FAKULTET TEHNIČKIH NAUKA

UNIVERZITET U NOVOM SADU

WPF PROJEKAT

BIBLIOTEKA

MIŠAN MILICA / IT8-2022

Asistenti: Dušan Krstić, Todorić Anđela

2023/24

SADRŽAJ

[1. Opis realnog sistema 3](#_Toc154393201)

[2. Zadatak 3](#_Toc154393202)

[3. Opis tehnologije 3](#_Toc154393203)

[3.1 Microsoft Structured Query Language (SQL) 4](#_Toc154393204)

[3.2 Microsoft Visual Studio IDE 4](#_Toc154393205)

[3.3 Microsoft ActiveX Data Objects (ADO).NET 4](#_Toc154393206)

[3.4 Windows Presentation Foundation (WPF) 5](#_Toc154393207)

[4. UML 5](#_Toc154393208)

[4.1 Dijagram slučajeva upotrebe 6](#_Toc154393209)

[4.2 Dijagram klasa 11](#_Toc154393210)

[4.3 Dijagram sekvenci 12](#_Toc154393211)

[5. Baze podataka 13](#_Toc154393212)

[5.1 Kreiranje baze podataka 13](#_Toc154393213)

[5.2 Upiti i DML naredbe 22](#_Toc154393214)

[6. Kreiranje WPF aplikacije 23](#_Toc154393215)

[7. Testiranje aplikacije 52](#_Toc154393216)

[8. Zaključak 56](#_Toc154393217)

# Opis realnog sistema

U ovom projektu će detaljno biti opisan i razrađen sistem osnovnih operacija koje se nalaze u sistemu jedne biblioteke. Postoje dve osnovne operacije koje članovi mogu da urade, a to su iznajmljivanje i povratak iznajmljene knjige u biblioteku. U ovom projektu su uzete u obzir i sve one operacije i zadaci koje bibliotekar ima kao što su sam kontakt sa članovima, dodavanje i brisanje knjiga, sama evidencija svih podataka o članovima i knjigama.

# Zadatak

Cilj zadatka je da se napravi infromacioni sistem koji će predstavljati podršku poslovanju rada sistema biblioteke . U prvom delu zadatka vrši se modelovanje realnog sistema pomoću UML dijagrama koji na adekvatan način reprezentuju posmatrani sistem ili neki njegov deo. Zatim se, na osnovu tako modelovanog sistema, kreira WPF aplikacija koja simulira rad biblioteke. Komunikacija aplikacije sa bazom podataka ostvarivaće se putem ADO.NET tehnologije. Studenti treba da upravljaju celokupnim procesom razvoja WPF aplikacije, od modelovanja sistema (u okviru kojeg treba da kreiraju dijagram slučajeva upotrebe i dijagram klasa), do razvoja i testiranja WPF aplikacije koja predstavlja informacioni sistem za podršku poslovanja biblioteke. Informacioni sistem treba da pruži mogućnost prikazivanja podataka o članovima biblioteke, knjigama koje su na stanju, izdavanja knjiga, kao i povratka knjiga. Takođe, kroz aplikaciju je potrebno omogućiti dodavanje, ažuriranje i brisanje podataka nad klasama od interesa, definisanih dijagramom klasa.

# Opis tehnologije

U ovom poglavlju predstavljene su tehnologije koje će biti korišćene u izradi projektnih  
zadataka.

## 3.1 Microsoft Structured Query Language (SQL)

Microsoft SQL Server je vodeći relacionalni upravljač bazama podataka (RDBMS) koji pruža snažne mogućnosti za upravljanje, skladištenje i dohvaćanje podataka. Razvio ga je Microsoft, a koristi se za pohranu i upravljanje podacima u različitim aplikacijama, uključujući web stranice, poslovne sisteme i analitičke platforme. SQL Server podržava T-SQL, proširen SQL jezik, koji omogućava napredne upite, pohranjene procedure i funkcije. Integrira se s drugim Microsoft alatima poput Visual Studio i Excela, podržavajući poslovnu inteligenciju i analizu podataka. SQL Server Management Studio (SSMS) pruža platformu za administraciju baza podataka, olakšavajući rad s bazama podataka i održavanje njihove integriteta. Sistem takođe podržava različite nivoe izolacije transakcija, osiguravajući doslednost i integritet podataka u višekorisničkim okruženjima. Redovno dobija ažuriranja i nadogradnje kako bi pratilo najnovije trendove u industriji podataka i tehnologiji baza podataka.

## 3.2 Microsoft Visual Studio IDE

Microsoft Visual Studio je integrisano razvojno okruženje (IDE) koje podržava širok spektar programskih jezika. Omogućuje razvoj aplikacija za Windows, web, mobilne uređaje i oblak. Sadrži alate za uređivanje koda, debagovanje, dizajn interfejsa korisnika i upravljanje izvorima. Visual Studio podržava integraciju s Azure Cloud, olakšavajući izradu i implementaciju aplikacija u oblaku. Takođe uključuje bogate alate za testiranje i analizu performansi, pružajući celovito rješenje za razvoj modernih softverskih aplikacija. Visual Studio ima podršku za rad u timu putem alata za verzioniranje koda (poput Git-a) i mogućnosti saradnje, olakšavajući zajednički rad na projektima. Takođe uključuje podršku za popularne tehnologije i okvire poput .NET, ASP.NET, Xamarin, i mnoge druge.

## 3.3 Microsoft ActiveX Data Objects (ADO).NET

Microsoft ActiveX Data Objects (ADO.NET) je tehnologija koja omogućava programerima pristup i manipulaciju podacima u .NET okruženju. Ovo je deo Microsoftove platforme za razvoj softvera i predstavlja evoluciju ranijeg ActiveX Data Objects (ADO) modela. ADO.NET pruža konzistentan pristup podacima bez obzira na izvor podataka, bilo da se radi o bazama podataka, XML datotekama ili drugim izvorima podataka. Ključni elementi ADO.NET uključuju DataSet, DataReader i DataAdapter. DataSet predstavlja u memoriji podatke koji se mogu nezavisno manipulisati, DataReader omogućava brzo i učinkovito čitanje podataka iz izvora podataka, dok DataAdapter deluje kao most između DataSeta i izvora podataka, omogućavajući ažuriranje podataka između memorije i izvora.

## 3.4 Windows Presentation Foundation (WPF)

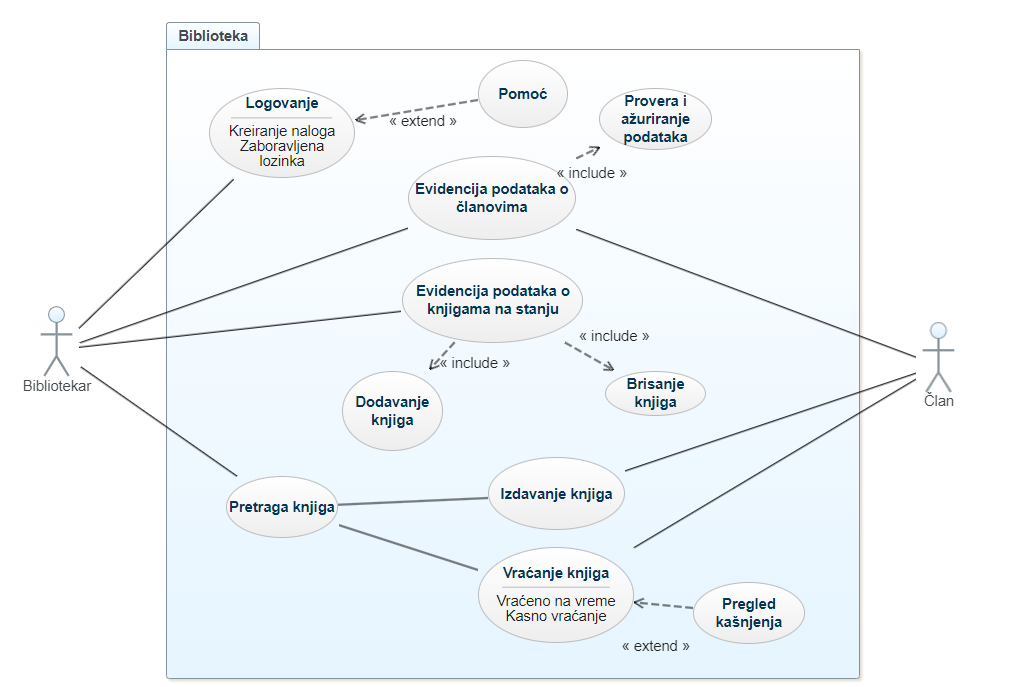
Windows Presentation Foundation (WPF) je tehnologija razvoja korisničkog interfejsa (UI) koja je deo Microsoft .NET okvira. Nekoliko ključnih karakteristika i informacija o Windows Presentation Foundation: WPF koristi XAML, deklarativni jezik označavanja, za definisanje korisničkog interfejsa i izgleda aplikacija. XAML omogućava odvojeni dizajn od logike aplikacije. WPF podržava snažan mehanizam vezivanja podataka koji omogućava automatsko ažuriranje korisničkog interfejsa kada se podaci promene, što olakšava rad s podacima u stvarnom vremenu. WPF omogućava izradu složenih i dinamičnih korisničkih interfejsa pomoću grafičkih elemenata, animacija, transformacija i efekata.Stilovi omogućavaju definisanje izgleda kontrola na jednom jestu, dok *templates* omogućavaju prilagodjavanje izgleda celih kontrola ili skupova kontrola. WPF se često koristi za razvoj desktop aplikacija gde je potrebno sofisticirani korisnički interfejs, poput poslovnih alata, kreativnih aplikacija, ili aplikacija koje zahtevaju naprednu interakciju s korisnikom.

# UML

Unified Modeling Language UML - formalni jezik, namenjen za: formalno specificiranje sistema (mogućih stanja i ponašanja) , modelovanje sistema, vizuelizaciju sistema, analizu sistema, dokumentovanje sistema, razvoj (analizu i projektovanje) softverskih sistema koji predstavljaju modele drugih, realnih sistema. Zasnovan je na principima objektne orijentacije. UML je standardni jezik čiji su rečnik i pravila usmerena na konceptualnu i fizičku reprezentaciju sistema kao na primer sistema sa intenzivnom primenom softvera, ali i za modelovanje drugih sistema. Rečnik i pravila UML jezika govore kako treba formirati i čitati dobro formirane modele, ali ne govore koje modele treba formirati i kada to treba činiti. Uloga dobro definisanog procesa razvoja softvera je da se definišu modeli koje treba formirati, koje aktivnosti i koje subjekte treba koristiti i kada, kako koristiti modele u oceni i kontroli projekta kao celine itd. UML modeli nezavisni su od jezika u kojem softver treba da bude realizovan.

## 4.1 Dijagram slučajeva upotrebe

Dijagram slučajeva upotrebe (Use Case Diagram) je vrsta dijagrama u okviru UML (Unified Modeling Language) koja se koristi za modelovanje interakcija između korisnika (ili sistema) i sistema softvera. Ovi dijagrami se često koriste za definisanje funkcionalnih zahteva sistema i opisuju kako korisnici interaguju sa sistemom radi ostvarivanja određenih ciljeva ili funkcionalnosti. Elementi dijagrama slučajeva upotrebe su: akteri, slučajevi upotrebe, veze (asocijacija gde spadaju include i extend, generalizacija). Veze su predstavljene strelicama, slučajevi upotrebe elipsama, akteri su predstavljeni ikoničnim načinom.



*Slika 4.1 – Dijagram slučajeva upotrebe za informacioni sistem biblioteke*

1. ***Slučaj upotrebe:*** Logovanje

***Kratak opis:*** Logovanje u sistem biblioteke.

***Učesnici:*** Bibliotekar.

***Uslovi koji moraju biti zadovoljeni pre izvršavanja:*** Bibliotekar zna podatke neophodne za logovanje u sistem. Takođe mora da poseduje nalog u sistemu.

***Opis:*** Sistem prikazuje bibliotekaru formu za prijavu u sistem. Ona se sastoji iz 2 dela, prvi deo je za unos lozinke, a drugi deo je za unos korisničkog imena. Kada bibliotekar unese svoje podatke vrši se provera da li uneti podaci postoje, kao i da li su dobro uneti podaci ukoliko nalog postoji. [***Izuzetak:*** Pogrešno uneta lozinka ili korisničko ime ]. Nakon što je izvršena provera podataka i utvrđeno da su oni tačni bibliotekar može da pristupi funkcionalnostima informacionog sistema.

***Izuzeci:*** ***[Pogrešno uneta lozinka ili korisničko ime ]:*** U slučaju da bibliotekar pogrešno unese lozinku ili korisnički nalog, sistem obaveštava bibliotekara putem ispisa poruke na ekranu.

***Uslovi koji moraju biti zadovoljeni nakon izvršavanja:*** Bibliotekar je u mogućnosti da koristi sve podatke iz sistema.

1. ***Slučaj upotrebe:*** Evidencija podataka o članovima

***Kratak opis:*** Prikupljanje neophodnih podataka o članovima biblioteke.

***Učesnici:*** Bibliotekar, Član

***Uslovi koji moraju biti zadovoljeni pre izvršavanja:*** Bibliotekar zna sve podatke neophodne za prijavu u sistem, te je prijavljen u sistem.

***Opis:*** Prilikom dolaska u biblioteku i razgovora sa članom, bibliotekar proverava da li je osoba već član biblioteke, zatim se ukoliko je potrebno ažuriraju stari podaci ili pak ako je član novi prikupljaju se novi podaci i evidentiraju u sistem. [***Izuzetak:*** Greška prilikom unošenja podataka o članu.]

***Izuzeci: [Greška prilikom unošenja podataka o članu]*** U slučaju da je došlo do pogrešno unetih podataka o članu, bibliotekar mora da ponovo razgovara sa članom i da ispravi grešku.

***Uslovi koji moraju biti zadovoljeni nakon izvršavanja:*** U sistemu biblioteke postoje svi tačni podaci o članu, te je moguće pristupiti sledećem koraku koje član zahteva.

1. ***Slučaj upotrebe:*** Evidencija podataka o knjigama na stanju

***Kratak opis:*** Prikupljanje svih potrebnih podataka o knjizi.

***Učesnici:*** Bibliotekar

***Uslovi koji moraju biti zadovoljeni pre izvršavanja:*** Bibliotekar je prikupio i evidentirao sve podatke neophodne o knjizi.

***Opis:*** Bibliotekar prilikom pristupa sistemu proverava da li knjiga već postoji u sistemu. Ukoliko postoji i potrebno je ažurira neke podatke, a ako knjiga ne postoji on evidentira nove podatke o knjizi. Takođe prilikom zahteva člana za nekom knjigom on mora vršiti pretragu knjige u sistemu. [***Izuzetak:*** Neki podaci o knjizi su uneti pogrešno.]

***Izuzeci:[ Neki podaci o knjizi su uneti pogrešno]*** Ukoliko je došlo prilikom greške kod unošenja podataka o knjizi, bibliotekar mora da izmeni i ponovo unese podatke.

***Uslovi koji moraju biti zadovoljeni nakon izvršavanja***: Bibliotekaru je omogućen pristup svim funkcionalnostima sistema.

1. ***Slučaj upotrebe:*** Dodavanje novih knjiga

***Kratak opis:*** Unos novih knjiga u sistem.

***Učesnici:*** Bibliotekar

***Uslovi koji moraju biti zadovoljeni pre izvršavanja:*** Neophodno je da bibliotekar zna sve podatke neophodne za jednu knjigu kako bi je mogao dodati u sistem.

***Opis:*** Bibliotekar unosi podatke o knjizi kao što su: ime autora, žanr, naziv knjige i vrši se provera da li knjiga već postoji. Ako knjiga ne postoji sistem potvrđuje da je provera uspela i da se knjiga ne nalazi u sistemu, te se onda nova knjiga upisuje u sistem. Ukoliko knjiga već postoji u sistemu bibliotekaru se onemogućava unos iste knjige u sistem. [***Izuzetak:*** Pogrešno uneti podaci o knjizi.]

***Izuzeci:[Pogrešno uneti podaci o knjizi ]***  Kada se desi greška pri unosu podataka o knjizi, sistem obaveštava korisniku u vidu iskačućeg prozora na ekranu.

***Uslovi koji moraju biti zadovoljeni nakon izvršavanja:*** Knjiga je uspešno upisana u sistem.

1. ***Slučaj upotrebe:*** Brisanje knjiga

***Kratak opis:*** Brisanje knjiga iz sistema biblioteke.

***Učesnici:*** Bibliotekar

***Uslovi koji moraju biti zadovoljeni pre izvršavanja:*** Knjiga čije podatke je uneo bibliotekar mora da postoji u sistemu.

***Opis:*** Bibliotekar unosi osnovne podatke o knjizi. Vrši se provera da li unešena knjiga postoji u sistemu. Ukoliko knjiga postoji ona se automatski briše iz sistema. [***Izuzetak:***] Pogrešno uneti podaci o knjizi.

***Izuzeci:[ Pogrešno uneti podaci o knjizi ]***  Kada se desi greška pri unosu podataka o knjizi, sistem obaveštava korisniku u vidu iskačućeg prozora na ekranu.

***Uslovi koji moraju biti zadovoljeni nakon izvršavanja:*** Knjiga je uspešno obrisana iz sistema, moguće je dalje adekvatno korišćenje istog

1. ***Slučaj upotrebe:*** Pretraga knjiga

***Kratak opis:*** Pretraga knjiga u sistemu.

***Učesnici:*** Bibliotekar

***Uslovi koji moraju biti zadovoljeni pre izvršavanja:*** Bibliotekar mora da bude prijavljen u sistem.

***Opis:*** Bibliotekar unosi podatke o knjizi, autor, naziv, žanr, izdavač. Zatim proverava da li je data knjiga na stanju u sistemu ili je možda izdata. Kada proveri dostupnost, nastavlja dalje izvršavanje podnetog zahteva koji je uzrokovao samu pretragu. [***Izuzetak:*** Pogrešno uneti podaci o knjizi.]

***Izuzeci:[ Pogrešno uneti podaci o knjizi ]***  Kada se desi greška pri unosu podataka o knjizi, sistem obaveštava korisniku u vidu iskačućeg prozora na ekranu.

***Uslovi koji moraju biti zadovoljeni nakon izvršavanja:*** U sistemu postoje adekvatni podaci u knjizi i može se pristupiti sledećem koraku podnetog zahteva.

1. ***Slučaj upotrebe:*** Izdavanje knjiga

***Kratak opis:*** Izdavanje knjiga članovima biblioteke.

***Učesnici:*** Član, Bibliotekar

***Uslovi koji moraju biti zadovoljeni pre izvršavanja:*** U sistemu biblioteke mora da postoji knjiga koju član zahteva, ona mora biti na stanju , takođe član mora biti učlanjen u bilioteku.

***Opis:*** Prilikom dolaska člana u biblioteku bibliotekar u toku razgovora sa njim saznaje koju knjigu on želi da iznajmi. Nakon tog saznanja on proverava da li biblioteka poseduje tu knjigu, da li je već izdata. Ukoliko je poseduje knjigu je moguće izdati, a ako je nema u sistemu ili je već izdata, predlaže članu ostale moguće opcije. [***Izuzetak:*** Pogrešno uneti podaci o knjizi.]

***Izuzeci:[ Pogrešno uneti podaci o knjizi ]***  Kada se desi greška pri unosu podataka o knjizi, sistem obaveštava korisniku u vidu iskačućeg prozora na ekranu.

***Uslovi koji moraju biti zadovoljeni nakon izvršavanja:*** Bibliotekar može da pristupi procesu izdavanja.

1. ***Slučaj upotrebe:*** Vraćanje knjiga

***Kratak opis:*** Vraćanje pročitane knjige u biblioteku.

***Učesnici:*** Bibliotekar, Član biblioteke

***Uslovi koji moraju biti zadovoljeni pre izvršavanja:*** Knjiga mora da postoji u sistemu i da bude izdata članu.

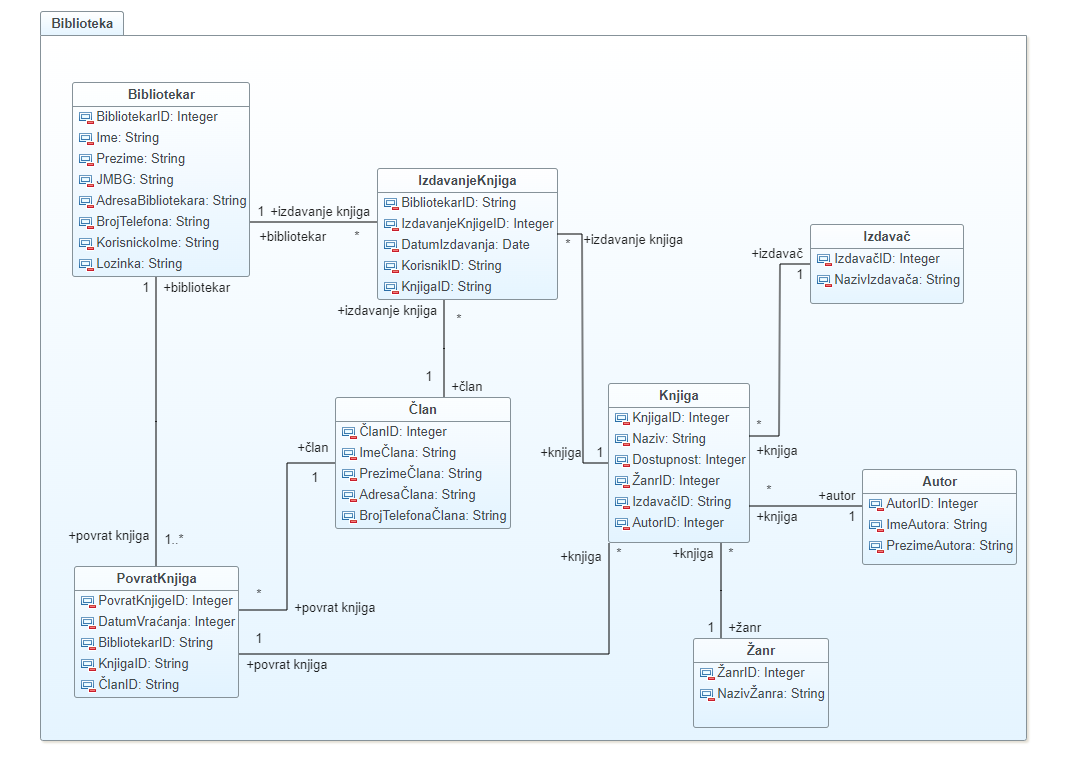
***Opis:*** Član dolazi u biblioteku sa knjigom koju želi da vrati. Prilikom razgovora sa članom bibliotekar proverava podatke o članu i podatke o samoj knjizi. Ukoliko je ispoštovan obavezan rok za povratak knjige, knjiga je uspešno vraćena. [***Izuzetak:*** Prekoračenje roka ].

***Izuzeci:[Prekoračenje roka].*** U slučaju da je knjiga kasno vraćena član će snositi određene posledice propisane zakonom.

***Uslovi koji moraju biti zadovoljeni nakon izvršavanja:*** Knjiga je vraćena i trenutno je na stanju, bibliotekar može da pristupi njenim podacima i ukoliko neko zahteva može da mu izda istu.

## 4.2 Dijagram klasa

Dijagram klasa je vrsta dijagrama u okviru UML (Unified Modeling Language) koji se koristi za vizualizaciju i modeliranje strukturalnih odnosa i organizacije klasa u softverskom sistemu. Ovo je korisno sredstvo za modeliranje objektno orijentisanog softvera. Elementi dijagrama klasa su: klase- predstavljaju osnovne građevne blokove sistema i sadrže naziv klase, atribute i metode i veze između klasa ( asocijacija koja se deli na agregaciju i kompoziciju, zavisnost, realizacija i generalizacija).

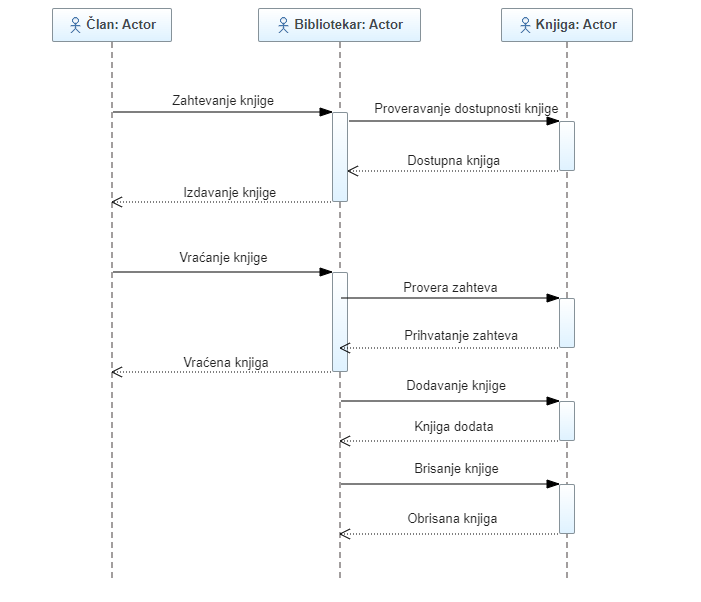


*Slika 4.2 – Dijagram klasa za informacioni sistem biblioteke*

Identifikovano je osam klasa: *Bibliotekar, IzdavanjeKnjiga, PovratKnjiga, Član, Knjiga, Autor, Izdavač, Žanr.* Bibliotekar može da izda knjigu ukoliko je ima na stanju. Član može da uzme ili vrati više knjiga, minimalno jedna knjiga može biti izdata ili vraćena. Sama knjiga može da ima jednog izdavača, dok jedan izdavač može da izda više knjiga. Autor može da ima više napisanih knjiga, dok jedna knjiga mora da ima minimalno jednog autora. Isto tako žanr knjige mora biti određen u jednoj knjizi, ali isti žanr mogu da imaju više knjiga istovremeno. U svim pojedinačnim klasama ovog dijagrama klasa nalaze se ID obeležja, nazivi, ali i sve dodatne informacije vezane za bibliotekara i člana biblioteke kao što su kontakt, mesto stanovanja; sve infromacije vezane za vreme izdavanja i vraćanja knjiga kao i osnovnih podataka o samoj knjizi autor, žanr, izdavač…

## 4.3 Dijagram sekvenci

Dijagram sekvenci je još jedan tip dijagrama u okviru UML (Unified Modeling Language) koji se koristi za modelovanje interakcija između objekata u softverskom sistemu tokom vremena. Ovaj dijagram omogućava vizualizaciju redosleda poruka i poziva metoda između objekata, čime se ilustruju dinamički aspekti sistema. Osnovini elementi dijagrama sekvence su objekti: predstavljaju učesnike ili objekte u interakciji i obično su navedeni vertikalnim linijama sa nazivom entiteta iznad linije; životni ciklus entiteta: koje opisuju vremenski tok aktivnosti ili poruka koje se razmenjuju između entiteta; poruke: strelice koje povezuju entitete i označavaju komunikaciju između njih, obično s dodatnim informacijama poput naziva poruke, argumenata i povratnih informacija; aktivnosti: blokovi koji predstavljaju izvođenje određenih operacija ili funkcija.

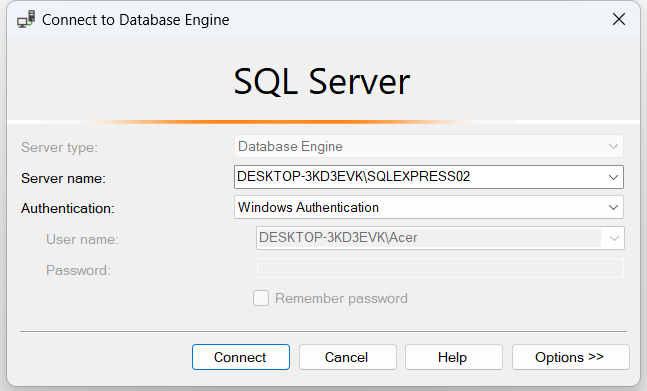


*Slika 4.3 – Sekvencijalni dijagram za informacioni sistem aerodrome*

# Baze podataka

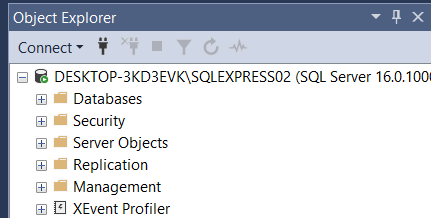
## 5.1 Kreiranje baze podataka

Sledeći korak u izradi informacionog sistema jeste kreiranje baze podataka. Na samom  
početku, potrebno je pokrenuti *SQL Server Management Studio Express* okruženje. Kada se to  
uradi, pojaviće se prozor sa dijalogom, kao što je prikazano na slici 5.1. U prikazanom  
dijalogu, potrebno je podesiti parametre *Server type i Authentication*, kao što je prikazano na  
slici 5.1. Server name nije potrebno menjati, ono predstavlja naziv uređaja na  
kojem se kreira baza podatak



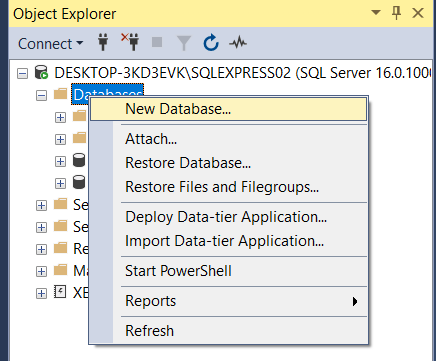
*Slika 5.1 Povezivanje sa serverom*

Klikom tastera miša na dugme **Connect** dijalog će se zatvoriti, a otvoriće se novi prozor koji treba da izgleda kao što je prikazano na slici 5.2



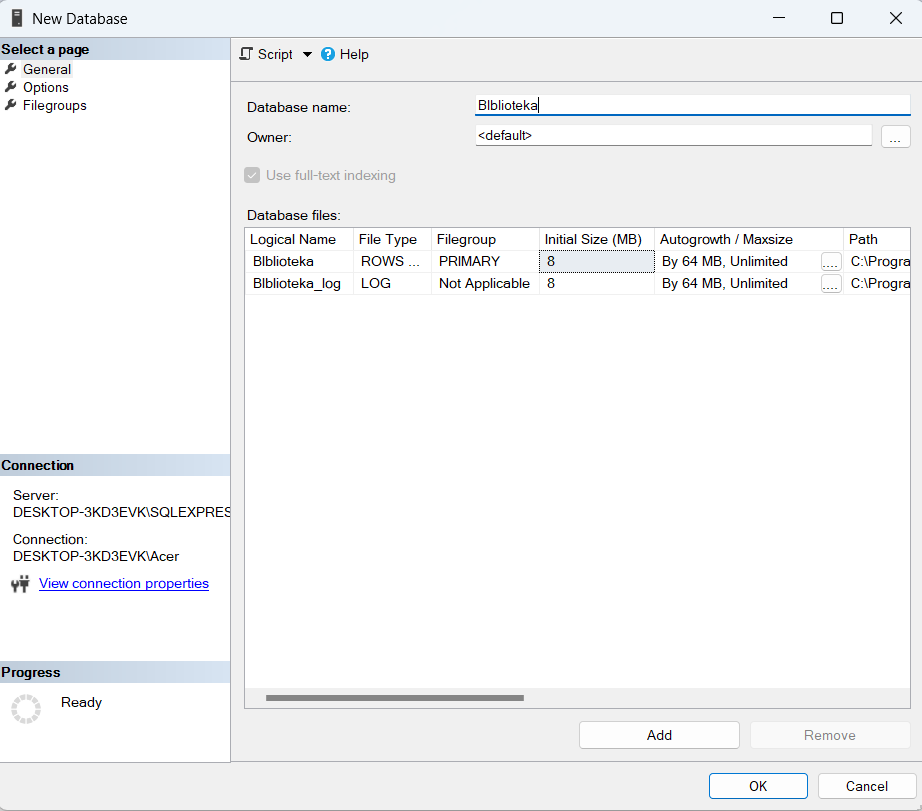
*Slika 5.2 Izlged Object Explorer-a nakon povezivanja sa bazom*

Sada je potrebno kreirati bazu podataka za aerodromsku aplikaciju. Baza podataka se kreira  
desnim klikom tastera miša na **Database**u *treeviewu-u* koji se nalazi sa leve strane prozora u  
okviru *Object Explorer-a.* Otvoriće se lista u kojoj je potrebno označiti **New Database**.



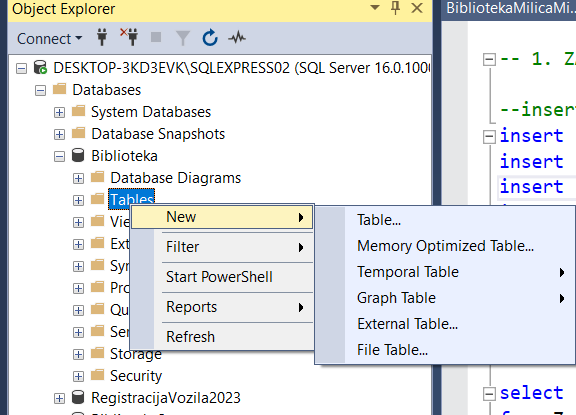
*Slika 5.3 Kreiranje baze podataka*

Novi prozor ponudiće opciju za kreiranje baze podataka. Potrebno je dati naziv bazi podataka,  
u ovom slučaju naziv će biti **Biblioteka.** Ostale parameter nije potrebno menjati.



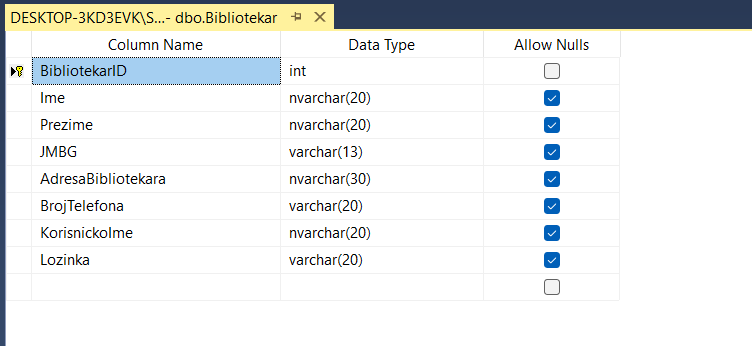
*Slika 5.4 Podešavanje parametara pri samom kreiranju baze podataka*

Posle uspešnog kreiranja baze podataka u *Object Explorer-u* u folderu *Database* se pojavljuje  
kreirana baza podataka *Biblioteka*. Proširivanjem baze podataka Biblioteka, u stablu Object Explorer-a, pojavljuje se više datoteka. Desnim klikom tastera miša kliknuti na datoteku Tables i odabrati Table (slika 5.5). Nakon klika na ovu opciju, sa desne strane otvara se prozor za definisanje tabele.

**

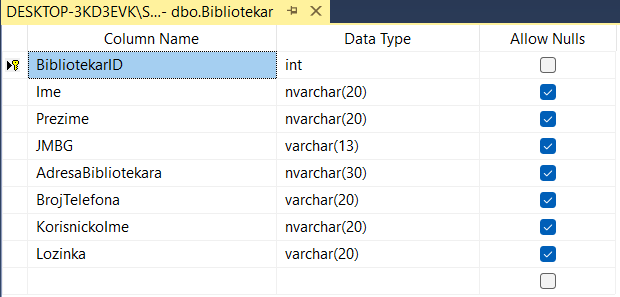
*Slika 5.5 Kreiranje tabela u bazi podataka*

Sledeće što je potrebno uraditi jeste kreirati kolone tabele tako što će se popuniti polja Column  
Name, Data Type, Allow Nulls. U kolonu Column Name se upisuje naziv atributa, u koloni  
Data Type se bira tip podatka za željeni atribut, dok se u koloni Allow Nulls označava da li  
određeno polje može imat null vrednosti. Preporučuje se da Allow Nulls bude označeno za sve  
kolone, osim za primarni ključ, kako bi se izbegli određeni problemi prilikom upisa rekorda u  
tabelu. Svaka tabela određene baze podataka mora posedovati primarni ključ - ID i bilo koja kolona  
može da se deklariše kao primarni ključ. U ovom slučaju, kolona VlasnikID treba da se  
deklariše kao primarni ključ. Kao takva, ova kolona će zbog svoje jedinstvene vrednosti imati  
tip podatka int i neće imati null vrednosti (slika 5.6)



*Slika 5.6 Prikaz kreiranih obeležja*

Da bi se kolona BibliotekarID deklarisala kao primarni ključ, potrebno je kliknuti desnim klikom tastera miša na crnu strelicu koja se nalazi pored naziva ove kolone. Otvoriće lista u kojoj treba odabrati opciju *Set Primary Key*. Ovom komandom se kolona BibliotekarID deklariše kao primarni ključ (slika 5.7).



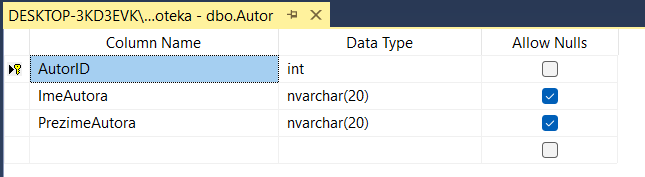
*Slika 5.7 Kreiran primarni ključ*

Da bi sistem za upravljanje bazom podataka - SUBP automatski kreirao primarni ključ,  
potrebno je u *Column Properties* pronaći opciju *Identity Specification*. Proširivanjem opcije  
*Identity Specification* dobija se opcija *Is Identity*, koju treba promeniti na *Yes*. Ovde je, takođe,  
moguće podesiti parametre *Identity Increment* i *Identity Seed*. *Identity Increment*-om se definiše  
za koliko raste sledeća vrednost ključa, dok *Identity Seed* označava od koje vrednosti počinju  
da se memorišu ključevi (slika 5.8).

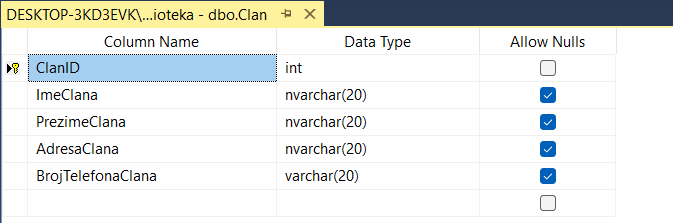


*Slika 5.8 Podešavanje automatske inkrementacije vrednosti obeležja primarnog ključa*

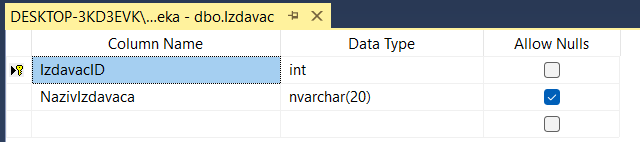
Čuvanje tabele vrši se pritiskanjem tastera *Ctrl i S* nakon čega će SQL Server Managament  
Studio zahtevati unos naziva tabele. Isti proces treba ponoviti za kreiranje tabela *Autor, Clan, Izdavac, IzdavanjeKnjiga, PovratKnjiga, Knjiga, Zanr.*



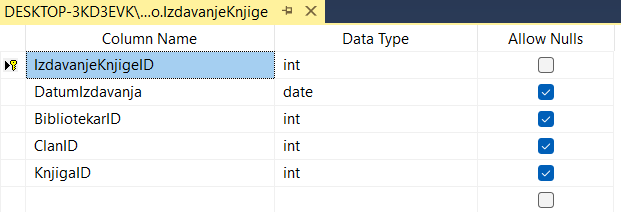
*Slika 5.9 Kreiranje tabele Autor*



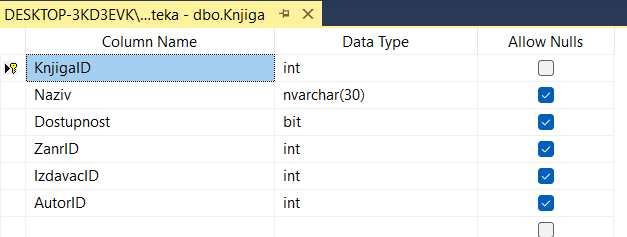
*Slika 5.10 Kreiranje tabele Clan*

**

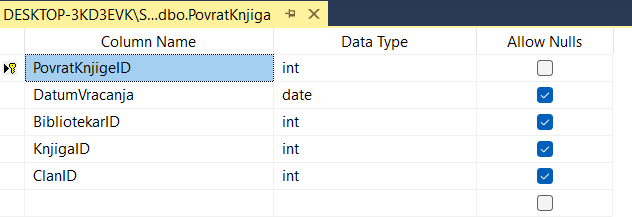
*Slika 5.11 Kreiranje tabele Izdavac*

**

*Slika 5.12 Kreiranje tabele IzdavanjeKnjige*

**

*Slika 5.13 Kreiranje tabele Knjiga*

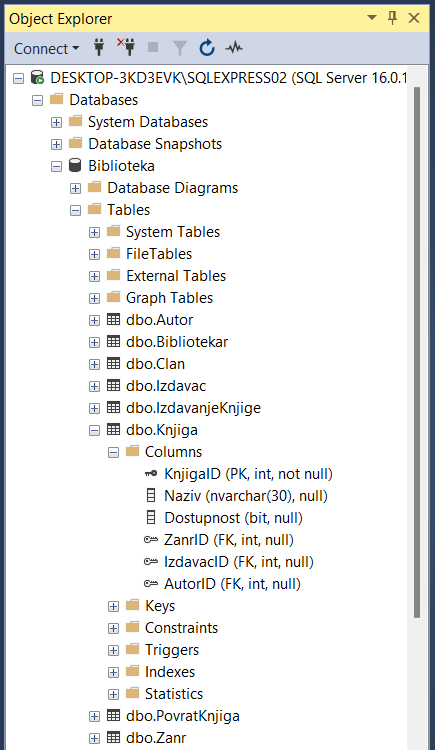
**

*Slika 5.14 Kreiranje tabele PovratKnjiga*

**

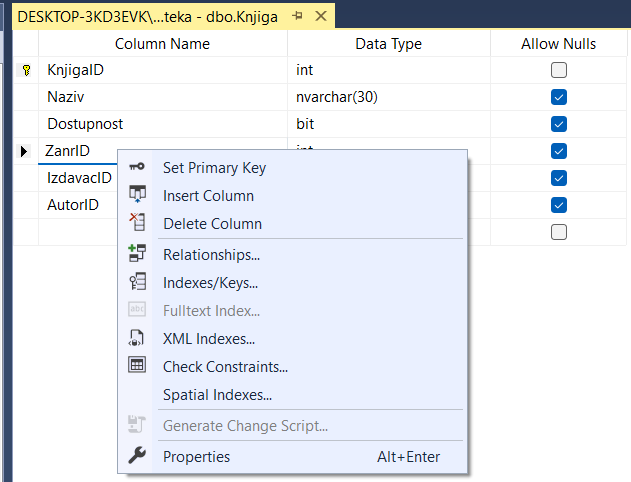
*Slika 5.15 Kreiranje table Zanr*

Nakon toga bi u *treeview*-u, koji se nalazi sa leve strane prozora u okviru *Object Explorer*-a,  
trebala da se nalazi baza podataka Biblioteka, u kojoj će se, pored već kreirane tabele *Knjiga*, naći I novokreirane tabele sa odgovarajućim atributima (slika 5.16).



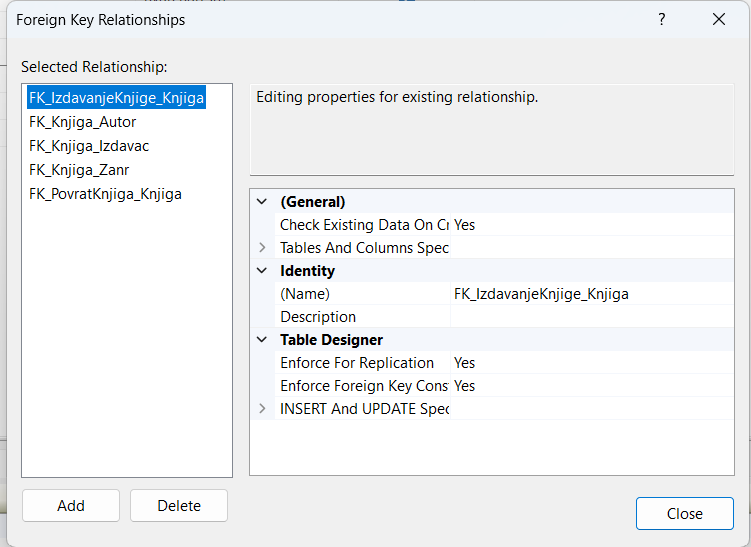
*Slika 5.16 Prikaz kreiranih tabela i kolona tabele Knjiga*

Kao što se može videti na slici 5.16, u tabeli Knjigase nalaze tri strana ključa (*ZanrID, IzdavacID, AutorID*). Prilikom unosa atributa u tabelu *Knjiga*, potrebno je uneti i atribute koji imaju nazive *ZanrID, IzdavacID i AutorID,* a vrednost *DataType*-a za njih staviti na int. Zatim je potrebno kliknuti desnim klikom tastera miša na crnu strelicu koja se nalazi pored naziva ove kolone.Otvoriće se lista u kojoj treba odabrati opciju *Relationships* (slika 5.17).



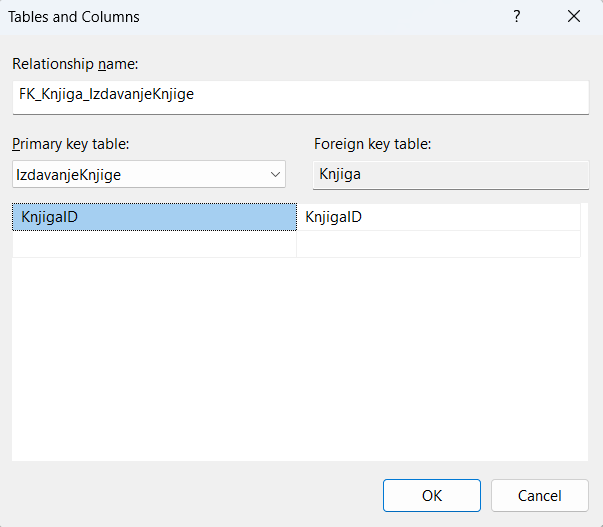
*Slika 5.17 Dodavanje veza*

Nakon odabrica opcije *Relationships,* otvara se prozor *Foreign key Relationships* (slika 5.18) u  
kojem je potrebno prvo kliknuti tasterom miša na dugme ***Add****¸*a zatim proširiti polje *Tabels and  
Columns Specifications* i kliknuti na tri tačke koje se nalaze desno.



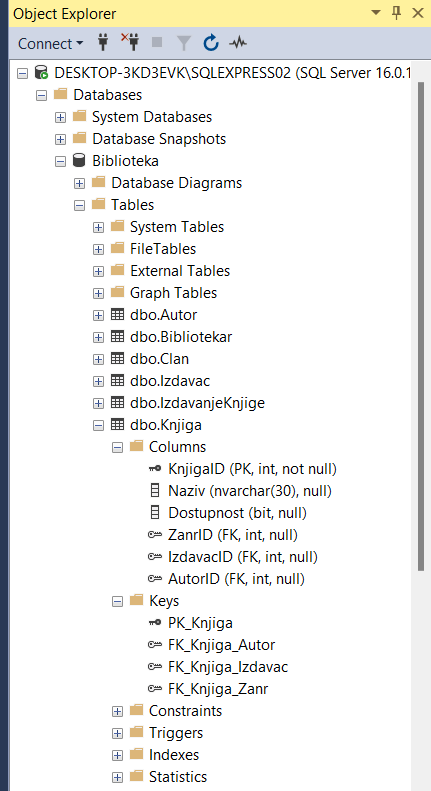
*Slika 5.18 Dodavanje stranog ključa*

U otvorenom prozoru *Tables and Columns* potrebno je u okviru *Foreign key table,* konkretno  
iz tabele *IzdavanjeKnjige,* iz padajuće liste izabrati atribut definisan prilikom kreiranja tabele,  
a koji je potrebno povezati sa drugom tabelom. Zatim, u okviru *Primary key table* treba izabrati  
tabelu sa kojom se vrši povezivanje, i iz padajuće liste odabrati ključ te tabele sa kojim se  
povezuje željeni atribut. Nakon što je sve ovo urađeno, prozor treba da izgleda kao na sledećoj  
slici (slika 5.19).



*Slika 5.19 Kreiranje veza između tabela*

Dodavanjem *Relationships*-a atributima *ZanrID, IzdavacID* i *AutorID* u tabeli *Knjiga*, oni postaju  
strani ključevi. Da li je povezivanje tabele uspešno urađeno znaće se proverom postojanja u  
*treeview*-u, u okviru tabele *Knjiga*, pod kategorijom *Keys* (slika 5.20)

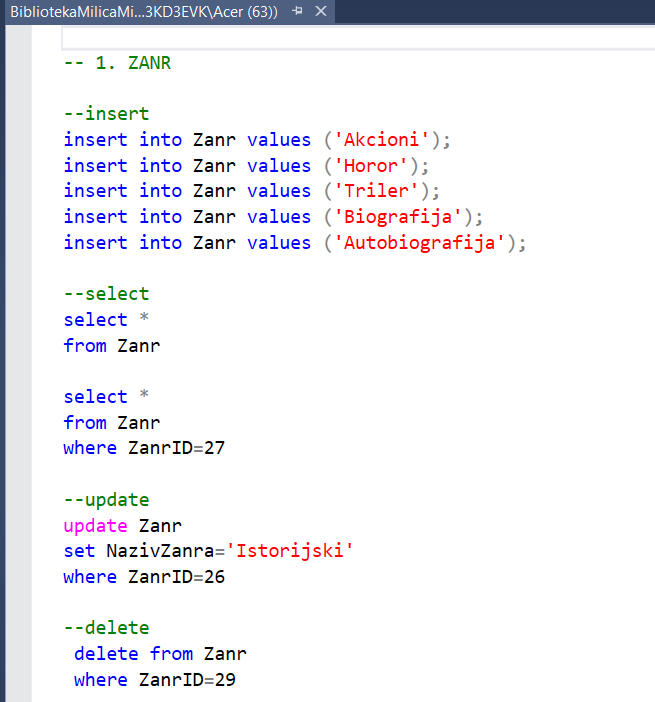


Slika 5.20 TreeView prikaz ključeva

Isti postputak potrebno je ponoviti za sve tabele koje u sebi sadrže strane ključeve koji su primarni ključevi u drugoj tabeli *Knjiga (KnjigaID)*. Baza podataka kreirana je na način koji je prikazan kroz celo poglavlje 5. Sledeći korak u izradi projektnog zadatka je kreiranje WPF aplikacije.

## 5.2 Upiti i DML naredbe

Na slici 5.21 prikazan je kod *SQL* naredbi sa kojim je izvršen unos inicijalnih pet vrednosti u tabelu Zanr, brisanje, prikaz podataka kao i ažuriranje podataka baze podataka *Biblioteka.* Isti je princip i za sve ostale tabele u bazi.

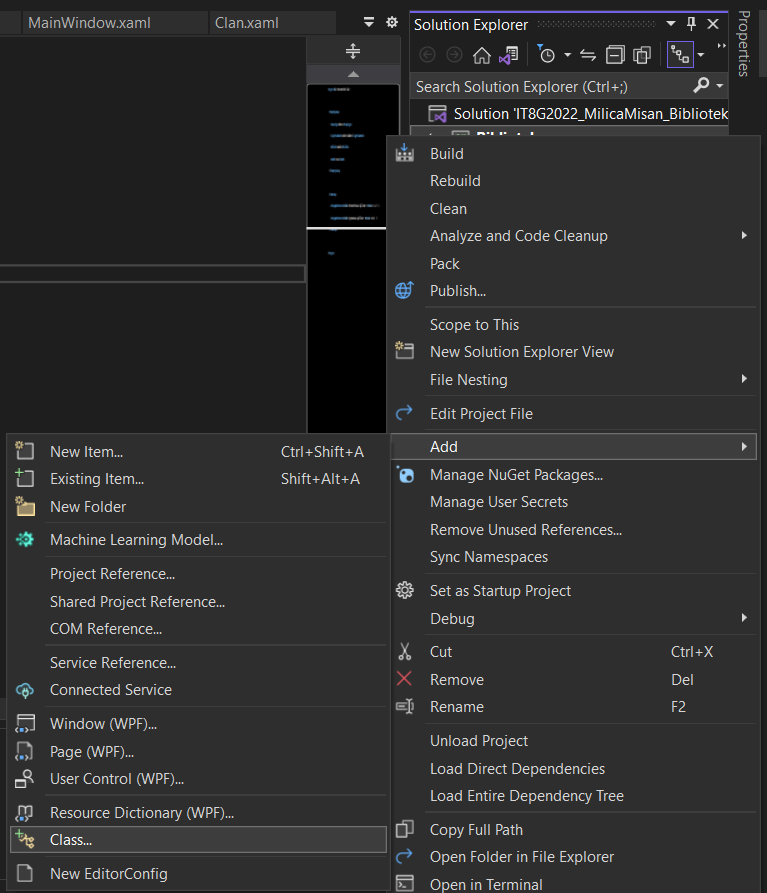


*Slika 5.21 Rad sa podacima tabele Zanr*

# Kreiranje WPF aplikacije

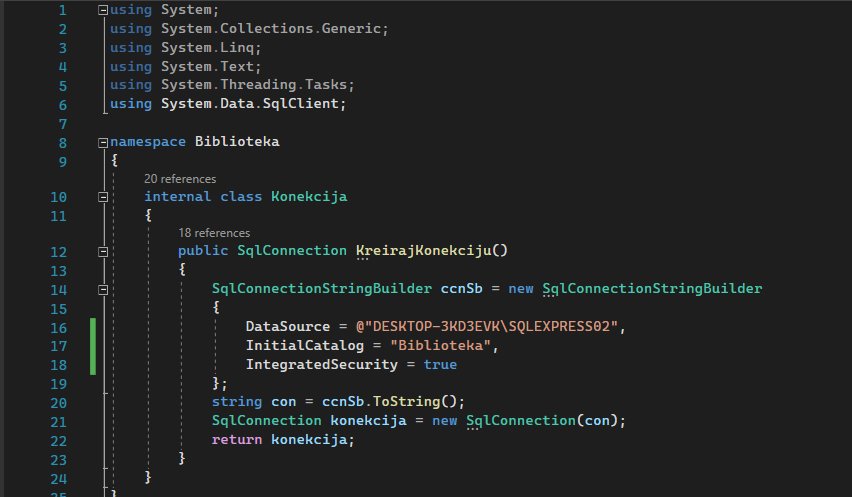
U okviru ovog poglavlja biće prikazano na koji način je kreirana WPF aplikacija za informacioni sistem Biblioteka. Nakon pokretanja razvojnog okruženja Visual Studio, potrebno je tasterom miša kliknuti na File menu, a zatim odabrati New Project. Odabrati Windows Classic Desktop, zatim WPF Application. Navesti ime aplikacije i kliknuti OK. U Solution Explorer-u pojaviće se napravljena aplikacija.

Sledeći korak je kreiranje konekcije u aplikaciji, ka bazi podataka. Za početak, kreira se klasa u kojoj će se nalaziti metoda putem koje će se ostvarivati veza na prethodno kreiranu bazu. Klasa se kreira desnim klikom tastera miša na naziv projekta (IT8G2022\_MisanMilica\_Biblioteka), a zatim Add/Class (slika 6.1)



*Slika 6.1 Kreiranje klase*

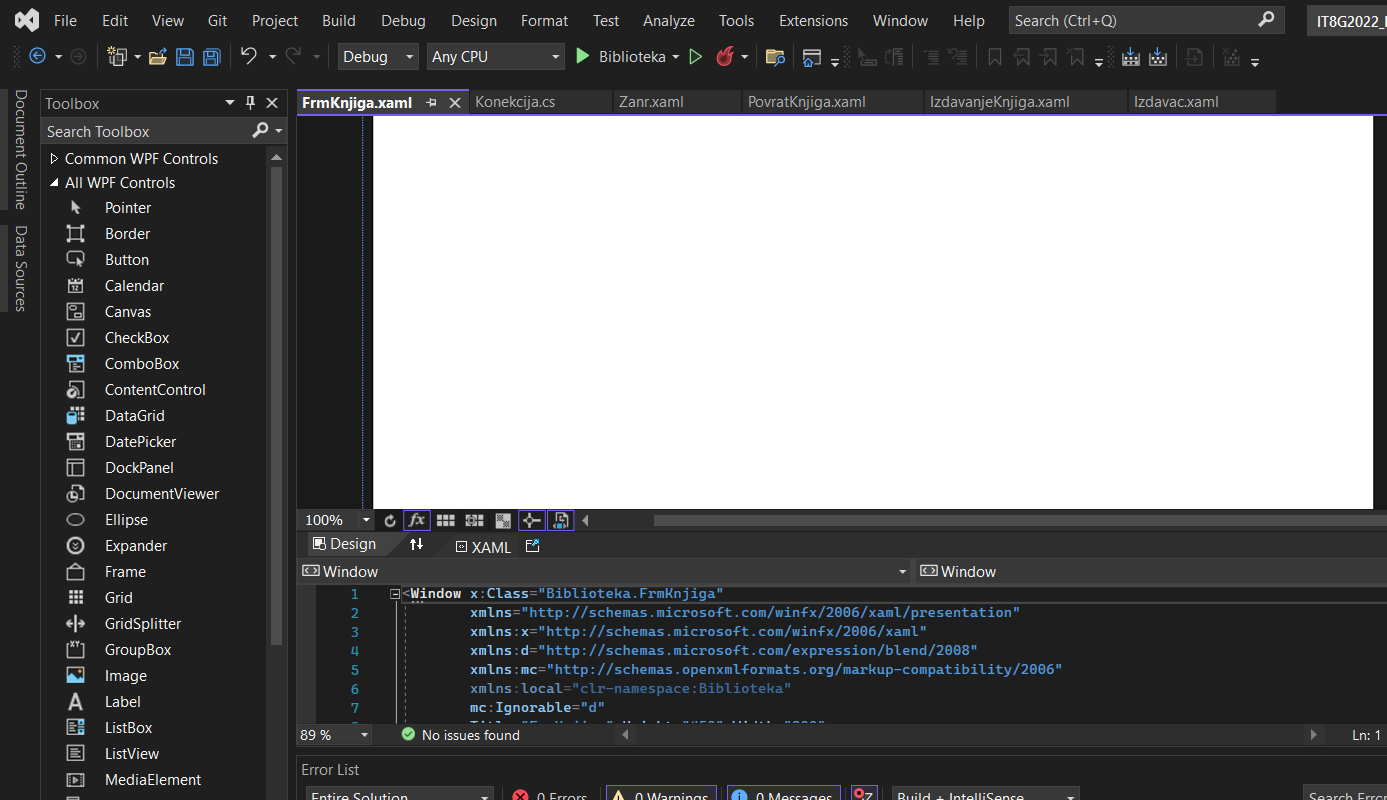
Novokreiranoj klasi se dodeljuje naziv ,,Konekcija" i njen sadržaj prikazan je u listingu 6.1. Na samom početku potrebno je uključiti imenski prostor System.Data.SqlClient. Telo klase sadrži samo jednu metodu pod nazivom "KreirajKonekciju", koja će se u kasnijem razvoju aplikacije pozivati po potrebi, odnosno svaki put kada je potrebno ostvariti vezu sa bazom podataka, stoga je povratni tip ove metode instanca klase SqlConnection. Unutar same metode prikazana su podešavanja parametara konekcije i Add to Source Control.  
  
1) DataSource - naziv servera na kojem je baza podataka smeštena,  
2) Initial Catalog - naziv baze podataka kojoj je potrebno pristupiti i   
3) IntegratedSecurity - ukoliko se baza nalazi na lokalnoj mašini, ovaj parametar postavlja se  
na true.  
  
Nakon podešavanja ovih parametara, ceo string se prosleđuje instanci klase SqlConnection.



*Listing 6.1 Kreiranje klase Konekcija*

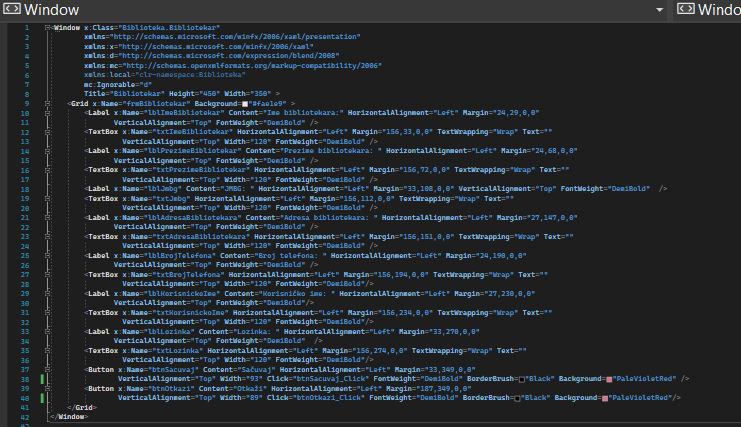
Sledeći korak u izradi WPF aplikacije je kreiranje ekranskih formi (eng. Forms) - prozora, za dodavanje objekata u bazu podataka

Nakon kreiranja formi, potrebno je dodati novi prozor za svaku od formi. WPF prozor kreira se desnim klikom tastera miša na ime baze, a zatim Add/Window. Nakon toga, prikazaće se prozor za dodavanje novog Item-a unutar kog je potrebno podesiti naziv i zatim kliknuti na Add. Kada je novi prozor izgenerisan, potrebno je, unutar njega, dodati određene elemente putem kojih će se ostvarivati potrebne funkcionalnosti. Na levoj strani prozora, klikom tastera miša na Toolbox, prikazaće se lista svih ugrađenih WPF komandi (slika 6.3). Jednostavnim prevlačenjem tih komandi na centralni prozor, izgenerisaće se kod u XAML deklarativnom jeziku unutar kojeg će se vršiti detaljna podešavanja svih komandi unutar prozora.

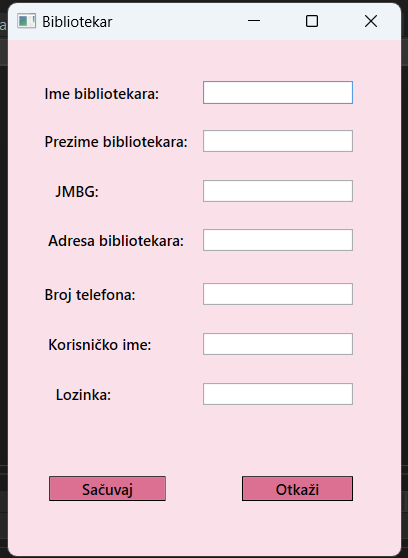


*Slika 6.2 Dodavanje komandi iz ToolBox-a*

Forma za dodavanje novog Bibliotekara u bazu podataka treba da sadrži elemente prikazane u listingu 6.3. To je osnovni TextBox unutar kog će biti upisana vrednost obeležja. Takođe, uz svaki TextBox, mora postojati naziv (Label) koji će opisivati njegovu namenu. Na kraju, svaka forma mora imati dugme Sačuvaj, na čiji klik tasterom miša će se vrednosti upisivati u bazu podataka, i dugme Otkaži, koje korisniku omogućava zatvaranje forme. Ove funkcionalnosti dugmića postižu se dvostrukim klikom tastera miša na odgovarajuće dugme u okviru forme, čime se u Code Behind-u generišu metode btnSacuvaj\_Click i btnOtkazi\_Click.

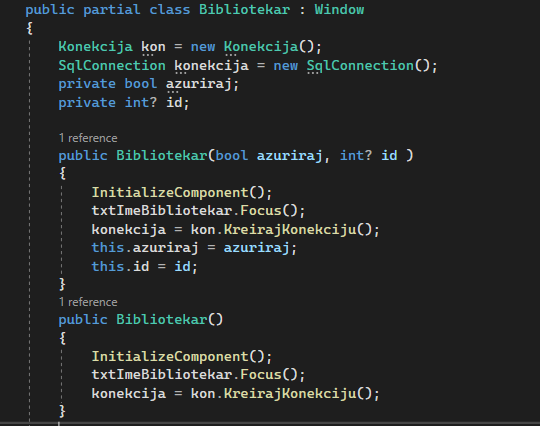


*Listing 6.2 – XAML kod kreiranih komponenti*



*Slika 6.3 – Vizuelni prikaz forme za dodavanje/izmenu bibliotekara*

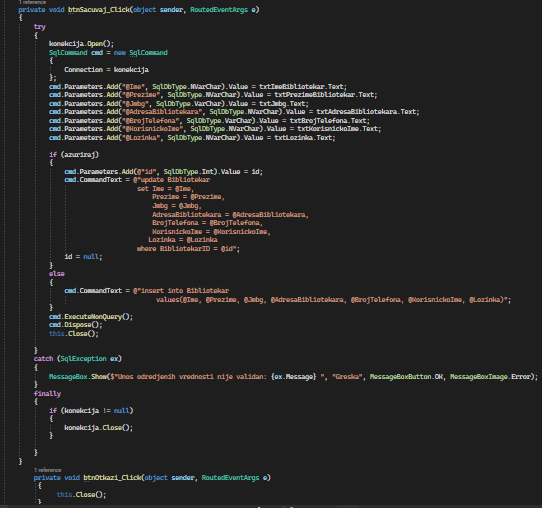
U Code Behind-u ove forme, nalazi se sledeća poslovna logika (listing 6.3). Najpre je potrebno uključiti imenski proctor System.Data.SqlClient. Zatim, potrebno je napraviti novi objekat SqlConnection klase, putem kojeg će se ostvarivati veza ka prethodno kreiranoj bazi podataka, nakon klika tasterom miša na dugme Sačuvaj. Takođe, u konstruktoru ovog prozora poželjno je podesiti metodu Focus(), koja pozicionira kursor na prvo polje koje korisnik popunjava pri pokretanju forme. U slučaju forme Bibiliotekar, to je tekst polje za unos imena bibliotekara.



*Listing 6.3 – Sadržaj CodeBehind-a forme Bibliotekar*

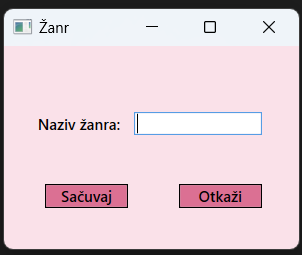
Ovakav prikaz pozadinskog koda gotovo je identičan za svaku od osam kreiranih formi. Ono u čemu je svaka od ovih formi jedinstvena jeste pozadinska funkcionalnost koja se dešava u trenutku kada korisnik tasterom miša klikne na dugme Sačuvaj. U listingu 6.4 prikazan je sadržaj izgenerisane metode btnSacuvaj\_Click.

Na samom početku, potrebno je otvoriti konekciju ka bazi podataka. Zatim, kreira se string koji u sebi sadrži insert into naredbu putem koje se dodaje novi objekat u bazu podataka. U odeljak values ove naredbe, prosleđuju se vrednosti koje su unete u TextBox-ove ove forme. Potrebno je voditi računa da se vrednosti string-ova ograde apostrofima. Nakon uspešno napravljene insert komande, ovaj string se prosleđuje objektu klase SqlCommand. Ovaj objekat, kao parametre prima prethodno kreirani string, ali i objekat klase Connection putem kojeg će znati u koju bazu podataka treba da upiše prosleđeni zapis. Nakon toga, komanda se izvršava putem metode ExecuteNonQuery(), nakon čega se prozor zatvara. Ovaj deo koda potrebno je ograditi try blokom iz razloga što postoji mogućnost nastanka greške usled prosleđivanja unosa pogrešnog tipa podatka od strane korisnika. Kako prilikom ovakvog unosa, program ne bi "pukao", potrebno je dodati catch blok unutar kojeg će se "hvatati" izuzetak tipa SqlException. Na kraju, u finally bloku, potrebno je postarati se da se prethodno otvorena konekcija sigurno zatvori. Ukoliko konekcija ostane otvorena, niko drugi neće moći da pristupi bazi podataka na isti način dok se ova instanca konekcije ne zatvori. Takođe, u okviru ovog listinga se nalazi i sadržaj izgenerisane metode btnOtkazi\_Click.

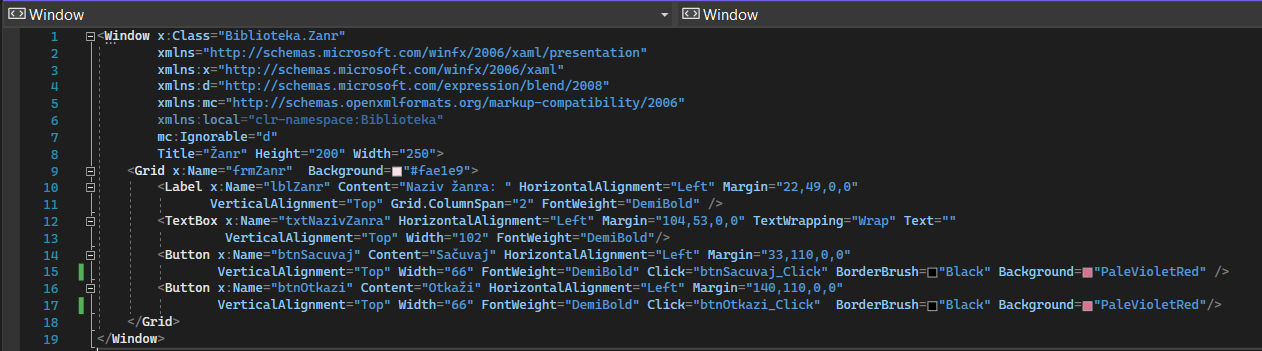


*Listing 6.4 Sadržaj metoda btnSacuvaj\_Click i btnOtkazi\_Click za dodavanje/izmenu bibliotekara*

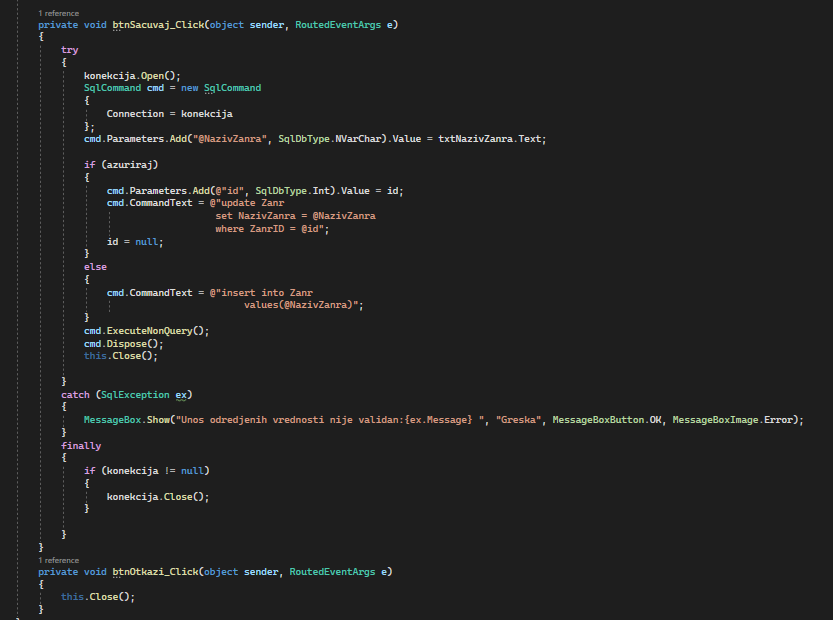
U nastavku će biti prikazan izgled i ostalih sedam formi.  
Forme za dodavanje autora, izdavača, žanra i člana imaju sličan dizajn i XAML kod te će biti prikazane na sledeće tri slike i šest listinga.



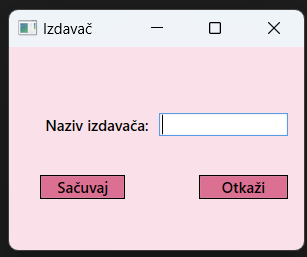
*Slika 6.3 Dizajn forme za dodavanje/izmenu žanra*



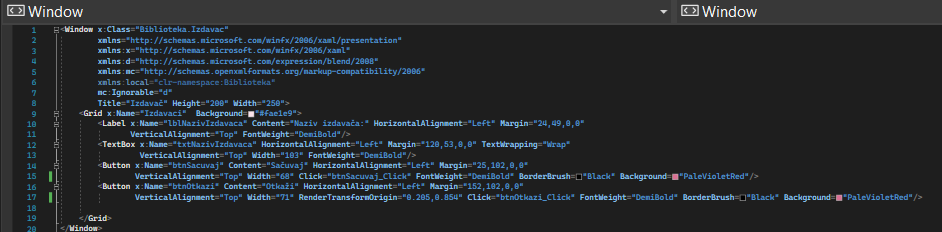
*Listing 6.5 – XAML kod forme za dodavanje/izmenu žanra*



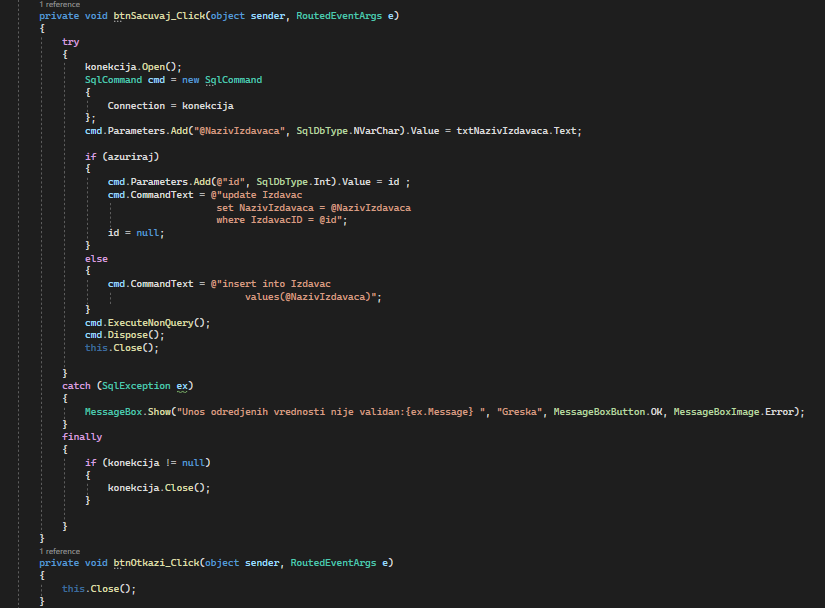
*Listing 6.6 - Sadržaj metoda btnSacuvaj\_Click i btnOtkazi\_Click za dodavanje/izmenu žanra*

**

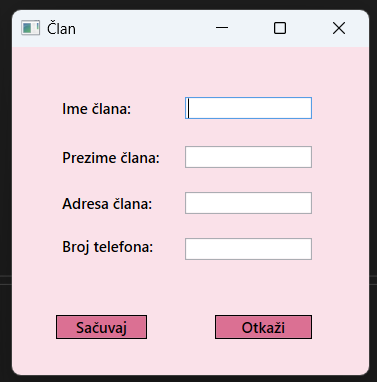
*Slika 6.4 -Dizajn forme za dodavanje/izmenu izdavača*



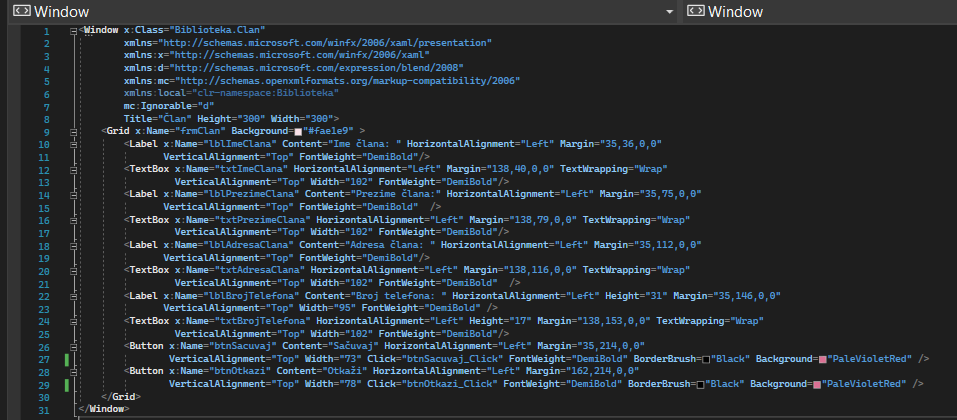
*Listing 6.7- XAML kod forme za dodavanje/izmenu izdavača*



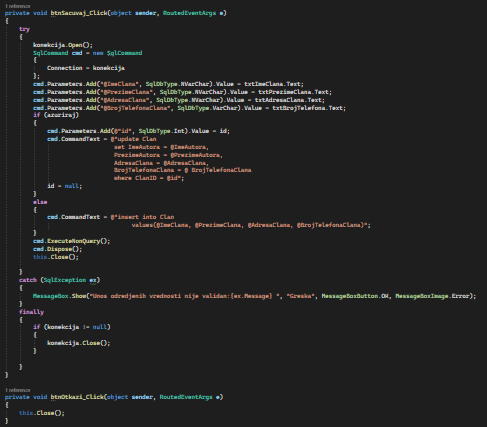
*Listing 6.8- Sadržaj metoda btnSacuvaj\_Click i btnOtkazi\_Click za dodavanje/izmenu izdavača*

**

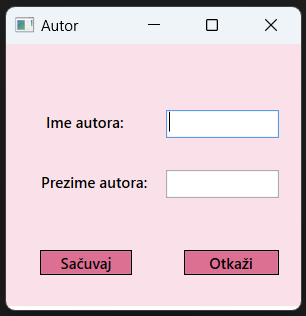
*Slika 6.5- Dizajn forme za dodavanje/izmenu članova*

**

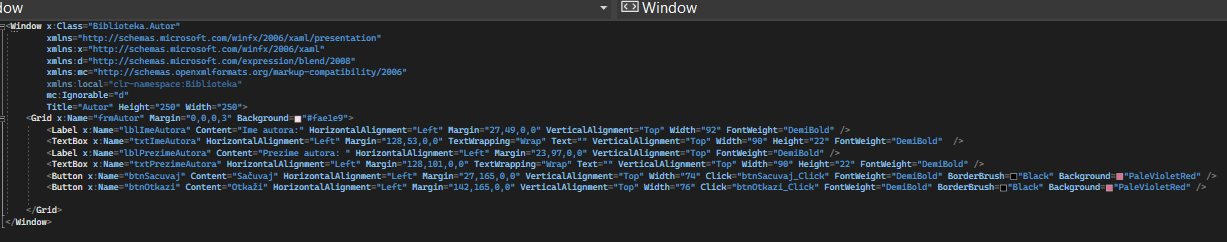
*Listing 6.9* *-XAML kod forme za izmenu/dodavanje člana*

**

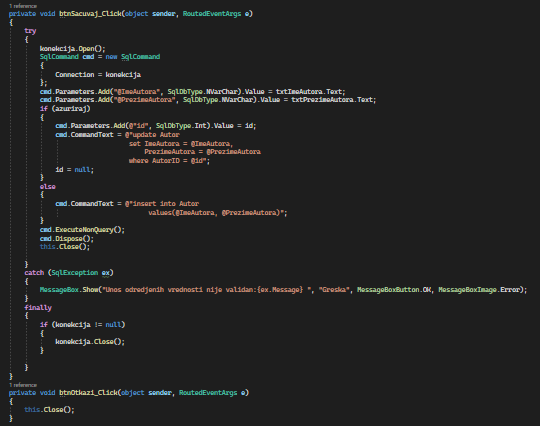
*Listing 6.10* -*Sadržaj metoda btnSacuvaj\_Click i btnOtkazi\_Click za dodavanje/izmenu člana*



*Slika 6.6 -Dizajn forme za dodavanje/izmenu autora*

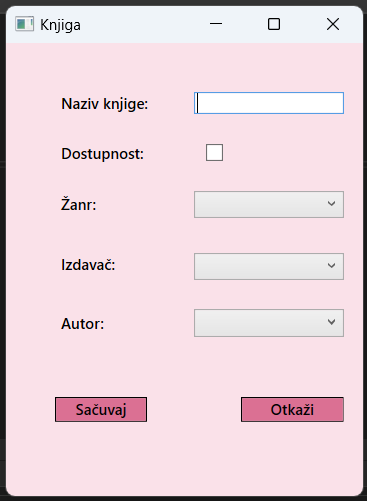


*Listing 6.11- XAML kod forme za izmenu/dodavanje autora*

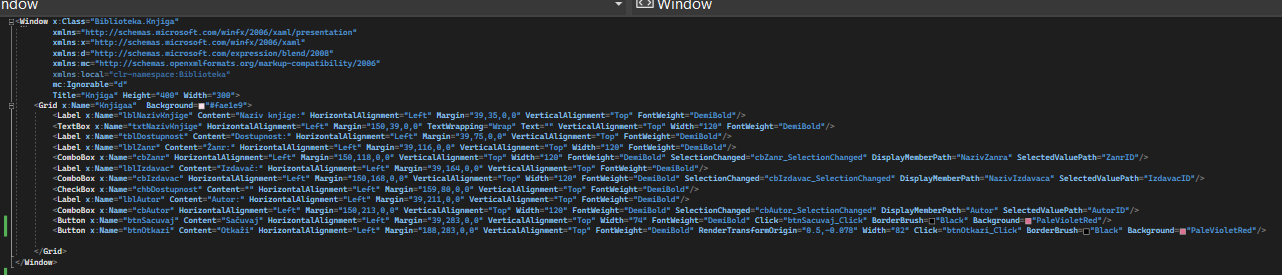
**

*Listing 6.12- Sadržaj metoda btnSacuvaj\_Click i btnOtkazi\_Click za dodavanje/izmenu autora*

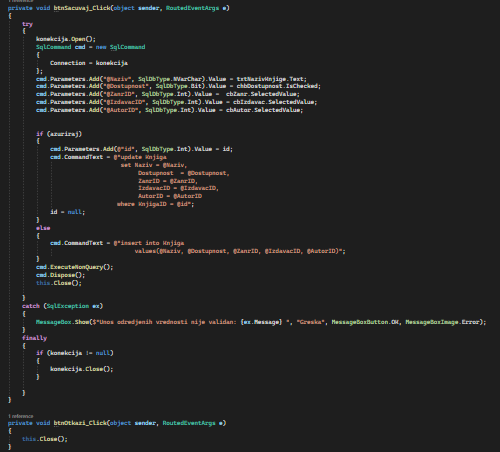
Tabele Knjiga, IzdavanjeKnjige, PovratKnjiga u sebi sadrže strane ključeve drugih tabela na osnovu kojih se pristupa sadržajima tih tabela. U samoj aplikaciji, potrebno je korisniku obezbediti vizuelni prikaz željenih obeležja neke od povezanih tabela. U WPF tehnologiji se to postiže korišćenjem ComboBox-a. Vizuelni prikaz forme za dodavanje nove knjige prikazana je na slici 6.7. Još jedna stvar koja sem stranih ključeva razlikuje ove tabele od ostalih jeste to što se u konstruktorima tabela poziva metoda Popuni iz klase PopunjavanjePolja koja služi za popunjavanje ComboBox-ova. Sadržaj metode prikazan je na Listingu 6.15.



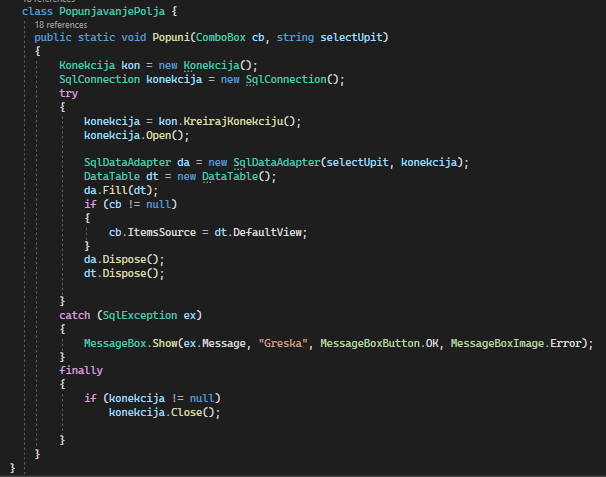
*Slika 6.7- Dizajn forme za dodavanje/izmenu knjige*



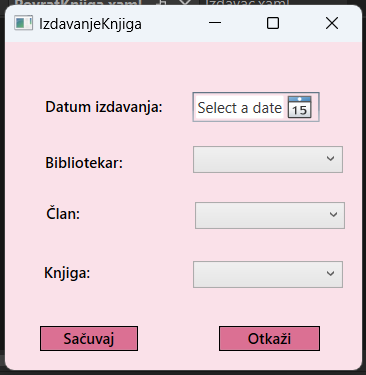
*Listing 6.13- XAML kod forme za dodavanje/izmenu knjige*

**

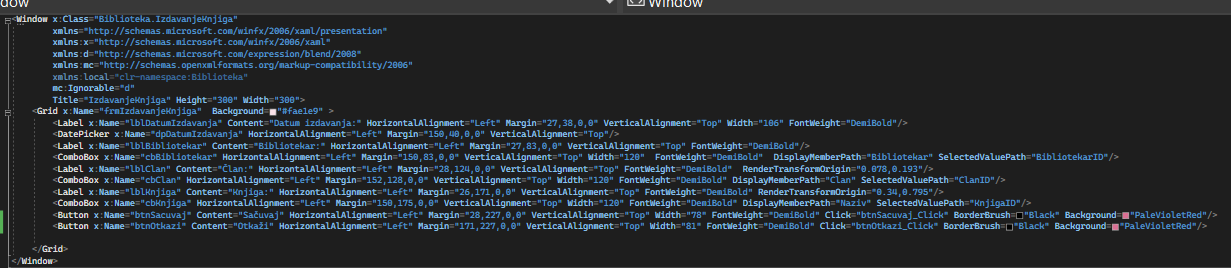
*Listing 6.14 -Sadržaj metoda btnSacuvaj\_Click i btnOtkazi\_Click za dodavanje/izmenu knjige*

**

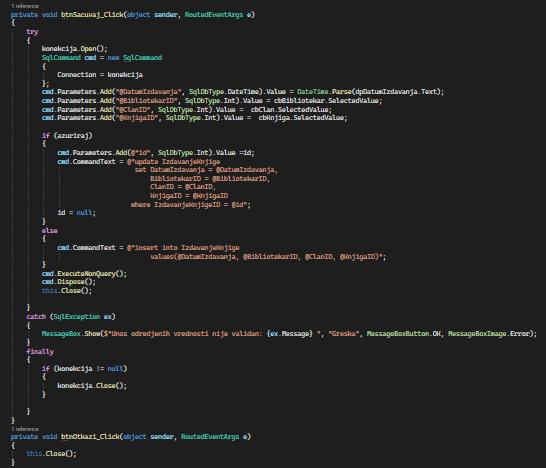
*Listing 6.15 – Sadržaj metode Popuni*

**

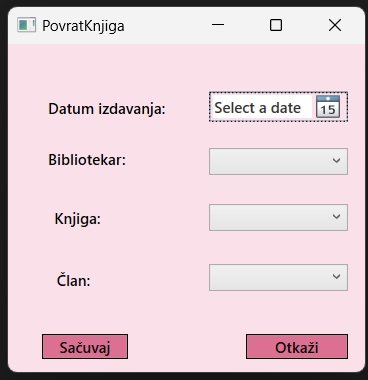
*Slika 6.8- Dizajn forme za dodavanje/izmenu podataka o izdavanjima knjige*

**

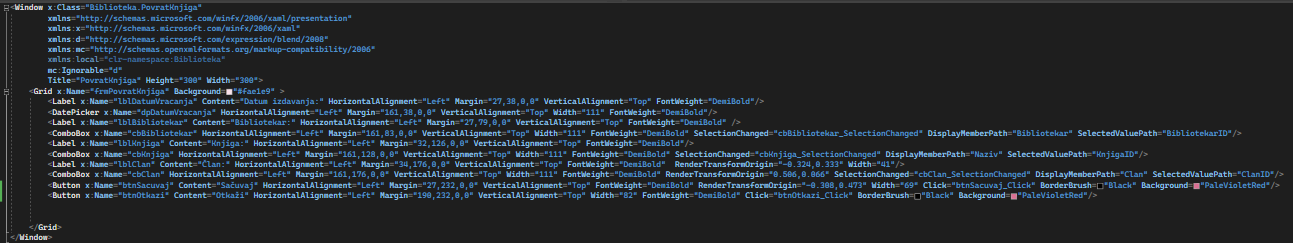
*Listing 6.16- XAML kod forme za izdavanje/izmenu podataka o izdavanjima knjige*

**

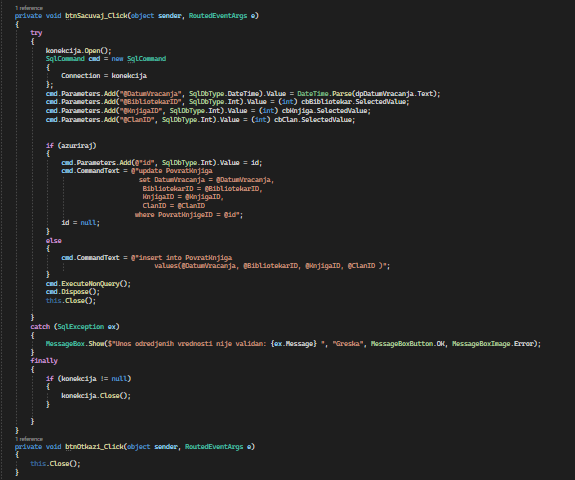
*Listing 6.17* *-Sadržaj metoda btnSacuvaj\_Click i btnOtkazi\_Click za izdavanje/izmenu podataka o izdavanju knjige*

**

*Slika 6.9- Dizajn forme za dodavanje/izmenu o povratu knjiga*

**

*Listing 6.18 -XAML kod forme za dodavanje/izmenu informacija o povratu knjiga*

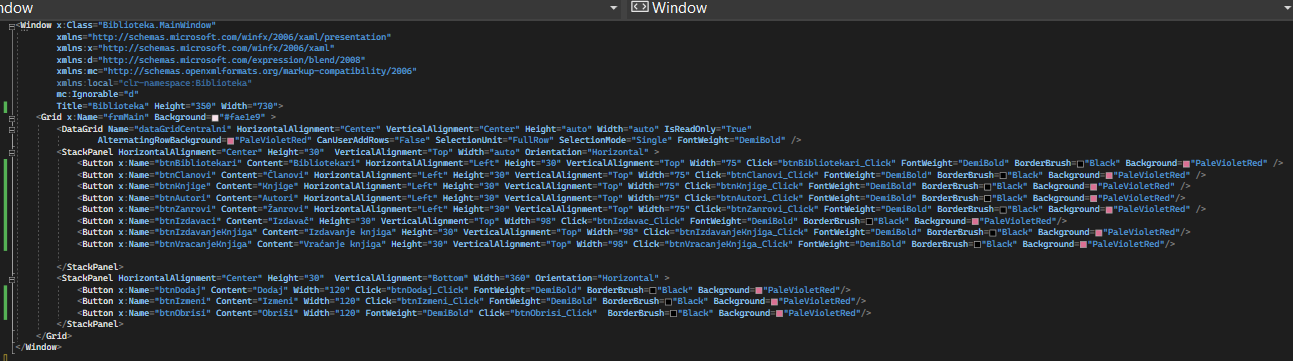
**

*Listing 6.19- Sadržaj metoda btnSacuvaj\_Click i btnOtkazi\_Click za izdavanje/izmenu podataka o povratu knjiga*

Nakon kreiranja formi za svaku od tabela, potrebno je nešto slično napraviti i za glavni prozor  
(eng. *Main Window*). Glavni prozor predstavlja prvi, inicijalni, kontakt korisnika sa aplikacijom  
i ujedno i osnovni prozor iz kojeg će se otvarati prethodno kreirane forme i u koji će se korisnik  
vraćati nakon zatvaranja tih formi. Na njemu bi trebalo da se nalazi dugme za svaku od osam  
kreiranih tabela u bazi, čijim će se klikom tastera miša, na sredini forme iščitavati ti podaci.  
Takođe, nalzi se 3 dugmeta, sa 3 operacije koje mogu da se izvrše sa ovim podacima. Operacije su dodavanje podataka, menjanje podataka kao i brisanje podataka. Dodavanjem odgovarajućih  
funkcionalnosti tim dugmićima u okviru glavne forme, obezbediće se izvršavanje osnovnih  
*CRUD* (eng. *Create, Read, Update, Delete*) operacija, što je i bio cilj ovog zadatka.

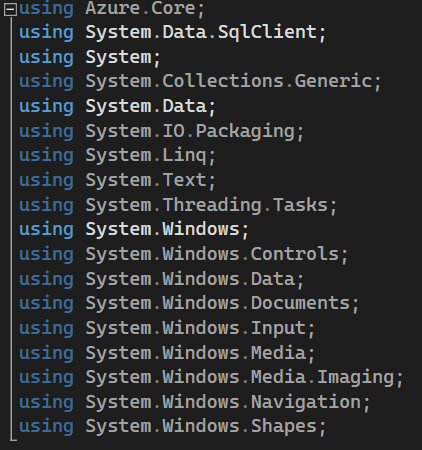


*Slika 6.10- Dizajn Main Window-a*

**

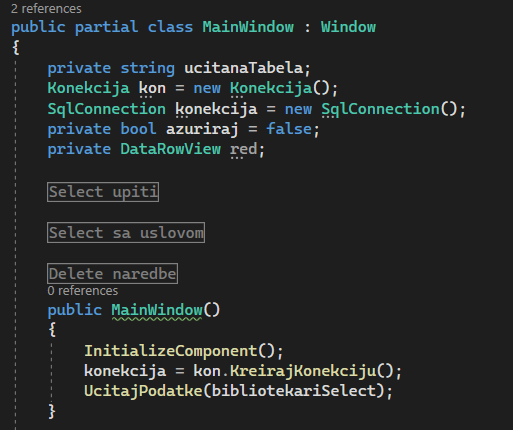
*Listing 6.20 -XAML kod Main Window-a*

Kako bi elementi unutar ovog prozora bili uredno raspoređeni, koristiće se poseban panel za njihovo smeštanje - StackPanel. U okviru gornjeg StackPanel-a, biće smešteni dugmići za izlistavanje tabela na centralnom gridu. Potrebno je podesiti poravnanja kao što je prikazano u listingu 6.17, a zatim izgenerisati metode koje će sprovoditi odgovarajuće akcije klikom tastera miša na to dugme. U nastavku će biti prikazana CodeBehind sadržina glavnog prozora, gde je smešten najveći deo poslovne logike. U sledećem listingu prikazani su korišteni imenski prostori.

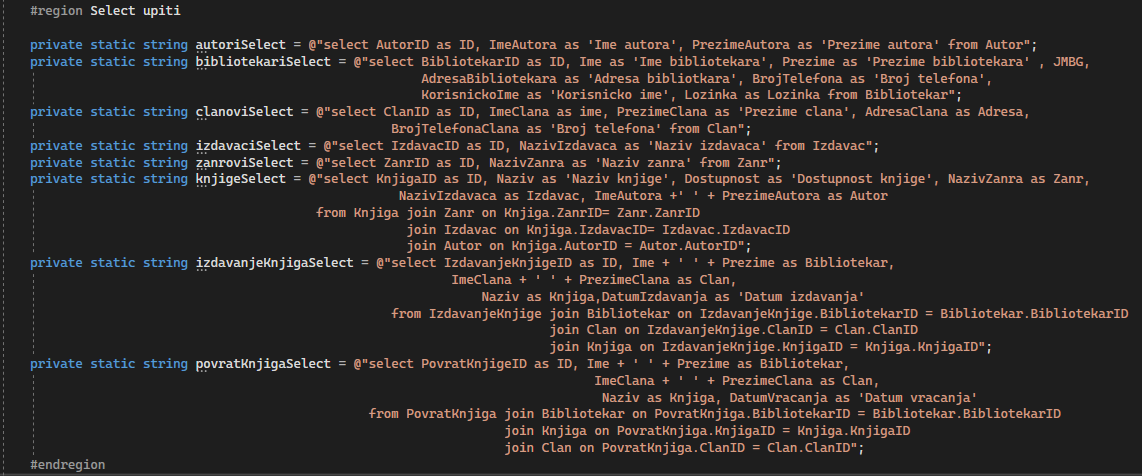


*Listing 6.21 – Imenski prostori za glavni prozor*

Za početak, sadržina glavnog prozora trebalo bi da izgleda kao u listingu 6.21. Sadržaće instancu konekcije ka bazi podataka, metodu za popunjavanje početnog grida pri pokretanju prozora, konstruktor za kreiranje novog prozora, kao i izgenerisane metode iz XAML koda, putem kojih će se, klikom tastera miša na odgovarajuće dugme, obavljati određene funkcionalnosti.



*Listing 6.22- Sadržaj CodeBehind-a glavnog prozora*



*Listing 6.23 - SQL kod Select upita*

**

*Listing 6.24 – SQL kod Select upita sa uslovom*

**

*Listing 6.25 – SQL kod Delete*

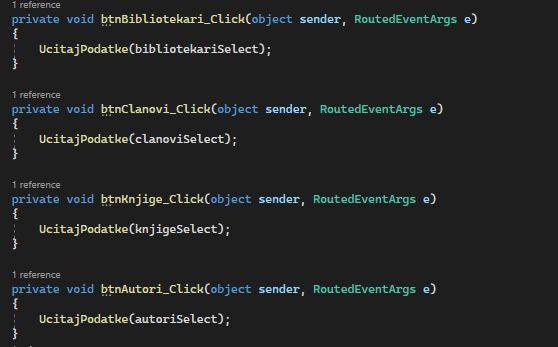
U prethodna tri listinga je prikazana komunikacija sa bazom podataka.

Nakon instanciranja konekcije, sledeća stvar je kreiranje metode koja će popunjavati početni DataGrid pri kreiranju novog prozora. U ovom primeru, tabela Bibliotekar će biti podrazumeumevana tabela, odnosno tabela čiji će se prikaz generisati pri pokretanju ovog prozora. U ovom slučaju, nakon otvaranja konekcije, prikazani upit se kreira na osnovu povezivanja pet tabela iz baze podataka. Zatim, na isti način kao kod popunjavanja ComboBoxova pri instanciranju prozora neke od formi, ovi podaci se smeštaju u odgovarajući DataSet kojeg zatim popunjava SqlDataAdapter. U konstruktoru se poziva ova metoda i prosleđuje joj se select upit za tabelu Bibliotekar.

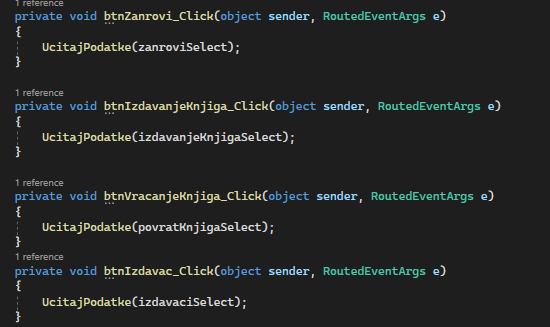


*Listing 6.26 – Sadrzaj metode UcitajPodatke*

Sledi pregled svake od posebnih metoda, koje se pozivaju nakon klika tasterom miša na odgovarajuće dugme iz gornjeg StackPanel-a, koje služe za ispis podataka iz tabele na centralnom gridu. Kao odgovor na klik korisnika i prosleđenih select upita, poziva se ova metoda i prosleđuje joj se odgovarajući upit.

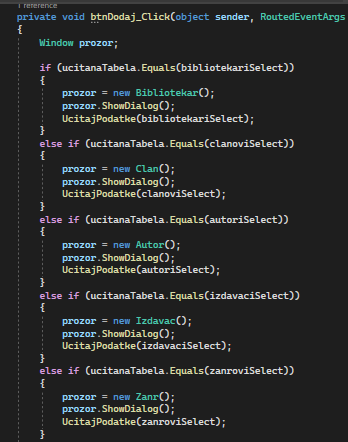


*Listing 6.26.1 – Prikaz svih metoda koje učitavaju podatke*

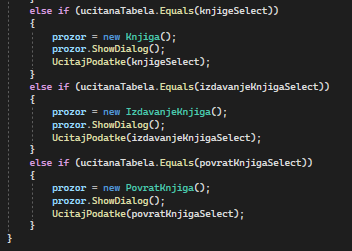


*Listing 6.26.2 – Prikaz svih metoda koje učitavaju podatke*

Sadržaj metoda generisanih na klik tasterom miša na dugme *Dodaj* je poprilično jednostavan.  
Klikom tastera miša na dugme *Dodaj*, instancira se prozor odgovarajuće forme i prikazuje  
korisniku na ekranu. Nakon zatvaranja tog prozora forme, osvežava se centralni grid sa  
aktuelnim podacima nakon eventualnog dodavanja novog zapisa u bazu. U listingu 6.27  
prikazan je sadržaj metode za dodavanje nove torke u odgovarajuću tabelu u bazi podataka.

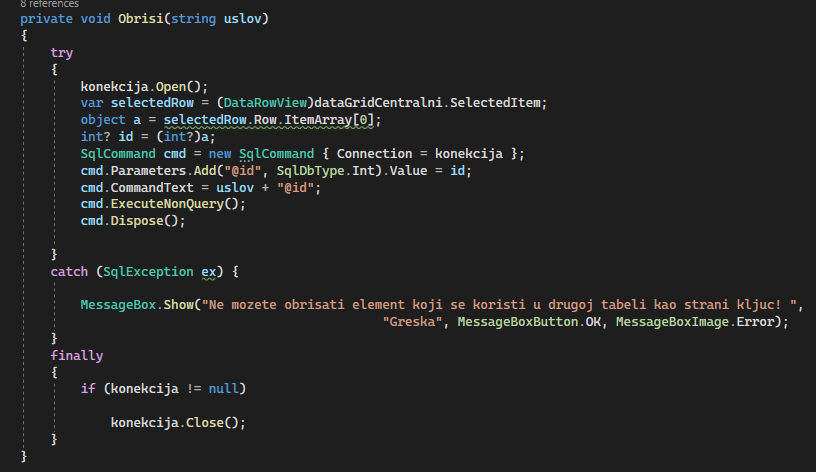


*Listing 6.27.1 – Sadržaj metode btnDodaj\_Click*

**

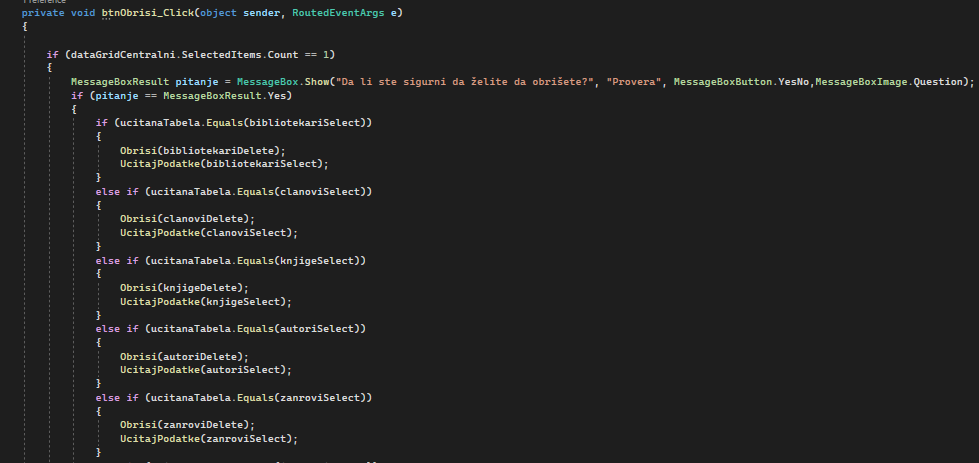
*Listing 6.27.2 – Sadržaj metode btnDodaj\_Click*

Brisanje podataka iz baze podataka obavlja se putem klika tasterom miša na dugme Obriši na način prikazan u listingu 6.28. Nakon otvaranja konekcije, kreira se objekat klase DataRowView koji u sebi čuva informaciju o indeksu reda koji je korisnik prethodno označio. Nakon toga kreira se Delete SQL komanda putem koje će se u samoj bazi obrisati željena stavka. Pre samog izvršavanja komande, pred korisnikom se pojavljuje dijalog prozor (MessageBox) putem kojeg on potvrđuje ili odustaje od brisanja. Izuzetak koji se javlja jeste taj da nije moguće obrisati element koji se u drugoj tabeli koristi kao strani ključ.

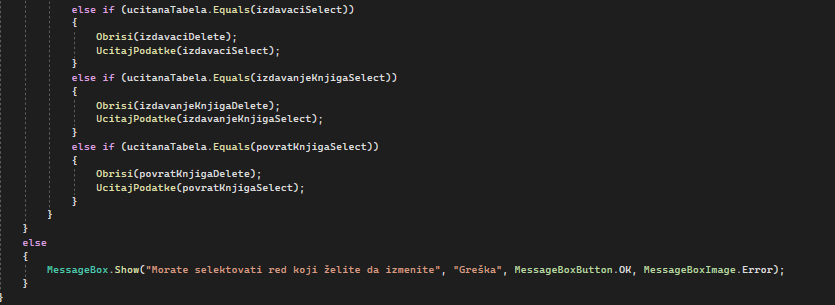


*Listing 6.28 – Sadržaj metode Obrisi*

Kada je tasterom miša kliknuto na dugme Obriši, u zavisnosti od toga koja tabela je trenutno učitana i koju stavku tabele je korisnik izabrao, metodi Obrisi se prosleđuju parametri i nakon toga se ponovo učitavaju podaci iz baze podataka, kako bi oni bili ažurni nakon brisanja označenog reda (listing 6.29). U okviru prvog if-a proveravamo koliko je selektovano redova, ukoliko je više ili manje od 1 redova selektovano pojavljuje se dijalog prozor (Message Box) sa odgovarajućom porukom. U okviru drugog if-a proveravamo da li je korisnik siguran da želi da obriše podatak.



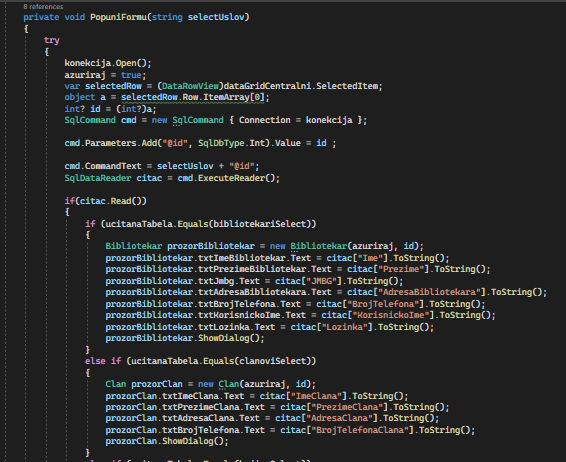
*Listing 6.29.1 Sadržaj metode btnObrisi\_Click*



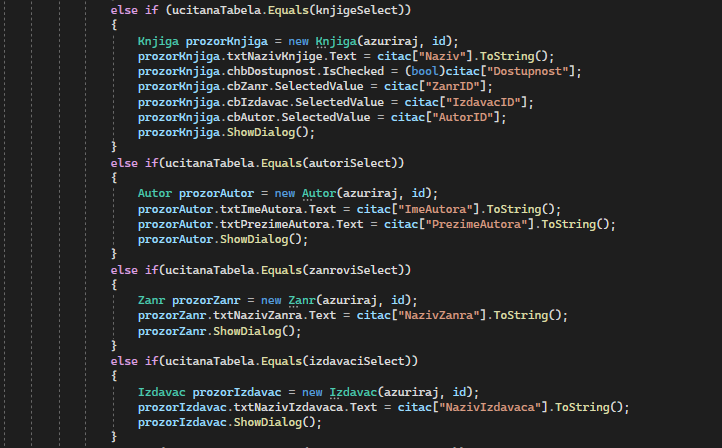
*Listing 6.29.2 Sadržaj btnObrisi\_Click*

Za modifikaciju podataka, kao i za dodavanje, potrebno je da postoji odgovarajuća forma u kojoj će korisnik menjati, odnosno unositi željene vrednosti. U ovom primeru, za svaku od tabela već je kreirana forma putem koje se novi zapisi upisuju u bazu podataka. Kako bi se izbeglo kreiranje novih formi specijalno za modifikaciju podataka, može se iskoristiti postojeća forma za dodavanje, s tim što će jedina razlika biti u tome koju akciju izvršava dugme Sačuvaj te forme kada korisnik tasterom miša klikne na njega. Na samom početku potrebno je, u okviru glavnog prozora deklarisati tri promenljive. Prva je tipa podatka boolean i koristiće se kao uslov koji će dugmetu Sačuvaj govoriti da li je potrebno podatke upisati u bazu podataka kao novi entitet ili izmeniti. Druga je tipa DataRowView i služi za pamćenje vrednosti označenog reda, kada iz glavnog prozora pređemo u neku drugu formu. Treća promenljiva, tipa podatka string, koja govori koja je trenutno tabela učitana.

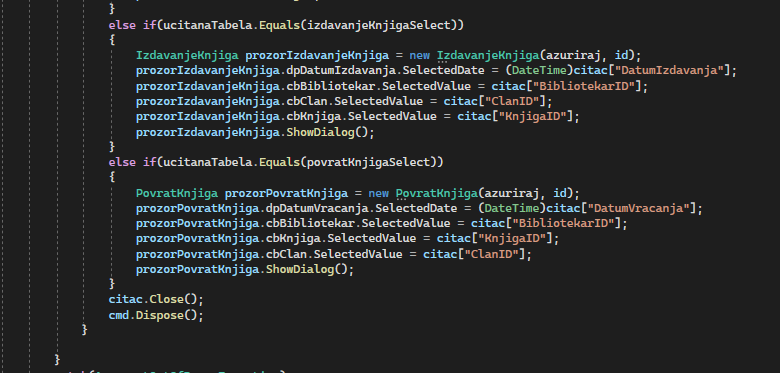
U naredna dva listinga prikazana je metoda koja se koristi za izmenu. Na samom početku, deklarisana promenljiva se postavlja na true. Nakon toga instancira se prozor njene forme i otvara konekcija. Pomoću DataRowView objekta, kao i kod brisanja, pamti se koji red je označen. Razlika je u tome što se sada, u drugu promenljivu (označen) dodaje ta vrednost kako bi se iskoristila u drugoj formi. Nakon toga se, pomoću select upita kojem se u where uslovu zadaje vrednost primarnog ključa zapisa koji se modifikuje, potrebni podaci smeštaju u odgovarajuća polja forme, pomoću SqlDataReader-a. Nakon toga, poziva se metoda prozora ShowDialog i on se prikazuje na ekranu sa popunjenim vrednostima traženog zapisa. U okviru ove metode, u finally bloku, konekcija se zatvara, a promenljiva ažuriraj, vraća na false, u slučaju da, nakon izmene, korisnik želi da doda novi zapis u bazu podataka.



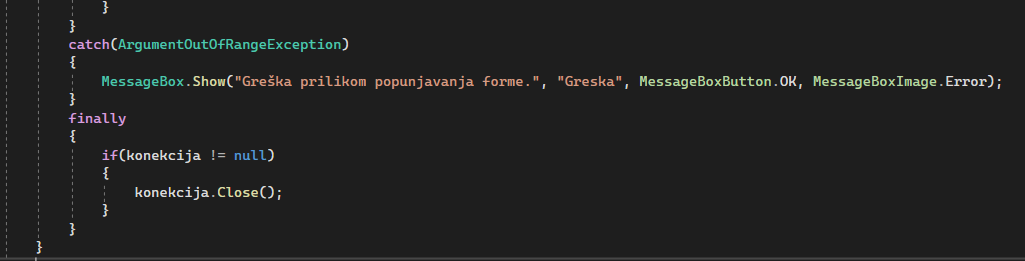
*Listing 6.30.1 Sadržaj metode PopuniFormu*



*Listing 6.30.2 Sadržaj metod PopuniFormu*

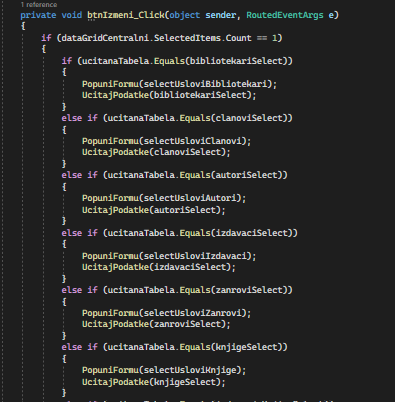


*Listing 6.30.3 Sadržaj metode PopuniFormu*

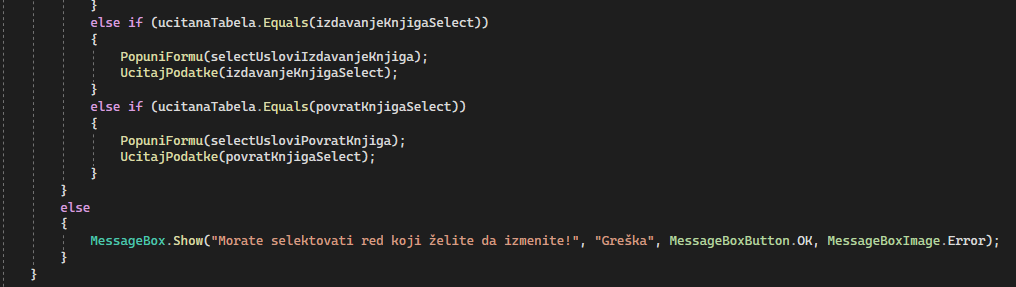
**

*Listing 6.30.4 Sadržaj metode PopuniFormu*

Kada Korisnik izabere stavku koju želi da izmeni, poziva se metoda btnIzmeni\_Click koja u zavisnosti od učitane tabele prosleđuje odgovarajući parameter metodi PopuniFormu i nakon izvršene ismene ponovo učitava podatke kako bi podaci bili ažurni.



*Listing 6.31.1 Sadržaj btnIzmeni\_Click*



*Listing 6.31.2 Sadržaj btnIzmeni\_Click*

# Testiranje aplikacije

Kada je aplikacija kreirana i kada su implementirane sve neophodne funkcionalnosti koje ona treba da podrži, neophodno je testirati aplikaciju kako bi se proverila ispravnost tih funkcionalnosti. U ovom poglavlju dat je prikaz testiranja WPF aplikacije za podršku poslovanja biblioteke.

Pri pokretanju aplikacije, generiše se početni prozor kao na slici 7.1.



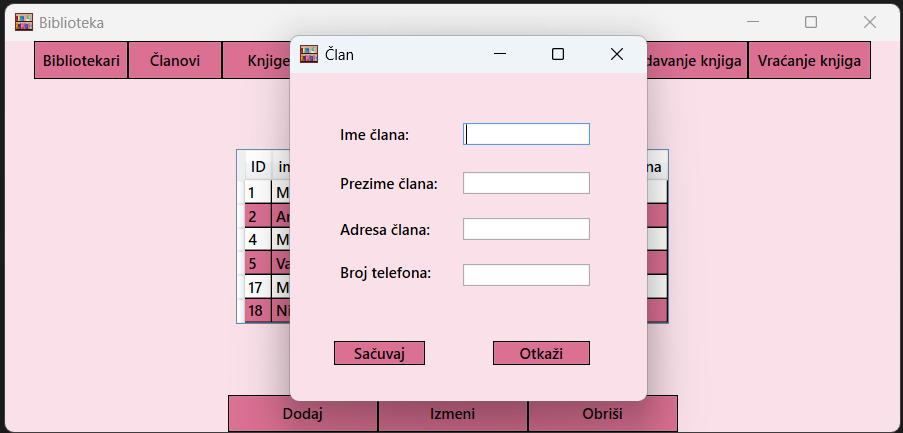
*Slika 7.1– Početni prozor pri pokretanju aplikacije i prikaz podataka iz tabele Bibliotekar*

Klikom tastera miša na dugmiće iz centralnog menija vrši se prikaz podataka željene tabele na centralnom gridu.



*Slika 7.2 – Prikaz podataka iz tabele Članovi*

Klikom tastera miša na dugme ***Dodaj***, otvara se forma za dodavanje novog zapisa u bazu podataka. U formi se klikom na taster ***Sačuvaj***, sačuvavaju odnosno upisuju podaci u bazu i tabela se, nakon zatvaranja forme, automatski ažurira.

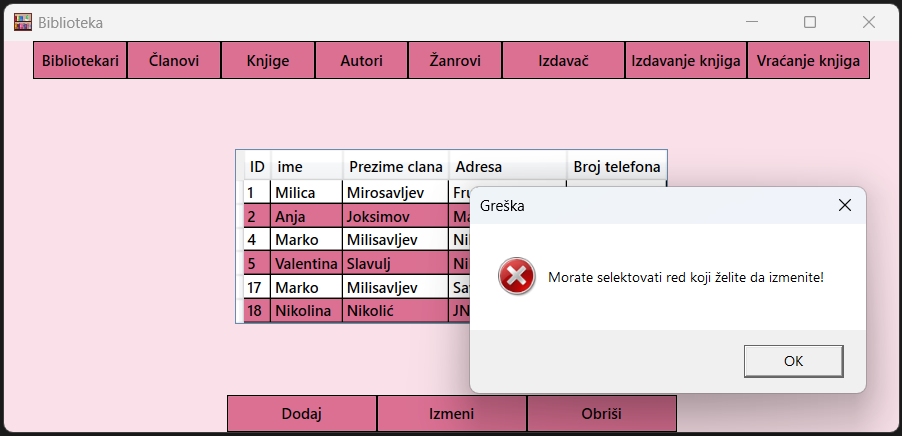


*Slika 7.3 – Dodavanje zapisa u bazu podataka*

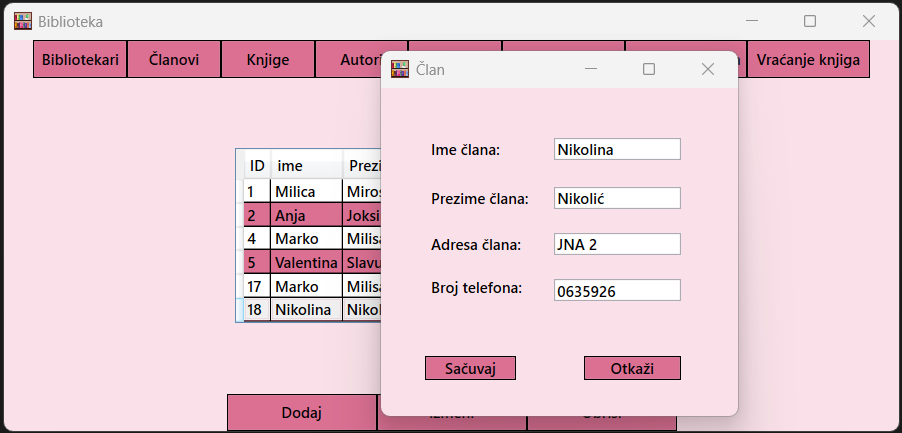


*Slika 7.4 – Dodavanje zapisa u bazu podataka*

Označavanjem odgovarajućeg reda otvara se forma sa popunjenim vrednostima prethodnog zapisa koje korisnik moze modifikovati. Ukoliko Korisnik klikne na dugme Izmeni, a pritom nije izabrao red, onda će mu program ukazati na tu grešku.

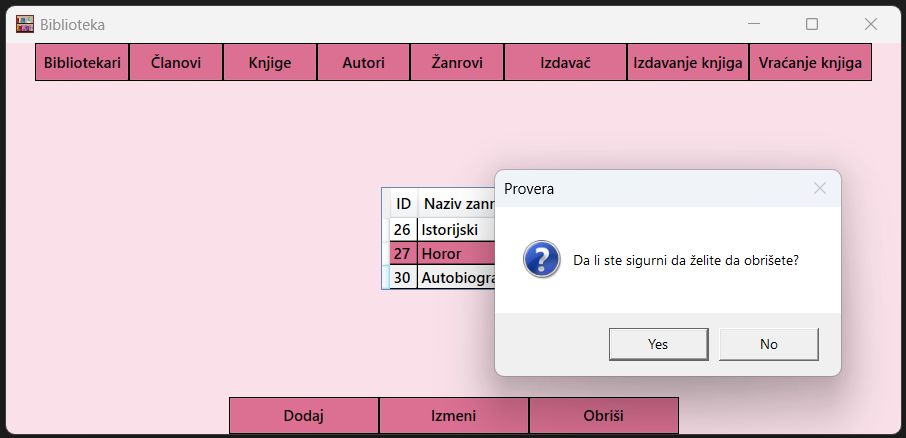


*Slika 7.5 – Prikaz greške prilikom neselektovanja reda*



*Slika 7.6 – Prikaz forme sa prethodno učitanim podacima*

Brisanje podataka iz tabele se vrši tako što se prvo selektuje red, pa potom klikne na dugme Obriši. Nakon toga će program pitati korisnika da li je siguran da želi da obriše podatak i na osnovu odgovora će izvršiti željenu operaciju.



*Slika 7.7 – Prikaz forme prilikom brisanja podataka*

# Zaključak

Izradom ovog projekta stekla sam osnovno znanje o funkcionisanju poslovnih sistema, ponovila znanje što se tiče izrade baza podataka i naučila kako da kreiram WPF aplikacije. Rad na projektu mi je bio veoma zanimljiv, pa sam ga stoga radila sa lakoćom i bez opterećenja. Mislim da ću se u nekoj skorijoj budućnosti mnogo više posvetiti samom kreiranju WPF aplikacija.