Универзитет у Крагујевцу – Факултет инжењерских наука

Смер: Рачунарска техника и софтверско инжењерство

Дипломски рад

Развој информационог система за виртуелну галерију акварела

Студент:
Јаковљевић Софија 574/2015

Ментор:
Проф. Др Милан Ерић

Крагујевац, октобар 2019. године

Универзитет у Крагујевцу
Факултет инжењерских наука

Назив студијског програма: Рачунарска техника и софтверско инжењерство

Ниво студија: Основне академске студије

Предмет: Пројектовање информационих система и база података

Име и презиме: Софија Јаковљевић

Број индекса: 574/2015

Развој информационог система за виртуелну галерију акварела

Комисија за преглед и одбрану

1. Проф. Др Милан Ерић
2.
3.

Датум одбране:

Оцена:

Изјава о самосталној изради рада

Изјављујем да сам овај дипломски рад урадила самостално, служећи се стеченим знањем и искуством као и наведеном литературом.

Јаковљевић Софија, 574/2015

_____

Универзитет у Крагујевцу
Факултет инжењерских наука

Основне студије: Рачунарска техника и софтверско инжењерство

Име и презиме: Софија Јаковљевић

Број индекса: 574/2015

ПРИЈАВА ЗАВРШНОГ (ДИПЛОМСКОГ) РАДА

Тема рада: **Развој информационог система за виртуелну галерију акварела**

Задатак: У оквиру овог завршног рада кандидат треба да дефинише технологије које ће користити, као и архитектуру апликације, опис реалног система, опис развојног модела и развојног софтвера. Кандидат треба да развије апликацију за развој информационих система виртуелне галерије акварела. Потребно је јасно дефинисати примену поменуте апликације. Закључним разматрањима описати и могућности имплементације.

Ментор:

Крагујевац, 16.04.2019

Проф. Др Милан Д. Ерић

Садржај

Увод

У оквиру дипломског рада „Развој информационог система за виртуелну галерију акварела“ реализована је апликација за управљање набавком и продајом акварела (уметничких слика). Апликација треба да помогне корисницима (запосленима) у њиховим радним задацима.

Услови које програм треба да испуни су следећи:

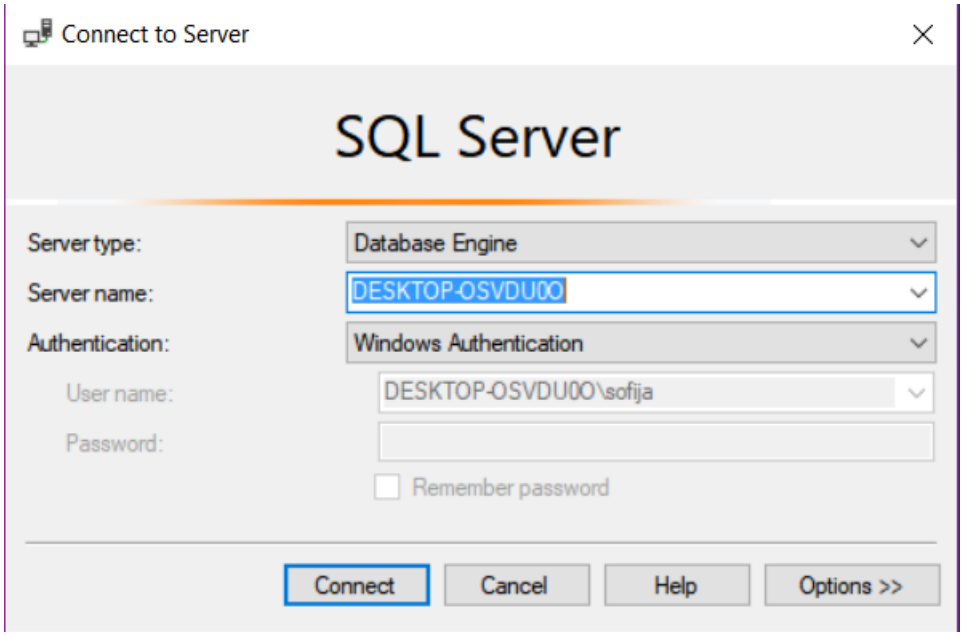
1. Приступ је омогућен само ауторизованим корисницима.
2. Ауторизовани корисници могу бити: администратор, комерцијалиста и радник.
3. Права администратора:
    1. Додавање новог акварела
    2. Уклањање акварела из продајног асортимана
    3. Промена атрибута постојећег акварела
    4. Претрага акварела по одређеним критеријумима
    5. Додељивање акцијских цена на одређене групе акварела
    6. Приказ акварела на екрану са сликама и могућношћу повезивања на одређене интернет странице са детаљним описима акварела
4. Права радника:
    1. Продаја акварела и генерисање рачуна у дигиталној форми
    2. Претрада акварела по одређеним критеријумима
    3. Наручивање акварела којих нема на стању
    4. Резервисање акварела и генерисање резервације у дигиталној форми
    5. Слање резервације путем мејла
    6. Приказ акварела на екрану са сликама и могућношћу повезивања на одређене интернет странице са детаљним описима акварела
5. Права комерцијалисте:
    1. Приступ листи наручених акварела
    2. Наручивање акварела од регистрованих добављача путем мејла
    3. Пријем робе и промена стања залиха
    4. Приказ фактура из архиве
    5. Промена цена акварела услед промене набавне цене

Апликација треба бити једноставна за коришћење тренутним корисницима као и потенцијалним будућим корисницима који ће доћи у контакт са њом. С обзиром да је приступ омогућен само ауторизованим корисницима, неопходно је имати налог ради приступа и рада на апликацији. Омогућена је манипулација производима (сликама) што дозвољава измене у било ком тренутку и обављање задатака над њима. Такође, ради лакшег информисања, корисници за сваки акварел из свог продајног асортимана, у сваком тренутку, имају приступ детаљним информацијама о сваком појединачном акварелу као и приказ фотографија. Сваки ауторизовани корисник, ради лакшег сналажења у апликацији, има могућност одабира помоћног алата (Помоћ), где може наћи детаљна упутства за манипулацију и коришћење апликације.

Коришћене технологије

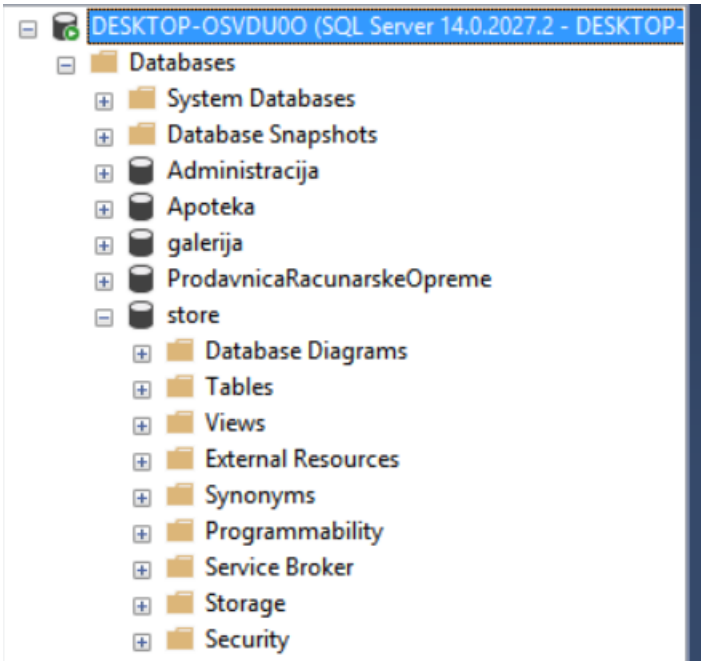
Апликација је реализована у MS Visual Studio 2017, док је база која се користи у оквиру апликације реализована у MS SQL 2017. Користи се C# програмски језик. Апликација треба да садржи форме преко којих се корисницима омогућава манипулација са сликама као и њихово излиставање са детаљним информацијама.

Miscrosoft SQL Server



Слика 1: Повезивање са сервером

Miscrosoft SQL Server је систем за управљање релационим базама података које развија Мајкрософт. Главна функција му је складиштење и враћање података које захтевају остале апликације (апликације могу бити на истом рачунару као сервер а могу бити и на удаљеном рачунару, комуницирајући посредством мреже).



Слика 2: Списак база унутар програма

Програмски језик C#

C# се појавио 2001. год. C# је елегантан, једноставан и потпуно објектно-оријентисан програмски језик, и представља врло добар избор за програмере. И у језику C#, као и у C/C++ или „Java“, могу се дефинисати промењљиве, могу се користити аритметички оператори, могу се дефинисати методе тј. функције и користити аргументи метода тј. функција. Такође, if-искази и while-петље су део језика C#. У C# се користе изузеци-exceptions, да би се процесирале грешке у програму. C# је потпуно објектно-оријентисан језик, где се могу користити сви принципи објектно-оријентисаног програмирања, нпр. наслеђе класа, сакривање података, конструктор-методе итд. Посебно је битно коришћење специфичне номенклатуре приликом именовања. Име треба да буде описно тј. да што више описује оно што представља. Уколико је за то потребно више од једне речи, по C# номенклатури, таква имена почињу малим словом, а свака нова реч великим. На пример: firstName, lastName, phoneNumber… Разлози за увођење номенклатуре су јасни: одржавање кода, рад у тиму итд. C# је Case sensitive, то јест препознаје мала и велика слова[2].

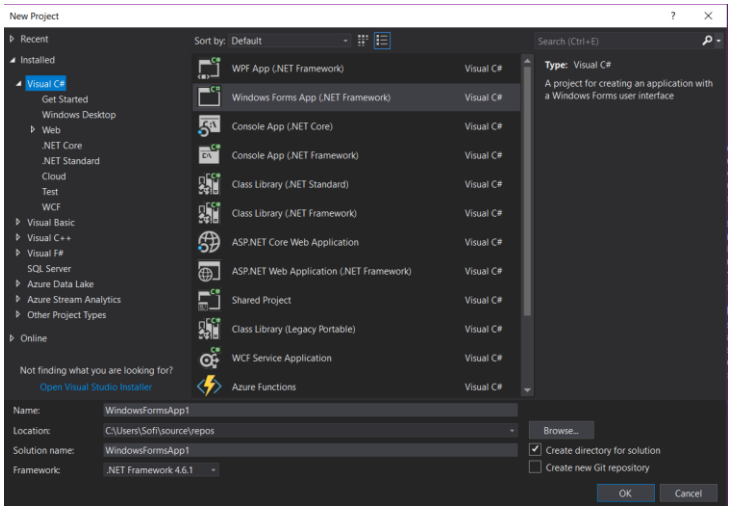
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace store.UI
{
    class Class1
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello World!");
        }
    }
}
```

Слика 3: Једноставан C# код

Visual Studio

Microsoft Visual Studio 2017[6] је интегрисано развојно окружење које је направила корпорација Мајкрософт. Користи платформе за развој софтвера попут Windows API-ја, Windows Forms-а, Windows Presentation Foundation-а и осталих. У њему је имплементиран уређивач програмског кода који садржи и IntelliSense (аутоматско предвиђање и допуњавање програмског кода) и систем за рефакторисање кода. Поседује, такође, и уграђени debager, алат за лако дизајнирање форми, алат за дизајнирање класа и алат за дизајн шеме базе података. Подржава мноштво програмских језика (приближно око 36 језика) између којих су C++ и C# (главно окружење за овај програмски језик). Пружа подршку за друге програмске језике путем језичких услуга које се морају инсталирати одвојено.



Слика 4: Креирање пројекта унутар Visual Studia

Visual C# је специјализован за прављење форми, где се под формом подразумева неки прозор на екрану где се приказују одређени подаци, одређена структура података, у одређеном унапред дефинисаном и погодно дизајнираном формату. Форма-дизајнер је графички алат у оквиру Visual Studio-а који омогућава лако и брзо постављање појединих графичких елемената на форму (објекти Button тј. дугме, Label тј. налепница, TextBox тј. део за унос текста итд). DataForm Wizard, DataGrid, DataGridView и остали расположиви су такође у Visual C#. Ово су објекти којима се обезбеђује размена информација између апликације и корисника. Ти објекти ће приказивати податке кориснику или ће преносити податке од корисника до апликације. Они се називају контроле. Форма има карактеристике (својства – *properties*) као и остали објекти у Visual C#. У Windows апликацији је веома битан догађај – Event, нпр. притискање дугмета. Он је праћен одговором тј. реакцијом програма. Наиме, врши се писање инструкција у програму тј. кодирање апликације, и ове инструкције тј. неки блок програмског кода, ће бити извршен када се деси неки догађај. Кодирајући едитор (Code Editor) служи за укуцавање програмских инструкција. Када се отвори кодирајући едитор, неке инструкције су аутоматски генерисане од стране Visual Studio IDE, а неке је потребно да укуца програмер. Погодност Visual Studio-а је да аутоматски генерише део програма да би олакшало програмерски посао. У Visual C# су веома корисне такозване библиотеке класа (Class Library) које омогућавају вишеструку употребу кода (code reusability). Једном написане класе се компајлирају у DLL библиотеку и могу се прикључити разним апликацијама. Потребно је само да се у апликацији дода референца на библиотеку класа. Постоји алат у оквиру Visual Studio-а, Visual Studio Debug, који може доста да помогне код откривања извршних грешки, тј код детекције и елиминације дефеката у програму, и који се зове debugger. Помоћу debugger -а може се програм извршавати линија-по-линија и посматрати шта се дешава, или програм зауставити и онда испитивати вредности појединих променљивих. Ово су особине debugger-а које помажу програмеру да уради дијагнозу грешке, и после исправи извршне грешке у програму. Међутим, треба нагласити да debugger само помаже код откривања дефеката у програму, али грешке мора да открије пре свега ипак сам програмер, путем тестирања програма и размишљања и инспекције инструкција у програму.

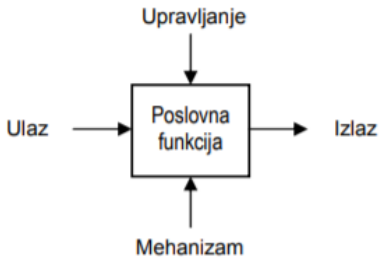
Функционално моделирање

У оквиру функционалног моделирања, спроводи се анализа свих елемената потребних за развој базе података и омогућава декомпоновање пословних функција и планирање потребних ресурса за реализацију функција. Функционално моделирање је везано за технике моделирања активности базираних на комбинацији графике и текста, који су представљени на организован и систематичан начин да би се повећала разумљивост, која подржава анализу, обезбеђује логику за потенцијалне измене, специфицира захтеве и подржава анализу система по нивоима и интегрише активности.

Модел се састоји од хијерархијског низа дијаграма који постепено приказују све више детаља о функцијама и њиховој међувези (interface) са осталим деловима система и омогућава анализу особина одређеног пословног процеса ради његовог максималног унапређења.

Дијаграм контекста

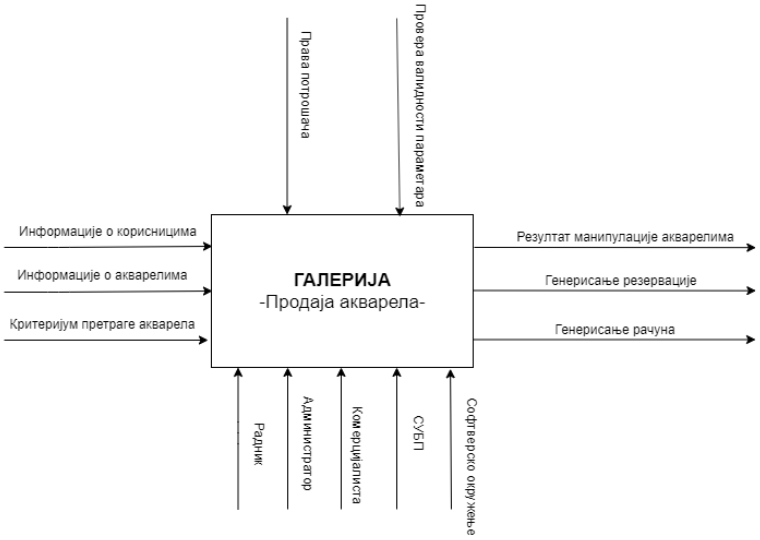
Први корак у оквиру функционалног моделирања је израда дијаграма контекста. Дијаграм контекста је кључан за дефинисање граница система. Дијаграм је дефинисан једним павоугаоником који представља границу модела који се проучава. У систему и ван њега, информације теку преко стрелица. Контекстни дијаграм је навиши ниво апстракције који се касније преводи у нижи ниво апстракције. Изглед дијаграма контекста приказан је на слици



Слика 5: Општи облик дијаграма контекста

- Улаз (Input) представља нешто што се троши у процесу
- Управљање (Control) представља ограничење на извршавање процеса
- Излаз (Outpu) представља резултат процеса
- Механизам (Mechanism) представља оно што се користи у процесу али се не троши

Дијаграм контекста за апликацију која је обрађена у оквиру овог дипломског рада, приказан је на слици 6.



Слика 6: Дијаграм контекста за галерију акварела

Стабло активности

Након креирања дијаграма контекста, следи дефинисање стабла активности. На основу дефинисаних граница система, помоћу стабла активности се успостављају вертикалне (хијерархијске) везе између активности. Сложена активност се растављa на више подређених активности. На тај начин, стабло активности представља хијерархију дефинисаних активности и омогућава функционалну декомпозицију разматраног проблема. Стабло активности се дефинише применом методе решавања проблема одозго на доле (top-down). Сложена активност се растављa на више подређених активности, а затим се приступа решавању једноставних, подређених, активности. Полазна сложена активност се развија у структуру типа стабла. Корен стабла (то је највиши чвор стабла) садржи полазну активност, док листови, тј. чворови који немају потомке, садрже активности чије је решавање релативно једноставно. Решавањем свих активности из листова решена је и полазна сложена активност. Разбијање активности „родитељ“ на своју децу треба да има минимум две до максималног броја шест подређених активности.

Уколико је број подређених активности већи од дозвољеног броја, значи да је покушано да се смести превише детаља за један ниво. За апликацију која се разматра у оквиру овог дипломског рада, стабло активности приказано је на слици 7.

Слика 7: Стабло активности за галерију акварела

Информационо моделирање

Информационо моделирање изводи се на основу дефинисаних критичних функција за претходно урађену активност функционално моделирање. Критичне функције представљају, свака за себе, по један главни пројекат који се развија у оквиру активности информационо моделирање. Информациони модел приказује у каквом су међусобном односу подаци у неком реалном систему. Информационо моделирање омогућава:

1) Дефинисање системске документације која се може користити за базу података и за развој апликација да би корисници могли да дефинишу системске захтеве
2) Бољу комуникацију међусобно и са крајњим корисницима
3) Јасну слику о пословним правилима
4) Логичку слику базе података коју могу користити аутоматски алати за генерисање система за управљање базама података

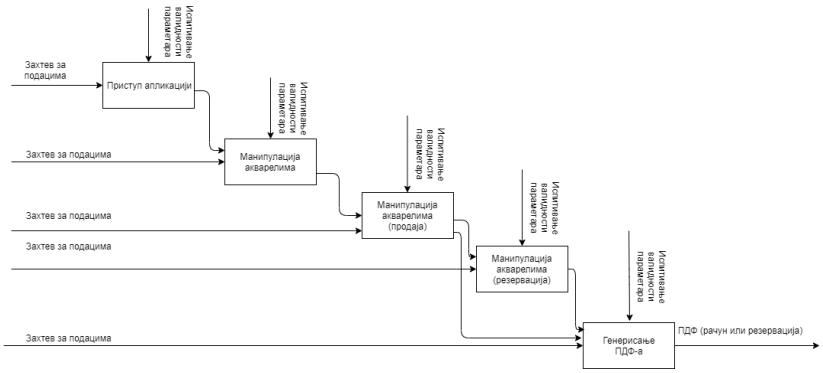
Дефинисање детаљних захтева

Дефинисање детаљних захтева се изводи на основу претходно урађене активности функционално моделирање, тј. дефинисане студије којом се одређују оквири који се у овој активности, за изабране подсистеме, детаљно разграђују.

Дефинисање детаљних захтева треба да представља ревизију постављених елемената из студије уз даље детаљно продубљивање за изабрану функцију за коју се спроводи реинжењеринг пословних процеса.

Дефинисање декомпозиционог дијаграма

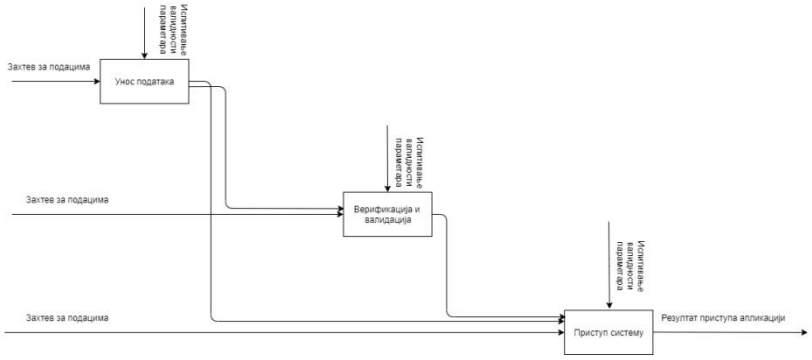
Дефинисање декомпозиционог дијаграма следи након дефинисања дијаграма контекста и стабла активности. Помоћу декомпозиционог дијаграма успостављају се хоризонталне везе између подактивности које су повезане стрелицама. Активности су смештене у правоугаоницима који се цртају у дијагоналном смеру. Свакој активности мора се доделити назив у облику глаголске фразе, те мора имати најмање једну контролну и једну излазну стрелицу. Декомпозициони дијаграм за апликацију из дипломског рада приказан је на слици 8.



Слика 8: Демопозициони дијаграм галерије акварела

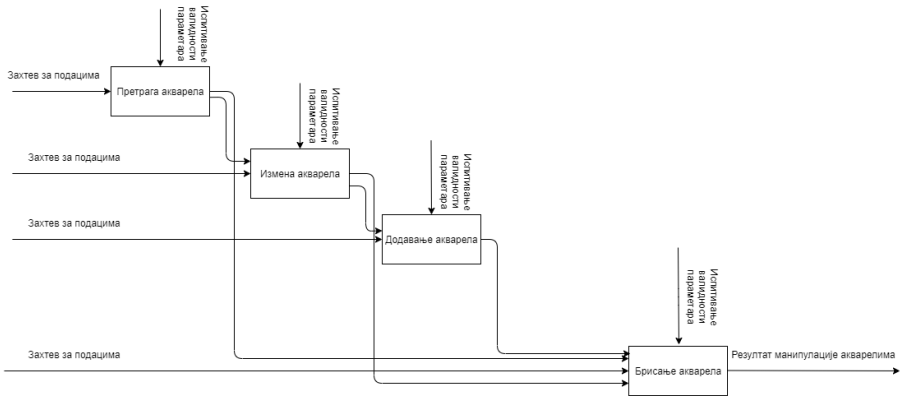
Као што се може видети, на основу стабла активности, приказане су све подактивности на које је подељена главна (root) активност.

Четри активности које се виде на дијаграму декомпозиције су приступ апликацији тј. LogIn корисника, манипулација акварелима (претрага, додавање, измена и брисање акварела), манипулација акварелима у виду куповине и манипулација акварелима у виду резервације. Јако битна ствар код коришћења овог дијаграма јесте могућност приказивања дијаграма за сваку појединачну активност из стабла активности. Активности из листова стабла састоје се од појединачних малих активности које су задужене за извршавање поменутих активности. Ово је јако добра могућност визуелизације проблема јер се програмеру пружа увид у решавање задатка до најситнијих делова.



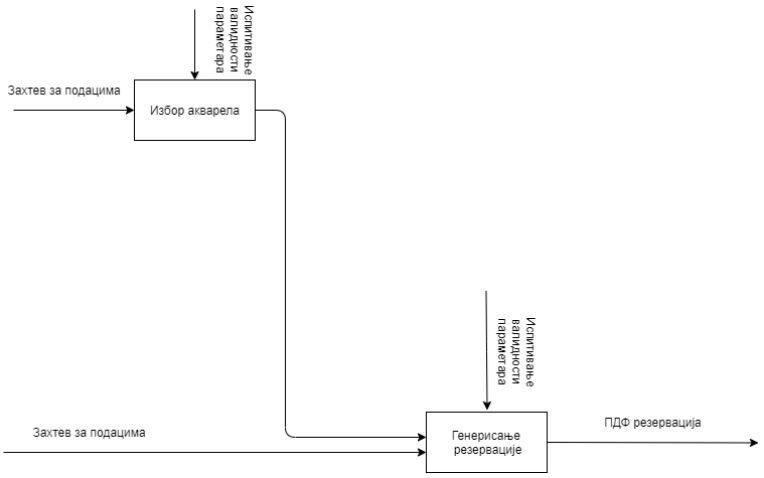
Слика 9: Декомпозициони дијаграм за приступ апликацији

Активност приступ апликацији састоји се од три подактивности. Прва од њих је унос података, затим следи верификација и валидација и на крају приступ систему.



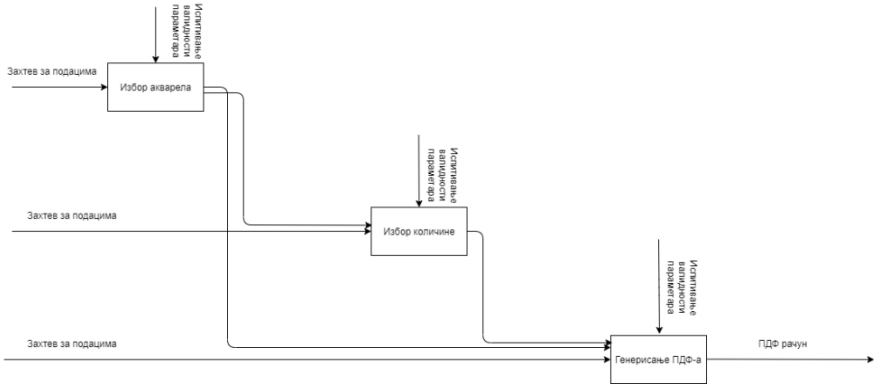
Слика 10: Декомпозициони дијаграм за манипулацију акварелима

Активност манипулација акварелима се састоји од четри подактивности. Прва је претрага акварела, затим иде додавање акварела, измена акварела и на крају брисање акварела.



Слика 11: Декомпозициони дијаграм за резервисање акварела

Активност манипулација акварелима у виду резервације састоји се од две подактивности. Прва је избор акварела док је друга подактивност генерисање резервације.



Слика 12: Декомпозициони дијаграм за продају акварела

Последња активност је манипулација акварелима у виду продаје. Она се састоји од три подактивности. Прва је избор акварела, затим следи избор количине и на крају генерирање ПДФ-а тј. продаја.

Креирање ЕР дијаграма

Креирање ЕР дијаграма представља моделирање података тј. дефинисање ентитетног дијаграма. За дијаграм треба користити технику за описивање структуре података и пословнох правила под називом модел података, којим се дефинишу ентитети. Сваки ентитет има своје особине (атрибуте) а све је то повезано везама. Према устаљеним конвенцијама, великим словима у једнини пишу се ентитети док се атрибути и везе пишу малим словима.

Идентификација ентитета и веза

За активност идентификација ентитета полази се од објеката посматрања. Објекат посматрања је све што се може једнозначно идентификовати, па самим тим и изоловати из околине и описати. Тако је објекат посматрања и сам ентитет. Ентитет је особа, ствар, догађај, појам (реални или апстрактни) који је од трајног интереса тј. нешто што се жели појединачно посматрати. Конкретно за ову апликацију, ентитети су:

1. Галериста
2. Производ
3. Добављач
4. Садржи
5. Листа
6. Рачун
7. Резервација
8. СеНалази

Оваквим одабиром ентитета обезбеђено је да редуданса података приликом складиштења буде минимална.

Идентификација веза дефинише везе између ентитета. Веза представља интеракцију међу објектима тј. знање о њиховој међусобној повезаности. Везама се дефинишу зависности између ентитета односно описују начини повезивања (узајамна дејства) ентитета. Везе у реченици су глаголске фразе које показују какав је однос међу ентитетима. Премда, глаголске фразе не описују правила прецизно, оне дозвољавају особи која посматра модел да стекне основни осећај о повезаности ентитета. Добра је пракса ако читање модела даје ваљане реченице (исказе). У оквиру ове апликације, везе које се користе за ентитете су:

1. Галериста позива Добављача
2. Производ се налази на Рачуну
3. Листа садржи Производ
4. Резервација садржи Производ
5. Галериста штампа Рачун
6. Галериста штампа Резервацију
7. Добављач наручује Производ

Креирање атрибута

Сваки објекат реалног света је дефинисан особинама па по тој аналогији и ентитети имају атрибуте којима се описују карактеристична својства. Након дефинисања атрибута, одређују се примарни кључеви који једнозначно одређују сваки ентитет. Атрибути или групе атрибута који могу бити изабрани као примарни кључеви називају се атрибути кандидати. Кандидат за кључ мора јединствено да идентификује сваки примерак ентитета. Из тога следи да ниједан део примарног кључа не сме бити Null односно празан или недостајући. Од кандидата за кључеве бира се један који се назива примарни кључ. Избор кључа неког ентитета може бити један атрибут којим се дефинише једноставни примарни кључ или комбинација атрибута којом се дефинише сложени примарни кључ. Примарни кључеви се додељују аутоматски и раде по принципу аутоматског инкрементовања, што значи да корисницима није омогућена ручна додела вредности за примарне кључеве.

1. Ентитет Галериста има следеће атрибуте:

Галериста(ID, име презиме, позиција, корисничко_име, шифра)

2. Ентитет Добављач има следеће атрибуте:

Добављач(ID, назив, имејл, ID_галериста)

3. Ентитет Производ има следеће атрибуте

Производ(ID, назив, категорија, произвођач, цена, набавна_цена, гаранција, опис, количина, линк, слика, број_продатих, резервисано, ID_добављача)

4. Ентитет Рачун има следеће атрибуте:

Рачун(ID, време_продаје, ID_галериста)

5. Ентитет сеНалази има следеће атрибуте:

сеНалази(ID_рачуна, ID _производа, количина)

6. Ентитет Листа има следеће атрибуте:

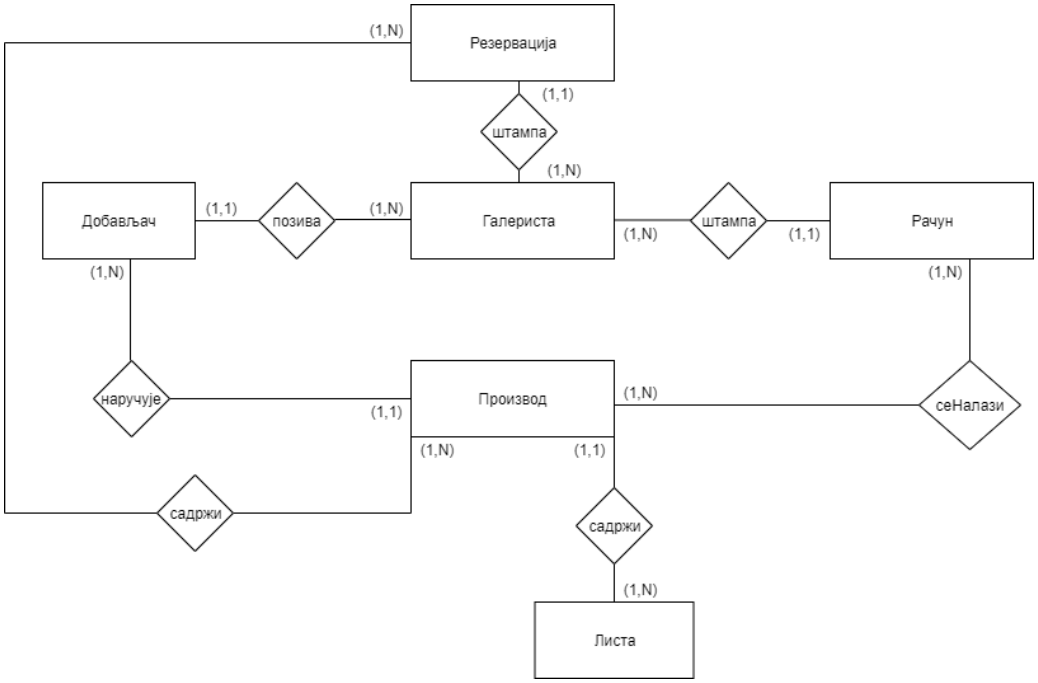
Листа(ID, ID _производа)

7. Ентитет Резервација има следеће атрибуте:

Резервација(ID, време_резервације, ID_галериста)

8. Ентитет Садржи има следеће атрибуте:

Садржи (ID_резервације, ID _производа)



Слика 13: ЕР модел базе података

Дефинисање физичког дизајна базе података

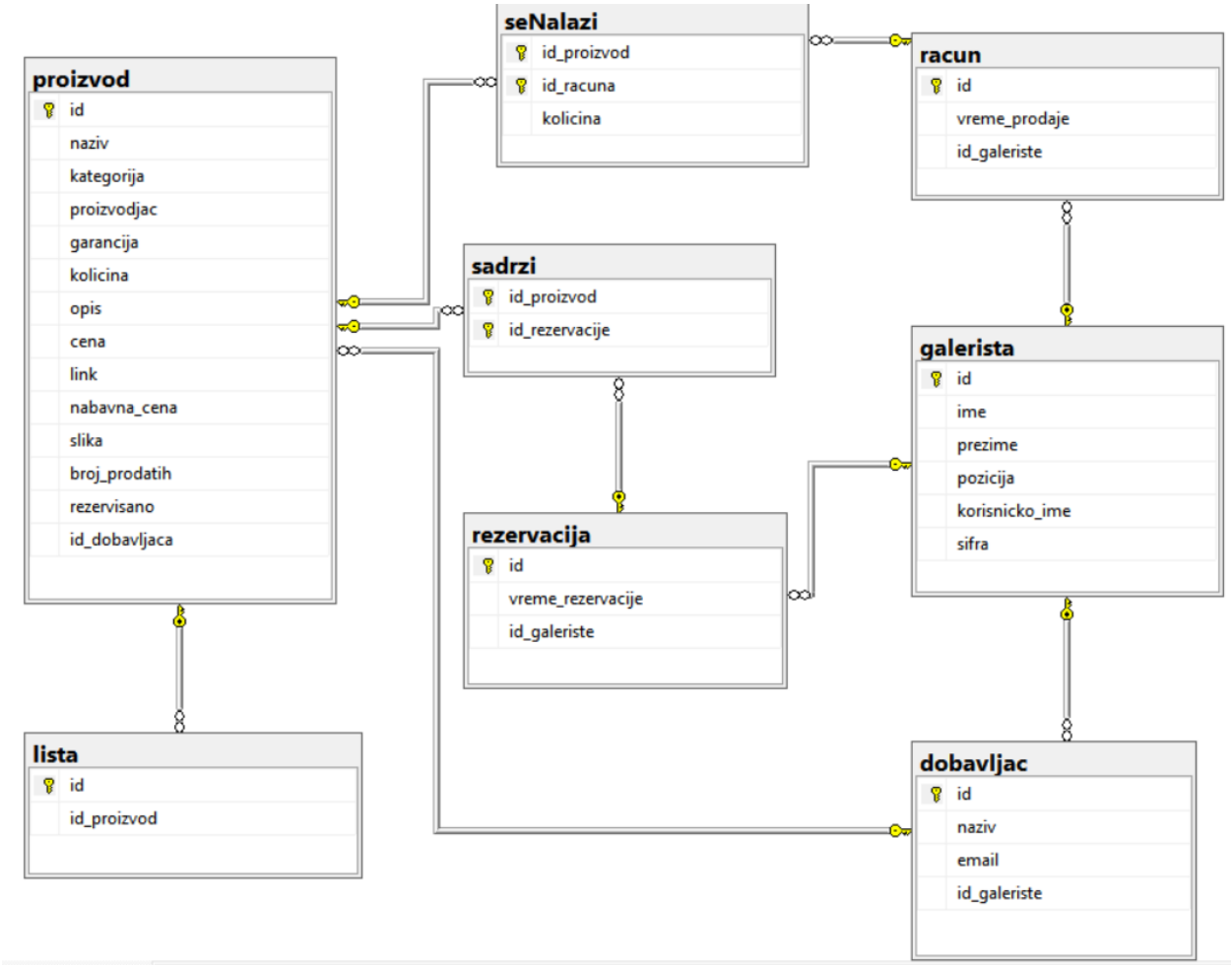
Циљ ове фазе је превођење логичког у физички дизајн, односно превођење ентитета у табеле, атрибута у колоне, као и одговарајућа ограничења. Најпре је потребно извршити избор СУБП (Систем за Управљање Базом Података). База података ће бити креирана у оквиру MS SQL Server-а. Затим је потребно дефинисати табеле и колоне.

Релациони модел базе података тј. сама база података је скуп информација које се односе на постављену тему, односно систем. Тему, односно систем, чине одређени ентитети базе података. Подаци у оквиру базе су уређени и груписани по ентитетима. Сваки ентитет са собом носи одређене врсте података који га карактеришу, а то су атрибути ентитета. Складиштење (чување) података врши се у табелама. У релационим базама података, то је више табела које носе податке о ентитетима.

Табеле су међусобно повезане и функционишу као целина. Релациони модел базе података омогућава максималну флексибилност и економичност у чувању и коришћењу података.
Информације о ентитетима се смештају у табеле. За сваки ентитет формирана је засебна табела. Ентитет чини мноштво субјеката, а атрибутима се одређују карактеристични подаци за све субјекте ентитета. Атрибут има своје име по којем се разликује од осталих у истој реалацији. Вредност једног атрибута су подаци истог типа. Дакле дефинисан је скуп дозвољених вредности за атрибут који се зове домен атрибута[5].
Вредност атрибута је једнострука и једноставна (не може се раставити на делове). Неки атрибути захтевају своје атрибуте, онда их треба сматрати новим ентитетом.
У табели сваки субјекат је једнозначно одређен, што подразумева постојање врсте података, односно атрибута који то омогућава. Поље јединствених вредности дефинише се као поље примарног кључа. На основу те врсте податка, односно преко поља примарног кључа врши се повезивање табела.
Пројектовано решење у основи је релациона база података која се налази на централном серверу којем корисници приступају ради уноса и обраде података и користе јединствене шифарнике за идентификацију чиме се омогућава конзистентност базе података.
База је смештена на потпуно независној инстанци SQL-servera, који има сопствену главну базу, и овакво решење обезбеђује врло висок ниво заштите података и практично приступ подацима није могућ ван апликације[4]. Комплетно решење је реализовано тако да омогућава лак рад у мрежи која ради по принципу клијент - сервер.

Генерисање шеме базе података

Шему базе података чине физичке табеле, колоне и релације. Процес генерисања шеме базе података из логичког модела података назива се директни инжењеринг. Када се генерише шема базе података, ентитети прелазе у табеле, атрибути у колоне, а везе у релације и дефинишу се референцијални интегритети и друге особине које подржава MS SQLServer.



Слика 14: Шема базе података

Галериста(ID, име презиме, позиција, корисничко_име, шифра)

Добављач(ID, назив, имејл, ID_галериште)

Производ(ID, назив, категорија, произвођач, цена, набавна_цена, гаранција, опис, количина, линк, слика, број_продатих, резервисано, ID_добављача)

Рачун(ID, време_продаје, ID_галериште)

сеНалази(ID_рачуна, ID _производа, количина)

Листа(ID, ID _производа)

Резервација(ID, време_резервације, ID_галериште)

Садржи (ID_резервације, ID _производа)

Међурелациона ограничења:

Производ (ID_добављача) > **Добављач**(ID)
Листа (ID_производа) > **Производ** (ID)
Садржи (ID_производа) > **Производ** (ID)
Садржи (ID_резервације) > **Резервације** (ID)
СеНалази (ID_производа) > **Производ** (ID)
СеНалази (ID_рачуна) > **Рачун** (ID)
Резервација (ID_галериста) > **Галериста**(ID)
Рачун (ID_ галериста) > **Галериста** (ID)
Добављач (ID_ галериста) > **Галериста** (ID)

У наставку текста ће бити приказан код којим су креиране табеле и релације између табела у бази података.

Табела Добављач

```
USE [galerija]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[dobavljac](
	[id] [int] IDENTITY(1,1) NOT NULL,
	[naziv] [varchar](50) NULL,
	[email] [varchar](50) NULL,
	[id_galeriste] [int] NULL,
 CONSTRAINT [PK_dobavljac] PRIMARY KEY CLUSTERED
(
	[id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[dobavljac]  WITH CHECK ADD  CONSTRAINT [FK_dobavljac_galerista]
FOREIGN KEY([id_galeriste])
REFERENCES [dbo].[galerista] ([id])
GO

ALTER TABLE [dbo].[dobavljac] CHECK CONSTRAINT [FK_dobavljac_galerista]
GO
```

Табела Производ

```
USE [galerija]
GO
SET ANSI_NULLS ON
```

```
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[proizvod](
	[id] [int] IDENTITY(1,1) NOT NULL,
	[naziv] [varchar](150) NULL,
	[kategorija] [varchar](150) NULL,
	[proizvodjac] [varchar](50) NULL,
	[garancija] [int] NULL,
	[kolicina] [int] NULL,
	[opis] [varchar](500) NULL,
	[cena] [float] NULL,
	[link] [varchar](500) NULL,
	[nabavna_cena] [int] NULL,
	[slika] [varchar](500) NULL,
	[broj_prodatih] [int] NULL,
	[rezervisano] [int] NULL,
	[id_dobavljaca] [int] NULL,
 CONSTRAINT [PK_proizvod] PRIMARY KEY CLUSTERED
(
	[id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
ALTER TABLE [dbo].[proizvod]  WITH CHECK ADD  CONSTRAINT [FK_proizvod_dobavljac] FOREIGN
KEY([id_dobavljaca])
REFERENCES [dbo].[dobavljac] ([id])
GO

ALTER TABLE [dbo].[proizvod] CHECK CONSTRAINT [FK_proizvod_dobavljac]
GO
```

Табела Садржи

```
USE [galerija]
GO

SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[sadrzi](
	[id_proizvod] [int] NOT NULL,
	[id_rezervacije] [int] NOT NULL,
 CONSTRAINT [PK_sadrzi] PRIMARY KEY CLUSTERED
(
	[id_proizvod] ASC,
	[id_rezervacije] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[sadrzi]  WITH CHECK ADD  CONSTRAINT [FK_sadrzi_proizvod] FOREIGN
KEY([id_proizvod])
REFERENCES [dbo].[proizvod] ([id])
GO

ALTER TABLE [dbo].[sadrzi] CHECK CONSTRAINT [FK_sadrzi_proizvod]
GO

ALTER TABLE [dbo].[sadrzi]  WITH CHECK ADD  CONSTRAINT [FK_sadrzi_rezervacija] FOREIGN
KEY([id_rezervacije])
REFERENCES [dbo].[rezervacija] ([id])
GO

ALTER TABLE [dbo].[sadrzi] CHECK CONSTRAINT [FK_sadrzi_rezervacija]
GO
```

UML дијаграми

Unified Modeling Language (UML) је стандардни графички језик за моделовање објектно-оријентисаног софтвера. Unified нам говори да он уједињује све постоједе нотације, Modeling зато што се користи за моделовање софтверских елемената док Language као што и сама реч каже - представља средство комуникације. Основни елементи UML-a su:

1. Елементи (битни концепти - класе, објекти, поруке)
2. Релације (повезивање индивидуалних ствари)
3. Дијаграми (груписање међусобно повезаних колекција ствари и релација)

Оно што је битно напоменути је да се UML дијаграми деле на две целине, и то на:

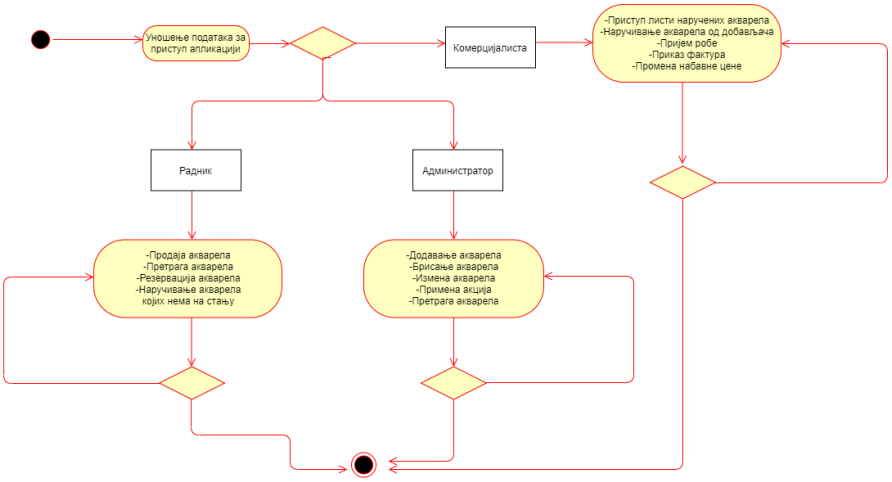
1. Структурне дијаграме (Structure diagram) унутар којих се налазе дијаграми класа (class diagram), објеката (object diagram), компоненти (component diagram), постављања (deployment diagram), пакета (package diagram), профила (profile diagram), композитне структуре (composite structure diagram)
2. Дијаграме понашања (Behaviour diagram) који обухватају дијаграм случајева коришћења (use case diagram), активности (activity diagram), машинског стања (state machine diagram), као и дијаграме интеракције (interaction diagram) унутар којих се налазе дијаграми комуникација (communication diagram), редоследа/секвенци (sequence diagram) и времена (timing diagram)

У оквиру овог дипломског рада биће урађена три UML дијаграма и то они који се најчешће користе а то су:
1. Дијаграм активности (Behaviour diagram -> Activity diagram)
2. Дијаграм стања машине (Behaviour diagram -> State Machine diagram)
3. Дијаграм случајева коришћења (Behaviour diagram -> Use Case diagram)
4. Дијаграм секвенци

Дијаграм активности

Дијаграми активности су намењени моделирању динамичких аспеката (понашања) система. Овај дијаграм приказује ток активности коју извршавају објекти као и евентуално ток објеката између корака активности.

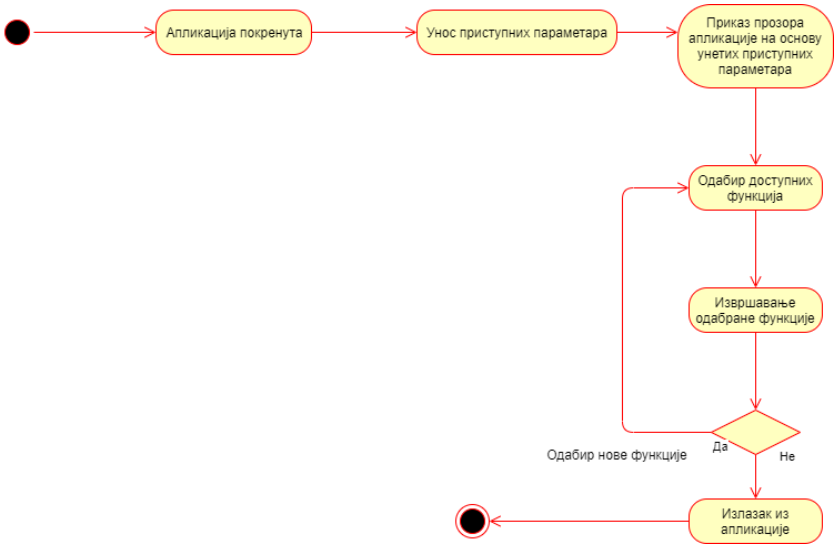
Активност је спецификација понашања која се изражава кроз ток извршења преко секвенцирања и конкурисања подактивности.Акција је основна јединица спецификације понашања која репрезентује неку трансформацију или обраду у моделираном систему.



Слика 15: Дијаграм активности

Дијаграм стања машине

Дијаграм стања је граф који приказује аутомат (машину) стања - чворови су стања док су гране прелази. Ови дијаграми се углавном фокусирају на ток активности као и на понашање самих догађаја. Они се креирају за ентитете који показују битно динамичко понашање унутар система.



Слика 16: Дијаграм стања машине

Дијаграм случајева коришћења

Дијаграм случајева коришћења приказује скуп случајева коришћења и актера. Типично се користи да специфицира неку функционалност и понашање неког субјекта - углавном пројектованог система. Дијаграм визуелизује понашање система, подсистема или чак класе и интерфејса.

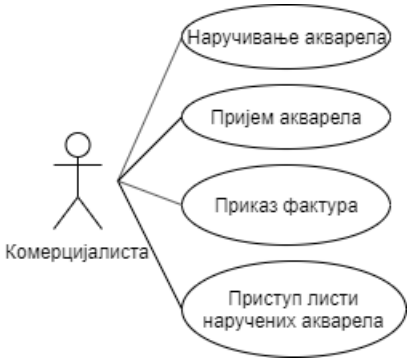
Ауторизован корисник администратор има следеће случајеве коришћења:

1. Додавање акварела
2. Измена акварела
3. Брисање акварела
4. Претрага акварела
5. Примењивање акција

Слика 17: Случај коришћења Администратор

Ауторизован корисник комерцијалиста има следеће случајеве коришћења:

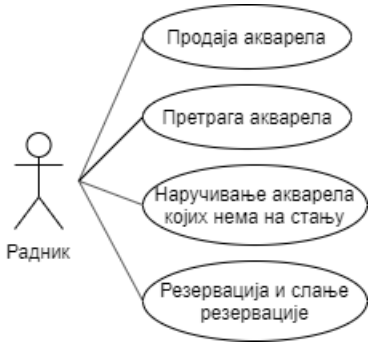
1. Наручивање акварела
2. Пријем акварела
3. Приказ фактура
4. Приступ листи наручених акварела



Слика 18: Случај коришћења Комерцијалиста

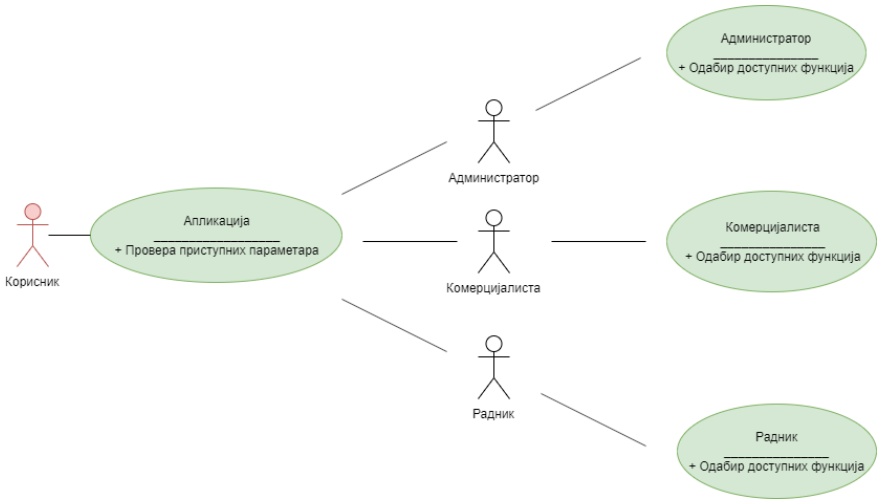
Ауторизован корисник радник има следеће случајеве коришћења:

1. Продаја акварела
2. Претрага акварела
3. Наручивање акварела којих нема на стању
4. Резервација и слање резервације



Слика 19: Случај коришћења Радник

Приказ дијаграма случајева коришћења за целокупну апликацију дат је на слици 20.



Слика 20: Случај коришћења галерије акварела

Дијаграм секвенци

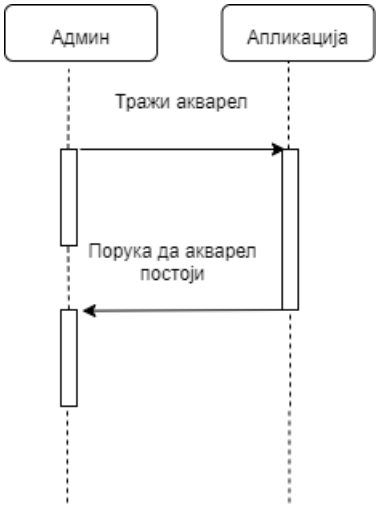
Дијаграм секвенци користи се за спецификацију временских захтева у опису сложених сценарија – опис тока порука између објеката којима се реализује одговарајућа операција у систему. Користи се за моделовање динамичких аспеката система. Идентификују се системски догађаји и системске операције за одговарајући случај употребе. Дијаграм секвенци је један од дијаграма интеракције. Интеракција је понашање које обухвата скуп порука које се размењују између скупа објеката у неком контексту са неком наменом. Порука је спецификација комуникације између објеката која преноси информацију. Пријем поруке изазива акцију (извршење наредбе).

Администратор

Додавање новог акварела:
1. Администратор уноси податке о новом производу (акварелу) и позива програм ради провере података;
2. Апликација проверава постојање података;
3. Администратор врши додавање акварела;
4. Апликација извршава измене;

Слика 21: Дијаграм секвенци за додавање акварела

Изузеци:
Администратор позива апликацију ради провере о постојању акварела, апликација обавештава да дати акварел већ постоји.



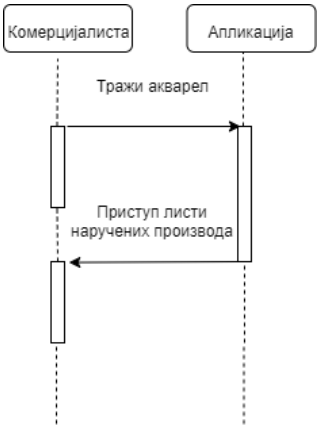
Слика 22: Дијаграм секвенци за додавање акварела (Изузетак)

Промена атрибута постојећег акварела:
1. Администратор уноси податке о акварелима и позива програм ради провере постојања уноса;
2. Програм проверава постојање уноса;
3. Администратор врши промену атрибута и позива програм да евидентира измене;
4. Програм извршава измене;

Слика 23: Дијаграм секвенци за измену акварела

Комерцијалиста

Приступ листи наручених акварела:
1. Комерцијалиста проверава да ли постоје наручени акварели у програму;
2. Програм приказује тражене податке;



Слика 24: Дијаграм секвенци за приступ листи наручених акварела

Наручивање акварела од регистрованих добављача:
1. Комерцијалиста позива програм да прикаже све акваре добављача;
2. Комерцијалиста уноси жељени акварел одређеног добављача;
3. Програм проверава постојање добављача и акварела;
4. Комерцијалиста врши наручивање код регистрованих добављача и позива програм да евидентира измене;
5. Програм извршава измене;

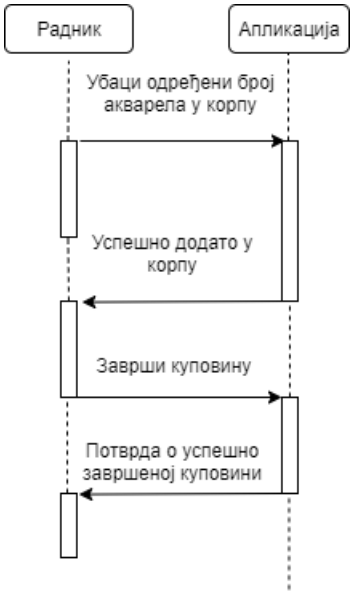
Слика 25: Дијаграм секвенци за наручивање од добављача

Пријем робе, промена стања залиха и чување фактуре у електронском облику:
1. Комерцијалиста прима робу која је стигла и позива програм да провери да ли су то наручени акварели;
2. Програм проверава унете податке и приказује податке комерцијалисти;
3. Комерцијалиста врши уписивање акварела и фактура у програм и позива програм да изврши измене;
4. Програм извршава измене;

Слика 26: Дијаграм секвенци за пријем и промену стања акварела

Радник

Продаја акварела:
1. Радник бира акварел и његову количину из табеле свих акварела у продавници;
2. Програм проверава да ли постоји на стању одговарајући акварел;
3. Ако има довољно акварела на стању, програм их смешта у корпу;
4. Након додавања свих акварела које жели да купи, радник потврђује куповину;
5. Програм генерише рачун и смешта га у меморију рачунара и смањује број на стању купљених акварела;



Слика 27: Дијаграм секвенци за продају акварелу

Изузеци:
Ако радник покуша да у корпу убаци већи број акварела него што тренутно постоји на стању програм му то неће дозволити уз обавештење. Слично, ако је акварел већ додат у корпу, програм му онемогућава поновно додавање уз обавештење.

Слика 28: Дијаграм секвенци за продају акварела (Изузетак)

Претрага акварела:
1. Радник уноси одговарајуће атрибуте за претрагу акварела
2. Програм проверава да ли тражени акварели постоје у бази података акварела
3. Акварели, и информације о њима, који задовољавају критеријуме претраге се једини појављују у табели акварела

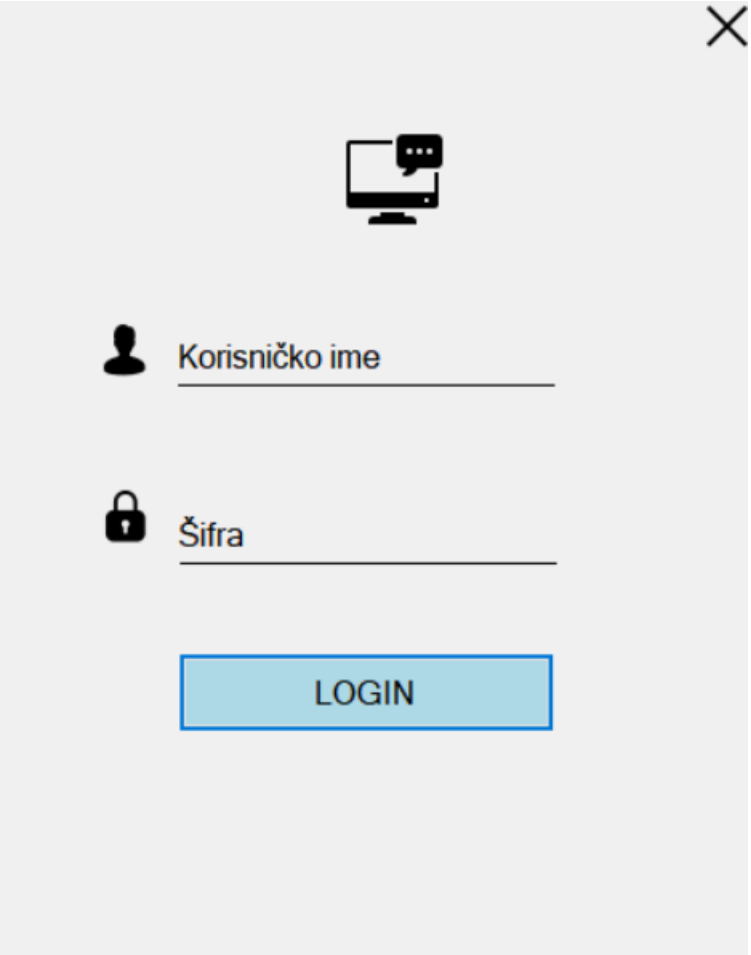
Слика 29: Дијаграм секвенци за претрагу акварела

Наручивање акварела којих нема на стању:
1. Радник пролази кроз листу акварела и има увуд којих акварела нема на стању
2. Радник бира акварел ког нема на стању и жељену количину коју жели да наручи
3. Радник потврђује наручивање одговарајућих акварела
4. Програм складишти захтеве радника и прослеђује их комерцијалисти

Слика 30: Дијаграм секвенци за наручивање акварела којих нема на стању

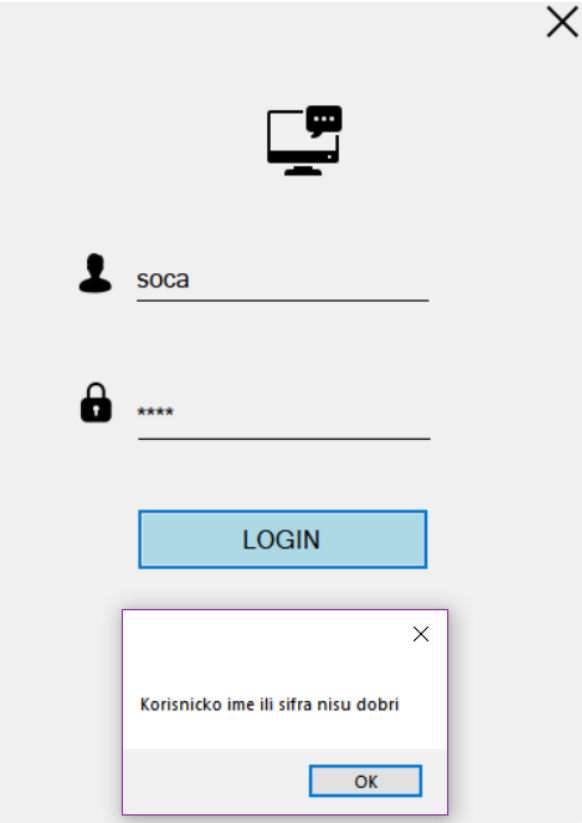
Приказ рада апликације

При покретању програма, отвара се прозор за пријављивање корисника. Програму могу приступити само ауторизовани корисници (администратори, комерцијалисте и радници)[3].

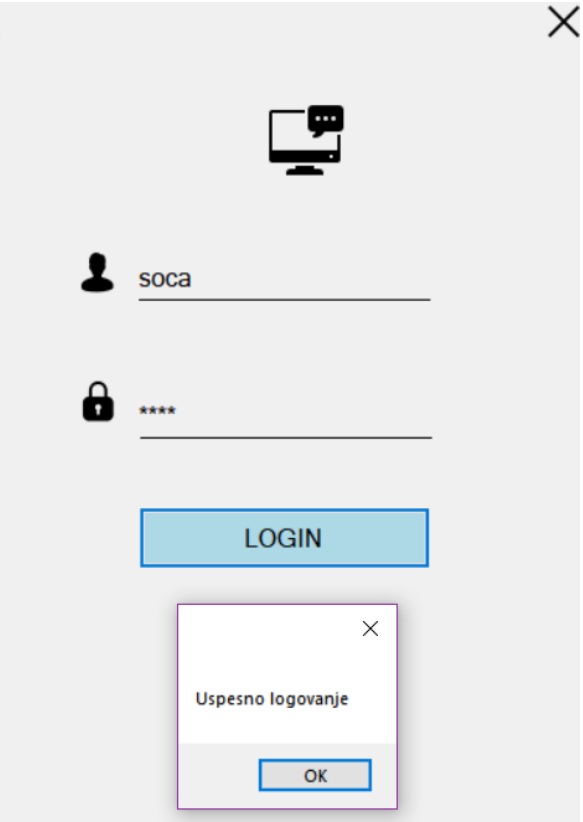


Слика 31: Приступ апликацији

У случају уношења неисправних података корисника, приказује се прозор који обавештава корисника да је неисправно унео податке(слика 32). Када корисник унесе исправне податке, отвориће се одговарајући прозор, у зависности од његове функције (администратора, комерцијалисте, радника), слика 33.

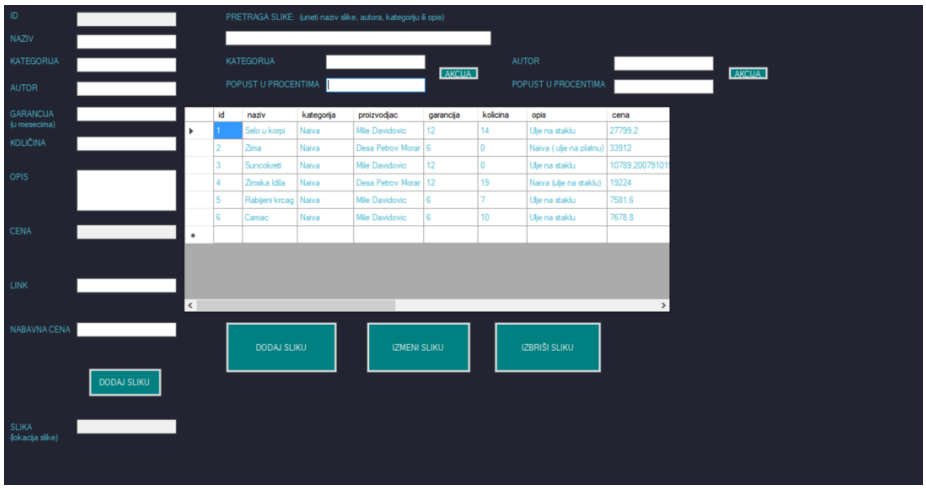


Слика 32: Приступ са неисправним подацима



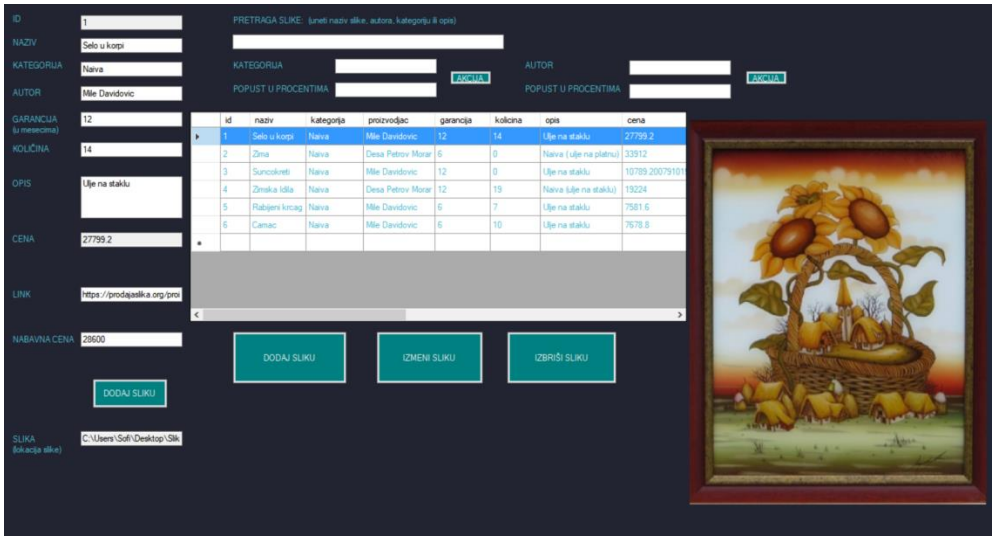
Слика 33: Приступ са исправним подацима

Уколико се корисник улогује као администратор, на почетној страни приказује се табела свих производа. Корисник може да врши манипулацију са производима.

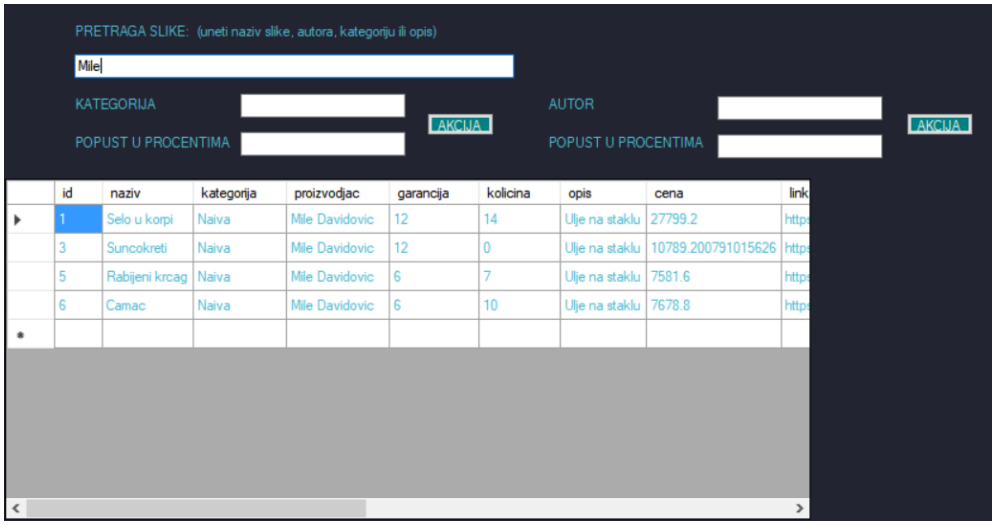


Слика 34: Почетни екран администратора

Битна ставка је да се при одабиру акварела, са десне стране се отвара слика појединачног акварела. Администратор у сваком тренутку може да врши претрагу акварела по одређеним ставкама. На слици 36 види се како функционише претрага по аутору. Уколико жели да дода нове акварѐле, администратор мора да попуни поља (атрибуте акварела), с тим да се ID не може унети јер се сам инкрементује као и цена, јер се цена подешава сама након уношења набавне цене (20% већа од набавне цене). У сваком тренутку, може се избрисати одређени акварели. Ако има захтева за неком изменом, администратор може вршити измене акварела.

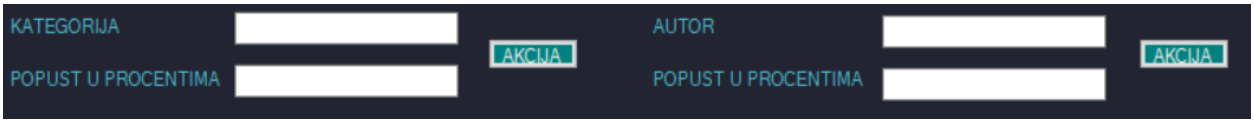


Слика 35: Приказ фотографија акварела



Слика 36: Претрага акварела по одређеној категорији

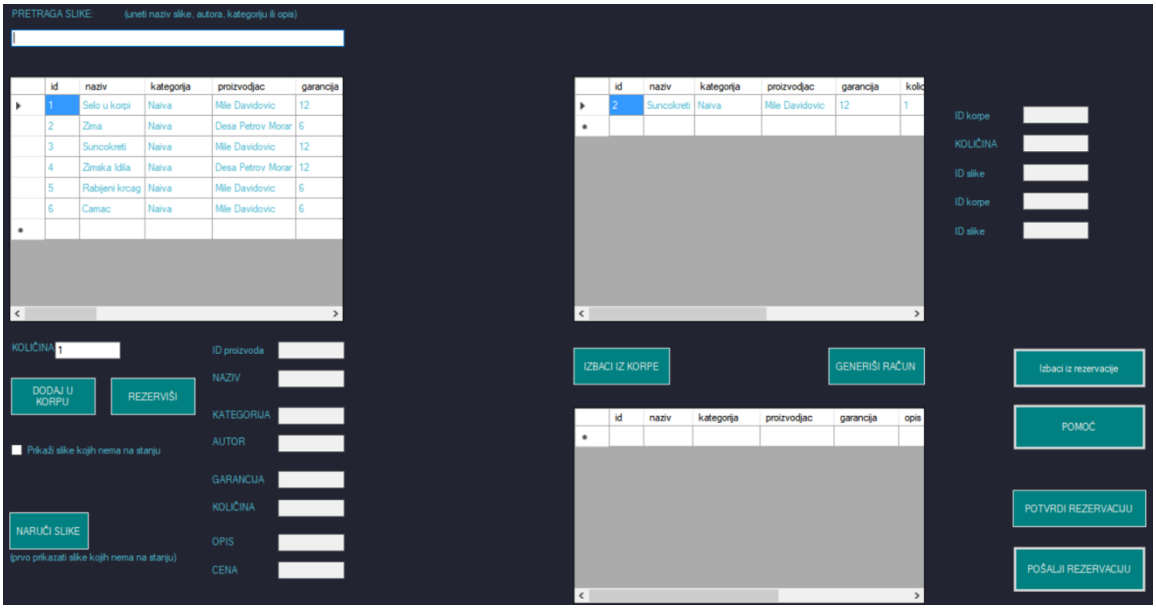
Свака слика (акварел) у оквиру својих атрибута има опис где се могу пронаћи неке основне информације о слици. Ако је заинтересован за детаљнија објашњења, једноставним кликом на колону линк, отвара се интернет страна за тражену слику [1].



Слика 37: Примена акцијских цена на акварале

На слици 37 види се још једна могућност администратора. Када се одабере аутор или категорија, администратор може доделити број процената у оквиру доделе попуста. Ограничење је да попуст иде од 0 до 100 процената.

Ако корисник апликације има тип улоге радник, прозор који му се отвара приказан је слици 38.



Слика 38: Прозор радника

Радник, исто као и администратор, има приступ листи свих акварела из галерије само што нема могућност манипулације над њима. Може вршити претраге по различитим категоријама али било какве измене му нису дозвољене. Кључна улога радника је продаја и резервација акварела тако да је јако битно да у сваком тренутку може отворити интернет странице са детаљним објашњењима или приказати слике акварела. Уколико радник врши продају акварела, пре свега је неопходно да означи акварел, затим да одабере количину односно број акварела који жели клијент да купи, и на крају да то убаци све у корпу.

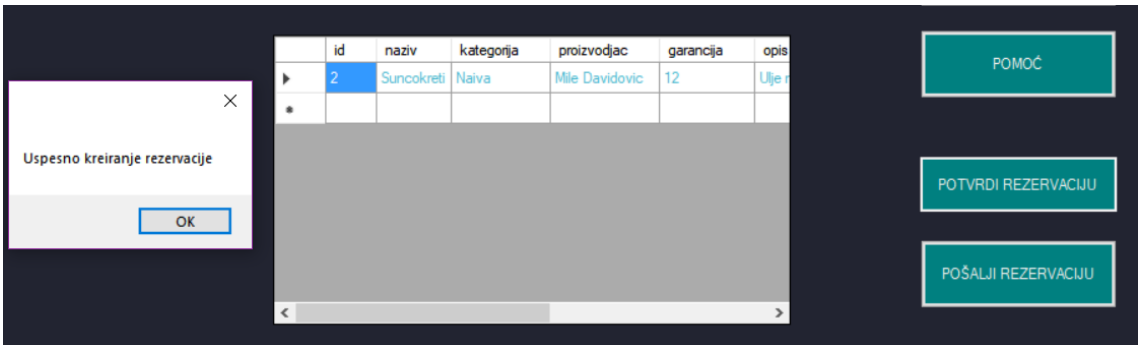
Ако је изабрана количина акварела већа од доступне, јавиће се порука да галерија нема толику количину акварела за тражени производ. Када се заврши са одабиром акварела, неопходно је завршити продају и тада се аутоматски генерише ПДФ рачуна.

Често се дешава да слике ,које су добро продаване, брзо нестају са стања тако да радник у сваком тренутку мора имати увид у оне слике које тренутно нису расположиве за продају. Једноставним одабиром „Прикажи слике којих нема на стању“, раднику се табела са сликама своди на табелу са сликама само оних производа код којих је количина 0. Они су обојени црвеном бојом да би се разликовали од осталих акварела.

Слика 39: Акварели којих нема на стању

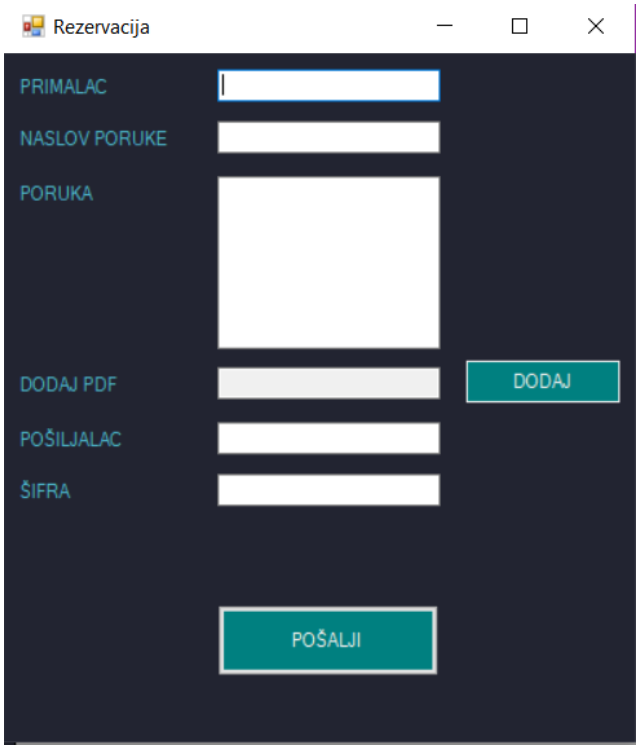
Због великог броја слика које галерија садржи, често се дешава да клијенти не могу да се одлуче коју слику да купе. Зато је још једна могућност радника, управо резервисање слика. Приликом одабира слика, оне се пакују у табелу и на крају се од њих генерише ПДФ. То доста олакшава клијентима јер могу да протумаче и одлуче коју слику да купе.

Сваки пдф се састоји од слика које су изабране, док свака слика садржи назив, аутора и цену.



Слика 40: Креирање резервације акварела

Уколико клијент не жели да има пример резервације на папиру, радник може да изврши слање резервације (пдф-а резервације) преко мејла. Неопходно је унети сва поља са слике 41 да би се мејл успешно послао.

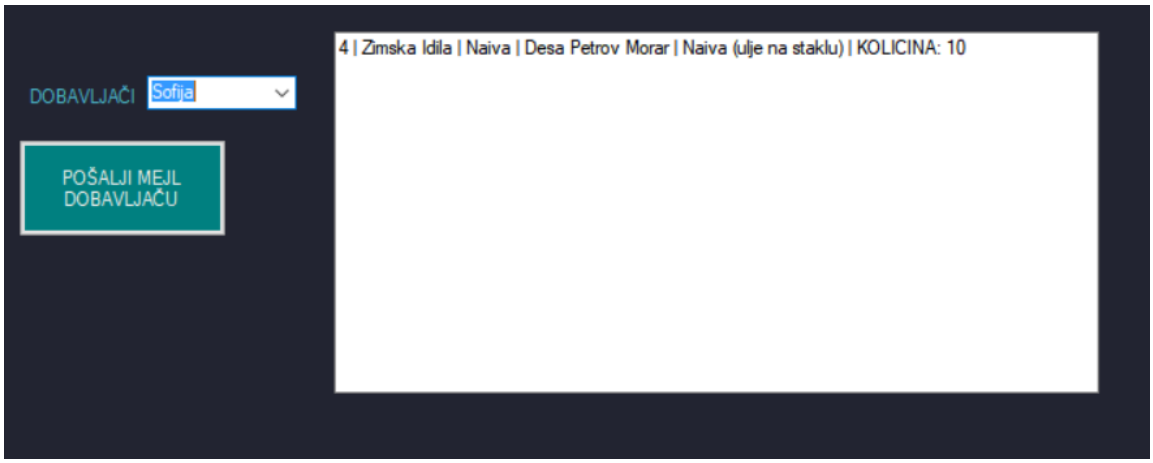


Слика 41: Слање резервације клијенту

Последњи ауторизовани корисник је комерцијалиста. Његове могућности се потпуно разликују од претходна два ауторизована корисника. Комерцијалиста има могућност приступа листи наручених акварела.

Слика 42: Прозор комерцијалисте

Као што се види на слици 43, радник је наручио производ који је одмах као захтев стигао комерцијалисти. Он одлучује коју количину тог одређеног производа жели наручити и од код добављача. Минимум је један производ. Када заврши са одабиром количине, све податке смешта у поруку која се путем мејла шаље добављачу.



Слика 43: Слање захтева добављачу за одређени акварел

Једине могућности у виду манипулације производима код комерцијалисте су промена цена одређених акварела услед промене набавне цене као и промена стања акварела (количине) услед пријема робе од стране добављача.

Закључак

Уз детаљну анализу захтева система, користећи се модерним алатима, направљен је програм који може користити мањим продавницама акварела у вођењу евиденције о свакодневном обављању послова. Програм, различитим корисницима, омогућава различит степен приступа и управљања подацима у зависности од њиховог статуса (администратор, комерцијалиста или радник). Уз минималан број компоненти у прозорима, довољан за обављање свих постављених захтева, програм треба да буде лак за учење новим корисницима и лак за коришћење постојећим корисницима.

Кретање и сналажење кроз апликацију током рада је једноставно, приступање и проналажење информација је лако. Дизајниран је савремен интерфејс који прати сва правила дизајнирања интерфејса. Код дизајнирања апликације највише пажње је дато корисности, тј. функционалности десктоп апликације која излази у сусрет потребама корисника, затим употребљивости која се односи на способност корисника да користе десктоп апликацију да би постигли одређени циљ, креирали нов производ нпр.

Јако битна ставка јесте то што се апликација може надоградити и користити за неке озбиљније компаније које се баве пословима продаје одређених производа.

Литература

1. https://stackoverflow.com/ - Интернет портал за консултације програмера (приступљено 17.09.2019. године)

2. https://docs.microsoft.com/en-us/dotnet/csharp/index - Водич за C# (приступљено 24.09.2019. године)

3. https://www.tutorialspoint.com/csharp/ - Водич за C# (приступљено 24.09.2019. године)

4. Полишчук Ј.: Пројектовање информационих система, Електротехнички факултет, Подгорица, 2007.

5. Лазаревић Б.: Базе података, ФОН Београд, Београд 2003.

6. https://www.youtube.com/watch?v=G49hj6Us0xo – Упутства за рад у Visual Studiu (приступљено 09.09.2019. године)