

**УНИВЕРЗИТЕТ У БЕОГРАДУ
ФАКУЛТЕТ ОРГАНИЗАЦИОНИХ НАУКА**

ЗАВРШНИ РАД

**Тема: Развој софтверског система за вођење
евиденције о складиштењу књига применом
.NET технологија**

Ментор:
проф. др Саша Лазаревић

Студент:
Милош Ристић 2017/0375

Београд, 2022. године

Садржај

1. Увод	7
2. Опис коришћених технологија	8
2.1 .NET платформа	9
2.1.1 Windows Forms	11
2.1.2 C#	12
2.1.3 LINQ	17
2.2 Релациона база података	18
2.3 Патерни	19
2.3.1 MVC патерн	20
2.3.2 Singleton патерн	21
2.3.3 Template method патерн	22
3. Студијски пример	22
3.1 Ларманова метода	22
3.2 Прикупљање корисничких захтева	23
3.2.1 Вербални опис	23
3.2.2 Случајеви коришћења	24
СК1: Случај коришћења – Креирање требовања складишту	27
СК2: Случај коришћења – Креирање пријемнице добављачу	28
СК3: Случај коришћења – Креирање отпремнице за складиште	29
СК4: Случај коришћења – Креирање рекламације на основу комисијског записника	30
СК5: Случај коришћења – Креирање доставнице на основу отпремнице добављача	31
СК6: Случај коришћења – Пријем књига у књижару	32
3.3 Анализа	33
3.3.1 Системски дијаграми секвенци	33
ДС1: Дијаграм секвенци случаја коришћења – Креирање требовања складишту	33
ДС2: Дијаграм секвенци случаја коришћења – Креирање пријемнице добављачу	35
ДС3: Дијаграм секвенци случаја коришћења – Креирање отпремнице за складиште	37
ДС4: Дијаграм секвенци случаја коришћења – Креирање рекламације на основу комисијског записника	39
ДС5: Дијаграм секвенци случаја коришћења – Креирање доставнице на основу отпремнице добављача	40
ДС6: Дијаграм секвенци случаја коришћења – Пријем књига у књижару	42
3.3.2 Понашање софтверског система – Дефинисање уговора о системским	45
операцијама	45
3.3.3 Структура софтверског система – Концептуални (доменски) модел	50
3.4 Пројектовање	51

3.4.1 Пројектовање складишта података – Релациони модел	52
3.4.2 Пројектовање корисничког интерфејса	57
СК1: Случај коришћења – Креирање требовања складишту	58
СК2: Случај коришћења – Креирање пријемнице добављачу	60
СК3: Случај коришћења – Креирање отпремнице за складиште	61
СК4: Случај коришћења – Креирање рекламације на основу комисијског записника	63
СК5: Случај коришћења – Креирање доставнице на основу отпремнице добављача	65
СК6: Случај коришћења – Пријем књига у књижару	67
3.4.3 Пројектовање апликационе логике	69
3.5 Имплементација	88
3.5.1 Структура софтверског система	88
3.5.2 Имплементација складишта података	89
3.5.3 Имплементација апликативне логике	91
3.5.4 Брокер базе података	108
3.5.5 Имплементација корисничког интерфејса	110
СК1: Случај коришћења – Креирање требовања складишту	110
СК2: Случај коришћења – Креирање пријемнице добављачу	112
СК3: Случај коришћења – Креирање отпремнице за складиште	114
СК4: Случај коришћења – Креирање рекламације на основу комисијског записника	116
СК5: Случај коришћења – Креирање доставнице на основу отпремнице добављача	118
СК6: Случај коришћења – Пријем књига у књижару	120
3.6 Тестирање	122
4. Закључак	122
5. Референта литература	123

Списак слика

Слика 1 Преглед .NET технологија. Извор: https://devblogs.microsoft.com/dotnet/wp-content/uploads/sites/10/2019/05/dotnet5_platform.png	10
Слика 2 Преглед функционалности по верзијама C#-а. Извор: https://dev.to/jai00271/c-features-from-5-0-6-0-and-7-0-2lbn	13
Слика 3 CLR у .NET-у, Извор: https://www.geeksforgeeks.org/common-language-runtime-clr-in-c-sharp	14
Слика 4 Функционисање сокета, Извор: https://www.geeksforgeeks.org/socket-programming-cc/	17
Слика 5 Дијаграм случајева коришћења	25
Слика 6 Креирање требовања складишту	34
Слика 7 Неуспешно креирање требовања	35
Слика 8 Креирање пријемнице добављачу	36
Слика 9 Неуспешно креирање пријемнице добављачу	36
Слика 10 Креирање отпремнице за складиште	37

Слика 11 Креирање отпремнице за складиште.....	38
Слика 12 Креирање рекламације.....	39
Слика 13 Неуспешно креирање рекламације.....	40
Слика 14 Креирање доставнице на основу отпремнице добављача.....	41
Слика 15 Неуспешно креирање доставнице на основу отпремнице добављача.....	42
Слика 16 Пријем књига у књижу.....	43
Слика 17 Пријем књига у књижу.....	43
Слика 18 Концептуални модел.....	50
Слика 19 Тронивојска архитектура.....	51
Слика 20 Форма за унос требовања.....	58
Слика 21 Бирање књиге и потребне количине.....	59
Слика 22 Успешан унос требовања.....	59
Слика 23 Неуспешан унос требовања.....	59
Слика 24 Форма са списком свих постојећих требовања.....	60
Слика 25 Опција Креирање Пријемнице за добављача.....	61
Слика 26 Успешно креирање пријемнице.....	61
Слика 27 Неуспешно креирање пријемнице.....	61
Слика 28 Списак свих пријемница послатих добављачу.....	62
Слика 29 Креирање отпремнице за складиште.....	62
Слика 30 Успешно креирање отпремнице.....	63
Слика 31 Успешно мењање отпремнице.....	63
Слика 32 Списак постојећих требовања према датом складишту.....	64
Слика 33 Опција Креирај Комисиони записник о пријему књига.....	64
Слика 34 Успешно креирање записника.....	64
Слика 35 Неуспешно креирање записника.....	65
Слика 36 Списак свих требовања за дато складиште.....	65
Слика 37 Опција Креирај доставницу за књижу.....	66
Слика 38 Успешно креирање доставнице.....	66
Слика 39 Неуспешно креирање доставнице.....	66
Слика 40 Списак доставница за које треба извршити пријем.....	67
Слика 41 Опција Пријем књига из складишта.....	68
Слика 42 Успешан пријем књига.....	68
Слика 43 Неуспешан пријем књига.....	68
Слика 44 ДС:Уговор - ВратиСписакКњига.....	70
Слика 45 ДС:Уговор - ВратиМагацинЗаКњижу.....	71
Слика 46 ДС:Уговор - ВратиСписакПријемница.....	72
Слика 47 ДС:Уговор - ВратиСтавкеДокумента.....	73
Слика 48 ДС:Уговор - ВратиПодаткеОДокументу.....	74
Слика 49 ДС:Уговор - ВратиВјуДата.....	74
Слика 50 ДС:Уговор - СинхроТребовање.....	75
Слика 51 ДС:Уговор - ВратиКњижареЗаМагацин.....	76
Слика 52 ДС:Уговор - ВратиСписакМагацина.....	77
Слика 53 ДС:Уговор - ВратиПрометЗа.....	77
Слика 54 ДС:Уговор - ВратиСтањеЗа.....	78
Слика 55 ДС:Уговор - ВратиОбјектИнфо.....	79
Слика 56 ДС:Уговор - ВратиОтпремницуЗаРекламацију.....	80
Слика 57 ДС:Уговор - ОбрадаДокумента.....	81
Слика 58 ДС:Уговор - Промет.....	81
Слика 59 ДС:Уговор - АжурИзнос.....	82
Слика 60 ДС:Уговор - Повежи.....	83
Слика 61 ДС:Уговор - МакеРекламација.....	84
Слика 62 ДС:Уговор - ЈеднаСтрана.....	84
Слика 63 ДС:Уговор - УпдатеСтатус.....	85
Слика 64 ДС:Уговор - ВратиПодаткеОКњизи.....	86

Слика 65 ДС:Уговор - АжурирајПодаткеОКњизи	86
Слика 66 ДС:Уговор – ВратиВјуБуттонс	87
Слика 67 Бројачи	89
Слика 68 Документ	89
Слика 69 Књиге	90
Слика 70 Објекти	90
Слика 71 Промет	90
Слика 72 Ставка	90
Слика 73 Веза	90
Слика 74 ВрстаДокумента	91
Слика 75 ВрстаОбјекта	91
Слика 76 ВрстаРМ	91
Слика 77 Запослени	91
Слика 78 Коначна архитектура	109
Слика 79 Форма за унос требовања	111
Слика 80 Бирање књиге и потребне количине	111
Слика 81 Успешан унос требовања	112
Слика 82 Неуспешан унос требовања	112
Слика 83 Форма са списком свих постојећих требовања	113
Слика 84 Опција Креирање Пријемнице за добављача	113
Слика 85 Успешно креирање пријемнице	113
Слика 86 Неуспешно креирање пријемнице	113
Слика 87 Списак свих пријемница послатих добављачу	114
Слика 88 Креирање отпремнице за складиште	115
Слика 89 Успешно креирање отпремнице	115
Слика 90 Успешно мењање отпремнице	115
Слика 91 Списак постојећих требовања према датом складишту	116
Слика 92 Опција Креирај Комисиони записник о пријему књига	116
Слика 93 Успешно креирање записника	117
Слика 94 Неуспешно креирање записника	117
Слика 95 Списак свих требовања за дато складиште	118
Слика 96 Опција Креирај доставницу за књижару	119
Слика 97 Успешно креирање доставнице	119
Слика 98 Неуспешно креирање доставнице	119
Слика 99 Списак доставница за које треба извршити пријем	120
Слика 100 Опција Пријем књига из складишта	120
Слика 101 Успешан пријем књига	121
Слика 102 Неуспешан пријем књига	121

Списак табела

Табела 1 Приказ нативних и мултиплатформских оквира	8
Табела 2 Документ	52
Табела 3 Књиге	53
Табела 4 Објекти	53
Табела 5 Ставке	53
Табела 6 Промет	54
Табела 7 Страна	54
Табела 8 ВрстаДоц	54
Табела 9 ВрстаОбјекта	54
Табела 10 ВрстаРМ	55
Табела 11 Запослени	55
Табела 12 Бројачи	55
Табела 13 Веза	56

1. Увод

Пословање у данашње време скоро да је незамисливо без адекватне софтверске подршке. Софтверска подршка пословању почиње од економских пословних процеса, као што су издавање рачуна, пријем рачуна, издавање робе, пријем робе и слично. У зависности од типа фирме и делатности, пословни софтвер прилагођава се специфичним пословним процесима.

У том смислу, пословни софтвер обухвата различите категорије решења: системе за компанијско управљање и планирање (ЕРП), системе за управљање документима (ДМС), системе за управљање односима с купцима (ЦРМ), системе подршке одлучивању, пословно извештавање (БИ/ДСС). С друге стране, свака од ових категорија даље се специјализује за одређену делатност, односно за специфичну индустријску грану. Та специјалност одређује и примарну намену пословног софтвера из угла корисника.

Сваки пословни софтвер развија се да би решио одређену групу пословних проблема. У veleпродајним компанијама нису исти проблеми као у производним компанијама. Различити су пословни процеси, кључни параметри успешности, начин рада, приоритет активности (брзина или детаљи) и слично.

Дакле, када произвођач софтвера каже да је решење намењено некој одређеној групи компанија (делатност), то значи да је успео да одређене пословне проблеме реши на начин прихватљив управо за ту специфичну групу компанија. Ово је врло важан параметар јер, примера ради, софтвер оптимизован за рачуноводствене агенције обично није адекватно решење за производну компанију. А чињеница је да ниједан софтвер не може бити све свима, тј. нема универзалног решења.

2. Опис коришћених технологија

У овом раду иницијална идеја јесте да се информациони систем за вођење евиденције о пословању књижара и складиштењу књига, који ће детаљније бити описан у следећим секцијама, имплементира као десктоп апликација, конкретно Windows Forms апликација.

Десктоп апликације су оне које се креирају за рад на стоним и преносним рачунарима. Постоје много језика и оквира који се користе за креирање ових апликација. Они се генерално могу категоризовати по платформи коју подржавају или као мултиплатформске заједно са програмским језицима и библиотекама које користе за прилагођавање[1]. На следећој слици можемо видети преглед ових технологија.

Технологије и оквири за платформске (нативне) апликације	Технологије и оквири за развој мултиплатформских апликација
Windows: .NET(WPF и Winforms), C# or VB.NET	C++, C++ са QT Framework-ом
MacOS: ObjectiveC	Java
Linux: C++	Electron
	Uniti framework
	Python (wxPython, pyQT, pyGKT)

Табела 1 Приказ нативних и мултиплатформских оквира

Конкретно у овом раду, фокус је стављен на израду нативне апликације, јер је сама тема апликације погодна за овај тип апликације.

Неке **предности** нативних апликација су следеће:

- бржа и боља респонзивност система,
- кориснички интерфејс који се може лако ажурирати са променама оперативног система,
- лако одржавање,
- коришћење оквира који се стално ажурирају заједно са оперативним системом.

Са друге стране, постоје и неке **негативне** стране овог начина развоја апликација као што су:

- више различитих кодова базираних на различитим платформама,
- додатни трошкови за додатне програмере и управљање кодном базом за сваку платформу,
- време утрошено на вишеструким верзијама за различите платформе за свако ажурирање апликација,
- потпуно другачији систем регресионих тестова за сваку платформу.

Даље у овом поглављу биће описан технологије у којима је развијан студијски пример.

2.1 .NET платформа

Платформа отвореног кода, .NET, створена је од стране *Microsoft*-а и користи се за развој различитих типова апликација. Она подржава више типова програмских језика и библиотека за израду мобилних, десктоп, апликација које се користе у интернету интелигентних уређаја и многих других. Преглед је доступан на следећој слици:

.NET – A unified platform



Слика 1 Преглед .NET технологија. Извор:
https://devblogs.microsoft.com/dotnet/wp-content/uploads/sites/10/2019/05/dotnet5_platform.png

Језици које *Microsoft* директно подржава су:

- **C#:** савремени објектно оријентисани програмски језик који припада породици Ц језика.
- **F#:** Функционално оријентисан програмски језик који припада породици језика МЛ. Такође подржава парадигму објектно - оријентисаног програмирања.
- **Visual basic:** *Microsoft*-ов историјски програмски језик. Развио се у потпуно објектно оријентисан програмски језик као део .NET[2].

.NET подржава *Common Language Infrastructure (CLI)* тј. општу језичку инфраструктуру, па се изворни код компајлира у *Common Intermediate Language (CIL)* тј. општи интермедијални језик, без обзира на програмски језик који се користи. Ово гарантује високу интероперабилност између језика платформе[3].

Поред горе наведених језика које *Microsoft* директно подржава, многи други програмски језици могу се компајлирати у *.NET CIL*. На пример, *ClojureCLR*, *Eiffel*, *IronPython*, *PowerBuilder* и многи други[4].

Неки од основних оквира које подржава платформа у развоју апликација јесу:

- *Blazor*: Оквир за израду клијентских веб апликација коришћењем C#. Такође може да генерише клијентске веб апликације у веб асемблерском коду.
- *Windows Presentation Foundation (WPF)*: графички кориснички интерфејс за израду мултиплатформских десктоп апликација
- *ML.NET*: Оквир машинског учења који поједностављује интеграцију модела машинског учења у .NET апликацију.
- *ASP.NET*: Оквир који омогућава израду веб апликација и веб АПИ -ја.
- *Xamarin*: Оквир за изградњу мобилних, ТВ и десктоп апликација за више платформи.

За све остале потребе које нису уграђене у оквир, могу се пронаћи велики број специфичних библиотека у јавном спремишту Нугет. Нугет је менаџер пакета на .NET платформи. Помоћу њега се могу имплементирати већ готове .NET библиотеке за готово било коју потребу или функционалност.

2.1.1 Windows Forms

Windows Forms (WinForms) је бесплатна и графичка библиотека отвореног кода која је део Мајкрософт .NET, .NET Framework или Mono Framework-а [5], која пружа платформу за писање богатих клијентских апликација за десктоп, лаптоп, и таблет рачунаре [6]. Иако се посматра као замена за ранију и сложенију библиотеку класа *Microsoft Foundation Class* засновану на C++ програмском језику, оне нису међусобно упоредиве и ради само као платформа за ниво корисничког интерфејса у вишеслојној архитектури. [7]

На догађају Мајкрософт Концерт 4. децембра 2018. Мајкрософт је најавио објављивање Windows Forms-а као пројекат отвореног кода под МИТ лиценцом. Овим издањем, Windows Forms је постао доступан за пројекте који користе .NET Core оквир. Међутим, оквир је и даље доступан само на Windows платформи, а непотпуна имплементација Mono Framework-а остаје једина имплементација Windows Forms-а која је доступна на више платформи. [8]

Архитектура

Ова библиотека се у целости заснива на догађајима. За разлику од скриптних програма, она већину свог времена проводи само чекајући да корисник учини нешто, тј. да направи неки догађај. Тај догађај може да буде попуњавање текстуалног поља или клик на дугме. Код за апликацију може да буде написан коришћењем неког .NET програмског језика, као што су C# или Visual Basic. Windows Forms омогућава приступ уобичајеним контролама корисничког интерфејса Windows-а тако што код пребацује у постојећи Windows API код. Уз помоћ Windows Forms-а, .NET Framework пружа свеобухватнију апстракцију Win32 API-ја него што је то био случај код Visual Basic-а.

Функционалности

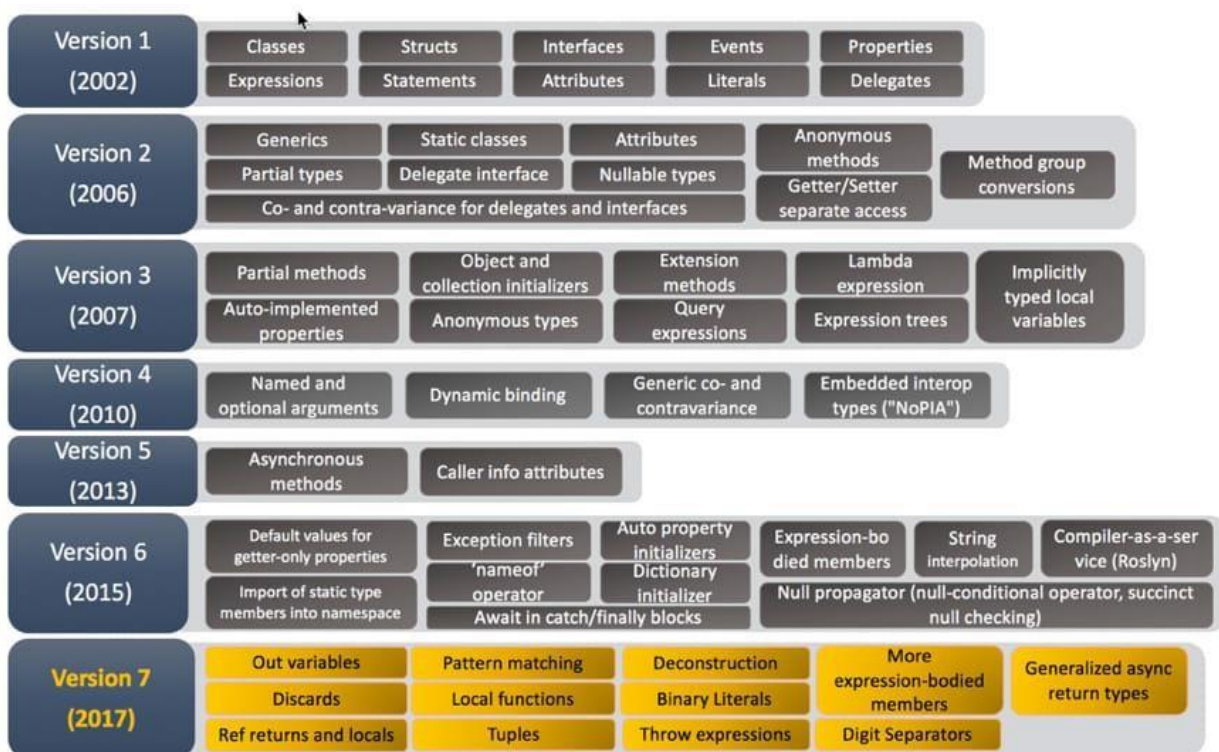
Сви визуелни елементи Windows Forms-а наслеђују класу *Control*. Она пружа основне функционалности једног графичког корисничког интерфејса као што су локација, величина, боја, фонт, текст, као и најчешће коришћене догађаје као што су клик и превлачење елемената. Ова класа такође омогућава изведеним класама да подесе свој положај у односу на родитеља. Осим овога, додатно пружа функционалности које помажу корисницима са ослабљеним видом да неометано користе Windows Forms.

Осим што је омогућио приступ изворним Windows контролама као што су дугме, текстуално поље, приказ листи и радио дугме, Windows Forms је додао и сопствене контроле, као што су уређивање распореда елемената, валидацију уноса и повезивање података. Ове контроле се приказују помоћу *Graphic Device Interface ++* библиотеке.

2.1.2 C#

C# је типски сигуран, модеран, објектно оријентисан програмски језик. Уз C#, програмери могу да креирају многе врсте сигурних и робусних апликација које раде на .NET платформи. C# има своје корене у породици Ц језика[9]. Неутралан је што се тиче платформе.

Microsoft је први пут представио програмски језик C# јавности 2000. године и од тада је полако проширавао могућности које Ц# нуди. Језик је еволуирао кроз шест верзија, додајући све од генеричких и ламбда израза до асинхроних метода и интерполације стрингова. На следећој слици можемо видети како је *Microsoft* убацивао нове функционалности у овај језик по годинама и продукцијским верзијама закључно са верзијом 7 из 2017. године:

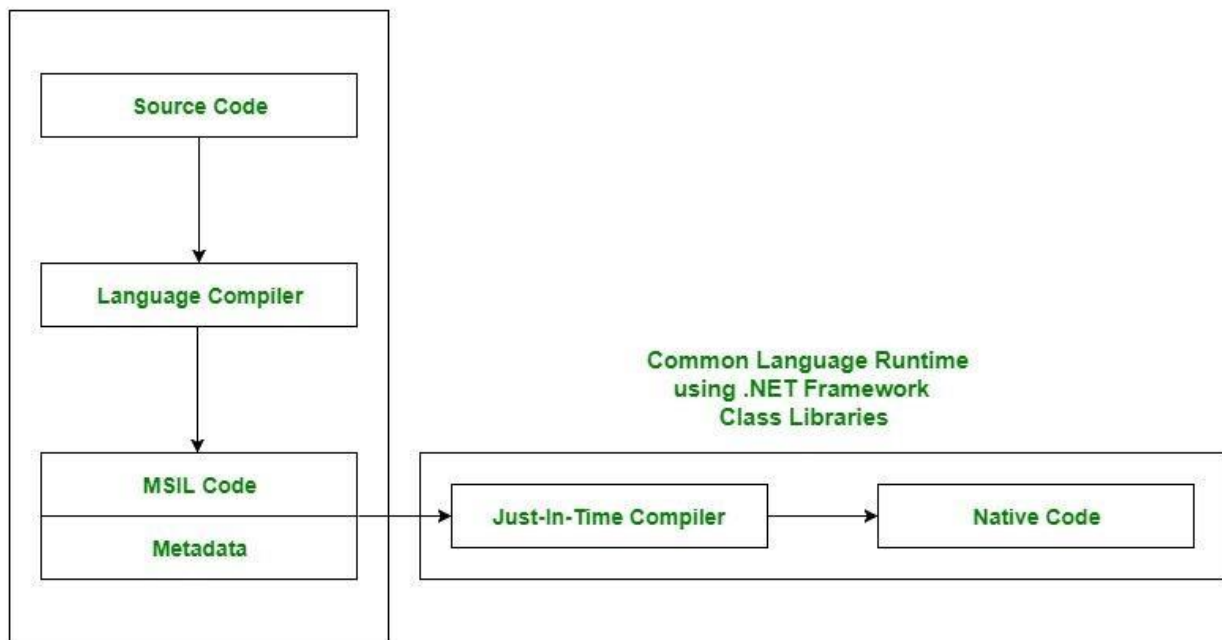


Слика 2 Преглед функционалности по верзијама C#-а. Извор: <https://dev.to/jai00271/c-features-from-5-0-6-0-and-7-0-2ldn>

Након верзије 7, изашле су и верзије: 7.1, 7.2, 7.3, 8.0 и 9.0. Неке од занимљивих функционалности језика које су имплементирани у овим издањима јесу[10]:

- Асинхроне *main* методе
- Технике за писање безбедоносно ефикасног кода
- *Private protected* модификатор приступа
- Условни референтни изрази
- *Readonly* атрибути
- *Using* блокови
- *Nullable* референтни типови
- Асинхрони токови података
- Статичке локалне функције
- Побољшање интерполарних вербатим стрингова
- Коваријантни повратни типови
- Нове функционалности за парцијалне методе

C# апликације се извршавају на .NET виртуелном систему извршења који се назива *CLR (common language runtime)* заједно са скупом библиотека. *CLR* је имплементација *Microsoft*-ове заједничке језичке инфраструктуре - *CLI (common language infrastructure)*, међународног стандарда. Изворни код који је написан у C# језику се даље компајлира у средњи језик *IL (intermediate language)* који треба да буде у складу са *CLI* спецификацијом. Ресурси (битмапе и стрингови) заједно са *IL*-ом су складиштени у меморији у извршни фајл који се зове асембли (.exe или .dll екстензије). Након извршавања програма, асембли се учитава у *CLR* који извршава Just-In-Time компајлирање како би превео *IL* у машински код. JIT компајлери су добро прилагођени за дуготрајне сценарије у облаку и клијенте. Они су у стању да генеришу код који таргетира на одређену конфигурацију машине, укључујући специфична упутства за процесор. JIT такође може регенерисати методе током извршавања, што је техника која се користи за убрзавање JIT-а, а истовремено има могућност генерисања високо оптимизоване верзије кода када то постане доста коришћена метода[2].



Слика 3 CLR у .NET-у, Извор: <https://www.geeksforgeeks.org/common-language-runtime-clr-in-c-sharp>

Humu

Нити (threads) решавају проблем комуникације и размене података између различитих процеса јер на тај начин што деле исти меморијски процес. У конкретној имплементацији то би значило да омогућавају да се више процеса одвијају паралелно. Тако можемо закључити да је C# језик који подржава вишенивно

програмирање. Извршавање програма аутоматски креира и извршава једну нит која се назива главна програмска нит.

Нит је независна путања извршавања кода која се истовремено може покренути са другим извршавањима тј. другим нитима. С# клијентски програми као што су конзолне апликације, WPF, Windows форме, покрећу се у једној нити коју аутоматски креира CLR и оперативни систем. Њу називамо главном нити (“*main thread*”). Она је у позадини направљена од више стране више нити[11].

Од главне нити касније настају друге нити. Када се покрене, атрибут *IsAlive*, који говори о томе да ли је нит активна (жива) добија позитивну вредност и остаје тавка док се не заврши извршавање нити. Нит се завршава прослеђени делегат у *Thread* конструктору заврши своје извршавање. Када се једном нит заврши, не може поново да се покрене[11].

Вишенитни приступ се контролише интерно од стране планера нити (*Thread scheduler*), а ту функционалност обично CLR делегира процесору. Планер нити осигурава да се свим активним нитима додели одговарајуће време извршавања и да нити које чекају или су блокиране (на пример, на ексклузивном закључавању или на корисничком уносу) не троше процесорско време.

На рачунару са више процесора, мултитхрединг је имплементиран мешавином временског пресека и истините истовремености, где различите нити извршавају код истовремено на различитим ЦПУ-има. Готово је сигурно да ће се неки временски прекид ипак догодити, будући да „оперативни систем мора служити“ властитом низу и онима других апликација.

За нит се каже да је искључена када се њено извршавање прекине због спољног фактора, као што је скраћивање времена. У већини ситуација, а нит нема контролу над тиме када и где је искључена[11].

Кад год се покрене нит, потроши се неколико стотина микросекунди на организацију неких компоненти као што су чист приватни локални променљиви стек. Свака нит троши око 1 МВ меморије. Ту у причу долази концепт мреже нити, тј. *Thread pool* који смањује ове трошкове дељењем и рециклирањем нити, при чему омогућава примену вишенитности са минималним трошковима на нивоу перформанси такође. Ово је корисно када се користе вишејезгрени процесори за паралелно извршавање рачунарски интензивног кода применом концепта „подели па освоји“. Мреже нити такође контролишу укупан број нити радника које ће се истовремено покренути. Превише активних нити могу да спутавају оперативни систем административним оптерећењем и чине кеш меморије процесора

неефикасним. Једном када је граница достигнути, нове нити се стављају у ред и почињу тек када други заврши. Ово омогућава постојање арбитрарно конкурентних апликације, као што је веб сервер.

У нашем студијском примеру нису коришћене нити и то из следећих разлога:

1. Није коришћен механизам асинхроне комуникације која се искључиво обавља преко нити
2. Скуп података које апликација користи се протежу на десктоп платформи тако да није потребна серијализација података приликом комуникације апликације са извором података
3. Како је у питању десктоп апликација није неопходно водити рачуна о окружењу у коме се апликација компајлира у run time-у те нема потребе за конверзијом података између извора података и апликације
4. Извор података који апликација користи је садржан у оперативном систему и подтипа је System.Data који представља Open Source Standard те самим тим нема потребе за конверзијом података између извора података и апликације. Апликација користи System.Data.DataTable као примарни облик извора података на које се каче корисничке контроле (DataGridView, ComboBox i sl.)

Сокети

Сокет, у ширем смислу, је механизам који омогућава комуникацију између програма који се извршавају на различитим рачунарима у мрежи. Подржава имплементацију клијент/серверских апликација. За повезивање сокети користе ТСР/ИР протокол, а такође и за контролу и преносу података између два или више програма[12].

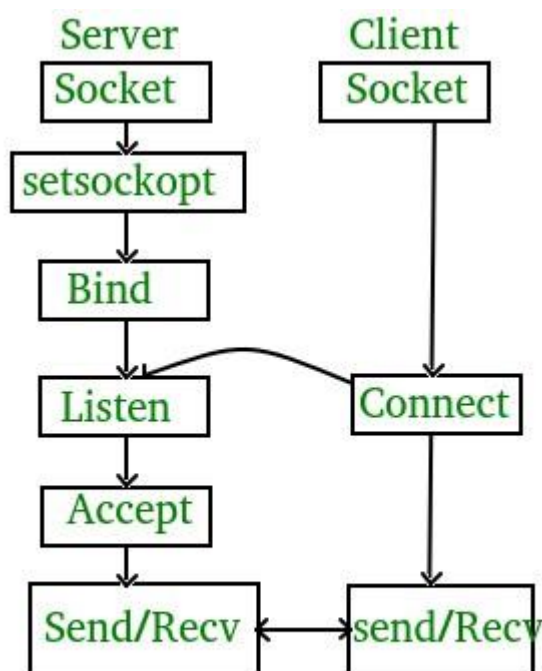
Сокети су посебни дескриптори датотека који се користе за референцирање мрежних веза. Они дефинишу:

- Одређени комуникацијски домен, као што је мрежна веза или UNIX међупроцесни комуникациоону (IPC) цев
- Одређени протокол, као што је ТСР или ИР
- Одређену врсту комуникације попут тока или датаграма

Након што је сокет креиран, мора бити везана за одређену мрежну адресу и порт на систему или за удаљену мрежну адресу и порт. Након што је сокет везана, може се користити за слање и примање података из мреже[13].

По природи, сокет је улазно-излазни ток и као такав се понаша на сличан начин као објекат System.in, тј. може да се користи да прихвата податке са

стандардног улаза, док се код сокета подаци прихватају са спољашњег улаза (са мреже) од другог сокета. Такође може да се понаша и као објекат System.out, помоћу кога се подаци шаљу ка стандардном излазу (екрану), док се код сокета подаци шаљу ка спољашњем излазу (ка мрежи) до другог сокета. Сокети се повезују са улазно-излазним токовима на сличан начин као што је то случај са System.in и System.out објектима, када се жели извршити обрада података коју сокети размењују[12].



Слика 4 Функционисање сокета, Извор: <https://www.geeksforgeeks.org/socket-programming-cc/>

Како се сокети користе у случајевим када је потребно обезбедити комуникацију у сигурној зони – заштићена IP адреса и порт и комуницирају искључиво у серијализованим подацима, а с обзиром да је у нашем студијском примеру имплементирана вишеслојна десктоп апликација нема потребе за конверзијом података па сам тим ни серијализација није потребна.

2.1.3 LINQ

Linq је акроним за Language Integrated Query (језички интегрисани упит) који је део

.NET технологије. Он пружа могућности постављања упита на нивоу језика и као и АПИ функције вишег реда за C# и Visual basic, који омогућавају писање изражајног декларативног кода.

Linq технологија има широку примену у програмирању и конкретно обрати и приступу подацима. Велики број данашњих софтвера користи неку од технологија као што су базе података, JSON, XML и друге за обраду и манипулацију подацима. Често то подразумева да је потребно научити нови API да би се приступало подацима што може да буде замарајуће. LИНК ово поједностављује апстраховањем уобичајених елемената приступа подацима у синтаксу упита која изгледа исто без обзира на извор података који се одабере.

Можемо то видети на следећем примеру[14]:

```
public static IEnumerable<XElement> FindAllElementsWithAttribute(XElement
documentRoot, string elementName, string attributeName, string value)
{
    return from el in documentRoot.Elements(elementName)
    where (string)el.Element(attributeName) == value
    select el;
}
```

Писање кода за ручно извлачење података из XML фајла би било доста захтевније.

У студијском примеру обрада података је смештена у три врсте SQL server објекта и то Stored Procedures, Table Functions, и Value Functions којима се обезбеђује заштићена обрада података (не постоји могућност коришћења стандардних sql исказа из апликације већ је она омогућена искључиво из омотача, за SQL server објекте који су смештени у делу SrednjiSloj), тако да Linq нисмо користили.

2.2 Релациона база података

Релациона база података је врста базе података која складишти и омогућава приступ табелама података које су међусобно повезане. Релационе базе података засноване су на релационом моделу, интуитивном, једноставном начину представљања података у табелама. У релационој бази података, сваки ред у табели је запис са јединственим ИД-ем који се назива примарни кључ. Колоне табеле садрже

атрибуте података, а сваки запис обично има вредност за сваки атрибут, што олакшава успостављање односа међу табелама података[15].

Релационим базама података се управља преко програма који се називају системи за управљање релационим базама података (РДБМС). У овом студијском примеру је коришћен SQL SERVER, који је првенствено дизајниран и развијен од стране *Microsoft-a*, да би био конкуренција *Oracle*-овом MySQL – у[16].

SQL SERVER подржава ANSI SQL, који је стандардни језик SQL. Међутим, СКЈ Сервер долази са сопственом имплементацијом SQL језика, T- SQL.

T- SQL је *Microsoft* - ов језик познат као трансакциони SQL. Он пружа даље могућности декларисања променљиве, руковања изузетцима, ускладиштене процедуре итд.

SQL Server Management Studio (SSMS) је главни алат за интерфејс за SQL SERVER и подржава 32-битна и 64-битна окружења.

2.3 Патерни

Израз „дизајнерски патерн“ (*design pattern*) настао је 1995. године како би ујединио дизајнерска решења искусних програмера која су се примењивала у тада актуелним програмима. Намера је била да почну да се структурирају генеричка решења за проблеме са којима се сви програмери сусрећу на дневном нивоу док развијају и дизајнирају апликације[17].

Постоји много дефиниција патерна. По једној од дефиниција патерн се дефинише као: „Сваки образац је троделно правило које изражава однос између одређеног контекста, проблема и решења.“ Друга дефиниција је шире применљива: „Узорак је апстракција из конкретног облика који се стално понавља у специфичним произвољним контекстима[17].

Дизајн обрасци [18] олакшавају стварање висококвалитетних дизајна. Они нису само корисни у стварању софтверских система, већ помажу и у анализи постојећих система, нпр. у реконструкцији документације наслеђеног система из његовог изворног кода, односно обрнути инжењеринг. Аутоматизација обрнутог инжењеринга може бити ефикаснија ако се препознају обрасци пројектовања. Ако се правилно препознају, анализа може адекватно одразити намере дизајна. Ако се ове намере забележе у реконструисаној документацији, будуће одржавање и развој наслеђеног система требало би да буду много лакши и јефтинији.

2.3.1 MVC патерн

Model-view-controller (познатији по скраћеници МВЦ) је архитектурални патерн за развој софтвера који се обично користи за развој корисничких интерфејса који програмску логику деле на три повезана елемента, и то модел, поглед и контролер. Ово се ради како би се интерна репрезентација информација одвојила од начина на који се те исте информације представљају и прихватају од корисника [19].

Традиционално коришћен за графичке корисничке интерфејсе, овај патерн је постао нарочито популаран за дизајнирање веб апликација[20]. За популарне програмске језике најчешће постоје радни оквири који олакшавају имплементацију МВЦ-а.

Три компоненте МВЦ-а су модел, поглед и контролер.

1. Модел је централна компонента овог патерна. Он представља динамичку структуру података саме апликације и независан је од корисничког интерфејса. Модел директно управља подацима, логиком и правилима апликације.
2. Поглед представља сваки приказ информација, било да је у питању график, дијаграм, табела или обични текст. Могуће је имати више погледа који одговарају истим информацијама.
3. Контролер прихвата улазе и пребацује их у команде за модел или поглед.

У то што апликацију дели на ове компоненте, МВЦ дефинише и везе између њих:

1. Модел је задужен за управљање подацима апликације. Своје улазе прима од контролера.
2. Поглед даје визуелну репрезентацију модела у одређеном формату.
3. Контролер одговара на корисничке улазе и обавља интеракцију са објектима модела. Контролер прима улаз, опционо га валидира, те га потом шаље моделу.

Као и са другим добрим софтверским патернима, МВЦ даје „основу решења“ за проблем, али ипак дозвољава прилагођавање конкретном систему. Одређене имплементације МВЦ-а могу знатно да варирају једна од друге.

МВЦ је био један од првих приступа архитектуралном дизајну апликација који је описао и имплементирао софтверске концепте према њиховим задужењима. [21] Овај патерн од свог постанка знатно развијао, те су настали други патерни

инспирисани њим као што су *hierarchical model–view–controller* (HMVC), *model–view–adapter* (MVA), *model–view–presenter* (MVP) и *model–view–viewmodel* (MVVM).

МВЦ је постао популаран међу Јава програмерима развојем оквира као што је Spring који је базиран на овом патерну, а његовом даљем развоју у том погледу су у великој мери помогли оквири као што су Django и Ruby on Rails.

Иако је оригинално развијен за десктоп програмирање, МВЦ је током времена прилагођен као дизајн за World Wide Web, тј. за развој веб апликација. Креирано је много веб оквира који користе овај патерн. Ови оквири се најчешће разликују према томе који део задужења се налази на клијентској, а који на серверској страни.

2.3.2 Singleton патерн

Синглтон има за циљ да осигура да класа има само једну своју инстанцу. То значи да треба омогућити да синглтон инстанца може поново да се користи. Једини начин да се сачува синглтон инстанца за каснију поновну употребу је да јесте у статичком (или глобалном) контексту. Присуство статичке (или глобалне) променљиве синглтон типа је само неопходан услов. Да би се идентификовао прави синглтон, морамо осигурати да не постоји неправилна употреба класе, тј. нова инстанца синглтона се инстанцира само за иницијализацију статичке (или глобалне) синглтон променљиве. Будући да би могло бити тешко проверити за сваку путању извршења да ли је додељена инстанца статичке променљиве, лакше је да се провери само да ли таква путања извршења постоји[22].

Пример имплементације у студијском примеру у програмском језику C#:

```
public static bool Connect(ref string _Error)
{
    bool fja = false;
    if (m_DB == null)
    {
        if (Properties.Settings.Default.Trusted)
        {
            try
            {
                m_DB = new
```

Како је апликација реализована у трослојном моделу (архитектури) ради могућности коришћења делова апликације у другим архитектурама (web сервис, datawarehouse,) потребно је обезбедити једноструко приступање извору података (без обзира на број

корисника апликације оставрује се само једна конекција ка извору података). Такав приступ је омогућен кроз реализовање *SrednjiSloj* као Синглтон инстанца објекта.

2.3.3 Template method патерн

Дефинише скелет алгоритма у операцији, препуштајући извршење неких корака операција подкласама. Template method патерн омогућава подкласама да редефинишу неке од корака алгоритма без промене алгоритамске структуре[23]. Ови кораци су имплементирани уз помоћ додатних помоћних (*helper*) метода у оквиру самих класа над којима се позива шаблонска метода. Помоћне методе могу бити или апстрактне методе, при чему поткласе морају пружити конкретну имплементацију или *hook* методе које имају празна тела у суперкласи. Подкласе могу прилагодити операције преписивањем ових метода. Сврха шаблонске методе је да дефинише целокупну структуру операције при чему поткласама дозвољава да редефинишу или прецизирају одређене кораке[24].

3. Студијски пример

У овом поглављу ће бити представљен студијски пример развоја софтверског система вођење евиденције о пословању књижара и складиштењу књига применом .NET технологија. За развој софтверског система у овом примеру је коришћења упрошћена Ларманова метода.

3.1 Ларманова метода

Ларманова метода за развој софтвера се заснива на итеративно-инкременталном моделу животног циклуса софтвера. Упрошћена Ларманова метода се састоји од следећих фаза[25]:

- **Фаза прикупљања захтева од корисника** – у овој фази се дефинишу својства и услови које софтверски систем или шире гледајући пројекат треба да задовољи. Она постоји да би се идентификовали и дефинисали захтеви корисника везаних за пројекат, тј. софтверски систем. Захтеви могу

бити функционални и нефункционални. У функционалним се дефинишу захтеване функције система, док се остали сматрају за нефункционалне (захтеви везани за перформансе поузданост, употребљивост, подржаност система). Функционални захтеви су описани преко модела случајева коришћења, који садрже елементе структуре и понашања система.

- **Фаза анализе** – у овој фази се описује логичка структура и понашање система односно пословна логика. Понашање се описује преко дијаграма секвенци (за сваки случај коришћења) и системских операција (на основу дијаграма секвенци), а структура се описује преко релационог и концептуалног модела.
- **Фаза пројектовања** – у овој фазу се приказује архитектура софтверског система односно понашање и физичка структура . Углавном је то тронивојска архитектура која се састоји из корисничког интерфејса, апликационе логике и складишта података. Ниво корисничког интерфејса се налази на клијентској страни, док се апликациона логика и складиште података налазе на страни сервера.
- **Фаза имплементације** – у овој фази се имплементирају све компоненте које су добијене пројектовањем архитектуре система и реализују у некој од постојећих технологија (у овом случају су коришћене *.NET* технологије).
- **Фаза тестирања** – у овој фази се свака од компоненти које су имплементирани тестира.

3.2 Прикупљање корисничких захтева

Ово је први корак у процесу развоја софтвера коришћењем упрошћене Ларманове методе. За овај корак је потребно одвојити довољно времена јер је јако битно да се захтеви корисника идентификују на одговарајући начин. Потенцијалне грешке које не буду откривене у овој фази развоја могу значајно да утичу на ток развоја јер може бити потребно доста времена за њихово разрешавање у каснијим фазама.

3.2.1 Вербални опис

Потребно је имплементирати апликацију за вођење евиденције о пословању књижара и складиштењу књига. У систему постоји више различитих књижара које поседују по једно главно складиште и добављач који је задужен за добављање књига. Поред претходно наведених ентитета, постоји финансијска служба и управа.

Потребно је водити евиденцију о књигама које се налазе у складишту, књигама које су продате другим књижарама, требовању књига из складишта, као и свим документима који се налазе у оквиру система.

Корисници система су администратор складишта књижаре, запослени радници у књижарама, администратор финансијске службе, администратор управе и добављач.

Систем нуди различит спектар функционалности за сваки тип корисника.

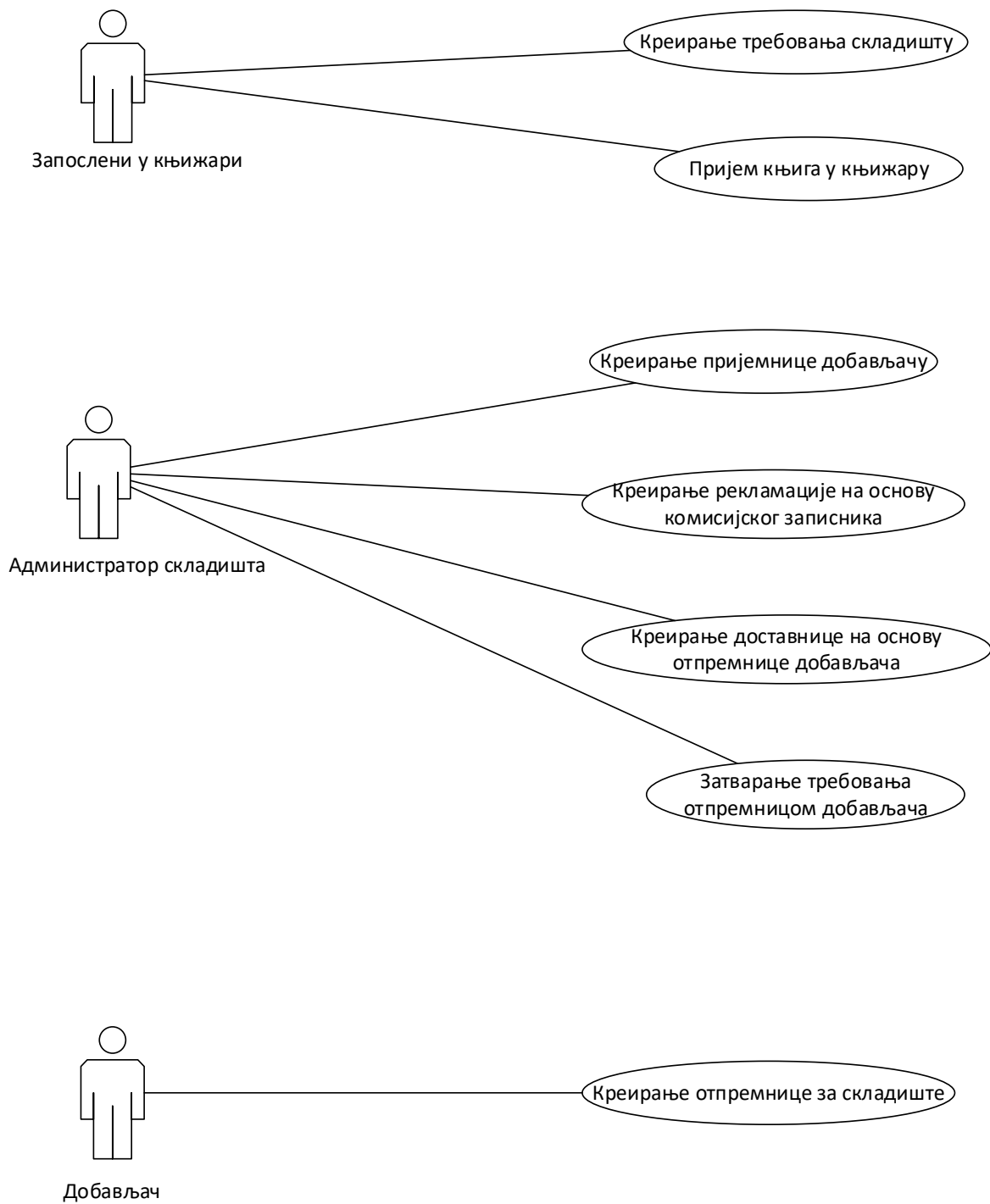
Запослени у књижарама могу да затраже добављање књига из складишта. Приликом тога попуњавају требовање које потом шаљу складишту. Администратор складишта евидентира требовање. Након тога, администратор складишта шаље пријемницу достављачу и копију тог документа доставља финансијској служби. По пријему пријемнице, добављач формира отпремницу и шаље је складишту.

Након пријема, отпремнице и робе, администратор складишта формира комисијски записник о пријему робе и прослеђује га управи, формира доставницу и шаље је назад књижари и шаљу по примерак доставнице финансијској служби. Администратори складишта су у могућности да пријаве рекламацију над робом коју су примили.

Треба омогућити добру повезаност између форми у апликацији.

3.2.2 Случајеви коришћења

1. Креирање требовања складишту
2. Креирање пријемнице добављачу
3. Креирање отпремнице за складиште
4. Креирање рекламације на основу комисијског записника
5. Креирање доставнице на основу отпремнице добављача
6. Затварање требовања отпремницом добављача
7. Пријем књига у књижару



Слика 5 Дијаграм случајева коришћења

За спровођење било ког од случајева коришћења предуслов је да је систем укључен и да било ко од запослених са главне форме бира опцију "Активности - Промет". Тада нам систем омогућава приступ форми са које је могућ даљи рад кроз апликацију као и могућност логовања за различите улоге (запослене) у апликацији.

СК1: Случај коришћења – Креирање требовања складишту

Назив СК

Креирање требовања складишту

Актори СК

Запослени у књижари

Учесници СК

Запослени у књижари и систем (програм)

Предуслов: Систем је укључен и **запослени у књижари** је улогован под својом шифром. **Запослени у књижари** бира опцију „Требовање“ са главне форме. Систем приказује форму за унос **требовања**. Учитана је листа књига.

Основни сценарио СК

1. **Запослени у књижари** додаје књиге и жељене количине. (АПУСО)
2. **Запослени у књижари** контролише да ли је коректно унео податке за тражено **требовање**. (АНСО)
3. **Запослени у књижари** позива **систем** да запамти податке о датом **требовању** (АПСО)
4. **Систем** памти податке о изабраним књигама. (СО)
5. **Систем** аутоматски **креира пријемницу** за складиште и **приказује запосленом у књижари** поруку: „Подаци су успешно унети“. (ИА)

Алтернативни сценарио

- 5.1 Уколико **систем** није у могућности да креира требовање због неправилног уноса, он **приказује запосленом у књижари** поруку: „Мора се унети број књига веће од 0“. (ИА)

СК2: Случај коришћења – Креирање пријемнице добављачу

Назив СК

Креирање пријемнице добављачу

Актори СК

Администратор складишта

Учесници СК

Администратор складишта и систем (програм)

Предуслов: Систем је укључен и администратор складишта је улогован под својом шифром. Администратор складишта бира опцију „Креирање Пријемнице за добављача“. Систем враћа списак свих постојећих требовања која су извршена према датом складишту.

Основни сценарио СК

1. Администратор складишта бира требовање за које жели да креира пријемницу. (АПУСО)
2. Администратор складишта позива систем да запамти податке о пријемници. (АПСО)
3. Систем памти податке о креираном пријемници. (СО)
4. Систем аутоматски креира пријемницу за добављача и приказује администратору складишта поруку: „Подаци су успешно унети“. (ИА)

Алтернативни сценарио

- 4.1 Уколико је пријемница већ креирана, систем приказује администратору складишта поруку: „За ову пријемницу је већ послат захтев добављачу“. (ИА)

СКЗ: Случај коришћења – Креирање отпремнице за складиште

Назив СК

Креирање отпремнице

Актори СК

Добављач

Учесници СК

Добављач и систем (програм)

Предуслов: Систем је укључен и **Добављач** је улогован под својом шифром. **Добављач** бира опцију „Да ли има захтева за испоруком књига (пријемница)“. Систем враћа списак свих постојећих пријемница која су послате.

Основни сценарио СК

1. **Добављач бира** жељену пријемницу за коју жели да креира отпремницу за складиште. (АПУСО)
2. **Добављач контролише** да ли је коректно унео податке за отпремницу. (АНСО)
3. **Добављач позива систем** да запамти податке о креираној отпремници. (АПСО)
4. **Систем памти** податке о креираној отпремници. (СО)
5. **Систем приказује добављачу** поруку „Подаци су успешно унети“. (ИА)

Алтернативни сценарио

- 5.1 У случају да је **добављач** променио потребну количину књига за отпремницу **систем приказује добављачу** поруку „Подаци су успешно унети“. (ИА)

СК4: Случај коришћења – Креирање рекламације на основу комисијског записника

Назив СК

Креирање рекламације

Актори СК

Администратор складишта

Учесници СК

Администратор складишта и систем (програм)

Предуслов: Систем је укључен и администратор складишта је улогован под својом шифром. Администратор складишта бира опцију „Креирање Пријемнице за добављача“. Систем враћа списак свих постојећих требовања која су извршена према датом складишту.

Основни сценарио СК

1. Администратор складишта бира отпремницу коју жели комисијски да прими и за њу креира комисиони записник. (АПУСО)
2. Администратор складишта контролише да ли је коректно унео податке за комисиони записник. (АНСО)
3. Администратор складишта позива систем да запамти податке о изабраним књигама. (АПСО)
4. Систем памти податке о изабраним књигама. (СО)
5. Систем приказује администратору складишта поруку: „Подаци су успешно унети“. (ИА)

Алтернативни сценарио

- 5.1 Уколико подаци о књигама нису успешно унети, систем приказује администратору складишта поруку: „Подаци нису успешно унети“. (ИА)

СК5: Случај коришћења – Креирање доставнице на основу отпремнице добављача

Назив СК

Креирање доставнице

Актери СК

Администратор складишта

Учесници СК

Администратор складишта и систем (програм)

Предуслов: Систем је укључен и администратор складишта је улогован под својом шифром. Администратор складишта бира опцију „Да ли има захтева из књижара (требовања)“. Систем враћа списак свих постојећих требовања која су упућена према датом складишту.

Основни сценарио СК

1. Администратор складишта бира требовање за које жели да креира доставницу за књижару. (АПУСО)
2. Администратор складишта контролише да ли је коректно унео податке за доставницу. (АНСО)
3. Администратор складишта позива систем да запамти податке о изабраној доставници. (АПСО)
4. Систем памти податке о креираној доставници. (СО)
5. Систем аутоматски креира доставницу за запосленог у књижари и приказује администратору складишта поруку: „Подаци су успешно унети“. (ИА)

Алтернативни сценарио

- 5.1 Уколико подаци нису успешно унети, систем приказује администратору складишта поруку: „Подаци нису успешно унети“. (ИА)

СК6: Случај коришћења – Пријем књига у књижару

Назив СК

Пријем Књига

Актери СК

Запослени у књижари

Учесници СК

Запослени у књижари и систем (програм)

Предуслов: Систем је укључен и **Запослени у књижари** је улогован под својом шифром. **Запослени у књижари** бира опцију „Да ли има нешто за пријем из складишта“. Систем враћа списак доставница за које треба извршити пријем књига. Учитана је листа књига.

Основни сценарио СК

1. **Запослени у књижари** бира доставницу за коју жели да прими **књиге** из складишта. (АПУСО)
2. **Администратор складишта контролише** да ли је коректно унео податке за **књиге** које ће бити примљене. (АНСО)
3. **Запослени у књижари позива систем** да запамти податке о **књигама** које ће бити примљене у књижару. (АПСО)
4. **Систем памти** податке. (СО)
5. **Систем приказује запосленом у књижари** поруку „Подаци су успешно унети“. (ИА)

Алтернативни сценарио

5.1 Уколико подаци нису успешно унети, **систем приказује запосленом у књижари** поруку: „Подаци нису успешно унети“. (ИА)

3.3 Анализа

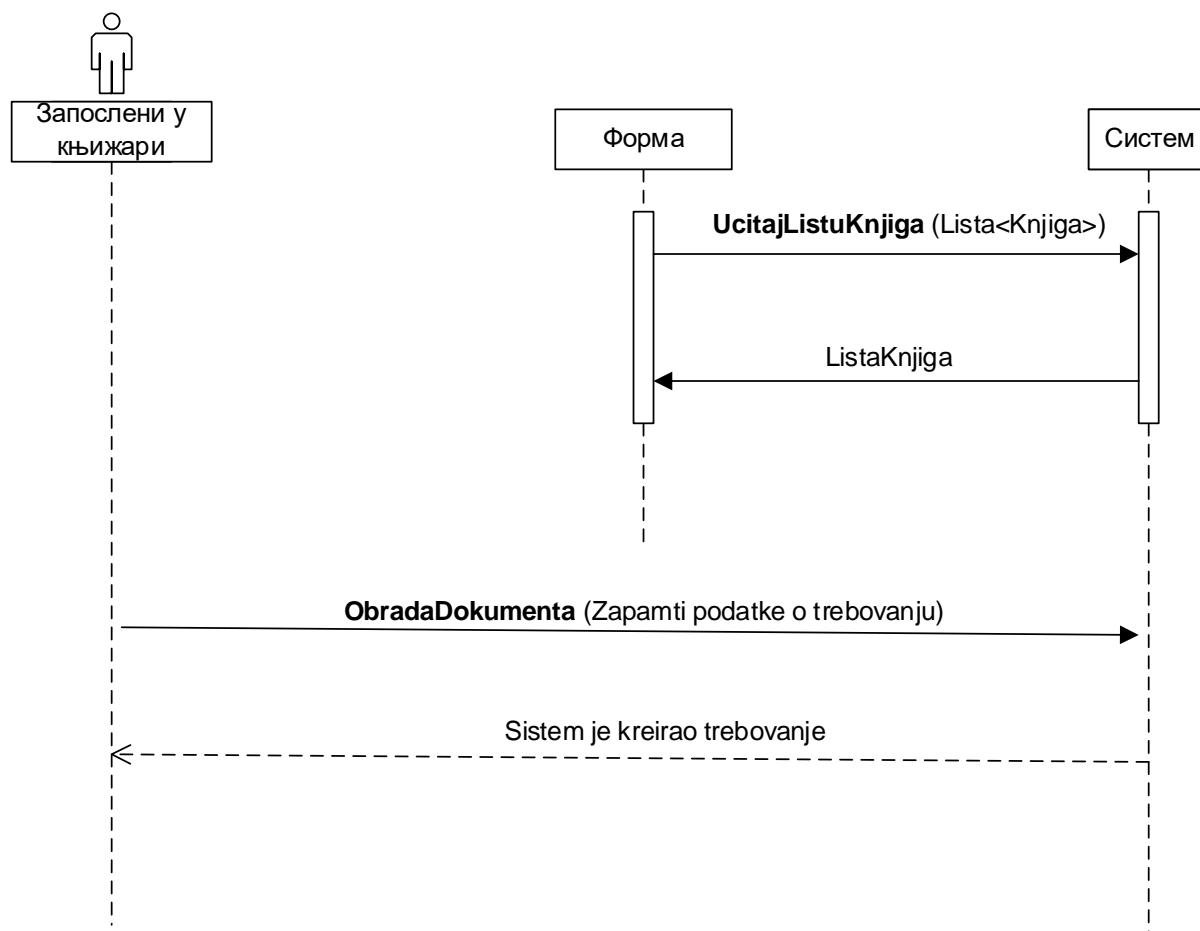
Ова фаза дефинише пословну логику система. Понашање је приказано кроз системске дијаграме секвенци који идентификују системске операције које треба бити имплементирани у фази имплементацији. Концептуални и релациони модел дају опис структуре софтверском система[25].

3.3.1 Системски дијаграми секвенци

У овом поглављу ће бити приказани системски дијаграми секвенци случајева коришћења који су идентификовани у овом раду. Они приказују догађаје у одређеном редоследу преко којих је дата интеракција актора (конкретно запосленог у књижари, администратора складишта и добављача) и софтверског система, за посебне сценарије случајева коришћења.

ДС1: Дијаграм секвенци случаја коришћења – Креирање требовања складишту

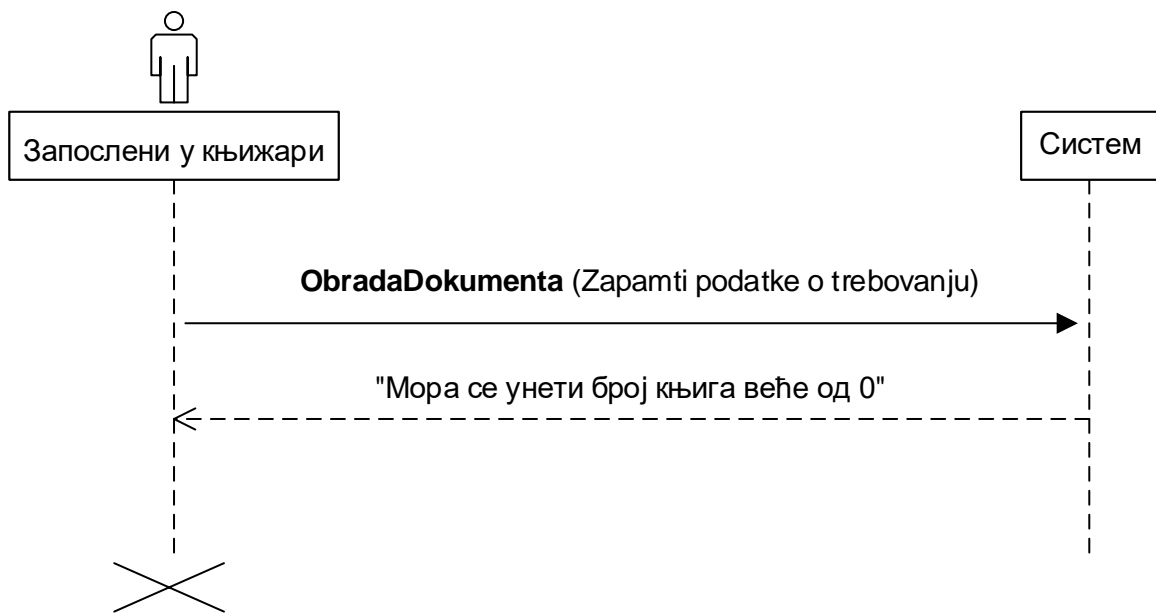
1. **Форма** позива **систем** да учита књиге. (АПСО)
2. **Систем** читава **листу књига**.(ИА)
3. **Запослени у књижари** **позива систем** да запамти податке о изабраним књигама (АПСО)
4. **Систем** аутоматски **креира требовање** за складиште и **приказује запосленом у књижари** поруку: „Подаци су успешно унети“. (ИА)



Слика 6 Креирање требовања складишту

Алтернативни сценарио

- 5.1. Уколико **систем** није у могућности да креира требовање због неправилног уноса, он **приказује зависленом у књижари** поруку: „Мора се унети број књига веће од 0“. (ИА)



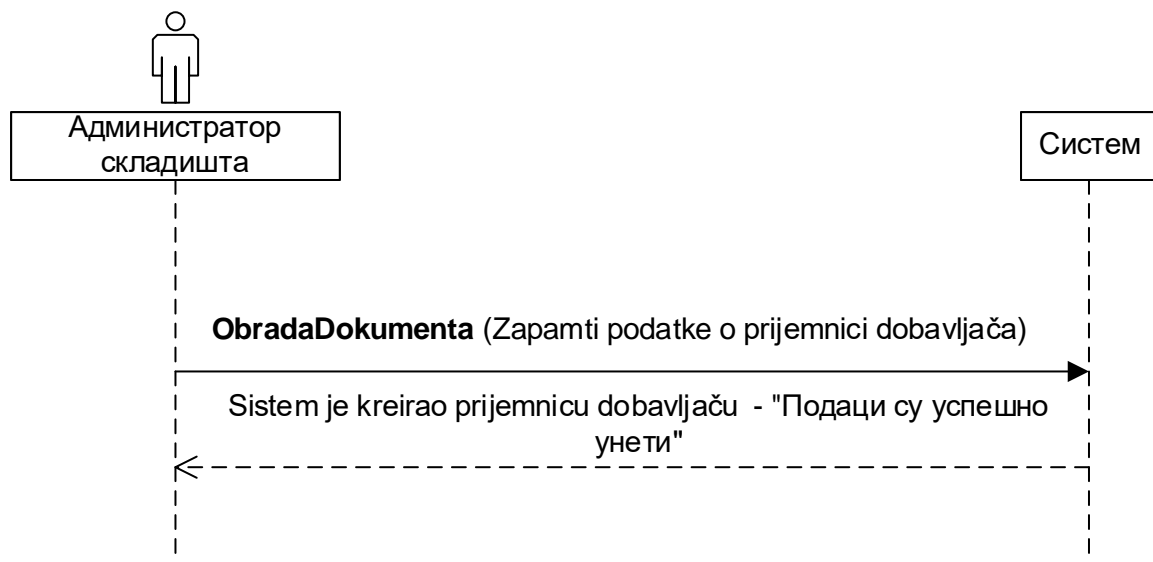
Слика 7 Неуспешно креирање требовања

Са наведених секвенцих дијаграма уочава се једна системска операција:

1. Signal **UcitajListuKnjiga** (Lista <Knjiga>)
2. Signal **ObradaDokumenta** (Trebovanje)

ДС2: Дијаграм секвенци случаја коришћења – Креирање пријемнице добављачу

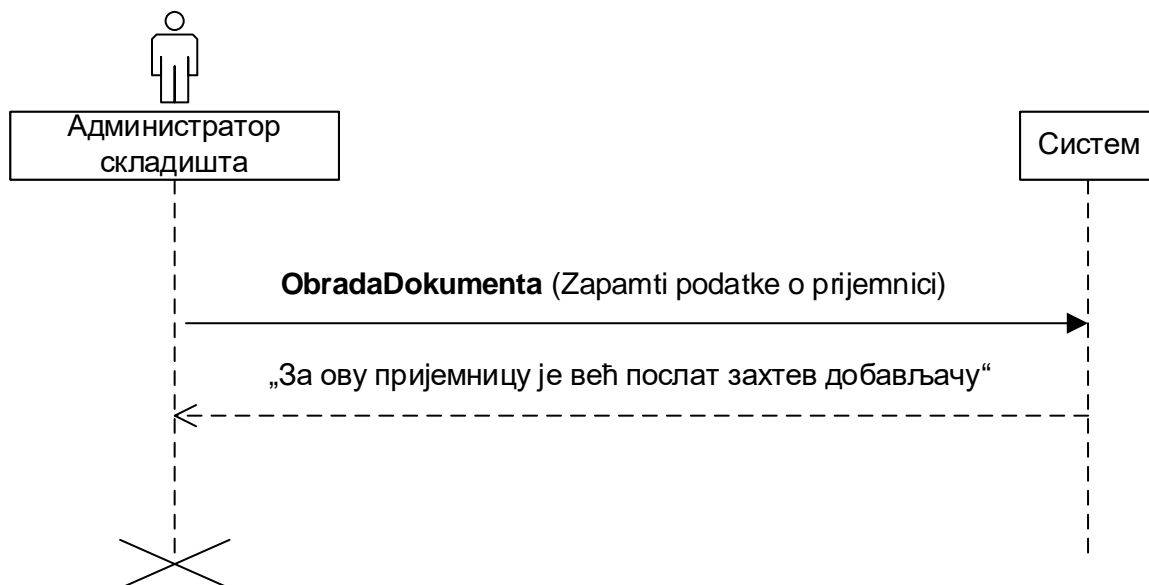
1. **Администратор складишта** **позива** **систем** да запамти податке о **пријемници**. (АПСО)
2. **Систем** аутоматски **креира** **пријемницу** за добављача и **приказује** **администратору складишта** поруку: „Подаци су успешно унети“. (ИА)



Слика 8 Креирање пријемнице добављачу

Алтернативни сценарио

4.1 Уколико је **пријемница** већ креирана, **систем** приказује **администратору складишта** поруку: „За ову **пријемницу** је већ послат захтев добављачу“. (ИА)



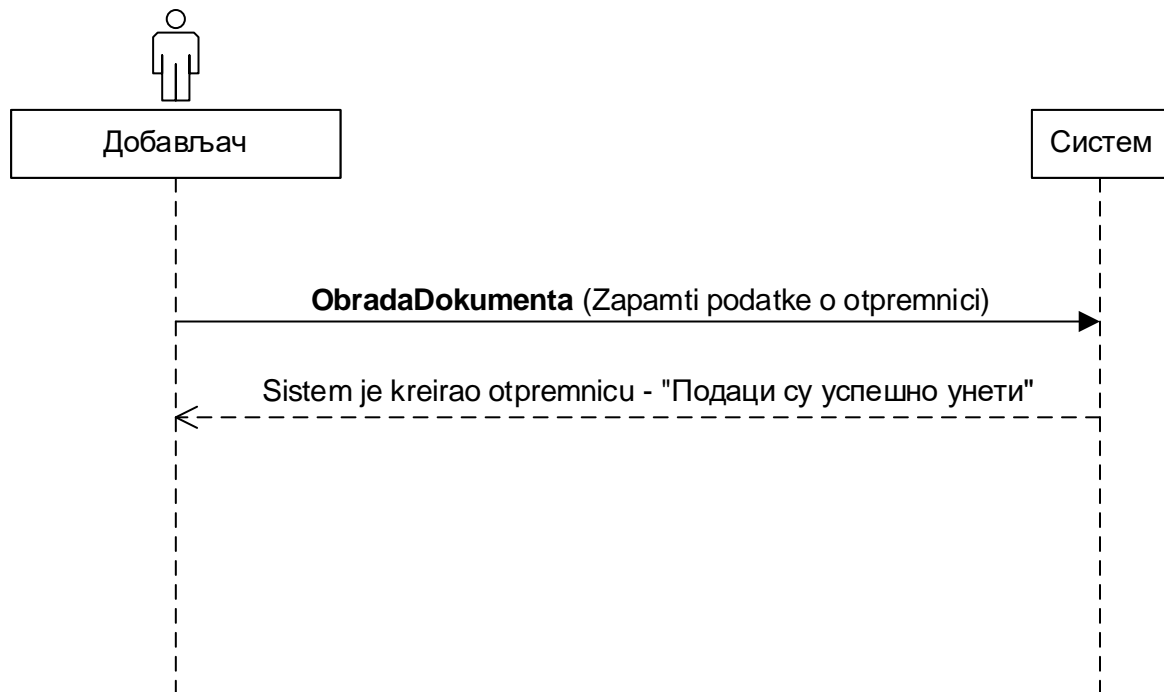
Слика 9 Неуспешно креирање пријемнице добављачу

Са наведених секвенцих дијаграма уочава се једна системска операција:

1. Signal **ObradaDokumenta** (Prijemnica)

ДСЗ: Дијаграм секвенци случаја коришћења – Креирање отпремнице за складиште

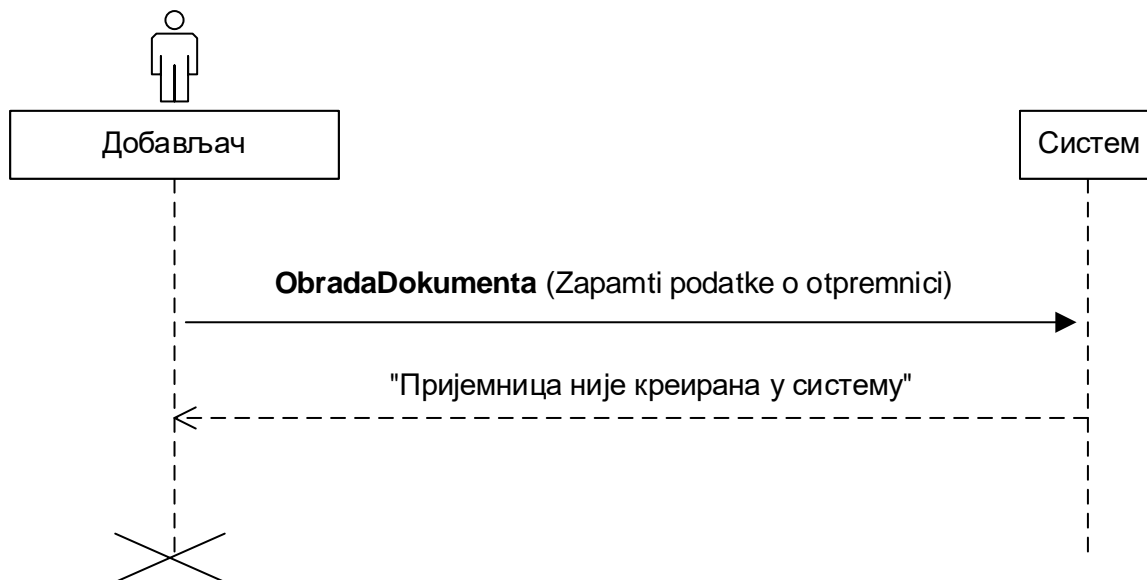
1. **Добављач** позива **систем** да запамти податке о креираној **отпремници**. (АПСО)
2. **Систем** приказује **добављачу** поруку „Подаци су успешно унети“. (ИА)



Слика 10 Креирање отпремнице за складиште

Алтернативни сценарио

5.1 У случају да је **добављач** променио потребну количинну књига за **отпремницу** **систем** приказује **добављачу** поруку „Подаци су успешно унети“. (ИА)



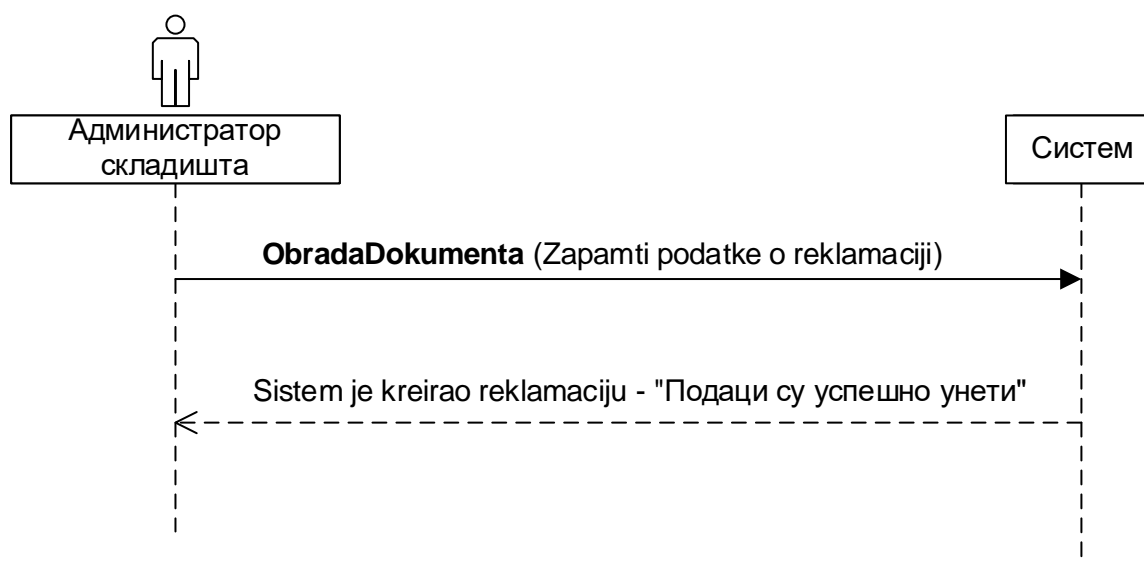
Слика 11 Креирање отпремнице за складиште

Са наведених секвенцих дијаграма уочава се једна системска операција:

1. Signal **ObradaDokumenta** (docid)

ДС4: Дијаграм секвенци случаја коришћења – Креирање рекламације на основу комисијског записника

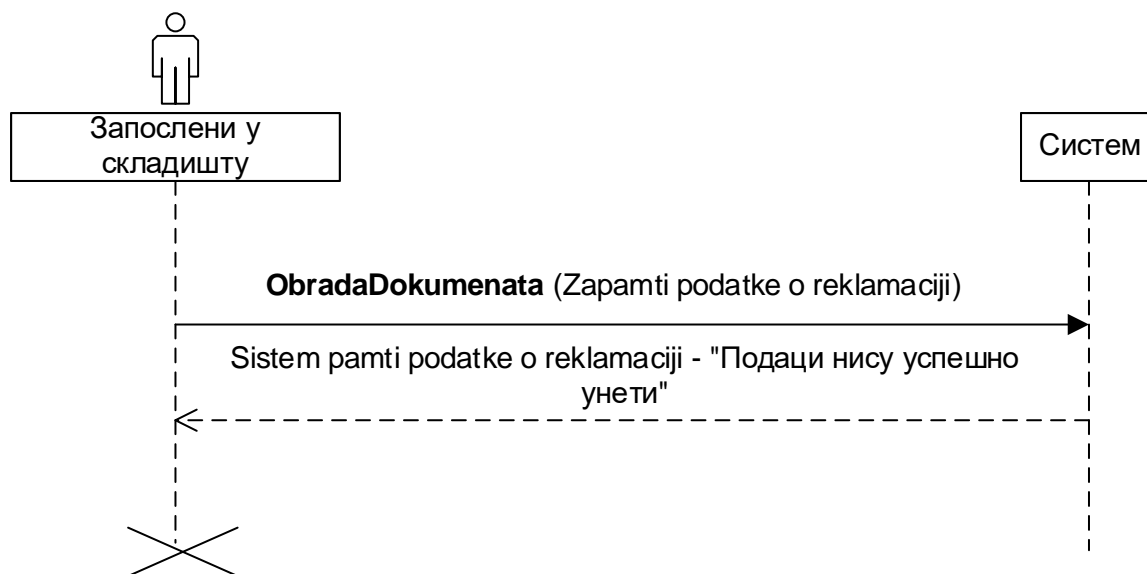
5. **Администратор складишта** позива систем да запамти податке о изабраним **књигама**. (АПСО)
6. **Систем приказује администратору складишта** поруку: „Подаци су успешно унети“. (ИА)



Слика 12 Креирање рекламације

Алтернативни сценарио

- 5.1 Уколико подаци о **књигама** нису успешно унети, **систем приказује администратору складишта** поруку: „Подаци нису успешно унети“. (ИА)



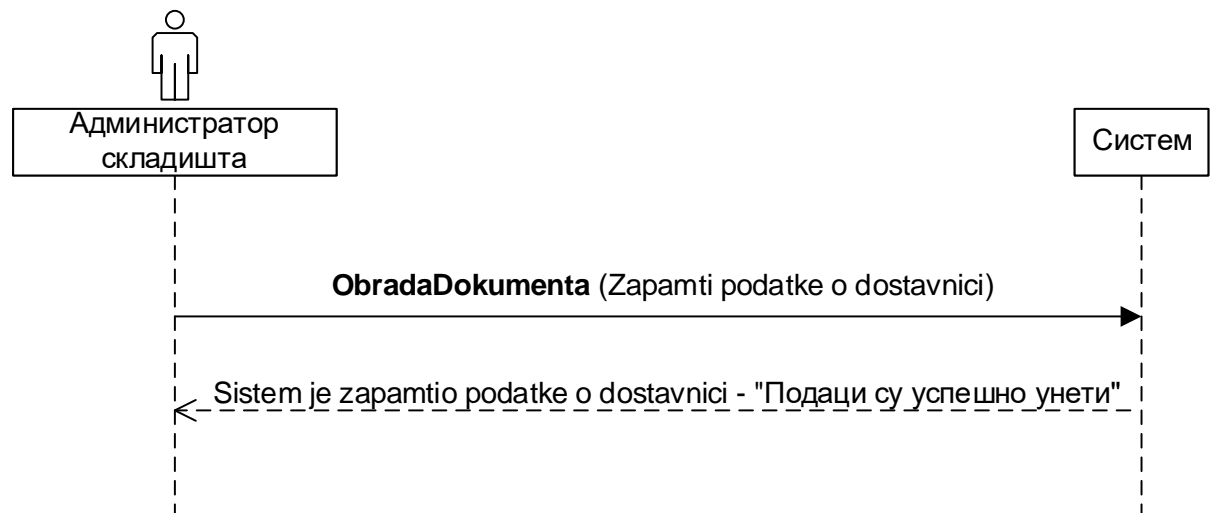
Слика 13 Неуспешно креирање рекламације

Са наведених секвенцих дијаграма уочавају се три системска операције:

1. Signal **ObradaDokumenta** (Reklamacija)

ДС5: Дијаграм секвенци случаја коришћења – Креирање доставнице на основу отпремнице добављача

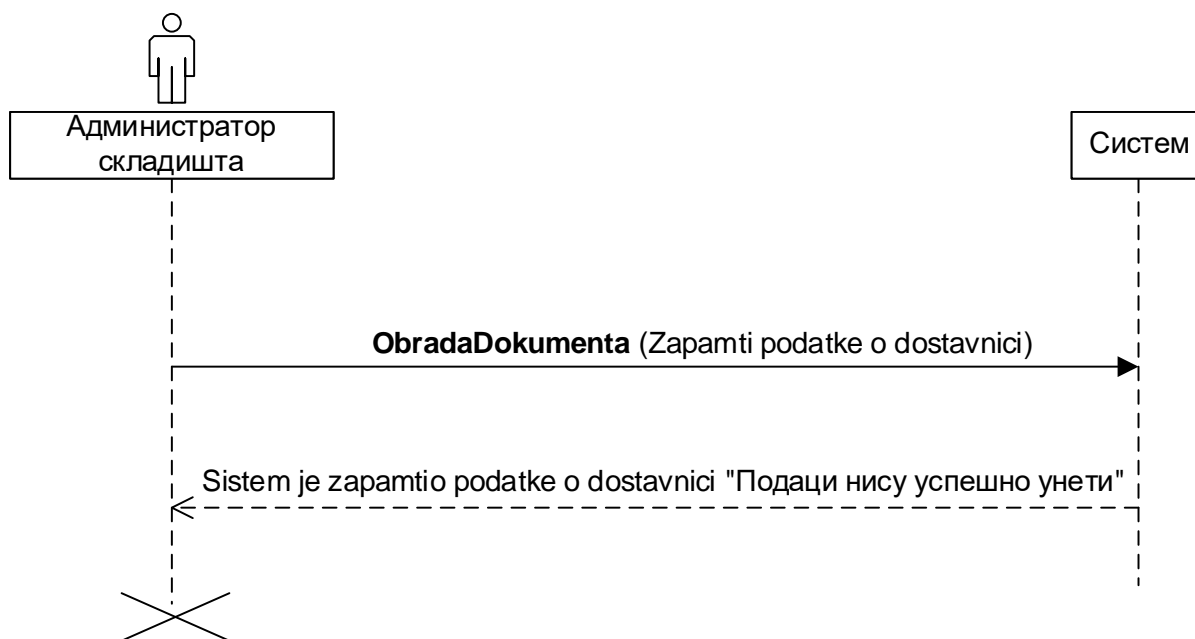
4. **Администратор складишта** позива систем да запамти податке о изабраној доставници. (АПСО)
5. **Систем** аутоматски **креира доставницу** за запосленог у књижари и **приказује администратору складишта** поруку: „Подаци су успешно унети“. (ИА)



Слика 14 Креирање доставнице на основу отпремнице добављача

Алтернативни сценарио

- 5.1 Уколико подаци нису успешно унети, **систем приказује администратору складишта** поруку: „Подаци нису успешно унети“. (ИА)



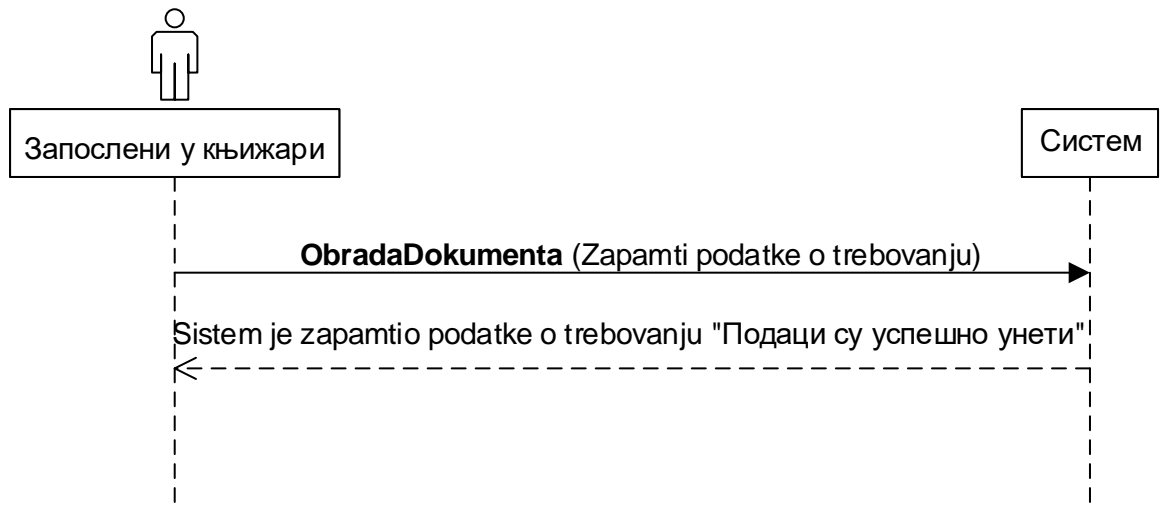
Слика 15 Неуспешно креирање доставнице на основу отпремнице добављача

Са наведених секвенцих дијаграма уочавају се две системске операције:

1. Signal **ObradaDokumenta** (Dostavnica)

ДС6: Дијаграм секвенци случаја коришћења – Пријем књига у књижару

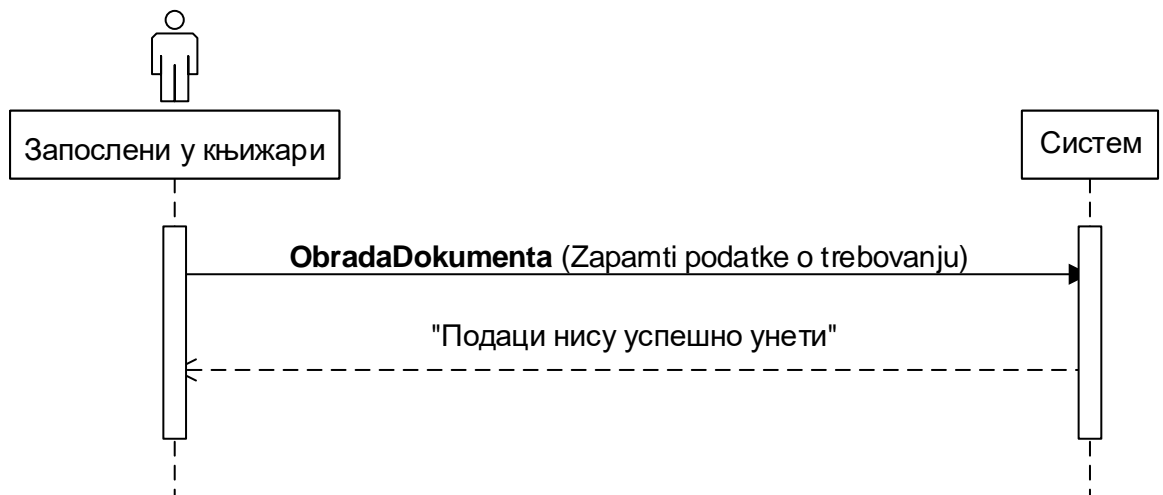
4. **Запослени у књижари** **позива** **систем** да запамти податке о **књигама** које ће бити примљене у књижару.(АПСО)
5. **Систем** **приказује** **запосленом у књижари** поруку „Подаци су успешно унети“. (ИА)



Слика 16 Пријем књига у књижару

Алтернативни сценарио

5.1 Уколико подаци нису успешно унети, **систем приказује запосленом у књижари** поруку: „Подаци нису успешно унети“. (ИА)



Слика 17 Пријем књига у књижару

Са наведених секвенцих дијаграма уочавају се три системске операције:

1. Signal **ObradaDokumenta** (Trebovanje)

На основу анализе сценарија добијене су 2 системске операције:

1. Signal **UcitajListuKnjiga** (Lista <Knjiga>)
2. Signal **ObradaDokumenta** (docid)

3.3.2 Понашање софтверског система – Дефинисање уговора о системским операцијама

У овом одељку ће бити представљени уговори свих системских операција. Сваки уговор је дефинисан својим називом, везом са случајевима коришћења у којима је искоришћен, условима који морају бити задовољени пре извршења системске операције, као и стањима у којима мора да буде систем након извршења системске операције.

Уговор УГ1: **VratiSpisakKnjiga** (DataTable): Signal;

Веза са СК: СК1, СК2, СК3, СК4, СК5, СК6, СК7

Предуслови: /

Постуслови: /

Уговор УГ2: **VratiMagacinZaKnjzaru** (Magacin): Signal;

Веза са СК: СК1, СК2, СК3, СК4, СК5, СК6, СК7

Предуслови: /

Постуслови: /

Уговор УГ3: **VratiSpisakPrijemnica**: Signal;

Веза са СК: СК2

Предуслови: /

Постуслови: /

Уговор УГ4: **VratiStavkeDokumenata** (Dokument): Signal;

Веза са СК: СК1, СК2, СК3, СК4, СК5, СК6, СК7

Предуслови: /

Постуслови: /

Уговор УГ5: **VratiPodatkeODokumentu** (Dokument): Signal;

Веза са СК: СК1, СК2, СК3, СК4, СК5, СК6, СК7

Предуслови: /

Постуслови: /

Уговор УГ6: **VratiViewData** (DataTable): Signal;

Веза са СК: СК1, СК2, СК3, СК4, СК5, СК6, СК7

Предуслови: /

Постуслови: /

Уговор УГ7: **SinhroTrebovanje** (Dokument, Objekti): Signal;

Веза са СК: СК2

Предуслови: Вредносна и структурна ограничења над објектом **Dokument** и **Objekti** морају бити задовољена – позитивна вредност улазног параметра

Постуслови: Синхронизује статус требовања на основу пријемнице добављачу за дати магацин.

Уговор УГ8: **VratiKnjizareZaMagacin** (Objekti): Signal;

Веза са СК: СК1, СК2, СК3, СК4, СК5, СК6

Предуслови: /

Постуслови: /

Уговор УГ9: **VratiSpisakMagacina** (): Signal;

Веза са СК: СК1, СК2, СК3, СК4, СК5, СК6, СК7

Предуслови: /

Постуслови: /

Уговор УГ10: **VratiPrometZa** (Magacin): Signal;

Веза са СК: СК3

Предуслови: /

Постуслови: /

Уговор УГ11: **VratiStanjeZa** (Objekti): Signal;

Веза са СК: СК1, СК2, СК3, СК4, СК6, СК7

Предуслови: /

Постуслови: /

Уговор УГ12: **VratiObjekatInfo** (Objekat): Signal;

Веза са СК: СК1, СК2, СК3, СК4, СК5, СК6, СК7

Предуслови: /

Постуслови: /

Уговор УГ13: **VratiOtpremnicuZaReklamaciju** (Dokument): Signal;

Веза са СК: СК4

Предуслови: /

Постуслови: /

Уговор УГ14: **ObradaDokumenta** (Dokument): Signal;

Веза са СК: СК1, СК2, СК3, СК4, СК5, СК6

Предуслови: Вредносна и структурна ограничења над објектом **Dokument**, **Objekti** и **VrstaRM** морају бити задовољена.

Постуслови: Евидентирани и запамћени су подаци о документу (требовање, доставница, отпремница...)

Уговор УГ15: **Promet** (Dokument): Signal;

Веза са СК: СК1, СК2, СК3, СК4, СК5, СК6

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

Постуслови: Евидентиран промет, додата је ставка у оквиру промета.

Уговор УГ16: **AzurIznos** (Dokument): Signal;

Веза са СК: СК4

Предуслови: Вредносна и структурна ограничења над објектом **Dokument** морају бити задовољена – позитивна вредност улазног параметра

Постуслови: Враћа ажуриран статус документа у случају да постоји рекламација.

Уговор УГ17: **Povezi** (Dokument, Dokument): Signal;

Веза са СК: СК6

Предуслови: Вредносна и структурна ограничења над објектом **Dokument** морају бити задовољена – вредности параметара морају бити различити

Постуслови: Повезује жељена документа (требовање и пријемницу, пријемницу и отпремницу,...)

Уговор УГ18: **MakeReklamacija** (Dokument): Signal;

Веза са СК: СК4

Предуслови: Вредносна и структурна ограничења над објектом **Dokument** и **VrstaDokumenta** морају бити задовољена – позитивна вредност улазног параметра

Постуслови: Враћа креирану рекламацију за отпремницу и пријемницу које немају исте количине на истим књигама

Уговор УГ19: **JednaStrana** (Dokument): Signal;

Веза са СК: СК4

Предуслови: Вредносна и структурна ограничења над објектом **Dokument**, **VrstaDokumenta** и **Objekat** морају бити задовољена – позитивна вредност улазног параметра

Постуслови: Синхронизовано стање на документу на основу кога је креирана рекламација

Уговор УГ20: **UpdateStatus** (Dokument): Signal;

Веза са СК: СК2, СК3, СК4, СК5, СК6

Предуслови: Вредносна и структурна ограничења над објектом **Dokument** морају бити задовољена – позитивна вредност улазног параметра

Постуслови: Ажуриран статус избраног документа

Уговор УГ21: **VratiPodatkeOKnjizi** (Knjiga): Signal;

Веза са СК: СК1

Предуслови: /

Постуслови: /

Уговор УГ22: **AzurirajPodatkeOKnjizi** (Knjiga): Signal;

Веза са СК: СК1

Предуслови: /

Постуслови: /

Уговор УГ23: **VratiViewButtons** (VrstaDokumenta): Signal;

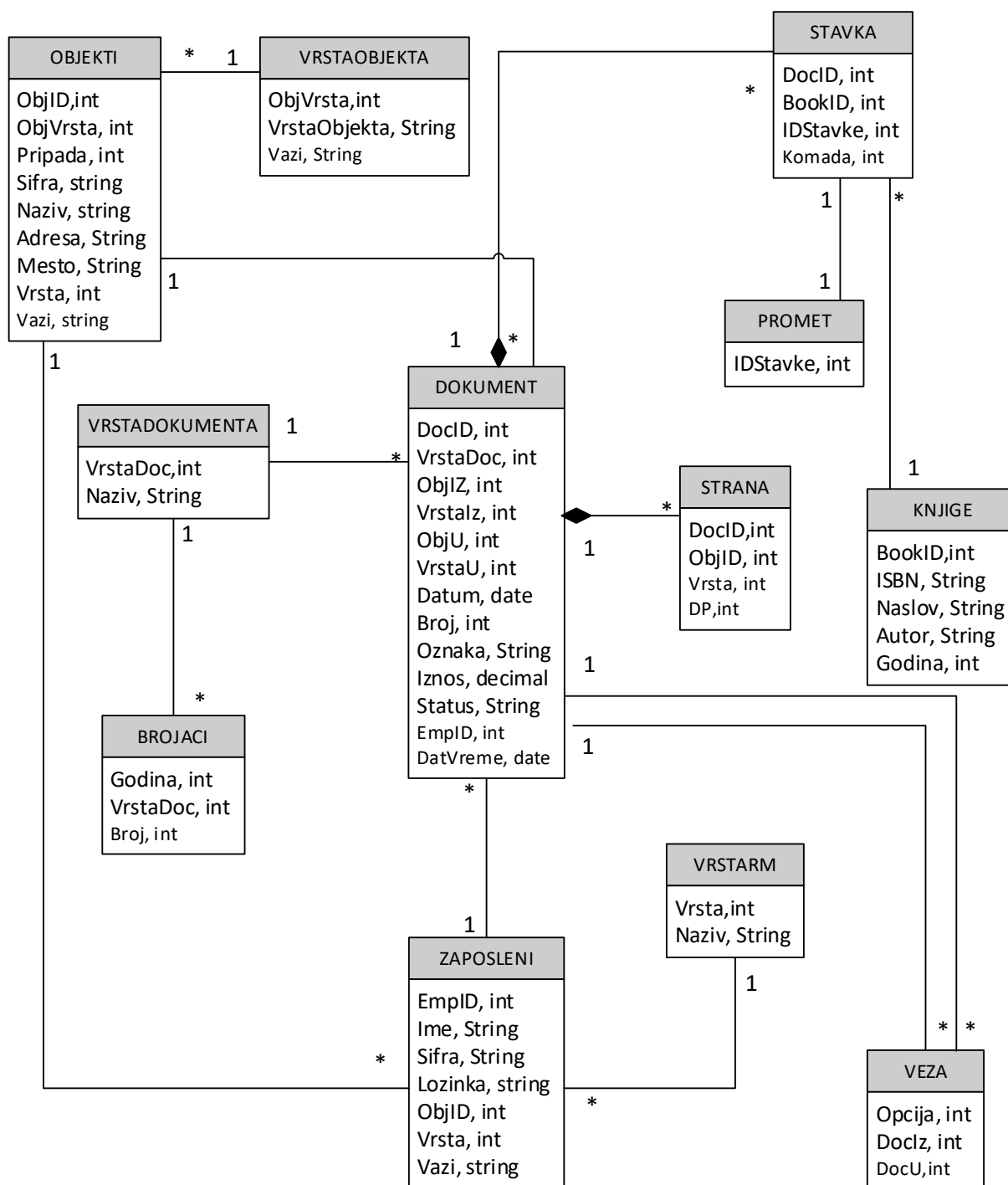
Веза са СК: сви

Предуслови: /

Постуслови: /

3.3.3 Структура софтверског система – Концептуални (доменски) модел

Концептуални модел даје приказ софтверског система кроз опис концептуалних класа у доменском моделу. Треба да буде независан од софтвера или структуре складиштења. У моделу су приказани ентитети, атрибути, и међусобне релације између ентитета.



Слика 18 Концептуални модел

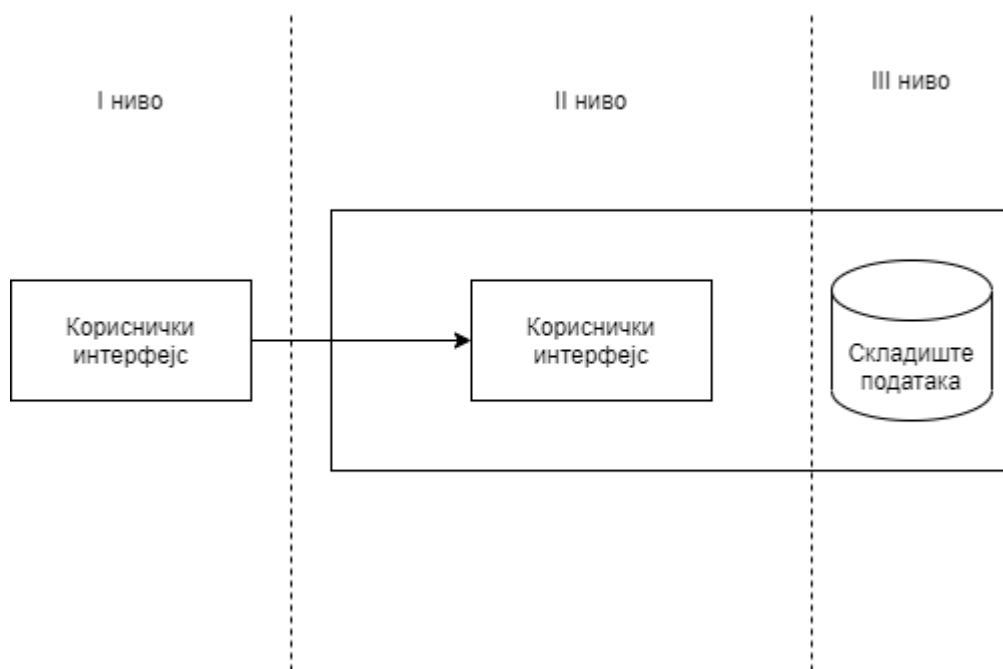
3.4 Пројектовање

Фаза пројектовања даје опис физичке структуре и понашања софтверског система (архитектуру софтверског система). Пројектовање архитектуре софтверског система обухвата пројектовање корисничког интерфејса, апликационе логике и складишта података.

Архитектура која се најчешће користи је тронивојска архитектура и код ње се софтверски систем састоји се из три слоја. У првом се налази кориснички интерфејс и он представља презентациони слој. Он искључиво врши функцију приказа података крајњим корисницима и уноса података. Не сме да садржи пословну логику. Треба да врши позиве искључиво ка логичком нивоу и не сме да приступа физичком нивоу директно.

Логички ниво садржи искључиво пословну логику (доменски објекти, системске операције) и може само да позива физички ниво.

Трећи слој је физички и састоји се од уређља и софтвера за физичко складиштење података. У њему такође не сме да буде део који садржи пословну логику ниди део чија би улога била приказ података



Слика 19 Тронивојска архитектура

3.4.1 Пројектовање складишта података – Релациони модел

Dokument (docid, vrstadoc, datum, broj, oznaka, iznos, status, datvreme, *empid*, *objiz*, *vrstaiz*, *obju*, *vrstau*)

Knjige (bookid, isbn, naslov, autor, godina, cena)

Objekti (objid, sifra, naziv, adresa, mesto, vrsta, *objvrsta*, *pripada*)

Stavka (idstavke, *objid*, *bookid*)

Promet (idstavke, komada)

Strana (dp, *docid*, *objid*, *vrsta*)

VrstaDokumenta (vrstadoc, dokument)

VrstaObjekta (*objvrsta*, vrstaobjekta)

VrstaRM (*vrsta*, naziv)

Zaposleni (empid, ime, sifra, lozinka, *vrsta*, *objid*)

Brojaci (vrstadoc, godina, broj)

Veza(opcija, *dociz*, *docu*)

Tabela Dokument		Prosto vrednosno ograničenje		Složeno vrednosno ograničenje		Strukturno ograničenje
Atributi	Ime	Tip atributa	Vrednost atributa	Međuzavisnost atributa jedne tabele	Međuzavisnost atributa više tabele	INSERT RESTRICTED VrstaDokumenta Objekti Zaposleni
	docid	Int, PK ¹	Identity			UPDATE RESTRICTED VrstaDokumenta Objekti Zaposleni
	vrstadoc	int	Not null	VrstaDoc, FK		
	objiz	int	Not null	Objekti, FK		
	vrstaiz	int	Not null	VrstaRM, FK		
	obju	int	Not null	Objekti, FK		
	vrstau	int	Not null	VrstaRM, FK		UPDATE CASCADE Stavke Veza Strana
	datum	date	Not null			
	broj	int	Not null			
	oznaka	varchar(20)	Not null			
	iznos	decimal(18, 2)	Not null			CASCADE DELETE Stavke Veza Strana
	status	char(1)	Not null			
	empid	int	Not null			
	datvreme	datetime	Not null			

Табела 2 Документ

¹ PK – Primary Key, FK – Foreign Key

Tabela Knjige		Prosto vrednosno ograničenje		Složeno vrednosno ograničenje		Strukturno ograničenje
Atributi	Ime	Tip atributa	Vrednost atributa	Međuzavisnost atributa jedne tabele	Međuzavisnost atributa više tabele	INSERT RESTRICTED Stavke
	bookid	Int PK	Identity			UPDATE RESTRICTED Stavke
	isbn	varchar(50)	Not null			
	naslov	nvarchar(75)	Not null			
	autor	nvarchar(50)	Not null			
	godina	int	Not null			
	cena	decimal(18, 2)	Not null			DELETE /

Табела 3 Књиге

Tabela Objekti		Prosto vrednosno ograničenje		Složeno vrednosno ograničenje		Strukturno ograničenje
Atributi	Ime	Tip atributa	Vrednost atributa	Međuzavisnost atributa jedne tabele	Međuzavisnost atributa više tabele	INSERT RESTRICTED VrstaObjekta
	objid	Int, PK	Identity			UPDATE RESTRICTED VrstaObjekta
	objvrsta	int	Not null	VrstaObjekta, FK		
	pripada	int	Not null	Objekti, FK		
	sifra	varchar(10)	Not null			
	naziv	nvarchar(50)	Not null			UPDATE CASCADES Dokumenta, Zaposleni
	adresa	nvarchar(50)	Not null			
	mesto	nvarchar(50)	Not null			
	vrsta	int	Not null	VrstaRM, FK		DELETE RESTRICTED Dokumenta, Zaposleni
	vazi	char(1)	Not null			

Табела 4 Објекти

Tabela Stavka		Prosto vrednosno ograničenje		Složeno vrednosno ograničenje		Strukturno ograničenje
Atributi	Ime	Tip atributa	Vrednost atributa	Međuzavisnost atributa jedne tabele	Međuzavisnost atributa više tabele	INSERT RESTRICTED Dokument, Knjiga
	docid	Int, PK	Not null	Dokument, FK		UPDATE RESTRICTED Dokument, Knjiga
	bookid	Int, PK	Not null	Knjige, FK		
	idstavke	Int,PK	Not null			
						UPDATE CASCADES Promet
						DELET CASCADES Promet

Табела 5 Ставке

Tabela Promet		Prosto vrednosno ograničenje		Složeno vrednosno ograničenje		Strukturno ograničenje
Atributi	Ime	Tip atributa	Vrednost atributa	Međuzavisnost atributa jedne tabele	Međuzavisnost atributa više tabele	INSERT RESTRICTED Stavke
	idstavke	Int, PK	Not null	Stavke, FK		
	komada	Int	Not null			UPDATE RESTRICTED Stavke

Табела 6 Промет

Tabela Strana		Prosto vrednosno ograničenje		Složeno vrednosno ograničenje		Strukturno ograničenje
Atributi	Ime	Tip atributa	Vrednost atributa	Međuzavisnost atributa jedne tabele	Međuzavisnost atributa više tabele	INSERT RESTRICTED Dokument
	docid	Int, PK	Not null	Dokument, FK		
	objid	Int, PK	Not null	Objekti, FK		UPDATE RESTRICTED Dokument
	vrsta	Int, PK	Not null	VrstaRM, FK		
	dp	Int, PK	Not null			
						DELETE /

Табела 7 Страна

Tabela VrstaDokumenta		Prosto vrednosno ograničenje		Složeno vrednosno ograničenje		Strukturno ograničenje
Atributi	Ime	Tip atributa	Vrednost atributa	Međuzavisnost atributa jedne tabele	Međuzavisnost atributa više tabele	UPDATE CASCADE Dokument, Brojaci
	vrstadoc	Int, PK	Identity			
	dokument	nvarchar(50)	Not null			
	vazi	char(1)	Not null			DELETE RESTRICTED Dokument, Brojaci

Табела 8 ВрстаДоц

Tabela VrstaObjekta		Prosto vrednosno ograničenje		Složeno vrednosno ograničenje		Strukturno ograničenje
Atributi	Ime	Tip atributa	Vrednost atributa	Međuzavisnost atributa jedne tabele	Međuzavisnost atributa više tabele	UPDATE CASCADE Objekti
	objvrsta	Int, PK	Identity			
	vrstaobjekta	nvarchar(50)	Not null			RESTRICTED DELETE Objekti
	vazi	char(1)	Not null			

Табела 9 ВрстаОбјекта

Tabela VrstaRM		Prosto vrednosno ograničenje		Složeno vrednosno ograničenje		Strukturno ograničenje
Atributi	Ime	Tip atributa	Vrednost atributa	Međuzavisnost atributa jedne tabele	Međuzavisnost atributa više tabele	UPDATE CASCADE Zaposleni
	vrsta	Int, PK	Identity			
	naziv	nvarchar(50)	Not null			RESTRICTED DELETE Zaposleni

Табела 10 ВрстаРМ

Tabela Zaposleni		Prosto vrednosno ograničenje		Složeno vrednosno ograničenje		Strukturno ograničenje
Atributi	Ime	Tip atributa	Vrednost atributa	Međuzavisnost atributa jedne tabele	Međuzavisnost atributa više tabele	INSERT RESTRICTED VrstaRM, Objekti
	empid	Int, PK	Identity			
	ime	nvarchar(50)	Not null			
	sifra	varchar(50)	Not null			UPDATE RESTRICTED VrstaRM, Objekti
	lozinka	varchar(20)	Not null			
	objid	int	Not null	Objekti,FK		
	vrsta	int	Not null	VrstaRM,FK		UPDATE CASCADDES Dokument
	vazi	char(1)	Not null			
						DELETE RESTRICTED Dokument

Табела 11 Запослени

Tabela Brojaci		Prosto vrednosno ograničenje		Složeno vrednosno ograničenje		Strukturno ograničenje
Atributi	Ime	Tip atributa	Vrednost atributa	Međuzavisnost atributa jedne tabele	Međuzavisnost atributa više tabele	INSERT RESTRICTED VrstaDoc
	vrstadoc	Int, PK	Not null	VrstaDoc, FK		
	godina	Int, PK	Not null			
	broj	int	Not null			UPDATE RESTRICTED VrstaDoc
						DELETE /

Табела 12 Бројачи

Tabela Veza		Prosto vrednosno ograničenje		Složeno vrednosno ograničenje		Strukturno ograničenje
Atributi	Ime	Tip atributa	Vrednost atributa	Međuzavisnost atributa jedne tabele	Međuzavisnost atributa više tabele	INSERT RESTRICTED Dokument
	opcija	Int, PK	Not null			UPDATE RESTRICTED Dokument
	dociz	Int, PK	Not null	Dokument, FK		
	docu	Int, PK	Not null	Dokument, FK		
						DELETE /

Табела 13 Веза

Модел базе података је креиран на *MS SQL* платформи. Приликом дизајнирања складишта података, пројекат је заснован на следећим основама:

Примарни кључеви у шифарским табелама (Књиге, Објекти, ВрстаОбјекта, ВрстаРМ, Запослени, Документ)

су подаци типа *int*, а вредности *Identity*, тј. база података је задужена за доделу њихових вредности приликом употребе INSERT-а команде.

Подаци у овим пољима се не могу ажурирати (не може се применити промена вредности примарног кључа), тако да се CASCADE UPDATE за везане табеле, преко FOREIGN KEY-а искључује.

Ова особина обезбеђује, да се једном обрисани податак у табели, не може поново регистровати под истом вредношћу, већ се увек креира само нови слог.

Приликом брисања слога из шифарских табела, сви подаци у везаним табелама (преко FOREIGN KEY-а) се бришу.

Брисање података из прометних табела, се неће радити, због извештаја за Финансијску службу и Управу, већ се "брисање" података реализује на начин, да подаци "нису видљиви" - у ову сврху је обезбеђен индикатор видљивати, поље Вази, које има 2 вредности - 1 - подаци су доступни, и 0 подаци су сакривени.

На овај начин се обезбеђује да се Објекти који престану са употребом, и даље постоје у финансијским извештајима, али да се Промет више нне може реализовати преко њих, док се поново не активирају.

3.4.2 Пројектовање корисничког интерфејса

Кориснички интерфејс је дефинисан преко скупа екранских форми. Сценарија коришћења екранских форми су директно повезана са сценаријима случајева коришћења. Екранска форма има улогу да прихвати податке које уноси актор, прихвата догађаје које прави актор, позива контролера корисничког интерфејса како би му проследио те податке и приказује податке добијене од контролера корисничког интерфејса.

Контролер корисничког интерфејса је одговоран да:

1. прихвата податке које шаље екранска форма;
2. конвертује податке у објекат који представља улазни аргумент који прихвата СО;
3. шаље захтев за извршење системске операције;
4. прихвата објекат који настаје као резултат извршења системске операције;
5. конвертује објекат у податке графичких елемената.

СК1: Случај коришћења – Креирање требовања складишту

Назив СК

Креирање требовања складишту

Актори СК

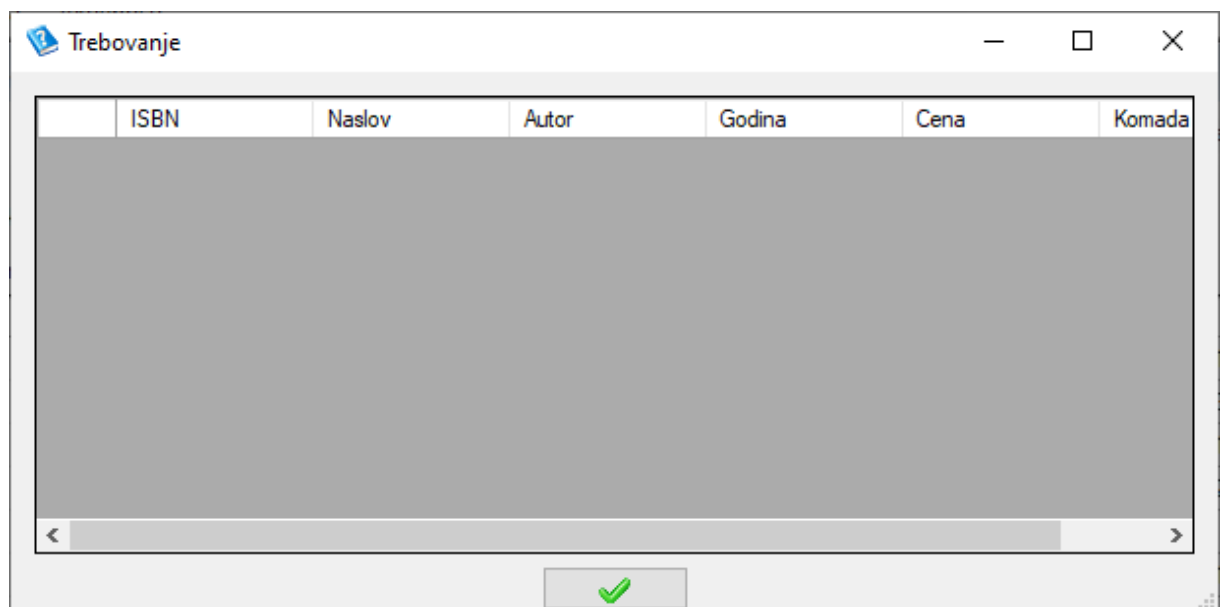
Запослени у књижари

Учесници СК

Запослени у књижари и систем (програм)

Предуслов: Систем је укључен и запослени у књижари је улогован под својом шифром. Запослени у књижари бира опцију „Требовање“ са главне форме. Систем приказује форму за унос требовања. Учитана је листа књига.

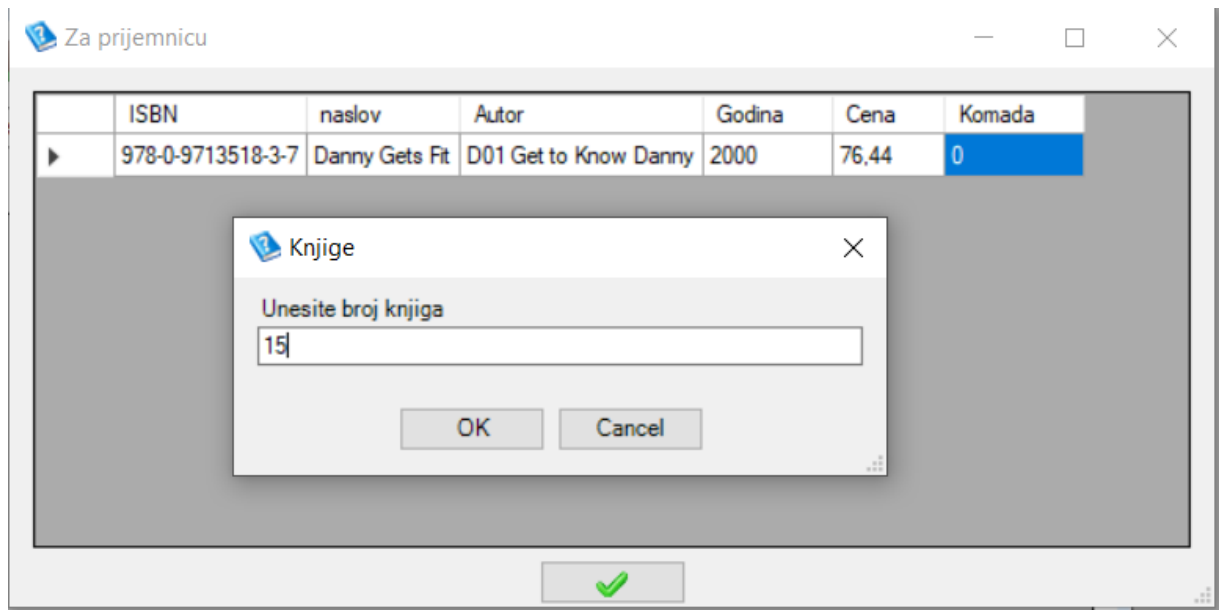
Основни сценарио СК



ISBN	Naslov	Autor	Godina	Cena	Komada
------	--------	-------	--------	------	--------

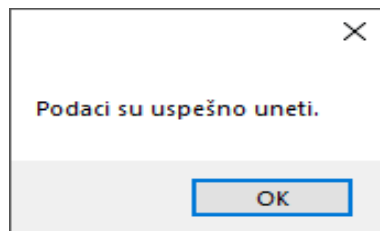
Слика 20 Форма за унос требовања

1. **Запослени у књижари** додаје књиге и жељене количине (АПУСО)



Слика 21 Бирање књиге и потребне количине

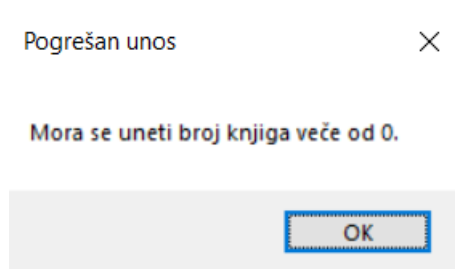
2. **Запослени у књижари** контролише да ли је коректно унео податке за тражено **требовање**. (АНСО)
3. **Запослени у књижари** позива **систем** да запамти податке о датом **требовању**. (АПСО)
4. **Систем памти** податке о **изабраним књигама**. (СО)
5. **Систем** аутоматски **креира пријемницу** за складиште и **приказује** **запосленом у књижари** поруку: „Подаци су успешно унети“. (ИА)



Слика 22 Успешан унос требовања

Алтернативни сценарио

- 5.1 Уколико **систем** није у могућности да креира требовање због неправилног уноса, он **приказује** **запосленом у књижари** поруку: „**Мора се унети број књига веће од 0**“. (ИА)



Слика 23 Неуспешан унос требовања

СК2: Случај коришћења – Креирање пријемнице добављачу

Назив СК

Креирање пријемнице добављачу

Актори СК

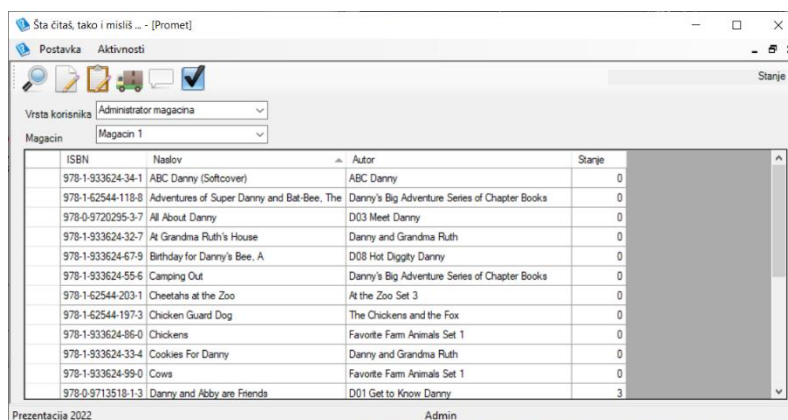
Администратор складишта

Учесници СК

Администратор складишта и систем (програм)

Предуслов: Систем је укључен и администратор складишта је улоган под својом шифром. Администратор складишта бира опцију „Да ли има захтева из књижара (Требовање)“ са главне форме. Систем враћа списак свих постојећих требовања која су извршена према датом складишту.

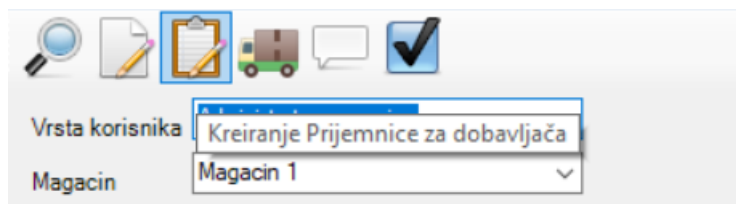
Основни сценарио СК



ISBN	Naslov	Autor	Stanje
978-1-933624-34-1	ABC Danny (Softcover)	ABC Danny	0
978-1-62544-118-8	Adventures of Super Danny and Bat-Bee, The	Danny's Big Adventure Series of Chapter Books	0
978-0-9720295-3-7	All About Danny	D03 Meet Danny	0
978-1-933624-32-7	At Grandma Ruth's House	Danny and Grandma Ruth	0
978-1-933624-67-9	Birthday for Danny's Bee, A	D08 Hot Diggy Danny	0
978-1-933624-55-6	Camping Out	Danny's Big Adventure Series of Chapter Books	0
978-1-62544-203-1	Cheetahs at the Zoo	At the Zoo Set 3	0
978-1-62544-197-3	Chicken Guard Dog	The Chickens and the Fox	0
978-1-933624-86-0	Chickens	Favorite Farm Animals Set 1	0
978-1-933624-33-4	Cookies For Danny	Danny and Grandma Ruth	0
978-1-933624-99-0	Cows	Favorite Farm Animals Set 1	0
978-0-9713518-1-3	Danny and Abby are Friends	D01 Get to Know Danny	3

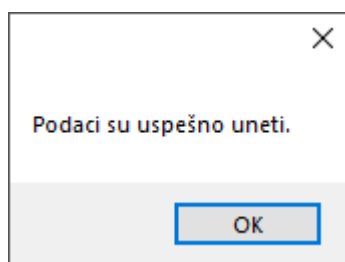
Слика 24 Форма са списком свих постојећих требовања

1. Администратор складишта бира требовање за које жели да креира пријемницу”. (АПУСО)



Слика 25 Опција Креирање Пријемнице за добављача

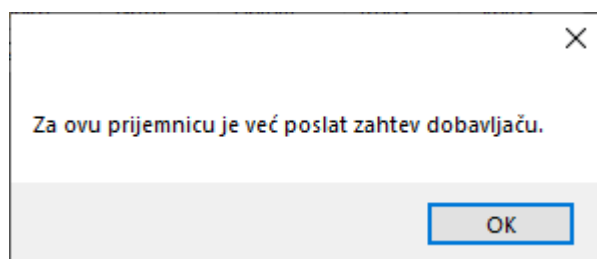
2. **Администратор складишта** **позива** **систем** да запамти податке о **пријемници** (АПСО)
3. **Систем** **памти** податке о **креираној пријемници**. (СО)
4. **Систем** аутоматски **креира** **пријемницу** за добављача и **приказује** **администратору складишта** поруку: „Подаци су успешно унети“. (ИА)



Слика 26 Успешно креирање пријемнице

Алтернативни сценарио

- 4.1 Уколико је **пријемница** већ креирана, **Систем** **приказује** **администратору складишта** поруку: „За ову **пријемницу** је већ послат захтев добављачу“. (ИА)



Слика 27 Неуспешно креирање пријемнице

СКЗ: Случај коришћења – Креирање отпремнице за складиште

Назив СК

Креирање отпремнице

Актери СК

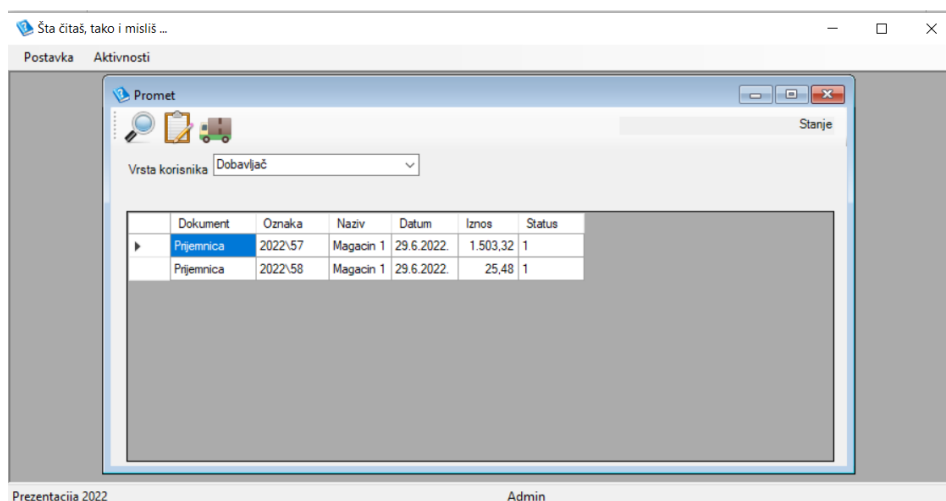
Добављач

Учесници СК

Добављач и систем (програм)

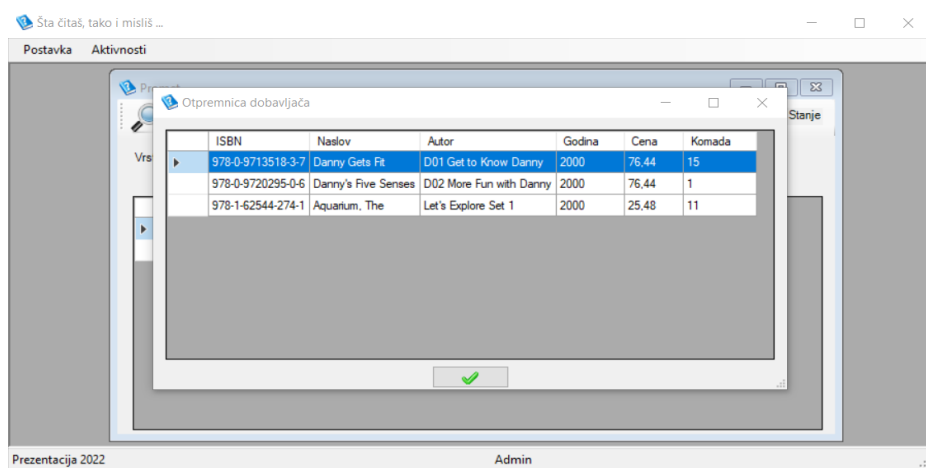
Предуслов: Систем је укључен и Добављач је улогован под својом шифром. Добављач бира опцију „Да ли има захтева за испоруком књига (пријемница)“. Систем враћа списак свих постојећих пријемница која су послате добављачу.

Основни сценарио СК



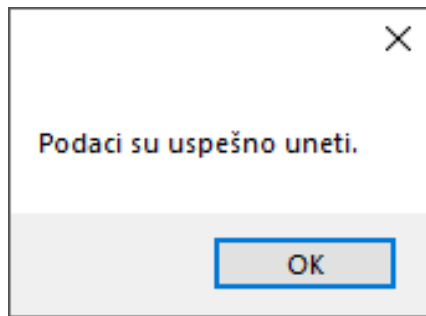
Слика 28 Списак свих пријемница послатих добављачу

1. Добављач бира жељену пријемницу за коју жели да креира отпремницу за складиште. (АПУСО)



Слика 29 Креирање отпремнице за складиште

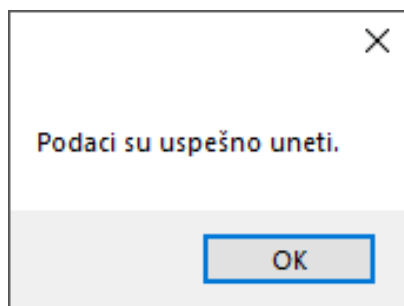
2. Добављач контролише да ли је коректно унео податке за отпремницу. (АНСО)
3. Добављач позива систем да запамти податке о креираној отпремници. (АПСО)
4. Систем памти податке о креираној отпремници. (СО)
5. Систем приказује добављачу поруку „Подаци су успешно унети“. (ИА)



Слика 30 Успешно креирање отпремнице

Алтернативни сценарио

5.1 У случају да је **добављач** променио потребну количинну књига за **отпремницу** **систем** приказује **добављачу** поруку „Подаци су успешно унети“. (ИА)



Слика 31 Успешно мењање отпремнице

СК4: Случај коришћења – Креирање рекламације на основу комисијског записника

Назив СК

Креирање рекламације

Актери СК

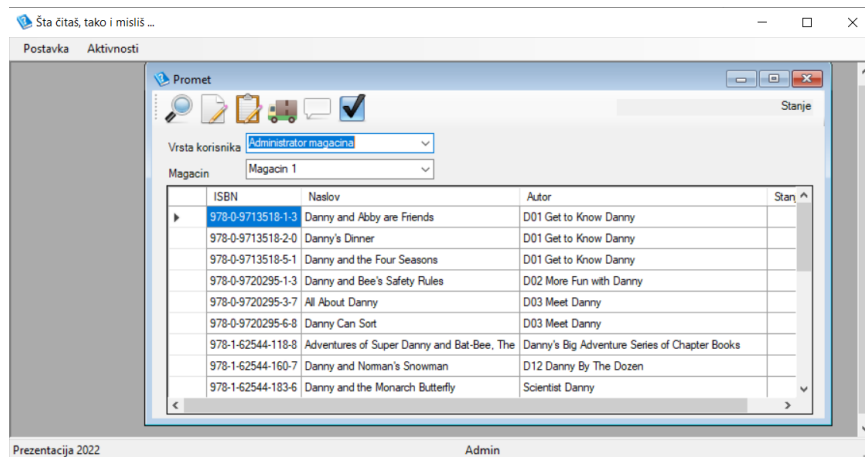
Администратор складишта

Учесници СК

Администратор складишта и систем (програм)

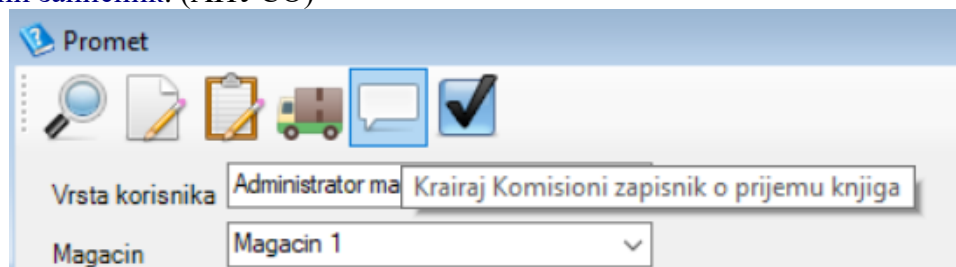
Предуслов: Систем је укључен и администратор складишта је улогован под својом шифром. Администратор складишта бира опцију „Креирање Пријемнице за добављача“. Систем враћа списак свих постојећих требовања која су извршена према датом складишту.

Основни сценарио СК



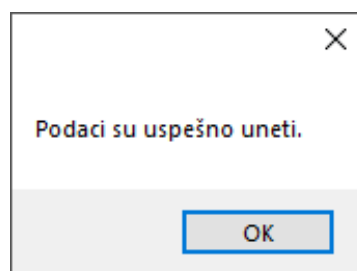
Слика 32 Списак постојећих требовања према датом складишту

1. **Администратор складишта бира** отпремницу коју жели **комисијски** да прими и за њу креира **комисиони записник**. (АПУСО)



Слика 33 Опција Креирај Комисиони записник о пријему књига

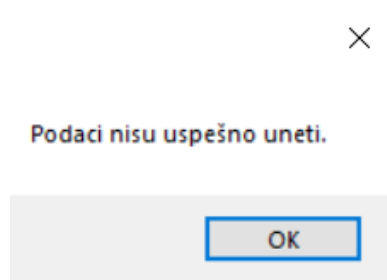
2. **Администратор складишта контролише** да ли је коректно унео податке за **комисиони записник**. (АНСО)
3. **Администратор складишта позива систем** да запамти податке о изабраним **књигама**. (АПСО)
4. **Систем памти** податке о изабраним **књигама**. (СО)
5. **Систем приказује администратору складишта** поруку: „Подаци су успешно унети“. (ИА)



Слика 34 Успешно креирање записника

Алтернативни сценарио 1

Уколико подаци о **књигама** нису успешно унети, **систем приказује администратору складишта** поруку: „Подаци нису успешно унети“. (ИА)



Слика 35 Неуспешно креирање записника

СК5: Случај коришћења – Креирање доставнице на основу отпремнице добављача

Назив СК

Креирање доставнице

Актори СК

Администратор складишта

Учесници СК

Администратор складишта и систем (програм)

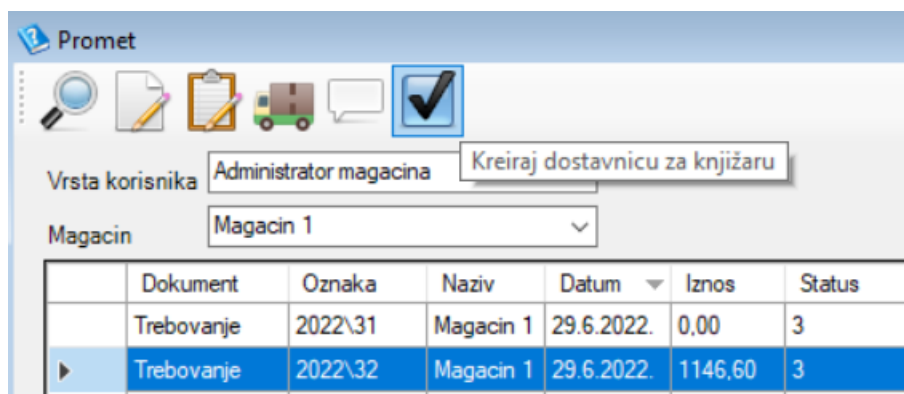
Предуслов: Систем је укључен и администратор складишта је улоган под својом шифром. Администратор складишта бира опцију „Да ли има захтева из књиžара (требовања)“. Систем враћа списак свих постојећих требовања која су упућена према датом складишту.

Основни сценарио СК

Dokument	Oznaka	Naziv	Datum	Iznos	Status
Trebovanje	2022\31	Magacin 1	29.6.2022.	0.00	3
Trebovanje	2022\32	Magacin 1	29.6.2022.	1146.60	3
Trebovanje	2022\33	Magacin 1	29.6.2022.	280.28	3
Trebovanje	2022\34	Magacin 1	29.6.2022.	25.48	3
Trebovanje	2022\20	Magacin 1	23.6.2022.	229.32	3
Trebovanje	2022\21	Magacin 1	23.6.2022.	0.00	3
Trebovanje	2022\19	Magacin 1	22.6.2022.	76.44	3
Trebovanje	2022\14	Magacin 1	21.6.2022.	0.00	3
Trebovanje	2022\15	Magacin 1	21.6.2022.	127.40	3
Trebovanje	2022\16	Magacin 1	21.6.2022.	0.00	3

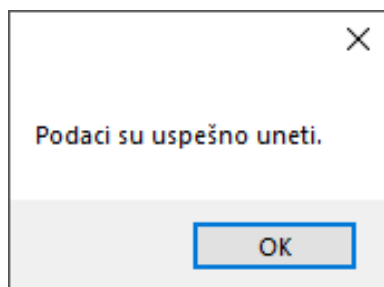
Слика 36 Списак свих требовања за дато складиште

1. **Администратор складишта бира** требовање за које жели да креира **доставницу** за књижару. (АПУСО)



Слика 37 Опција Креирај доставницу за књижару

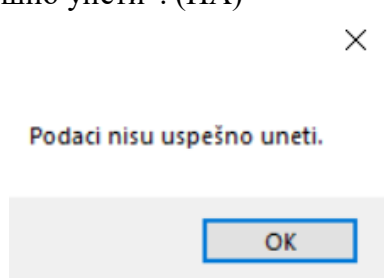
2. **Администратор складишта контролише** да ли је коректно унео податке за **доставницу**. (АНСО)
3. **Администратор складишта позива систем** да запамти податке о изабраној **доставници**. (АПСО)
4. **Систем памти** податке о **креираној доставници**. (СО)
5. **Систем аутоматски креира доставницу** за запосленог у књижари и **приказује** **администратору складишта** поруку: „Подаци су успешно унети“. (ИА)



Слика 38 Успешно креирање доставнице

Алтернативни сценарио

- 5.1 Уколико подаци нису успешно унети, **систем приказује** **администратору складишта** поруку: „Подаци нису успешно унети“. (ИА)



Слика 39 Неуспешно креирање доставнице

СК6: Случај коришћења – Пријем књига у књижару

Назив СК

Пријем **Књига**

Актори СК

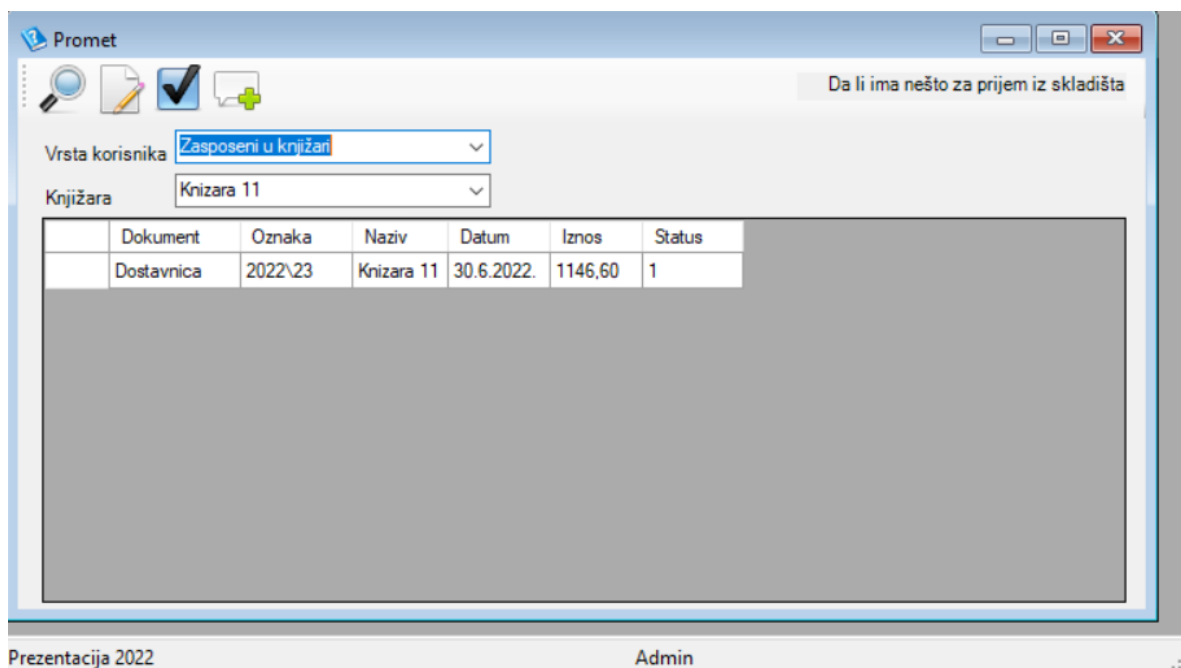
Запослени у књижари

Учесници СК

Запослени у књижари и **систем** (програм)

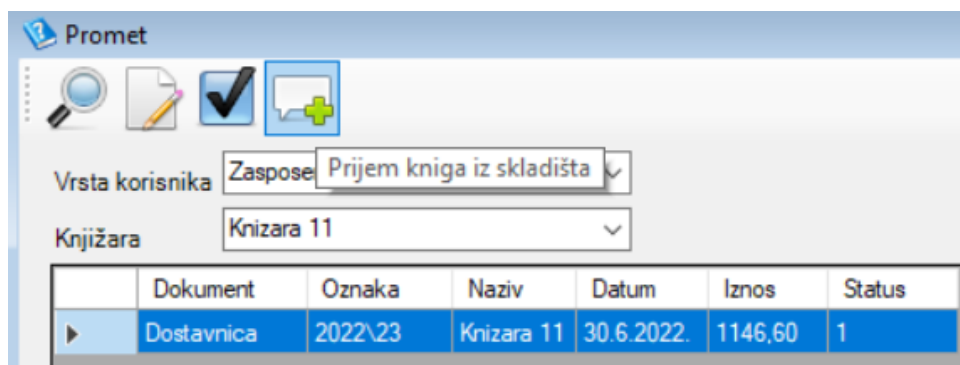
Предуслов: Систем је укључен и **Запослени у књижари** је улогован под својом шифром. **Запослени у књижари** бира опцију „Да ли има нешто за пријем из складишта“. Систем враћа списак доставница за које треба извршити пријем **књига**.

Основни сценарио СК



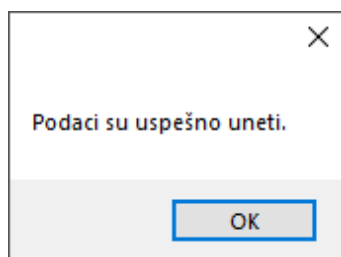
Слика 40 Списак доставница за које треба извршити пријем

1. **Запослени у књижари бира** доставницу за коју жели да прими **књиге** из складишта. (АПУСО)



Слика 41 Опција Пријем књига из складишта

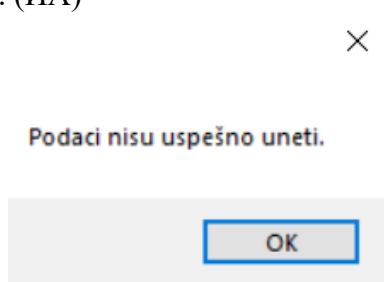
2. **Администратор складишта контролише** да ли је коректно унео податке за **књиге** које ће бити примљене. (АНСО)
3. **Запослени у књижари позива систем** да запамти податке о **књигама** које ће бити примљене у књижару. (АПСО)
4. **Систем памти** податке. (СО)
5. **Систем приказује запосленом у књижари** поруку „Подаци су успешно унети“. (ИА)



Слика 42 Успешан пријем књига

Алтернативни сценарио

- 5.1 Уколико подаци нису успешно унети, **систем приказује запосленом у књижари** поруку: „Подаци нису успешно унети“. (ИА)



Слика 43 Неуспешан пријем књига

3.4.3 Пројектовање апликационе логике

У оквиру пројектовања апликационе логике пројектују се пословна логика и брокер базе података. Брокер базе података је задужен за комуникацију између апликације и складишта података.

Операције које обезбеђују су:

Операција	Повратна вредност	Опис
Connect (ref string)	<i>bool</i>	Успоставља комуникацију између апликације и складишта података. У случају успеха враћа True, а у случају неуспеха враћа False и опис грешке у излазном параметру
Disconnect(ref string)	<i>bool</i>	Прекида комуникацију између апликације и складишта података. У случају успеха враћа True, а у случају неуспеха враћа False и опис грешке у излазном параметру
Execute(string,ref string)	<i>bool</i>	Извршава DDL команду над складиштем података
Execute(SQLClient.SQLCommand,ref string)	<i>bool</i>	Извршава Stored Procedure-у команду над складиштем података.
GetData(string,ref string)	<i>System.Data.DataTable</i>	Враћа податке у облику табеле, на основу SQL упита. У случају неуспеха, враћа null вредност и опис грешке у излазном параметру
GetData(SQLClient.SQLCommand,ref string)	<i>System.Data.DataTable</i>	Враћа податке у облику табеле, на основу Stored Procedure. У случају неуспеха, враћа null вредност и опис грешке у излазном параметру
GetValue(string,ref string)	<i>object</i>	Враћа вредност произвољног типа, а у случају грешке враћа null вредност и опис грешке у излазном параметру

Пројектовање понашања софтверског система – Системске операције

Пројектовање понашања софтверског система укључује пројектовање опште класе и конкретних класа које су одговорне за извршење системских операција.

Пре извршења системске операције проверава се предуслов уколико постоји и отвара се трансакција.

Пре извршења системске операције проверава се предуслов уколико постоји и отвара се трансакција.

За сваку системску операцију треба направити концептуална решења која су директно повезана са логиком проблема. Концептуалне релизације се могу описати преко објектног псеудокода, дијаграма сарадње, секвенцних дијаграма, дијаграма активности, дијаграма прелаза стања или дијаграма структуре.

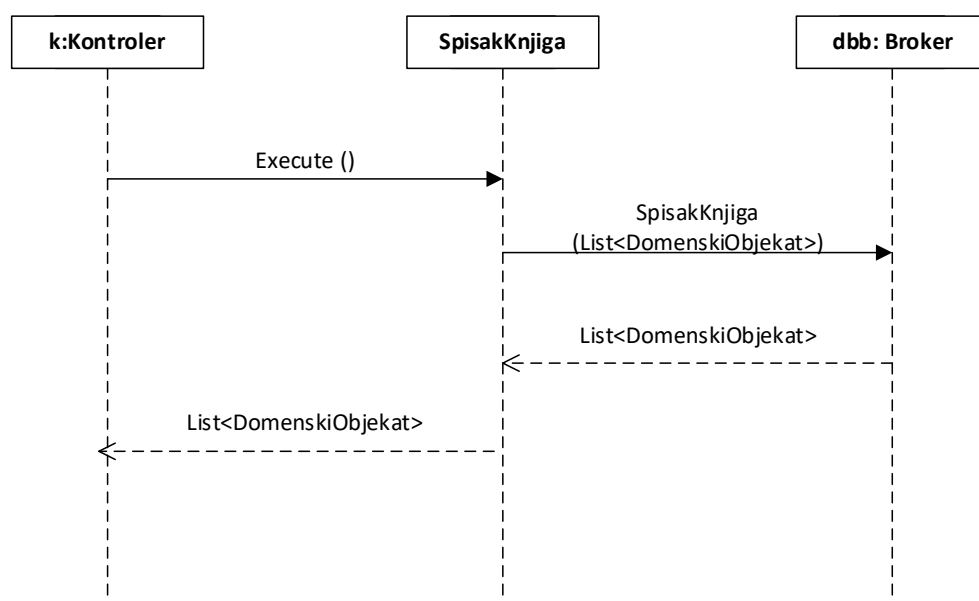
За сваки од уговора пројектује се концептуална реализација преко секвенцног дијаграма који приказује секвенцу порука у времену.

Уговор УГ1: **VratiSpisakKnjiga** (DataTable): Signal;

Веза са СК: СК1, СК2, СК3, СК4, СК5, СК6, СК7

Предуслови: /

Постуслови: /



Слика 44 ДС:Уговор - ВратиСписакКњига

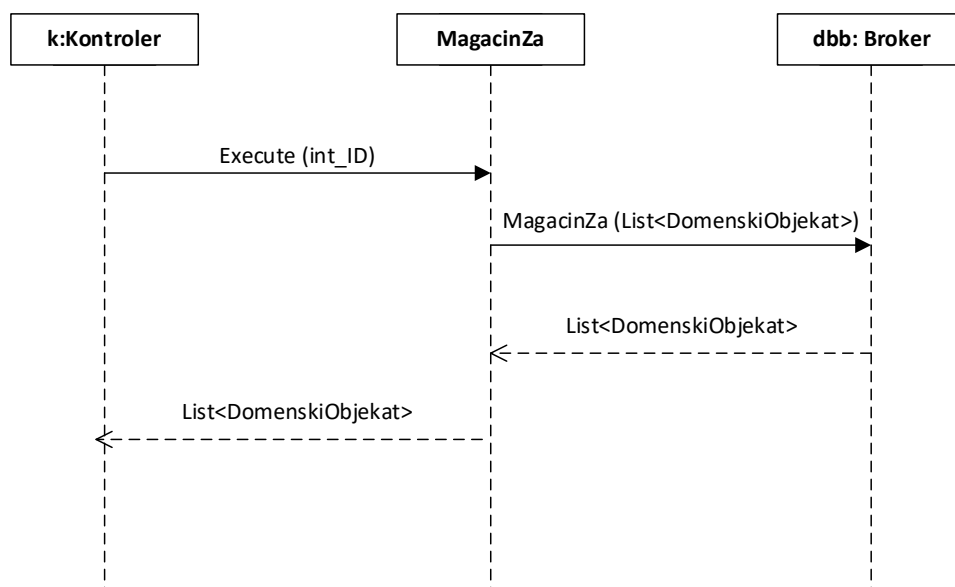
Уговор УГ2: **VratiMagacinZaKnjizaru**

(Magacin): Signal;

Веза са СК: СК1, СК2, СК3, СК4, СК5, СК6, СК7

Предуслови: /

Постуслови: /



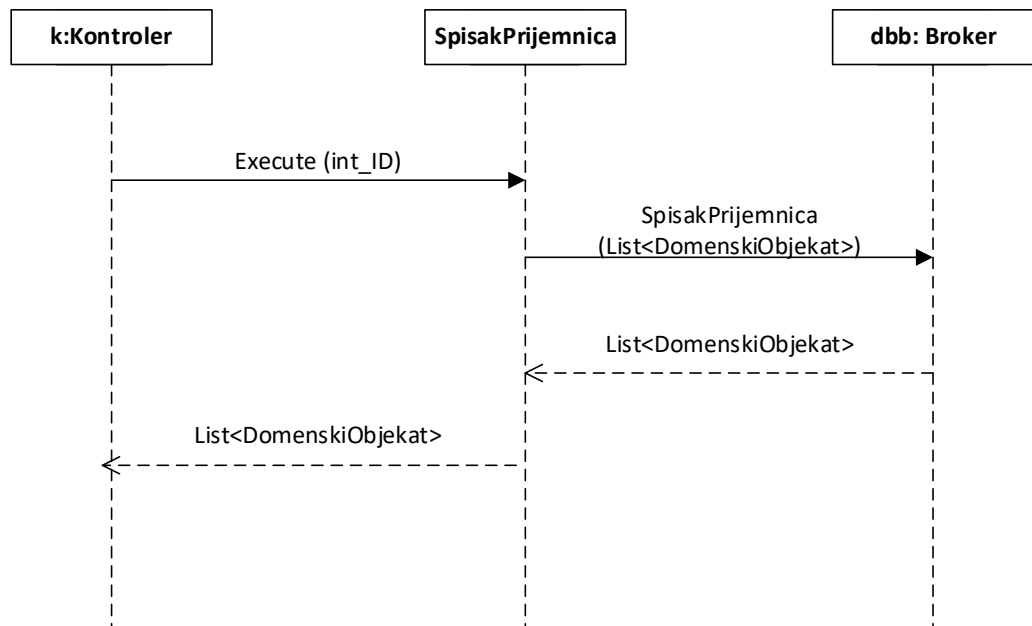
Слика 45 ДС:Уговор - ВратиМагацинЗаКњижару

Уговор УГ3: **VратиSpisakPrijemnica**: Signal;

Веза са СК: СК2

Предуслови: /

Постуслови: /



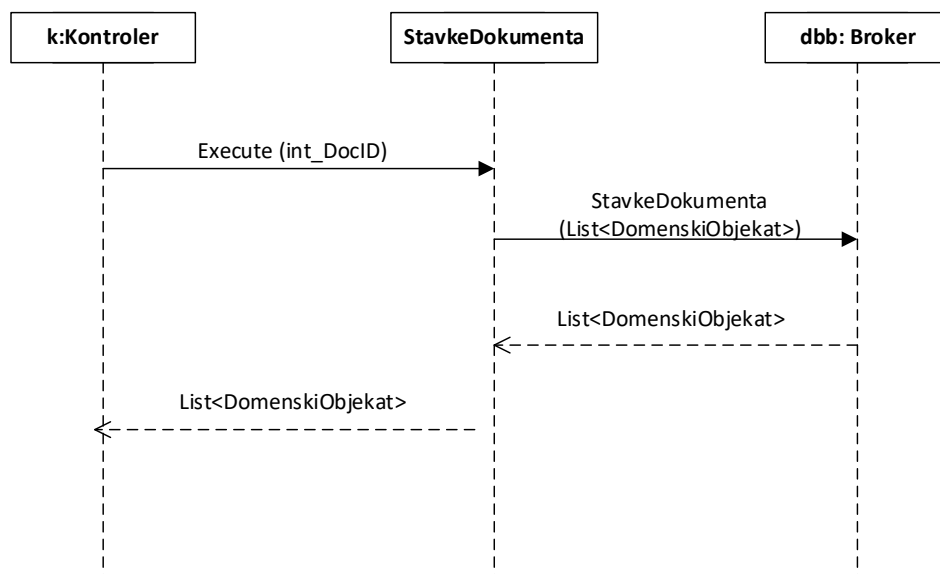
Слика 46 ДС:Уговор - ВратиСписакПријемница

Уговор УГ4: **VratiStavkeDokumenata** (Dokument): Signal;

Веза са СК: СК1, СК2, СК3, СК4, СК5, СК6, СК7

Предуслови: /

Постуслови: /



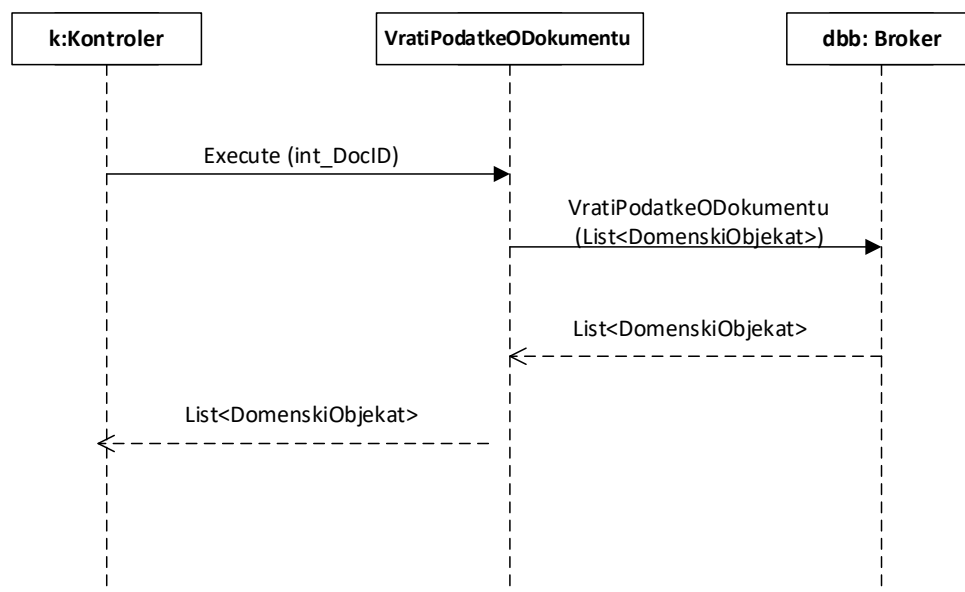
Слика 47 ДС:Уговор - ВратиСтавкеДокумента

Уговор УГ5: **VratiPodatkeODokumentu** (Dokument): Signal;

Веза са СК: СК1, СК2, СК3, СК4, СК5, СК6, СК7

Предуслови: /

Постуслови: /



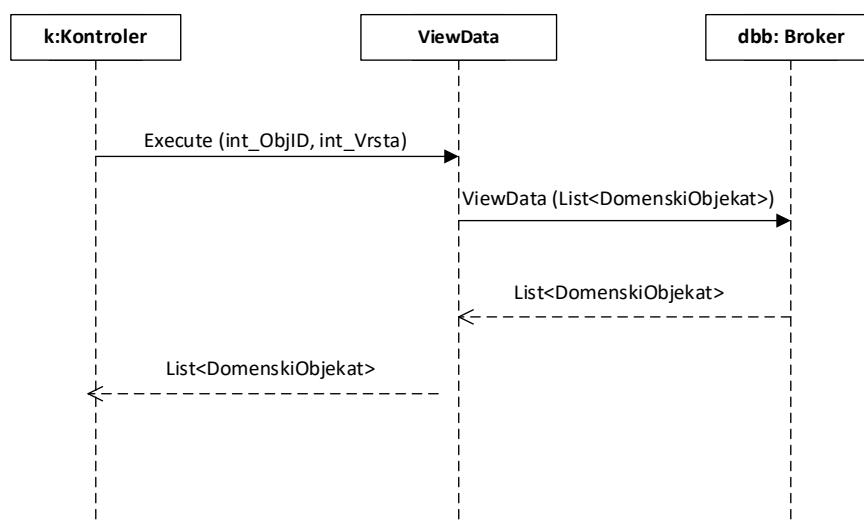
Слика 48 ДС:Уговор - ВратиПодаткеОДokumentу

Уговор УГ6: **VratiViewData** (DataTable): Signal;

Веза са СК: СК1, СК2, СК3, СК4, СК5, СК6, СК7

Предуслови: /

Постуслови: /



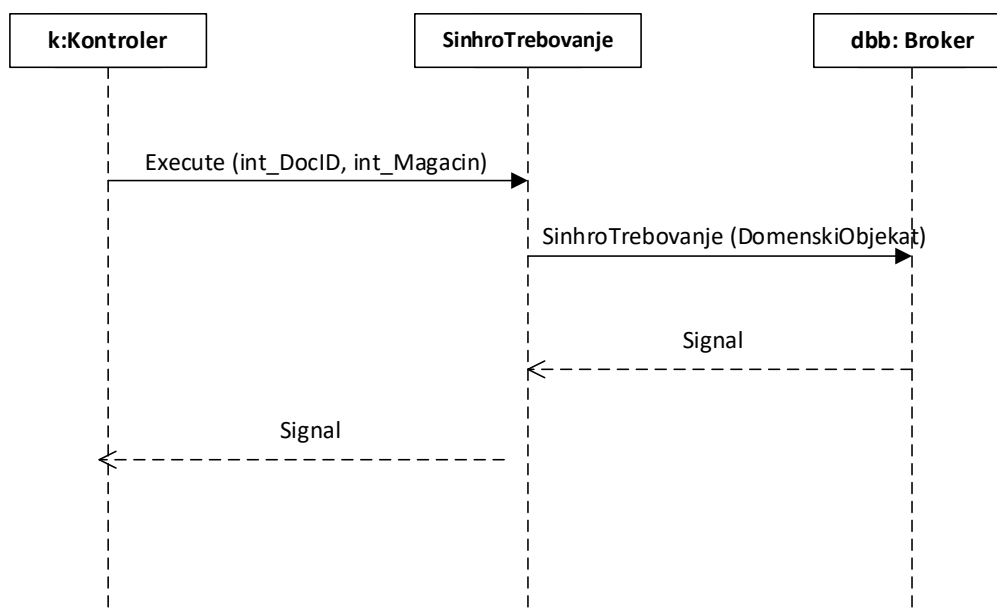
Слика 49 ДС:Уговор - ВратиВјудата

Уговор УГ7: **SinhroTrebovanje** (Dokument, Objekti): Signal;

Веза са СК: СК2

Предуслови: Вредносна и структурна ограничења над објектом **Dokument** и **Objekti** морају бити задовољена – позитивна вредност улазног параметра

Постуслови: Синхронизује статус требовања на основу пријемнице добављачу за дати магацин.



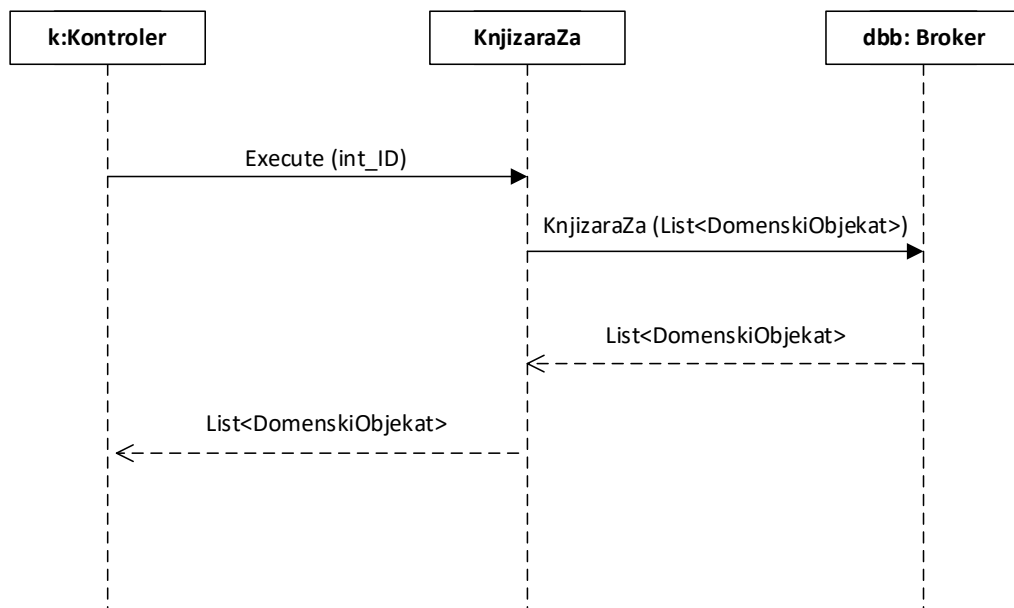
Слика 50 ДС:Уговор - СинхроТребовање

Уговор УГ8: **VratiKnjizareZaMagacin** (Objekti): Signal;

Веза са СК: СК1, СК2, СК3, СК4, СК5, СК6

Предуслови: /

Постуслови: /



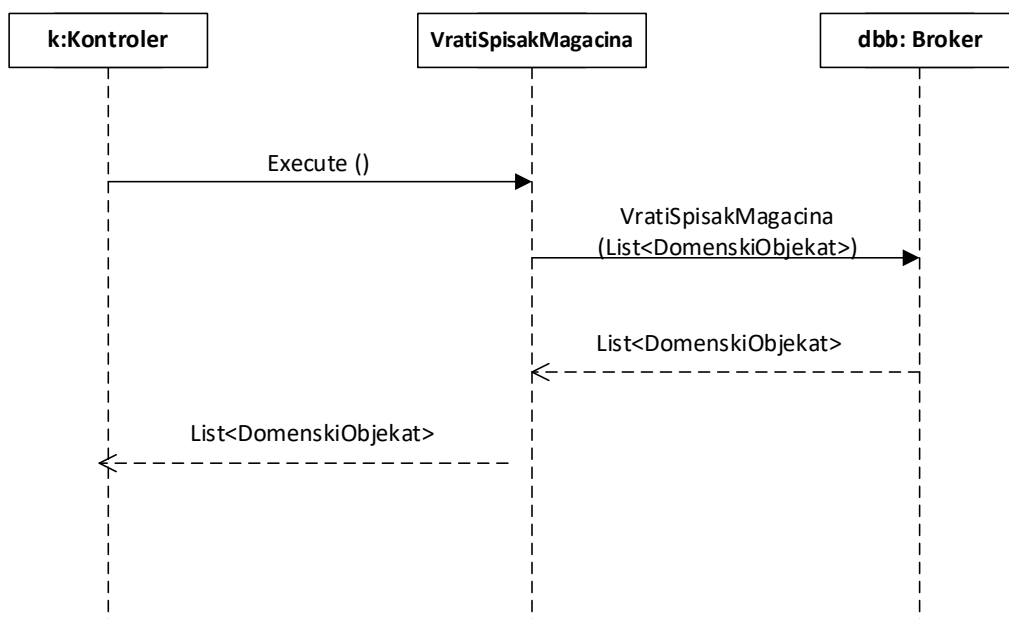
Слика 51 ДС:Уговор - ВратиКњижареЗаМагацин

Уговор УГ9: **VratiSpisakMagacina ()**: Signal;

Веза са СК: СК1, СК2, СК3, СК4, СК5, СК6, СК7

Предуслови: /

Постуслови: /



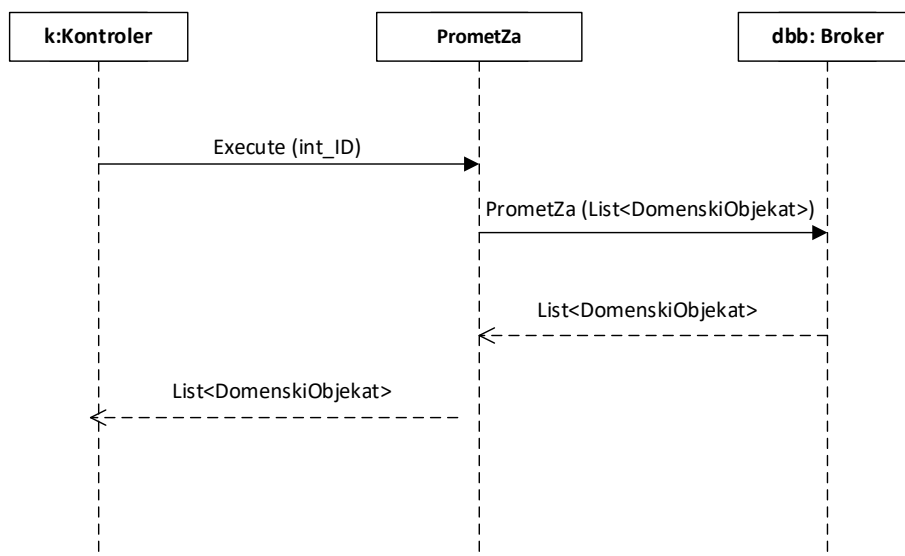
Слика 52 ДС:Уговор - ВратиСписакМагацина

Уговор УГ10: **VratiPrometZa** (Magacin): Signal;

Веза са СК: СК3

Предуслови: /

Постуслови: /



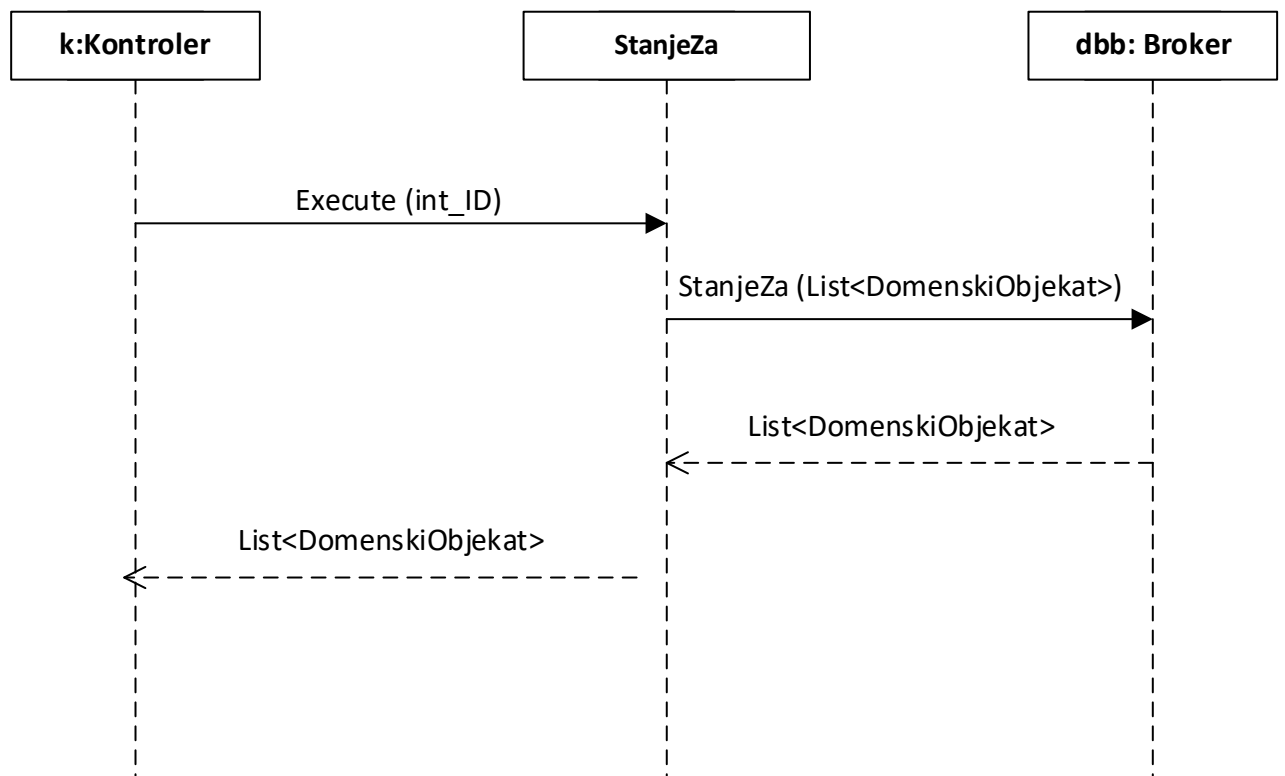
Слика 53 ДС:Уговор - ВратиПрометЗа

Уговор УГ11: **VratiStanjeZa** (Objekti): Signal;

Веза са СК: СК1, СК2, СК3, СК4, СК6, СК7

Предуслови: /

Постуслови: /



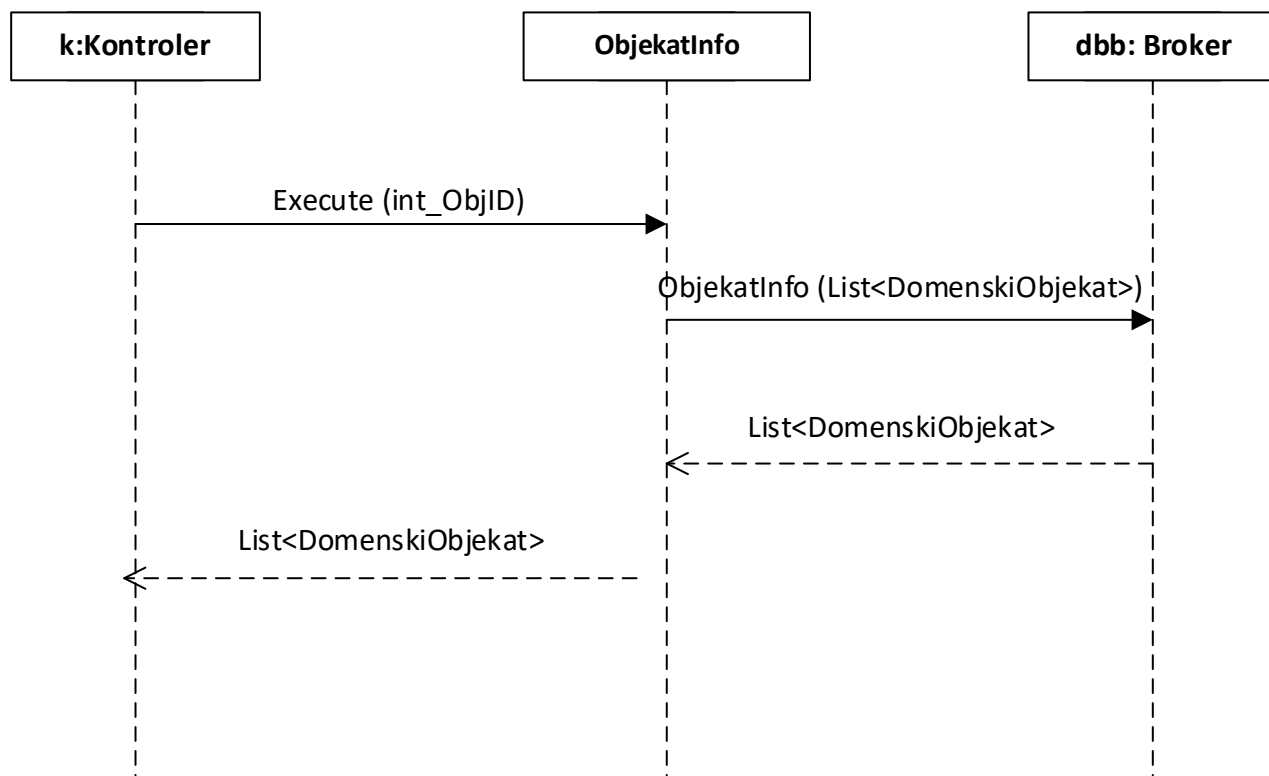
Слика 54 ДС:Уговор - ВратиСтањеЗа

Уговор УГ12: **VratiObjekatInfo** (Objekat): Signal;

Веза са СК: СК1, СК2, СК3, СК4, СК5, СК6, СК7

Предуслови: /

Постуслови: /



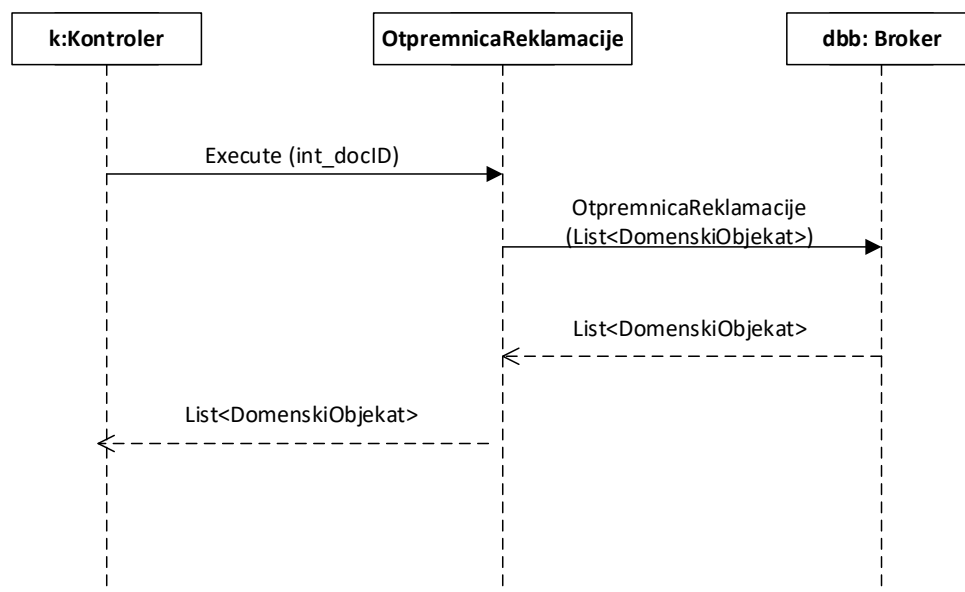
Слика 55 ДС:Уговор - ВратиОбјекатИнфо

Уговор УГ13: **VratiOtpremnicuZaReklamaciju** (Dokument): Signal;

Веза са СК: СК4

Предуслови: /

Постуслови: /



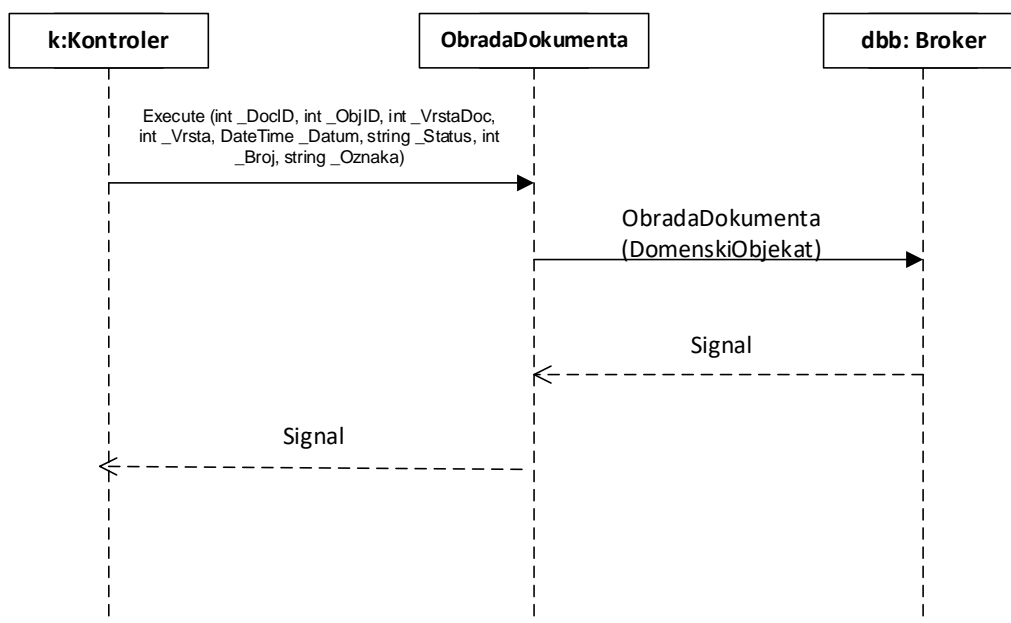
Слика 56 ДС:Уговор - ВратиОтпремницуЗаРекламацију

Уговор УГ14: **ObradaDokumenta** (Dokument): Signal;

Веза са СК: СК1, СК2, СК3, СК4, СК5, СК6

Предуслови: Вредносна и структурна ограничења над објектом **Dokument**, **Objekti** и **VrstaRM** морају бити задовољена.

Постуслови: Евидентирани и запамћени су подаци о документу (требовање, доставница, отпремница...)



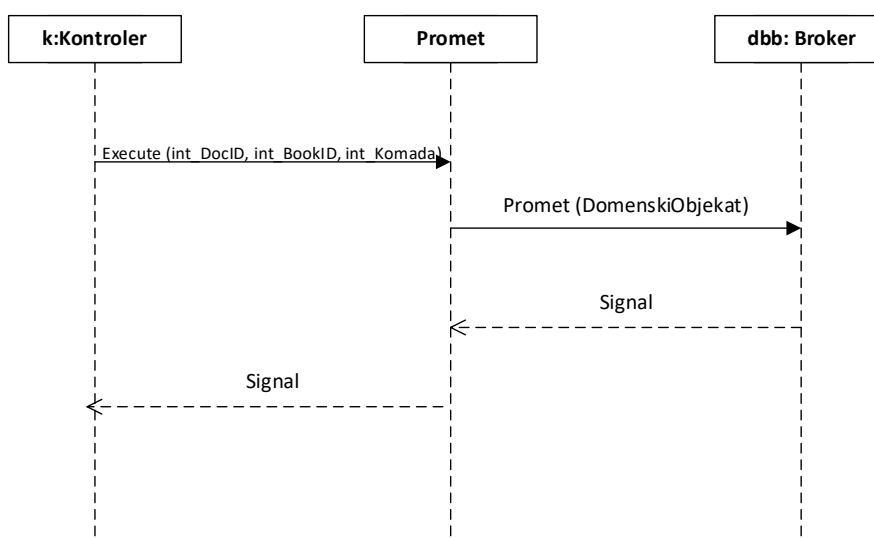
Слика 57 ДС:Уговор - ОбрадаДокумента

Уговор УГ15: **Promet** (Dokument): Signal;

Веза са СК: СК1, СК2, СК3, СК4, СК5, СК6

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

Постуслови: Евидентиран промет, додата је ставка у оквиру промета.



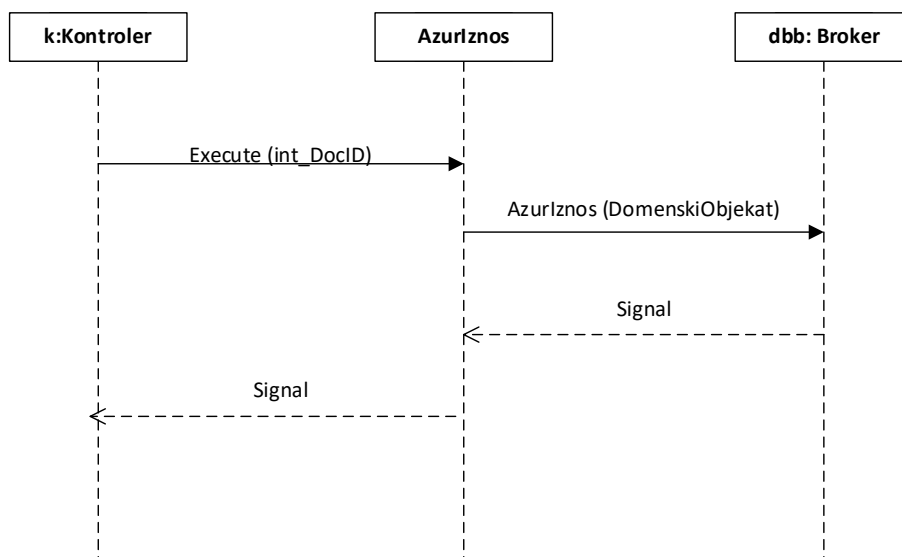
Слика 58 ДС:Уговор - Промет

Уговор УГ16: **AzurIznos** (Dokument): Signal;

Веза са СК: СК4

Предуслови: Вредносна и структурна ограничења над објектом **Dokument** морају бити задовољена – позитивна вредност улазног параметра

Постуслови: Враћа ажуриран статус документа у случају да постоји рекламација.



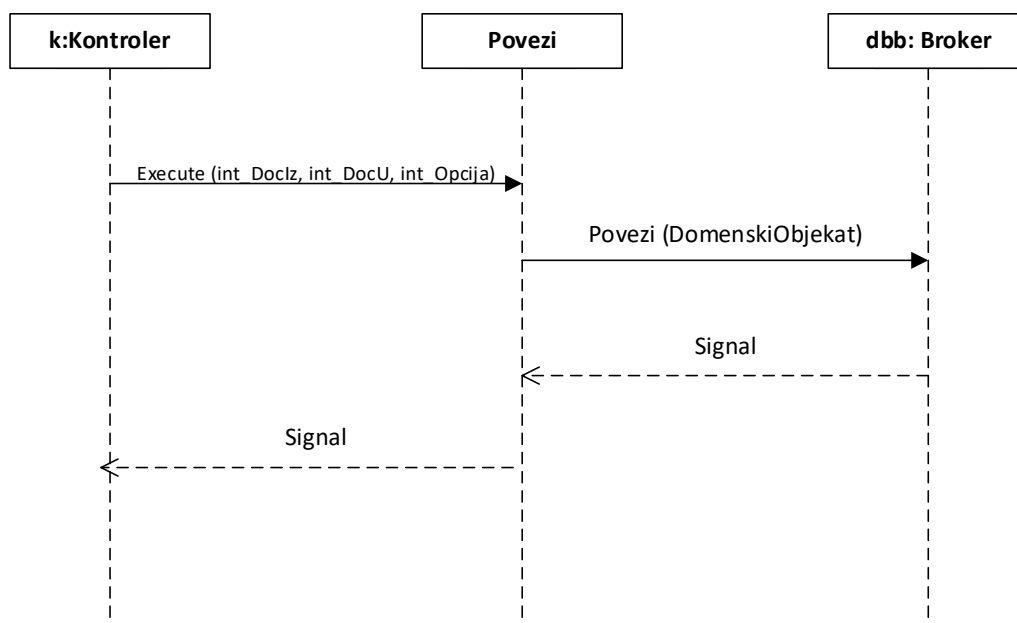
Слика 59 ДС:Уговор - АжурИзнос

Уговор УГ17: **Povezi** (Dokument, Dokument): Signal;

Веза са СК: СК6

Предуслови: Вредносна и структурна ограничења над објектом **Dokument** морају бити задовољена – вредности параметара морају бити различити

Постуслови: Повезује жељена документа (требовање и пријемницу, пријемницу и отпремницу,...)



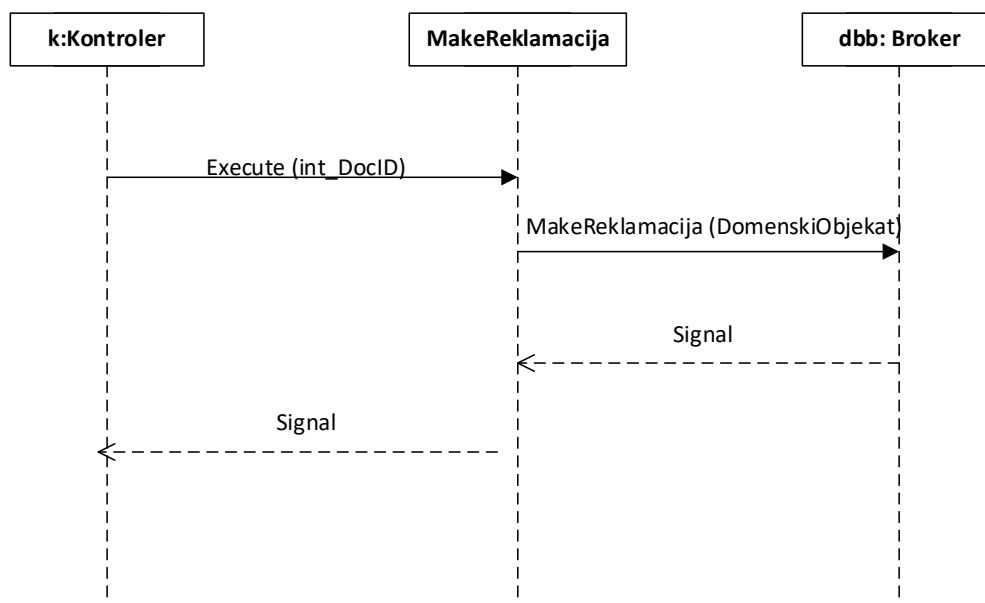
Слика 60 ДС:Уговор - Повежи

Уговор УГ18: **MakeReklamacija** (Dokument): Signal;

Веза са СК: СК4

Предуслови: Вредносна и структурна ограничења над објектом **Dokument** и **VrstaDokumenta** морају бити задовољена – позитивна вредност улазног параметра

Постуслови: Враћа креирану рекламацију за отпремницу и пријемницу које немају исте количине на истим књигама



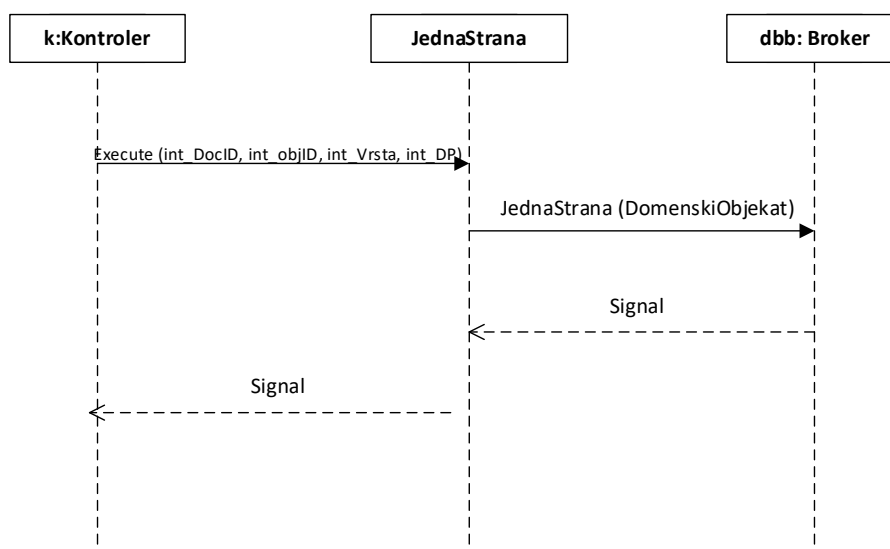
Слика 61 ДС:Уговор - MakeРекламација

Уговор УГ19: **JednaStrana** (Dokument): Signal;

Веза са СК: СК4

Предуслови: Вредносна и структурна ограничења над објектом **Dokument** и **VrstaDokumenta, Objekat** морају бити задовољена – позитивна вредност улазног параметра

Постуслови: Синхронизовано стање на документу на основу кога је креирана рекламација



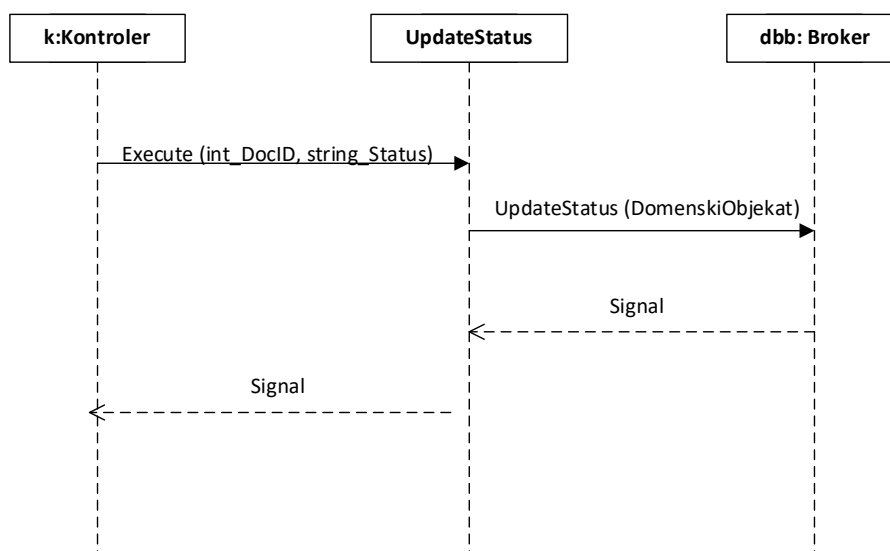
Слика 62 ДС:Уговор - JedнаСтрана

Уговор УГ20: **UpdateStatus** (Dokument): Signal;

Веза са СК: СК2, СК3, СК4, СК5, СК6

Предуслови: Вредносна и структурна ограничења над објектом **Dokument** морају бити задовољена – позитивна вредност улазног параметра

Постуслови: Ажуриран статус изабраног документа



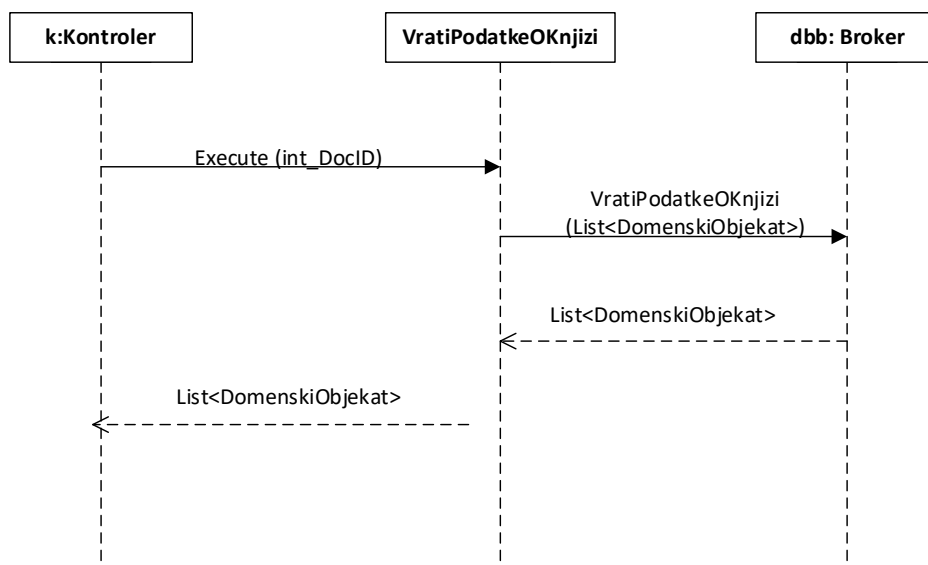
Слика 63 ДС:Уговор - УпдатеСтатус

Уговор УГ21: **VratiPodatkeOKnjizi** (Knjiga): Signal;

Веза са СК: СК1

Предуслови: /

Постуслови: /



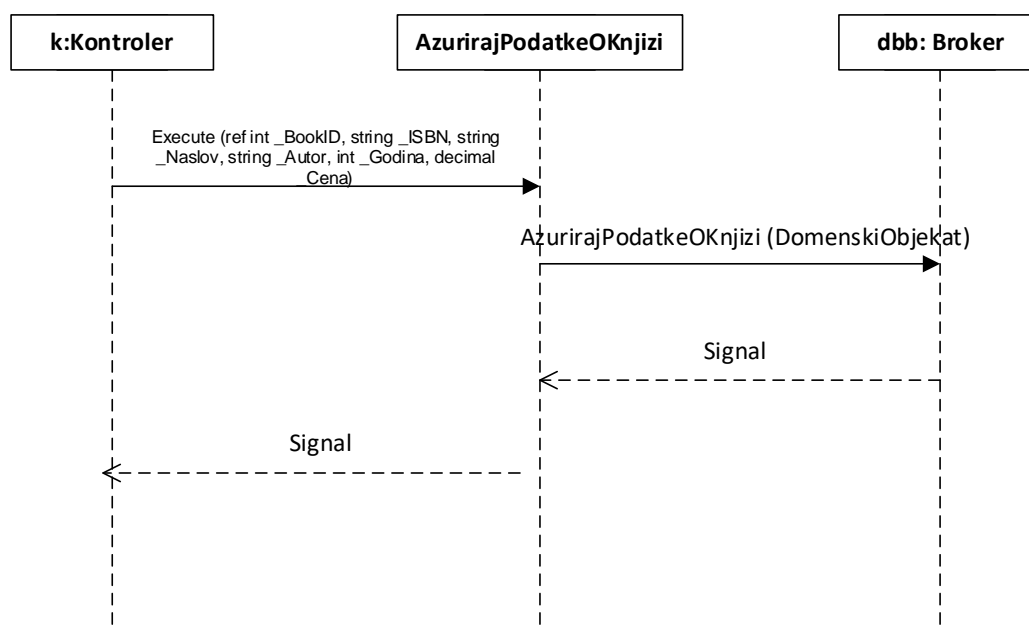
Слика 64 ДС:Уговор - ВратиПодаткеОКњизи

Уговор УГ22: **AzurirajPodatkeOKnjizi** (Knjiga): Signal;

Веза са СК: СК1

Предуслови: /

Постуслови: /



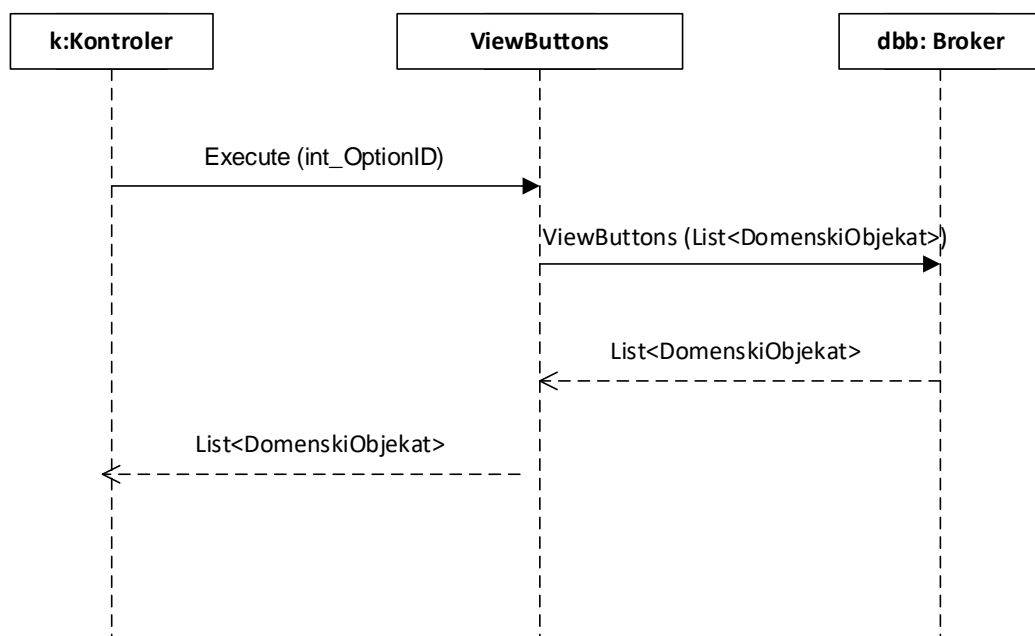
Слика 65 ДС:Уговор - АжурирајПодаткеОКњизи

Уговор УГ23: **VratiViewButtons** (VrstaDokumenta): Signal;

Веза са СК: СК1, СК2, СК3, СК4, СК5, СК6

Предуслови: /

Постуслови: /



Слика 66 ДС:Уговор – ВратиВјуБуттонс

3.5 Имплементација

Пројекат је реализован у програмском језику C#. Као развојно окружење коришћен је Visual Studio 2022. За систем за управљањем базом података коришћен је Microsoft SQL Server Management Studio 2018.

3.5.1 Структура софтверског система

Систем се састоји од седам пројеката: DbBroker, PoslovnaLogika, Klijent, AdministratorskiNalog, SistemskeOperacije, Zajednicki.

1. DbBroker
 - i. SrednjiSloj.DB.cs
2. Poslovna Logika
 - i. SrednjiSloj.Pravila.cs
3. Klijent
 - i. Exceptions
 - i. ServerException.cs
 - ii. Forme
 - i. frmMain.cs
 - ii. frmDoc.cs
 - iii. frmLogin.cs
 - iv. frmKnjiga.cs
 - v. frmIzbor.cs
 - vi. frmObjekti.cs
 - vii. frmProdaja.cs
 - viii. frmZaposleni.cs
 - ix. SplashScreen1.cs
 - x. frmInput.cs
 - xi. frmChangePWD.cs
 - xii. frmPromet.cs
 - xiii. frmPregled.cs
 - xiv. frmPromet2.cs
 - xv. frmPregled2.cs
 - xvi. frm
 - iii. Korisničke Kontrole
 - i. BookSearch.cs
4. Administratorski Nalog
 - i. Admin (Aplikativni korisnik sql server)
5. SistemskeOperacije (u SrednjiSloj)
 - xvi. 1. Signal ObradaDokumenta (Dokument)



xvii. 2. Signal UcitajListuKnjiga (Lista <Knjiga>)

6. Zajednicki


- i. BookSearch.cs
- ii. SplashScreen1.cs

3.5.2 Имплементација складишта података

На основу софтверских класа структуре пројектоване су табеле (складишта података) релационог система за управљање базом података: Brojac, Dokument, Knjige, Objekti, Promet, Strana, Stavka, Veza, VrstaDokumenta, VrstaObjekta, VrstaRM, Zaposleni.

	Column Name	Data Type	Allow Nulls
	vrstadoc	int	<input type="checkbox"/>
	godina	int	<input type="checkbox"/>
	broj	int	<input type="checkbox"/>

Слика 67 Бројачи

	Column Name	Data Type	Allow Nulls
	docid	int	<input type="checkbox"/>
	vrstadoc	int	<input type="checkbox"/>
	objiz	int	<input type="checkbox"/>
	vrstaiz	int	<input type="checkbox"/>
	obju	int	<input type="checkbox"/>
	vrstau	int	<input type="checkbox"/>
	datum	date	<input type="checkbox"/>
	broj	int	<input type="checkbox"/>
	oznaka	varchar(20)	<input type="checkbox"/>
	iznos	decimal(18, 2)	<input type="checkbox"/>
	status	char(1)	<input type="checkbox"/>
	empid	int	<input type="checkbox"/>
	datvreme	datetime	<input type="checkbox"/>

Слика 68 Документ

	Column Name	Data Type	Allow Nulls
🔑	bookid	int	<input type="checkbox"/>
	isbn	varchar(50)	<input type="checkbox"/>
	naslov	nvarchar(75)	<input checked="" type="checkbox"/>
	autor	nvarchar(50)	<input type="checkbox"/>
	godina	int	<input type="checkbox"/>
	cena	decimal(18, 2)	<input type="checkbox"/>

Слика 69 Књиге

	Column Name	Data Type	Allow Nulls
🔑	objid	int	<input type="checkbox"/>
	objvrsta	int	<input type="checkbox"/>
	pripada	int	<input type="checkbox"/>
	sifra	varchar(10)	<input type="checkbox"/>
	naziv	nvarchar(50)	<input type="checkbox"/>
	adresa	nvarchar(50)	<input type="checkbox"/>
	mesto	nvarchar(50)	<input type="checkbox"/>
	vrsta	int	<input type="checkbox"/>
	vazi	char(1)	<input type="checkbox"/>

Слика 70 Објекти

	Column Name	Data Type	Allow Nulls
🔑	docid	int	<input type="checkbox"/>
🔑	bookid	int	<input type="checkbox"/>
	komada	int	<input type="checkbox"/>

Слика 71 Промет

	Column Name	Data Type	Allow Nulls
🔑	docid	int	<input type="checkbox"/>
🔑	objid	int	<input type="checkbox"/>
🔑	idstavke	int	<input type="checkbox"/>

Слика 72 Ставка

	Column Name	Data Type	Allow Nulls
	opcija	int	<input type="checkbox"/>
	dociz	int	<input type="checkbox"/>
	docu	int	<input type="checkbox"/>

Слика 73 Веза

	Column Name	Data Type	Allow Nulls
?	vrstadoc	int	<input type="checkbox"/>
	dokument	nvarchar(50)	<input type="checkbox"/>
	vazi	char(1)	<input type="checkbox"/>

Слика 74 ВрстаДокумента

	Column Name	Data Type	Allow Nulls
?	objvrsta	int	<input type="checkbox"/>
	vrstaobjekta	nvarchar(50)	<input type="checkbox"/>
	vazi	char(1)	<input type="checkbox"/>

Слика 75 ВрстаОбјекта

	Column Name	Data Type	Allow Nulls
?	vrsta	int	<input type="checkbox"/>
	naziv	nvarchar(50)	<input type="checkbox"/>

Слика 76 ВрстаРМ

	Column Name	Data Type	Allow Nulls
?	empid	int	<input type="checkbox"/>
	ime	nvarchar(50)	<input type="checkbox"/>
	sifra	varchar(50)	<input checked="" type="checkbox"/>
	lozinka	varchar(20)	<input type="checkbox"/>
	objid	int	<input type="checkbox"/>
	vrsta	int	<input type="checkbox"/>
	vazi	char(1)	<input type="checkbox"/>

Слика 77 Запослени

3.5.3 Имплементација апликативне логике

Овај одељак приказује имплементирање комуникације пословне логике и брокера базе података.

Пословна логика – системске операције

У овом одељку ће бити представљена имплементација системских операција. Прво ће бити представљена општа системска операција која је дефиниса преко Template Method патерна. Све друге системске операције наслеђују општу системску операцију и дају конкретну имплементацију за методе *Execute* и *Validate*.

```

using System;
using System.Collections;
using System.Collections.Generic;
using System.Data;
using System.Diagnostics;
using System.Linq;
using System.Xml.Linq;

namespace SrednjiSloj
{
    public class Pravila
    {
        public enum ERMesta
        {
            Prodavac = 0,
            Magacioner = 1,
            Uprava = 2,
            Finansije = 3,
            Dobavljac = 4
        }

        public static int EmpID {get; set;}

        private static string StrA =
"ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789.,!?:-_" ;
        private static string StrB = "qrstuJ?@:-
_mnopLMNOPQRSabTUV89.,;EFGABCDK!WXYZc456defghijklvxyz01237HI";

        private static string Crypt(string _Source)
        {
            string fja = "";
            for (int i = 0; i < _Source.Count(); i++)
            {
                string slovo = _Source.Substring(i, 1);
                if (StrA.IndexOf(slovo) >= 0)
                {
                    fja += StrB.Substring(_Source.IndexOf(slovo), 1);
                }
                else
                {
                    fja += slovo;
                }
            }
            return fja;
        }

        private static string Decrypt(string _Source)
        {
            string fja = "";
            for (int i = 0; i < StrB.Count(); i++)
            {
                if (StrB.Contains(_Source.Substring(i, 1)))
                {
                    fja =
StrA.Substring(StrB.IndexOf(_Source.Substring(i, 1)), 1);
                }
                else
                {
                    fja += StrB.Substring(i, 1);
                }
            }
        }
    }
}

```

```

    }
    }
    return fja;
}

public static bool Dokumenta(ref int _DocID, int _VrstaDoc, int
_ObjIz, int _VrstaIz, int _ObjU, int _VrstaU, DateTime _Datum, ref int _Broj,
ref string _Oznaka, decimal _Iznos, string _Status, ref string _Error)
{
    System.Data.SqlClient.SqlCommand comm = new
System.Data.SqlClient.SqlCommand("DokumentWork");
    comm.CommandType = CommandType.StoredProcedure;

    comm.Parameters.AddWithValue("@docid", _DocID);
    comm.Parameters.AddWithValue("@vrstadoc", _VrstaDoc);
    comm.Parameters.AddWithValue("@objiz", _ObjIz);
    comm.Parameters.AddWithValue("@vrstaiz", _VrstaIz);
    comm.Parameters.AddWithValue("@obju", _ObjU);
    comm.Parameters.AddWithValue("@vrstau", _VrstaU);
    comm.Parameters.AddWithValue("@datum",
Convert.ToDateTime(_Datum));
    comm.Parameters.AddWithValue("@broj", _Broj);
    comm.Parameters.AddWithValue("@oznaka", _Oznaka);
    comm.Parameters.AddWithValue("@iznos", (decimal)_Iznos);
    comm.Parameters.AddWithValue("@status", _Status);
    comm.Parameters.AddWithValue("@empid", EmpID);
    DataTable tt = DB.GetData(comm, ref _Error);
    if (_Error == null)
    {
        if (tt.Rows.Count > 0)
        {
            _DocID = Convert.ToInt32(tt.Rows[0][0]);
            _Broj = Convert.ToInt32(tt.Rows[0]["broj"]);
            _Oznaka = tt.Rows[0]["oznaka"].ToString();
            return true;
        }
        else
        {
            _Error = "No data.";
            return false;
        }
    }
    else
    {
        return false;
    }
}

public static bool DodajStranu(int _DocID, ref string _Error)
{
    System.Data.SqlClient.SqlCommand comm = new
System.Data.SqlClient.SqlCommand("DodajStranu");
    comm.CommandType = CommandType.StoredProcedure;
    comm.Parameters.AddWithValue("@docid", _DocID);
    return DB.Execute(comm, ref _Error);
}

```

```

        public static bool Promet(int _DocID, int _BookID, int _Komada, ref
string _Error)
        {
            System.Data.SqlClient.SqlCommand comm = new
System.Data.SqlClient.SqlCommand("PrometWork");
            comm.CommandType = CommandType.StoredProcedure;
            comm.Parameters.AddWithValue("@docid", _DocID);
            comm.Parameters.AddWithValue("@bookid", _BookID);
            comm.Parameters.AddWithValue("@komada", _Komada);
            return DB.Execute(comm, ref _Error);
        }

        public static DataTable VrtiViewData(int _ObjID, int _Vrsta, ref
string _Error)
        {
            System.Data.SqlClient.SqlCommand comm = new
System.Data.SqlClient.SqlCommand("ViewData");
            comm.CommandType = CommandType.StoredProcedure;
            comm.Parameters.AddWithValue("@vrsta", _Vrsta);
            comm.Parameters.AddWithValue("@objid", _ObjID);
            return DB.GetData(comm, ref _Error);
        }

        public static bool AzurIznos(int _DocID, ref string _Error)
        {
            System.Data.SqlClient.SqlCommand comm = new
System.Data.SqlClient.SqlCommand("AzurStanje");
            comm.CommandType = CommandType.StoredProcedure;
            comm.Parameters.AddWithValue("@docid", _DocID);
            return DB.Execute(comm, ref _Error);
        }

        public static bool Povezi(int _DocIz, int _DocU, int _OPcija, ref
string _Error)
        {
            System.Data.SqlClient.SqlCommand comm = new
System.Data.SqlClient.SqlCommand("VezaWork");
            comm.CommandType = CommandType.StoredProcedure;
            comm.Parameters.AddWithValue("@dociz", _DocIz);
            comm.Parameters.AddWithValue("@docu", _DocU);
            comm.Parameters.AddWithValue("@opcija", _OPcija);
            return DB.Execute(comm, ref _Error);
        }

        public static bool MakeReklamacija(int _DocID, ref string _Error)
        {
            System.Data.SqlClient.SqlCommand comm = new
System.Data.SqlClient.SqlCommand("MakeReklamacija");
            comm.CommandType = CommandType.StoredProcedure;
            comm.Parameters.AddWithValue("@docid", _DocID);
            return DB.Execute(comm, ref _Error);
        }

        public static DataTable SinhroTrebovanje(int _DocID, int _Magacin,
ref string _Error)
        {
            System.Data.SqlClient.SqlCommand comm = new
System.Data.SqlClient.SqlCommand("SinhroTrebovanje");
            comm.CommandType = CommandType.StoredProcedure;

```

```

        comm.Parameters.AddWithValue("@docid", _DocID);
        comm.Parameters.AddWithValue("@magacin", _Magacin);
        return DB.GetData(comm, ref _Error);
    }

    public static bool UpdateStatus(int _DocID, string _Status, ref
string _Error)
    {
        string s = "UPDATE Dokumenta Set Status = '" + _Status + "'
WHERE docid = " + _DocID.ToString();
        return DB.Execute(s, ref _Error);
    }

    public static bool JednaStrana(int _DocID, int _ObjID, int _Vrsta,
int _DP, ref string _Error)
    {
        string s = "INSERT INTO Strana SELECT " + _DocID.ToString() +
", " + _ObjID.ToString() + ", " + _Vrsta.ToString() + ", " + _DP.ToString();
        return DB.Execute(s, ref _Error);
    }

    public static DataTable VratiOtpremnicuZaReklamaciju(int _DocID, ref
string _Error)
    {
        string s = "select * from OtpremnicaReklamacije(" +
_DocID.ToString() + ")";
        return DB.GetData(s, ref _Error);
    }

    public static DataTable VratiListuDokumenata(int _ObjID, int _Vrsta,
DateTime _FromDate, DateTime _ToDate, ref string _Error)
    {
        System.Data.SqlClient.SqlCommand comm = new
System.Data.SqlClient.SqlCommand("previewdoc");
        comm.CommandType = CommandType.StoredProcedure;
        comm.Parameters.AddWithValue("@vrsta", _Vrsta);
        comm.Parameters.AddWithValue("@objid", _ObjID);
        comm.Parameters.AddWithValue("@oddana", _FromDate);
        comm.Parameters.AddWithValue("@dodana", _ToDate);
        return DB.GetData(comm, ref _Error);
    }

    public static DataTable Zaposleni(int _ID, int _RM, int _Objekat,
string _Ime, string _Sifra, string _Vazi, ref string _Error)
    {
        System.Data.SqlClient.SqlCommand comm = new
System.Data.SqlClient.SqlCommand("ZaposleniWork");
        comm.CommandType = CommandType.StoredProcedure;
        comm.Parameters.AddWithValue("@empid", _ID);
        comm.Parameters.AddWithValue("@vrsta", _RM);
        comm.Parameters.AddWithValue("@objid", _Objekat);
        comm.Parameters.AddWithValue("@ime", _Ime);
        comm.Parameters.AddWithValue("@sifra", _Sifra);
        comm.Parameters.AddWithValue("@vazi", _Vazi);
        return DB.GetData(comm, ref _Error);
    }

    public static bool UpdatePWD(int _ID, string _PWD, ref string _Error)
    {

```

```

        System.Data.SqlClient.SqlCommand comm = new
System.Data.SqlClient.SqlCommand("UpdateLogin");
        comm.CommandType = CommandType.StoredProcedure;
        comm.Parameters.AddWithValue("@empid", _ID);
        _PWD = Crypt(_PWD);
        comm.Parameters.AddWithValue("@newpwd", _PWD);
        DataTable tbl = DB.GetData(comm, ref _Error);
        if (_Error == null)
        {
            if (tbl.Rows.Count == 0)
            {
                return false;
            }
            else
            {
                return true;
            }
        }
        else
        {
            return false;
        }
    }

public static bool IsUserValid(string _Login, string _PWD, ref string
_Error)
{
    bool fja = false;
    if (_Login == "admin" && _PWD == "Maestro")
    {
        fja = true;
    }
    else
    {
        if (_Login != _PWD)
        {
            _PWD = Crypt(_PWD);
        }
        if (Convert.ToInt32(DB.GetValue("SELECT dbo.IsUserValid('"
+ _Login + "',' + _PWD + "')", ref _Error)) == 1)
        {
            EmpID = Convert.ToInt32(DB.GetValue("SELECT empid
from Zaposleni where sifra = '" + _Login + "' And vazi = '1'", ref _Error));
            fja = true;
        }
        else
        {
            fja = false;
        }
    }
    return fja;
}

public static bool Connect(ref string _Error)
{
    return DB.Connect(ref _Error);
}

```



```

        public static DataTable Magacin(int _ID, string _Sifra, string
_Naziv, string _Adresa, string _Mesto, string _Vazi, ref string _Error)
        {
            System.Data.SqlClient.SqlCommand comm = new
System.Data.SqlClient.SqlCommand("ObjektiWork");
            comm.CommandType = CommandType.StoredProcedure;
            comm.Parameters.AddWithValue("@objid", _ID);
            comm.Parameters.AddWithValue("@objvrsta", 2);
            comm.Parameters.AddWithValue("@pripada", 0);
            comm.Parameters.AddWithValue("@sifra", _Sifra);
            comm.Parameters.AddWithValue("@naziv", _Naziv);
            comm.Parameters.AddWithValue("@adresa", _Adresa);
            comm.Parameters.AddWithValue("@mesto", _Mesto);
            comm.Parameters.AddWithValue("@vazi", _Vazi);
            return DB.GetData(comm, ref _Error);
        }

        public static DataTable Knjizara(int _ID, int _Magacin, string
_Sifra, string _Naziv, string _Adresa, string _Mesto, string _Vazi, ref string
_Error)
        {
            System.Data.SqlClient.SqlCommand comm = new
System.Data.SqlClient.SqlCommand("ObjektiWork");
            comm.CommandType = CommandType.StoredProcedure;
            comm.Parameters.AddWithValue("@objid", _ID);
            comm.Parameters.AddWithValue("@objvrsta", 1);
            comm.Parameters.AddWithValue("@pripada", _Magacin);
            comm.Parameters.AddWithValue("@sifra", _Sifra);
            comm.Parameters.AddWithValue("@naziv", _Naziv);
            comm.Parameters.AddWithValue("@adresa", _Adresa);
            comm.Parameters.AddWithValue("@mesto", _Mesto);
            comm.Parameters.AddWithValue("@vazi", _Vazi);
            return DB.GetData(comm, ref _Error);
        }

        public static DataTable Knjige(int _ID, string _ISBN, string _Naslov,
string _Autor, int _Godina, ref string _Error)
        {
            System.Data.SqlClient.SqlCommand comm = new
System.Data.SqlClient.SqlCommand("KnjigeWork");
            comm.CommandType = CommandType.StoredProcedure;
            comm.Parameters.AddWithValue("@bookid", _ID);
            comm.Parameters.AddWithValue("@isbn", _ISBN);
            comm.Parameters.AddWithValue("@naslov", _Naslov);
            comm.Parameters.AddWithValue("@autor", _Autor);
            comm.Parameters.AddWithValue("@godina", _Godina);
            //comm.Parameters.AddWithValue("@vazi", _Vazi)
            return DB.GetData(comm, ref _Error);
        }

        public static DataTable VратиSpisakMagacina(ref string _Error)
        {
            return DB.GetData("SELECT * FROM Magacini()", ref _Error);
        }

        public static DataTable VратиKnjizareZaMagacin(int _ID, ref string
_Error)
        {
            if (_ID == 0)

```

```

        {
            return DB.GetData("SELECT * FROM Knjizare()", ref _Error);
        }
        else
        {
            return DB.GetData("SELECT * FROM KnjizareZa(" +
                _ID.ToString() + ")", ref _Error);
        }
    }

    public static DataTable SviZaposleni(ref string _Error)
    {
        return DB.GetData("SELECT * FROM VZaposleni", ref _Error);
    }

    public static DataTable ObradaDokumenta (int _DocID, int _ObjID, int
        _VrstaDoc, int _Vrsta, DateTime _Datum, string _Status, int _Broj, string
        _Oznaka, ref string _Error)
    {
        System.Data.SqlClient.SqlCommand comm = new
        System.Data.SqlClient.SqlCommand("DokumentaWork");
        comm.CommandType = CommandType.StoredProcedure;

        comm.Parameters.AddWithValue("@objid", _ObjID);
        comm.Parameters.AddWithValue("@vrstadoc", _VrstaDoc);
        comm.Parameters.AddWithValue("@broj", _Broj);
        comm.Parameters.AddWithValue("@vrsta", _Vrsta);
        comm.Parameters.AddWithValue("@docid", _DocID);
        comm.Parameters.AddWithValue("@oznaka", _Oznaka);
        comm.Parameters.AddWithValue("@datum",
        Convert.ToDateTime(_Datum));
        comm.Parameters.AddWithValue("@empid", EmpID);
        comm.Parameters.AddWithValue("@status", _Status);
        return DB.GetData(comm, ref _Error);
    }

    public static bool TrebovanjeInfo(int _DocID, int _ObjID, int _Vrsta,
        DateTime _Datum, string _Status, ref int _Broj, ref string _Onaka, ref string
        _Error)
    {
        bool fja = false;
        DataTable t = ObradaDokumenta (_DocID, _ObjID, 1, _Vrsta,
        _Datum, _Status, 0, "0", ref _Error);
        if (_Error == null)
        {
            if (t.Rows.Count > 0)
            {
                _Broj = Convert.ToInt32(t.Rows[0]["broj"]);
                _Onaka = t.Rows[0]["oznaka"].ToString();
                fja = true;
            }
        }
        return fja;
    }

    public static bool Prijemnica(int _DocID, int _ObjID, int _Vrsta,
        DateTime _Datum, string _Status, ref int _Broj, ref string _Onaka, ref string
        _Error)

```

```

        {
            bool fja = false;
            DataTable t = ObradaDokumenta (_DocID, _ObjID, 2, _Vrsta,
            _Datum, _Status, 0, "0", ref _Error);
            if (_Error == null)
            {
                if (t.Rows.Count > 0)
                {
                    _Broj = Convert.ToInt32(t.Rows[0]["broj"]);
                    _Onaka = t.Rows[0]["oznaka"].ToString();
                    fja = true;
                }
            }
            return fja;
        }

        public static DataTable TrebovanjaZaPakovanje(int _Vrsta, int _ID,
        ref string _Error)
        {
            if (_Vrsta == 1)
            {
                return DB.GetData("SELECT * FROM Dokumenta WHERE vrstadoc
                = 1 and objid = " + _ID.ToString() + " AND Status = '1'", ref _Error);
            }
            else
            {
                return DB.GetData("SELECT * FROM Dokumenta WHERE vrstadoc
                = 1 and objid in (select objid from Objekti where pripada = " + _ID.ToString()
                + ") AND Status = '1'", ref _Error);
            }
        }

        public static bool EvidPromet(int _DocID, int _GookID, int
        _Zahtevano, int _Odobreno, ref string _Error)
        {
            System.Data.SqlClient.SqlCommand comm = new
            System.Data.SqlClient.SqlCommand("PrometWork");
            comm.CommandType = CommandType.StoredProcedure;

            comm.Parameters.AddWithValue("@docid", _DocID);
            comm.Parameters.AddWithValue("@bookid", _GookID);
            comm.Parameters.AddWithValue("@zahtevano", _Zahtevano);
            comm.Parameters.AddWithValue("@odobreno", _Odobreno);
            return DB.Execute(comm, ref _Error);
        }

        public static DataTable AktivniZaposeni(ref string _Error)
        {
            return DB.GetData("SELECT empid,vrsta,objid,ime,sifra,vazi FROM
            Zaposleni where vazi = '1' and sifra <> 'admin'", ref _Error);
        }

        public static DataTable VrstaRM(ref string _Error, int _Bez = 0)
        {
            if (_Bez == 0)
            {
                return DB.GetData("SELECT * FROM VrstaRM", ref _Error);
            }
            else

```

```

        {
            return DB.GetData("SELECT * FROM VrstaRM where vrsta <> "
+ _Bez.ToString(), ref _Error);
        }
    }

    public static DataTable VратиPrometZa(int _ID, ref string _Error)
    {
        return DB.GetData("SELECT * FROM PrometZa(" + _ID.ToString() +
        ")", ref _Error);
    }

    public static DataTable VратиStanjeZa(int _ID, ref string _Error)
    {
        return DB.GetData("SELECT * FROM Stanje(" + _ID.ToString() +
        ")", ref _Error);
    }

    public static int VратиMagacinZaKnjizaru (int _ID, ref string _Error)
    {
        int i = 0;
        try
        {
            i = Convert.ToInt32(DB.GetValue("SELECT dbo.
VратиMagacinZaKnjizaru(" + _ID.ToString() + ")", ref _Error));
        }
        catch (Exception ex)
        {
            if (_Error != null)
            {
                _Error += "\r\n" + ex.Message;
            }
            else
            {
                _Error = ex.Message;
            }
        }
        return i;
    }

    public static DataTable VратиSpisakPrijemnica (int _ID, ref string
_Error)
    {
        return DB.GetData("SELECT * FROM ZaPrijemnicu(" +
_ID.ToString() + ")", ref _Error);
    }

    public static DataTable VратиStavkeDokumenta(int _DocID, ref string
_Error)
    {
        return DB.GetData("SELECT * FROM DocStavke(" +
_DocID.ToString() + ")", ref _Error);
    }

    public static DataTable VратиPodatkeOKnjizi(int _DocID, ref string
_Error)
    {
        return DB.GetData("SELECT * FROM Dokumenta WHERE docid = " +
_DocID.ToString(), ref _Error);
    }

```

```

    }

    public static DataTable VratiObjekatInfo(int _ObjID, ref string
_Error)
    {
        return DB.GetData("SELECT * FROM Objekti WHERE objid = " +
_ObjID.ToString(), ref _Error);
    }

    public static DataTable VratiViewButtons(int _OptionID, ref string
_Error)
    {
        return DB.GetData("SELECT * FROM Moze(" + _OptionID.ToString()
+ ")", ref _Error);
    }

    public static DataTable KnjigeZaProdaju(int _ObjID, ref string
_Error)
    {
        string s = "select
a.objid,a.bookid,a.ISBN,a.Naslov,a.Autor,k.Cena,a.Stanje from stanje(" +
_ObjID.ToString() + ") a inner join knjige k on a.bookid = k.bookid where stanje
> 0";
        return DB.GetData(s, ref _Error);
    }

    public static DataTable VratiPodatkeOPrijavljenomKorisniku(ref string
_Error)
    {
        return DB.GetData("SELECT * FROM Zaposleni WHERE empid = " +
EmpID.ToString(), ref _Error);
    }

    public static DataTable VrstaRMInfo(int _RM, ref string _Error)
    {
        return DB.GetData("SELECT naziv from VrstaRM where vrsta = " +
_RM.ToString(), ref _Error);
    }

    public static bool UpdateStatus(string _NewStatus, int _VrstaDoc,
string _ActiveStatus, int _ObjID, ref string _Error)
    {
        string s = "update Dokumenta set status = '" + _NewStatus + "'
where vrstadoc = " + _VrstaDoc.ToString() + " and status = '" + _ActiveStatus +
"' and obju = " + _ObjID.ToString();
        return DB.Execute(s, ref _Error);
    }

    public static DataTable VratiSpisakKnjiga(ref string _Error)
    {
        return DB.GetData("SELECT * FROM Knjige", ref _Error);
    }

    public static DataTable VratiPodatkeOKnjizi(int _bookID, ref string
_Error)
    {
        return DB.GetData("SELECT * FROM Knjige WHERE bookid = " +
_bookID.ToString(), ref _Error);
    }

```

```

        public static bool AzurirajPodatkeOKnjizi(ref int _BookID, string
        _ISBN, string _Naslov, string _Autor, int _Godina, decimal _Cena, ref string
        _Error)
        {
            bool fja = false;
            if (Convert.ToInt32(DB.GetValue("SELECT COUNT(*) As Broj FROM
Knjige WHERE ISBN = '" + _ISBN + "'", ref _Error)) > 0)
            {
                _BookID = Convert.ToInt32(DB.GetValue("SELECT BookID FROM
Knjige WHERE ISBN = '" + _ISBN + "'", ref _Error));
                fja = DB.Execute("UPDATE Knjige SET ISBN = '" + _ISBN +
"',Autor = '" + _Autor + "',Naslov = '" + _Naslov + "',Godina = " +
_Godina.ToString() + ",Cena = " +
Microsoft.VisualBasic.Strings.FormatNumber(_Cena, 2,
Microsoft.VisualBasic.TriState.True, Microsoft.VisualBasic.TriState.UseDefault,
Microsoft.VisualBasic.TriState.False).Replace(",", "."), ref _Error);
            }
            else
            {
                fja = DB.Execute("INSERT INTO
Knjige(ISBN,NASLOV,AUTOR,GODINA,CENA) SELECT '" + _ISBN + "','" + _Naslov +
 "','" + _Autor + "','" + _Godina.ToString() + "','" +
Microsoft.VisualBasic.Strings.FormatNumber(_Cena, 2,
Microsoft.VisualBasic.TriState.True, Microsoft.VisualBasic.TriState.UseDefault,
Microsoft.VisualBasic.TriState.False).Replace(",", "."), ref _Error);
                if (fja)
                {
                    _BookID = Convert.ToInt32(DB.GetValue("SELECT
Max(BookID) As BookID from Knjige", ref _Error));
                    fja = (_Error == null);
                }
            }
            return fja;
        }
    }
}

```

Имплементација брокера

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Data;
using System.Diagnostics;
using System.Linq;
using System.Xml.Linq;

namespace SrednjiSloj
{
    public class DB
    {
        private static System.Data.SqlClient.SqlConnection m_DB;

        public static bool Connect(ref string _Error)
        {
            bool fja = false;
            if (m_DB == null)
            {
                if (Properties.Settings.Default.Trusted)
                {
                    try
                    {
                        m_DB = new
System.Data.SqlClient.SqlConnection("Server=" +
Properties.Settings.Default.DBServer + ";Database=" +
Properties.Settings.Default.DBName + ";Trusted_Connection=True;");
                        m_DB.Open();
                        m_DB.Close();
                        fja = true;
                    }
                    catch (Exception ex)
                    {
                        _Error = ex.Message;
                    }
                }
                else
                {
                    if (Properties.Settings.Default.DBPWD.Trim().Length
> 0)
                    {
                        try
                        {
                            m_DB = new
System.Data.SqlClient.SqlConnection("Server=" +
Properties.Settings.Default.DBServer + ";Database=" +
Properties.Settings.Default.DBName + ";User Id=Knjige;Password=" +
Properties.Settings.Default.DBPWD + ";");
                            m_DB.Open();
                            m_DB.Close();
                            fja = true;
                        }
                        catch (Exception ex1)
                        {
                            _Error = ex1.Message;
                        }
                    }
                }
            }
        }
    }
}
```

```

        }
        else
        {
            _Error = "Mora se definisati lozinka za
prijavu na bazu.";
        }
    }
    else
    {
        fja = true;
    }
    return fja;
}

public static bool Disconnect(ref string _Error)
{
    bool fja = false;
    if (m_DB == null)
    {
        fja = true;
    }
    else
    {
        try
        {
            m_DB.Close();
            m_DB.Dispose();
            m_DB = null;
            fja = true;
        }
        catch (Exception ex)
        {
            _Error = ex.Message;
        }
    }
    return fja;
}

public static DataTable GetData(string _SQL, ref string _Error)
{
    DataTable fja = null;
    if (DB.Connect(ref _Error))
    {
        System.Data.SqlClient.SqlDataAdapter ada = null;
        try
        {
            ada = new System.Data.SqlClient.SqlDataAdapter(_SQL,
m_DB);

            m_DB.Open();
            fja = new DataTable();
            ada.Fill(fja);
        }
        catch (Exception ex)
        {
            _Error = ex.Message;
        }
        finally
        {
            ada.Dispose();

```



```

        ada = null;
        m_DB.Close();
    }
}
return fja;
}
public static DataTable GetData(System.Data.SqlClient.SqlCommand
_SQL, ref string _Error)
{
    DataTable fja = null;
    if (DB.Connect(ref _Error))
    {
        System.Data.SqlClient.SqlDataAdapter ada = null;
        try
        {
            _SQL.CommandType = CommandType.StoredProcedure;
            _SQL.Connection = m_DB;
            ada = new
System.Data.SqlClient.SqlDataAdapter(_SQL);
            m_DB.Open();
            fja = new DataTable();
            ada.Fill(fja);
        }
        catch (Exception ex)
        {
            _Error = ex.Message;
        }
        finally
        {
            ada.Dispose();
            ada = null;
            m_DB.Close();
        }
    }
    return fja;
}
public static DataSet GetDS(string _SQL, ref string _Error)
{
    DataSet fja = null;
    if (DB.Connect(ref _Error))
    {
        System.Data.SqlClient.SqlDataAdapter ada = null;
        try
        {
            ada = new System.Data.SqlClient.SqlDataAdapter(_SQL,
m_DB);

            m_DB.Open();
            fja = new DataSet();
            ada.Fill(fja);
        }
        catch (Exception ex)
        {
            _Error = ex.Message;
        }
        finally
        {
            ada.Dispose();
            ada = null;
            m_DB.Close();
        }
    }
}

```

```

        }
    }
    return fja;
}

public static DataSet GetDS(System.Data.SqlClient.SqlCommand _SQL,
ref string _Error)
{
    DataSet fja = null;
    if (DB.Connect(ref _Error))
    {
        System.Data.SqlClient.SqlDataAdapter ada = null;
        try
        {
            _SQL.CommandType = CommandType.StoredProcedure;
            _SQL.Connection = m_DB;
            ada = new
System.Data.SqlClient.SqlDataAdapter(_SQL);
            m_DB.Open();
            fja = new DataSet();
            ada.Fill(fja);
        }
        catch (Exception ex)
        {
            _Error = ex.Message;
        }
        finally
        {
            ada.Dispose();
            ada = null;
            m_DB.Close();
        }
    }
    return fja;
}

public static bool Execute(string _SQL, ref string _Error)
{
    bool fja = false;
    if (DB.Connect(ref _Error))
    {
        System.Data.SqlClient.SqlCommand comm = new
System.Data.SqlClient.SqlCommand(_SQL);
        comm.CommandType = CommandType.Text;
        try
        {
            m_DB.Open();
            comm.Connection = m_DB;
            comm.ExecuteNonQuery();
            fja = true;
        }
        catch (Exception ex)
        {
            _Error = ex.Message;
        }
        finally
        {
            comm.Dispose();
            comm = null;
            m_DB.Close();
        }
    }
}

```

```

        }
    }
    return fja;
}
public static bool Execute(System.Data.SqlClient.SqlCommand _SQL, ref
string _Error)
{
    bool fja = false;
    if (DB.Connect(ref _Error))
    {
        _SQL.CommandType = CommandType.StoredProcedure;
        try
        {
            m_DB.Open();
            _SQL.Connection = m_DB;
            _SQL.ExecuteNonQuery();
            fja = true;
        }
        catch (Exception ex)
        {
            _Error = ex.Message;
        }
        finally
        {
            _SQL.Dispose();
            _SQL = null;
            m_DB.Close();
        }
    }
    return fja;
}
public static object GetValue(string _SQL, ref string _Error)
{
    object fja = null;
    DataTable tbl = GetData(_SQL, ref _Error);
    if (_Error == null)
    {
        if (tbl != null)
        {
            if (tbl.Rows.Count > 0)
            {
                fja = tbl.Rows[0][0];
            }
            else
            {
                _Error = "No data.";
            }
        }
        else
        {
            _Error = "Not exist.";
        }
    }
    return fja;
}
}
}
}

```

3.5.4 Брокер базе података

Брокер базе података је одговоран за комуникацију између пословне логике и складишта података. Улогу брокера базе података има класа `Broker`. Класа `Broker` обезбеђује перзистентни оквир који посредује у свим операцијама над базом података

Све методе класе `Broker` су пројектоване као генеричке што значи да могу да прихвате различите доменске објекте прослеђене као аргументе. Над тим објектима позивају се методе специфициране у интерфејсу за које свака од класа која имплементира тај интерфејс имплементира на себи својствен начин. На тај начин добија се потпуно генерички брокер базе података.

На слици испод приказана је структура као и понашање софтверског система.

3.5.5 Имплементација корисничког интерфејса

У овом одељку ће бити представљена имплементација корисничког интерфејса који је дефинисан преко скупа екранских форми док су сценарија коришћења екранских форми директно повезана са сценаријима случајева коришћења. Улогу екранске форме јесте да прихвати податке које уноси актор, да прихвати догађаје које он прави, да позива контролер корисничког интерфејса како би му проследила податке и да прикаже податке добијене од контролера корисничког интерфејса.

СК1: Случај коришћења – Креирање требовања складишту

Назив СК

Креирање **требовања складишту**

Актори СК

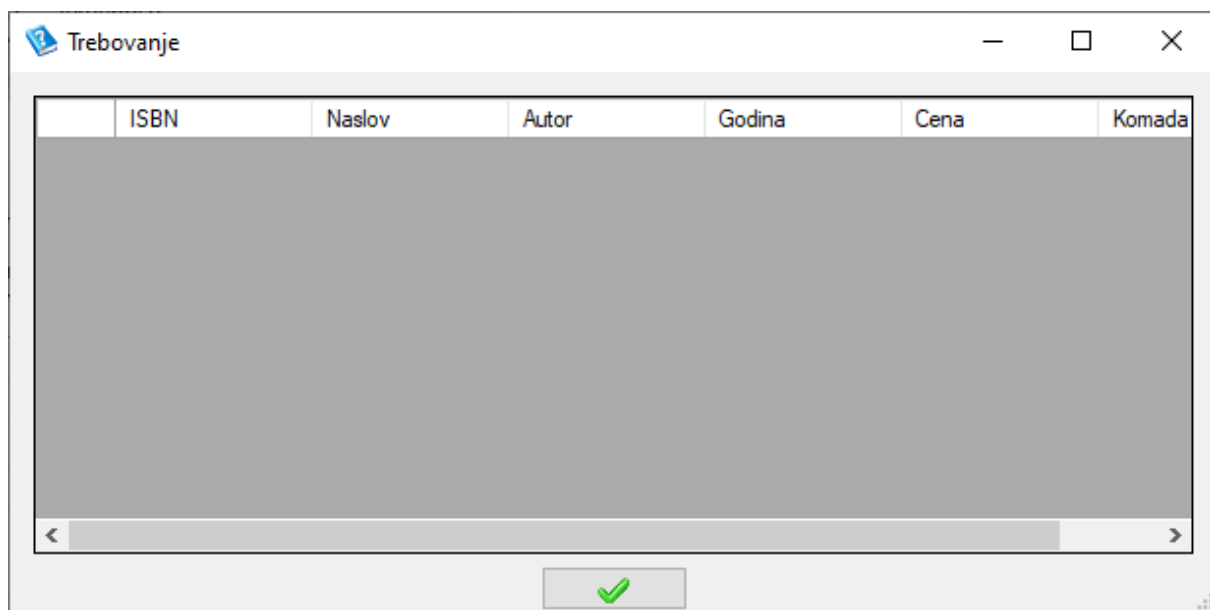
Запослени у књижари

Учесници СК

Запослени у књижари и **систем** (програм)

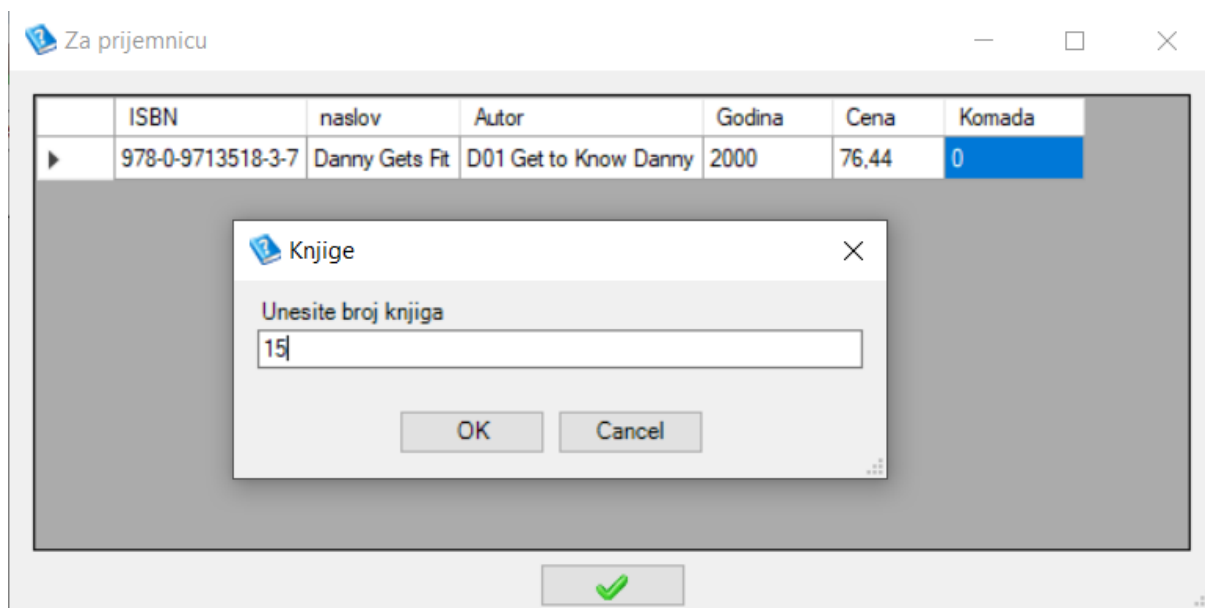
Предуслов: **Систем** је укључен и **запослени у књижари** је улогован под својом шифром. **Запослени у књижари** бира опцију „Требовање“ са главне форме. Систем приказује форму за унос **требовања**. Учитана је листа књига.

Основни сценарио СК



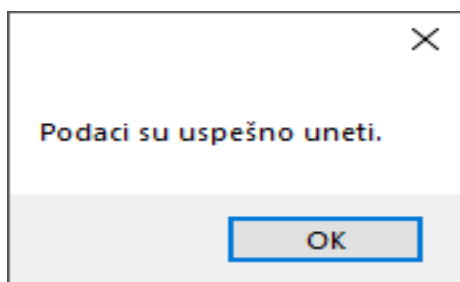
Слика 79 Форма за унос требовања

1. **Запослени у књижари** додаје књиге и жељене количине (АПУСО)



Слика 80 Бирање књиге и потребне количине

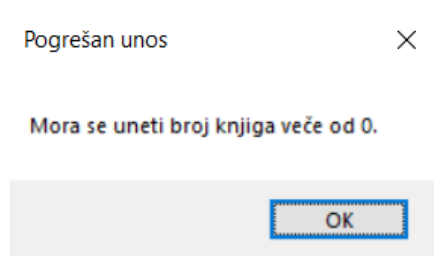
2. **Запослени у књижари** контролише да ли је коректно унео податке за тражено **требовање**. (АНСО)
3. **Запослени у књижари** **позива систем** да запамти податке о датом **требовању**. (АПСО)
Опис акције: **Запослени у књижари** кликом на дугме „ОК“ позива системску операцију ObradaDokumenta (Dokument) и Promet (docid)
4. **Систем памти** податке о **изабраним књигама**. (СО)
5. **Систем** аутоматски **креира пријемницу** за складиште и **приказује** **запосленом у књижари** поруку: „Подаци су успешно унети“. (ИА)



Слика 81 Успешан унос требовања

Алтернативни сценарио

- 5.1 Уколико **систем** није у могућности да креира требовање због неправилног уноса, он приказује **запосленом у књижари** поруку: „Мора се унети број књига веће од 0“. (ИА)



Слика 82 Неуспешан унос требовања

СК2: Случај коришћења – Креирање пријемнице добављачу

Назив СК

Креирање **пријемнице добављачу**

Актори СК

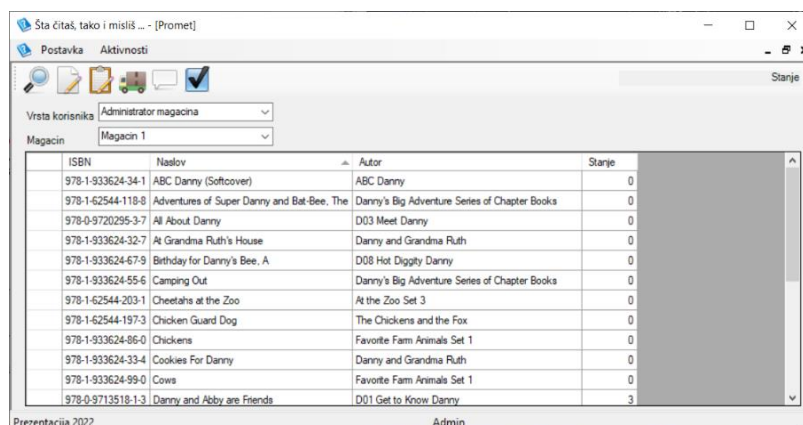
Администратор складишта

Учесници СК

Администратор складишта и **систем** (програм)

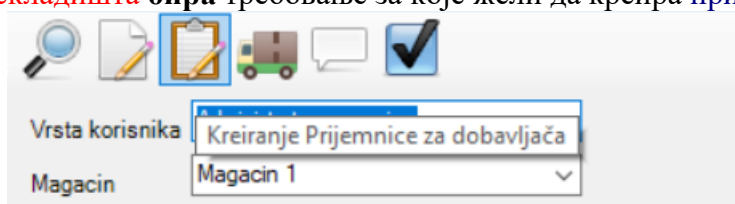
Предуслов: Систем је укључен и **администратор складишта** је улогован под својом шифром. **Администратор складишта** бира опцију „Да ли има захтева из књижара (Требовање)“ са главне форме. Систем враћа списак свих постојећих требовања која су извршена према датом складишту.

Основни сценарио СК



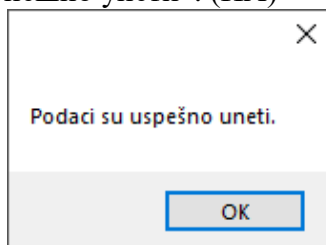
Слика 83 Форма са списком свих постојећих требовања

1. **Администратор складишта бира** требовање за које жели да креира **пријемницу**". (АПУСО)



Слика 84 Опција Креирање Пријемнице за добављача

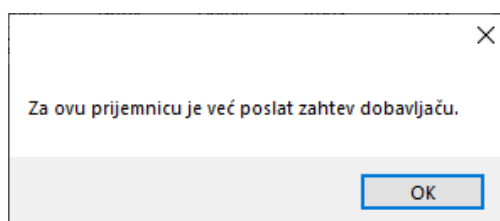
1. **Администратор складишта позива систем** да запамти податке о **пријемници** (АПСО)
Опис акције: **Администратор складишта** кликом на дугме „ОК“ позива системску операцију ObradaDokumenta (Dokument) и Promet (docid)
2. **Систем памти** податке о **креираној пријемници**. (СО)
3. **Систем** аутоматски **креира пријемницу** за добављача и **приказује администратору складишта** поруку: „Подаци су успешно унети“. (ИА)



Слика 85 Успешно креирање пријемнице

Алтернативни сценарио

- 4.1 Уколико је **пријемница** већ креирана, **Систем приказује администратору складишта** поруку: „За ову **пријемницу** је већ послат захтев добављачу“. (ИА)



Слика 86 Неуспешно креирање пријемнице

СКЗ: Случај коришћења – Креирање отпремнице за складиште

Назив СК

Креирање отпремнице

Актори СК

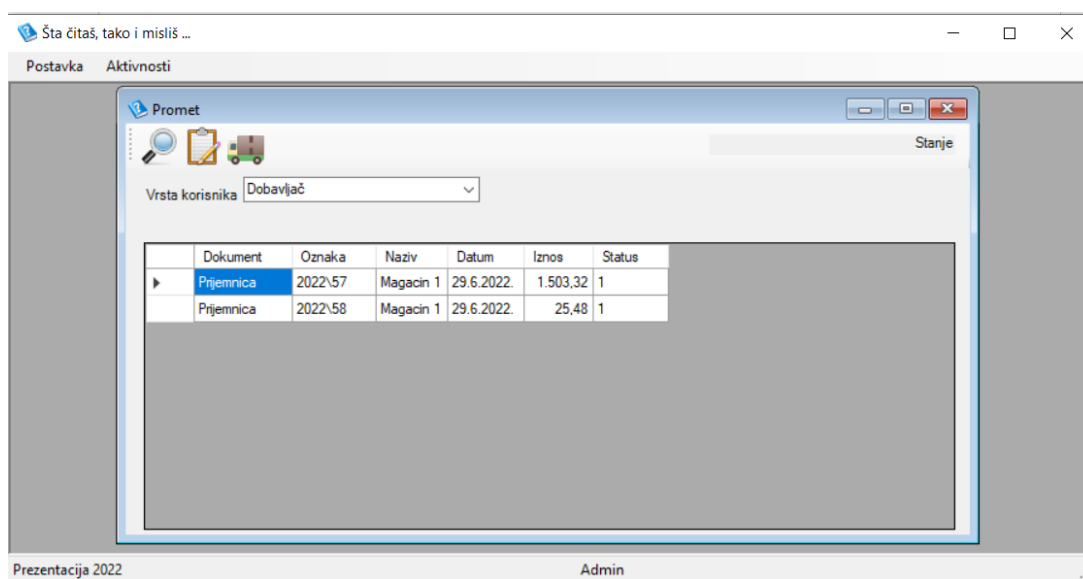
Добављач

Учесници СК

Добављач и систем (програм)

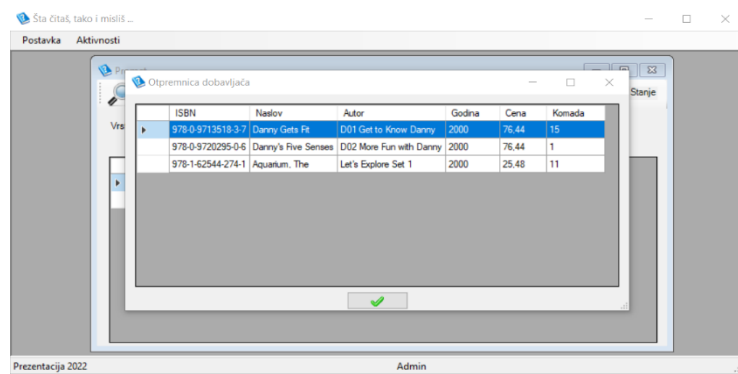
Предуслов: Систем је укључен и Добављач је улогован под својом шифром. Добављач бира опцију „Да ли има захтева за испоруком књига (пријемница)“. Систем враћа списак свих постојећих пријемница која су послате добављачу.

Основни сценарио СК



Слика 87 Списак свих пријемница послатих добављачу

1. **Добављач** бира жељену пријемницу за коју жели да креира **отпремницу** за складиште. (АПУСО)



Слика 88 Креирање отпремнице за складиште

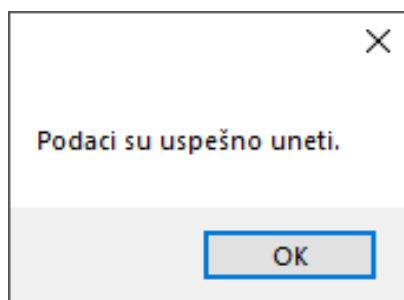
2. **Добављач** контролише да ли је коректно унео податке за **отпремницу**. (АНСО)

3. **Добављач** позива **систем** да запамти податке о креираној **отпремници**. (АПСО)

Опис акције: **Добављач** кликом на дугме „ОК“ позива системску операцију ObradaDokumenta (Dokument) и Promet (docid)

4. **Систем** памти податке о креираној **отпремници**. (СО)

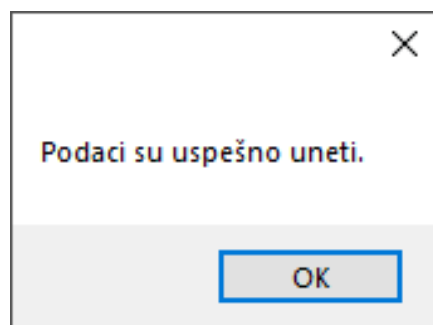
5. **Систем** приказује **добављачу** поруку „Подаци су успешно унети“. (ИА)



Слика 89 Успешно креирање отпремнице

Алтернативни сценарио 1

- 5.1 У случају да је **добављач** променио потребну количинну књига за **отпремницу** **систем** приказује **добављачу** поруку „Подаци су успешно унети“. (ИА)



Слика 90 Успешно мењање отпремнице

СК4: Случај коришћења – Креирање рекламације на основу комисијског записника

Назив СК

Креирање рекламације

Актери СК

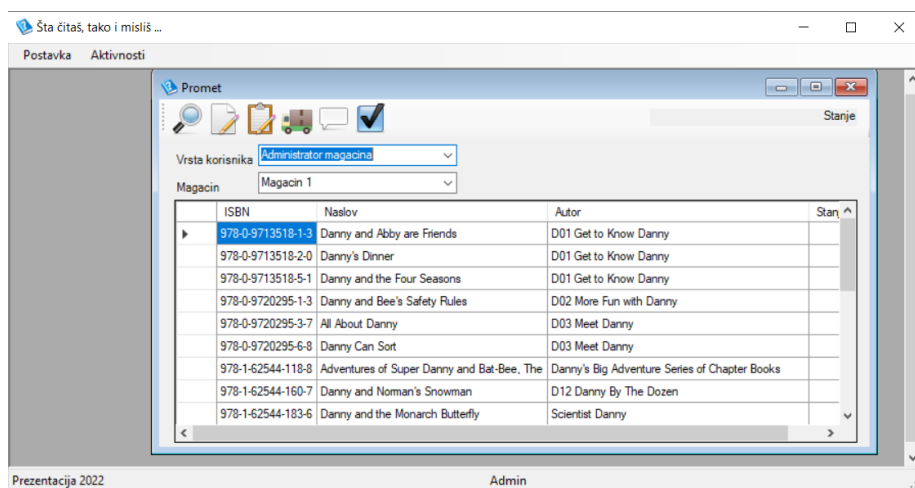
Администратор складишта

Учесници СК

Администратор складишта и систем (програм)

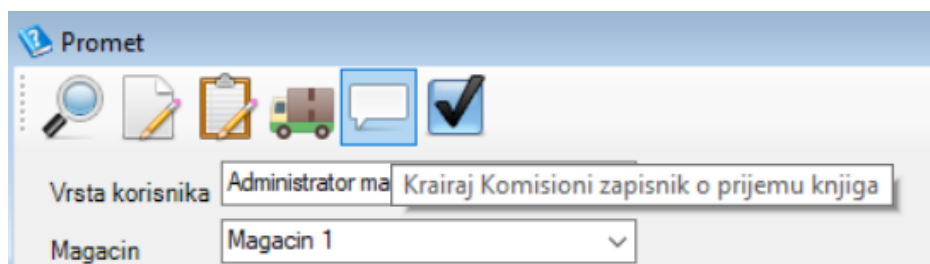
Предуслов: Систем је укључен и администратор складишта је улоган под својом шифром. Администратор складишта бира опцију „Креирање Пријемнице за добављача“. Систем враћа списак свих постојећих требовања која су извршена према датом складишту.

Основни сценарио СК



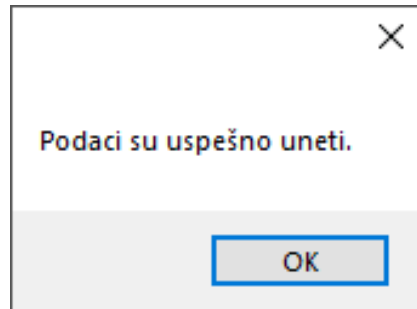
Слика 91 Списак постојећих требовања према датом складишту

1. Администратор складишта бира отпремницу коју жели комисијски да прими и за њу креира комисиони записник. (АПУСО)



Слика 92 Опција Креирај Комисиони записник о пријему књига

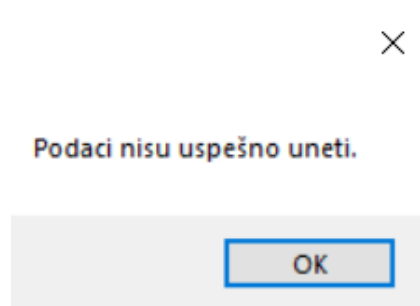
2. **Администратор складишта** контролише да ли је коректно унео податке за **комисиони записник**. (АНСО)
3. **Администратор складишта** **позива** систем да запамти податке о изабраним **књигама**. (АПСО)
Опис акције: **Администратор складишта** кликом на дугме „ОК“ позива системску операцију ObradaDokumenta(Dokument), Promet (docid) и AzurIznos (docid)
4. **Систем** **памти** податке о изабраним **књигама**. (СО)
5. **Систем** **приказује** **администратору складишта** поруку: „Подаци су успешно унети“. (ИА)



Слика 93 Успешно креирање записника

Алтернативни сценарио 1

- 5.1 Уколико подаци о **књигама** нису успешно унети, **систем** **приказује** **администратору складишта** поруку: „Подаци нису успешно унети“. (ИА)



Слика 94 Неуспешно креирање записника

СК5: Случај коришћења – Креирање доставнице на основу отпремнице добављача

Назив СК

Креирање доставнице

Актори СК

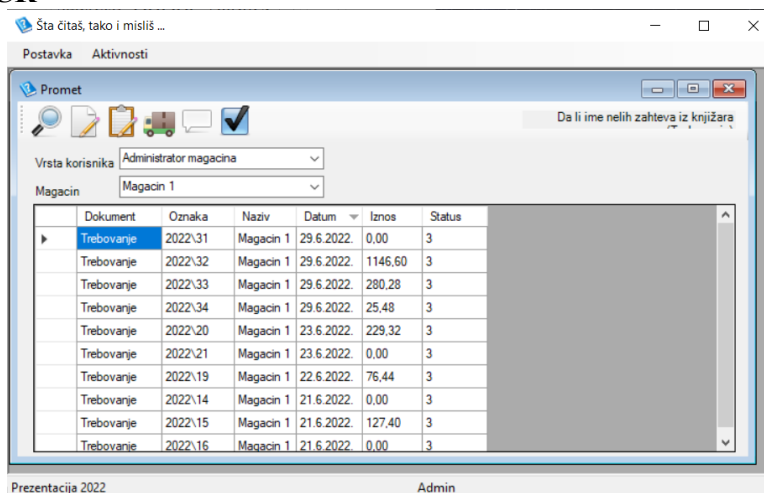
Администратор складишта

Учесници СК

Администратор складишта и систем (програм)

Предуслов: Систем је укључен и администратор складишта је улоган под својом шифром. Администратор складишта бира опцију „Да ли има захтева из књижара (требовања)“. Систем враћа списак свих постојећих требовања која су упућена према датом складишту.

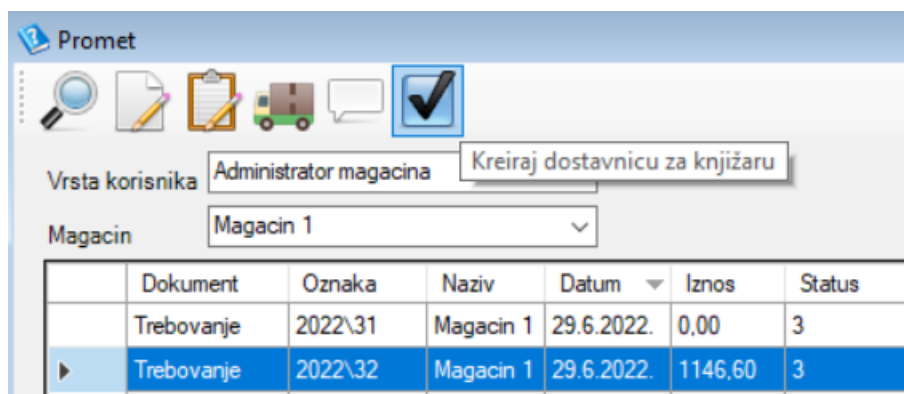
Основни сценарио СК



Dokument	Oznaka	Naziv	Datum	Iznos	Status
Trebovanje	2022\31	Magacin 1	29.6.2022.	0.00	3
Trebovanje	2022\32	Magacin 1	29.6.2022.	1146.60	3
Trebovanje	2022\33	Magacin 1	29.6.2022.	280.28	3
Trebovanje	2022\34	Magacin 1	29.6.2022.	25.48	3
Trebovanje	2022\20	Magacin 1	23.6.2022.	229.32	3
Trebovanje	2022\21	Magacin 1	23.6.2022.	0.00	3
Trebovanje	2022\19	Magacin 1	22.6.2022.	76.44	3
Trebovanje	2022\14	Magacin 1	21.6.2022.	0.00	3
Trebovanje	2022\15	Magacin 1	21.6.2022.	127.40	3
Trebovanje	2022\16	Magacin 1	21.6.2022.	0.00	3

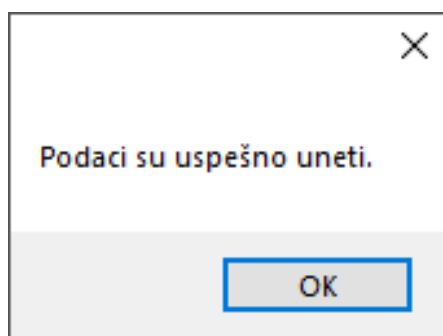
Слика 95 Списак свих требовања за дато складиште

1. Администратор складишта бира требовање за које жели да креира доставницу за књижару. (АПУСО)



Слика 96 Опција Креирај доставницу за књижару

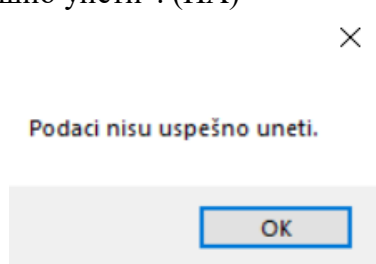
2. **Администратор складишта контролише** да ли је коректно унео податке за **доставницу**. (АНСО)
3. **Администратор складишта позива систем** да запамти податке о изабраној **доставници**. (АПСО)
Опис акције: **Администратор складишта** кликом на дугме „ОК“ позива системску операцију ObradaDokumenta (Dokument) и Promet (docid).
4. **Систем памти** податке на изабраној **доставници**. (СО)
5. **Систем аутоматски креира доставницу** за запосленог у књижари и **приказује администратору складишта** поруку: „Подаци су успешно унети“. (ИА)



Слика 97 Успешно креирање доставнице

Алтернативни сценарио

- 5.1 Уколико подаци нису успешно унети, **систем приказује администратору складишта** поруку: „Подаци нису успешно унети“. (ИА)



Слика 98 Неуспешно креирање доставнице

СК6: Случај коришћења – Пријем књига у књижару

Назив СК

Пријем Књига

Актори СК

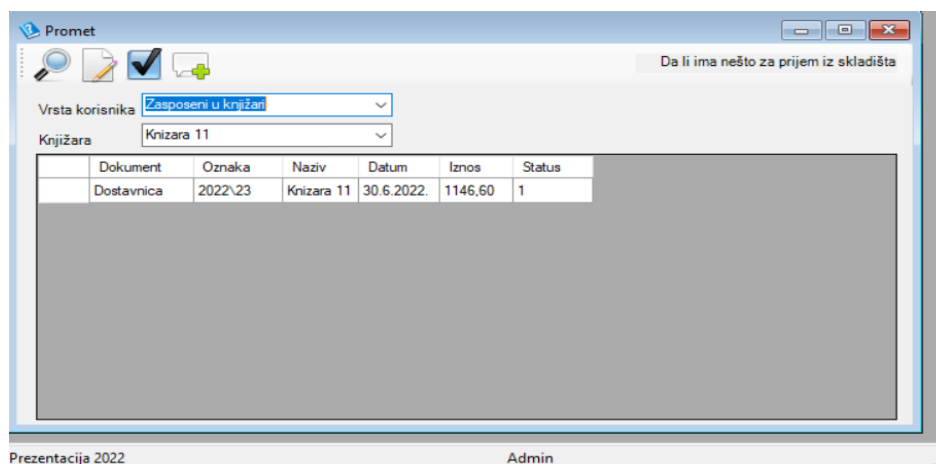
Запослени у књижари

Учесници СК

Запослени у књижари и систем (програм)

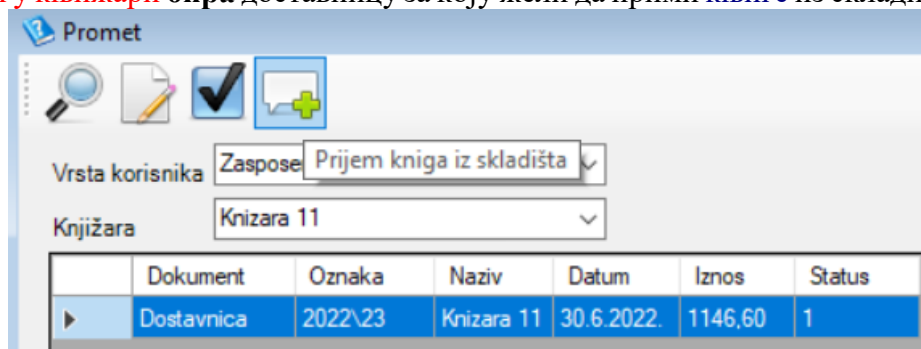
Предуслов: Систем је укључен и **Запослени у књижари** је улогован под својом шифром. **Запослени у књижари** бира опцију „Да ли има нешто за пријем из складишта“. Систем враћа списак доставница за које треба извршити пријем књига.

Основни сценарио СК



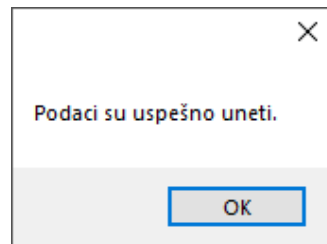
Слика 99 Списак доставница за које треба извршити пријем

1. **Запослени у књижари бира** доставницу за коју жели да прими књиге из складишта. (АПУСО)



Слика 100 Опција Пријем књига из складишта

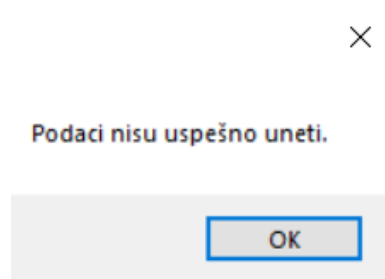
2. **Администратор складишта контролише** да ли је коректно унео податке за **књиге** које ће бити примљене. (АНСО)
3. **Запослени у књижари позива систем** да запамти податке о **књигама** које ће бити примљене у књижару. (АПСО)
Опис акције: **Запослени у књижари** кликом на дугме „ОК“ позива системску операцију ObradaDokumenta(Dokument), Povezi (docid,docid) и UpdateStatus (Dokument).
4. **Систем памти** податке. (СО)
5. **Систем приказује запосленом у књижари** поруку „Подаци су успешно унети“. (ИА)



Слика 101 Успешан пријем књига

Алтернативни сценарио

- 5.1 Уколико подаци нису успешно унети, **систем приказује запосленом у књижари** поруку: „Подаци нису успешно унети“. (ИА)



Слика 102 Неуспешан пријем књига

3.6 Тестирање

Покретањем апликације и уносом неисправних података, вршене су провере неправности валидација. Уношени су и правилни подаци, како би се тестирали сви приказани случајеви коришћења. Уколико је дошло до одређеног недостатка приликом тестирања, они су уклоњени и исправљени.

Након одређеног броја тестова, закључак је да апликација правилно функционише и испуњава све задате захтеве.

План за даље тестирање апликације је писање Unit тестова.

4. Закључак

Завршни рад *Развој софтверског система за вођење евиденције о пословању књижара и складиштењу књига применом .NET технологија* даје потпун приказ поступка развоја софтверског система. У њему је приказан заокружен систем који подржава функционисање више различитих књижара које поседују по једно главно складиште и добављача који је задужен за добављање књига.

Имплементирана је евиденција о књигама које се налазе у складишту, књигама које су продате другим књижарама, требовању књига из складишта, као и свим документима који се налазе у оквиру система. Такође значајан акценат је бачен на дизајн и сами изгледа корисничких форми како би софтверски систем био што лакши за коришћење.

Кроз три поглавља, описане су све технологије и значајни концепти који су коришћени у развоју софтверског система. Дат је приказ .NET платформе и свих њених технологија, програмског језика C# са битним концептима сокета и нити у објектно оријентисаном програмирању. Уз то су обрађени патерни који су коришћени у изради софтверског система. Такође је дат преглед релационе базе за управљање базом података – *SQL Server*.

За развој софтвера коришћена је упрошћена Ларманова метода која је дала систематичан приступ изради, при чему је омогућила да урађени софтверски систем у студијском примеру буде пропраћен адекватном документацијом. У изради је акценат стављен на прву **фазу развоја** – дефинисање корисничких захтева јер је то кључни елемент у развоју и треба уложити довољно време у ту фазу како касније не би дошло до евентуалних промена у захтевима, што би пропорционално узело много више времена у самом развоју. У раду је прво зато приказан вербални опис самог проблема, који је разрашћен кроз случајеве коришћења. **Фаза анализе и пројектовања** даље разрађују дефинисане захтеве и дају одговарајуће дијаграме који описују понашање система, логичку и физичку структуру. У **фази имплементације** је урађена реализација претходно дефинисаних компоненти система, које су тестиране мануелно у фази тестирања.

За сада овај систем представља једну заокружену целину у којем су на одговарајући начин имплементирани и тестирани сви случајеви коришћења. Тестирање је урађено мануелно, тако да би први корак у унапређивању стабилности апликације био тестирање кроз *Unit* тестове. Нове функционалности које би могле да прошире постојећи систем подразумевају продају књига крајњим купцима, као и доступност књига на другим језицима или могућност куповине онлајн.

5. Референта литература

- [1] (2020, Март 18). Desktop Application Development *Developex Blog*
<https://developex.com/blog/desktop-applications-development/>
- [2] Lander, R. (2019, Мај 6). Introducing .NET 5 *.NET Blog*
<https://devblogs.microsoft.com/dotnet/introducing-net-5/>
- [3] Microsoft. (2020, Октобар 22) Common Language Runtime (CLR) overview
<https://docs.microsoft.com/en-us/dotnet/standard/clr?redirectedfrom=MSDN>
- [4] Chiarelli, A. (2021, Фебруар 17). What is .NET? An Overview of the Platform *Auth0 blog*
<https://auth0.com/blog/what-is-dotnet-platform-overview/>
- [5] Sells, C. (2003, Септембар 6). Windows Forms Programming in C# (1st ed.). AddisonWesley Professional. p. xxxviii.
- [6] Microsoft. (2010, Јул 28). Design and Implementation Guidelines for Web Clients
[https://docs.microsoft.com/en-us/previous-versions/msp-np/ff647339\(v=pandp.10\)?redirectedfrom=MSDN](https://docs.microsoft.com/en-us/previous-versions/msp-np/ff647339(v=pandp.10)?redirectedfrom=MSDN)
[https://docs.microsoft.com/en-us/previous-versions/msp-np/ff647339\(v=pandp.10\)?redirectedfrom=MSDN](https://docs.microsoft.com/en-us/previous-versions/msp-np/ff647339(v=pandp.10)?redirectedfrom=MSDN)
- [7] Sells, C; Weinhardt, M. (2006, Мај 16). "Appendix B". Moving from MFC, Windows Forms 2.0 Programming (2nd ed.). Addison-Wesley Professional.
- [8] Martin, J. (2018, Децембар 4). Microsoft Open Sources WPF, WinForms, and WinUI. *InfoQ*. <https://www.infoq.com/news/2018/12/msft-open-source-wpf-winforms/>
- [9] Albahari, J., & Albahari, B. (2014). *C# 5.0 Mali referentni priručnik* (1st ed.). Mikro knjiga. <https://docplayer.net/53344411-C-5-0-mali-referentni-prirucnik.html>
- [10] Microsoft. (2021, јун 18). The history of C#
<https://docs.microsoft.com/en-us/dotnet/csharp/whats-new/csharp-version-history>
<https://docs.microsoft.com/en-us/dotnet/csharp/whats-new/csharp-version-history>
- [11] Albahari, J. (2011, април 27). Threading in C#. O'Reilly Media,
<http://www.albahari.info/threading/threading.pdf>

- [12] Влајић, С; Савић Д; Станојивећ В; Милић М.(2008) Пројектовање софтвера – напредне јава технологије
- [13] Blum, R. (2003) C# Network Programming
- [14] Microsoft. (2016, јун 20) LINQ overview
<https://docs.microsoft.com/en-us/dotnet/standard/linq/>
- [15] Oracle. What is Relational Database (RDMBS)?
<https://www.oracle.com/database/what-is-a-relational-database/>
- [16] Peterson, R. (2021, Август 27) What is SQL Server? Introduction, Version History.
<https://www.guru99.com/sql-server-introduction.html>
- [17] Kaisler, S., Armour, F. (2002). Design Patterns.
https://www.researchgate.net/publication/229807376_Design_Patterns
- [18] Gamma, E., Helm, R., Johnson, R.E., Vlissides, J.M. (1994). Design Patterns. AddisonWesley, Reading
- [19] Reenskaug, T; Coplien, J. O. (2009, Март 20). "The DCI Architecture: A New Vision of Object-Oriented Programming". Artima Developer.
https://web.archive.org/web/20090323032904/https://www.artima.com/articles/dci_vision.html
- [20] Davis, I.(2008, Децембар 9) What Are The Benefits of MVC? *Ian Davis Blog*.
<https://blog.iandavis.com/2008/12/what-are-the-benefits-of-mvc/>
- [21] (2012, Мај 11) Model–View–Controller History. *C2.com*.
<http://wiki.c2.com/?ModelViewControllerHistory>
- [22] Stencel, K., Wegrzynowicz, P. (2008). Implementation Variants of the Singleton Design Pattern
https://www.researchgate.net/publication/220830975_Implementation_Variants_of_the_Singleton_Design_Pattern
- [23] Vlajić, S. (2014). Softverski paterni. Zlatni Presek
- [24] Freeman, E., Robson, E., Bates, B., Sierra, K. (2004). Head First Design Patterns. O'Reilly Media.
- [25] Vlajić, S. (2020, Октобар 25). Projektovanje softvera (skripta). Beograd, Srbija, FON.