:

Mesto:  Odustani

Unos novog pacijenta

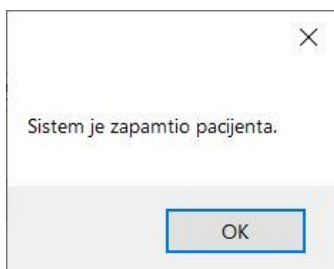
5. **Лекар** контролише да ли је коректно унео податке о **пацијенту**. (АНСО)

6. **Лекар** позива **систем** да запамти податке о **пацијенту**. (АПСО)

Опис акције: **Лекар** кликом на дугме “Zapamti” позива системску операцију **PromeniPacijent(Pacijent)**.

7. **Систем** памти податке о **пацијенту**. (СО)

8. **Систем** приказује **лекару** **пацијента** и поруку: “Систем је запамтио **пацијента**.” (ИА)



Sistem za praćenje rada doma zdravlja | Pacijent

Kriterijumi pretrage

Ime:

Prezime:

Mesto:

Akcije

Pretraži

Kreiraj novog

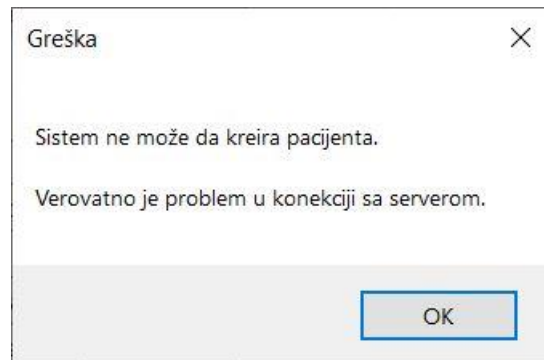
Detalji

	Id	Ime	Prezime	Email	Mesto
	16	Milica	Tomić	milica.tomic@gmail.com	Niš
	17	Goran	Radovanović	goran827@gmail.com	Jagodina
	18	Teodora	Živanović	teodorakg@mail.com	Kragujevac
	19	Vladimir	Blagojević	vblagojevic@outlook.com	Subotica
	20	Sofija	Janković	sofija23112@gmail.com	Beograd
	21	Nemanja	Maksimović	nemanjam@gmail.com	Novi Sad
	22	Andrej	Vučković	andrejvuckovic9211@gmail.com	Niš
	23	Martin	Jovanović	mjovanovic554@gmail.com	Beograd

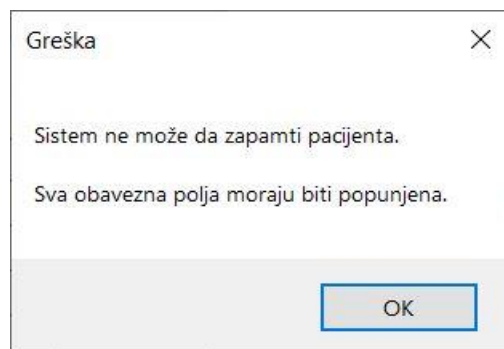


### Алтернативна сценарија:

3.1 Уколико **систем** не може да креира **пацијента** он **приказује** **лекару** поруку: “**Систем** не може да креира **пацијента**”. Прекида се извршење сценарија. (ИА)



8.1 Уколико **систем** не може да запамти податке о **пацијенту** он **приказује** **лекару** поруку: “**Систем** не може да запамти **пацијента**”. (ИА)



## СК6- Претражи пацијент

### Назив СК

Претражи пацијент

### Актори СК

Лекар

### Учесници СК

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

**Предуслови:** Кориснички интерфејс (клијентски програм) и систем (серверски програм) су покренути. Лекар је пријављен под својом шифром. Кориснички интерфејс приказује форму за рад са пацијентом. На наведеној екранској форми су дефинисани критеријуми, који се односе на: а) Пацијент б) Место, који ће да врате листу пацијената.

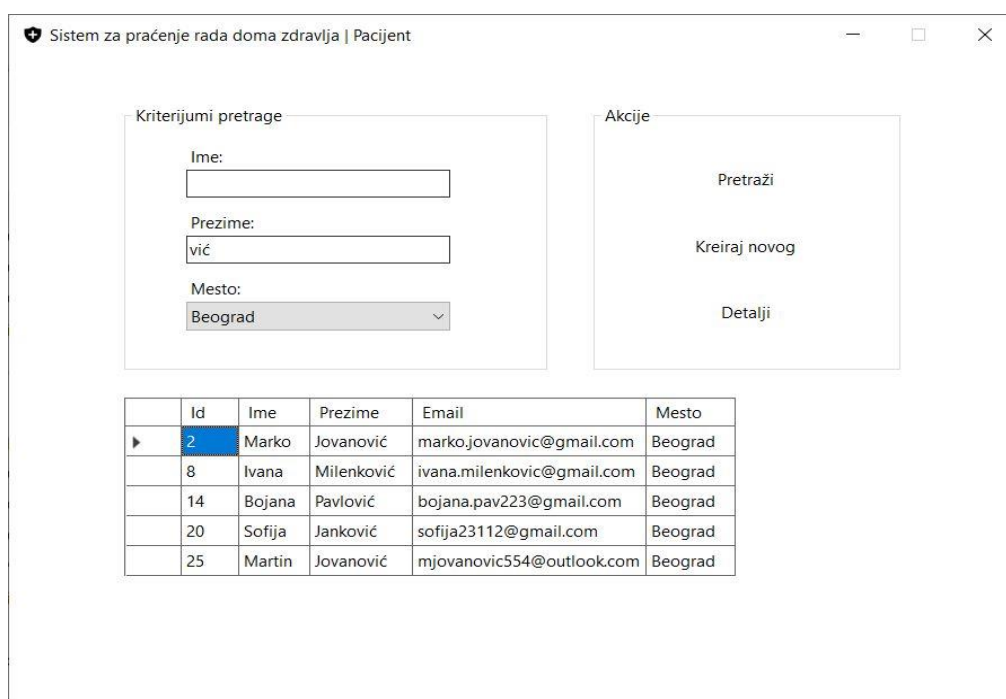
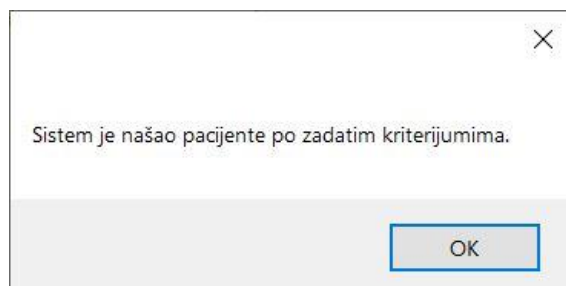
### Основни сценарио СК:

1. Лекар бира критеријуме на основу којих претражује пацијенте. (АПУСО)

2. **Лекар** **позива систем** да нађе **пацијенте** по задатим критеријумима. (АПСО)

**Опис акције:** **Лекар** кликом на дугме “Pretraži” позива системске операције за враћање листе **пацијената**.

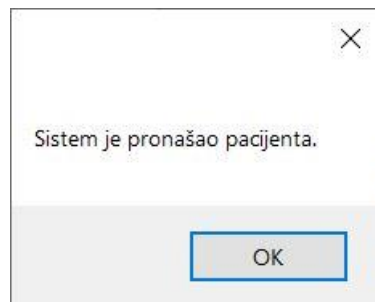
3. **Систем** **тражи пацијенте** по задатим критеријумима. (СО)  
4. **Систем приказује лекару пацијенте** и поруку: “**Систем** је нашао **пацијенте** по задатим критеријумима”. (ИА)



5. **Лекар бира пацијента**. (АПУСО)  
6. **Лекар позива систем** да нађе **пацијента**. (АПСО)

**Опис акције:** **Лекар** кликом на дугме “Detalji” позива системску операцију **PretraziPacijent (Pacijent)**.

7. Систем **тражи** пацијента. (CO)
8. Систем **приказује** **лекару** пацијента и поруку: “Систем је нашао пацијента”. (ИА)



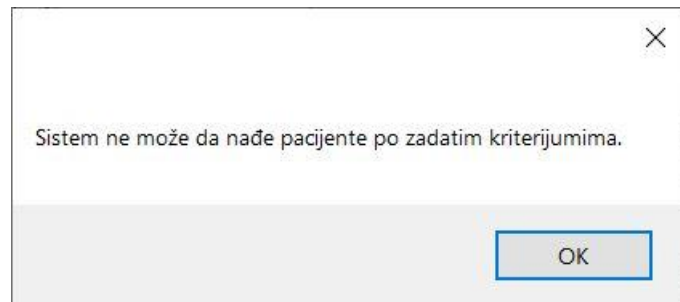
Sistem za praćenje rada doma zdravlja | Pacijent

Ime:	<input type="text" value="Martin"/>	Zapamti
Prezime:	<input type="text" value="Jovanović"/>	Obriši
Email:	<input type="text" value="mjovanovic554@outlook.com"/>	
Mesto:	<input type="text" value="Beograd"/>	Odustani

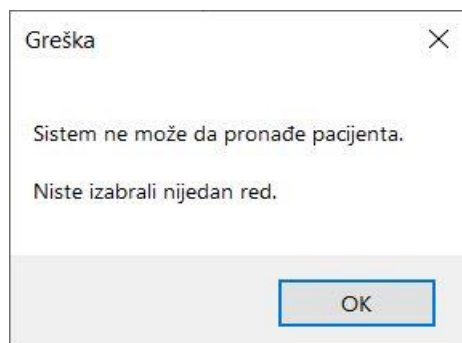
Pacijent ID: 23

### Алтернативна сценарија:

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



8.1 Уколико **систем** не може да нађе **пацијента** он **приказује лекару** поруку: “**Систем** не може да нађе **пацијента** ”.(ИА)



## СК7- Промени пацијент

### Назив СК

Промени пацијент

### Актори СК

Лекар

### Учесници СК

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

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

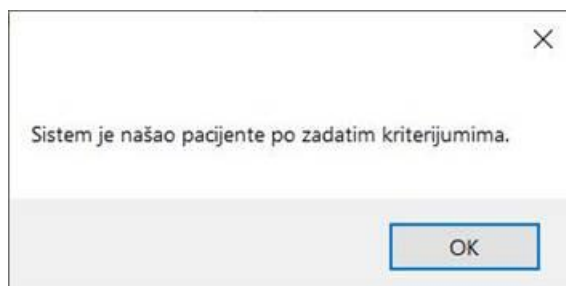
### Основни сценарио СК:

1. Лекар бира критеријуме на основу којих претражује пацијенте. (АПУСО)

2. **Лекар** позива **систем** да нађе **пацијенте** по задатим критеријумима. (АПСО)

**Опис акције:** **Лекар** кликом на дугме “Pretraži” позива системске операције за враћање листе **пацијента**.

3. **Систем** **тражи** **пацијенте** по задатим критеријумима. (СО)  
4. **Систем** **приказује** **лекару** **пацијенте** и поруку: “**Систем** је нашао **пацијенте** по задатим критеријумима”. (ИА)



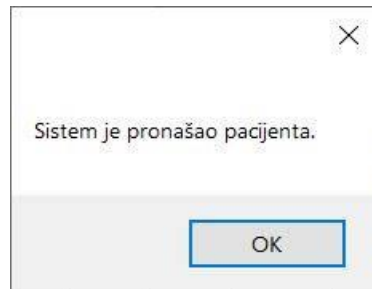
The screenshot shows a web application window titled "Sistem za praćenje rada doma zdravlja | Pacijent". It contains two main sections: "Kriterijumi pretrage" (Search criteria) and "Akcije" (Actions). The "Kriterijumi pretrage" section has input fields for "Ime:" (Name), "Prezime:" (Surname), and "Mesto:" (Location), with a dropdown menu for "Mesto:" showing "Beograd". The "Akcije" section has three buttons: "Pretraži" (Search), "Kreiraj novog" (Create new), and "Detalji" (Details). Below these sections is a table of patients.

	Id	Ime	Prezime	Email	Mesto
▶	2	Marko	Jovanović	marko.jovanovic@gmail.com	Beograd
	8	Ivana	Milenković	ivana.milenkovic@gmail.com	Beograd
	14	Bojana	Pavlović	bojana.pav223@gmail.com	Beograd
	20	Sofija	Janković	sofija23112@gmail.com	Beograd
	25	Martin	Jovanović	mjovanovic554@outlook.com	Beograd

5. **Лекар** **бира** **пацијента**. (АПУСО)  
6. **Лекар** позива **систем** да нађе **пацијента**. (АПСО)

**Опис акције:** **Лекар** кликом на дугме “Detalji” позива системску операцију **PretraziPacijent (Pacijent)**.

7. Систем тражи пацијента. (СО)
8. Систем приказује лекару пацијента и поруку: “Систем је нашао пацијента”. (ИА)



Sistem za praćenje rada doma zdravlja | Pacijent

Ime:  Zapamti

Prezime:  Obriši

Email:

Mesto:  Odustani

Pacijent ID: 23

9. Лекар уноси (мења) податке о пацијенту. (АПУСО)

Sistem za praćenje rada doma zdravlja | Pacijent

Ime:  Zapamti

Prezime:  Obriši

Email:

Mesto:  Odustani

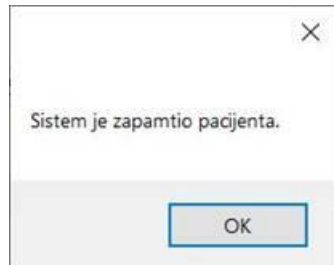
Pacijent ID: 23



10. **Лекар контролише** да ли је коректно унео податке о **пацијенту**. (АНСО)  
11. **Лекар позива систем** да запамти податке о **пацијенту**. (АПСО)

Опис акције: **Лекар** кликом на дугме “Zapamti” позива системску операцију PromeniPacijent(Pacijent).

12. Систем памти податке о пацијенту. (CO)
13. Систем приказује лекару пацијента и поруку: “Систем је запамтио пацијента.” (IA)



Sistem za praćenje rada doma zdravlja | Pacijent

Kriterijumi pretrage

Ime:

Prezime:

Mesto:

-- Bez izbora --

Akcije

Pretraži

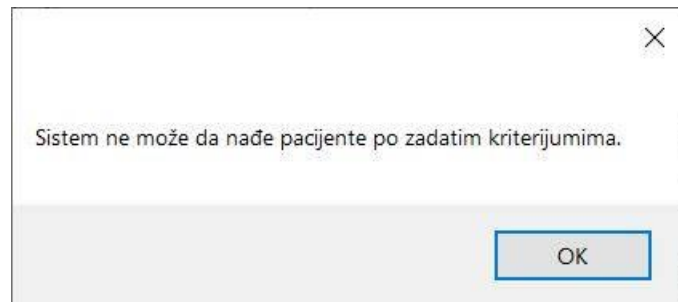
Kreiraj novog

Detalji

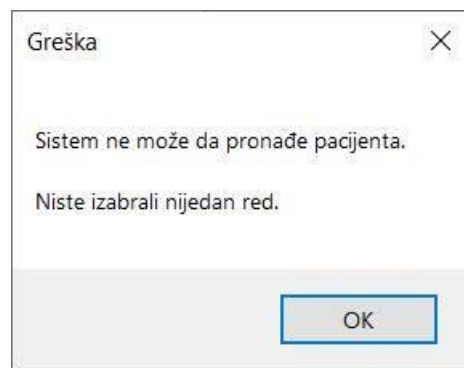
	Id	Ime	Prezime	Email	Mesto
	16	Milica	Tomić	milica.tomic@gmail.com	Niš
	17	Goran	Radovanović	goran827@gmail.com	Jagodina
	18	Teodora	Živanović	teodorakg@mail.com	Kragujevac
	19	Vladimir	Blagojević	vblagojevic@outlook.com	Subotica
	20	Sofija	Janković	sofija23112@gmail.com	Beograd
	21	Nemanja	Maksimović	nemanjam@gmail.com	Novi Sad
	22	Andrej	Vučković	andrejvuckovic9211@gmail.com	Niš
	23	Martin	Jovanović	martin@gmail.com	Novi Sad

### Алтернативна сценарија:

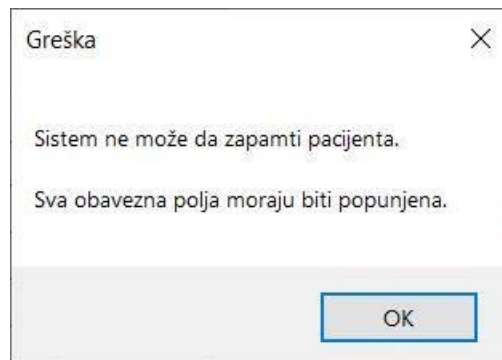
4.1 Уколико **систем** не може да нађе **пацијенте** он **приказује лекару** поруку: “**Систем** не може да нађе **пацијенте** по задатом критеријуму”. Прекида се извршење сценарија. (ИА)



8.1 Уколико **систем** не може да нађе **пацијента** он **приказује лекару** поруку: “**Систем** не може да нађе **пацијента** ”. Прекида се извршење сценарија. (ИА)



13.1 Уколико **систем** не може да запамти податке о **пацијенту** он **приказује лекару** поруку: “**Систем** не може да запамти **пацијента**”. (ИА)



## СК8- Обриши пацијент

### Назив СК

Обриши пацијент

### Актори СК

Лекар

### Учесници СК

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

**Предуслови:** Кориснички интерфејс (клијентски програм) и систем (серверски програм) су покренути. Лекар је пријављен под својом шифром. Кориснички интерфејс приказује форму за рад са пацијентом. На наведеној екранској форми су дефинисани критеријуми, који се односе на: а) Пацијент б) Место, који ће да врате листу пацијената.

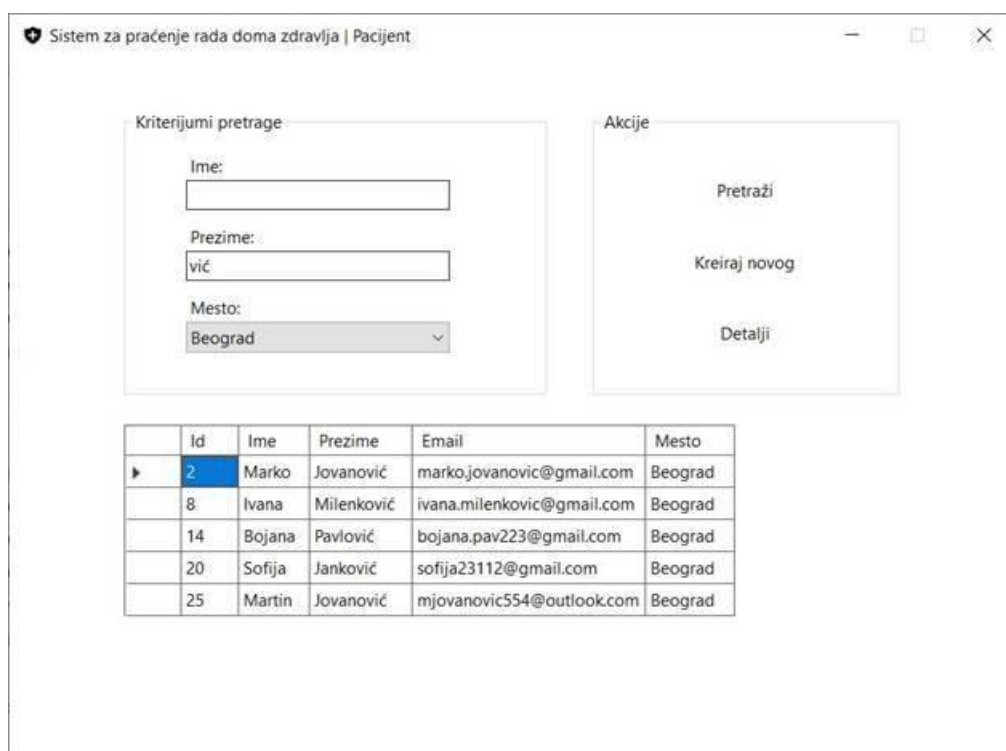
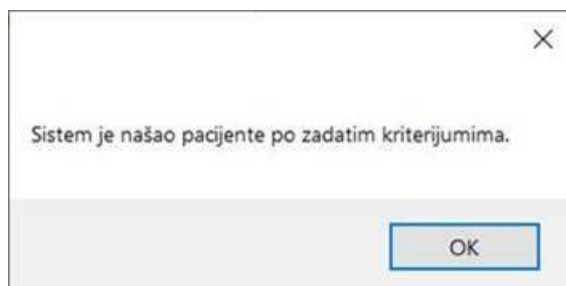
### Основни сценарио СК:

1. Лекар бира критеријуме на основу којих претражује пацијенте. (АПУСО)

2. **Лекар** **позива** **систем** да нађе **пацијенте** по задатим критеријумима. (АПСО)

**Опис акције:** **Лекар** кликом на дугме “Pretraži” позива системске операције за враћање листе **пацијента**.

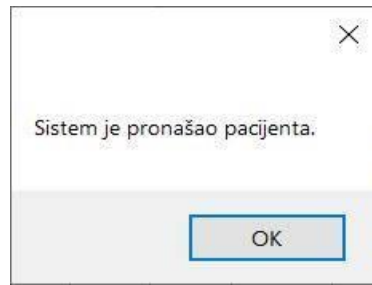
3. **Систем** **тражи** **пацијенте** по задатим критеријумима. (СО)  
4. **Систем** **приказује** **лекару** **пацијенте** и поруку: “**Систем** је нашао **пацијенте** по задатим критеријумима”. (ИА)



5. **Лекар** **бира** **пацијента**. (АПУСО)  
6. **Лекар** **позива** **систем** да нађе **пацијента**. (АПСО)

**Опис акције:** **Лекар** кликом на дугме “Detalji” позива системску операцију **PretraziPacijent (Pacijent)**.

7. Систем тражи пацијента. (CO)
8. Систем приказује лекару пацијента и поруку: “Систем је нашао пацијента”. (ИА)



The form is titled "Sistem za praćenje rada doma zdravlja | Pacijent". It contains the following fields and controls:

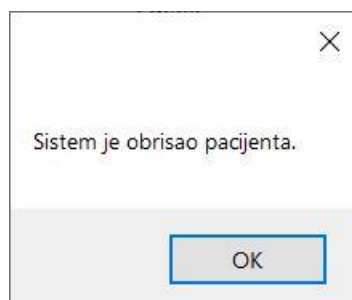
- Ime:** Text input field containing "Martin". To its right is a button labeled "Zapamti".
- Prezime:** Text input field containing "Jovanović". To its right is a button labeled "Obriši".
- Email:** Text input field containing "mjovanovic554@outlook.com".
- Mesto:** Dropdown menu showing "Beograd". To its right is a button labeled "Odustani".

At the bottom left of the form, it says "Pacijent ID: 23".

9. Лекар позива систем да обрише пацијента. (АПСО)

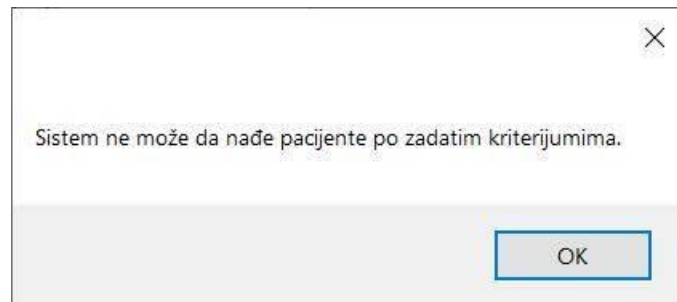
Опис акције: Лекар кликом на дугме “Obriši” позива системску операцију **ObrisiPacijent (Pacijent)**.

10. Систем брише пацијента. (CO)
11. Систем приказује лекару поруку: “Систем је обрисао пацијента.” (ИА)

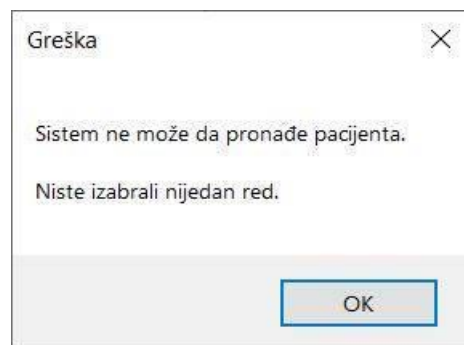


### Алтернативна сценарија:

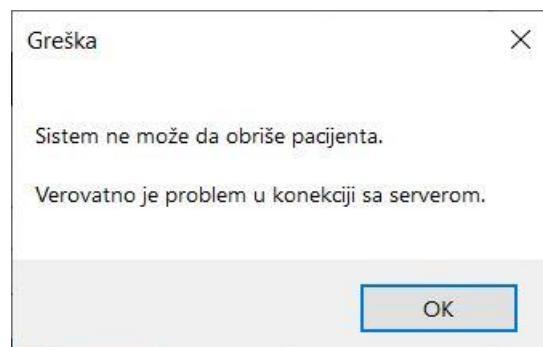
4.1 Уколико **систем** не може да нађе **пацијенте** он **приказује лекару** поруку: “**Систем** не може да нађе **пацијенте** по задатим критеријумима”. Прекида се извршење сценарија. (ИА)



8.1 Уколико **систем** не може да нађе **пацијента** он **приказује лекару** поруку: “**Систем** не може да нађе **пацијента** ”. Прекида се извршење сценарија. (ИА)



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



## СК9- Пријави лекар

### Назив СК

Пријави лекар

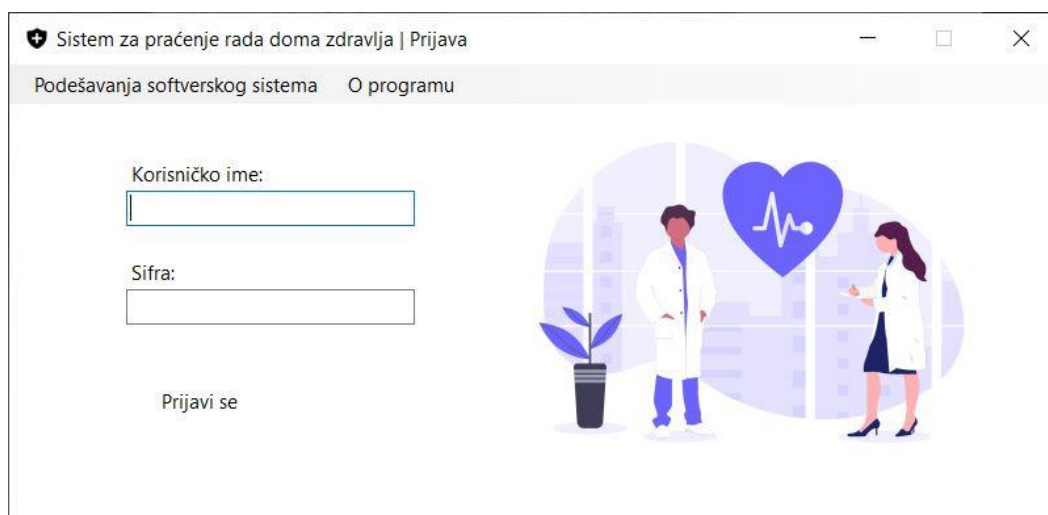
### Актори СК

Лекар

### Учесници СК

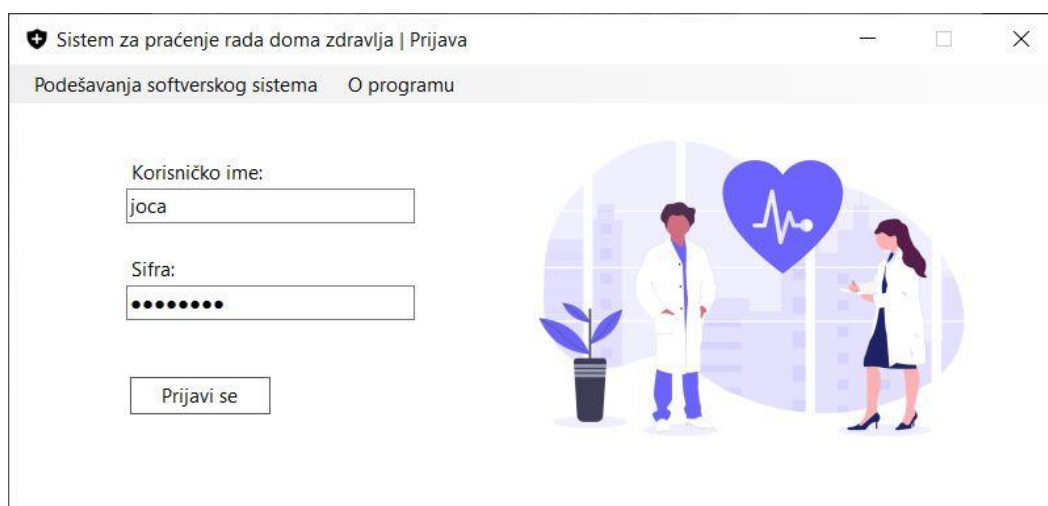
Лекар, кориснички интерфејс (клијентски програм) и систем (серверски програм)

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



### Основни сценарио СК:

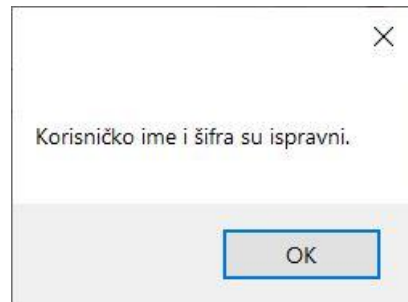
1. **Лекар** уноси корисничко име и шифру. (АПУСО)



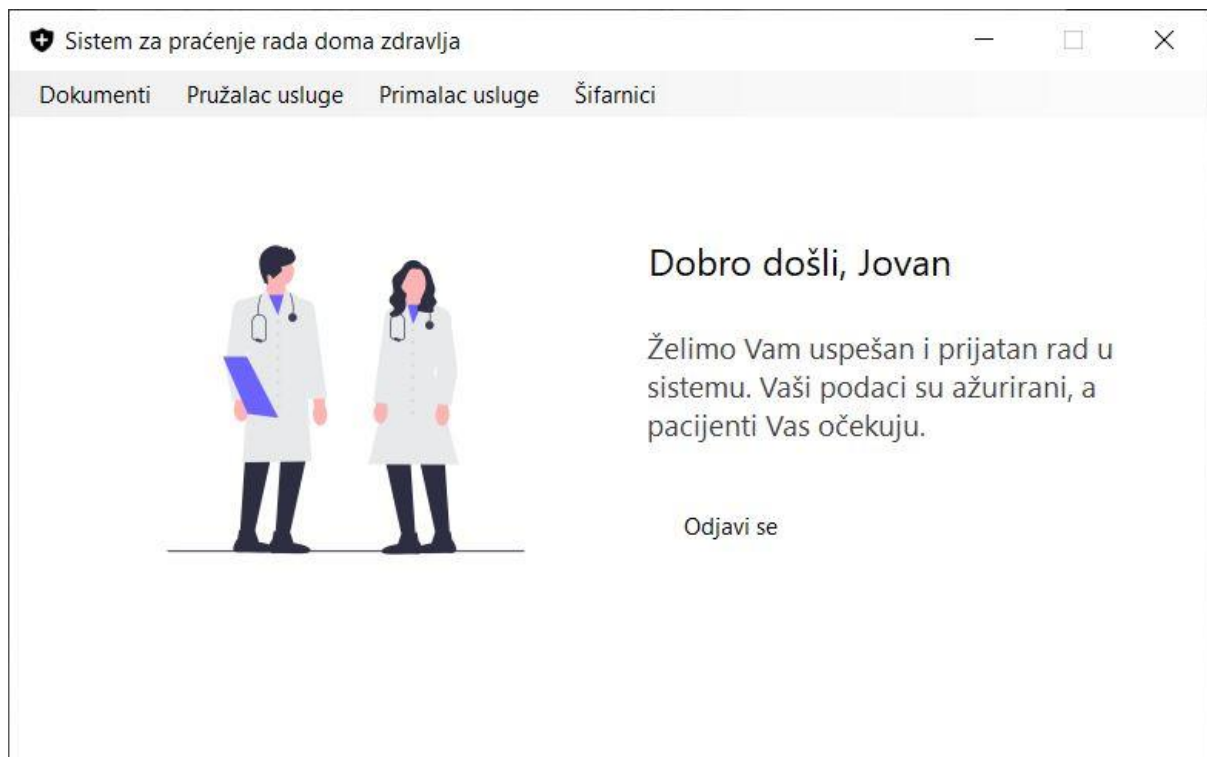
2. **Лекар** контролише да ли је коректно унео корисничко име и шифру. (АНСО)
3. **Лекар** позива систем да провери корисничко име и шифру. (АПСО)

**Опис акције:** Лекар кликом на дугме "Prijava se" позива системску операцију **PrijaviLekar(korisnickolme, sifra)**

4. Систем **проверава** корисничко име и шифру. (CO)
5. Систем **приказује** **лекару** поруку: "Корисничко име и шифра су исправни." (ИА)



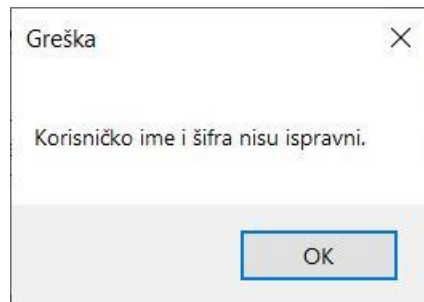
6. Кориснички интерфејс **позива** главни форму и мени. (КИПГФМ)



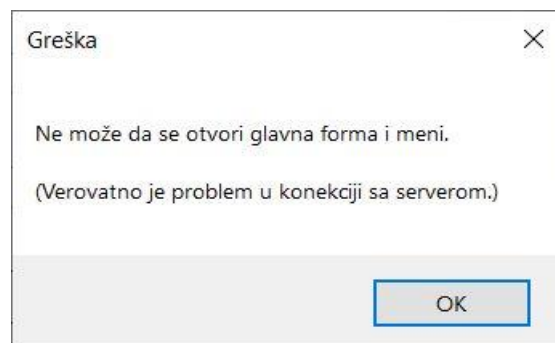


### Алтернативна сценарија:

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



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



## СК22- Убацн сертнфкат

### Назнв СК

Убацн сертнфкат

### Акторн СК

Лекар

### Учесннцн СК

Лекар, корнсннчкн ннтерфејс (клнјентскн програм) н снстем (серверскн програм)

**Предусловн:** Корнсннчкн ннтерфејс (клнјентскн програм) н снстем (серверскн програм) су покренутн. Лекар је прнјављен под својом шнфром. Снстем прнказује форму за рад са сертнфкатом.

Sistem za praćenje rada doma zdravlja | Sertifikat

Opis:

Zapamti

Obriši

Odustani

Unos novog sertifikata

### Основнн сценарно СК:

1. Лекар уносн податке о сертнфкату. (АПУСО)

Sistem za praćenje rada doma zdravlja | Sertifikat

Opis:

Sertifikat odbora za kardiologiju (EBC)

Zapamti

Obriši

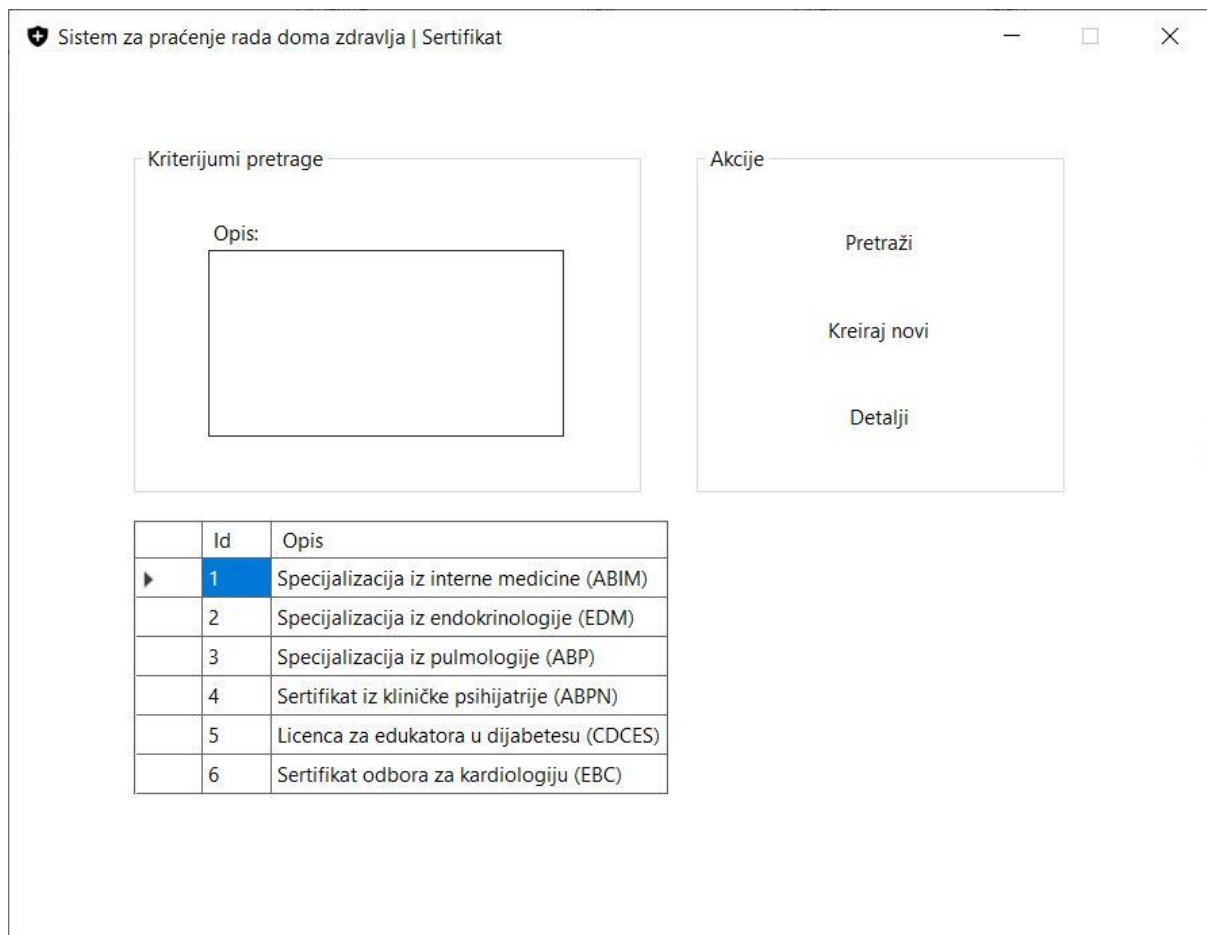
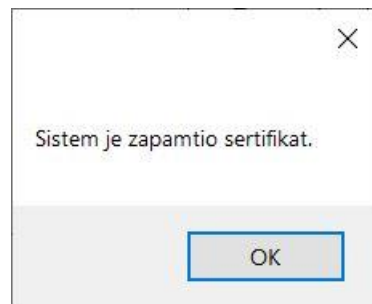
Odustani

Unos novog sertifikata

2. Лекар контролнше да лн је коректно унео податке о сертнфкату. (АНСО)
3. Лекар познва снстем да запамтн податке о сертнфкату. (АПСО)

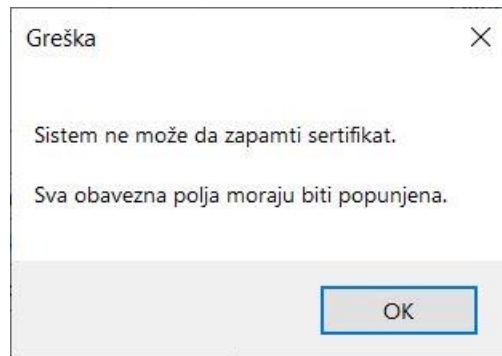
**Опнс акцнје:** Лекар кликом на дугме “Zapamti” познва снстемску операцију **UbaciSertifikat (Sertifikat)**.

4. Систем памти податке о сертификату. (CO)
5. Систем приказује лекару сертификат и поруку: “Систем је запамтио сертификат.” (ИА)



### Алтернативна сценарија:

5.1 Уколико **систем** не може да запамти податке о **сертификату** он **приказује** **лекару** поруку: “**Систем** не може да запамти **сертификат**”. (ИА)



### Пројектовање контролера корисничког интерфејса

Контролер корисничког интерфејса има улогу да:

- прихвати графичке објекте које шаље екранска форма,
- конвертује графички објекат у доменски објекат који представља улазни аргумент системске операције која ће бити позвана,
- пошаље захтев за извршење системске операције до апликационог сервера,
- прихвати одговор софтверског система настао као резултат извршења системске операције, и
- конвертује доменски објекат одговора у податке графичких елемената.

## **4.2 Пројектовање апликационе логике**

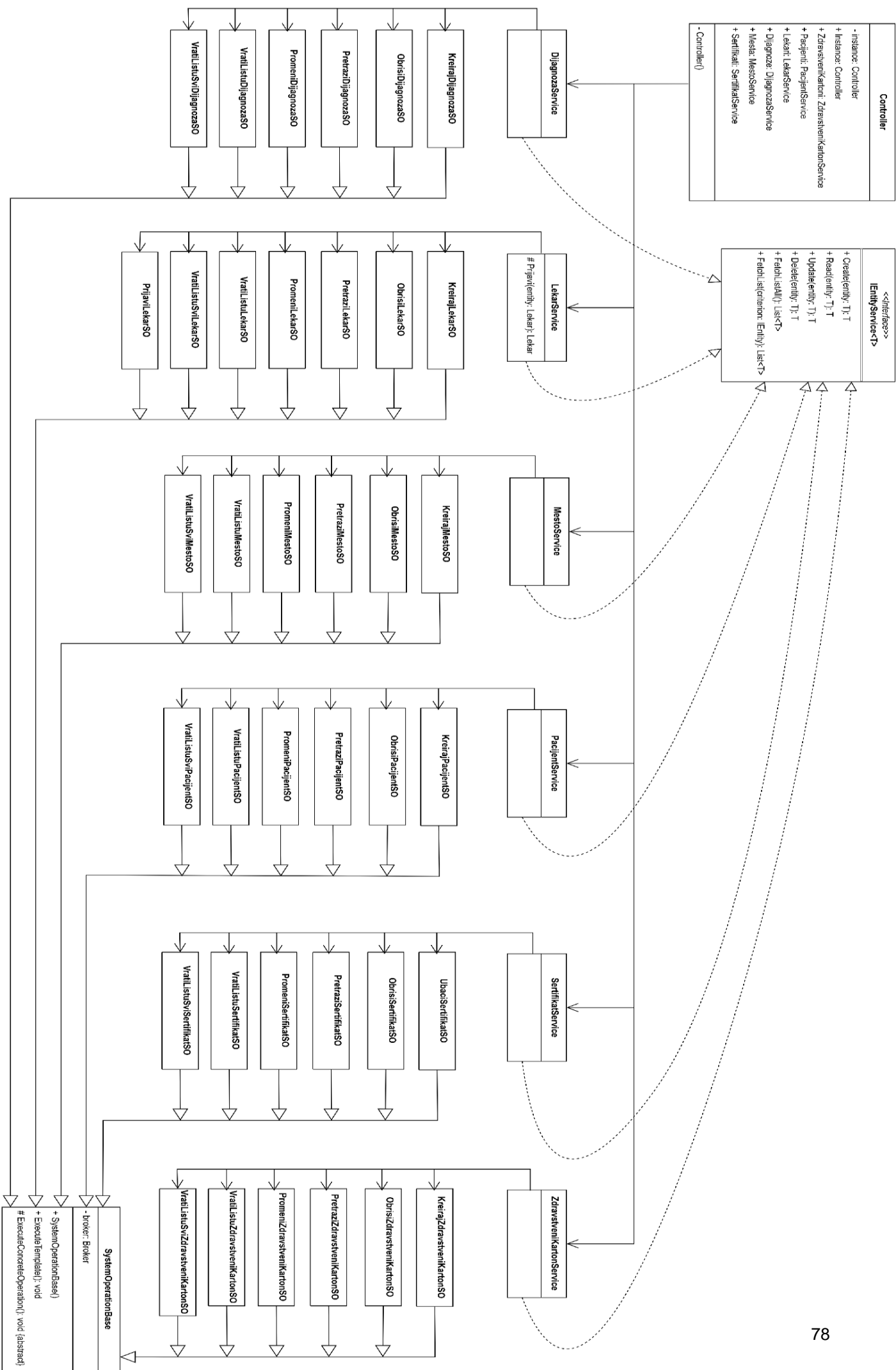
Контролер апликационе логике подиже серверски сокет који ће да ослушкује мрежу. Када клијентски сокет успостави конекцију са серверским сокетом, сервер инстанцира нит која омогућава двосмерну комуникацију са клијентом.

Комуникација између клијента и сервера се постиже разменом објеката класе Request и Response, који се преносе преко сокета. Клијент шаље захтев за извршење системских операција серверској нити придруженој том клијенту.

Ова нит прима захтев и прослеђује га контролеру апликационе логике. Након што се системска операција изврши, резултат се враћа кроз контролер апликационе логике и доставља назад клијентској нити. Ова нит затим преноси резултат до клијента.

### **Контролер апликационе логике**

Контролер апликационе логике прима захтев за извршење системске операције који долази од клијентске нити и прослеђује га као одговарајућој сервисној класи. Свака сервисна класа представља посредника између контролера и системских операција и задужена је за позивање одговарајуће system operation класе, у зависности од конкретне функционалности. Сервис инстанцира одговарајућу класу системске операције, прослеђује јој потребне параметре и покреће њено извршавање. Након што се системска операција заврши, резултат се враћа сервису, затим контролеру, који га прослеђује даље позиваоцу — односно клијентској нити. Оваква архитектура омогућава јасну поделу одговорности, бољу организованост кода и лакше тестирање појединачних компоненти система. Системске операције су имплементиране као посебне класе које наслеђују заједнички шаблон и инкапсулирају пословну логику везану за рад са појединачним ентитетима.



## Пословна логика

Пословна логика је описана структуром (доменским класама) и понашањем (системским операцијама).

### *Пројектовање понашања софтверског система - системске операције*

За сваку системску операцију потребно је направити концептуална решења која су директно повезана са логиком проблема.

За сваки уговор пројектује се концептуално решење. Овде ће бити представљена концептуална решења за раније дефинисане типске уговоре.

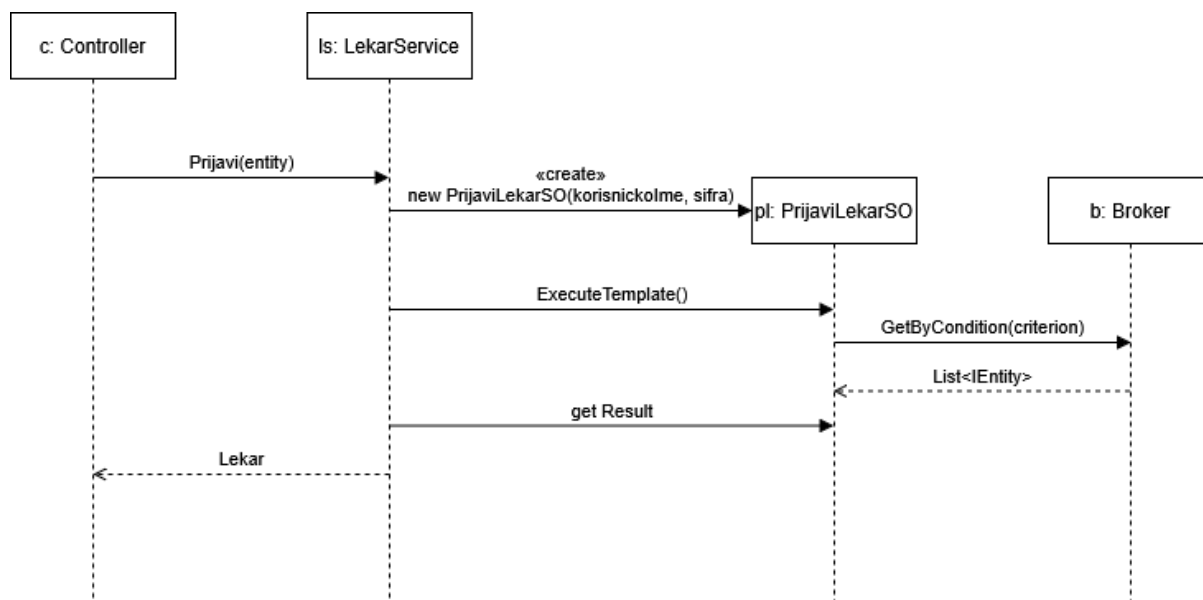
#### **1. Уговор UG1: *PrijaviLekar(korisnickolme, Sifra)***

**Операција:** *PrijaviLekar (Korisnickolme, Sifra):signal;*

**Веза са СК:** CK9

**Предуслови:**

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



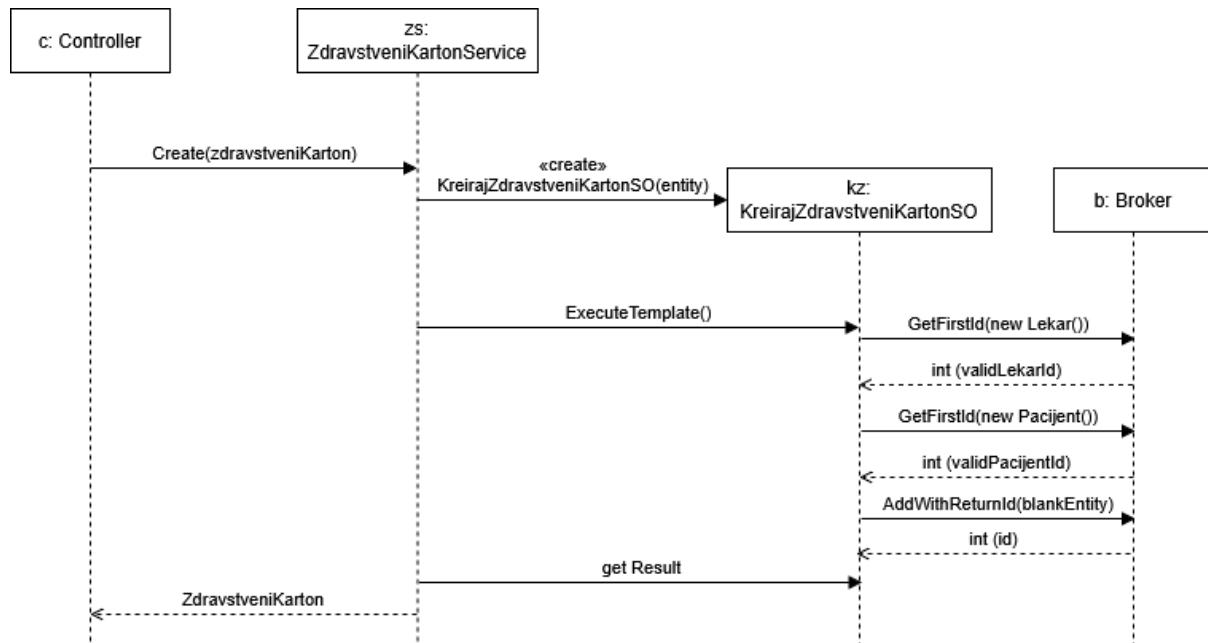
## 2. Уговор UG2: KreirajZdravstveniKarton(ZdravstveniKarton)

Операција: KreirajZdravstveniKarton(ZdravstveniKarton):signal;

Веза са СК: СК1

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

Постуслови: Направљен је нови објекат класе ЗдравствениКартон.



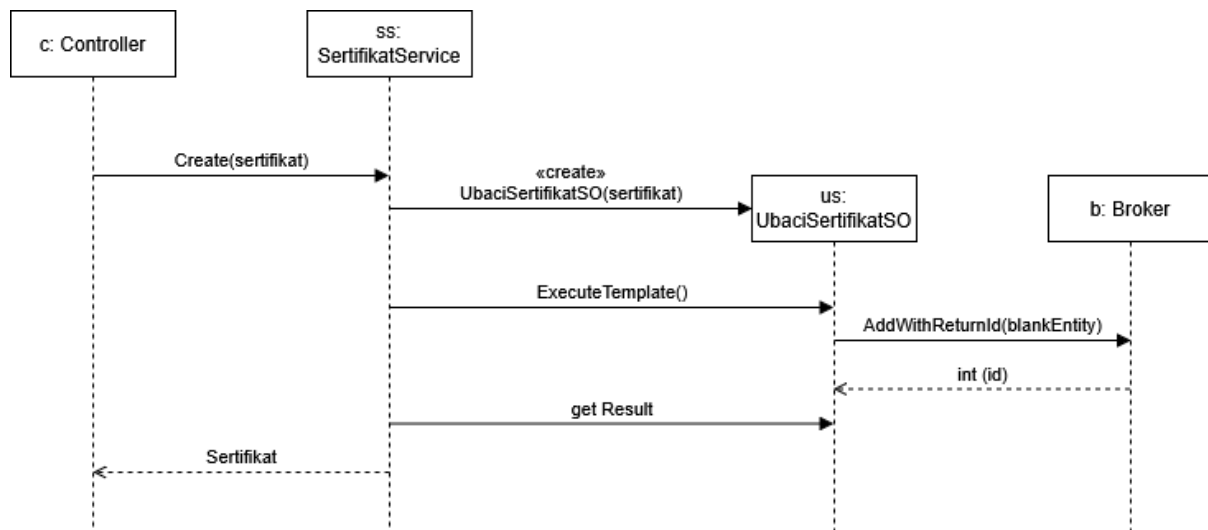
## 3. Уговор UG3: UbaciSertifikat(Sertifikat)

Операција: UbaciSertifikat(Sertifikat):signal;

Веза са СК: СК22

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

Постуслови: Направљен је нови објекат класе Сertификат.





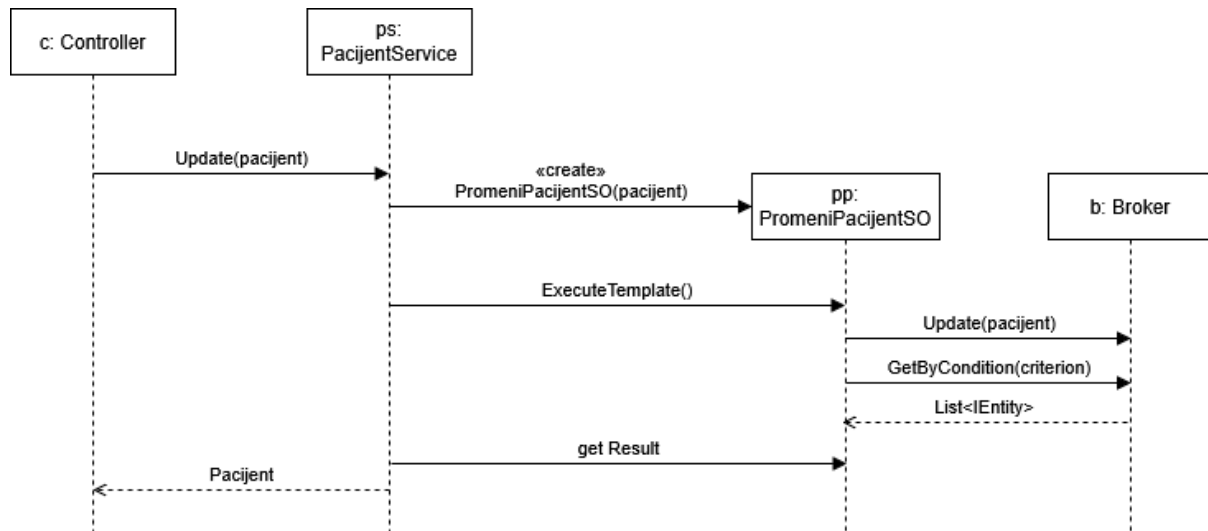
#### 4. Уговор UG4: PromeniPacijent(Pacijent)

Операција: PromeniPacijent(Pacijent):signal;

Веза са СК: СК5, СК7

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

Постуслови: Објекат класе Пацијент је промењен.



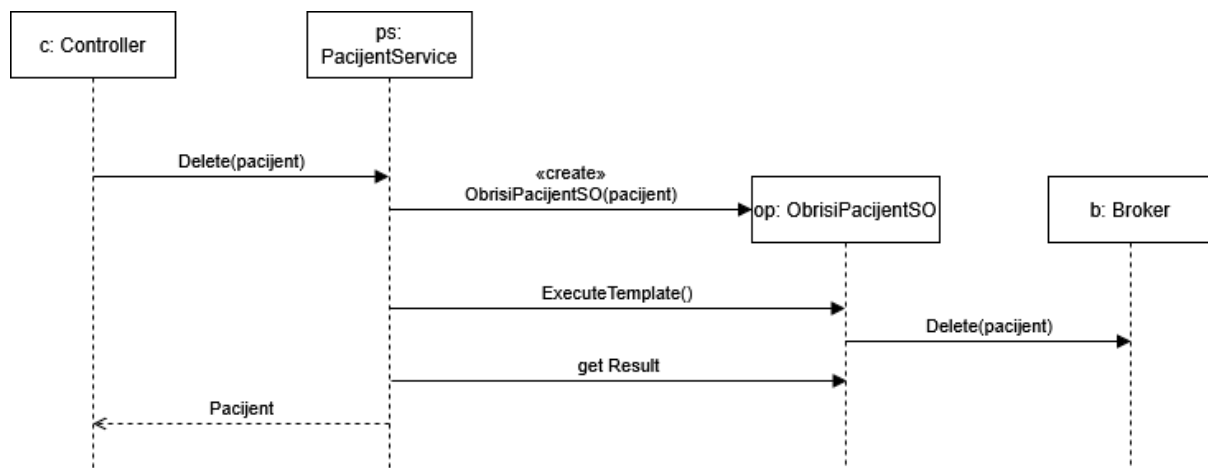
#### 5. Уговор UG5: ObrisiPacijent(Pacijent)

Операција: ObrisiPacijent(Pacijent):signal;

Веза са СК: СК8

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

Постуслови: Објекат класе Пацијент је обрисан.



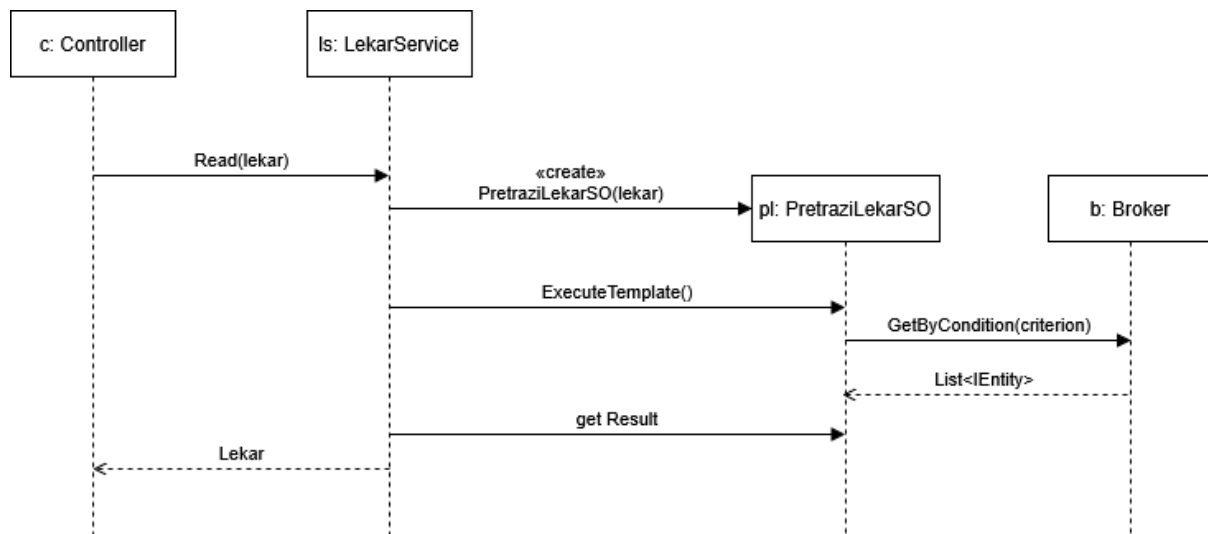
## 6. Уговор UG6: PretraziLekar

Операција: PretraziLekar(X):signal;

Веза са СК: СК11, СК12

Предуслови:

Постуслови: Пронађен је тражени објект класе Лекар.



## 7. Уговор UG7: vratiListuZdravstveniKarton

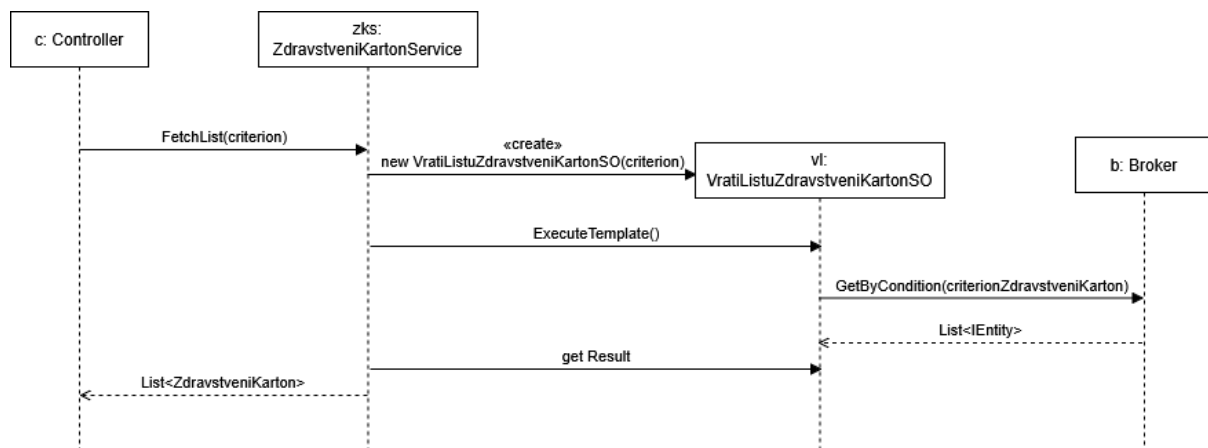
Операција: vratiListuZdravstveniKarton

(KriterijumZdravstveniKarton,Lista<ZdravstveniKarton> ):signal;

Веза са СК: СК2, СК4

Предуслови:

Постуслови: Пронађена је листа тражених објеката класе ЗдравствениКартон.



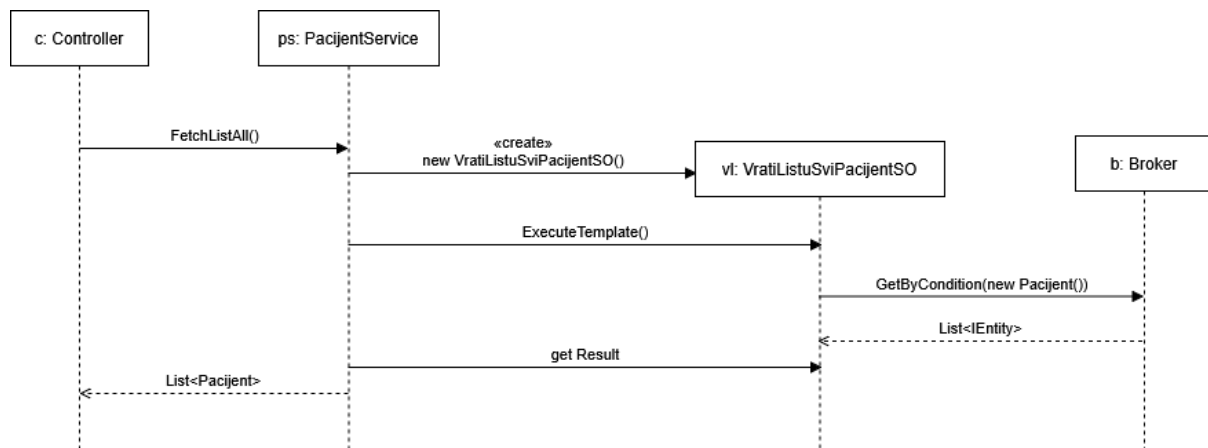
## 8. Уговор UG8: vratiListuSviPacijent

Операција: `vratiListuSviPacijent (Lista<Pacijent> ):signal;`

Веза са СК: СК1, СК3

Предуслови:

Постуслови: Пронађена је листа свих објеката класе Пацијент.



Класе које обављају системске операције наслеђују апстрактну класу *SystemOperationBase* са циљем централизованог руковања повезивањем са базом података. Класа *SystemOperationBase* дефинише темплејт метод *ExecuteTemplate()* који омогућава стандардизован ток извршавања било које системске операције. Тај ток обухвата: отварање конекције са базом, покретање трансакције, извршавање конкретне логике операције (*ExecuteConcreteOperation()*), као и потврду трансакције (*Commit()*) или њено поништавање (*Rollback()*) у случају неуспеха. Свака појединачна системска операција (нпр. *KreirajPacijentSO*, *ObrisiLekarSO*, *VratiListuZdravstveniKartonSO*) имплементира своју верзију метода за проверу предуслова и конкретну логику операције, чиме се обезбеђује флексибилан и безбедан рад са подацима.

## Пројектовање структуре софтверског система - доменске класе

На основу концептуалних класа праве се софтверске класе структуре:

```
public class Dijagnoza : ICrudEntity
{
    public int Id { get; set; }
    public string Naziv { get; set; }
    public string Opis { get; set; }
    public Double BazniSkor { get; set; }

    public string TableName => "dijagnoza";
    public string ClassNameAccusativeSingular => "dijagnozu";
    public string ClassNameAccusativePlural => "dijagnoze";
    public string Columns => "naziv, opis, bazniSkor";
    public string ValuesClause => "@Naziv, @Opis, @BazniSkor";
    public string SetClause => "naziv = @Naziv, opis = @Opis, bazniSkor = @BazniSkor";
    public string PrimaryKey => "idDijagnoza";
    public string PrimaryKeyCondition => "idDijagnoza = @Id";
    public string DisplayValue => Naziv;

    public (string whereClause, List<SqlParameter> parameters) GetWhereClauseWithParameters()
    {
        var conditions = new List<string>();
        var parameters = new List<SqlParameter>();

        if (Id > 0)
        {
            conditions.Add("idDijagnoza = @Id");
            parameters.Add(new SqlParameter("@Id", Id));
        }

        if (!string.IsNullOrEmpty(Naziv))
        {
            conditions.Add("naziv LIKE @Naziv");
            parameters.Add(new SqlParameter("@Naziv", $"%{Naziv}%"));
        }

        if (!string.IsNullOrEmpty(Opis))
        {
            conditions.Add("opis LIKE @Opis");
            parameters.Add(new SqlParameter("@Opis", $"%{Opis}%"));
        }

        return (conditions.Count > 0 ? string.Join(" AND ", conditions) : "1=1", parameters);
    }

    public List<IEntity> GetReaderList(SqlDataReader reader)
    {
        var lista = new List<IEntity>();
        while (reader.Read())
        {
            lista.Add(new Dijagnoza
            {
                Id = (int)reader["idDijagnoza"],
                Naziv = (string)reader["naziv"],
                Opis = (string)reader["opis"],
                BazniSkor = (double)(reader["bazniSkor"])
            });
        }
        return lista;
    }

    public List<SqlParameter> GetSqlParameter()
    {
        return new List<SqlParameter>
        {
            new SqlParameter("@Id", Id),
            new SqlParameter("@Naziv", Naziv),
            new SqlParameter("@Opis", Opis),
            new SqlParameter("@BazniSkor", BazniSkor)
        };
    }

    public List<SqlParameter> GetPrimaryKeyParameters()
    {
        return new List<SqlParameter>
        {
            new SqlParameter("@Id", Id)
        };
    }

    public override string ToString() => DisplayValue;
}
```

```

public class Lekar : ICrudEntity
{
    public int Id { get; set; }
    public string Ime { get; set; }
    public string Prezime { get; set; }
    public string Email { get; set; }
    public string KorisnickoIme { get; set; }
    public string Sifra { get; set; }
    public List<Sertifikat> Sertifikati { get; set; } = new List<Sertifikat>();

    public string TableName => "lekar";
    public string ClassNameAccusativeSingular => "lekara";
    public string ClassNameAccusativePlural => "lekare";

    public string Columns => "ime, prezime, email, sifra, korisnickoIme";
    public string ValuesClause => "@Ime, @Prezime, @Email, @Sifra, @KorisnickoIme";
    public string SetClause => "ime = @Ime, prezime = @Prezime, email = @Email";
    public string PrimaryKey => "idLekar";
    public string PrimaryKeyCondition => "idLekar = @Id";
    public string DisplayValue => $"{Ime} {Prezime}";
    public string JoinTableName =>
        "lekar l " +
        "LEFT JOIN les les ON l.idLekar = les.idLekar " +
        "LEFT JOIN sertifikat s ON les.idSertifikat = s.idSertifikat";
    public string SelectClause =>
        "l.idLekar, l.ime, l.prezime, l.email, l.sifra, l.korisnickoIme, " +
        "s.idSertifikat AS SertifikatId, s.opis AS SertifikatOpis";

    public (string whereClause, List<SqlParameter> parameters) GetWhereClauseWithParameters()
    {
        var conditions = new List<string>();
        var parameters = new List<SqlParameter>();

        if (Id > 0)
        {
            conditions.Add("l.idLekar = @Id");
            parameters.Add(new SqlParameter("@Id", Id));
        }

        if (!string.IsNullOrEmpty(Ime))
        {
            conditions.Add("l.ime LIKE @Ime");
            parameters.Add(new SqlParameter("@Ime", $"{Ime}%"));
        }

        if (!string.IsNullOrEmpty(Prezime))
        {
            conditions.Add("l.prezime LIKE @Prezime");
            parameters.Add(new SqlParameter("@Prezime", $"{Prezime}%"));
        }

        if (!string.IsNullOrEmpty(Email))
        {
            conditions.Add("l.email = @Email");
            parameters.Add(new SqlParameter("@Email", Email));
        }

        if (!string.IsNullOrEmpty(KorisnickoIme))
        {
            conditions.Add("l.korisnickoIme = @KorisnickoIme");
            parameters.Add(new SqlParameter("@KorisnickoIme", KorisnickoIme));
        }

        if (Sertifikati?.Count > 0 && Sertifikati[0]?.Id > 0)
        {
            conditions.Add("l.idLekar IN (SELECT idLekar FROM les WHERE idSertifikat = @IdSertifikat)");
            parameters.Add(new SqlParameter("@IdSertifikat", Sertifikati[0].Id));
        }

        return (conditions.Count > 0 ? string.Join(" AND ", conditions) : "1=1", parameters);
    }

    public List<IEntity> GetReaderList(SqlDataReader reader)
    {
        var lekari = new List<IEntity>();
        var lekarDict = new Dictionary<int, Lekar>();

        while (reader.Read())
        {
            int idLekar = (int)reader["idLekar"];

            if (!lekarDict.TryGetValue(idLekar, out var lekar))
            {
                lekar = new Lekar
                {
                    Id = idLekar,
                    Ime = (string)reader["ime"],
                    Prezime = (string)reader["prezime"],
                    Email = (string)reader["email"],
                    Sifra = (string)reader["sifra"],
                    KorisnickoIme = (string)reader["korisnickoIme"],
                    Sertifikati = new List<Sertifikat>()
                }
            }
        }
    }
}

```

```

        };
        lekarDict[idLekar] = lekar;
    }

    if (reader["SertifikatId"] != DBNull.Value)
    {
        var sertifikat = new Sertifikat
        {
            Id = (int)reader["SertifikatId"],
            Opis = (string)reader["SertifikatOpis"]
        };

        if (!lekar.Sertifikati.Exists(s => s.Id == sertifikat.Id))
            lekar.Sertifikati.Add(sertifikat);
    }

    lekar.AddRange(lekarDict.Values);
    return lekar;
}

public List<SqlParameter> GetSqlParameter()
{
    return new List<SqlParameter>
    {
        new SqlParameter("@Id", Id),
        new SqlParameter("@Ime", Ime),
        new SqlParameter("@Prezime", Prezime),
        new SqlParameter("@Email", Email),
        new SqlParameter("@Sifra", Sifra),
        new SqlParameter("@KorisnickoIme", KorisnickoIme),
    };
}

public List<SqlParameter> GetPrimaryKeyParameters()
{
    return new List<SqlParameter>
    {
        new SqlParameter("@Id", Id)
    };
}

public override string ToString() => DisplayValue;
}

```

```

public class LeS : IEntity
{
    public int IdLekar { get; set; }
    public int IdSertifikat { get; set; }
    public DateTime DatumIzdavanja { get; set; }

    public string TableName => "les";
    public string Columns => "idLekar, idSertifikat, datumIzdavanja";
    public string ValuesClause => "@IdLekar, @IdSertifikat, @DatumIzdavanja";
    public string SetClause => "datumIzdavanja = @DatumIzdavanja";
    public string PrimaryKey => "idLekar AND idSertifikat";
    public string PrimaryKeyCondition => "idLekar = @IdLekar AND idSertifikat = @IdSertifikat";
    public string DisplayValue => DatumIzdavanja.ToShortDateString();

    public (string whereClause, List<SqlParameter> parameters) GetWhereClauseWithParameters()
    {
        var conditions = new List<string>();
        var parameters = new List<SqlParameter>();

        if (IdLekar > 0)
        {
            conditions.Add("idLekar = @IdLekar");
            parameters.Add(new SqlParameter("@IdLekar", IdLekar));
        }

        if (IdSertifikat > 0)
        {
            conditions.Add("idSertifikat = @IdSertifikat");
            parameters.Add(new SqlParameter("@IdSertifikat", IdSertifikat));
        }

        string whereClause = conditions.Count > 0 ? string.Join(" AND ", conditions) : "1=1";
        return (whereClause, parameters);
    }

    public List<IEntity> GetReaderList(SqlDataReader reader)
    {
        var lista = new List<IEntity>();
        while (reader.Read())
        {
            lista.Add(new LeS
            {
                IdLekar = (int)reader["idLekar"],
                IdSertifikat = (int)reader["idSertifikat"],
                DatumIzdavanja = (DateTime)reader["datumIzdavanja"],
            });
        }
        return lista;
    }

    public List<SqlParameter> GetSqlParameter()
    {
        return new List<SqlParameter>
        {
            new SqlParameter("@IdLekar", IdLekar),
            new SqlParameter("@IdSertifikat", IdSertifikat),
            new SqlParameter("@DatumIzdavanja", DatumIzdavanja),
        };
    }

    public List<SqlParameter> GetPrimaryKeyParameters()
    {
        return new List<SqlParameter>
        {
            new SqlParameter("@IdLekar", IdLekar),
            new SqlParameter("@IdSertifikat", IdSertifikat)
        };
    }

    public override string ToString() => DisplayValue;
}

```

```

public class Mesto : ICrudEntity
{
    public int Id { get; set; }
    public string Naziv { get; set; }
    public string PostanskiBroj { get; set; }

    public string TableName => "mesto";
    public string ClassNameAccusativeSingular => "mesto";
    public string ClassNameAccusativePlural => "mesta";
    public string Columns => "naziv, postanskiBroj";
    public string ValuesClause => "@Naziv, @PostanskiBroj";
    public string SetClause => "naziv = @Naziv, postanskiBroj = @PostanskiBroj";
    public string PrimaryKey => "idMesto";
    public string PrimaryKeyCondition => "idMesto = @Id";
    public string DisplayValue => $"{Naziv}";

    public (string whereClause, List<SqlParameter> parameters) GetWhereClauseWithParameters()
    {
        var conditions = new List<string>();
        var parameters = new List<SqlParameter>();

        if (Id > 0)
        {
            conditions.Add("idMesto = @Id");
            parameters.Add(new SqlParameter("@Id", Id));
        }

        if (!string.IsNullOrEmpty(Naziv))
        {
            conditions.Add("naziv LIKE @Naziv");
            parameters.Add(new SqlParameter("@Naziv", $"%{Naziv}%"));
        }

        if (!string.IsNullOrEmpty(PostanskiBroj))
        {
            conditions.Add("postanskiBroj LIKE @PostanskiBroj");
            parameters.Add(new SqlParameter("@PostanskiBroj", $"%{PostanskiBroj}%"));
        }

        return (conditions.Count > 0 ? string.Join(" AND ", conditions) : "1=1", parameters);
    }

    public List<IEntity> GetReaderList(SqlDataReader reader)
    {
        var lista = new List<IEntity>();
        while (reader.Read())
        {
            lista.Add(new Mesto
            {
                Id = (int)reader["idMesto"],
                Naziv = (string)reader["naziv"],
                PostanskiBroj = (string)reader["postanskiBroj"],
            });
        }
        return lista;
    }

    public List<SqlParameter> GetSqlParameter()
    {
        return new List<SqlParameter>
        {
            new SqlParameter("@Id", Id),
            new SqlParameter("@Naziv", Naziv),
            new SqlParameter("@PostanskiBroj", PostanskiBroj),
        };
    }

    public List<SqlParameter> GetPrimaryKeyParameters()
    {
        return new List<SqlParameter>
        {
            new SqlParameter("@Id", Id)
        };
    }

    public override string ToString() => DisplayValue;
}

```



```

public class Pacijent : ICrudEntity
{
    public int Id { get; set; }
    public string Ime { get; set; }
    public string Prezime { get; set; }
    public string Email { get; set; }
    public Mesto Mesto { get; set; }

    public string TableName => "pacijent";
    public string ClassNameAccusativeSingular => "pacijenta";
    public string ClassNameAccusativePlural => "pacijente";
    public string Columns => "ime, prezime, email, idMesto";
    public string ValuesClause => "@Ime, @Prezime, @Email, @IdMesto";
    public string SetClause => "ime = @Ime, prezime = @Prezime, email = @Email, idMesto = @IdMesto";
    public string PrimaryKey => "idPacijent";
    public string PrimaryKeyCondition => "idPacijent = @Id";
    public string JoinTableName => "pacijent p JOIN mesto m ON p.idMesto = m.idMesto";
    public string SelectClause =>
        "p.idPacijent, p.ime, p.prezime, p.email, " +
        "p.idMesto AS PacijentIdMesto, " +
        "m.idMesto AS MestoId, m.naziv AS MestoNaziv, m.postanskiBroj AS MestoPostanskiBroj";
    public string DisplayValue => $"{Ime} {Prezime}";

    public (string whereClause, List<SqlParameter> parameters) GetWhereClauseWithParameters()
    {
        var conditions = new List<string>();
        var parameters = new List<SqlParameter>();

        if (Id > 0)
        {
            conditions.Add("p.idPacijent = @Id");
            parameters.Add(new SqlParameter("@Id", Id));
        }

        if (!string.IsNullOrEmpty(Ime))
        {
            conditions.Add("p.ime LIKE @Ime");
            parameters.Add(new SqlParameter("@Ime", $"%{Ime}%"));
        }

        if (!string.IsNullOrEmpty(Prezime))
        {
            conditions.Add("p.prezime LIKE @Prezime");
            parameters.Add(new SqlParameter("@Prezime", $"%{Prezime}%"));
        }

        if (Mesto != null && Mesto.Id > 0)
        {
            conditions.Add("p.idMesto = @IdMesto");
            parameters.Add(new SqlParameter("@IdMesto", Mesto.Id));
        }

        return (conditions.Count > 0 ? string.Join(" AND ", conditions) : "1=1", parameters);
    }

    public List<IEntity> GetReaderList(SqlDataReader reader)
    {
        var lista = new List<IEntity>();
        while (reader.Read())
        {
            lista.Add(new Pacijent
            {
                Id = (int)reader["idPacijent"],
                Ime = (string)reader["ime"],
                Prezime = (string)reader["prezime"],
                Email = (string)reader["email"],
                Mesto = new Mesto
                {
                    Id = (int)reader["MestoId"],
                    Naziv = (string)reader["MestoNaziv"],
                    PostanskiBroj = (string)reader["MestoPostanskiBroj"]
                }
            });
        }
        return lista;
    }

    public List<SqlParameter> GetSqlParameters()
    {
        return new List<SqlParameter>
        {
            new SqlParameter("@Id", Id),
            new SqlParameter("@Ime", Ime),
            new SqlParameter("@Prezime", Prezime),
            new SqlParameter("@Email", Email),
            new SqlParameter("@IdMesto", Mesto?.Id ?? (object)DBNull.Value)
        };
    }

    public List<SqlParameter> GetPrimaryKeyParameters()
    {
        return new List<SqlParameter>
        {
            new SqlParameter("@Id", Id)
        };
    }

    public override string ToString() => DisplayValue;
}

```

```

public class Sertifikat : ICrudEntity
{
    public int Id { get; set; }
    public string Opis { get; set; }

    public string TableName => "sertifikat";
    public string ClassNameAccusativeSingular => "sertifikat";
    public string ClassNameAccusativePlural => "sertifikate";
    public string Columns => "opis";
    public string ValuesClause => "@Opis";
    public string SetClause => "opis = @Opis";
    public string PrimaryKey => "idSertifikat";
    public string PrimaryKeyCondition => "idSertifikat = @Id";
    public string DisplayValue => Opis;

    public (string whereClause, List<SqlParameter> parameters) GetWhereClauseWithParameters()
    {
        var conditions = new List<string>();
        var parameters = new List<SqlParameter>();

        if (Id > 0)
        {
            conditions.Add("idSertifikat = @Id");
            parameters.Add(new SqlParameter("@Id", Id));
        }

        if (!string.IsNullOrEmpty(Opis))
        {
            conditions.Add("opis LIKE @Opis");
            parameters.Add(new SqlParameter("@Opis", $"%{Opis}%"));
        }

        return (conditions.Count > 0 ? string.Join(" AND ", conditions) : "1=1", parameters);
    }

    public List<IEntity> GetReaderList(SqlDataReader reader)
    {
        var lista = new List<IEntity>();
        while (reader.Read())
        {
            lista.Add(new Sertifikat
            {
                Id = (int)reader["idSertifikat"],
                Opis = (string)reader["opis"]
            });
        }
        return lista;
    }

    public List<SqlParameter> GetSqlParameter()
    {
        return new List<SqlParameter>
        {
            new SqlParameter("@Id", Id),
            new SqlParameter("@Opis", Opis)
        };
    }

    public List<SqlParameter> GetPrimaryKeyParameters()
    {
        return new List<SqlParameter>
        {
            new SqlParameter("@Id", Id)
        };
    }

    public override string ToString() => DisplayValue;
}

```

```

public class StavkaZdravstvenogKartona : IEntity
{
    public int RedniBroj { get; set; }
    public ZdravstveniKarton ZdravstveniKarton { get; set; } = new();
    public DateTime DatumUpisa { get; set; }
    public double Ponder { get; set; }
    public double Skor { get; set; }
    public Dijagnoza Dijagnoza { get; set; } = new();

    public string TableName => "StavkaZdravstvenogKartona";
    public string Columns => "idZdravstveniKarton, datumUpisa, ponder, idDijagnoza";
    public string ValuesClause => "@IdZdravstveniKarton, @DatumUpisa, @Ponder, @IdDijagnoza";
    public string SetClause => "datumUpisa=@DatumUpisa, ponder=@Ponder, idDijagnoza=@IdDijagnoza";
    public string PrimaryKey => "idZdravstveniKarton, rb";
    public string PrimaryKeyCondition => "idZdravstveniKarton = @IdZdravstveniKarton AND rb = @RedniBroj";
    public string DisplayValue => $"{Dijagnoza?.Naziv} | {Ponder}";

    public (string whereClause, List<SqlParameter> parameters) GetWhereClauseWithParameters()
    {
        var conditions = new List<string>();
        var parameters = new List<SqlParameter>();

        if (ZdravstveniKarton != null && ZdravstveniKarton.Id > 0)
        {
            conditions.Add("idZdravstveniKarton IN (SELECT idZdravstveniKarton FROM ZdravstveniKarton WHERE idZdravstveniKarton = @IdZdravstveniKarton)");
            parameters.Add(new SqlParameter("@IdZdravstveniKarton", ZdravstveniKarton.Id));
        }

        string whereClause = conditions.Count > 0 ? string.Join(" AND ", conditions) : "1=1";
        return (whereClause, parameters);
    }

    public List<IEntity> GetReaderList(SqlDataReader reader)
    {
        var lista = new List<IEntity>();
        while (reader.Read())
        {
            lista.Add(new StavkaZdravstvenogKartona
            {
                RedniBroj = (int)reader["rb"],
                ZdravstveniKarton = new ZdravstveniKarton { Id = (int)reader["idZdravstveniKarton"] },
                DatumUpisa = (DateTime)reader["datumUpisa"],
                Ponder = (double)reader["ponder"],
                Skor = (double)reader["skor"],
                Dijagnoza = new Dijagnoza { Id = (int)reader["idDijagnoza"] }
            });
        }
        return lista;
    }

    public List<SqlParameter> GetSqlParameters() => new()
    {
        new SqlParameter("@RedniBroj", RedniBroj),
        new SqlParameter("@IdZdravstveniKarton", ZdravstveniKarton.Id),
        new SqlParameter("@DatumUpisa", DatumUpisa),
        new SqlParameter("@Ponder", Ponder),
        new SqlParameter("@IdDijagnoza", Dijagnoza.Id),
    };

    public List<SqlParameter> GetPrimaryKeyParameters() => new()
    {
        new SqlParameter("@IdZdravstveniKarton", ZdravstveniKarton.Id),
        new SqlParameter("@RedniBroj", RedniBroj)
    };

    public override string ToString() => DisplayValue;
}

```

```

public class ZdravstveniKarton : ICrudEntity
{
    public int Id { get; set; }
    public DateTime DatumOtvaranja { get; set; }
    public string Napomena { get; set; }
    public string Stanje { get; set; }
    public double UkupniSkor { get; set; }
    public Lekar Lekar { get; set; } = new();
    public Pacijent Pacijent { get; set; } = new();
    public List<StavkaZdravstvenogKartona> Stavke { get; set; } = new();

    public string TableName => "ZdravstveniKarton";
    public string ClassNameAccusativeSingular => "zdravstveni karton";
    public string ClassNameAccusativePlural => "zdravstvene kartone";
    public string JoinTableName =>
        @"ZdravstveniKarton zk
        JOIN Lekar l ON zk.idLekar = l.idLekar
        JOIN Pacijent p ON zk.idPacijent = p.idPacijent
        JOIN Mesto m ON p.idMesto = m.idMesto
        LEFT JOIN StavkaZdravstvenogKartona sz ON zk.idZdravstveniKarton =
sz.idZdravstveniKarton
        LEFT JOIN Dijagnoza d ON sz.idDijagnoza = d.idDijagnoza";
    public string SelectClause =>
        @"zk.idZdravstveniKarton, zk.datumOtvaranja, zk.napomena, zk.ukupniSkor, zk.stanje,
        l.idLekar, l.ime AS imeLekara, l.prezime AS prezimeLekara, l.email AS emailLekara,
        p.idPacijent, p.ime AS imePacijenta, p.prezime AS prezimePacijenta, p.email AS
        emailPacijenta,
        m.idMesto, m.naziv AS nazivMesta, m.postanskiBroj,
        sz.rb, sz.datumUpisa, sz.ponder, sz.skor,
        d.idDijagnoza, d.naziv AS nazivDijagnoze, d.opis AS opisDijagnoze, d.bazniSkor";
    public string Columns => "datumOtvaranja, napomena, idLekar, idPacijent";
    public string ValuesClause => "@DatumOtvaranja, @Napomena, @IdLekar, @IdPacijent";
    public string SetClause => "datumOtvaranja=@DatumOtvaranja, napomena=@Napomena, idLekar=@IdLekar,
idPacijent=@IdPacijent";
    public string PrimaryKey => "idZdravstveniKarton";
    public string PrimaryKeyCondition => "idZdravstveniKarton = @Id";
    public string DisplayValue => $"ZK #{Id} - {Stanje}";

    public (string whereClause, List<SqlParameter> parameters) GetWhereClauseWithParameters()
    {
        var conditions = new List<string>();
        var parameters = new List<SqlParameter>();

        if (Id > 0)
        {
            conditions.Add("zk.idZdravstveniKarton = @Id");
            parameters.Add(new SqlParameter("@Id", Id));
        }

        if (Pacijent?.Ime is not null)
        {
            conditions.Add("p.ime LIKE @ImePacijent");
            parameters.Add(new SqlParameter("@ImePacijent", $"{Pacijent.Ime}%"));
        }

        if (Lekar?.Ime is not null)
        {
            conditions.Add("l.ime LIKE @ImeLekar");
            parameters.Add(new SqlParameter("@ImeLekar", $"{Lekar.Ime}%"));
        }

        if (DatumOtvaranja > DateTime.MinValue)
        {
            conditions.Add("zk.datumOtvaranja >= @DatumOtvaranja");
            parameters.Add(new SqlParameter("@DatumOtvaranja", DatumOtvaranja));
        }

        if (Stavke?.Count > 0 && Stavke[0]?.Dijagnoza?.Id != -1)
        {
            conditions.Add("d.idDijagnoza = @IdDijagnoza");
            parameters.Add(new SqlParameter("@IdDijagnoza", Stavke[0].Dijagnoza.Id));
        }

        return (conditions.Count > 0 ? string.Join(" AND ", conditions) : "1=1", parameters);
    }

    public List<IEntity> GetReaderList(SqlDataReader reader)
    {
        var mapa = new Dictionary<int, ZdravstveniKarton>();
        while (reader.Read())
        {
            int zkId = (int)reader["idZdravstveniKarton"];
            if (!mapa.ContainsKey(zkId))
            {
                var zk = new ZdravstveniKarton
                {
                    Id = zkId,
                    DatumOtvaranja = (DateTime)reader["datumOtvaranja"],
                    Napomena = (string)reader["napomena"],
                    UkupniSkor = (double)reader["ukupniSkor"],
                    Stanje = (string)reader["stanje"],
                    Lekar = new Lekar
                    {
                        Id = (int)reader["idLekar"],
                        Ime = (string)reader["imeLekara"],
                        Prezime = (string)reader["prezimeLekara"],
                        Email = (string)reader["emailLekara"]
                    }
                };
            }
        }
    }
}

```

```

    },
    Pacijent = new Pacijent
    {
        Id = (int)reader["idPacijent"],
        Ime = (string)reader["imePacijenta"],
        Prezime = (string)reader["prezimePacijenta"],
        Email = (string)reader["emailPacijenta"],
        Mesto = new Mesto
        {
            Id = (int)reader["idMesto"],
            Naziv = (string)reader["nazivMesta"],
            PostanskiBroj = (string)reader["postanskiBroj"]
        }
    },
    Stavke = new List<StavkaZdravstvenogKartona>()
};

mapa.Add(zkId, zk);
}

if (reader["rb"] != DBNull.Value)
{
    mapa[zkJd].Stavke.Add(new StavkaZdravstvenogKartona
    {
        RedniBroj = (int)reader["rb"],
        DatumUpisa = (DateTime)reader["datumUpisa"],
        Ponder = (double)reader["ponder"],
        Skor = (double)reader["skor"],
        Dijagnoza = new Dijagnoza
        {
            Id = (int)reader["idDijagnoza"],
            Naziv = (string)reader["nazivDijagnoze"],
            Opis = (string)reader["opisDijagnoze"],
            BazniSkor = (double)reader["bazniSkor"]
        }
    });
}

return new List<IEntity>(mapa.Values);
}

public List<SqlParameter> GetSqlParameter() => new()
{
    new SqlParameter("@Id", Id),
    new SqlParameter("@DatumOtvaranja", DatumOtvaranja),
    new SqlParameter("@Napomena", Napomena ?? (object)DBNull.Value),
    new SqlParameter("@IdLekar", Lekar?.Id ?? (object)DBNull.Value),
    new SqlParameter("@IdPacijent", Pacijent?.Id ?? (object)DBNull.Value),
};

public List<SqlParameter> GetPrimaryKeyParameters() =>
    new() { new SqlParameter("@Id", Id) };

public override string ToString() => DisplayValue;
}

```

У оквиру овог пројекта реализована је врста мини ORM-а (Object-Relational Mapping) система који омогућава лакшу и јаснију комуникацију између апликације и базе података. Доменске класе имплементирају интерфејс *IEntity* и дефинишу своје мапирање на табеле базе кроз одговарајуће SQL параметре и упите, чиме апстрахују приступ бази података. Кључне особине овог интерфејса укључују:

- *TableName* – одређује назив табеле у бази на коју ентитет мапира
- *Columns* и *ValuesClause* – дефинишу које колоне се убацују и како се вредности параметризују приликом уметања података
- *SetClause* – користи се у SQL наредбама за ажурирање постојећих записа
- *PrimaryKey* и *PrimaryKeyCondition* – одређују примарни кључ и услов за идентификацију појединачног записа
- *GetWhereClauseWithParameters()* – генерише услове и параметре за претрагу према тренутним вредностима својстава објекта
- *GetSqlParameters()* и *GetPrimaryKeyParameters()* – припремају листу SQL параметара за извршавање упита
- *GetReaderList()* – трансформише резултате из базе (SQL reader) у листу конкретних доменских објеката.

На овај начин свака доменска класа „носи“ своју логику за мапирање и рад са подацима, чиме се постиже висок ниво апстракције и смањује понављање кода. Посебно је значајно што овај приступ омогућава рад са комплексним ентитетима који имају везе са другим класама, као што су релације између здравственог картона, лекара, пацијента и дијагнозе. Коришћењем својстава као што су *JoinTableName* и *SelectClause*, ове класе могу да конструишу сложене SQL упите за добијање свеобухватних и повезаних података.

Оваква имплементација мини ORM-а даје добру равнотежу између контроле над SQL кодом и лакоће рада са подацима на објектно-оријентисан начин, што доприноси одрживости и проширивости апликације.

## Пројектовање брокера базе података

Класа *Broker* из пројекта *DBBroker* представља перзистентан оквир који посредује у свим операцијама над базом података и реализује следеће методе:

```
public void Add(IEntity obj);
public int AddWithReturnId(IEntity entity);
public void Update(IEntity obj);
public void Delete(IEntity obj);
public List<IEntity> GetAll(IEntity entity);
public List<IEntity> GetByCondition(IEntity entity);
public int GetFirstId(IEntity entity);
```

Све методе класе *Broker* су пројектоване као генеричке, што значи да могу да прихвате различите доменске објекте преко параметара. Ово је остварено дефинисањем интерфејса *IEntity* који имплементирају све доменске класе:

```
public interface IEntity
{
    string TableName { get; }
    string Columns { get; }
    string ValuesClause { get; }
    string SetClause { get; }
    public string PrimaryKey { get; }
    string PrimaryKeyCondition { get; }
    string? JoinTableName => null;
    string? SelectClause => null;
    string DisplayValue { get; }
    (string whereClause, List<SqlParameter> parameters) GetWhereClauseWithParameters();

    List<IEntity> GetReaderList(SqlDataReader reader);
    List<SqlParameter> GetSqlParameters();
    List<SqlParameter> GetPrimaryKeyParameters();
}

public interface ICrudEntity : IEntity
{
    int Id { get; set; }
    string ClassNameAccusativeSingular { get; }
    string ClassNameAccusativePlural { get; }
}
```

### 4.3 Пројектовање складишта података

На основу доменских класа софтвера пројектоване су табеле у оквиру релационог система за управљање базом података. Као систем за управљање базом података коришћен је **Microsoft SQL Server**, а за дефинисање и манипулацију подацима употребљен је језик **Transact-SQL (T-SQL)**, који представља проширење стандардног SQL-а, специфично за ову платформу.

#### Преглед табела

- Табела Lekar

	TABLE_NAME	COLUMN_NAME	DATA_TYPE	CHARACTER_MAXIMUM_LENGTH	IS_NULLABLE	COLUMN_DEFAULT
1	Lekar	idLekar	int	NULL	NO	NULL
2	Lekar	ime	nvarchar	100	NO	NULL
3	Lekar	prezime	nvarchar	100	NO	NULL
4	Lekar	email	nvarchar	100	NO	NULL
5	Lekar	korisnickolme	nvarchar	100	NO	NULL
6	Lekar	sifra	char	64	NO	NULL

	TableName	ColumnName	CheckConstraintName	CheckDefinition
1	Lekar	email	CK__Lekar__email__2EA5EC27	(([email] like '_%@_%._%'))

- Табела Dijagnoza

	TABLE_NAME	COLUMN_NAME	DATA_TYPE	CHARACTER_MAXIMUM_LENGTH	IS_NULLABLE	COLUMN_DEFAULT
1	Dijagnoza	idDijagnoza	int	NULL	NO	NULL
2	Dijagnoza	naziv	nvarchar	100	NO	NULL
3	Dijagnoza	opis	nvarchar	255	NO	NULL
4	Dijagnoza	bazniSkor	float	NULL	NO	NULL

	TableName	ColumnName	CheckConstraintName	CheckDefinition
1	Dijagnoza	bazniSkor	CK__Dijagnoza__bazni__318258D2	(([bazniSkor]>(0)))

- Табела Mesto

	TABLE_NAME	COLUMN_NAME	DATA_TYPE	CHARACTER_MAXIMUM_LENGTH	IS_NULLABLE	COLUMN_DEFAULT
1	Mesto	idMesto	int	NULL	NO	NULL
2	Mesto	naziv	nvarchar	100	NO	NULL
3	Mesto	postanskiBroj	char	5	NO	NULL

	TableName	ColumnName	CheckConstraintName	CheckDefinition
1	Mesto	postanskiBroj	CK__Mesto__postanski__345EC57D	(([postanskiBroj] like '[0-9][0-9][0-9][0-9][0-9]'))

- Табела Sertifikat

	TABLE_NAME	COLUMN_NAME	DATA_TYPE	CHARACTER_MAXIMUM_LENGTH	IS_NULLABLE	COLUMN_DEFAULT
1	Sertifikat	idSertifikat	int	NULL	NO	NULL
2	Sertifikat	opis	nvarchar	255	NO	NULL



- Табела Pacijent

	TABLE_NAME	COLUMN_NAME	DATA_TYPE	CHARACTER_MAXIMUM_LENGTH	IS_NULLABLE	COLUMN_DEFAULT
1	Pacijent	idPacijent	int	NULL	NO	NULL
2	Pacijent	ime	nvarchar	100	NO	NULL
3	Pacijent	prezime	nvarchar	100	NO	NULL
4	Pacijent	email	nvarchar	100	NO	NULL
5	Pacijent	idMesto	int	NULL	NO	NULL

	TableName	ColumnName	CheckConstraintName	CheckDefinition
1	Pacijent	email	CK_Pacijent_email__39237A9A	((email) like ' _%@_%. _%')
2	Pacijent	idMesto	CK_Pacijent_idMest__3A179ED3	((idMesto)>(0))

- Табела ZdravstveniKarton

	TABLE_NAME	COLUMN_NAME	DATA_TYPE	CHARACTER_MAXIMUM_LENGTH	IS_NULLABLE	COLUMN_DEFAULT
1	ZdravstveniKarton	idZdravstveniKarton	int	NULL	NO	NULL
2	ZdravstveniKarton	datumOtvaranja	date	NULL	NO	NULL
3	ZdravstveniKarton	napomena	nvarchar	255	NO	NULL
4	ZdravstveniKarton	ukupniSkor	float	NULL	NO	((0))
5	ZdravstveniKarton	stanje	nvarchar	20	NO	('Zdrav')
6	ZdravstveniKarton	idLekar	int	NULL	NO	NULL
7	ZdravstveniKarton	idPacijent	int	NULL	NO	NULL

	TableName	ColumnName	CheckConstraintName	CheckDefinition
1	ZdravstveniKarton	stanje	CK_Zdravstve__stanj__3EDC53F0	((stanje)='Teže bolestan' OR (stanje)='Lakše bolestan' OR (stanje)='Zdrav')
2	ZdravstveniKarton	idLekar	CK_Zdravstve__idLek__40C49C62	((idLekar)>(0))
3	ZdravstveniKarton	idPacijent	CK_Zdravstve__idPac__41B8C09B	((idPacijent)>(0))

- Табела StavkaZdravstvenogKartona

	TABLE_NAME	COLUMN_NAME	DATA_TYPE	CHARACTER_MAXIMUM_LENGTH	IS_NULLABLE	COLUMN_DEFAULT
1	StavkaZdravstvenogKartona	idZdravstveniKarton	int	NULL	NO	NULL
2	StavkaZdravstvenogKartona	rb	int	NULL	NO	NULL
3	StavkaZdravstvenogKartona	datumUpisa	date	NULL	NO	NULL
4	StavkaZdravstvenogKartona	ponder	float	NULL	NO	((1))
5	StavkaZdravstvenogKartona	skor	float	NULL	NO	((0))
6	StavkaZdravstvenogKartona	idDijagnoza	int	NULL	NO	NULL

	TableName	ColumnName	CheckConstraintName	CheckDefinition
1	StavkaZdravstvenogKartona	idZdravstveniKarton	CK_StavkaZdr__idZdr__467D75B8	((idZdravstveniKarton)>(0))
2	StavkaZdravstvenogKartona	ponder	CK_StavkaZdr__ponde__4865BE2A	((ponder)>(0))
3	StavkaZdravstvenogKartona	idDijagnoza	CK_StavkaZdr__idDij__4A4E069C	((idDijagnoza)>(0))

- Табела LeS

	TABLE_NAME	COLUMN_NAME	DATA_TYPE	CHARACTER_MAXIMUM_LENGTH	IS_NULLABLE	COLUMN_DEFAULT
1	LeS	idLekar	int	NULL	NO	NULL
2	LeS	idSertifikat	int	NULL	NO	NULL
3	LeS	datumIzdavanja	date	NULL	NO	NULL

	TableName	ColumnName	CheckConstraintName	CheckDefinition
1	LeS	idLekar	CK_LeS_idLekar__4F12BBB9	((idLekar)>(0))
2	LeS	idSertifikat	CK_LeS_idSertifika__5006DFF2	((idSertifikat)>(0))

Резултати приказани изнад добијени су упитима над системским погледима базе података. Конкретно, коришћени су *INFORMATION\_SCHEMA.COLUMNS* за добијање информација о колонама (назив, тип, дужина, NULL ограничење, подразумевана вредност) и *sys.check\_constraints* у комбинацији са *sys.columns* и *sys.objects* за приказ дефиниција CHECK ограничења. Ови системски погледи представљају метаподатке које SQL Server аутоматски одржава и омогућавају директан преглед и анализу структуре базе.

## Тригери

У оквиру дизајна складишта података, дефинисани су и тригери који омогућавају аутоматско одржавање доследности и ажурирање агрегираних вредности.

Конкретно, након уноса или измене ставке здравственог картона, тригер *trg\_AzuriranjeSkor* рачуна вредност појединачне ставке на основу пондера и базног скорa дијагнозе.

```
CREATE TRIGGER trg_AzuriranjeSkor
ON StavkaZdravstvenogKartona
AFTER INSERT, UPDATE
AS
BEGIN
    SET NOCOUNT ON;

    UPDATE szk
    SET skor = szk.ponder * d.bazniSkor
    FROM StavkaZdravstvenogKartona szk
    INNER JOIN inserted i ON szk.idZdravstveniKarton = i.idZdravstveniKarton AND szk.rb = i.rb
    INNER JOIN Dijagnoza d ON szk.idDijagnoza = d.idDijagnoza;
END;
```

Други тригер, *trg\_AzuriranjeUkupniSkorStanje*, аутоматски ажурира укупни скор и стање здравственог картона у складу са унетим ставкама, чиме се обезбеђује доследност без потребе за ручним интервенцијама.

```
CREATE TRIGGER trg_AzuriranjeUkupniSkorStanje
ON StavkaZdravstvenogKartona
AFTER INSERT, UPDATE, DELETE
AS
BEGIN
    SET NOCOUNT ON;

    DECLARE @AffectedKartons TABLE (idZdravstveniKarton INT PRIMARY KEY);

    INSERT INTO @AffectedKartons(idZdravstveniKarton)
    SELECT idZdravstveniKarton FROM inserted
    UNION
    SELECT idZdravstveniKarton FROM deleted;

    UPDATE zk
    SET
        ukupniSkor = ISNULL(suma.sumaSkor, 0),
        stanje = CASE
            WHEN ISNULL(suma.sumaSkor, 0) <= 100 THEN 'Zdrav'
            WHEN ISNULL(suma.sumaSkor, 0) <= 500 THEN 'Lakše bolestan'
            ELSE 'Teže bolestan'
        END
    FROM ZdravstveniKarton zk
    INNER JOIN @AffectedKartons ak ON zk.idZdravstveniKarton = ak.idZdravstveniKarton
    LEFT JOIN (
        SELECT idZdravstveniKarton, SUM(skor) AS sumaSkor
        FROM StavkaZdravstvenogKartona
        GROUP BY idZdravstveniKarton
    ) suma ON zk.idZdravstveniKarton = suma.idZdravstveniKarton;
END;
```

Оваквим приступом, део пословне логике се премешта са апликативног слоја у саму базу података, што омогућава **растерећење клијентске апликације**, смањење потенцијалних грешака у синхронизацији и **обезбеђује интегритет и конзистентност података директно на нивоу базе**.

## 5. Имплементација

Софтверски систем реализован је у програмском језику **C#**, уз коришћење .NET Windows Forms технологије за развој графичког корисничког интерфејса и **Microsoft SQL Server** као система за управљање базом података. Апликација је развијена као **трослојна клијент-сервер архитектура**, при чему сервер прихвата захтеве клијента преко **TCP комуникације**, са JSON сериализацијом захтева и одговора.

Пројекат је подељен у четири главна модула:

- **Client** – садржи Windows Forms форме, логику за приказ и унос података, сервисне класе које обављају CRUD операције, као и механизме за креирање и слање захтева ка серверу. Клијент користи образац *Template Method* у сервисним класама (нпр. *BaseEntityService*) како би дефинисао општи ток CRUD операција са могућношћу прилагођавања специфичних корака за сваки ентитет. Управљање формама је централизовано кроз класу *FormManager*, која помаже у отварању, затварању и валидацији форми.
- **Server** – садржи компоненте које примају захтеве од клијента, декодирају их и прослеђују одговарајућим системским операцијама (класама које реализују конкретне пословне логике). Овде је примењен *Template Method* образац у класи *SystemOperationBase*, која дефинише општи ток извршења системских операција, док специфичне операције имплементирају конкретне класе (нпр. *KreirajLekarSO*). Сервер такође управља конфигурацијом и одржава конекцију са базом.
- **Common** – обухвата доменске класе које представљају ентитете у систему и класе за дефиницију захтева и одговора. Омогућава дељење заједничких компоненти између клијента и сервера.
- **DBBroker** – задужен за приступ релационој бази података. Овде се налазе класе које реализују CRUD операције ка бази података, кроз коришћење SQL упита. Апстрахује конкретне детаље рада са базом, омогућавајући једноставнију измену и одржавање.

Имплементација је у потпуности усклађена са претходно дефинисаним слојевима, дизајном контролера и сценаријима коришћења. Серверски пројекат је апсолутно *thread safe*, што значи да је све конструисано тако да истовремени приступи више различитих нити не изазивају нежељена стања или грешке. Оваква организација омогућила је јасну поделу одговорности, лако одржавање и проширивост система у будућности.

У току развоја примењени су **неки од дизајн патерна**, од којих су најзначајнији:

- **Template Method pattern** – примењен је у апстрактној класи *SystemOperationBase* на серверу, која дефинише општи ток извршавања системских операција, са конкретним корацима који се дефинишу у наследницима. Овај образац је такође коришћен у сервисним класама на клијенту (нпр. у *BaseEntityService*), чиме је обезбеђена унификована и проширива логика CRUD операција.
- **Builder pattern** – коришћен у клијентском слоју за креирање критеријума претраге (*CriteriaBuilderBase*, *LekarCriteriaBuilder*, итд), што омогућава флексибилну и типизiranу изградњу SQL WHERE услова.
- **Singleton pattern** – примењен је у менаџерским класама као што су *FormManager* и у контролерима, како би се обезбедило постојање само једне инстанце која управља ресурсима као што су отворене форме и општа логика контроле протока. Ово омогућава централизацију управљања и спречава стварање конфликта или вишка инстанци.
- **Observer pattern** – коришћен на серверу за обавештавање корисничког интерфејса о промени броја повезаних клијената. Сервер издаје догађаје које слуша GUI, чиме је омогућена реактивна и синхронизована комуникација између слојева без директне зависности.

Уочљива је и примена *Data Mapper* патерна, где ентитети имплементирају интерфејс *IEntity* и енкапсулирају логику за мапирање између релационог модела базе података и објекта доменског модела. Поред тога, у целокупној архитектури приметна је и прилагођена примена *MVC (Model–View–Controller)* патерна, при чему су модел (ентитети и логика приступа подацима), приказ (*Windows Forms* форме) и контролер (сервисне и управљачке класе) логички одвојени. Иако није реч о класичном *MVC* патерну, ова подела одговара духу *MVC*-а и прилагођена је захтевима *Windows Forms* апликације.

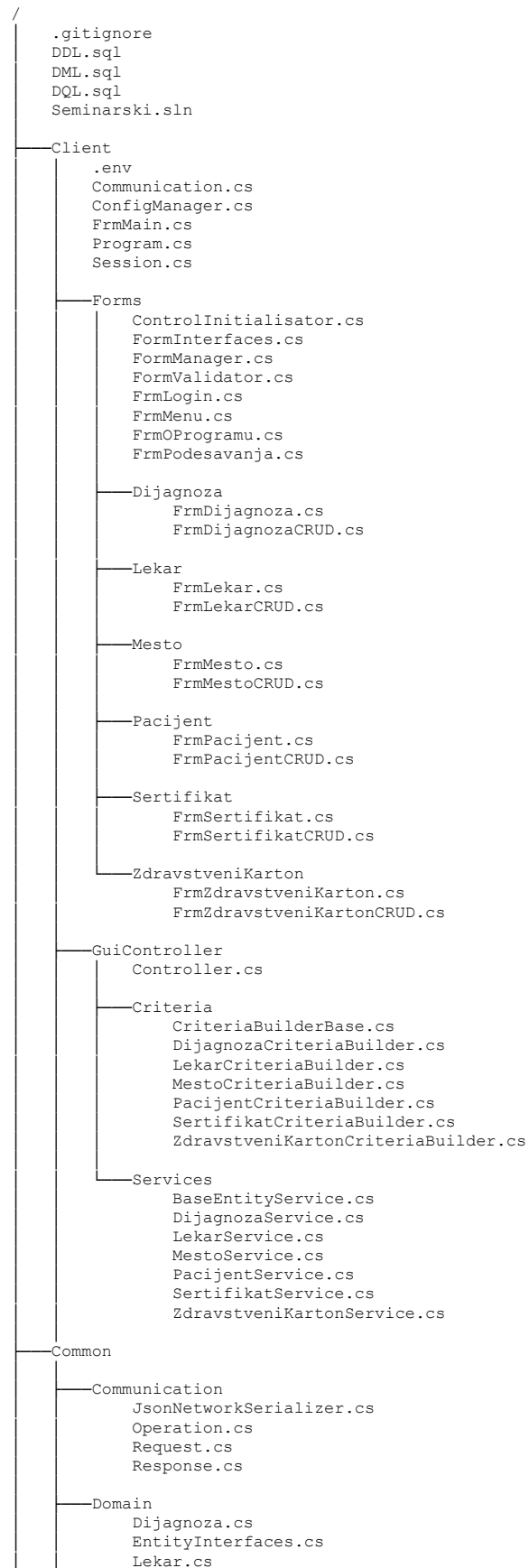
Осим примене дизајн патерна, у имплементацији су коришћени и генерички типови, што омогућава унификовану обраду различитих ентитета без дуплирања кода.

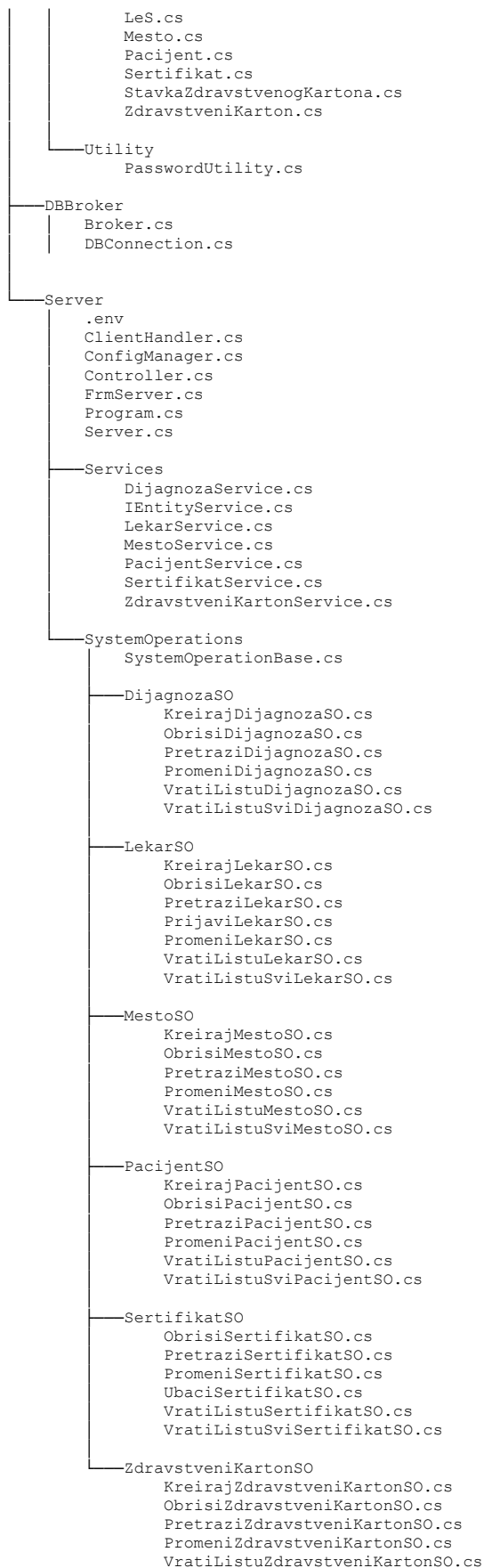
Такође, током развоја пројекта, водило се рачуна о поштовању савремених принципа безбедности, одрживости и професионалног развоја софтвера. Испоштоване су све праксе које се сматрају добрим у индустрији, као што су:

**Лозинке се не чувају у отвореном тексту**, већ се пре уноса у базу података хешују применом **SHA-256** алгоритма, а добијени хеш се енкодира и чува у бази у **Base64** формату.

За потребе верзионисања кода коришћен је **Git систем за контролу верзија**, чиме је омогућено праћење историје измена и сигурно управљање кодом. Конфигурациони подаци као што су IP адреса сервера, порт и connection string издвојени су у *.env* датотеке на клијенту и серверу, чиме је омогућено **лакше одржавање и независност од окружења**.

## Структура пројекта





## 6. Тестирање

Након имплементације, систем је подвргнут свеобухватном ручном тестирању у циљу провере његове исправности, поузданости и усклађености са функционалним захтевима. Сваки од дефинисаних сценарија случајева коришћења пажљиво је испитан, укључујући типичне токове рада, као и граничне и неочекиване ситуације. Посебна пажња посвећена је валидацији уноса, где су тестирани и исправни и неисправни подаци, како би се проверило да ли систем адекватно реагује у складу са очекиваним понашањем – односно да ли правилно обрађује исправне податке и благовремено пријављује грешке у случају невалидног уноса.

Тестирани су и сви механизми заштите интегритета података, укључујући CHECK ограничења, стране кључеве, подразумеване вредности и тригере који аутоматски одржавају доследност израчунатих вредности. Провера је обухватила и функционалности графичког корисничког интерфејса, као што су приказ, претрага, креирање, измена и брисање података, како би се уверило да корисничко искуство буде интуитивно, стабилно и без грешака.

Такође, посебно је наглашено да је серверски део апликације у потпуности *thread safe*. Сви делови кода који обрађују вишеструке клијентске захтеве обезбеђени су адекватном синхронизацијом и контролом приступа заједничким ресурсима, што спречава *race condition* и друге проблеме везане за паралелно извршавање. У оквиру тога, све серверске класе које су имплементирание као *singleton* инстанце представљају *thread safe singleton* обрасце, што значи да је створена само једна инстанца сваке класе, без могућности конфликтних стања у вишенитном окружењу.

Тестирање је спроведено ручно, симулацијом реалног рада корисника у различитим сценаријима, што је омогућило да се уоче и исправе потенцијалне грешке пре пуштања система у рад.

Након успешно завршене фазе тестирања, систем се сматра зрелим за продукциону употребу и спреман је за коришћење од стране крајњег корисника.

## 7. Закључак

Резултат овог семинарског рада је стабилан, функционалан и прегледно структуриран **софтверски систем за праћење рада дома здравља**, који омогућава ефикасно управљање подацима о пацијентима, лекарима, дијагнозама, сертификатима и здравственим картонима.

Систем је развијен у складу са савременим принципима софтверске архитектуре, уз поштовање концепта раздвајања слојева и примену више проверених дизајн патерна. Комбинација графичког интерфејса и поуздане комуникације између клијента и сервера чини апликацију употребљивом у реалним условима рада једне здравствене установе.

Интеграција базе података са логиком апликације решена је тако да обезбеђује доследност података, а систем је направљен тако да га је лако одржавати, проширивати и прилагођавати новим потребама. Посебно вреди истаћи добру организацију кода, као и посвећеност безбедности података и флексибилности.

Из личне перспективе, израда овог система представљала је значајно искуство. Суочавао сам се са бројним изазовима – од повезивања слојева, преко дизајнирања базе, па до решавања неочекиваних успутних грешака – али управо су ти изазови допринели да стекнем дубље разумевање начина на који комплексни системи функционишу. Овај пројекат ми је омогућио да повежем теорију са праксом, али и да стекнем веће самопоуздање у своје способности.

Сматрам да је направљен систем који има **стварну применљивост**, што ми је био и главни циљ при његовој реализацији.



## Литература

[1] Влајић, С. (2020). Пројектовање софтвера (скрипта). Београд, Србија: Факултет организационих наука