

УНИВЕРЗИТЕТ У БЕОГРАДУ
ФАКУЛТЕТ ОРГАНИЗАЦИОНИХ НАУКА

ЗАВРШНИ РАД

Софтверски систем за праћење рада
вртића у ЈАВА окружењу

Ментор

Др Синиша Влајић,
Редовни професор

Студент

Теодора Бошковић, 2020/0488

Београд, 2024. године

Садржај

| | | |
|-------|---|----|
| 1. | Увод..... | 9 |
| 2. | Упрошћена Ларманова метода развоја софтвера..... | 10 |
| 2.1 | Прикупљање корисничких захтева | 10 |
| 2.2 | Анализа | 11 |
| | Понашање софтверског система | 11 |
| | Структура софтверског система..... | 12 |
| 2.3 | Пројектовање | 13 |
| 2.4 | Имплементација и тестирање | 14 |
| 3. | Јава технологије..... | 14 |
| 3.1. | Концепти објектно оријентисаног програмирања | 14 |
| | Класе и методе | 14 |
| | Типови података | 15 |
| | Наслеђивање..... | 16 |
| | Апстракције | 18 |
| | Изузеци..... | 20 |
| 3.2. | Графички интерфејс у Јави | 21 |
| | Swing компоненте..... | 21 |
| | Догађаји..... | 23 |
| 3.3 | Нити | 25 |
| 3.4 | Рад у мрежи..... | 27 |
| | Сокети..... | 27 |
| 3.5 | Рад са базом података..... | 32 |
| 4. | Студијски пример | 34 |
| 4.1 | Прикупљање корисничких захтева | 34 |
| 4.1.1 | Вербални опис система | 34 |
| 4.1.2 | Случајеви коришћења..... | 34 |
| | СК1: Случај коришћења – Креирање детета..... | 35 |
| | СК2: Случај коришћења – Претраживање детета..... | 36 |
| | СК3: Случај коришћења – Креирање родитеља/старатеља | 37 |
| | СК4: Случај коришћења – Креирање запосленог | 38 |
| | СК5: Случај коришћења – Креирање изборног програма | 39 |
| | СК6: Случај коришћења –Измена изборног програма | 40 |
| | СК7: Случај коришћења –Брисање изборног програма | 42 |
| | СК8: Случај коришћења –Креирање похађања(сложен СК) | 43 |
| | СК9: Случај коришћења –Измена похађања(сложен СК) | 44 |

| | |
|--|-----|
| СК10:Случај коришћења – Претраживање похађања | 45 |
| 4.2 Фаза анализе | 46 |
| 4.2.1 Понашање софтверског система- Системски дијаграм секвенци | 46 |
| ДС1: Дијаграм секвенци случаја коришћења - Креирање детета | 47 |
| ДС2: Дијаграм секвенци случаја коришћења - Претраживање детета | 48 |
| ДС3: Дијаграм секвенци случаја коришћења - Креирање родитеља/старатеља..... | 50 |
| ДС4: Дијаграм секвенци случаја коришћења - Креирање запосленог..... | 51 |
| ДС5: Дијаграм секвенци случаја коришћења - Креирање изборног програма | 53 |
| ДС6: Дијаграм секвенци случаја коришћења - Измена изборног програма..... | 54 |
| ДС7: Дијаграм секвенци случаја коришћења - Брисање изборног програма..... | 57 |
| ДС8: Дијаграм секвенци случаја коришћења - Креирање похађања | 60 |
| ДС9: Дијаграм секвенци случаја коришћења - Измена похађања | 62 |
| ДС10: Дијаграм секвенци случаја коришћења - Претраживање похађања | 66 |
| 4.2.2 Понашање софтверског система – Дефинисање уговора о системским операцијама.. | 69 |
| 4.2.3 Структура софтверског система – Концептуални (доменски) модел..... | 71 |
| 4.2.4 Структура софтверског система – Релациони модел | 71 |
| 4.3 Пројектовање | 76 |
| 4.3.1 Архитектура софтверског система..... | 76 |
| 4.3.2 Пројектовање корисничког интерфејса | 76 |
| 4.3.2.1 Пројектовање екранских форми | 77 |
| СК1: Случај коришћења – Креирање детета..... | 79 |
| СК2: Случај коришћења – Претраживање детета..... | 83 |
| СК3: Случај коришћења – Креирање родитеља/старатеља | 88 |
| СК4: Случај коришћења – Креирање запосленог | 92 |
| СК6: Случај коришћења –Измена изборног програма | 98 |
| СК7: Случај коришћења –Брисање изборног програма | 104 |
| СК8: Случај коришћења –Креирање похађања(сложен СК) | 110 |
| СК9: Случај коришћења –Измена похађања(сложен СК) | 113 |
| СК10:Случај коришћења – Претраживање похађања | 119 |
| 4.3.2.2 Пројектовање контролера корисничког интерфејса | 124 |
| 4.3.3 Пројектовање апликационе логике | 124 |
| 4.3.3.1 Контролер апликационе логике | 124 |
| 4.3.3.2 Пословна логика..... | 125 |
| Пројектовање структуре софтверског система (доменске класе) | 132 |
| 4.3.4 Пројектовање складишта података | 135 |
| 4.4 Фаза имплементације | 139 |

| | |
|---------------------|-----|
| 4.5 Тестирање | 143 |
| 5. Закључак | 159 |
| 6. Литература | 160 |

Попис слика

| | |
|---|----|
| Слика 1. Развој софтверског система по упрошћеној Пармановој методи развоја софтвера[2] | 10 |
| Слика 2. Општа структура модела случаја коришћења..... | 11 |
| Слика 3. Структура комплекснијег модела случаја коришћења | 11 |
| Слика 4. Пример секвенцног дијаграма за случај коришћења Креирање запосленог | 12 |
| Слика 5. Пример концептуалног модела софтверског система | 12 |
| Слика 6. Тронивојска архитектура[2]..... | 13 |
| Слика 7. Структура класе Тенисер..... | 15 |
| Слика 8. Креирање објеката класе Тенисер | 15 |
| Слика 9. Класификација типова података у Јави[5]..... | 16 |
| Слика 10. Наслеђивање..... | 17 |
| Слика 11. Структура родитељске класе Спортиста | 17 |
| Слика 12. Структура подкласе Тенисер | 17 |
| Слика 13. Пример апстрактне класе | 18 |
| Слика 14. Пример имплементације апстрактне методе исписи у класи Тенисер | 18 |
| Слика 15. Пример имплементације апстрактне методе исписи у класи Кошаркаш..... | 18 |
| Слика 16. Пример интерфејса Спортиста | 19 |
| Слика 17. Пример интерфејса ТурнирИграч..... | 19 |
| Слика 18. Пример имплементације интерфејса Спортиста и ТурнирИграч | 19 |
| Слика 19. Хијерархија изузетака у Јави[8]..... | 20 |
| Слика 20. Пример обраде грешке унутар try-catch блока..... | 20 |
| Слика 21. Пример обраде грешке помоћу throws у потпису методе..... | 21 |
| Слика 22.Основне Swing текстуалне компоненте | 22 |
| Слика 23. Основне Swing компоненте дугмића..... | 23 |
| Слика 24.Обрада догађаја кликом на дугме..... | 24 |
| Слика 25. Резултат обраде догађаја кликом на дугме | 24 |
| Слика 26. Код графичког интерфејса..... | 24 |
| Слика 27. Креирање нити помоћу класе Runnable | 25 |
| Слика 28. Креирање нити помоћу класе Thread | 25 |
| Слика 29. Графички приказ Deadlock проблема[9]..... | 26 |
| Слика 30. Графички приказ синхронизације нити[9] | 26 |
| Слика 31. Приказ кода за подизање серверског сокета | 27 |
| Слика 32. Приказ кода за успостављање везе са серверским сокетом | 28 |
| Слика 33. Приказ адресе сокета[6] | 28 |
| Слика 34. Приказ улазних и излазних токова података[5] | 28 |
| Слика 35. Приказ кода серверског програма..... | 29 |
| Слика 36. Приказ кода клијентског програма | 30 |

| | |
|--|----|
| Слика 37. Основни кораци при сокет програмирању[5]..... | 31 |
| Слика 38. Веза Јава програма са СУБП[3] | 32 |
| Слика 39. Кораци за повезивање са базом података и извршавање SQL упита у Јава апликацији[7] | 33 |
| Слика 40. Повезивање са базом и извршавање SELECT наредбе..... | 33 |
| Слика 41. Модел случајева коришћења | 34 |
| Слика 42. ДС1 - Основни сценарио..... | 47 |
| Слика 43. ДС1 - Алтернативни сценарио 1..... | 47 |
| Слика 44. ДС2 - Основни сценарио..... | 48 |
| Слика 45. ДС2 - Алтернативни сценарио 1..... | 48 |
| Слика 46. ДС2 - Алтернативни сценарио 2..... | 49 |
| Слика 47. ДС3 - Основни сценарио..... | 50 |
| Слика 48. ДС3 - Алтернативни сценарио 1..... | 50 |
| Слика 49. ДС4 - Основни сценарио..... | 51 |
| Слика 50. ДС4 - Алтернативни сценарио 1..... | 52 |
| Слика 51. ДС5 - Основни сценарио..... | 53 |
| Слика 52. ДС5 - Алтернативни сценарио 1..... | 53 |
| Слика 53. ДС6 - Основни сценарио..... | 54 |
| Слика 54. ДС6 - Алтернативни сценарио 1..... | 55 |
| Слика 55. ДС6 - Алтернативни сценарио 2..... | 55 |
| Слика 56. ДС6 - Алтернативни сценарио 3..... | 56 |
| Слика 57. ДС7 - Основни сценарио..... | 57 |
| Слика 58. ДС7 - Алтернативни сценарио 1..... | 58 |
| Слика 59. ДС7 - Алтернативни сценарио 2..... | 58 |
| Слика 60. ДС7 - Алтернативни сценарио 3..... | 59 |
| Слика 61. ДС8 - Основни сценарио..... | 60 |
| Слика 62. ДС8 - Алтернативни сценарио 1..... | 61 |
| Слика 63. ДС9 - Основни сценарио..... | 62 |
| Слика 64. ДС9 - Алтернативни сценарио 1..... | 63 |
| Слика 65. ДС9 - Алтернативни сценарио 2..... | 64 |
| Слика 66. ДС9 - Алтернативни сценарио 3..... | 65 |
| Слика 67. ДС10 - Основни сценарио..... | 66 |
| Слика 68. ДС10 - Алтернативни сценарио 1..... | 66 |
| Слика 69. ДС10 - Алтернативни сценарио 2..... | 67 |
| Слика 70. Концептуални модел | 71 |
| Слика 71. Понашање и структура система | 75 |
| Слика 72. Тронивојска архитектура..... | 76 |
| Слика 73. Структура корисничког интерфејса[2]..... | 76 |
| Слика 74. Серверска форма пре покретања | 77 |
| Слика 75. Серверска форма након покретања | 77 |
| Слика 76. Login форма..... | 78 |
| Слика 77. Login форма-порука о успешном пријављивању | 78 |
| Слика 78. Главна клијентска форма | 78 |
| Слика 79. Форма за креирање новог детета..... | 79 |
| Слика 80. Унос података за ново дете | 80 |
| Слика 81. Успешно креирано дете | 81 |

| | |
|--|-----|
| Слика 82. Систем не може да креира дете..... | 82 |
| Слика 83. Форма за претраживање деце..... | 83 |
| Слика 84. Унос вредности за претрагу детета..... | 84 |
| Слика 85. Систем је пронашао дете по задатој вредности | 84 |
| Слика 86. Одабир детета | 85 |
| Слика 87. Систем је учитао одабрано дете | 85 |
| Слика 88. Систем показује податке о одабраном детету | 86 |
| Слика 89. Систем не може да пронађе дете по задатој вредности | 86 |
| Слика 90. Систем не може да учита одабрано дете | 87 |
| Слика 91. Форма за креирање новог детета..... | 88 |
| Слика 92. Унос података у новог родитеља/старатеља | 89 |
| Слика 93. Успешно креиран родитељ/старатељ | 90 |
| Слика 94. Систем не може да креира родитеља/старатеља | 91 |
| Слика 95. Форма за креирање новог запосленог | 92 |
| Слика 96. Унос података за новог запосленог..... | 93 |
| Слика 97. Успешно креиран запослени | 93 |
| Слика 98. Систем не може да креира запосленог..... | 94 |
| Слика 99. Форма за креирање изборног програма | 95 |
| Слика 100. Унос података за нови изборни програм | 96 |
| Слика 101. Успешно креиран изборни програм..... | 96 |
| Слика 102. Систем не може да креира изборни програм | 97 |
| Слика 103. Форма за претраживање изборних програма..... | 98 |
| Слика 104. Унос вредности за претрагу изборних програма..... | 99 |
| Слика 105. Систем је пронашао изборне програме по задатој вредности..... | 99 |
| Слика 106. Одабир изборног програма..... | 100 |
| Слика 107. Систем је учитао одабрани изборни програм | 100 |
| Слика 108. Систем показује податке о одабраном изборном програму | 101 |
| Слика 109. Измена података одабраног изборног програма..... | 101 |
| Слика 110. Систем је замаптио податке о одабраном изборном програму | 102 |
| Слика 111. Систем не може да пронађе изборне програме по задатој вредности | 102 |
| Слика 112. Систем не може да учита одобрани изборни програм..... | 103 |
| Слика 113. Систем не може да запамти одобрани изборни програм | 103 |
| Слика 114. Форма за претраживање изборних програма | 104 |
| Слика 115. Унос вредности за претрагу изборних програма | 105 |
| Слика 116. Систем је пронашао изборне програме по задатој вредности | 105 |
| Слика 117. Одабир изборног програма | 106 |
| Слика 118. Систем је учитао одобрани изборни програм | 106 |
| Слика 119. Систем показује податке о одобраним изборним програмима | 107 |
| Слика 120. Систем је обрисао одобрани изборни програм..... | 107 |
| Слика 121. Систем не може да пронађе изборне програме по задатој вредности | 108 |
| Слика 122. Систем не може да учита одобрани изборни програм | 108 |
| Слика 123. Систем не може да обрише одобрани изборни програм | 109 |
| Слика 124. Форма за креирање новог похађања | 110 |
| Слика 125. Унос података за ново похађање | 111 |

| | |
|--|-----|
| Слика 126. Успешно креирано похађање | 111 |
| Слика 127. Систем не може да креира похађање..... | 112 |
| Слика 128. Форма за претраживање похађања | 113 |
| Слика 129. Унос вредности за претрагу изборних програма..... | 114 |
| Слика 130. Систем је пронашао похађања по задатој вредности | 114 |
| Слика 131. Одабир похађања..... | 115 |
| Слика 132. Систем је учитао одабрано похађање | 115 |
| Слика 133. Систем показује податке о одабраном похађању | 116 |
| Слика 134. Измена података одабраног похађања | 116 |
| Слика 135. Систем је запамтио одабрано похађање..... | 117 |
| Слика 136. Систем не може да пронађе похађање по задатој вредности | 117 |
| Слика 137. Систем не може да учита одабрано похађање | 118 |
| Слика 138. Систем не може да запамти одабрано похађање..... | 118 |
| Слика 139. Форма за претраживање похађања | 119 |
| Слика 140. Унос вредности за претрагу похађања | 120 |
| Слика 141. Систем је пронашао похађања по задатој вредности | 120 |
| Слика 142. Одабир похађања..... | 121 |
| Слика 143. Систем је учитао одабрано похађање | 121 |
| Слика 144. Систем приказује податке о одабраном похађању | 122 |
| Слика 145. Систем не може да пронађе похађања по задатој вредности | 122 |
| Слика 146. Систем не може да учита одабрано похађање | 123 |
| Слика 147. Уговор УГ1..... | 125 |
| Слика 148. Уговор УГ2..... | 125 |
| Слика 149. Уговор УГ3..... | 126 |
| Слика 150. Уговор УГ4..... | 126 |
| Слика 151. Уговор УГ5..... | 127 |
| Слика 152. Уговор УГ6..... | 127 |
| Слика 153. Уговор УГ7..... | 128 |
| Слика 154. Уговор УГ8..... | 128 |
| Слика 155. Уговор УГ9..... | 129 |
| Слика 156. Уговор УГ10..... | 129 |
| Слика 157. Уговор УГ11..... | 130 |
| Слика 158. Уговор УГ12..... | 130 |
| Слика 159. Уговор УГ13..... | 131 |
| Слика 160. Уговор УГ14..... | 131 |
| Слика 161. Класа Корисник..... | 132 |
| Слика 162. Класа Родитељ/старатељ | 132 |
| Слика 163. Класа Изборни програм | 132 |
| Слика 164. Класа Запослени | 133 |
| Слика 165. Класа Дете | 133 |
| Слика 166. Класа Похађање | 133 |
| Слика 167. Класа Статус корисника | 133 |
| Слика 168. Класа Доменски објекат | 134 |
| Слика 169. Класа Захтев | 134 |
| Слика 170. Класа Одговор | 135 |
| Слика 171. Табела Корисник..... | 135 |

| | |
|---|-----|
| Слика 172. Табела Родитељ/старатељ..... | 135 |
| Слика 173. Табела Изборни програм | 136 |
| Слика 174. Табела Запослени | 136 |
| Слика 175. Табела Дете | 136 |
| Слика 176. Табела Похађање..... | 136 |
| Слика 177. Дијаграм класа добијен након пројектовања доменских класа и класе ОпштиДоменскиОбјекат | 137 |
| Слика 178. Архитектура софтверског система..... | 138 |
| Слика 179а. Тестирање системске операције Login | 145 |
| Слика 180а. Тестирање системске операције GetAllAttendances | 148 |
| Слика 181а. Тестирање системске операције EditAndDeleteAttendances | 150 |
| Слика 182а. Тестирање системске операције AddNewAttendance | 156 |

Попис табела

| | |
|---|----|
| Табела 1. Ограничења за табелу Запослени..... | 72 |
| Табела 2. Ограничења за табелу РодитељСтаратељ..... | 72 |
| Табела 3. Ограничења за табелу Дете..... | 73 |
| Табела 4. Ограничења за табелу ИзборниПрограм | 73 |
| Табела 5. Ограничења за табелу Похађање | 74 |
| Табела 6. Ограничења за табелу Корисник | 74 |

1. Увод

У данашњем времену, убрзани технолошки развој и дигитализација захтевају све ефикасније софтверске системе који ће задовољити потребе корисника. Софтверски инжењери су одговорни за креирање ових система, фокусирајући се на њихову функционалност, флексибилност и могућност даљег прилагођавања. Корисници очекују да софтвер који користе буде интуитиван, поуздан и ефикасан, због чега је квалитет производа кључан за њихово задовољство. Стога, развој софтвера захтева добро структурисан процес који осигурава квалитетан крајњи производ, прилагођен потребама корисника.

У овом раду ће бити приказан процес развоја софтверског система за праћење рада вртића у Јава програмском окружењу. На почетку ће бити објашњена упрошћена Ларманова метода развоја софтвера[2], која је донела значајна побољшања у организацији и управљању софтверским пројектима, са фокусом на корисничке потребе и ефикасност система.

Јава се од настанка до данас истакла као један од најчешће коришћених програмских језика за развој софтверских апликација високог квалитета захваљујући својој објектно-оријентисаној структури, платформској независности и аутоматском управљању меморијом. Једноставна синтакса и типска безбедност додатно су допринели њеној популарности[4]. У трећем поглављу, детаљно ћемо обрадити Јава програмски језик и његове могућности у контексту развоја сложених система.

Четврто поглавље ће обухватити развој софтверског решења за праћење рада вртића, водећи се упрошћеном Лармановом методом развоја софтвера која укључује пет фаза: прикупљање корисничких захтева, анализу, пројектовање, имплементацију и тестирање система.[2]

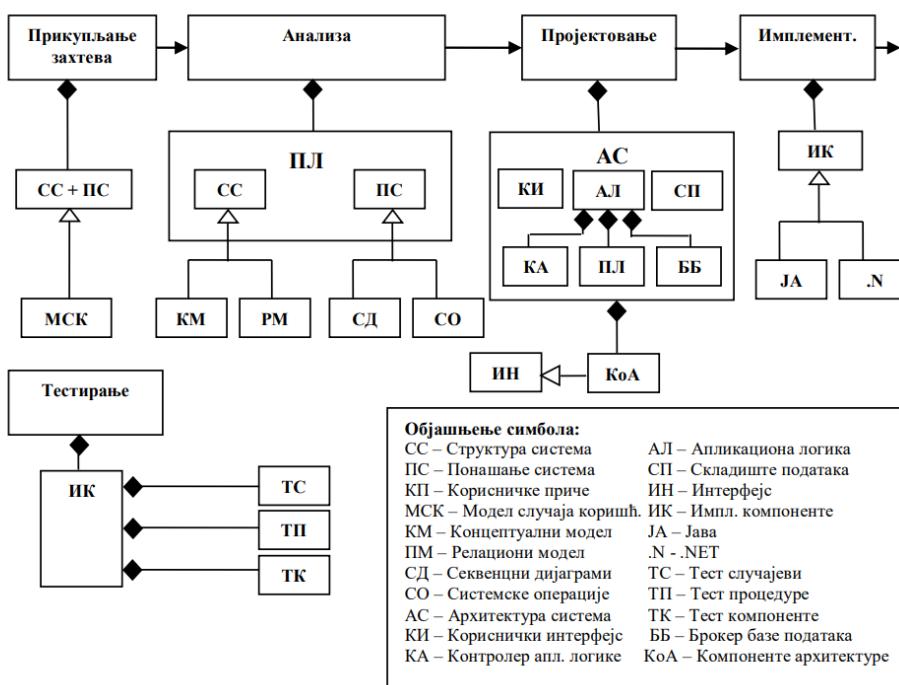
У претпоследњем поглављу ћемо извести закључак на основу резултата рада, а са освртом на изазове и предности са којима смо се сусретали, док ће последње поглавље обухватати коришћену литературу.

2. Упрошћена Ларманова метода развоја софтвера

Према овој поједностављеној методологији развоја софтвера, процес развоја софтверског система обухвата пет фаза:

- Прикупљање корисничких захтева
- Анализа
- Пројектовање
- Имплементација
- Тестирање.[2]

Дијаграм који илуструје ову методологију приказан је на слици 1.



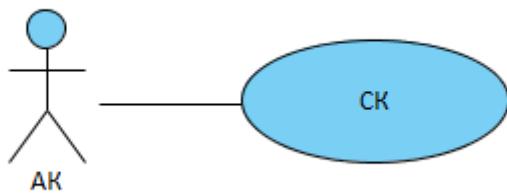
Слика 1. Развој софтверског система по упрощеној Лармановој методи развоја софтвера[2]

2.1 Прикупљање корисничких захтева

Прикупљање корисничких захтева је кључно за развој софтверског система јер омогућава да апликација буде усклађена са потребама и очекивањима корисника, што је од суштинског значаја за њен успех.

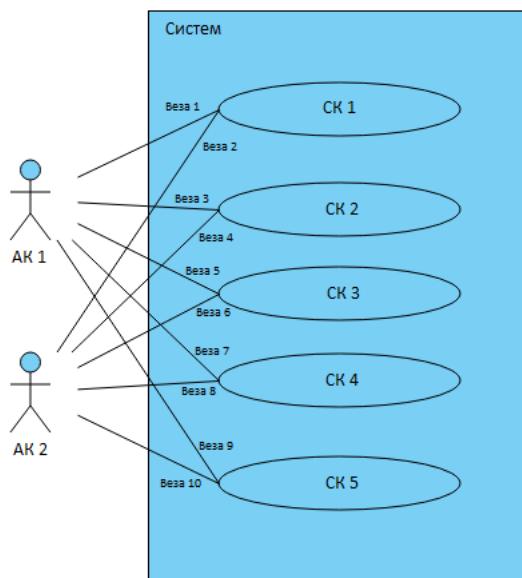
Кориснички захтев је својство или услов који неки систем мора да задовољи. Ови захтеви се описују кроз UML модел случаја коришћења. Структура модела укључује:

- случајеве коришћења (СК),
- акторе који учествују у случајевима коришћења,
- и везе између актора и самих случајева коришћења.[2]



Слика 2. Општа структура модела случаја коришћења

Један модел случаја коришћења може садржати више актора и случајева коришћења где сваки актор може бити повезан са више случајева коришћења, али и сваки случај коришћења може бити повезан са више актора. Стога, модел може имати више веза између актора и случајева коришћења али се свака веза односи на тачно један њихов пар.[2]



Слика 3. Структура комплекснијег модела случаја коришћења

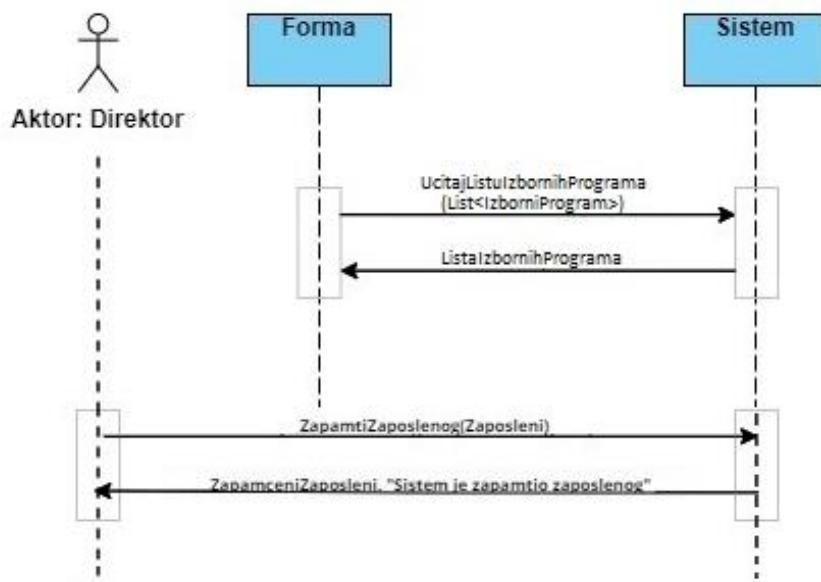
Случај коришћења је описан кроз скуп сценарија који илуструју могуће начине на које актер користи систем. Сваки случај коришћења обухвата један главни сценарио, који представља типичну интеракцију, као и неколико алтернативних сценарија који покривају различите варијације и изузетке у коришћењу система.[2]

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].

Понашање софтверског система

Дијаграм секвенци приказује догађаје у одређеном редоследу, који успостављају интеракцију између актора и софтверског система. За сваки идентификовани сценарио случаја коришћења креира се одговарајући секвенцни дијаграм.[2]

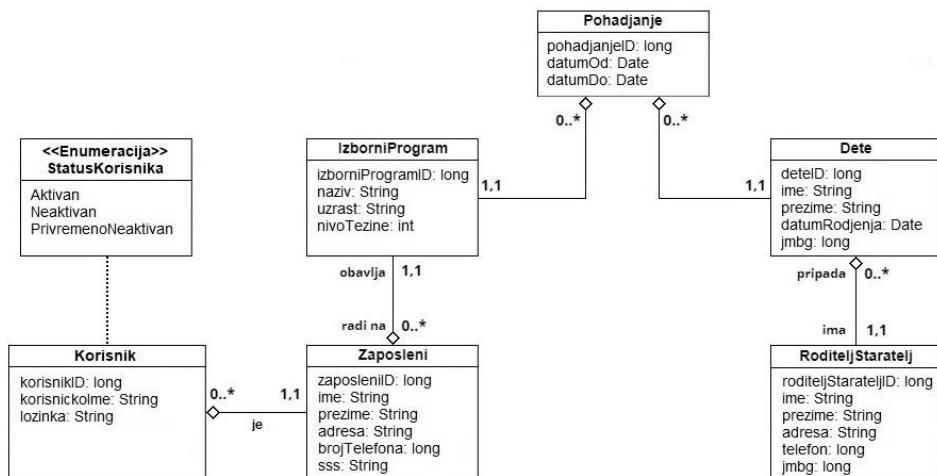


Слика 4. Пример секвенциног дијаграма за случај коришћења Креирање запосленог

Поред секвенцних дијаграма, понашање система описано је и преко системских операција. Свака системска операција(CO) има свој потпис који обухвата назив методе и улазне и/или излазне аргументе уколико постоје. Уговори се креирају за сваку CO и концизно описују шта је улога сваке од њих, без објашњења на који начин ће се то постићи.[2]

Структура софтверског система

Структура софтверског система описује се кроз два основна модела: концептуални и релациони модел. Концептуални модел дефинише концептуалне класе и њихове асоцијације, тј. везе између тих класа, пружајући апстрактан поглед на посматрани систем. Концептуалне класе представљају основне елементе система, док се односи између ових елемената успостављају посредством асоцијационих веза, што омогућава разумевање саме структуре софтверског система и његових функција.



Слика 5. Пример концептуалног модела софтверског система

На основу концептуалног модела изводи се релациони модел, који служи за пројектовање релационе базе података. Релациони модел је основа релационих база података који омогућава организовање података у међусобно повезане табеле.

Релациони модел изведен на основу концептуалног модела са слике 5 изгледа овако:

Zaposleni(zaposleniID, ime, prezime, adresa, brojTelefona, sss, *izborniProgramID*)

RoditeljStaratelj(roditeljStarateljID, ime, prezime, adresa, telefon, jmbg)

Dete(deteljID, ime, prezime, datumRodjenja, jmbg, *roditeljStarateljID*)

IzborniProgram(izborniProgramID, naziv, uzrast, nivoTezine)

Korisnik(korisnikID, korisnickolme, lozinka, *zaposleniID*)

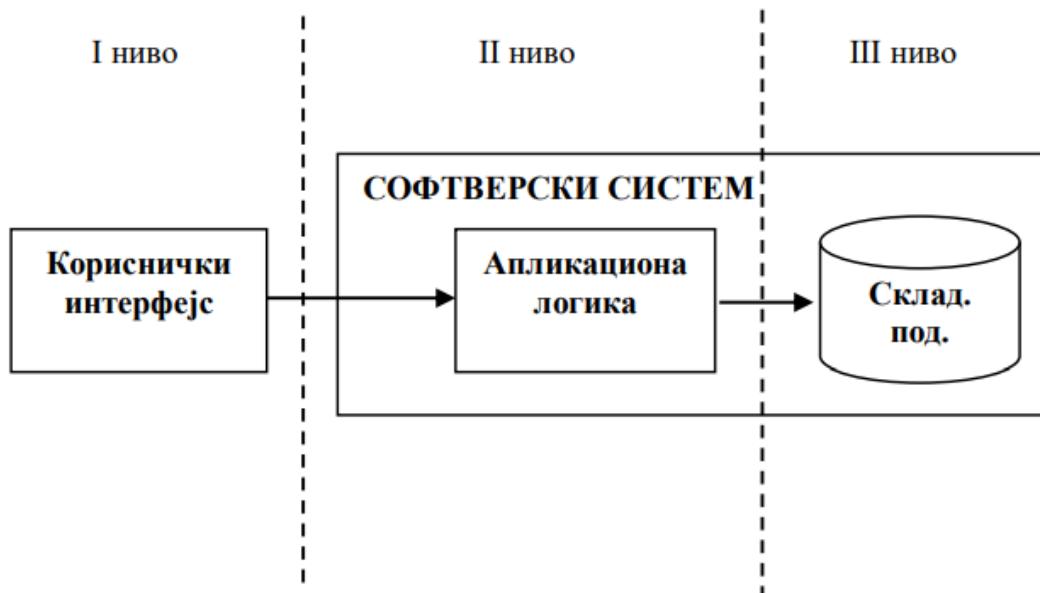
Pohadjanje(pohadjanjeID, datumOd, datumDo, *izborniProgramID*, *deteljID*)

2.3 Пројектовање

У фази пројектовања софтверског система дефинише се његова архитектура, при чему се најчешће користи трослојна архитектура. Овај модел обухвата три основна слоја:

1. **кориснички интерфејс** чије пројектовање обухвата пројектовање екранских форми и контролера корисничког интерфејса
2. **апликациону логику** чије пројектовање обухвата пројектовање контролера апликационе логике, пословне логике и брокера базе података
3. **базу података.[2]**

Кориснички интерфејс представља улазно – излазну репрезентацију софтверског система, апликационна логика описује структуру и понашање софтверског система, док складиште података чува стање атрибуута софтверског система.[2]



Слика 6. Тронивојска архитектура[2]

2.4 Имплементација и тестирање

Након фазе пројектовања, следе фазе имплементације и тестирања софтверског система. Фаза имплементације подразумева креирање софтверског решења у одабраном програмском језику које је у складу са раније дефинисаним корисничким захтевима и које се заснива на структури и понашању утврђеним током фазе анализе. Последња фаза у развоју софтвера подразумева његово тестирање које за циљ има да утврди да ли систем функционише исправно и у складу са захтевима корисника.

3. Јава технологије

Током фазе имплементације нашег софтверског система, технологија која је одабрана због својих значајних предности јесте Јава. Први пут је представљена 1995. године као производ компаније Sun Microsystems. Јава је програмски језик који омогућава развој различитих апликација захваљујући карактеристикама као што су објектно-оријентисано програмирање, једноставна синтакса, и богат екосистем библиотека које убрзавају развој апликација јер нуде готова решења за различите функционалности. Јава омогућава модуларни развој и поновно коришћење кода, као и скривање информација тј. интерне имплементације објекта, уз правило да се свака класа налази у засебној датотеци са екstenзијом .java. Компилација у бајт-код, који Јава виртуелна машина (JVM) интерпретира, омогућава извршавање апликације на различитим платформама, чинећи Јаву независном од оперативног система. Аутоматско управљање меморијом путем "Garbage Collector"-а смањује ризик од грешака повезаних са меморијом. Такође, Јава пружа подршку за конкурентност, што омогућава ефикасно извршавање више нити унутар програма, повећавајући перформансе апликације на вишепроцесорским системима.[4] У наставку рада, детаљно ћемо обрадити концепте објектно-оријентисаног програмирања које Јава подржава, као што су класе, објекти, наслеђивање, интерфејси, апстрактне класе,..., итд.

3.1. Концепти објектно оријентисаног програмирања

Класе и методе

У објектно оријентисаном програмирању (OOP), класа представља шаблон који описује особине и понашање објекта. Класа се састоји од атрибута који дефинишу њене особине и метода које дефинишу њено понашање. Објекат представља појављивање класе са конкретним вредностима атрибута и могућношћу коришћења дефинисаних метода.[4]

На пример, класа Тенисер може имати атрибуте као што су име, презиме, године и ранг ATP и методе као што су испиши тренутни ранг на ATP листи или измени ранг на ATP листи.

```

public class Teniser {
    String ime;
    String prezime;
    int godine;
    int rangATP;

    public void ispisiTrenutniRangNaATPListi() {
        System.out.println("Trenutni rang na ATP listi: "+rangATP);
    }
    public void izmeniRangNaATPListi(int noviRang) {
        rangATP=noviRang;
    }
}

```

Слика 7. Структура класе Тенисер

Приликом креирања објеката, као што су два конкретна тенисера, ови атрибути добијају специфичне вредности, што омогућава да се сваки објекат понаша на начин дефинисан у класи, али са својим јединственим стањем.

```

public class DiplomskiPrimeri {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        Teniser t1=new Teniser();
        t1.godine=37;
        t1.ime="Novak";
        t1.prezime="Djokovic";
        t1.setRangATP(rangATP:1);
        Teniser t2=new Teniser();
        t2.godine=38;
        t2.ime="Rafael";
        t2.prezime="Nadal";
        t2.setRangATP(rangATP:158);
        t2.ispisiTrenutniRangNaATPListi();
    }
}

```

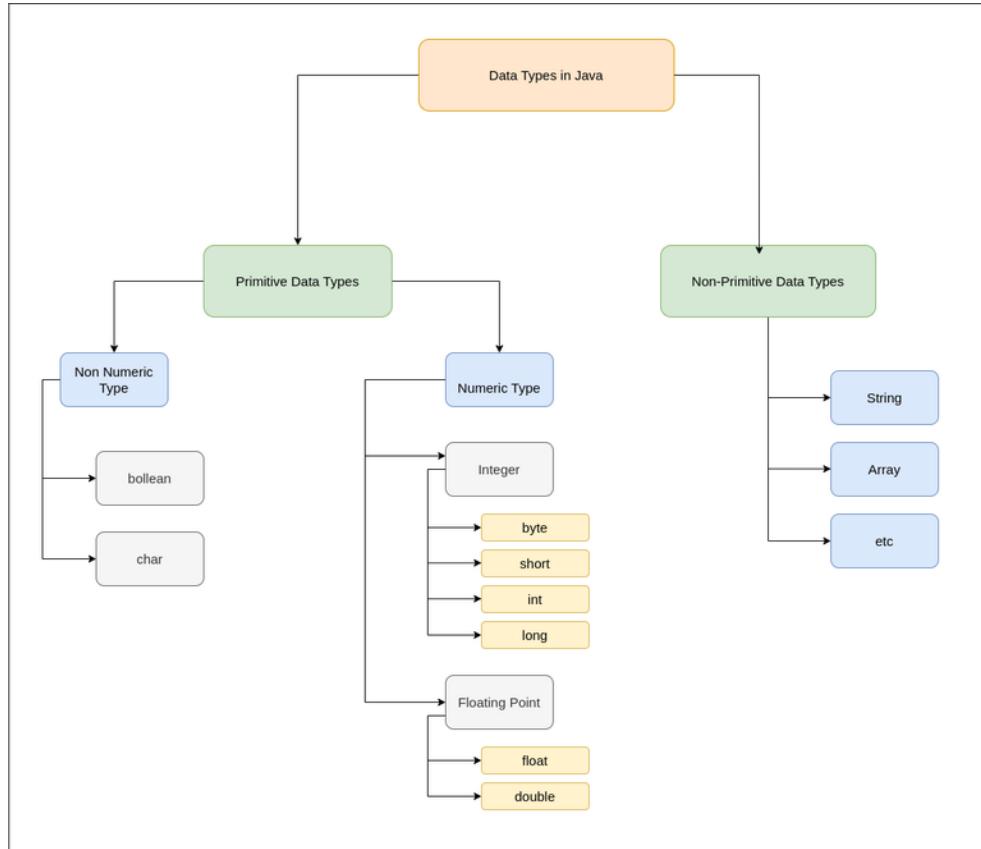
Слика 8. Креирање објекта класе Тенисер

Типови података

Јава је статички типизиран програмски језик, што значи да типови променљивих морају бити декларисани пре него што се користе. Статичка типизација подразумева да се типови података проверавају у време компајлирања, а не у време извршавања програма. Ово омогућава рано откривање грешака у коду, што доприноси већој поузданости и стабилности софтвера. На пример, ако се покуша доделити вредност типа String променљивој која је декларисана као int, компајлер ће пријавити грешку, спречавајући потенцијалне проблеме током извршавања. Конверзија некомпабилних типова се у Јави може извршити имплицитно или експлицитно такозваним кастовањем.

Подаци се могу класификовати на примитивне и референтне типове. Примитивни типови укључују основне типове података као што су *byte*, *short*, *int*, и *long* за представљање целобројних вредности различитих опсега, *float* и *double* за представљање вредности са покретним зарезом, *char* за појединачне *Unicode* карактере, и *boolean* који има две вредности – *true* и *false*.

С друге стране, референтни типови укључују објекте и низове, а најчешће коришћени референтни тип је *String*, који представља низ карактера. Сви референтни типови су имплементирани као класе, што значи да могу имати методе и атрибуте. Класа *Object* је родитељска класа свих објеката у Јави, тако да сваки референтни тип наслеђује њене методе, попут *equals()*, *hashCode()*, и *toString()*.[1][4]

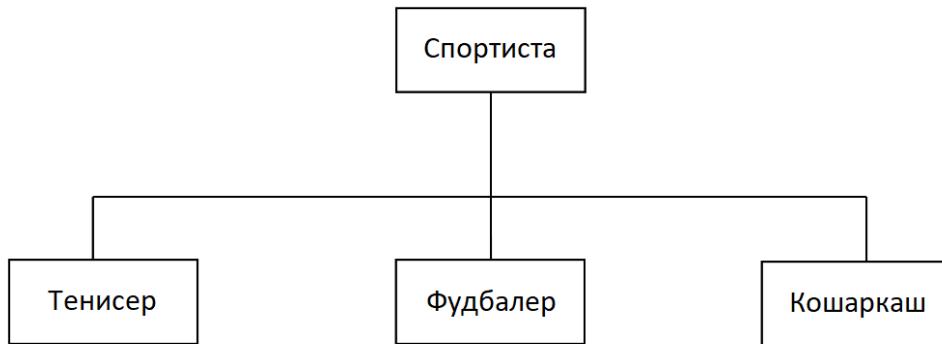


Слика 9. Класификација типова података у Јави[5]

Наслеђивање

У Јави, наслеђивање је механизам објектно-оријентисаног програмирања који омогућава креирање нових класа на основу постојећих. Кључна реч која се користи за наслеђивање је *extends*, која указује да нова класа (подкласа) наслеђује својства и методе од постојеће класе (родитељске класе). То значи да подкласа може да користи поља и методе родитељске класе, али и да их прилагођава или проширује новим функционалностима.

За Јаву је карактеристично једнострукото наслеђивање, што значи да класа може наследити само једну родитељску класу. Иако више класа може наследити исту родитељску класу, једна класа не може имати више од једног непосредног родитеља. Овај приступ помаже у решавању проблема који може да се јави када класа наслеђује више родитељских класа са конфлктним методама или пољима. У Јави се ова ограничења могу превазићи коришћењем интерфејса уз кључну реч *implements*.[4]



Слика 10. Наслеђивање

Уколико класа *Спортиста* има атрибуте *име*, *презиме* и *године* сваког конкретног спортисте, као и редефинисану *toString* методу која ове атрибуте исписује, и уколико класа *Тенисер* наслеђује класу *Спортиста*, структуру класе *Тенисер* приказану на слици 5 кориговаћемо тако да се састоји само од поља и метода које су својствене искључиво објектима типа *Тенисер*. Надкласу *Спортиста* у том случају могу наследити и друге подкласе (*Фудбалер*, *Кошаркаш* и сл.) користећи поља и методе родитељске класе које су заједничке свим спортистима уз могућност проширења јединственим карактеристикама конкретне класе.

```

public class Sportista {
    String ime;
    String prezime;
    int godine;

    @Override
    public String toString() {
        return "ime=" + ime + ", prezime=" + prezime + ", godine=" + godine;
    }
}
  
```

Слика 11. Структура родитељске класе Спортиста

Поред наслеђених поља и методе *toString* из класе *Спортиста*, класа *Тенисер* имаће додатни атрибут који је специфичан за тенисере – ранг на ATP листи. Такође, ова класа ће имати и методе за испис и измену тренутног ATP ранга. Оваква организација омогућава примену концепта наслеђивања у Јави чиме се избегава дуплирање кода у наслеђеним класама.

```

public class Tennisler extends Sportista{
    int rangATP;

    public void ispisiTrenutniRangNaATPListi(){
        System.out.println("Trenutni rang na ATP listi: "+rangATP);
    }
    public void izmeniRangNaATPListi(int noviRang){
        rangATP=noviRang;
    }
}
  
```

Слика 12. Структура подкласе Тенисер

Апстракције

Апстракција је један од најважнијих логичких концепата у Јави који омогућава скривање детаља имплементације. Према томе, фокус је на томе шта објекат ради, а не како то ради. У Јави постоје два начина за постизање апстракције: коришћењем апстрактних класа и коришћењем интерфејса.

Апстрактна класа у Јави је класа која се не може директно инстанцирати. Може имати апстрактне методе (без имплементације) које морају бити реализоване у подкласама, као и методе са имплементацијом. На овај начин омогућава се полиморфизам и поновна употреба кода. Апстрактне класе се обично користе када више класа дели заједничку логику, али су специфичне имплементације различите. Испред апстрактне класе и методе се користи кључна реч *abstract*, а приликом наслеђивања испред назива класе наводимо *extends*.[4]

```
public abstract class Sportista {  
    String ime;  
    String prezime;  
    int godine;  
  
    public abstract void ispisi();
```

Слика 13. Пример апстрактне класе

```
public class Teniser extends Sportista{  
    int rangATP;  
  
    @Override  
    public void ispisi() {  
        System.out.println("Teniser: "+ime+" "+prezime+", "+godine);  
    }  
}
```

Слика 14. Пример имплементације апстрактне методе исписи у класи Тенисер

```
public class Kosarkas extends Sportista{  
    int rangFIBA;  
  
    @Override  
    public void ispisi() {  
        System.out.println("Kosarkas: "+ime+" "+prezime+", "+godine);  
    }  
}
```

Слика 15. Пример имплементације апстрактне методе исписи у класи Кошаркаш

За разлику од апстрактних класа које омогућавају делимичну апстракцију, интерфејси у Јави омогућавају постизање потпуне апстракције. Све методе унутар интерфејса су подразумевано апстрактне и јавне, што значи да немају имплементацију. Интерфејси подржавају вишеструко наслеђивање, што значи да класа може имплементирати више интерфејса истовремено. Приликом имплементације интерфејса, користи се кључна реч *implements*.[4]

```
public interface Sportista {  
    public void ispisiSportistu();  
}
```

Слика 16. Пример интерфејса Спортиста

```
public interface TurnirIgrac {  
    public void ispisiTurnir();  
}
```

Слика 17. Пример интерфејса ТурнирИграч

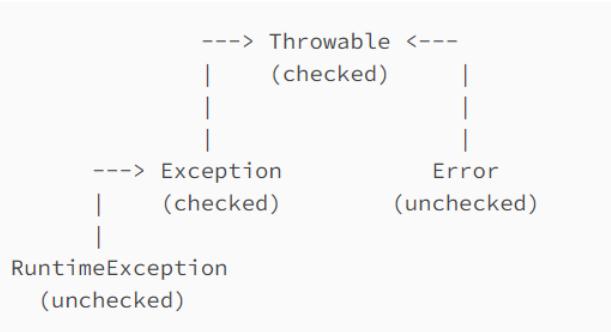
```
public class Teniser implements Sportista, TurnirIgrac{  
    String ime;  
    String prezime;  
    String nazivTurnira;  
    int godine;  
    int rangATP;  
  
    @Override  
    public void ispisiSportistu() {  
        System.out.println("Teniser: " + ime + " " + prezime + ", " + godine);  
    }  
    @Override  
    public void ispisiTurnir() {  
        System.out.println("Turnir: " + nazivTurnira);  
    }  
}
```

Слика 18. Пример имплементације интерфејса Спортиста и ТурнирИграч

Изузеци

У Јави, изузетак представља објекат који садржи податке о грешци која се дододила током извршавања програма. Сви изузети су појављивање класе *Throwable* или њених подкласа, што значи да сваки изузетак садржи детаље као што су тип грешке и стање програма у тренутку када је грешка настала.

У Јави, изузети су класификовани као проверени и непроверени. Проверени изузети представљају грешке које програм не може да контролише, попут недоступне датотеке, и морају се обраћивати у коду. Они се проверавају у време компајлирања. Непроверени изузети означавају логичке грешке у самом програму, као што је дељење са нулом, и не захтевају обраду. Проверени изузети наслеђују класу *Exception*, док непроверени изузети потичу од класе *RuntimeException*. [4]



Слика 19. Хијерархија изузетака у Јави[8]

Када се грешка појави, постоје два основна начина за њену обраду:

1. Try-catch блок

Код који може изазвати изузетак ставља се унутар *try* блока. Ако дође до грешке, прелази се на одговарајући *catch* блок, где се обрада изузетка одвија. Блок *finally* се увек извршава након *try* или *catch*, без обзира да ли је изузетак настало. Кључна реч *throw* се користи за експлицитно бацање изузетка.[4]

```
public class Teniser {  
    String ime;  
    String prezime;  
    int godine;  
    int rangATP;  
  
    public int getRangATP() {  
        return rangATP;  
    }  
  
    public void setRangATP(int rangATP) {  
        if (rangATP <= 0) {  
            throw new IllegalArgumentException("ATP rang mora biti pozitivan broj.");  
        }  
        this.rangATP = rangATP;  
    }  
  
    public static void main(String[] args) {  
        try {  
            Teniser t = new Teniser();  
            t.setRangATP(rangATP:-1);  
        } catch (IllegalArgumentException ex) {  
            System.out.println("Greška: " + ex.getMessage());  
        }  
    }  
}
```

Слика 20. Пример обраде грешке унутар try-catch блока

2. Клаузула *throws* у потпису методе

Овај начин преноси одговорност за обраду изузетка на позиваоца методе. Додавањем *throws* у потпис методе, програмер означава које изузетке метода може бацити, омогућавајући позиваоцу да их обради. [4]

```
public class Teniser implements Sportista, TurnirIgrac {  
  
    String ime;  
    String prezime;  
    int godine;  
    int rangATP;  
  
    public void proveriAtpRang() throws Exception {  
        if (rangATP <= 0) {  
            throw new Exception(message:"Nevažeći ATP rang. Rang mora biti veći od 0.");  
        }  
    }  
  
    public static void main(String[] args) {  
        try {  
            Teniser t = new Teniser();  
            t.proveriAtpRang();  
        } catch (Exception ex) {  
            System.out.println("Greška: " + ex.getMessage());  
        }  
    }  
}
```

Слика 21. Пример обраде грешке помоћу *throws* у потпису методе

3.2. Графички интерфејс у Јави

Java Foundation Classes (JFC) представљају скуп компоненти које омогућавају израду графичких корисничких интерфејса (GUI) у Јави. Обухвата и широко популаран Swing пакет компоненти за развој GUI апликација. Овај пакет нуди спектар компоненти као што су дугмад, табеле и текстуална поља. Swing компоненте су тзв. "лаке" (lightweight) јер њихов приказ не зависи од оперативног система, чиме се обезбеђује конзистентан изглед. Неки од кључних пакета за Swing су *javax.swing* за основне компоненте и *javax.swing.event* за руковање догађајима.[8]

Swing компоненте

У Јава Swing-у, контејнери су компоненте које могу садржати друге компоненте, пружајући основну структуру за графички кориснички интерфејс. Постоје три контејнерске класе на највишем нивоу: *JFrame*, *JDialog* и *JApplet*.

1. **JFrame** - користи се за креирање главног (top-level) прозора најчешће са навигационим менијем.
2. **JDialog** - користи се за приказивање модалних или немодалних подпрозора.
3. **JApplet** - омогућава креирање апликација коришћењем Swing компоненти које се могу извршавати унутар веб прегледача. Данас је мање коришћен због безбедносних проблема и преласка на модерније технологије.[8]

У хијерархији Јава Swing-а, класа *JTextComponent* је основна апстрактна класа за све компоненте које омогућавају приказивање и уређивање текста, док је *AbstractButton* основна апстрактна класа за све типове дугмади. Наслеђују класу *JComponent*, која је део пакета *javax.swing*.[8]

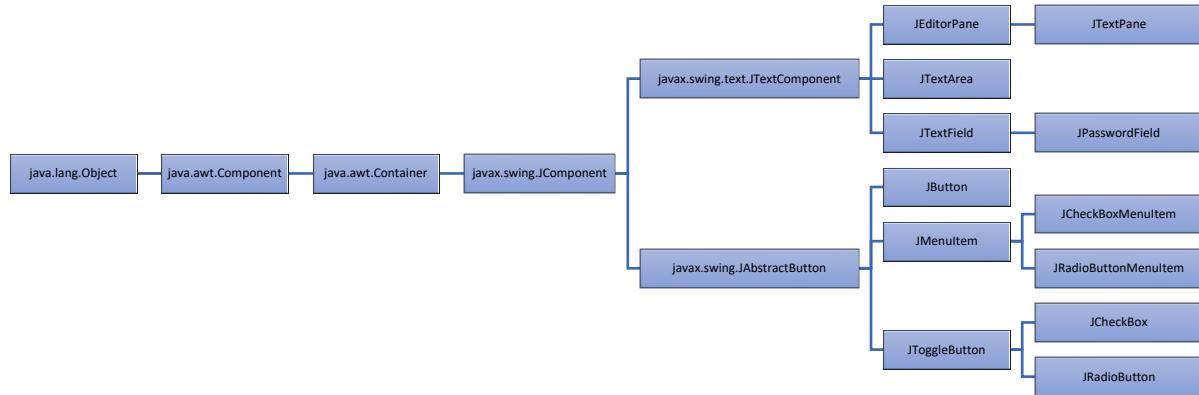
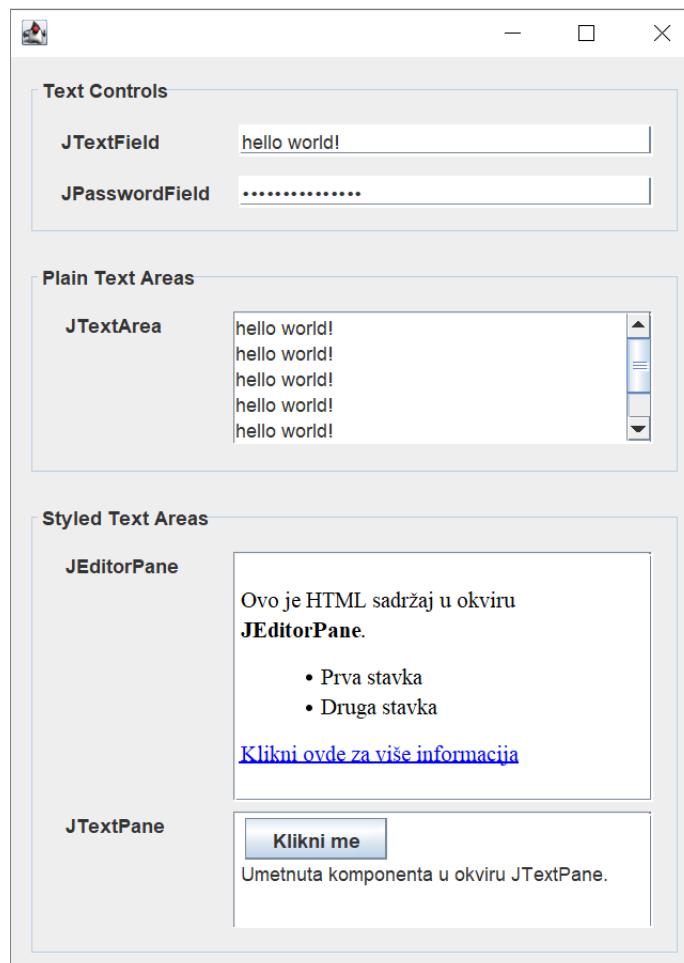
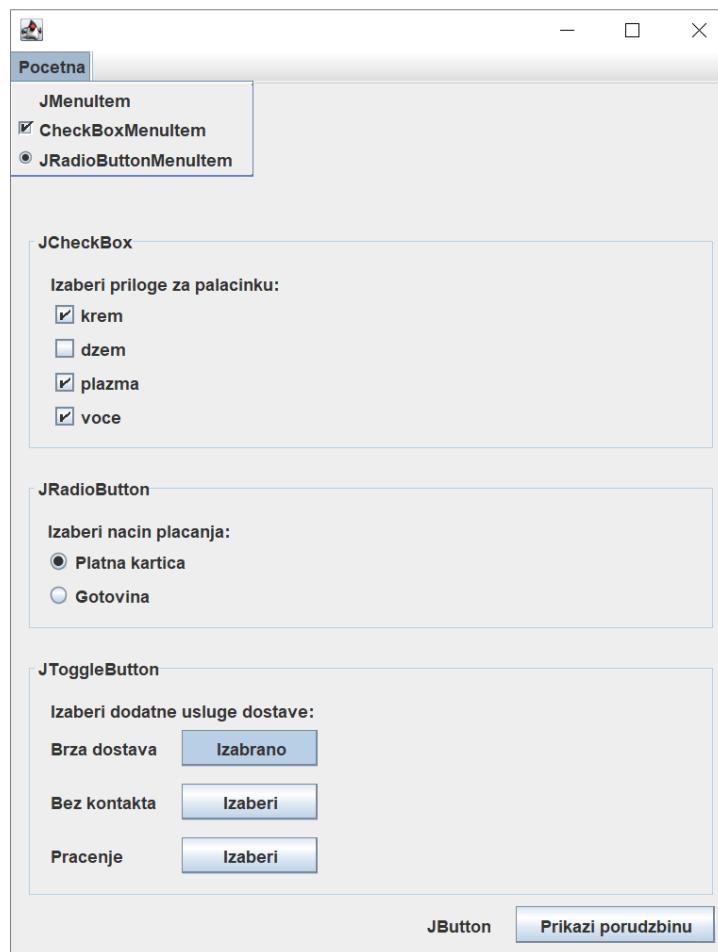


График 1. Хијерархија основних Јава Swing компоненти



Слика 22. Основне Swing текстуалне компоненте



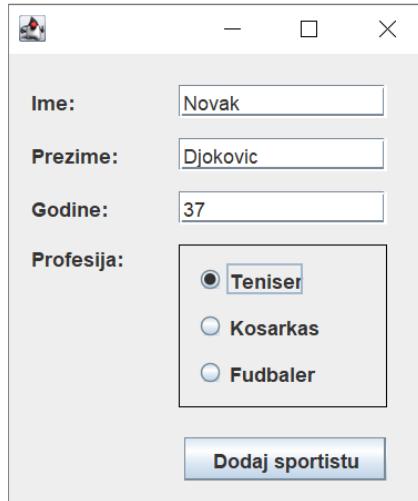
Слика 23. Основне Swing компоненте дугмића

Догађаји

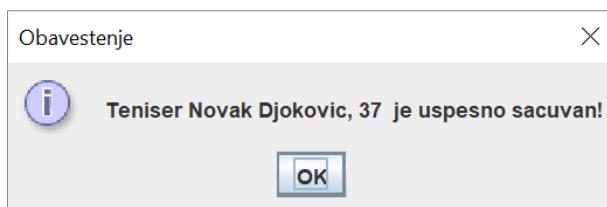
Догађај представља промену стања неког објекта. У контексту графичког корисничког интерфејса (GUI), догађај може бити, на пример, клик на дугме, клик миша или његово померање, унос преко тастатуре итд. У Јави се за руковање догађајима (*event handling*) користи *Delegation Event Model*, који је заснован на томе да извор догађаја генерише догађај и шаље га једном или већем броју слушалаца који су регистровани за тај догађај. Кључни елементи овог модела су:

1. **Извор (Event Source)**: Објекат који генерише догађај. Најчешће су то GUI компоненте као што су дугмад, погља за унос текста или ставке менија. Када се деси догађај, извор догађаја га шаље слушалацу.
2. **Слушалац (Event Listener)**: Објекат који ослушкије догађај и реагује на њега тако што извршава одређену акцију. На пример, *ActionListener* реагује на клик дугмета имплементирајући методу *actionPerformed()*.
3. **Догађај (Event Object)**: Овај објекат учајује детаље о догађају, као што су његов извор и додатни параметри.[8]

Уколико на форми имамо текстуална поља у које корисник уноси текст и дугме на чији клик се унесене информације даље чувају и приказују кориснику, онда дугме у том случају представља *Event Source* објекат. Када се на дугме кликне, *Event Listener* који је повезан са тим објектом позива функцију *actionPerformed()* која има задатак да прикаже поруку обавештења.



Слика 24. Обрада догађаја кликом на дугме



Слика 25. Резултат обраде догађаја кликом на дугме

Код који одговара примеру приказаном на сликама 24 и 25 дат је на слици 26.

```
public class Dogadjaji extends javax.swing.JFrame {

    public Dogadjaji() {
        initComponents();
    }

    @SuppressWarnings("unchecked")
    Generated Code

    private void btnAddActionPerformed(java.awt.event.ActionEvent evt) {
        String ime = txtFirstname.getText();
        String prezime = txtLastname.getText();
        String godine = txtAge.getText();
        String profesija;
        if (rbBasketball.isSelected()) {
            profesija = "Kosarkas";
        } else if (rbFootball.isSelected()) {
            profesija = "Fudbaler";
        } else {
            profesija = "Teniser";
        }
        JOptionPane.showMessageDialog(parentComponent:this, profesija + " " + ime + " " + prezime +
            ", " + godine + " je uspesno sacuvan!", title: "Obavestenje",
            messageType: JOptionPane.INFORMATION_MESSAGE);
    }
}
```

Слика 26. Код графичког интерфејса

3.3 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. Тада проблем је решен појавом нити (енг. threads) које деле исти меморијски простор.[3]

У Јави, нити су основни елементи конкурентног програмирања које омогућава извршавање више задатака истовремено. Свака нит наслеђује класу *Thread* или имплементира интерфејс *Runnable*.

Главна нит (main thread) је почетна нит у Јави, која се аутоматски креира приликом покретања програма. Сви додатни задаци могу се извршавати у новим нитима које апликација креира. Уобичајен начин креирања нити је проширивањем класе *Thread* или имплементацијом интерфејса *Runnable*, а нова нит се покреће позивањем методе *start()*, која аутоматски извршава *run()* методу.[8]

The screenshot shows an IDE interface with a code editor and a terminal window. The code editor contains the following Java code:

```
public class Nit1 implements Runnable {  
    @Override  
    public void run() {  
        System.out.println("Kreirana je nit implementacijom interfejsa Runnable.");  
    }  
    public static void main(String[] args) {  
        Nit1 nit = new Nit1();  
        Thread t = new Thread(task:nit);  
        t.start();  
    }  
}
```

The terminal window below the code editor shows the output of the program:

```
- DiplomskiPrimeri (run) X Test Results Search Results  
run:  
Kreirana je nit implementacijom interfejsa Runnable.  
BUILD SUCCESSFUL (total time: 0 seconds)
```

Слика 27. Креирање нити помоћу класе *Runnable*

The screenshot shows an IDE interface with a code editor and a terminal window. The code editor contains the following Java code:

```
public class Nit2 extends Thread {  
    @Override  
    public void run() {  
        System.out.println("Kreirana je nit prosirivanjem klase Thread.");  
    }  
    public static void main(String[] args) {  
        Nit2 nit = new Nit2();  
        Thread t = new Thread(task:nit);  
        t.start();  
    }  
}
```

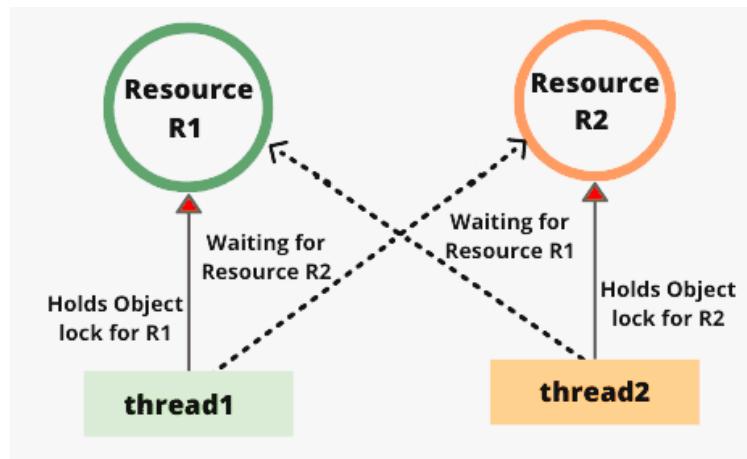
The terminal window below the code editor shows the output of the program:

```
out - DiplomskiPrimeri (run) X Test Results Search Results  
run:  
Kreirana je nit prosirivanjem klase Thread.  
BUILD SUCCESSFUL (total time: 0 seconds)
```

Слика 28. Креирање нити помоћу класе *Thread*

Deadlock у Јави настаје када две или више нити остану блокиране, јер свака од њих чека на ресурс који је заузет од стране друге нити. [8][4]

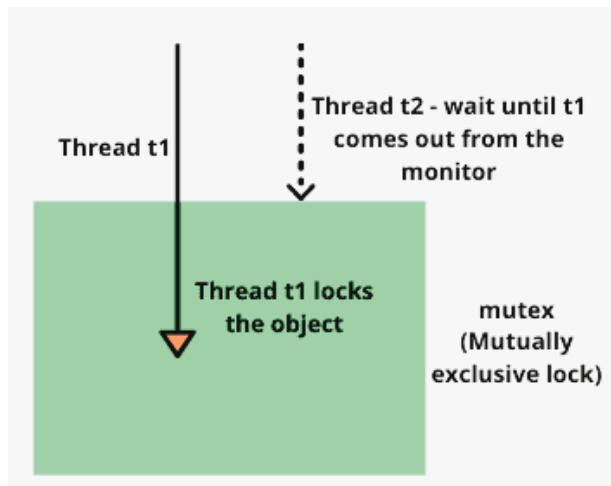
На пример, можемо да замислим две нити које покушавају да закључају два ресурса у различитом редоследу. Ако нит 1 закључа ресурс R1 и чека на ресурс R2, док нит 2 закључује ресурс R2 и чека на ресурс R1, ниједна нит не може да настави са извршавањем, што доводи до deadlock-а. [8][4]



Слика 29. Графички приказ Deadlock проблема[9]

Синхронизација у Јави се користи за контролу приступа ресурсима које деле више нити. За синхронизацију се у Јави најчешће користи кључна реч *synchronized*. *Synchronized* блокови или методе обезбеђују да само једна нит може приступити траженом ресурсу у датом тренутку. [8][4]

Када нит уђе у синхронизовани блок или се синхронизована метода позове, ресурс се закључава. Ако друга нит покуша да приступи том истом блоку или позове исту методу, мораће да сачека да прва нит ослободи тражени ресурс. Ово спречава истовремени приступ који може довести до грешака због неконзистентних стања података. [8][4]



Слика 30. Графички приказ синхронизације нити[9]

3.4 Рад у мрежи

Сокети

Сокет је механизам који омогућава комуникацију између програма који се извршавају на различитим рачунарима у мрежи. Сокет програмирање у Јави омогућава комуникацију између клијентских и серверских апликација, користећи TCP/IP протокол за пренос података.[8]

Главне класе за сокет програмирање у Јави су *ServerSocket* и *Socket*:

- **ServerSocket** се користи на серверу за ослушкивање мреже ради остваривања конекције на одређеном порту. Када се захтев за конекцијом прими, сервер га приhvата коришћењем методе *accept()*, која враћа ново појављивање класе *Socket* за обраду комуникације са клијентом. На овај начин, сервер може истовремено одржавати више веза са различитим клијентима, јер ће *accept()* метода за сваки долазни захтев вратити ново појављивање класе *Socket*, која представља везу са одређеним клијентом.

```
public class Server {  
  
    /**  
     * @param args the command line arguments  
     */  
    public static void main(String[] args) {  
        int port = 9000;  
        try {  
            ServerSocket serverSocket = new ServerSocket(port);  
            Socket socket = serverSocket.accept();  
            //klijent-server komunikacija  
        } catch (IOException ex) {  
            System.out.println(ex.getMessage());  
        }  
    }  
}
```

Слика 31. Приказ кода за подизање серверског сокета

- **Socket** представља везу између клијента и сервера. Обе стране, и клијентска и серверска, имају своје појављивање ове класе. На страни клијента, користи се за успостављање везе са сервером коришћењем IP адресе и порта сервера. Успешним успостављањем ове везе, омогућена је комуникација између серверске и клијентске стране.[8]

```

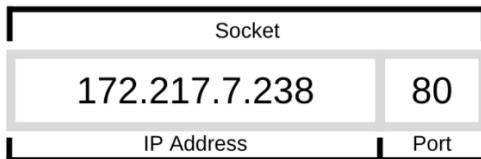
public class Klijent {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        int port = 9000;
        String adresa = "127.0.0.1";
        try {
            Socket clientSocket = new Socket(host:adresa, port);
            //klijent-server komunikacija
        } catch (IOException ex) {
            System.out.println(: ex.getMessage());
        }
    }
}

```

Слика 32. Приказ кода за успостављање везе са серверским сокетом

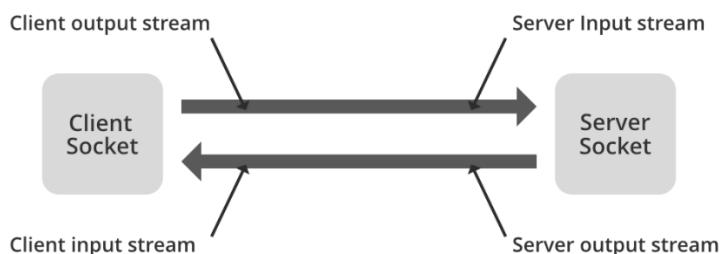
Са слике 32 видимо да су за успостављање везе са серверским сокетом клијентском програму неопходне информације о адреси рачунара на коме се налази серверски програм и броју порта које јединствено идентификују тај серверски сокет. Тачније, сама адреса сокета се састоји из ова два дела.[8]



Слика 33. Приказ адресе сокета[6]

По успешној конекцији, потребно је отворити токове за слање и примање података. Методе у оквиру класе *Socket* *getInputStream()* и *getOutputStream()* у Јави омогућавају приступ улазном и излазном току података повезаног сокета, који се користе за читање и писање тј. размену података између клијента и сервера:

1. ***getInputStream()*** метода враћа објекат типа *InputStream* који представља улазни ток података. Овај ток прима податке који долазе са повезаног сокета (клијентског или серверског).
2. ***getOutputStream()*** метода враћа објекат типа *OutputStream* који претставља излазни ток података. Овај ток се користи за слање података другој страни преко повезаног сокета.[8]

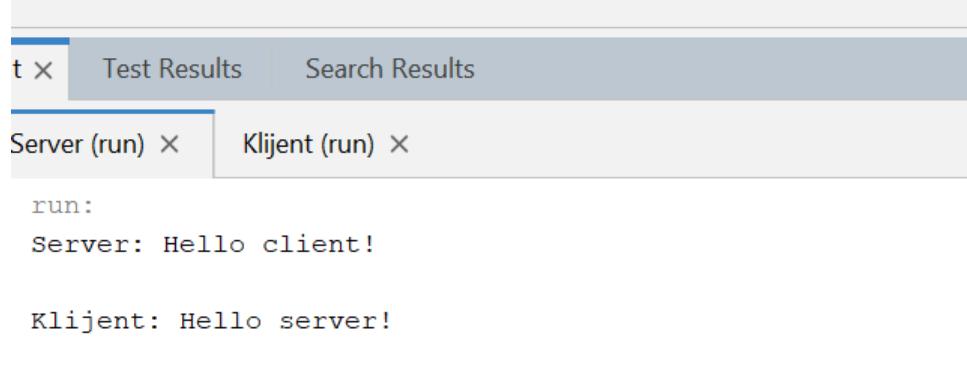


Слика 34. Приказ улазних и излазних токова података[5]

BufferedReader и *PrintWriter* су кључне класе у Јави које се најчешће користе за слање и примање података преко мреже јер пружају једноставне и ефикасне механизме за рад са текстуалним подацима. Оне служе за читање и писање података са улазног, односно на излазни ток сокета.[8]

На следећим slikama дат је пример размене података на једној клијент-сервер апликацији.

```
public class Server {
    public static void main(String[] args) {
        int port = 9000;
        Socket socket = null;
        PrintWriter out = null;
        BufferedReader in = null;
        try {
            ServerSocket serverSocket = new ServerSocket(port);
            socket = serverSocket.accept();
            //slanje poruke klijentu
            out = new PrintWriter(out.socket.getOutputStream(), autoFlush: true);
            String message="Hello client!";
            out.println(message);
            System.out.println("Server: "+message+"\n");
            //citanje poruke klijenta
            in = new BufferedReader(new InputStreamReader(in: socket.getInputStream()));
            String receivedMessage = in.readLine();
            System.out.println("Klijent: "+receivedMessage+"\n");
        } catch (IOException ex) {
            System.out.println(ex.getMessage());
        } finally {
            try {
                if (out != null)
                    out.close();
                if (in != null)
                    in.close();
                if (socket != null & !socket.isClosed())
                    socket.close();
            } catch (IOException ex) {
                System.out.println(ex.getMessage());
            }
        }
    }
}
```



Слика 35. Приказ кода серверског програма

```

public class Klijent {
    public static void main(String[] args) {
        int port = 9000;
        String adresa = "127.0.0.1";
        Socket clientSocket = null;
        BufferedReader in = null;
        PrintWriter out = null;
        try {
            clientSocket = new Socket(host:adresa, port);
            //citanje poruke servera
            in = new BufferedReader(new InputStreamReader(in: clientSocket.getInputStream()));
            String receivedMessage = in.readLine();
            System.out.println("Server: "+receivedMessage+"\n");
            //slanje poruke serveru
            out = new PrintWriter(out:clientSocket.getOutputStream(), autoFlush: true);
            String message="Hello server!";
            out.println(: message);
            System.out.println("Klijent: "+message+"\n");
        } catch (IOException ex) {
            System.out.println(: ex.getMessage());
        } finally {
            try {
                if (out != null)
                    out.close();
                if (in != null)
                    in.close();
                if (clientSocket != null && !clientSocket.isClosed())
                    clientSocket.close();
            } catch (IOException ex) {
                System.out.println(: ex.getMessage());
            }
        }
    }
}

```

The screenshot shows an IDE interface with a toolbar at the top. The 'Test Results' tab is active. Below it, two tabs are visible: 'Server (run)' and 'Klijent (run)'. The 'Klijent (run)' tab is highlighted with a blue border. The main area displays the following text:

```

run:
Server: Hello client!

Klijent: Hello server!

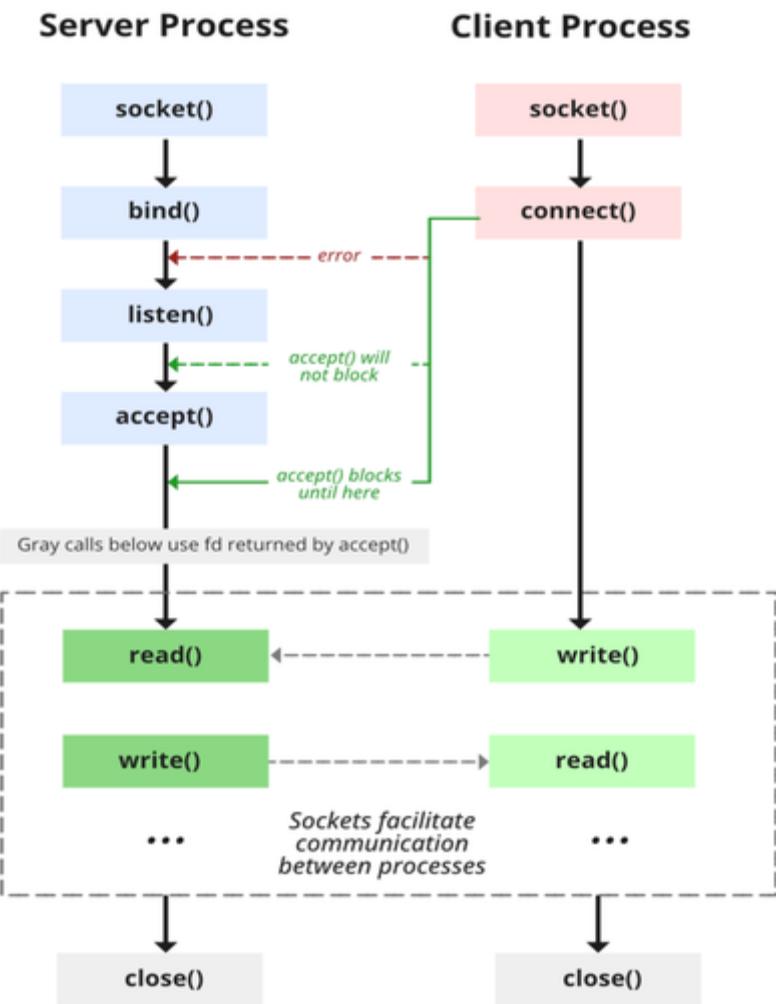
BUILD SUCCESSFUL (total time: 0 seconds)

```

Слика 36. Приказ кода клијентског програма

Као последњи корак при раду са мрежним сокетима, затварање улазних и излазних токова, као и самог сокета, је кључно како би се осигурало правилно ослобађање системских ресурса и избегле потенцијалне грешке у програму. Овај корак је важан за одржавање стабилности и перформанси апликације.

Графички приказ основних корака сокет програмирања дат је на слици 37.



Слика 37. Основни кораки при сокет програмирању[5]

Као основне кораке при сокет програмирању можемо издвојити:

Подизање серверског сокета: Сервер креира `ServerSocket` који ослушкује везу и чека клијенте, држећи отворен порт за конекцију, а метода `accept()` блокира извршавање програма све док се клијент не повеже.

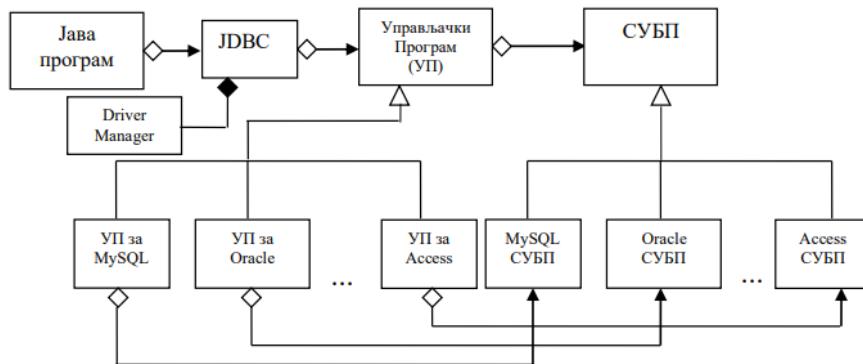
Повезивање клијената: Клијент креира `Socket`, наводећи IP адресу и порт сервера. Ако је веза успешна, успоставља се комуникација.

Размена података: Након успостављања везе, сервер и клијент отварају улазне и излазне токове сокета за слање и примање података.

Затварање токова и сокета: Када је комуникација завршена, обе стране затварају своје токове и сокете како би ослободиле ресурсе и избегле потенцијалне грешке.

3.5 Рад са базом података

Повезивање неког програма који је написан у Јави и неког од система за управљање базом података(MySQL, Oracle, SQL Server...) се ради преко Јавиног JDBC(Java Database Connectivity) и управљачког програма (драјвера) који се прави посебно за сваки СУБП.[3]



Слика 38. Веза Јава програма са СУБП[3]

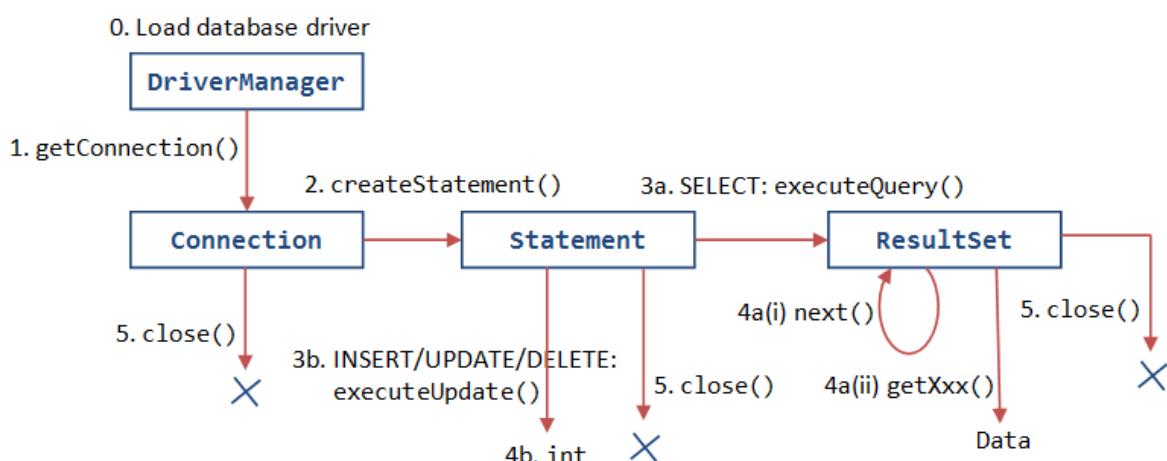
Основни кораци при повезивању Јава програма и базе података изабраног система за управљање базом података као и извршавању SQL упита су:

- Преузимање и додавање JDBC драјвера:** Преузимање одговарајућег .jar фајла драјвера изабраног система за управљање базом података и његово додавање.
- Учитавање JDBC драјвера:** Учитавање управљачког програма (драјвера) у Јава програм представља корак у којем се иницијализује JDBC драјвер који омогућава комуникацију са базом података. У овом процесу, Јава апликација мора да учита JDBC драјвер у меморију, како би он могао да обради захтеве за везу са базом података. У пракси, то се обично ради коришћењем *Class.forName()* методе, која учитава класу драјвера у JVM (Java Virtual Machine). Ова наредба омогућава да Јава програм учита специфичан драјвер који управља комуникацијом са базом, што је неопходно пре успостављања саме везе са базом података. Ово се ради како би апликација могла да користи *DriverManager* за повезивање са базом, а сам драјвер садржи логичке операције за обраду SQL упита и комуникацију са базом података. У новијим верзијама Јаве, експлицитно учитавање драјвера помоћу *Class.forName()* методе више није неопходно. Увођењем JDBC 4.0, драјвери се аутоматски региструју и учитавају приликом додавања. Ипак, због компатibilности са старијим верзијама Јаве, овај корак је понекад неопходан.
- Креирање везе са базом података:** За успостављање конекције, користи се *getConnection()* метода класе *DriverManager*. Као аргументе методе потребно је навести URL базе, корисничко име и лозинку.
- Извршавање SQL упита:** Једном када је веза успостављена, могуће је користити *Statement* или *PreparedStatement* објекте за извршавање SQL упита, као што су *SELECT*, *INSERT*, *UPDATE*, и *DELETE*.

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. У случају коришћења *PreparedStatement-a*, објекат се креира помоћу методе *prepareStatement()* над *Connection* објектом коме се такође прослеђује SQL упит као параметар методе.

За извршавање INSERT, UPDATE или DELETE наредби користи се метода `executeUpdate()` позvana над објектима Statement или PreparedStatement. Ова метода, као повратну вредност враћа цео број који представља број редова у табели који су изменени. За SELECT упите, користи се метода `executeQuery()`, који враћа објекат типа ResultSet који омогућава кретање кроз редове резултата извршеног упита.

5. **Обрада резултата:** Резултати упита могу се обраћивати помоћу ResultSet објекта који омогућава читање података враћених упитом.
6. **Затварање ресурса:** Када се заврши рад са базом података, сви коришћени ресурси као што су `ResultSet`, `Statement` и `Connection` објекти затварају се да би се ослободили системски ресурси који су били заузети. Затварањем објеката се омогућава да JVM (Java Virtual Machine) и база података правилно управљају ресурсима, чиме се оптимизује рад апликације и избегавају потенцијални проблеми са недостатком меморије.[8]



Слика 39. Кораци за повезивање са базом података и извршавање SQL упита у Јава апликацији[7]

У наставку је приказан поступак повезивања Јава програма са базом података, као и пример извршавања SELECT наредбе.

```

try {
    String url = "jdbc:mysql://127.0.0.1:3306/nursery_school";
    String user = "root";
    String password = "root";
    Connection connection = DriverManager.getConnection(url, user, password);
    Statement statement=connection.createStatement();
    String query="SELECT username,firstname FROM user";
    ResultSet rs=statement.executeQuery(string: query);
    while(rs.next()){
        System.out.println("Ime: "+rs.getString(string: "firstname")+", Korisnicko ime: "+rs.getString(string: "username"));
    }
    statement.close();
    connection.close();
} catch (SQLException ex) {
    System.out.println(x: ex.getMessage());
}

```

Слика 40. Повезивање са базом и извршавање SELECT наредбе

4. Студијски пример

4.1 Прикупљање корисничких захтева

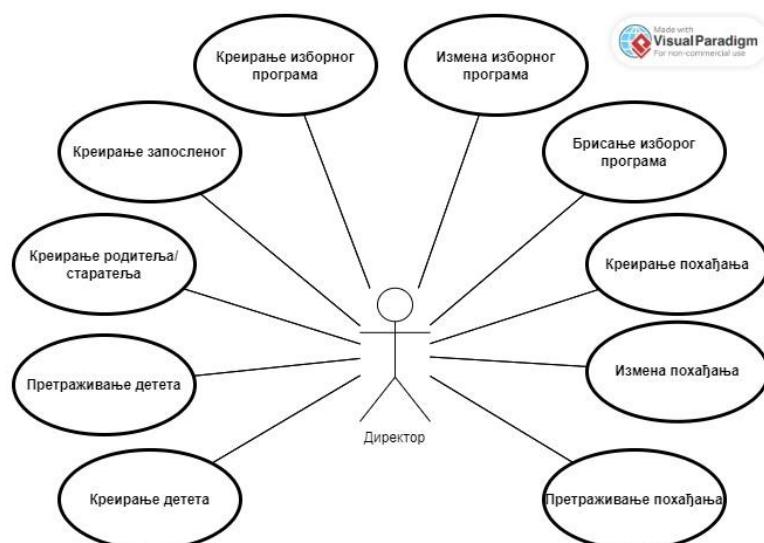
4.1.1 Вербални опис система

Директору предшколске установе потребан је увид у децу уписану у текућој радној години, њихове родитеље, односно, старатеље, запослене на изборним програмима као и у доступне програме за децу у циљу обрачуна зараде од организације истих и планирања кадрова и ресурса. Омогућено му је креирање новог профиле детета и његово претраживање, креирање новог профиле запосленог као и новог профиле родитеља, односно, старатеља детета. Такође, може креирати изборни програм, вршити измене над њим или га по потреби (уколико је застарео) и обрисати. Када се дете определи за један или више изборних програма директор креира ново похађање са подацима о детету и изабраним програмима. Омогућена је и накнадна измена похађања.

4.1.2 Случајеви коришћења

У овој апликацији идентификовано је десет случајева коришћења:

1. Креирање детета
2. Претраживање детета
3. Креирање родитеља/старатеља
4. Креирање запосленог
5. Креирање изборног програма
6. Измена изборног програма
7. Брисање изборног програма
8. Креирање похађања (сложен СК)
9. Измена похађања (сложен СК)
10. Претраживање похађања



Слика 41. Модел случајева коришћења

СК1: Случај коришћења – Креирање детета

Назив СК

Креирање детета

Актори СК

Директор

Учесници СК

Директор и систем (програм)

Предуслов: Систем је укључен и директор је пријављен под својом шифром. Систем приказује форму за рад са дететом.

Основни сценарио СК

1. Директор уноси податке у дете. (АПУСО)
2. Директор контролише да ли је коректно унео податке у дете. (АНСО)
3. Директор позива систем да запамти податке о детету. (АПСО)
4. Систем памти податке о детету. (СО)
5. Систем приказује директору запамћени дете и поруку: "Систем је запамтио дете". (ИА)

Алтернативна сценарија

- 5.1 Уколико систем не може да запамти податке о детету он приказује директору поруку "Систем не може да запамти дете". (ИА)

СК2: Случај коришћења – Претраживање детета

Назив СК

Претраживање [детета](#)

Актори СК

[Директор](#)

Учесници СК

[Директор](#) и [систем](#) (програм)

Предуслов: Систем је укључен и [директор](#) је пријављен под својом шифром. Систем приказује форму за рад са [дететом](#).

Основни сценарио СК

1. [Директор уноси](#) вредност по којој претражује [децу](#). (АПУСО)
2. [Директор позива](#) [систем](#) да нађе [децу](#) по задатој вредности. (АПСО)
3. [Систем тражи](#) [децу](#) по задатој вредности. (СО)
4. [Систем](#) приказује [директору](#) податке о [дечи](#) и поруку: “[Систем](#) је нашао [децу](#) по задатој вредности”. (ИА)
5. [Директор бира](#) [дете](#). (АПУСО)
6. [Директор позива](#) [систем](#) да учита [дете](#). (АПСО)
7. [Систем учитава](#) [дете](#). (СО)
8. [Систем приказује](#) [директору](#) податке о [детету](#) и поруку: “[Систем](#) је учитao [дете](#).” (ИА)

Алтернативна сценарија

4.1 Уколико [систем](#) не може да нађе [децу](#) он приказује [директору](#) поруку: “[Систем](#) не може да нађе [децу](#) по задатој вредности”. Прекида се извршење сценарија. (ИА)

8.1 Уколико [систем](#) не може да учита [дете](#) он приказује [директору](#) поруку: “[Систем](#) не може да учита [дете](#). (ИА)

СК3: Случај коришћења – Креирање родитеља/старатеља

Назив СК

Креирање родитеља/старатеља

Актори СК

Директор

Учесници СК

Директор и систем (програм)

Предуслов: Систем је укључен и директор је пријављен под својом шифром. Систем приказује форму за рад са родитељем/старатељем.

Основни сценарио СК

1. Директор уноси податке у родитеља/старатеља. (АПУСО)
2. Директор контролише да ли је коректно унео податке у родитеља/старатеља. (АНСО)
3. Директор позива систем да запамти податке о родитељу/старатељу. (АПСО)
4. Систем памти податке о родитељу/старатељу. (СО)
5. Систем приказује директору запамћеног родитеља/старатеља и поруку: “Систем је запамтио родитеља/старатеља”. (ИА)

Алтернативна сценарија

- 5.1 Уколико систем не може да запамти податке о родитељу/старатељу он приказује директору поруку “Систем не може да запамти родитеља/старатеља”. (ИА)

СК4: Случај коришћења – Креирање запосленог

Назив СК

Креирање запосленог

Актори СК

Директор

Учесници СК

Директор и систем (програм)

Предуслов: Систем је укључен и директор је пријављен под својом шифром. Систем приказује форму за рад са запосленим . Учитана је листа изборних програма.

Основни сценарио СК

1. Директор уноси податке у запосленог. (АПУСО)
2. Директор контролише да ли је коректно унео податке у запосленог. (АНСО)
3. Директор позива систем да запамти податке о запосленом. (АПСО)
4. Систем памти податке о запосленом. (СО)
5. Систем приказује директору запамћеног запосленог и поруку: “Систем је запамтио запосленог.”. (ИА)

Алтернативна сценарија

- 5.1 Уколико систем не може да запамти податке о запосленом он приказује директору поруку “Систем не може да запамти запосленог”. (ИА)

СК5: Случај коришћења – Креирање изборног програма

Назив СК

Креирање изборног програма

Актори СК

Директор

Учесници СК

Директор и систем (програм)

Предуслов: Систем је укључен и директор је пријављен под својом шифром. Систем приказује форму за рад са изборним програмом.

Основни сценарио СК

1. Директор уноси податке у изборни програм. (АПУСО)
2. Директор контролише да ли је коректно унео податке у изборни програм. (АНСО)
3. Директор позива систем да запамти податке о изборном програму. (АПСО)
4. Систем памти податке о изборном програму. (СО)
5. Систем приказује директору запамћени изборни програм и поруку: "Систем је запамтио изборни програм". (ИА)

Алтернативна сценарија

- 5.1 Уколико систем не може да запамти податке о изборном програму он приказује директору поруку "Систем не може да запамти изборни програм". (ИА)

СК6: Случај коришћења –Измена изборног програма

Назив СК

Промена изборног програма

Актори СК

Директор

Учесници СК

Директор и систем (програм)

Предуслов: Систем је укључен и директор је пријављен под својом шифром. Систем приказује форму за рад са изборним програмом.

Основни сценарио СК

1. Директор уноси вредност по којој претражује изборне програме. (АПУСО)
2. Директор позива систем да нађе изборне програме по задатој вредности. (АПСО)
3. Систем тражи изборне програме по задатој вредности. (СО)
4. Систем приказује директору изборне програме и поруку: "Систем је нашао изборне програме по задатој вредности". (ИА)
5. Директор бира изборни програм. (АПУСО)
6. Директор позива систем да учита изборни програм. (АПСО)
7. Системчитава изборни програм. (СО)
8. Систем приказује директору податке о изборном програму и поруку: "Систем је учитао изборни програм." (ИА)
9. Директор уноси (мења) податке о изборном програму. (АПУСО)
10. Директор контролише да ли је коректно унео податке о изборном програму. (АНСО)
11. Директор позива систем да запамти податке о изборном програму. (АПСО)
12. Систем памти податке о изборном програму. (СО)
13. Систем приказује директору запамћени изборни програм и поруку: "Систем је запамтио изборни програм." (ИА)

Алтернативна сценарија

4.1 Уколико систем не може да нађе изборне програме он приказује директору поруку: "Систем не може да нађе изборне програме по задатој вредности". Прекида се извршење сценарија. (ИА)

8.1 Уколико систем не може да учита изборни програм он приказује директору поруку: "Систем не може да учита изборни програм. Прекида се извршење сценарија. (ИА)

13.1 Уколико **систем** не може да запамти податке о **изборном програму** он приказује **директору** поруку “**Систем** не може да запамти изборни програм”. (ИА)

СК7: Случај коришћења –Брисање изборног програма

Назив СК

Брисање изборног програма

Актори СК

Директор

Учесници СК

Директор и систем (програм)

Предуслов: Систем је укључен и директор је пријављен под својом шифром. Систем приказује форму за рад са изборним програмом.

Основни сценарио СК

1. Директор уноси вредност по којој претражује изборне програме. (АПУСО)
2. Директор позива систем да нађе изборне програме по задатој вредности. (АПСО)
3. Систем тражи изборне програме по задатој вредности. (СО)
4. Систем приказује директору изборне програме и поруку: "Систем је нашао изборне програме по задатој вредности". (ИА)
5. Директор бира изборни програм. (АПУСО)
6. Директор позива систем да учита изборни програм. (АПСО)
7. Системчитава изборни програм. (СО)
8. Систем приказује директору податке о изборном програму и поруку: "Систем је учитао изборни програм." (ИА)
9. Директор позива систем да обрише изборни програм. (АПСО)
10. Систем брише изборни програм. (СО)
11. Систем приказује директору поруку: "Систем је обрисао изборни програм." (ИА)

Алтернативна сценарија

4.1 Уколико систем не може да нађе изборне програме он приказује директору поруку: "Систем не може да нађе изборне програме по задатој вредности". Прекида се извршење сценарија. (ИА)

8.1 Уколико систем не може да учита изборни програм он приказује директору поруку: "Систем не може да учита изборни програм. Прекида се извршење сценарија. (ИА)

11.1 Уколико систем не може да обрише изборни програм он приказује директору поруку "Систем не може да обрише изборни програм". (ИА)

СК8: Случај коришћења –Креирање похађања(сложен СК)

Назив СК

Креирање похађања

Актори СК

Директор

Учесници СК

Директор и систем (програм)

Предуслов: Систем је укључен и директор је пријављен под својом шифром. Систем приказује форму за рад са похађањем. Учитана је листа деце и листа изборних програма.

Основни сценарио СК

1. Директор уноси податке у похађање. (АПУСО)
2. Директор контролише да ли је коректно унео податке у похађање. (АНСО)
3. Директор позива систем да запамти податке о похађању. (АПСО)
4. Систем памти податке о похађању. (СО)
5. Систем приказује директору запамћено похађање и поруку: “Систем је запамтио похађање”. (ИА)

Алтернативна сценарија

- 5.1 Уколико систем не може да запамти податке о похађању он приказује директору поруку “Систем не може да запамти похађање”. (ИА)

СК9: Случај коришћења –Измена похађања(сложен СК)

Назив СК

Промена похађања

Актори СК

Директор

Учесници СК

Директор и систем (програм)

Предуслов: Систем је укључен и директор је пријављен под својом шифром. Систем приказује форму за рад са похађањем. Учитана је листа деце и листа изборних програма.

Основни сценарио СК

1. Директор уноси вредност по којој претражује похађања. (АПУСО)
2. Директор позива систем да нађе похађања по задатој вредности. (АПСО)
3. Систем тражи похађања по задатој вредности. (СО)
4. Систем приказује директору похађања и поруку: "Систем је нашао похађања по задатој вредности". (ИА)
5. Директор бира похађање. (АПУСО)
6. Директор позива систем да учита похађање. (АПСО)
7. Систем учитава похађање. (СО)
8. Систем приказује директору податке о похађању и поруку: "Систем је учитао похађање." (ИА)
9. Директор уноси (мења) податке о похађању. (АПУСО)
10. Директор контролише да ли је коректно унео податке о похађању. (АНСО)
11. Директор позива систем да запамти податке о похађању. (АПСО)
12. Систем памти податке о похађању. (СО)
13. Систем приказује директору запамћено похађање и поруку: "Систем је запамтио похађање." (ИА)

Алтернативна сценарија

- 4.1 Уколико систем не може да нађе похађања он приказује директору поруку: "Систем не може да нађе похађања по задатој вредности". Прекида се извршење сценарија. (ИА)
- 8.1 Уколико систем не може да учита похађање он приказује директору поруку: "Систем не може да учита похађање". Прекида се извршење сценарија. (ИА)
- 13.1 Уколико систем не може да запамти податке о похађању он приказује директору поруку "Систем не може да запамти похађање". (ИА)

СК10: Случај коришћења – Претраживање похађања

Назив СК

Претраживање похађања

Актори СК

Директор

Учесници СК

Директор и систем (програм)

Предуслов: Систем је укључен и директор је пријављен под својом шифром. Систем приказује форму за рад са похађањем.

Основни сценарио СК

1. Директор уноси вредност по којој претражује похађања. (АПУСО)
2. Директор позива систем да нађе похађања по задатој вредности. (АПСО)
3. Систем тражи похађања по задатој вредности. (СО)
4. Систем приказује директору податке о похађањима и поруку: "Систем је нашао похађања по задатој вредности". (ИА)
5. Директор бира похађање. (АПУСО)
6. Директор позива систем да учита похађање. (АПСО)
7. Систем учитава похађање. (СО)
8. Систем приказује директору податке о похађању и поруку: "Систем је учитао похађање." (ИА)

Алтернативна сценарија

4.1 Уколико систем не може да нађе похађања он приказује директору поруку: "Систем не може да нађе похађања по задатој вредности". Прекида се извршење сценарија. (ИА)

8.1 Уколико систем не може да учита похађање он приказује директору поруку: "Систем не може да учита похађање. (ИА)

4.2 Фаза анализе

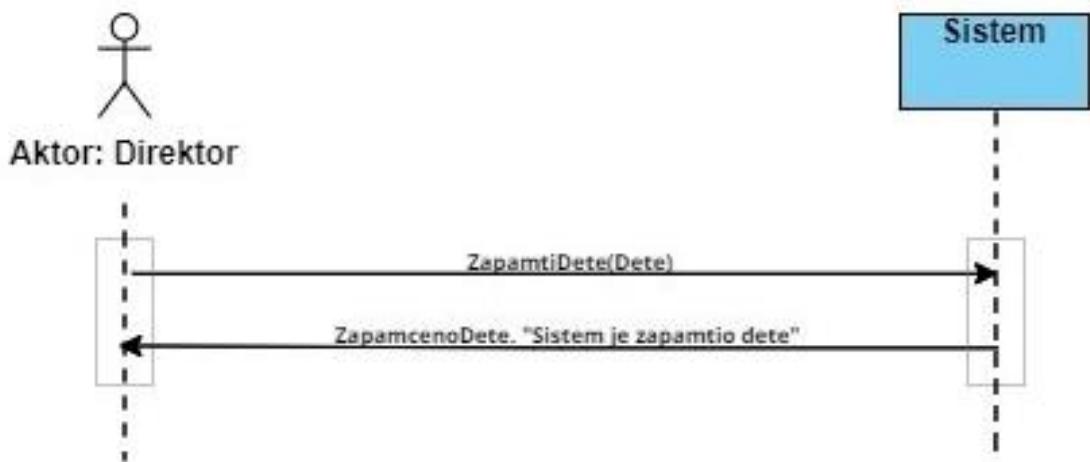
У фази анализе разматрамо логичку структуру и понашање софтверског система, дефинишући његову пословну логику. Најпре описујемо понашање помоћу дијаграма секвенци и системских операција, а затим приказујемо структуру кроз концептуални и релациони модел.[2]

4.2.1 Понашање софтверског система- Системски дијаграм секвенци

Понашање софтверског система приказујемо кроз системске дијаграме секвенци за сваки случај коришћења идентификован у фази прикупљања захтева. Дијаграми моделују интеракције између актора и система, пратећи активности у одређеном редоследу. Приказују се само АПСО и ИА акције.[2]

ДС1: Дијаграм секвенци случаја коришћења - Креирање детета

1. **Директор позива систем да запамти податке о детету.** (АПСО)
2. **Систем приказује директору запамћено дете и поруку:** "Систем је запамтио дете". (ИА)



Слика 42. ДС1 - Основни сценарио

Алтернативна сценарија

- 2.1 Уколико **систем** не може да запамти податке о **детету** он приказује **директору** поруку "**Систем не може да запамти дете**". (ИА)



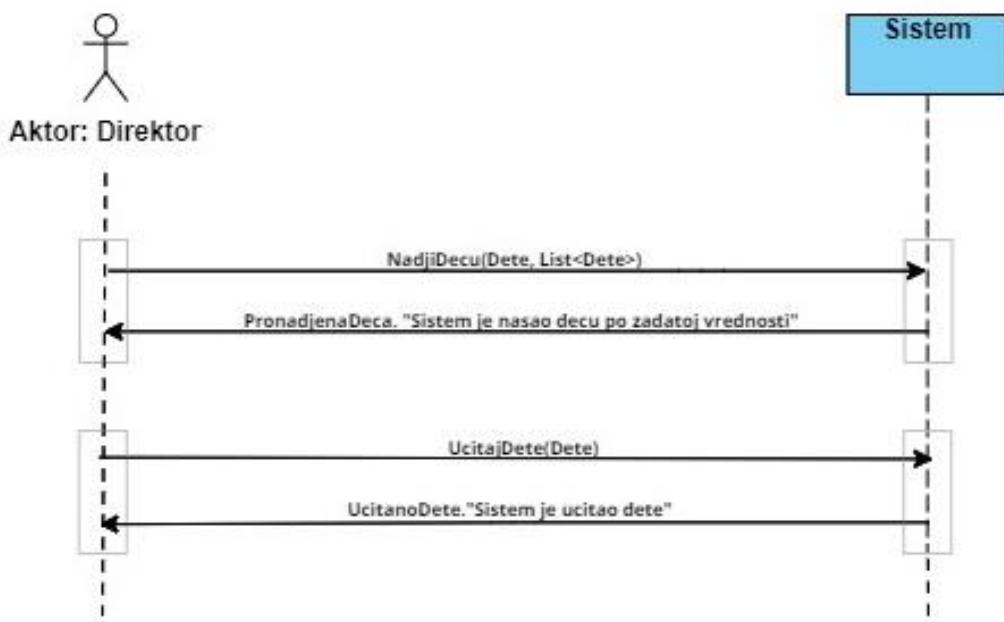
Слика 43. ДС1 - Алтернативни сценарио 1

Са наведених секвенцних дијаграма уочава се 1 системска операција:

1. Signal **ZapamtiDete(Dete)**

ДС2: Дијаграм секвенци случаја коришћења - Претраживање детета

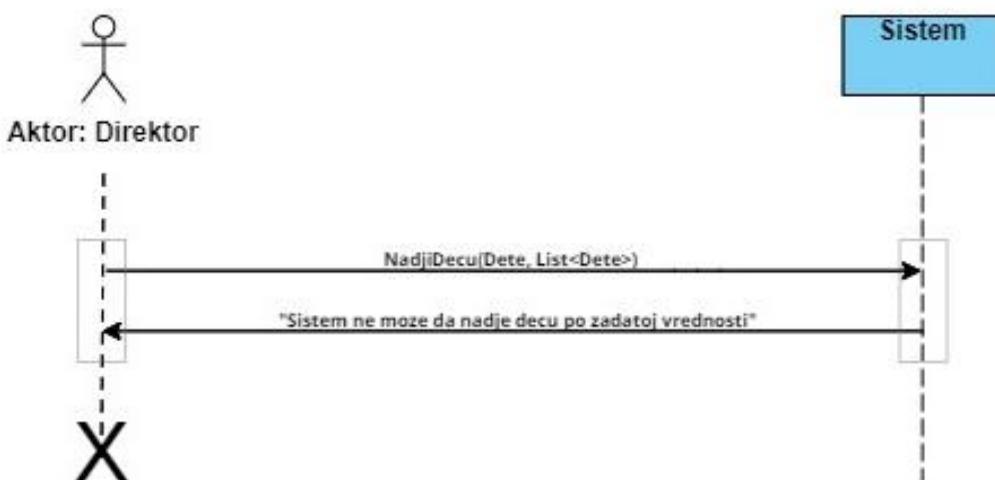
1. **Директор** позива **систем** да нађе **децу** по задатој вредности. (АПСО)
2. **Систем** приказује **директору** податке о **деци** и поруку: "Систем је нашао децу по задатој вредности". (ИА)
3. **Директор** позива **систем** да учита **дете**. (АПСО)
4. **Систем** приказује **директору** податке о **детету** и поруку: "Систем је учитао дете". (ИА)



Слика 44. ДС2 - Основни сценарио

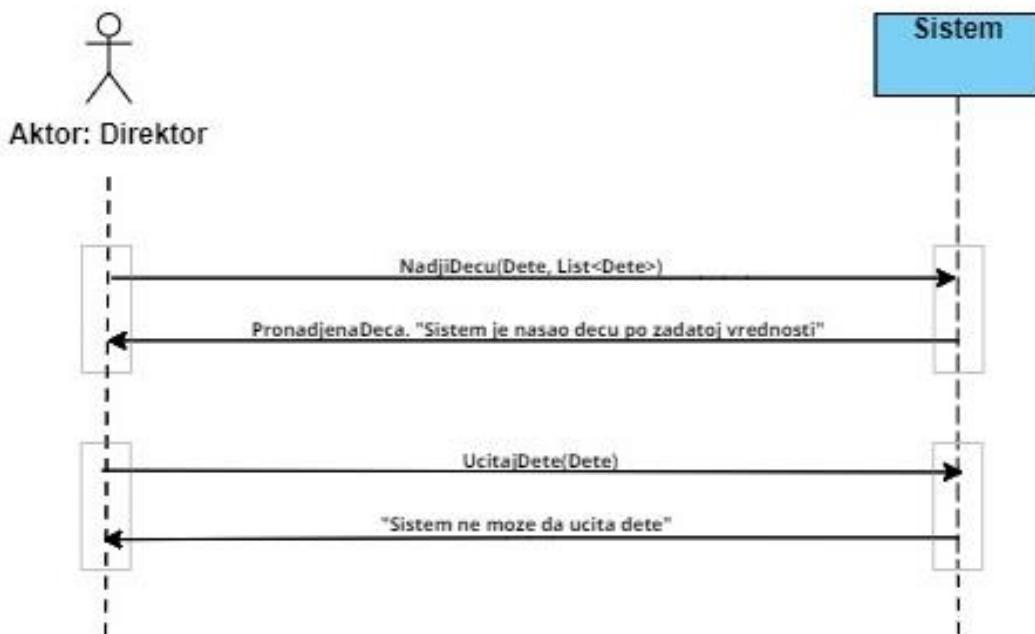
Алтернативна сценарија

2.1 Уколико **систем** не може да нађе **децу** он приказује **директору** поруку: "Систем не може да нађе **децу** по задатој вредности". Прекида се извршење сценарија. (ИА)



Слика 45. ДС2 - Алтернативни сценарио 1

4.1 Уколико **систем** не може да учита **дете** он приказује **директору** поруку: “**Систем** не може да учита **дете**”. (ИА)



Слика 46. ДС2 - Алтернативни сценарио 2

Са наведених секвенцних дијаграма уочавају се 2 системске операције:

1. Signal **Nadjidecu(Dete, List<Dete>)**
2. Signal **UcitajDete(Dete)**

ДСЗ: Дијаграм секвенци случаја коришћења - Креирање родитеља/старатеља

1. **Директор** позива **систем** да запамти податке о **родитељу/старатељу**. (АПСО)
2. **Систем приказује директору** запамћеног **родитеља/старатеља** и поруку: "Систем је запамтио **родитеља/старатеља**". (ИА)



Слика 47. ДСЗ - Основни сценарио

Алтернативна сценарија

- 2.1 Уколико **систем** не може да запамти податке о **родитељу/старатељу** он приказује **директору** поруку "Систем не може да запамти **родитеља/старатеља**". (ИА)



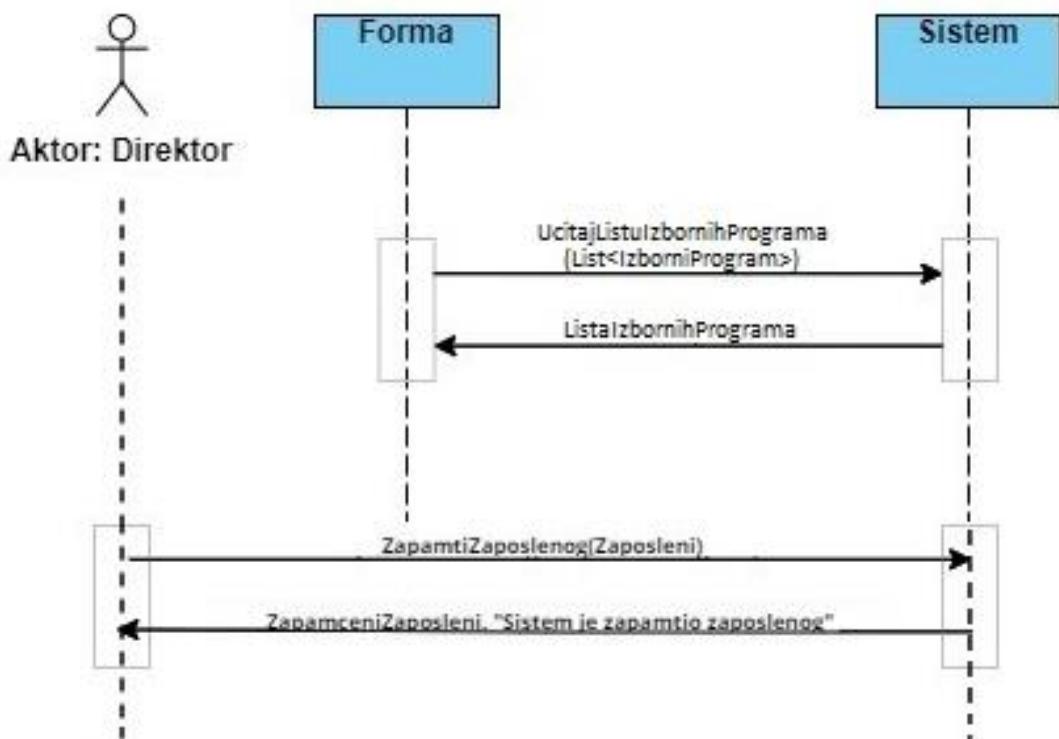
Слика 48. ДСЗ - Алтернативни сценарио 1

Са наведених секвенцних дијаграма уочава се 1 системска операција:

1. Signal **ZapamtiRoditeljaStaratelja(RoditeljStaratelj)**

ДС4: Дијаграм секвенци случаја коришћења - Креирање запосленог

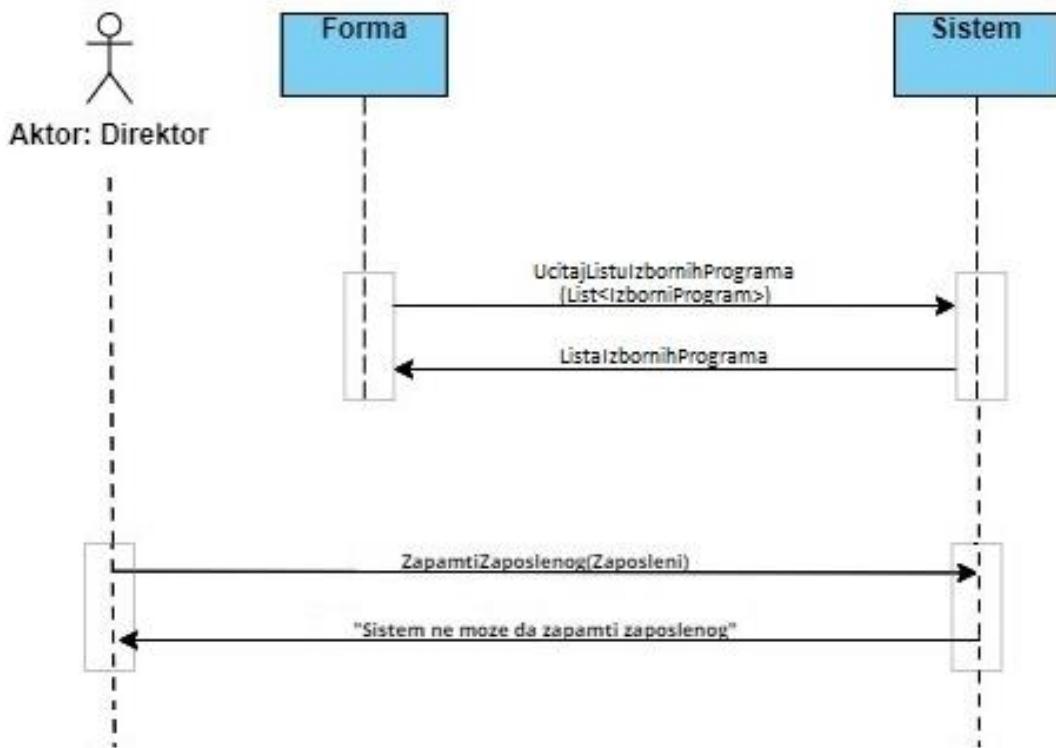
1. **Форма** позива **систем** да учита листу изборних програма. (АПСО)
2. **Систем** враћа **форми** листу изборних програма.(ИА)
3. **Директор** позива **систем** да запамти податке о **запосленом**. (АПСО)
4. **Систем** приказује **директору** запамћеног **запосленог** и поруку: “**Систем** је запамтио **запосленог**“. (ИА)



Слика 49. ДС4 - Основни сценарио

Алтернативна сценарија

4.1 Уколико **систем** не може да запамти податке о **запосленом** он приказује **директору** поруку “**Систем** не може да запамти **запосленог**”. (ИА)



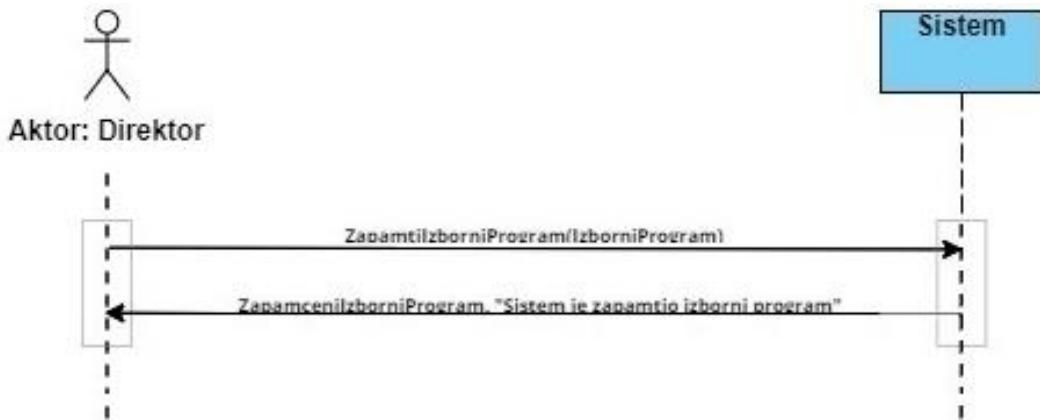
Слика 50. ДС4 - Алтернативни сценарио 1

Са наведених секвенцних дијаграма уочавају се 2 системске операције:

1. Signal **UcitajListulIzbornihPrograma(List<IzborniProgram>)**
2. Signal **ZapamtiZaposlenog(Zaposleni)**

ДС5: Дијаграм секвенци случаја коришћења - Креирање изборног програма

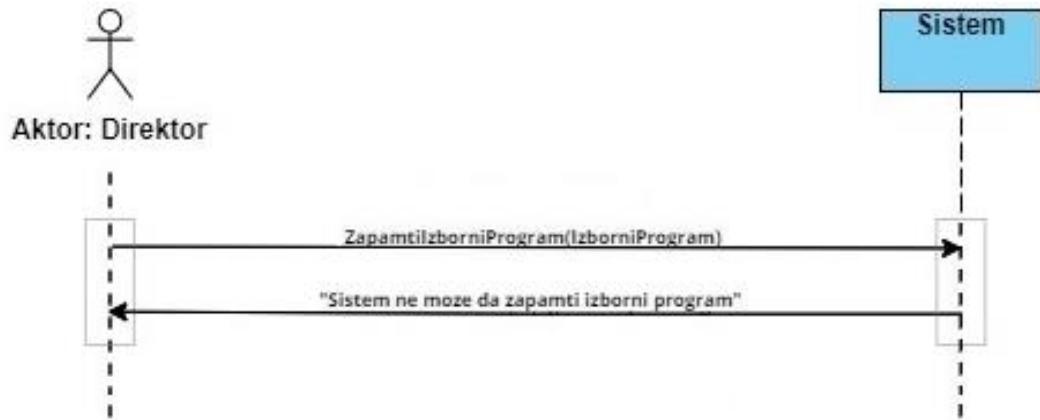
1. **Директор** позива **систем** да запамти податке о **изборном програму**. (АПСО)
2. **Систем** приказује **директору** запамћени изборни програм и поруку: “**Систем** је запамтио **изборни програм**“. (ИА)



Слика 51. ДС5 - Основни сценарио

Алтернативна сценарија

- 2.1 Уколико **систем** не може да запамти податке о **изборном програму** он приказује **директору** поруку “**Систем** не може да запамти **изборни програм**”. (ИА)



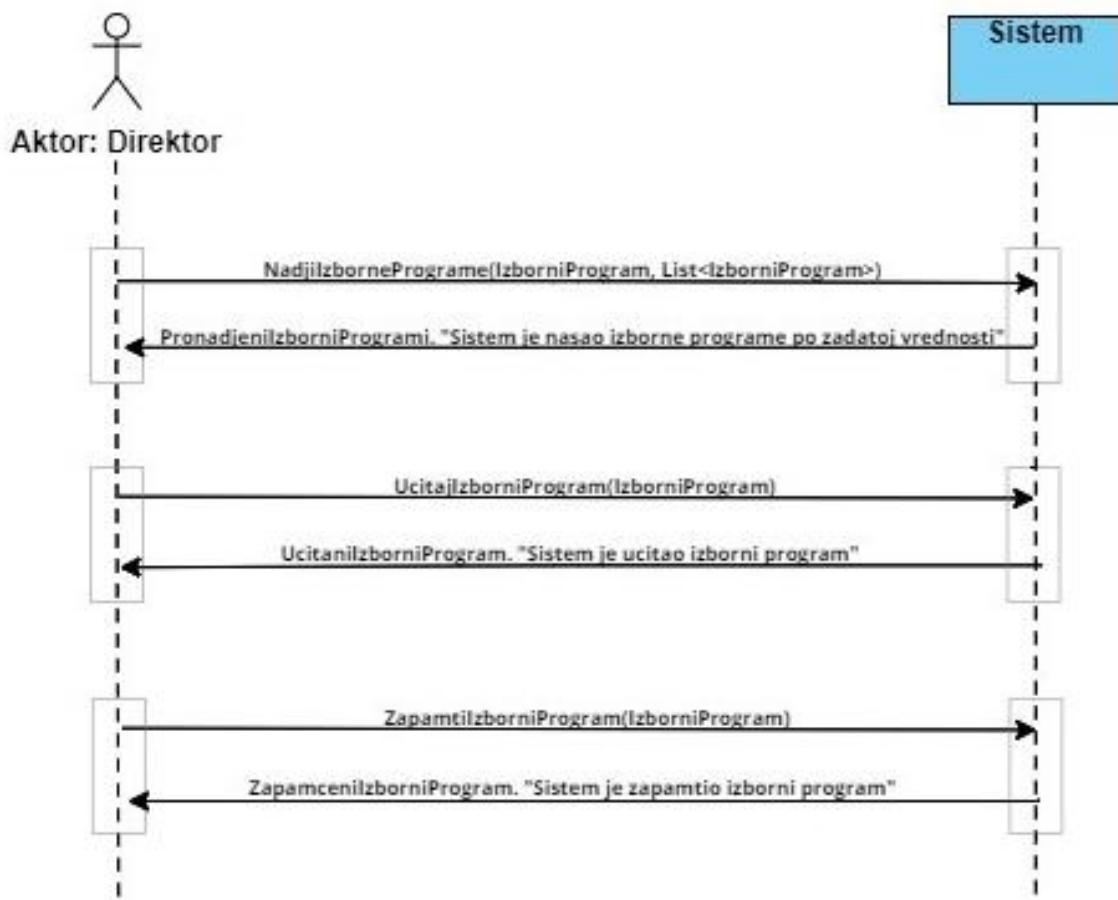
Слика 52. ДС5 - Алтернативни сценарио 1

Са наведених секвенцних дијаграма уочава се 1 системска операција:

1. Signal **ZapamtilzborniProgram(IzborniProgram)**

ДС6: Дијаграм секвенци случаја коришћења - Измена изборног програма

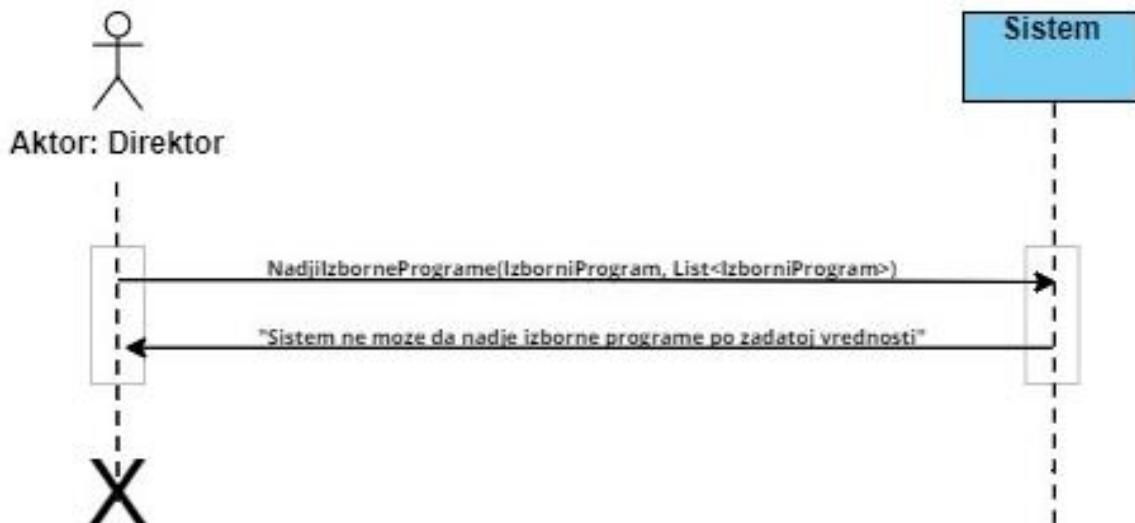
1. **Власник** **позива** **систем** да нађе изборне програме по задатој вредности. (АПСО)
2. **Систем** **приказује** **власнику** изборне програме и поруку: "Систем је нашао изборне програме по задатој вредности". (ИА)
3. **Власник** **позива** **систем** да учита изборни програм. (АПСО)
4. **Систем** **приказује** **власнику** податке о изборном програму и поруку: "Систем је учитао изборни програм". (ИА)
5. **Власник** **позива** **систем** да запамти податке о изборном програму. (АПСО)
6. **Систем** **приказује** **власнику** запамћени изборни програм и поруку: "Систем је запамтио изборни програм." (ИА)



Слика 53. ДС6 - Основни сценарио

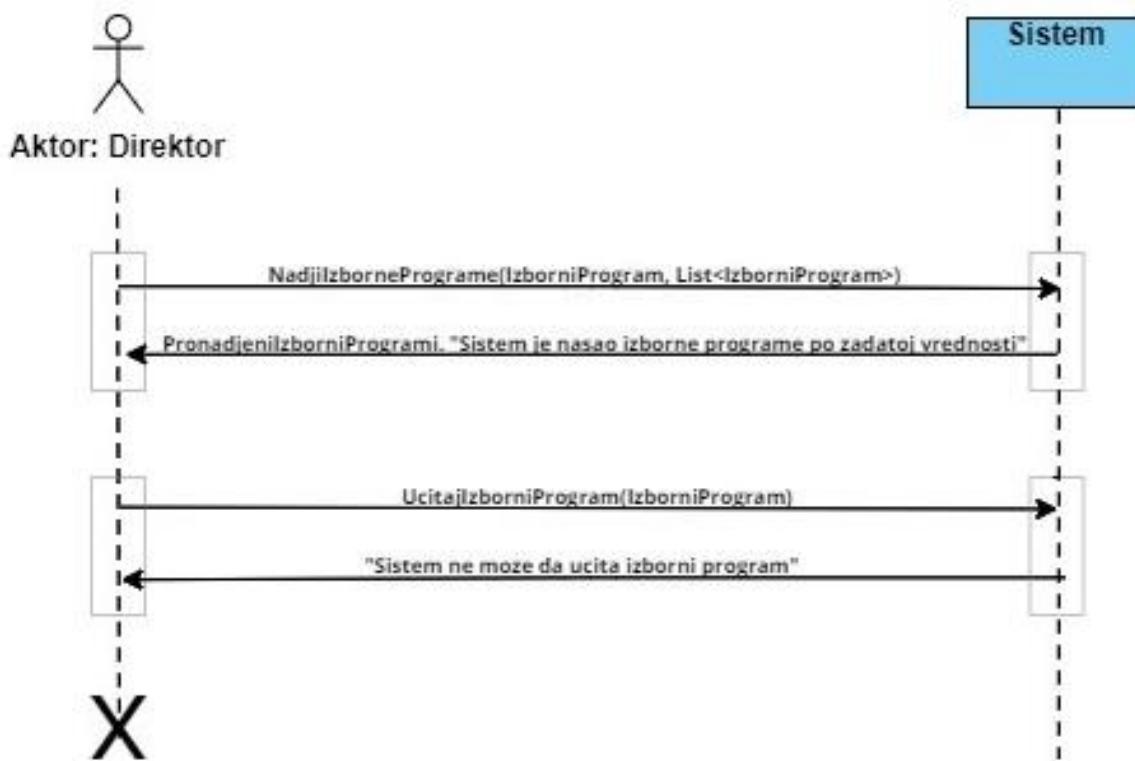
Алтернативна сценарија

2.1 Уколико **систем** не може да нађе **изборне програме** он приказује **власнику** поруку: "Систем не може да нађе изборне програме по задатој вредности". Прекида се извршење сценарија. (ИА)



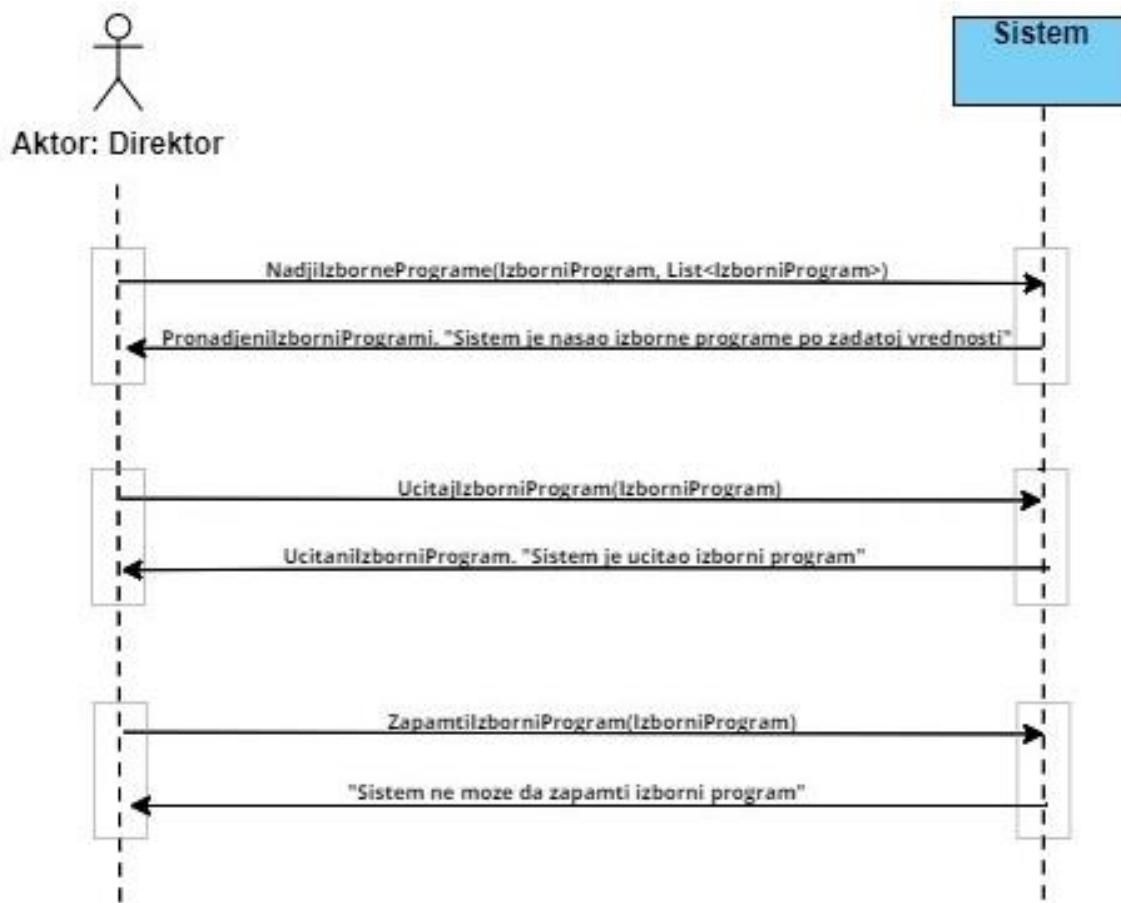
Слика 54. ДС6 - Алтернативни сценарио 1

4.1 Уколико **систем** не може да учита **изборни програм** он приказује **власнику** поруку: "Систем не може да учита изборни програм". Прекида се извршење сценарија. (ИА)



Слика 55. ДС6 - Алтернативни сценарио 2

6.1 Уколико **систем** не може да запамти податке о изборном програму он приказује **власнику** поруку “**Систем** не може да запамти изборни програм”. (ИА)



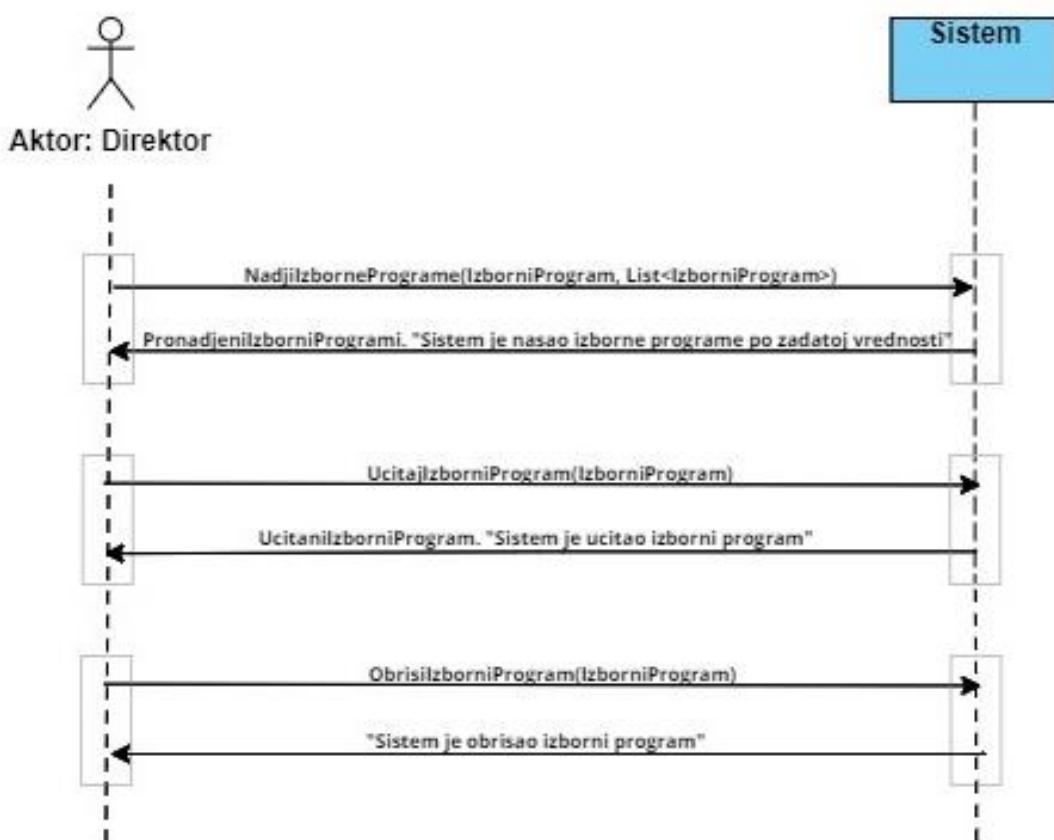
Слика 56. ДС6 - Алтернативни сценарио 3

Са наведених секвенцних дијаграма уочавају се 3 системске операције:

1. Signal **NadjilzbornePrograme(IzborniProgram, List<IzborniProgram>)**
2. Signal **UcitajIzborniProgram(IzborniProgram)**
3. Signal **ZapamtilzborniProgram(IzborniProgram)**

ДС7: Дијаграм секвенци случаја коришћења - Брисање изборног програма

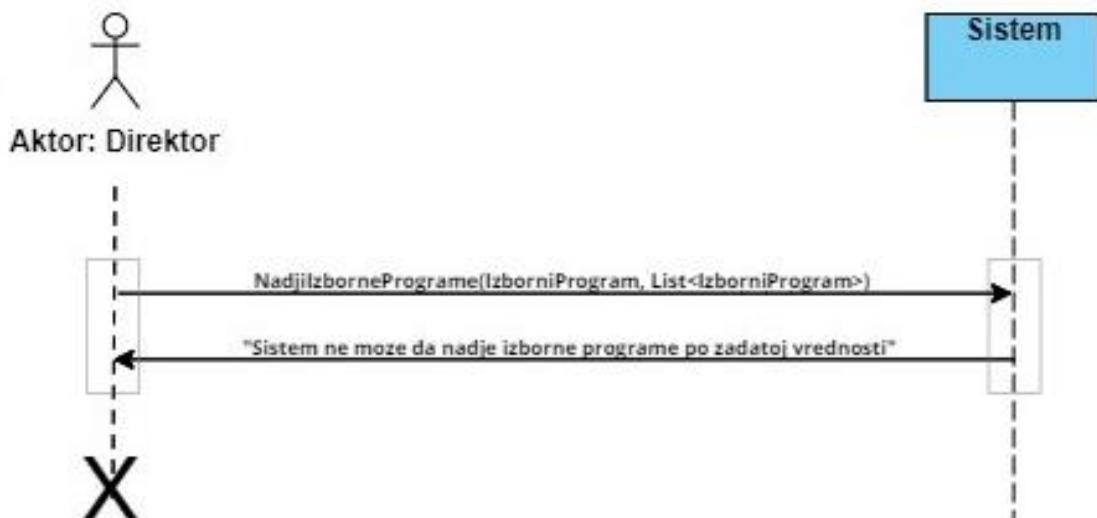
1. **Власник** **позива** **систем** да нађе изборне програме по задатој вредности. (АПСО)
2. **Систем** **приказује** **власнику** изборне програме и поруку: "Систем је нашао изборне програме по задатој вредности". (ИА)
3. **Власник** **позива** **систем** да учита изборни програм. (АПСО)
4. **Систем** **приказује** **власнику** податке о изборном програму и поруку: "Систем је учитао изборни програм". (ИА)
5. **Власник** **позива** **систем** да обрише изборни програм. (АПСО)
6. **Систем** **приказује** **власнику** поруку: "Систем је обрисао изборни програм." (ИА)



Слика 57. ДС7 - Основни сценарио

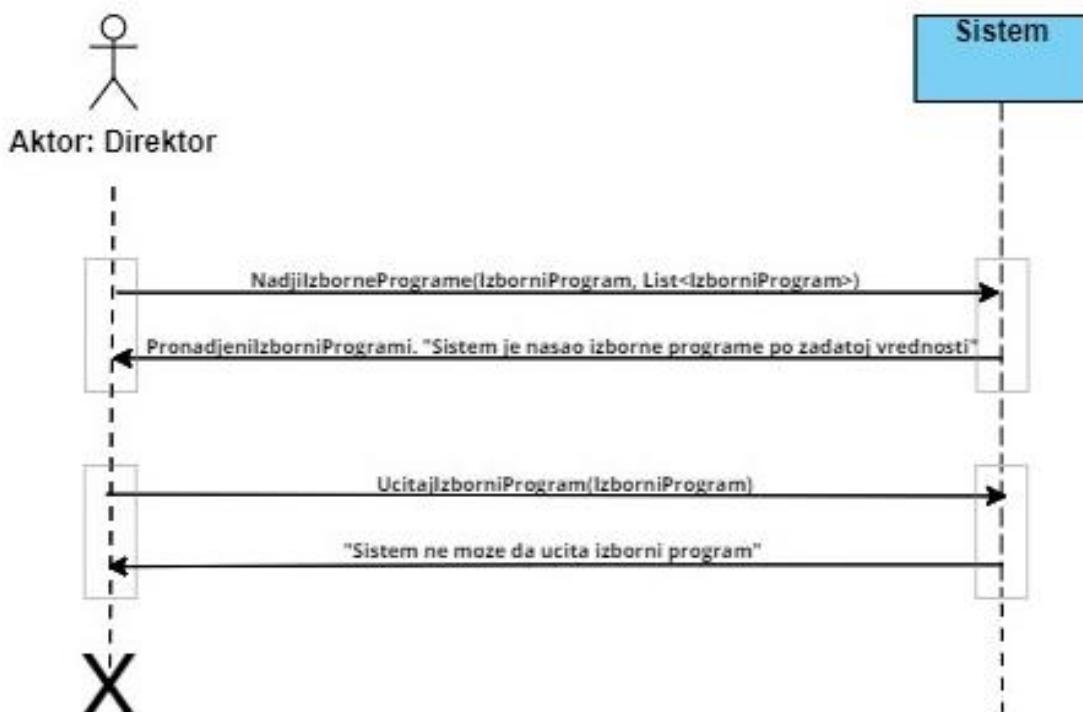
Алтернативна сценарија

2.1 Уколико **систем** не може да нађе **изборне програме** он приказује **власнику** поруку: "Систем не може да нађе изборне програме по задатој вредности". Прекида се извршење сценарија. (ИА)



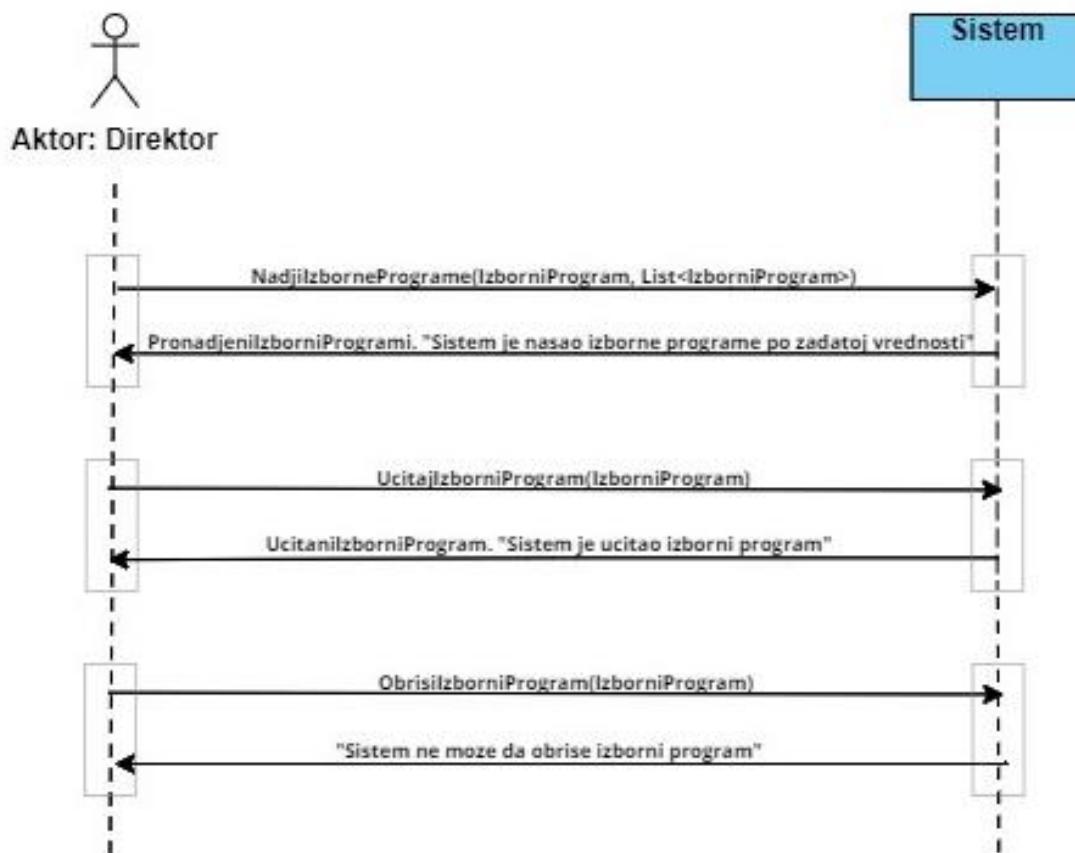
Слика 58. DC7 - Алтернативни сценарио 1

4.1 Уколико **систем** не може да учита **изборни програм** он приказује **власнику** поруку: "Систем не може да учита изборни програм". Прекида се извршење сценарија. (ИА)



Слика 59. DC7 - Алтернативни сценарио 2

6.1 Уколико **систем** не може да избрише **изборни програм** он приказује **власнику** поруку “Систем не може да обрише изборни програм”. (ИА)



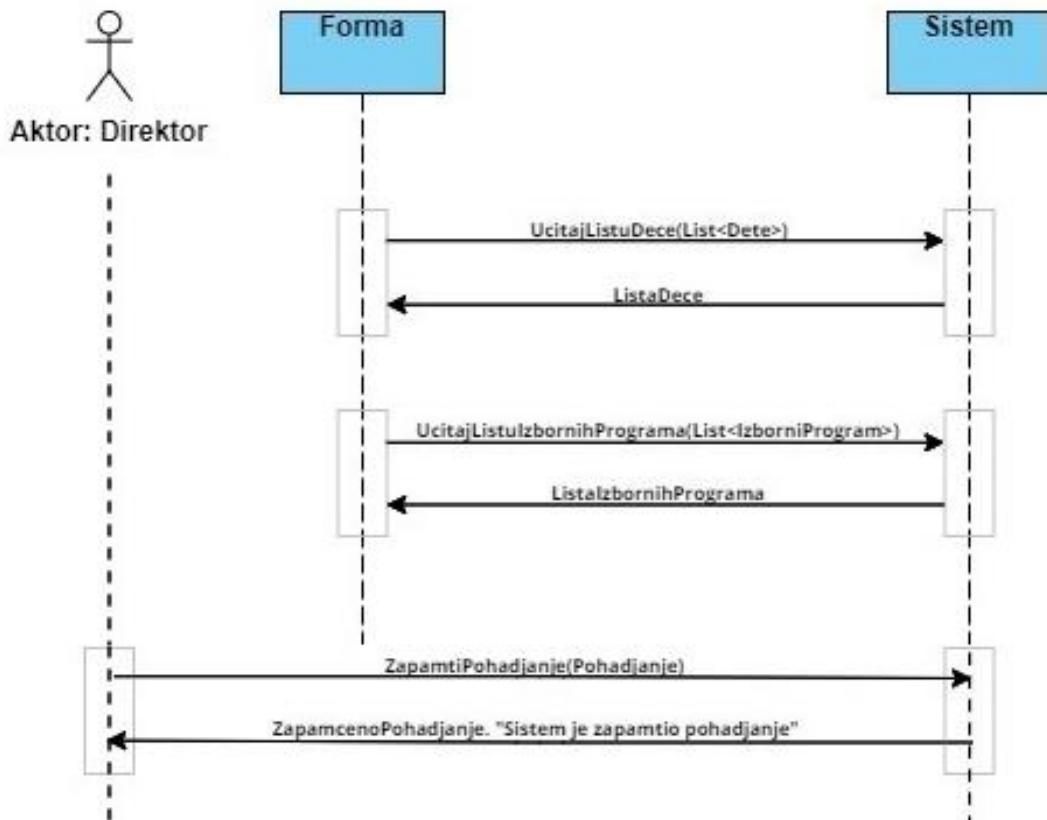
Слика 60. DC7 - Алтернативни сценарио 3

Са наведених секвенцних дијаграма уочавају се 3 системске операције:

1. Signal **NadjilzbornePrograme(IzborniProgram, List<IzborniProgram>)**
2. Signal **UcitajIzborniProgram(IzborniProgram)**
3. Signal **ObrisilzborniProgram(IzborniProgram)**

ДС8: Дијаграм секвенци случаја коришћења - Креирање похађања

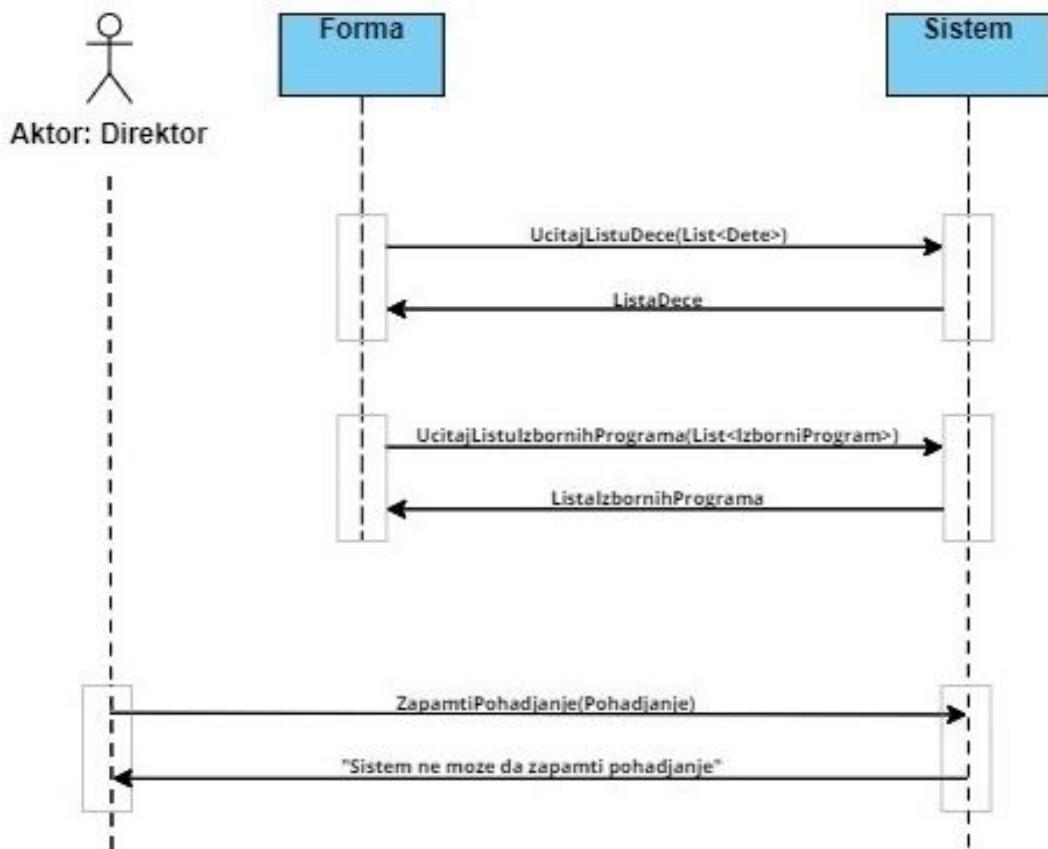
1. **Форма** позива **систем** да учита листу деце. (АПСО)
2. **Систем враћа** **форми** листу деце.(ИА)
3. **Форма** позива **систем** да учита листу изборних програма. (АПСО)
4. **Систем враћа** **форми** листу изборних програма.(ИА)
5. **Директор** позива **систем** да запамти податке о **пohaђању**. (АПСО)
6. **Систем приказује** **директору** запамћено **пohaђање** и поруку: “**Систем** је запамтио **пohaђање**“. (ИА)



Слика 61. ДС8 - Основни сценарио

Алтернативна сценарија

6.1 Уколико **систем** не може да запамти податке о **похађању** он приказује **директору** поруку “**Систем** не може да запамти **похађање**”. (ИА)



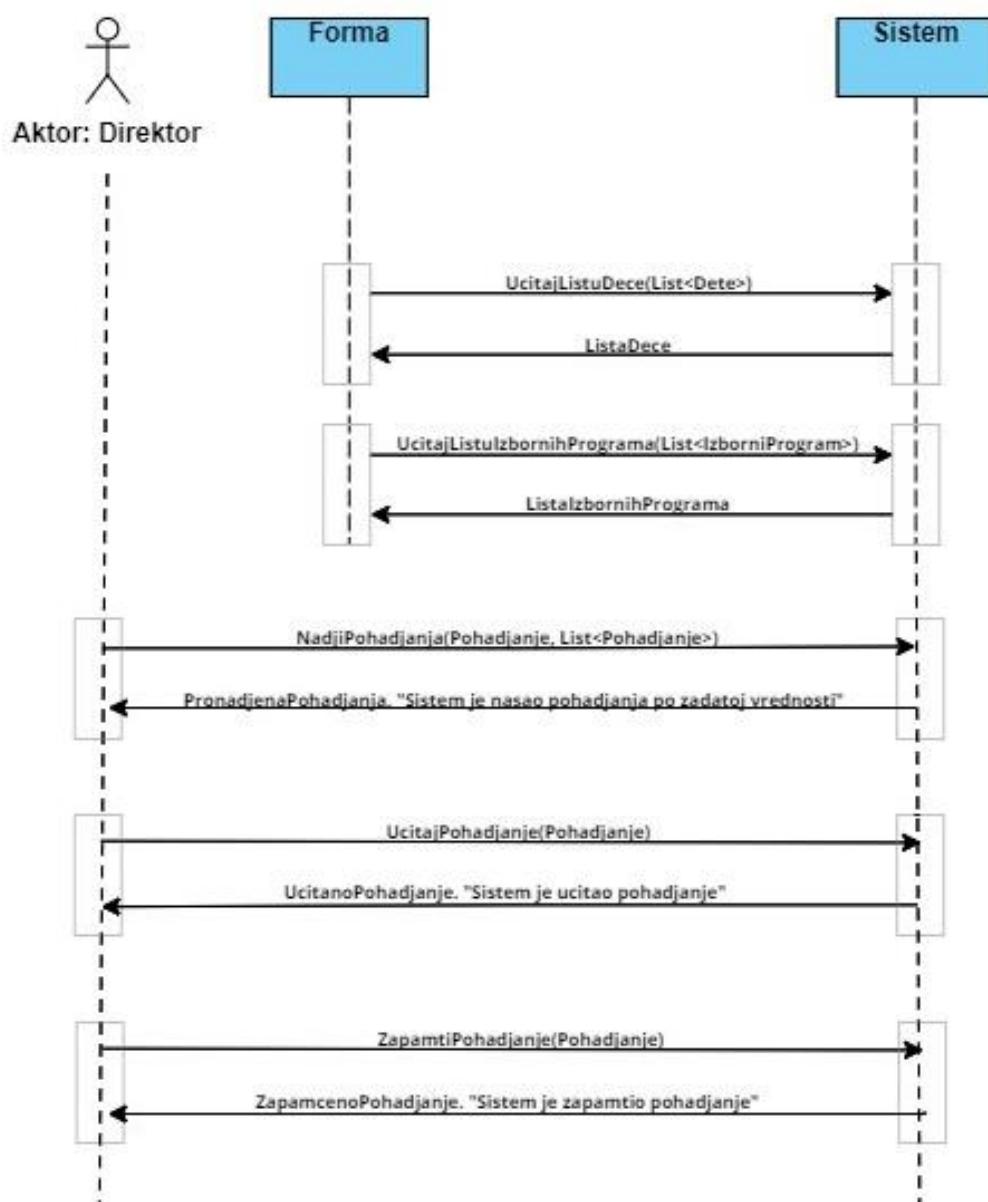
Слика 62. ДС8 - Алтернативни сценарио 1

Са наведених секвенцних дијаграма уочавају се 3 системске операције:

1. Signal **UcitajListuDece(List<Dete>)**
2. Signal **UcitajListulzbornihPrograma(List<IzborniProgram>)**
3. Signal **ZapamtiPohadjanje(Pohadjanje)**

ДС9: Дијаграм секвенци случаја коришћења - Измена похађања

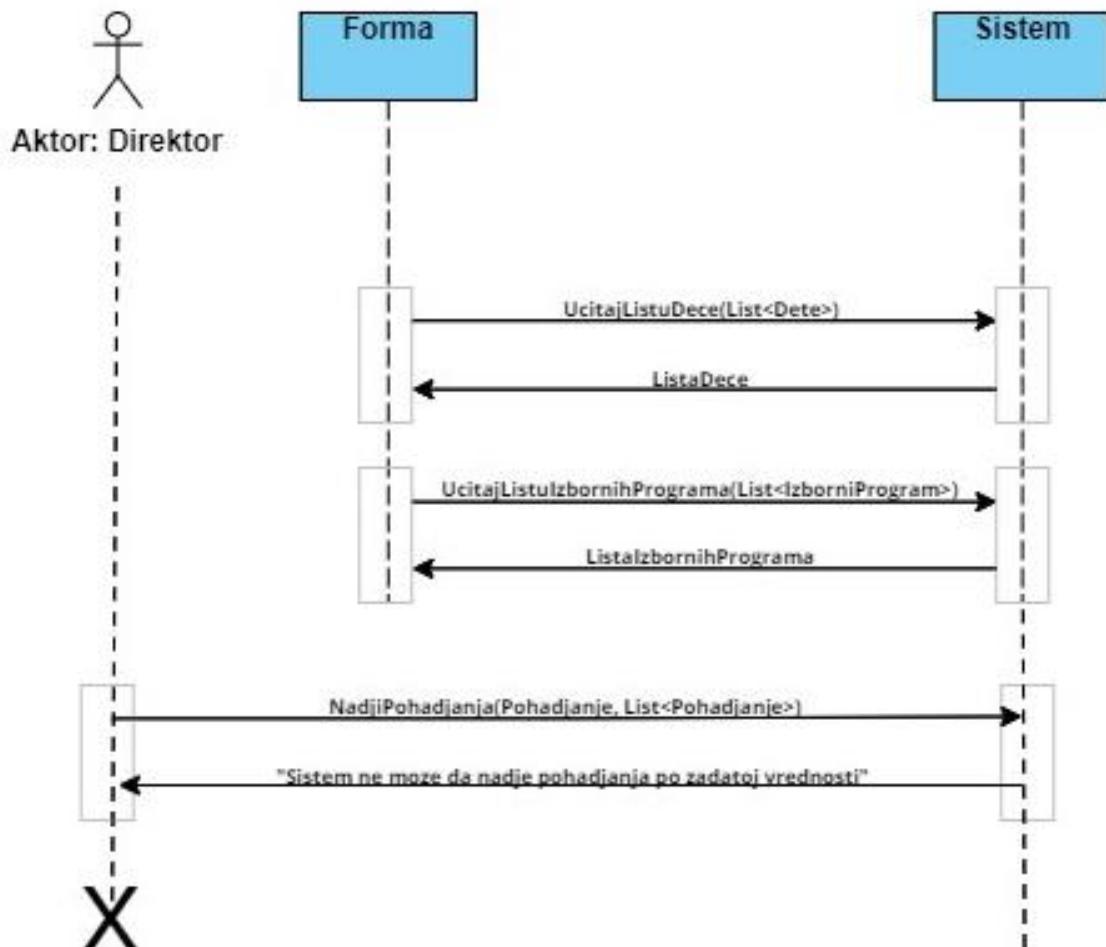
1. **Форма** позива **систем** да учита листу деце. (АПСО)
2. **Систем** враћа **форми** листу деце. (ИА)
3. **Форма** позива **систем** да учита листу изборних програма. (АПСО)
4. **Систем** враћа **форми** листу изборних програма. (ИА)
5. **Власник** позива **систем** да нађе похађања по задатој вредности. (АПСО)
6. **Систем** приказује **власнику** похађања и поруку: "Систем је нашао похађања по задатој вредности". (ИА)
7. **Власник** позива **систем** да учита похађање. (АПСО)
8. **Систем** приказује **власнику** податке о похађању и поруку: "Систем је учитао похађање". (ИА)
9. **Власник** позива **систем** да запамти податке о похађању. (АПСО)
10. **Систем** приказује **власнику** запамћено похађање и поруку: "Систем је запамтио похађање." (ИА)



Слика 63. ДС9 - Основни сценарио

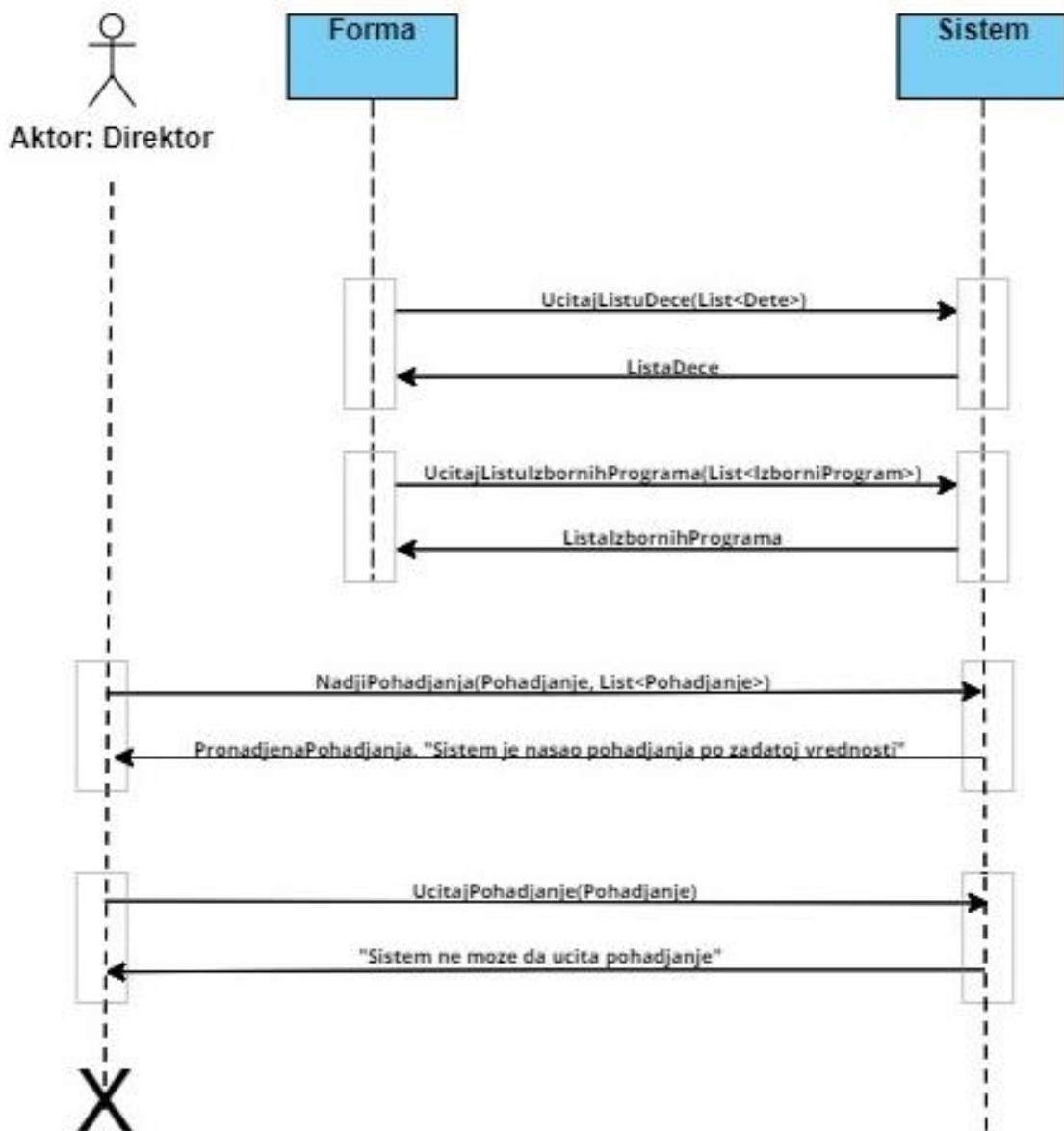
Алтернативна сценарија

6.1 Уколико **систем** не може да нађе **похађања** он приказује **власнику** поруку: “**Систем** не може да нађе **похађања** по задатој вредности”. Прекида се извршење сценарија. (ИА)



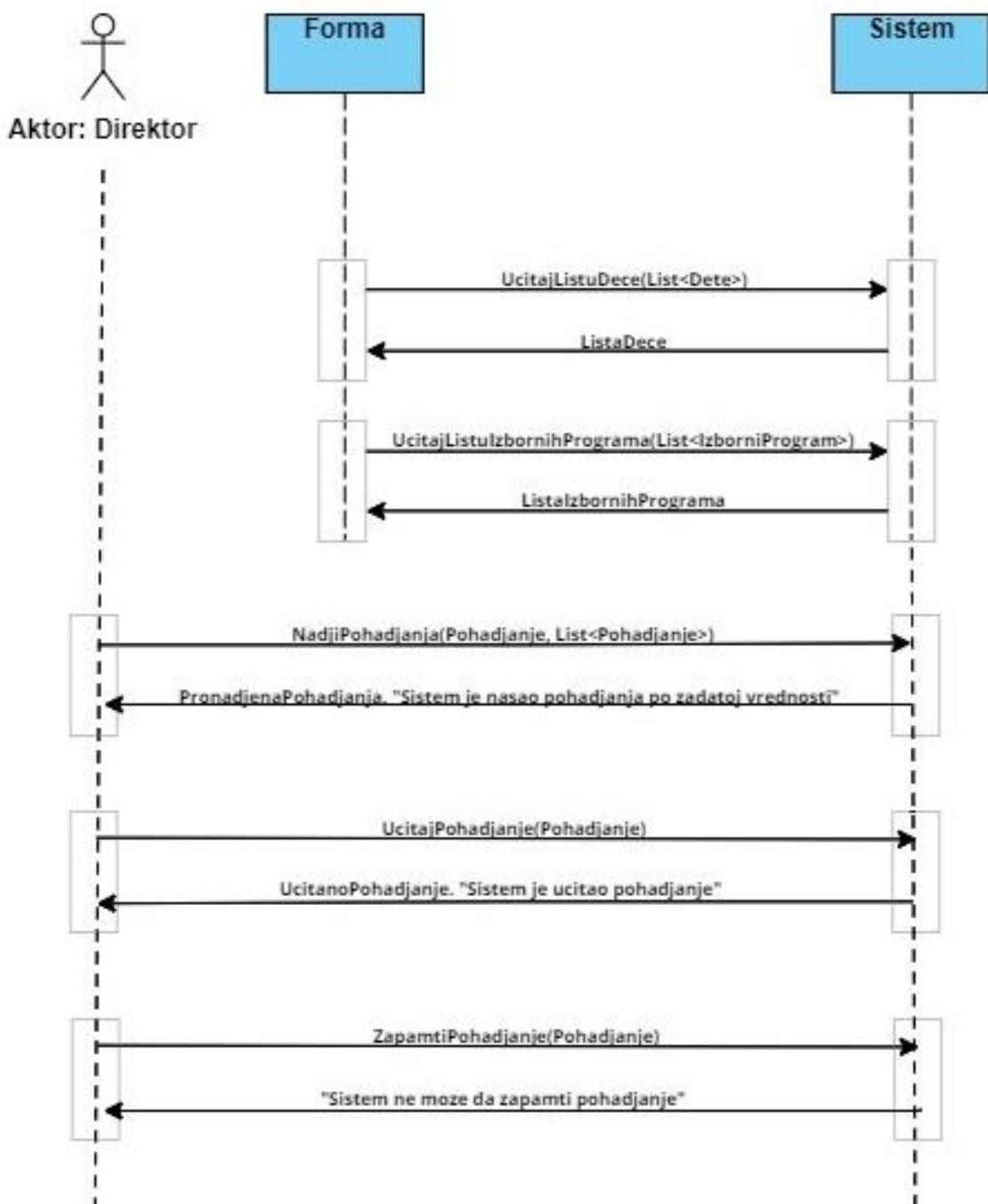
Слика 64. ДС9 - Алтернативни сценарио 1

8.1 Уколико **систем** не може да учита **похађање** он приказује **власнику** поруку: "Систем не може да учита похађање". Прекида се извршење сценарија. (ИА)



Слика 65. ДС9 - Алтернативни сценарио 2

10.1 Уколико **систем** не може да запамти податке о **похађању** он приказује **власнику** поруку “**Систем** не може да запамти похађање”. (ИА)



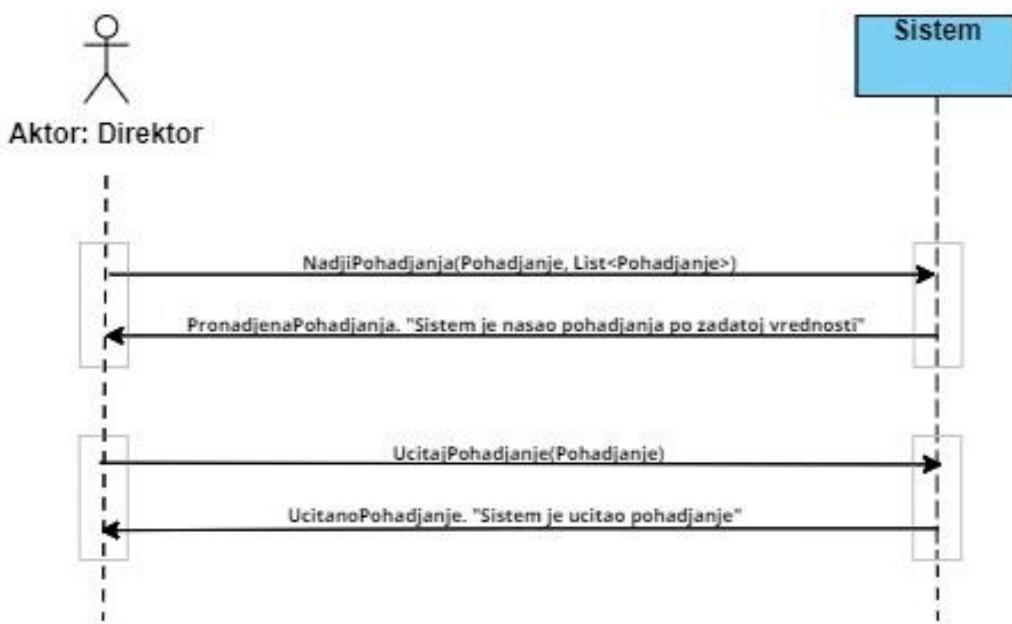
Слика 66. ДС9 - Алтернативни сценарио 3

Са наведених секвенцних дијаграма уочавају се 5 системских операција:

1. Signal **UcitajListuDece(List<Dete>)**
2. Signal **UcitajListulzbornihPrograma (List<IzborniProgram>)**
3. Signal **NadjiPohadjanja(Pohadjanje, List<Pohadjanje>)**
4. Signal **UcitajPohadjanje(Pohadjanje)**
5. Signal **ZapamtiPohadjanje(Pohadjanje)**

ДС10: Дијаграм секвенци случаја коришћења - Претраживање похађања

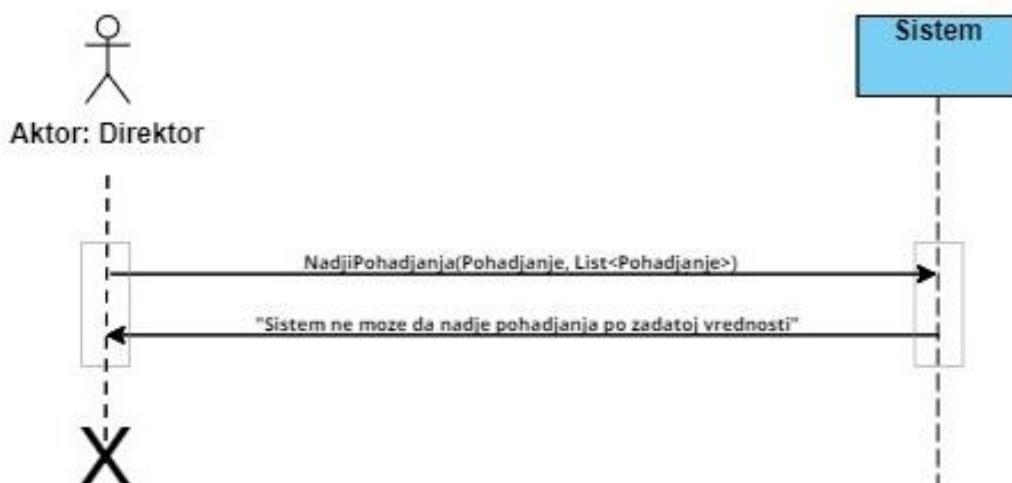
1. **Директор** позива **систем** да нађе похађања по задатој вредности. (АПСО)
2. **Систем** приказује **директору** податке о похађањима и поруку: "Систем је нашао похађања по задатој вредности". (ИА)
3. **Директор** позива **систем** да учита похађање. (АПСО)
4. **Систем** приказује **директору** податке о похађању и поруку: "Систем је учитао похађање". (ИА)



Слика 67. ДС10 - Основни сценарио

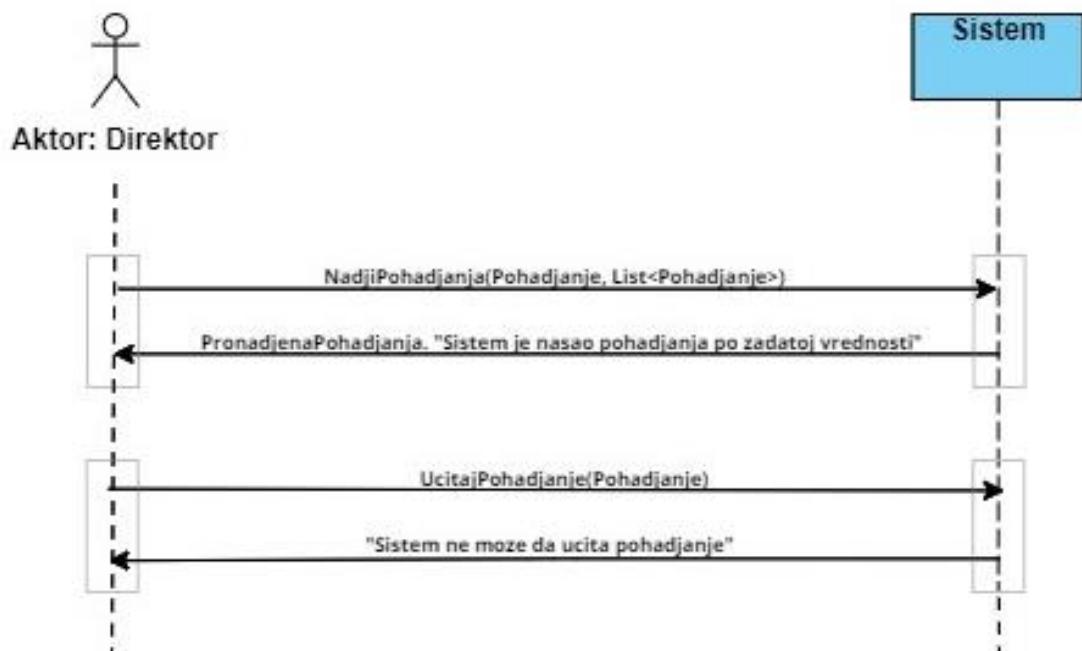
Алтернативна сценарија

2.1 Уколико **систем** не може да нађе похађања он приказује **директору** поруку: "Систем не може да нађе похађања по задатој вредности". Прекида се извршење сценарија. (ИА)



Слика 68. ДС10 - Алтернативни сценарио 1

4.1 Уколико **систем** не може да учита **похађање** он приказује **директору** поруку: “**Систем** не може да учита **похађање**”. (ИА)



Слика 69. ДС10 - Алтернативни сценарио 2

Са наведених секвенцних дијаграма уочавају се 2 системске операције:

1. Signal **NadjiPohadjanja(Pohadjanje, List<Pohadjanje>)**
2. Signal **UcitajPohadjanje(Pohadjanje)**

Како резултат анализе сценарија добијено је 14 системских операција које треба пројектовати:

1. Signal **ZapamtiDete(Dete)**
2. Signal **NadjiDecu(Dete, List<Dete>)**
3. Signal **UcitajDete(Dete)**
4. Signal **ZapamtiRoditeljaStaratelja(RoditeljStaratelj)**
5. Signal **ZapamtiZaposlenog(Zaposleni)**
6. Signal **UcitajListulzbornihPrograma(List<IzborniProgram>)**
7. Signal **ZapamtilzborniProgram(IzborniProgram)**
8. Signal **NadjilzbornePrograme(IzborniProgram, List<IzborniProgram>)**
9. Signal **UcitajIzborniProgram(IzborniProgram)**
10. Signal **ObrisilzborniProgram(IzborniProgram)**
11. Signal **UcitajListuDece(List<Dete>)**
12. Signal **ZapamtiPohadjanje(Pohadjanje)**
13. Signal **NadjiPohadjanja(Pohadjanje, List<Pohadjanje>)**
14. Signal **UcitajPohadjanje(Pohadjanje)**

4.2.2 Понашање софтверског система – Дефинисање уговора о системским операцијама

Уговор УГ1: ЗапамтиДете

Операција: ZaramtiDete(Dete): signal;

Веза са СК: СК1

Предуслови: Вредносна и структурна ограничења над објектом Дете морају бити задовољена

Постуслови: Дете је запамћено

Уговор УГ2: НађиДецу

Операција: NadjiDecu(Dete, List<Dete>): signal;

Веза са СК: СК2

Предуслови: Учитана је листа деце

Постуслови: Учитана је листа деце која задовољавају задату вредност

Уговор УГ3: УчитајДете

Операција: UcitajDete(Dete) : signal;

Веза са СК: СК2

Предуслови: /

Постуслови: Учитано је одабрано дете

Уговор УГ4: ЗапамтиРодитељаСтаратеља

Операција: ZaramtiRoditeljaStaratelja(RoditeljStaratelj) : signal;

Веза са СК: СК3

Предуслови: Вредносна и структурна ограничења над објектом РодитељСтаратељ морају бити задовољена

Постуслови: РодитељСтаратељ је запамћен

Уговор УГ5: ЗапамтиЗапосленог

Операција: ZaramtiZaposlenog(Zaposleni) : signal;

Веза са СК: СК4

Предуслови: Вредносна и структурна ограничења над објектом Запослени морају бити задовољена

Постуслови: Запослени је запамћен

Уговор УГ6: УчитајИзборнеПрограме

Операција: UcitajListulizbornihPrograma(List<IzborniProgram>): signal;

Веза са СК: СК4, СК8

Предуслови: /

Постуслови: Учитана је листа Изборних програма

Уговор УГ7: ЗапамтиИзборниПрограм

Операција: ZaramtilzborniProgram(IzborniProgram): signal;

Веза са СК: СК5, СК6

Предуслови: Вредносна и структурна ограничења над објектом Изборни програм морају бити задовољена

Постуслови: Изборни програм је запамћен

Уговор УГ8: НађиИзборнеПрограме

Операција: NadjiIzbornePrograme(IzborniProgram, List<IzborniProgram>): signal;

Веза са СК: СК6, СК7

Предуслови: Учитана је листа изборних програма

Постуслови: Учитана је листа изборних програма који задовољавају задату вредност

Уговор УГ9: УчитајИзборниПрограм

Операција: UcitajIzborniProgram(IzborniProgram): signal;

Веза са СК: СК6, СК7

Предуслови: /

Постуслови: Учитан је одабрани изборни програм

Уговор УГ10: ОбришиИзборниПрограм

Операција: ObrisIzborniProgram(IzborniProgram): signal;

Веза са СК: СК7

Предуслови: Структурна ограничења над објектом Изборни програм морају бити задовољена.

Постуслови: Изборни програм је обрисан

Уговор УГ11: УчитајДецу

Операција: UcitajListuDece(List<Dete>): signal;

Веза са СК: СК8, СК9

Предуслови: /

Постуслови: Учитана је листа деце

Уговор УГ12: ЗапамтиПохађање

Операција: ZaramtiPohadjanje(Pohadjanje): signal;

Веза са СК: СК8, СК9

Предуслови: Вредносна и структурна ограничења над објектом Похађање морају бити задовољена

Постуслови: Похађање је запамћено

Уговор УГ13: НађиПохађања

Операција: NadjiPohadjanja(Pohadjanje, List<Pohadjanje>): signal;

Веза са СК: СК9, СК10

Предуслови: Учитана је листа похађања

Постуслови: Учитана је листа похађања која задовољавају задату вредност

Уговор УГ14: УчитајПохађање

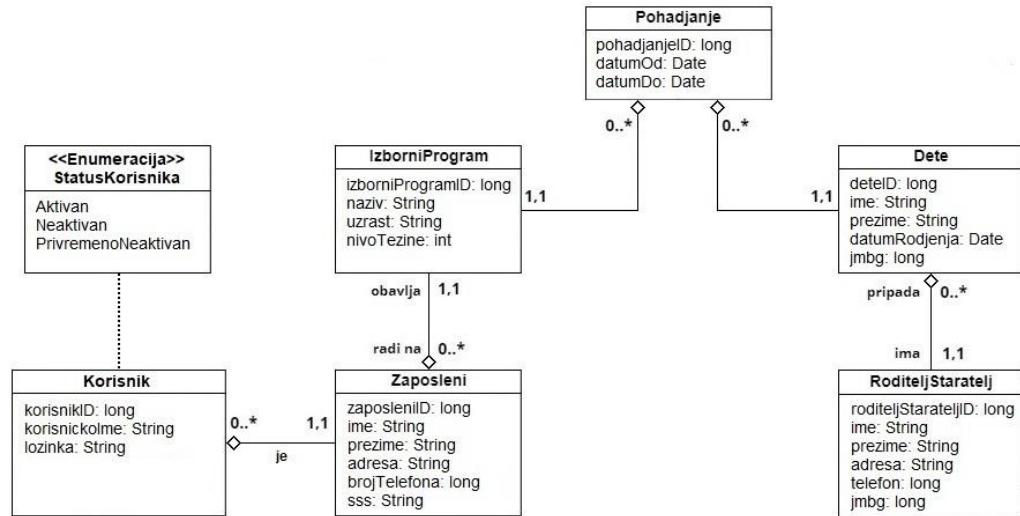
Операција: UcitajPohadjanje(Pohadjanje): signal;

Веза са СК: СК9, СК10

Предуслови: /

Постуслови: Учитано је одабрано похађање

4.2.3 Структура софтверског система – Концептуални (доменски) модел



Слика 70. Концептуални модел

4.2.4 Структура софтверског система – Релациони модел

Zaposleni(zaposleniID, ime, prezime, adresa, brojTelefona, sss, izborniProgramID)

RoditeljStaratelj(roditeljStarateljID, ime, prezime, adresa, telefon, jmbg)

Dete(deteID, ime, prezime, datumRodjenja, jmbg, roditeljStarateljID)

IzborniProgram(izborniProgramID, naziv, uzrast, nivoTezine)

Korisnik(korisnikID, korisnickolme, lozinka, zaposleniID)

Pohadjanje(pohadjanjeID, datumOd, datumDo, izborniProgramID, deteID)

| Табела Zaposleni | | Просторно вредносно ограничење | | Сложено вредносно ограничење | | Структурно ограничење |
|-------------------------|-------------|--------------------------------|-------------------|------------------------------------|-----------------------------------|--|
| Атрибути | Име | Тип атрибута | Вредност атрибута | Међувисиност атрибута једне табеле | Међувисиност атрибута више табела | INSERT RESTRICTED IzborniProgram UPDATE RESTRICTED IzborniProgram DELETE / |
| | zaposleniID | Long | Not null and >0 | | | |
| | ime | String | Not null | | | |
| | prezime | String | Not null | | | |
| | adresa | String | Not null | | | |
| | telefon | Long | Not null and >0 | | | |
| | sss | String | Not null | | | |

Табела 1. Ограниченија за табелу Запослени

| Табела RoditeljStaratelj | | Просторно вредносно ограничење | | Сложено вредносно ограничење | | Структурно ограничење |
|---------------------------------|---------------------|--------------------------------|-------------------|------------------------------------|-----------------------------------|---|
| Атрибути | Име | Тип атрибута | Вредност атрибута | Међувисиност атрибута једне табеле | Међувисиност атрибута више табела | INSERT/ UPDATE CASCADES Dete DELETE RESTRICTED Dete |
| | roditeljStarateljID | Long | Not null and >0 | | | |
| | ime | String | Not null | | | |
| | prezime | String | Not null | | | |
| | adresa | String | Not null | | | |
| | telefon | Long | Not null and >0 | | | |
| | jmbg | Long | Not null and >0 | | | |

Табела 2. Ограниченија за табелу РодитељСтаратељ

| Табела Dete | | Просторно вредносно ограничење | | Сложено вредносно ограничење | | Структурно ограничење |
|-------------|---------------|--------------------------------|-------------------|-------------------------------------|------------------------------------|--|
| Атрибути | Име | Тип атрибута | Вредност атрибута | Међузависност атрибута једне табеле | Међузависност атрибута више табела | INSERT RESTRICTED RoditeljStaratelj UPDATE RESTRICTED RoditeljStaratelj CASCADES Pohadjanje DELETE RESTRICTED Pohadjanje |
| | deteljID | Long | Not null and >0 | | | |
| | ime | String | Not null | | | |
| | prezime | String | Not null | | | |
| | datumRodjenja | Date | Not null | | | |
| | jmbg | Long | Not null and >0 | | | |

Табела 3. Ограниченија за табелу Дете

| Табела IzborniProgram | | Просторно вредносно ограничење | | Сложено вредносно ограничење | | Структурно ограничење |
|-----------------------|------------------|--------------------------------|-------------------|-------------------------------------|------------------------------------|--|
| Атрибути | Име | Тип атрибута | Вредност атрибута | Међузависност атрибута једне табеле | Међузависност атрибута више табела | INSERT / UPDATE CASCADES Pohadjanje, Zaposleni DELETE RESTRICTED Pohadjanje, Zaposleni |
| | izborniProgramID | Long | Not null and >0 | | | |
| | naziv | String | Not null | | | |
| | uzrast | String | Not null | | | |
| | nivoTezine | Integer | Not null and >0 | | | |

Табела 4. Ограниченија за табелу ИзборниПрограм

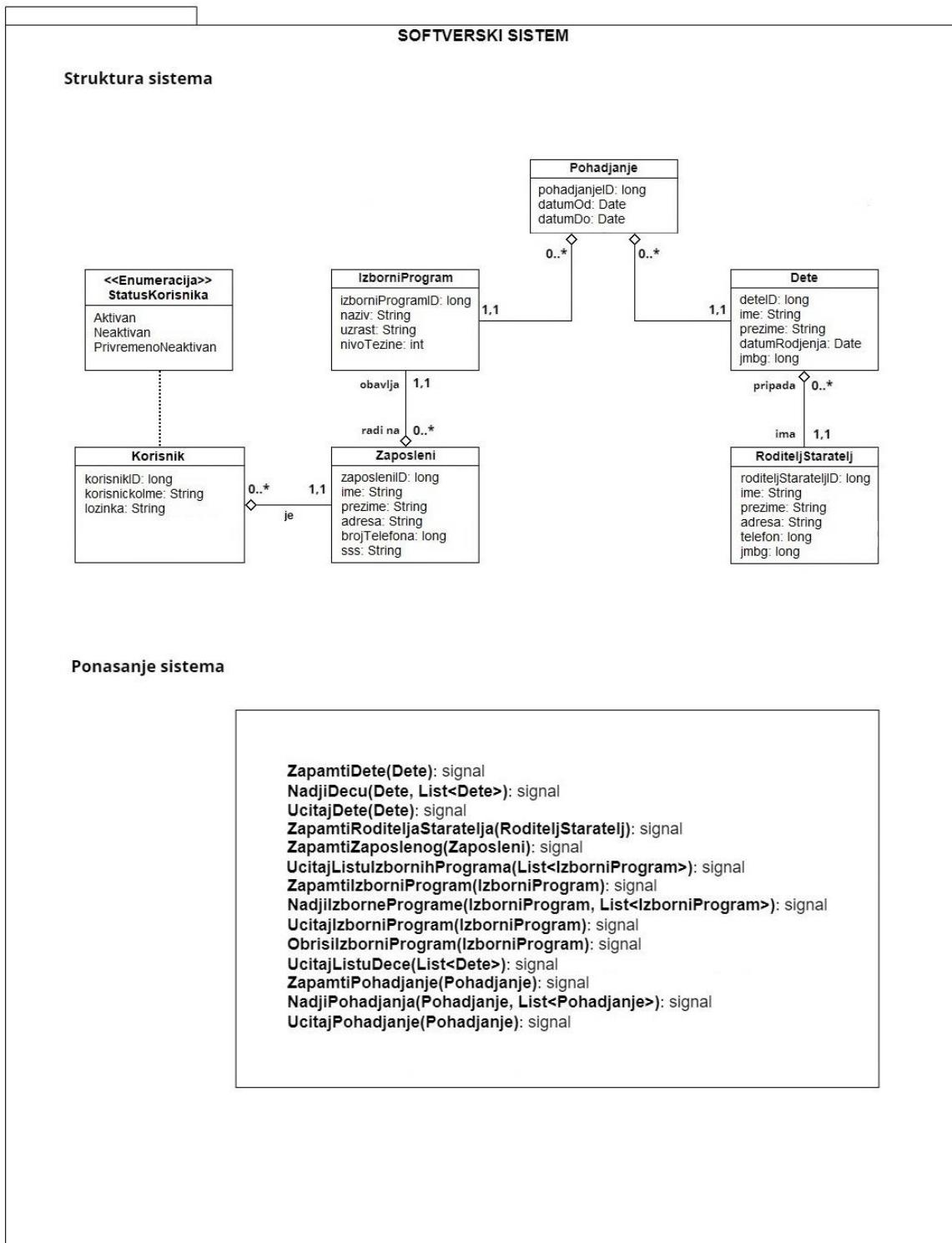
| Табела Pohadjanje | | Просторно вредносно ограничење | | Сложено вредносно ограничење | | Структурно ограничење |
|--------------------------|---------------|--------------------------------|--------------------|-------------------------------------|------------------------------------|--|
| Атрибути | Име | Тип атрибута | Вредност атрибута | Међузависност атрибута једне табеле | Међузависност атрибута више табела | INSERT RESTRICTED Dete, IzborniProgram UPDATE RESTRICTED Dete, IzborniProgram DELETE / |
| | datumOd | Date | Not null | | | |
| | datumDo | Date | Not null | | | |
| | pohadjanjelID | Long | Not null and >0 | | | |

Табела 5. Ограниченија за табелу Пohaђањe

| Табела Korisnik | | Просторно вредносно ограничење | | Сложено вредносно ограничење | | Структурно ограничење |
|------------------------|-----------------|--------------------------------|--------------------|-------------------------------------|------------------------------------|--|
| Атрибути | Име | Тип атрибута | Вредност атрибута | Међузависност атрибута једне табеле | Међузависност атрибута више табела | INSERT RESTRICTED Zaposleni UPDATE RESTRICTED Zaposleni DELETE / |
| | korisnikID | Long | Not null and >0 | | | |
| | korisnickolme | String | Not null | | | |
| | lozinka | String | Not null | | | |
| | statusKorisnika | String | Not null | | | |

Табела 6. Ограниченија за табелу Корисник

Како резултат анализе сценарија СК и израде концептуалног модела, можемо дефинисати логичку структуру и понашање софтверског система:



Слика 71. Понашање и структура система

4.3 Пројектовање

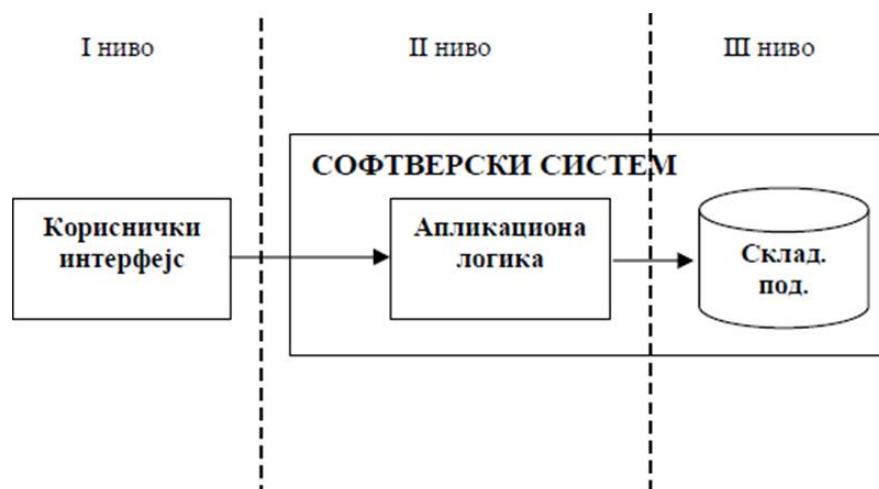
Фаза пројектовања описује физичку структуру и понашање софтверског система, односно, архитектуру софтверског система. Пројектовање архитектуре софтверског система подразумева пројектовање апликационе логике, складишта података и корисничког интерфејса, које обухвата пројектовање екранских форми и контролера корисничког интерфејса. Пројектовање апликационе логике, са друге стране, подразумева пројектовање контролера апликационе логике, брокера базе података и пословне логике, односно, логичке структуре и понашања система.[2]

4.3.1 Архитектура софтверског система

У фази пројектовања софтверског система приликом дефинисања његове архитектуре, најчешће се користи трослојна архитектура. Овај модел обухвата:

1. **кориснички интерфејс**
2. **апликациону логику**
3. **базу података.**[2]

Кориснички интерфејс представља улазно – излазну репрезентацију софтверског система, апликациона логика описује структуру и понашање софтверског система, док складиште података чува стање атрибуата софтверског система.[2]

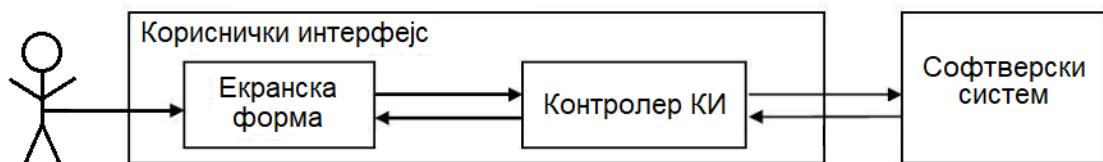


Слика 72. Тронивојска архитектура

4.3.2 Пројектовање корисничког интерфејса

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

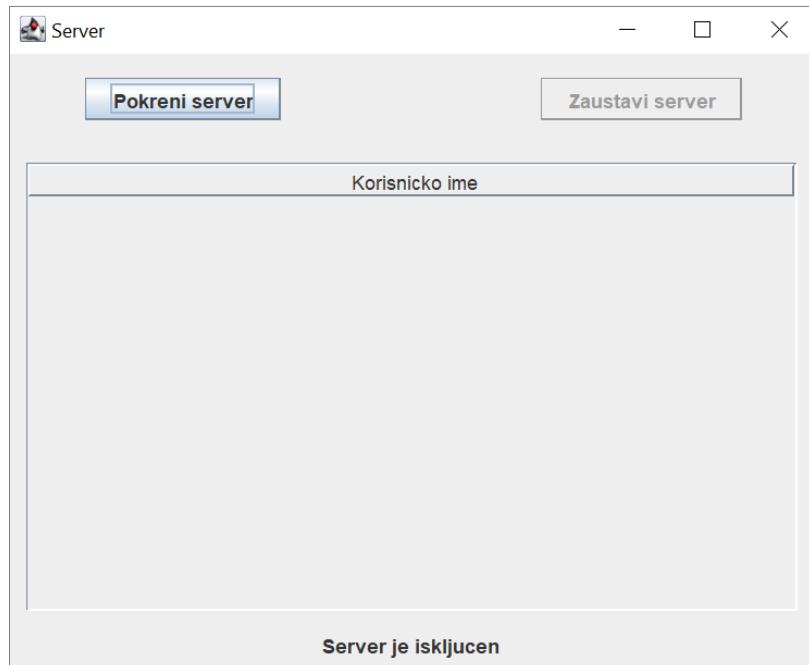
1. Екранске форме
2. Контролера корисничког интерфејса[2]



Слика 73. Структура корисничког интерфејса[2]

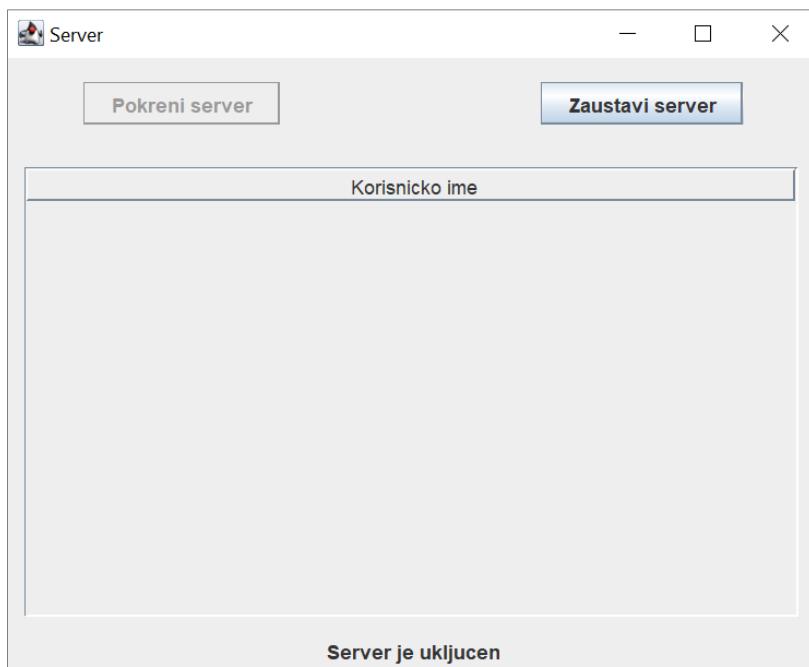
4.3.2.1 Пројектовање екранских форми

Кориснички интерфејс је изграђен кроз низ екранских форми чија су сценарија коришћења директно повезана са сценаријима случајева коришћења. На серверској страни програма пројектована је серверска екранска форма која пре активације изгледа овако:



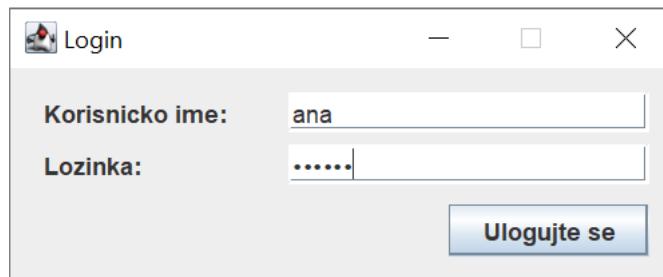
Слика 74. Серверска форма пре покретања

Након активације, серверска форма изгледа овако:



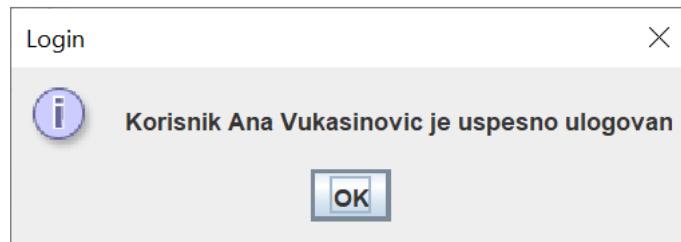
Слика 75. Серверска форма након покретања

Да би се апликација користила, клијент се претходно треба пријавити са својим креденцијалима на клијентској страни. Login форма изгледа овако:

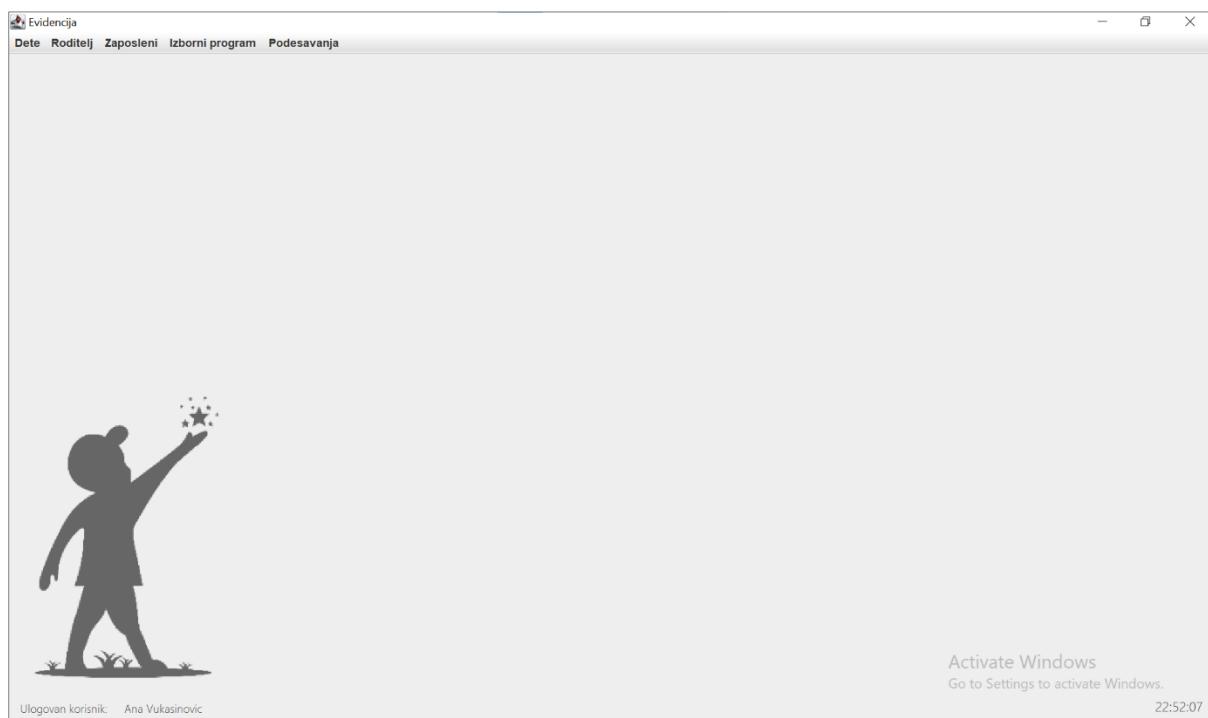


Слика 76. Login форма

У случају успешног пријављивања клијент добија обавештење и приказује му се главна екранска форма са навигационим менијем ка осталим екранским формама.



Слика 77. Login форма-порука о успешном пријављивању



Слика 78. Главна клијентска форма

СК1: Случај коришћења – Креирање детета

Назив СК

Креирање детета

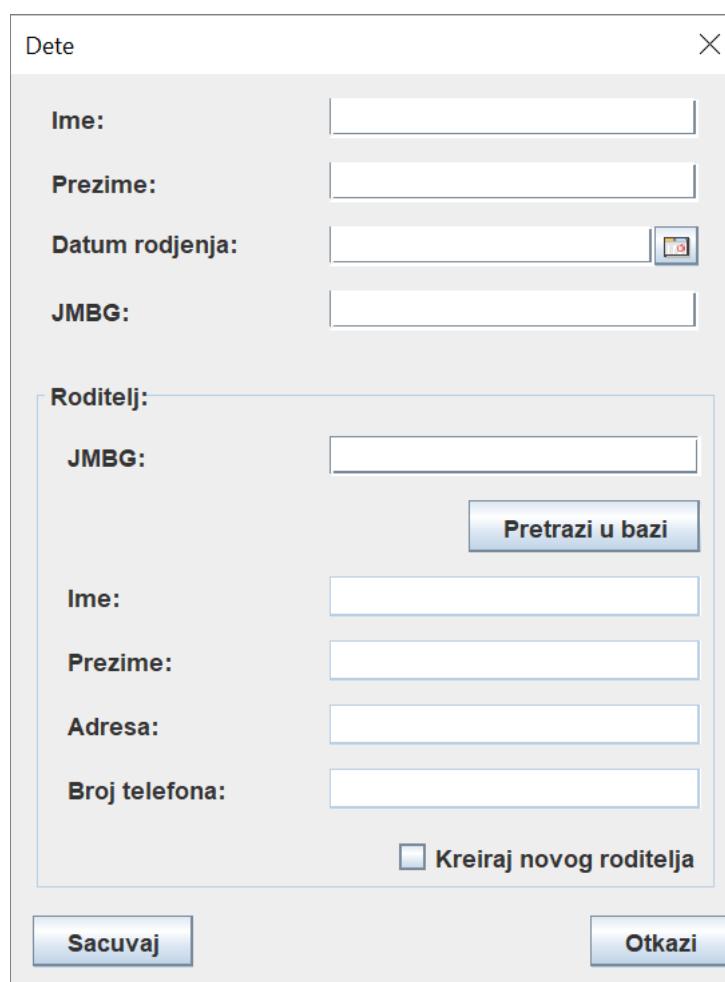
Актори СК

Директор

Учесници СК

Директор и систем (програм)

Предуслов: Систем је укључен и **директор** је пријављен под својом шифром. Систем приказује форму за рад са **дететом**.



Dete

Ime: _____

Prezime: _____

Datum rodjenja: _____

JMBG: _____

Roditelj:

JMBG: _____

Ime: _____

Prezime: _____

Adresa: _____

Broj telefona: _____

Kreiraj novog roditelja

Слика 79. Форма за креирање новог детета

Основни сценарио СК

1. **Директор уноси** податке у **дете**. (АПУСО)

Опис акције: Директор у поља на форми уноси вредности за име, презиме, датум рођења, јмбг детета и јмбг родитеља уколико постоји информација да је родитељ већ сачуван у систему¹. Директор кликом на дугме „Претражи у бази“ позива системску операцију *UcitajRoditelja(Roditelj)*. Систем учитава вредности за родитеља и попуњава преостала поља на форми.

Dete

| | |
|-----------------|--|
| Ime: | Marija |
| Prezime: | Petrasinovic |
| Datum rodjenja: | 24. 8. 2024. <input type="button" value=""/> |
| JMBG: | 2408024715231 |

Roditelj:

| | |
|--|------------------|
| JMBG: | 1111111111111111 |
| <input type="button" value="Pretrazi u bazi"/> | |
| Ime: | Marina |
| Prezime: | Petrasinovic |
| Adresa: | Miliceva 5 |
| Broj telefona: | 381656451770 |

Kreiraj novog roditelja

Слика 80. Унос података за ново дете

¹ Уколико не постоји информација да је родитељ већ сачуван у систему, директор има могућност да креира новог родитеља чекирањем дугмета „Креирај новог родитеља“ што је детаљно описано у наставку у одељку СК3-Креирање родитеља/старатеља

2. **Директор контролише** да ли је коректно унео податке у **дете**. (АНСО)
3. **Директор позива систем** да запамти податке о **детету**. (АПСО)
Опис акције: Директор кликом на дугме „Сачувати“ позива системску операцију **ZapamtiDete(Dete)**
4. **Систем памти** податке о **детету**. (СО)
5. **Систем приказује директору** запамћени **дете** и поруку: “Систем је запамтио **дете**“. (ИА)

Dete

Ime: Marija

Prezime: Petrasinovic

Datum rodjenja: 24. 8. 2024.

JMBG: 2408024715231

Obavestenje

i Sistem je zapamtio dete [28] Marija Petrasinovic, JMBG: 2408024715231

OK

Prezime: Petrasinovic

Adresa: Miliceva 5

Broj telefona: 381656451770

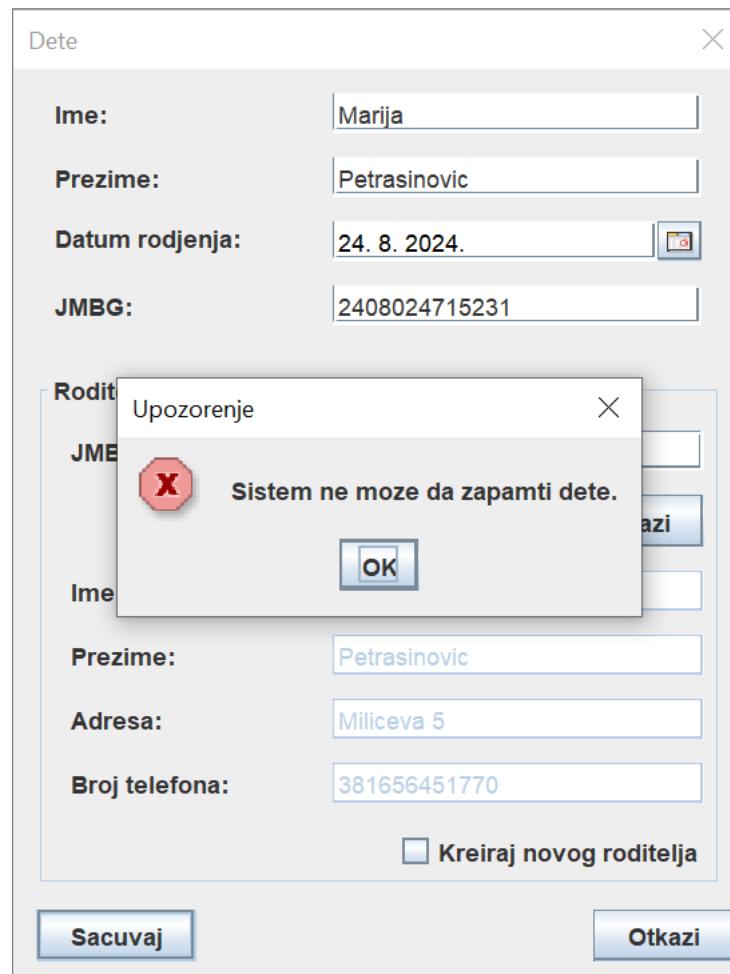
Kreiraj novog roditelja

Sacuvaj Otkazi

Слика 81. Успешно креирано дете

Алтернативна сценарија

5.1 Уколико **систем** не може да запамти податке о **детету** он приказује **директору** поруку “**Систем не може да запамти дете**”. (ИА)



Слика 82. Систем не може да креира дете

СК2: Случај коришћења – Претраживање детета

Назив СК

Претраживање [детета](#)

Актори СК

[Директор](#)

Учесници СК

[Директор](#) и [систем](#) (програм)

Предуслов: Систем је укључен и [директор](#) је пријављен под својом шифром. Систем приказује форму за рад са [дететом](#).

| Ime | Prezime | JMBG | Datum rodjenja | Ime i prezime roditelja |
|---------|--------------|---------------|----------------|-------------------------|
| Marko | Petrasinovic | 6666666666666 | 13-12-2020 | Marina Petrasinovic |
| Elena | Vojvodic | 7777777777777 | 09-07-2018 | Aleksa Vojvodic |
| Lana | Petrovic | 8888888888888 | 22-05-2019 | Marta Petrovic |
| Teodora | Petrovic | 9999999999999 | 16-07-2020 | Marta Petrovic |
| Luka | Vucetic | 1010101010101 | 10-06-2020 | Jovan Vucetic |
| Jovan | Vucetic | 1212121212121 | 09-09-2019 | Jovan Vucetic |
| Nikola | Jovanovic | 1313131313131 | 14-01-2018 | Marija Jovanovic |
| Ana | Anic | 2508024715231 | 25-08-2024 | Marija Anic |
| Ana | Petrasinovic | 1111111111112 | 01-09-2024 | Marina Petrasinovic |
| Marija | Petrasinovic | 2408024715231 | 24-08-2024 | Marina Petrasinovic |

Слика 83. Форма за претраживање деце

Основни сценарио СК

1. **Директор уноси** вредност по којој претражује **децу**. (АПУСО)

Опис акције: Директор уноси вредност за јмбг детета у поље под називом „ЈМБГ”.

| Ime | Prezime | JMBG | Datum rođenja | Ime i prezime roditelja |
|---------|--------------|---------------|---------------|-------------------------|
| Marko | Petrasinovic | 6666666666666 | 13-12-2020 | Marina Petrasinovic |
| Elena | Vojvodic | 7777777777777 | 09-07-2018 | Aleksa Vojvodic |
| Lana | Petrovic | 8888888888888 | 22-05-2019 | Marta Petrovic |
| Teodora | Petrovic | 9999999999999 | 16-07-2020 | Marta Petrovic |
| Luka | Vucetic | 1010101010101 | 10-06-2020 | Jovan Vucetic |
| Jovan | Vucetic | 1212121212121 | 09-09-2019 | Jovan Vucetic |
| Nikola | Jovanovic | 1313131313131 | 14-01-2018 | Marija Jovanovic |
| Ana | Anic | 2508024715231 | 25-08-2024 | Marija Anic |
| Ana | Petrasinovic | 1111111111112 | 01-09-2024 | Marina Petrasinovic |
| Marija | Petrasinovic | 2408024715231 | 24-08-2024 | Marina Petrasinovic |

Слика 84. Унос вредности за претрагу детета

2. **Директор позива** систем да нађе **децу** по задатој вредности. (АПСО)

Опис акције: Директор кликом на дугме „Претражи“ позива системску операцију NadjiDecu(Dete,List<Dete>)

3. **Систем тражи** **децу** по задатој вредности. (СО)
4. **Систем приказује** **директору** податке о **деци** и поруку: “**Систем** је нашао **децу** по задатој вредности”. (ИА)

| Ime | Prezime | JMBG | Datum rođenja | Ime i prezime roditelja |
|--------|--------------|---------------|---------------|-------------------------|
| Marija | Petrasinovic | 2408024715231 | 24-08-2024 | Marina Petrasinovic |

Слика 85. Систем је пронашао дете по задатој вредности

5. **Директор бира дете.** (АПУСО)

Опис акције: Директор бира ред табеле где се налази дете чије податке жели да учита.

The screenshot shows a Windows application window titled 'Evidencija dece'. At the top left is a small icon of a child. Below the title bar is a search bar labeled 'JMBG:' containing '2408024715231'. To the right of the search bar are two buttons: 'Pretrazi' (Search) and 'Prikazi sve' (Show all). The main area contains a table with columns: 'Ime' (Name), 'Prezime' (Last Name), 'JMBG', 'Datum rodjenja' (Date of Birth), and 'Ime i prezime roditelja' (Name of parent/guardian). One row is highlighted in blue, showing 'Marija' in the Name column and 'Petrasinovic' in the Last Name column. At the bottom of the window are three buttons: 'Sacuvaj' (Save), 'Detalji' (Details), 'Izmeni' (Edit), and 'Obrisi' (Delete).

Слика 86. Одабир детета

6. **Директор позива систем да учита дете.** (АПСО)

Опис акције: Директор кликом на дугме „Детаљи“ позива системску операцију *UcitajDete(Dete)*

7. **Систем учитава дете.** (СО)

8. **Систем приказује директору податке о детету и поруку: "Систем је учитао дете."** (ИА)

The screenshot shows the same application window as before. In the search bar, '2508024715231' is entered. A modal dialog box titled 'Obavestenje' (Information) is displayed in the center. It contains an information icon and the text 'Sistem je ucitao dete' (System has read the child). At the bottom of the dialog is an 'OK' button. The bottom of the window features the standard buttons: 'Sacuvaj', 'Detalji', 'Izmeni', and 'Obrisi'.

Слика 87. Систем је учитао одабрано дете

The screenshot shows a software interface for managing child records. A specific child record is selected, and a modal window displays detailed information about the child and their parent.

| Dete | | Roditelj | |
|----------------|---------------|-----------------|----------------------|
| Ime i prezime | Ana Anic | Ime i prezime | Marija Anic |
| JMBG | 2508024715231 | JMBG | 2403985715432 |
| Datum rodjenja | 2024-08-25 | Kontakt telefon | 381642308754 |
| | | Adresa | Jovanke Radakovic 5a |

Buttons at the bottom: Sacuvaj (Save), Detalji (Details), Izmeni (Edit), Obrisi (Delete), and Otkazi (Cancel).

Слика 88. Систем показује податке о одабраном детету

Алтернативна сценарија

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

The screenshot shows a search results table for children. A search query was entered, but no results were found, triggering an error message.

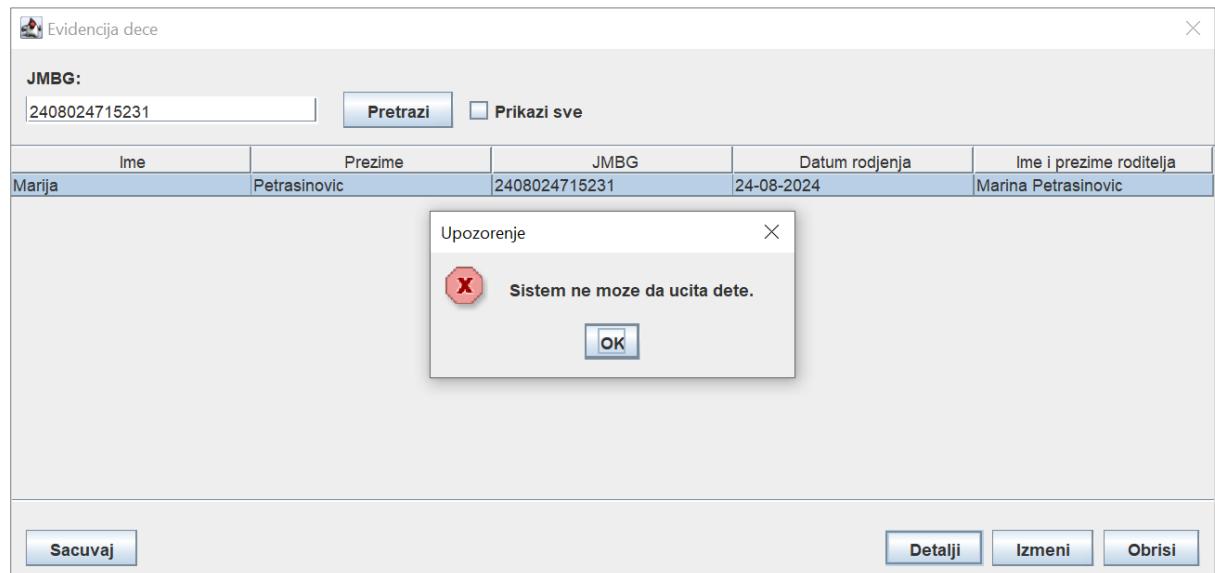
| Ime | Prezime | JMBG | Datum rodjenja | Ime i prezime roditelja |
|---------|--------------|---------------|----------------|-------------------------|
| Marko | Petrasinovic | 6666666666666 | 13-12-2020 | Marina Petrasinovic |
| Elena | Vojvodic | 7777777777777 | 09-07-2018 | Aleksa Vojvodic |
| Lana | Petrovic | | | Marta Petrovic |
| Teodora | Petrovic | | | Marta Petrovic |
| Luka | Vucetic | | | Jovan Vucetic |
| Jovan | Vucetic | | | Jovan Vucetic |
| Nikola | Jovanovic | | | Marija Jovanovic |
| Ana | Anic | | | Marija Anic |
| Ana | Petrasinovic | | | Marina Petrasinovic |
| Marija | Petrasinovic | 2408024715231 | 24-08-2024 | Marina Petrasinovic |

Search fields: JMBG (2345674328765), Pretrazi (Search), Prikazi sve (Show all). Error message: Upozorenje (Warning): Sistem ne moze da nadje decu po zadatoj vrednosti. (System cannot find the child by the specified value.)

Buttons at the bottom: Sacuvaj (Save), Detalji (Details), Izmeni (Edit), Obrisi (Delete), and OK.

Слика 89. Систем не може да пронађе дете по задатој вредности

8.1 Уколико **систем** не може да учита **дете** он приказује **директору** поруку: “**Систем не може да учита дете.** (ИА)



Слика 90. Систем не може да учита одабрано дете

СК3: Случај коришћења – Креирање родитеља/старатеља

Назив СК

Креирање родитеља/старатеља

Актори СК

Директор

Учесници СК

Директор и систем (програм)

Предуслов: Систем је укључен и **директор** је пријављен под својом шифром. Систем приказује форму за рад са **родитељем/старатељем**.

Dete

Ime: _____

Prezime: _____

Datum rodjenja: _____

JMBG: _____

Roditelj:

JMBG: _____

Ime: _____

Prezime: _____

Adresa: _____

Broj telefona: _____

Kreiraj novog roditelja

Sacuvaj Otkazi

Слика 91. Форма за креирање новог детета

Основни сценарио СК

1. **Директор уноси** податке у **родитеља/старатеља**. (АПУСО)

Опис акције: Директор у поља на форми уноси вредности име, презиме, датум рођења и јмбг за дете као и вредности јмбг, име презиме, адреса и број телефона за родитеља након што је чекирао дугме „Креирај новог родитеља“.

| | |
|---|--|
| Ime: | Ana |
| Prezime: | Anic |
| Datum rodjenja: | 1. 9. 2024. <input type="button" value="X"/> |
| JMBG: | 0109024715231 |
| Roditelj: | |
| JMBG: | 0405085715231 |
| Ime: | Marija |
| Prezime: | Anic |
| Adresa: | Narodnih heroja 31 |
| Broj telefona: | 381656451770 |
| <input checked="" type="checkbox"/> Kreiraj novog roditelja | |
| <input type="button" value="Sacuvaj"/> | <input type="button" value="Otkazi"/> |

Слика 92. Унос података у новог родитеља/старатеља

2. **Директор контролише** да ли је коректно унео податке у **родитеља/старатеља**. (АНСО)
3. **Директор позива систем** да запамти податке о **родитељу/старатељу**. (АПСО)
Опис акције: Директор кликом на дугме „Сачувати“ позива системску операцију *ZapamtiRoditeljaStaratelja(RoditeljStaratelj)*.
4. **Систем памти** податке о **родитељу/старатељу**. (СО)
5. **Систем приказује директору** запамћеног родитеља/старатеља и поруку: “**Систем је запамтио родитеља/старатеља**”. (ИА)

Dete

Ime: Ana

Prezime: Anic

Datum rodjenja: 1. 9. 2024.

JMBG: 0109024715231

Obavestenje

Sistem je zapamtio roditelja/staratelja [8] Marija Anic, JMBG: 405085715231

OK

Prezime: Anic

Adresa: Narodnih heroja 31

Broj telefona: 381656451770

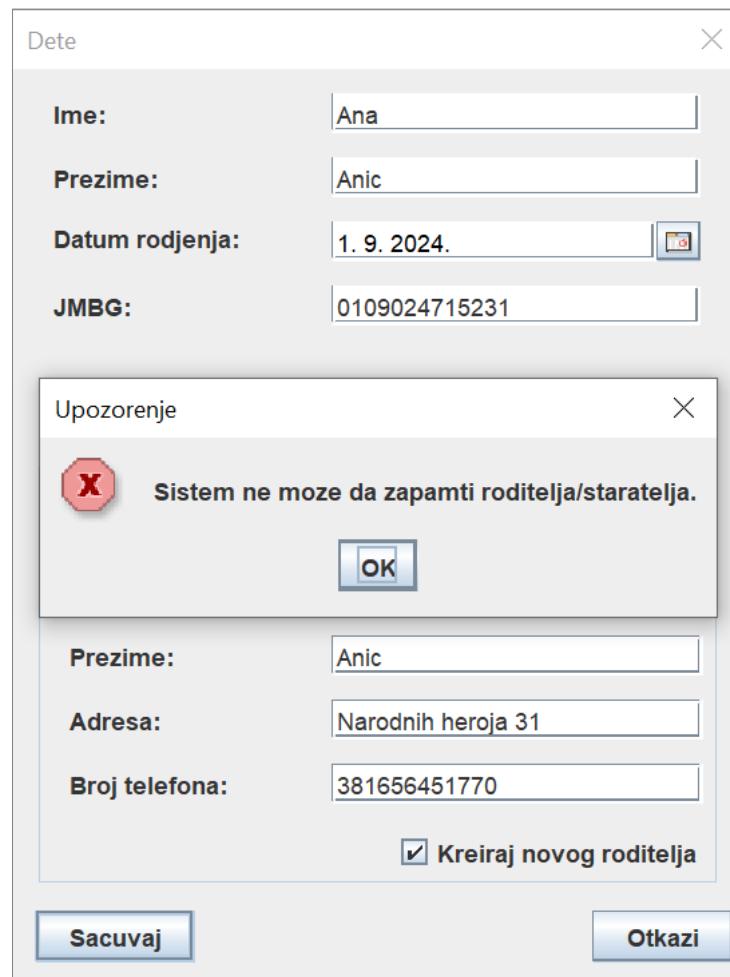
Kreiraj novog roditelja

Sacuvaj Otkazi

Слика 93. Успешно креиран родитељ/старатељ

Алтернативна сценарија

5.1 Уколико систем не може да запамти податке о родитељу/старатељу он приказује директоруј поруку “Систем не може да запамти родитеља/старатеља”. (ИА)



Слика 94. Систем не може да креира родитеља/старатеља

СК4: Случај коришћења – Креирање запосленог

Назив СК

Креирање запосленог

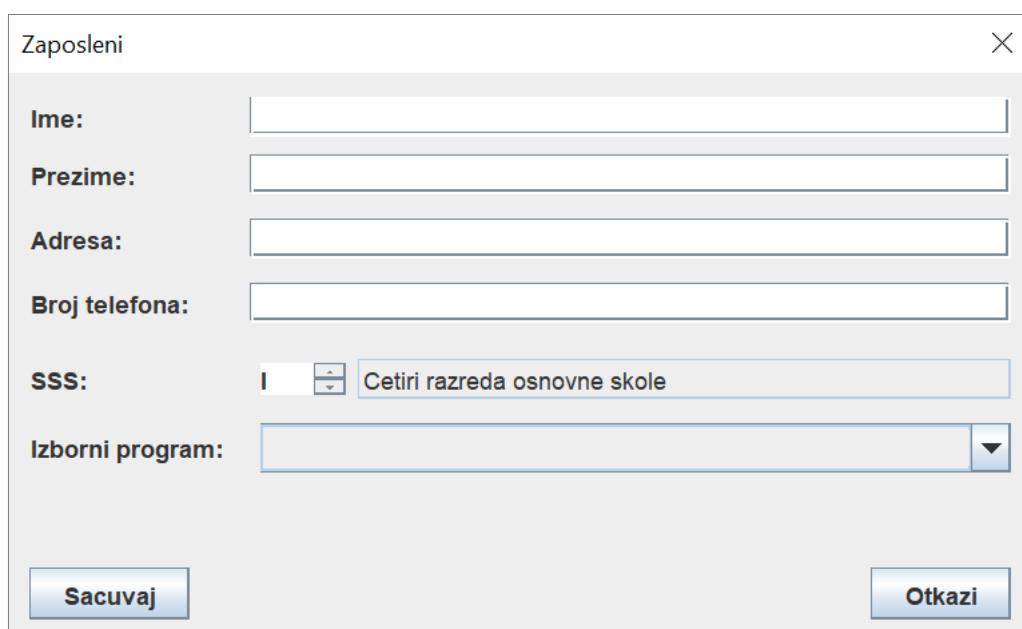
Актори СК

Директор

Учесници СК

Директор и систем (програм)

Предуслов: Систем је укључен и **директор** је пријављен под својом шифром. Систем приказује форму за рад са **запосленим**. Учитана је листа изборних програма.



The form is titled 'Zaposleni'. It contains the following fields:

- Ime: [Text input field]
- Prezime: [Text input field]
- Adresa: [Text input field]
- Broj telefona: [Text input field]
- SSS: [Text input field] with a dropdown menu showing 'Cetiri razreda osnovne skole'
- Izborni program: [Dropdown menu]

At the bottom are two buttons: 'Sacuvaj' (Save) and 'Otkazi' (Cancel).

Слика 95. Форма за креирање новог запосленог

Основни сценарио СК

1. **Директор уноси податке у запосленог.** (АПУСО)

Опис акције: Директор у поља на форми уноси вредности име, презиме, адреса, број телефона, степен стручне спреме и бира одговарајући изборни програм запосленог.

| | |
|------------------|------------------|
| Ime: | Jovana |
| Prezime: | Ivanovic |
| Adresa: | Juzni bulevar 56 |
| Broj telefona: | 381642357643 |
| SSS: | VII-1 |
| Izborni program: | gluma |

Слика 96. Унос података за новог запосленог

2. **Директор контролише да ли је коректно унео податке у запосленог.** (АНСО)
3. **Директор позива систем да запамти податке о запосленом.** (АПСО)
Опис акције: Директор кликом на дугме „Сачувај“ позива системску операцију `ZapamtiZaposlenog(Zaposleni)`
4. **Систем памти податке о запосленом.** (СО)
5. **Систем приказује директору запамћеног запосленог и поруку: "Систем је запамтио запосленог".** (ИА)

Obavestenje

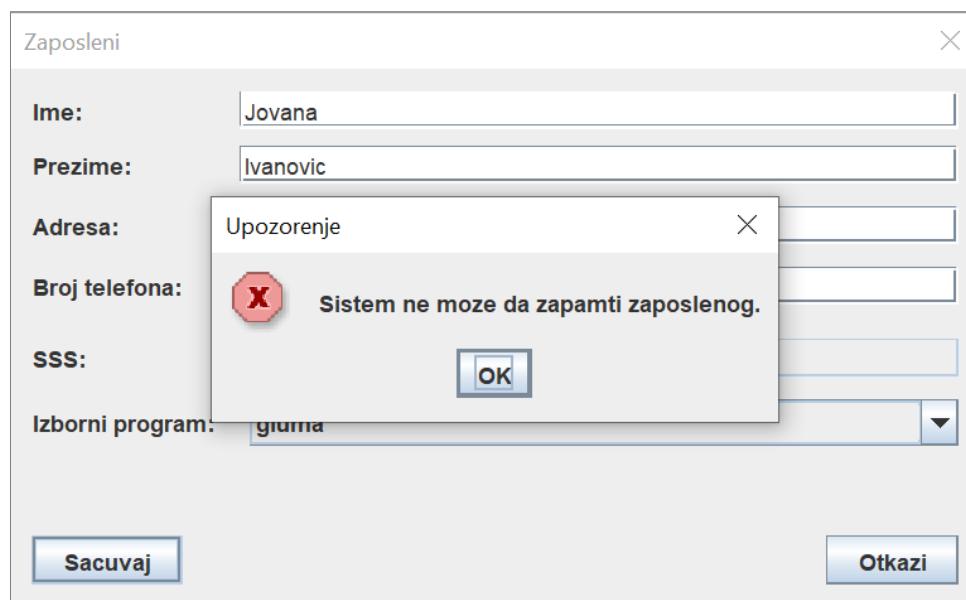
Sistem je zapamtio zaposlenog [14] Jovana Ivanovic

OK

Слика 97. Успешно креиран запослени

Алтернативна сценарија

5.1 Уколико **систем** не може да запамти податке о **запосленом** он приказује **директору** поруку “**Систем** не може да запамти **запосленог**”. (ИА)



Слика 98. Систем не може да креира запосленог

СК5: Случај коришћења – Креирање изборног програма

Назив СК

Креирање изборног програма

Актори СК

Директор

Учесници СК

Директор и систем (програм)

Предуслов: Систем је укључен и **директор** је пријављен под својом шифром. Систем приказује форму за рад са **изборним програмом**.

Izborni program

Naziv:

Predvidjeni uzrast:

Nivo tezine:

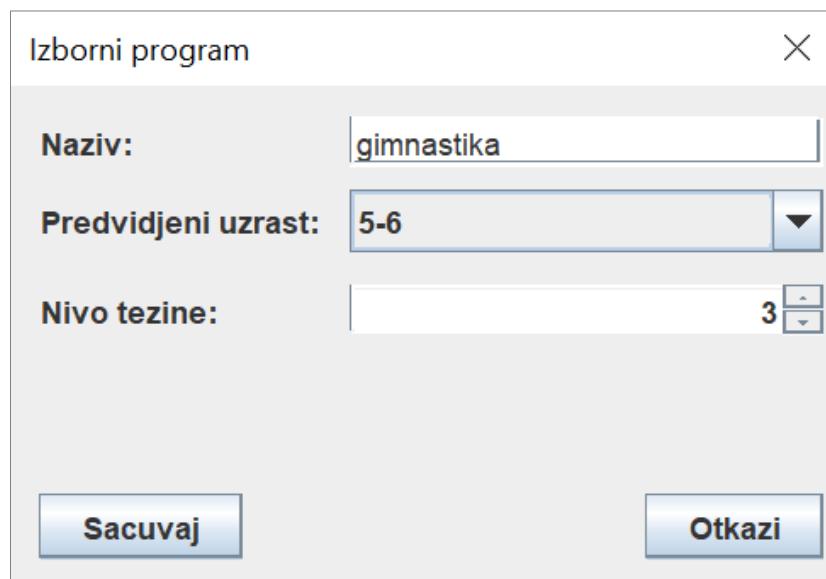
Sacuvaj **Otkazi**

Слика 99. Форма за креирање изборног програма

Основни сценарио СК

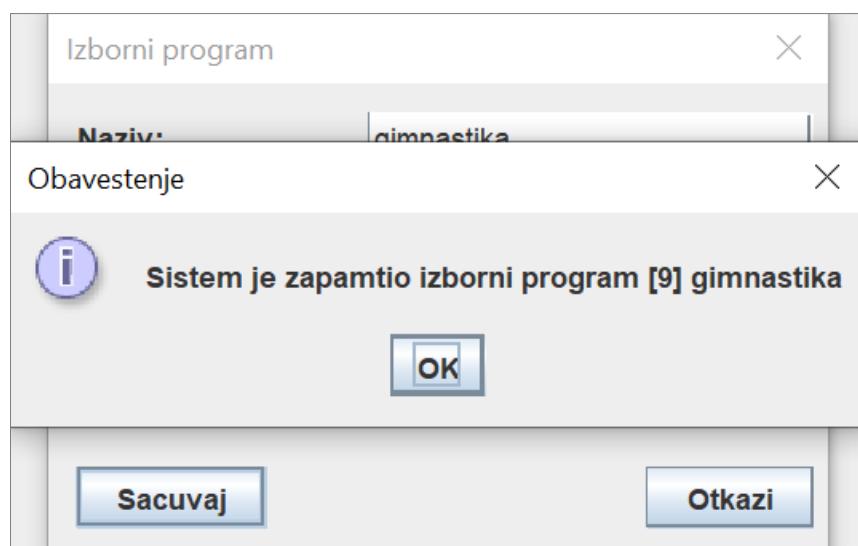
1. **Директор уноси** податке у **изборни програм**. (АПУСО)

Опис акције: Директор у поља на форми уноси вредност назив и бира одговарајући узраст и тежину изборног програма.



Слика 100. Унос података за нови изборни програм

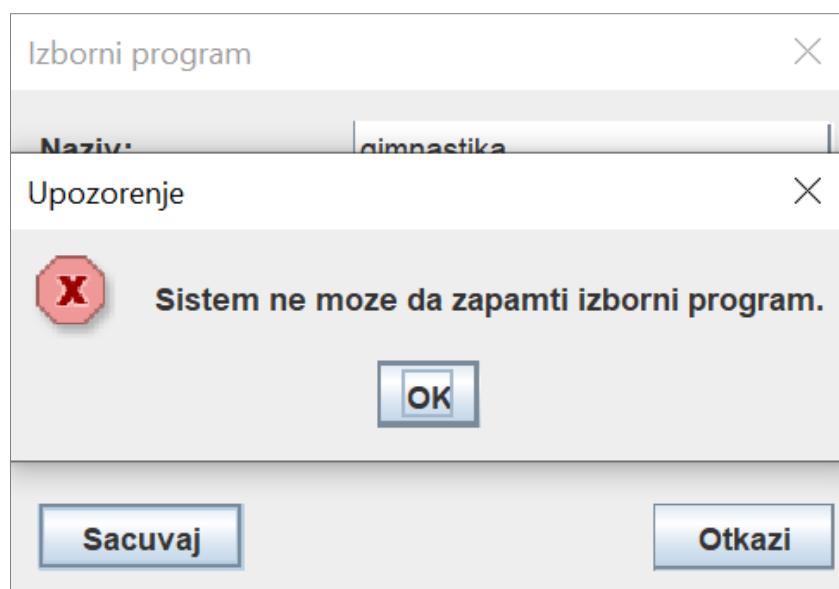
2. **Директор контролише** да ли је коректно унео податке у **изборни програм**. (АНСО)
3. **Директор позива** систем да запамти податке о **изборном програму**. (АПСО)
Опис акције: Директор кликом на дугме „Сачувај“ позива системску операцију `ZapamtilzborniProgram(IzborniProgram)`
4. Систем **памти** податке о **изборном програму**. (СО)
5. Систем **приказује** директору запамћени изборни програм и поруку: “Систем је запамтио изборни програм“. (ИА)



Слика 101. Успешно креиран изборни програм

Алтернативна сценарија

5.1 Уколико **систем** не може да запамти податке о **изборном програму** он приказује **директору** поруку “**Систем** не може да запамти изборни програм”. (ИА)



Слика 102. Систем не може да креира изборни програм

СК6: Случај коришћења –Измена изборног програма

Назив СК

Промена изборног програма

Актори СК

Директор

Учесници СК

Директор и систем (програм)

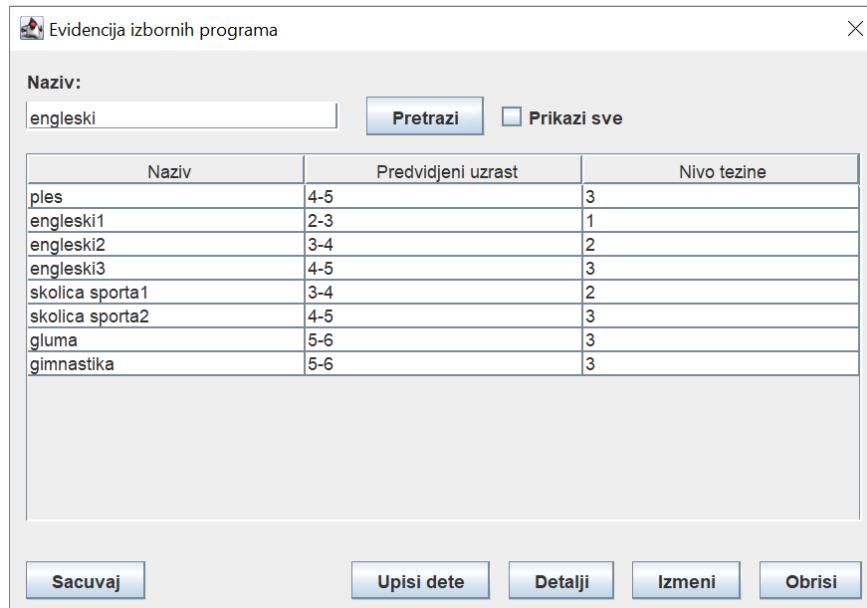
Предуслов: Систем је укључен и директор је пријављен под својом шифром. Систем приказује форму за рад са изборним програмом.

| Naziv | Predviđeni uzrast | Nivo tezine |
|-----------------|-------------------|-------------|
| ples | 4-5 | 3 |
| engleski1 | 2-3 | 1 |
| engleski2 | 3-4 | 2 |
| engleski3 | 4-5 | 3 |
| skolica sporta1 | 3-4 | 2 |
| skolica sporta2 | 4-5 | 3 |
| gluma | 5-6 | 3 |
| gimnastika | 5-6 | 3 |

Слика 103. Форма за претраживање изборних програма

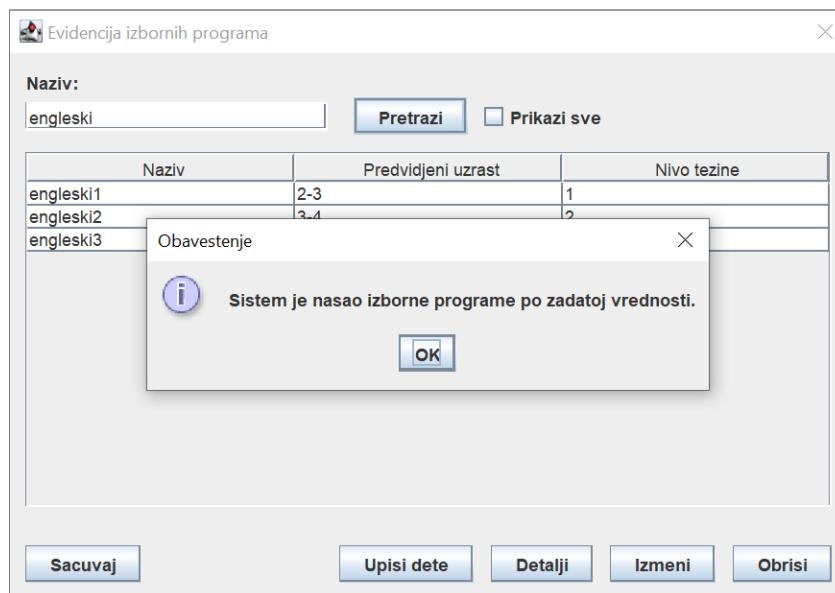
Основни сценарио СК

1. **Директор уноси** вредност по којој претражује изборне програме. (АПУСО)
Опис акције: Директор уноси вредност за назив изборног програма у поље под називом „Назив”.



Слика 104. Унос вредности за претрагу изборних програма

2. **Директор позива** систем да нађе изборне програме по задатој вредности. (АПСО)
Опис акције: Директор кликом на дугме „Претражи“ позива системску операцију `NadjIzbornePrograme(IzborniProgram,List<IzborniProgram>)`
3. Систем тражи изборне програме по задатој вредности. (СО)
4. Систем приказује директору изборне прораме и поруку: “Систем је нашао изборне програме по задатој вредности”. (ИА)



Слика 105. Систем је пронашао изборне програме по задатој вредности

5. **Директор бира изборни програм.** (АПУСО)

Опис акције: Директор бира ред табеле где се налази изборни програм чије податке жели да учита и измени.

| Naziv | Predviđeni uzrast | Nivo tezine |
|-----------|-------------------|-------------|
| engleski1 | 2-3 | 1 |
| engleski2 | 3-4 | 2 |
| engleski3 | 4-5 | 3 |

Слика 106. Одабир изборног програма

6. **Директор позива систем да учита изборни програм.** (АПСО)

Опис акције: Директор кликом на дугме „Измени“ позива системску операцију *UcitajIzborniProgram(IzborniProgram)*

7. **Систем учитава изборни програм.** (СО)

8. **Систем приказује директору податке о изборном програму и поруку: “Систем је учитао изборни програм.”** (ИА)

Слика 107. Систем је учитао одабрани изборни програм

Evidencija izbornih programa

Naziv: engleski

Izborni program

Naziv: engleski3

Predviđeni uzrast: 4-5

Nivo tezine: 3

Sacuvaj **Otkazi**

Sacuvaj **Upisi dete** **Detalji** **Izmeni** **Obrisi**

Слика 108. Систем показује податке о одабраном изборном програму

9. **Директор уноси (мења) податке о изборном програму.** (АПУСО)

Опис акције: Директор у поља на форми уноси нову вредност назив или мења одабир одговарајућег узраста и тежине изборног програма.

Evidencija izbornih programa

Naziv: engleski

Izborni program

Naziv: engleski C

Predviđeni uzrast: 4-5

Nivo tezine: 3

Sacuvaj **Otkazi**

Sacuvaj **Upisi dete** **Detalji** **Izmeni** **Obrisi**

Слика 109. Измена података одабраног изборног програма

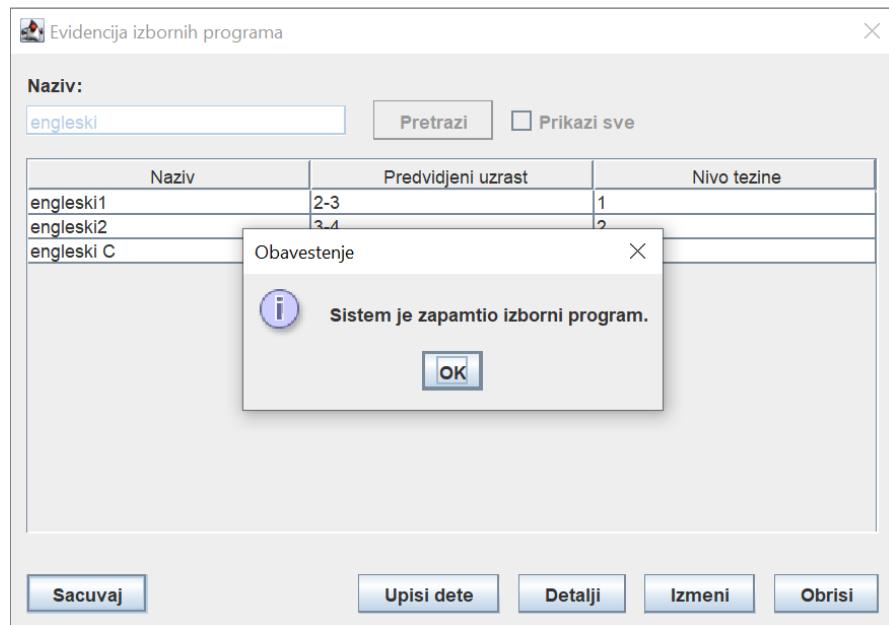
10. **Директор контролише** да ли је коректно унео податке о изборном програму. (АНСО)

11. **Директор позива систем** да запамти податке о изборном програму. (АПСО)

Опис акције: Директор кликом на дугме „Сачувати“ позива системску операцију `ZapamtilzborniProgram(IzborniProgram)`.

12. **Систем памти** податке о изборном програму. (СО)

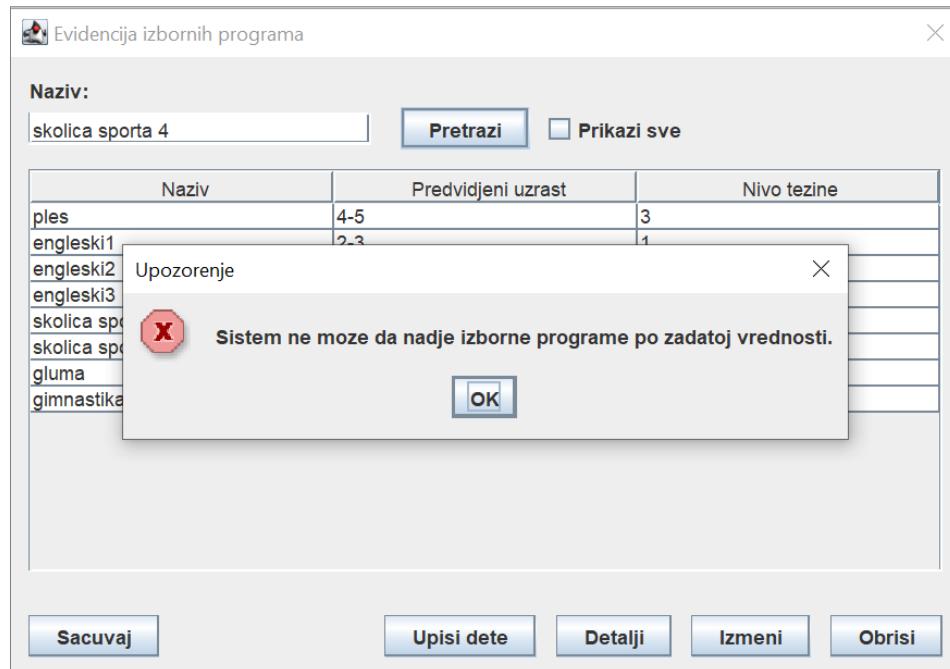
13. Систем приказује директорију запамћени изборни програм и поруку: "Систем је запамтио изборни програм." (ИА)



Слика 110. Систем је замаптио податке о одабраном изборном програму

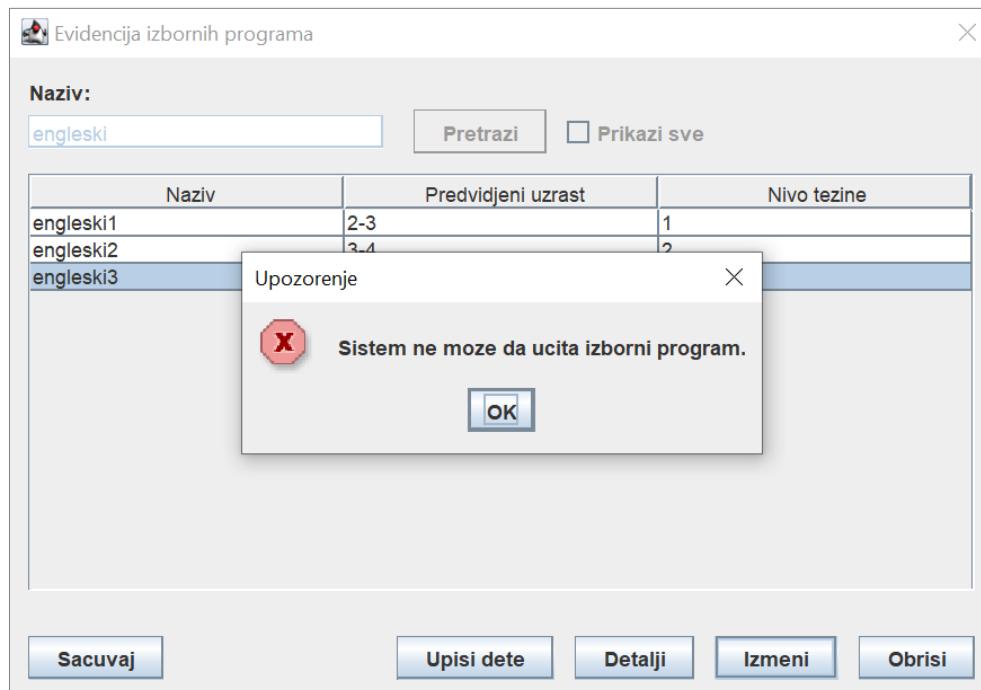
Алтернативна сценарија

4.1 Уколико систем не може да нађе изборне програме он приказује директорију поруку: "Систем не може да нађе изборне програме по задатој вредности". Прекида се извршење сценарија. (ИА)



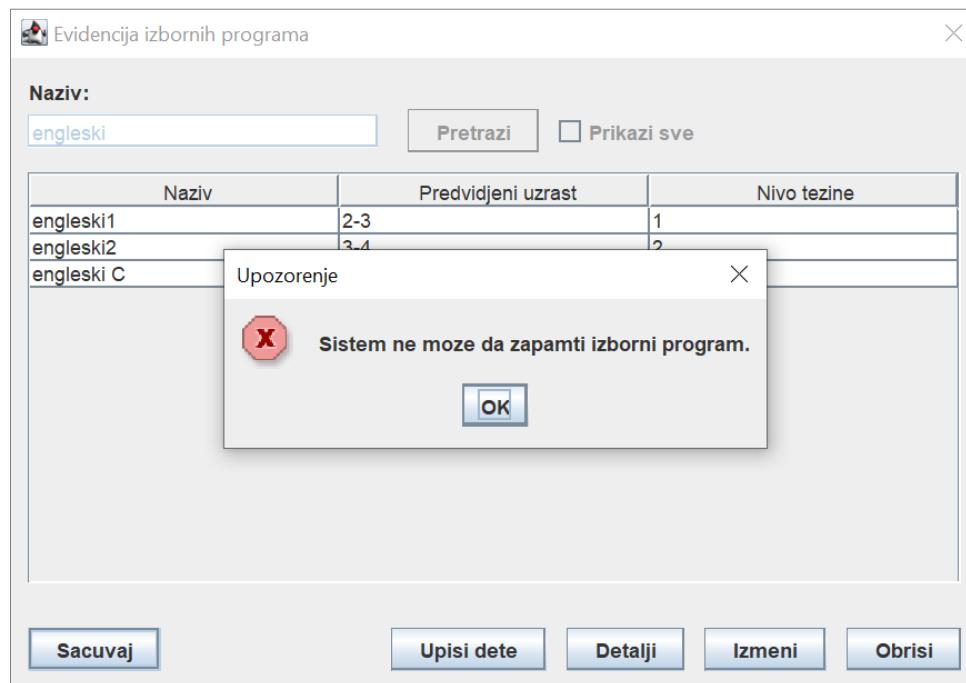
Слика 111. Систем не може да пронађе изборне програме по задатој вредности

8.1 Уколико **систем** не може да учита изборни програм он приказује директору поруку: “**Систем** не може да учита изборни програм. Прекида се извршење сценарија. (ИА)



Слика 112. Систем не може да учита одабрани изборни програм

13.1 Уколико **систем** не може да запамти податке о изборном програму он приказује директору “**Систем** не може да запамти изборни програм”. (ИА)



Слика 113. Систем не може да запамти одабрани изборни програм

СК7: Случај коришћења –Брисање изборног програма

Назив СК

Брисање изборног програма

Актори СК

Директор

Учесници СК

Директор и систем (програм)

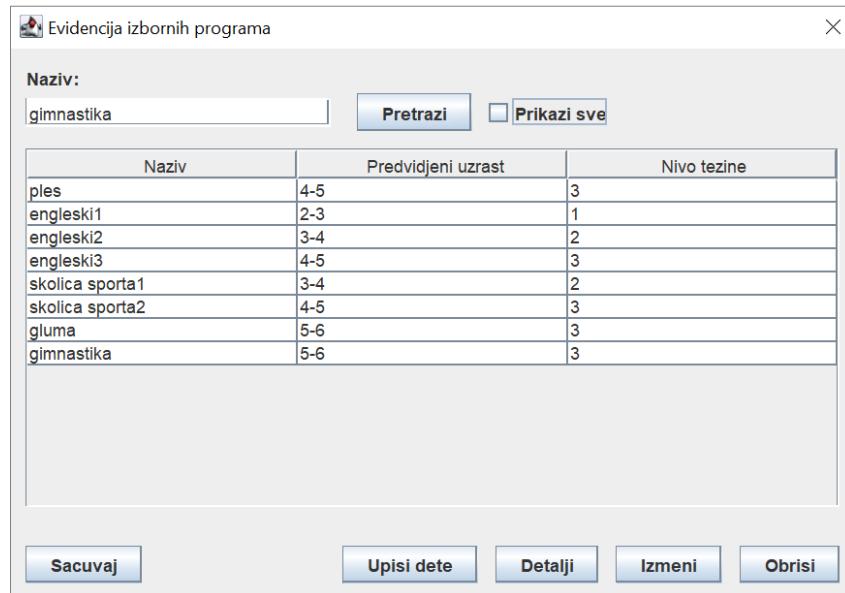
Предуслов: Систем је укључен и директор је пријављен под својом шифром. Систем приказује форму за рад са изборним програмом.

| Naziv | Predviđeni uzrast | Nivo tezine |
|-----------------|-------------------|-------------|
| ples | 4-5 | 3 |
| engleski1 | 2-3 | 1 |
| engleski2 | 3-4 | 2 |
| engleski3 | 4-5 | 3 |
| skolica sporta1 | 3-4 | 2 |
| skolica sporta2 | 4-5 | 3 |
| gluma | 5-6 | 3 |
| gimnastika | 5-6 | 3 |

Слика 114. Форма за претраживање изборних програма

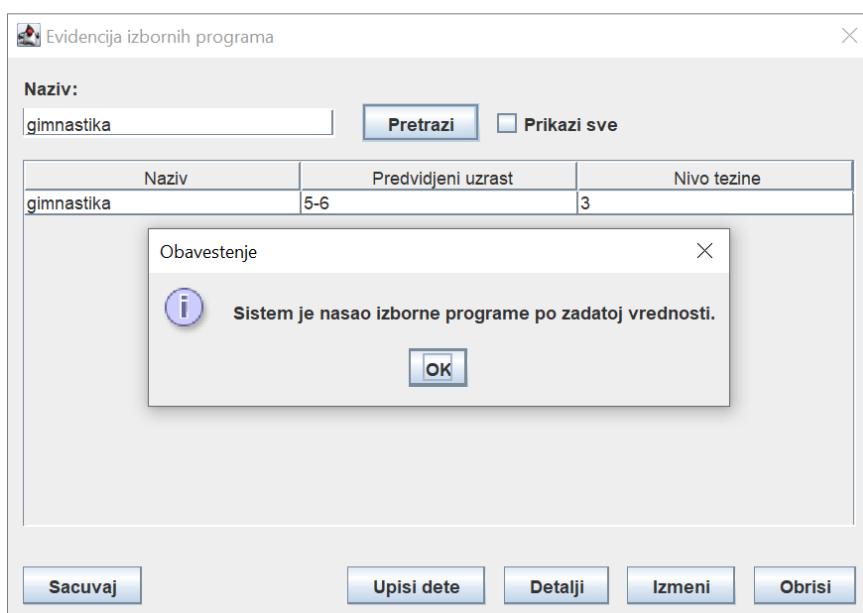
Основни сценарио СК

1. **Директор уноси** вредност по којој претражује изборне програме. (АПУСО)
Опис акције: Директор уноси вредност за назив изборног програма у поље под називом „Назив”.



Слика 115. Унос вредности за претрагу изборних програма

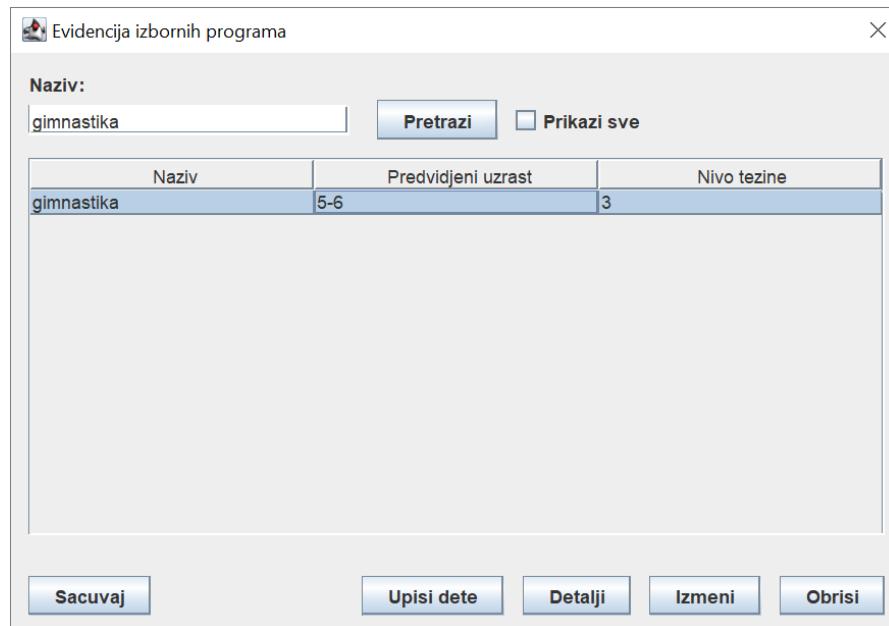
2. **Директор позива** систем да нађе изборне програме по задатој вредности. (АПСО)
Опис акције: Директор кликом на дугме „Претражи“ позива системску операцију `NadjIzbornePrograme(IzborniProgram,List<IzborniProgram>)`
3. Систем тражи изборне програме по задатој вредности. (СО)
4. Систем приказује директору изборне програме и поруку: “Систем је нашао изборне програме по задатој вредности”. (ИА)



Слика 116. Систем је пронашао изборне програме по задатој вредности

5. **Директор бира изборни програм.** (АПУСО)

Опис акције: Директор бира ред табеле где се налази изборни програм чије податке жели да учита и обрише.



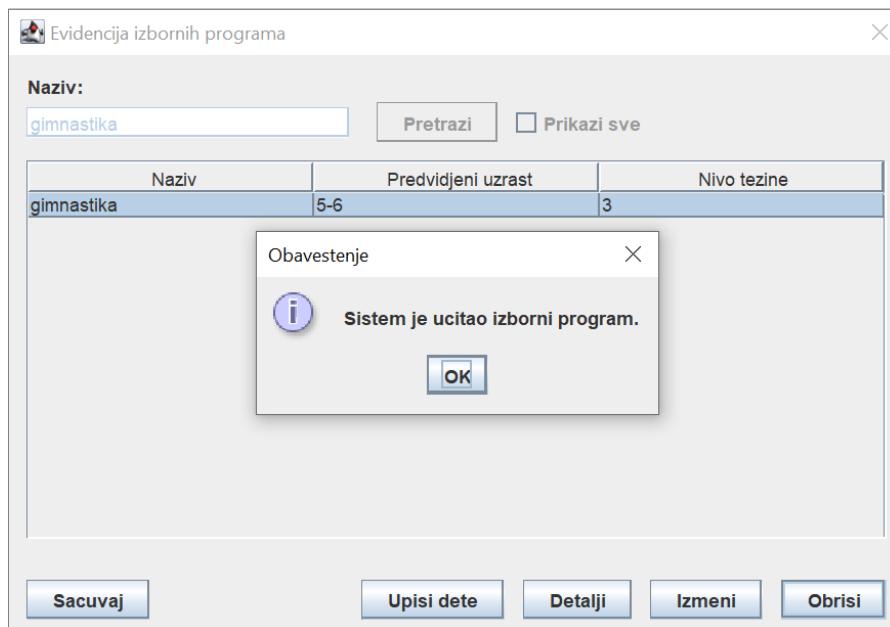
Слика 117. Одабир изборног програма

6. **Директор позива систем да учита изборни програм.** (АПСО)

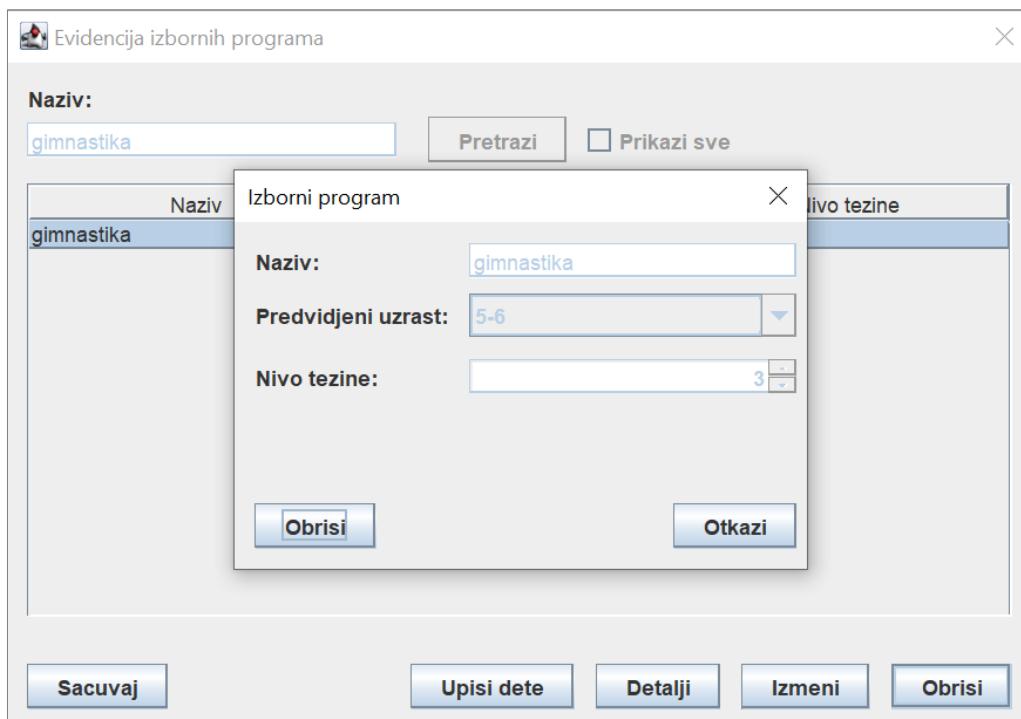
Опис акције: Директор кликом на дугме „Обриши“ позива системску операцију `UcitajIzborniProgram(IzborniProgram)`

7. **Систем учитава изборни програм.** (СО)

8. **Систем приказује директору податке о изборном програму и поруку: "Систем је учитао изборни програм."** (ИА)

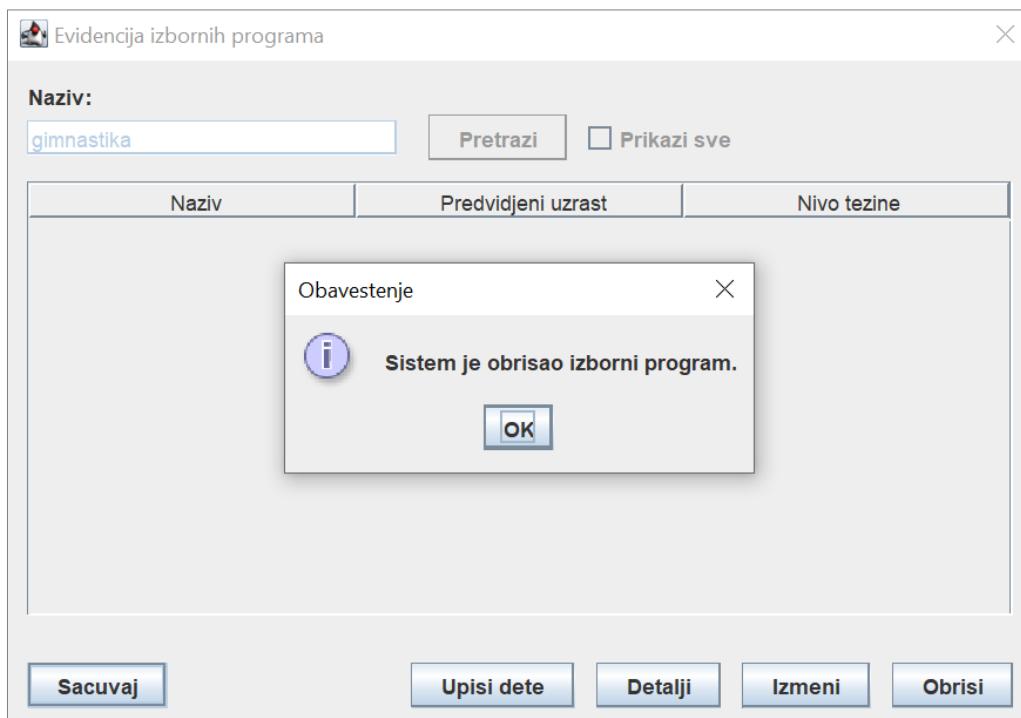


Слика 118. Систем је учитао одабрани изборни програм



Слика 119. Систем показује податке о одабраном изборном програму

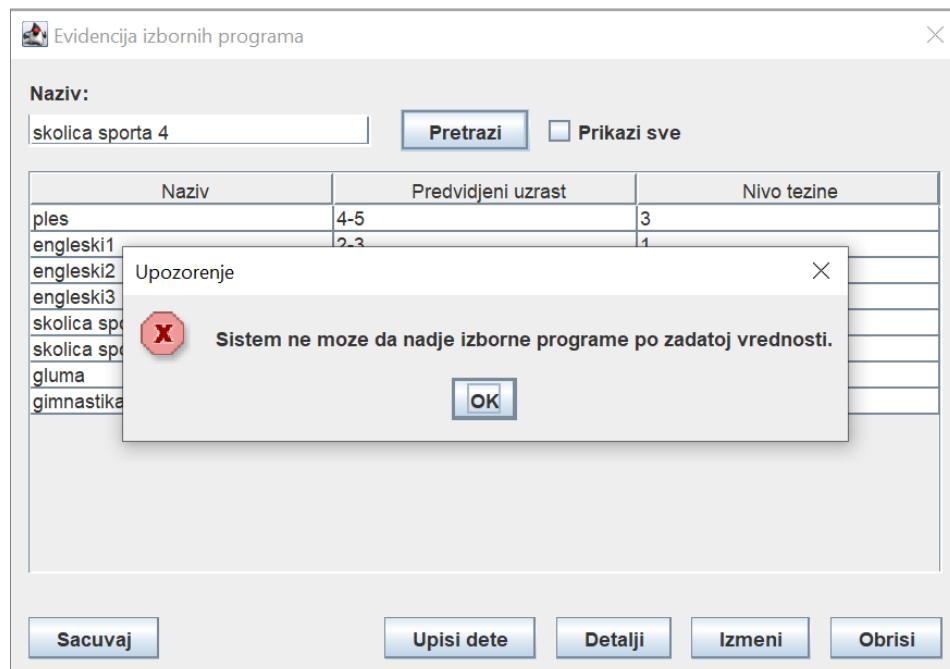
9. Директор позива систем да обрише изборни програм. (АПСО)
Опис акције: Директор кликом на дугме „Обриши“ позива системску операцију `ObrišiIzborniProgram(IzborniProgram)`.
10. Систем брише изборни програм. (СО)
11. Систем приказује директору поруку: “Систем је обрисао изборни програм.” (ИА)



Слика 120. Систем је обрисао одабрани изборни програм

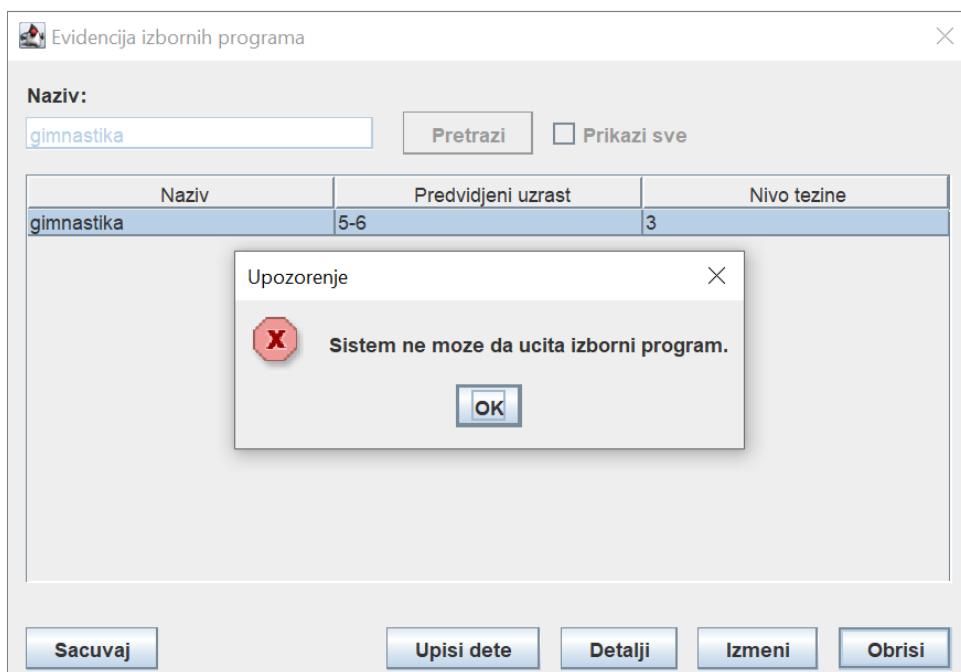
Алтернативна сценарија

4.1 Уколико систем не може да нађе изборне програме он приказује директору поруку: "Систем не може да нађе изборне програме по задатој вредности". Прекида се извршење сценарија. (ИА)



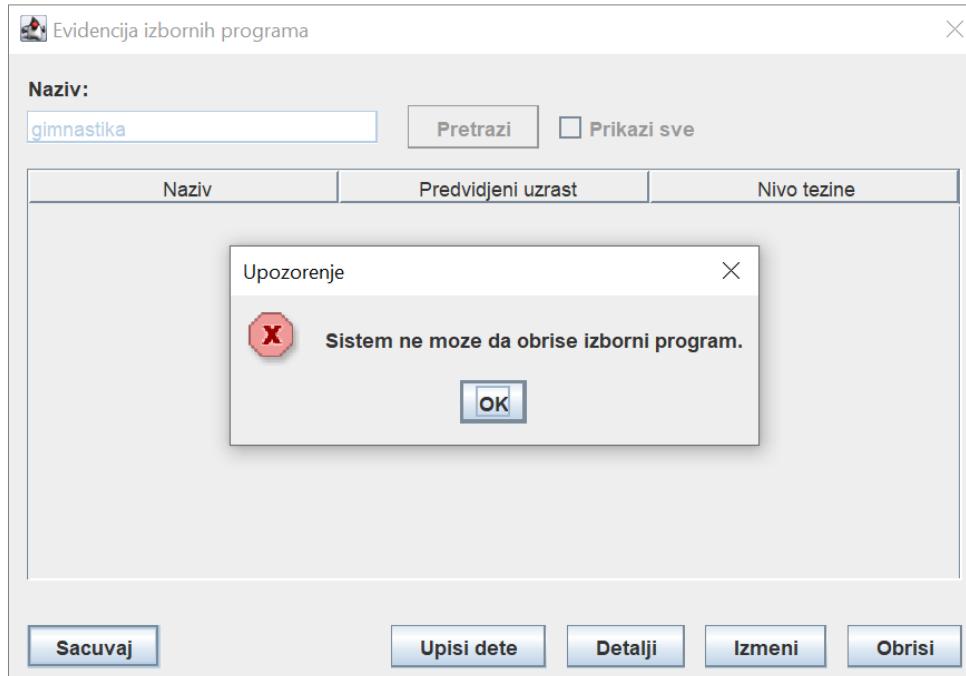
Слика 121. Систем не може да пронађе изборне програме по задатој вредности

8.1 Уколико систем не може да учита изборни програм он приказује директору поруку: "Систем не може да учита изборни програм. Прекида се извршење сценарија. (ИА)



Слика 122. Систем не може да учита одабрани изборни програм

11.1 Уколико систем не може да обрише изборни програм он приказује директору поруку “Систем не може да обрише изборни програм”. (ИА)



Слика 123. Систем не може да обрише одабрани изборни програм

СК8: Случај коришћења –Креирање похађања(сложен СК)

Назив СК

Креирање похађања

Актори СК

Директор

Учесници СК

Директор и систем (програм)

Предуслов: Систем је укључен и **директор** је пријављен под својом шифром. Систем приказује форму за рад са **похађањем**. Учитана је листа деце и листа изборних програма.

| Ime | Prezime | JMBG |
|---------|--------------|----------------|
| Marko | Petrasinovic | 66666666666666 |
| Elena | Vojvodic | 77777777777777 |
| Lana | Petrovic | 88888888888888 |
| Teodora | Petrovic | 99999999999999 |
| Luša | Vucetic | 1010101010101 |
| Jovan | Vucetic | 1212121212121 |
| Nikola | Jovanovic | 1313131313131 |
| Ana | Anic | 2508024715231 |
| Ana | Petrasinovic | 1111111111112 |
| Marija | Petrasinovic | 2408024715231 |

| Ime | Prezime | Jmbg | Izborni program | Datum upisa | Datum ispisa |
|-----|---------|------|-----------------|-------------|--------------|
| | | | | | |

Слика 124. Форма за креирање новог похађања

Основни сценарио СК

1. Директор уноси податке у похађање. (АПУСО)

Опис акције: Директор на форми бира одговарајуће дете, датум уписа и изборни програм похађања.

Слика 125. Унос података за ново похађање

2. Директор контролише да ли је коректно унео податке у похађање. (АНСО)

Опис акције: Директор кликом на дугме „Додај“ попуњава табелу са похађањима пре самог чувања у бази. Кликом на дугме „Обриши“ може изменити креирану листу похађања, брисањем неодговарајућих.

3. Директор позива систем да запамти податке о похађању. (АПСО)

Опис акције: Директор кликом на дугме „Сачувай сва похађања“ позива системску операцију *ZapamtiPohadjanje(Pohadjanje)*.

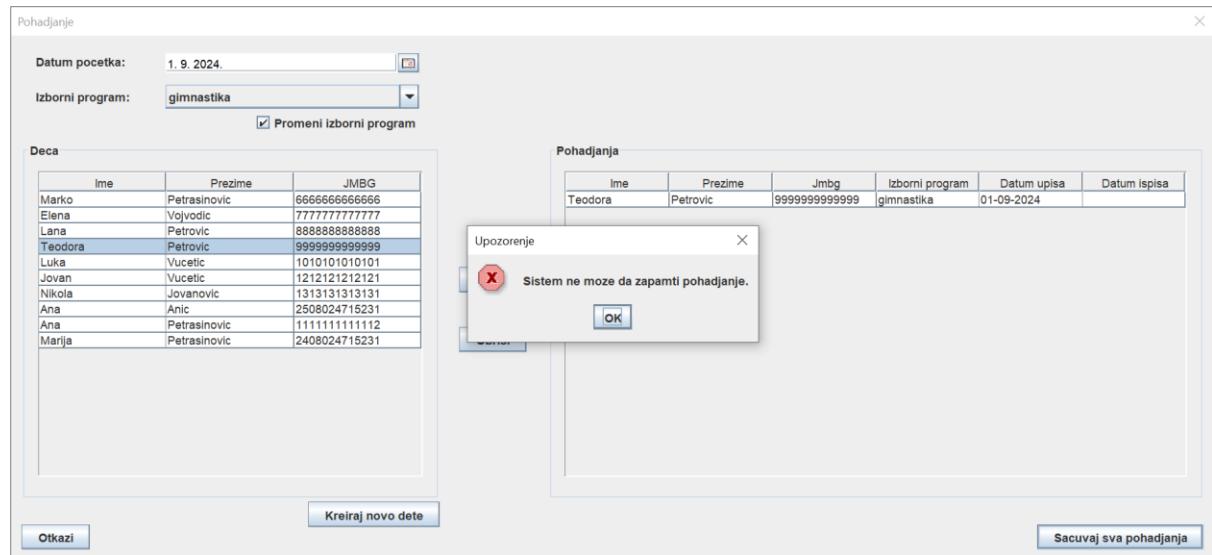
4. Систем памти податке о похађању. (СО)

5. Систем приказује директору запамћено похађање и поруку: “Систем је запамтио похађање“. (ИА)

Слика 126. Успешно креирано похађање

Алтернативна сценарија

5.1 Уколико **систем** не може да запамти податке о **похађању** он приказује **директору** поруку “**Систем** не може да запамти **похађање**”. (ИА)



Слика 127. Систем не може да креира похађање

СК9: Случај коришћења –Измена похађања(сложен СК)

Назив СК

Промена похађања

Актори СК

Директор

Учесници СК

Директор и систем (програм)

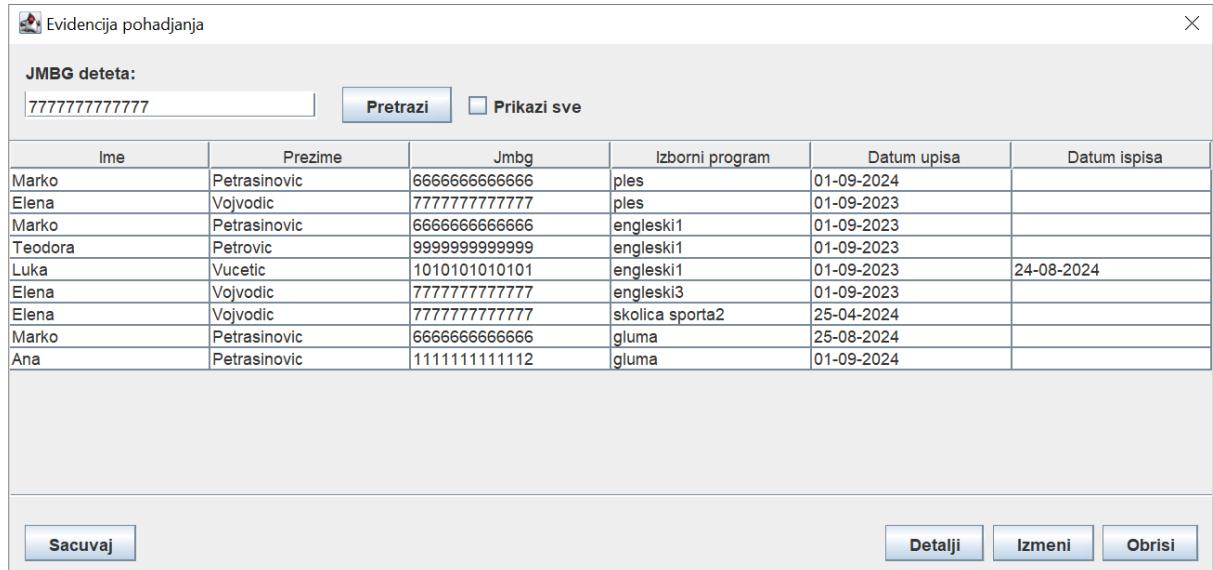
Предуслов: Систем је укључен и директор је пријављен под својом шифром. Систем приказује форму за рад са похађањем. Учитана је листа деце и листа изборних програма.

| Evidencija pohadjanja | | | | | | |
|-----------------------|--------------|---|-----------------|--------------------------------------|--------------|--|
| JMBG deteta: | | <input type="button" value="Pretrazi"/> | | <input type="checkbox"/> Prikazi sve | | |
| Ime | Prezime | Jmbg | Izborni program | Datum upisa | Datum ispisa | |
| Marko | Petrasinovic | 6666666666666 | ples | 01-09-2024 | | |
| Elena | Vojvodic | 7777777777777 | ples | 01-09-2023 | | |
| Marko | Petrasinovic | 6666666666666 | engleski1 | 01-09-2023 | | |
| Teodora | Petrovic | 9999999999999 | engleski1 | 01-09-2023 | | |
| Luka | Vucetic | 1010101010101 | engleski1 | 01-09-2023 | 24-08-2024 | |
| Elena | Vojvodic | 7777777777777 | engleski3 | 01-09-2023 | | |
| Elena | Vojvodic | 7777777777777 | skolica sporta2 | 25-04-2024 | | |
| Marko | Petrasinovic | 6666666666666 | gluma | 25-08-2024 | | |
| Ana | Petrasinovic | 1111111111112 | gluma | 01-09-2024 | | |

Слика 128. Форма за претраживање похађања

Основни сценарио СК

1. **Директор уноси** вредност по којој претражује **похађања**. (АПУСО)
Опис акције: Директор уноси вредност за јмбг детета у поље под називом „ЈМБГ детета”.

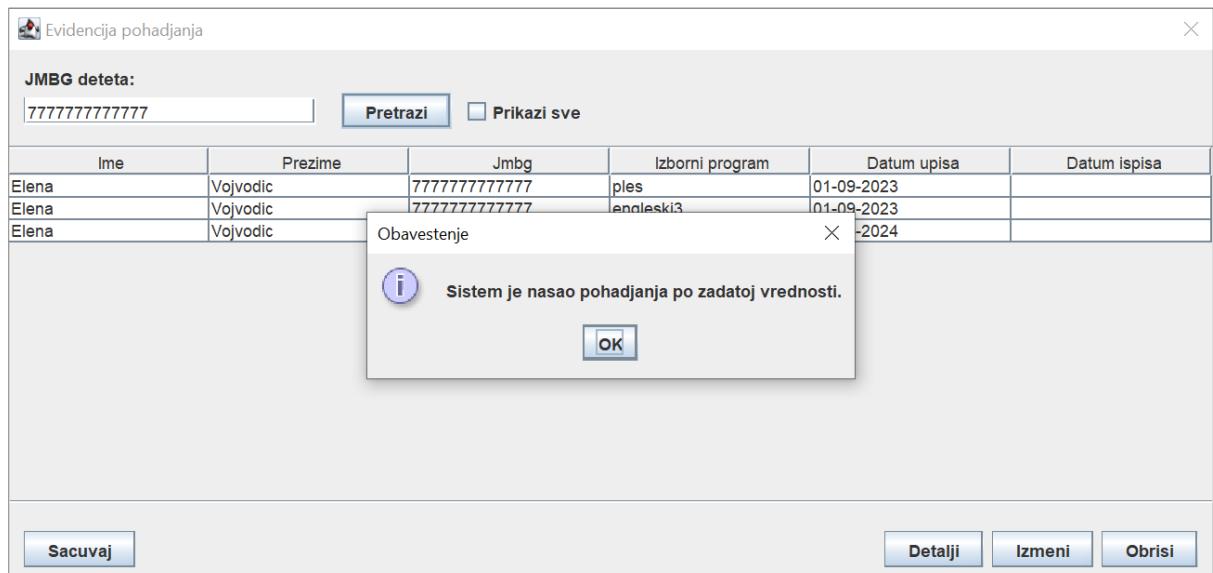


The screenshot shows a Windows application window titled 'Evidencija pohadjanja'. At the top left is a small icon of a document with a red stamp. To its right is the window title. On the far right is a close button (X). Below the title is a section labeled 'JMBG deteta:' with a text input field containing '777777777777' and two buttons: 'Pretrazi' (Search) and 'Prikazi sve' (Show all). The main area is a table with columns: Ime (Name), Prezime (Last Name), Jmbg, Izborni program (Electoral program), Datum upisa (Enrollment date), and Datum ispisa (Issue date). The table contains 10 rows of sample data. At the bottom left is a 'Sacuvaj' (Save) button. At the bottom right are three buttons: 'Detalji' (Details), 'Izmeni' (Edit), and 'Obrisi' (Delete).

| Ime | Prezime | Jmbg | Izborni program | Datum upisa | Datum ispisa |
|---------|--------------|---------------|-----------------|-------------|--------------|
| Marko | Petrasinovic | 666666666666 | ples | 01-09-2024 | |
| Elena | Vojvodic | 777777777777 | ples | 01-09-2023 | |
| Marko | Petrasinovic | 666666666666 | engleski1 | 01-09-2023 | |
| Teodora | Petrovic | 999999999999 | engleski1 | 01-09-2023 | |
| Luka | Vucetic | 1010101010101 | engleski1 | 01-09-2023 | 24-08-2024 |
| Elena | Vojvodic | 777777777777 | engleski3 | 01-09-2023 | |
| Elena | Vojvodic | 777777777777 | skolica sporta2 | 25-04-2024 | |
| Marko | Petrasinovic | 666666666666 | gluma | 25-08-2024 | |
| Ana | Petrasinovic | 111111111112 | gluma | 01-09-2024 | |

Слика 129. Унос вредности за претрагу изборних програма

2. **Директор позива** систем да нађе **похађања** по задатој вредности. (АПСО)
Опис акције: Директор кликом на дугме „Претражи“ позива системску операцију NadjiPohadjanja(Pohadjanje,List<Pohadjanje>)
3. Систем тражи похађања по задатој вредности. (СО)
4. Систем приказује директору похађања и поруку: “Систем је нашао похађања по задатој вредности”. (ИА)



The screenshot shows the same application window as before. The search results grid now displays only two rows of data. A modal dialog box titled 'Obavestenje' (Information) appears in the center. It contains an information icon and the text 'Sistem je nasao pohadjanja po zadatoj vrednosti.' (The system found attendance records by the specified value.) At the bottom of the dialog is an 'OK' button. The bottom buttons 'Sacuvaj', 'Detalji', 'Izmeni', and 'Obrisi' are visible at the bottom of the window.

Слика 130. Систем је пронашао похађања по задатој вредности

5. **Директор бира похађање.** (АПУСО)

Опис акције: Директор бира ред табеле где се налази похађање чије податке жели да учита и измени.

Evidencija pohadjanja

JMBG deteta: Pretrazi Prikazi sve

| Ime | Prezime | Jmbg | Izborni program | Datum upisa | Datum ispisa |
|-------|----------|--------------|-----------------|-------------|--------------|
| Elena | Vojvodic | 777777777777 | ples | 01-09-2023 | |
| Elena | Vojvodic | 777777777777 | engleski3 | 01-09-2023 | |
| Elena | Vojvodic | 777777777777 | skolica sporta2 | 25-04-2024 | |

Sacuvaj Detalji Izmeni Obrisi

Слика 131. Одабир похађања

6. **Директор позива систем да учита похађање.** (АПСО)

Опис акције: Директор кликом на дугме „Измени“ позива системску операцију *UcitajPohadjanje(Pohadjanje)*

7. **Систем учитава похађање.** (СО)

8. **Систем приказује директору податке о похађању и поруку: “Систем је учитао похађање.”** (ИА)

Evidencija pohadjanja

JMBG deteta: Pretrazi Prikazi sve

| Ime | Prezime | Jmbg | Izborni program | Datum upisa | Datum ispisa |
|-------|----------|--------------|-----------------|-------------|--------------|
| Elena | Vojvodic | 777777777777 | ples | 01-09-2023 | |
| Elena | Vojvodic | 777777777777 | engleski3 | 01-09-2023 | |
| Elena | Vojvodic | 777777777777 | Obavestenje | 25-04-2024 | |

Obavestenje

i Sistem je ucitao pohadjanje

OK

Sacuvaj Detalji Izmeni Obrisi

Слика 132. Систем је учитао одабрано похађање

Слика 133. Систем показује податке о одабраном похађању

9. **Директор уноси (мења) податке о похађању.** (АПУСО)

Опис акције: Директор у поља на форми уноси нове вредности датума уписа и исписа или мења одабир детета и изборног програма.

Слика 134. Измена података одабраног похађања

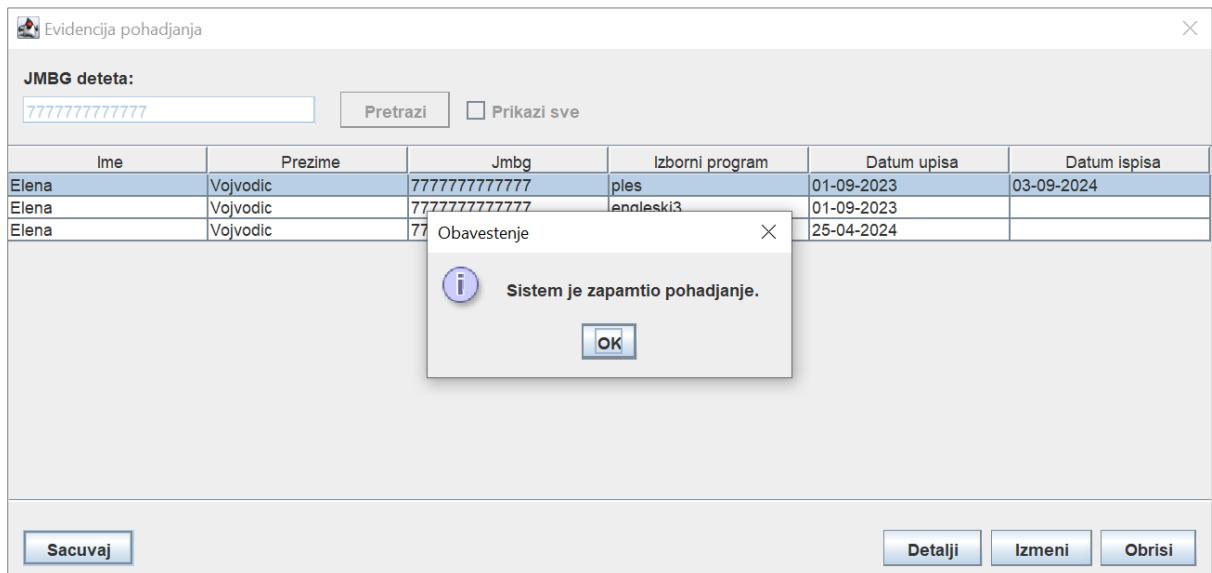
10. **Директор контролише да ли је коректно унео податке о похађању.** (АНСО)

11. **Директор позива систем да запамти податке о похађању.** (АПСО)

Опис акције: Директор кликом на дугме „Сачувати“ позива системску операцију *ZapamtiPohadjanje(Pohadjanje)*.

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

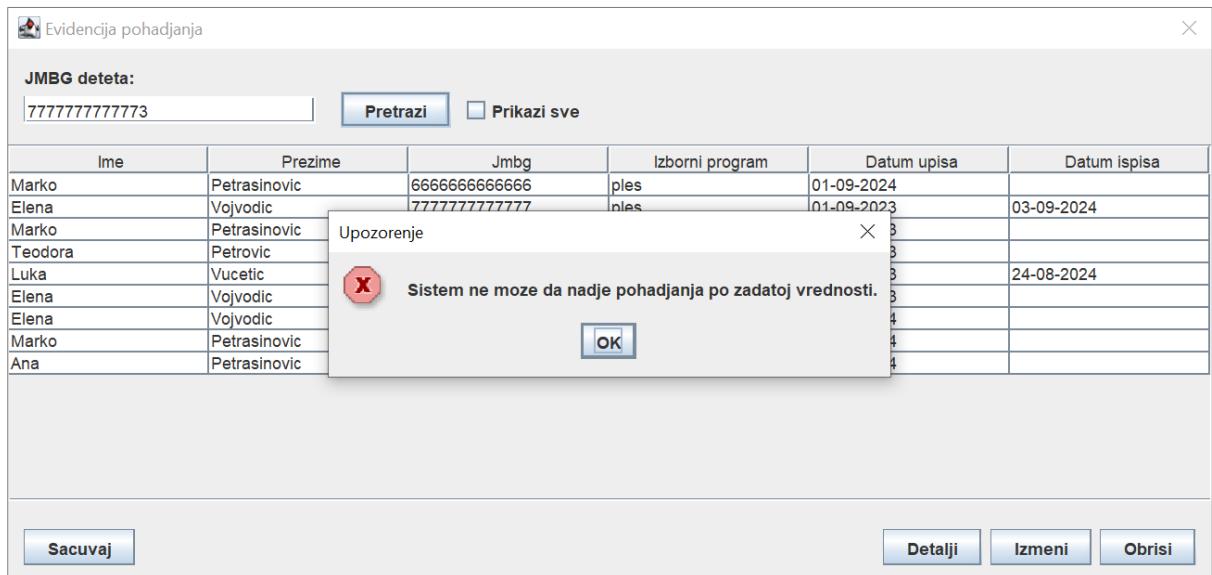
13. Систем приказује директору запамћено похађање и поруку: "Систем је запамтио похађање." (ИА)



Слика 135. Систем је запамтио одабрано похађање

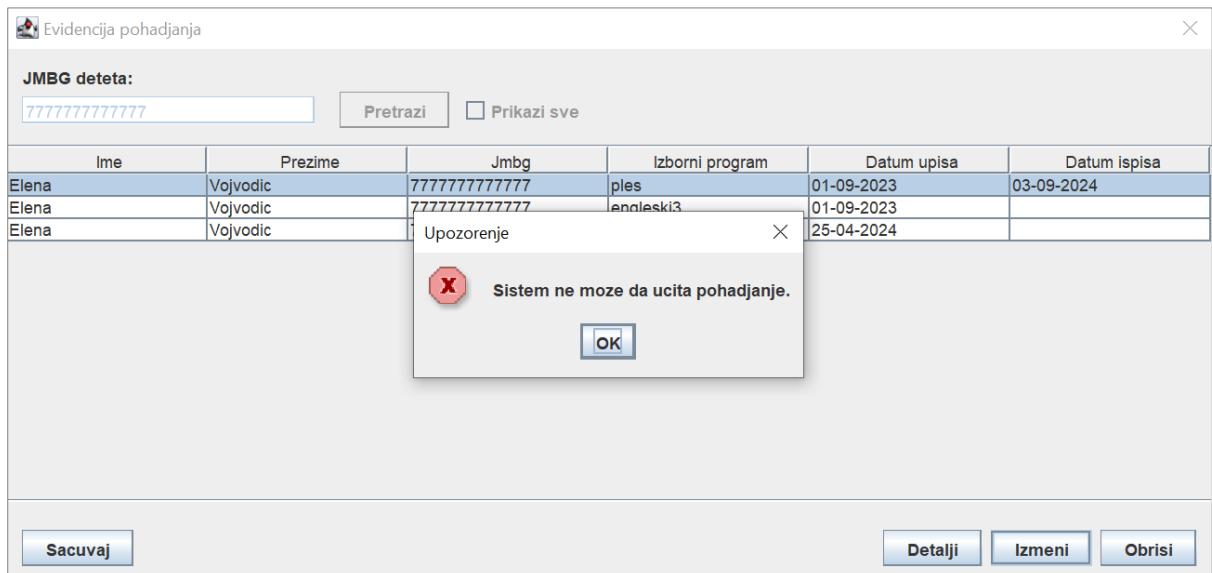
Алтернативна сценарија

4.1 Уколико систем не може да нађе похађања он приказује директору поруку: "Систем не може да нађе похађања по задатој вредности". Прекида се извршење сценарија. (ИА)



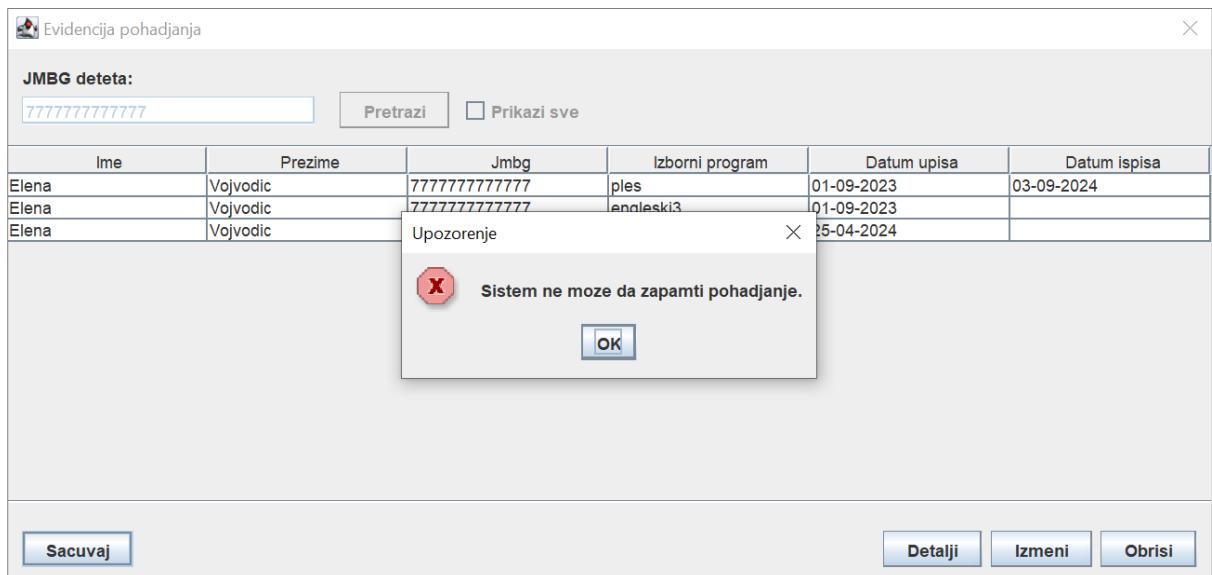
Слика 136. Систем не може да пронађе похађање по задатој вредности

8.1 Уколико систем не може да учита похађање он приказује директору поруку: "Систем не може да учита похађање. Прекида се извршење сценарија. (ИА)



Слика 137. Систем не може да учита одабрано похађање

13.1 Уколико систем не може да запамти податке о похађању он приказује директору "Систем не може да запамти похађање". (ИА)



Слика 138. Систем не може да запамти одабрано похађање

СК10: Случај коришћења – Претраживање похађања

Назив СК

Претраживање похађања

Актори СК

Директор

Учесници СК

Директор и систем (програм)

Предуслов: Систем је укључен и **директор** је пријављен под својом шифром. Систем приказује форму за рад са **похађањем**.

| Ime | Prezime | Jmbg | Izborni program | Datum upisa | Datum ispisa |
|---------|--------------|---------------|-----------------|-------------|--------------|
| Marko | Petrasinovic | 6666666666666 | ples | 01-09-2024 | |
| Elena | Vojvodic | 7777777777777 | ples | 01-09-2023 | |
| Marko | Petrasinovic | 6666666666666 | engleski1 | 01-09-2023 | |
| Teodora | Petrovic | 9999999999999 | engleski1 | 01-09-2023 | |
| Luka | Vucetic | 1010101010101 | engleski1 | 01-09-2023 | 24-08-2024 |
| Elena | Vojvodic | 7777777777777 | engleski3 | 01-09-2023 | |
| Elena | Vojvodic | 7777777777777 | skolica sporta2 | 25-04-2024 | |
| Marko | Petrasinovic | 6666666666666 | gluma | 25-08-2024 | |
| Ana | Petrasinovic | 1111111111112 | gluma | 01-09-2024 | |
| Teodora | Petrovic | 9999999999999 | gimnastika | 01-09-2024 | |

Слика 139. Форма за претраживање похађања

Основни сценарио СК

1. **Директор уноси** вредност по којој претражује **похађања**. (АПУСО)
Опис акције: Директор уноси вредност за јмбг детета у поље под називом „ЈМБГ детета”.

| Ime | Prezime | Jmbg | Izborni program | Datum upisa | Datum ispisa |
|---------|--------------|---------------|-----------------|-------------|--------------|
| Marko | Petrasinovic | 6666666666666 | ples | 01-09-2024 | |
| Elena | Vojvodic | 7777777777777 | ples | 01-09-2023 | |
| Marko | Petrasinovic | 6666666666666 | engleski1 | 01-09-2023 | |
| Teodora | Petrovic | 9999999999999 | engleski1 | 01-09-2023 | |
| Luka | Vucetic | 1010101010101 | engleski1 | 01-09-2023 | 24-08-2024 |
| Elena | Vojvodic | 7777777777777 | engleski3 | 01-09-2023 | |
| Elena | Vojvodic | 7777777777777 | skolica sporta2 | 25-04-2024 | |
| Marko | Petrasinovic | 6666666666666 | gluma | 25-08-2024 | |
| Ana | Petrasinovic | 1111111111112 | gluma | 01-09-2024 | |
| Teodora | Petrovic | 9999999999999 | gimnastika | 01-09-2024 | |

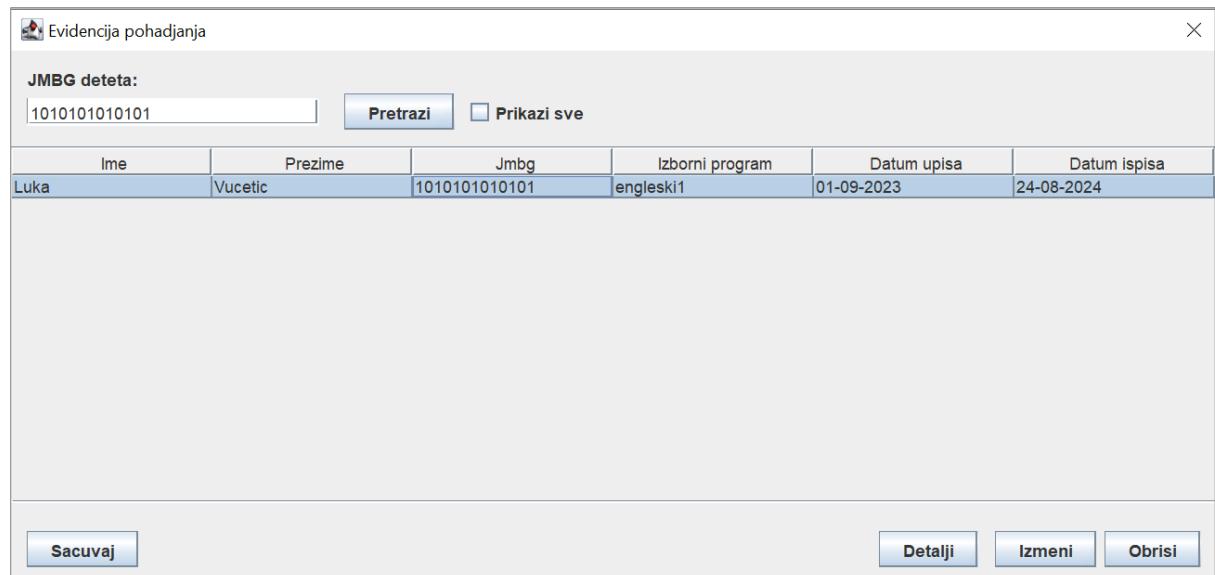
Слика 140. Унос вредности за претрагу похађања

2. **Директор позива** систем да нађе **похађања** по задатој вредности. (АПСО)
Опис акције: Директор кликом на дугме „Претражи“ позива системску операцију NadjiPohadjanja(Pohadjanje,List<Pohadjanje>)
3. Систем тражи похађања по задатој вредности. (СО)
4. Систем приказује **директору** податке о **похађањима** и поруку: “Систем је нашао похађања по задатој вредности”. (ИА)

Слика 141. Систем је пронашао похађања по задатој вредности

5. **Директор бира похађање.** (АПУСО)

Опис акције: Директор ред табеле где се налази похађање чије податке жели да учита.



| Ime | Prezime | Jmbg | Izborni program | Datum upisa | Datum ispisa |
|------|---------|---------------|-----------------|-------------|--------------|
| Luka | Vucetic | 1010101010101 | engleski1 | 01-09-2023 | 24-08-2024 |

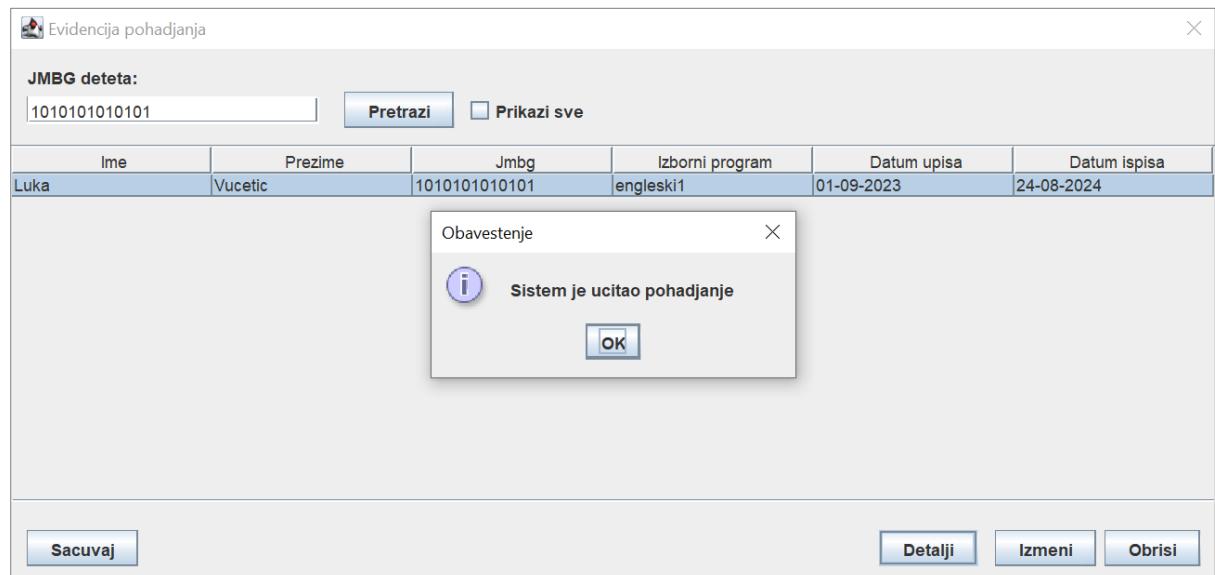
Слика 142. Одабир похађања

6. **Директор позива систем да учита похађање.** (АПСО)

Опис акције: Директор кликом на дугме „Детаљи“ позива системску операцију *UcitajPohadjanje(Pohadjanje)*

7. **Систем учитава похађање.** (СО)

8. **Систем приказује директору податке о похађању и поруку: “Систем је учитао похађање.”** (ИА)



Obavestenje

Sistem je ucitao pohadjanje

OK

Слика 143. Систем је учитао одабрано похађање

Слика 144. Систем приказује податке о одабраном похађању

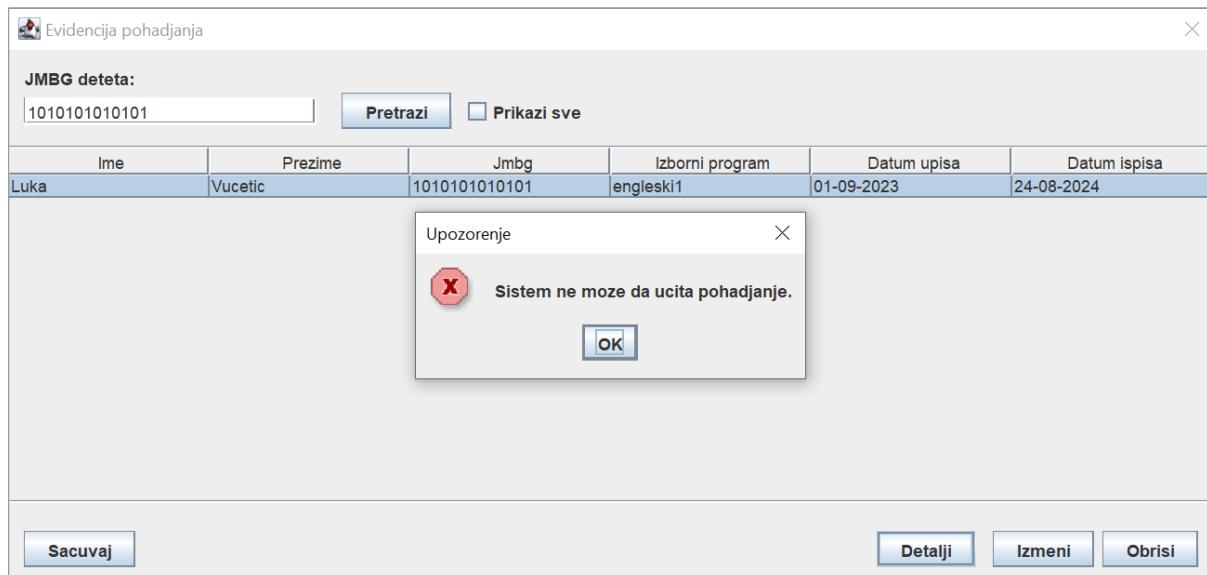
Алтернативна сценарија

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

| Ime | Prezime | Jmbg | Izborni program | Datum upisa | Datum ispisa |
|---------|--------------|---------------|-----------------|-------------|--------------|
| Marko | Petrasinovic | 6666666666666 | ples | 01-09-2024 | |
| Elena | Vojvodic | 7777777777777 | ples | 01-09-2023 | |
| Marko | Petrasinovic | | | | |
| Teodora | Petrovic | | | | |
| Luka | Vucetic | | | | |
| Elena | Vojvodic | | | | |
| Elena | Vojvodic | | | | |
| Marko | Petrasinovic | | | | |
| Ana | Petrasinovic | | | | |
| Teodora | Petrovic | 9999999999999 | gimnastika | 01-09-2024 | |

Слика 145. Систем не може да пронађе похађања по задатој вредности

8.1 Уколико **систем** не може да учита похађање он приказује **директору** поруку:
“Систем не може да учита похађање. (ИА)“



Слика 146. Систем не може да учита одабрано похађање

4.3.2.2 Пројектовање контролера корисничког интерфејса

Контролер корисничког интерфејса има следеће одговорности:

1. Прихвата податке од екранске форме.
2. Конвертује податке из графичких елемената у објекат који ће послужити као улазни аргумент системске операције (СО).
3. Прослеђује захтев за извршење системске операције до апликационог сервера (софтверског система).
4. Прихвата објекат који софтверски систем генерише као резултат извршења системске операције (СО).
5. Конвертује добијени објекат у податке који ће бити приказани у графичким елементима.[2]

4.3.3 Пројектовање апликационе логике

Апликациона логика дефинише структуру и понашање система. Апликациони сервер обухвата следеће компоненте:

1. **Контролер апликационе логике** – одговоран је за покретање серверског сокета који ће ослушкивати мрежне захтеве. Овај контролер служи за комуникацију са клијентом, прихвата захтеве за извршење системских операција и прослеђује их пословној логици, која је задужена за њихово извршење
2. **Пословна логика** – описана је кроз структуру (доменске класе) и понашање (системске операције)
3. **Брокер базе података** – посредује у комуникацији између пословне логике и базе података[2]

4.3.3.1 Контролер апликационе логике

Контролер апликационе логике прихвата захтев за извршење системске операције од клијентске нити и прослеђује га одговарајућим класама задуженим за извршење те операције. По завршетку системске операције, контролер прихвата резултат и шаље га назад клијентској нити која је упутила захтев.[2]

4.3.3.2 Пословна логика

Пројектовање понашања софтверског система (системске операције)

За сваки уговор креирајмо системску операцију која пројектује понашање софтверског система наслеђивањем апстрактне класе *AbstractSO*. Ова класа садржи методу *Execute* која као параметар има *DomainObject* и позива апстрактне методе *validate* и *execute* које свака системска операција имплементира на јединствен начин. Након провере предуслове и извршавања операције, метода *commit* позива истоимену методу над конекцијом *DatabaseBroker-a* која чува измене над подацима у бази, уколико није дошло до грешке. На основу *Response* објекта клијент закључује да ли је операција успешно извршена на серверској страни или је дошло до грешке.

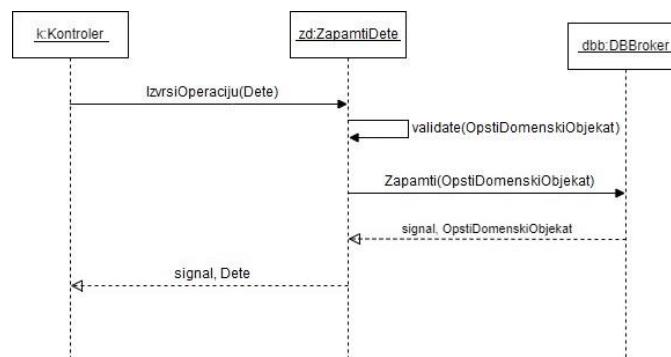
Уговор УГ1: ЗапамтиДете

Операција: *ZapamtiDete(Dete): signal;*

Веза са СК: СК1

Предуслови: Вредносна и структурна ограничења над објектом Дете морају бити задовољена

Постуслови: Дете је запамћено



Слика 147. Уговор УГ1

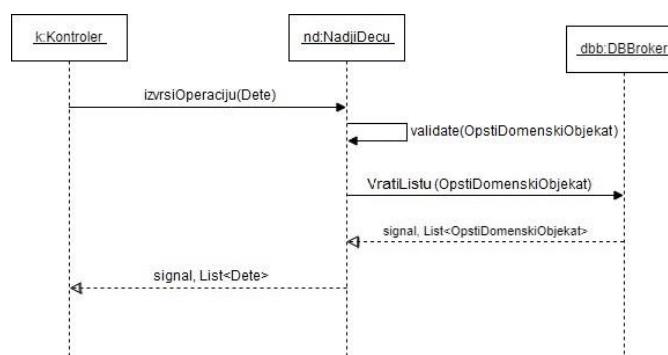
Уговор УГ2: НађиДецу

Операција: *NadjiDecu(Dete, List<Dete>): signal;*

Веза са СК: СК2

Предуслови: Учитана је листа деце

Постуслови: Учитана је листа деце која задовољавају задату вредност



Слика 148. Уговор УГ2

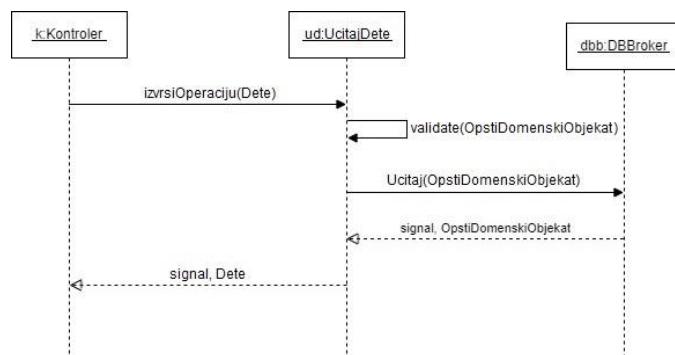
Уговор УГ3: УчитајДете

Операција: UcitajDete(Dete) : signal;

Веза са СК: СК2

Предуслови: /

Постуслови: Учитано је одабрано дете



Слика 149. Уговор УГ3

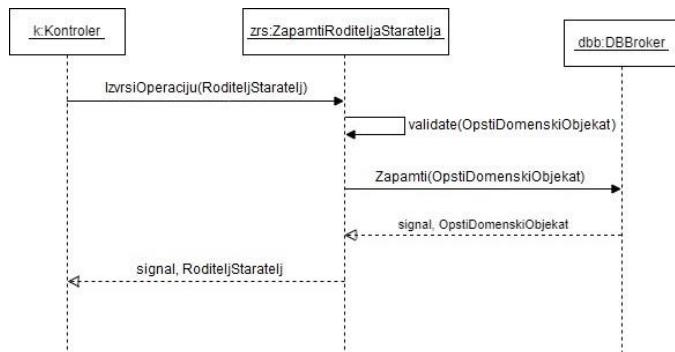
Уговор УГ4: ЗапамтиРодитељаСтаратеља

Операција: ZapamtiRoditeljaStaratelja(RoditeljStaratelj) : signal;

Веза са СК: СК3

Предуслови: Вредносна и структурна ограничења над објектом РодитељСтаратељ морају бити задовољена

Постуслови: РодитељСтаратељ је запамћен



Слика 150. Уговор УГ4

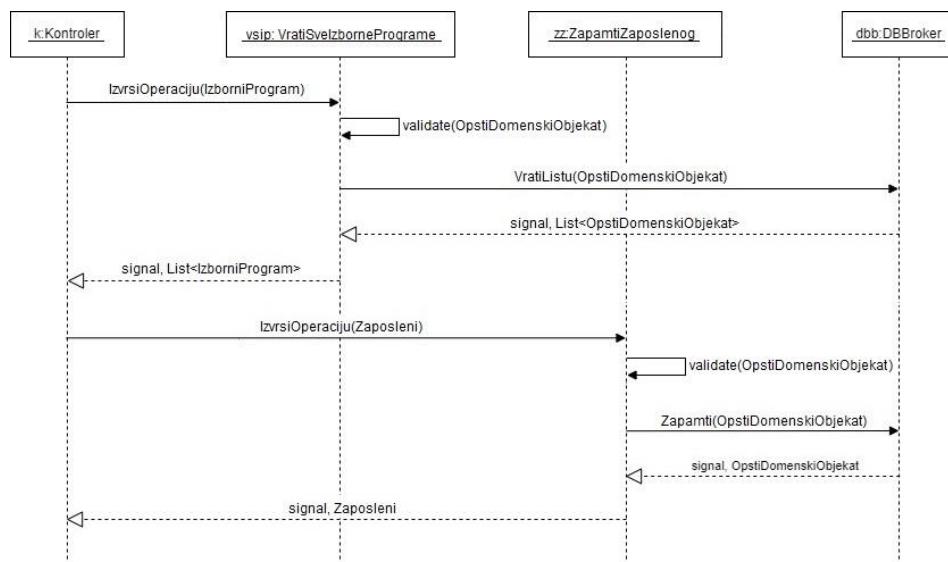
Уговор УГ5: ЗапамтиЗапосленог

Операција: ZapamtiZaposlenog(Zaposleni) : signal;

Веза са СК: СК4

Предуслови: Вредносна и структурна ограничења над објектом Запослени морају бити задовољена

Постуслови: Запослени је запамћен



Слика 151. Уговор УГ5

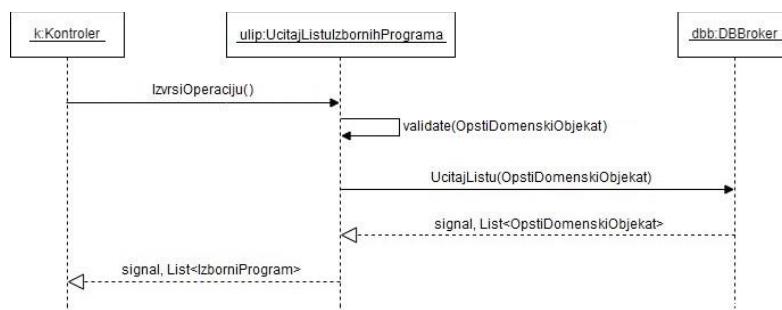
Уговор УГ6: УчитајИзборнеПрограме

Операција: UcitajListuIzbornihPrograma(List<IzborniProgram>): signal;

Веза са СК: СК4, СК8

Предуслови: /

Постуслови: Учитана је листа Изборних програма



Слика 152. Уговор УГ6

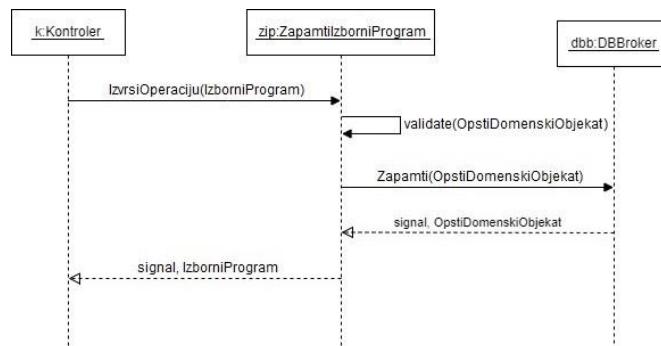
Уговор УГ7: ЗапамтиИзборниПрограм

Операција: ZapamtilzborniProgram(IzborniProgram): signal;

Веза са СК: СК5, СК6

Предуслови: Вредносна и структурна ограничења над објектом Изборни програм морају бити задовољена

Постуслови: Изборни програм је запамћен



Слика 153. Уговор УГ7

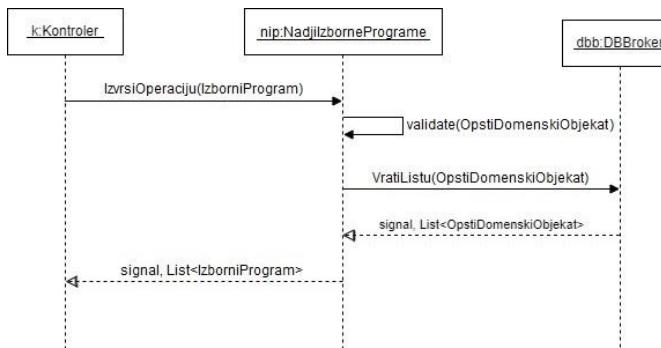
Уговор УГ8: НађиИзборнеПрограме

Операција: NadjilzbornePrograme(IzborniProgram, List<IzborniProgram>): signal;

Веза са СК: СК6, СК7

Предуслови: Учитана је листа изборних програма

Постуслови: Учитана је листа изборних програма који задовољавају задату вредност



Слика 154. Уговор УГ8

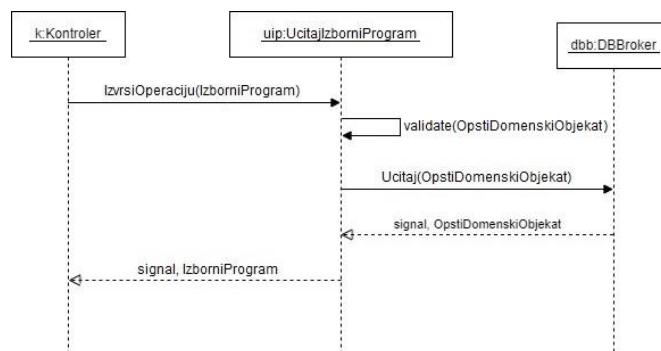
Уговор УГ9: УчитајИзборниПрограм

Операција: UcitajIzborniProgram(IzborniProgram): signal;

Веза са СК: СК6, СК7

Предуслови: /

Постуслови: Учитан је одабрани изборни програм



Слика 155. Уговор УГ9

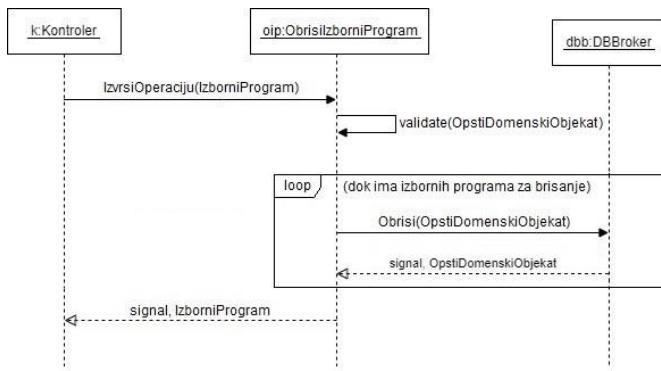
Уговор УГ10: ОбришиИзборниПрограм

Операција: ObrisilzborniProgram(IzborniProgram): signal;

Веза са СК: СК7

Предуслови: Структурна ограничења над објектом Изборни програм морају бити задовољена.

Постуслови: Изборни програм је обрисан



Слика 156. Уговор УГ10

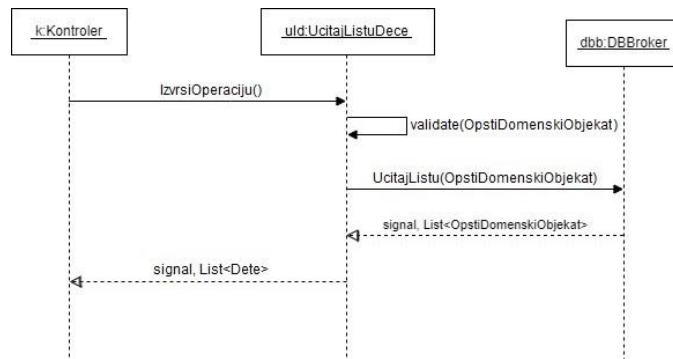
Уговор УГ11: УчитајДецу

Операција: UcitajListuDece(List<Dete>): signal;

Веза са СК: СК8, СК9

Предуслови: /

Постуслови: Учитана је листа деце



Слика 157. Уговор УГ11

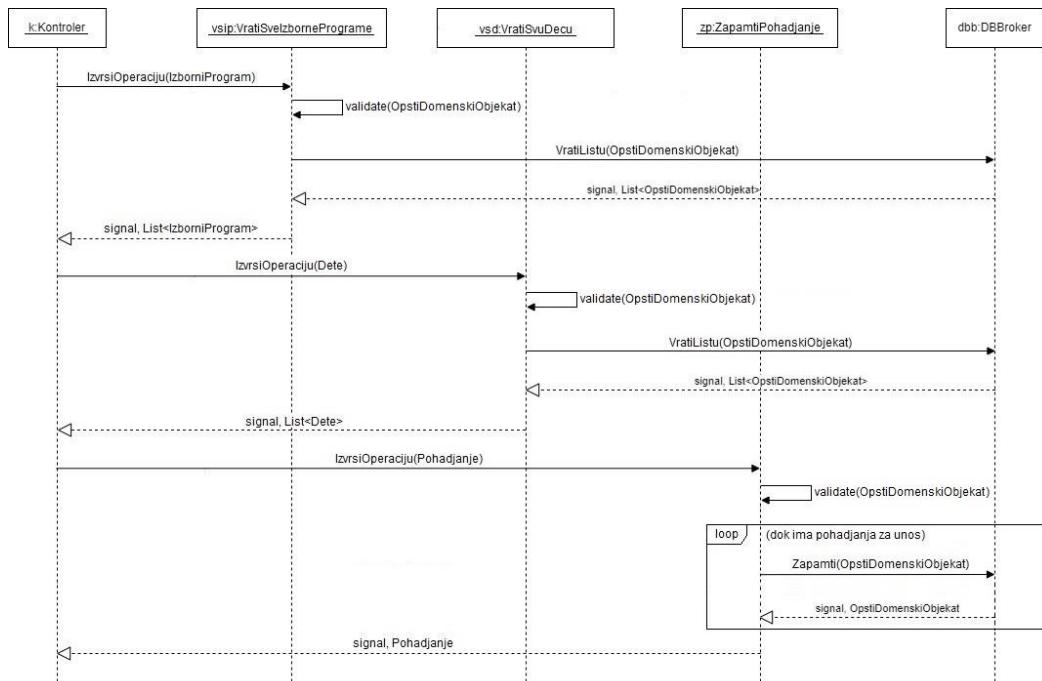
Уговор УГ12: ЗапамтиПохађање

Операција: ZaramtiPohadjanje(Pohadjanje): signal;

Веза са СК: СК8, СК9

Предуслови: Вредносна и структурна ограничења над објектом Похађање морају бити задовољена

Постуслови: Похађање је запамћено



Слика 158. Уговор УГ12

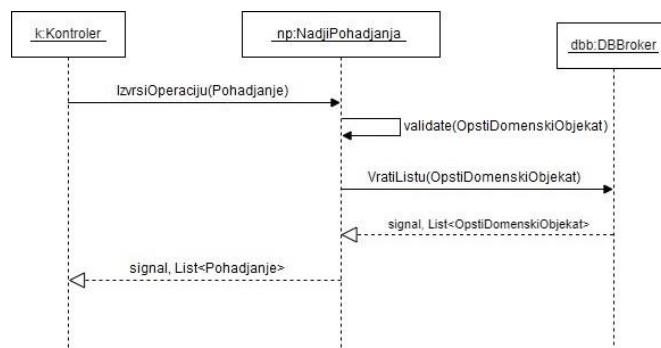
Уговор УГ13: НадјиПохађања

Операција: NadjiPohadjanja(Pohadjanje, List<Pohadjanje>): signal;

Веза са СК: СК9, СК10

Предуслови: Учитана је листа похађања

Постуслови: Учитана је листа похађања која задовољавају задату вредност



Слика 159. Уговор УГ13

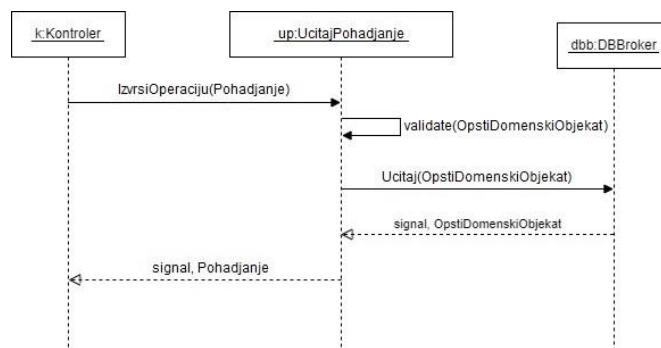
Уговор УГ14: УчитајПохађање

Операција: UcitajPohadjanje(Pohadjanje): signal;

Веза са СК: СК9, СК10

Предуслови: /

Постуслови: Учитано је одабрано похађање



Слика 160. Уговор УГ14

Пројектовање структуре софтверског система (доменске класе)

Софтверске класе структуре:

```
public class UserProfile extends DomainObject implements Serializable {  
  
    private Long userId;  
    private String username;  
    private String password;  
    private Employer employer;  
    private UserStatus userStatus;  
  
    public UserProfile() {}  
  
    public UserProfile(Long userId, String username, String password, Employer employer, UserStatus userStatus) {  
        this.userId = userId;  
        this.username = username;  
        this.password = password;  
        this.employer = employer;  
        this.userStatus = userStatus;  
    }  
}
```

Слика 161. Класа Корисник

```
public class Parent extends DomainObject implements Serializable {  
  
    private Long id;  
    private String firstname;  
    private String lastname;  
    private String adress;  
    private Long phoneNumber;  
    private Long jmbg;  
  
    public Parent() {}  
  
    public Parent(Long id, String firstname, String lastname, String adress, Long phoneNumber, Long jmbg) {  
        this.id = id;  
        this.firstname = firstname;  
        this.lastname = lastname;  
        this.adress = adress;  
        this.phoneNumber = phoneNumber;  
        this.jmbg = jmbg;  
    }  
}
```

Слика 162. Класа Родитељ/старател

```
public class OptionalProgram extends DomainObject implements Serializable {  
  
    private Long id;  
    private String name;  
    private String age;  
    private int difficultyLevel;  
  
    public OptionalProgram() {}  
  
    public OptionalProgram(Long id, String name, String age, int difficultyLevel) {  
        this.id = id;  
        this.name = name;  
        this.age = age;  
        this.difficultyLevel = difficultyLevel;  
    }  
}
```

Слика 163. Класа Изборни програм

```

public class Employer extends DomainObject implements Serializable {

    private Long id;
    private String firstname;
    private String lastname;
    private String adress;
    private Long phoneNumber;
    private String sss;
    private OptionalProgram optionalProgram;

    public Employer() {
    }

    public Employer(Long id, String firstname, String lastname, String adress, Long phoneNumber,
                    String sss, OptionalProgram optionalProgram) {
        this.id = id;
        this.firstname = firstname;
        this.lastname = lastname;
        this.adress = adress;
        this.phoneNumber = phoneNumber;
        this.sss = sss;
        this.optionalProgram = optionalProgram;
    }
}

```

Слика 164. Класа Запослени

```

public class Child extends DomainObject implements Serializable {

    private Long id;
    private String firstname;
    private String lastname;
    private Date birthday;
    private Parent parent;
    private Long jmbg;

    public Child() {
    }

    public Child(Long id, String firstname, String lastname, Date birthday, Parent parent, Long jmbg) {
        this.id = id;
        this.firstname = firstname;
        this.lastname = lastname;
        this.birthday = birthday;
        this.parent = parent;
        this.jmbg = jmbg;
    }
}

```

Слика 165. Класа Дете

```

public class Attendance extends DomainObject implements Serializable {

    private OptionalProgram optionalProgram;
    private Child child;
    private Date startDate;
    private Date endDate;
    private Attendance oldAttendance;

    public Attendance() {
    }

    public Attendance(OptionalProgram optionalProgram, Child child, Date startDate, Date endDate) {
        this.optionalProgram = optionalProgram;
        this.child = child;
        this.startDate = startDate;
        this.endDate = endDate;
    }
}

```

Слика 166. Класа Похађање

```

public enum UserStatus implements Serializable{
    ACTIVE, NOT_ACTIVE, DISABLED
}

```

Слика 167. Класа Статус корисника

Поред ових додате су и следеће класе:

DomainObject - апстрактна класа које све доменске класе наслеђују.

```
public abstract class DomainObject {  
  
    public abstract String getTableName();  
  
    public abstract String getColumnsForAdd();  
  
    public abstract String getParamsForAdd();  
  
    public abstract void setParamsForAdd(PreparedStatement ps, DomainObject domainObject) throws SQLException;  
  
    public boolean containsAutogeneratedKey() {  
        return true;  
    }  
  
    public abstract void setAutogeneratedKey(long key);  
  
    public abstract String getSelectValues();  
  
    public abstract DomainObject setValuesForGet(ResultSet rs);  
  
    public abstract String getUpdateValues();  
  
    public abstract void setValuesForEdit(PreparedStatement ps);  
  
    public abstract String getDeleteValues();  
  
    public abstract void setValuesForDelete(PreparedStatement ps);  
  
}
```

Слика 168. Класа Доменски објекат

Request - објекат за слање података од клијента ка серверу. Састоји се од два атрибута:

1. *argument* - представља објекат над којим треба да се изврши захтевана операција
2. *operationType* - представља операцију која треба да се изврши над објектом

```
public class Request implements Serializable {  
  
    private Object argument;  
    private OperationType operationType;  
  
    public Request() {  
    }  
  
    public Request(Object argument, OperationType operationType) {  
        this.argument = argument;  
        this.operationType = operationType;  
    }
```

Слика 169. Класа Захтев

Response - објекат се користи за слање података од сервера ка клијенту. Садржи два атрибута:

1. *result* - представља објекат који је резултат операције извршене на серверу.
2. *exception* - представља евентуални изузетак који је настао током извршења операције.

```
public class Response implements Serializable{

    private Object result;
    private Exception exception;

    public Response() {
    }

    public Response(Object result, Exception exception) {
        this.result = result;
        this.exception = exception;
    }
}
```

Слика 170. Класа Одговор

4.3.4 Пројектовање складишта података

На основу репационог модела и ограничења пројектоване су табеле базе података које користи наш софтверски систем:

| | Column Name | Data Type | Length | Default | PK? | Not Null? | Unsigned? | Auto Incr? | Zerofill? | On Update | Comment |
|--|-------------|-----------|--------|---------|-------------------------------------|-------------------------------------|-------------------------------------|-------------------------------------|--------------------------|--------------------------|---------|
| | id | bigint | 20 | | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | |
| | username | varchar | 30 | | <input type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | |
| | password | varchar | 30 | | <input type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | |
| | employer_id | bigint | 20 | | <input type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | |
| | user_status | varchar | 30 | | <input type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | |

Слика 171. Табела Корисник

| | Column Name | Data Type | Length | Default | PK? | Not Null? | Unsigned? | Auto Incr? | Zerofill? | On Update | Comment |
|--|--------------|-----------|--------|---------|-------------------------------------|-------------------------------------|-------------------------------------|-------------------------------------|--------------------------|--------------------------|---------|
| | id | bigint | 20 | | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | |
| | firstname | varchar | 30 | | <input type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | |
| | lastname | varchar | 30 | | <input type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | |
| | adress | varchar | 40 | | <input type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | |
| | phone_number | bigint | 20 | | <input type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | |
| | jmbg | bigint | 20 | | <input type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | |

Слика 172. Табела Родитељ/старатељ

| Column Name | Data Type | Length | Default | PK? | Not Null? | Unsigned? | Auto Incr? | Zerofill? | On Update | Comment |
|------------------|-----------|--------|---------|-------------------------------------|-------------------------------------|-------------------------------------|-------------------------------------|--------------------------|--------------------------|---------|
| id | bigint | 20 | | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | |
| name | varchar | 30 | | <input type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | |
| age | varchar | 4 | | <input type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | |
| difficulty_level | int | 11 | | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | |

Слика 173. Табела Изборни програм

| Column Name | Data Type | Length | Default | PK? | Not Null? | Unsigned? | Auto Incr? | Zerofill? | On Update | Comment |
|------------------|-----------|--------|---------|-------------------------------------|-------------------------------------|-------------------------------------|-------------------------------------|--------------------------|--------------------------|---------|
| id | bigint | 20 | | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | |
| firstname | varchar | 30 | | <input type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | |
| lastname | varchar | 30 | | <input type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | |
| adress | varchar | 40 | | <input type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | |
| phone_number | bigint | 20 | | <input type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | |
| sss | varchar | 11 | | <input type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | |
| optional_program | bigint | 20 | | <input type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | |

Слика 174. Табела Запослени

| Column Name | Data Type | Length | Default | PK? | Not Null? | Unsigned? | Auto Incr? | Zerofill? | On Update | Comment |
|-------------|-----------|--------|---------|-------------------------------------|-------------------------------------|-------------------------------------|-------------------------------------|--------------------------|--------------------------|---------|
| id | bigint | 20 | | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | |
| firstname | varchar | 30 | | <input type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | |
| lastname | varchar | 30 | | <input type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | |
| birthday | date | | | <input type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | |
| parent_id | bigint | 20 | | <input type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | |
| jmbg | bigint | 20 | | <input type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | |

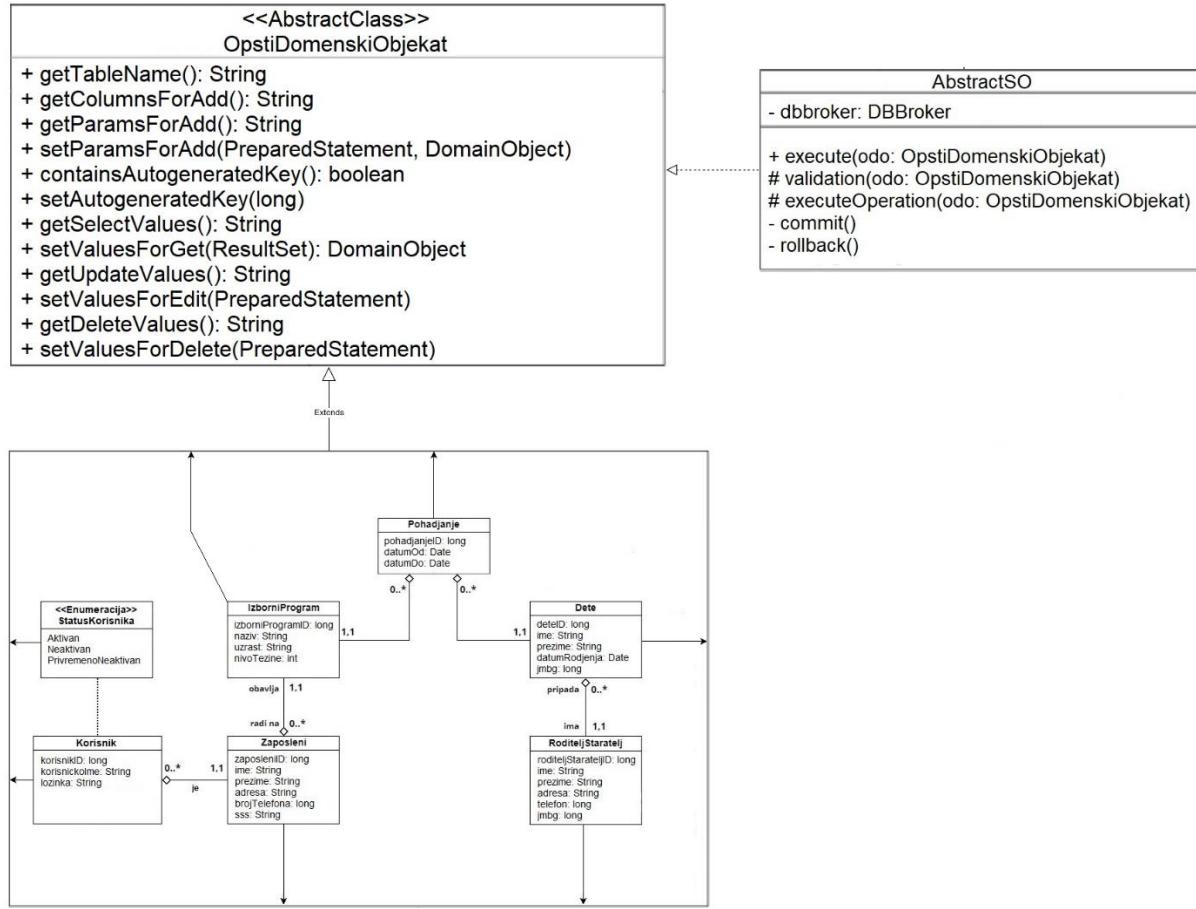
Слика 175. Табела Дете

| Column Name | Data Type | Length | Default | PK? | Not Null? | Unsigned? | Auto Incr? | Zerofill? | On Update | Comment |
|------------------|-----------|--------|---------|-------------------------------------|-------------------------------------|-------------------------------------|--------------------------|--------------------------|--------------------------|---------|
| optional_program | bigint | 20 | | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | |
| child_id | bigint | 20 | | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | |
| start_date | date | | | <input type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | |
| end_date | date | | | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | |

Слика 176. Табела Похађање

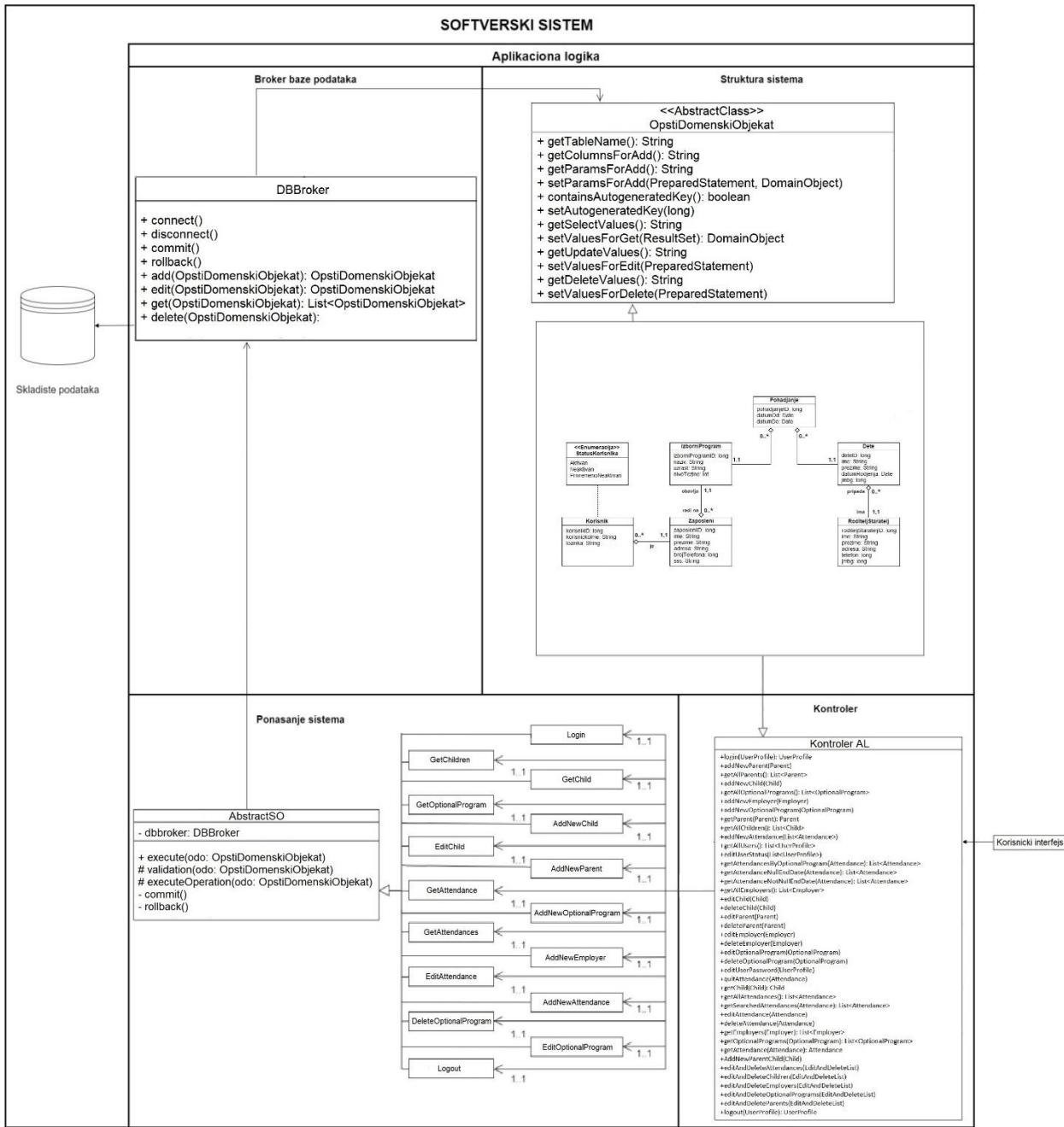
За комуникацију са базом података, креирамо генеричку класу *AbstractSO*, која садржи апстрактне методе за валидацију и извршење трансакција. Те методе ће бити имплементиране у свакој специфичној класи системских операција која се буде извршавала. Поред тога, *AbstractSO* укључује и методе за потврђивање и поништавање трансакција. Класа се ослања на *DBBroker*, која је реализацизована помоћу *Singleton* патерна и која чита параметре за повезивање са базом података из текстуалне датотеке са свим потребним подацима. На овај начин, *AbstractSO* успоставља конекцију са базом података.

Као резултат пројектовања класе *AbstractSO*, *DatabaseConnection* и доменских објеката добијамо следећи дијаграм класа:



Слика 177. Дијаграм класа добијен након пројектовања доменских класа и класе ОпштиДоменскиОбјекат

На основу претходних целина, може се саставити цела архитектура софтверског система за евидентију похађања изборних програма у предшколској установи.



Слика 178. Архитектура софтверског система

4.4 Фаза имплементације

Софтверски систем је развијен у програмском језику Java 21 и пројектован као клијент-сервер апликација. За управљање базом података коришћен је 10.4.32-MariaDB, док је као развојно окружење коришћен NetBeans IDE 19. На основу архитектуре система дефинисане су следеће софтверске класе:

ClientApp:

Images/pozadina_resized_1
main/Main
thread/SSSExplanationThread
thread/TimeThread
uicontroller/Controller
view.components/AttendancesTableModel
view.components/ChildrenTableModel
view.components/EmployersTableModel
view.components/OptionalProgramsTableModel
view.components/ParentsTableModel
view.components/SelectChildTableModel
view.components/SpecificOptionalProgramTableModel
view.components/UserProfileTableModel
view.form/AddNewAttendanceForm
view.form/AddNewChildParentForm
view.form/AddNewEmployerForm
view.form/AddNewOptionalProgramForm
view.form/DetailsAttendanceForm
view.form/DetailsChildrenForm
view.form/DetailsEmployersForm
view.form/DetailsOptionalProgramForm
view.form/EditAttendanceForm
view.form/EditChildForm
view.form/EditParentForm
view.form/EditPasswordForm
view.form/EditUserStatusForm
view.form/LoginForm

view.form/MainForm
view.form/UsersStatusForm
view.form/ViewAllAttendanceForm
view.form/ ViewAllChildrenForm
view.form/ ViewAllEmployersForm
view.form/ ViewAllOptionalProgramForm
view.form/ ViewAllParentsForm
view.form.util/Mode

CommonLib:

communication/OperationType
communication/Receiver
communication/Request
communication/Response
communication/Sender
domain/Attendance
domain/Child
domain/DomainObject
domain/EditAndDeleteList
domain/Employer
domain/OptionalProgram
domain/Parent
domain/UserProfile
domain/UserStatus

ServerApp:

config/dbconfig.properties
constants/ServerConstants
controller/Controller
main/Main
repository/DBBroker
repository/DBConnection
so/AbstractSO
so/AddNewAttendance

so/AddNewChild
so/AddNewEmployer
so/AddNewOptionalProgram
so/AddNewParent
so/AddNewParentChild
so/DeleteAttendance
so/DeleteChild
so/DeleteEmployer
so/DeleteOptionalProgram
so/DeleteParent
so/EditAndDeleteAttendances
so/EditAndDeleteChildren
so/EditAndDeleteEmployers
so/EditAndDeleteOptionalPrograms
so/EditAndDeleteParents
so/EditAttendance
so/EditChild
so/EditEmployer
so/EditOptionalProgram
so/EditParent
so/EditUserPassword
so/EditUserStatus
so/GetAllAttendances
so/GetAllChildren
so/GetAllEmployers
so/GetAllOptionalPrograms
so/GetAllParents
so/GetAllUsers
so/GetAttendance
so/GetAttendancesByOptionalProgram
so/GetAttendancesByOptionalProgramNotNullEndDate
so/GetAttendancesByOptionalProgramNullEndDate

so/GetChild
so/GetEmployers
so/GetOptionalPrograms
so/GetParent
so/GetSearchedAttendances
so/Login
so/QuitAttendance
thread/ClientThread
thread/LoggedInUsersThread
thread/ServerThread
view.components/LoggedInUsersTableModel
view.form/MainServerForm

4.5 Тестирање

Током фазе тестирања проверавани су имплементирани случајеви коришћења. За сваки случај коришћења, поред правилно унетих података, уношени су и неправилни подаци како би се проверило да ли систем реагује на адекватан начин. По завршетку ове фазе, софтвер је спреман за употребу од стране крајњег корисника.

JUnit је широко коришћени алат који се користи за тестирање кода у Јави. Он помаже у верификацији исправности програма тако што се могу написати и покренути тестови који независно тестирају мање, јединице кода (нпр. методе или функције). Овај оквир омогућава тестирање појединачних компоненти апликације како би се осигурала исправност и предвиђене функционалности. Основна сврха јединичног тестирања је рано откривање грешака у коду, како би се проблеми идентификовали у почетним фазама развоја. Појединачним тестирањем сваке компоненте, програмери могу брзо отклонити недостатке пре него што се пређе на развој других делова система. Јединични тестови пружају додатну сигурност, осигурујући стабилност и поузданост.

У JUnit-у, методе које представљају тестове означавају се анотацијом `@Test`. Ова анотација омогућава JUnit-у да препозна те методе као тест случајеве. Свака метода обележена са `@Test` биће извршена током тестирања, а резултати (успешни или неуспешни) биће приказани програмеру.

Анотација `@Test(expected = Exception.class)` се користи за тестирање очекиваних изузетака. Коришћењем ове анотације очекујете се да ће тестирана метода бацити одређену грешку (`exception`). Ако се очекивана изузетна ситуација (грешка) не деси током извршавања теста, тест ће бити сматран неуспешним.

JUnit обезбеђује и анотације `@Before` и `@After`, које се користе за припрему и очување доследности након извршења сваког теста.

- `@Before`: Метода означена овом анотацијом ће се извршити пре сваког теста. Најчешће се користи за подешавање почетног стања, као што је креирање потребних објеката или иницијализација података.
- `@After`: Ова метода се извршава након сваког теста. Најчешће се користи за затварање отворених ресурса (нпр. база података или датотека).

За потребе тестирања овог рада коришћен је Junit 4.13.2.

У наставку ће бити приказана класа *LoginTest* која служи за проверу исправности пријаве на систем. Тестови су пројектовани да покрију неколико кључних аспеката, укључујући успешну и неуспешну пријаву корисника, валидацију корисничких података, као и исправно руковање грешкама које могу настати услед некоректних или недостајућих података.

Метода *setUp()* иницијализује неопходне објекте пре сваког тестирања како би се обезбедило да сваки тест почиње са независним окружењем. Док метода *tearDown()*, након сваког тестирања, ресетује објекте на вредност *null*.

Функционалност методе *executeOperation()* тестирана је кроз два сценарија. У тесту *testExecuteOperationSuccessful()* проверава се успешна пријава корисника са исправним подацима. Овај тест користи методу *assertNotNull()* да би осигурао да је корисник успешно учитан након пријаве, као и методу *assertEquals()* да провери да ли су унети подаци коректни. Са друге стране, тест *testExecuteOperationUnsuccessful()* проверава неуспешну пријаву услед нетачне лозинке, где се очекује бацање изузетка, што је дефинисано помоћу анотације *@Test(expected = Exception.class)*.

Метода *validation* је такође тестирана кроз различите сценарије. Тестом *testValidationValidObject()* проверава се валидност исправног објекта, док *testValidationInvalidObject()* тестира шта се дешава када уместо очекиваног објекта корисник проследи неважећи тип податка, што резултира бацање изузетка. Тестови *testValidationEmptyUsername()* и *testValidationEmptyPassword()* проверавају понашање метода када су корисничко име или лозинка празни, односно постављени на *null*, а тестови *testValidationInvalidUsername()* и *testValidationInvalidPassword()*, тестирају шта се дешава ако корисничко име или лозинка премашују дозвољену дужину карактера. У оба случаја очекује се да метода баци изузетак, чиме се осигурује да програм правилно управља неважећим подацима.

Тестирање системске операције *Login*:

```
17 public class LoginTest {
18
19     private Login so;
20     private UserProfile user;
21
22     @Before
23     public void setUp() throws Exception {
24         so = new Login();
25         user = new UserProfile();
26     }
27
28     @After
29     public void tearDown() {
30         so = null;
31         user = null;
32     }
33
34     @Test
35     public void testExecuteOperationSuccessful() {
36         try {
37             user.setUsername(username:"ana");
38             user.setPassword(password:"ana123");
39             so.executeOperation(object: user);
40             assertNotNull(object: so.getUser());
41             assertEquals(expected: "ana", actual: so.getUser().getUsername());
42             assertEquals(expected: "ana123", actual: so.getUser().getPassword());
43         } catch (Exception ex) {
44             fail(message: ex.getMessage());
45         }
46     }
47
48     @Test(expected = Exception.class)
49     public void testExecuteOperationUnsuccessful() throws Exception {
50         user.setUsername(username:"ana");
51         user.setPassword(password:"ana1234");
52         so.executeOperation(object: user);
53     }
```

Слика 179а. Тестирање системске операције *Login*

```
55 @Test
56 public void testValidationValidObject() {
57     try {
58         user.setUsername(username: "username");
59         user.setPassword(password: "password");
60         so.validation(object: user);
61     } catch (Exception ex) {
62         fail(message: ex.getMessage());
63     }
64 }
65
66 @Test(expected = Exception.class)
67 public void testValidationInvalidObject() throws Exception {
68     so.validation(new Object());
69 }
70
71 @Test(expected = Exception.class)
72 public void testValidationEmptyUsername() throws Exception {
73     user.setUsername(username: null);
74     user.setPassword(password: "password");
75     so.validation(object: user);
76 }
77
78 @Test(expected = Exception.class)
79 public void testValidationEmptyPassword() throws Exception {
80     user.setUsername(username: "username");
81     user.setPassword(password: null);
82     so.validation(object: user);
83 }
84
85 @Test(expected = Exception.class)
86 public void testValidationInvalidUsername() throws Exception {
87     user.setUsername(username: "usernamewithmorethanthirtycharacters");
88     user.setPassword(password: "password");
89     so.validation(object: user);
90 }
```

Слика 1796. Тестирање системске операције Login

```
92     @Test(expected = Exception.class)
93     public void testValidationInvalidPassword() throws Exception {
94         user.setUsername(username: "username");
95         user.setPassword(password: "passwordwithmorethanthirtycharacters");
96         so.validation(object: user);
97     }
98
99 }
100
101
102
```

Output - ServerApp (test) Test Results **X** Search Results

so.EditAndDeleteAttendancesTest X so.AddNewAttendanceTest X so.LoginTest **X**

Tests passed: 100,00 %

All 8 tests passed. (0,304 s)

so.LoginTest **passed**

- ✓ testValidationValidObject **passed** (0,198 s)
- ✓ testValidationInvalidObject **passed** (0,0 s)
- ✓ testValidationEmptyUsername **passed** (0,0 s)
- ✓ testValidationEmptyPassword **passed** (0,0 s)
- ✓ testValidationInvalidUsername **passed** (0,0 s)
- ✓ testValidationInvalidPassword **passed** (0,0 s)
- ✓ testExecuteOperationUnsuccessful **passed** (0,017 s)
- ✓ testExecuteOperationSuccessful **passed** (0,003 s)

Слика 179в. Тестирање системске операције Login

Тестирање системске операције *GetAllAttendances*:

```
19 public class GetAllAttendancesTest {
20
21     private GetAllAttendances so;
22
23     @Before
24     public void setUp() throws Exception {
25         so = new GetAllAttendances();
26     }
27
28     @After
29     public void tearDown() {
30         so = null;
31     }
32
33     @Test
34     public void testExecuteOperation() {
35         try {
36             so.executeOperation(new Attendance());
37             List<Attendance> list = so.getAttendances();
38             assertNotNull(object: list);
39             assertTrue(list.size()>0);
40         } catch (Exception ex) {
41             fail(message: ex.getMessage());
42         }
43     }
44
45     @Test
46     public void testValidationValidObject() {
47         try {
48             so.validation(new Attendance());
49         } catch (Exception ex) {
50             fail(message: ex.getMessage());
51         }
52     }
}
```

Слика 180а. Тестирање системске операције *GetAllAttendances*

```
54     @Test(expected = Exception.class)
55     public void testValidationInvalidObject() throws Exception {
56         so.validation(new Object());
57     }
58 }
59
60
```

Output - ServerApp (test) Test Results X Search Results

so.EditAndDeleteAttendancesTest X so.AddNewAttendanceTest X so.LoginTest X so...

Tests passed: 100,00 %

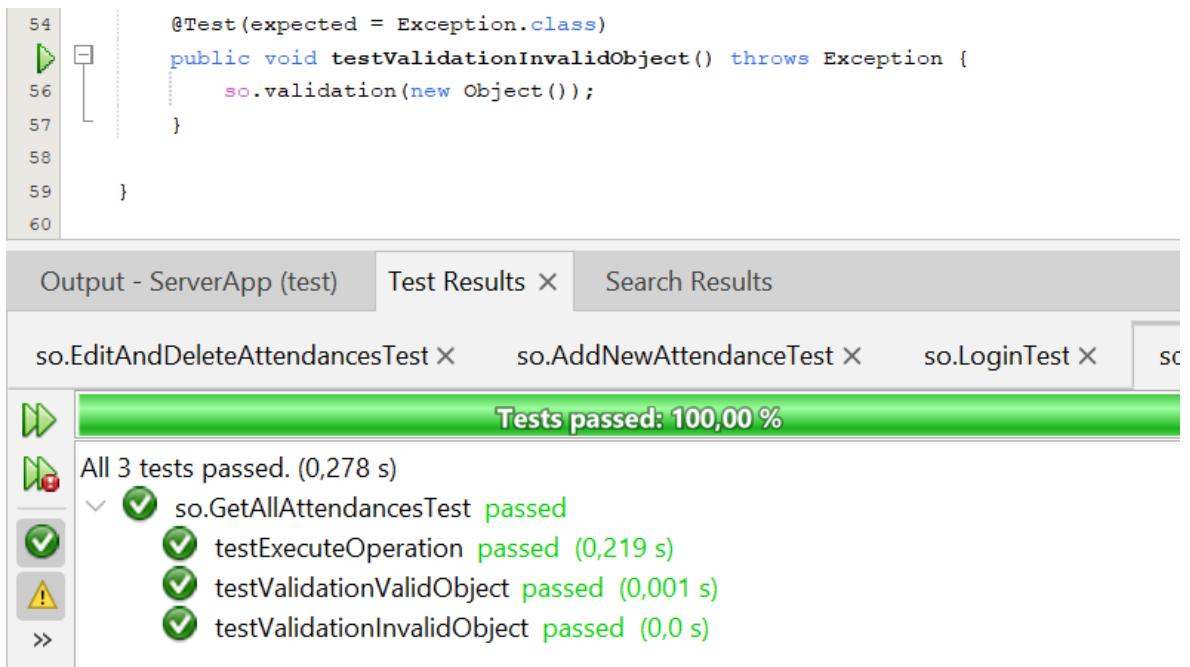
All 3 tests passed. (0,278 s)

 so.GetAllAttendancesTest passed

 testExecuteOperation passed (0,219 s)

 testValidationValidObject passed (0,001 s)

 testValidationInvalidObject passed (0,0 s)



Слика 1806. Тестирање системске операције GetAllAttendances

Тестирање системске операције *EditAndDeleteAttendances*:

```
24 public class EditAndDeleteAttendancesTest {  
25  
26     private EditAndDeleteAttendances so;  
27     private EditAndDeleteList editAndDeleteList;  
28     private Attendance attendance;  
29     List<Attendance> editedA;  
30     List<Attendance> deletedA;  
31  
32     @Before  
33     public void setUp() throws Exception {  
34         so = new EditAndDeleteAttendances();  
35         attendance = new Attendance();  
36         editedA = new ArrayList<>();  
37         deletedA = new ArrayList<>();  
38         editAndDeleteList =  
39             new EditAndDeleteList(editList: editedA, deleteList: deletedA);  
40     }  
41  
42     @After  
43     public void tearDown() {  
44         so = null;  
45         attendance = null;  
46         editedA = null;  
47         deletedA = null;  
48         editAndDeleteList = null;  
49     }  
}
```

Слика 181а. Тестирање системске операције *EditAndDeleteAttendances*

```
51  | @Test
52  | public void testExecuteOperation() throws Exception {
53  |     // Kreiramo objekat za izmenu
54  |     OptionalProgram optionalProgram1 = new OptionalProgram();
55  |     optionalProgram1.setId(id: 61);
56  |     Child child1 = new Child();
57  |     child1.setId(id: 21);
58  |     Attendance attendanceToEdit = new Attendance();
59  |     attendance.setChild(child: child1);
60  |     attendance.setOptionalProgram(optionalProgram: optionalProgram1);
61  |     attendanceToEdit.setChild(child: child1);
62  |     attendanceToEdit.setOptionalProgram(optionalProgram: optionalProgram1);
63  |     attendanceToEdit.setStartDate(new java.sql.Date(date: new Date().getTime()));
64  |     attendanceToEdit.setEndDate(new java.sql.Date(date: new Date().getTime()));
65  |     attendanceToEdit.setOldAttendance(oldAttendance: attendance);
66  |     editedA.add(e: attendanceToEdit);
67  |
68  |     // Kreiramo objekat za brisanje
69  |     OptionalProgram optionalProgram2 = new OptionalProgram();
70  |     optionalProgram2.setId(id: 41);
71  |     Child child2 = new Child();
72  |     child2.setId(id: 21);
73  |     Attendance attendanceToDelete = new Attendance();
74  |     attendanceToDelete.setChild(child: child2);
75  |     attendanceToDelete.setOptionalProgram(optionalProgram: optionalProgram2);
76  |     deletedA.add(e: attendanceToDelete);
77  |
78  |     editAndDeleteList.setEditList(editList: editedA);
79  |     editAndDeleteList.setDeleteList(deleteList: deletedA);
80  |     so.executeOperation(object: editAndDeleteList);
81  |     so.broker.getConnection().commit();
```

Слика1816. Тестирање системске операције *EditAndDeleteAttendances*:

```
83     GetAllAttendances getAllAttendances = new GetAllAttendances();
84     getAllAttendances.executeOperation(new Attendance());
85     List<Attendance> allAttendances = getAllAttendances.getAttendances();
86
87     // Proveravamo da li objekat za izmenu postoji i da je ažuriran
88     Attendance editedFromDB = null;
89     for (Attendance att : allAttendances) {
90         if (Objects.equals(a: att.getChild().getId(),
91                            b: attendanceToEdit.getChild().getId())
92             && Objects.equals(a: att.getOptionalProgram().getId(),
93                                b: attendanceToEdit.getOptionalProgram().getId())) {
94             editedFromDB = att;
95             break;
96         }
97     }
98     assertNotNull(object: editedFromDB);
99     assertEquals(expected: optionalProgram1.getId(),
100                  actual: editedFromDB.getOptionalProgram().getId());
101    assertEquals(expected: child1.getId(),
102                  actual: editedFromDB.getChild().getId());
103    assertEquals(expected: attendanceToEdit.getStartDate(),
104                  actual: editedFromDB.getStartDate());
105    assertEquals(expected: attendanceToEdit.getEndDate(),
106                  actual: editedFromDB.getEndDate());
```

Слика 181e. Тестирање системске операције *EditAndDeleteAttendances*

```
// Proveravamo da objekat za brisanje ne postoji u bazi
Attendance deletedFromDB = null;
for (Attendance att : allAttendances) {
    if (Objects.equals(a: att.getChild().getId(),
                        b: attendanceToDelete.getChild().getId())
        && Objects.equals(a: att.getOptionalProgram().getId(),
                           b: attendanceToDelete.getOptionalProgram().getId())) {
        deletedFromDB = att;
        break;
    }
}
assertNull(object: deletedFromDB);
}

@Test
public void testValidationValidObject() {
    try {
        OptionalProgram optionalProgram = new OptionalProgram();
        Child child = new Child();
        attendance.setChild(child);
        attendance.setOptionalProgram(optionalProgram);
        attendance.setStartDate(new Date());
        attendance.setEndDate(new Date());
        editedA.add(e: attendance);
        editAndDeleteList.setEditList(editList: editedA);
        deletedA.add(e: attendance);
        editAndDeleteList.setDeleteList(deleteList: deletedA);
        so.validation(object: editAndDeleteList);
    } catch (Exception ex) {
        fail(message:ex.getMessage());
    }
}
```

Слика 181г. Тестирање системске операције *EditAndDeleteAttendances*

```
141     @Test(expected = Exception.class)
142     public void testValidationInvalidObject() throws Exception {
143         so.validation(new Object());
144     }
145
146     @Test(expected = Exception.class)
147     public void testValidationEmptyChild() throws Exception {
148         OptionalProgram optionalProgram = new OptionalProgram();
149         Child child = null;
150         attendance.setChild(child);
151         attendance.setOptionalProgram(optionalProgram);
152         attendance.setStartDate(new Date());
153         editedA.add(e: attendance);
154         editAndDeleteList.setEditList(editList: editedA);
155         so.validation(object: editAndDeleteList);
156     }
157
158     @Test(expected = Exception.class)
159     public void testValidationEmptyStartDate() throws Exception {
160         OptionalProgram optionalProgram = new OptionalProgram();
161         Child child = new Child();
162         attendance.setChild(child);
163         attendance.setOptionalProgram(optionalProgram);
164         attendance.setStartDate(startDate: null);
165         editedA.add(e: attendance);
166         editAndDeleteList.setEditList(editList: editedA);
167         so.validation(object: editAndDeleteList);
168     }
```

Слика 181д. Тестирање системске операције *EditAndDeleteAttendances*

The screenshot shows an IDE interface with two main panes. The top pane displays Java test code for a class named `so.EditAndDeleteAttendancesTest`. The code includes annotations like `@Test(expected = Exception.class)` and methods such as `testValidationEmptyOptionalProgram()`. The bottom pane shows the test results for the `so>EditAndDeleteAttendancesTest` class. A green header bar indicates "Tests passed: 100,00 %". Below it, a tree view lists all six tests under the class node, each marked with a green checkmark and labeled "passed".

```
170     @Test(expected = Exception.class)
171     public void testValidationEmptyOptionalProgram() throws Exception {
172         OptionalProgram optionalProgram = null;
173         Child child = new Child();
174         attendance.setChild(child);
175         attendance.setOptionalProgram(optionalProgram);
176         attendance.setStartDate(new Date());
177         editedA.add(e: attendance);
178         editAndDeleteList.setEditList(editList: editedA);
179         so.validation(object: editAndDeleteList);
180     }
181 }
```

Output - ServerApp (test) | Search Results | Test Results X

so.EditAndDeleteAttendancesTest X

Tests passed: 100,00 %

All 6 tests passed. (0,815 s)

- so.EditAndDeleteAttendancesTest passed
 - testValidationValidObject passed (0,592 s)
 - testValidationEmptyChild passed (0,002 s)
 - testExecuteOperation passed (0,07 s)
 - testValidationEmptyStartDate passed (0,001 s)
 - testValidationEmptyOptionalProgram passed (0,0 s)
 - testValidationInvalidObject passed (0,001 s)

Слика 181 Џ. Тестирање системске операције `EditAndDeleteAttendances`

Тестирање системске операције *AddNewAttendance*:

```
22 public class AddNewAttendanceTest {
23
24     private AddNewAttendance so;
25     private GetAllAttendances getAllAttendances;
26     private Attendance attendance;
27     List<Attendance> attendances;
28
29     @Before
30     public void setUp() throws Exception {
31         so = new AddNewAttendance();
32         attendance = new Attendance();
33         getAllAttendances = new GetAllAttendances();
34         attendances=new ArrayList<>();
35     }
36
37     @After
38     public void tearDown() {
39         so = null;
40         attendance = null;
41         getAllAttendances = null;
42         attendances=null;
43     }
}
```

Слика 182а. Тестирање системске операције *AddNewAttendance*

```
45 @Test
46     public void testExecuteOperation() throws Exception {
47         OptionalProgram optionalProgram = new OptionalProgram();
48         optionalProgram.setId(id: 61);
49         Child child = new Child();
50         child.setId(id: 41);
51         attendance.setChild(child);
52         attendance.setOptionalProgram(optionalProgram);
53         attendance.setStartDate(new Date());
54         attendances.add(e: attendance);
55         getAllAttendances = new GetAllAttendances();
56         getAllAttendances.executeOperation(new Attendance());
57         List<Attendance> attendancesBeforeAdd = getAllAttendances.getAttendances();
58         so.executeOperation(object: attendances);
59         so.broker.getConnection().commit();
60         getAllAttendances = new GetAllAttendances();
61         getAllAttendances.executeOperation(new Attendance());
62         List<Attendance> attendancesAfterAdd = getAllAttendances.getAttendances();
63         assertEquals(attendancesBeforeAdd.size() + 1, actual: attendancesAfterAdd.size());
64     }
}
```

Слика 182б. Тестирање системске операције *AddNewAttendance*

```

66 @Test
67 public void testValidationValidObject() {
68     try {
69         OptionalProgram optionalProgram = new OptionalProgram();
70         Child child = new Child();
71         attendance.setChild(child);
72         attendance.setOptionalProgram(optionalProgram);
73         attendance.setStartDate(new Date());
74         List<Attendance> attendances = new ArrayList<>();
75         attendances.add(e: attendance);
76         so.validation(object: attendances);
77     } catch (Exception ex) {
78         fail(message:ex.getMessage());
79     }
80 }
81
82 @Test(expected = Exception.class)
83 public void testValidationInvalidObject() throws Exception {
84     so.validation(new Object());
85 }
86
87 @Test(expected = Exception.class)
88 public void testValidationEmptyChild() throws Exception {
89     OptionalProgram optionalProgram = new OptionalProgram();
90     Child child = null;
91     attendance.setChild(child);
92     attendance.setOptionalProgram(optionalProgram);
93     attendance.setStartDate(new Date());
94     List<Attendance> attendances = new ArrayList<>();
95     attendances.add(e: attendance);
96     so.validation(object: attendances);
97 }

```

Слика 182в. Тестирање системске операције AddNewAttendance

```

99 @Test(expected = Exception.class)
100 public void testValidationEmptyStartDate() throws Exception {
101     OptionalProgram optionalProgram = new OptionalProgram();
102     Child child = new Child();
103     attendance.setChild(child);
104     attendance.setOptionalProgram(optionalProgram);
105     attendance.setStartDate(startDate: null);
106     List<Attendance> attendances = new ArrayList<>();
107     attendances.add(e: attendance);
108     so.validation(object: attendances);
109 }

```

Слика 182г. Тестирање системске операције AddNewAttendance

The screenshot shows an IDE interface with two main sections. On the left is a code editor displaying Java code for a test method:

```
111  @Test(expected = Exception.class)
112  public void testValidationEmptyOptionalProgram() throws Exception {
113      OptionalProgram optionalProgram = null;
114      Child child = new Child();
115      attendance.setChild(child);
116      attendance.setOptionalProgram(optionalProgram);
117      attendance.setStartDate(new Date());
118      List<Attendance> attendances = new ArrayList<>();
119      attendances.add(e: attendance);
120      so.validation(object: attendances);
121  }
122 }
```

On the right is a test results window titled "Test Results". It shows the status of six tests under the category "so.AddNewAttendanceTest". All tests are marked as "passed" with green checkmarks and execution times.

| Test | Status | Time |
|------------------------------------|--------|-----------|
| so.AddNewAttendanceTest | passed | (0,561 s) |
| testExecuteOperation | passed | (0,561 s) |
| testValidationEmptyChild | passed | (0,001 s) |
| testValidationValidObject | passed | (0,001 s) |
| testValidationInvalidObject | passed | (0,001 s) |
| testValidationEmptyStartDate | passed | (0,0 s) |
| testValidationEmptyOptionalProgram | passed | (0,001 s) |

All 6 tests passed. (0,724 s)

so.AddNewAttendanceTest

testExecuteOperation passed (0,561 s)
testValidationEmptyChild passed (0,001 s)
testValidationValidObject passed (0,001 s)
testValidationInvalidObject passed (0,001 s)
testValidationEmptyStartDate passed (0,0 s)
testValidationEmptyOptionalProgram passed (0,001 s)

Tests passed: 100,00 %

Слика 182δ. Тестирање системске операције AddNewAttendance

5. Закључак

Закључак овог рада наглашава успешност примене упрошћене Ларманове методе развоја софтвера у креирању софтверског система за евидентију похађања изборних програма у предшколским установама. Током процеса, систематски су праћене фазе прикупљања корисничких захтева, анализе, пројектовања, имплементације и тестирања, што је резултирало израдом функционалног и интуитивног решења које се може успешно користити у пракси. Овако развијен софтверски систем може се успешно користити у пракси, али истовремено отвара широке могућности за даљи развој и унапређење. Постоји потенцијал за имплементацију додатних функционалности, као и за отклањање постојећих недостатака. Овај систем омогућава поједностављено праћење активности, али уз примену савремених технологија, постоји могућност значајног побољшања перформанси апликације и унапређења корисничког искуства, што би допринело већој ефикасности и задовољству корисника.

Посебна пажња посвећена је избору програмског језика, где се Јава истакла као изузетно погодна због своје објектно-оријентисане структуре и платформске независности. Јава је већ годинама један од најпопуларнијих програмских језика са великим бројем корисника широм света. Њене могућности, попут високог нивоа сигурности и подршке за развој сложених софтверских система, чине је идеалним избором за пројекте који захтевају поузданост и флексибилност. Поред тога, јака заједница програмера омогућава лакше решавање проблема и бржи напредак у раду.

Током рада на пројекту за предмет Пројектовање софтвера, имала сам прилику да интегришем знања из различитих области које сам стекла током студија. Реализација овог пројекта омогућила ми је да дубље разумем комплексне аспекте развоја софтвера, од иницијалног прикупљања захтева и анализе, до пројектовања и имплементације у програмском језику Јава. Израда апликације није била само технички изазов, већ и прилика да развијем своје вештине у решавању проблема и критичком размишљању. Свака фаза пројекта захтевала је одређену дозу иновативности и пажње према детаљима, како би апликација испунила постављене захтеве. Истовремено, ово искуство ми је пружило прилику да видим како теоријска знања која сам усвојила могу бити примењена у пракси. Развијање апликације за реалну примену ми је у великој мери обогатило моје техничко знање и пружило ми самопоуздање за будуће професионалне изазове.

6. Литература

1. Бојан Томић, Јелена Јовановић, Никола Миликић, Зоран Шеварац, Драган Ђурић, Принципи програмирања, Београд 2018.
2. Проф.др. Синиша Влајић, *Пројектовање софтвера(скрипта)*, Београд 2020.
3. Проф.др. Синиша Влајић, др. Душан Савић, др. Илија Антовић, мр. Војислав Станојевић, мр. Милош Милић, *Пројектовање софтвера-Напредне Јава технологије*, Београд 2016.
4. Baeldung, <https://www.baeldung.com/>
5. GeeksforGeeks, <https://www.geeksforgeeks.org/>
6. DigitalMediaGlobe, <https://digitalmediaglobe.com/>
7. Nanyang Technological University, Singapore, <https://www3.ntu.edu.sg/>
8. Oracle документација о Јава програмском језику, <https://docs.oracle.com/>
9. Scientech Easy, <https://www.scientecheeasy.com/>