

**УНИВЕРЗИТЕТ У БЕОГРАДУ**  
**ФАКУЛТЕТ ОРГАНИЗАЦИОНИХ НАУКА**

**ЗАВРШНИ РАД**

**Развој софтверског система за праћење  
студентских обавеза током семестра у Јава  
окружењу**

**Ментор**

**Др Синиша Влајић,  
Редовни професор**

**Студент**

**Владимир Лековац 105/16**

**Београд, 2022. године**

## Садржај:

1. Увод.....	4
2. Упрошћена ларманова метода развоја софтвера .....	5
2.1. Прикупљање корисничких захтева.....	6
2.1.1. Захтеви .....	6
2.1.2. Опис захтева помоћу случаја коришћења .....	6
2.1.3. Начин представљања модела случаја коришћења .....	7
2.2. Анализа.....	7
2.2.1. Понашање софтверског система.....	7
2.2.2. Структура софтверског система .....	8
2.3. Пројектовање .....	9
2.4. Имплементација и тестирање.....	10
3. Јава, јава технологије.....	11
3.1. Концепти објектно оријентисаног програмирања у Јави.....	12
3.1.1. Основни појмови.....	12
3.1.2. Наслеђивање .....	13
3.1.3. Појам апстракције, интерфејси, апстрактне класе.....	14
3.1.4. Изузеци .....	15
3.2. Кориснички интерфејс .....	16
3.2.1. Догађаји .....	17
3.3. Мрежно програмирање.....	18
3.3.1. Адреса рачунара, URL адреса.....	18
3.3.2. Сокети .....	19
3.4. Нити.....	21
3.5. Рад са базом података.....	22
4. Студијски пример .....	23
4.1. Прикупљање корисничких захтева .....	23
4.1.1. Вербални опис .....	23
4.1.2 Модел случаја коришћења .....	24
СК1: Случај коришћења – Пријављивање корисника.....	25
СК2: Случај коришћења – Креирање новог предмета (Сложен случај коришћења) ...	26
СК3: Случај коришћења – Претраживање предмета.....	27
СК4: Случај коришћења – Измена предмета(Сложен случај коришћења) .....	28
СК5: Случај коришћења – Брисање предмета .....	29

СК6: Случај коришћења – Креирање новог плана(Сложен случај коришћења) .....	30
СК7: Случај коришћења – Претраживање плана.....	31
СК8: Случај коришћења – Измена плана (Сложен случај коришћења) .....	32
СК9: Случај коришћења – Брисање плана .....	33
СК10: Случај коришћења – Одјављивање корисника .....	34
4.2. Анализа .....	35
4.2.1. Системски дијаграми секвенци .....	35
ДС1:Дијаграм секвенци случаја коришћења-Пријављивање корисника .....	35
ДС2:Дијаграм секвенци случаја коришћења-Креирање новог предмета.....	37
ДС3:Дијаграм секвенци случаја коришћења- Претраживање предмета .....	39
ДС4:Дијаграм секвенци случаја коришћења- Измена предмета.....	42
ДС5:Дијаграм секвенци случаја коришћења- Брисање предмета.....	46
ДС6:Дијаграм секвенци случаја коришћења-Креирање новог плана.....	50
ДС7:Дијаграм секвенци случаја коришћења-Претраживање плана .....	52
ДС8:Дијаграм секвенци случаја коришћења-Измена плана.....	55
ДС9:Дијаграм секвенци случаја коришћења-Брисање плана.....	59
ДС10:Дијаграм секвенци случаја коришћења-Одјављивање корисника .....	63
4.2.2. Понашање софтверског система – Дефинисање уговора о системским операцијама .....	65
4.2.3. Структура софтверског система – Концептуални модел.....	67
4.2.4. Структура софтверског система – Релациони модел .....	68
4.3. Пројектовање.....	73
4.3.1. Пројектовање корисничког интерфејса .....	74
4.3.2. Пројектовање апликационе логике .....	106
4.3.3. Пројектовање складишта података .....	113
4.4. Имплементација .....	116
4.5. Тестирање .....	118
5. Закључак .....	119
6. Литература.....	120

## 1. Увод

Машинско процесирање информација које надалеко премашује ограничене интелектуалне могућности човека, довело је до разних погодности у његовом животу. Неки од примера су употреба информационих технологија у медицини, или приступ квалитетном образовању без обзира где се налазите или колико новца имате. Међутим, заједно са погодностима, револуција интернет технологија донела је са собом изазове са којима се човечанство није раније сусретало. На пример, веома је лако изгубити појам о времену и провести подоста времена на друштвеним мрежама, што није у складу са људском природом, и на дуже стазе штети хармонији којој човек тежи. Из тог разлога, сматрам да би развој информационих технологија у будућности требало да се заснива на развоју софтверских система, који помажу човеку да успешно организује свој живот и живи у хармонији, уместо да му у томе одмажу. Софтверски систем, развијен у овом раду, за циљ има мали допринос овој тези и намењен је студентима како би лакше водили белешке о својим обавезама на факултету и успешно их савладали.

Циљ рада је комплетан, свеобухватан развој софтверског система који помаже студентима да организују своје обавезе током студирања. У првом поглављу дат је теоријски осврт на свих пет фаза Ларманове методе развоја софтвера - прикупљање корисничких захтева, анализа, пројектовање, имплементација и тестирање.

У имплементацији софтверског система, коришћен је програмски језик Јава. Јава је објектно-орјентисани, платформски независан програмски језик развијен 1995. године од стране инжењера из компаније Sun Microsystems. Јава је постала основ за развој других програмских језика (php, c#) , и задржала је позицију најкоришћенијег програмског језика до данашњег дана. Концепти Јаве и Јава технологија коришћених у овом раду, детаљније су описани у трећем поглављу.

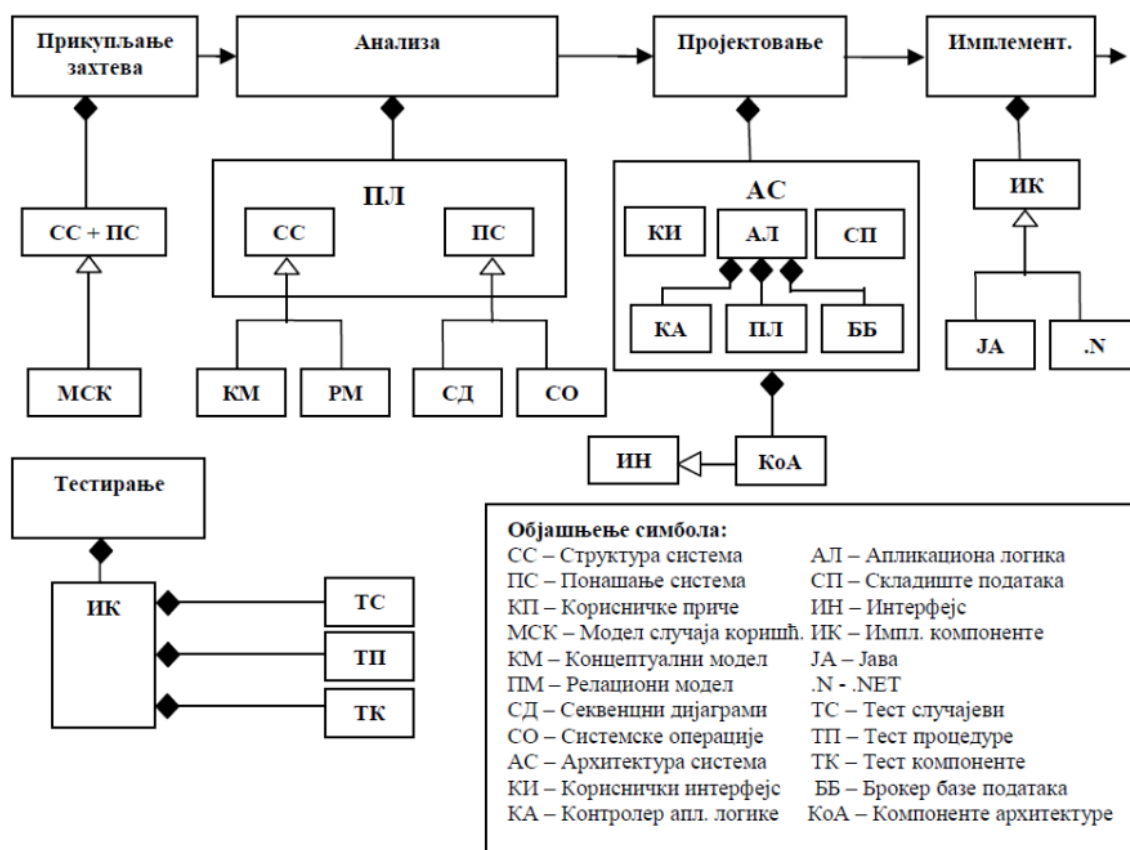
У четвртном поглављу се на студијском примеру примењују концепти Јава технологија и упрошћене Ларманове методе, објашњени у ранијим поглављима. Пето поглавље је везано за закључак , док је у последњем, шестом поглављу наведена литература коришћена у писању овог рада.

## 2. Упрости́ена ларманова метода развоја софтвера

Методологија развоја софтвера коришћена у о овом раду јесте упрости́ена Ларманова метода развоја софтвера [1]. Према упрости́еној Лармановој методи, развој софтверског система састоји се из следећих пет фаза:

- Прикупљање захтева од корисника
- Анализе
- Пројектовања
- Имплементације
- Тестирања

Дијаграм Ларманове методе приказан је на слици 1.



Слика 1. Развој софтверског система по Лармановој методи [1]

## **2.1. Прикупљање корисничких захтева**

### **2.1.1. Захтеви**

Захтеви представљају својства и услове које систем или шире гледајући пројекат мора да задовољи. Сваки софтверски систем мора задовољити различите типове захтева који су категорисани према FURPS+ (Functional – функционалност, Usability - употребљивост, Reliability - поузданост, Performance - перформансе, Supportability - подрживост). У FURPS+ моделу знак ‘+’ указује на помоћне захтеве који се односе на – имплементацију, интерфејсе, операције, паковање и легалност система.

Захтеви се често категоризују као функционални и нефункционални захтеви. Функционални захтеви служе за дефинисање захтеване функције система, док нефункционални захтеви (употребљивост, поузданост, перформансе и подрживост) дефинишу све остале захтеве, и они у том смислу представљају атрибуте квалитета.

У овом раду, главни акценат је на функционалним захтевима, тј. њиховом разматрању од фазе прикупљања до имплементације. У студијском примеру биће уведени неки од нефункционалних захтева који су директно повезани са функционалношћу система (поузданост и подрживост система) као и неки од помоћних захтева (имплементација, операције и паковање система). Ипак, остали нефункционални и помоћни захтеви неће бити обухваћени овим радом, јер у највећој мери нису повезани са функционалношћу система, што представља фокус изучавања.

### **2.1.2. Опис захтева помоћу случаја коришћења**

Код Лармана захтеви се описују помоћу UML модела случаја коришћења. Модел случаја коришћења састоји се од скупа случајева коришћења, актора и веза између случајева коришћења и актора. Модел случаја коришћења везан је за више случајева коришћења, више веза између случајева коришћења и више актора. Сваки случај коришћења може имати више веза са акторима. Свака веза се односи на тачно једног актора и тачно један случај коришћења.

Случај коришћења описује скуп сценарија, односно скуп жељених коришћења система од стране актора. Сваки сценарио описује једно коришћење система од стране актора и састоји се од главног и алтернативних сценарија. У току интеракције актора и система, актор позива системске операције које представљају основне функције система.

Актор представља спољног корисника система, који поставља захтев систему да изврши једну или више системских операција по унапред утврђеном сценарију. Као одговор, актор од система прима вредност излазних аргумената као резултат извршења операција.

### **2.1.3. Начин представљања модела случаја коришћења**

Случајеви коришћења (СК) се у почетним фазама развоја софтвера представљају текстуално док се у наредним фазама користе дијаграми (секвенцни дијаграми, дијаграми сарадње, дијаграми прелаза стања или дијаграм активности).

Текстуални опис СК има следећу структуру:

- Назив СК
- Акторе СК
- Учеснике СК
- Предуслови који морају бити задовољени да би СК почео да се извршава
- Основни сценарио извршења СК
- Постуслови који морају бити задовољени да би се потврдило да је СК успешно извршен
- Алтернатива сценарија извршења СК
- Специјални захтеви
- Технолошки захтеви
- Отворена питања

## **2.2. Анализа**

Фаза анализе описује логичку структуру и понашање софтверског система. Понашање софтверског система описује се помоћу системских дијаграма секвенци и преко системских операција док се структура софтверског система описује коришћењем концептуалног и релационог модела.

### **2.2.1. Понашање софтверског система**

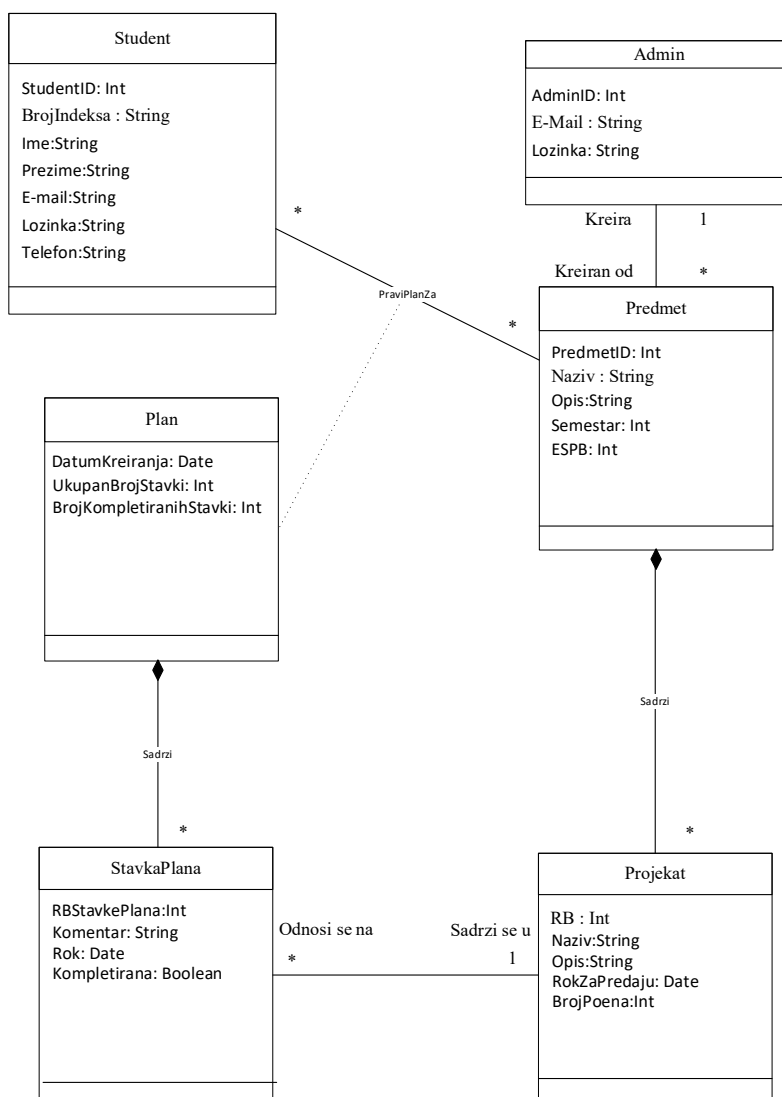
За посматрани случај коришћења, системски дијаграм секвенци приказује редослед догађаја, који успостављају интеракцију између актора и софтверског система. Догађај који направи актор је побуда за позив системске операције. То значи да актор не позива директно системске операције, већ то врши преко примаоца догађаја који прихвата догађај и позива системску операцију. Системски дијаграм секвенци скицира се за сваки случај коришћења.

Осим системских дијаграма секвенци, за опис понашања софтверског система користе се системске операције. Системска операција одређена је својим потписом који садржи методе и опционо улазне и/излазне аргументе. За сваку системску операцију прави се уговор који описује шта операција треба да ради, без објашњења како ће то да ради. Уговори се састоје од операција, веза са случајевима коришћења, предуслова који морају бити испуњени пре извршења системске операције и постуслова који морају бити испуњени након извршавања системске операције.

### 2.2.2. Структура софтверског система

Структура софтверског система се описује помоћу концептуалног и релационог модела.

Концептуални модел садржи концептуалне класе и асоцијације међу њима. Концептуалне класе представљају атрибуте софтверског система, што значи да концепти описују структуру софтверског система. Концептуалне класе састоје се од атрибута, који описују особине класе.



Слика 2. Пример концептуалног модела софтверског система



На основу концептуалног модела може се направити релациони модел, који даје основу за пројектовање релационе базе података. Као пример, у наставку је приказан релациони модел направљен на основу концептуалног модела са слике 2.

Admin(AdminID, Email, Lozinka)

Student(StudentID, BrojIndeksa, Ime, Prezime, Email, Lozinka, Telefon)

Predmet(PredmetID, Naziv, Opis, Semestar, Espb, *AdminID*)

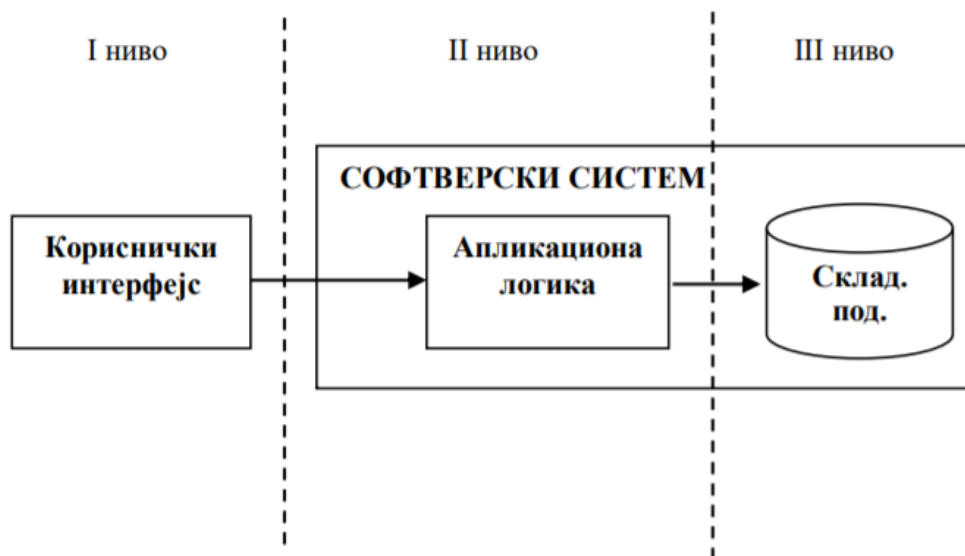
Projekat(PredmetID, RB, Naziv, Opis, BrojPoena, RokZaPredaju)

Plan(StudentID, PredmetID, DatumKreiranja, UkupanBrojStavki, BrojKompletiranihStavki)

StavkaPlana(StudentID, PredmetID, RBStavkePlana, Komentar, Rok, Kompletirana, *PredmetProjektaID*, *RBProjekta*)

### 2.3. Пројектовање

Фаза пројектовања описује физичку структуру и понашање софтверског система, тј. његову архитектуру. У овом раду коришћена је класична тронивојска архитектура која се састоји од корисничког интерфејса, апликационе логике и складишта података, па пројектовање архитектуре софтверског система обухвата пројектовање сваког од поменутих нивоа.



Слика 3. Тронивојска архитектура [1]

На основу тронивојске архитектуре направљени су савремени апликациони сервери који су одговорни да обезбеде сервисе који ће да омогуће реализацију апликационе логике софтверског система. Апликациони сервер састоји се од дела за комуникацију са клијентом, дела за комуникацију са складиштем података и дела који садржи пословну логику.

## **2.4. Имплементација и тестирање**

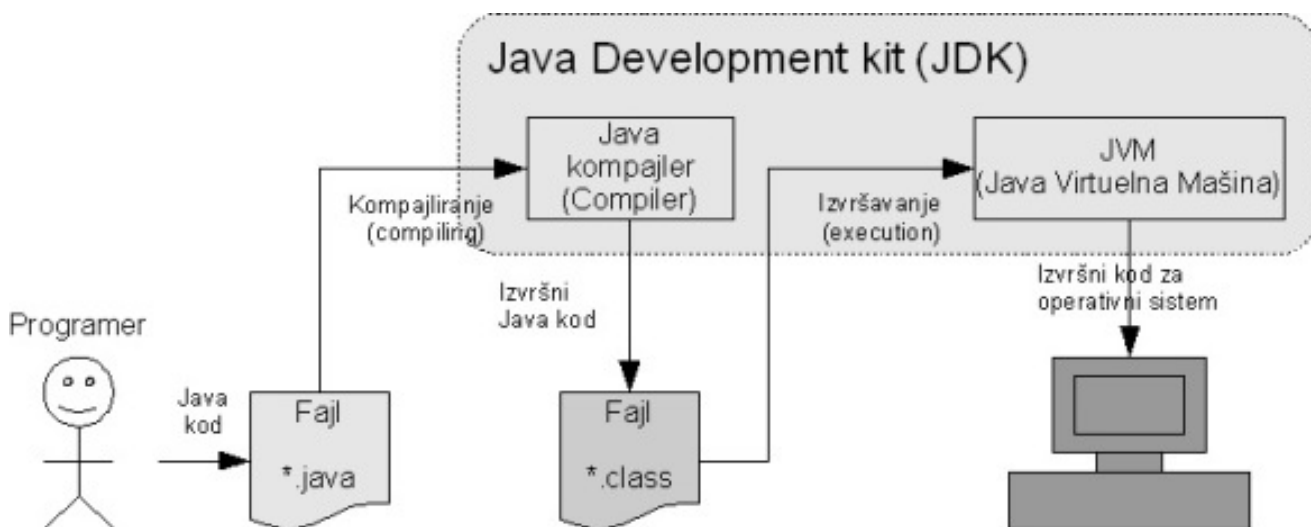
Након фаза прикупљања корисничких захтева, анализе и пројектовања приступа се имплементацији софтверског система. Имплементација је кодирање софтверског система у изабраном програмском језику. У студијском примеру овог рада, коришћене су Јава технологије. Као развојно окружење коришћен је развојни алат NetBeans 8.2. Фаза обједињена са фазом имплементације је фаза тестирања, чији је циљ провера да ли софтверски систем испуњава функционалности дефинисане у фазама који му претходе и отклањање евентуалних грешака.

### 3. Јава, јава технологије

У имплементацији софтверског система, коришћен је програмски језик Јава. Јава је програмски језик опште намене развијен 1995. године од стране инжењера из компаније Sun Microsystems. Јава је постала основ за развој других програмских језика (php, c#) , и задржала је позицију најкоришћенијег програмског језика до данашњег дана. Један од примера где је језик пронашао огромну примену јесте web програмирање, где се Јава користи као имплементациони језик на страни сервера. Такође, Google је изабрао Јаву као имплементациони језик за израду Android апликација, а Јава је веома коришћена и у изради тзв. Desktop апликација, што је случај и у овом раду.

Најзначајнија особина по којој се Јава разликовала од ранијих програмских језика јесте платформска независност, што значи да се једном написан код у Јави може извршавати на било којој платформи, уколико је на њој инсталирана JVM(Јава виртуелна машина). Наиме, изворни Јава код најпре компајлира Javac (главни јавин компајлер који је садржан у JDK(Java Development Kit)) и као резултат генерише бајт-код. Бајт-код се затим извршава од стране Јавине виртуалне машине, и као резултат се добија машински код. Треба имати у виду да без инсталиране Јавине виртуалне машине чија је имплементација различита за сваки оперативни систем није могуће читати бајт-код [3].

На слици којој следи, графички је приказан начин извршавања програма у Јави.



Слика 4. Извршавање програма у јави

## 3.1. Концепти објектно оријентисаног програмирања у Јави

### 3.1.1. Основни појмови

Класа је кориснички дефинисани нацрт или прототип из којег се креирају објекти [3]. Чине је атрибути и методе. Атрибутима се дефинише стање, а методама понашање класе. На слици 5. је приказан пример Јавине класе са припадајућим атрибутима и методама.

```
/**
 *
 * @author Vladimir
 */
public class Zaposleni {

    private String naziv;
    private String radnoMesto;
    private double koeficijent;
    public Zaposleni() {
    }

    public Zaposleni(String naziv, double koeficijent, String radnoMesto) {
        this.naziv = naziv;
        this.koeficijent = koeficijent;
        this.radnoMesto = radnoMesto;
    }

    public double izracunajPlatu(){

        System.out.println("Zaposleni - " + naziv + " ima platu : " + koeficijent * 50000);
        return koeficijent * 50000;
    }
}
```

Слика 5. Пример класе са припадајућим атрибутима и методама

Свака метода одређена је типом који враћа, називом, листом параметара и телом методе [2]. Уколико метода враћа вредност, тада се користи кључна реч “return”. Конкретно на примеру са слике, назив методе је “izracunajPlatu”, нема улазне параметере, тип података који враћа је double, а тело методе садржи логику методе у којој се исписује и враћа вредност плате запосленог.

Објекат представља једно конкретно појављивање (примерак, инстанцу) своје класе. У програмском језику Јава, објекат се инстанцира на следећи начин.

**Zaposleni zaposleni = new Zaposleni();**

Овим кодом се алоцира меморија за нови објекат класе запослени и дефинише се променљива која представља референцу на тај објекат. Приликом инстанцирања, у дати објекат убацују се вредности дефинисане у конструктору класе. Конструктор може бити параметарски и подразумевани. У случају подразумеваног конструктора, вредности свих атрибута објекта дате класе биће постављени на њихове подразумеване вредности зависно од типа података, док у случају параметарског конструктора програмер додељује вредности приликом инстанцирања, на следећи начин.

```
public Zaposleni(String naziv, double koeficijent, String radnoMesto) {  
    this.naziv = naziv;  
    this.koeficijent = koeficijent;  
    this.radnoMesto = radnoMesto;  
}
```

Слика 6. Дефинисање парамететарског конструктора у оквиру класе

```
public static void main(String[] args) {  
  
    Zaposleni parametarski = new Zaposleni("Vladimir", 1, "Junior_programer");  
    Zaposleni podrazumevani = new Zaposleni();  
  
}
```

Слика 7. Инстанцирање објекта коришћењем параметарског и подразумеваног конструктора

### 3.1.2. Наслеђивање

Наслеђивање је важан концепт у оквиру Јава програмског језика који омогућава једној класи да наследи карактеристике (методе и атрибуте) друге класе. Класа чије су особине наслеђене назива се надкласа (superclass), док се класа која наслеђује назива подкласа(subclass). Јава не подржава вишеструко наслеђивање класа. Наслеђивање се врши помоћу кључне речи “extends” на следећи начин.

```
public class Programer extends Zaposleni {  
  
}
```

Слика 8. Наслеђивање

У примеру са слике 8, класа програмер наслеђује све карактеристике класе запослени, уз могућност дефинисања додатних сопствених атрибута и метода . Такође, подкласа може редифинисати одређену методу надкласе помоћу анотације `@Override`.

### 3.1.3. Појам апстракције, интерфејси, апстрактне класе

Апстракција представља процес скривања детаља имплементације, чиме се кориснику приказују само функционалности.

Постоје два начина за постизање апстракције у Јави

- Апстрактна класа
- Интерфејс

Апстрактне класе дефинишу се на сличан начин као и обичне класе, с тим што се испред назива апстрактних класа, као и припадајућих апстрактних метода додаје кључна реч “`abstract`”. Увођење апстрактне класе има смисла уколико постоји потреба за бар једном апстрактном методом. Апстрактна метода нема тело и одговорност реализације те методе се преноси до методе класе која је наследила апстрактну класу.

```
public abstract class ApstraktnaKlasa {  
    public abstract void apstraktnaMetoda();  
}
```

Слика 9. Дефинисање апстрактне класе

На слици која следи, дефинише се класа која наслеђује апстрактну класу и анотацијом `override` реализује њену апстрактну методу. Уколико обична класа наслеђује апстрактну класу, она мора реализовати све њене припадајуће апстрактне методе. У Јави није могуће вишеструко наслеђивање апстрактних класа.

```
public class KlasaNasledjujeApstraktnu extends ApstraktnaKlasa{  
    @Override  
    public void apstraktnaMetoda() {  
        System.out.println("Obicna klasa realizuje metodu apstraktnе klase");  
    }  
}
```

Слика 10. Наслеђивање апстрактне класе

Интерфејс је концепт који раздваја спецификацију метода од њихове имплементације. За разлику од апстрактне класе која може имати апстрактне и обичне методе, интерфејси могу имати само апстрактне методе, чиме се обезбеђује потпуна апстракција. Интерфејс подржава могућност вишеструког наслеђивања, али за разлику од апстрактних класа у оквиру њега није могуће дефинисати атрибуте. Није могуће инстанцирати интерфејс као ни апстрактну класу. Кључна реч у Јави преко које се остварује наслеђивање интерфејса је “implements”.

```
public interface Interfejs {  
  
    public abstract void apstraktnaMetoda();  
  
}
```

Слика 11. Дефинисање интерфејса

```
public class KlasaNasledjujeInterfejs implements Interfejs{  
  
    @Override  
    public void apstraktnaMetoda() {  
        System.out.println("Apstraktna metoda je realizovana!");  
    }  
  
}
```

Слика 12. Наслеђивање интерфејса

### 3.1.4. Изузеци

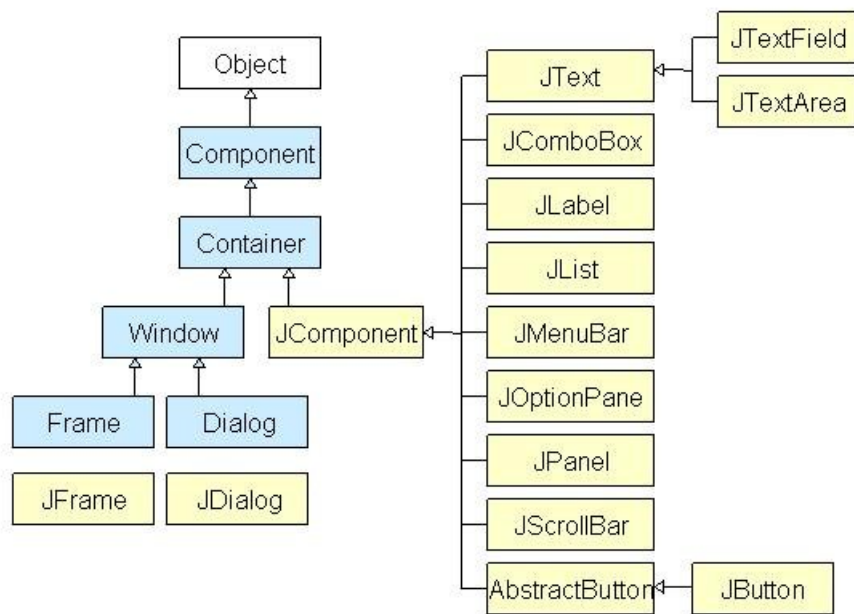
Изузетак је нежељени или неочекивани догађај, који се дешава током извршавања програма и који омета нормалан ток инструкција програма. Када се деси нека грешка у програму, метода у којој се десила грешка баца изузетак који се односи на ту грешку. Изузетак представља објекат Јавине класе “throwable” или објекат неке класе која је наслеђује. Када се деси грешка у оквиру методе такав објекат, који у себи садржи информације о називу и опису изузетка као и о стању програма где се десио тај изузетак, шаље се јавином runtime систему.

Постоје два механизма за управљање изузецима у Јави:

- Коришћењем try-catch блока који служи за покушај обраде методе и хватање изузетка уколико до њега дође.
- Спецификањем методе да баца изузетак коришћењем throws клаузуле, чиме се одговорност за обраду изузетка преноси на методу изнад у хијерархији.

## 3.2. Кориснички интерфејс

Java foundation classes (JFC) представља графички оквир (framework) који омогућава развој преносивих графичких интерфејса у Јави [4] . У студијском примеру овог рада је за израду корисничког интерфејса коришћен Swing, који је део JFC - а. Swing пружа богат скуп widgeta и пакета за прављење софистицираних графичких компоненти за Јава апликације. Скуп свих класа садржаних у оквиру Swing-а могу се видети на слици 13.



Слика 13 Дијаграм хијерархије java swing класа

Имплементирање корисничког интерфејса састоји се од креирања форме а затим и додавања графичких компоненти по избору, у зависности од захтева софтверског система.

Постоји више начина за имплементацију форми у Јави коришћењем Swing-а

- непосредним креирањем објекта класе JFrame, JDialog.
- наслеђивањем класе JFrame, JDialog.

Када је форма креирана на њу се додају графичке компоненте чија је улога прихватање и приказ података. Најкоришћеније графичке компоненте су текстуална поља и дугмићи, али поред њих се користе падајући мени, radiobox, panel итд.



### 3.2.1. Догађаји

Догађаји се генеришу као резултат интеракције корисника са компонентама графичког корисничког интерфејса. На пример, клик на дугме, померање миша, унос карактера преко тастатуре су активности које изазивају догађај.

Догађај може да иницира нека компонента графичког интерфејса (на пример JButton) и за ту компоненту се онда каже да представља **извор догађаја**. Кликком миша на дугме креира се објекат који представља и идентификује тај догађај. Објекат догађај (који је у случају клика мишем на дугме објекат класе `ActionEvent`) садржи информације о извору и типу догађаја и прослеђује се до објекта **слушаоца догађаја**. Прослеђивањем објекта догађаја позива се метода у оквиру слушаоца догађаја. Да би се објекат догађај пренео до објекта слушаоца догађаја најпре је неопходно да се повеже извор догађаја и слушалац догађаја. Следи пример руковања догађајем из студијског примера.

```
private void addActionListenerLoginStudent() {
    frmLogin.loginStudentAddActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent actionEvent) {
            loginUser(actionEvent);
        }

        private void loginUser(ActionEvent actionEvent) {
            resetForm();
            try {

                String username = frmLogin.getTxtUserStudent().getText();
                String password = frmLogin.getTxtPassStudent().getText();

                validateForm(username, password);

                //User user = Controller.getInstance().login(username, password);
                Student student = Communication.getInstance().login(username, password);
                JOptionPane.showMessageDialog(
                    frmLogin,
                    "Welcome " + student.getFirstName() + " " + student.getLastName(),
                    "Login", JOptionPane.INFORMATION_MESSAGE
                );
                frmLogin.dispose();
                MainCoordinator.getInstance().addParam(Constants.CURRENT_STUDENT, student);
                MainCoordinator.getInstance().openMainStudent();
            } catch (Exception e) {
                JOptionPane.showMessageDialog(frmLogin, e.getMessage(), "Login error", JOptionPane.ERROR_MESSAGE);
            }
        }
    });
}
```

Слика 14. Инстанцирање објекта слушаоца догађаја

На слици 14. приказана је метода која се налази у оквиру `LoginControllera`. `LoginController` задужен је за управљање `Login` формом, па самим тим чува референцу на форму као свој атрибут. У оквиру методе инстанцира се објекат слушаоца догађаја и дефинише припадајућа метода слушаоца догађаја која се позива приликом пријема

објекта догађаја (`actionPerformed`). Остало је да се повеже извор догађаја (дугме за логовање) са слушаоцем догађаја. `LoginController` преко референце на форму позива њену методу `loginStudentAddActionListener`. Као улазни параметер прослеђена је референца на новокреирани објекат слушаоца догађаја. Улога ове методе дефинисане у оквиру форме јесте да повеже дугме за логовање са слушаоцем догађаја и њена имплементација изгледа овако:

```
public void loginStudentAddActionListener(ActionListener actionListener) {  
    btnLoginStudent.addActionListener(actionListener);  
}
```

Слика 15. Повезивање графичке компоненте са слушаоцем догађаја

### 3.3. Мрежно програмирање

#### 3.3.1. Адреса рачунара, URL адреса

Адреса рачунара је скуп бројева који јединствено одређују рачунар у оквиру глобалне мреже (Интернета). Други назив који се често користи за адресу рачунара је IP адреса, јер се приликом комуникације између рачунара на интернету користи TCP/IP протокол. Већина IP адреса састоји се од четири троцифрена броја чији је распон од 0 до 255 и који су међусобно одвојени тачкама. Пример једне адресе је 192.168.1.42. Поред нумеричке адресе рачунара користи се и симболичка адреса рачунара, која се представља помоћу знакова. Једна симболичка адреса може бити везана за једну или више нумеричких адреса рачунара. Симболичка адреса састоји се из матичног имена (host name) рачунара и имена домена коме припада матични рачунар. Сервис који служи за повезивање симболичких и интернет адреса је DNS(Domain Naming Service).

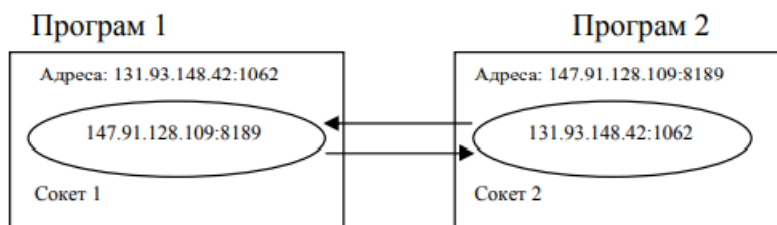
URL адреса представља путању до јединственог ресурса на интернету. Ресурси којима се приступа могу бити HTML странице, CSS документи, слике итд.

URL адреса се састоји од четири дела:

- Протокол - одвојен од остатка адресе (најкоришћенији протокол је http).
- Адреса рачунара – може да буде симболичка или нумеричка. Испред адресе се ставља '//', а на крај '/' уколико нема порт, односно ':' уколико има.
- Број порта - опциони део.
- Путања до датотеке (ресурса).

### 3.3.2. Сокети

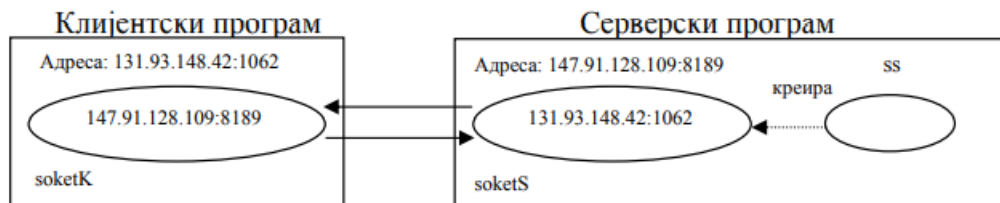
Сокети се користе за комуникацију између програма који раде на различитим JRE (Java Runtime Environment). Када се повезују два програма, за сваки од њих се генерише по један сокет, при чему сваки од сокета садржи референцу на сокет другог програма. Адреса сокета састоји се од адресе рачунара на коме се налази програм који је генерисао сокет и број порта који је генерисан помоћу сокета.



Слика 16. Повезивање два програма помоћу сокета [2]

Уколико је реч о клијент/серверским апликацијама, са једне стране везе је серверски сокет, одређен портом и адресом рачунара на ком је покренут серверски програм, који ослушкује и прихвата клијенте. Приликом прихватања клијента, креира се нови објекат класе сокет који формира са клијентским сокетом обострану конекцију.

На клијентској страни се дефинише клијентски сокет. Приликом иницијализације клијентског сокета, као улазни параметар се убацује адреса серверског сокета (Адреса рачунара и порт). На овај начин, повезивањем сокета оба програма, формирана је конекција између клијентског и серверског дела софтверског система, при чему сваки од сокета садржи референцу на сокет другог програма.



Слика 17. Повезивање клијентског и серверског програма помоћу сокета [2]

Сада када су клијентски и серверски део система повезани, потребно је развити механизам за размену података између сокета, тако да исти могу да се обраде преко стандардних улазно-излазних уређаја. Сокети се повезују са улазно-излазним објектима на следећи начин:

```
BufferedReader in = new BufferedReader(new InputStreamReader(X.getInputStream()));
PrintWriter out = new PrintWriter(X.getOutputStream(),true);
```

Где је `X` променљива која представља клијентски или серверски сокет за који се врши повезивање. Сада је могуће писање и пријем порука помоћу метода:

`out.println("Slanje poruke");` - слање поруке

`String message = in.readLine();` - пријем поруке и смештање у променљиву

```
try {
    ServerSocket ss = new ServerSocket(9999);
    System.out.println("Server je pokrenut i ceka na klijente!");

    while(true) {
        System.out.println("Server prihvata klijente");
        Socket clientSocket = ss.accept();

        PrintWriter out = new PrintWriter(clientSocket.getOutputStream());
        BufferedReader in = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));

        // Kada prihvati klijenta, pokrece se nova iteracija while petlje koja je beskona cime je server implementiran da radi sa vise klijenata
    }
} catch (IOException ex) {
    Logger.getLogger(ZavrzniServer.class.getName()).log(Level.SEVERE, null, ex);
}
```

Слика 18. Имплементација сокета на серверској страни

```
try {
    Socket clientSocket = new Socket("localhost",9999);

    PrintWriter out = new PrintWriter(clientSocket.getOutputStream());
    BufferedReader in = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));

} catch (IOException ex) {
    Logger.getLogger(ZavrzniKlijent.class.getName()).log(Level.SEVERE, null, ex);
}
```

Слика 19. Имплементација сокета на клијентској страни

### 3.4. Нити

Multithreading је важна одлика Јава програмског језика која омогућава истовремено извршавање два или више делова програма, чиме се постиже максимална искоришћеност процесора рачунара. Сваки део таквог програма назива се нит, а свака нит дефинише засебну путању извршења. Сваки Јава програм увек има најмање једну нит, која се зове главна нит. Главна нит се такође назива parent (родитељска) нит и из ње се генеришу остале нити програма.

Постоје два начина за генерисање нити у јави:

- 1) Наслеђивањем класе Thread
- 2) Имплементирањем интерфејса Runnable

```
public class NitThread extends Thread{

    @Override
    public void run() {
        System.out.println(" 1) Pokrenuta je nit generisana nasledjivanjem klase Thread");
    }

}
```

Слика 20. Дефинисање нити наслеђивањем класе Thread

```
public class NitRunnable implements Runnable {

    @Override
    public void run() {
        System.out.println(" 2) Pokrenuta je nit generisana implementiranjem interfejsa Runnable");
    }

}
```

Слика 21. Дефинисање нити имплементирањем интерфејса Runnable

```
public class Main {

    public static void main(String[] args) {
        System.out.println("Glavna nit programa je pokrenuta!");

        Thread prvaNit = new NitThread();
        Thread drugaNit = new Thread(new NitRunnable());

        prvaNit.start();
        drugaNit.start();
    }

}
```

Output x

Run (BookingApi) x BazaDiplomski (run) x

run:  
Glavna nit programa je pokrenuta!  
1) Pokrenuta je nit generisana nasledjivanjem klase Thread  
2) Pokrenuta je nit generisana implementiranjem interfejsa Runnable  
BUILD SUCCESSFUL (total time: 0 seconds)

Слика 22. Покретање нити

### 3.5. Рад са базом података

JDBC је API (application programming interface) који се користи у Јава програмирању за интеракцију са базом података. Да би се програм успешно повезао са системом за управљање базом податка (СУБП) коришћењем JDBC-а, у пројекат је неопходно убацити JDBC Driver, чија је имплементација различита за сваки СУБП. Идеја је да се један Јавин код за приступ бази података коришћењем JDBC-а може користити за различите СУБП, с тим што се мења Driver у зависности од тога који се СУБП користи.

Кораци при повезавању Јава програма и система за управљање базом података су следећи:

- укључивање у Јава програм JDBC API-а
- учитавање Driver-а у Јава програм
- успостављање конекције (везе) између Јава програма и базе података изабраног СУБП

Први корак у оквиру извршавања операције над претходног повезаним СУБП је прављење објекта класе Statement. Објекат класе Statement се добија као повратна вредност методе createStatement позваном над објектом класе Connection. Над креираним објектом класе Statement позива се метода executeQuery која као улазни параметар прима жељени упит над базом података. Упит се прослеђује као променљива са типом податка String. У примеру SELECT наредбе, као резултат методе executeQuery добија се објекат класе ResultSet који садржи све слоге табеле над којом се врши операција.

```
try {
    Class.forName("com.mysql.jdbc.Driver");

    String url = "jdbc:mysql://localhost:3306/diplomski_primer";
    String name = "root";
    String password = "";

    Connection connection = DriverManager.getConnection(url,name,password);
    Statement statement = connection.createStatement();

    String upit = "SELECT * FROM GRAD";
    ResultSet rs = statement.executeQuery(upit);

    while(rs.next()){

        System.out.println("Grad : " + rs.getString("naziv") + " ima PTT broj " + rs.getLong("ptt"));

    }

}
catch (SQLException ex) {
    System.out.println("Desila se sql greska!");
}
catch (ClassNotFoundException ex) {
    System.out.println("Neuspesno ucitan driver!");
}
```

Слика 21. Пример извршавања SELECT наредбе над табелом Град

## 4. Студијски пример

### 4.1. Прикупљање корисничких захтева

#### 4.1.1. Вербални опис

Потребно је развити софтверски систем за праћење евиденције о предметима и пројектима који представљају обавезе на сваком предмету. У оквиру једног предмета дефинише се произвољан број пројеката.

Корисници система су админ и студент:

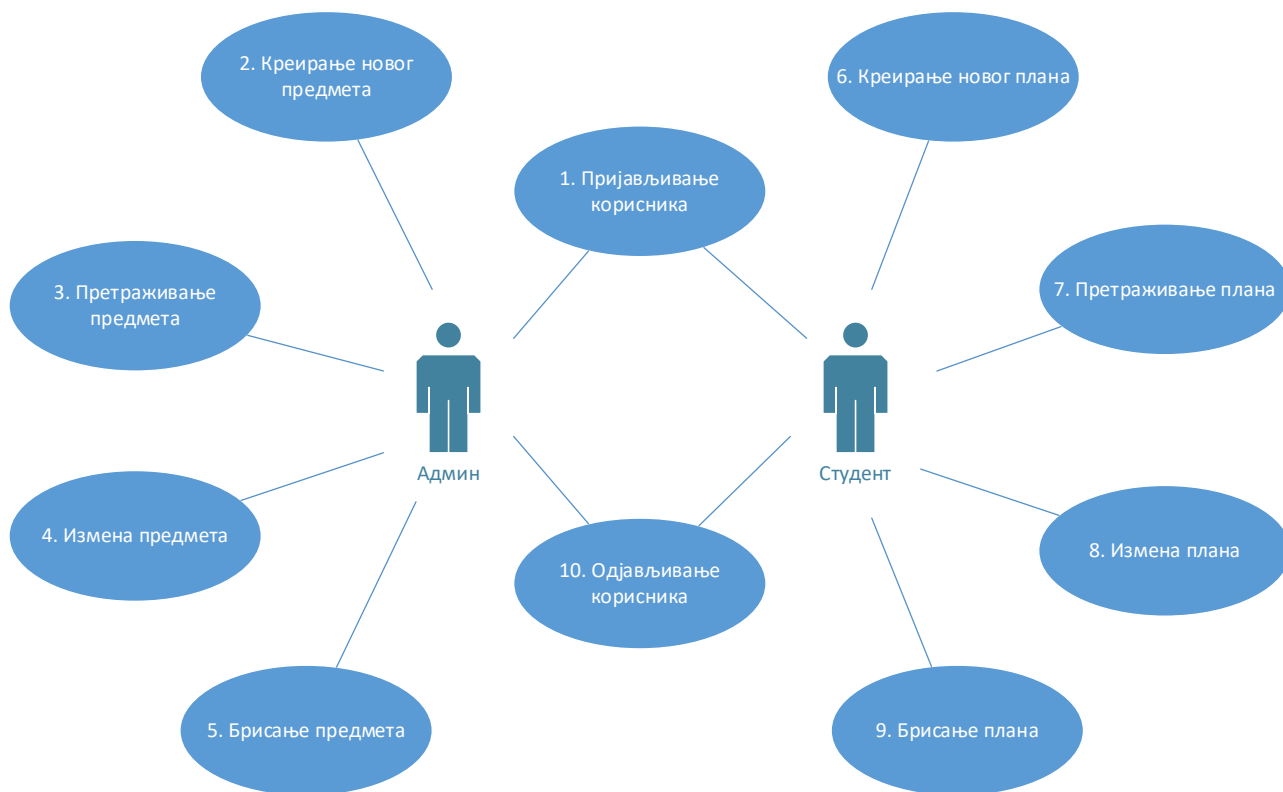
- (1) Систем треба да омогући админу да креира, мења, брише и претражује предмете уз могућност дефинисања и ажурирања њихових припадајућих пројеката.
- (2) Са друге стране, студенту систем треба да пружи могућност креирања плана за сваки предмет, као и њихову претрагу измену и брисање. Приликом рада са планом, студент дефинише произвољан број ставки плана који представљају његове интерне задатке. Свака ставка плана везује се за један пројекат а изабрани пројекат мора бити у оквиру предмета за који је план креиран.

Систем такође треба да омогући пријаву и одјаву за оба типа корисника.

#### 4.1.2 Модел случаја коришћења

У овом софтверском систему, идентификовано је 10 случајева коришћења:

1. Пријављивање корисника на систем
2. Креирање новог предмета (Сложен случај коришћења)
3. Претраживање предмета
4. Измена предмета (Сложен случај коришћења)
5. Брисање предмета
6. Креирање новог плана (Сложен случај коришћења)
7. Претраживање плана
8. Измена плана (Сложен случај коришћења)
9. Брисање плана
10. Одјављивање корисника са система



Слика 23. Случајеви коришћења



## **СК1: Случај коришћења – Пријављивање корисника**

### **Назив СК**

Пријављивање на систем

### **Актори СК:**

Корисник

### **Учесници СК:**

Корисник и систем

**Предуслов:** Систем је укључен. Систем приказује форму за пријављивање на систем.

### **Основни сценарио**

1. Корисник уноси корисничко име и шифру. (АПУСО)
2. Корисник контролише да ли је коректно унео податке. (АНСО)
3. Корисник позива систем да изврши пријаву. (АПСО)
4. Систем пријављује корисника на систем. (СО)
5. Систем приказује кориснику поруку: “Пријављени сте на систем”. (ИА)

### **Алтернативна сценарија**

- 5.1 Уколико систем не може да пријави корисника, систем приказује кориснику поруку: “Неуспешна пријава на систем“ (ИА)

## **СК2: Случај коришћења – Креирање новог предмета (Сложен случај коришћења)**

### **Назив СК**

Креирање новог предмета

### **Актори СК**

Админ

### **Учесници СК**

Админ и систем (програм)

**Предуслов:** Систем је укључен и админ је улогован под својом шифром. Систем приказује форму за рад са предметом.

### **Основни сценарио СК**

- 1.Админ уноси податке у нови предмет. (АПУСО)
- 2.Админ контролише да ли је коректно унео податке у нови предмет. (АНСО)
- 3.Админ позива систем креира нови предмет. (АПСО)
- 4.Систем креира нови предмет. (СО)
- 5.Систем приказује админу запамћени предмет и поруку: “Систем је креирао нови предмет “. (ИА)

### **Алтернативна сценарија**

- 5.1 Уколико систем не може да креира предмет он приказује админу поруку “Систем не може да креира предмет”. (ИА)

### **СКЗ: Случај коришћења – Претраживање предмета**

#### **Назив СК**

Претраживање предмета

#### **Актори СК**

Админ

#### **Учесници СК**

Админ и систем (програм)

**Предуслов:** Систем је укључен и админ је улогован под својом шифром. Систем приказује форму за рад са предметом. Учитана је листа предмета.

#### **Основни сценарио СК**

- 1.Админ уноси критеријум по којем претражује предмете. (АПУСО)
- 2.Админ позива систем да нађе предмете по задатом критеријуму. (АПСО)
- 3.Систем тражи предмете по задатом критеријуму. (СО)
- 4.Систем обавештава админа о успешно извршеној претрази одговарајућом поруком и приказује пронађене предмете.(ИА)
- 5.Админ бира предмет који жели да прегледа.(АПУСО)
- 6.Админ позива систем да учита податке о одабраном предмету.(АПСО)
- 7.Систем учитава податке о одабраном предмету.(СО)
- 8.Систем обавештава админа о успешном учитавању података о предмету поруком “Одабрани предмет је приказан !” и приказује податке о одабраном предмету.(ИА)

#### **Алтернативна сценарија**

- 4.1 Уколико систем не може да нађе предмете он приказује админу поруку: “Систем не може да нађе предмете по задатом критеријуму“. Прекида се извршавање сценарија. (ИА)
- 8.1 Уколико систем не може да учита изабрани предмет он приказује админу поруку „Систем не може да учита изабрани предмет.“ Прекида се извршење сценарија.(ИА)

#### **СК4: Случај коришћења – Измена предмета(Сложен случај коришћења)**

##### **Назив СК**

Промена предмета

##### **Актори СК**

Админ

##### **Учесници СК**

Админ и систем (програм)

**Предуслов:** Систем је укључен и админ је улогован под својом шифром. Систем приказује форму за рад са предметом. Учитана је листа предмета.

##### **Основни сценарио СК**

- 1.Админ уноси критеријум по којем претражује предмете. (АПУСО)
- 2.Админ позива систем да нађе предмете по задатом критеријуму. (АПСО)
- 3.Систем тражи предмете по задатом критеријуму. (СО)
- 4.Систем обавештава админа о успешно извршеној претрази одговарајућом поруком и приказује пронађене предмете.(ИА)
- 5.Админ бира предмет који жели да прегледа.(АПУСО)
- 6.Админ позива систем да учита податке о одабраном предмету.(АПСО)
- 7.Систем учитава податке о одабраном предмету.(СО)
- 8.Систем обавештава админа о успешном учитавању података о предмету поруком “Одабрани предмет је приказан !” и приказује податке о одабраном предмету.(ИА)
- 9.Админ уноси (мења) податке о предмету. (АПУСО)
- 10.Админ контролише да ли је коректно унео податке о предмету. (АНСО)
- 11.Админ позива систем да запамти податке о предмету. (АПСО)
- 12.Систем памти податке о предмету. (СО)
- 13.Систем приказује админу запамћени предмет и поруку: “Систем је запамтио предмет.” (ИА)

##### **Алтернативна сценарија**

- 4.1 Уколико систем не може да нађе предмете он приказује админу поруку: “Систем не може да нађе предмете по задатом критеријуму”. Прекида се извршење сценарија. (ИА)
- 8.1 Уколико систем не може да учита изабрани предмет он приказује админу поруку „Систем не може да учита изабрани предмет.“ Прекида се извршење сценарија.(ИА)
- 13.1 Уколико систем не може да запамти податке о предмету он приказује админу поруку “Систем не може да запамти предмет”. Прекида се извршење сценарија. (ИА)

## **СК5: Случај коришћења – Брисање предмета**

### **Назив СК**

Брисање предмета

### **Актори СК**

Админ

### **Учесници СК**

Админ и систем (програм)

**Предуслов:** Систем је укључен и админ је улогован под својом шифром. Систем приказује форму за рад са предметом. Учитана је листа предмета.

### **Основни сценарио СК**

- 1.Админ уноси критеријум по којем претражује предмете. (АПУСО)
- 2.Админ позива систем да нађе предмете по задатом критеријуму. (АПСО)
- 3.Систем тражи предмете по задатом критеријуму. (СО)
- 4.Систем обавештава админа о успешно извршеној претрази одговарајућом поруком и приказује пронађене предмете.(ИА)
- 5.Админ бира предмет који жели да прегледа.(АПУСО)
- 6.Админ позива систем да учита податке о одабраном предмету.(АПСО)
- 7.Систем учитава податке о одабраном предмету.(СО)
- 8.Систем обавештава админа о успешном учитавању података о предмету поруком “Одабрани предмет је приказан !” и приказује податке о одабраном предмету.(ИА)
- 9.Админ позива систем да обрише предмет. (АПСО)
- 10.Систем брише предмет. (СО)
- 11.Систем приказује админу поруку: “Систем је обрисао предмет.” (ИА)

### **Алтернативна сценарија**

- 4.1 Уколико систем не може да нађе предмете он приказује админу поруку: “Систем не може да нађе предмете по задатом критеријуму”. Прекида се извршење сценарија. (ИА)
- 8.1 Уколико систем не може да учита изабрани предмет он приказује админу поруку „Систем не може да учита изабрани предмет.“ Прекида се извршење сценарија.(ИА)
- 11.1 Уколико систем не може да обрише предмет он приказује админу поруку “Систем не може да обрише предмет”. (ИА)

## **СК6: Случај коришћења – Креирање новог плана(Сложен случај коришћења)**

### **Назив СК**

Креирање новог предмета

### **Актори СК**

Студент

### **Учесници СК**

Студент и систем (програм)

**Предуслов:** Систем је укључен и студент је улогован под својом шифром. Систем приказује форму за рад са планом. Учитана је листа предмета.

### **Основни сценарио СК**

- 1.Студент уноси податке у нови план. (АПУСО)
- 2.Студент контролише да ли је коректно унео податке у нови план. (АНСО)
- 3.Студент позива систем да креира нови план. (АПСО)
- 4.Систем креира нови план. (СО)
- 5.Систем приказује студенту запамћени предмет и поруку: “Систем је креирао нови план“. (ИА)

### **Алтернативна сценарија**

- 5.1 Уколико систем не може да креира нови план он приказује студенту поруку “Систем не може да запамти план“. (ИА)

## **СК7: Случај коришћења – Претраживање плана**

### **Назив СК**

Претраживање предмета

### **Актори СК**

Студент

### **Учесници СК**

Студент и систем (програм)

**Предуслов:** Систем је укључен и студент је улогован под својом шифром. Систем приказује форму за рад са планом. Учитана је листа планова.

### **Основни сценарио СК**

- 1.Студент уноси критеријум по којем претражује планове. (АПУСО)
- 2.Студент позива систем да нађе планове по задатом критеријуму. (АПСО)
- 3.Систем тражи планове по задатом критеријуму. (СО)
- 4.Систем обавештава студента о успешно извршеној претрази одговарајућом поруком и приказује пронађене планове.(ИА)
- 5.Студент бира план који жели да прегледа.(АПУСО)
- 6.Студент позива систем да учита податке о одабраном плану.(АПСО)
- 7.Систем учитава податке о одабраном плану.(СО)
- 8.Систем обавештава студента о успешном учитавању података о предмету поруком “Одабрани план је приказан !” и приказује податке о одабраном плану.(ИА)

### **Алтернативна сценарија**

- 4.1 Уколико систем не може да нађе планове он приказује студенту поруку: “Систем не може да нађе планове по задатом критеријуму“. Прекида се извршавање сценарија.(ИА)
- 8.1 Уколико систем не може да учита изабрани план он приказује студенту поруку „Систем не може да учита изабрани план.“ Прекида се извршење сценарија.(ИА)

## **СК8: Случај коришћења – Измена плана (Сложен случај коришћења)**

### **Назив СК**

Промена предмета

### **Актори СК**

Студент

### **Учесници СК**

Студент и систем (програм)

**Предуслов:** Систем је укључен и студент је улогован под својом шифром. Систем приказује форму за рад са планом. Учитана је листа планова. Учитана је листа предмета.

### **Основни сценарио СК**

1. Студент уноси критеријум по којем претражује планове. (АПУСО)
2. Студент позива систем да нађе планове по задатом критеријуму. (АПСО)
3. Систем тражи планове по задатом критеријуму. (СО)
4. Систем обавештава студента о успешно извршеној претрази одговарајућом поруком и приказује пронађене планове. (ИА)
5. Студент бира план који жели да прегледа. (АПУСО)
6. Студент позива систем да учита податке о одабраном плану. (АПСО)
7. Систем учитава податке о одабраном плану. (СО)
8. Систем обавештава студента о успешном учитавању података о плану поруком “Одабрани план је приказан !” и приказује податке о одабраном плану. (ИА)
9. Студент уноси (мења) податке о плану. (АПУСО)
10. Студент контролише да ли је коректно унео податке о плану. (АНСО)
11. Студент позива систем да запамти податке о плану. (АПСО)
12. Систем памти податке о плану. (СО)
13. Систем приказује студенту запамћени план и поруку: “Систем је запамтио план.” (ИА)

### **Алтернативна сценарија**

- 4.1 Уколико систем не може да нађе планове он приказује студенту поруку: “Систем не може да нађе планове по задатом критеријуму”. Прекида се извршење сценарија. (ИА)
- 8.1 Уколико систем не може да учита изабрани план он приказује студенту поруку „Систем не може да учита изабрани план.“ Прекида се извршење сценарија. (ИА)
- 13.1 Уколико систем не може да запамти податке о плану он приказује студенту поруку “Систем не може да запамти план”. Прекида се извршење сценарија. (ИА)



## **СК9: Случај коришћења – Брисање плана**

### **Назив СК**

Брисање предмета

### **Актори СК**

Студент

### **Учесници СК**

Студент и систем (програм)

**Предуслов:** Систем је укључен и студент је улогован под својом шифром. Систем приказује форму за рад са планом. Учитана је листа планова.

### **Основни сценарио СК**

- 1.Студент уноси критеријум по којем претражује планове. (АПУСО)
- 2.Студент позива систем да нађе планове по задатом критеријуму. (АПСО)
- 3.Систем тражи планове по задатом критеријуму. (СО)
- 4.Систем обавештава студента о успешно извршеној претрази одговарајућом поруком и приказује пронађене планове.(ИА)
- 5.Студент бира план који жели да прегледа.(АПУСО)
- 6.Студент позива систем да учита податке о одабраном плану.(АПСО)
- 7.Систем учитава податке о одабраном плану.(СО)
- 8.Систем обавештава студента о успешном учитавању података о плану поруком “Одабрани план је приказан !“ и приказује податке о одабраном плану.(ИА)
- 9.Студент позива систем да обрише план. (АПСО)
- 10.Систем брише план. (СО)
- 11.Систем приказује кориснику поруку: “Систем је обрисао план.” (ИА)

### **Алтернативна сценарија**

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

8.1 Уколико систем не може да учита изабрани план он приказује кориснику поруку „Систем не може да учита изабрани план.“ Прекида се извршење сценарија.(ИА)

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

## **СК10: Случај коришћења – Одјављивање корисника**

### **Назив СК**

Одјављивање са система

### **Актори СК:**

Корисник

### **Учесници СК:**

Корисник и систем

**Предуслов:** Систем је укључен. Систем приказује форму за одјављивање са система.

### **Основни сценарио**

1. Корисник позива систем да изврши одјаву са система. (АПСО)
2. Систем одјављује корисника са система. (СО)
3. Систем приказује кориснику поруку: “Одјављени сте са система”. (ИА)

### **Алтернативна сценарија**

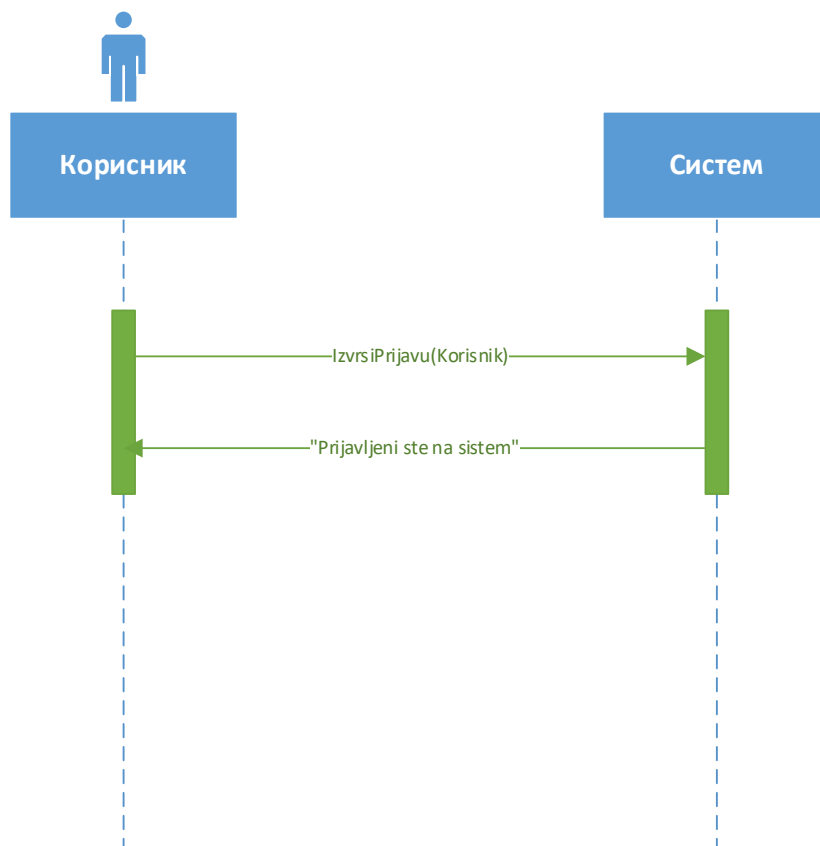
- 3.1 Уколико систем не може да одјави корисника, систем приказује кориснику поруку: “Неуспешна одјава са система“ (ИА)

## 4.2. Анализа

### 4.2.1. Системски дијаграми секвенци

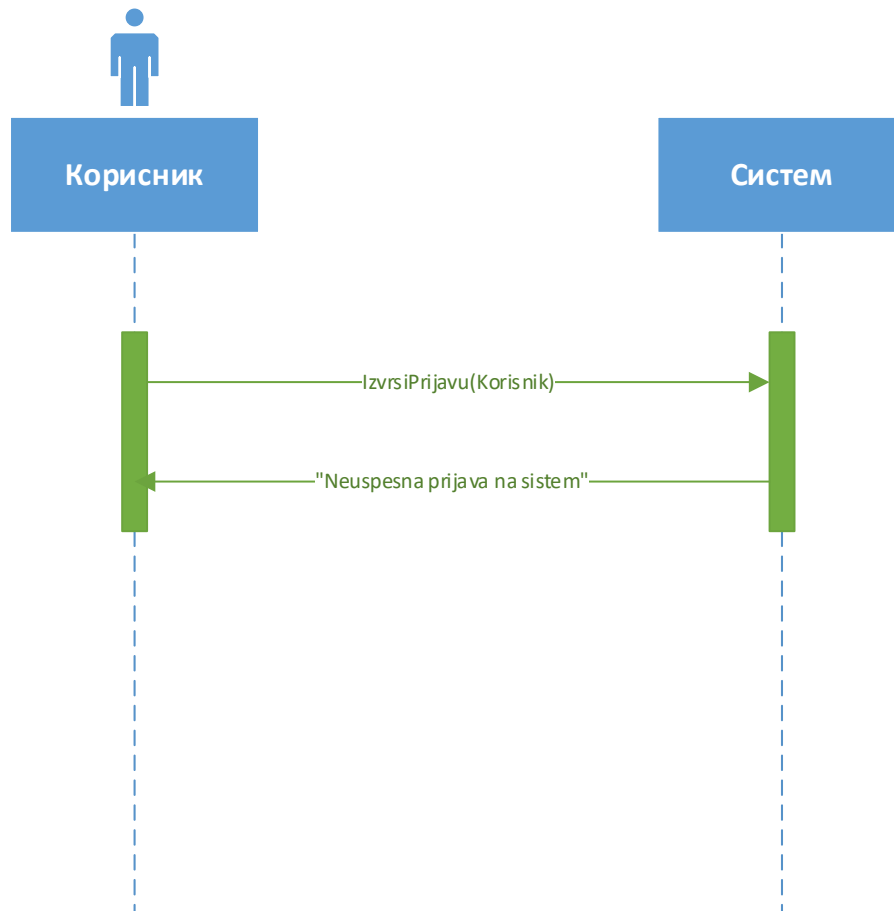
#### ДС1:Дијаграм секвенци случаја коришћења-Пријављивање корисника

- 1.Корисник позива систем да изврши пријаву. (АПСО)
- 2.Систем приказује кориснику поруку: “Пријављени сте на систем”. (ИА)



## Алтернативна сценарија

2.1 Уколико систем не може да пријави корисника, систем приказује кориснику поруку:  
“Неуспешна пријава на систем“ (ИА)

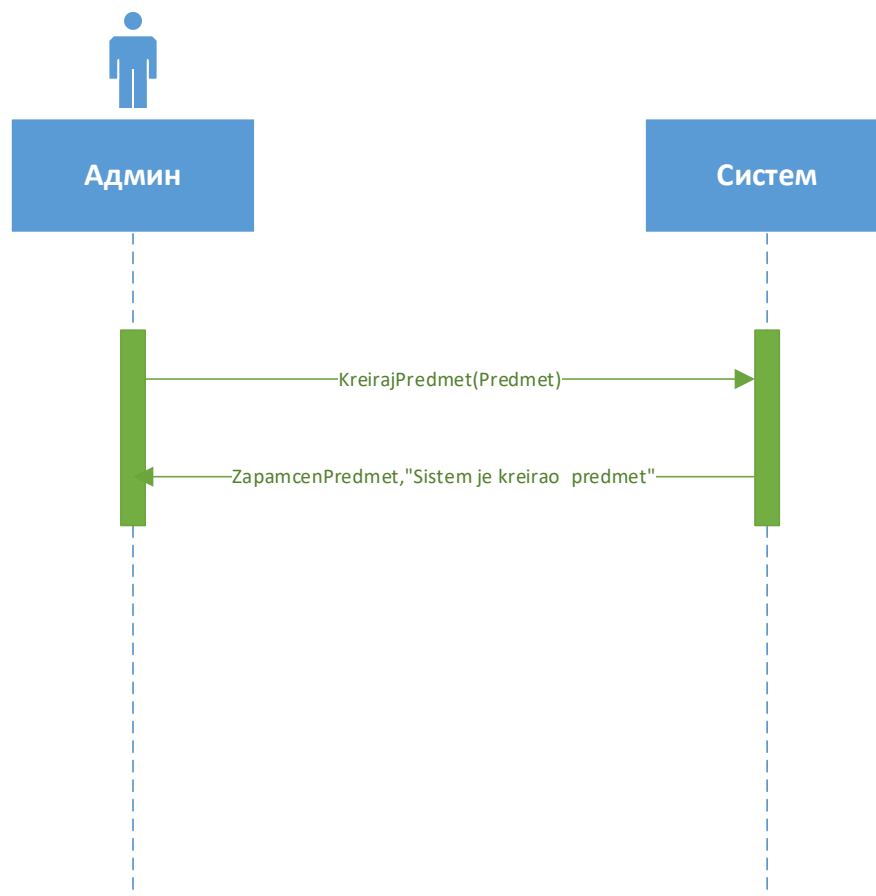


Са наведених секвенцијалних дијаграма уочава се једна системска операција:

1)Signal **IzvrsiPrijavu(Korisnik)**

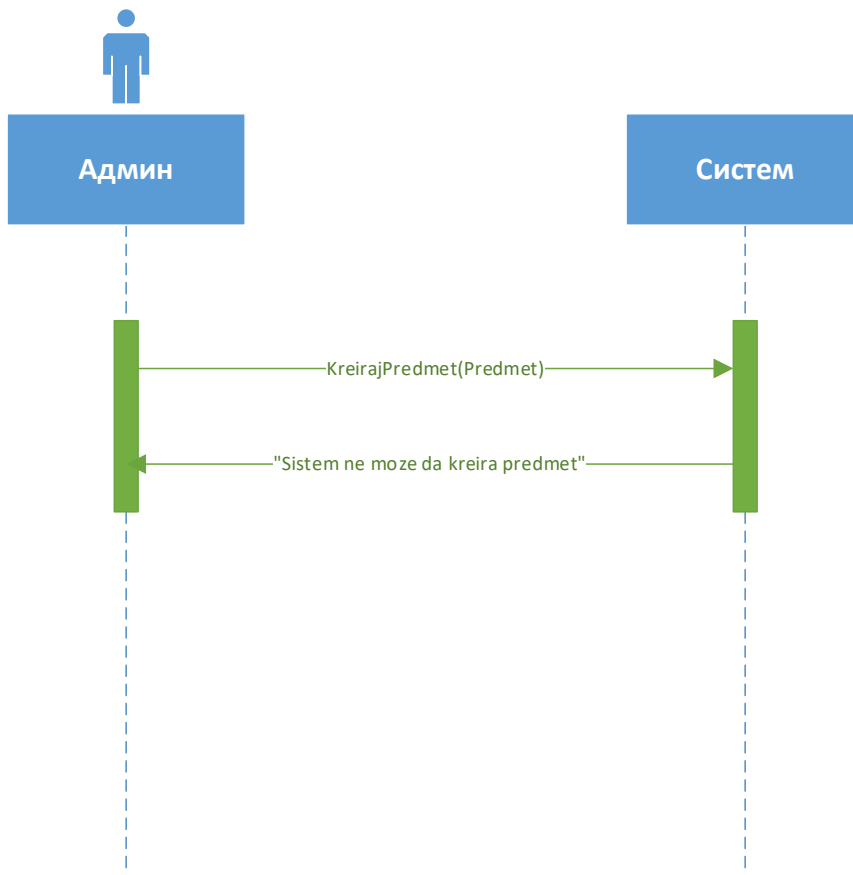
## ДС2:Дијаграм секвенци случаја коришћења-Креирање новог предмета

- 1.Админ позива систем да креира нови предмет. (АПСО)
- 2.Систем приказује админу креирани предмет и поруку: “Систем је креирао предмет”. (ИА)



## Алтернативна сценарија

2.1 Уколико систем не може да запамти податке о предмету он приказује админу поруку “Систем не може да креира предмет”. (ИА)

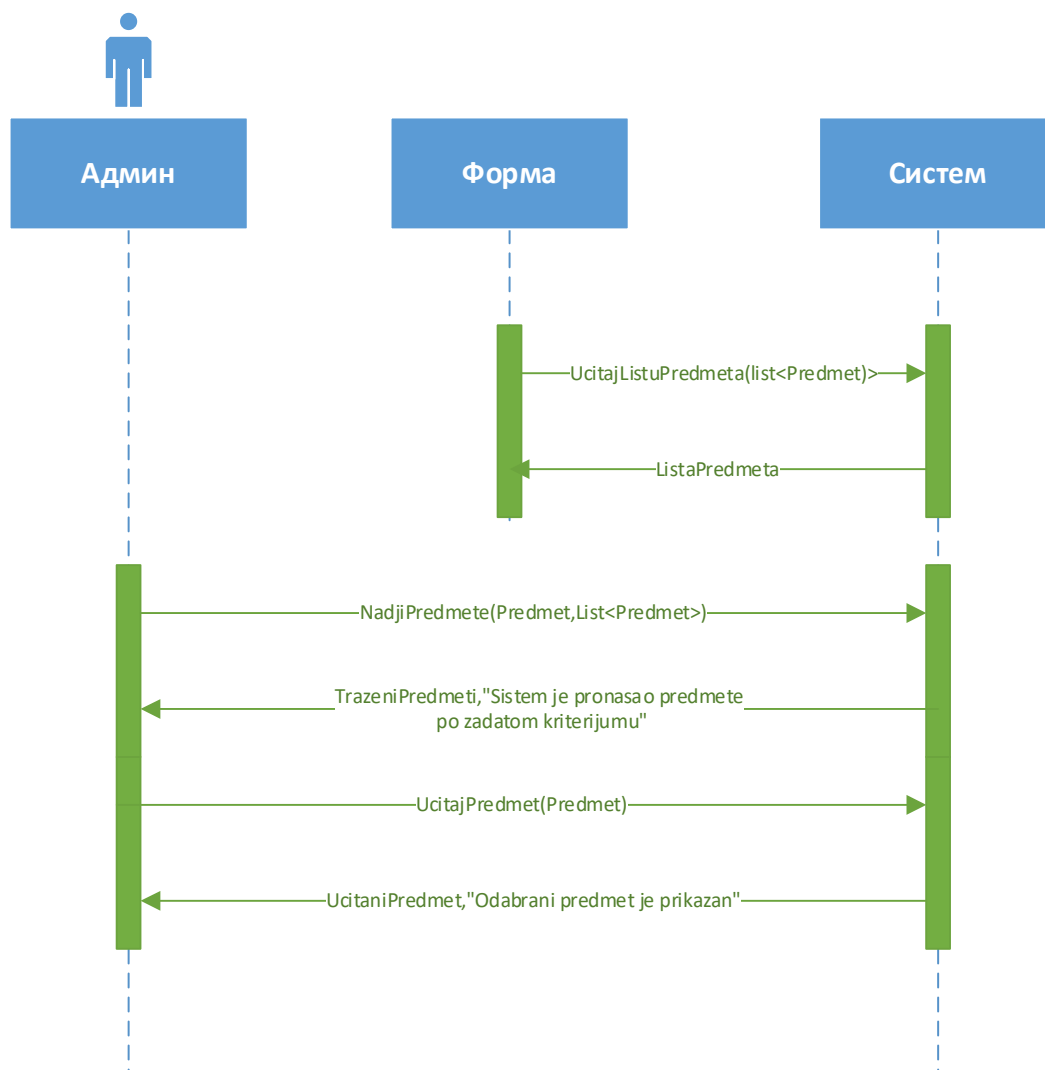


Са наведених секвенцијалних дијаграма уочава се једна системска операција:

1) Signal **KreirajPredmet(Predmet)**

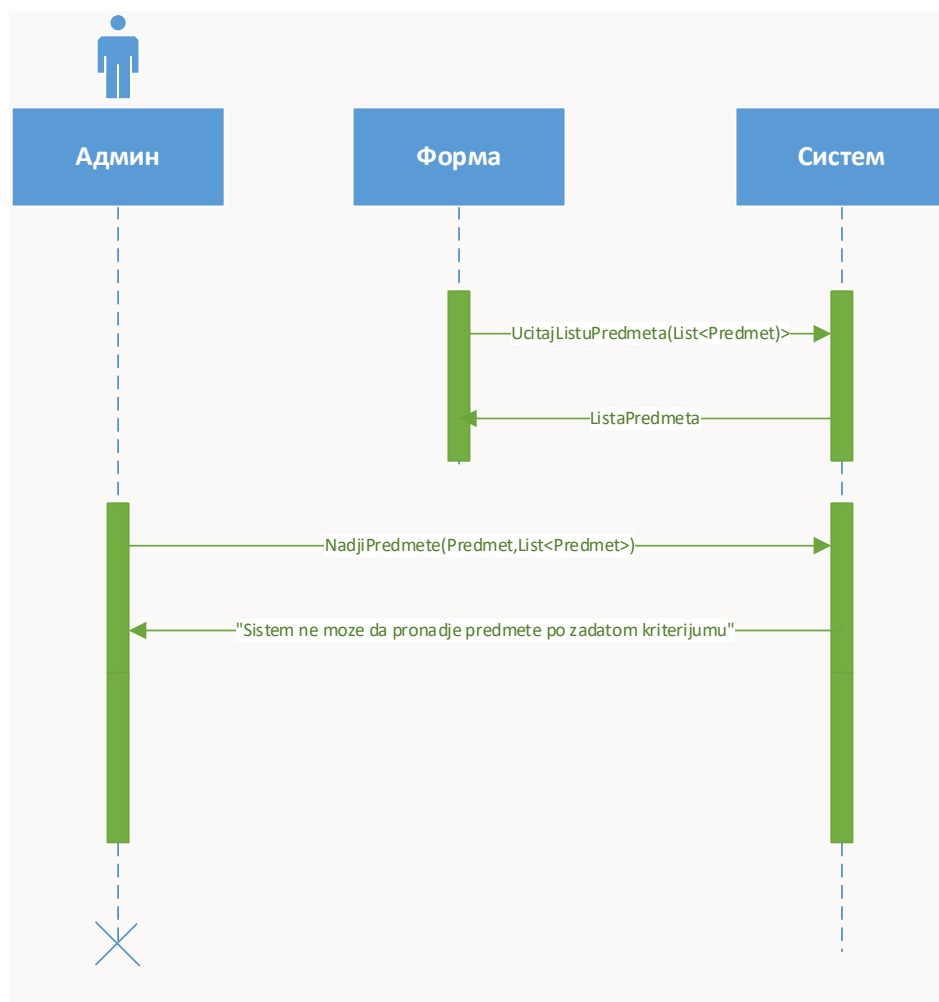
### ДСЗ:Дијаграм секвенци случаја коришћења- Претраживање предмета

- 1.Форма позива систем да учита листу предмета.(АПСО)
- 2.Систем враћа форми листу предмета.(ИА)
- 3.Админ позива систем да нађе предмете по задатом критеријуму. (АПСО)
- 4.Систем обавештава админа о успешно извршеној претрази одговарајућом поруком и приказује пронађене предмете.(ИА)
- 5.Админ позива систем да учита податке о одабраном предмету.(АПСО)
- 6.Систем обавештава админа о успешном учитавању података о предмету поруком “Одабрани предмет је приказан !“ и приказује податке о одабраном предмету.(ИА)



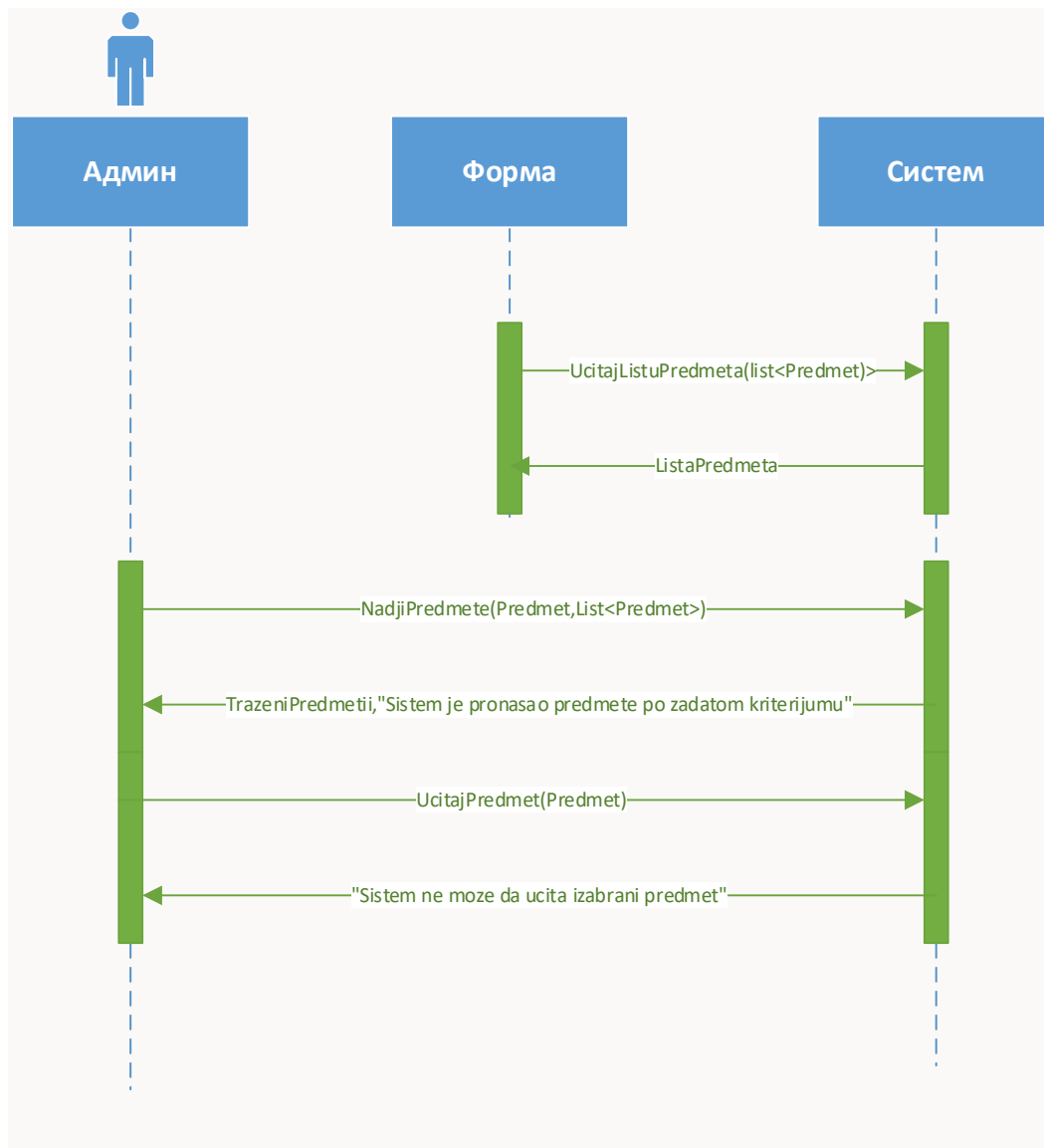
## Алтернативна сценарија

4.1 Уколико систем не може да нађе предмете он приказује админу поруку: “Систем не може да нађе предмете по задатом критеријуму“. Прекида се извршавање сценарија. (ИА)





6.1 Уколико систем не може да учита изабрани предмет он приказује админу поруку  
„Систем не може да учита изабрани предмет.“.(ИА)

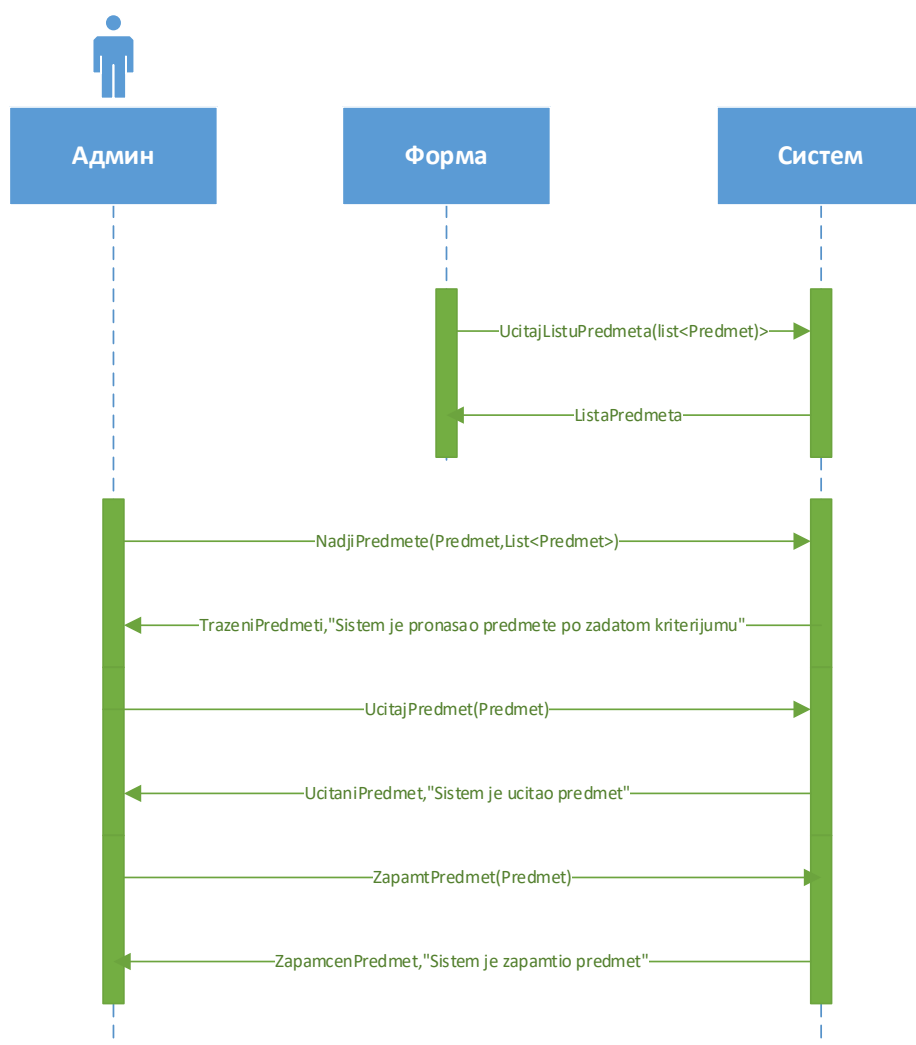


Са наведених секвенцијалних дијаграма уочавају се три системске операције:

- 1) Signal **UcitajListuPredmeta**
- 2) Signal **NadjiPredmete(Predmet, List<Predmet>)**
- 3) Signal **UcitajPredmet(Predmet)**

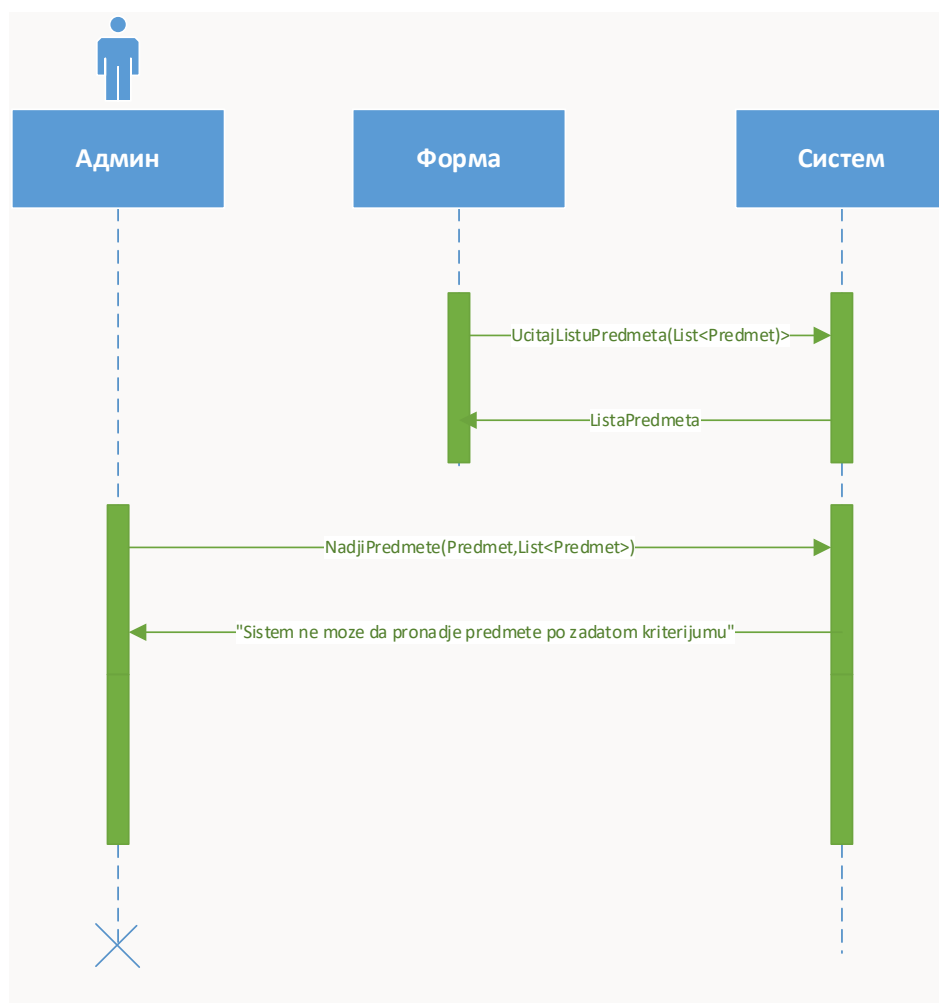
#### ДС4:Дијаграм секвенци случаја коришћења- Измена предмета

- 1.Форма позива систем да учита листу предмета.(АПСО)
- 2.Систем враћа форми листу предмета.(ИА)
- 3.Админ позива систем да нађе предмете по задатом критеријуму. (АПСО)
- 4.Систем обавештава админа о успешно извршеној претрази одговарајућом поруком и приказује пронађене предмете.(ИА)
- 5.Админ позива систем да учита податке о одабраном предмету.(АПСО)
- 6.Систем обавештава админа о успешном учитавању података о предмету поруком “Одабрани предмет је приказан !” и приказује податке о одабраном предмету.(ИА)
- 7.Админ позива систем да запамти податке о предмету. (АПСО)
- 8.Систем приказује админу запамћени предмет и поруку: “Систем је запамтио предмет.” (ИА)

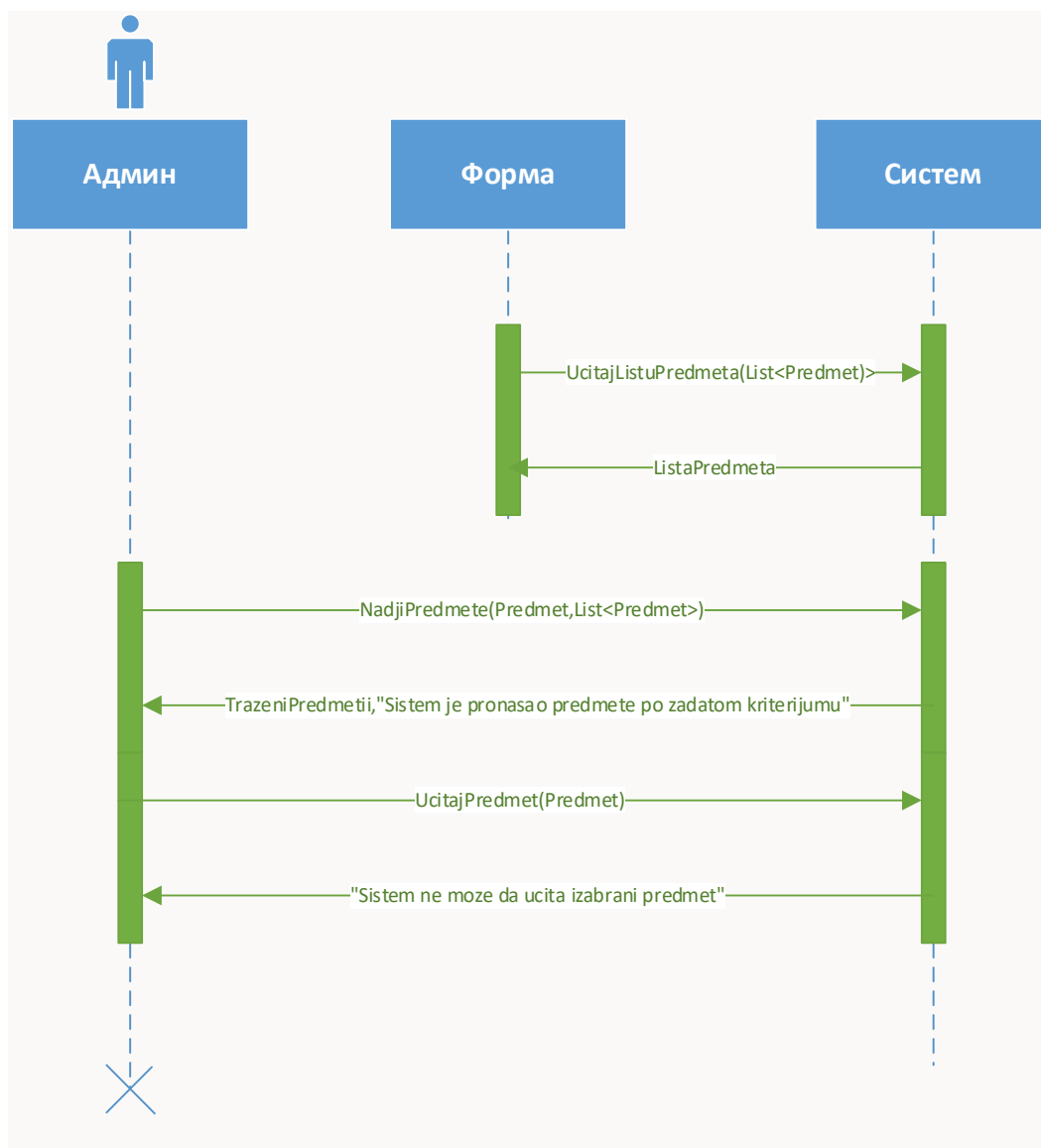


## Алтернативна сценарија

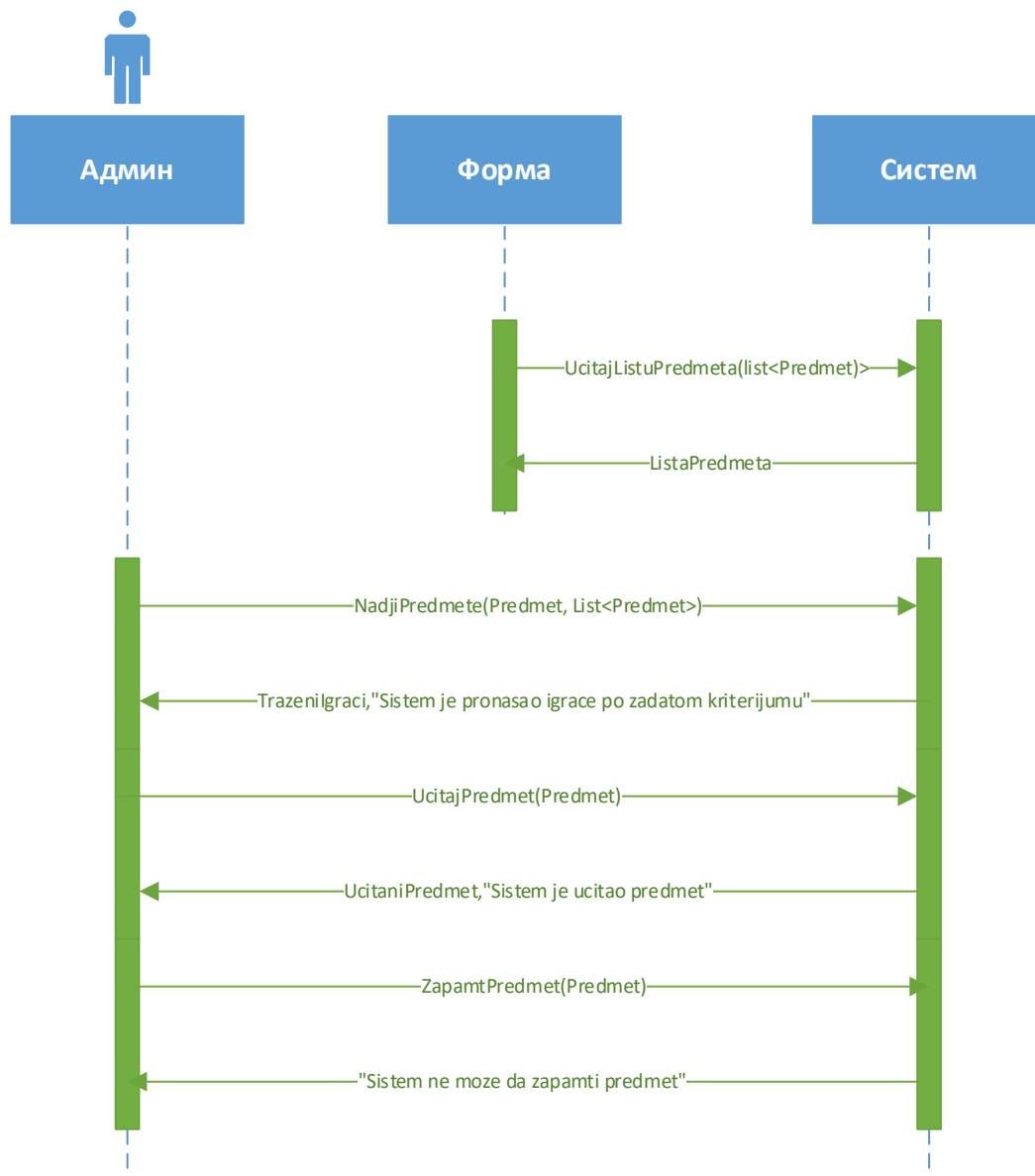
4.1 Уколико систем не може да нађе предмете он приказује админу поруку: “Систем не може да нађе предмете по задатом критеријуму“. Прекида се извршавање сценарија. (ИА)



6.1 Уколико систем не може да учита изабрани предмет он приказује админу поруку „Систем не може да учита изабрани предмет.“ Прекида се извршење сценарија.(ИА)



8.1. Уколико систем не може да запамти податке о предмету он приказује админу поруку “Систем не може да запамти предмет”. (ИА)

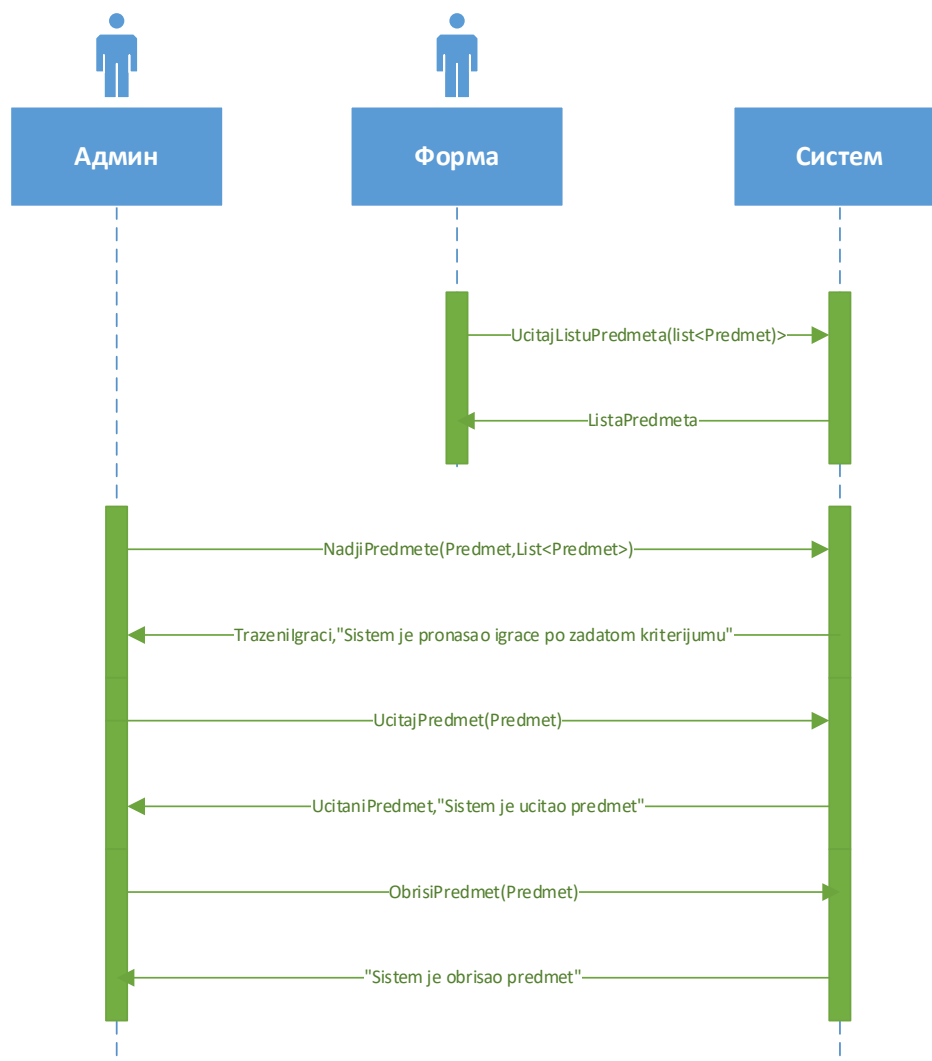


Са наведених секвенцијалних дијаграма уочавају се четири системске операције:

- 1) Signal **UcitajListuPredmeta(List<Predmet>)**
- 2) Signal **NadjiPredmete(Predmet, List<Predmet>)**
- 3) Signal **UcitajPredmet(Predmet)**
- 4) Signal **ZapamtiPredmet(Predmet)**

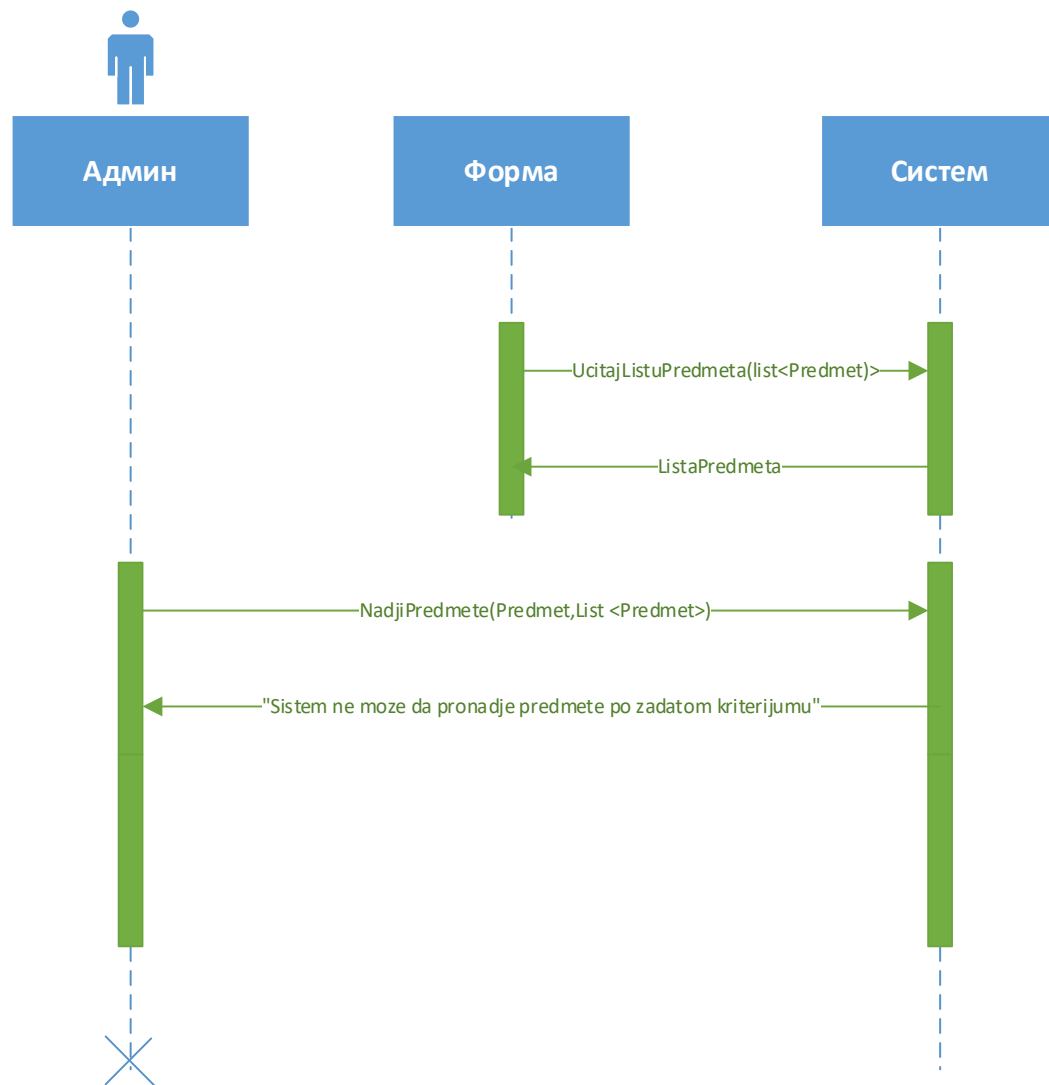
### ДС5:Дијаграм секвенци случаја коришћења- Брисање предмета

- 1.Форма позива систем да учита листу предмета.(АПСО)
- 2.Систем враћа форми листу предмета.(ИА)
- 3.Админ позива систем да нађе предмете по задатом критеријуму. (АПСО)
- 4.Систем обавештава админа о успешно извршеној претрази одговарајућом поруком и приказује пронађене предмете.(ИА)
- 5.Админ позива систем да учита податке о одабраном предмету.(АПСО)
- 6.Систем обавештава админа о успешном учитавању података о предмету поруком “Одабрани предмет је приказан !” и приказује податке о одабраном предмету.(ИА)
- 7.Админ позива систем да обрише предмет. (АПСО)
- 8.Систем приказује админу поруку: “Систем је обрисао предмет.” (ИА)

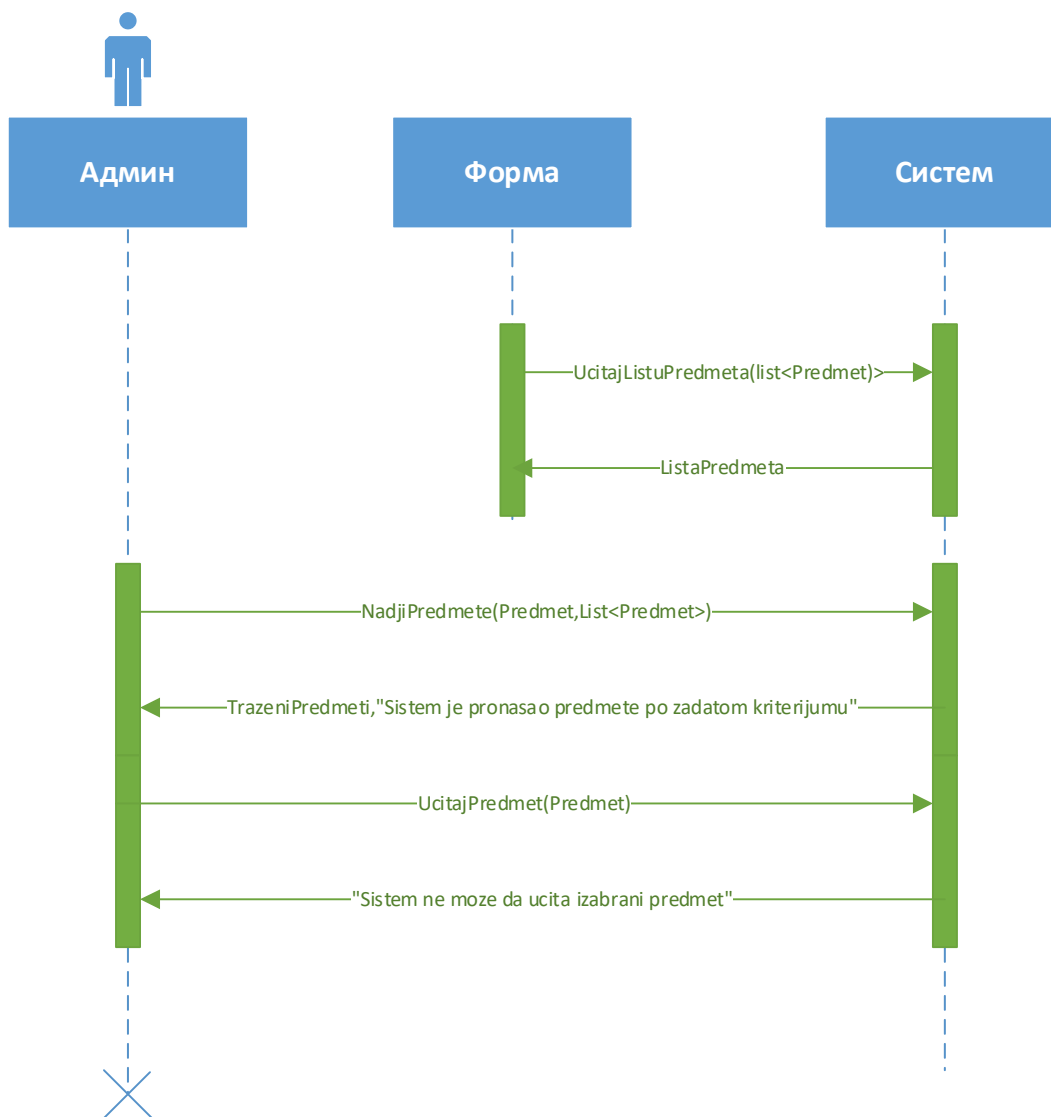


## Алтернативна сценарија

4.1 Уколико систем не може да нађе предмете он приказује админу поруку: “Систем не може да нађе предмете по задатом критеријуму“. Прекида се извршавање сценарија. (ИА)

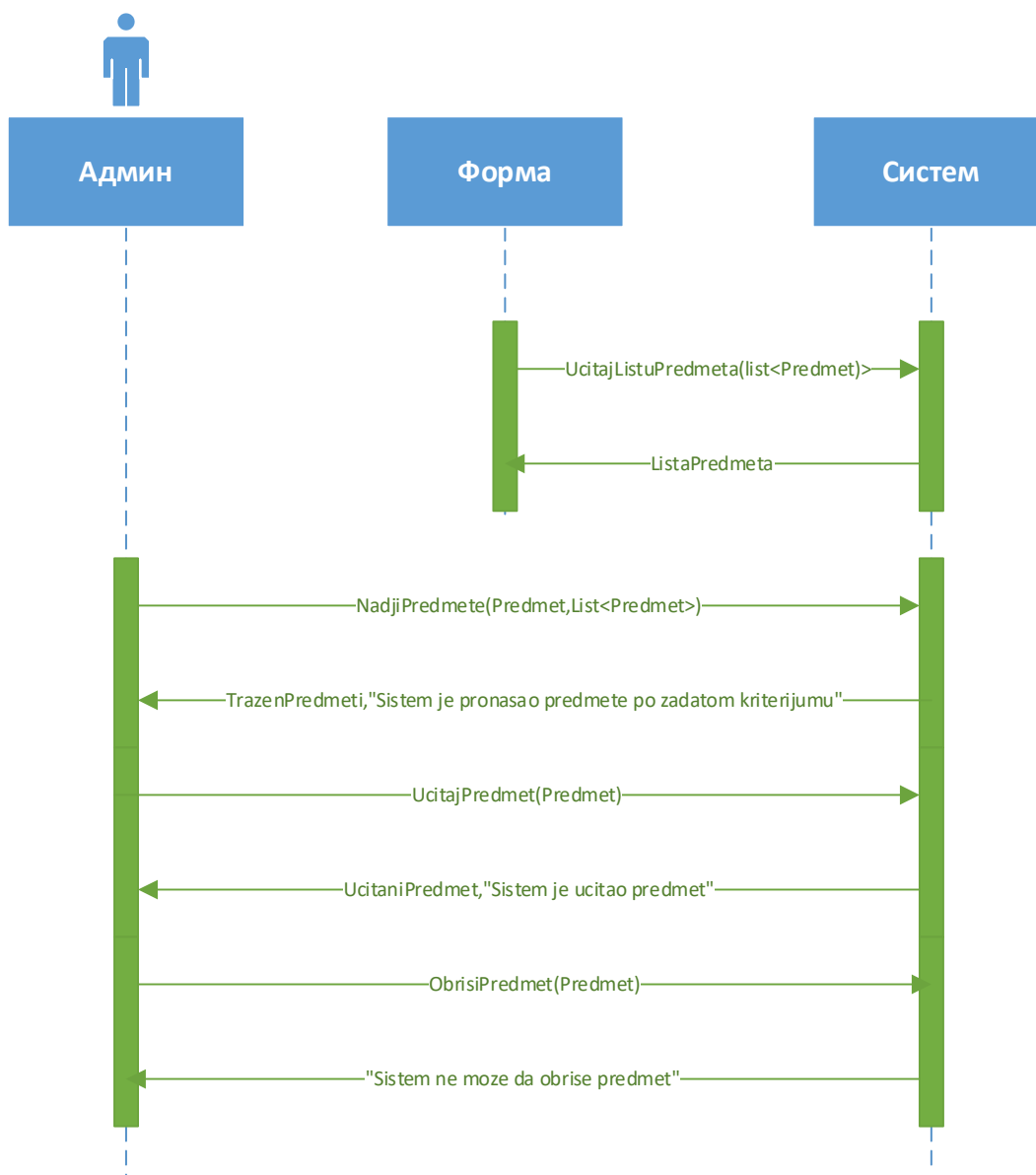


6.1 Уколико систем не може да учита изабрани предмет он приказује админу поруку „Систем не може да учита изабрани предмет.“ Прекида се извршење сценарија.(ИА)





8.1. Уколико систем не може да обрише предмет он приказује админу поруку “Систем не може да обрише предмет”. (ИА)

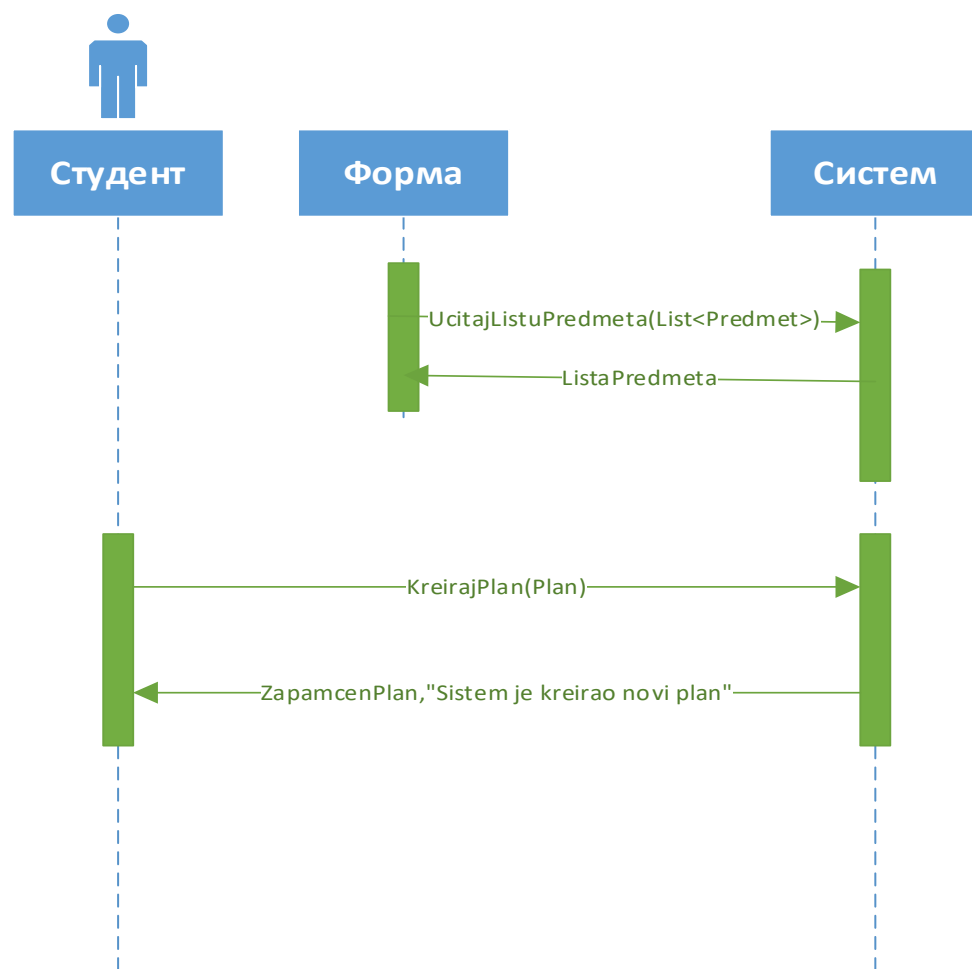


Са наведених секвенцијалних дијаграма уочавају се четири системске операције:

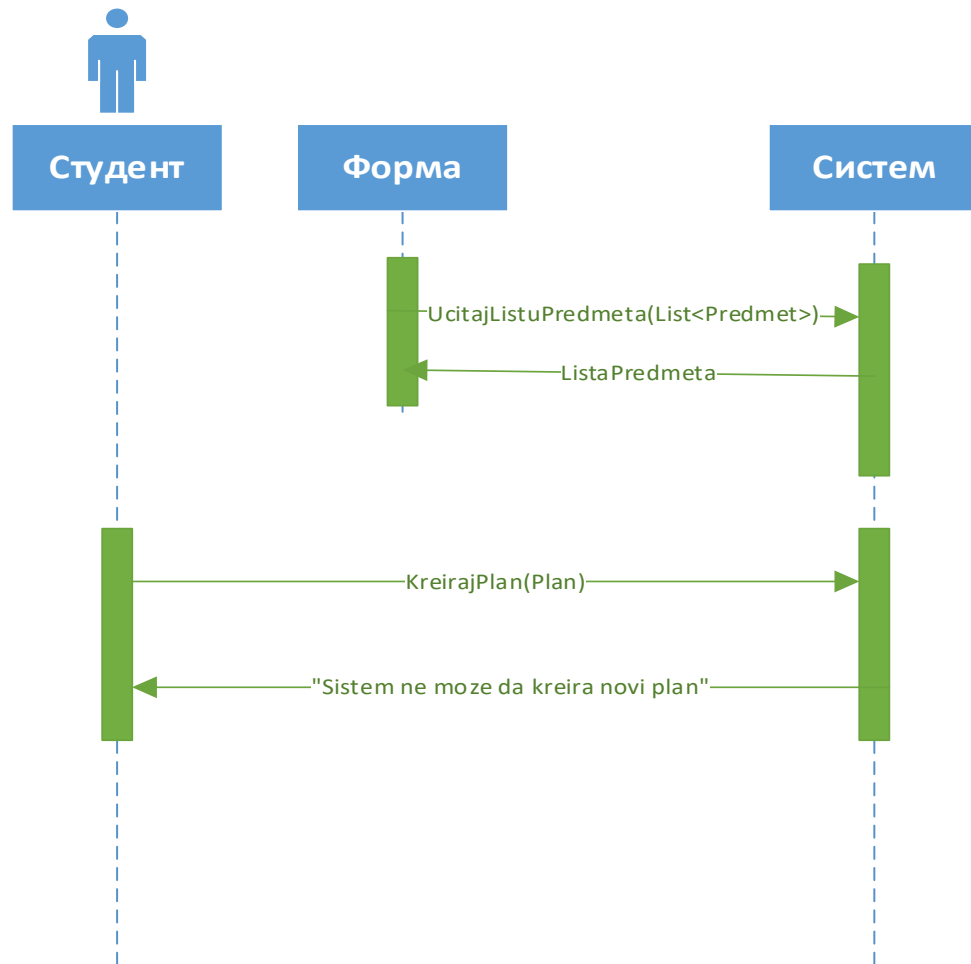
- 1) Signal **UcitajListuPredmeta(List<Predmet>)**
- 2) Signal **NadjiPredmete(Predmet,List<Predmet>)**
- 3) Signal **UcitajPredmet(Predmet)**
- 4) Signal **ObrisiPredmet(Predmet)**

### ДС6:Дијаграм секвенци случаја коришћења-Креирање новог плана

- 1.Форма позива систем да прочита листу предмета.(АПСО)
- 2.Систем враћа форми листу предмета.(ИА)
- 3.Студент позива систем да креира нови план. (АПСО)
- 4.Систем приказује студенту креирани план и поруку: “Систем је креирао нови план“.  
(ИА)



2.1 Уколико систем не може да запамти податке о плану он приказује студенту поруку “Систем не може да креира план”. (ИА)

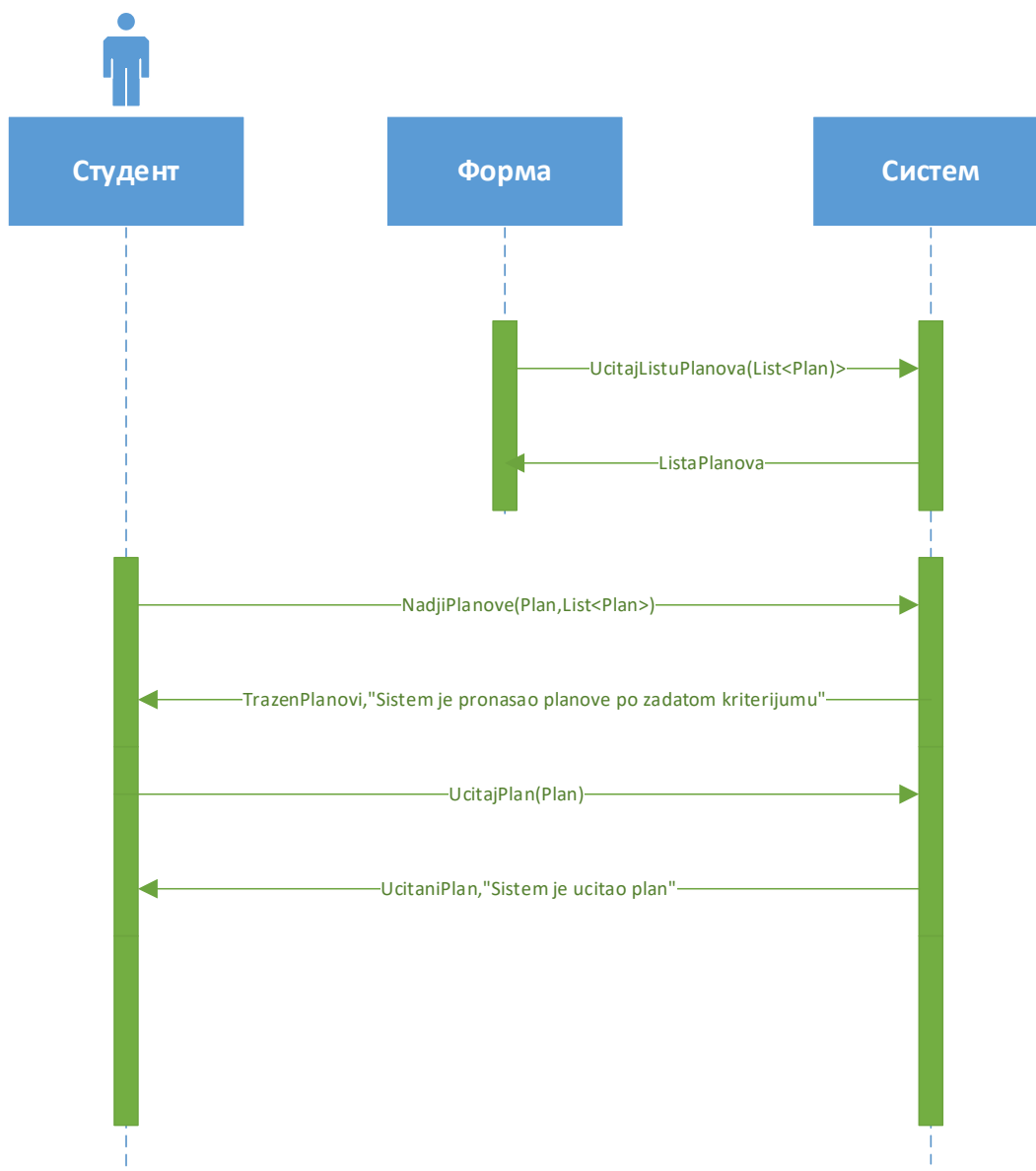


Са наведених секвенцијалних дијаграма уочавају се две системске операције:

- 1) Signal **UcitajListuPredmeta(List<Predmet>)**
- 2) Signal **KreirajPlan(Plan)**

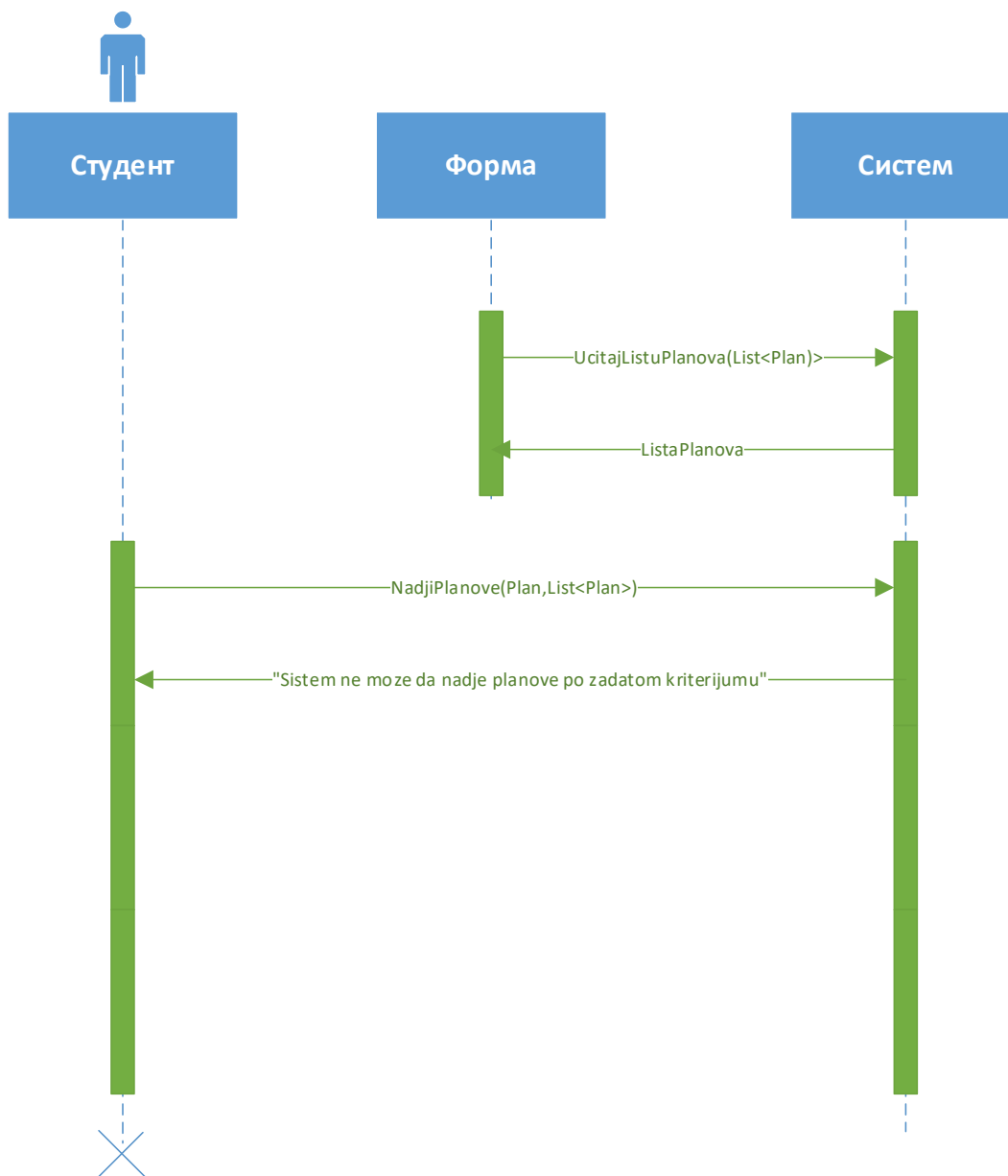
## ДС7:Дијаграм секвенци случаја коришћења-Претраживање плана

- 1.Форма позива систем да учита листу планова.(АПСО)
- 2.Систем враћа форми листу планова.(ИА)
- 3.Студент позива систем да нађе планове по задатом критеријуму. (АПСО)
- 4.Систем обавештава студента о успешно извршеној претрази одговарајућом поруком и приказује пронађене планове.(ИА)
- 5.Студент позива систем да учита податке о одабраном плану.(АПСО)
- 6.Систем обавештава студента о успешном учитавању података о плану поруком “Одабрани план је приказан !” и приказује податке о одабраном плану.(ИА)

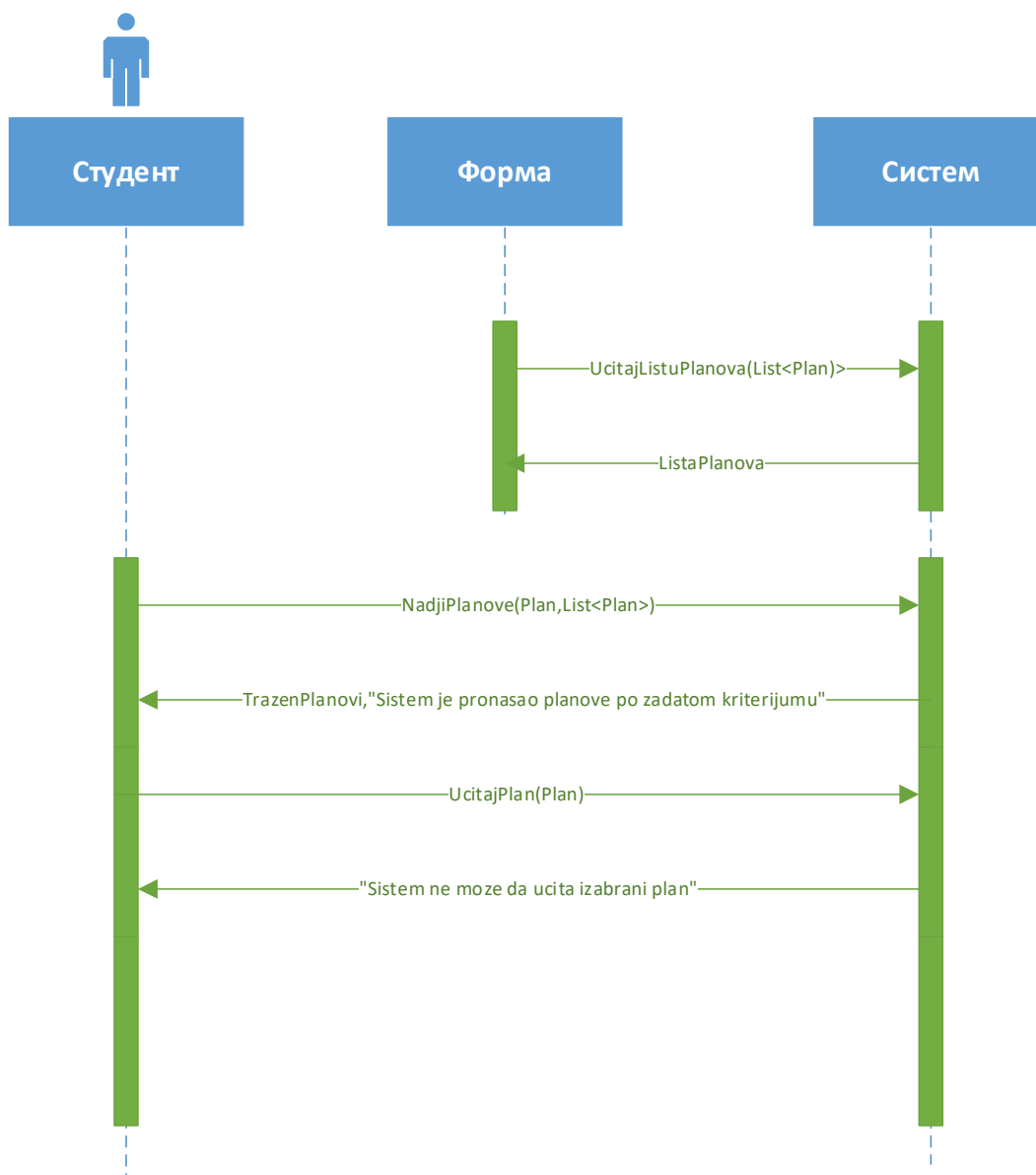


## Алтернативна сценарија

4.1 Уколико систем не може да нађе планове он приказује студенту поруку: “Систем не може да нађе планове по задатом критеријуму“. Прекида се извршавање сценарија. (ИА)



6.1. Уколико систем не може да учита изабрани план он приказује студенту поруку „Систем не може да учита изабрани план.“.(ИА)

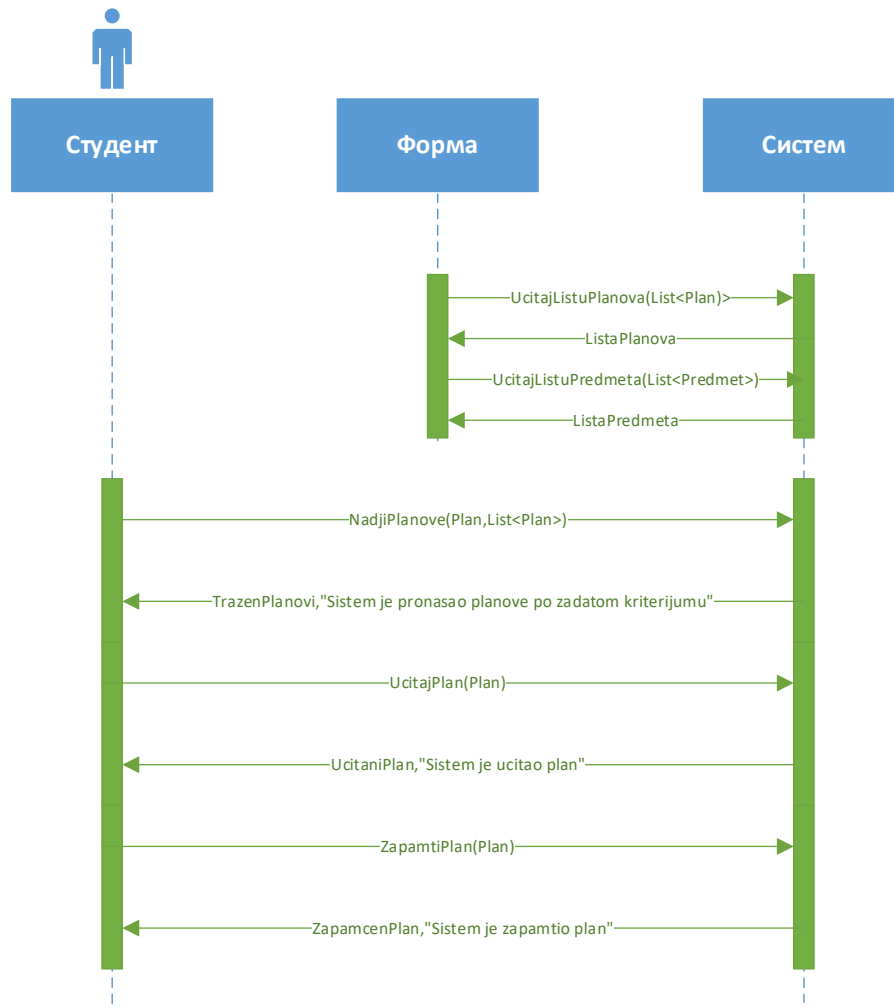


Са наведених секвенцијалних дијаграма уочавају се три системске операције:

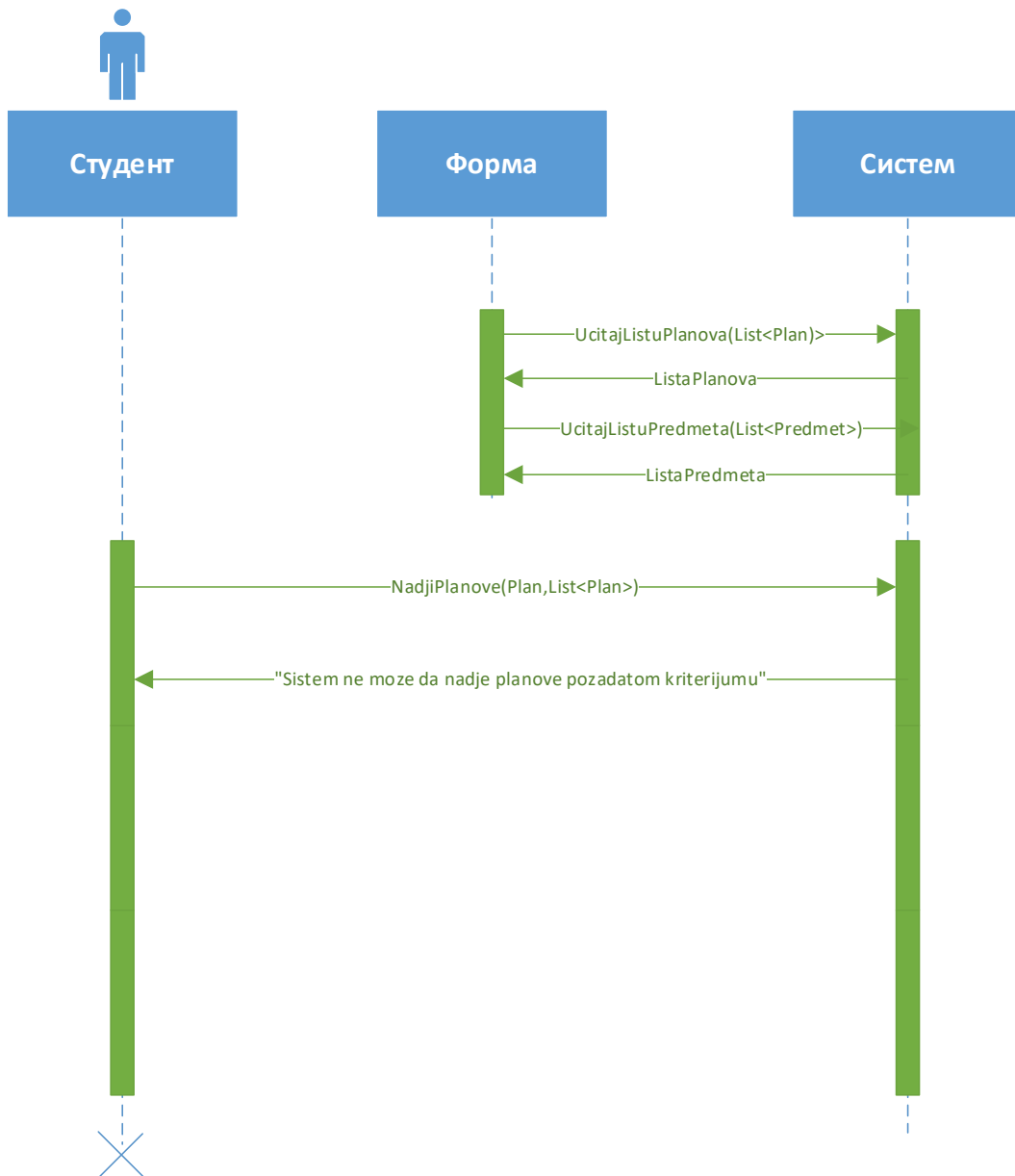
- 1) Signal **UcitajListuPlanova(List<Plan>)**
- 2) Signal **NadjiPlanove(Plan,List<Plan>)**
- 3) Signal **UcitajPlan(Plan)**

## ДС8:Дијаграм секвенци случаја коришћења-Измена плана

- 1.Форма позива систем да учита листу планова.(АПСО)
- 2.Систем враћа форми листу планова.(ИА)
- 3.Форма позива систем да учита листу предмета.(АПСО)
- 4.Систем враћа форми листу предмета.(ИА)
- 5.Студент позива систем да нађе планове по задатом критеријуму. (АПСО)
- 6.Систем обавештава студента о успешно извршеној претрази одговарајућом поруком и приказује пронађене планове.(ИА)
- 7.Студент позива систем да учита податке о одабраном плану.(АПСО)
- 8.Систем обавештава студента о успешном учитавању података о плану поруком “Одабрани план је приказан !“ и приказује податке о одабраном плану.(ИА)
- 9.Студент позива систем да запамти податке о плану. (АПСО)
- 10.Систем приказује студенту запамћени план и поруку: “Систем је запамтио план.” (ИА)

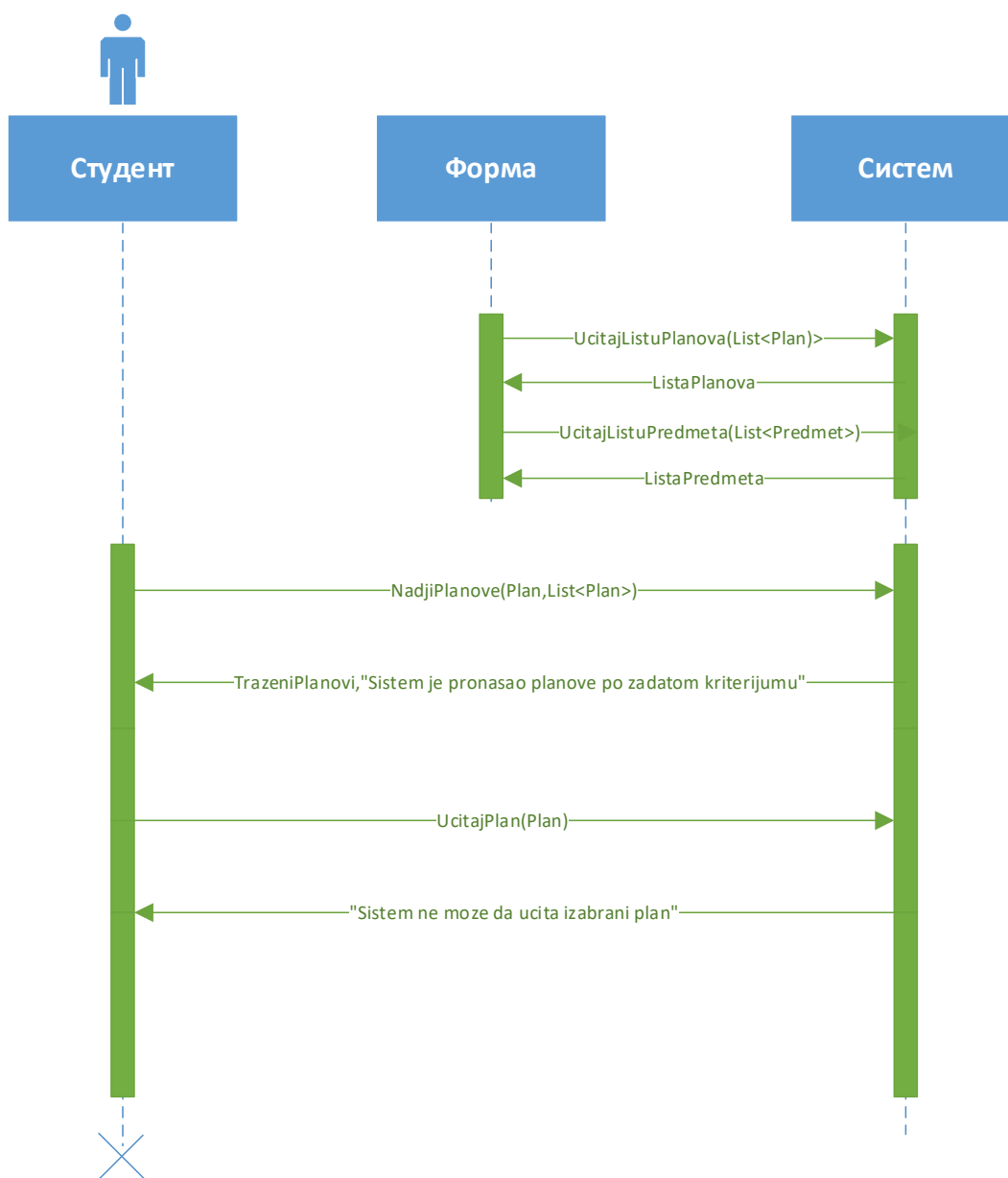


6.1 Уколико систем не може да нађе планове он приказује студенту поруку: “Систем не може да нађе планове по задатом критеријуму“. Прекида се извршавање сценарија. (ИА)

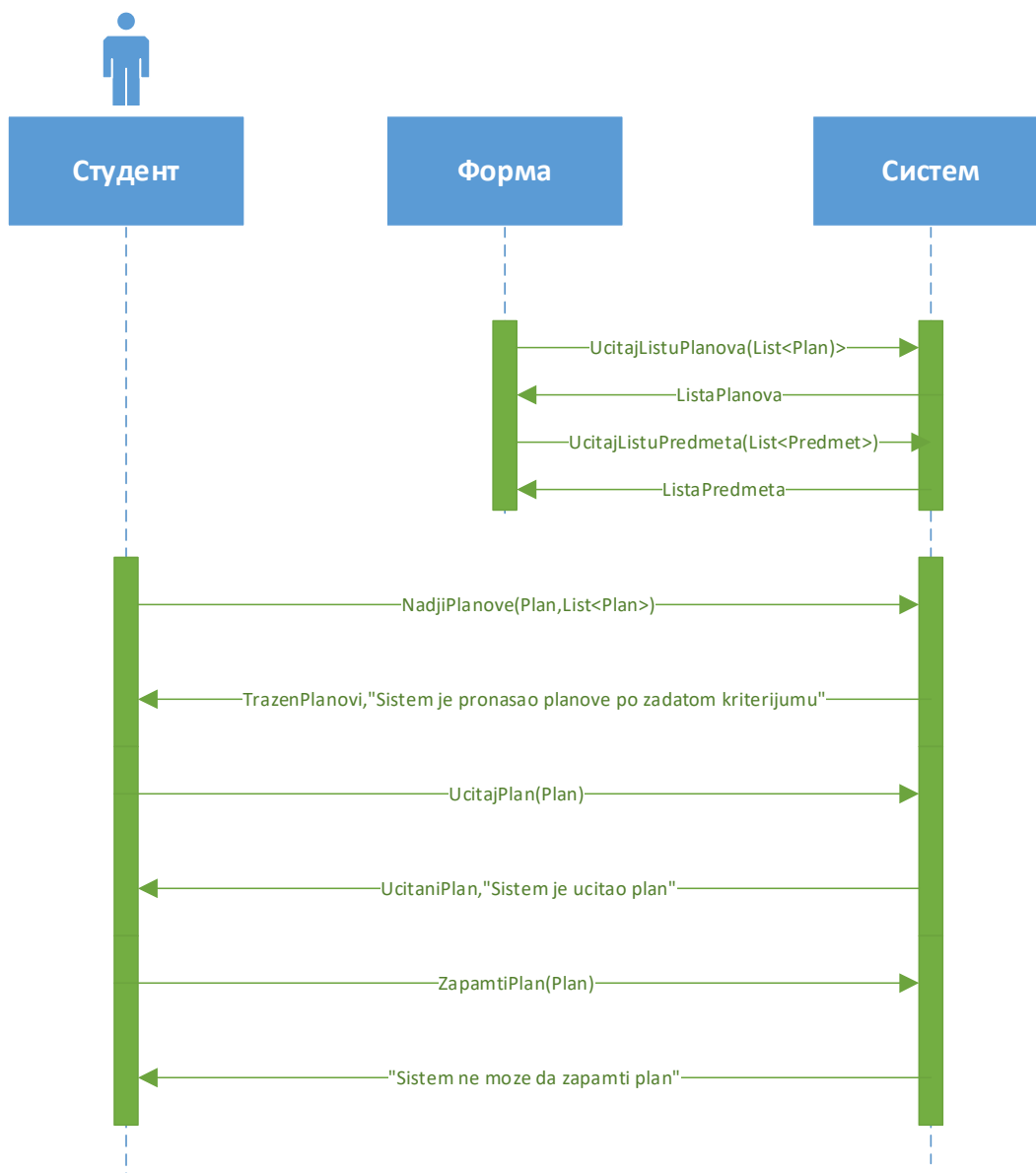




8.1. Уколико систем не може да учита изабрани план он приказује студенту поруку „Систем не може да учита изабрани план.“ Прекида се извршење сценарија.(ИА)



10.1. Уколико систем не може да запамти податке о плану он приказује студенту поруку “Систем не може да запамти план”. (ИА)

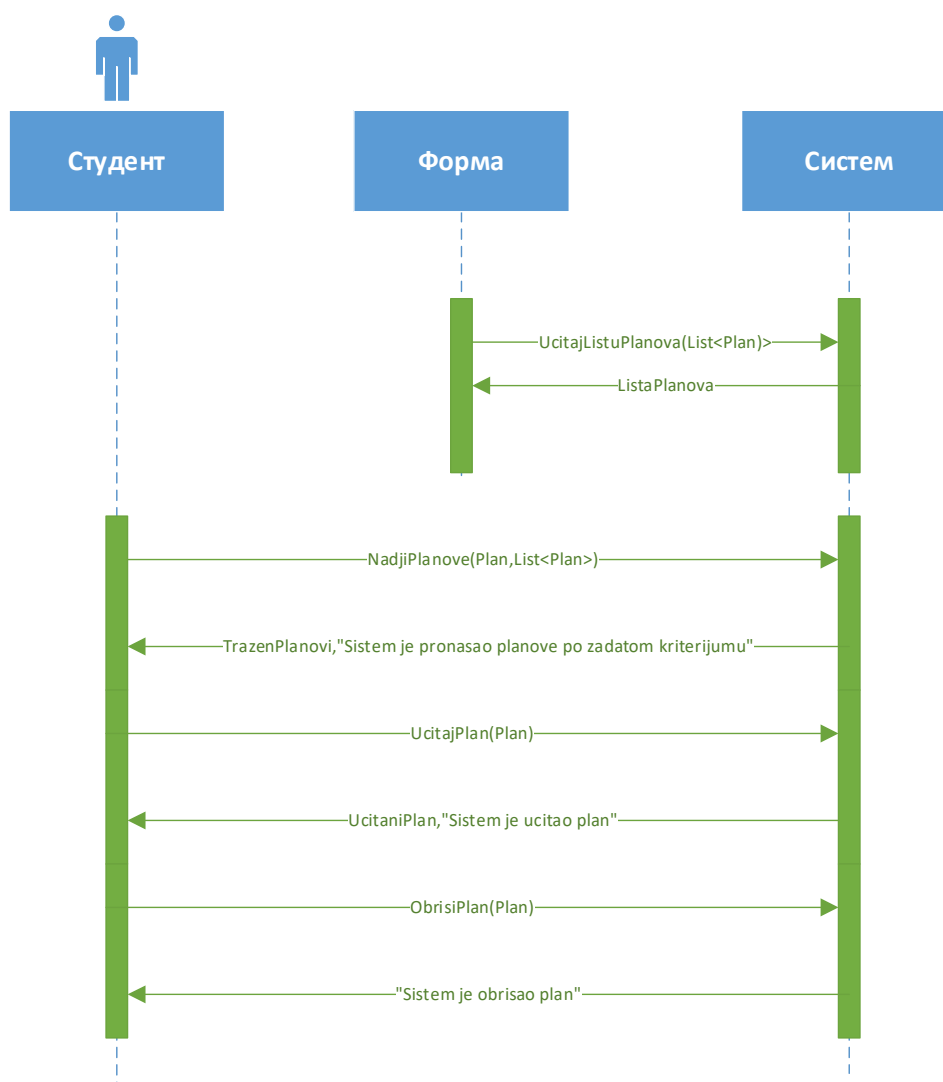


Са наведених секвенцијалних дијаграма уочавају се пет системских операција:

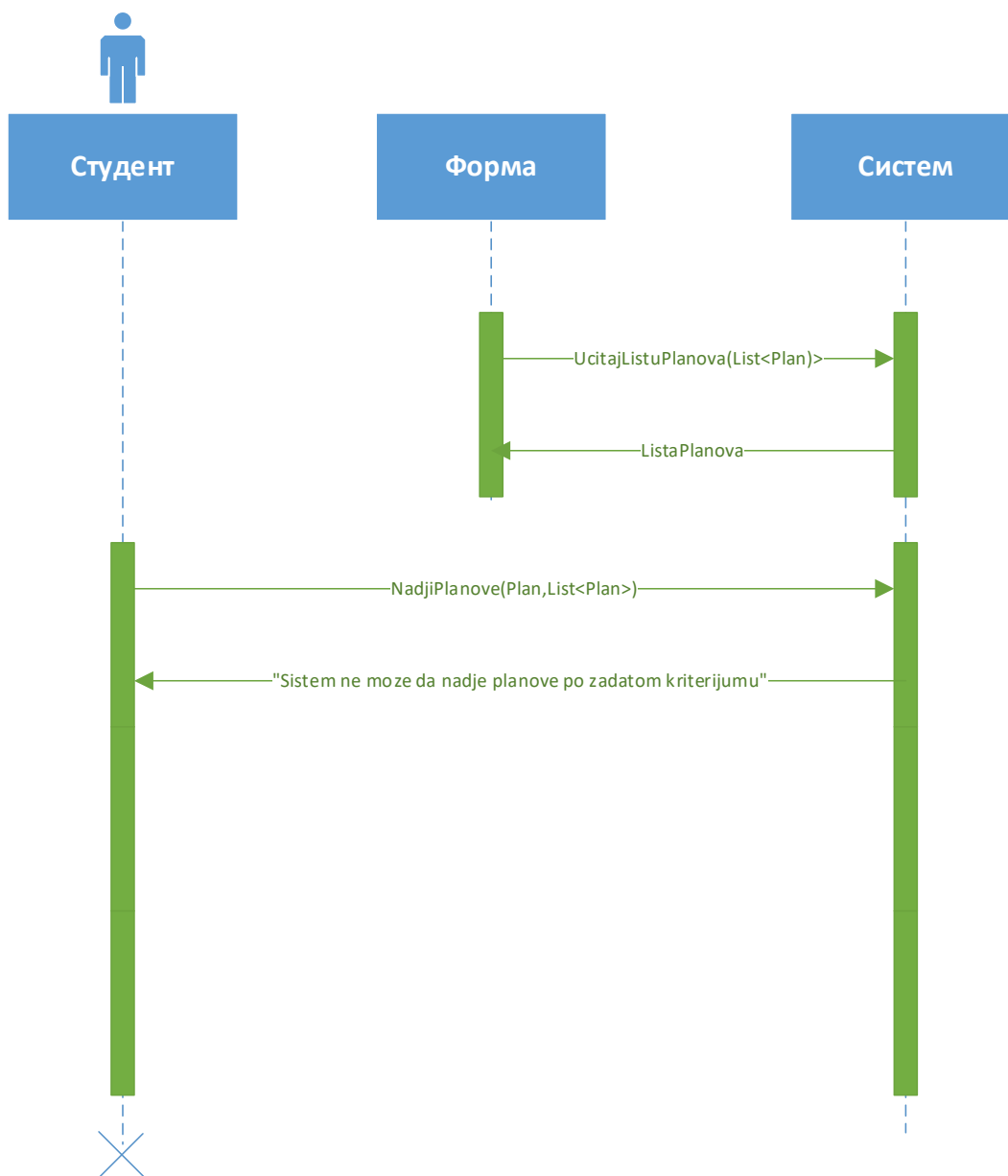
- 1) Signal **UcitajListuPlanova(List<Plan>)**
- 2) Signal **UcitajListuPredmeta(List<Predmet>)**
- 3) Signal **NadjiPlanove(Plan,List<Plan>)**
- 4) Signal **UcitajPlan(Plan)**
- 5) Signal **ZapamtiPlan(Plan)**

## ДС9:Дијаграм секвенци случаја коришћења-Брисање плана

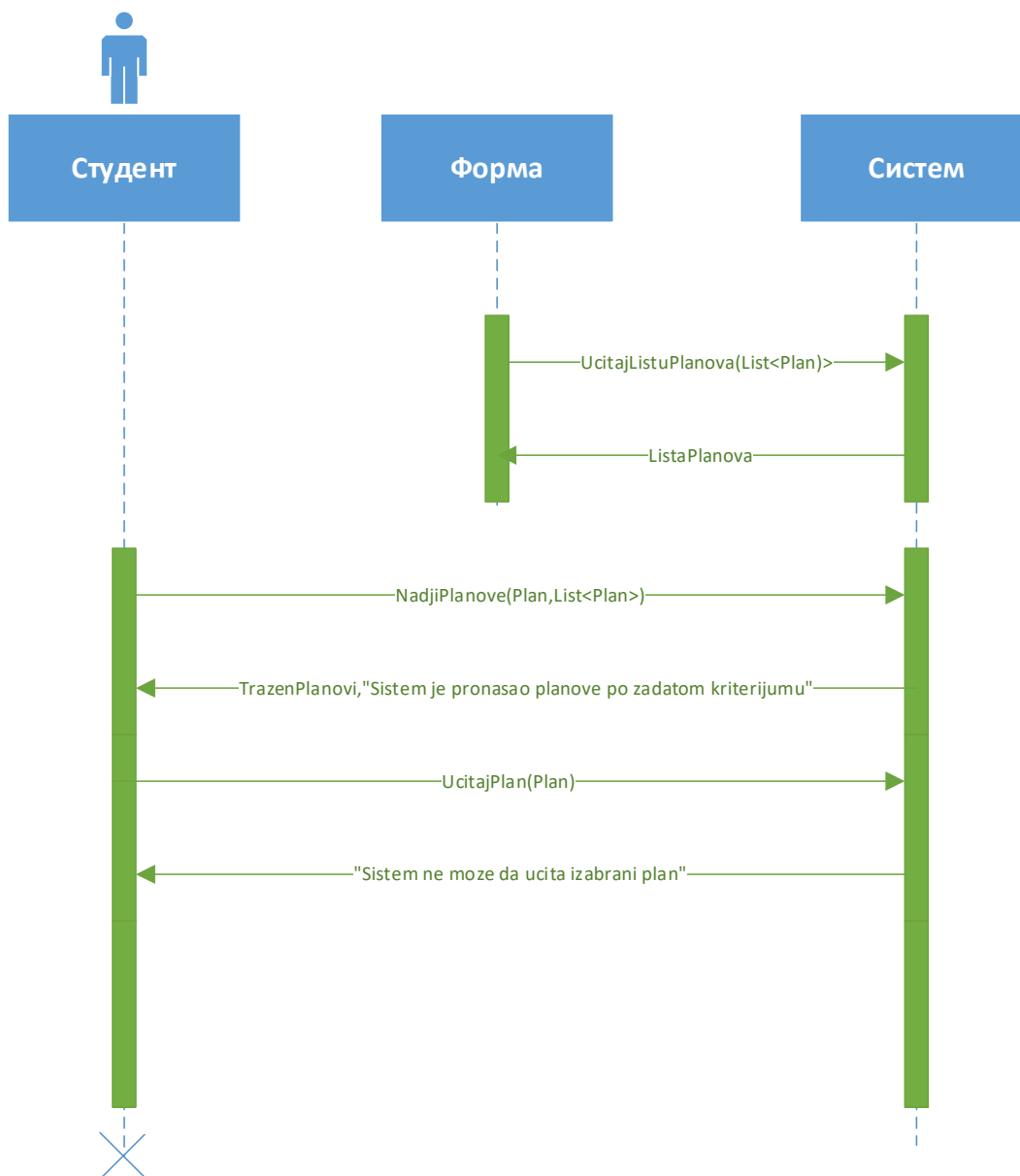
- 1.Форма позива систем да учита листу планова.(АПСО)
- 2.Систем враћа форми листу планова.(ИА)
- 3.Студент позива систем да нађе планове по задатом критеријуму. (АПСО)
- 4.Систем обавештава студента о успешно извршеној претрази одговарајућом поруком и приказује пронађене планове.(ИА)
- 5.Студент позива систем да учита податке о одабраном плану.(АПСО)
- 6.Систем обавештава студента о успешном учитавању података о плану поруком “Одабрани план је приказан !” и приказује податке о одабраном плану.(ИА)
- 7.Студент позива систем да обрише план. (АПСО)
- 8.Систем приказује кориснику поруку: “Систем је обрисао план.” (ИА)



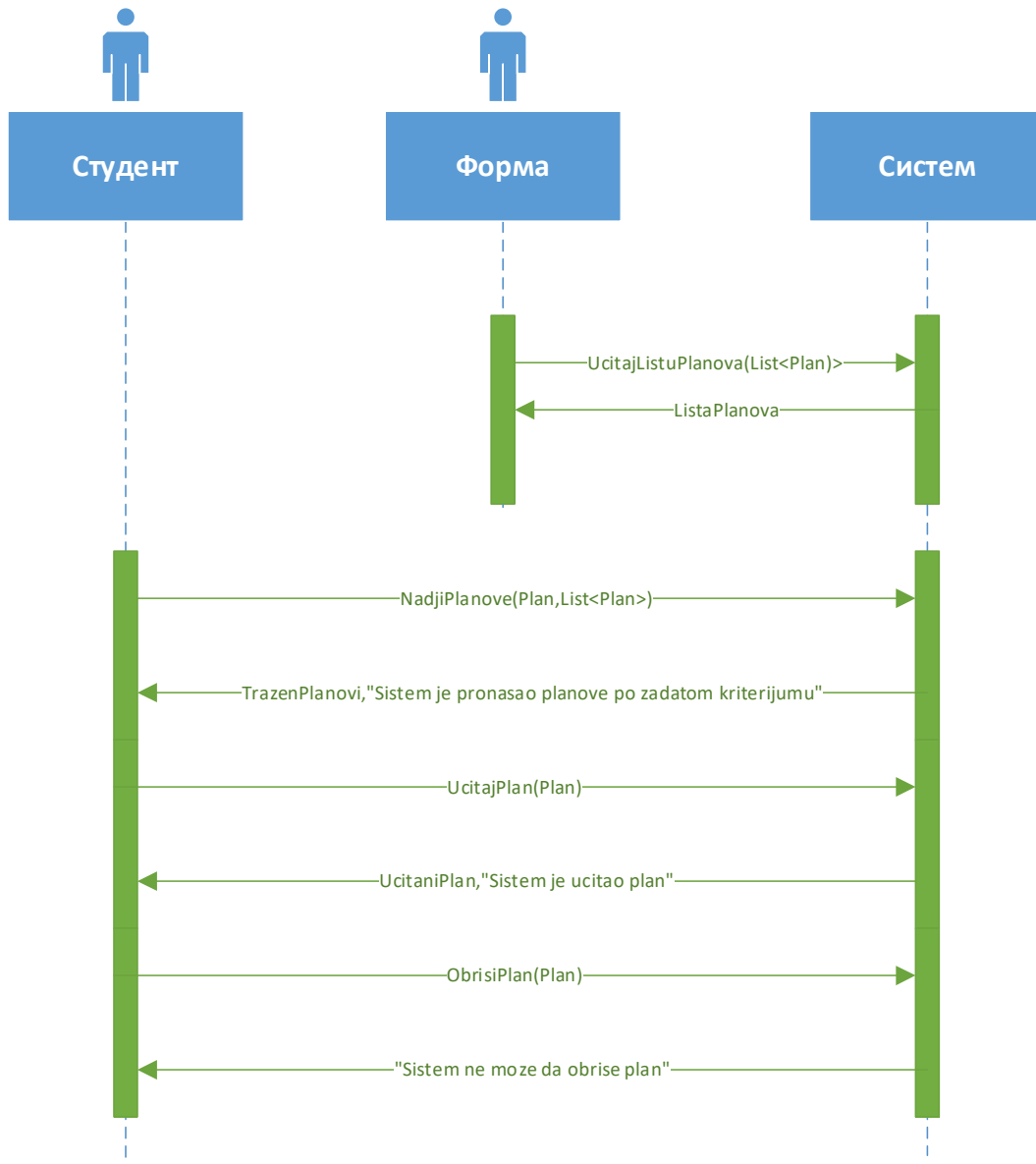
4.1 Уколико систем не може да нађе планове он приказује студенту поруку: “Систем не може да нађе планове по задатом критеријуму“. Прекида се извршавање сценарија. (ИА)



6.1. Уколико систем не може да учита изабрани план он приказује студенту поруку „Систем не може да учита изабрани план.“ Прекида се извршење сценарија. (ИА)



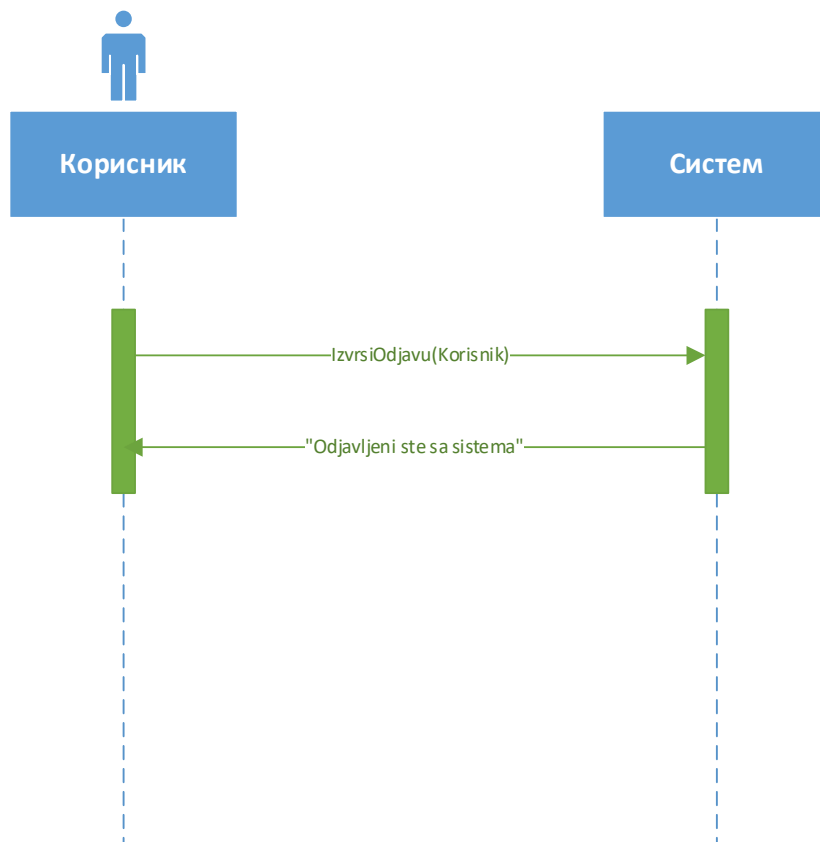
8.1 Уколико систем не може да обрише план он приказује кориснику поруку “Систем не може да обрише план”. (ИА)



- 1) Signal **UcitajListuPlanova(List<Plan>)**
- 2) Signal **NadjiPlanove(Plan,List<Plan>)**
- 3) Signal **UcitajPlan(Plan)**
- 4) Signal **ObrisiPlan (Plan)**

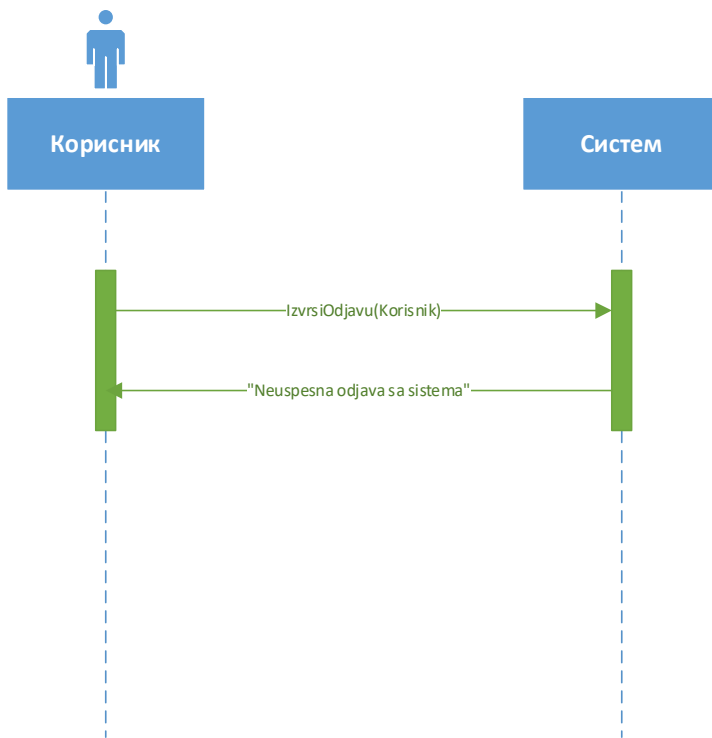
### ДС10:Дијаграм секвенци случаја коришћења-Одјављивање корисника

- 1.Корисник позива систем да изврши одјаву са система.(АПСО)
- 2.Систем онемогућава кориснику да приступи систему и приказује кориснику поруку:  
“Одјављени сте са система”. (ИА)



### Алтернативна сценарија

- 2.1 Уколико систем не може да одјави корисника, систем приказује кориснику поруку:  
“Неуспешна одјава са система“ (ИА)



Са наведених секвенцијалних дијаграма уочава се једна системска операција:

1) Signal **IzvršiOdjavu(Korisnik)**

На основу анализе сценарија добијено је 12 системских операција:

- 1) Signal **IzvršiPrijavu(Korisnik)**
- 2) Signal **ZapamtiPredmet(Predmet)**
- 3) Signal **NadjiPredmete(Predmet, List<Predmet>)**
- 4) Signal **UcitajPredmet(Predmet)**
- 5) Signal **ObrisiPredmet(Predmet)**
- 6) Signal **ZapamtiPlan(Plan)**
- 7) Signal **UcitajPlan(Plan)**
- 8) Signal **NadjiPlanove(Plan, List<Plan>)**
- 9) Signal **ObrisiPlan(Plan)**
- 10) Signal **UcitajListuPredmeta(List<Predmet>)**
- 11) Signal **UcitajListuPlanova(List<Plan>)**
- 12) Signal **IzvršiOdjavu(Korisnik)**
- 13) Signal **KreirajPredmet(Predmet)**
- 14) Signal **KreirajPlan(Plan)**



#### **4.2.2. Понашање софтверског система – Дефинисање уговора о системским операцијама**

**Уговор УГ1:** IzvrsiPrijavu(Korisnik) Signal;

Веза са СК: СК1

Предуслови:

Постуслови:

**Уговор УГ2:** ZapamtiPredmet(Predmet) Signal;

Веза са СК: СК4

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

Постуслови: Подаци о предмету су запамћени.

**Уговор УГ3:** NadjiPredmete(Predmet, List<Predmet>) Signal;

Веза са СК: СК3,СК4 ,СК5

Предуслови: /

Постуслови: /

**Уговор УГ4:** UcitajPredmet(Predmet) Signal;

Веза са СК: СК3,СК4 ,СК5

Предуслови: /

Постуслови: /

**Уговор УГ5:** ObrisiPredmet(Predmet) Signal;

Веза са СК: СК5

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

Постуслови: Подаци о предмету су избрисани.

**Уговор УГ6:** ZapamtiPlan(Plan) Signal;

Веза са СК: СК8

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

Постуслови: Подаци о плану су запамћени.

**Уговор УГ7:** NadjiPlanove (Plan, List<Plan>) Signal;

Веза са СК: СК7,СК8 ,СК9

Предуслови: /

Постуслови: /

**Уговор УГ8:** UcitajPlan (Plan) Signal;

Веза са СК: СК7,СК8 ,СК9

Предуслови: /

Постуслови: /

**Уговор УГ9:** ObrisiPlan (Plan) Signal;

Веза са СК: СК9

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

Постуслови: Подаци о плану су избрисани.

**Уговор УГ10:** UcitajListuPredmeta(List<Predmet>) Signal;

Веза са СК: СК3, СК4, СК5, СК6, СК8

Предуслови: /

Постуслови: /

**Уговор УГ11:** UcitajListuPlanova(List<Plan >) Signal;

Веза са СК: СК7, СК8, СК9

Предуслови: /

Постуслови: /

**Уговор УГ12:** IzvrsiOdjavu (Korisnik) Signal;

Веза са СК: СК10

Предуслови: /

Постуслови: /

**Уговор УГ13:** KreirajPredmet(Predmet) Signal;

Веза са СК: СК2

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

Постуслови: Креиран је нови предмет.

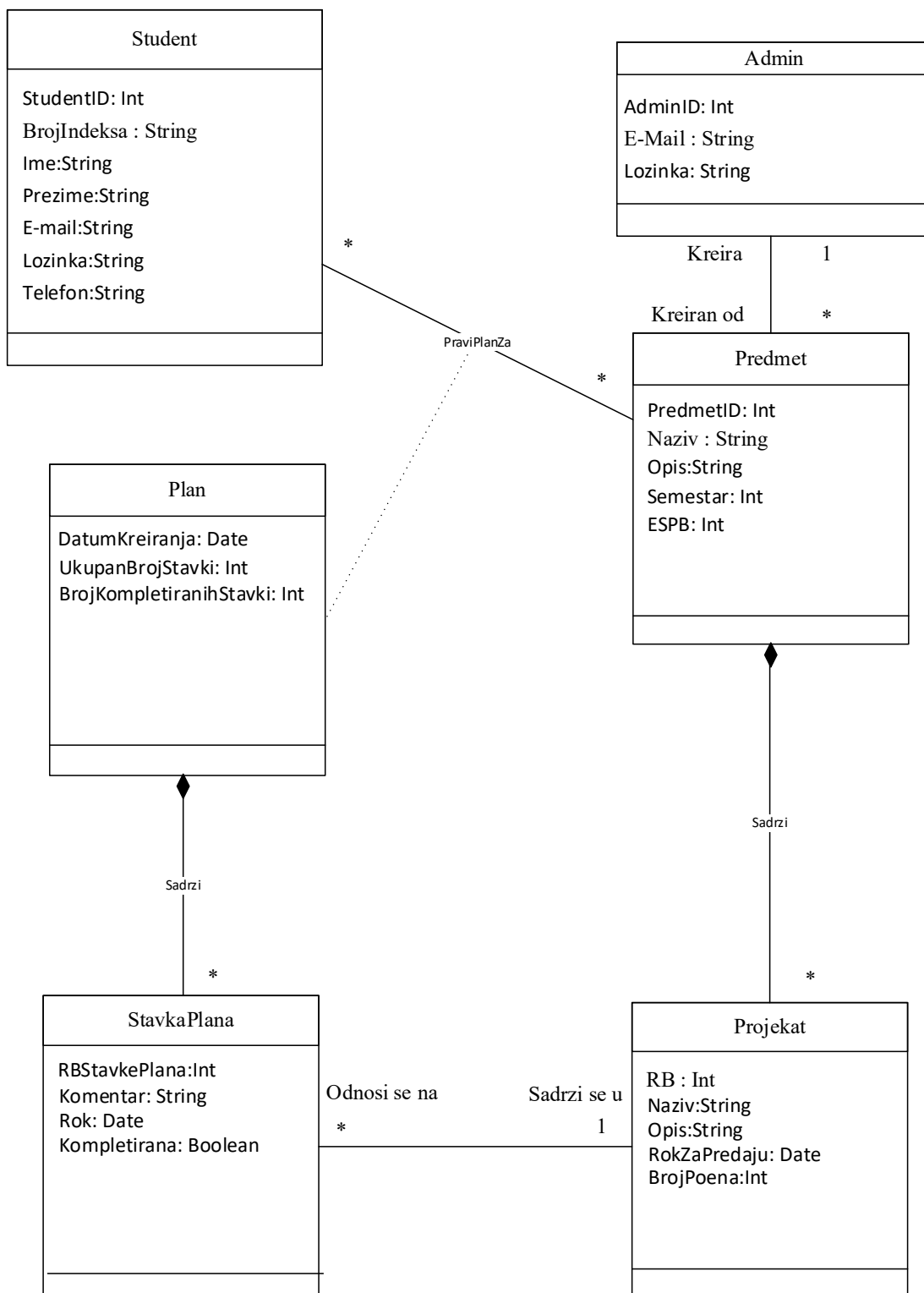
**Уговор УГ14 :** KreirajPlan(Plan) Signal;

Веза са СК: СК6

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

Постуслови: Креиран је нови план.

### 4.2.3. Структура софтверског система – Концептуални модел



#### 4.2.4. Структура софтверског система – Релациони модел

Admin(AdminID, Email, Lozinka)

Student(StudentID, BrojIndeksa, Ime, Prezime, Email, Lozinka, Telefon)

Predmet(PredmetID, Naziv, Opis, Semestar, Espb, *AdminID*)

Projekat(PredmetID, RB, Naziv, Opis, BrojPoena, RokZaPredaju)

Plan(StudentID, PredmetID, DatumKreiranja, UkupanBrojStavki, BrojKompletiranihStavki)

StavkaPlana(StudentID, PredmetID, RBStavkePlana, Komentar, Rok, Kompletirana, *PredmetProjektaID*, *RBProjekta*)

Tabela Student		Prosto vrednosno ograničenje		Složeno vrednosno ograničenje		Strukturno ograničenje
Atributi	Ime	Tip atributa	Vrednost atributa	Medjuzavisnost atributa jedne tabele	Medjuzavinost atributa vise tabela	Insert /  Update Cascades Plan  Delete Cascade Plan
	StudentID	Int	Not null and >0			
	BrojIndeksa	String	Not null			
	Ime	String	Not null			
	Prezime	String	Not null			
	Email	String	Not null			
	Lozinka	String	Not null			
	Telefon	String	Not null			

Tabela Admin		Prosto vrednosno ograničenje		Složeno vrednosno ograničenje		Strukturno ograničenje
Admin	Ime	Tip atributa	Vrednost atributa	Medjuzavisnost atributa jedne tabele	Medjuzavinost atributa vise tabela	Insert /  Update Cascades Predmet  Delete Restricted Predmet
	AdminID	Int	Not null and > 0			
	Email	String	Not null			
	Lozinka	String	Not null			

Tabela Predmet		Prosto vrednosno ograničenje		Složeno vrednosno ograničenje		Strukturno ograničenje
Atributi	Ime	Tip atributa	Vrednost atributa	Medjuzavisnost atributa jedne tabele	Medjuzavinost atributa vise tabela	Insert Restricted Admin  Update Restricted Admin Cascades Plan, Projekat  Delete Restricted Plan Cascades Projekat
	PredmetID	Int	Not null and >0			
	Naziv	String	Not null			
	Opis	String	Not null			
	Semestar	Int	Not null and >0 and <5			
	Epsb	Int	Not null and >0			
	AdminID	Int	Not null and >0			

Tabela Plan		Prosto vrednosno ograničenje		Složeno vrednosno ograničenje		Strukturno ograničenje
Atributi	Ime	Tip atributa	Vrednost atributa	Medjuzavisnost atributa jedne tabele	Medjuzavinost atributa vise tabela	Insert Restricted Student,Predmet  Update Restricted Student, Predmet Cascades StavkaPlana  Delete Cascades StavkaPlana
	StudentID	Int	Not null and > 0			
	PredmetID	Int	Not null and > 0			
	DatumKreiranja	Date	Not null			
	UkupanBrojStavki	Int	Not null			
	BrojKompletiranihStavki	Int	Not null		BrojKompletiranihStavki=Count(StavkaPlana) Where StavkaPlana.Kompletirana==TRUE	

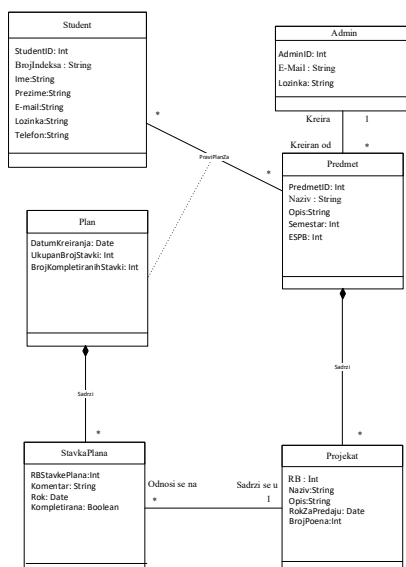
Tabela StavkaPlana		Prosto vrednosno ograničenje		Složeno vrednosno ograničenje		Strukturno ograničenje
Atributi	Ime	Tip atributa	Vrednost atributa	Medjuzavisnost atributa jedne tabele	Medjuzavinost atributa vise tabela	Insert Restricted Plan, Projekat  Update Restricted Plan,Projekat  Delete /
	StudentID	Int	Not null and > 0			
	PredmetID	Int	Not null and > 0			
	RBStavkePLana	Int	Not null and > 0			
	Komentar	String	Not null			
	Rok	String	Not null			
	Kompletirana	Boolean	NotNull			
	PredmetProjektaID	Int	Not null and > 0			
	RBProjekta	Int	Not null and > 0			

Tabela Projekat		Prosto vrednosno ograničenje		Složeno vrednosno ograničenje		Strukturno ograničenje
Atributi	Ime	Tip atributa	Vrednost atributa	Medjuzavisnost atributa jedne tabele	Medjuzavinost atributa vise tabela	Insert Restricted Predmet  Update Restricted Predmet Cascades StavkaPlana  Delete Restricted StavkaPlana
	PredmetID	Int	Not null and > 0			
	RB	Int	Not null and > 0			
	Naziv	String	Not null			
	Opis	String	Not null			
	BrojPoena	Int	Not null and >0			
	RokZaPredaju	Date	Not null			

Kao rezultat scenarija SK i pravljena konceptualnog modela добија се логичка структура и понашање софтверског система:

## Softverski sistem

### Struktura Sistema- Konceptualni Model



### Ponasanje Sistema – 14 Sistemskih operacija

Signal **IzvršiPrijavu(Korisnik)**  
 Signal **KreirajPredmet(Predmet)**  
 Signal **ZapamtiPredmet(Predmet)**  
 Signal **NadjiPredmete(Predmet,List<Predmet>)**  
 Signal **UcitajPredmet(Predmet)**  
 Signal **ObrisiPredmet(Predmet)**  
 Signal **KreirajPlan(Plan)**  
 Signal **ZapamtiPlan(Plan)**  
 Signal **UcitajPlan(Plan)**  
 Signal **NadjiPlanove(Plan, List<Plan>)**  
 Signal **ObrisiPlan(Plan)**  
 Signal **UcitajListuPredmeta(List<Predmet>)**  
 Signal **UcitajListuPlanova(List<Plan>)**  
 Signal **IzvršiOdjavu(Korisnik)**



### 4.3. Пројектовање

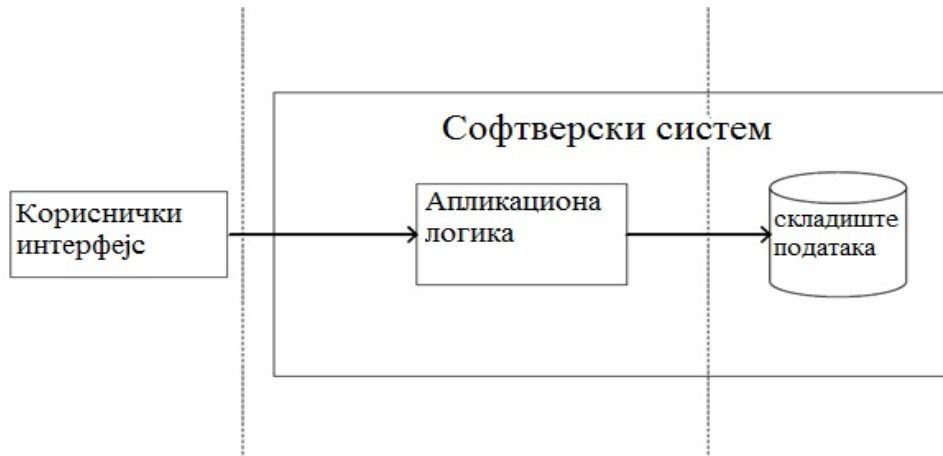
Фаза пројектовања описује физичку структуру и понашање софтверског система (архитектуру софтверског система).

#### Архитектура софтверског система

У овом раду је коришћена класична тронивојска архитектура. Тронивојска архитектура се састоји од следећа три нивоа:

- Ниво корисничког интерфејса
- Ниво апликационе логике
- Ниво складишта података

На основу тронивојске архитектуре направљени су савремени апликациони сервери чија је улога да обезбеде сервисе који омогућавају реализацију апликационе логике софтверског система [1]. Самим тим, софтверски систем је подељен на клијентски и серверски део, који комуницирају између себе помоћу сокета. Ниво корисничког интерфејса је на страни клијента, а апликациона логика и складиште података су на страни сервера.



Слика 24. Тронивојска архитектура[1]

### 4.3.1. Пројектовање корисничког интерфејса

Кориснички интерфејс представља реализацију улаза и/или излаза софтверског система и састоји се од екранске форме и контролера корисничког интерфејса.

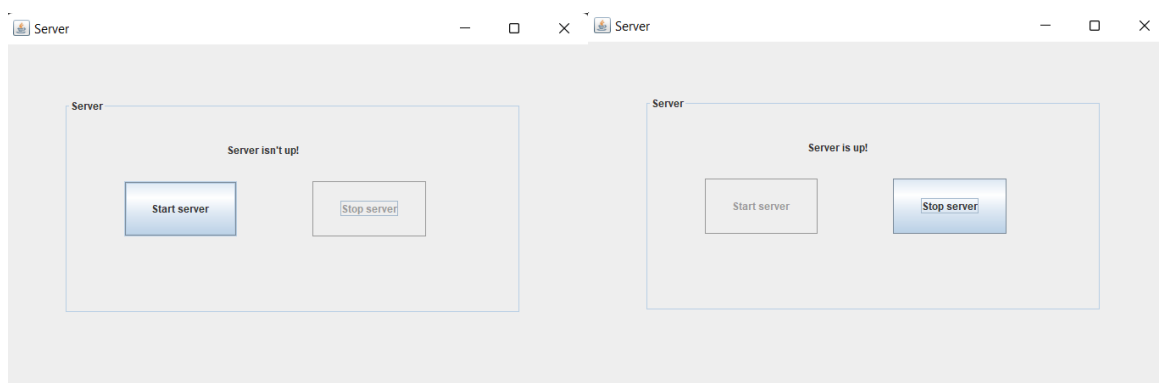


Слика 25. Кориснички интерфејс[1]

#### 4.3.1.1. Пројектовање екранских форми

Кориснички интерфејс састоји се од екранске форме и контролера корисничког интерфејса. Улоге екранске форме јесу да прихвата податке и догађаје које уноси односно прави актор, затим да позива контролера корисничког интерфејса и прослеђује му прихваћене податке и за крај да приказује податке које добије назад од контролера графичког интерфејса након извршене системске операције.

На серверској страни програма пројектована је корисничка форма која пре и после активације изгледа овако:



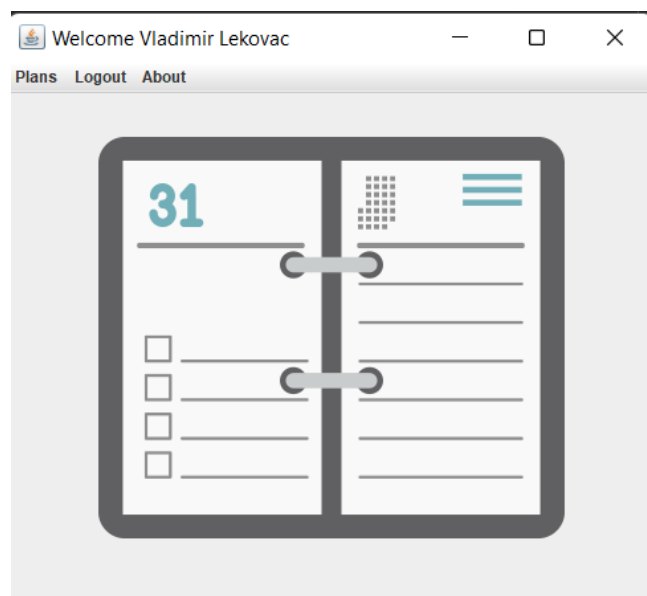
Изглед корисничког интерфејса клијентског дела апликације (форма за пријављивање корисника на систем, админ и студент):

The image displays two screenshots of a login application window titled "Login". The window has a standard Windows-style title bar with minimize, maximize, and close buttons.

**Top Screenshot:** The window shows the "Student" login form. At the top, there are three buttons: "Admin", "Log as admin or student", and "Student". The "Student" button is highlighted. Below these buttons is a form titled "Login - Student" containing two input fields labeled "Email:" and "Password:", and a "Login" button.

**Bottom Screenshot:** The window shows the "Admin" login form. The "Admin" button is highlighted. Below the buttons is a form titled "Login - Admin" containing two input fields labeled "Email:" and "Password:", and a "Login" button.

Изглед корисничког интерфејса клијентског дела апликације која се отвара након логовања (главна клијентска форма админа и студента):



Кроз случајеве коришћења пројектоване су и остале екранске форме које ће апликација поседовати, које се позивају из менија главне екранске форме клијентског дела апликације

## СК1: Случај коришћења – Пријављивање корисника

### Назив СК

Пријављивање на систем

### Актори СК:

Корисник

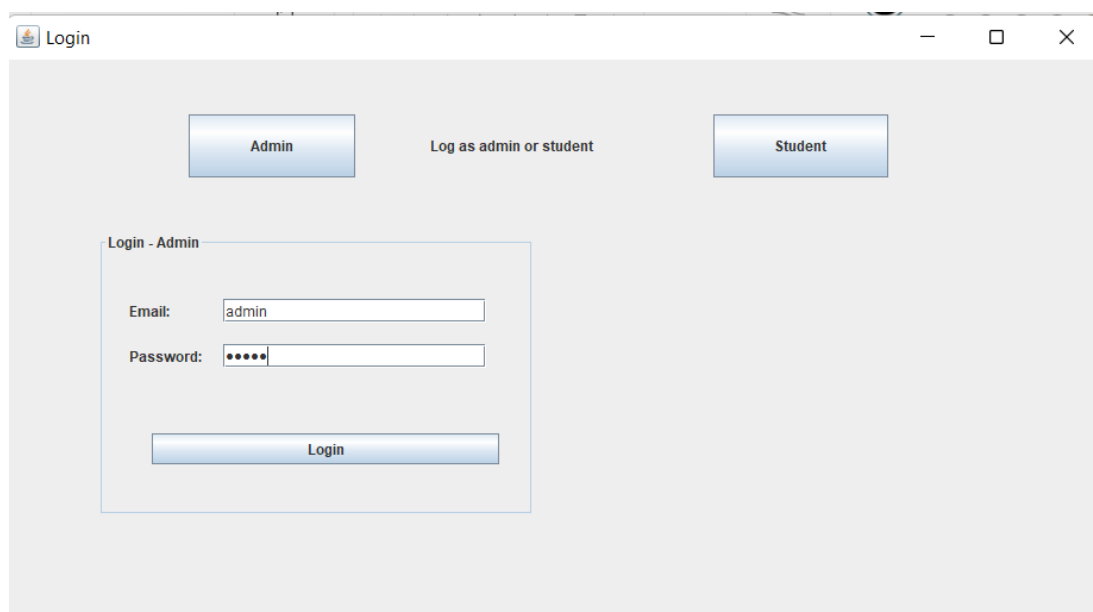
### Учесници СК:

Корисник и систем

**Предуслов:** Систем је укључен. Систем приказује форму за пријављивање на систем.

### Основни сценарио

1. Корисник уноси корисничко име и шифру. (АПУСО)



The screenshot shows a web browser window titled "Login". Inside the window, there are three buttons at the top: "Admin", "Log as admin or student", and "Student". Below these buttons is a section titled "Login - Admin". This section contains two input fields: "Email:" with the text "admin" and "Password:" with masked characters (dots). Below the input fields is a "Login" button.

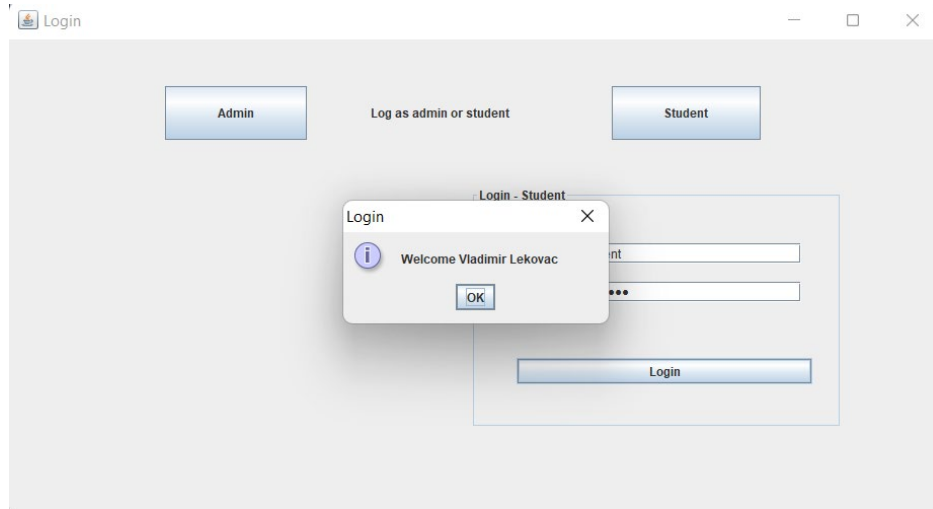
2. Корисник контролише да ли је коректно унео податке. (АНСО)

3. Корисник позива систем да изврши пријаву. (АПСО)

Опис акције: Админ/студент кликом на дугме „login “ позива системску операцију `izvrsiPrijavu(Korisnik)`

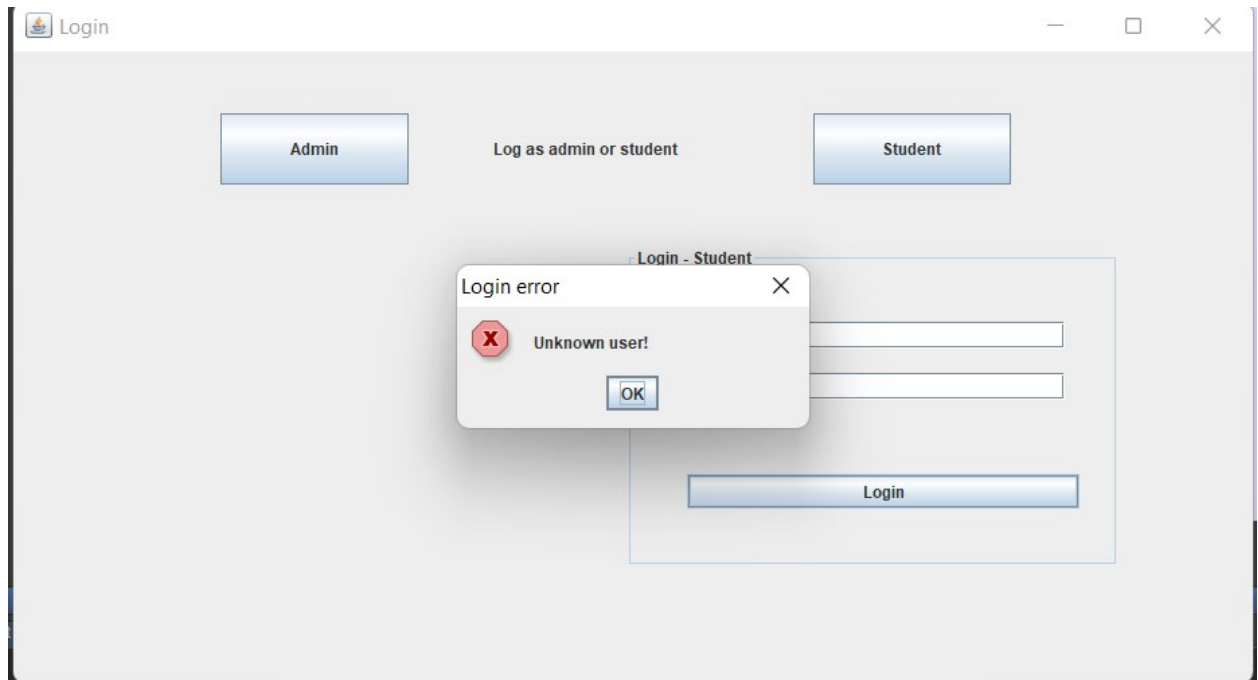
4.Систем пријављује корисника на систем. (СО)

5.Систем приказује кориснику поруку: “Пријављени сте на систем”. (ИА)



### Алтернативна сценарија

5.1 Уколико систем не може да пријави корисника, систем приказује кориснику поруку: “Неуспешна пријава на систем“ (ИА)



## СК2: Случај коришћења – Креирање новог предмета (Сложен случај коришћења)

### Назив СК

Креирање новог предмета

### Актори СК

Админ

### Учесници СК

Админ и систем (програм)

**Предуслов:** Систем је укључен и админ је улогован под својом шифром. Систем приказује форму за рад са предметом.

### Основни сценарио СК

1.Админ уноси податке у нови предмет. (АПУСО)

Working with subject

Name:  Semester:

Description:  ESPB:

Projects

Name	Description	Deadline	Max points
Plamen	2h	06.06.2021	30

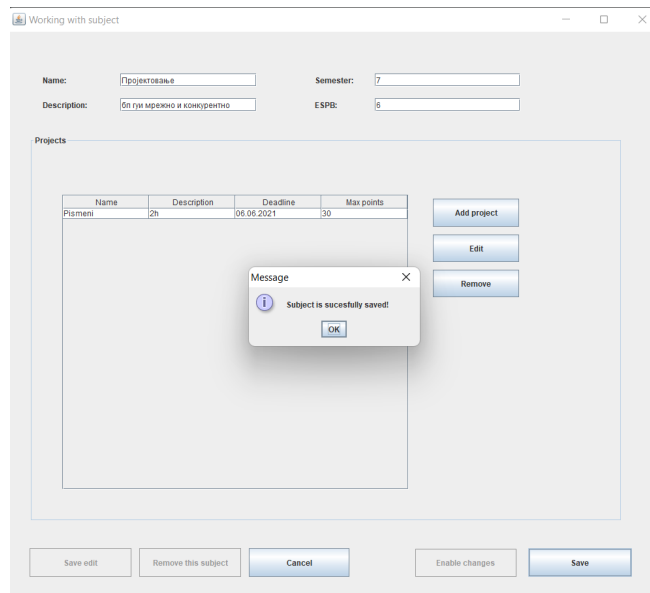
2.Админ контролише да ли је коректно унео податке у нови предмет. (АНСО)

3.Админ позива систем да креира нови предмет (АПСО)

Опис акције: Админ кликом на дугме „Save“ позива системску операцију kreirajPredmet(Predmet)

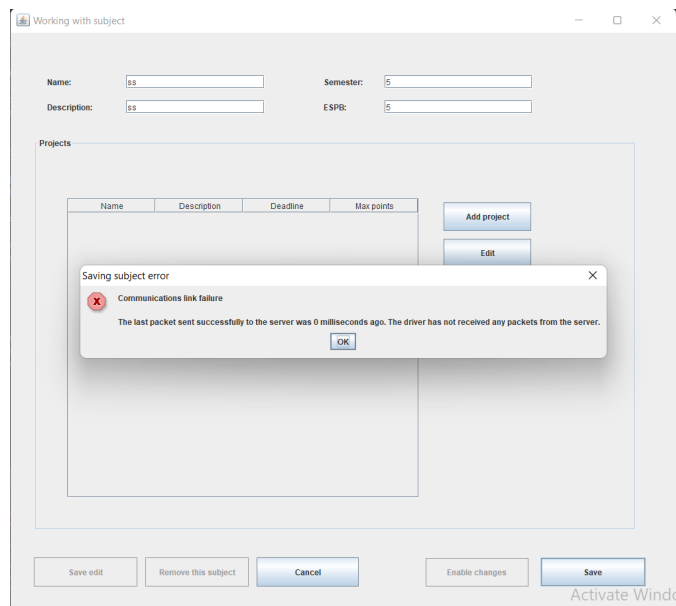
4.Систем креира нови предмет. (СО)

5.Систем приказује админу креирани предмет и поруку: “Систем је креирао нови предмет“. (ИА)



## Алтернативна сценарија

5.1 Уколико систем не може да запамти податке о предмету он приказује админу поруку “Систем не може да креира предмет”. (ИА)





### СКЗ: Случај коришћења – Претраживање предмета

#### Назив СК

Претраживање предмета

#### Актори СК

Админ

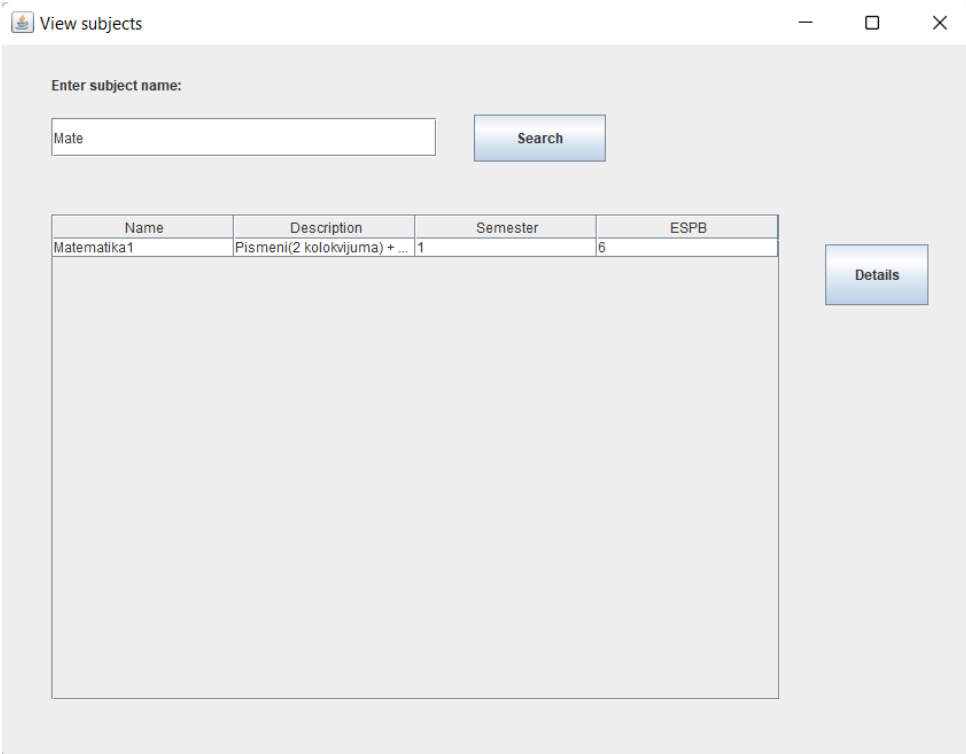
#### Учесници СК

Админ и систем (програм)

**Предуслов:** Систем је укључен и админ је улогован под својом шифром. Систем приказује форму за рад са предметом. Учитана је листа предмета.

#### Основни сценарио СК

1.Админ уноси критеријум по којем претражује предмете. (АПУСО)



The screenshot shows a web application window titled "View subjects". It contains a search form with a text input field labeled "Enter subject name:" containing the text "Mate", and a "Search" button. Below the search form is a table with the following data:

Name	Description	Semester	ESPB
Matematika1	Pismeni(2 kolokvijuma) + ...	1	6

To the right of the table is a "Details" button.

2.Админ позива систем да нађе предмете по задатом критеријуму. (АПСО)

Опис акције: Админ кликом на дугме „Search “ позива системску операцију `nadjiPredmete(Predmet, List<Predmet>)`

- 3.Систем тражи предмете по задатом критеријуму. (СО)
- 4.Систем обавештава админа о успешно извршеној претрази одговарајућом поруком и приказује пронађене предмете.(ИА)
- 5.Админ бира предмет који жели да прегледа.(АПУСО)
- 6.Админ позива систем да учита податке о одабраном предмету.(АПСО)

Опис акције: Админ кликом на дугме „Details“ позива системску операцију ucitajPredmet(Predmet)

- 7.Систем учитава податке о одабраном предмету.(СО)

Working with subject

Name:  Semester:

Description:  ESPB:

Projects

Name	Description	Deadline	Max points
Kolokvijum1	Radi se 2h. prvih 7 ne.	03.03.2020	25
Kolokvijum2	Radi se 2h. od 7-13 v.	06.03.2021	25
Usmeni	2 pitanja iz oba dela	07.02.2020	50
Dodatni Bodovi	Aktivnot na casu	07.03.2020	10
sd	sd	02.02.2021	233

Add project

Edit

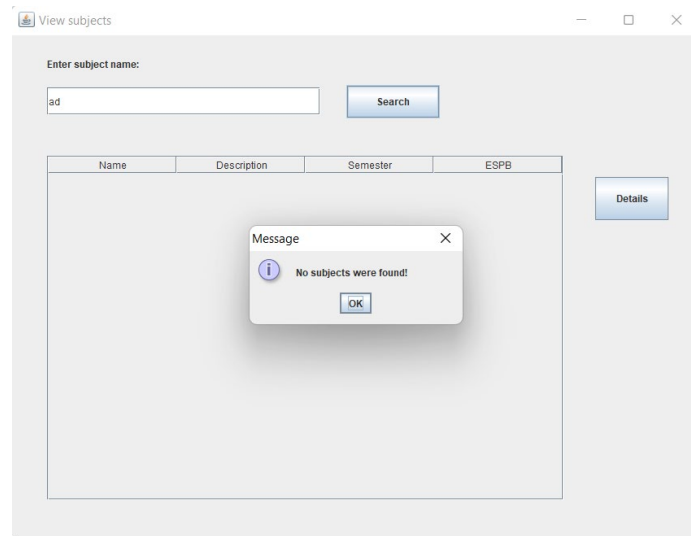
Remove

Save edit Remove this subject Cancel Enable changes Save

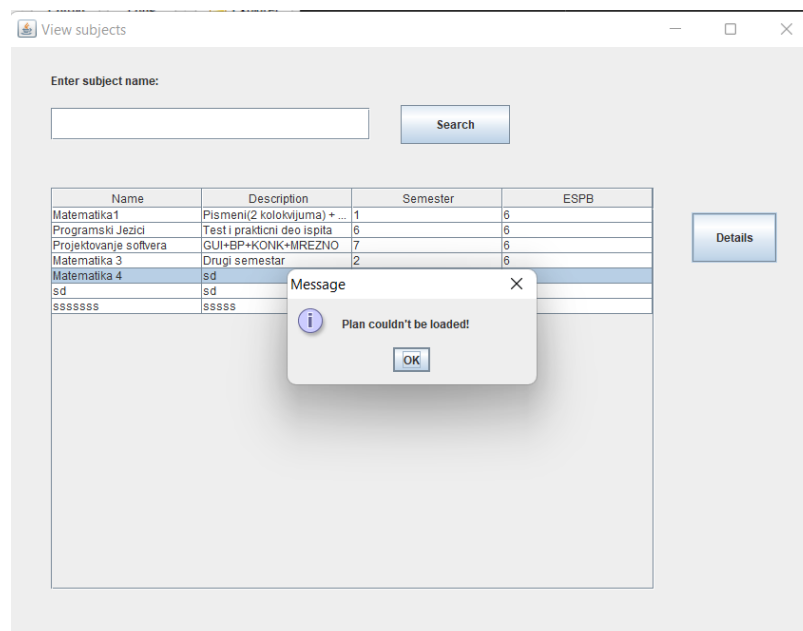
- 8.Систем обавештава админа о успешном учитавању података о предмету поруком “Одабрани предмет је приказан !” и приказује податке о одабраном предмету.(ИА)

## Алтернативна сценарија

4.1 Уколико систем не може да нађе предмете он приказује админу поруку: “Систем не може да нађе предмете по задатом критеријуму“. Прекида се извршавање сценарија. (ИА)



8.1 Уколико систем не може да учита изабрани предмет он приказује админу поруку „Систем не може да учита изабрани предмет.“ Прекида се извршење сценарија.(ИА)



## СК4: Случај коришћења – Измена предмета(Сложен случај коришћења)

### Назив СК

Промена предмета

### Актори СК

Админ

### Учесници СК

Админ и систем (програм)

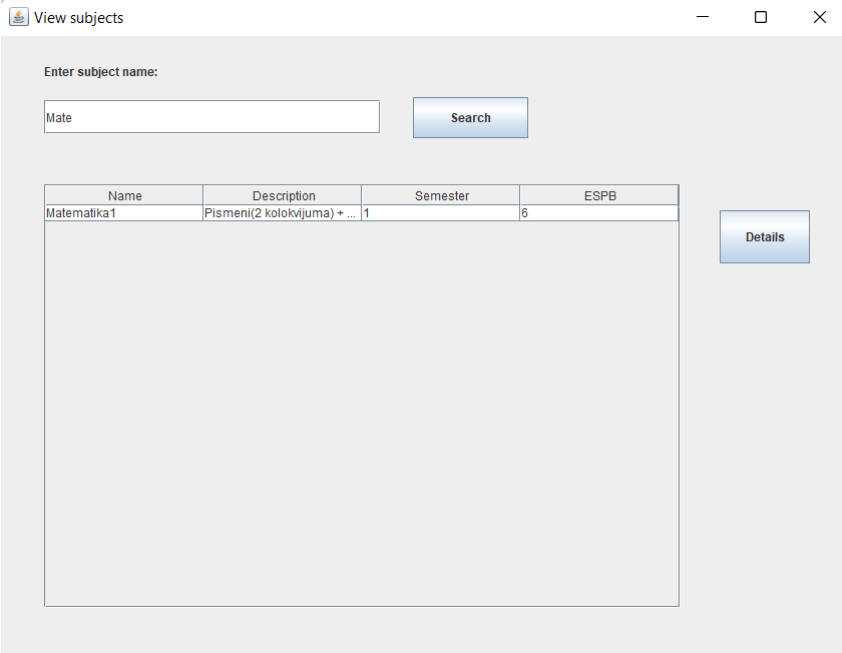
**Предуслов:** Систем је укључен и админ је улогован под својом шифром. Систем приказује форму за рад са предметом. Учитана је листа предмета.

### Основни сценарио СК

- 1.Админ уноси критеријум по којем претражује предмете. (АПУСО)
- 2.Админ позива систем да нађе предмете по задатом критеријуму. (АПСО)

Опис акције: Админ кликом на дугме „Search “ позива системску операцију `nađjiPredmete(Predmet, List<Predmet>)`

- 3.Систем тражи предмете по задатом критеријуму. (СО)
- 4.Систем обавештава админа о успешно извршеној претрази одговарајућом поруком и приказује пронађене предмете.(ИА)



The screenshot shows a window titled "View subjects" with a search interface. At the top, there is a label "Enter subject name:" followed by a text input field containing the word "Mate" and a "Search" button. Below the search area is a table with four columns: "Name", "Description", "Semester", and "ESPB". The table contains one row with the following data: "Matematika1", "Pismeni(2 kolokvijuma) + ...", "1", and "6". To the right of the table is a "Details" button.

Name	Description	Semester	ESPB
Matematika1	Pismeni(2 kolokvijuma) + ...	1	6

5.Админ бира предмет који жели да прегледа.(АПУСО)

6.Админ позива систем да учита податке о одабраном предмету.(АПСО)

Опис акције: Админ кликом на дугме „Details“ позива системску операцију ucitajPredmet(Predmet)

7.Систем учитава податке о одабраном предмету.(СО)

8.Систем обавештава админа о успешном учитавању података о предмету поруком “Одабрани предмет је приказан !“ и приказује податке о одабраном предмету.(ИА)

9.Админ уноси (мења) податке о предмету. (АПУСО)

The screenshot shows a web application window titled "Working with subject". It contains several input fields for subject information:

- Name: Matematika1
- Semester: 1
- Description: Pismeni(2 kolokijuma) + Usmeni
- ESPB: 6

Below these fields is a section titled "Projects" containing a table with the following data:

Name	Description	Deadline	Max points
Kolokijum1	Radi se 2h, pish 7 na	01.03.2020	25
Kolokijum2	Radi se 2h, od 7-13 v.	06.03.2021	25
Usmeni	2 pitanja iz oba dela	07.02.2020	50
Dodatni Bodovi	Radovi na case	07.03.2020	10
sd	sd	02.02.2021	233

To the right of the table are three buttons: "Add project", "Edit", and "Remove". At the bottom of the window are five buttons: "Save edit", "Remove this subject", "Cancel", "Enable changes", and "Save".

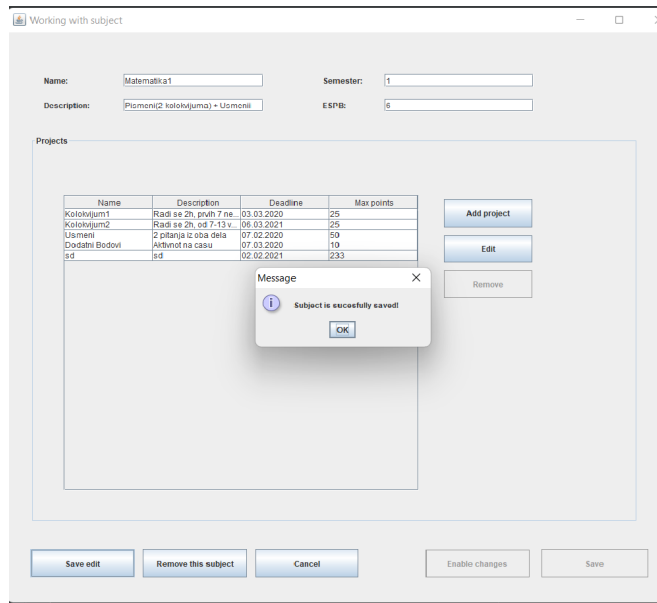
10.Админ контролише да ли је коректно унео податке о предмету. (АНСО)

11.Админ позива систем да запамти податке о предмету. (АПСО)

Опис акције: Админ кликом на дугме „Save edit “ позива системску операцију zapamtiPredmet(Predmet)

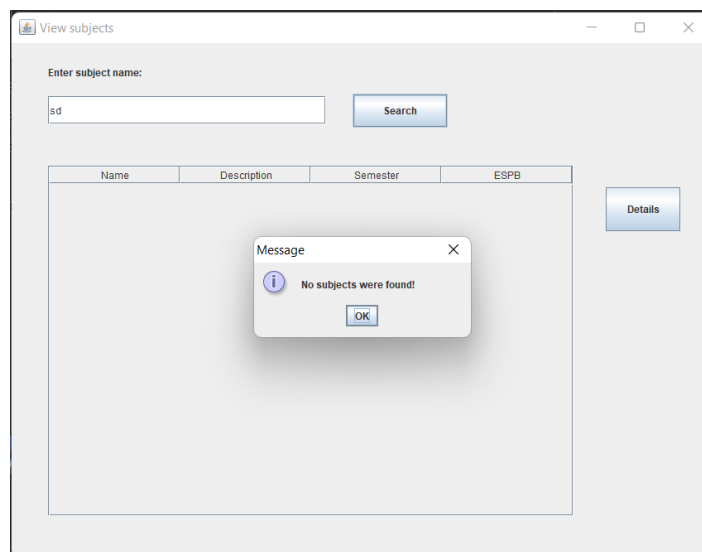
12.Систем памти податке о предмету. (СО)

13.Систем приказује админу запамћени предмет и поруку: “Систем је запамтио предмет.” (ИА)

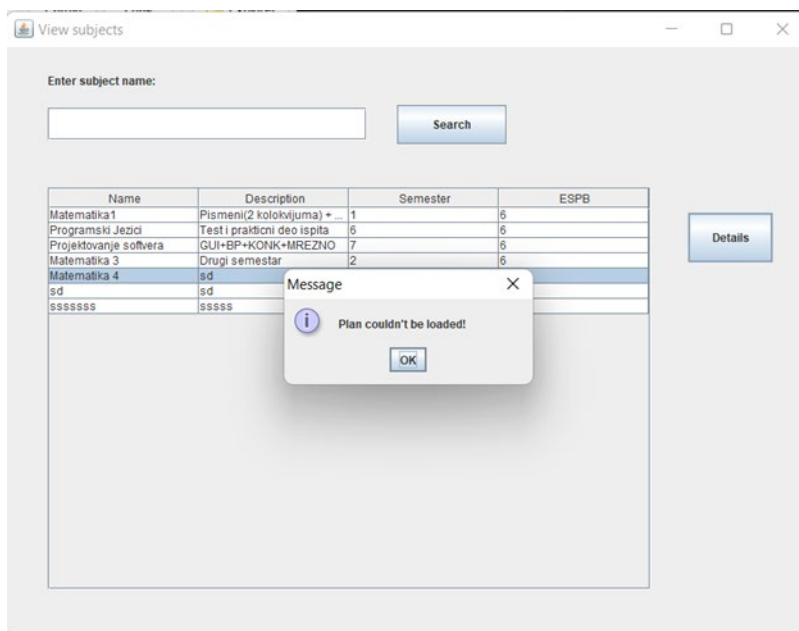


## Алтернативна сценарија

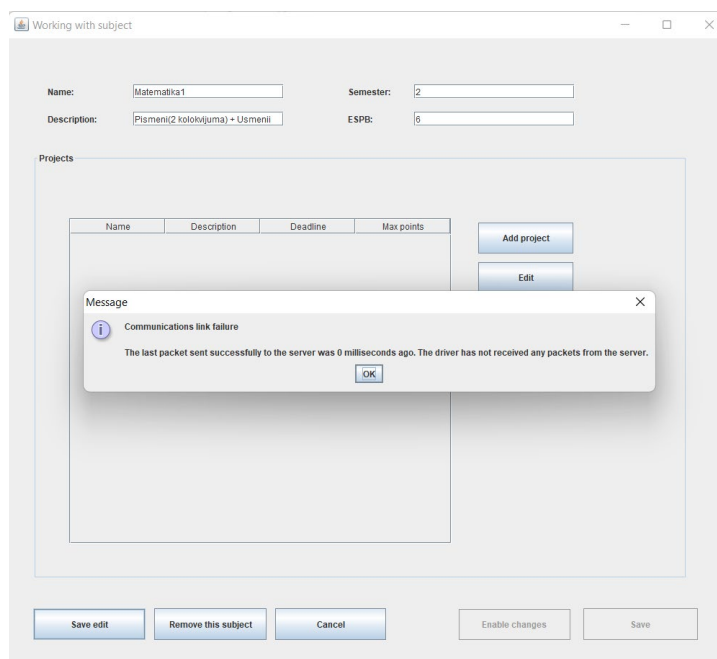
4.1 Уколико систем не може да нађе предмете он приказује админу поруку: “Систем не може да нађе предмете по задатом критеријуму”. Прекида се извршење сценарија. (ИА)



8.1 Уколико систем не може да учита изабрани предмет он приказује админу поруку „Систем не може да учита изабрани предмет.“ Прекида се извршење сценарија.(ИА)



13.1 Уколико систем не може да запамти податке о предмету он приказује админу поруку “Систем не може да запамти предмет”. Прекида се извршење сценарија. (ИА)



## СК5: Случај коришћења – Брисање предмета

### Назив СК

Брисање предмета

### Актори СК

Админ

### Учесници СК

Админ и систем (програм)

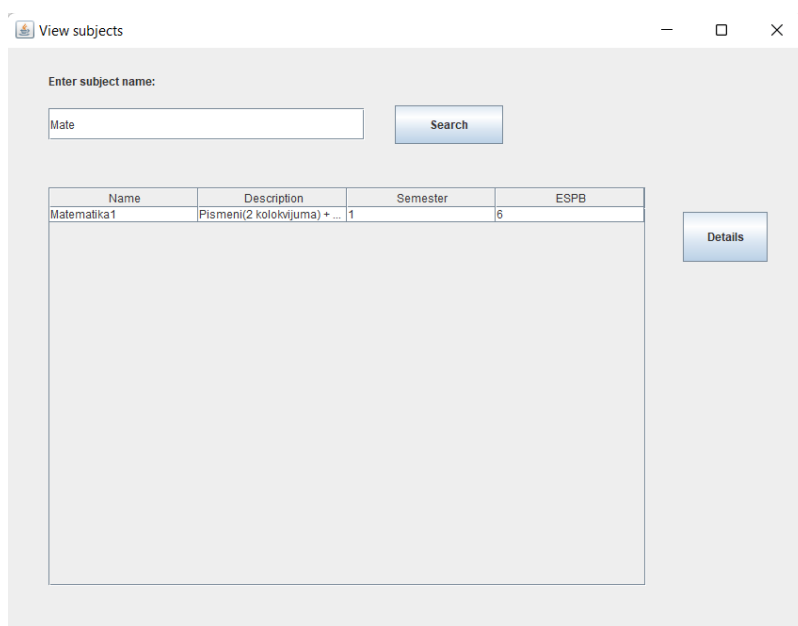
**Предуслов:** Систем је укључен и админ је улогован под својом шифром. Систем приказује форму за рад са предметом. Учитана је листа предмета.

### Основни сценарио СК

- 1.Админ уноси критеријум по којем претражује предмете. (АПУСО)
- 2.Админ позива систем да нађе предмете по задатом критеријуму. (АПСО)

Опис акције: Админ кликом на дугме „Search “ позива системску операцију `nađjiPredmete(Predmet, List<Predmet>)`

- 3.Систем тражи предмете по задатом критеријуму. (СО)
- 4.Систем обавештава админа о успешно извршеној претрази одговарајућом поруком и приказује пронађене предмете.(ИА)



The screenshot shows a window titled "View subjects" with a search interface. At the top, there is a label "Enter subject name:" followed by a text input field containing the word "Male" and a "Search" button. Below the search area is a table with four columns: "Name", "Description", "Semester", and "ESPB". The table contains one row with the following data: "Matematika 1", "Pismeni(2 kolokvijuma) + ...", "1", and "6". To the right of the table is a "Details" button.

Name	Description	Semester	ESPB
Matematika 1	Pismeni(2 kolokvijuma) + ...	1	6



5.Админ бира предмет који жели да прегледа.(АПУСО)

6.Админ позива систем да учита податке о одабраном предмету.(АПСО)

Опис акције: Админ кликом на дугме „Details“ позива системску операцију ucitajPredmet(Predmet)

7.Систем учитава податке о одабраном предмету.(СО)

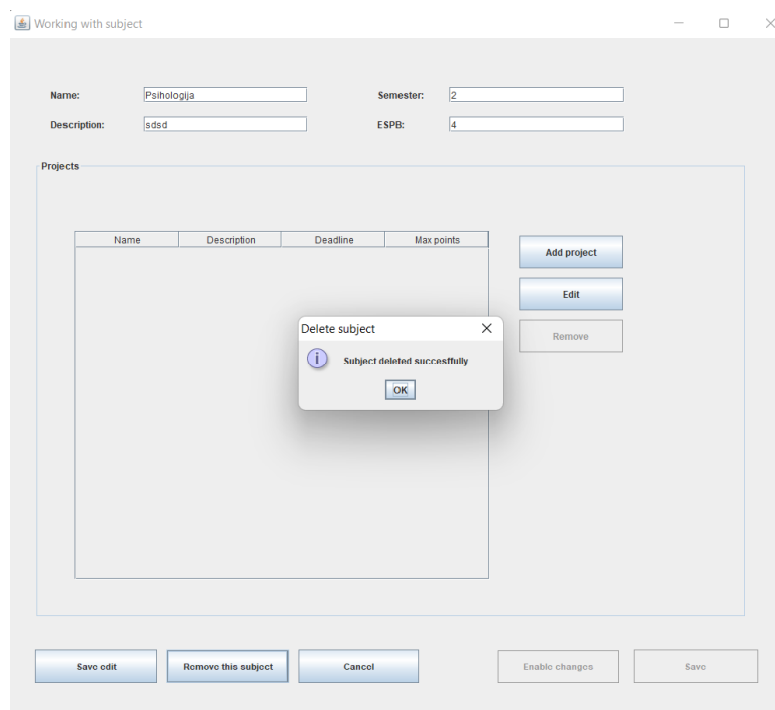
8.Систем обавештава админа о успешном учитавању података о предмету поруком “Одабрани предмет је приказан !“ и приказује податке о одабраном предмету.(ИА)

9.Админ позива систем да обрише предмет. (АПСО)

Опис акције: Админ кликом на дугме „Remove this subject “ позива системску операцију obrisiPredmet(Predmet)

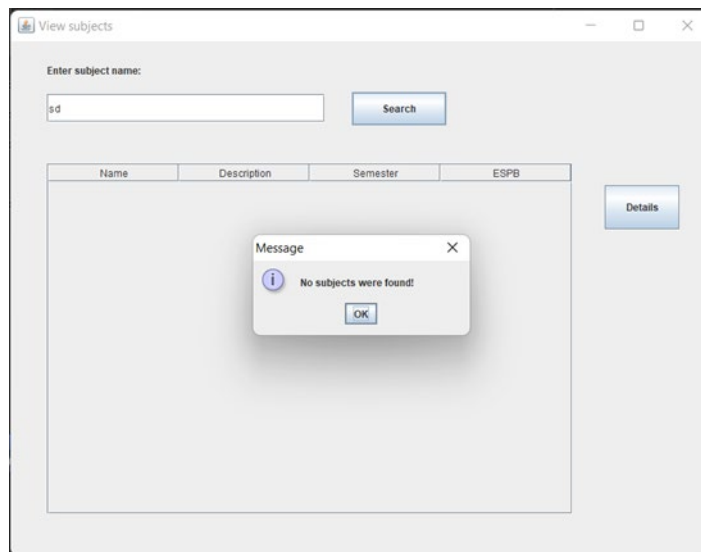
10.Систем брише предмет. (СО)

11.Систем приказује админу поруку: “Систем је обрисао предмет.” (ИА)

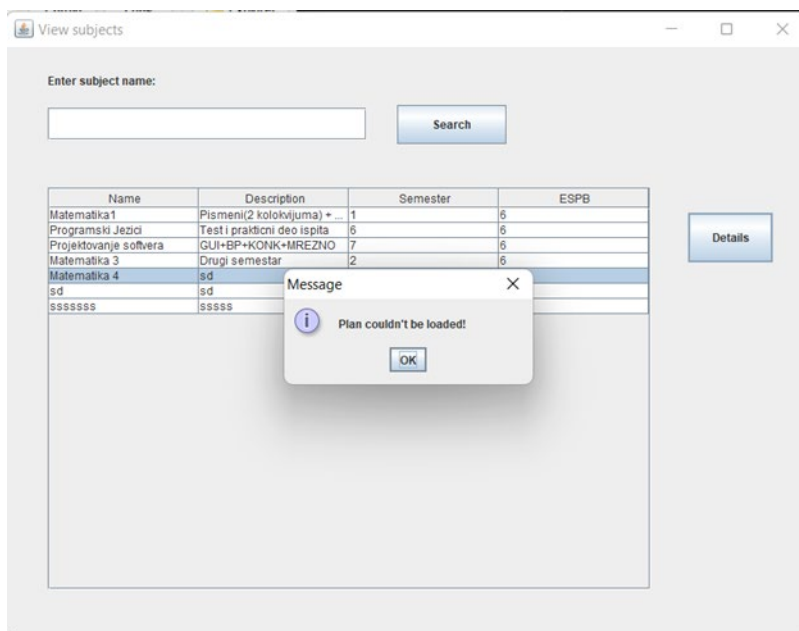


## Алтернативна сценарија

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



8.1 Уколико систем не може да учита изабрани предмет он приказује админу поруку „Систем не може да учита изабрани предмет.“ Прекида се извршење сценарија.(ИА)



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

Working with subject

Name: 
Semester:

Description: 
ESPБ:

Projects

Name	Description	Deadline	Max points
Kolokvijum1	Radi se 2h, prvih 7 ne.	03.03.2020	25
Kolokvijum2	Radi se 2h, od 7-13v.	06.03.2021	25
Usmeni	2 pitanja iz oba dela	07.02.2020	50
Dodatni Bodovi	Aktivnot na casu	07.03.2020	10
sd	sd	02.02.2021	233

Add project
Edit

te subject

Error deleting subject!

Cannot delete or update a parent row: a foreign key constraint fails ('seminarski', 'plan', CONSTRAINT 'fk\_subject' FOREIGN KEY ('subjectID') REFERENCES 'subject' ('subjectID') ON UPDATE NO ACTION)

OK

Save edit

Remove this subject

Cancel

Enable changes

Save

## СК6: Случај коришћења – Креирање новог плана(Сложен случај коришћења)

### Назив СК

Креирање новог предмета

### Актори СК

Студент

### Учесници СК

Студент и систем (програм)

**Предуслов:** Систем је укључен и студент је улогован под својом шифром. Систем приказује форму за рад са планом. Учитана је листа предмета.

### Основни сценарио СК

1.Студент уноси податке у нови план. (АПУСО)

Working with plan

Subject:  Choose

Plan Items

Project	Comment	Deadline	Completed
Drugi kolokvijum	Научи гуи	12.12.2021	<input type="checkbox"/>

Add new item

Remove

Save edit Delete plan Cancel Enable Changes Save

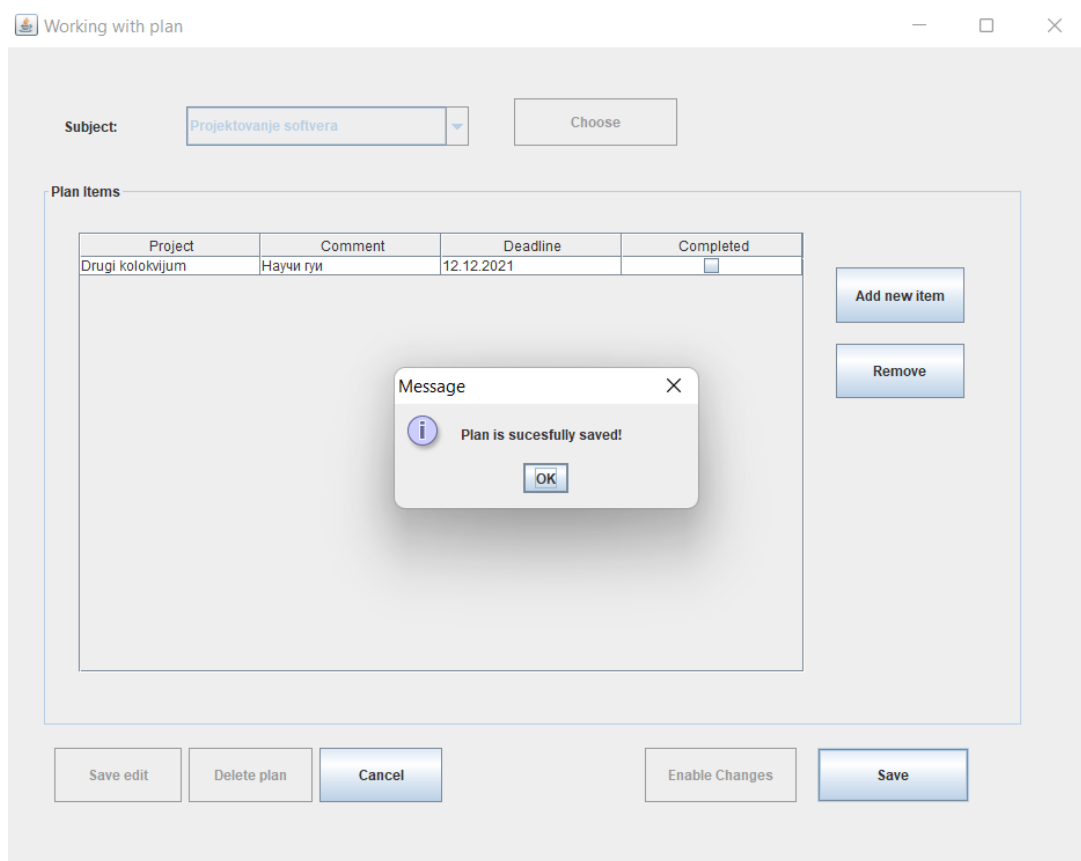
2.Студент контролише да ли је коректно унео податке у нови план. (АНСО)

3.Студент позива систем да креира нови план. (АПСО)

Опис акције: Студент кликом на дугме „Save“ позива системску операцију kreirajPlan(Plan)

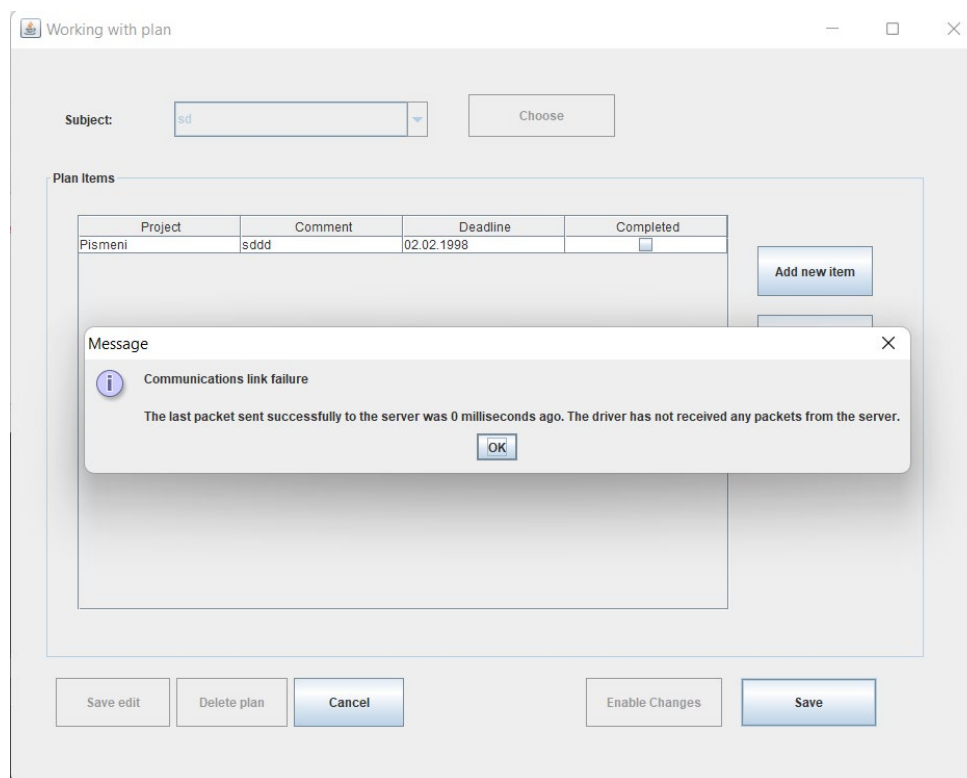
4.Систем креира нови план. (СО)

5.Систем приказује студенту креирани план и поруку: “Систем је креирао нови план“. (ИА)



## Алтернативна сценарија

5.1 Уколико систем не може да запамти податке о плану он приказује студенту поруку “Систем не може да креира нови план”. (ИА)



## СК7: Случај коришћења – Претраживање плана

### Назив СК

Претраживање предмета

### Актори СК

Студент

### Учесници СК

Студент и систем (програм)

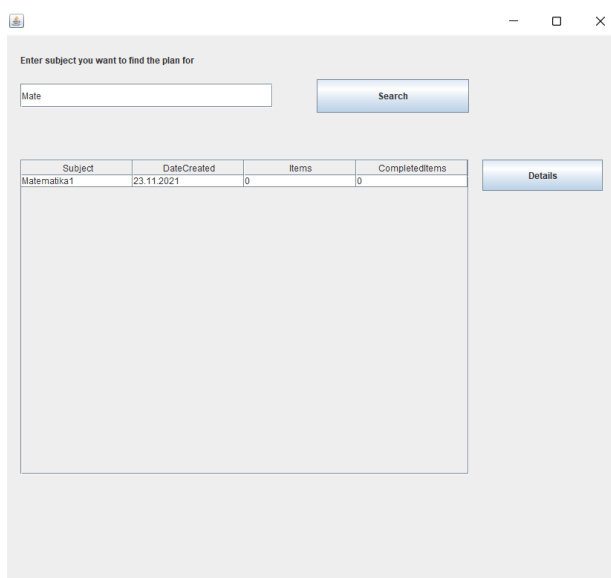
**Предуслов:** Систем је укључен и студент је улогован под својом шифром. Систем приказује форму за рад са планом. Учитана је листа планова.

### Основни сценарио СК

- 1.Студент уноси критеријум по којем претражује планове. (АПУСО)
- 2.Студент позива систем да нађе планове по задатом критеријуму. (АПСО)

Опис акције: Студент кликом на дугме „Search“ позива системску операцију `nađjiPlanove(Plan, List<Plan>)`

- 3.Систем тражи планове по задатом критеријуму. (СО)
- 4.Систем обавештава студента о успешно извршеној претрази одговарајућом поруком и приказује пронађене планове.(ИА)



Subject	DateCreated	Items	CompletedItems
Matematika 1	23.11.2021	0	0

- 5.Студент бира план који жели да прегледа.(АПУСО)

6.Студент позива систем да учита податке о одабраном плану.(АПСО)

Опис акције: Студент кликом на дугме „Details“ позива системску операцију ucitajPlan(Plan)

7.Систем учитава податке о одабраном плану.(СО)

8.Систем обавештава студента о успешном учитавању података о предмету поруком “Одабрани план је приказан !” и приказује податке о одабраном плану.(ИА)

Working with plan

Subject: Matematika1 Choose

Plan Items

Project	Comment	Deadline	Completed
---------	---------	----------	-----------

Add new item  
Remove

Save edit Delete plan Cancel Enable Changes Save

## Алтернативна сценарија

4.1 Уколико систем не може да нађе планове он приказује студенту поруку: “Систем не може да нађе планове по задатом критеријуму“. Прекида се извршавање сценарија.(ИА)

Enter subject you want to find the plan for

sdds Search

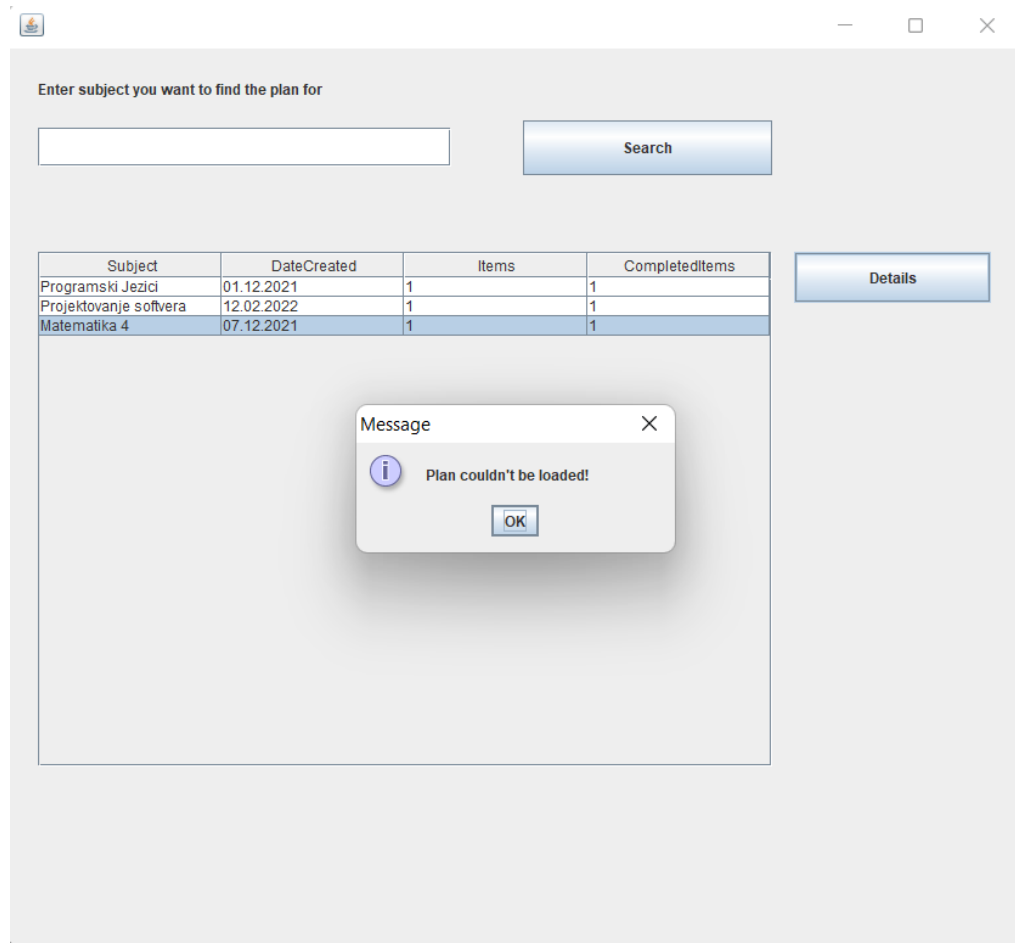
Subject	DateCreated	Items	CompletedItems
Matematika1	23.11.2021	0	0

Details

Message  
No plans were found!  
OK



8.1 Уколико систем не може да учита изабрани план он приказује студенту поруку „Систем не може да учита изабрани план.“ Прекида се извршење сценарија.(ИА)



## СК8: Случај коришћења – Измена плана (Сложен случај коришћења)

### Назив СК

Промена предмета

### Актори СК

Студент

### Учесници СК

Студент и систем (програм)

**Предуслов:** Систем је укључен и студент је улогован под својом шифром. Систем приказује форму за рад са планом. Учитана је листа планова. Учитана је листа предмета.

### Основни сценарио СК

1. Студент уноси критеријум по којем претражује планове. (АПУСО)
2. Студент позива систем да нађе планове по задатом критеријуму. (АПСО)

Опис акције: Студент кликом на дугме „Search“ позива системску операцију `nadjiPlanove(Plan, List<Plan>)`

3. Систем тражи планове по задатом критеријуму. (СО)
4. Систем обавештава студента о успешно извршеној претрази одговарајућом поруком и приказује пронађене планове. (ИА)

Subject	DateCreated	Items	CompletedItems
Matematika 1	23.11.2021	0	0

5. Студент бира план који жели да прегледа. (АПУСО)
6. Студент позива систем да учита податке о одабраном плану. (АПСО)

Опис акције: Студент кликом на дугме „Details“ позива системску операцију ucitajPlan(Plan)

7.Систем учитава податке о одабраном плану.(СО)

8.Систем обавештава студента о успешном учитавању података о плану поруком “Одабрани план је приказан !” и приказује податке о одабраном плану.(ИА)

Project	Comment	Deadline	Completed
Prt test	dca	02.02.1997	<input checked="" type="checkbox"/>

9.Студент уноси (мења) податке о плану. (АПУСО)

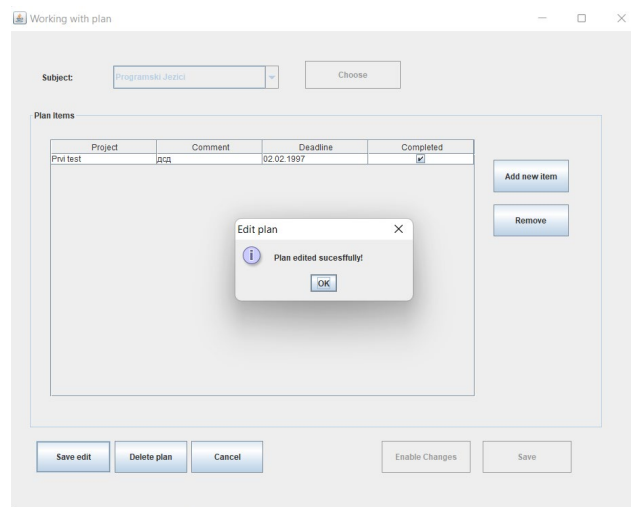
10.Студент контролише да ли је коректно унео податке о плану. (АНСО)

11.Студент позива систем да запамти податке о плану. (АПСО)

Опис акције: Студент кликом на дугме „Save edit“ позива системску операцију zapamtiPlan(Plan)

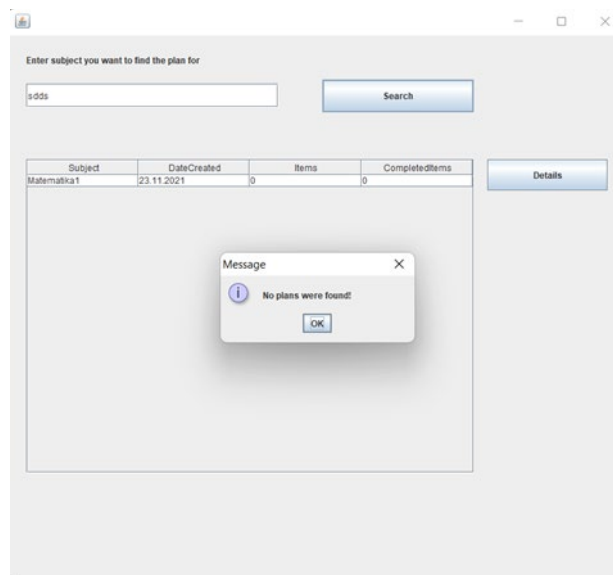
12.Систем памти податке о плану. (СО)

13.Систем приказује студенту запамћени план и поруку: “Систем је запамтио план.(ИА)

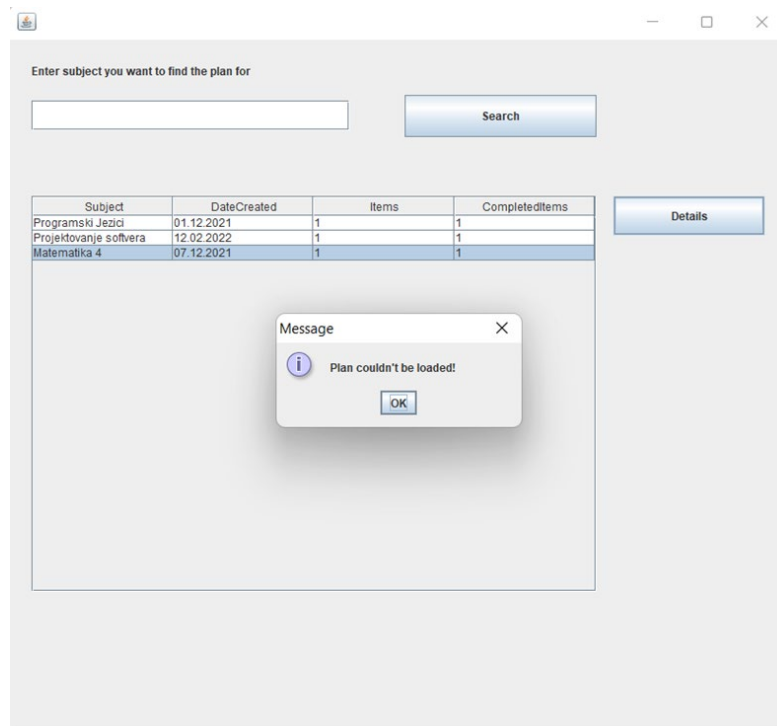


## Алтернативна сценарија

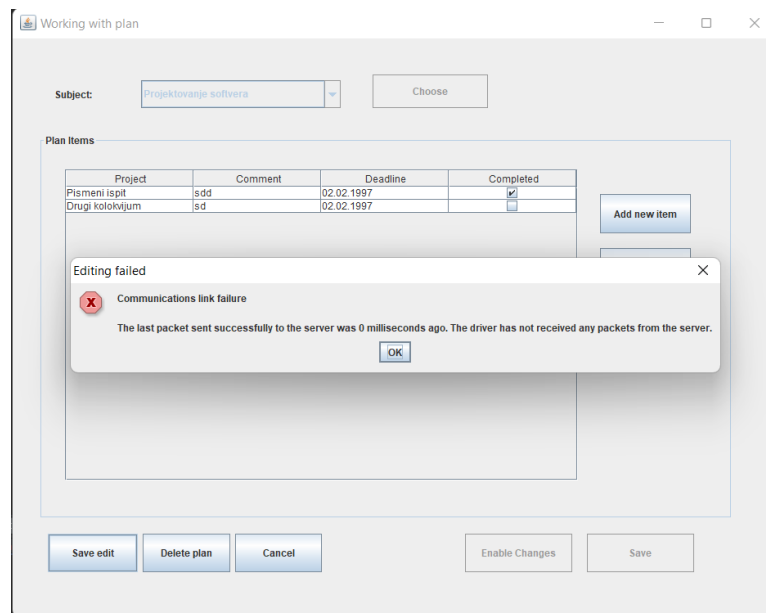
4.1 Уколико систем не може да нађе планове он приказује студенту поруку: “Систем не може да нађе планове по задатом критеријуму”. Прекида се извршење сценарија. (ИА)



8.1 Уколико систем не може да учита изабрани план он приказује студенту поруку „Систем не може да учита изабрани план.“ Прекида се извршење сценарија.(ИА)



13.1 Уколико систем не може да запамти податке о плану он приказује студенту поруку “Систем не може да запамти план”. Прекида се извршење сценарија. (ИА)



## СК9: Случај коришћења – Брисање плана

### Назив СК

Брисање предмета

### Актори СК

Студент

### Учесници СК

Студент и систем (програм)

**Предуслов:** Систем је укључен и студент је улогован под својом шифром. Систем приказује форму за рад са планом. Учитана је листа планова.

### Основни сценарио СК

- 1.Студент уноси критеријум по којем претражује планове. (АПУСО)
- 2.Студент позива систем да нађе планове по задатом критеријуму. (АПСО)

Опис акције: Студент кликом на дугме „Search“ позива системску операцију `nađjiPlanove(Plan, List<Plan>)`

- 3.Систем тражи планове по задатом критеријуму. (СО)
- 4.Систем обавештава студента о успешно извршеној претрази одговарајућом поруком и приказује пронађене планове.(ИА)

Subject	DateCreated	Items	CompletedItems
Matematika1	23.11.2021	0	0

- 5.Студент бира план који жели да прегледа.(АПУСО)
  - 6.Студент позива систем да учита податке о одабраном плану.(АПСО)
- Опис акције: Студент кликом на дугме „Details“ позива системску операцију `ucitajPlan(Plan)`

7.Систем учитава податке о одабраном плану.(СО)

8.Систем обавештава студента о успешном учитавању података о плану поруком  
“Одабрани план је приказан !“ и приказује податке о одабраном плану.(ИА)

The screenshot shows a window titled "Working with plan". At the top, there is a "Subject:" label followed by a dropdown menu currently showing "Matematika1" and a "Choose" button. Below this is a section titled "Plan Items" containing a table with the following data:

Project	Comment	Deadline	Completed
Kolokvijum1	Nauci prvi zadatak	02.02.2021	<input type="checkbox"/>

To the right of the table are two buttons: "Add new item" and "Remove". At the bottom of the window, there are four buttons: "Save edit", "Delete plan", "Cancel", and "Enable Changes". A "Save" button is also present on the far right.

9.Студент позива систем да обрише план. (АПСО)

Опис акције: Студент кликом на дугме „Delete Plan“ позива системску операцију obrisiPlan(Plan)

10.Систем брише план. (СО)

11.Систем приказује кориснику поруку: “Систем је обрисао план.” (ИА)

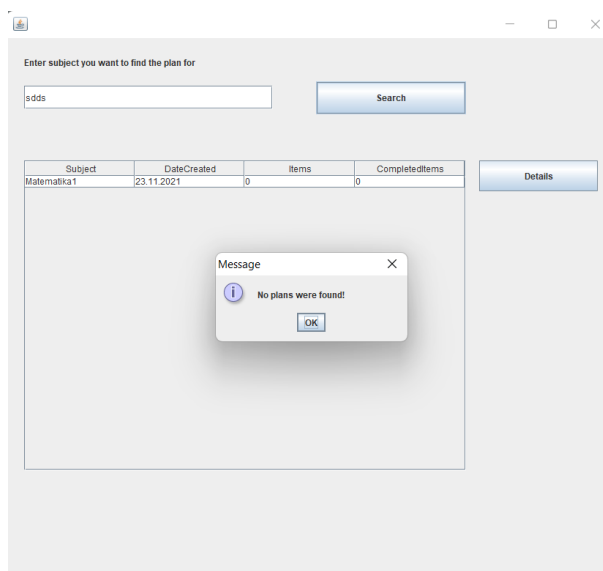
This screenshot shows the same "Working with plan" window, but with the "Subject" dropdown set to "Projektovanje softvera". The "Plan Items" table now contains two rows:

Project	Comment	Deadline	Completed
Pismeni ispit	sdd	02.02.1997	<input checked="" type="checkbox"/>
Drugi kolokvijum	sd	02.02.1997	<input type="checkbox"/>

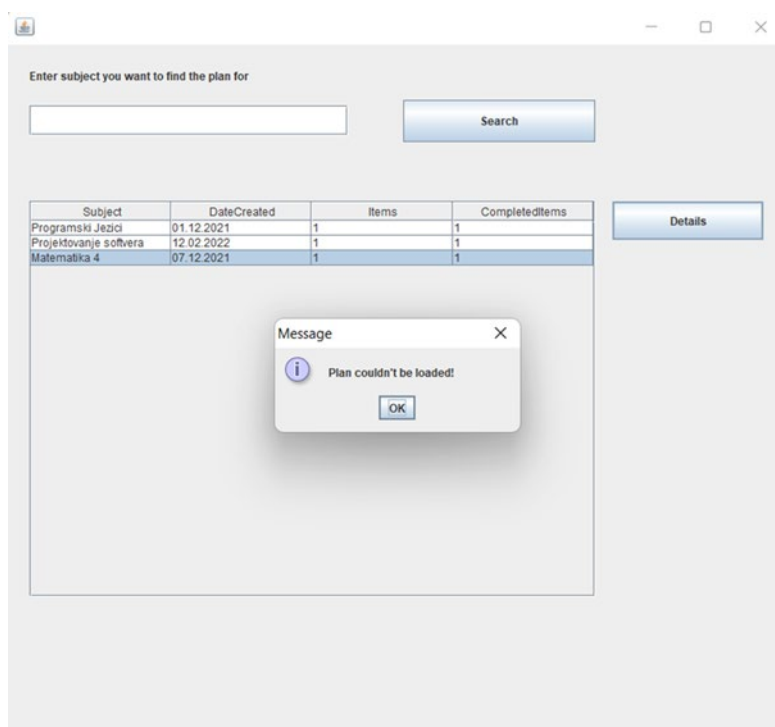
A modal dialog box titled "Delete plan" is centered over the table. It contains an information icon, the text "Plan deleted successfully", and an "OK" button. The "Delete plan" button from the main window is visible at the bottom.

## Алтернативна сценарија

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

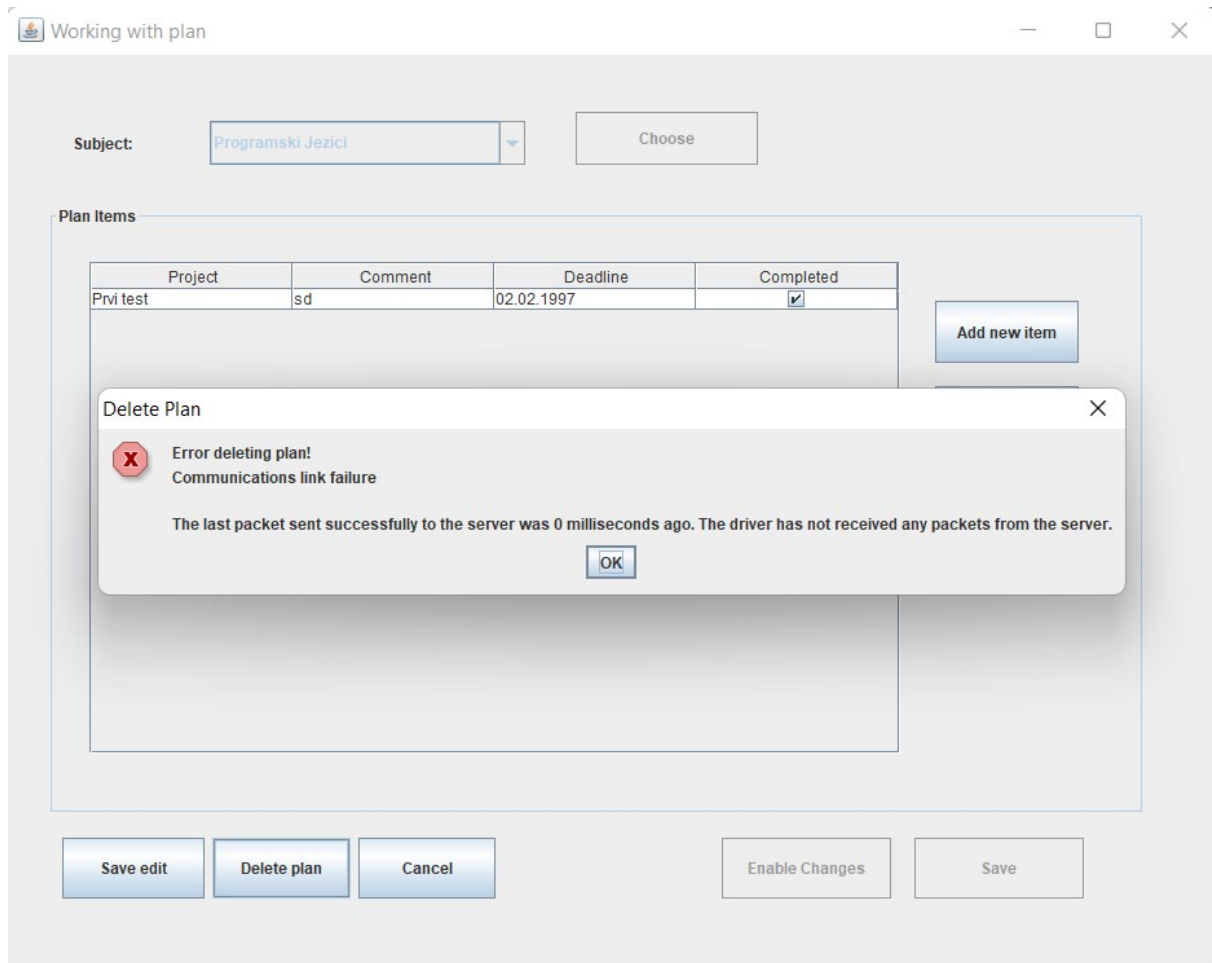


8.1 Уколико систем не може да учита изабрани план он приказује кориснику поруку „Систем не може да учита изабрани план.“ Прекида се извршење сценарија.(ИА)





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



#### 4.3.1.2. Контролер корисничког интерфејса

Контролер корисничког интерфејса састоји се од :

- општег дела који је независан од екранских форми и чија је улога комуникација са серверском страном ради извршења системских операција (Општи контролер)
- конкретног дела који је везан за домен екранске форме (Конкретни контролер)

Општи контролер корисничког интерфејса је задужен за:

- Успостављање везе између екранске форме и апликационе логике.
- Прихватање захтева за извршење системске операције.
- Креирање доменских објеката.
- Прослеђивање захтева за извршење системске операције и доменске објекте до апликационог сервера.
- Прихватање доменских објеката и сигнала које враћа апликациони сервер као резултат извршења системске операције.

Конкретни контролер корисничког интерфејса је задужен за:

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

#### 4.3.2. Пројектовање апликационе логике

Апликациона логика налази се на страни сервера. Апликациони сервери обезбеђују сервисе који омогућавају реализацију апликационе логике софтверског система.

Апликациони сервер садржи:

- Део у оквиру којег се врши комуникација са клијентима.
- Контролера апликационе логике.
- Брокер који врши комуникацију са складиштем података.
- Део који садржи пословну логику.

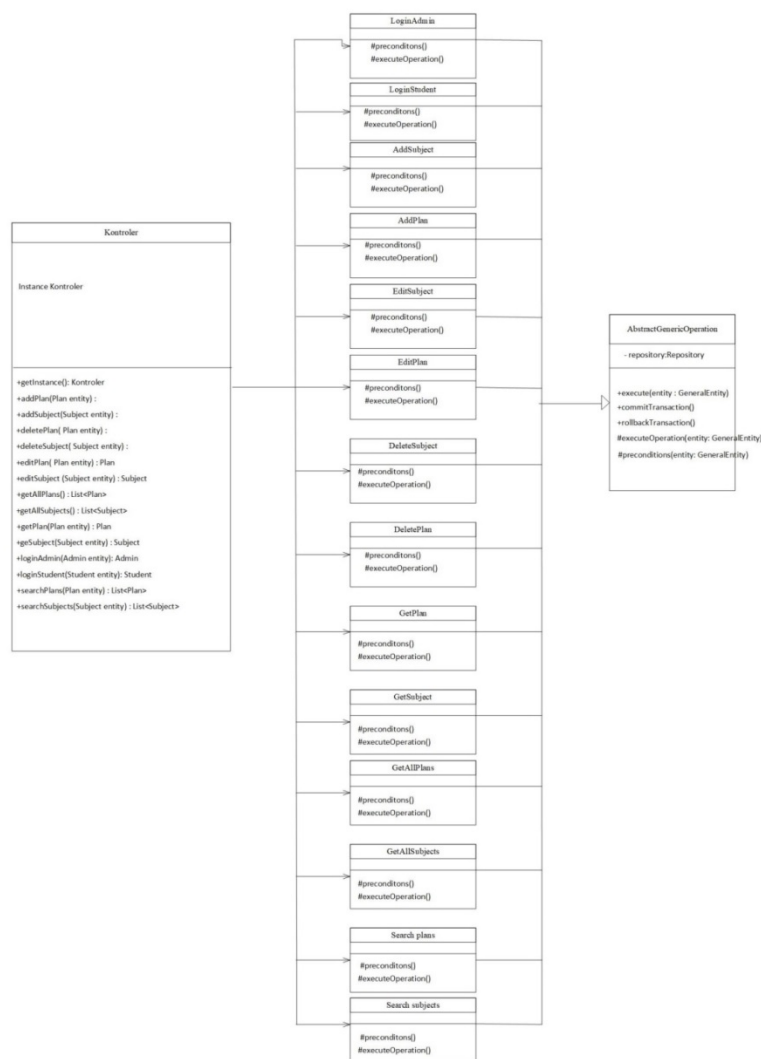
##### 4.3.2.1. Комуникација са клијентима

Када се покрене сервер, аутоматски се покреће серверска нит. У оквиру серверске нити инстанцира се серверски сокет, који је задужен за примање клијената. Приликом

прихватања клијента, сервер генерише сокет који се повезује са клијентском сокетом. Тај генерисани сокет прослеђује се као улазни параметар приликом инстанцирања нити која ће бити задужена за обраду захтева тог конкретног клијента. Сходно томе, са сваким новим клијентом, генерише се нова нит, чиме је омогућено да сервер ради са више клијената.

#### 4.3.2.2. Контролер апликационе логике

Приликом слања захтева, корисник позива сервер да изврши одређену системску операцију. Сервер прима тај захтев у оквиру нити која је повезана са датим клијентом преко сокета. Захтев се затим прослеђује контролеру који га прослеђује класама које су задужене за обављање системских операција. Након извршења системске операције резултат се враћа до апликационе логике, која тај резултат шаље назад до клијента.



Слика 26. Контролер апликационе логике позива системске операције

#### 4.3.2.3. Пословна логика

##### Пројектовање понашања софтверског система

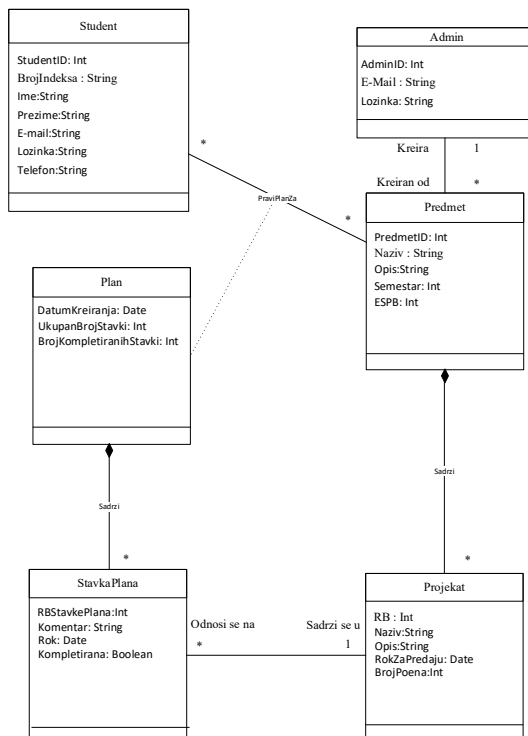
У ранијим фазама дефинисано је следећих четрнаест системских операција:

- 1) Signal IzvrsiPrijavu(Korisnik)
- 2) Signal ZapamtiPredmet(Predmet)
- 3) Signal NadjiPredmete(Predmet,List<Predmet>)
- 4) Signal UcitajPredmet(Predmet)
- 5) Signal ObrisiPredmet(Predmet)
- 6) Signal ZapamtiPlan(Plan)
- 7) Signal UcitajPlan(Plan)
- 8) Signal NadjiPlanove(Plan, List<Plan>)
- 9) Signal ObrisiPlan(Plan)
- 10) Signal UcitajListuPredmeta(List<Predmet>)
- 11) Signal UcitajListuPlanova(List<Plan>)
- 12) Signal IzvrsiOdjavu(Korisnik)
- 13) Signal KreirajPredmet(Predmet)
- 14) Signal KreirajPlan(Plan)

Кључна апстрактна класа која се користи у пројектовању системских операција је `AbstractGenericOperation`. `AbstractGenericOperation` садржи методу `execute`, у оквиру које се позивају методе `executeOperation`, `preconditions`, `startTransaction` и `commitTransaction/rollbackTransaction`. Свака системска операција наслеђује општу системску операцију (`AbstractGenericOperation`) и имплементира апстрактне методе `executeOperation` и `preconditions`. У телу поменутих метода, имплементира се логика која је специфична за сваку системску операцију. Системске операције се инстанцирају у оквиру контролера, где се и позива метода `execute`.

## Пројектовање структуре софтверског система

На основу концептуалног модела праве се софтверске класе структуре.



```
public class Admin implements GenericEntity {

    private Long adminID;
    private String email;
    private String password;
    private List<Subject> listPredmet;

    public Admin() {
    }

    public Admin(Long adminID, String email, String password, List<Subject> listPredmet) {
        this.adminID = adminID;
        this.email = email;
        this.password = password;
        this.listPredmet = listPredmet;
    }
}
```

```

public class Plan implements GenericEntity{
    //Plan(StudentID, PredmetID, DatumKreiranja, UkupanBrojStavki, BrojKompletiranihStavki)

    private Subject subject;
    private Student student;
    private Date dateCreated;
    private int allItems;
    private int completedItems;
    private List<PlanItem> items;

    public Plan() {
    }

    public Plan(Subject subject, Student student, Date dateCreated, int allItems, int completedItems, List<PlanItem> items) {
        this.subject = subject;
        this.student = student;
        this.dateCreated = dateCreated;
        this.allItems = allItems;
        this.completedItems = completedItems;
        this.items = items;
    }
}

```

```

public class Project implements GenericEntity{

    private long projectID;
    private String name;
    private String description;
    private Date deadline;
    private int maxPoints;
    private Subject predmet;

    public Project() {
    }

    public Project(long projectID, String name, String description, Date deadline, int maxPoints, Subject predmet) {
        this.projectID = projectID;
        this.name = name;
        this.description = description;
        this.deadline = deadline;
        this.maxPoints = maxPoints;
        this.predmet = predmet;
    }
}

```

```

public class Subject implements GenericEntity {
    // Predmet(PredmetID, Naziv, Opis, Semestar, Espb, AdminID)

    private long predmetID;
    private String name;
    private String description;
    private int semestar;
    private int espb;
    private Admin admin;
    private List<Project> projects;

    public Subject() {
    }

    public Subject(long predmetID, String name, String description, int semestar, int espb, Admin admin, List<Project> projects) {
        this.predmetID = predmetID;
        this.name = name;
        this.description = description;
        this.semestar = semestar;
        this.espb = espb;
        this.admin = admin;
        this.projects = projects;
    }
}

```

```

public class Student implements GenericEntity {

    //Student(StudentID, BrojIndeksa, Ime, Prezime, Email, Lozinka, Telefon)

    private long StudentID;
    private String brIndeksa;
    private String firstName;
    private String lastName;
    private String email;
    private String password;
    private String phoneNumber;
    private List<Plan> plans;

    public Student() {
    }

    public Student(long StudentID, String brIndeksa, String firstName, String lastName, String email, String password, String phoneNumber, List<Plan> plans) {
        this.StudentID = StudentID;
        this.brIndeksa = brIndeksa;
        this.firstName = firstName;
        this.lastName = lastName;
        this.email = email;
        this.password = password;
        this.phoneNumber = phoneNumber;
        this.plans = plans;
    }
}

```

```

public class PlanItem implements GenericEntity{

    private Plan plan;
    private Project project;
    private long planItemID;
    private String comment;
    private Date deadline;
    private Boolean completed;

    public PlanItem() {
    }

    public PlanItem(Plan plan, Project project, long planItemID, String comment, Date deadline, Boolean completed) {
        this.plan = plan;
        this.project = project;
        this.planItemID = planItemID;
        this.comment = comment;
        this.deadline = deadline;
        this.completed = completed;
    }
}

```

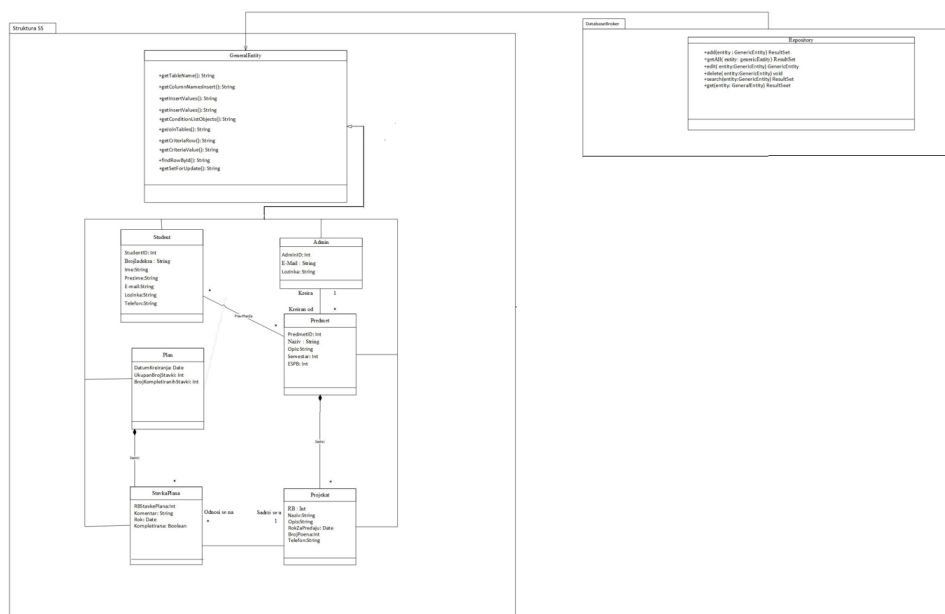
### 4.3.2.3. Брокер базе података

Класа `RepositoryDBGeneric` представља перзистентни оквир који посредује у свим операцијама над базом података и реализује следеће методе:

```
public GenericEntity add(GenericEntity entity);
public ResultSet getAll(GenericEntity param)
public GenericEntity edit(GenericEntity param)
public void delete(GenericEntity param)
public ResultSet search(GenericEntity param)
public ResultSet get(GenericEntity param)
```

Све методе класе `RepositoryDBGeneric` су генеричке, што значи да могу приме као улазни параметар било који објекат класе која наслеђује интерфејс `GenericEntity`. Класа `GenericEntity` садржи следеће методе:

```
String getTableName();
String getColumnNamesForInsert();
String getInsertValues();
void setId(Long id);
String getConditionListObjects();
String getJoinTables();
String getCriteriaRow();
String getCriteriaValue();
String findRowByID();
String getSetForUpdate();
```



Слика 27. Брокер базе података



### 4.3.3. Пројектовање складишта података

На основу концептуалног модела (који је првобитно скициран), направљен је релациони модел који описује структуру софтверског система и представља основ за пројектовање релационе базе података. У оквиру релационе базе података, направљено је следећих шест табела. Коришћени систем за управљање базом података је MySQL.

Табела админ

1 Columns 2 Indexes 3 Foreign Keys 4 Advanced 5 SQL Preview										
<input type="checkbox"/>	Column Name	Data Type	Length	Default	PK?	Not Null?	Unsigned?	Auto Incr?	Zerofill?	Comment
<input type="checkbox"/>	adminID	bigint	55		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	email	varchar	55		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	password	varchar	55		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>					<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Табела план

1 Columns 2 Indexes 3 Foreign Keys 4 Advanced 5 SQL Preview										
<input type="checkbox"/>	Column Name	Data Type	Length	Default	PK?	Not Null?	Unsigned?	Auto Incr?	Zerofill?	Comment
<input type="checkbox"/>	studentID	bigint	20		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	subjectID	bigint	20		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	dateCreated	date			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	allItems	varchar	55		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	completedItems	varchar	55		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>					<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Табела ставке плана

1 Columns 2 Indexes 3 Foreign Keys 4 Advanced 5 SQL Preview										
<input type="checkbox"/>	Column Name	Data Type	Length	Default	PK?	Not Null?	Unsigned?	Auto Incr?	Zerofill?	Comment
<input type="checkbox"/>	subjectID	bigint	55		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	studentID	bigint	55		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	planItemID	bigint	55		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	comment	varchar	55		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	deadline	date			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	completed	tinyint	4		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	projectID	bigint	55		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	subjectProjectID	bigint	55		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>					<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Табела пројекат

1 Columns 2 Indexes 3 Foreign Keys 4 Advanced 5 SQL Preview										
<input type="checkbox"/>	Column Name	Data Type	Length	Default	PK?	Not Null?	Unsigned?	Auto Incr?	Zerofill?	Comment
<input type="checkbox"/>	projectID	bigint	55		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	name	varchar	55		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	description	varchar	55		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	deadline	date			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	maxPoints	int	55		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	subjectID	bigint	55		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>					<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

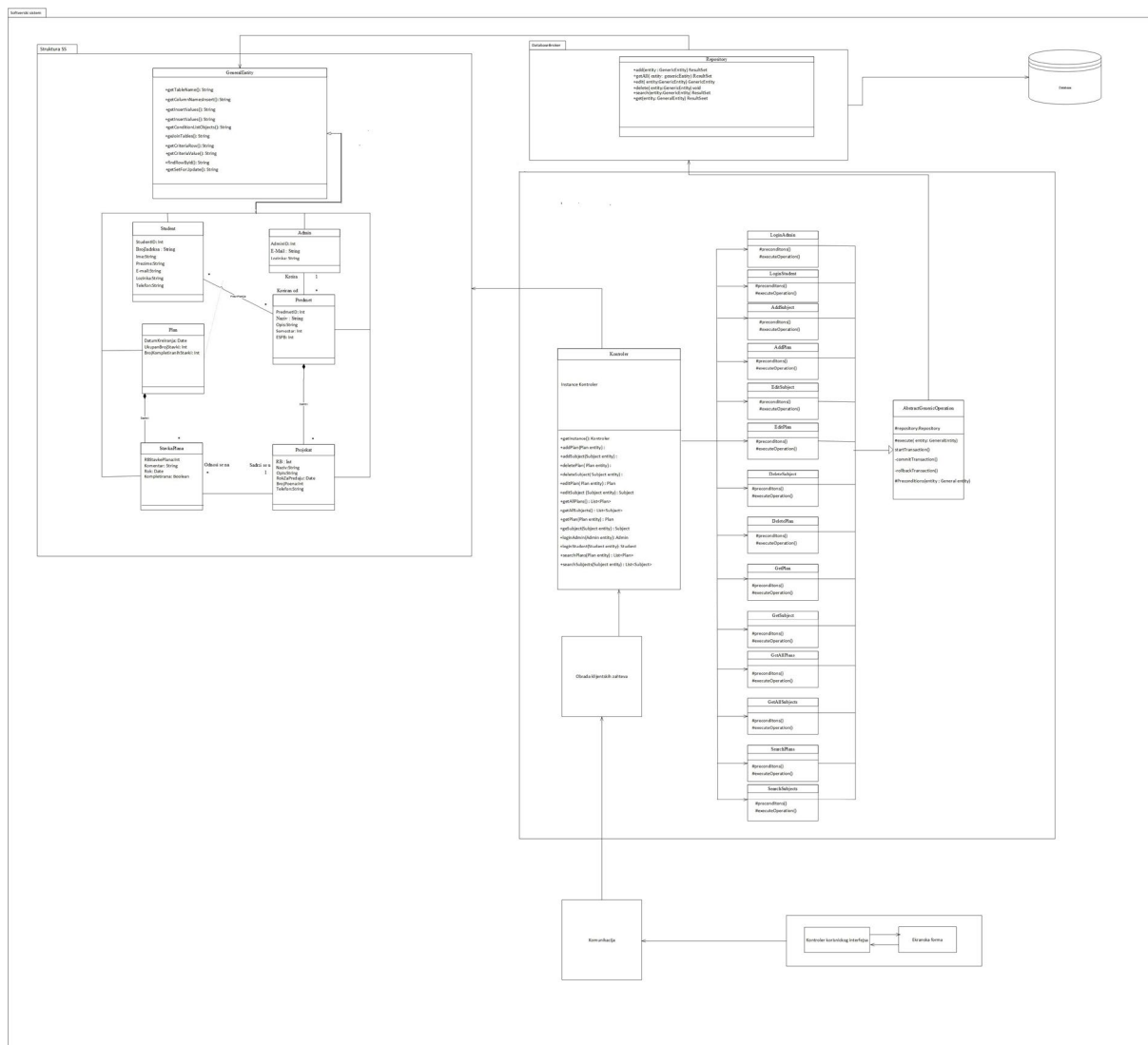
Табела студент

1 Columns 2 Indexes 3 Foreign Keys 4 Advanced 5 SQL Preview										
<input type="checkbox"/>	Column Name	Data Type	Length	Default	PK?	Not Null?	Unsigned?	Auto Incr?	Zerofill?	Comment
<input type="checkbox"/>	studentID	bigint	55		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	index	varchar	55		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	firstName	varchar	55		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	lastName	varchar	55		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	email	varchar	55		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	password	varchar	55		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	phoneNumber	varchar	55		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>					<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Табела предмет

1 Columns 2 Indexes 3 Foreign Keys 4 Advanced 5 SQL Preview										
<input type="checkbox"/>	Column Name	Data Type	Length	Default	PK?	Not Null?	Unsigned?	Auto Incr?	Zerofill?	Comment
<input type="checkbox"/>	subjectID	bigint	55		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	name	varchar	55		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	description	varchar	55		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	semestar	int	55		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	espb	int	55		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	adminID	bigint	55		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>					<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Након фазе пројектовања, на слици 28. приказана је целокупна архитектура софтверског система

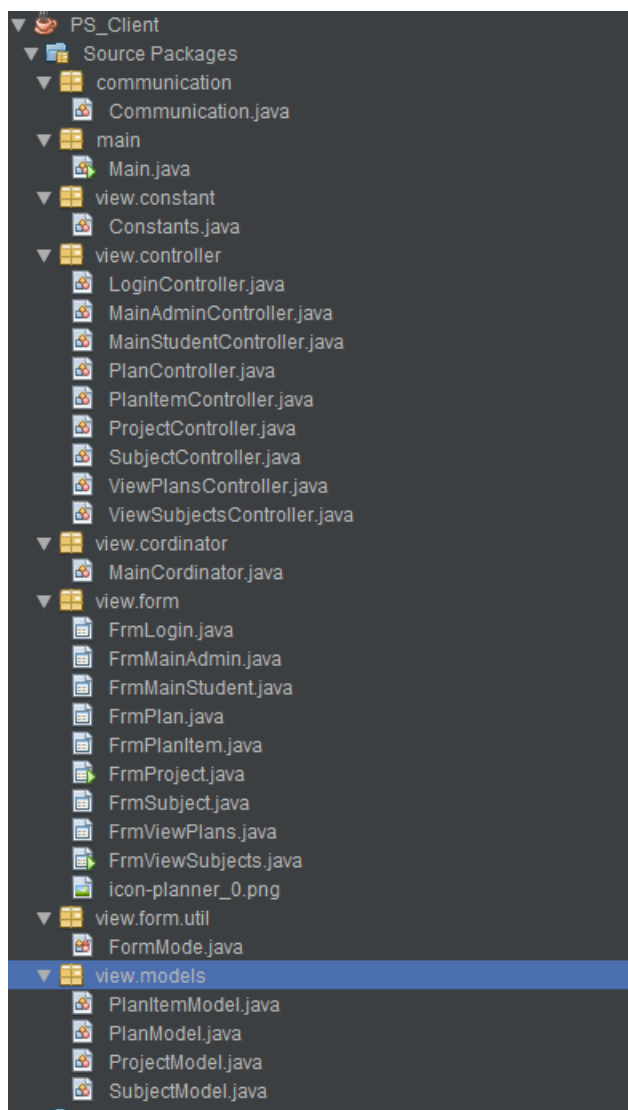


**Слика 28. Целокупна архитектура софтверског система**

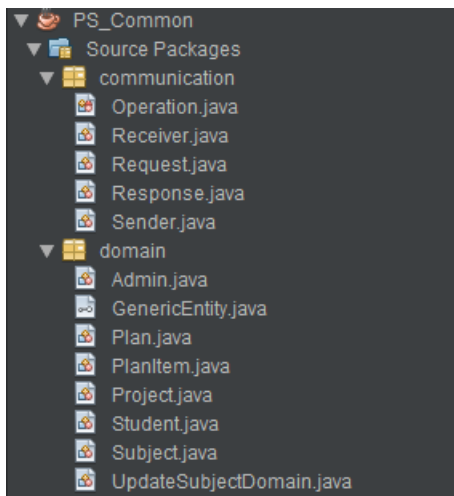
## 4.4. Имплементација

Софтверски систем имплементиран је у програмском језику Јава. Као развојно окружење коришћен је развојни алат NetBeans 8.2. На основу архитектуре софтверског система имплементирани су следеће класе:

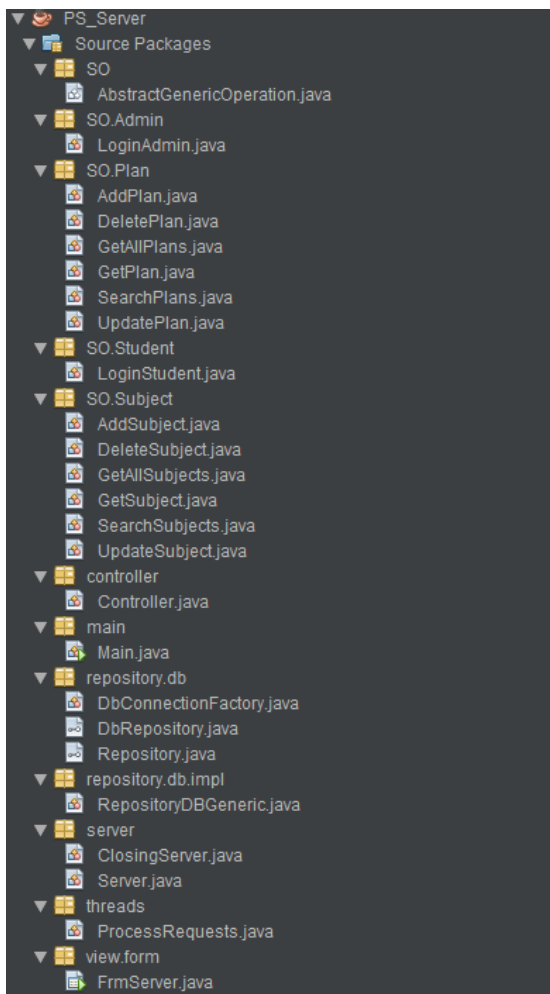
### 1.Client



## 2.Common



## 3.Server



## 4.5. Тестирање

Фаза тестирања представља последњу фазу у оквиру упрошћене Ларманове методе развоја софтвера и за циљ има проверу исправности софтверског система.

Спроводи се коришћењем три типа тестова [1]:

- Тест случајева.
- Тест процедура
- Тест компоненте.

Главни разлози због којих се спроводи фаза тестирања су снижење трошкова који би настали приликом кvara система, сигурност, квалитет и задовољство корисника. Постоји више стратегија које се користе приликом тестирања у софтверском инжењерству [7] :

- Unit Testing – тестирање јединице по јединице програма
- Integration Testing – тестирање да ли јединице функционишу заједно
- System Testing – тестирање система као целине

Такође, могуће је и ручно тестирати софтверски систем, проласком кроз сваки случај коришћења уношењем исправних али и неисправних улазних параметара како би се утврдила успешност извршења софтверског система и у тим ситуацијама.

## 5. Закључак

Одлуку да пројекат из предмета Пројектовање софтвера искористим као основу за завршни рад, донео сам због чињенице да сам радом на пројекту обухватио и интегрисао теоријска и практична знања стечена на различитим предметима у оквиру основних студија. Неки од примера су рад са базом података, рад са корисничким интерфејсом или повезивање различитих делова програма и имплементација механизма за комуникацију међу њима. Већина ових концепата била ми је позната од раније, али сам радећи на једном комплетном софтверском систему који садржи све ове елементе, достигао виши ниво њиховог разумевања. Такође било је изазовно учити и мноштво нових концепата као што су паралелно програмирање или имплементација генеричких системских операција. У софтверском систему, који комбинује новостечена и ранија знања у један интегрисани пројекат, видео сам идеалну прилику да заокружим своје студије.

Сва знања која вуку корене из више предмета на основним студијама, а интегрисана су у овом раду, представљају темељ над којим ћу усвајати нова знања. Тај процес сам већ започео, израдом пројекта из предмета Напредне Јава технологије. У оквиру тог предмета, научио сам како да знање примењено на изради Desktop апликација, проширим на израду серверског дела Web апликације написаног у Јави. Користио сам различите технологије, које представљају оквире за олакшању имплементацију Java Web апликација као што су Spring Boot и Hibernate. На клијентској страни коришћен је Angular. Тема тог рада је била израда апликације за резервисање карата, и било је бескрајно занимљиво истраживати могућности које пружају нове технологије.

Свестан сам да је ово само почетак учења и да крај учења у области софтверског инжењерства вероватно и не постоји. Узбуђен сам због нових технологија и занимљивих пројеката на којима ћу радити и захвалан на темељу који сам саградио на факултету а над којим ћу заснивати даљи напредак.

## 6. Литература

1. Проф.др Синиша Влајић - Пројектовање софтвера (СКРИПТА - РАДНИ МАТЕРИЈАЛ - 2020)
- 2 Проф.др. Синиша Влајић, др. Душан Савић, др. Илија Антовић, мр. Војислав Станојевић, дипл.инг. Милош Милић, Пројектовање софтвера – Напредне Јава технологије, Златни пресек, Београд 2008.
3. Портал рачунарских наука, <https://www.geeksforgeeks.org>
4. Stack Overflow форум, <https://www.stackoverflow.com/>
5. Baeldung портал за јава програмере, <https://www.baeldung.com/>
6. Javatpoint портал за јава програмере, <https://www.javatpoint.com/>
7. T. Hamilton, What is Software Testing? Definition, Basics & Types in Software Engineering, <https://www.guru99.com/software-testing-introduction-importance.html>