УНИВЕРЗИТЕТ У БЕОГРАДУ

ФАКУЛТЕТ ОРГАНИЗАЦИОНИХ НАУКА

ЗАВРШНИ РАД

Тема: Развој софтверског система за евиденцију робних палета у регалним складиштима применом .NET технологија

Ментор:

Проф. др Саша Лазаревић

Студент:

Ђорђе Врањеш 0155/2017

Београд, 2021. године

Садржај

Списак слика:

Списак табела:

1. Увод

Дигитализација је процес превођења једног објекта слике, звука, документа или сигнала у дигиталан облик. Држава Србија иако мала земља почела је да примењује технологију како би прешла са бирократског система пуног писања и књига у систем којем је све одређено кликом. Софтверски системи ће постати свакодневница за сваки начин пословања који захтева било какво вођење евиденције или координисање.

У великој количини привреда у Србији се директно односи на пољопривредне, месне и друге производе. Сви ти производи у већој или мањој количини не стижу одмах до крајњег купца, што због тражње или стратегије трговца. То значи да они морају бити складиштени по хладњачама, магацинима и другим облицима складишта. Ту наилазите на проблем на који начин спровести евиденцију свих придошлих и одлазних производа. Свесни смо да су изразито неисплатива складишта са малим простором, тако да треба бити свестан саме комплексности и величине складишта. Наилазите на податак да у Србији постоје преко 50 хладњача са просеком од хиљаду тона капацитета робе у својим регалима где свако место има максималну носивост од тону. Податак се односи искључиво на хладњаче, где се сходно томе поставља питање колико је заправо робе могуће складиштити у Србији.

У овом раду биће приказана десктоп апликација креирана уз помоћ .NET Frameworka коришћењем програмског језика C#, клијент-сервер архитектуре, софтверских патерна и Microsoft SQL Server-a.

Циљ овог рада јесте направити софтверски систем који ће обезбедити лаку евиденцију робних палета у регалним складиштима. Конкретно овај рад је рађен према хладњачи у којој се тренутно и примењује, али то не значи да није могуће користити у другим складиштима. Једина разлика је у самим позицијама које су различите за свако складиште понаособ. Хладњача према којој је изграђен софтверски систем поседује 2650 палетних места, односно максималну количину робе од 2650 килограма. Ови подаци се односе на хладњачу високог капацитета које чине 5 вертикала (спратова), 18 колона (A-S), 7 редова (0-7) и 4 палетна места. Како би се комплексност свела на што мањи њиво, уз софтверски систем је неопходно обезбедити и мапу која ће запосленима омогућити брз и једноставан рад. Једна од основних карактеристика оваквих система где је рад у тешким условима (хладњаче раде на температури и до -20 степени) потребна је тачност и поузданост у сваком тренутку. Овај софтверски систем то обезбеђује на изузетно високом нивоу уз помоћ свог једноставног интерфејса. Софтверско решење омогућава комплетну евиденцију за управника складишта да у сваком тренутку може да приступи целокупној историји једног палетног места у задњих пар година.

Рад је дефинисан кроз 16 поглавља.

2. Технологије развоја софтвера

У овом раду коришћене су .NET технологије. У наставку можете видети детаљније објашњење о самој платформи, као и неизбежним компонентама ове технологије.

2.1 .NET платформа

.NET (дот-нет) је име најмодерније платформе које Microsoft везује за софтверске технологије будућности. У називу је основна порука коју носи ова технологија – доступност у сваком тренутку на сваком месту. Ова платформа представља дугорочни и стратегијски план развоја не само у *Microsoft-u*. .NET даје реалне основе да постане основна платформа развоју модерних апликација. Радни оквир .NET развијен са циљем да обезбеди окружење за развој свих модерних апликација на Windows оперативном систему.

Једна од основних особина ове платформе је њена оријентација ка дистрибуираним апликацијама преко интернета. Ово са собом носи још предности које се не односе искључиво на дистрибуиране апликације:

- Дистрибуиране апликације су више објектно оријентисане
- Овај стил програмирања убрзава стварање колекције специфичног кода на једном месту, насупрот досадашњем стилу где се стварају редундантне копије на много места. Ово свакако повећава ефикасност и опредељује за овакве апликације;
- Овај тип апликације пружа софтверске целине расположиве различитим уређајима преко интерфејса;
- Контролисањем приступа у реалном времену (*real-time*) ка дистрибуираним чворовима (делови једне софтверске целине) могуће је лакше контролисање рад таквих апликација.
- Библиотека класа развијена је од самог почетка користећи вишегодишња искуства програмера на различитим платформама. Ово даје веома лепу особину, а то је добар дизајн, доследно и добро дефинисање основних типова;
- Подршка за Web (XML) сервисе. .NET има развијене алате за једноставно писање XML сервиса;
- Побољшани приступ динамичким Web страницама базиран на ASP.NET технологији;
- Ефикаснији приступ подацима преко ADO.NET класа;
- .NET поставља и нови приступ за заједничко коришћење кода. Насупрот традиционалних dll библиотека, уводи се концепт склопова (assembly);[1]

Заједничко функционисање различитих платформи императив је у дистрибуираним апликацијама. Зато је било неопходно обезбедити стандардизацију у размени података.

Сви језици који су обухваћени новим пакетом Microsoft Visual Studio .NET имају подршку за .NET класе. C# је посебно направљен нови програмски језик управо за .NET. Други програмски језици на овој платформи су постојали и пре настанка .NET. То овом програмском језику омогућава оптимално коришћење окружења.

[1] Редундантност је појам који се користи у техници када је реч о непотребном вишку или преобимности.

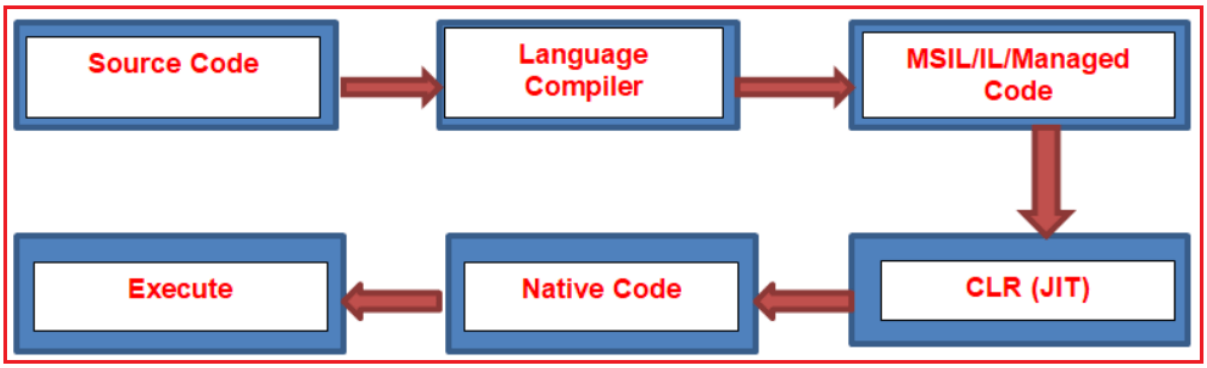
2.2 Компоненте .NET Framework-а

Две најзначајније компоненте .NET Frameworka су CLR (*Common Language Runtime*) и *.NET Framework Class Library*.

- *Common Language Runtime* (CLR) је извршни део који обезбеђује покретање апликације. Он је задужен за сервисе као што су *thread management*, *garbage collection*, *type-safety*, *exception handling.*
- *Class Library* обезбеђује сет API-ја и типова за нормалну функционалност. На тај начин могуће је применити типове података као што су: стрингови, датуми, бројеви и др. Исто тако укључује API-је за читање и писање докумената, конекцију са базама података, цртање и др.

.NET апликације су написане у C#, F#, или Visual Basic програмским језицима. Код је компајлиран у Common Inermediate Language (CIL). Компајлиран код је сачуван у асемблеру – фајловима са екстензијама као што су .dll или .exe.

Када се апликација покрене, Common Language Runtime (CLR) узима из асемблера и користи just-in-time compailer (JIT) како би претворио у машински код и потом извршио на тренутној архитектури на којој је рачунар покренут.



Слика 1 - Компоненте .NET Frameworka

3. C# програмски језик

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

C# је програмски језик који припада породици C језика, што практично значи да је његова синтакса слична синтакси C језика. С обзиром на то да овој породици припадају и језици C++ и Јава, може се рећи да постоји доста сличности између језика C# и ових језика. C# је најмлађи од споменутих језика, те је тако развијен по угледу на језике C,

C++ и Јава, као унапређена верзија, једноставнија за програмирање. C# је један од језика вишег нивоа.

3.1 Класе и методе

Основни механизам путем којег C# остварује принципе објективно оријентисаног програмирања су класе. Као објективно оријентисан програмски језик, захтева се постојање макар једног објекта, односно класе, која садржи код програма. Класа служи као контејнер за податке (као што су поља и константе) и функције.

Класу дефинише резервисана реч class, иза које следи идентификатор, односно име класе иза којег следе витичасте заграде. Унутар класе се дефинишу променљиве и функције који представљају њене чланове.

```
class ImeKlase
{
  //следе променљиве и методе који су чланови класе
}
```

Већина библиотека се налазе у простору под називом System. Да би се могло приступити том именском простору потребно је написати комплетно име објекта.

Методом описујемо понашања објекта у одређеној ситуацији и под неким одређеним условима. На тај начин обезбеђујемо функционалност објекта. Пре позива метода везаног за класу неопходно је помоћу конструктора направити одређени објекат на који ће се тај метод применити.

Метода класе је именовани блок наредби који се састоји из заглавља и тела метода. У заглављу наводимо повратни тип (ако метода не производи вредност коју враћа, наводимо резервисану реч void), затим име метода за којим следи у малим заградама списак параметара метода. За сваки параметар наводи се тип коме тај параметар припада као и име параметра. После заглавља у витичастим заградама наводимо тело метода које се састоји из одговарајућих наредби програмског језика C#.

```
//Navodimo tip metode i parametre koje metoda koristi
string IspisiIme( string ime, string prezime, int razred )
{
   //Povratna vrednost nase metode je string i njome ispisujemo zelenu recenicu
   return  " Ucenik " + ime +  " "  + prezime +  "je " + razred +  .razred .
}
```

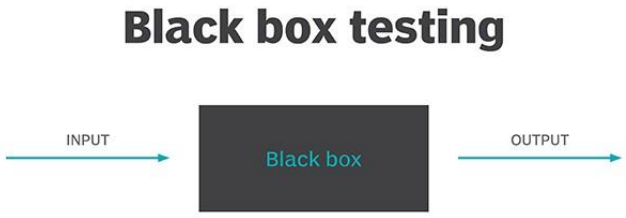
3.2 Наслеђивање, Енкапсулација, Полиморфизам

Једно од најбитнијих концепата објектно оријентисаног програмирања који се везују за C#:

- Енкапсулација

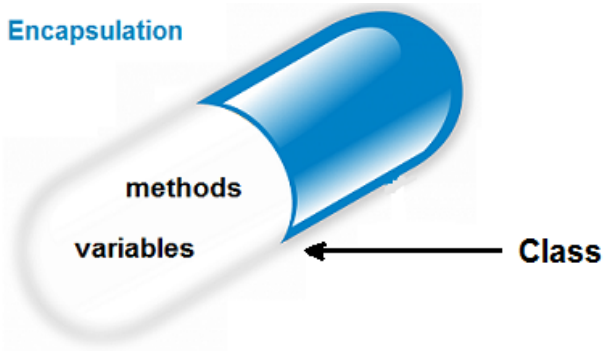
- Наслеђивање
- Полиморфизам

Прво је битно споменути и сам појам апстракције. Односно занемаривање небитних детаља у усредсређивање на битне. То се везује за појам “црне кутије”, који нам говори да можемо да користимо велики број објеката без да познајемо њихову унутрашњу структуру.



Слика 2 - Појам "црне кутије"

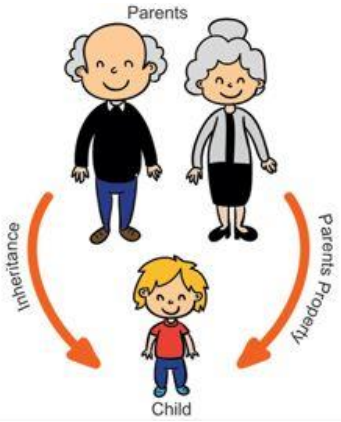
Учаурење или енкапсулација је другачија врста апстракције која подразумева скривање података од спољних приступа. Поштовањем енкапсулације обезбеђујемо да објекти имају строго контролисане улазе и излазе, а самим тим смањујемо могућност грешке, логичке недоследности или грешака у програму.



Слика 3 – Енкапсулација

Скривање података обезбеђујемо употребом кључне речи *private*. На тај начин нико ван те класе не може непланирано да промени податак.

Наслеђивање је такође један од основних концепата објектно-оријентисаног програмирања, представља механизам који обезбеђује да једна класа наследи све атрибуте и понашања неке друге (надређене) класе. Идеја је да није потребно да правимо нове класе са сличним атрибутима већ да наслеђујемо функционалност већ постојеће класе, али исто тако имамо право да променимо или додамо неке нова понашања.



Слика 4 - Наслеђивање

Дакле, различите класе могу имати нека заједничка својства која се, рецимо, чувају у некој општој класи коју онда могу делити разне класе. Класа која наслеђује неку другу класу назива се изведена класа (потклаcа), а класа чија се својства наслеђују суперкласа (наткласа).

Полиморфизам нам омогућава да се један исти позив методе понаша различито у зависности од типа објекта над којим се метод примењује. Када имамо променљиву типа базне класе, нпр. Возило, осим што у њој можете да чувате референцу на објекат те класе, исто тако можете чувати референцу на објекат произвољне класе нпр. Ауто, Авион, Мотор… То је једно од услова који мора бити испуњен како би полиморфизам функционисао.

```
Pas pas = new Pas();
Zivotinja zivotinja = pas;
Object obj = pas;
// primer polimorfizma
```

3.3 Апстрактне класе

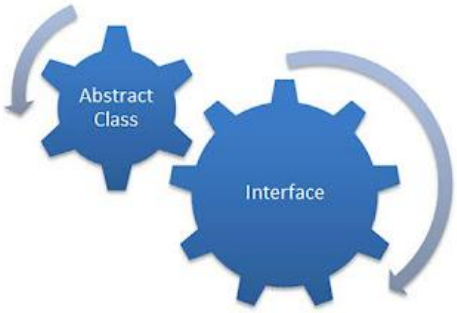
Свака класа која садржи кључну реч *abstract* и која има најмање једну апстрактну методу се назива апстрактна класа. Апстрактна метода је метода која садржи кључну реч *abstract* и која нема ни тело ни имплементацију. Она не може да се инстанцира док свака њена метода мора да буде имплементирана. Тада би добили закључак да апстрактну класу користимо када нисмо сигурни каква ће бити имплементација неке методе али са сигурношћу знамо да нам је потребна. Још један назив оваквих класа је да су оне непотпуне.

Апстрактне класе се користе и за безбедносне сврхе, односно да онемогуће иницијализацију класе.

3.4 Интерфејси

Познато је да класа не може истовремено да наслеђује више основних класа. Али зато може да имплементира више интерфејса истовремено. Док основна класа дефинише шта она представља, интерфејс ће да дефинише шта та класа ради. Интерфејс је најједноставније објаснити као уговор који комплетно одваја имена и потписе методе, пропертије, индексере, делегата и догађаје од њихове имплементације. Интерфејс не садржи поља и сличан је апстрактним класама, с тим да оне искључиво могу да имају само наведене потписе.

Могу се користити заједно са апстрактним класама да би се градила програмерска радна окружења која се проширују. Интерфејс, као и апстрактна класа се не могу инстанцирати. Још једном, најважнија собина интерфејса јесте да класе могу имплементирати више интерфејса, као и оне које позивају друге интерфејсе. Када се из једног интерфејса позива други интерфејс, онда се то назива ре-имплементација.



Слика 5 - Абстрактна класа и интерфејс

3.5 Изузеци

Грешке могу да се десе у било којој етапи извршавања програма и већина њих није изазвана грешком у нашој апликацији. Откривати, претпостављати, уочити или исправљати грешке у програму је често један од најтежих послова програмера и због тога је овај процес захтева пуно времена и тестирања. Што се пре открије нека грешка она ће вас мање коштати у даљем процесу развоја вашег пројекта.

Постоје три основна типа грешака:

- Синтаксне грешке које компајлер не може да разуме – оне се најлакше исправљају. Настају када у коду откуцате неправилно кључне речи или кад нешто заборавите да откуцате.
- Грешке у времену извршавања – јављају се кад се покуша извршити процес који се не може спровести. Једно од примера оваквих грешака јесте дељење нулом у неком изразу.
- Логичке грешке – најтеже се откривају. За њих не постоји никакав наговештај да грешка постоји док не видите да су очекивани резултати вашег програма погрешни. Они се решавају преко break режима и дебаговањем.

Изузеци су инстанце специјализованих класа које су све изведене из базне класе System.Exception. Изузеци укључују својства која олакшавају дијагнозу и руковање грешком. Својства која то дозвољавају су *Message*, које може да има читљив опис грешке као и било које друге важне информације о грешци. StackTrace, који садржи Trace – запис стека да би се омогућило праћење и прецизно проналажење места где се појавила грешка у коду. Уз помоћ изузетака можете да обезбедите средства за опоравак ваше апликације у случају неочекиваних грешака или да обезбедите да се подаци сачувају пре затварања апликације. Најпознатија конструкција која се и користи и која је и коришћена у овом раду је try-catch-finally исказ.

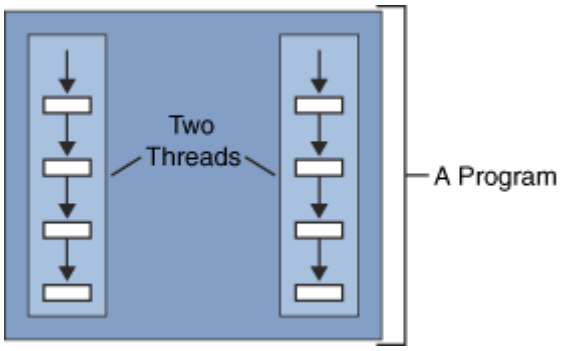
3.6 Нити

Сваки рачунар користи један CPU – *Central Processing Unit* - централни процесор, који у суштини обавља само једну операцију у датом времену. Због велике брзине извршавања стиче се утисак да се апликације извршавају истовремено. У том случају наилазите на проблем да уколико се појави нека грешка, за чије решавање процесору је потребно много више времена, јавља се смрзнута слика и делује као да је монитор

укочен. Да се ово не би дешавало превише често и да би се избегао овај проблем, створен је концепт нити.

Свака апликација покреће свој властити процес који је изолован и независан од других апликација. То значи да свака апликација формира своју нит. Нит је секвенца извршавања програма. Користећи нити ми извршавамо процесе без чекања да се претходни процес заврши. На тај начин се формира илузија да се извршава више нити истовремено. Код вишепроцесорских рачунара где имамо више процесора CPU, користи се паралелизам.

Уколико развијамо апликацију која ће омогућити приступ више корисника неком систему, тада је од изузетног значаја коришћење нити. Последица конкурентности може да буде да у истом тренутну оба корисника у систему желе исти ресурс. Једно од решења овог проблема јесте међусобно искључење. Међутим, међусобно искључење може довести до проблема попут мртвог закључавања (*deadlock*) и гладовања (*starvation*).



Слика 6 - Нити

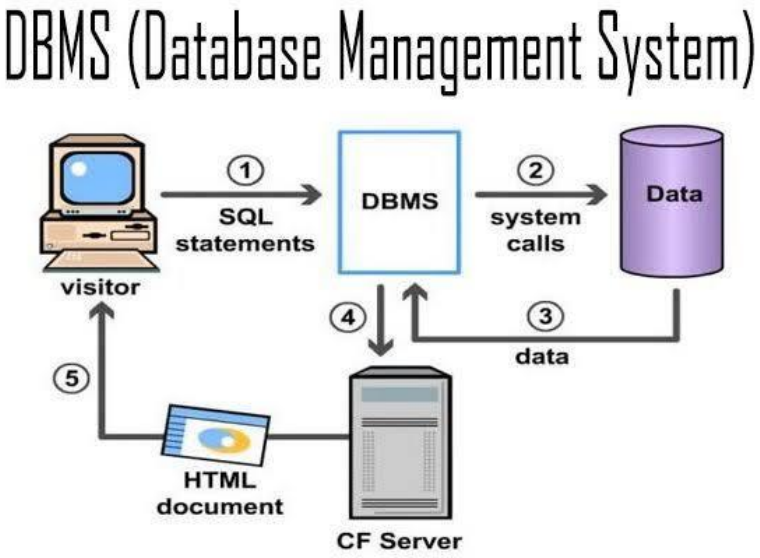
Deadlock је мултитаскинг проблем који настаје у ситуацији када два процеса заузму ресурсе и међусобно чекају један другог да ослободе те ресурсе. ***Starvation*** је такође мултитаскинг проблем који се јавља у ситуацијама када један процес заузме ресурсе и не дозвољава другим процесима да користе те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

Апликација која подржава рад са више нити је подржана од стране C# језика. Када програм крене да се извршава, он аутоматски креира једну нит која је заправо главна, док све остале нити се везују за кориснике које су пријављени на систем. Свака од нити ће бити опслужена од стране система, водећи рачуна да се не јаве проблеми мултитаскинга. Монитор је механизам који обезбеђује да уколико нека нит уђе у монитор, ниједна друга не може ући све док тренутна не изађе. На тај начин се постиже механизам који се назива синхронизација.

4. Sql Server

Microsoft SQL Server је софтвер који је намењен за управљање релационим базама података. Као што јој и само име каже, развијена је од стране Microsoft-а и сматра се врло скалабилним. Основно циљ је да опслужује корисника информацијама из релационих табела преко мреже на што једноставнији и ефикаснији начин.

Постојање Microsoft SQL Server-а има значајну функцију у свету када се односи на управљање базама података. Овај софтвер управља базама података користећи упите или SQL наредбе.



Слика 7 - Систем за управљање базама података

Предности Microsoft SQL Server-а:

- ради добро на свим верзијама Windows оперативним системима
- групише податке
- централизована контрола базе података
- могућност прављења сигурносне копије базе података

4.1 Sql упитни језик

SQL је упитни језик који омогућује писање упита над релационим базама података. Иако је на почетку свог развоја био прилично једноставан, близак кориснику и у великој мери декларативан, данас се сматра комплексним, процедурално декларативним језиком.

Закључно са SQL-92 стандардом SQL наредбе су сврстане у једну од следеће 3 категорије:

- наредбе за дефинисање података (data definition statements)
- наредбе за манипулисање подацима (data manipulation statements)
- наредбе за контролне функције (data control statements)

Наредне за дефинисање података омогућавају дефинисање објеката базе. Примери наредби ове категорије су:

- CREATE TABLE (креирање табеле базе података)
- CREATE VIEW (креирање виртуелне табеле – “погледа”)
- CREATE INDEX (креирање индекса над комбинацијом колона табеле)
- ALTER TABLE (измена дефиниције табеле)
- DROP TABLE (избацивање табеле из базе података)

Наредбе за манипулисање (руковање) подацима омогућавају ажурирање и приказ података базе:

- SELECT (приказ садржаја релационе базе података)
- UPDATE (измена вредности колона табеле)
- DELETE (избацивање редова табеле)
- INSERT (додавање редова постојеће табеле)

Наредбе за контролне (управљачке) функције омогућују опоравак, конкурентност, сигурност и интегритет релационе базе података:

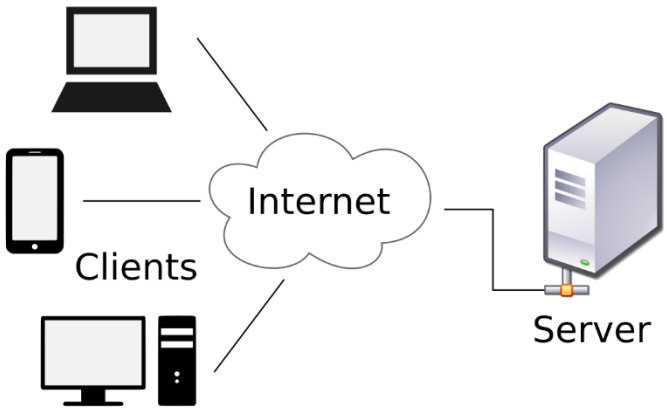
- GRANT (додела права коришћења сопствене табеле другим корисницима)
- REVOKE (одузимање права коришћења сопствене табеле од других корисника)
- COMMIT (пренос дејстава трансакције на базу података)
- ROLLBACK (поништавање дејстава трансакције)

5. Клијент сервер архитектура

У клијент-сервер архитектури, серверска компонента је она која пружа услуге већем броју клијентских компонената. Клијентске компонента захтевају услуге сервера које он опслужује. Сервери су стално активни и они представљају компјутере који могу специјализовано бити намењени серверима, али и не мора. Једна од њихових основних задатака јесте да ослушкују да ли има захтева од клијената. Захтеви се шаљу комуникационим каналима који зависе од машина на којима се сервер и клијент налазе.

Типични примери клијент-сервер архитектуре су апликације са удаљеним приступом базама података (клијентска апликација захтева услугу од сервера базе података), удаљени фајл системи (клијент приступа датотекама на серверу и у локалу), или веб апликација (захтева податке од веб сервера).

Приспели захтеви бивају опслужени у засебним нитима на серверу. Познато је да приликом покретања програма се стартује главна нит и касније свака нит за новог корисника софтверског система. У комуникацији често има “празног хода” који настаје како због саобраћаја тако и због трансформације резултата у различитим форматима. У дистрибуираним системима са више сервера, мора се обезбедити транспарентност, што значи да клијент не треба да разликује сервере.



Слика 8 - Клијент сервер архитектура

5.1 TCP протокол

Под термином TCP (*Transmission Control Protocol*) протокол мисли се на поуздан пренос података са успостављањем везе који се користи на транспортном слоју интернета. Познато је да једна од првих ствари које је потребно дефинисати приликом развоја софтверског система је да ли ћете користити UDP (User Datagram Protocol) или TCP протокол. TCP протокол размењује контролне информације транспортног слоја између клијента и сервера пре него што почне проток порука на апликативном нивоу. За овај протокол се каже да он успоставља везу, зато што пре почетка слања података из једне апликације у другу, та два процеса морају прво "да се рукују" – тј. мора морају један другом да пошаљу уводне сегменте. TCP протокол се извршава само у крајњим системима, а не на успутни елементима мреже (рутери и комутаторима слоја везе). Протокол обезбеђује да веза између рачунара А и Б може да се обавља у оба смера у исто време.

Сегменти овог протокола се састоје од поља заглавља и поља података. Поље података садржи одсечак података апликације. Код заглавља је то мало комплексније. Оно садржи бројеве изворног и одредишног порта, поље контролног збира, поље опција, поље ознака, поље дужине заглавља, пријемног прозора и поље редног броја и потврде пријема.

5.2 UDP протокол

За UDP се сматра да је поједностављени транспортни протокол који нуди најосновније услуге. Остварује се без успостављања везе, па нема усаглашавања између два процеса пре почетка размене података. UDP не нуди услугу поузданог преноса података, не постоји гаранција да ће порука заиста стићи до пријемног процеса. Поред тога што стигну, не постоји потврда да су стигле у тачном редоследу.

Протокол нема механизам за контролу загушења, тако да предајна страна UDP протокола може да убацује податке у слој испод себе брзином коју жели. Овај протокол најчешће користе апликације које захтевају велику брзину преноса нпр. апликације за видео стримовање (npr. *Skype*).

6. Софтверски патерни

Познато је да сваки патерн описује неки проблем који се стално понавља на различите начине и помажемо да на што ефикаснији начин заобиђемо исте. Решење се код патерна може применити за скуп сличних проблема који покривају неку класу проблема. Сматрано је да документовањем ових патерна може помоћи људима и олакшати налажење решења у сличној проблемској ситуацији. Корени патерна се налазе у радовима Кристофера Александра (*Cristopher Alexander*), он је дао следећу дефиницију патерна:

"Сваки патерн је троделно правило, које успоставља релацију између неког проблема, његовог решења и њиховог контекста. Патерн је у исто време и ствар, која се дешава у стварности, и правило које говори када и како се креира наведена ствар*"*.
[Alexander Cristopher, 1974]

Софтверски патерни се могу класификовати на следеће патерне:

- Тронивојски патерн – Разликујемо три ниова апстракције којима могу бити придружени неки од патерна. Први ниво се односи на идиоме (најнижи ниво апстракције), што значи да зависе од имплементационе технологије – нпр. Програмског језика. Патерни пројектовања су независни од конкретне имплементационе технологије, али нису довољно сложена да би се за њу рекло да је подсистем. Оквири спадају у трећи ниво апстракције и сматрају патернима системског нивоа. Оквир обезбеђује опште решење, а различите апликације га користе и проширују својим специфичностима.
- Анти-патерни – Патерни који су се показали као лоши у пракси.
- Мета патерни – Имају структуру тако да омогућавају да се из њих направе други патерни.

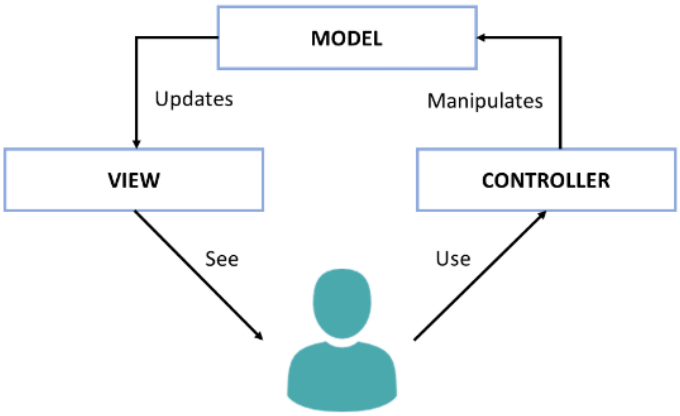
У основи сваког софтверског система налази се архитектура. Она се састоји од компоненти које се између себе повезане преко њихових интерфејса. Разликујемо макро и микро архитектуру. Макро архитектура је реализована преко ECF (*Enterpraise Component Framework*) или MVC (*Model-View-Controler*). Док за микро архитектура је реализована преко узора пројектовања (креационих, структурних и патерна понашања).

Три кључна елемента дефинишу патерн: проблем, решење и контекст. Контекст дефинише софтверски систем и његова ограничења морају бити задовољена кад се даје решење неког проблема. Као најбитнија особина патерна се издваја да се може применити као решење за више различитих проблема.

6.1 MVC патерн

Спада у макроархитектурне патерне који деле софтверски ситем на три дела, где на тај начин сваки део носи засебну улогу. Основна улога овог патерна је да омогући лаке измене неког елемента, без крупних интервенција у коду. Та три дела су:

- Модел (model) – модел садржи главне програмске податке, као што су информације о објектима из базе података. Састоји се од скупа класа које моделирају и подржавају решавање проблема. Сматра се централним делом апликације који садржи променљиву структуру података. Она директно управља подацима и логиком о правилима апликације.
- Приказ (view) – везује се за клијентски део апликације и директно је у контакту са самим корисником софтверског система. Он обезбеђује кориснички интерфејс, помоћу којег корисник уноси податке и позива одговарајуће системске операције. Најбоље речено, приказује кориснику стање модела.
- Контролер (controller) – основна улога контролера јесте да глуми “диспечера” између приказа и самог модела. Он ослушкује и прихвата захтеве од клијента за извршавање операција, након тога их позива у дефинисаном моделу. Уколико дође до промене стања, он је у обавези да обавести приказ да је дошло до промене.

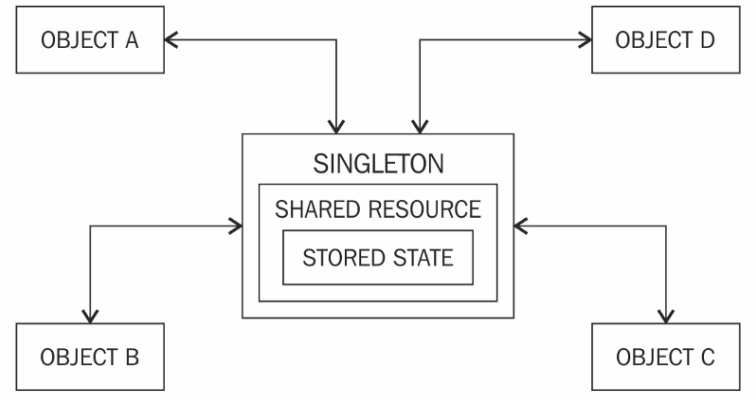


Слика 9 - MVC патерн

6.2 Singleton патерн

Уз помоћ овог патерна се обезбеђује да класа има само једну инстанцу. Најчешће се користи када желите да та једна инстанца координише акције у софтверском систему. На тај начин се генерализује и цео систем постаје ефикаснији. Постоји могућност да се ограничи на одређени број инстанци, односно не мора да буде нужно један објекат.

```
1.  public sealed class Singleton1 {
2.      private Singleton1() {}
3.      private static Singleton1 instance = null;
4.      public static Singleton1 Instance {
5.          get {
6.              if (instance == null) {
7.                  instance = new Singleton1();
8.              }
9.              return instance;
10.         }
11.     }
12. }
```



Слика 10 - Singleton патерн

7. Безбедност мреже

Када се успостави комуникација између две особе, акценат је стављен на тајност и интегритет комуникације. Тајност података подразумева да су информације доступне само овлашћеним лицима. Заштитом интегритета података обезбеђује се целовитост и тачност поруке која се преноси, односно спречава се могућност промене поруке. Безбедност мреже искључиво зависи од безбедности података, где се под ти термином мисли на стање заштићености од различитих претњи, приликом њиховог пријема, обраде, чувања, преноса и коришћења.

Пожељне особине безбедне комуникације су :

- Поверљивост – само пошиљалац и прималац коме је порука намењена треба да разумеју садржај пренете поруке. То изискује да порука буде шифрована тако да пресретнуту поруку, прислушкивач не може да разуме.
- Интегритет поруке – да није дошло до модификација поруке. То се обезбеђује техником контролног збира.
- Провера аутентичности крајњих тачака – пошиљалац и прималац би требало да могу да се увере у идентитет друге стране комуникације. Када учесници комуникације размењују поруке преко медијума, провера идентитета није једноставна. Она се заобилази различитим аутентификацијама, као што је нпр. двофакторска аутентификација.
- Безбедност у свакодневном раду – све организације су повезане са јавним интернетом. Што их чини рањивим на разне врсте сајбер напада. Једно од начина да се одбране од таквих напада су мрежне баријере (*firewall*) и системи за откривање уљеза.

Начин на који је имплементирана безбедност мреже у самом дипломском се односи на криптовање шифре приликом пријаве. Наиме, циљ је да се приликом комуникације и провере тачности шифре које магационер шаље серверу створи сигурност и немогућност да уљез сачека тај податак. То се заобилази алгоритмима за криптовање.

7.1 Принципи криптографије

Криптографске технике обезбеђују пошиљаоцу да маскира податке, тако да уљез не може да добије никакву информацију из пресретнутих података. Замислисте ситуацију где особа А шаље поруку особи Б. Порука коју шаље особа А је позната као отворен или јасан текст. Особа А мора да шифрује своју поруку користећи алгоритам за шифровање, тако да сваком уљезу шифрован текст буде неразумљив. Проблем настаје када су свима познате технике шифровања, па и самом уљезу који има лоше намере. Овај проблем се заобилази увођењем кључева који се користе у самим алгоритмима за шифровање. Алгоритам за шифровање узима кључ и поруку са отвореним текстом као улаз и од њих прави шифрован текст. Слично томе особа Б обезбеђује свој кључ који ће се користити за дешифровање шифрованог текста и враћање у првобитан смислен текст. Разликујемо две врсте кључа:

- Систем са симетричним кључем – Особа А и особа Б имају истоветне и тајне кључеве. Ова врста кључа се јављала кроз историју као Цезарова шифра. Када говоримо о модерним начинима, разликујемо две опште врсте техника: проточно

шифровање и блоковско шифровање. Блоковско шифровање претвара текст који треба да се шифрује у блокове од по k битова. Сваки тај одсечак се обрађује у табелама које су добијене функцијама које симулирају случајно пермутовање. Када се добију засебни излази, од сваког тог одсечака се формира комплетан излаз. Једно од популарних алгоритама за блоковска шифровања су: DES (Data Encryption Standard), 3DES и AES (Advanced Encryption Standard).

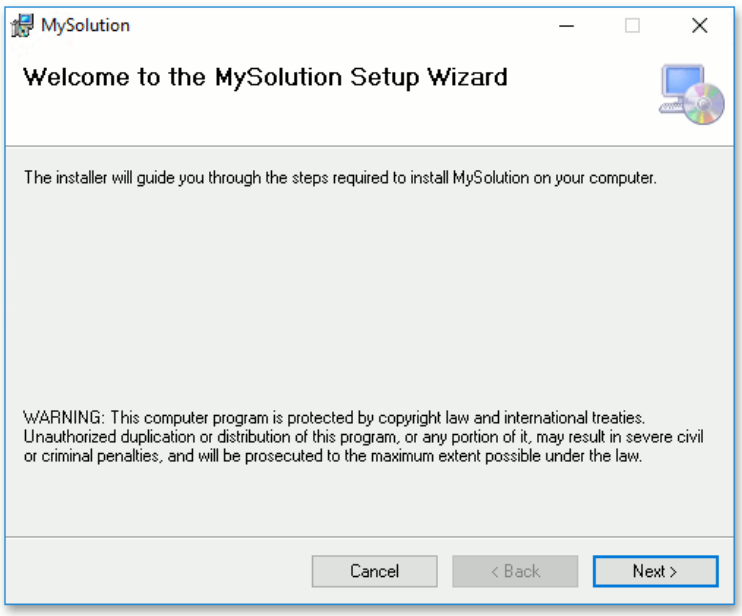
- Систем са јавним кључем – Проблем који настаје са симетричним кључем се огледа у обавезној размени кључа пре почетка комуникације. То је у ранијим периодима сматрано нормалним, док сада где се све одвија у умреженом свету велика је могућност да се особе које комуницирају никада не сретну и разговарају, изузев преко мреже. Ова врста криптографије користи пар кључева од који је један јаван, односно свима доступан и приватни кључ који се везује само за једну особу. Замислите да особа А хоће да комуницира са особом Б. На почетку комуникације, особа А је свесна јавног кључа особе Б, што користи као кључ за шифровање отвореног текста. Док са друге стране особа Б користи свој приватни кључ, како би дешифровала текст у смислен и њој јасан формат. Једно од битних делова који онемогућује да било ко пошаље шифровану поруку особи Б је дигиталан потпис.

8. Инсталација програма на серверском и клијентском рачунару

Када се донесе одлука за развој софтверског система са клијент-сервер архитектуром, поставља се питање на који начин инсталирати. Наиме, сведоци смо да кроз школовање радимо пројекте које у већем делу су намењени искључиво за школу. Знамо да он ради приликом покретања програма из алата у који је рађен, али никада се нећете сретнути са самим начином пуштања апликације у продукцију. Као што је речено, највиши утицај код пуштања софтверског система има сама његова архитектура.

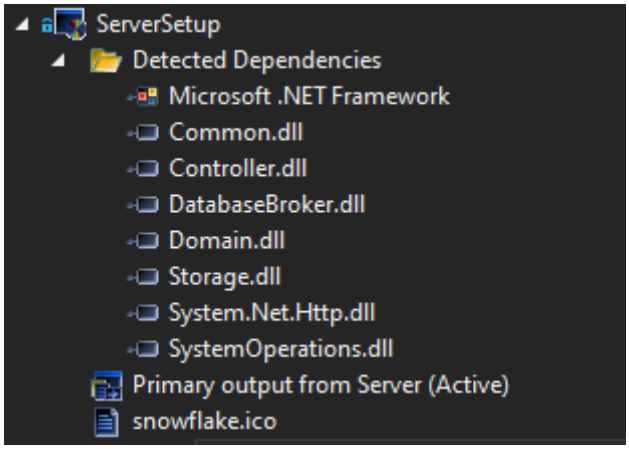
Рецимо да се говори о клијент-сервер апликацији која је рађена у овом раду и њеном начину инсталације засебно на клијентском и серверском рачунару. Велики број људи мисли да се серверски рачунари посебно нешто разликују од клијентског. Истина је да могу, али то није неопходно. Приликом развоја софтвера обезбедили смо архитектуру тако да постоје два засебна дела апликације. Један ће бити директно намењен клијенту и он се везује за његов приказ, односно кориснички интерфејс. Други део се налази са серверске стране.

Потребно је да обезбедимо посебну инсталацију за серверски део апликације, посебно за клијентски. У том случају најједноставије је користити алата под називом *Visual Studio Setup Project*. Након инсталације, поставља се питање како повезати та два рачунара да комуницирају. Велику улогу у овом делу имају *config* фајлови, који се аутоматски добијају приликом инсталације. Једна од лоших страна десктоп апликација јесте да оне раде у локалу. То значи да морате обезбедити да се клијентски и серверски рачунар буду под истом адресом излаза односно мреже. Након тога је потребно подесити порт и адресу рачунара, наиме упарити уређаје како би имали несметану комуникацију на заједничком порту.



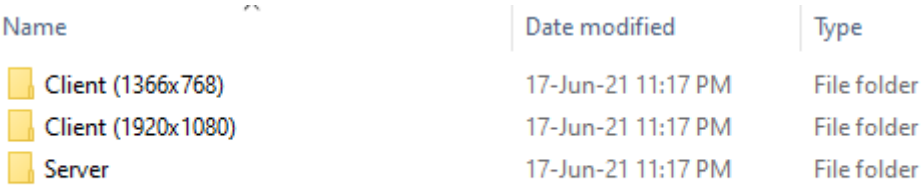
Слика 11 - Инсталација клијент сервер апликације

Након завршене инсталације добијамо засебни пројекат у Visual Studio са следећим библиотекама:



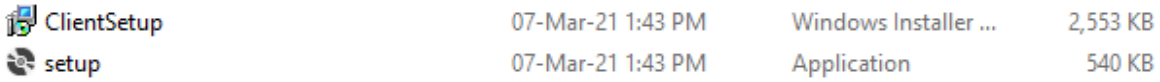
Слика 12 - Setup пројекат

На слици 13. је приказан пројекат који се односи на серверски део апликације, док на исти начин добијате и клијентски. Сада постоји могућност да инсталирамо пројекат на други рачунар. Проблем на који се наилази се односи на резолуцију монитора која варира у зависности од клијента. Једно од решења јесте обезбедити посебну инсталацију за сваки клијентски рачунар. Изглед са више инсталација према клијенту можете видети на слици 14.



Слика 13 - Различите инсталације

Изглед самих фолдера је сличан на први поглед, док заправо садрже потпуно раздвојене апликације које требају да међусобно комуницирају и на тај начин оправдају клијентско-серверску архитектуру.



Слика 14 - Инсталација

9. Студијски пример развоја софтверског система

Фазе које су коришћене у развоју софтверског система се везују за Ларманову методу. Ларманова метода користи стратегију управљања према случајевима коришћења и објектно-оријентисану методу пројектовања софтвера. Фазе развоја према Ларману:

- Спецификација захтева
- Анализа захтева
- Пројектовање
- Имплементација
- Тестирање

Спецификација или прикупљање захтева – одређујемо захтеве, односно функционалности које ће наш систем да има. На тај начин дефинишемо модел случаја коришћена који ће задовољити наше будуће кориснике. Случај коришћења (SK) описује скуп сценарија, односно скуп жељених коришћења система од стране актора. Под актором се сматра сваки спољни корисник који је у интеракцији са системом.

Анализа захтева – фаза анализе описује логичку структуру и понашање софтверског система. Описано је уз помоћ системских дијаграма секвенци које се праве за сваки сценарио случаја коришћења и уговора о системским операцијама. Резултат фазе анализе је пословна логика.

Фаза пројектовања – овде се описује физичка структура и понашање софтверског система. Обухвата пројектовање апликационе логике, складишта података и корисничког интерфејса. У оквиру апликационе логике се пројектују контролер, пословна логика и брокер базе података. Пословна логика обухвата пројектовање логичке структуре и понашања софтверског система.

Фаза имплементације – намењена је програмерима, где се очекује све претходно дефинисане ствари да буду примењене и коришћене прилико развоја софтверског система. Имплементира се у програмском језику, заједно са свим додатним алатима за које се договоре програмери.

Тестирање – финална фаза развоја софтверског система. Она се може поделити у неколико независних јединица тестирања. Тестирање подразумева прављење тест случаја коришћења, тест процедура (описују како ће се тест завршити) и тест компоненте које треба да аутоматизују тест процедуре.

10. Фаза прикупљања корисничких захтева

Захтеви представљају својства и услове које софтверски систем гледа да задовољи. Постоје различити типови захтева које систем мора да задовољи и они се категоризују као функционални и не функционални захтеви. Функционални захтеви дефинишу захтеване функције система, док не функционални захтеви дефинишу остале захтеве. У том смислу не функционални захтеви представљају атрибуте квалитета софтверског система.

10.1 Вербални опис

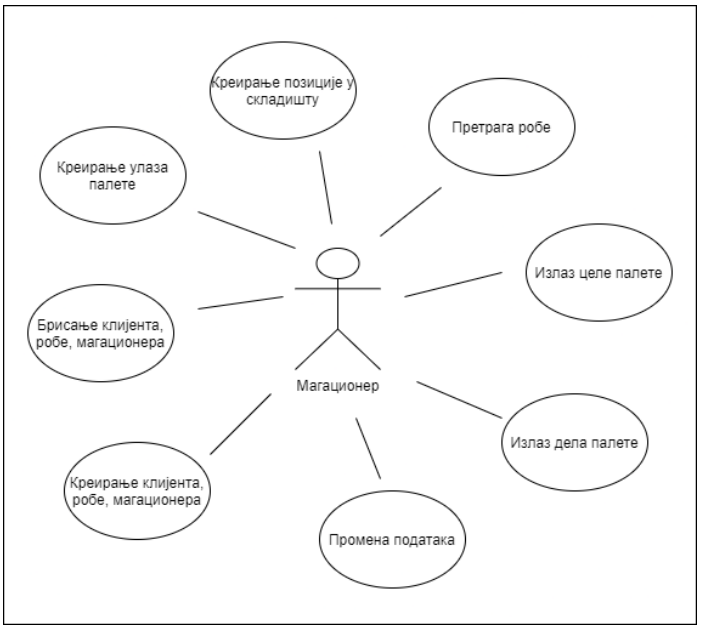
Потребно је направити апликацију која ће водити евиденцију о уласку/изласку робе у великим складиштима (царинска складишта, хладњаче, магацини и др.). У апликацији ће се водити евиденција палета са робом на складишним регалима као и њена сама позиција у складишту.

На почетку, систем омогућава унос самог садржаја палете (ручно/бар-код читач), који систем у сарадњи са корисником региструје и даје избор слободних позиција у складишту. Постоји могућност уноса различитих филтера које корисник самостално дефинише приликом одабира позиције. Доступно је претраживање робе, креирање улаза, као и излаз целе или делова палете. Систем омогућава вођење складишта уз помоћ јединственог интерфејса који приказује чак 2650 палетних места.

10.2 Модел случаја коришћења

У овој апликацији, идентификовано је осам случајева коришћења:

1. Креирање улаза палета (сложен случај коришћења)
2. Креирање позиције у складишту (сложен случај коришћења)
3. Претраживање робе по позицији, клијенту, роби
4. Излаз целе палете
5. Излаз дела палета
6. Промена података
7. Креирање клијента, артикла, магационера
8. Брисање клијента, артикла, магационера



Слика 15 - Случајеви коришћења

10.3 Спецификација случаја коришћења

Сваки случај коришћења се састоји од основног и алтернативног сценарија. Поред тога потребно је укључити актере софтверског система и све предуслове који морају бити испуњени како би дошло до извршавања случаја коришћења.

10.3.1 СК1: Случај коришћења- Креирање улаза палета

Назив СК:
Креирање улаза палете

Актори СК:
Магационер

Учесници СК:
Магационер и систем (програм)

Предуслови:
Систем је укључен и магационер је пријављен под својом шифром. Систем приказује форму за рад са улазом палета у складиште. Учитана је листа свих клијената са својим подацима.

Основни сценарио СК:

1. Магационер уноси податке у улаз. (АПУСО)
2. Магационер контролише да ли је коректно унео податке о улазу. (АНСО)
3. Магационер позива систем да запамти податке о улазу палета. (АПСО)
4. Систем памти податке о улазној палети (СО)

5. Систем приказује магационеру запамћени улаз и поруку: “Систем је запамтио улаз“. (ИА)

Алтернативна сценарија:

5.1 Уколико систем не може да креира улаз он приказује магационеру поруку: “Систем не може да креира улаз”. Прекида се извршење сценарија. (ИА)
5.2 Уколико систем не може да запамти податке о улазу палета он приказује магационеру поруку “Систем не може да запамти улаз”. (ИА)

10.3.2 СК2: Случај коришћења – Креирање позиције у складишту

Назив СК
Креирање позиције

Актори СК
Магационер

Учесници СК
Магационер и систем (програм)

Предуслов:
Систем је укључен и магационер је пријављен под својом шифром. Систем приказује форму за рад са позицијом. Учитани су подаци о улазу.

Основни сценарио СК

1. Магационер **позива** систем да креира позицију. (АПСО)
2. Систем **креира** позиције. (СО)
3. Систем **приказује** магационеру позицију. (ИА)
4. Магационер **уноси** податке о позицији. (АПУСО)
5. Магационер **контролише** да ли је коректно унео податке о позицији. (АНСО)
6. Магационер **позива** систем да запамти податке о позицији. (АПСО)
7. Систем **памти** податке о позицији. (СО)
8. Систем **приказује** магационеру запамћену позицију и поруку: “Систем је запамтио позицију“. (ИА)

Алтернативна сценарија:
3.1 Уколико систем не може да креира позицију он приказује магационеру поруку: “Систем не може да креира позицију”. Прекида се извршење сценарија. (ИА)
8.1 Уколико систем не може да запамти податке о позицији он приказује магационеру поруку “Систем не може да запамти позицију”. (ИА)

10.3.3 СК3: Случај коришћења – Претраживање робе по палетама

Назив СК
Претраживање робе

Актори СК
Магационер

Учесници СК
Магационер и систем (програм)

Предуслов:
Систем је укључен и магационер је пријављен под својом шифром. Систем приказује форму за претраживање робе. Учитана је листа позиција са свом робом.

Основни сценарио СК:

1. Магационер **уноси** критеријум по којој претражује робу. (АПУСО)
2. Магационер **позива** систем да нађе робу по задатом критеријуму. (АПСО)
3. Систем **тражи** робу по задатом критеријуму. (СО)
4. Систем **приказује** магационеру податке о роби. (ИА)
5. Магационер **одабира** жељену робу. (АПСО)
6. Систем **тражи** робу на основу одабира магационера. (СО)
7. Систем приказује магационеру податке о изабраној роби и поруку: "Систем је нашао робу ". (ИА)

Алтернативна сценарија:

4.1 Уколико систем не може да нађе робу он приказује магационеру поруку: "Систем не може да нађе робу по задатом критеријуму". Прекида се извршење сценарија. (ИА).

7.1. Уколико систем не може да нађе робу он приказује магационери поруку: "Систем не може да нађе робу по задатом критеријуму". (ИА)

10.3.4 СК4: Случај коришћења – Излаз целе палете

Назив СК
Промена излазне палете

Актори СК
Магационер

Учесници СК
Магационер и систем (програм)

Предуслов:
Систем је укључен и Магационер је пријављен под својом шифром. Систем приказује форму за рад са излазом целе палете. Учитана је листа свих палета са својим позицијама које се налазе у систему.

Основни сценарио СК

1. Магационер **уноси** вредност по којој претражује целу излазну палету. (АПУСО)

2. Магационер **позива** систем да нађе целу излазну палету по задатој вредности. (АПСО)
3. Систем **тражи** целу излазну палету по задатој вредности. (СО)
4. Систем приказује Магационеру целу излазну палету. (ИА)
5. Магационер **уноси (мења)** податке о излазној палети. (АПУСО)
6. Магационер **контролише** да ли је коректно унео податке о излазној палети. (АНСО)
7. Магационер **позива** систем да запамти податке о излазној палети. (АПСО)
8. Систем **памти** податке о излазној палети. (СО)
9. Систем **приказује** Магационеру запамћени излаз целе палете и поруку: "Систем је запамтио целу излазну палету." (ИА)

Алтернативна сценарија:

4.1 Уколико систем не може да нађе излазну палету он приказује Магационеру поруку: "Систем не може да нађе целу излазну палету по задатој вредности". Прекида се извршење сценариа. (ИА)

9.1 Уколико систем не може да запамти податке о излазној палети он приказује Магационеру поруку "Систем не може да запамти целу излазну палету".(ИА)

10.3.5 СК5: Случај коришћења - Излаз дела палете

Назив СК
Креирање излазне палете

Актори СК
Магационер

Учесници СК
Магационер и систем (програм)

Предуслов:
Систем је укључен и Магационер је пријављен под својом шифром. Систем приказује форму за рад са излазом дела палете.

Основни сценарио СК:

1. Магационер **позива** систем да креира део излазне палете. (АПСО)
2. Систем **креира** део излазне палете. (СО)
3. Систем **приказује** Магационеру део излазне палете. (ИА)
4. Магационер **уноси** податке за део излазне палете. (АПУСО)
5. Магационер **контролише** да ли је коректно унео податке за део излазне палете. (АНСО)
6. Магационер **позива** систем да запамти податке о излазној палети. (АПСО)
7. Систем **памти** податке о излазној палети. (СО)
8. Систем **приказује** Магационеру запамћени излаз дела палете и поруку: "Систем је запамтио излаз дела палете". (ИА)

Алтернативна сценариа:

3.1 Уколико систем не може да креира део излазне палете он приказује Магационеру поруку: “Систем не може да креира део излазне палете”. Прекида се извршење сценарија. (ИА)

8.1 Уколико систем не може да запамти податке о излазној палети он приказује Магационеру поруку “Систем не може да запамти излаз дела палете”. (ИА)

10.3.6 СК6: Случај коришћења – Креирање робе

Назив СК
Креирање робе

Актори СК
Магационер

Учесници СК
Магационер и систем (програм)

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

Основни сценарио СК

1. Магационер **позива** систем да креира робу. (АПСО)
2. Систем **креира** робу. (СО)
3. Систем **приказује** Магационеру робу. (ИА)
4. Магационер **уноси** податке о роби. (АПУСО)
5. Магационер **контролише** да ли је коректно унео податке. (АНСО)
6. Магационер **позива** систем да запамти податке о роби. (АПСО)
7. Систем **памти** робу. (СО)
8. Систем **приказује** Магационеру запамћени податак и поруку: “Систем је запамтио робу“. (ИА)

Алтернативна сценариа:

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

8.1 Уколико систем не може да запамти податке он приказује Магационеру поруку “Систем не може да запамти робу”. (ИА)

10.3.7 СК7: Случај коришћења – Промена робе

Назив СК
Промена робе

Актори СК
Магационер

Учесници СК
Магационер и систем (програм)

Предуслов:
Систем је укључен и магационер је пријављен под својом шифром. Систем приказује форму за рад са робом. Учитана је листа свих података.

Основни сценарио СК:

1. Магационер **уноси** вредност по којој претражује робу. (АПУСО)
2. Магационер **позива** систем да нађе робу по задатој вредности. (АПСО)
3. Систем **тражи** промену робе по задатој вредности. (СО)
4. Систем приказује Магационеру робу. (ИА)
5. Магационер **уноси (мења)** робу. (АПУСО)
6. Магационер **контролише** да ли је коректно унео податке о роби. (АНСО)
7. Магационер **позива** систем да запамти податке о роби. (АПСО)
8. Систем **памти** промењене податке о роби. (СО)
9. Систем **приказује** Магационеру запамћену промену и поруку: "Систем је запамтио робу." (ИА)

Алтернативна сценариа:

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

9.1 Уколико систем не може да запамти податке о промени он приказује Магационеру поруку "Систем не може да запамти податак". (ИА)

10.3.8 СК8: Случај коришћења – Брисање робе

Назив СК
Брисање робе

Актори СК
Магационер

Учесници СК
Магационер и систем (програм)

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

Основни сценарио СК:

1. Магационер **уноси** вредност по којој претражује робу. (АПУСО)
2. Магационер **позива** систем да нађе робу по задатој вредности. (АПСО)
3. Систем **тражи** робу по задатој вредности. (СО)

4. Систем приказује Магационеру робу и поруку: “Систем је нашао робу по задатој вредности”. (ИА)
5. Магационер **позива** систем да обрише робу. (АПСО)
6. Систем **брише** робу. (СО)
7. Систем **приказује** Магационеру поруку: “Систем је обрисао робу.” (ИА)

Алтернативна сценариа:

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

7.1 Уколико систем не може да обрише податак он приказује Магационеру поруку “Систем не може да обрише податак”. (ИА)

11.Фаза анализе

Ова фаза описује пословну логику апликације која се приказује преко понашања и структуре софтверског система. Понашање се дефинише кроз дијаграме секвенци и уговоре о системским операцијама. Док структуру дефинишемо кроз концептуални и релациони модел.

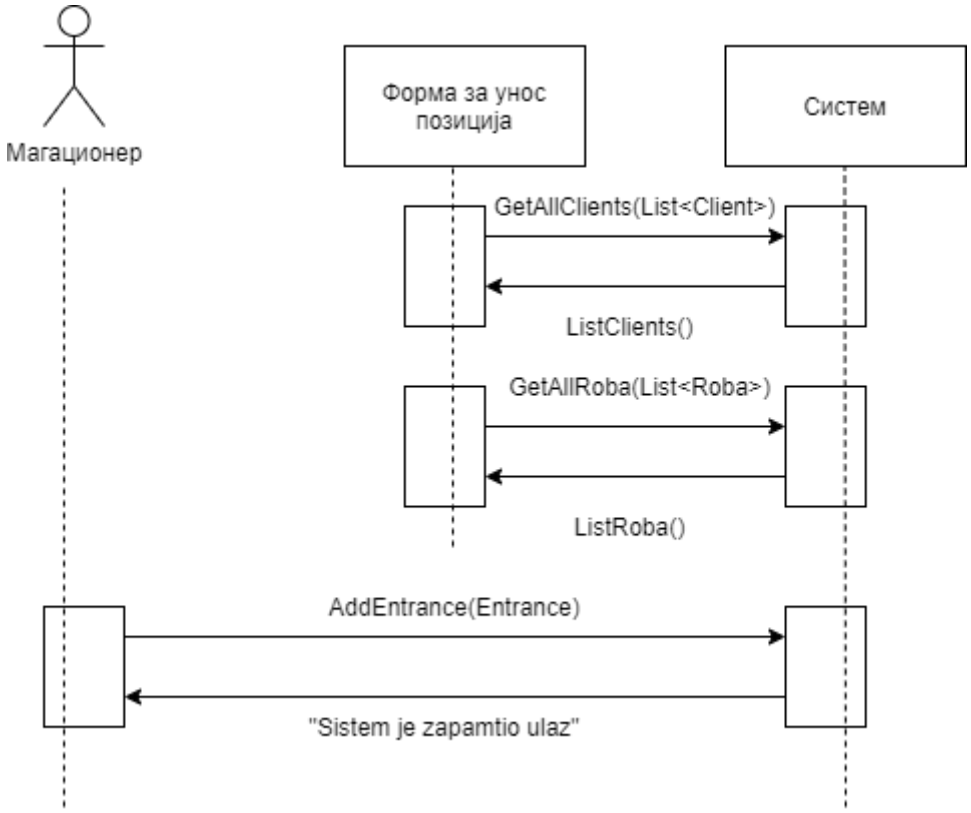
11.1 Понашање софтверског система – Системски дијаграм секвенци

Дијаграм секвенци приказује графички приказ тока порука између објеката саме апликације. Ток порука реализује одређене операције унутар система. Сваки случај корисника се реализује између корисника и софтверског система. У наставку ће бити приказани сви случаји коришћења.

11.1.1 ДС1: Креирање улаза палета

Основни сценарио СК:

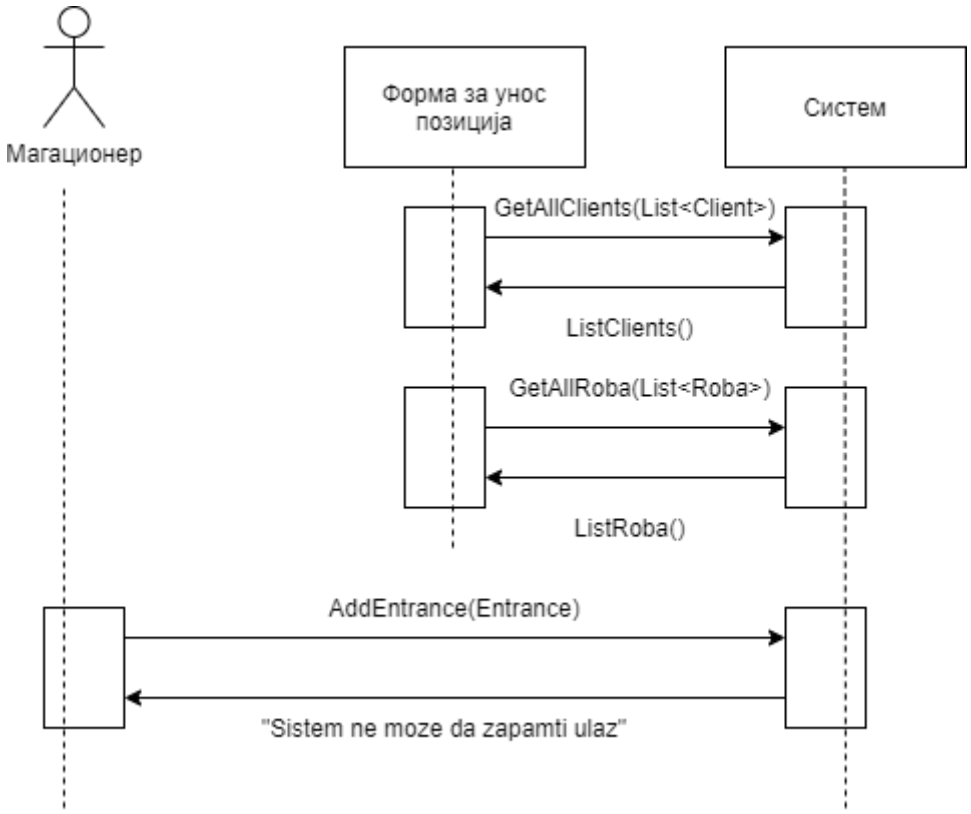
1. Форма **позива** систем да учита листу клијената. (АПСО)
2. Систем **враћа** форми листу клијената.(ИА)
3. Форма **позива** систем да учита листу робе. (АПСО)
4. Систем **враћа** форми листу робе.(ИА)
5. Магационер **позива** систем да запамти податке о улазу палета. (АПСО)
6. Систем **приказује** магационеру запамћени улаз и поруку: “Систем је запамтио улаз“. (ИА)



Слика 16 – ДС1 Основни сценарио

Алтернативни сценарио:

5.1 Уколико систем не може да запамти податке о улазу палета он приказује магационеру поруку “Систем не може да запамти улаз”. (ИА)



Слика 17 – ДС1 Алтернативни сценарио

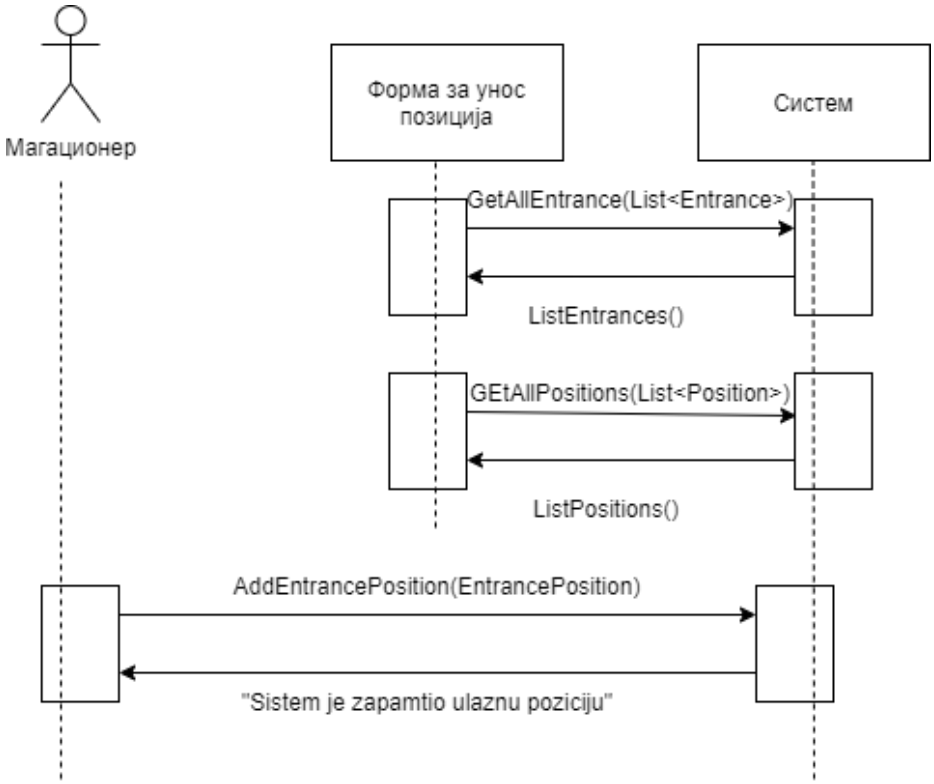
Са наведених секвенцних дијаграма уочавају се 3 системске операције:

1. Signal **GetAllClients**(List<Client>)
2. Signal **GetAllRoba**(List<Roba>)
3. Signal **AddEntrance**(Ulaz)

11.1.2 ДС2: Креирање позиције

Основни сценарио СК:

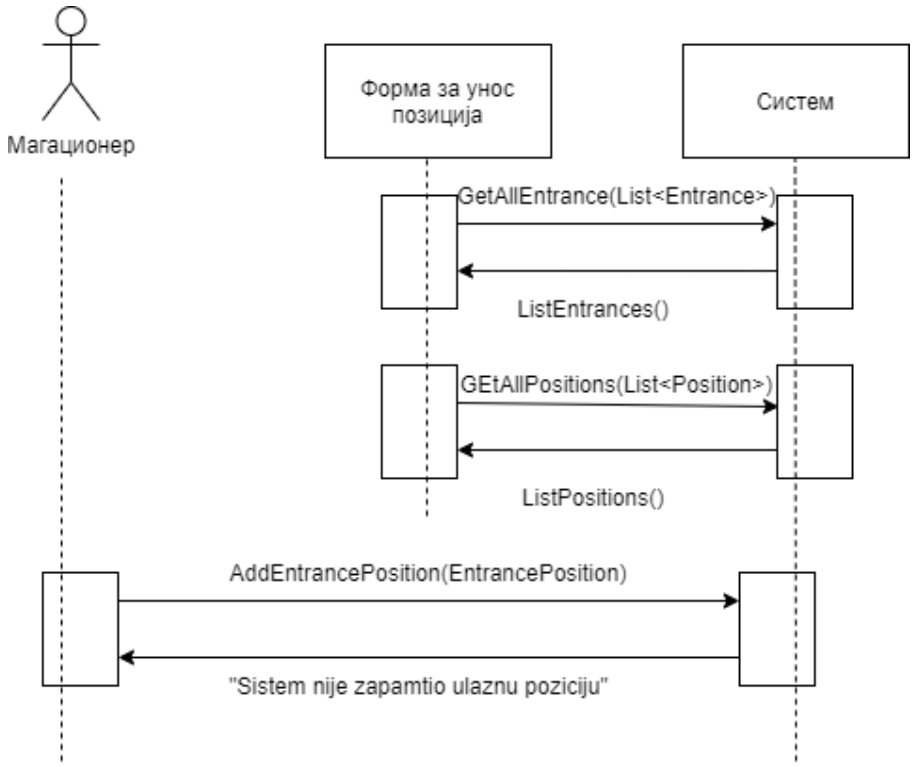
1. Форма **позива** систем да учита листу клијената. (АПСО)
2. Систем **враћа** форми листу клијената.(ИА)
3. Форма **позива** систем да учита листу робе. (АПСО)
4. Систем **враћа** форми листу робе.(ИА)
5. Магационер **позива** систем да креира позицију. (АПСО)
6. Систем **приказује** магационеру позицију. (ИА)
7. Магационер **позива** систем да запамти податке о позицији. (АПСО)
8. Систем **приказује** магационеру запамћени улаз и поруку: "Систем је запамтио улазну позицију". (ИА)



Слика 18 - ДС2 Основни сценарио

Алтернативни сценарио:

8.1 Уколико систем не може да запамти податке о позицији он приказује магационеру поруку “Систем није запамтио позицију”. (ИА)



Слика 19 - ДС3 Алтернативни сценарио

Са наведених секвенцних дијаграма уочавају се 3 системске операције:

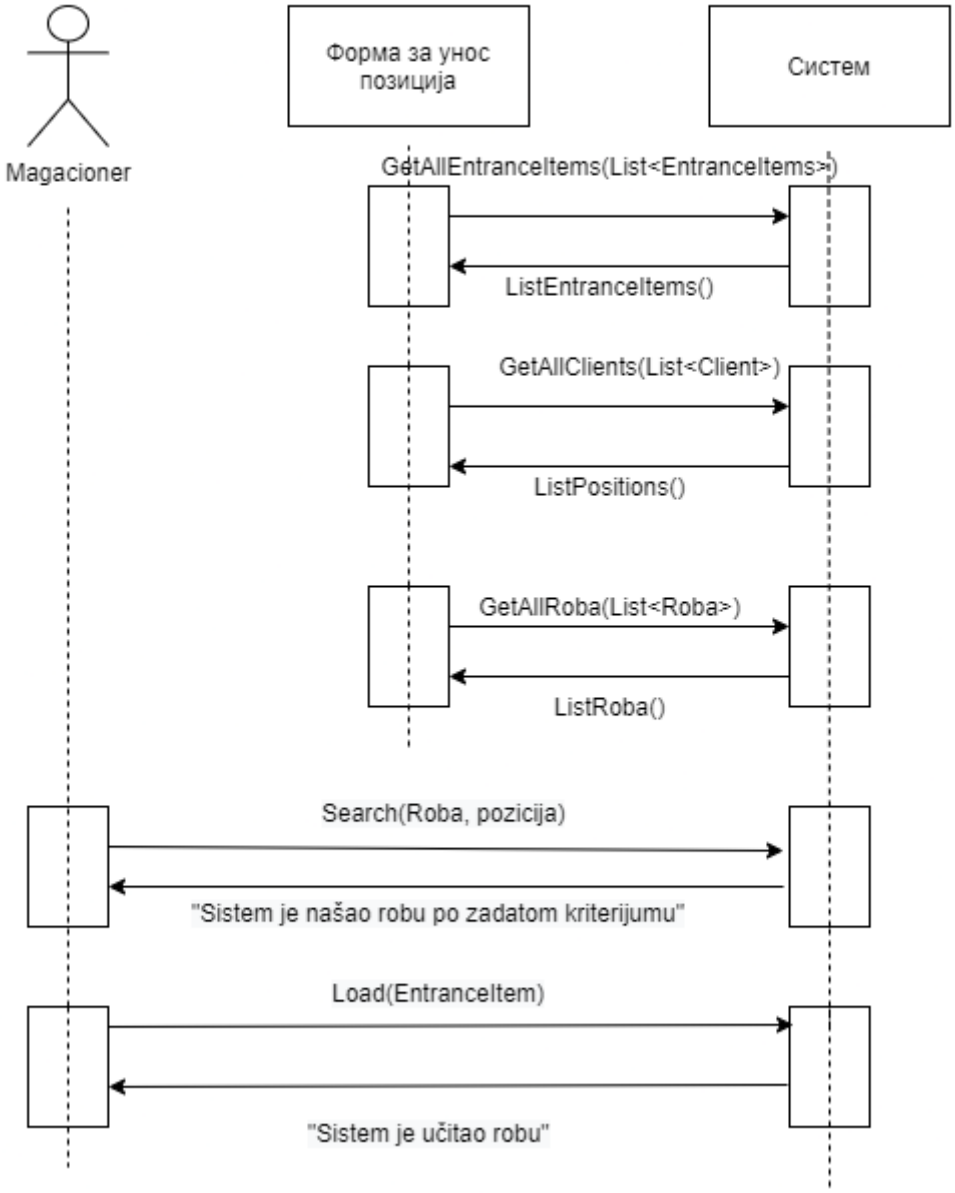
1. Signal **GetAllEntrance(List<Entrance>)**
2. Signal **GetAllPositions(List<Positions>)**
3. Signal **AddEntrancePositions(Entrance,Position)**

11.1.3 ДС3: Претраживање робе по палетама

Основни сценарио СК:

1. Форма **позива** систем да учита листу улазних ставки. (АПСО)
2. Систем **враћа** форми листу улазних ставки.(ИА)
3. Форма **позива** систем да учита листу клијената. (АПСО)
4. Систем **враћа** форми листу клијената.(ИА)
5. Форма **позива** систем да учита листу робе. (АПСО)
6. Систем **враћа** форми листу робе.(ИА)
7. Магационер **позива** систем да нађе робу по задатом критеријуму. (АПСО)
8. Систем приказује магационеру податке о роби и поруку: “Систем је нашао робу по задатом критеријуму”. (ИА)
9. Магационер **одабира** жељену робу. (АПСО)

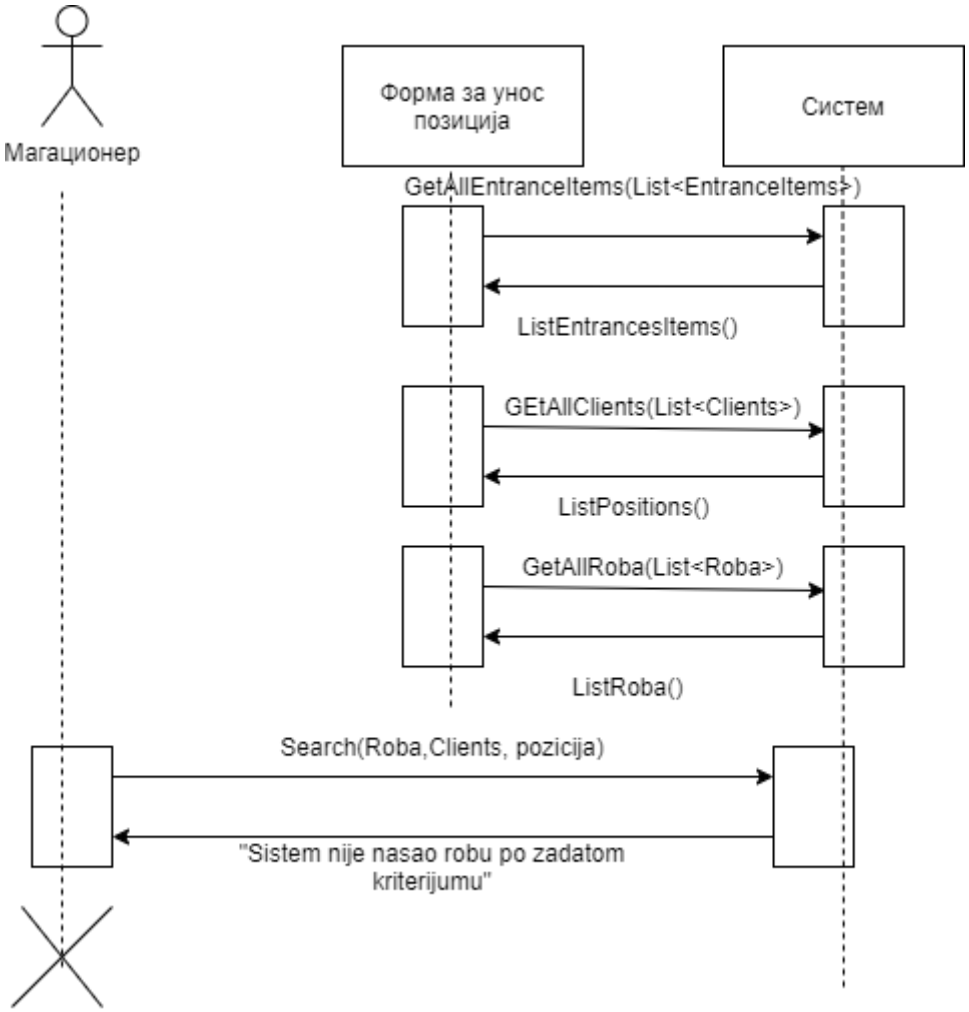
10. Систем приказује магационеру податке о изабраној роби и поруку: “Систем је нашао робу ”. (ИА)



Слика 20 - ДС3 Основни сценарио

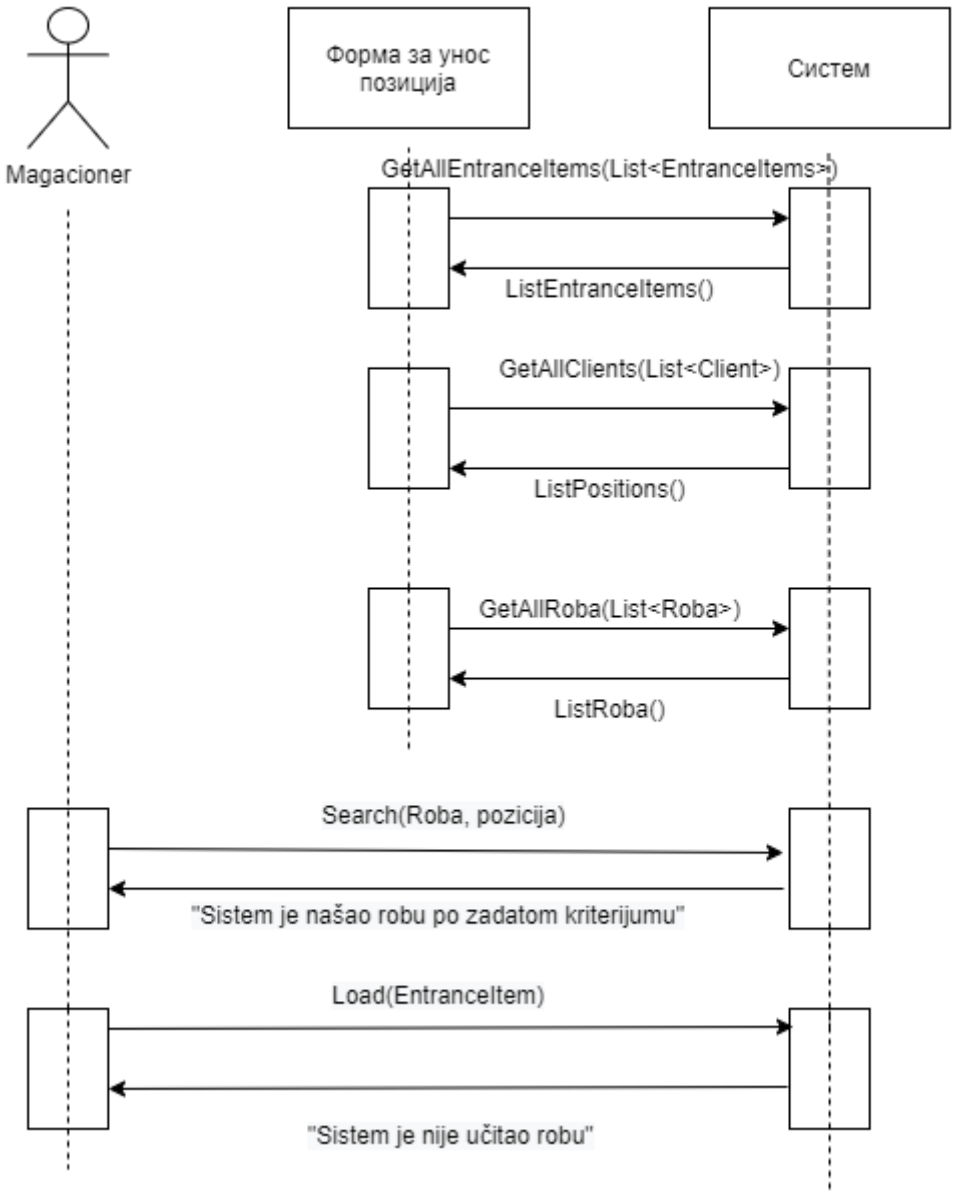
Алтернативни сценарио:

8.1 Уколико систем не може да нађе више роба он приказује магационеру поруку: “Систем не може да нађе робу по задатом критеријуму”. Прекида се извршење сценариа. (ИА).



Слика 21 - ДС3 Алтернативни сценарио 1

10.1 Уколико систем не може да учита робу он приказује магационеру поруку: “Систем није учитао робу”. (ИА)



Слика 22 - ДС3 Алтернативни сценарио 2

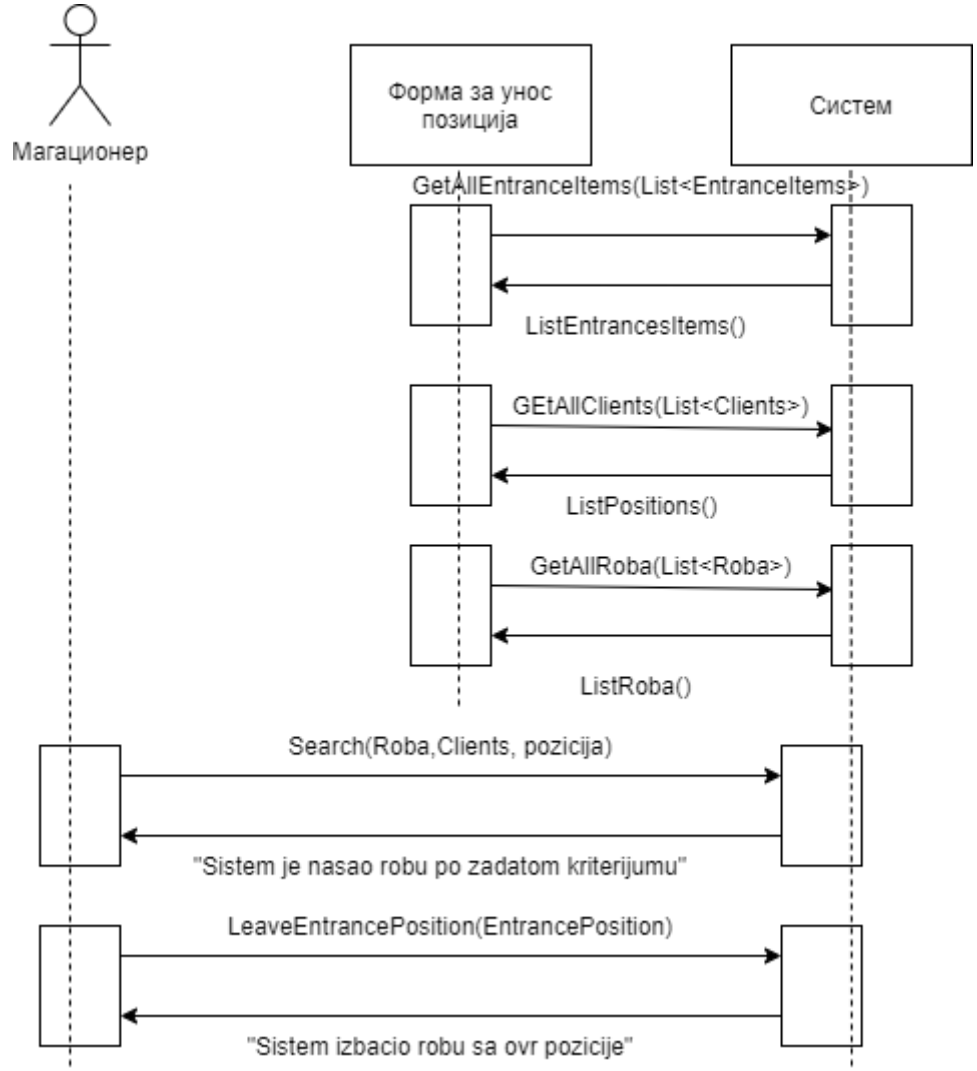
Са наведених секвенцних дијаграма уочавају се 5 системске операције:

1. **Signal GetAllEntranceItems(List<EntranceItems>)**
2. **Signal GetAllClients(List<Client>)**
3. **Signal GetAllRoba(List<Roba>)**
4. **Signal Search(Roba,Client,pozicija)**
5. **Signal Load(EntranceItem)**

11.1.4 ДС4: Излаз целе палете

Основни сценарио СК:

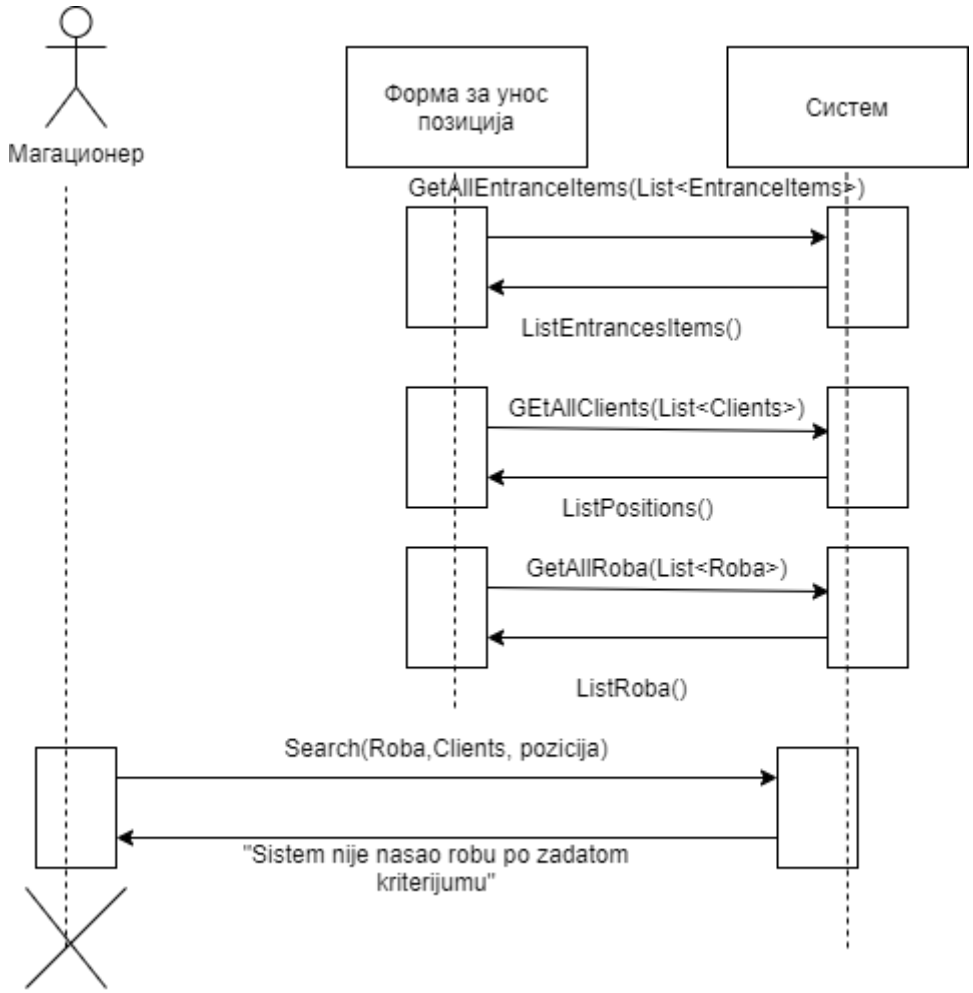
1. Форма **позива** систем да учита листу улазних ставки. (АПСО)
2. Систем **враћа** форми листу улазних ставки.(ИА)
3. Форма **позива** систем да учита листу клијената. (АПСО)
4. Систем **враћа** форми листу клијената.(ИА)
5. Форма **позива** систем да учита листу робе. (АПСО)
6. Систем **враћа** форми листу робе.(ИА)
7. Магационер **позива** систем да нађе целу излазну палету по задатој вредности. (АПСО)
8. Систем приказује Магационеру целу излазну палету и поруку: “Систем је нашао целу излазну палету по задатој вредности”. (ИА)
9. Магационер **позива** систем да запамти податке о излазној палети. (АПСО
10. Систем **приказује** Магационеру запамћени излаз целе палете и поруку: “Систем је запамтио целу излазну палету.” (ИА)



Слика 23 - СД4 Основни сценарио

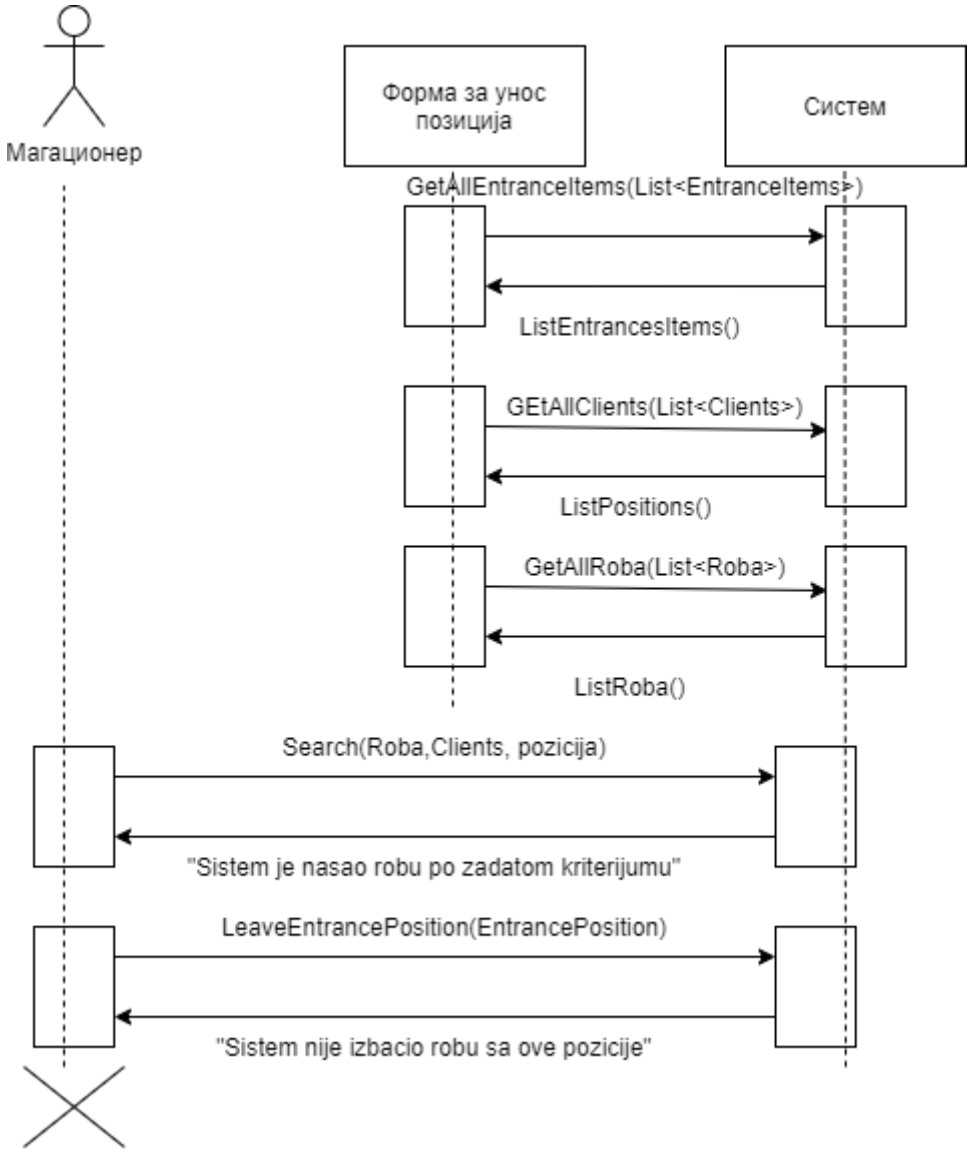
Алтернативна сценариа:

8.1 Уколико систем не може да нађе целу излазну палету он приказује Магационеру поруку: “Систем не може да нађе целу излазну палету по задатој вредности”. Прекида се извршење сценариа. (ИА)



Слика 24 - ДС4 Алтернативни сценарио 1

10.1 Уколико систем не може да запамти податке о излазној палети он приказује Магационеру поруку “Систем не може да запамти целу излазну палету”. Прекида се извршење сценариа. (ИА)



Слика 25 - ДС4 Алтернативни сценарио 2

Са наведених секвенцних дијаграма уочавају се 5 системске операције:

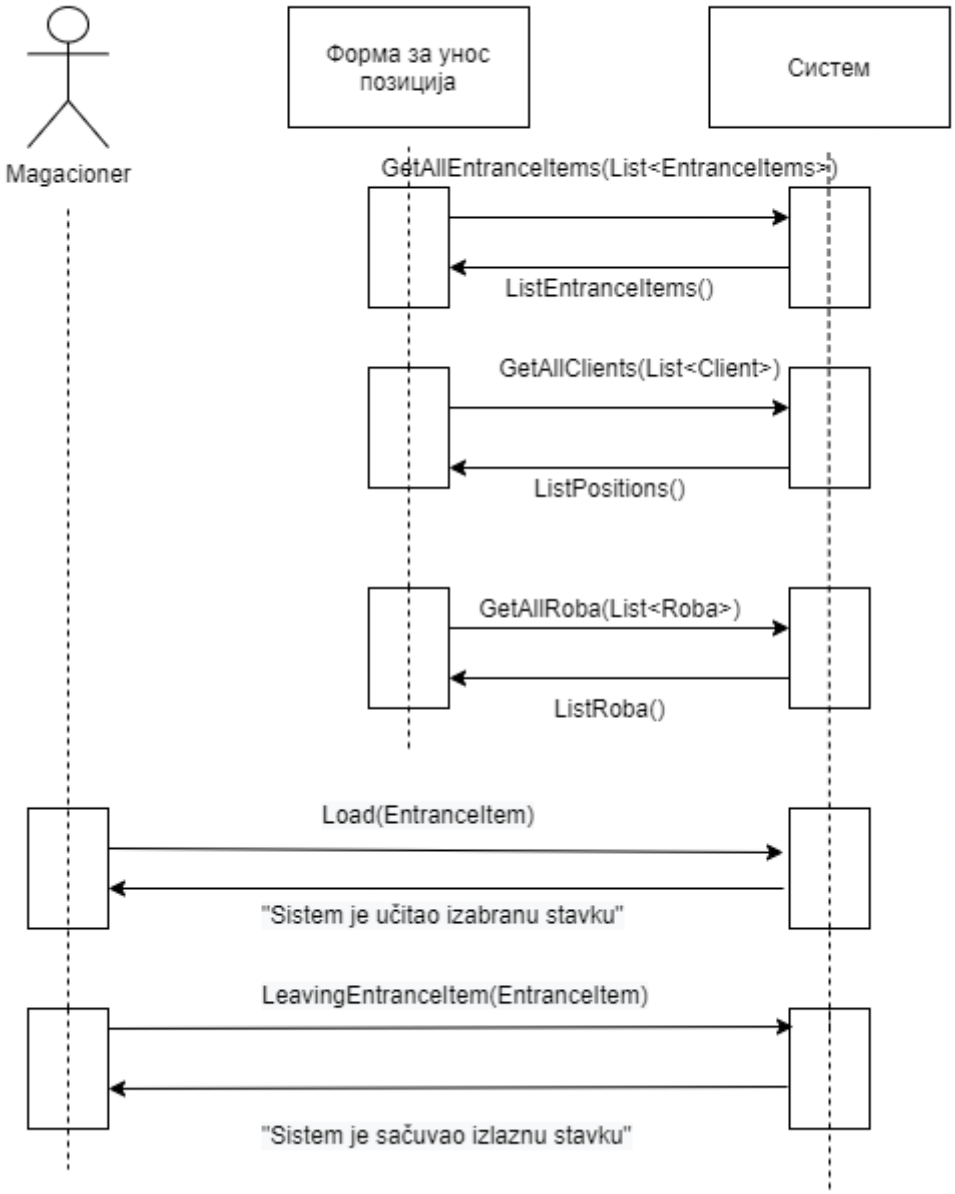
1. **Signal GetAllEntranceItems(List<EntranceItems>)**
2. **Signal GetAllClients(List<Client>)**
3. **Signal GetAllRoba(List<Roba>)**
4. **Signal Search(Roba,Client,pozicija)**
5. **Signal LeaveEntrancePosition(Entrance,Position)**

11.1.5 ДС5: Излаз дела палете

Основни сценарио СК:

1. Форма **позива** систем да учита листу улазних ставки. (АПСО)

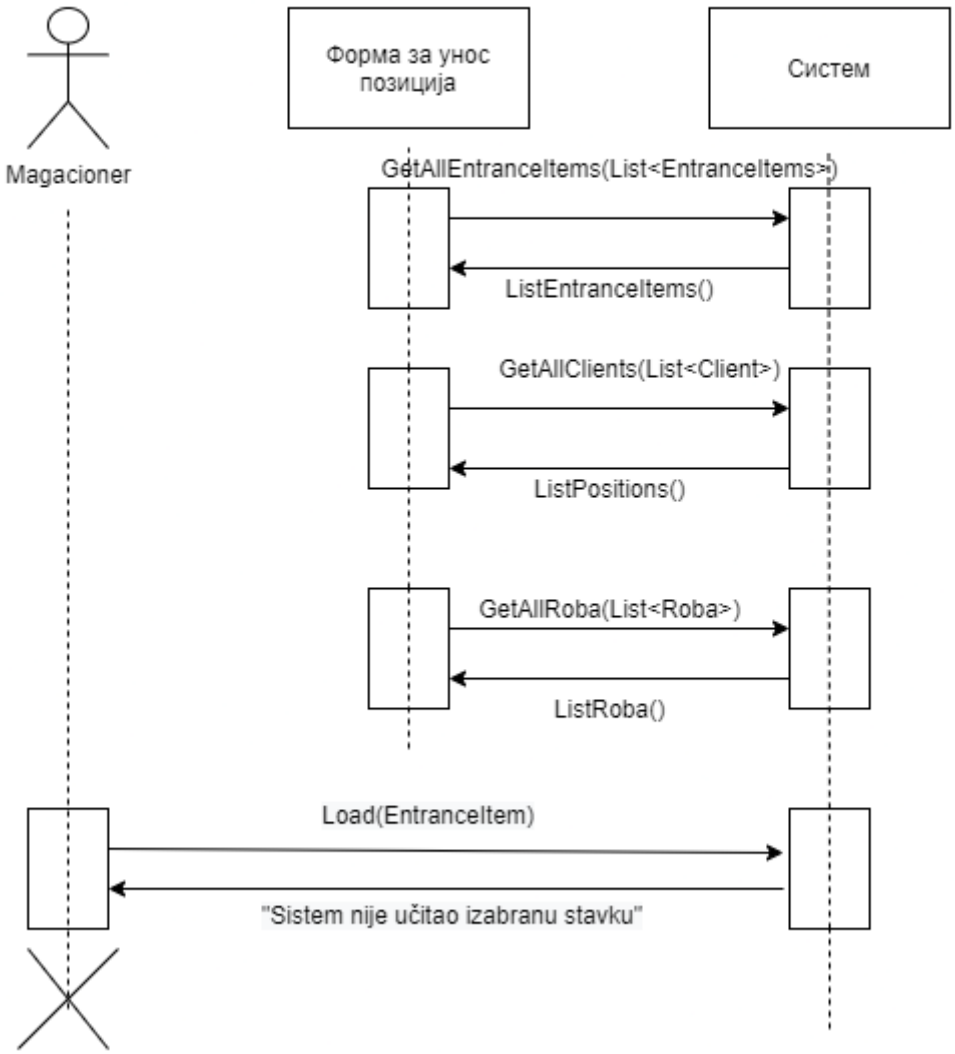
2. Систем **враћа** форми листу улазних ставки.(ИА)
3. Форма **позива** систем да учита листу клијената. (АПСО)
4. Систем **враћа** форми листу клијената.(ИА)
5. Форма **позива** систем да учита листу робе. (АПСО)
6. Систем **враћа** форми листу робе.(ИА)
7. Магационер **позива** систем да креира део излазне палете. (АПСО)
8. Систем **приказује** Магационеру део излазне палете и поруку: “Систем је учитао изабрану ставку“. (ИА)
9. Магационер **позива** систем да запамти податке о излазној палети. (АПСО)
10. Систем **приказује** Магационеру запамћени излаз дела палете и поруку: “Систем је сачувао излазну ставку“. (ИА)



Слика 26 - ДС5 Основни сценарио

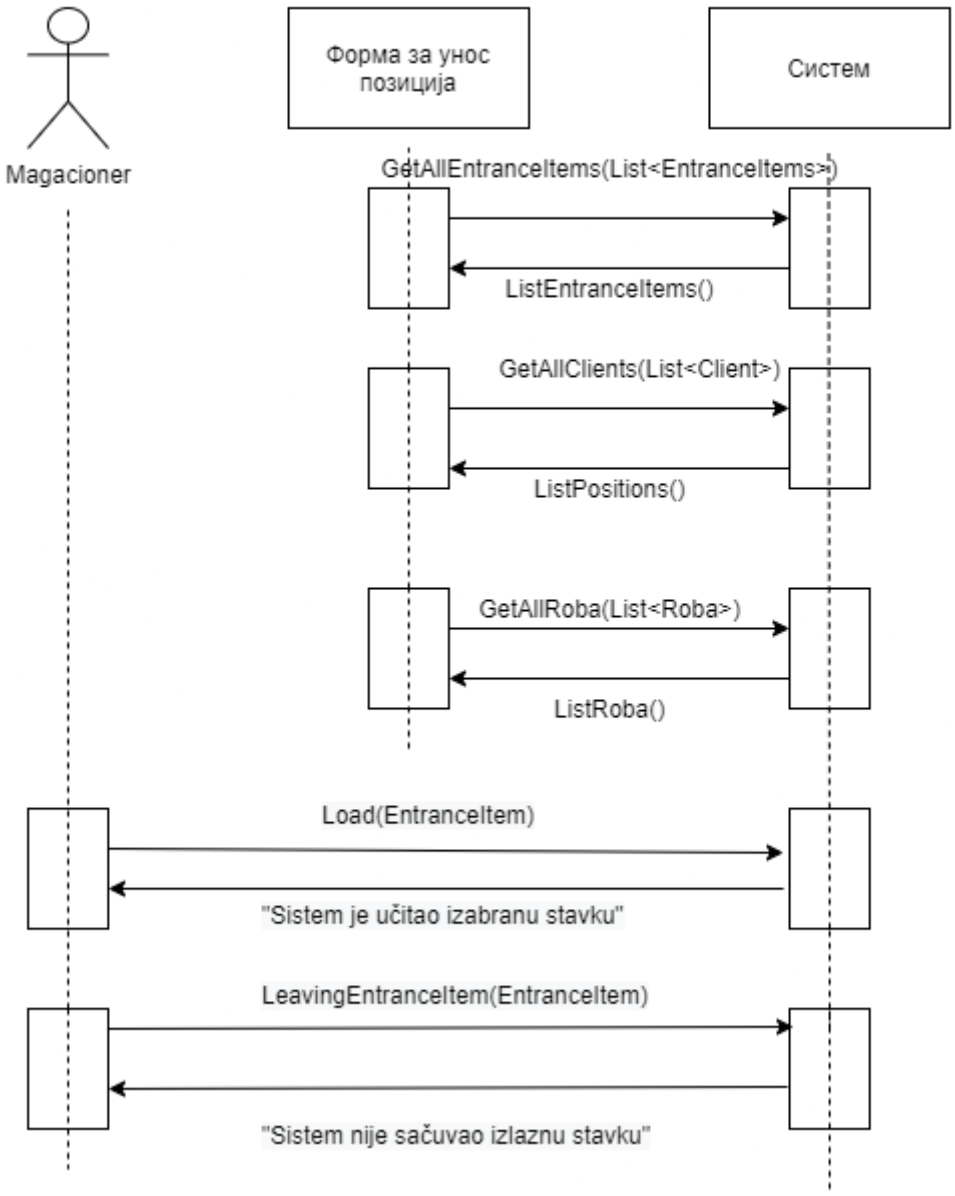
Алтернативна сценариа:

8.1 Уколико систем не може да креира део излазне палете он приказује Магационеру поруку: “Систем није учитао изабрану ставку”. Прекида се извршење сценариа. (ИА)



Слика 27 - ДС5 Алтернативни сценарио 1

10.1 Уколико систем не може да запамти податке о излазној палети он приказује Магационеру поруку “Систем није сачувао излазну ставку”. (ИА)



Слика 28 - ДС5 Алтернативни сценарио 2

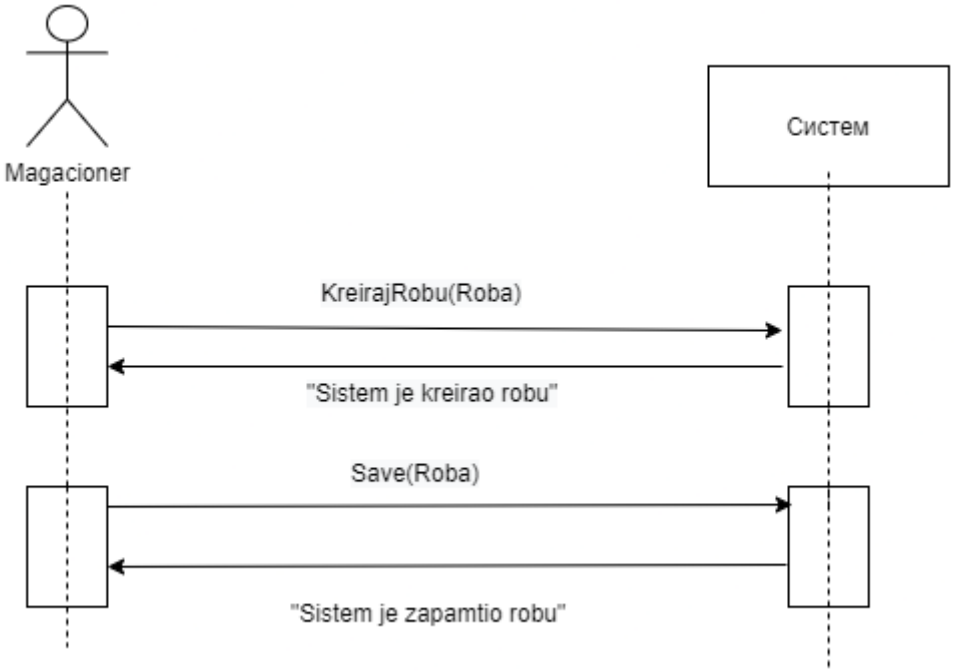
Са наведених секвенцних дијаграма уочавају се 5 системске операције:

1. **Signal GetAllEntranceItems(List<EntranceItems>)**
2. **Signal GetAllClients(List<Client>)**
3. **Signal GetAllRoba(List<Roba>)**
4. **Signal Load(EntranceItem)**
5. **Signal LeaveEntranceItem(EntranceItem)**

11.1.6 ДС6: Креирање робе

Основни сценарио СК:

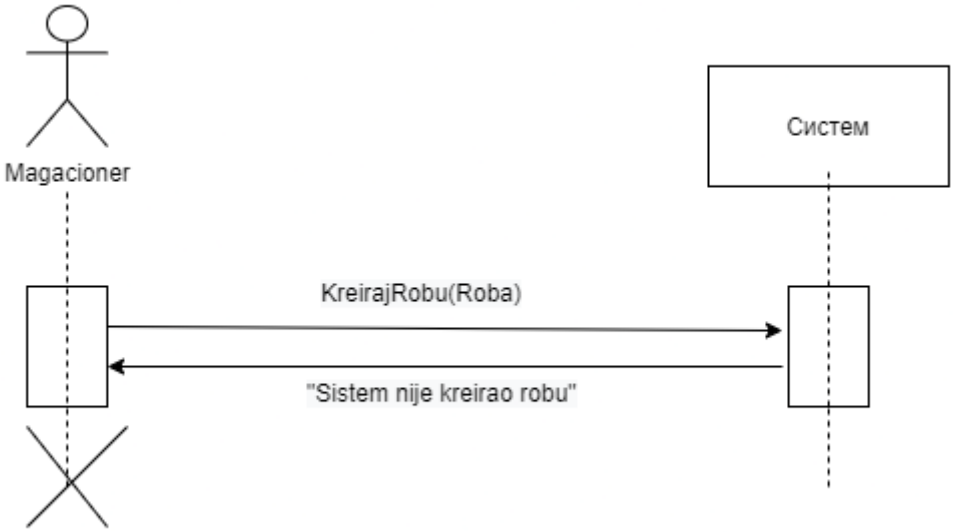
1. Магационер **позива** систем да креира робу. (АПСО)
2. Систем **приказује** Магационеру податак и поруку: “Систем је креирао робу“. (ИА)
3. Магационер **позива** систем да запамти податке о роби. (АПСО)
4. Систем **приказује** Магационеру запамћени податак и поруку: “Систем је запамтио робу“. (ИА)



Слика 29 – ДС6 Основни сценарио

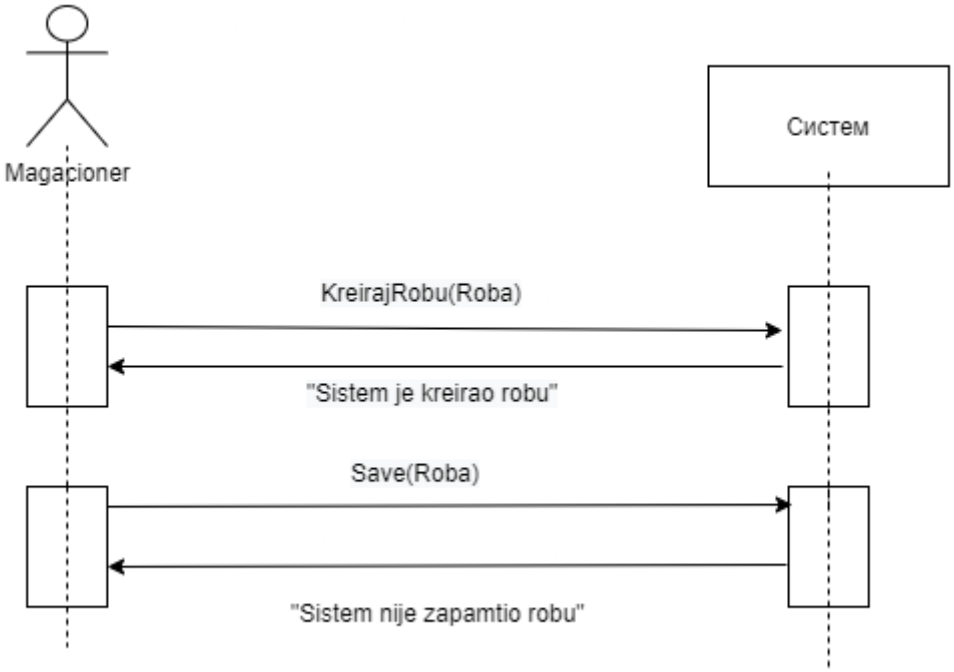
Алтернативни сценарио:

2.1 Уколико систем не може да креира податак он приказује Магационеру поруку: “Систем није креирао робу”. Прекида се извршење сценариа. (ИА)



Слика 30 – ДС6 Алтернативни сценарио 1

4.1 Уколико систем не може да запамти податке он приказује Магационеру поруку “Систем није запамтио податак”. (ИА)



Слика 31 – ДС6 Алтернативни сценарио 2

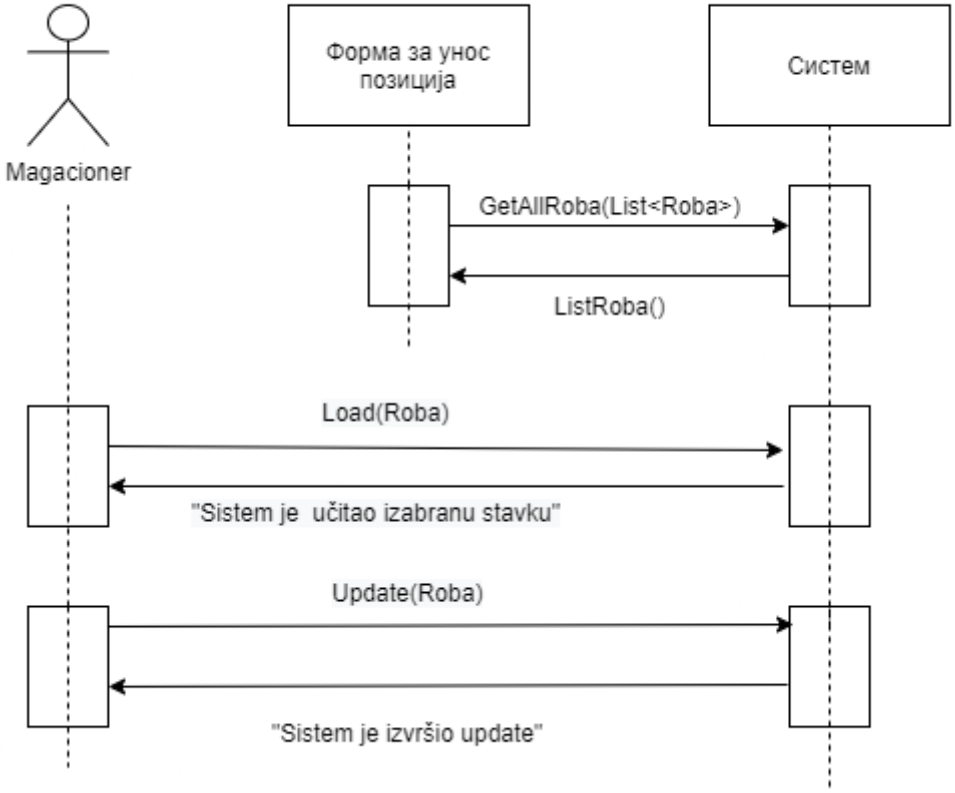
Са наведених секвенцних дијаграма уочавају се 2 системске операције:

1. **Signal KreirajRobu(Roba);**
2. **Signal Save(Roba);**

11.1.7 ДС7: Промена робе

Основни сценарио СК:

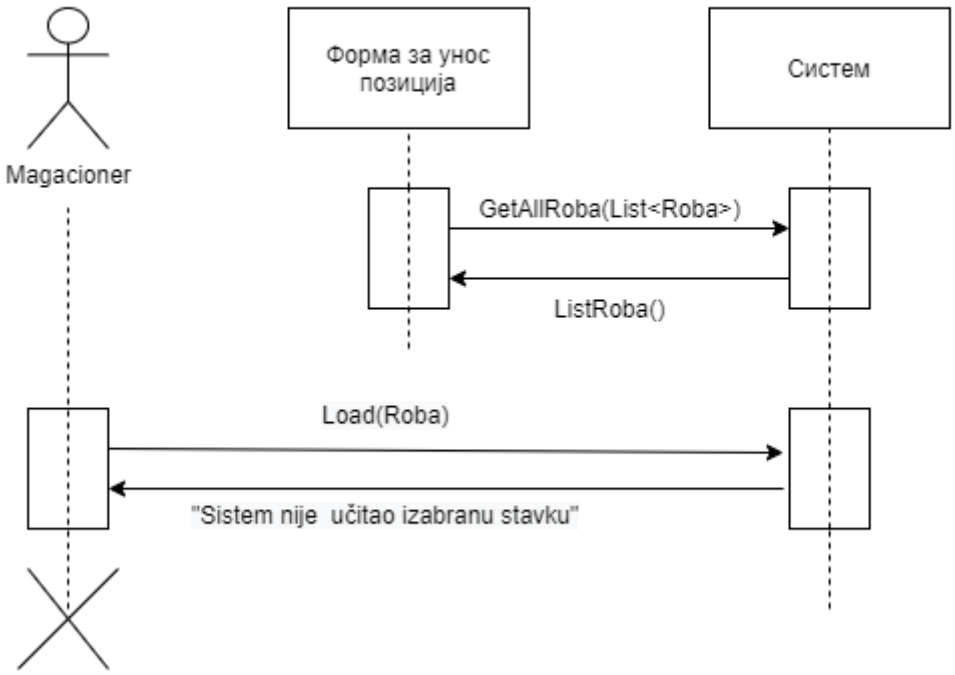
1. Форма **позива** систем да учита листу робе. (АПСО)
2. Систем **враћа** форми листу робе.(ИА)
3. Магационер **позива** систем да нађе робу по задатој вредности. (АПСО)
4. Систем приказује Магационеру податак и поруку: “Систем је учитао изабрану ставку”. (ИА)
5. Магационер **позива** систем да запамти податке о промени робе. (АПСО)
6. Систем **приказује** Магационеру запамћену робу и поруку: “Систем је извршио промену.” (ИА)



Слика 32 – ДС7 Основни сценарио

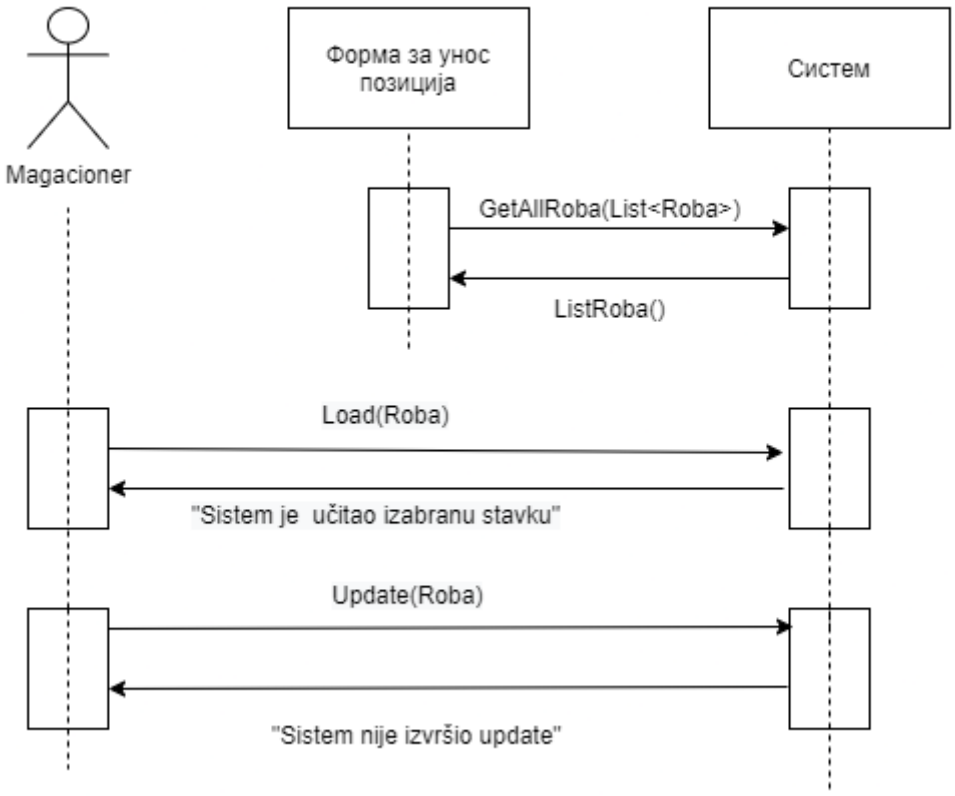
Алтернативни сценарио:

4.1 Уколико систем не може да нађе робу он приказује Магационеру поруку: “Систем није учитао изабрану ставку”. Прекида се извршење сценариа. (ИА)



Слика 33 – ДС7 Алтернативни сценарио 1

6.1 Уколико систем не може да запамти податке о промени он приказује Магационеру поруку “Систем не може да запамти робу”. (ИА)



Слика 34 – ДС7 Алтернативни сценарио 2

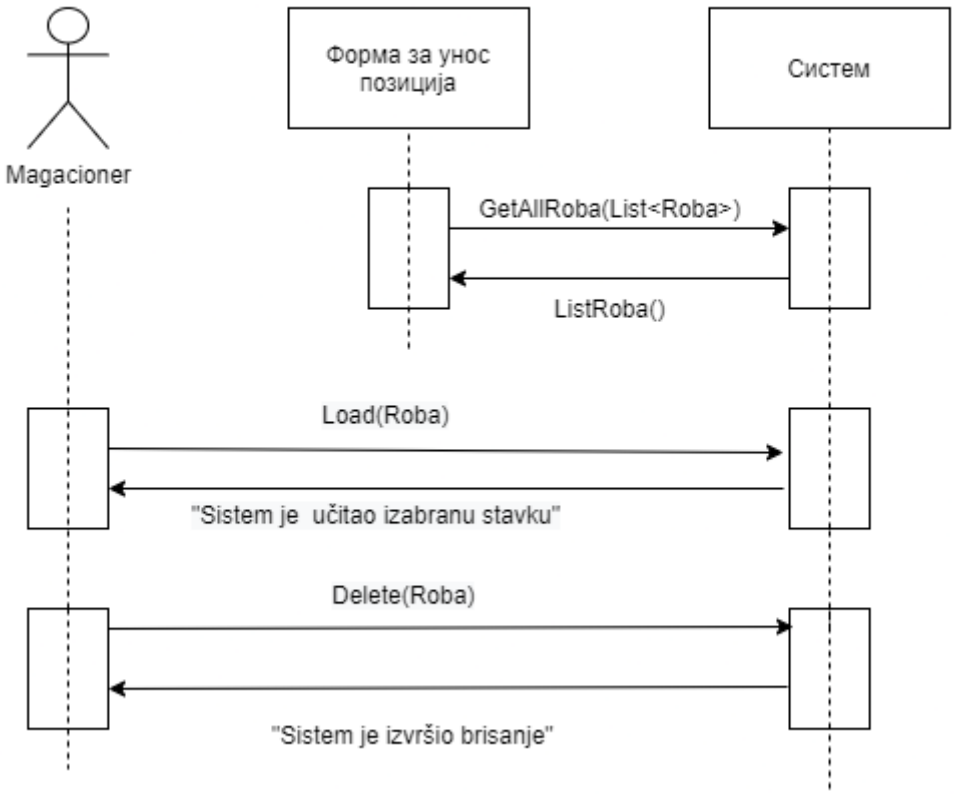
Са наведених секвенцних дијаграма уочавају се 5 системске операције:

1. **Signal GetAllRoba(List<Roba>)**
2. **Signal Load(Roba)**
3. **Signal Update(Roba)**

11.1.8 ДС8: Брисање робе

Основни сценарио СК:

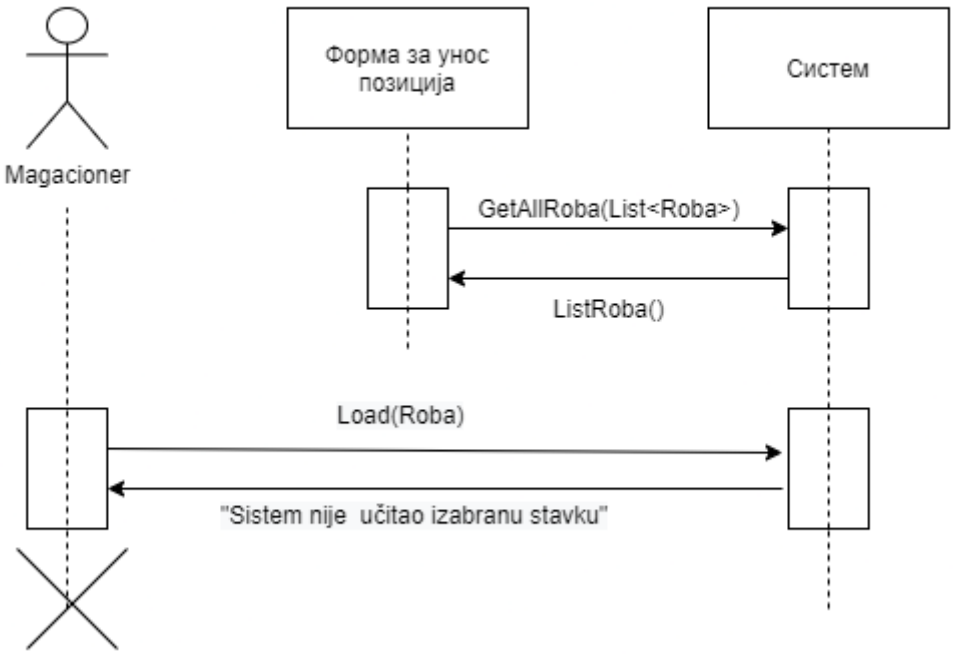
1. Форма **позива** систем да учита листу робе. (АПСО)
2. Систем **враћа** форми листу робе.(ИА)
3. Магационер **позива** систем да нађе податак по задатој вредности. (АПСО)
4. Систем приказује Магационеру податак и поруку: “Систем је учитао изабрану ставку”. (ИА)
5. Магационер **позива** систем да обрише податак. (АПСО)
6. Систем **приказује** Магационеру поруку: “Систем је обрисао податак.” (ИА)



Слика 35 - ДС8 Основни сценарио

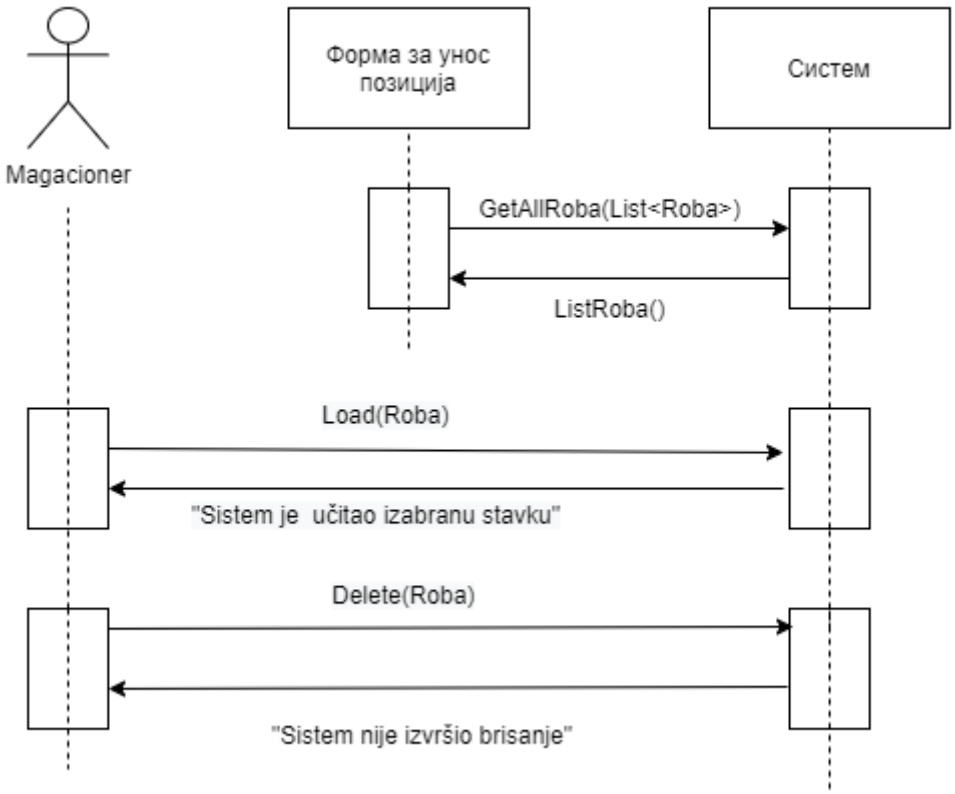
Алтернативни сценарио:

4.1 Уколико систем не може да нађе робу он приказује Магационеру поруку: “Систем није учитао изабрану ставку”. Прекида се извршење сценариа. (ИА)



Слика 36 - ДС8 Алтернативни сценарио 1

6.1 Уколико систем не може да обрише робу он приказује Магационеру поруку “Систем није извршио брисање”. (ИА)



Слика 37 - ДС8 Алтернативни сценарио 2

Са наведених секвенцних дијаграма уочава се 5 системска операција:

1. **Signal GetAllRoba(List<Roba>)**
2. **Signal Load(Roba)**
3. **Signal Delete(Roba)**

11.2 Понашање софтверског система – Дефинисање уговора о системским операцијама

За сваку системску операцију се дефинишу уговори који описује понашање саме операције. Један уговор се односни искључиво на једну операцију и за њега се дефинишу предуслови и постуслови.

11.2.1 Уговор УГ1: ВратиСвеКлијенте

Операција: GetAllClients(List<Client>)Signal;

Веза са СК: СК1, СК3, СК4, СК5

Предуслови: /

Постуслови: /

11.2.2 Уговор УГ2: ВратиСвуРобу

Операција: GetAllRoba(List<Roba>)Signal;

Веза са СК: СК3, СК5, СК6, СК8

Предуслови: /

Постуслови: /

11.2.3 Уговор УГ3: ВратиСвеУлазнеСтавке

Операција: GetAllEntranceItems(List<Pozicija>)Signal;

Веза са СК: СК3, СК5

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

Постуслови: /

11.2.4 Уговор УГ4: ВратиСвеУлазе

Операција: GetAllEntrance(List<Entrance>)Signal;

Веза са СК: СК2

Предуслови: /

Постуслови: /

11.2.5 Уговор УГ5: ВратиСвеПозиције

Операција: GetAllPositions(List<Position>)Signal;

Веза са СК: СК2

Предуслови: Вредносна и структурна ограничења над објектом морају бити задовољена. Ако је обрачун обрађен или сторниран не може се извршити системска операција. Листа свих улаза је учитана.

Постуслови:

- Израчуната је вредност сваке од ставки обрачуна.
- Израчуната је укупна вредност обрачуна.

11.2.6 Уговор УГ6: ОбришиРобу

Операција: Delete(Roba)Signal;

Веза са СК: СК8

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

Постуслови: Роба је обрисана.

11.2.7 Уговор УГ7: АжурирајРобу

Операција: Update(Roba)Signal;

Веза са СК: СК

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

Постуслови: Роба је промењена.

11.2.8 Уговор УГ8: Учитај

Операција: Load(Podatak)Signal;

Веза са СК: СК3, СК5, СК7, СК8

Предуслови: Вредносна и структурна ограничења над објектом Роба, Клијент, Магационер морају бити задовољена.

Постуслови: Роба, клијент или магационер је учитан.

11.2.9 Уговор УГ19: СачувајРоба

Операција: Save(Roba)Signal;

Веза са СК: СК6

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

Постуслови: Роба је сачувана.

11.2.10 Уговор УГ10: ИзлазПалетнеСтавке

Операција: LeaveEntranceItem(EntranceItem)Signal;

Веза са СК: СК5

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

Постуслови: Излазна ставка је склоњена са складишта.

11.2.11 Уговор УГ11: ИзлазПалетнеПозиције

Операција: LeaveEntrancePosition(Entrance,Position)Signal;

Веза са СК: СК4

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

Постуслови: Податак је промењен.

11.2.12 Уговор УГ12: Претрага

Операција: Search(Roba, Client, Pozicija)Signal;

Веза са СК: СК3, СК4

Предуслови: Вредносна и структурна ограничења над објектом Роба, Клијент, Позиција морају бити задовољена.

Постуслови: Пронађена је тражена роба.

11.2.13 Уговор УГ13: ПовежиПозицијуСаУлазом

Операција: AddEntrancePosition(Entrance,Position)Signal;

Веза са СК: СК2

Предуслови: Вредносна и структурна ограничења над објектом Улаз, Позиција морају бити задовољена.

Постуслови: Улазна позиција је повезана са улазом.

11.2.14 Уговор УГ14: ДодајУлаз

Операција: AddEntrance(Entrance)Signal;

Веза са СК: СК1

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

Постуслови: Улаз је сачуван.

11.2.15 Уговор УГ15: КреирајРобу

Операција: CreateRoba()Signal;

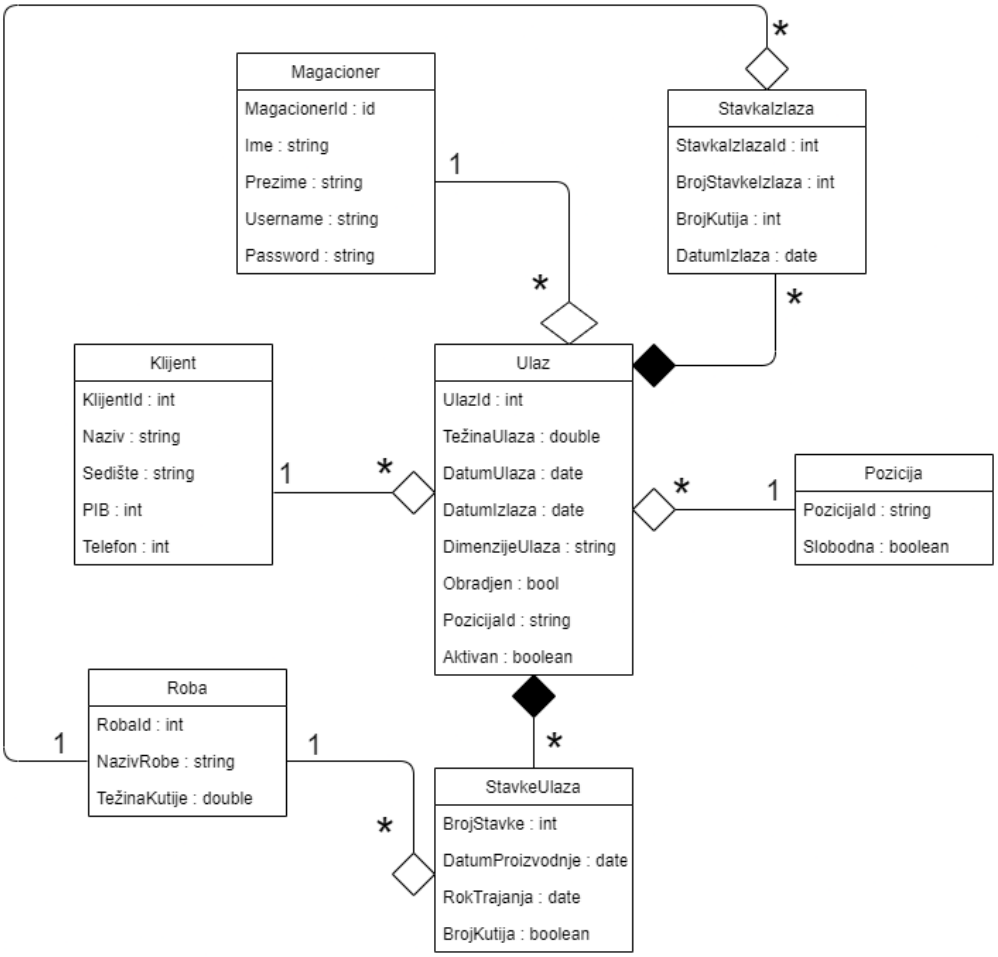
Веза са СК: СК6

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

Постуслови: Роба је сачувана.

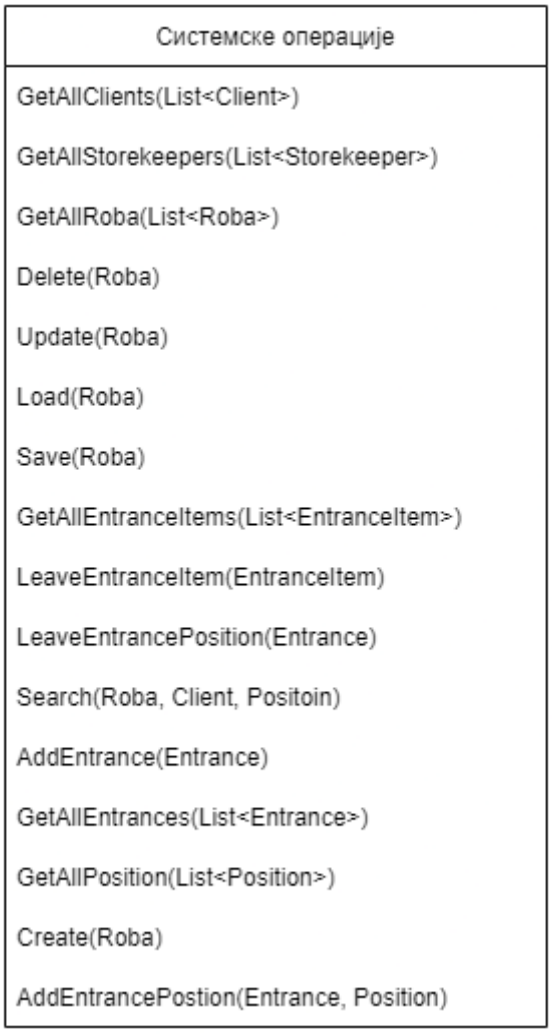
11.3 Структура софтверског система – Концептуални (доменски) модел

Структура софтверског система описана је помоћу следећег концептуалног модела који је приказан на слици 36.



Слика 38 - Концептуални модел

Слика 39. приказује све системске операције које треба имплементирати.



Слика 39 - Системске операције

12 Фаза пројектовања

Пројектовање апликације се одређује на основу информација које сте сакупили у фази планирања. У овом делу пројекта се објашњава пројектовање корисничког интерфејса, екранских форми, контролера корисничког интерфејса, апликационе логике и складишта података.

12.1 Архитектура софтверског система

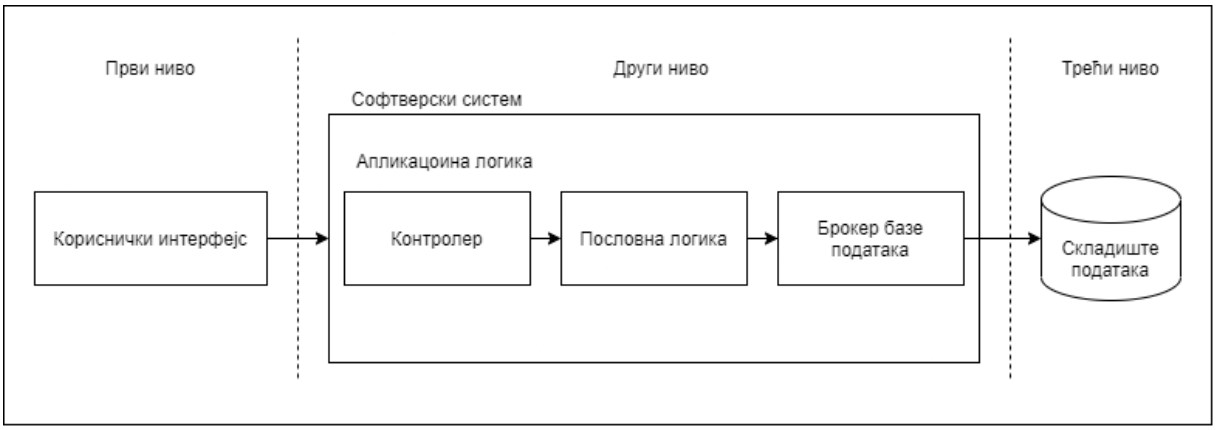
Архитектура система је тронивојска и састоји се од следећих нивоа:

1. Презентациони слој (кориснички интерфејс) – овај слој обезбеђује приказ података крајњем кориснику користећи неку од расположивих технологија за кориснички интерфејс.
2. Слој пословне логике (апликациона логика) – овај слој имплементира пословну логику апликације. Пословна логику чине пословни процеси и пословне

компоненте. Овде се такође имплементирају и пословна правила добијена у процесу анализе.

3. Слој података (складиште података) – већина пословних апликација користи релационе базе података за складиштење података.

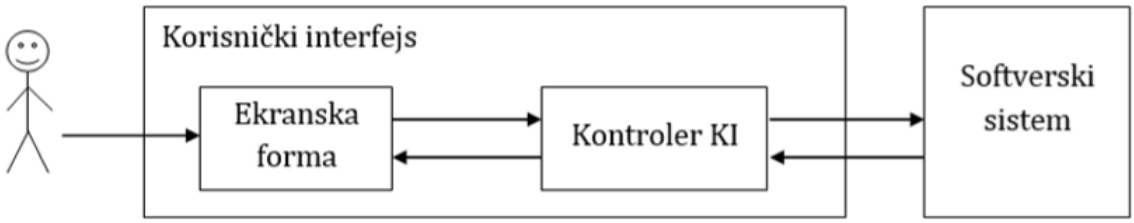
Познато је да се презентациони слој налази на страни клијента, док слој пословне логике и података на страни сервера.



Слика 40 - Тронивојска архитектура

12.2 Пројектовање корисничког интерфејса

Кориснички интерфејс представља начин на који програм комуницира са корисником. Он се састоји од екранске форме и контролера корисничког интерфејса. Контролер корисничког интерфејса комуницира са сервером и прослеђује све улазе или излазе ка екранској форми.

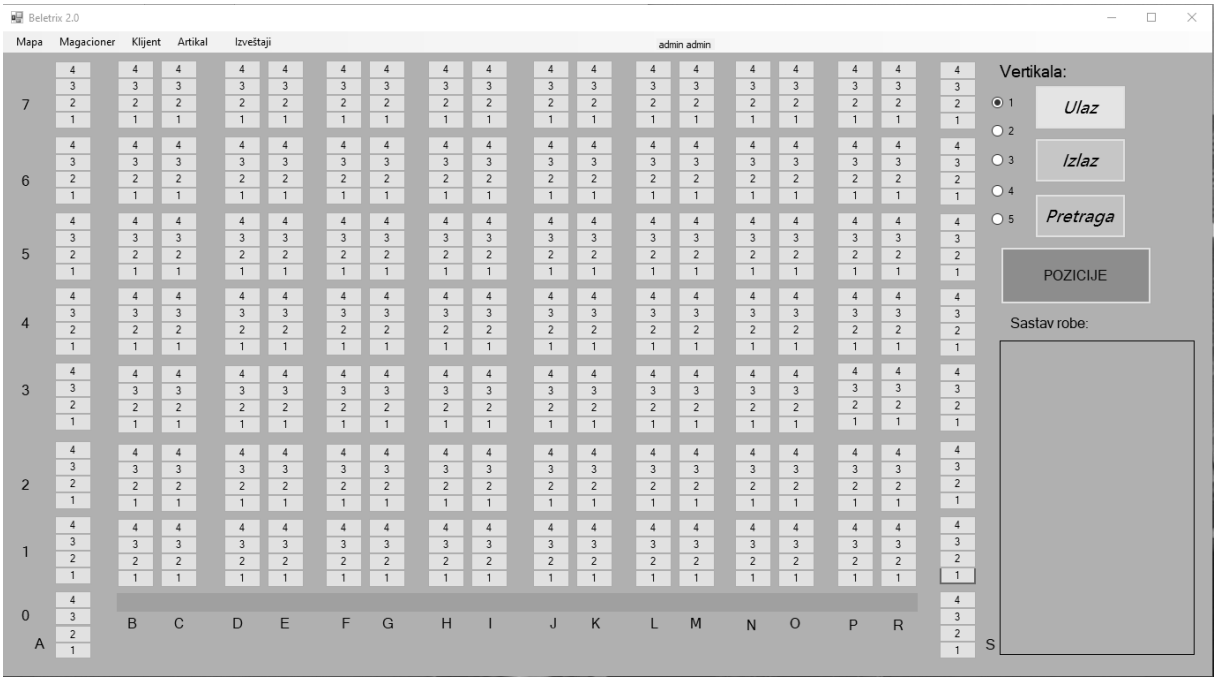


Слика 41 - Структура корисничког интерфејса

12.3 Пројектовање екранских форми

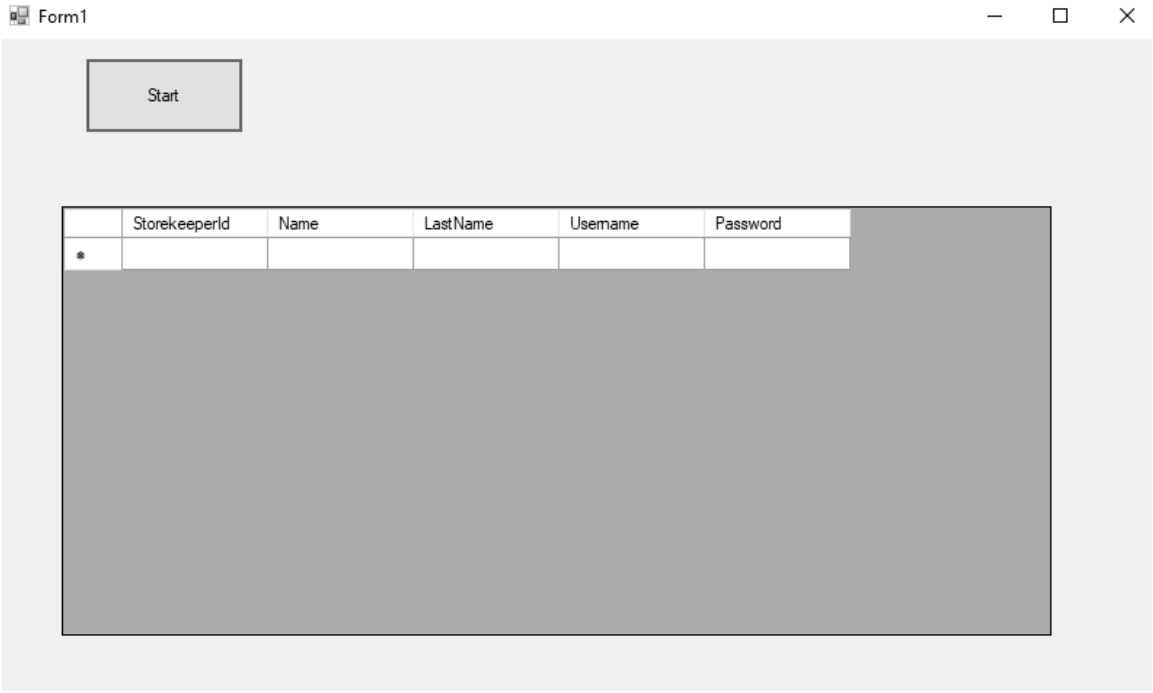
Свака екранска форма је намењена за одређени случај коришћења, тако да приликом клика на дугме ми позивамо да се акција изврши. Контролер корисничког интерфејса прихвата догађаје са екранских форми и прослеђује их контролеру апликационе логике. Сва манипулација података између клијента и сервера захтева рад контролера, док сви излази у улази са којима комуницира корисник софтверског система су везани за екранске форме. Екранске форме спадају у изузетан битан фактор софтверског система који директно комуницира са корисником и представља сам изглед апликације.

Главна екранска форма клијентског дела апликације изгледа овако:



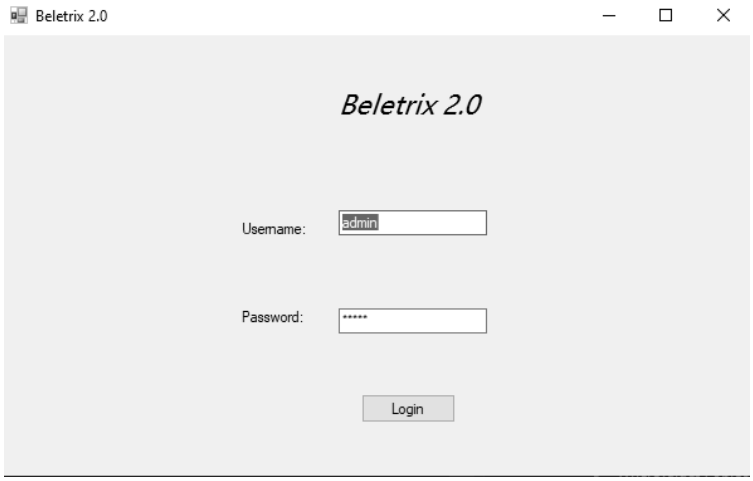
Слика 42 - Главна екранска форма клијента

Главна екранска форма серверског дела апликације је урађена овако:



Слика 43 - Главна екранска форма сервера

Клијент се мора улоговати како би користио апликацију, односно дошао до главне екранске форме. Форма за пријаву изгледа овако:



Слика 44 - Екранска форма за пријаву

12.3.1 СК1: Случај коришћења – Креирање улаза палета

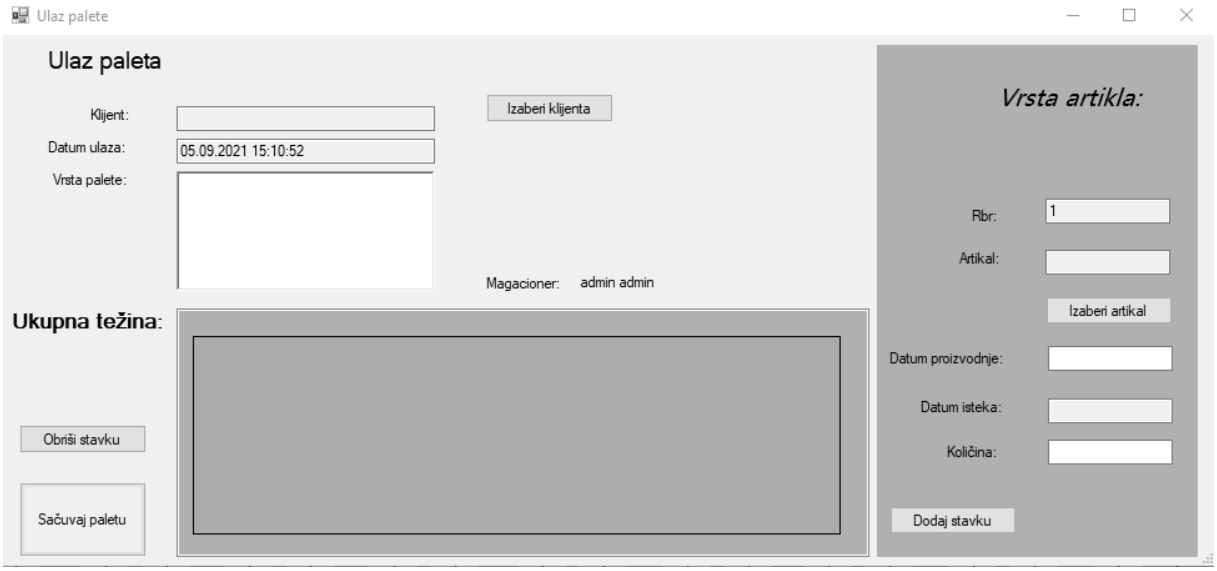
Назив СК
Креирање улаза палета

Актори СК
Магационер

Учесници СК
Магационер и систем (програм)

Предуслов: Систем је укључен и магационер је пријављен под својом шифром. Систем приказује форму за рад са улазом палета у складиште. Учитана је листа свих клијената и артикала са својим подацима.

Систем приказује форму за унос улаза (палете) у складиште.



Слика 45 - СК1 Креирање улаза

Основни сценарио СК

1. Магационер **уноси** податке о улазу. (АПУСО)

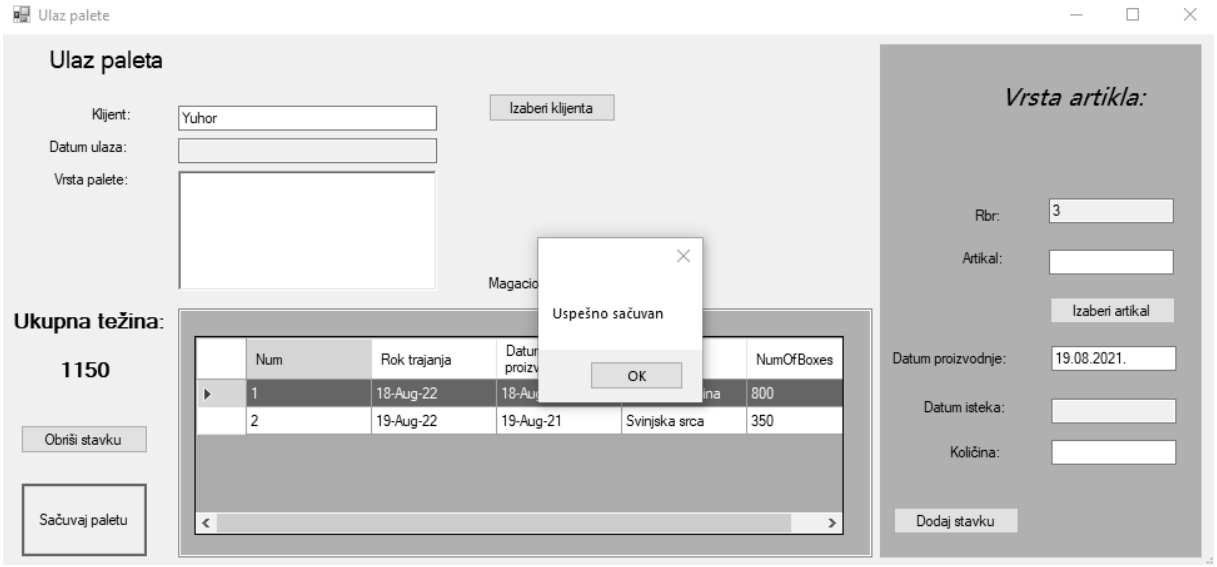
Опис акције: Магационер уноси податке у празна поља као што су клијент, врста палете, артикал, датум производње и количина.

Слика 46 - СК1 Креирање улаза - Основни сценарио - унос података

2. Магационер **контролише** да ли је коректно унео податке о улазу. (АНСО)
3. Магационер **позива** систем да запамти податке о улазу палете. (АПСО)

Опис акције: Магационер притиска дугме "Сачувај палету" и тиме позива одређену системску операцију.

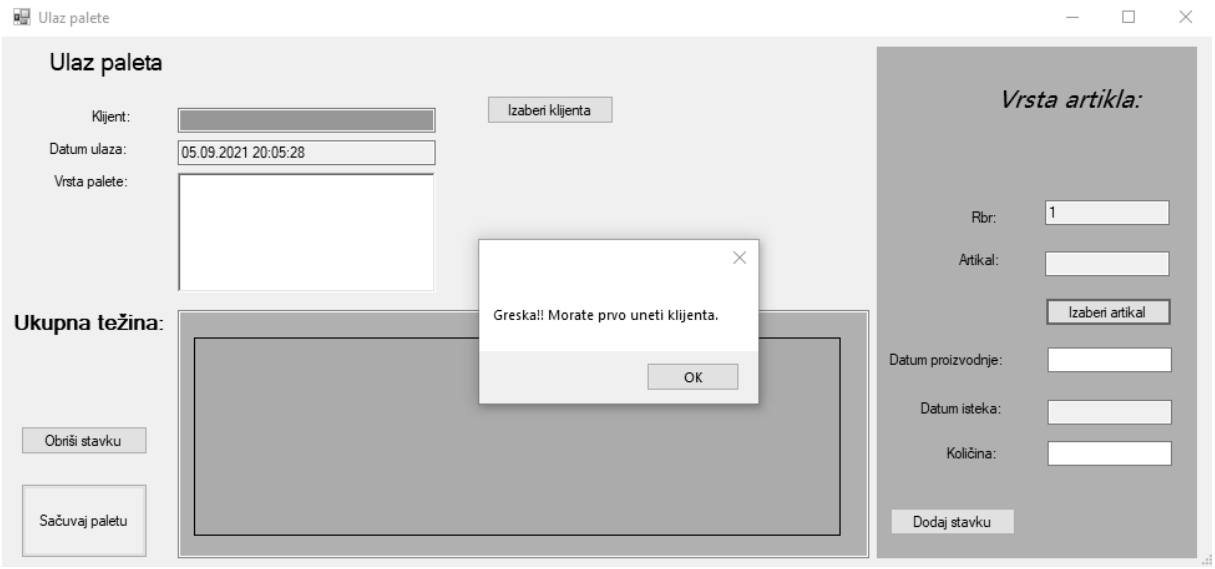
4. Систем **памти** податке о улазној палети. (СО)
5. Систем **приказује** магационеру запамћени улаз и поруку: “Успешно сачуван“. (ИА)



Слика 47 - СК1 Креирање улаза - Основни сценарио - улаз је додат у систем

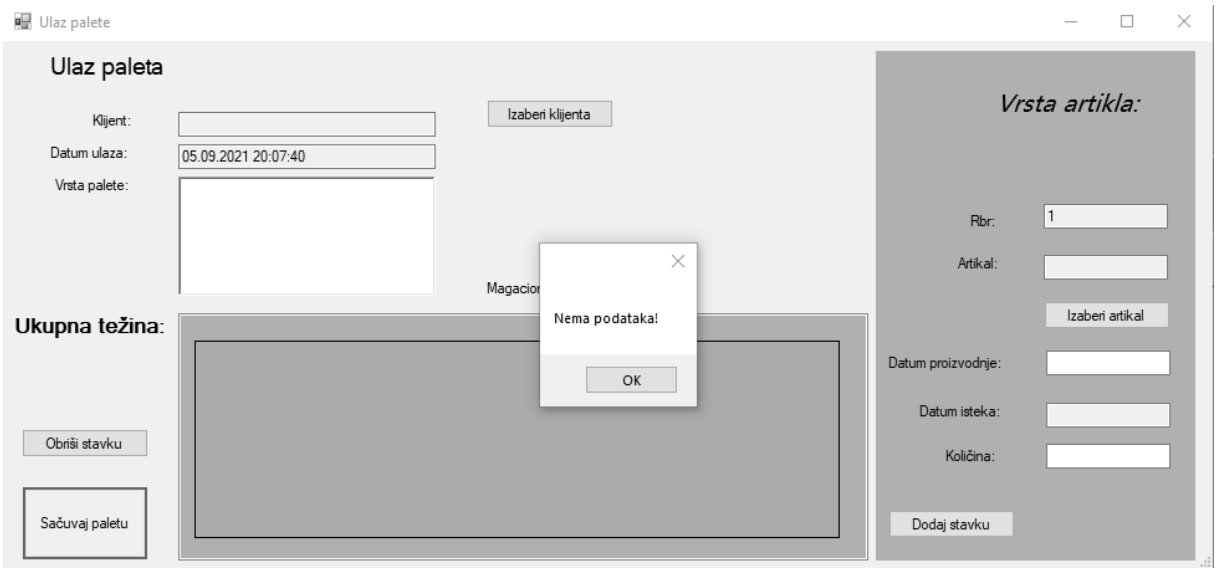
Алтернативни сценарио:

5.1 Уколико систем не може да креира улаз он приказује магационеру поруку: “Грешка!!! Морате прво унети клијента.”. Прекида се извршење сценарија. (ИА)



Слика 48 - СК1 Креирање улаза - Алтернативни сценарио 1

5.2 Уколико систем не може да запамти податке о улазу палете он приказује магационеру поруку “Нема података”. (ИА)



Слика 49- СК1 Креирање улаза - Алтернативни сценарио 2

12.3.2 СК2: Случај коришћења – Креирање позиције у складишту

Назив СК
Креирање позиције

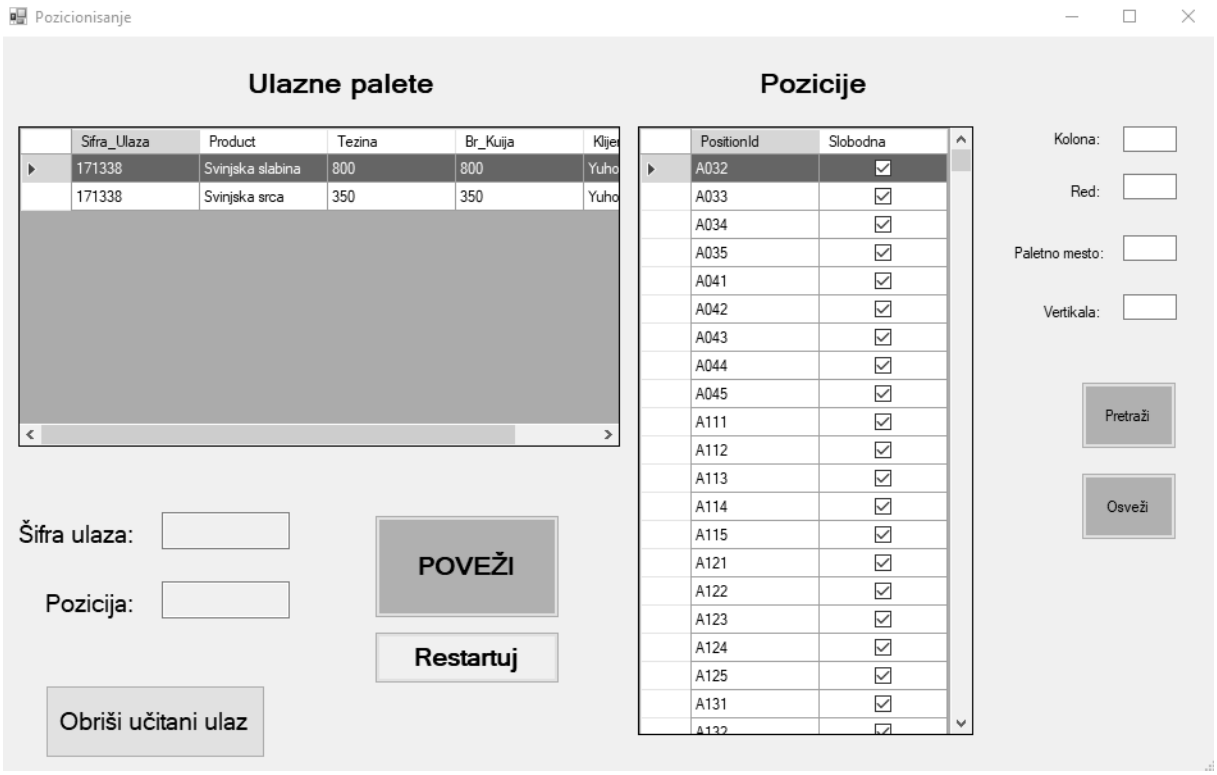
Актори СК
Магационер

Учесници СК

Магационер и систем (програм)

Предуслов: Систем је укључен и магационер је пријављен под својом шифром. Систем приказује форму за рад са позицијом. Учитани су подаци о улазу и позицијама.

Систем приказује форму за креирање позиције у складишту.

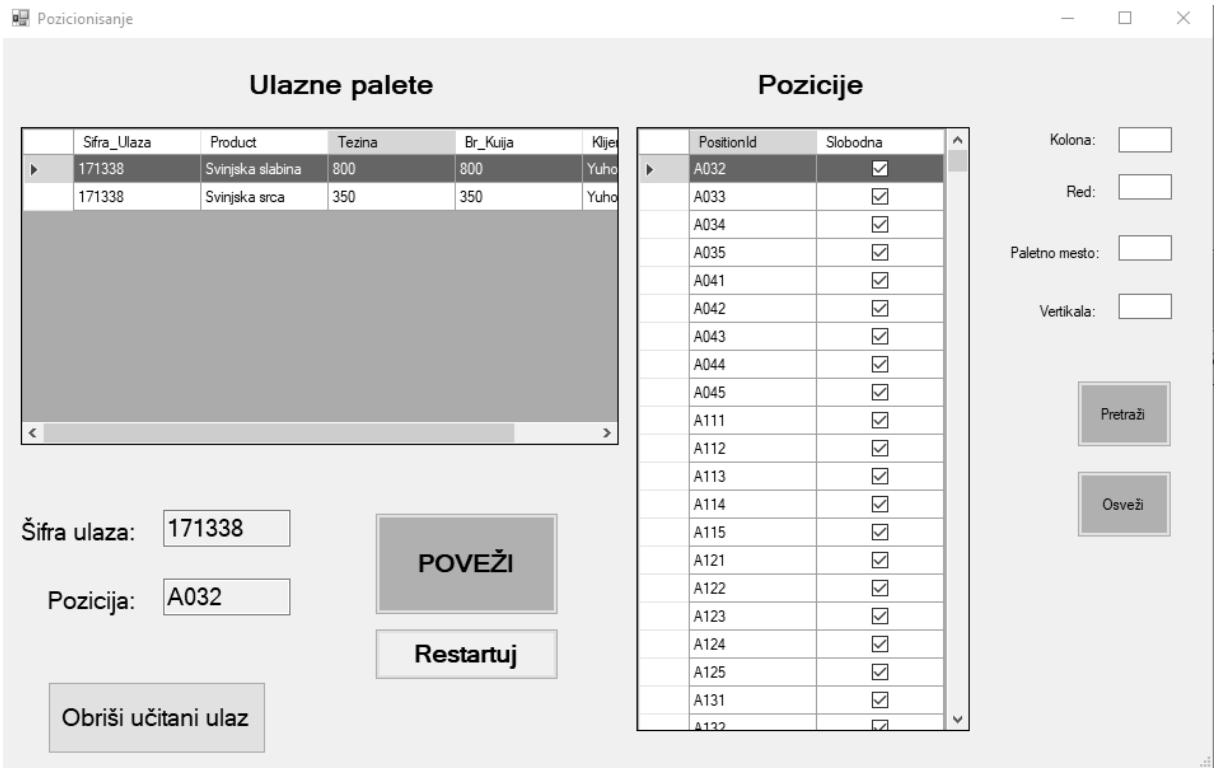


Слика 50 - СК2 Креирање позиције

Основни сценарио СК:

1. Магационер **позива** систем да креира позицију. (АПСО)
2. Систем **креира** позицију. (СО)
3. Систем **приказује** форму“. (ИА)
4. Магационер **уноси** податке о позицији. (АПУСО)

Опис акције: Магационер уноси податке у празна поља као што су шифра улаза и позиција.

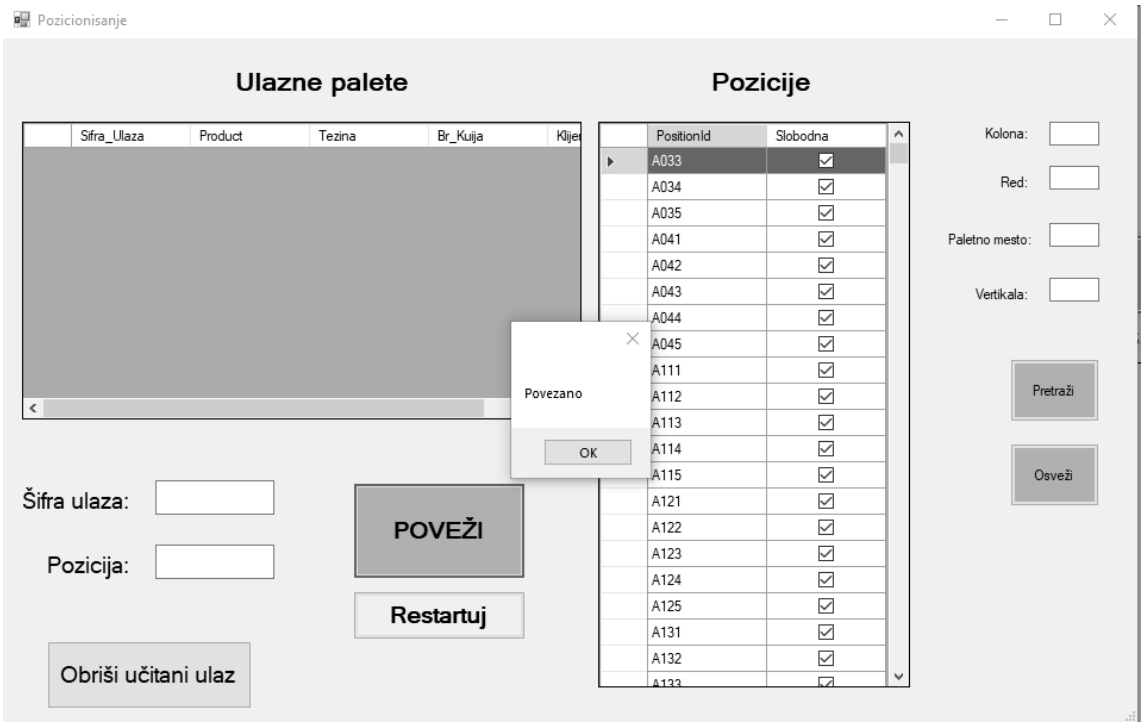


Слика 51 - СК2 Креирање позиције - Основни сценарио - унос података

5. Магационер **контролише** да ли је коректно унео податке о позицији. (АНСО)
6. Магационер **позива** систем да запамти податке о позицији. (АПСО)

Опис акције: Магационер притиска дугме "Повежи" и тиме повезује слободну позицију са улазном палетом.

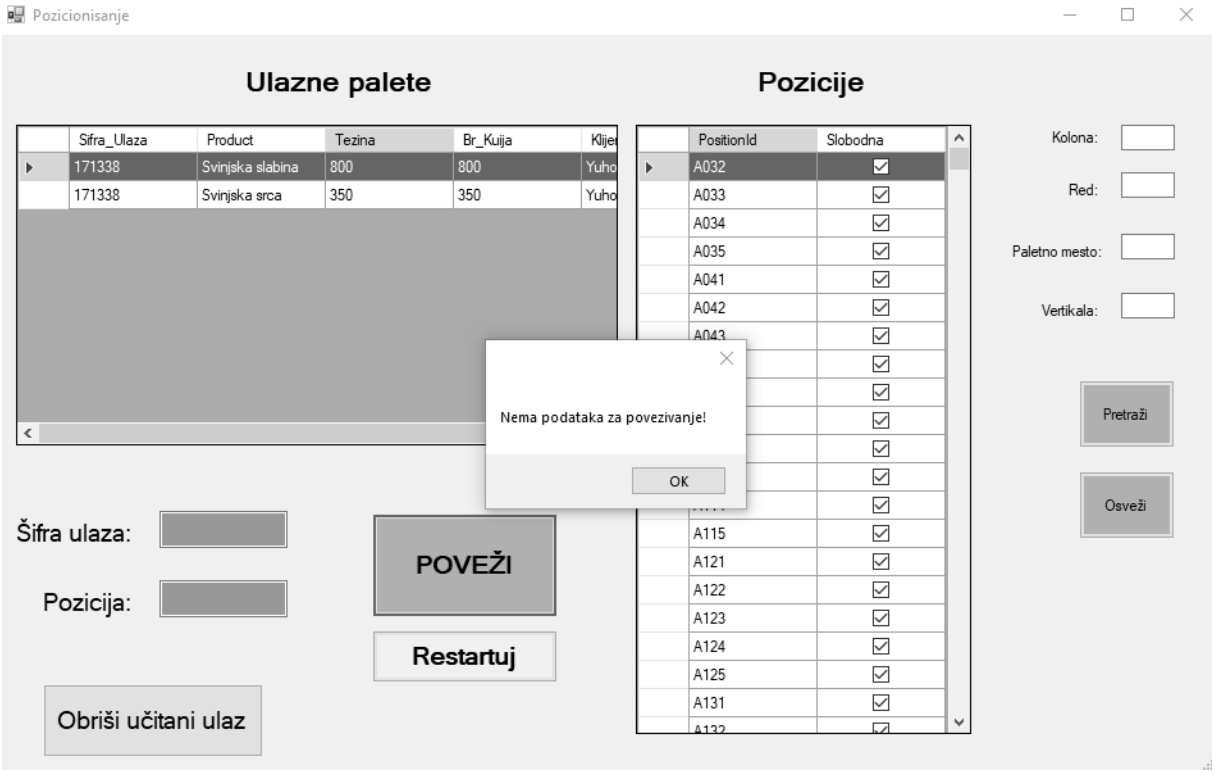
7. Систем **памти** податке о позицији. (СО)
8. Систем **приказује** магационеру запамћену позицију и поруку: “Повезано“. (ИА)



Слика 52 - СК2 Креирање позиције - Основни сценарио - успешно повезивање

Алтернативни сценарио:

8.1 Уколико систем не може да запамти податке о позицији он приказује магационеру поруку “Нема података за повезивање”. (ИА)



Слика 53 - СК2: Креаирање позиције - Алтернативни сценарио - неуспешно повезивање позиције и улаза

12.3.3 СК3: Случај коришћења – Претраживање робе по палетама

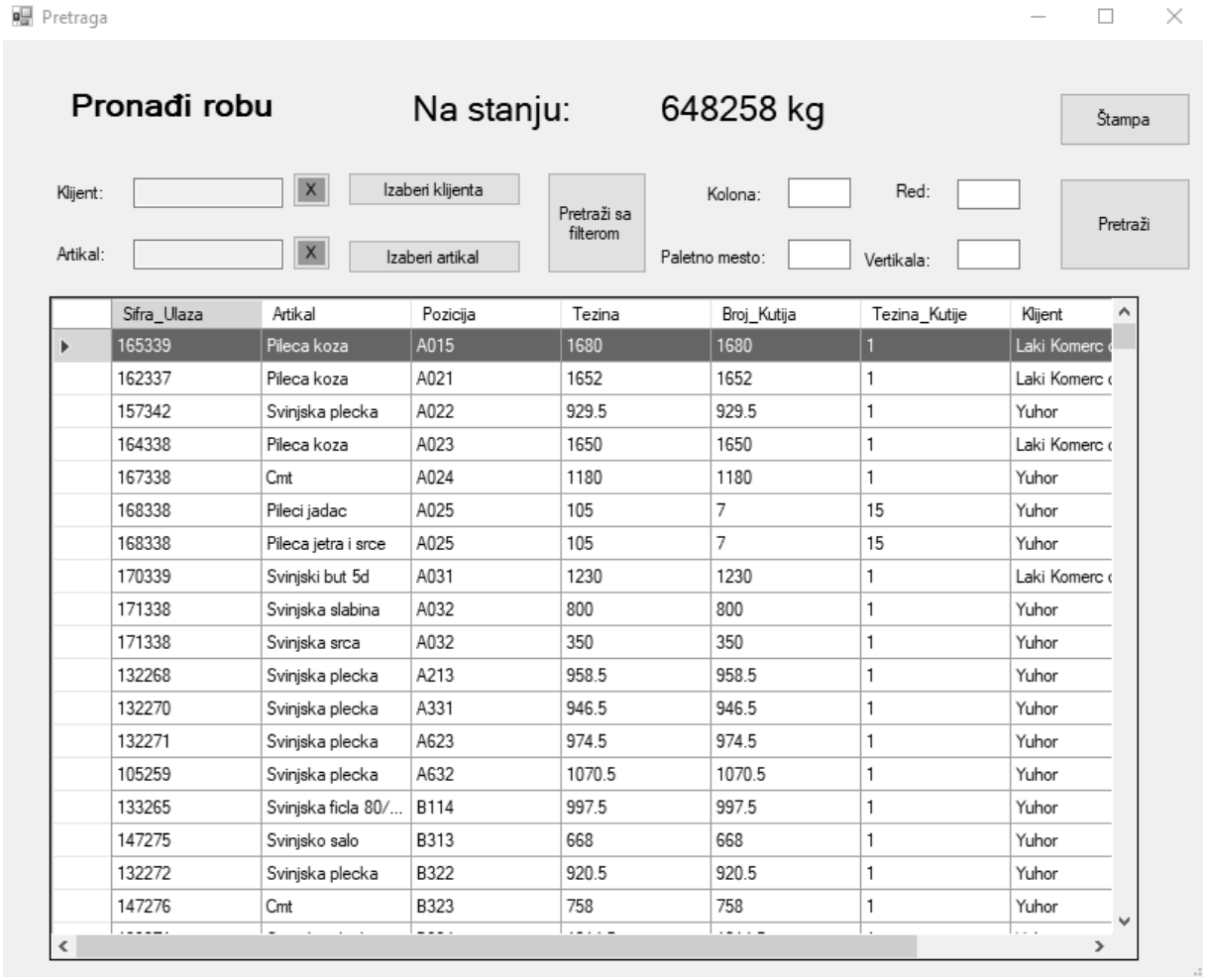
Назив СК
Претраживање робе

Актори СК
Магационер

Учесници СК
Магационер и систем (програм)

Предуслов: Систем је укључен и магационер је пријављен под својом шифром. Систем приказује форму за рад са робом. Учитана је листа свих улазних ставки у складишту.

Систем приказује форму за претраживање улаза (палете) у складиште.

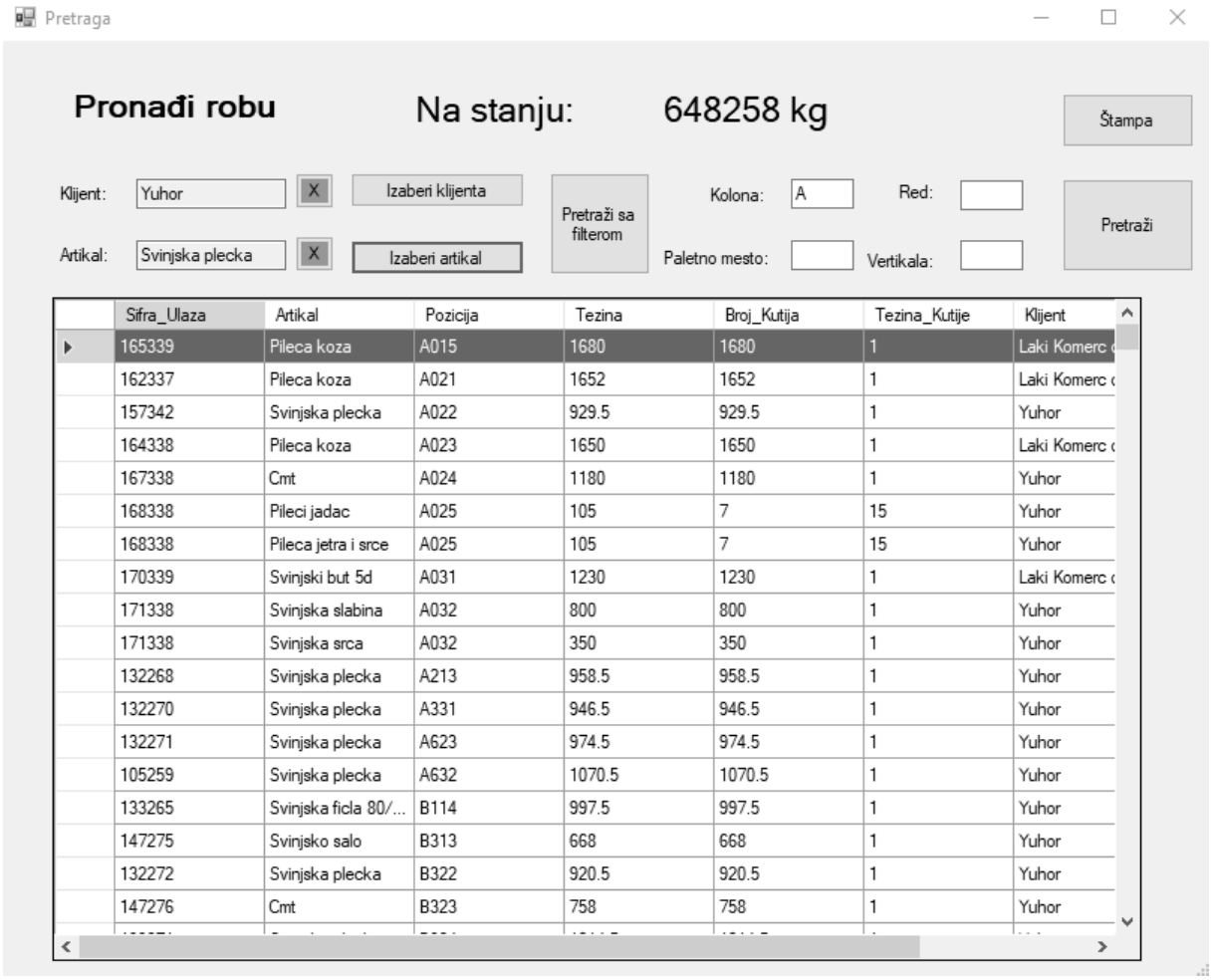


Слика 54 - СК3: Претраживање робе

Основни сценарио СК

1. Магационер **уноси** критеријум по којој претражује робу. (АПУСО)

Опис акције: Магационер уноси податке као што су колона, ред, палетно место и вертикала како би нашао жељену позицију. Исто тако може да сортира према клијенту и артиклу.

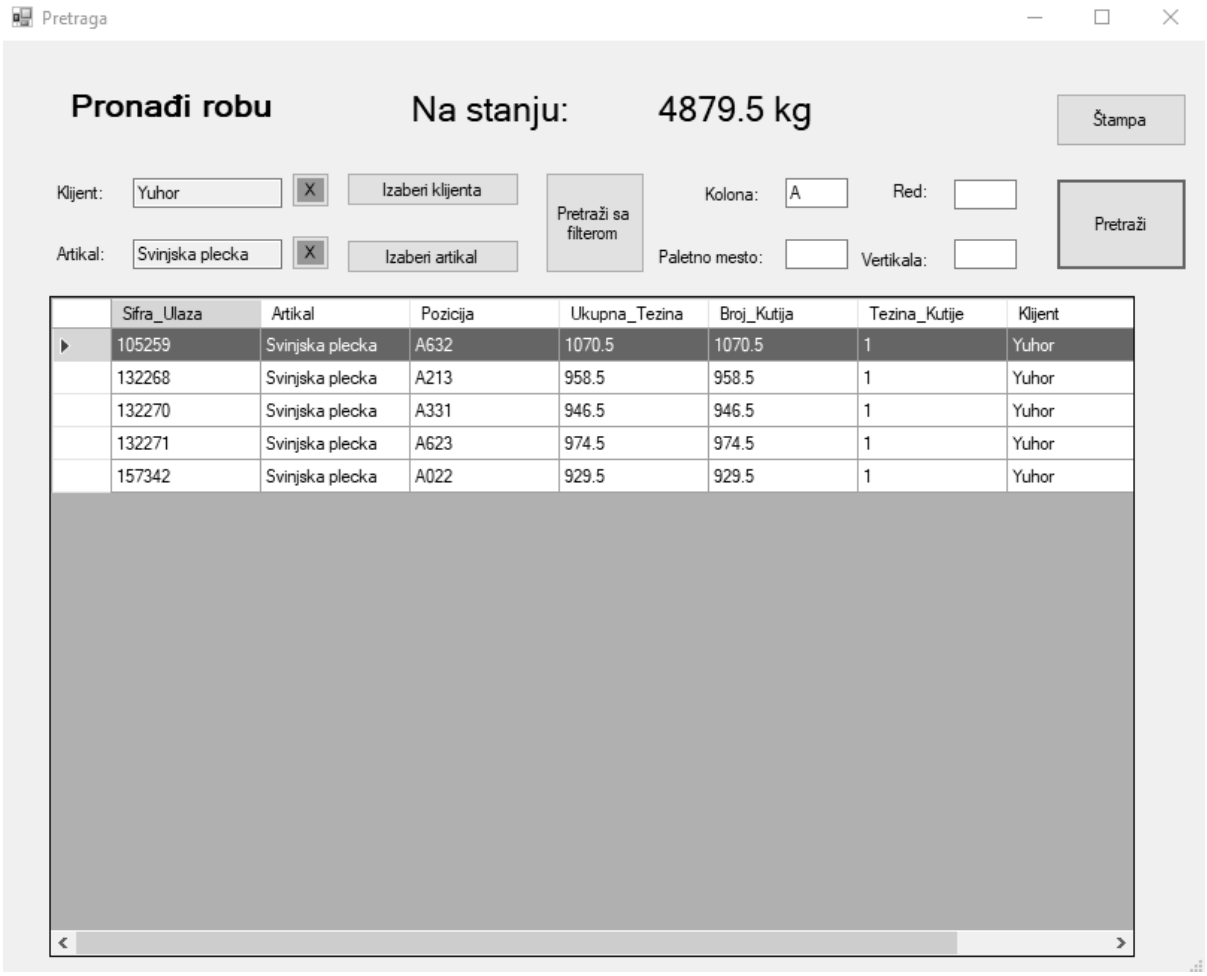


Слика 55 - СК3: Претраживање робе - Основни сценарио - одабир клијента, робе и позиције

2. Магационер **позива** систем да нађе робу по задатом критеријуму. (АПСО)

Опис акције: Магационер притиска дугме "Претражи са филтером" која претражује по клијенту и артиклу или дугме "'Претражи" која обухвата и случај са позицијом.

3. Систем **тражи** робу по задатом критеријуму. (СО)
4. Систем приказује магационеру податке о роби коју је понашао. (ИА)



Слика 56 - СК3: Претраживање робе - Основни сценарио - претражена роба

Алтернативни сценарио:

1.1 Уколико систем не може да нађе робу он приказује празну табелу.(ИА).

Слика 57 - СК3: Претраживање робе - Алтернативни сценарио

12.3.4 СК4: Случај коришћења – Излаз целе палете

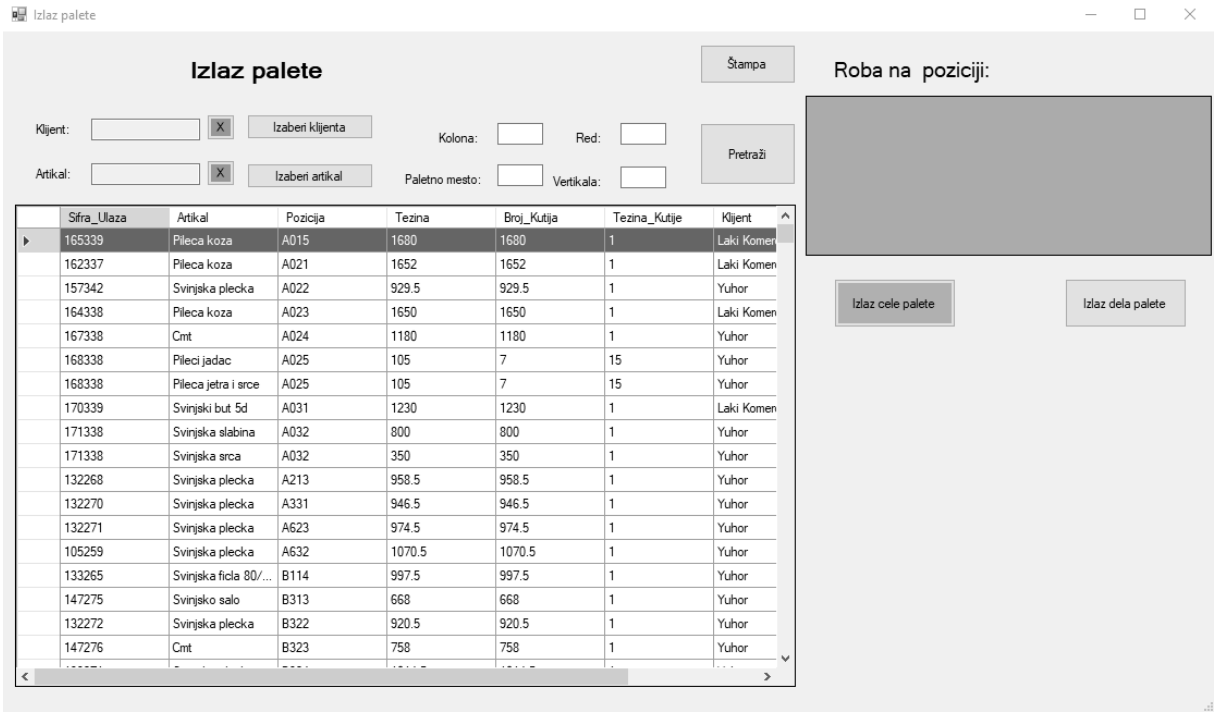
Назив СК
Промена палете (улаза)

Актори СК
Магационер

Учесници СК
Магационер и систем (програм)

Предуслов: Систем је укључен и Магационер је пријављен под својом шифром. Систем приказује форму за рад са излазом целе палете. Учитана је листа свих палета са својим позицијама које се налазе у систему.

Систем приказује форму за излаз целе палете.

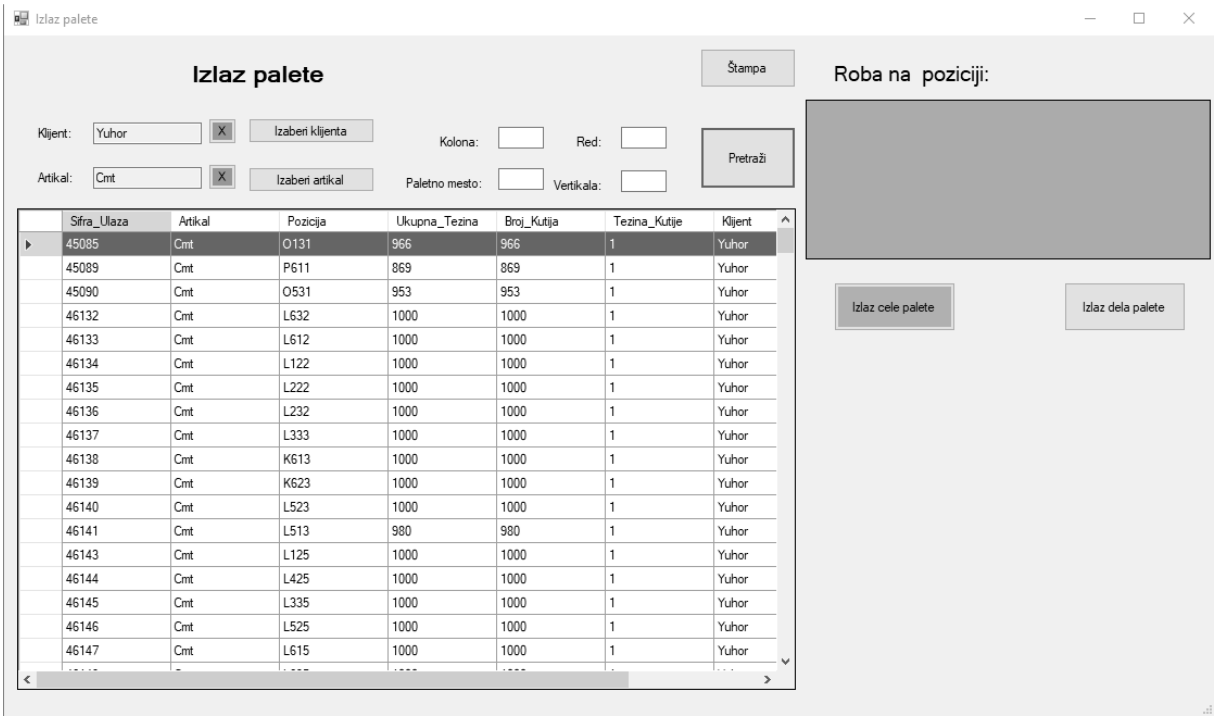


Слика 58 - СК4: Излаз целе палете

Основни сценарио СК

1. Магационер **уноси** вредност по којој претражује целу излазну палету. (АПУСО)

Опис акције: Магационер уноси податке као што су колона, ред, палетно место и вертикалу како би нашао жељену позицију. Исто тако може да сортира према клијенту и артиклу.

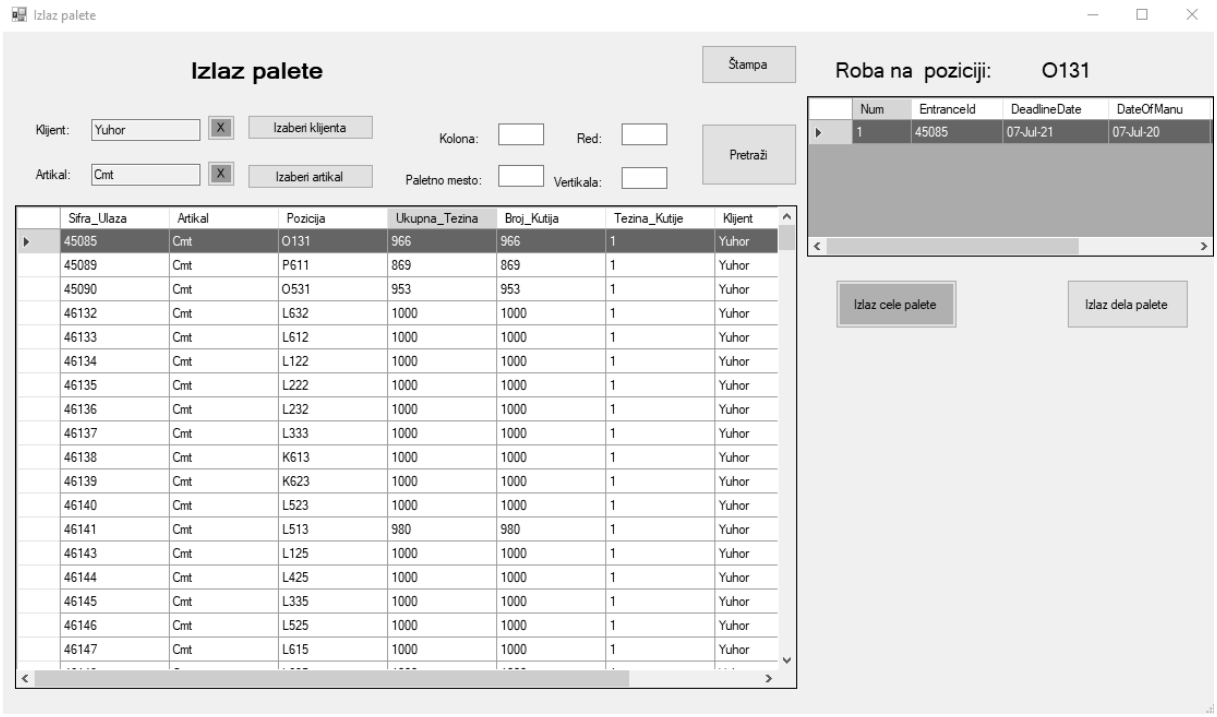


Слика 59- СК4: Излаз целе палете - Основни сценарио - претрага робе

2. Магационер **позива** систем да нађе целу излазну палету по задатој вредности. (АПСО)

Опис акције: Магационер притиска дугме "Претражи са филтером" која претражује по клијенту и артиклу или дугме "'Претражи" која обухвата и случај са позицијом.

3. Систем **тражи** целу излазну палету по задатој вредности. (СО)
4. Систем приказује Магационеру целу излазну палету. (ИА)



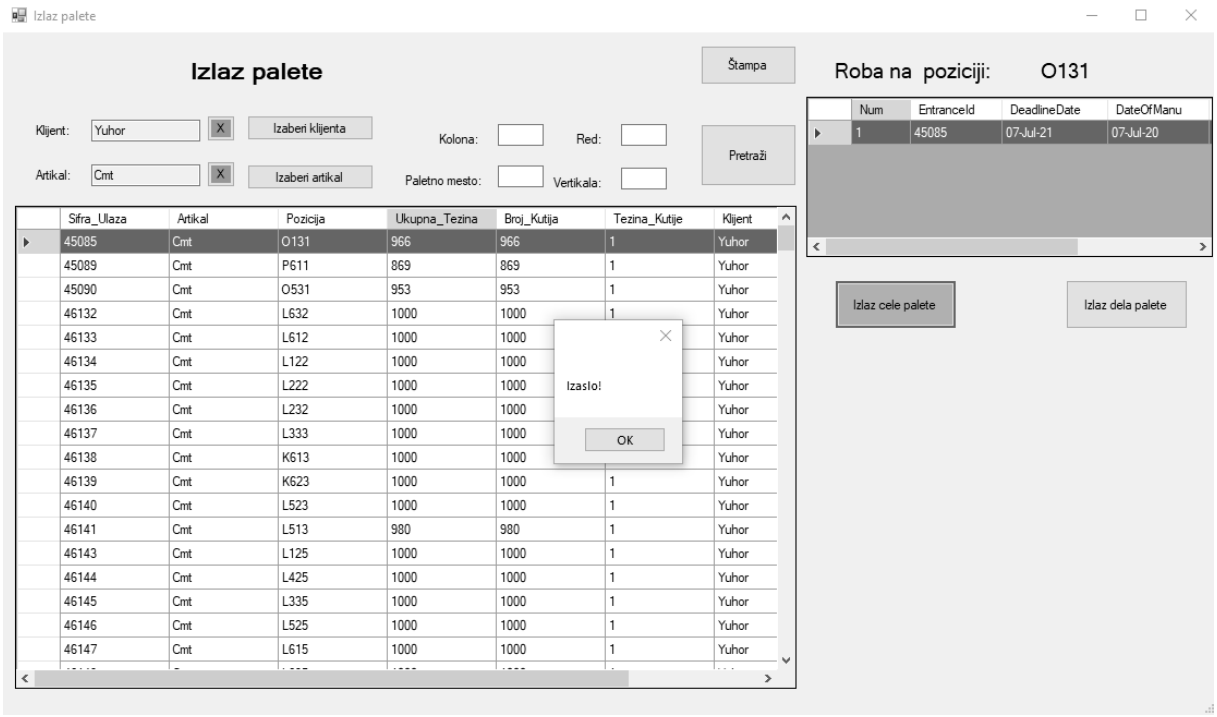
Sifra_Ulaza	Artikal	Pozicija	Ukupna_Tezina	Broj_Kutija	Tezina_Kutije	Klijent
45085	Cmt	O131	966	966	1	Yuhor
45089	Cmt	P611	869	869	1	Yuhor
45090	Cmt	O531	953	953	1	Yuhor
46132	Cmt	L632	1000	1000	1	Yuhor
46133	Cmt	L612	1000	1000	1	Yuhor
46134	Cmt	L122	1000	1000	1	Yuhor
46135	Cmt	L222	1000	1000	1	Yuhor
46136	Cmt	L232	1000	1000	1	Yuhor
46137	Cmt	L333	1000	1000	1	Yuhor
46138	Cmt	K613	1000	1000	1	Yuhor
46139	Cmt	K623	1000	1000	1	Yuhor
46140	Cmt	L523	1000	1000	1	Yuhor
46141	Cmt	L513	980	980	1	Yuhor
46143	Cmt	L125	1000	1000	1	Yuhor
46144	Cmt	L425	1000	1000	1	Yuhor
46145	Cmt	L335	1000	1000	1	Yuhor
46146	Cmt	L525	1000	1000	1	Yuhor
46147	Cmt	L615	1000	1000	1	Yuhor

Слика 60 - СК4: Излаз целе палете - Основни сценарио – приказ излазне палете

5. Магационер **контролише** да ли је коректно унео податке о излазној палети. (АНСО)
6. Магационер **позива** систем да запамти податке о излазној палети. (АПСО)

Опис акције: Магационер притиска дугме "Излаз целе палете" која избацује целу палету из система.

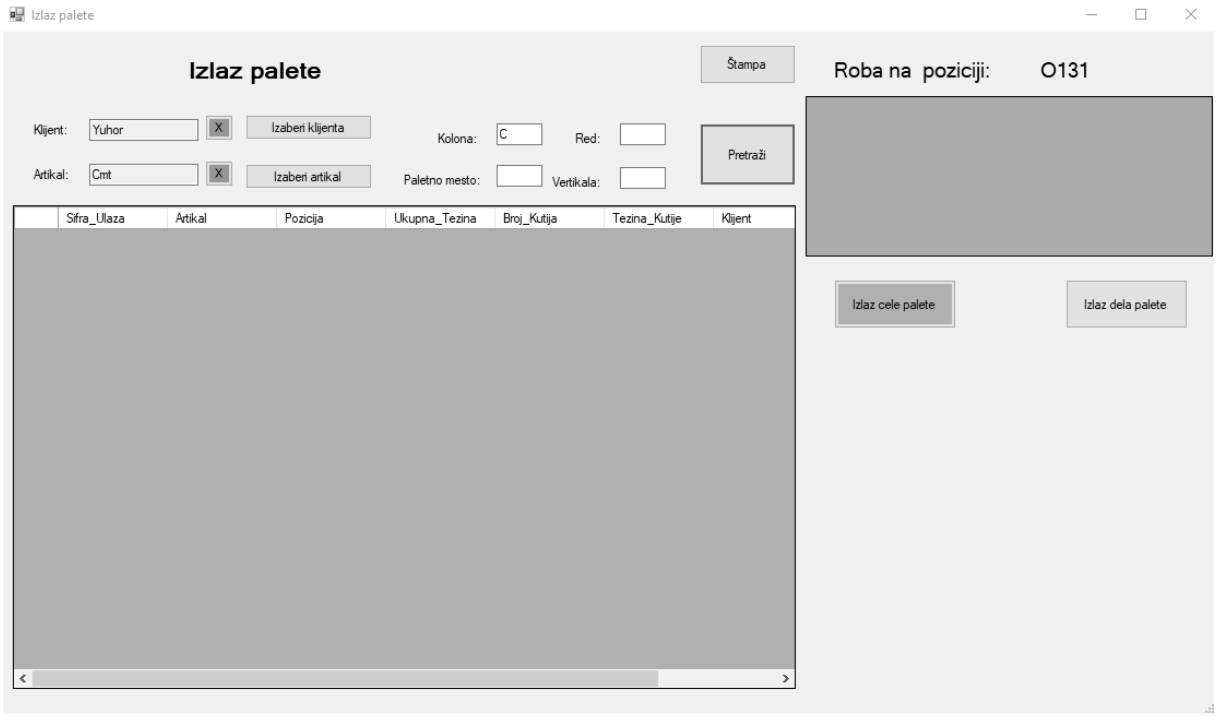
7. Систем **памти** податке о излазној палети. (СО)
8. Систем **приказује** Магационеру запамћени излаз целе палете и поруку: “Изашло” (ИА)



Слика 61 - СК4: Излаз целе палете - Основни сценарио - излаз целе палете

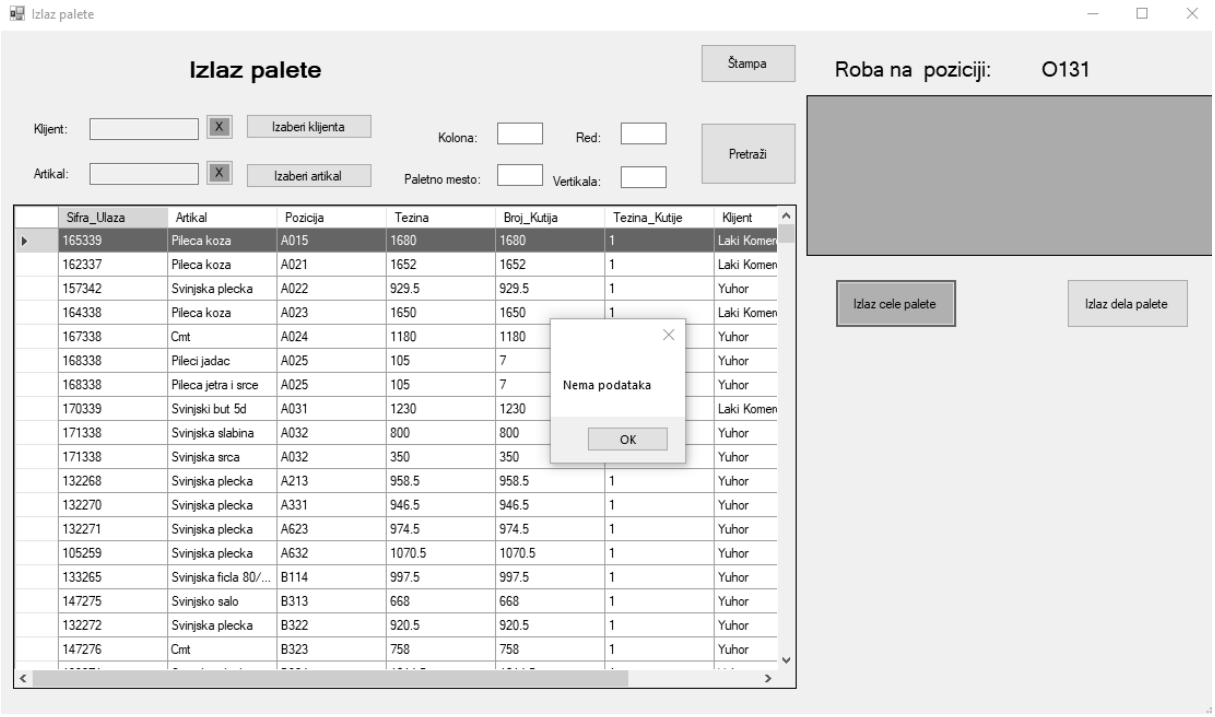
Алтернативни сценарио:

1.1 Уколико систем не може да нађе целу излазну палету, приказаће празну табелу.Прекида се извршење сценариа. (ИА)



Слика 62 - СК4: Излаз целе палете - Алтернативни сценарио - лоша претрага

7.1 Уколико систем не може да запамти податке о излазној палети он приказује Магационеру поруку “Нема података”.



Слика 63 - СК4: Излаз целе палете - Алтернативни сценарио - нема података

12.3.5 СК5: Случај коришћења – Излаз дела палете

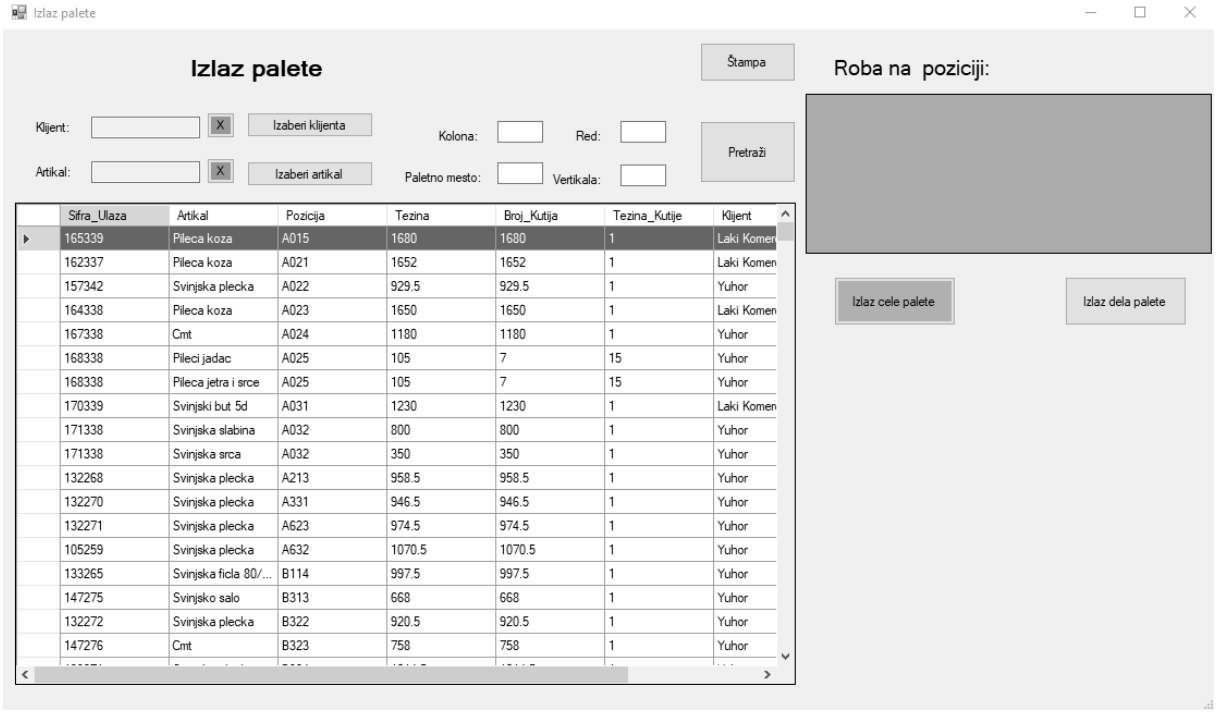
Назив СК
Креирање излазне палете

Актори СК
Магационер

Учесници СК
Магационер и систем (програм)

Предуслов: Систем је укључен и Магационер је пријављен под својом шифром. Систем приказује форму за рад са излазом дела палете. Учитана је листа свих ставки улаза у складишту.

Систем приказује форму за излаз целе (дела) палете.



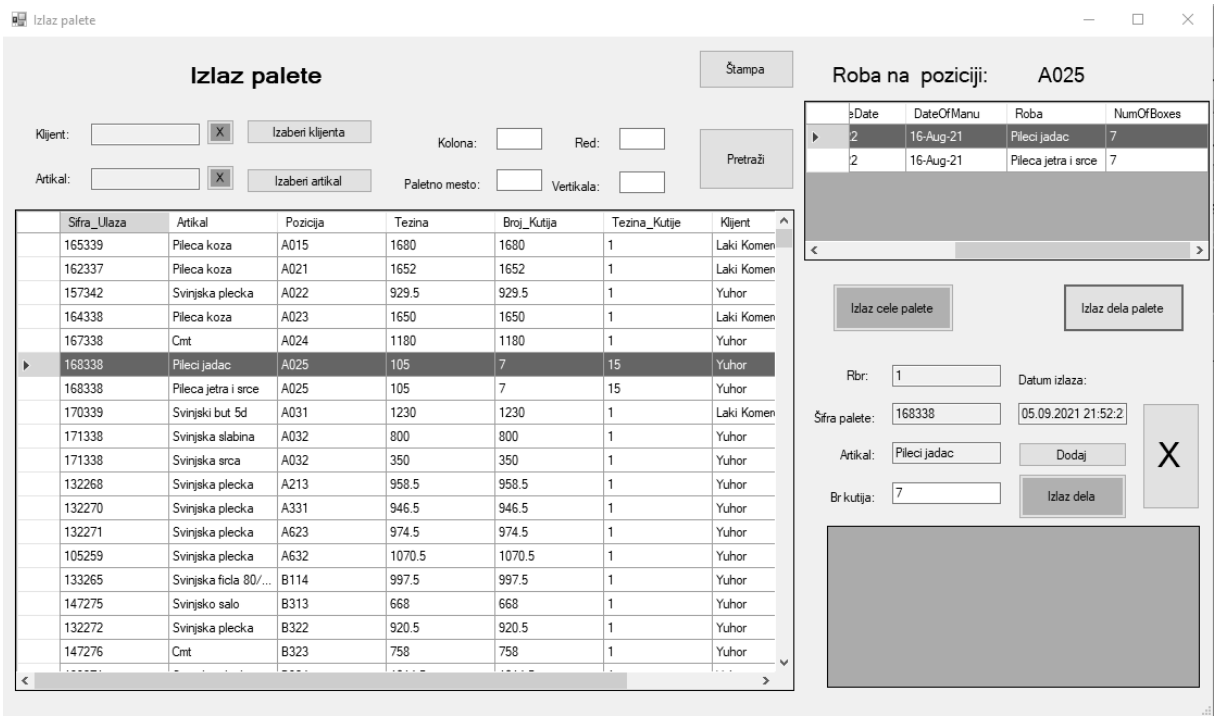
Слика 64 - СК5: Излаз дела палете

Основни сценарио СК

1. Магационер **позива** систем да креира део излазне палете. (АПСО)

Опис акције: Магационер одабира одређену палету коју намерава да избаци и потом му се отварају све ставке које се односе на ту палету.

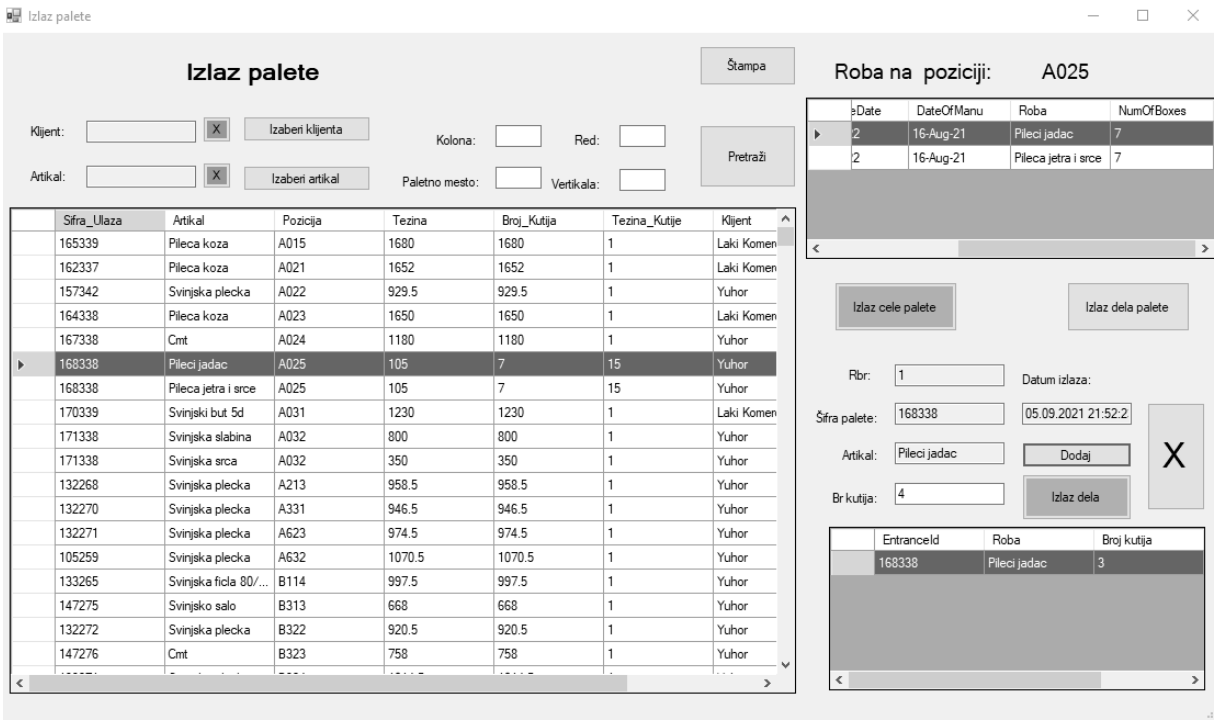
2. Систем **креира** део излазне палете. (СО)
3. Систем **приказује** Магационеру део излазне палете (ИА)



Слика 65 - СК5: Излаз дела палете - Основни сценарио - приказ излаза дела палете

4. Магационер **уноси** податке за део излазне палете. (АПУСО)

Опис акције: Магационер уноси податке као што су ставка палете и број кутија који излази за одабрану ставку.



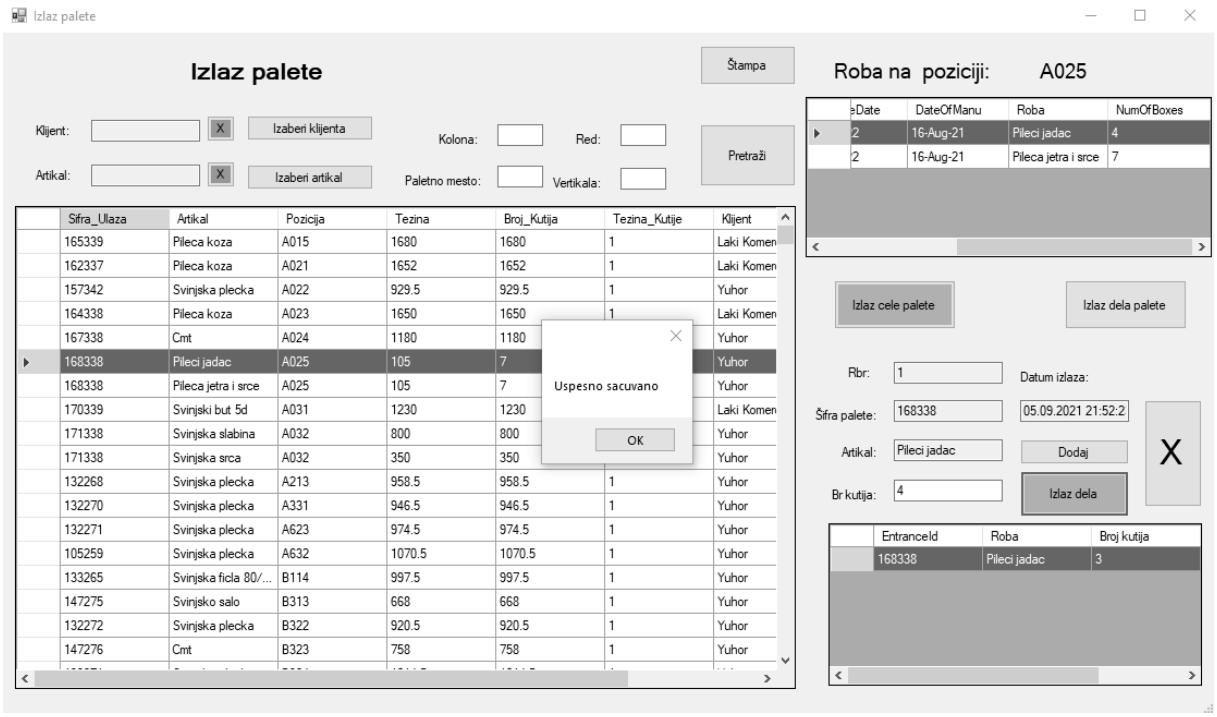
Слика 66 - СК5: Излаз дела палете - Основни сценарио - унос излаза дела палете

5. Магационер **контролише** да ли је коректно унео податке у део излазне палете. (АНСО)

6. Магационер **позива** систем да запамти податке о излазној палети. (АПСО)

Опис акције: Магационер притиска дугме "Излаз дела" који прави излаз за ставке које су унешене у табели
.

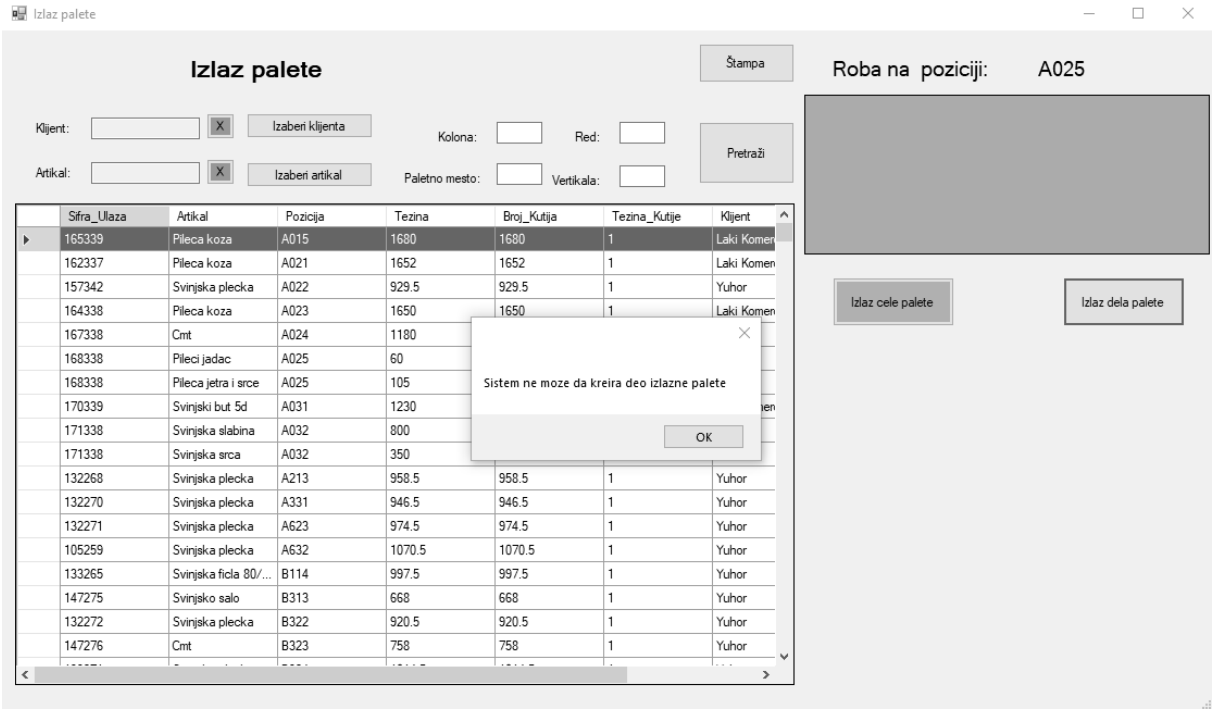
7. Систем **памти** податke о излазној палети. (СО)
8. Систем **приказује** Магационеру запамћени излаз дела палете и поруку: “Успешно сачувано“. (ИА)



Слика 67 - СК5: Излаз дела палете - Основни сценарио

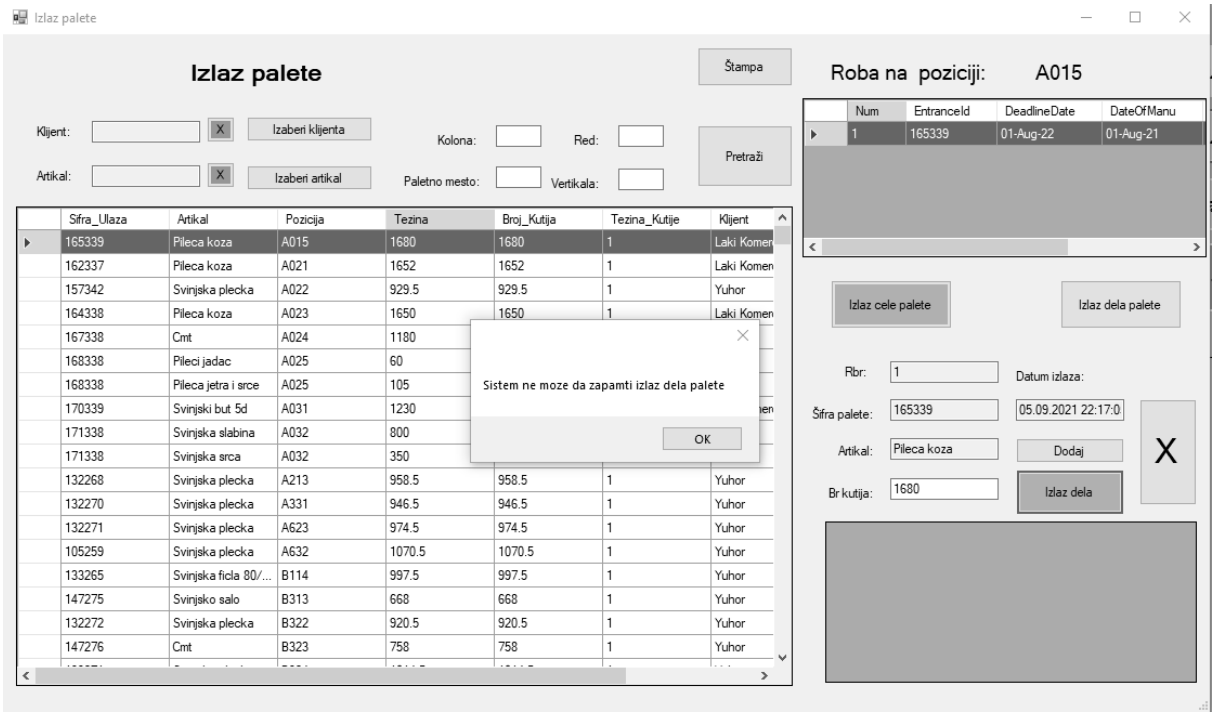
Алтернативни сценарио:

3.1 Уколико систем не може да креира део излазне палете он приказује Магационеру поруку: “Систем не може да креира део излазне палете”. Прекида се извршење сценариа. (ИА)



Слика 68 - СК5: Излаз дела палете - Алтернативни сценарио 1

1.1 Уколико систем не може да запамти податке о излазној палети он приказује Магационеру поруку “Систем не може да запамти излаз дела палете”. (ИА)



Слика 69 - СК5: Излаз дела палете - Алтернативни сценарио 2

12.3.6 СК6: Случај коришћења – Креирање робе

Назив СК

Креирање робе

Актори СК
Магационер

Учесници СК
Магационер и систем (програм)

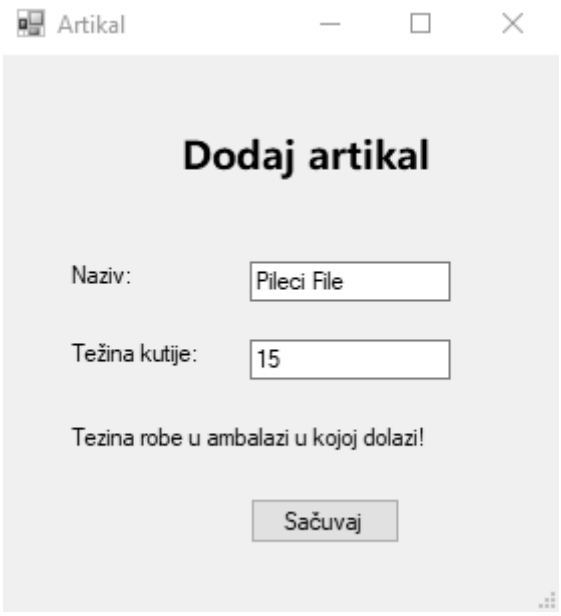
Предуслов: Систем је укључен и Магационер је пријављен под својом шифром. Систем приказује форму за рад са робом.

Слика 70 – СК6: Креирање робе - почетна форма

Основни сценарио СК:

1. Магационер **уноси** робу. (АПУСО)

Опис акције: Магационер уноси податке као што су назив робе и колика је очекивана тежина једне јединице тог производа.

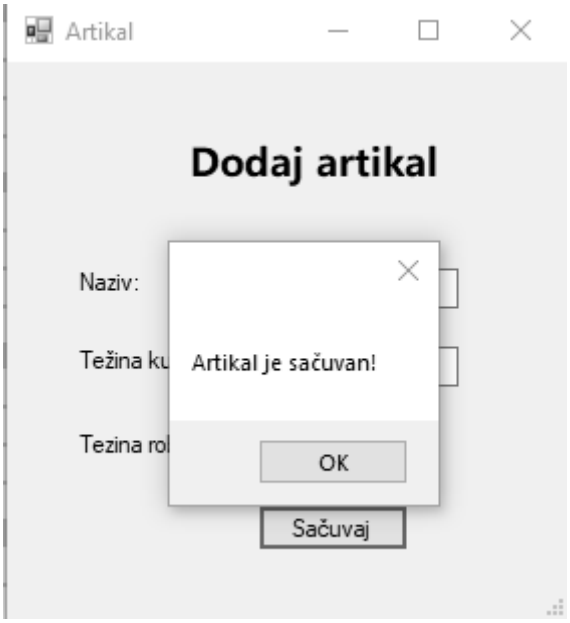


Слика 71 – СК6: Креирање робе - Основни сценарио - унос података

2. Магационер **контролише** да ли је коректно унео робу. (АНСО)
3. Магационер **позива** систем да запамти робу. (АПСО)

Опис акције: Магационер притиском на дугме "Сачувај" позива системску операцију коју чува робу.

4. Систем **памти** робу. (СО)
5. Систем **приказује** Магационеру запамћени податак и поруку: “Артикал је сачуван“. (ИА)



Слика 72 - СК7: Креирање робе - Основни сенарио - податак је сачуван

Алтернативна сценариа:

5.1 Уколико систем не може да запамти податке о роби. Обојиће поље за унос црвеном бојом. (ИА)

Слика 73 - СК7: Креирање података - Алтернативни сценарио

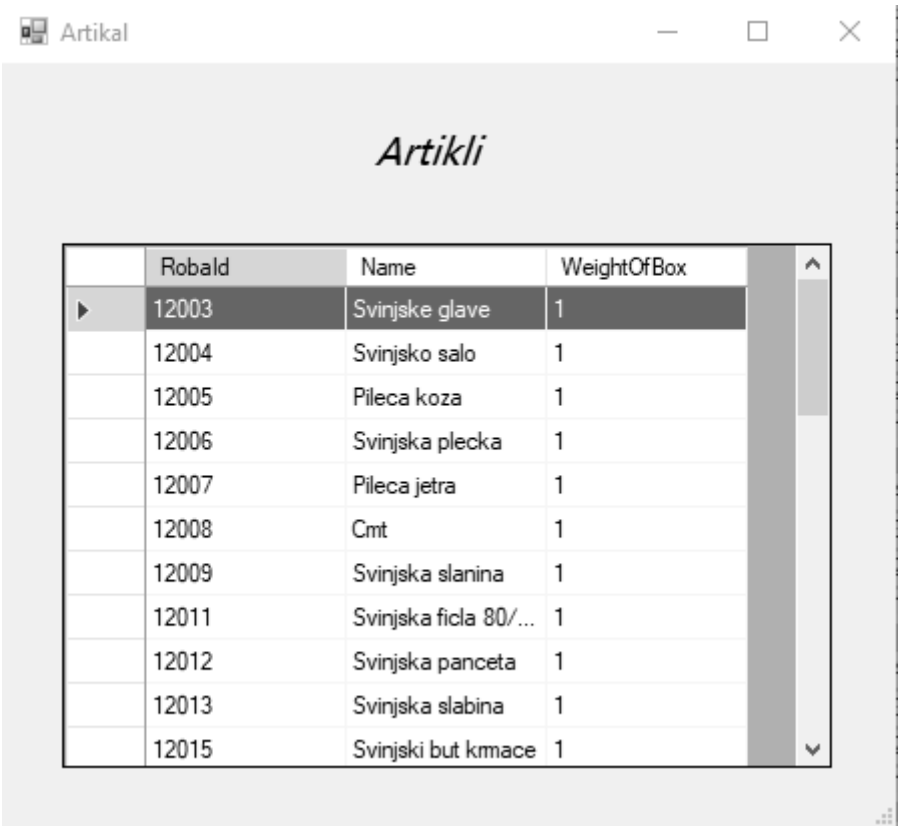
12.3.7 СК7: Случај коришћења – Промена робе

Назив СК
Промена робе

Актори СК
Магационер

Учесници СК
Магационер и систем (програм)

Предуслов: Систем је укључен и Магационер је пријављен под својом шифром. Систем приказује форму за рад са робом. Учитана је листа Робе.



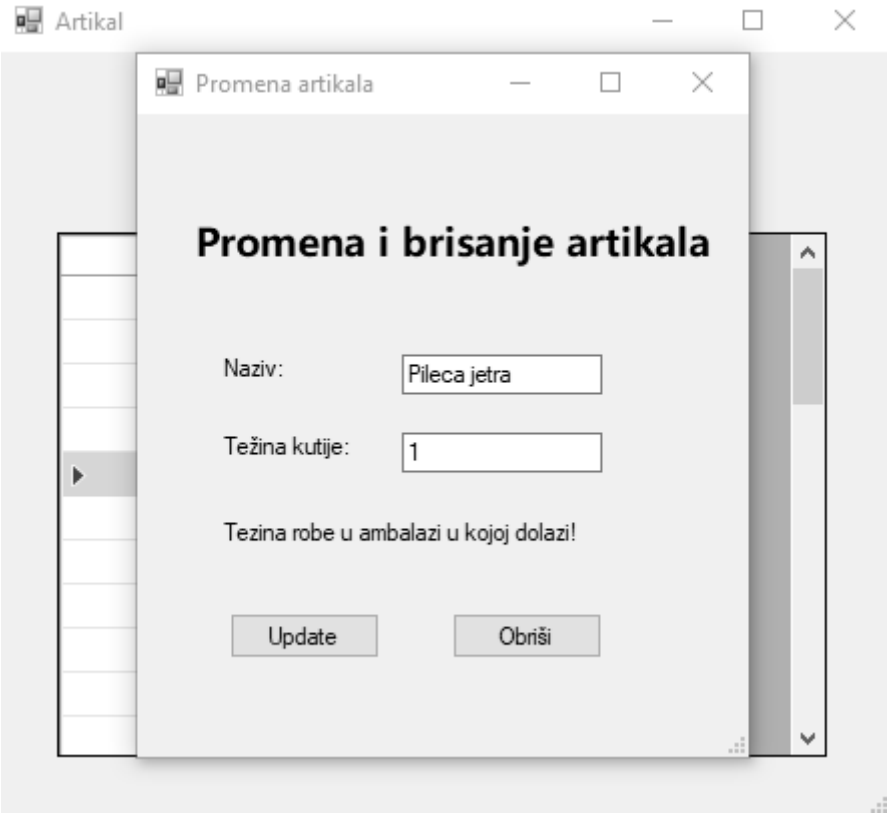
Слика 74 - СК6: Промена robe

Основни сценарио СК

1. Магационер **позива** систем да нађе робу по одабраној вредности. (АПСО)

Опис акције: Магационер бира робу коју намерава да ажурира.

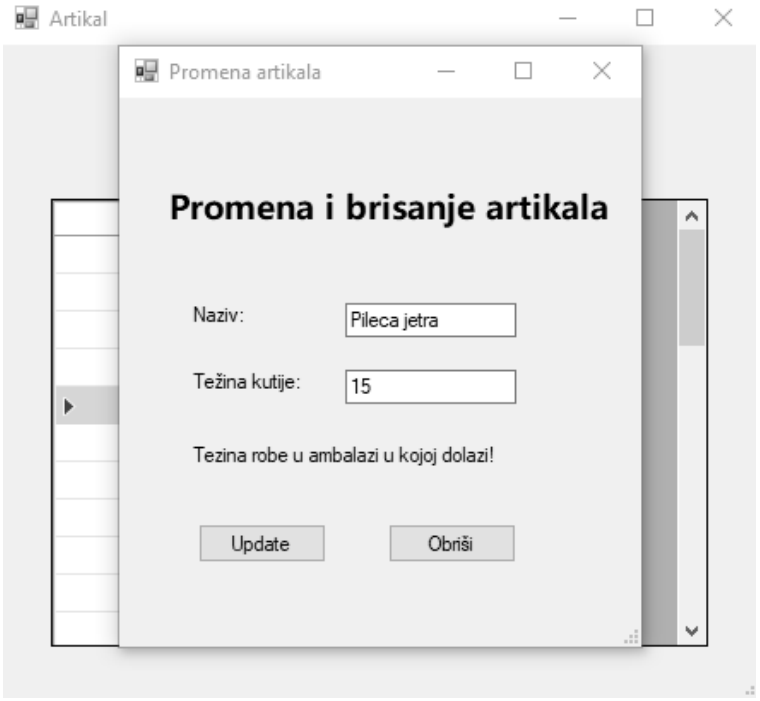
2. Систем **тражи** робу по задатој вредности. (СО)
3. Систем приказује Магационеру робу. (ИА)



Слика 75 – СК7: Промена робе- Основни сценарио - приказ податка

4. Магационер **уноси (мења)** робу. (АПУСО)

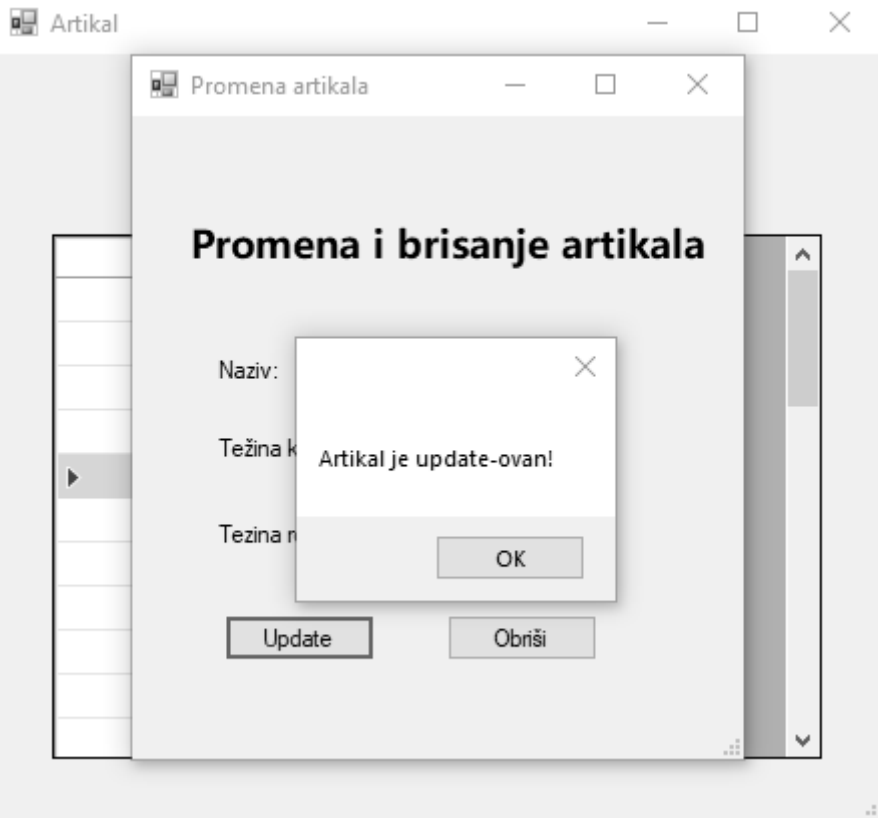
Опис акције: Магационер може да промени назив робе или тежину јединице у којој долази.



Слика 76 – СК7: Промена robe - Основни сценарио - унос нове вредности

5. Магационер **контролише** да ли је коректно унео робу. (АНСО)
6. Магационер **позива** систем да запамти промену робе. (АПСО)
   Опис акције: Магационер притиска дугме "Update" и на тај начин позива систем да ажурира одабрану робу.

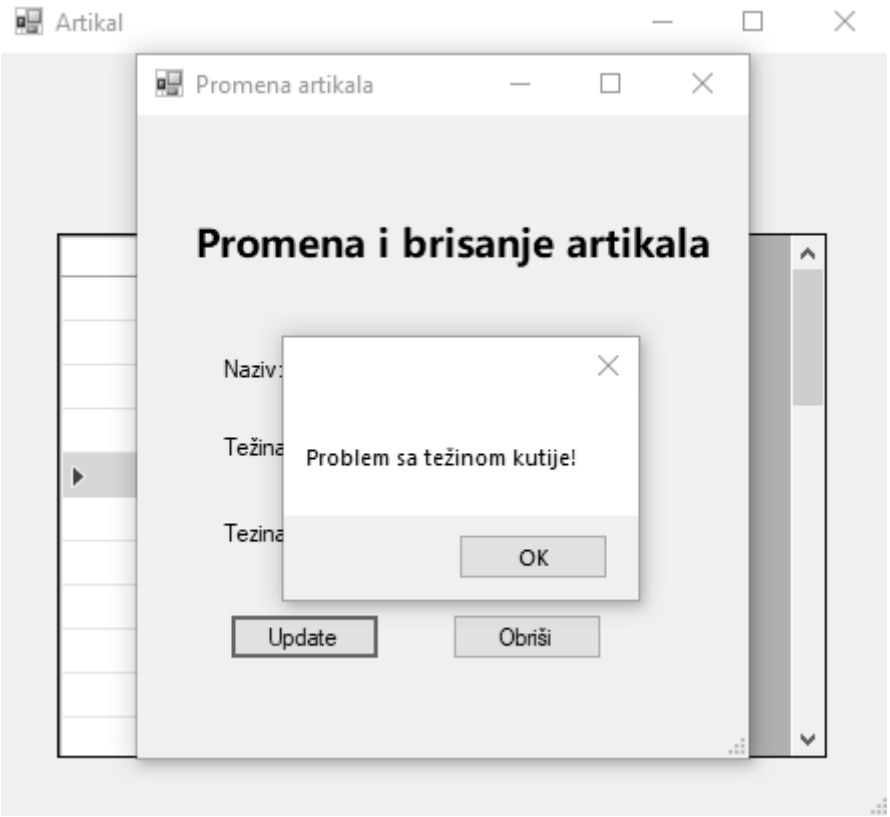
7. Систем **памти** робу. (СО)
8. Систем **приказује** Магационеру запамћену промену и поруку: “Артикал је ажуриран!.” (ИА)



Слика 77 - СК6: Промена података - Основни сценарио - успешно ажурирано

Алтернативни сценарио:

8.1 Уколико систем не може да запамти робу он приказује Магационеру поруку. Прекида се извршење сценариа. (ИА)



Слика 78 - СК6: Промена робе - Алтернативни сценарио - лоше унесени подаци

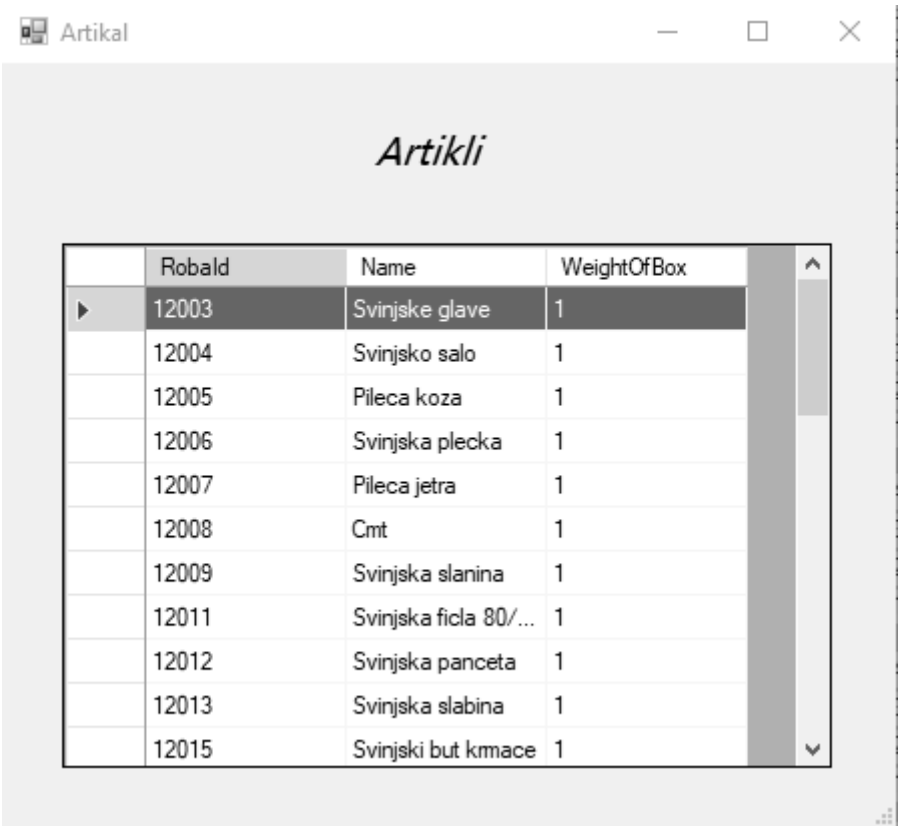
12.3.8 СК8: Случај коришћења – Брисање робе

Назив СК
Брисање робе

Актори СК
Магационер

Учесници СК
Магационер и систем (програм)

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



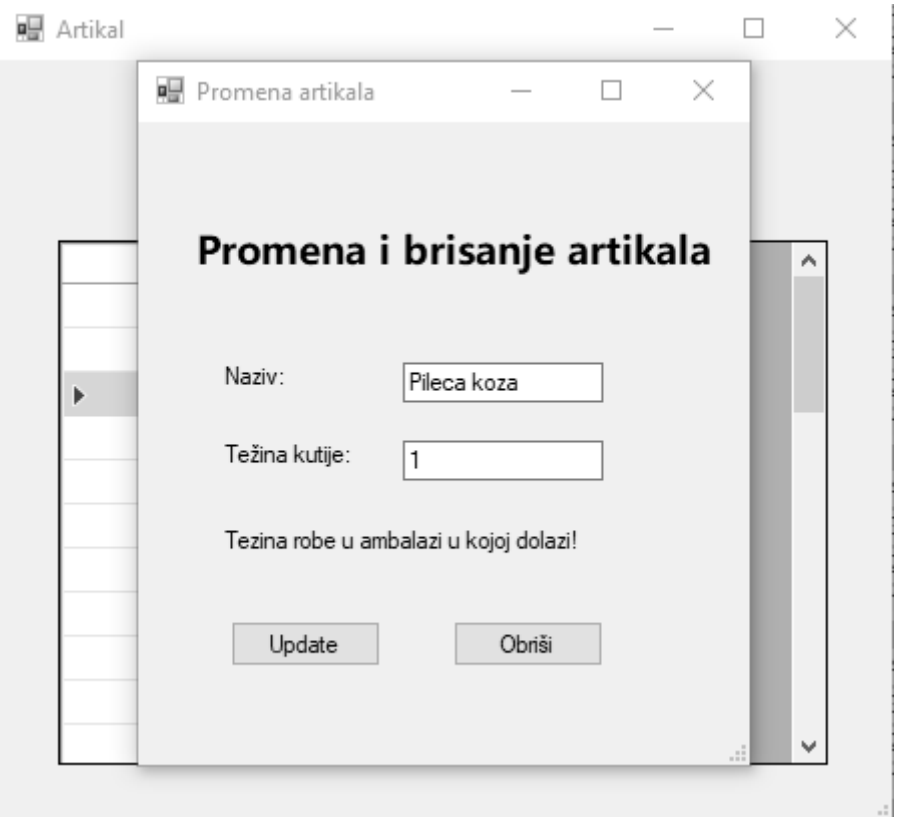
Слика 79 - СК8: Брисање робе - почетна форма

Основни сценарио СК:

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

Опис акције: Магационер бира робу коју намерава да обрише.

3. Систем **тражи** робу по задатој вредности. (СО)
4. Систем приказује Магационеру робу. (ИА)



Слика 80 - СК8: Брисање робе - Основни сценарио - одабир података за брисање

5. Магационер **позива** систем да обрише робу. (АПСО)

Опис акције: Магационер притиска дугме "Обриши" која позива системску операцију да обрише одабрану робу из система.

6. Систем **брише** робу. (СО)
7. Систем **приказује** Магационеру поруку: “Обрисано.” (ИА)

Слика 81 - СК8: Брисање робе - Основни сценарио - брисање

12.4 Пројектовање контролера корисничког интерфејса

Контролер корисничког интерфејса је одговоран за:

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

12.5 Пројектовање апликационе логике

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

- Део за комуникацију са клијентом, обрада клијента
- Контролер апликационе логике
- Део за комуникацију са складиштем података (Брокер базе података)
- Део који садржи пословну логику

12.5.1 Комуникација са клијентима

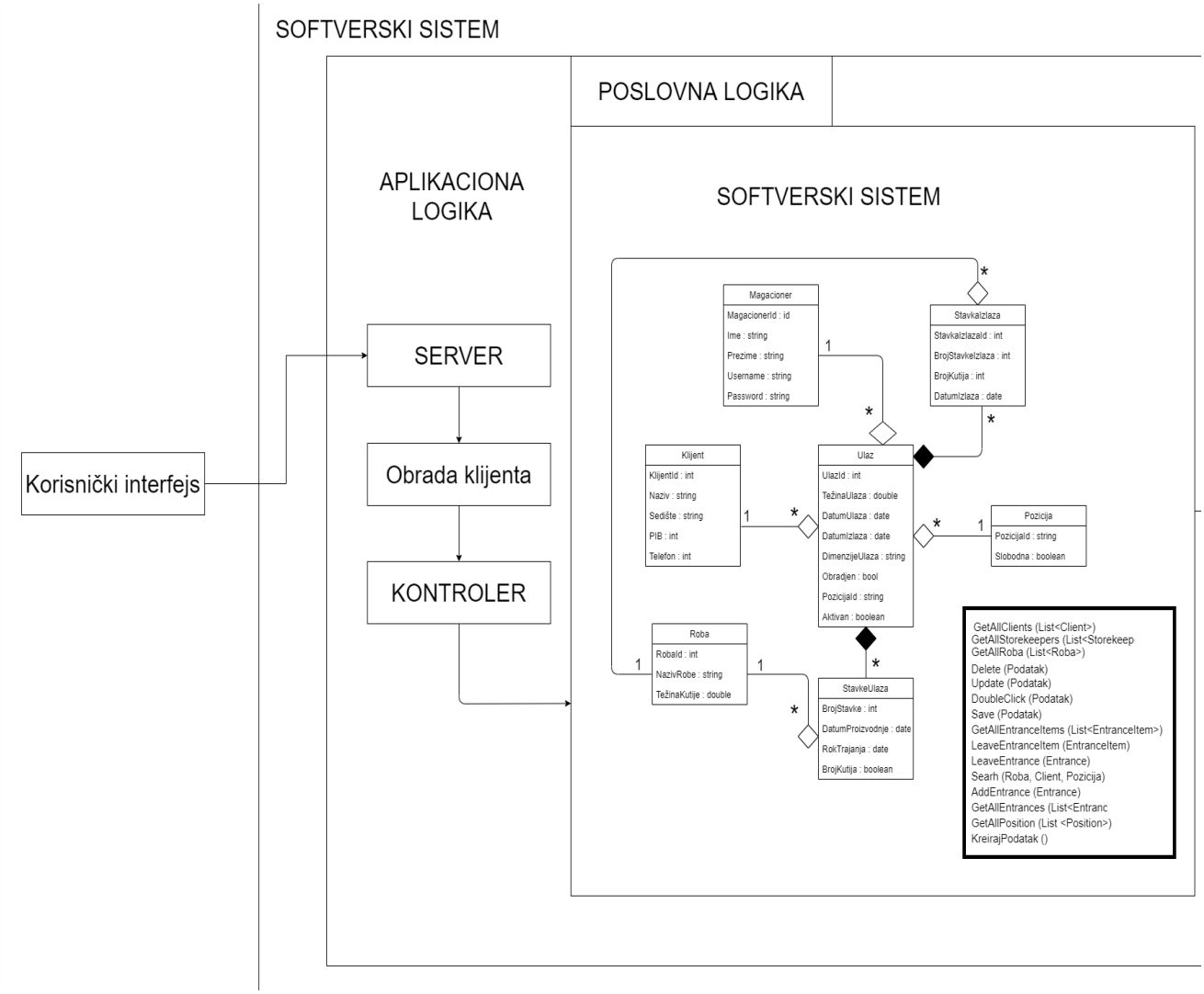
Клијент сервер комуникација се одвија преко порта и адресе рачунара. Наиме, када клијентски сокет успостави везу са серверским сокетом који све време прислушкује, отвара се посебна нит ка кориснику и она ће бити задужена за ту комуникацију. Клијент шаље захтев за извршење системских операција преко своје нити. Све те захтеве

прихвата контролер апликационе логике. Након извршења системске операције, резултат се враћа до нити клијента који се манифестује преко екранске форме.

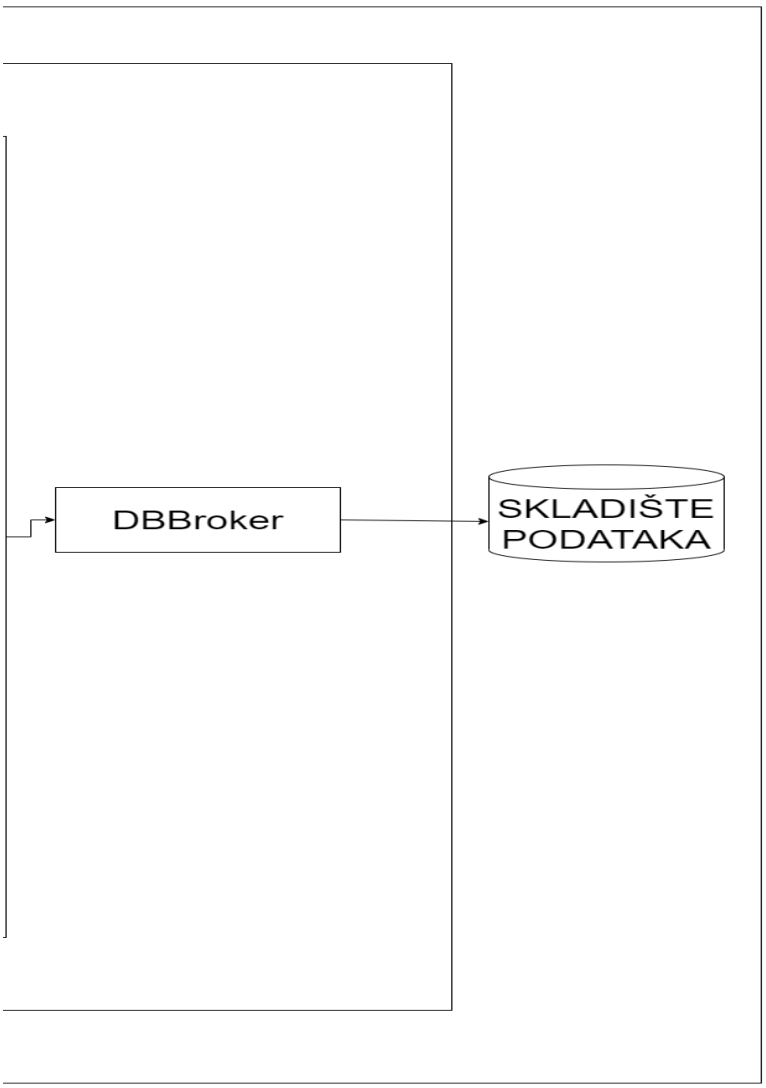
12.5.2 Контролер апликационе логике

У оквиру апликационе логике пројектује се контролер апликационе логике, пословна логика и брокер базе података. Контролер апликационе логике је одговоран да прихвати захтев за извршење системске операције од клијента и да га проследи до пословне логике која је одговорна за извршење системских операција.

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



Слика 82 - Архитектура софтверског система након пројектовања контролера апликационе логике – 1 део



Слика 83 - Архитектура софтверског система након пројектовања контролера апликационе логике – 2 део

12.5.3 Пословна логика

Пословна логика је описана са структуром (доменским класама) и понашањем (системским операцијама). На основу фазе за прикупљање захтева и анализе добили смо спецификацију структуре и понашања система, односно спецификацију пословне логике софтверског система. За сваку системску операцију потребно је направити концептуално решења која су директно повезана са логиком проблема. За сваки претходно дефинисан уговор прави се концептуално решење.

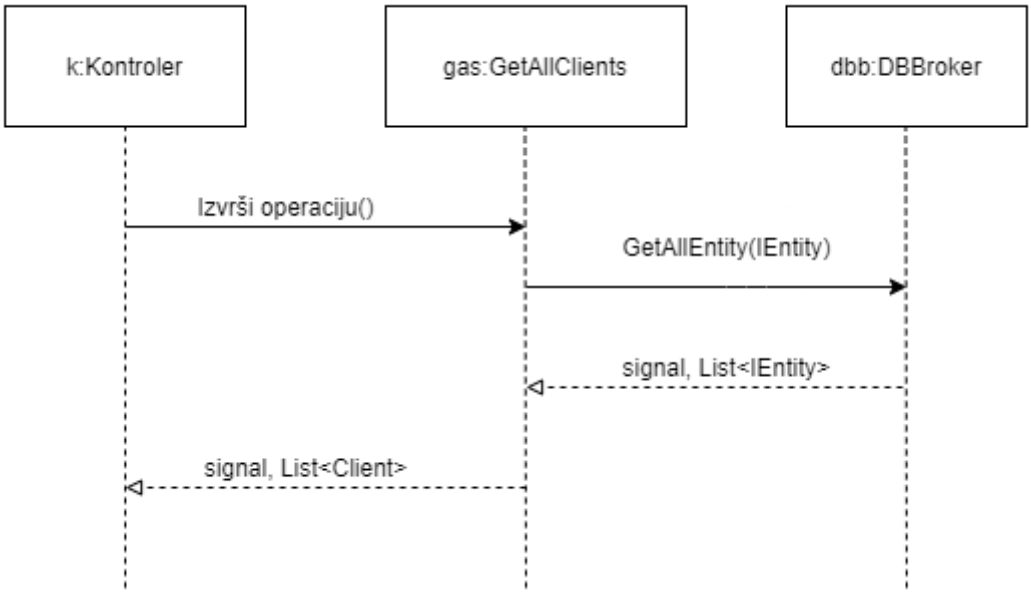
12.5.3.1 Уговор УГ1: ВратиСвеКлијенте

Операција: GetAllClients(List<Client>):signal

Веза са СК: СК1, СК3, СК4, СК5

Предуслови: /

Постуслови: /



Слика 84 - УГ1: ВратиСвеКлијенте

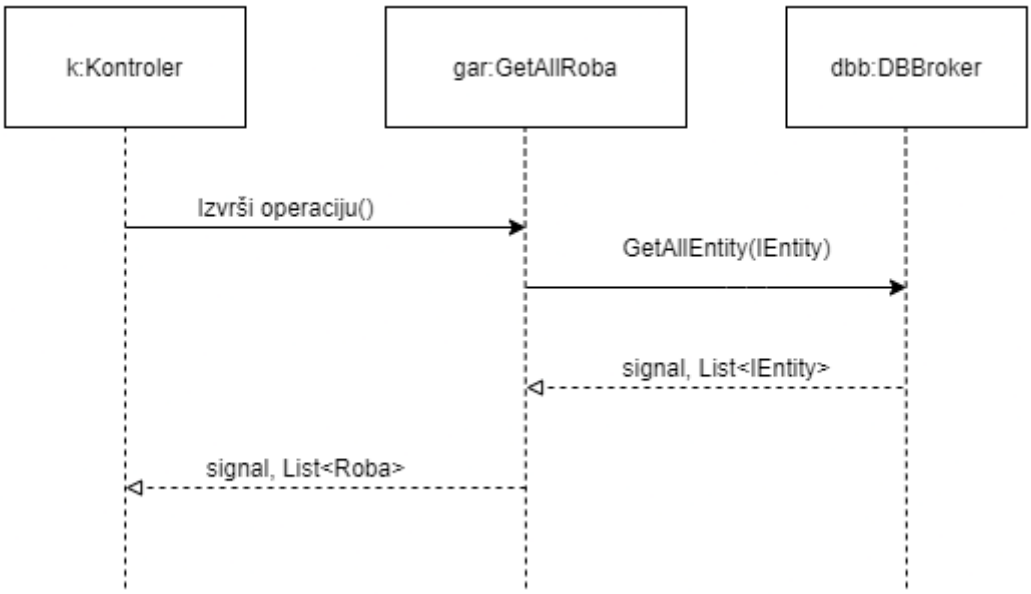
12.5.3.2 Уговор УГ2: ВратиСвеАртикле

Операција: GetAllRoba(List<Client>):signal

Веза са СК: СК3, СК5, СК6, СК8

Предуслови: /

Постуслови: /



Слика 85 – УГ2: ВратиСвеАртикле

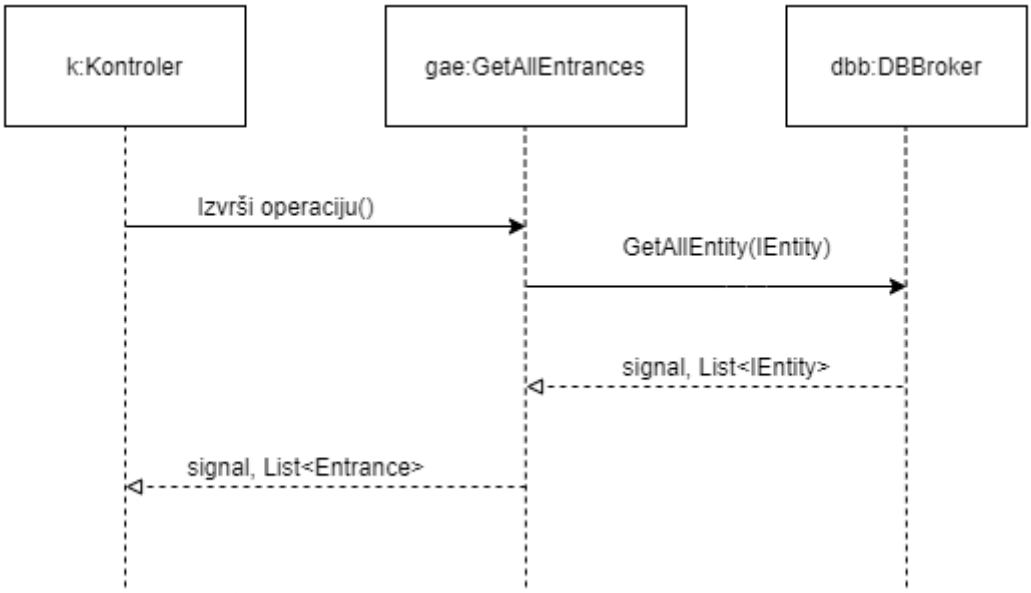
12.5.3.3 Уговор УГ3: ВратиСвеУлазе

Операција: GetAllEntrance(List<Entrance>):signal

Веза са СК: СК2

Предуслови: /

Постуслови: /



Слика 86 – УГ3: ВратиСвеУлазе

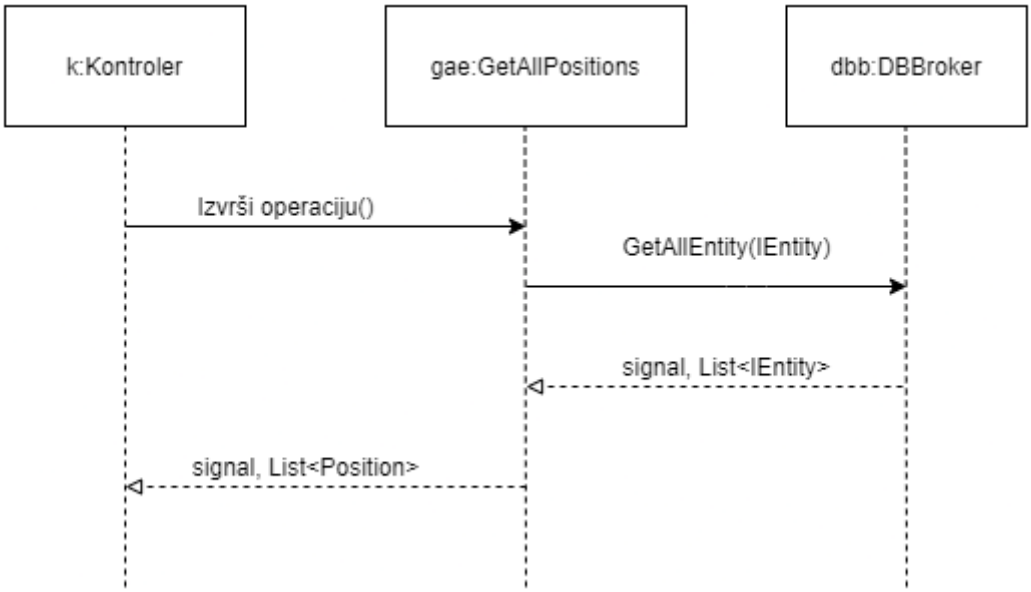
12.5.3.4 Уговор УГ4: ВратиСвеПозиције

Операција: GetAllPosition(List<Position>):signal

Веза са СК: СК2

Предуслови: /

Постуслови: /



Слика 87 – УГ4: ВратиСвеПозиције

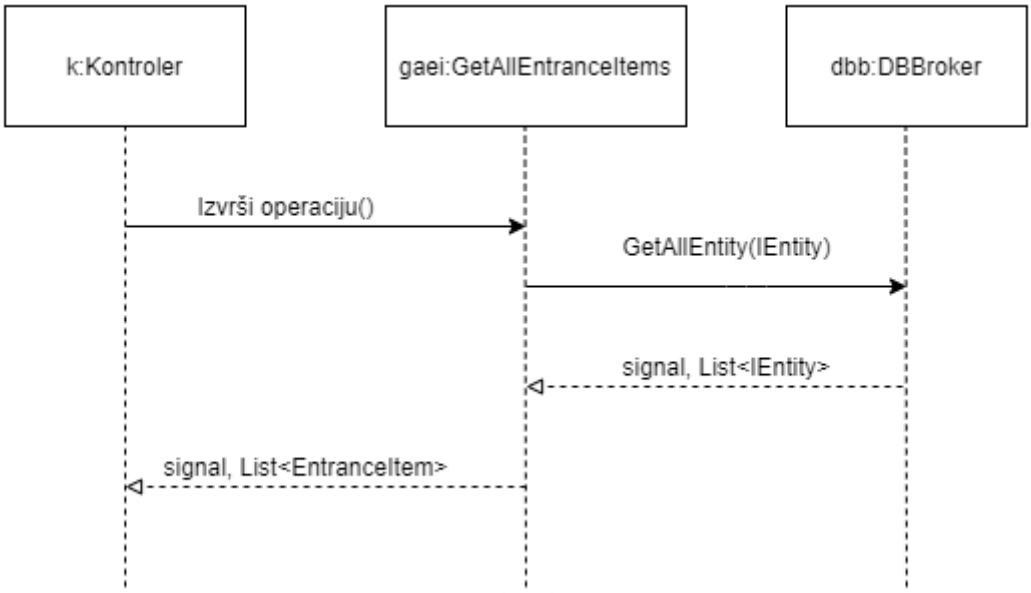
12.5.3.5 Уговор УГ5: ВратиСвеУлазнеСтавке

Операција: GetAllEntranceItems(List<Position>):signal

Веза са СК: СК3, СК5

Предуслови: /

Постуслови: /



Слика 88 – УГ5: ВратиСвеУлазнеСтавке

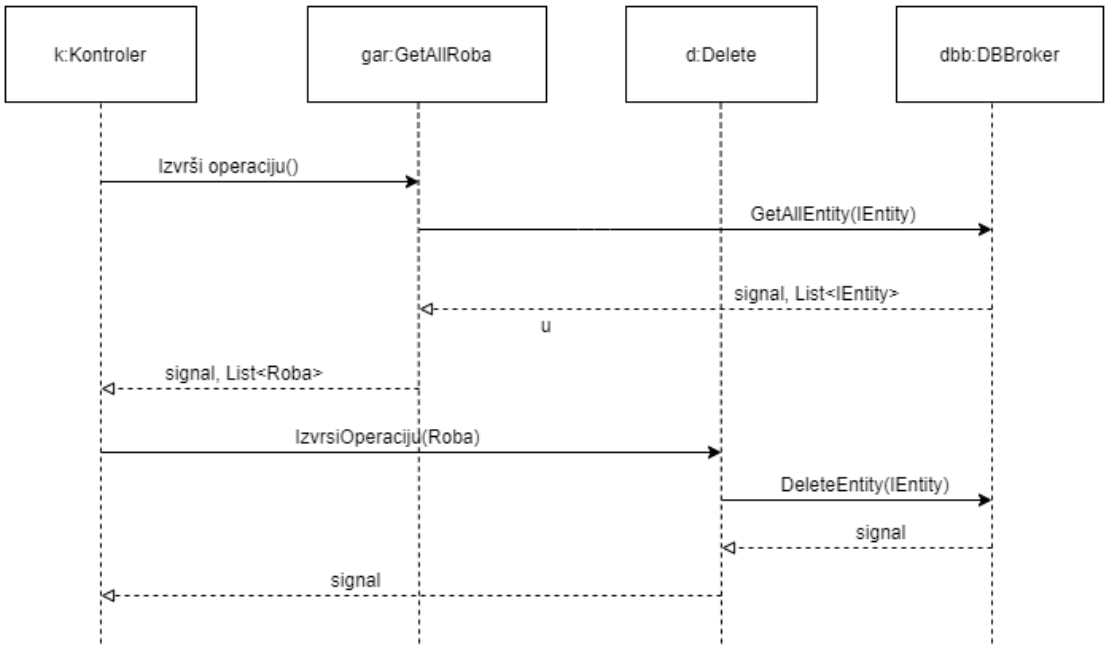
12.5.3.6 Уговор УГ6: ОбришиРобу

Операција: Delete(Roba):signal

Веза са СК: СК8

Предуслови: Вредносна и структурна ограничења над објектом Податак морају бити задовољена. Учитана је листа података Робе.

Постуслови: Роба је обрисана.



Слика 89 – УГ6: Обриши робу

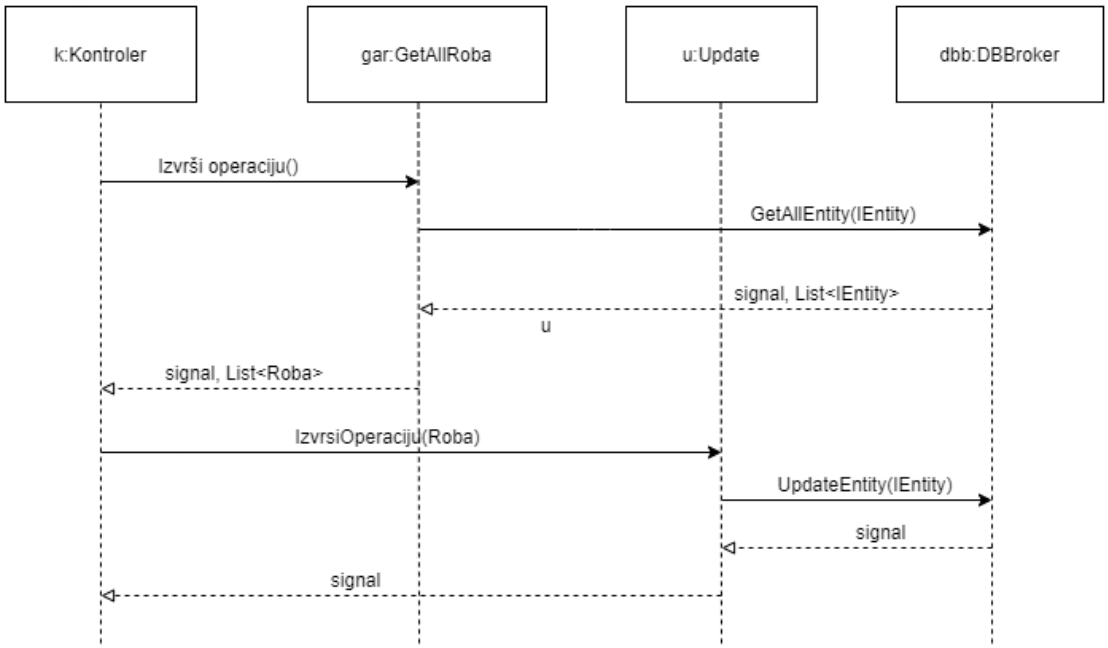
12.5.3.7 Уговор УГ7: АжурирајРобу

Операција: Update(Roba):signal

Веза са СК: СК7

Предуслови: Вредносна и структурна ограничења над објектом Роба морају бити задовољена. Учитана је листа Робе.

Постуслови: Роба је ажурирана.



Слика 90 – УГ7: АжурирајРобу

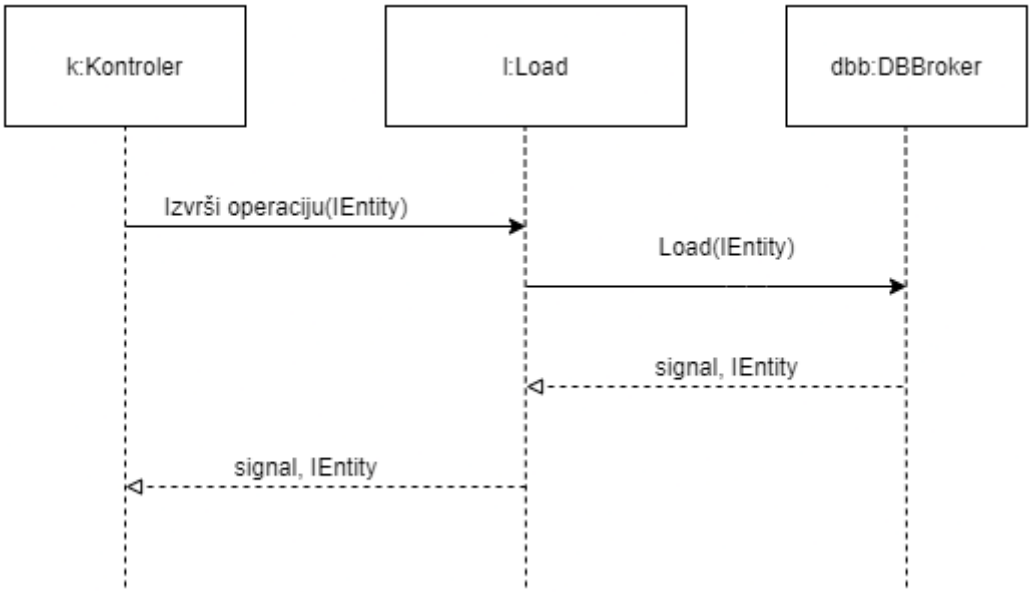
12.5.3.8 Уговор УГ8: Учитај

Операција: Load(Roba):signal

Веза са СК: СК3, СК5, СК7, СК8

Предуслови: Вредносна и структурна ограничења над објектом Податак морају бити задовољена. Учитана је листа података (Клијент, Роба, Магационер).

Постуслови: Податак је ажуриран.



Слика 91 – УГ8: Учитај

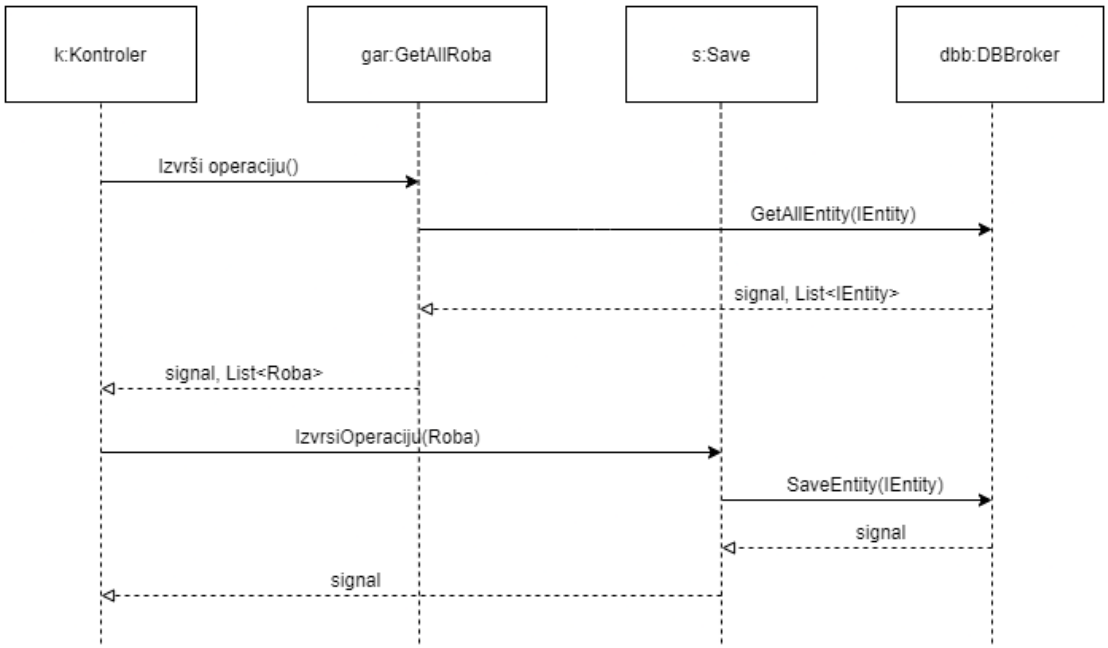
12.5.3.9 Уговор УГ9: СачувајРобу

Операција: Save(Roba):signal

Веза са СК: СК6

Предуслови: Вредносна и структурна ограничења над објектом Роба морају бити задовољена. Учитана је листа Робе.

Постуслови: Роба је унешена.



Слика 92 – УГ9: СачувајПодатак

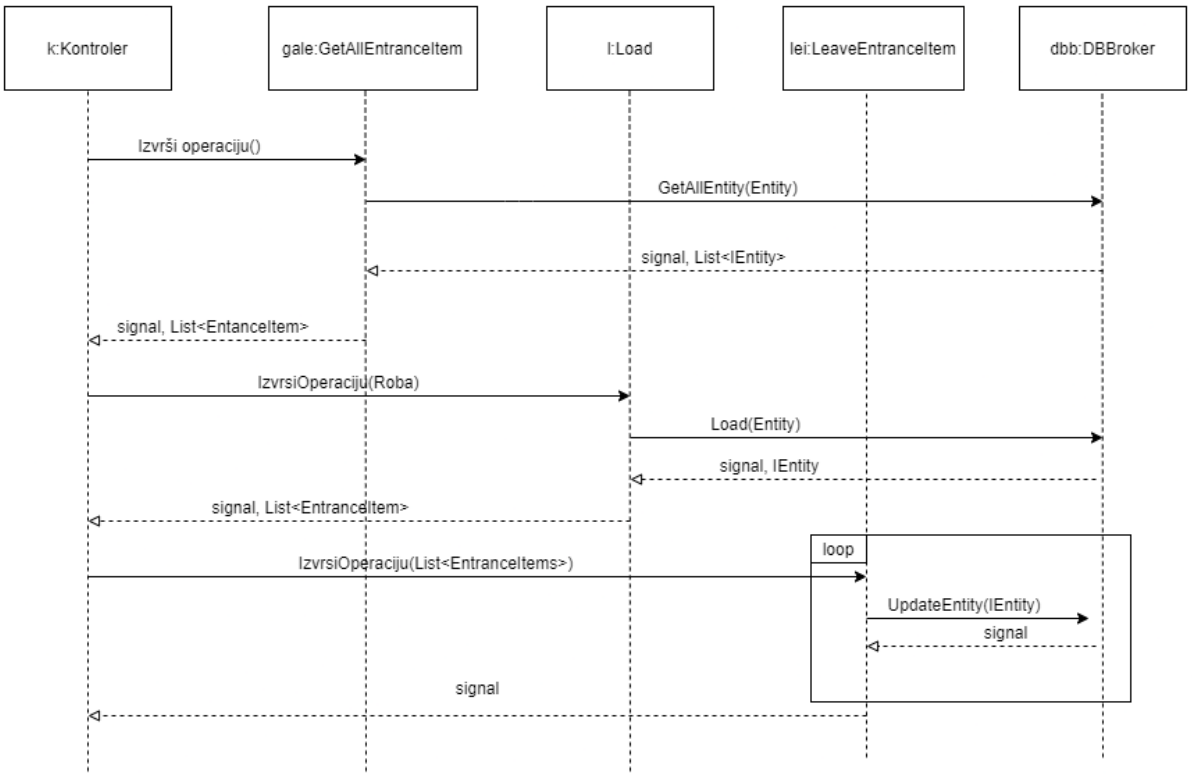
12.5.3.10 Уговор УГ10: ИзлазСтавкеПалете

Операција: LeaveEntranceItem(EntranceItem):signal

Веза са СК: СК5

Предуслови: Вредносна и структурна ограничења над објектом УлазнаСтавка морају бити задовољена. Учитана је листа улаза као и листа података (Клијент, Роба, Магационер).

Постуслови: Ставка је изашла.



Слика 93 – УГ10: ИзлазСтавкеПалете

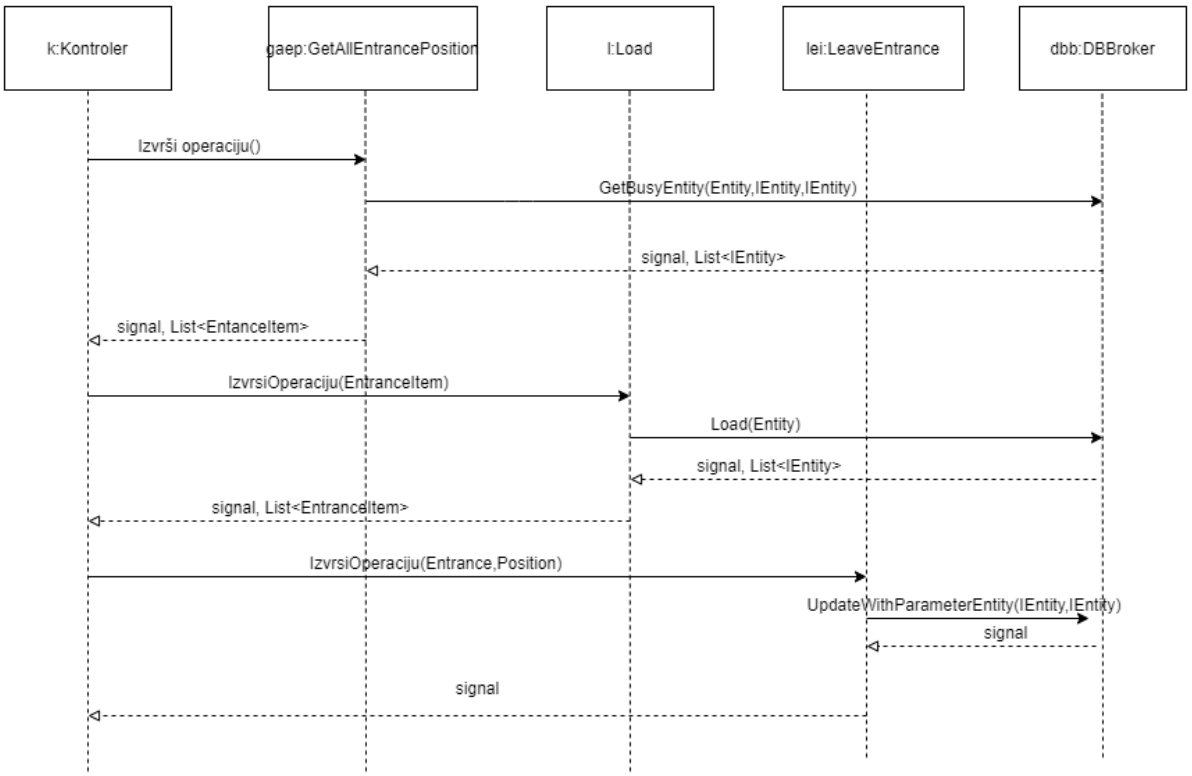
12.5.3.11 Уговор УГ11: ИзбациРобуСаПозиције

Операција: LeaveEntrancePosition(EntrancePosition):signal

Веза са СК: СК4

Предуслови: Вредносна и структурна ограничења над објектом Улаз и Позиција морају бити задовољена. Учитана је листа свих палета у складишту.

Постуслови: Палета је изашла.



Слика 94 - УГ11: ИзбациРобуСаПозиције

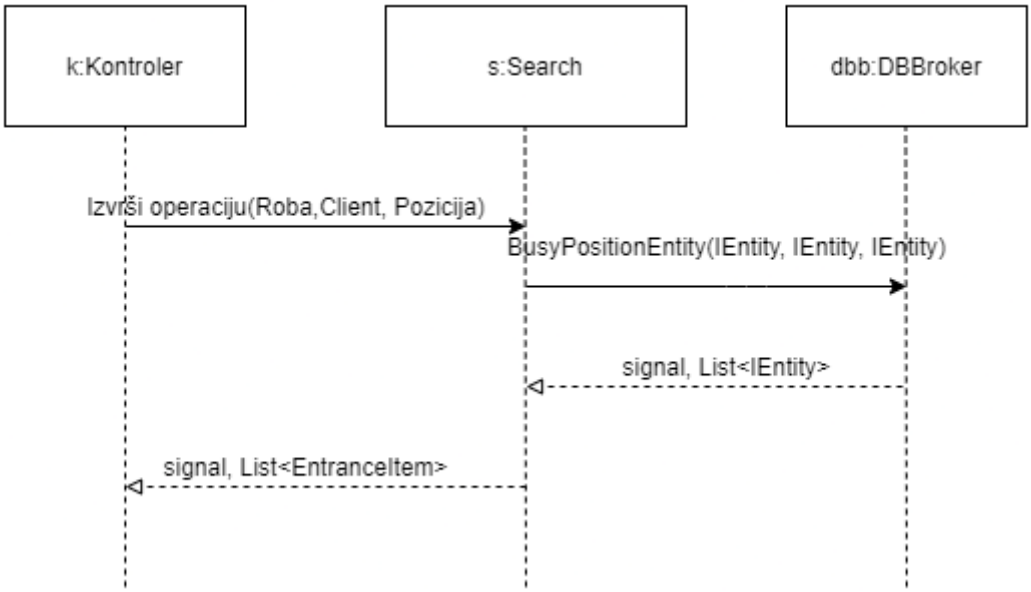
12.5.3.12 Уговор УГ12: Претрага

Операција: Search(Roba,Client,Position):signal

Веза са СК: СК3, СК4

Предуслови: Вредносна и структурна ограничења над објектом Роба, Клијент, Позиција морају бити задовољена.

Постуслови: Палета је пронађена.



Слика 95 - УГ12: Претрага

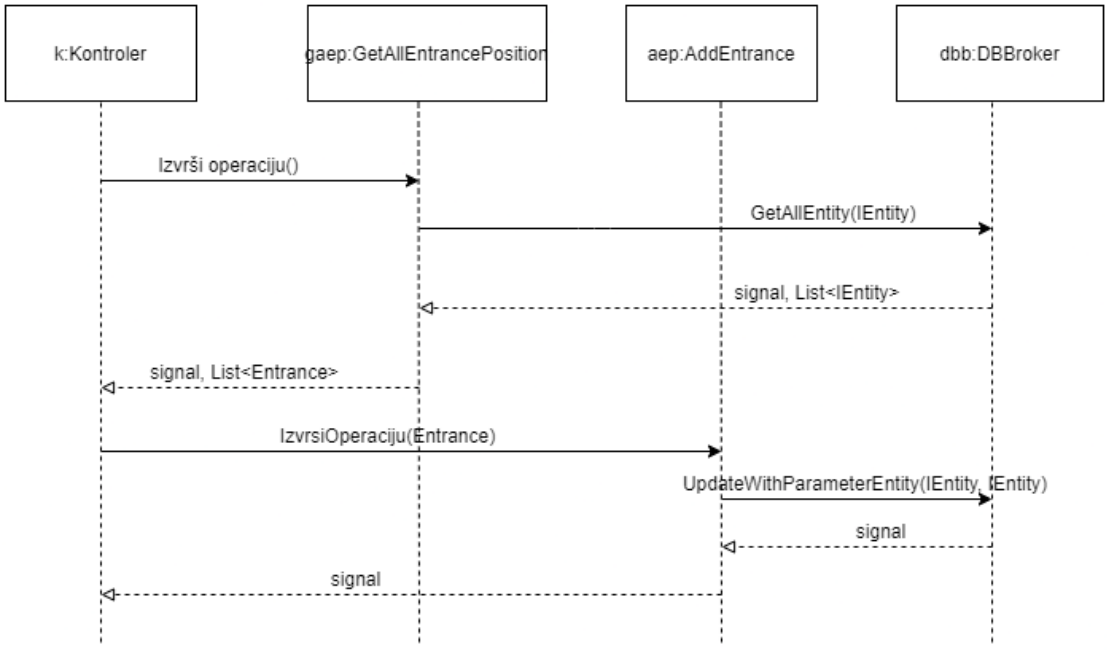
12.5.3.13 Уговор УГ13: УнесиПалету

Операција: AddEntrancePosition(Entrance,Position):signal

Веза са СК: СК2

Предуслови: Вредносна и структурна ограничења над објектом Улаз и Позиција морају бити задовољена. Учитана је листа улазних позиција.

Постуслови: Улазна позиција са палетом је унешена.



Слика 96 - УГ13: УнесиПалету

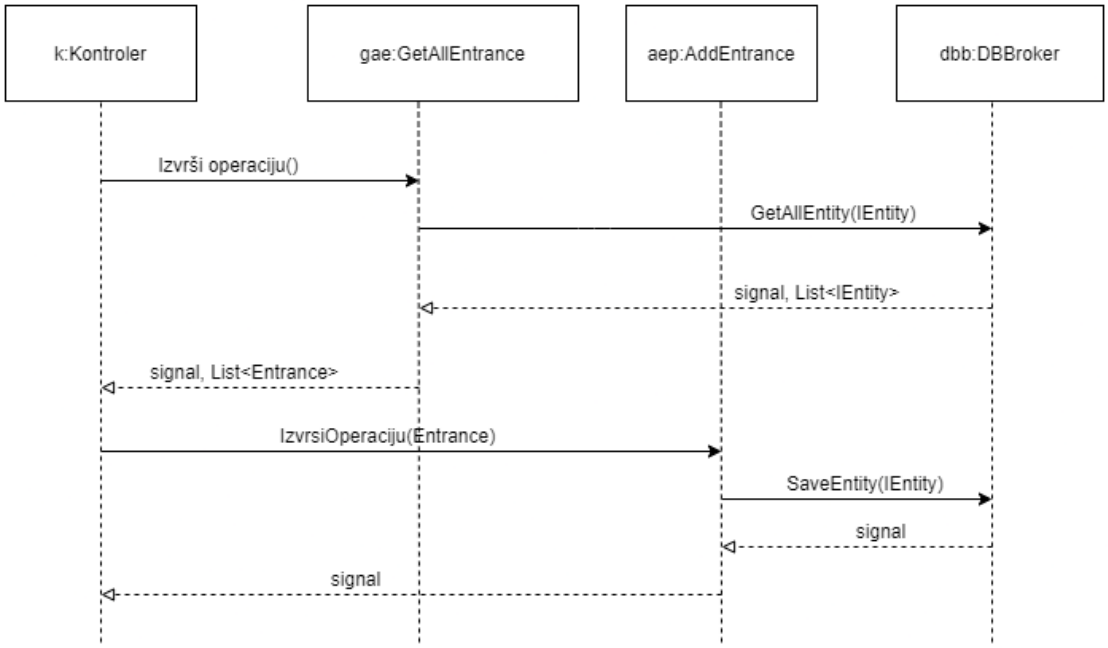
12.5.3.14 Уговор УГ14: ДодајПалету

Операција: AddEntrance (Entrnace):signal

Веза са СК: СК1

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

Постуслови: Улазна палета је унешена.



Слика 97 - УГ14: ДодајПалету

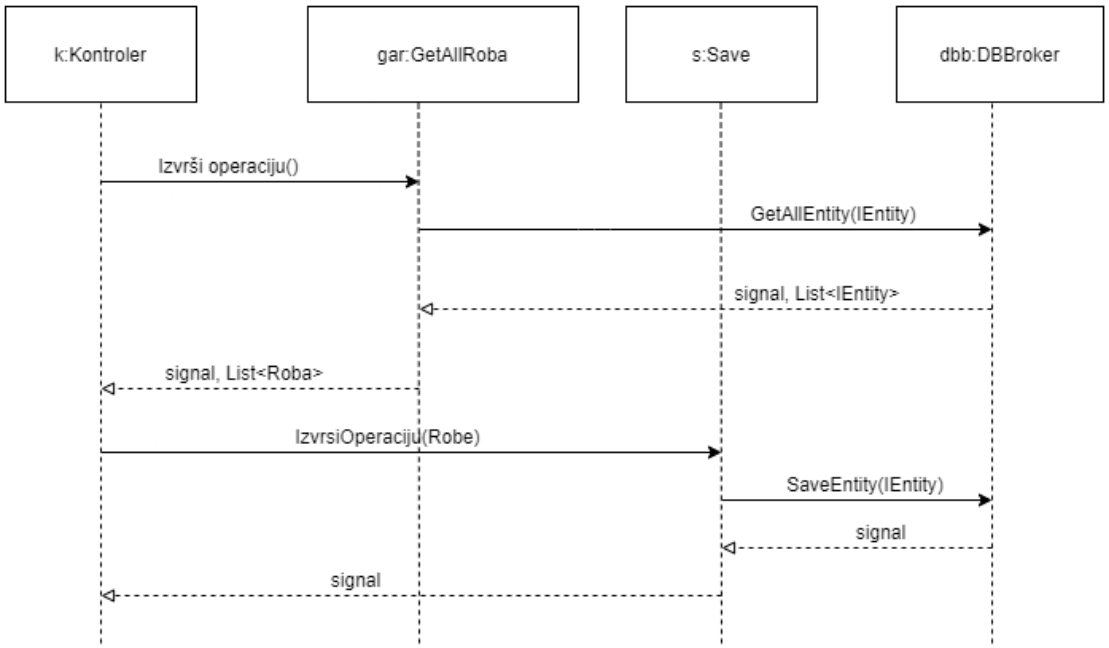
12.5.3.15 Уговор УГ15: КреирајРобу

Операција: Kreiraj(Roba):signal

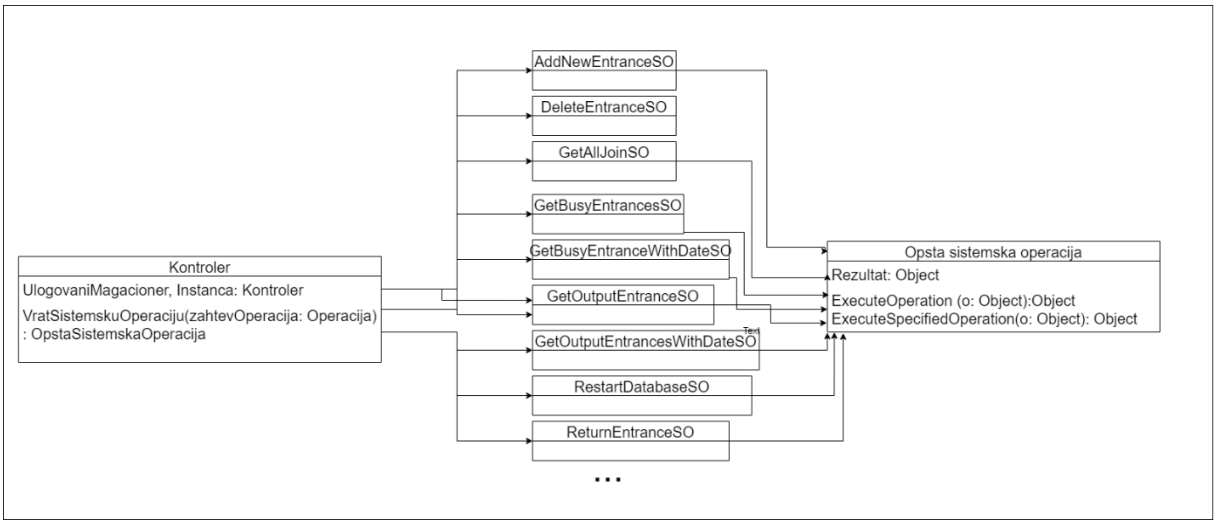
Веза са СК: СК7

Предуслови: Вредносна и структурна ограничења над објектом Роба морају бити задовољена. Учитана је листа Робе.

Постуслови: Роба је креирана.



Слика 98 - УГ15: КреирајРобу



Слика 99- Дијаграм класа који повезује везу између контролера аплик. логике и класа за извршење сист. операција

12.5.4 Брокер базе података

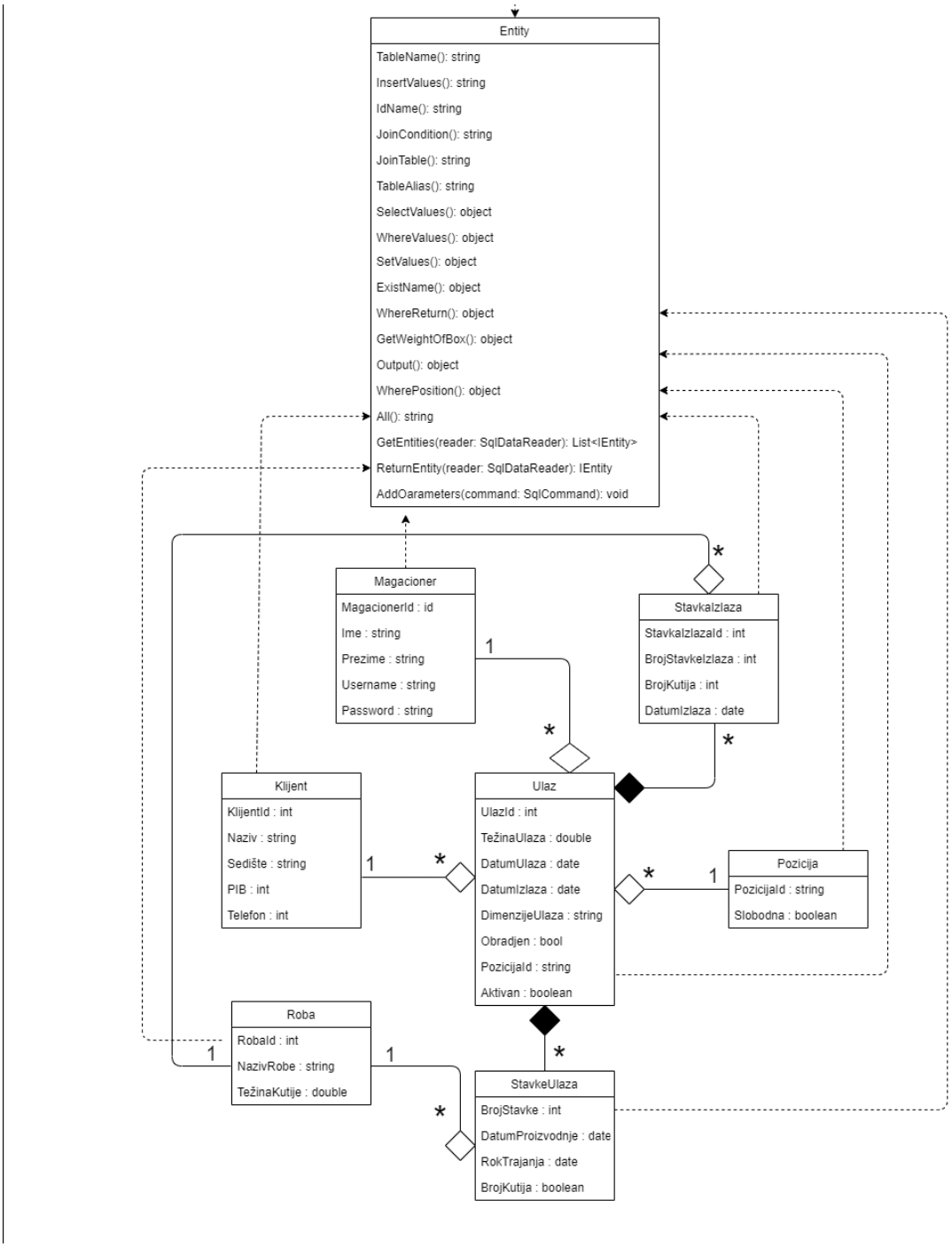
Класа Брокер представља перзистентни оквир који посредује у свим операцијама над базом података и реализује следеће методе:

- public List<IEntity> GetAllEntity(IEntity entity)
- public void SaveEntity(IEntity entity)
- public bool GetBoolEntity(IEntity entity, object pozicija)
- public DataTable BusyEnity(IEntity entity, IEntity entity, IEntity entity)

- public DataTable OutputEntity(IEntity entity, IEntity entity, IEntity entity, IEntity entity)
- public DataTable OutputEntityWithDate(IEntity entity, IEntity entity, IEntity entity, object datumOd, object datumDo)
- public DataTable BusyEntityWithDate(Entrance entity, IEntity entity, IEntity entity, object datumOd, object datumDo)
- public void UpdateEntity(IEntity entity)
- public void UpdateWithParameterEntity(IEntity entity, object value)
- public void UpdateWithParameterEntity(IEntity entity, IEntity entity)
- public void DeleteEntity(IEntity entity)
- public bool ExistEntity(IEntity entity)
- public IEntity ReturnEntity(IEntity entity ,object uslov)
- public IEntity ReturnEntity(IEntity entity)
- public object GetEntity(IEntity entity, object uslov)
- public object SaveEntity(IEntity entity)
- public DataTable GetAllEntity(IEntity entity)
- public DataTable GetAllEntityWithParameter(IEntity entity, object value)
- public List<IEntity> SearchEntity(IEntity entity, object uslov)
- public DataTable BusyPositionEntity(IEntity entity,IEntity entity, IEntity entity)
- public DataTable BusyPositionEntityWithPosition(IEntity entity, IEntity entity, IEntity entity, object values)
- public void OpenConnection()
- public void CloseConnection()
- public void Commit()
- public void Rollback()
- public void BeginTransaction()

Брокер базе података је задужен да обезбеди комуникацију пословне логике софтверског система са складиштем података. Ова софтверска класа садржи генеричке методе које се прво повезују са самом базом, а потом и извршавају упите. Сваки од упита врши промену или жељени приказ, у складу са самим упитом. Методе су генеричке, што значи да могу да прихвате објекте различитих класа и да над њима креирају упите. То све омогућава интерфејс *IEntity*, односно општа доменска класа *Entity*. Свака класа имплементира дати интерфејс и његове методе.

Слика 100 - Брокер класа се повезује са општим објектом Entity 1 део



Слика 101 - Брокер класа се повезује са општим објектом Entity 2 део

12.6 Пројектовање складишта података

Након формирања концептуалног модела и веза између самих доменских класа потребно је да креирамо релациони модел. Релациони модел је битан за пројектовање базе података. Приказане су све табеле као и њихови примарни и спољни кључеви:

Релациони модел:

- **Klijent** (KlijentId, NazivKlijenta, Sedište, PIB, Telefon)
- **Roba** (RobaId, NazivRobe, TežinaKutije)

- **Magacioner** (MagacionerId, Ime, Prezime, Username, Password)
- **Ulaz** (UlazId, TežinaUlaza, DatumUlaza, DatumIzlaza, DimenzijaUlaza, Aktivan Obrađen, *KlijentId, MagacionerId, PozicijaId*)
- **StavkaUlaza** (BrojStavkeUlaza, UlazId, DatumProizvodnje, RokTrajanja, BrojKutije, *RobaId*)
- **Pozicija** (PozicijaId, Slobodna)
- **StavkaIzlaza** (StavkaIzlazaId, BrojStavke, UlazId, BrojKutije, *RobaId*, DatumIzlaza)

Tabela Klijent		Prosto vrednosno ogranicenje		Slozeno vrednosno ogranicenje		Strukturno ogranicenje
Atributi		Tip atributa	Vrednost atributa	Medjuzav. atributa jedne tabele	Medjuzav. atributa vise tabela	INSERT / UPDATE CASCADE Ulaz DELETE RESTRICTED Ulaz
	KlijentId	Int	Not null			
	NazivKlijenta	String				
	Sedište	String				
	Mesto	String	Not null			

Табела 1 - Табела Клијент

Tabela Roba		Prosto vrednosno ogranicenje		Slozeno vrednosno ogranicenje		Strukturno ogranicenje
		Tip atributa	Vrednost atributa	Medjuzav. atributa jedne tabele	Medjuzav. atributa vise tabela	INSERT / UPDATE CASCADES StavkaUlaza, StavkaIzlaza DELETE RESTRICTED StavkaUlaza, StavkaIzlaza
Atributi	RobaId	Int	Not null			
	NazivRobe	String	Not null			
	TežinaKutije	Double	Not null			

Табела 2 - Tabela Roba

Tabela Magacioner		Prosto vrednosno ogranicenje		Slozeno vrednosno ogranicenje		Strukturno ogranicenje
Atributi		Tip atributa	Vrednost atributa	Medjuzav. atributa jedne tabele	Medjuzav. atributa vise tabela	INSERT / UPDATE CASCADE Ulaz
	MagacionerId	Int	Not null			

	Ime	String	Not null			DELETE RESTRICTED Ulaz
	Prezime	String	Not null			
	Username	String	Not null			
	Password	Date	Not null			

Табела 3 - Табела Магационер

Tabela Ulaz		Prosto vrednosno ogranicenje		Slozeno vrednosno ogranicenje		Strukturno ogranicenje
		Tip atributa	Vrednost atributa	Medjuzav. atributa jedne tabele	Medjuzav. atributa vise tabela	INSERT RESTRICTED Klijent, Magacioner UPDATE RESTRICTED StavkaUlaza, StavkaIzlaza DELETE RESTRICTED StavkaUlaza, StavkaIzlaza
Atributi	UlazId	Int	Not null			
	TežinaUlaza	Double	Not null		UkupnaTežinaUlaza= SUM(TežinaKutije*BrojKutija)	
	DatumUlaza	Date	Not null			
	DatumIzlaza	Date				
	DimenzijaUlaza	String				
	KlijentId	Int	Not null			
	Obradjen	Boolean	Not null			
	MagacionerId	Int	Not null			
	PozicijaId	String	Not null			
	Aktivan	Boolean	Not null			

Табела 4 - Табела Улаз

Tabela Stavka Ulaza		Prosto vrednosno ogranicenje		Slozeno vrednosno ogranicenje		Strukturno ogranicenje
Atributi		Tip atributa	Vrednost atributa	Medjuzav. atributa jedne tabele	Medjuzav. atributa vise tabela	INSERT RESTRICTED Ulaz, Roba UPDATE RESTRICTED Ulaz, Roba
	BrojStavkeUlaza	Int	Not null			
	UlazId	Int	Not null			

	RokTrajanja	Date	Not null			DELETE /
	BrojKutije	Int	Not null			
	RobaId	Int	Not null			

Табела 5 - Табела Ставка Улаза

Tabela Pozicija		Prosto vrednosno ogranicenje		Slozeno vrednosno ogranicenje		Strukturno ogranicenje
		Tip atributa	Vrednost atributa	Medjuzav. atributa jedne tabele	Medjuzav. atributa vise tabela	INSERT / UPDATE CASCADES Ulaz DELETE RESTRICTED Ulaz
Atributi	PoziijaId	String	Not null			
	Slobodna	Boolean	Not null			

Табела 6 - Табела Позиција

Tabela Stavka Izlaza		Prosto vrednosno ogranicenje		Slozeno vrednosno ogranicenje		Strukturno ogranicenje
Atributi		Tip atributa	Vrednost atributa	Medjuzav. atributa jedne tabele	Medjuzav. atributa vise tabela	INSERT RESTRICTED Ulaz, Roba UPDATE RESTRICTED Ulaz, Roba DELETE /
	BrojStavkeIzlaza	Int	Not null			
	StavkaIzlazaId	Int	Not null			
	UlazId	Int	Not null			
	DatumIzlaza	Date	Not null			
	BrojKutija	Int	Not null			
	RobaId	Int	Not null			

Табела 7 - Табела Ставка Излаза

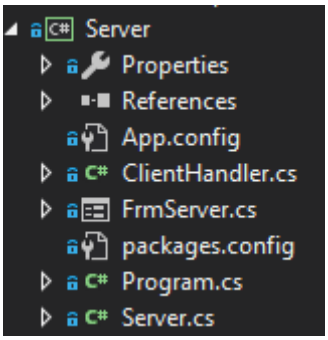
13. Фаза имплементације

Софтверски систем је развијен у програмском језику C#. Систем је пројектован по клијент-сервер архитектури. Као систем за управљање базом података коришћен је „Microsoft SQL Server“, док је развојно окружење „Microsoft Visual Studio 2019“.

13.1 Структура софтверског система

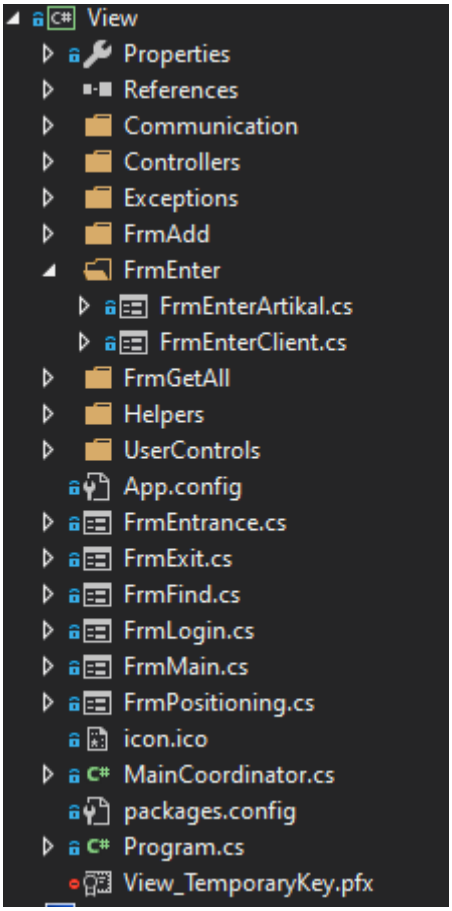
На основу архитектуре софтверског система добијени су следећи софтверски пројекти и њихове класе које ћемо поделити на делове:

- Сервер:



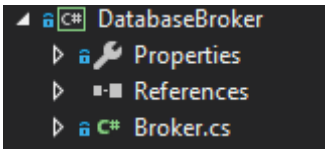
Слика 102- Сервер пројекат

- Клијент:



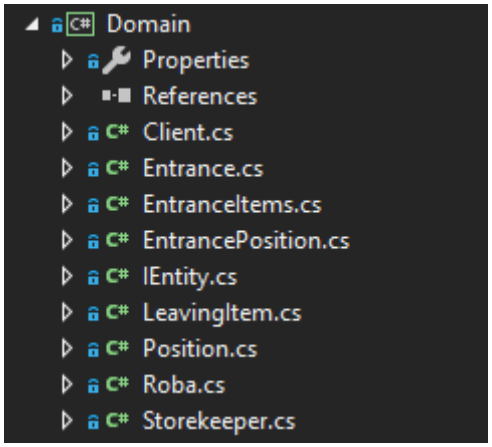
Слика 103 - Клијент пројекат

- Брокер базе података:



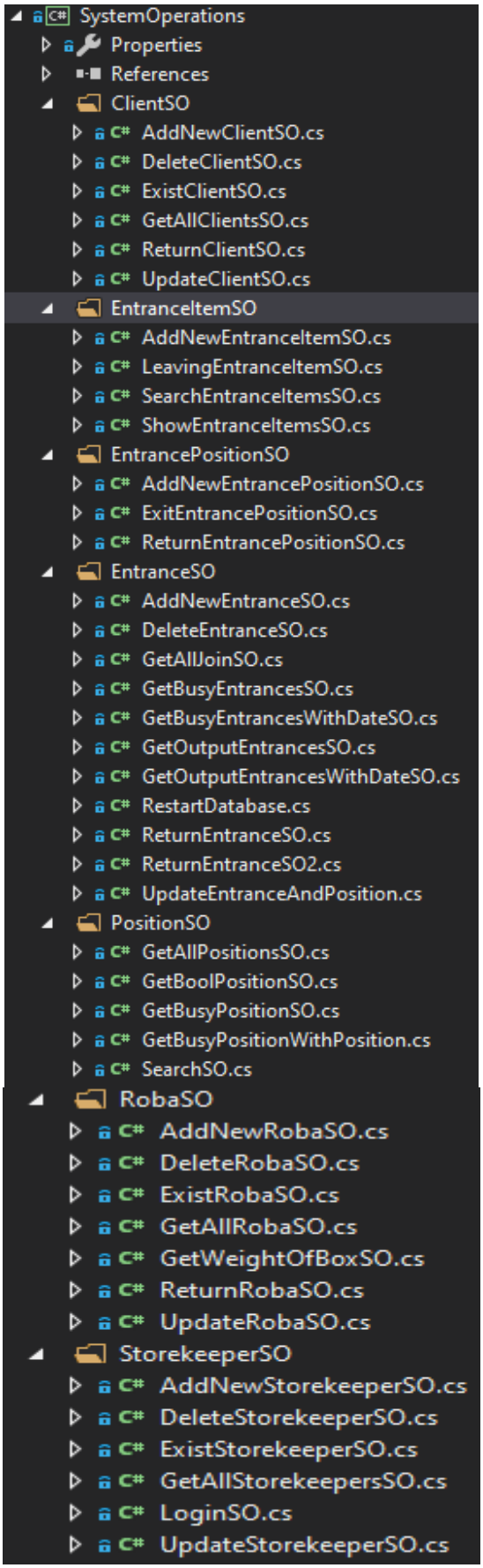
Слика 104 - Брокер пројекат

- Домени:



Слика 105 - Домен пројекат

- Системске операције:



Слика 106 - Системске операције пројекат

13.2 Имплементација имплементационе логике

Конкретна имплементациона логика ће бити у овом поглављу. Сав програмски код који буде наведен представља решење софтверског система.

13.2.1 Комуникација са клијентом

Да би било који клијент приступио софтверском систему потребно је да сервер буде покренут, односно да ослушкују нове клијенте. Метода Start() класе Server је имплементирана на следећи начин.

```
public void Start() {
        listener.Bind(new
IPEndPoint(IPAddress.Parse(ConfigurationManager.AppSettings["IP"]),int.Parse(ConfigurationManager.AppSettings[
"Port"])));
    }
```

Приликом покретања сервера креира се сокет који ослушкује мрежу на одређеној IP адреси и порту. Након креирања сокета покреће се главна нит која ослушкује мрежу. Метода Listen() класе Server је имплементирана на следећи начин.

```
public void Listen() {
        listener.Listen(5);
        bool kraj = false;
        try
        {

            while (!kraj)
            {
                Socket socketClient = listener.Accept();
                ClientHandler handler = new ClientHandler(this,socketClient);
                onlineKlijenti.Add(handler);
                Thread thread = new Thread(handler.StartHandler);
                thread.Start();
            }

        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
            kraj = true;
        }
    }
```

Ослушкивање мреже се одвија тако што покретањем клијентске стране покреће се метода Connect() која успоставља конекцију са сервером.

```
internal bool Connect()
    {
        try
        {
            Communication.Communication.Instance.Connect();
            return true;
        }
        catch (Exception ex)
        {
            MessageBox.Show("Greska prilikom povezivanja sa serverom");
            return false;
        }
```

```
    }
```

Да би се успоставила конекција са сервером, потребно је да на серверској страни постоји сокет који ослушкује мрежу и чека прилику на повезивање. Ако је покренут, тада се формира посебна комуникациона нит само према том клијенту, а главна нит наставља да ослушкује за потенцијалне друге клијенте. Клијенту се приказује почетна форма уколико се добро пријави. Сва даља комуникација између клијента и сервера се одвија путем креираног комуникационог канала, односно одговарајућих сокета и IP адреса. Обрада клијента се врши путем методе ProcessRequest() на серверској страни.

```
private Response PorocessRequest(Request request)
    {
        Response response = new Response();
        response.IsSuccessful = true;
        switch (request.Operation) {
            case Operation.Login:
                Storekeeper = Controler.Instance.Login((Storekeeper)request.RequestObject);
                if (Storekeeper != null) Server.OnlineKorisnici.Add(Storekeeper);
                response.Result = Storekeeper;
                break;
            case Operation.SaveStorekeeper:
                Controler.Instance.AddStorekeeper((Storekeeper)request.RequestObject);
                break;
            case Operation.GetAllStorekeepers:
                response.Result = Controler.Instance.GetAllStorekeepers();
                break;
            case Operation.UpdateStorekeeper:
                Controler.Instance.UpdateStorekeeper((Storekeeper)request.RequestObject);
                break;
            case Operation.DeleteStorekeeper:
                Controler.Instance.DeleteStorekeeper((Storekeeper)request.RequestObject);
                break;
            case Operation.FindClient:
               response.Result = Controler.Instance.FindClient((Client)request.RequestObject);
                break;
            case Operation.SaveClient:
                Controler.Instance.AddClient((Client)request.RequestObject);
                break;
            case Operation.GetAllClients:
                response.Result = Controler.Instance.GetAllClient();
                break;
            case Operation.UpdateClient:
                Controler.Instance.UpdateClient((Client)request.RequestObject);
                break;
            case Operation.DeleteClient:
                Controler.Instance.DeleteClient((Client)request.RequestObject);
                break;
            case Operation.FindRoba:
                response.Result = Controler.Instance.FindRoba((Roba)request.RequestObject);
                break;
            case Operation.SaveRoba:
                Controler.Instance.AddRoba((Roba)request.RequestObject);
                break;
            case Operation.GetAllRoba:
                response.Result = Controler.Instance.GetAllRoba();
                break;
            case Operation.UpdateRoba:
                Controler.Instance.UpdateRoba((Roba)request.RequestObject);
                break;
            case Operation.DeleteRoba:
                Controler.Instance.DeleteRoba((Roba)request.RequestObject);
```

```
                break;
            case Operation.SearchProductWith:
                response.Result = Controler.Instance.FindBusyPositions((Client)request.RequestObject,
(Roba)request.RequestObject2);
                break;
            case Operation.SearchAllParameters:
                response.Result = Controler.Instance.FindBusyPositionsWithPosition((Client)request.RequestObject,
(Roba)request.RequestObject2, request.Text);
                break;
            case Operation.AddEntrance:
                Controler.Instance.AddEntrance((Entrance)request.RequestObject);
                break;
            case Operation.GetAllEntrances:
                response.Result = Controler.Instance.GetAllEntrancecs();
                break;
            case Operation.GetAllPositions:
                response.Result= Controler.Instance.GetAllPositions();
                break;
            case Operation.FindPositions:
                response.Result = Controler.Instance.FindPositions(request.Text);
                break;
            case Operation.AddEntrancePosition:
                Controler.Instance.AddEntrancePosition((EntrancePosition)request.RequestObject);
                break;
            case Operation.FindEntrance:
                response.Result = Controler.Instance.FindEntrance((int)request.RequestObject);
                break;
            case Operation.FindBusyPosition:
                response.Result = Controler.Instance.FindBusyPositions((Client)request.RequestObject,
(Roba)request.RequestObject2);
                break;
            case Operation.ReturnEntrancePosition:
                response.Result = Controler.Instance.ReturnEntrancePosition((string)request.RequestObject);
                break;
            case Operation.ReturnEntranceItems:
                response.Result = Controler.Instance.ReturnEntranceItems((int)request.RequestObject);
                break;
            case Operation.LeaveEntrance:
                Controler.Instance.LeaveEntrance((Entrance)request.RequestObject);
                break;
            case Operation.LeavingEntranceItem:
                Controler.Instance.LeavingEntranceItems((List<LeavingItem>)request.RequestObject,
(List<EntranceItems>)request.RequestObject2);
                break;
            case Operation.ShowEntranceItems:
                response.Result = Controler.Instance.ShowEntranceItems((string)request.RequestObject);
                break;
            case Operation.ReturnClient:
                response.Result = Controler.Instance.ReturnClient((string)request.RequestObject);
                break;
            case Operation.ReturnRoba:
                response.Result = Controler.Instance.ReturnRoba((string)request.RequestObject);
                break;
            case Operation.GetWeightOfBox:
                response.Result = Controler.Instance.GetWeightOfBox((int)request.RequestObject);
                break;
            case Operation.FindStorekeeper:
                response.Result = Controler.Instance.FindStorekeeper((Storekeeper)request.RequestObject);
                break;
            case Operation.UpdateEntranceAndPositoin:
                Controler.Instance.UpdateEntranceAndPosition(Int32.Parse((string)(request.RequestObject)),
(string)request.RequestObject2);
                break;
            case Operation.RestartDatabase:
```

```
                Controler.Instance.RestartDatabase();
                break;
            case Operation.ReturnEntrance:
                response.Result = Controler.Instance.ReturnEntrances((string)request.RequestObject);
                break;
            case Operation.DeleteEntrance:
                Controler.Instance.DeleteEntrance((string)request.RequestObject);
                break;
            case Operation.FindBusyEntrances:
                response.Result = Controler.Instance.FindBusyEntrances((Client)request.RequestObject,
(Roba)request.RequestObject2);
                break;
            case Operation.FindBusyEntrancesWithDate:
                response.Result = Controler.Instance.FindBusyEntrancesWithDate((Client)request.RequestObject,
(Roba)request.RequestObject2, (DateTime)request.RequestObject3, (DateTime)request.RequestObject4);
                break;
            case Operation.FindOutputEntrances:
                response.Result = Controler.Instance.FindOutputEntrances((Client)request.RequestObject,
(Roba)request.RequestObject2);
                break;
            case Operation.FindOutputEntrancesWithDate:
                response.Result = Controler.Instance.FindOutputEntrancesWithDate((Client)request.RequestObject,
(Roba)request.RequestObject2, (DateTime)request.RequestObject3, (DateTime)request.RequestObject4);
                break;
            case Operation.ZauzetaPozicija:
                response.Result = Controler.Instance.ZauzetaPozicija((string)request.RequestObject);
                break;
        }
        return response;
    }
```

Класе које су одговорне да комуникација буде успешна и да се подаци успешно преносе су: Request и Response.

```
public class Request
  {
    public Object RequestObject { get; set; }
    public Operation Operation { get; set; }
    public Object RequestObject2 { get; set; }
    public Object RequestObject3 { get; set; }
    public Object RequestObject4 { get; set; }
    public string Text { get; set; }
  }

public class Response
  {
    public string Error { get; set; }
    public bool IsSuccessful { get; set; }
    public Object Result { get; set; }

  }
```

13.2.2 Пословна логика – доменске класе

Доменске класе су описане преко приватних атрибута, подразумеваних конструктора, преклапања *toString* метода као и имплементацију свих наслеђених метода интерфејса и доменских објеката.

Пример доменске класе *Магацоиер* је дато у наставку.

```
[Serializable]
   public class Storekeeper : IEntity
   {
      public int StorekeeperId { get; set; }
      public string Name { get; set; }
      public string LastName { get; set; }
      public string Username { get; set; }
      public string Password { get; set; }

      public override string ToString()
      {
         return $"{Name} {LastName}";
      }

      [Browsable(false)]
      public string TableName => "Storekeepers";
      [Browsable(false)]
      public string InsertValues => $"'{Name}','{LastName}','{Username}', '{Password}'";
      [Browsable(false)]
      public string IdName => "StorekeeperId";
[Browsable(false)]
      public object WhereValues => $" where StorekeeperId = {StorekeeperId}";
      [Browsable(false)]
      public object SetValues => $" set Name = '{Name}',LastName = '{LastName}', Username='{Username}'," +
         $" Password = '{Password}' where StorekeeperId = {StorekeeperId}";
      [Browsable(false)]
      public object ExistName => $" where Username = '{Username}'";
      // za log in
      [Browsable(false)]
      public object WhereReturn => $" where Username= '{Username}' and Password='{Password}'";

}
```

13.2.3 Пословна логика – системске операције

Свака системска операција је изведена из класе *SystemOperationBase* која ће бити приказана у наставку:

```
public abstract class SystemOperationBase
   {
      protected IGenericRepository repository;

      public SystemOperationBase()
      {
         repository = new GenericRepository();
      }

      public void ExecuteTemplate(IEntity entity)
      {
         try
         {
            repository.OpenConnection();
            repository.BeginTransaction();
            ExecuteOperation(entity);
            repository.Commit();
         }
         catch (Exception ex)
         {
            repository.Rollback();
            throw;
         }
         finally
```

```
        {
            repository.CloseConnection();
        }
    }

    protected abstract void ExecuteOperation(IEntity entity);
}
```

У наставку ће кроз уговоре бити приказана имплементација сваке системске операције.

Уговор У1: ВратиСвеКлијенте

Операција: GetAllClients(List<Client>):signal

Веза са СК: СК1, СК3, СК4, СК5

Предуслови: /

Постуслови: /

```
public class GetAllClientsSO : SystemOperationBase
{
    public List<Client> Result{ get; private set; }
    protected override void ExecuteOperation(IEntity entity)
    {
        Result = repository.GetAll(entity).Cast<Client>().ToList();
    }
}
```

Уговор У2: ВратиСвеАртикле

Операција: GetAllRoba(List<Client>):signal

Веза са СК: СК1, СК3, СК5, СК8, СК6

Предуслови: /

Постуслови: /

```
public class GetAllRobaSO : SystemOperationBase
{
    public List<Roba> Result{ get; set; }
    protected override void ExecuteOperation(IEntity entity)
    {
        Result = repository.GetAll(entity).Cast<Roba>().ToList();
    }
}
```

Уговор У3: ВратиСвеУлазе

Операција: GetAllEntrance(List<Entrance>):signal

Веза са СК: СК2

Предуслови: /

Постуслови: /

```
public class GetBusyEntrancesSO : SystemOperationBase
  {
    public Client Client { get; set; }
    public Roba Roba { get; set; }
    public DataTable Result { get; set; }
    protected override void ExecuteOperation(IEntity entity)
    {
      Result = repository.GetBusyEntrances((Entrance)entity, Client, Roba);
    }
  }
```

Уговор У4: ВратиСвеПозиције

Операција: GetAllPosition(List<Position>):signal

Веза са СК: СК2

Предуслови: /

Постуслови: /

```
public class GetAllPositionsSO : SystemOperationBase
  {
    public List<Position> Result { get; set; }
    protected override void ExecuteOperation(IEntity entity)
    {
      Result = repository.GetAll(entity).Cast<Position>().ToList();
    }
  }
```

Уговор У5: ВратиСвеУлазнеСтавке

Операција: GetAllEntranceItems(List<Position>):signal

Веза са СК: СК3,СК5

Предуслови: /

Постуслови: /

```
public class SearchEntranceItemsSO : SystemOperationBase
  {
    public List<EntranceItems> Result{ get; set; }
    public int Uslov { get; set; }
    protected override void ExecuteOperation(IEntity entity)
    {
      Result = repository.Search(entity, Uslov).Cast<EntranceItems>().ToList();
    }
  }
```

Уговор У6: ОбришиРобу

Операција: Delete(Roba):signal

Веза са СК: СК8

Предуслови: Вредносна и структурна ограничења над објектом Roba морају бити задовољена. Учитана је листа Робе.

Постуслови: Роба је обрисана.

```
public class DeleteRobaSO : SystemOperationBase
  {
    protected override void ExecuteOperation(IEntity entity)
    {
      repository.Delete(entity);
    }
  }
```

Уговор У7: АжурирајРобу

Операција: Update(Roba):signal

Веза са СК: СК7

Предуслови: Вредносна и структурна ограничења над објектом Роба морају бити задовољена. Учитана је листа Робе.

Постуслови: Роба је ажурирана.

```
public class UpdateRobaSO : SystemOperationBase
  {
    protected override void ExecuteOperation(IEntity entity)
    {
      repository.Update(entity);
    }
  }
```

Уговор У8: Учитај

Операција: Load(Roba):signal

Веза са СК: СК3, СК5, СК7, СК8

Предуслови: Вредносна и структурна ограничења над објектом Податак морају бити задовољена. Учитана је листа Робе.

Постуслови: Роба је ажурирана.

```
public class ReturnRobaSO : SystemOperationBase
  {
    public Roba Result{ get; set; }
    public string Uslov{ get; set; }
    protected override void ExecuteOperation(IEntity entity)
    {
      Result = (Roba)repository.Return(entity, Uslov);
    }
  }
```

Уговор У9: СачувајРобу

Операција: Save(Roba):signal

Веза са СК: СК6

Предуслови: Вредносна и структурна ограничења над објектом Роба морају бити задовољена. Учитана је листа Робе.

Постуслови: Роба је унешена.

```
public class AddNewClientSO : SystemOperationBase
  {
    protected override void ExecuteOperation(IEntity entity)
    {
      repository.Save(entity);
    }
  }
```

Уговор У10: ИлазСтавкеПалете

Операција: LeaveEntranceItem(EntranceItem):signal

Веза са СК: СК5

Предуслови: Вредносна и структурна ограничења над објектом УлазнаСтавка морају бити задовољена. Учитана је листа улаза као и листа података (Клијент, Роба, Магационер).

Постуслови: Ставка је изашла.

```
public class LeavingEntranceItemSO : SystemOperationBase
  {
    public List<LeavingItem> LeavingItem { get; set; }
    public List<EntranceItems> EntranceItem { get; set; }
    protected override void ExecuteOperation(IEntity entity)
    {
      foreach (EntranceItems ei in EntranceItem)
      {
        repository.Update(ei);
      }
      foreach (LeavingItem li in LeavingItem)
      {
        repository.Save(li);
      }
    }
  }
```

Уговор У11: ИзбациРобуСаПозиције

Операција: LeaveEntrancePosition(Entrance,Position):signal

Веза са СК: СК4

Предуслови: Вредносна и структурна ограничења над објектом Улаз и Позиција морају бити задовољена. Учитана је листа свих палета у складишту.

Постуслови: Палета је изашла.

```
public class UpdateEntranceAndPosition : SystemOperationBase
  {
    public int EntranceId { get; set; }
    public string PositionId { get; set; }
    protected override void ExecuteOperation(IEntity entity)
    {
      Entrance e = (Entrance)repository.Return(entity, EntranceId);
      Position p = (Position)repository.Return3(new Position(), PositionId);
```

```
            repository.UpdateWithParameters3(e, PositionId);
            repository.UpdateWithParameters(p, p.PositionId);
        }
    }
```

Уговор У12: Претрага

Операција: Search(Roba,Client,pozicija):signal

Веза са СК: СК3, СК4

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

Постуслови: Палета је пронађена.

```
public class SearchSO : SystemOperationBase
    {
        public List<Position> Result{ get; set; }
        public string Uslov{ get; set; }
        protected override void ExecuteOperation(IEntity entity)
        {
            Result = repository.Search(entity, Uslov).Cast<Position>().ToList();
        }
    }
```

Уговор У13: УнесиПалету

Операција: AddEntrancePosition(Entrance,Position):signal

Веза са СК: СК2

Предуслови: Вредносна и структурна ограничења над објектом Улаз и Позиција морају бити задовољена. Учитана је листа улазних позиција.

Постуслови: Улазна позиција са палетом је унешена.

```
public class UpdateEntranceAndPosition : SystemOperationBase
    {
        public int EntranceId { get; set; }
        public string PositionId { get; set; }
        protected override void ExecuteOperation(IEntity entity)
        {
            Entrance e = (Entrance)repository.Return(entity, EntranceId);
            Position p = (Position)repository.Return3(new Position(), PositionId);

            repository.UpdateWithParameters3(e, PositionId);
            repository.UpdateWithParameters(p, p.PositionId);
        }
    }
```

Уговор У14: ДодајПалету

Операција: AddEntrance (Entrnace):signal

Веза са СК: СК1

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

Постуслови: Улазна палета је унешена.

```
public class AddNewEntranceSO : SystemOperationBase
  {

    protected override void ExecuteOperation(IEntity entity)
    {
      Entrance e = (Entrance)entity;
      e.EntranceId = (int)repository.SaveEntrance(entity);
      foreach (EntranceItems ei in e.Items)
      {
        ei.EntranceId = e.EntranceId;
        repository.SaveEntranceItem(ei);
      }
    }
  }
```

Уговор У15: КреирајРобу

Операција: Kreiraj(Roba):signal

Веза са СК: СК7

Предуслови: Вредносна и структурна ограничења над објектом Роба морају бити задовољена. Учитана је листа Робе.

Постуслови: Роба је креиран.

```
public class AddNewRobaSO : SystemOperationBase
  {
    protected override void ExecuteOperation(IEntity entity)
    {
      repository.Save(entity);
    }
  }
```

13.2.4 Брокер базе података

Брокер базе података има задатак да приступи и успостави комуникацију са базом података. Када се креира инстанца класе успоставља се конекција са базом преко конекционог стринга. Методе коју су основа брокера базе података су: отварање конекције, затварање конекције, покретање трансакције, успешно извршена трансакција и поништавање трансакције. Трансакција је јако важна при уносу сложених објеката у базу.

Брокер поседује CRUD операције, односно операције за креирање, читање, измену и брисање објеката из базе података.

```
public class Broker
  {
    private SqlTransaction transaction;
    private SqlConnection connection;
```

```
        public Broker()
        {
            connection = new
SqlConnection(ConfigurationManager.ConnectionStrings["BeletrixDatabase"].ConnectionString);
        }

        public List<IEntity> GetAll(IEntity entity)
        {
            List<IEntity> result;
            SqlCommand command = new SqlCommand("", connection, transaction);
            command.CommandText = $"select * from {entity.TableName} {entity.WherePosition}";
            SqlDataReader reader = command.ExecuteReader();
            result = entity.GetEntities(reader);
            reader.Close();
            return result;
        }

        public void Save(IEntity entity)
        {
            SqlCommand command = new SqlCommand("", connection, transaction);
            command.CommandText = $"insert into {entity.TableName} values ({entity.InsertValues})";
            if (command.ExecuteNonQuery() != 1)
            {
                throw new Exception("Database error!");
            }
        }
            public void Update(IEntity entity)
        {
            SqlCommand command = new SqlCommand("", connection, transaction);
            command.CommandText = $"update {entity.TableName} {entity.SetValues}";
            if (command.ExecuteNonQuery() != 1)
            {
                throw new Exception("Database error!");
            }
        }
            public void Delete(IEntity entity) {

            SqlCommand command = new SqlCommand("", connection, transaction);
            command.CommandText = $"delete from {entity.TableName} {entity.WhereValues}";
            if (command.ExecuteNonQuery() != 1)
            {
                throw new Exception("Database error!");
            }
        }
            public void OpenConnection() {
            connection.Open();
        }

        public void CloseConnection() {
            connection.Close();
        }

        public void Commit() {
            transaction?.Commit();
        }

        public void Rollback() {
            transaction?.Rollback();
        }

        public void BeginTransaction() {
            transaction = connection.BeginTransaction();
        }
```

}

13.3 Имплементација складишта података

На основу софтверских класа, пројектоване су табеле (складиште података) релационог система за управљање базом података. У овом раду коришћен је *Microsoft SQL Server*.

	Name	Data Type	Allow Nulls	Default
🔑	ClientId	int	☐	
	Name	varchar(50)	☐	
	Place	varchar(50)	☑	
	PIB	int	☐	
	Telephone	int	☑	
	Email	varchar(50)	☑	
			☐	

Слика 107 - Табела Клијент

	Name	Data Type	Allow Nulls	Default
🔑	EntranceId	int	☐	
	Weight	float	☑	
	DateOfEntrance	datetime	☐	
	DateOfExit	datetime	☑	(NULL)
	Dimension	varchar(50)	☑	
	Obradjen	bit	☑	
	ClientId	int	☐	
	StorekeeperId	int	☐	
	PositionId	nchar(4)	☑	(NULL)
	Aktivno	bit	☑	
			☐	

Слика 108 - Табела Улазна палета

	Name	Data Type	Allow Nulls	Default
🔑	Num	int	☐	
🔑	EntranceId	int	☐	
	RobaId	int	☐	
	DeadlineDate	date	☐	
	Deadline	bit	☐	((0))
	NumOfBoxes	float	☐	
	DateOfManu	date	☐	
			☐	

Слика 109 - Табела Улазна ставка

	Name	Data Type	Allow Nulls	Default
🔑	LeavingItemId	int	☐	
🔑	EntranceId	int	☐	
🔑	Num	int	☐	
	RobaId	int	☐	
	NumOfBoxes	float	☐	
	DateOfLeaving	datetime	☐	
			☐	

Слика 110 - Табела Излазна ставка

Update | Script File: dbo.Positions.sql

	Name	Data Type	Allow Nulls	Default
🔑	PositionId	varchar(4)	☐	
	Slobodno	bit	☑	((0))
			☐	

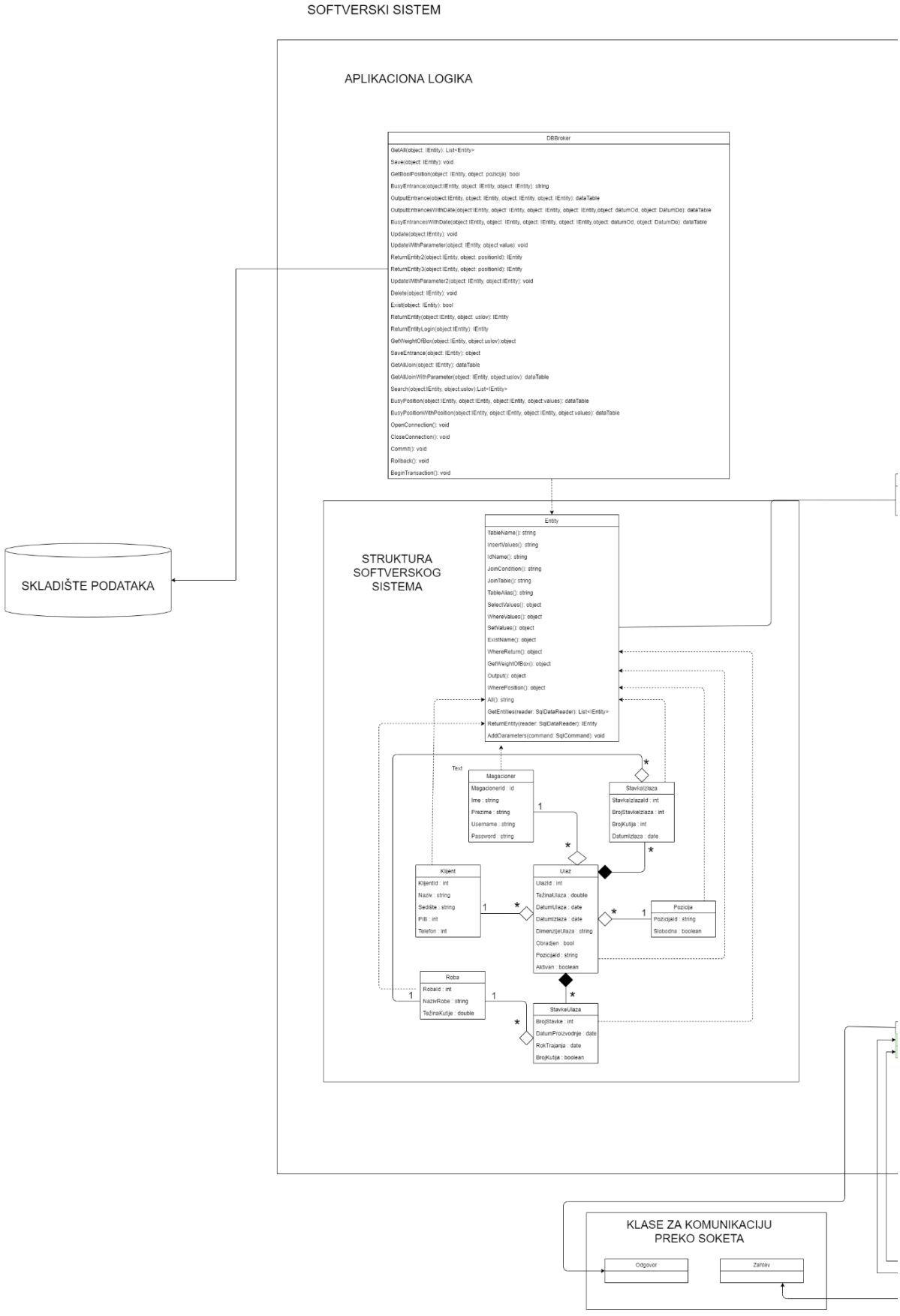
Слика 111 - Табела Позиција

	Name	Data Type	Allow Nulls	Default
🔑	RobaId	int	☐	
	Name	varchar(50)	☐	
	WeightOfBox	float	☑	
			☐	

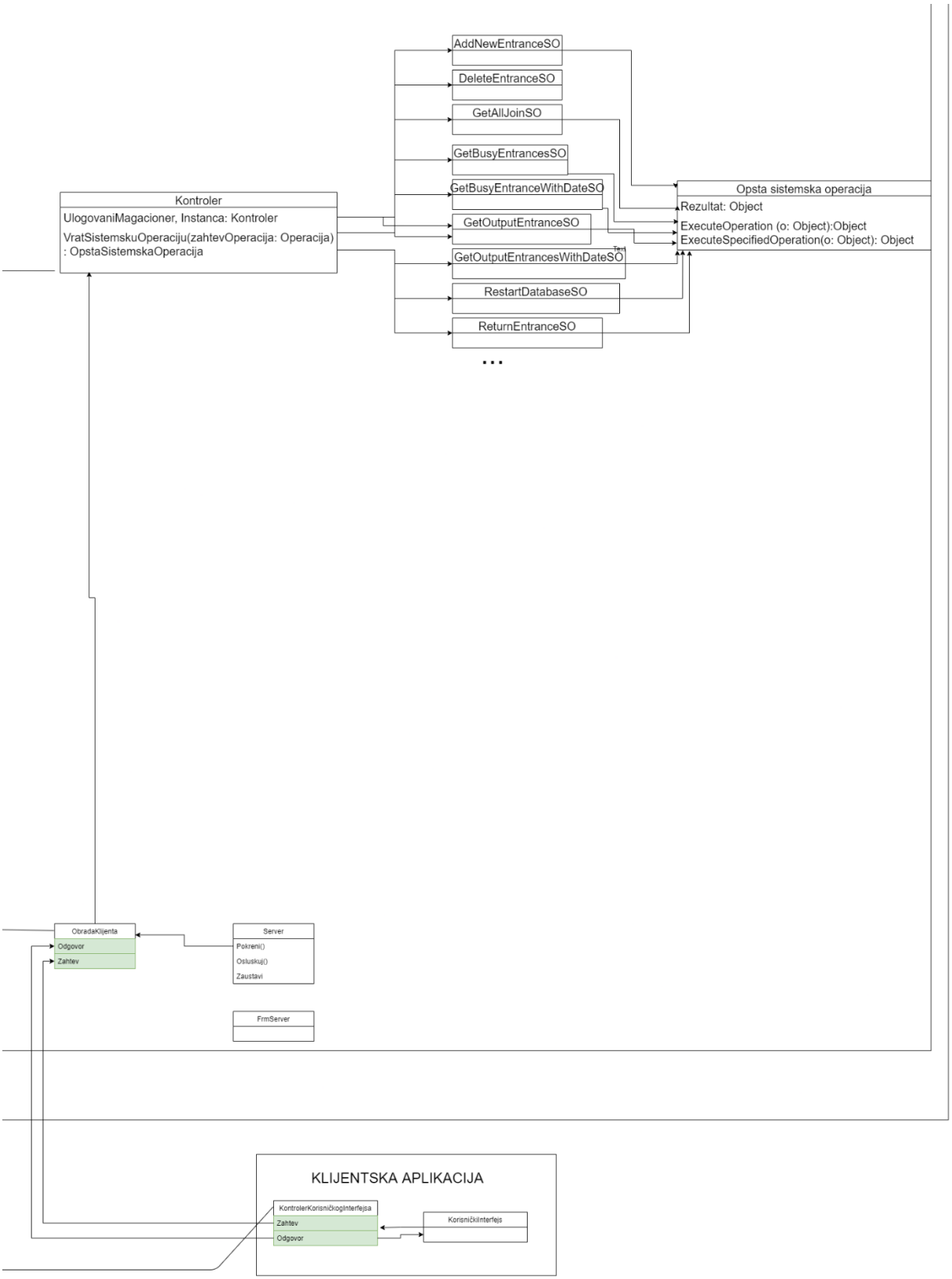
Слика 112 - Табела Роба

	Name	Data Type	Allow Nulls	Default
🔑	StorekeeperId	int	☐	
	Name	varchar(50)	☑	
	LastName	varchar(50)	☑	
	Username	varchar(50)	☐	
	Password	varchar(50)	☐	
			☐	

Слика 113 - Табела Магационер



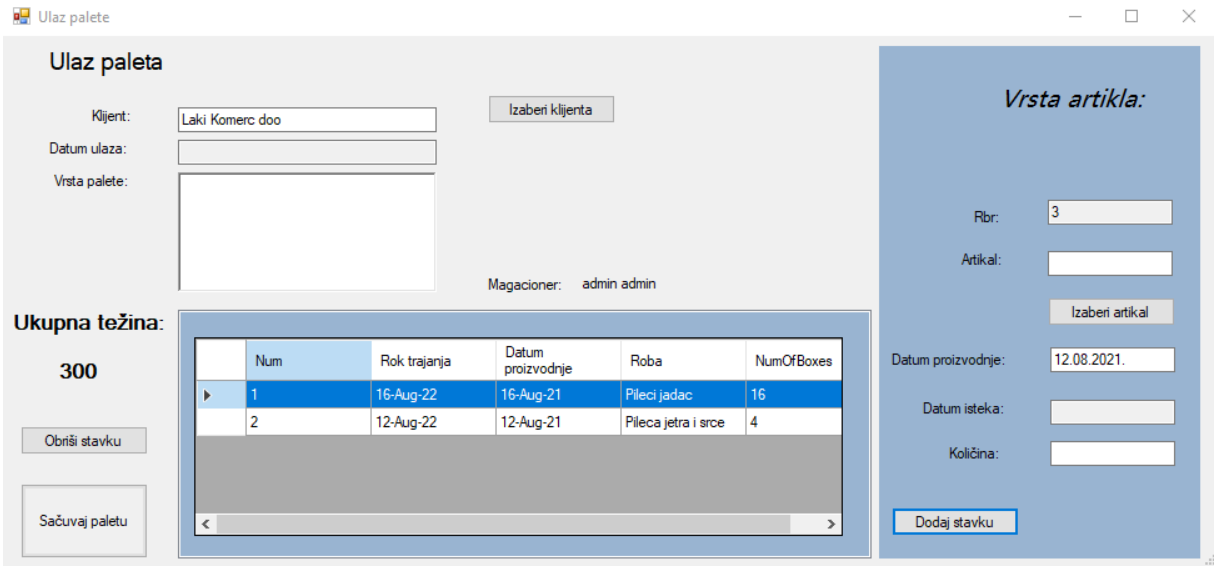
Слика 114 - Архитектура софтверског система 1. део



Слика 115 - Архитектура софтверског система 2. део

13.4 Имплементација корисничког интерфејса

Пример имплементације корисничког интерфејса ће бити објашњен путем форме за креирање улаза (палете) у складиште, као и контролера корисничког интерфејса. Форма позива свој контролер и тиме покреће операције. Операције које се односе на приказ директно ка кориснику се налазе на клијентској страни, док операције које захтевају приступ бази захтевају слање захтева ка серверу. На тај начин за сваку интервенцију сервера клијент шаље захтев, и све добијене излазе приказује путем корисничког интерфејса.



Слика 116 - Пример форме за унос новог улаза (палете)

На слици 112. приказана је форма за унос нове палете у складиште. У том случају се позивају форме које то и обезбеђују. Поред горе представљене форме, позивају се и још 2 форме када се бирају клијент и роба.

```
public partial class FrmEntrance : Form
   {
private void btnSaveComplete_Click(object sender, EventArgs e)
      {
         entranceController.Save(this);
      }
}
```

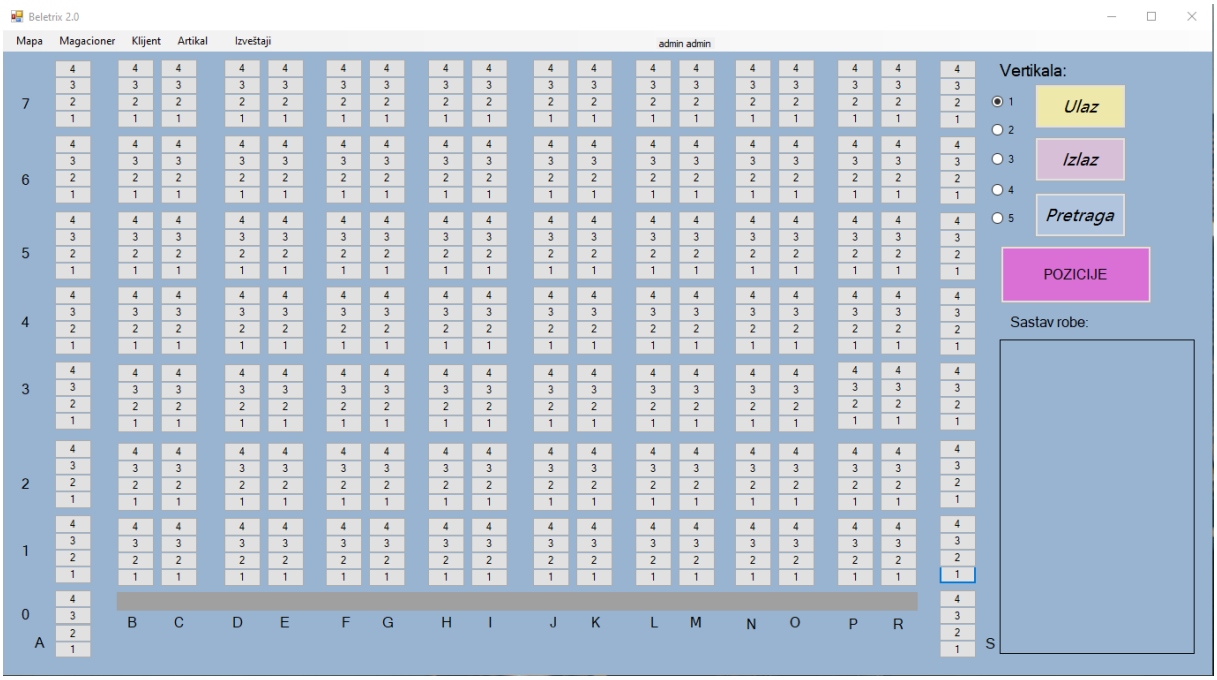
Након притиска на дугме “Sačuvaj paletu” позива се горе поменута метода која је имплементирана на следећи начин:

```
internal void Save(FrmEntrance frmEntrance)
      {
         if (string.IsNullOrEmpty(frmEntrance.LblTotalWeight.Text))
         {
            MessageBox.Show("Nema podataka!");
            return;
         }

         double TotalWeight = double.Parse(frmEntrance.LblTotalWeight.Text);

         try
         {
```

```
            Entrance.TotalWeight = TotalWeight;
            Entrance.Items = Items;
            Communication.Communication.Instance.AddEntrance(Entrance);
            MessageBox.Show("Uspešno sačuvan");
            frmEntrance.Dispose();
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
        }
    }
```

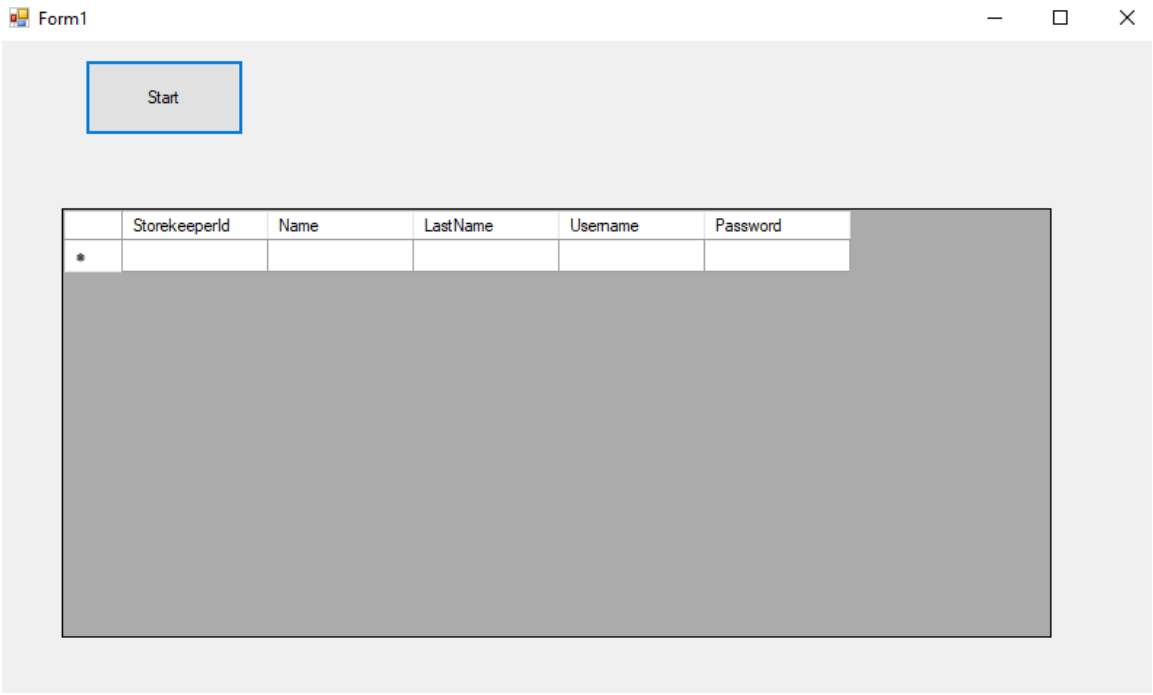
У наставку се могу видети остали кориснички интерфејси:

Главна екранска форма клијентског дела апликације изгледа овако:



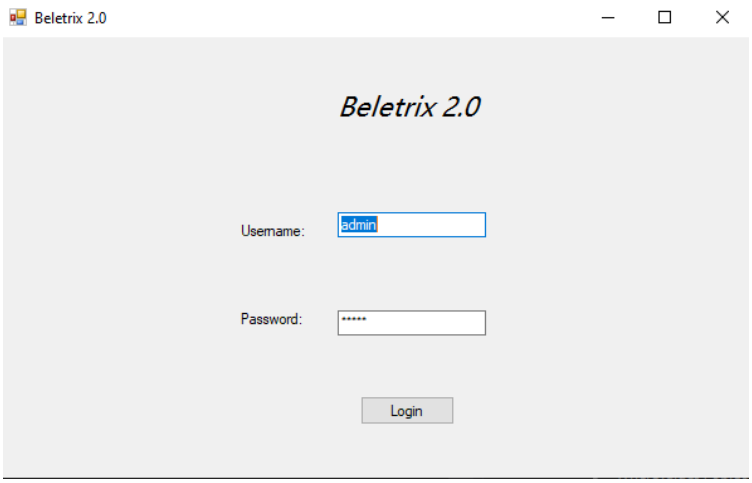
Слика 117 - Почетни интерфејс

Главна екранска форма серверског дела апликације је урађена овако:



Слика 118 - Главна екранска форма сервера

Клијент се мора улоговати како би користио апликацију, односно дошао до главне екранске форме. Форма за пријаву изгледа овако:



Слика 119 - Екранска форма за пријаву

13.4.1 СК1: Креирање улаза палета

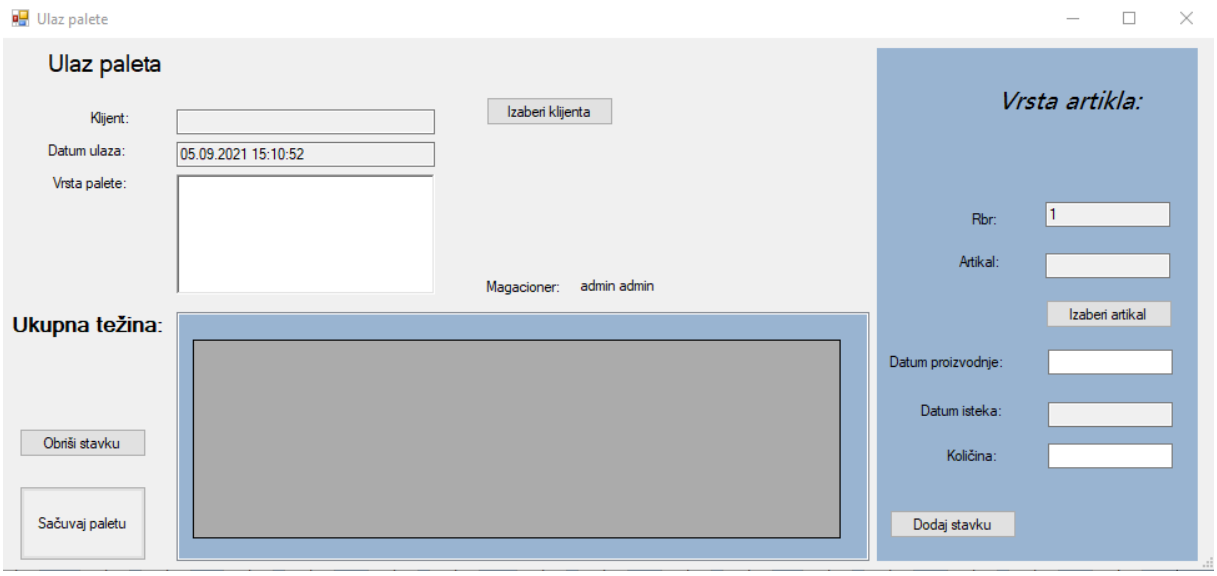
Назив СК
Креирање улаза палета

Актори СК
Магационер

Учесници СК
Магационер и систем (програм)

Предуслов: Систем је укључен и магационер је пријављен под својом шифром. Систем приказује форму за рад са улазом палета у складиште. Учитана је листа свих клијената и артикала са својим подацима.

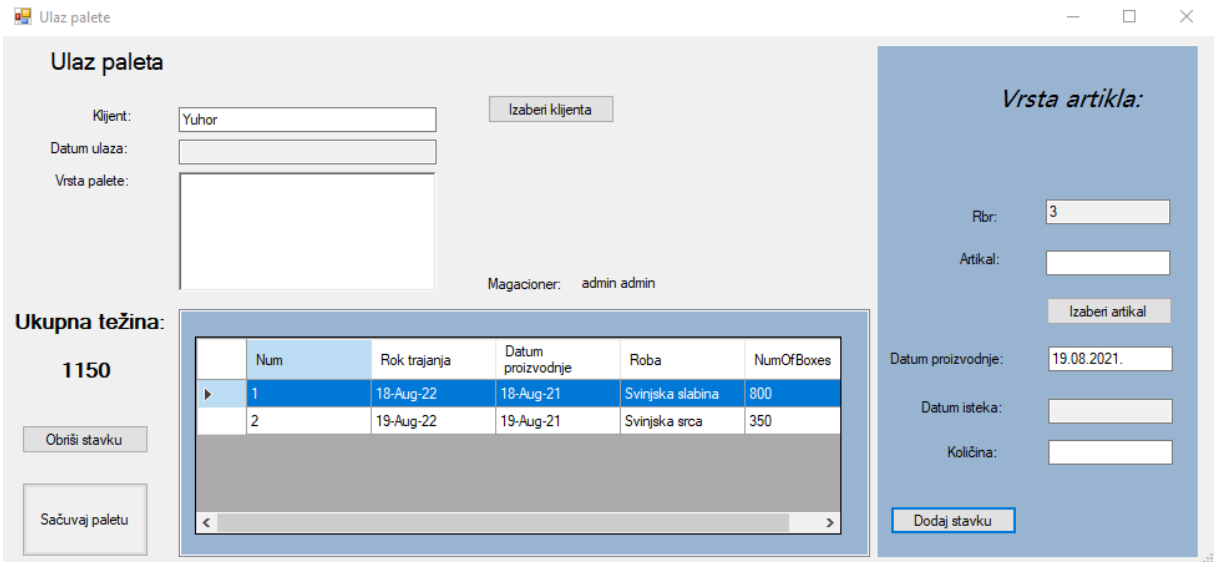
Систем приказује форму за унос улаза (палете) у складиште.



Слика 120 - СК1 Креирање улаза

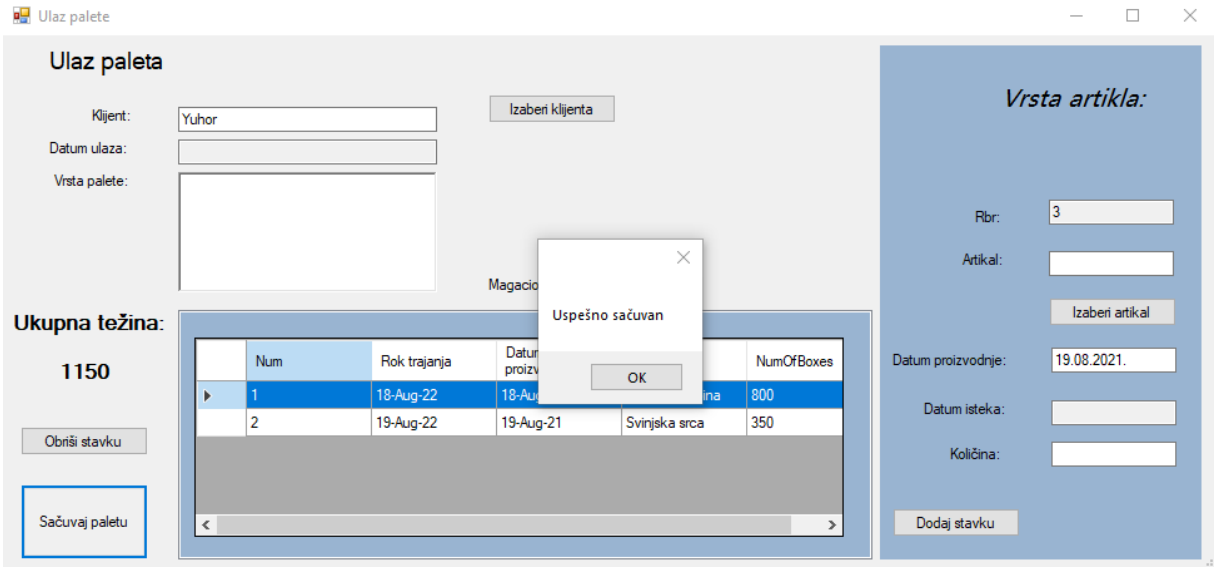
Основни сценарио СК

1. Магационер **уноси** податке у улазу. (АПУСО)



Слика 121 - СК1 Креирање улаза - Основни сценарио - унос података

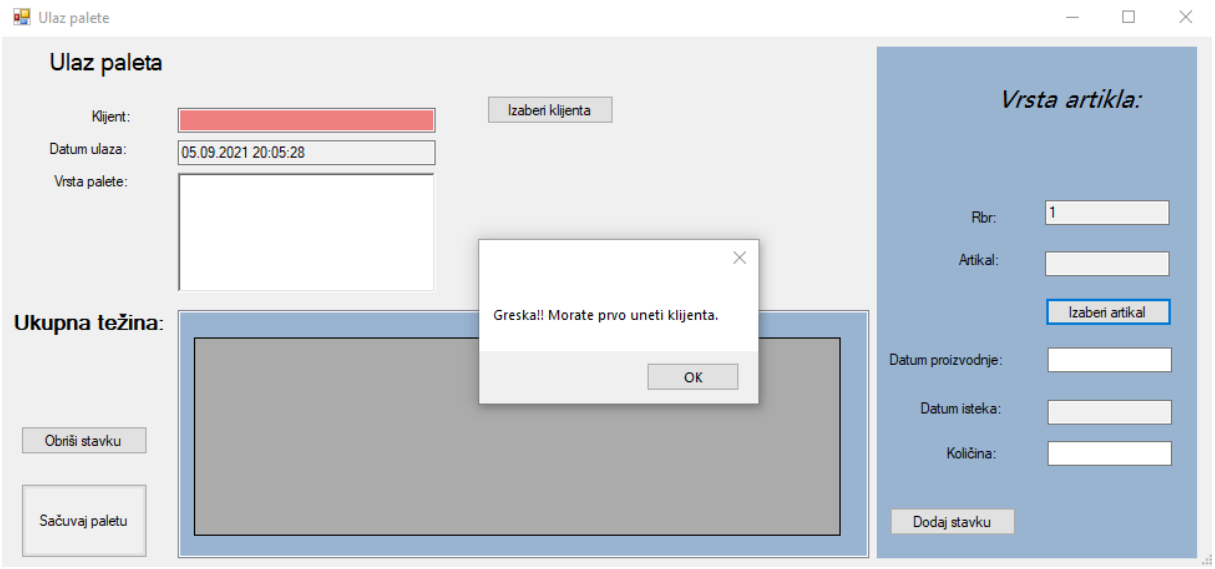
2. Магационер **контролише** да ли је коректно унео податке о улазу. (АНСО)
3. Магационер **позива** систем да запамти податке о улазу палете. (АПСО)
4. Систем **памти** податке о улазној палети. (СО)
5. Систем **приказује** магационеру запамћени улаз и поруку: "Успешно сачуван". (ИА)



Слика 122 - СК1 Креирање улаза - Основни сценарио - улаз је додат у систем

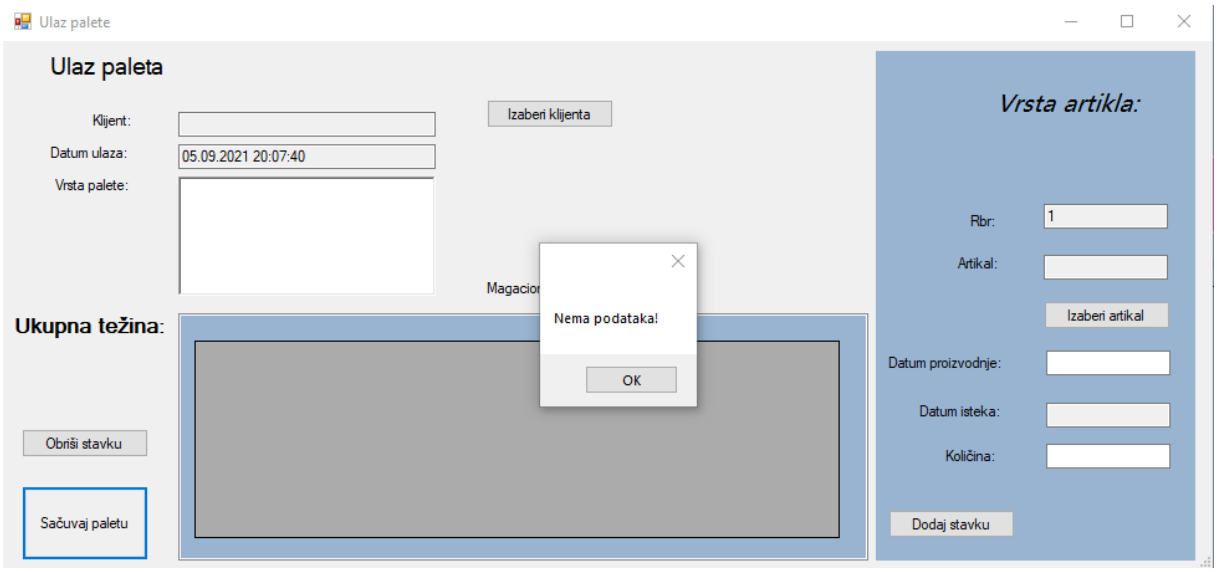
Алтернативни сценарио:

5.1 Уколико систем не може да креира улаз он приказује магационеру поруку: “Грешка!!! Морате прво креирати клијента.”. Прекида се извршење сценарија. (ИА)



Слика 123 - СК1 Креирање улаза - Алтернативни сценарио 1

5.1 Уколико систем не може да запамти податке о улазу палете он приказује магационеру поруку “Нема података”. (ИА)



Слика 124- СК1 Креирање улаза - Алтернативни сценарио 2

13.4.2 СК2: Креирање позиције у складишту

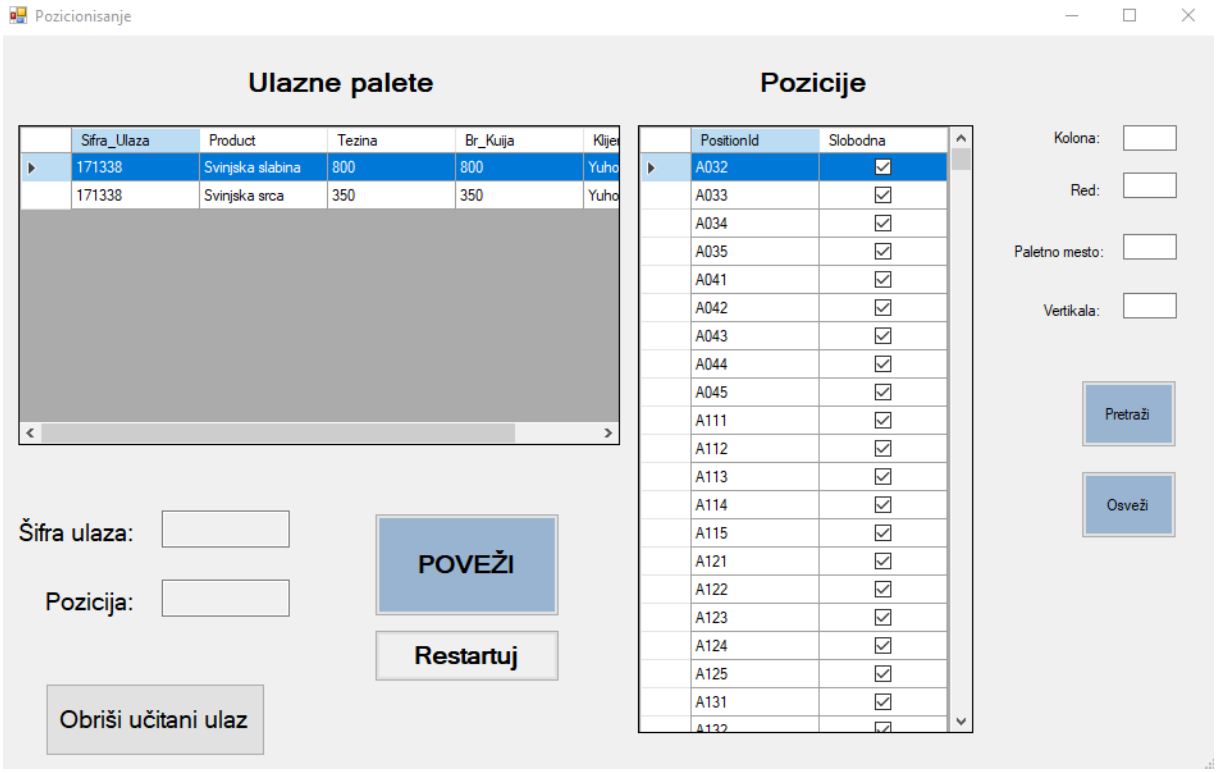
Назив СК
Креирање позиције

Актори СК
Магационер

Учесници СК
Магационер и систем (програм)

Предуслов: Систем је укључен и магационер је пријављен под својом шифром. Систем приказује форму за рад са позицијом. Учитани су подаци о улазу и позицијама.

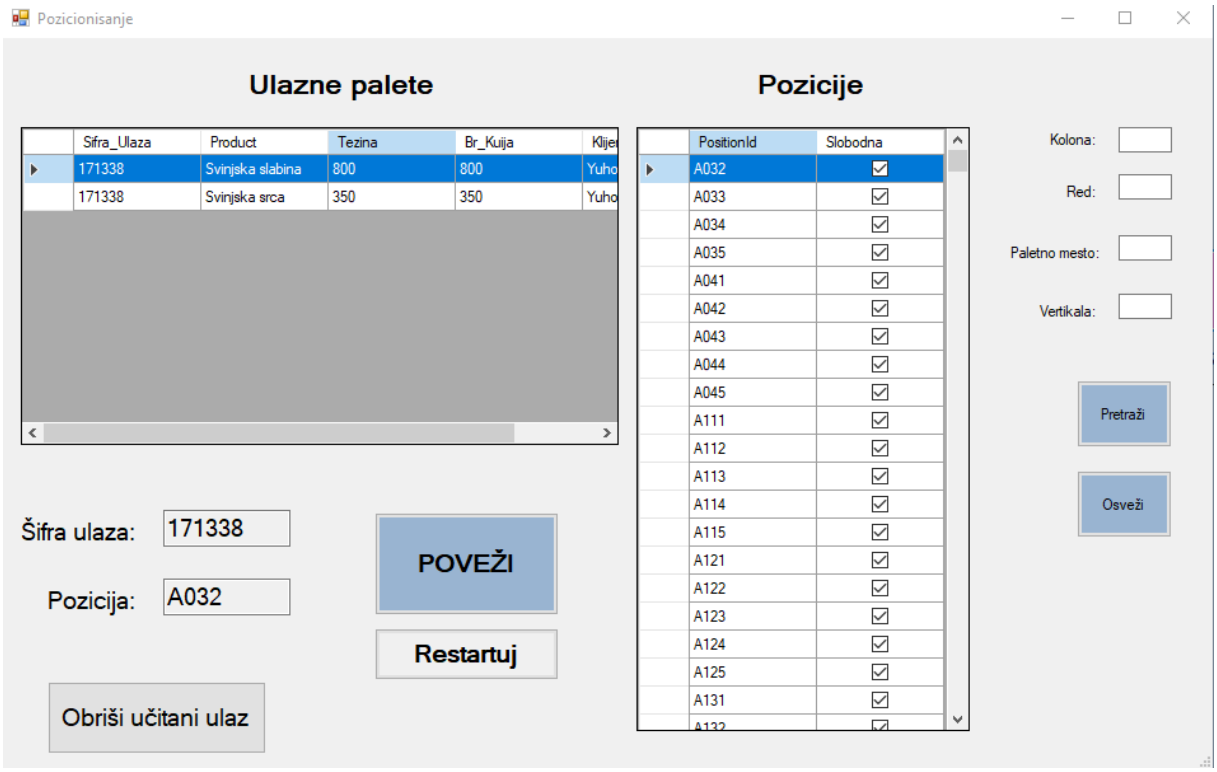
Систем приказује форму за креирање позиције у складишту.



Слика 125 - СК2 Креирање позиције

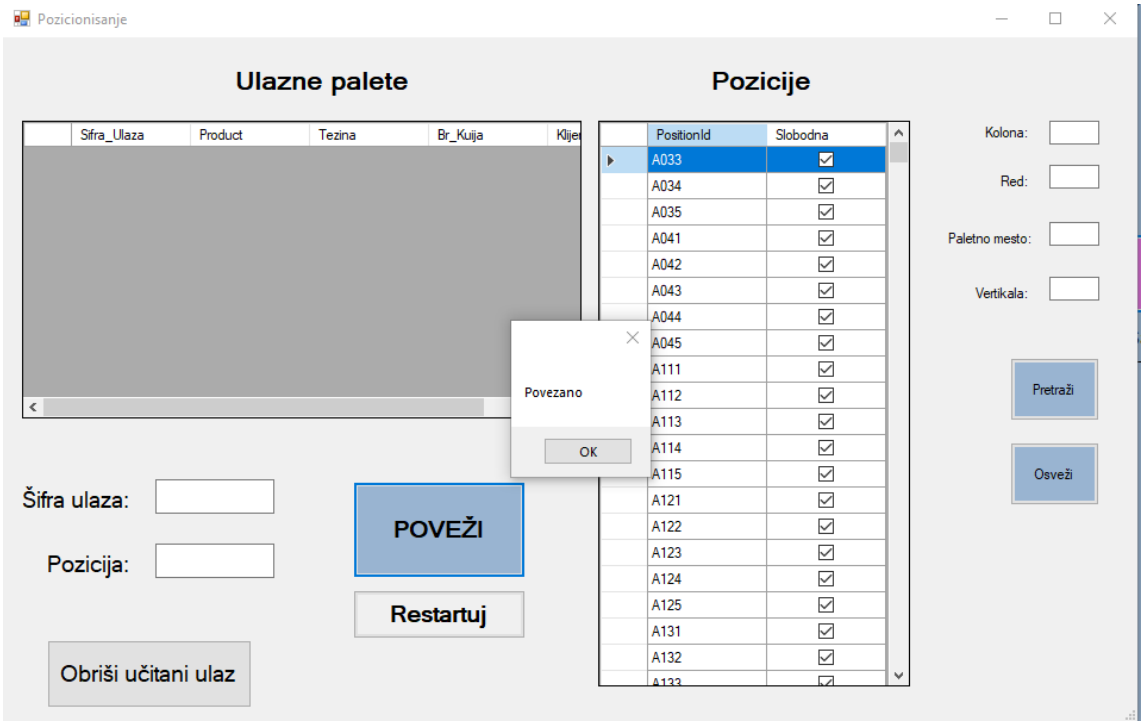
Основни сценарио СК:

1. Магационер **позива** систем да креира позицију. (АПСО)
2. Систем **креира** позицију. (СО)
3. Систем **приказује** форму“. (ИА)
4. Магационер **уноси** податке о позицији. (АПУСО)



Слика 126 - СК2 Креирање позиције - Основни сценарио - унос података

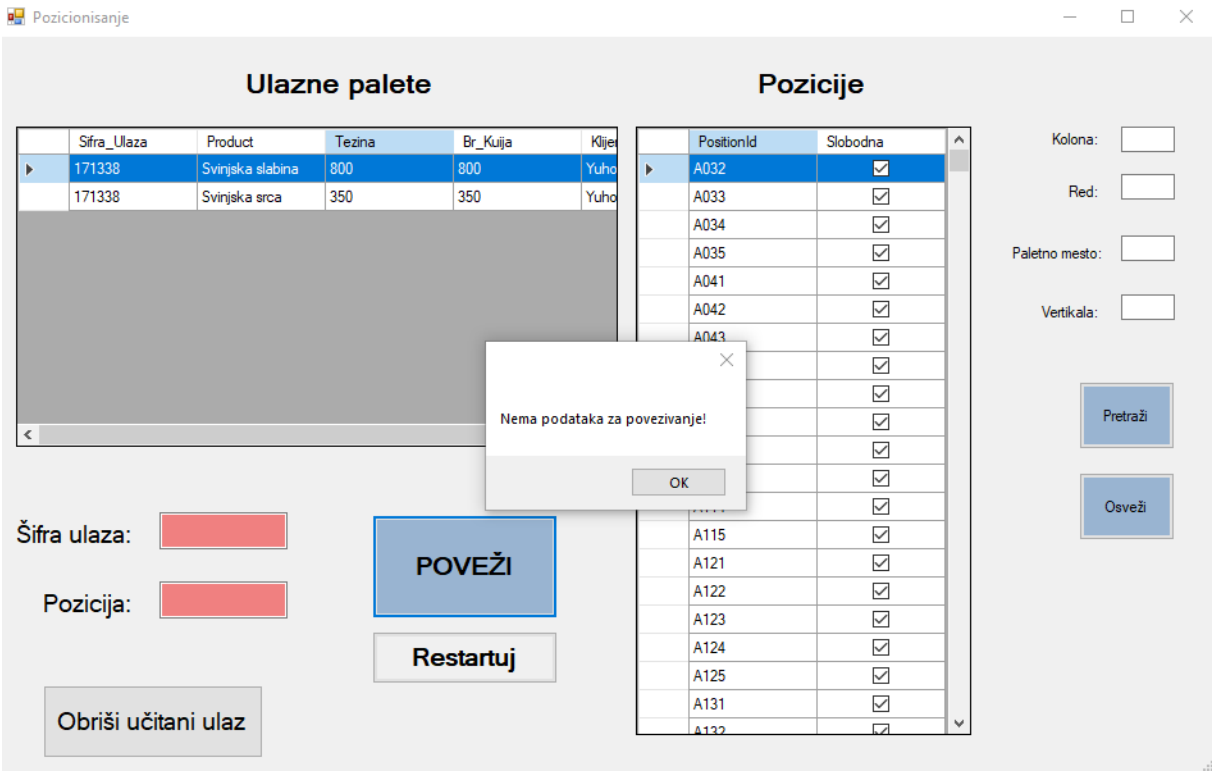
5. Магационер **контролише** да ли је коректно унео податке о позицији. (АНСО)
6. Магационер **позива** систем да запамти податке о позицији. (АПСО)
7. Систем **памти** податке о позицији. (СО)
8. Систем **приказује** магационеру запамћену позицију и поруку: "Повезано". (ИА)



Слика 127 - СК2 Креирање позиције - Основни сценарио - успешно повезивање

Алтернативни сценарио:

8.1 Уколико систем не може да запамти податке о позицији он приказује магационеру поруку “Нема података за повезивање”. (ИА)



Слика 128 - СК2: Креаирање позиције - Алтернативни сценарио - неуспешно повезивање позиције и улаза

13.4.3 СК3: Претраживање робе по палетама

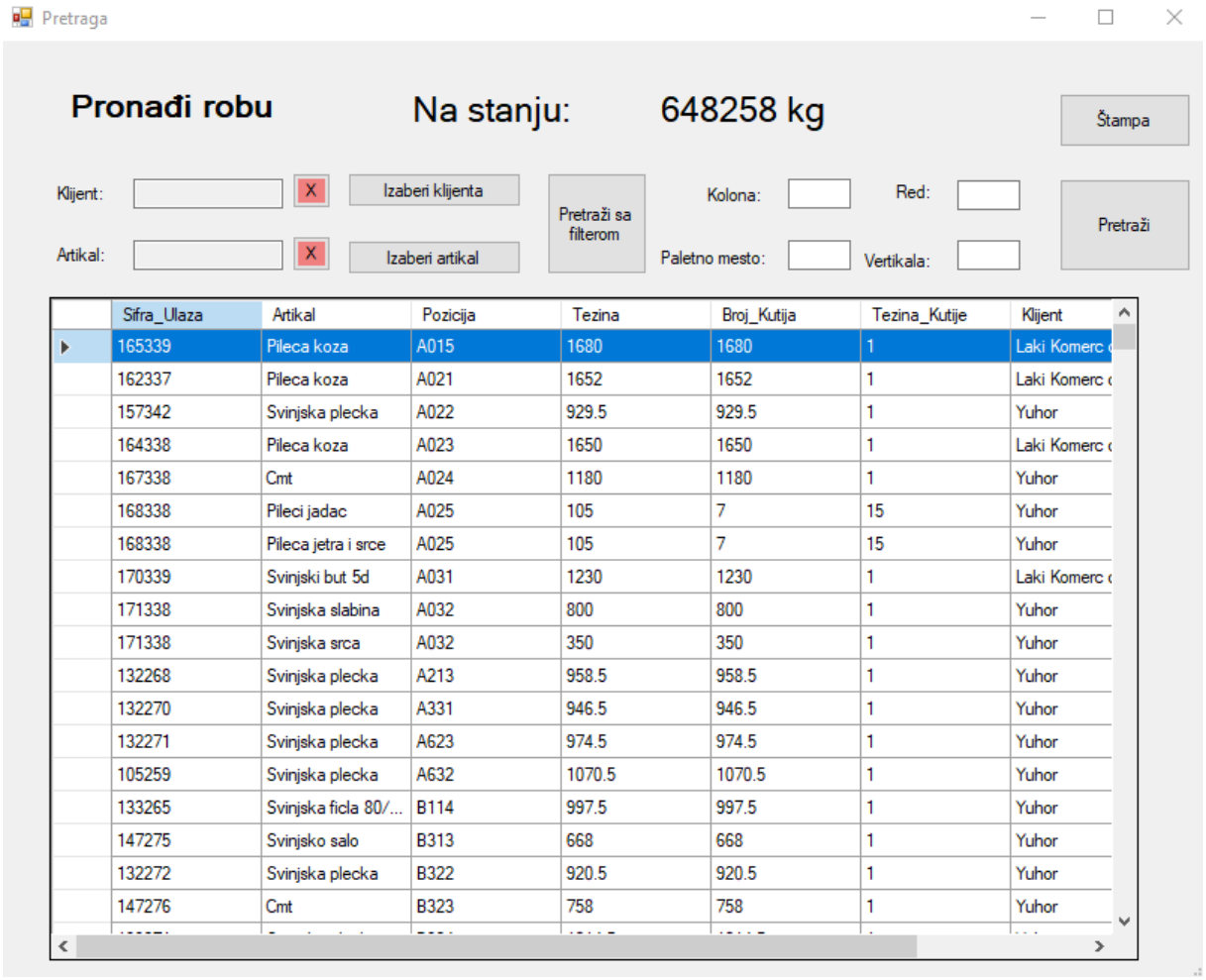
Назив СК
Претраживање робе

Актори СК
Магационер

Учесници СК
Магационер и систем (програм)

Предуслов: Систем је укључен и магационер је пријављен под својом шифром. Систем приказује форму за рад са робом. Учитана је листа свих улазних ставки у складишту.

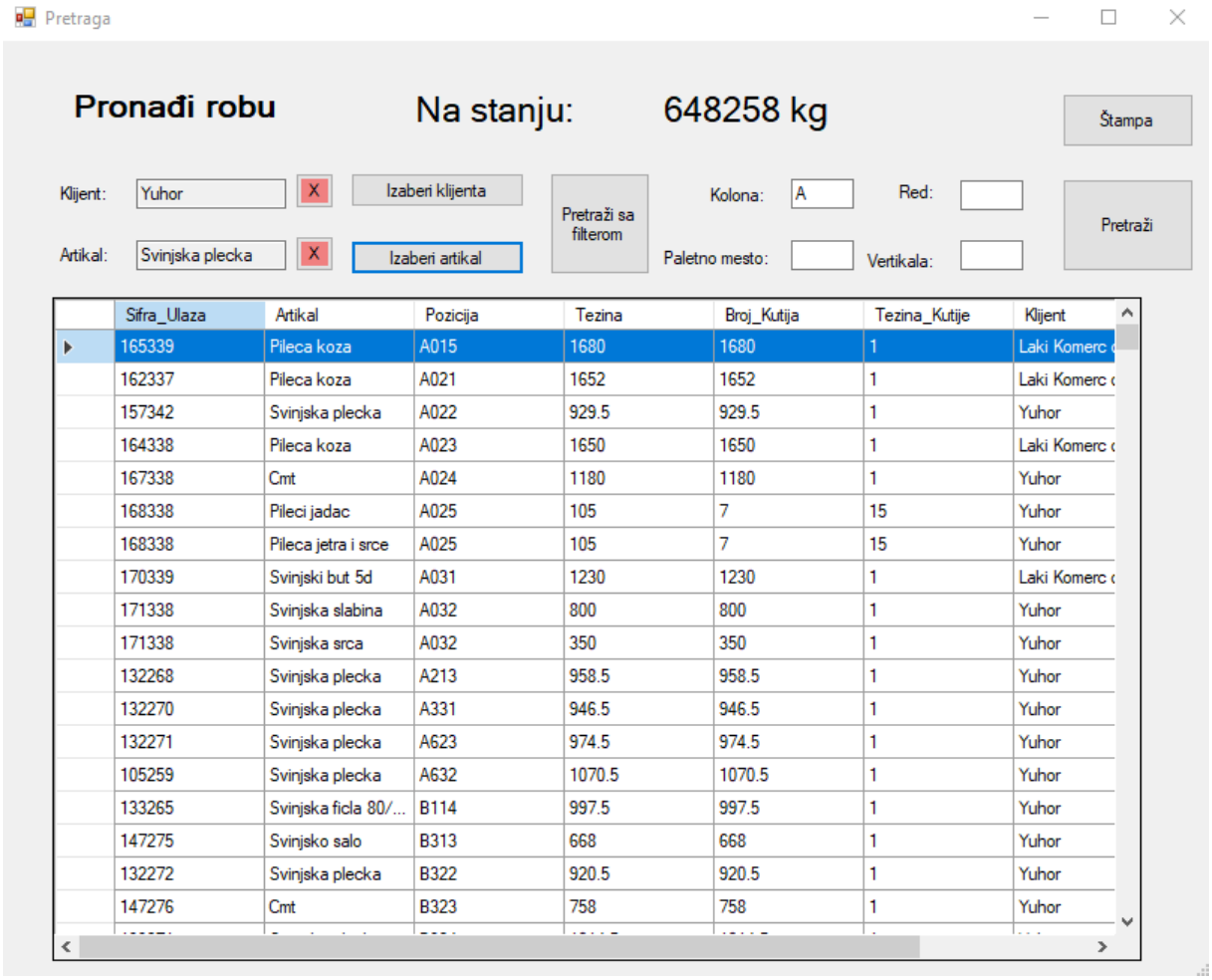
Систем приказује форму за претраживање улаза (палете) у складиште.



Слика 129 - СК3: Претраживање робе

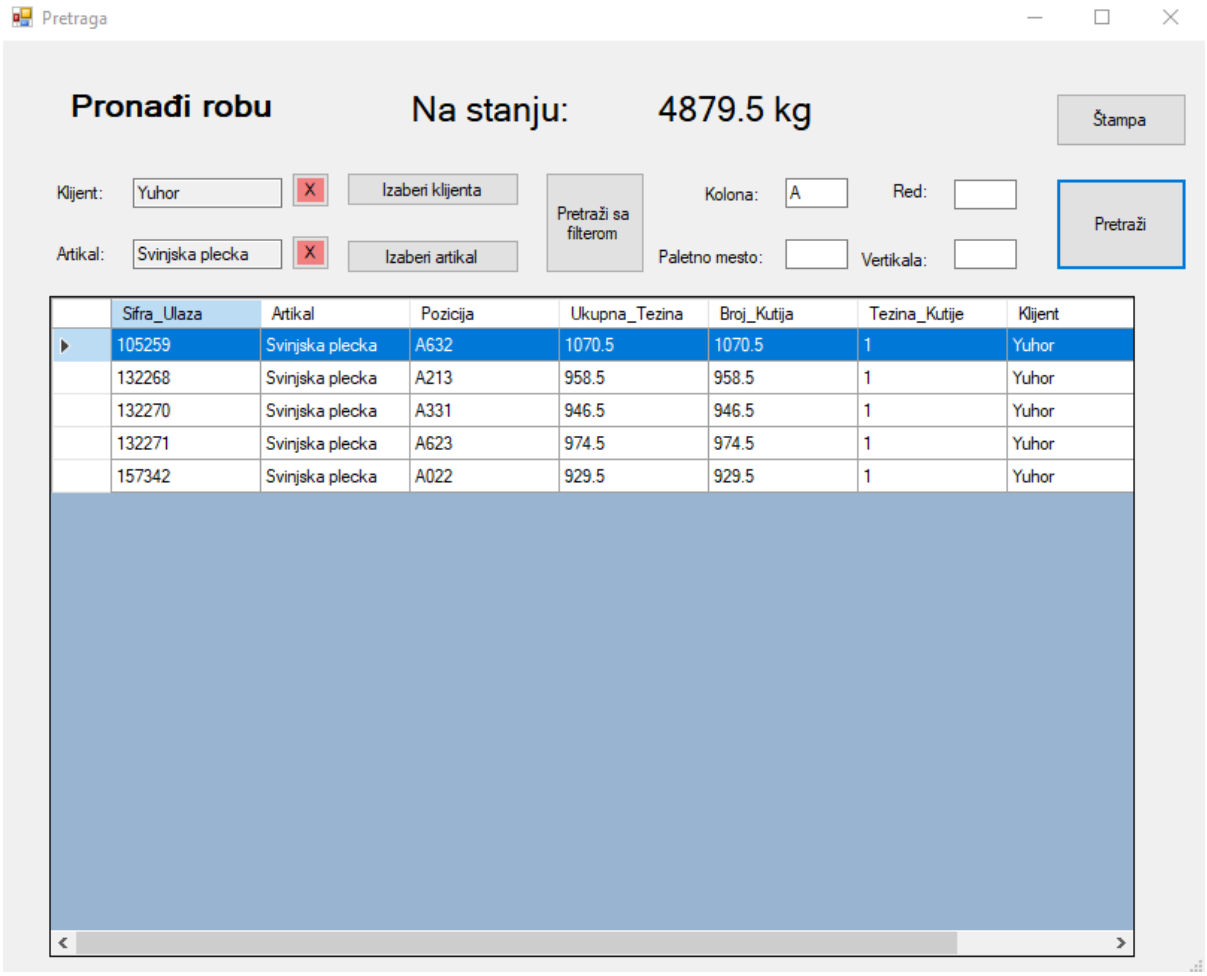
Основни сценарио СК

1. Магационер **уноси** критеријум по којој претражује робу. (АПУСО)



Слика 130 - СК3: Претраживање робе - Основни сценарио - одабир клијента, робе и позиције

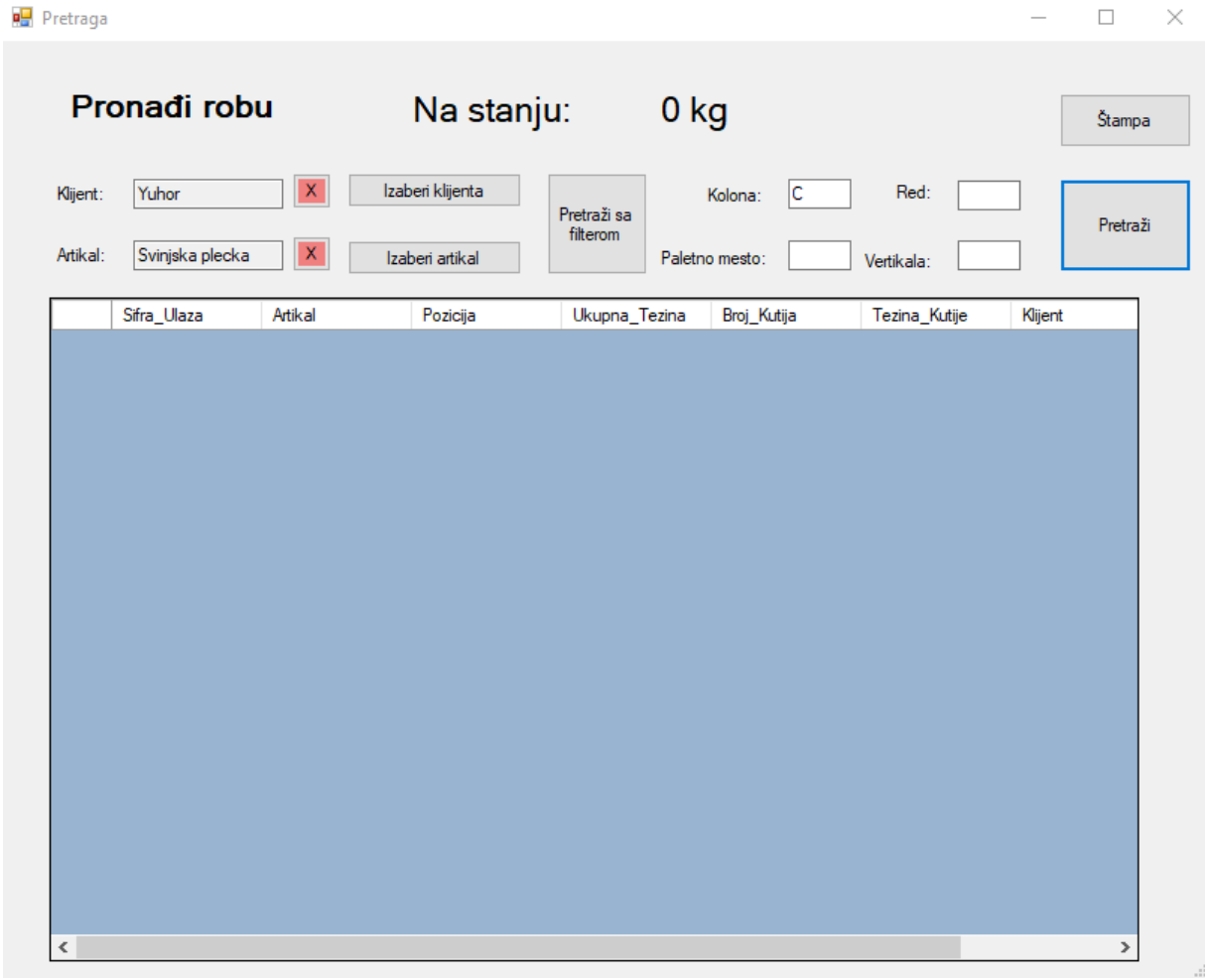
2. Магационер **позива** систем да нађе робу по задатом критеријуму. (АПСО)
3. Систем **тражи** робу по задатом критеријуму. (СО)
4. Систем приказује магационеру податке о роби коју је понашао. (ИА)



Слика 131 - СК3: Претраживање робе - Основни сценарио - претражена роба

Алтернативни сценарио:

1.1 Уколико систем не може да нађе робу он приказује празну табелу.(ИА).



Слика 132 - СК3: Претраживање робе - Алтернативни сценарио

13.4.4 СК4: Излаз целе палете

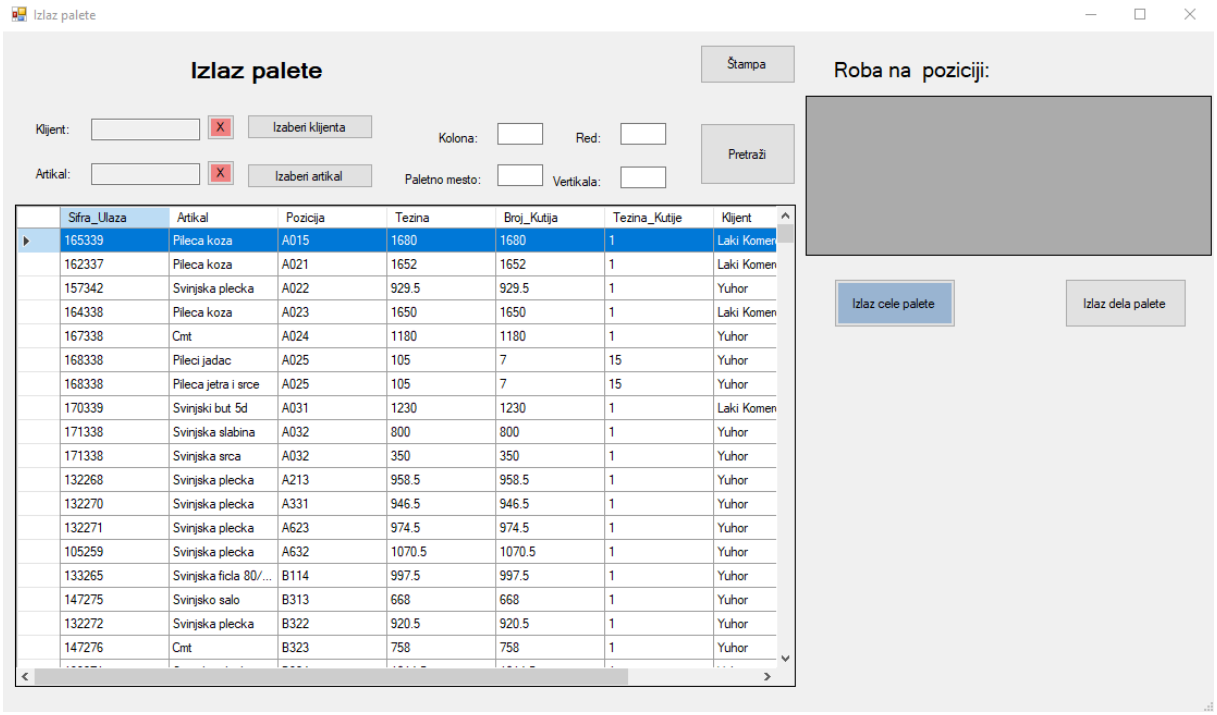
Назив СК
Промена палете (улаза)

Актори СК
Магационер

Учесници СК
Магационер и систем (програм)

Предуслов: Систем је укључен и Магационер је пријављен под својом шифром. Систем приказује форму за рад са излазом целе палете. Учитана је листа свих палета са својим позицијама које се налазе у систему.

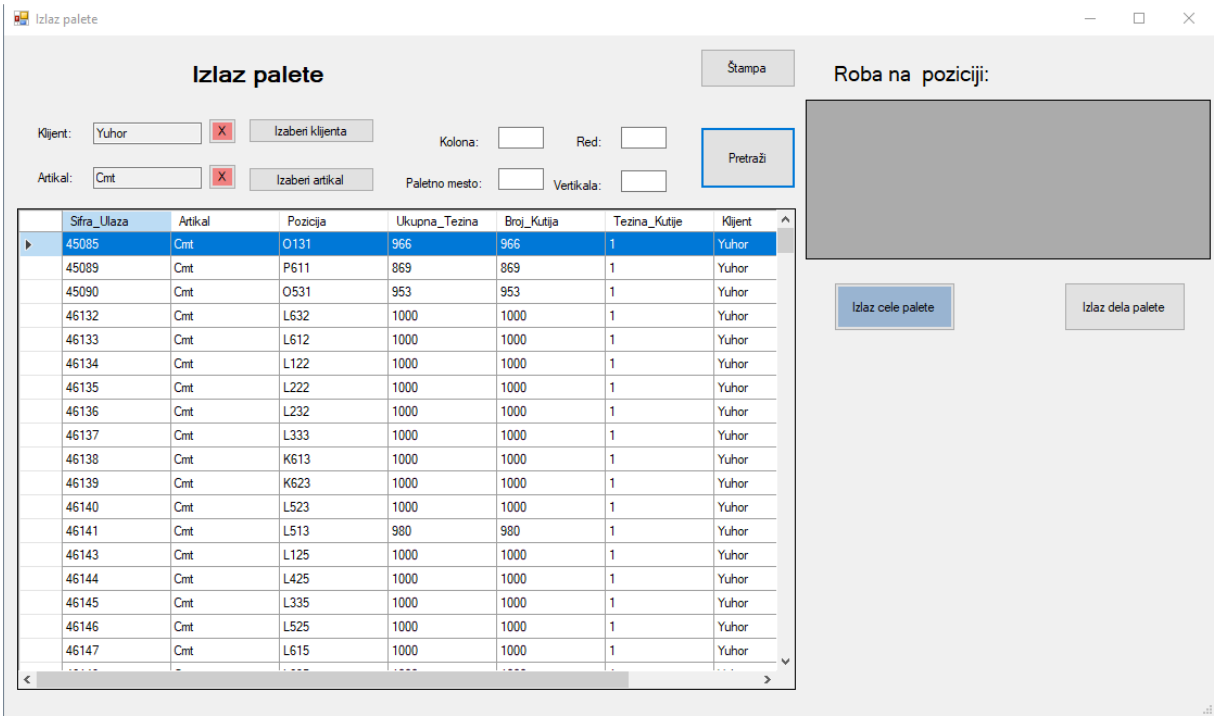
Систем приказује форму за излаз целе палете.



Слика 133 - СК4: Излаз целе палете

Основни сценарио СК

1. Магационер **уноси** вредност по којој претражује целу излазну палету. (АПУСО)

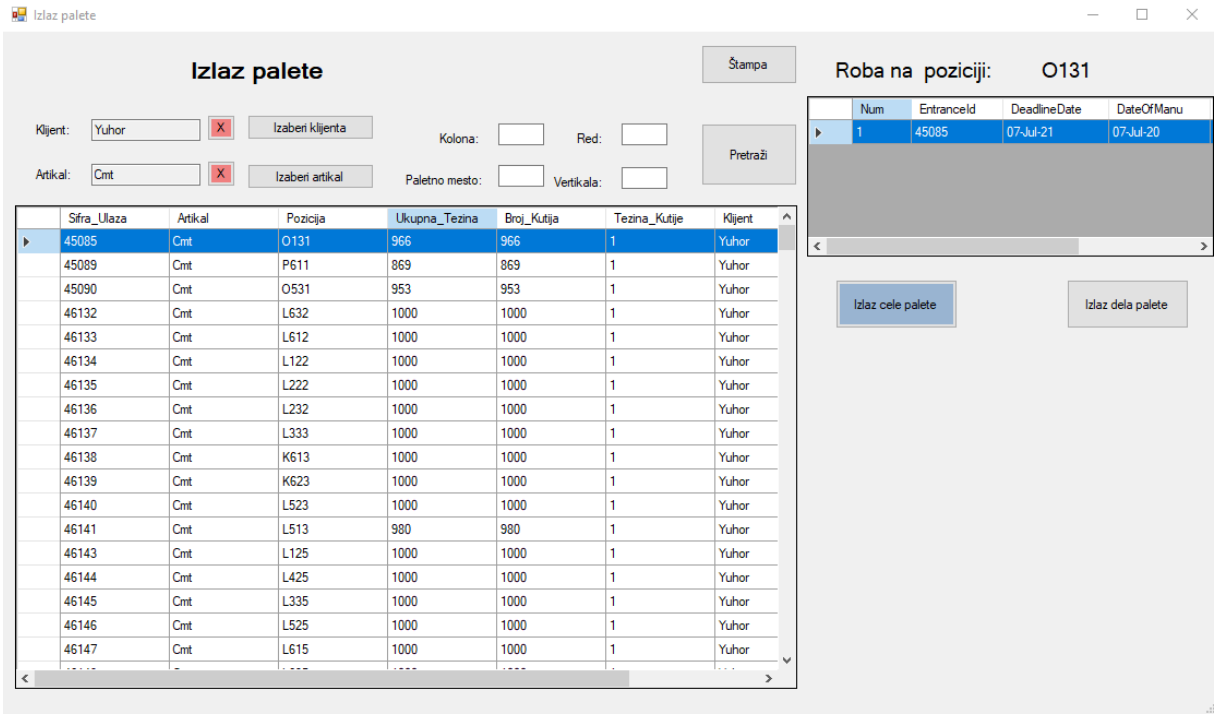


Слика 134- СК4: Излаз целе палете - Основни сценарио - претрага робе

2. Магационер **позива** систем да нађе целу излазну палету по задатој вредности. (АПСО)

3. Систем **тражи** целу излазну палету по задатој вредности. (СО)

4. Систем приказује Магационеру целу излазну палету. (ИА)



Sifra_Ulaza	Artikal	Pozicija	Ukupna_Tezina	Broj_Kutija	Tezina_Kutije	Klijent
45085	Cmt	O131	966	966	1	Yuhor
45089	Cmt	P611	869	869	1	Yuhor
45090	Cmt	O531	953	953	1	Yuhor
46132	Cmt	L632	1000	1000	1	Yuhor
46133	Cmt	L612	1000	1000	1	Yuhor
46134	Cmt	L122	1000	1000	1	Yuhor
46135	Cmt	L222	1000	1000	1	Yuhor
46136	Cmt	L232	1000	1000	1	Yuhor
46137	Cmt	L333	1000	1000	1	Yuhor
46138	Cmt	K613	1000	1000	1	Yuhor
46139	Cmt	K623	1000	1000	1	Yuhor
46140	Cmt	L523	1000	1000	1	Yuhor
46141	Cmt	L513	980	980	1	Yuhor
46143	Cmt	L125	1000	1000	1	Yuhor
46144	Cmt	L425	1000	1000	1	Yuhor
46145	Cmt	L335	1000	1000	1	Yuhor
46146	Cmt	L525	1000	1000	1	Yuhor
46147	Cmt	L615	1000	1000	1	Yuhor

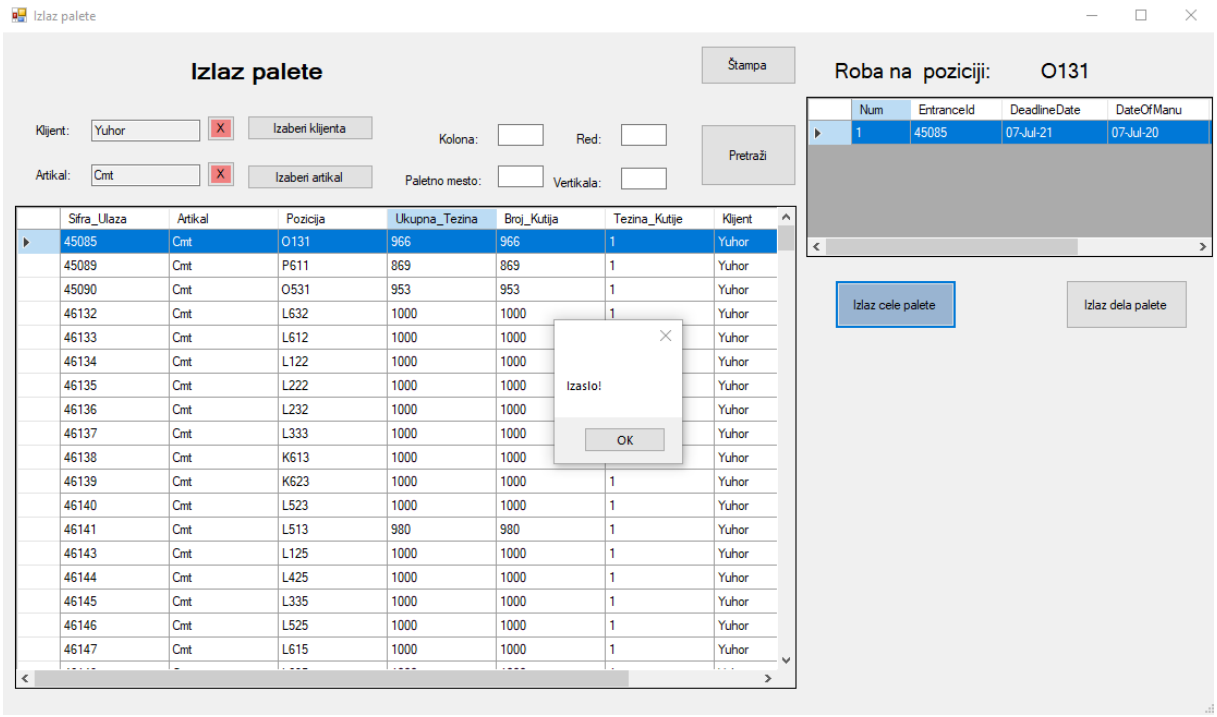
Слика 135 - СК4: Излаз целе палете - Основни сценарио – приказ излазне палете

5. Магационер **контролише** да ли је коректно унео податке о излазној палети. (АНСО)

6. Магационер **позива** систем да запамти податке о излазној палети. (АПСО)

7. Систем **памти** податке о излазној палети. (СО)

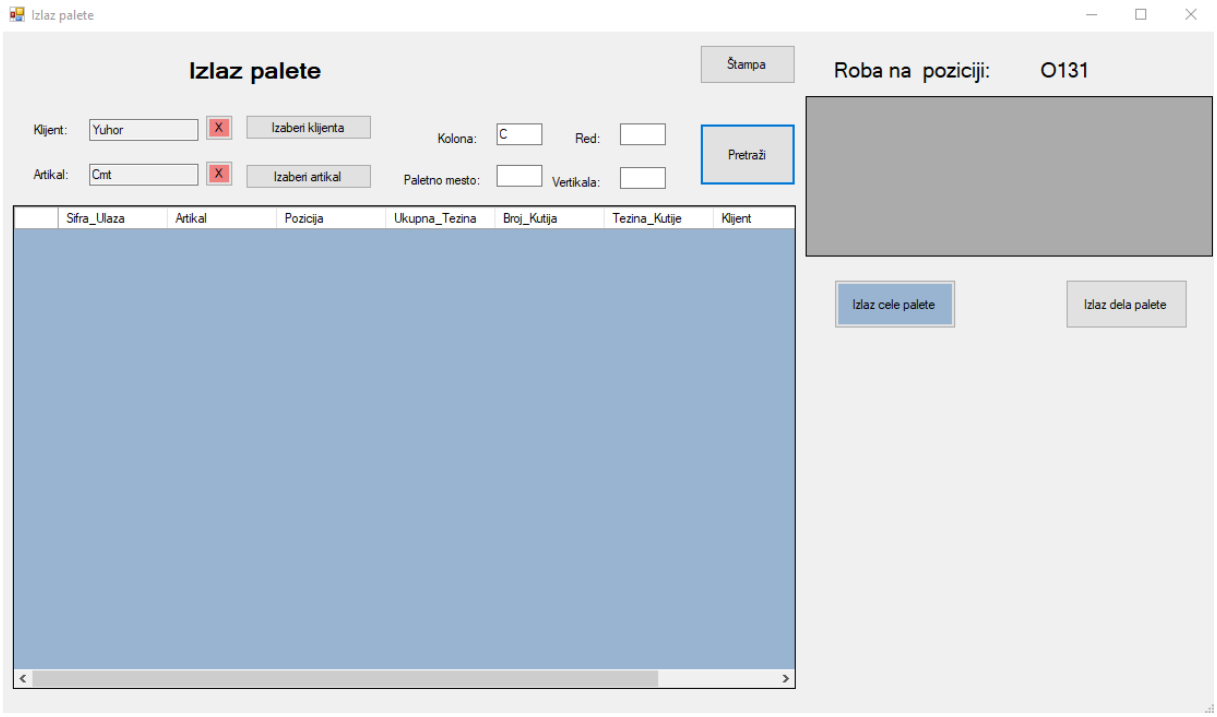
8. Систем **приказује** Магационеру запамћени излаз целе палете и поруку: “Изашло” (ИА)



Слика 136 - СК4: Излаз целе палете - Основни сценарио - излаз целе палете

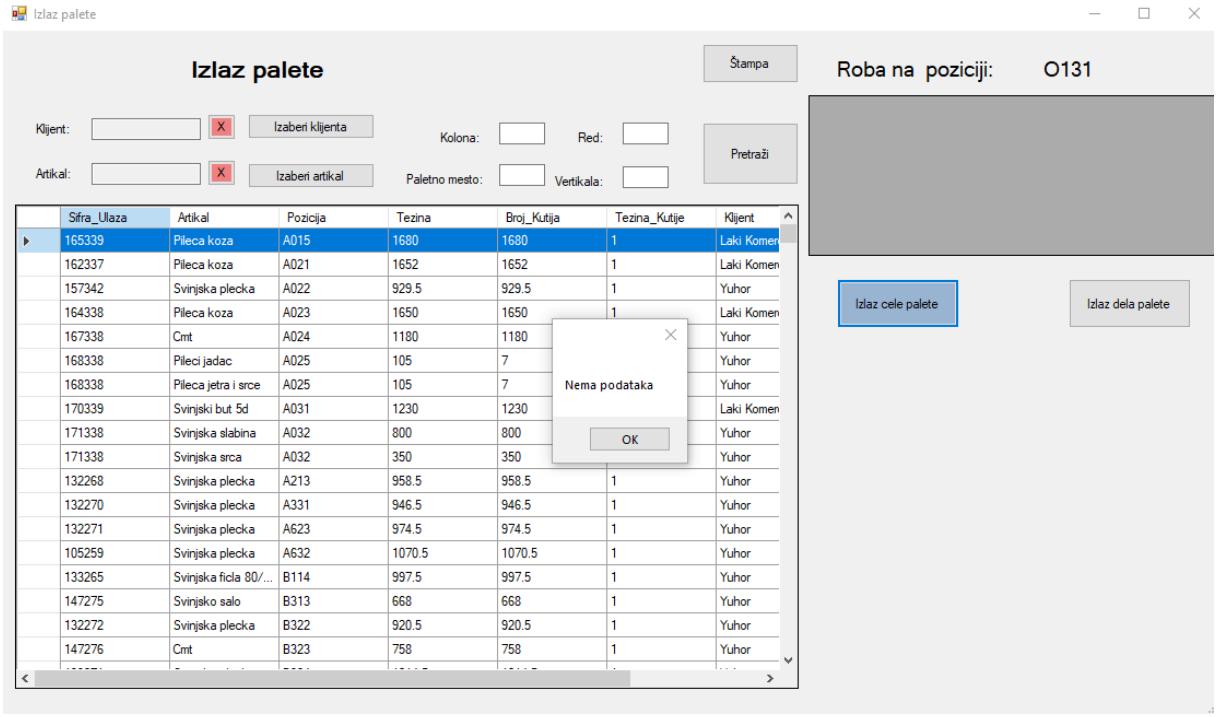
Алтернативни сценарио:

1.2 Уколико систем не може да нађе целу излазну палету, приказаће празну табелу.Прекида се извршење сценариа. (ИА)



Слика 137 - СК4: Излаз целе палете - Алтернативни сценарио - лоша претрага

7.1 Уколико систем не може да запамти податке о излазној палети он приказује Магационеру поруку “Нема података”.



Слика 138 - СК4: Излаз целе палете - Алтернативни сценарио - нема података

13.4.5 СК5: Излаз дела палете

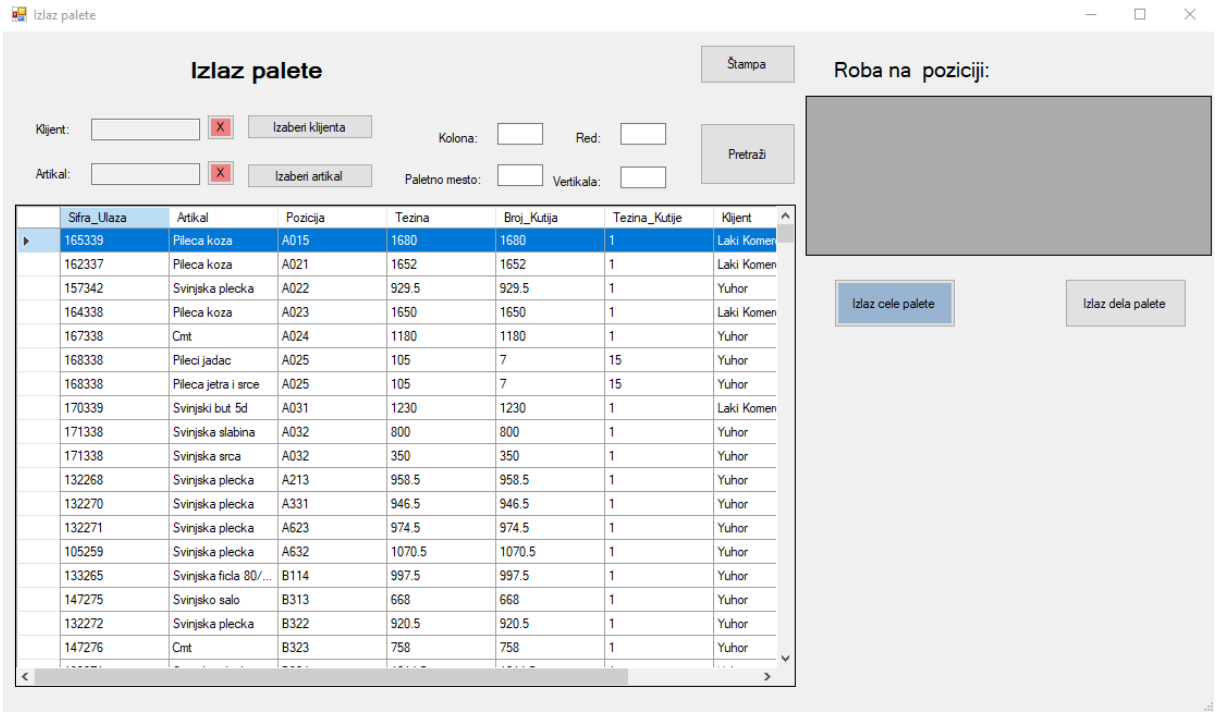
Назив СК
Креирање излазне палете

Актори СК
Магационер

Учесници СК
Магационер и систем (програм)

Предуслов: Систем је укључен и Магационер је пријављен под својом шифром. Систем приказује форму за рад са излазом дела палете. Учитана је листа свих ставки улаза у складишту.

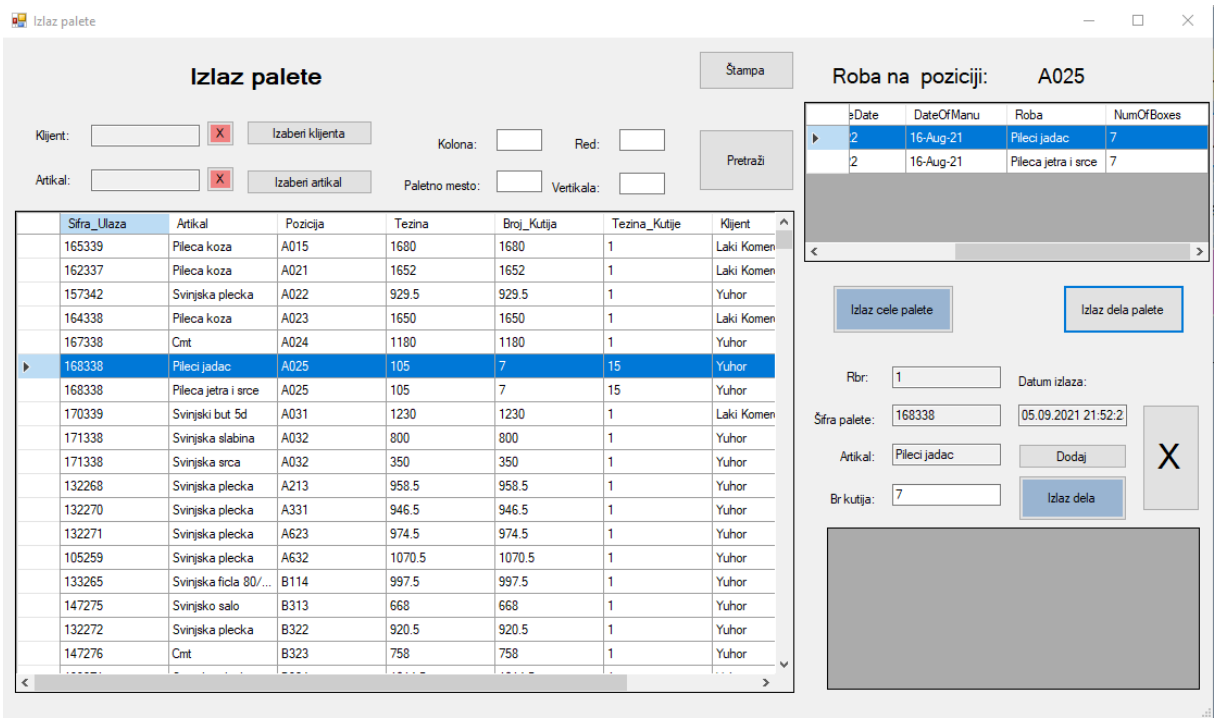
Систем приказује форму за излаз целе (дела) палете.



Слика 139 - СК5: Излаз дела палете

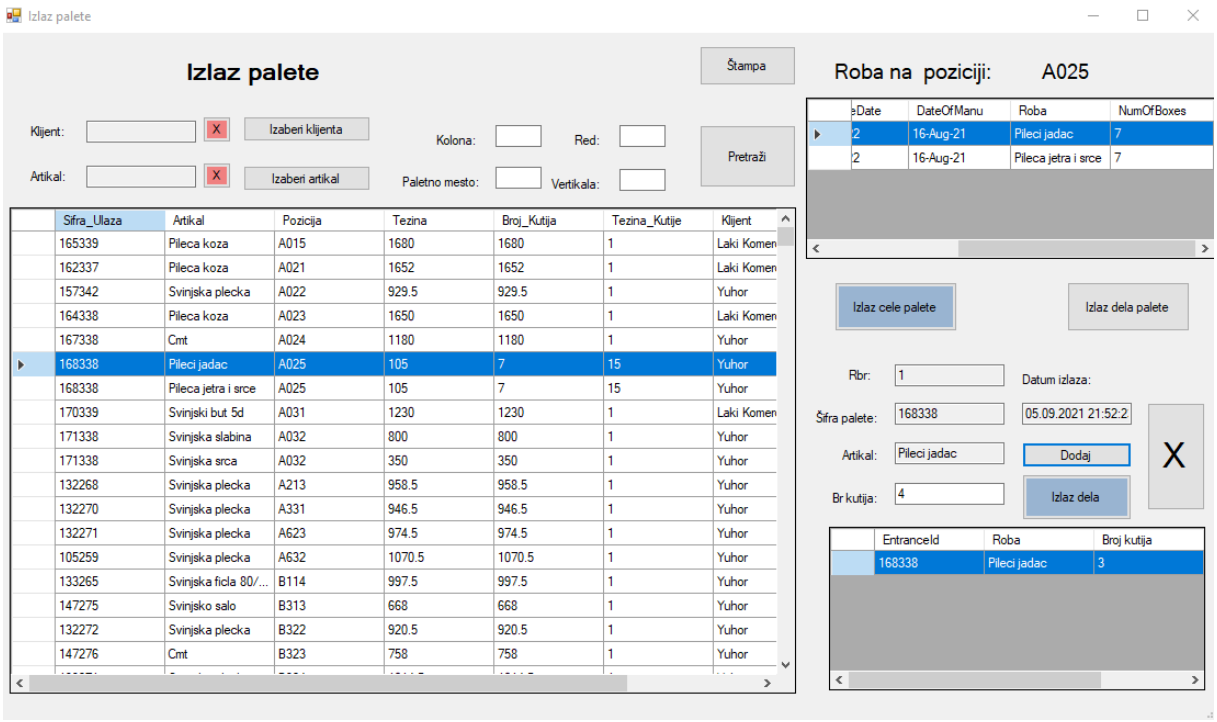
Основни сценарио СК

1. Магационер **позива** систем да креира део излазне палете. (АПСО)
2. Систем **креира** део излазне палете. (СО)
3. Систем **приказује** Магационеру део излазне палете (ИА)



Слика 140 - СК5: Излаз дела палете - Основни сценарио - приказ излаза дела палете

4. Магационер **уноси** податке за део излазне палете. (АПУСО)



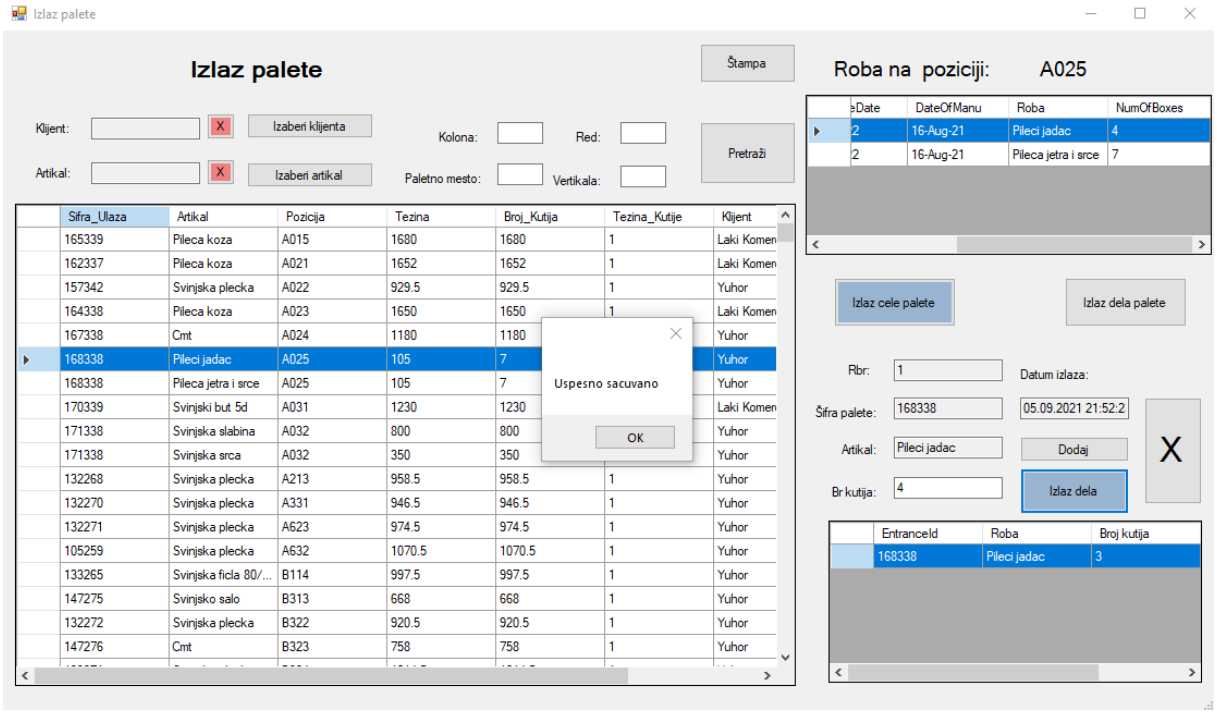
Слика 141 - СК5: Излаз дела палете - Основни сценарио - унос излаза дела палете

5. Магационер **контролише** да ли је коректно унео податке у део излазне палете. (АНСО)

6. Магационер **позива** систем да запамти податке о излазној палети. (АПСО)

7. Систем **памти** податке о излазној палети. (СО)

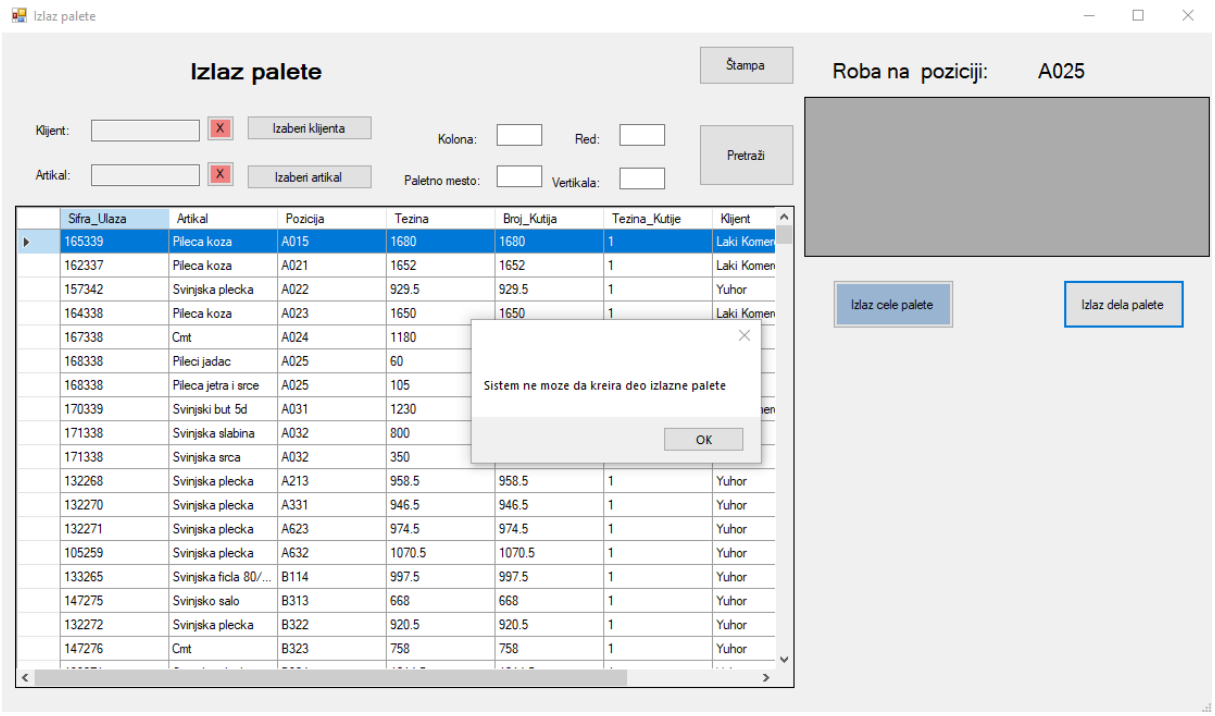
8. Систем **приказује** Магационеру запамћени излаз дела палете и поруку: “Успешно сачувано“. (ИА)



Слика 142 - СК5: Излаз дела палете - Основни сценарио

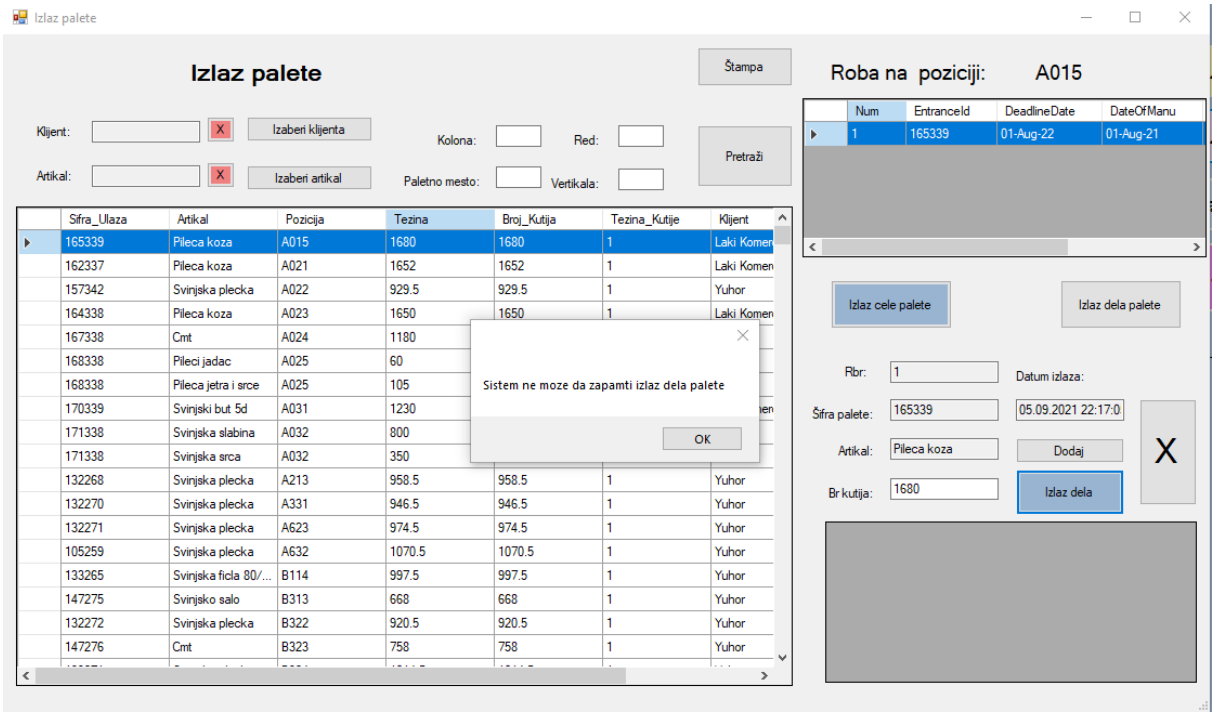
Алтернативни сценарио:

3.1 Уколико систем не може да креира део излазне палете он приказује Магационеру поруку: “Систем не може да креира део излазне палете”. Прекида се извршење сценариа. (ИА)



Слика 143 - СК5: Излаз дела палете - Алтернативни сценарио 1

8.1 Уколико систем не може да запамти податке о излазној палети он приказује Магационеру поруку “Систем не може да запамти излаз дела палете”. (ИА)



Слика 144 - СК5: Излаз дела палете - Алтернативни сценарио 2

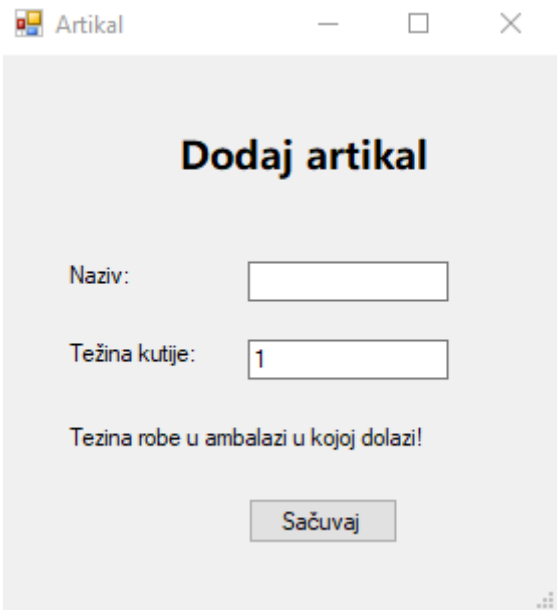
13.4.6 СК6: Креирање робе

Назив СК
Креирање робе

Актори СК
Магационер

Учесници СК
Магационер и систем (програм)

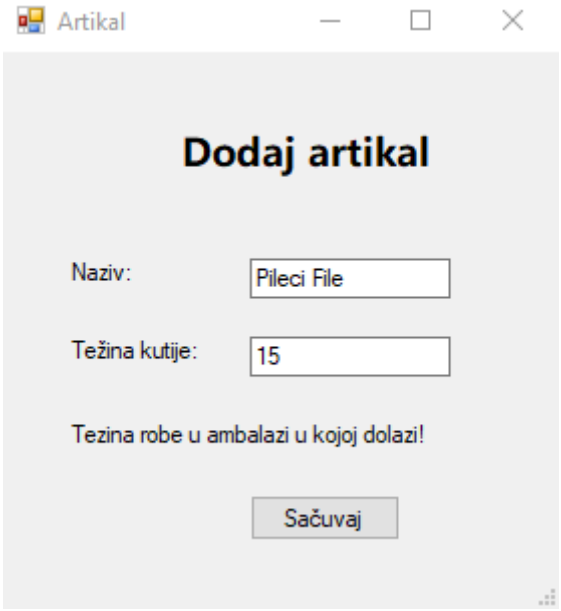
Предуслов: Систем је укључен и Магационер је пријављен под својом шифром. Систем приказује форму за рад са робом.



Слика 145 – СК6: Креирање робе - почетна форма

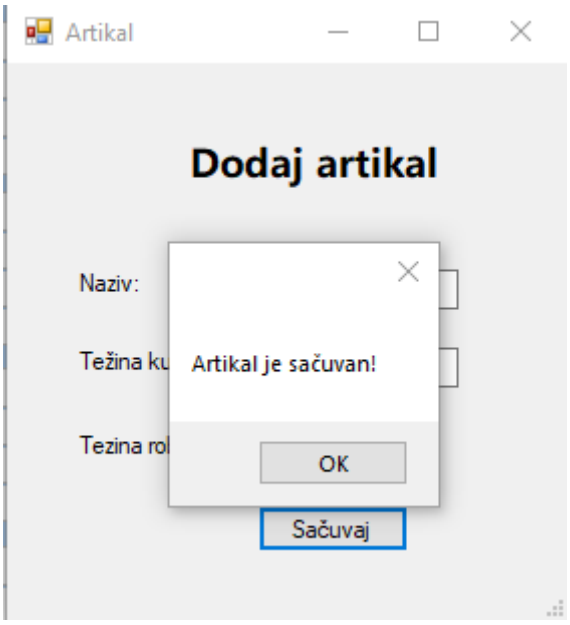
Основни сценарио СК:

1. Магационер **уноси** робу. (АПУСО)



Слика 146 – СК6: Креирање робе - Основни сценарио - унос података

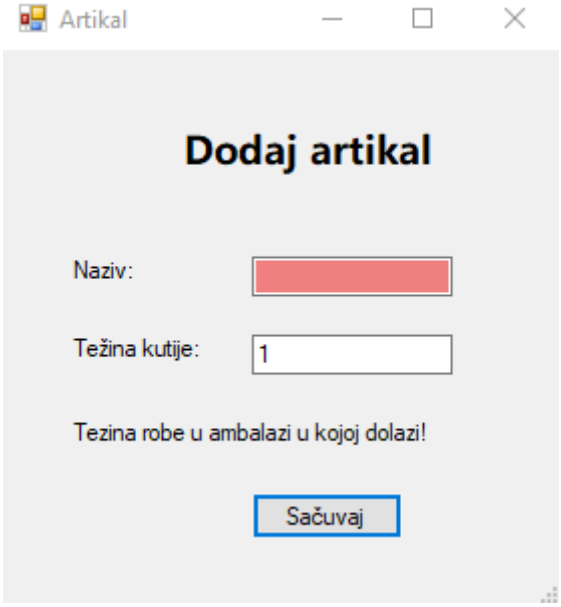
2. Магационер **контролише** да ли је коректно унео робу. (АНСО)
3. Магационер **позива** систем да запамти робу. (АПСО)
4. Систем **памти** робу. (СО)
5. Систем **приказује** Магационеру запамћени податак и поруку: "Артикал је сачуван". (ИА)



Слика 147 - СК7: Креирање робе - Основни сенарио - податак је сачуван

Алтернативна сценариа:

5.2 Уколико систем не може да запамти податке о роби. Обојиће поље за унос црвеном бојом. (ИА)



Слика 148 - СК7: Креирање податка - Алтернативни сценарио

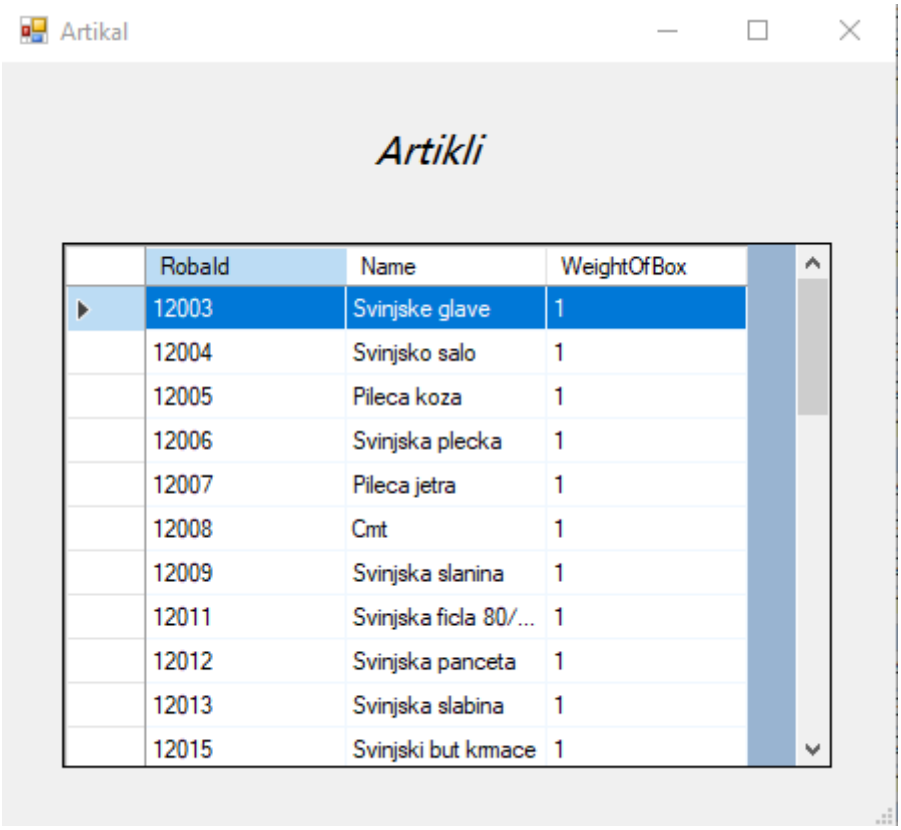
13.4.7 СК7: Промена робе

Назив СК
Промена робе

Актори СК
Магационер

Учесници СК
Магационер и систем (програм)

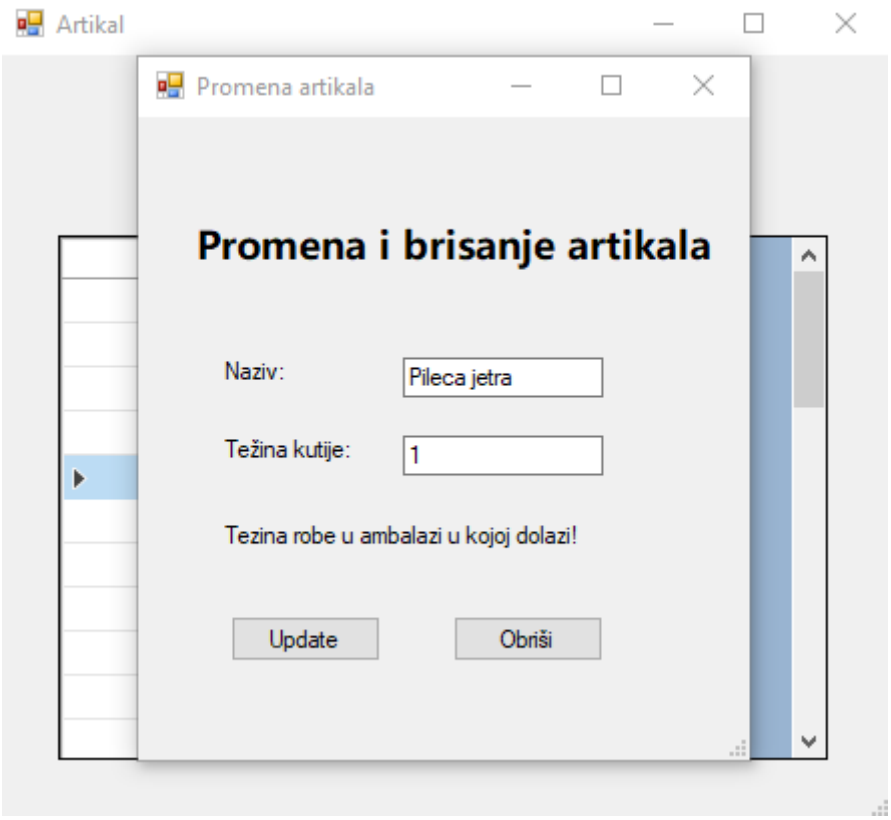
Предуслов: Систем је укључен и Магационер је пријављен под својом шифром. Систем приказује форму за рад са робом. Учитана је листа Робе.



Слика 149 - СК6: Промена robe

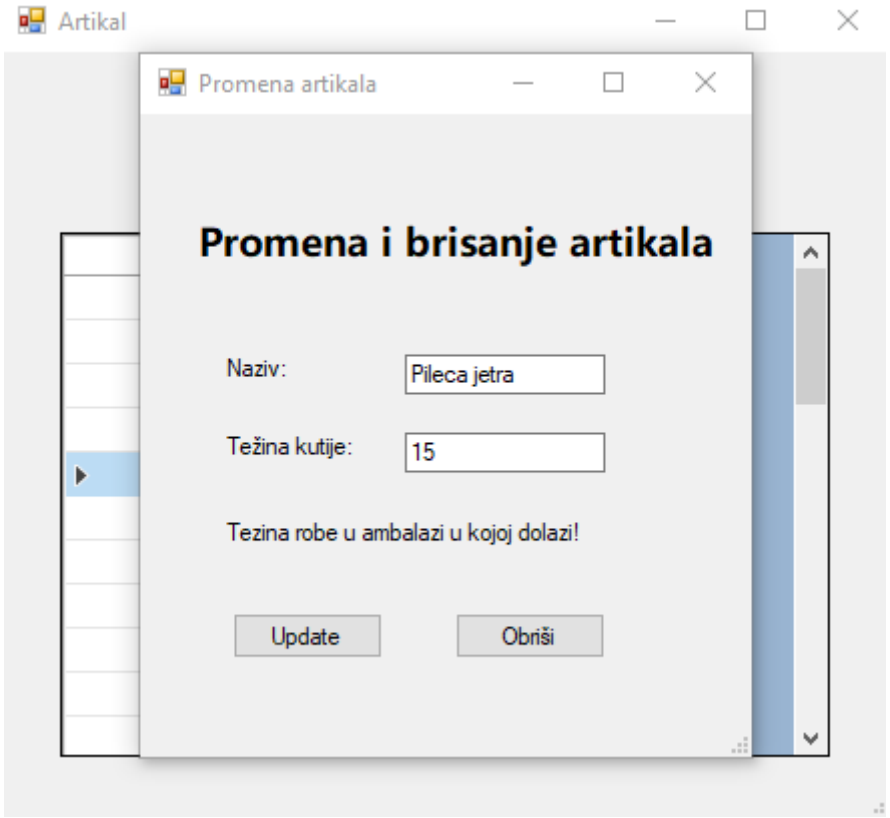
Основни сценарио СК

1. Магационер **позива** систем да нађе робу по одабраној вредности. (АПСО)
2. Систем **тражи** робу по задатој вредности. (СО)
3. Систем приказује Магационеру робу. (ИА)



Слика 150 – СК7: Промена robe- Основни сценарио - приказ података

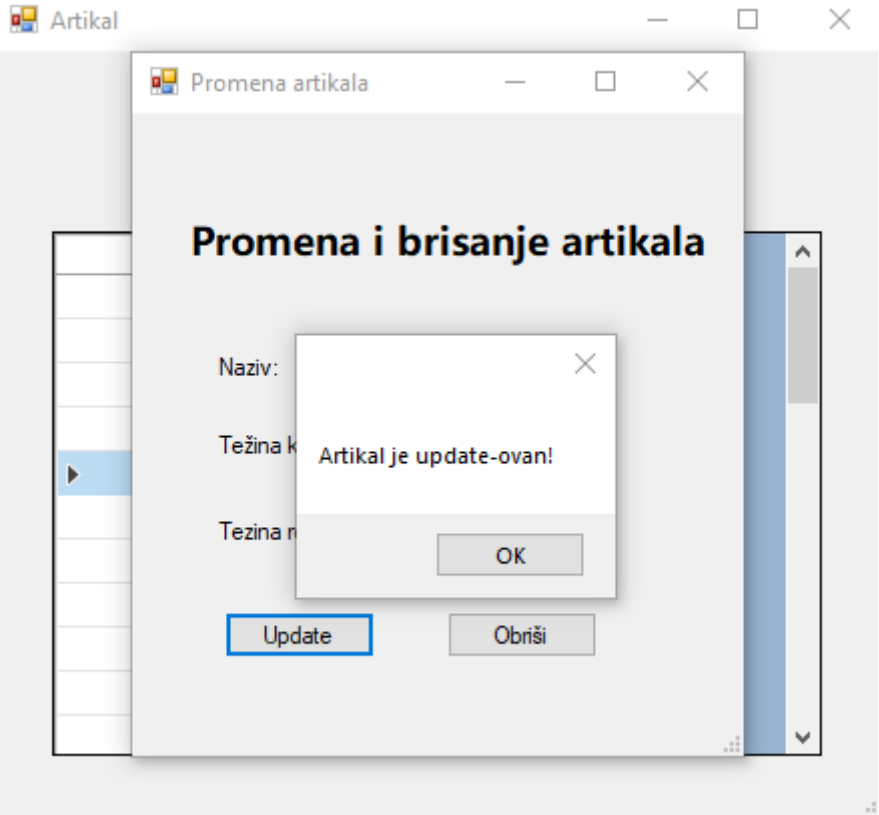
4. Магационер **уноси (мења)** робу. (АПУСО)



Слика 151 – СК7: Промена robe - Основни сценарио - унос нове вредности

5. Магационер **контролише** да ли је коректно унео робу. (АНСО)

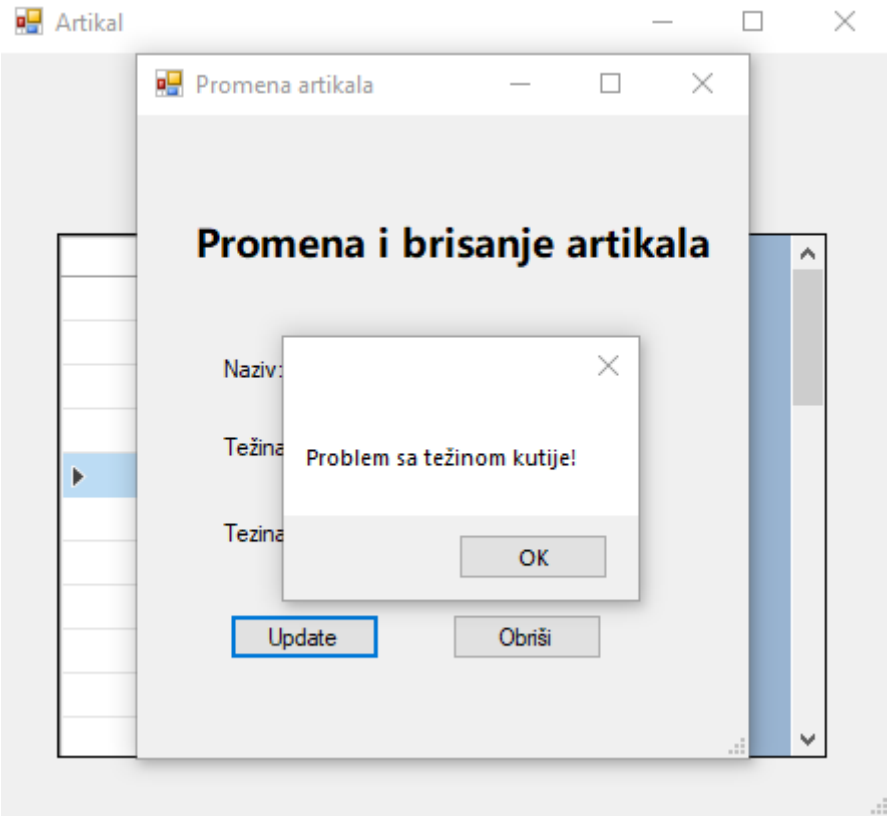
6. Магационер **позива** систем да запамти промену робе. (АПСО)
7. Систем **памти** робу. (СО)
8. Систем **приказује** Магационеру запамћену промену и поруку: “Артикал је ажуриран!.” (ИА)



Слика 152 - СК6: Промена података - Основни сценарио - успешно ажурирано

Алтернативни сценарио:

8.1 Уколико систем не може да запамти робу он приказује Магационеру поруку. Прекида се извршење сценариа. (ИА)



Слика 153 - СК6: Промена робе - Алтернативни сценарио - лоше унесени подаци

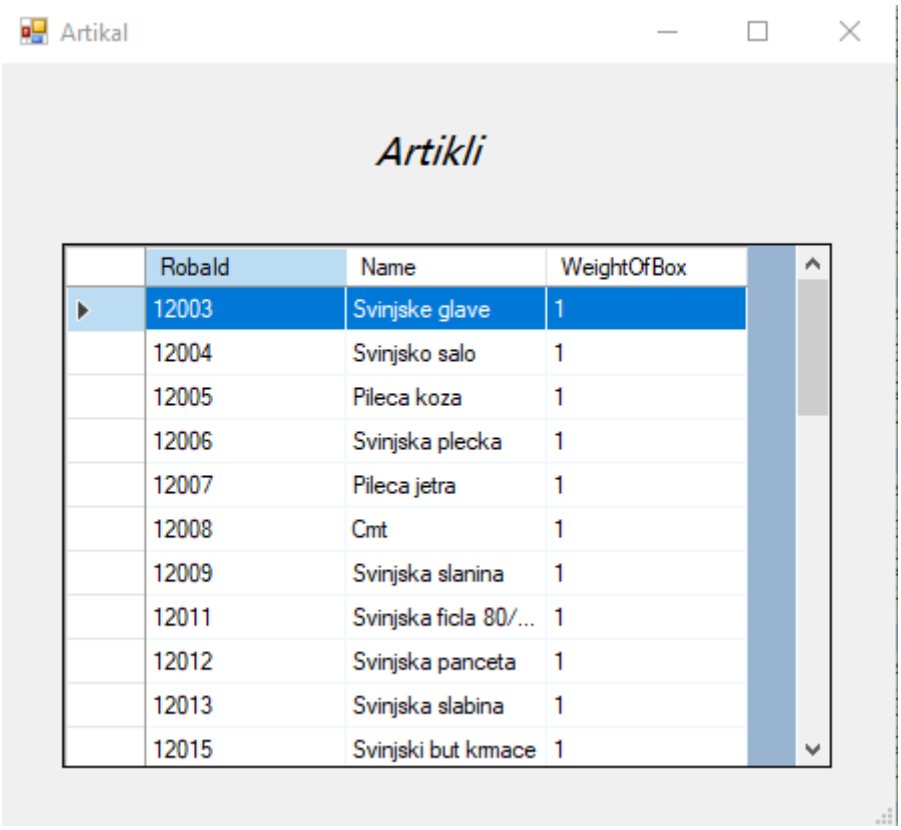
13.4.8 СК8: Брисање робе

Назив СК
Брисање робе

Актори СК
Магационер

Учесници СК
Магационер и систем (програм)

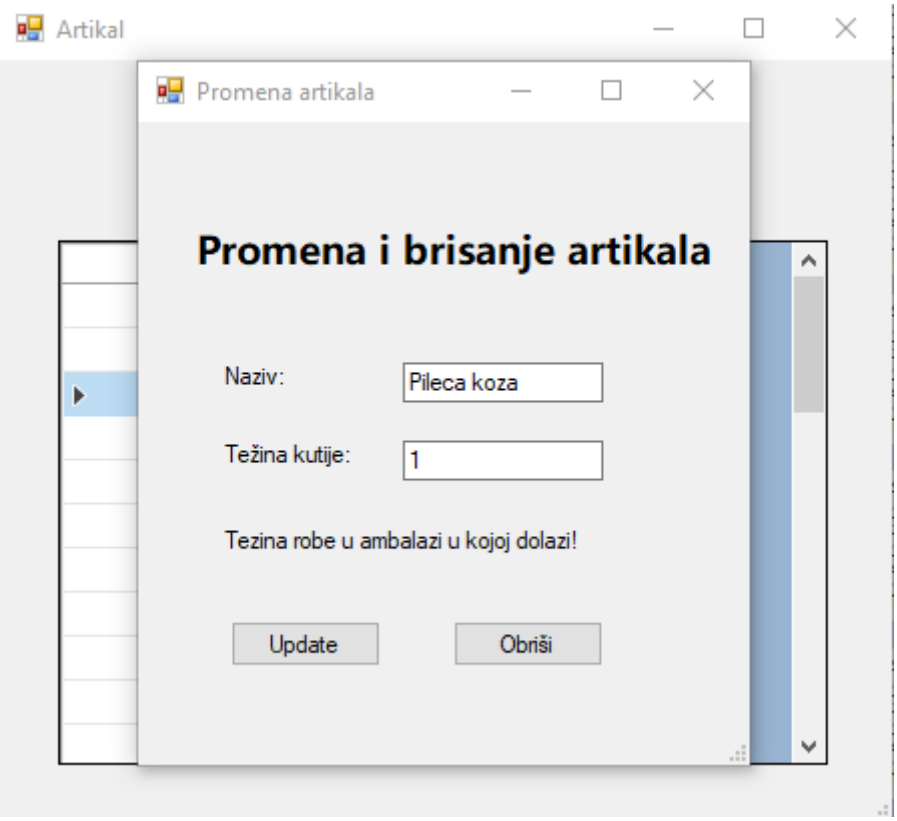
Предуслов: Систем је укључен и Магационер је пријављен под својом шифром. Систем приказује форму за рад са робом.



Слика 154 - СК8: Брисање робе - почетна форма

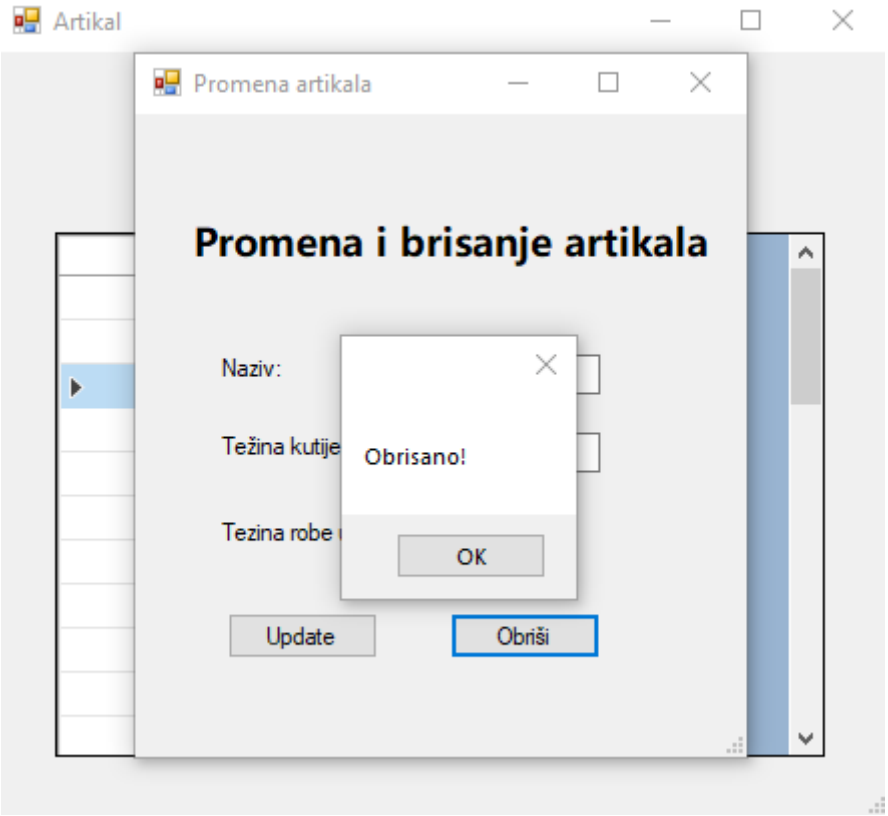
Основни сценарио СК:

1. Магационер **уноси** вредност по којој претражује робу. (АПУСО)
2. Магационер **позива** систем да нађе робу по задатој вредности. (АПСО)
3. Систем **тражи** робу по задатој вредности. (СО)
4. Систем приказује Магационеру робу. (ИА)



Слика 155 - СК8: Брисање robe - Основни сценарио - одабир података за брисање

5. Магационер **позива** систем да обрише робу. (АПСО)
6. Систем **брише** робу. (СО)
7. Систем **приказује** Магационеру поруку: “Обрисано.” (ИА)



Слика 156 - СК8: Брисање робе - Основни сценарио - брисање

14.Фаза тестирања

Софтверски систем који је описан у овоме раду је прошао све тестове које би обезбедили пуштање програма у продукцију. Тестови су обухватили мануелан унос током самог развитка софтверског система, као и тестни период продукције од два месеца. Исправљене су све могуће ситне грешке захтеване од стране корисника система, као и визуелни захтеви самог програма. Свака потенцијална грешка је обрађена на начин да омогући сваком кориснику јасно и недвосмислено објашњење о самом проблему. У случајевима погрешног формата, корисник је упозорен да је потребно да се прилагоди одређеном формату. Датуми који се користе у самом систему зависе од конфигурације клијентског рачунара и његовог оперативног система.

Фазу тестирања је потребно стално спроводити, приликом ажурирања постојеће верзије па до саме крајње, како би обезбедили безбедност софтверског система и самих података.

Сви захтеви корисника су испуњени и функционални на највишем нивоу, постоји простора за унапређење система за рад са ПДА уређајима (скенерима) који би олакшао и убрзао сам улаз/излаз палета кроз складиште.

15. Закључак

Тема овог завршног рада јесте *Развој софтверског система евиденцију робних палета у регалним складиштима .NET технологија* која реализована кроз две целине. Прва целина се односи на теоријски део о свим коришћеним технологијама током израде поменутог рада. Док друга целина представља практичан рад који је описан комплетном документацијом.

Основу овог софтверског система представља Ларманова метода и клијент-сервер архитектура која је намењена за мрежну комуникацију. Остале технологије које су коришћене за реализацију завршног рада можемо поделити на серверску и клијентску страну. Серверска страна служи за успостављање јединственог тока комуникације са клијентом и врши обраду свих његових захтева. Сервер комуницира са базом података путем брокера и податке које добија заузврат прослеђује клијенту. Клијентска страна представља *desktop* апликацију која садржи графички интерфејс у виду *windows* форме који је прилагођен и интуитиван будућим корисницима. У раду су коришћени основни концепти програмског језика C#. Коришћење упрошћене Ларманове методе доноси пет фаза: прикупљање захтева, анализу, пројектовање, имплементацију и тестирање. За сваку фазу треба издвојити време и сагледати ширу слику како у наредним фазама не би долазило до несугласица. Све фазе су засебно јако битне за даље развијање софтверског система.

Коришћењем .NET Framework-а, програмског језика C#, клијент-сервер архитектуре, софтверских патерна и Microsoft SQL Server-а унапредио сам своје знање и искуство. Створен је документован и функционалан софтверски систем који се може применити. Унапређење овог софтверског система јесте веб и мобилна апликација која ће бити доступна у сваком тренутку сваком кориснику који ради са складиштима.

16. Литература

[1] Aggarwal, A. (2020, June 5). *Common Language Runtime (CLR) in C#.* Retrieved from www.geeksforgeeks.org/: https://www.geeksforgeeks.org/common-language-runtime-clr-in-c-sharp/

[2] Dr Vlajic, S. (2015). *Projektovanje softvera (skripta), FON, Beograd, 2015*

[3] Kampffmeyer, H., Yschaler, S. (2007). *Finding the Pattern You Need: The Design Intent Ontology, in Models, Springer-Verlag, volume 4735/2007, 211-225*

[4] Alexander, C. (1974). Notes on the Synthesis of Form, Cambridge, MA: Harvard University Press, 1974

[5] Tomasevic, V. (2012), Razvoj aplikativnog softvera, Beograd, 2012

[6] Melton, J., Simon, A. (2002). *SQL:1999, Understanding Relational Language Components, Academic Press, 2002.*

[7] Densmore, S. (2004, May 25). *Why Singletons are Evil.* Retrieved from docs.microsoft: https://docs.microsoft.com/en-us/archive/blogs/scottdensmore/why-singletons-are-evil

[8] Garg, V. (2019, May 8). *Singleton Design Pattern.* Retrieved from geeksforgeeks: https://www.geeksforgeeks.org/singleton-design-pattern-introduction/

[9] Kurose J., Ross, K. (2017), Computer Networking: A Top-Down Approach, University of Massachuetes, Amherst, University of New York, 2017

[10] Kaur, H. (2019, May 1). *.NET Framework Class Library (FCL).* Retrieved from www.geeksforgeeks.org/: https://www.geeksforgeeks.org/net-framework-class-library-fcl/

[11] Kleiman, S., Shah, D., & Smaalders, B. . (1996). *Programming with threads.* Mountain View: Sun Soft Press.

[12] Landwerth, I. (2017, September). *.NET Standard.* Retrieved from https://docs.microsoft.com/: https://docs.microsoft.com/en-us/archive/msdn-magazine/2017/september/net-standard-demystifying-net-core-and-net-standard

[13] Troelsen, A. (2001). *C# and the. NET Platform.* Apress.

[14] Vidovic, A. (2017, April 4). *Uvod u MVC dizajn patern.* Retrieved from startit: https://startit.rs/dizajn-paterni-mvc/