1. **Opis realnog sistema**

Projekat "Teretana" napravljen je da simulira kako funkcioniše process vežbanja u samoj teretani. Teretana je mesto čija je osnovna svrha vežbanje i akcenat na vođenju zdravog života.

Sam proces se realizuje tako što se budući član prvo registruje u sam sistem teretane, potom dobije svoju člansku karticu na osnovu koje se vrši njegova identifikacija.

Zaposleni u teretani su dužni za izdavanje samih članskih kartica kao i za za vođenje evidencije o članovima.

1. **Zadatak**

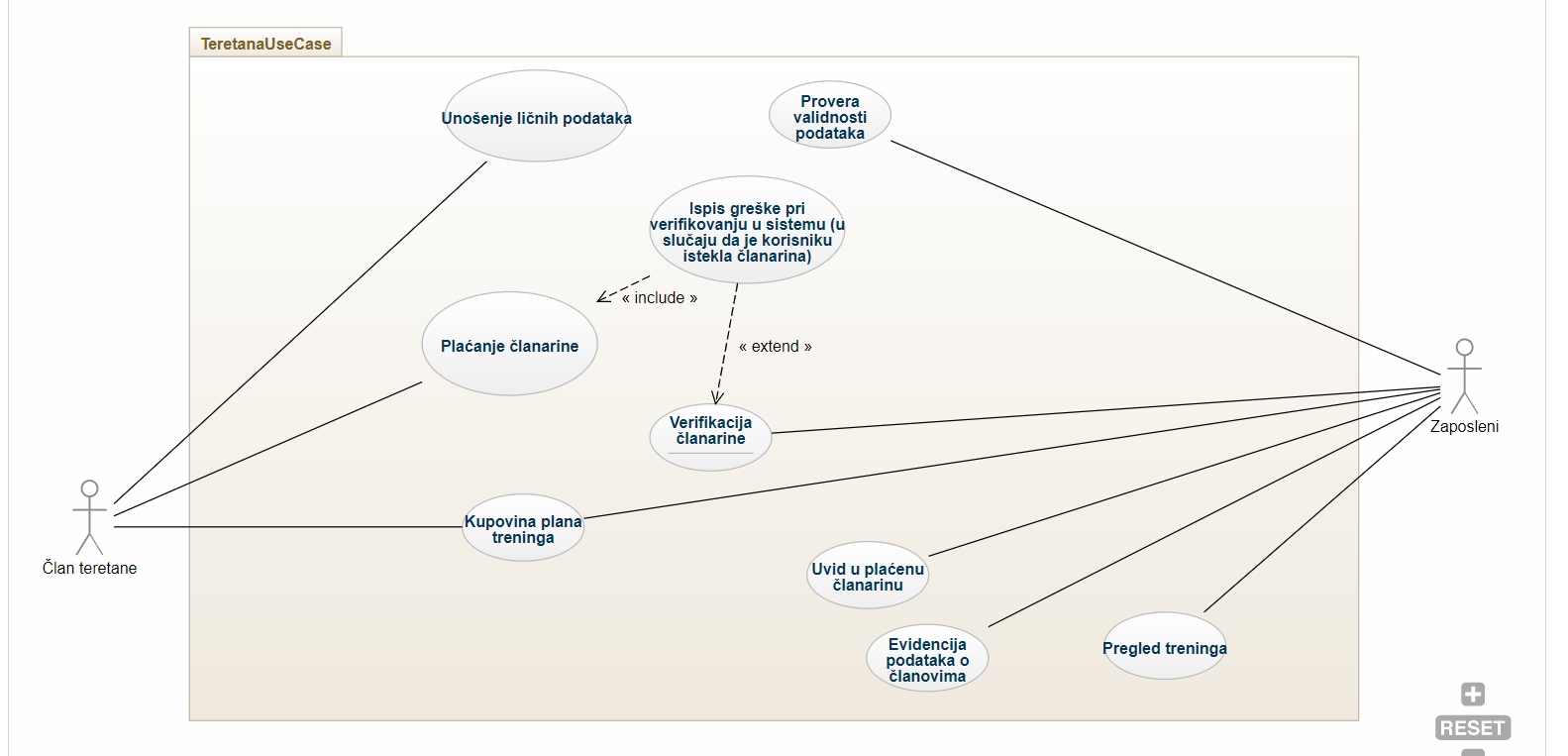
Zadatak ovog projekta je da se kreira desktop aplikacija koju će koristiti samo zaposleni u teretani. Osnovne mogućnosti aplikacije su vođenje evidencija o članiovima,zaposlenima i

spravama za vežbanje.

1. **Opis tehnologija**

Sajt GenMyModel je korišćen za izradu UML dijagrama (use case dijagram, class dijagram, sequence dijagram). Aplikacija je napravljena u Microsoft VisualStudio IDE, kod je pisan u C# programskom jeziku. Za čuvanje podataka u bazi je korišćen SQL server i Microsoft SQL Server Management Studio. Unutar Microsoft Visual Studio kreirana je sama WPF (Windows Presentation Foundation) aplikacija. WPF predstavlja grafički podsistem koji se koristi za kreiranje aplikacija sa korisničkim interfejsom baziranom na XAML-u (Extensible Application Markup Language).

4.**UML**



-Tekstualni opis slučaja upotrebe:

**Slučaj upotrebe:** Unošenje ličnih podataka

**Kratak opis:** Unošenje ličnih podataka

**Učesnici:** Korisnik, zaposleni

**Uslovi:** Podaci moraju biti pravilno unešeni

**Opis:** Korisnik unosi podatke radi registrovanja u sistem teretane(ime,prezime,broj kreditne kartice,e-mail, godine,matični broj I sl.))

**Izuzeci:** U slučaju da korisnik unese nepravilno podatke, neće biti uneti u sistem

**Uslovi koji moraju biti zadovoljeni nakon izvršavanja:** Lični podaci moraju biti evidentirani u sistem

**Slučaj upotrebe:** Verifikacija članarine

**Kratak opis:** Provera isteka članarine

**Učesnici:** Zaposleni

**Uslovi:** Nema specifičnih uslova

**Opis:** Zaposleni proverava validnost važenja članarine, tj. da li je/nije istekla.

**Izuzeci:** nema

**Uslovi koji moraju biti zadovoljeni nakon izvršavanja:** Čekiranje u sistemu da je kartica uspešno verifikovana (u slučaju da je verifikovanje prošlo uspešno), odnosno, ukoliko kartica nije verifikovana, korisnik je u obavezi da istu uplati.

**Slučaj upotrebe:** Plaćanje članarine

**Kratak opis:** Plaćanje članarine za članstvo u teretani od strane korisnika

**Učesnici:** Korisnici

**Uslovi:** Nema specifičnih uslova

**Opis:** Korisnik uplaćuje mesečnu članarinu u naznačenom iznosu.

**Izuzeci:** Ukoliko korisnik ne plati članarinu, kartica mu neće biti verifikovana

**Uslovi koji moraju biti zadovoljeni nakon izvršavanja:** Uplatu korisnika evidentirati u system I čekirati validnost kartice.

**Slučaj upotrebe:** Kupovina plana treninga

**Kratak opis:** Korisnik kupuje plan treninga od strane zaposlenog

**Učesnici:** Korisnici,zaposleni

**Uslovi:** Neophodno je da korisnik bude prethodno registrovan u sistem teretane

**Opis:** Korisnik kupuje plan treninga sa specifičnom svrhom (fizička sprema, skidanje kilaže, bolja funkcionalnost)

**Izuzeci:** Nema izuzetaka

**Uslovi koji moraju biti zadovoljeni nakon izvršavanja:** Evidentirati kupljeni plan treninga u sistem kako bi korisnik mogao da mu pristupi u bilo kom periodu.

**Slučaj upotrebe:** Pregled treninga

**Kratak opis:** Korisnik pregleda redosled treninga za dati period

**Učesnici:** Korisnici

**Uslovi:** Nema specifičnih uslova

**Opis:** Korisnik pregleda koji deo tela radi u kojem danu kao I dane odmora od istog

**Izuzeci:** Nema izuzetaka

**Uslovi koji moraju biti zadovoljeni nakon izvršavanja:** Nema specifičnih uslova

**Slučaj upotrebe:** Uvid u plaćenu članarinu

**Kratak opis:** Zaposleni pregleda status plaćene članarine

**Učesnici:** Zaposleni

**Uslovi:** Neophodno je da je korisnik prvi put uplatio članarinu.

**Opis:** Zaposleni pregleda da li je članarina plaćena I ako jeste, proverava da li je rok plaćene članarine istekao

**Izuzeci:** Ukoliko u uvidu piše da korisnik nije platio članarinu, neophodno je da istu I uplati kako bi mogao koristiti samu teretanu.

**Uslovi koji moraju biti zadovoljeni nakon izvršavanja:** Ukoliko je korisnik platio članarinu, istu mu verifikovati.

**Slučaj upotrebe:** Ispis greške pri verifikovanju u sistemu (u slučaju da je korisniku istekla članarina)

**Kratak opis:** Ispis greške

**Učesnici:** Korisnici

**Uslovi:** Korisnik nije platio članarinu

**Opis:** Korisniku se ispisuje greška ukoliko mu je članarina istekla I naglašava mu se da istu uplati.

**Izuzeci:** Nema izuzetaka

**Uslovi koji moraju biti zadovoljeni nakon izvršavanja:** Nema specifičnih uslova

**Slučaj upotrebe:** Evidencija podataka o članovima

**Kratak opis:** Zaposleni proverava podatke članova/korisnika

**Učesnici:** Zaposleni

**Uslovi:** Nema specificiranih uslova

**Opis:** Zaposleni provera status članova, tj. Uvid u njihove članarine

**Izuzeci:** Nema specificiranih izuzetaka

**Uslovi koji moraju biti zadovoljeni nakon izvršavanja:** Ukoliko je nekome od članova istekla članarina,

Neophodno mu je naglasiti da je obnovi.

**Slučaj upotrebe:** Provera validnosti podataka

**Kratak opis:** Zaposleni pregledava unete podatke člana

**Učesnici:** Zaposleni

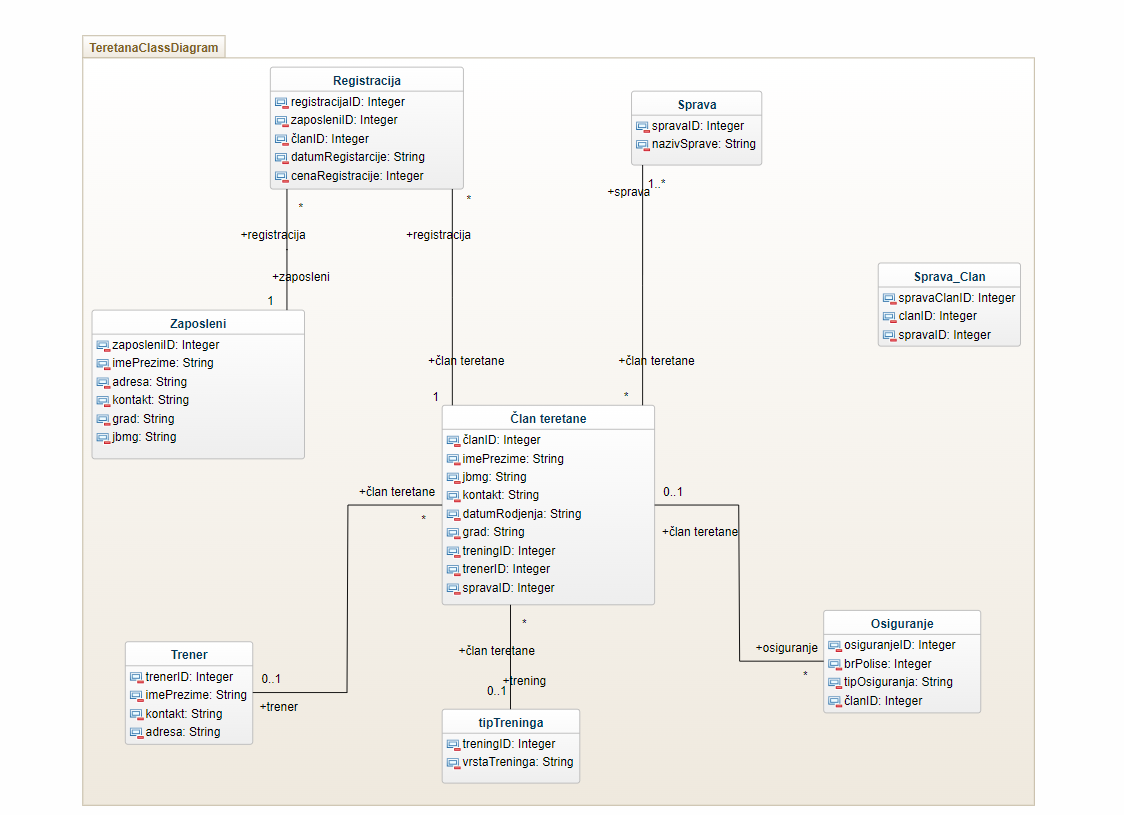
**Uslovi:** Neophodno je da se korisnik prethodno registruje

**Opis:** Zaposleni pregleda da li su podaci koje je korisnik uneo validni.

**Izuzeci:** Ukoliko uneti podaci nisu validni, korisnik se ne može registrovati u sistem teretane.

**Uslovi koji moraju biti zadovoljeni nakon izvršavanja:** Ukoliko su podaci validni, vrši se dalji process registraicje.

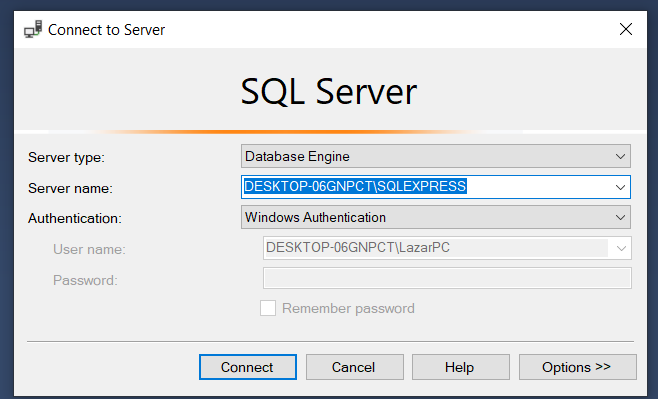
**Class diagram:**



Dijagram klasa služi nam da jednostavnije prikažemo bazu podataka I pomogne nam u njenoj izradi. Mogu se videti veze između klasa kao I podaci vezani za svaku klasu.

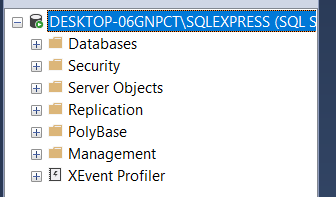
5.**Baza podataka**

Na samom početku, potrebbi je pokrenuti SQL Server Management Studio Express. Kada se to uradi, pojaviće se prozor sa dijalogom, kao što je prikazano na slici 1.1 . U prikazanom dijalogu, potrebno je podesiti parameter Server type I Authentication, kao što je prikazano na slici 1.2. Napomena: Server name nije potrebno menjati, ono predstavlja naziv uređaja na kojem se kreira baza podataka.



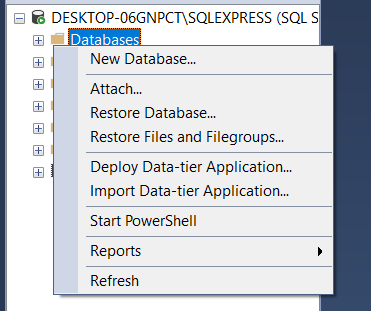
Slika 1.1 – Ostvarivanje konekcije na server

Nakon podešavanja parametara, 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 1.2



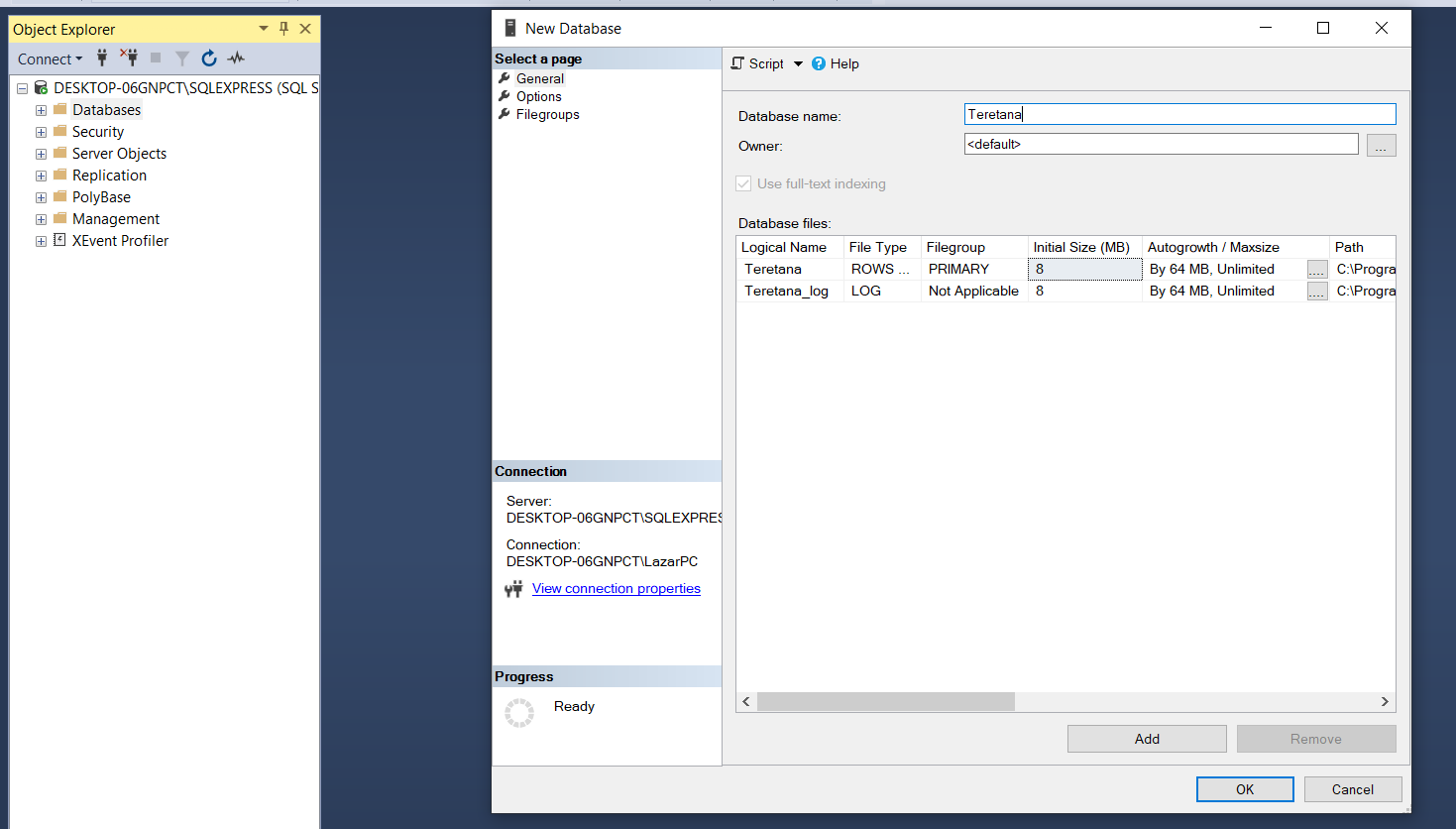
Slika 1.2 – Izgled Object Explorer-a nakon ostvarivanja konekcije sa bazom podataka

Sada je potrebno kreirati bazu podataka za aplikaciju. Baza se kreira tako što se klikne desnim klikom tastera miša na Database u treeview-u koji se nalazi sa leve strane prozora u okviru Object Explorer-a. Otvoriće se lista u kojoj je potrebno označiti New DataBase(slika 1.3).



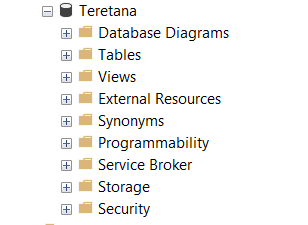
Slika 1.3 – Kreiranje baze podataka

Novi prozor će ponuditi opciju za kreiranje baze podataka. Potrebno je dati naziv bazi, u ovom slučaju naziv će biti Teretana. Ostale parameter nije potrebno menjati. Kliknuti OK (slika 1.4)



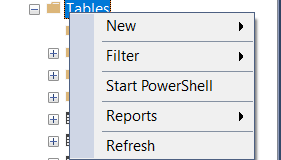
Slika 1.4 – Podešavanje parametara pri kreiranju baze podataka

Posle uspešnog kreiranja baze podataka u Object Explorer-u, u folderu Database se pojavljuje kreirana baza podataka Teretana (slika 1.5)



Slika 1.5 – Prikaz kreirane baze podataka

Proširivanjem baze Teretana u stablu Object Explorer-a pojavljuje se više datoteka. Desnim klikom tastera miša kliknuti na datoteku tables I odabrati Table (slika 1.6). Nakon klika na ovu opciju, sa desne strane otvara se prozor za definisanje tabele.



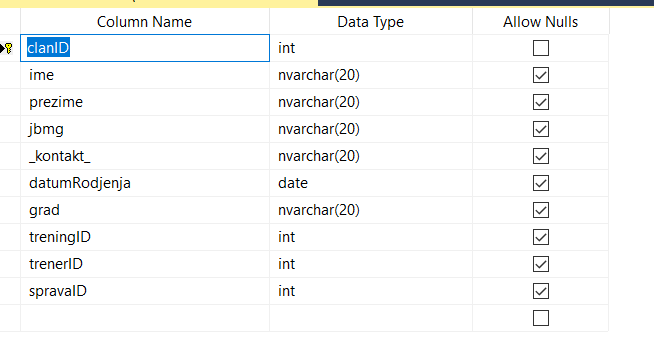
Slika 1.6 – Kreiranje tabele unutar baze podataka

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 imati null vrednosti. Preporučuje se da Allow Nulls bude označeno za sve kolone, osim za primarni ključ, kako bi se izbegli određeni problem prilikom upisa rekorda u tabelu (slika 1.7)



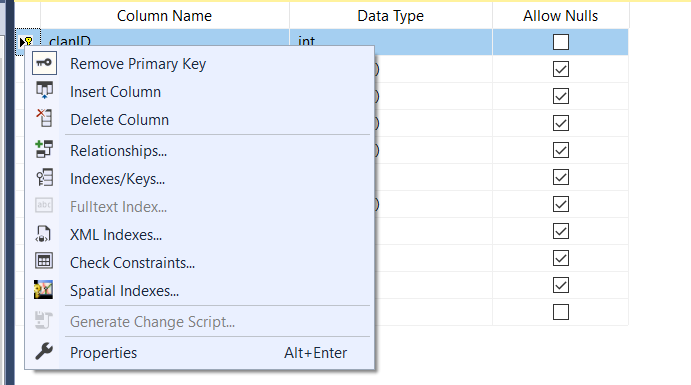
Slika 1.7 – Dodavanje obeležja unutar tabele

Svaka tabela neke baze podataka mora posedovati primarni ključ – ID I bilo koja kolona može da se deklariše kao primarni ključ. U ovom slučaju, kolona clanID 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 1.8).



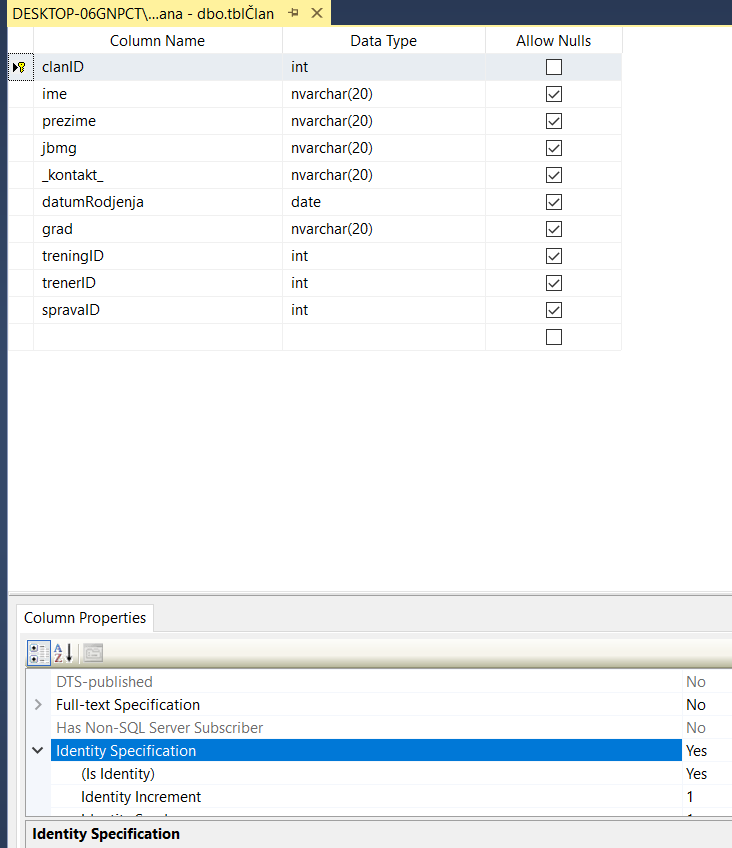
Slika 1.8 – Prikaz kreiranih obeležja

Da bi se kolona clanID deklarisala kao primarni ključ, potrebno je kliknuti desnim klikom tastera miše na crnu strelicu koja se nalazi pored naziva ove kolone. Otvoriće se lista u kojoj treba odabrati opciju Set Primary Key (slika 1.9). Ovom komandom se kolona clanID deklariše kao primarni ključ.



Slika 1.9 – Definisanje kolone clanID za primarni ključ

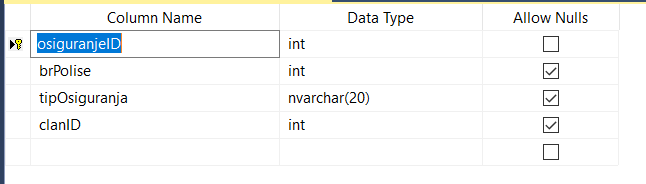
Da bi SUBP automatski kreirao primarni ključ, potrebno je u Column Properties pronaći opciju Identity Specification. Proširivanjem opcije Identity Specification dobija se opcija Is Identity, koju treba promeniti na Yes. Ovde je, takođe, moguće podesiti parameter Identity Increment I Identity Seed. Identity Increment-om se definiše za koliko raste sledeća vrednost ključa, dok Identity Seed označava od koje vrednosti počinju da se memorišu ključevi (slika 2.0).



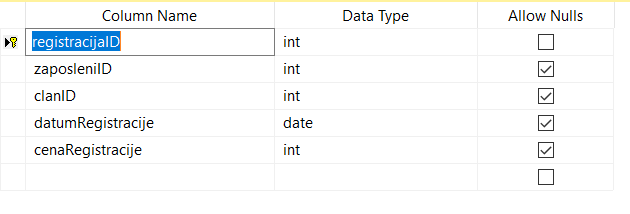
Slika 2.0 -Podešavanja automatske inkrementacije vrednosti obeležja primarnog ključa

Čuvanje tabele vrši se pritiskanjem tastera Ctrl I S (Ctrl + S) nakon čega će SQL Server Management Studio pitati pod kojim nazivom se želi sačuvati tabela. Naziv tabele se najčešće navodi sa prefiksom “tbl” u ovom slučaju tblČlan.

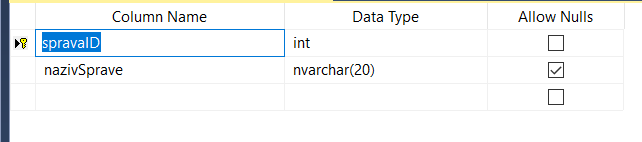
Isti process bi trebalo ponoviti za kreiranje tabela tblOsiguranje, tblRegistracija, tblSprava, tblSprava\_Clan, tblTipTreninga, tblTrener, tblZaposleni (slika 2.1 – 2.7).



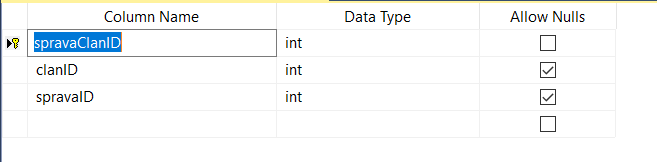
Slika 2.1 – Kreiranje tabele tblOsiguranje



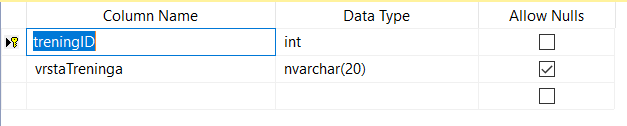
Slika 2.2 – kreiranje tabele tblRegistracija



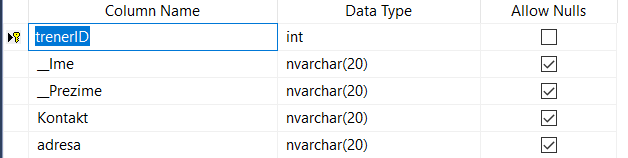
Slika 2.3 – kreiranje tabele tblSprava



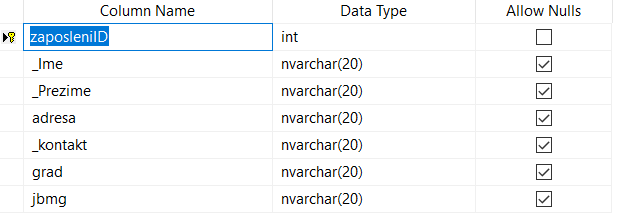
Slika 2.4 – kreiranje tabele tblSprava\_Clan



Slika 2.5 – kreiranje tabele tblTipTreninga

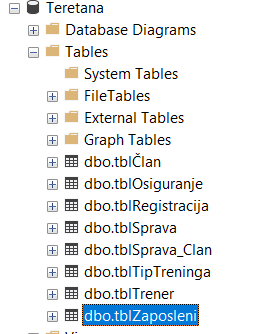


Slika 2.6 – kreiranje tabele tblTrener



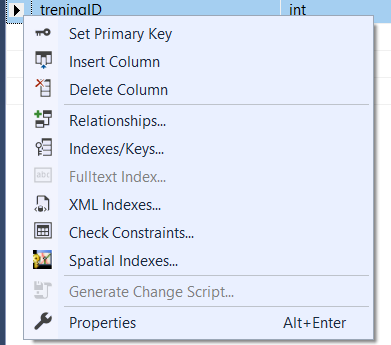
Slika 2.7 – kreiranje tabele tblZaposleni

Nakon toga bi trebalo u treeview-u, koji se nalazi sa leve strane prozora u okviru Object Explorer-a, trebalo da se nalazi baza podataka Teretana, u kojoj će se, pored već kreirane tblČlan, naći I novokreirane tabele sa odgovarajućim atributima (slika 2.8).



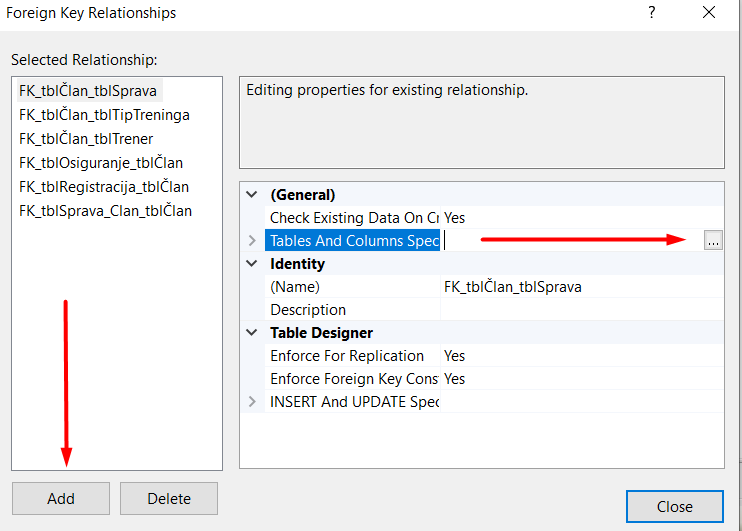
Slika 2.8 – Object Explorer prikaz kreiranih tabela I kolona

U tblČlan se nalaze 3 strana ključa (treningID, trenerID,spravaID). Prilikom unosa atributa u tabelu tblČlan potrebno je uneti I attribute koji imaju nazive treningID, trenerID, spravaID, 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 2.9).



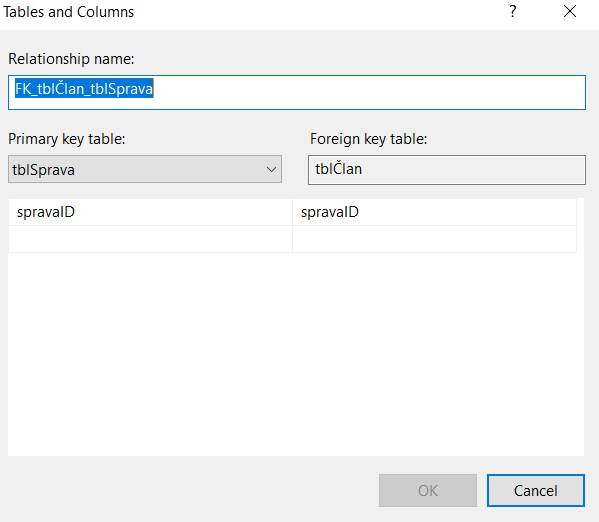
Slika 2.9 – Dodavanje veza

Nakon odabira opcije Relationships, otvara se prozor Foreign key Relationships (slika 3.0) u kojem je potrebno prvo kliknuti tasterom miša na dugme Add, a zatim proširiti polje Tables and Columns Specification I kliknuti na tri tačke koje se nalaze desno.



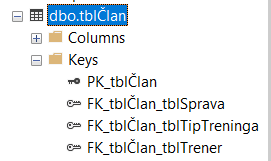
Slika 3.0 Dodavanje stranog ključa

U otvorenom prozoru Tables and Columns potrebno je u okviru Foreign key table, konkretno iz tabele tblČlan, 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 izabrati ključ te tabele sa kojim se povezuje željeni atribut. Nakon što je sve ovo urađeno, prozor treba da izgleda kao na sledećoj slici (3.1).



Slika 3.1 – Kreiranje veza između tabela

Dodavanjem Relationships-a atributima treningID, trenerID,spravaID u tabeli tblČlan, oni postaju strani ključevi. Da li je povezivanje tabela uspešno odrađeno, znaće se tako što se proveri da li se oni nalaze u treeview-u u okviru tabele tblČlan, pod kategorijom Keys (slika 3.2).

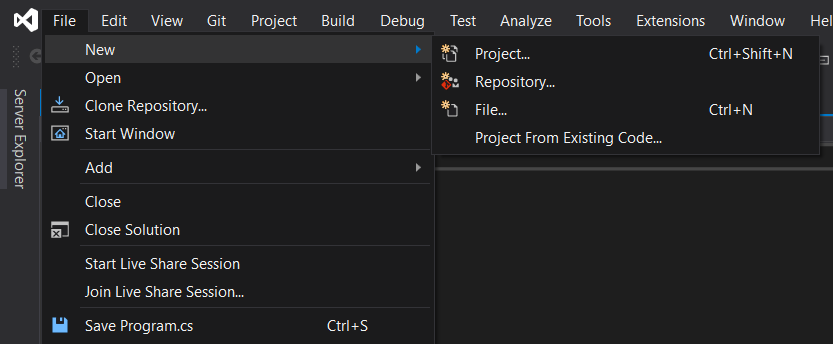


Slika 3.2 – Prikaz kreirane tabele I dodavanje podataka

Baza podataka kreirana je na način koji je opisan u ovom poglavlju. Sledeći korak u izradi projektnog zadatka je kreiranje WPF aplikacije.

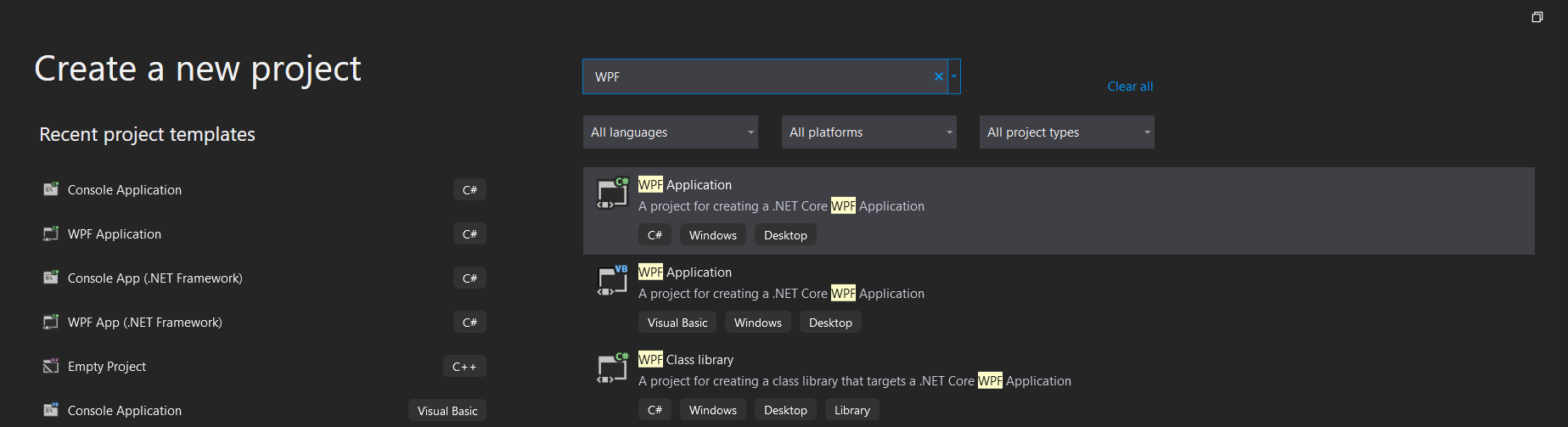
**6. Kreiranje WPF aplikacije**

Nakon pokretanja razvojnog okruženja VisualStudio, potrebno je tasterom miša kliknuti na File menu, a zatim odabrati New Project (slika 3.3).



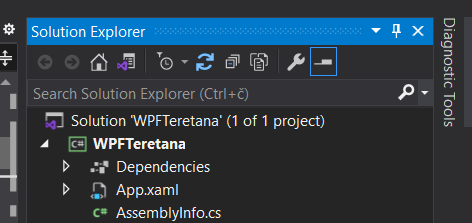
Slika 3.4 – Kreiranje novog projekta u Visual Studiju

Odabrati Windows Classic Desktop, zatim WPF Application. Navesti ime aplikacije I kliknuti OK (slika 3.4).



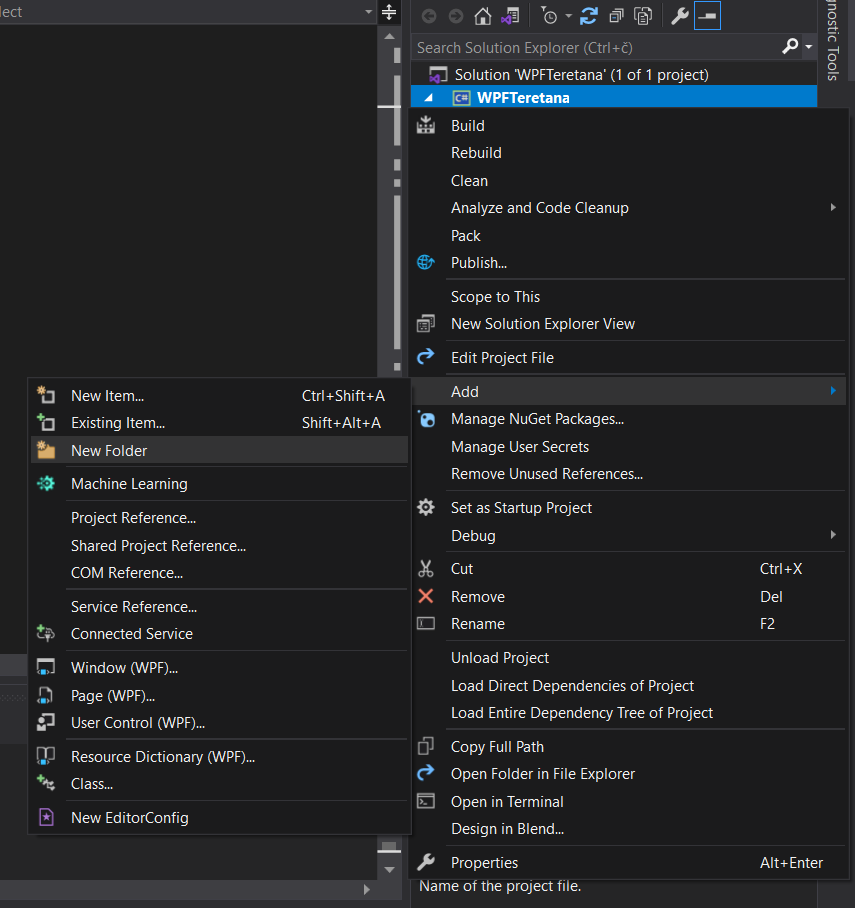
Slika 3.4 – Kreiranje WPF aplikacije

U solution Explorer-u pojaviće se napravljena aplikacija (slika 3.5).



Slika 3.5 – Solution Explorer kreirane WPF aplikacije

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 (WPFTeretana), a zatim Add/Class (slika 3.6).

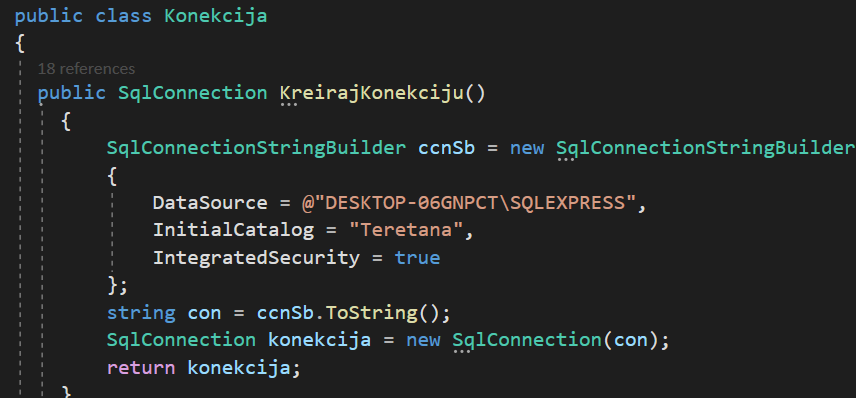


Slika 3.6 – Kreiranje klase

Novokreiranoj klsi se dodeljuje naziv “Konekcija” I njen sadržaj prikazan je na slici 3.7. Na samom početku, potrebno je uključiti imenski proctor 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 instance klase SqlConnection. Unutar same metode prikazana su podešavanja parametara konekcije I to:

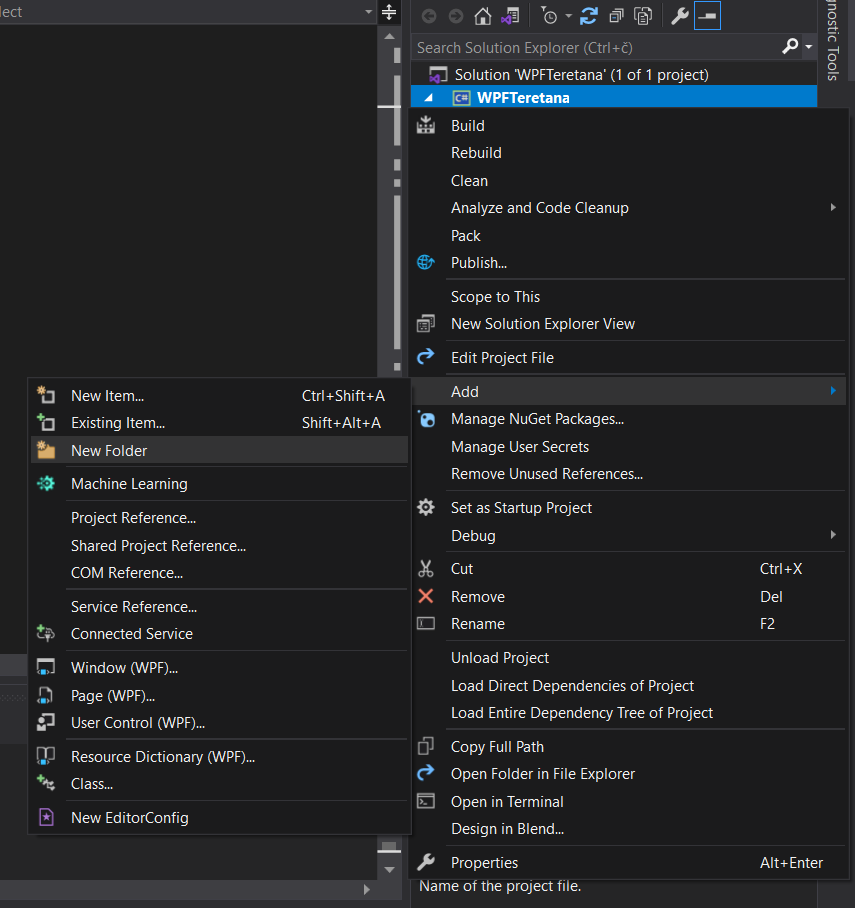
1. DataSource – Naziv servera na kojem je baza podataka smeštena,
2. InitialCatalog – Naziv baze podataka kojoj je potrebno pristupiti, i
3. IntegratedSecurity – Ukoliko se baza nalazi na lokalnoj mašini, ovaj parameter postavlja se na true.

Nakon podešavanja ovih parametara, ceo string se prosleđuje instance klase SqlConnection.



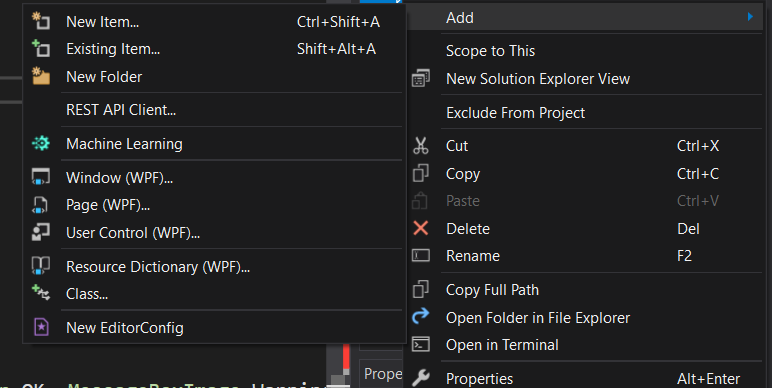
Slika 3.7- Klasa Konekcija

Sledeći korak u izradi WPF aplikacije je kreiranje formi za dodavanje objekata u bazu podataka. Kako bi sve forme bile uredno smeštene na jednom mestu, potrebno je kreirati folder pod nazivom Forme u kom će se nalaziti svih sedam formi za, prethodno kreiranih, sedam tabela u bazi podataka. Kreiranje foldera se vrši na sličan način kao kreiranje klase, desnim klikom tastera miša na projekat, a zatim Add/New Folder (slika 3.8).



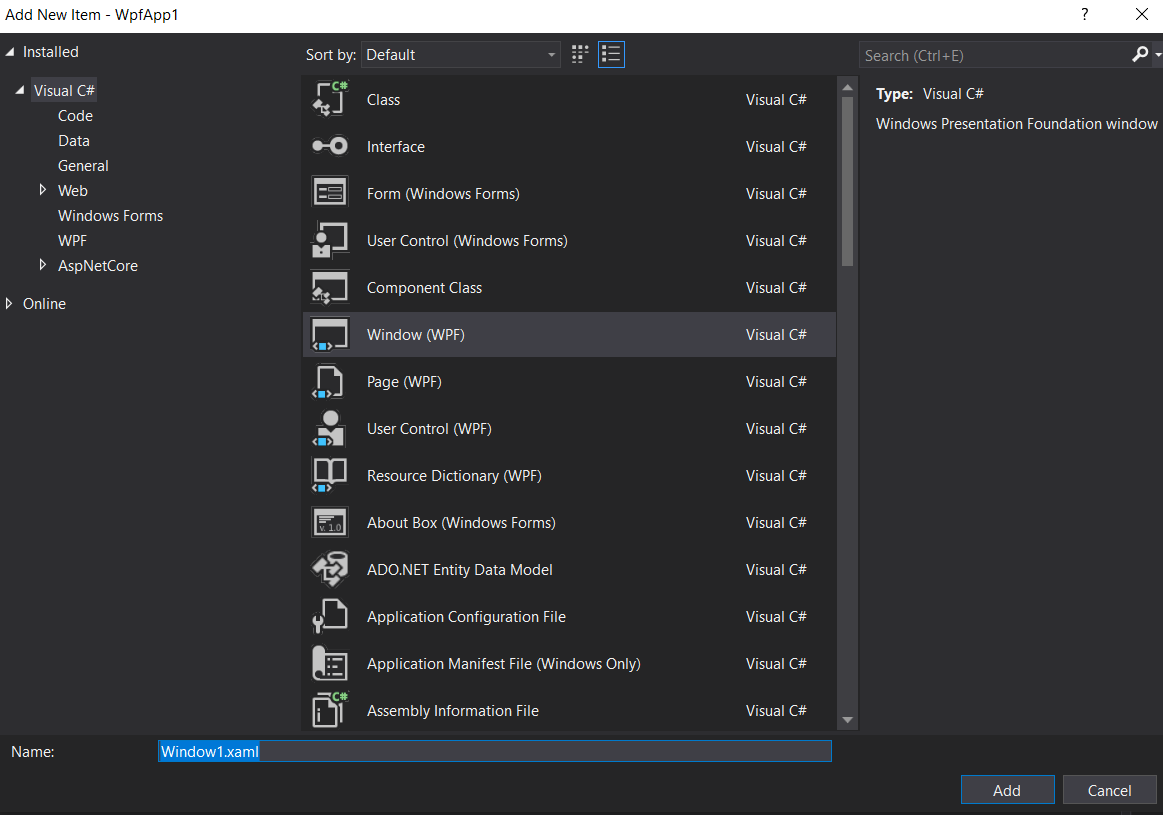
Slika 3.8 – Kreiranje novog foldera.

Nakon kreiranja foldera, potrebno je dodati novi prozor za svaku od formi. WPF prozor, unutar folder, kreira se desnim klikom tastera miša na folder, a zatim Add/Window (Slika 3.9).



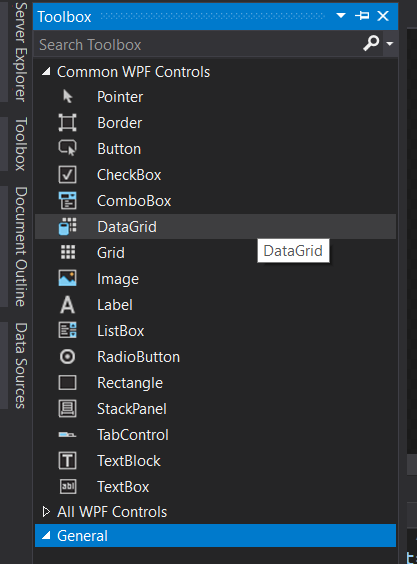
Slika 3.9- DOdavanje formi unutar foldera

Nakon toga prikazaće se I prozor za dodavanje novog Item-a, unutar kog je potrebno podesiti naziv I zatim kliknuti na Add (slika 4.0).



Slika 4.0 – Dodavanje novog prozora

Kada je novi prozor izgenerisan, potrebno je unutar njega, dodati određene elemente putem kojić se se ostvarivati potrebne funkcionalnosti. Na levoj strani prozora, klikom na Toolboxm prikazaće se lista svih ugrađenih WPF komandi (slika 4.1). 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.



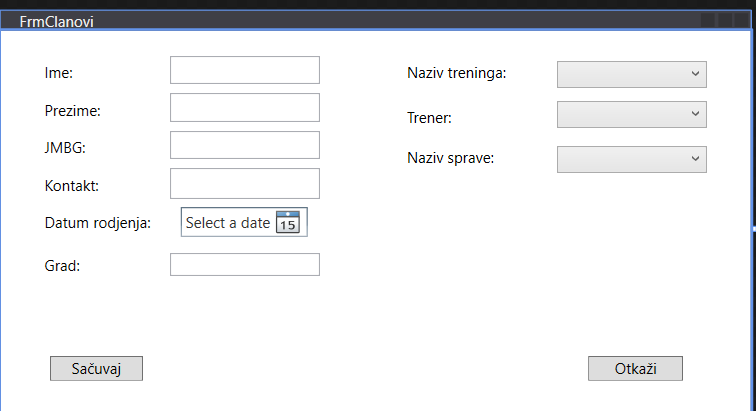
Slika 4.1 – Dodavanje komandi iz Toolbox-a

Forma za dodavanje novog člana u bazu podataka, treba da sadrži sledeće elemente (slika 4.2). 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 Poništi, 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 btnZatvori\_Click.



Slika 4.2 – XAML kod kreiranih komponenti.

Vizuelni prikaz forme za unos novog člana izgledaće kaon a slici 4.3



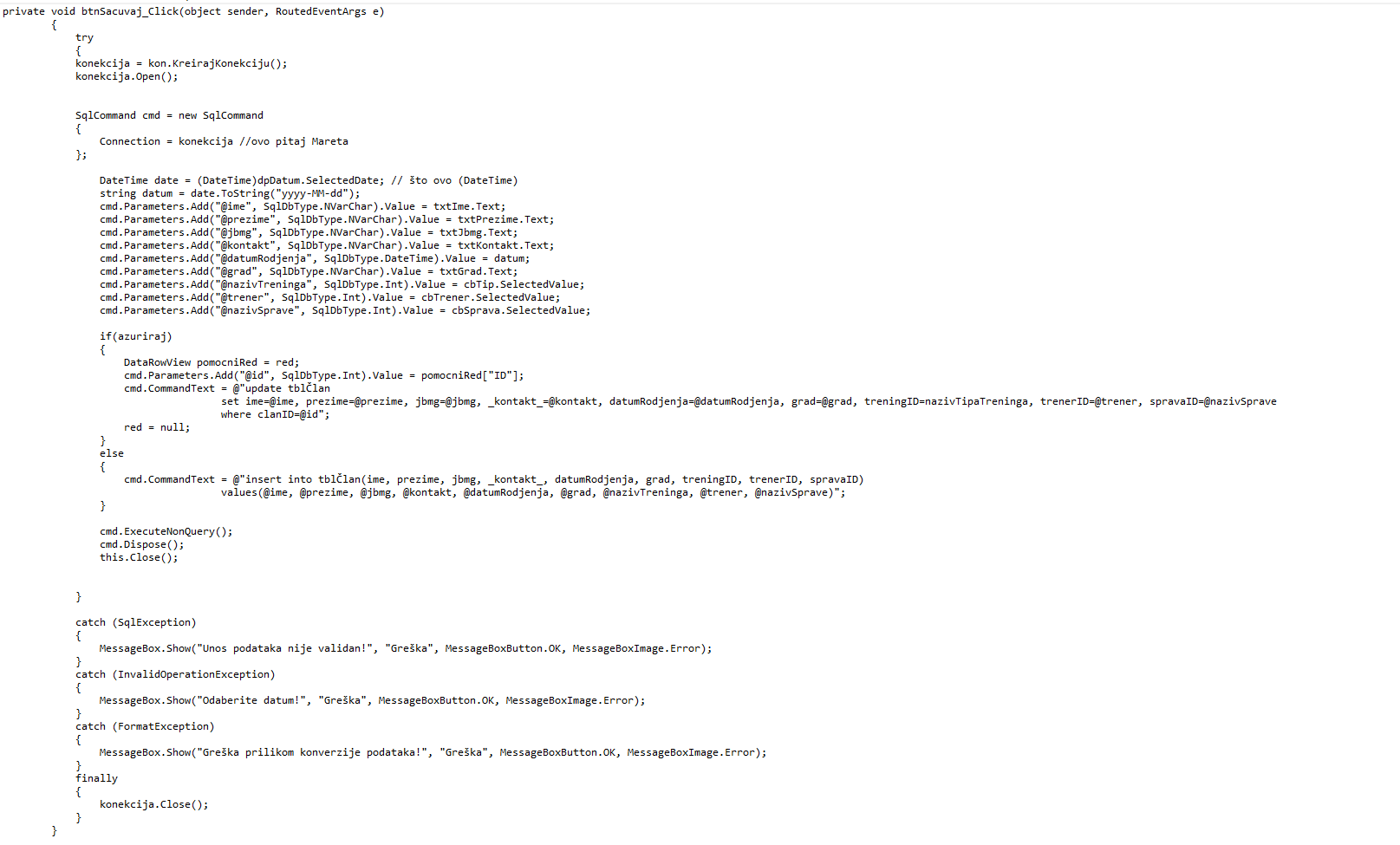
Slika 4.3 – Vizuelni prikaz forme za dodavanje/ izmenu člana

U Code Behind-u ove forme, trebalo bi da se nalazi sledeća poslovna logika (slika 4.4). Najpre je potrebno uključiti već poznati 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 ftmČlanovi, to je tekst za polje za unos imena člana.



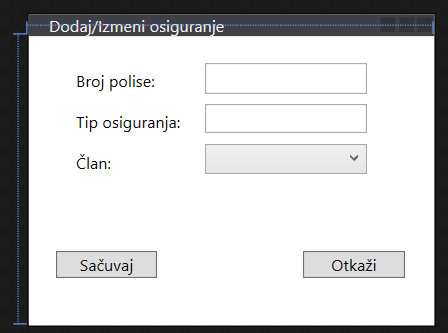
Slika 4.4 – Sadržaj CodeBehind-a forme FrmClanovi

Ovakav prikaz pozadinskog koda gotovo je identičan za svaku od sedam kreiranih formi. Ono u čemu je svaka od ovih formi jedinstvena jeste pozadinska funkcionalnost koja se dešava u trenutku kada korisnik klikne tasterom miša na dugme Sačuvaj. Na slici 4.5 prikazan je sadržaj izgenerisane metoda Btn\_Sacuvaj\_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 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 parameter prima prethodno kreirani string, ali I objekat klase Connection putem kojeg će se znati u koju bazu podataka treba da upiše prosleđeni zapis. Nakon toga, komanda se izvršava puten 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 instance konekcije ne zatvori.

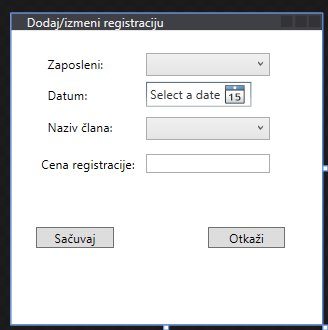


Slika 4.5 – Sadržaj metode BtnSacuvaj\_Click za dodavanje novog člana

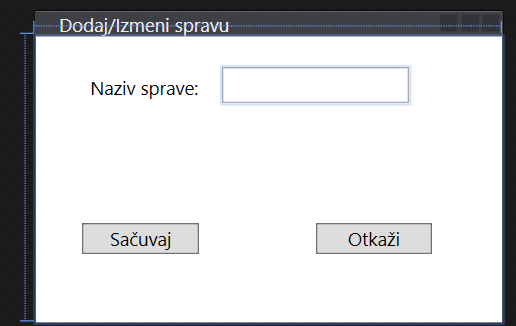
U nastavku će biti prikazan izgled I kod ostalih formi.



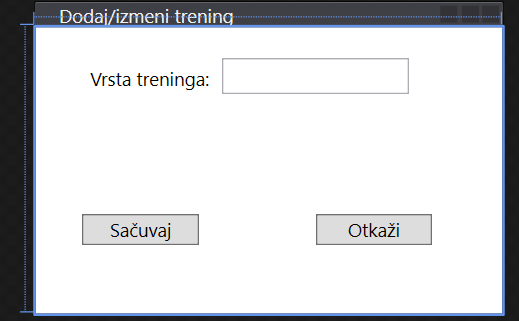
Slika 4.6 – Dizajn forme za dodavanje/Izmenu osiguranja



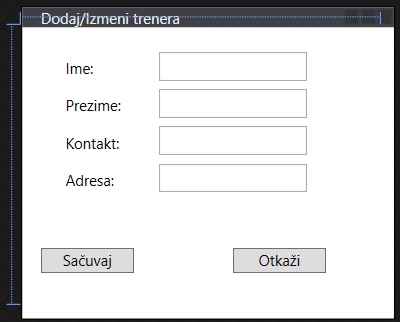
Slika 4.7 – Dizajn forme za dodavanje/Izmenu registracije



Slika 4.8 – Dizajn forme za dodavanje/izmenu sprave



Slika 4.9 – Dizajn forme za dodavanje/izmenu treninga



Slika 5.0 – Dizajn forme za dodavanje/izmenu trenera



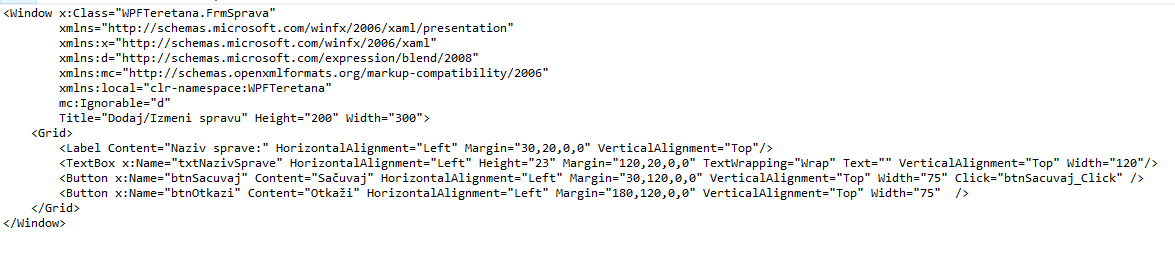
Slika 5.1 – XAML kod forme za dodavanje/izmenu člana



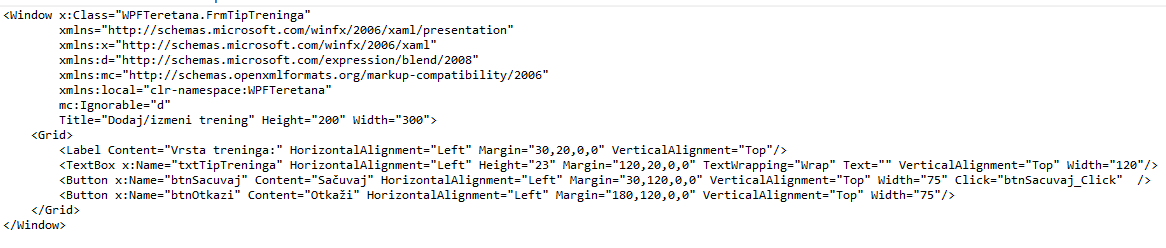
Slika 5.2 – XAML kod forme za dodavanje/izmenu osiguranja



Slika 5.3 – XAML kod forme za dodavanje/izmenu registracije



Slika 5.4 – XAML kod forme za dodavanje/izmenu sprave



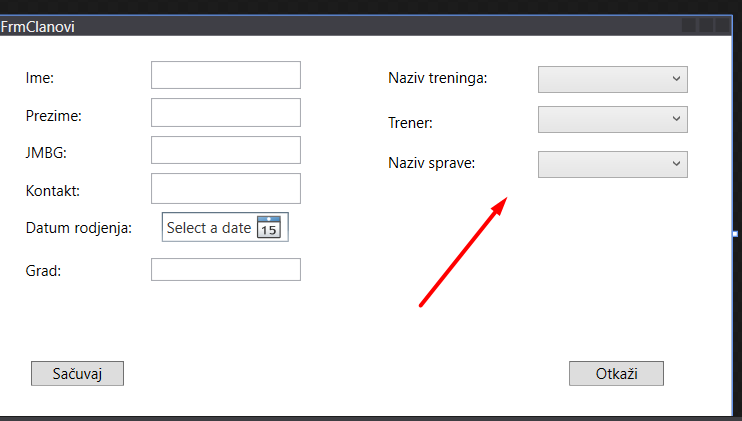
Slika 5.5 – XAML kod forme za dodavanje/izmenu tipaTreninga



Slika 5.6 – XAML kod forme za dodavanje/izmenu trenera

Objašnjenje za ComboBox-a:

Tabele tblRegistracija, tblČlan, tblOsiguranje, tblSprava\_Clan 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 to se na najjednostavniji način postiže upravo korišćenjem ComboBox-a. Vizuelni prikaz forme za dodavanje novog člana člana prikazan je na slici 5.7.



Slika 5.7 – Vizuelni prikaz forme za dodavanje/izmenu čkana

Prilikom kreiranja novog ComboBox-a potrebno je da se definiše DisplayMemberPath, koji predstavlja obeležje povezane tabele koje će se prikazati u padajućoj listi kada korisnik klikne na taj ComboBox, kao I SelectedValuePath koji predstavlja obeležje primarnog ključa na osnovu kog će se uneta vrednost iz prikazanog ComboBox-a povezati sa svojim identifikacionim obeležjem I kao takva proslediti u bazu podataka.

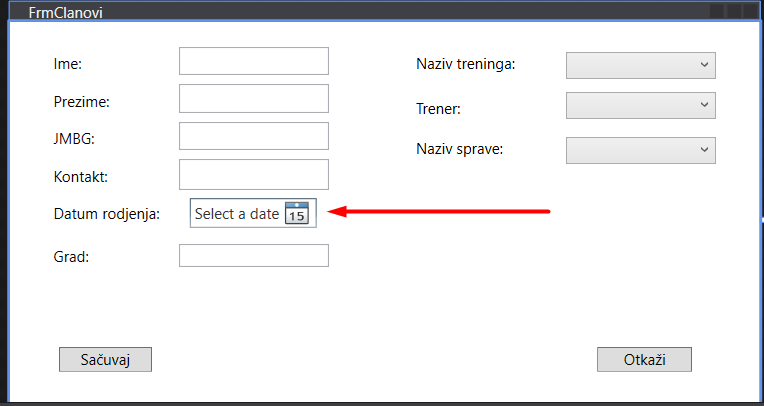
Način popunjavanja comboBox-a (primer kod tblČlan, slika 5.8):



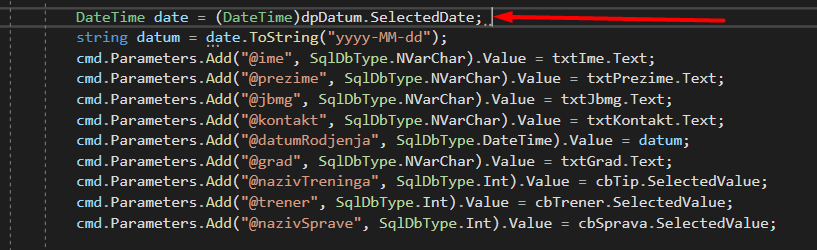
Slika 5.8 – princip filovanja ComboBox-a

Sličnim postupkom, isto se postiže I za ostale forme koje imaju ComboBox-ove.

Primer implementiranja DatePicker-a (slika 5.9):

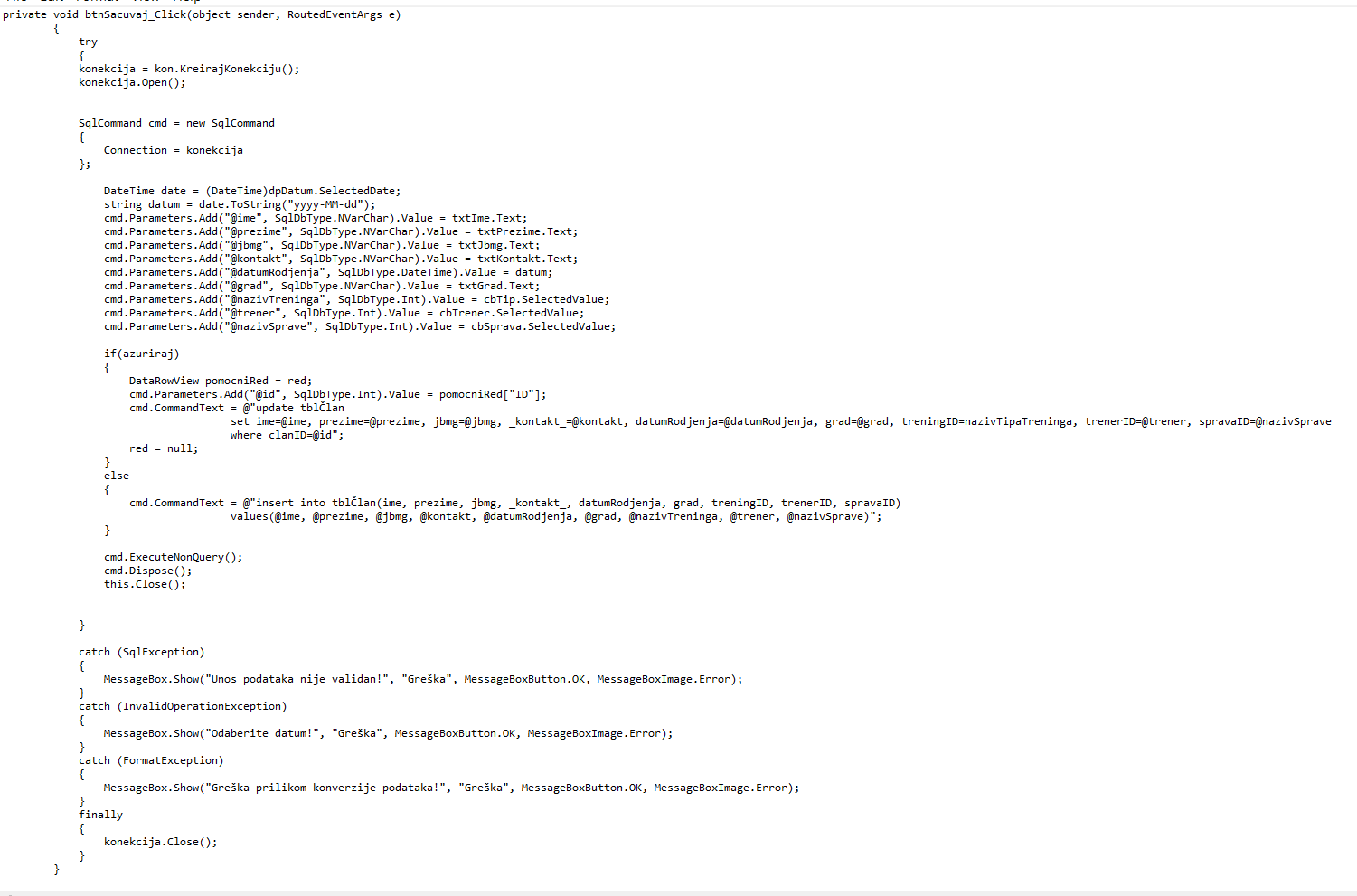


Slika 5.9 – Primer implementiranja DatePicker-a (prvi deo)



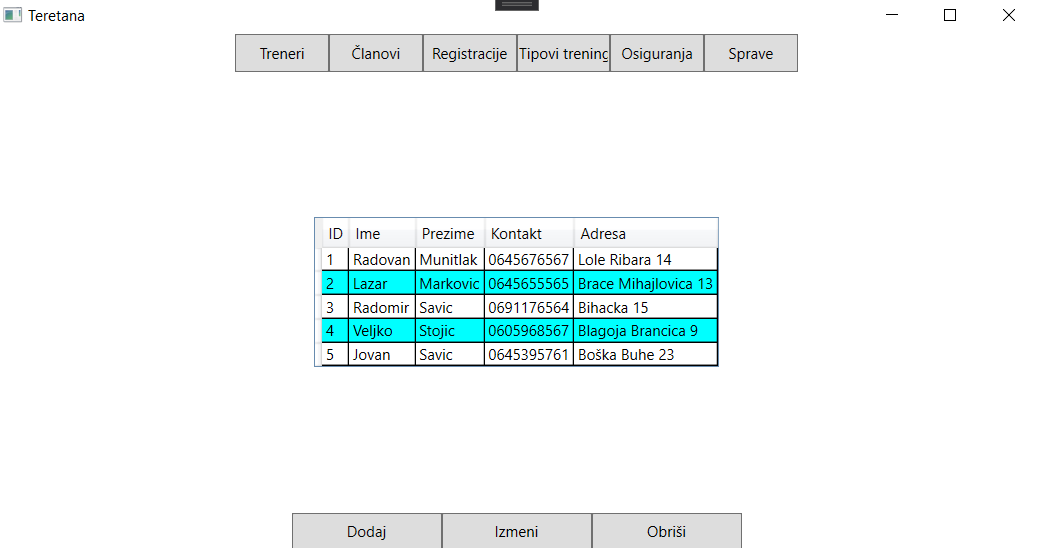
Slika 5.9 – Primer implementiranja DatePicker-a (drugi deo)

Sadržaj implementacije btn-a Sačuvaj kod tblČlan (slika 6.0):



Slika 6.0 – implementacija btn-a Sačuvaj kod tblČlanovi

Nakon kreiranja formi za svaku od tabela, potrebno je nešto slično napraviti I za glavni (Main) prozor. Main prozor predstavlja prvi, inicijalni, kontakt korisnika sa aplikacijom I ujedno I bazni 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 kreiranih tabela u bazi, čijim će se klikom tastera miša, na sredini forme iščitavati ti podaci. Takođe, klikom tastera miša na dugme određene tabele, potrebno je da se korisniku obezbede dugmići za dodavanje, izmenu I brisanje prikazanog sadržaja. Dodavanjem odgovarajućih funkcionalnosti tim dugmićima u okviru glavne forme, obezbediće se izvršavanje osnovnih CRUD operacija, što je I bio cilj ovog zadatka. Prikaz ove forme predstavljen je na slici 6.1.



Slika 6.1- Vizuelni prikaz glavnog prozora

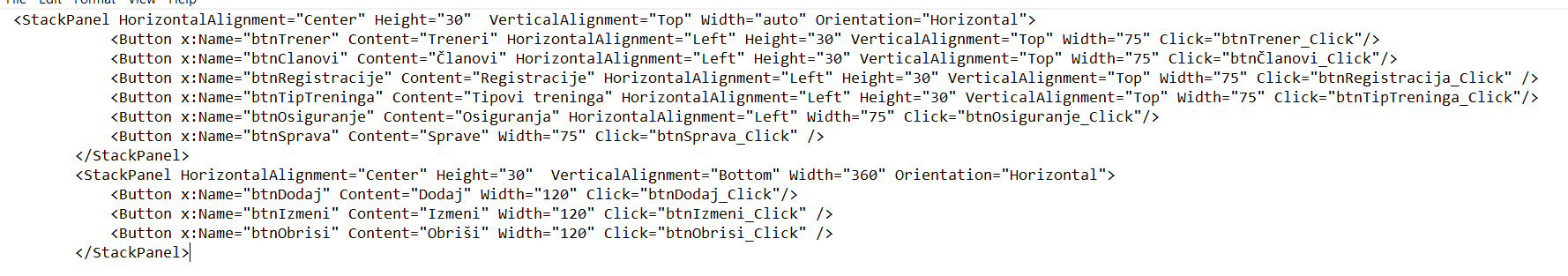
U nastavku treba sagledati XAML kod ove forme.

Na samom početku, potrebno je definisati centralni DataGrid u okviru kojeg će biti ispisani učitani podaci iz baze.



Slika 6.2 – XAML kod centralnog grida

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 na slici, a zatim izgenerisati metode koje će sprovoditi odgovarajuće akcije klikom tastera miša na to dugme, a čija će implementacija biti dodata nešto kasnije.

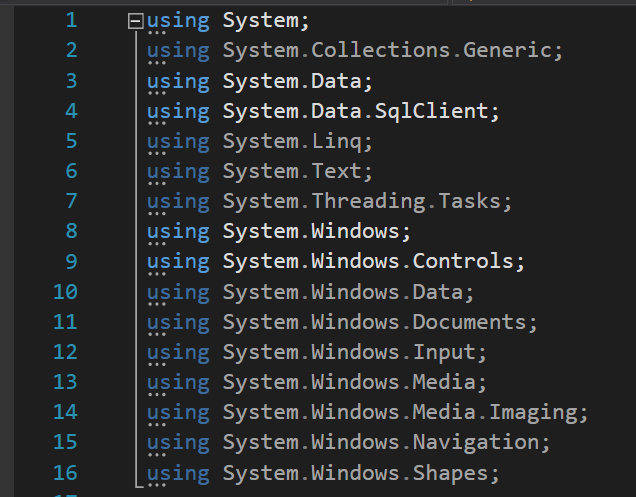


Slika 6.3 – StackPaneli glavnog prozora

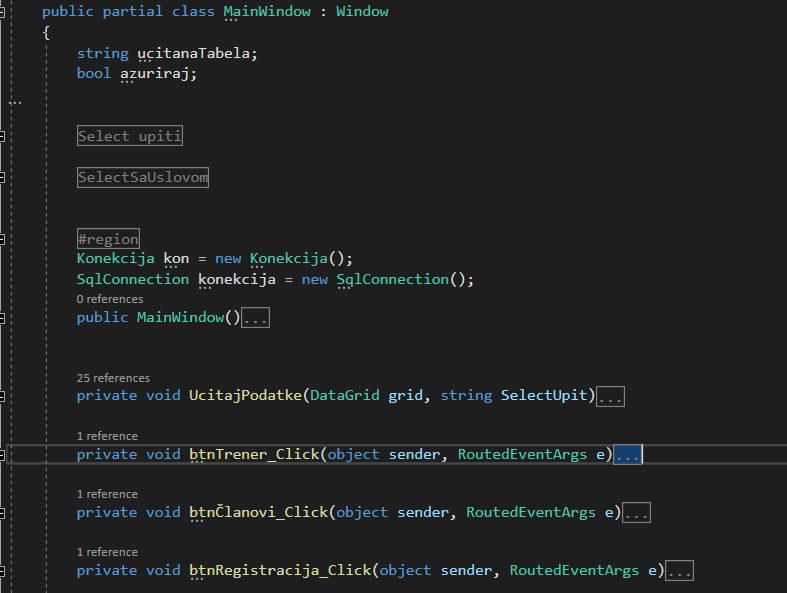
U okviru glavnog StackPanela, prikazan je I donji StackPanel, u okviru kog su kreirani dugmići za dodavanje, izmenu I brisanje podataka iz svake od posebnih tabela, I za svaki od dugmića kreirana je metoda koja će obavljati neku funkcionalnot klikom tastera miša na to dugme. Prilikom kreiranja ovog StackPanela, potrebno je obratiti pažnju na širinu I visinu Panel-a ali I svakog posebnog dugmeta. U okviru širine StackPanel-a, potrebno je da se nađu tačno 3 dugmeta za odgovarajuću formu.Takođe, visina StackPanel-a treba da bude jednaka visini dugmeta. Na ovaj način, dugmići će biti preklopljeni jedni preko drugih I kasnije će se prikazivati u zavisnosti od toga koje nam je dugme potrebno. Naravno, ovo je samo jedno od dizajnerskih rešenja za ovakav raspored dugmića I nije nužno pridržavati se istog prilikom kreiranja I rasporeda ovih komponenti u XAML kodu.

U nastavku će biti prikazan CodeBehind sadržina glavnog prozora, gde je smešten najveći deo poslovne logike.

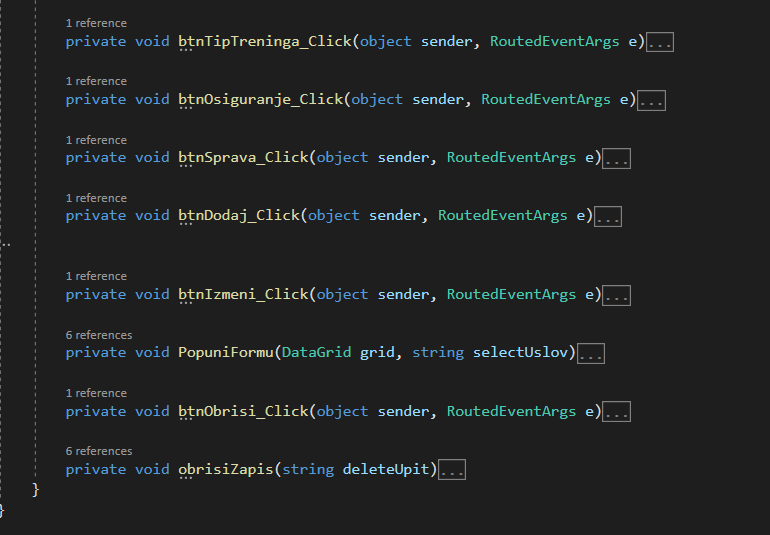
Na samom početku, potrebno je uključiti odgovarajuće imenske prostore.



Slika 6.4- Imenski prostori za glavni prozor

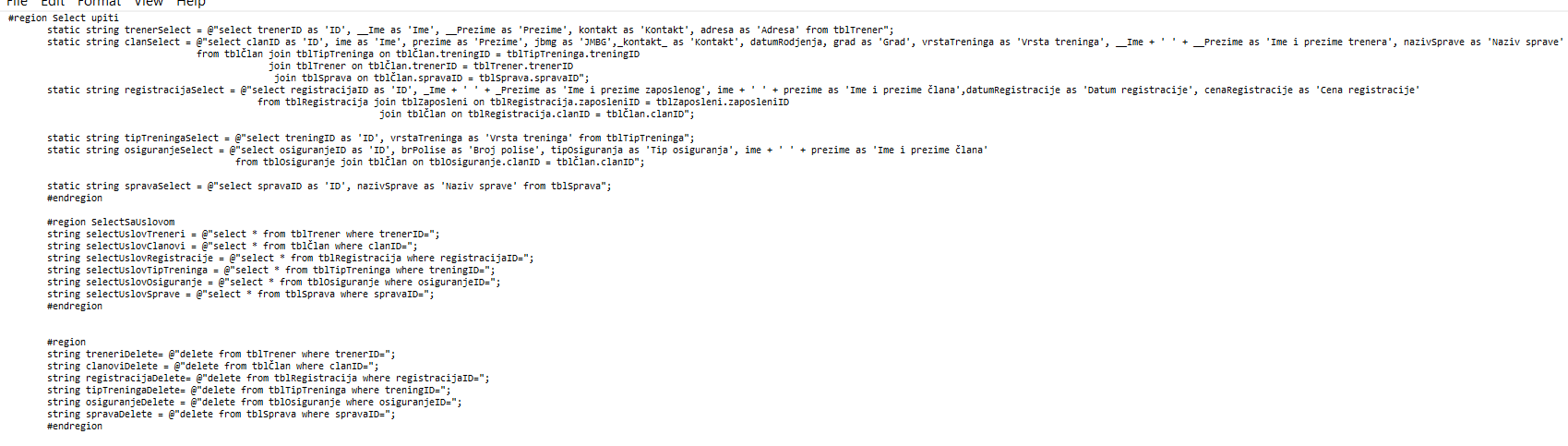


Slika 6.5- Sadržaj CodeBehind-a glavnog prozora (prvi deo)



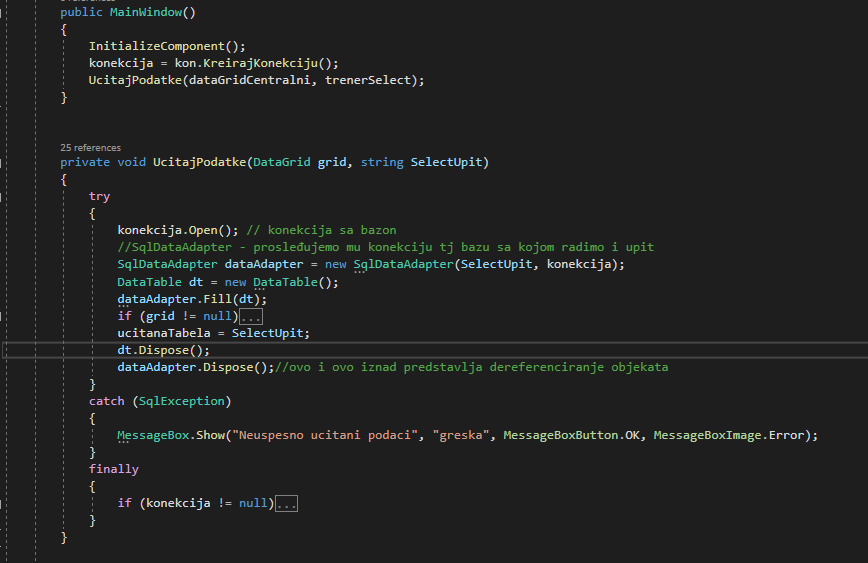
Slika 6.6 – Sadržaj CodeBehind-a glavnog prozora (drugi deo)

Upite koji su potrebni za uspešnu komunikaciju sa bazom podataka izdvojiti u posebne promenljive (slika 6.7).



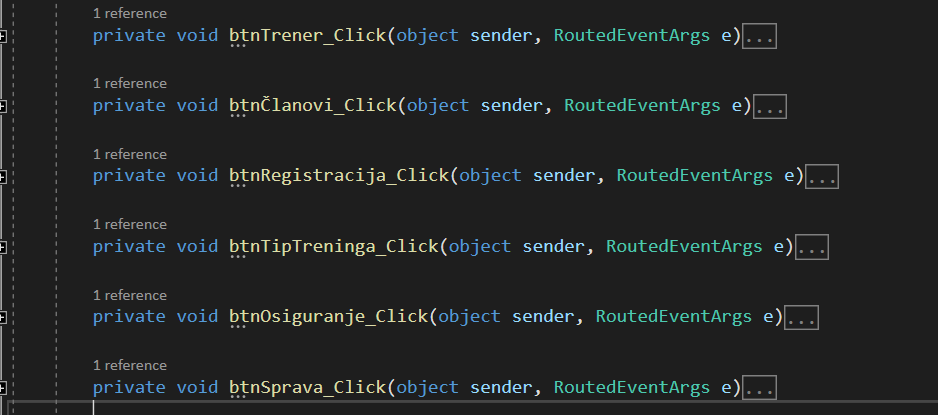
Slika 6.7 – Upiti za komunikaciju sa bazom podataka

Nakon instanciranja konekcije, sledeća stvar je kreiranje metode koja će popunjavati početni DataGrid pri kreiranju novog prozora. U ovom primeru, table tblTrener će biti podrazumevana tabela, odnosno tabela čiji će se prikaz generisati pri pokretanju ovog prozora. Zatim, na isti način kao kod popunjavanja ComboBox-ova 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 tblTrener.



Slika 6.8 – Sadržaj metode UcitajPodatke I konstruktor

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. Na osnovu prethodno kreirane metode UcitajPodatke I select upita, za sve druge metode, koje predstavljaju odgovor na klik korisnika, poziva se upravo ova metoda I prosleđuje joj se odgovarajući upit.



Slika 6.9- Sadržaji metoda za učitavanje podataka iz baze

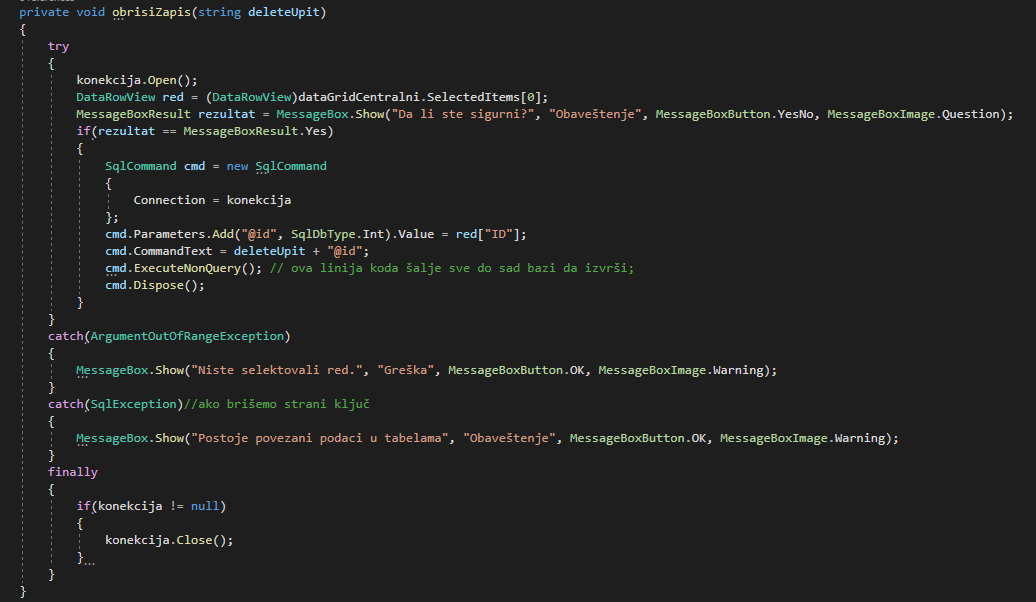
Sadržaj metoda generisanih na klik tasterom miša na dugme Dodaj je poprilično jednostavan. Klikom tastera miša na dugme Dodaj, instancira se prozor odgovarajuće forme I prikazuje korisniku na ekranu. Nakon zatvaranje tog prozora forme, osvežava se centralni grid sa aktuelnim podacima, nakon eventualnog dodavanja novog zapisa u bazu.



Slika 7.0 – Sadržaj metode btnDodaj\_Click

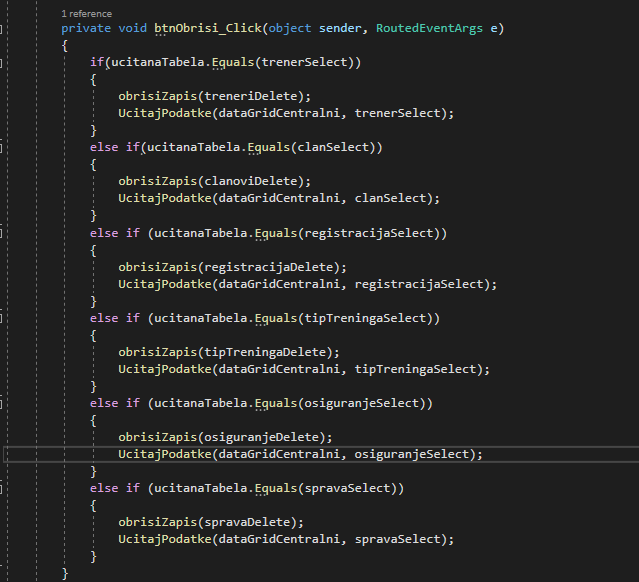
Brisanje podataka iz baze podataka obavlja se putem klike tasterom miša na dugme Obriši. Nakon otvaranja konekcija, kreira se objekat klase DataRowView koji u sebi čuva informaciju o indeksu reda koji je korisnik prethodno označio. Nakon toga kreira se Delete SQL komanda putem koje će se u samoj bazi obrisati željena Stavka. Pre samog izvršavanja komande, pred korisnikom se pojavljuje dijalog prozor (Message Box) putem kojeg on potvrđuje ili odustaje od brisanja. Dva osnovna izuzetka koja se ovde mogu javiti su:

-ArgumentOutOfRangeException I SqlException. Prvi se javlja pri pokušaju brisanja ukoliko nije označen nijedan red, a pritisnuto je dugme za brisanje, a drugi ukoliko se pokuša obrisati entitet iz tabele koji je primarnim ključem povezan sa nekom drugom tabelom.



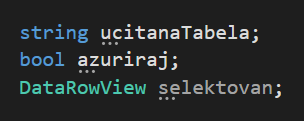
Slika 7.1- sadržaj metode obrisiZapis

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 ObrisiZapis prosleđuju se parametri I nakon toga ponovo učitavaju podaci iz baze podataka, kako bi oni bili ažurni nakon brisanja.



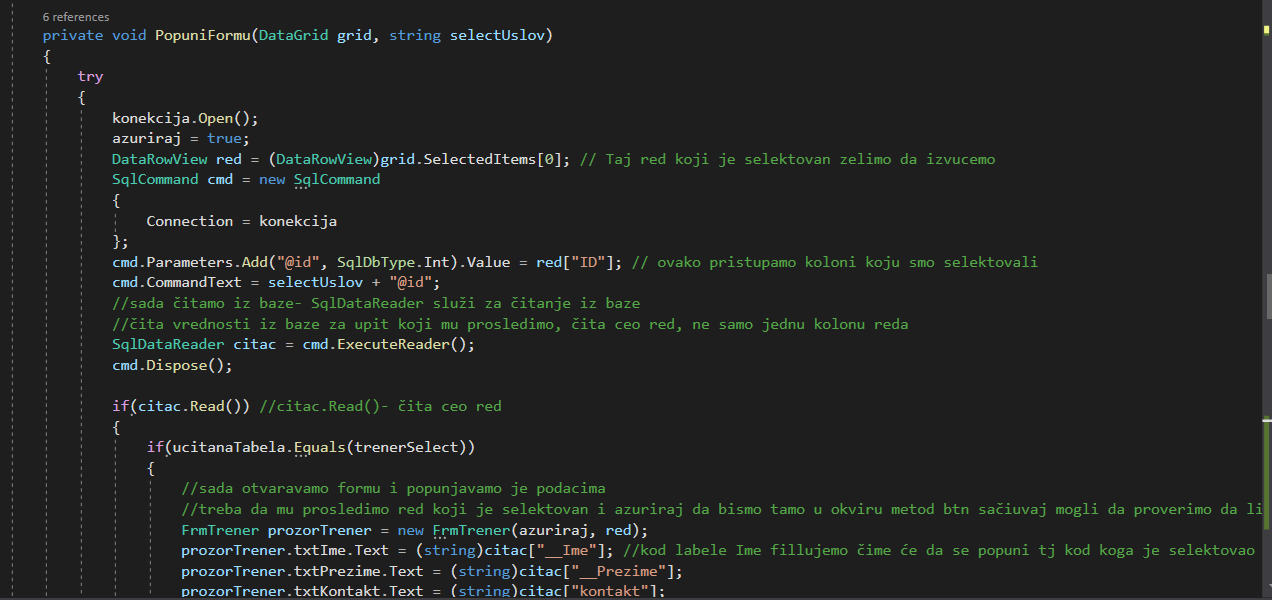
Slika 7.2 – sadržaj metode btnObrisi\_Click

Za modifikaciju podataka, kao I za dodavanje, potrebno je da postoji odgovarajuća forma u koju 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 koristiti 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 je tipa podatka string, koja govori koja je tabela trenutno učitana.

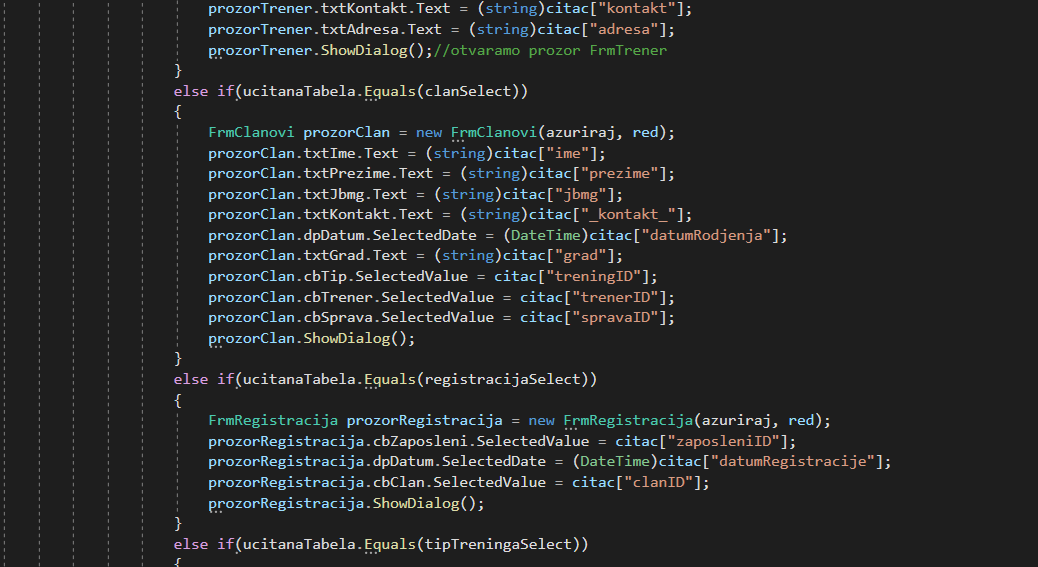


Slika 7.3 – Deklaracija promenljivih

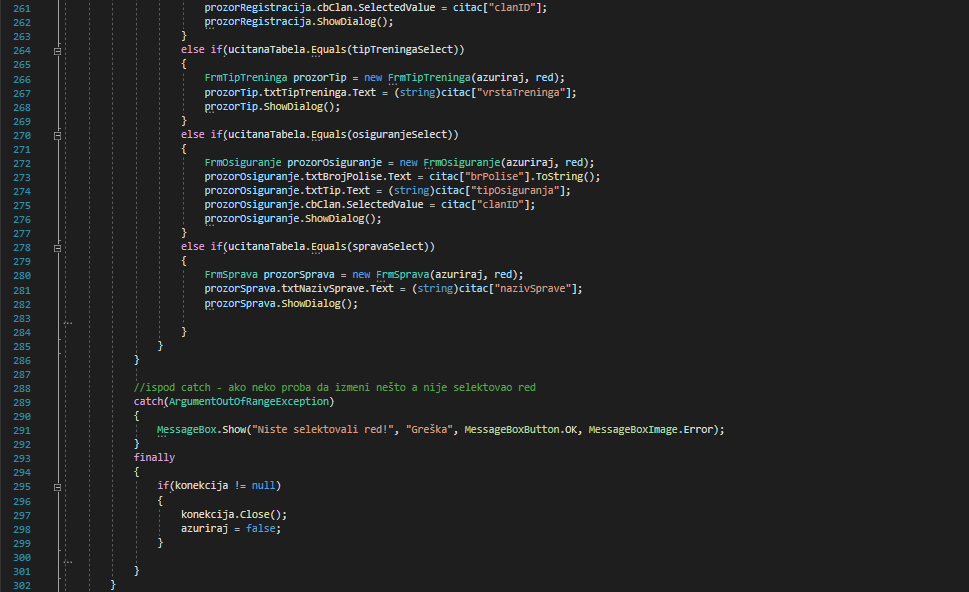
Na samom početku, deklarisana promenljiva postavlja se na true. Nakon toga instancira se prozor njene forme I otvara konekcija. Pomoću DataRowView objekta, kao I kod brisanja, pamti se koji je red označen. Razlike je u tome što se sada, u drugu promenljivu dodaje ta vrednost kako bi se iskoristila u 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 pomiću SqlDataReader-a. Nakon toga, poziva se metoda prozora ShowDialog I on se prikazuje na ekranu sa popunjenim vrednostima traženog zapisa. Dalja funkcionalnost obavlja se unutar te forme I dugmeta Sačuvaj I biće objašnjena kasnije. 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.



Slika 7.4- Sadržaj metode PopuniFormu – prvi deo



Slika 7.4- Sadržaj metode PopuniFormu – drugi deo



Slika 7.4- Sadržaj metode PopuniFormu – treći deo

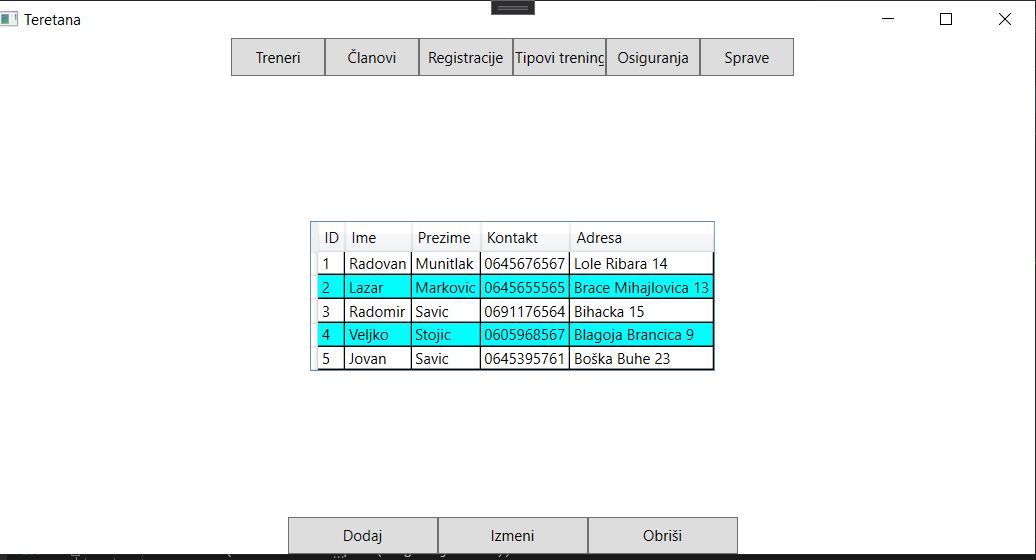
Kada korisnik izabere željenu stavku koju želi da izmeni, poziva se metoda btnIzmeni\_Click, koja u zavisnosti od učitane tabele prosleđuje odgovarajući parameter metodi PopuniFormu I nakon izvršene izmene ponovo učitava podatke kako bi podaci bili relevantni.



Slika 7.5 – Sadržaj metode btnIzmeni\_Click

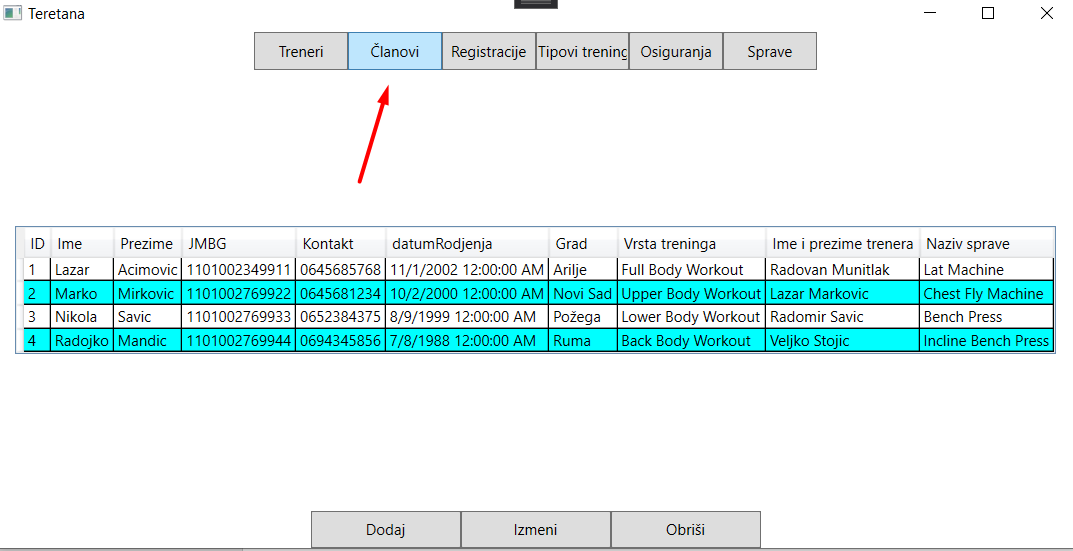
**7.Testiranje aplikacije**

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. Pri pokretanju aplikacije, generiše se sledeći prozor (slika 7.6):



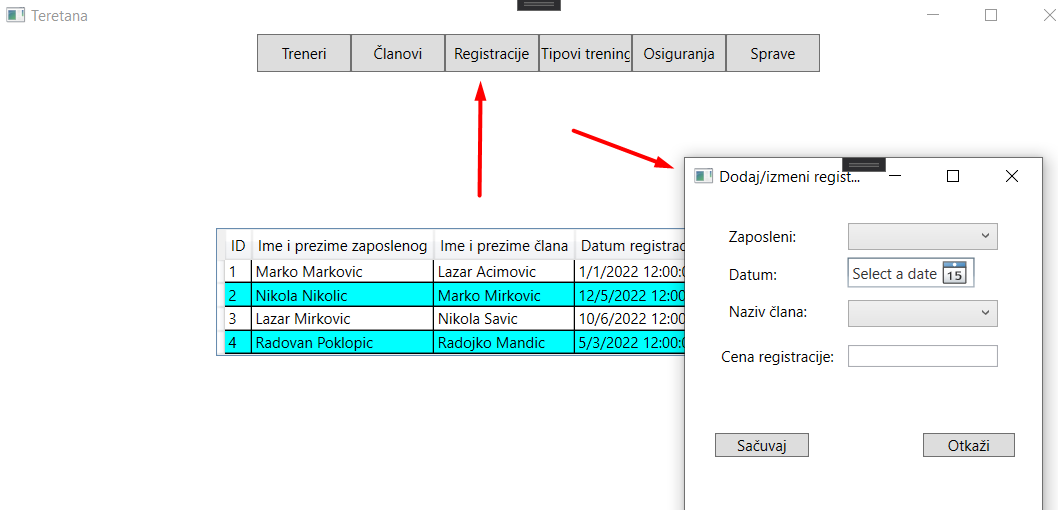
Slika 7.6 – Početni prozor pri pokretanju aplikacije

Klikom tastera miša na dugmiće iz gornjeg StackPanel-a, vrši se prikaz podataka željene tabele na centralnom gridu (slika 7.7).



Slika 7.7 – Izlistavanje podataka o članovima na početnom gridu

Klikom tastera miša na dugme Dodaj, otvara se forma za dodavanje novog zapisa u bazu podataka (slika 7.8):



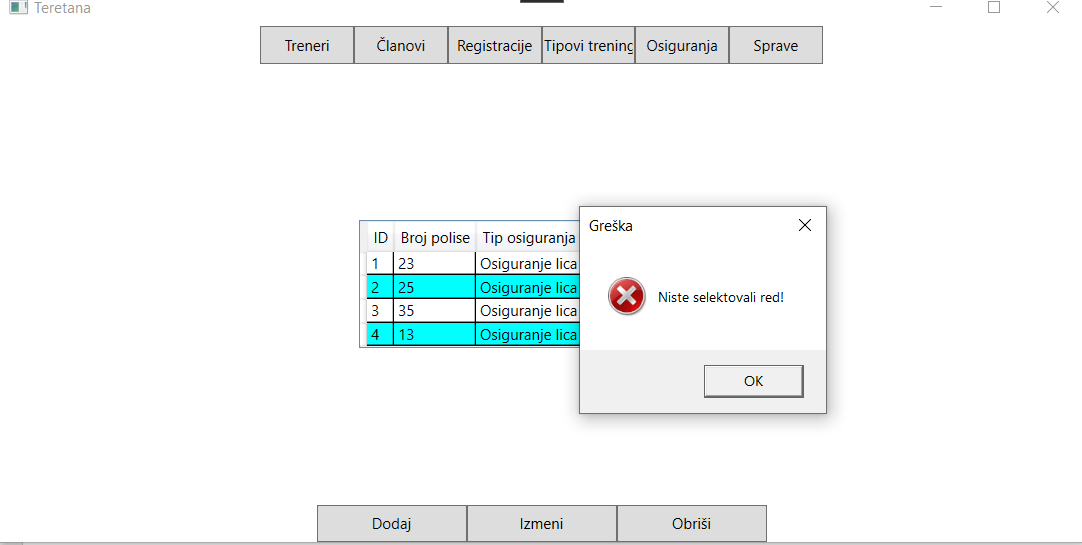
Slika 7.8 – Dodavanje nove registracije u bazu podataka

Klikom tastera miša na dugme Sačuvaj, novi zapis dodaje se u bazu podataka I prikazuje na centralnom gridu (slika 7.9).



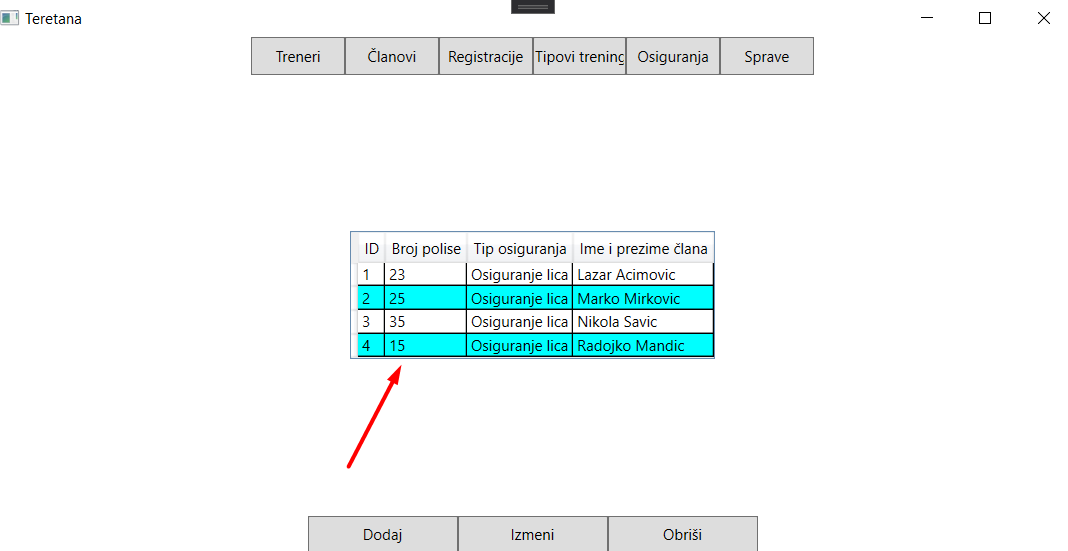
Slika 7.9 – Ispis dodatog zaposlenog

Klik na dugme Izmeni ili Obriši, a da pritom nije označen nijedan red, uzrokuje sledeću grešku/obaveštenje (slika 8.0):



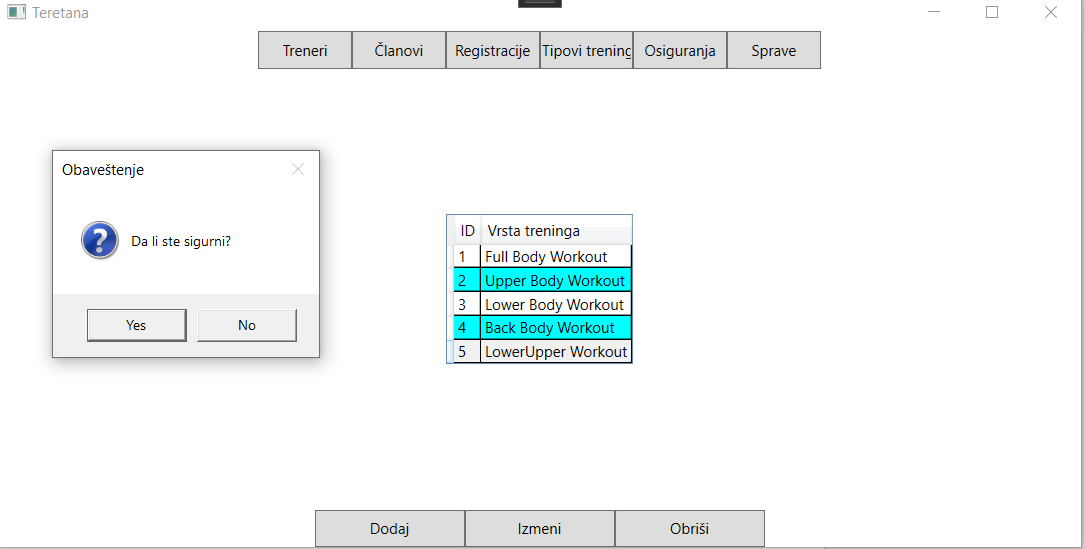
Slika 8.0 – Pokušaj klika na dugme izmeni ili obriši brz označavanja reda

Nakon izmene željenih podataka, ažurirani zapis se prikazuje na ekranu (slika 8.1):



Slika 8.1- Ažuriran sadržaj nakon promene podataka

Brisanje podataka vrši se klikom tastera miša na dugme Obriši, nakon što je korisnik prethodno označio željeni red za brisanje. Klikom tastera miša na dugme Da, korisnik potvrđuje brisanje, a na ekranu se generiše ažuriran ispis. (slika 8.2).



Slika 8.2 – brisanje podataka