**Факултет техничких наука**

Нови Сад, Трг Доситеја Обрадовића 6

Пројекта документација

**Банка**

Асистент: Студент:

Сара Кијановић Војин Ћетковић

Нови Сад, јануар 2023. године

Садржај

[1. Опис реалног система 3](#_Toc124005704)

[1.1. Како банке и банкарска индустрија раде 3](#_Toc124005705)

[2. Задатак 4](#_Toc124005706)

[3. Опис технологија 5](#_Toc124005707)

[3.1. GenMyModel 5](#_Toc124005708)

[3.2. Microsoft SQL Server Menagment Studio 5](#_Toc124005709)

[3.3. Visual Studio 5](#_Toc124005710)

[4. UML 6](#_Toc124005711)

[4.1. Дијаграм случајева употребе 6](#_Toc124005712)

[4.2. Дијаграм класа 10](#_Toc124005713)

[4.3. Дијаграм секвенци 12](#_Toc124005714)

[5. База података 13](#_Toc124005715)

[5.1. Креирање базе података 13](#_Toc124005716)

[5.2. Упити и *DML* наредбе 24](#_Toc124005717)

[6. Креирање *WPF* апликације 26](#_Toc124005718)

[6.1. Додавање 33](#_Toc124005719)

[6.2. Измена 54](#_Toc124005720)

[6.3. Брисање 57](#_Toc124005721)

[6.4. Изглед апликације 59](#_Toc124005722)

[7. Тестирање апликације 64](#_Toc124005723)

[8. Закључак 68](#_Toc124005724)

# 1. Опис реалног система

Банка је финансијска институција регулисана на савезном, државном или на оба нивоа. Могу понудити широк спектар производа и услуга, укључујући:

* Депозитне рачуне (текучи рачун, штедни рачун..),
* Кредите, укључујући хипотекарне кредите, ауто кредите и личне зајмове,
* Кредитне картице,
* Услуге уновчавања чекова,
* Осигурање,
* Пословно банкарство...

Финансијска институција је ентитет који се бави трансакцијама које укључују кретање новца или финансијске имовине са једног места на друго. Примери финансијских институција укључују:

* Банке,
* Кредитне уније,
* Штедионичка и кредитна друштва,
* Инвестиционе компаније за мала предузећа,
* Осигуравајућа друштва...

Тип финансијске институције обично дефинише врсту активности или финансијских трансакција којима се бави.

## 1.1. Како банке и банкарска индустрија раде

Банке, без обзира да ли су обичне институције или онлајн, управљају протоком новца између људи и предузећа. Тачније, банке нуде депозитне рачуне који су сигурна места за људе да чувају свој новац. Банке користе новац на депозитним рачунима да дају кредите другим људима или предузећима. Заузврат, банка прима камату на те кредите од зајмопримаца. Део те камате се затим враћа првобитном власнику депозитног рачуна у облику камате – обично на штедни рачун. Банке првенствено зарађују од камата на кредите и накнада које наплаћују својим клијентима. Ове накнаде могу бити везане за одређене производе, као што су банковни рачуни или у вези са финансијским услугама. На пример, инвестициона банка која инвеститорима нуди управљање портфолиом може наплатити накнаду за ту услугу. Или, банка може да наплати накнаду за почетак приликом одобравања хипотекарног кредита купцу куће.Банкарство је високо регулисана индустрија. Систем федералних резерви надгледа банке и друге финансијске институције и координира са државним регулаторним агенцијама како би осигурао да банке следе одговарајуће смернице. Банке такође подлежу регулацији других савезних агенција, укључујући Канцеларију контролора валуте (*OCC*), Канцеларију за надзор штедње (*OTC*) и Федералну корпорацију за осигурање депозита (*FDIC*).

# 2. Задатак

У оквиру овог пројекта, анализираћемо систем банке, како банка фукционише и направићемо апликацију коју ће корисити запослени у банци.

Технологије као што су *GenMyModel*, *Microsoft SQL Server Menagment Studio*, *Visaul Studio* нам омогућавају да представимо систем банке, креирамо базу података која ће служити за чување података које запослени у банци свакодневно користе, као и за креирање апликације.

Уз помоћ дијаграма случајева употребе представићемо интеракцију корисника са системом, и разне случајеве употребе које та интеракција подразумева. Дијаграмом класа представићемо систем што ће нам омогућити моделовање структуре апликације, као и лакше креирање апликације. Дијаграм секвенце ће нам омогућити да лакше схватимо како се интеракције између корисника и система одвијају у времену.

Описаћемо процес креирања базе уз помоћ *Microsoft SQL*-а, како се креирају и повезују табеле... Све то ће нам омогућити да наша апликација коју ћемо креирати уз помоћ *Visual Studio*-a ради како треба, и врши операције над подацима који су везани за банку.

Такође ћемо описати и поступак креирања апликације, како се креирају форме, конекција са базом података, методе које омогућавају функционалност дугмића, и разне друге функицоналности.

# 3. Опис технологија

Представићемо кратак опис технологија које смо користили у реализацији пројекта.

## 3.1. GenMyModel

*GenMyModel UML* је професионални онлајн софтвер за моделирање који долази са значајном подршком за документацију, анализу процеса и операција. Софтвер превазилази све потребе пословања обезбеђујући богату пословну архитектуру која заузврат доноси футуристичке резултате у смислу тока пословања и продуктивности, чиме се постиже већа компактност. Не постоје више рањивости или компликације у генерисању докумената јер софтвер аутоматизује начин генерисања и дистрибуирања докумената у целој организацији.

Поседује уграђен онлајн репозиторијум, аутоматизовану онлајн документацију, *API* и интеграциону подршку. Омогућава лак приступ моделу са веба, поседује обједињно скалдиште модела... Обезбеђује подршку за све индустријске стандарде и беспрекорно повезује те стандарде у једном алату.

## 3.2. Microsoft SQL Server Menagment Studio

*Microsoft SQL Server Menagment Studio* (*SSMS*) је интегрисано окружење за управљање инфраструктуром *SQL* Cервера. Пружа кориснички интерфејс и групу алата са богатим уређивачима скрипти који су у интеракцији са *SQL* Сервером. *SSMS* пружа алате за конфигурисање, управљање и администрирање инстанци *Microsoft SQL* Сервера, и обједињује низ алата за графички и визуелни дизајн и богате уређиваче скрипти како би се поједноставио рад са *SQL* Сервером. *SSMS* комбиноване функције долазе из *Enterprise Menager*-а, *Query Analyzer*-a i *Analysis Manager*-a. Подржава већину административних задатака *SQL* Сервера и одржава јединствено, интегрисано окружење за управљање *SQL* *Server*-ом.

## 3.3. Visual Studio

*Microsoft Visual Studio* је интегрисано програмско окружење, програмирано од стране компаније *Microsoft*. *Visual Studio* се користи за програмирање рачунарских игара, програма, веб-сајтова, веб-сервиса и веб-апликација на *Microsoft* *Windows*-у. *Visual Studio* користи *Micorsoft* програмске софтвере:

* *Windows Forms*,
* *Windows API*,
* *Window Presentation Foundation*,
* *Windows Store*.

Visual Studio подржава 36 различитих програмских језика. Уграђени језици укључују *C*,*C++, C#*, *F#* ,и *TypeScript*. Подршка за друге програмске језике као што су *Python*, *Ruby*, *Node.js*, и *M* доступна је путем језичких услуга које се инсталирају одвојено.

# 4. UML

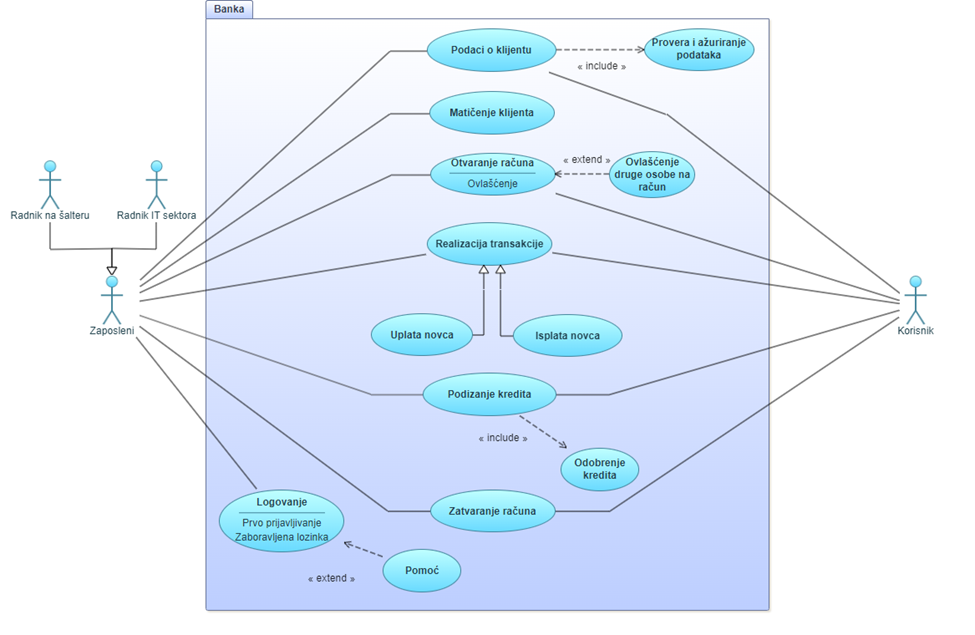
*Unified Modeling Language* је формални језик намењен за:

* Формално специфирање система (могућих стања и понашања),
* Моделовање система,
* Визуелизацију система,
* Анализу система,
* Документовање система.
* Развој софтверских система који представљају моделе других, реалних система,

Заснован је на принципима објектне оријентације.

## 4.1. Дијаграм случајева употребе

Дијаграм случајева употребе (енгл. *Use case diagram*) приказ је интеракције корисника са системом који показује однос између корисника и различитих случајева коришћења у којима је корисник укључен. Дијаграм случаја коришћења може идентификовати различите типове корисника система и различите случајеве коришћења.



Слика 1. Дијаграм случајева употребе

1) **Случај употребе**: Логовање запосленог у систем.

Кратак опис: Запослени се пријављују у систем.

Учесници: Запослени.

Услови који морају бити задовољени пре извршавања: Запослени је потписао уговор са банком, и тако ступио у радни однос.

Опис: Запослени се пријављује у систем тако што уноси своје корисничко име и лозинку. То му омогућава да ради, и да његов рад буде евидентиран у систему. [Изузеци: Прво пријављивање запосленог у систем. ; Запослени је унео погрешне податке за пријаву. ; Запослени је заборавио лозинку.]

Изузеци:

*Прво пријављивање запосленог у систем:*

Уколико запослени није регистрован, он мора да изврши прво пријављивање тако што ће унети све потребне податке.

*Запослени је унео погрешне податке за пријаву:*

Уколико је запослени унео неисправне податке, пријава се враћа запосленом да унесе тачне податке.

*Запослени је заборавио лозинку:*

Уколико се деси да запослени заборави лозинку, омогућава му се обнова лозинке.

Услови који морају бити задовољени након извршавања: Запослени је пријављен у систем, спреман за рад са клијентима.

2) **Случај употребе:** Прикупљање података о клијенту.

Кратак опис: При старту процеса отварања рачуна неоходно је прикупити податке о клијенту, то врши запослени.

Учесници: Запослени, клијент.

Услови који морају бити задовољени пре извршавања: Клијент је пунолетан. У случају да није , родитељ мора бити присутан.

Опис: Неопходно је да клијент попуни формулар у ком ће исписати податке о себи, ту спадају име, презиме, матични број клијента, адреса становања, контакт. Ти подаци се шаљу даље, на следећи корак. [Изузетак: Провера и ажурирање података.

Изузеци:

*Провера и ажурирање података:*

Уколико дође до промене података о клијенту, неопходно је то и евидентирати у систему. . Поступак такође укључује константно проверавање тачности података о клијенту, и њихово ажурирање.

Услови који морају бити задовољени након извршавања: Сви неопходни подаци о клијенту за отварање рачуна су прикупљени.

3) **Случај употребе:** Матичење клијента.

Кратак опис: Убацивање клијента у систем банке.

Учесници: Запослени.

Услови који морају бити задовољени пре извршавања: Подаци о клијенту морају бити сакупљени.

Опис: Након што су подаци у клијенту прикупљени, на ред долази матичење клијента, то јест, убацивање клијента у систем банке, што омогућава наредни корак, а то је отвараје рачуна.

Услови који морају бити задовољени након извршавања: Корисник мора бити унет у систем.

4) **Случај употребе:** Отварање рачуна.

Кратак опис: Клијенту се отвара одговарајући рачун у банци.

Учесници: Запослени, клијент.

Услови који морају бити задовољени пре извршавања: Клијент мора бити у систему, и одлучити који тип рачуна отвара.

Опис: Након матичења клијента, клијент износи податак који тип рачуна жели да отвори, након тога омогућено је отварање одговарајућег рачуна клијенту. Клијент том приликом добија уговор, картицу, као и приступ онлине апликацији банке. [Изузетак: Овлашћење.]

Изузеци:

*Овлашћење друге особе на рачун:*

Клијент има могућност да приликом отварања рачуна овласти другу особу на његов рачун, то врши запослени.

Услови који морају бити задовољени након извршавања: Клијенту је отворен рачун.

5) **Случај употребе:** Реализација трансакције.

Кратак опис: Основна операција са новцем.

Учесници: Клијент, запослени.

Услови који морају бити задовољени пре извршавања: Клијент је одредио коју танскацију врши, да ли уплату или исплату новца.

Опис: Зависно од типа трансакције, који може бити уплата или исплата новца, на шалтеру клијент уплаћује новац или подиже чиме ступа у интеракцију са запосленим, који ће у систему банке евидентирати те трансакције.

Услови који морају бити задовољени након извршавања: Трансакција је извршена и евидентирана у систему.

6) **Случај употребе:** Подизање кредита.

Кратак опис: Клијент подноси захтев за позајмљивање новца од стране банке.

Учесници: Клијент, Запослени.

Услови који морају бити задовољени пре извршавања: Клијент је поднео захтев за одобрење кредита.

Опис: Клијент подноси захтев за кредит код запосленог, износи податак који кредит жели, запослени информишсе клијента о том кредиту, и прослеђује захтев за кредит даље на одобрење. [Изузетак: Одобрење кредита.]

Изузеци:

*Одобрење кредита:*

У зависности од тога да ли је кредит одобрен, клијенту ће бити исплаћен новац, уз кредитни уговор у коме су садржане све информације везане за ту позајмицу.

Услови који морају бити задовољени након извршавања: Клијент мора бити обавештен о крајњем исходу, да ли је му је кредит одобрен или је његов захтев одбијен.

7) **Случај употребе:** Затварање рачуна.

Кратак опис: Клијент затвара рачун у банци јер му није потребан.

Учесници: Клијент, запослени.

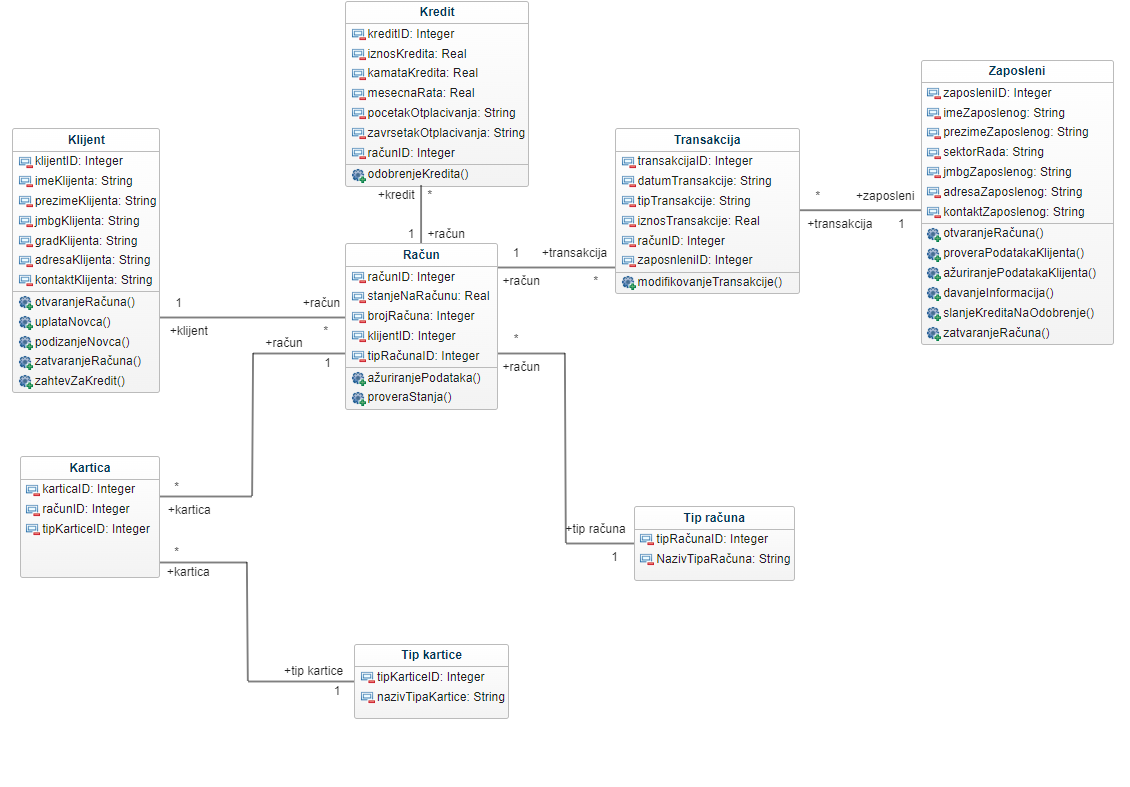
Услови који морају бити задовољени пре извршавања: Клијент има рачун у одговарајућој банци, подноси захтев за затварање рачуна.

Опис: Клијент код запосленог износи жељу за затварање рачуна, запослени обавља све потребне процедуре да би се рачун затворио. Затварање рачуна је неопходно евидентирати у систему банке.

Услови који морају бити задовољени након извршавања: Клијенту је затворен рачун.

## 4.2. Дијаграм класа

Дијаграм класа или класни дијаграм у обједињеном језику за моделовање (*UML*) је тип дијаграма статичке структуре који описује структуру система приказивањем класа система, њихових атрибута, операција (метода) и односа између објеката. Дијаграм класа је главни градивни блок објектно-оријентисаног моделовања. Користи се за опште концепутално моделовање структуре апликације, као и за детаљно моделовање превођења модела у програмски код.



Слика 2. Дијаграм класа

1) **Клијент:**

Обележја класе нам дају основне информације о клијенту као што су његово име, презиме, *ID*, ЈМБГ, адреса, контакт... Све то омогућава да управљање рачунима буде лакше, као и њихово евидентирање у банковном систему, затим и запосленим приликом рада са клијентима.

2) **Запослени:**

Обележја класе нам дају основне информације о запосленима у банци, ту имамо *ID*, име, презиме, адресу, ЈМБГ, адресу, контакт, такође говоре у ком сектору банке раде. Омогућава да рад запослених буде евидентиран у систему банке, као и да објасни класичне радње између корисника и банке, као што су вршење трансакције, подношење захтева за кредит...

3) **Рачун:**

Класа „Рачун“ омогућава лакши рад са рачунима у систему, праћење трансакција које су извршене преко њих, као и да преко броја рачуна и *ID* клијента видимо коме тај рачун припада. Такође можемо да видимо стање одговарајућег рачуна. Можемо да видимо и о ком се типу рачуна ради уз помоћ страног кључа из класе „Тип рачуна“.

4) **Тип рачуна:**

Класа „Тип рачуна“ нам говори о ком рачуну се ради, док нам назив типа рачуна говори да ли је то текући, штедни или девизни. Обележја попут *ID*-а, нам омогућавају да разликујемо рачуне у систему банке. Класа служи за специјализацију класе „Рачун“.

5) **Транскација:**

Класа нам омогућава да трансакције евидентирамо у систему, и даје могућност њиховог праћења. Обележја класе нам говоре о датуму и времену извршења тансакције, њеном износу, типу, као и рачунима који су у њој учествовали. Такође нам говори који запослени је извршио дату трансакцију, што је евидентирано у систему.

6) **Кредит:**

Класа „Кредит“ нам даје детаљније информације о кредиту, колики је износ кредита, камата, месечна рата, када креће отплаћивање кредита, а када се завршава. Помоћу *ID* кредит-a могу да се разликују кредити, као и да се види о ком рачуну се ради преко *ID*-a рачуна.

7) **Картица:**

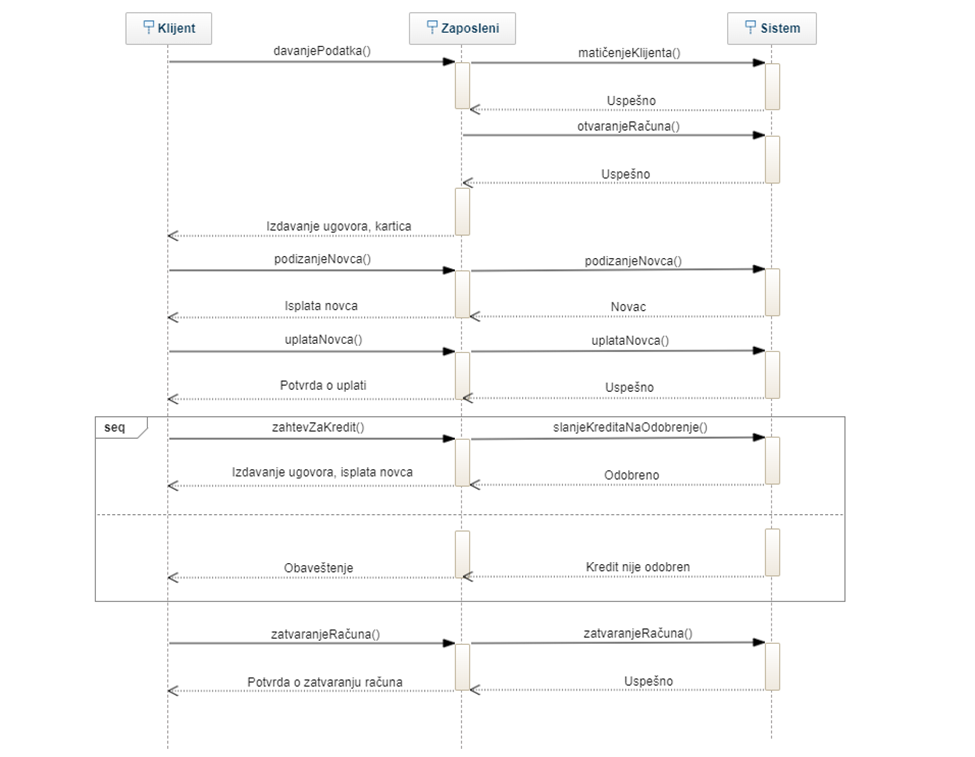
Обележја класе „Картица“ нам омогућавају да имамо евидентиране у систему све картице које су у употреби, као и да обришемо оне које нису, такође да блокирамо изгубљене картице... Можемо да видимо ком рачуну одређене картице припадају уз помоћ страног кључа из класе „Рачун“, као и да видимо о ком типу картице је реч уз помоћ страног кључа из класе „Тип картице“.

8) **Тип картице:**

Обележја класе нам омогућавају да разликујемо типове картица које имамо у систему, док нам обележје „Назив типа картице“ говори о ком типу картице је реч. Ова класа нам омогућава специјализацију класе „Картица“.

## 4.3. Дијаграм секвенци

Дијаграм секвенци (енгл. *Sequence Diagram*) описује време трајања поруке и начин на који објекти у систему међусобно комуницирају, остварујући очекивано понашање. Дакле, приказује се временска компонента и поруке које се прослеђују између објеката у циљу извршења посматране операције.



Слика 3. Дијаграм секвенци

Опис:

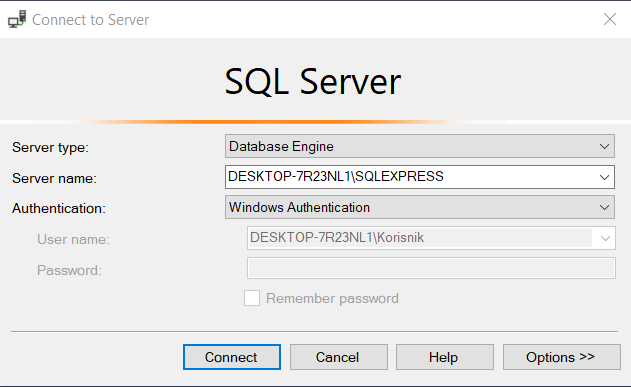
Клијент приликом отварања рачуна даје своје податке запосленом како би он извршио матичење клијента у систему банке, тј. омогућио да се настави процес отварања рачуна, када је процес готов, клијент мора да потпише уговор, када се све то заврши, клијенту се издају картице и примерак уговора који он задржава. У следећим корацима описан је поступак при реализацији неке трансакције (уплата/исплата). У случају подизања новца, клијент запосленом даје средство идентификације како би уопште било могуће извршити трансакцију, након тога запослени исплаћује клијенту новац, даје потврду уплате, а та трансакција се евидентира у систему банке. Сличан је поступак и у случају уплате новца, где клијент даје информације потребне за извршење трансакције (рачун на који се новац шаље, износ..), када се процес заврши, запослени издаје клијенту потврду о изврешеној уплати, такође и ова трансакција се едвидентира у систему. Приликом подизања кредита, клијент прво треба да поднесе захтев за кредит, који банка може да одобри или не одобри у зависноти од примања клијента и разних других фактора. У случају да је кредит одобрен врши се издавање уговора, исплата новца.. А уколико кредит није одобрен, клијенту се доставља извештај да кредит није одобрен, као и разлог зашто није. Приликом процеса затварања рачуна клијент изражава ту „жељу“, док запослени извршава тај процес (затвара рачун, уклања га из система банке...). Када је процес завршен клијенту се преноси информација да је рачун затворен.

# 5. База података

У овом делу ћемо објаснити како се креира база података која нам је потребна да би употпунили смисао апликације.

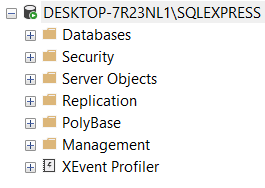
## 5.1. Креирање базе података

Следећи корак у изради информационог система јесте креирање базе података. На самом почетку потребно је покренути SQL Server Management Studio окружење. Када се то уради, појавће се прозор са дијалогом, као што је приказано на слици 4. У приказаном дијалогу, потребно је подесити параметре Server Type и Authentication, као што је приказано на слици 4. Server name није потребно мењати, оно представља назив уређаја на којем се креира база података.

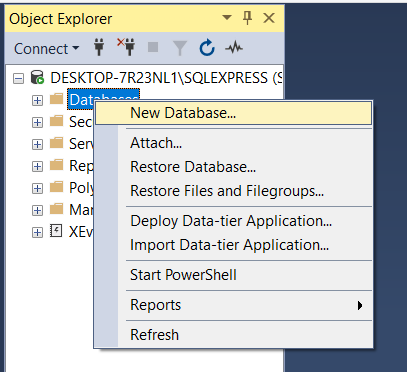


Слика 4. Остваривање конекције на сервер

Након подешавања параметара, кликом тастера миша на дугме *Connect,* дијалог ће се затворити, а отвориће се нови прозор који треба да изгледа као што је приказано на слици 5.

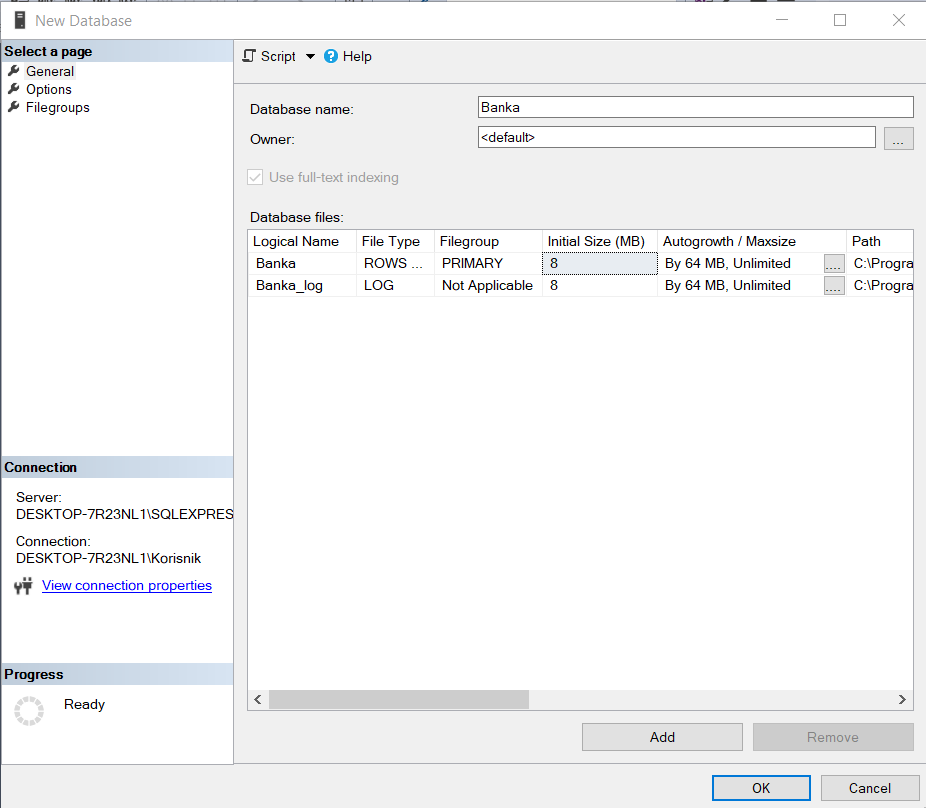


Слика 5. Изглед Object Explorer-а након остваривања конекције са базом података

Сада је потребно креирање базе података за банковну апликацију. База података се креира десним кликом тастера миша на *Database* у *treeview-*у који се налази са леве стране прозора у оквиру *Object Explorer*-а. Отвориће се листа у којој је потребно означити *New database (слика 6).*

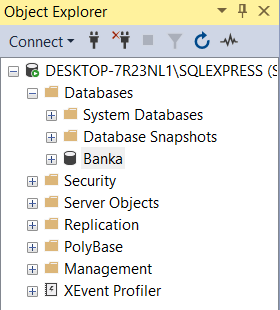
Слика 6. Креирање базе података

Нови прозор ће понудити опцију за креирање базе података. Потребно је дати назив бази података у овом случају назив ће бити „Банка“. Остале параметре није потребно мењати. Кликнути дугме *OK* (Слика 7)*.*



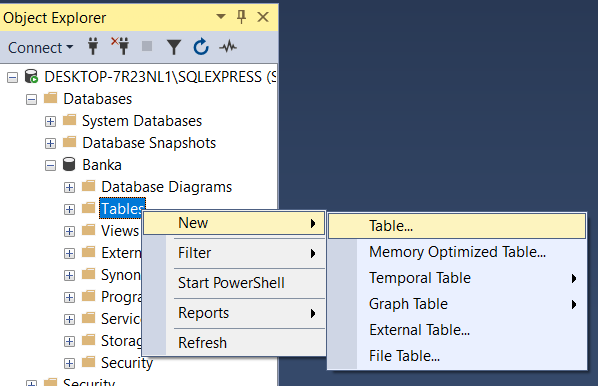
Слика 7. Подешавање параметара при креирању базе података

После успешног креирања базе података у *Object Explorer*-у, у фолдеру *Database* се појављује креирана база података „Банка“ (Слика 8).



Слика 8. Приказ креиране базе података

Проширивањем базе података „Банка“, у стаблу *Object Explorer*-а, појављује се више датотека. Десним кликом тастера миша кликнути на датотеку *Tables* и одабрати *Table* (слика 9).Након клика на ову опцију, са десне стране отвара се прозор за дефинисање табеле, при чему ће табела имати предефинисани назив *Table\_1*.



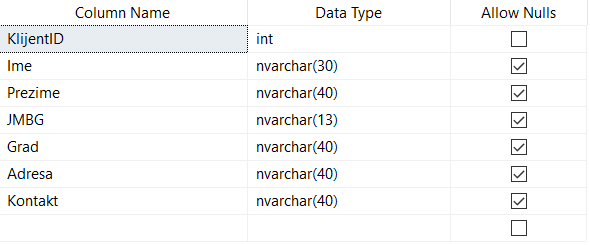
Слика 9. Креирање табеле унутар базе података

Следеће што је потребно урадити јесте креирати колоне табеле тако што че се попунити поља *Column Name, Data Type, Allow Nulls.* У колону *Column Name* се уписује назив атрибута, у колони *, Data Type* се бира тип података за жељени атрибут, док се у колони *Allow Nulls* означава да ли одређено поље може имати *null* вредноси. Препоручује се да *Allow Nulls* буде означено за све колоне, осим за примарни кључ, како би се избегли одређени проблеми приликом уписа података у табелу (слика 10).



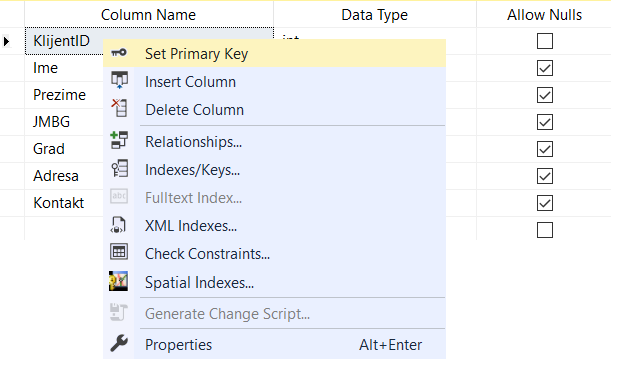
Слика 10. Додавање обележја унутар табеле

Свака табела одређене базе података мора поседовати примарни кључ – *ID* и било која колона може да се декларише као примарни кључ. У овом случају, колона *KlijentID* треба да се декларише као примарни кључ. Као таква, ова колона ће због своје јединствене вредности имати тип података *int* и неће имати *null* вредности (слика 11).



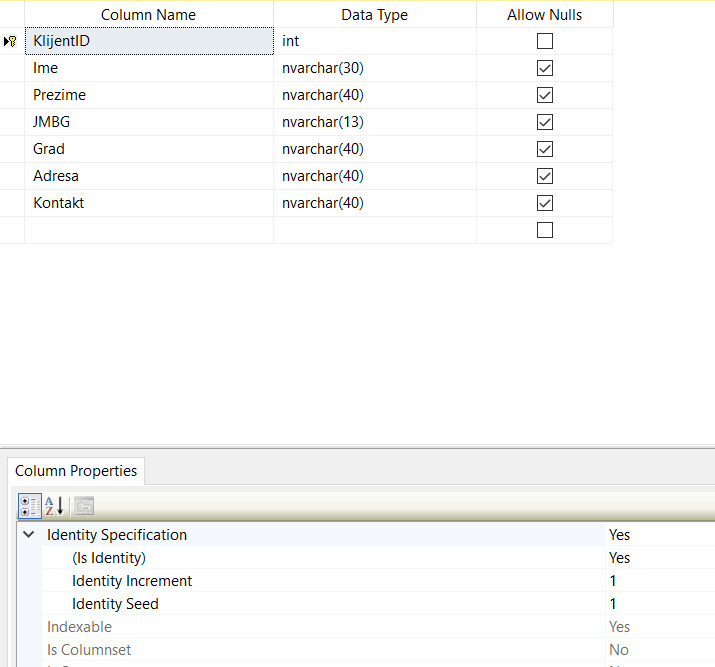
Слика 11. Приказ креираних обележја

Да би се колона *KlijentID* декларисала као примарни кључ, потребно је кликнути десним кликом тастера миша на црну срелицу која се налази поред назива ове колоне. Отвориће се листа у којој треба одабрати опцију *Set primary Key* (слика 12.). Овом командом се колона *KlijentID* декларише као примарни кључ.



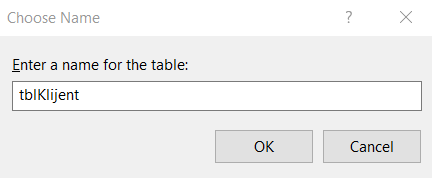
Слика 12. Дефинисање колоне KlijentID за примарни кључ

Да би систем за управљање базом података аутоматски креирао примарни кључ, потребно је у *Column Properties* пронаћи опцију *Identity Specification.* Проширивањем опције *Identity Specification* добија се опција *Is Identity,* коју треба променити на *Yes.* Овде је такође могуће подесити параметре *Identity Increment* и *Identity Seed. Identity Increment*-ом се дефинише за колико расте следећа вредност кључа, док *Identity Seed* означава од које вредности почињу да се меморишу кључеви (слика 13).



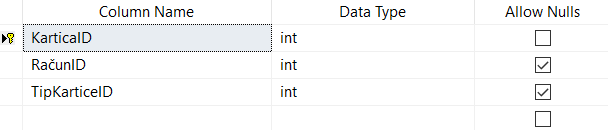
Слика 13. Подешавање аутоматске инкрементације вредности обележја примарног кључа

Чување табеле се врши притискањем тастера *Ctrl* и *S (Ctrl+S)*, након чега ће *SQL Server Managment Studio* захтевати унос назива табеле. Назив табеле се најчешће наводи са префиксом „*tbl*“ у овом случају *tblKlijent*.

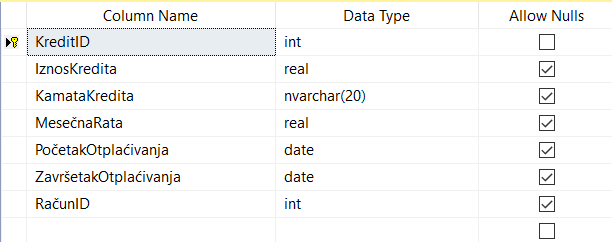


Слика 14. Чување креиране табеле

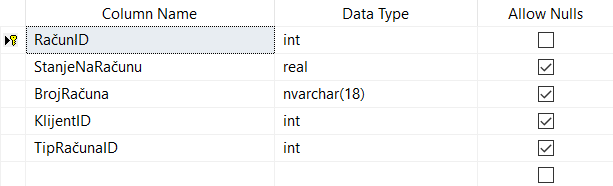
Исти процес би требало применити за креирање табела *tblKartica*, *tblKredit*, *tblRačun*, *tblTipKartice*, *tblTipRačuna*, *tblTransakcija* и *tblZaposleni* (слика 15-слика 21).



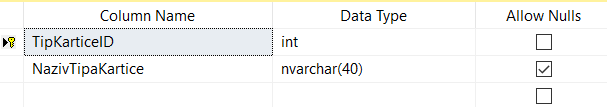
Слика 15. Креирање табеле tblKartica



Слика 16. Креирање табеле tblKredit



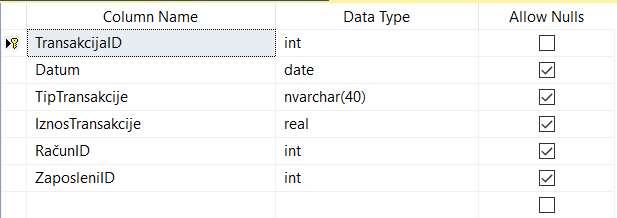
Слика 17. Креирање табеле tblRačuni



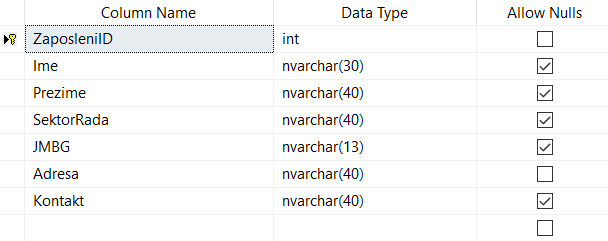
Слика 18. Креирање табеле tblTipKartice



Слика 19. Креирање табеле tblTipRačuna

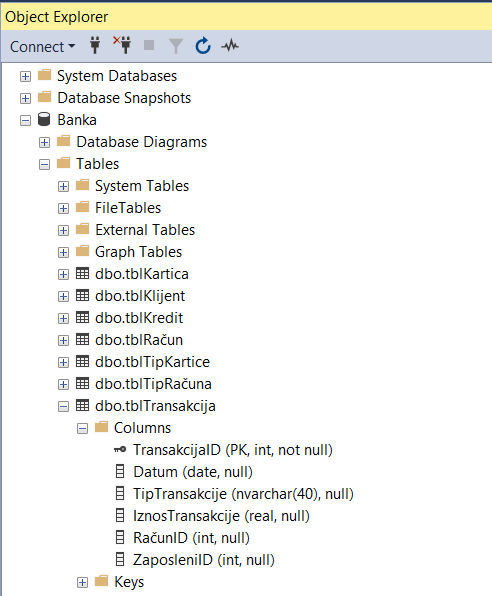


Слика 20. Креирање табеле tblTransakcija



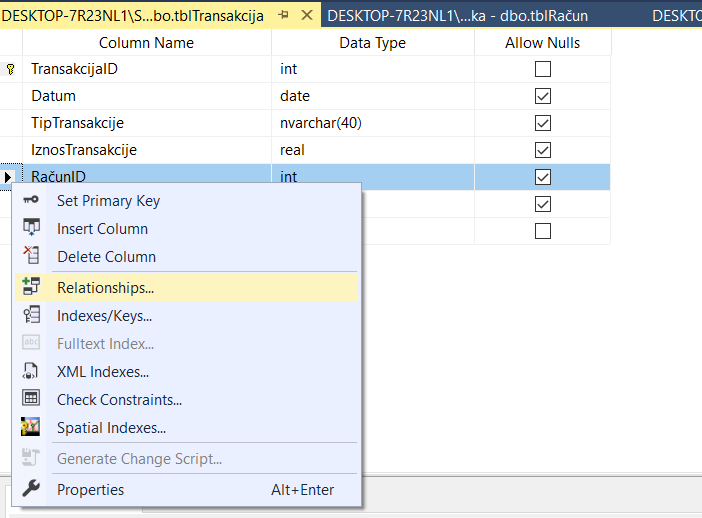
Слика 21. Креирање табеле tblZaposleni

Након тога би у *treeview*-у који се налази са леве стране прозора у оквиру *Object Explorer*-а, требала да се налази база података „Банка“, у којој ће, поред већ креиране табеле *tblKlijent* наћи и новокреиране табеле са одговарајућим атрибутима.



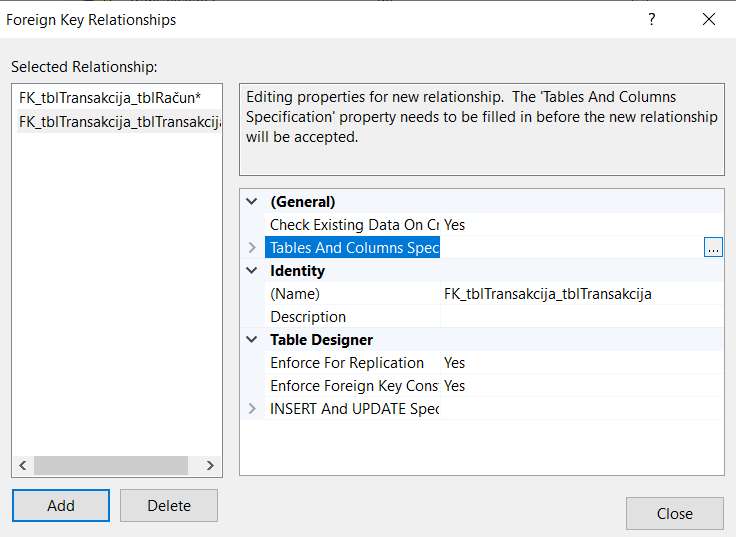
Слика 22. Object Explorer приказ креираних табела и колона

Као што се може видети на слици 22, у табели *tblTransakcija* се налазе два страна кључа (*RačuniID* и *ZaposleniID*). Приликом уноса атрибута у табелу *tblTransakcija* потребно је унети и атрибуте који имају називе *RačuniID* и *ZaposleniID*, а вредност *Data Type* за њих ставити на *int.* Затим је потребно кликнути десним кликом тастера миша на црну стрелицу која се налази поред назива ове колоне. Отвориће се листа у којој треба одабрати опцију *Relationships* (слика 23).



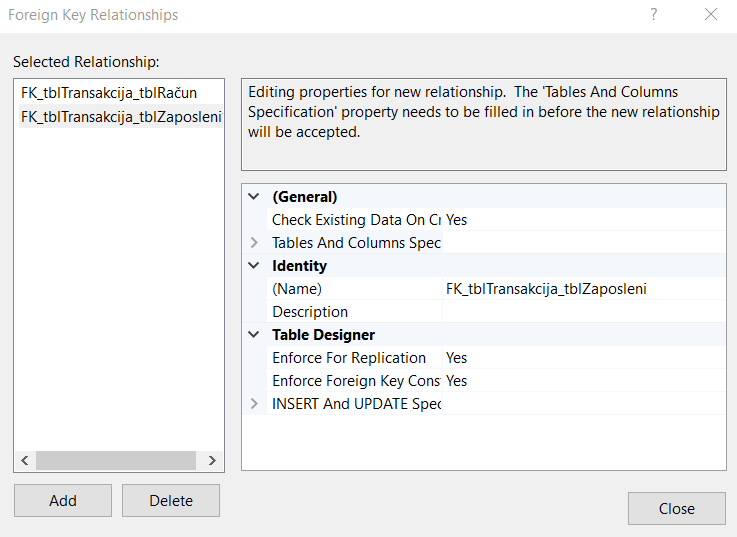
Слика 23. Додавање веза

Након одабира опције *Relationships,* отвара се прозор *Foreign key Relationships* (слика 24) у којем је потребно прво кликнути тастером миша на дугме *Add* а затим проширити поље *Tables and Columns Specification* и кликнути на три тачке које се налазе десно.



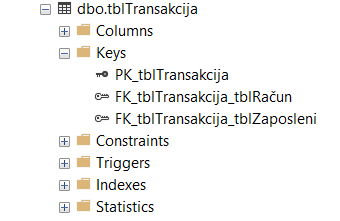
Слика 24. Додавање страног кључа

У отвореном прозору *Tables and Columns* потребно је у оквиру *Foreign key table,* конкретно из табеле *tblTransakcija,* из падајуће листе изабрати атрибут дефинисан приликом креирања табеле, а који је потребно повезати са другом табелом. Затим, у оквиру *Primary key table* треба изабрати табелу са којом се врши повезивање, и из падајуће листе одабрати кључ те табеле са којим се повезује жељени атрибут. Након што је све ово урађено, прозор треба да изгледа као на следећој слици (слика 25).



Слика 25. Креирање веза између табела

Додавањем *Relationships*-а атрибутима *RačuniID* и *ZaposleniID* у табели *tblTransakcija* они постају страни кључеви. Да ли је повезивање табеле успешно урађено знаће се провером постојања у *treeview*-у, у оквиру табеле *tblTransakcija,* под категоријом *Keys* (слика 26).



Слика 27. TreeView приказ креираних кључева

Исти поступак је потребно поновити за све остале табеле које у себи садрже стране кључеве који су примарни кључеви у другој табели, *tblRačun, tblKredit.*

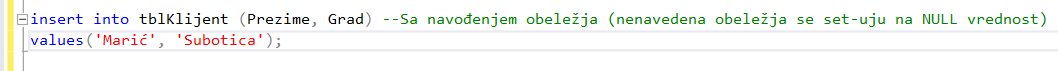
## 5.2. Упити и *DML* наредбе

Када је креирано све што је потребно, можемо извршити унос података. То можемо урадити на два начина који су приказани на слици 27 и слици 28. За пример додавања ћемо узети табелу без страних кључева, као на пример *tblKlijent.* Где уз помоћ речи *insert* специфирамо у коју табелу желимо да додамо податке, а уз реч *values* пишемо податке које желимо да унесемо. Овакав унос је унос без навођења обележја.

**

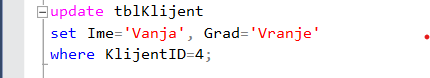
Слика 27. Додавање података у базу без навођења обележја

Додавање можемо извести и наводећи обележја. Пример таквог додавања је приказан на слици 28.



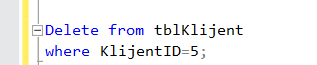
Слика 28. Додавање података у базу са навођењем обележја

Када је реч о измени постојећих података у бази, то вршимо уз помоћ кључне речи *update.* Представићемо унос у табелу клијената (слика 27). Уз реч *update* специфирамо коју табелу желимо да изменимо, *set* нам омогућава да тачно одређене податке ажурирамо. Док уз помоћ реч *where* мењамо податке тачно одређеног клијента. Подаци се могу изменити и за све колоне у табели тако што се не наведе где желимо тачно да променимо податке.



Слика 29. Измена података у бази

Брисање се врши уз помоћ речи *delete from* где специфирамо у којој табели желимо да обришемо податке. Затим наводећи реч *where* и уз њу тачну позицију колоне у табели коју желимо да обришемо. На слици 30 је наведен пример оваквог брисања.

**

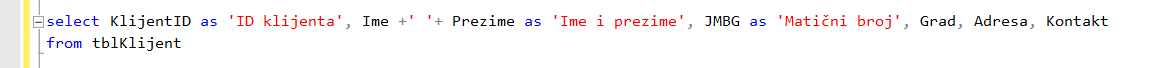
Слика 30. Брисање података из базе

Брисање се може извршити и без навођења речи *where* и специфирања тачно једне колоне. У том случају брише се садржај читаве табеле. На слици 31 је наведен пример таквог брисања.



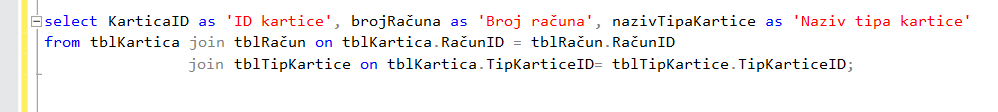
Слика 31. Брисање свих података из табеле у бази

Приказ садржаја из табеле врши се уз помоч речи *select* и онда наводимо обележја које желимо да прикажемо (слика 32), такође можемо да променимо име у приказаној табели, односно да у приказаној табели не пише име обележја из базе већ име које смо ми навели у *select* упиту (*select KlijentID as ‘ID klijenta’*). На овај начин *KlijentID,* име обележја из базе мењамо у име *‘ID klijenta’* које нам више одговара и које ће бити приказано у табели приликом покретања наредби.



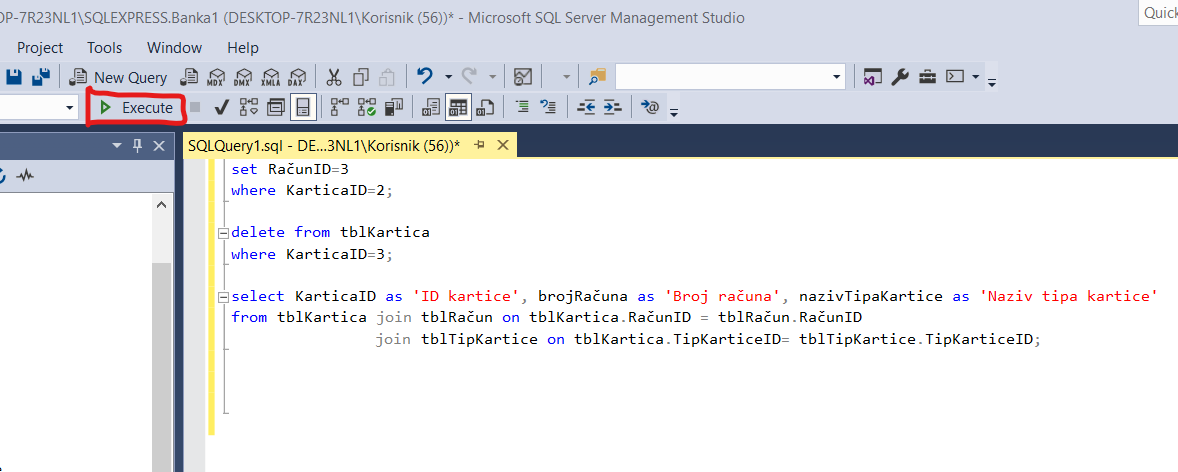
Слика 32. Приказ садржаја из табеле

Код табела које имају стране кључеве, како приликом приказа не бисмо приказивали само бројеве у табели, потребно је допунити *select* упит. Уз помоћ речи *join* уз коју наводимо име табеле из које долази страни кључ и специфирајући уз помоћ речи *on* обележја из класа преко којих се формира веза (*join tblRačun on tblKartica.RačunID=tblRačun.RačunID*) повезујемо табеле (ону у којој се налази примарни кључ и ону у којој се налази страни кључ), и на тај начин приликом приказа приказује стварне податке а не само бројеве. Пример овога можемо видети на слици 33.



Слика 33. Приказ обележја из табеле која има стране кључеве

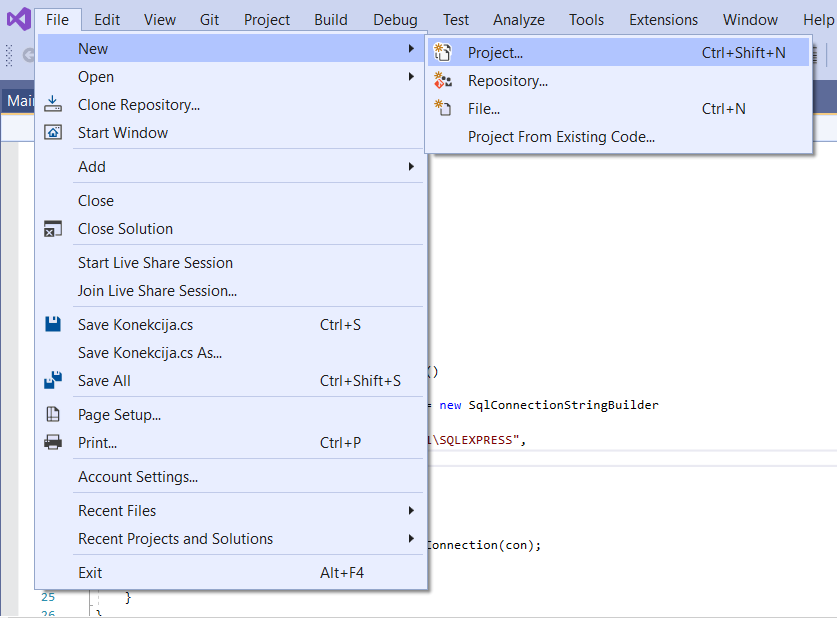
Када се испише наредба потребно је кликнути дугме *execute.* Место где се то дугме налази приказано је на слици 34.

**

Слика 34. Приказ места где се налази дугме execute

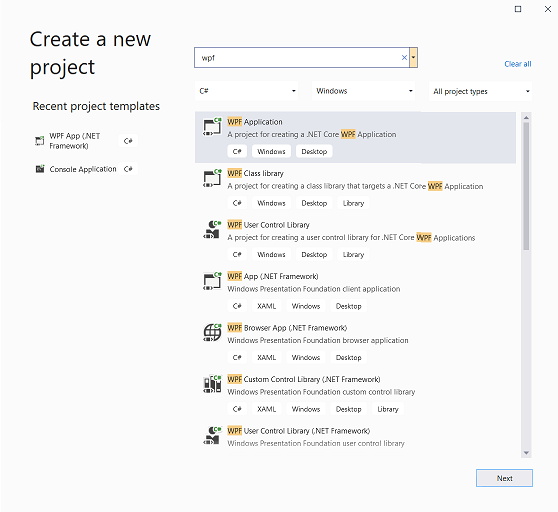
# 6. Креирање *WPF* апликације

У оквиру овог поглавља ће бити објашњено како се креира *WPF* апликација. Након покретања развојног окружења *Visual Studio*, потребно је тастером миша кликнути на *File menu*, а затим одабрати *New project* (слика 35).



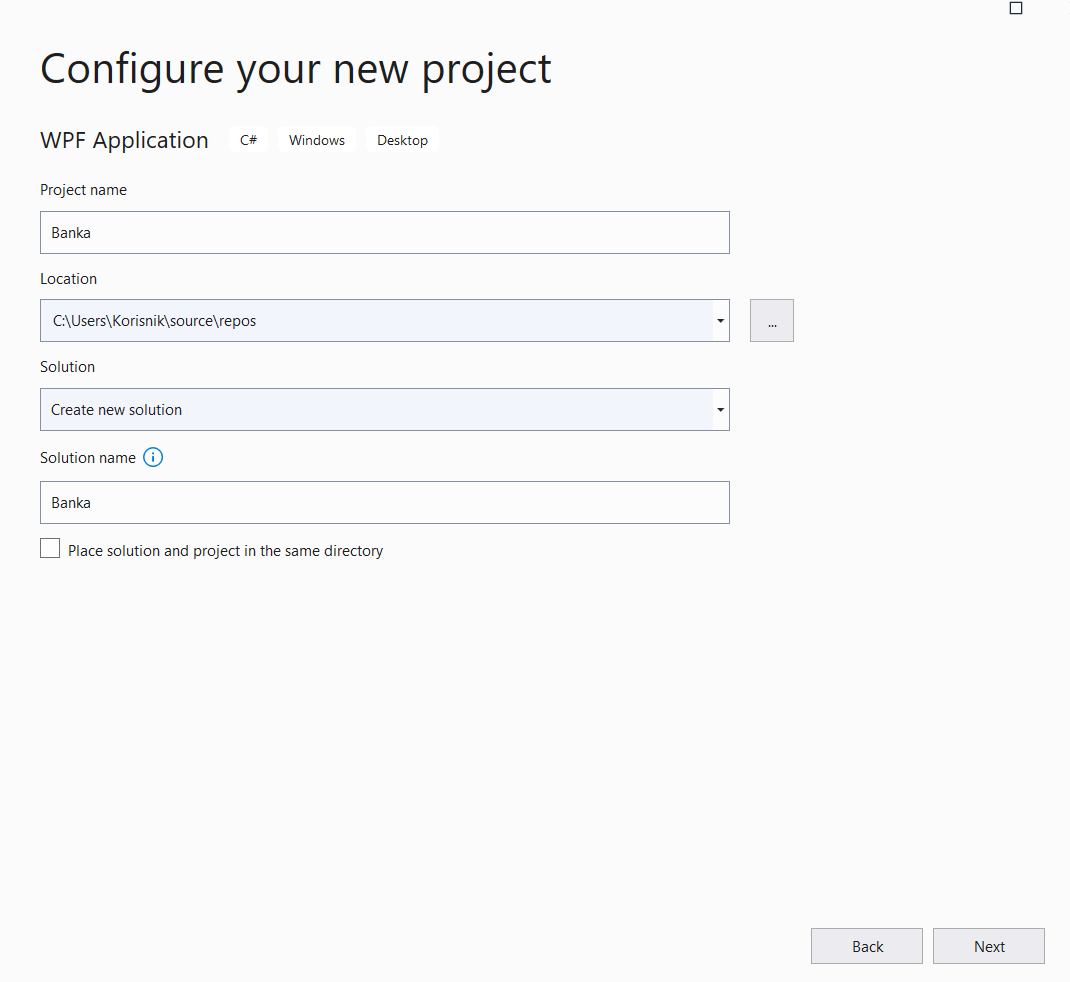
Слика 35. Креирање новог пројекта у Visual Studiо-u

Одабрати *Windows Classic Desktop*, затим *WPF Application*. Навести име апликације и кликнути *OK* (слика 36).



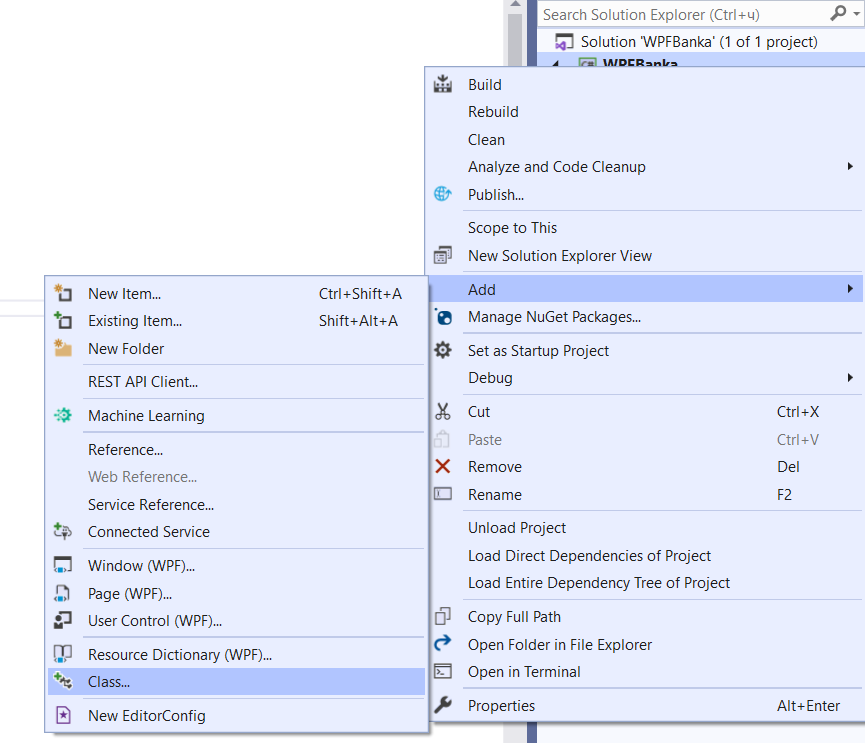
Слика 36. Креирање WPF апликације

Након тога потребно је дати назив апликацији у пољу *Project name* (слика 37).



Слика 37. Именовање пројекта

Следећи корак је креирање конекције у апликацији, ка бази податка. За почетак, креира се класа у којој ће се налазити метода путем које ће се остварити веза на претходно краирану базу. Класа се креира десним кликом тастера миша на назив пројекта (*WPF Banka*), а затим *Add/Class* (слика 38).

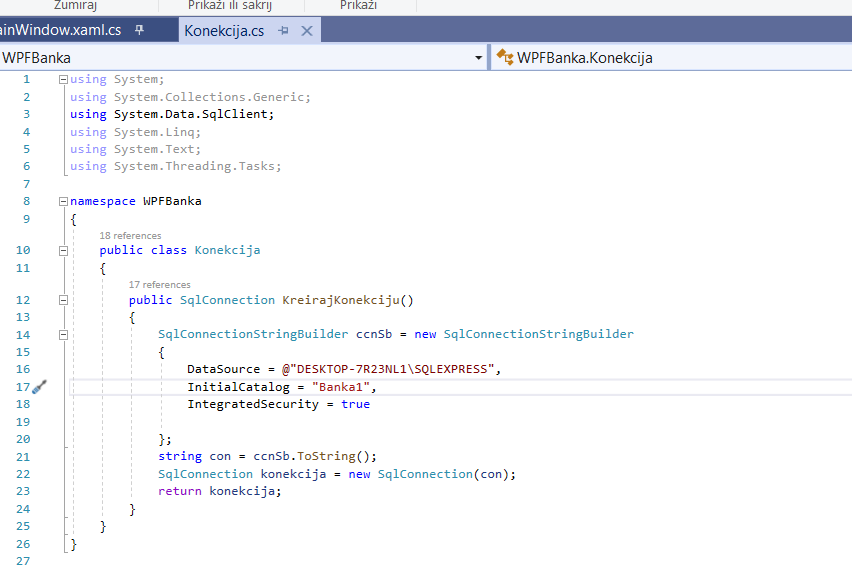


Слика 38. Креирање класе

Новокреираној класи се додељује назив „Конекција“ и њен садржај приказан је на слици 39. На самом почетку потребно је укључити именски простор *System.Data.SqlClient*. Тело класе садржи само једну методу под називом „КреирајКонекцију“, која ће се у каснијем развоју апликације позвати по потреби, односно сваки пут када је потребно остварити везу са базом података, стога је повратни тип ове методе инстанца класе *SqlConnection.* Унутар саме методе приказана су подешавања параметара конекције и то:

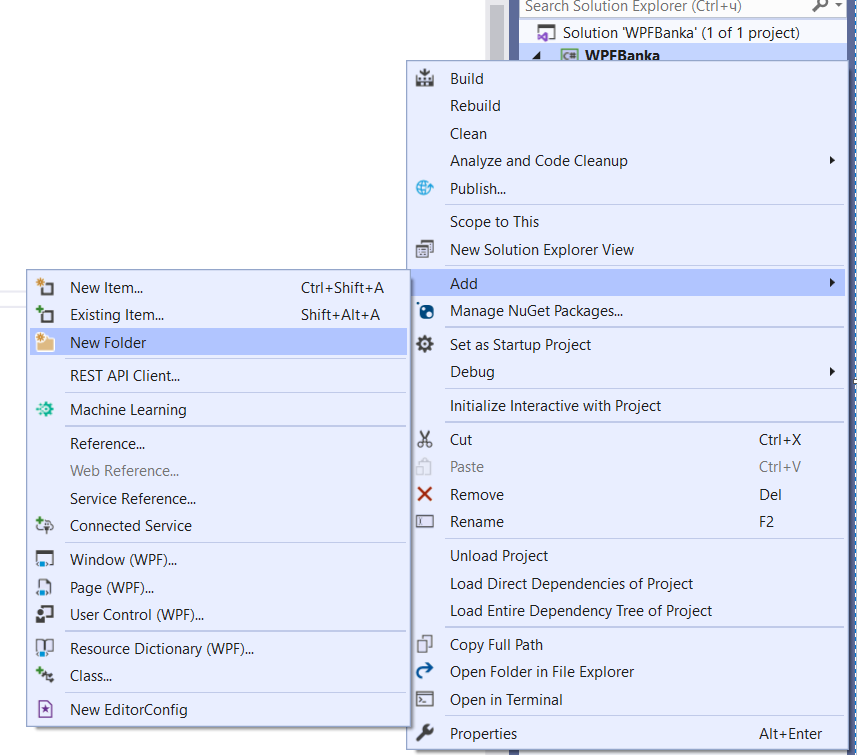
1. *DataSource* – назив сервера на којем је база података смештена,
2. *InitialCatalog* – назив базе података којој је потребно приступити,
3. *IntegratedSecurity* – уколико се база налази на локалној машини, овај параметар поставља се на *true.*

Након подешавања ових параметара, цео *string* се прослеђује инстанци класе *SqlConnection*.



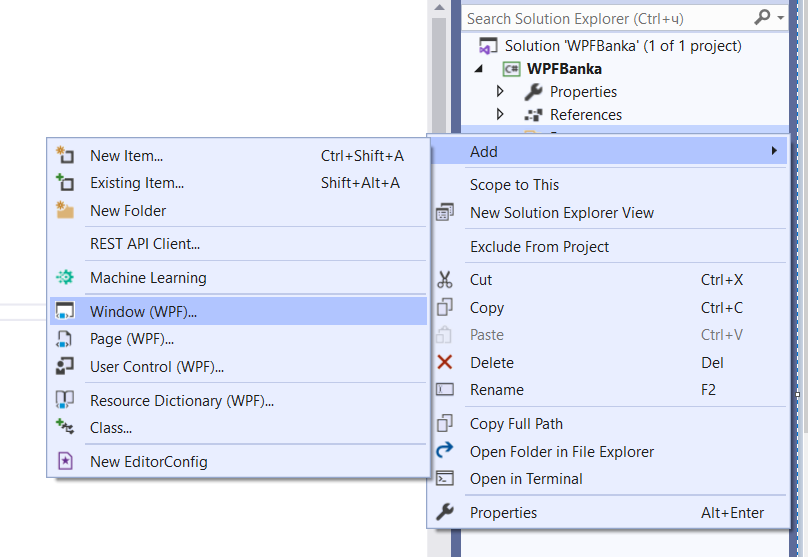
Слика 39. Класа конекција

Следећи корак у изради апликације је креирање екранских форми (енг. *Forms*) – прозора, за додавање објеката у базу података. Како би се све форме биле уредно смештене на једном месту, потребно је креирати фолдер по називом „Форме“ у ком ће се налазити свих осам форми за претходно креираних осам табела у бази података. Креирање фолдера врши се на сличан начин као креирање класе, десним кликом тастера миша на пројекат, а затим *Add/New Folder* (слика 40).



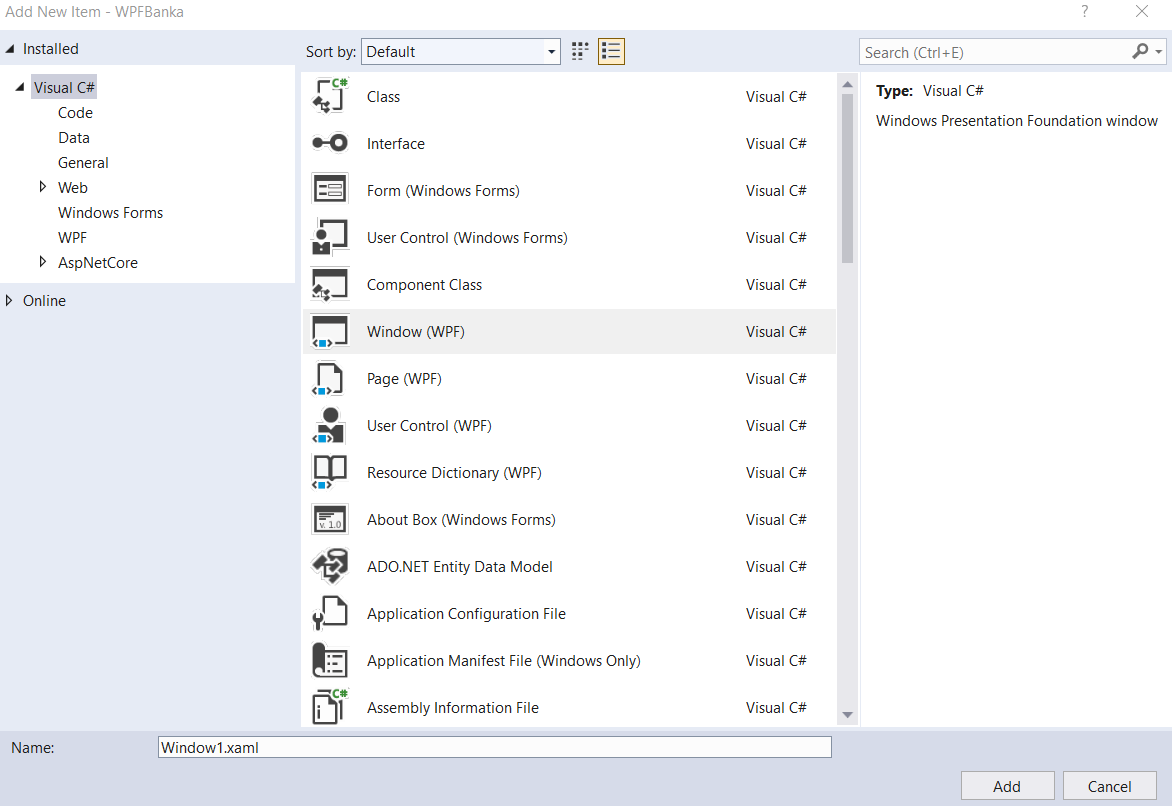
Слика 40. Креирање новог фолдера

Након креирања фолдера, потребно је додати нови прозор за сваку о форми. Прозор, унутар фолдера креира се деним кликом тастера миша на фолдер, а затим *Add/Window* (слика 41).



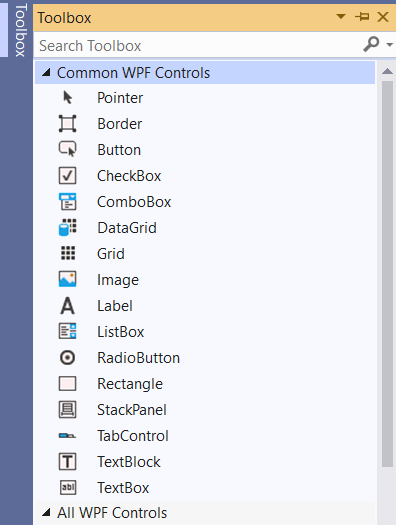
Слика 41. Додавање форми унутар фолдера

Након тога, приказаће се прозор за додавање новог *Item-*а унутар ког је потребно подесити назив и затим клинкнути на *Add* (слика 42).



Слика 42. Додавање новог прозора

Како је нови прозор изгенерисан, потребно је, унутар њега, додати одређене елементе путемо којих ће се остварити потребне функционалности. На левој страни прозора, кликом тастера на *Toolbox*, приказаће се листа свих уграђених *WPF* команди (слика 43). Једноставним превлачењем тих команди на централни прозор, изгенерисаће се код у *XAML*  декларативном језику унутар којег ће се вршити детаљна подешавања свих команди унутар прозора.

а

Слика 43 Додавање команди из Toolbox-а

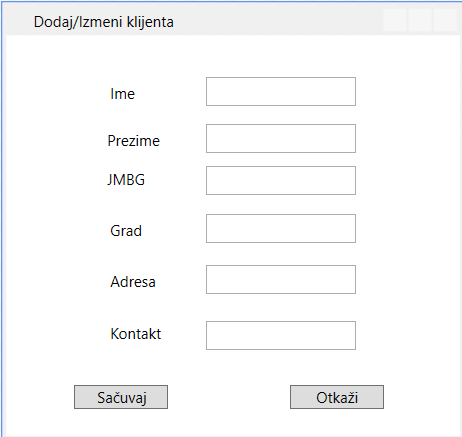
## 6.1. Додавање

Форма за додавање новог клијента у базу података, треба да садржи елементе приказане на слици 44. То су шест основних *TextBox*-ова унутар којих ће бити уписане вредности обележја. Такође уз сваки *TextBox* мора постојати назив (*Label*) који ће описивати његову намену. На крају, свака форма мора имати дугме „Сачувај“, на чији клик тастером миша ће се вредности уписивати у бази података и дугме „Откажи“ које кориснику омогућава затварање форме. Ове функционалности дугмића постижу се двоструким кликом тастера миша на одговарајуће дугме у оквиру форме, чиме се у *Code behind*-у генеришу методе *BtnSačuvaj\_Click* и *BtnOtkaži\_Click*.



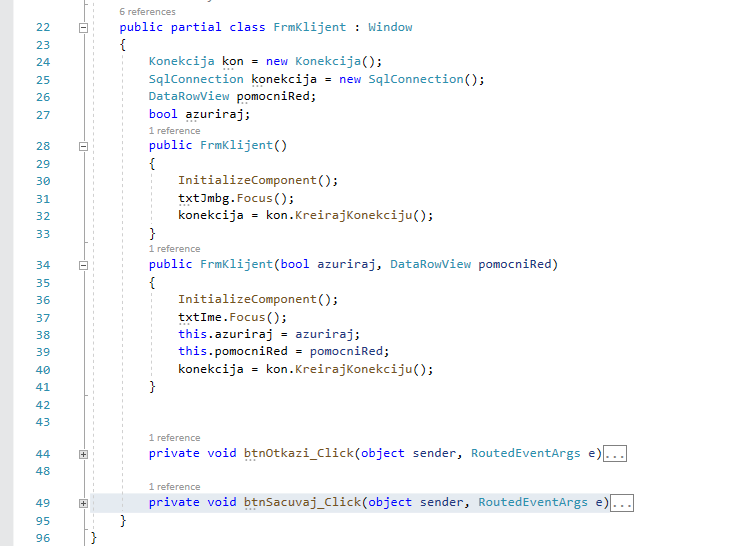
Слика 44. XAML код код форме „Клијент“

Визуелни приказ форме за унос новог клијента, изгледаће као на слици 45.



Слика 45. Изглед фомре „Клијент“

У коду ове форме, требало би да се налази следећа пословна логика (слика 46). Најпре је потребно укључити именски простор *System.Data.SqlClient*. Затим је потребно направити нови објекат класе *SqlConnection* класе, путем које ће се остварити веза ка преходно креираној бази података, након клика тастером миша на дугме „Сачувај“. Такође, у конструктору овог прозора пожељно је подесити *Focus()*, која позиционира курсор на прво поље које корисник попуњава при покретању форме. У овом слуцају то је поље ѕа унос матичног броја.



Слика 46. Садржај CodeBehind-а форме за додавање клијента

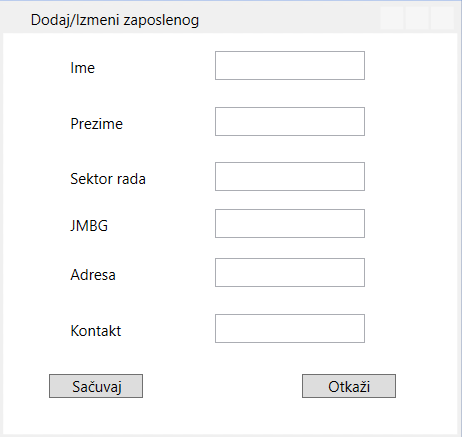
Овакав приказ позадинског кода је идентичан за сваку од осам креираних форми. Оно у чему је свака од ових форми јединствена јесте позадинска функционалност која се дешава у тренутку када корисник кликне на дугме „Сачувај“. На слици 47 је приказан код изгенерисане методе *BtnSačuvaj\_Click.* На самом почетку потребно је отворити конекцију ка бази података. Затим, креира се *string* који у себи садржи *insert into* наредбу путем које се додаје нови објекат у базу података. У одељак *values* ове наредбе, прослеђују се вредности које су унете у *TextBox*-ове ове форме. Потребно је водити рачуна да се вредности *string*-ова ограде апострофима. Након успешно нарављене *insert* команде, овај *string* се прослеђује објекту класе *SqlCommand*. Овај објекат као параметре прима претходно креирани *string*, али и објекат класе *SqlConnection* путем којег ће знати у коју базу података треба да упише прослеђени запис. Након тога, команда се извршава путем методе *ExecuteNonQuerry()*  након чега се прозор затвара. Овај део кода потребно је оградити *try* блоком из разлога што постоји могућност настанка грешке услед прослеђивања уноса погресног типа података од стране корисника. Потребно је додати *catch* блок унутар ког ће се „ухватити“ изузетак. На крају у *finally* блоку, потребно је постарати се да се претходно отворена конекција сигурно завтвори. Уколико конекција остане отворена, нико други неће моћи да приступи бази података на исти начин док се ова инстанца конекција не затвори.



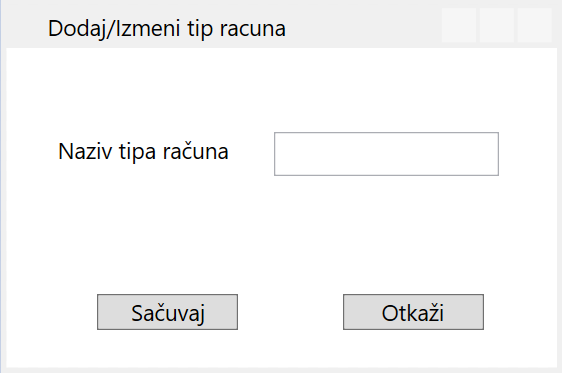
Слика 47. Садршај методе BtnSačuvaj\_Click за форму „Клијент“

У наставку ће бити приказан излед и осталих осам форми.

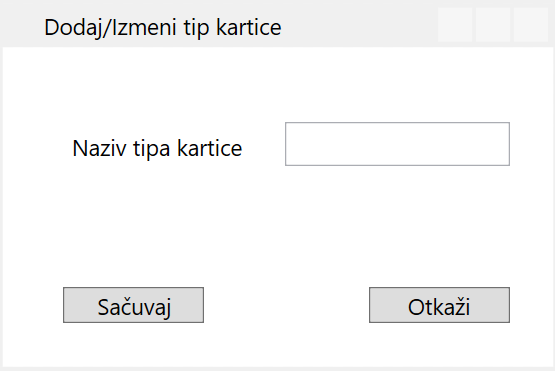
Форме за додавање запосленог, типа рачуна и типа картице имају веома сличан дизајн, а слични су и *XAML* кодови ових форми (слика 48,49 и 50).



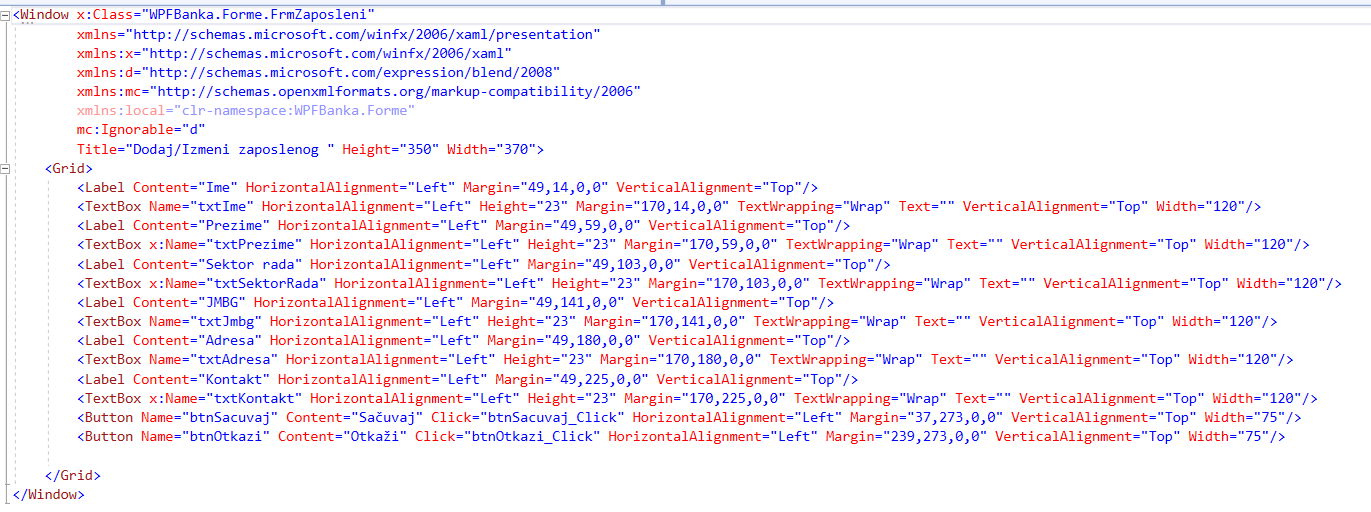
Слика 48. Изглед форме „Запослени“



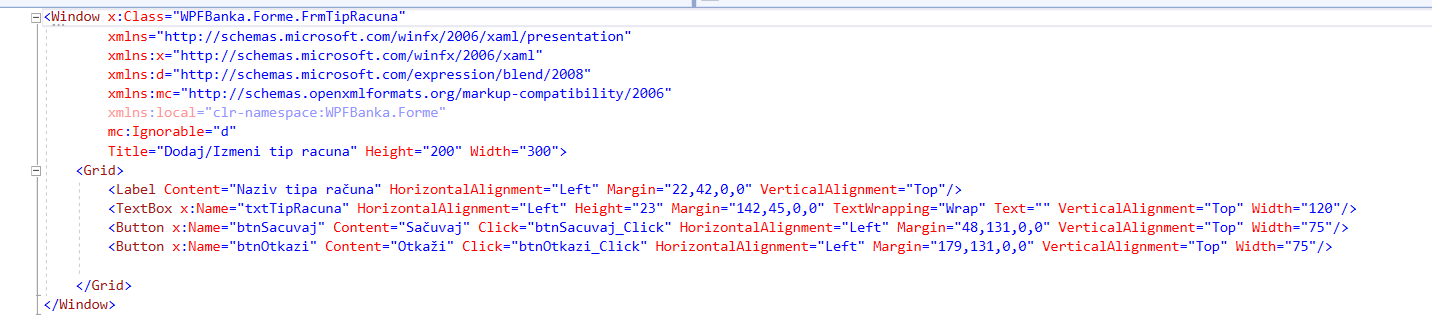
Слика 49. Изглед форме „Тип рачуна“



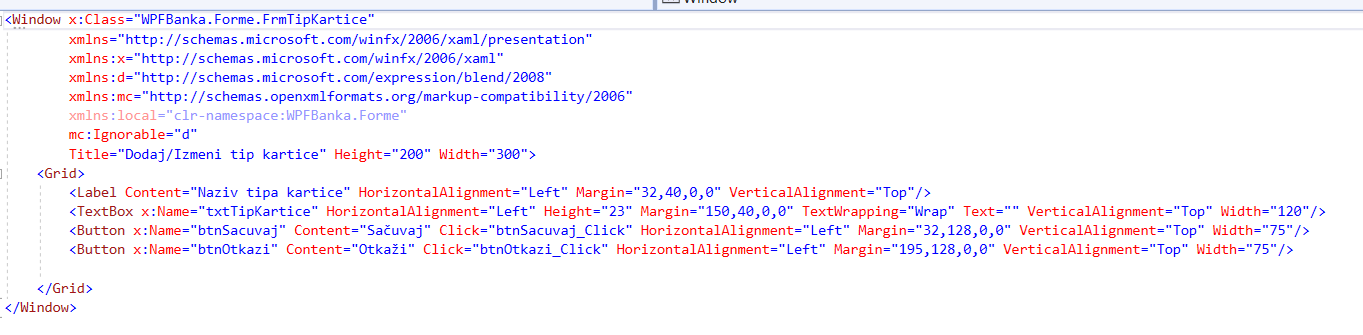
Слика 50. Изглед форме „Тип картице“



Слика 51. XAML код форме „Запослени“



Слика 52. XAML код форме „Тип рачуна“



Слика 53. XAML код форме „Тип картице“

Разлика се поново угледа у *insert* упиту смештеном унутар методе „Сачувај“.



Слика 54. Садршај методе BtnSačuvaj\_Click за форму „Запослени“

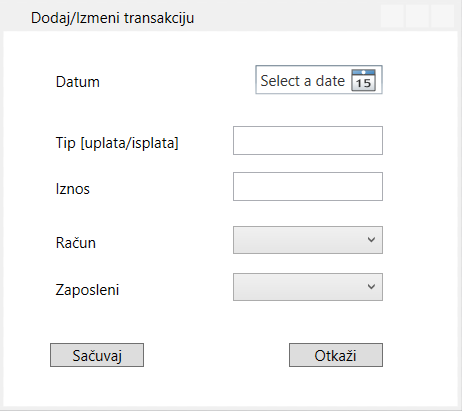


Слика 55. Садршај методе BtnSačuvaj\_Click за форму „Тип рачуна“



Слика 56. Садршај методе BtnSačuvaj\_Click за форму „Тип картице“

Остале табеле као што су *tblTransakcija, tblKredit, tblRačun* и *tblKartica* имају стране кључеве других табела на основу којих се приступа садржајима тих табела. У самој апликацији потребно је кориснику обезбеити визуелни приказ обележја. То се постиже коришћењем *ComboBox*-ова. Визуелни приказ форме за додавање нове трансакције је приказан на слици 57.



Слика 57 Изглед форме „Запослени“

*XAML* код ове форме приказан је на слици 58. Приликом креирања новог *ComboBox*-а потребно је да се дефинише *DisplayMemberPath* који представља обележје повезане табеле које ће се приказати у падајућој листи када корисник клинке на тај *ComboBox*, као и *SelectedValuePath* који представља обележје примарног кључа на основу ког ће се унета вредност из приказаног *ComboBox*-а повезати са својим идентификационом обележјем и као таква проследити у базу података.



Слика 58. XAML код форме „Трансакција“

*CodeBehind* форми које у себи садрже *ComboBox-*ове разликоваће се од осталих.



Слика 59. Садржај CodeBehind-а форме „Трансакција“

Имамо методу „Попуни падајуће листе“ која се позива у контруктору ове класе, с обзиром на то да се, при инстанцирању новог порозора ове форме, подаци који че бити смештени у *ComboBox*-овима мораају негде привремено ускладиштити. За постизање тога, користи се објекат *DataTable* класе који представља копију табеле из базе података сачувану у меморији апликације. Објекат класе *DataTable* се попуњава помоћу објекта класе *SqlDataAdapter* којем се прослеђује одговарајући *select* упит као и конекција ка одговарајућој бази података. Затим се тај *DataTable* поствља као *ItemsSource* за одговарајући *ComboBox*. На самом крају у *finally*  блоку потребно је обезбедити да се конекција ка бази података безбедно затвори.



Слика 60. Садржај методе „Попуни падајуће листе“

Садржај методе која се активира на клик дугмета „Сачувај“ је слична као и у претходним корацима. Вредности из *ComboBox*-ова преузима се уз помоћ вредности *SelectedValue*. Израбрана вредност датума из команде *DataPicker* убацује у *select* команду помоћу *SelectedDate*, неопходно је конвертовати датум, због разлике у типу података *Date* у бази података и типа података *DataTime* у овој апликацији.

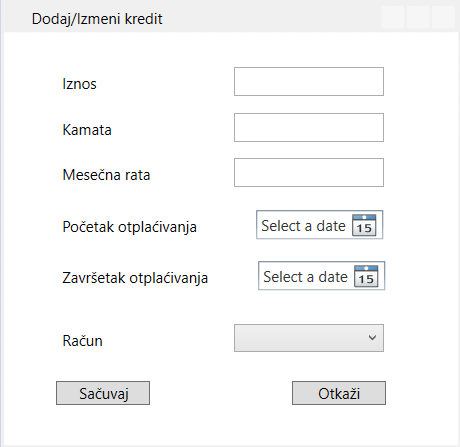


Слика 61. Садршај методе BtnSačuvaj\_Click за форму „Трансакција“ – први део



Слика 62. Садршај методе BtnSačuvaj\_Click за форму „Трансакција“ – други део

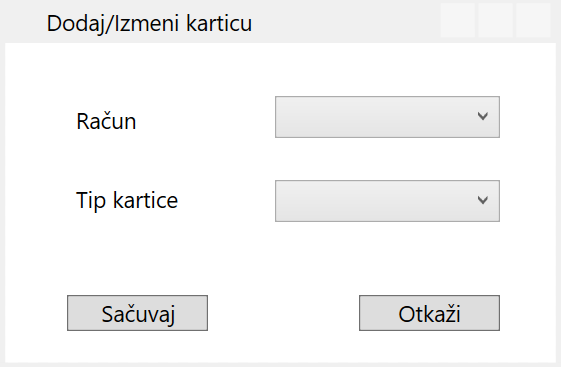
Преостале три форме функционису на исти начин, тако да ћемо приказати изглед форми, *XAML* код и методу *BtnSačuvaj\_Click*, метода за попуњавање падајућих листа је такође слична.



Слика 63. Излед форме „Кредит“



Слика 64. Изглед форме „Рачун“



Слика 65. Изглед форме „Картица“

Сада ћемо приказати *XAML* кодове ове три форме.



Слика 66. XAML код форме „Кредит“



Слика 67. XAML код форме „Рачун“



Слика 68. XAML код форме „Картица“

Мале разлике између ових форми можемо приметити и у *BtnSačuvaj\_Click*, чији садржај ћемо сада приказати.



Слика 69. Садршај методе BtnSačuvaj\_Click за форму „Кредит“

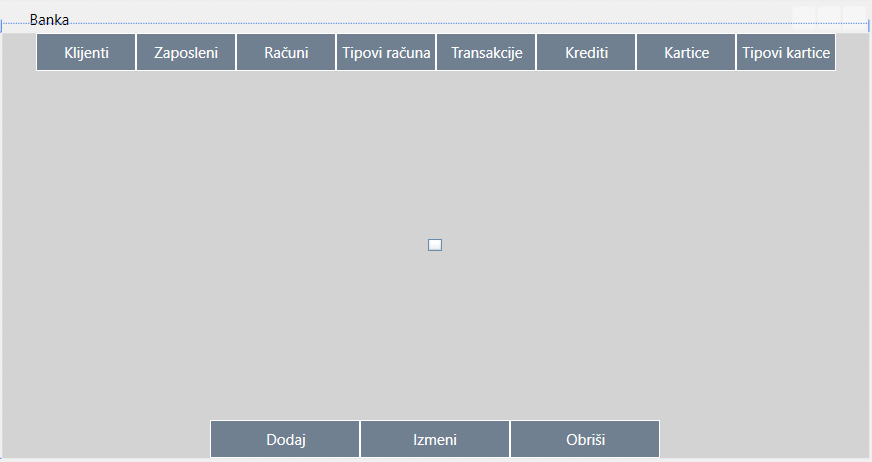


Слика 70. Садршај методе BtnSačuvaj\_Click за форму „Рачун“



Слика 71. Садршај методе BtnSačuvaj\_Click за форму „Картица“

Након креирања форми, потебно је исто урадити и за главни прозор (енгл. *MainWindow*). Предтавља први контакт корисника са апликацијом помоћу ког ће корисник остваривати интеракцију. На њему се налази дугме за сваку од форми, као и дугме за додавање, брисање и измену. Кликом тастера миша на неко дугме на средини форме ти подаци ће бити приказани. Приказ форме дат је на слици 72.



Слика 72. Излед главног прозора апликације

У наставку ће бити приказан *XAML* код главног прозора. Како би елементи били правилно распоређени смештају се у *StackPanel*. Поравњања ће бити приказана у оквиру слике *XAML* кода. Такође подешене су и боје главног проозора и форми уз помоћ *Foregraund* и *Background* метода.



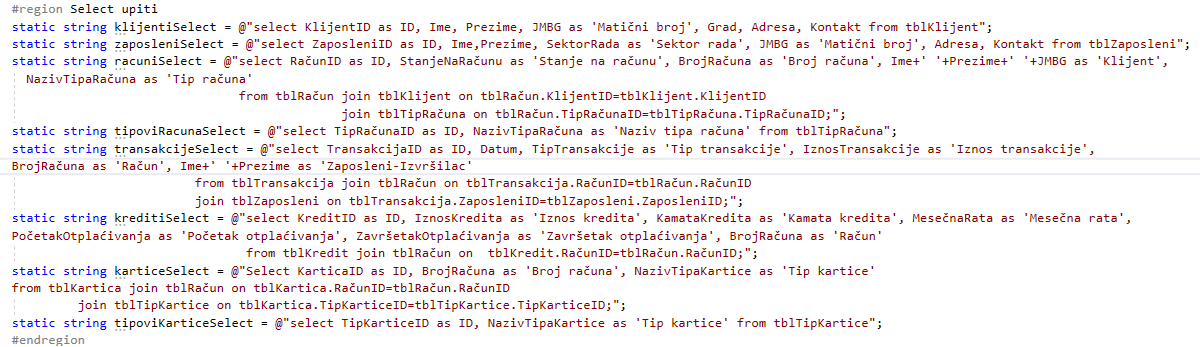
Слика 73. Излед XAML кода главног прозора

У наредним корацима биће приказан излед *CodeBehind*-а. Као прво потребно је да се увезу у именски простор форме. У оквиру њега налазе се упити који су неопходни да би се обављале одговарајуће функционалности. Код ће бити приказан на слици 74.

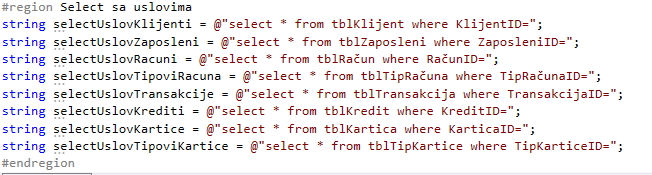


Слика 74. Излед CodeBehind-а главног прозора

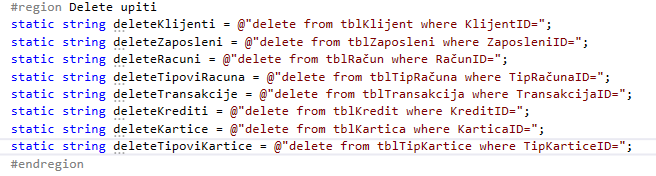
Упити за испит података у главном прозору, брисање и измену података биће приказани на наредне три слике.



Слика 75. Insert упити



Слика 76. Упити за измену података



Слика 77. Упити за брисање података

Након тога потребно је креирати методу која ће омогућити попуњавање почетног *Grid*-а. Након отврања конекције, упит се креира на основу повезивања осам таблеа из базе података. Затим се на исти начин врши попуњавање *ComboBox*-ова при инстанцирању прозора неке од форми, ови подаци се смештају у *DataSet* којег затим попуњава *SqlDataAdapter*, када се метода позива прослеђује се и *Select* упит.



Слика 78. Садржај методе „Учитај податке“

Сада је потребно обезбедити да кликом на дугме прикаже садржај табеле из базе података. На основу креиране методе „Учитај податке“ и *select* упита позива се управо ова метода и прослеђује јој се одговрајући упит.



Слика 79. Садржаји метода за учитавање података из базе

Метода за додавање је врло једноставна. Кликом тастера на „Додај“ инстанцира се прозор одговарајуће форме и приказује приказ на екрану. Након затврања тог прозора освежава се централни грид са актуелним подацима након евентуалног додавања у базу. На слици 80 је приказан садржај те методе.



Слика 80. Садржај методе BtnDodaj­­­\_Click

## 6.2. Измена

Потрено је да постоји одговарајућа форма у којој ће корисник мењати податке. Како би се избегло креирање нове специјално за модификацију података, може се искористити постојећа форма за додавање. Потебно је кликнути на дугме „Сачувај“ да би се покренуо поступак. Прва је типа података *boolean* и користиће се као услов који ће дугмету „Сачувај“ говорити да ли је потребно податке уписати у базу података као нови ентитет или изменити. Друга је типа *DataRowView* и служи за памћење вредности означеног реда, када из главног прозора пређемо у неку другу форму. Трећа променљива, типа података *string* која говори која је тренутно табела учитана.



Слика 81. Садржај методе BtnIzmeni\_Click

Метода за измену података приказан је на слици 82 и 83. Декларисана је променљива ажурирај на *true.* Након тога инстанцира се прозор нјене форме и отвара конекција. Помоћу *DataRowView* објекта, као и код брисања, памти се ред који је означен. Помоћу *select* упита којем се у *where* услову задаје вредност примарног кључа селектованог реда. Потребни подци се смештају у одговарајућа поља форме, помоћу *SqlDataReader*-а. Након тога се приказује прозор, даља функционалност се обавља се унутар те форме. У *finally* блоку затвара се конекција, а променљива ажурирај се враћа на *false* у случају да корисник жели да дода нови записе у базу у следећем кораку.



Слика 82. Садржај методе „ПопуниФорму“ – први део



Слика 83. Садржај методе „ПопуниФорму“ – други део

## 6.3. Брисање

Бирсање података остварује се кликом на дугме „Обриши“. Након отварања конекције, креира се објекат класе *DataRowView* који у себи чува информацију о индексу реда који је корисник претходно означио. Након тога креира се *Delete SQL* команда путем које ће се у самој бази обрисати жељена ствар. Пре брисања кориснику се приказује дијалог да потврди да ли жели да обрише селектовану ствар. Два основа изузетка која се могу јавити су *ArgumentOutOfRangeException* и *SqlException*. Прво се јавља при покушају брисања уколико није означен ниједан ред а притиснуто је „Обриши“. А други уколико се покуша брисање ентитета из табеле који је примарним кључем побезан са неком другом табелом (слика 84). Када је тастером миша кликнуто на дугме „Обриши“, у зависности од тога која табела је тренутно учитана и коју ставку табеле је корисник изабрао. Методи „ОбришиЗапис“ се прослеђују параметри и накомн тога се поново учитавају подаци из базе података, како би они били ажурирани након брисања означеног реда (слика 85).



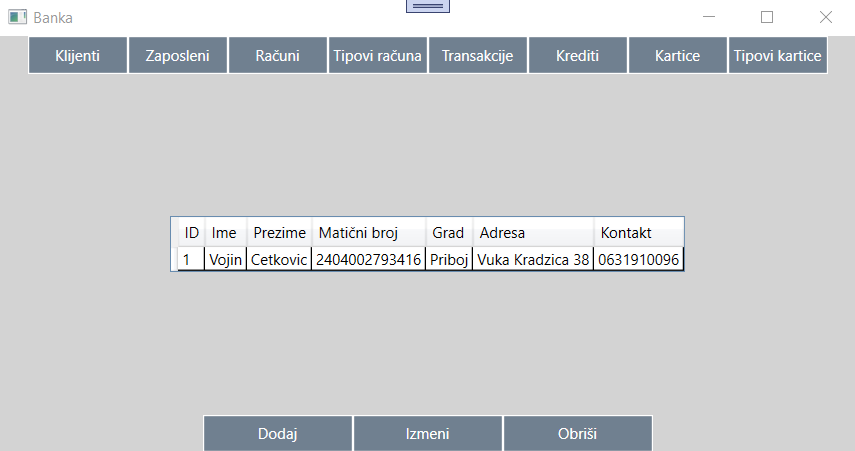
Слика 84. Садрзај методе „ОбришиЗапис“



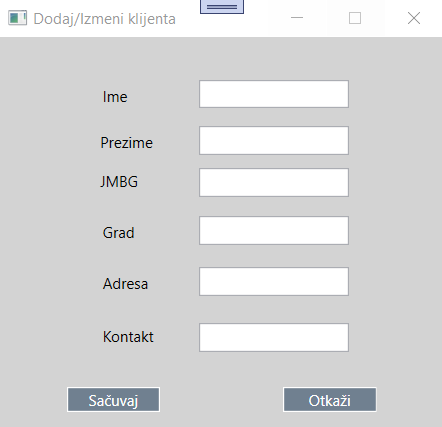
Слика 85. Садржај методе BtnObriši\_Click

## 6.4. Изглед апликације

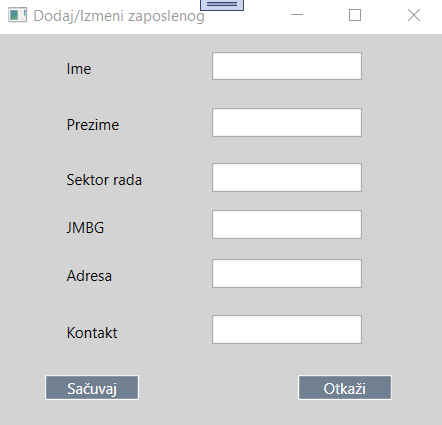
На наредним сликама биће приказан изглед форми апликације.



Слика 86. Изглед главног прозора



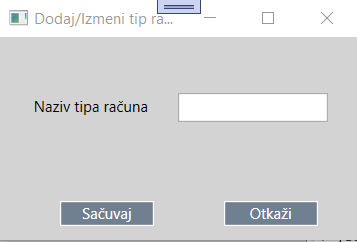
Слика 87. Изглед форме „Клијент“



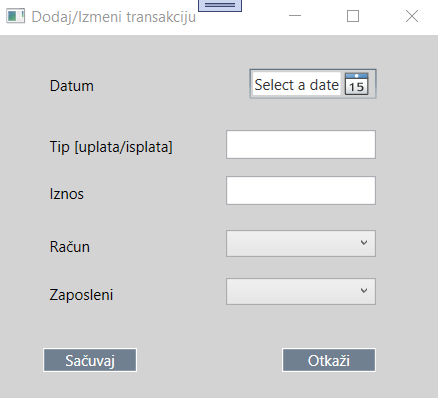
Слика 88. Изглед форме „Запослени“



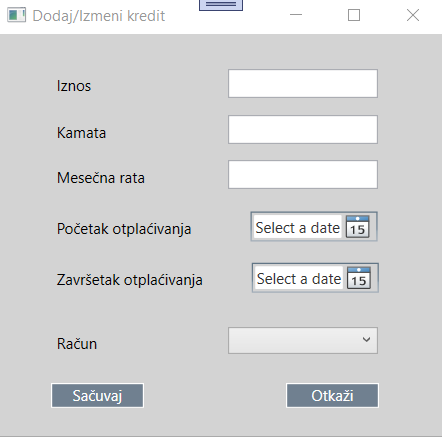
Слика 89. Изглед форме „Рачун“



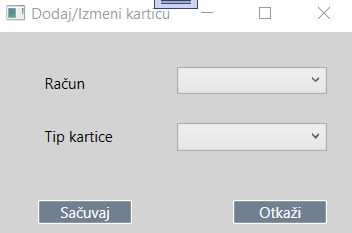
Слика 90. Изглед форме „Тип рачуна“



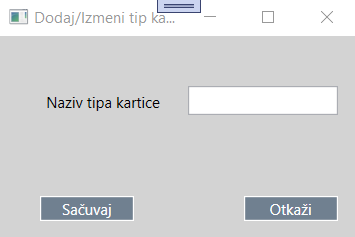
Слика 91. Изглед форме „Трансакција“



Слика 92. Изглед форме „Кредит“



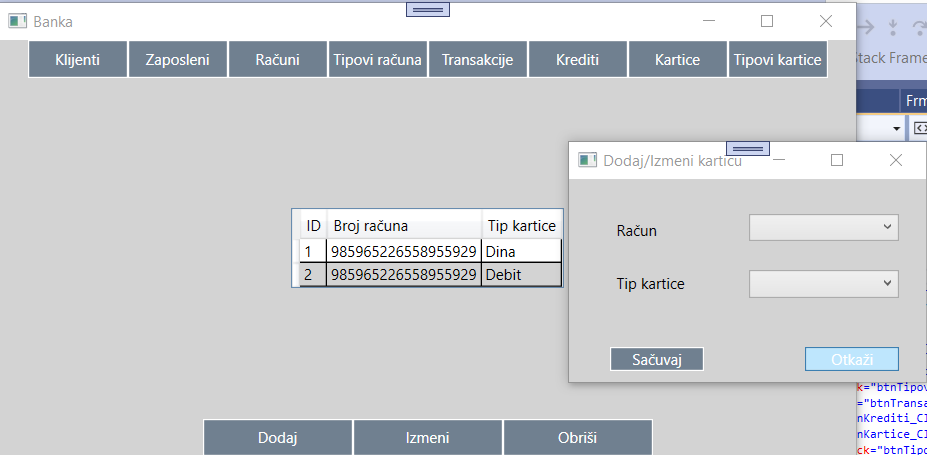
Слика 93. Игллед форме „Картица“



Слика 94. Изглед форме „Тип картице“

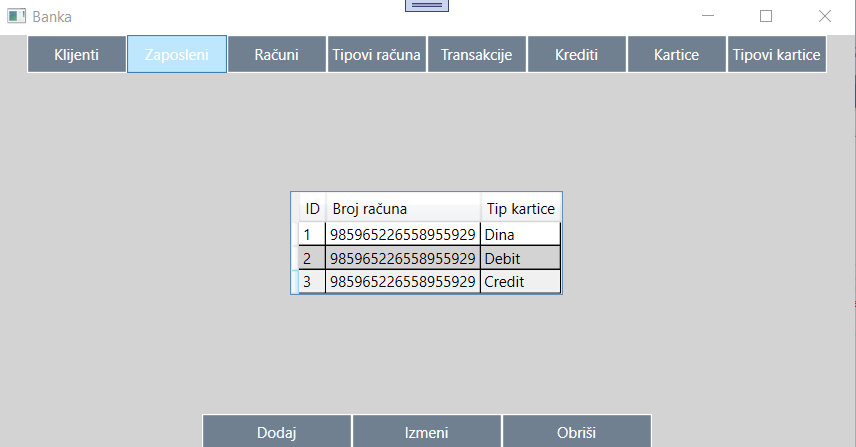
# 7. Тестирање апликације

Кликом на дугме „Додај“ отвара се дијалог за упис података у базу. Кликом на неко од дугмића у *StackPanel*-у у центру прозора приказују се подаци табеле из базе података.



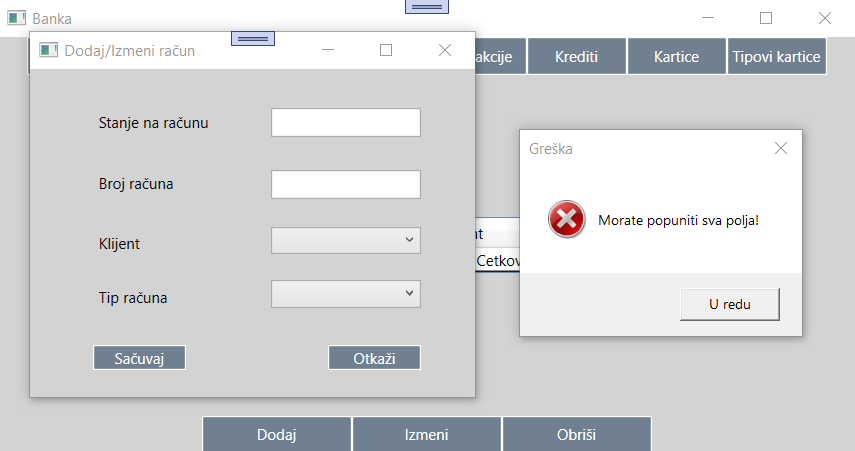
Слика 94. Додавање нове картице у базу података

Када се кликне на „Сачувај“ нова картица биће приказана у табели, а уколико се кликне на „Откажи“ онда се ништа неће чувати, већ само затворити дијалог.



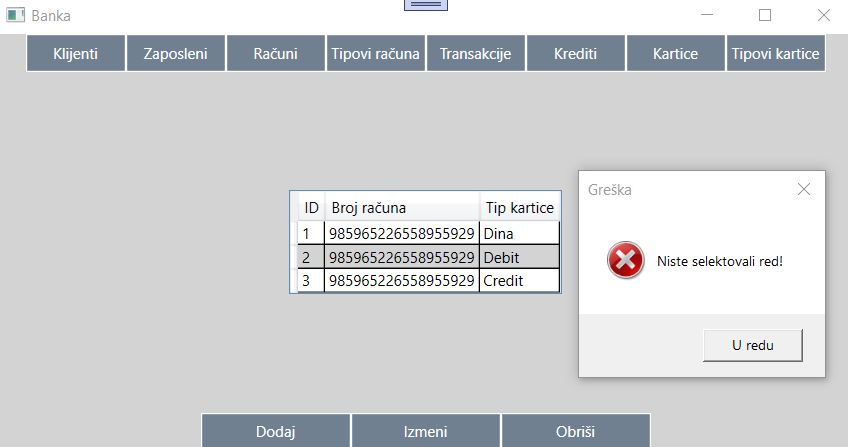
Слика 95. Табела након додавања нове картице

Уколико нисмо попунили неко од поља изаћи ће грешка. Што је приказано на слици 96.



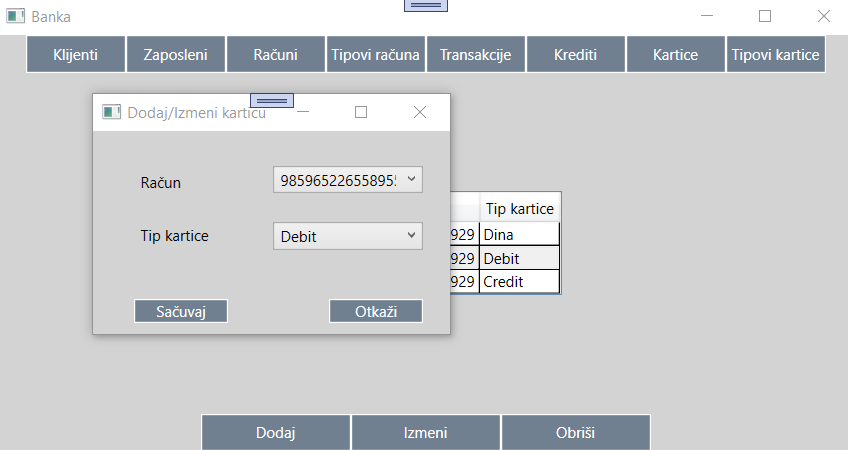
Слика 96. Грешка при уносу података

Кликом на дугме „Обриши“ или „Измени“, а да притом није означен ниједан ред, узрокује приказивање следеће поруке.

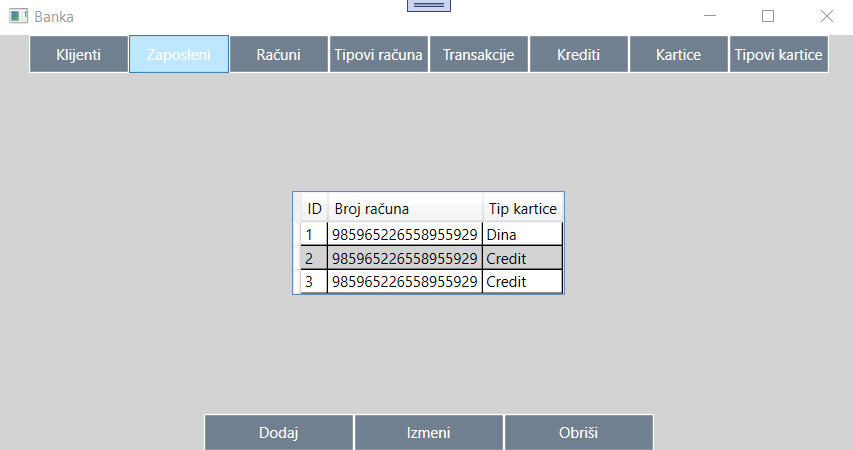


Слика 97. Покушај клика на дугме „Измени“ или „Обриши“ без селектовања реда

Уколико се селектује ред и кликне на дугме „Измени“, доћи ће до отварања дијалога и уколико се кликне „Сачувај“ измењени податак ће заменити стари податак у бази података.

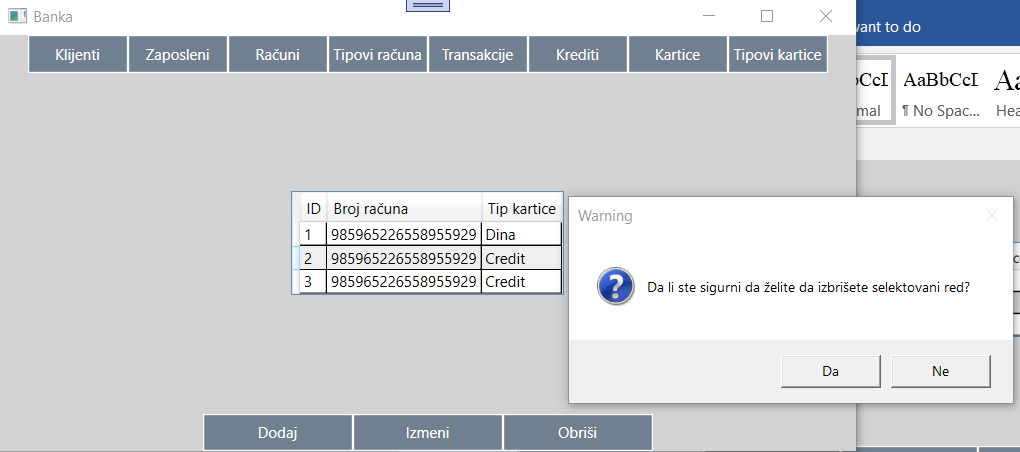


Слика 98. Отварање форме за измену

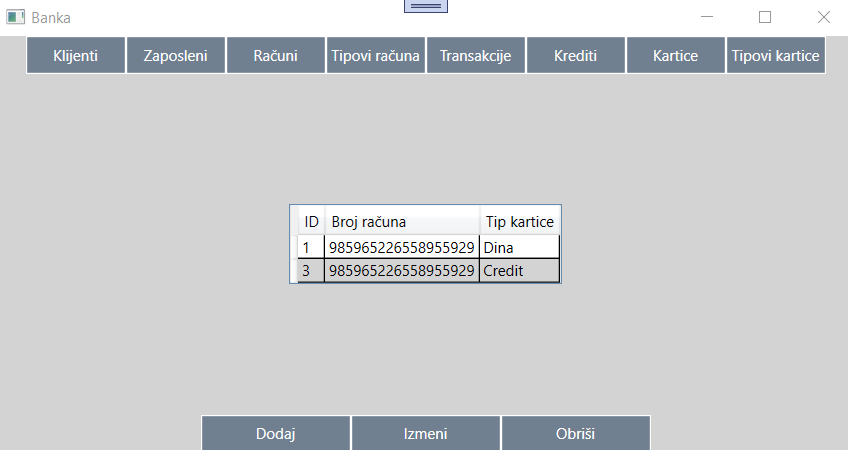


Слика 99. Измењени подаци у табели

Селектовањем реда и кликом на дугме „Обриши“ отвара се дијалог који тражи потврду да ли корисник жели да обрише тај ред и базе, уколико је одговор да, тај податак се брише из базе.



Слика 100. Брисање података



Слика 101. Ажурирани подаци у бази након брисања

# 8. Закључак

Кроз кораке смо објаснили начин креирања апликације за запослене у банкама, која пружа основне функције потребне за запослене у банци. По потреби апликација се може надоградити. На почетку смо креирали дијаграме који су нам дали увид у функционисање банке. Управо ти дијаграми су нам омогућили даљи рад у процесу креирања апликације. На основу дијаграма класа креирали смо базу података са табелама која ће нам служити да скалдиштимо податке које запослени уносе у систем и такође смо повезали табеле које имају заједничке информације преко страних кључева. Све то је потребно да се изгради темељ за настанак апликације. Креирана апликација има могућност додавања у базу нових података за осам различитих табела, такође омогућава измену већ унетих података и брисање уписаних података.