

Универзитет у Београду
Факултет организационих наука

Завршни рад

Тема: Развој софтверског система за праћење
рада кошаркашког турнира у Јава окружењу

Ментор:

Проф.др. Синиша Влајић

Студент:

Дарко Цмиљанић 167/13

Београд, 2017.

Садржај

| | |
|---|----|
| 1. Увод | 5 |
| 2. Упрошћена Ларманова метода развоја софтвера | 7 |
| 2.1. Прикупљање корисничких захтева | 8 |
| 2.2. Анализа | 9 |
| 2.2.1. Понашање софтверског система | 9 |
| 2.2.2. Структура софтверског система | 10 |
| 2.3. Пројектовање | 11 |
| 2.4. Имплементација и тестирање | 12 |
| 3. Јава, Јава технологије | 13 |
| 3.1. Концепти објектно-оријентисаног програмирања и Јава програмског језика | 14 |
| Класе објеката | 14 |
| Типови поља | 15 |
| Наслеђивање | 16 |
| Апстракција, апстрактне класе, интерфејси | 16 |
| Изузеци | 18 |
| 3.2. Графички интерфејс у Јави | 19 |
| Swing компоненте | 19 |
| Догађаји | 21 |
| 3.3. Рад у мрежи | 23 |
| URL адреса, IP адреса | 23 |
| Сокети | 24 |
| 3.4. Нити | 26 |
| 3.5. Рад са базом података | 28 |
| 4. Студијски пример | 30 |
| 4.1. Прикупљање корисничких захтева | 30 |
| 4.1.1. Вербални опис | 30 |
| 4.1.2. Случајеви коришћења | 30 |

| | |
|---|-----|
| СК1: Случај коришћења – Креирање тима..... | 31 |
| СК2: Случај коришћења – Креирање играча..... | 32 |
| СК3: Случај коришћења – Претраживање играча | 33 |
| СК4: Случај коришћења – Претраживање тима | 34 |
| СК5: Случај коришћења – Измена података играча..... | 35 |
| СК6: Случај коришћења – Измена података тима..... | 37 |
| СК7: Случај коришћења – Креирање утакмице..... | 39 |
| СК8: Случај коришћења – Претраживање утакмице | 40 |
| 4.2. Анализа | 41 |
| 4.2.1. Системски дијаграми секвенци | 41 |
| ДС1: Дијаграм секвенце случаја коришћења – Креирање тима..... | 41 |
| ДС2: Дијаграм секвенце случаја коришћења – Креирање играча..... | 42 |
| ДС3: Дијаграм секвенце случаја коришћења – Претраживање играча | 43 |
| ДС4: Дијаграм секвенце случаја коришћења – Претраживање тима | 45 |
| ДС5: Дијаграм секвенце случаја коришћења – Измена података играча..... | 47 |
| ДС6: Дијаграм секвенце случаја коришћења – Измена података тима..... | 50 |
| ДС7: Дијаграм секвенце случаја коришћења – Креирање утакмице..... | 53 |
| ДС8: Дијаграм секвенце случаја коришћења – Претраживање утакмице | 54 |
| 4.2.2. Понашање софтверског система – Дефинисање уговора о системским операцијама | 56 |
| 4.2.3. Структура софтверског система – Концептуални(доменски) модел..... | 58 |
| 4.2.4. Структура софтверског система – релациони модел..... | 58 |
| 4.3. Пројектовање | 64 |
| Архитектура софтверског система | 64 |
| 4.3.1. Пројектовање корисничког интерфејса | 65 |
| 4.3.1.1. Пројектовање екранских форми..... | 65 |
| 4.3.1.2. Пројектовање контролера корисничког интерфејса | 99 |
| 4.3.2. Пројектовање апликационе логике..... | 99 |
| 4.3.2.1. Контролер апликационе логике | 100 |
| 4.3.2.2. Пословна логика..... | 101 |

| | |
|--|-----|
| 4.3.2.3. Брокер базе података | 110 |
| 4.3.3. Пројектовање складишта података | 111 |
| 4.4 Имплементација..... | 114 |
| 4.5 Тестирање | 116 |
| 5. Закључак | 117 |
| 6. Литература | 118 |

1. Увод

Целокупна људска цивилизација заснива се на коришћењу доступних ресурса. У раним данима, економски појмови као што су земљиште или финансијски капитал бивали су најзначајнији ресурси. Након индустријске револуције и открића електричне енергије, информација као ресурс полако почиње да узима примат у значајности, што доводи до доба информационих технологија које траје и данас. Под појмом информационе технологија, подразумева се било шта што подржава процесе као што су прикупљање, представљање, пренос и коришћење информација. То могу бити многобројне машине и уређаји, као што су телевизори, рачунари, паметни телефони или таблет уређаји, или виртуелни појмови као што су програмски језици, софтвери итд.

Са експоненцијалним растом доступности информацијама, јављају се многобројни проблеми везани за њихову обраду и доступност. Због тога, дефинише се програмирање као концепт разраде проблема, излагања концептуалног решења и имплементације решења коришћењем програмског језика. Програмски језик је вештачки језик коришћен као контрола понашања одређене машине. Кроз историју, развијали су се разни стилови програмирања (парадигме), као што су машински кодови прве и друге генерације, као и процедурални језици. Праћени распрострањеном употребом процедуралних језика, направљени су објектно-оријентисани језици, где су подаци и методе манипулисања подацима чувани као једна јединица називана објектом.

Са појавом читавих организација задужених за пројектовање софтвера и информационих система, Крег Ларман развија идеје које редизајнирају организацију софтвера, усвајајући итеративност као концепт, као и агилне методе програмирања. О упрошћеној Лармановој методи биће приказано више у поглављу 2.

Поглавље 3 резервисано је за Јава програмски језик и технологије јаве програмског језика. Почетком 1990-их година, Џејмс Гозлинг, Патрик Нотон и Мајк Шеридан из компаније Sun Microsystems почињу да развијају програмски језик Јава, под утицајем језика као што су C, C++, Smalltalk, Eiffel итд. Концепт модула је избачен, а уведени су пакети, који се ослањају на фајл системе, и концепт класа као једна од главних карактеристика објектно-оријентисане парадигме. Прва званична верзија Јава програмског језика званично је објављена 1995. године. Као изузетно прихваћен језик, утицао је на каснији развој многобројних програмских језика, као што су C#, D, J#, Ada 2005, PHP, Scala итд.

У четвртом поглављу описан је студијски пример развоја система за праћење рада кошаркашког турнира у Јава окружењу, ослањајући се на претходне Јава технологије и водећи се Лармановом методом сачињене од 5 фаза (детаљније о фазама у поглављу):

- Прикупљање корисничких захтева,
- Анализа,
- Пројектовање,
- Имплементација,
- Тестирање.

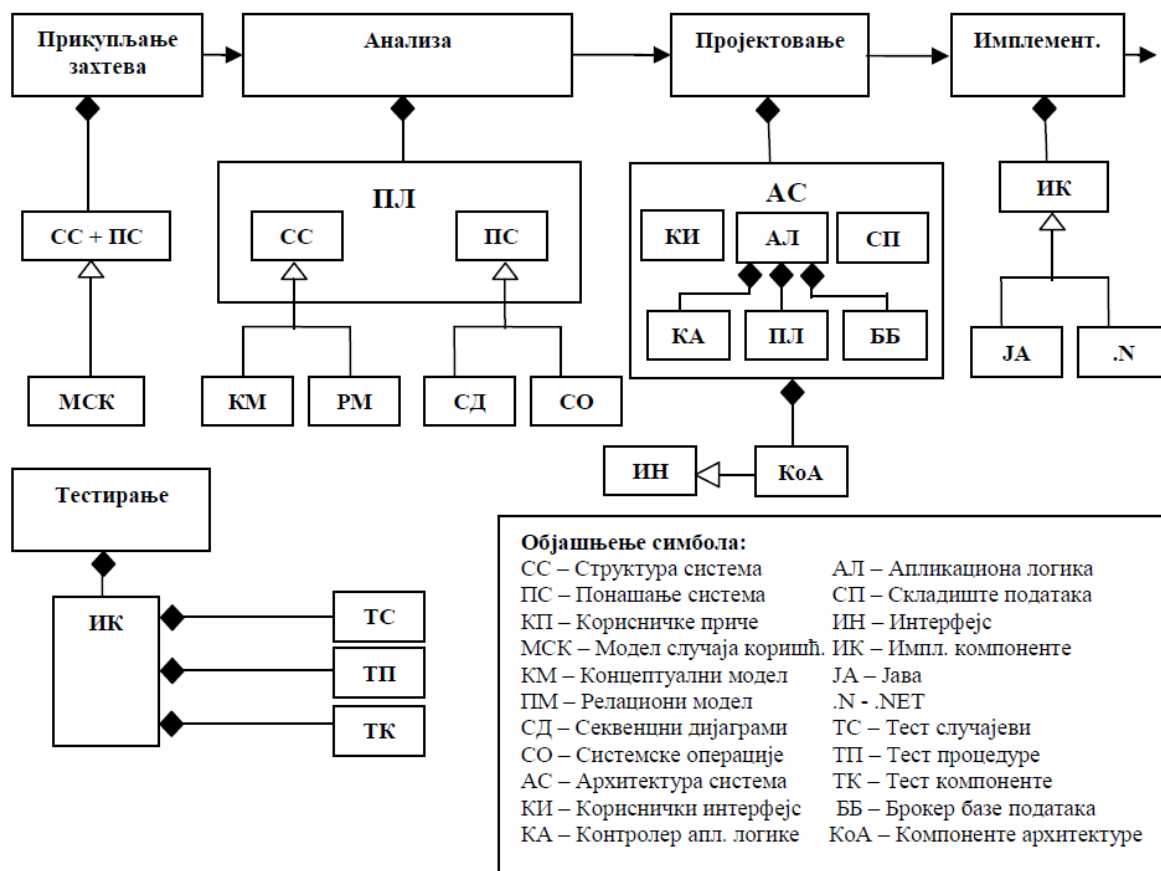
Пето поглавље овог рада представља закључак, док је шесто поглавље резервисано за коришћену литературу.

2. Упрошћена Ларманова метода развоја софтвера

Према упрошћеној Лармановој методи развоја софтвера, развој (животни циклус) софтверског система се састоји из следећих пет фаза:

- Прикупљање корисничких захтева,
- Анализа,
- Пројектовање,
- Имплементација,
- Тестирање.

Дијаграм Ларманове методе приказан је на слици 1.



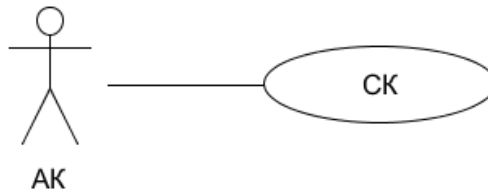
Слика 1. Развој софтверског система по Лармановој методи¹

¹ Синиша Влајић, Пројектовање софтвера (Скрипта), Београд 2015.

2.1. Прикупљање корисничких захтева

Уопштено гледано према упрошћеној Лармановој методи, захтев је својство или услов који неки систем или пројекат мора да задовољи. Захтеви се описују помоћу UML модела случаја коришћења.

Структуру модела случаја коришћења чине случајеви коришћења (СК), актори у случају коришћења и везе између актора и случајева коришћења.



Слика 2. Структура модела случаја коришћења

Правила модела случаја коришћења јесу да један модел може имати више случајева коришћења, више актора и више веза између случајева коришћења и актора; један случај коришћења може имати више веза са акторима; један актор може имати више веза са случајем коришћења, док веза постоји између тачно једног пара актора и случаја коришћења.

Било који случај коришћења описан је скупом сценарија, односно скупом могућих коришћења система од стране актора. Сваки случај коришћења има један главни сценарио и више алтернативних. У сваком сценарију, актор позива једном или више пута системске операције софтверског система.

Случај коришћења се у првим фазама представља текстуално, док се касније представља одређеним дијаграмом по избору. Текстуални опис СК има следећу структуру:

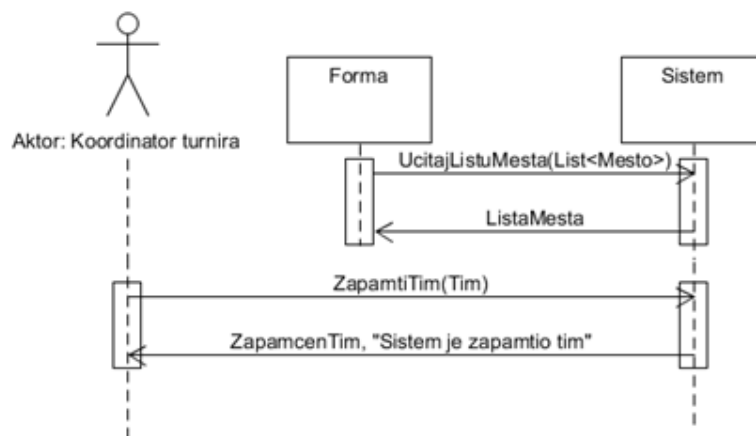
- Назив СК,
- Актори СК,
- Учесници СК,
- Предуслови неопходни да се СК изврши,
- Основни сценарио извршења СК,
- Постуслови неопходни за потврду извршења СК,
- Алтернативни сценарији извршења СК,
- Специјални захтеви, технолошки захтеви, отворена питања.

2.2. Анализа

Фаза анализе описује пословну логику софтверског система, која се састоји из структуре система и понашања система. Понашање система се описује преко системских дијаграма секвенци и системских операција, док се структура система описује преко концептуалних и релационих модела.

2.2.1. Понашање софтверског система

Дијаграм секвенци приказује догађаје у одређеном редоследу, који успостављају интеракцију између актора и софтверског система. За сваки сценарио СК, прави се системски дијаграм секвенци.

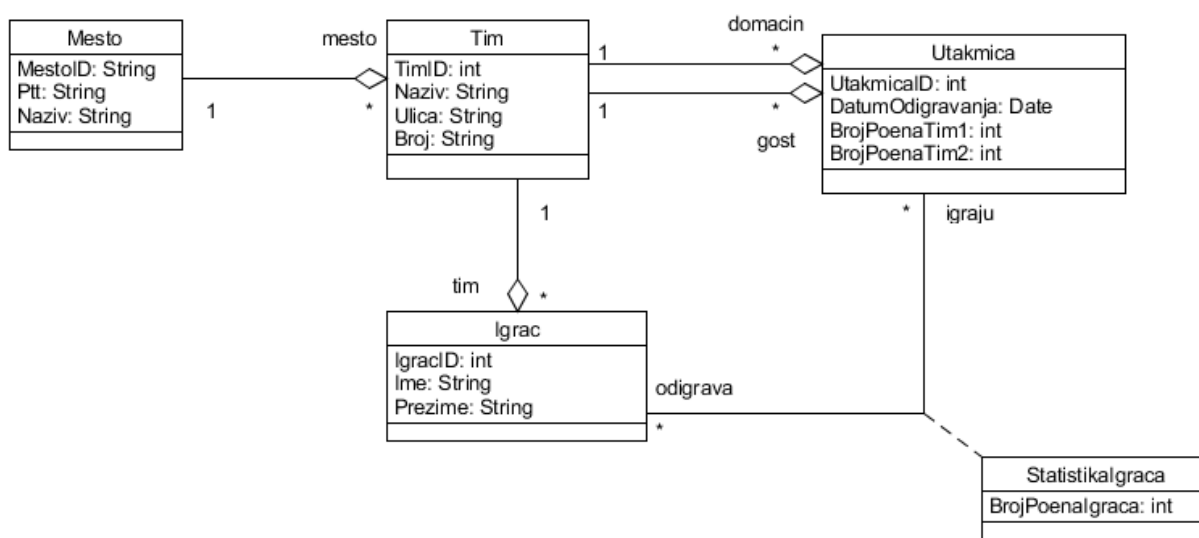


Слика 3. Пример секвенчног дијаграма за случај коришћења Креирање играча

Системска операција, као и дијаграм секвенци, описује понашање система. Свака системска операција има свој потпис, који садржи име методе и опционо улазне и/или излазне аргументе. За сваку системску операцију праве се уговори који описују шта сама системска операција ради, без подробног објашњења како ради. Уговор се састоји из операције, везе са СК, условима неопходним пред извршење системске операције као и условима који морају бити задовољени након извршења саме операције.

2.2.2. Структура софтверског система

Структура система описује се преко концептуалног и релационог модела. Концептуални модел садржи концептуалне класе и веза - асоцијација између концептуалних класа. Концептуалне класе су особине које описују саму структуру софтверског система, и треба их разликовати од софтверских класа, док сваки крај асоцијације има улогу концептуалне класе која учествује у асоцијацији.



Слика 4. Пример концептуалног модела софтверског система

Из концептуалног модела може се направити релациони модел, који даје основу за пројектовање релационе базе података. На пример, релациони модел на основу концептуалног модела са слике 4 би изгледао овако:

Tim(TimID, Naziv, Ulica, Broj, MestoID)

Igrac(IgracID, Ime, Prezime, TimID)

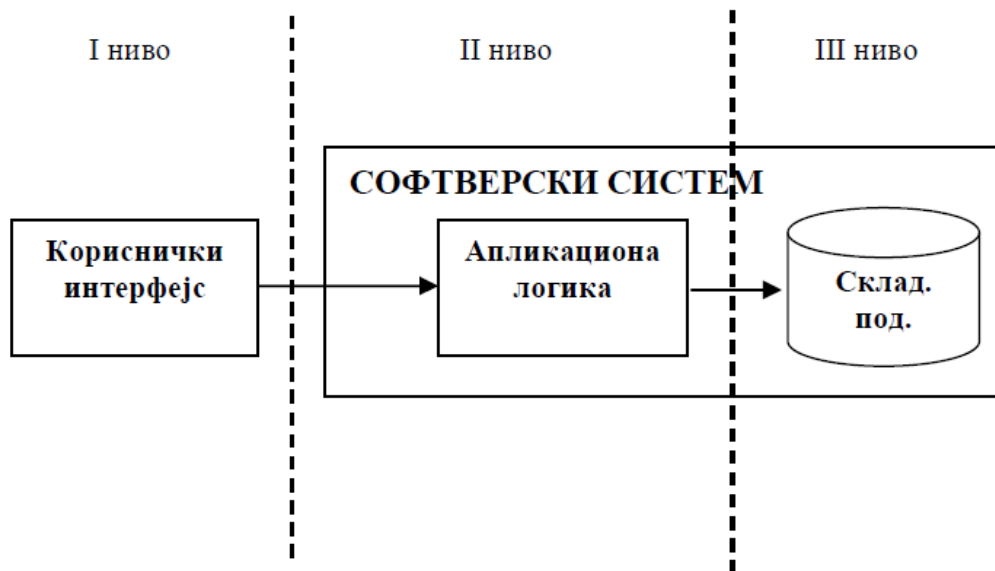
Mesto(MestoID, Ptt, Naziv)

Utakmica(UtakmicaID, DatumOdigravanja, BrojPoenaTim1, BrojPoenaTim2, TimID, TimID2)

Statistikalgraca(IgracID, UtakmicaID, BrojPoenaIgraca)

2.3. Пројектовање

Фаза пројектовања описује саму архитектуру софтверског система. Стандардно се користи тронивојска архитектура, која обухвата пројектовање корисничког интерфејса, апликационе логике и складишта података. У оквиру корисничког интерфејса, пројектују се екранске форме и контролери корисничког интерфејса; док се апликациона логика састоји од контролера апликационе логике, пословне логике и брокера базе података.



Слика 5. Тронивојска архитектура²

На основу тронивојске архитектуре, направљени су савремени апликациони сервери, који су одговорни да обезбеде сервисе за могућност реализације апликационе логике софтверског система. Сваки апликациони сервер се састоји из три основна дела:

1. Део за комуникацију са клијентом (контролер)
2. Део за комуникацију са неким складиштем података(база података)
3. Део који садржи пословну логику

² Синиша Влајић, Пројектовање софтвера(скрипта), Београд 2015.

2.4. Имплементација и тестирање

Након пројектовања, последње две фазе развоја софтверског система су имплементација и тестирање. У фази имплементације, врши се кодирање софтверског система у одређеном програмском језику, који представља имплементациону компоненту. Процес развоја софтверског система обједињује се његовим тестирањем, које се такође може поделити у неколико независних јединица. Тестирање се обично дели на три типа тестова: тест случајева, тест процедуре и тест компоненте.

3. Јава, Јава технологије

За потребе имплементације софтверског система из студијског примера, коришћен је програмски језик Јава. Прва верзија Јава програмског језика издата је 1995. године, као продукт компаније Sun Microsystems. Јава као програмски језик је објектно-оријентисан, и поштује правило да се једна класа налази у једном фајлу. Изворни код Јава програмског језика се чува у фајловима са екстензијом .java. Независност од платформе постиже се Јава виртуелном машином, која служи за интерпретирање бајт-кода. Дакле, уместо традиционалног машинског кода, Јава компајлер преводи програм из .java класе у бајт-код и чува га под екстензијом .class, а такав код се интерпретира у Јава виртуелној машини. Тиме се постиже да се исти бајт-код може извршавати на сваком оперативном систему који има инсталирану Јава виртуелну машину.

Пример простог Hello World кода је следећи:

```
public class HelloWorld {  
  
    public static void main(String[] args) {  
        System.out.println("Hello world!");  
    }  
  
}
```

Слика 6. Код HelloWorld апликације

- Име фајла који садржи класу HelloWorld мора бити HelloWorld.java
- У оквиру Јава апликације, једна класа мора имати main методу – главну функцију одговорну за покретање програма, у којој се позивају остале функције неопходне за функционисање самог програма.

3.1. Концепти објектно-оријентисаног програмирања и Јава програмског језика

Објектно-оријентисани програми засновани су на концепту објекта из стварног света. Када погледамо око себе, све што видимо представља неки објекат (столица, клупа, светло итд). Објекти из стварног света деле две карактеристике: стање и понашања. На пример, аутомобил има стања као што су тренутна брзина кретања, тренутно убачена брзина(зупчаник) и слична, као и понашања као што су покренути, укочи, пребаци у већу брзину итд. Такође, може се рећи да и неки објекат садржи неки други објекат.

Све ове опсервације из стварног света се преносе на концепте објектно-оријентисаног програмирања. Објекти у ООП језицима се такође састоје из стања и понашања. Стања се чувају у пољима објекта, док се понашања реализују преко метода објекта. Моделирањем тих стања и понашања у програмском језику објекат добија могућност да буде контролисано коришћен (на пример, ако је стање аутомобила 5 брзина, не може се користити метода за промену брзине на неку вредност која није између 1 и 5). Организовањем кода у објекте постижу се многобројне погодности:

- Модуларност: Изворни код једног објекта може бити читан и писан независно од изворног кода других објеката. Једном тако креиран, објекат може бити прослеђиван кроз систем.
- Скривање информација: Интерна имплементација објекта скривена је од спољног утицаја тако што се само дозвољава интеракција са методама тог објекта.
- Поновно коришћење кода: Ако је објекат већ креиран у програму(нпр. други пројектант), код тог објекта може се поново користити унутар програма, што дозвољава побољшање/тестирање тог објекта и сл.

Класе објеката

У стварном свету, постоји много објеката исте врсте. Нпр. постоји на хиљаде аутомобила истог модела и произвођача. Сваки појединачни аутомобил направљен је са истим скупом подешавања. Као и код објекта, овај концепт преноси се на објектно-оријентисане језике. Класа неког објекта представља шему, заједничке карактеристике које ће сваки објекат те класе садржати. Широко коришћен термин за једно појављивање објекта одређене класе јесте инстанца објекта одређене класе.

Нпр. класа Аутомобил би могла изгледати овако:

```

public class Automobil {
    String proizvođač;
    String model;
    int brojBrzina = 5;
    int trenutnaBrzina = 1;

    public void ubaciUBrzinu(int brzina){
        if(brzina > brojBrzina || brzina < 1){
            System.out.println("Automobil nema brzinu "+brzina);
            return;
        }
        System.out.println("Automobil je sada u brzini "+brzina);
    }
}

```

Слика 7. Код класе *Automobil*

Поља `proizvođač`, `model`, `brojBrzina` и `trenutnaBrzina` су стања објекта, док метода `ubaciUBrzinu` представља понашање објекта у спољном свету.

Пошто класа нема `main` методу, апликација није готова, већ се објекат Аутомобил може користити негде у апликацији. Надлежност креирања и коришћења објеката ове класе делегира се некој другој класи.

Типови поља

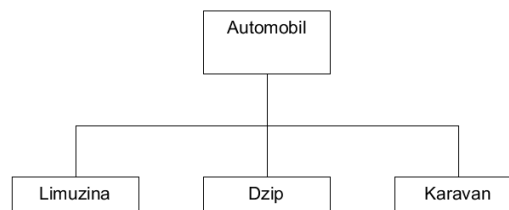
Јава програмски језик је статичног типа, што значи, сва поља морају бити декларисана пре него што могу бити коришћена. Декларација поља врши се писањем типа поља и његовим именом, нпр.

```
int brzina = 1;
```

што значи да у програму постоји поље са именом `brzina` које је нумеричког типа и има почетну вредност 1. Тип поља одређује дозвољене вредности за поље, и поред `int` простог типа, постоји још седам простих типова поља: `byte`, `short`, `long`, `float`, `double`, `boolean` и `char`. Прва три типа представљају целобројне вредности различитих опсега, `float` и `double` служе за представу децималних записа, `boolean` има вредности тачно/нетачно, док `char` представља Unicode карактер. Поред њих, постоје још два сложена типа, `String` и `Object`. `String` представља низ карактера, док је `Object` инстанца објекта одређене класе.

Наслеђивање

Поред истих карактеристика објеката, они могу имати и карактеристике специфичне за неку своју подгрупу (нпр. аутомобили могу бити лимузине, каравани, џипови...). Ипак, потребно је задржати основне карактеристике које су заједничке за све објекте. Концепт наслеђивања је један од најважнијих у Јава програмском језику.



Слика 8. Наслеђивање

У Јава програмском језику, свака класа сме имати највише једну надкласу, док свака класа може имати више подкласа. Кључна реч за наслеђивање у Јави је `extends`.

```
public class Limuzina extends Automobil{
    //kod koji određuje samo limuzinu..
}
```

Слика 9. Подкласа *Limuzina*

Овим се постиже да класа Лимузина наслеђује сва стања и методе дефинисане у класи Аутомобил, али може имати и своја дефинисана својства.

Апстракција, апстрактне класе, интерфејси

Апстракција, као још један важан концепт у Јава програмском језику, је процес скривања детаља имплементације, чиме се кориснику показују само функционалности. Тиме се фокус скреће на то шта објекат ради, а не како то ради. Постоје два начина за постизање апстракције у Јави:

- Апстрактне класе
- Интерфејси

Главна разлика између ова два концепта јесу да се апстрактном класом постиже делимична апстракција, док се интерфејсом постиже потпуна апстракција.

Апстрактна класа може прослеђивати своје методе и поља, али сама не може бити инстанцирана (не може се креирати њен објекат). Свака класа која садржи барем једну апстрактну методу мора бити декларисана као апстрактна. Кључна реч за апстрактну класу јесте `abstract`.

```
public abstract class Oblik {  
    abstract public void nacrtaj();  
}
```

Слика 10. Апстрактна класа Облик

Дакле, свака подкласа класе `Oblik` ће садржати методу `nacrtaj`, али ће имати слободу да је реимплементирају (`override`) по жељи.

```
public class Krug extends Oblik{  
  
    @Override  
    public void nacrtaj() {  
        System.out.println("Nacrtan je krug");  
    }  
  
}
```

Слика 11. Јава код подкласе Круг

```
public class Kvadrat extends Oblik{  
  
    @Override  
    public void nacrtaj() {  
        System.out.println("Nacrtan je kvadrat");  
    }  
  
}
```

Слика 12. Јава код подкласе Квадрат

За разлику од апстрактне класе, где се могу појавити и методе које нису апстрактне, код интерфејса све методе морају бити апстрактне. Иако је ова могућност доступна и код апстрактних класа, употребност интерфејса је широка. Поред пуне апстракције, интерфејс подржава могућност вишеструког наслеђивања (за разлику од апстрактних класа), као и концепт лабавог повезивања, које подразумева нижу међузависност класа. Кључна реч у Јава програмском језику за наслеђивање интерфејса јесте `implements`.

```
public interface PrenosiviUredjaj {

    public void upaliUredjaj();
    public void ugasiUredjaj();

}
```

Слика 13. Интерфејс

```
public class MobilniTelefon implements PrenosiviUredjaj{

    @Override
    public void upaliUredjaj() {
        System.out.println("Mobilni telefon je upaljen");
    }

    @Override
    public void ugasiUredjaj() {
        System.out.println("Mobilni telefon je ugasen");
    }

}
```

Слика 14. Подкласа МобилниТелефон

Изузеци

Изузеци у Јава програмском језику представљају догађај који представља поремећај у програму приликом његовог извршавања. Када се догоди грешка у методи, метода креира објекат који шаље runtime систему. Тај објекат, звани exception, носи информације о грешци, као што су њен тип и стање програма када се грешка догодила. Креирање те грешке и њено слање систему уобичајено се назива и бацање грешке или бацање изузетка (exception throw). За обрађивање, тј. хватање грешке, користи се одабрани exception handler. Један валидан Јава код мора поштовати Catch or Specify Requirement, што значи да уколико неки део кода може бацити грешку, мора бити учаурен на један од два начина:

- try-catch блок који служи за покушај обраде методе и хватање грешке уколико до ње дође, или
- метода мора бити специфицирана да може да баци изузетак, што се постиже додавањем throws клаузуле.

Такође, могуће је додати и кориснички специфициран изузетак.

3.2. Графички интерфејс у Јави

Јава програмски језик подржава и развој апликација које имају богат графички интерфејс. Скуп компоненти графичког интерфејса и сервиса који који упрошћавају развој графичких апликација се назива JFC(Java Foundation Classes). Главна карактеристика JFC-а јесте Swing пакет компоненти. Swing API садржи укупно 18 јавних пакета, али главна два пакета:

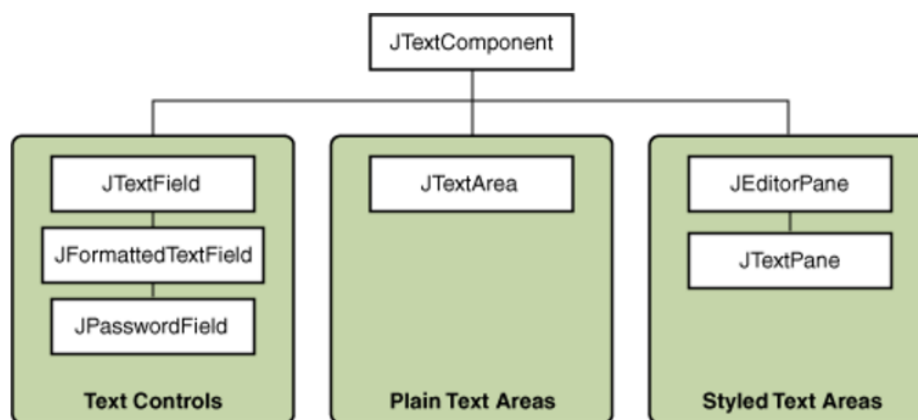
- javax.swing и
- javax.swing.event.

Предуслов за покретање Swing програма јесте да Java Development Kit буде инсталиран на рачунару.

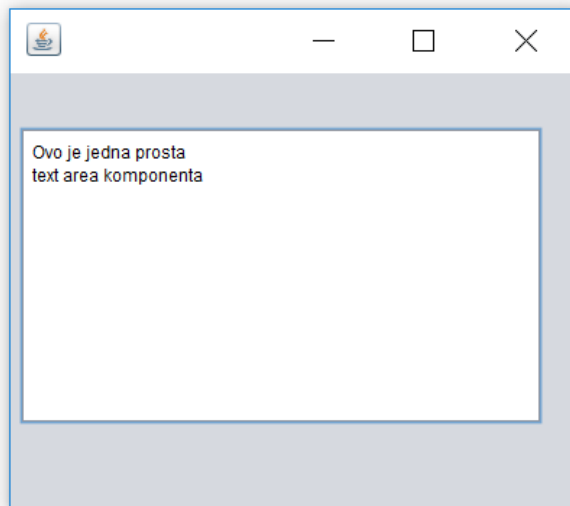
Swing компоненте

Главне Swing компоненте јесу контејнери. Контејнери обезбеђују простор за учитавање неке компоненте на графичком интерфејсу. Пошто је контејнер и сам компонента, може бити додат сам себи. Постоје три контејнерске класе на највишем нивоу: JFrame, JDialog и JApplet. JFrame је класичан top-level прозор, JDialog је независни подпрозор који приказује информације, док је JApplet класа која омогућава аpletима да користе Swing компоненте.

Најраспрострањеније компоненте су текстуалне компоненте и дугмићи. Текстуалне компоненте су изведене од надкласе JTextComponent.



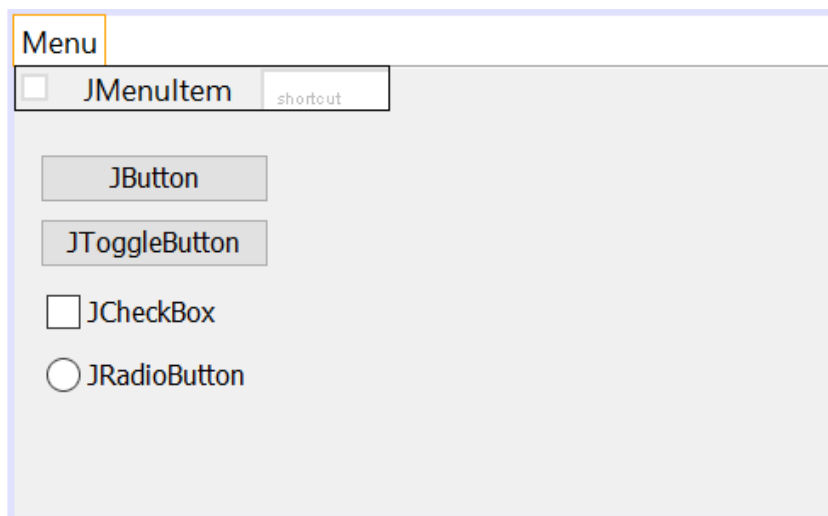
Слика 15. Хијерархија текстуалних компоненти, Swing



Слика 16. JTextArea компонента

Дугмићи су поткласе AbstractButton класе. Типови дугмића које Swing подржава су:

- JButton, регуларно дугме,
- JCheckBox, дугме за штиклирање,
- JRadioButton, радио дугме,
- JMenuItem, ставка у менију,
- JCheckBoxMenuItem, ставка у менију са дугметом за штиклирање
- JRadioButtonMenuItem, ставка у менију са радио дугметом,
- JToggleButton, on/off тип дугмета.

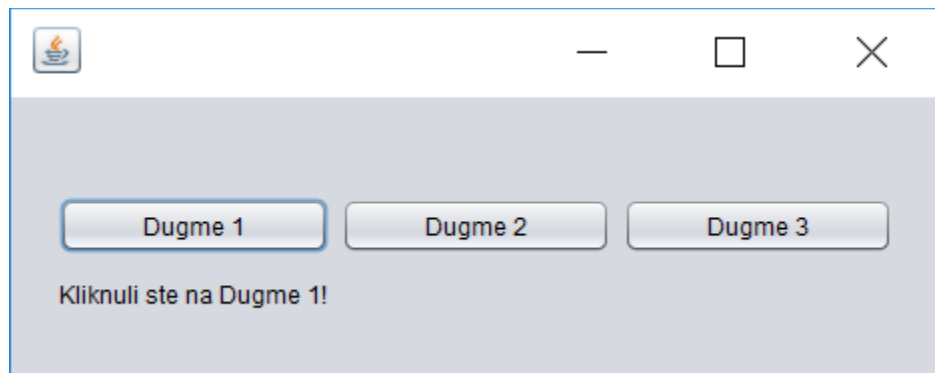


Слика 17. Неколико типова дугмића

Догађаји

Догађај представља промену стања неког објекта. На пример, клик на дугме, померање миша, укуцавање карактера на тастатури итд. Када се деси неки догађај у Јави, механизам који служи за обраду тог догађаја назива се event handling. Конкретан event handling модел Јава програмског језика јесте Delegation Event Model. Delegation Event Model има два кључна учесника:

- Source – Source представља објекат над којим се десио догађај. Његова улога јесте да проследи информацију од значаја event handler учеснику када се догађај деси.
- Listener – Објекат који представља event handler-а. Посао Listener-а јесте да генерише одговор на догађај. Са становишта Јава имплементације, Listener је такође објекат. Једном када прихвати догађај, он га обрађује и шаље одговор назад.



Слика 18. Обрада захтева кликом на Дугме 1

Дугме 1 представља Source објекат, и шаље Listener објекту да је неопходно да позове методу која ће поставити текстуално поље на вредност дугмета које је притиснуто.

Дугмићи са форме и лабела која садржи одговарајући текст су поља која су дефинисана у пакету `javax.swing`, и свако дугме представља Source објекат. Када се кликне дугме, Listener који је подешен над тим објектом позива функцију `actionPerformed` која има задатак да подеси одговарајући текст на лабели у зависности од тога које је дугме кликнуто. Код који одговара слици 18 приказан је на слици 19.

```

public class NewJFrame1 extends javax.swing.JFrame {

    @SuppressWarnings("unchecked")
    Generated Code

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        jLabel1.setText("Kliknuli ste na Dugme 1!");
    }

    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
        jLabel1.setText("Kliknuli ste na Dugme 2!");
    }

    private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
        jLabel1.setText("Kliknuli ste na Dugme 3!");
    }

    public NewJFrame1() {
        initComponents();
    }

    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new NewJFrame1().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JButton jButton1;
    private javax.swing.JButton jButton2;
    private javax.swing.JButton jButton3;
    private javax.swing.JLabel jLabel1;
    // End of variables declaration
}

```

Слика 19. Код графичког интерфејса

3.3. Рад у мрежи

URL адреса, IP адреса

URL адреса представља путању до ресурса на Интернету. Када желимо приступити одређеном фајлу на Интернету, шаљемо URL свом web browser клијенту по сличном принципу као што је писање адресе на писму у пошти. URL се чува у облику стринга и има две главне компоненте: протокол неопходан да приступи ресурсу и локацију ресурса. На адреси <http://facebook.com>, http представља идентификатор протокола(HyperText Transfer Protocol), док је facebook.com локација ресурса. Локација може садржати више компоненти, као што су путања до фајла на удаљеној машини, број порта на који је неопходно бити конектован, као и референца, уколико се тражи локација унутар фајла. Најпростији начин за креирање URL адресе у Јава програму јесте да се направи URL објект (предефинисана класа у Јави) и проследити јој URL као параметар у облику стринг променљиве:

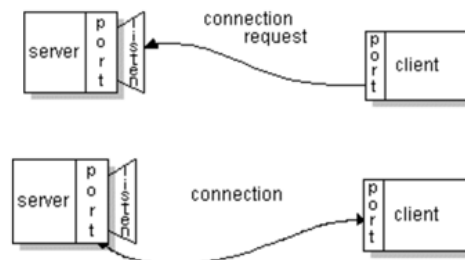
```
URL adresa = new URL("http://facebook.com/");
```

Затим, URL класа има различите методе за манипулацију адресе, као што су `getHost`, `getPort` итд. Како би се извршила конекција на URL адресу, потребно је позвати методу `openConnection`, која као излазни аргумент даје спремну конекцију на URL – `URLConnection` објект, над којим се само позива метода `connect`.

IP адреса је јединствена комбинација бројева коју има сваки рачунар приликом међусобног саобраћаја на интернету уз коришћење интернет протокола (IP – Internet Protocol). Структура IP адресе се садржи од 4 осмобитна блока, нпр. 172.16.254.1. Вредности сваког блока се крећу од 0 до 255. Поред нумеричке вредности адресе, постоји и симболичка адреса, као што је `fon.bg.ac.rs`. Једна симболичка адреса може бити везана за више нумеричких адреса. Сервис за повезивање симболичких и нумеричких адреса јесте DNS (Domain Naming Service).

Сокети

Да би се омогућило повезивање између клијента и сервера на некој мрежној апликацији, сваки од ова два дела програма отвара сокет на крају своје конекције преко које се читају и пишу подаци. Адреса сокета се састоји из два дела: адресе рачунара на коме се налази програм који је генерисао сокет, и број порта који је генерисан помоћу сокета (нпр. 147.90.131.23:8999). Постоје две главне класе сокета у Јава програмском језику, `ServerSocket` и `Socket`. Задатак `ServerSocket` класе јесте да „ослушкује“ мрежу, држећи отворен порт за конекцију. Када се одређени клијент преко своје `Socket` класе конектује на серверски софтвер „гађајући“ адресу и порт сервера, конекција је успостављена.



Слика 20. Упрошћен приказ клијент-сервер повезивања

По успешној конекцији, потребно је отворати токове у конекцији који су задужени за слање и примање података. Када су сокети спремни, могуће је дефинисати објекте `BufferedReader` и `PrintWriter` са улазних и излазних токова сокета. Ова два објекта служе за читање и писање података са улазног, односно на излазни ток сокета.

Пре него што се било који клијент конектује, сервер програм почиње креирањем серверског сокета на одређеном порту. Након тога, командом `accept` почиње ослушкивање мреже и чекање на клијента. Када се клијент закачи за порт на адреси сервера, отварају се токови, а затим креирају објекти за читање и писање података.

```
int brojPorta = 9000;
try {
    ServerSocket serverSocket = new ServerSocket(brojPorta);
    Socket clientSocket = serverSocket.accept();
    PrintWriter out
        = new PrintWriter(clientSocket.getOutputStream(), true);
    BufferedReader in = new BufferedReader(
        new InputStreamReader(clientSocket.getInputStream()));
} catch (IOException ex) {
    System.out.println(ex.getMessage());
}
```

Слика 21. Приказ кода за подизање серверског сокета

Основни кораци клијентске стране су следећи:

1. Отвара се клијентски сокет, са улазним параметрима `adresa` и `port` који представљају адресу серверског рачунара и порт који чека на конекцију (порт који је отворио серверски сокет).
2. Отварају се излазни и улазни токови сокета.
3. Врши се читање и писање преко токова у складу са протоколом сокета.
4. Затварају се токови.
5. Затвара се сокет.

Једини корак који се може разликовати од клијента до клијента је корак 3, док су остали предефинисани за сваког клијента.

```
try {
    Socket klijentSoket = new Socket(adresa, port);
    PrintWriter out
        = new PrintWriter(klijentSoket.getOutputStream(), true);
    BufferedReader in
        = new BufferedReader(
            new InputStreamReader(klijentSoket.getInputStream()));
    BufferedReader stdIn
        = new BufferedReader(
            new InputStreamReader(System.in));
} catch (IOException ex) {
    System.out.println(ex.getMessage());
}
```

Слика 22. Приказ кода за клијентски сокет

У досадашњем примеру, показана је интеракција између једног клијента са сервером. Малом изменом кода, сервер може подржавати операције са више клијената:

```
int brojPorta = 9000;
try {
    ServerSocket serverSocket = new ServerSocket(brojPorta);
    while (true) {
        Socket clientSocket = serverSocket.accept();
        PrintWriter out
            = new PrintWriter(clientSocket.getOutputStream(), true);
        BufferedReader in = new BufferedReader(
            new InputStreamReader(clientSocket.getInputStream()));
    }
} catch (IOException ex) {
    System.out.println(ex.getMessage());
}
```

Слика 23. Сервер који може радити са више клијената

Угњеждавањем асерпт методе унутар `while` петље са параметром `true`, `while` петља се непрестано покреће, а самим тим и сваки пут окида асерпт методу када дође нови клијент. По сваком пристизању клијента, отварају се посебни улазни и излазни токови ка клијентском сокету, омогућавајући да серверски сокет буде повезан са више клијентских сокета.

3.4. Нити

Уколико више процеса међусобно сарађују у извршењу неког задатка, јавља се проблем њихове међусобне комуникације и размене података јер сваки процес заузима посебан меморијски простор. Тај проблем је решен појавом нити (енг. threads) које деле исти меморијски простор. Јава као програмски језик подржава вишенитно програмирање, што значи да један програм(процес) може да обавља више нити истовремено. Нит представља део програма који може истовремено да се извршава са другим нитима истог програма. Међутим, потребно је пажљиво имплементирати нити, јер може доћи до замршене комуникације. Сваки програм има барем једну, главну нит, која има способност да креира додатне нити. Апликација која има креирану инстанцу класе Thread мора на један од постојећа два начина да имплементира код који ће се извршавати у нити:

- Креирањем Runnable објекта. Runnable интерфејс дефинише само једну методу, run, која треба да садржи код који се извршава у нити. Затим, тај објекат се провлачи кроз Thread конструктор.

```
public class HelloRunnable implements Runnable {  
  
    public void run() {  
        System.out.println("Hello from a thread!");  
    }  
  
    public static void main(String args[]) {  
        (new Thread(new HelloRunnable())).start();  
    }  
}  
  
run:  
Hello from a thread!  
BUILD SUCCESSFUL (total time: 0 seconds)
```

Слика 24. Runnable класа и њен испис

- Креирањем подкласе класе Thread. Класа Thread сама по себи имплементира интерфејс Runnable, и њена run метода не извршава ништа, остављајући простора за реимплементације те методе према жељи самог пројектанта.

```
public class HelloThread extends Thread {  
    public void run() {  
        System.out.println("Hello from a thread!");  
    }  
  
    public static void main(String args[]) {  
        (new Thread(new HelloThread())).start();  
    }  
}  
  
run:  
Hello from a thread!  
BUILD SUCCESSFUL (total time: 0 seconds)
```

Слика 25. Thread класа и њен испис

У оба случаја, нит се позива методом start. Постоје разни облици манипулације нитима, као што су успављивање, прекидање, буђење, чекање на завршетак друге нити итд.

Као што је већ напоменуто, синхронизација нити је појам на који је неопходно обратити пажњу, јер је веома лако упасти у deadlock – стање блокираних нити.

3.5. Рад са базом података

Повезивање неког програма који је написан у Јава програмском језику и неког од система за управљање базом података (MySQL, Oracle, MS Access...) ради се преко Јавиног JDBC (Java Database Connectivity) интерфејса, и управљачког програма (драјвера) који се прави посебно за сваки систем за управљање базом података. JDBC има способност да приступи било каквом табеларном приказу, а посебно је погодан за релационе шеме података. Четири компоненте које сачињавају JDBC су:

- JDBC API - служи за приступ релационом систему за управљање базом података из Јава програмског језика, направљен за интеракцију са хетерогеном околином,
- JDBC Driver Manager – DriverManager класа дефинише објекте који повезују Јава апликацију са JDBC драјвером,
- JDBC Test Suite – за тестирање JDBC драјвера који покрећу програм,
- JDBC-ODBC Bridge – служи за приступ JDBC-у преко ODBC (Open Database Connectivity) драјвера.

Кораци повезивања Јава програма и базе података изабраног система за управљање базом података су:

1. Укључивање у Јава програм JDBC API-ја,
2. Учитавање управљачког програма у Јава програм,
3. Успостављање конекције између Јава програма и базе података изабраног система за управљање базом података.

Како би се извршила одређена операција над базом података, потребно је направити објекат класе Statement. Објекат класе Statement прави се преко операције createStatement над објектом класе Connection. Након тога, објекту класе Statement може се проследити упит над базом у облику String променљиве. Резултат SELECT наредбе се чува у објекту класе ResultSet, која садржи све слоге табеле над којом се извршава наредба.

```

try {
    String dbUrl = "jdbc:mysql://127.0.0.1:3306/baza";
    String user = "root";
    String pass = "root";
    Class.forName("com.mysql.jdbc.Driver");

    Connection konekcija = DriverManager.getConnection(dbUrl, user, pass);
    Statement naredba = konekcija.createStatement();
    String upit = "SELECT jmbg, ime, prezime FROM Osoba";
    ResultSet rs = naredba.executeQuery(upit);
    while (rs.next()) {
        System.out.println(rs.getString("jmbg") + " " + rs.getString("ime")
            + " " + rs.getString("prezime"));
    }
    naredba.close();
    konekcija.close();
} catch (ClassNotFoundException cnfe) {
    System.out.println("Niје ucitan upravljacki program: " + cnfe);
} catch (SecurityException se) {
    System.out.println("Nedozvoljena operacija: " + se);
} catch (SQLException sqle) {
    System.out.println("Greska konekcije: " + sqle);
}

```

Слика 26. Извршавање SELECT наредбе над базом података

4. Студијски пример

Студијски пример направљен је у Јава програмском језику, служећи се описаним Јава технологијама и поштујући пет фаза упрошћене Ларманове методе.

4.1. Прикупљање корисничких захтева

4.1.1. Вербални опис

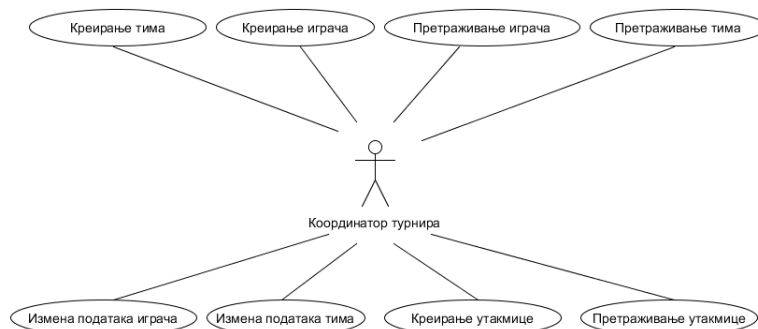
Потребно је направити апликацију која ће водити евиденцију о тимовима учесницима кошаркашког турнира. У апликацији ће се водити евиденција играча у тимовима, као и евиденција тимова који су пријављени за турнир.

Апликација омогућава унос(креирање) новог тима, као и новог играча; промену података тимовима и играчима уколико су неисправни подаци. Такође, потребно је омогућити и унос нове утакмице. Могуће је и претражити одређене играче, тимове или утакмице како би се стекао увид у њихове податке.

4.1.2. Случајеви коришћења

У овој апликацији, идентификовано је осам случајева коришћења:

1. Креирање тима
2. Креирање играча
3. Претраживање играча
4. Претраживање тима
5. Измена података играча
6. Измена података тима
7. Креирање утакмице
8. Претраживање утакмице



Слика 27. Случајеви коришћења

СК1: Случај коришћења – Креирање тима

Назив СК

Креирање тима

Актори СК

Координатор турнира

Учесници СК

Координатор турнира и систем (програм)

Предуслов: Систем је укључен и координатор турнира је улогован под својом шифром. Систем приказује форму за рад са тимом. Учитана је листа места.

Основни сценарио СК

1. Координатор турнира уноси податке у тим. (АПУСО)
2. Координатор турнира контролише да ли је коректно унео податке у тим. (АНСО)
3. Координатор турнира позива систем да запамти податке о тиму. (АПСО)
4. Систем памти податке о тиму. (СО)
5. Систем приказује координатору турнира запамћени тим и поруку: “Систем је запамтио тим”. (ИА)

Алтернативна сценарија

5.1 Уколико систем не може да запамти податке о тиму он приказује координатору турнира поруку “Систем не може да запамти тим”. (ИА)

Назив СК

Креирање играча

Актори СК

Координатор турнира

Учесници СК

Координатор турнира и систем (програм)

Предуслов: Систем је укључен и координатор турнира је улогован под својом шифром. Систем приказује форму за рад са играчем. Учитана је листа тимова.

Основни сценарио СК

1. Координатор турнира уноси податке у играча. (АПУСО)
2. Координатор турнира контролише да ли је коректно унео податке у играча. (АНСО)
3. Координатор турнира позива систем да запамти податке о играчу. (АПСО)
4. Систем памти податке о играчу. (СО)
5. Систем приказује координатору турнира запамћени играч и поруку: “Систем је запамтио играча”. (ИА)

Алтернативна сценарија

5.1 Уколико систем не може да запамти податке о играчу он приказује координатору турнира поруку “Систем не може да запамти играча”. (ИА)

Назив СК

Претраживање играча

Актори СК

Координатор турнира

Учесници СК

Координатор турнира и систем (програм)

Предуслов: Систем је укључен и координатор турнира је улогован под својом шифром. Систем приказује форму за рад са играчем.

Основни сценарио СК

1. Координатор турнира уноси вредност по којој претражује играче. (АПУСО)
2. Координатор турнира позива систем да нађе играче по задатој вредности. (АПСО)
3. Систем тражи играче по задатој вредности. (СО)
4. Систем приказује координатору турнира податке о играчима и поруку: “Систем је нашао играче по задатој вредности”. (ИА)
5. Координатор турнира бира играча. (АПУСО)
6. Координатор турнира позива систем да учита играча. (АПСО)
7. Систем учитава играча. (СО)
8. Систем приказује координатору турнира податке о играчу и поруку: “Систем је прочитао играча”. (ИА)

Алтернативна сценарија

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

8.1 Уколико систем не може да учита играча он приказује координатору турнира поруку: “Систем не може да учита играча”. (ИА)

Назив СК

Претраживање тима

Актори СК

Координатор турнира

Учесници СК

Координатор турнира и систем (програм)

Предуслов: Систем је укључен и координатор турнира је улогован под својом шифром. Систем приказује форму за рад са тимом.

Основни сценарио СК

1. Координатор турнира уноси вредност по којој претражује тимове. (АПУСО)
2. Координатор турнира позива систем да нађе тимове по задатој вредности. (АПСО)
3. Систем тражи тимове по задатој вредности. (СО)
4. Систем приказује координатору турнира податке о тимовима и поруку: “Систем је нашао тимове по задатој вредности”. (ИА)
5. Координатор турнира бира тим. (АПУСО)
6. Координатор турнира позива систем да учита тим. (АПСО)
7. Систем учитава тим. (СО)
8. Систем приказује координатору турнира податке о играчу и поруку: “Систем је прочитао тим”. (ИА)

Алтернативна сценарија

- 4.1 Уколико систем не може да нађе тимове он приказује координатору турнира поруку: “Систем не може да нађе тимове по задатој вредности”. Прекида се извршење сценарија. (ИА)
- 8.1 Уколико систем не може да учита тим он приказује координатору турнира поруку: “Систем не може да учита тим”. (ИА)

Назив СК

Измена података **играча**

Актори СК

Координатор турнира

Учесници СК

Координатор турнира и **систем** (програм)

Предуслов: **Систем** је укључен и **координатор турнира** је улогован под својом шифром. Систем приказује форму за рад са **играчем**. Учитана је листа тимова.

Основни сценарио СК

1. **Координатор турнира** уноси вредност по којој претражује **играче**. (АПУСО)
2. **Координатор турнира** позива **систем** да нађе **играче** по задатој вредности. (АПСО)
3. **Систем** тражи **играче** по задатој вредности. (СО)
4. **Систем** приказује **координатору турнира** **играче** и поруку: “**Систем** је нашао **играче** по задатој вредности”. (ИА)
5. **Координатор турнира** бира **играча**. (АПУСО)
6. **Координатор турнира** позива **систем** да учита **играча**. (АПСО)
7. **Систем** учитава **играча**. (СО)
8. **Систем** приказује **координатору турнира** податке о **играчу** и поруку: “**Систем** је прочитао **играча** ”. (ИА)
9. **Координатор турнира** уноси (**мења**) податке о **играчу**. (АПУСО)
10. **Координатор турнира** контролише да ли је коректно унео податке о **играчу**. (АНСО)
11. **Координатор турнира** позива **систем** да запамти податке о **играчу**. (АПСО)
12. **Систем** памти податке о **играчу**. (СО)
13. **Систем** приказује **координатору турнира** запамћеног **играча** и поруку: “**Систем** је запамтио **играча**.” (ИА)

Алтернативна сценарија

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

8.1 Уколико **систем** не може да учита **играча** он приказује **координатору турнира** поруку: “**Систем** не може да учита **играча**”. Прекида се извршење сценарија. (ИА)

13.1 Уколико **систем** не може да запамти податке о **играчу** он приказује **координатору турнира** поруку “**Систем** не може да запамти **играча**”. (ИА)

Назив СК

Измена података тима

Актори СК

Координатор турнира

Учесници СК

Координатор турнира и систем (програм)

Предуслов: Систем је укључен и координатор турнира је улогован под својом шифром. Систем приказује форму за рад са тимом. Учитана је листа места.

Основни сценарио СК

1. Координатор турнира уноси вредност по којој претражује тимове. (АПУСО)
2. Координатор турнира позива систем да нађе тимове по задатој вредности. (АПСО)
3. Систем тражи тимове по задатој вредности. (СО)
4. Систем приказује координатору турнира тимове и поруку: “Систем је нашао тимове по задатој вредности”. (ИА)
5. Координатор турнира бира тим. (АПУСО)
6. Координатор турнира позива систем да прочита тим. (АПСО)
7. Систем читава тим. (СО)
8. Систем приказује координатору турнира податке о тиму и поруку: “Систем је прочитао тим”. (ИА)
9. Координатор турнира уноси (мења) податке о тиму. (АПУСО)
10. Координатор турнира контролише да ли је коректно унео податке о тиму. (АНСО)
11. Координатор турнира позива систем да запамти податке о тиму. (АПСО)
12. Систем памти податке о тиму. (СО)
13. Систем приказује координатору турнира запамћени тим и поруку: “Систем је запамтио тим.” (ИА)

Алтернативна сценарија

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

8.1 Уколико **систем** не може да учита **тим** он приказује **координатору турнира** поруку: “**Систем** не може да учита **тим**”. Прекида се извршење сценарија. (ИА)

13.1 Уколико **систем** не може да запамти податке о **тиму** он приказује **координатору турнира** поруку “**Систем** не може да запамти **тим**”. (ИА)

СК7: Случај коришћења – Креирање утакмице

Назив СК

Креирање утакмице

Актори СК

Координатор турнира

Учесници СК

Координатор турнира и систем (програм)

Предуслов: Систем је укључен и координатор турнира је улогован под својом шифром. Систем приказује форму за рад са утакмицом. Учитана је листа тимова.

Основни сценарио СК

1. Координатор турнира уноси податке у утакмицу. (АПУСО)
2. Координатор турнира контролише да ли је коректно унео податке у утакмицу. (АНСО)
3. Координатор турнира позива систем да запамти податке о утакмици. (АПСО)
4. Систем памти податке о утакмици. (СО)
5. Систем приказује координатору турнира запамћену утакмицу и поруку: “Систем је запамтио утакмица”. (ИА)

Алтернативна сценарија

- 5.1 Уколико систем не може да запамти податке о утакмици он приказује координатору турнира поруку “Систем не може да запамти утакмицу”. (ИА)

СК8: Случај коришћења – Претраживање утакмице

Назив СК

Претраживање утакмице

Актори СК

Координатор турнира

Учесници СК

Координатор турнира и систем (програм)

Предуслов: Систем је укључен и координатор турнира је улогован под својом шифром. Систем приказује форму за рад са утакмицом.

Основни сценарио СК

1. Координатор турнира уноси вредност по којој претражује утакмице. (АПУСО)
2. Координатор турнира позива систем да нађе утакмице по задатој вредности. (АПСО)
3. Систем тражи утакмице по задатој вредности. (СО)
4. Систем приказује координатору турнира податке о утакмицама и поруку: “Систем је нашао утакмице по задатој вредности”. (ИА)

Алтернативна сценарија

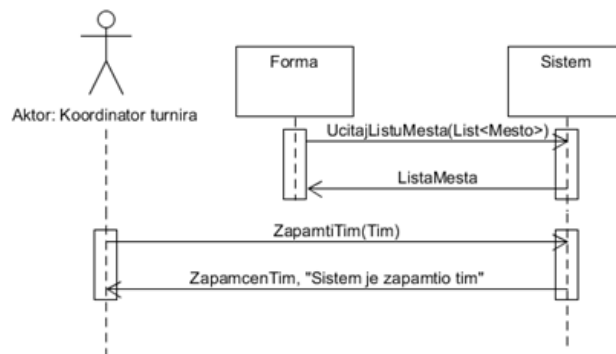
- 4.1 Уколико систем не може да нађе утакмице он приказује координатору турнира поруку: “Систем не може да нађе утакмице по задатој вредности”. (ИА)

4.2. Анализа

4.2.1. Системски дијаграми секвенци

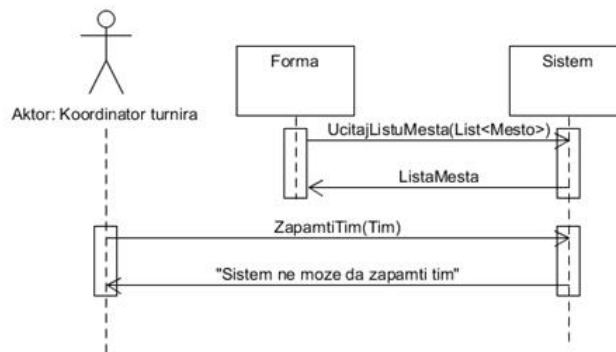
ДС1: Дијаграм секвенце случаја коришћења – Креирање тима

1. **Форма** **позива** **систем** да учита листу места. (АПСО)
2. **Систем** **враћа** **форми** листу места.(ИА)
3. **Координатор турнира** **позива** **систем** да запамти податке о **тиму**. (АПСО)
4. **Систем** **приказује** **координатору турнира** запамћени **тим** и поруку: “**Систем** је запамтио **тим**”. (ИА)



Алтернативна сценарија

4.1 Уколико **систем** не може да запамти податке о **тиму** он приказује **координатору турнира** поруку “**Систем** не може да запамти **тим**”. (ИА)

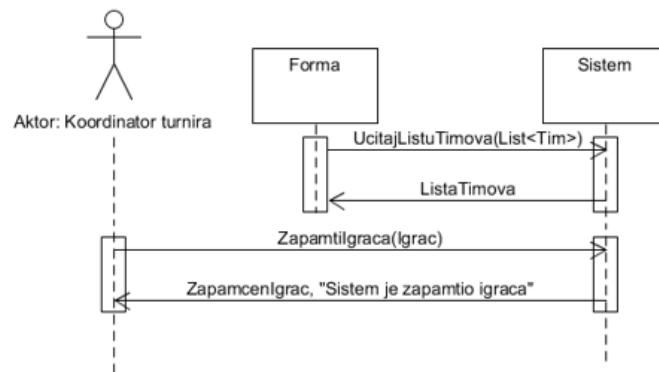


Са наведених секвенцих дијаграма уочавају се 2 системске операције:

1. Signal **UcitajListuMesta(List<Mesto>);**
2. Signal **ZapamtiTim(Tim).**

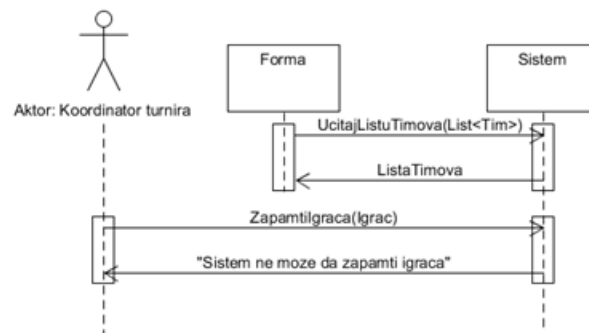
ДС2: Дијаграм секвенце случаја коришћења – Креирање играча

1. **Форма** **позива** **систем** да учита листу тимова. (АПСО)
2. **Систем** **враћа** **форми** листу тимова. (ИА)
3. **Координатор турнира** **позива** **систем** да запамти податке о **играчу**. (АПСО)
4. **Систем** **приказује** **координатору турнира** запамћени **играч** и поруку: “**Систем** је запамтио **играча**”. (ИА)



Алтернативна сценарија

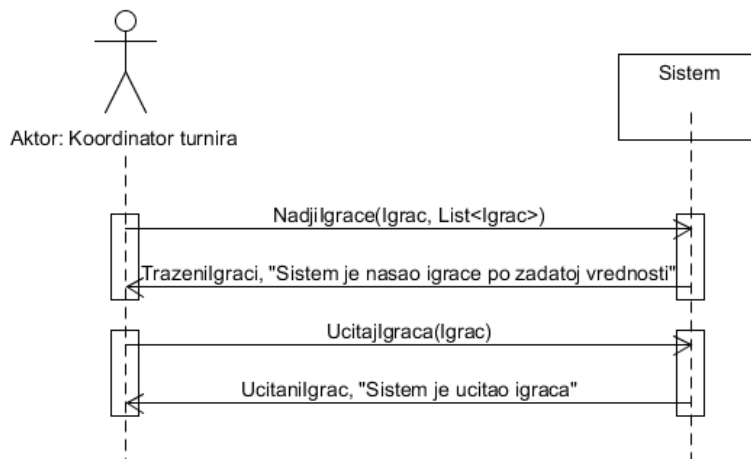
- 6.1 Уколико **систем** не може да запамти податке о **играчу** он приказује **координатору турнира** поруку “**Систем** не може да запамти **играча**”. (ИА)



Са наведених секвенцих дијаграма уочавају се 2 системске операције:

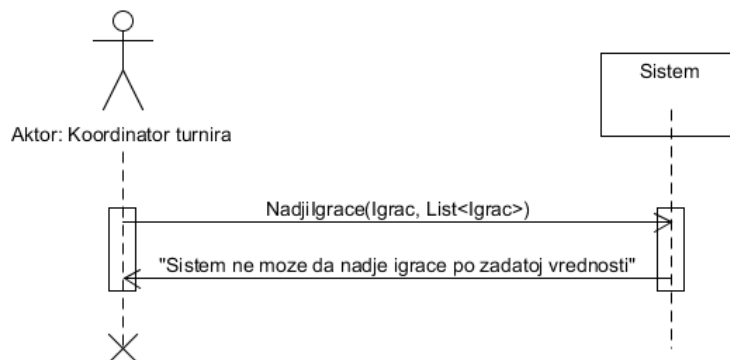
1. Signal **UcitajListuTimova(List<Tim>);**
2. Signal **Zapamtilgraca(lgrac).**

1. **Координатор турнира** **позива** **систем** да нађе **играче** по задатој вредности. (АПСО)
2. **Систем** приказује **координатору турнира** податке о **играчима** и поруку: “**Систем** је нашао **играче** по задатој вредности”. (ИА)
3. **Координатор турнира** **позива** **систем** да учита **играча**. (АПСО)
4. **Систем** приказује **координатору турнира** податке о **играчу** и поруку: “**Систем** је прочитао **играча**”. (ИА)

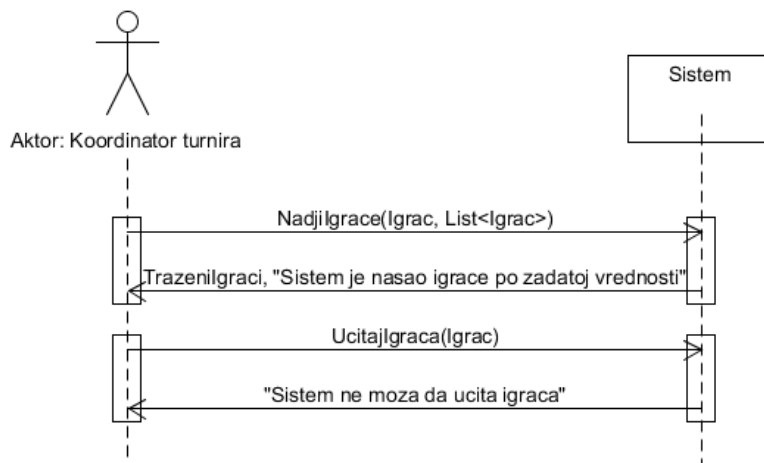


Алтернативна сценарија

- 2.1 Уколико **систем** не може да нађе **играче** он приказује **координатору турнира** поруку: “**Систем** не може да нађе **играче** по задатој вредности”. Прекида се извршење сценарија. (ИА)



- 4.1 Уколико **систем** не може да учита **играча** он приказује **координатору турнира** поруку: “**Систем** не може да учита **играча**”. (ИА)

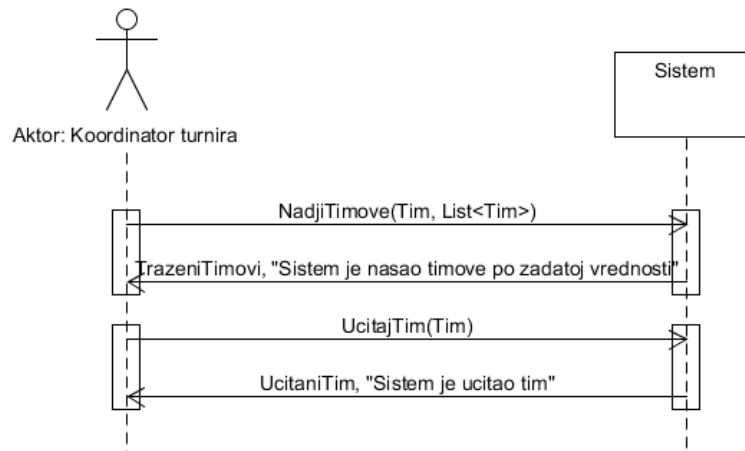


Са наведених секвенцих дијаграма уочавају се 2 системске операције:

1. Signal **NadjiIgrace(Igrac, List<Igrac>);**
2. Signal **UcitajIgraca(Igrac).**

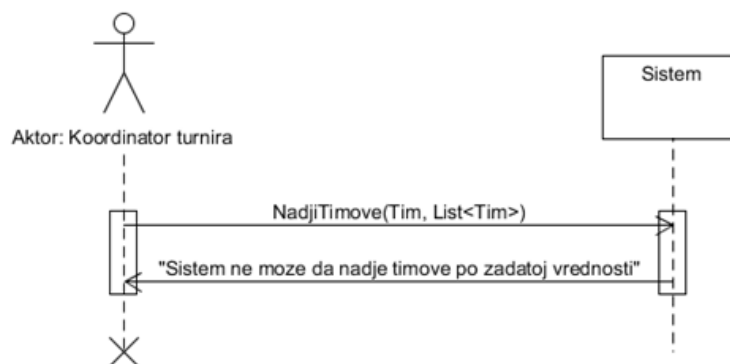
ДС4: Дијаграм секвенце случаја коришћења – Претраживање тима

1. **Координатор турнира** **позива** **систем** да нађе **тимове** по задатој вредности. (АПСО)
2. **Систем** приказује **координатору турнира** податке о **тимовима** и поруку: “**Систем** је нашао **тимове** по задатој вредности”. (ИА)
3. **Координатор турнира** **позива** **систем** да учита **тим**. (АПСО)
4. **Систем** приказује **координатору турнира** податке о **играчу** и поруку: “**Систем** је прочитао **тим**”. (ИА)

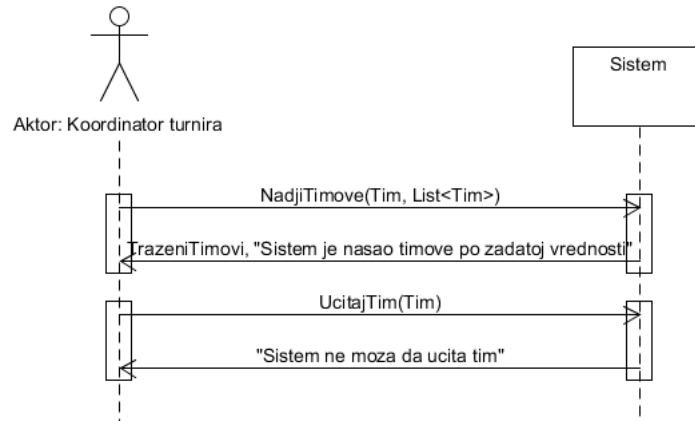


Алтернативна сценарија

- 2.1 Уколико **систем** не може да нађе **тимове** он приказује **координатору турнира** поруку: “**Систем** не може да нађе **тимове** по задатој вредности”. Прекида се извршење сценарија. (ИА)



- 4.1 Уколико **систем** не може да учита **тим** он приказује **координатору турнира** поруку: “**Систем** не може да учита **тим**”. (ИА)

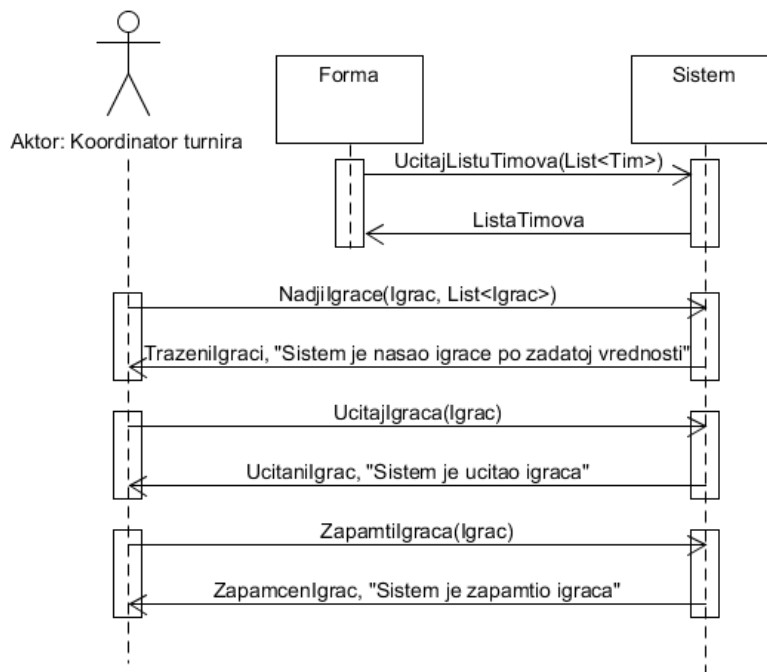


Са наведених секвенцих дијаграма уочавају се 2 системске операције:

1. Signal **NadjiTimove(Tim, List<Tim>);**
2. Signal **UcitajTim(Tim).**

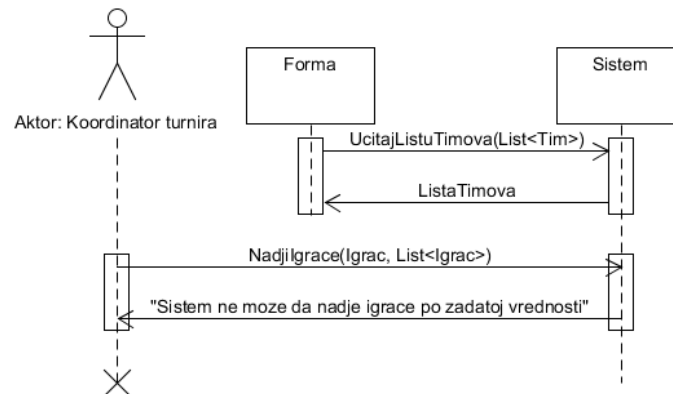
ДС5: Дијаграм секвенце случаја коришћења – Измена података играча

1. **Форма** **позива** **систем** да учита листу тимова. (АПСО)
2. **Систем** **враћа** **форми** листу тимова. (ИА)
3. **Координатор турнира** **позива** **систем** да нађе **играче** по задатој вредности. (АПСО)
4. **Систем** приказује **координатору турнира** **играче** и поруку: “**Систем** је нашао **играче** по задатој вредности”. (ИА)
5. **Координатор турнира** **позива** **систем** да учита **играча**. (АПСО)
6. **Систем** приказује **координатору турнира** податке о **играчу** и поруку: “**Систем** је учитао **играча**”. (ИА)
7. **Координатор турнира** **позива** **систем** да запамти податке о **играчу**. (АПСО)
8. **Систем** **приказује** **координатору турнира** запамћеног **играча** и поруку: “**Систем** је запамтио **играча**.” (ИА)

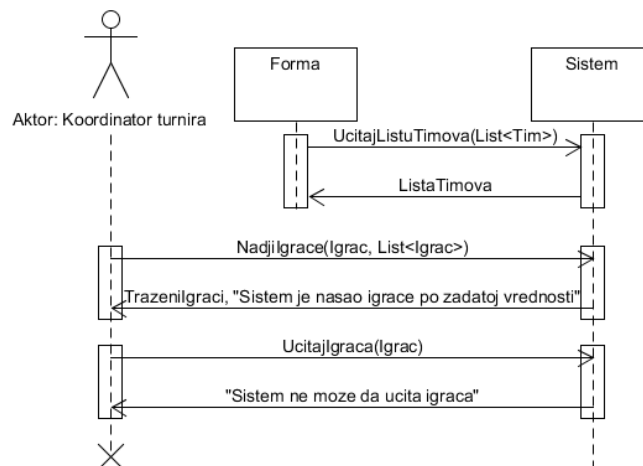


Алтернативна сценарија

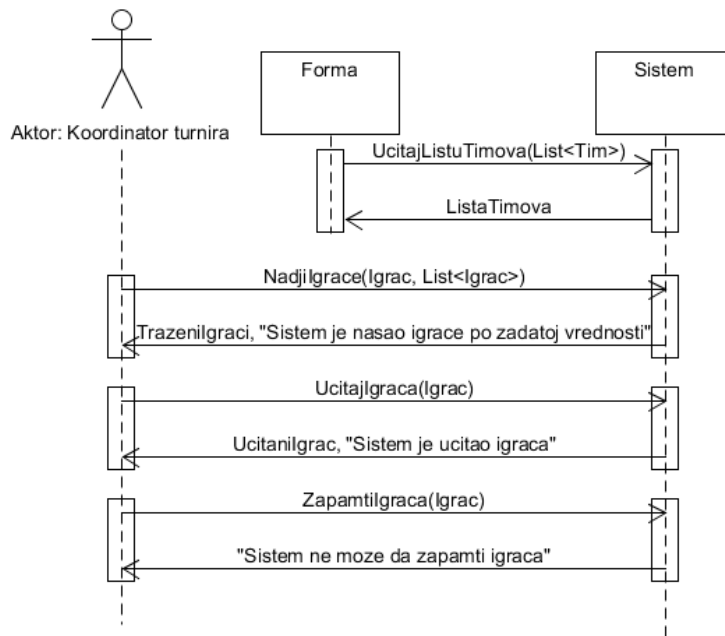
4.1 Уколико **систем** не може да нађе **играче** он приказује **координатору турнира** поруку: “**Систем** не може да нађе **играче** по задатој вредности”. Прекида се извршење сценарија. (ИА)



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



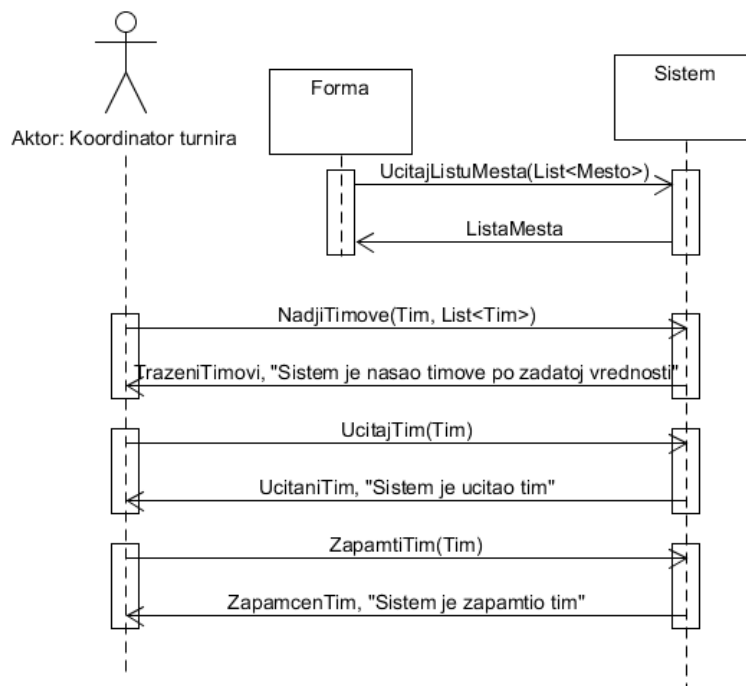
8.1 Уколико **систем** не може да запамти податке о **играчу** он приказује **координатору турнира** поруку “**Систем** не може да запамти **играча**”. (ИА)



Са наведених секвенцих дијаграма уочавају се 4 системске операције:

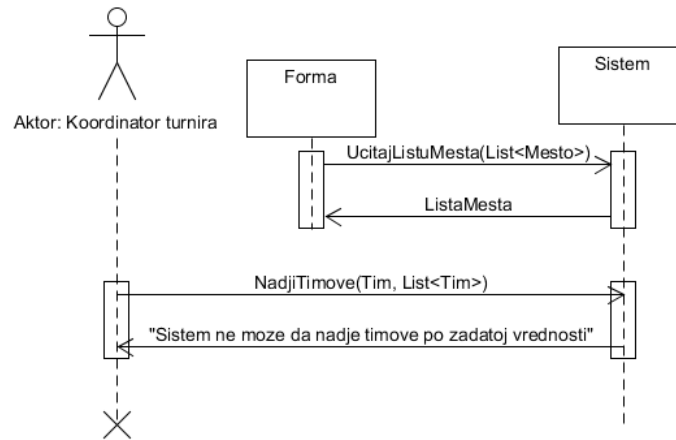
1. Signal **UcitajListuTimova(List<Tim>)**;
2. Signal **NadjiIgrace(Igrac, List<Igrac>)**;
3. Signal **UcitajIgraca(Igrac)**;
4. Signal **ZapamtiIgraca(Igrac)**.

1. **Форма** **позива** **систем** да учита листу места. (АПСО)
2. **Систем** **враћа** **форми** листу места.(ИА)
3. **Координатор турнира** **позива** **систем** да нађе **тимове** по задатој вредности. (АПСО)
4. **Систем** приказује **координатору турнира** **тимове** и поруку: “**Систем** је нашао **тимове** по задатој вредности”. (ИА)
5. **Координатор турнира** **позива** **систем** да учита **тим**. (АПСО)
6. **Систем** приказује **координатору турнира** податке о **тиму** и поруку: “**Систем** је учитао **тим**”. (ИА)
7. **Координатор турнира** **позива** **систем** да запамти податке о **тиму**. (АПСО)
8. **Систем** **приказује** **координатору турнира** запамћени **тим** и поруку: “**Систем** је запамтио **тим**.” (ИА)

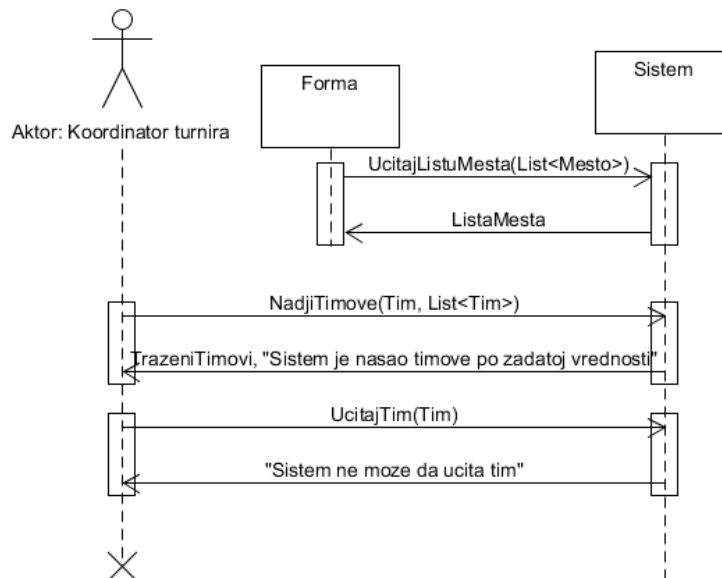


Алтернативна сценарија

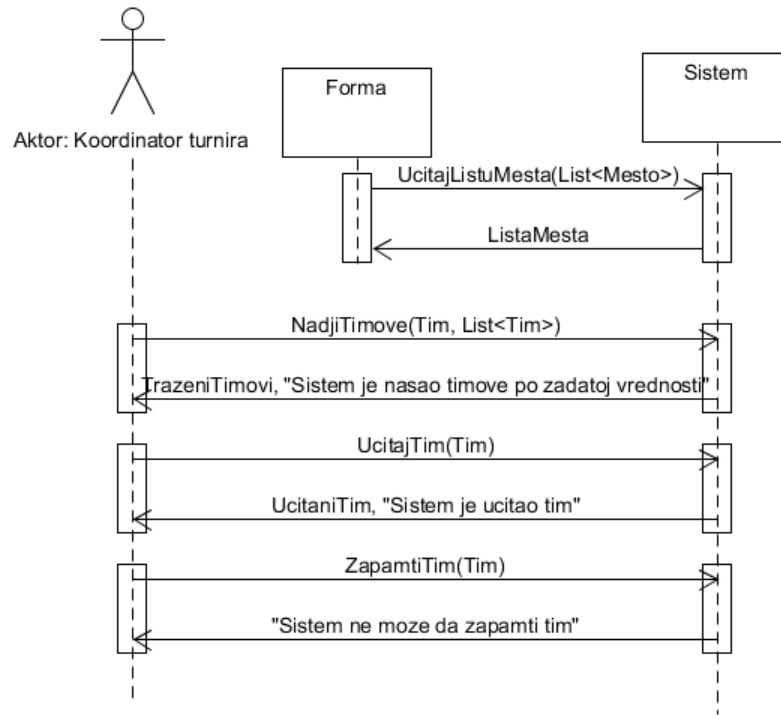
4.1 Уколико **систем** не може да нађе **тимове** он приказује **координатору турнира** поруку: “**Систем** не може да нађе **тимове** по задатој вредности”. Прекида се извршење сценарија. (ИА)



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



8.1 Уколико **систем** не може да запамти податке о **тиму** он приказује **координатору турнира** поруку “**Систем** не може да запамти **тим**”. (ИА)



Са наведених секвенцих дијаграма уочавају се 4 системске операције:

1. Signal **UcitajListuMesta(List<Mesto>);**
2. Signal **NadjiTimove(Tim, List<Tim>);**
3. Signal **UcitajTim(Tim);**
4. Signal **ZapamtiTim(Tim).**

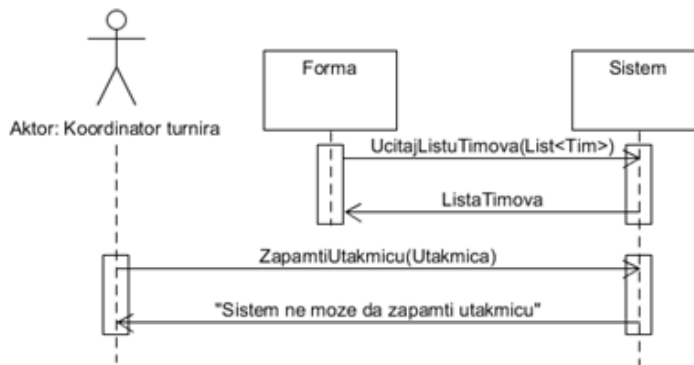
ДС7: Дијаграм секвенце случаја коришћења – Креирање утакмице

1. **Форма** позива **систем** да учита листу тимова. (АПСО)
2. **Систем** враћа **форми** листу тимова. (ИА)
3. **Координатор турнира** позива **систем** да запамти податке о **утакмици**. (АПСО)
4. **Систем** приказује **координатору турнира** запамћени **утакмица** и поруку: “Систем је запамтио **утакмицу**”. (ИА)



Алтернативна сценарија

- 6.1 Уколико **систем** не може да запамти податке о **утакмици** он приказује **координатору турнира** поруку “Систем не може да запамти **утакмицу**”. (ИА)

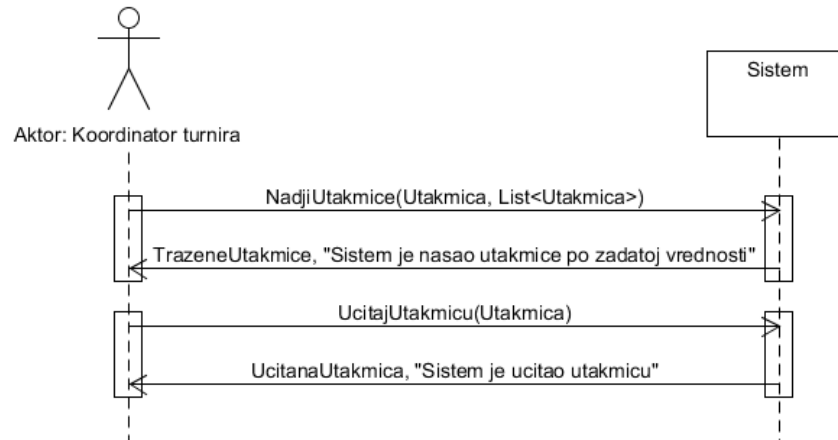


Са наведених секвенчних дијаграма уочавају се 2 системске операције:

1. Signal **UcitajListuTimova(List<Tim>);**
2. Signal **ZapamtiUtkmicu(Utkmica).**

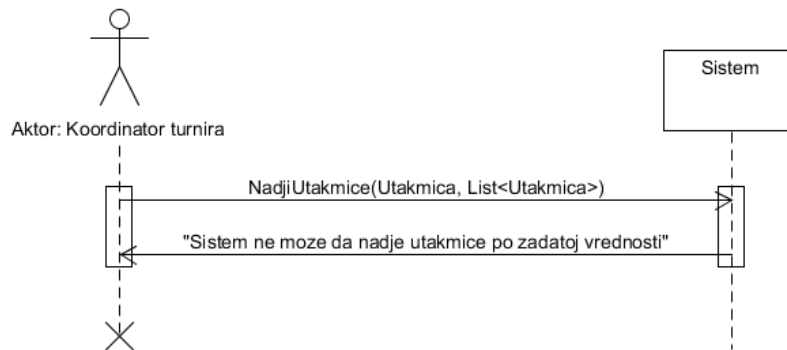
ДС8: Дијаграм секвенце случаја коришћења – Претраживање утакмице

1. **Координатор турнира** **позива систем** да нађе **утакмице** по задатој вредности. (АПСО)
2. **Систем** приказује **координатору турнира** податке о **утакмицама** и поруку: “**Систем** је нашао **утакмице** по задатој вредности”. (ИА)



Алтернативна сценарија

- 2.1 Уколико **систем** не може да нађе **утакмице** он приказује **координатору турнира** поруку: “**Систем** не може да нађе **утакмице** по задатој вредности”. Прекида се извршење сценарија. (ИА)



Са наведених секвенчних дијаграма уочавају се једна системска операција:

1. Signal **NadjiUtkmice(Utakmica, List<Utkmica>);**

На основу анализе сценарија добијено је 10 системских операција:

1. Signal **ZapamtiTim**(Tim);
2. Signal **ZapamtiIgraca**(Igrac);
3. Signal **Nadjilgrace**(Igrac, List<Igrac>);
4. Signal **UcitajIgraca**(Igrac);
5. Signal **NadjiTimove**(Tim, List<Tim>);
6. Signal **UcitajTim**(Tim);
7. Signal **ZapamtiUtakmicu**(Utakmica);
8. Signal **NadjiUtakmice**(Utakmica, List<Utakmica>);
9. Signal **UcitajListuMesta**(List<Mesto>);
10. Signal **UcitajListuTimova**(List<Tim>).

4.2.2. Понашање софтверског система – Дефинисање уговора о системским операцијама

Уговор УГ1: ZapamtiTim(Tim) Signal;

Веза са СК: СК1, СК6

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

Постуслови: Подаци о тиму су запамћени.

Уговор УГ2: ZapamtiIgraca(Igrac) Signal;

Веза са СК: СК2, СК5

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

Постуслови: Подаци о играчу су запамћени.

Уговор УГ3: NadjiIgrace(Igrac, List<Igrac>) Signal;

Веза са СК: СК3, СК5

Предуслови:

Постуслови:

Уговор УГ4: UcitajIgraca(Igrac) Signal;

Веза са СК: СК3, СК5

Предуслови:

Постуслови:

Уговор УГ5: NadjiTimove(Tim, List<Tim>) Signal;

Веза са СК: СК4, СК6

Предуслови:

Постуслови:

Уговор УГ6: UcitajTim(Tim) Signal;

Веза са СК: СК4, СК6

Предуслови:

Постуслови:

Уговор УГ7: ZapamtiUtakmicu(Utakmica) Signal;

Веза са СК: СК7

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

Постуслови: Подаци о утакмици су запамћени.

Уговор УГ8: NadjiUtakmice(Utakmica, List<Utakmica>) Signal;

Веза са СК: СК8

Предуслови:

Постуслови:

Уговор УГ9: UcitajListuMesta(List<Mesto>) Signal;

Веза са СК: СК1, СК6

Предуслови:

Постуслови:

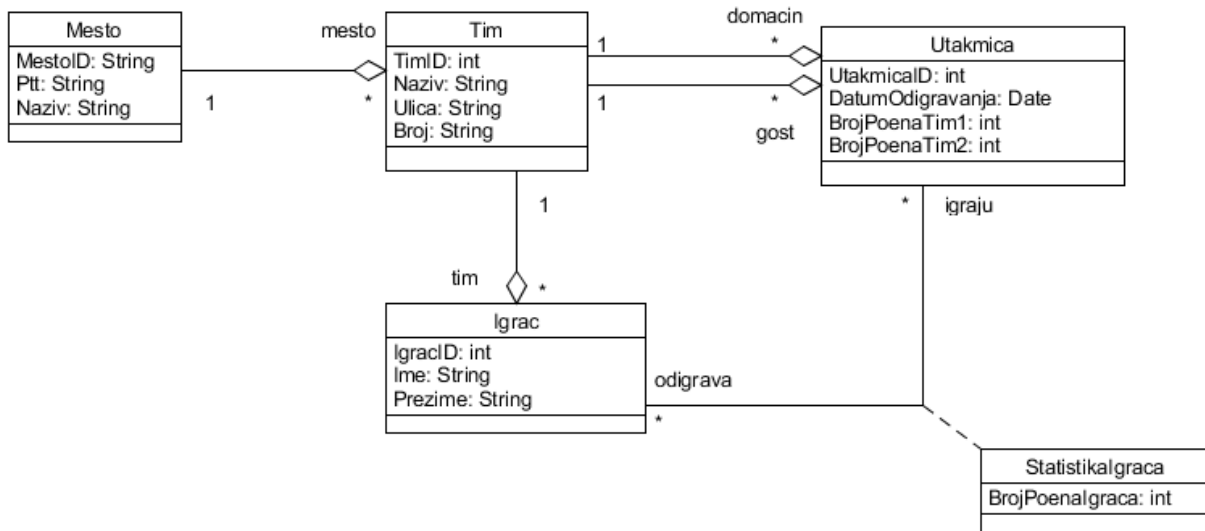
Уговор УГ10: UcitajListuTimova(List<Tim>) Signal;

Веза са СК: СК2, СК5, СК7

Предуслови:

Постуслови:

4.2.3. Структура софтверског система – Концептуални(доменски) модел



Слика 28. Концептуални модел

4.2.4. Структура софтверског система – релациони модел

Tim(TimID, Naziv, Ulica, Broj, MestoID)

Igrac(IgracID, Ime, Prezime, TimID)

Mesto(MestoID, Ptt, Naziv)

Utakmica(UtakmicaID, DatumOdigravanja, BrojPoenaTim1, BrojPoenaTim2, TimID, TimID2)

Statistikalgraca(IgracID, UtakmicaID, BrojPoenaIgraca)

| Tabela Tim | | Prosto vrednosno ograničenje | | Složeno vrednosno ograničenje | | Strukturno ograničenje |
|------------|---------|------------------------------|--------------------|-------------------------------------|------------------------------------|--|
| Atributi | Ime | Tip atributa | Vrednost atributa | Međuzavisnost atributa jedne tabele | Međuzavisnost atributa više tabela | INSERT RESTRICTED Mesto |
| | TimID | Integer | not null and >0 | | | UPDATE CASCADES Igrac, Utakmica |
| | Naziv | String | not null | | | |
| | Ulica | String | not null | | | |
| | Broj | String | not null | | | |
| | MestoID | String | not null | | | UPDATE RESTRICTED Mesto DELETE CASCADE Igrac, Utakmica |

| Tabela Igrac | | Prosto vrednosno ograničenje | | Složeno vrednosno ograničenje | | Strukturno ograničenje |
|--------------|---------|------------------------------|--------------------|-------------------------------------|------------------------------------|-----------------------------|
| Atributi | Ime | Tip atributa | Vrednost atributa | Međuzavisnost atributa jedne tabele | Međuzavisnost atributa više tabela | INSERT RESTRICTED Tim |
| | IgracID | Integer | not null and >0 | | | |

| | | | | | | |
|--|---------|---------|--------------------|--|--|--|
| | Ime | String | not null | | | UPDATE RESTRICTED Tim |
| | Prezime | String | not null | | | |
| | TimID | Integer | not null and >0 | | | UPDATE CASCADES Statistikalgraca DELETE CASCADES Statistikalgraca |

| Tabela Utakmica | | Prosto vrednosno ograničenje | | Složeno vrednosno ograničenje | | Strukturno ograničenje |
|-----------------|------------------|------------------------------|--------------------|-------------------------------------|------------------------------------|--|
| Atributi | Ime | Tip atributa | Vrednost atributa | Međuzavisnost atributa jedne tabele | Međuzavisnost atributa više tabela | INSERT RESTRICTED Tim |
| | UtakmicaID | Integer | not null and >0 | | | UPDATE RESTRICTED Tim |
| | DatumOdigravanja | Date | not null | | | |
| | TimID | Integer | not null and >0 | | | UPDATE CASCADES Statistikalgraca |
| | TimID2 | Integer | not null and >0 | | | |

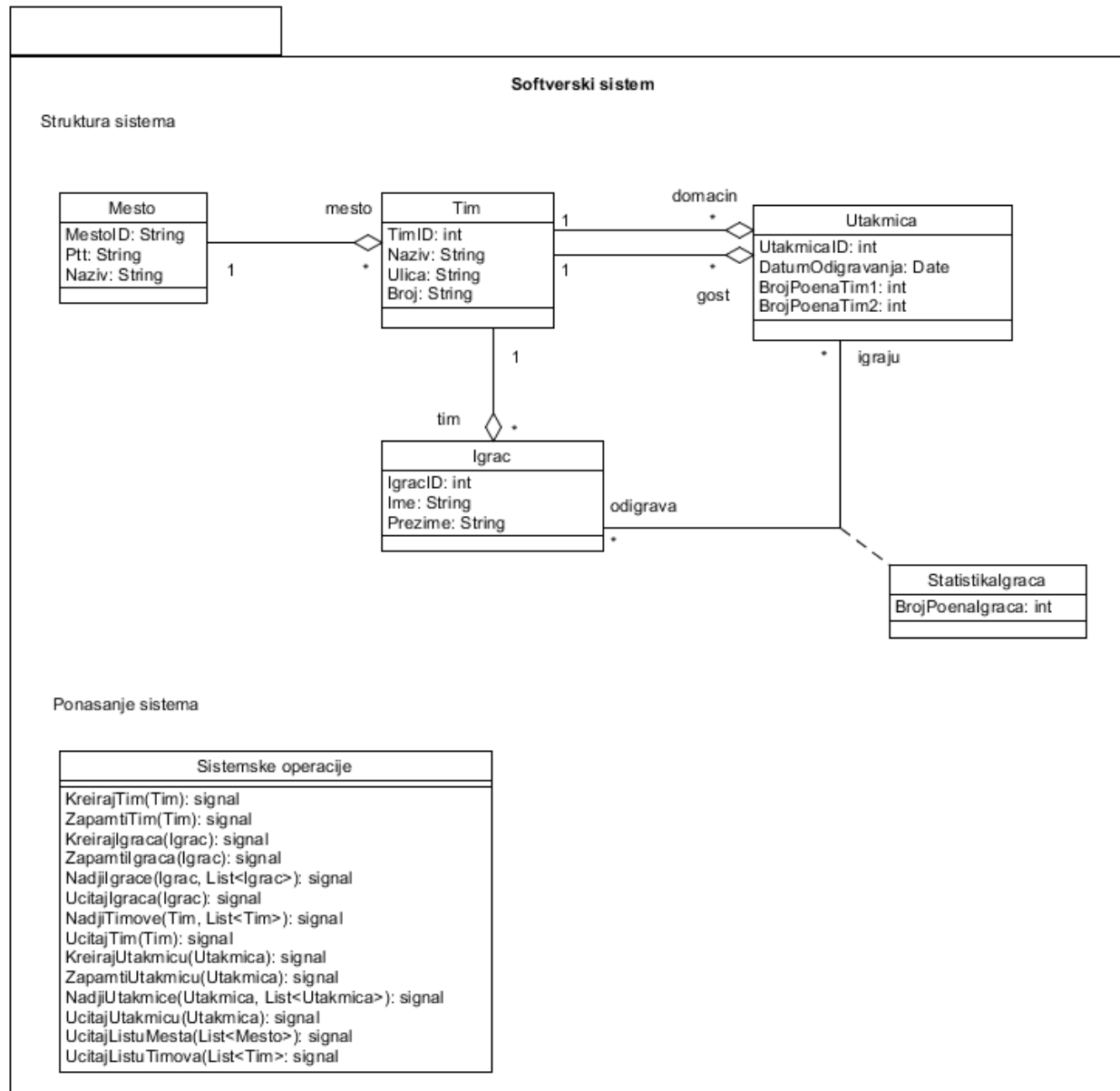
| | | | | | | |
|--|---------------|---------|-------------------|--|--|--|
| | BrojPoenaTim1 | Integer | >0 (default:0) | | BrojPoenaTim1 = SUM(Statistikalgraca, BrojPoenalgraca) | DELETE CASCADES Statistikalgraca |
| | BrojPoenaTim2 | Integer | >0 (default:0) | | BrojPoenaTim2 = SUM(Statistikalgraca, BrojPoenalgraca) | |

| Tabela Statistikalgraca | | Prosto vrednosno ograničenje | | Složeno vrednosno ograničenje | | Strukturno ograničenje |
|-------------------------|-----------------|---------------------------------|----------------------|---|--|--|
| Atributi | Ime | Tip atributa | Vrednost atributa | Međuzavisnost atributa jedne tabele | Međuzavisnost atributa više tabela | INSERT RESTRICTED Igrac, Utakmica |
| | IgraciD | Integer | not null and >0 | | | UPDATE RESTRICTED Igrac, Utakmica |
| | UtakmicaID | Integer | not null and >0 | | | |
| | BrojPoenalgraca | Integer | >0 (default:0) | | | DELETE / |

| Tabela Mesto | Prosto vrednosno ograničenje | | Složeno vrednosno ograničenje | | Strukturno ograničenje |
|--------------|---------------------------------|--|-------------------------------|--|---------------------------|
|--------------|---------------------------------|--|-------------------------------|--|---------------------------|

| Atributi | Ime | Tip atributa | Vrednost atributa | Međuzavisnost atributa jedne tabele | Međuzavisnost atributa više tabela | INSERT / UPDATE CASCADES Tim |
|----------|---------|--------------|-------------------|-------------------------------------|------------------------------------|---|
| | MestoID | String | not null | | | DELETE RESTRICTED Tim |
| | Ptt | String | not null | | | |
| | Naziv | String | not null | | | |

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



4.3. Пројектовање

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

Архитектура софтверског система

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

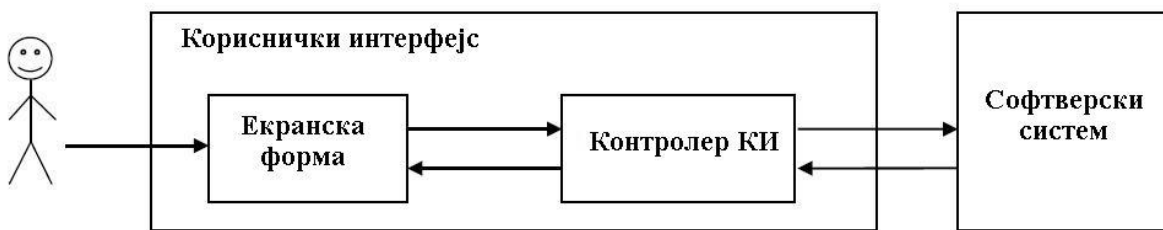
- Кориснички интерфејс
- Апликациона логика
- Складиште података

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



4.3.1. Пројектовање корисничког интерфејса

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



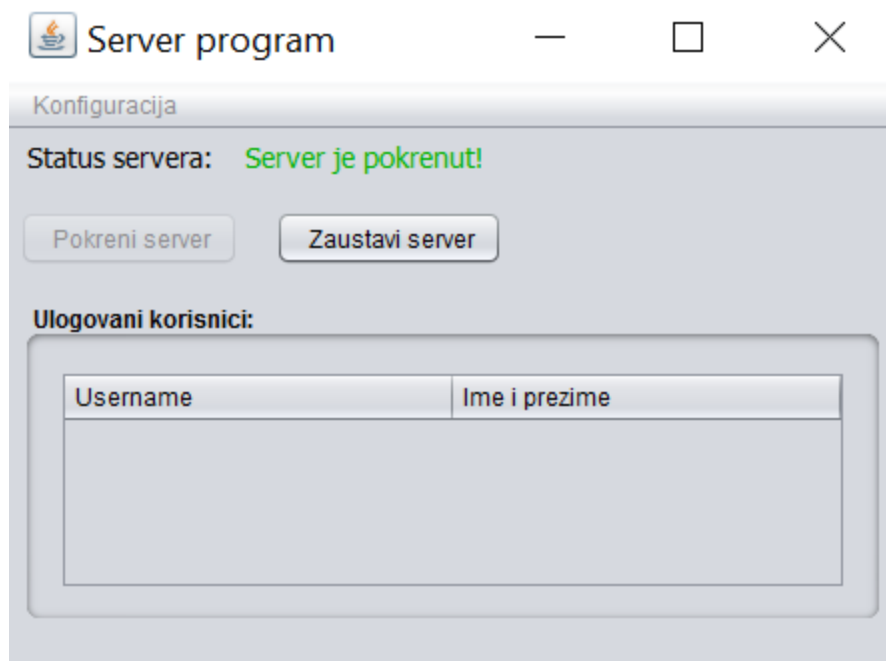
4.3.1.1. Пројектовање екранских форми

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

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

The screenshot shows a window titled 'Server program' with standard Windows window controls. The window contains a 'Konfiguracija' (Configuration) section. Under this section, the 'Status servera:' (Server status:) is displayed in red text as 'Server nije pokrenut!' (Server is not running!). Below the status, there are two buttons: 'Pokreni server' (Start server) and 'Zaustavi server' (Stop server). Further down, there is a section titled 'Ulogovani korisnici:' (Logged-in users:). This section contains a table with two columns: 'Username' and 'Ime i prezime' (Name and surname). The table is currently empty.

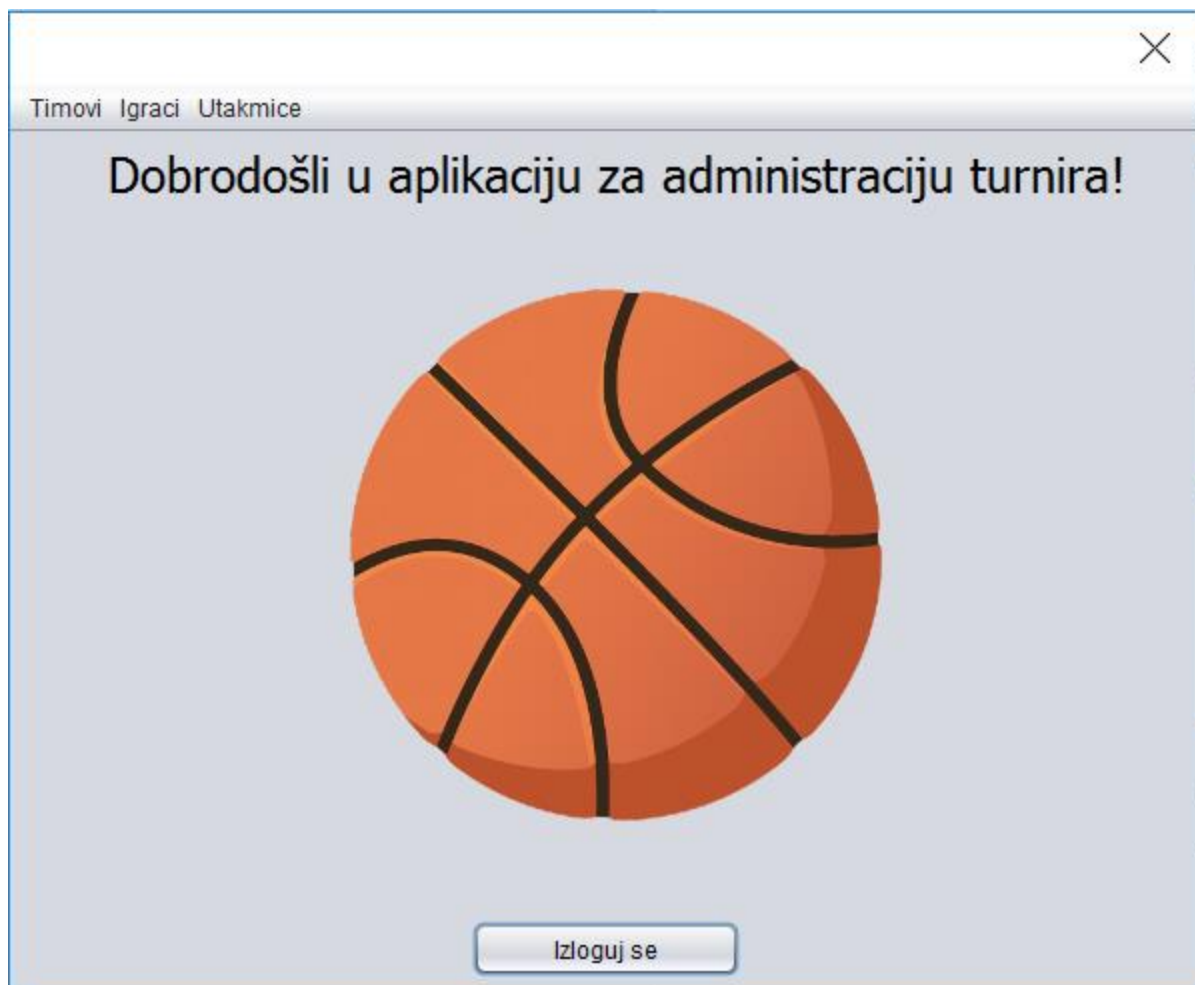
Након активације, серверска форма изгледа овако:



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

The image shows a window titled 'Uloguj se' with a standard Windows-style title bar. The main content area contains two labels: 'Korisnicko ime:' and 'Sifra:'. Each label is followed by a text input field. Below these fields is a button labeled 'Uloguj se'.

Након логовања, систем приказује клијенту главну екранску форму из које се може доћи до осталих екранских форми.



СК1: Случај коришћења – Креирање тима

Назив СК

Креирање тима

Актори СК

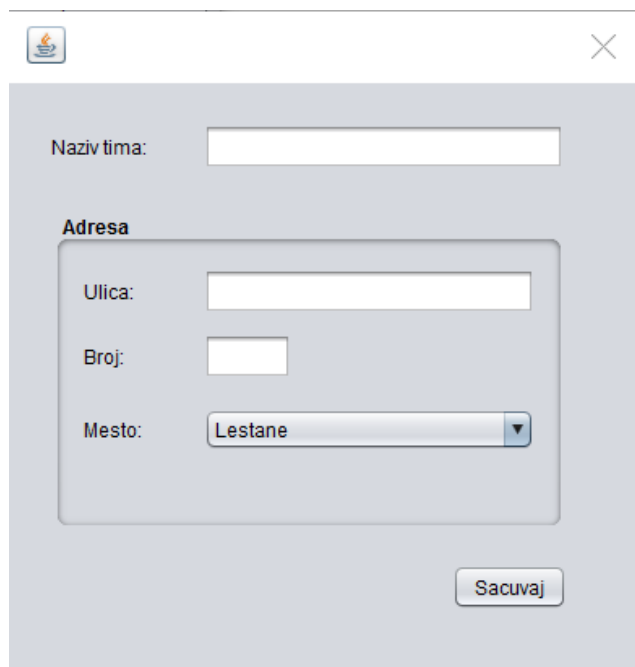
Координатор турнира

Учесници СК

Координатор турнира и систем (програм)

Предуслов: Систем је укључен и координатор турнира је улогован под својом шифром.

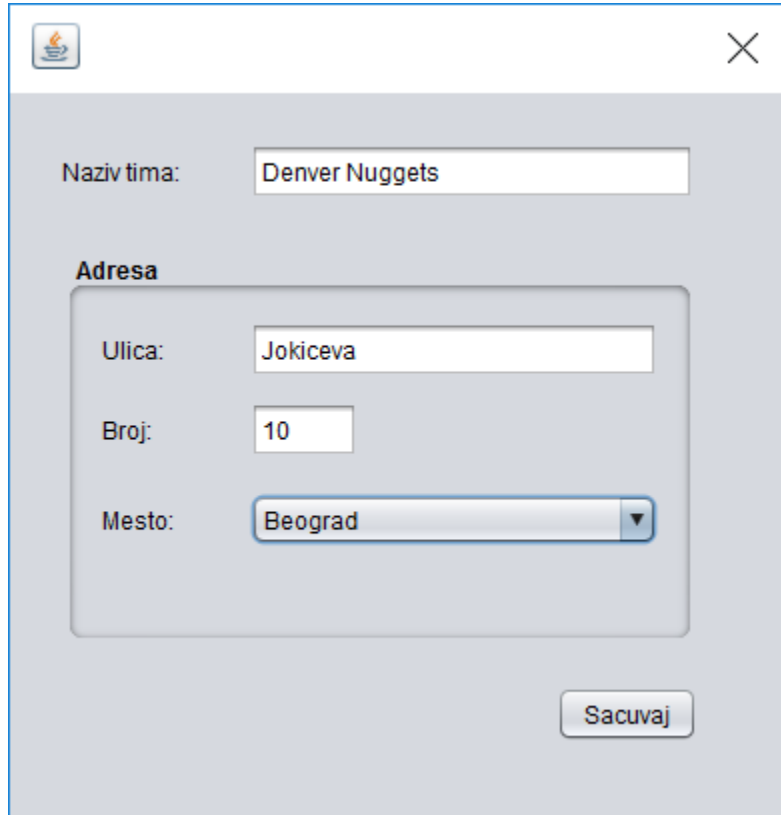
Учитана је листа места.



The screenshot shows a web form titled "Naziv tima:" with a text input field. Below it is a section titled "Adresa" containing three fields: "Ulica:" with a text input field, "Broj:" with a text input field, and "Mesto:" with a dropdown menu showing "Lestane". A "Sacuvaj" button is located at the bottom right of the form.

Основни сценарио СК

1. **Координатор турнира уноси** податке у **тим**. (АПУСО)



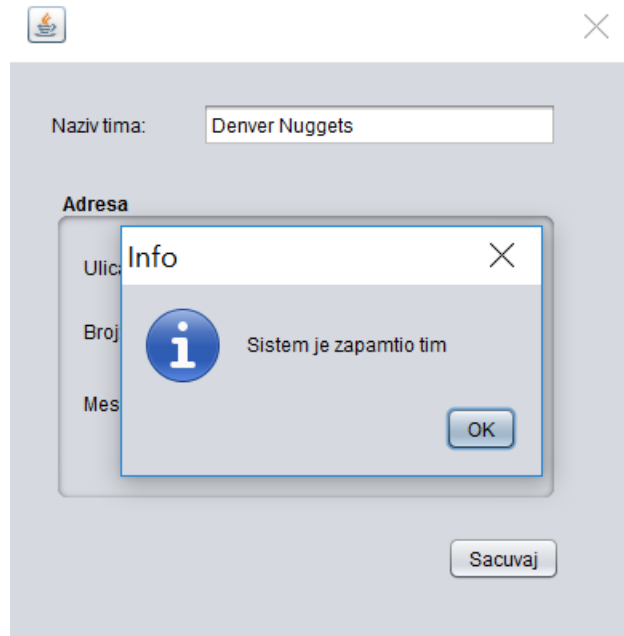
The screenshot shows a software window with a title bar containing a small icon and a close button. The window has a light gray background. It contains the following elements:

- A label "Naziv tima:" followed by a text input field containing "Denver Nuggets".
- A section titled "Adresa" containing a group box with three fields:
 - A label "Ulica:" followed by a text input field containing "Jokiceva".
 - A label "Broj:" followed by a text input field containing "10".
 - A label "Mesto:" followed by a dropdown menu showing "Beograd".
- A button labeled "Sacuvaj" at the bottom right.

2. **Координатор турнира контролише** да ли је коректно унео податке у **тим**. (АНСО)
3. **Координатор турнира позива систем** да запамти податке о **тиму**. (АПСО)

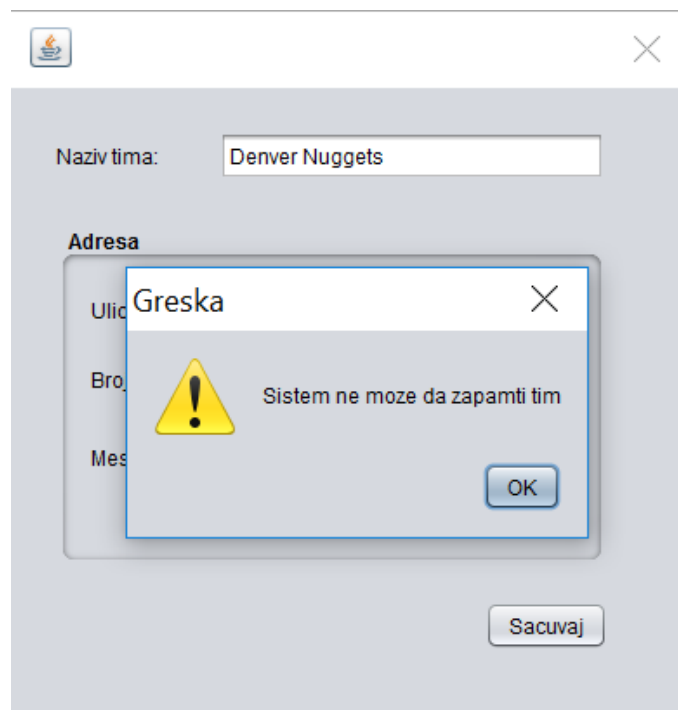
Опис акције: Координатор кликом на дугме „Сачувај“ позива системску операцију **ZapamtiTim(Tim)**

4. **Систем памти** податке о **тиму**. (СО)
5. **Систем приказује координатору турнира** запамћени **тим** и поруку: „Систем је запамтио **тим**“. (ИА)



Алтернативна сценарија

- 5.1 Уколико **систем** не може да запамти податке о **тиму** он приказује **координатору турнира** поруку “**Систем** не може да запамти **тим**”. (ИА)



СК2: Случај коришћења – Креирање играча

Назив СК

Креирање играча

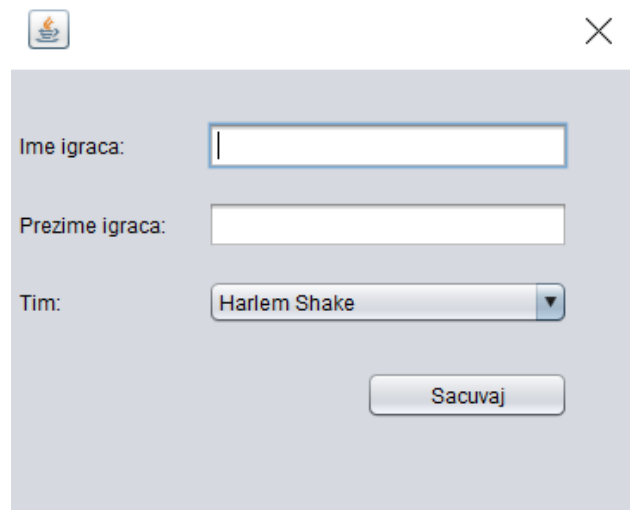
Актори СК

Координатор турнира

Учесници СК

Координатор турнира и систем (програм)

Предуслов: Систем је укључен и координатор турнира је улогован под својом шифром. Систем приказује форму за рад са играчем. Учитана је листа тимова.



The screenshot shows a software window titled with a small icon and a close button (X). Inside the window, there is a form for creating a player. It contains three input fields: 'Ime igraca:' (Player Name) with a text box, 'Prezime igraca:' (Player Surname) with a text box, and 'Tim:' (Team) with a dropdown menu currently showing 'Harlem Shake'. Below these fields is a button labeled 'Sacuvaj' (Save).

Основни сценарио СК

1. Координатор турнира уноси податке у играча. (АПУСО)

Ime igrača: Darko

Prezime igrača: Cmiljanić

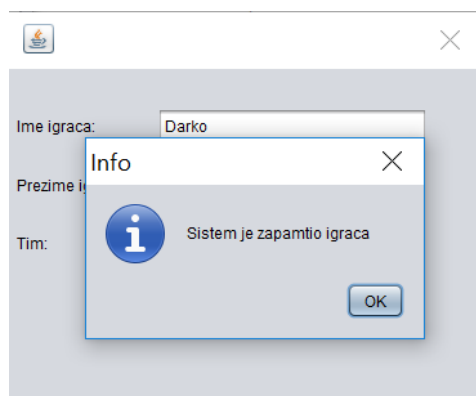
Tim: Harlem Shake

Sacuvaj

2. **Координатор турнира контролише** да ли је коректно унео податке у **играча**. (АНСО)
3. **Координатор турнира позива систем** да запамти податке о **играчу**. (АПСО)

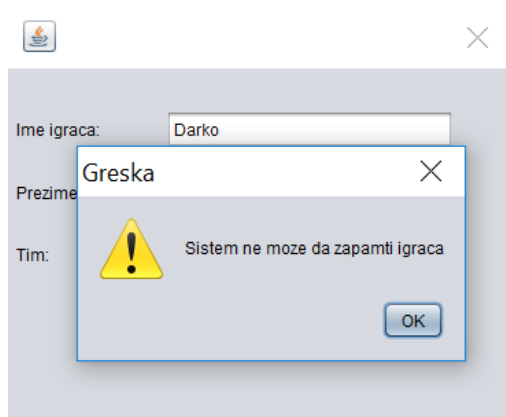
Опис акције: Координатор кликом на дугме „Сачувај“ позива системску операцију **Zapamtilgraca(Igrac)**

4. **Систем памти** податке о **играчу**. (СО)
5. **Систем приказује координатору турнира** запамћеног **играча** и поруку: **“Систем је запамтио играча”**. (ИА)



Алтернативна сценарија

- 5.1 Уколико **систем** не може да запамти податке о **играчу** он приказује **координатору турнира** поруку **“Систем не може да запамти играча”**. (ИА)



СКЗ: Случај коришћења – Претраживање играча

Назив СК

Претраживање играча

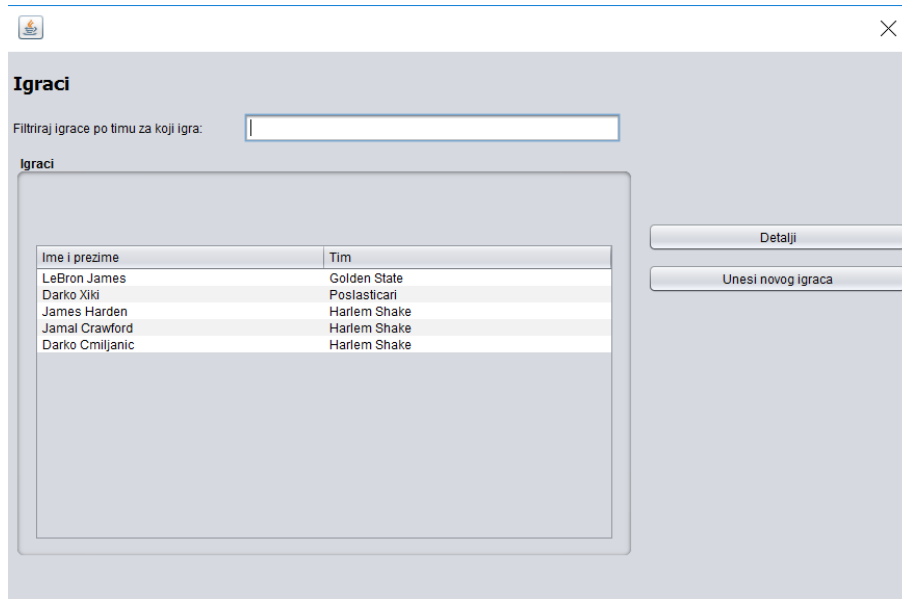
Актори СК

Координатор турнира

Учесници СК

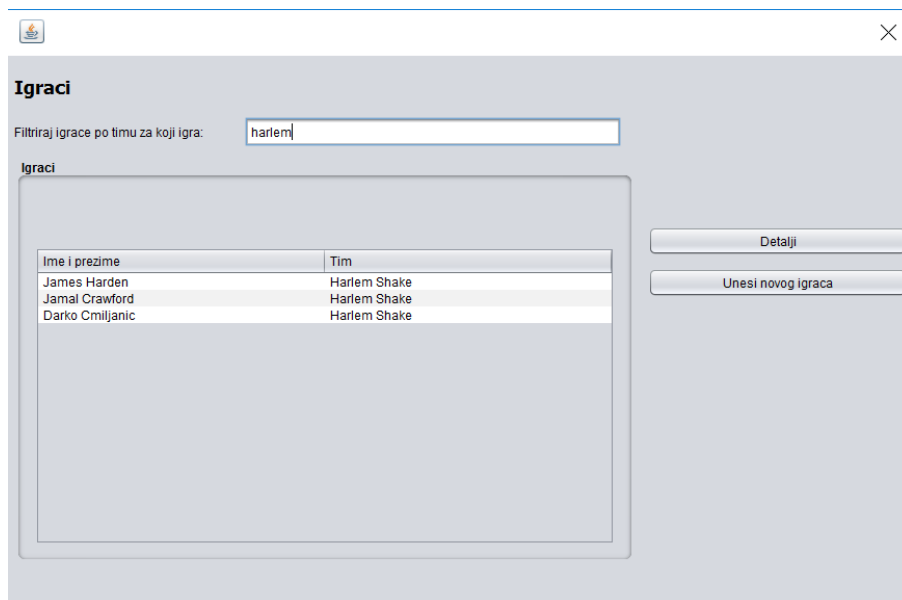
Координатор турнира и систем (програм)

Предуслов: Систем је укључен и координатор турнира је улогован под својом шифром. Систем приказује форму за рад са играчем.



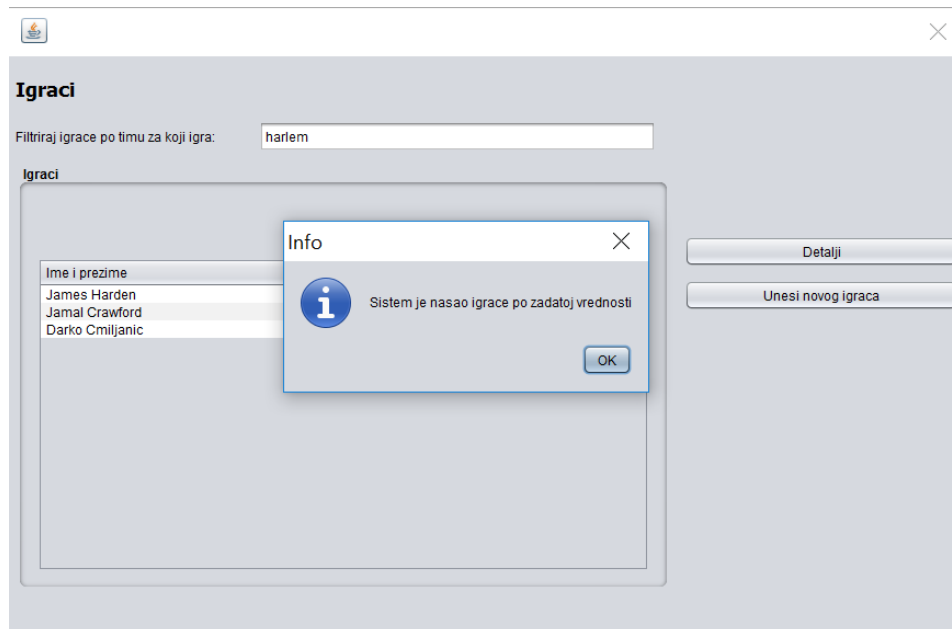
Основни сценарио СК

1. **Координатор турнира** уноси вредност по којој претражује **играче**. (АПУСО)

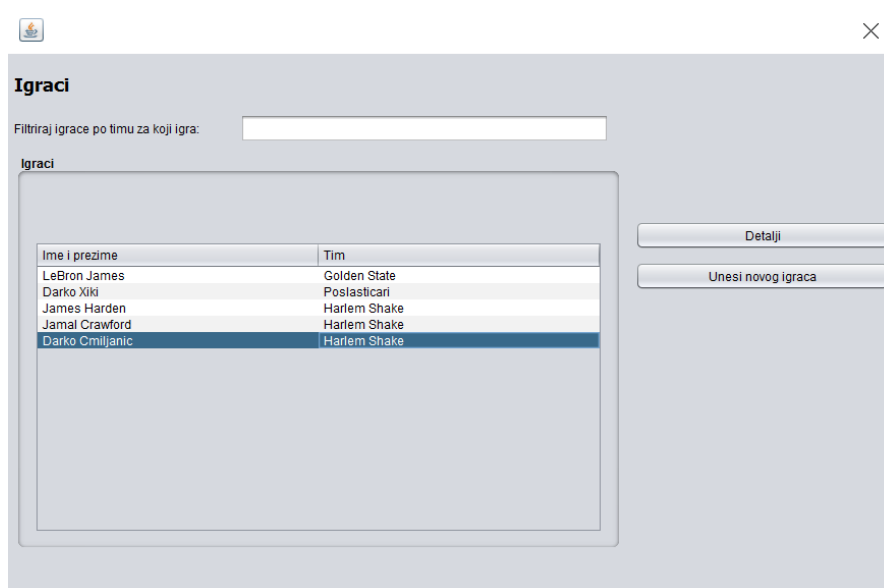


2. **Координатор турнира** позива **систем** да нађе **играче** по задатој вредности. (АПСО)
Опис акције: Координатор притиском типке тастатуре позива системску операцију **Nadjilgrace(Igrac,List<Igrac>)**

3. Систем **тражи** **играче** по задатој вредности. (CO)
4. Систем приказује **координатору турнира** податке о **играчима** и поруку: “Систем је нашао **играче** по задатој вредности”. (ИА)



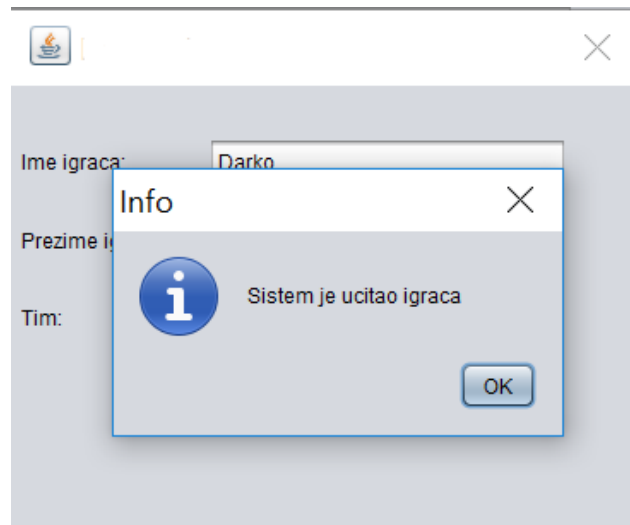
5. **Координатор турнира бира играча.** (АПУСО)



6. **Координатор турнира позива систем** да учита **играча**. (АПСО)

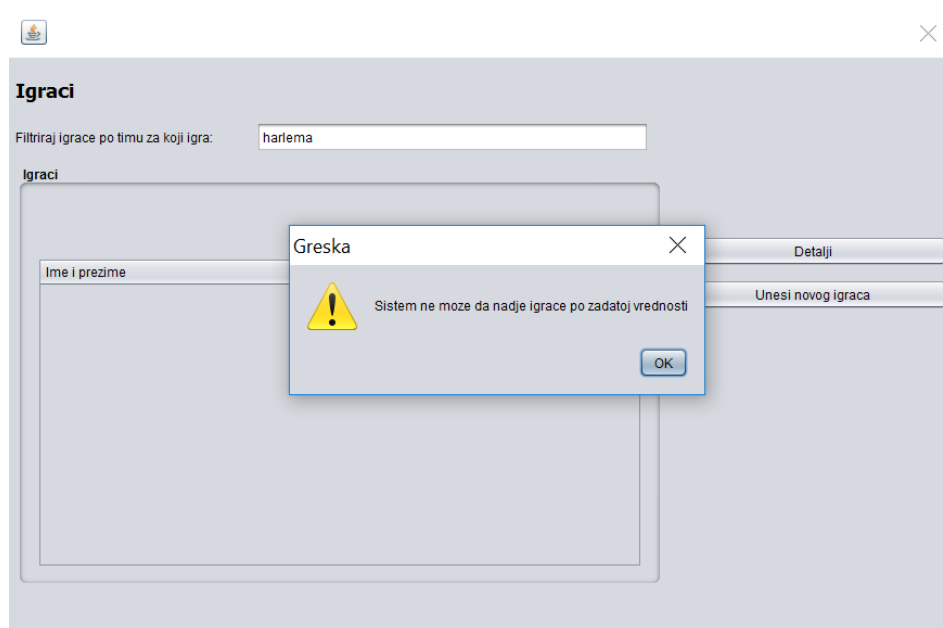
Опис акције: Координатор кликом на дугме „Детаљи“ позива системску операцију **UcitajIgraca(Igrac)**

7. Систем учитава играча. (CO)
8. Систем приказује координатору турнира податке о играчу и поруку: “Систем је прочитао играча”. (ИА)

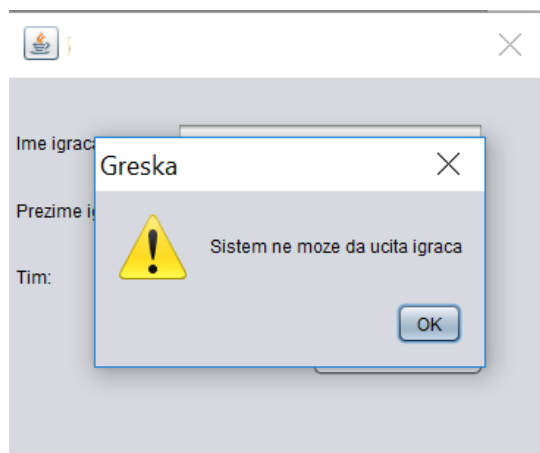


Алтернативна сценарија

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



- 8.1 Уколико систем не може да прочита играча он приказује координатору турнира поруку: “Систем не може да прочита играча”. (ИА)



СК4: Случај коришћења – Претраживање тима

Назив СК

Претраживање тима


Актори СК

Координатор турнира

Учесници СК

Координатор турнира и систем (програм)

Предуслов: Систем је укључен и координатор турнира је улогован под својом шифром. Систем приказује форму за рад са тимом.


×

Timovi

Filtriraj timove po nazivu:

Timovi


| Naziv | Ulica i broj | Mesto |
|-------------------|------------------|----------|
| Harlem Shake | Cikijeva 5 | Lestane |
| Golden State | Oracle Arena 5 | Beograd |
| Denver Nuggets | Jokicevaa 10 | Beograd |
| Novi Sad Al Wahda | Laze Kostica 5 | Novi Sad |
| Denver Nuggets | Jokiceva 10 | Novi Sad |
| Poslastican | Vojvode Stepe 21 | Niš |

Detalji

Unesi novi tim

Основни сценарио СК

1. **Координатор турнира уноси** вредност по којој претражује **тимове**. (АПУСО)


×

Timovi

Filtriraj timove po nazivu:

Timovi

| Naziv | Ulica i broj | Mesto |
|--------------|----------------|---------|
| Golden State | Oracle Arena 5 | Beograd |

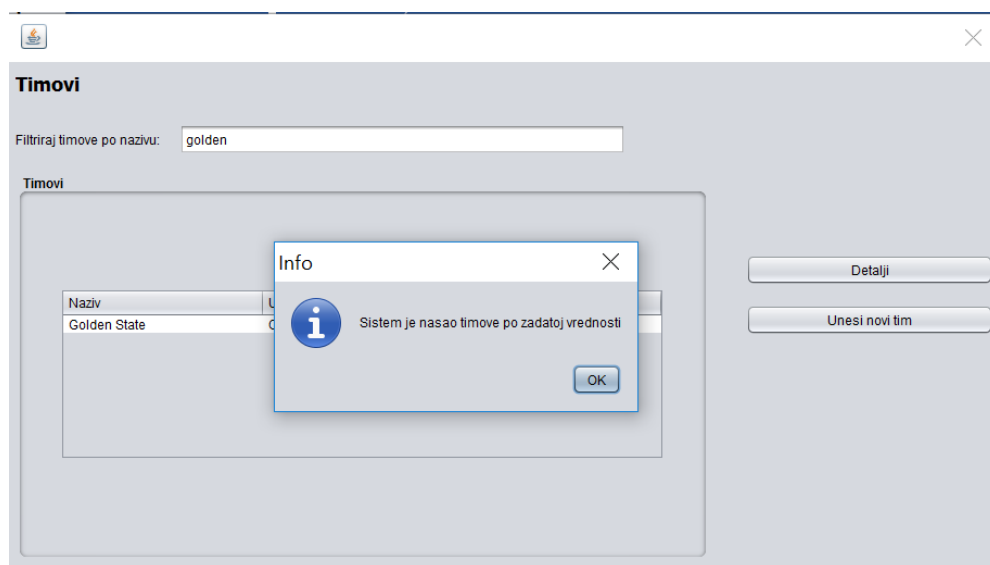
Detalji

Unesi novi tim

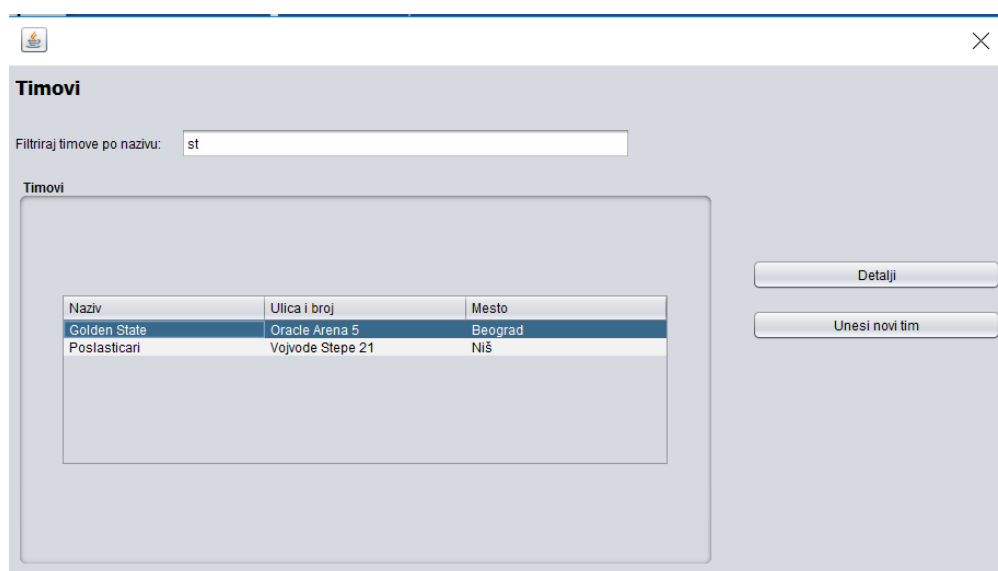
2. **Координатор турнира** **позива** **систем** да нађе **тимове** по задатој вредности. (АПСО)

Опис акције: Координатор притиском типке тастатуре позива системску операцију **Nadjilgrace(Igrac,List<Igrac>)**

3. **Систем** **тражи** **тимове** по задатој вредности. (СО)
4. **Систем** приказује **координатору турнира** податке о **тимовима** и поруку: “**Систем** је нашао **тимове** по задатој вредности”. (ИА)



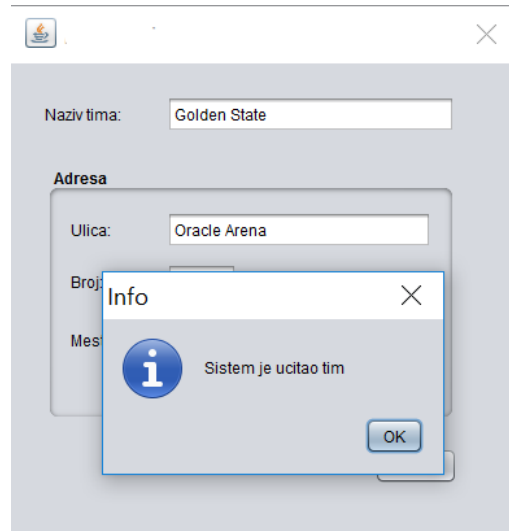
5. **Координатор турнира** **бира** **тим**. (АПУСО)



6. **Координатор турнира** позива **систем** да учита **тим**. (АПСО)

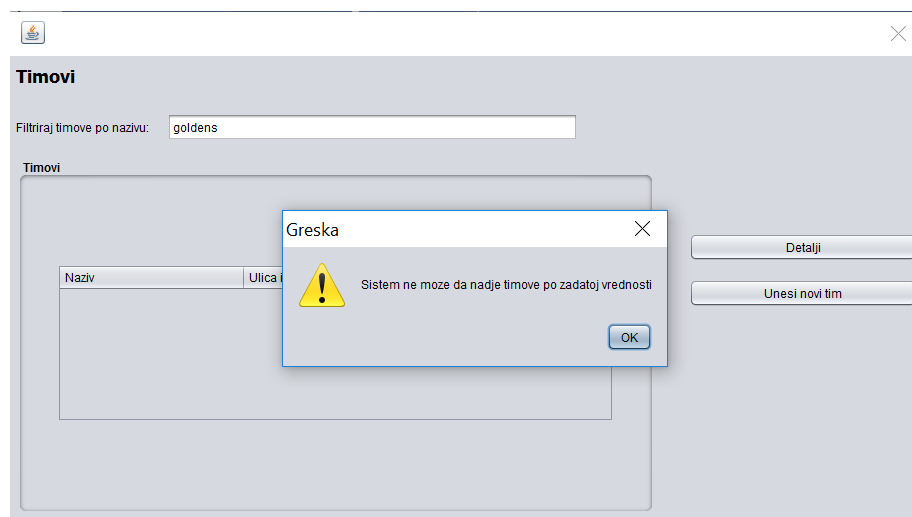
Опис акције: Координатор кликом на дугме „Детаљи“ позива системску операцију **UcitajTim(Tim)**

7. **Систем** **учитава тим**. (СО)
8. **Систем** приказује **координатору турнира** податке о **играчу** и поруку: “**Систем** је учитао **тим**”. (ИА)

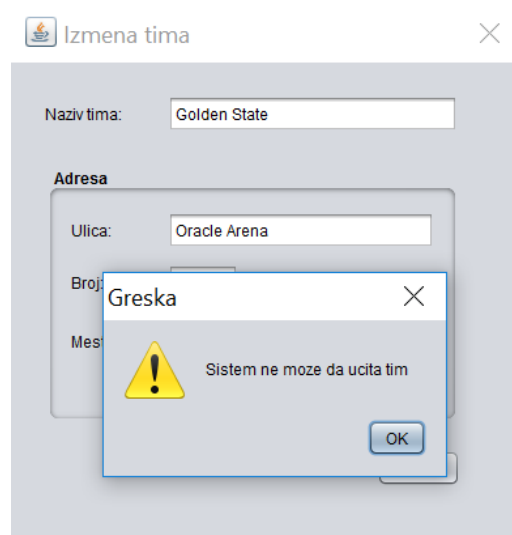


Алтернативна сценарија

- 4.1 Уколико **систем** не може да нађе **тимове** он приказује **координатору турнира** поруку: “**Систем** не може да нађе **тимове** по задатој вредности”. Прекида се извршење сценарија. (ИА)



8.1 Уколико **систем** не може да учита **тим** он приказује **координатору турнира** поруку: “**Систем** не може да учита **тим**”. (ИА)



СК5: Случај коришћења – Измена података играча

Назив СК

Измена података **играча**

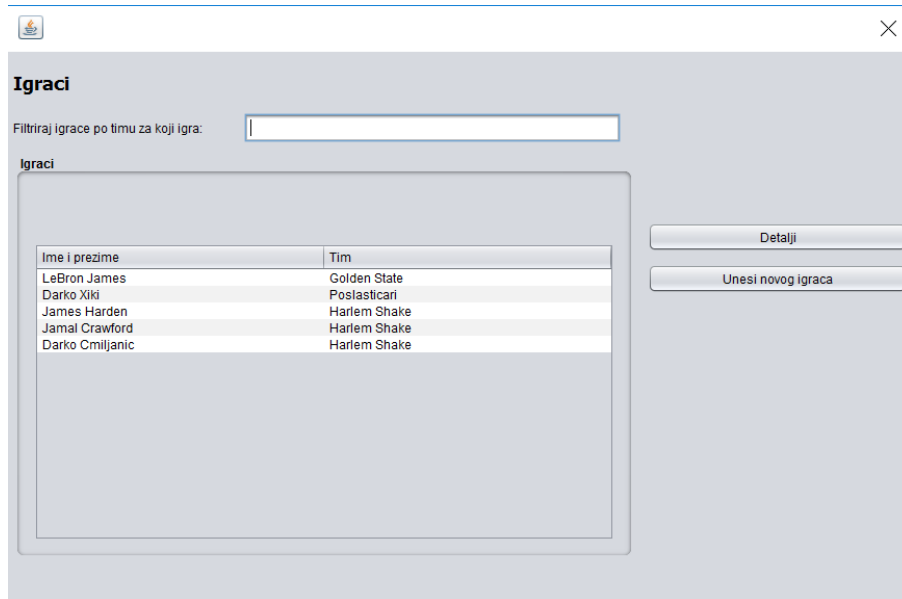
Актори СК

Координатор турнира

Учесници СК

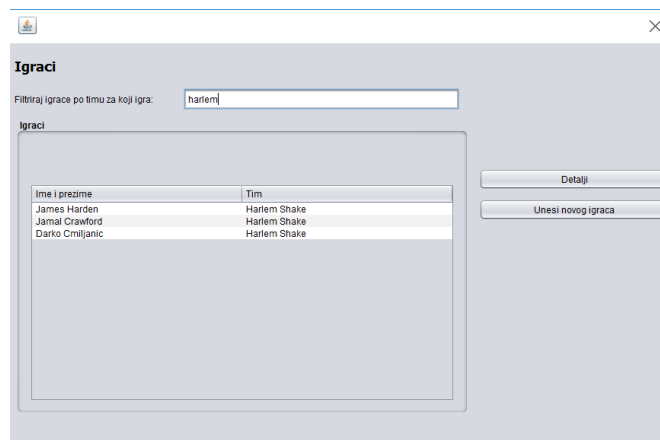
Координатор турнира и **систем** (програм)

Предуслов: **Систем** је укључен и **координатор турнира** је улогован под својом шифром. Систем приказује форму за рад са **играчем**. Учитана је листа тимова.



Основни сценарио СК

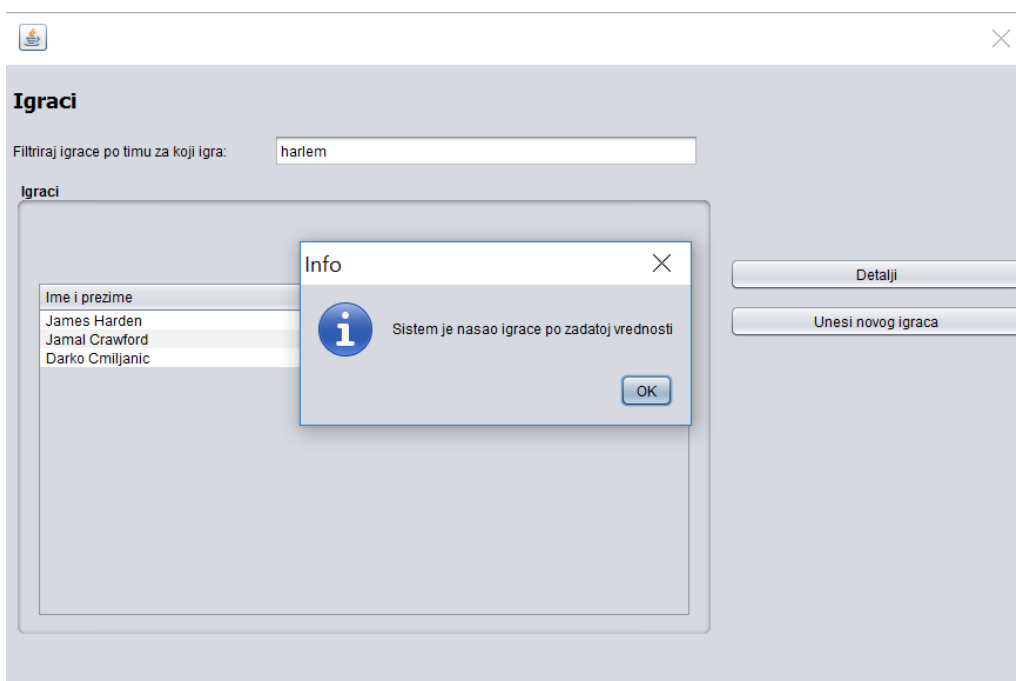
1. **Координатор турнира** уноси вредност по којој претражује **играче**. (АПУСО)



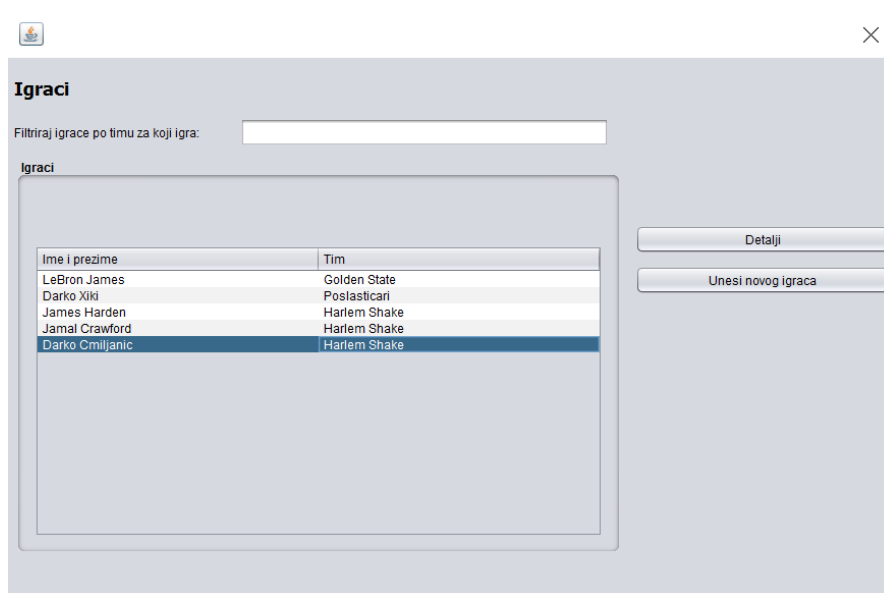
2. **Координатор турнира** позива **систем** да нађе **играче** по задатој вредности. (АПСО)

Опис акције: Координатор притиском типке тастатуре позива системску операцију **Nadjilgrace(Igrac,List<Igrac>)**

3. **Систем** тражи **играче** по задатој вредности. (СО)
4. **Систем** приказује **координатору турнира** податке о **играчима** и поруку: “**Систем** је нашао **играче** по задатој вредности”. (ИА)



5. **Координатор турнира бира играча.** (АПУСО)

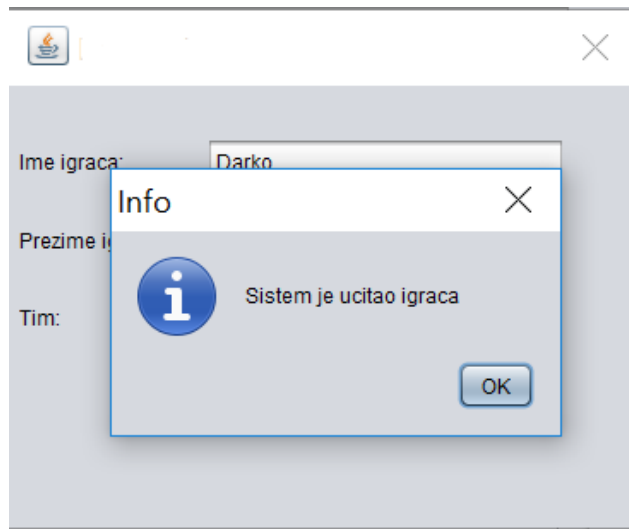


6. **Координатор турнира позива систем** да учита **играча.** (АПСО)

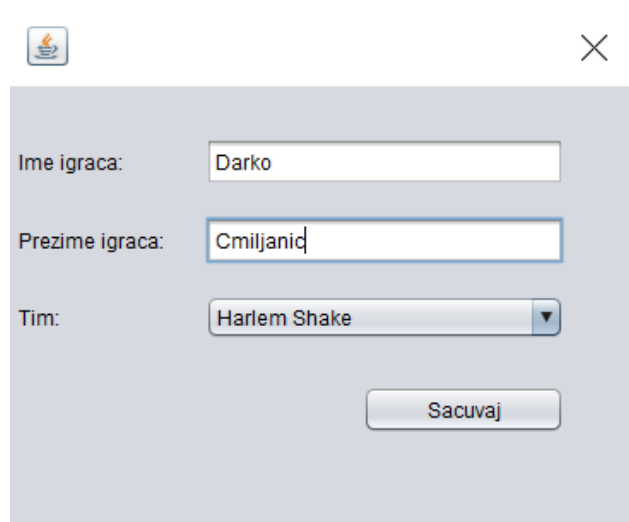
Опис акције: Координатор кликом на дугме „Детаљи“ позива системску операцију **UcitajIgraca(Igrac)**

7. **Систем учитава играча.** (СО)

8. **Систем** приказује **координатору турнира** податке о **играчу** и поруку: “**Систем** је прочитао играча”. (ИА)



9. **Координатор турнира** уноси податке у **играча**. (АПУСО)

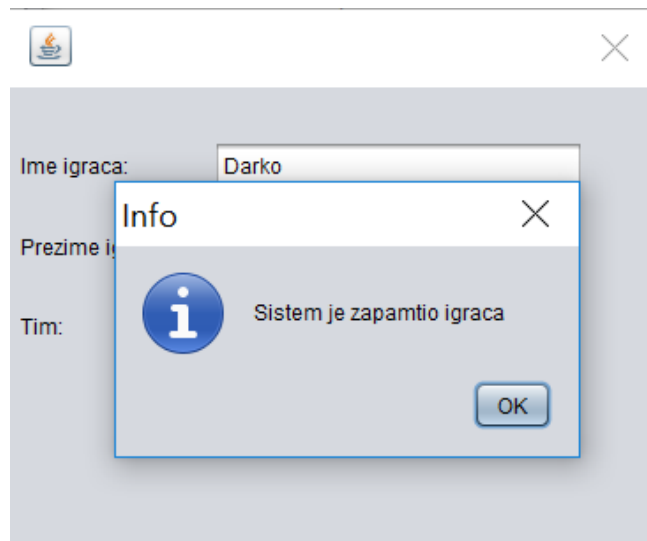


10. **Координатор турнира** **контролише** да ли је коректно унео податке у **играча**. (АНСО)
11. **Координатор турнира** **позива** **систем** да запамти податке о **играчу**. (АПСО)

Опис акције: Координатор кликом на дугме „Измени“ позива системску операцију **Zapamtilgraca(Igrac)**

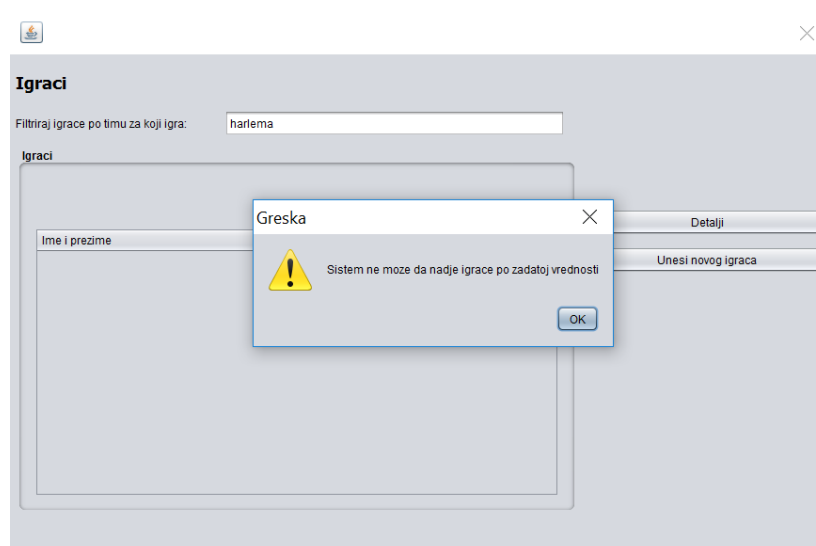
12. **Систем** **памти** податке о **играчу**. (СО)

13. Систем приказује координатору турнира запамћеног играча и поруку: “Систем је запамтио играча”. (ИА)

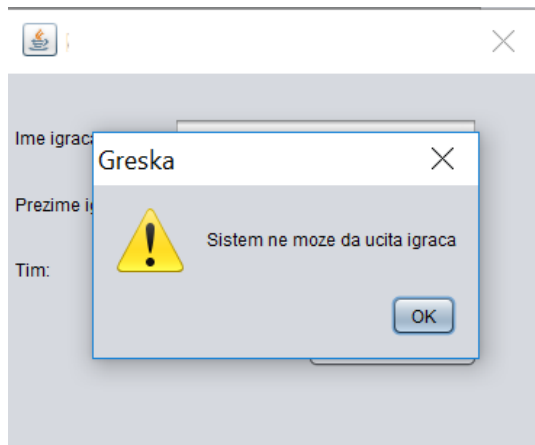


Алтернативна сценарија

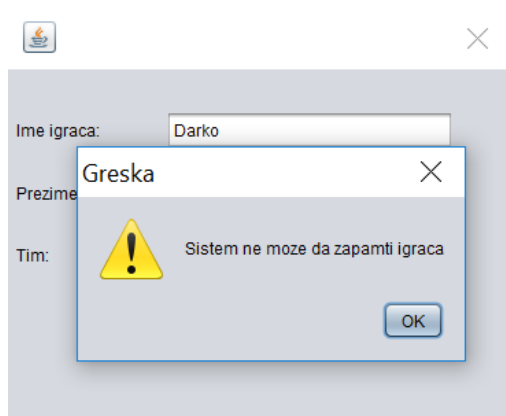
- 4.1 Уколико систем не може да нађе играче он приказује координатору турнира поруку: “Систем не може да нађе играче по задатој вредности”. Прекида се извршење сценарија. (ИА)



8.1 Уколико **систем** не може да учита **играча** он приказује **координатору турнира** поруку: “**Систем** не може да учита **играча**”. (ИА)



13.1 Уколико **систем** не може да запамти податке о **играчу** он приказује **координатору турнира** поруку “**Систем** не може да запамти **играча**”. (ИА)



СК6: Случај коришћења – Измена података тима

Назив СК

Измена података **тима**

Актори СК

Координатор турнира

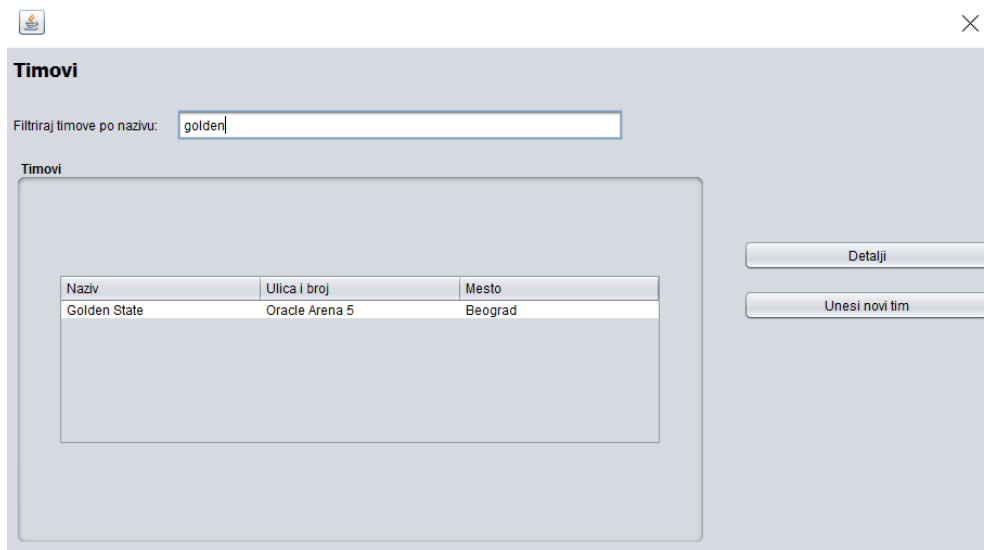
Учесници СК

Координатор турнира и **систем** (програм)

Предуслов: **Систем** је укључен и **координатор турнира** је улогован под својом шифром. Систем приказује форму за рад са **тимом**. Учитана је листа места.

Основни сценарио СК

1. **Координатор турнира** **уноси** вредност по којој претражује **тимове**. (АПУСО)



Timovi

Filtriraj timove po nazivu:

Timovi

| Naziv | Ulica i broj | Mesto |
|--------------|----------------|---------|
| Golden State | Oracle Arena 5 | Beograd |

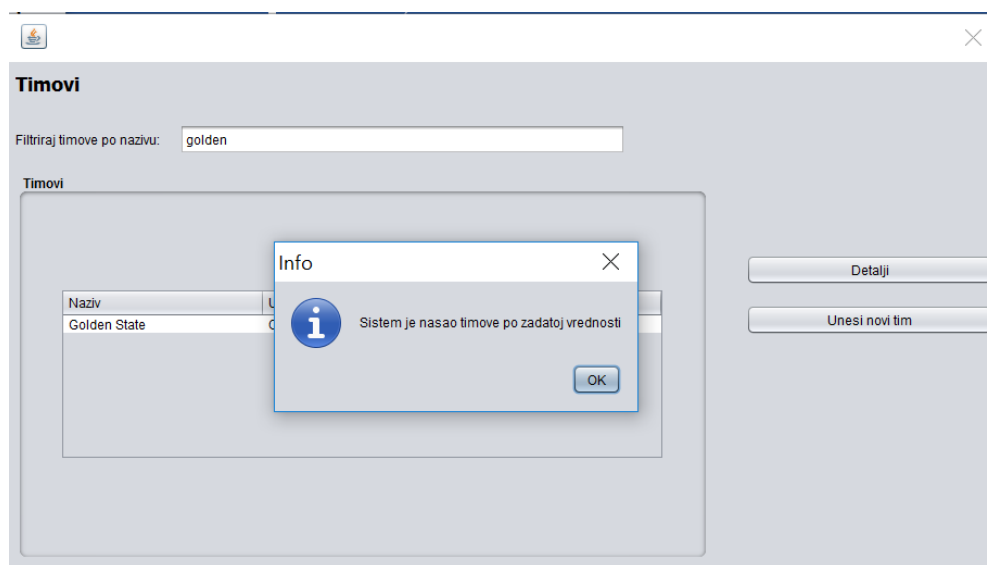
Detalji

Unesi novi tim

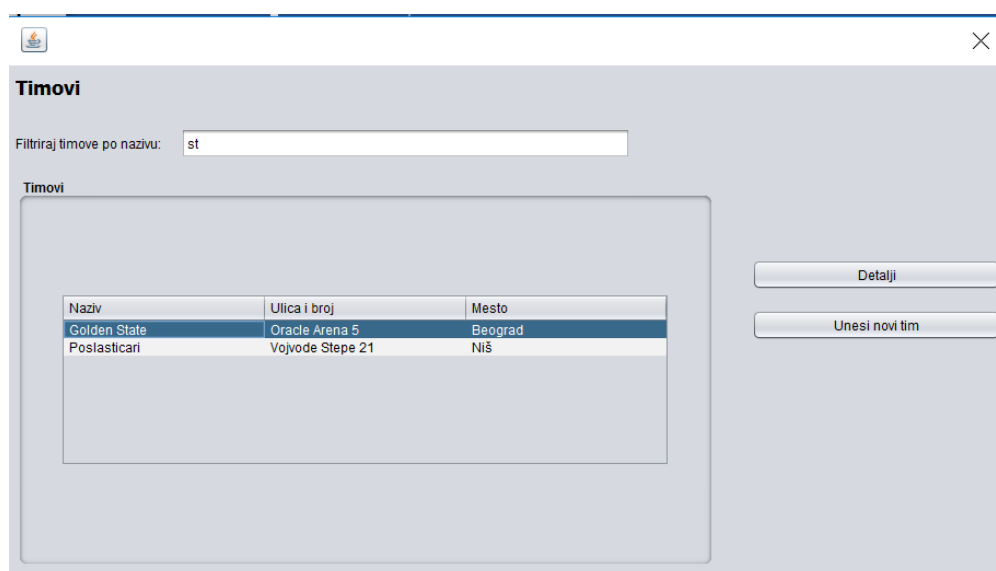
2. **Координатор турнира** **позива** **систем** да нађе **тимове** по задатој вредности. (АПСО)

Опис акције: Координатор притиском типке тастатуре позива системску операцију **NadjiTimove(Tim,List<Tim>)**

3. **Систем** **тражи** **тимове** по задатој вредности. (СО)
4. **Систем** приказује **координатору турнира** податке о **тимовима** и поруку: “**Систем** је нашао **тимове** по задатој вредности”. (ИА)



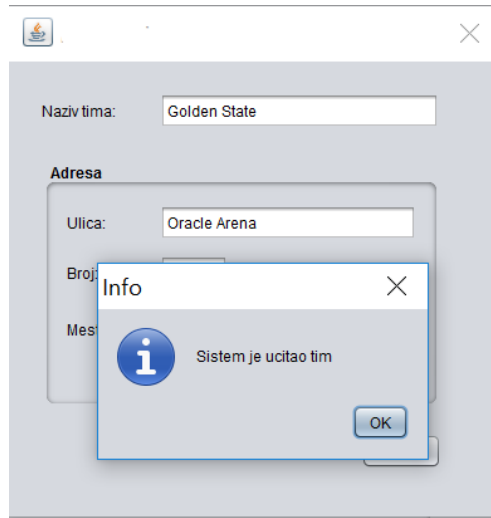
5. **Координатор турнира** **бира** **тим**. (АПУСО)



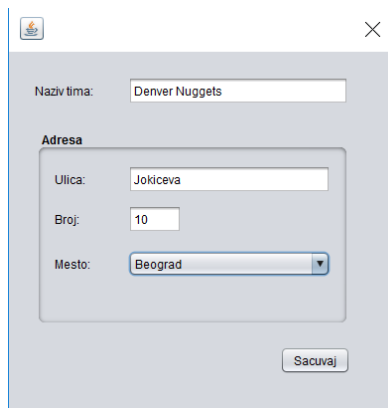
6. **Координатор турнира** **позива** **систем** да учита **тим**. (АПСО)

Опис акције: Координатор кликом на дугме „Детаљи“ позива системску операцију **UcitajTim(Tim)**

7. **Систем** **учитава** **тим**. (СО)
8. **Систем** приказује **координатору турнира** податке о **играчу** и поруку: “**Систем** је учитао **тим**”. (ИА)



9. **Координатор турнира** **уноси (мења)** податке о **тиму**. (АПУСО)

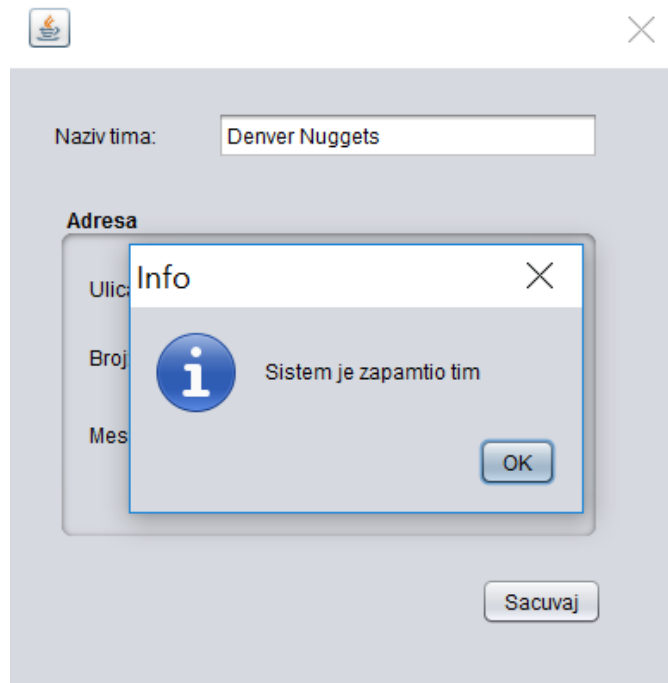


10. **Координатор турнира** **контролише** да ли је коректно унео податке у **тим**. (АНСО)
11. **Координатор турнира** **позива** **систем** да запамти податке о **тиму**. (АПСО)

Опис акције: Координатор кликом на дугме „Сачувај“ позива системску операцију **ZapamtiTim(Tim)**

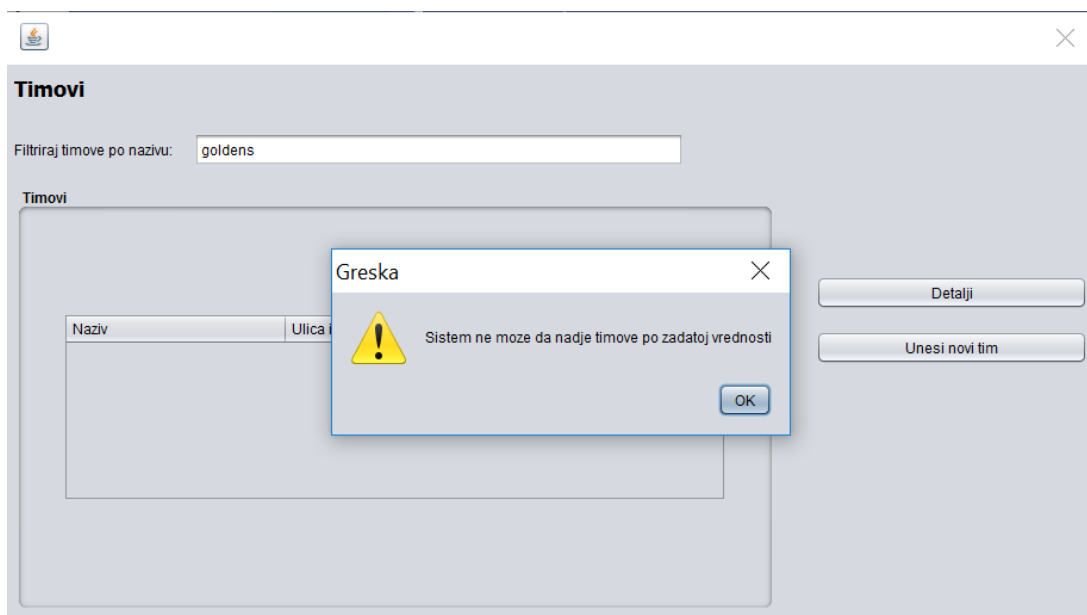
12. **Систем** **памти** податке о **тиму**. (СО)

13. Систем приказује координатору турнира запамћени тим и поруку: “Систем је запамтио тим”. (ИА)

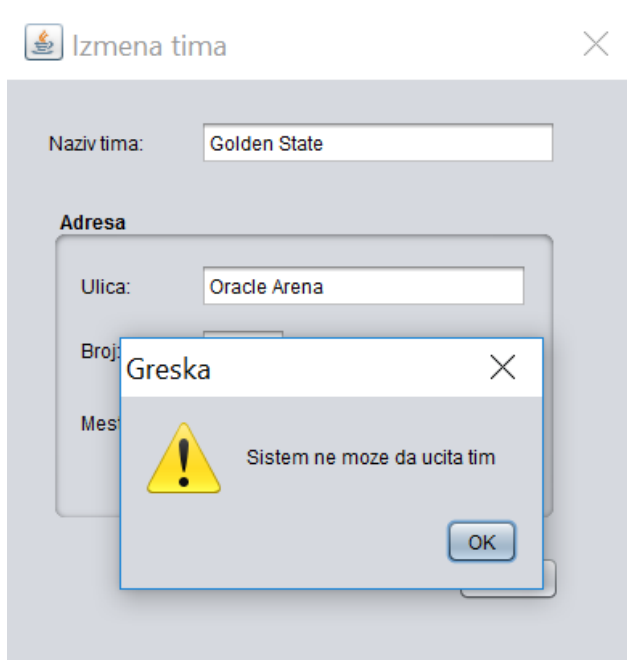


Алтернативна сценарија

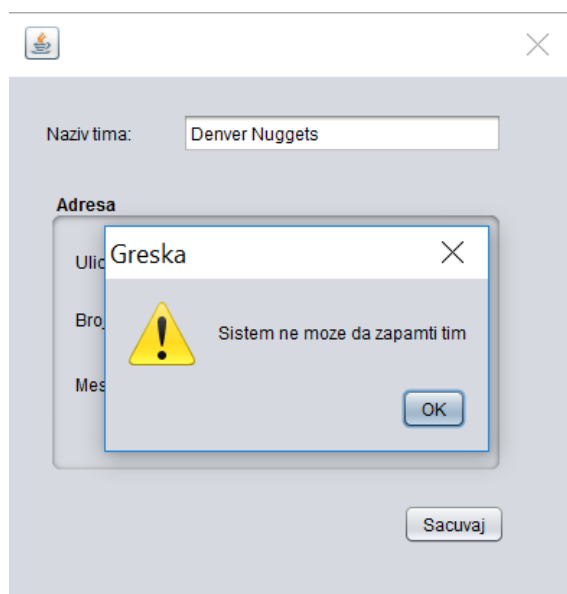
- 4.1 Уколико систем не може да нађе тимове он приказује координатору турнира поруку: “Систем не може да нађе тимове по задатој вредности”. Прекида се извршење сценарија. (ИА)



- 8.1. Уколико **систем** не може да учита **тим** он приказује **координатору турнира** поруку: “**Систем** не може да учита **тим**”. (ИА)



- 13.1 Уколико **систем** не може да запамти податке о **тиму** он приказује **координатору турнира** поруку “**Систем** не може да запамти **тим**”. (ИА)



СК7: Случај коришћења – Креирање утакмице

Назив СК

Креирање утакмице

Актори СК

Координатор турнира

Учесници СК

Координатор турнира и систем (програм)

Предуслов: Систем је укључен и координатор турнира је улогован под својом шифром. Систем приказује форму за рад са утакмицом. Учитана је листа тимова.

The screenshot shows a web application interface for creating a match. The form is titled "ID utakmice:" and includes a text input field. Below it is a field for "Datum odigravanja utakmice(format dd.mm.yyyy):". The "Domaci tim:" field is a dropdown menu with "Harlem Shake" selected. The "Gostujući tim:" field is also a dropdown menu. A button "Učitaj igrace timova" is located below the guest team dropdown. The "Statistika Igraca na utakmici" section contains a table with two columns: "Igrac" and "Broj poena igrača". To the right of the table is a "Dodaj statistiku igrača" section with a dropdown for "Igrac", a text input for "Broj poena:", and a "Dodaj statistiku" button. Below the table are two text input fields: "Poeni domacina:" and "Poeni gosta:". At the bottom of the form are two buttons: "Ponisti podatke utakmice" and "Sacuvaj".

| Igrac | Broj poena igrača |
|-------|-------------------|
|-------|-------------------|

Poeni domacina:

Poeni gosta:

Ponisti podatke utakmice Sacuvaj

Основни сценарио СК

1. **Координатор турнира** уноси податке у **утакмицу**. (АПУСО)

The screenshot shows a software window for managing basketball matches. It includes fields for match ID, date, home and away teams, and a table for player statistics. Buttons for saving, deleting, and adding statistics are also visible.

ID utakmice: 5

Datum odigravanja utakmice(format dd.mm.yyyy): 05.07.2017

Domaci tim: Poslasticari

Gostujuci tim: Harlem Shake

Učitaj igrače timova

Statistika igrača na utakmici

| Igrac | Broj poena igrača |
|----------------|-------------------|
| Darko Xiki | 24 |
| James Harden | 12 |
| Jamal Crawford | 11 |

Dodaj statistiku igrača

Igrac: Jamal Crawford

Broj poena: -1

Dodaj statistiku

Obrisi selektovanu statistiku

Poeni domaćina: 24

Poeni gosta: 23

Poniši podatke utakmice

Sačuvaj

2. **Координатор турнира** контролише да ли је коректно унео податке у **утакмицу**. (АНСО)
3. **Координатор турнира** позива **систем** да запамти податке о **утакмици**. (АПСО)

Опис акције: Координатор кликом на дугме „Сачувај“ позива системску операцију **ЗапамтиUtakmicu(Utakmica)**

4. **Систем** памти податке о **утакмици**. (СО)
5. **Систем** приказује **координатору турнира** запамћену **утакмицу** и поруку: “**Систем** је запамтио **утакмицу**”. (ИА)

ID utakmice: 5

Datum odigravanja utakmice(format dd.mm.yyyy): 05.07.2017

Domaci tim: Poslasticari

Gostujuci tim: Harlem Shake

Ucitaj igrace timova

Statistika Igraca na utakmici

| Igrac | Broj poena igraca |
|----------------|-------------------|
| Darko Xiki | |
| James Harden | |
| Jamal Crawford | |

Dodaj statistiku igraca

Igrac: Jamal Crawford

Broj poena: 1

Dodaj statistiku

Obrisi selektovanu statistiku

Poeni domacina: 24

Poeni gosta: 23

Ponisti podatke utakmice

Sacuvaj


Info

Sistem je zapamtio utakmicu

OK

Алтернативна сценарија

5.1 Уколико **систем** не може да запамти податке о **утакмици** он приказује **координатору турнира** поруку “**Систем** не може да запамти **утакмицу**”. (ИА)



ID utakmice:

5

Datum odigravanja utakmice(format dd.mm.yyyy):

05.07.2017

Domaci tim:

Poslasticari

Gostujuci tim:

Harlem Shake

Ucitaj igrace timova

Statistika Igraca na utakmici

| Igrac | Broj poena igraca |
|----------------|-------------------|
| Darko Xiki | 24 |
| James Harden | 12 |
| Jamal Crawford | 11 |

Dodaj statistiku igraca

Igrac:

Jamal Crawford

Broj poena:

1

Dodaj statistiku

Obrisi selektovanu statistiku

Poeni domacina:

24


Poeni gosta:

23

Ponisti podatke utakmice

Sacuvaj

Greska

 Sistem ne moze da zapamti utakmicu

OK

95

СК8: Случај коришћења – Претраживање утакмице

Назив СК

Претраживање **утакмице**

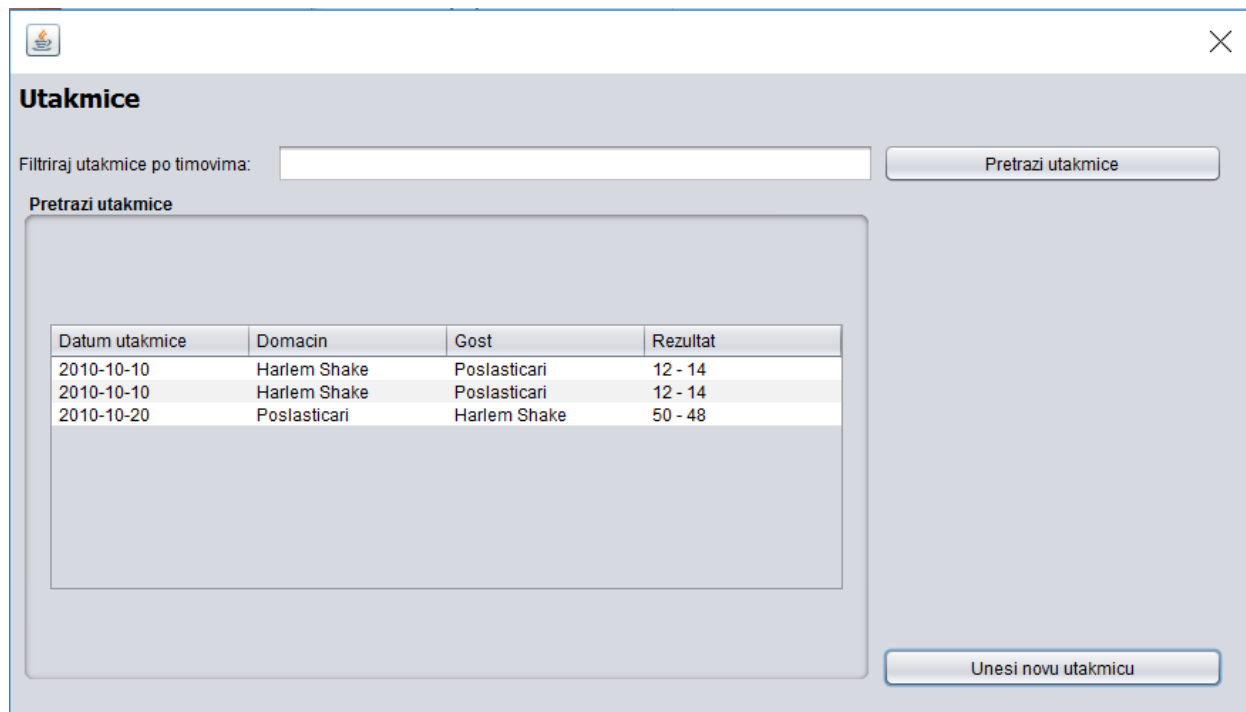
Актори СК

Координатор турнира

Учесници СК

Координатор турнира и **систем** (програм)

Предуслов: Систем је укључен и **координатор турнира** је улогован под својом шифром. Систем приказује форму за рад са **утакмицом**.



| Datum utakmice | Domacin | Gost | Rezultat |
|----------------|--------------|--------------|----------|
| 2010-10-10 | Harlem Shake | Poslasticari | 12 - 14 |
| 2010-10-10 | Harlem Shake | Poslasticari | 12 - 14 |
| 2010-10-20 | Poslasticari | Harlem Shake | 50 - 48 |

Основни сценарио СК

1. **Координатор турнира** уноси вредност по којој претражује **утакмице**. (АПУСО)

The screenshot shows a window titled "Utakmice" with a search filter "Filtriraj utakmice po timovima:" containing the text "poslasticari". A "Pretrazi utakmice" button is to the right. Below the filter is a table with the following data:

| Datum utakmice | Domacin | Gost | Rezultat |
|----------------|--------------|----------------|----------|
| 2010-10-20 | Poslasticari | Harlem Shake | 50 - 48 |
| 2010-10-10 | Harlem Shake | Poslasticari | 12 - 14 |
| 2010-10-10 | Harlem Shake | Poslasticari | 12 - 14 |
| 2017-07-05 | Poslasticari | Denver Nuggets | 50 - 0 |

At the bottom right of the window is a button labeled "Unesi novu utakmicu".

2. **Координатор турнира** позива **систем** да нађе **утакмице** по задатој вредности. (АПСО)

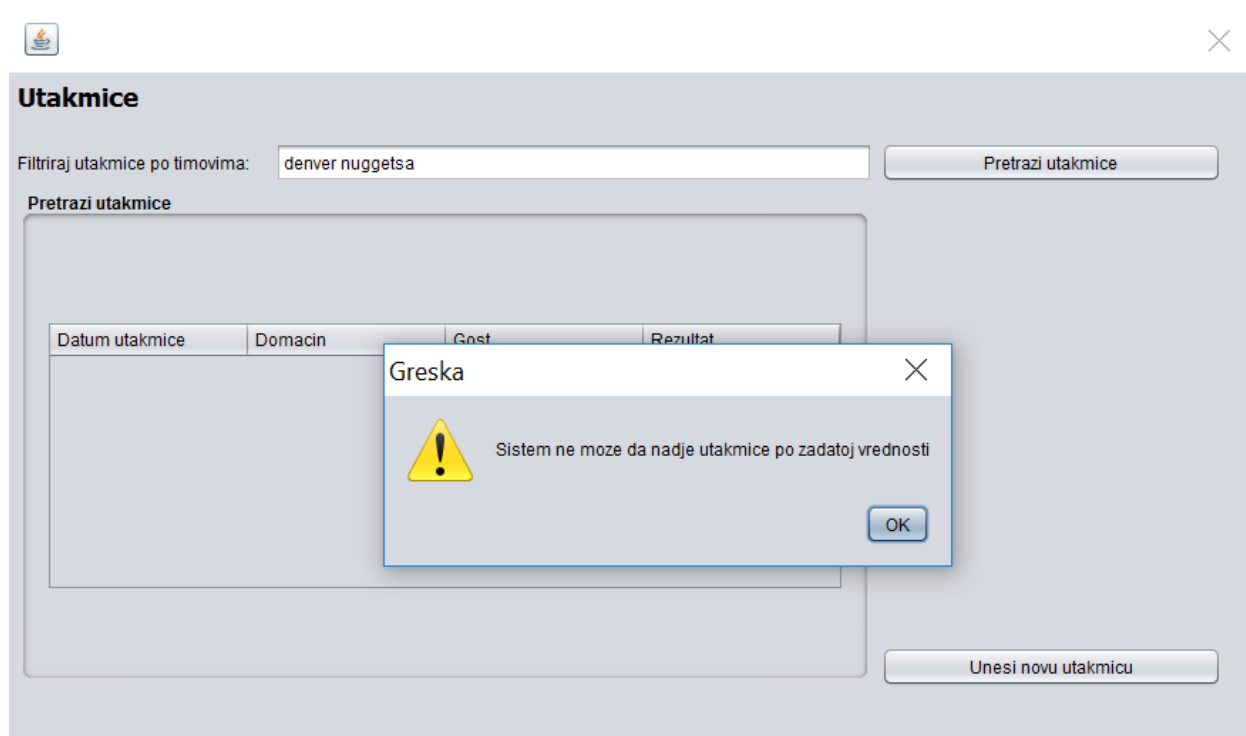
Опис акције: Координатор притиском типке тастатуре позива системску операцију **NadjiUtakmice(Utakmica,List<Utakmica>)**

3. **Систем** **тражи** **утакмице** по задатој вредности. (СО)
4. **Систем** приказује **координатору турнира** податке о **утакмицама** и поруку: “**Систем** је нашао **утакмице** по задатој вредности”. (ИА)

The screenshot shows the same "Utakmice" window as before, but with an "Info" dialog box overlaid in the center. The dialog box contains an information icon and the text "Sistem je nasao utakmice po zadatoj vrednosti". An "OK" button is at the bottom right of the dialog box. The table in the background is partially visible and contains the same data as in the previous screenshot.

Алтернативна сценарија

- 4.1 Уколико **систем** не може да нађе **утакмице** он приказује **координатору турнира** поруку:
“**Систем** не може да нађе **утакмице** по задатој вредности”. (ИА)



4.3.1.2. Пројектовање контролера корисничког интерфејса

Контролер корисничког интерфејса је одговоран за:

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

4.3.2. Пројектовање апликационе логике

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

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

Комуникација са клијентима

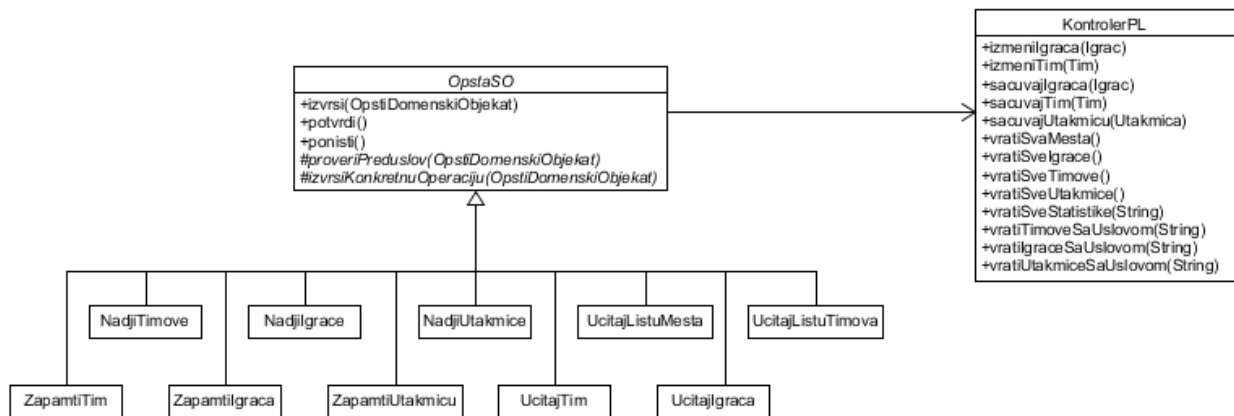
Део за комуникацију подиже серверски сокет који ослушкује мрежу. Када клијентски сокет успостави конекцију са серверским сокетом, тада сервер генерише нит која ће успоставити двосмерну комуникацију са клијентом.

Слање и примање података од клијента се обавља разменом објеката класе `KlijentskiZahtev` и `ServerskiOdgovor` и остварује се преко сокета.

Клијент шаље захтев за извршење неке од системских операција до одговарајуће нити која је повезана са тим клијентом. Та нит прихвата захтев и прослеђује га до контролера апликационе логике. Након извршења системске операције, резултат се преко контролера апликационе логике враћа до нити клијента која тај резултат шаље назад до клијента.

4.3.2.1. Контролер апликационе логике

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



4.3.2.2. Пословна логика

Пројектовање понашања софтверског система – системске операције

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

За сваки уговор пројектује се концептуално решење.

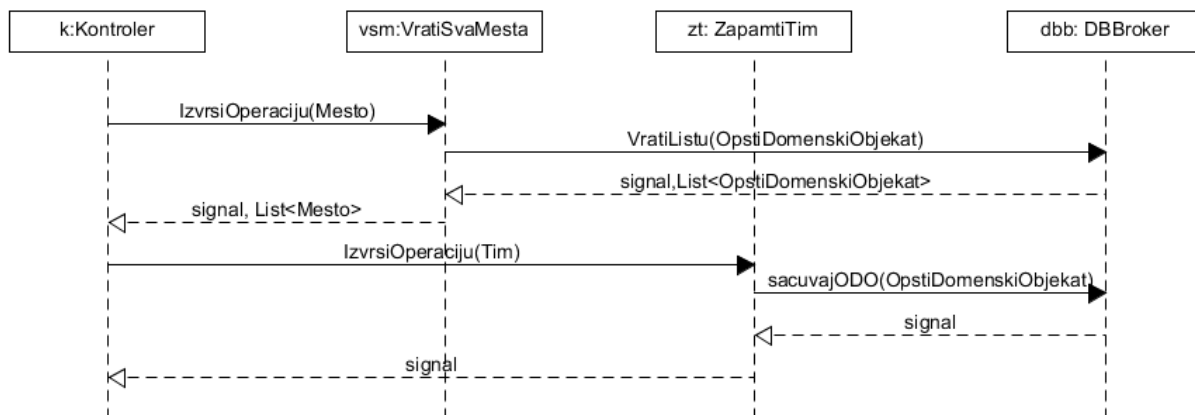
Уговор УГ1: ZapamtiTim

Операција: ZapamtiTim(Tim): signal;

Веза са СК: СК1, СК6

Предуслови: Вредносна и структурна ограничења над објектом **Tim** морају бити задовољена. Учитана листа места из базе података у комбо боксу.

Постуслови: Подаци о тиму су запамћени.



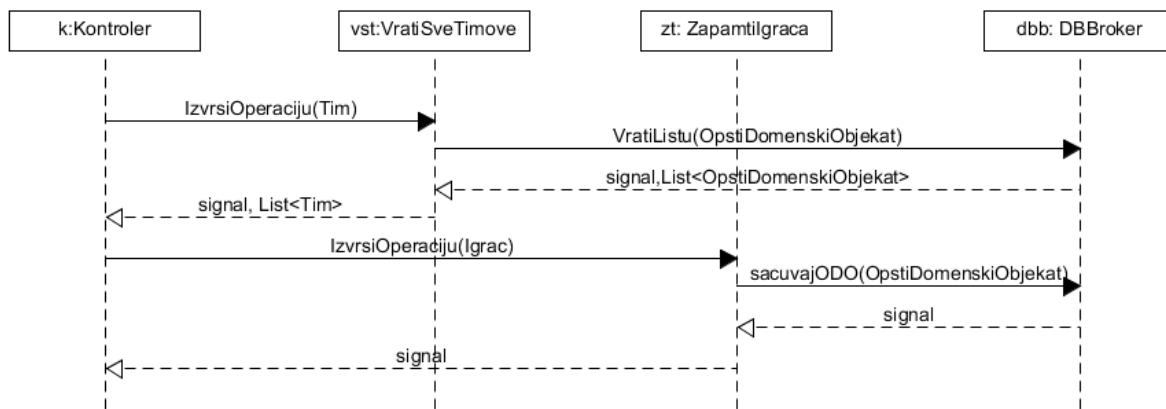
Уговор УГ2: Zapamtilgraca

Операција: Zapamtilgraca(Igrac): signal;

Веза са СК: СК2, СК5

Предуслови: Вредносна и структурна ограничења над објектом **Igrac** морају бити задовољена. Учитана листа тимова из базе података у комбо боксу.

Постуслови: Подаци о играчу су запамћени.



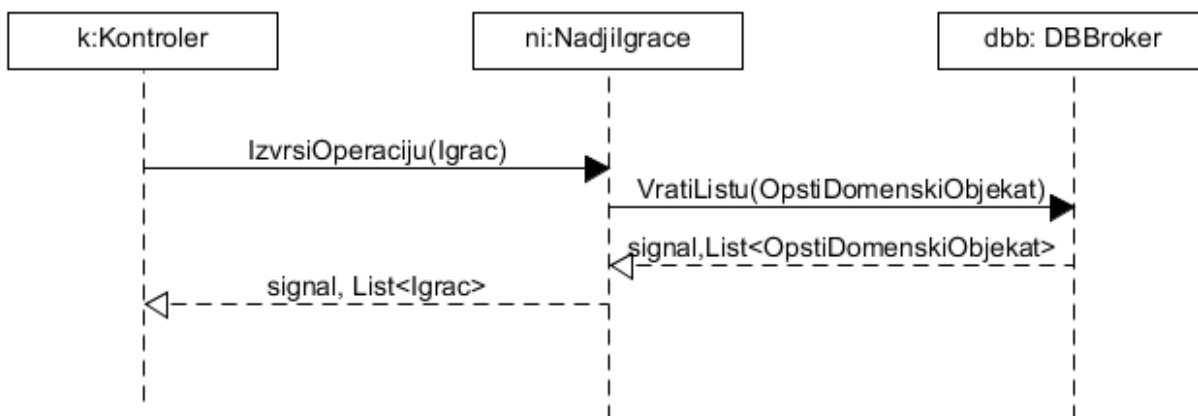
Уговор УГ3: Nadjilgrace

Операција: Nadjilgrace(Igrac, List<Igrac>): signal;

Веза са СК: СК3, СК5

Предуслови:

Постуслови:



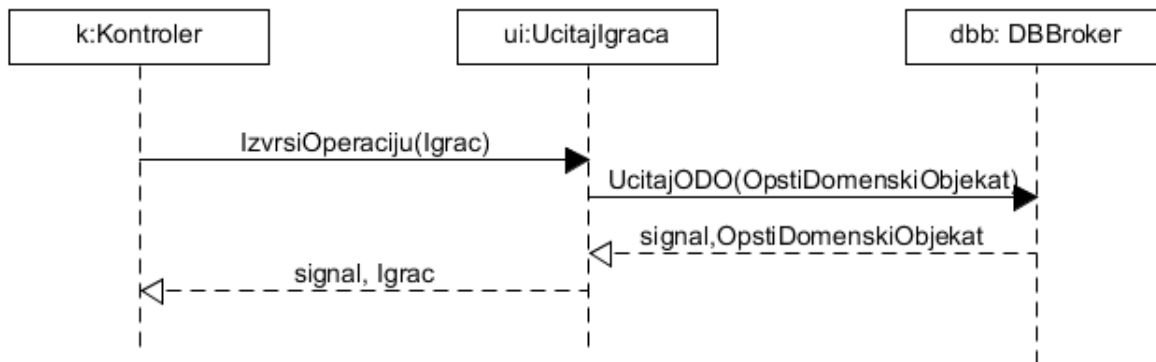
Уговор УГ4: UcitajIgraca

Операција: UcitajIgraca(Igrac):signal;

Веза са СК: СК3, СК5

Предуслови:

Постуслови:



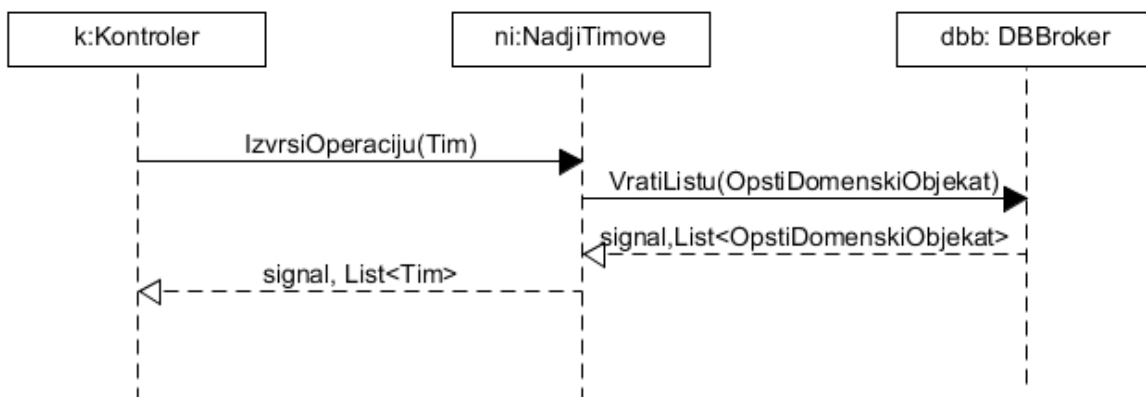
Уговор УГ5: NadjiTimove

Уговор УГ5: NadjiTimove(Tim, List<Tim>) Signal;

Веза са СК: СК4, СК6

Предуслови:

Постуслови:



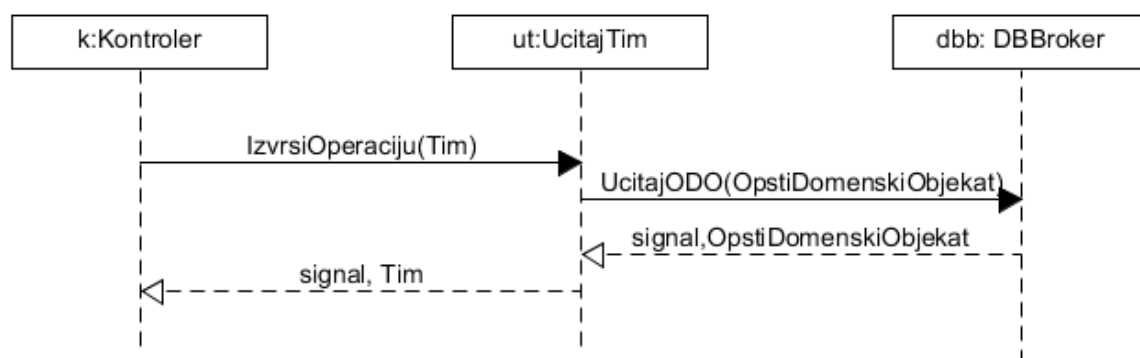
Уговор УГ6: UcitajTim

Операција:UcitajTim(Tim): signal;

Веза са СК: СК4, СК6

Предуслови:

Постуслови:



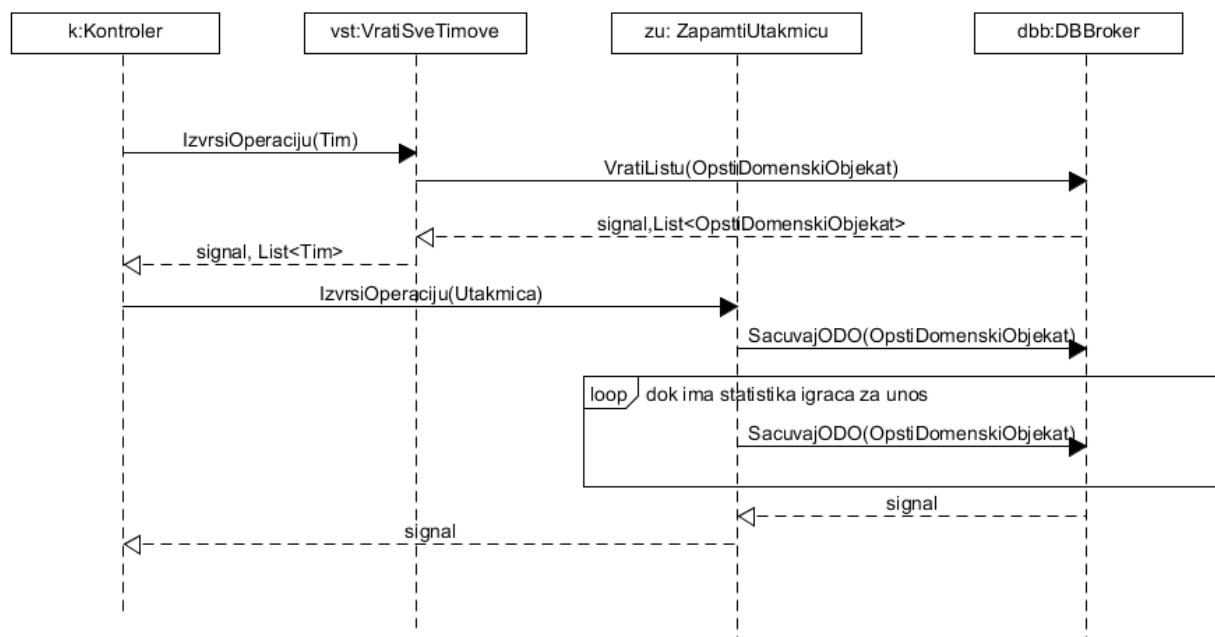
Уговор УГ7: ZapamtiUtakmicu

Операција: ZapamtiUtakmicu(Utakmica):signal;

Веза са СК: СК7

Предуслови: Вредносна и структурна ограничења над објектом **Utakmica** морају бити задовољена. Учитана листа тимова из базе података у комбо боксу.

Постуслови: Подаци о утакмици су запамћени.



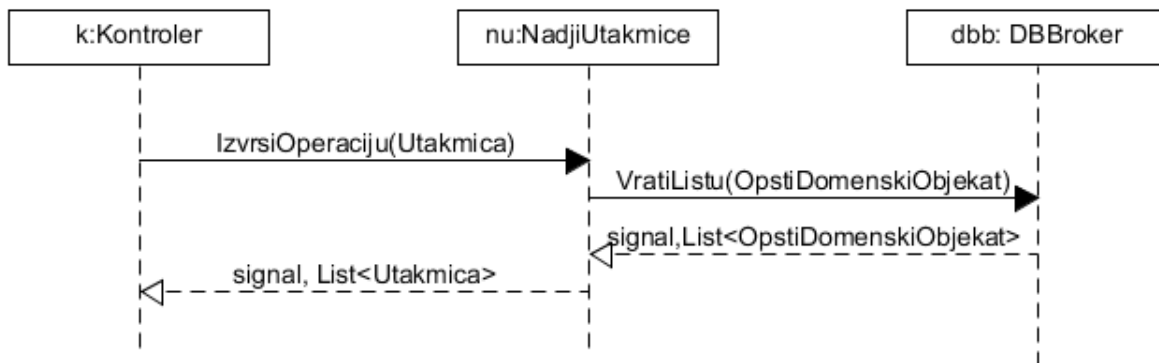
Уговор УГ8: NadjiUtakmice

Операција: NadjiUtakmice(Utakmica, List<Utakmica>) :signal;

Веза са СК: CK8

Предуслови:

Постуслови:



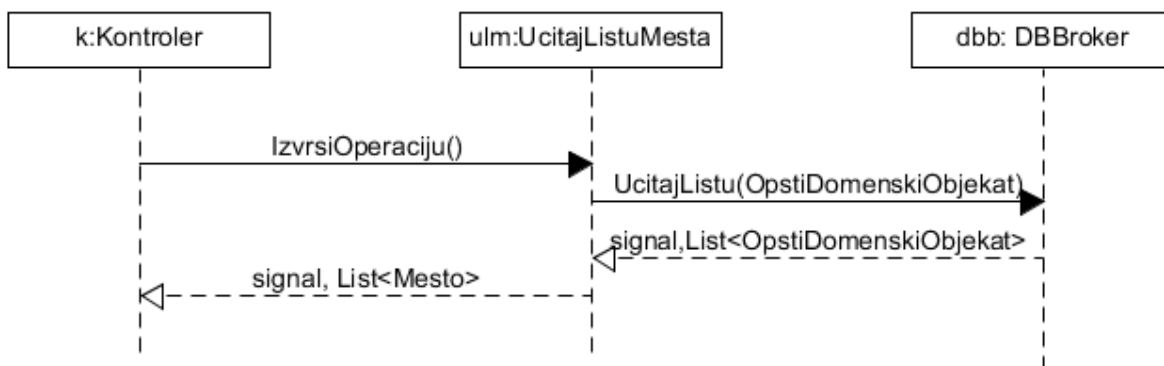
Уговор УГ9: UcitajListuMesta

Операција: UcitajListuMesta(List<Mesto>) :signal;

Веза са СК: CK1, CK6

Предуслови:

Постуслови:



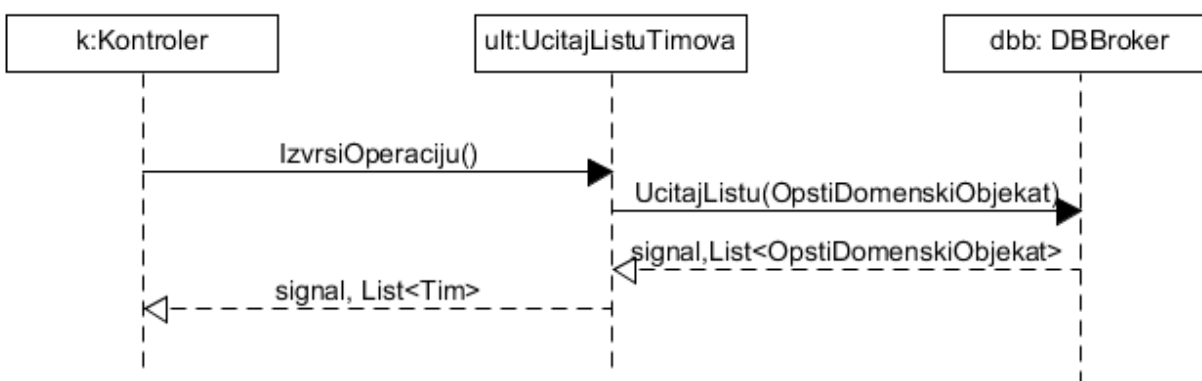
Уговор УГ10: UcitajListuTimova

Операција: UcitajListuTimova(List<Tim>) :signal;

Веза са СК: СК2, СК5, СК7

Предуслови:

Постуслови:

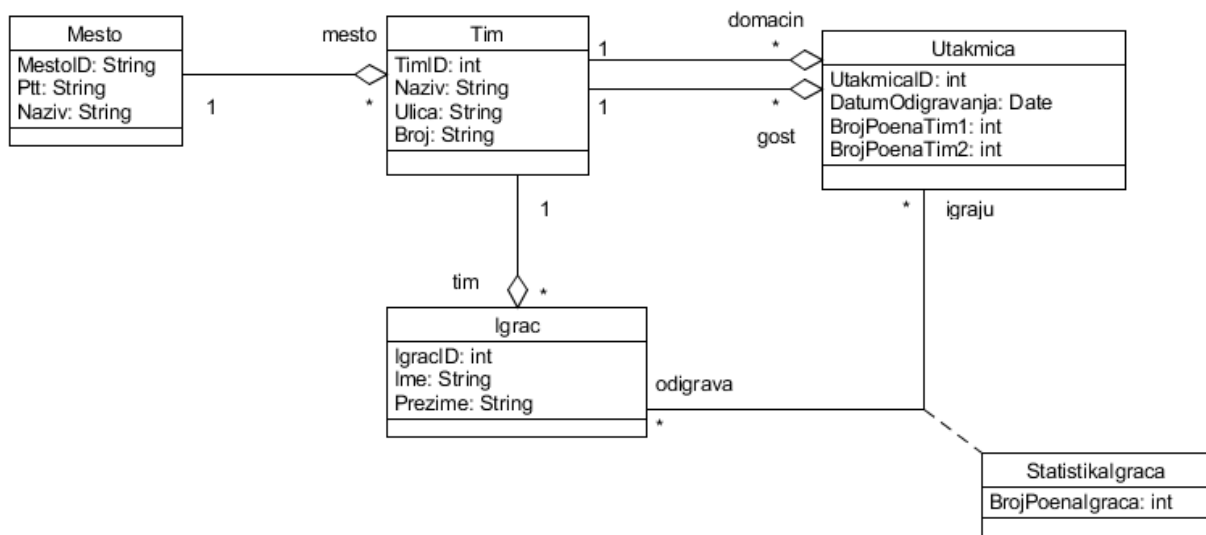


Класе које су одговорне за извршење системских операција наслеђују класу OpstaSO како би могле да се повежу са базом. OpstaSO представља апстрактну класу чија главна метода (izvrsi()) у себи садржи отварање конекције са базом, валидацију, проверу предуслова, извршење операције, потврду у бази уколико је извршење операције успешно, поништавање уколико извршење операције није било успешно и затварање конекције. Свака од системских операција даје своју имплементацију методе за проверу предуслова, уколико постоји, и методе за извршење конкретне системске операције.

Пројектовање структуре софтверског система

На основу концептуалних класа праве се софтверске класе структуре.

Концептуалне класе:



Софтверске класе структуре:

```
public class Mesto implements Serializable, OpstiDomenskiObjekat {

    private int mestoID;
    private String ptt;
    private String naziv;

    public Mesto() {
    }

    public Mesto(int mestoID, String ptt, String naziv) {
        this.mestoID = mestoID;
        this.ptt = ptt;
        this.naziv = naziv;
    }
}
```

```

public class Tim implements Serializable, OpstiDomenskiObjekat {

    private int timID;
    private String naziv;
    private String ulica;
    private String broj;
    private Mesto mesto;

    public Tim() {
    }

    public Tim(int timID, String naziv, String ulica, String broj, Mesto mesto){
        this.timID = timID;
        this.naziv = naziv;
        this.ulica = ulica;
        this.broj = broj;
        this.mesto = mesto;
    }
}

public class Utakmica implements Serializable, OpstiDomenskiObjekat {

    private int utakmicaID;
    private Date datumOdigravanja;
    private int poeniPrvi;
    private int poeniDrugi;
    private Tim tim1;
    private Tim tim2;
    private List<StatistikaIgraca> statistikaIgraca;

    public Utakmica() {
        statistikaIgraca = new ArrayList<>();
    }

    public Utakmica(int utakmicaID, Date datumOdigravanja, int poeniPrvi, int poeniDrugi, Tim tim1, Tim tim2) {
        this.utakmicaID = utakmicaID;
        this.datumOdigravanja = datumOdigravanja;
        this.poeniPrvi = poeniPrvi;
        this.poeniDrugi = poeniDrugi;
        this.tim1 = tim1;
        this.tim2 = tim2;
        statistikaIgraca = new ArrayList<>();
    }
}

public class Igrac implements Serializable, OpstiDomenskiObjekat {

    private int igracID;
    private String ime;
    private String prezime;
    private Tim tim;

    public Igrac() {
    }

    public Igrac(int igracID, String ime, String prezime, Tim tim) {
        this.igracID = igracID;
        this.ime = ime;
        this.prezime = prezime;
        this.tim = tim;
    }
}

```

```
public class StatistikaIgraca implements Serializable, OpstiDomenskiObjekat {  
  
    private Igrac igrac;  
    private Utakmica utakmica;  
    private int brojPoenaIgraca;  
  
    public StatistikaIgraca() {  
    }  
  
    public StatistikaIgraca(Igrac igrac, Utakmica utakmica, int brojPoenaIgraca) {  
        this.igrac = igrac;  
        this.utakmica = utakmica;  
        this.brojPoenaIgraca = brojPoenaIgraca;  
    }  
}
```

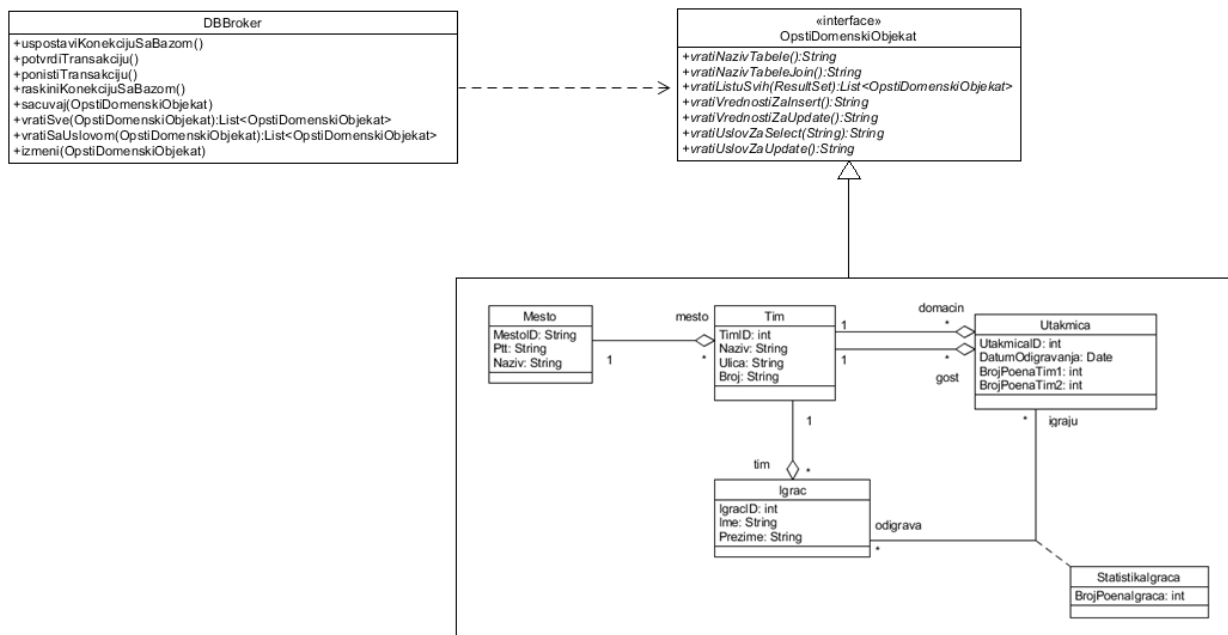
4.3.2.3. Брокер базе података

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

- public void uspostaviKonekcijuSaBazom();
- public void potvrdiTransakciju();
- public void ponistiTransakciju();
- public void raskiniKonekcijuSaBazom();
- public void sacuvaj (OpstiDomenskiObjekat odo);
- public List<OpstiDomenskiObjekat> vratiSve(OpstiDomenskiObjekat odo);
- public List<OpstiDomenskiObjekat> vratiSaUslovom(OpstiDomenskiObjekat odo);
- public void izmeni (OpstiDomenskiObjekat odo);

Све методе класе DBBroker су пројектоване као генеричке, што значи да могу да прихвате различите доменске објекте преко параметара. Ово је остварено дефинисањем интерфејса OpstiDomenskiObjekat кога имплементирају све доменске класе.

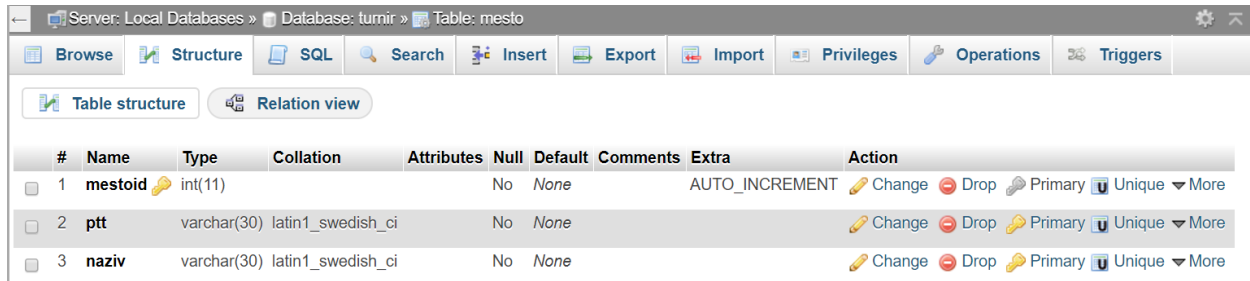
- public abstract String vratiNazivTabele();
- public abstract String vratiNazivTabeleJoin();
- public abstract List<OpstiDomenskiObjekat> vratiListuSvih(ResultSet rs);
- public abstract String vratiVrednostiZaInsert();
- public abstract String vratiVrednostiZaUpdate();
- public abstract String vratiUslovZaSelect(String kriterijumZaPretragu);
- public abstract String vratiUslovZaUpdate();



4.3.3. Пројектовање складишта података

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

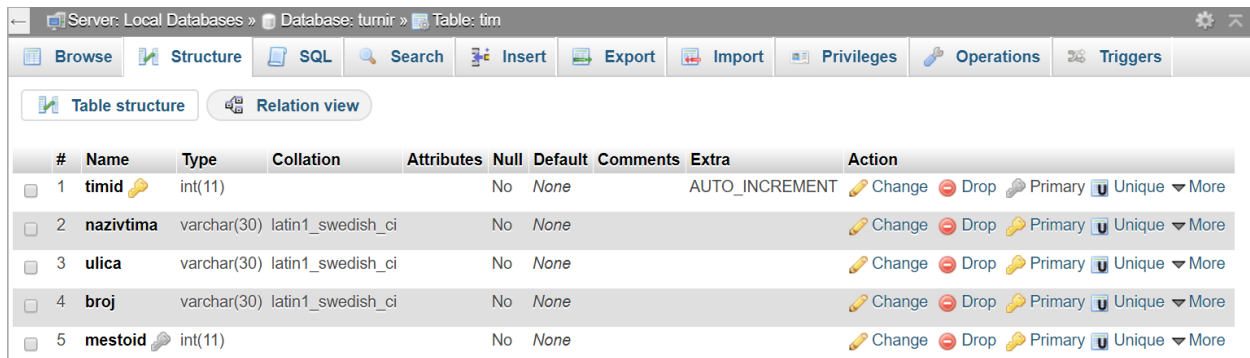
Табела Mesto



The screenshot shows the MySQL Table Structure window for the 'mesto' table in the 'tumir' database. The table has three columns: 'mestoid' (int(11), AUTO_INCREMENT, Primary, Unique), 'ptt' (varchar(30), latin1_swedish_ci, Primary, Unique), and 'naziv' (varchar(30), latin1_swedish_ci, Primary, Unique). Each column has a 'Change' icon, a 'Drop' icon, and a 'More' dropdown menu.

| # | Name | Type | Collation | Attributes | Null | Default | Comments | Extra | Action |
|---|---------|-------------|-------------------|------------|------|---------|----------|----------------|---------------------------------|
| 1 | mestoid | int(11) | | | No | None | | AUTO_INCREMENT | Change Drop Primary Unique More |
| 2 | ptt | varchar(30) | latin1_swedish_ci | | No | None | | | Change Drop Primary Unique More |
| 3 | naziv | varchar(30) | latin1_swedish_ci | | No | None | | | Change Drop Primary Unique More |

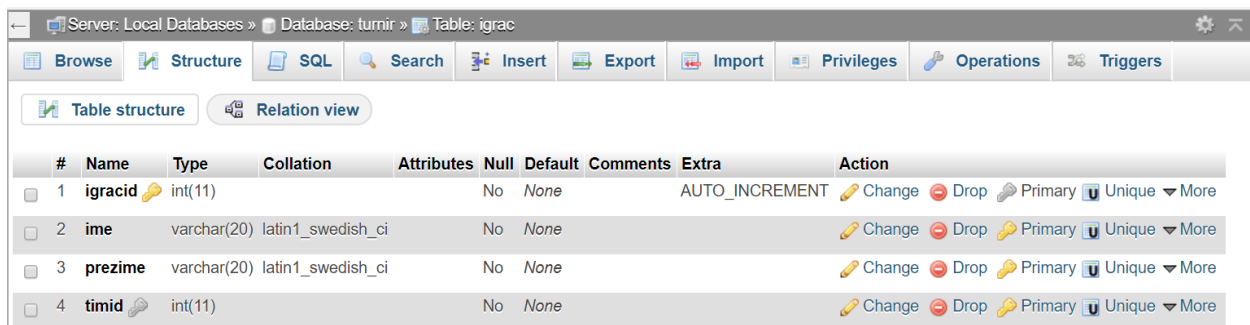
Табела Tim



The screenshot shows the MySQL Table Structure window for the 'tim' table in the 'tumir' database. The table has five columns: 'timid' (int(11), AUTO_INCREMENT, Primary, Unique), 'nazivtima' (varchar(30), latin1_swedish_ci, Primary, Unique), 'ulica' (varchar(30), latin1_swedish_ci, Primary, Unique), 'broj' (varchar(30), latin1_swedish_ci, Primary, Unique), and 'mestoid' (int(11), Primary, Unique). Each column has a 'Change' icon, a 'Drop' icon, and a 'More' dropdown menu.

| # | Name | Type | Collation | Attributes | Null | Default | Comments | Extra | Action |
|---|-----------|-------------|-------------------|------------|------|---------|----------|----------------|---------------------------------|
| 1 | timid | int(11) | | | No | None | | AUTO_INCREMENT | Change Drop Primary Unique More |
| 2 | nazivtima | varchar(30) | latin1_swedish_ci | | No | None | | | Change Drop Primary Unique More |
| 3 | ulica | varchar(30) | latin1_swedish_ci | | No | None | | | Change Drop Primary Unique More |
| 4 | broj | varchar(30) | latin1_swedish_ci | | No | None | | | Change Drop Primary Unique More |
| 5 | mestoid | int(11) | | | No | None | | | Change Drop Primary Unique More |

Табела Igrac



The screenshot shows the MySQL Table Structure window for the 'igrac' table in the 'tumir' database. The table has four columns: 'igracid' (int(11), AUTO_INCREMENT, Primary, Unique), 'ime' (varchar(20), latin1_swedish_ci, Primary, Unique), 'prezime' (varchar(20), latin1_swedish_ci, Primary, Unique), and 'timid' (int(11), Primary, Unique). Each column has a 'Change' icon, a 'Drop' icon, and a 'More' dropdown menu.

| # | Name | Type | Collation | Attributes | Null | Default | Comments | Extra | Action |
|---|---------|-------------|-------------------|------------|------|---------|----------|----------------|---------------------------------|
| 1 | igracid | int(11) | | | No | None | | AUTO_INCREMENT | Change Drop Primary Unique More |
| 2 | ime | varchar(20) | latin1_swedish_ci | | No | None | | | Change Drop Primary Unique More |
| 3 | prezime | varchar(20) | latin1_swedish_ci | | No | None | | | Change Drop Primary Unique More |
| 4 | timid | int(11) | | | No | None | | | Change Drop Primary Unique More |

Табела Utakmica

Server: Local Databases » Database: tumir » Table: utakmica

Table structure Relation view

| # | Name | Type | Collation | Attributes | Null | Default | Comments | Extra | Action |
|---|------------------|---------|-----------|------------|------|---------|----------|-------|---|
| 1 | utakmicaid | int(11) | | | No | None | | | Change Drop Primary Unique Index Spatial More |
| 2 | datumodigravanja | date | | | No | None | | | Change Drop Primary Unique Index Spatial More |
| 3 | brojpoenatim1 | int(10) | | | No | None | | | Change Drop Primary Unique Index Spatial More |
| 4 | brojpoenatim2 | int(10) | | | No | None | | | Change Drop Primary Unique Index Spatial More |
| 5 | timid1 | int(11) | | | No | None | | | Change Drop Primary Unique Index Spatial More |
| 6 | timid2 | int(11) | | | No | None | | | Change Drop Primary Unique Index Spatial More |

Табела Statistikaigraca

Server: Local Databases » Database: tumir » Table: statistikaigraca

Table structure Relation view

| # | Name | Type | Collation | Attributes | Null | Default | Comments | Extra | Action |
|---|-----------------|---------|-----------|------------|------|---------|----------|-------|---|
| 1 | igracid | int(11) | | | No | None | | | Change Drop Primary Unique Index Spatial More |
| 2 | utakmicaid | int(11) | | | No | None | | | Change Drop Primary Unique Index Spatial More |
| 3 | brojpoenaigraca | int(11) | | | No | None | | | Change Drop Primary Unique Index Spatial More |

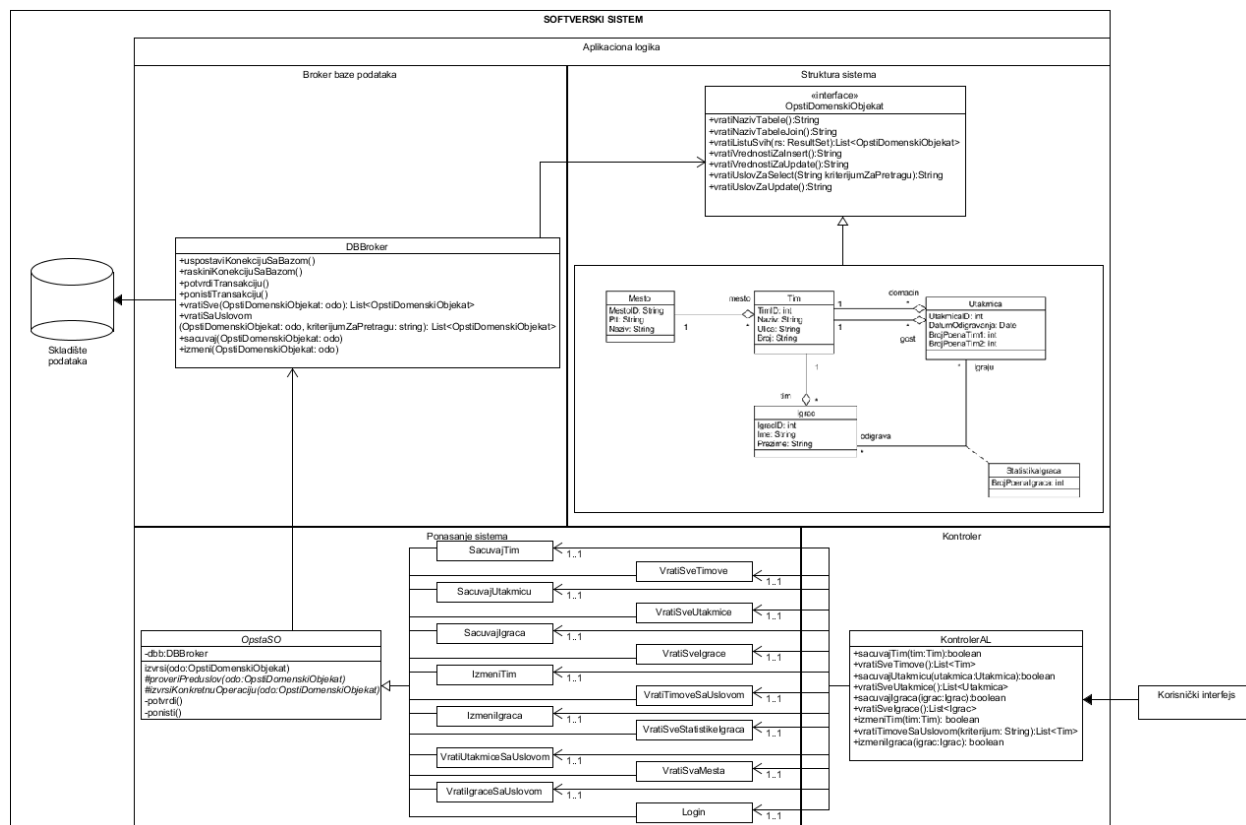
Додата је и табела Administrator са креденцијалима за почетно логовање на систем.

Server: Local Databases » Database: tumir » Table: administratori

Table structure Relation view

| # | Name | Type | Collation | Attributes | Null | Default | Comments | Extra | Action |
|---|----------|-------------|-------------------|------------|------|---------|----------|----------------|---------------------------------|
| 1 | adminID | int(11) | | | No | None | | AUTO_INCREMENT | Change Drop Primary Unique More |
| 2 | ime | varchar(20) | latin1_swedish_ci | | No | None | | | Change Drop Primary Unique More |
| 3 | prezime | varchar(20) | latin1_swedish_ci | | No | None | | | Change Drop Primary Unique More |
| 4 | username | varchar(20) | latin1_swedish_ci | | No | None | | | Change Drop Primary Unique More |
| 5 | pass | varchar(20) | latin1_swedish_ci | | No | None | | | Change Drop Primary Unique More |

На основу претходних целина, може се саставити цела архитектура софтверског система за праћење рада кошаркашког турнира.



4.4 Имплементација

Софтверски систем је развијан у програмском језику “Java”. Систем је пројектован као клијент-сервер апликација. Као систем за управљање базом података коришћен је MySQL, док је развојно окружење “NetBeans IDE 7.4”. На основу архитектуре софтверског система добијене су следеће софтверске класе:

- ProjekatKlijent
 - forme/DialogGlavni
 - forme/DialogIgrac
 - forme/DialogIgracDetails
 - forme/DialogTim
 - forme/DialogTimDetails
 - forme/DialogUtkmica
 - forme/DialogUtkmicaDetails
 - forme/Login
 - komunikacija/KomunikacijaSaServerom
 - kontroler/KontrolerIgrac
 - kontroler/KontrolerLogin
 - kontroler/KontrolerTim
 - kontroler/KontrolerUtkmica
 - tablemodel/TableModelIgrac
 - tablemodel/TableModelStatistikalgraca
 - tablemodel/TableModelTim
 - tablemodel/TableModelUtkmica
- ProjekatServer
 - baza/DBBroker
 - forme/ServerBaza
 - forme/ServerConfig
 - forme/ServerForma
 - logika/Kontroler
 - niti/NitObradaZahtevaKlijenta
 - niti/NitPokretanjeServera
 - so/OpstaSO
 - so/SOLzmenilgraca
 - so/SOLzmeniTim
 - so/SOLogin
 - so/SOSacuvajIgraca
 - so/SOSacuvajTim

- so/SOSacuvajUtakmicu
- so/SOVratilgraceSaUslovom
- so/SOVratiSvaMesta
- so/SOVratiSvelgrace
- so/SOVratiSveStatistikelgraca
- so/SOVratiSveTimove
- so/SOVratiSveUtakmice
- so/SOVratiTimoveSaUslovom
- so/SOVratiUtakmiceSaUslovom
- tablemodel/TableModelAdministrator
- ProjekatZajednicki
 - config/Konfiguracija
 - config/Operacije
 - config/StatusOdgovora
 - config/Utility
 - domen/Administrator
 - domen/Igrac
 - domen/Mesto
 - domen/OpstiDomenskiObjekat
 - domen/Statistikalgraca
 - domen/Tim
 - domen/Utakmica
 - transfer/KlijentskiZahtev
 - transfer/ServerskiOdgovor

4.5 Тестирање

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

5. Закључак

Јава се данас распознаје као један од популарнијих програмских језика са огромним бројем корисника. Пошто је Јава платформски независна, јасно је зашто велики број пројектаната програмира баш у овом програмском језику. Према Википедији, процењује се да постоји око 10 милиона Јава програмера на свету, а ТИОВЕ заједница програмера рангира Јаву као тренутно најпопуларнији програмски језик, а у протеклих 15 година један од три најпопуларнија програмска језика за развој софтвера поред C и C++ програмских језика. Пошто се на Јава API-ју заснивају и новије технологије, као што је Андроид оперативни систем, може се створити слика о распрострањеној употреби Јава програмског језика.

Израда једне десктоп апликације у Јава окружењу је за мене представљало један занимљив и изазован задатак. Оно што ме је навело на одабир ове теме за завршни рад јесте то што је било неопходно користити многобројне концепте које сам изучавао током целокупних студија, а који сви заједно чине целину неопходну за развој једног софтверског система. На овом четворогодишњем путу, током прве године фокус је био на уводу у свет информационих технологија, као и упознавањем са информационим системима. На другој години, први пут сам се сусрео са Јава програмским језиком слушајући основне принципе програмирања и почео да пишем своје простије Јава програме, што је на мене оставило веома јак утисак и навело ме да константно развијам своје вештине програмирања. У трећој години, продубио сам своје знање учећи о мрежном програмирању, раду са базом података и генерално о основама програмских језика. Кулминацију мог студирања сам доживео на четвртој години, где сам за свој изборни предмет одабрао Софтверске патерне, где сматрам и да сам можда највише напредовао као један пројектант. Софтверски патерни су на мене оставили веома јак утисак – сво моје програмирање до тог тренутка је деловало небитно у односу на програме који су се барем мало придржавали патерна пројектовања! Након седам семестара, последњи семестар и Пројектовање софтвера су комплетирали све ове претходне концепте, тако да је развој једног софтверског система за мене савршен одабир завршног рада.

6. Литература

1. Проф.др. Сениша Влајић, *Пројектовање софтвера(скрипта)*, Београд 2015.
2. Проф.др. Сениша Влајић, *Софтверски патерни*, Златни пресек, Београд 2015.
3. Проф.др. Сениша Влајић, др. Душан Савић, др. Илија Антоновић, мр. Војислав Станојевић, дипл.инг. Милош Милић, *Пројектовање софтвера – Напредне Јава технологије*, Златни пресек, Београд 2008.
4. Бојан Томић, Јелена Јовановић, Никола Миликић, Зоран Шеварац, Драган Ђурић, *Принципи програмирања*, Београд 2013.
5. Java programming, Wikipedia, [https://sh.wikipedia.org/wiki/Java \(programski jezik\)](https://sh.wikipedia.org/wiki/Java_(programski_jezik))
6. Stack Overflow форум, <https://stackoverflow.com/>
7. Oracle документација о Јава програмском језику, <http://docs.oracle.com/>
8. Java Course, <http://www.vias.org/javacourse/>