Универзитет у Београду

Факултет организационих наука

Катедра за софтверско инжењерство

Завршни рад

**Тема: Развој софтверског система за кастинг у**

**филмској индустрији у Јава окружењу**

Ментор: Студент:

Проф. др Душан Савић Ђорђе Ражнатовић 1070/18

Београд, 2020

**Садржај**

[Увод 4](#_Toc76515307)

[Упрошћена Ларманова метода развоја софтвера 5](#_Toc76515308)

[Прикупљање корисничких захтева 5](#_Toc76515309)

[Пројектовање 6](#_Toc76515310)

[Јава, Јава технологије 7](#_Toc76515311)

[Концепти објектно-оријентисаног програмирања и Јава програмског језика 8](#_Toc76515312)

[Графички интерфејс у Јави 12](#_Toc76515313)

[Рад у мрежи 16](#_Toc76515314)

[Нити 19](#_Toc76515315)

[Рад са базом података 21](#_Toc76515316)

[Прикупљање корисничких захтева 23](#_Toc76515317)

[1.1 Вербални опис 23](#_Toc76515318)

[1.2 Случајеви коришћења: 23](#_Toc76515319)

[1.2.1 СК1: Случај коришћења – Креирање профила 25](#_Toc76515320)

[1.2.2 СК2: Случај коришћења – Промена профила 26](#_Toc76515321)

[1.2.3 СК3: Случај коришћења – Брисање профила 27](#_Toc76515322)

[1.2.4 СК4: Случај коришћења – Креирање корисничког налога 28](#_Toc76515323)

[1.2.5 СК5: Случај коришћења – Измена корисничког налога 29](#_Toc76515324)

[1.2.6 СК6: Случај коришћења – Брисање корисничког налога 30](#_Toc76515325)

[1.2.7 СК7: Случај коришћења – Креирање пројекта 31](#_Toc76515326)

[1.2.8 СК8: Случај коришћења – Брисање пројекта 32](#_Toc76515327)

[1.2.9 СК9: Случај коришћења – Резервисање глумца 33](#_Toc76515328)

[Анализа 34](#_Toc76515329)

[1.3 Системски дијаграм секвенци 34](#_Toc76515330)

[1.3.1 ДС1: Дијаграм секвенце случаја коришћења – Креирање профила 34](#_Toc76515331)

[1.3.2 ДС2: Дијаграм секвенце случаја коришћења – Промена профила 36](#_Toc76515332)

[1.3.3 ДС3: Дијаграм секвенце случаја коришћења – Брисање профила 38](#_Toc76515333)

[1.3.4 ДС4: Дијаграм секвенце случаја коришћења – Креирање корисничког налога 40](#_Toc76515334)

[1.3.5 ДС5: Дијаграм секвенце случаја коришћења – Измена корисничког налога 42](#_Toc76515335)

[1.3.6 ДС6: Дијаграм секвенце случаја коришћења – Брисање корисничког налога 44](#_Toc76515336)

[1.3.7 ДС7: Дијаграм секвенце случаја коришћења – Креирање пројекта 46](#_Toc76515337)

[1.3.8 ДС8: Дијаграм секвенце случаја коришћења – Брисање пројекта 48](#_Toc76515338)

[1.3.9 ДС9: Дијаграм секвенце случаја коришћења – Резервисање глумаца 51](#_Toc76515339)

[1.4 Понашање софтверског система – Дефинисање уговора о системским операцијама 56](#_Toc76515340)

[1.5 Структура софтверског система – Концептуални (доменски) модел 58](#_Toc76515341)

[1.6 Структура софтверског система – релациони модел 58](#_Toc76515342)

[Пројектовање 63](#_Toc76515343)

[Имплементација 75](#_Toc76515344)

[Тестирање 77](#_Toc76515345)

[Литература 78](#_Toc76515346)

# 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 дан данас. Под појмом информациона 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.

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.

Поглавље 3 резервисано је за Јава програмски језик и технологије јава програмског језика. Почетком 1990-их година, Џејмс Гозлинг, Патрик Нотон и Мајк Шеридан из компаније Sun Microsystems почињу да развијају програмски језик Jава, 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 објектнооријентисане 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.

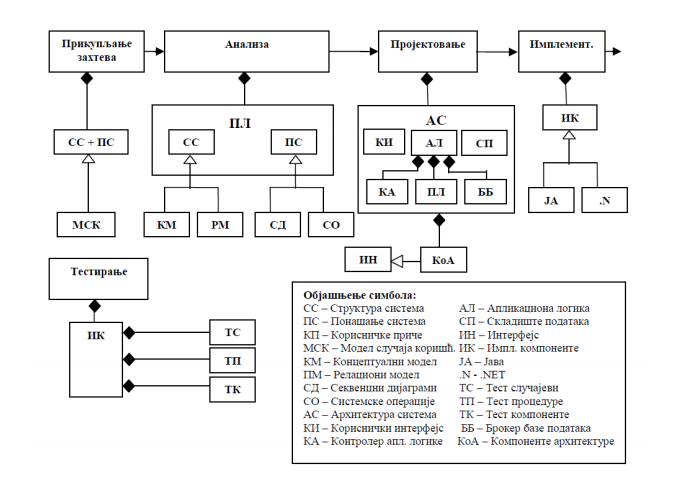
У четвртом поглављу описан је студијски пример развоја система за праћење рада кошаркашког турнира у Јава окружењу, ослањајући се на претходне Јава технологије и водећи се Лармановом методом сачињене од 5 фаза (детаљније о фазама у поглављу):

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

## 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.

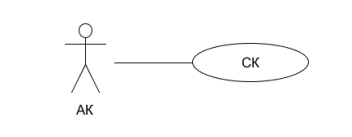


Слика 1 Дијаграм Ларманове методе

## 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.



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.

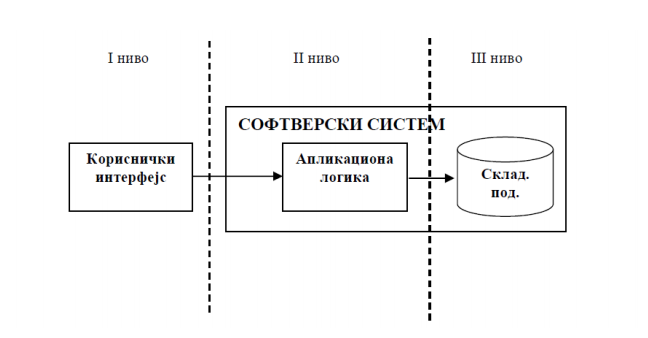
Случај коришћења се у првим фазама представља текстуално, док се касније представља одређеним дијаграмом по избору. Текстуални опис СК има следећу структуру:

* 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.

## 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.



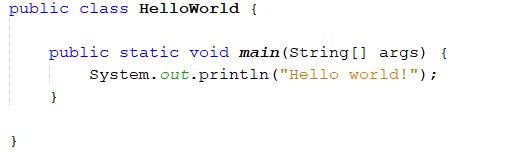
На основу тронивојске архитектуре, направљени су савремени апликациони сервери, који су одговорни да обезбеде сервисе за могућност реализације апликационе логике софтверског система. Сваки апликациони сервер се састоји из три основна дела:

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

## 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:



*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.

## **Концепти објектно-оријентисаног програмирања и Јава програмског језика**

Објектно-оријентисани програми засновани су на концепту објеката 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 покрени, укочи, пребаци у већу брзину итд. Такође, може се рећи да и неки објекат садржи неки други објекат.

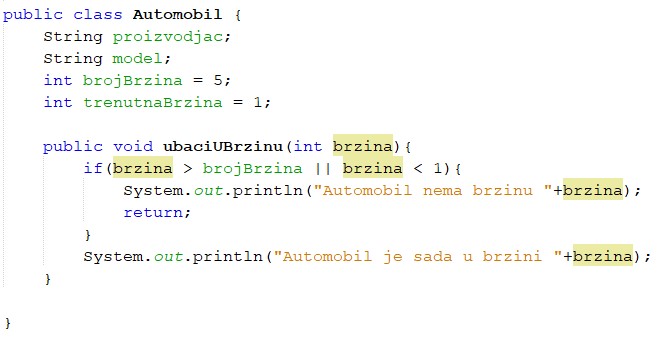
Све ове опсервације из стварног света се преносе на концепте објектно-оријентисаног програмирања. Објекти у ООП језицима се такође састоје из стања и понашања. Стања се чувају у пољима објекта, док се понашања реализују преко метода објекта. Моделирањем тих стања и понашања у програмском језику објекат 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 објеката, 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:



*Слика 7. Код класе Automobil*

Поља proizvodjac, 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

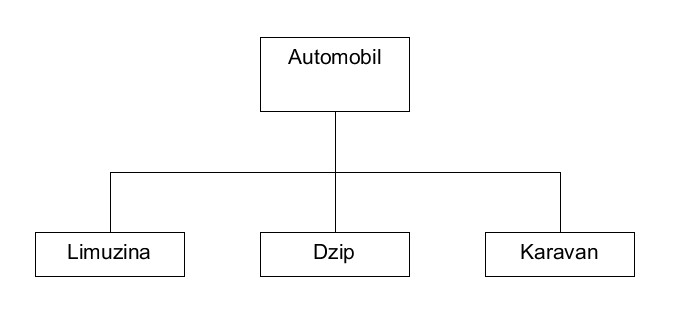
Јава програмски језик је статичног типа, што значи, сва поља морају бити декларисана пре него што могу бити коришћена. Декларација поља врши се писањем типа поља и његовим именом, нпр.

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.

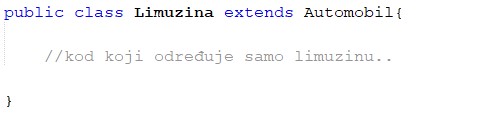
Наслеђивање

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.



*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.

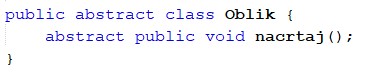
Апстракција, апстрактне класе, инферфејси

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

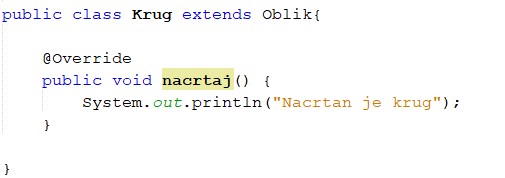
Главна разлика између ова два концепта јесу да се апстрактном класом постиже делимична апстракција, док се интерфејсом постиже потпуна апстракција.

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. Kључна реч за апстрактну класу јесте abstract.

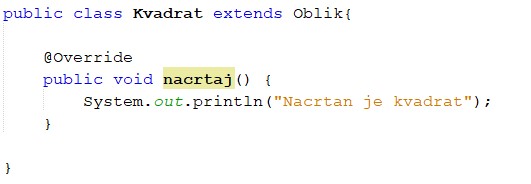


*Слика 10. Апстрактна класа Облик*

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

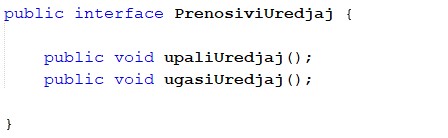


*Слика 11. Јава код подкласе Круг*

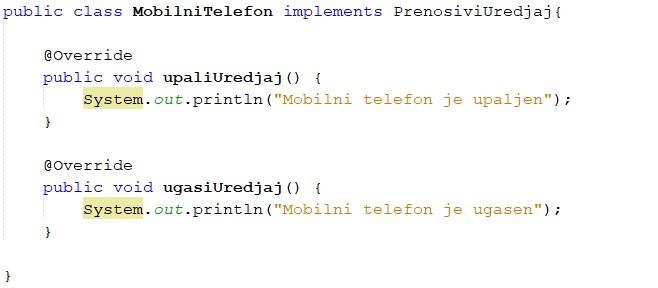


*Слика 12. Јава код подкласе Квадрат*

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.



*Слика 13. Интерфејс*



*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.

## **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). Главнa карактеристикa JFC-a јесте Swing пакет компоненти. Swing API садржи укупно 18 јавних пакета, али главна два пакета:

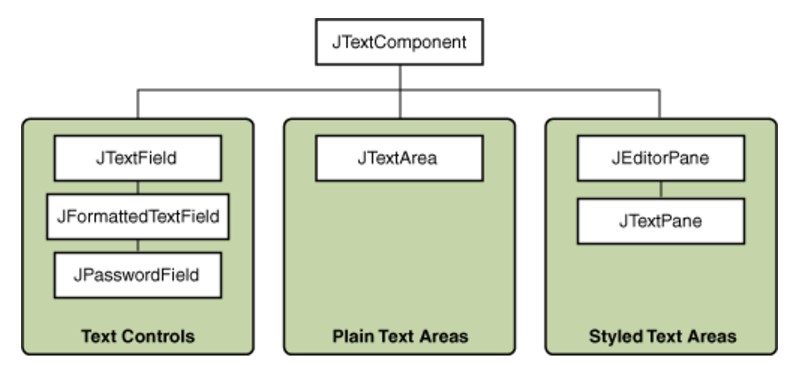
* javax.swing и
* javax.swing.event.

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

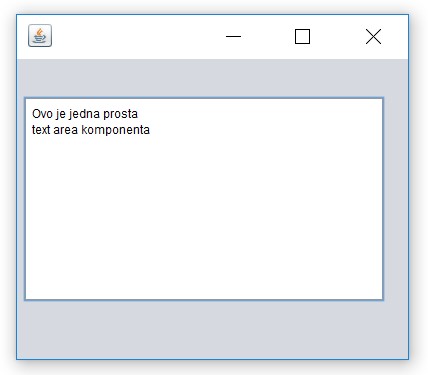
Swing компоненте

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 je независни подпрозор који приказује информације, док је JApplet класа која омогућава аплетима да користе Swing компоненте.

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



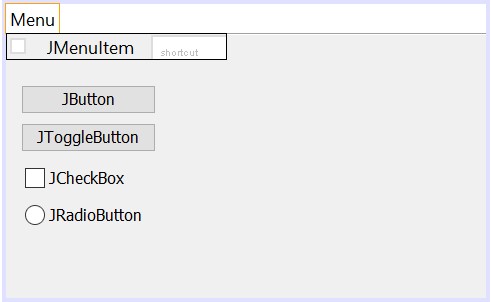
*Слика 15. Хијерархија текстуалних компоненти, Swing*



*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.

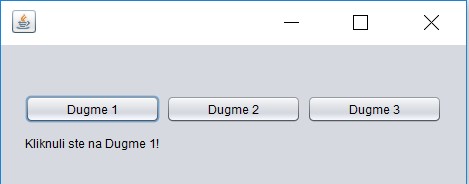


*Слика 17. Неколико типова дугмића*

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-a 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 objekat, и шаље Listener objektu 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.



*Слика 19. Код графичког интерфејса*

## **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://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 објекат (предефинисана класа у Јави) и проследити јој URL као параметар у облику стринг променљиве:

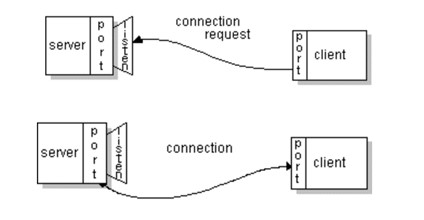
URL adresa = new URL("http://facebook.com/");

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 објекат, 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

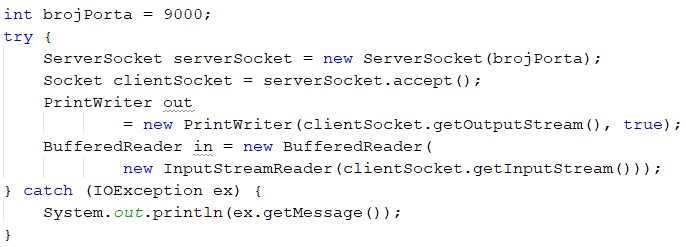
Да би се омогућило повезивање између клијента и сервера на некој мрежној апликацији, сваки од ова два дела програма отвара сокет на крају своје конекције преко које се читају и пишу подаци. Адреса сокета се састоји из два дела: адресе рачунара на коме се налази програм који је генерисао сокет, и број 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 отварати 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.

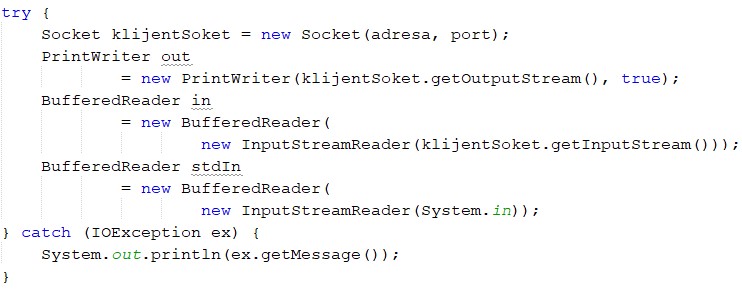


*Слика 21. Приказ кода за подизање серверског сокета*

Основни кораци клијентске стране су следећи:

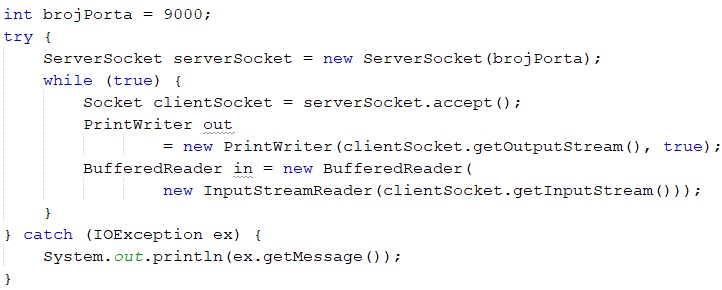
1. Отвара се клијентски сокет, са улазним параметрима adresa и port који представљају адресу серверског рачунара и порт који чека на конекцију(порт који је отворио серверски сокет).
2. Отварају се излазни и улазни токови сокета.
3. Врши се читање и писање преко токова у складу са протоколом сокета.
4.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



*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:



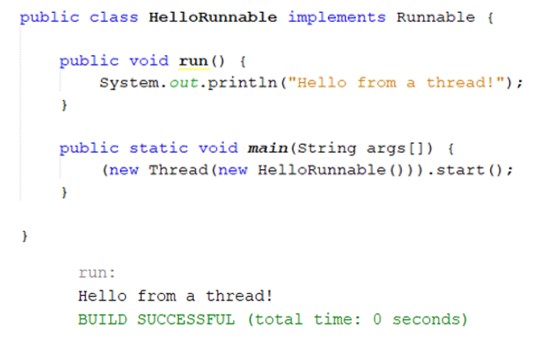
*Слика 23. Сервер који може радити са више клијената*

Угњеждавањем accept методе унутар while петље са параметром true, while петља се непрестано покреће, а самим тим и сваки пут окида accept 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.

## **Нити**

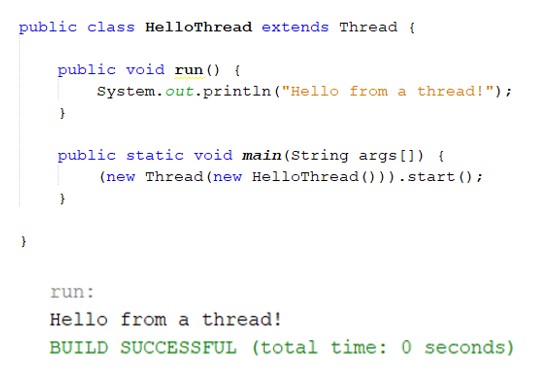
Уколико више процеса међусобно сарађују у извршењу неког задатка, јавља се проблем њихове међусобне комуникације и размене података јер сваки процес заузима посебан меморијски простор. Тај проблем је решен појавом нити (енг. threads) које деле исти меморијски простор. Јава као програмски језик подржава 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.



*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.



*Слика 25. Тhread класа и њен испис*

У оба случаја, нит се позива методом start. Постоје разни облици манипулације нитима, као што су успављивање, прекидање, буђeње, 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.

## Рад са базом података

Повезивање неког прогама који је написан у Јава програмском језику и неког од система за управљање базом податaкa (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-ja,
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 слогове табеле над којом се извршава наредба.



*Слика 26. Извршавање SELECT наредбе над базом података*

# Прикупљање корисничких захтева

## Вербални опис

Потребно је направити апликацију која ће омогућити продукцијама из филмске и маркетинг индустрије да пронађу глумце за њихове потребе. Продукције ће имати своје корисничке налоге на којима ће се чувати информације о њима. Глумци ће имати своје профиле на које ће окачити своје слике као и потребне податке о себи. Глумци ће моћи временом да развијају своје профиле и постављају нове ствари.

Продукције могу да претражују профиле и да добију контакте људи како би могли да раде са глумцима на пројектима. Циљ ове апликације је да се избаце сувишне и скупе кастинг агенције.

## Случајеви коришћења:

Уовој апликацији идентификована су девет случаја коришћења:

1. Креирање профила
2. Промена профила
3. Брисање профила
4. Креирање корисничког налога
5. Измена корисничког налога
6. Брисање корисничког налога
7. Креирање пројекта (сложен)
8. Претраживање глумаца
9. Резервисање глумца (сложен)

**Случајеви коришћења глумца**

Креирање профила

Промена профила

Брисање профила

**Случајеви коришћења продукције**

**Продукција**

**Глумац**

Резервисање глумца

Креирање пројекта

Претраживање глумца

Брисање корисничког налога

Измена корисничког налога

Креирање корисничког налога

### СК1: Случај коришћења – Креирање профила

**Назив СК**

Креирање профила

**Актори СК**

Глумац

**Учесници СК**

Глумац и систем (програм)

**Предуслов**: Систем је укључен. Систем приказује форму за рад са профилом.

**Основни сценарио СК**

1. Глумац **позива** систем да креира профил. (АПСО)
2. Систем **креира** профил. (СО)
3. Систем **приказује** глумцу профил и поруку: “Систем је креирао профил“. (ИА)
4. Глумац **уноси** податке у профил. (АПУСО)
5. Глумац **контролише** да ли је коректно унео податке у профил. (АНСО)
6. Глумац **позива** систем да запамти податке о профилу. (АПСО)
7. Систем **памти** податке о профилу. (СО)
8. Систем **приказује** глумцу запамћени профил и поруку: “Систем је запамтио профил“. (ИА)

**Алтернативна сценарија**

3.1 Уколико систем не може да креира профил он приказује глумцу поруку:

“Систем не може да креира профил”. Прекида се извршење сценариа. (ИА)

8.1 Уколико систем не може да запамти податке о профилу он приказује глумцу поруку “Систем не може да запамти профил”. (ИА)

### СК2: Случај коришћења – Промена профила

**Назив СК**

Промена профила

**Актори СК**

Глумац

**Учесници СК**

Глумац и систем (програм)

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

**Основни сценарио СК**

1. Глумац **позива** систем да учита профил. (АПСО)
2. Систем **учитава** профил. (СО)
3. Систем **приказује** глумцу податке о профилу и поруку: “Систем је учитао профил ”. (ИА)
4. Глумац **уноси** **(мења)** податке о профилу. (АПУСО)
5. Глумац **контролише** да ли је коректно унео податке о профилу. (АНСО)
6. Глумац **позива** систем да запамти податке о профилу. (АПСО)
7. Систем **памти** податке о профилу. (СО)
8. Систем **приказује** глумцу запамћени профил и поруку: “Систем је запамтио профил.” (ИА)

**Алтернативна сценарија**

3.1 1 Уколико систем не може да учита профил он приказује глумцу поруку:

“Систем не може да учита профил”. Прекида се извршење сценариа. (ИА)

8.1 Уколико систем не може да запамти податке о профилу он приказује глумцу поруку “Систем не може да запамти профил”. Прекида се извршење сценариа.

(ИА)

### СК3: Случај коришћења – Брисање профила

**Назив СК**

Брисање профила

**Актори СК**

Глумац

**Учесници СК**

Глумац и систем (програм)

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

**Основни сценарио СК**

1. Глумац **позива** систем да учита профил. (АПСО)
2. Систем **учитава** профил. (СО)
3. Систем **приказује** глумцу податке о профилу и поруку: “Систем је учитао профил ”. (ИА)
4. Глумац **позива** систем да обрише профил. (АПСО)
5. Систем **брише** профил. (СО)
6. Систем **приказује** глумцу поруку: “Систем је обрисао профил.” (ИА)

**Алтернативна сценарија**

3.1 Уколико систем не може да обрише профил он приказује глумцу поруку

“Систем не може да обрише профил”. (ИА)

### СК4: Случај коришћења – Креирање корисничког налога

**Назив СК**

Креирање корисничког налога

**Актори СК**

Продукција

**Учесници СК**

Продукција и систем (програм)

**Предуслов**: Систем је укључен. Систем приказује форму за рад са корисничким налогом.

**Основни сценарио СК**

1. Продукција **позива** систем да креира кориснички налог. (АПСО)
2. Систем **креира** кориснички налог. (СО)
3. Систем **приказује** продукцији кориснички налог и поруку: “Систем је креирао кориснички налог“. (ИА)
4. Продукција **уноси** податке у кориснички налог. (АПУСО)
5. Продукција **контролише** да ли је коректно унела податке у кориснички налог. (АНСО)
6. Продукција **позива** систем да запамти податке о корисничком налогу. (АПСО)
7. Систем **памти** податке о корисничком налогу. (СО)
8. Систем **приказује** продукцији запамћени кориснички налог и поруку: “Систем је запамтио кориснички налог“. (ИА)

**Алтернативна сценарија**

3.1 Уколико систем не може да креира кориснички налог он приказује продукцији поруку: “Систем не може да креира кориснички налог”. Прекида се извршење сценариа. (ИА)

8.1 Уколико систем не може да запамти податке о корисничком налогу он приказује продукцији поруку “Систем не може да запамти кориснички налог”. (ИА)

### СК5: Случај коришћења – Измена корисничког налога

**Назив СК**

Промена корисничког налога

**Актори СК**

Продукција

**Учесници СК**

Продукција и систем (програм)

**Предуслов**: Систем је укључен и продукција je улогована под својом шифром. Систем приказује форму за рад са корисничким налогом.

**Основни сценарио СК**

1. Продукција **уноси** **(мења)** податке о корисничком налогу. (АПУСО)
2. Продукција **контролише** да ли је коректно унела податке о корисничком налогу. (АНСО)
3. Продукција **позива** систем да запамти податке о корисничком налогу. (АПСО)
4. Систем **памти** податке о корисничком налогу. (СО)
5. Систем **приказује** продукцији запамћени кориснички налог и поруку: “Систем је запамтио кориснички налог.” (ИА)

**Алтернативна сценарија**

5.1 Уколико систем не може да запамти податке о корисничком налогу он приказује продукцији поруку “Систем не може да запамти кориснички налог”. Прекида се извршење сценариа. (ИА)

### СК6: Случај коришћења – Брисање корисничког налога

**Назив СК**

Брисање корисничког налога

**Актори СК**

Продукција

**Учесници СК**

Продукција и систем (програм)

**Предуслов**: Систем је укључен и продукција je улогована под својом шифром. Систем приказује форму за рад са корисничким налогом.

**Основни сценарио СК**

1. Продукција **уноси** вредностпо којој претражује кориснички налог. (АПУСО)
2. Продукција **позива** систем да нађе кориснички налог по задатој вредности. (АПСО)
3. Систем **тражи** кориснички налог по задатој вредности. (СО)
4. Систем приказује продукцији кориснички налог и поруку: “Систем је нашао кориснички налог по задатој вредности”. (ИА)
5. Продукција **позива** систем да обрише кориснички налог. (АПСО)
6. Систем **брише** кориснички налог. (СО)
7. Систем **приказује** продукцији поруку: “Систем је обрисао кориснички налог.” (ИА)

**Алтернативна сценарија**

4.1 Уколико систем не може да нађе кориснички налог он приказује продукцији поруку: “Систем не може да нађе кориснички налог по задатој вредности”. Прекида се извршење сценариа. (ИА)

7.1 Уколико систем не може да обрише кориснички налог он приказује продукцији поруку “Систем не може да обрише кориснички налог”. (ИА)

### СК7: Случај коришћења – Креирање пројекта

**Назив СК**

Креирање пројекта

**Актори СК**

Продукција

**Учесници СК**

Продукција и систем (програм)

**Предуслов**: Систем је укључен и продукција je улогована под својом шифром. Систем приказује форму за рад са пројектом.

**Основни сценарио СК**

1. Продукција **позива** систем да креира пројекат. (АПСО)
2. Систем **креира** пројекат. (СО)
3. Систем **приказује** продукцији пројекат и поруку: “Систем је креирао пројекат“. (ИА)
4. Продукција **уноси** податке у пројекат. (АПУСО)
5. Продукција **контролише** да ли је коректно унео податке у пројекат. (АНСО)
6. Продукција **позива** систем да запамти податке о пројекту. (АПСО)
7. Систем **памти** податке о пројекту. (СО)
8. Систем **приказује** продукцији запамћени пројекат и поруку: “Систем је запамтио пројекат“. (ИА)

**Алтернативна сценарија**

3.1 Уколико систем не може да креира пројекат он приказује продукцији поруку: “Систем не може да креира пројекат”. Прекида се извршење сценариа. (ИА)

8.1 Уколико систем не може да запамти податке о пројекту он приказује продукцији поруку “Систем не може да запамти пројекат”. (ИА)

### СК8: Случај коришћења – Брисање пројекта

**Назив СК**

Брисање пројекта

**Актори СК**

Продукција

**Учесници СК**

Продукција и систем (програм)

**Предуслов**: Систем је укључен и продукција je улогована под својом шифром. Систем приказује форму за рад са пројектима. Учитана је листа пројекта.

**Основни сценарио СК**

1. Продукција **уноси** вредностпо којој претражује кориснички налог. (АПУСО)
2. Продукција **позива** систем да нађе пројекат по задатој вредности. (АПСО)
3. Систем **тражи** пројекат по задатој вредности. (СО)
4. Систем приказује продукцији кориснички налог и поруку: “Систем је нашао пројекат по задатој вредности”. (ИА)
5. Продукција **позива** систем да обрише пројекат. (АПСО)
6. Систем **брише** пројекат. (СО)
7. Систем **приказује** продукцији поруку: “Систем је обрисао пројекат.” (ИА)

**Алтернативна сценарија**

8.1 Уколико систем не може да учита пројекте он приказује продукцији поруку: “Систем не може да учита пројекте”. (ИА)

### СК9: Случај коришћења – Резервисање глумца

**Назив СК**

Креирање резервације глумца

**Актори СК**

Продукција

**Учесници СК**

Продукција и систем (програм)

**Предуслов**: Систем је укључен и продукција je улогована под својом шифром. Систем приказује форму за рад са резервацијом глумца. Учитана је листа глумаца.

**Основни сценарио СК**

1. Продукција **уноси** вредностпо којој претражује профиле глумаца. (АПУСО)
2. Продукција **позива** систем да нађе профиле глумаца по задатој вредности. (АПСО)
3. Систем **тражи** профиле глумаца по задатој вредности. (СО)
4. Систем приказује продукцији профиле глумаца и поруку: “Систем је нашао профиле глумаца по задатој вредности”. (ИА)
5. Продукција **бира** профил глумца. (АПУСО)
6. Продукција **позива** систем да учита профил глумца. (АПСО)
7. Систем **учитава** профил глумца. (СО)
8. Систем приказује продукцији профил глумца и поруку: “Систем је учитао профил глумца”. (ИА)
9. Продукција **позива** систем да креира резервацију глумца. (АПСО)
10. Систем **креира** резервацију глумца. (СО)
11. Систем **приказује** продукцији резервацију глумца и поруку: “Систем је креирао резервацију глумца“. (ИА)

**Алтернативна сценарија**

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

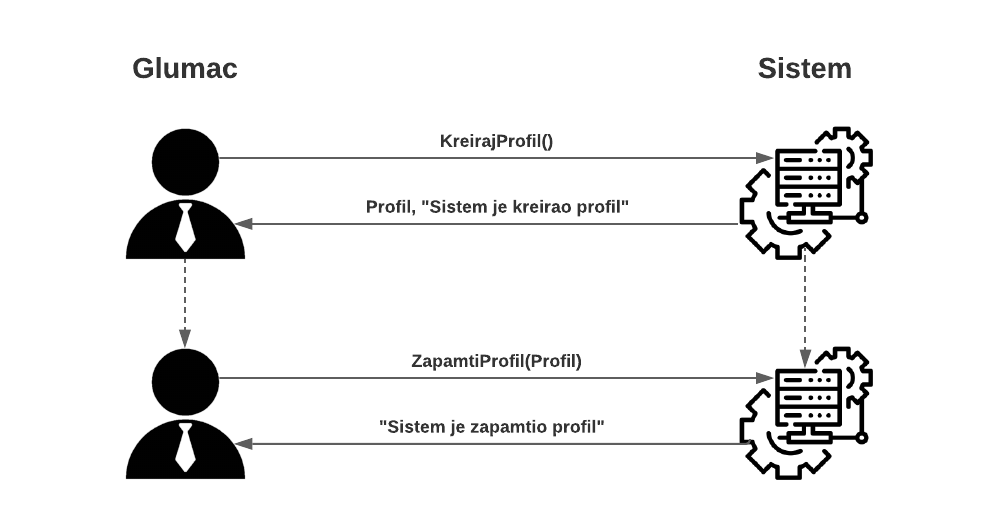
8.1 Уколико систем не може да учита профиле глумаца он приказује продукцији поруку: “Систем не може да учита профиле глумаца”. (ИА)

11.1 Уколико систем не може да креира резервацију глумца он приказује продукцији поруку: “Систем не може да креира резервацију глумца”.

# Анализа

## Системски дијаграм секвенци

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



**Предуслов**: Систем је укључен. Систем приказује форму за рад са профилом.

**Основни сценарио СК**

1. Глумац **позива** систем да креира профил. (АПСО)
2. Систем **приказује** глумцу профил и поруку: “Систем је креирао профил“. (ИА)
3. Глумац **позива** систем да запамти податке о профилу. (АПСО)
4. Систем **приказује** глумцу запамћени профил и поруку: “Систем је запамтио профил“. (ИА)

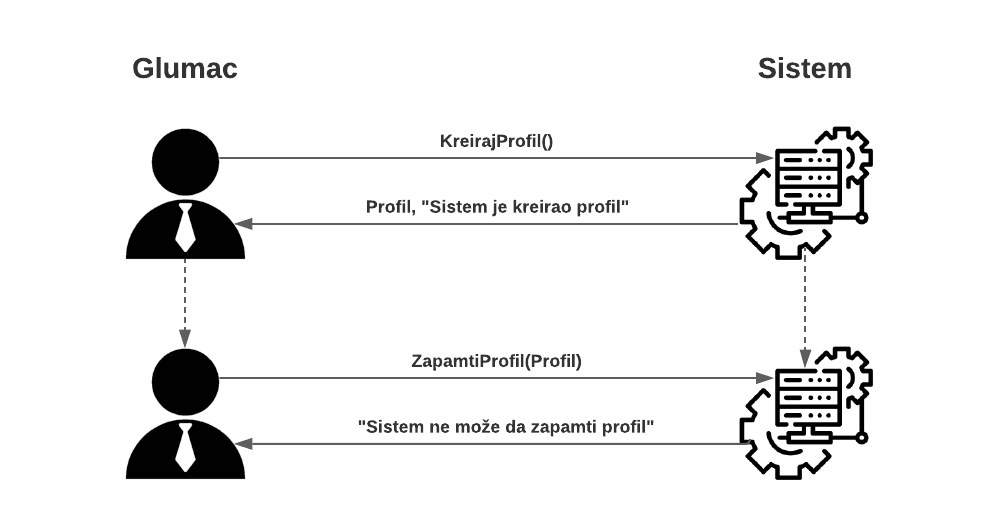
**Алтернативна сценарија**

2.1 Уколико систем не може да креира профил он приказује глумцу поруку:

“Систем не може да креира профил”. Прекида се извршење сценариа. (ИА)

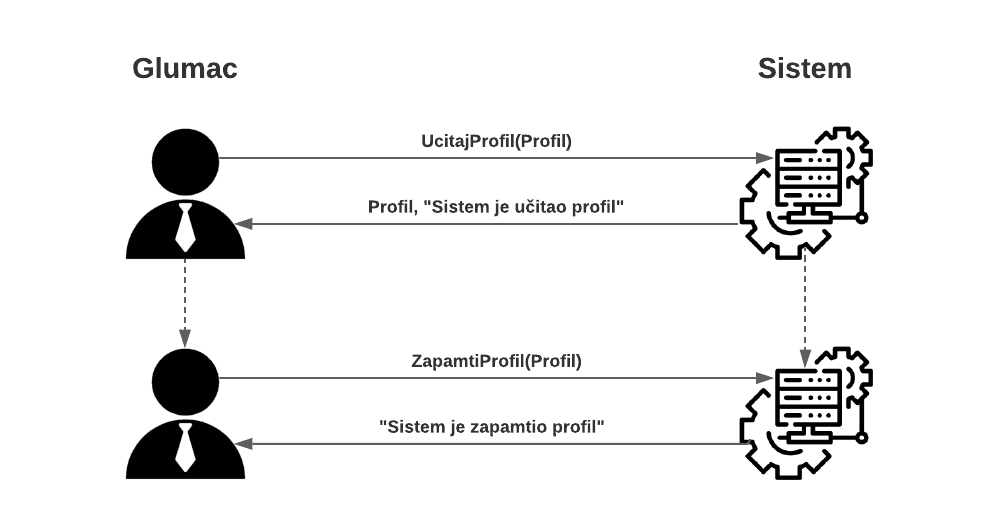


4.1 Уколико систем не може да запамти податке о профилу он приказује глумцу поруку “Систем не може да запамти профил”. (ИА)



Са наведених секвенцних дијаграма уочавају се 2 системске операције:

1. Signal **KreirajProfil();**
2. Signal **ZapamtiProfil(Profil)**.
   * 1. ДС2: Дијаграм секвенце случаја коришћења – Промена профила



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

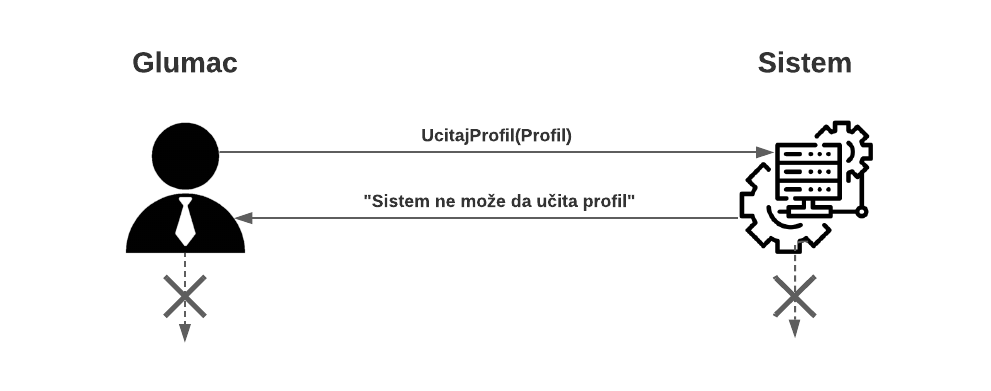
**Основни сценарио СК**

1. Глумац **позива** систем да учита профил. (АПСО)
2. Систем **приказује** глумцу податке о профилу и поруку: “Систем је учитао профил ”. (ИА)
3. Глумац **позива** систем да запамти податке о профилу. (АПСО)
4. Систем **приказује** глумцу запамћени профил и поруку: “Систем је запамтио профил.” (ИА)

**Алтернативна сценарија**

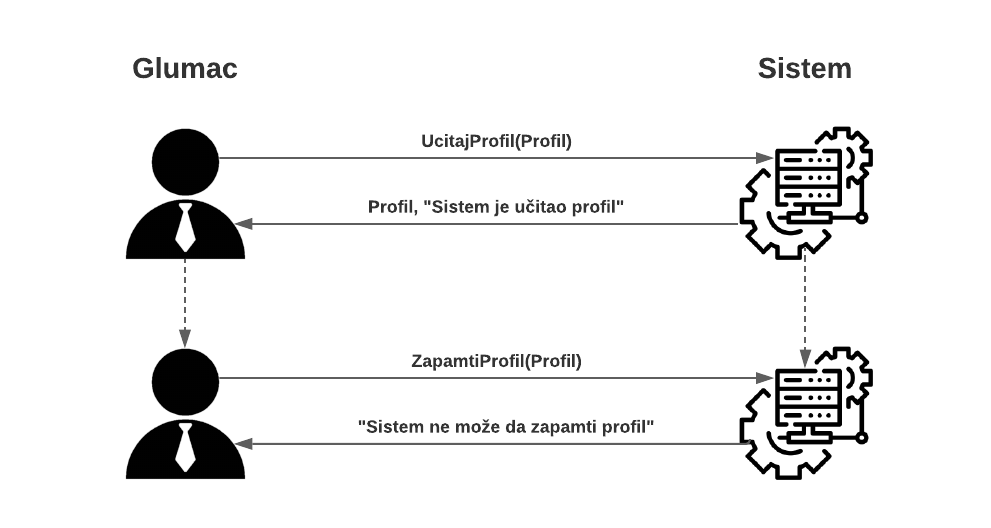
2.1 Уколико систем не може да учита профил он приказује глумцу поруку:

“Систем не може да учита профил”. Прекида се извршење сценариа. (ИА)



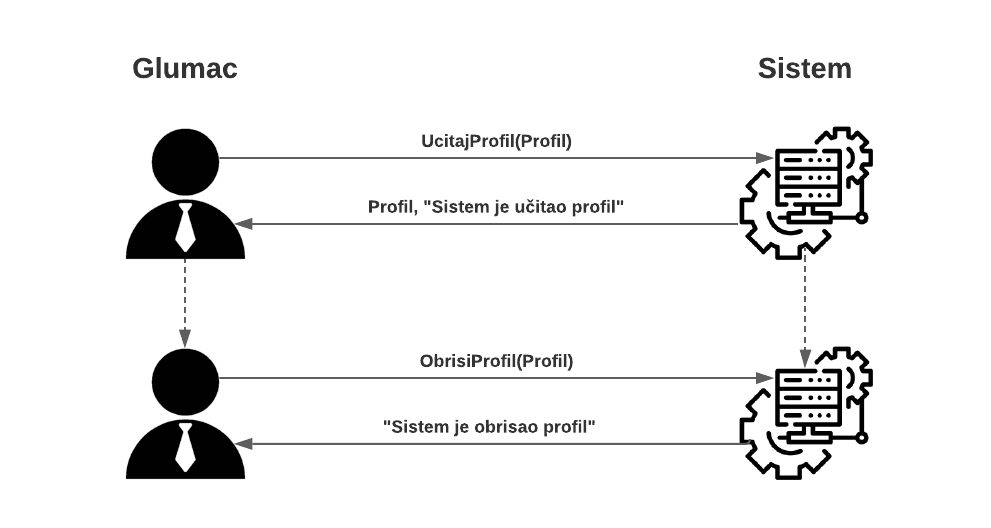
4.1 Уколико систем не може да запамти податке о профилу он приказује глумцу поруку “Систем не може да запамти профил”. Прекида се извршење сценариа.

(ИА)



Са наведених секвенцних дијаграма уочавају се 2 системске операције:

1. Signal **UcitajProfil(Profil);**
2. Signal **ZapamtiProfil(Profil)**.
   * 1. ДС3: Дијаграм секвенце случаја коришћења – Брисање профила



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

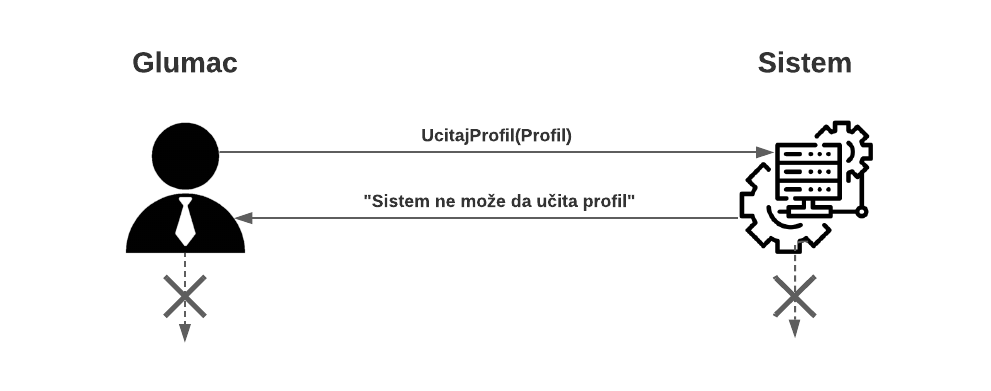
**Основни сценарио СК**

1. Глумац **позива** систем да учита профил. (АПСО)
2. Систем **приказује** глумцу податке о профилу и поруку: “Систем је учитао профил ”. (ИА)
3. Глумац **позива** систем да обрише профил. (АПСО)
4. Систем **приказује** глумцу поруку: “Систем је обрисао профил.” (ИА)

**Алтернативна сценарија**

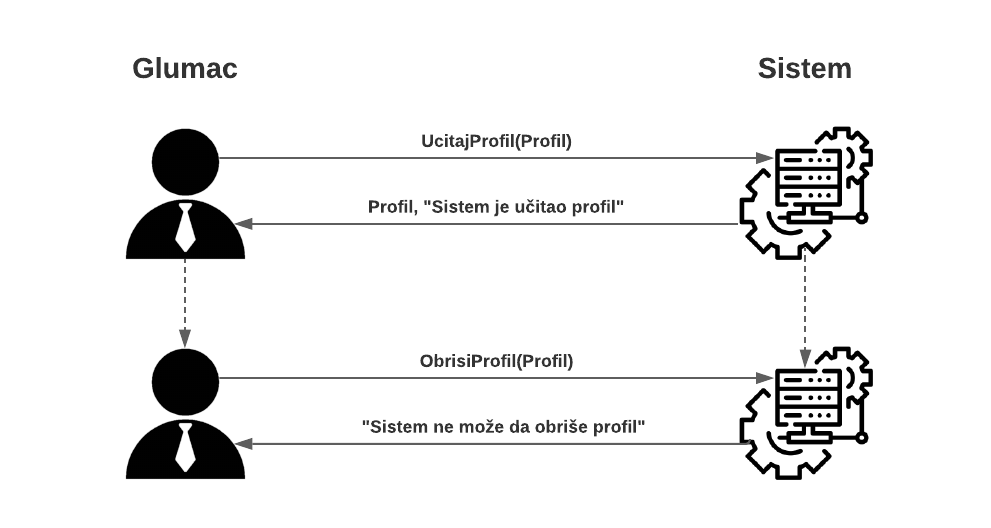
2.1 Уколико систем не може да учита профил он приказује глумцу поруку:

“Систем не може да учита профил”. Прекида се извршење сценариа. (ИА)



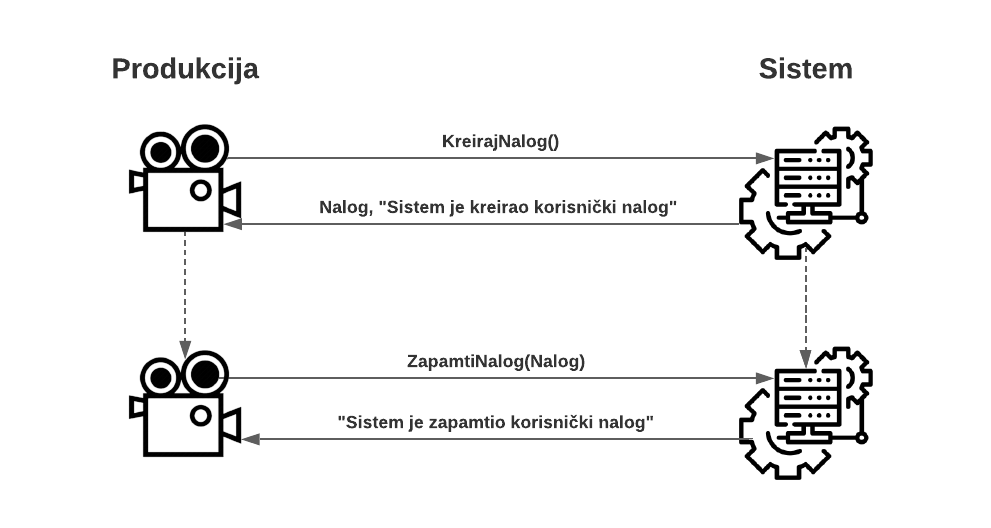
4.1 Уколико систем не може да обрише профил он приказује глумцу поруку

“Систем не може да обрише профил”. (ИА)



Са наведених секвенцних дијаграма уочавају се 2 системске операције:

1. Signal **UcitajProfil(Profil);**
2. Signal **ObrisiProfil(Profil)**.
   * 1. ДС4: Дијаграм секвенце случаја коришћења – Креирање корисничког налога



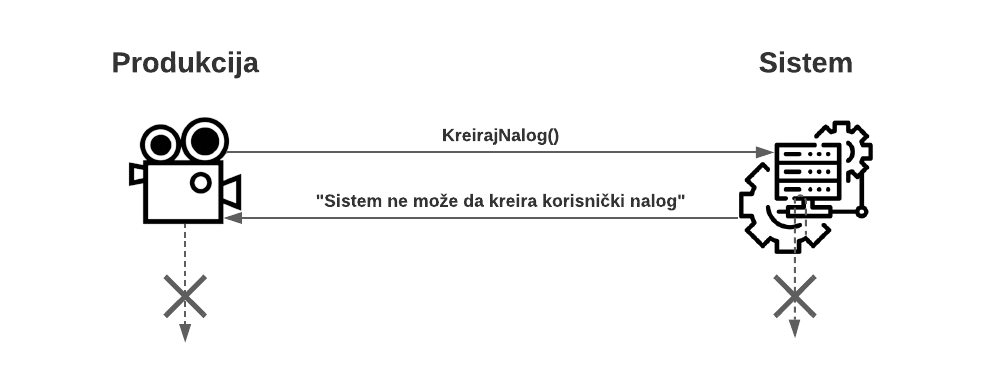
**Предуслов**: Систем је укључен. Систем приказује форму за рад са корисничким налогом.

**Основни сценарио СК**

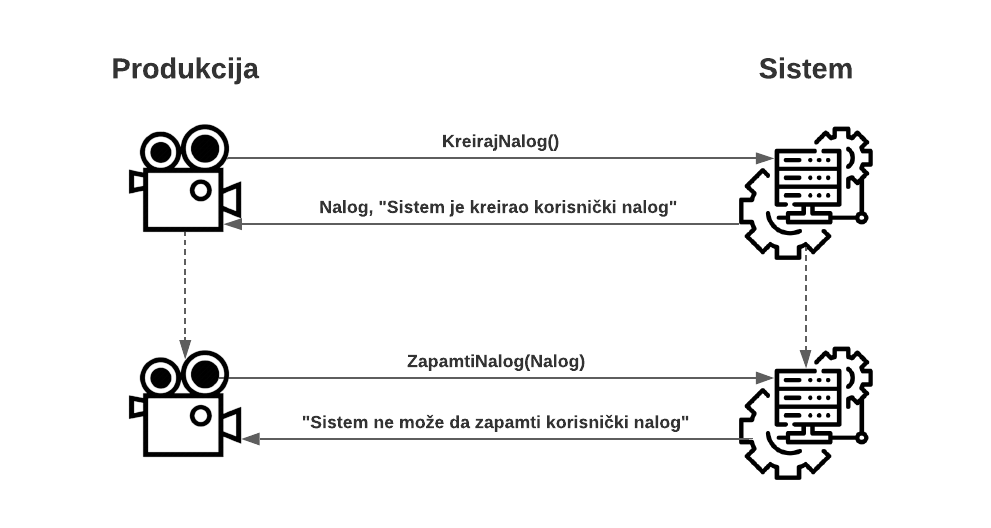
1. Продукција **позива** систем да креира кориснички налог. (АПСО)
2. Систем **приказује** продукцији кориснички налог и поруку: “Систем је креирао кориснички налог“. (ИА)
3. Продукција **позива** систем да запамти податке о корисничком налогу. (АПСО)
4. Систем **приказује** продукцији запамћени кориснички налог и поруку: “Систем је запамтио кориснички налог“. (ИА)

**Алтернативна сценарија**

2.1 Уколико систем не може да креира кориснички налог он приказује продукцији поруку: “Систем не може да креира кориснички налог”. Прекида се извршење сценариа. (ИА)

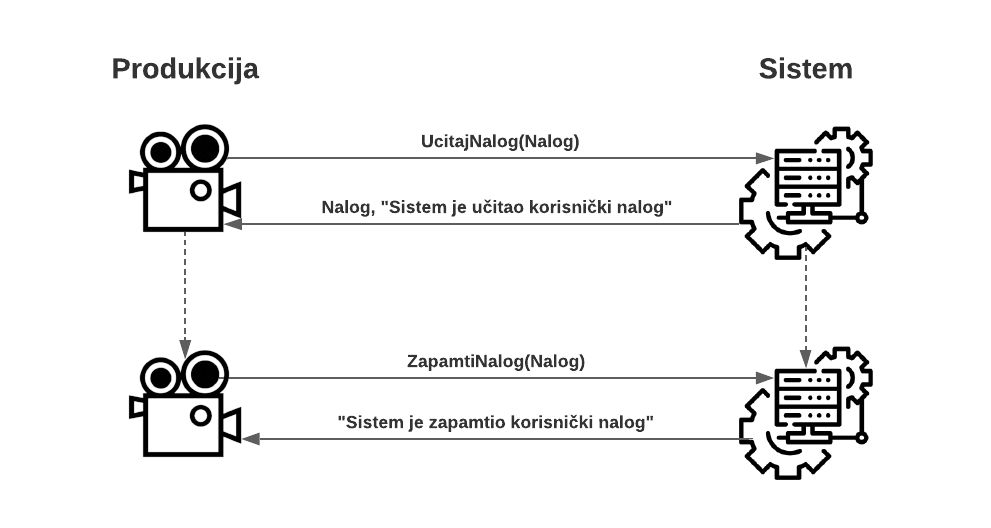


4.1 Уколико систем не може да запамти податке о корисничком налогу он приказује продукцији поруку “Систем не може да запамти кориснички налог”. (ИА)



Са наведених секвенцних дијаграма уочавају се 2 системске операције:

1. Signal **KreirajNalog();**
2. Signal **ZapamtiNalog(Nalog)**.
   * 1. ДС5: Дијаграм секвенце случаја коришћења – Измена корисничког налога



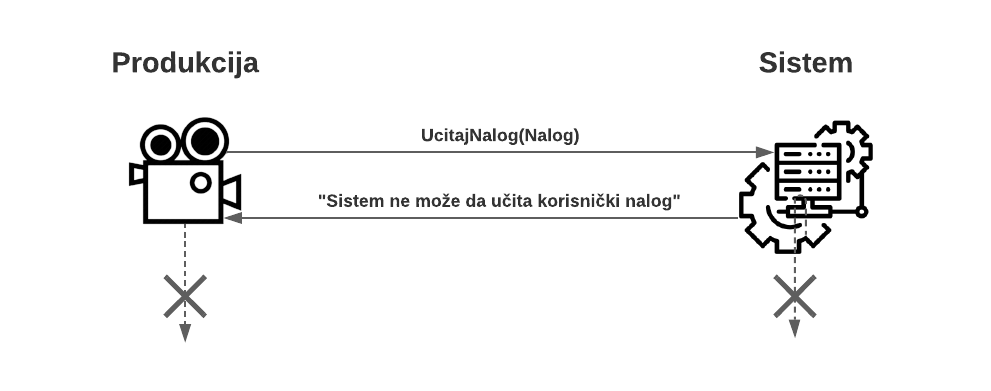
**Предуслов**: Систем је укључен и продукција је улогована под својом шифром. Систем приказује форму за рад са корисничким налогом.

**Основни сценарио СК**

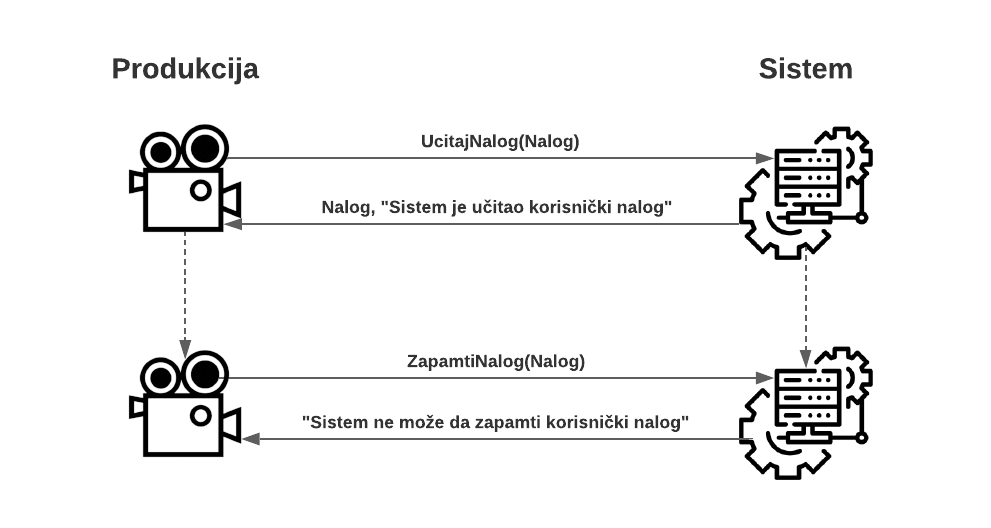
1. Продукција **позива** систем да учита кориснички налог. (АПСО)
2. Систем **приказује** продукцији кориснички налог и поруку: “Систем је учитао кориснички налог“. (ИА)
3. Продукција **позива** систем да запамти податке о корисничком налогу. (АПСО)
4. Систем **приказује** продукцији запамћени кориснички налог и поруку: “Систем је запамтио кориснички налог“. (ИА)

**Алтернативна сценарија**

2.1 Уколико систем не може да учита кориснички налог он приказује продукцији поруку: “Систем не може да учита кориснички налог”. Прекида се извршење сценариа. (ИА)

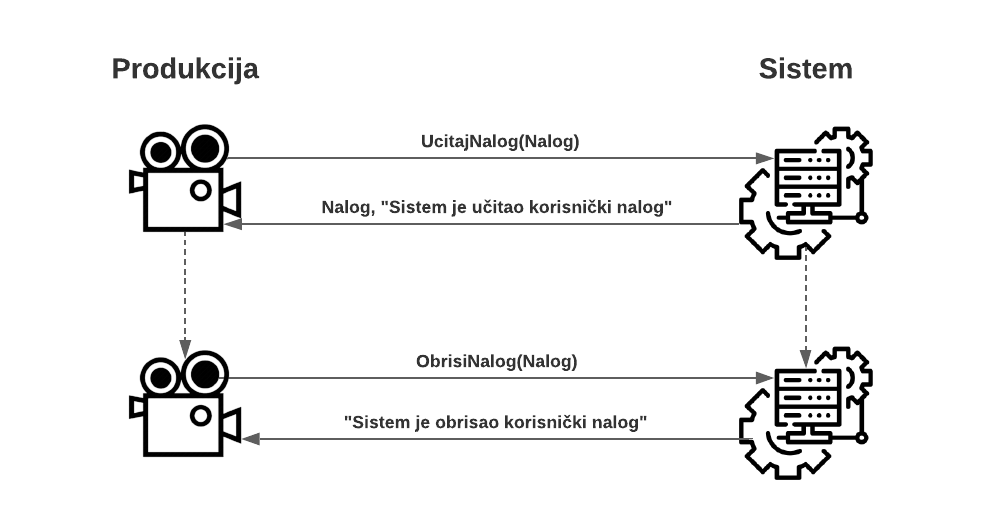


4.1 Уколико систем не може да запамти податке о корисничком налогу он приказује продукцији поруку “Систем не може да запамти кориснички налог”. (ИА)



Са наведених секвенцних дијаграма уочавају се 2 системске операције:

1. Signal **UcitajNalog(Nalog);**
2. Signal **ZapamtiNalog(Nalog)**.
   * 1. ДС6: Дијаграм секвенце случаја коришћења – Брисање корисничког налога



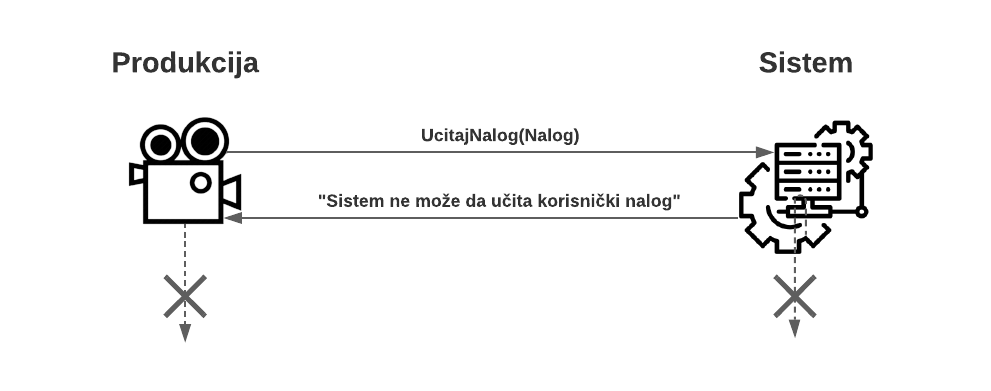
**Предуслов**: Систем је укључен и продукција је улогована под својом шифром. Систем приказује форму за рад са корисничким налогом.

**Основни сценарио СК**

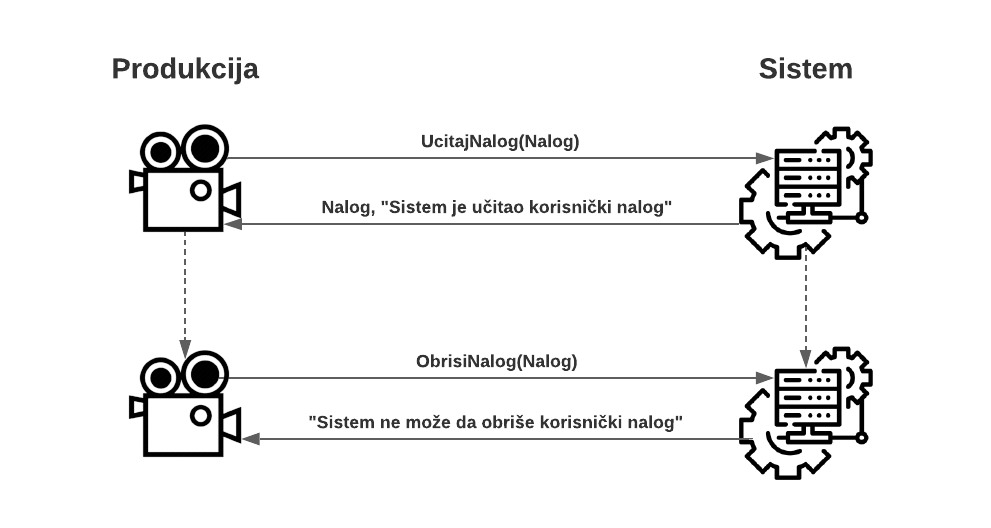
1. Продукција **позива** систем да учита кориснички налог. (АПСО)
2. Систем **приказује** продукцији кориснички налог и поруку: “Систем је учитао кориснички налог“. (ИА)
3. Продукција **позива** систем да обрише кориснички налог. (АПСО)
4. Систем **приказује** продукцији обрисани кориснички налог и поруку: “Систем је обрисано кориснички налог“. (ИА)

**Алтернативна сценарија**

2.1 Уколико систем не може да учита кориснички налог он приказује продукцији поруку: “Систем не може да учита кориснички налог”. Прекида се извршење сценариа. (ИА)

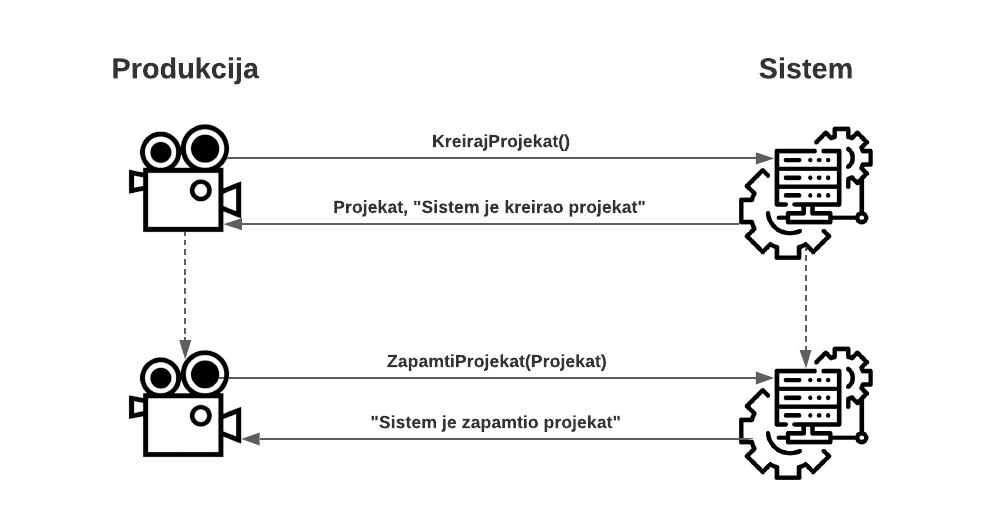


4.1 Уколико систем не може да обрише податке о корисничком налогу он приказује продукцији поруку “Систем не може да обрише кориснички налог”. (ИА)



Са наведених секвенцних дијаграма уочавају се 2 системске операције:

1. Signal **UcitajNalog(Nalog);**
2. Signal **ObrisiNalog(Nalog)**.
   * 1. ДС7: Дијаграм секвенце случаја коришћења – Креирање пројекта



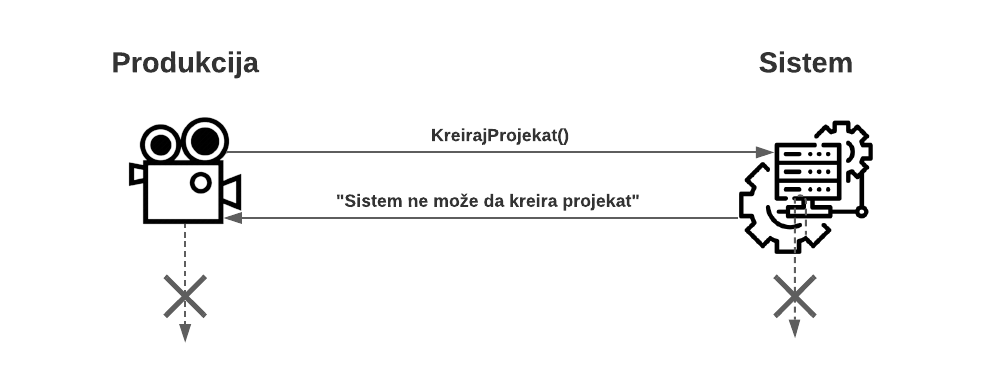
**Предуслов**: Систем је укључен и продукција je улогована под својом шифром. Систем приказује форму за рад са пројектом.

**Основни сценарио СК**

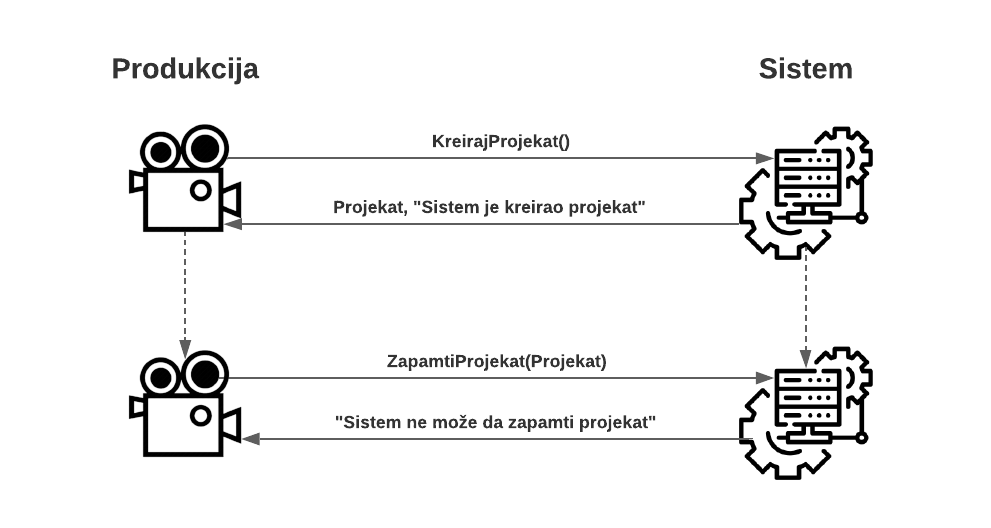
1. Продукција **позива** систем да креира пројекат. (АПСО)
2. Систем **приказује** продукцији пројекат и поруку: “Систем је креирао пројекат“. (ИА)
3. Продукција **позива** систем да запамти податке о пројекту. (АПСО)
4. Систем **приказује** продукцији запамћени пројекат и поруку: “Систем је запамтио пројекат“. (ИА)

**Алтернативна сценарија**

2.1 Уколико систем не може да креира пројекат он приказује продукцији поруку: “Систем не може да креира пројекат”. Прекида се извршење сценариа. (ИА)

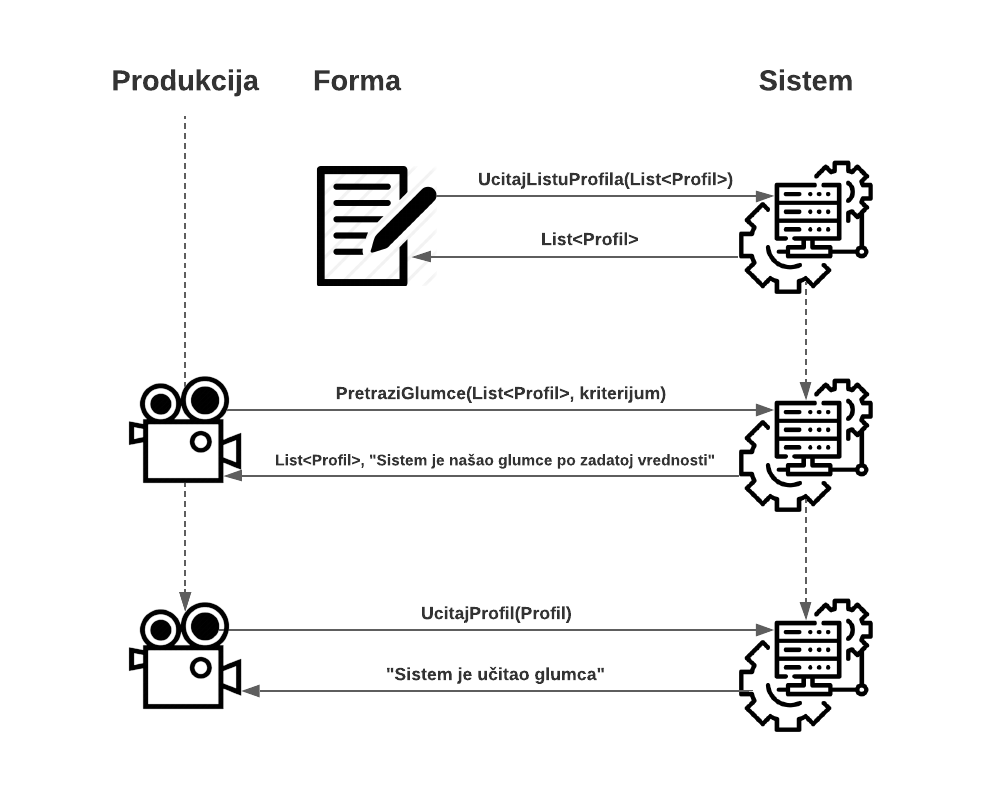


4.1 Уколико систем не може да запамти податке о пројекту он приказује продукцији поруку “Систем не може да запамти пројекат”. (ИА)



Са наведених секвенцних дијаграма уочавају се 2 системске операције:

1. Signal **KreirajProjekat();**
2. Signal **ZapamtiProjekat(Projekat)**.
   * 1. ДС8: Дијаграм секвенце случаја коришћења – Брисање пројекта



**Предуслов**: Систем је укључен и продукција je улогована под својом шифром. Систем приказује форму за рад са пројектима. Учитана је листа пројеката.

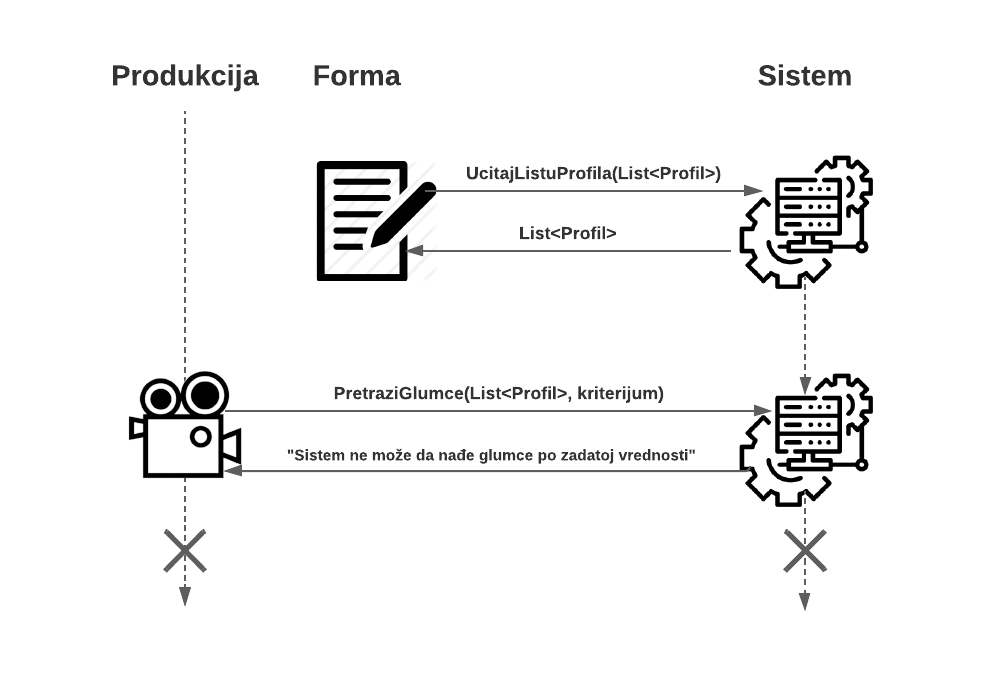
**Основни сценарио СК**

1. Форма позива систем да учита листу пројеката. (АПСО)
2. Систем враћа форми листу пројеката. (ИА)
3. Продукција **позива** систем да нађе пројекат по задатој вредности. (АПСО)
4. Систем приказује продукцији податке о глумцима и поруку: “Систем је нашао пројекат по задатој вредности”. (ИА)
5. Продукција **позива** систем да обрише пројекат. (АПСО)
6. Систем приказује продукцији поруку: “Систем је обрисао пројекат”. (ИА)

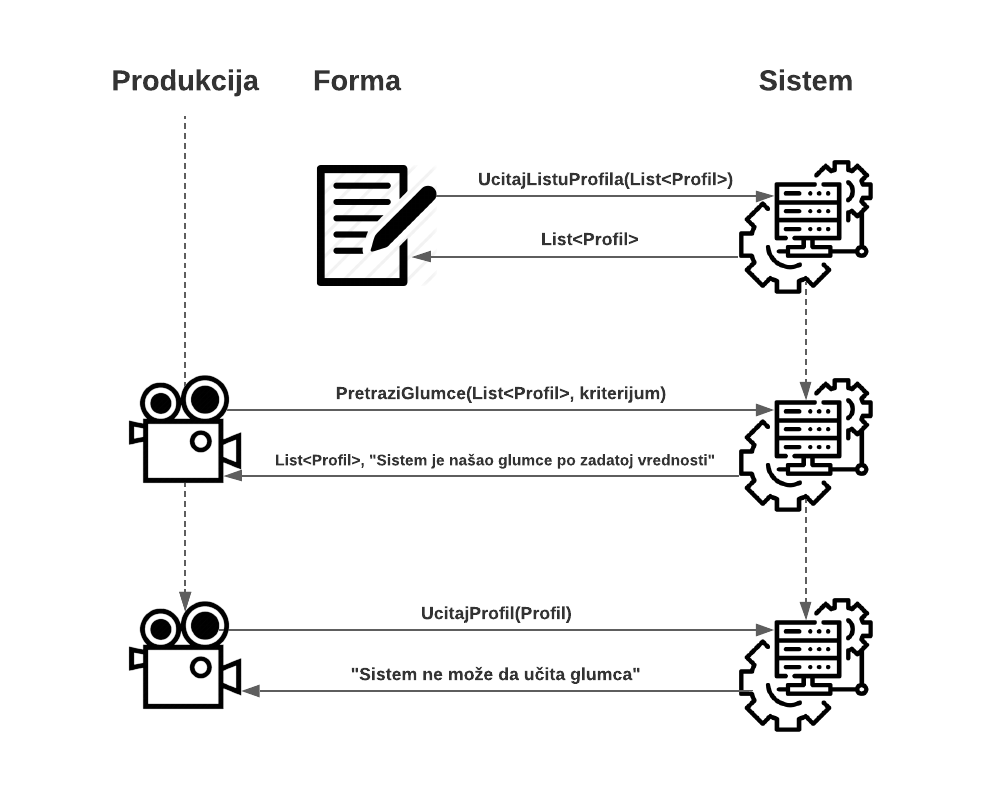
**Алтернативна сценарија**

4.1 Уколико систем не може да нађе глумца он приказује продукцији поруку:

“Систем не може да нађе глумце по задатој вредности”. Прекида се извршење сценарија. (ИА)

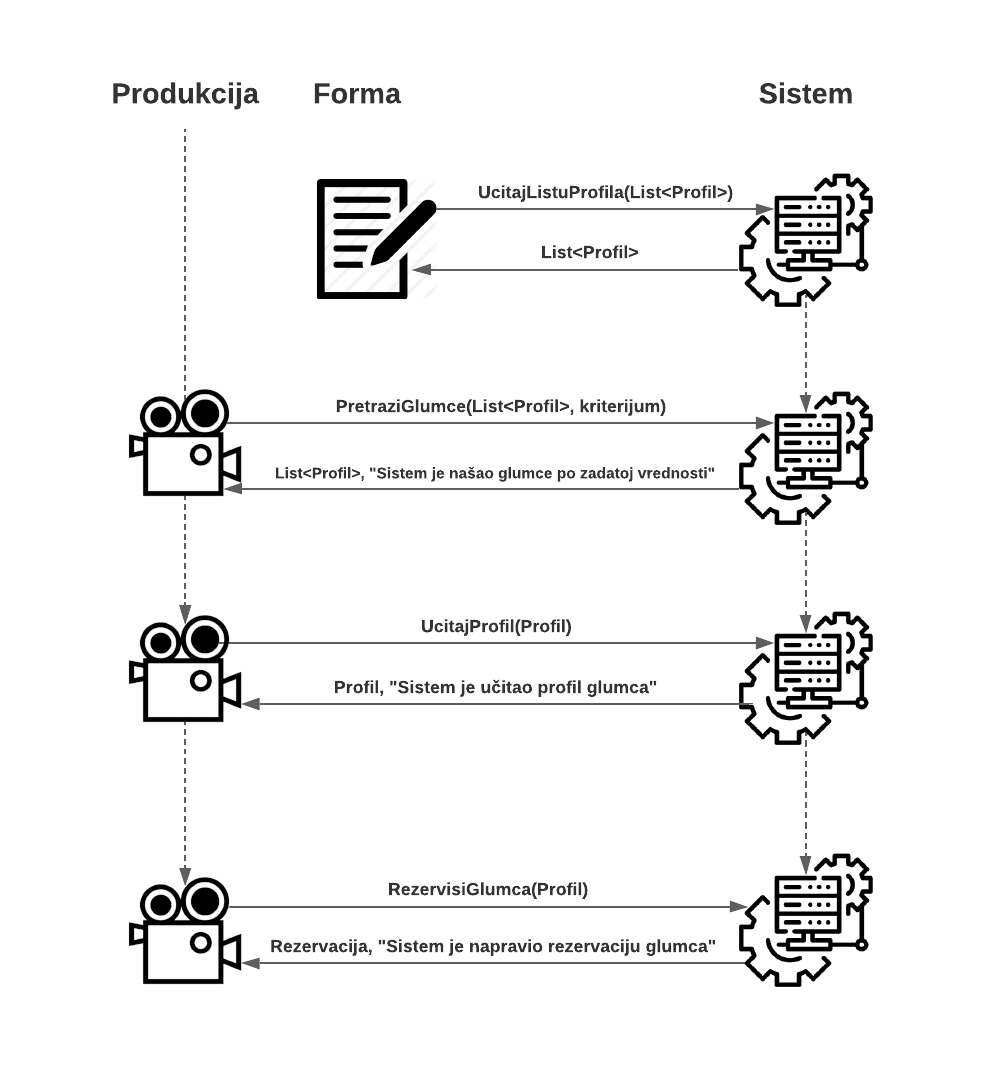


6.1 Уколико систем не може да учита глумца он приказује продукцији поруку: “Систем не може да учита глумца”. (ИА)



Са наведених секвенцних дијаграма уочавају се 3 системске операције:

1. Signal **UcitajListuProfila(List<Profil);**
2. Signal **PretraziGlumce(List<Profil>, kriterijum);**
3. Signal **UcitajProfil(Profil).**
   * 1. ДС9: Дијаграм секвенце случаја коришћења – Резервисање глумаца



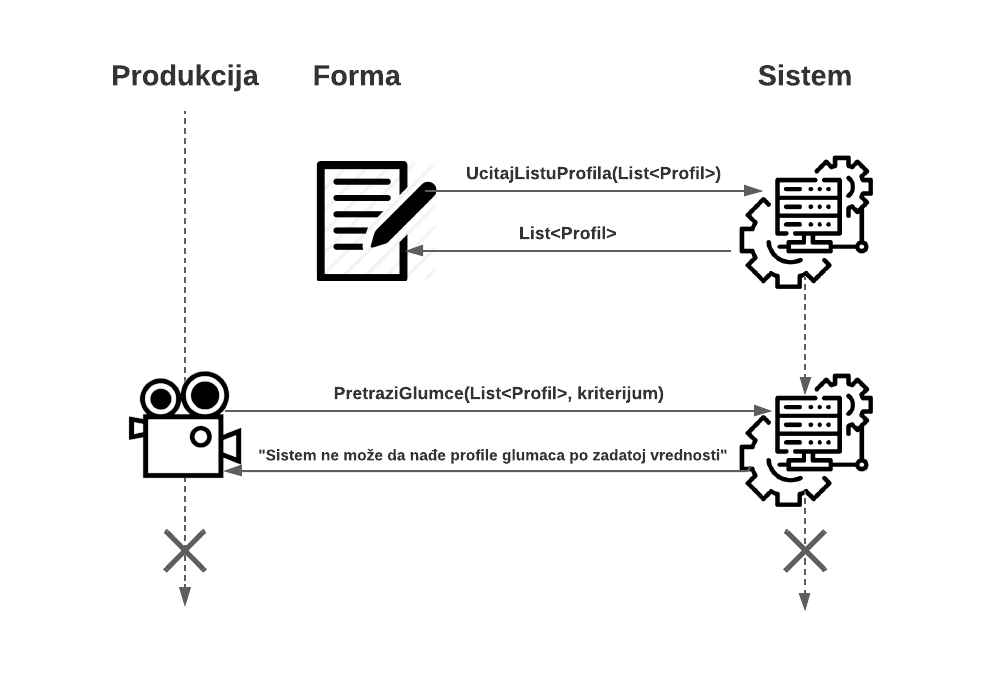
**Предуслов**: Систем је укључен и продукција je улогована под својом шифром. Систем приказује форму за рад са резервацијом глумца. Учитана је листа глумаца.

**Основни сценарио СК**

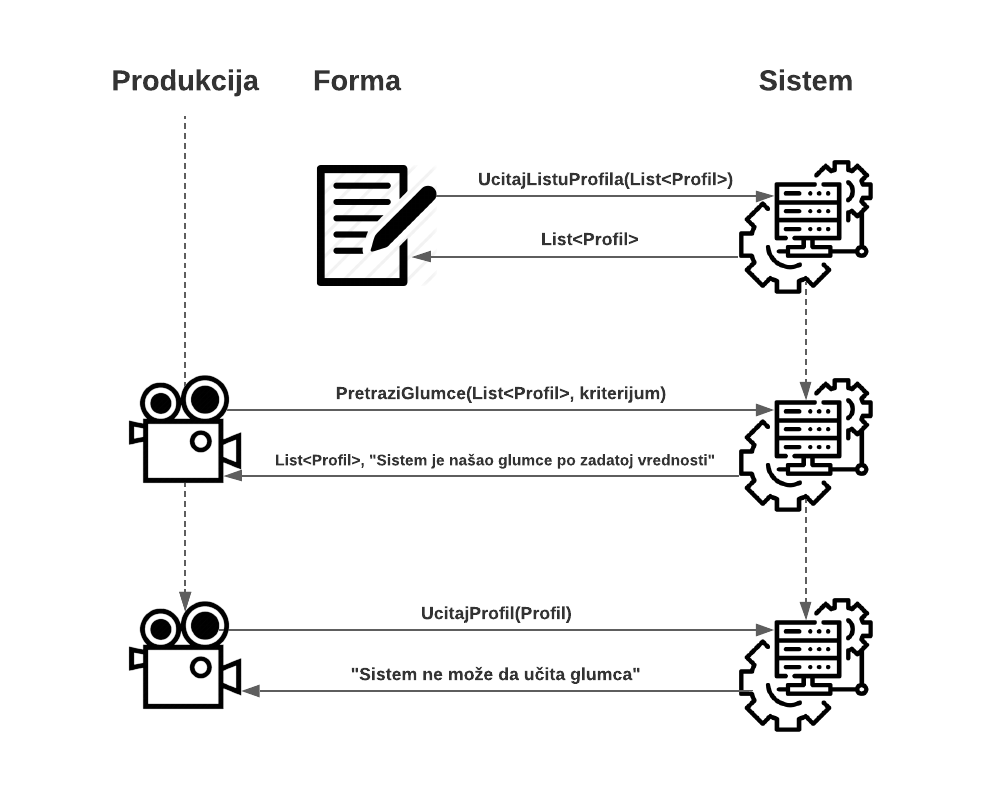
1. Продукција **позива** систем да нађе профиле глумаца по задатој вредности. (АПСО)
2. Систем приказује продукцији профиле глумаца и поруку: “Систем је нашао профиле глумаца по задатој вредности”. (ИА)
3. Продукција **позива** систем да учита профил глумца. (АПСО)
4. Систем приказује продукцији профил глумца и поруку: “Систем је учитао профил глумца”. (ИА)
5. Продукција **позива** систем да креира резервацију глумца. (АПСО)
6. Систем **приказује** продукцији резервацију глумца и поруку: “Систем је креирао резервацију глумца“. (ИА)

**Алтернативна сценарија**

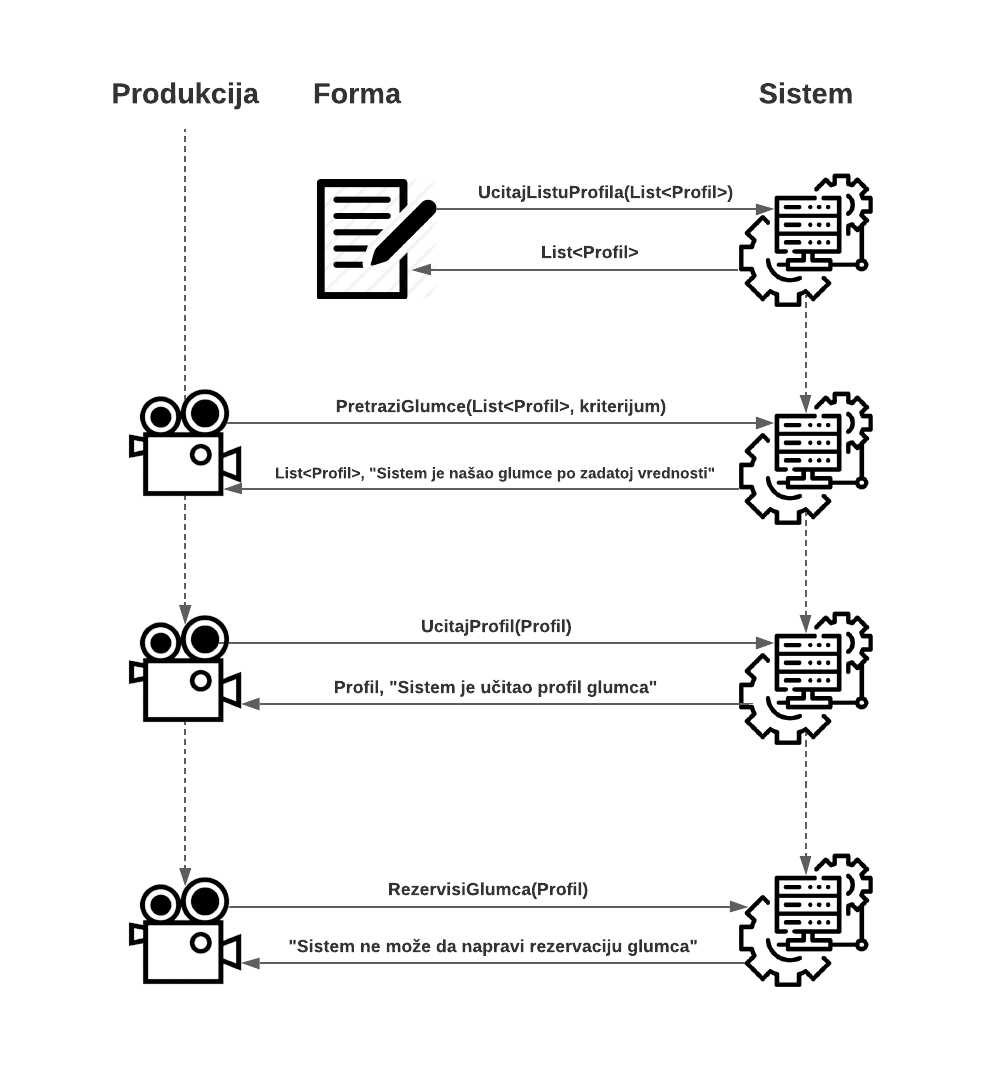
2.1 Уколико систем не може да нађе профиле глумаца по задатој вредности он приказује продукцији поруку: “Систем не може да нађе профиле глумаца по задатој вредности”. Прекида се извршење сценарија. (ИА)



4.1 Уколико систем не може да учита профиле глумаца он приказује продукцији поруку: “Систем не може да учита профиле глумаца”. (ИА)



* 1. Уколико систем не може да креира резервацију глумца он приказује продукцији поруку: “Систем не може да креира резервацију глумца”.



Са наведених секвенцних дијаграма уочавају се 3 системске операције:

1. Signal **UcitajListuProfila(List<Profil);**
2. Signal **PretraziGlumce(List<Profil>, kriterijum);**
3. Signal **UcitajProfil(Profil);**
4. Signal **RezervisiGlumca(Profil).**

На основу анализе сценарија добијено је 13 системских операција:

1. Signal **KreirajProfil();**
2. Signal **ZapamtiProfil(Profil);**
3. Signal **UcitajProfil(Profil);**
4. Signal **ObrisiProfil(Profil);**
5. Signal **KreirajNalog();**
6. Signal **ZapamtiNalog(Nalog);**
7. Signal **UcitajNalog(Nalog);**
8. Signal **ObrisiNalog(Nalog);**
9. Signal **KreirajProjekat();**
10. Signal **ZapamtiProjekat(Projekat);**
11. Signal **UcitajListuProfila(List<Profil);**
12. Signal **RezervisiGlumca(Profil).**

## Понашање софтверског система – Дефинисање уговора о системским операцијама

Уговор УГ1: **KreirajProfil()** Signal**;**

Веза са СК: СК1

Предуслови: Вредносна и структурна ограничења над објектом **Профил** морају да буду задовољена.

Постуслови: Профил је креиран.

Уговор УГ2: **ZapamtiProfil(Profil)** Signal**;**

Веза са СК: СК1, СК2

Предуслови: Вредносна и структурна ограничења над објектом **Профил** морају да буду задовољена.

Постуслови: Подаци о профилу глумца су запамћени.

Уговор УГ3: **UcitajProfil(Profil)** Signal**;**

Веза са СК: СК2, СК3, СК8, СК9

Предуслови:

Постуслови:

Уговор УГ4: **ObrisiProfil(Profil)** Signal**;**

Веза са СК: СК3

Предуслови:

Постуслови: Профил је обрисан.

Уговор УГ5: **KreirajNalog(Nalog)**Signal**;**

Веза са СК: СК4

Предуслови: Вредносна и структурна ограничења над објектом **Налог** морају да буду задовољена.

Постуслови: Налог је креиран.

Уговор УГ6: **ZapamtiNalog(Nalog)** Signal**;**

Веза са СК: СК4, СК5

Предуслови: Вредносна и структурна ограничења над објектом **Налог** морају да буду задовољена.

Постуслови: Подаци о налогу су запамћени.

Уговор УГ7: **UcitajNalog(Nalog)** Signal**;**

Веза са СК: СК5, СК6

Предуслови:

Постуслови:

Уговор УГ8: **ObrisiNalog(Nalog)** Signal**;**

Веза са СК: СК6

Предуслови:

Постуслови: Налог је обрисан.

Уговор УГ9: **KreirajProjekat(Projekat)** Signal**;**

Веза са СК: СК7

Предуслови: Вредносна и структурна ограничења над објектом **Пројекат** морају да буду задовољена.

Постуслови: Пројекат је креиран.

Уговор УГ10: **ZapamtiProjekat(Projekat)** Signal**;**

Веза са СК: СК7

Предуслови: Вредносна и структурна ограничења над објектом **Пројекат** морају да буду задовољена.

Постуслови: Подаци о пројекту су запамћени.

Уговор УГ11: **UcitajListuProfila(List<Profil)** Signal**;**

Веза са СК: СК8, СК9

Предуслови:

Постуслови:

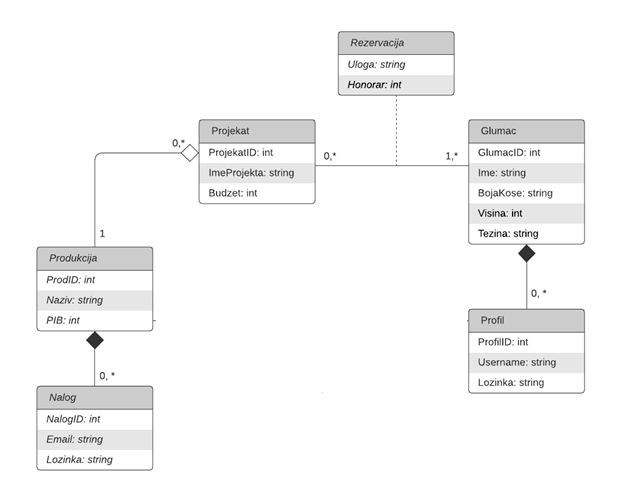
Уговор УГ13: **RezervisiGlumca(Profil)** Signal**;**

Веза са СК: СК9

Предуслови:

Постуслови:

## Структура софтверског система – Концептуални (доменски) модел



## Структура софтверског система – релациони модел

Glumac(GlumacID, Ime, BojaKose, Visina, Tezina)

Profil(*GlumacID*, ProfilID, Username, Lozinka)

Produkcija(ProdID, Naziv, PIB)

Nalog(*ProdID*, NalogID, Email, Lozinka)

Projekat(ProjekatID, ImeProjekta, Budzet, *ProdID*)

Rezervacija(*ProjekatID*, *GlumacID*, Uloga, Honorar)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Tabela Produkcija | | 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 Projekat, Nalog, Favoriti  DELETE RESTRICTED Projekat, Nalog, Favoriti |
| ProdID | Integer | not null and >0 |  |  |
| Naziv | String | not null |  |  |
| PIB | Integer | not null |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Tabela Nalog | | Prosto vrednosno ograničenje | | Složeno vrednosno ograničenje | | Strukturno ograničenje |
| Atributi | Ime | Tip atributa | Vrednost atributa | Međuzavisnost atributa jedne tabele | Međuzavisnost atributa više tabela | INSERT RESTRICTED Produkcija  UPDATE RESTRICTED Produkcija  DELETE / |
| ProdID | Integer | not null and >0 |  |  |
| NalogID | String | not null and >0 |  |  |
| Emali | String | not null |  |  |
| Lozinka | String | not null |  |  |

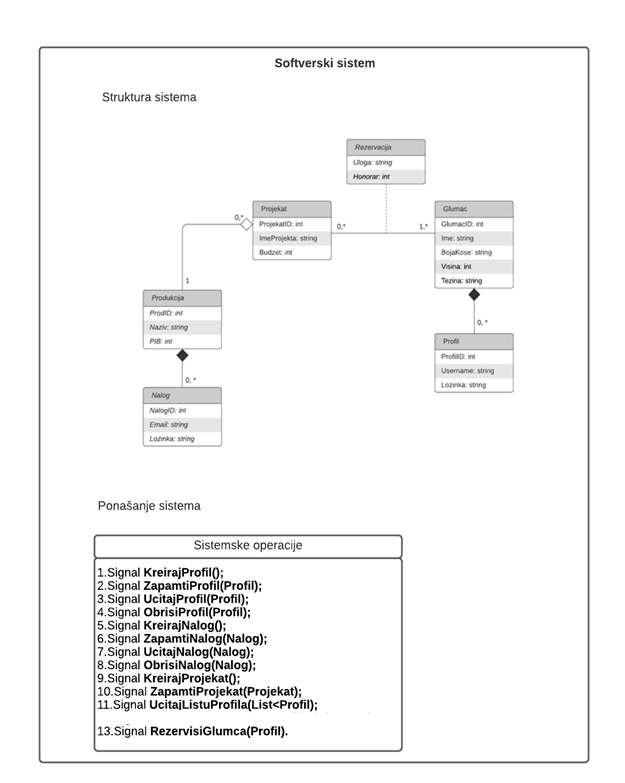
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Tabela Glumac | | Prosto vrednosno ograničenje | | Složeno vrednosno ograničenje | | Strukturno ograničenje |
| Atributi | Ime | Tip atributa | Vrednost atributa | Međuzavisnost atributa jedne tabele | Međuzavisnost atributa više tabela | INSERT /  UPDATE CASCADES Profil, Rezervacija  DELETE RESTRICTED Profil, Rezervacija |
| GlumacID | Integer | not null and >0 |  |  |
| Ime | String | not null |  |  |
| BojaKose | String | not null |  |  |
| Visina | Integer | not null |  |  |
| Tezina | Integer | not null |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Tabela Profil | | Prosto vrednosno ograničenje | | Složeno vrednosno ograničenje | | Strukturno ograničenje |
| Atributi | Ime | Tip atributa | Vrednost atributa | Međuzavisnost atributa jedne tabele | Međuzavisnost atributa više tabela | INSERT RESTRICTED Glumac  UPDATE RESTRICTED Glumac  UPDATE CASCADES  Favoriti  DELETE RESTRICTED Favoriti |
| GlumacID | Integer | not null and >0 |  |  |
| ProfilID | String | not null and >0 |  |  |
| Username | String | not null |  |  |
| Lozinka | String | not null |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Tabela Projekat | | Prosto vrednosno ograničenje | | Složeno vrednosno ograničenje | | Strukturno ograničenje |
| Atributi | Ime | Tip atributa | Vrednost atributa | Međuzavisnost atributa jedne tabele | Međuzavisnost atributa više tabela | INSERT RESTRICTED Produkcija  UPDATE RESTRICTED Produkcija  UPDATE CASCADES  Rezervacija  DELETE  RESTRICTED  Rezervacija |
| ProjekatID | Integer | not null and >0 |  |  |
| ImeProjekta | String | not null |  |  |
| Budzet | Integer | not null |  |  |
| ProdID | Integer | not null |  |  |

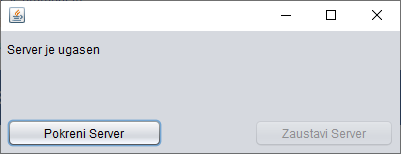
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Tabela Rezervacija | | Prosto vrednosno ograničenje | | Složeno vrednosno ograničenje | | Strukturno ograničenje |
| Atributi | Ime | Tip atributa | Vrednost atributa | Međuzavisnost atributa jedne tabele | Međuzavisnost atributa više tabela | INSERT RESTRICTED Glumac, Projekat  UPDATE RESTRICTED Glumac, Projekat  DELETE / |
| GlumacID | Integer | not null and >0 |  |  |
| ProjekatID | Integer | not null and >0 |  |  |
| Uloga | String | not null |  |  |
| Honorar | Integer | not null |  |  |

Као резултат анализе сценарија СК и прављења концептуалног модела добија се логичка структура и понашање софтверског система:

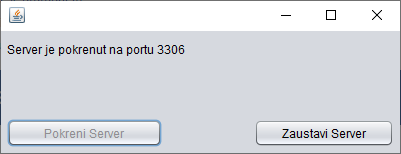


# Пројектовање

На самом почетку потребно је да покренемо сервер који ће обрађивати наше захтеве и радити са базом података. Када се покрене апликација појављује се јава прозор на коме можемо притиснути дугме Покрени Сервер:



Притиском на дугме прозор нам избацује поруку о покретању сервера.



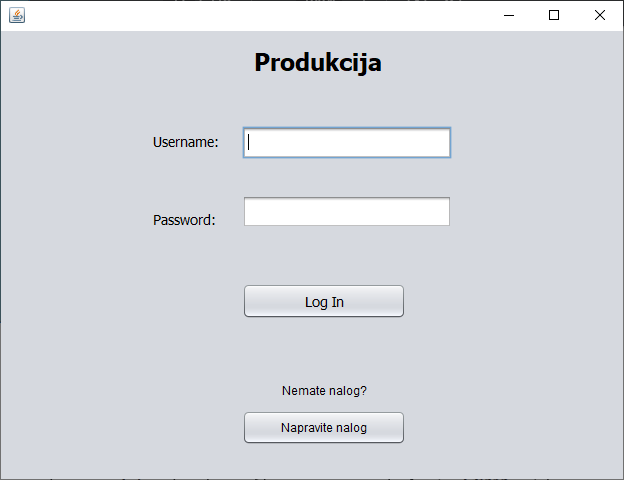
Када се покрене сервер дугме за покретање се онемогућује, да се ни би 2 пута покренуо сервер на истом порту.

Сада можемо да покренемо клијентску апликацију. На њој нам се приказује почетна форма, на којој бирамо коју улогу имамо као корисник. Тачније да ли представљамо продукцијску кућу и тражимо људе или представљамо глумце који траже посао.

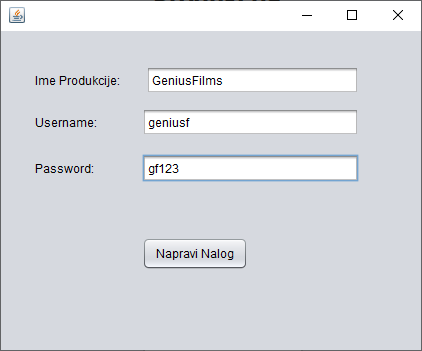


У овом тренутку се клијентска апликација дели на 2 целине. Прва целина се односи на продукције које траже глумце за свој пројекат. Оне имају одвојене форме и цео интерфејс им изгледа другачије. Њихова главна улога је да након уношења својих креденцијала, могу да креирају пројекте и везују глумце за те пројекте, као и да управљају својим налозима.

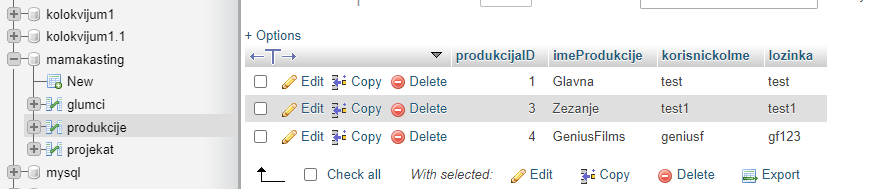
Притиском на дугме тражи глумце долазимо на Логин форму за продукције:



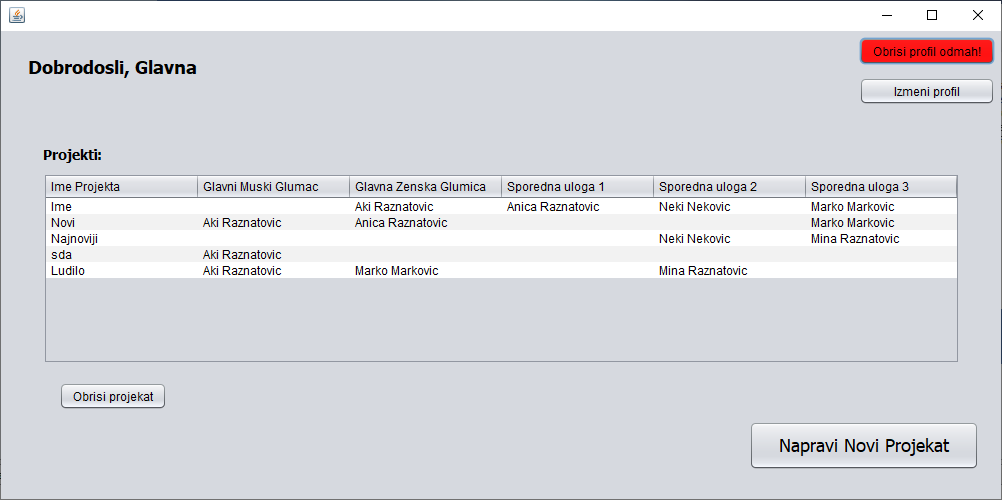
Овде уносимо своје креденцијале ако имамо налог, а ако немамо креирамо нови. Показаћемо на следећој слици форму за креирање налога:



Када унесемо податке, контролер шаље те податке серверу да их унесе у базу.



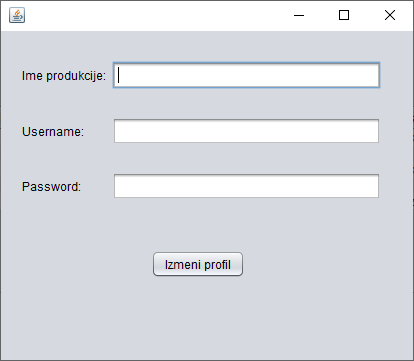
Затим се са истим подацима пријављујемо и појављује нам се главна форма за продукцију:



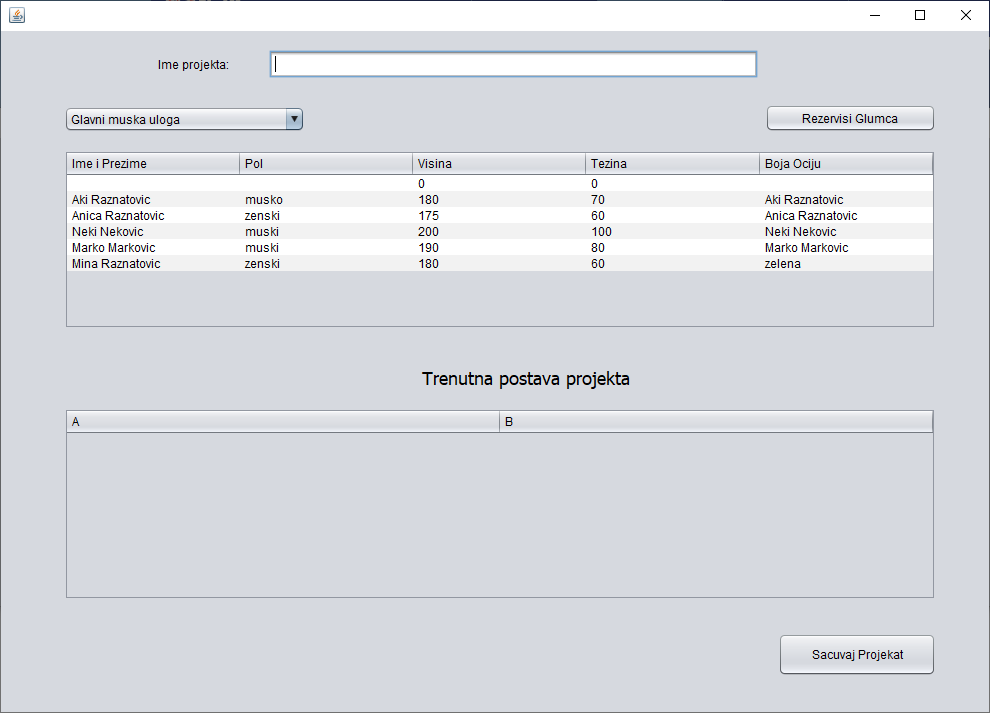
Ту имамо неколико могућности. У централној табели нам се приказују постојећи пројекти везани за нашу продукцију.

Овде притиском на дугме обруши пројекат можемо врло једноставно да уклонимо неки пројекат.Притиском на дугме обриши профил одмах, уклањанмо постојећи профил и апликација се гаси.

Притисом на дугме измени профил, добијамо прилику да мењамо корисничко име, лозинку или име продукције:



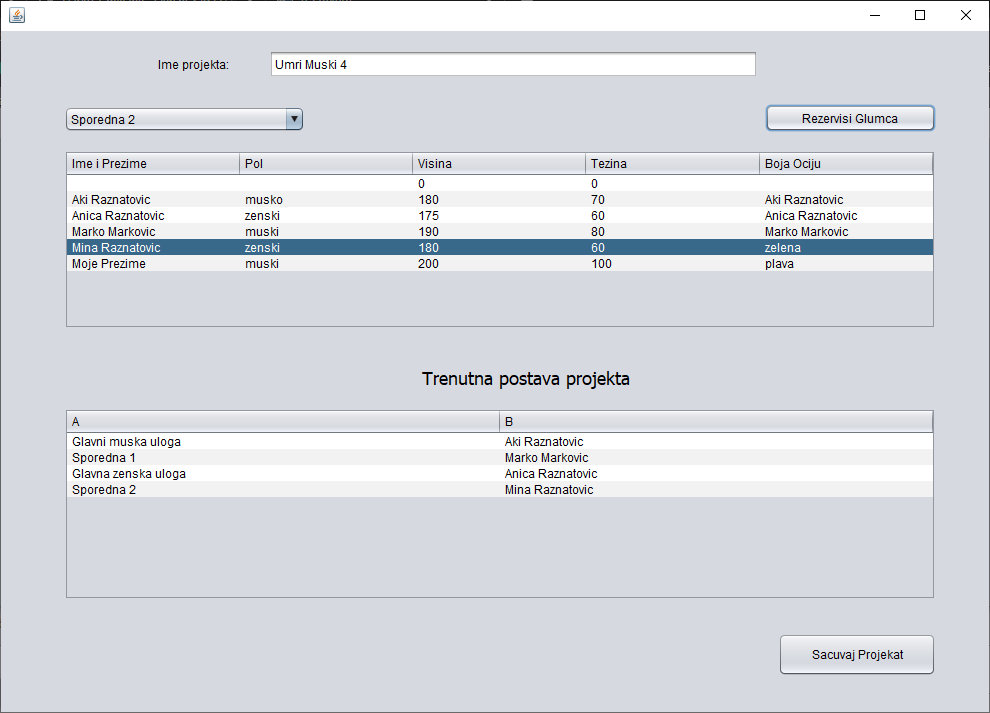
И на крају, дугме које служи да се направи нови пројекат. Притиском на њега отвара се нова форма која се састоји од неколико целина:



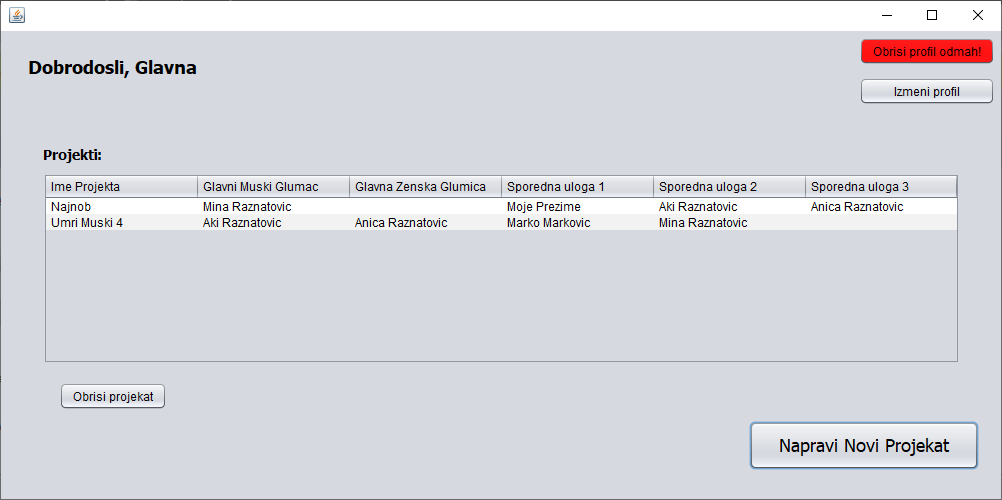
У тексуталном пољу Име Пројекта уносимо како желимо да нам се зове пројекат.

Затим из падајућег менија бирамо за коју улогу желимо да одредимо глумца.

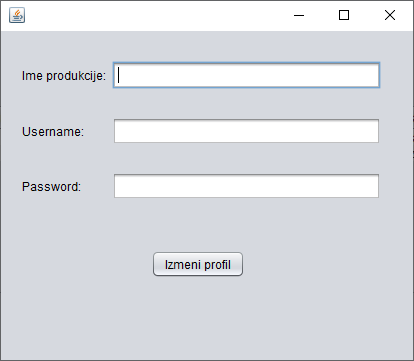
Када одлучимо притиском на дугме Резервиши глумца, везујемо на нашем пројекту одређеног глумца из базе глумаца.



Притиском на дугме Сачувај Пројекат, овај пројекат се чува и додаје се у листу постојећих пројеката продукције.



Поред ових фунционалности можемо да обришемо профил и да га изменимо позивом одговарајућих дугмића на форми. Конкретно да бисмо изменили податке о профилу потребно је да притиснемо дугме Измени профил, након чега нам се отвара форма:



Након уноса жељених информација, притиском на дугме измени профил, чувају се унесене промене.

Притиском на дугме Обиши профил одмах, апликација се гаци, а налог продукције се брише из базе.

То представља неке основне функционалности везане за продукције.

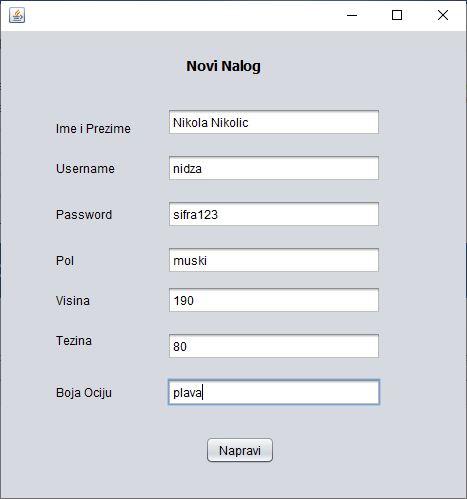
Сада ћемо се осврнути на интефејс за глумце:



Притиском на дугме Тражим посао, отвара се форма за Логин глумаца, слично као малопре.

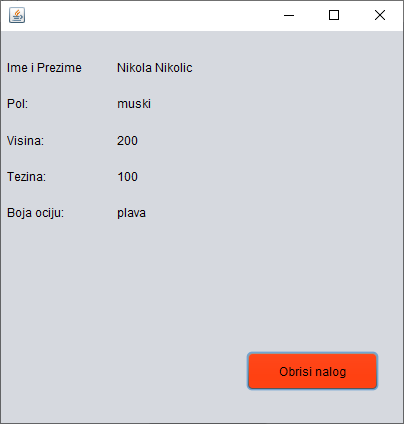


Уколико је особа нова на овој платформи може једноставно направити налог кликом на дуге Направите налог након чега се отвара нова форма:



Након уноса података, они се чувају притиском на дугме Направи. Сада можемо да се пријавимо са овим креденцијалима.

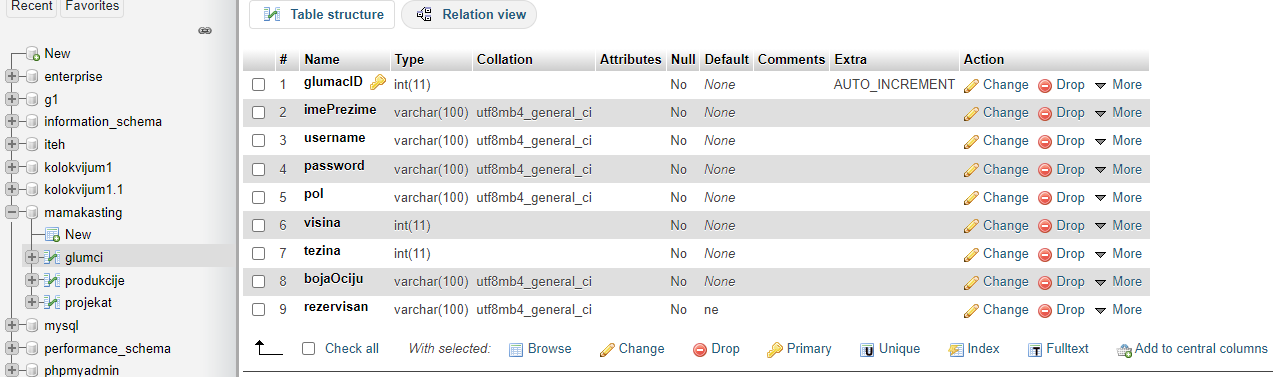
Након што се пријавимо појављује нам се форма на којој се налазе наше информације и могућност да обришемо профил:



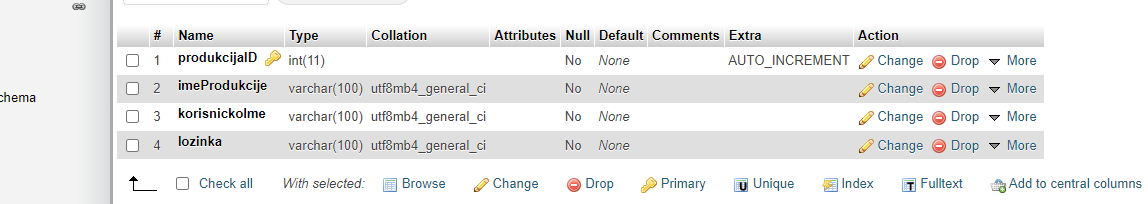
Са направљеним налогом за глумце, продукције могу сада да виде наше податке и да нас букирају за пројекте.

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.

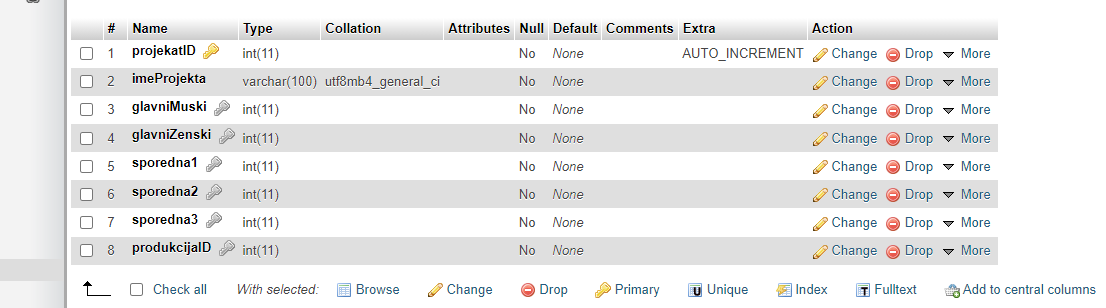
**Складиште глумци:**



**Складиште продукције:**



**Складиште пројекти:**



# Имплементација

Софтверски систем је развијан у програмском језику “Java”. Систем је пројектован као клијентска апликација. Као систем за управљање базом података коришћен je MySQL, док је развојно окружење “NetBeans IDE 8.2”. За рад са сервером се користио XAMPP стек. На основу архитектуре софтверског система добијене су следеће софтверске класе:

(Због боље прегледности и тачности следећи приказ ће бити на латиничном писму)

На клијентској страни структура изгледа овако:

1. Connection

* Session

1. Controller

* ClientController

1. Forme.Glumac

* GlavnaGlumac
* LoginG
* NapraviGlumca

1. Forme.Oba

* Pocetna

1. Forme.Produkcija

* CreateP
* GlavnaP
* IzmenaP
* LoginP
* ProjekatForma

1. Modeli

* ModelFinal
* ModelProjekti
* ModelRezervacija

Клијентска страна се састоји од следећих пакета и класа:

1. Controller

* ServerController

1. Db

* DBBroker

1. Forme

* ServerForm

1. So

* AbstractSO

1. So.glumac

* SoCreateGlumac
* SoDeleteGlumac
* SoGetAllGlumci

1. So.produkcija

* SoCreateProdukcija
* SoDeleteProdukcija
* SoGetAllProdukcije
* SoUpdateProdukcija

1. So.projekat

* SoCreateProjekat
* SoDeleteProjekat
* SoGetAllProjekat

1. Thread

* ThreadClient
* ThreadServer

Заједничка апликација изгледа овако:

1. Domen

* AbstractDomainObject
* Glumac
* Izabrani
* Produkcija
* Projekat

1. Transfer

* Operation
* Request
* Response

# Тестирање

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

# Литература

1. Синиша Влајић, Пројектовање софтвера (Скрипта), Београд 2015.
2. Проф.др. Синиша Влајић, др. Душан Савић, др. Илија Антовић, мр. Војислав Станојевић, дипл.инг. Милош Милић, Пројектовање софтвера – Напредне Јава технологије, Златни пресек, Београд 2008.
3. Java programming, Wikipedia, <https://sh.wikipedia.org/wiki/Java_(programski_jezik)>
4. Stack Overflow форум, <https://stackoverflow.com/>