

УНИВЕРЗИТЕТ У БЕОГРАДУ
ФАКУЛТЕТ ОРГАНИЗАЦИОНИХ НАУКА

ЗАВРШНИ РАД

**Тема: Развој софтверског система за вођење
евиденције о сеансама психолошког
саветовалишта у .NET окружењу**

Ментор

Др Саша Лазаревић

Студент

Марија Милутиновић 2015/0183

Београд, 2020. године

Садржај

1.... Увод	1
2.... Архитектура софтвера	2
2.1.Трослојна архитектура	3
3.... Технологије за развој софтвера.....	5
3.1. .NET платформа.....	5
3.2. Компоненте .NET Framework-а.....	7
3.3. C# програмски језик.....	9
3.3.1.Извршавање C# програма на .NET Framework платформи	11
3.3.2.Класе и методе.....	12
3.3.3.Наслеђивање.....	14
3.3.4.Апстрактне класе.....	15
3.3.5.Интерфејси	16
3.3.6.Изузеци	16
3.3.7.Нити	18
3.4.Систем за управљање базом података – DBMS Microsoft SQL Server.....	19
4.... Алати за развој софтвера.....	20
4.1. Microsoft Visual Studio 2017.....	20
4.2. NuGet пакет менаџер	23
4.3. MSTest framework	24
4.4. Microsoft SQL Management Server Studio (SSMS)	24
5.... Рад у мрежи	25
5.1. Адреса рачунара	25
5.2. URL адреса ...	26
5.3. Сокети	26
6.... Студијски пример развоја софтверског система	28
7.... Кориснички захтеви	29
7.1 Вербални опис модела	29
7.2. Спецификација захтева помоћу модела случајева коришћења	29
Случајеви коришћења.....	31
СК1: Пријава психотерапеута на систем	31
СК2: Унос новог клијента	31
СК3: Претраживање клијената	33
СК4: Измена података о клијенту	33
СК5: Брисање клијента	34
СК6: Унос нове сеансе	35
СК7: Претраживање сеанси.....	35
СК8: Измена садржаја сеансе	36

СК9: Брисање сеансе.....	37
8.Фаза анализе	38
8.1. Понашање софтверског система: Системски дијаграм секвенци.....	38
ДС1: дијаграм секвенци случаја коришћења - Пријава психотерапеута на систем	38
ДС2: дијаграм секвенци случаја коришћења - Унос новог клијента	39
ДС3: дијаграм секвенци случаја коришћења - Претраживање клијената	41
ДС4: дијаграм секвенци случаја коришћења - Измена података о клијенту	43
ДС5: дијаграм секвенци случаја коришћења - Брисање клијента	47
ДС6: дијаграм секвенци случаја коришћења - Унос нове сеансе	51
ДС7: дијаграм секвенци случаја коришћења - Претраживање сеанси.....	54
ДС8: дијаграм секвенци случаја коришћења - Измена садржаја сеансе	56
ДС9: дијаграм секвенци случаја коришћења - Брисање сеансе.....	61
Списак системских операција.....	65
8.2. Понашање софтверског система: Дефинисање уговора о системским операцијама	66
УГ1: PrijavaPsihoterapeuta(Psihoterapeut)	66
УГ2: UnesiKlijenta(Klijent)	66
УГ3: UčitajListuKlijenata()	66
УГ4: UčitajGradove().....	66
УГ5: NadjiKlijente(kriterijumPretrage, List<Klijent>)	66
УГ6: VratiKlijenta(Klijent).....	66
УГ7: IzmeniKlijenta(Klijent)	66
УГ8: ObrišiKlijenta(Klijent).....	67
УГ9: UčitajOblikePsihoterapije().....	67
УГ10: UnesiSeansu(Seansa)	67
УГ11: UčitajListuSeansi().....	68
УГ12: NadjiSeanse(kriterijumPretrage, List<Seansa>).....	68
УГ13: VratiSeansu(Seansa).....	68
УГ14: IzmeniSeansu(Seansa).....	68
УГ15: ObrišiSeansu(Seansa).....	68
8.3. Структура софтверског система: концептуални (доменски) модел	69
9.... Фаза пројектовања	71
9.1. Архитектура софтверског система.....	71
9.2. Пројектовање складишта података - Релациони модел	72
9.3.Пројектовање апликационе логике	78
Комуникација са клијентима.....	78
Контролер апликационе логике	78
Пословна логика	80

УГ1: PrijavaPsihoterapeuta (Psihoterapeut).....	80
УГ3: UnesiKlijenta (Klijent).....	80
УГ4: UčitajListuKlijenata()	81
УГ5: UčitajGradove().....	81
УГ6: NadjiKlijente(kriterijumPretrage, List<Klijent>)	82
УГ7: VratiKlijenta(Klijent).....	82
УГ8: IzmeniKlijenta(Klijent)	83
УГ9: ObrišiKlijenta(Klijent).....	83
УГ10: UčitajOblikePsihoterapije().....	84
УГ11: UnesiSeansu (Seansa)	84
УГ12: UčitajListuSeansi().....	86
УГ13: NadjiSeanse (kriterijumPretrage, List<Seansa>).....	86
УГ14: VratiSeansu (Seansa).....	87
УГ15: IzmeniSeansu (Seansa).....	87
УГ16: ObrišiSeansu(Seansa).....	88
Брокер базе података	89
9.4. Пројектовање корисничког интерфејса.....	91
Пројектовање екранских форми.....	91
СК2: Унос новог клијента	93
СК3: Претраживање клијената	97
СК4: Измена података о клијенту.....	100
СК5: Брисање клијента	105
СК6: Унос нове сеансе	110
СК7: Претраживање сеанси.....	114
СК8: Измена садржаја сеансе	118
СК9: Брисање сеансе.....	124
Комплетна архитектура софтверског система	128
10.Фаза имплементација	129
10.1. Структура софтверског решења.....	130
10.2. Имплементација складишта података	131
Имплементација апликационе логике.....	136
Комуникација са клијентима	136
Контролер апликационе логике	139
Пословна логика	142
УГ1: PrijavaPsihoterapeuta (Psihoterapeut).....	161
УГ2: UnesiKlijenta (Klijent).....	161
УГ3: UčitajListuKlijenata()	162

УГ4: УчитајGradove()	163
УГ5: НадјиKlijente(kriterijumPretrage, List<Klijent>)	163
УГ6: VратиKlijenta(Klijent)	163
УГ7: IzмениKlijenta(Klijent)	164
УГ8: ОбришиKlijenta(Klijent)	164
УГ9: УчитајOblikePsihoterapije()	165
УГ10: UnesiSeansu (Seansa)	165
УГ11: УчитајListuSeansi()	168
УГ12: НадјиSeanse (kriterijumPretrage, List<Seansa>)	168
УГ13: VратиSeansu (Seansa)	169
УГ14: IzмениSeansu (Seansa)	170
УГ15: ОбришиSeansu(Seansa)	173
Брокер базе података	176
10.3. Имплементација корисничког интерфејса	180
СК1: Пријава психотерапеута на систем	180
СК2: Унос новог клијента	183
СК3: Претраживање клијената	185
СК4: Измена података о клијенту	190
СК5: Брисање клијента	196
СК6: Унос нове сеансе	200
СК7: Претраживање сеанси	204
СК8: Измена садржаја сеансе	208
СК9: Брисање сеансе	214
11.Фаза тестирања	220
12.Закључак	227
13. Литература	228

Списак слика:

Слика 1 – Трослојна архитектура софтвера	4
Слика 2 - Проблем вишестуких библиотека	6
Слика 3 - .Нет Стандард библиотека	6
Слика 4 - Компајлирање у .Нет-у.....	7
Слика 5 - Компоненте .NET Framework-а.....	9
Слика 6 - Начин извршења рада C# програма	12
Слика 7 - Чланови класе у C# програмском језику	14
Слика 8 - Индиректно наслеђивање и концепт специјализације	15
Слика 9 - Изузеци у C# програмском језику	17
Слика 10 - Животни циклус и стања нити у C# програмском језику	19
Слика 11 - Squiggles и Quick Actions својство у C#-у.....	22
Слика 12 - Refactoring својство у C#-у.....	22
Слика 13 - IntelliSense својство у C#-у	22
Слика 14 - Go To Definition својство у C#-у	22
Слика 15 - Peek Definition својство у C#-у	23
Слика 16 - Приватно и јавно NuGet хостовање	24
Слика 17 - Веза два програма преко сокета	27
Слика 18 - Дијаграм Случајева коришћења	30
Слика 19 - ДС Пријава психотерапеута	38
Слика 20 - ДС Неуспешно пријављивање.....	39
Слика 21 - ДС Унос клијента.....	40
Слика 22 - ДС Клијент није сачуван	40
Слика 23 - ДС Претраживање клијената.....	41
Слика 24 - ДС Клијенти који одговарају задатој вредности нису пронађени	42
Слика 25 - ДС Клијент није пронађен.....	43
Слика 26 - ДС Измена података о клијенту.....	44
Слика 27 - ДС Клијенти који одговарају задатој вредности нису пронађени	45
Слика 28 - ДС Клијент није пронађен.....	46
Слика 29 - ДС Клијент није измењен.....	47
Слика 30 - ДС Брисање клијента.....	48
Слика 31 - ДС Клијенти који одговарају задатој вредности нису пронађени	49
Слика 32 - ДС Клијент није пронађен.....	50
Слика 33 - ДС Клијент није обрисан.....	51
Слика 34 - ДС Унос нове сеансе.....	52
Слика 35 - ДС Сеанса није сачувана.....	53
Слика 36 - ДС Претраживање сеанси	54
Слика 37 - ДС Сеансе које одговарају задатој вредности нису пронађене.....	55
Слика 38 - ДС Сеанса није пронађена	55
Слика 39 - ДС Измена садржаја сеансе	57
Слика 40 - ДС Сеансе које одговарају задатој вредности нису пронађене.....	58
Слика 41 - ДС Сеанса није пронађена	59
Слика 42 - ДС Сеанса није измењена	60
Слика 43 - ДС Брисање сеансе	61
Слика 44 - ДС Сеансе које одговарају задатој вредности нису пронађене.....	62
Слика 45 - ДС Сеанса није пронађена	63
Слика 46 - ДС Сеанса није обрисана	64
Слика 47 - Концептуални дијаграм класа (Логичка структура софтверског система)	69
Слика 48 - Структура и понашање софтверског система	70
Слика 49 - Тронивојска архитектура софтверског система	71

Слика 50 - Софтверски систем.....	72
Слика 51 - Архитектура софтверског система након пројектовања комуникације са клијентом и контролером апликационе логике	79
Слика 52 - Повезаност контролера (KontrolerPL) са општом систем операцијом (OpstaSistemskaOperacija)	79
Слика 53 - Дијаграм секвенци "LoginSO"	80
Слика 54 - Дијаграм секвенци "UnesiKlijentaSO"	81
Слика 55 - Дијаграм секвенци "UcitajListuKlijenataSO"	81
Слика 56 - Дијаграм секвенци "UcitajListuGradovaSO"	82
Слика 57 - Дијаграм секвенци "PretragaKlijenataSO"	82
Слика 58 - Дијаграм секвенци "VratiKlijentaSO"	83
Слика 59 - Дијаграм секвенци "IzmeniKlijentaSO"	83
Слика 60 - Дијаграм секвенци "ObrisiKlijentaSO"	84
Слика 61 - Дијаграм секвенци "UcitajListuOblikaSO"	84
Слика 62 - Дијаграм секвенци "UnesiSeansuSO"	85
Слика 63 - Дијаграм секвенци "UcitajListuSeansiSO"	86
Слика 64 - Дијаграм секвенци "PretragaSeansiSO"	86
Слика 65 - Дијаграм секвенци "VratiSeansuSO"	87
Слика 66 - Дијаграм секвенци "IzmeniSeansuSO"	88
Слика 67 - Дијаграм секвенци "ObrisiSeansuSO"	89
Слика 68 - Класа DBBroker повезана је са интерфејсом IDomenskimObjekat кога имплементирају софтверске класе модела	90
Слика 69 - Пројектовање корисничког интерфејса	91
Слика 70 - Приказ форме за пријаву на систем	92
Слика 71 - Обавештење грешке у пријављивању	92
Слика 72 - Обавештење при неуспешном уносу корисничког имена или шифре.....	92
Слика 73 - Приказ успешне пријаве	93
Слика 74 - Приказ главне форме.....	93
Слика 75 – Приказ форме за рад са клијентима	94
Слика 76 - Унос података о клијенту	95
Слика 77 - Приказ успешног уноса клијента	95
Слика 78 - Приказ неуспешног уноса клијента	96
Слика 79 - Приказ листе клијената.....	97
Слика 80 - Претрага клијената по критеријуму.....	98
Слика 81 - Приказ резултата претраге клијената	98
Слика 82 - Обавештење о успешном проналаску клијента.....	99
Слика 83 - Приказ одабраног клијента	99
Слика 84 - Приказ неуспешне претраге према критеријуму	100
Слика 85 - Обавештење о неуспешном проналаску података клијента.....	100
Слика 86 - Приказ претраге клијената	101
Слика 87 - Претрага клијената према унесеном критеријуму.....	102
Слика 88 - Обавештење о успешној претрази	102
Слика 89 - Обавештење о успешно пронађеном клијенту	103
Слика 90 - Приказ промене података клијента	103
Слика 91 - Обавештење о успешној измени података клијента	104
Слика 92 - Обавештење о неуспешној претрази	104
Слика 93 - Обавештење о неуспешном проналаску података клијента.....	105
Слика 94 - Обавештење о неуспешној измени података клијента	105
Слика 95 - Приказ учитане листе клијената.....	106
Слика 96 - Приказ претраге према критеријуму	106
Слика 97 - Обавештење о успешном проналаску клијента.....	107

Слика 98 - Обавештење о успешном проналаску података клијента.....	107
Слика 99 - Приказ података клијента	108
Слика 100 - Обавештење о успешном брисању клијента	108
Слика 101 - Обавештење о неуспешној претрази према критеријуму	109
Слика 102 - Обавештење о неуспешном проналаску података клијента	109
Слика 103 - Приказ обавештења о неуспешном брисању клијента	110
Слика 104 - Приказ уноса нове сеансе	111
Слика 105 - Приказ попуњене форме о новој сеанси.....	111
Слика 106 - Приказ попуњене форме о новој сеанси.....	112
Слика 107 - Обавештење о успешном уносу РЕБТ сеансе.....	112
Слика 108 - Обавештење о успешном уносу психоаналитичке сеансе	113
Слика 109 - Обавештење о неуспешном уносу РЕБТ сеансе.....	114
Слика 110 - Обавештење о неуспешном уносу психоаналитичке сеансе.....	114
Слика 111 - Приказ форме за претрагу сеанси	115
Слика 112 - Претрага сеанси према заадатом критеријуму	115
Слика 113 - Обавештење о успешној претрази.....	116
Слика 114 - Обавештење о успешном проналаску сеансе	116
Слика 115 - Приказ одабране сеансе.....	117
Слика 116 - Обавештење о неуспешној претрази	117
Слика 117 - Обавештење о неуспешном проналаску одабране сеансе	118
Слика 118 - Приказ форме претраге сеанси.....	119
Слика 119 - Претрага према критеријуму	119
Слика 120 - Обавештење о успешном проналаску	120
Слика 121 - Обавештење о успешном проналаску података о сеанси	120
Слика 122 - Приказ одабране сеансе.....	121
Слика 123 - Измена података одабране сеансе.....	121
Слика 124 - Обавештење о успешној измени података	122
Слика 125 - Обавештење о неуспешном проналаску	122
Слика 126 - Обавештење о неуспешном проналаску података сеансе	123
Слика 127 - Обавештење о неуспешној измени података	123
Слика 128 - Приказ форме за претрагу сеанси	124
Слика 129 - Претрага према изабраном критеријуму	125
Слика 130 - Обавештење о успешном проналаску	125
Слика 131 - Обавештење о успешном проналаску података сеансе	126
Слика 132 - Обавештење о успешном брисању сеансе	126
Слика 133 - Обавештење о неуспешном проналаску	127
Слика 134 - Обавештење о неуспешном проналаску података сеансе	127
Слика 135 - Обавештење о успешном брисању.....	128
Слика 136 - Архитектура софтверског система	129
Слика 137 – База података – Психолошко саветовалиште	132
Слика 138 - Табела Град	132
Слика 139 - Табела Клијент	133
Слика 140 - Табела ОбликПсихотерапије.....	133
Слика 141 -Табела Психоанализа.....	133
Слика 142 - Табела РЕБТ	133
Слика 143 - Табела Сеанса	134
Слика 144 - Табела СтавкаСеансе.....	134
Слика 145 - Табела Психотерапеут.....	134
Слика 146 - Приказ дијаграма повезаности табела релационе базе података	135
Слика 147 - Приказ форме за пријављивање.....	181
Слика 148 - Унос корисничког имена и шифре	181

Слика 149- Обавештење о успешној пријави	182
Слика 150- Обавештење о неуспешној пријави	182
Слика 151 - Приказ форме та унос ногог клијента	183
Слика 152 - Попуњена форма уноса клијента	184
Слика 153 - Обавештење о успешном уносу новог клијента	184
Слика 154 - Обавештење о неуспешном уносу новог клијента	185
Слика 155 - Приказ форме за претрагу клијената	186
Слика 156 - Претрага клијената према изабраном критеријуму.....	186
Слика 157 - Обавештење о успешном проналаску	187
Слика 158 - Обавештење о успешном проналаску података клијента.....	187
Слика 159 - Приказ података изабраног клијента	188
Слика 160 - Обавештење о неуспешном проналаску	188
Слика 161 - Обавештење о неуспешном проналаску података клијента.....	189
Слика 162 - Приказ форме за претрагу клијената	190
Слика 163 - Претрага према одабраном критеријуму	191
Слика 164 - Обавештење о успешном проналаску	191
Слика 165 - Обавештење о успешном проналаску података клијента.....	192
Слика 166 - Приказ података одабраног клијента	192
Слика 167 - Измена података одабраног клијента	193
Слика 168 - Обавештење о успешној измени података	193
Слика 169 - Обавештење о неуспешном проналаску	194
Слика 170 - Обавештење о неуспешном проналаску података клијента.....	194
Слика 171 - Обавештење о неуспешној измени података	195
Слика 172 - Приказ форме за претрагу клијената	196
Слика 173 - Претрага према изабраном критеријуму	196
Слика 174 - Обавештење о успешном проналаску	197
Слика 175 - Обавештење о успешном проналаску података клијента.....	197
Слика 176 - Приказ података изабраног клијента	198
Слика 177 - Обавештење о успешном брисању изабраног клијента	198
Слика 178 - Обавештење о неуспешном проналаску	199
Слика 179 - Обавештење о неуспешном проналаску података клијента.....	199
Слика 180 - Обавештење о неуспешном брисању изабраног клијента.....	200
Слика 181 - Приказ форме уноса нове сеансе	201
Слика 182 - Унос нове РЕБТ сеансе	201
Слика 183 - Унос нове психоаналитичке сеансе	202
Слика 184 - Обавештење о успешном уносу РЕБТ сеансе.....	202
Слика 185 - Обавештење о успешном уносу психоаналитичке сеансе	203
Слика 186 - Обавештење о неуспешном уносу РЕБТ сеансе.....	204
Слика 187 - Обавештење о неуспешном уносу психоаналитичке сеансе.....	204
Слика 188 - Приказ форме за претрагу сеанси	205
Слика 189 - Претрага сеанси према изабраном критеријуму	205
Слика 190 - Обавештење о успешном проналаску	206
Слика 191- Обавештење о успешном проналаску података сеансе	207
Слика 192 - Приказ изабране сеансе	207
Слика 193- Обавештење о неуспешном проналаску	208
Слика 194 - Обавештење о неуспешном проналаску података сеансе	208
Слика 195 - Приказ форме за претрагу сеанси	209
Слика 196 - Претрага према изабраном критеријуму	209
Слика 197 - Обавештење о успешном проналаску	210
Слика 198 - Обавештење о успешном проналаску података сеансе	211
Слика 199 - Приказ изабране сеансе	211

Слика 200 - Измена података изабране сеансе	212
Слика 201 - Обавештење о успешној измени података РЕБТ сеансе.....	212
Слика 202 - Обавештење о неуспешном проналаску	213
Слика 203 - Обавештење о неуспешном проналаску података сеансе	213
Слика 204 - Обавештење о неуспешној измени података	214
Слика 205 - Приказ форме за претрагу сеанси	215
Слика 206 - Претрага сеанси према унесеном критеријуму	215
Слика 207 - Обавештење о успешном проналаску	216
Слика 208 - Обавештење о успешном проналаску података сеансе	216
Слика 209 - Обавештење о успешном брисању сеансе	217
Слика 210 - Обавештење о неуспешном проналаску	218
Слика 211 - Обавештење о неуспешном проналаску података сеансе	218
Слика 212 - Обавештење о неуспешном брисању	219
Слика 213 - Мануелно и аутоматско тестирање.....	220
Слика 214 - Тест пирамида.....	221

Списак табела:

Табела 1 - Психотерапеут.....	73
Табела 3 - Клијент	73
Табела 2 - Град	73
Табела 6 - Табела ОбликПсихотерапије.....	74
Табела 4 - Табела РЕБТ	75
Табела 5 - Табела Психоаналитика	75
Табела 7 - Табела Сеанса	76
Табела 8 - Табела СтавкаСеансе.....	76

1. Увод

Софтверски системи су данас заступљени и користе се у готово свакој индустрији. Њихова сврха у том смислу је да одрже, контролишу и унапреде ефикасност пословања. Софтверски систем одређене компаније представља један од њених најважнијих ресурса, а компаније које схватају значај поседовања модерног и ефикасног софтвера су далеко испред оних које то не схватају. Улога софтвера је да управља свим процесима и задацима једне компаније. Карактеристике софтвера који се креира зависе од величине и комплексности организације за коју се изводи, корисничких захтева итд.

У овом раду биће приказан развој софтверског система који представља десктоп апликацију за вођење евиденције о клијентима и сеансама психолошког саветодавања. Савремене технолошке иновације доста олакшавају и убрзавају обављање свакодневних активности. Са друге стране, млади људи 21. века налазе се под притиском да савладају исте и да иду у корак са временом. Очекивања се мењају, оно што се некада сматрало успехом данас се подразумева. Одрастање у таквом окружењу неретко доводи до неповољних психичких стања попут анксиозности и депресије, о чему сведочи статистика. Одатле идеја за десктоп апликацију у овом раду. Циљ је био да се и на овај начин скрене пажња на значај психотерапије и разбије стигма везана за исту. Како су технологије убрзале начин живота, циљ је био да се исте искористе у борби против стресне свакодневнице коју су изазвале, тиме што ће олакшати непроценљив рад и труд психотерапеута.

Софтверски систем је развијен коришћењем Ларманове методе развоја софтвера. Рад је организован у једанаест поглавља. Подељен је на теоријски и практични део. Теоријски део почиње описом архитектуре софтвера, коришћене технологије и алата. Реч је о .NET платформи. Такође се говори о C# језику као објектно - оријентисаном програмском језику, где су дати његови основни концепти. Након тога објашњава се конкурентно програмирање у C# коришћењем нити. У наставку се разматра програмирање у мрежи. Посебно је наглашен концепт сокета као кључног механизма у мрежном програмирању.

Практични део започиње описом Ларманове методе развоја софтвера, након чега је кроз низ поглавља детаљно објашњен процес развоја софтверског система коришћењем исте кроз фазе прикупљања захтева, анализе, пројектовања, имплементације и тестирања.

Завршни рад је заснован на десктоп апликацији, написаној у програмском језику C#, у окружењу MS Visual Studio 2017 при коришћењу Microsoft SQL Server Management Studio-a као система за управљање релационом базом података. При пројектовању апликације примењена је тронивојска архитектура софтверског система.

2. Архитектура софтвера

Данас још увек не постоји јединствена дефиниција софтверске архитектуре. Постоји много класичних и модерних тумачења и дефиниција од стране ИТ заједнице, а поставља се питање и да ли је добро тежити општој дефиницији софтверске архитектуре која ће бити применљива на сваки систем и сваку ситуацију.

Једну интересантну дефиницију дао је Eoin Woods: “Софтверска архитектура је скуп одлука о дизајну које, ако се не донесу коректно, могу довести до неуспеха пројекта”. [5]

Софтверска архитектура се може дефинисати и као структура неког рачунарског система или програма, која обухвата и описује спољашње видљиве особине софтверских компоненти, и релације тј. интеракције између њих. Архитектуру занима само јавни део компоненти, а не детаљи који се односе искључиво на унутрашњу имплементацију. Термин се такође односи и на документацију софтверске архитектуре система. Документовање софтверске архитектуре олакшава комуникацију између свих учесника током развоја софтверског система.

Првобитни концепти софтверске архитектуре постављени су у радовима Дијкстре и Парнаса 60-их и 70-их година прошлог века. Ови научници у својим истраживањима наглашавају значај структуре софтверског система и говоре о томе колико је важно добро испланирати структуру неког система. Истраживања на овом пољу се настављају средином 1990. године са фокусирањем на архитектуралне стилове, језике за опис архитектуре, архитектуралну документацију итд. [5]

Неки од језика који се користе за опис архитектуре софтвера (Architecture description language (ADLs): AADL (SAE стандард), Wright, Acme (оба развијена од Carnegie Mellon), xADL (развијен од UCL), Darwin (развијен од Imperial College London).

Софтверска архитектура је уобичајено организована преко погледа, који представљају различите врсте нацрта и шема које се праве у архитектури. Различита гледишта постоје да опишу архитектуру из угла заинтересованих страна и њихових различитих интереса.

Неки од могућих погледа према стандарду ANSI/IEEE 1471-2000 су: Функционални / Логички поглед, Код/модул поглед, Развојни/структурални поглед, Конкурентни/Thread поглед, Физички/имплементациони поглед, Кориснички/повратни поглед, Поглед података.

Као што је случај и са дефиницијом софтверске архитектуре, тако је осмишљено и неколико језика за опис софтверске архитектуре, али није постигнут договор који скуп симбола и систем погледа треба да се усвоји као стандардни. UML је успостављен као стандард за моделовање система, посебно софтверских система, што се односи и на погледе софтверске архитектуре. [5]

Са друге стране, треба имати у виду да ефикасан развој софтвера зависи од система за који се прави и његових ограничења. Дакле, универзална нотација је осуђена на неуспех, јер сваки систем за који се развија софтвер поседује одређене специфичности које воде до њене неупотребљивости у одређеним ситуацијама.

2.1.Трослојна архитектура

Да би се пројектовала архитектура одређеног софтверског система, треба пре свега одабрати неки од постојећих типова вишеслојне архитектуре у односу на који ће се дизајнирати софтверски систем. У пракси, архитектура софтверског система најчешће представља комбинацију неколико следећих стилова :

Клијент/Сервер - представља раздвајање система на клијентску страну, која се налази код корисника и на базу података са апликативном логиком

Архитектура заснована на компонентама - систем чине функционалне или логичке компоненте које обезбеђују интерфејс за спољну комуникацију

Доменски вођен дизајн - Објектно оријентисан приступ који подразумева моделовање система креирањем класа у односу на идентификоване објекте пословног домена који се моделира

Сервисно оријентисана архитектура (COA) - односи се на апликације које обезбеђују и користе функционалности као сервисе користећи уговоре и поруке

Н-слојева / 3 слоја - трослојна архитектура је једна од најчешће коришћених у пракси на основу досадашњег искуства. У трослојном генеричком моделу јасно се одваја управљање подацима, апликациона логика и кориснички интерфејс. Прилагодљива је брзим променама, како у корисничком (пословном), тако и у имплементационом (технолошком) окружењу. Трослојна архитектура је генеричка за вишеслојне архитектуре које постају општеприхваћени стандард.

Трослојну архитектуру чине: презентациони, слој пословне логике и слој података.

Презентациони – овај слој обезбеђује приказ података крајњем кориснику користећи неку од расположивих технологија за кориснички интерфејс. У случају да правите десктоп апликације то може бити Windows Forms или WPF док рецимо ако правите веб апликације то је ASP .NET и веб технологије попут HTML5, JavaScript, CSS итд.

Слој пословне логике – овај слој имплементира пословну логику апликације. Пословну логику чине пословни процеси и пословне компоненте. Такође ,овде се најчешће имплементирају и пословна правила добијена у процесу анализе.

Слој података – већина пословних апликација користи релационе базе података за складиштење података. Неке од база : MySQL, Oracle, PostgreSQL, Microsoft SQL Server, MongoDB итд.

Неке од предности овакве архитектуре су:

Скалабилност - способност програма да успешно обради пораст корисника, нпр. у веб апликацијама, скалабилност би била способност апликације да успешно функционише код повећаног броја посета.Треба имати у виду да на скалабилност утичу и хардверске компоненте које покрећу систем.

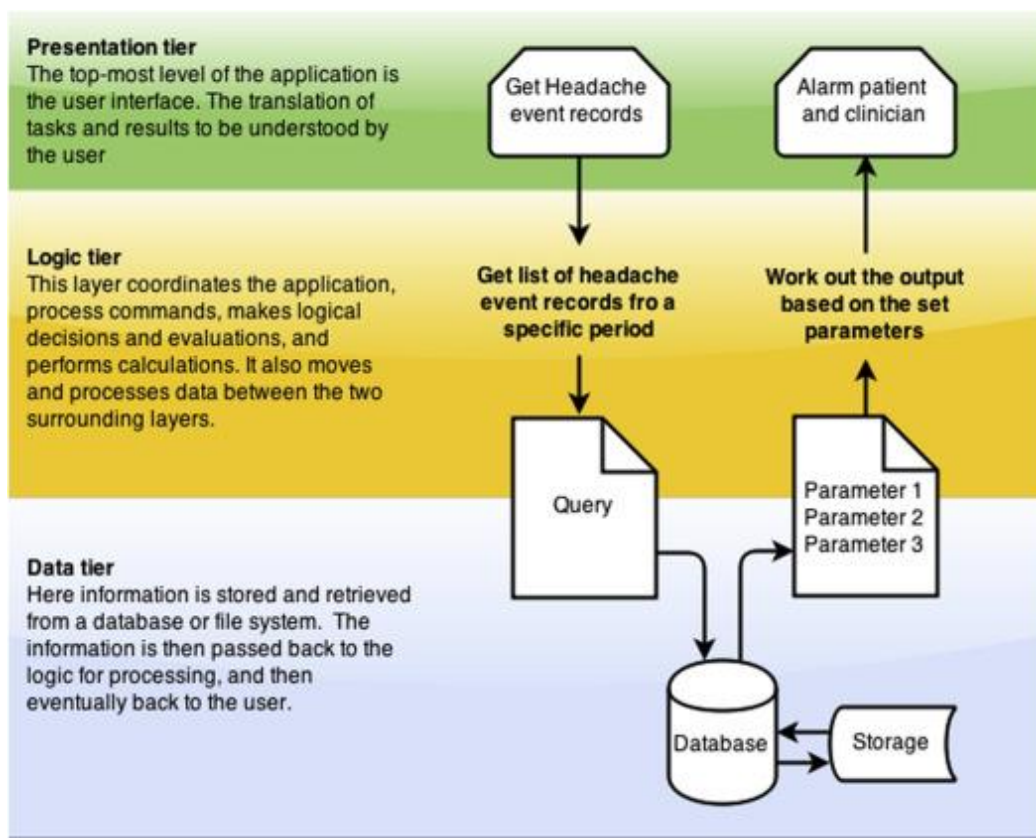
Флексибилност - способност програма да се лако прилагоди новим захтевима корисника.Трослојна архитектура обезбеђује флексибилност кроз могућност рада на неком делу апликације независно од осталих делова. На пример, изглед и дизајн

корисничких форми може бити унапређен без мењања логике пословног слоја и слоја података.

Доступност - вероватноћа да систем исправно функционише било кад, осим када је у питању одржавање - ова предност се огледа у томе што је архитектура система модуларна.

Суштину ове архитектуре одражава средњи слој који се различито назива: *апликациони сервер, трансакциони сервер, сервер компоненти, сервер пословних правила*, чиме се посебно истиче нека функционалност овога слоја.

Како је већ напоменуто, трослојна архитектура је генеричка за вишеслојне архитектуре преко средњег слоја у коме се различите функције раслојавају, да би се преко већег броја слојева, односно већег степена индирекције, омогућила већа модуларност, хетерогеност и еластичност система.



Слика 1 – Трослојна архитектура софтвера

3. Технологије за развој софтвера

У овом раду коришћене су .NET технологије. У наставку су исте детаљније објашњене.

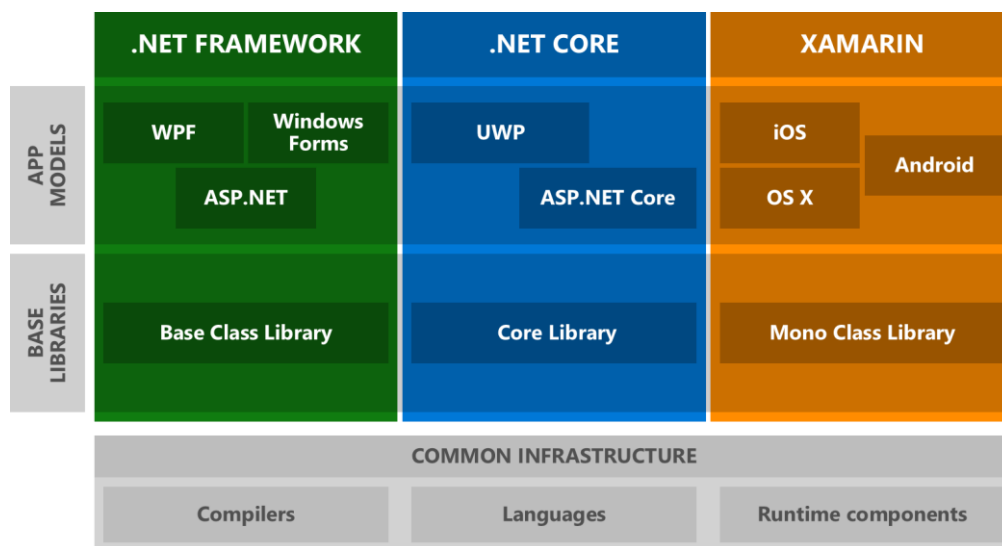
3.1. .NET платформа

.NET платформа је окружење отвореног кода за развој софтвера, развијено од стране Microsoft-а првенствено за Windows платформу, а касније и за iOS и Android OS када постаје вишеплатформско окружење. Користи се за изградњу много различитих типова апликација. Помоћу .NET -а можете направити апликације за веб, мобилне уређаје, десктоп, игре и IoT.

Укључује велику библиотеку класа (Framework class library). Microsoft је са развојем .NET-а почео раних 1990-тих, под називом Next Generation Windows Services. Почетком 2000-тих прва бета верзија .NET 1.0 је објављена, а у августу 2000. у сарадњи са Intel-ом и HP-ом, Microsoft је почео са стандардизацијом Common Language Инфраструктуре (CLI), која ће омогућити извршавање различитих програмских језика на различитим архитектурама-платформама.

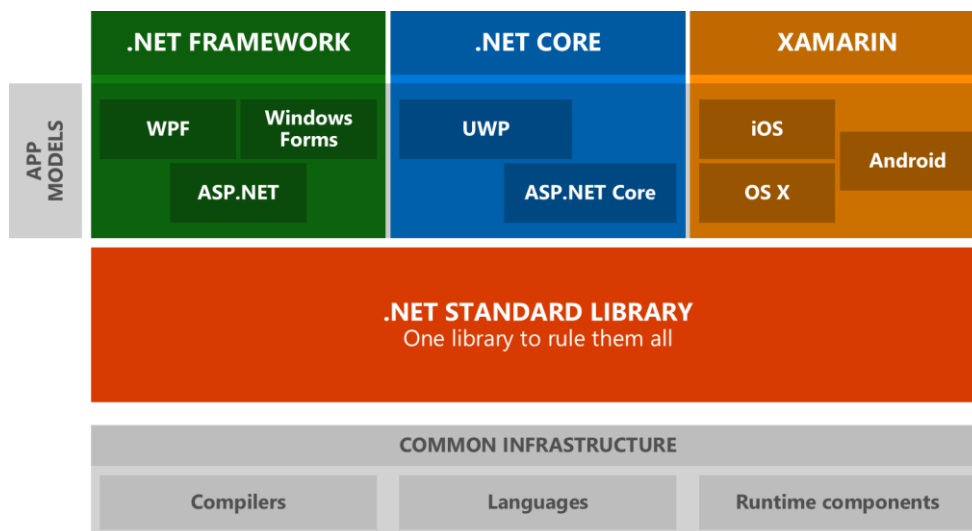
Програми се извршавају кроз софтверско окружење CLR, (Common Language Runtime), виртуалну машину која садржи: memory management, exception handling, garbage collector итд. .NET Framework подржава више од 60 програмских језика од којих је 11 дизајнирао и развио Microsoft, од којих су најпопуларнији C#, F#, C++ и VisualBasic. C# је једноставан, модеран и објектно оријентисан програмски језик. F# је функционални програмски језик који укључује објектно оријентисано и императивно програмирање. Visual Basic је приступачан језик са једноставном синтаксом за изградњу објектно оријентисаних апликација. Главни развојни алат је Visual Studio.

.NET standard је Microsoft решење за проблем са вишеструким библиотекама. Овима је спречена даља фрагментација .NET платформе. У овом тренутку постоје три главне платформе за развој у оквиру .NET еко система: .NET Framework, .NET Core и Xamarin. Као што се види на слици, свака од њих је имала посебан Base Class Library. Да би се развила нека функционалност која може да се користи на све три платформе, што данас већина модерних апликација захтева, било је неопходно да се познају карактеристике сваке.



Слика 2 - Проблем вишестуких библиотека

Microsoft је креирао .NET standard да би олакшао дељење и поновно коришћење кода између различитих .NET платформи. За дивелопере то значи да је довољно да савладају само један тип библиотека уместо више досадашњих. Све библиотеке које су креиране као .NET standard библиотеке ће моћи да буду коришћене на било којој .NET платформи. [Vuleta,2018]



Слика 3 - .Нет Стандард библиотека

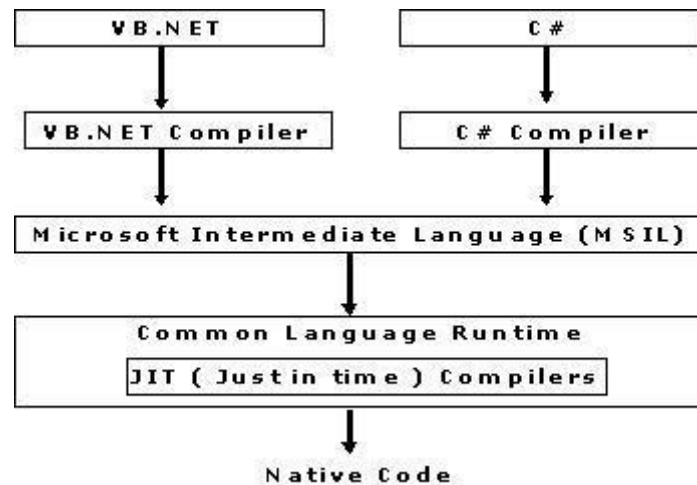
3.2. Компоненте .NET Framework-a

Као што је већ речено, .Net Framework је платформа која обезбеђује потребне алате и технологије програмерима да би они развили, користили и дистрибуирали Windows и Web апликације.

Уопштено говорећи, .Net Framework се састоји од две главне компоненте:

1. *Common Language Runtime (CLR)*
2. *.Net Framework Class Library(FCL)*

1. .Net Framework садржи извршно окружење које се зове Common Language Runtime (CLR). То окружење пружа могућност извршавања свих .NET програма без обзира на ком су програмском језику писани. Код који обрађује и извршава CLR се зове *Managed Code*. CLR пружа услуге као што су ремодинг, управљање нитима, безбедност типа, управљање меморијом, робусност итд. На пример, програмери не треба да брину о алоцирању и де-алоцирању меморије приликом покретања програма, јер то за њих ради CLR.



Слика 4 - Компајлирање у .Net-у

Језици се (нпр. C#, F# ,VB), сваки преко свог компајлера, компајлирају у Common Intermediate Language (CIL) међу-језик. Затим, у зависности од тога на којој се платформи извршава, CLR компајлира CIL у машински код. То је једна од главних улога CLR –а. Тренутно је направљено преко 15 језичких компајлера од стране Microsoft-а и других компанија.

2. Збирка виšekратних, објектно оријентисаних библиотека класа које садрже велики број предефинисаних својстава и метода које се могу интегрисати са CLR-ом. Може се направити аналогија између FCL библиотека у .NET-у и пакета у JAVA окружењу.

FCL може бити дефинисан и као сет API(Application Programing Interface)-ја написаних у C#-у уз помоћ којих се пишу .NET програми. На пример, Console.WriteLine је један API у оквиру FCL-а. FCL зову и Base Class Library јер је иста за све типове апликација које се праве. На пример, начин на који се приступа и користи FCL у VB.NET-у ће бити исти као и у C#-у, и исти за све остале .NET језике.

Неке од апликација које користе FCL: Windows Application, Console Application, Web Application, XML Web Services, Windows Services.

Укратко, да би се користила FCL у било ком .NET језику за било који тип апликације, потребно је да програмер учита FCL у свој код/програм и тиме ће моћи да користи предефинисане сложене методе за нпр. читање и уписивање у фајл, графичко рендеровање, интеракцију са базом и манипулацију XML документима.

Важно је описати још неколико концепата који су део .NET framework-а поред компоненти описаних изнад.

3. *Common Type System (CTS)*

Дефинише који су то дозвољени типови података које могу да користе различити .NET језици. Циљ успостављања заједничких типова података за различите језике је да би се омогућило да објекти писани у различитим .NET језицима могу бити у интеракцији једни са другима. Да би програми писани у различитим .NET језицима комуницирали међусобно, типови података треба да буду компатибилни на основном нивоу.

CTS подржава две основне категорије типова:

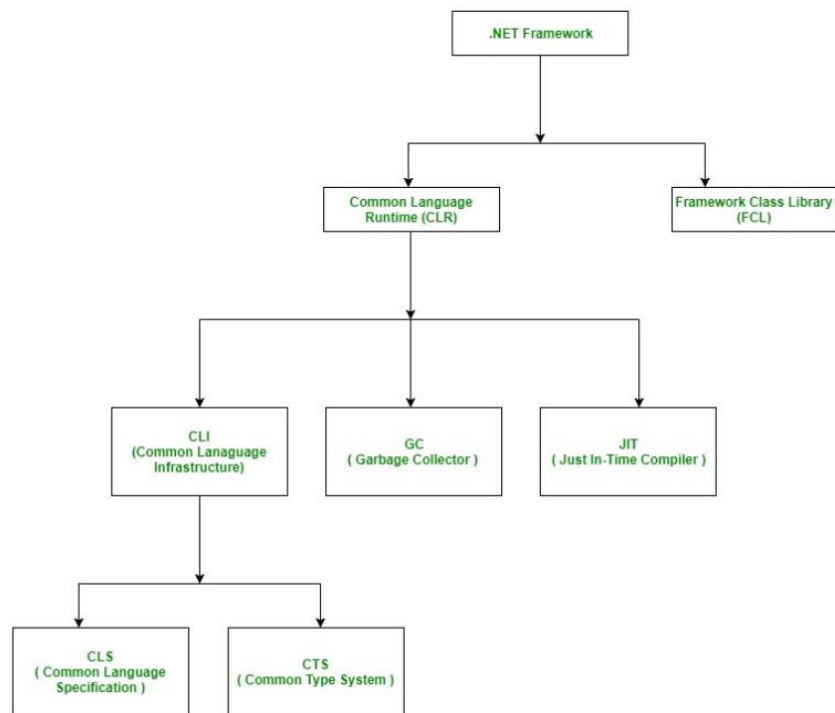
Вредносни типови (Предефинисани целобројни, типови података са покретним зарезом, `boolean` и `char` вредносни типови, кориснички дефинисани: `Struct`, `Enum`). Ови типови података се увек чувају на стеку. Када се заврши са коришћењем истих, кроз се бришу и ослобађају меморију. Када се врши копирање вредносног типа прави се нова променљива и зато се промене код оригинала и обратно не одражавају.

Референтни типови (`string`, `object`, `Class`, `Array`, `Interface`, `Delegate`) - То су типови који чувају само адресу или референцу на вредност. Када се креира референтни тип, одваја се само мали део меморије који чува адресу другог блока меморије у контролисаном хипу. Референтни тип се налази на две меморијске локације. Прави подаци су на хипу док променљива која садржи показивач на тај објект се налази на стеку. Када та променљива обави своју функцију, референца објекта се брише али не и сам објект. [Радовановић, 2015]

4. *Common Language Specification (CLS)*

Дефинише сет правила која језички компајлери морају задовољити да би се компајлирање наставило у CLR-у. Правила су написана тако да дефинишу минимални и комплетни сет особина које код мора да задовољи да би дошао до CLR окружења. Улога CLS-а је да обезбеди правилан рад компајлера у .NET окружењу. Пример правила: Репрезентација стрингова, интерна репрезентација енумерација итд.

Microsoft је креирао Common Language Infrastructure (CLI), чији су делови претходно описани концепти CLS и CTS, у циљу да програми писани у различитим .NET језицима могу комуницирати међусобно. [4]



Слика 5 - Компоненте .NET Framework-a

3.3. C# програмски језик

C# је објектно оријентисани програмски језик. У објектно оријентисаном програмирању полази се од објеката којима се жели манипулисати, а не од логике која је потребна за ту манипулацију. Дакле, првенствено се у реалном систему идентификују објекти и везе које постоје између њих, а након тога се одлучује о начинима манипулисања тим објектима. [M. Vučković, N. Turajlić, M. Petrović, 2009/2010]

C# синтакса је веома изражајна, али је и једноставна и лака за учење. Синтакса овог језика поједностављује многе сложености C++ и пружа моћне функције као што су nullable типови вредности, еnumerације, делегати, ламбда изрази и директан приступ меморији, који се не налазе у Јави. Процес креирања C# кода је једноставнији у поређењу са C и C++ и флексибилнији него у Јави. Не постоје одвојене датотеке заглавља и нема захтева да се методе и типови декларишу у одређеном редоследу. Изворна C# датотека може дефинисати било који број класа, структура, интерфејса и догађаја.

Концепти објектно оријентисаног програмирања које C# подржава :

- енкапсулација
- наслеђивање
- полиморфизам.

Пре него што се опишу наведени концепти, треба објаснити шта је то апстракција. Она представља занемаривање небитних детаља и усредсређивање на битне, или још простије – селективно незнање.

Приступ “црне кутије” значи да постоји много објеката које можемо да користимо иако не разумемо њихову унутрашњу структуру. Два објекта могу имати исту функцију иако им је унутрашњост потпуно различита.

Учаурење или енкапсулација је додатна апстракција којом се “сакривају” детаљи имплементације објекта.

Постоје два битна аспекта учаурења: обједињавање података и функција у јединствен ентитет (класа) и контрола могућности приступа члановима ентитета (модификатори приступа). Основни разлог постојања овог концепта је чињеница да је директан приступ подацима непожељан и непотребан.

Модификатори за одређивање права приступа члановима класе :

- Public - јавни приступ члановима без ограничења
- Private - приватни приступ, само чланови класе имају приступ
- Protected – заштићени приступ, само чланови класе и чланови изведене класе имају приступ
- Internal – интерни приступ, само елементи који су у оквиру истог асемблија
- Protected internal – унија заштићеног и интерног приступа
- Private protected – приступ из изведене класе у оквиру истог асемблија.

Енкапсулација омогућава контролу коришћења, тј. објекат се може користити искључиво преко јавних метода. Омогућава се смањивање утицаја промена. Уколико су детаљи имплементације објекта приватни могу се променити, а да те промене не утичу директно на корисничке објекте (који једино могу да приступе јавним методама).

Основна одлика објектно-оријентисаног програмирања је наслеђивање. Приликом наслеђивања, поткласа наслеђује и методе и структуру од своје надкласе. Чињеница да поткласа наслеђује методе значи да њих не треба писати поново. Треба дописати само нове ствари, као и оне које се разликују. Ова особина се зове *code reuse*.

Способност променљиве да референцира објекте различитих типова и да аутоматски позива одговарајућу методу објекта који се референцира се назива полиморфизам. Полиморфизам се заснива на следећем концепту: Метода која је декларисана у базној класи може да се имплементира на више различитих начина у различитим изведеним класама. Полиморфизам се реализује помоћу виртуелних метода. За декларацију виртуелне методе се користи кључна реч *virtual*. Приликом дефинисања, виртуелне методе се морају имплементирати. Процес имплементације виртуелне методе у изведеној класи се назива реимплементација, при чему се користи кључна реч *override*.

Поред ових основних објектно оријентисаних принципа, C# олакшава развој софтверских компоненти кроз неколико иновативних језичких конструкција, укључујући следеће:

- Енкапсулирани потписи метода названи делегати, који омогућавају обавештења о догађајима који су сигурни за тип.
- Својства (*property*), која служе као приступи за приватне члан варијабле.
- Атрибути који обезбеђују декларативне метаподатке о типовима у време извођења.
- Инлине XML коментари документације.

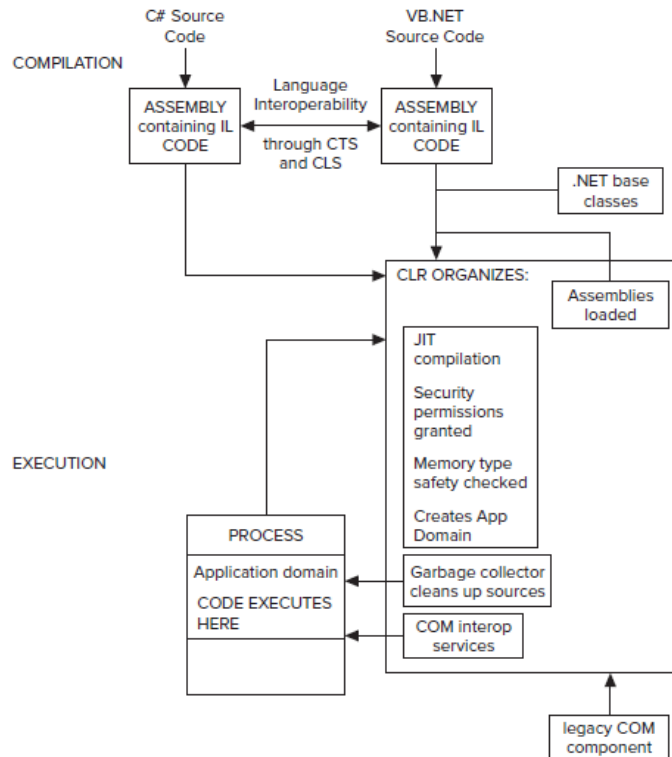
- Language-Integrated Query (LINQ) који пружа уграђене могућности упита у различитим изворима података.

3.3.1. Извршавање C# програма на .NET Framework платформи

Програми написани у C# раде на .NET Framework-у, интегралној компоненти Windows-а која укључује виртуелни извршни систем који се зове CLR и обједињени скуп библиотека класа FCL. CLR је комерцијална имплементација од стране Microsoft-а заједничке језичке инфраструктуре (CLI), међународног стандарда који је основа за стварање окружења за извршавање и развој у којима различити језици и библиотеке раде без проблема.

Изворни код написан у C# компајлиран је у средњи језик (IL) који је у складу са CLI спецификацијом. IL код и ресурси, као што су битмапе и низови, складиште се на диску у извршној датотеци која се зове assembly, обично са екстензијом .exe или .dll. Assembly садржи манифест који пружа информације о типовима assembly-ја, верзији, култури и сигурносним захтевима.

Када се C# програм изврши, assembly се учитава у CLR, што може да предузме различите акције на основу информација у манифесту. Затим, ако су задовољени сигурносни захтеви, CLR изводи "just in time" (JIT) компајлирање за претварање IL кода у изворне машинске инструкције. CLR такође пружа и друге услуге које се односе на аутоматско прикупљање смећа, руковање изузецима и управљање ресурсима. Код који се извршава од стране CLR-а понекад се назива "управљани код". На следећем дијаграму приказани су односи времена компајлирања и времена извршавања датотека изворног кода C#, .NET Framework библиотека, assembly-ја и CLR-а. (Wenzel, Wagner & Latham, 2015)



Слика 6 - Начин извршења рада C# програма

Језичка интероперабилност је кључна карактеристика .NET Framework-а. Због тога што је IL код који је креирао C# компајлер у складу са Common Type Specification (CTS), IL код генерисан у C# може да узајамно делује са кодом који је генерисан из .NET верзија Visual Basic-а, Visual C++-а или било ког од више од 20 других комплементарних језика.

3.3.2. Класе и методе

У модерним програмским језицима класе се користе као облик организовања комплексног програмског кода у логичке целине које се лакше пишу и тестирају. Често је веома корисно да се неколико података третира као једна целина са становишта програмског језика. Тако се, рецимо, подаци о некој особи као што је име, презиме, датум рођења, адреса, пол и ЈМБГ могу спаковати у један “пакет података”, уместо да се третирају као шест независних променљивих. [Машуловић, 2020]

Класа даје генеричку дефиницију објекта, описује заједничку структуру и понашање које ће имати једна врста објекта. Објекти који припадају некој класи зову се инстанце те класе, односно објекат представља једно конкретно појављивање своје класе.

Класа је основна јединица програмирања у C#-у. Она се састоји од атрибута и метода. Атрибутима се дефинише стање (структура), а методама понашање класа које ће наследити све њене инстанце.

Класа је референтни тип. Декларацијом се не креира инстанца класе (објекат), већ референца. Инстанцирање класе, односно креирање објекта врши се из два корака:

- алоцира се меморија за објект помоћу оператора *new*.
- креирање објекта се врши помоћу конструктора којим се алоцирана меморија “претвара” у објект и објект иницијализује. Приступ члановима објекта остварује се навођењем пуног квалификованог имена члана.

Klijent k; ← deklaracija
 k = new Klijent() ; ← poziv konstruktora
 k.ID = 11 ; ← puno kvalifikovano ime

То значи да се у статичкој меморији чува само референца на објект, док се меморијски простор неопходан да објект запамти своје стање алоцира у динамичкој меморији. Дакле, инстанце класе су динамичке, налазе се у динамичкој меморији и за њих морамо експлицитно алоцирати простор. Тако долазимо до појма конструктора и до разлога његове примене.

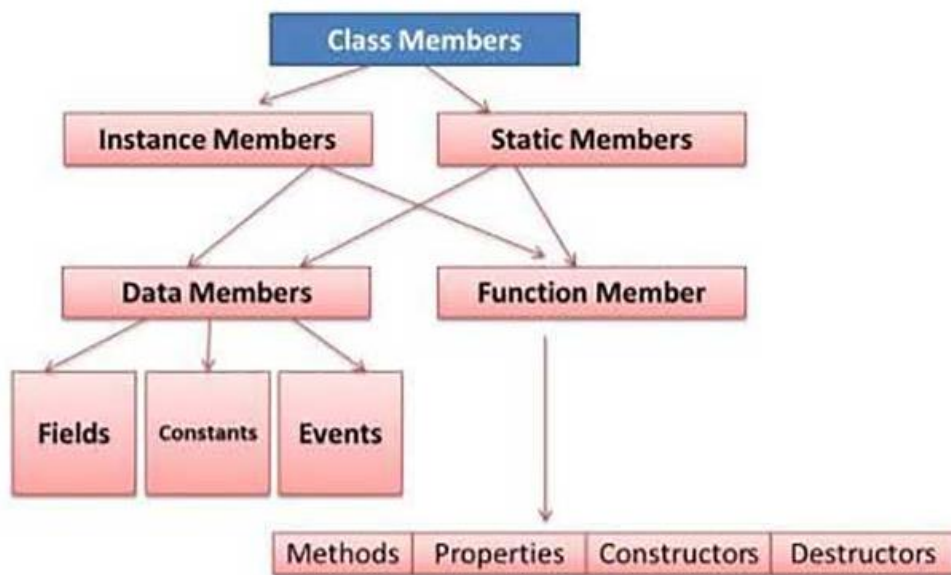
Конструктор је специјална метода која служи за иницијализацију објекта након њиховог креирања. Позива се експлицитно, и у њој се наводи низ инструкција које постављају стање објекта на почетне вредности. Конструктор обезбеђује да објект има добро дефинисано почетно стање пре него што се употреби. Ако не успе иницијализација неће постојати објект. [M. Vučković, N. Turajlić, M. Petrović, 2009/2010]

Што се тиче синтаксе креирања конструктора ,за назив конструктора узима се име класе (метода има исто име као и класа) иза кога следе заграде. Нема повратну вредност, па чак ни типа void. Постоје две врсте конструктора, то су параметарски и непараметарски конструктор.

Методе су процедуре у којима је дефинисано понашање класе. Оне омогућују приступ пољима објекта, мењају његово стање итд. Свака метода садржи:

- 1) Тип који враћа (или воид уколико не враћа никакав тип) У случају да метода враћа неку вредност користи се кључна реч *return*. Преко ње метода враћа вредност.
- 2) Назив
- 3) Листу параметара
- 4) Тело методе

C# подржава преклапање (overloaded) метода у оквиру једне класе. То подразумева да једна класа може имати две или више истоимених метода. Да би све ово било могуће, те преклопљене методе се морају разликовати по потпису. Разлике у потпису методе подразумевају разлике у броју и/или типу параметара које метода добија као улазне параметре. На исти начин, као што се могу преклапати методе, то је могуће и са конструкторима.



Слика 7 - Чланови класе у C# програмском језику

3.3.3. Наслеђивање

Основна одлика објектно-оријентисаног програмирања је наслеђивање. Наслеђивање је релација међу класама која омогућује да се дефиниција и имплементација једне класе базира на дефиницији и имплементацији неке већ постојеће класе. На тај начин се штеди време, новац, али и меморијски простор. [Mašulović,2020]

Приликом наслеђивања, поткласа наслеђује и методе и структуру од своје надкласе. Чињеница да поткласа наслеђује методе значи да њих не треба писати поново. Треба дописати само нове ствари, као и оне које се разликују. Ова особина се зове *code reuse*.

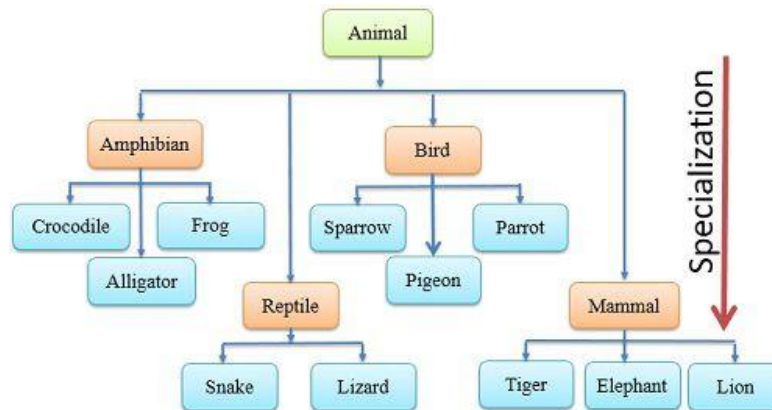
Наслеђивањем се остварује веза, тачније успоставља се однос између објеката. *IS* веза ("је типа" веза) се формира између базне (супер, надређене) класе и изведене (подкласе, подређене) класе која наслеђује базну класу.

Треба разликовати наслеђивање класа и наслеђивање интерфејса. Класа може да имплементира више интерфејса, међутим класа не може да наследи више класа, већ може наследити само једну класу. Класа која наслеђује интерфејс мора да имплементира све функције тог интерфејса. Такође, класа може у исто време да наследи класу и да имплементира један или више интерфејса. Још нека правила наслеђивања у C#-у : Интерфејс може да наследи више интерфејса, и интерфејс не може да наследи класу.

Наслеђивање може бити једноструко и вишеструко. C# подржава само једноструко наслеђивање, односно не подржава могућност вишеструког наслеђивања. Дакле, подкласа може наследити само једну надкласу. Приликом дефинисања класа могуће је користити кључну реч *final*. То значи да се такве класе не могу наслеђивати.

Такође, наслеђивање може бити директно и индиректно. Свака изведена класа може

даље бити базна некој следећој класи, чиме се постиже индиректно наслеђивање. Група класа које су повезане наслеђивањем формирају структуру која се назива хијерархија класа. Класе на вишим нивоима су општије (концепт генерализације), док су оне на нижим нивоима у хијерархији специфичније (концепт специјализације). Дубина хијерархије представља број нивоа наслеђивања. Препорука је да број нивоа не буде већи од седам. [M. Vučković, N. Turajlić, M. Petrović, 2009/2010]



Слика 8 - Индиректно наслеђивање и концепт специјализације

3.3.4. Апстрактне класе

Апстрактна класа је класа која има бар једну апстрактну методу. Класа која нема апстрактне методе се зове конкретна класа. Апстрактна класа мора бити декларисана као апстрактна одмах на почетку кључном речи *abstract*.

Веома је важно истаћи да апстрактне класе не могу да имају инстанце, јер инстанца не зна како ради ниједна од апстрактних метода класе. Тек конкретне класе које наследе апстрактну класу и имплементирају неку од њених апстрактних метода могу да се инстанцирају.

Што се апстрактних метода тиче, оне се разликују од обичних метода. Немају тело методе. Реализација апстрактних метода се преноси на методе класе које наслеђују апстрактну класу. Апстрактна метода постоји као концепт јер се често у пракси јавља потреба за потпуно различитим имплементацијама апстрактних метода у подкласама апстрактне класе.

Подкласа апстрактне класе не мора понудити имплементацију за апстрактне методе. Штавише, подкласа апстрактне класе може додати и нове апстрактне методе. Битно је само то да за сваку апстрактну класу постоји њена подкласа која је конкретна. Тиме се обезбеђује да свака апстрактна метода буде некад и негде имплементирана.

Такође, могуће је да подкласа неке конкретне класе буде апстрактна. За њу тада важи све што важи и за остале апстрактне класе. [Mašulović, 2020]

3.3.5. Интерфејси

Док основна класа дефинише шта она у ствари представља, интерфејс дефинише шта класа ради. Интерфејс не зависи од класе и нема уопште знања о томе шта класа представља, али га класа свакако може имплементирати и користити у многим ситуацијама. Интерфејс је концепт који раздваја спецификацију методе од њене имплементације. Све методе у интерфејсу су апстрактне методе чија се имплементација врши у класама које тај интерфејс имплементирају. Интерфејс не може да садржи поља.

Једна класа не може да наследи више основних класа, али зато може истовремено да имплементира више интерфејса. Такође, интерфејс може да позива друге интерфејсе, што се назива *re-implementation*.

“Интерфејси јесу слични апстрактним класама, само што не могу да садрже ништа осим наведених потписа и није реткост да се интерфејси користе заједно са апстрактним класама да би се градила програмерска радна окружења која се проширују.”[Радовановић,2015]

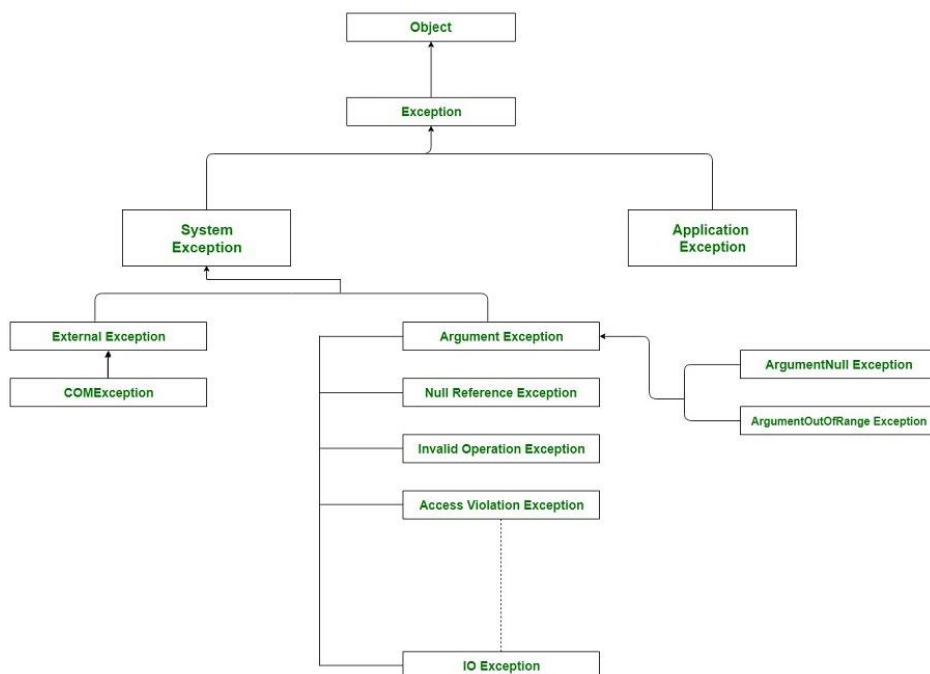
Интерфејс се као и апстрактна класа не може инстанцирати. Имплементација интерфејса у потпуности зависи од класе која га имплементира. Сваки члан интерфејса мора бити јавни и сваки члан мора бити имплементиран у имплементирајућој класи.

За дефиницију интерфејса користи се кључна реч *Interface*. Неписано је правило да се у називу интерфејса стави префикс 'I' како би се знало да је реч о интерфејсу. [Радовановић,2015]

```
interface IBankAccount
{
    void Withdraw(decimal amount);
    void PayIn(decimal amount);
    decimal Balance { get; }
}
```

3.3.6. Изузеци

Изузеци су инстанце специјализованих класа које су све изведене из базне класе *Exception*. Она се даље грана на *SystemException* и *ApplicationException*. *SystemException* је базна класа која обезбеђује изузетке за све CLR или програмске грешке. Дакле, ова класа покрива све системске грешке. Ту су сви познати изузеци попут: *DivideByZeroException* or *NullReferenceException* итд. *ApplicationException* је базна класа изузетака који обрађују грешке специфичне за саму апликацију. Зову се још и кориснички дефинисани изузеци, јер корисник може сам направити своју *exception* класу у складу са потребама његове апликације.



Слика 9 - Изузеци у C# програмском језику

Често програмери у мањим програмима користе само *Exception* изузетке за хватање изузетка. Битно је да се разуме да листа изузетака има своју властиту хијерархију и да се само по њеном редоследу може користити више *catch* блокова. Постоји доста различитих изузетака који се могу појавити у C# програму. Неке од различитих *SystemException* класа:

Divide By Zero exception - Догађа се када корисник у коду дели нулом одређени број.

Out of Memory exceptions - Изузетак који је бачен када нема довољно меморије да би се програм извршио.

Index out of bound Exception- Изузетак који настаје приликом приступа елементу низа или индексу који не постоји у том низу.

Stackoverflow Exception- Узрокован бесконачним рекурзивним процесом. Настаје када стек „поплави“ јер садржи превише угњеждених метода.

Null Reference Exception - Догађа се када корисник покуша да референцира објект који је типа NULL.

Изузетак у контексту рада програма можемо дефинисати као нежељени тј. неочекивани догађај. Он се одиграва током извршавања програма ,па самим тим и омета планирани ток извршења програмских инструкција. Међутим, изузетак постоји са циљем да обавести о насталој грешки у коду. Када се деси нека грешка, метода у којој се десила грешка баца изузетак који се односи на ту грешку. Изузетак се затим хвата (*catch*) и обрађује, како не би дошло до прекидања рада програма. Да би се изузетак обрадио потребно је да метода буде у оквиру *Try-catch-finally* конструкције специјално намењене за хватање, спречавање и исправљање грешака. Место у програму где се хвата грешка, назива се тачка пресретања грешке.

Exception класа поседује бројна својства која помажу кориснику да добије информације о изузетку током његовог догађања, олакшају дијагнозу и руковање грешком до које долази

у току рада програма. Својство *Message* садржи читљив опис грешке као и било које друге важне информације о грешки. Пружа детаљне информације о главном узроку изузетка. Такође, својство *StackTrace*, које садржи *Trace* – запис стека да би се омогућило праћење и прецизно проналажење места где се појавила грешка у коду. *TargetSite* је својство које помаже у проналажењу имена методе у којој је изузетак бачен. *HelpLink* својство које садржи URL одређеног изузетка, и својство *InnerException* које обезбеђује информације о серији изузетака који су се потенцијално догодили.

Још један израз који се често користи приликом манипулације изузецима је *Throwing Exception* тачније исказ *throw* којим се избацује изузетак тј. грешка. Бацање изузетака се користи када се не може у потпуности разрешити грешка у време извршавања, али постоји потреба за делимичном обрадом. Избацивање изузетака није увек пожељно, али је корисно кад је у питању тестирање.

У C++ програмском језику програмери обазриво користе изузетке због пратећег пада перформанси. У C# програмском језику то није такав случај. У Visual Basic-у су грешке сведене на *On Error Go To*, тако да рад са изузецима у C# програмском језику јесте изузетна предност у односу на друге програмске језике. [Радовановић,2015]

3.3.7. Нити

Конкурентност представља ситуацију када два или више процеса захтевају истовремено извршавање. Последица конкурентности је и истовремено коришћење ресурса система од стране процеса. Ова појава може изазвати неко непредвиђено понашање система, за које би ваљало пронаћи решење. Једно решење је увођење међусобног искључења чиме би се могао отклонити овај проблем. Међутим, међусобно искључење може довести и до проблема попут мртвог закључавања (*deadlock*) и гладовања (*starvation*).

Пре него што се опише суштина ова два проблема, треба прокоментарисати појам мултитаскинг-а. У случају мултипроцесорских компјутера, истовремено коришћења ресурса од стране више процеса није проблем јер је постојањем више CPU јединица омогућено паралелно извршавање процеса.

Међутим, када је потребно омогућити исто на компјутерима са једном CPU долази до претходно поменутих проблема. Мултитаскинг је метода која решава проблем истовременог коришћења дељених ресурса од стране више процеса (задатака, таск-ова) попут CPU (Central Processing Unit) ресурса. Постојање једног CPU-а подразумева да се у неком тренутку времена само један задатак може извршити. Мултитаскинг омогућава да се да се за одређени период времена више процеса извршава тако што се прави распоред где сваки процес добија свој редослед у извршавању као и време извршавања. Додељивање процесорског времена задацима се назива контекстна додела. Када се она догађа учестало, ствара се илузија паралелног извршења више задатака.

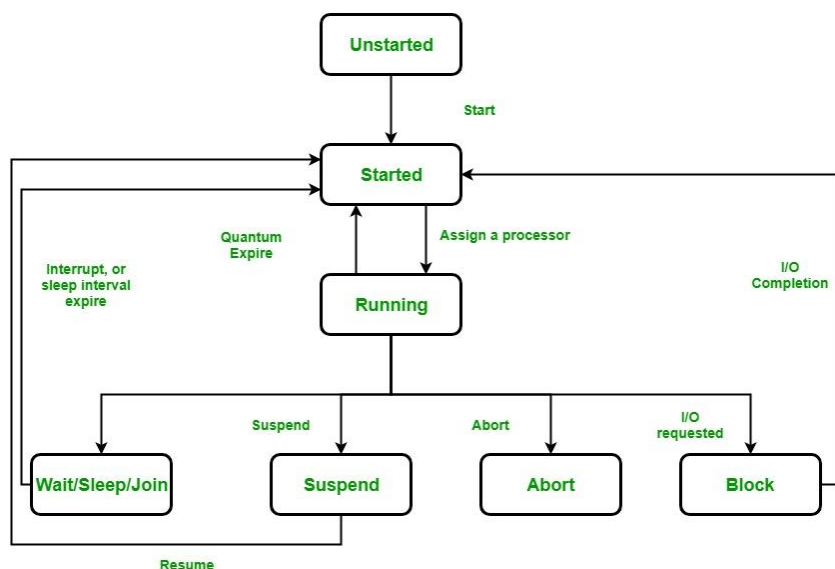
Deadlock је мултитаскинг проблем који настаје у ситуацији када два процеса заузму ресурсе и међусобно чекају један другог да ослободе те ресурсе. *Starvation* је такође мултитаскинг проблем који се јавља у ситуацијама када један процес заузме ресурсе и не дозвољава другим процесима да користе те ресурсе, услед чега остали процеси не могу до краја да се изврше.

Појавом нити (*threads*) решен је проблем комуникације и размене података између процеса јер они сада деле исти меморијски простор. То значи да један програм може да обавља више нити истовремено. C# је један од програмских језика који подржава вишенитно програмирање. Када програм у C#-у почне да се извршава, он аутоматски креира и извршава једну нит која се зове главна програмска нит.

Нити, као што је речено, деле исти адресни простор у оквиру једног процеса и комуникација између њих је доста једноставнија у односу на комуникацију између процеса. Радом са више процеса управља оперативни систем, док радом са више нити управља C# окружење.

Нити би требало периодично да позивају *sleep()* методу како би другим нитима пружиле шансу да се изврше и како би се укинула могућност да из неког разлога имају монопол над системом. Нити које ипак не дају шансу другима да се изврше, називају се “себичне” нити.

Уколико се јави потреба да две или више нити деле заједнички ресурс, мора се обезбедити механизам које ће омогућити искључиви (ексклузивни) приступ једне нити до заједничког ресурса. Монитор обезбеђује тај механизам. Када нека нит уђе у монитор, ниједна друга нит не може у њега ући, све док тренутна нит не изађе из њега. Поступак којим монитор обезбеђује наведени механизам назива се синхронизација.



Слика 10 - Животни циклус и стања нити у C# програмском језику

3.4. Систем за управљање базом података – DBMS Microsoft SQL Server

Microsoft SQL Server је систем за управљање релационим базама података (database management system - DBMS) чија се улога огледа у процесирању бројних трансакција,

пословној интелигенцији и у подржавању разних апликација корпоративних ИТ окружења.

Microsoft SQL Server спада у једну од три водеће технологије за базе података, поред Oracle Database и IBM's DB2. У претходној реченици, акценат треба ставити на реч систем, јер је улога Microsoft SQL Server-а као система да управља великим бројем различитих база података. DBMS је и моћан систем кога користе бројне организације различитих величина: од малих фирми до великих корпорација.

Такође, DBMS доноси и скуп алата који помажу у креирању, мењању и генерално манипулацији података. Ту су и алати за писање извештаја, импортовање/експортовање података као и алати за анализу података.

Историја настанка Microsoft SQL Server система и свих његових верзија које су се смењивале до данас се може представити кроз пар реченица. Наиме, Sybase и Microsoft су објавили прву верзију 1.0 1989. године, међутим сарадња између ове две корпорације је већ почетком 90-их била прекинута, а Microsoft је задржао право на име SQL Server. Верзије које су се даље смењивале: SQL Server 2000, 2005, 2008, 2012, 2014, 2016 и последња најновија верзија објављена у новембру 2019. године.

Као и многи системи за управљање релационим базама података, и Microsoft SQL Server користи Structured Query Language (SQL) – упитни језик кога администратори и ИТ стручњаци користе да управљају подацима у бази и да писањем упита претражују базу. Међутим, SQL Server систем садржи T-SQL (Transact-SQL) – унапређену верзију SQL-а имплементирану од стране Microsoft-а са посебним екстензијама додатим стандардном упитном језику.

4. Алати за развој софтвера

.NET framework, као технологија отвореног кода нуди широк спектар алата за развој различитих апликација. Софтвер за психолошко саветовалиште који је развијен за потребе овог рада, имплементиран је у Visual Studio-у 2017.

4.1. Microsoft Visual Studio 2017

Visual Studio је интегрисано развојно окружење (IDE) које служи за писање, унапређивање и тестирање кода различитих врста апликација, као и публиковање истих.

IDE је окружење богато алатима за развој софтвера. Поред основних алата попут код едитора и дибагера који се налазе и у многим другим окружењима, Visual Studio укључује и компајлере, графичке дизајнере, code completion алате и друге plugin-ове који олакшавају процес развоја софтвера и које је могуће инсталирати ради побољшања функционалности.

Microsoft Visual Studio је као програмерско окружење првенствено створено због израде десктоп апликација за Microsoft Windows. Поред тога, Microsoft Visual Studio 2017 се користи и за израду одличних апликација и игрица за веб, Windows Store, Android и iOS. Visual Studio подржава 36 програмских језика и дозвољава код едитору и дибагеру да их мењају. Уграђени програмски језици су C, C++, VB.NET, C#, F# и TypeScript.

Неколико кључних алата у Visual Studio-у 2017 које ће свако некада користити:

Solution Explorer – користи се за преглед, навигацију и управљање код фајловима. Могуће их је организовати у фолдере, пројекте и solution-е.

Прозор editor-а – где се креира, пише, мења и приказује садржај фајлова. Овде се такође креира кориснички интерфејс у виду нпр. форми са дугмићима и лабелама.

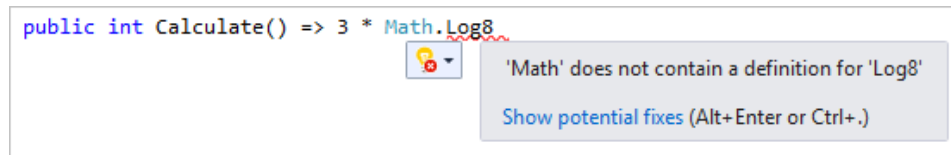
Team Explorer – омогућава размену (претрагу, приказ, дељење) фајлова и кодова између више људи уз помоћ version control технологија попут Git-а и Team Foundation Version Control - а(TFVC).

Новитети које је донео Visul Studio 2017 у односу на претходне верзије је:

- Редифинисане фундаменталне карактеристике. Нови доживљај setup-а подразумева брже инсталирање према потребама корисника.
- Перформансе и продуктивност. Усредсређен је на нове и модерне могућности развоја мобилних, клауд и десктоп технологија. Такође, Visual Studio се покреће брже, боље реагује и користи мање меморије него раније.
- Развој апликација у облаку са Azure. Уграђени пакет алата Azure омогућава лако креирање апликација у облаку које покреће Microsoft Azure. Visul Studio олакшава конфигурисање, прављење, отклањање грешака, паковање и примену апликација и услуга на Azure.
- Развој апликација за Windows. Употребом UWP template-а у Visul Studio-у 2017 може се креирати један пројекат за све Windows 10 уређаје - PC, таблет, XBox, HoloLens, Surface Hub и још много тога.
- Развој мобилних апликација уз Xamarin, који обједињује мобилне захтеве за више платформи са једним језгром базе и скупом вештина.
- Развој међу платформама. Беспрекорно испоручује софтвер на било коју циљану платформу.DevOps процеси се могу проширити на SQL Server преко Redgate Data Tools-а или .NET Core-а за писање апликација и библиотека које се не модификују преко Windows, Linux и MacOS оперативних система.
- Развој игара. Помоћу Visual Studio Tools for Unity (VSTU) могу се писати скрипте за игре , а затим помоћу његовог моћног дибагера пронаћи и исправити грешке.
- Развој вештачке интелигенције. Помоћу Visual Studio Tools-а за вештачку интелигенцију могу се изградити, тестирати и применити Deep Learning / AI решења која се без проблема интегришу са Azure Machine Learning за робусне експерименте.

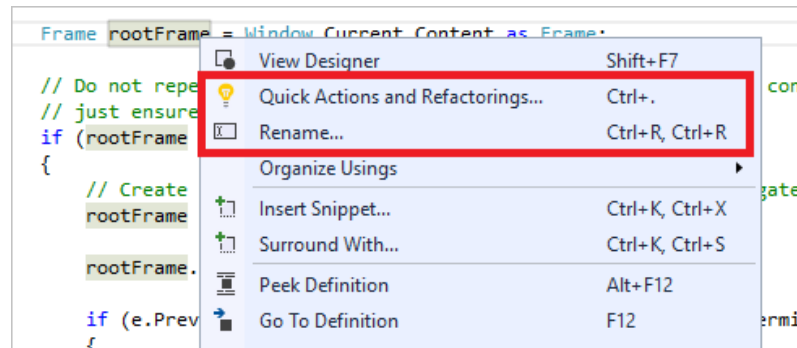
Испод је опис неколико својстава Visual Studio-а 2017 који олакшавају кодирање и убрзавају процес креирања софтвера тј. Апликације:

- **Squiggles и Quick Actions** – то су црвене линије које се појављују у току куцања испод делова кода у којима постоји грешка или потенцијални проблем. Овај визуелни приказ грешке у моменту док се куца код знатно олакшава уочавање грешака и штеди ресурсе, јер није потребно да се код билдује и покрене да би открили грешку.Ако се пређе мишем преко подвученог дела кода, добиће се и додатне информације о грешки. Са леве стране ће се појавити и жута лампица, позната као Quick Actions, која ће наћи начине да се код исправи.



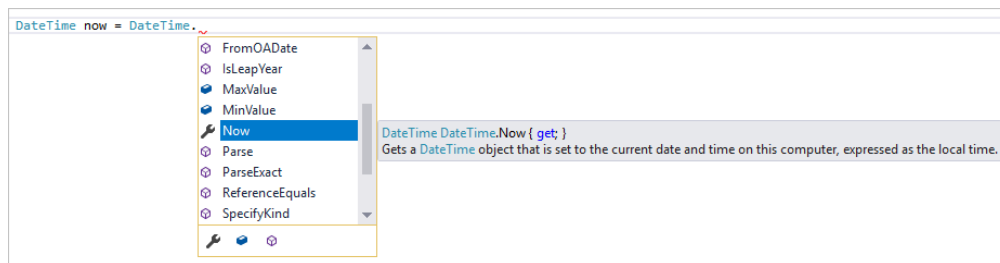
Слика 11 - Squiggles и Quick Actions својство у C#-у

- **Refactoring** – подржава интелигентна решења у виду операција попут оних које препознају када и да ли треба променити име варијабле, направити нову методу од једне или више линија кода, променити редослед параметара у методи итд.



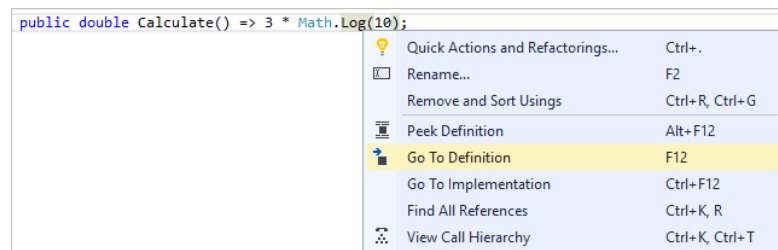
Слика 12 - Refactoring својство у C#-у

- **IntelliSense** – својство које обезбеђује да добијемо више информација о коду који куцамо, а у неким случајевима, пише мале делове кода уместо нас. Уз помоћ овог својства док пишемо код имамо основну документацију свуда и увек са нама, уместо да на другом месту тражимо то што нас занима.



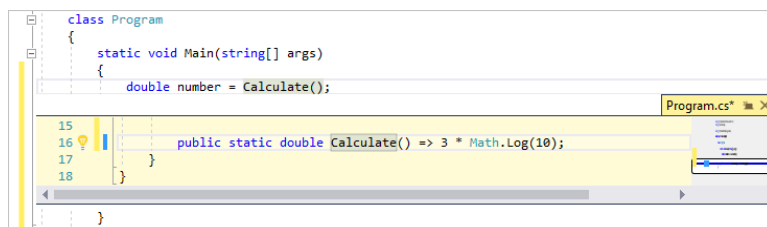
Слика 13 - IntelliSense својство у C#-у

- **Go To Definition** – својство које нас води директно до локације на којој је дефинисана тражена функција или тип податка.



Слика 14 - Go To Definition својство у C#-у

- **Peek Definition** – је прозор који показује дефиницију тражене методе или типа податка без отварања засебног фајла у коме се налази метода.



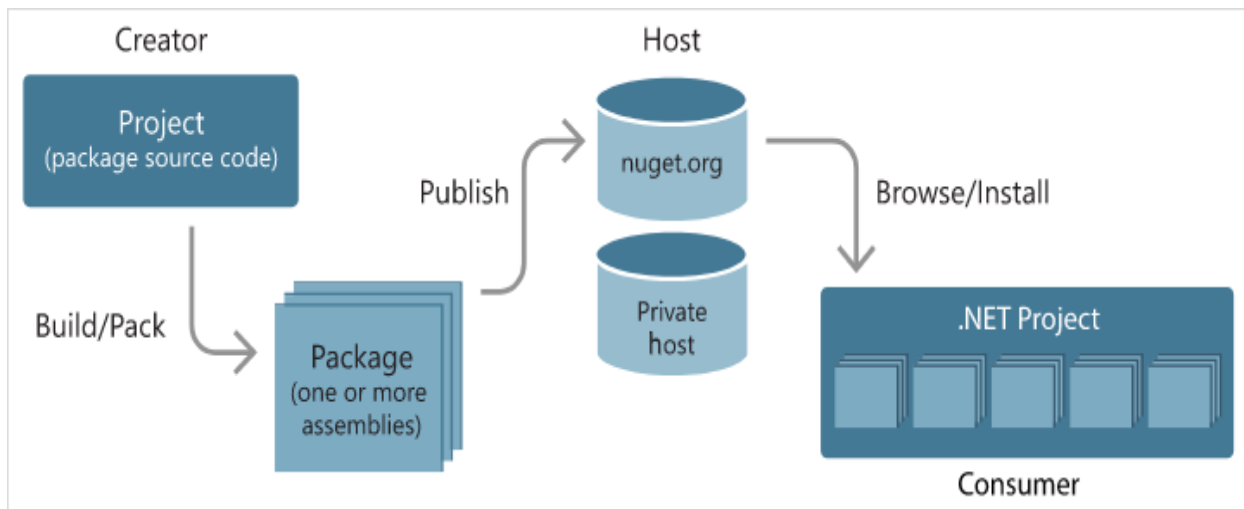
Слика 15 - Peek Definition својство у C#-у

4.2. NuGet пакет менаџер

NuGet је бесплатни и *open-source* пакет менаџер за .NET окружење које омогућује ИТ стручњацима да креирају, деле и користе .NET библиотеке. Често су те библиотеке спаковане у пакете који садрже компајлиран код (DLLs) и додатни садржај неопходан за крајње коришћење пакета у пројектима.

У својој улози као јавни сервис, NuGet у својој централној респозиторији садржи више од 100,000 јединствених пакета који се могу пронаћи на nuget.org. NuGet пакете користе милиони .NET/.NET Core програмера свакога дана. Такође, NuGet омогућује приватно хостовање пакета (на облаку нпр. Azure DevOps, на приватној мрежи или на локалном фајл систему). Приватно хостовање значи да су пакети доступни тј. могу их користити само они програмери који имају приступ систему на коме су хостовани пакети и то је добро уколико желимо да контролишемо број људи који имају приступ пакетима.

Као што је приказано на слици, било да је јавни или приватни, хост је спона између стручњака који креирају пакете и крајњих корисника који имплементирају пакете у својим пројектима. Програмери креирају пакете и објављују их на хосту, а крајњи корисници их користе као помоћ при креирању својих апликација тако што пре свега скину пакете преко доступног хоста, а затим их инсталирају и импортују у свој пројекат. Од тог тренутка, API-ији пакета су доступни у целом пројекту.



Слика 16 - Приватно и јавно NuGet хостовање

4.3. MSTest framework

Битан део сваког framework-а за тестирање корисничког интерфејса је употреба UNIT test framework-а. Један од најпопуларнијих у .NET свету је MSTest. MSTest framework омогућава писање једноставних UNIT тестова коришћењем кључних речи као што су [TestClass] испред назива класе у којој ће тестови бити написани и [TestMethod] испред назива методе која тестира неку од функционалности апликације. У поглављу 11 биће приказана конкретна имплементација ових тестова.

4.4. Microsoft SQL Management Server Studio (SSMS)

SQL Server Management Studio (SSMS) је интегрисано окружење за управљање било којом SQL инфраструктуром. SSMS се користи за приступ, конфигурисање, управљање, администрирање и развој свих компоненти SQL Сервера, Azure SQL базе података и SQL складишта података.

SSMS је популарно и широко коришћено окружење од стране програмера и администратора база података. Разлог су следеће предности: SSMS је бесплатно окружење, пружа корисницима напредно искуство при коришћењу, садржи бројне add-in опције, лако се инсталира.

SQL Server Management Studio обезбеђује свеобухватни услужни програм који поседује широку групу графичких алата са великим бројем богатих едитора скрипти како би омогућио приступ SQL Server-у за програмере и администраторе база података било ког нивоа образовања. [Milener, 2019]

Основне компоненте SQL Server Management Studio-а:

Object Explorer - пружа приказ свих објеката SQL Server-а. Кориснички интерфејс који се

користи да би се управљало свим објектима једне или више инстанци SQL Server-а.

Template Explorer - SQL Server поседује велики број темплејта(стандардизованих фајлова) који се користе да би се убрзало креирање разних објеката у бази. Уз помоћ темплејта могуће је креирати објекте попут саме базе података, табела, погледа, индекса, процедура, тригера, статистика и функција.

Solution Explorer - подржава смештање контекстно повезаних објеката (скрипте, упите, конекције база и фајлове) у контејнере које називамо пројектима. Један или више пројеката који су повезани међусобно могу затим бити смештени у контејнер који називамо solution. Solution се састоји од једног или више пројеката, плус фајлова и метадејта који помажу у дефинисању solution-а као целине. Пројекат са састоји од скупа фајлова , плус метадејта попут конекционих информација.

У овом окружењу се најпре креирају табеле за базу података, након њих се прави дијаграм базе и утврђују везе и ограничења међу табелама. Након креираног дијаграма, база је спремна за коришћење. Да би се апликација повезала за базом неопходан је конекциони стринг ка бази на серверској страни апликације. С обзиром да је овај завршни рад имплементиран на .NET платформи развијеној од стране Microsoft-а, баш као што је и овај систем за управљање базом података, повезивање саме базе и апликације је веома једноставно.

5. Рад у мрежи

У овом поглављу у фокусу ће бити опис сокета као механизма који омогућава комуникацију између програма који се извршавају на различитим рачунарима у мрежи. Сокети су пројектовани тако да подрже имплементацију клијент/сервер апликација.

5.1. Адреса рачунара

Сваки рачунар на глобалној мрежи (Интернету) мора имати своју јединствену адресу (Internet address) која га идентификује у мрежи. Та адреса се састоји од 4 троцифрена броја раздвојена тачкама, при чему бројеви могу узимати вредност из опсега од 0 – 255. Први од бројева дефинише државу, или регион унутар државе, а последњи одређује рачунар који прима поруку.

Постоје три типа адреса: Адреса типа А, Б, И Ц од којих Б и Ц има још увек на располагању, али Б има веома ограничену дистрибуцију. Ограничење до 255 ће представљати проблем у будућности. Зато се ради на еволуцији протокола IP ка протоколу IPng (ng – next generation) у коме се простор шири од 32 бита на 128 бита.

Поред нумеричке адресе рачунара постоји и симболичка адреса рачунара, која се представља преко скупа симбола (знакова). Уведена је јер корисницима није погодно коришћење нумеричких адреса које се тешко памте. Симболичке адресе се састоје од имена рачунара и домена коме рачунар припада. Претварање симболичких у нумеричке адресе обавља се аутоматски од стране Сервиса за именовање домена (Domain Naming

Service - DNS). Пример симболичке адресе: sdl.fon.bg.ac.rs

Домени су тако организовани да прате хијерархију институција или организација којима рачунар припада. У горњој адреси, фон означава Факултет организационих наука, бг Београд, ац академску мрежу, а rs је ознака државе и то је домен највишег нивоа.

Повезивање рачунара у мрежу и пренос података између њих се остварује најчешће помоћу TCP/IP (Transmission Control Protocol/Internet Protocol) протокола. IP протокол је одговоран да пронађе интернет адресу рачунара коме се шаљу подаци и да усмери податке ка њему од изворног рачунара који је пошиљалац. TCP је задужен за успостављање и раскидање везе између рачунара, као и за одређене контролне функције. Често се за интернет адресу каже да је то IP адреса.

5.2. URL адреса

Нумеричке и симболичке адресе рачунара служе да се преко њих приступи жељеним рачунарима у мрежи. Уколико се жели приступити до одређених сервиса и датотека на рачунару користи се УРЛ (URL - Uniform Resource Locator) адреса. Треба је схватити као мрежно проширење стандардног датотечног концепта.

Општи облик УРЛ-а увео је Тим Бернерс-Ли за примену на World Wide Web-у, на пример: **šema://[korisnik:lozinka]@domen[:porta]/[putanja]**.

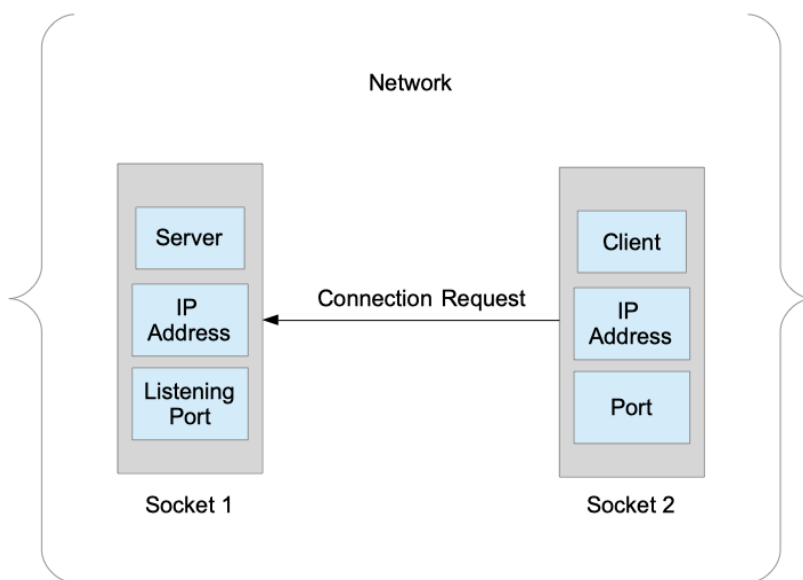
Синтакса почиње протоколом који се користи (http, ftp, gopher или file), који је раздвојен ":" од остатка адресе, у последње време то је најчешће "http:" или "https:". Угласе заграде у овом опису указују да је елемент описа опциони, односно није обавезан. На пример, датотечни УРЛ спецификује се на следећи начин. Ако документ *primer.txt* постоји на анонимном FTP серверу чији је домен *ftp.izmisljen.com* у каталогу */pub/datoteke* онда је УРЛ адреса тог документа: **file://ftp.izmisljen.com/pub/datoteke/primer.txt**

5.3. Сокети

Сокети, у ширем смислу, представљају механизам који омогућава комуникацију између програма који се извршавају на различитим рачунарима у мрежи. У ужем смислу, сокети представљају објекте помоћу којих се шаљу/прихватају подаци ка/од других сокета.

Он је пројектован тако да подржи имплементацију клијент/сервер апликација. Сокети користе TCP/IP протокол при повезивању, контроли и преносу података између два или више програма. Конекција између два програма је остварена када се успостави веза између њихових сокета. При повезивању два програма преко сокета, по један сокет се генерише за сваки програм. Сваки од сокета садржи референцу на други сокет. То практично значи да први сокет садржи референцу на други сокет, док други сокет садржи референцу на први сокет.

Сокет је по својој природи улазно-излазни ток па се може направити аналогија у понашању са системским објектима *System.in* и *System.out*. Помоћу објекта *System.in* подаци се прихватају са стандардног улаза (тастатуре), док се код сокета подаци прихватају са спољашњег улаза (са мреже) од другог сокета. Јасно је како би гласила аналогија за системски објект *System.out*. Такође, сокети се повезују са улазно-излазним токовима на сличан начин као што то раде са објектима *System.in* и *System.out*, када желе извршити обраду података међу собом.



Слика 17 - Веза два програма преко сокета

6. Студијски пример развоја софтверског система

Развој студијског примера заснован је на Лармановој методи развоја софтверског система. Ларманова метода састоји се из следећих фаза [Vlajić, 2015]:

- Прикупљање захтева
- Анализа
- Пројектовање
- Имплементација
- Тестирање

Фаза прикупљања захтева постоји да би се идентификовали и дефинисали кориснички захтеви везани за пројекат, тј. софтверски систем. Овде се прикупљају информације о томе како корисник жели да софтвер изгледа и што је много битније, функционише. Захтеви могу бити функционални и нефункционални. У функционалним захтевима дефинисане су све функције које систем треба да обавља. Нефункционални захтеви познати и као квалитативни атрибути софтверског система, дефинишу све остале захтеве: поузданост, употребљивост, перформансе и подрживост система. Функционални захтеви се описују преко модела случаја коришћења. Модели случаја коришћења садрже структуру система и понашање система.

У фази анализе описује се пословна логика система која обухвата структуру система и понашање система. Структура софтверског система се описује преко концептуалног и релационог модела. Понашање система се описује преко секвенчних дијаграма и системских операција.

У фази пројектовања саставља се архитектура будућег софтверског система. Ради се о тронивовској архитектури која се састоји од корисничког интерфејса, апликационе логике и складишта података. Сваки слој тронивовске архитектуре је дефинисан у овој фази. Кориснички интерфејс је дефинисан преко скупа екранских форми тј. преко сценарија коришћења екранских форми који је директно повезан са сценаријима случајева коришћења. Пословна логика, која се добија у фази анализе, преноси се у фазу пројектовања као саставни део апликационе логике. Складиште података се пројектује на основу релационог модела.

У фази имплементације праве се имплементационе компоненте које се реализују у некој технологији (нпр. .NET-у). Имплементационе компоненте су реализација компоненти које су добијене у фази пројектовања.

Фаза тестирања подразумева тестирање сваке од компоненти које су имплементиране у претходној фази. Тестови се спроводе тако што се за сваку имплементациону компоненту прави тест случајева, тест процедуре и тест компоненте.

7. Кориснички захтеви

Овај сегмент односи се на фазу прикупљања корисничких захтева. Представљен је вербални опис модела као и спецификација захтева помоћу модела случајева коришћења.

7.1 Вербални опис модела

Потребно је креирати софтверски систем који ће се користити приликом вођења евиденције о клијентима и њиховим сеансама у психолошком саветовалишту.

Сусрет клијента и психотерапеута се обично одвија на следећи начин: Клијент исприча како се осећа, о чему размишља и која понашања му праве проблем, а психотерапеут бележи битне информације током или на крају сеансе и труди се да процени проблем и пронађе адекватно решење.

Психотерапеут уноси податке о сваком новом клијенту и бележи податке са сваке његове сеансе. Током рада, психотерапеут може променити забележене информације о одређеном клијенту, као и садржај било које сеансе тог клијента. Када се психотерапеут и клијент сложе да је проблем решен, психотерапеут брише клијента из система. Приступ систему има психотерапеут и како би све активности биле извршене потребно је да буде пријављен на систем.

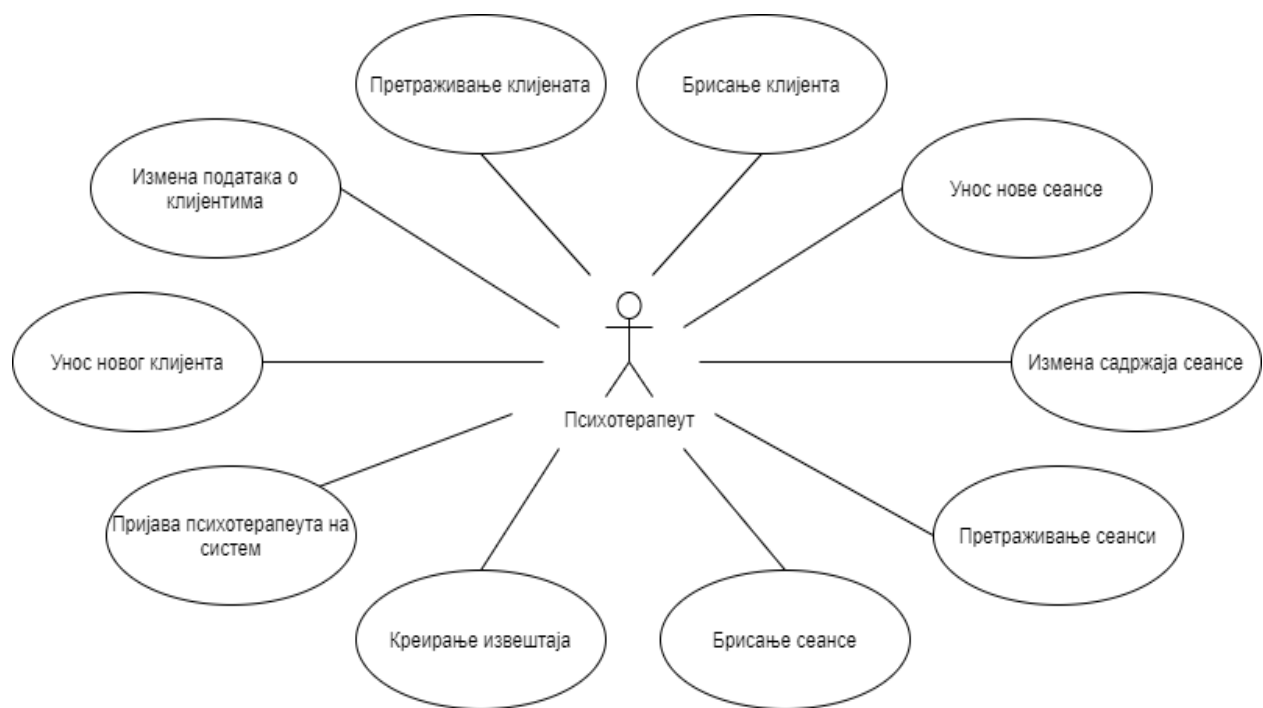
Софтверски систем треба да омогући лакше чување и управљање подацима клијената и њихових сеанси.

7.2. Спецификација захтева помоћу модела случајева коришћења

Модел СК се састоји од скупа случаја коришћења (СК), актора (АК) и веза између случаја коришћења и актора. Случај коришћења описује скуп сценарија односно скуп жељених коришћења система од стране актора.

Психотерапеуту, као кориснику система, треба омогућити следеће функционалности:

- пријаву на систем,
- унос новог клијента,
- претраживање клијената,
- измене података о клијентима,
- брисање клијента,
- **унос нове сеансе (сложен случај коришћења)**
- претрагу сеанси,
- измену садржаја сеансе,
- брисање сеансе
- креирање извештаја



Слика 18 - Дијаграм Случајева коришћења

Случајеви коришћења

Случајеви коришћења се у почетним фазама развоја софтвера представљају текстуално док се касније они представљају преко секвенчних дијаграма, дијаграма сарадње, дијаграма прелаза стања или дијаграма активности.

Текстуални опис СК има следећу структуру:

- Назив СК.
- Акторе СК.
- Учеснике СК.
- Предуслови који морају бити задовољени да би СК почео да се извршава.
- Основни сценаријо извршења СК.
- Постуслови који морају бити задовољени да би се потврдило да је СК успешно извршен.
- Алтернативна сценарија извршења СК.
- Специјални захтеви.
- Технолошки захтеви.
- Отворена питања. [Влајић,2015]

СК1: Пријава психотерапеута на систем

Назив СК

Пријава психотерапеута на систем

Актори СК

Психотерапеут

Учесници СК

Психотерапеут и систем (програм)

Предуслов: Систем је укључен. Систем приказује форму за пријављивање.

Основни сценарио СК

1. Психотерапеут уноси корисничко име и лозинку. (АПУСО)
2. Психотерапеут контролише да ли је коректно унео корисничко име и лозинку. (АНСО)
3. Психотерапеут позива систем да се пријави (провери податке). (АПСО)
4. Систем проверава податке о психотерапеуту. (СО)
5. Систем приказује психотерапеуту поруку: "Uspešno ste se prijavili!" (ИА)

Алтернативна сценарија

5.1. Уколико систем не може да нађе психотерапеута он приказује психотерапеуту поруку: "Neuspešno prijavljivanje!" (ИА)

СК2: Унос новог клијента

Назив СК

Унос новог клијента

Актори СК

Психотерапеут

Учесници СК

Психотерапеут и систем (програм)

Предуслов: Систем је укључен и психотерапеут је улогован под својом шифром. Систем приказује форму за рад са клијентима. Учитана је листа градова.

Основни сценарио СК

1. Психотерапеут уноси податке о клијенту. (АПУСО)
2. Психотерапеут контролише да ли је коректно унео податке о клијенту. (АНСО)
3. Психотерапеут позива систем да запамти податке о клијенту. (АПСО)
4. Систем памти податке о клијенту. (СО)
5. Систем приказује психотерапеуту поруку: "Klijent uspešno sačuvan!" (ИА)

Алтернативна сценарија

5.1. Уколико систем не може да запамти податке о клијенту он приказује психотерапеуту поруку: "Klijent nije sačuvan!" (ИА)

СК3: Претраживање клијената

Назив СК

Претраживање клијената

Актори СК

Психотерапеут

Учесници СК

Психотерапеут и систем (програм)

Предуслов: Систем је укључен и психотерапеут је улогован под својом шифром. Систем приказује форму за рад са клијентима. Учитана је листа клијената.

Основни сценарио СК

1. Психотерапеут уноси вредност по којој претражује клијенте. (АПУСО)
2. Психотерапеут позива систем да нађе клијенте по задатој вредности. (АПСО)
3. Систем тражи клијенте по задатој вредности. (СО)
4. Систем приказује психотерапеуту листу клијената и поруку: "Sistem je pronašao klijente". (ИА)
5. Психотерапеут бира клијента чије податке жели да учита. (АПУСО)
6. Психотерапеут позива систем да учита податке о одабраном клијенту. (АПСО)
7. Систем учитава податке о одабраном клијенту. (СО)
8. Систем приказује психотерапеуту податке о клијенту и поруку: "Sistem je pronašao klijenta" (ИА)

Алтернативна сценарија

4.1. Уколико систем не може да нађе клијенте, систем приказује психотерапеуту поруку: "Klijenti koji odgovaraju zadatoj vrednosti nisu pronađeni!" Прекида се извршење сценарија. (ИА)

8.1. Уколико систем не може да пронађе податке о клијенту, систем приказује психотерапеуту поруку: "Klijent nije pronađen!" (ИА)

СК4: Измена података о клијенту

Назив СК

Измена података о клијенту

Актори СК

Психотерапеут

Учесници СК

Психотерапеут и систем (програм)

Предуслов: Систем је укључен и психотерапеут је улогован под својом шифром. Систем приказује форму за рад са клијентима. Учитана је листа клијената. Учитана је листа градова.

Основни сценарио СК

1. Психотерапеут уноси вредност по којој претражује клијенте. (АПУСО)
2. Психотерапеут позива систем да нађе клијенте по задатој вредности. (АПСО)
3. Систем тражи клијенте по задатој вредности. (СО)
4. Систем приказује психотерапеуту листу клијената и поруку: "Sistem je pronašao klijente". (ИА)

5. **Психотерапеут** бира клијента чије податке жели да измени. (АПУСО)
6. **Психотерапеут** позива систем да учита податке о одабраном клијенту. (АПСО)
7. **Систем** учитава податке о одабраном клијенту. (СО)
8. **Систем** приказује психотерапеуту податке о клијенту и поруку: "Sistem je pronašao klijenta". (ИА)
9. **Психотерапеут** мења податке о клијенту. (АПУСО)
10. **Психотерапеут** контролише да ли је коректно унео податке о клијенту. (АНСО)
11. **Психотерапеут** позива систем да запамти податке о клијенту. (АПСО)
12. **Систем** памти податке о клијенту. (СО)
13. **Систем** приказује психотерапеуту поруку: "Klijent uspešno izmenjen!" (ИА)

Алтернативна сценарија

4.1. Уколико **систем** не може да нађе клијенте он приказује психотерапеуту поруку: "Klijenti koji odgovaraju zadatoj vrednosti nisu pronađeni!" Прекида се извршење сценарија. (ИА)

8.1. Уколико **систем** не може да пронађе податке о клијенту, **систем** приказује психотерапеуту поруку: "Klijent nije pronađen!" Прекида се извршење сценарија. (ИА)

13.1. Уколико **систем** не може да запамти податке о клијенту он приказује психотерапеуту поруку: "Klijent nije izmenjen!" (ИА)

СК5: Брисање клијента

Назив СК

Брисање клијента

Актери СК

Психотерапеут

Учесници СК

Психотерапеут и **систем** (програм)

Предуслов: **Систем** је укључен и **психотерапеут** је улогован под својом шифром. **Систем** приказује форму за рад са клијентима. Учитана је листа клијената.

Основни сценарио СК

1. **Психотерапеут** уноси вредност по којој претражује клијенте. (АПУСО)
2. **Психотерапеут** позива систем да нађе клијенте по задатој вредности. (АПСО)
3. **Систем** тражи клијенте по задатој вредности. (СО)
4. **Систем** приказује психотерапеуту листу клијената и поруку: "Sistem je pronašao klijente". (ИА)
5. **Психотерапеут** бира клијента ког жели да обрише. (АПУСО)
6. **Психотерапеут** позива систем да учита податке о одабраном клијенту. (АПСО)
7. **Систем** учитава податке о одабраном клијенту. (СО)
8. **Систем** приказује психотерапеуту податке о клијенту и поруку: "Sistem je pronašao klijenta". (ИА)
9. **Психотерапеут** позива систем да обрише клијента. (АПСО)
10. **Систем** брише клијента. (СО)
11. **Систем** приказује психотерапеуту поруку: "Klijent obrisan!" (ИА)

Алтернативна сценарија

4.1. Уколико **систем** не може да нађе **клијенте** он приказује **психотерапеуту** поруку: "Klijenti koji odgovaraju zadatoj vrednosti nisu pronađeni!" Прекида се извршење сценарија. (ИА)

8.1. Уколико **систем** не може да пронађе податке о **клијенту**, **систем** приказује **психотерапеуту** поруку: "Klijent nije pronađen!" Прекида се извршење сценарија. (ИА)

11.1. Уколико **систем** не може да обрише **клијента** он приказује **психотерапеуту** поруку: "Klijent nije obrisan!" (ИА)

СК6: Унос нове сеансе

Назив СК

Унос нове **сеансе**

Актери СК

Психотерапеут

Учесници СК

Психотерапеут и **систем** (програм)

Предуслов: **Систем** је укључен и **психотерапеут** је улогован под својом шифром. **Систем** приказује форму за рад на **сеансама**. Учитана је листа клијената. Учитане су врста и облик психотерапије.

Основни сценарио СК

1. **Психотерапеут** **попуњава** податке о **сеанси**. (АПУСО)
2. **Психотерапеут** **контролише** да ли је коректно унео податке о **сеанси**. (АНСО)
3. **Психотерапеут** **позива** **систем** да запамти податке о **сеанси**. (АПСО)
4. **Систем** **памти** податке о **сеанси**. (СО)
5. **Систем** **приказује** **психотерапеуту** поруку: "Seansa uspešno sačuvana!" (ИА)

Алтернативна сценарија

- 5.1. Уколико **систем** не може да запамти податке о **сеанси** он приказује **психотерапеуту** поруку: "Seansa nije sačuvana!" (ИА)

СК7: Претраживање сеанси

Назив СК

Претраживање **сеанси**

Актери СК

Психотерапеут

Учесници СК

Психотерапеут и **систем** (програм)

Предуслов: **Систем** је укључен и **психотерапеут** је улогован под својом шифром. **Систем** приказује форму за рад са **сеансама**. Систем приказује листу **сеанси**.

Основни сценарио СК

1. **Психотерапеут** **уноси** вредност по којој претражује **сеансе**. (АПУСО)
2. **Психотерапеут** **позива** **систем** да нађе **сеансе** по задатој вредности. (АПСО)

3. Систем тражи сеансе по задатој вредности. (СО)
4. Систем приказује психотерапеуту листу сеанси и поруку: "Sistem je pronašao seanse". (ИА)
5. Психотерапеут бира сеансу чије податке жели да учита. (АПУСО)
6. Психотерапеут позива систем да учита податке о одабраној сеанси. (АПСО)
7. Систем учитава податке о одабраној сеанси. (СО)
8. Систем приказује психотерапеуту податке о сеанси и поруку: "Sistem je pronašao seansu". (ИА)

Алтернативна сценарија

- 4.1. Уколико систем не може да нађе сеансе, систем приказује психотерапеуту поруку: "Seanse koje odgovaraju zadatoj vrednosti nisu pronađene!" Прекида се извршење сценарија. (ИА)
- 8.1. Уколико систем не може да пронађе сеансу он приказује психотерапеуту поруку: "Seansa nije pronađena!" (ИА)

СК8: Измена садржаја сеансе

Назив СК

Измена садржаја сеансе

Актери СК

Психотерапеут

Учесници СК

Психотерапеут и систем (програм)

Предуслов: Систем је укључен и психотерапеут је улогован под својом шифром. Систем приказује форму за рад са сеансама. Учитана је листа сеанси као и листа клијената. Учитана је врста психотерапије као и облик психотерапије.

Основни сценарио СК

1. Психотерапеут уноси вредност по којој претражује сеансе. (АПУСО)
2. Психотерапеут позива систем да нађе сеансе по задатој вредности. (АПСО)
3. Систем тражи сеансе по задатој вредности. (СО)
4. Систем приказује психотерапеуту листу сеанси и поруку: "Sistem je pronašao seanse". (ИА)
5. Психотерапеут бира сеансу чије податке жели да измени. (АПУСО)
6. Психотерапеут позива систем да учита податке о одабраној сеанси. (АПСО)
7. Систем учитава податке о одабраној сеанси. (СО)
8. Систем приказује психотерапеуту податке о сеанси и поруку: "Sistem je pronašao seansu". (ИА)
9. Психотерапеут мења податке о сеанси. (АПУСО)
10. Психотерапеут контролише да ли је коректно унео податке о сеанси. (АНСО)
11. Психотерапеут позива систем да запамти податке о сеанси. (АПСО)
12. Систем памти податке о сеанси. (СО)
13. Систем приказује психотерапеуту поруку: "Seansa uspešno izmenjena!" (ИА)

Алтернативна сценарија

4.1. Уколико **систем** не може да нађе **сеансе** он приказује **психотерапеуту** поруку: "Seanse koje odgovaraju zadatoj vrednosti nisu pronađene!". Прекида се извршење сценарија. (ИА)

8.1 Уколико **систем** не може да пронађе **сеансу** он приказује **психотерапеуту** поруку: "Seansa nije pronađena!" Прекида се извршење сценарија. (ИА)

13.1. Уколико **систем** не може да запамти податке о **сеанси** он приказује **психотерапеуту** поруку: "Seansa nije izmenjena!" (ИА)

СК9: Брисање сеансе

Назив СК

Брисање **сеансе**

Актери СК

Психотерапеут

Учесници СК

Психотерапеут и **систем** (програм)

Предуслов: **Систем** је укључен и **психотерапеут** је улогован под својом шифром. **Систем** приказује форму за рад са **сеансама**. Учитана је листа **сеанси**.

Основни сценарио СК

1. **Психотерапеут** уноси вредност по којој претражује **сеансе**. (АПУСО)
2. **Психотерапеут** позива **систем** да нађе **сеансе** по задатој вредности. (АПСО)
3. **Систем** тражи **сеансе** по задатој вредности. (СО)
4. **Систем** приказује **психотерапеуту** листу **сеанси** и поруку: "Sistem je pronašao seanse". (ИА)
5. **Психотерапеут** бира **сеансу** чије податке жели да учита. (АПУСО)
6. **Психотерапеут** позива **систем** да учита податке о одабраној **сеанси**. (АПСО)
7. **Систем** тражи податке о одабраној **сеанси**. (СО)
8. **Систем** приказује **психотерапеуту** податке о **сеанси** и поруку: „Sistem je pronašao seansu“. (ИА)
9. **Психотерапеут** позива **систем** да обрише одабрану **сеансу**. (АПСО)
10. **Систем** брише **сеансу**. (СО)
11. **Систем** приказује **психотерапеуту** поруку: "Seansa obrisana!" (ИА)

Алтернативна сценарија

4.1. Уколико **систем** не може да нађе **сеансе** он приказује **психотерапеуту** поруку: "Seanse koje odgovaraju zadatoj vrednosti nisu pronađene!". Прекида се извршење сценарија. (ИА)

8.1 Уколико **систем** не може да пронађе податке о **сеанси** он приказује **психотерапеуту** поруку: "Seansa nije pronađena!" Прекида се извршење сценарија. (ИА)

11.1. Уколико **систем** не може да обрише **сеансу** он приказује **психотерапеуту** поруку: "Seansa nije obrisana!". (ИА)

8.Фаза анализе

У фази анализе представљена је *логичка структура* софтверског система, као и његово *понашање*, односно резултат ове фазе је *пословна логика будућег софтверског система*.

Прво следи опис понашања система, што се представља путем *системских дијаграма секвенци* и *системских операција*, а потом ће бити представљена структура система преко *концептуалног* и *релационог модела*.

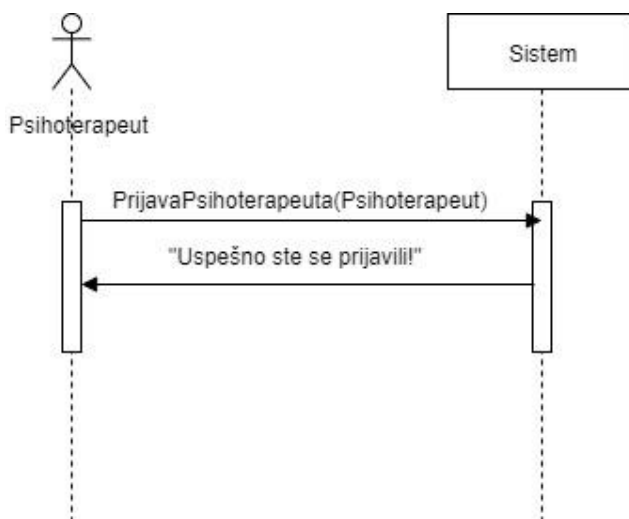
8.1. Понашање софтверског система: Системски дијаграм секвенци

Како би се представило понашање софтверског система, за сваки случај коришћења уочен у *фази прикупљања захтева* приказује се један системски дијаграм секвенци, којим се моделује интеракција између актера и система путем активности у одређеном редоследу.

ДС1: дијаграм секвенци случаја коришћења - Пријава психотерапеута на систем

Основни сценарио СК

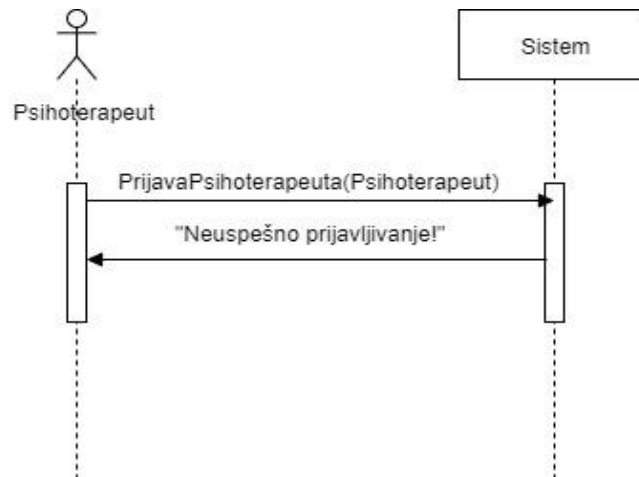
1. **Психотерапеут** **позива** **систем** да се пријави (провери податке). (АПСО)
2. **Систем** **приказује** **психотерапеуту** поруку: "Uspešno ste se prijavili!" (ИА)



Слика 19 - ДС Пријава психотерапеута

Алтернативна сценарија

2.1. Уколико **систем** не може да нађе **психотерапеута** он му приказује поруку: "Neuspešno prijavljivanje! ". (ИА)



Слика 20 - ДС Неуспешно пријављивање

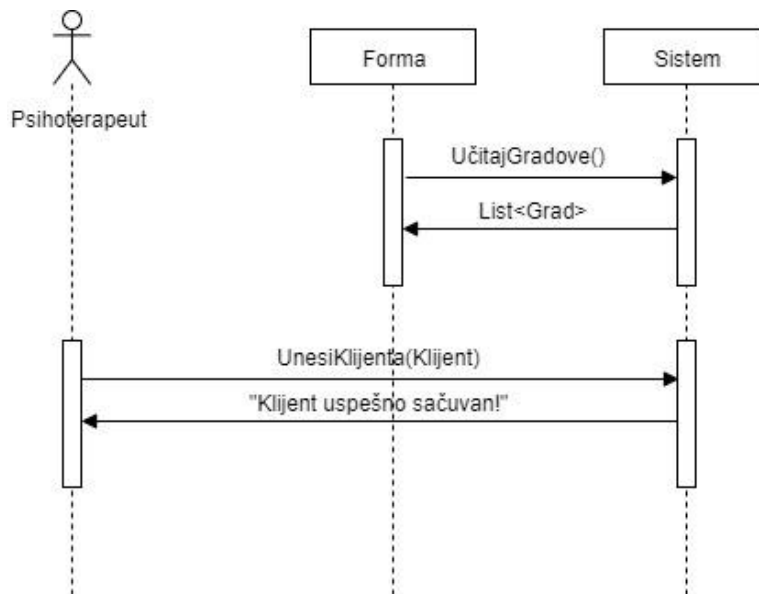
Са наведених дијаграма секвенци уочавају се системске операције које треба пројектовати:

1. *signa*PrijavaPsihoterapeuta(Psihoterapeut)

ДС2: дијаграм секвенци случаја коришћења - Унос новог клијента

Основни сценарио СК

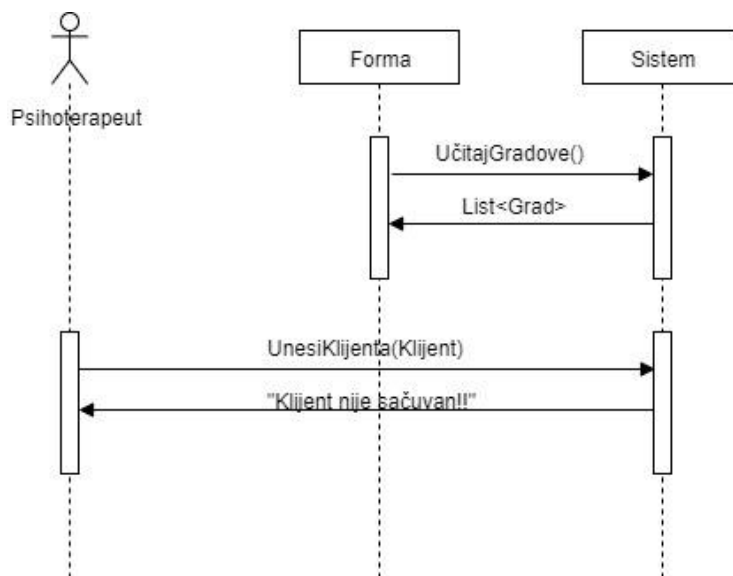
1. Форма **позива систем** да учита листу **градова**. (АПСО)
2. **Систем приказује** листу **градова**. (ИА)
3. **Психотерапеут позива систем** да запамти податке о **клијенту**. (АПСО)
4. **Систем приказује психотерапеуту** поруку: "Klijent uspešno sačuvan!" (ИА)



Слика 21 - ДС Унос клијента

Алтернативна сценарија

4.1. Уколико **систем** не може да запамти податке о **клијенту** он приказује **психотерапеуту** поруку: "Klijent nije sačuvan!" (ИА)



Слика 22 - ДС Клијент није сачуван

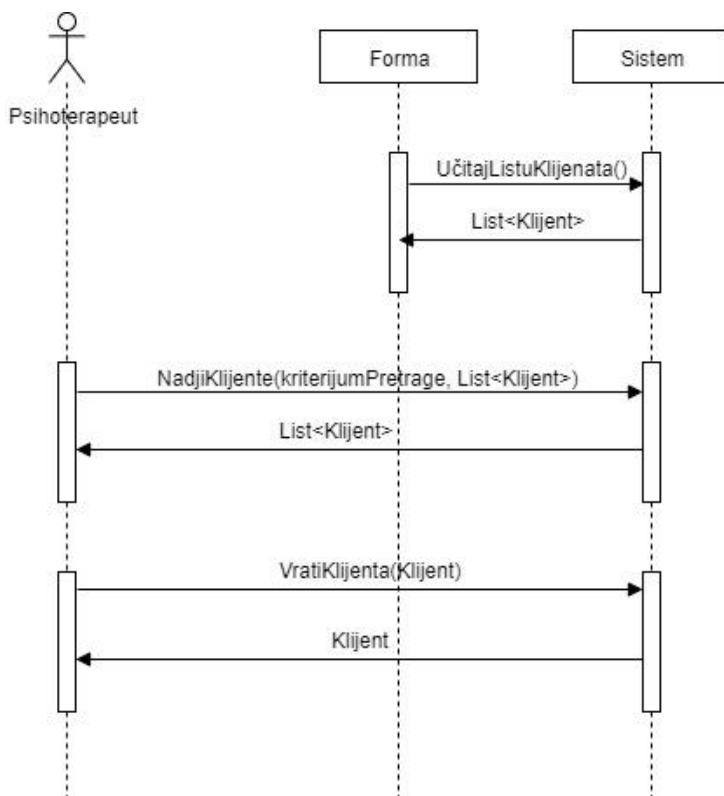
Са наведених дијаграма секвенци уочавају се системске операције које треба пројектовати:

1. *signal* UnesiKlijenta(Klijent)
2. *signal* UčitajGradove(List<Grad>)

ДС3: дијаграм секвенци случаја коришћења - Претраживање клијената

Основни сценарио СК

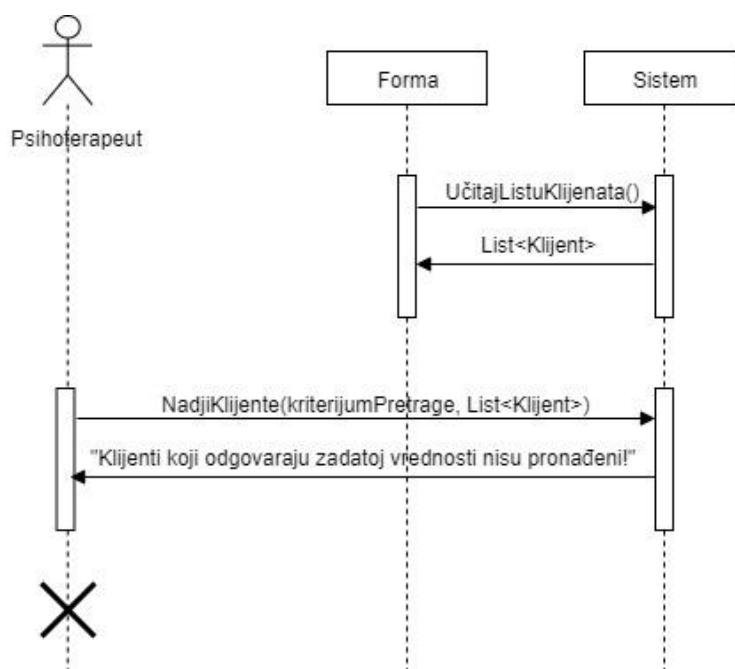
1. Форма **позива** систем да учита листу **клијената**. (АПСО)
2. **Систем приказује** листу **клијената**. (ИА)
3. **Психотерапеут позива** систем да нађе **клијенте** по задатој вредности. (АПСО)
4. **Систем приказује** **психотерапеуту** листу **клијената**. (ИА)
5. **Психотерапеут позива** систем да учита податке о одабраном **клијенту**. (АПСО)
6. **Систем приказује** **психотерапеуту** податке о **клијенту**. (ИА)



Слика 23 - ДС Претраживање клијената

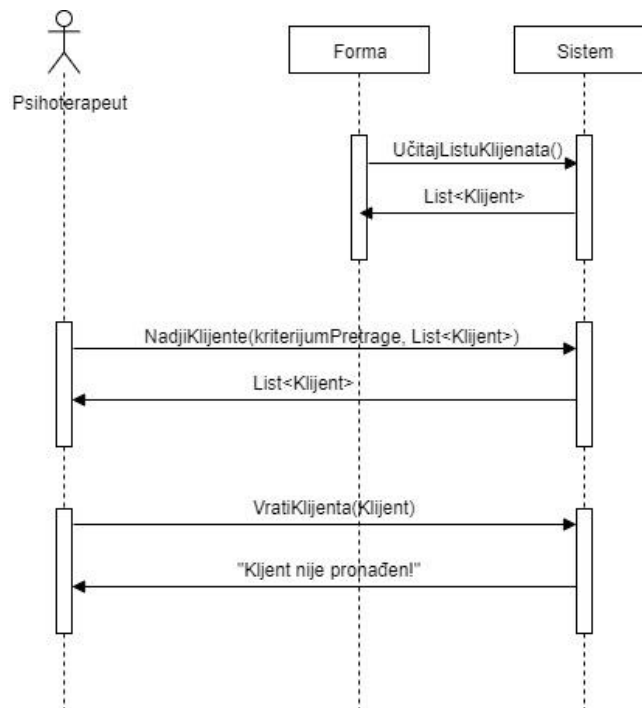
Алтернативна сценарија

4.1. Уколико **систем** не може да нађе **клијенте**, **систем** приказује **психотерапеуту** поруку: "Klijenti koji odgovaraju zadatoj vrednosti nisu pronađeni!" Прекида се извршење сценарија. (ИА)



Слика 24 - ДС Клијенти који одговарају задатој вредности нису пронађени

7.1. Уколико **систем** не може да пронађе податке о **клијенту**, систем приказује **психотерапеуту** поруку: "Klijent nije pronađen!" (ИА)



Слика 25 - ДС Клијент није пронађен

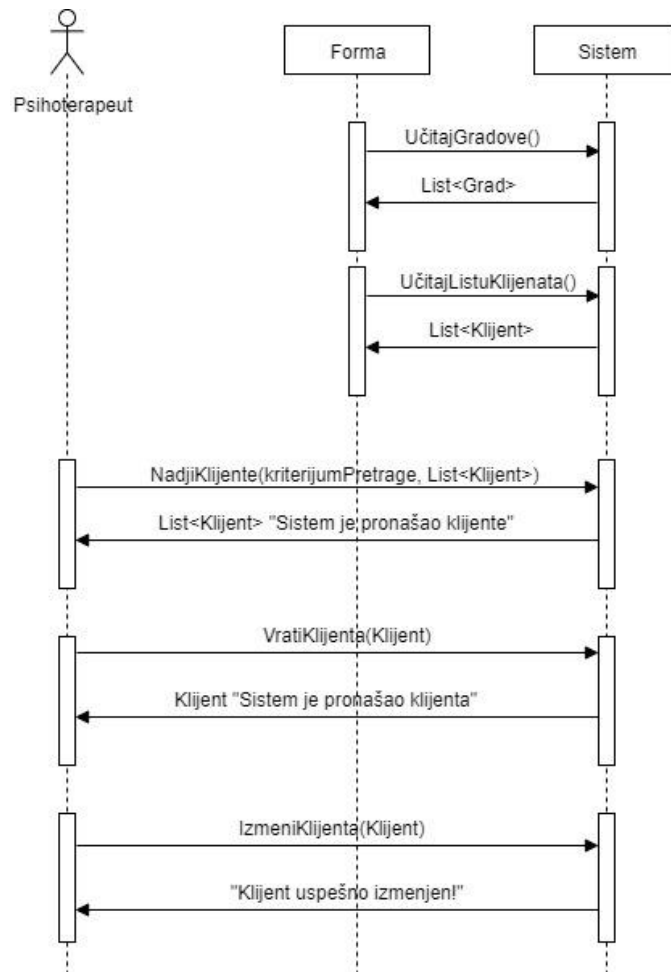
Са наведених дијаграма секвенци уочавају се системске операције које треба пројектовати:

1. *signal* UčitajListuKlijenata(List<Klijent>)
2. *signal* NadjiKlijente(kriterijumPretrage, List<Klijent>)
3. *signal* VratiKlijenta(Klijent)

ДС4: дијаграм секвенци случаја коришћења - Измена података о клијенту

Основни сценарио СК

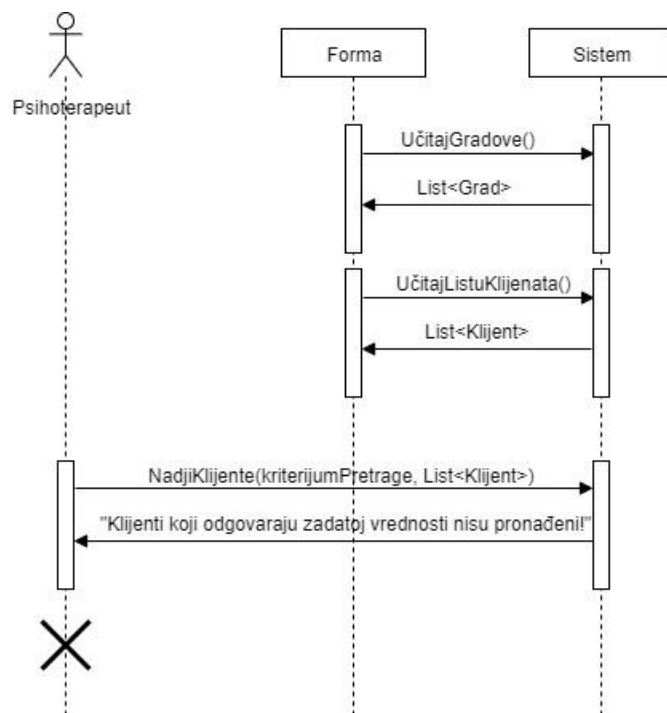
1. Форма **позива** систем да учита листу градова. (АПСО)
2. Систем **приказује** листу градова. (ИА)
3. Форма **позива** систем да учита листу клијената. (АПСО)
4. Систем **приказује** листу клијената. (ИА)
5. **Психотерапеут** **позива** систем да нађе клијенте по задатој вредности. (АПСО)
6. Систем **приказује** психотерапеуту листу клијената. (ИА)
7. **Психотерапеут** **позива** систем да учита податке о одабраном клијенту. (АПСО)
8. Систем **приказује** психотерапеуту податке о клијенту. (ИА)
9. **Психотерапеут** **позива** систем да запамти податке о клијенту. (АПСО)
10. Систем **приказује** психотерапеуту поруку: "Klijent uspešno izmenjen!" (ИА)



Слика 26 - ДС Измена података о клијенту

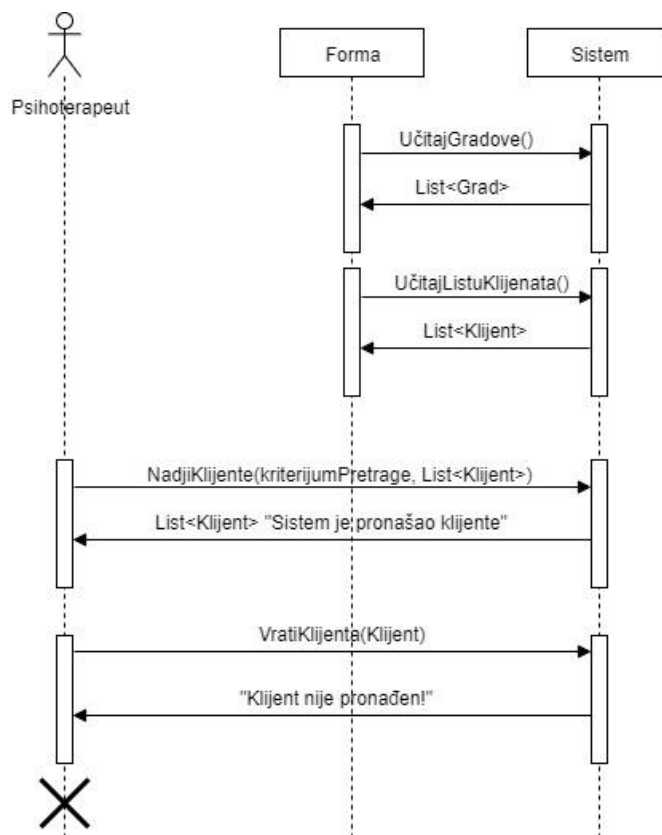
Алтернативна сценарија

6.1. Уколико **систем** не може да нађе **клијенте** он приказује **психотерапеуту** поруку: "Клијенти који одговарају задатој вредности нису пронађени!" Прекида се извршење сценарија. (ИА)



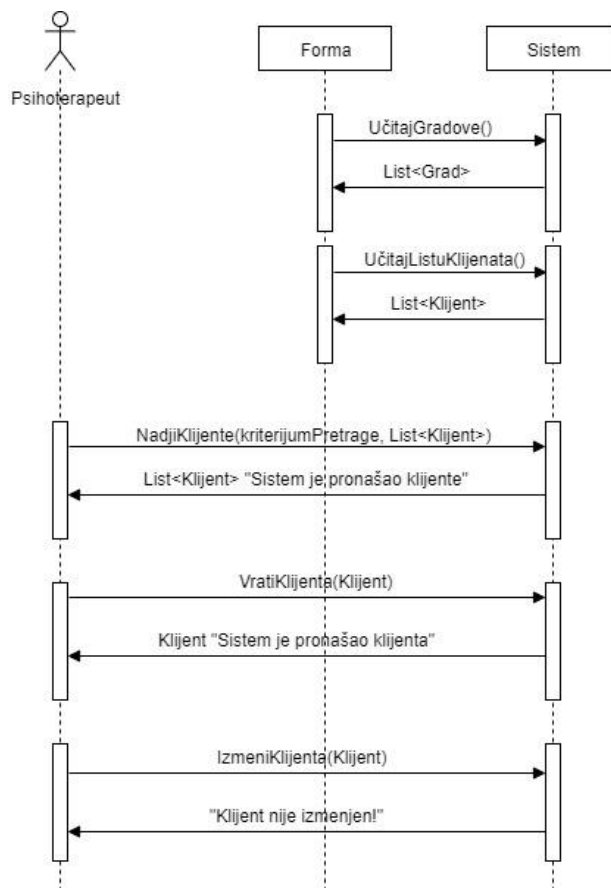
Слика 27 - ДС Клијенти који одговарају садамој вредности нису пронађени

8.1. Уколико **систем** не може да пронађе податке о **клијенту**, **систем** приказује **психотерапеуту** поруку: "Клијент није пронађен!" Прекида се извршење сценарија. (ИА)



Слика 28 - ДС Клијент није пронађен

- 10.1. Уколико **систем** не може да запамти податке о **клијенту** он приказује **психотерапеуту** поруку: "Klijent nije izmenjen!" (ИА)



Слика 29 - ДС Клијент није измењен

Са наведених дијаграма секвенци уочавају се системске операције које треба пројектовати:

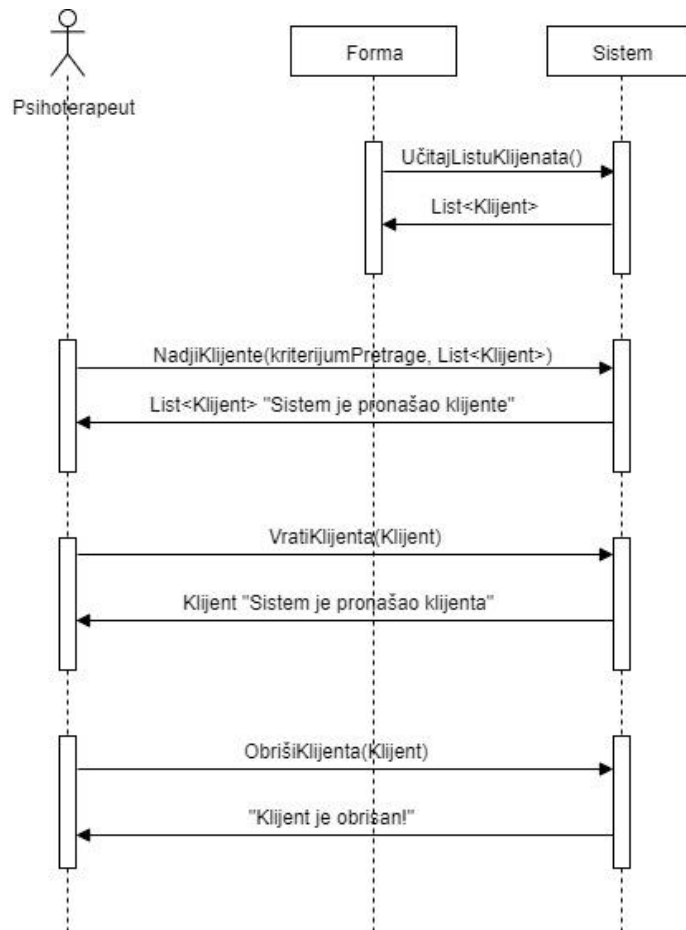
1. *signal* UčitajListuKlijenata(List<Klijent>)
2. *signal* UčitajGradove(List<Grad>)
3. *signal* NadjiKlijente(kriterijumPretrage, List<Klijent>)
4. *signal* VратиKlijenta(Klijent)
5. *signal* IzmeniKlijenta(Klijent)

ДС5: дијаграм секвенци случаја коришћења - Брисање клијента

Основни сценарио СК

1. Форма **позива** систем да учита листу клијената. (АПСО)
2. Систем **приказује** листу клијената. (ИА)
3. **Психотерапеут** **позива** систем да нађе клијенте по задатој вредности. (АПСО)
4. Систем **приказује** психотерапеуту листу клијената. (ИА)
5. **Психотерапеут** **позива** систем да учита податке о одабраном клијенту. (АПСО)

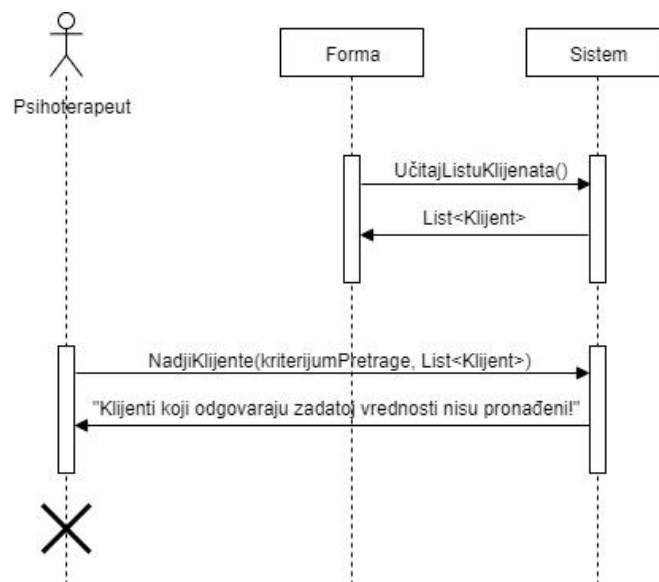
6. **Систем** приказује **психотерапеуту** податке о **клијенту**. (ИА)
7. **Психотерапеут** позива **систем** да обрише **клијента**. (АПСО)
8. **Систем** приказује **психотерапеуту** поруку: "Klijent obrisan!" (ИА)



Слика 30 - ДС Брисање клијента

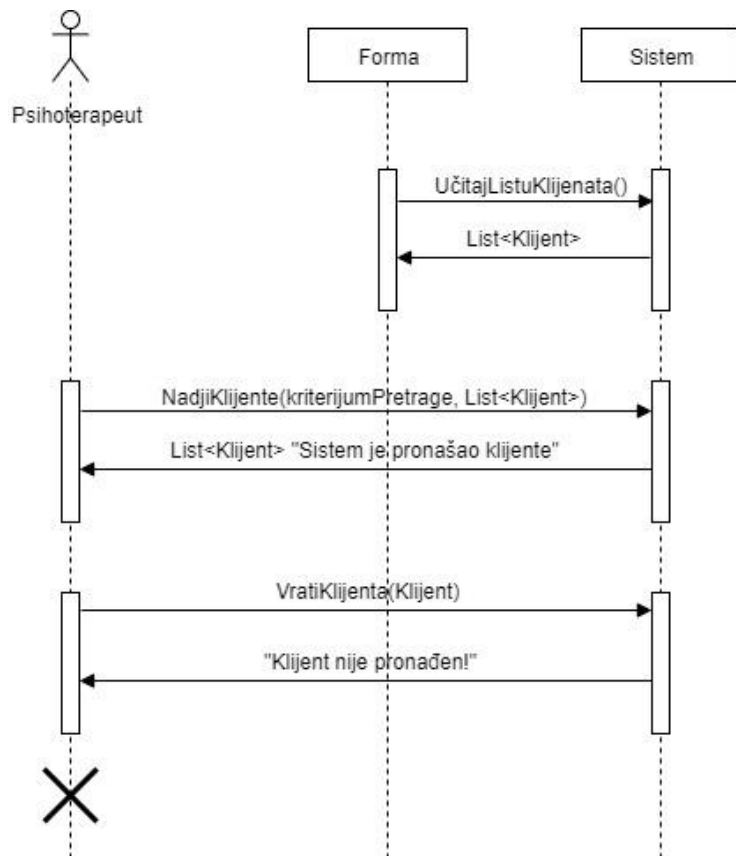
Алтернативна сценарија

4.1. Уколико **систем** не може да нађе **клијенте** он приказује **психотерапеуту** поруку: "Klijenti koji odgovaraju zadatoj vrednosti nisu pronađeni!" Прекида се извршење сценарија. (ИА)



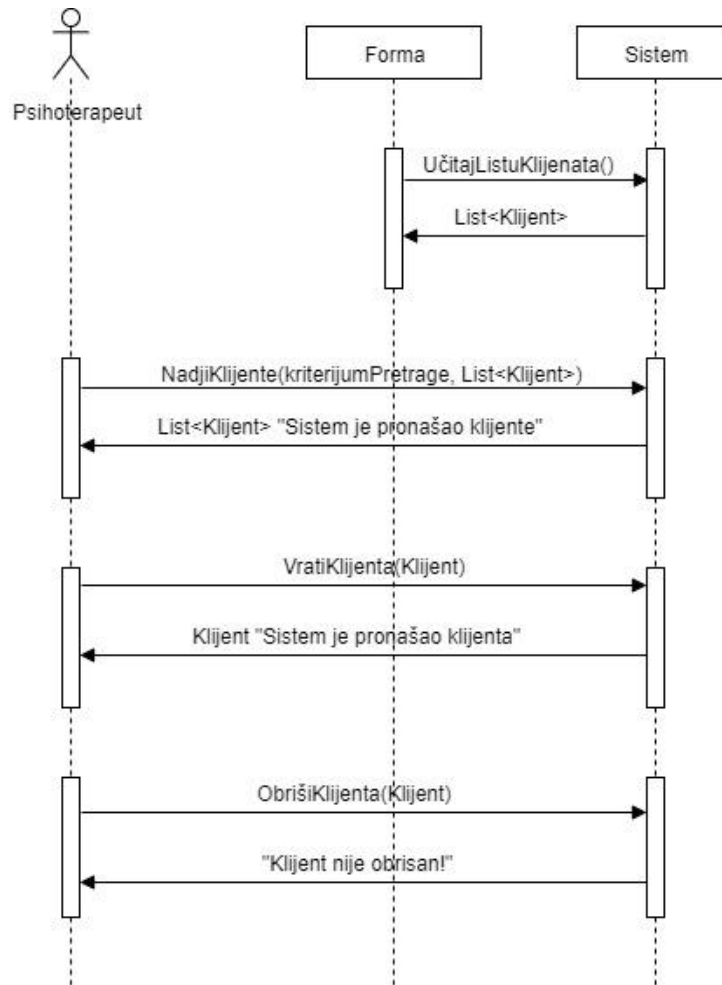
Слика 31 - ДС Клијенти који одговарају задатој вредности нису пронађени

6.1. Уколико **систем** не може да пронађе податке о **клијенту**, **систем** приказује **психотерапеуту** поруку: "Клијент није пронађен!" Прекида се извршење сценарија. (ИА)



Слика 32 - ДС Клијент није пронађен

- 8.1. Уколико **систем** не може да обрише **клијента** он приказује **психотерапеуту** поруку: "Klijent nije obrisan!" Прекида се извршење сценарија. (ИА)



Слика 33 - ДС Клијент није обрисан

Са наведених дијаграма секвенци уочавају се системске операције које треба пројектовати:

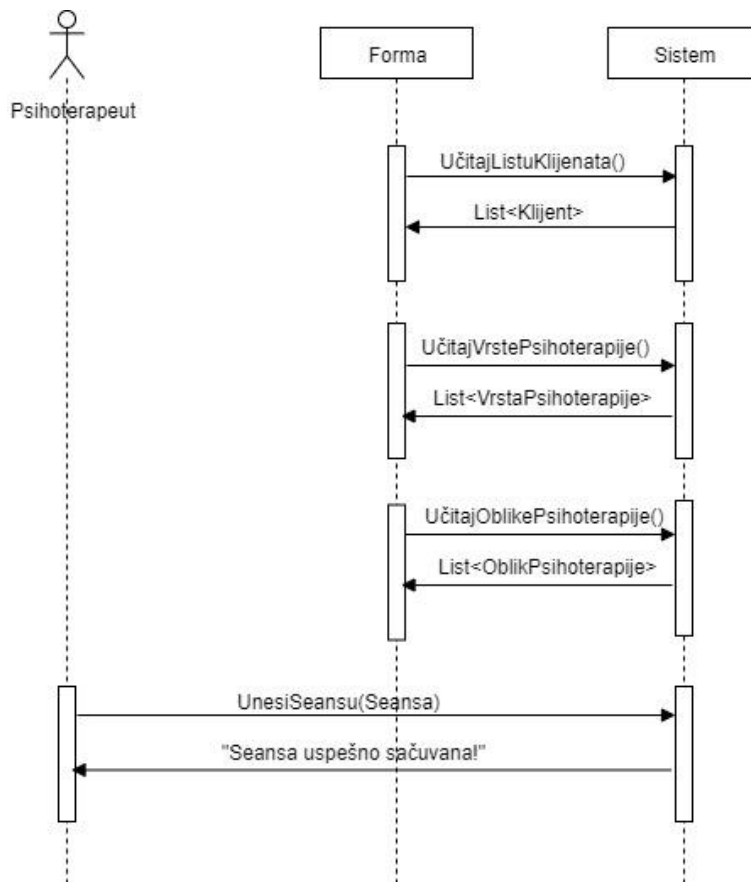
1. *signal* UčitajListuKlijenata(List<Klijent>)
2. *signal* NadjiKlijente(kriterijumPretrage, List<Klijent>)
3. *signal* VратиKlijenta(Klijent)
4. *signal* ObrišiKlijenta(Klijent)

ДС6: дијаграм секвенци случаја коришћења - Унос нове сеансе

Основни сценарио СК

1. Форма **позива** **систем** да учита листу клијената. (АПСО)
2. **Систем приказује** листу клијената. (ИА)

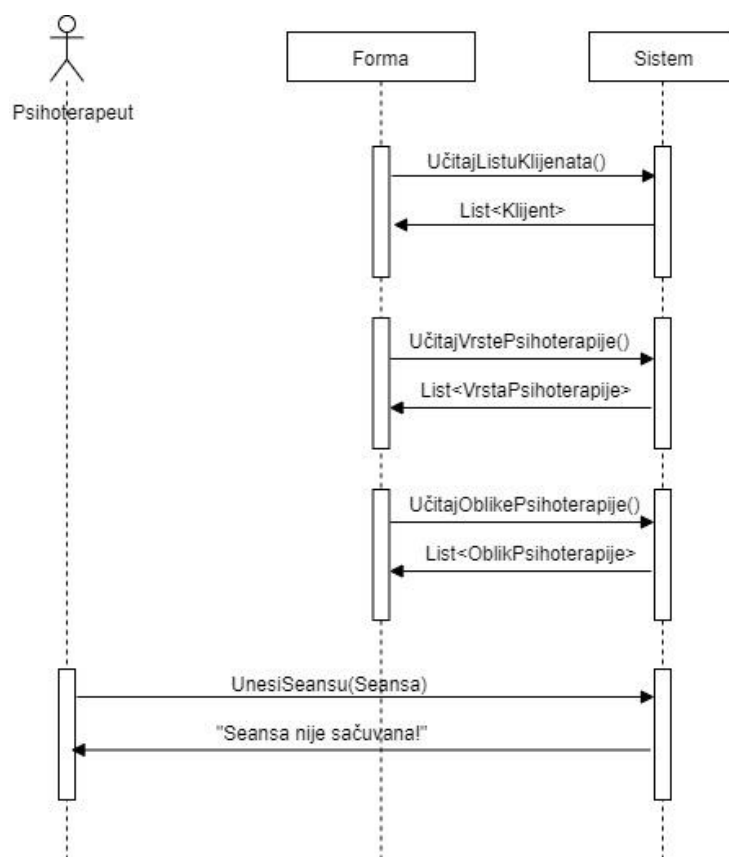
3. Форма **позива** **систем** да учита листу врста психотерапије. (АПСО)
4. **Систем приказује** листу врста психотерапије. (ИА)
5. Форма **позива** **систем** да учита листу облика психотерапије. (АПСО)
6. **Систем приказује** листу облика психотерапије. (ИА)
7. **Психотерапеут позива** **систем** да запамти податке о **сеанси**. (АПСО)
8. **Систем приказује психотерапеуту** поруку: "Seansa uspešno sačuvana!" (ИА)



Слика 34 - ДС Унос нове сеансе

Алтернативна сценарија

8.1. Уколико **систем** не може да запамти податке о **сеанси** он приказује **психотерапеуту** поруку: "Seansa nije sačuvana!" (ИА)



Слика 35 - ДС Сеанса није сачувана

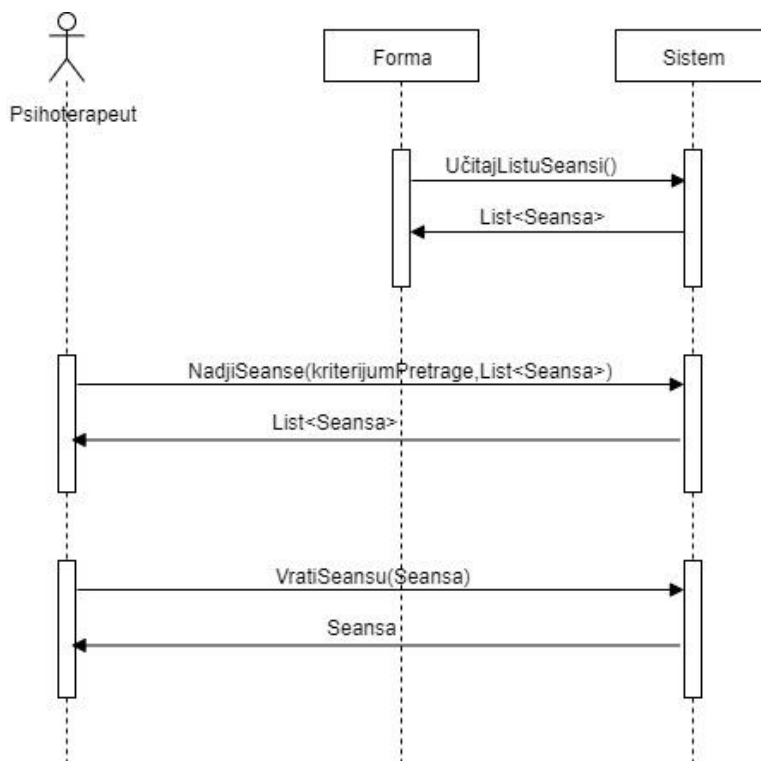
Са наведених дијаграма секвенци уочавају се системске операције које треба пројектовати:

1. *signal* UčitajListuKlijenata(List<Klijent>)
2. *signal* UčitajVrstePsihoterapije(List<VrstaPsihoterapije>)
3. *signal* UčitajOblikePsihoterapije(List<OblikPsihoterapije>)
4. *signal* UnesiSeansu(Seansa)

ДС7: дијаграм секвенци случаја коришћења - Претраживање сеанси

Основни сценарио СК

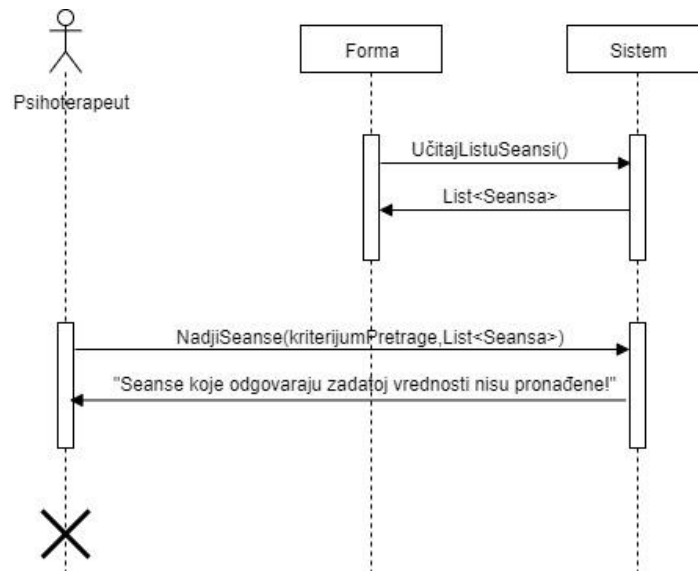
1. Форма **позива** систем да учита листу **сеанси**. (АПСО)
2. **Систем** **приказује** листу **сеанси**. (ИА)
3. **Психотерапеут** **позива** систем да нађе **сеансе** по задатој вредности. (АПСО)
4. **Систем** **приказује** **психотерапеуту** листу **сеанси**. (ИА)
5. **Психотерапеут** **позива** систем да учита податке о одабраној **сеанси**. (АПСО)
6. **Систем** **приказује** **психотерапеуту** податке о **сеанси**. (ИА)



Слика 36 - ДС Претраживање сеанси

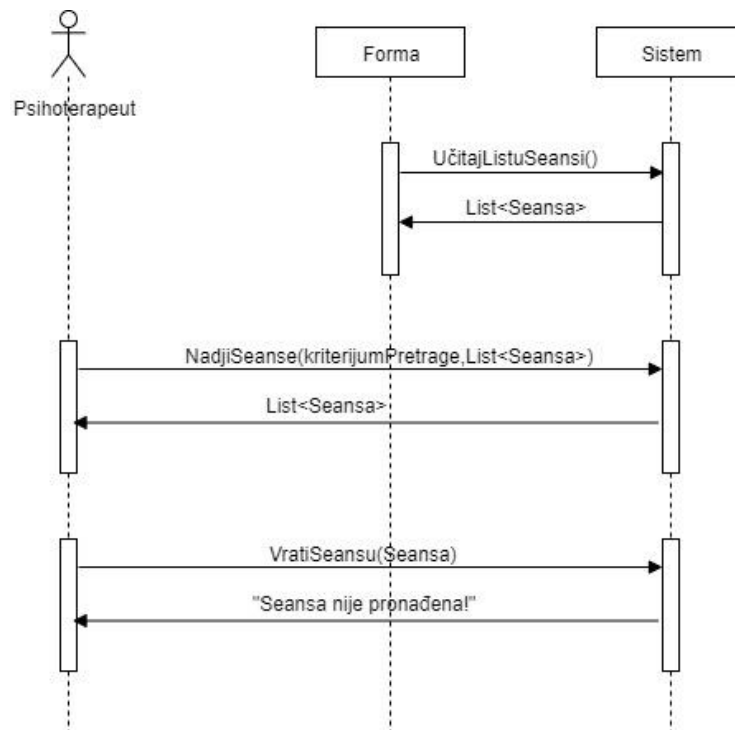
Алтернативна сценарија

- 4.1. Уколико **систем** не може да нађе **сеансе**, **систем** приказује **психотерапеуту** поруку: "Seanse koje odgovaraju zadatoj vrednosti nisu pronadene!" Прекида се извршење сценарија. (ИА)



Слика 37 - ДС Сеансе које одговарају задатој вредности нису пронађене

6.1. Уколико **систем** не може да пронађе **сеансу** он приказује **психотерапеуту** поруку: "Seansa nije pronađena!" (ИА)



Слика 38 - ДС Сеанса није пронађена

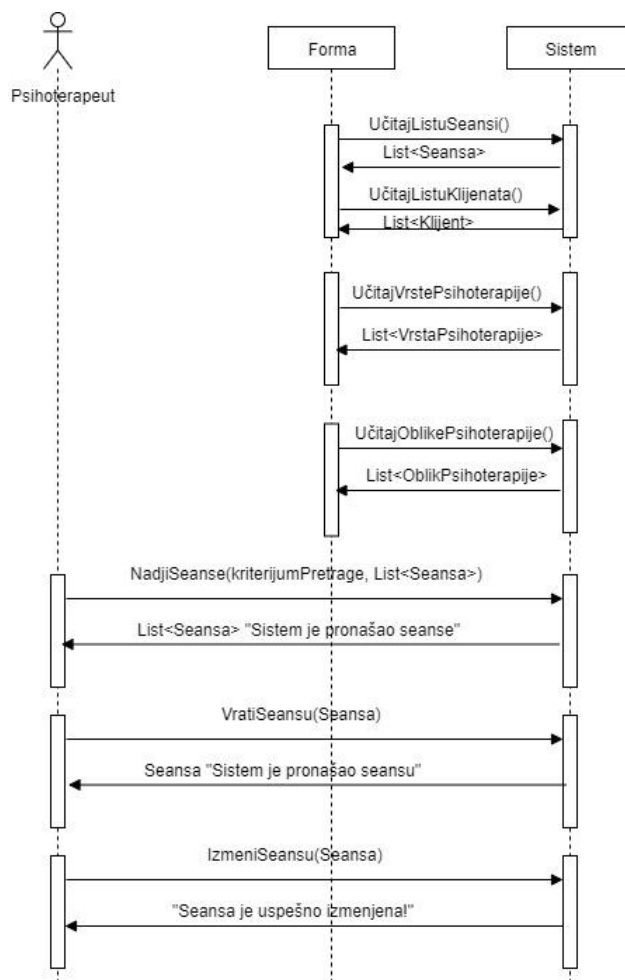
Са наведених дијаграма секвенци уочавају се системске операције које треба пројектовати:

1. *signal* / UčitajListuSeansi(List<Seansa>)
2. *signal* / NadjiSeanse(kriterijumPretrage, List<Seansa>)
3. *signal* / VратиSeansu(Seansa)

ДС8: дијаграм секвенци случаја коришћења - Измена садржаја сеансе

Основни сценарио СК

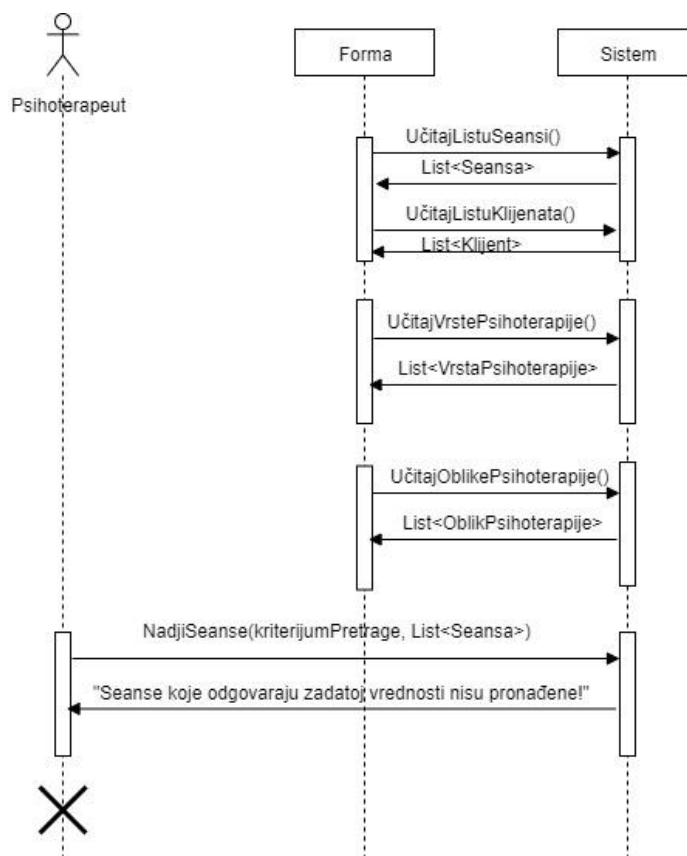
1. Форма **позива систем** да учита листу **сеанси**. (АПСО)
2. **Систем приказује** листу **сеанси**. (ИА)
3. Форма **позива систем** да учита врсте психотерапије. (АПСО)
4. **Систем приказује** врсте психотерапије. (ИА)
5. Форма **позива систем** да учита облике психотерапије. (АПСО)
6. **Систем приказује** облике психотерапије. (ИА)
7. Форма **позива систем** да учита листу клијената. (АПСО)
8. **Систем приказује** листу клијената. (ИА)
9. **Психотерапеут позива систем** да нађе **сеансе** по задатој вредности. (АПСО)
10. **Систем приказује психотерапеуту** листу **сеанси**. (ИА)
11. **Психотерапеут позива систем** да учита податке о одабраној **сеанси**. (АПСО)
12. **Систем приказује психотерапеуту** податке о **сеанси**. (ИА)
13. **Психотерапеут контролише** да ли је коректно унео податке о **сеанси**. (АНСО)
14. **Систем приказује психотерапеуту** поруку: "Seansa uspešno izmenjena!" (ИА)



Слика 39 - ДС Измена садржаја сеансе

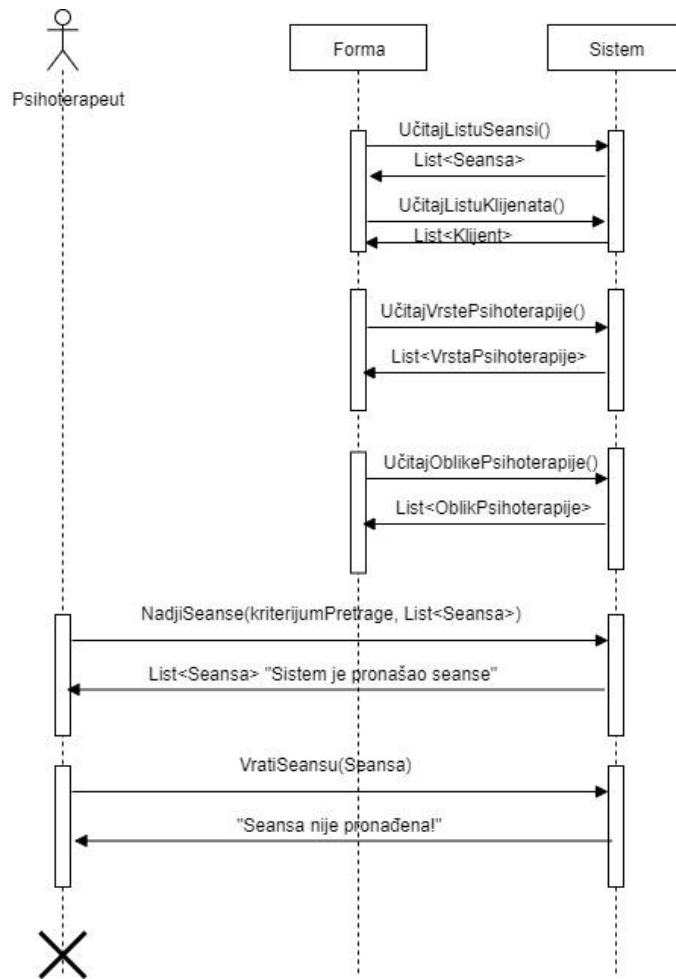
Алтернативна сценарија

10.1. Уколико **систем** не може да нађе **сеансе** он приказује **психотерапеуту** поруку: "Seanse koje odgovaraju zadatoj vrednosti nisu pronađene!". Прекида се извршење сценарија. (ИА)



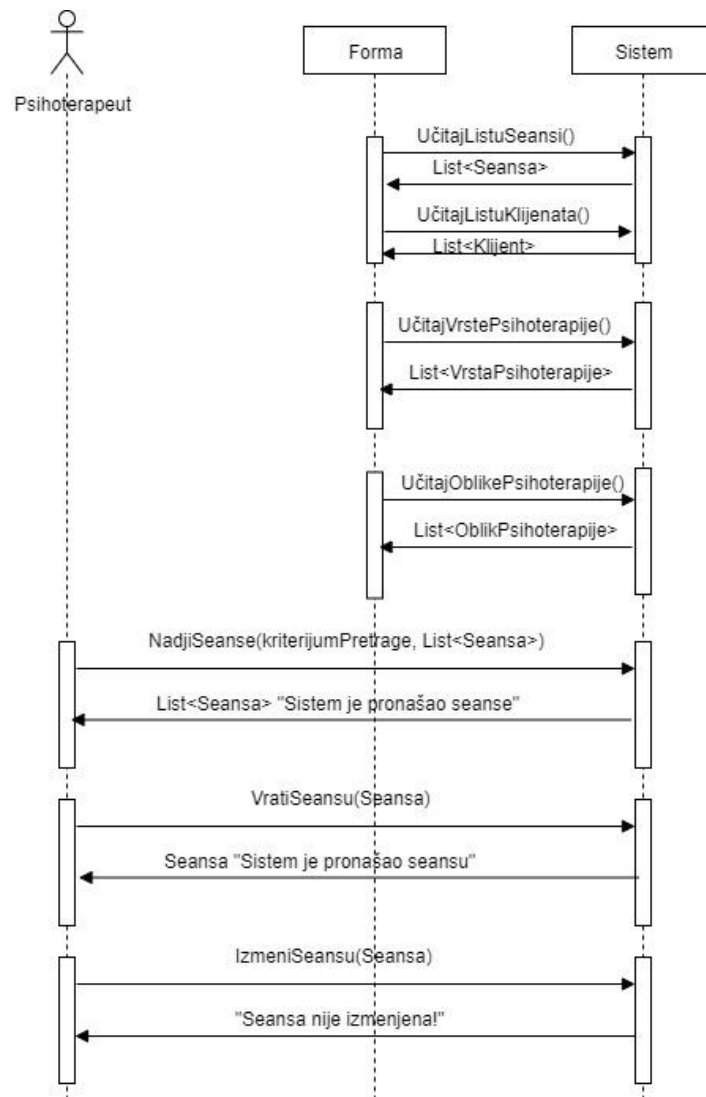
Слика 40 - ДС Сеансе које одговарају задатој вредности нису пронађене

12.1. Уколико **СИСТЕМ** не може да пронађе **сеансу** он приказује **психотерапеуту** поруку: "Seansa nije pronađena!" (ИА)



Слика 41 - ДС Сеанса није пронађена

- 14.1. Уколико **систем** не може да запамти податке о **сеанси** он приказује **психотерапеуту** поруку: "Seansa nije izmenjena!" (ИА)



Слика 42 - ДС Сеанса није измењена

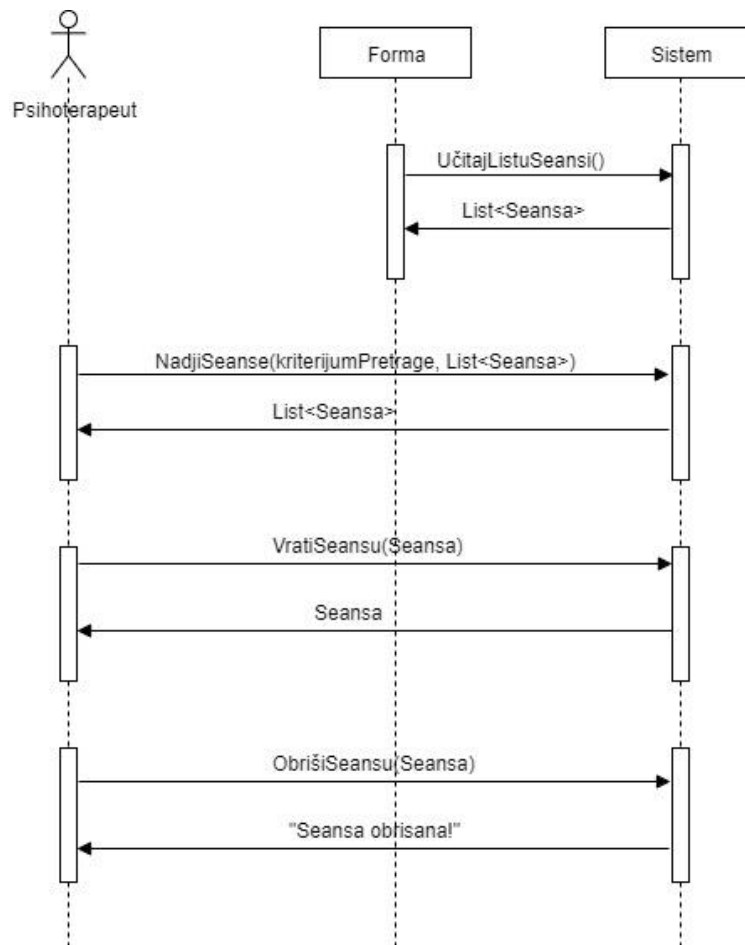
Са наведених дијаграма секвенци уочавају се системске операције које треба пројектовати:

1. *signal* UčitajListuSeansi(List<Seansa>)
2. *signal* UčitajVrstePsihoterapije(List<VrstaPsihoterapije>)
3. *signal* UčitajOblikePsihoterapije(List<OblikPsihoterapije>)
4. *signal* UčitajListuKlijenata(List<Klijent>)
5. *signal* NadjiSeanse(kriterijumPretrage, List<Seansa>)
6. *signal* VратиSeansu(Seansa)
7. *signal* IzmeniSeansu(Seansa)

ДС9: дијаграм секвенци случаја коришћења - Брисање сеансе

Основни сценарио СК

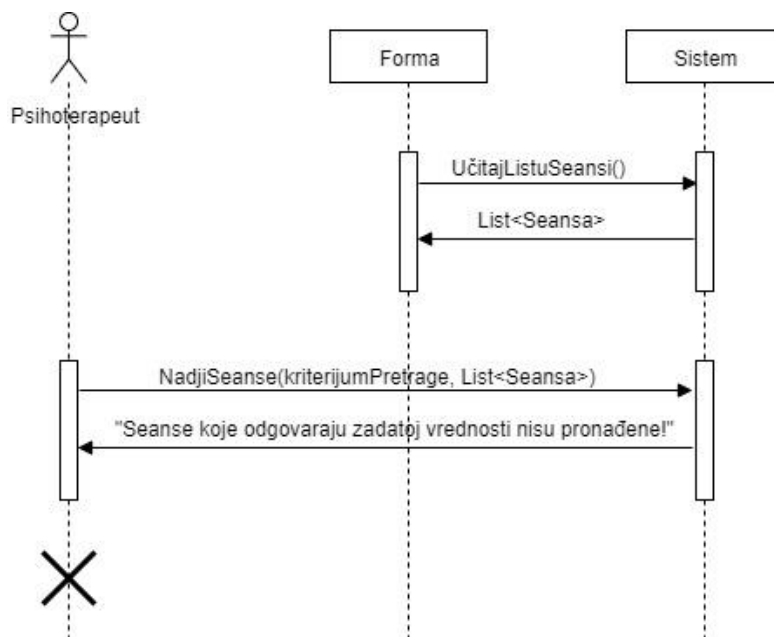
1. Форма **позива** **систем** да учита листу **сеанси**. (АПСО)
2. **Систем** **приказује** листу **сеанси**. (ИА)
3. **Психотерапеут** **позива** **систем** да нађе **сеансе** по задатој вредности. (АПСО)
4. **Систем** **приказује** **психотерапеуту** листу **сеанси**. (ИА)
5. **Психотерапеут** **позива** **систем** да учита податке о одабраној **сеанси**. (АПСО)
6. **Систем** **приказује** **психотерапеуту** податке о **сеанси**. (ИА)
7. **Психотерапеут** **позива** **систем** да обрише **сеансу**. (АПСО)
8. **Систем** **приказује** **психотерапеуту** поруку: "Seansa obrisana!" (ИА)



Слика 43 - ДС Брисање сеансе

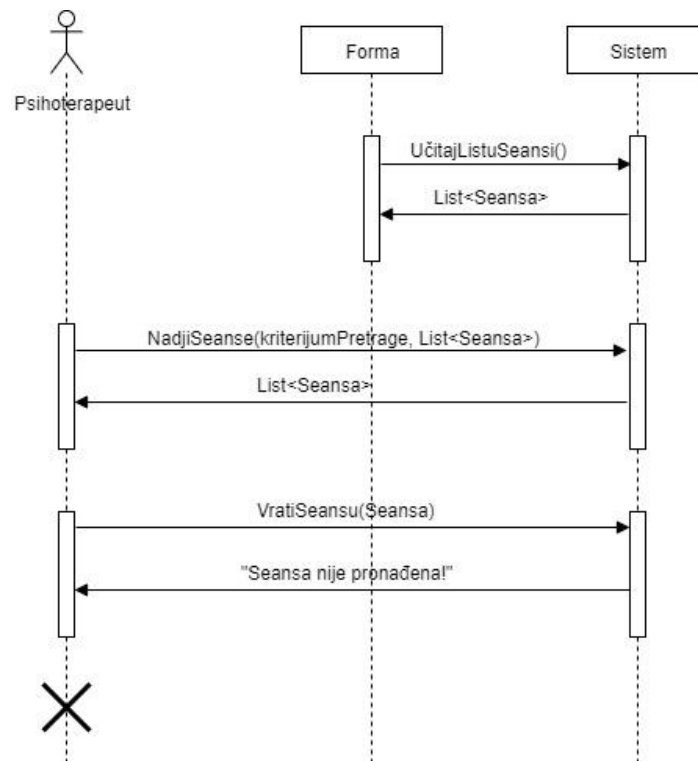
Алтернативна сценарија

4.1. Уколико **систем** не може да нађе **сеансе** он приказује **психотерапеуту** поруку: "Seanse koje odgovaraju zadatoj vrednosti nisu pronađene!" Прекида се извршење сценарија. (ИА)



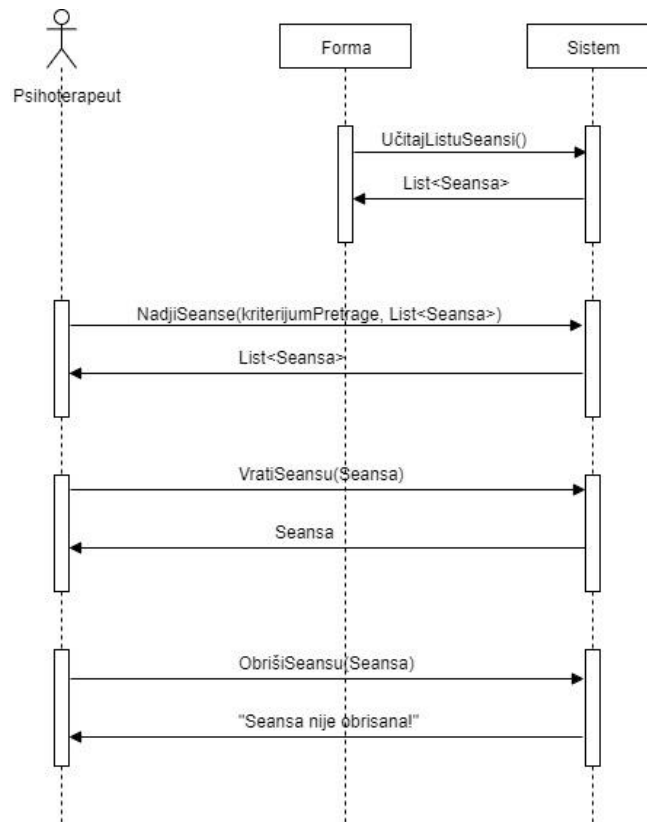
Слика 44 - ДС Сеансе које одговарају задатој вредности нису пронађене

6.1. Уколико **систем** не може да пронађе **сеансу** он приказује **психотерапеуту** поруку: "Seansa nije pronađena!" (ИА)



Слика 45 - ДС Сеанса није пронађена

- 8.1. Уколико **систем** не може да обрише **сеансу** он приказује **психотерапеуту** поруку: "Seansa nije obrisana!". (ИА)



Слика 46 - ДС Сеанса није обрисана

Са наведених дијаграма секвенци уочавају се системске операције које треба пројектовати:

1. *signal* UčitajListuSeansi(List<Seansa>)
2. *signal* NadjiSeanse(kriterijumPretrage, List<Seansa>)
3. *signal* VратиSeansu(Seansa)
4. *signal* ОбришиSeansu(Seansa)

Списак системских операција

Као резултат анализе сценарија добијено је укупно 15 системских операција које треба пројектовати:

1. *signal* **PrijavaPsihoterapeuta**(Psihoterapeut)
2. *signal* **UnesiKlijenta**(Klijent)
3. *signal* **UčitajListuKlijenata**(List<Klijent>)
4. *signal* **UčitajGradove**(List<Grad>)
5. *signal* **NadjiKlijente**(kriterijumPretrage, List<Klijent>)
6. *signal* **VratiKlijenta**(Klijent)
7. *signal* **IzmeniKlijenta**(Klijent)
8. *signal* **ObrišiKlijenta**(Klijent)
9. *signal* **UčitajOblikePsihoterapije**(List<OblikPsihoterapije>)
10. *signal* **UnesiSeansu**(Seansa)
11. *signal* **UčitajListuSeansi**(List<Seansa>)
12. *signal* **NadjiSeanse**(kriterijumPretrage, List<Seansa>)
13. *signal* **VratiSeansu**(Seansa)
14. *signal* **IzmeniSeansu**(Seansa)
15. *signal* **ObrišiSeansu**(Seansa)

8.2. Понашање софтверског система: Дефинисање уговора о системским операцијама

Системска операција представља понашање софтверског система. За сваку од уочених системских операција се праве *уговори*, који описују понашање саме операције, односно оно што она треба да одради, али не описују начин на који ће се то одрадити. Један уговор се везује за једну системску операцију.

УГ1: PrijavaPsihoterapeuta(Psihoterapeut)

Операција: PrijavaPsihoterapeuta(Psihoterapeut): signal

Веза са СК: СК1

Предуслови: /

Постуслови: /

УГ2: UnesiKlijenta(Klijent)

Операција: UnesiKlijenta(Klijent): signal

Веза са СК: СК2

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

Постуслови: Убачен је нови клијент.

УГ3: UčitajListuKlijenata()

Операција: UčitajListuKlijenata(): signal

Веза са СК: СК3, СК4, СК5, СК6, СК8

Предуслови: /

Постуслови: /

УГ4: UčitajGradove()

Операција: UčitajGradove(): signal

Веза са СК: СК2, СК4

Предуслови: /

Постуслови: /

УГ5: NadjiKlijente(kriterijumPretrage, List<Klijent>)

Операција: NadjiKlijente(kriterijumPretrage, List<Klijent>): signal

Веза са СК: СК3, СК4, СК5

Предуслови: /

Постуслови: /

УГ6: VратиKlijenta(Klijent)

Операција: VратиKlijenta(Klijent): signal

Веза са СК: СК3, СК4, СК5

Предуслови: /

Постуслови: /

УГ7: IzmeniKlijenta(Klijent)

Операција: IzmeniKlijenta(Klijent): signal

Веза са СК: СК4

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

Постуслови: Унети подаци о клијенту су измењени.

УГ8: ObrišiKlijenta(Klijent)

Операција: ObrišiKlijenta(Klijent): signal

Веза са СК: СК5

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

Постуслови: Клијент је обрисан.

УГ9: UčitajOblikePsihoterapije()

Операција: UčitajOblikePsihoterapije(): signal

Веза са СК: СК6,СК8

Предуслови: /

Постуслови: /

УГ10: UnesiSeansu(Seansa)

Операција: UnesiSeansu(Seansa)

Веза са СК: СК6

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

Постуслови: Евидентирана је нова сеанса.

УГ11: UčitajListuSeansi()

Операција: UčitajListuSeansi(): signal

Веза са СК: СК7, СК8, СК9

Предуслови: /

Постуслови: /

УГ12: NadjiSeanse(kriterijumPretrage, List<Seansa>)

Операција: NadjiSeanse(kriterijumPretrage, List<Seansa>): signal

Веза са СК: СК7, СК8, СК9

Предуслови: /

Постуслови: /

УГ13: VратиSeansu(Seansa)

Операција: VратиSeansu(Seansa): signal

Веза са СК: СК7, СК8, СК9

Предуслови: /

Постуслови: /

УГ14: IzmeniSeansu(Seansa)

Операција: IzmeniSeansu(Seansa): signal

Веза са СК: СК8

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

Постуслови: Унети подаци о сеанси су измењени.

УГ15: ObrišiSeansu(Seansa)

Операција: ObrišiSeansu(Seansa): signal

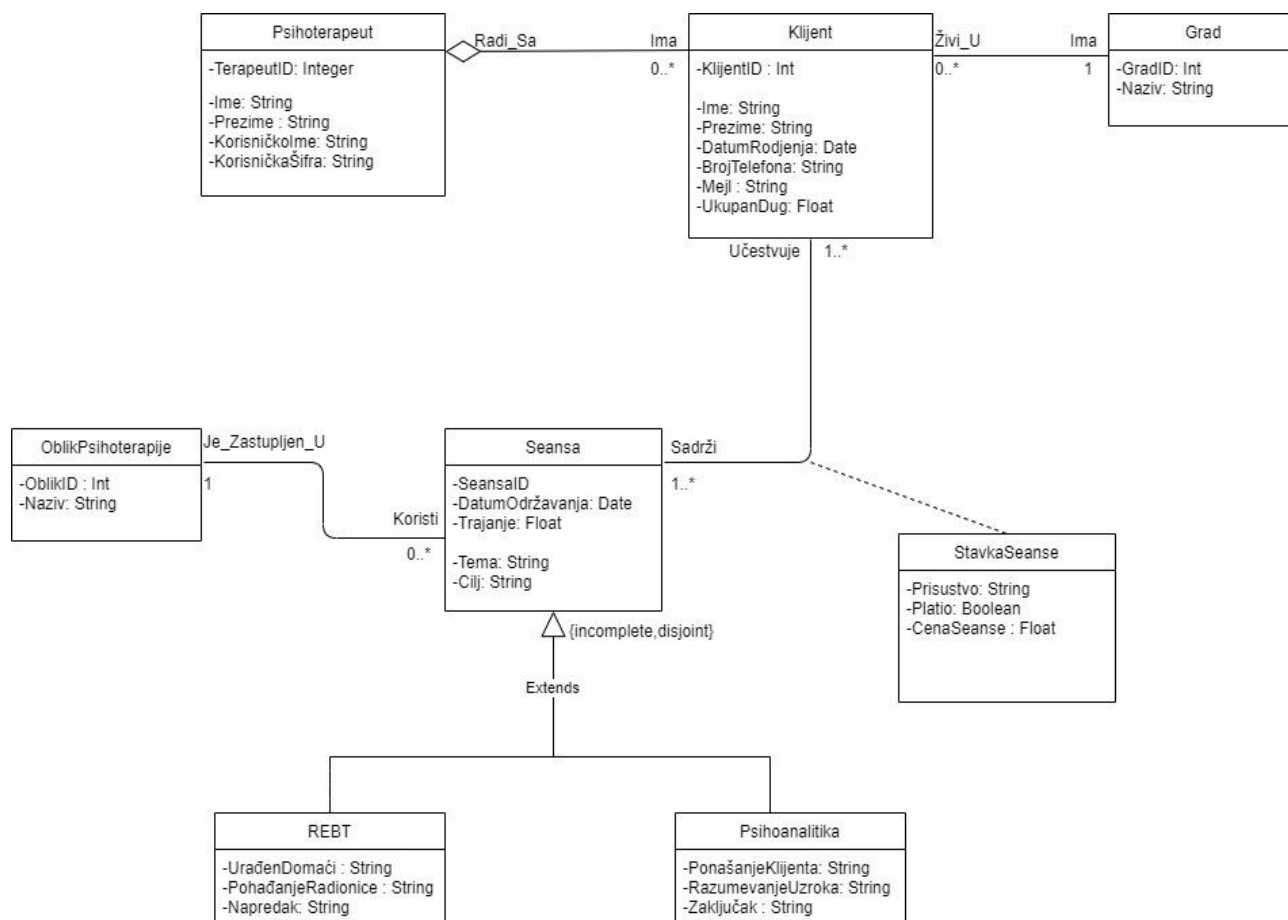
Веза са СК: СК5

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

Постуслови: Сеанса је обрисана.

8.3. Структура софтверског система: концептуални (доменски) модел

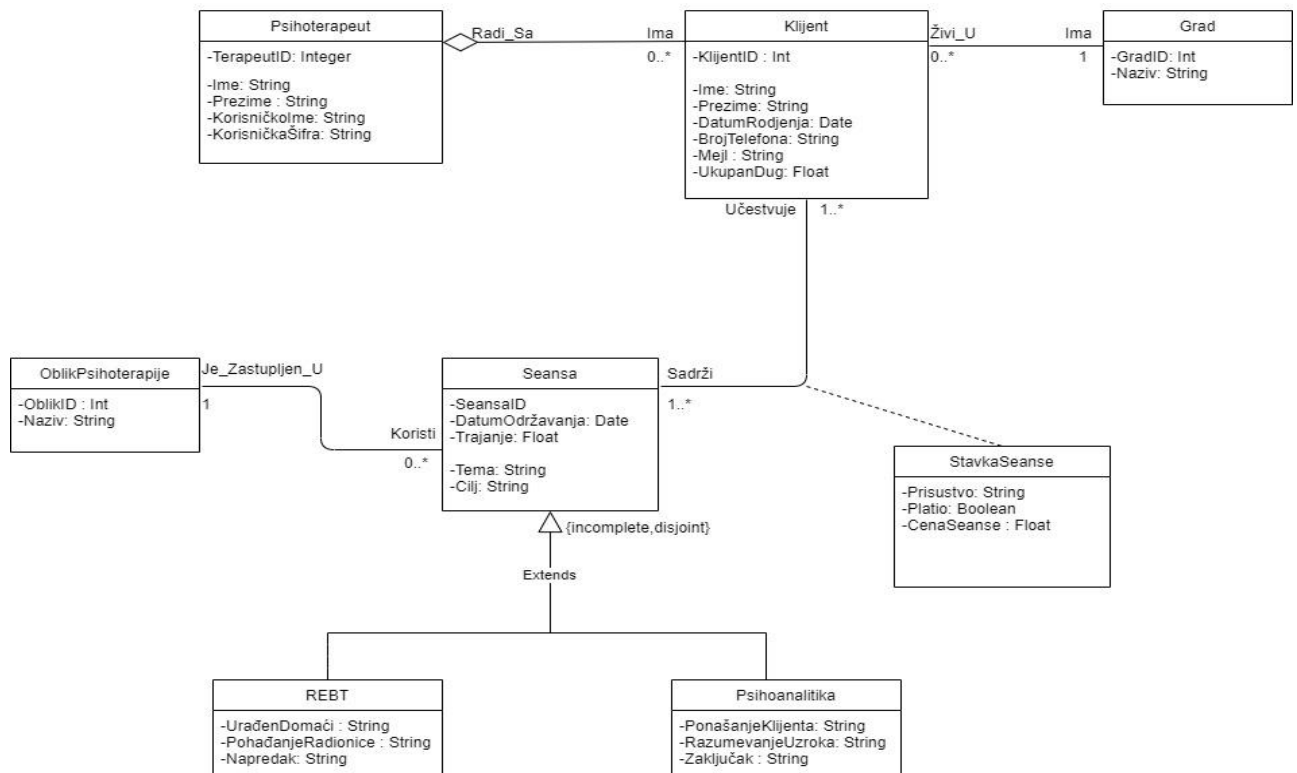
Структура софтверског система се описује помоћу *концептуалног модела*. Концептуални модел садржи концептуалне класе (доменске објекте) и асоцијације између концептуалних класа. Често се још и називају *доменским моделима* или *моделима објектне анализе*.



Слика 47 - Концептуални дијаграм класа (Логичка структура софтверског система)

Као резултат анализе сценарија случајева коришћења и прављења концептуалног модела добија се *логичка структура и понашање система*.

Struktura sistema



Ponašanje sistema



Слика 48 - Структура и понашање софтверског система

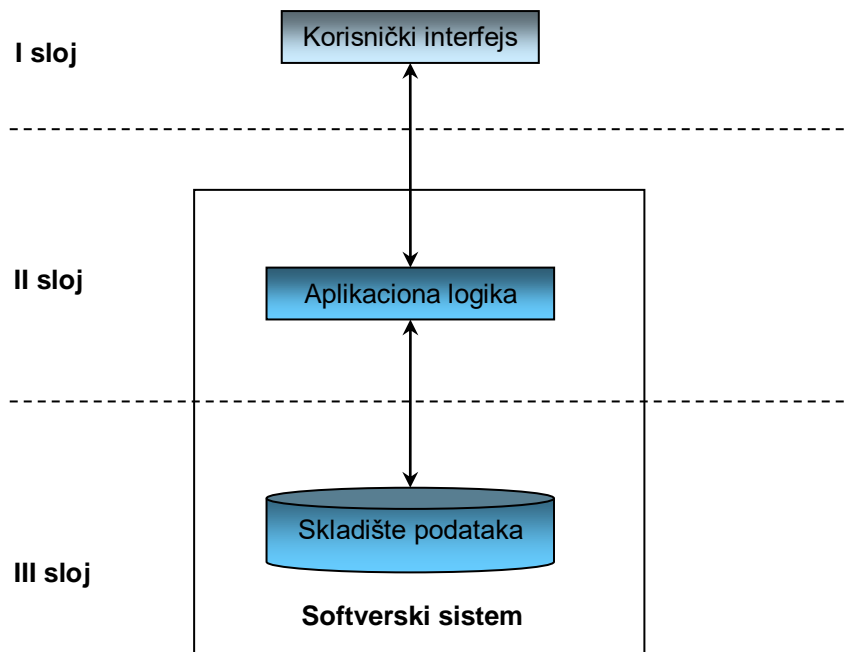
9. Фаза пројектовања

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

9.1. Архитектура софтверског система

Најчешће коришћена архитектура је тронивовска архитектура која се састоји из следећих нивоа:

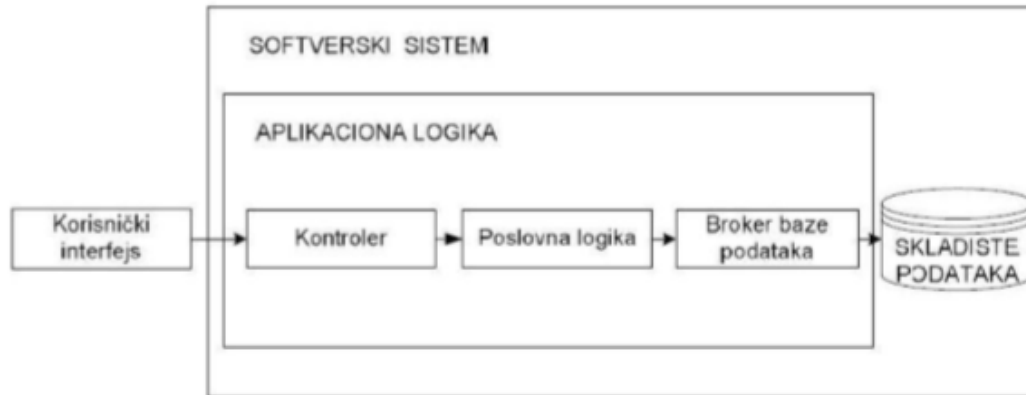
1. **корисничког интерфејса** који представља улазно-излазну репрезентацију софтверског система,
2. **апликационе логике** која описује структуру и понашање софтверског система и
3. **складишта података** које чува стање атрибута софтверског система.



Слика 49 - Тронивовска архитектура софтверског система

Пројектовање архитектуре софтверског система обухвата пројектовање наведена три слоја. Ниво корисничког интерфејса смештен је на страни клијента, док се апликациона логика и складиште података налазе на страни сервера.

Слој апликационе логике садржи три компоненте — део за комуникацију са клијентом (контролер), део за комуникацију са складиштем података (брокер базе података) и део који садржи пословну логику.



Слика 50 - Софтверски систем

До почетка фазе пројектовања, већ је дефинисана пословна логика софтверског система, односно његова логичка структура и понашање. У наставку ћемо пројектовати сваки од наведених елемената тринивојске архитектуре.

9.2. Пројектовање складишта података - Релациони модел

На основу концептуалног модела прави се *релациони модел*, који ће нам служити као основа за пројектовање релационе базе података.

Идентификовали смо следеће класе у концептуалном моделу: **Klijent**, **Psihoterapeut**, **Grad**, **OblikPsihoterapije**, **Seansa**, **REBT**, **Psihoanaliza**, **StavkaSeanse**. Њих ћемо потом представити као табеле у релационом моделу.

Psihoterapeut (TerapeutID, Ime, Prezime, KorisničkoIme, KorisničkaŠifra)

Grad (GradID, Naziv)

Klijent (KlijentID, Ime, Prezime, DatumRodjenja, BrojTelefona, Mejl, UkupniDug, *GradID*, *TerapeutID*)

OblikPsihoterapije (OblikID, Naziv)

Seansa (SeansaID, DatumOdržavanja, Trajanje, Tema, Cilj, *OblikID*)

REBT (SeansaID, UrađenDomaći, PohađanjeRadionice, Napredak)

Psihoanaliza (SeansaID, PonašanjeKlijenta, RazumevanjeUzroka, Zaključak)

StavkaSeanse (KlijentID, SeansaID, Rbr, Prisustvo, Platio, CenaSeanse)

Tabela Psihoterapeut		Prosto vrednosno ograničenje		Složeno vrednosno ograničenje		Strukturno ograničenje
Atributi	Ime	Tip atributa	Vrednost atributa	Međuzavisnost atributa jedne tabele	Međuzavisnost atributa više tabela	<ul style="list-style-type: none"> • Insert / Update Cascades Klijent • Delete Restricted Klijent
	TerapeutID	Integer	not null			
	Ime	String	not null			
	Prezime	String	not null			
	KorisničkoIme	String	not null			
	KorisničkaŠifra	String	not null			

Табела 1 - Психотерапеут

Tabela Grad		Prosto vrednosno ograničenje		Složeno vrednosno ograničenje		Strukturno ograničenje
Atributi	Ime	Tip atributa	Vrednost atributa	Međuzavisnost atributa jedne tabele	Međuzavisnost atributa više tabela	<ul style="list-style-type: none"> • Insert / Update Cascades Klijent • Delete Restricted Klijent
	GradID	Integer	not null			
	NazivG	String	not null			

Табела 3 - Град

Tabela Klijent		Prosto vrednosno ograničenje		Složeno vrednosno ograničenje		Strukturno ograničenje
Atributi	Ime	Tip atributa	Vrednost atributa	Međuzavisnost atributa jedne tabele	Međuzavisnost atributa više tabela	<ul style="list-style-type: none"> • Insert Restricted Psihoterapeut, Grad • Update Cascades StavkaSeanse Restricted Psihoterapeut, Grad • Delete Restricted StavkaSeanse
	KlijentID	Integer	not null			
	Ime	String	not null			
	Prezime	String	not null			
	DatumRodjenja	Date	not null			
	BrojTelefona	String	not null			
	Mejl	String	not null			
	UkupniDug	Double	not null		UkupniDug= SUM (StavkaSeanse.CenaSeanse)	
	GradID	Integer	Not null			
	TerapeutID	Integer	Not null			

Табела 2 - Клијент

Tabela OblikPsihoterapije		Prosto vrednosno ograničenje		Složeno vrednosno ograničenje		Strukturno ograničenje
Atributi	Ime	Tip atributa	Vrednost atributa	Medjuzavisnost atributa jedne tabele	Međuzavisnost atributa vise tabela	<ul style="list-style-type: none"> • Insert / • Update Cascade Seansa • Delete Restricted Seansa
	OblikID	Integer	not null			
	Naziv	String	Not null			

Табела 4 - Табела ОбликПсихотерапије

Tabela REBT		Prosto vrednosno ograničenje		Složeno vrednosno ograničenje		Strukturno ograničenje
Atributi	Ime	Tip atributa	Vrednost atributa	Medjazavisnost atributa jedne tabele	Medjazavisnost atributa vise tabela	<ul style="list-style-type: none"> • Insert Restricted Seansa • Update Restricted Seansa • Delete /
	SeansaID	Integer	not null			
	UrađenDomaći	String	Not null			
	PohađanjeRadionice	String	Not null			
	Napredak	String	Not null			

Табела 5 - Табела РЕБТ

Tabela Psihoanalitika		Prosto vrednosno ograničenje		Složeno vrednosno ograničenje		Strukturno ograničenje
Atributi	Ime	Tip atributa	Vrednost atributa	Medjazavisnost atributa jedne tabele	Medjazavisnost atributa vise tabela	<ul style="list-style-type: none"> • Insert Restricted Seansa • Update Restricted Seansa • Delete /
	SeansaID	Integer	not null			
	PonašanjeKlijenta	String	Not null			
	Razumevanje Uzroka	String	Not null			
	Zaključak	String	Not null			

Табела 6 - Табела Психоаналитика

Tabela Seansa		Prosto vrednosno ograničenje		Složeno vrednosno ograničenje		Strukturno ograničenje
Atributi	Ime	Tip atributa	Vrednost atributa	Međuzavisnost atributa jedne tabele	Međuzavisnost atributa više tabela	<ul style="list-style-type: none"> Insert Restricted OblikPsihoterapije Update Cascades StavkaSeanse Restricted OblikPsihoterapije Cascades REBT, Psihoanalitika Delete Cascades StavkaSeanse Cascades REBT, Psihoanalitika
	SeansaID	Integer	not null			
	Trajanje	Double	not null			
	DatumOdržavanja	Date	not null			
	Tema	String	not null			
	Cilj	String	not null			
	Uvid	String	not null			
	OblikID	Integer	not null			

Табела 7 - Табела Сеанса

Tabela StavkaSeanse		Prosto vrednosno ograničenje		Složeno vrednosno ograničenje		Strukturno ograničenje
Atributi	Ime	Tip Atributa	Vrednost atributa	Međuzavisnos t atributa jedne tabele	Međuzavisnost atributa više tabela	<ul style="list-style-type: none"> Insert Restricted Seansa, Klijent Update Restricted Seansa, Klijent Delete /
	SeansaID	Integer	not null			
	KlijentID	Integer	not null			
	Rbr	Integer	Not null			
	Prisustvo	String	Not null			
	Platio	Boolean	Not null			
	CenaSeanse	Double	Not null			

Табела 8 - Табела СтавкаСеансе

9.3. Пројектовање апликационе логике

Апликациона логика служи за описивање структуре и понашања софтверског система и пројектује се независно од корисничког интерфејса и обрнуто. Другим речима, апликациона логика (која представља *Модел у MVC патерну*) нема знања о томе где се налази кориснички интерфејс (који представља *View у MVC патерну*).

Контролер је одговоран да прихвати захтев за извршење системске операције од клијента и да га проследи до пословне логике која је одговорна за извршење системске операције. Апликациони сервери треба да обезбеде сервисе који ће омогућити реализацију апликационе логике софтверског система.

Пројектовани апликациони сервер садржи:

- део за комуникацију са клијентима
- контролер апликационе логике
- део за комуникацију са складиштем података (брокер базе података)
- део који садржи пословну логику

Комуникација са клијентима

Део за комуникацију подиже серверски сокет који ће да ослушкује мрежу. Када клијент (клијентски сокет) успостави конекцију са контролером (серверским сокетом), тада контролер генерише нит која ће бити одговорна за двосмерну везу са клијентом. Слање и пријем података од клијента се остварује преко сокета.

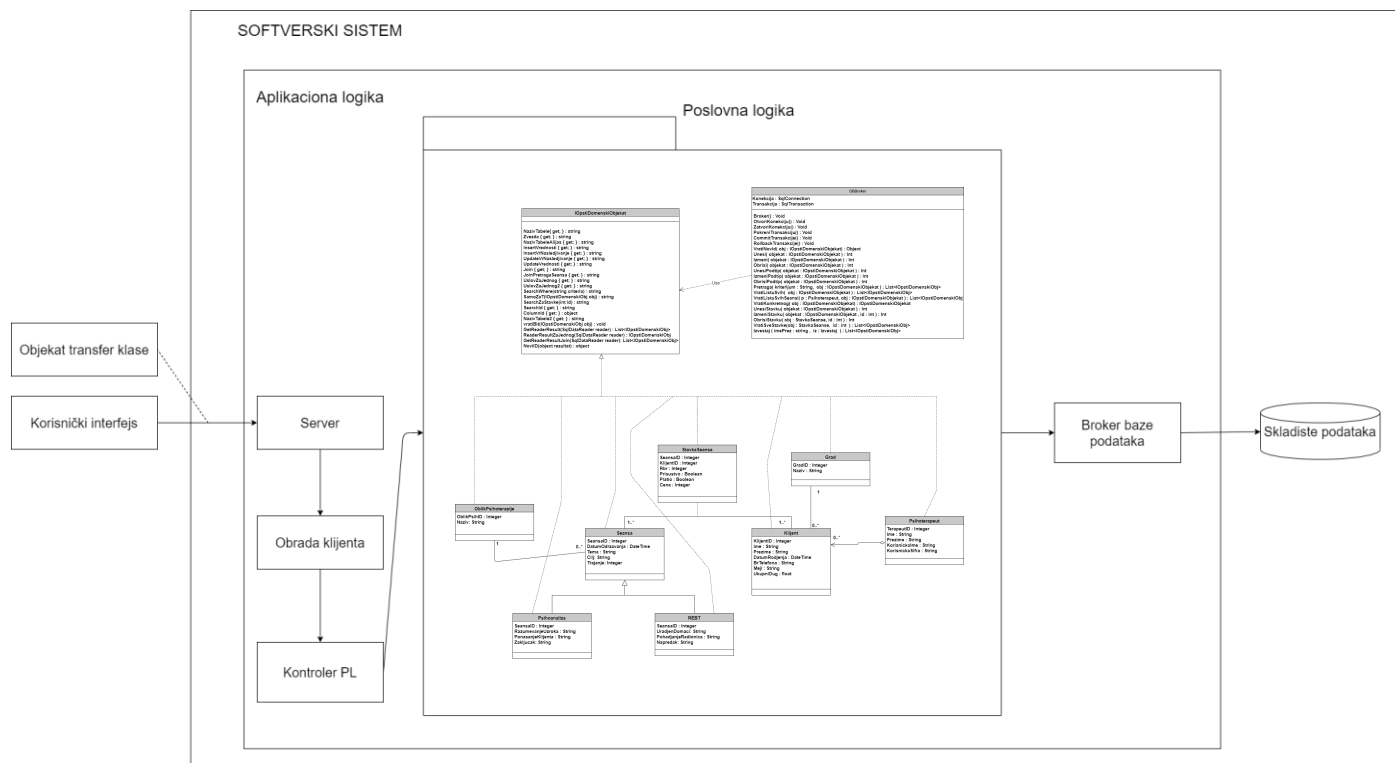
Клијент шаље захтев за извршење неке од системских операција, док одговарајућа нит (додељена клијенту) прихвата захтев и прослеђује га до контролера апликационе логике. Након извршења системске операције, контролер враћа резултат извршења операције до нити која је задужена за тог клијента. Резултат се затим прослеђује клијенту.

Комуникацију између клијента и сервера обавља се разменом трансфер објекта.

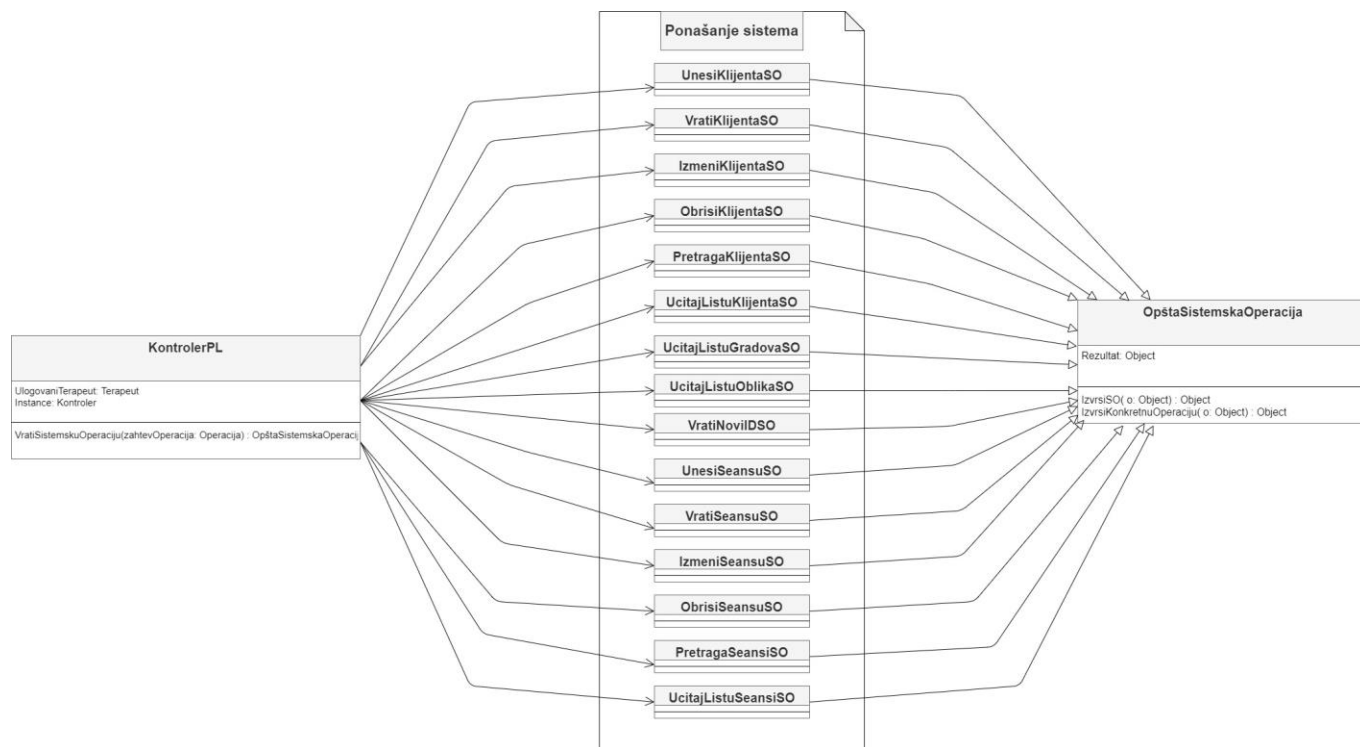
Контролер апликационе логике

Контролер апликационе логике прихвата захтеве за извршење системских операција и исте прослеђује до конкретне системске операције. Након извршења системске операције, контролер прихвата одговор и враћа назад позиваоцу (нити клијента).

Како је у фазама прикупљања захтева и анализе дата спецификација структуре и понашања софтверског система, односно спецификација пословне логике софтверског система, следећа слика даје опис система након фазе пројектовања комуникације са клијентом и контролера апликационе логике.



Слика 51 - Архитектура софтверског система након пројектовања комуникације са клијентом и контролерот апликационе логике



Слика 52 - Повезаност контролера (KontrolerPL) са општом систем операцијом (OpstaSistemskaOperacija)

Пословна логика

Пословна логика : Доменске класе

На основу концептуалних класа праве се софтверске класе структуре. Свака класа садржи приватна поља атрибута, модификаторе приступа за те атрибуте, конструкторе (непараметарске и параметарске).

Доменске класе имплементирају интерфејс *IOpstiDomenskiObjekat*, како би се омогућило лакше имплементирање метода брокера базе података. Брокер базе података прима интерфејс уместо самих класа и олакшава креирање генеричких упита.

Пословна логика : Пројектовање понашања софтверског система – системске операције

Препорука је да се на самом почетку пројектовања системских операције направе концептуалне реализације (решења) за сваку системску операцију тако да она буде директно повезана са логиком проблема. Аспекти реализације који се односе на конекцију са базом, перзистентност и трансакције треба избећи како би се логика решавања проблема независно развијала.

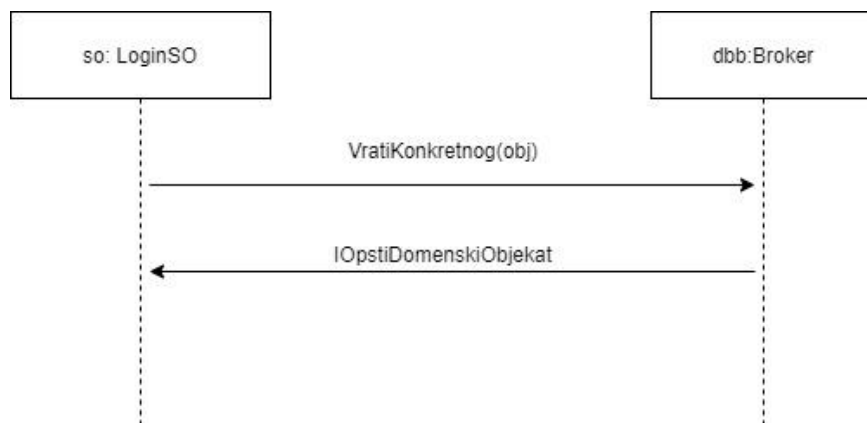
УГ1: PrijavaPsihoterapeuta (Psihoterapeut)

Операција: PrijavaPsihoterapeuta(Psihoterapeut): signal

Веза са СК: СК1

Предуслови: /

Постуслови: /



Слика 53 - Дијаграм секвенци "LoginSO"

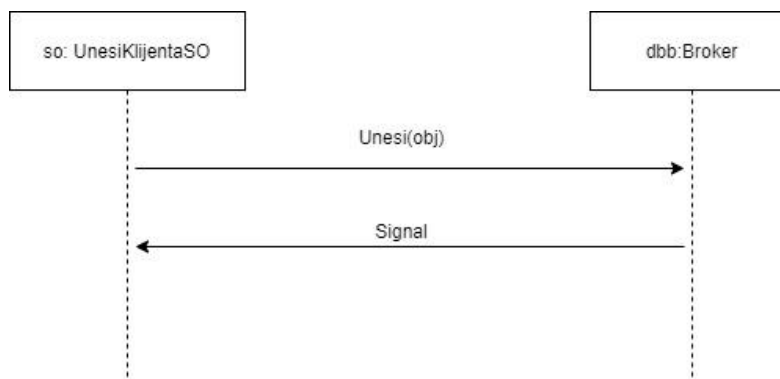
УГ3: UnesiKlijenta (Klijent)

Операција: UnesiKlijenta(Klijent): signal

Веза са СК: СК2

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

Постуслови: Убачен је нови клијент.



Слика 54 - Дијаграм секвенци "UnesiKlijentaSO"

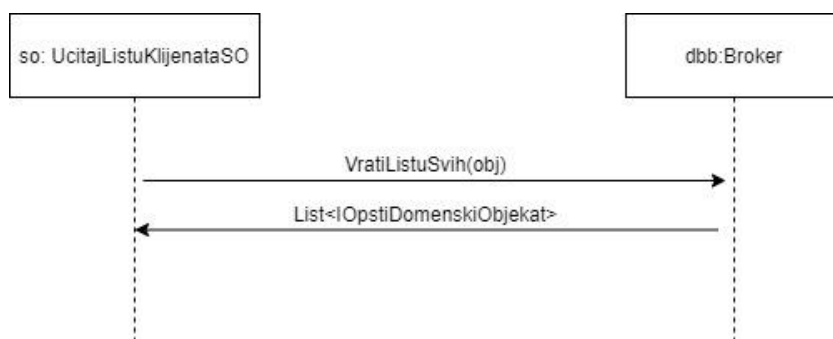
УГ4: UčitajListuKlijenata()

Операција: UčitajListuKlijenata(): signal

Веза са СК: СК3, СК4, СК5, СК6,СК8

Предуслови: /

Постуслови: /



Слика 55 - Дијаграм секвенци "UcitajListuKlijenataSO"

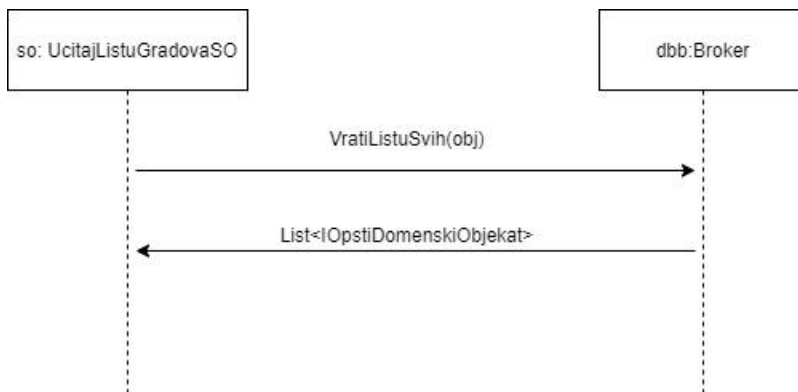
УГ5: UčitajGradove()

Операција: UčitajGradove(): signal

Веза са СК: СК2, СК4

Предуслови: /

Постуслови: /



Слика 56 - Дијаграм секвенци "UcitajListuGradovaSO"

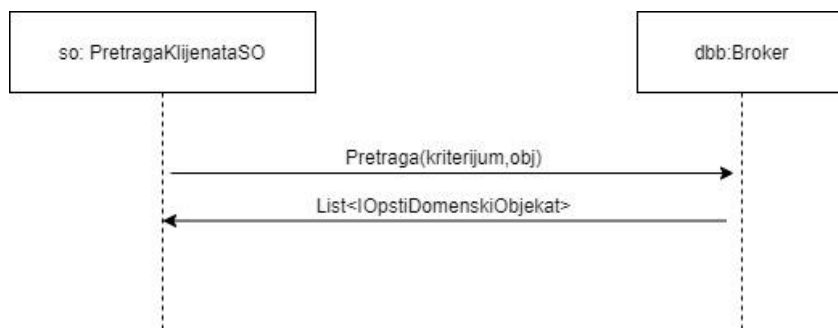
УГ6: NadjiKlijente(kriterijumPretrage, List<Klijent>)

Операција: NadjiKlijente(kriterijumPretrage, List<Klijent>): signal

Веза са СК: СК3, СК4, СК5

Предуслови: /

Постуслови: /



Слика 57 - Дијаграм секвенци "PretragaKlijenataSO"

УГ7: VratiKlijenta(Klijent)

Операција: VratiKlijenta(Klijent): signal

Веза са СК: СК3, СК4, СК5

Предуслови: /

Постуслови: /



Слика 58 - Дијаграм секвенци "VratiKlijentaSO"

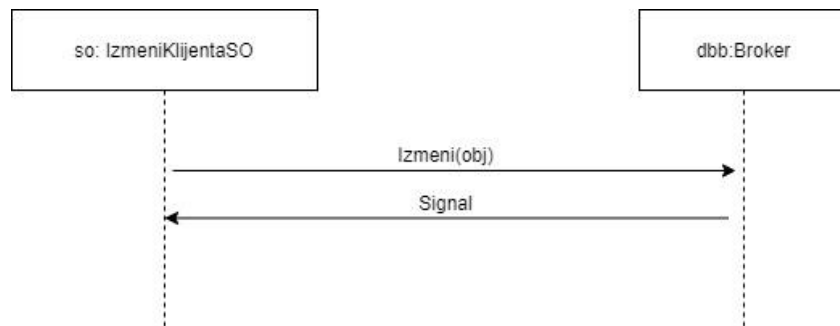
УГ8: IzmeniKlijenta(Klijent)

Операција: IzmeniKlijenta(Klijent): signal

Веза са СК: СК4

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

Постуслови: Унети подаци о клијенту су измењени.



Слика 59 - Дијаграм секвенци "IzmeniKlijentaSO"

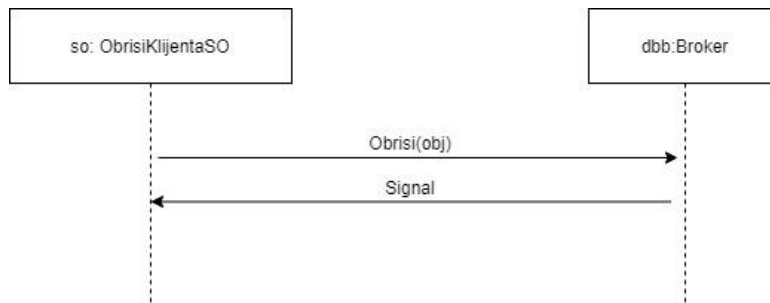
УГ9: ObrišiKlijenta(Klijent)

Операција: ObrišiKlijenta(Klijent): signal

Веза са СК: СК5

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

Постуслови: Клијент је обрисан.



Слика 60 - Дијаграм секвенци "ObrisiKlijentaSO"

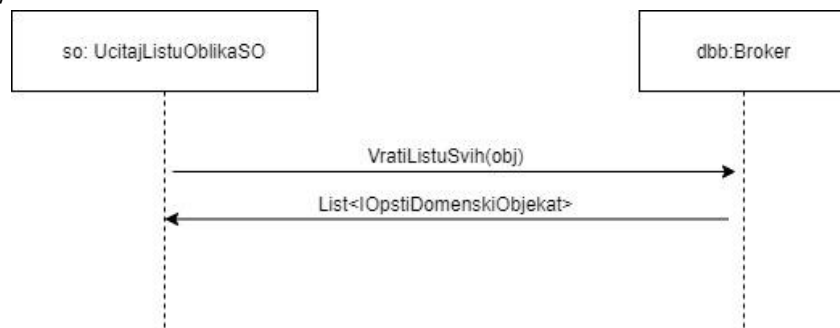
УГ10: UčitajOblikePsihoterapije()

Операција: UčitajOblikePsihoterapije(): signal

Веза са СК: СК6,СК8

Предуслови: /

Постуслови: /



Слика 61 - Дијаграм секвенци "UcitajListuOblikaSO"

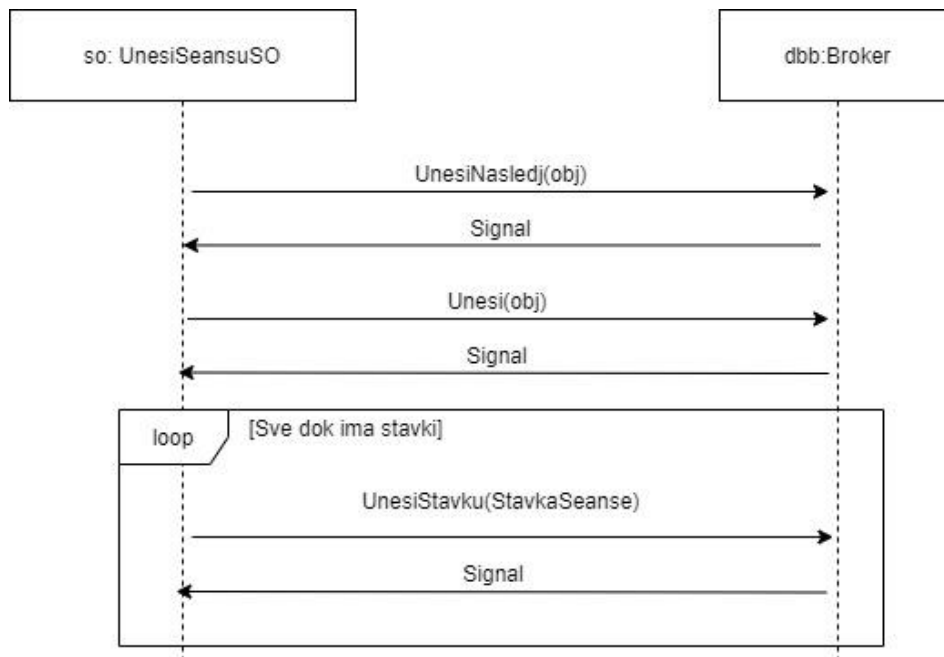
УГ11: UnesiSeansu (Seansa)

Операција: UnesiSeansu(Seansa)

Веза са СК: СК6

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

Постуслови: Евидентирана је нова сеанса.



Слика 62 - Дијаграм секвенци "UnesiSeansuSO"

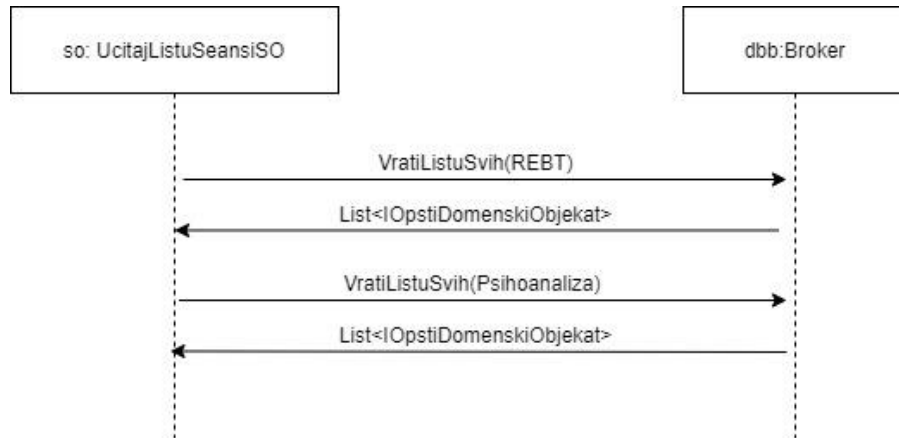
УГ12: UčitajListuSeansi()

Операција: UčitajListuSeansi(): signal

Веза са СК: CK7, CK8, CK9

Предуслови: /

Постуслови: /



Слика 63 - Дијаграм секвенци "UčitajListuSeansiSO"

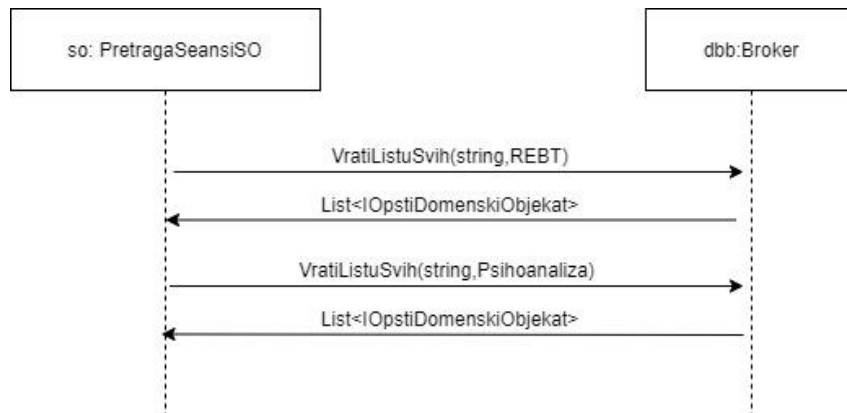
УГ13: NadjiSeanse (kriterijumPretrage, List<Seansa>)

Операција: NadjiSeanse(kriterijumPretrage, List<Seansa>): signal

Веза са СК: CK7, CK8, CK9

Предуслови: /

Постуслови: /



Слика 64 - Дијаграм секвенци "PretragaSeansiSO"

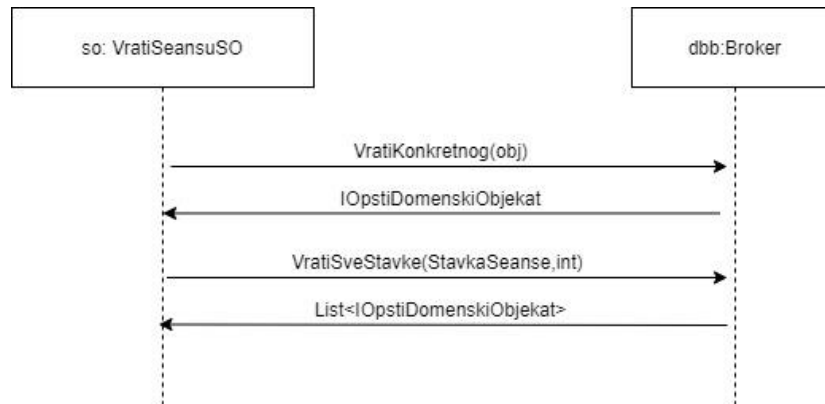
УГ14: VратиSeansu (Seansa)

Операција: VратиSeansu(Seansa): signal

Веза са СК: СК7, СК8, СК9

Предуслови: /

Постуслови: /



Слика 65 - Дијаграм секвенци "VратиSeansuSO"

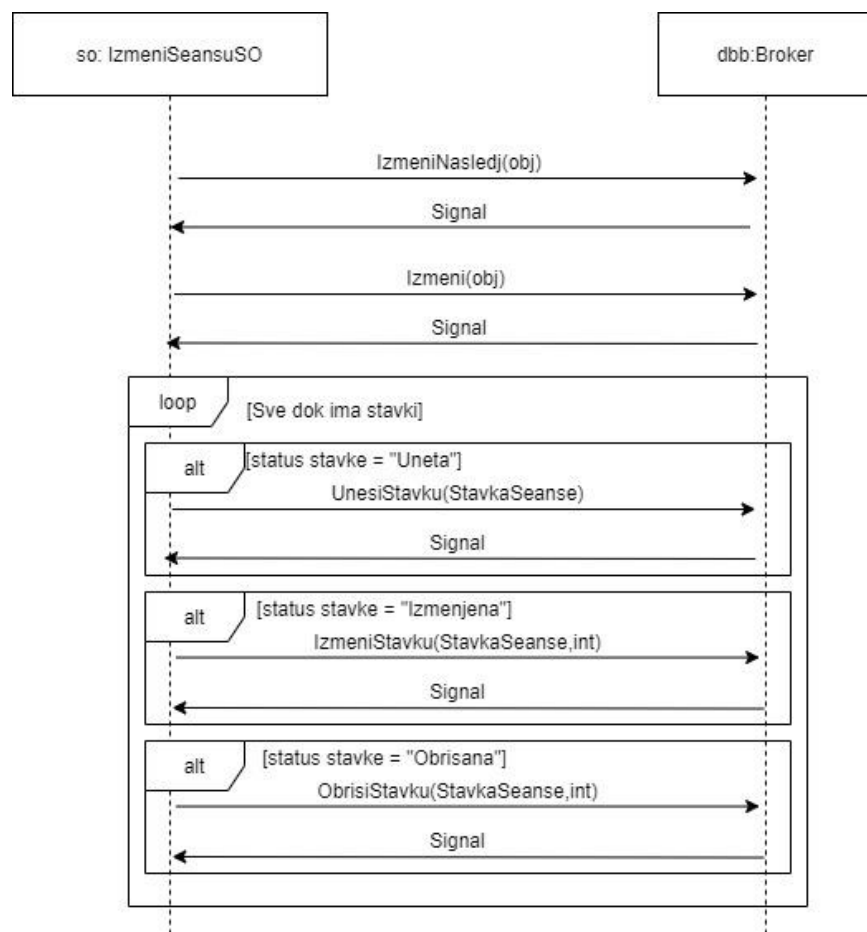
УГ15: IzmeniSeansu (Seansa)

Операција: IzmeniSeansu(Seansa): signal

Веза са СК: СК8

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

Постуслови: Унети подаци о сеанси су измењени.



Слика 66 - Дијаграм секвенци "IzmeniSeansuSO"

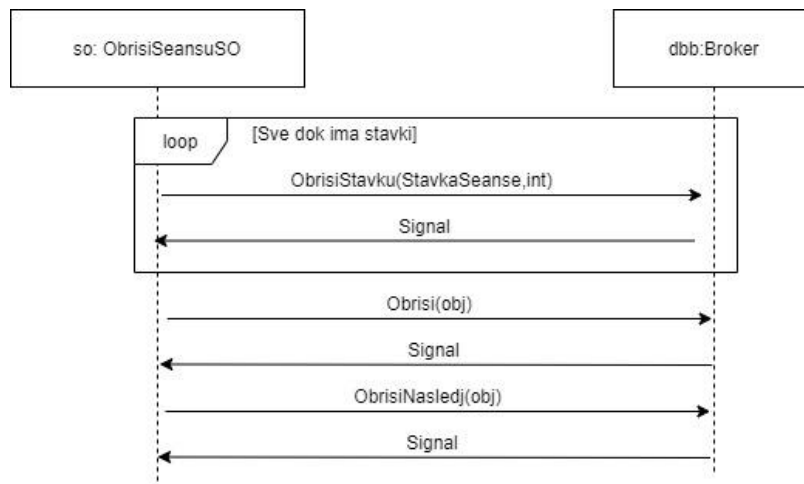
УГ16: ОбришиSeansu(Seansa)

Операција: ОбришиSeansu(Seansa): signal

Веза са СК: СК5

Предуслови: Структурна ограничења над објектом Сеанса морају бити задовољена.

Постуслови: Сеанса је обрисана.

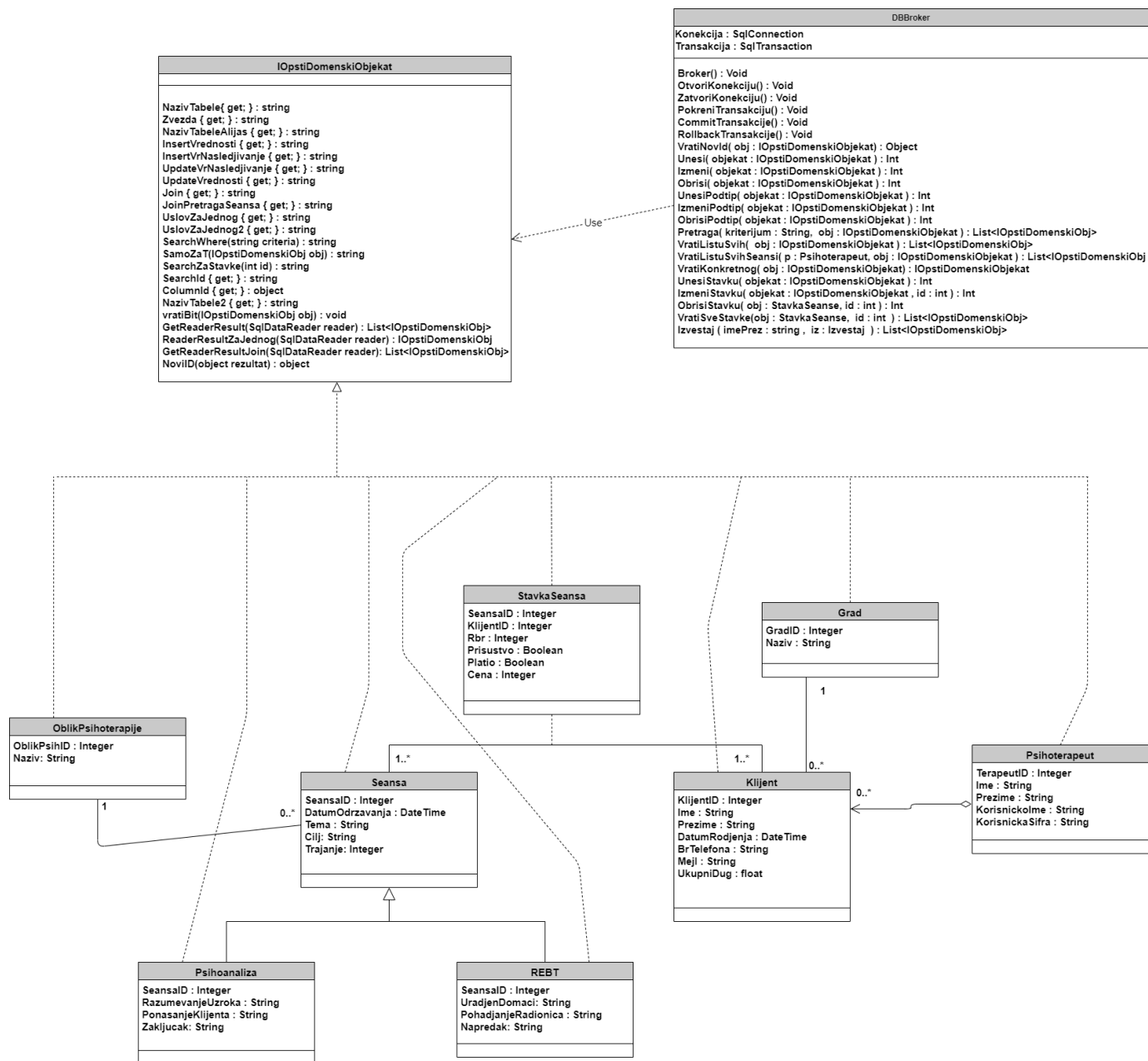


Слика 67 - Дијаграм секвенци "ObrisiSeansuSO"

Брокер базе података

DBBroker је софтверска класа одговорна за комуникацију између пословне логике и складишта података.

Класа DBBroker представља перзистентни оквир који посредује у свим операцијама над базом. Све методе у оквиру брокера базе података дефинисане су као генеричке, што значи да могу да прихвате објекте различитих класа и да над њима, прилагођавањем генеричких, креирају специфичне упите. Резултат је мања комплексност класе DBBroker. Да би се то омогућило коришћен је интерфејс *IOpstiDomenskiObjekat*. Свака класа из домена модела имплементира дати интерфејс, и све његове методе и гетере.

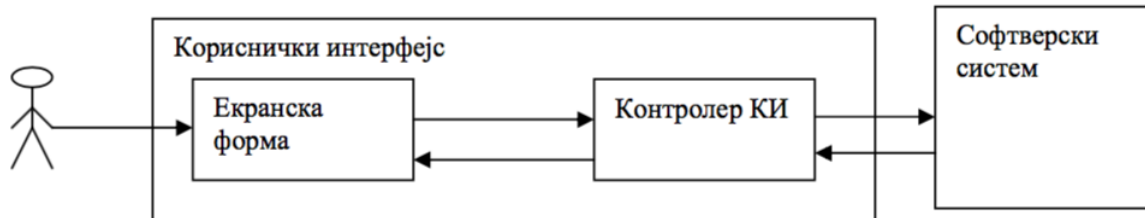


Слика 68 - Класа DBBroker повезана је са интерфејсом IDomenskimObjekat кога имплементирају софтверске класе модела

9.4. Пројектовање корисничког интерфејса

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

1. Екранске форме
2. Контролера корисничког интерфејса



Слика 69 - Пројектовање корисничког интерфејса

Екранска форма је одговорна да:

- прихвата податке које уноси актор,
- прихвата догађаје које прави актор,
- позива контролера графичког интерфејса, прослеђујући му прихваћене податке и
- приказује податке које је добила од контролера графичког интерфејса.

Контролер корисничког интерфејса је одговоран да:

- прихвати податке које шаље екранска форма,
- конвертује податке (који се налазе у графичким елементима) у објекат који представља улазни аргумент СО која ће бити позвана,
- шаље захтев за извршење СО до апликационог сервера (софтверског система),
- прихвата објекат (излаз) софтверског система који настаје као резултат извршења СО и
- конвертује објекат у податке графичких елемената.

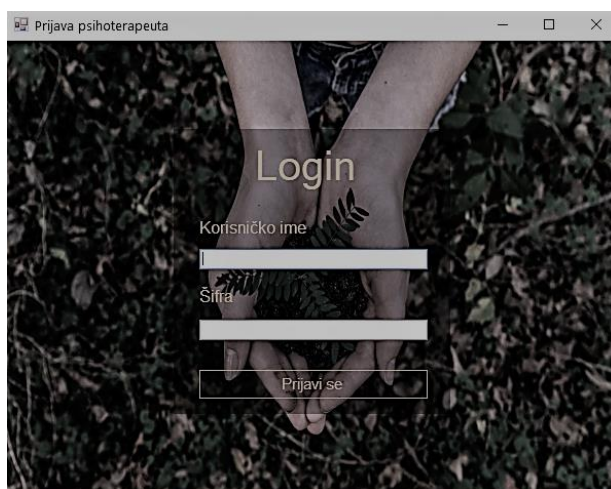
Пројектовање екранских форми

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

Постоје два аспекта пројектовања екранске форме:

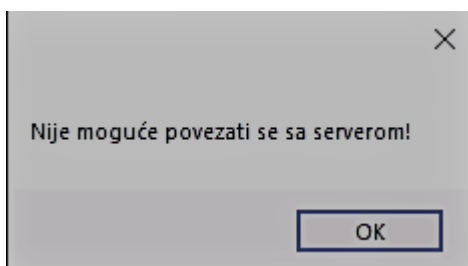
1. Пројектовање сценарија СК који се изводе преко екранске форме
2. Пројектовање метода екранске форме

Након покретања система, психотерапеуту се отвара форма за пријаву на систем:



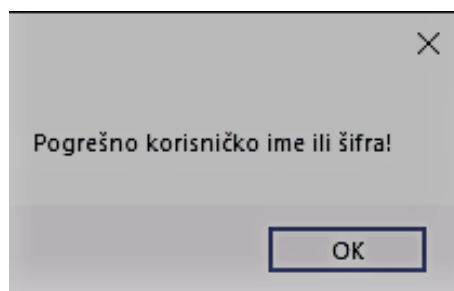
Слика 70 - Приказ форме за пријаву на систем

Уколико приликом пријављивања на систем дође до грешке због дисконекције сервера психотерапеуту ће се приказати следећа порука:



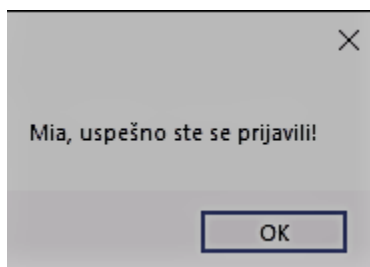
Слика 71 - Обавештење грешке у пријављивању

На наредној слици можете видети поруку која се приказује психотерапеуту при неуспешном пријављивању на систем:

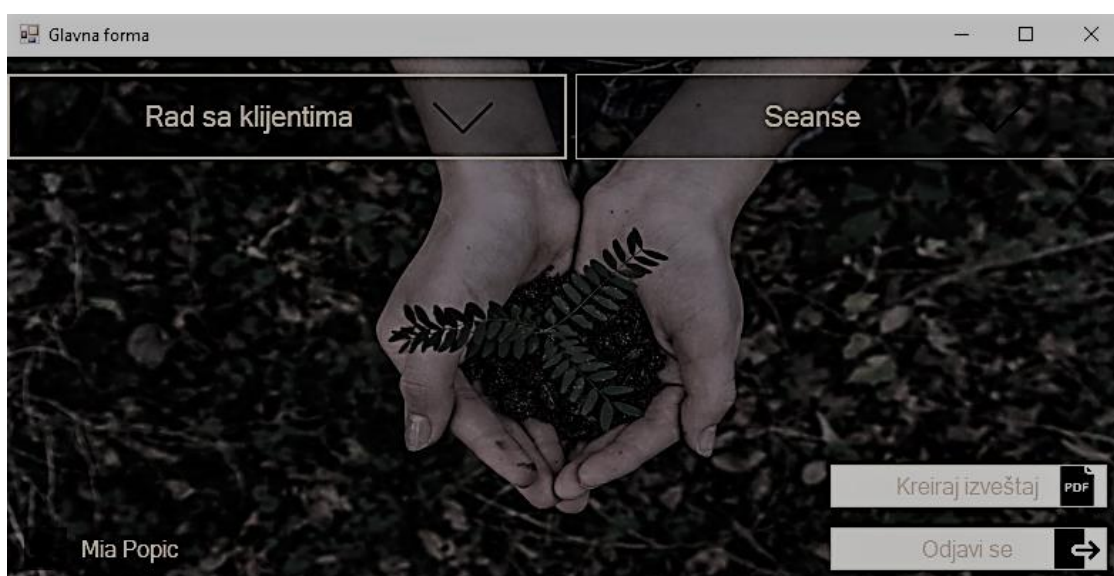


Слика 72 - Обавештење при неуспешном уносу корисничког имена или шифре

Уколико је пријављивање на систем било успешно, психотерапеуту се приказује следећа порука и отвара главна програмска форма са које он бира жељене операције система:



Слика 73 - Приказ успешне пријаве



Слика 74 - Приказ главне форме

Пројектовање сценарија СК

СК2: Унос новог клијента

Назив СК

Унос новог клијента

Актори СК

Психотерапеут

Учесници СК

Психотерапеут и систем (програм)

Предуслов: Систем је укључен и психотерапеут је улогован под својом шифром. Систем приказује форму за рад са клијентима. Учитана је листа градова.

Unos klijenta

ID

Lični podaci

Ime

Prezime

Datum rođenja(dd.MM.yyyy)

Prebivalište

Kontakt

Broj telefona:

E-mail:

Unesi klijenta

Mia Popic

Слика 75 – Приказ форме за рад са клијентима

Основни сценарио СК

6. Психотерапеут уноси податке о клијенту. (АПУСО)

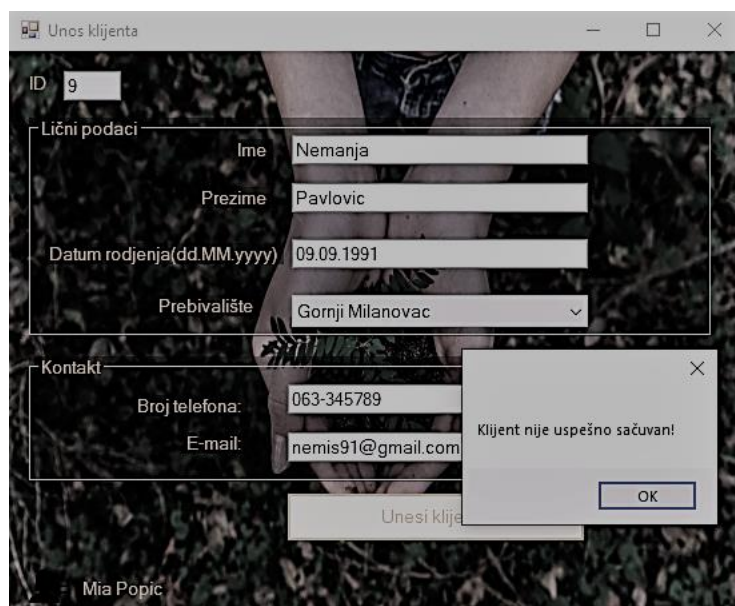
Слика 76 - Унос података о клијенту

7. **Психотерапеут контролише** да ли је коректно унео податке о **клијенту**. (АНСО)
8. **Психотерапеут позива систем** да запамти податке о **клијенту**. (АПСО)
9. **Систем памти** податке о **клијенту**. (СО)
10. **Систем приказује психотерапеуту** поруку: "Klijent uspešno sačuvan!" (ИА)

Слика 77 - Приказ успешног уноса клијента

Алтернативна сценарија

5.1. Уколико **систем** не може да запамти податке о **клијенту** он приказује **психотерапеуту** поруку: “Klijent nije sačuvan!” (ИА)



Слика 78 - Приказ неуспешног уноса клијента

СК3: Претраживање клијената

Назив СК

Претраживање клијената

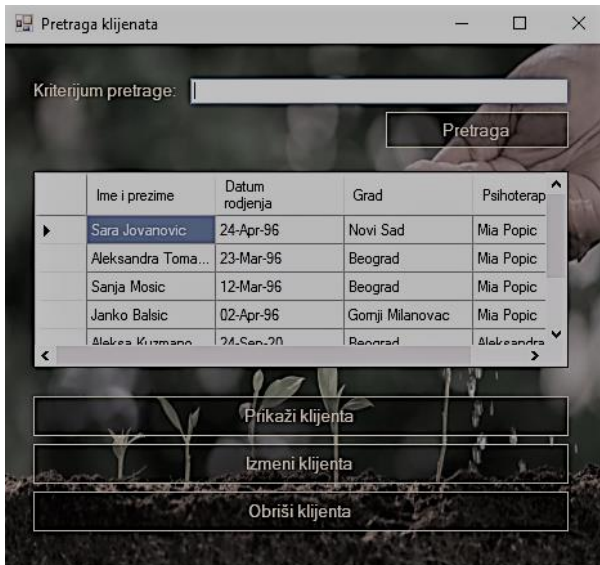
Актори СК

Психотерапеут

Учесници СК

Психотерапеут и систем (програм)

Предуслов: Систем је укључен и психотерапеут је улогован под својом шифром. Систем приказује форму за рад са клијентима. Учитана је листа клијената.



Pretraga klijenata

Kriterijum pretrage:

Pretraga

	Ime i prezime	Datum rođenja	Grad	Psihoterap
►	Sara Jovanovic	24-Apr-96	Novi Sad	Mia Popic
	Aleksandra Toma...	23-Mar-96	Beograd	Mia Popic
	Sanja Masic	12-Mar-96	Beograd	Mia Popic
	Janko Balsic	02-Apr-96	Gornji Milanovac	Mia Popic
	Aleksa Krizmanic	24-Sep-96	Beograd	Aleksandra

< >

Prikaži klijenta

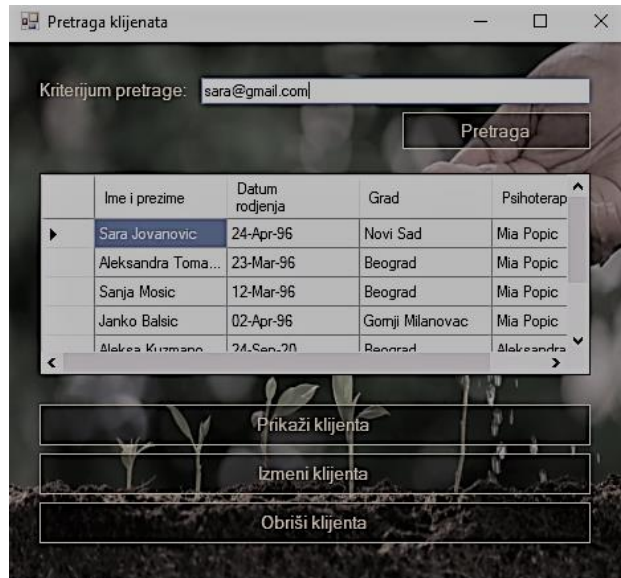
Izmeni klijenta

Obriši klijenta

Слика 79 - Приказ листе клијената

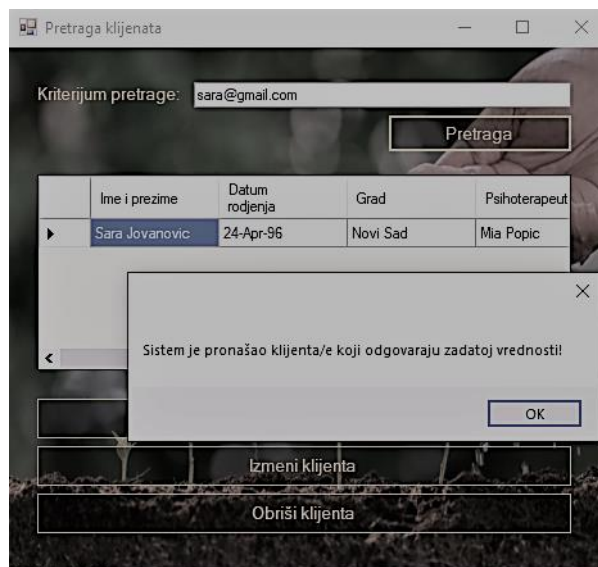
Основни сценарио СК

3. Психотерапеут уноси вредност по којој претражује клијенте. (АПУСО)



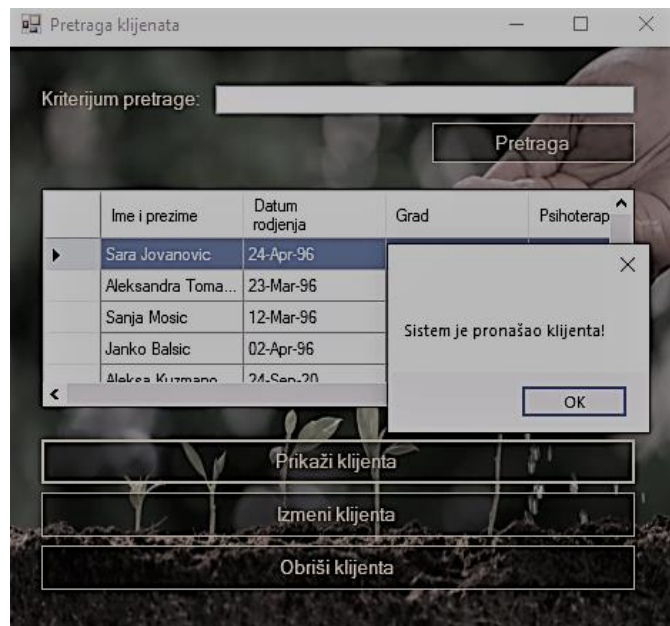
Слика 80 - Претрага клијената по критеријуму

4. **Психотерапеут** позива **систем** да нађе **клијенте** по задатој вредности. (АПСО)
5. **Систем** тражи **клијенте** по задатој вредности. (СО)
6. **Систем** приказује **психотерапеуту** листу **клијената** и поруку: "Sistem je pronašao klijente". (ИА)

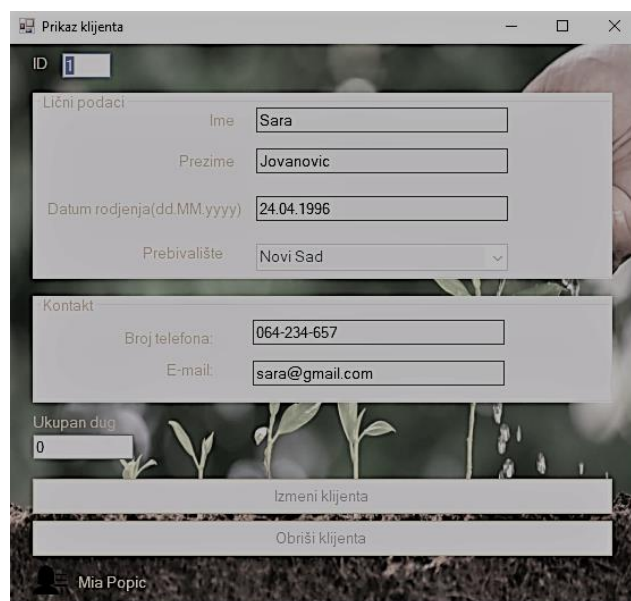


Слика 81 - Приказ резултата претраге клијената

7. **Психотерапеут** бира **клијента** чије податке жели да учита. (АПУСО)
8. **Психотерапеут** позива **систем** да учита податке о одабраном **клијенту**. (АПСО)
9. **Систем** учитава податке о одабраном **клијенту**. (СО)
10. **Систем** приказује **психотерапеуту** податке о **клијенту** и поруку: "Sistem je pronašao klijenta" (ИА)



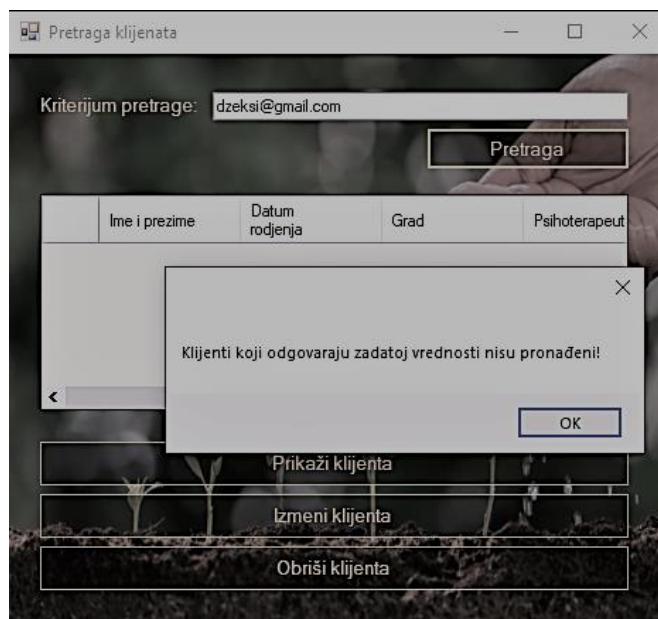
Слика 82 - Обавештење о успешном проналаску клијента



Слика 83 - Приказ одабраног клијента

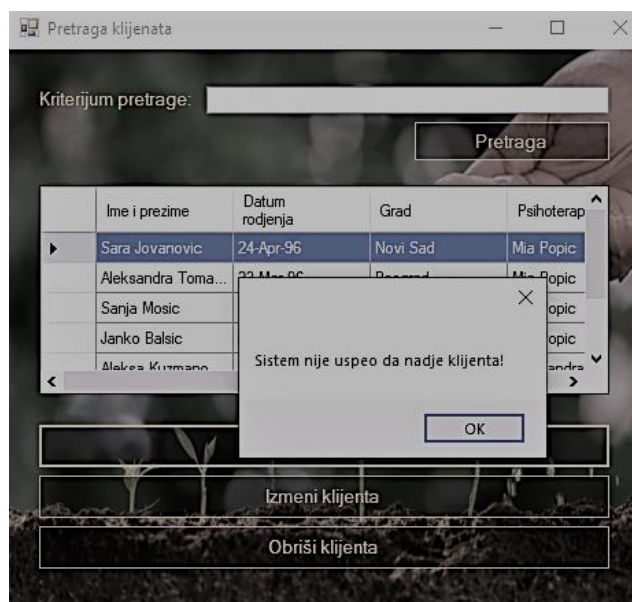
Алтернативна сценарија

- 4.1. Уколико **систем** не може да нађе **клијенте**, **систем** приказује **психотерапеуту** поруку: "Klijenti koji odgovaraju zadatoj vrednosti nisu pronađeni!" Прекида се извршење сценарија. (ИА)



Слика 84 - Приказ неуспешне претраге према критеријуму

8.1. Уколико **систем** не може да пронађе податке о **клијенту**, систем приказује **психотерапеуту** поруку: “Klijent nije pronađen!” (ИА)



Слика 85 - Обавештење о неуспешном проналазку података клијента

СК4: Измена података о клијенту

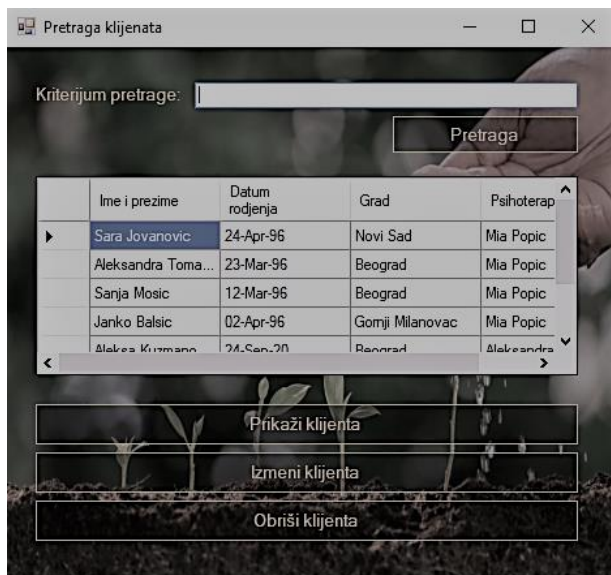
Назив СК

Измена података о **клијенту**

Актори СК
Психотерапеут

Учесници СК
Психотерапеут и **систем** (програм)

Предуслов: **Систем** је укључен и **психотерапеут** је улогован под својом шифром. **Систем** приказује форму за рад са **клијентима**. Учитана је листа **клијената**. Учитана је листа градова.



Pretraga klijenata

Kriterijum pretrage:

Pretraga

	Ime i prezime	Datum rođenja	Grad	Psihoterap
►	Sara Jovanovic	24-Apr-96	Novi Sad	Mia Popic
	Aleksandra Toma...	23-Mar-96	Beograd	Mia Popic
	Sanja Masic	12-Mar-96	Beograd	Mia Popic
	Janko Balsic	02-Apr-96	Gornji Milanovac	Mia Popic
	Aleksa Kirmann	24-Sep-20	Beograd	Aleksandra

< >

Prikaži klijenta

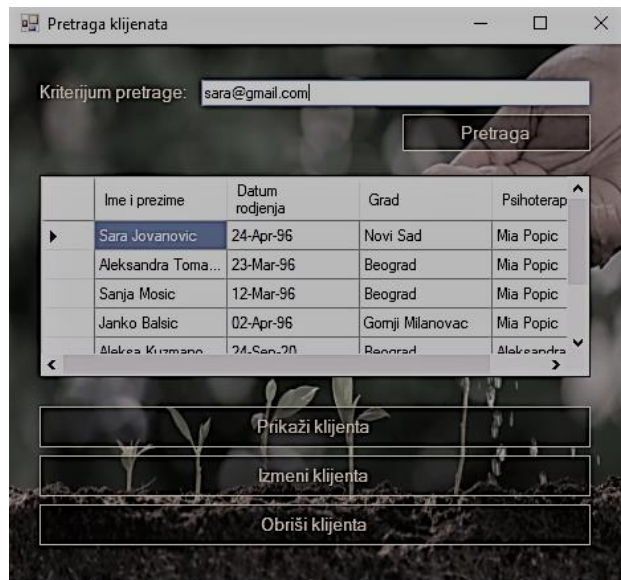
Izmeni klijenta

Obriši klijenta

Слика 86 - Приказ претраге клијената

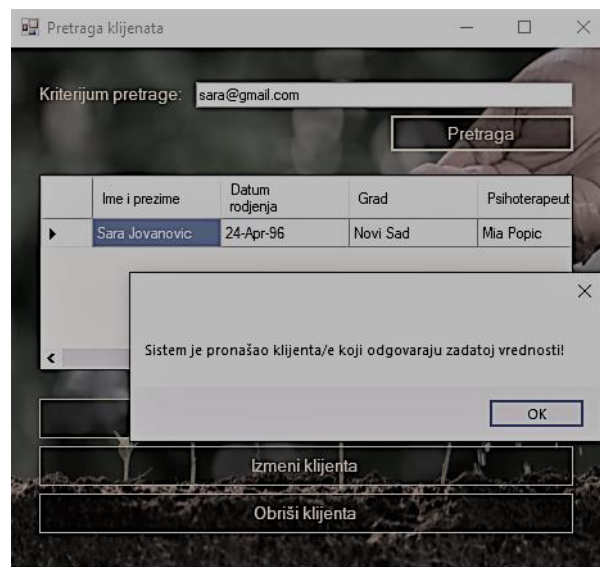
Основни сценарио СК

14. **Психотерапеут** уноси вредност по којој претражује **клијенте**. (АПУСО)



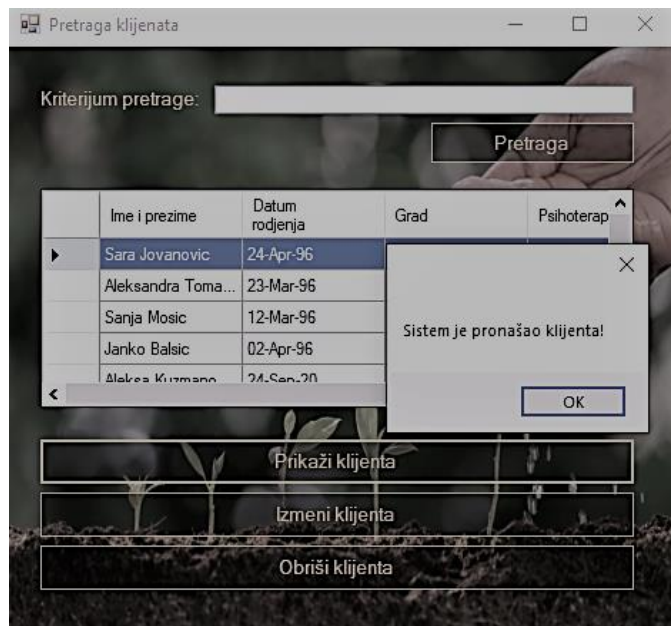
Слика 87 - Претрага клијената према унесеном критеријуму

15. **Психотерапеут** позива **систем** да нађе **клијенте** по задатој вредности. (АПСО)
16. **Систем** тражи **клијенте** по задатој вредности. (СО)
17. **Систем** приказује **психотерапеуту** листу **клијената** и поруку: "Sistem je pronašao klijente". (ИА)



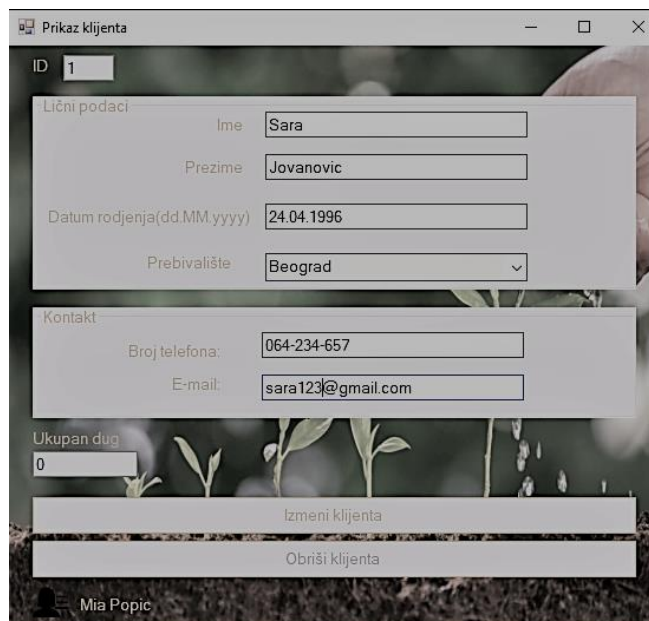
Слика 88 - Обавештење о успешној претрази

18. **Психотерапеут** бира **клијента** чије податке жели да измени. (АПУСО)
19. **Психотерапеут** позива **систем** да учита податке о одабраном **клијенту**. (АПСО)
20. **Систем** учитава податке о одабраном **клијенту**. (СО)
21. **Систем** приказује **психотерапеуту** податке о **клијенту** и поруку: "Sistem je pronašao klijenta". (ИА)



Слика 89 - Обавештење о успешно пронађеном клијенту

22. **Психотерапеут** мења податке о **клијенту**. (АПУСО)



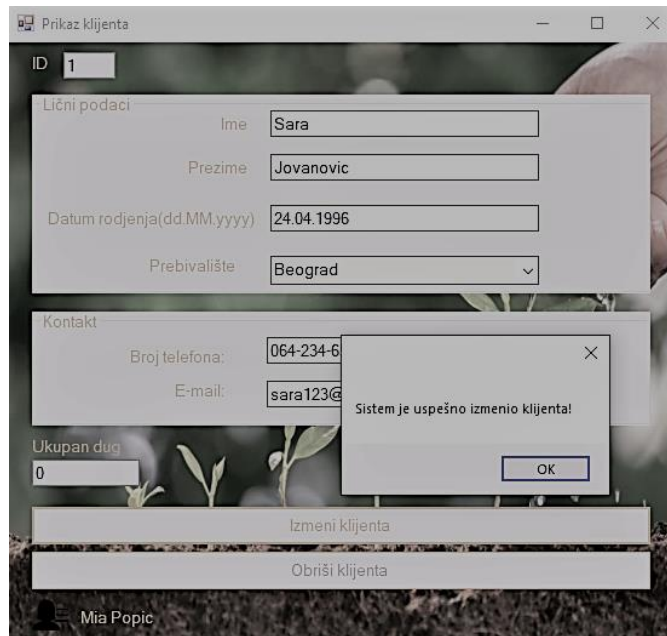
Слика 90 - Приказ промене података клијента

23. **Психотерапеут** контролише да ли је коректно унео податке о **клијенту**. (АНСО)

24. **Психотерапеут** позива **систем** да запамти податке о **клијенту**. (АПСО)

25. **Систем** памти податке о **клијенту**. (СО)

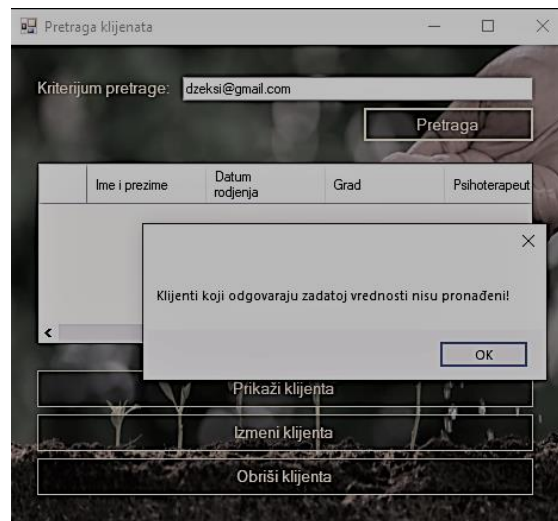
26. **Систем** приказује **психотерапеуту** поруку: "Klijent uspešno izmenjen!" (ИА)



Слика 91 - Обавештење о успешној измени података клијента

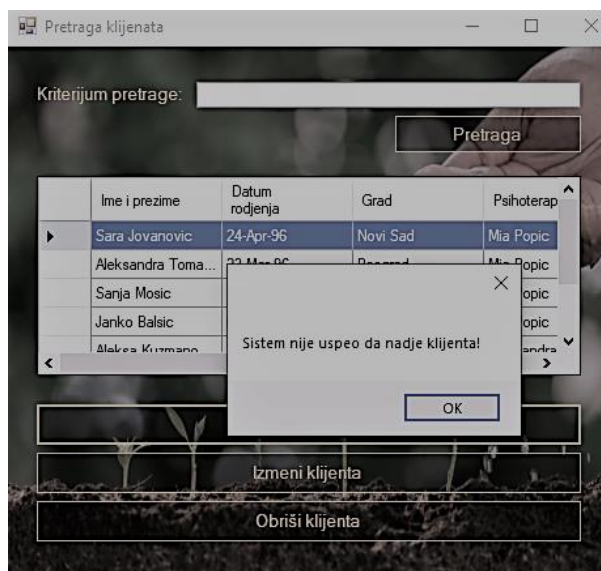
Алтернативна сценарија

4.1. Уколико **систем** не може да нађе **клијенте** он приказује **психотерапеуту** поруку: “Klijenti koji odgovaraju zadatoj vrednosti nisu pronadeni!” Прекида се извршење сценарија. (ИА)



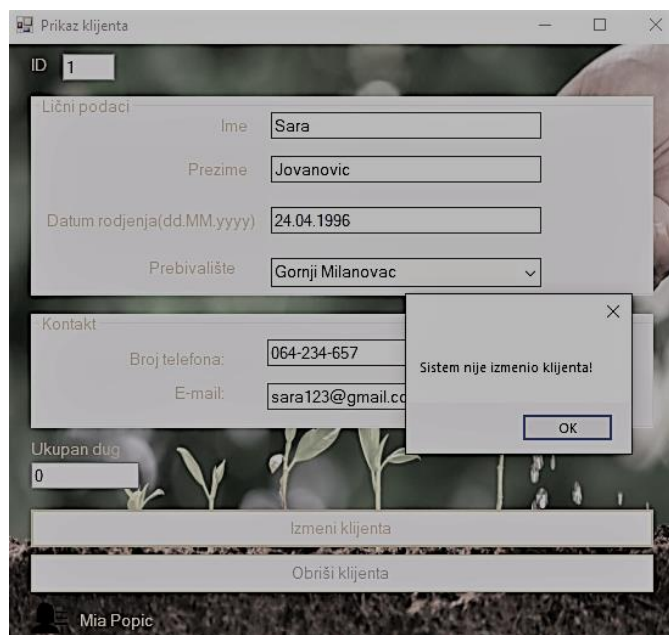
Слика 92 - Обавештење о неуспешној претрази

8.1. Уколико **систем** не може да пронађе податке о **клијенту**, **систем** приказује **психотерапеуту** поруку: “Klijent nije pronaden!” Прекида се извршење сценарија. (ИА)



Слика 93 - Обавештење о неуспешном проналаску података клијента

- 13.1. Уколико **систем** не може да запамти податке о **клијенту** он приказује **психотерапеуту** поруку: "Klijent nije izmenjen!" (ИА)



Слика 94 - Обавештење о неуспешној измени података клијента

СК5: Брисање клијента

Назив СК

Брисање клијента

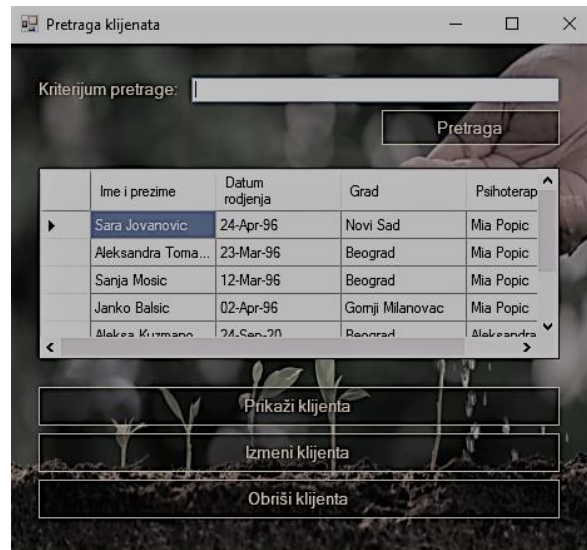
Актори СК

Психотерапеут

Учесници СК

Психотерапеут и систем (програм)

Предуслов: Систем је укључен и психотерапеут је улогован под својом шифром. Систем приказује форму за рад са клијентима. Учитана је листа клијената.



Kriterijum pretrage:

Pretraga

	Ime i prezime	Datum rođenja	Grad	Psihoterap
►	Sara Jovanovic	24-Apr-96	Novi Sad	Mia Popic
	Aleksandra Toma...	23-Mar-96	Beograd	Mia Popic
	Sanja Masic	12-Mar-96	Beograd	Mia Popic
	Janko Balsic	02-Apr-96	Gornji Milanovac	Mia Popic
	Aleksa Krivman	24-Sep-96	Beograd	Aleksandra

< >

Prikaži klijenta

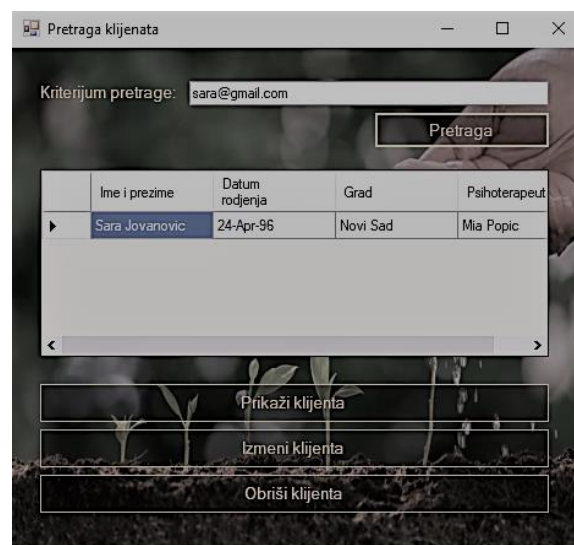
Izmeni klijenta

Obriši klijenta

Слика 95 - Приказ учитане листе клијената

Основни сценарио СК

12. Психотерапеут уноси вредност по којој претражује клијенте. (АПУСО)



Kriterijum pretrage: sara@gmail.com

Pretraga

	Ime i prezime	Datum rođenja	Grad	Psihoterapeut
►	Sara Jovanovic	24-Apr-96	Novi Sad	Mia Popic

< >

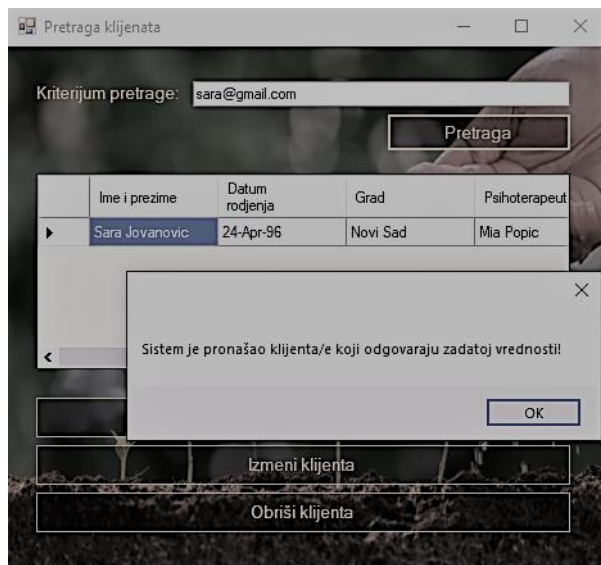
Prikaži klijenta

Izmeni klijenta

Obriši klijenta

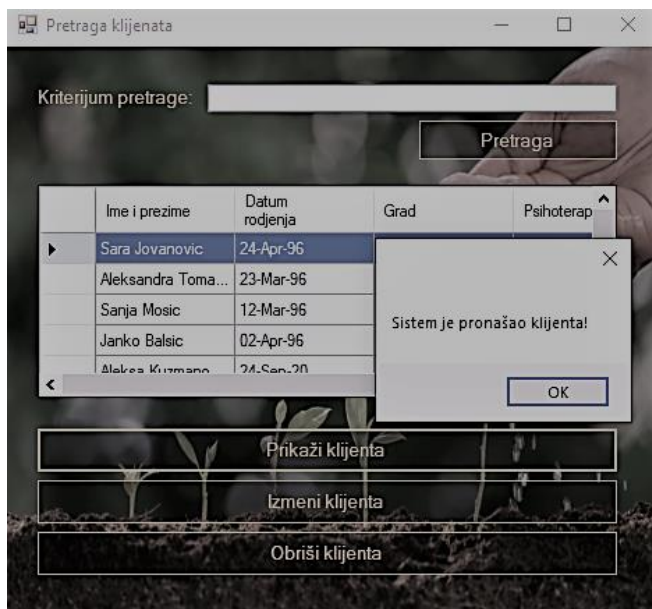
Слика 96 - Приказ претраге према критеријуму

13. **Психотерапеут** **позива** **систем** да нађе **клијенте** по задатој вредности. (АПСО)
14. **Систем** **тражи** **клијенте** по задатој вредности. (СО)
15. **Систем** **приказује** **психотерапеуту** листу **клијената** и поруку: “Sistem je pronašao klijente”. (ИА)

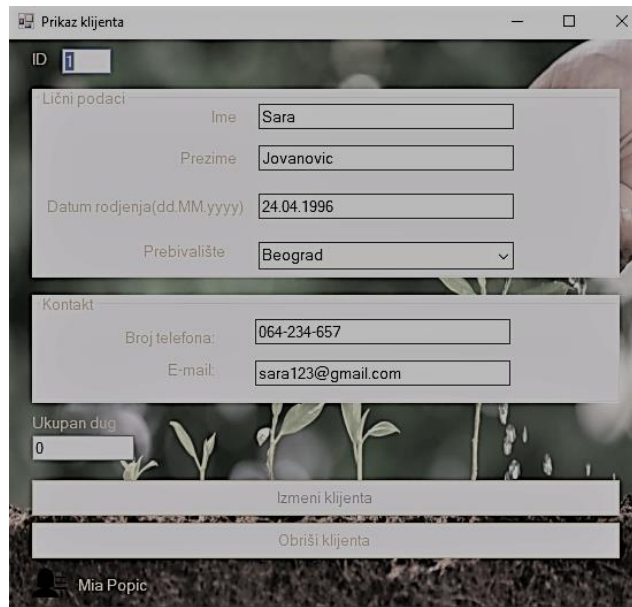


Слика 97 - Обавештење о успешном проналаску клијента

16. **Психотерапеут** **бира** **клијента** ког жели да обрише. (АПУСО)
17. **Психотерапеут** **позива** **систем** да учита податке о одабраном **клијенту**. (АПСО)
18. **Систем** **учитава** податке о одабраном **клијенту**. (СО)
19. **Систем** **приказује** **психотерапеуту** податке о **клијенту** и поруку: “Sistem je pronašao klijenta”. (ИА)

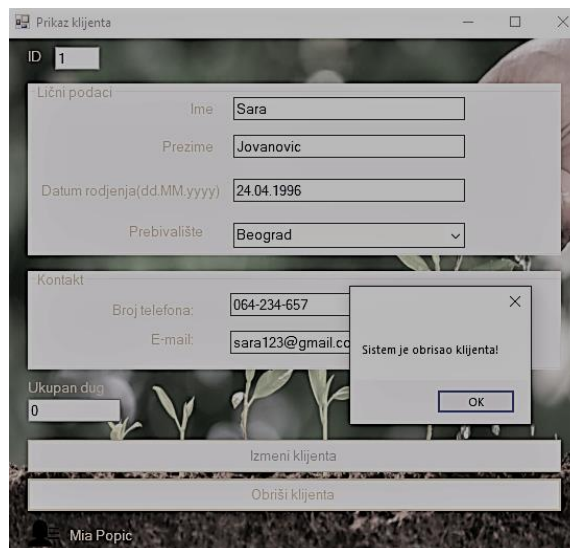


Слика 98 - Обавештење о успешном проналаску података клијента



Слика 99 - Приказ података клијента

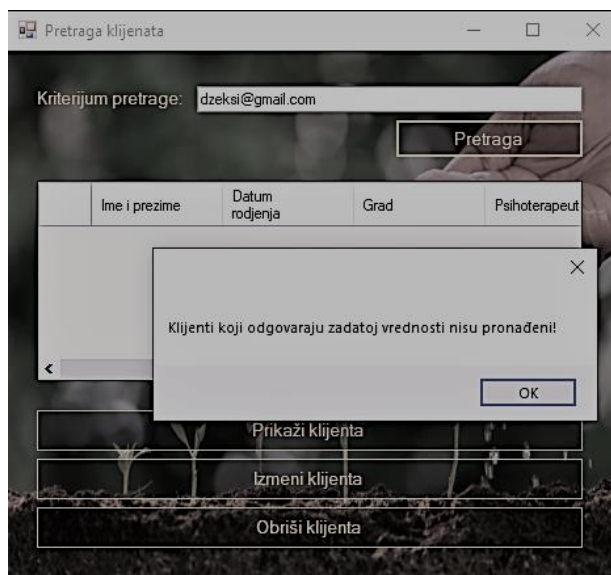
20. Психотерапеут позива систем да обрише клијента. (АПСО)
21. Систем брише клијента. (СО)
22. Систем приказује психотерапеуту поруку: "Klijent obrisan!" (ИА)



Слика 100 - Обавештење о успешном брисању клијента

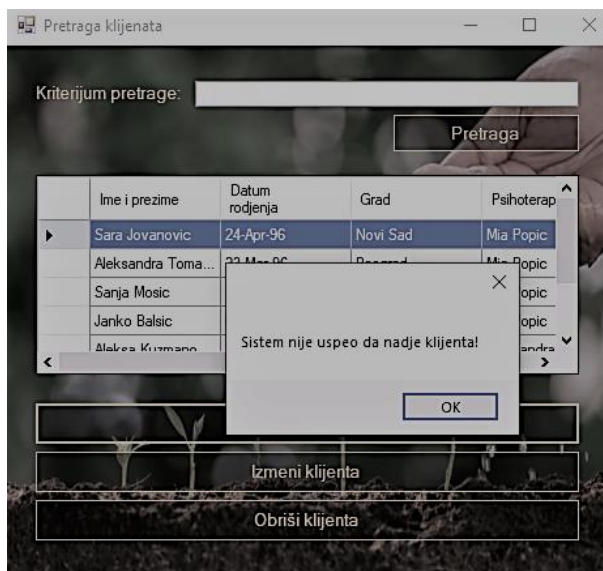
Алтернативна сценарија

4.1. Уколико систем не може да нађе клијенте он приказује психотерапеуту поруку: "Klijenti koji odgovaraju zadatoj vrednosti nisu pronađeni!" Прекида се извршење сценарија. (ИА)



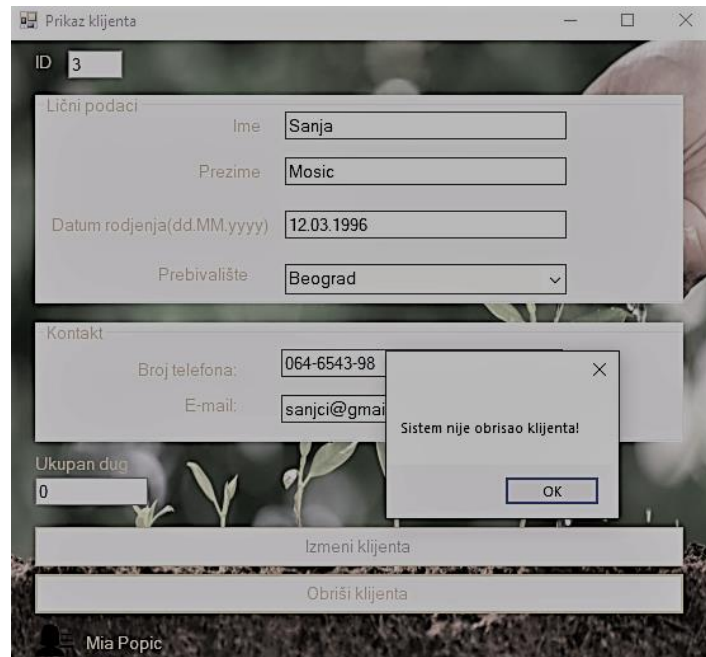
Слика 101 - Обавештење о неуспешној претрази према критеријуму

8.1. Уколико **систем** не може да пронађе податке о **клијенту**, **систем** приказује **психотерапеуту** поруку: “Klijent nije pronaden!” Прекида се извршење сценарија. (ИА)



Слика 102 - Обавештење о неуспешном проналазку података клијента

11.1. Уколико **систем** не може да обрише **клијента** он приказује **психотерапеуту** поруку: “Klijent nije obrisan!” (ИА)



Слика 103 - Приказ обавештења о неуспешном брисању клијента

СК6: Унос нове сеансе

Назив СК

Унос нове **сеансе**

Актори СК

Психотерапеут

Учесници СК

Психотерапеут и **систем** (програм)

Предуслов: **Систем** је укључен и **психотерапеут** је улогован под својом шифром. **Систем** приказује форму за рад на **сеансама** . Учитана је листа клијената. Учитане су врста и облик психотерапије.

Unos nove seanse

ID:

Trajanje seanse:

Datum(format dd.MM.yyyy):

--izaberi vrstu terapije--

Tema:

Generalni cilj:

Oblik psihoterapije:

Unos stavke

Klijent:

☐ Bio/la na seansi

Cena seanse:

☐ Plaćena seansa

Mia Popic

Слика 104 - Приказ уноса нове сеансе

Основни сценарио СК

6. Психотерапеут попуњава податке о сеанси. (АПУСО)

Unos nove seanse

ID:

Trajanje seanse:

Datum(format dd.MM.yyyy):

REBT

Tema:

Generalni cilj:

Uradjen domaci:

Pohadjanje radionica:

Napredak:

Oblik psihoterapije:

Unos stavke

Klijent:

☐ Bio/la na seansi

Cena seanse:

☐ Plaćena seansa

	Rbr	Klijent	Prisustvo	Cena seansa	Plaćena seansa	Status
▶	1	Bojana ...	<input checked="" type="checkbox"/>	1500	<input type="checkbox"/>	Uneta

Mia Popic

Слика 105 - Приказ попуњене форме о новој сеанси

Unos nove seanse

ID: 1

Trajanje seanse: 50

Datum(format dd.MM.yyyy): 17.07.2020

Psihoanaliza

Tema: Tema 1

Generalni cilj: Cilj 1

Ponašanje/govor tela: ok

Razumevanje uzroka: ok

Zaključak: Napreduje se!

Oblik psihoterapije: Parovi

Unos stavke

Klijent: [dropdown]

Potvrdi izmene

☐ Bio/la na seansi

Cena seanse: [input]

☐ Plaćena seansa

Unesi stavku

Izmeni odabranu stavku

Obriši odabranu stavku

Rbr	Klijent	Prisustvo	Cena seansa	Plaćena seansa	Status
1	Janko ...	<input checked="" type="checkbox"/>	1500	<input checked="" type="checkbox"/>	Uneta
2	Aleksa...	<input checked="" type="checkbox"/>	1500	<input checked="" type="checkbox"/>	Uneta

Sačuvaj seansu

Слика 106 - Приказ попуњене форме о новој сеанси

7. **Психотерапеут контролише** да ли је коректно унео податке о **сеанси**. (АНСО)
8. **Психотерапеут позива систем** да запамти податке о **сеанси**. (АПСО)
9. **Систем памти** податке о **сеанси**. (СО)
10. **Систем приказује психотерапеуту** поруку: "Seansa uspešno sačuvana!" (ИА)

Unos nove seanse

ID: 1

Trajanje seanse: 50

Datum(format dd.MM.yyyy): 17.07.2020

REBT

Tema: Tema 1

Generalni cilj: Cilj 1

Uradjen domaci: da

Pohadjanje radionica: da

Napredak: veliki

Oblik psihoterapije: Individualna terapija

Unos stavke

Klijent: [dropdown]

Potvrdi izmene

☐ Bio/la na seansi

Cena seanse: [input]

☐ Plaćena seansa

Unesi stavku

Izmeni odabranu stavku

Obriši odabranu stavku

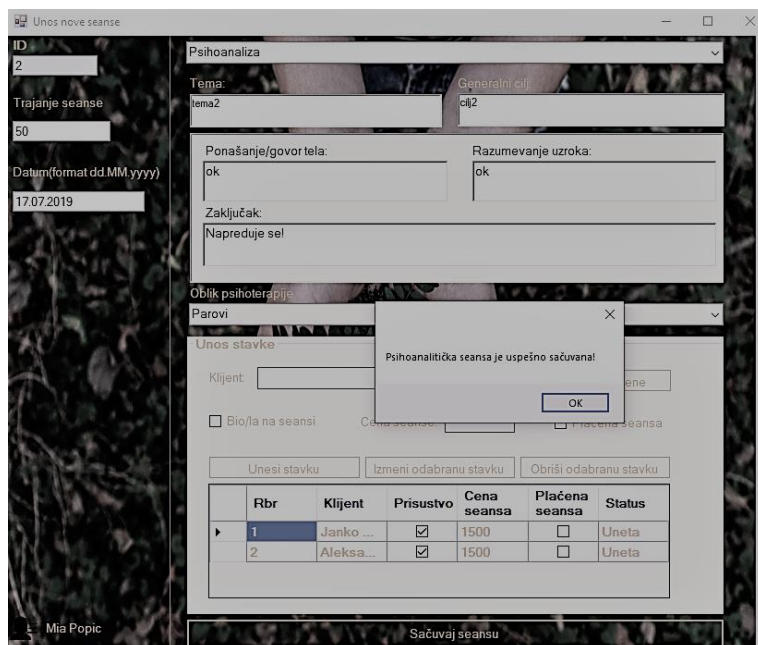
Rbr	Klijent	Prisustvo	Cena seansa	Plaćena seansa	Status
1	Bojana ...	<input checked="" type="checkbox"/>	1500	<input type="checkbox"/>	Uneta

Sačuvaj seansu

REBT seansa je uspešno sačuvana!

OK

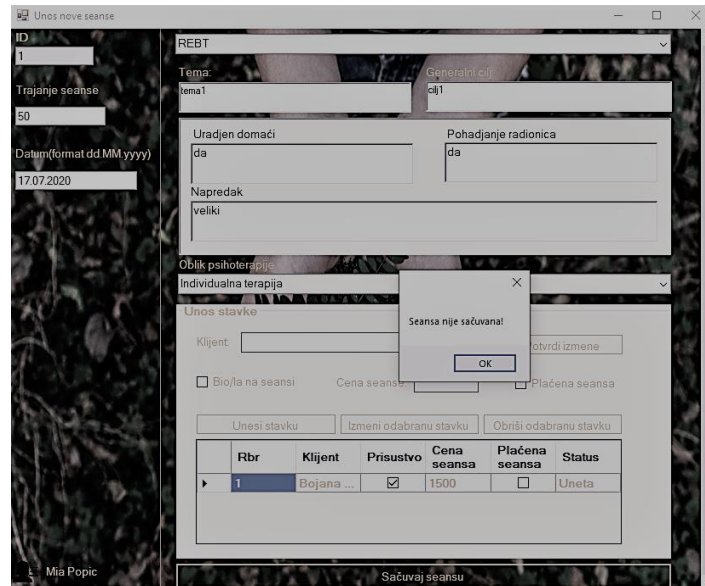
Слика 107 - Обавештење о успешном уносу РЕБТ сеансе



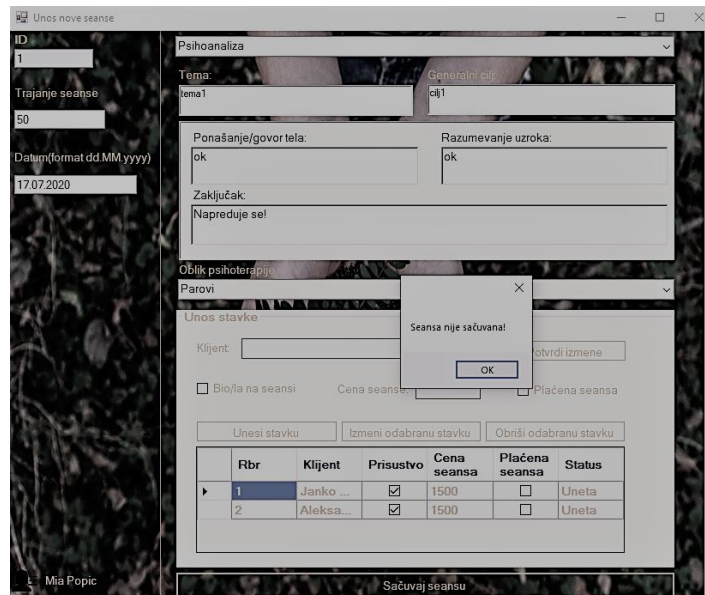
Слика 108 - Обавештење о успешном уносу психоаналитичке сеансе

Алтернативна сценарија

5.1. Уколико **систем** не може да запамти податке о **сеанси** он приказује **психотерапеуту** поруку: "Seansa nije sačuvana!" (IA)



Слика 109 - Обавештење о неуспешном уносу РЕБТ сеансе



Слика 110 - Обавештење о неуспешном уносу психоаналитичке сеансе

СК7: Претраживање сеанси

Назив СК

Претраживање сеанси

Актори СК

Психотерапеут

Учесници СК

Психотерапеут и систем (програм)

Предуслов: Систем је укључен и психотерапеут је улогован под својом шифром. Систем приказује форму за рад са сеансама. Систем приказује листу сеанси.

The screenshot shows a window titled "Pretraga seansi". At the top, there is a text input field labeled "Kriterijum:" followed by a search button labeled "Pretraga". Below this is a table with the following columns: "DatumOdržavanja", "Trajanje", "Tema", and "Cilj". The table contains four rows of data. Below the table are three buttons: "Prikaži seansu", "Izmeni seansu", and "Obriši seansu".

	DatumOdržavanja	Trajanje	Tema	Cilj
▶	17-Jul-20	50	tema1	cilj1
	01-Apr-19	50	tema3	cilj3
	17-Jul-19	50	tema2	cilj2
	03-May-20	50	tema4	cilj4

Слика 111 - Приказ форме за претрагу сеанси

Основни сценарио СК

11. Психотерапеут уноси вредност по којој претражује сеансе. (АПУСО)

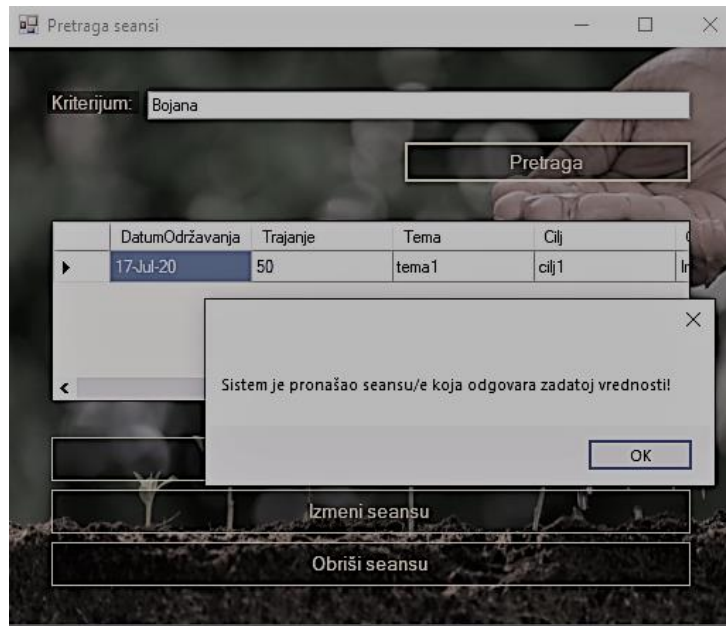
This screenshot is identical to the previous one, but the "Kriterijum:" field now contains the text "BojanaJ".

Слика 112 - Претрага сеанси према заадатом критеријуму

12. Психотерапеут позива систем да нађе сеансе по задатој вредности. (АПСО)

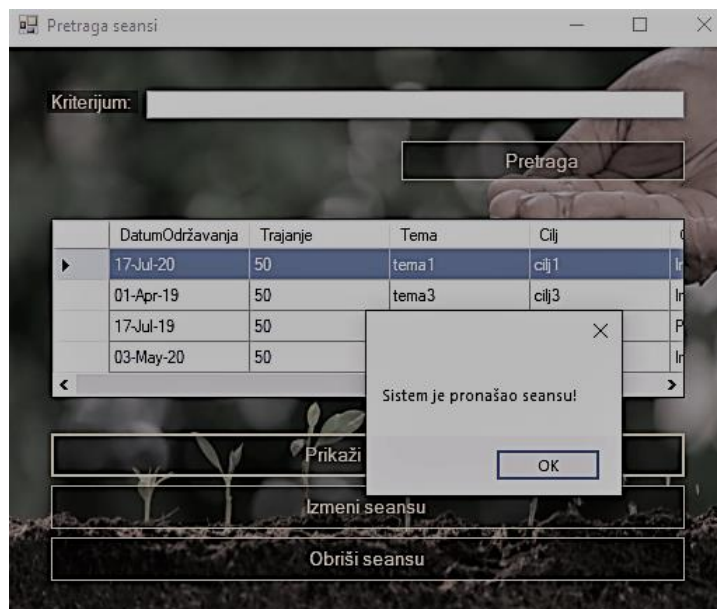
13. Систем тражи сеансе по задатој вредности. (СО)

14. Систем приказује психотерапеуту листу сеанси и поруку: "Sistem je pronašao seanse". (ИА)



Слика 113 - Обавештење о успешној претрази

15. **Психотерапеут** бира **сеансу** чије податке жели да учита. (АПУСО)
16. **Психотерапеут** **позива систем** да учита податке о одабраној **сеанси**. (АПСО)
17. **Систем** **учитава** податке о одабраној **сеанси**. (СО)
18. **Систем** **приказује** **психотерапеуту** податке о **сеанси** и поруку: "Sistem je pronašao seansu". (ИА)



Слика 114 - Обавештење о успешном проналаску сеансе

Prikaz Seanse

ID: 1

Trajanje seanse: 50

Datum (format dd.MM.yyyy): 17.07.2020

Tema: REBT

Generalni cilj: cilj1

Uradjen domaći: da

Pohadjanje radionica: da

Napredak: veliki

Oblik psihoterapije: Individualna terapija

Unos stavke:

Klijent: ~Izaberi~

Potvrdi izmene

☐ Bio/la na seansi

Cena seanse:

☐ Plaćena seansa

Unesi stavku

Izmeni odabranu stavku

Obrisi odabranu stavku

Rbr	Klijent	Prisustvo	Cena seansa	Plaćena seansa	Status
1	Bojana ...	<input checked="" type="checkbox"/>	1500	<input type="checkbox"/>	NijeMe...

Слика 115 - Приказ одабране сеансе

Алтернативна сценарија

4.1. Уколико **систем** не може да нађе **сеансе**, **систем** приказује **психотерапеуту** поруку: "Seanse koje odgovaraju zadatoj vrednosti nisu pronađene!" Прекида се извршење сценарија. (ИА)

Pretraga seansi

Kriterijum: Nevena

Pretraga

DatumOdržavanja	Trajanje	Tema	Cilj
-----------------	----------	------	------

Seanse koje odgovaraju zadatoj vrednosti nisu pronađene!

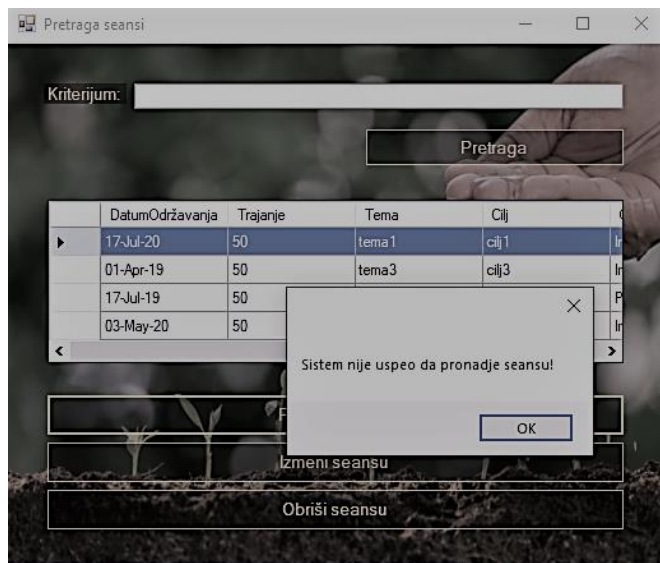
OK

Izmeni seansu

Obrisi seansu

Слика 116 - Обавештење о неуспешној претрази

8.1. Уколико **систем** не може да пронађе **сеансу** он приказује **психотерапеуту** поруку: "Seansa nije pronađena!" (ИА)



Слика 117 - Обавештење о неуспешном проналаску одабране сеансе

СК8: Измена садржаја сеансе

Назив СК

Измена садржаја **сеансе**

Актори СК

Психотерапеут

Учесници СК

Психотерапеут и **систем** (програм)

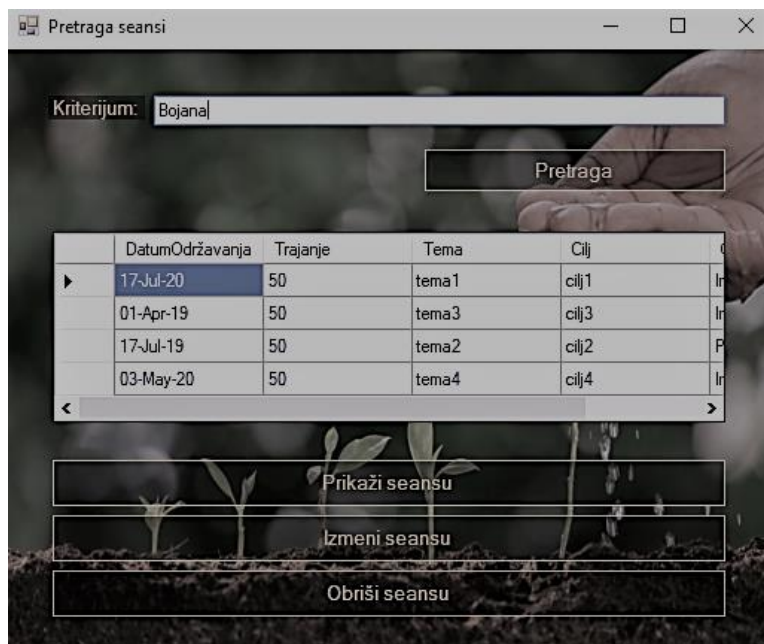
Предуслов: **Систем** је укључен и **психотерапеут** је улогован под својом шифром. **Систем** приказује форму за рад са **сеансама**. Учитана је листа **сеанси** као и листа клијената. Учитана је врста психотерапије као и облик психотерапије.



Слика 118 - Приказ форме претраге сеанси

Основни сценарио СК

14. **Психотерапеут** уноси вредност по којој претражује **сеансе**. (АПУСО)

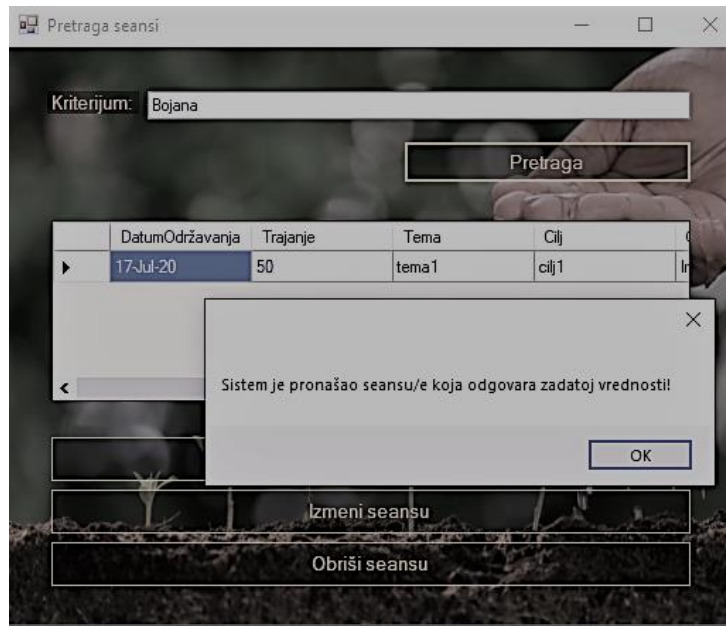


Слика 119 - Претрага према критеријуму

15. **Психотерапеут** позива **систем** да нађе **сеансе** по задатој вредности. (АПСО)

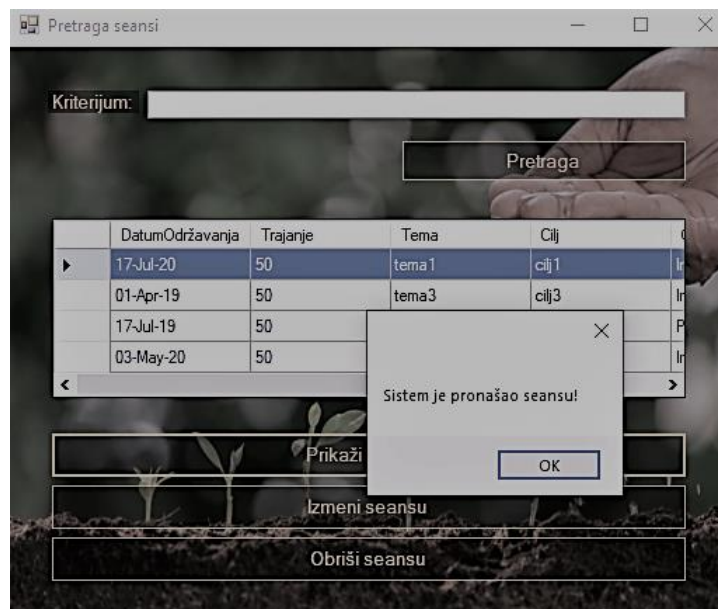
16. **Систем** тражи **сеансе** по задатој вредности. (СО)

17. **Систем** приказује **психотерапеуту** листу **сеанси** и поруку: "Sistem je pronašao seanse". (ИА)



Слика 120 - Обавештење о успешном проналаску

18. **Психотерапеут** бира сеансу чије податке жели да измени. (АПУСО)
19. **Психотерапеут** позива систем да учита податке о одабраној сеанси. (АПСО)
20. **Систем** учитава податке о одабраној сеанси. (СО)
21. **Систем** приказује психотерапеуту податке о сеанси и поруку: "Sistem je pronašao seansu". (ИА)



Слика 121 - Обавештење о успешном проналаску података о сеанси

Prikaz Seanse

ID: 1

Trajanje seanse: 50

Datum(format dd.MM.yyyy): 17.07.2020

REBT

Tema: tema1

Generalni cilj: cilj1

Uradjen domaci: da

Pohadjanje radionica: da

Napredak: veliki

Oblik psihoterapije: Individualna terapija

Unos stavke:

Klijent: -Izaberi--

Potvrdi izmene

☐ Bio/la na seansi

Cena seanse:

☐ Placena seansa

Unesi stavku

Izmeni odabranu stavku

Obrisi odabranu stavku

Rbr	Klijent	Prisustvo	Cena seansa	Placena seansa	Status
1	Bojana ...	<input checked="" type="checkbox"/>	1500	<input type="checkbox"/>	NijeMe...

Слика 122 - Приказ одабране сеансе

22. **Психотерапеут** мења податке о сеанси. (АПУСО)

Prikaz Seanse

ID: 1

Trajanje seanse: 60

Datum(format dd.MM.yyyy): 17.07.2020

REBT

Tema: tema1

Generalni cilj: cilj1

Uradjen domaci: da

Pohadjanje radionica: da

Napredak: veliki

Oblik psihoterapije: Individualna terapija

Unos stavke:

Klijent:

Potvrdi izmene

☐ Bio/la na seansi

Cena seanse:

☐ Placena seansa

Unesi stavku

Izmeni odabranu stavku

Obrisi odabranu stavku

Rbr	Klijent	Prisustvo	Cena seansa	Placena seansa	Status
1	Bojana ...	<input checked="" type="checkbox"/>	2000	<input type="checkbox"/>	Izmenje...

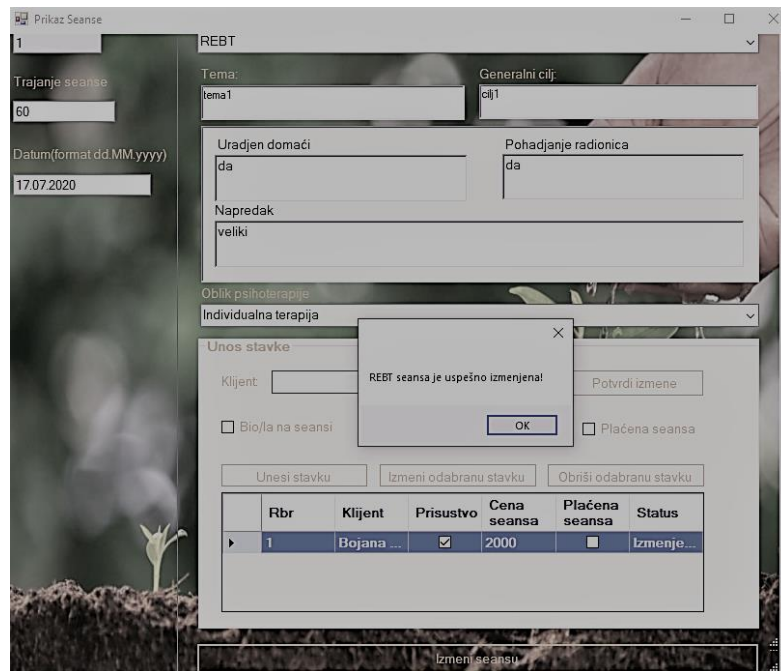
Слика 123 - Измена података одабране сеансе

23. **Психотерапеут** контролише да ли је коректно унео податке о сеанси. (АНСО)

24. **Психотерапеут** позива систем да запамти податке о сеанси. (АПСО)

25. Систем памти податке о сеанси. (CO)

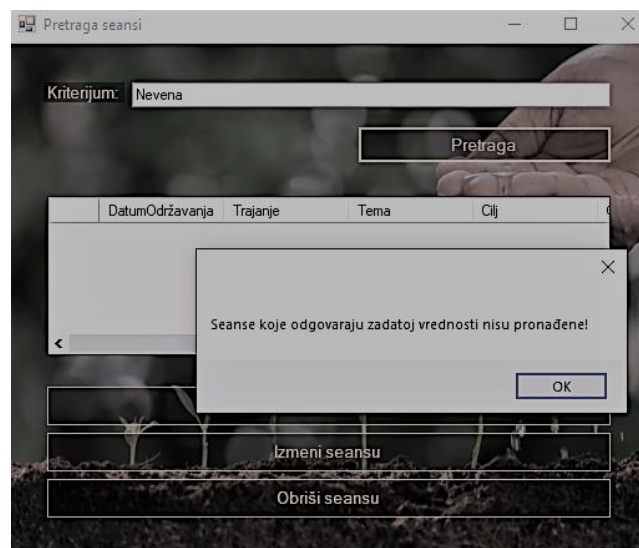
26. Систем приказује психотерапеуту поруку: “Seansa uspešno izmenjena!” (IA)



Слика 124 - Обавештење о успешној измени података

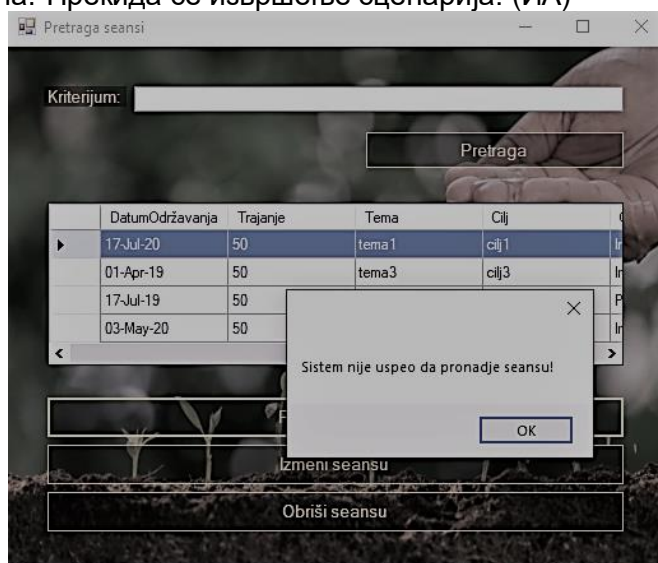
Алтернативна сценарија

4.1. Уколико систем не може да нађе сеансе он приказује психотерапеуту поруку: “Seanse koje odgovaraju zadatoj vrednosti nisu pronađene!”. Прекида се извршење сценарија. (IA)



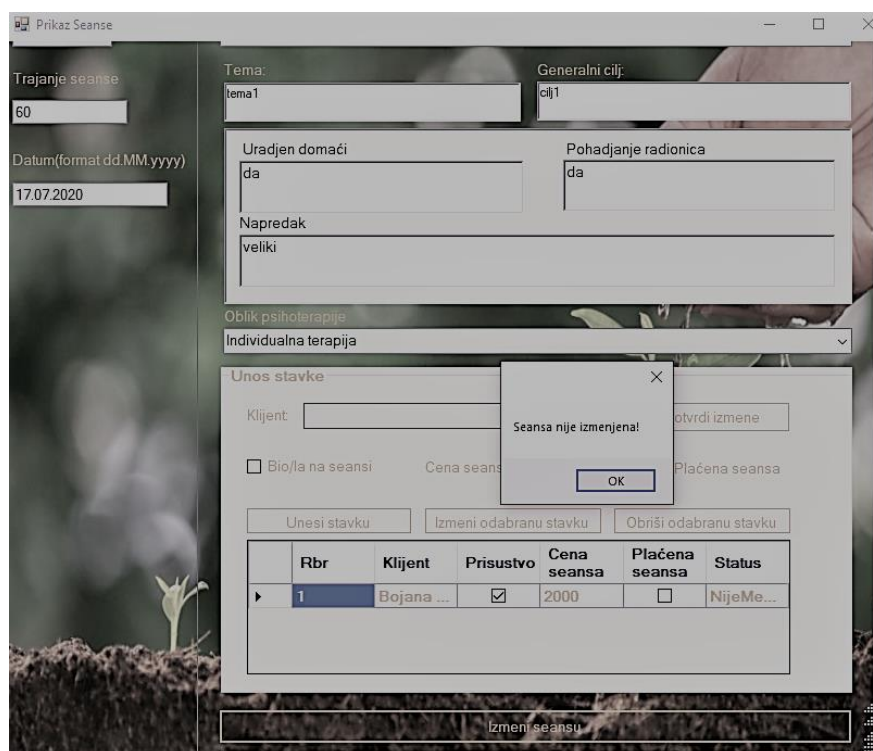
Слика 125 - Обавештење о неуспешном проналаску

8.1 Уколико **систем** не може да пронађе **сеансу** он приказује **психотерапеуту** поруку: “Seansa nije pronadena!” Прекида се извршење сценарија. (ИА)



Слика 126 - Обавештење о неуспешном проналаску података сеансе

13.1. Уколико **систем** не може да запамти податке о **сеанси** он приказује **психотерапеуту** поруку: “Seansa nije izmenjena!” (ИА)



Слика 127 - Обавештење о неуспешној измени података

СК9: Брисање сеансе

Назив СК

Брисање **сеансе**

Актери СК

Психотерапеут

Учесници СК

Психотерапеут и **систем** (програм)

Предуслов: **Систем** је укључен и **психотерапеут** је улогован под својом шифром. **Систем** приказује форму за рад са **сеансама**. Учитана је листа **сеанси**.



	Datum Održavanja	Trajanje	Tema	Cilj
►	17-Jul-20	50	tema1	cilj1
	01-Apr-19	50	tema3	cilj3
	17-Jul-19	50	tema2	cilj2
	03-May-20	50	tema4	cilj4

Слика 128 - Приказ форме за претрагу сеанси

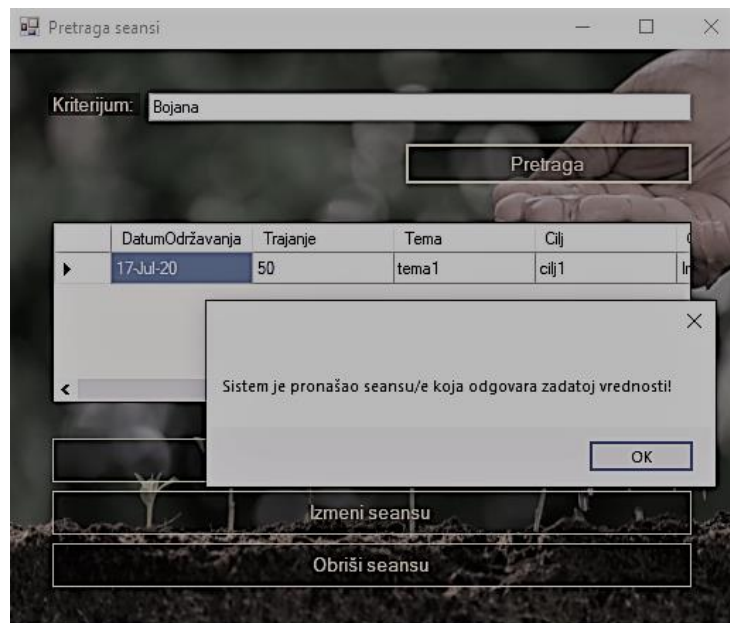
Основни сценарио СК

12. **Психотерапеут** уноси вредност по којој претражује **сеансе**. (АПУСО)



Слика 129 - Претрага према изабраном критеријуму

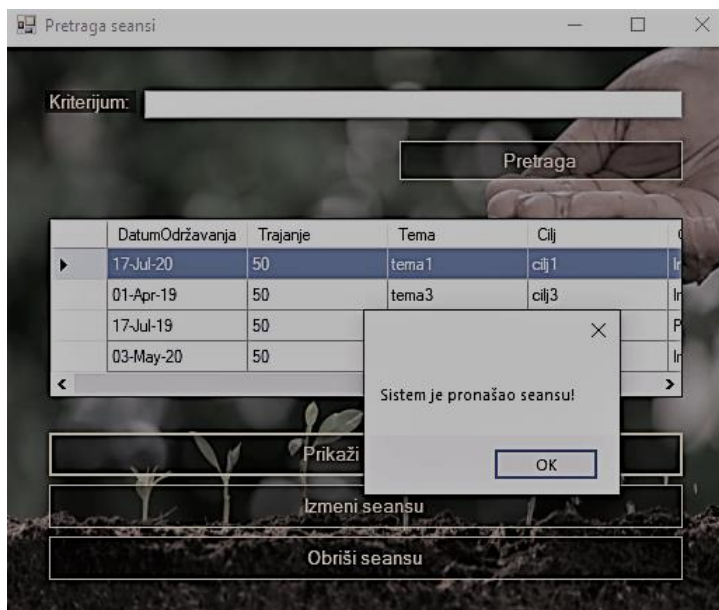
13. **Психотерапеут** **позива** **систем** да нађе **сеансе** по задатој вредности. (АПСО)
14. **Систем** **тражи** **сеансе** по задатој вредности. (СО)
15. **Систем** **приказује** **психотерапеуту** листу **сеанси** и поруку: "Sistem je pronašao seanse". (ИА)



Слика 130 - Обавештење о успешном проналаску

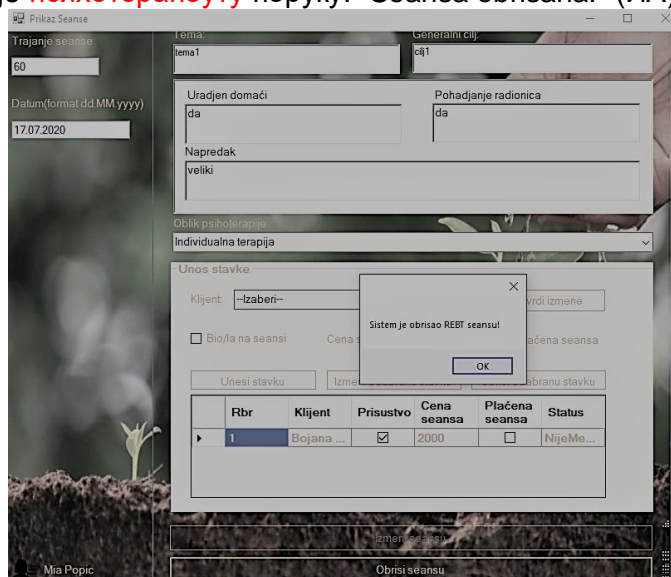
16. **Психотерапеут** **бира** **сеансу** чије податке жели да учита. (АПУСО)
17. **Психотерапеут** **позива** **систем** да учита податке о одабраној **сеанси**. (АПСО)
18. **Систем** **тражи** податке о одабраној **сеанси**. (СО)

19. Систем приказује психотерапеуту податке о сеанси и поруку: „Sistem je pronašao seansu“. (ИА)



Слика 131- Обавештење о успешном проналаску података сеансе

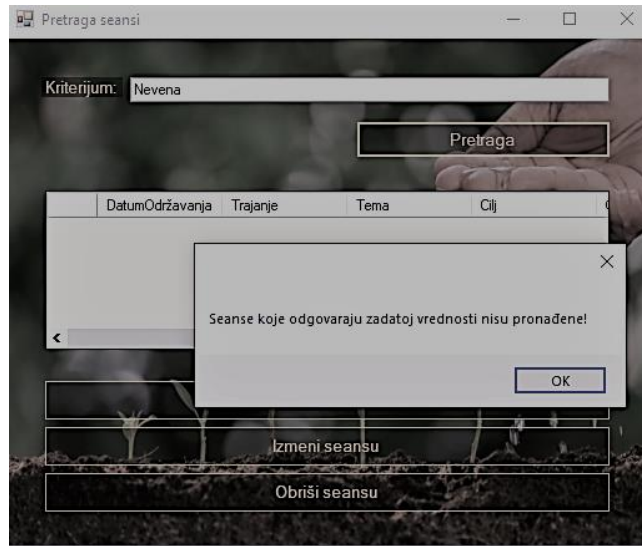
20. Психотерапеут позива систем да обрише одабрану сеансу. (АПСО)
 21. Систем брише сеансу. (СО)
 22. Систем приказује психотерапеуту поруку: “Seansa obrisana!” (ИА)



Слика 132 - Обавештење о успешном брисању сеансе

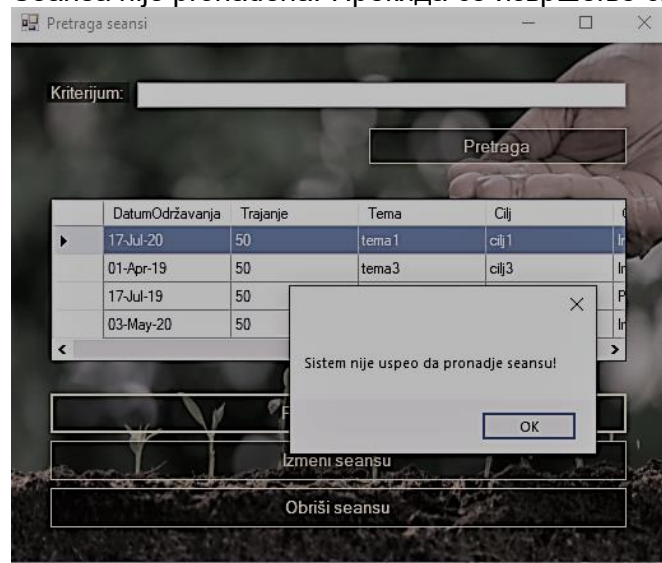
Алтернативна сценарија

4.1. Уколико систем не може да нађе сеансе он приказује психотерапеуту поруку: “Seanse koje odgovaraju zadatoj vrednosti nisu pronađene!”. Прекида се извршење сценарија. (ИА)



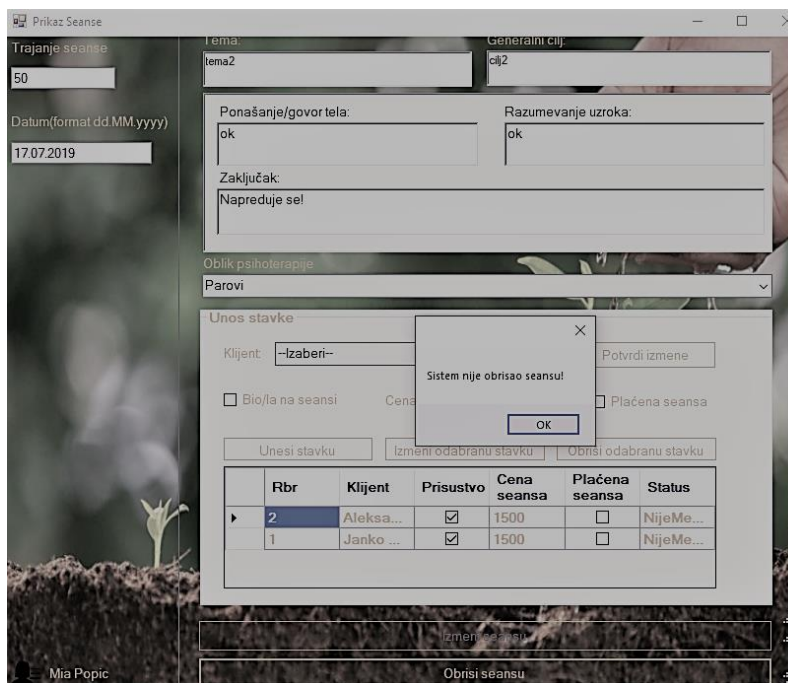
Слика 133 - Обавештење о неуспешном проналаску

8.1 Уколико **систем** не може да пронађе податке о **сеанси** он приказује **психотерапеуту** поруку: "Seansa nije pronađena!" Прекида се извршење сценарија. (ИА)



Слика 134 - Обавештење о неуспешном проналаску података сеансе

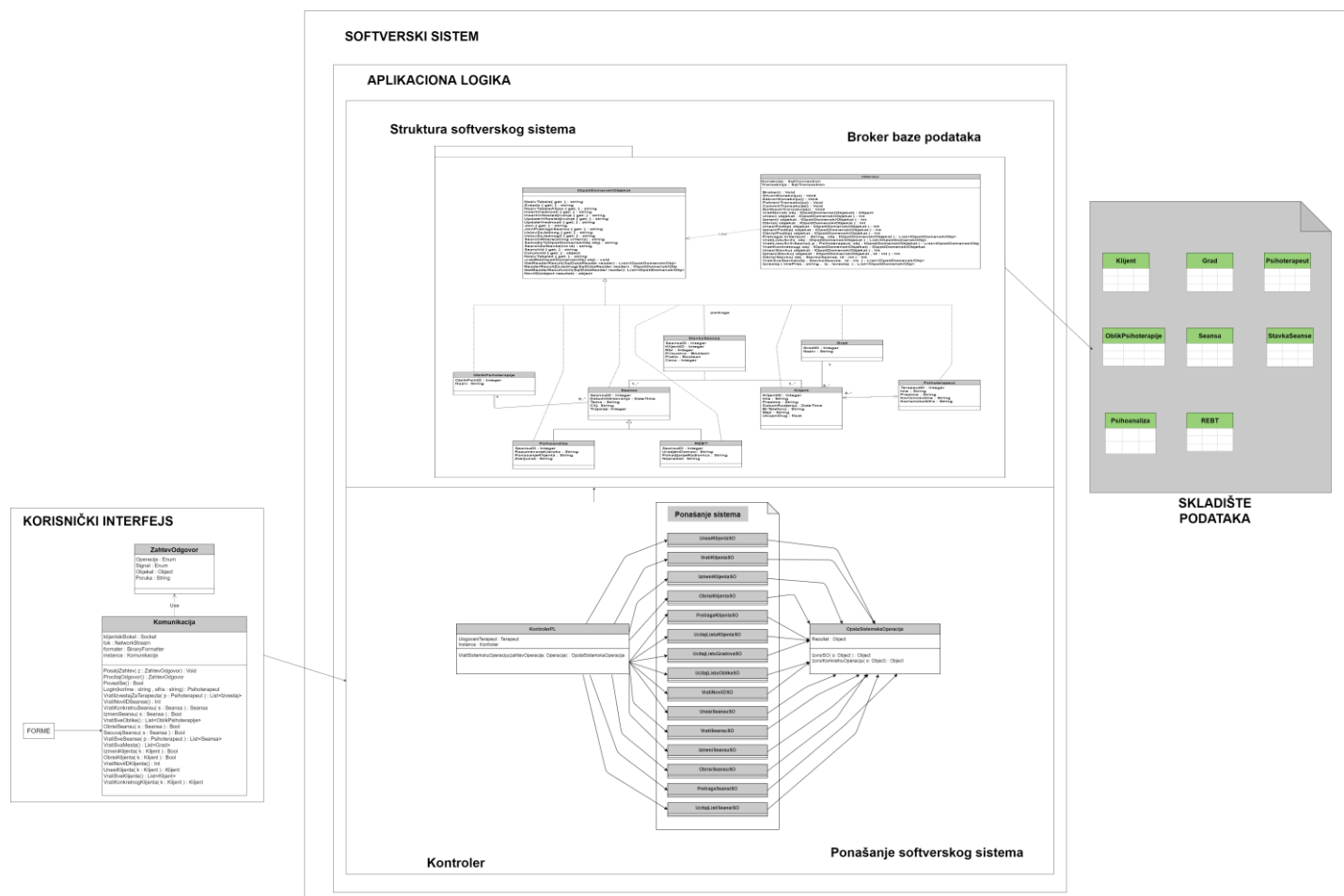
10.1. Уколико **систем** не може да обрише **сеансу** он приказује **психотерапеуту** поруку: "Seansa nije obrisana!". (ИА)



Слика 135 - Обавештење о успешном брисању

Комплетна архитектура софтверског система

Након завршене фазе пројектовања добијена је следећа архитектура софтверског система:



Слика 136 - Архитектура софтверског система

10. Фаза имплементација

Имплементација овог софтверског система рађена је у C# програмском језику, у .NET окружењу. Систем је пројектован као клијент-сервер. Као систем за управљање базом података коришћен је Microsoft SQL Server Management Studio 2017, док је развојно окружење MS Visual Studio 2017.

10.1. Структура софтверског решења

Систем је реализован у седам пројеката у оквиру solution-a Psihološko savetovalište – Domen, KorisničkiInterfejs, BrokerBazePodataka, Server, Kontroler, SistemskeOperacije и UnitTestovi.

Solution Psihološko savetovalište:

1. Domen

- Grad.cs
- Klijent.cs
- OblikPsihoterapije.cs
- Izvestaj.cs
- IOpstiDomenskiObjekat.cs
- Psihoanaliza.cs
- Psihoterapeut.cs
- REBT.cs
- Seansa.cs
- StavkaSeanse.cs
- ZahtevOdgovor.cs

2. KorisničkiInterfejs

- FrmGlavna.cs
- FrmLogin.cs
- Izvestaj.cs
- PretragaKlijenata.cs
- PretragaSeansi.cs
- PrikazKlijenta.cs
- PrikazSeanse.cs
- UnosKlijenta.cs
- UnosSeanse.cs
- UnosSeansePsihoanaliza.cs
- UnosSeanseREBT.cs
- Komunikacija
- KontrolerKI
- Sesija

3. BrokerBazePodataka

- Broker.cs

4. Server

- ObradaKlijenta.cs
- Server.cs
- FormaServer.cs

5. Kontroler

- KontrolerPL.cs

6. SistemskeOperacije

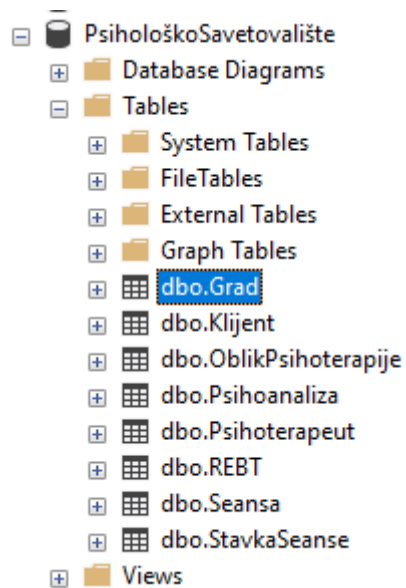
- UcitajListuGradovaSO.cs
- UnesiKlijentaSO.cs
- IzmeniKlijentaSO.cs
- ObrisiKlintaSO.cs
- VратиKlijentaSO.cs
- PretragaKlijenataSO.cs
- UcitajListuKlijenataSO.cs
- VратиNoviIDSO.cs
- UcitajListuOblikaSO.cs
- UnesiSeansuSO.cs
- IzmeniSeansuSO.cs
- ObrisiSeansuSO.cs
- VратиSeansuSO.cs
- PretragaSeansiSO.cs
- UcitajListuSeansiSO.cs
- IzvestajSO.cs
- LoginSO.cs
- OpstaSistemskaOperacija.cs

7. UnitTestovi

- KlijentUnitTests.cs
- GradoviUnitTest.cs
- KSIDUnitTest.cs
- OblikTerapijeUnitTest.cs
- SeanseUnitTests
- TerapeutUnitTest

10.2. Имплементација складишта података

На основу структуре софтверских класа пројектоване су табеле (складишта података) релационог система са управљање базом података. У овом раду коришћен је Microsoft SQL Server Management Studio 2017.



Слика 137 – База података – Психолошко саветовалиште

DESKTOP-N2D8V2E\S...valište - dbo.Grad			
	Column Name	Data Type	Allow Nulls
▶	GradID	int	<input type="checkbox"/>
	Naziv	nvarchar(50)	<input type="checkbox"/>

Слика 138 - Табела Град

DESKTOP-N2D8V2E\S...ište - dbo.Klijent			
	Column Name	Data Type	Allow Nulls
▶	KlijentID	int	<input type="checkbox"/>
	Ime	nvarchar(50)	<input type="checkbox"/>
	Prezime	nvarchar(50)	<input type="checkbox"/>
	DatumRodjenja	date	<input checked="" type="checkbox"/>
	BrojTelefona	nvarchar(50)	<input type="checkbox"/>
	Mejl	nvarchar(50)	<input type="checkbox"/>
	UkupniDug	float	<input type="checkbox"/>
	GradID	int	<input type="checkbox"/>
	TerapeutID	int	<input type="checkbox"/>

Слика 139 - Табела Клијент

DESKTOP-N2D8V2E\S...blikPsihoterapije			
	Column Name	Data Type	Allow Nulls
▶	OblikPsihID	int	<input type="checkbox"/>
	Naziv	nvarchar(50)	<input type="checkbox"/>

Слика 140 - Табела ОбликПсихотерапије

DESKTOP-N2D8V2E\S...dbo.Psihoanaliza			
	Column Name	Data Type	Allow Nulls
▶	SeansaID	int	<input type="checkbox"/>
	PonašanjeKlijenta	nvarchar(300)	<input checked="" type="checkbox"/>
	RazumevanjeUzroka	nvarchar(300)	<input checked="" type="checkbox"/>
	Zaključak	nvarchar(300)	<input checked="" type="checkbox"/>
			<input type="checkbox"/>

Слика 141 -Табела Психоанализа

DESKTOP-N2D8V2E\S...valište - dbo.REBT			
	Column Name	Data Type	Allow Nulls
▶	SeansaID	int	<input type="checkbox"/>
	UradjenDomaći	nvarchar(300)	<input checked="" type="checkbox"/>
	PohadjanjeRadionice	nvarchar(300)	<input checked="" type="checkbox"/>
	Napredak	nvarchar(300)	<input checked="" type="checkbox"/>

Слика 142 - Табела РЕБТ

DESKTOP-N2D8V2E\S...lište - dbo.Seansa			
	Column Name	Data Type	Allow Nulls
▶	SeansaID	int	<input type="checkbox"/>
	DatumOdržavanja	date	<input type="checkbox"/>
	Trajanje	int	<input checked="" type="checkbox"/>
	Tema	nvarchar(100)	<input type="checkbox"/>
	Cilj	nvarchar(300)	<input type="checkbox"/>
	OblikID	int	<input type="checkbox"/>

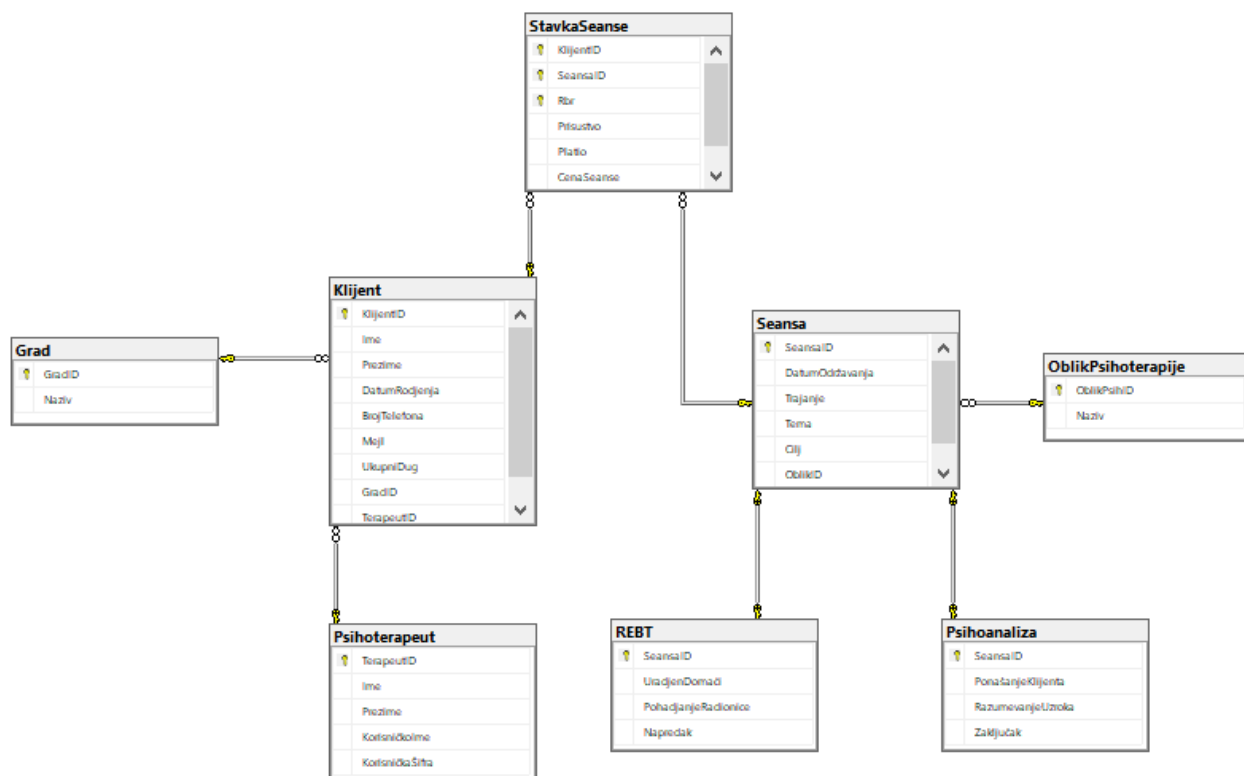
Слика 143 - Табела Сеанса

DESKTOP-N2D8V2E\S...dbo.StavkaSeanse			
	Column Name	Data Type	Allow Nulls
▶	KlijentID	int	<input type="checkbox"/>
▶	SeansaID	int	<input type="checkbox"/>
▶	Rbr	int	<input type="checkbox"/>
	Prisustvo	bit	<input type="checkbox"/>
	Platio	bit	<input type="checkbox"/>
	CenaSeanse	int	<input type="checkbox"/>

Слика 144 - Табела СтавкаСеансе

DESKTOP-N2D8V2E\S...dbo.Psihoterapeut			
	Column Name	Data Type	Allow Nulls
▶	TerapeutID	int	<input type="checkbox"/>
	Ime	nvarchar(50)	<input checked="" type="checkbox"/>
	Prezime	nvarchar(50)	<input checked="" type="checkbox"/>
	KorisničkoIme	nvarchar(50)	<input type="checkbox"/>
	KorisničkaŠifra	nvarchar(50)	<input type="checkbox"/>

Слика 145 - Табела Психотерапеут



Слика 146 - Приказ дијаграма повезаности табела релационе базе података

Имплементација апликационе логике

Комуникација са клијентима

У оквиру софтверксе класе **Server** имплментирана је метода *public void PokreniServer()*, која подиже серверски сокет који ће да ослушкује мрежу.

Када клијент (клијентски сокет) успостави конекцију са серверским сокетом, тада контролер треба да генерише нит која ће успоставити двосмерну везу са клијентом (улазну и излазну). Слање и примање података од клијента се остварује преко сокета.

Клијент шаље захтев за извршење неке од СО до одговарајуће нити (коју смо назвали “ОбрадаКлијента”), која је повезана са тим клијентом. Нит клијената прима захтев и даље га преусмерава до класа које су одговорне за извршење СО. Након извршења СО резултат се враћа до апликационе логике, односно до нити клијента, која тај резултат шаље назад до клијента.

```
public class Server
{
    private Socket osluskujuciSoket;

    public List<Psihoterapeut> KorisniciSistema { get; internal set; } = new
List<Psihoterapeut>();

    public delegate void PrijavljenNovEventHandler();
    public event PrijavljenNovEventHandler PrijavljenNov;

    public bool PokreniServer()
    {
        try
        {
            osluskujuciSoket = new
Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);
            osluskujuciSoket.Bind(new IPEndPoint(IPAddress.Parse("127.0.0.1"),
8888));
            osluskujuciSoket.Listen(5);

            return true;
        }
        catch (SocketException)
        {
            throw;
        }
    }

    public void Osluskuj()
    {
        bool kraj = false;
        while (!kraj)
        {
            Socket klijentskiSoket = osluskujuciSoket.Accept();
            Obrada obrada = new Obrada(klijentskiSoket, this);
            Thread nitKlijenta = new Thread(obrada.ObradaZahteva);
```

```

        nitKlijenta.Start();
    }
}

internal void OnPrijavljen()
{
    PrijavljenNov?.Invoke();
}
}

public class ObradaKlijenta
{
    private Socket klijentskiSoket;
    private readonly Server s;
    private NetworkStream tok;
    private BinaryFormatter formater = new BinaryFormatter();

    public Obrada(Socket klijentskiSoket , Server s)
    {
        this.klijentskiSoket = klijentskiSoket;
        this.s = s;
        tok = new NetworkStream(klijentskiSoket);
    }

    public void ObradaZahteva()
    {
        bool kraj = false;
        while (!kraj)
        {
            try
            {
                ZahtevOdgovor z = (ZahtevOdgovor)formater.Deserialize(tok);
                ZahtevOdgovor o = IzvrsiZahtev(z);
                formater.Serialize(tok, o);
            }
            catch (Exception ex)
            {
                Debug.WriteLine($">>> {ex.Message}");
                kraj = true;
            }
        }
    }

    public ZahtevOdgovor IzvrsiZahtev(ZahtevOdgovor z)
    {
        ZahtevOdgovor o = new ZahtevOdgovor();
        try
        {
            switch (z.Operacija)
            {
                case Operacija.Login:
                    o.Objekat
                    =
                    KontrolerPL.Instanca.Login( (Psihoterapeut)z.Objekat);
                    o.Poruka = "Sistem je uspešno pronašao terapeuta!";
            }
        }
    }
}

```

```

        s.KorisniciSistema.Add(o.Objekat as Psihoterapeut);
        s.OnPrijavljen();
        break;
    case Operacija.VratiNoviIDS:
        o.Objekat
        KontrolerPL.Instanca.VratiNoviIDSeanse((Seansa)z.Objekat);
        o.Poruka = "Sistem je uspešno izgenerisao id!";
        break;
    case Operacija.VratiNoviIDK:
        o.Objekat
        KontrolerPL.Instanca.VratiNoviIDKlijenta((Klijent)z.Objekat);
        o.Poruka = "Sistem je uspešno izgenerisao id!";
        break;
    case Operacija.VratiSveKlijente:
        o.Objekat
        KontrolerPL.Instanca.VratiSveKlijente(z.Objekat as Klijent);
        o.Poruka = "Sistem je uspešno pronašao klijente!";
        break;
    case Operacija.VratiKonkretnogKlijenta:
        o.Objekat
        KontrolerPL.Instanca.VratiKonkretnogKlijenta((Klijent)z.Objekat);
        o.Poruka = "Sistem je uspešno pronašao klijenta!";
        break;
    case Operacija.IzmeniKlijenta:
        o.Objekat
        KontrolerPL.Instanca.IzmeniKlijenta((Klijent)z.Objekat);
        o.Poruka = "Sistem je uspešno izmenio klijenta!";
        break;
    case Operacija.ObrisiKlijenta:
        o.Objekat
        KontrolerPL.Instanca.ObrisiKlijenta((Klijent)z.Objekat);
        o.Poruka = "Sistem je uspešno obrisao klijenta!";
        break;
    case Operacija.PretraziKlijente:
        o.Objekat
        KontrolerPL.Instanca.PretragaKlijenata(z.Objekat.ToString());
        o.Poruka = "Sistem je uspešno pretražio klijente!";
        break;
    case Operacija.VratiSvaMesta:
        o.Objekat
        KontrolerPL.Instanca.VratiSvaMesta((Grad)z.Objekat);
        o.Poruka = "Sistem je uspešno pronašao mesta!";
        break;

    case Operacija.VratiSveOblike:
        o.Objekat
        KontrolerPL.Instanca.VratiSveOblike((OblikPsihoterapije)z.Objekat);
        o.Poruka = "Sistem je uspešno pronašao oblike!";
        break;
    case Operacija.UnesiKlijenta:
        o.Objekat
        KontrolerPL.Instanca.UnesiKlijenta((Klijent)z.Objekat);
        o.Poruka = "Sistem je uspešno sačuvao klijenta!";
        break;
    case Operacija.SacuvajSeansu:
        o.Objekat = KontrolerPL.Instanca.UnesiSeansu(z.Objekat as
Seansa);

        o.Poruka = "Sistem je uspešno sačuvao seansu!";
        break;

```



```

        case Operacija.PretraziSeanse:
            o.Objekat =
KontrolerPL.Instanca.PretragaSeansi(z.Objekat.ToString());
            o.Poruka = "Sistem je uspešno pretražio seanse!";
            break;
        case Operacija.VratiSveSeanse:
            o.Objekat = KontrolerPL.Instanca.VratiSveSeanse(z.Objekat
as Seansa);
            o.Poruka = "Sistem je uspešno vratio seanse!";
            break;
        case Operacija.VratiKonkretnuSeansu:
            o.Objekat =
KontrolerPL.Instanca.VratiKonkretnuSeansu(z.Objekat as Seansa);
            o.Poruka = "Sistem je uspešno pronasao seansu!";
            break;
        case Operacija.IzmeniSeansu:
            o.Objekat =
KontrolerPL.Instanca.IzmeniSeansu((Seansa)z.Objekat);
            o.Poruka = "Sistem je uspešno izmenio seansu!";
            break;

        case Operacija.ObrisiSeansu:
            o.Objekat =
KontrolerPL.Instanca.ObrisiSeansu((Seansa)z.Objekat);
            o.Poruka = "Sistem je uspešno obrisao seansu!";
            break;

        default:
            throw new Exception("Greska");
            //break;
    }
}
catch (Exception e)
{
    o.Signal = Signal.Error;
    o.Poruka = e.Message;
}
return o;
}
}

```

Контролер апликационе логике

Контролер апликационе логике прихвата захтеве за извршење системских операција и исте прослеђује до конкретне системске операције. Након извршења системске операције, контролер прихвата одговор и враћа назад позиваоцу (нити клијента).

```

namespace Kontroler
{
    public class KontrolerPL
    {
        private Broker broker = new Broker();
    }
}

```

```

#region Singleton
static KontrolerPL instanca;

KontrolerPL()
{

}

public static KontrolerPL Instanca
{
    get
    {
        if (instancja == null)
            instanca = new KontrolerPL();
        return instanca;
    }
}
#endregion

public Psihoterapeut Login(Psihoterapeut p)
{
    OpstaSistemskaOperacija op = new LoginSO();
    op.Izvrsi(p);
    return ((LoginSO)op).p;
}

//klijent

public List<Klijent> VratiSveKlijente(Klijent k)
{
    OpstaSistemskaOperacija op = new UcitajListuKlijenataSO();
    op.Izvrsi(k);
    return ((UcitajListuKlijenataSO)op).ListaKl;
}

public Klijent UnesiKlijenta(Klijent k)
{
    OpstaSistemskaOperacija sistemskaOperacija = new UnesiKlijentaSO();
    sistemskaOperacija.Izvrsi(k);
    return ((UnesiKlijentaSO)sistemskaOperacija).Klijent;
}

public bool IzmeniKlijenta(Klijent k)
{
    OpstaSistemskaOperacija operacija = new IzmeniKlijentaSO();
    operacija.Izvrsi(k);
    return ((IzmeniKlijentaSO)operacija).Izmenjen;
}

public bool ObrisiKlijenta(Klijent k)
{
    OpstaSistemskaOperacija operacija = new ObrisiKlijentaSO();
    operacija.Izvrsi(k);
    return ((ObrisiKlijentaSO)operacija).Obrisan;
}

public Klijent VratiKonkretnogKlijenta(Klijent k)

```

```

{
    OpstaSistemskaOperacija operacija = new VratiklijentaSO();
    operacija.Izvrsi(k);
    return ((VratiklijentaSO)operacija).Klijent;
}
public List<Klijent> PretragaKlijenata(string krit)
{
    OpstaSistemskaOperacija operacija = new PretragaKlijenataSO();
    operacija.Izvrsi(krit);
    return ((PretragaKlijenataSO)operacija).rez;
}

public List<Grad> VratISvaMesta(Grad g)
{
    OpstaSistemskaOperacija op = new UcitajListuGradovaSO();
    op.Izvrsi(g);
    return ((UcitajListuGradovaSO)op).ListaG;
}

public int VratINoviID(IOpstiDomenskiObj o)
{
    OpstaSistemskaOperacija op = new VratINoviIDSO();
    op.Izvrsi(o);
    return ((VratINoviIDSO)op).sifra;
}

//seansa

public bool UnesiSeansu(Seansa s)
{
    OpstaSistemskaOperacija operacija = new UnesiSeansuSO();
    operacija.Izvrsi(s);
    return ((UnesiSeansuSO)operacija).Sacuvana;
}

public List<Seansa> VratISveSeanse(Psihoterapeut p)
{
    OpstaSistemskaOperacija operacija = new UcitajListuSeansiSO();
    operacija.Izvrsi(p);
    return ((UcitajListuSeansiSO)operacija).rez;
}

public List<Seansa> PretragaSeansi(string krit)
{
    OpstaSistemskaOperacija operacija = new PretragaSeansiSO();
    operacija.Izvrsi(krit);
    return ((PretragaSeansiSO)operacija).rez;
}

public List<OblikPsihoterapije> VratISveOblike(OblikPsihoterapije oblik)
{
    OpstaSistemskaOperacija op = new UcitajListuOblikaSO();
    op.Izvrsi(oblik);
    return ((UcitajListuOblikaSO)op).listaOP;
}

```

```

    }

    public Seansa VратиKonkretnuSeansu(Seansa s)
    {
        OpstaSistemskaOperacija operacija = new VратиSeansuSO();
        operacija.Izvrsi(s);
        return ((VратиSeansuSO)operacija).seansa;
    }

    public bool IzmeniSeansu(Seansa s)
    {
        OpstaSistemskaOperacija operacija = new IzmeniSeansuSO();
        operacija.Izvrsi(s);
        return ((IzmeniSeansuSO)operacija).izmenjena;
    }

    public bool ObrisiSeansu(Seansa s)
    {
        OpstaSistemskaOperacija operacija = new ObrisiSeansuSO();
        operacija.Izvrsi(s);
        return ((ObrisiSeansuSO)operacija).obrisana;
    }

    public List<Izvestaj> VратиIzvestajZaT(Psihoterapeut psihoterapeut)
    {
        OpstaSistemskaOperacija operacija = new izvestajSO();
        operacija.Izvrsi(psihoterapeut);
        return ((izvestajSO)operacija).lista;
    }
}

```

Пословна логика

Пословна логика – доменске класе

На основу концептуалних класа праве се софтверске класе структуре система. Свака класа садржи приватна поља атрибута, модификаторе приступа за те атрибуте и конструкторе (непараметарске и параметарске).

Доменске класе имплементирају интерфејс *IOpstiDomenskiObjekat*, како би се омогућило лакше имплементирање метода брокера базе података. Брокер базе података прима интерфејс уместо самих класа, што олакшава креирање генеричких упита.

У наставку је дат код свих софтверских класа.

```

[Serializable]
public class Klient : IOpstiDomenskiObj
{
    public override string ToString()
    {
        return Ime+" "+Prezime;
    }

    [Browsable(false)]

```

```

public int KlientID { get; set; }
[Browsable(false)]
public string Ime { get; set; }
[Browsable(false)]
public string Prezime { get; set; }

[DisplayName("Ime i prezime")]
public string ImePrez { get { return Ime + " " + Prezime; } }

[DisplayName("Datum rođenja")]
public DateTime DatumRodjenja { get; set; }

[Browsable(false)]
public string BrojTelefona { get; set; }
[Browsable(false)]
public string Mejl { get; set; }
[Browsable(false)]
public double UkupanDug { get; set; }

public Grad Grad { get; set; }
public Psihoterapeut Psihoterapeut { get; set; }

[Browsable(false)]
public string NazivTabele => "Klijent";
[Browsable(false)]
public string NazivTabele2 => "";
[Browsable(false)]
public string InsertVrednosti =>
"${KlientID},{Ime},{Prezime},{DatumRodjenja},{BrojTelefona},{Mejl},{Uku
panDug},{Grad.GradID},{Psihoterapeut.TerapeutID}";
[Browsable(false)]
public string UpdateVrednosti => $"KlijentID = {KlientID}, Ime = '{Ime}',
Prezime = '{Prezime}', DatumRodjenja = '{DatumRodjenja}',
BrojTelefona='{BrojTelefona}', Mejl = '{Mejl}', UkupniDug = {UkupanDug}, GradID =
{Grad.GradID}, TerapeutID={Psihoterapeut.TerapeutID}";
[Browsable(false)]
public string Join => "join Grad g on k.GradID = g.GradID join
Psihoterapeut p on k.TerapeutID = p.TerapeutID ";

public string SearchWhere(string criteria)
{
    return $"where k.Ime like '{criteria}%' or k.Prezime like
'{criteria}%' or p.Ime LIKE '{criteria}%' or p.Prezime LIKE '{criteria}%' or
k.Mejl LIKE '{criteria}%'";
}

[Browsable(false)]
public string SearchId => $"where {ColumnId} = {KlientID}";
[Browsable(false)]
public object ColumnId => "KlijentID";

[Browsable(false)]
public string UslovZaJednog { get { return $"where k.{ColumnId} =
{KlientID}"; } }

[Browsable(false)]
public string UslovZaJednog2 => "";

[Browsable(false)]

```

```

public string InsertVrNasledjivanje => "";

[Browsable(false)]
public string NazivTabeleAlijas => "Klijent k";

[Browsable(false)]
public string UpdateVrNasledjivanje => "";

[Browsable(false)]
public string JoinPretragaSeansa => "join Grad g on k.GradID = g.GradID
join Psihoterapeut p on k.TerapeutID = p.TerapeutID";

[Browsable(false)]
public string Zvezda => "*";

public List<IOpstiDomenskiObj> GetReaderResult(SqlDataReader citac)
{
    List<IOpstiDomenskiObj> lista = new List<IOpstiDomenskiObj>();
    while (citac.Read())
    {
        Klijent k = new Klijent
        {
            KlijentID = citac.GetInt32(0),
            Ime = citac.GetString(1),
            Prezime = citac.GetString(2),
            DatumRodjenja = citac.GetDateTime(3),
            BrojTelefona = citac.GetString(4),
            Mejl = citac.GetString(5),
            UkupanDug = citac.GetDouble(6),

            Grad = new Grad
            { GradID = citac.GetInt32(9), Naziv = citac.GetString(10) },

            Psihoterapeut = new Psihoterapeut
            { TerapeutID = citac.GetInt32(11) , Ime = citac.GetString(12),
            Prezime = citac.GetString(13) }

        };
        lista.Add(k);
    }
    return lista;
}

public IOpstiDomenskiObj ReaderResultZaJednog(SqlDataReader reader)
{
    if (reader.Read())
    {
        Klijent k = new Klijent
        {
            KlijentID = (int)reader["KlijentID"],
            Ime = (string)reader["Ime"],
            Prezime = (string)reader["Prezime"],
            DatumRodjenja = DateTime.Parse(reader["DatumRodjenja"].ToString()),
            BrojTelefona = (string)reader["BrojTelefona"],
            Mejl = (string)reader["Mejl"],
            UkupanDug = (double)reader["UkupniDug"],

            Grad = new Grad

```

```

        { GradID = (int)reader["GradID"] },

        Psihoterapeut = new Psihoterapeut
        { TerapeutID = (int)reader["TerapeutID"] }
    };

    return k;
}
return null;
}

public object NoviID(object rezultat)
{
    if (rezultat is DBNull)
    {
        return 1;
    }
    int maxid = (int)rezultat;
    return maxid + 1;
}

public void vratiBit(IOpstiDomenskiObj obj)
{
    throw new NotImplementedException();
}

public string SearchZaStavke(int id)
{
    throw new NotImplementedException();
}

public List<IOpstiDomenskiObj> GetReaderResultJoin(SqlDataReader citac)
{
    List<IOpstiDomenskiObj> lista = new List<IOpstiDomenskiObj>();
    while (citac.Read())
    {
        Klient k = new Klient
        {
            KlientID = citac.GetInt32(0),
            Ime = citac.GetString(1),
            Prezime = citac.GetString(2),
            DatumRodjenja = citac.GetDateTime(3),
            BrojTelefona = citac.GetString(4),
            Mejl = citac.GetString(5),
            UkupanDug = citac.GetDouble(6),

            Grad = new Grad
            { GradID = citac.GetInt32(9), Naziv = citac.GetString(10) },

            Psihoterapeut = new Psihoterapeut
            { TerapeutID = citac.GetInt32(11), Ime = citac.GetString(12),
            Prezime = citac.GetString(13) }
        };
        lista.Add(k);
    }
    return lista;
}

```

```

[Serializable]
public class Grad : IOpstiDomenskiObj
{
    public int GradID { get; set; }
    public string Naziv { get; set; }

    public override string ToString()
    {
        return Naziv;
    }

    public override bool Equals(object obj)
    {
        if (obj is Grad g)
        {
            return g.GradID == GradID;
        }
        return false; ;
    } //zbog cmb-a

    public string NazivTabele => "Grad";

    public string InsertVrednosti => $"{GradID},{Naziv}";

    public string UpdateVrednosti => $"GradID = {GradID}, Naziv = '{Naziv}'";

    public string Join => "";

    public string SearchId => "";

    public object ColumnId => "GradID";

    public string UslovZaJednog => $"GradID = {GradID}";

    public string NazivTabele2 => "";

    public string UslovZaJednog2 => "";

    public string InsertVrNasledjivanje => "";

    public string NazivTabeleAlijas => "Grad g";

    public string UpdateVrNasledjivanje => "";

    public string JoinPretragaSeansa => throw new NotImplementedException();

    public string Zvezda => throw new NotImplementedException();

    public List<IOpstiDomenskiObj> GetReaderResult(SqlDataReader reader)
    {
        List<IOpstiDomenskiObj> lista = new List<IOpstiDomenskiObj>();
        while (reader.Read())
        {
            Grad g = new Grad
            {
                GradID = (int)reader["GradID"],
                Naziv = (string)reader["Naziv"]
            }
        }
    }
}

```



```

        };

        lista.Add(g);
    }
    return lista;
}

public string SearchWhere(string criteria)
{
    return "";
}

public IOpstiDomenskiObj ReaderResultZaJednog(SqlDataReader reader)
{
    if (reader.Read())
    {
        Grad g = new Grad
        {
            GradID = (int)reader["GradID"],
            Naziv = (string)reader["Naziv"]
        };
        return g;
    }
    return null;
}

public object NoviID(object rezultat)
{
    throw new NotImplementedException();
}

public void vratiBit(IOpstiDomenskiObj obj)
{
    throw new NotImplementedException();
}

public string SearchZaStavke(int id)
{
    throw new NotImplementedException();
}

public List<IOpstiDomenskiObj> GetReaderResultJoin(SqlDataReader reader)
{
    throw new NotImplementedException();
}
}

[Serializable]
public class Psihoanaliza : Seansa , IOpstiDomenskiObj
{
    [Browsable(false)]
    public string PonašanjeKlijenta { get; set; }
    [Browsable(false)]
    public string RazumevanjeUzroka { get; set; }
    [Browsable(false)]

```

```

public string Zaključak { get; set; }

[Browsable(false)]
public new string NazivTabele => "Psihoanaliza";
[Browsable(false)]
public new string NazivTabele2 => "Seansa ";
[Browsable(false)]
public new string InsertVrNasledjivanje =>
$" {SeansaID}, '{DatumOdržavanja}', '{Trajanje}', '{Tema}', '{Cilj}', '{Oblik.OblikPsihID}'";
;
[Browsable(false)]
public new string InsertVrednosti =>
$" {SeansaID}, '{PonašanjeKlijenta}', '{RazumevanjeUzroka}', '{Zaključak}'";
[Browsable(false)]
public new string UpdateVrednosti =>
$"SeansaID={SeansaID}, PonašanjeKlijenta='{PonašanjeKlijenta}', RazumevanjeUzroka='{RazumevanjeUzroka}', Zaključak='{Zaključak}'";
[Browsable(false)]
public new string UpdateVrNasledjivanje => $"SeansaID
={SeansaID}, DatumOdržavanja='{DatumOdržavanja}', Trajanje={Trajanje}, Tema='{Tema}',
Cilj='{Cilj}', OblikID = {Oblik.OblikPsihID}";
[Browsable(false)]
public new string NazivTabeleAlijas => "Psihoanaliza p";
[Browsable(false)]
public new string Join => "join Seansa s on p.SeansaID=s.SeansaID join
OblikPsihoterapije o on o.OblikPsihID=s.OblikID";
[Browsable(false)]
public new string SearchId => $"where {ColumnId}={SeansaID}";
[Browsable(false)]
public new object ColumnId => "SeansaID";
[Browsable(false)]
public new string UslovZaJednog => $"where p.{ColumnId}={SeansaID}";
[Browsable(false)]
public new string UslovZaJednog2 => "";
[Browsable(false)]
public new string JoinPretragaSeansa => $"join Seansa s on
p.SeansaID=s.SeansaID join OblikPsihoterapije o on o.OblikPsihID=s.OblikID join
StavkaSeanse st on s.SeansaID= st.SeansaID join Klijent k on
st.KlijentID=k.KlijentID join Psihoterapeut ps on k.TerapeutID=ps.TerapeutID";

[Browsable(false)]
public new string Zvezda =>
"s.SeansaID,s.DatumOdržavanja,s.Trajanje,s.Tema,s.Cilj,o.OblikPsihID,o.Naziv";

public new List<IOpstiDomenskiObj> GetReaderResult(SqlDataReader citac)
{
    List<IOpstiDomenskiObj> lista = new List<IOpstiDomenskiObj>();
    while (citac.Read())
    {
        Psihoanaliza r = new Psihoanaliza
        {
            SeansaID = citac.GetInt32(0),
            PonašanjeKlijenta = citac.GetString(1),
            RazumevanjeUzroka = citac.GetString(2),
            Zaključak = citac.GetString(3),
            DatumOdržavanja = citac.GetDateTime(5),
            Trajanje = Convert.ToDouble(citac.GetValue(6)),
            Tema = citac.GetString(7),

```

```

        Cilj = citac.GetString(8),

        Oblik = new OblikPsihoterapije
        {
            OblikPsihID = citac.GetInt32(10),    Naziv =
citac.GetString(11) }

        };
        lista.Add(r);
    }
    return lista;
}

public new List<IOpstiDomenskiObj> GetReaderResultJoin(SqlDataReader
citac)
{
    List<IOpstiDomenskiObj> lista = new List<IOpstiDomenskiObj>();
    while (citac.Read())
    {
        Psihoanaliza r = new Psihoanaliza();

        r.SeansaID = citac.GetInt32(0);

        r.DatumOdržavanja = citac.GetDateTime(1);
        r.Trajanje = Convert.ToDouble(citac.GetValue(2));
        r.Tema = citac.GetString(3);
        r.Cilj = citac.GetString(4);

        r.Oblik = new OblikPsihoterapije();
        r.Oblik.OblikPsihID = citac.GetInt32(5);
        r.Oblik.Naziv = citac.GetString(6);

        lista.Add(r);
    }
    return lista;
}

public new string SearchWhere(string criteria)
{
    try
    {
        if (int.TryParse(criteria, out int deo))
        {
            return $"where {deo} = DATEPART(MM,s.DatumOdržavanja) or {deo}
= DATEPART(yyyy,s.DatumOdržavanja) group by
s.SeansaID,s.DatumOdržavanja,s.Trajanje,s.Tema,s.Cilj,o.OblikPsihID,o.Naziv";
        }
        if (DateTime.TryParse(criteria, out DateTime datum))
        {
            return $"where cast('{datum.ToString("yyyy-MM-dd")}' as date)=
cast(s.DatumOdržavanja as date) group by
s.SeansaID,s.DatumOdržavanja,s.Trajanje,s.Tema,s.Cilj,o.OblikPsihID,o.Naziv";
        }
        if (criteria is string)
        {
            return $"where k.Ime='{criteria}' or k.Prezime='{criteria}' or
s.Tema='{criteria}' or s.Cilj='{criteria}' or o.Naziv='{criteria}' or
ps.Ime='{criteria}' or ps.Prezime='{criteria}' group by
s.SeansaID,s.DatumOdržavanja,s.Trajanje,s.Tema,s.Cilj, o.OblikPsihID,o.Naziv ";
        }
        return "";
    }
}

```

```

    }
    catch (Exception)
    {
        throw;
    }
}

public new IOpstiDomenskiObj ReaderResultZaJednog(SqlDataReader citac)
{
    if (citac.Read())
    {
        Psihoanaliza p = new Psihoanaliza
        {
            SeansaID = citac.GetInt32(0),
            PonašanjeKlijenta = citac.GetString(1),
            RazumevanjeUzroka = citac.GetString(2),
            Zaključak = citac.GetString(3),
            DatumOdržavanja = citac.GetDateTime(5),
            Trajanje = Convert.ToDouble(citac.GetValue(6)),
            Tema = citac.GetString(7),
            Cilj = citac.GetString(8),

            Oblik = new OblikPsihoterapije
            {
                OblikPsihID = citac.GetInt32(10), Naziv =
citac.GetString(11) }
            };

            return p;
        }
        return null;
    }
}

public new object NoviID(object rezultat)
{
    if (rezultat is DBNull)
    {
        return 1;
    }
    int maxid = (int)rezultat;
    return maxid + 1;
}
}

```

```

[Serializable]
public class REBT : Seansa , IOpstiDomenskiObj
{
    [Browsable(false)]
    public string Domaći { get; set; }
    [Browsable(false)]
    public string Radionica { get; set; }
    [Browsable(false)]
    public string Napredak { get; set; }

    [Browsable(false)]
    public new string NazivTabele => "REBT";
}

```

```

[Browsable(false)]
public new string NazivTabele2 => "Seansa";
[Browsable(false)]
public new string InsertVrednosti =>
$"{SeansaID},{Domaći},{Radionica},{Napredak}";
[Browsable(false)]
public new string InsertVrNasledjivanje =>
$"{SeansaID},{DatumOdržavanja},{Trajanje},{Tema},{Cilj},{Oblik.OblikPsihID}";
;
[Browsable(false)]
public new string UpdateVrNasledjivanje => $"SeansaID
={SeansaID},DatumOdržavanja='{DatumOdržavanja}',Trajanje={Trajanje},Tema='{Tema}',
Cilj='{Cilj}',OblikID = {Oblik.OblikPsihID}";
[Browsable(false)]
public new string UpdateVrednosti =>
$"{SeansaID},{UradjenDomaći='{Domaći}',PohadjanjeRadionice='{Radionica}',N
apredak='{Napredak}'";
[Browsable(false)]
public new string NazivTabeleAlias => "REBT r";
[Browsable(false)]
public new string Join => "join Seansa s on r.SeansaID=s.SeansaID join
OblikPsihoterapije o on o.OblikPsihID=s.OblikID";
[Browsable(false)]
public new string SearchId => $"where {ColumnId}={SeansaID}";
[Browsable(false)]
public new object ColumnId => "SeansaID";
[Browsable(false)]
public new string UslovZaJednog => $"where r.{ColumnId}={SeansaID}";
[Browsable(false)]
public new string UslovZaJednog2 => "";
[Browsable(false)]
public new string JoinPretragaSeansa => $"join Seansa s on
r.SeansaID=s.SeansaID join OblikPsihoterapije o on o.OblikPsihID=s.OblikID join
StavkaSeanse st on s.SeansaID= st.SeansaID join Klijent k on
st.KlijentID=k.KlijentID join Psihoterapeut ps on k.TerapeutID=ps.TerapeutID";

[Browsable(false)]
public new string Zvezda =>
"s.SeansaID,s.DatumOdržavanja,s.Trajanje,s.Tema,s.Cilj,o.OblikPsihID,o.Naziv";

public new List<IOpstiDomenskiObj> GetReaderResult(SqlDataReader citac)
{
    List<IOpstiDomenskiObj> lista = new List<IOpstiDomenskiObj>();
    while (citac.Read())
    {
        REBT r = new REBT();

        r.SeansaID = citac.GetInt32(0);
        r.Domaći = citac.GetString(1);
        r.Radionica = citac.GetString(2);
        r.Napredak = citac.GetString(3);
        r.DatumOdržavanja = citac.GetDateTime(5);
        r.Trajanje = Convert.ToDouble(citac.GetValue(6));
        r.Tema = citac.GetString(7);
        r.Cilj = citac.GetString(8);

        r.Oblik = new OblikPsihoterapije();
        r.Oblik.OblikPsihID = citac.GetInt32(10);
        r.Oblik.Naziv = citac.GetString(11);
    }
}

```

```

        lista.Add(r);
    }
    return lista;
}

public new List<IOpstiDomenskiObj> GetReaderResultJoin(SqlDataReader
citac)
{
    List<IOpstiDomenskiObj> lista = new List<IOpstiDomenskiObj>();
    while (citac.Read())
    {
        REBT r = new REBT();

        r.SeansaID = citac.GetInt32(0);
        r.DatumOdržavanja = citac.GetDateTime(1);
        r.Trajanje = Convert.ToDouble(citac.GetValue(2));
        r.Tema = citac.GetString(3);
        r.Cilj = citac.GetString(4);

        r.Oblik = new OblikPsihoterapije();
        r.Oblik.OblikPsihID = citac.GetInt32(5);
        r.Oblik.Naziv = citac.GetString(6);

        lista.Add(r);
    }
    return lista;
}

public new string SearchWhere(string criteria)
{
    try
    {
        if (int.TryParse(criteria, out int deo))
        {
            return $"where {deo} = DATEPART(MM,s.DatumOdržavanja) or {deo}
= DATEPART(yyyy,s.DatumOdržavanja) group by
s.SeansaID,s.DatumOdržavanja,s.Trajanje,s.Tema,s.Cilj,o.OblikPsihID,o.Naziv";
        }
        if (DateTime.TryParse(criteria, out DateTime datum))
        {
            return $"where cast('{datum.ToString("yyyy-MM-dd")}' as date)=
cast(s.DatumOdržavanja as date) group by
s.SeansaID,s.DatumOdržavanja,s.Trajanje,s.Tema,s.Cilj,o.OblikPsihID,o.Naziv";
        }
        if (criteria is string)
        {
            return $"where k.Ime='{criteria}' or k.Prezime='{criteria}' or
s.Tema='{criteria}' or s.Cilj='{criteria}' or o.Naziv='{criteria}' or
ps.Ime='{criteria}' or ps.Prezime='{criteria}' group by
s.SeansaID,s.DatumOdržavanja,s.Trajanje,s.Tema,s.Cilj,o.OblikPsihID,o.Naziv";
        }
        return "";
    }
    catch (Exception)
    {
        throw;
    }
}

```

```

    }

    public new IOpstiDomenskiObj ReaderResultZaJednog(SqlDataReader citac)
    {
        if (citac.Read())
        {
            REBT p = new REBT
            {
                SeansaID = citac.GetInt32(0),
                Domaći = citac.GetString(1),
                Radionica = citac.GetString(2),
                Napredak = citac.GetString(3),
                DatumOdržavanja = citac.GetDateTime(5),
                Trajanje = Convert.ToDouble(citac.GetValue(6)),
                Tema = citac.GetString(7),
                Cilj = citac.GetString(8),

                Oblik = new OblikPsihoterapije()
                { OblikPsihID= citac.GetInt32(10) , Naziv= citac.GetString(11)
            }

        };

        return p;
    }
    return null;
}

public new object NoviID(object rezultat)
{
    if (rezultat is DBNull)
    {
        return 1;
    }
    int maxid = (int)rezultat;
    return maxid + 1;
}

}

[Serializable]
public class Seansa : IOpstiDomenskiObj
{
    public Seansa()
    {
        ListaStavki = new BindingList<StavkaSeanse>();
    }

    protected BindingList<StavkaSeanse> listaStavki;

    [Browsable(false)]
    public int SeansaID { get; set; }
    public DateTime DatumOdržavanja { get; set; }
    public double Trajanje { get; set; }
    public string Tema { get; set; }
    public string Cilj { get; set; }
}

```

```

        public OblikPsihoterapije Oblik { get; set; }
        public BindingList<StavkaSeansa> ListaStavki { get => listaStavki; set =>
listaStavki = value; }

        [Browsable(false)]
        public string NazivTabele => "Seansa ";
        [Browsable(false)]
        public string InsertVrednosti =>
$" {SeansaID}, '{DatumOdržavanja}', {Trajanje}, '{Tema}', '{Cilj}', {Oblik.OblikPsihID}"
;

        [Browsable(false)]
        public string UpdateVrednosti => "";
        [Browsable(false)]
        public string Join => "";
        [Browsable(false)]
        public string SearchId => "";
        [Browsable(false)]
        public object ColumnId => "SeansaID";
        [Browsable(false)]
        public string UslovZaJednog => "";
        [Browsable(false)]
        public string NazivTabele2 => "Seansa";
        [Browsable(false)]
        public string UslovZaJednog2 => "";
        [Browsable(false)]
        public string InsertVrNasledjivanje => "";
        [Browsable(false)]
        public string NazivTabeleAlijas => "Seansa s";
        [Browsable(false)]
        public string UpdateVrNasledjivanje => "";
        [Browsable(false)]
        public string JoinPretragaSeansa => "";
        [Browsable(false)]
        public string Zvezda => "";

        public List<IOpstiDomenskiObj> GetReaderResult(SqlDataReader reader)
        {
            return null;
        }

        public string SearchWhere(string criteria)
        {
            return null;
        }

        public IOpstiDomenskiObj ReaderResultZaJednog(SqlDataReader reader)
        {
            return null;
        }

        public object NoviID(object rezultat)
        {
            if (rezultat is DBNull)
            {
                return 1;
            }
            int maxid = (int)rezultat;
            return maxid + 1;
        }

```



```

public void vratiBit(IOpstiDomenskiObj obj)
{
    return;
}

public string SearchZaStavke(int id)
{
    throw new NotImplementedException();
}

public List<IOpstiDomenskiObj> GetReaderResultJoin(SqlDataReader reader)
{
    throw new NotImplementedException();
}

}

public enum Status { NijeMenjana, Uneta, Izmenjena, Obrisana }
[Serializable]
public class StavkaSeanse : IOpstiDomenskiObj
{
    public int Rbr { get; set; }
    public Klient Klient { get; set; }
    [Browsable(false)]
    public int SeansaID { get; set; }
    public bool Prisustvo { get; set; }
    [DisplayName("Cena seansa")]
    public int CenaSeanse { get; set; }
    [DisplayName("Plaćena seansa")]
    public bool Plaćeno { get; set; }

    public Status Status { get; set; }
    int placeno;
    int prisustvo;

    [Browsable(false)]
    public string NazivTabele => "StavkaSeanse";
    [Browsable(false)]
    public string InsertVrednosti =>
    $"{Klient.KlijentID},{SeansaID},{Rbr},{prisustvo},{placeno},{CenaSeanse}";
    [Browsable(false)]
    public string UpdateVrednosti =>
    $"{KlijentID={Klijent.KlijentID},SeansaID={SeansaID},Rbr={Rbr},Prisustvo={prisustvo},Platio={placeno},CenaSeanse={CenaSeanse}";
    [Browsable(false)]
    public string Join => "join Klient k on s.KlijentID=k.KlijentID";
    [Browsable(false)]
    public string SearchId => $"where s.SeansaID={SeansaID}";
    [Browsable(false)]
    public object ColumnId => "";
    [Browsable(false)]
    public string UslovZaJednog => "";
    [Browsable(false)]
    public string NazivTabele2 => "";
    [Browsable(false)]
    public string UslovZaJednog2 => "";
    [Browsable(false)]

```

```

public string InsertVrNasledjivanje => "";
[Browsable(false)]
public string NazivTabeleAlijas => "StavkaSeanse s";

[Browsable(false)]
public string UpdateVrNasledjivanje => "";

[Browsable(false)]
public string JoinPretragaSeansa => throw new NotImplementedException();
[Browsable(false)]
public string Zvezda => throw new NotImplementedException();

public string SearchZaStavke(int id)
{
    return $"where SeansaID={id} and KlientID={Klijent.KlijentID} and
Rbr={Rbr}";
}

//jel moze ovde da se napravi nesto sto je spec, a ne iz IODO?
public string SearchZaStavkePoSeansi(int id)
{
    return $"where s.SeansaID={id}";
}

public string SearchZaStavkePoKlijentu(int id)
{
    return $"where s.KlijentID={id} and s.Platio=0";
}

public List<IOpstiDomenskiObj> GetReaderResult(SqlDataReader citac)
{
    List<IOpstiDomenskiObj> lista = new List<IOpstiDomenskiObj>();
    while (citac.Read())
    {
        StavkaSeanse s = new StavkaSeanse();
        s.Klijent = new Klijent();
        s.Klijent.KlijentID = citac.GetInt32(0);
        s.Klijent.Ime = citac.GetString(7);
        s.Klijent.Prezime = citac.GetString(8);
        s.SeansaID = citac.GetInt32(1);
        s.Rbr = citac.GetInt32(2);
        s.Prisustvo = citac.GetBoolean(3);
        s.Plaćeno = citac.GetBoolean(4);
        s.CenaSeanse = citac.GetInt32(5);

        lista.Add(s);
    }
    return lista;
}

public object NoviID(object rezultat)
{
    return null;
}

public void vratiBit(IOpstiDomenskiObj obj)
{
    StavkaSeanse s = (StavkaSeanse)obj;
    if (s.Plaćeno)
        placeno = 1;
}

```

```

        else placeno = 0;

        if (s.Prisustvo)
            prisustvo = 1;
        else prisustvo = 0;
    }

    public IOpstiDomenskiObj ReaderResultZaJednog(SqlDataReader reader)
    {
        return null;
    }

    public string SearchWhere(string criteria)
    {
        //int id = Convert.ToInt32(criteria);
        return "";
    }

    public List<IOpstiDomenskiObj> GetReaderResultJoin(SqlDataReader reader)
    {
        throw new NotImplementedException();
    }
}

[Serializable]
public class OblikPsihoterapije : IOpstiDomenskiObj
{
    public int OblikPsihID { get; set; }
    public string Naziv { get; set; }

    public override string ToString()
    {
        return Naziv;
    }

    public string NazivTabele => "OblikPsihoterapije";

    public string InsertVrednosti => "";

    public string UpdateVrednosti => "";

    public string Join => "";

    public string SearchId => "";

    public object ColumnId => "";

    public string UslovZaJednog => "";

    public string NazivTabele2 => "";

    public string UslovZaJednog2 => "";

    public string InsertVrNasledjivanje => "";

    public string NazivTabeleAlias => "OblikPsihoterapije";
}

```

```

        public string UpdateVrNasledjivanje => throw new
        NotImplementedException();

        public string JoinPretragaSeansa => throw new NotImplementedException();

        public string Zvezda => throw new NotImplementedException();

        public List<IOpstiDomenskiObj> GetReaderResult(SqlDataReader citac)
        {
            List<IOpstiDomenskiObj> rez = new List<IOpstiDomenskiObj>();

            while (citac.Read())
            {
                OblikPsihoterapije op = new OblikPsihoterapije
                {
                    OblikPsihID = (int)citac["OblikPsihID"],
                    Naziv = (string)citac["Naziv"]
                };
                rez.Add(op);
            }

            return rez;
        }

        public string SearchWhere(string criteria)
        {
            throw new NotImplementedException();
        }

        public IOpstiDomenskiObj ReaderResultZaJednog(SqlDataReader reader)
        {
            throw new NotImplementedException();
        }

        public object NoviID(object rezultat)
        {
            throw new NotImplementedException();
        }

        public void vratiBit(IOpstiDomenskiObj obj)
        {
            throw new NotImplementedException();
        }

        public string SearchZaStavke(int id)
        {
            throw new NotImplementedException();
        }

        public List<IOpstiDomenskiObj> GetReaderResultJoin(SqlDataReader reader)
        {
            throw new NotImplementedException();
        }
    }

```

[Serializable]

```

public class Psihoterapeut : IOpstiDomenskiObj
{
    public int TerapeutID { get; set; }
    public string Ime { get; set; }
    public string Prezime { get; set; }
    public string KorisničkoIme { get; set; }
    public string KorisničkaŠifra { get; set; }

    public override string ToString()
    {
        return Ime + " " + Prezime;
    }

    [Browsable(false)]
    public string NazivTabele => "Psihoterapeut";

    [Browsable(false)]
    public string InsertVrednosti => "";

    [Browsable(false)]
    public string UpdateVrednosti => "";

    [Browsable(false)]
    public string Join => "";

    [Browsable(false)]
    public string SearchId => "";

    [Browsable(false)]
    public object ColumnId => "";

    [Browsable(false)]
    public string UslovZaJednog => $"where KorisničkoIme= '{KorisničkoIme}'
and KorisničkaŠifra='{KorisničkaŠifra}'";

    [Browsable(false)]
    public string NazivTabele2 => "";

    [Browsable(false)]
    public string UslovZaJednog2 => "";

    public string InsertVrNasledjivanje => "";

    public string NazivTabeleAlijas => "Psihoterapeut";

    public string UpdateVrNasledjivanje => throw new
NotImplementedException();

    public string JoinPretragaSeansa => throw new NotImplementedException();

    public string Zvezda => throw new NotImplementedException();

    [Browsable(false)]
    public List<IOpstiDomenskiObj> GetReaderResult(SqlDataReader reader)
    {
        return null;
    }

    [Browsable(false)]
    public string SearchWhere(string criteria)

```

```

{
    return null;
}

public IOpstiDomenskiObj ReaderResultZaJednog(SqlDataReader reader)
{
    if (reader.Read())
    {
        Psihoterapeut p = new Psihoterapeut
        {
            TerapeutID = (int)reader["TerapeutID"],
            Ime = (string)reader["Ime"],
            Prezime = (string)reader["Prezime"],
            KorisničkoIme = (string)reader["KorisničkoIme"],
            KorisničkaŠifra = (string)reader["KorisničkaŠifra"]
        };
        return p;
    }
    return null;
}

public object NoviID(object rezultat)
{
    throw new NotImplementedException();
}

public void vratiBit(IOpstiDomenskiObj obj)
{
    throw new NotImplementedException();
}

public string SearchZaStavke(int id)
{
    throw new NotImplementedException();
}

public List<IOpstiDomenskiObj> GetReaderResultJoin(SqlDataReader reader)
{
    throw new NotImplementedException();
}
}

```

Пословна логика – системске операције

OpstaSistemskaOperacija

```

public abstract class OpstaSistemskaOperacija
{
    protected Broker broker = new Broker();

    protected abstract void IzvrsiKonkretnuOperaciju(object obj);
    protected abstract void Validacija(object obj);
}

```

```

    public void Izvrsi(object objekat)
    {
        try
        {
            Validacija(objekat);
            broker.OtvoriKonekciju();
            broker.PokreniTransakciju();
            IzvrsiKonkretnuOperaciju(objekat);
            broker.CommitTransakcije();
        }
        catch (Exception)
        {
            broker.RollbackTransakcije();
            throw;
        }
        finally
        {
            broker.ZatvoriKonekciju();
        }
    }
}

```

УГ1: PrijavaPsihoterapeuta (Psihoterapeut)

Операција: PrijavaPsihoterapeuta(Psihoterapeut): signal

Веза са СК: СК1

Предуслови: /

Постуслови: /

```

public class LoginSO : OpstaSistemskaOperacija
{
    public Psihoterapeut p { get; set; }

    protected override void IzvrsiKonkretnuOperaciju(object obj)
    {
        p = broker.VratiKonkretnog((Psihoterapeut)obj) as Psihoterapeut;
    }

    protected override void Validacija(object obj)
    {
        if (!(obj is Psihoterapeut))
            throw new ArgumentException();
    }
}

```

УГ2: UnesiKlijenta (Klijent)

Операција: UnesiKlijenta(Klijent): signal

Веза са СК: СК2

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

Постуслови: Убачен је нови клијент.

```
public class UnesiKlijentaSO : OpstaSistemskaOperacija
{
    public Klijent Klijent { get; private set; }

    protected override void IzvrsiKonkretnuOperaciju(object obj)
    {
        Klijent k = (Klijent)obj;

        int rez = broker.Unesi(k);
        if (rez != 0)
            Klijent = k;
        else
            Klijent = null;
    }

    protected override void Validacija(object objekat)
    {
        if (!(objekat is Klijent))
        {
            throw new ArgumentException();
        }
    }
}
```

УГ3: UčitajListuKlijenata()

Операција: UčitajListuKlijenata(): signal

Веза са СК: СК3, СК4, СК5, СК6,СК8

Предуслови: /

Постуслови: /

```
public class UcitajListuKlijenataSO : OpstaSistemskaOperacija
{
    public List<Klijent> ListaKl { get; private set; }

    protected override void IzvrsiKonkretnuOperaciju(object obj)
    {
        Klijent k = (Klijent)obj;
        ListaKl = broker.VratilistuSvih(k).OfType<Klijent>().ToList();
    }

    protected override void Validacija(object obj)
    {
        if (!(obj is Klijent))
            throw new ArgumentException();
    }
}
```


УГ4: UčitajGradove()

Операција: UčitajGradove(): signal

Веза са СК: CK2, CK4

Предуслови: /

Постуслови: /

```
public class UcitajListuGradovaSO : OpstaSistemskaOperacija
{
    public List<Grad> ListaG { get; private set; }

    protected override void IzvrsiKonkretnuOperaciju(object obj)
    {
        Grad g = (Grad)obj;
        ListaG = broker.VratiListuSvih(g).OfType<Grad>().ToList();
    }

    protected override void Validacija(object obj)
    {
        if (!(obj is Grad))
            throw new ArgumentException();
    }
}
```

УГ5: NadjiKlijente(kriterijumPretrage, List<Klijent>)

Операција: NadjiKlijente(kriterijumPretrage, List<Klijent>): signal

Веза са СК: CK3, CK4, CK5

Предуслови: /

Постуслови: /

```
public class PretragaKlijenataSO : OpstaSistemskaOperacija
{
    public List<Domen.Klijent> rez;
    protected override void IzvrsiKonkretnuOperaciju(object obj)
    {
        string krit = (string)obj;
        rez = broker.Pretraga(krit, new
Domen.Klijent()).OfType<Domen.Klijent>().ToList();
    }

    protected override void Validacija(object obj)
    {
        if (!(obj is string))
            throw new ArgumentException();
    }
}
```

УГ6: VratiKlijenta(Klijent)

Операција: VratiKlijenta(Klijent): signal

Веза са СК: CK3, CK4, CK5

Предуслови: /

Постуслови: /

```

public class VратиKlijentaSO : OpstaSistemskaOperacija
{
    public Klijent Klijent { get; set; }

    protected override void IzvrsiKonkretnuOperaciju(object obj)
    {
        Klijent = broker.VратиKonkretnog((Klijent)obj) as Klijent;
    }

    protected override void Validacija(object objekat)
    {
        if (!(objekat is Klijent))
        {
            throw new ArgumentException();
        }
    }
}

```

УГ7: IzmeniKlijenta(Klijent)

Операција: IzmeniKlijenta(Klijent): signal

Веза са СК: СК4

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

Постуслови: Унети подаци о клијенту су измењени.

```

public class IzmeniKlijentaSO : OpstaSistemskaOperacija
{
    public bool Izmenjen { get; set; }

    protected override void IzvrsiKonkretnuOperaciju(object obj)
    {
        Klijent k = (Klijent)obj;
        int rez = broker.Izmeni(k);
        if (rez != 0)
            Izmenjen = true;
        else
            Izmenjen = false;
    }

    protected override void Validacija(object obj)
    {
        if (!(obj is Klijent))
        {
            throw new ArgumentException();
        }
    }
}

```

УГ8: ObrišiKlijenta(Klijent)

Операција: ObrišiKlijenta(Klijent): signal

Веза са СК: СК5

Предуслови: Структурна ограничења над објектом *Клијент* морају бити задовољена.
Постуслови: Клијент је обрисан.

```
public class ObrisiKlijentaSO : OpstaSistemskaOperacija
{
    public bool Obrisan { get; set; }
    protected override void IzvrsiKonkretnuOperaciju(object obj)
    {
        Domen.Klijent k = (Domen.Klijent)obj;
        int rez = broker.Obrisi(k);
        if (rez != 0)
            Obrisan = true;
        else
            Obrisan = false;
    }
    protected override void Validacija(object obj)
    {
        if (!(obj is Domen.Klijent))
            throw new ArgumentException();
    }
}
```

УГ9: UčitajOblikePsihoterapije()

Операција: UčitajOblikePsihoterapije(): signal

Веза са СК: СК6,СК8

Предуслови: /

Постуслови: /

```
public class UcitajListuOblikaSO : OpstaSistemskaOperacija
{
    public List<OblikPsihoterapije> listaOP;

    protected override void IzvrsiKonkretnuOperaciju(object obj)
    {
        listaOP =
broker.VratiListuSvih((OblikPsihoterapije)obj).OfType<OblikPsihoterapije>().ToList();
    }

    protected override void Validacija(object obj)
    {
        if (!(obj is OblikPsihoterapije))
            throw new ArgumentException();
    }
}
```

УГ10: UnesiSeansu (Seansa)

Операција: UnesiSeansu(Seansa)

Веза са СК: СК6

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

Постуслови: Евидентирана је нова сеанса.

```

public class UnesiSeansuSO : OpstaSistemskaOperacija
{
    public bool Sacuvana { get; set; } = false;

    protected override void IzvrsiKonkretnuOperaciju(object obj)
    {
        if (obj is REBT)
        {
            REBT r = (REBT)obj;
            int rez1 = broker.UnesiPodtip((REBT)obj);
            if (rez1 != 0)
            {
                int rez2 = broker.Unesi((Seansa)obj);
                if (rez2 != 0)
                {
                    Sacuvana = true;

                    foreach (StavkaSeanse stavka in r.ListaStavki)
                    {
                        if (stavka.Plaćeno)
                        {
                            int razlika = stavka.CenaSeanse - 2000;
                            if (razlika > 0)
                                stavka.Klijent.UkupanDug -= razlika;
                            else if (razlika < 0)
                                stavka.Klijent.UkupanDug += (2000 -
stavka.CenaSeanse);

                        }
                        if (!stavka.Plaćeno)
                        { stavka.Klijent.UkupanDug += stavka.CenaSeanse; }
                        broker.Izmeni(stavka.Klijent);

                        int rez3 = broker.UnesiStavku(stavka);
                        if (rez3 == 0)
                            Sacuvana = false;
                    }
                }
            }
        }
        else if (obj is Psihoanaliza)
        {
            Psihoanaliza p = (Psihoanaliza)obj;
            int rez1 = broker.UnesiPodtip((Psihoanaliza)obj);
            if (rez1 != 0)
            {
                int rez2 = broker.Unesi((Seansa)obj);
                if (rez2 != 0)
                {
                    Sacuvana = true;

                    foreach (StavkaSeanse stavka in p.ListaStavki)
                    {
                        if (stavka.Plaćeno)
                        {

```

```

        int razlika = stavka.CenaSeanse - 2000;
        if (razlika > 0)
            stavka.Klijent.UkupanDug -= razlika;
        else if (razlika < 0)
            stavka.Klijent.UkupanDug += (2000 -
stavka.CenaSeanse);

    }
    if (!stavka.Plaćeno)
    { stavka.Klijent.UkupanDug += stavka.CenaSeanse; }
    broker.Izmeni(stavka.Klijent);

    int rez3 = broker.UnesiStavku(stavka);
    if (rez3 == 0)
        Sacuvana = false;
    }
}

}

}

}

protected override void Validacija(object obj)
{
    if (!(obj is Seansa))
        throw new ArgumentException();
}
}

```

УГ11: UčitajListuSeansi()

Операција: UčitajListuSeansi(): signal

Веза са СК: CK7, CK8, CK9

Предуслови: /

Постуслови: /

```
public class UcitajListuSeansiSO : OpstaSistemskaOperacija
{
    public List<Seansa> rez;

    protected override void IzvrsiKonkretnuOperaciju(object obj)
    {
        rez = new List<Seansa>();
        Psihoterapeut p = (Psihoterapeut)obj;
        List<REBT> listaR = broker.VratiListuSvihSeansi(p,new
REBT()).OfType<REBT>().ToList();
        List<Psihoanaliza> listaP = broker.VratiListuSvihSeansi(p,new
Psihoanaliza()).OfType<Psihoanaliza>().ToList();
        foreach (REBT r in listaR)
        {
            rez.Add(r);
        }
        foreach (Psihoanaliza ps in listaP)
        {
            rez.Add(ps);
        }
    }

    protected override void Validacija(object obj)
    {
        if (!(obj is Psihoterapeut))
            throw new ArgumentException();
    }
}
```

УГ12: NadjiSeanse (kriterijumPretrage, List<Seansa>)

Операција: NadjiSeanse(kriterijumPretrage, List<Seansa>): signal

Веза са СК: CK7, CK8, CK9

Предуслови: /

Постуслови: /

```
public class PretragaSeansiSO : OpstaSistemskaOperacija
{
    public List<Seansa> rez;
    protected override void IzvrsiKonkretnuOperaciju(object obj)
    {
        rez = new List<Seansa>();
        string krit = (string)obj;
        List<REBT> listaR = broker.Pretraga(krit, new
REBT()).OfType<REBT>().ToList();
        List<Psihoanaliza> listaPS = broker.Pretraga(krit, new
Psihoanaliza()).OfType<Psihoanaliza>().ToList();
    }
}
```

```

        foreach (REBT s in listaR)
        {
            rez.Add(s);
        }
        foreach (Psihoanaliza s in listaPS)
        {
            rez.Add(s);
        }
    }

    protected override void Validacija(object obj)
    {
        if (!(obj is string))
            throw new ArgumentException();
    }
}

```

УГ13: VратиSeansu (Seansa)

Операција: VратиSeansu(Seansa): signal

Веза са СК: CK7, CK8, CK9

Предуслови: /

Постуслови: /

```

public class VратиSeansuSO : OpstaSistemskaOperacija
{
    public Domen.Seansa seansa;

    protected override void IzvrsiKonkretnuOperaciju(object obj)
    {
        seansa = new Domen.Seansa();
        StavkaSeanse s = new StavkaSeanse();
        if (obj is REBT)
        {
            REBT r = (REBT)obj;
            r = broker.VратиKonkretnog(r) as REBT;
            r.ListaStavki = new BindingList<StavkaSeanse>(broker.VратиSveStavke(s,
r.SeansaID).OfType<StavkaSeanse>().ToList());
            seansa = r;
        }
        else if (obj is Psihoanaliza)
        {
            Psihoanaliza p = (Psihoanaliza)obj;
            p = broker.VратиKonkretnog(p) as Psihoanaliza;
            p.ListaStavki = new BindingList<StavkaSeanse>(broker.VратиSveStavke(s,
p.SeansaID).OfType<StavkaSeanse>().ToList());
            seansa = p;
        }
    }

    protected override void Validacija(object obj)
    {
        if (!(obj is Domen.Seansa))
    }
}

```

```
        throw new ArgumentException();
    }
}
```

```
}
```

УГ14: IzmeniSeansu (Seansa)

Операција: IzmeniSeansu(Seansa): signal

Веза са СК: СК8

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

Постуслови: Унети подаци о сеанси су измењени.

```
public class IzmeniSeansuSO : OpstaSistemskaOperacija
{
    public bool izmenjena { get; set; }

    protected override void IzvrsiKonkretnuOperaciju(object obj)
    {
        if (obj is REBT)
        {
            REBT r = (REBT)obj;
            int rez1 = broker.IzmeniPodtip(r);
            if (rez1 != 0)
            {
                int rez2 = broker.Izmeni((Seansa)obj);
                if (rez2 != 0)
                {
                    izmenjena = true;

                    foreach (StavkaSeanse stavka in r.ListaStavki)
                    {
                        int rez3 = 1;
                        switch (stavka.Status)
                        {
                            case Status.NijeMenjana:
                                break;
                            case Status.Uneta:
                                if (stavka.Plaćeno)
                                {
                                    int razlika = stavka.CenaSeanse - 2000;
                                    if (razlika > 0)
                                        stavka.Klijent.UkupanDug -= razlika;
                                    else if (razlika < 0)
                                        stavka.Klijent.UkupanDug += (2000 -
stavka.CenaSeanse);

                                }
                                if (!stavka.Plaćeno)
                                { stavka.Klijent.UkupanDug += stavka.CenaSeanse; }
                                broker.Izmeni(stavka.Klijent);

                                rez3 = broker.UnesiStavku(stavka);
                                break;
                            case Status.Izmenjena:
```



```

        StavkaSeanse neizmenjena =
        (StavkaSeanse)broker.VratiKonkretnog(stavka);
        if (stavka.Plaćeno && !neizmenjena.Plaćeno)
        {
            int razlika = stavka.CenaSeanse - 2000;
            if (razlika > 0)
                stavka.Klijent.UkupanDug -= razlika;
            if (razlika == 0)
                stavka.Klijent.UkupanDug -= 2000;
            if (razlika < 0)
                stavka.Klijent.UkupanDug += (2000 -
stavka.CenaSeanse);

        }
        if (!stavka.Plaćeno && neizmenjena.Plaćeno &&
        (neizmenjena.CenaSeanse - 2000) >= 0)
            stavka.Klijent.UkupanDug +=
neizmenjena.CenaSeanse;
        if (!stavka.Plaćeno && neizmenjena.Plaćeno &&
        (stavka.CenaSeanse - 2000) < 0)
            stavka.Klijent.UkupanDug += 2000;
        broker.Izmeni(stavka.Klijent);

        rez3 = broker.IzmeniStavku(stavka, r.SeansaID);
        break;
        case Status.Obrisana:
            rez3 = broker.ObrisiStavku(stavka, r.SeansaID);
            break;
        default:
            break;
    }
    if (rez3 == 0)
        izmenjena = false;
    }
}
}
else if (obj is Psihoanaliza)
{
    Psihoanaliza r = (Psihoanaliza)obj;
    int rez1 = broker.IzmeniPodtip(r);
    if (rez1 != 0)
    {
        int rez2 = broker.Izmeni((Seansa)obj);
        if (rez2 != 0)
        {
            izmenjena = true;

            foreach (StavkaSeanse stavka in r.ListaStavki)
            {
                int rez3 = 1;

                switch (stavka.Status)
                {
                    case Status.NijeMenjana:
                        break;
                    case Status.Uneta:
                        if (stavka.Plaćeno)

```

```

        {
            int razlika = stavka.CenaSeanse - 2000;
            if (razlika > 0)

                stavka.Klijent.UkupanDug -= razlika;
            else if (razlika < 0)
                stavka.Klijent.UkupanDug += (2000 -
stavka.CenaSeanse);

        }
        if (!stavka.Plaćeno)
        { stavka.Klijent.UkupanDug += stavka.CenaSeanse; }
        broker.Izmeni(stavka.Klijent);

        rez3 = broker.UnesiStavku(stavka);
        break;
        case Status.Izmenjena:
            StavkaSeanse neizmenjena =
(StavkaSeanse)broker.VratiKonkretnog(stavka);
            if (stavka.Plaćeno && !neizmenjena.Plaćeno)
            {
                int razlika = stavka.CenaSeanse - 2000;
                if (razlika > 0)
                    stavka.Klijent.UkupanDug -= razlika;
                if (razlika == 0)
                    stavka.Klijent.UkupanDug -= 2000;
                if (razlika < 0)
                    stavka.Klijent.UkupanDug += (2000 -
stavka.CenaSeanse);

            }
            if (!stavka.Plaćeno && neizmenjena.Plaćeno )
                stavka.Klijent.UkupanDug += stavka.CenaSeanse;

            broker.Izmeni(stavka.Klijent);

            rez3 = broker.IzmeniStavku(stavka, r.SeansaID);
            break;
            case Status.Obrisana:
                rez3 = broker.ObrisiStavku(stavka, r.SeansaID);
                break;
            default:
                break;
        }
        if (rez3 == 0)
            izmenjena = false;
    }
}
}

}

}

protected override void Validacija(object obj)
{
    if (!(obj is Seansa))
        throw new ArgumentException();
}

```

}

}

УГ15: ОбришиSeansu(Seansa)

Операција: ОбришиSeansu(Seansa): signal

Веза са СК: СК5

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

Постуслови: Сеанса је обрисана.

```
public class ObrisiSeansuSO : OpstaSistemskaOperacija
{
    public bool obrisana { get; set; } = true;
    protected override void IzvrsiKonkretnuOperaciju(object obj)
    {
        if (obj is REBT)
        {
            REBT r = (REBT)obj;

            foreach (StavkaSeanse stavka in r.ListaStavki)
            {
                if (broker.ObrisiStavku(stavka, r.SeansaID) == 0)
                    obrisana = false;
            }
            if (broker.Obrisi((Seansa)obj) == 0)
                obrisana = false;

            if (broker.ObrisiPodtip(r) == 0)
                obrisana = false;
        }
        else if (obj is Psihoanaliza)
        {
            Psihoanaliza r = (Psihoanaliza)obj;

            foreach (StavkaSeanse stavka in r.ListaStavki)
            {
                if (broker.ObrisiStavku(stavka, r.SeansaID) == 0)
                    obrisana = false;
            }
            if (broker.Obrisi((Seansa)obj) == 0)
                obrisana = false;

            if (broker.ObrisiPodtip(r) == 0)
                obrisana = false;
        }
    }

    protected override void Validacija(object obj)
    {
        if (!(obj is Seansa))
```

```
        throw new ArgumentException();  
    }
```


Брокер базе података

Broker је софтверска класа одговорна за комуникацију између пословне логике и складишта података.

Класа Broker представља перзистентни оквир који посредује у свим операцијама над базом података.

```
private SqlConnection konekcija;
private SqlTransaction transakcija;

public Broker()
{
    konekcija = new SqlConnection(@"Data Source=.\SQLEXPRESS;Initial
Catalog=PsihološkoSavetovalište;Integrated
Security=True;MultipleActiveResultSets=True");
}

public void OtvoriKonekciju()
{
    konekcija.Open();
}

public void ZatvoriKonekciju()
{
    konekcija.Close();
}

public void PokreniTransakciju()
{
    transakcija = konekcija.BeginTransaction();
}

public void CommitTransakcije()
{
    transakcija.Commit();
}

public void RollbackTransakcije()
{
    transakcija.Rollback();
}

public object VratiNovId(IOpstiDomenskiObj obj)
{
    SqlCommand komanda = new SqlCommand();
    komanda.Connection = konekcija;
    komanda.Transaction = transakcija;

    komanda.CommandText = $"select max({obj.ColumnId}) from
{obj.NazivTabele}";
    object rez = komanda.ExecuteScalar();
    return obj.NoviID(rez);
}
```

```

    }

    public int Unesi(IOpstiDomenskiObj objekat)
    {
        SqlCommand komanda = new SqlCommand();
        komanda.Transaction = transakcija;
        komanda.Connection = konekcija;

        komanda.CommandText = $"Insert into {objekat.NazivTabele}
values({objekat.InsertVrednosti})";
        return komanda.ExecuteNonQuery();
    }

    public int Izmeni(IOpstiDomenskiObj obj)
    {
        SqlCommand komanda = new SqlCommand();
        komanda.Transaction = transakcija;
        komanda.Connection = konekcija;

        komanda.CommandText = $"update {obj.NazivTabele} set
{obj.UpdateVrednosti} {obj.SearchId}";
        return komanda.ExecuteNonQuery();
    }

    public int Obrisi(IOpstiDomenskiObj obj)
    {
        SqlCommand komanda = new SqlCommand();
        komanda.Connection = konekcija;
        komanda.Transaction = transakcija;

        komanda.CommandText = $"delete from {obj.NazivTabele} {obj.SearchId}";
        return komanda.ExecuteNonQuery();
    }

    public int UnesiNasledj(IOpstiDomenskiObj objekat)
    {
        SqlCommand komanda = new SqlCommand();
        komanda.Transaction = transakcija;
        komanda.Connection = konekcija;

        komanda.CommandText = $"Insert into {objekat.NazivTabele2}
values({objekat.InsertVrNasledjivanje})";
        return komanda.ExecuteNonQuery();
    }

    public int IzmeniNasledj(IOpstiDomenskiObj objekat)
    {
        SqlCommand komanda = new SqlCommand();
        komanda.Transaction = transakcija;
        komanda.Connection = konekcija;

        komanda.CommandText = $"Update {objekat.NazivTabele2} set
{objekat.UpdateVrNasledjivanje} {objekat.SearchId}";
        return komanda.ExecuteNonQuery();
    }

    public int ObrisiNasledj(IOpstiDomenskiObj obj)

```

```

    {
        SqlCommand komanda = new SqlCommand();
        komanda.Transaction = transakcija;
        komanda.Connection = konekcija;

        komanda.CommandText = $"Delete from {obj.NazivTabele2}
{obj.SearchId}";
        return komanda.ExecuteNonQuery();
    }

    public List<IOpstiDomenskiObj> Pretraga(string kriterijum,
IOpstiDomenskiObj obj)
    {
        SqlCommand command = new SqlCommand();
        command.Transaction = transakcija;
        command.Connection = konekcija;

        command.CommandText = $"select {obj.Zvezda} from
{obj.NazivTabeleAlijas} {obj.JoinPretragaSeansa} {obj.SearchWhere(kriterijum)}";
        SqlDataReader reader = command.ExecuteReader();
        return obj.GetReaderResultJoin(reader);
    }

    public List<IOpstiDomenskiObj> VratiListuSvih(IOpstiDomenskiObj obj)
    {
        SqlCommand command = new SqlCommand();
        command.Transaction = transakcija;
        command.Connection = konekcija;

        command.CommandText = $"select * from {obj.NazivTabeleAlijas}
{obj.Join} ";
        using (SqlDataReader reader = command.ExecuteReader())
        {
            return obj.GetReaderResult(reader);
        }
    }

    public IOpstiDomenskiObj VratiKonkretnog(IOpstiDomenskiObj obj)
    {
        SqlCommand komanda = new SqlCommand();
        komanda.Transaction = transakcija;
        komanda.Connection = konekcija;

        komanda.CommandText = $"select * from {obj.NazivTabeleAlijas}
{obj.Join} {obj.UslovZaJednog}";
        using (SqlDataReader citac = komanda.ExecuteReader())
            return obj.ReaderResultZaJednog(citac);
    }

    public int UnesiStavku(IOpstiDomenskiObj obj)
    {
        SqlCommand komanda = new SqlCommand();
        komanda.Transaction = transakcija;
        komanda.Connection = konekcija;

        obj.vratiBit(obj);
    }

```



```

        komanda.CommandText = $"Insert into {obj.NazivTabele}
values({obj.InsertVrednosti})";
        return komanda.ExecuteNonQuery();
    }

    public int IzmeniStavku(IOpstiDomenskiObj obj, int id)
    {
        SqlCommand komanda = new SqlCommand();
        komanda.Transaction = transakcija;
        komanda.Connection = konekcija;

        obj.vratiBit(obj);
        komanda.CommandText = $"Update {obj.NazivTabele} set
{obj.UpdateVrednosti} {obj.SearchZaStavke(id)}";
        return komanda.ExecuteNonQuery();
    }

    public int ObrisiStavku(StavkaSeanse obj, int id)
    {
        SqlCommand komanda = new SqlCommand();
        komanda.Transaction = transakcija;
        komanda.Connection = konekcija;

        komanda.CommandText = $"Delete from {obj.NazivTabele}
{obj.SearchZaStavke(id)}";
        return komanda.ExecuteNonQuery();
    }

    public List<IOpstiDomenskiObj> VratiSveStavke(StavkaSeanse obj, int id)
    {
        SqlCommand command = new SqlCommand();
        command.Transaction = transakcija;
        command.Connection = konekcija;

        command.CommandText = $"select * from {obj.NazivTabeleAlijas}
{obj.Join} {obj.SearchZaStavkePoSeansi(id)}";
        using (SqlDataReader reader = command.ExecuteReader())
        {
            return obj.GetReaderResult(reader);
        }
    }

    public List<IOpstiDomenskiObj> VratiSveStavkeDug(StavkaSeanse obj, int id)
    {
        SqlCommand command = new SqlCommand();
        command.Transaction = transakcija;
        command.Connection = konekcija;

        command.CommandText = $"select * from {obj.NazivTabeleAlijas}
{obj.Join} {obj.SearchZaStavkePoKlijentu(id)}";
        using (SqlDataReader reader = command.ExecuteReader())
        {
            return obj.GetReaderResult(reader);
        }
    }

```

```
}
```

Све методе рада су пројектоване као генеричке, што значи да могу да прихвате различите објекте и различите упите преко улазних параметара како се не би за сваку системску операцију креирала метода у класи Broker.

Резултат је мања комплексност саме класе Broker. Да би се то омогућило коришћен је интерфејс **IOpstiDomenskiObjekat**.

```
public interface IOpstiDomenskiObj
{
    string NazivTabele{ get; }
    string Zvezda { get; }
    string NazivTabeleAlijas { get; }
    string InsertVrednosti { get; }
    string InsertVrNasledjivanje { get; }
    string UpdateVrNasledjivanje { get; }
    string UpdateVrednosti { get; }
    string Join { get; }
    string JoinPretragaSeansa { get; }
    string UslovZaJednog { get; }
    string UslovZaJednog2 { get; }
    string SearchWhere(string criteria);
    string SamoZaT(IOpstiDomenskiObj obj);
    string SearchZaStavke(int id);
    string SearchId { get; }
    object ColumnId { get; }
    string NazivTabele2 { get; }
    void vratiBit(IOpstiDomenskiObj obj);
    List<IOpstiDomenskiObj> GetReaderResult(SqlDataReader reader);
    IOpstiDomenskiObj ReaderResultZaJednog(SqlDataReader reader);
    List<IOpstiDomenskiObj> GetReaderResultJoin(SqlDataReader reader);
    object NoviID(object rezultat);
}
```

Свака класа из домена имплементира дати интерфејс, и све његове методе и гетере. На тај начин је омогућено да методе класе Broker буду генеричке.

10.3. Имплементација корисничког интерфејса

Кориснички интерфејс је дефинисан преко скупа екранских форми. Сценарија коришћења екранских форми су директно повезана са сценаријима извршења случајева коришћења.

СК1: Пријава психотерапеута на систем

Назив СК

Пријава психотерапеута на систем

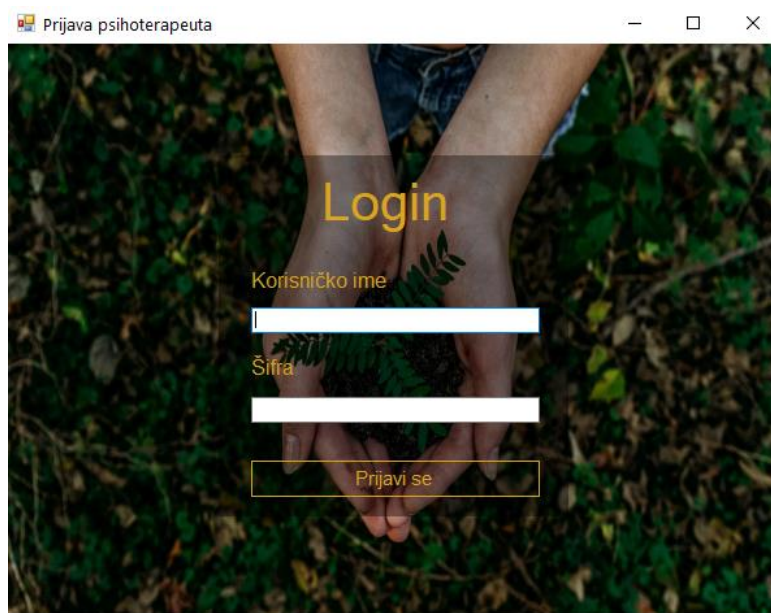
Актори СК

Психотерапеут

Учесници СК

Психотерапеут и систем (програм)

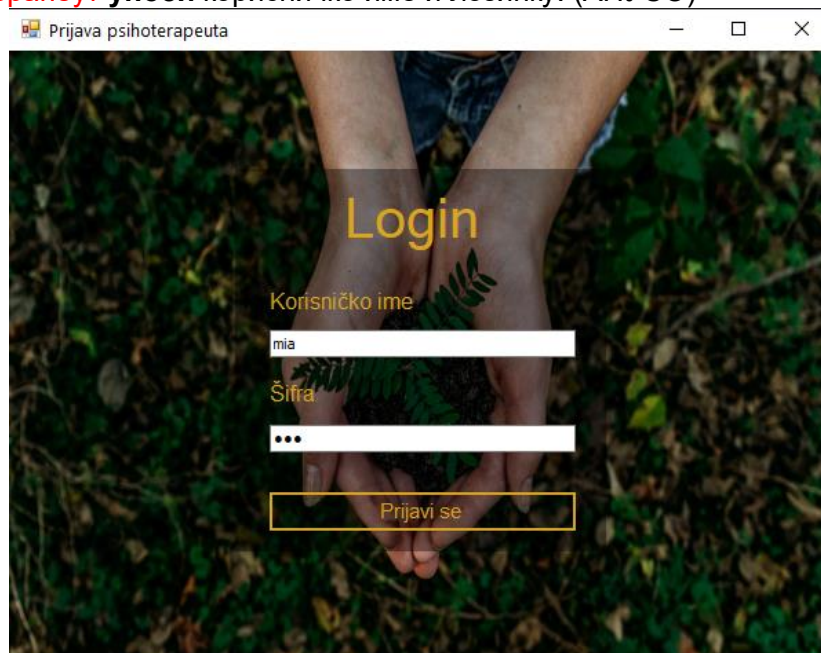
Предуслов: Систем је укључен. Систем приказује форму за пријављивање.



Слика 147 - Приказ форме за пријављивање

Основни сценарио СК

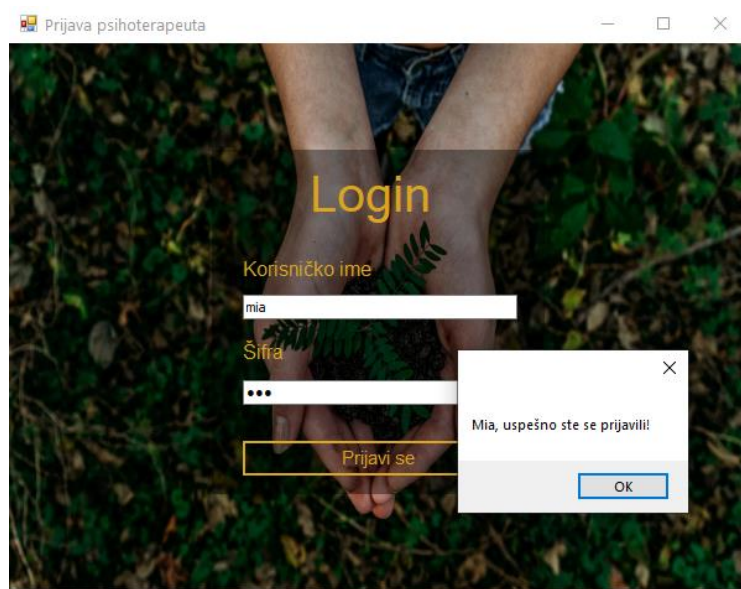
6. **Психотерапеут** уноси корисничко име и лозинку. (АПУСО)



Слика 148 - Унос корисничког имена и шифре

7. **Психотерапеут** контролише да ли је коректно унео корисничко име и лозинку. (АНСО)

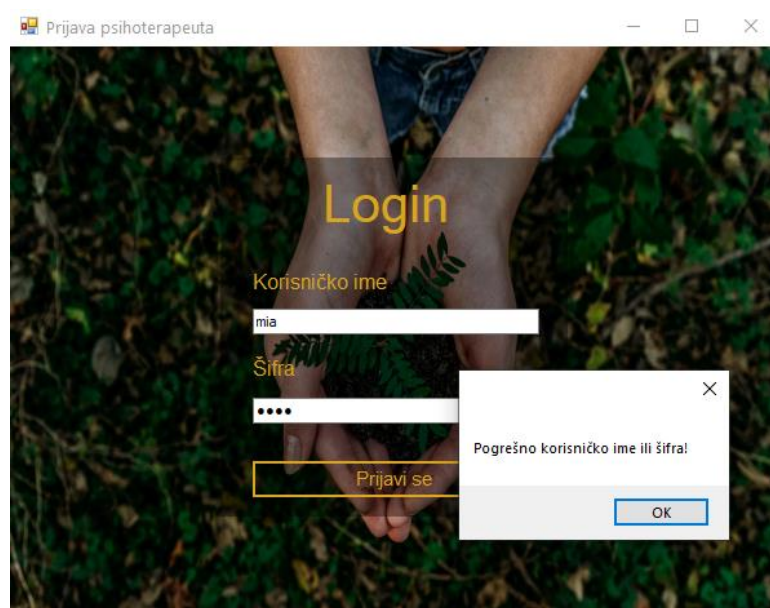
8. **Психотерапеут** **позива** **систем** да се пријави (провери податке). (АПСО)
9. **Систем** **проверава** податке о **психотерапеуту**. (СО)
10. **Систем** **приказује** **психотерапеуту** поруку: “Uspešno ste se prijavili!” (ИА)



Слика 149- Обавештење о успешној пријави

Алтернативна сценарија

- 10.2. Уколико **систем** не може да нађе **психотерапеута** он приказује **психотерапеуту** поруку: “Neuspešno prijavljivanje!” (ИА)



Слика 150- Обавештење о неуспешној пријави

СК2: Унос новог клијента

Назив СК

Унос новог клијента

Актори СК

Психотерапеут

Учесници СК

Психотерапеут и систем (програм)

Предуслов: Систем је укључен и психотерапеут је улогован под својом шифром. Систем приказује форму за рад са клијентима. Учитана је листа градова и ID клијента.

Unos kljenta

ID 9

Lični podaci

Ime

Prezime

Datum rođenja(dd.MM.yyyy)

Prebivalište Beograd

Kontakt

Broj telefona:

E-mail:

Unesi kljenta

Mia Popic

Слика 151 - Приказ форме та унос ногог клијента

Основни сценарио СК

11. Психотерапеут уноси податке о клијенту. (АПУСО)

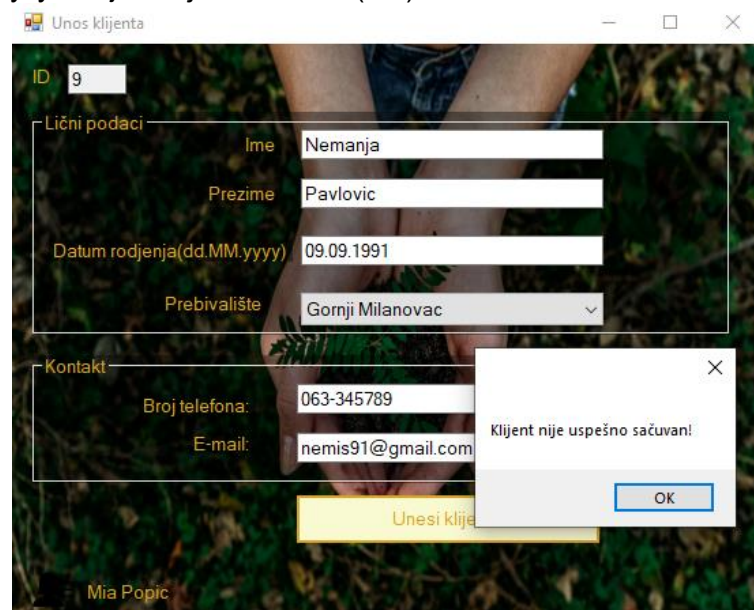
Слика 152 - Попуњена форма уноса клијента

12. **Психотерапеут** контролише да ли је коректно унео податке о **клијенту**. (АНСО)
13. **Психотерапеут** позива **систем** да запамти податке о **клијенту**. (АПСО)
14. **Систем** памти податке о **клијенту**. (СО)
15. **Систем** приказује **психотерапеуту** поруку: "Klijent uspešno sačuvan!" (ИА)

Слика 153 - Обавештење о успешном уносу новог клијента

Алтернативна сценарија

5.1. Уколико **СИСТЕМ** не може да запамти податке о **клијенту** он приказује **психотерапеуту** поруку: “Klijent nije sačuvan!” (ИА)



Слика 154 - Обавештење о неуспешном уносу новог клијента

СК3: Претраживање клијената

Назив СК

Претраживање **клијената**

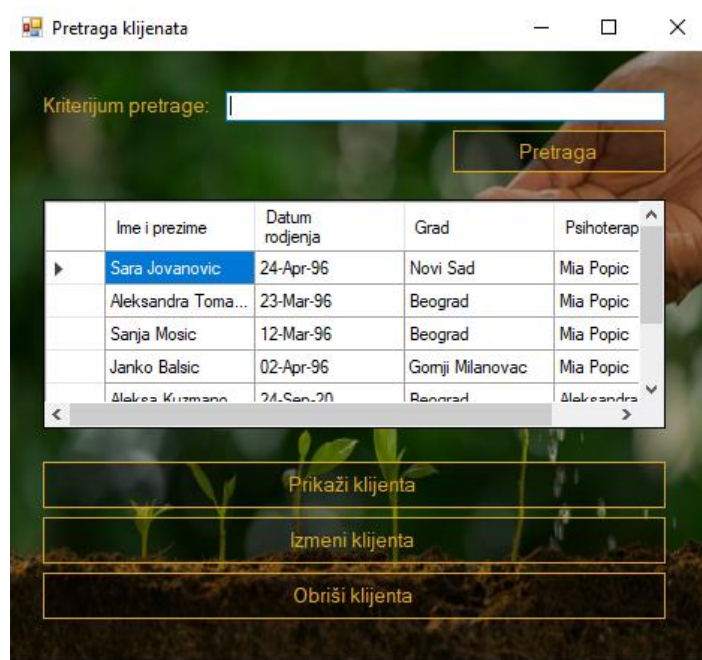
Актори СК

Психотерапеут

Учесници СК

Психотерапеут и **систем** (програм)

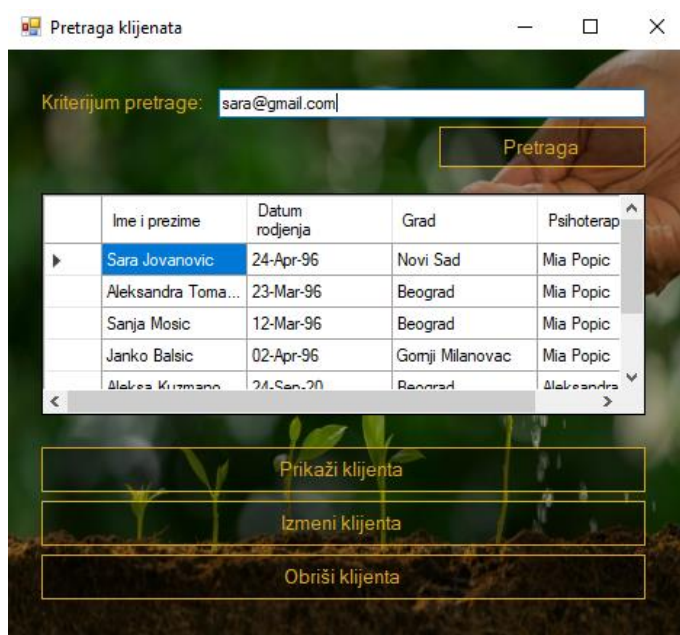
Предуслов: **Систем** је укључен и **психотерапеут** је улогован под својом шифром. **Систем** приказује форму за рад са **клијентима**. Учитана је листа **клијената**.



Слика 155 - Приказ форме за претрагу клијената

Основни сценарио СК

9. **Психотерапеут** уноси вредност по којој претражује клијенте. (АПУСО)

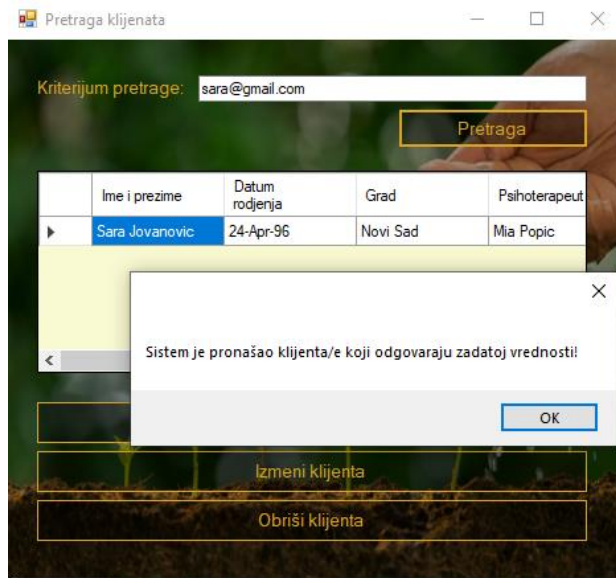


Слика 156 - Претрага клијената према изабраном критеријуму

10. **Психотерапеут** позива **систем** да нађе клијенте по задатој вредности. (АПСО)

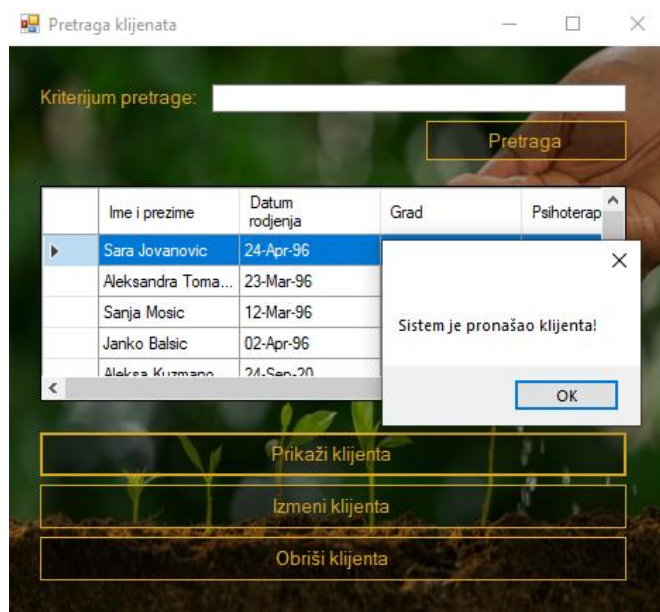
11. **Систем** тражи клијенте по задатој вредности. (СО)

12. **Систем** приказује **психотерапеуту** листу клијената и поруку: "Sistem je pronašao klijente". (ИА)



Слика 157 - Обавештење о успешном проналаску

13. **Психотерапеут бира клијента** чије податке жели да учита. (АПУСО)
14. **Психотерапеут позива систем** да учита податке о одабраном **клијенту**. (АПСО)
15. **Систем учитава** податке о одабраном **клијенту**. (СО)
16. **Систем приказује психотерапеуту** податке о **клијенту** и поруку: "Sistem je pronašao klijenta" (ИА)



Слика 158 - Обавештење о успешном проналаску података клијента

Prikaz klijenta

ID: 1

Lični podaci

Ime: Sara

Prezime: Jovanovic

Datum rođenja(dd.MM.yyyy): 24.04.1996

Prebivalište: Novi Sad

Kontakt

Broj telefona: 064-234-657

E-mail: sara@gmail.com

Ukupan dug

0

Izmeni klijenta

Obrisi klijenta

Mia Popic

Слика 159 - Приказ података изабраног клијента

Алтернативна сценарија

- 4.2. Уколико **систем** не може да нађе **клијенте**, **систем** приказује **психотерапеуту** поруку: “Klijenti koji odgovaraju zadatoj vrednosti nisu pronađeni!” Прекида се извршење сценарија. (ИА)

Pretraga klijenata

Kriterijum pretrage: dzeksi@gmail.com

Pretraga

	Ime i prezime	Datum rođenja	Grad	Psihoterapeut
Klijenti koji odgovaraju zadatoj vrednosti nisu pronađeni!				

OK

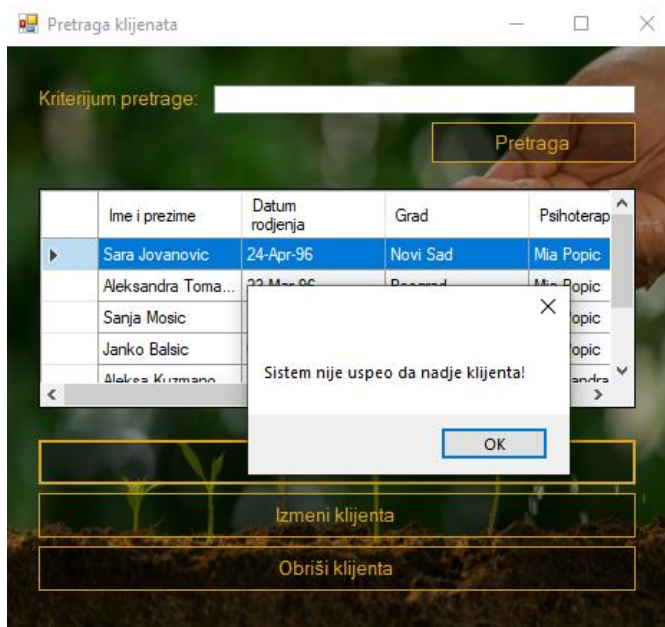
Prikaži klijenta

Izmeni klijenta

Obrisi klijenta

Слика 160 - Обавештење о неуспешном проналаску

8.2. Уколико **систем** не може да пронађе податке о **клијенту**, систем приказује **психотерапеуту** поруку: “Klijent nije pronaden!” (ИА)



Слика 161 - Обавештење о неуспешном проналаску података клијента

СК4: Измена података о клијенту

Назив СК

Измена података о клијенту

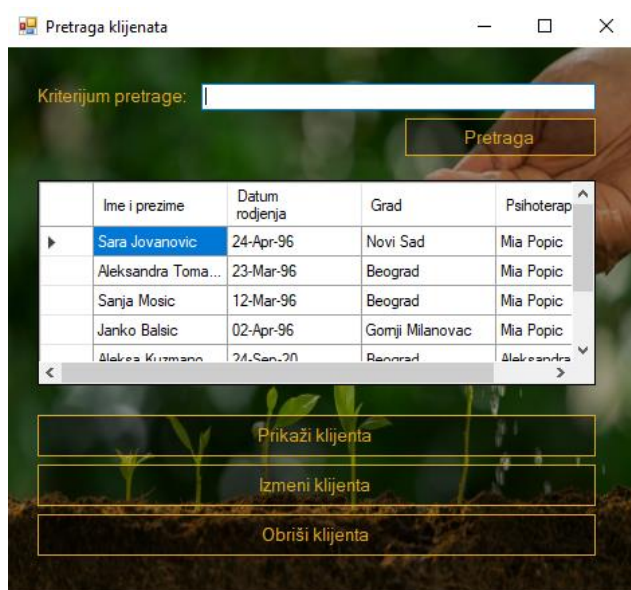
Актери СК

Психотерапеут

Учесници СК

Психотерапеут и систем (програм)

Предуслов: Систем је укључен и психотерапеут је улогован под својом шифром. Систем приказује форму за рад са клијентима. Учитана је листа клијената. Учитана је листа градова.



Pretraga klijenata

Kriterijum pretrage:

Pretraga

	Ime i prezime	Datum rođenja	Grad	Psihoterap
▶	Sara Jovanovic	24-Apr-96	Novi Sad	Mia Popic
	Aleksandra Toma...	23-Mar-96	Beograd	Mia Popic
	Sanja Masic	12-Mar-96	Beograd	Mia Popic
	Janko Balsic	02-Apr-96	Gornji Milanovac	Mia Popic
	Aleksa Kirmann	24-Sep-90	Beograd	Aleksandra

Prikaži klijenta

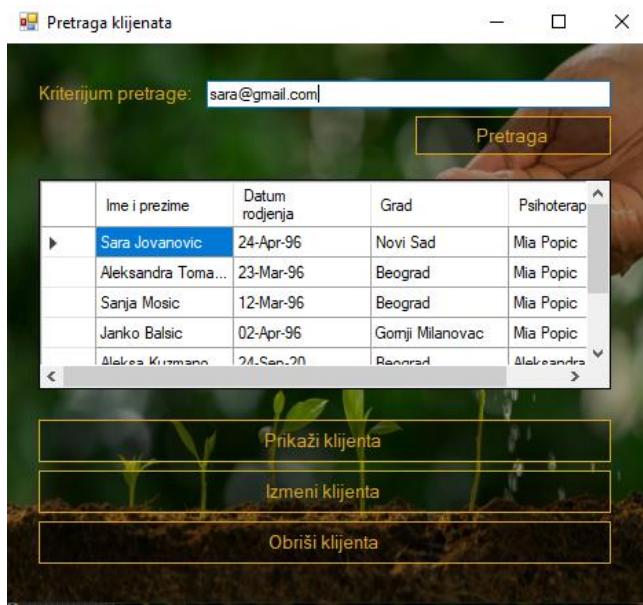
Izmeni klijenta

Obriši klijenta

Слика 162 - Приказ форме за претрагу клијената

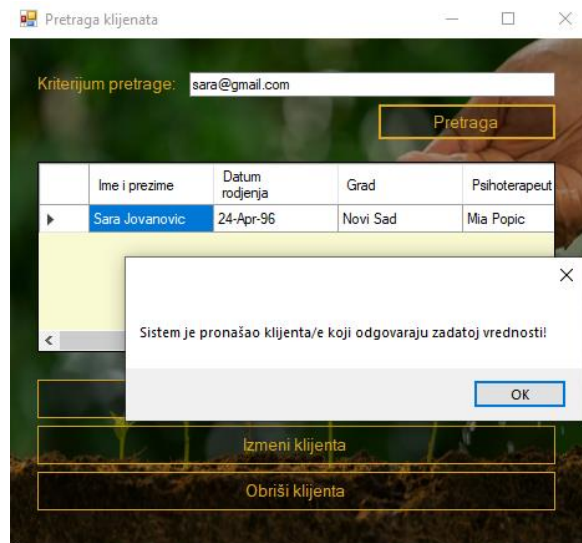
Основни сценарио СК

27. Психотерапеут уноси вредност по којој претражује клијенте. (АПУСО)



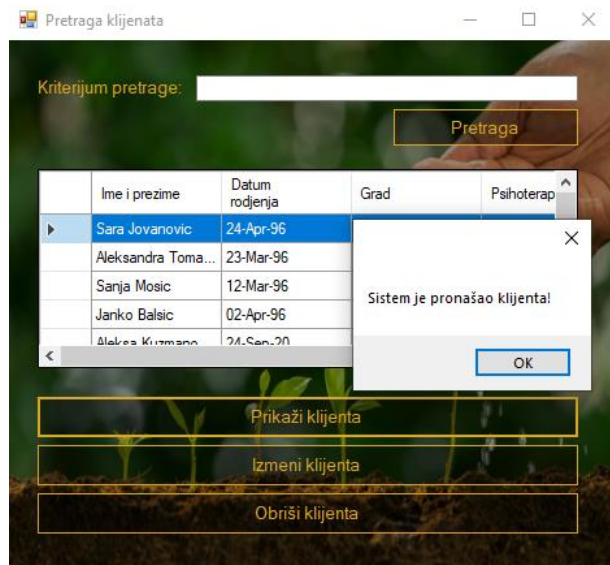
Слика 163 - Претрага према одабраном критеријуму

28. **Психотерапеут** позива **систем** да нађе **клијенте** по задатој вредности. (АПСО)
29. **Систем** тражи **клијенте** по задатој вредности. (СО)
30. **Систем** приказује **психотерапеуту** листу **клијената** и поруку: "Sistem je pronašao klijente". (ИА)

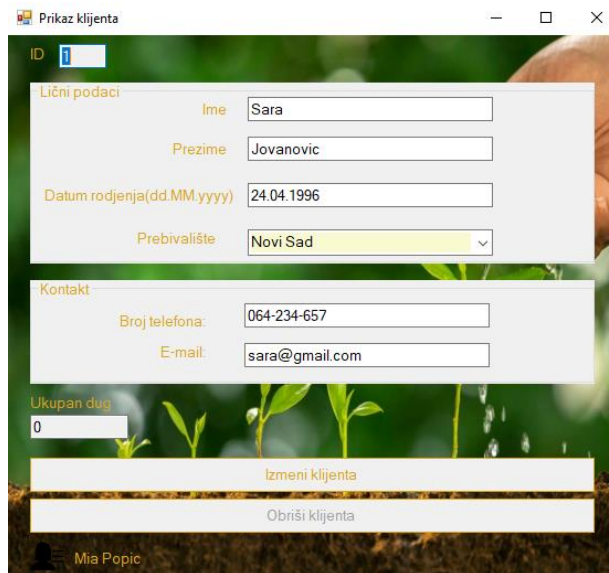


Слика 164 - Обавештење о успешном проналаску

31. **Психотерапеут** бира **клијента** чије податке жели да измени. (АПУСО)
32. **Психотерапеут** позива **систем** да учита податке о одабраном **клијенту**. (АПСО)
33. **Систем** учитава податке о одабраном **клијенту**. (СО)
34. **Систем** приказује **психотерапеуту** податке о **клијенту** и поруку: "Sistem je pronašao klijenta". (ИА)



Слика 165 - Обавештење о успешном проналаску података клијента



Слика 166 - Приказ података одабраног клијента

35. **Психотерапеут** мења податке о **клијенту**. (АПУСО)

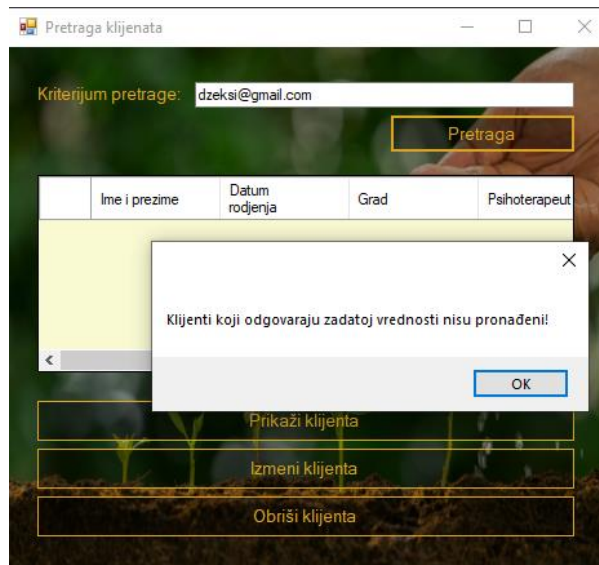
Слика 167 - Измена података одабраног клијента

36. **Психотерапеут контролише** да ли је коректно унео податке о **клијенту**. (АНСО)
37. **Психотерапеут позива систем** да запамти податке о **клијенту**. (АПСО)
38. **Систем памти** податке о **клијенту**. (СО)
39. **Систем приказује психотерапеуту** поруку: "Klijent uspešno izmenjen!" (ИА)

Слика 168 - Обавештење о успешној измени података

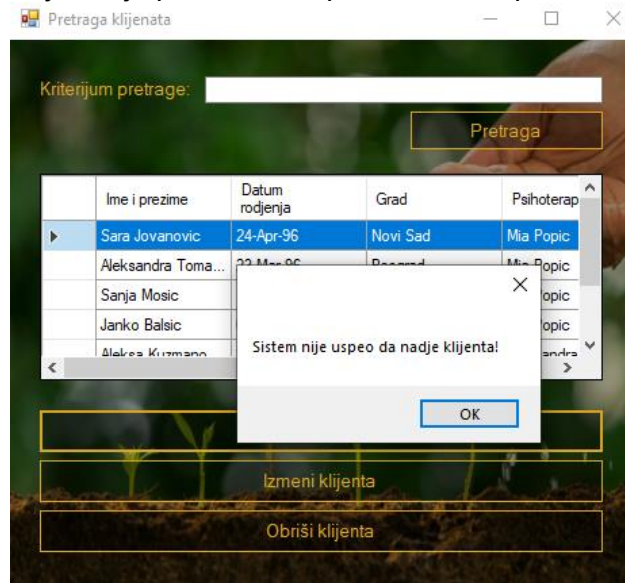
Алтернативна сценарија

4.1. Уколико **систем** не може да нађе **клијенте** он приказује **психотерапеуту** поруку: "Klijenti koji odgovaraju zadatoj vrednosti nisu pronađeni!" Прекида се извршење сценарија. (ИА)



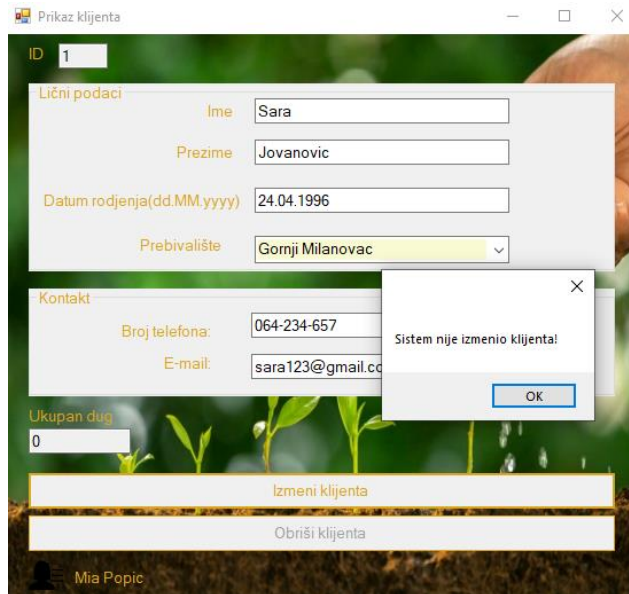
Слика 169 - Обавештење о неуспешном проналаску

8.1. Уколико **систем** не може да пронађе податке о **клијенту**, **систем** приказује **психотерапеуту** поруку: "Klijent nije pronaden!" Прекида се извршење сценарија. (ИА)



Слика 170 - Обавештење о неуспешном проналаску података клијента

26.1. Уколико **систем** не може да запамти податке о **клијенту** он приказује **психотерапеуту** поруку: "Klijent nije izmenjen!" (ИА)



Слика 171 - Обавештење о неуспешној измени података

СК5: Брисање клијента

Назив СК

Брисање клијента

Актери СК

Психотерапеут

Учесници СК

Психотерапеут и систем (програм)

Предуслов: Систем је укључен и психотерапеут је улогован под својом шифром. Систем приказује форму за рад са клијентима. Учитана је листа клијената.

Kriterijum pretrage:

Pretraga

	Ime i prezime	Datum rođenja	Grad	Psihoterap
▶	Sara Jovanovic	24-Apr-96	Novi Sad	Mia Popic
	Aleksandra Toma...	23-Mar-96	Beograd	Mia Popic
	Sanja Masic	12-Mar-96	Beograd	Mia Popic
	Janko Balsic	02-Apr-96	Gornji Milanovac	Mia Popic
	Aleksa Kuzmanovic	24-Sep-90	Beograd	Aleksandra

Prikaži klijenta

Izmeni klijenta

Obriši klijenta

Слика 172 - Приказ форме за претрагу клијената

Основни сценарио СК

23. Психотерапеут уноси вредност по којој претражује клијенте. (АПУСО)

Kriterijum pretrage: sara@gmail.com

Pretraga

	Ime i prezime	Datum rođenja	Grad	Psihoterap
▶	Sara Jovanovic	24-Apr-96	Novi Sad	Mia Popic
	Aleksandra Toma...	23-Mar-96	Beograd	Mia Popic
	Sanja Masic	12-Mar-96	Beograd	Mia Popic
	Janko Balsic	02-Apr-96	Gornji Milanovac	Mia Popic
	Aleksa Kuzmanovic	24-Sep-90	Beograd	Aleksandra

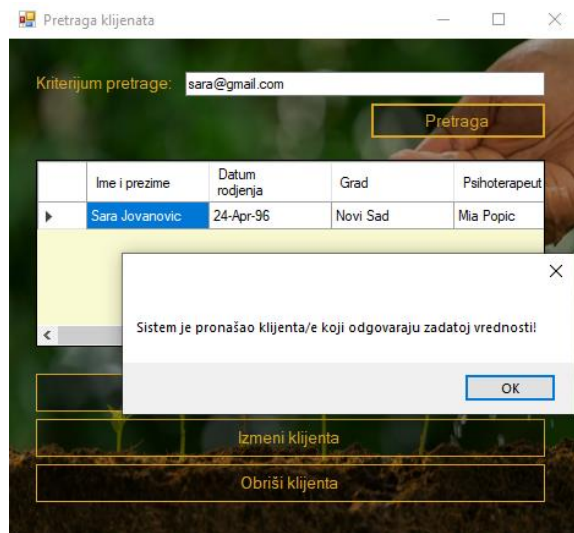
Prikaži klijenta

Izmeni klijenta

Obriši klijenta

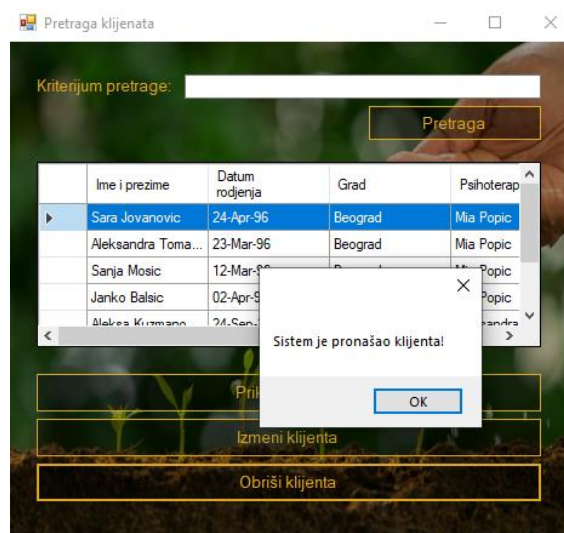
Слика 173 - Претрага према изабраном критеријуму

24. **Психотерапеут** **позива** **систем** да нађе **клијенте** по задатој вредности. (АПСО)
25. **Систем** **тражи** **клијенте** по задатој вредности. (СО)
26. **Систем** **приказује** **психотерапеуту** листу **клијената** и поруку: “Систем је пронашао клијенте”. (ИА)



Слика 174 - Обавештење о успешном проналаску

27. **Психотерапеут** **бира** **клијента** ког жели да обрише. (АПУСО)
28. **Психотерапеут** **позива** **систем** да учита податке о одабраном **клијенту**. (АПСО)
29. **Систем** **учитава** податке о одабраном **клијенту**. (СО)
30. **Систем** **приказује** **психотерапеуту** податке о **клијенту** и поруку: “Систем је пронашао клијента”. (ИА)



Слика 175 - Обавештење о успешном проналаску података клијента

The screenshot shows a web application window titled "Prikaz klijenta". At the top, there is an "ID" field with the value "1". Below this, the "Lični podaci" (Personal data) section contains fields for "Ime" (Name) with "Sara", "Prezime" (Surname) with "Jovanovic", "Datum rođenja (dd.MM.yyyy)" (Date of birth) with "24.04.1996", and "Prebivalište" (Residence) with a dropdown menu showing "Beograd". The "Kontakt" (Contact) section has fields for "Broj telefona:" (Phone number) with "064-234-657" and "E-mail:" with "sara123@gmail.com". Below these is the "Ukupan dug" (Total debt) field with the value "0". At the bottom, there are two buttons: "Izmeni klijenta" (Edit client) and "Obriši klijenta" (Delete client). The user "Mia Popic" is logged in, as indicated by a profile icon and name at the bottom left.

Слика 176 - Приказ података изабраног клијента

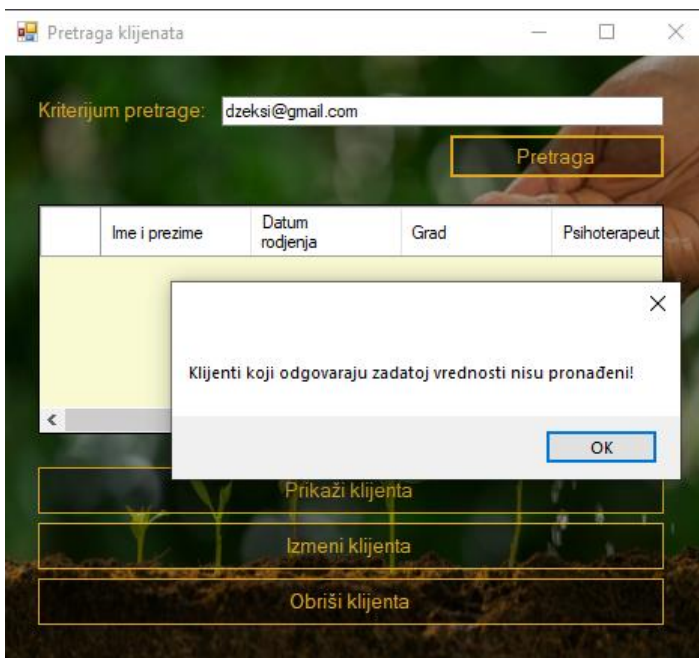
31. Психотерапеут позива систем да обрише клијента. (АПСО)
32. Систем брише клијента. (СО)
33. Систем приказује психотерапеуту поруку: "Klijent obrisan!" (ИА)

This screenshot shows the same "Prikaz klijenta" window as before, but with a small modal dialog box open over the "Obriši klijenta" button. The dialog box has a title bar with a close button (X) and contains the text "Sistem je obrisao klijenta!" (The system has deleted the client!). There is an "OK" button at the bottom of the dialog. The background window remains the same, showing the client data for Sara Jovanovic.

Слика 177 - Обавештење о успешном брисању изабраног клијента

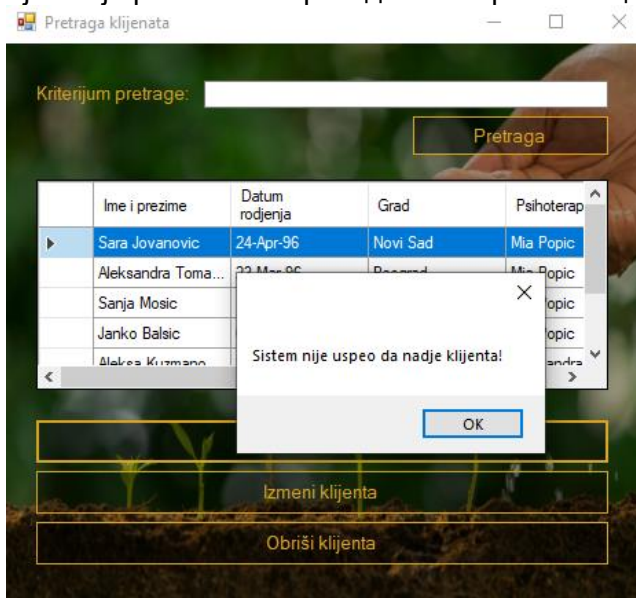
Алтернативна сценарија

4.1. Уколико **систем** не може да нађе **клијенте** он приказује **психотерапеуту** поруку: “Klijenti koji odgovaraju zadatoj vrednosti nisu pronađeni!” Прекида се извршење сценарија. (ИА)



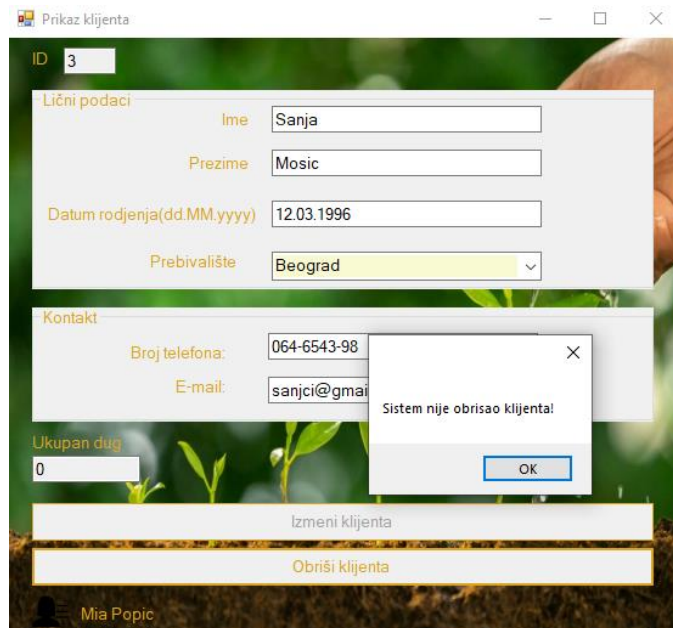
Слика 178 - Обавештење о неуспешном проналаску

8.1. Уколико **систем** не може да пронађе податке о **клијенту**, **систем** приказује **психотерапеуту** поруку: “Klijent nije pronađen!” Прекида се извршење сценарија. (ИА)



Слика 179 - Обавештење о неуспешном проналаску података клијента

22.1. Уколико **систем** не може да обрише **клијента** он приказује **психотерапеуту** поруку: “Klijent nije obrisan!” (ИА)



Слика 180 - Обавештење о неуспешном брисању изабраног клијента

СК6: Унос нове сеансе

Назив СК

Унос нове **сеансе**

Актори СК

Психотерапеут

Учесници СК

Психотерапеут и **систем** (програм)

Предуслов: **Систем** је укључен и **психотерапеут** је улогован под својом шифром. **Систем** приказује форму за рад на **сеансама** . Учитана је листа клијената и ID сеансе . Учитане су врста и облик психотерапије.

Unos nove seanse

ID: 1

Trajanje seanse:

Datum(format dd.MM.yyyy):

--Izaberi vrstu terapije--

Tema:

Generalni cilj:

Oblik psihoterapije: Grupa terapija

Unos stavke

Klijent: Aleksandra Tomasevic

Potvrdi izmene

☐ Bio/la na seansi

Cena seanse:

☐ Placena seansa

Unesi stavku

Izmeni odabranu stavku

Obrisi odabranu stavku

Mia Popic

Sacuvaj seansu

Слика 181 - Приказ форме уноса нове сеансе

Основни сценарио СК

11. Психотерапеут попуњава податке о сеанси. (АПУСО)

Unos nove seanse

ID: 1

Trajanje seanse: 50

Datum(format dd.MM.yyyy): 17.07.2020

REBT

Tema: tema1

Generalni cilj: cilj1

Uradjen domaci: da

Pohadjanje radionica: da

Napredak: veliki

Oblik psihoterapije: Individualna terapija

Unos stavke

Klijent:

Potvrdi izmene

☐ Bio/la na seansi

Cena seanse:

☐ Placena seansa

Unesi stavku

Izmeni odabranu stavku

Obrisi odabranu stavku

Rbr	Klijent	Prisustvo	Cena seansa	Placena seansa	Status
1	Bojana ...	<input checked="" type="checkbox"/>	1500	<input type="checkbox"/>	Uneta

Mia Popic

Sacuvaj seansu

Слика 182 - Унос нове РЕБТ сеансе

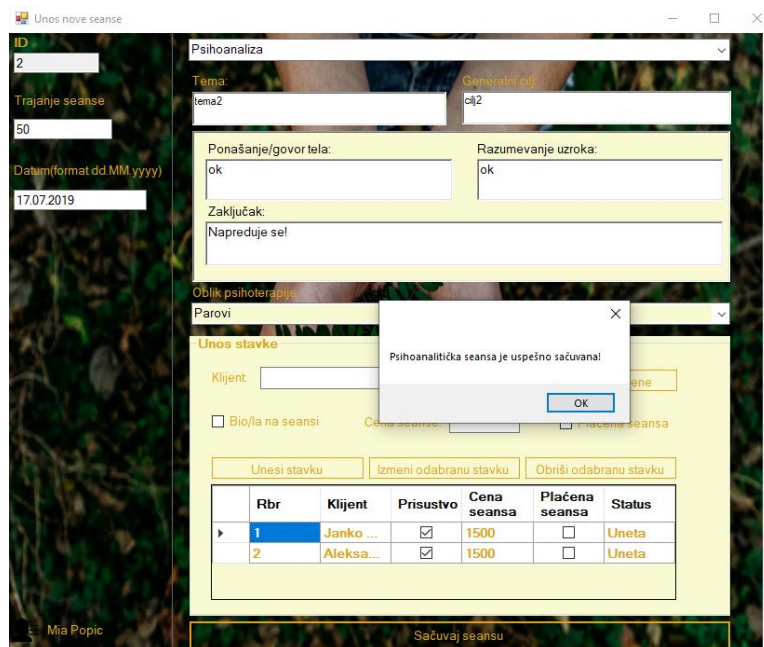
Rbr	Klijent	Prisustvo	Cena seansa	Plaćena seansa	Status
1	Janko ...	<input checked="" type="checkbox"/>	1500	<input checked="" type="checkbox"/>	Uneta
2	Aleksa ...	<input checked="" type="checkbox"/>	1500	<input checked="" type="checkbox"/>	Uneta

Слика 183 - Унос нове психоаналитичке сеансе

12. **Психотерапеут контролише** да ли је коректно унео податке о **сеанси**. (АНСО)
13. **Психотерапеут позива систем** да запамти податке о **сеанси**. (АПСО)
14. **Систем памти** податке о **сеанси**. (СО)
15. **Систем приказује психотерапеуту** поруку: "Seansa uspešno sačuvana!" (ИА)

Rbr	Klijent	Prisustvo	Cena seansa	Plaćena seansa	Status
1	Bojana ...	<input checked="" type="checkbox"/>	1500	<input type="checkbox"/>	Uneta

Слика 184 - Обавештење о успешном уносу РЕБТ сеансе



Слика 185 - Обавештење о успешном уносу психоаналитичке сеансе

Алтернативна сценарија

5.1. Уколико **систем** не може да запамти податке о **сеанси** он приказује **психотерапеуту** поруку: "Seansa nije sačuvana!" (IA)

Unos nove seanse

ID: 1

Trajanje seanse: 50

Datum(format dd.MM.yyyy): 17.07.2020

REBT

Tema: Tema 1

Generelni cilj: cilj 1

Urađjen domaći: da

Pohodjanje radionica: da

Napredak: veliki

Oblik psihoterapije: Individualna terapija

Unos stavke

Klijent: [empty]

☐ Bio/la na seansi

Cena seansa: 1500

☐ Plaćena seansa

Unesi stavku

Izmeni odabranu stavku

Obrisi odabranu stavku

Rbr	Klijent	Prisustvo	Cena seansa	Plaćena seansa	Status
1	Bojana ...	<input checked="" type="checkbox"/>	1500	<input type="checkbox"/>	Uneta

Mia Popic

Sačuvaj seansu

Слика 186 - Обавештење о неуспешном уносу РЕБТ сеансе

Unos nove seanse

ID: 1

Trajanje seanse: 50

Datum(format dd.MM.yyyy): 17.07.2020

Psihoanaliza

Tema: Tema 1

Generelni cilj: cilj 1

Ponašanje/govor tela: ok

Razumevanje uzroka: ok

Zaključak: Napreduje se!

Oblik psihoterapije: Parovi

Unos stavke

Klijent: [empty]

☐ Bio/la na seansi

Cena seansa: 1500

☐ Plaćena seansa

Unesi stavku

Izmeni odabranu stavku

Obrisi odabranu stavku

Rbr	Klijent	Prisustvo	Cena seansa	Plaćena seansa	Status
1	Janko ...	<input checked="" type="checkbox"/>	1500	<input type="checkbox"/>	Uneta
2	Aleksa ...	<input checked="" type="checkbox"/>	1500	<input type="checkbox"/>	Uneta

Mia Popic

Sačuvaj seansu

Слика 187 - Обавештење о неуспешном уносу психоаналитичке сеансе

СК7: Претраживање сеанси

Назив СК

Претраживање сеанси

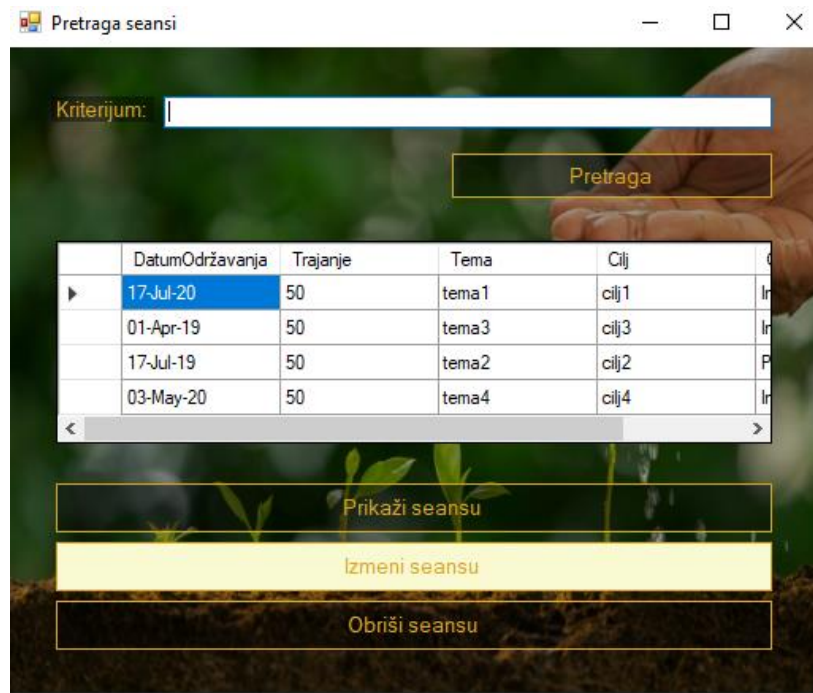
Актори СК

Психотерапеут

Учесници СК

Психотерапеут и систем (програм)

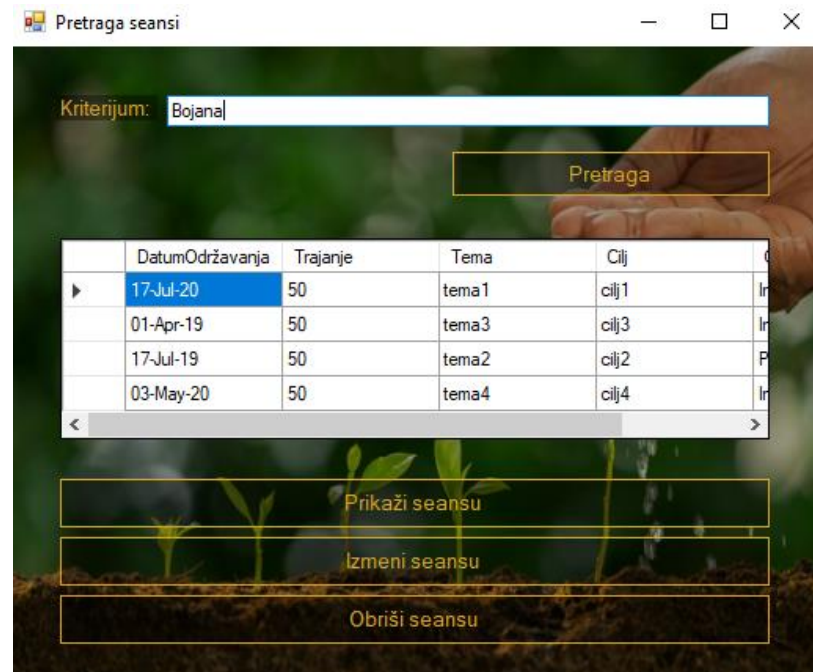
Предуслов: Систем је укључен и психотерапеут је улогован под својом шифром. Систем приказује форму за рад са сеансама. Систем приказује листу сеанси.



Слика 188 - Приказ форме за претрагу сеанси

Основни сценарио СК

17. **Психотерапеут** уноси вредност по којој претражује **сеансе**. (АПУСО)

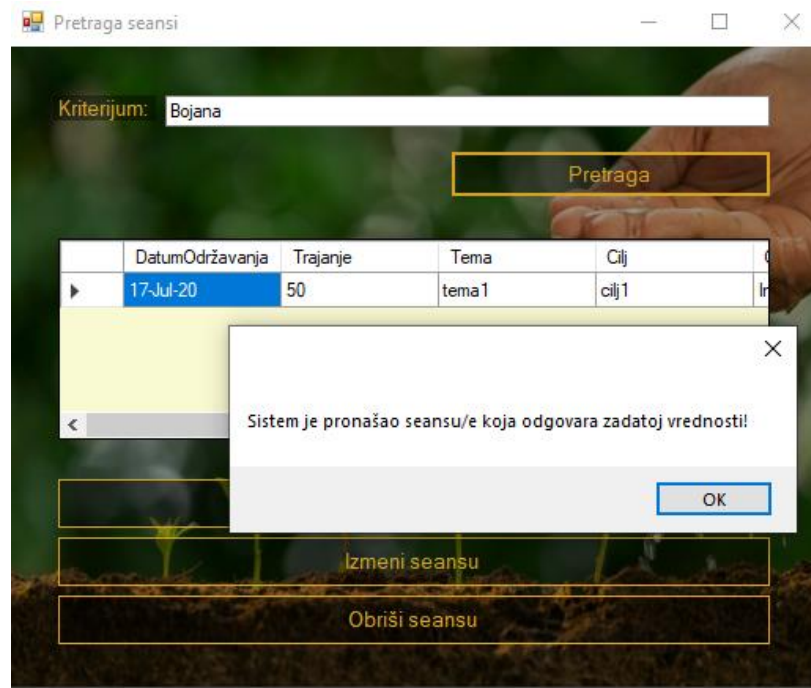


Слика 189 - Претрага сеанси према изабраном критеријуму

18. **Психотерапеут** позива **систем** да нађе **сеансе** по задатој вредности. (АПСО)

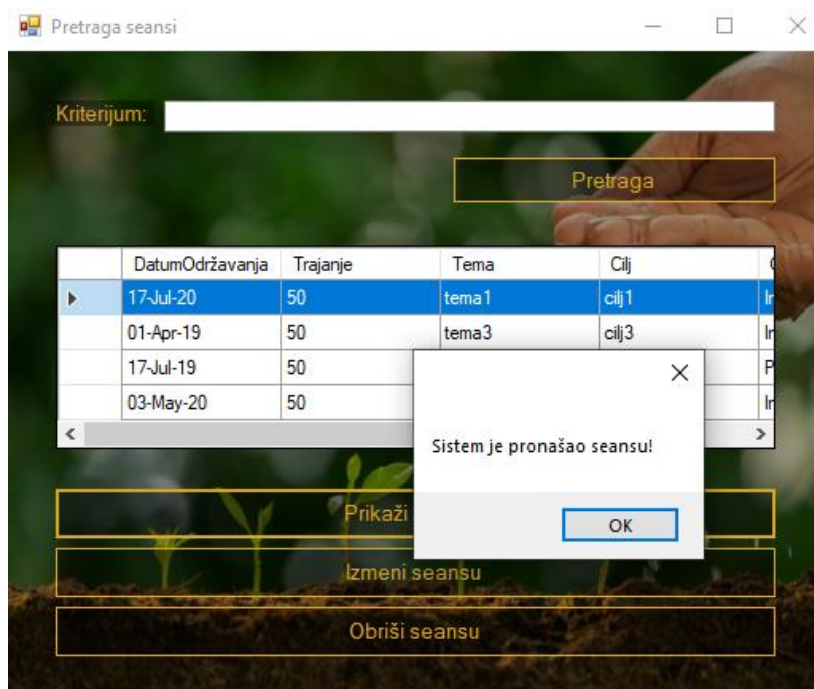
19. **Систем** тражи **сеансе** по задатој вредности. (СО)

20. Систем приказује психотерапеуту листу сеанси и поруку: “Sistem je pronašao seanse”. (ИА)

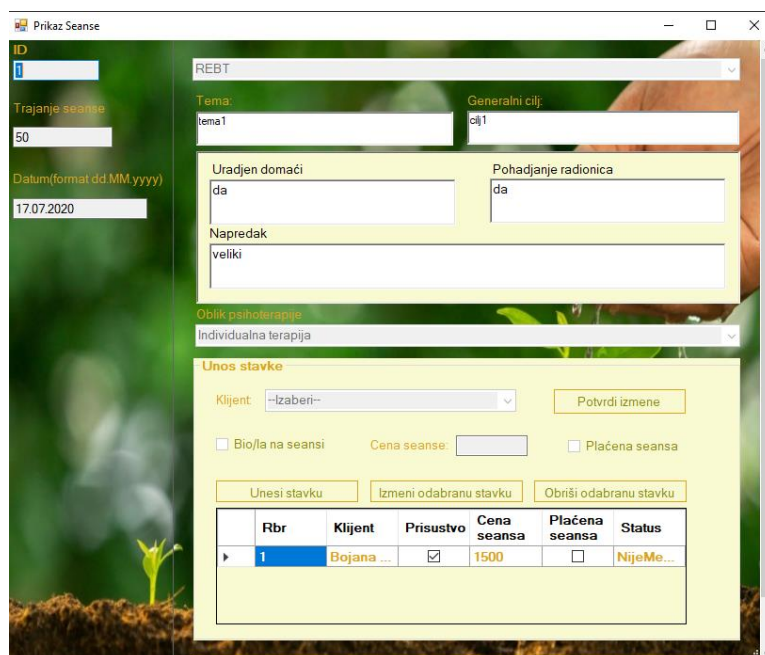


Слика 190 - Обавештење о успешном проналаску

21. Психотерапеут бира сеансу чије податке жели да учита. (АПУСО)
22. Психотерапеут позива систем да учита податке о одабраној сеанси. (АПСО)
23. Систем учитава податке о одабраној сеанси. (СО)
24. Систем приказује психотерапеуту податке о сеанси и поруку: “Sistem je pronašao seansu”. (ИА)



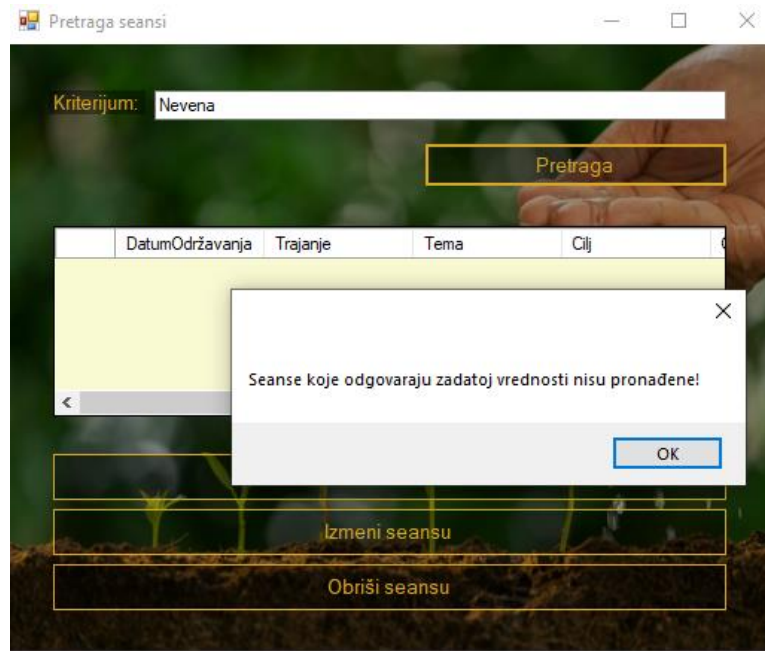
Слика 191- Обавештење о успешном проналаску података сеансе



Слика 192 - Приказ изабране сеансе

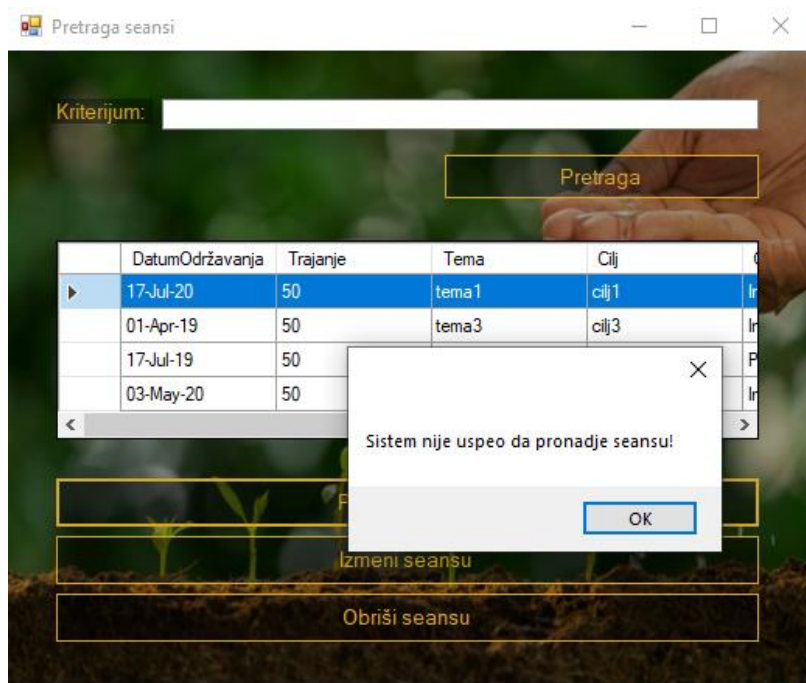
Алтернативна сценарија

4.1. Уколико **систем** не може да нађе **сеансе**, **систем** приказује **психотерапеуту** поруку: "Seanse koje odgovaraju zadatoj vrednosti nisu pronađene!" Прекида се извршење сценарија. (ИА)



Слика 193- Обавештење о неуспешном проналаску

8.1. Уколико **СИСТЕМ** не може да пронађе **сеансу** он приказује **психотерапеуту** поруку: "Seansa nije pronađena!" (IA)



Слика 194 - Обавештење о неуспешном проналаску података сеансе

СК8: Измена садржаја сеансе

Назив СК

Измена садржаја **сеансе**

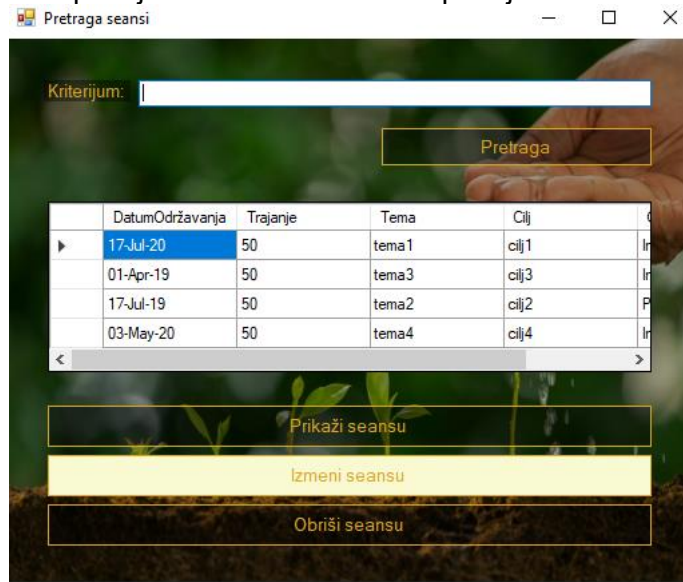
Актери СК

Психотерапеут

Учесници СК

Психотерапеут и **систем** (програм)

Предуслов: **Систем** је укључен и **психотерапеут** је улогован под својом шифром. **Систем** приказује форму за рад са **сеансама**. Учитана је листа **сеанси** као и листа клијената. Учитана је врста психотерапије као и облик психотерапије.



The screenshot shows a window titled "Pretraga seansi". At the top, there is a label "Kriterijum:" followed by an empty text input field. Below the input field is a yellow button labeled "Pretraga". Underneath the button is a table with the following data:

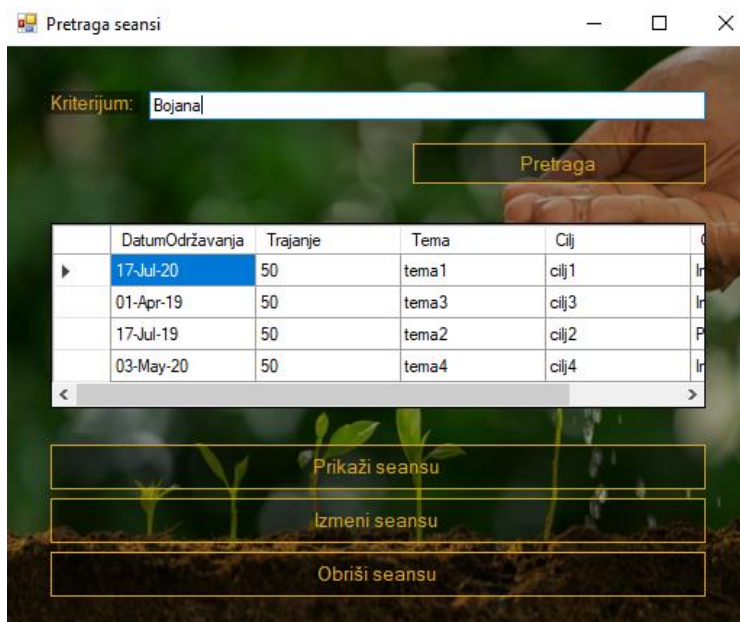
	DatumOdržavanja	Trajanje	Tema	Cilj
►	17-Jul-20	50	tema1	cilj1
	01-Apr-19	50	tema3	cilj3
	17-Jul-19	50	tema2	cilj2
	03-May-20	50	tema4	cilj4

Below the table are three yellow buttons: "Prikaži seansu", "Izmeni seansu", and "Obriši seansu".

Слика 195 - Приказ форме за претрагу сеанси

Основни сценарио СК

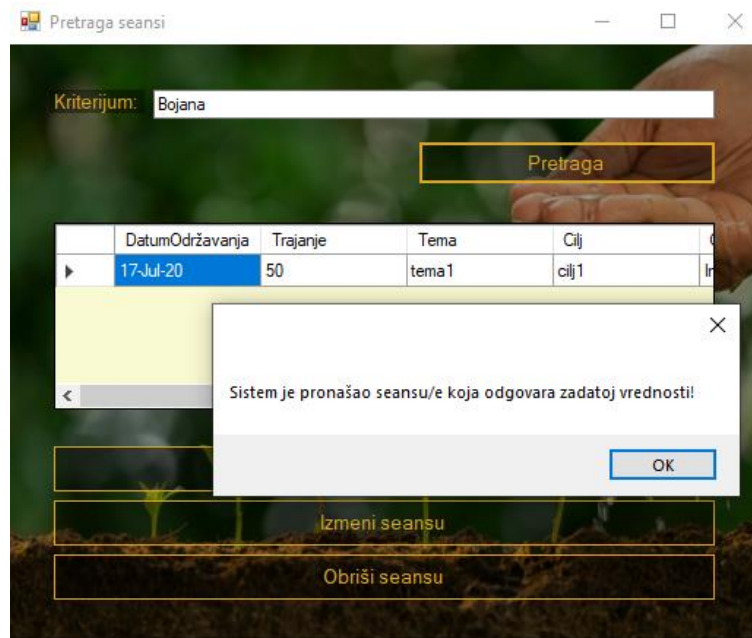
27. **Психотерапеут** уноси вредност по којој претражује **сеансе**. (АПУСО)



The screenshot shows the same "Pretraga seansi" window, but now the "Kriterijum:" input field contains the text "Bojana". The "Pretraga" button is still present. The table below it remains the same as in the previous screenshot.

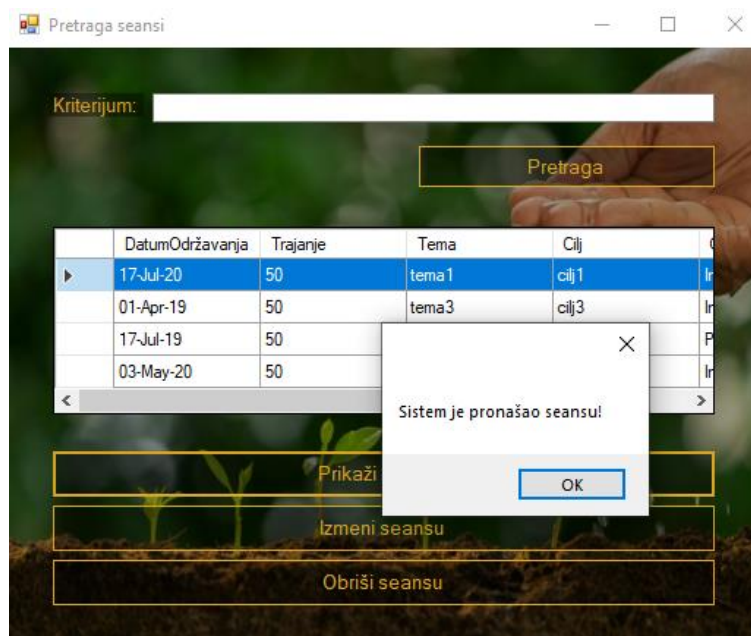
Слика 196 - Претрага према изабраном критеријуму

28. **Психотерапеут** **позива** **систем** да нађе **сеансе** по задатој вредности. (АПСО)
29. **Систем** **тражи** **сеансе** по задатој вредности. (СО)
30. **Систем** **приказује** **психотерапеуту** листу **сеанси** и поруку: “Sistem je pronašao seanse”. (ИА)

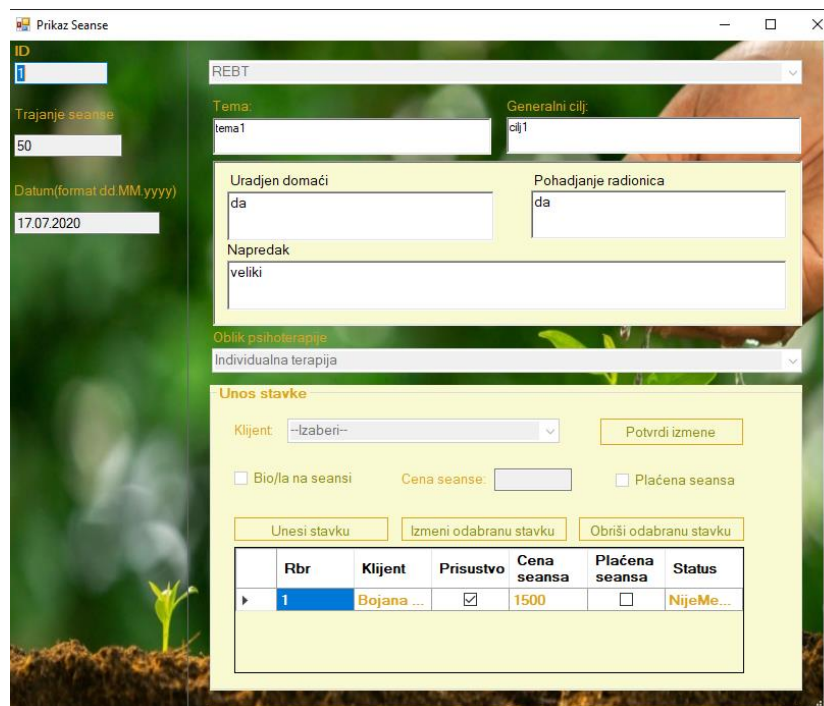


Слика 197 - Обавештење о успешном проналаску

31. **Психотерапеут** **бира** **сеансу** чије податке жели да измени. (АПУСО)
32. **Психотерапеут** **позива** **систем** да учита податке о одабраној **сеанси**. (АПСО)
33. **Систем** **учитава** податке о одабраној **сеанси**. (СО)
34. **Систем** **приказује** **психотерапеуту** податке о **сеанси** и поруку: “Sistem je pronašao seansu”. (ИА)



Слика 198 - Обавештење о успешном проналаску података сеансе



Слика 199 - Приказ изабране сеансе

35. Психотерапеут мења податке о сеанси. (АПУСО)

Prikaz Seanse

ID: 1

Trajanje seanse: 60

Datum(format dd.MM.yyyy): 17.07.2020

REBT

Tema: tema1

Generalni cilj: cilj1

Uradjen domaći: da

Pohadjanje radionica: da

Napredak: veliki

Oblik psihoterapije: Individualna terapija

Unos stavke

Klijent: [dropdown]

☐ Bio/la na seansi

Cena seanse: [input]

☐ Plaćena seansa

Potvrdi izmene

Unesi stavku

Izmeni odabranu stavku

Obriši odabranu stavku

Rbr	Klijent	Prisustvo	Cena seansa	Plaćena seansa	Status
1	Bojana ...	<input checked="" type="checkbox"/>	2000	<input type="checkbox"/>	Izmenje...

Слика 200 - Измена података изабране сеансе

36. **Психотерапеут контролише** да ли је коректно унео податке о **сеанси**. (АНСО)
37. **Психотерапеут позива систем** да запами податке о **сеанси**. (АПСО)
38. **Систем памти** податке о **сеанси**. (СО)
39. **Систем приказује психотерапеуту** поруку: "Seansa uspešno izmenjena!" (ИА)

Prikaz Seanse

ID: 1

Trajanje seanse: 60

Datum(format dd.MM.yyyy): 17.07.2020

REBT

Tema: tema1

Generalni cilj: cilj1

Uradjen domaći: da

Pohadjanje radionica: da

Napredak: veliki

Oblik psihoterapije: Individualna terapija

Unos stavke

Klijent: [dropdown]

☐ Bio/la na seansi

Cena seanse: [input]

☐ Plaćena seansa

Potvrdi izmene

Unesi stavku

Izmeni odabranu stavku

Obriši odabranu stavku

Rbr	Klijent	Prisustvo	Cena seansa	Plaćena seansa	Status
1	Bojana ...	<input checked="" type="checkbox"/>	2000	<input type="checkbox"/>	Izmenje...

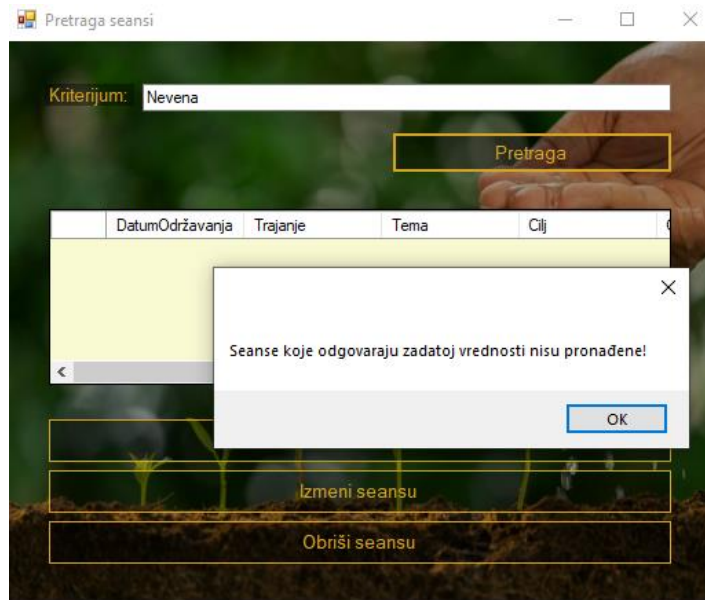
REBT seansa je uspešno izmenjena!

OK

Слика 201 - Обавештење о успешној измени података РЕБТ сеансе

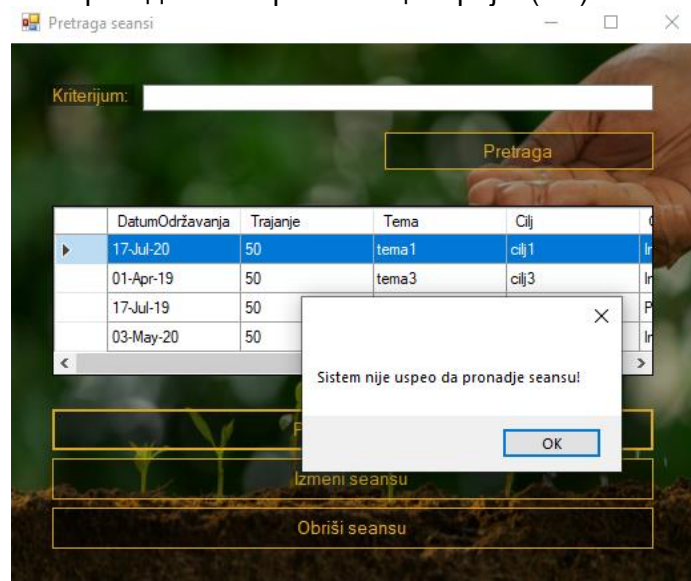
Алтернативна сценарија

4.1. Уколико **систем** не може да нађе **сеансе** он приказује **психотерапеуту** поруку: “Seanse koje odgovaraju zadatoj vrednosti nisu pronadene!”. Прекида се извршење сценарија. (ИА)



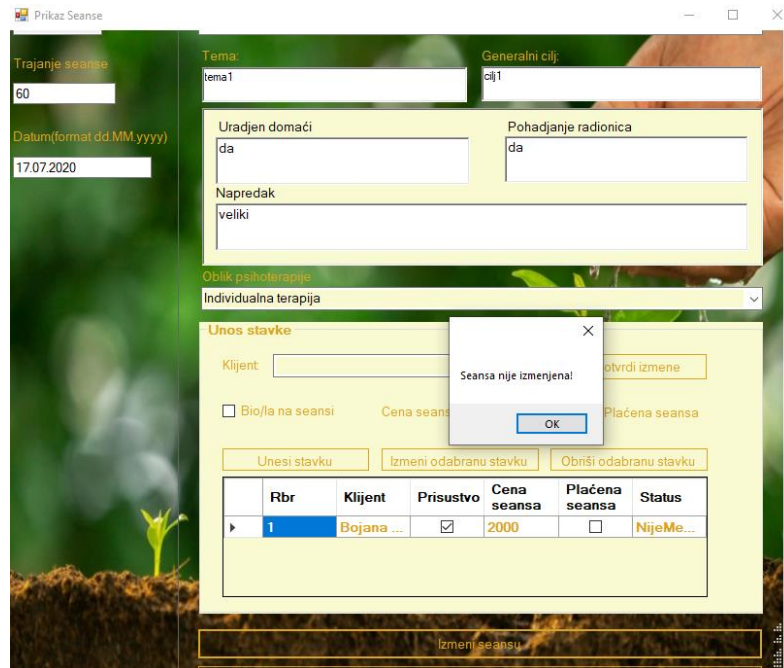
Слика 202 - Обавештење о неуспешном проналаску

8.1 Уколико **систем** не може да пронађе **сеансу** он приказује **психотерапеуту** поруку: “Seansa nije pronadena!”. Прекида се извршење сценарија. (ИА)



Слика 203 - Обавештење о неуспешном проналаску података сеансе

13.1. Уколико **систем** не може да запамти податке о **сеанси** он приказује **психотерапеуту** поруку: “Seansa nije izmenjena!”. (ИА)



Слика 204 - Обавештење о неуспешној измени података

СК9: Брисање сеансе

Назив СК

Брисање сеансе

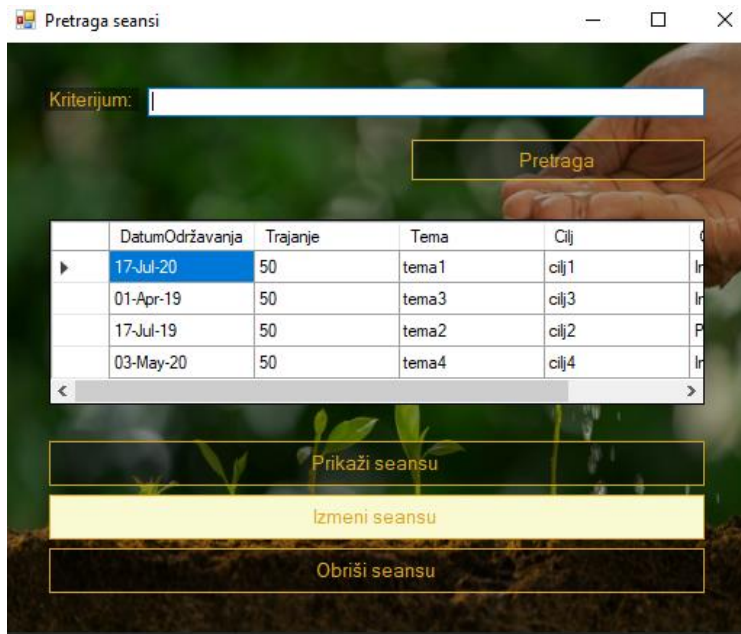
Актери СК

Психотерапеут

Учесници СК

Психотерапеут и систем (програм)

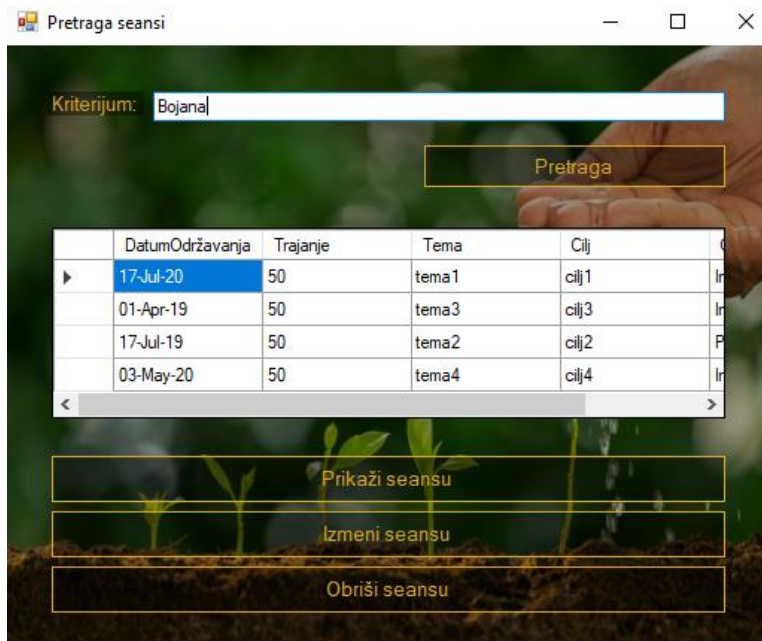
Предуслов: Систем је укључен и психотерапеут је улогован под својом шифром. Систем приказује форму за рад са сеансама. Учитана је листа сеанси.



Слика 205 - Приказ форме за претрагу сеанси

Основни сценарио СК

23. **Психотерапеут** уноси вредност по којој претражује **сеансе**. (АПУСО)

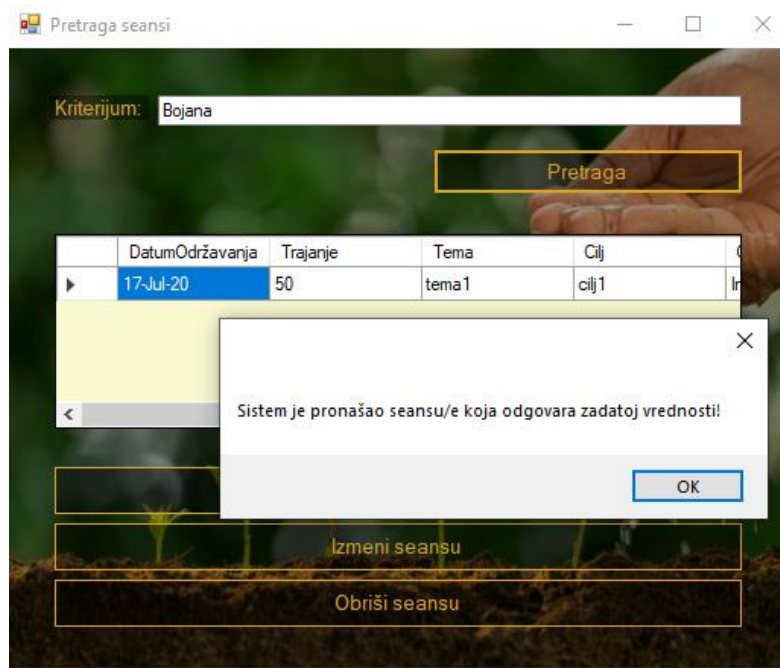


Слика 206 - Претрага сеанси према унесеном критеријуму

24. **Психотерапеут** позива **систем** да нађе **сеансе** по задатој вредности. (АПСО)

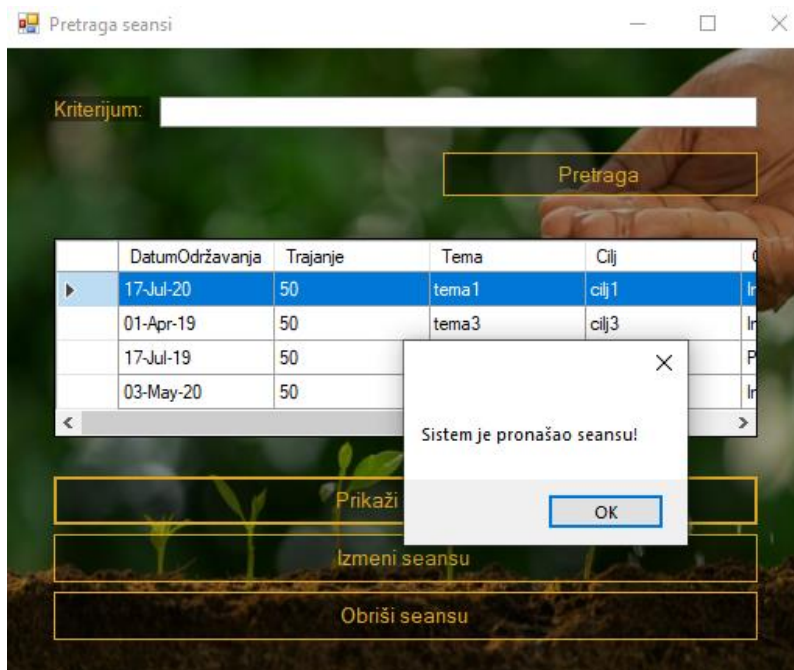
25. **Систем** тражи **сеансе** по задатој вредности. (СО)

26. **Систем** приказује **психотерапеуту** листу **сеанси** и поруку: "Sistem je pronašao seanse". (ИА)



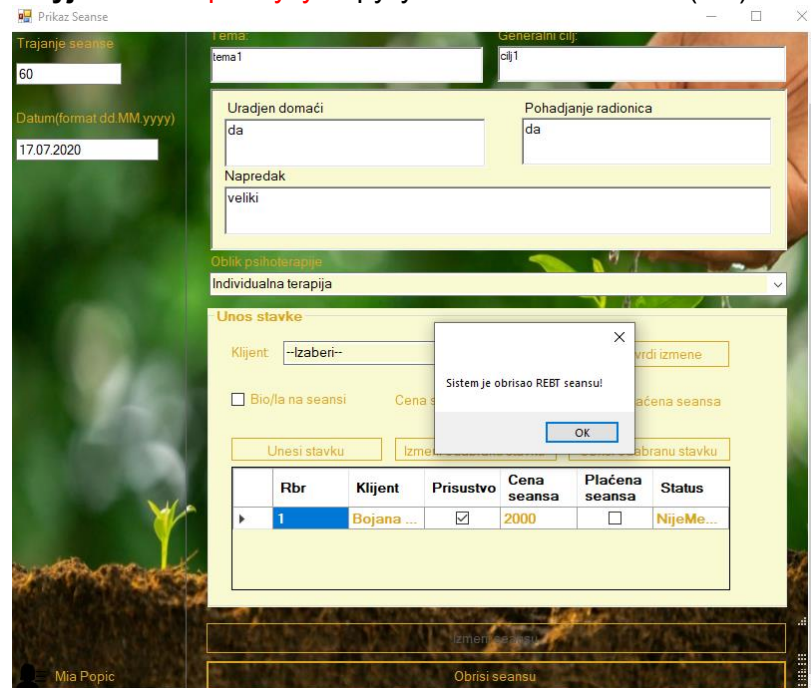
Слика 207 - Обавештење о успешном проналаску

27. **Психотерапеут** бира **сеансу** чије податке жели да учита. (АПУСО)
28. **Психотерапеут** позива **систем** да учита податке о одабраној **сеанси**. (АПСО)
29. **Систем** тражи податке о одабраној **сеанси**. (СО)
30. **Систем** приказује **психотерапеуту** податке о **сеанси** и поруку: „Sistem je pronašao seansu“. (ИА)



Слика 208 - Обавештење о успешном проналаску података сеансе

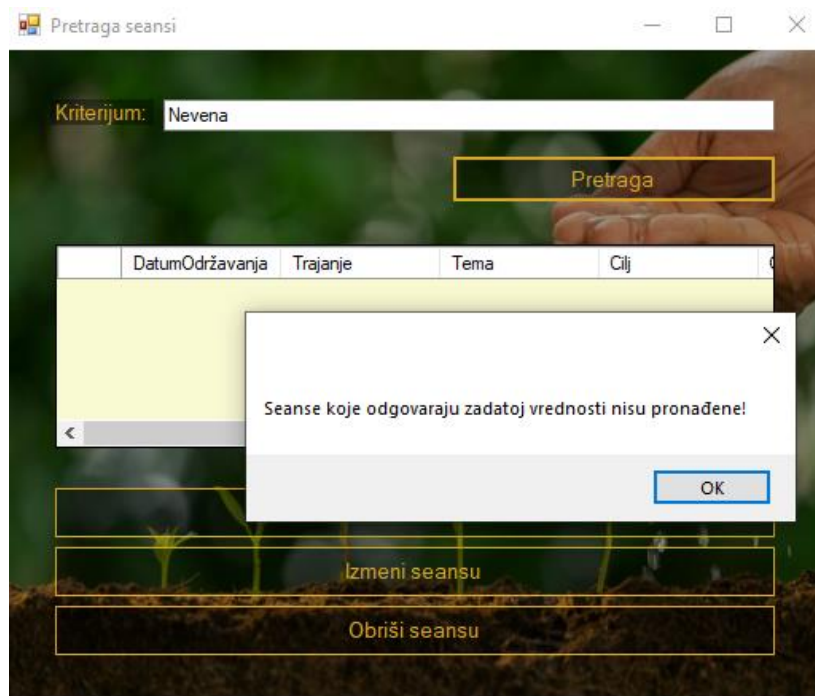
31. **Психотерапеут** **позива** **систем** да обрише одабрану **сеансу**. (АПСО)
32. **Систем** **брише** **сеансу**. (СО)
33. **Систем** **приказује** **психотерапеуту** поруку: "Seansa obrisana!" (ИА)



Слика 209 - Обавештење о успешном брисању сеансе

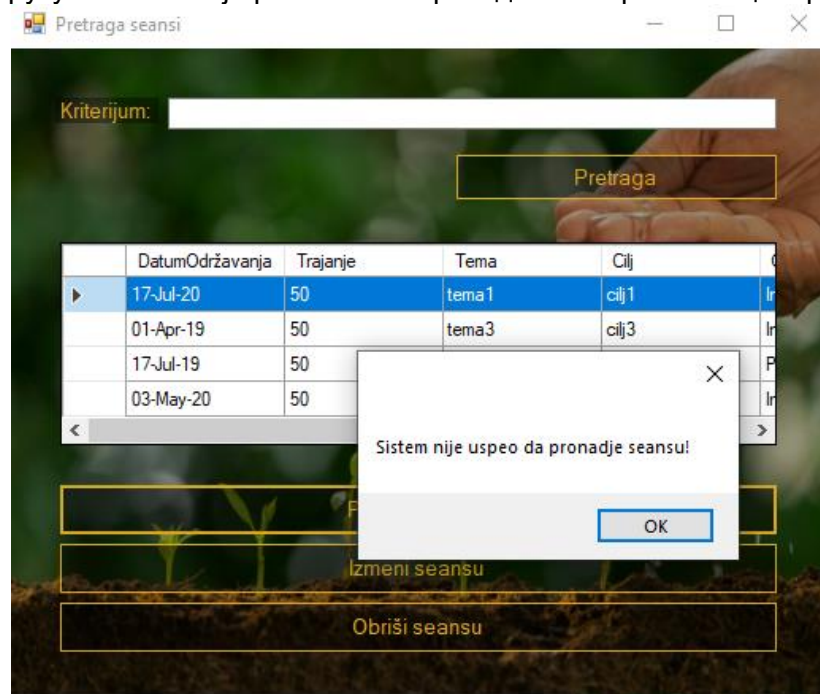
Алтернативна сценарија

4.1. Уколико **систем** не може да нађе **сеансе** он приказује **психотерапеуту** поруку: "Seanse koje odgovaraju zadatoj vrednosti nisu pronadene!". Прекида се извршење сценарија. (ИА)



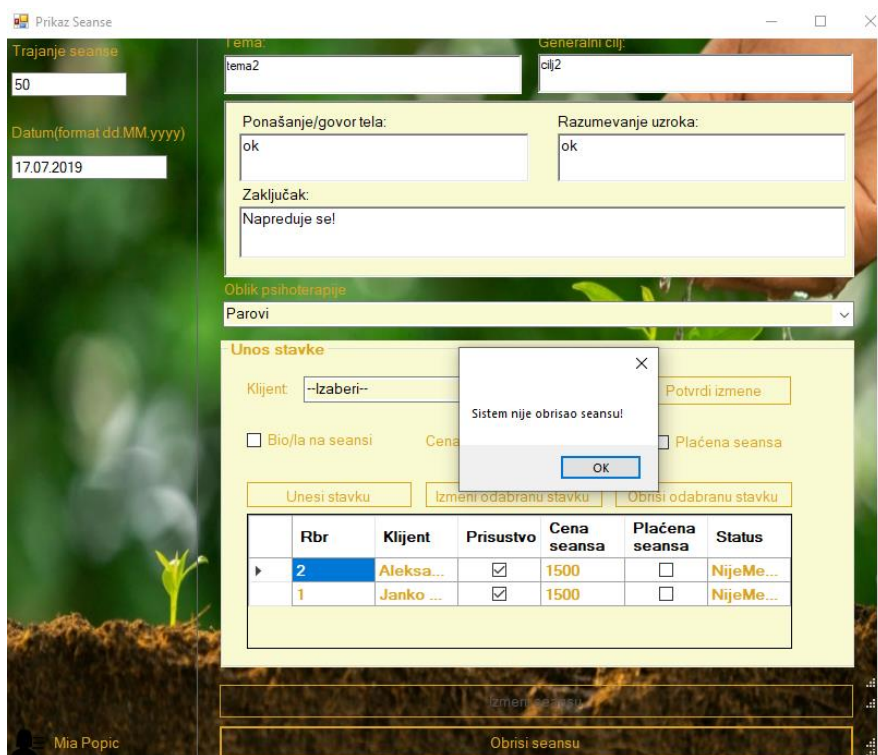
Слика 210 - Обавештење о неуспешном проналаску

8.1 Уколико **СИСТЕМ** не може да пронађе податке о **сеанси** он приказује **психотерапеуту** поруку: “Seansa nije pronađena!” Прекида се извршење сценарија. (ИА)



Слика 211 - Обавештење о неуспешном проналаску података сеансе

11.1. Уколико **СИСТЕМ** не може да обрише **сеансу** он приказује **психотерапеуту** поруку: “Seansa nije obrisana!”. (ИА)

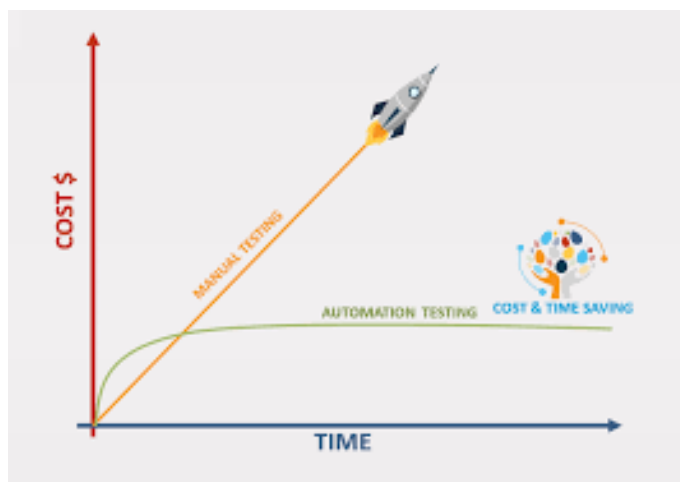


Слика 212 - Обавештење о неуспешном брисању

11.Фаза тестирања

Тестирање апликације се може вршити мануелно и аутоматски. Мануелно тестирање се врши у више корака: Потребно је да се апликација покрене, затим је најчешће потребно попунити форму за логин, доћи до форме где се налази функција за тестирање, попунити ту форму, кликнути на дугме и тек онда проверити шта се добија као резултат. Међутим, у пракси је потребно много пута поновити ове кораке да би се тестирала функција за друге вредности које се уносе или док се не открије зашто функција не враћа оно што би требало. Проблем се утростручује када у апликацији постоје десетине функција које треба тестирати.

Из овога се може закључити да је једна од мана ове врсте тестирања споро извршавање. Како апликација буде расла у својој комплексности и величини и трошкови мануелног тестирања ће експоненцијално расти тј. коришћење овакве врсте тестирања биће неисплативо.



Слика 213 - Мануелно и аутоматско тестирање

Са друге стране, аутоматски тестови подразумевају писање кода у коме се директно позива функција за тестирање и где се мењају њени инпути, а лако и брзо проверавају добијени резултати. Овакви тестови се могу покретати безброј пута, све док се не добије жељени резултат, и што је најбитније, тестирана функција ће савршено радити а да цела апликација ниједном није била покренута. Време потребно да се изврши један овакав тест је углавном испод секунде, највише неколико секунди.

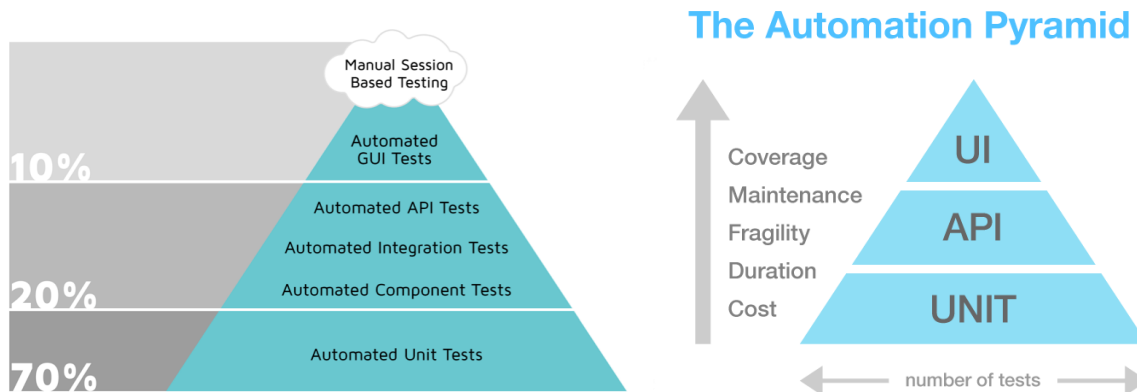
Још неке предности аутоматског тестирања у односу на мануелно:

- Написане функције се могу тестирати безброј пута, за много краће време
- Могу се пронаћи и решити сви багови пре покретања саме апликације
- Омогућују самопоуздано покретање апликације
- Омогућују refactoring са самопоуздањем, тј. мењање структуре кода (нпр. промена назива методе или одвајање пар линија кода у приватну методу) са циљем да код буде одрживији и чистији, при чему се не мења функционалност истог.
- Помажу програмерима да се више фокусирају на квалитет написаног кода.

Постоје три врсте аутоматских тестова: Unit тестови, Интеграциони тестови и End-To-End

ТЕСТОВИ.

- Unit тестови тестирају неку логичку јединицу апликације без њених екстерних зависности попут фајлова, база података, веб сервиса итд. Они се брзо пишу и извршавају, али нису у потпуности поуздани, јер тестирају код ван реалних података који су део апликације.
- Ту наступају интеграциони тестови који тестирају класу или неку другу компоненту укључујући њене екстерне зависности. Њима треба мало више времена да се изврше, али су поузданији.
- End-To-End тестови тестирају апликацију кроз коришћење корисничког интерфејса. Постоји доста алата помоћу којих се реализују ови тестови, најпопуларнији је Selenium. Selenium је алат који омогућује снимање коришћења КИ од стране корисника и анализирање снимака контролишући да ли се апликација понашала како би требало. Они су најпоузданији, али и најспорији. Такође, мале промене у КИ-су могу лако збунити написан тест.



Слика 214 - Тест пирамида

Тест пирамида налаже да би у свакој апликацији приликом тестирања било идеално користити све три врсте теста. Међутим, процентуална заступљеност три врсте теста би требало да буде веома различита. Највише треба да има Unit тестова, затим интеграционих који ће попунити недостатке претходних, и на крају само за кључне компоненте апликације треба користити End-To-End тестове. Наравно, сваки пројекат је специфичан тако да не треба слепо пратити препоручену пропорцију, али треба имати у виду предности и мане сваког теста, а са друге стране карактеристике пројекта па у складу са тим одлучити.

За потребе овог пројекта, све системске операције су тестиране аутоматски коришћењем UNIT тестова и Mocking технологије која подразумева симулирање очекиваног понашања неке екстерне зависности апликације, попут базе података, фајла итд. На основу уочених проблема у методама, исте су унапређене и дорађене. Написани тестови су подељени у следеће целине:

- Тестови за рад са клијентима
- Тестови за рад са сеансама

- Тестови за логовање психотерапеута
- Тестови за учитавање градова, облика терапије и ИД-јева клијента и сеансе

У наставку су приказани тестови за системске операције које се односе на класу Клијент и релациону табелу Клијент, тј. за системске операције које раде са објектом ове класе.

```
namespace UnitTests
{
    [TestClass]
    public class KljentUnitTests
    {
        private List<Klijent> LazniKlijenti()
        {
            List<Klijent> rezultat = new List<Klijent>
            {
                new Klijent
                {
                    KlijentID = 2,
                    Ime="Aleksandra",
                    Prezime="Tomasevic",
                    DatumRodjenja= new DateTime(1996,3,23) ,
                    BrojTelefona="063-468-382",
                    Mejl="aleks@gmail.com",
                    UkupanDug =0,
                    Grad = new Grad{ GradID = 1 },
                    Psihoterapeut = new Psihoterapeut{ TerapeutID = 2}
                },
                new Klijent
                {
                    KlijentID = 3,
                    Ime="Sanja",
                    Prezime="Mosic",
                    DatumRodjenja= new DateTime(1996,3,12),
                    BrojTelefona="064-6543-98",
                    Mejl="sanjci123@gmail.com",
                    UkupanDug =4000,
                    Grad = new Grad{ GradID = 1 },
                    Psihoterapeut = new Psihoterapeut{ TerapeutID = 2}
                },
                new Klijent
                {
                    KlijentID = 4,
                    Ime="Janko",
                    Prezime="Balsic",
                    DatumRodjenja= new DateTime(1996,4,2),
                    BrojTelefona="064-34567892",
                    Mejl="jankec@gmail.com",
                    UkupanDug =2000,
                    Grad = new Grad{ GradID = 2 },
                    Psihoterapeut = new Psihoterapeut{ TerapeutID = 2}
                },
                new Klijent
                {
                    KlijentID = 8,
                    Ime="Bojana",
                    Prezime="Radulovic",
                    DatumRodjenja= new DateTime(1996,8,13) ,
                    BrojTelefona="064-345678",
                    Mejl="boksi96@gmail.com",
                    UkupanDug =0,
                    Grad = new Grad{ GradID = 1 },
                }
            }
        }
    }
}
```

```

        Psihoterapeut = new Psihoterapeut{ TerapeutID = 2}
    },
    new Klijent
    {
        KlijentID = 9,
        Ime="Nemanja",
        Prezime="Pavlovic",
        DatumRodjenja= new DateTime(1991,10,9) ,
        BrojTelefona="063-345789",
        Mejl="nemis91@gmail.com",
        UkupanDug =2000,
        Grad = new Grad{ GradID = 2 },
        Psihoterapeut = new Psihoterapeut{ TerapeutID = 1}
    },
    new Klijent
    {
        KlijentID = 10,
        Ime="Dragana",
        Prezime="Nesic",
        DatumRodjenja= new DateTime(1990,2,24) ,
        BrojTelefona="0659902402",
        Mejl="dragana.nesic90@gmail.com",
        UkupanDug =0,
        Grad = new Grad{ GradID = 1 },
        Psihoterapeut = new Psihoterapeut{ TerapeutID = 2}
    },
    new Klijent
    {
        KlijentID = 13,
        Ime="Milomir",
        Prezime="Milutinovic",
        DatumRodjenja= new DateTime(1960,5,5) ,
        BrojTelefona="064-84-16-001",
        Mejl="milomir123@gmail.com",
        UkupanDug =2500,
        Grad = new Grad{ GradID = 2 },
        Psihoterapeut = new Psihoterapeut{ TerapeutID = 1}
    },
    new Klijent
    {
        KlijentID = 14,
        Ime="Andriana",
        Prezime="Adamovic",
        DatumRodjenja= new DateTime(1996,5,25),
        BrojTelefona="063-345-827",
        Mejl="adksi@gmail.com",
        UkupanDug =2000,
        Grad = new Grad{ GradID = 1 },
        Psihoterapeut = new Psihoterapeut{ TerapeutID = 3}
    },
    new Klijent
    {
        KlijentID = 15,
        Ime="Katarina",
        Prezime="Gojkovic",
        DatumRodjenja= new DateTime(1996,5,25) ,
        BrojTelefona="069-178-3998",
        Mejl="kaca123@gmail.com",
        UkupanDug =0,
        Grad = new Grad{ GradID = 2 },
        Psihoterapeut = new Psihoterapeut{ TerapeutID = 2}
    },
    new Klijent

```

```

        {
            KljentID = 16,
            Ime="Slavica",
            Prezime="Milutinovic",
            DatumRodjenja= new DateTime(1961,7,13) ,
            BrojTelefona="063 498 883",
            Mejl="slavica123@gmail.com",
            UkupanDug =5000,
            Grad = new Grad{ GradID = 1 },
            Psihoterapeut = new Psihoterapeut{ TerapeutID = 2}
        }

};
return rezultat;
}

[TestMethod]
public void UcitajListuKlijenataSO()
{
    OpstaSistemskaOperacija so = new UcitajListuKlijenataSO();

    var expected = LazniKlijenti();
    so.Izvrsi(new Kljent());
    List<Kljent> actual = ((UcitajListuKlijenataSO)so).ListaKl;

    Assert.IsTrue(actual != null);
    CollectionAssert.AreEquivalent(expected, actual);
}

[TestMethod]
public void IzmeniKlijentaSO()
{
    OpstaSistemskaOperacija so = new IzmeniKlijentaSO();

    var expected = new Kljent
    {
        KljentID = 16,
        Ime = "Slavica",
        Prezime = "Milutinovic",
        DatumRodjenja = new DateTime(1961,7,13),
        BrojTelefona = "063 498 883",
        Mejl = "slavica.milutin@gmail.com",
        UkupanDug = 5000,
        Grad = new Grad { GradID = 1 },
        Psihoterapeut = new Psihoterapeut { TerapeutID = 2 }
    };
    so.Izvrsi(expected);
    bool izmenjen = ((IzmeniKlijentaSO)so).Izmenjen;
    Assert.IsTrue(izmenjen);
}

[TestMethod]
public void UnesiKlijentaSO()
{
    OpstaSistemskaOperacija so = new UnesiKlijentaSO();

    var expected = new Kljent
    {
        KljentID = 17,
        Ime = "Mina",
        Prezime = "Milutinovic",

```

```

        DatumRodjenja = new DateTime(1961, 7, 13),
        BrojTelefona = "063 498 883",
        Mejl = "mina.milutin@gmail.com",
        UkupanDug = 0,
        Grad = new Grad { GradID = 1 },
        Psihoterapeut = new Psihoterapeut { TerapeutID = 3 }
    };
    so.Izvrsi(expected);
    Klient actual = ((UnesiKlijentaSO)so).Klijent;
    Assert.IsTrue(actual != null);
    Assert.AreEqual(expected, actual);
}

[TestMethod]
public void ObrisiKlijentaSO()
{
    OpstaSistemskaOperacija so = new ObrisiKlijentaSO();

    var expected = new Klient
    {
        KlientID = 17,
        Ime = "Mina",
        Prezime = "Milutinovic",
        DatumRodjenja = new DateTime(1961, 7, 13),
        BrojTelefona = "063 498 883",
        Mejl = "mina.milutin@gmail.com",
        UkupanDug = 0,
        Grad = new Grad { GradID = 1 },
        Psihoterapeut = new Psihoterapeut { TerapeutID = 3 }
    };
    so.Izvrsi(expected);
    bool obrisan = ((ObrisiKlijentaSO)so).Obrisan;
    Assert.IsTrue(obrisan);
}

[TestMethod]
public void VратиKlijentaSO()
{
    OpstaSistemskaOperacija so = new VратиKlijentaSO();

    var expected = new Klient
    {
        KlientID = 16,
        Ime = "Slavica",
        Prezime = "Milutinovic",
        DatumRodjenja = new DateTime(1961, 7, 13),
        BrojTelefona = "063 498 883",
        Mejl = "slavica.milutin@gmail.com",
        UkupanDug = 5000,
        Grad = new Grad { GradID = 1 },
        Psihoterapeut = new Psihoterapeut { TerapeutID = 2 }
    };
    so.Izvrsi(expected);
    Klient actual = ((VратиKlijentaSO)so).Klijent;
    Assert.IsTrue(actual != null);
    Assert.AreEqual(expected, actual);
}

[TestMethod]
public void PretragaKlijenataSO()
{
    OpstaSistemskaOperacija so = new PretragaKlijenataSO();
    List<Klijent> k = new List<Klijent>

```

```

    {
        new Klient
        {
            KlientID = 16,
            Ime = "Slavica",
            Prezime = "Milutinovic",
            DatumRodjenja = new DateTime(1961, 7, 13),
            BrojTelefona = "063 498 883",
            Mejl = "slavica.milutin@gmail.com",
            UkupanDug = 5000,
            Grad = new Grad { GradID = 1 },
            Psihoterapeut = new Psihoterapeut { TerapeutID = 2 }
        }
    };
    var expected = k;
    so.Izvrsi("slavica.mil");
    List<Klijent> actual = ((PretragaKlijenataSO)so).rez;
    Assert.IsTrue(actual != null);
    CollectionAssert.AreEqual(expected, actual);
}
}
}

```

Аналогно приказаним тестовима, написани су тестови за све системске операције. Сви тестови су прошли, што доводи до закључка да све функционалности које у позадини позивају системске операције раде без грешке.

12.Закључак

У овом завршном раду приказан је комплетан поступак креирања софтверског система за одређено психолошко саветовалиште. Као резултат бројних разговора са запосленим стручним лицима психолошког саветовалишта у Београду, захтеви су прикупљени и дефинисани. Систем је у потпуности креиран у односу на тражена својства и велики део реалних захтева корисника је имплементиран. Приликом имплементирања, најважније је било да апликација добије захтеване функционалности, али поред тога доста пажње посвећено је дизајну и естетици корисничког интерфејса.

Иако је креиран систем који би могао већ сада да се користи у циљаном психолошком саветовалишту, сигурно има места за разна побољшања, како би исти у потпуности задовољио корисника. У складу са реченим, приликом тестирања софтвера од стране корисника, планира се спровођење End-to-End тестова уз помоћ Selenium алата. Тако ће се пратити и анализирати коришћење софтвера од стране корисника у реалном систему и понашање софтвера у таквом систему. Видеће се да ли апликација пружа оно што се од ње очекује и да ли су потребна додатна прилагођавања.

У овом раду приказани су и основни концепти програмског језика C#. Описани су концепти који су коришћени у развоју овог софтверског система: рад у мрежи, рад са базом података и концепт нити. Све наведене и описане технологије су коришћене и омогућиле су развој овог софтверског система.

Резултат практичног дела рада је десктоп апликација за вођење евиденције о клијентима и сеансама психолошког саветовалишта, развијена на .NET платформи. Ларманова метода развоја софтвера је омогућила да развој система буде испраћен одговарајућом документацијом кроз све фазе развоја - прикупљање захтева, анализу, пројектовање, имплементацију и тестирање. Коришћена је упрошћена Ларманова метода, по узору на семинарски рад из предмета Пројектовање Софтвера.

Почетак пројекта ,тачније фазе прикупљања захтева и анализе, представљале су веома изазовни део читавог процеса креирања софтвера. Требало је унапред испланирати и дефинисати како ће апликација изгледати и шта ће радити, а све то са одређеном дозом апстракције. Било је битно осмислити модеран софтвер, интуитиван за коришћење, и што је најважније функционалан и поуздан. Такође, један од тежих задатака представљало је решавање проблема чувања C# класа у релациону базу података и превазилажење проблема који су настали са имплементацијом концепта наслеђивања.

Након израде пројекта тестиране су одређене логичке јединице апликације. Системске операције су тестиране аутоматским UNIT тестовима уз симулацију очекиваних резултата и њиховог поређења са резултатима из базе података. Остале функционисности тестиране су маунелно у току и након израде апликације. Учење о тестовима, њиховом писању, покретању и позитивном утицају који имају на сам развој апликације , учинило је читав процес занимљивим и много лакшим.

13. Литература

1. Introduction to the C# language and the .NET Framework, Welcome to the Visual Studio IDE, An introduction to NuGet , SQL Server Management Studio . Retrieved August 15, 2020, from <https://docs.microsoft.com/en-us/>
2. C# | Exception. Retrieved August 05, 2020, from <https://www.geeksforgeeks.org/c-sharp-exception/?ref=rp>
3. Vučković M. , Turajlić N. , Petrović M. (2009/2010) . Objektno-orjentisano programiranje . Retrieved August 20, 2020, from <http://is.fon.bg.ac.rs/wp-content/uploads/2020/04/Objektno-orijentisani-pristup.pdf>
4. Components of .Net Framework . Retrieved August 25, 2020, from <http://www.developerin.net/a/39-Intro-to-.Net-FrameWork/23-Components-of-.Net-Framework>
5. Softverska arhitektura. Retrieved August 06, 2020, from <https://www.linkelearning.com/site/kursevi/lekcija/4322>
6. Mašulović D. , Vasiljević N. , Vugdelija M. (2020). Uvod u programiranje u programskom jeziku C# . Retrieved August 10, 2020, from <https://petljamediastorage.blob.core.windows.net/root/Media/Default/Kursevi/spec-it/csharpprirucnik.pdf>
7. Hamedani M. Unit Testing for C# Developers . Retrieved August 31, 2020 from <https://codewithmosh.com/p/unit-testing-for-csharp-developers/>
8. Vlajić S. , Savić D. , Stanojević V. , Antović I. , Milić M. (2015). Projektovanje softvera (knjiga). Beograd, Srbija: FON.
9. Radovanović M. (2015). Vrednosni i referentni tipovi podataka u C# programskom jeziku. Retrieved August 18, 2020 , from <https://www.manuelradovanovic.com/2015/12/vrednosni-i-referentni-tipovi-podataka.html>
10. Vlajić S. (2015). Projektovanje softvera (skripta). Beograd, Srbija: FON.
11. Vuleta S. (2018) .NET Standard. Retrieved August 29, 2020 ,from <https://blog.itkonekt.com/2018/04/03/sta-je-to-net-standard/>

