Nova Tehnika

Projekat iz predmeta „Microsoft tehnologije za pristup podacima”

Nikola V.

Beograd, 2018.

Ovaj dokument predstavlja sveobuhvatan prolaz kroz implementaciju rešenja prvog projekta iz predmeta ,,Microsoft tehnologije za pristup podacima” Računarskog fakulteta u Beogradu.

S A D R Ž A J

Strana

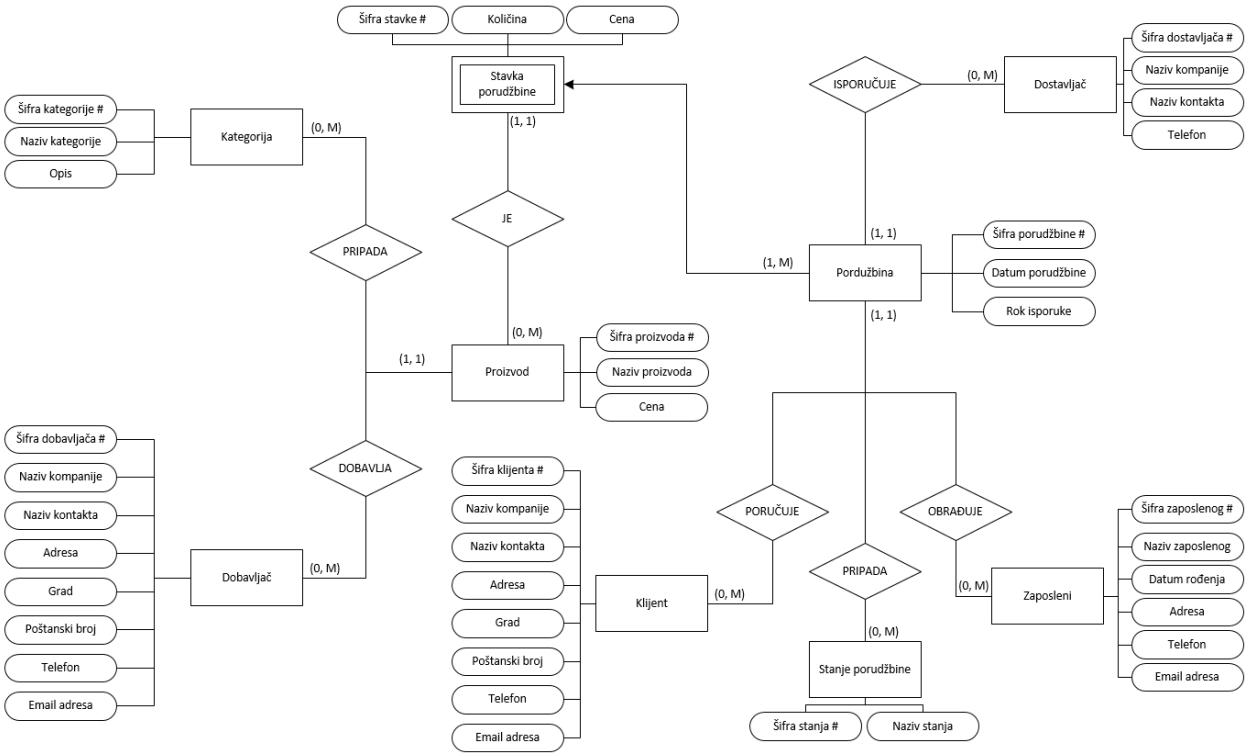
1. Specifikacije projekta

1. Potrebno je dati **verbalni opis problema** (u oko 7-8 rečenica) za koji se razvija deo Informacionog sistema.
2. Na osnovu verbalnog opisa sistema razviti **konceptualni model** (oko 7-8 objekata, tj. klasa). Konceptualni model dati kroz: model-objekti veze (MOV (eng. ER)) ili UML dijagram klasa.
3. Konceptualni model prevesti u **relacioni model podataka**.
4. **Isprojektovati bazu u SQL Serveru** na osnovu relacionog modela putem DDL komandi, a putem DML komandi ubaciti neke testne podatke.
5. Odabrati **jednu tehniku denormalizacije** i napraviti denormalizovanu tabelu u postojećoj bazi, zatim za denormalizovanu **tabelu dati i opisati tabelu specifikacije trigera** i **izvršiti implementaciju trigera** putem kojih se kontroliše integritet podataka.
6. Upotrebom ADO.NET-a i modela **konektovane arhitekture** izvršiti nad 3 povezane tabele CRUD operacije, a **od te 3 tabele neka 2 tabele budu u kontekstu MOV-a jak-slab objekat**, tj. dokument i stavka dokumenta (npr. račun i stavka računa, narudžbenica i stavka narubžbenice – prilikom unosa / izmene / brisanja stavke dokumenta, voditi računa o izmeni iznosa na zaglavlju dokumenta)
7. Upotrebom ADO.NET-a izvršiti **USKLADIŠTENE PROCEDURE** koje rade CRUD operacije nad 1 tabelom, a koja je različita od tabela iz zahteva 6.
8. Upotrebom ADO.NET-a izvršiti **CRUD operacije nad tabelom koja je denormalizovana u zahtevu 5 i iskontrolisati izvršenje trigera**.
9. Upotrebom ADO.NET-a i modela **diskonektovane arhitekture** izvršiti nad 2 tabele CRUD operacije.
10. **Za projekat je moguće odabrati aplikacije: Windows Forms, WPF Forms i Web Forms (dizajn se ne ocenjuje, već samo pobrojane funkcionalnosti u radu sa bazom)**

2. Verbalni opis rešenja

Nova Tehnika je kompanija koja predstavlja lanac snabdevanja maloprodajnih radnji širokim asortimanom računarske i druge opreme. U njenom informacionom sistemu ključno je obuhvatiti međukompanijsku integraciju sa poštanskom tj. **dostavljačkom kompanijom** za realizaciju distribucije **naručenih proizvoda**, usled činjenice da su kompanije uvek pod pritiskom da nadvladaju konkurente na tržištu pružanjem većeg kvaliteta **proizvoda** po povoljnijoj ceni i obezbeđivanjem dostave proizvoda u zahtevanom obimu i u zahtevano vreme za svoje **klijente**. Zbog gorespomenutih razloga takođe je neophodno obezbediti međukompanijsku integraciju sa **dobavljačima proizvoda**. Proizvode je neophodno grupisati po **kategorijama**, a porudžbine moraju biti u nekom **stanju**. Osim evidencije procesa snabdevanja i isporuke proizvoda, informacioni sistem mora da reši problem **upravljanja kadrovima (zaposlenima)** - potrebno je voditi evidenciju za svakog zaposlenog, gde će svaki zaposleni imati jedinstveni identifikacioni broj.

3. Konceptualni model



4. Relacioni model

Dobavljač(Šifra dobavljača, Naziv kompanije, Naziv kontakta, Adresa, Grad, Poštanski broj, Telefon, Email adresa)
Kategorija(Šifra kategorije, Naziv kategorije, Opis)
Proizvod(Šifra proizvoda, Naziv proizvoda, Cena, *Šifra kategorije*, *Šifra dobavljača*)
Klijent(Šifra klijenta, Naziv kompanije, Naziv kontakta, Adresa, Grad, Poštanski broj, Telefon, Email adresa)
Zaposleni(Šifra zaposlenog, Naziv zaposlenog, Datum rođenja, Adresa, Telefon, Email adresa)
Stanje porudžbine(Šifra stanja, Naziv stanja)
Dostavljač(Šifra dostavljača, Naziv kompanije, Naziv kontakta, Telefon)
Porudžbina(Šifra porudžbine, Datum porudžbine, Rok isporuke, *Šifra klijenta*, *Šifra stanja*, *Šifra zaposlenog*, *Šifra dostavljača*)
Stavka porudžbine(Šifra stavke, Šifra porudžbine, Količina, Cena, *Šifra proizvoda*)

5. Implementacija baze podataka u SQL serveru

5.1. DDL naredbe

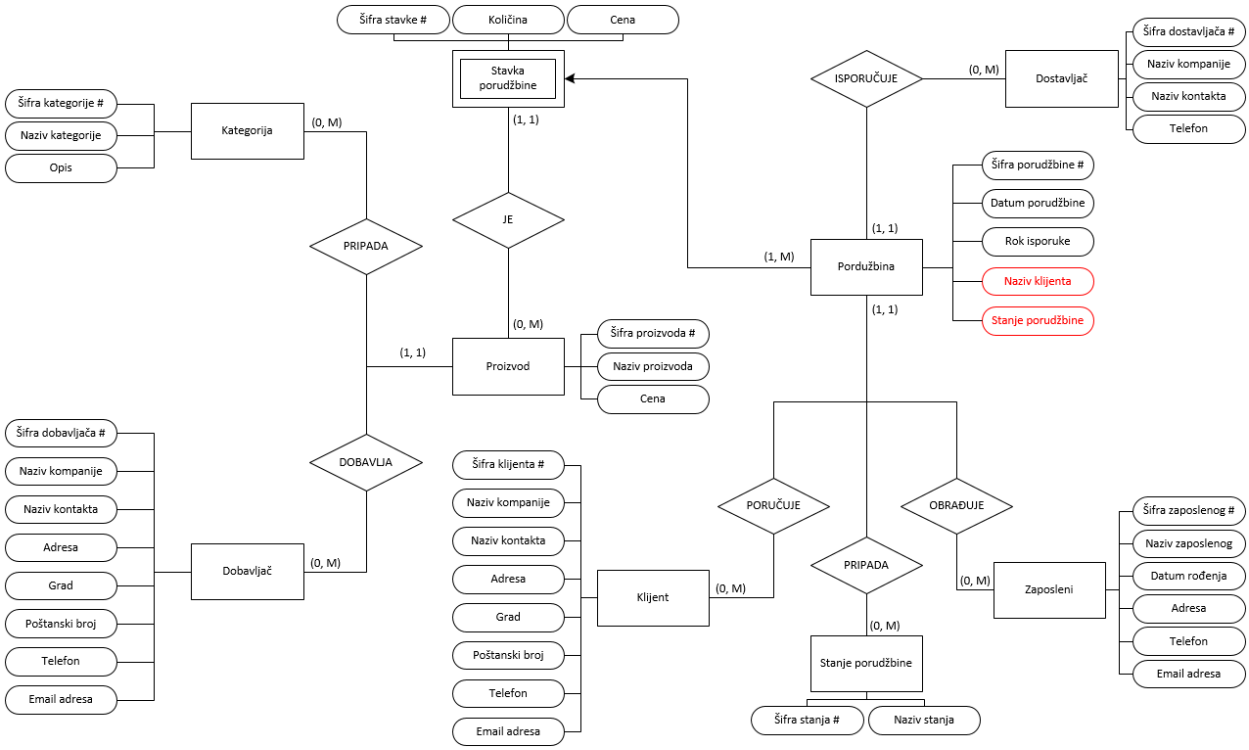
```
CREATE DATABASE NovaTehnika;
USE DATABASE NovaTehnika;
GO

CREATE TABLE Dobavljac(
    SifraDobavljaca INT NOT NULL IDENTITY(1,1),
    NazivKompanije NVARCHAR(100) NOT NULL,
    NazivKontakta NVARCHAR(50) NOT NULL,
    Adresa NVARCHAR(50) NOT NULL,
    Grad NVARCHAR(50) NOT NULL,
    PostanskiBroj NVARCHAR(6) NOT NULL,
    Telefon NVARCHAR(10) NOT NULL,
    Email NVARCHAR(50) NOT NULL,
    CONSTRAINT PK_Dobavljac PRIMARY KEY(SifraDobavljaca)
);

CREATE TABLE Kategorija(
    SifraKategorije INT NOT NULL IDENTITY(1,1),
    NazivKategorije NVARCHAR(50) NOT NULL,
    Opis NVARCHAR(50) NULL,
    CONSTRAINT PK_Kategorija PRIMARY KEY(SifraKategorije)
);

CREATE TABLE Proizvod(
    SifraProizvoda INT NOT NULL IDENTITY(1,1),
    NazivProizvoda NVARCHAR(100) NOT NULL,
    Cena DECIMAL(8,2) NOT NULL,
    SifraKategorije INT NOT NULL,
    SifraDobavljaca INT NOT NULL,
    CONSTRAINT PK_Proizvod PRIMARY KEY(SifraProizvoda),
    CONSTRAINT FK_Proizvod_Kategorija FOREIGN KEY(SifraKategorije) REFERENCES
Kategorija(SifraKategorije),
    CONSTRAINT FK_Proizvod_Dobavljac FOREIGN KEY(SifraDobavljaca) REFERENCES
Dobavljac(SifraDobavljaca)
);

CREATE TABLE Klijent(
    SifraKlijenta INT NOT NULL IDENTITY(1,1),
    NazivKompanije NVARCHAR(100) NOT NULL,
    NazivKontakta NVARCHAR(50) NOT NULL,
    Adresa NVARCHAR(50) NOT NULL,
    Grad NVARCHAR(50) NOT NULL,
    PostanskiBroj NVARCHAR(6) NOT NULL,
    Telefon NVARCHAR(10) NOT NULL,
    Email NVARCHAR(50) NOT NULL,
    CONSTRAINT PK_Klijent PRIMARY KEY(SifraKlijenta)
);

CREATE TABLE Zaposleni(
    SifraZaposlenog INT NOT NULL IDENTITY(1,1),
    NazivZaposlenog NVARCHAR(30) NOT NULL,
    DatumRodjenja DATE NOT NULL,
    Adresa NVARCHAR(50) NOT NULL,
    Telefon NVARCHAR(10) NOT NULL,
    Email NVARCHAR(50) NOT NULL
    CONSTRAINT PK_Zaposleni PRIMARY KEY(SifraZaposlenog)
);
```

```
CREATE TABLE StanjePorudzbine(
    SifraStanja INT NOT NULL IDENTITY(1,1),
    NazivStanja NVARCHAR(50) NOT NULL,
    CONSTRAINT PK_StanjePorudzbine PRIMARY KEY(SifraStanja)
);

CREATE TABLE Dostavljac(
    SifraDostavljaca INT NOT NULL IDENTITY(1,1),
    NazivKompanije NVARCHAR(50) NOT NULL,
    NazivKontakta NVARCHAR(50) NOT NULL,
    Telefon NVARCHAR(10) NOT NULL,
    CONSTRAINT PK_Dostavljac PRIMARY KEY(SifraDostavljaca)
);

CREATE TABLE Porudzbina(
    SifraPorudzbine INT NOT NULL IDENTITY(1,1),
    DatumPorudzbine DATE NOT NULL,
    RokIsporuke DATE NOT NULL,
    SifraKlijenta INT NOT NULL,
    SifraStanja INT NOT NULL,
    SifraZaposlenog INT NOT NULL,
    SifraDostavljaca INT NOT NULL,
    CONSTRAINT PK_Porudzbina PRIMARY KEY(SifraPorudzbine),
    CONSTRAINT FK_Porudzbina_Klijent FOREIGN KEY(SifraKlijenta) REFERENCES
Klijent(SifraKlijenta),
    CONSTRAINT FK_Porudzbina_StanjePorudzbine FOREIGN KEY(SifraStanja) REFERENCES
StanjePorudzbine(SifraStanja),
    CONSTRAINT FK_Porudzbina_Zaposleni FOREIGN KEY(SifraZaposlenog) REFERENCES
Zaposleni(SifraZaposlenog),
    CONSTRAINT FK_Porudzbina_Dostavljac FOREIGN KEY(SifraDostavljaca) REFERENCES
Dostavljac(SifraDostavljaca)
);

CREATE TABLE StavkaPorudzbine(
    SifraStavke INT NOT NULL IDENTITY(1,1),
    SifraPorudzbine INT NOT NULL,
    Kolicina INT NOT NULL,
    Cena DECIMAL(8,2) NOT NULL,
    SifraProizvoda INT NOT NULL,
    CONSTRAINT PK_StavkaPorudzbine PRIMARY KEY(SifraStavke, SifraPorudzbine),
    CONSTRAINT FK_StavkaPorudzbine_Proizvod FOREIGN KEY(SifraProizvoda) REFERENCES
Proizvod(SifraProizvoda),
    CONSTRAINT FK_StavkaPorudzbine_Porudzbina FOREIGN KEY(SifraPoruzbine) REFERENCES
Porudzbina(SifraPorudzbine)
);
```

5.2. DML naredbe

```
INSERT INTO Dobavljac(NazivKompanije, NazivKontakta, Adresa, Grad, PostanskiBroj, Telefon,
Email) VALUES
('InterKomerc D.O.O.', 'Marko Ilin', 'Miška Kranjca 14', 'Beograd', '11000', '0631488262',
'milin@gmail.com'),
('Stara Elektronika D.O.O.', 'Jovana Saveski', 'Vukasovićeva 50', 'Beograd', '11090',
'0603582541', 'jovanasaveski@elektronika.rs'),
('LEONI Wiring Systems Southeast D.O.O.', 'Stefan Josipović', 'Pane Đukića', 'Prokuplje',
'18400', '0', 'stefanj@leoni.com');

INSERT INTO Kategorija(NazivKategorije) VALUES
('Hard diskovi'),
('Procesori'),
('Grafičke kartice');

INSERT INTO Proizvod(NazivProizvoda, Cena, SifraKategorije, SifraDobavljaca) VALUES
('HDD SATA3 7200 1TB WD Black WD1003FZEX', 8585, 1, 1),
('HDD SATA3 2.5" 5400 1TB Hitachi Travelstar 5K1000', 6790, 1, 1),
('HDD SATA3 2.5" 5400 1TB Seagate Baracuda', 5292, 1, 1),
('HDD Fioka 2.5" Kolink SATA USB 2.0', 990, 1, 3),
('CPU FM2+ AMD Athlon™ X4 Quad-Core 840', 4963, 2, 2),
('APU AM4 AMD A10-9700', 10006, 2, 2),
('CPU AM3+ AMD FX-4320', 6990, 2, 2),
('CPU AM4 AMD Ryzen 3 1300X', 14290, 2, 2),
('AMD Radeon R5 230 ASUS 2GB GDDR3', 5530, 3, 2),
('AMD Radeon R7 250 Sapphire 2GB DDR3', 9990, 3, 1),
('AMD Radeon RX550 Sapphire PULSE 2GB OC GDDR5', 13990, 3, 1);

INSERT INTO Klijent(NazivKompanije, NazivKontakta, Adresa, Grad, PostanskiBroj, Telefon,
Email) VALUES
('Kelco', 'Nemanja Kojić', 'Bulevar kralja Aleksandra 326', 'Beograd', '11000',
'0113403376', 'nemanja@kelco.rs'),
('Comet Electronics D.O.O.', 'Mirko Ahmedovski', 'Blok 19a', 'Beograd', '11000',
'0116134180', 'ahmedovskim@comet.rs'),
('Mikro Princ D.O.O.', 'Aleksandar Janković', 'Kralja Milutina 31', 'Beograd', '11000',
'011362900', 'ajankovic@mikroprinc.com');

INSERT INTO Zaposleni(NazivZaposlenog, DatumRodjenja, Adresa, Telefon, Email) VALUES
('Ivan Marić', '1991-12-13', 'Ivana Mičurina 38', '0611418030', 'ivanmaric@yahoo.com'),
('Ana Petrović', '1995-03-22', 'Makedonska 13', '0622245810', 'anap@gmail.com');

INSERT INTO StanjePorudzbine(NazivStanja) VALUES
('Otkazano'),
('U obradi'),
('Isporučeno');

INSERT INTO Dostavljac(NazivKompanije, NazivKontakta, Telefon) VALUES
('Yugotrans D.O.O.', 'Marko Ledović', '0642477373'),
('Milšped D.O.O.', 'Petar Stojanović', '0605449274');

INSERT INTO Porudzbina(DatumPorudzbine, RokIsporuke, SifraKlijenta, SifraStanja,
SifraZaposlenog, SifraDostavljaca) VALUES
('2018-02-08', '2018-02-24', 1, 3, 2, 1),
('2018-03-11', '2018-03-17', 3, 1, 1, 2),
('2018-04-02', '2018-04-21', 2, 2, 2, 1);

INSERT INTO StavkaPorudzbine(SifraPorudzbine, Kolicina, Cena, SifraProizvoda) VALUES
(1, 2, 17170, 1),
(1, 1, 10006, 6),
(1, 1, 14290, 8),
(2, 10, 55300, 9),
(3, 1, 13990, 11),
(3, 1, 14290, 8);
```

6. Denormalizacija tabele „Porudžbina”



6.1. Tabela specifikacije okidača

Tabela	Tip okidača	Kolona	Potreban	Akcija
Klijent	Insert		Ne	
	Update	NazivKontakta	**Da**	Prilikom izmene vrednosti kolone *NazivKontakta* u tabeli *Klijent*, pokreće se okidač koji izmenjenu vrednost ažurira u tabeli *Porudžbina*.
	Delete	ŠifraKlijenta	**Da**	Zabraniti uklanjanje ukoliko se vrednost kolone *ŠifraKlijenta* nalazi u koloni *ŠifraKlijenta* neke n-torke tabele *Porudžbina*.
Stanje porudžbine	Insert		Ne	
	Update	NazivStanja	**Da**	Prilikom izmene vrednosti kolone *NazivStanja* u tabeli *StanjePorudžbine*, pokreće se okidač koji izmenjenu vrednost ažurira u tabeli *Porudžbina*.
	Delete	ŠifraStanja	**Da**	Zabraniti uklanjanje ukoliko se vrednost kolone *ŠifraStanja* nalazi u koloni *ŠifraStanja* neke n-torke tabele *Porudžbina*.
Porudžbina	Insert	ŠifraKlijenta	**Da**	Zabraniti dodavanje ako vrednost kolone *ŠifraKlijenta* ne postoji u koloni *ŠifraKlijenta* tabele *Klijent*.
		ŠifraStanja		Zabraniti dodavanje ako vrednost kolone *ŠifraStanja* ne postoji u koloni *ŠifraStanja* tabele *StanjePorudžbine*.
	Update	NazivKlijenta	**Da**	Zabraniti izmenu ove kolone ukoliko u tabeli *Klijent* ne postoji n-torka sa istom šifrom klijenta i nazivom.
		StanjePorudžbine		Zabraniti izmenu ove kolone ukoliko u tabeli *StanjePorudžbine* ne postoji n-torka sa istom šifrom stanja i nazivom.
	Delete		Ne	

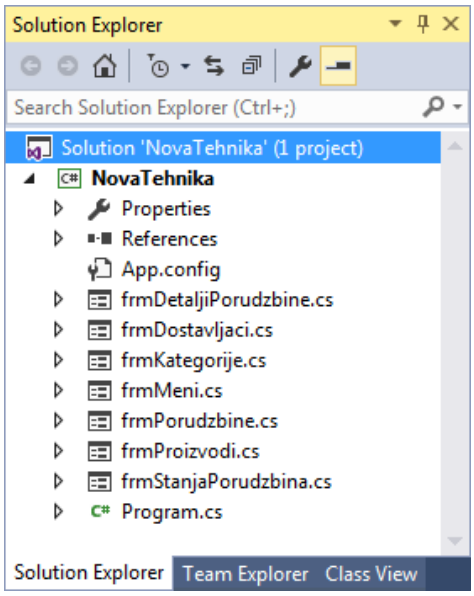
6.2. Projektovanje denormalizacije i okidača

```
ALTER TABLE Porudzbina
    ADD NazivKlijenta NVARCHAR(50),
        StanjePorudzbine NVARCHAR(50);

UPDATE Porudzbina
SET Porudzbina.NazivKlijenta = Klijent.NazivKontakta
FROM Porudzbina INNER JOIN Klijent ON Porudzbina.SifraKlijenta =
Klijent.SifraKlijenta;

UPDATE Porudzbina
SET Porudzbina.StanjePorudzbine = StanjePorudzbine.NazivStanja
FROM Porudzbina INNER JOIN StanjePorudzbine ON Porudzbina.SifraStanja =
StanjePorudzbine.SifraStanja;

CREATE TRIGGER dbo.AzurirajNazivKlijenta
ON Klijent
AFTER UPDATE
AS
 BEGIN
    IF (UPDATE (NazivKontakta))
    BEGIN
        UPDATE Porudzbina
        SET Porudzbina.NazivKlijenta = INSERTED.NazivKontakta
        FROM Porudzbina JOIN INSERTED ON Porudzbina.SifraKlijenta =
INSERTED.SifraKlijenta
    END
 END

CREATE TRIGGER dbo.ZabraniBrisanjeKlijenta
ON Klijent
FOR DELETE
AS
 BEGIN
    DECLARE @SifraKlijenta INT
    SELECT @SifraKlijenta = SifraKlijenta
    FROM DELETED

    IF EXISTS (SELECT SifraKlijenta FROM Porudzbina WHERE SifraKlijenta =
@SifraKlijenta)
        BEGIN
         THROW 5100, 'Brisanje n-torke nije dozvoljeno - Klijent postoji u
Porudžbinama.', 1;
         ROLLBACK TRANSACTION
        END
 END

CREATE TRIGGER dbo.AzurirajNazivStanjaPorudzbine
ON StanjePorudzbine
AFTER UPDATE
AS
 BEGIN
    IF (UPDATE (NazivStanja))
    BEGIN
        UPDATE Porudzbina
        SET Porudzbina.StanjePorudzbine = INSERTED.NazivStanja
        FROM Porudzbina JOIN INSERTED ON Porudzbina.SifraStanja = INSERTED.SifraStanja
    END
 END
```

```
CREATE TRIGGER dbo.ZabraniBrisanjeStanja
ON StanjePorudzbine
FOR DELETE
AS
 BEGIN
    DECLARE @SifraStanja INT
    SELECT @SifraStanja = SifraStanja
    FROM DELETED

    IF EXISTS (SELECT SifraStanja FROM Porudzbina WHERE SifraStanja =
@SifraStanja)
        BEGIN
         THROW 5100, 'Brisanje n-torke nije dozvoljeno - Stanje postoji u
Porudžbinama.', 1;
         ROLLBACK TRANSACTION
        END
 END

CREATE TRIGGER dbo.ZabraniDodavanjeIzmenuPorudzbine
ON Porudzbina
FOR INSERT, UPDATE
AS
 BEGIN
    IF EXISTS (SELECT SifraKlijenta FROM INSERTED)
     BEGIN
      DECLARE @SifraKlijenta INT
      SELECT @SifraKlijenta = SifraKlijenta FROM INSERTED
      IF NOT EXISTS (SELECT SifraKlijenta FROM Klijent WHERE SifraKlijenta =
@SifraKlijenta)
       BEGIN
        THROW 5100, 'Dodavanje ili izmena nije uspešna - Klijent ne postoji u
bazi.', 1;
        ROLLBACK TRANSACTION
       END
     END

    IF EXISTS (SELECT SifraStanja FROM INSERTED)
     BEGIN
      DECLARE @SifraStanja INT
      SELECT @SifraStanja = SifraStanja FROM INSERTED
      IF NOT EXISTS (SELECT SifraStanja FROM StanjePorudzbine WHERE
SifraStanja = @SifraStanja)
       BEGIN
        THROW 5100, 'Dodavanje ili izmena nije uspešna - Stanje ne postoji u
bazi.', 1;
        ROLLBACK TRANSACTION
       END
     END
 END
```

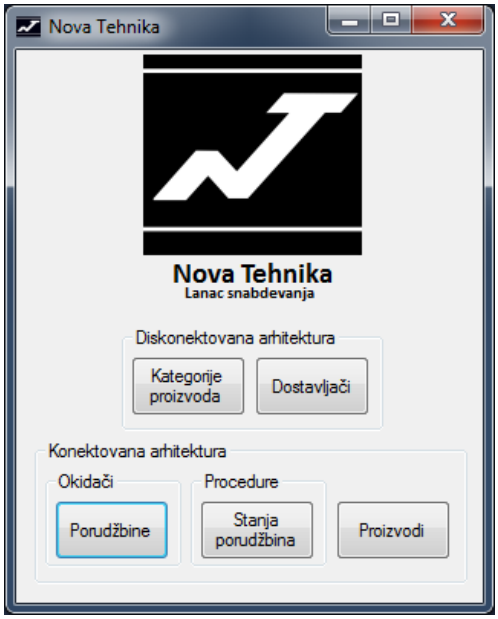
7. Implementacija rešenja

7.1. Struktura rešenja



Prikaz svih formi rešenja.

7.2. Navigacija



Izgled navigacije iz koje se pristupa formama.

7.2.1. Navigacija - kod

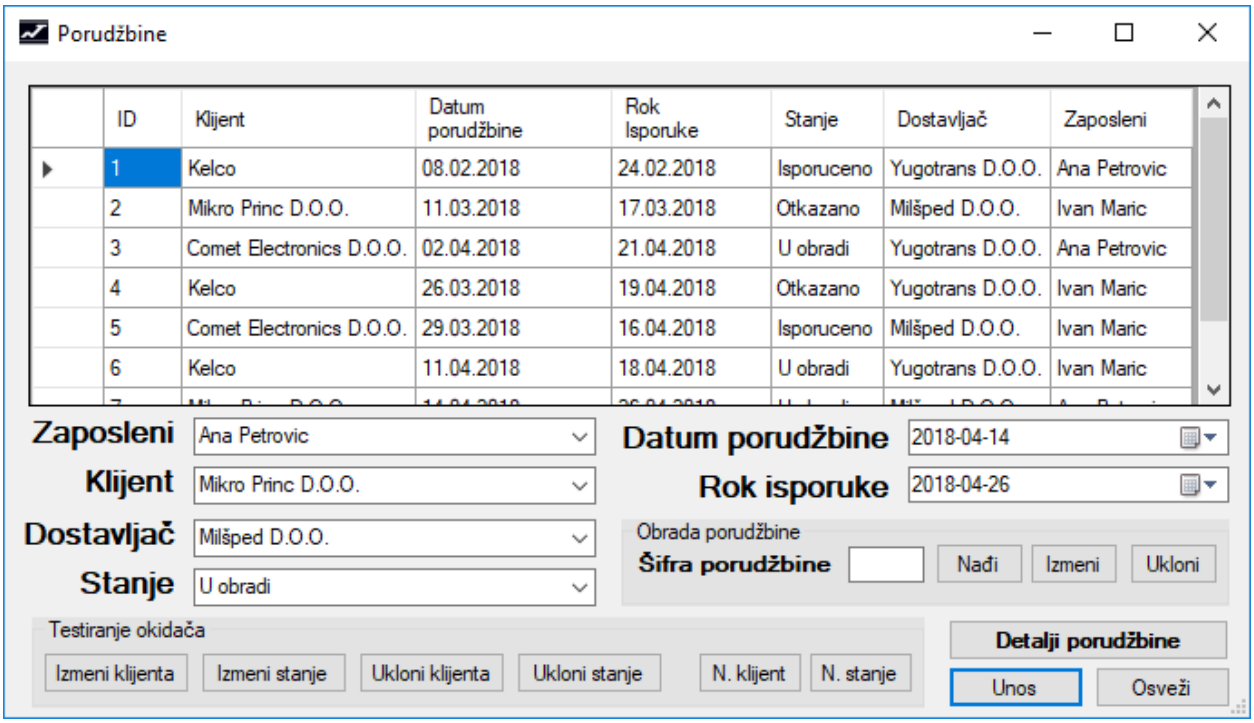
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace NovaTehnika
{
    public partial class frmMeni : Form
    {
        public frmMeni()
        {
            InitializeComponent();
        }

        private void btnDodajPorudzbinu_Click(object sender, EventArgs e)
        {
            frmPorudzbine formaPorudzbina = new frmPorudzbine();
            formaPorudzbina.ShowDialog();
        }

        private void btnProizvodi_Click(object sender, EventArgs e)
        {
            frmProizvodi formaProizvodi = new frmProizvodi();
            formaProizvodi.ShowDialog();
        }

        private void btnStanjaPorudžbine_Click(object sender, EventArgs e)
        {
            frmStanjaPorudzbina formaStanjaPorudzbina = new frmStanjaPorudzbina();
            formaStanjaPorudzbina.ShowDialog();
        }

        private void btnKategorije_Click(object sender, EventArgs e)
        {
            frmKategorije formaKategorije = new frmKategorije();
            formaKategorije.ShowDialog();
        }

        private void btnDostavljaci_Click(object sender, EventArgs e)
        {
            frmDostavljaci formaDostavljaci = new frmDostavljaci();
            formaDostavljaci.ShowDialog();
        }
    }
}
```

7.3. Konektovana arhitektura

Tabele korišćene :

- ***Porudžbina*** - *frmPorudzbine* - ***Ovde su takođe implementirani okidači iz zahteva 8.***
- ***Detalji porudžbine*** - *frmDetaljiPorudzbine* - ***Ovoj formi se pristupa preko frmPorudzbine.***
- ***Proizvodi*** - *frmProizvodi*

7.3.1. frmPorudzbine



*Izgled forme. Obratiti pažnju na dugme **Detalji porudžbine** koje njegovim pritiskom otvara frmDetaljiPorudzbine koja ima poseban konstruktor pomoću kojeg joj se prosleđuje odgovarajuća šifra porudžbine iz polja **Šifra porudžbine**.*

7.3.1.1. frmPorudzbine - kod

```
using …;
using System.Configuration;
using System.Data.SqlClient;

namespace NovaTehnika
{
    public partial class frmPorudzbine : Form
    {
        string KonekcioniString;
        SqlConnection Konekcija;
        SqlCommand Komanda;

        private void OsveziEkran()
        {
            using (Konekcija = new SqlConnection(KonekcioniString))
            {
                string KomandaTabela = "SELECT Porudzbina.SifraPorudzbine AS [ID],
Klijent.NazivKompanije AS [Klijent], Porudzbina.DatumPorudzbine AS [Datum porudžbine],
Porudzbina.RokIsporuke AS [Rok Isporuke], StanjePorudzbine.NazivStanja AS [Stanje],
Dostavljac.NazivKompanije AS [Dostavljač],  Zaposleni.NazivZaposlenog AS [Zaposleni] FROM Porudzbina
INNER JOIN Zaposleni ON Zaposleni.SifraZaposlenog = Porudzbina.SifraZaposlenog INNER JOIN Klijent ON
Klijent.SifraKlijenta = Porudzbina.SifraKlijenta INNER JOIN Dostavljac ON
Dostavljac.SifraDostavljaca = Porudzbina.SifraDostavljaca INNER JOIN StanjePorudzbine ON
StanjePorudzbine.SifraStanja = Porudzbina.SifraStanja";

                Komanda = new SqlCommand(KomandaTabela, Konekcija);
                Konekcija.Open();
                SqlDataReader Reader = Komanda.ExecuteReader();

                if (Reader.HasRows)
                {
                    DataTable Tabela = new DataTable();
                    Tabela.Load(Reader);
                    dataGridView1.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.AllCells;
                    dataGridView1.DataSource = Tabela;
                }
            }
        }

        private void UcitajZaposlene()
        {
            using(Konekcija = new SqlConnection(KonekcioniString))
            {
                try
                {
                    string Query = "SELECT SifraZaposlenog, NazivZaposlenog FROM Zaposleni";
                    SqlDataAdapter Adapter = new SqlDataAdapter(Query, Konekcija);
                    DataSet setZaposleni = new DataSet();
                    Adapter.Fill(setZaposleni, "Zaposleni");
                    cmbZaposleni.DisplayMember = "NazivZaposlenog";
                    cmbZaposleni.ValueMember = "SifraZaposlenog";
                    cmbZaposleni.DataSource = setZaposleni.Tables["Zaposleni"];
                }
                catch(Exception ex)
                {
                    MessageBox.Show(ex.Message);
                }
            }
        }

        private void UcitajKlijente()
        {
```

```
        using (Konekcija = new SqlConnection(KonekcioniString))
        {
            try
            {
                string Query = "SELECT SifraKlijenta, NazivKompanije FROM Klijent";
                SqlDataAdapter Adapter = new SqlDataAdapter(Query, Konekcija);
                DataSet setKlijent = new DataSet();
                Adapter.Fill(setKlijent, "Klijent");
                cmbKlijent.DisplayMember = "NazivKompanije";
                cmbKlijent.ValueMember = "SifraKlijenta";
                cmbKlijent.DataSource = setKlijent.Tables["Klijent"];
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }
    }

    private void UcitajDostavljace()
    {
        using (Konekcija = new SqlConnection(KonekcioniString))
        {
            try
            {
                string Query = "SELECT SifraDostavljaca, NazivKompanije FROM Dostavljac";
                SqlDataAdapter Adapter = new SqlDataAdapter(Query, Konekcija);
                DataSet setDostavljac = new DataSet();
                Adapter.Fill(setDostavljac, "Dostavljac");
                cmbDostavljac.DisplayMember = "NazivKompanije";
                cmbDostavljac.ValueMember = "SifraDostavljaca";
                cmbDostavljac.DataSource = setDostavljac.Tables["Dostavljac"];
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }
    }

    private void UcitajStanja()
    {
        using (Konekcija = new SqlConnection(KonekcioniString))
        {
            try
            {
                string Query = "SELECT SifraStanja, NazivStanja FROM StanjePorudzbine";
                SqlDataAdapter Adapter = new SqlDataAdapter(Query, Konekcija);
                DataSet setStanje = new DataSet();
                Adapter.Fill(setStanje, "Stanje");
                cmbStanje.DisplayMember = "NazivStanja";
                cmbStanje.ValueMember = "SifraStanja";
                cmbStanje.DataSource = setStanje.Tables["Stanje"];
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }
    }

    public frmPorudzbine()
    {
        InitializeComponent();
```

```
            KonekcioniString =
ConfigurationManager.ConnectionStrings["KonekcioniString"].ConnectionString;
            Konekcija = new SqlConnection(KonekcioniString);
        }

        private void frmUnosPorudzbine_Load(object sender, EventArgs e)
        {
            OsveziEkran();
            UcitajZaposlene();
            UcitajDostavljace();
            UcitajKlijente();
            UcitajStanja();
        }

        private void btnUnos_Click(object sender, EventArgs e)
        {
            // cmb.SelectedValue.ToString for ValueMember.
            // cmb.Text for DisplayMember.
            using(Konekcija = new SqlConnection(KonekcioniString))
            {
                SqlCommand Komanda = new SqlCommand("INSERT INTO Porudzbina(DatumPorudzbine,
RokIsporuke, SifraKlijenta, SifraStanja, SifraZaposlenog, SifraDostavljaca, NazivKlijenta,
StanjePorudzbine)VALUES ('" + dateDatumPorudzbine.Text + "', '" + dateRokIsporuke.Text + "', " +
cmbKlijent.SelectedValue + ", " + cmbStanje.SelectedValue + ", " + cmbZaposleni.SelectedValue + ", "
+ cmbDostavljac.SelectedValue + ", '" + cmbKlijent.Text + "', '" + cmbStanje.Text + "');",
Konekcija);
                Konekcija.Open();
                try
                {
                    Komanda.ExecuteNonQuery();
                    MessageBox.Show("Porudžbina je uspešno uneta.", "Informacija",
MessageBoxButtons.OK, MessageBoxIcon.Information);
                }
                catch(Exception ex)
                {
                    MessageBox.Show("Nastala je greška - " + ex.Message, "Greška",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
                finally
                {
                    OsveziEkran();
                    Konekcija.Close();
                }
            }
        }

        private void btnOsvezi_Click(object sender, EventArgs e)
        {
            cmbDostavljac.SelectedItem = 0;
            cmbKlijent.SelectedItem = 0;
            cmbStanje.SelectedItem = 0;
            cmbZaposleni.SelectedItem = 0;
            dateDatumPorudzbine.Text = DateTime.Now.ToString();
            dateRokIsporuke.Text = DateTime.Now.ToString();
            txtSifraPorudzbine.Text = "";
            OsveziEkran();
        }

        private void btnNadji_Click(object sender, EventArgs e)
        {
            if(txtSifraPorudzbine.Text == "")
            {
                MessageBox.Show("Polje 'šifra porudžbine' je prazno - unesite šifru porudžbine.");
            }
            else
```

```
            {
                using(Konekcija=new SqlConnection(KonekcioniString))
                {
                    SqlCommand Komanda = new SqlCommand("SELECT * FROM Porudzbina WHERE
SifraPorudzbine = " + int.Parse(txtSifraPorudzbine.Text), Konekcija);
                    Konekcija.Open();
                    try
                    {
                        SqlDataReader Reader = Komanda.ExecuteReader();

                        if (Reader.Read())
                        {
                            // SifraPorudzbine, DatumPorudzbine, RokIsporuke, SifraKlijenta,
SifraStanja, SifraZaposlenog, SifraDostavljaca, NazivKlijenta, StanjePorudzbine
                            dateDatumPorudzbine.Text = Reader[1].ToString();
                            dateRokIsporuke.Text = Reader[2].ToString();
                            cmbKlijent.SelectedValue = Reader[3];
                            cmbStanje.SelectedValue = Reader[4];
                            cmbZaposleni.SelectedValue = Reader[5];
                            cmbDostavljac.SelectedValue = Reader[6];
                        }
                        else
                        {
                            MessageBox.Show("Porudžbina sa zadatom šifrom nije pronađena.");
                        }
                        Reader.Close();
                        txtSifraPorudzbine.Focus();
                    }
                    catch (Exception ex)
                    {
                        MessageBox.Show("Dogodila se greška - " + ex.Message, "Greška",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                    }
                    finally
                    {
                        Konekcija.Close();
                    }
                }
            }
        }

        private void btnIzmeni_Click(object sender, EventArgs e)
        {
            if (txtSifraPorudzbine.Text == "")
            {
                MessageBox.Show("Polje 'šifra porudžbine' je prazno - unesite šifru porudžbine.");
            }
            else
            {
                using (Konekcija = new SqlConnection(KonekcioniString))
                {
                    SqlCommand Komanda = new SqlCommand("SELECT * FROM Porudzbina WHERE
SifraPorudzbine = " + int.Parse(txtSifraPorudzbine.Text), Konekcija);
                    Konekcija.Open();
                    try
                    {
                        SqlDataReader Reader = Komanda.ExecuteReader();

                        if (Reader.Read())
                        {
                            Reader.Close();
                            var PotvrdiIzmenu = MessageBox.Show("Potvrdite izmenu porudžbine " +
txtSifraPorudzbine.Text + ".", "Potvrdite izmenu", MessageBoxButtons.OKCancel, MessageBoxIcon.None);

                            if (PotvrdiIzmenu == DialogResult.OK)
```

```
                            {
                                // SifraPorudzbine, DatumPorudzbine, RokIsporuke, SifraKlijenta,
SifraStanja, SifraZaposlenog, SifraDostavljaca, NazivKlijenta, StanjePorudzbine
                                Komanda = new SqlCommand("UPDATE Porudzbina SET DatumPorudzbine ='"
+ dateDatumPorudzbine.Text + "', RokIsporuke = '" + dateRokIsporuke.Text + "', SifraKlijenta = " +
cmbKlijent.SelectedValue + ", SifraStanja = " + cmbStanje.SelectedValue + ", SifraZaposlenog = " +
cmbZaposleni.SelectedValue + ", SifraDostavljaca = " + cmbDostavljac.SelectedValue + ",
NazivKlijenta = '" + cmbKlijent.Text + "', StanjePorudzbine = '" + cmbStanje.Text + "' WHERE
SifraPorudzbine = " + int.Parse(txtSifraPorudzbine.Text) + ";", Konekcija);
                                try
                                {
                                    Komanda.ExecuteNonQuery();
                                    MessageBox.Show("Porudžbina je uspešno izmenjena",
"Informacija", MessageBoxButtons.OK, MessageBoxIcon.Information);
                                }
                                catch (Exception Ex)
                                {
                                    MessageBox.Show("Dogodila se greška - " + Ex.Message, "Greška",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                                }
                            }
                        }
                        else
                        {
                            MessageBox.Show("Porudžbina sa zadatom šifrom nije pronađena.");
                            txtSifraPorudzbine.Text = "";
                            txtSifraPorudzbine.Focus();
                        }
                    }
                    catch (Exception Ex)
                    {
                        MessageBox.Show("Dogodila se greška - " + Ex.Message, "Greška",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                    }
                    finally
                    {
                        Konekcija.Close();
                        OsveziEkran();
                    }
                }
            }
        }

        private void btnUkloni_Click(object sender, EventArgs e)
        {
            if (txtSifraPorudzbine.Text == "")
            {
                MessageBox.Show("Polje 'šifra porudžbine' je prazno - unesite šifru porudžbine.");
            }
            else
            {
                using (Konekcija = new SqlConnection(KonekcioniString))
                {
                    SqlCommand Komanda = new SqlCommand("SELECT * FROM Porudzbina WHERE
SifraPorudzbine = " + int.Parse(txtSifraPorudzbine.Text), Konekcija);
                    Konekcija.Open();
                    try
                    {
                        SqlDataReader Reader = Komanda.ExecuteReader();

                        if (Reader.Read())
                        {
                            Reader.Close();
```

```
                            var PotvrdiUklanjanje = MessageBox.Show("Potvrdite uklanjanje porudžbine
" + txtSifraPorudzbine.Text + ".", "Potvrdite uklanjanje", MessageBoxButtons.OKCancel,
MessageBoxIcon.Warning);

                            if (PotvrdiUklanjanje == DialogResult.OK)
                            {
                                SqlCommand KomandaStavke = new SqlCommand("DELETE FROM
StavkaPorudzbine WHERE SifraPorudzbine = " + int.Parse(txtSifraPorudzbine.Text), Konekcija);
                                Komanda = new SqlCommand("DELETE FROM Porudzbina WHERE
SifraPorudzbine = " + int.Parse(txtSifraPorudzbine.Text), Konekcija);
                                try
                                {
                                    KomandaStavke.ExecuteNonQuery();
                                    Komanda.ExecuteNonQuery();
                                    MessageBox.Show("Porudžbina je uspešno uklonjena.", "Uklanjanje
porudžbine", MessageBoxButtons.OK, MessageBoxIcon.Information);
                                }
                                catch (Exception Ex)
                                {
                                    MessageBox.Show("Dogodila se greška - " + Ex.Message, "Greška",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                                }
                            }
                        }
                        else
                        {
                            MessageBox.Show("Porudžbina sa zadatom šifrom nije pronađena.");
                            txtSifraPorudzbine.Text = "";
                            txtSifraPorudzbine.Focus();
                        }
                    }
                    catch (Exception Ex)
                    {
                        MessageBox.Show("Dogodila se greška - " + Ex.Message, "Greška",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                    }
                    finally
                    {
                        Konekcija.Close();
                        OsveziEkran();
                    }
                }
            }
        }

        private void btnDetaljiPorudzbine_Click(object sender, EventArgs e)
        {
            if (txtSifraPorudzbine.Text == "")
            {
                MessageBox.Show("Polje 'šifra porudžbine' je prazno - unesite šifru porudžbine.");
            }
            else
            {
                frmDetaljiPorudzbine DetaljiPorudzbine = new
frmDetaljiPorudzbine(int.Parse(txtSifraPorudzbine.Text));
                DetaljiPorudzbine.ShowDialog();
            }
        }
```

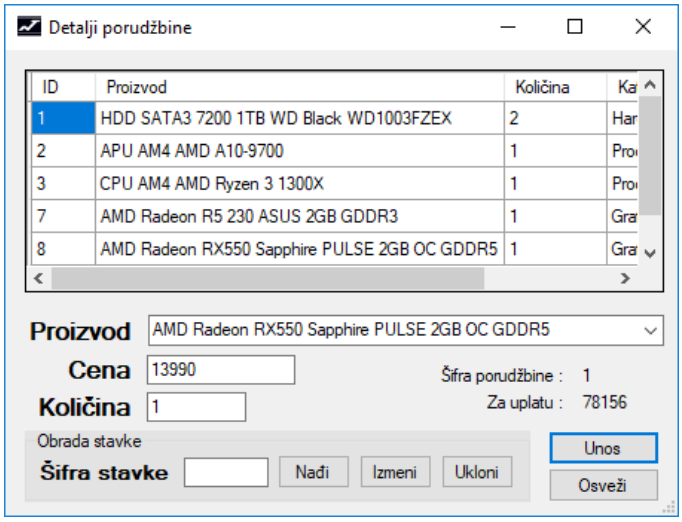
7.3.1.2. frmPorudzbine - Kod za okidače

```
        private void btnIzmeniKlijenta_Click(object sender, EventArgs e)
        {
            MessageBox.Show("Ovaj okidač će prilikom izmene vrednosti kolone `NazivKontakta` u
tabeli `Klijent` ažurirati vrednost kolone `NazivKlijenta` tabele `Porudzbina`.");
            using (Konekcija = new SqlConnection(KonekcioniString))
            {
                SqlCommand Komanda = new SqlCommand("UPDATE Klijent SET NazivKontakta =
'KlijentNazivOkidač' WHERE SifraKlijenta = 1", Konekcija);
                Konekcija.Open();
                try
                {
                    Komanda.ExecuteNonQuery();
                    MessageBox.Show("Naziv klijenta sa šifrom klijenta 1 je uspešno ažuriran u
`KlijentNazivOkidač`.", "Informacija", MessageBoxButtons.OK, MessageBoxIcon.Information);
                }
                catch (Exception Ex)
                {
                    MessageBox.Show("Dogodila se greška - " + Ex.Message, "Greška",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
                finally
                {
                    Konekcija.Close();
                    OsveziEkran();
                }
            }
        }

        private void btnIzmeniStanje_Click(object sender, EventArgs e)
        {
            MessageBox.Show("Ovaj okidač će prilikom izmene vrednosti kolone `NazivStanja` u
tabeli `StanjePorudzbine` ažurirati vrednost kolone `StanjePorudzbine` tabele `Porudzbina`.");
            using (Konekcija = new SqlConnection(KonekcioniString))
            {
                SqlCommand Komanda = new SqlCommand("UPDATE StanjePorudzbine SET NazivStanja =
'StanjeNazivOkidač' WHERE SifraStanja = 1", Konekcija);
                Konekcija.Open();
                try
                {
                    Komanda.ExecuteNonQuery();
                    MessageBox.Show("Naziv stanja sa šifrom stanja 1 je uspešno ažuriran u
`StanjeNazivOkidač`.", "Informacija", MessageBoxButtons.OK, MessageBoxIcon.Information);
                }
                catch (Exception Ex)
                {
                    MessageBox.Show("Dogodila se greška - " + Ex.Message, "Greška",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
                finally
                {
                    Konekcija.Close();
                    OsveziEkran();
                }
            }
        }
```

```
        private void btnUkloniKlijenta_Click(object sender, EventArgs e)
        {
            MessageBox.Show("Ovaj okidač će zabraniti uklanjanje n-torke ukoliko se vrednost
kolone `SifraKlijenta` te n-torke nalazi u koloni `SifraKlijenta` neke n-torke tabele
`Porudzbina`.");
            using (Konekcija = new SqlConnection(KonekcioniString))
            {
                SqlCommand Komanda = new SqlCommand("DELETE FROM Klijent WHERE SifraKlijenta =
1", Konekcija);
                Konekcija.Open();
                try
                {
                    Komanda.ExecuteNonQuery();
                    MessageBox.Show("Klijent sa šifrom 1 je uspešno uklonjen.", "Informacija",
MessageBoxButtons.OK, MessageBoxIcon.Information);
                }
                catch (Exception Ex)
                {
                    MessageBox.Show("Dogodila se greška - " + Ex.Message, "Greška",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
                finally
                {
                    Konekcija.Close();
                    OsveziEkran();
                }
            }
        }

        private void btnUkloniStanje_Click(object sender, EventArgs e)
        {
            MessageBox.Show("Ovaj okidač će zabraniti uklanjanje n-torke ukoliko se vrednost
kolone `SifraStanja` te n-torke nalazi u koloni `SifraStanja` neke n-torke tabele
`Porudzbina`.");
            using (Konekcija = new SqlConnection(KonekcioniString))
            {
                SqlCommand Komanda = new SqlCommand("DELETE FROM StanjePorudzbine WHERE
SifraStanja = 1", Konekcija);
                Konekcija.Open();
                try
                {
                    Komanda.ExecuteNonQuery();
                    MessageBox.Show("Stanje sa šifrom 1 je uspešno uklonjeno.", "Informacija",
MessageBoxButtons.OK, MessageBoxIcon.Information);
                }
                catch (Exception Ex)
                {
                    MessageBox.Show("Dogodila se greška - " + Ex.Message, "Greška",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
                finally
                {
                    Konekcija.Close();
                    OsveziEkran();
                }
            }
        }
```

```
        private void btnNepostojeciKlijent_Click(object sender, EventArgs e)
        {
            MessageBox.Show("Ovaj okidač će zabraniti ažuriranje n-torke iz tabele
`Porudzbina` ukoliko n-torka sa vrednostima kolona `SifraKlijenta` i `NazivKlijenta` ne
postoji u tabeli `Klijent`.");
            using (Konekcija = new SqlConnection(KonekcioniString))
            {
                SqlCommand Komanda = new SqlCommand("UPDATE Porudzbina SET SifraKlijenta =
9999, NazivKlijenta = 'Nepostojeći klijent' WHERE SifraPorudzbine = 1", Konekcija);
                Konekcija.Open();
                try
                {
                    Komanda.ExecuteNonQuery();
                    MessageBox.Show("Porudžbina sa šifrom 1 je uspešno ažurirana.",
"Informacija", MessageBoxButtons.OK, MessageBoxIcon.Information);
                }
                catch (Exception Ex)
                {
                    MessageBox.Show("Dogodila se greška - " + Ex.Message, "Greška",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
                finally
                {
                    Konekcija.Close();
                    OsveziEkran();
                }
            }
        }

        private void btnNepostojeceStanje_Click(object sender, EventArgs e)
        {
            MessageBox.Show("Ovaj okidač će zabraniti ažuriranje n-torke iz tabele
`Porudzbina` ukoliko n-torka sa vrednostima kolona `SifraStanja` i `StanjePorudzbine` ne
postoji u tabeli `StanjePorudzbine`.");
            using (Konekcija = new SqlConnection(KonekcioniString))
            {
                SqlCommand Komanda = new SqlCommand("UPDATE Porudzbina SET SifraStanja = 9999,
StanjePorudzbine = 'Nepostojeće stanje' WHERE SifraPorudzbine = 1", Konekcija);
                Konekcija.Open();
                try
                {
                    Komanda.ExecuteNonQuery();
                    MessageBox.Show("Porudžbina sa šifrom 1 je uspešno ažurirana.",
"Informacija", MessageBoxButtons.OK, MessageBoxIcon.Information);
                }
                catch (Exception Ex)
                {
                    MessageBox.Show("Dogodila se greška - " + Ex.Message, "Greška",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
                finally
                {
                    Konekcija.Close();
                    OsveziEkran();
                }
            }
        }
```

7.3.2. frmDetaljiPorudzbine



Izgled forme.

7.3.2.1. frmDetaljiPorudzbine - Kod

```
using …;
using System.Data.SqlClient;
using System.Configuration;

namespace NovaTehnika
{
    public partial class frmDetaljiPorudzbine : Form
    {
        int SifraPorudzbine;
        double ZaUplatu;
        double Cena;
        string KonekcioniString;
        SqlConnection Konekcija;
        SqlCommand Komanda;

        private void OsveziEkran()
        {
            using (Konekcija = new SqlConnection(KonekcioniString))
            {
                string KomandaTabela = "SELECT StavkaPorudzbine.SifraStavke AS [ID],
Proizvod.NazivProizvoda AS [Proizvod], StavkaPorudzbine.Kolicina AS [Količina],
Kategorija.NazivKategorije AS [Kategorija], StavkaPorudzbine.Cena AS [Cena] FROM
StavkaPorudzbine INNER JOIN Proizvod ON StavkaPorudzbine.SifraProizvoda =
Proizvod.SifraProizvoda INNER JOIN Kategorija ON Kategorija.SifraKategorije =
Proizvod.SifraKategorije WHERE StavkaPorudzbine.SifraPorudzbine =" + SifraPorudzbine;

                Komanda = new SqlCommand(KomandaTabela, Konekcija);
                Konekcija.Open();
                SqlDataReader Reader = Komanda.ExecuteReader();

                if (Reader.HasRows)
                {
                    DataTable Tabela = new DataTable();
```

```
                    Tabela.Load(Reader);
                    dataGridView1.AutoSizeColumnsMode =
DataGridViewAutoSizeColumnsMode.AllCells;
                    dataGridView1.DataSource = Tabela;
                }
                Reader.Close();
                string ZaUplatuUpit = "SELECT
SUM(StavkaPorudzbine.Cena*StavkaPorudzbine.Kolicina) FROM StavkaPorudzbine WHERE
StavkaPorudzbine.SifraPorudzbine = " + SifraPorudzbine;
                Komanda = new SqlCommand(ZaUplatuUpit, Konekcija);
                if(Komanda.ExecuteScalar() != DBNull.Value)
                {
                    ZaUplatu = Convert.ToDouble(Komanda.ExecuteScalar());
                }
                else
                {
                    ZaUplatu = 0;
                }
                lblZaUplatuValue.Text = ZaUplatu.ToString();
            }
        }

        private void UcitajProizvode()
        {
            using (Konekcija = new SqlConnection(KonekcioniString))
            {
                try
                {
                    string Query = "SELECT SifraProizvoda, NazivProizvoda FROM Proizvod";
                    SqlDataAdapter Adapter = new SqlDataAdapter(Query, Konekcija);
                    DataSet setProizvod = new DataSet();
                    Adapter.Fill(setProizvod, "Proizvodi");
                    cmbProizvod.DisplayMember = "NazivProizvoda";
                    cmbProizvod.ValueMember = "SifraProizvoda";
                    cmbProizvod.DataSource = setProizvod.Tables["Proizvodi"];
                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.Message);
                }
            }
        }

        public frmDetaljiPorudzbine(int SifraPor)
        {
            InitializeComponent();
            SifraPorudzbine = SifraPor;
            KonekcioniString =
ConfigurationManager.ConnectionStrings["KonekcioniString"].ConnectionString;
            Konekcija = new SqlConnection(KonekcioniString);
        }

        private void frmDetaljiPorudzbine_Load(object sender, EventArgs e)
        {
            OsveziEkran();
            UcitajProizvode();
            lblSifraPorudzbineValue.Text = SifraPorudzbine.ToString();
        }
```

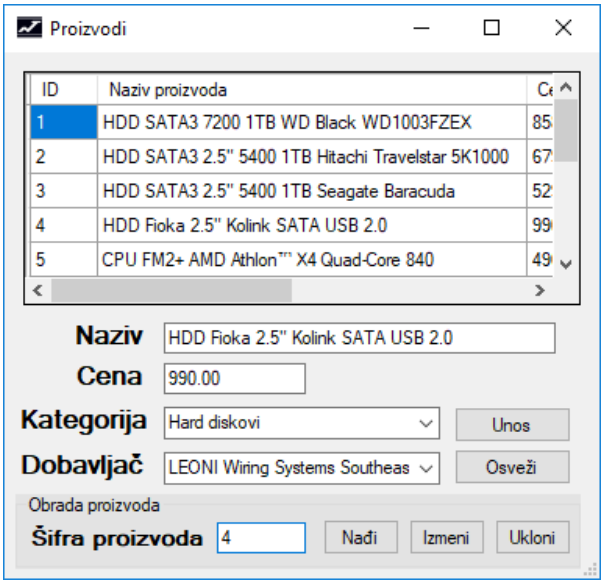
```
        private void cmbProizvod_SelectedIndexChanged(object sender, EventArgs e)
        {
            using (Konekcija = new SqlConnection(KonekcioniString))
            {
                string Upit = "SELECT Cena FROM Proizvod WHERE SifraProizvoda = " +
cmbProizvod.SelectedValue;
                Komanda = new SqlCommand(Upit, Konekcija);
                Konekcija.Open();
                Cena = Convert.ToDouble(Komanda.ExecuteScalar());
                txtCena.Text = Cena.ToString();
                txtKolicina.Text = "1";
                Konekcija.Close();
            }
        }

       private void btnUnos_Click(object sender, EventArgs e)
        {
            if (txtCena.Text == "" || txtKolicina.Text == "")
            {
                MessageBox.Show("Popunite sva polja pre unosa.");
            }
            else
            {
                using (Konekcija = new SqlConnection(KonekcioniString))
                {
                    SqlCommand Komanda = new SqlCommand("INSERT INTO StavkaPorudzbine
(SifraPorudzbine, Kolicina, Cena, SifraProizvoda) VALUES (" + SifraPorudzbine + ", " +
int.Parse(txtKolicina.Text) + ", " + double.Parse(txtCena.Text) + ", " +
cmbProizvod.SelectedValue + ");", Konekcija);
                    Konekcija.Open();
                    try
                    {
                        Komanda.ExecuteNonQuery();
                        MessageBox.Show("Stavka je uspešno uneta.", "Informacija",
MessageBoxButtons.OK, MessageBoxIcon.Information);
                    }
                    catch (Exception ex)
                    {
                        MessageBox.Show("Nastala je greška - " + ex.Message, "Greška",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                    }
                    finally
                    {
                        OsveziEkran();
                        Konekcija.Close();
                    }
                }
            }
        }

       private void btnOsvezi_Click(object sender, EventArgs e)
        {
            cmbProizvod.SelectedItem = 0;
            txtCena.Text = "";
            txtKolicina.Text = "";
        }
```

```
        private void btnNadji_Click(object sender, EventArgs e)
        {
            if (txtSifraStavke.Text == "")
            {
                MessageBox.Show("Polje 'šifra stavke' je prazno - unesite šifru
stavke.");
            }
            else
            {
                using (Konekcija = new SqlConnection(KonekcioniString))
                {
                    SqlCommand Komanda = new SqlCommand("SELECT * FROM StavkaPorudzbine
WHERE SifraStavke = " + int.Parse(txtSifraStavke.Text) + " AND SifraPorudzbine = " +
SifraPorudzbine, Konekcija);
                    Konekcija.Open();
                    try
                    {
                        SqlDataReader Reader = Komanda.ExecuteReader();

                        if (Reader.Read())
                        {
                            // SifraStavke, SifraPorudzbine, Kolicina, Cena,
SifraProizvoda
                            txtKolicina.Text = Reader[2].ToString();
                            txtCena.Text = Reader[3].ToString();
                            cmbProizvod.SelectedValue = Reader[4];
                        }
                        else
                        {
                            MessageBox.Show("Stavka sa zadatom šifrom unutar trenutne
porudžbine nije pronađena.");
                        }
                        Reader.Close();
                        txtSifraStavke.Focus();
                    }
                    catch (Exception ex)
                    {
                        MessageBox.Show("Dogodila se greška - " + ex.Message, "Greška",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                    }
                    finally
                    {
                        OsveziEkran();
                        Konekcija.Close();
                    }
                }
            }
        }

       private void btnIzmeni_Click(object sender, EventArgs e)
        {
            if (txtSifraStavke.Text == "")
            {
                MessageBox.Show("Polje 'šifra stavke' je prazno - unesite šifru stavke.");
            }
            else
            {
                using (Konekcija = new SqlConnection(KonekcioniString))
```

```
                {
                    SqlCommand Komanda = new SqlCommand("SELECT * FROM StavkaPorudzbine WHERE
SifraStavke = " + int.Parse(txtSifraStavke.Text), Konekcija);
                    Konekcija.Open();
                    try
                    {
                        SqlDataReader Reader = Komanda.ExecuteReader();

                        if (Reader.Read())
                        {
                            Reader.Close();
                            var PotvrdiIzmenu = MessageBox.Show("Potvrdite izmenu stavke " +
txtSifraStavke.Text + ".", "Potvrdite izmenu", MessageBoxButtons.OKCancel,
MessageBoxIcon.None);

                            if (PotvrdiIzmenu == DialogResult.OK)
                            {
                                // SifraStavke, SifraPorudzbine, Kolicina, Cena,
SifraProizvoda
                                string UpdateNaredba = "UPDATE StavkaPorudzbine SET Kolicina
=" + int.Parse(txtKolicina.Text) + ", Cena=" + double.Parse(txtCena.Text) + ",
SifraProizvoda=" + cmbProizvod.SelectedValue + "WHERE SifraStavke =" +
int.Parse(txtSifraStavke.Text);
                                Komanda = new SqlCommand(UpdateNaredba, Konekcija);
                                try
                                {
                                    Komanda.ExecuteNonQuery();
                                    MessageBox.Show("Stavka je uspešno izmenjena",
"Informacija", MessageBoxButtons.OK, MessageBoxIcon.Information);
                                }
                                catch (Exception Ex)
                                {
                                    MessageBox.Show("Dogodila se greška - " + Ex.Message,
"Greška", MessageBoxButtons.OK, MessageBoxIcon.Error);
                                }
                            }
                        }
                        else
                        {
                            MessageBox.Show("Stavka sa zadatom šifrom nije pronađena.");
                            txtSifraStavke.Text = "";
                            txtSifraStavke.Focus();
                        }
                    }
                    catch (Exception Ex)
                    {
                        MessageBox.Show("Dogodila se greška - " + Ex.Message, "Greška",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                    }
                    finally
                    {
                        Konekcija.Close();
                        OsveziEkran();
                    }
                }
            }
        }

        private void btnUkloni_Click(object sender, EventArgs e)
        {
            if (txtSifraStavke.Text == "")
```

```
            {
                MessageBox.Show("Polje 'šifra stavke' je prazno - unesite šifru stavke.");
            }
            else
            {
                using (Konekcija = new SqlConnection(KonekcioniString))
                {
                    SqlCommand Komanda = new SqlCommand("SELECT * FROM StavkaPorudzbine WHERE
SifraStavke = " + int.Parse(txtSifraStavke.Text), Konekcija);
                    Konekcija.Open();
                    try
                    {
                        SqlDataReader Reader = Komanda.ExecuteReader();

                        if (Reader.Read())
                        {
                            Reader.Close();
                            var PotvrdiUklanjanje = MessageBox.Show("Potvrdite uklanjanje
porudžbine " + txtSifraStavke.Text + ".", "Potvrdite uklanjanje", MessageBoxButtons.OKCancel,
MessageBoxIcon.Warning);

                            if (PotvrdiUklanjanje == DialogResult.OK)
                            {
                                // SifraStavke, SifraPorudzbine, Kolicina, Cena,
SifraProizvoda
                                string DeleteNaredba = "DELETE FROM StavkaPorudzbine WHERE
SifraStavke =" + int.Parse(txtSifraStavke.Text);
                                Komanda = new SqlCommand(DeleteNaredba, Konekcija);
                                try
                                {
                                    Komanda.ExecuteNonQuery();
                                    MessageBox.Show("Stavka je uspešno izmenjena",
"Informacija", MessageBoxButtons.OK, MessageBoxIcon.Information);
                                }
                                catch (Exception Ex)
                                {
                                    MessageBox.Show("Dogodila se greška - " + Ex.Message,
"Greška", MessageBoxButtons.OK, MessageBoxIcon.Error);
                                }
                            }
                        }
                        else
                        {
                            MessageBox.Show("Stavka sa zadatom šifrom nije pronađena.");
                            txtSifraStavke.Text = "";
                            txtSifraStavke.Focus();
                        }
                    }
                    catch (Exception Ex)
                    {
                        MessageBox.Show("Dogodila se greška - " + Ex.Message, "Greška",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                    }
                    finally
                    {
                        Konekcija.Close();
                        OsveziEkran();
                    }
                }
            }
        }}}
```

7.3.3. frmProizvodi



Izgled forme.

7.3.3.1. frmProizvodi - Kod

```
using …;
using System.Configuration;
using System.Data.SqlClient;

namespace NovaTehnika
{
    public partial class frmProizvodi : Form
    {
        string KonekcioniString;
        SqlConnection Konekcija;
        SqlCommand Komanda;

        public frmProizvodi()
        {
            InitializeComponent();
            KonekcioniString =
ConfigurationManager.ConnectionStrings["KonekcioniString"].ConnectionString;
            Konekcija = new SqlConnection(KonekcioniString);
        }

        private void frmProizvodi_Load(object sender, EventArgs e)
        {
            OsveziEkran();
            UcitajKategorije();
            UcitajDobavljace();
        }

       private void OsveziEkran()
        {
            using (Konekcija = new SqlConnection(KonekcioniString))
            {
```

```
                string KomandaTabela = "SELECT Proizvod.SifraProizvoda AS [ID],
Proizvod.NazivProizvoda AS [Naziv proizvoda], Proizvod.Cena, Kategorija.NazivKategorije
AS [Kategorija], Dobavljac.NazivKompanije AS [Dobavljač] FROM Proizvod INNER JOIN
Kategorija ON Proizvod.SifraKategorije = Kategorija.SifraKategorije INNER JOIN Dobavljac
on Proizvod.SifraDobavljaca = Dobavljac.SifraDobavljaca";

                Komanda = new SqlCommand(KomandaTabela, Konekcija);
                Konekcija.Open();
                SqlDataReader Reader = Komanda.ExecuteReader();

                if (Reader.HasRows)
                {
                    DataTable Tabela = new DataTable();
                    Tabela.Load(Reader);
                    dataGridView1.AutoSizeColumnsMode =
DataGridViewAutoSizeColumnsMode.AllCells;
                    dataGridView1.DataSource = Tabela;
                }
            }
        }

        private void UcitajKategorije()
        {
            using (Konekcija = new SqlConnection(KonekcioniString))
            {
                try
                {
                    string Query = "SELECT SifraKategorije, NazivKategorije FROM
Kategorija";
                    SqlDataAdapter Adapter = new SqlDataAdapter(Query, Konekcija);
                    DataSet setKategorije = new DataSet();
                    Adapter.Fill(setKategorije, "Kategorije");
                    cmbKategorija.DisplayMember = "NazivKategorije";
                    cmbKategorija.ValueMember = "SifraKategorije";
                    cmbKategorija.DataSource = setKategorije.Tables["Kategorije"];
                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.Message);
                }
            }
        }

        private void UcitajDobavljace()
        {
            using (Konekcija = new SqlConnection(KonekcioniString))
            {
                try
                {
                    string Query = "SELECT SifraDobavljaca, NazivKompanije FROM
Dobavljac";
                    SqlDataAdapter Adapter = new SqlDataAdapter(Query, Konekcija);
                    DataSet setDobavljac = new DataSet();
                    Adapter.Fill(setDobavljac, "Dobavljac");
                    cmbDobavljac.DisplayMember = "NazivKompanije";
                    cmbDobavljac.ValueMember = "SifraDobavljaca";
                    cmbDobavljac.DataSource = setDobavljac.Tables["Dobavljac"];
                }
```

```
                catch (Exception ex)
                {
                    MessageBox.Show(ex.Message);
                }
            }
        }

        private void btnUnos_Click(object sender, EventArgs e)
        {
            if (txtNaziv.Text == "" || txtCena.Text == "")
            {
                MessageBox.Show("Popunite sva polja pre unosa.");
            }
            else
            {
                using (Konekcija = new SqlConnection(KonekcioniString))
                {
                    SqlCommand Komanda = new SqlCommand("INSERT INTO
Proizvod(NazivProizvoda, Cena, SifraKategorije, SifraDobavljaca) VALUES('" +
txtNaziv.Text + "', " + double.Parse(txtCena.Text) + ", " + cmbKategorija.SelectedValue +
", " + cmbDobavljac.SelectedValue + ");", Konekcija);
                    Konekcija.Open();
                    try
                    {
                        Komanda.ExecuteNonQuery();
                        MessageBox.Show("Proizvod je uspešno unet.", "Informacija",
MessageBoxButtons.OK, MessageBoxIcon.Information);
                    }
                    catch (Exception ex)
                    {
                        MessageBox.Show("Nastala je greška - " + ex.Message, "Greška",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                    }
                    finally
                    {
                        OsveziEkran();
                        Konekcija.Close();
                    }
                }
            }
        }

        private void btnOsvezi_Click(object sender, EventArgs e)
        {
            txtNaziv.Text = "";
            txtCena.Text = "";
            txtSifraProizvoda.Text = "";
            cmbDobavljac.SelectedItem = 1;
            cmbKategorija.SelectedItem = 1;
            OsveziEkran();
        }

        private void btnNadji_Click(object sender, EventArgs e)
        {
            if (txtSifraProizvoda.Text == "")
            {
                MessageBox.Show("Polje 'šifra proizvoda' je prazno - unesite šifru
proizvoda.");
```

```
            }
            else
            {
                using (Konekcija = new SqlConnection(KonekcioniString))
                {
                    SqlCommand Komanda = new SqlCommand("SELECT * FROM Proizvod WHERE
SifraProizvoda = " + int.Parse(txtSifraProizvoda.Text), Konekcija);
                    Konekcija.Open();
                    try
                    {
                        SqlDataReader Reader = Komanda.ExecuteReader();

                        if (!Reader.Read() || Reader[1].ToString() == "")
                        {
                            MessageBox.Show("Proizvod sa zadatom šifrom nije pronađen.");

                        }
                        else
                        {
                            // SifraProizvoda, NazivProizvoda, Cena, SifraKategorije,
SifraDobavljaca
                            txtNaziv.Text = Reader[1].ToString();
                            txtCena.Text = Reader[2].ToString();
                            cmbKategorija.SelectedValue = Reader[3];
                            cmbDobavljac.SelectedValue = Reader[4];
                        }
                        Reader.Close();
                        txtSifraProizvoda.Focus();
                    }
                    catch (Exception ex)
                    {
                        MessageBox.Show("Dogodila se greška - " + ex.Message, "Greška",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                    }
                    finally
                    {
                        Konekcija.Close();
                    }
                }
            }
        }

       private void btnIzmeni_Click(object sender, EventArgs e)
        {
            if (txtSifraProizvoda.Text == "")
            {
                MessageBox.Show("Polje 'šifra proizvoda' je prazno - unesite šifru
proizvoda.");
            }
            else
            {
                using (Konekcija = new SqlConnection(KonekcioniString))
                {
                    SqlCommand Komanda = new SqlCommand("SELECT * FROM Proizvod WHERE
SifraProizvoda = " + int.Parse(txtSifraProizvoda.Text), Konekcija);
                    Konekcija.Open();
                    try
                    {
```

```
                        SqlDataReader Reader = Komanda.ExecuteReader();

                        if (Reader.Read())
                        {
                            Reader.Close();
                            var PotvrdiIzmenu = MessageBox.Show("Potvrdite izmenu stavke
" + txtSifraProizvoda.Text + ".", "Potvrdite izmenu", MessageBoxButtons.OKCancel,
MessageBoxIcon.None);

                            if (PotvrdiIzmenu == DialogResult.OK)
                            {
                                // SifraProizvoda, NazivProizvoda, Cena, SifraKategorije,
SifraDobavljaca
                                string UpdateNaredba = "UPDATE Proizvod SET
NazivProizvoda = '" + txtNaziv.Text + "', Cena = " + Double.Parse(txtCena.Text) + ",
SifraKategorije = " + cmbKategorija.SelectedValue + ", SifraDobavljaca = " +
cmbDobavljac.SelectedValue + " WHERE SifraProizvoda = " +
int.Parse(txtSifraProizvoda.Text);
                                Komanda = new SqlCommand(UpdateNaredba, Konekcija);
                                try
                                {
                                    Komanda.ExecuteNonQuery();
                                    MessageBox.Show("Proizvod je uspešno izmenjen",
"Informacija", MessageBoxButtons.OK, MessageBoxIcon.Information);
                                }
                                catch (Exception Ex)
                                {
                                    MessageBox.Show("Dogodila se greška - " + Ex.Message,
"Greška", MessageBoxButtons.OK, MessageBoxIcon.Error);
                                }
                            }
                        }
                        else
                        {
                            MessageBox.Show("Proizvod sa zadatom šifrom nije pronađen.");
                            txtSifraProizvoda.Text = "";
                            txtSifraProizvoda.Focus();
                        }
                    }
                    catch (Exception Ex)
                    {
                        MessageBox.Show("Dogodila se greška - " + Ex.Message, "Greška",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                    }
                    finally
                    {
                        Konekcija.Close();
                        OsveziEkran();
                    }
                }
            }
        }

        private void btnUkloni_Click(object sender, EventArgs e)
        {
            if (txtSifraProizvoda.Text == "")
            {
```

```
                MessageBox.Show("Polje 'šifra proizvoda' je prazno - unesite šifru
proizvoda.");
            }
            else
            {
                using (Konekcija = new SqlConnection(KonekcioniString))
                {
                    SqlCommand Komanda = new SqlCommand("SELECT * FROM Proizvod WHERE
SifraProizvoda = " + int.Parse(txtSifraProizvoda.Text), Konekcija);
                    Konekcija.Open();
                    try
                    {
                        SqlDataReader Reader = Komanda.ExecuteReader();

                        if (Reader.Read())
                        {
                            Reader.Close();
                            var PotvrdiUklanjanje = MessageBox.Show("Potvrdite uklanjanje
proizvoda " + txtSifraProizvoda.Text + ".", "Potvrdite uklanjanje",
MessageBoxButtons.OKCancel, MessageBoxIcon.Warning);

                            if (PotvrdiUklanjanje == DialogResult.OK)
                            {
                                string DeleteNaredba = "DELETE FROM Proizvod WHERE
SifraProizvoda = " + int.Parse(txtSifraProizvoda.Text);
                                Komanda = new SqlCommand(DeleteNaredba, Konekcija);
                                try
                                {
                                    Komanda.ExecuteNonQuery();
                                    MessageBox.Show("Proizvod je uspešno uklonjen",
"Informacija", MessageBoxButtons.OK, MessageBoxIcon.Information);
                                }
                                catch (Exception Ex)
                                {
                                    MessageBox.Show("Dogodila se greška - " + Ex.Message,
"Greška", MessageBoxButtons.OK, MessageBoxIcon.Error);
                                }
                            }
                        }
                        else
                        {
                            MessageBox.Show("Proizvod sa zadatom šifrom nije pronađen.");
                            txtSifraProizvoda.Text = "";
                            txtSifraProizvoda.Focus();
                        }
                    }
                    catch (Exception Ex)
                    {
                        MessageBox.Show("Dogodila se greška - " + Ex.Message, "Greška",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                    }
                    finally
                    {
                        Konekcija.Close();
                        OsveziEkran();
                    }
                }
            }}}}
```

7.4. Uskladištene procedure

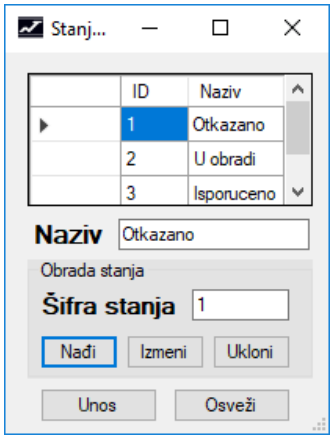
Tabele korišćene :

- ***Stanja porudžbina*** - *frmStanjaPorudzbina*

7.4.1. Projektovanje uskladištenih procedura

```
CREATE PROCEDURE sp_PrikaziStanja
AS
BEGIN
    SELECT SifraStanja AS [ID], NazivStanja AS [Naziv]
    FROM StanjePorudzbine
END

CREATE PROCEDURE sp_VratiStanjePoSifri
@SifraStanja INT,
@NazivStanja NVARCHAR(50) OUTPUT
AS
BEGIN
    SELECT @NazivStanja = NazivStanja
    FROM StanjePorudzbine
    WHERE SifraStanja = @SifraStanja
END

CREATE PROCEDURE sp_UnesiStanje
@NazivStanja NVARCHAR(50)
AS
BEGIN
    INSERT INTO StanjePorudzbine(NazivStanja) VALUES (@NazivStanja)
END

CREATE PROCEDURE sp_AzurirajStanje
@NazivStanja NVARCHAR(50),
@SifraStanja INT
AS
BEGIN
    UPDATE StanjePorudzbine
    SET NazivStanja = @NazivStanja
    WHERE SifraStanja = @SifraStanja
END

CREATE PROCEDURE sp_UkloniStanje
@SifraStanja INT
AS
BEGIN
    DELETE FROM StanjePorudzbine
    WHERE SifraStanja = @SifraStanja
END
```

7.4.2. frmStanjaPorudzbina



Izgled forme.

7.4.2.1. frmStanjaPorudzbina - Kod

```
using …;
using System.Configuration;
using System.Data.SqlClient;

namespace NovaTehnika
{
    public partial class frmStanjaPorudzbina : Form
    {
        string KonekcioniString;
        SqlConnection Konekcija;
        SqlCommand Komanda;

        public frmStanjaPorudzbina()
        {
            InitializeComponent();
        }

        private void frmStanjaPorudzbine_Load(object sender, EventArgs e)
        {
            KonekcioniString =
ConfigurationManager.ConnectionStrings["KonekcioniString"].ConnectionString;
            Konekcija = new SqlConnection(KonekcioniString);
            OsveziEkran();
        }

        private void OsveziEkran()
        {
            using (Konekcija = new SqlConnection(KonekcioniString))
            {
                Komanda = new SqlCommand("sp_PrikaziStanja", Konekcija);
                Komanda.CommandType = CommandType.StoredProcedure;

                Konekcija.Open();
                SqlDataReader Reader = Komanda.ExecuteReader();
```

```
                if (Reader.HasRows)
                {
                    DataTable Tabela = new DataTable();
                    Tabela.Load(Reader);
                    dataGridView1.AutoSizeColumnsMode =
DataGridViewAutoSizeColumnsMode.AllCells;
                    dataGridView1.DataSource = Tabela;
                }
            }
        }

        private void btnUnos_Click(object sender, EventArgs e)
        {
            if (txtNaziv.Text == "")
            {
                MessageBox.Show("Popunite sva polja pre unosa.");
            }
            else
            {
                using (Konekcija = new SqlConnection(KonekcioniString))
                {
                    Komanda = new SqlCommand("sp_UnesiStanje", Konekcija);
                    Komanda.CommandType = CommandType.StoredProcedure;
                    Komanda.Parameters.Add("@NazivStanja", SqlDbType.NVarChar).Value =
txtNaziv.Text;

                    Konekcija.Open();
                    try
                    {
                        Komanda.ExecuteNonQuery();
                        MessageBox.Show("Stanje je uspešno uneto.", "Informacija",
MessageBoxButtons.OK, MessageBoxIcon.Information);
                    }
                    catch (Exception ex)
                    {
                        MessageBox.Show("Nastala je greška - " + ex.Message, "Greška",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                    }
                    finally
                    {
                        OsveziEkran();
                        Konekcija.Close();
                    }
                }
            }
        }

        private void btnNadji_Click(object sender, EventArgs e)
        {
            if (txtSifraStanja.Text == "")
            {
                MessageBox.Show("Unesite šifru stanja.");
            }
            else
            {
                using (Konekcija = new SqlConnection(KonekcioniString))
                {
                    Komanda = new SqlCommand("sp_VratiStanjePoSifri", Konekcija);
```

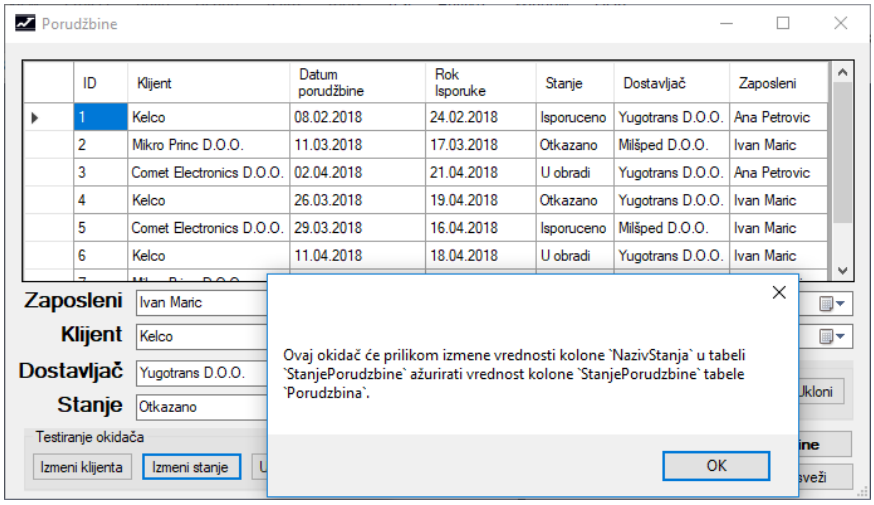
```
                    Komanda.CommandType = CommandType.StoredProcedure;
                    Komanda.Parameters.Add("@SifraStanja", SqlDbType.Int).Value =
int.Parse(txtSifraStanja.Text);
                    Komanda.Parameters.Add("@NazivStanja", SqlDbType.NVarChar, 50);
                    Komanda.Parameters["@NazivStanja"].Direction =
ParameterDirection.Output;
                    Konekcija.Open();
                    try
                    {
                        Komanda.ExecuteNonQuery();
                        txtNaziv.Text =
Komanda.Parameters["@NazivStanja"].Value.ToString();
                        if (txtNaziv.Text == "")
                            MessageBox.Show("Stanje sa zadatom šifrom nije pronađeno.");
                    }
                    catch (Exception ex)
                    {
                        MessageBox.Show("Nastala je greška - " + ex.Message, "Greška",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                    }
                    finally
                    {
                        Konekcija.Close();
                    }
                }
            }
        }

       private void btnIzmeni_Click(object sender, EventArgs e)
        {
            if (txtNaziv.Text == "" || txtSifraStanja.Text == "")
            {
                MessageBox.Show("Popunite sva polja pre ažuriranja.");
            }
            else
            {
                using (Konekcija = new SqlConnection(KonekcioniString))
                {
                    Komanda = new SqlCommand("sp_VratiStanjePoSifri", Konekcija);
                    Komanda.CommandType = CommandType.StoredProcedure;
                    Komanda.Parameters.Add("@SifraStanja", SqlDbType.Int).Value =
int.Parse(txtSifraStanja.Text);
                    Komanda.Parameters.Add("@NazivStanja", SqlDbType.NVarChar, 50);
                    Komanda.Parameters["@NazivStanja"].Direction =
ParameterDirection.Output;
                    Konekcija.Open();
                    try
                    {
                        Komanda.ExecuteNonQuery();
                        if (Komanda.Parameters["@NazivStanja"].Value.ToString() != "")
                        {
                            var PotvrdiIzmenu = MessageBox.Show("Potvrdite izmenu stanja
" + txtSifraStanja.Text + ".", "Potvrdite izmenu", MessageBoxButtons.OKCancel,
MessageBoxIcon.None);

                            if (PotvrdiIzmenu == DialogResult.OK)
                            {
                                Komanda = new SqlCommand("sp_AzurirajStanje", Konekcija);
```

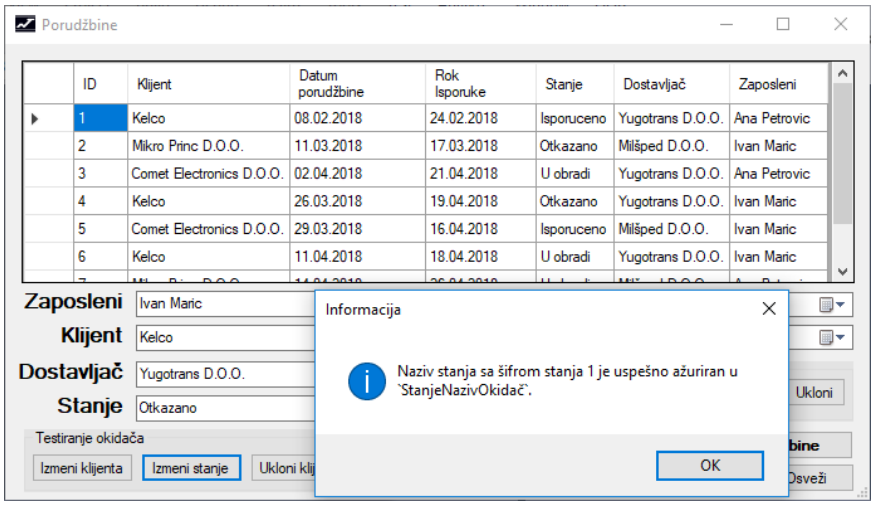
```
                                Komanda.CommandType = CommandType.StoredProcedure;
                                Komanda.Parameters.Add("@SifraStanja",
SqlDbType.Int).Value = int.Parse(txtSifraStanja.Text);
                                Komanda.Parameters.Add("@NazivStanja",
SqlDbType.NVarChar).Value = txtNaziv.Text;
                                try
                                {
                                    Komanda.ExecuteNonQuery();
                                    MessageBox.Show("Stanje je uspešno ažurirano.",
"Informacija", MessageBoxButtons.OK, MessageBoxIcon.Information);
                                }
                                catch (Exception ex)
                                {
                                    MessageBox.Show("Nastala je greška - " + ex.Message,
"Greška", MessageBoxButtons.OK, MessageBoxIcon.Error);
                                }
                            }
                        }
                        else
                        {
                            MessageBox.Show("Stanje sa zadatom šifrom nije pronađeno.");
                            txtSifraStanja.Text = "";
                            txtSifraStanja.Focus();
                        }
                    }
                    catch (Exception Ex)
                    {
                        MessageBox.Show("Dogodila se greška - " + Ex.Message, "Greška",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                    }
                    finally
                    {
                        Konekcija.Close();
                        OsveziEkran();
                    }
                }
            }
        }

        private void btnUkloni_Click(object sender, EventArgs e)
        {
            if (txtSifraStanja.Text == "")
            {
                MessageBox.Show("Unesite šifru stanja.");
            }
            else
            {
                using (Konekcija = new SqlConnection(KonekcioniString))
                {
                    Komanda = new SqlCommand("sp_VratiStanjePoSifri", Konekcija);
                    Komanda.CommandType = CommandType.StoredProcedure;
                    Komanda.Parameters.Add("@SifraStanja", SqlDbType.Int).Value =
int.Parse(txtSifraStanja.Text);
                    Komanda.Parameters.Add("@NazivStanja", SqlDbType.NVarChar, 50);
                    Komanda.Parameters["@NazivStanja"].Direction =
ParameterDirection.Output;
                    Konekcija.Open();
                    try
```

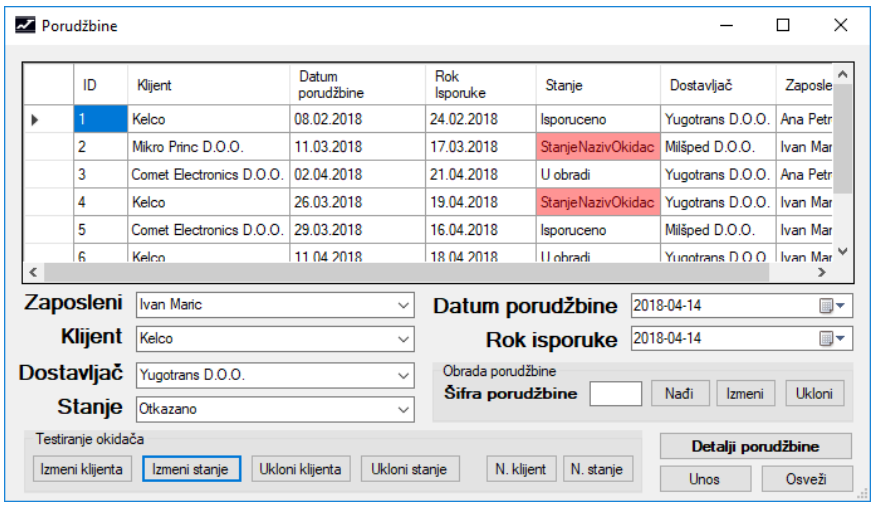
```
                    {
                        Komanda.ExecuteNonQuery();
                        if (Komanda.Parameters["@NazivStanja"].Value.ToString() != "")
                        {
                            var PotvrdiUklanjanje = MessageBox.Show("Potvrdite uklanjanje
stanja " + txtSifraStanja.Text + ".", "Potvrdite uklanjanje", MessageBoxButtons.OKCancel,
MessageBoxIcon.Warning);

                            if (PotvrdiUklanjanje == DialogResult.OK)
                            {
                                Komanda = new SqlCommand("sp_UkloniStanje", Konekcija);
                                Komanda.CommandType = CommandType.StoredProcedure;
                                Komanda.Parameters.Add("@SifraStanja",
SqlDbType.Int).Value = int.Parse(txtSifraStanja.Text);
                                try
                                {
                                    Komanda.ExecuteNonQuery();
                                    MessageBox.Show("Stanje je uspešno uklonjeno.",
"Informacija", MessageBoxButtons.OK, MessageBoxIcon.Information);
                                }
                                catch (Exception ex)
                                {
                                    MessageBox.Show("Nastala je greška - " + ex.Message,
"Greška", MessageBoxButtons.OK, MessageBoxIcon.Error);
                                }
                            }
                        }
                        else
                        {
                            MessageBox.Show("Stanje sa zadatom šifrom nije pronađeno.");
                            txtSifraStanja.Text = "";
                            txtSifraStanja.Focus();
                        }
                    }
                    catch (Exception Ex)
                    {
                        MessageBox.Show("Dogodila se greška - " + Ex.Message, "Greška",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                    }
                    finally
                    {
                        Konekcija.Close();
                        OsveziEkran();
                    }
                }
            }
        }

        private void btnOsvezi_Click(object sender, EventArgs e)
        {
            txtSifraStanja.Text = "";
            txtNaziv.Text = "";
        }
    }
}
```

7.5. Denormalizovana tabela

Implementaciju videti u poglavlju 7.3.1.2. (strana 21).





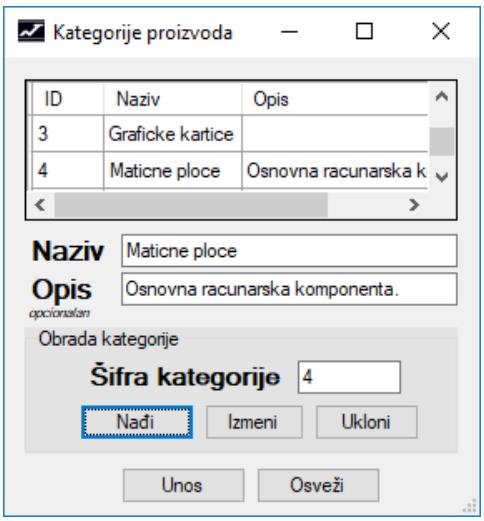


7.6. Diskonektovana arhitektura

Tabele korišćene :

- ***Kategorije proizvoda*** - *frmKategorije*
- ***Dostavljači*** - *frmDostavljaci*

7.6.1. frmKategorije



Izgled forme.

7.6.1.1. frmKategorije - Kod

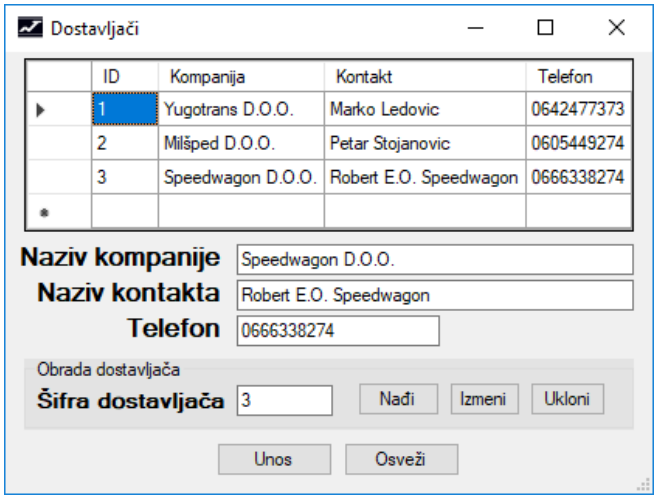
```
using …;
using System.Configuration;
using System.Data.SqlClient;

namespace NovaTehnika
{
    public partial class frmKategorije : Form
    {
        string KonekcioniString;
        SqlConnection Konekcija;
        SqlCommand Komanda;

        public frmKategorije()
        {
            KonekcioniString =
ConfigurationManager.ConnectionStrings["KonekcioniString"].ConnectionString;
            Konekcija = new SqlConnection(KonekcioniString);
            InitializeComponent();
        }

        private void frmKategorije_Load(object sender, EventArgs e)
        {
            OsveziEkran();
        }

        private void OsveziEkran()
        {
```

```
            SqlDataAdapter Adapter = new SqlDataAdapter("SELECT Kategorija.SifraKategorije AS
[ID], Kategorija.NazivKategorije AS [Naziv], Kategorija.Opis FROM Kategorija", Konekcija);
            DataSet KategorijaSet = new DataSet();
            DataTable KategorijaTabela = new DataTable();
            Adapter.Fill(KategorijaSet);
            Adapter.Fill(KategorijaTabela);
            dataGridView1.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.AllCells;
            dataGridView1.DataSource = KategorijaTabela;
        }

        private void btnUnos_Click(object sender, EventArgs e)
        {
            if(txtNaziv.Text == "")
            {
                MessageBox.Show("Morate uneti barem naziv kategorije pre unosa.");
            }
            else
            {
                using(Konekcija = new SqlConnection(KonekcioniString))
                {
                    if(txtOpis.Text == "")
                    {
                        Komanda = new SqlCommand("INSERT INTO Kategorija(NazivKategorije)
VALUES ('"+txtNaziv.Text+"');", Konekcija);
                    }
                    else
                    {
                        Komanda = new SqlCommand("INSERT INTO Kategorija(NazivKategorije,
Opis) VALUES ('" + txtNaziv.Text + "', '"+txtOpis.Text+"');", Konekcija);
                    }

                    SqlDataAdapter Adapter = new SqlDataAdapter();
                    Adapter.InsertCommand = Komanda;
                    Konekcija.Open();
                    try
                    {
                        Adapter.InsertCommand.ExecuteNonQuery();
                        MessageBox.Show("Kategorija je uspešno uneta.", "Informacija",
MessageBoxButtons.OK, MessageBoxIcon.Information);
                    }
                    catch (Exception ex)
                    {
                        MessageBox.Show("Nastala je greška - " + ex.Message, "Greška",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                    }
                    finally
                    {
                        OsveziEkran();
                        Konekcija.Close();
                    }
                }
            }
        }

        private void btnNadji_Click(object sender, EventArgs e)
        {
            if (txtSifraKategorije.Text == "")
            {
                MessageBox.Show("Unesite šifru kategorije.");
            }
            else
```

```
            {
                using (Konekcija = new SqlConnection(KonekcioniString))
                {
                    try
                    {
                        string SelectUpit = "SELECT NazivKategorije, Opis FROM Kategorija
WHERE SifraKategorije = " + int.Parse(txtSifraKategorije.Text);
                        SqlDataAdapter Adapter = new SqlDataAdapter(SelectUpit, Konekcija);
                        DataSet setKategorija = new DataSet();
                        Adapter.Fill(setKategorija, "Kategorija");
                        txtNaziv.Text =
setKategorija.Tables[0].Rows[0]["NazivKategorije"].ToString();
                        txtOpis.Text = setKategorija.Tables[0].Rows[0]["Opis"].ToString();
                    }
                    catch (Exception ex)
                    {
                        MessageBox.Show("Nastala je greška - " + ex.Message, "Greška",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                    }
                }
            }
        }

        private void btnIzmeni_Click(object sender, EventArgs e)
        {
            if (txtNaziv.Text == "" || txtSifraKategorije.Text == "")
            {
                MessageBox.Show("Unesite neophodne podatke.");
            }
            else
            {
                using (Konekcija = new SqlConnection(KonekcioniString))
                {
                    var PotvrdiIzmenu = MessageBox.Show("Potvrdite izmenu kategorije " +
txtSifraKategorije.Text + ".", "Potvrdite izmenu", MessageBoxButtons.OKCancel,
MessageBoxIcon.None);

                    if (PotvrdiIzmenu == DialogResult.OK)
                    {
                        if (txtOpis.Text == "")
                        {
                            Komanda = new SqlCommand("UPDATE Kategorija SET NazivKategorije =
'" + txtNaziv.Text + "' WHERE SifraKategorije =" + txtSifraKategorije.Text, Konekcija);
                        }
                        else
                        {
                            Komanda = new SqlCommand("UPDATE Kategorija SET NazivKategorije =
'" + txtNaziv.Text + "', Opis = '" + txtOpis.Text + "' WHERE SifraKategorije =" +
int.Parse(txtSifraKategorije.Text), Konekcija);
                        }

                        SqlDataAdapter Adapter = new SqlDataAdapter();
                        Adapter.UpdateCommand = Komanda;
                        Konekcija.Open();
                        try
                        {
                            Adapter.UpdateCommand.ExecuteNonQuery();
                            MessageBox.Show("Kategorija je uspešno ažurirana.", "Informacija",
MessageBoxButtons.OK, MessageBoxIcon.Information);
                        }
                        catch (Exception ex)
```

```
                        {
                            MessageBox.Show("Nastala je greška - " + ex.Message, "Greška",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                        }
                        finally
                        {
                            OsveziEkran();
                            Konekcija.Close();
                        }
                    }
                }
            }
        }

        private void btnUkloni_Click(object sender, EventArgs e)
        {
            if (txtSifraKategorije.Text == "")
            {
                MessageBox.Show("Unesite šifru kategorije.");
            }
            else
            {
                using (Konekcija = new SqlConnection(KonekcioniString))
                {
                    var PotvrdiUklanjanje = MessageBox.Show("Potvrdite uklanjanje kategorije "
+ txtSifraKategorije.Text + ".", "Potvrdite uklanjanje", MessageBoxButtons.OKCancel,
MessageBoxIcon.Warning);

                    if (PotvrdiUklanjanje == DialogResult.OK)
                    {
                        Komanda = new SqlCommand("DELETE FROM Kategorija WHERE SifraKategorije
=" + int.Parse(txtSifraKategorije.Text), Konekcija);
                        SqlDataAdapter Adapter = new SqlDataAdapter();
                        Adapter.DeleteCommand = Komanda;
                        Konekcija.Open();
                        try
                        {
                            Adapter.DeleteCommand.ExecuteNonQuery();
                            MessageBox.Show("Kategorija je uspešno uklonjena.", "Informacija",
MessageBoxButtons.OK, MessageBoxIcon.Information);
                        }
                        catch (Exception ex)
                        {
                            MessageBox.Show("Nastala je greška - " + ex.Message, "Greška",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                        }
                        finally
                        {
                            OsveziEkran();
                            Konekcija.Close();
                        }
                    }
                }
            }
        }
        private void btnOsvezi_Click(object sender, EventArgs e)
        {
            txtNaziv.Text = "";
            txtOpis.Text = "";
            txtSifraKategorije.Text = "";
        } }}
```

7.6.2. frmDostavljaci



Izgled forme.

7.6.2.1. frmDostavljaci - Kod

```
using …;
using System.Configuration;
using System.Data.SqlClient;

namespace NovaTehnika
{
    public partial class frmDostavljaci : Form
    {
        string KonekcioniString;
        SqlConnection Konekcija;
        SqlCommand Komanda;

        public frmDostavljaci()
        {
            KonekcioniString =
ConfigurationManager.ConnectionStrings["KonekcioniString"].ConnectionString;
            Konekcija = new SqlConnection(KonekcioniString);
            InitializeComponent();
        }

        private void frmDostavljaci_Load(object sender, EventArgs e)
        {
            OsveziEkran();
        }

       private void OsveziEkran()
        {
            SqlDataAdapter Adapter = new SqlDataAdapter("SELECT
Dostavljac.SifraDostavljaca AS [ID], Dostavljac.NazivKompanije AS [Kompanija],
Dostavljac.NazivKontakta AS [Kontakt], Dostavljac.Telefon FROM Dostavljac", Konekcija);
            DataSet DostavljacSet = new DataSet();
            DataTable DostavljacTabela = new DataTable();
            Adapter.Fill(DostavljacSet);
            Adapter.Fill(DostavljacTabela);
```

```
            dataGridView1.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.AllCells;
            dataGridView1.DataSource = DostavljacTabela;
        }

        private void btnUnos_Click(object sender, EventArgs e)
        {
            if (txtNazivKompanije.Text == "" || txtNazivKontakta.Text == "" ||
txtTelefon.Text == "")
            {
                MessageBox.Show("Unesite sve podatke.");
            }
            else
            {
                using (Konekcija = new SqlConnection(KonekcioniString))
                {
                    Komanda = new SqlCommand("INSERT INTO Dostavljac(NazivKompanije,
NazivKontakta, Telefon) VALUES ('" + txtNazivKompanije.Text + "', '" +
txtNazivKontakta.Text + "', '"+txtTelefon.Text+"');", Konekcija);
                    SqlDataAdapter Adapter = new SqlDataAdapter();
                    Adapter.InsertCommand = Komanda;
                    Konekcija.Open();
                    try
                    {
                        Adapter.InsertCommand.ExecuteNonQuery();
                        MessageBox.Show("Dostavljač je uspešno unet.", "Informacija",
MessageBoxButtons.OK, MessageBoxIcon.Information);
                    }
                    catch (Exception ex)
                    {
                        MessageBox.Show("Nastala je greška - " + ex.Message, "Greška",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                    }
                    finally
                    {
                        OsveziEkran();
                        Konekcija.Close();
                    }
                }
            }
        }

        private void btnNadji_Click(object sender, EventArgs e)
        {
            if (txtSifraDostavljaca.Text == "")
            {
                MessageBox.Show("Unesite šifru dostavljača.");
            }
            else
            {
                using (Konekcija = new SqlConnection(KonekcioniString))
                {
                    try
                    {
                        string SelectUpit = "SELECT NazivKompanije, NazivKontakta,
Telefon FROM Dostavljac WHERE SifraDostavljaca = " + int.Parse(txtSifraDostavljaca.Text);
                        SqlDataAdapter Adapter = new SqlDataAdapter(SelectUpit,
Konekcija);
                        DataSet setDostavljac = new DataSet();
```

```
                        Adapter.Fill(setDostavljac, "Dostavljac");
                        txtNazivKompanije.Text =
setDostavljac.Tables[0].Rows[0]["NazivKompanije"].ToString();
                        txtNazivKontakta.Text =
setDostavljac.Tables[0].Rows[0]["NazivKontakta"].ToString();
                        txtTelefon.Text =
setDostavljac.Tables[0].Rows[0]["Telefon"].ToString();
                    }
                    catch (Exception ex)
                    {
                        MessageBox.Show("Nastala je greška - " + ex.Message, "Greška",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                    }
                }
            }
        }

       private void btnIzmeni_Click(object sender, EventArgs e)
        {
            if (txtNazivKompanije.Text == "" || txtNazivKontakta.Text == "" ||
txtTelefon.Text == "" || txtSifraDostavljaca.Text == "")
            {
                MessageBox.Show("Unesite neophodne podatke.");
            }
            else
            {
                using (Konekcija = new SqlConnection(KonekcioniString))
                {
                    var PotvrdiIzmenu = MessageBox.Show("Potvrdite izmenu dostavljača " +
txtSifraDostavljaca.Text + ".", "Potvrdite izmenu", MessageBoxButtons.OKCancel,
MessageBoxIcon.None);

                    if (PotvrdiIzmenu == DialogResult.OK)
                    {
                        Komanda = new SqlCommand("UPDATE Dostavljac SET NazivKompanije =
'" + txtNazivKompanije.Text + "', NazivKontakta = '" + txtNazivKontakta.Text + "',
Telefon = '" + txtTelefon.Text + "' WHERE SifraDostavljaca =" +
int.Parse(txtSifraDostavljaca.Text), Konekcija);
                        SqlDataAdapter Adapter = new SqlDataAdapter();
                        Adapter.UpdateCommand = Komanda;
                        Konekcija.Open();
                        try
                        {
                            Adapter.UpdateCommand.ExecuteNonQuery();
                            MessageBox.Show("Dostavljač je uspešno ažuriran.",
"Informacija", MessageBoxButtons.OK, MessageBoxIcon.Information);
                        }
                        catch (Exception ex)
                        {
                            MessageBox.Show("Nastala je greška - " + ex.Message,
"Greška", MessageBoxButtons.OK, MessageBoxIcon.Error);
                        }
                        finally
                        {
                            OsveziEkran();
                            Konekcija.Close();
                        }
                    }
```

```
                }
            }
        }

        private void btnUkloni_Click(object sender, EventArgs e)
        {
            if (txtSifraDostavljaca.Text == "")
            {
                MessageBox.Show("Unesite šifru dostavljača.");
            }
            else
            {
                using (Konekcija = new SqlConnection(KonekcioniString))
                {
                    var PotvrdiUklanjanje = MessageBox.Show("Potvrdite uklanjanje
dostavljača " + txtSifraDostavljaca.Text + ".", "Potvrdite uklanjanje",
MessageBoxButtons.OKCancel, MessageBoxIcon.Warning);

                    if (PotvrdiUklanjanje == DialogResult.OK)
                    {
                        Komanda = new SqlCommand("DELETE FROM Dostavljac WHERE
SifraDostavljaca =" + int.Parse(txtSifraDostavljaca.Text), Konekcija);
                        SqlDataAdapter Adapter = new SqlDataAdapter();
                        Adapter.DeleteCommand = Komanda;
                        Konekcija.Open();
                        try
                        {
                            Adapter.DeleteCommand.ExecuteNonQuery();
                            MessageBox.Show("Dostavljač je uspešno uklonjen.",
"Informacija", MessageBoxButtons.OK, MessageBoxIcon.Information);
                        }
                        catch (Exception ex)
                        {
                            MessageBox.Show("Nastala je greška - " + ex.Message,
"Greška", MessageBoxButtons.OK, MessageBoxIcon.Error);
                        }
                        finally
                        {
                            OsveziEkran();
                            Konekcija.Close();
                        }
                    }
                }
            }
        }

        private void btnOsvezi_Click(object sender, EventArgs e)
        {
            txtNazivKompanije.Text = "";
            txtNazivKontakta.Text = "";
            txtSifraDostavljaca.Text = "";
            txtTelefon.Text = "";
        }
    }
}
```