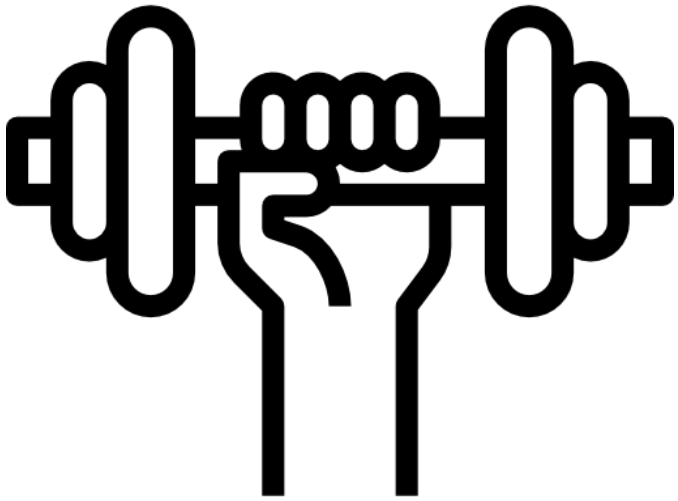


„WPF GYM“

Tehnička dokumentacija projekta iz predmeta „Poslovni informacioni sistemi“

Mladen Bajić(IT46/2018)

SADRŽAJ :

1.OPIS REALNOG SISTEMA: 2
2.ZADATAK 4
3.OPIS TEHNOLOGIJA 5
4.UML 10
4.1.Dijagram slučajeva upotrebe..... 10
4.2.Cass dijagram 14
4.3.Dijagram sekvenci: 18
5.BAZA PODATAKA 20
5.1.Kreiranje baze podataka: 20
5.2.Upiti i DML naredbe:..... 30
6.KREIRANJE WPF APLIKACIJE..... 36
6.1.Dodavanje 41
6.2.Izmjeni(Ažuriraj) 44
6.3.Obriši 47
6.4 Izgled aplikacije:..... 49
7.TESTIRANJE APLIKACIJE 63
8.ZAKLJUČAK 65
8.1.Lični osjećaj u radu:..... 66

1.OPIS REALNOG SISTEMA:

Teretana kao realni sistem se bazira na konkretnim usugama uz dodatne opcije preprodaje proizvoda. Djelatnost teretane se odnosi na pružanje konkretne usluge zarad ostvarivanja koristi. Konkretne akcije se odnose većinom na pružanje usluge i malom dijelom su uključene funkcije nabavke, trasporta i prodaje.

Realni sistem(teretana) se fokusira na zadovoljstvo potrošača i konkretan sistem je okrenut potrošačima(članovima teretane, kupcima programa, kupcima proizvoda). Realan sistem se trudi da pruži što bolje uslove kako bi pobjedio konkurenciju i održao konkurentnost na tržištu,a u tome mu svakako pomažu softverske alatke poput „WPFGYM-a“.

„WPF GYM“ je softverska alatka za upravljenje i menadžment realnih okruženja kao što su teretane, fitnes centri i ostale ustanove čija je djelatnost u okviru treninga i fitnesa.

Softverski alat koji je namjenjen za upravljanje sistema ove vrste mora pokrivati više kako povezanih tako i zasebnih funkcija najviše u oblasti interakcije korisnika i zaposlenih.

Uzmimo klasičan fitnes centar kao referent za primjer za primjenu konkretne „WPF“ aplikacije. Konkretne akcije koje se svakodnevno dešavaju u teretani su: Registracija novih korisnika, obnova članstva postojećih korisnika, prodaja opreme, kreiranje programa za konkretne korisnike, a tu su naravno i rjeđe izvršavane ali ne manje važne funkcije

dodavanja novih zaposlenih kao i vremensko ažuriranje i kreiranje novih ili postojećih vrsta članstva. Sve ove akcije treba da pokrije aplikacija i uz fluidnost da pruži sigurnost i olakša rad samom zaposlenom.

Registracija novog korisnika treba da prođe nekoliko faza, a to su: dodavanje konkretnih informacija o novom korisniku. U ovom koraku mora biti jasno definisano koje informacije su neophodne a koje korisnicu mogu sačuvati za sebe. Nakon dodavanja konkretnog korisnika potrebno je dodati njegovo članstvo, konkretno vrijeme trajanja članstva i koje konkretno članstvo koristi(specijalno,studentsko...).Dodatne informacije koje su neophodne su kada se ta registracija desila(datum registracije ne mora biti isti kao i datum početka članstva) i koliko je ista plaćena.

Evidencija dolazaka je vema važan dio svakodnevnog rada teretane, jer omogućava zaposlenima da prate aktivnost članova kao i da provjeravaju da su dati članovi još uvijek u vremenskom okviru svojih članarina. Obzirom da svi članovi pri dolašku dobijaju ključ važno je evidentirati broj kluča koji dobije korisnik u slučaju gubitka ključa ili krađe stvari.

Kreiranje programa za članove teretane je usluga teretane kojom se korisnicima nudi da po povoljnim cijenama kupe program treninga, ishrane ili drugog od strane zaposlenih trenera u teretani. Ova evidencija mora biti ažurna da ne bi došlo do kašnjenja u kreiranju programa ili treninzima te da se ne bi iznevjerili korisnici.

Prodaja opreme predstavlja svakodnevnu akciju u radu teretane, teretana posjeduje određeni asortiman kako nutritivnih tako i proizvoda za trening(majice,šorcevi,straps...) koje prodaje korisnicima i od toga privređuje.U ovom slučaju bitna je evidencija ko je prodao koliko proizvoda. Ove informacije su bitne radi opravdavanja odnosa sredstava i novca u blagajni teretane.

Vrste članstva su jedan od najdinamičnijih segmenata u teretani, one zavise od politike teretane, dijela godine i mnogo drugih stvari. Teretana može imati samo tradicionalne vrste članstava a može biti inovativna te u kratkom periodu kreirati veliki broj različitih akcija i specijalnih članstava.

Potrebno je voditi evidenciju o zaposlenima, obzirom da je teretana dinamično radno mjesto i za koje je potrebno imati iskustvo u vježbanu često se dešavaju promjene u okviru kadra. Zaposleni kao osobe koje upravljaju većinom akcija moraju dobro znati raspored programa i moraju biti dobro upoznati sa sitemom rada u teretani.

2.ZADATAK

Konkretan zadatak se odnosi na kreiranje softverskog alata koji će u potpunosti pokriti sve radnje neophodne za unos,izmjenu i brisanje podataka važnih za administraciju samog realnog sitema konkretno teretane.

Zadatak softverskog alata „WPF GYM “ jeste da omogući potpuno upravljanje i menadžemnt sistemom podataka u teretani. „WPF GYM“ treba u potpunosti da olakša rad zaposlenima i da uz veliku pouzdanost održava sistem podataka živim. Od ovog softverkog alata se očekuje da svojom pouzdanošću, fluidnošću kao i intuitivnošću obezbjedi lagano korištenje u svakodnevnim uslovima rada.

Konkretni zadaci ovog sistema se odnose na radnje vezane za upravljanje podacima, tačnije za ragistraciju novih i starih članova, evidenciju dolazaka, programa kao i evidenciju prodaje proizvoda. „WPF GYM“ treba da obezbjedi mogućnost dodavanja,izmjene i brisanje svih gore navedenih podataka vezanih za sve konkretne akcije koje se sprovode u ralnom sistemu. Neophodno je da sistemi ove vrste pored mogućnosti dodavanja, izmjene i brisanja posjeduju i opcije poput pretrage radi lakšeg upravljanja podacima ili log-in opcije radi sigurnosti u korištenju.

3.OPIS TEHNOLOGIJA

Kroz konkretan projekat je implementirano više tehnologija u oblasti modelovanja, baza podataka kao i tehnologije za sam razvoj aplikacije.

Objedinjeni jezik za modelovanje odnosno ili UML(Unified Modeling Language) koji je standardni jezik za prikazivanje objektnog modela u polju softverskog inženjerstva. UML je

opšti jezik za modelovanje pomoću kojeg se preko grafičkih simbola pravi apstraktni model sistema poznat kao UML model. Kada je riječ o modleovanju za konkretni projekat, korištena su tri modela: Dijagram slučajeva upotrebe(Use-Case diagram), diagram klasa(Class diagram) i diagram sekvenci(Sequence diagram).

Dijagram slučaja upotrebe(Use Case diagram) služi za specificiranje poslovne funkcije(procesa) sistema kao elementa funkcionalne dekompozicije i predstvalja opis (deklaraciju) niza akcija koje sitem izvršava prilikom realizacije poslovne funkcije koja se pokreće na zahtjev korisnika da bi proizvela rezultat bitan za korisnika u funkciji ostvarenja postavljenih ciljeva poslovanja. Dijagram slučaja upotrebe predstavlja opis funkcionalnosti sistema u kojem se izostavljaju detalji vezani za način realizacije. Modelom slučaja upotrebe definiše se granica izmedju učesnika koji komuniciraju sa sistemom, funkcija koje ti učesnici mogu inicirati i u čijem izvršavanju učestvuju i poruka i podataka koje učesnici razmjenjuju sa sistemom. Izvršavanje jednog slučaja upotrebe započinje proleđivanjem poruke od nekog učesnika do sistema i traje sve dok proslijeđena poruka ne izazove promjene u sistemu , a po završetku sistem se nalazi u stanju u kojem učesnik može da inicira izvršavanje istog ili nekog drugog slučaja upotrebe. Važni pojmovi u konkretnom digramu su: Učesnik, apstraktni učesnik, agregacija(veza), generalizacija(veza), extend i include.

Dijagram klasa(Class diagram) je tip strukturnog dijagrama u UML-u (Unified Modeling Language). Opisuje strukture sistema tako što prikazuje klase tog sistema, njihove attribute, metode, i odnose među klasama. Predstavlja statičke elemente sistema, a ne procese ili stanje sistema u nekom određenom trenutku. Namjena dijagrama klasa je vizualno iskazivanje mogućeg, konkretnog stanja sitema u cjelini , ili njegovog dijela strukture podataka koja reprezentuje pojavu nad bazom podataka ili podšemom, ili njegovog dijela bilo kakave strukture podataka privremenog karaktera koja se nalazi u zoni programa , u operativnoj memoriji. Kod dijagrama sklasa jedan od najbitnijih pojmova jesu veze tj. Asocijacija,agregacija, kompozicija, generalizacija.

Dijagram sekvenci(Sequence diagram) jednostavno prikazuje interakciju između objekata redoslijedom, tj. Redoslijedom kojim se te interakcije odvijaju. Za dijagram sekvenci možemo koristiti i dijagrame događaja ili scenarije događaja. Dijagrami sekvence opisuju kako i kojim redoslijedom funkcionišu objekti u sistemu. Ovi dijagrami široko se koriste od strane programera za dokumentiranje i razumijevanje zahtjeva za nove i postojeće sisteme. Dijagram sekvenci se koristi za modelovanje i vizuelizaciju logike iza sofisticirane funkcije, opeacije ili postupka. Također se koriste za prikazivanje detalja dijagrama slučajeva UML.Koristi se za razumijevanje detaljne funkcionalnosti sadašnjih ili budućih sistema.

Nakon upotrebe UML dijagrama na red dolazi upotreba baze podataka. Baza podataka je zbirka podataka koja se organizira za brzo pretraživanje i pristup, a koja zajedno sa sistemom za administraciju, organizaciju i pohranu tih podataka čine sistem baza podataka. Iz korisničke perspektive, podaci su na neki način logički povezani. Korisnici pristupaju bazi podataka prvenstveno putem upitnika. Korištenjem ključnih riječi i naredbi sortiranja, korisnici mogu brzo pronaći, preurediti, grupirati i odabrati područje u mnogim zapisima koje će se vratiti, ili po kojem se trebaju sastaviti izvještaji za određenu grupu podataka prema pravilima dotičnog sustava upravljanja bazama podataka. U konkretnom projktu za upravljanje bazama podataka korišten je SQL, konkretno „SQL Managament Studio“.

SQL(Structured Query language) je standardni relacioni upitni jezik. Odnosi se stvaraju s jednom naredbom i odmah su dostupni, što ga čini jednostavnim za korištenje. Jedinstven je jer su svi podaci i rezultati operacija predstavljeni u tabelarnom obliku i omogućuju interaktivno i klasično programiranje. Konkretni projekat zasnovaj je za samoj bazi podataka i sama aplikacija je ustvari aplikacija koja rukuje podacima iz baze i njima upravlja. SQL Server Management Studio (SSMS) je softverska aplikacija koja je prvi put pokrenuta s Microsoft SQL Server 2005 i koja se koristi za konfiguriranje, upravljanje i administraciju svih komponenti u Microsoft SQL Serveru.

Nasljednik je Enterprise Managera u SQL 2000 ili ranije. Alat uključuje uređivače skripti i grafičke alate koji rade sa objektima i značajkama poslužitelja. MySQL višenitni je višekorisnički SQL sistem za upravljanje bazama podataka. Sistem radi kao server, obezbjeđujući višekorisnički interfejs za pristup bazi podataka. Šemacki kreiranu bazu podataka u class dijagramu sada prenosimo u MySQL Managament studio i konkretno se bavimo bazom, tačnije bavimo se upitima iz baze podataka. Osnovni upiti korišteni u našoj bazi su : „insert“, „select“ i „delete“ sa njihovim varijacijama.

Nakon kreiranja tabela u bazi podataka i njihovog popunjavanja te upoznavanja sa upitima same baze podataka prelazi se na kreiranje samo aplikacije. Kreiranje same aplikacije je radi pomoću C# razvojnog okruženja, konkretno, aplikacija se kreira pomoću WPF-a u okviru Microsoft C#-a. Windows Presentation Foundation (WPF) je UI okvir koji stvara aplikacije za desktop klijente. Razvojna platforma WPF podržava širok skup alata razvoj aplikacija, uključujući aplikacijski model, resurse, kontrole, grafiku, izgled, povezivanje podataka, dokumente i sigurnost. WPFje dio .NET-a. WPF koristi Extensible Markup Language (XAML) za pružanje deklarativnog modela za programiranje aplikacija.

Microsoft .NET Framework je softverska platforma koja se može instalirati na radne uređaje koji posjeduju Majkrosoft Windows operativni sistem. On sadrži veliki broj biblioteka

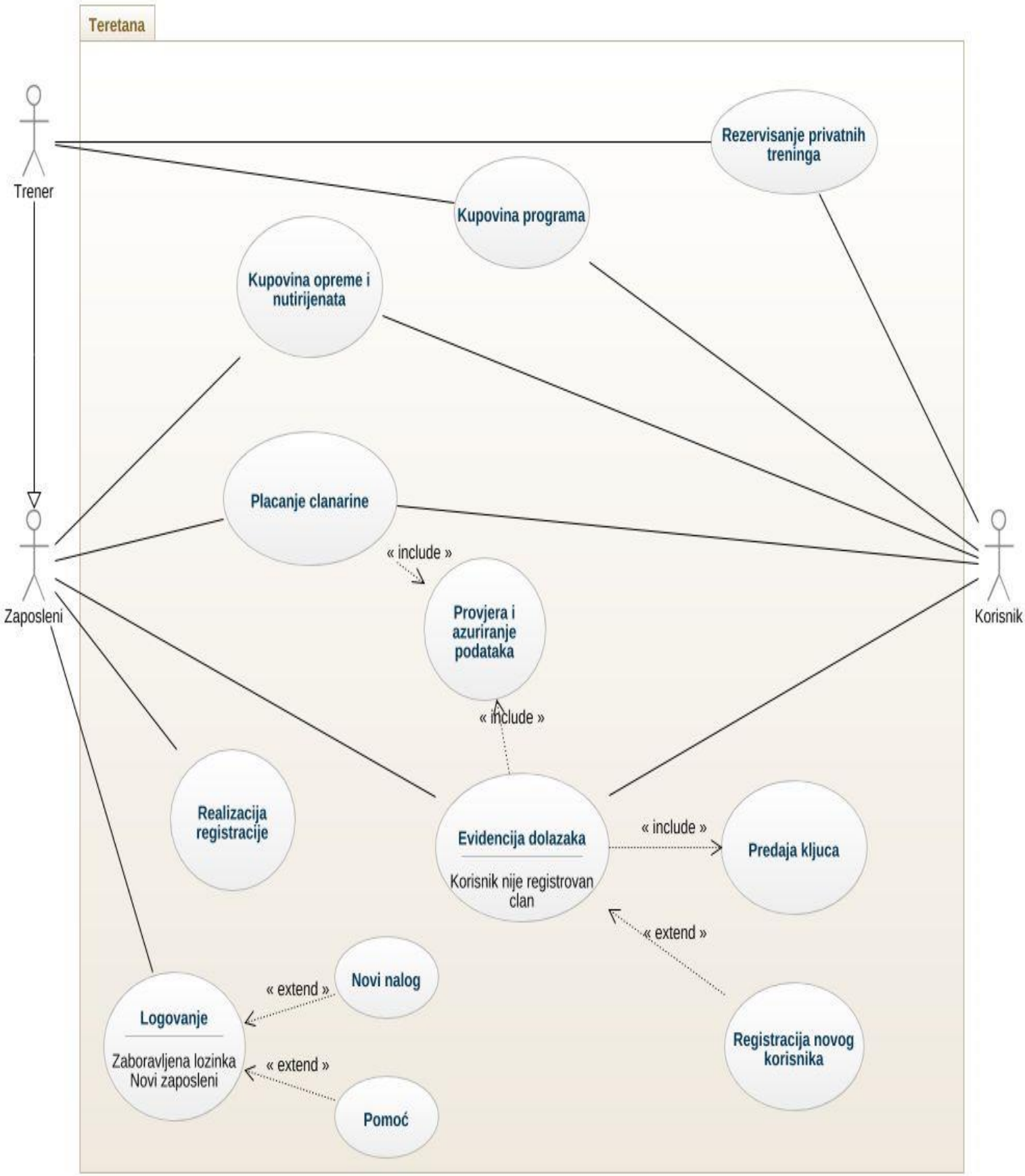
koda za riješavanje problema u programiranju i virtuelnoj mašini za upravljanje upravljačkim programima koji su izglasani sa programima pisanih posebno za .NET Framework. .NET podržava vaše softverske jezike, na način da komunicira interoperabilnost, pri čemu svaki jezik može biti napisan na drugom. Dostupan je na svim programskim jezicima jezika .NET Framework obuvala (VB.NET, C # ...). Kako biste mogli da primijenite aplikaciju, dovoljno jeje instalirati .NET Framework, koristiti Microsoftov SDK (Microsoft. Software Software Kit) i Visual Studio.

4.UML

Kao što je navedeno u opisu korištene tehnologije, konkretni projekat je otpočeo kreiranje UML dijagrama kao osnovice za dalji rak kako u bazi podataka, tako i u C#-u.

4.1.Dijagram slučajeva upotrebe

DSU kreiran je na osnovu glavnih aktivnosti koje se obavljaju u teretani, konkretno u interaciji korisnika i zaposlenog(u određenoim aktivnotima to je podklasa zaposlenog tj. Trener). Konkretne aktivnosti su: Rezervisanje privatnih treninga, kupovina programa, kupovina opreme i nutrijenata , plaćanje članarine, evidencija dolazaka, logovanje ,registracija, predaja ključa.


Teretana
Trener
Zaposleni
Korisnik
Rezervisanje privatnih treninga
Kupovina programa
Kupovina opreme i nutirijenata
Placanje clanarine
« include »
Provjera i azuriranje podataka
« include »
Realizacija registracije
Evidencija dolazaka
Korisnik nije registrovan clan
« include »
Predaja kljuca
« extend »
Registracija novog korisnika
« extend »
Novi nalog
Logovanje
Zaboravljena lozinka
Novi zaposleni
« extend »
Pomoć

Teorijski opis slučaja upotrebe:

1.Slučaj upotrebe: Logovanje

- *Kratak opis:* Logovanje zaposlenog radnika na nalog u teretani.
- *Učesnici:* Zaposleni.
- *Uslovi koji moraju biti zadovoljeni pre izvršavanja:* Svi zaposleni moraju posjedovati nalog, ukoliko to nije slučaj, potrebno je kreirati novi nalog.
- *Izuzeci:* Zaposleni se ne može ulogovati zbog greške kao što je pogrešna lozinka ili e-mail.
- *Uslovi koji moraju biti zadovoljeni nakon izvršavanja:* Rad zaposlenog na pultu.

2. Slučaj upotrebe: Realizacija registracije

- *Kratak opis:* Zaposlni registuje nove uplate i nove clanarine.
- *Učesnici:* Zaposleni
- *Uslovi koji moraju biti zadovoljeni pre izvršavanja:* Svi zaposleni moraju da imaju svoj nalog preko koga registruju članove i članovi moraju uplatiti novac za registraciju.
- *Izuzeci:* Zaposleni pogrešno unosi podatke,korisnik je dao pogrešne i neispravne podatke, greška u sistemu.
- *Uslovi koji moraju biti zadovoljeni nakon izvršavanja:* Korinik je u sistemu i može redovno da trenira.

3. Slučaj upotrebe: Evidencija Dolazaka

Kratak opis: Korisnici dolaze u teretanu i bivaju evidentirani od strane zaposlenog.

- *Učesnici:* Zaposleni i korisnici.
- *Uslovi koji moraju biti zadovoljeni pre izvršavanja:* Korisnik mora biti član teretane i biti unijet u sistem.
- *Izuzeci:* Korinsik nije član teretane.
- *Uslovi koji moraju biti zadovoljeni nakon izvršavanja:* Korisnik je evidentiran i trenira.

4. Slučaj upotrebe: Plaćanje članarine

- *Kratak opis:* Korisnici plaćaju članarinu zaposlenom.
- *Uslovi koji moraju biti zadovoljeni pre izvršavanja: /*
- *Izuzeci:* Kupac plaća pogrešan iznos.
- *Uslovi koji moraju biti zadovoljeni nakon izvršavanja:* Korisnik je platio članarinu i preuzima člansku kartu.

5. Slučaj upotrebe: Kupovina opreme i nutrijenata

- *Kratak opis:* Korisnici kupuju opremu ili suplementaciju od zaposlenog koji radi na pultu.
- *Uslovi koji moraju biti zadovoljeni pre izvršavanja:* Kupci su platili za opremu.
- *Izuzeci:* Zaposleni im daje pogrešnu opremu, kupci plaćaju pogrešan iznos.
- *Uslovi koji moraju biti zadovoljeni nakon izvršavanja:* Korisnik je preuzeo opremu i koristi je.

6. Slučaj upotrebe: Kupovina programa

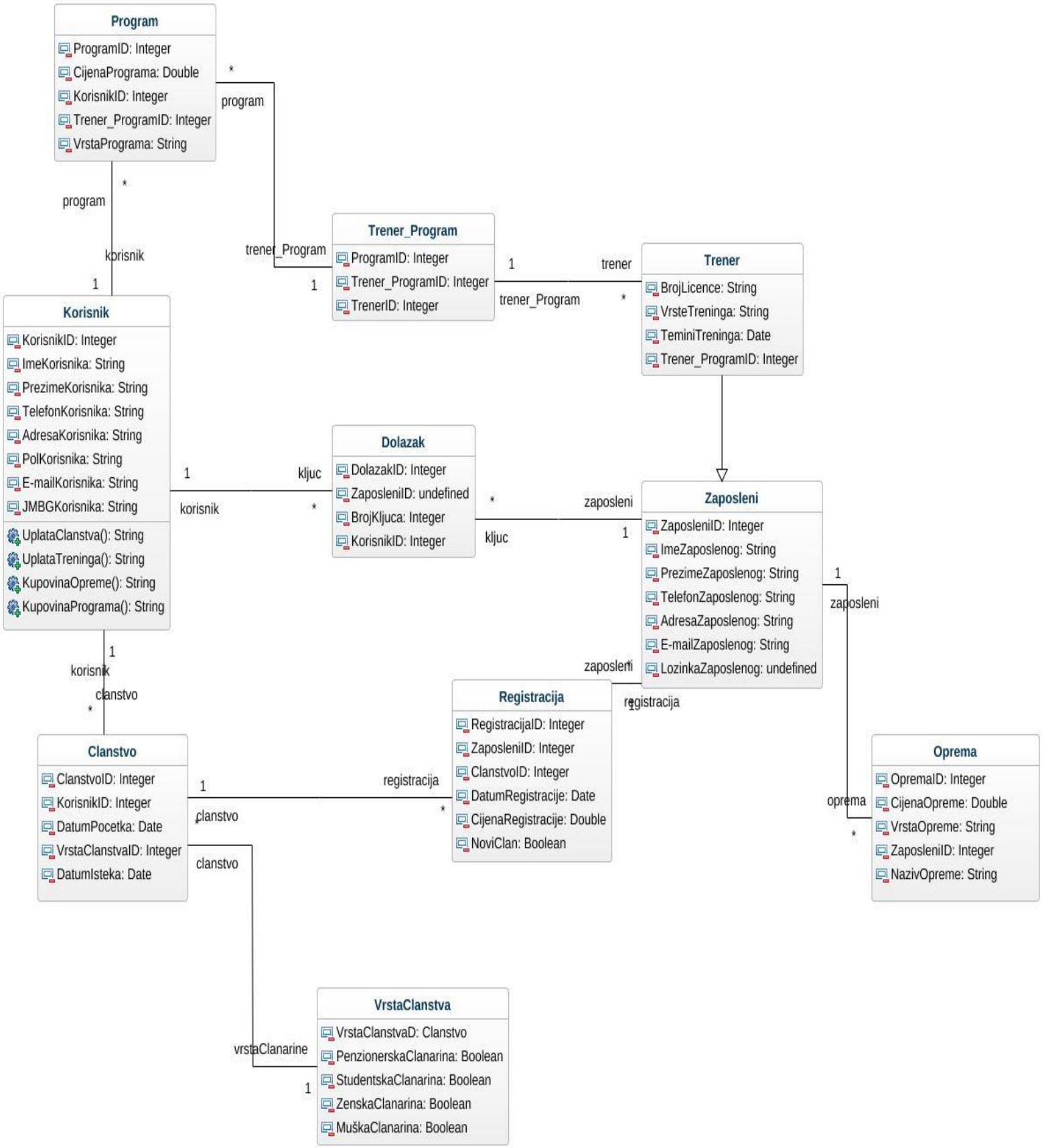
- *Kratak opis:* Korisnici kupuju poseban program za ishranu ili trening od trenera.
- *Uslovi koji moraju biti zadovoljeni pre izvršavanja:* Trener je sastavio program,program je plaćen.
- *Izuzeci:* Trener daje korisniku pogrešan program.
- *Uslovi koji moraju biti zadovoljeni nakon izvršavanja:* Korisnik je preuzeo program i nastavlja rad po njemu.

7. Slučaj upotrebe: Rezervisanje privatnih treninga.

- *Kratak opis:* Korisnici rezervišu privatne treninge kod jednog ili nekoliko trenera.
- *Uslovi koji moraju biti zadovoljeni pre izvršavanja:* Trener ima slobodan termin,korisnik je član teretane.
- *Izuzeci:*Trener nema slobodnih termina,Korisnik rezerviše pogrešan termin.
- *Uslovi koji moraju biti zadovoljeni nakon izvršavanja:* Korisnik je rezervisao trenera i trener je zauzeo slobodan termin.

4.2.Cass dijagram

Dijagram klase je dijagram na kojem se zasniva baza podataka kreirana za konkretan projekat „WPF GYM“.


Program
ProgramID: Integer
CijenaPrograma: Double
KorisnikID: Integer
Trener_ProgramID: Integer
VrstaPrograma: String
program
program
korisnik
trener_Program
Trener_Program
ProgramID: Integer
Trener_ProgramID: Integer
TrenerID: Integer
trener
trener_Program
Trener
BrojLicence: String
VrsteTreninga: String
TeminiTreninga: Date
Trener_ProgramID: Integer
Korisnik
KorisnikID: Integer
ImeKorisnika: String
PrezimeKorisnika: String
TelefonKorisnika: String
AdresaKorisnika: String
PolKorisnika: String
E-mailKorisnika: String
JMBGKorisnika: String
UplataClanstva(): String
UplataTreninga(): String
KupovinaOpreme(): String
KupovinaPrograma(): String
Dolazak
DolazakID: Integer
ZaposleniID: undefined
BrojKljuca: Integer
KorisnikID: Integer
kljuc
korisnik
zaposleni
kljuc
Zaposleni
ZaposleniID: Integer
ImeZaposlenog: String
PrezimeZaposlenog: String
TelefonZaposlenog: String
AdresaZaposlenog: String
E-mailZaposlenog: String
LozinkaZaposlenog: undefined
zaposleni
korisnik
clanstvo
zaposleni
registracija
Registracija
RegistracijaID: Integer
ZaposleniID: Integer
ClanstvoID: Integer
DatumRegistracije: Date
CijenaRegistracije: Double
NoviClan: Boolean
Clanstvo
ClanstvoID: Integer
KorisnikID: Integer
DatumPocetka: Date
VrstaClanstvaID: Integer
DatumIsteka: Date
registracija
clanstvo
clanstvo
oprema
Oprema
OpremaID: Integer
CijenaOpreme: Double
VrstaOpreme: String
ZaposleniID: Integer
NazivOpreme: String
vrstaClanarine
VrstaClanstva
VrstaClanstvaD: Clanstvo
PenzionerskaClanarina: Boolean
StudentskaClanarina: Boolean
ZenskaClanarina: Boolean
MuškaClanarina: Boolean

Konkretna **class dijagram** se sastoji od deset klasa od kojih su devet klasične klase uml dijagrama dok je jedna klasa ustvari podklasa klase Zaposleni. Veze korištene u class dijagramu su veza asocijacije u devet slučajeva i veza generalizacije u jednom slučaju. Klase:

1.Vrsta člnastva(VrstaClanstva) je klasa koja se odnosi na vrstu članstva koju korisnik odabere. Članstvo moće imati više vrsta koje određuje zaposleni ili administracija teretane. Vrsta članstva može biti redovna(Muška,Ženska),može imati s tarosne popuste(Studentska, Penzionerska), ali može i poprimati druge oblike poput specijalnih ili sezonskih. Aktivnost u oblasti kreiranja različitih vrsta članstva činiće teretanu atraktivnom te je ovaj okvir jedan od vrlo važnih u teretani.

2.Korisnik je klasa koja omugaćava evidenciju podataka o korisnicima, konkreno klasa „Korisnik“ ima popriličan broj atributa koji omogućavaju bolje poznavanje i sigurnost pri učlanjivanju korisnika u teretanu. Atributi korisnika se odnose na njegove lične podatke, ali pored podatka klasa Korisnik sadrži i akcije koje korisnik može da preduzme u okviru realnog sistema teretane. Većina podataka koje korisik daje su obavezni(Ime,Prezime,Pol,JMBG) ali nisu svi(Adresa,E-mail).

3.Članstvo(Clanstvo) predstvalja klasu koja atributima datum početka(DatumPocetka) i datum isteka(DatumIsteka) određuje trajanje koju je korisnik kupio, takođe strani ključ vrste članstva(VrstaClanstvaID) omogućava da se zna koje je konkretno članstvo kupio Korisnik. Naravno u okviru ove klase

se nalazi i strani ključ korisnika koji nam pokazuje na kojeg korisnika se odnosi članstvo.

4.Zaposleni je klasa koja se odnosi na zaposlenog radnika u teretani. Atributi ove klase su slični kao i kod korisnika, te se određeni moraju unjeti dok su drugi ostavljeni na izbor. Kao što se može vidjeti iz modela, klasa Zaposleni ima podklasu a to je Trener. Trener proširuje klasu Zaposleni i dodaje joj određene atribute kao i vezu sa klasom Program, konkretno Trener_Program kao međuklasom. Klasa Zaposleni pored veze sa klasom Program preko podklase Trener još ostvaruje veze sa klasama Registracija i Oprema. To su veze jadan na prema više u oba slučaja.

5.Oprema je klasa koja predstavlja pojedinačnu opremu različite cijene, vrste i naziva. Klasa oprema predstavlja opremu koju teretana prodaje i za koju je odgovoran Zaposleni u teretani te se zato u njoj nalazi strani ključ iz klase Zaposleni. Klasa Oprema ima samo jednu vezu asocijacije i to prema klasi Zaposleni.

6.Trener_Program klasa predstavlja međuklasu odnosno poveznicu dvije druge klase. Trener_Program klasa je poveznica izmedju klae Trener i klase Program, obzirom da jedan trener može napraviti više programa, ali i jedan program može biti napravljen od više trenera potrebno je ubaciti međuklasu koja će nositi ključeve ove dve klase i tako urediti njihovu vezu.

7.Program klasa odnosi se na programe koje treri kreiraju za korisnike.Program je određene vrste i ima konkretnu cijenu, te

dva strana ključa u njoj kako bi se koglo odrediti koji trener ili treneri i koje korisnik su povezani programom.

8.Dolazak klasa odnosi se na dnevnu evidenciju korisnika koji dolaze u teretanu. Ova klasa nema mnogo atributa jer je dovoljno evidentirati korisnike koji su došli(KorisikIDI), ključ koi su dobili(BrojKljuča) i koji zaposleni im je ključ u tom trenutku predao(Zaposleni ID).

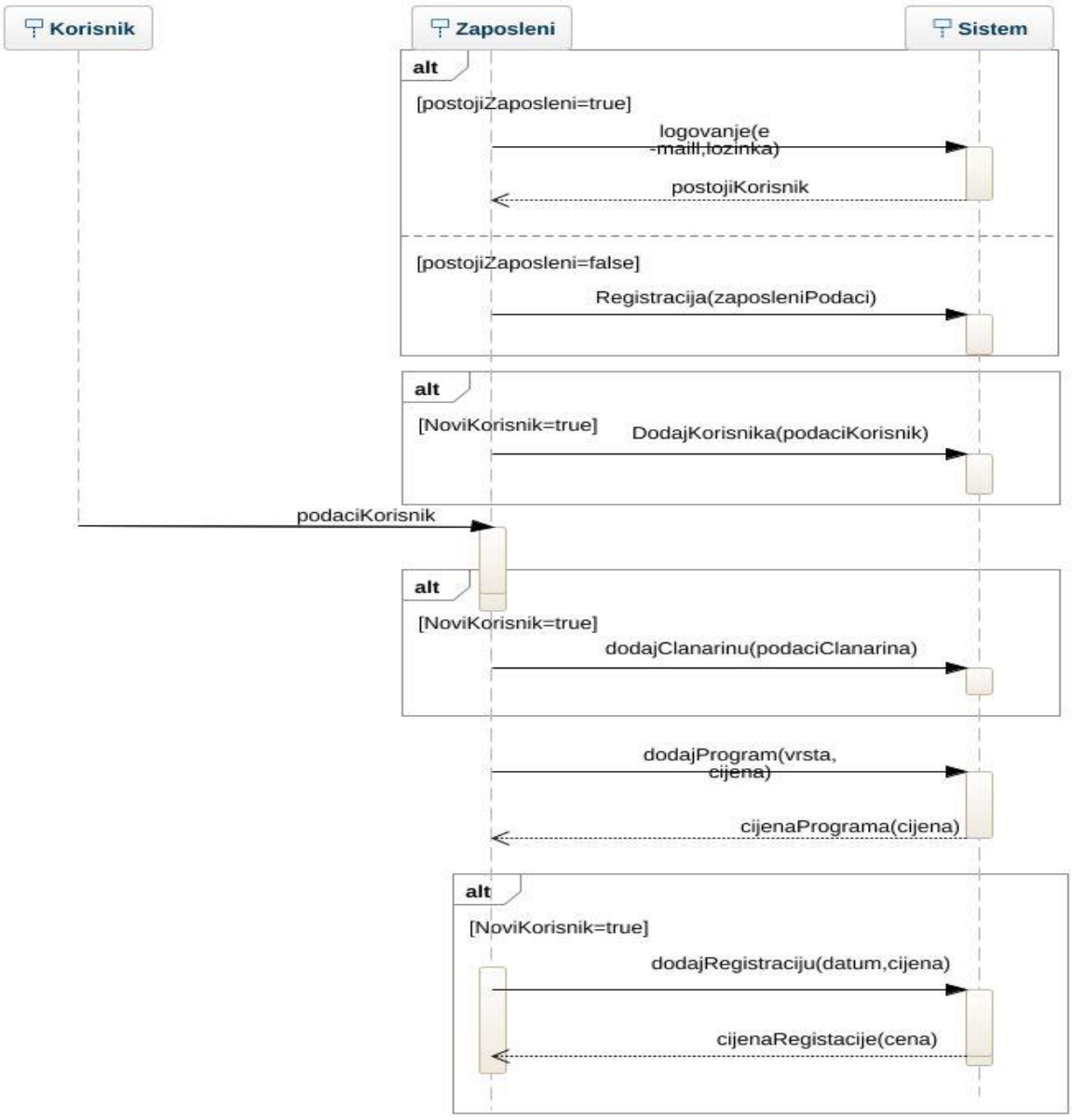
9.Registracija je klasa čiji je cilj evidentira podatke o konkretnim registracija članova koje izvršava Zaposleni. Strani ključevi u ovoj klasi su ClkanstviID i ZaposleniID te se oni odnose zaposlene koji vrše registraciju i konkretno članstvo koje je evidentirano ranije i odnosi se na konkretnog korisnika. Pored stranih ključeva atributi kojima raspolaže ova klasa su DatumRegistracije koji predstavlja datum kada se korisnik regitrovao i ne mora biti isti kao i dan početka članstva iz klase Članstvo. CijenaRegistracije predstavlja cijenu koju korisnik plača za registrovani period i NoviClan je atribut koji pokazuje da li je korisnik novi član teretane ili samo nastavlja već prijašnju registraciju.

4.3.Dijagram sekvenci: (sequence dijagram) spada u dijagrame interacija i konkretno modeluje dinamički aspekt sitema. Ovaj dijagram predstavlja implementaciju slučaja korišćenja i prikazuje interakcije između objekata u redoslijedu u kom se dešavaju. Fokus dijagrama sekvenci je na radoslijedu i tipu poruka koje se razmjenjuju a ne u detaljima

implementacije poruke. Dijagram sekvenci posjeduje dve dimenzije, a to su:

1.Vremenska(vertikala) – prikazuje životni vjek objekta.

2.Kolekcija objekta i aktera(horizontalna)



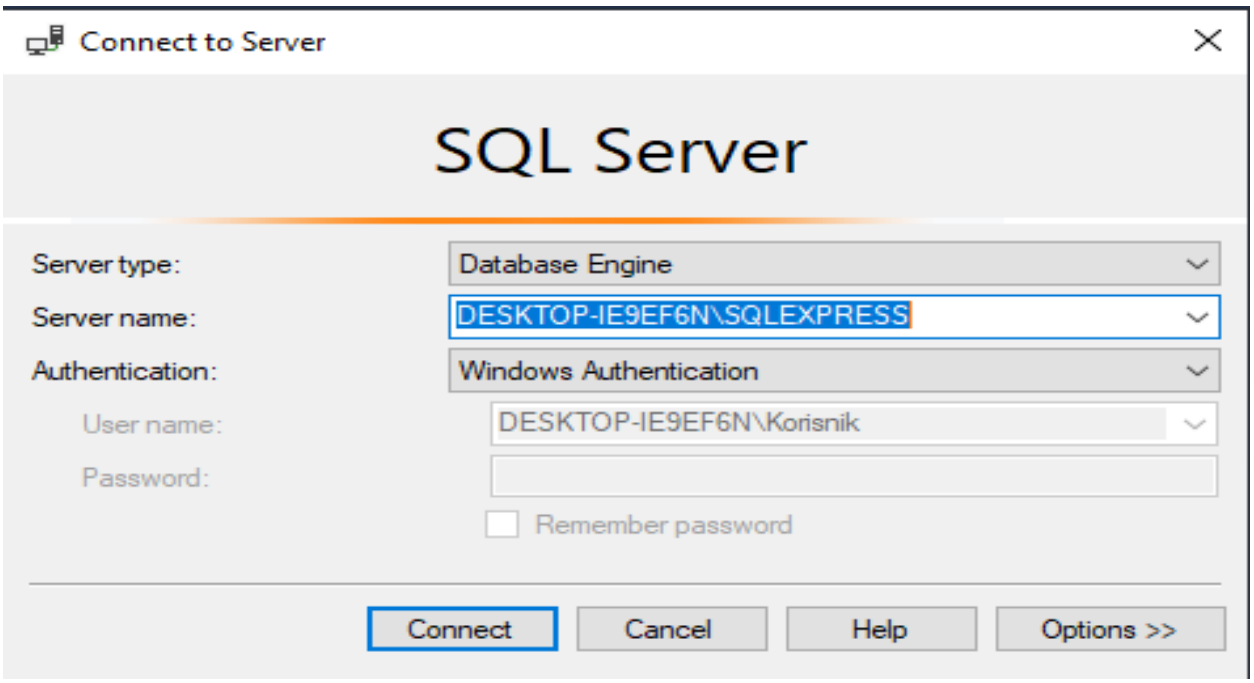
5.BAZA PODATAKA

5.1.Kreiranje baze podataka:

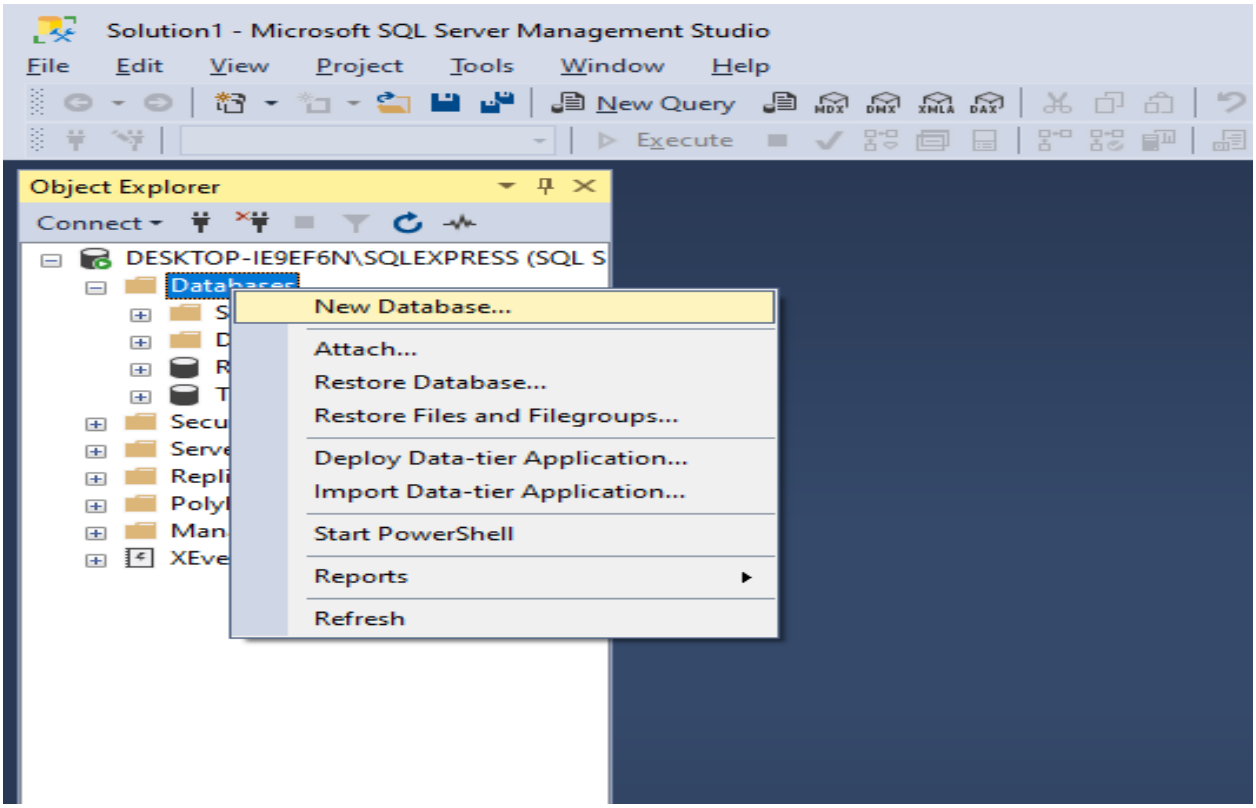
Baza podataka kreirana za „WPFGYM“ projekat satoji se od 8 tabela kreiranih u „Microsoft SQL server Managament Studio 18“. Sve kreirane tabele se odnose na klase ranije formirane u UML class dijagramu. SQL Server Management Studio (SSMS) je softverska aplikacija prvo pokrenuta s Microsoft SQL Server 2005 koja se koristi za konfiguriranje, upravljanje i administraciju svih komponenti u Microsoft SQL Serveru. Nasljednik je Enterprise Managera SQL 2000. Alat uključuje uređivače skripti i grafičke alate koji rade sa objektima i značajkama poslužitelja.

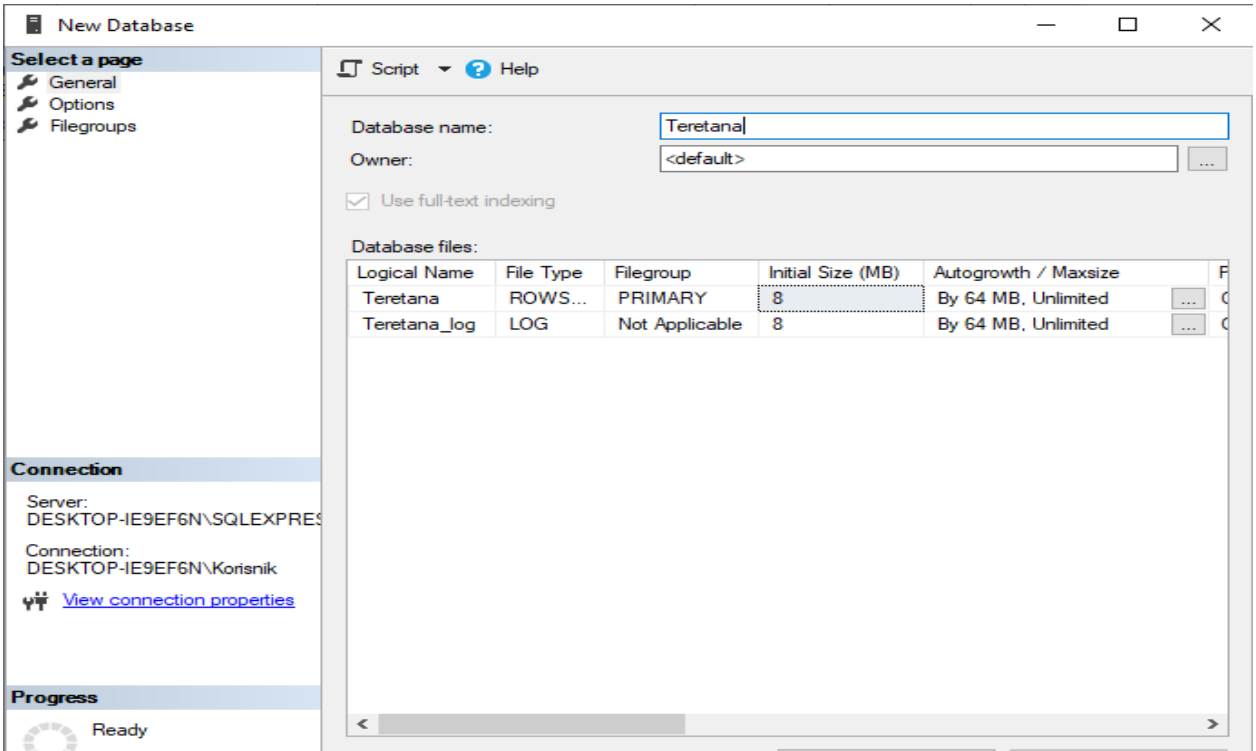
Baza podatka je kreirana na sledeći način:

Otvorimo Microsoft SQL server Managament Studio 18 i povežemo se na lokali server:

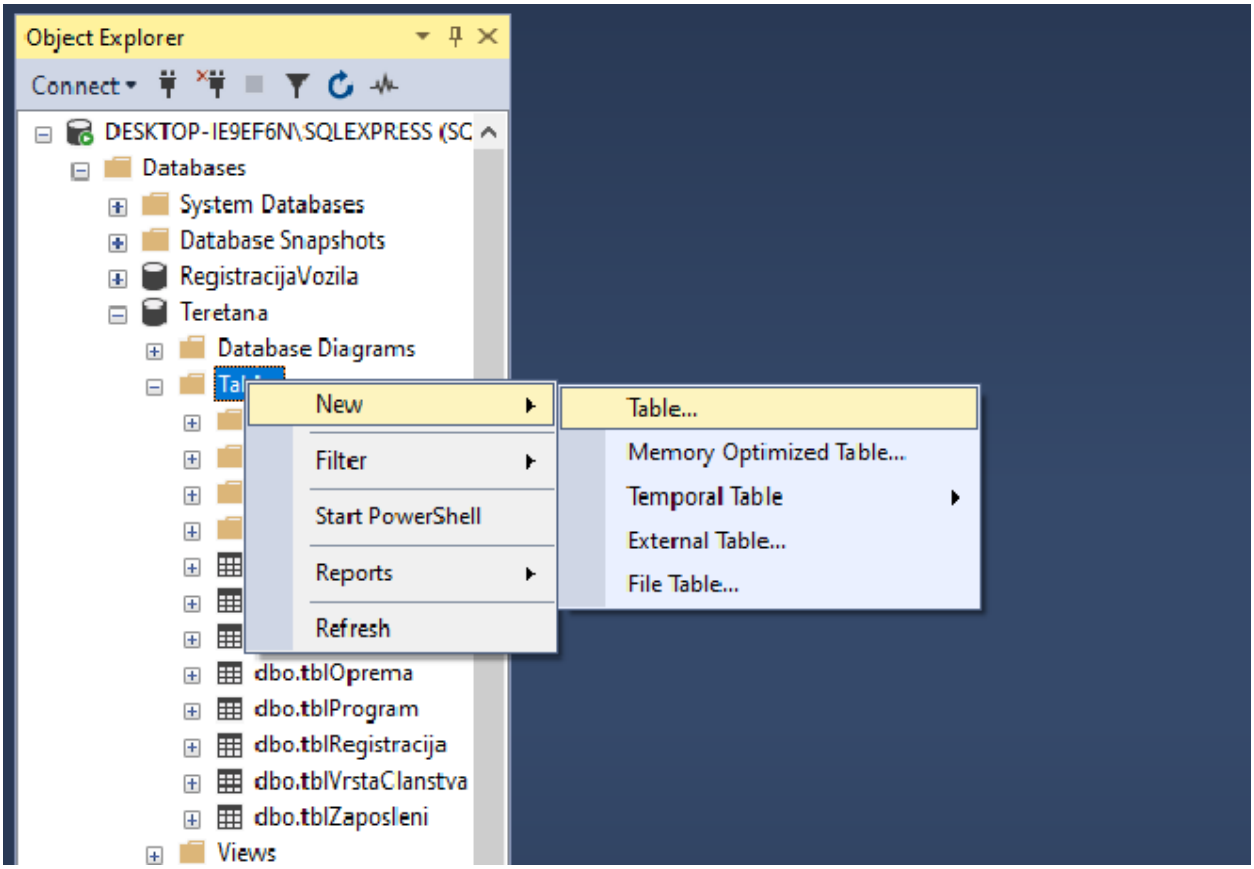


Zatim kreiramo novu bazu podataka I dajemo joj ime na način koji je prikazan na slikama ispod:

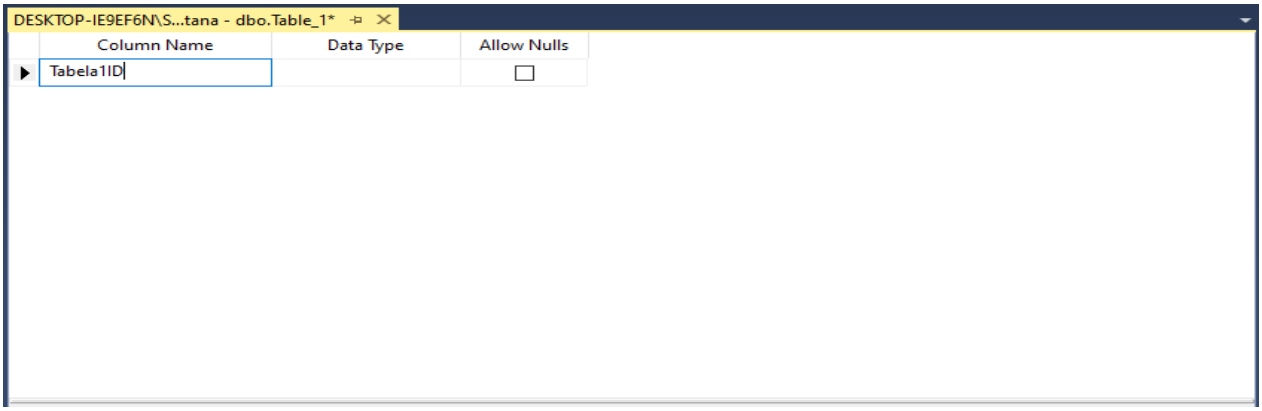




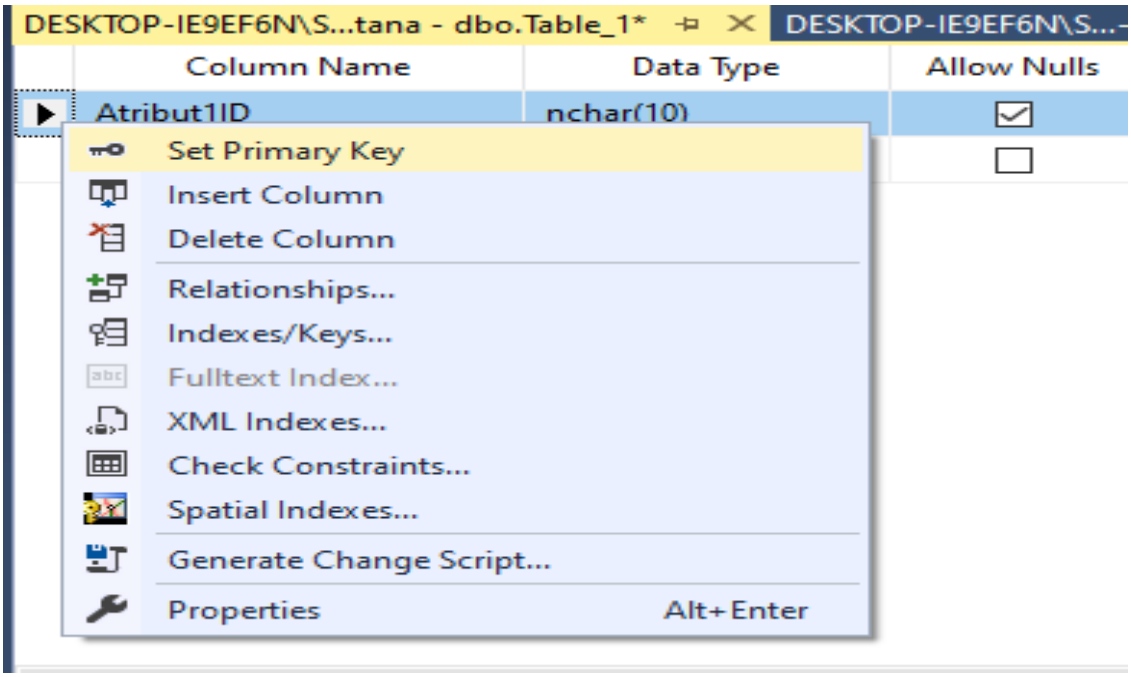
Nakon kreiranja baze podatka potrebno je kreirati tabele na osnovu tabela napravljenih u class dijagramu. Tebele se kreiraju na sledeći način:



Nakon kreiranja tabele, potrebno joj je definisati primarni ključ, atribute kao I strane ključeve ako ih tabela posjeduje. Atributi se kreiraju na osnovu atributa u class dijagramu. Kreiranje atributa izgleda ovako:



Setovanje primarnog ključa je jedna od osnovnih komandi u kreiranju tabela baze podataka.Primarni ključ se setuje na sledeći način:



Za konkretan projekat “WPFGYM” kreirano je 8 tabela neophodih za rad aplikacije, a to su: „tblClanstvo“, „tblDolazak“, „tblKorisnik“,“tblOprema“, „tblProgram“, „tblRegistracija“, „tblVrstaClanstva“ I „tblZaposleni“. Konkretne tabele su prikazane na sledećim fotografijama:

tblClanstvo(Članstvo):

DESKTOP-IE9EF6N\S...- dbo.tblClanstvo

Column Name	Data Type	Allow Nulls
ClanstvoID	int	☐
DatumPocetka	date	☑
DatumIsteka	date	☑
KorisnikID	int	☑
VrstaClanstvaID	int	☑
		☐

tblDolazak (Dolazak):

DESKTOP-IE9EF6N\S...a - dbo.tblDolazak | DESKTOP-IE9EF6N\S...- d

Column Name	Data Type	Allow Nulls
DolazakID	int	☐
BrojKljuca	int	☑
ZaposleniID	int	☑
KorisnikID	int	☑
		☐

tblKorisnik (Korisnik):

DESKTOP-IE9EF6N\SQ...- dbo.tblKorisnik | DESKTOP-IE9EF6N\S...a

Column Name	Data Type	Allow Nulls
KorisnikID	int	☐
ImeKorisnika	nvarchar(20)	☐
PrezimeKorisnika	nvarchar(20)	☐
JMBGKorisnika	nvarchar(13)	☐
TelefonKorisnika	nvarchar(20)	☑
AdresaKorisnika	nvarchar(50)	☑
PolKorisnika	nvarchar(20)	☐
EmailKorisnika	nvarchar(50)	☑
		☐

tblOprema (Oprema):

DESKTOP-IE9EF6N\S...a - dbo.tblOprema | DESKTOP-IE9EF6N\SQ...-

Column Name	Data Type	Allow Nulls
OpremaID	int	☐
CijenaOpreme	numeric(18, 0)	☑
VrstaOpreme	nvarchar(30)	☑
NazivOpreme	nvarchar(30)	☑
ZaposlenilD	int	☑
		☐

tblProgram (Program):

DESKTOP-IE9EF6N\S...- dbo.tblProgram | DESKTOP-IE9EF6N\S...a -

Column Name	Data Type	Allow Nulls
ProgramID	int	☐
VrstaPrograma	nvarchar(20)	☑
CijenaPrograma	nvarchar(20)	☑
KorisnikID	int	☑
ZaposlenilD	int	☑
		☐

tblRegistracija (Registracija):

DESKTOP-IE9EF6N\SQ...bo.tblRegistracija | DESKTOP-IE9EF6N\S...-

Column Name	Data Type	Allow Nulls
RegistracijaID	int	☐
DatumRegistracije	date	☑
CijenaRegistracije	decimal(18, 0)	☐
NoviClan	bit	☑
ZaposlenilD	int	☑
ClanstvoID	int	☑
		☐

tblVrstaClanstva(VrstaClanstva)

DESKTOP-IE9EF6N\S...o.tblVrstaClanstva | DESKTOP-IE9EF6N\SQ...

Column Name	Data Type	Allow Nulls
VrstaClanstvaID	int	☐
NazivVrsteClanstva	nvarchar(30)	☑
MuskaClanarina	bit	☐
ZenskaClanarina	bit	☐
PenzionerskaClanarina	bit	☐
StudentskaClanarina	bit	☐
		☐

tblZaposleni (Zaposleni)

DESKTOP-IE9EF6N\S...- dbo.tblZaposleni | DESKTOP-IE9EF6N\S...o.t

Column Name	Data Type	Allow Nulls
ZaposleniID	int	☐
ImeZaposlenog	nvarchar(20)	☐
PrezimeZaposlenog	nvarchar(20)	☐
Trener	bit	☐
AdresaZaposlenog	nvarchar(20)	☑
TelefonZaposlenog	nvarchar(20)	☑
EmailZaposlenog	nvarchar(20)	☐
Lozinka	nvarchar(20)	☐
		☐

tblClantvo(Članstvo) kao I sve tabele u konkretnoj bazi teretana sadrži primarni ključ “ClanstvoID”, datum početka trajanja članstva(DatumPocetka) koji je tima int,datum kraja trajanja članstva(DatumISteka) isto tipa int I dva strana ključa: „KorisnikID“ I „VrstaClanstvaID“ (vrsta članstva) tima int(primarni I strain ključevi su uvijek tipa int).Svi podaci u ovoj tabeli osim primarnog ključa mogu biti null.

tblDolazak(Dolazak) pored primarnog ključa „DolazakID“ posjeduje I dva strana ključa: KorisnikID koji nam pokazuje koji Korisnik uzima ključ, ZapolseniID koji nam pokazuje koji zaposleni daje ključ korisniku. Ova tebela pored ključeva posjeduje I jedan atribut: BrojKljuca(broj ključa) koji je tima Int I označava broj ključa je zaposleni dao korisniku.

tblKorisnik(Korisnik) je tabela sa primarnik ključem “KorisnikID” I sa nizom atributa koji su vezani za lične podatke o korisniku. Konkretni atribudti redom predstavljaju ime, prezime, JMBG, telefon, adresu, pol I email korisnika. Kao što se može vidjeti na slici ove tabele svi atributi su tipa “nvarchar(duzina)”. Ime, prezime I pol korisnika su najznačajnije za konkretni rad teretane pa oni ne mogu biti null, dok je ostalim atributima to dozvoljeno.

tblOprema(Oprema) je tabela koja se odnosi na opremu koju teretana prodaje te pored primarnog ključa “OpremaID” I stranog kluča ”ZaposleniID” koji se ondosi na zaposlenog koji je prodao konkretnu opremu, tblOprema posjeduje i tri atributa: CijenaOpreme(cijena opreme) koja je tipa „numeric“ jer se odnosi na novac,VrstaOpreme i NazivOpreme koji se odnose na vrstu i naziv konkretne opeme i koji su tipa „nvarchar(duzina)“.

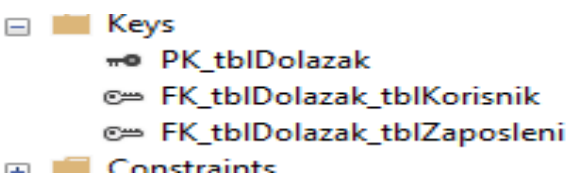
tblProgram(Program) je tabela koja reprezentuje program koji korisnik kupuje od zaposlenog odnosno koji zaposleni piše za korisnika. Pored primarnog ključa „ProgramID“ tu su i dva strana ključa: „KorisnikID“ i „ZaposleniID“ iz prthodno navedenih razloga. VrstaPrograma(vrsta programa) tipa „nvarchar()“ i CijenaPrograma(cijena programa) tipa „numeric“ se odnose na vrstu i konkretnu cijenu pisanog programa.

tblRegistracija(Registracija) je tabela sa primarnik ljučem „RegistracijaID“ i dva strana ključa: „ZaposleniID“ i „ČlanstvoID“ koji se odnose na zaposlenog koji je u sistem uneo konkretno članstvo, odnos no registraciju korisnika i konkretno članstvo koje korisik uzima. Obzirom da početni datum članstva i sama registracija ne moraju biti na isti dan atribut „DatumRegistracije“ reguliše dan regitrovanja, a „CijenaRegistracije“ tipa „numeric“ i „NoviClan“ tipa „bit“ nam pokazuju cijenu konkretne clanarine odnosno registracije i da li je korisnik novi član teretane ili samo produžava članstvo.

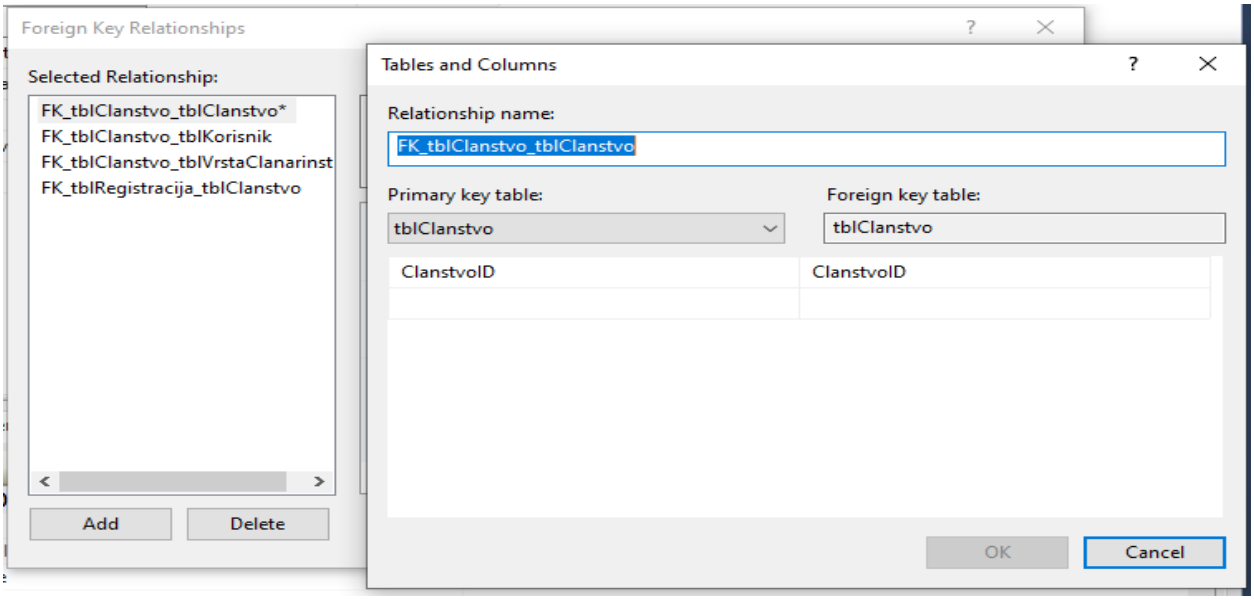
tblVrstaClanstva je table koja se odnosi na osobine članstva kojeg korisnik uzima. Pored primarnog ključa „VrstaClanstvaID“, tu se nalaze i tributi koji pokazuju kome je konkretno članstvo namjenjeno odnosno kojoj grupi ljudi: „MuskaClanarina“, „ZenskaClanarina“, „StudenstvaClanarina“ , „PenzionerskaClanarina“ su tipa „bit“ dok je atribut „NazivVrsteClanstva“ tipa „nvarchar()“ i pokazuje ime konkretne vrste članstva.

tblZaposleni kao i tabela tblKorisnik pored primarnog kluča „ZaposleniID“ ima niz atributa koji se odnose na podatke zaposlenog, ato su: ime, prezime, adresa, telefon, email koji su tipa „nvarchar()“. Atributi „Trener“ tipa bit i „Lozinka“ tipa „nvarchar()“ pokazuju da li je zaposleni ujedino i trener odnosno lozinku zaposlenog za logovanje u sam sistem tertane.

Kao što je ranije navedeno, veliki broj tabela posjeduje strane ključeve koji ih uvezuju sa drugim teretanama.Primjer tblDolazak:



Strani ključevi se dodaj na sledeći način(Primjer tblClanstvo):



Nakon kreiranja svih tabela neophodnih za rad nad bazom, potrebno je kreirate upite kojima se upravlja podacima u bazi podatataka. Tri osnovan upita su : „Insert“, „update“ i „delete“. Pomoću insert upita se unose podaci u tabelu, postoji više načina za pisanje ovog upita, a prikazani u nastvaku su najčešći načini. Update se koristi za azuriranje podataka u bazi i najčešće se radi pomoću ID-a. Delete upiti služe za brisanje podatka iz tabele, postoji mogućnost brisanja svih podataka ili po specifičnom indeksu. Upiti se kreiraju u Query stranica za kodiranje u bazi. U nastavku biti navedeni po jedan upit za svaku kreiranu tabelu u bazi.

5.2.Upiti i DML naredbe:

Upiti za tblKorisnik:

```
insert into tblKorisnik
values('Mladen','Bajić','2208999180865','064/256-645','Cara Uroša
44','Muško','mladen_ba@mile.com');


Update tblKorisnik
set EmailKorisnika='malden_ba@live.com'
where KorisnikID=4


delete from tblKorisnik
where KorisnikID=1;
```

Upiti za tblOprema:

```
insert  into tblOprema
values('20','trening','Majica XR','1');


Update tblOprema
set CijenaOpreme='27',NazivOpreme='Battery Protein 1.3KG'
where OpremaID=1;


delete from tblOprema
where OpremaID=1;
```

Upiti za tblVrstaClanstva:

```
insert into tblVrstaClanarinstva
values('Opšta','0','1','0','1');



Update tblVrstaClanstva set
ClanarinaPoPolu='Muška',PenzionerskaClanarina='0',StudentskaClanarina='0'
where VrstaClanstvaID=1;


delete from tblVrstaClanstva
where VrstaClanstvaID=1;
```

Upiti za tblClanstvo:

```
insert into
tblClanstvo(DatumIsteka,DatumPocetka,VrstaClanstvaID,KorisnikID)
values ('2019-10-2','2019-7-1','3','3')


update tblClanstvo
 set KorisnikID='5',VrstaClanstvaID='3'
 where ClanstvoID=3;


select*
from tblClanstvo
```

Upiti za tblZaposleni:

```
insert into tblZaposleni
values('Miloš','Knićanin','1','Šumaherova
11','064/221/222','milošfit24@gmail.com','Milos11')


update tblZaposleni
set AdresaZaposlenog='Krlosa Sainza 95',EmailZaposlenog='emina2@live.com'
where ZaposleniID=4;


delete from tblZaposleni
where ZaposleniID=4;
```

Upiti za tblRegistracija:

```
insert into tblRegistracija
values('2019-5-1','3000','1','1','1')


update tblRegistracija
set DatumRegistracije='2019-4-30'
where RegistracijaID=1;



delete from tblRegistracija
where RegistracijaID=4;
```

Upiti za tblDolazak:

```
insert into tblDolazak
values('201','1','1')


update tblDolazak
set BrojKluča='302'
where DolazakID=1;


delete from tblDolazak
where DolazakID=3;
```

Upiti za tblDolazak:

```
insert into
tblProgram(KorisnikID,ZaposleniID,VrstaPrograma,CijenaPrograma)
values('4','1','ishrana','1700');


update tblProgram
set VrstaPrograma='ishrana i trening',CijenaPrograma='4900'
where ProgramID=1;


delete from tblProgram
where ProgramID=3;
```

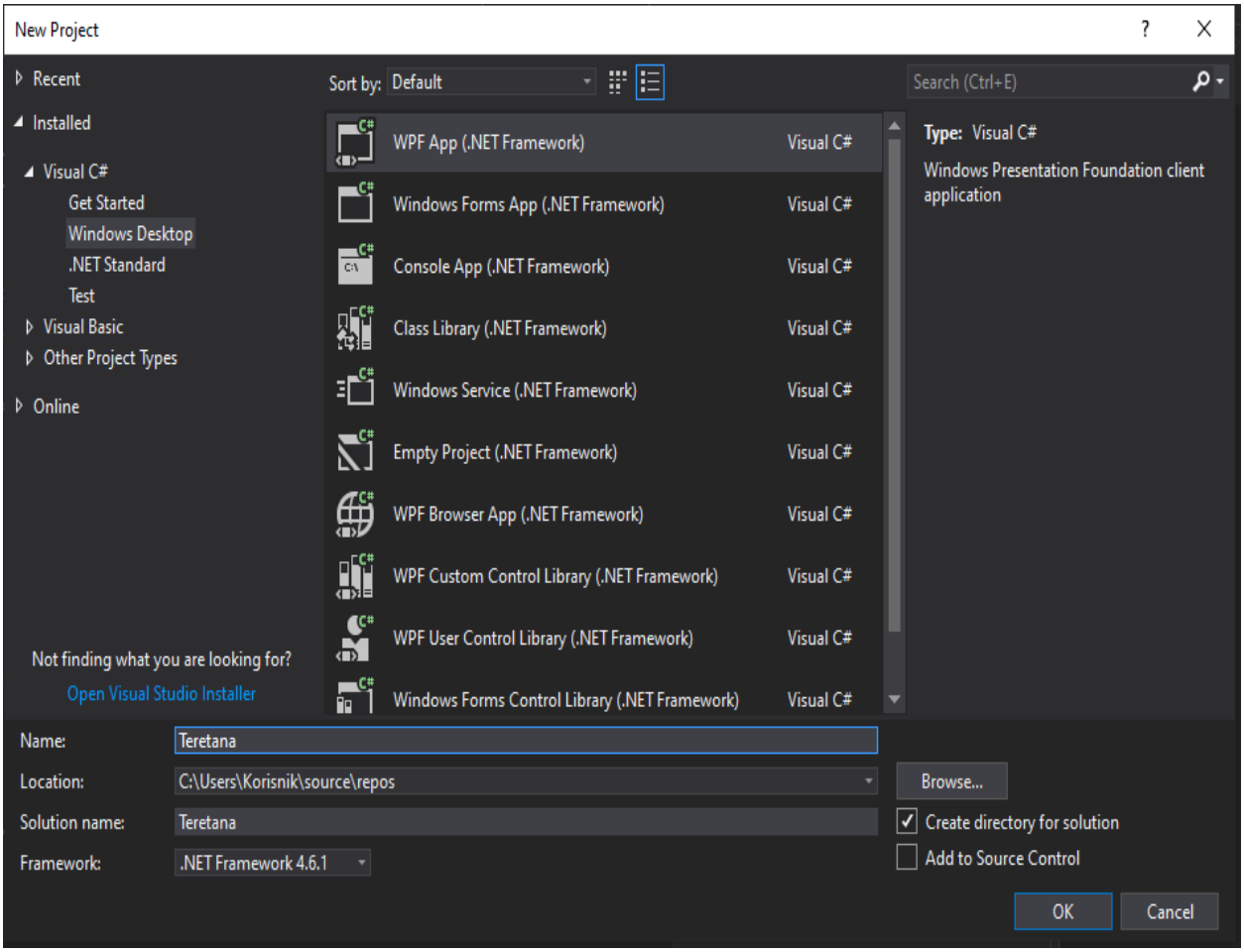
Nakon ubacivanja podataka u tabele, moguće je kombinovati podatke uz pomoć stranih upita i tako stvarati tabele ralnog sitema. Ovakvi upiti se nazivaju „Select“ upit i koriste se za izlistavanje podatka. Primjer iz baze teretana:

```
select tblKorisnik.KorisnikID, tblKorisnik.ImeKorisnika+' '+
tblKorisnik.PrezimeKorisnika as 'Ime i prezime',
tblRegistracija.DatumRegistracije as 'Datum
registrovanja',tblRegistracija.NoviClan as 'Novi član',
tblVrstaClanstva.NazivVrsteClanstva as 'Vrsta
članarine',tblClanstvo.DatumPocetka as 'Datum početka',
tblClanstvo.DatumIsteka as 'Datum
isteka',tblZaposleni.ImeZaposlenog+'
'+tblZaposleni.PrezimeZaposlenog as
'Odgovorni zaposleni'
from tblRegistracija join tblClanstvo on
tblRegistracija.ClanstvoID=tblClanstvo.ClanstvoID
                        join tblKorisnik on
tblClanstvo.KorisnikID=tblKorisnik.KorisnikID
                        join tblVrstaClanstva on
tblClanstvo.VrstaClanstvaID=tblVrstaClanstva.VrstaClanstvaID
                        join tblZaposleni on
tblRegistracija.ZaposleniID=tblZaposleni.ZaposleniID
```

Nakon kreiranja baze podataka i savladanih upita vrijeme je da se pređe na kreiranje WPF aplikacije u Microfot Visual Studio 2017.

6.Kreiranje WPF aplikacije

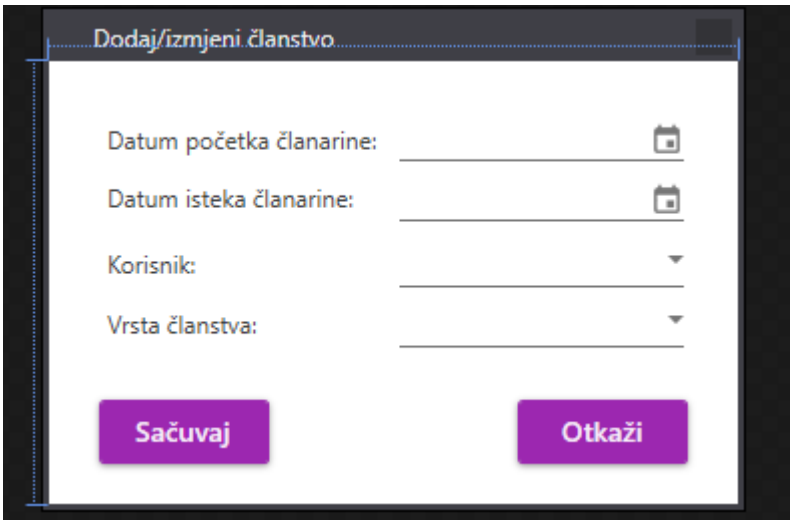
Prvi korak nakon što se pokrene Microsoft Visual Studio jeste da se kreira WPF projekat i to se rad na način prikazan na slici:



Nakon kreiranja WPF aplikacije potrebno je kreirati klasu „Konekcija“ kojom uspostavljamo konekciju sa bazom podataka. Kod za konkretnu klasu je sledeći:

```
class Konekcija
    {
        public static SqlConnection KreirajKonekciju()
        {
            SqlConnectionStringBuilder ccnSB = new SqlConnectionStringBuilder();
            ccnSB.DataSource = @"DESKTOP-IE9EF6N\SQLEXPRESS";
            ccnSB.InitialCatalog = "Teretana";
            ccnSB.IntegratedSecurity = true;

            string com = ccnSB.ToString();
            SqlConnection konekcija = new SqlConnection(com);
            return konekcija;
        }

    }
```

Pošto je konekcija kreirana, vrijeme je da se kreiraju forme i kod koji se nalazi iza tih formi. Primjer forme Članstvo:



Glavno mjesto sa koge se upravlja aplikacijom se naziva MainWinow odnosno glavni prozor i uz pomoć njega dodajemo, mijenjamo i brišemo podatke iz baze. Takođe na glavnom prozoru se prikazuju podaci koje želimo. MainWinow aplikacije „WPFGYM“:

Kao što vidimo glavni prozor nam omogućava da dodamo, izmjenimo i obrišemo podatke u tabelama baze podataka. Da bi se određeni podaci dodali potreno je da bude učitana tabela u koju želimo da dodamo podatke. Kada je riječ o opcijama „Izmjeni“ i „Obriši“ potrebno je selektovati red koji želimo da izmjenino ili obrišemo. Sve podatke koje vidimo prikazuje nam se na DataGrid-u koji možemo primjetiti na samoj sredini glavnog prozora.

Učitavanje podatka u DataGrid je proces koji se radi na osnovu „Select“ upita koji se iz aplikacije prosleđuju bazi podatka i iz baze podataka se iščitavaju tabele te se prikazuju na DataGrid.

```
public static void UcitajPodatke(DataGrid grid,string selectUpit)
        {
            try
            {
                konekcija.Open();
                DataTable dt = new DataTable();
                SqlDataAdapter da = new SqlDataAdapter(selectUpit, konekcija);
                da.Fill(dt);
                grid.ItemsSource = dt.DefaultView;
                ucitanaTabela = selectUpit;
            }
            catch (Exception)
            {
                  MessageBox.Show("Neuspešno učitani podaci!", "Greška",
                  MessageBoxButton.OK, MessageBoxImage.Error);
            }
            finally {
                if (konekcija != null)
                {
                    konekcija.Close();
                }
            }
        }
```

Primjer jednog „Select“ upita na osnovu kog se učitavaju podaci iz izabranih tabela na DataGrid:

```
static string clanstvoSelect = @"select ClanstvoID as ID,
Convert(nvarchar(10),DatumPocetka,103) as 'Datum početka',
Convert(nvarchar(10),DatumIsteka,103) as 'Datum
isteka',ImeKorisnika+' '+PrezimeKorisnika
as 'Korisnik',NazivVrsteClanstva as 'VrstaClanstva' from tblClanstvo
join tblKorisnik on tblKorisnik.KorisnikID=tblClanstvo.KorisnikID
join tblVrstaClanstva on
tblVrstaClanstva.VrstaClanstvaID=tblClanstvo.VrstaClanstvaID ";
```

Kada se pokrene glavni prozor moguće je odmah učitati podatke iz odabrane tabele samim pozivanjem funkcije u konstruktoru:

```
public MainWindow()
        {

           UcitajPodatke(DataGridCentralni, dolazakSelect);
        }
```

Nakon što osoba koja upravlja aplikacijom klikne određeno drugme koje predstavlja tabelu, potrebno je da se ta tabela učita u DataGrid. Potrebno je dodati event na svako dugme u MainWindow i u konkretan event ubaciti kreiranu metodu UcitajPodatke(). Primjer za dugmad „Dolazak“ i „Zaposleni“.

```
private void btnDolazak_Click(object sender, RoutedEventArgs e)
        {
            UcitajPodatke(DataGridCentralni, dolazakSelect);
        }

private void btnZaposleni_Click(object sender, RoutedEventArgs e)
        {
            UcitajPodatke(DataGridCentralni, zaposleniSelect);
        }
```

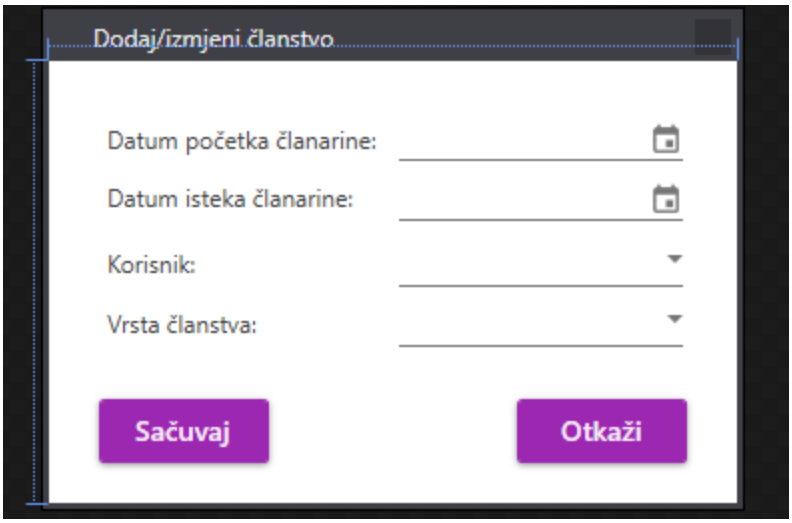
Nakon klika na određeno dugme učitaće se podatci iz konkretne tabele koja je izabrana klikom.

6.1.Dodavanje

Dodavanje je opcija pod kojom se podrazumjeva dodavanje objekta određene klase odnosno dodavanja jednog objekta u tabelu baze podataka. Primjer je dodavanje novog korisnika teretane. Potrebno je da zaposleni unese podatke koje mu korisnik da i zatim da sačuva te podatke koje kasnije može koristiti i koji će mu se prikazivati u DataGrid-u. Mora se provjeriti koja je učitana tabela da bi se mogao otvoriti prozor za dodavanje tog objekta.

```
        private void btnDodaj_Click(object sender, RoutedEventArgs e)
        {
            Window prozor;

            if (ucitanaTabela.Equals(clanstvoSelect)){

                prozor = new frmClanstvo();
                prozor.ShowDialog();

                UcitajPodatke(DataGridCentralni, clanstvoSelect);
            }
            else if (ucitanaTabela.Equals(dolazakSelect))
            {
                prozor = new frmDolazak();
                prozor.ShowDialog();

                UcitajPodatke(DataGridCentralni, dolazakSelect);
            }
            else if (ucitanaTabela.Equals(korisnikSelect))
            {
                prozor = new frmKorisnik();
                prozor.ShowDialog();

                UcitajPodatke(DataGridCentralni, korisnikSelect);
            }
    }
```

Nakon što se provjeri koja je učitana tabela otvara se prozor za dodavanje/izmjenu podataka.



Ukoliko je zaposleni pogrešio formu ili jednostavno želi da odustane od popunjavanja on može jednostavno pritisnuti dugme „Otkaži“ i time će se zatvoriti prozor bez toga da se podaci sačuvaju.

```
private void btnOtkazi_Click(object sender, RoutedEventArgs e)
        {
            this.Close();
        }
```

Nakon popunjavanja forme potrebno je da zaposleni sačuva podatke time što će pritisnuti na dugme sačuvaj. Pritiskom na ovo dugme izvršava se metoda koja unesene podatke prosleđuje bazi kao „insert“ upit:

```
Try {
Konekcija.open()
string insert = @"insert into
tblProgram(VrstaPrograma,CijenaPrograma,KorisnikID,ZaposleniID)
                                  values('" + txtVrstaPrograma.Text + "','" +
txtCijenaPrograma.Text + "'," +
                                 "'" + cbKorisnik.SelectedValue + "','" +
cbZaposleni.SelectedValue + "')";
                    SqlCommand cmd = new SqlCommand(insert, konekcija);
                    cmd.ExecuteNonQuery();

}
```

Ukoliko dođe do greške u kodu „catch“ blok će uhvatiti grešku i korisniku proslijediti poruku. „Finally“ blok zatvara konekciju. Ovo je slučaj sa svim formama i eventom „Sačuvaj“.

```
  catch (Exception)
            {
                MessageBox.Show("Unos određenih podataka nije validan! Molimo Vas
poušajte ponovo!",
                   "Greška", MessageBoxButton.OK, MessageBoxImage.Error);
            }
            finally
            {
                if (konekcija != null)
                {
                    konekcija.Close();
                }
            }
        }
```

6.2.Izmjeni(Ažuriraj) podrazumjeva da zaposleni može da izmjeni već postojeće podatke u aplikaciji tako što će selektovati red koji želi da izmjeni u DataGrid-u i pritiskom na dugme „Izmjeni“ mu se otvara forma već popunjena podacima.

Forme se popunjavaju na osnovu metode koja se kreira u MainWindow.xaml.cs:

```
  static void PopuniFormu(DataGrid grid,string selectUslov)
        {
            try
            {
                konekcija.Open();
                azuriraj = true;
                DataRowView red = (DataRowView)grid.SelectedItems[0];
                pomocniRed = red;
                string upit = selectUslov + red["ID"];
                SqlCommand cmd = new SqlCommand(upit, konekcija);
                SqlDataReader citac = cmd.ExecuteReader();
                while (citac.Read())
                {
                    if (ucitanaTabela.Equals(korisnikSelect))
                    {
                        frmKorisnik prozorKorisnik = new frmKorisnik();

                        prozorKorisnik.txtImeKorsinika.Text =
citac["ImeKorisnika"].ToString();
                        prozorKorisnik.txtPrezimeKorisnika.Text =
citac["PrezimeKorisnika"].ToString();
                        prozorKorisnik.txtJMBGKorisnika.Text =
citac["JMBGKorisnika"].ToString();
                        prozorKorisnik.txtTelefonKorisnika.Text =
citac["TelefonKorisnika"].ToString();
                        prozorKorisnik.txtAdresaKorisnika.Text =
citac["AdresaKorisnika"].ToString();
                        prozorKorisnik.txtPolKorisnika.Text =
citac["PolKorisnika"].ToString();
                        prozorKorisnik.txtEmailKorisnika.Text =
citac["EmailKorisnika"].ToString();

                        prozorKorisnik.ShowDialog();

                            }

                    }

                }

        }
```

Prije nego što se pozove sama metoda potrebno je kao i kod opcije „Dodaj“ da se provjeri učitana tabela:

```
private void btnIzmjeni_Click(object sender, RoutedEventArgs e)
        {
            if (ucitanaTabela.Equals(korisnikSelect))
            {
                PopuniFormu(DataGridCentralni, selectUslovKorisnik);
                UcitajPodatke(DataGridCentralni, korisnikSelect);

            }else if (ucitanaTabela.Equals(clanstvoSelect))
            {
                PopuniFormu(DataGridCentralni, selectUslovClanstvo);
                UcitajPodatke(DataGridCentralni, clanstvoSelect);

            }else if (ucitanaTabela.Equals(dolazakSelect))
            {
                PopuniFormu(DataGridCentralni,selectUslovDolazak);
                UcitajPodatke(DataGridCentralni, dolazakSelect);

            }

      }
```

Nakon što izmjeni ono što je naumio, potrebno je da zaposleni pritisne dugme sačuvaj na formi, provjerava se globalna promjeniva i aktivira se opcija ažuriraj:

```
konekcija.Open();
                if (MainWindow.azuriraj)
                {
                    DataRowView red = MainWindow.pomocniRed;

                    string update=@"update tblProgram set
VrstaPrograma='"+txtVrstaPrograma.Text+"',CijenaPograma='"+txtCijenaPrograma.Text+"',
" +

"KorisnikID='"+cbKorisnik.SelectedValue+"',ZaposleniID='"+cbZaposleni.SelectedValue+"'
where ProgramID=" +red["ID"];

                    SqlCommand cmd = new SqlCommand(update, konekcija);
                    cmd.ExecuteNonQuery();
                    MainWindow.pomocniRed = null;
                }
```

Kada je riječ o popunjavanju ComboBox polja, konkretno učitavanju ComboBox polja, potrebno ih je učitati u kontruktoru svake forme koja posjeduje ComboBox. Primjer:

```
try
            {
                string vratiKorisnike = @"select
KorisnikID,ImeKorisnika +' '+ PrezimeKorisnika +' '+JMBGKorisnika
as Korisnik from tblKorisnik";
                DataTable dtKorisnik = new DataTable();
                SqlDataAdapter daKorisnik = new
SqlDataAdapter(vratiKorisnike, konekcija);
                daKorisnik.Fill(dtKorisnik);
                cbKorisnik.ItemsSource = dtKorisnik.DefaultView;

                string vratiZaposlene = @"select
ZaposleniID,ImeZaposlenog +' '+ PrezimeZaposlenog as Zaposleni
from tblZaposleni where Trener='1'";
                DataTable dtZaposleni = new DataTable();
                SqlDataAdapter daZaposleni = new
SqlDataAdapter(vratiZaposlene, konekcija);
                daZaposleni.Fill(dtZaposleni);
                cbZaposleni.ItemsSource = dtZaposleni.DefaultView;

            }
            catch(Exception)
            {
                MessageBox.Show("Padajuće liste nisu popunjene!",
"Greška!", MessageBoxButton.OK, MessageBoxImage.Error);
            }
            finally
            {
                if (konekcija != null)
                {
                    konekcija.Close();
                }
            }
```

6.3.Obriši

Brisanje podataka se izvršava tako što zaposleni radnik selektuje određeni red i odabere opciju obriši. Kao i kod ostalih opcija aktivira se metoda ali ovaj put „Obriši“.

```
static void ObrisiZapis(DataGrid grid, string deleteUpit)
        {
            try
            {
                konekcija.Open();
                DataRowView red =
(DataRowView)grid.SelectedItems[0];
                string upit = deleteUpit + red["ID"];
                MessageBoxResult rezultat = MessageBox.Show("Da li
ste sigurni?", "Upozorenje!", MessageBoxButton.YesNo,
MessageBoxImage.Question);
                if (rezultat == MessageBoxResult.Yes)
                {
                    SqlCommand cmd = new SqlCommand(upit,
konekcija);
                    cmd.ExecuteNonQuery();
                }


            }
            catch(ArgumentOutOfRangeException)
            {
                MessageBox.Show("Niste selektovali red!", "Greška",
               MessageBoxButton.OK, MessageBoxImage.Error);
            }
            catch (SqlException)
            {
                MessageBox.Show("Postoje povezani podaci u drugim tabelama!Molimo Vas
da ih provjerite!", "Greška",
                MessageBoxButton.OK, MessageBoxImage.Error);
            }
            finally
            {
                if (konekcija != null)
                {
                    konekcija.Close();
                }
            }
        }
```

Kao što se može primjetiti u metodi potrebno je koristiti select upite za brisanje koji se uz pomoć metode obriši posleđuju u bazu podataka.

```
#region Delete Upiti
        string deleteClanstvo = @"delete from tblClanstvo where ClanstvoID=";
        string deleteDolazak = @"delete from tblDolazak where DolazakID=";
        string deleteKorisnik = @"delete from tblKorisnik where KorisnikID=";
        string deleteOprema = @"delete from tblOprema where OpremaID=";
        string deleteProgram = @"delete from tblProgram where ProgramID=";
        string deleteRegistracija = @"delete from tblRegistracija where
RegistracijaID=";
        string deleteVrstaClanstva = @"delete from tblVrstaClanstva where
VrstaClanstvaID=";
        string deleteZaposleni = @"delete from tblZaposleni where ZaposleniID=";
```

Naravno, kao i kod ostale dve opcije potrebna je provjera koja je tabela učitana da bi se mogla uopšte pozvati metoda obriši:

```
private void btnObrisi_Click(object sender, RoutedEventArgs e)
        {
            if (ucitanaTabela.Equals(clanstvoSelect))
            {
                ObrisiZapis(DataGridCentralni, deleteClanstvo);
                UcitajPodatke(DataGridCentralni, clanstvoSelect);

            }else if (ucitanaTabela.Equals(dolazakSelect))
            {
                ObrisiZapis(DataGridCentralni, deleteDolazak);
                UcitajPodatke(DataGridCentralni, dolazakSelect);

            }else if (ucitanaTabela.Equals(korisnikSelect))
            {
                ObrisiZapis(DataGridCentralni, deleteKorisnik);
                UcitajPodatke(DataGridCentralni, korisnikSelect

           }

     }
```

6.4 Izgled aplikacije:

Aplikacija se sastoji iz jedanaest normi od kojih su osam formi dodaj i izmeni, a ostale dvije su MainWindow i Log In forma.

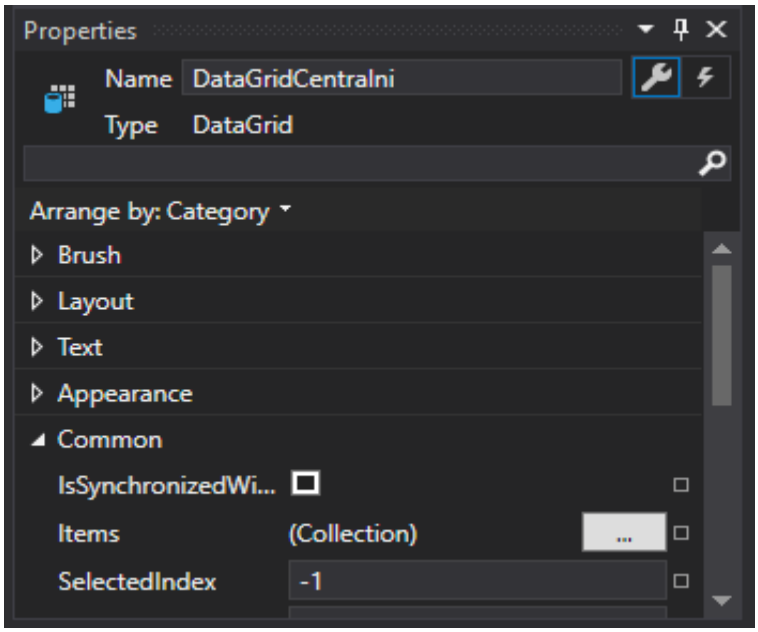
Konkretan dizajn osam formi je kreiran pomoću „Material Design“ toolkita koji se implementuje instaliranjem u Visual Studiju i dodavanjem nekoliko linija koda u App.xaml.

```
<Application.Resources>
        <ResourceDictionary>
            <ResourceDictionary.MergedDictionaries>
                <ResourceDictionary
Source="pack://application:,,,/MaterialDesignThemes.Wpf;component/The
mes/MaterialDesignTheme.Light.xaml" />
                <ResourceDictionary
Source="pack://application:,,,/MaterialDesignThemes.Wpf;component/The
mes/MaterialDesignTheme.Defaults.xaml" />
                <ResourceDictionary
Source="pack://application:,,,/MaterialDesignColors;component/Themes/
Recommended/Primary/MaterialDesignColor.Purple.xaml" />
                <ResourceDictionary
Source="pack://application:,,,/MaterialDesignColors;component/Themes/
Recommended/Accent/MaterialDesignColor.Lime.xaml" />
            </ResourceDictionary.MergedDictionaries>
        </ResourceDictionary>
    </Application.Resources>
```

Dodaje se još i linija koda za implementaciju u XAML svih formi:

```
xmlns:materialDesign="http://materialdesigninxaml.net/winfx/xaml/the
mes"
```

Za konkretne boje i font korištene su opcije koje se nalaze u sklopu Microsoft Visual Studio razvojnog okruženja:



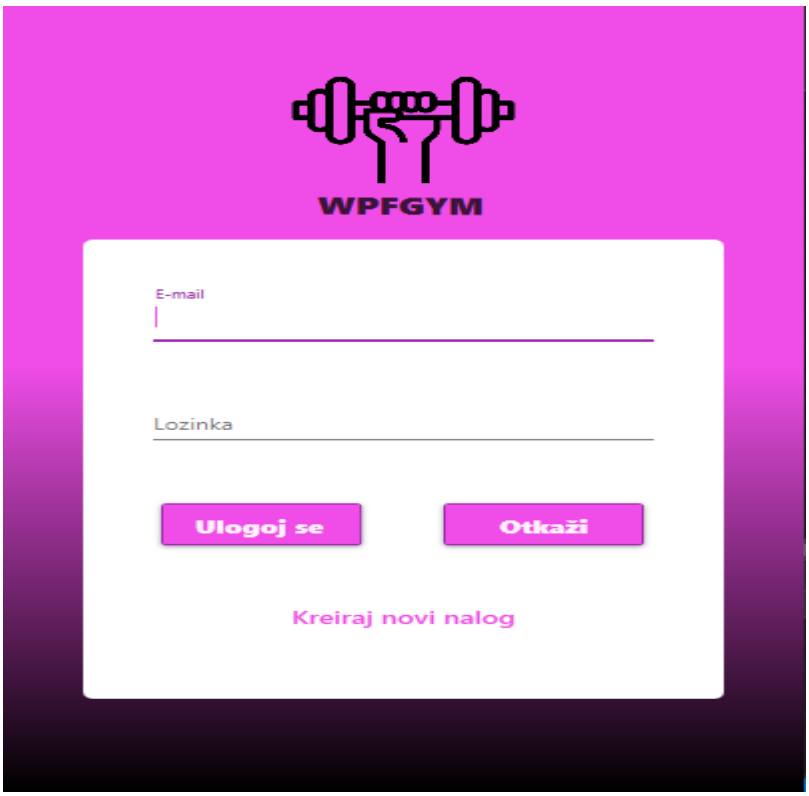
Što se tiče dugmadi u MainWinodw-u za učitavanje u DataGrid korištena je kombinacija slika i textbox-ova:

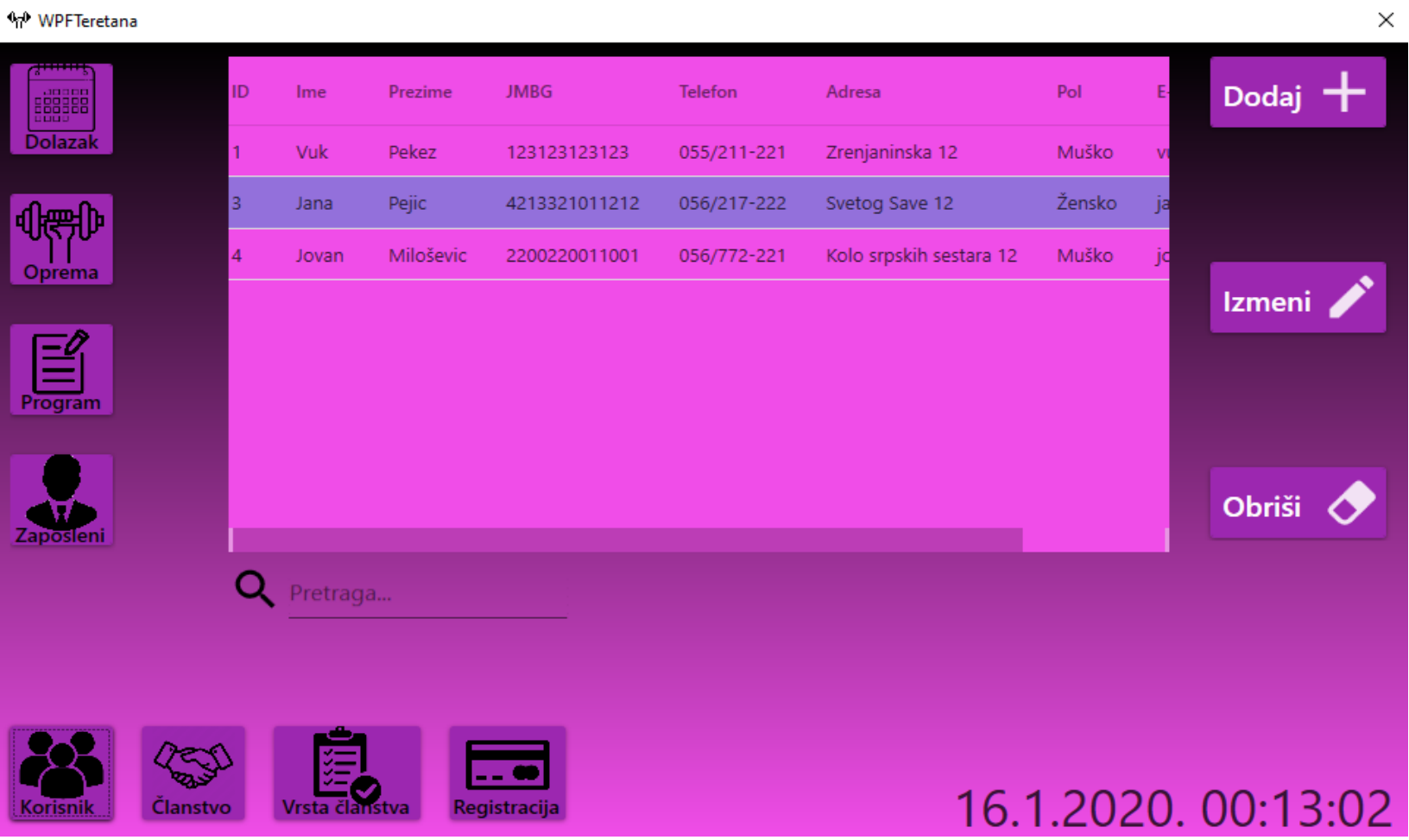
```
<Button x:Name="btnDolazak" Margin="0 15 0 13"
HorizontalAlignment="Left"  Width="70" Height="64"
Click="btnDolazak_Click" Background="#FF9C27B0">
                <Button.Content>
                    <Grid>
                        <Image Source="img/calendar.png" Margin="-
24,-12,-23,5" RenderTransformOrigin="0.444,0.525"/>
                        <TextBlock Text="Dolazak" Margin="-11,41,-
11,-2" Foreground="Black"></TextBlock>
                    </Grid>
                </Button.Content>
            </Button>
```

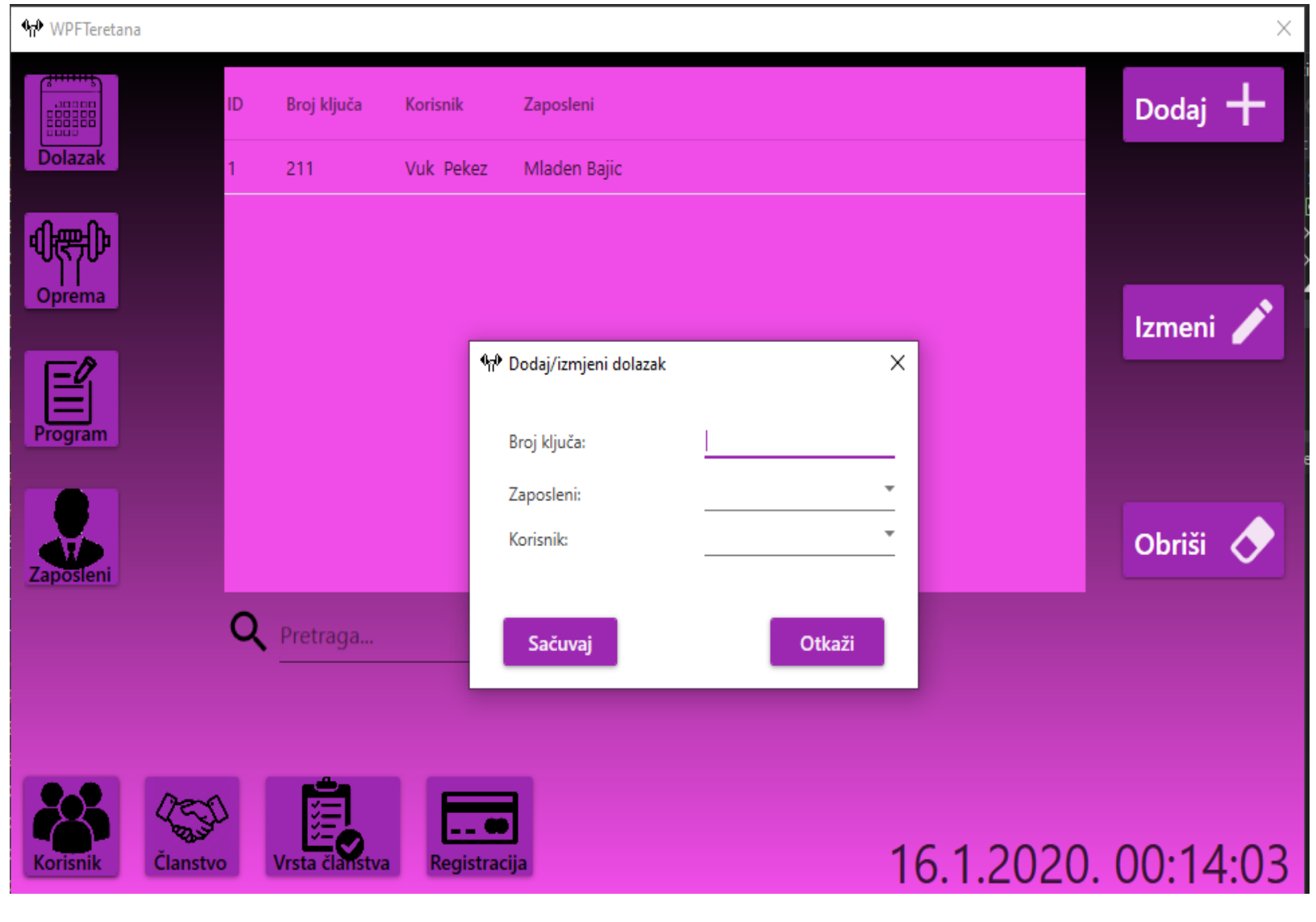
Dok su dugmad namjenjena glavnim opcijama: Dodaj, Izmjeni, Ažuriraj kreirana uz pomoć ikonica iz MaterialDesign toolkita i texboxova:

```
<Button x:Name="btnDodaj" Margin="0 0 0 40"   Width="120" Height="50"
Click="btnDodaj_Click" >
                <Button.Content>
                    <Grid Width="23">
                        <materialDesign:PackIcon Kind="Plus" Margin="18,-5,-45,-5"
Height="49" Width="50"/>
                        <TextBlock Text="Dodaj" Margin="-40,8,5,5"
Foreground="White" FontSize="20"/>
                    </Grid>
                </Button.Content>
            </Button>
```

Kombinacija MaterialDesing teme i dostupnih opcija dizajniranja u Visual Studio razvojnom okruženju omogućila je da konkretna aplikacija „WPFGYM“ izgleda:




WPFTeretana
Dolazak
Oprema
Program
Zaposleni
ID Ime Prezime JMBG Telefon Adresa Pol E
1 Vuk Pekez 123123123123 055/211-221 Zrenjaninska 12 Muško
3 Jana Pejic 4213321011212 056/217-222 Svetog Save 12 Žensko
4 Jovan Miloševic 2200220011001 056/772-221 Kolo srpskih sestara 12 Muško
Dodaj
Izmeni
Obriši
Pretraga...
Korisnik
Članstvo
Vrsta članstva
Registracija
16.1.2020. 00:13:02

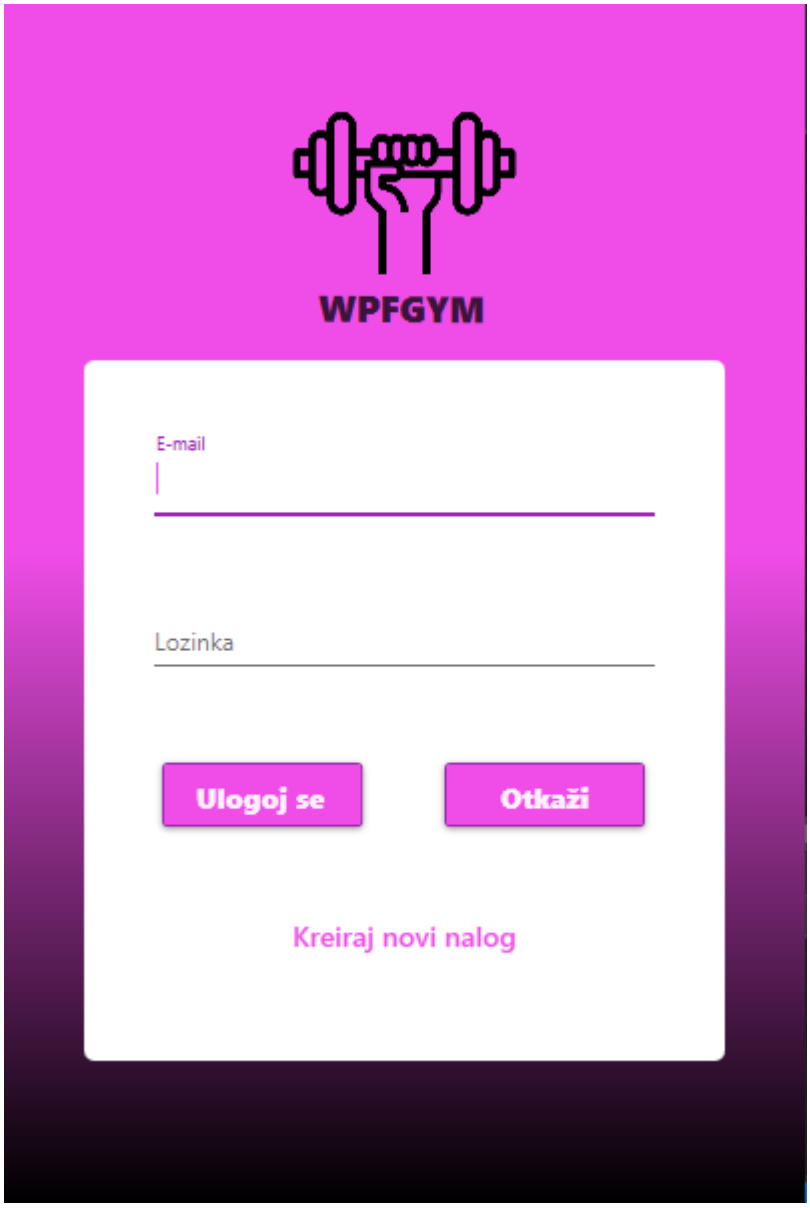

WPFTeretana
Dolazak
Oprema
Program
Zaposleni
ID Broj ključa Korisnik Zaposleni
1 211 Vuk Pekez Mladen Bajic
Dodaj
Izmeni
Obriši
Pretraga...
Dodaj/izmjeni dolazak
Broj ključa:
Zaposleni:
Korisnik:
Sačuvaj
Otkaži
Korisnik
Članstvo
Vrsta članstva
Registracija
16.1.2020. 00:14:03

Primjer XAML koda za frmČlanstvo:

```
<Label Content="Datum početka članarine:" HorizontalAlignment="Left"
Margin="25,27,0,0" VerticalAlignment="Top"/>
        <Label Content="Datum isteka članarine:"
HorizontalAlignment="Left" Margin="25,56,0,0"
VerticalAlignment="Top"/>
        <Label Content="Korisnik:" HorizontalAlignment="Left"
Margin="25,89,0,0" VerticalAlignment="Top"/>
        <Label Content="Vrsta članstva:" HorizontalAlignment="Left"
Margin="25,119,0,0" VerticalAlignment="Top"/>
        <DatePicker x:Name="dpDatumPocetka"
HorizontalAlignment="Left" Margin="175,25,0,0"
VerticalAlignment="Top" Width="142"/>
        <DatePicker x:Name="dpDatumIsteka"
HorizontalAlignment="Left" Margin="175,55,0,0"
VerticalAlignment="Top" Width="142"/>
        <ComboBox x:Name="cbKorisnik" HorizontalAlignment="Left"
Margin="175,85,0,0" VerticalAlignment="Top" Width="142"
DisplayMemberPath="Korisnik" SelectedValuePath="KorisnikID"/>
        <ComboBox x:Name="cbVrstaClanstva"
HorizontalAlignment="Left" Margin="175,115,0,0"
VerticalAlignment="Top" Width="142"
DisplayMemberPath="NazivVrsteClanstva"
SelectedValuePath="VrstaClanstvaID"/>
        <Button x:Name="btnSacuvaj" Content="Sačuvaj"
HorizontalAlignment="Left" Margin="25,169,0,0"
VerticalAlignment="Top" Width="85" Click="btnSacuvaj_Click"/>
        <Button x:Name="btnOtkazi" Content="Otkaži"
HorizontalAlignment="Right" Margin="0,169,25,0"
VerticalAlignment="Top" Width="85" Click="btnOtkazi_Click"/>
```

Nakon kreiranje osnovnih funkcionalnosi konketne WPF aplikacije i dizajniranja iste, bilo je vrijeme da se aplikacija podigne na viši nivo dodavanjem dodatnih funkcija. Konkretne dodatne funkcije implementirane u ovaj projekat ondose se na „log in“ ali i na pretragu u sklopu DataGrid-a.

1.Log in funckija je kreirana uz pomoć više metoda koje provjeravju dostupnost mail-a i lozinke u bezi. Može se reći da je log-in jako intuitivaj jer korisniku ove aplikacije ukazuje na konkretnu grešku u logovanju.



Metoda za provjeru postojanja E-mail adrese:

```
private bool postojiEmailZaposlenog(string EmailZaposlenog)

        {
            try
            {
                konekcija.Open();
                string upitKorisnickoIme = @"select count(*) from tblZaposleni where
EmailZaposlenog='" + txtLogMail.Text + "'";

                SqlCommand cmdKorisnickoIme = new SqlCommand(upitKorisnickoIme,
konekcija);

                int rezultatUpita =
Convert.ToInt32(cmdKorisnickoIme.ExecuteScalar());

                if (rezultatUpita > 0)
                {
                    return true;
                }
                else
                {
                    return false;
                }
            }
            catch (Exception e)
            {
                MessageBox.Show(e.Message, "Greška!");
                return false;
            }
            finally
            {
                if (konekcija != null)
                {
                    konekcija.Close();
                }
            }
```

Metoda za provjeru ispravnosti lozinke:

```
private bool ispravnaLozinka(string EmailZaposlenog, string
unetaLozinka)

        {
            try
            {
                konekcija.Open();
                string upitLozinka = @"select Lozinka from
tblZaposleni where EmailZaposlenog='" + txtLogMail.Text + "'";
                SqlCommand cmd = new SqlCommand(upitLozinka,
konekcija);
                string lozinkaIzBaze =
cmd.ExecuteScalar().ToString();

                if (String.Equals(lozinkaIzBaze, unetaLozinka))
                {
                    return true;
                }
                else
                {
                    return false;
                }
            }
            catch (Exception)
            {
                return false;
            }
            finally
            {
                if (konekcija != null)
                {
                    konekcija.Close();
                }
            }
        }
```

Event koji se dešava pri kliku na dugme „Uloguj se“.

```
private void btnUlogujSe_Click(object sender, RoutedEventArgs e)
        {
            try
            {


                if (string.IsNullOrEmpty(txtLogMail.Text))
                {
                    MessageBox.Show("E-mail nije unet!",
                                    "Greška", MessageBoxButton.OK, MessageBoxImage.Error);
                    txtLogMail.Focus();
                    return;
                }
                if (string.IsNullOrEmpty(txtLogPass.Password))
                {
                    MessageBox.Show("Lozinka nije uneta!",
                                    "Greška", MessageBoxButton.OK, MessageBoxImage.Error);
                    txtLogPass.Focus();
                    return;
                }


                string unetiEmail = txtLogMail.Text;
                string unetaLozinka = txtLogPass.Password;

                if (postojiEmailZaposlenog(unetiEmail) == false)
                {
                    MessageBox.Show("Uneti E-mail ne postoji u bazi podataka! Dodajte novi nalog! ",
                                    "Greška", MessageBoxButton.OK, MessageBoxImage.Error);
                    txtLogMail.Focus();
                    return;
                }

                if (ispravnaLozinka(unetiEmail, unetaLozinka) == false)
                {
                    MessageBox.Show("Lozinka nije ispravna! Pokušajte ponovo!",
                    "Greška", MessageBoxButton.OK, MessageBoxImage.Error);
                    txtLogPass.Focus();
                    return;
                }


                MainWindow main = new MainWindow();
                main.Show();
                this.Close();

            }
            catch (Exception)
```

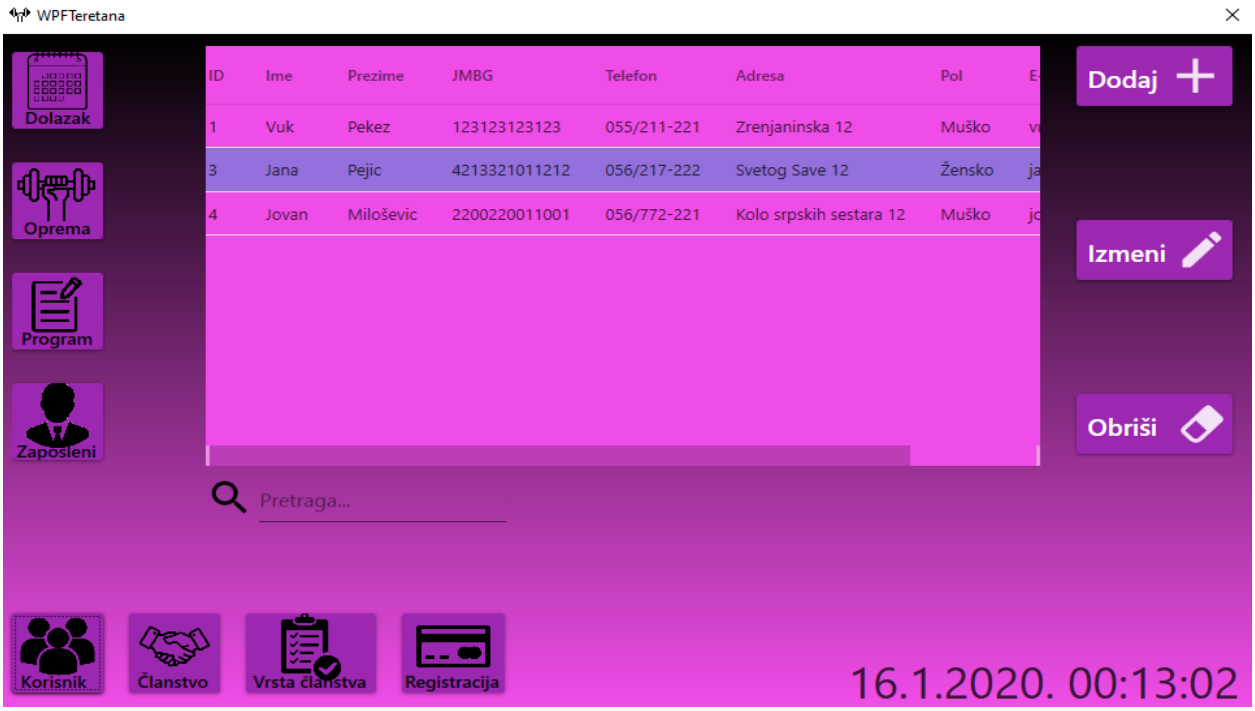
Logigika iza dugmadi „Otkaži“ i „Novi nalog“

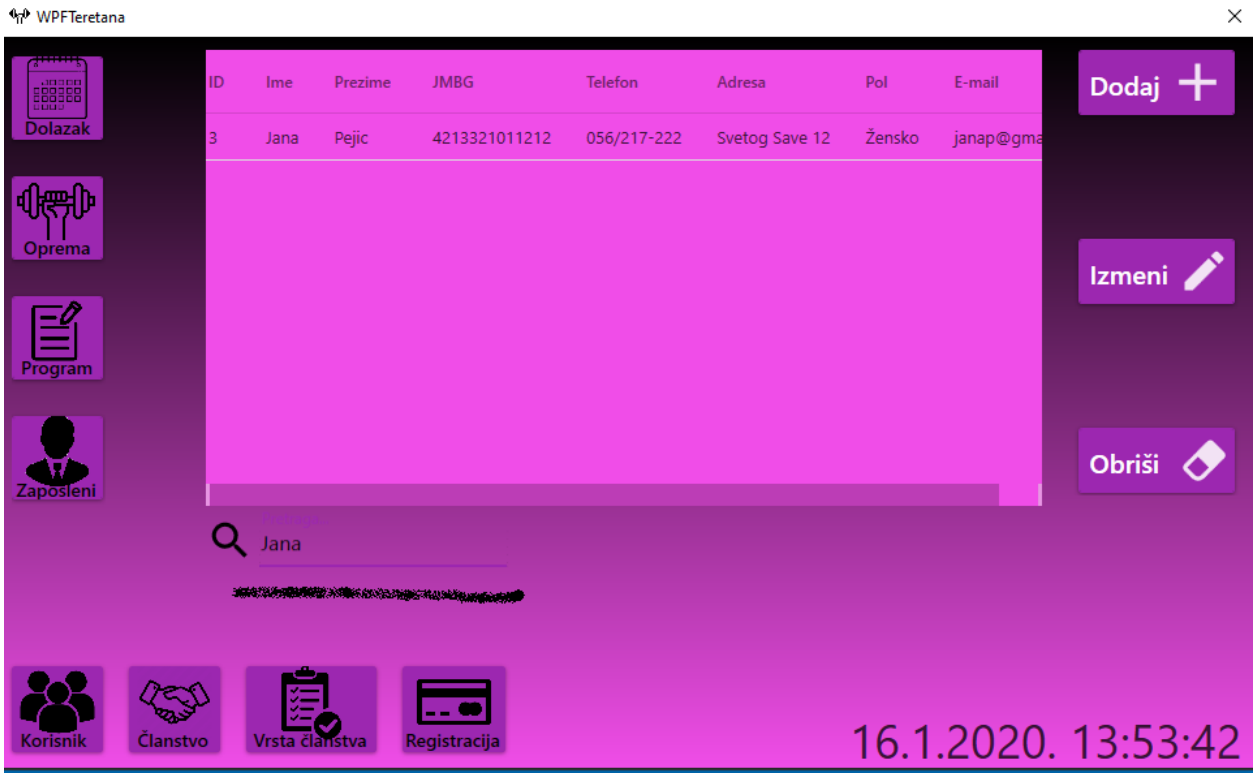
```
private void btnOtkazi_Click(object sender, RoutedEventArgs e)
        {
            this.Close();
        }


        private void btnNoviNalog_Click(object sender, RoutedEventArgs
e)
        {

            Window prozor = new frmZaposleni();
            prozor.ShowDialog();


        }
```

Kao što se da primjetiti logika iza prozora Log In je poprilično jednostavna, ali pruža potpuno jasno iskustvo u radu aplikacije. Dve glavne metode su validacionog tipa što je najbitnije kada se radi na funkciji tipa Log In. Pored logike bitno je obezbjediti i dobar dizajn aplikacije, a to se radi u dizajnerskom dijelu koda, tacnije XAML-u same forme „LogIn“:

```
<Window x:Class="WpfTeretana.Forme.frmLogIn"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:WpfTeretana.Forme"
        xmlns:materialDesign="http://materialdesigninxaml.net/winfx/xaml/themes"
        mc:Ignorable="d"
        Title="WPFGYM" Height="600" Width="400" ResizeMode="NoResize"
WindowStartupLocation="CenterScreen" WindowStyle="None" Foreground="Black"  >
    <Window.Background>
        <LinearGradientBrush EndPoint="0.5,1" MappingMode="RelativeToBoundingBox"
StartPoint="0.5,0">
            <GradientStop Color="Black" Offset="1"/>
            <GradientStop Color="#FFF04DE8" Offset="0.457"/>
        </LinearGradientBrush>
    </Window.Background>
    <Grid>
        <StackPanel Margin="30,40,30,40">
            <Image Source="C:\Users\Korisnik\Desktop\logo2.png" Height="111" Margin="90,0,90,-14"/>
            <Label Content="WPFGYM" HorizontalAlignment="Center"  VerticalAlignment="Center"
FontFamily="Segoe UI Black"  Width="94" Margin="100,0,100,0" RenderTransformOrigin="0.475,0.361"
FontSize="18">
                <Label.RenderTransform>
                    <TransformGroup>
                        <ScaleTransform/>
                        <SkewTransform AngleY="0.406"/>
                        <RotateTransform/>
                        <TranslateTransform Y="-0.343"/>
                    </TransformGroup>
                </Label.RenderTransform>
            </Label>
            <Border Background="White" Margin="10 10" CornerRadius="5" Height="350">
                <StackPanel Margin="20">
                    <TextBox x:Name="txtLogMail"  Margin="15" Style="{StaticResource
MaterialDesignFloatingHintTextBox}" materialDesign:HintAssist.Hint="E-mail"  Height="43"
CaretBrush="#FFF04DE8" Foreground="Black"/>
                    <PasswordBox x:Name="txtLogPass" Margin="15"  Style="{StaticResource
MaterialDesignFloatingHintPasswordBox}" materialDesign:HintAssist.Hint="Lozinka" Height="45"
CaretBrush="#FFF04DE8" Foreground="Black" SelectionBrush="#FFF04DE8" />
                    <StackPanel Orientation="Horizontal" HorizontalAlignment="Center" Margin="12,0"
Height="100">
                        <Button x:Name="btnUlogujSe" Margin="20,30" Content="Ulogoj se "
Background="#FFF04DE8" FontFamily="Segoe UI Black" Click="btnUlogujSe_Click" />
                        <Button x:Name="btnOtkazi" Margin="20,30" Content="Otkaži"
Background="#FFF04DE8" FontFamily="Segoe UI Black" Width="100" Click="btnOtkazi_Click"/>
                    </StackPanel>
                    <Button x:Name="btnNoviNalog" Margin="20,5" Content="Kreiraj novi nalog"
Height="28" Background="{x:Null}" FontSize="14" Foreground="#FFF04DE8" BorderBrush="{x:Null}"
Click="btnNoviNalog_Click" />
                </StackPanel>
            </Border>

        </StackPanel>



    </Grid>
</Window>
```

Druga dodatna funckija implementirana u konkretnu aplikaciju je pretraga odnosno „search“. Konkretna pretraga je kreirana tako da pretražuje tabelu koja je trenutno učitana na DataGridu. Nakon provjere ucitane tabele u tektualnom polju pretraga se očiatava unjeti text i pretražuju se svi pojmovi u kokretnoj tabeli.





Logika iza tekstualne pretrage izgleda ovako:

```
private void txtPretraga_TextChanged(object sender, TextChangedEventArgs e)
        {
            try
            {
                if (ucitanaTabela.Equals(korisnikSelect))
                {

                    upit = ucitanaTabela + "Where ImeKorisnika like " + "'%" + txtPretraga.Text + "%'"
                         + " or PrezimeKorisnika like " + "'%"+ txtPretraga.Text + "%'"
                         + " or JMBGKorisnika like " + "'%" + txtPretraga.Text + "%'"
                         + " or TelefonKorisnika like " + "'%" + txtPretraga.Text + "%'"
                         + " or AdresaKorisnika like " + "'%" + txtPretraga.Text + "%'"
                         + " or PolKorisnika like " + "'%" + txtPretraga.Text + "%'"
                         + " or EmailKorisnika like " + "'%" + txtPretraga.Text + "%'";

                }
                else if (ucitanaTabela.Equals(clanstvoSelect))
                {

                    upit = ucitanaTabela + "Where DatumPocetka like " + "'%" + txtPretraga.Text + "%'"
                     + " or DatumIsteka like " + "'%" + txtPretraga.Text + "%'"
                     +" or tblKorisnik.ImeKorisnika like " + "'%" + txtPretraga.Text + "%'"
                     + " or tblKorisnik.PrezimeKorisnika like " + "'%" + txtPretraga.Text + "%'"
                    + " or tblVrstaClanstva.NazivVrsteClanstva like " + "'%" + txtPretraga.Text + "%'";

                }
                else if (ucitanaTabela.Equals(dolazakSelect))
                {
                    upit = ucitanaTabela + "Where BrojKljuca like " + "'%" + txtPretraga.Text + "%'"
                    + " or tblZaposleni.ImeZaposlenog like " + "'%" + txtPretraga.Text + "%'"
                    + " or tblZaposleni.PrezimeZaposlenog like " + "'%" + txtPretraga.Text + "%'"
                    + " or tblKorisnik.ImeKorisnika like " + "'%" + txtPretraga.Text + "%'"
                   + " or   tblKorisnik.PrezimeKorisnika like " + "'%" + txtPretraga.Text + "%'";

                }
                else if (ucitanaTabela.Equals(opremaSelect))
                {

                    upit = ucitanaTabela + "Where CijenaOpreme like " + "'%" + txtPretraga.Text + "%'"
                   + " or tblZaposleni.ImeZaposlenog like " + "'%" + txtPretraga.Text + "%'"
                   + " or tblZaposleni.PrezimeZaposlenog like " + "'%" + txtPretraga.Text + "%'"
                   + " or NazivOpreme like " + "'%" + txtPretraga.Text + "%'"
                  + " or  VrstaOpreme like " + "'%" + txtPretraga.Text + "%'";
                }
               else if (ucitanaTabela.Equals(programSelect))
                {
                    upit = ucitanaTabela + "Where VrstaPrograma like " + "'%" + txtPretraga.Text + "%'"
                  + " or tblZaposleni.ImeZaposlenog like " + "'%" + txtPretraga.Text + "%'"
                  + " or tblZaposleni.PrezimeZaposlenog like " + "'%" + txtPretraga.Text + "%'"
                  + " or tblKorisnik.ImeKorisnika like " + "'%" + txtPretraga.Text + "%'"
                 + " or   tblKorisnik.PrezimeKorisnika like " + "'%" + txtPretraga.Text + "%'"
                 + " or   CijenaPrograma like " + "'%" + txtPretraga.Text + "%'";

                }
```

Ukoliko korisnik obriše sve iz teksualne pretrage, ponovo se učitava cijela tabela:

```
if (txtPretraga.Text == "")
                {
                    UcitajPodatke(DataGridCentralni, ucitanaTabela);
                }
                else
                {


                    konekcija.Open();
                    SqlDataAdapter adapter = new SqlDataAdapter(upit, konekcija);
                    DataTable tabela = new DataTable();
                    adapter.Fill(tabela);
                    DataGridCentralni.ItemsSource = tabela.DefaultView;
                }
            }
            catch (SqlException ex)
            {

                MessageBox.Show(ex.ToString());
            }
            finally
            {
                if (konekcija != null)
                {
                    konekcija.Close();
                }
            }
        }
```

XAML za konkretan TextBox izgleda ovako:

```
<TextBox x:Name="txtPretraga"  Margin="201,361,573,154" Style="{StaticResource
MaterialDesignFloatingHintTextBox}" materialDesign:HintAssist.Hint="Pretraga..."
CaretBrush="#FFF04DE8" Foreground="Black" RenderTransformOrigin="0.5,0.5"
TextChanged="txtPretraga_TextChanged" Background="Transparent" FontSize="16">
```

7.TESTIRANJE APLIKACIJE

Testiranje funkcionalnosti aplikacije je sprovedeno u više navrata pokušavajući da se stvori situacija u kojoj će aplikacija da reaguje neočekivano. Kroz veliki broj pokušaja ustanovljeno je da je aplikacija bezbjedna za korištenje. Sve metode su grupisane u Try-Cacth blokove da bi se obezbjedilo da ukoliko dođe do greške, korisnik bude obavješten a program nastavi normalno sa radom. Aplikacija je testirana od strane 3 lica da bi se izbjegla rutina u testiranju. Obzirom da je program administrativne prirode njegova neprekidnost u radu je obavezna i to smo obezbjedili upotrebom try-cacth blokova kao i optimalnim kodom koji je pisan. Primjere try-cacth metoda možete pogledati ispod.

Tokom rada u aplikaciji, korsink iste će biti obavješten ukoliko načini pogrešan korak, unese pogrešan podatak ili pritisne pogrešno dugme. Korisnik se obavještava kroz „Message.box“ kokretnim porukama koje ga upoćuju na konkretan put kojim treba da radi u aplikaciji. U trenutnoj verziji projekta validacija nije podržana, ali se u narednom periodu očekuje i ta funkcija, kao i dizajnerski rad na samom obavještenjima korisniku aplikacije.

```
try
            {
                konekcija.Open();
                if (MainWindow.azuriraj)
                {
                    DataRowView red = MainWindow.pomocniRed;

                    string update=@"update tblOprema set
CijenaOpreme='"+txtCijenaOpreme.Text+"',VrstaOpreme='"+txtVrstaOpreme.Text+"'," +
                        "NazivOpreme='"+txtNazivOpreme.Text+"',ZaposleniID='"+cbZaposleni.SelectedValue+"'
where OpremaID=" + red["ID"];
                    SqlCommand cmd = new SqlCommand(update, konekcija);
                    cmd.ExecuteNonQuery();
                    MainWindow.pomocniRed = null;

                }
                else {
                    string insert = @" insert into tblOprema(CijenaOpreme,VrstaOpreme,NazivOpreme,ZaposleniID)
                                values('" + txtCijenaOpreme.Text + "','" + txtVrstaOpreme.Text + "','"
                                    + txtNazivOpreme.Text + "','" + cbZaposleni.SelectedValue + "')";
                    SqlCommand cmd = new SqlCommand(insert, konekcija);
                    cmd.ExecuteNonQuery();
                    }
                this.Close();

            }
            catch (Exception)
            {
                MessageBox.Show("Unos određenih podataka nije validan! Molimo vas pokušajte ponovo",
                   "Greška", MessageBoxButton.OK, MessageBoxImage.Error);
            }
            finally
            {
                if (konekcija != null)
                {
                    konekcija.Close();
                }
            }
        }
```

Primjer se odnosi na dugme „Sačuvaj“ u formi „Oprema“. Kao što je ovde primjenjen, try-cacth se primjenjuje i u svim ostalim metodama kako u formama tako i u MainWindow-u.

8.ZAKLJUČAK

Nakon poslednjih provjera aplikacije, ustanovljeno je da je aplikacija potpuno osposobljena za rad u realnom sistemu. Iako su urađene osnovne i neke dodatne funkcije postoji veoma veliki prostor za unapređivanje ove male aplikacije. Može se reći da je ova aplikacija inicijalna, konkretno može poslužiti, ali za pravi, razvijeni realni sistem potrebno ju je još mnogo nadograditi. Obzirom na varijacije teratne kao realnog sistema potrebno je aplikaciju prilagoditi istom. Iako se na prvi pogled čini intuitivnom, trebalo bi ipak korisnika uputiti u njen rad, jer iako je mala i jednostavna opet je uže specijalizovana za konkretan sistem. Pored validacije i restrikcija u unosu podatka koje bi trebale da se ugrade u ovu aplikaciju, postoji raskoš dodatnih opcija koje se mogu ubaciti u konkretnu aplikaciju i koje bi život korisnika učinile lakšim. Međutim smatram da je aplikacija sasvim korektna za studentski projekat i za vrijeme koje je u nju uloženo. Kod konkretne aplikacije se posebno ističe dizajn koji je neobičan ali jako intuitivan za korisnika i prija oku. Sve u svemu, kreirana je jedna mala, ne previše složena ali u svakom slučaju fukcionalna i korisna aplikacija. Konstantim radom na ovoj aplikaciji u narednom periodu nadamo se da će jednog dana završiti u nekom realnom sistemu i stremimo ka tome.

8.1.Lični osjećaj u radu:

Kroz čitav proces izrade ovog projekta konstantno sam učio nove stvari. Rad na konkretnom projketu je bio zahtjevan ali svakako poučan. Iako sam najviše uživao u kreiranju same aplikacije u programskom jeziku C#, smatram da će mi znanje stečeno u kreiranju UML dijagrama, modelovanju procesa i radu sa bazama podatka u mnogome od pomoći. Smatram da smo od kreiranja UML dijagrama pa do završetka aplikacije napravili pun krug i da će mi znanje stečeno u radu na konkretnom projektu u mnogome doprinjeti u kasnijem radu i studiranju. Naravno i nakon predaje ovog projekata, nastaviću da radim na njemu i ko zna, možda se jednog dana nađe u nekoj teretani. Na kraju ovog projekta mogu reći da sam uživao u njemu u cjelini, sa izostankom uživanja kada je riječ o dokumentaciji, ali i to je važan dio ovog projekta. Kroz ovaj projekat shvatio sam da se radujem budućim projektima i znam da ću u njih uložiti isto truda, ako ne i više kao što sam uložio u ovaj projekat. Hvala.

Mladen Bajić(IT46/2018)

mladen_ba@live.com