



AUTO SERVIS

WPF aplikacija

-projektna dokumentacija-

Autor: Jovan Milošević, IT47/2018

Sadržaj

1.	OPIS REALNOG SISTEMA.....	4
2.	ZADATAK	5
3.	OPIS TEHNOLOGIJA.....	5
4.	UML.....	6
4.1.	Dijagram slučaja upotrebe	6
4.2.	Dijagram klasa	9
4.3.	Dijagram sekvenci.....	10
5.	BAZA PODATAKA.....	10
5.1.	Kreiranje baze podataka	10
5.2.	Upiti i DML naredbe.....	14
5.2.1.	Upiti	14
5.2.2.	DML naredbe	14
6.	RAZVOJ WPF APLIKACIJE.....	15
6.1.	Arhitektura aplikacije	15
6.1.1.	Interfejs IData	15
6.1.2.	Klasa AutoServisData	15
6.1.3.	Klasa Konekcija	19
6.1.4.	Modeli.....	20
6.1.5.	Prozori (Windows)	72
6.1.6.	Stranice (Pages)	75
6.2.	Dizajn korisničkog interfejsa	114
6.2.1.	App.xaml.....	114
6.2.2.	frmLogin.xaml	116
6.2.3.	frmPocetna.xaml	116
6.2.4.	pgMarke.xaml	117
6.2.5.	pgZaposleni.xaml	117
6.2.6.	pgVlasnici.xaml	118
6.2.7.	pgVozila.xaml.....	118
6.2.8.	pgRadniNalozi.xaml	119
6.2.9.	pgRadovi.xaml	119
6.2.10.	pgFakture.xaml	120
6.2.11.	pgGarancije.xaml	120
7.	TESTIRANJE APLIKACIJE.....	121
8.	ZAKLJUČAK.....	122

Bitnije stranice:

15-19: AutoServisData

20-21: Modeli

25-27: Osnovne metode iz modela

40-41: Metode u modelu RadniNalog

44-45: Metode u modelu Faktura

48-49: Metode u modelu Garancija

52-53: Metode u modelu NaruceniRadovi

71-72: Prozori

80-81: Način korišćenja metoda iz modela

93-94: Čitač saobraćajne dozvole

97-98: Primer korišćenja metoda iz modela

114-115: Dizajn korisničkog interfejsa

116-120: Slike aplikacije

121-122: Testiranje

122-123: Zaključak

1. Opis realnog sistema

Auto servis kao realan sistem predstavlja skup više ljudi različitih profesija čije delovanje mora biti usaglašeno prema definisanim poslovnim procesima da bi se postigli maksimalna efikasnost i efektivnost, a samim tim i pružila najbolja usluga mušteriji. Kada postoje jasno definisani poslovni procesi, vrlo je korisno razviti i implementirati informacioni sistem koji će da obuhvati bar neke poslovne procese, i time dodatno poboljša način poslovanja realnog sistema.

Aplikacija treba da pruži optimalno rešenje za problem evidencije zaposlenih, vozila, vlasnika vozila itd. na jednom mestu, kao i da olakša izdavanje računa, garancija i menadžment celokupnom administracijom. Korisnički interfejs mora da bude intuitivan, da se sve nalazi na svom mestu, uz brzu i laku pretragu svih podataka. Aplikacija treba da bude brza i efikasna, a sa druge strane ne sme da iziskuje previše hardverskih resursa. Dodatno se mogu razviti nove funkcionalnosti, kao što su pregled podataka sa mobilnog telefona ili tablet uređaja, pregled podataka na veb aplikaciji, zatim dodatne funkcije za još pomoći pri administraciji, kao što su evidencija uplata kupaca, evidencija dugovanja prema dobavljačima itd.

Ideja je da proces prijema vozila počinje evidencijom podataka o vozilu i vlasniku vozila u informacioni sistem, čime bi se izbegli problemi koji nastaju kada se ovi, veoma bitni podaci, prenose verbalnom komunikacijom, što je i dalje čest slučaj u današnjim auto servisima. Evidencija ovih podataka može da se vrši putem čitanja podataka sa čipa saobraćajne dozvole i lične karte, ili da se podaci unose ručno u sistem. Nakon toga, zaposleno lice evidentira kvarove i željene usluge koje vlasnik vozila izrazi, i otvara i štampa radni nalog koji vlasnik vozila mora da potpiše. Sledi otklanjanje kvarova i izvršavanje usluga ispisanih na radnom nalogu. Kada su kvarovi otklonjeni i svi radovi provereni, vrši se evidencija ugrađenih rezervnih delova i svih izvršenih usluga, nakon čega se štampa faktura i izdaje garancija.

Vrlo je bitno da se ovaj redosled koraka prati, bez izuzetaka.

2. Zadatak

Ovim projektom obuhvaćen je deo koji se tiče unosa, skladištenja, pretrage podataka, uz mogućnost čitanja podataka sa čipa saobraćajne dozvole. Deo koji je izuzet je štampanje radnog naloga, fakture i ostalih izveštaja i pregleda, zbog ograničenog vremena za razvoj aplikacije. Uz odgovarajuće resurse, aplikacija se može razviti da prati sve idejne korake uz dodatak još korisnih funkcionalnosti, ali je za ovaj projekat više nego dovoljno implementirati neke osnovne funkcije.

Autorova želja da se cela aplikacija razvija po **MVVM** dizajnerskom obrascu, uz potpunu moć **WPF-a**, ostaće ovaj put neispunjena zbog nedostatka prekopotrebnog vremena.

3. Opis tehnologija

Za kreiranje baze podataka korišćen je **Microsoft SQL Server Express Edition** iz 2019. godine. Za razvoj aplikacije korišćen je programski jezik **C#** uz **WPF** grafički podsistem za kreiranje i održavanje interfejsa. Aplikacija je razvijana u integrisanom razvojnom okruženju **Visual Studio**, i to u verziji iz 2015. godine.

Microsoft SQL Server je sistem za upravljanje relacionim bazama podataka razvijen od strane kompanije **Microsoft**. Primarna uloga mu je skladištenje i upravljanje podacima koje zahtevaju ostali softveri koji mogu biti pokrenuti na istom računaru, ili mogu pristupati serveru sa udaljenih računara putem interneta.

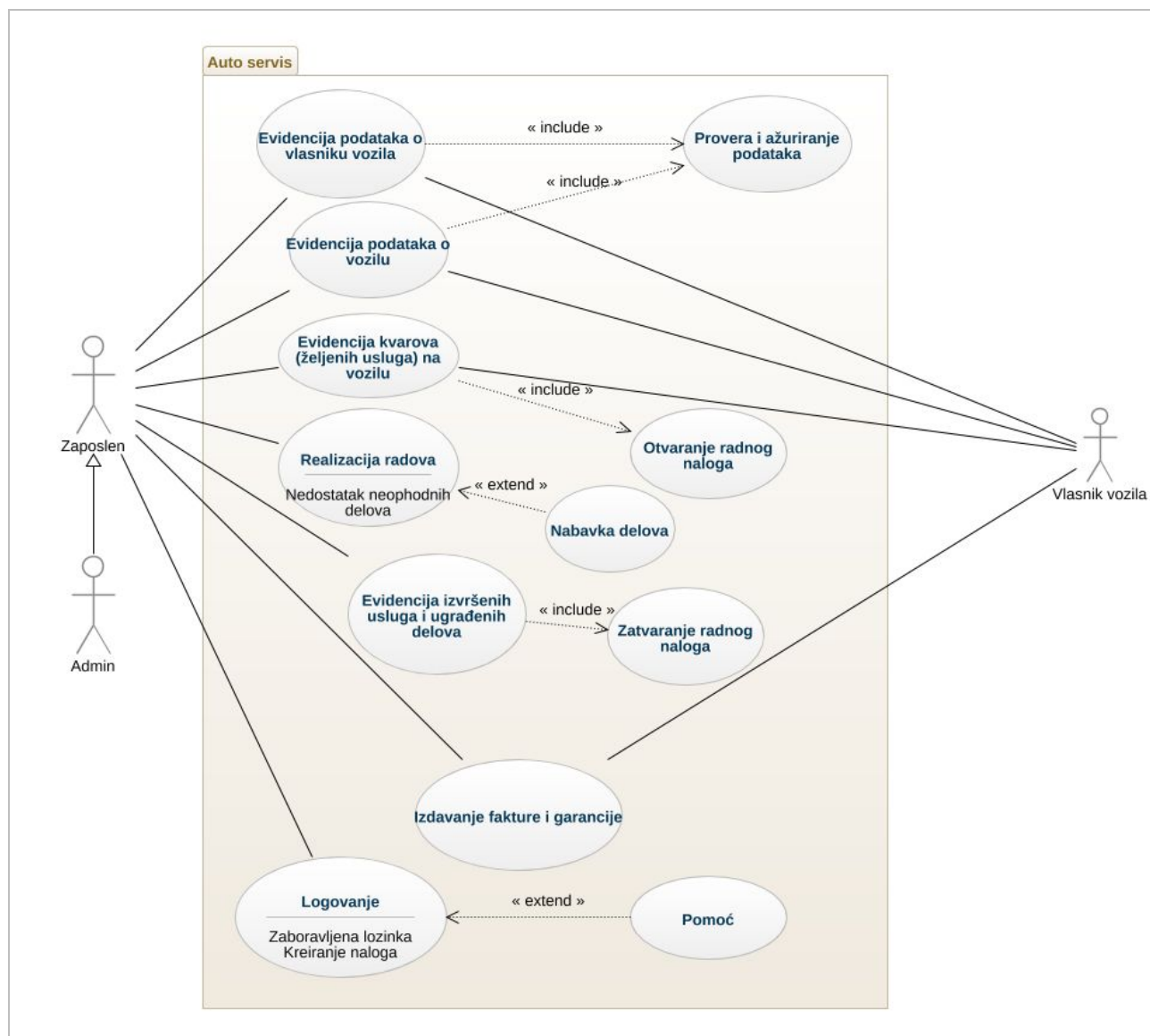
C# je objektno-orijentisani programski jezik opšte namene, višestruke paradigme razvijen, takođe, od strane kompanije **Microsoft**. Deo je **.NET** programske platforme.

Windows Presentation Foundation (WPF) je prezentacioni (**user interface**) podsistem **Windows**-ovog programskog modela koji se koristi za izradu vizuelno bogatih klijentskih aplikacija. **WPF** kombinuje aplikacije korisničkog interfejsa, 2D i 3D grafike, dokumente, animacije i multimediju u jedinstveni paket koji je uveden sa verzijom 3.0 **.NET framework-a**. Koristi vektorsku grafiku koja sa modernim grafičkim karticama i procesorima čini korisnički interfejs bržim i pruža nezavisnost od rezolucije monitora. Jedna od važnijih karakteristika **WPF-a** je ta što odvaja prikaz korisničkog interfejsa od definisanja njegovog ponašanja. Prikaz je specifikiran **XAML** jezikom.

Microsoft Visual Studio je integrisano razvojno okruženje za razvoj aplikacija u jezicima podržanim **.NET** programskom platformom.

4. UML

4.1. Dijagram slučaja upotrebe



Slučaj upotrebe: Evidencija podataka o vlasniku vozila

Kratak opis: Evidencija podataka o vlasniku vozila

Učesnici: Zaposleno lice, Vlasnik vozila

Uslovi koji moraju biti zadovoljeni pre izvršenja: Vlasnik vozila poseduje ličnu kartu

Opis: Zaposleno lice učitava podatke o vlasniku vozila sa čipa njegove lične karte i smešta ih u sistem. Sistem proverava i po potrebi ažurira podatke.

Izuzeci:

[Lična karta ne sadrži elektronski čip] Ukoliko ne postoji čip na ličnoj karti, zaposleno lice ručno unosi podatke u sistem.

Uslovi koji moraju biti zadovoljeni nakon izvršenja: U sistem su uneti (ažurirani) podaci o vlasniku vozila

Slučaj upotrebe: Evidencija podataka o vozilu

Kratak opis: Evidencija podataka o vozilu

Učesnici: Zaposleno lice, Vlasnik vozila

Uslovi koji moraju biti zadovoljeni pre izvršenja: Vlasnik vozila poseduje saobraćajnu dozvolu

Opis: Zaposleno lice učitava podatke sa čipa saobraćajne dozvole u sistem. Sistem proverava i po potrebi ažurira podatke.

Izuzeci:

[Aplikacija ne može da očita čip sa saobraćajne dozvole] Ukoliko čip ne može da bude očitán, zaposleno lice ručno unosi podatke u sistem.

Uslovi koji moraju biti zadovoljeni nakon izvršenja: U sistem su uneti (ažurirani) podaci o vozilu.

Slučaj upotrebe: Evidencija kvarova (željenih usluga) na vozilu

Kratak opis: Evidencija željenih usluga vlasnika vozila

Učesnici: Zaposleno lice, Vlasnik vozila

Uslovi koji moraju biti zadovoljeni pre izvršenja: U sistem su uneti podaci o vlasniku i vozilu

Opis: Zaposleno lice evidentira usluge i kvarove koje mu vlasnik vozila opiše i zatim otvara i štampa radni nalog. Vlasnik vozila mora potpisati odštampani nalog sa evidentiranim uslugama i mora biti saglasan sa mogućim nepredviđenim radovima proisteklim iz prepravki na vozilu ili zbog neadekvatnog održavanja.

Izuzeci:

[Vlasnik ne želi da se usaglasí sa nepredviđenim radovima] Ukoliko vlasnik ne želi da se usaglasí sa nepredviđenim radovima, vozilo neće biti primljeno u servis.

Uslovi koji moraju biti zadovoljeni nakon izvršenja: U sistem su unete informacije o željenim uslugama vlasnika vozila i otvoren je radni nalog.

Slučaj upotrebe: Realizacija radova

Kratak opis: Izvršavanje radova na vozilu

Učesnici: Zaposleno lice

Uslovi koji moraju biti zadovoljeni pre izvršenja: U sistem su uneti svi neophodni podaci o vlasniku, vozilu i željenim radovima

Opis: Zaposleno lice obavlja radove na vozilu u skladu sa evidentiranim željenim uslugama vlasnika vozila, takođe vodeći evidenciju o količini utrošenog vremena za potrebne radove.

Izuzeci:

[Nedostatak neophodnih rezervnih delova] Ukoliko nedostaju rezervni delovi vrši se njihova nabavka sa preciznom evidencijom naziva i cene.

Uslovi koji moraju biti zadovoljeni nakon izvršenja: Na vozilu su realizovani svi potrebni radovi i evidentirana je količina utrošenog vremena.

Slučaj upotrebe: Evidencija izvršenih usluga i ugrađenih delova

Kratak opis: Evidentiranje svih izvršenih usluga i svih rezervnih delova koji su ugrađeni

Učesnici: Zaposleno lice

Uslovi koji moraju biti zadovoljeni pre izvršenja: U sistem su uneti svi neophodni podaci i realizovani su i provereni svi radovi na vozilu

Opis: Zaposleno lice evidentira sve izvršene radove i ugrađene rezervne delove sa količinom i cenom, nakon čega zatvara i štampa zatvoren radni nalog.

Izuzeci: Nema

Uslovi koji moraju biti zadovoljeni nakon izvršenja: Evidentirani su svi realizovani radovi i zatvoren je radni nalog za to vozilo.

Slučaj upotrebe: Izdavanje fakture i garancije

Kratak opis: Izdavanje računa i garancije vlasniku vozila

Učesnici: Zaposleno lice, Vlasnik vozila

Uslovi koji moraju biti zadovoljeni pre izvršenja: U sistemu postoje neophodni podaci, realizovani su i evidentirani svi radovi, zatvoren je radni nalog

Opis: Zaposleno lice štampa fakturu sa precizno navedenim cenama usluga i ugrađenih rezervnih delova i izdaje je vlasniku zajedno sa garancijom (ili garancijama, ukoliko više usluga/delova zahteva više različitih garancija).

Izuzeci:

[Vlasnik vozila nije zadovoljan izvršenim radovima] Ukoliko vlasnik nije zadovoljan izvršenim radovima, faktura neće biti izdata dok se ne izvrše neophodne prepravke ili dok se ne umanja cena.

[Ne postoji garancija za određene usluge/delove] Ukoliko za određene usluge/delove ne postoji garancija, onda se vlasniku vozila izdaje samo faktura.

Uslovi koji moraju biti zadovoljeni nakon izvršenja: Vlasnik vozila je preuzeo svoje vozilo zajedno sa fakturom i garancijom.

Slučaj upotrebe: Logovanje

Kratak opis: Logovanje zaposlenih na sistem

Učesnici: Zaposleno lice

Uslovi koji moraju biti zadovoljeni pre izvršenja: Zaposleno lice poseduje nalog na sistemu

Opis: Zaposleno lice se loguje na sistem putem svog korisničkog imena i lozinke.

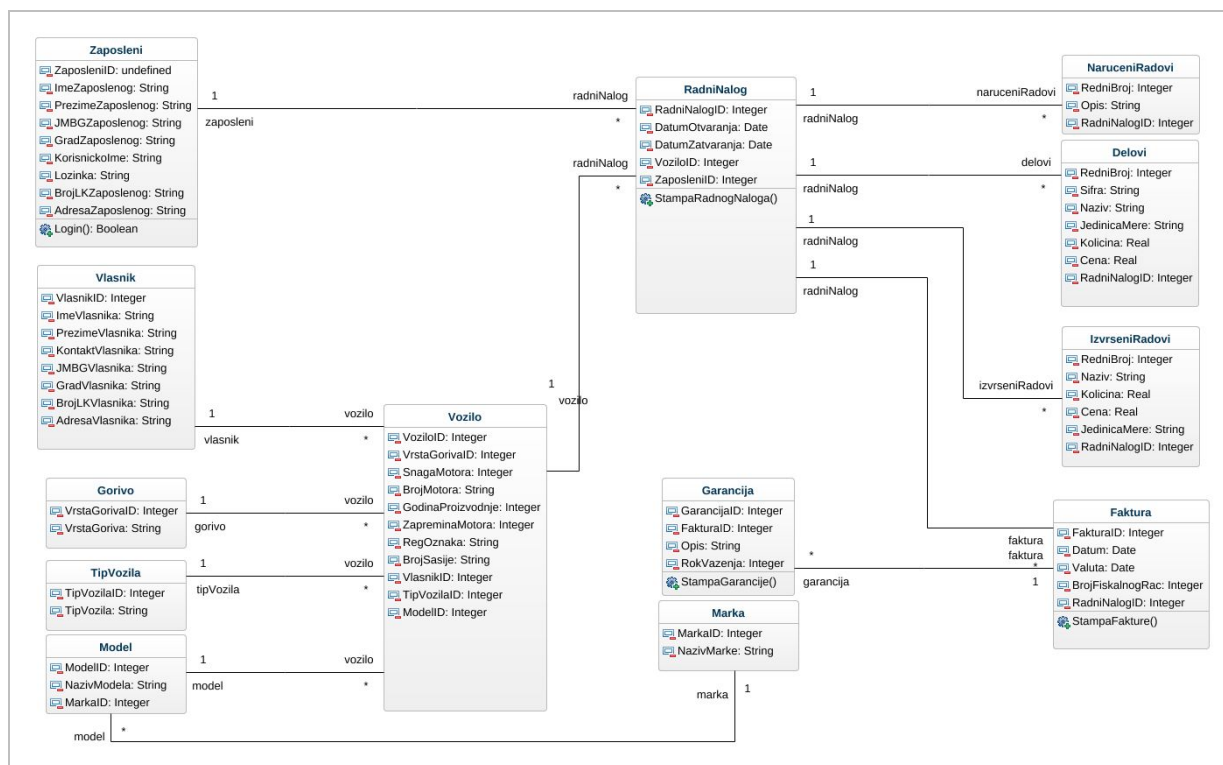
Izuzeci:

[Zaboravljena lozinka] Ukoliko je zaposleno lice zaboravilo lozinku, mora je prvo restartovati

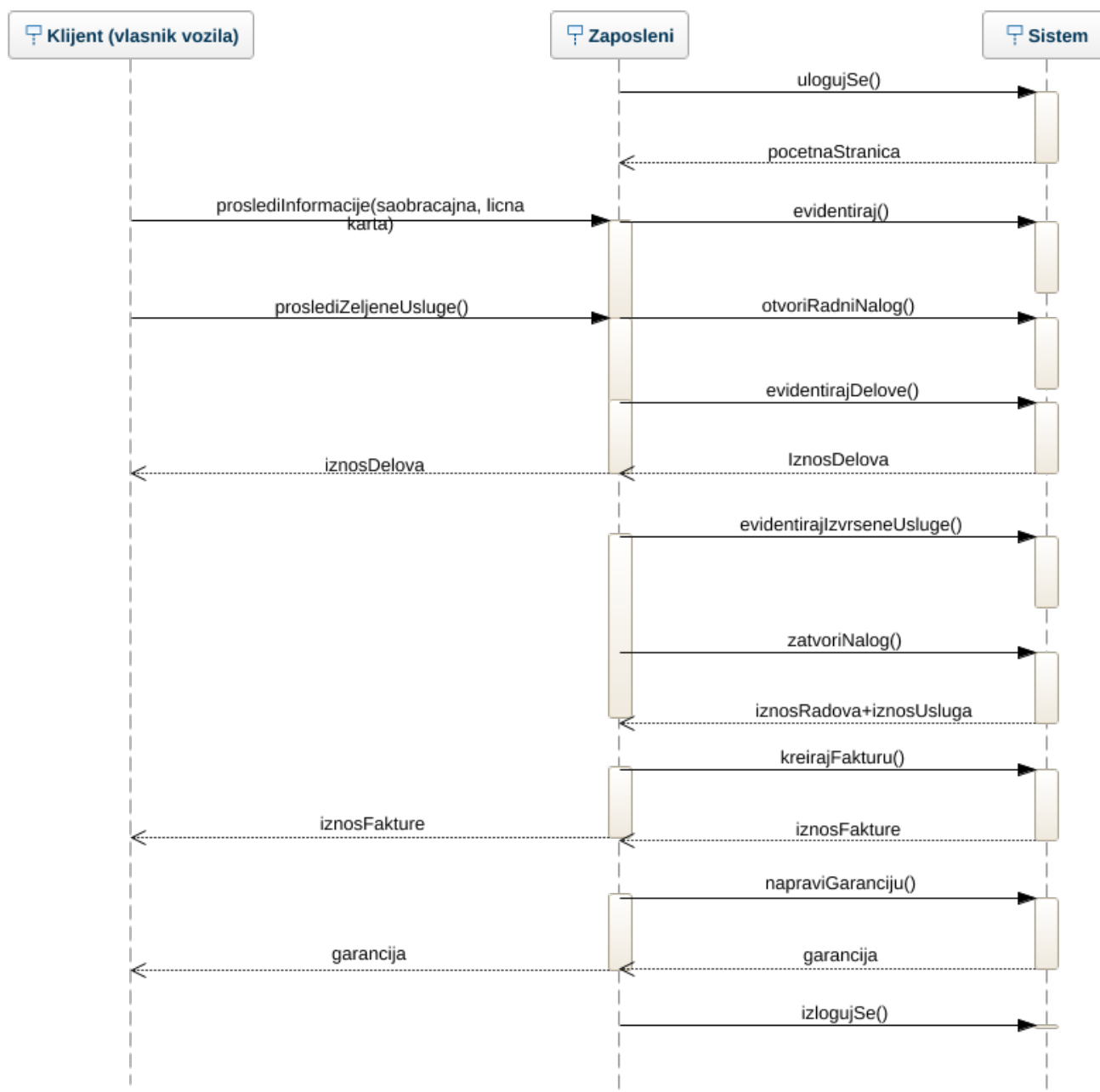
[Ne postoji nalog] Ukoliko zaposleno lice nema nalog na sistemu, mora ga prvo kreirati

Uslovi koji moraju biti zadovoljeni nakon izvršenja: Zaposleno lice je ulogovano na sistem.

4.2. Dijagram klasa



4.3. Dijagram sekvenci



5. Baza podataka

5.1. Kreiranje baze podataka

U bazi je kreirano 13 tabela. 10 tabela ima kolone u kojima je primarni ključ jedinstvena kolona, sa **Identity increment** i **Identity seed** osobinama podešenim na 1. Tri tabele imaju primarni ključ definisan kao kombinacija dve kolone, tačnije kolone **RedniBroj** i **RadniNalogID**. Detaljna podešavanja tabela i primarnih ključeva slede na slikama.

dbo.tblVozilo
Columns
VoziloID (PK, int, not null)
VrstaGorivaID (FK, int, not null)
SnagaMotora (int, null)
BrojMotora (nvarchar(50), null)
GodinaProizvodnje (int, null)
ZapreminaMotora (int, null)
RegOznaka (nvarchar(10), null)
BrojSasije (nvarchar(17), null)
VlasnikID (FK, int, not null)
TipVozilaID (FK, int, not null)
ModelID (FK, int, not null)
Keys
PK_tblVozilo
FK_tblVozilo_tblGorivo
FK_tblVozilo_tblModel
FK_tblVozilo_tblTipVozila
FK_tblVozilo_tblVlasnik
Constraints
Triggers
Indexes
PK_tblVozilo (Clustered)
Statistics

Tabela **tblVozilo**

dbo.tblRadniNalog
Columns
RadniNalogID (PK, int, not null)
DatumOtvaranja (date, not null)
DatumZatvaranja (date, null)
VoziloID (FK, int, not null)
ZaposleniID (FK, int, not null)
Keys
PK_tblRadniNalog
FK_tblRadniNalog_tblVozilo
FK_tblRadniNalog_tblZaposleni
Constraints
Triggers
Indexes
PK_tblRadniNalog (Clustered)
Statistics

Tabela **tblRadniNalog**

dbo.tblNaruceniRadovi
Columns
RedniBroj (PK, int, not null)
Opis (nvarchar(100), null)
RadniNalogID (PK, FK, int, not null)
Keys
PK_tblNaruceniRadovi
FK_tblNaruceniRadovi_tblRadniNalog
Constraints
Triggers
Indexes
PK_tblNaruceniRadovi (Clustered)
Statistics

Tabela **tblNaruceniRadovi**

dbo.tblIzvrzeniRadovi
Columns
RedniBroj (PK, int, not null)
Naziv (nvarchar(100), not null)
Kolicina (numeric(10,3), not null)
Cena (numeric(10,3), not null)
JedinicaMere (nvarchar(20), null)
RadniNalogID (PK, FK, int, not null)
Keys
PK_tblIzvrzeniRadovi
FK_tblIzvrzeniRadovi_tblIzvrzeniRadovi
Constraints
Triggers
Indexes
PK_tblIzvrzeniRadovi (Clustered)
Statistics

Tabela **tblIzvrzeniRadovi**

dbo.tblFaktura
Columns
FakturalID (PK, int, not null)
Datum (date, not null)
Valuta (date, not null)
BrojFiskalnogRacuna (int, not null)
RadniNalogID (FK, int, not null)
Keys
PK_tblFaktura
FK_tblFaktura_tblRadniNalog
Constraints
Triggers
Indexes
PK_tblFaktura (Clustered)
Statistics

Tabela **tblFaktura**

dbo.tblGarancija
Columns
GarancijaID (PK, int, not null)
FakturalID (FK, int, not null)
Opis (nvarchar(100), null)
RokVazenja (int, not null)
Keys
PK_tblGarancija
FK_tblGarancija_tblFaktura
Constraints
Triggers
Indexes
PK_tblGarancija (Clustered)
Statistics

Tabela **tblGarancija**

dbo.tblDelovi
Columns
RedniBroj (PK, int, not null)
Sifra (nvarchar(20), null)
Naziv (nvarchar(100), not null)
JedinicaMere (nvarchar(20), null)
Kolicina (numeric(10,3), not null)
Cena (numeric(10,2), not null)
RadniNalogID (PK, FK, int, not null)
Keys
PK_tblDelovi
FK_tblDelovi_tblRadniNalog
Constraints
Triggers
Indexes
PK_tblDelovi (Clustered)
Statistics

Tabela **tblDelovi**

AutoServis
Database Diagrams
Tables
System Tables
FileTables
External Tables
Graph Tables
dbo.tblDelovi
dbo.tblFaktura
dbo.tblGarancija
dbo.tblGorivo
dbo.tblIzvršeniRadovi
dbo.tblMarka
dbo.tblModel
dbo.tblNaruceniRadovi
dbo.tblRadniNalog
dbo.tblTipVozila
dbo.tblVlasnik
dbo.tblVozilo
dbo.tblZaposleni

Schema cele baze sa svim tabelama

5.2. Upiti i DML naredbe

5.2.1. Upiti

Svi upiti koji su korišćeni za prikaz podataka mogu se videti u narednim stranicama, tačnije code snippet-ima iz delova aplikacije. U najvećem broju slučajeva korišćeni su parametrizovani upiti da bi se izbegle moguće greške i izuzeci.

Primer jednog od upita:

```
SELECT tblGarancija.GarancijaID as 'ID garancije',  
tblFaktura.FakturaID as 'ID fakture',  
tblVlasnik.ImeVlasnika + ' ' + tblVlasnik.PrezimeVlasnika as 'Kupac',  
tblGarancija.Opis,  
CONVERT(VARCHAR(10), (DATEADD(month, tblGarancija.RokVazenja, tblFaktura.Datum)), 103)  
as 'Važi do datuma:'  
FROM tblGarancija join tblFaktura on tblGarancija.FakturaID = tblFaktura.FakturaID  
join tblRadniNalog on tblFaktura.RadniNalogID = tblRadniNalog.RadniNalogID  
join tblVozilo on tblRadniNalog.VoziloID = tblVozilo.VoziloID  
join tblVlasnik on tblVozilo.VlasnikID = tblVlasnik.VlasnikID;
```

5.2.2. DML naredbe

DML naredbe korišćene prilikom kreiranja baze podataka jesu sve 4 vrste **DML** naredbi (**SELECT**, **INSERT**, **UPDATE**, **DELETE**) zarad testiranja baze i tabela. U tabelama **tblGorivo** i **tblTipVozila** izvršene su **INSERT** naredbe podataka, jer se u te tabele neće čuvati i ažurirati podaci direktno iz aplikacije. Ostale naredbe nisu sačuvane, a podaci su obrisani kada su počeli da se unose i modifikuju iz aplikacije.

5.2.2.1. tblGorivo - INSERT

```
INSERT INTO tblGorivo(VrstaGoriva) VALUES ('Benzin');  
INSERT INTO tblGorivo(VrstaGoriva) VALUES ('Dizel');  
INSERT INTO tblGorivo(VrstaGoriva) VALUES ('TNG');  
INSERT INTO tblGorivo(VrstaGoriva) VALUES ('Hibrid');  
INSERT INTO tblGorivo(VrstaGoriva) VALUES ('Električna baterija');
```

5.2.2.2. tblTipVozila - INSERT

```
INSERT INTO tblTipVozila VALUES ('Putničko vozilo');  
INSERT INTO tblTipVozila VALUES ('Teretno vozilo');  
INSERT INTO tblTipVozila VALUES ('Autobus');
```

6. Razvoj WPF aplikacije

6.1. Arhitektura aplikacije

6.1.1. Interfejs IData

Aplikacija je razvijana sa idejom da svaka klasa iz dijagrama klasa ima svoju klasu (model) u aplikaciji. Da bi se osiguralo postojanje neophodnih metoda (dodavanje, brisanje, izmena) za sve modele, kreiran je interfejs **IData** sa definisanim nazivima i tipovima metoda. **IData** se nalazi u folderu **Interfaces** u okviru projekta.

```
namespace AutoServis.Interfaces
{
    interface IData<T>
    {
        void Sacuvaj();
        void Obrisi();
        void Azuriraj(T obj);
        bool PostojiDuplikat();
    }
}
```

6.1.2. Klasa AutoServisData

Pošto je prilikom razvoja bilo koje aplikacije cilj smanjiti glomaznost programskog koda, poželjno je grupisati kod u metode koje će se koristiti u celoj aplikaciji. U ovoj aplikaciji, metode koje se koriste širom ostalih klasa, smeštene su u klasi **AutoServisData**. Ova klasa se nalazi u folderu **Data** u okviru projekta.

Statička metoda **UcitajPodatke(string sqlUpit)** iz klase **AutoServisData** vraća **DataTable** sa podacima koje povlači iz baze, a koji su zahtevani putem prosleđenog stringa tjs. upita. Ukoliko dođe do izuzetka vratiće **Null**.

```
public static DataTable UcitajPodatke(string sqlUpit)
    //pomocna metoda za učitavanje podataka koje cemo prikazivati u datagridu
    //ili comboboxu
{
    try
    {
        konekcija.Open();
        DataTable dt = new DataTable();
        SqlDataAdapter da = new SqlDataAdapter(sqlUpit, konekcija);
        da.Fill(dt);
        return dt;
    }
    catch (Exception e)
    {
        MessageBox.Show($"Došlo je do greške: { e.Message }. Podaci nisu učitani.",
            "Greška",
            MessageBoxButton.OK, MessageBoxImage.Error);
        return null;
    }
    finally
    {
    }
}
```

```

        if (konekcija != null)
        {
            konekcija.Close();
        }
    }
}

```

Statička metoda **Obrisi(string tabela, string idKolona, int idVrednost)** je metoda tipa **void** koja za prosleđeni naziv tabele, naziv kolone koja će biti specificirana kao uslov i za prosleđenu vrednost tj. konkretan uslov, izvršava brisanje podataka u bazi.

```

public static void Obrisi(string tabela, string idKolona, int idVrednost)
    //pomocna metoda za brisanje podataka iz tabela
{
    try
    {
        string sqlUpit = $"delete from { tabela } where { idKolona } = @idVrednost;";
        konekcija.Open();
        SqlCommand cmd = new SqlCommand(sqlUpit, konekcija);
        cmd.Parameters.AddWithValue("@idVrednost", idVrednost);
        cmd.ExecuteNonQuery();
    }
    catch (SqlException)
    {
        MessageBox.Show($"Postoje povezani podaci u drugim tabelama!", "Greska",
            MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
    catch (Exception e)
    {
        MessageBox.Show($"Došlo je do greške prilikom brisanja podataka:
            { e.Message }", "Greška",
            MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
    finally
    {
        if (konekcija != null)
        {
            konekcija.Close();
        }
    }
}

```

Statička metoda **BrojRedovaUBazi(string imeTabele, string polje, string uslov)** je metoda tipa **int** koja se koristi za dobijanje broja redova izvršenog upita sa, ili bez definisanog uslova. Kao što se može primetiti, nije najoptimalnije napisana, međutim, zadovoljava svoju namenu u ovoj aplikaciji. U zavisnosti od toga da li je neki od prosleđenih argumenata prazan string ili **Null** string, izvršiće se jedan od dva upita, a dobijeni rezultat biće konvertovan u **int** i vraćen kao rezultat metode.

```

public static int BrojRedovaUBazi(string imeTabele, string polje, string uslov)
    //prosledjujemo upit sa ili bez uslova, vraca nam broj redova u bazi
{
    try
    {
        string sqlUpit = $"select count(*) from { imeTabele }";
        if (String.IsNullOrEmpty(polje) || String.IsNullOrEmpty(uslov))
        {
            sqlUpit = $"select count(*) from { imeTabele }";
        }
    }
}

```



```

    } else
    {
        sqlUpit = $"select count(*) from { imeTabele } where { polje } = { uslov }";
    }
    konekcija.Open();
    SqlCommand cmd = new SqlCommand(sqlUpit, konekcija);
    return Convert.ToInt32(cmd.ExecuteScalar());
}
catch (Exception e)
{
    MessageBox.Show($"Došlo je do greške prilikom povlačenja podataka iz baze:
        { e.Message }", "Greška",
        MessageBoxButtons.OK, MessageBoxIcon.Error);
    return -1;
}
finally
{
    if (konekcija != null)
    {
        konekcija.Close();
    }
}
}

```

Statička metoda **PovuciIzTabele(string imeKolone, string imeTabele, string uslov)** vraća samo jedan string (rezultat upita), tjs. podatak iz tabele, koja je prosleđena zajedno sa nazivom kolone i željenim uslovom. Sam upit nije baš najoptimalnije napisan, ali je to dovoljno za jednu demonstrativnu aplikaciju.

```

public static string PovuciIzTabele(string imeKolone, string imeTabele, string uslov)
    //metoda ekvivalentna metodi DLookup() u MsAccess-u i VBA programskom jeziku
{
    try
    {
        string sqlUpit = $"select { imeKolone } from { imeTabele } where { uslov }";
        konekcija.Open();
        SqlCommand cmd = new SqlCommand(sqlUpit, konekcija);
        return cmd.ExecuteScalar().ToString();
    }
    catch (Exception e)
    {
        MessageBox.Show($"Došlo je do greške prilikom povlačenja podataka
            iz baze: { e.Message }", "Greška",
            MessageBoxButtons.OK, MessageBoxIcon.Error);
        return null;
    }
    finally
    {
        if (konekcija != null)
        {
            konekcija.Close();
        }
    }
}
}

```

Statička metoda **IspravnoKorisnickolme(string unetoKorisnickolme)** tipa **bool**, vraća **true** ili **false** u zavisnosti od toga da li prosleđeno korisničko ime zaista postoji u bazi. Kao što se primećuje na snippet-ima, za prosleđivanje konkretnih vrednosti kroz

upite, u najvećem broju slučajeva se koriste parametri. U konkretnom primeru vidi se parametar **@korisnickolme**, kome se prosleđuje vrednost iz unetog korisničkog imena. U upitu je specificirano da se traži broj slučajeva gde je korisničko ime iz baze zapravo jednako unetom korisničkom imenu. Dobijeni rezultat se najoptimalnije vraća korišćenjem metode **ExecuteScalar()**, jer ona vraća samo prvi red prve kolone izvršenog upita, dok se ostali redovi i ostale kolone zanemaruju. Dalje se konvertuje u **int** i u zavisnosti od vrednosti vraća **true** ili **false**.

```
public static bool IspravnoKorisnickoIme(string unetoKorisnickoIme)
//ova metoda proverava da li uneto korisnicko ime zaista postoji u bazi
{
    try
    {
        string sqlUpit = @"select count(*) from tblZaposleni where
                           KorisnickoIme=@korisnickoIme;";
        konekcija.Open();
        SqlCommand cmd = new SqlCommand(sqlUpit, konekcija);
        cmd.Parameters.AddWithValue("@korisnickoIme", unetoKorisnickoIme);
        int rez = Convert.ToInt32(cmd.ExecuteScalar());
        if (rez>0)
        {
            return true;
        } else
        {
            return false;
        }
    }
    catch (Exception e)
    {
        MessageBox.Show($"Došlo je do greške prilikom provere korisničkog
                           imena: { e.Message }.", "Greška",
        MessageBoxButtons.OK, MessageBoxIcon.Error);
        return false;
    }
    finally
    {
        if (konekcija != null)
        {
            konekcija.Close();
        }
    }
}
```

Metoda **IspravnaLozinka(string unetoKorisnickolme, string unetaLozinka)** je takođe tipa **bool**, i vraća **true** ili **false** u zavisnosti od rezultata izvršenog upita. Rezultat se dobija tako što se iz tabele **tblZaposleni** povlači lozinka za uneto korisničko ime, i zatim upoređuje sa unetom lozinkom. Ukoliko su iste, vraća se **true**, u suprotnom **false**, kao i ako dođe do nepredviđenog izuzetka.

```
public static bool IspravnaLozinka(string unetoKorisnickoIme, string unetaLozinka)
//ova metoda proverava da li je uneta ispravna lozinka za uneto korisnicko ime
{
    try
    {
        string sqlUpit = @"select Lozinka from tblZaposleni where
                           KorisnickoIme=@korisnickoIme;";
```

```

    konekcija.Open();
    SqlCommand cmd = new SqlCommand(sqlUpit, konekcija);
    cmd.Parameters.AddWithValue("@korisnickoIme", unetoKorisnickoIme);
    string dobijenaLozinka = cmd.ExecuteScalar().ToString();
    if (unetaLozinka.Equals(dobijenaLozinka))
    {
        return true;
    } else
    {
        return false;
    }
}
catch (Exception e)
{
    MessageBox.Show($"Došlo je do greške prilikom provere
                    lozinke: { e.Message }.", "Greška",
    MessageBoxButtons.OK, MessageBoxIcon.Error);
    return false;
}
finally
{
    if (konekcija != null)
    {
        konekcija.Close();
    }
}
}

```

6.1.3. Klasa Konekcija

Veza sa bazom ostvaruje se uz pomoć klase Konekcija u kojoj je deklarirana statička metoda **KreirajKonekciju()** tipa **SqlConnection**. U njoj su definisani neophodni parametri da bi se ostvarila veza sa bazom i mogućnost da se konekcija otvori. Konekcija se otvara svakog puta kada želimo da izvršimo upit, a zatvara se čim je upit izvršen, uspešno ili neuspešno. Ova klasa se nalazi odvojeno u samom projektu.

```

class Konekcija
{
    public static SqlConnection KreirajKonekciju()
    {
        SqlConnectionStringBuilder ccnsb = new SqlConnectionStringBuilder();
        ccnsb.DataSource = @"JOVAN-PC";
        ccnsb.InitialCatalog = "AutoServis";
        ccnsb.IntegratedSecurity = true;

        string con = ccnsb.ToString();
        SqlConnection konekcija = new SqlConnection(con);
        return konekcija;
    }
}

```

6.1.4. Modeli

Modeli predstavljaju klase u aplikaciji koje reprezentuju klase iz dijagrama klasa. Svi modeli nasleđuju interfejs **IData** zbog osiguravanja postojanja metoda neophodnih za realizaciju **CRUD** (Create, Read, Update, Delete) operacija. Pored toga, neke klase sadrže i dodatne metode različitih tipova, da bi se postiglo optimalnije rešavanje nekih problema koji će biti izneti u nastavku (kao što je npr. problem pretrage podataka). Ovakva realizacija aplikacije proizašla je iz želje da korisnički interfejs bude maksimalno odvojen od pozadinskog koda i definisanja njegovog ponašanja. Imajući u vidu da je ovo demonstrativna aplikacija, i da za ograničeno vreme nije bilo moguće realizovati je kroz **MVVM** (Model-View-ViewModel) dizajnerski obrazac, izabran je ovaj samovoljni prilaz koji je ipak na kraju zadovoljio potrebe i očekivanja autora.

Sve klase koje reprezentuju modele se nalaze u folderu **Models** u okviru projekta.

U okviru prvog code snippet-a će biti objašnjeno kako se koja operacija izvršava, a uz sledeće će biti pojašnjeno samo ono što je izmenjeno u odnosu na prvi snippet, ili ukoliko je implementirana neka nova metoda koje nema u ostalim klasama.

6.1.4.1. Klasa Vozilo

```
class Vozilo : IData<Vozilo>
{
    private static SqlConnection konekcija = Konekcija.KreirajKonekciju();

    #region privatne promenljive
    private int _id;
    private int _snagaMotora;
    private string _brojMotora;
    private int _godinaProizvodnje;
    private int _zapreminaMotora;
    private string _registarskaOznaka;
    private string _brojSasiije;

    private Gorivo _vrstaGoriva;
    private Vlasnik _vlasnik;
    private TipVozila _tipVozila;
    private Model _model;
    #endregion

    #region getteri i setteri
    public int Id
    {
        get { return _id; }
        set { _id = value; }
    }
    public int SnagaMotora
    {
        get { return _snagaMotora; }
        set { _snagaMotora = value; }
    }
}
```

```

public string BrojMotora
{
    get { return _brojMotora; }
    set { _brojMotora = value; }
}
public int GodinaProizvodnje
{
    get { return _godinaProizvodnje; }
    set { _godinaProizvodnje = value; }
}
public int ZapreminaMotora
{
    get { return _zapreminaMotora; }
    set { _zapreminaMotora = value; }
}
public string RegistarskaOznaka
{
    get { return _registarskaOznaka; }
    set { _registarskaOznaka = value; }
}
public string BrojSasije
{
    get { return _brojSasije; }
    set { _brojSasije = value; }
}
public Gorivo VrstaGoriva
{
    get { return _vrstaGoriva; }
    set { _vrstaGoriva = value; }
}
public Vlasnik Vlasnik
{
    get { return _vlasnik; }
    set { _vlasnik = value; }
}
public TipVozila TipVozila
{
    get { return _tipVozila; }
    set { _tipVozila = value; }
}
public Model Model
{
    get { return _model; }
    set { _model = value; }
}
#endregion

#region konstruktor
public Vozilo()
{
}
#endregion

#region metode
public void Sacuvaj()
{
    string sqlUpit = @"insert into tblVozilo(VrstaGorivaID, SnagaMotora, BrojMotora,
        GodinaProizvodnje, ZapreminaMotora,
        RegOznaka, BrojSasije, VlasnikID, TipVozilaID, ModelID) values
        (@vrstaGorivaId, @snagaMotora, @brojMotora,
        @godinaProizvodnje, @zapreminaMotora, @regOznaka, @brojSasije,
        @vlasnikId, @tipVozilaId, @modelId)";
}

```

```

try
{
    konekcija.Open();
    SqlCommand cmd = new SqlCommand(sqlUpit, konekcija);
    cmd.Parameters.AddWithValue("@vrstaGorivaId", VrstaGoriva.Id);
    cmd.Parameters.AddWithValue("@snagaMotora", SnagaMotora);
    cmd.Parameters.AddWithValue("@brojMotora", BrojMotora);
    cmd.Parameters.AddWithValue("@godinaProizvodnje", GodinaProizvodnje);
    cmd.Parameters.AddWithValue("@zapreminaMotora", ZapreminaMotora);
    cmd.Parameters.AddWithValue("@regOznaka", RegistarskaOznaka);
    cmd.Parameters.AddWithValue("@brojSasiije", BrojSasiije);
    cmd.Parameters.AddWithValue("@vlasnikId", Vlasnik.Id);
    cmd.Parameters.AddWithValue("@tipVozilaId", TipVozila.Id);
    cmd.Parameters.AddWithValue("@modelId", Model.Id);
    cmd.ExecuteNonQuery();
}
catch (Exception e)
{
    MessageBox.Show($"Došlo je do greške: { e.Message }. Podaci nisu sačuvani.",
        "Greška",
        MessageBoxButtons.OK, MessageBoxIcon.Error);
}
finally
{
    if (konekcija != null)
    {
        konekcija.Close();
    }
}
}

public void Obrisi()
{
    AutoServisData.Obrisi("tblVozilo", "VoziloID", Id);
}

public void Azuriraj(Vozilo novoVozilo)
{
    try
    {
        string sqlUpit = @"update tblVozilo set VrstaGorivaID=@vrstaGorivaId,
            SnagaMotora=@snagaMotora,
            BrojMotora=@brojMotora,
            GodinaProizvodnje=@godinaProizvodnje,
            RegOznaka=@regOznaka, BrojSasiije=@brojSasiije,
            VlasnikID=@vlasnikId, TipVozilaID=@tipVozilaId,
            ModelID=@modelId where VoziloID=@id";

        konekcija.Open();
        SqlCommand cmd = new SqlCommand(sqlUpit, konekcija);
        cmd.Parameters.AddWithValue("@id", Id);
        cmd.Parameters.AddWithValue("@vrstaGorivaId", novoVozilo.VrstaGoriva.Id);
        cmd.Parameters.AddWithValue("@snagaMotora", novoVozilo.SnagaMotora);
        cmd.Parameters.AddWithValue("@brojMotora", novoVozilo.BrojMotora);
        cmd.Parameters.AddWithValue("@godinaProizvodnje", novoVozilo.GodinaProizvodnje);

;

        cmd.Parameters.AddWithValue("@zapreminaMotora", novoVozilo.ZapreminaMotora);
        cmd.Parameters.AddWithValue("@regOznaka", novoVozilo.RegistarskaOznaka);
        cmd.Parameters.AddWithValue("@brojSasiije", novoVozilo.BrojSasiije);
        cmd.Parameters.AddWithValue("@vlasnikId", novoVozilo.Vlasnik.Id);
        cmd.Parameters.AddWithValue("@tipVozilaId", novoVozilo.TipVozila.Id);
        cmd.Parameters.AddWithValue("@modelId", novoVozilo.Model.Id);
        cmd.ExecuteNonQuery();
    }
}

```

```

        catch (Exception e)
        {
            MessageBox.Show($"Došlo je do greške: { e.Message }.
            Podaci nisu ažurirani.", "Greška",
            MessageBoxButtons.OK, MessageBoxImage.Error);
        }
    finally
    {
        if (konekcija != null)
        {
            konekcija.Close();
        }
    }
}

public bool PostojiDuplikat()
{
    try
    {
        string sqlUpit = @"select count(*) from tblVozilo where ModelID=@modelId
        AND BrojSasije=@brojSasije AND VrstaGorivaID=@vrstaGorivaId
        AND SnagaMotora=@snagaMotora
        AND BrojMotora=@brojMotora AND
        GodinaProizvodnje=@godinaProizvodnje
        AND ZapreminaMotora=@zapreminaMotora;";

        konekcija.Open();
        SqlCommand cmd = new SqlCommand(sqlUpit, konekcija);
        cmd.Parameters.AddWithValue("@vrstaGorivaId", VrstaGoriva.Id);
        cmd.Parameters.AddWithValue("@snagaMotora", SnagaMotora);
        cmd.Parameters.AddWithValue("@brojMotora", BrojMotora);
        cmd.Parameters.AddWithValue("@godinaProizvodnje", GodinaProizvodnje);
        cmd.Parameters.AddWithValue("@zapreminaMotora", ZapreminaMotora);
        cmd.Parameters.AddWithValue("@brojSasije", BrojSasije);
        cmd.Parameters.AddWithValue("@tipVozilaId", TipVozila.Id);
        cmd.Parameters.AddWithValue("@modelId", Model.Id);
        int rez = Convert.ToInt32(cmd.ExecuteScalar());
        if (rez > 0)
        {
            return true;
        }
        else
        {
            return false;
        }
    }
    catch (Exception e)
    {
        MessageBox.Show($"Došlo je do greške prilikom provere duplikata u bazi:
        { e.Message }", "Greška",
        MessageBoxButtons.OK, MessageBoxImage.Error);
        return false;
    }
    finally
    {
        if (konekcija != null)
        {
            konekcija.Close();
        }
    }
}

public static DataTable ListaVozila(string filter)
{
    string sqlUpit = @"SELECT tblVozilo.voziloId as 'ID',

```

```

        tblMarka.NazivMarke + ' ' + tblModel.NazivModela as 'Vozilo',
        tblVlasnik.ImeVlasnika + ' ' + tblvlasnik.PrezimeVlasnika
        as 'Vlasnik',
        tblGorivo.VrstaGoriva as 'Pogonsko gorivo',
        tblTipVozila.TipVozila as 'Tip vozila',
        tblVozilo.RegOznaka as 'Registarska oznaka',
        tblVozilo.SnagaMotora as 'Snaga motora',
        tblVozilo.BrojMotora as 'Broj motora',
        tblVozilo.GodinaProizvodnje as 'Godina proizvodnje',
        tblVozilo.ZapreminaMotora as 'Zapremina motora',
        tblVozilo.BrojSasije as 'Broj šasije'
    FROM tblVozilo join tblVlasnik on
        tblVozilo.VlasnikID=tblVlasnik.VlasnikID
        join tblModel on tblVozilo.ModelID = tblModel.ModelID
        join tblMarka on tblModel.MarkaID = tblmarka.MarkaID
    join tblTipVozila on
        tblVozilo.TipVozilaID=tblTipVozila.TipVozilaID
    join tblGorivo on
        tblVozilo.VrstaGorivaID=tblGorivo.VrstaGorivaID
    where tblVozilo.VoziloID like '%' + filter + @%"' or
    (tblMarka.NazivMarke + ' ' + tblModel.NazivModela) like '%'
    + filter + @%"'
    or (tblVlasnik.ImeVlasnika + ' ' +
        tblVlasnik.PrezimeVlasnika) like '%' + filter + @%"'
    or tblGorivo.VrstaGoriva like '%' + filter + @%"'
    or tblTipVozila.TipVozila like '%' + filter + @%"'
    or tblVozilo.RegOznaka like '%' + filter + @%"'
    or tblVozilo.SnagaMotora like '%' + filter + @%"'
    or tblVozilo.BrojMotora like '%' + filter + @%"'
    or tblVozilo.GodinaProizvodnje like '%' + filter + @%"'
    or tblVozilo.ZapreminaMotora like '%' + filter + @%"'
    or tblVozilo.BrojSasije like '%' + filter + @%"';";

    return AutoServisData.UcitajPodatke(sqlUpit);
}

public static Vozilo UcitajVozilo(int id)
{
    try
    {
        string sqlUpit = @"select * from tblVozilo where VoziloID=@id";
        Vozilo vozilo = new Vozilo();
        konekcija.Open();
        SqlCommand cmd = new SqlCommand(sqlUpit, konekcija);
        cmd.Parameters.AddWithValue("@id", id);
        SqlDataReader citac = cmd.ExecuteReader();
        while (citac.Read())
        {
            vozilo.Id = Convert.ToInt32(citac["VoziloID"]);
            vozilo.SnagaMotora = Convert.ToInt32(citac["SnagaMotora"]);
            vozilo.BrojMotora = citac["BrojMotora"].ToString();
            vozilo.GodinaProizvodnje = Convert.ToInt32(citac["GodinaProizvodnje"]);
            vozilo.ZapreminaMotora = Convert.ToInt32(citac["ZapreminaMotora"]);
            vozilo.RegistarskaOznaka = citac["RegOznaka"].ToString();
            vozilo.BrojSasije = citac["BrojSasije"].ToString();

            vozilo.VrstaGoriva = Gorivo.UcitajGorivo(
                Convert.ToInt32(citac["VrstaGorivaID"]));
            vozilo.Vlasnik = Vlasnik.UcitajVlasnika(
                Convert.ToInt32(citac["VlasnikID"]));
            vozilo.TipVozila = TipVozila.UcitajTipVozila(
                Convert.ToInt32(citac["TipVozilaID"]));
            vozilo.Model = Model.UcitajModel(Convert.ToInt32(citac["ModelID"]));
        }
    }
}

```



```

        return vozilo;
    }
    catch (Exception e)
    {
        MessageBox.Show($"Došlo je do greške: { e.Message }.
                          Podaci nisu učitani.", "Greška",
        MessageBoxButtons.OK, MessageBoxIcon.Error);
        return null;
    }
    finally
    {
        if (konekcija != null)
        {
            konekcija.Close();
        }
    }
}

public static List<int> ListaNaloga(int id)
    //vraća listu brojeva radnih naloga za određeno vozilo,
    //da bismo mogli da prođemo kroz listu i obrišemo sve podatke za njih
{
    try
    {
        string sqlUpit = @"select RadniNalogID from tblRadniNalog where VoziloID=@id";
        List<int> list = new List<int>();
        konekcija.Open();
        SqlCommand cmd = new SqlCommand(sqlUpit, konekcija);
        cmd.Parameters.AddWithValue("@id", id);
        SqlDataReader citac = cmd.ExecuteReader();
        while (citac.Read())
        {
            list.Add(Convert.ToInt32(citac["RadniNalogID"]));
        }
        return list;
    }
    catch (Exception e)
    {
        MessageBox.Show($"Došlo je do greške prilikom povlačenja
                          naloga iz baze: { e.Message }", "Greška",
        MessageBoxButtons.OK, MessageBoxIcon.Error);
        return null;
    }
    finally
    {
        if (konekcija != null)
        {
            konekcija.Close();
        }
    }
}
}
#endregion
}

```

Metoda **Sacuvaj()** za atribute deklarisanе u okviru klase izvršava upit i koristi atribute kao parametre upita ukoliko su oni primitivnog tipa, a ukoliko su oni instance tjs. objekti nekih drugih klasa (kao što su u ovom slučaju **_tipVozila**, **_vlasnik** itd.) onda se kao parametri u upitu koriste atributi tih objekata (pa tako za privatni objekat **_tipVozila** postoji javni objekat **TipVozila** koji svoj atribut **Id** upisuje u tabelu **tblVozilo** koji je zapravo u toj tabeli strani ključ. Ova metoda je manje-više ista u svim

modelima, pa će ovde biti kraj sa njenim objašnjavanjem, sve do objašnjenja kako se ovi modeli i njihove metode koriste u korisničkom interfejsu. Koristi se za insert objekta, tjs. njegovih osobina u bazu.

Metoda **Obrisi()** tipa **void** u ovoj klasi samo poziva ranije objašnjenu metodu **Obrisi()** iz klase **AutoServisData**. Služi za brisanje objekta, tjs. njegovih osobina u bazi.

Metoda **Azuriraj(Vozilo novoVozilo)** tipa **void** se koristi za izmenu tjs. ažuriranje osobina instance klase **Vozilo**. Metodi se prosleđuje argument **novoVozilo** istog tipa, a zatim se objektu koji poziva metodu postavljaju vrednosti argumenta **novoVozilo**. To se dešava kroz izvršavanje upita u kome parametri dobijaju vrednosti od prosleđenog argumenta. I ova metoda je ista u svim modelima.

Metoda **PostojiDuplikat()** tipa **bool** vraća **true** ako se ustanovi da postoji duplikat objekta u bazi, ili **false** ukoliko ne postoji. Zapravo se proverava da li osobine objekta postoje u bazi. Provera se vrši parametrizovanim upitom, slično ranije opisanim metodama. Za ispitivanje rezultata se koristi konvertovanje **ExecuteScalar()** rezultata u tip **int**.

To su metode koje se u okviru ove klase pozivaju od strane instance te klase. Slede statičke metode.

Statička metoda tipa **void ListaVozila(string filter)** jednostavno vraća **DataTable** pozivajući statičku metodu **UcitajPodatke(string sqlUpit)** iz ranije pominjane klase **AutoServisData**. Izuzetak u odnosu na neke ostale modele je što se u ovoj klasi za listu vozila koristi i **filter** koji je definisan kao parametar tipa **string**, dok se u pozivu metode prosleđuje argument koji je u suštini **string** koji sadrži pojam za pretragu (prostije rečeno **TextBox.Text**). Iako nije najsrećnije implementirano rešenje, za ovakvu aplikaciju je više nego zadovoljavajuće. Mnogo je kvalitetnija varijanta da se filter prosleđivao kao parametar upitu, ali za implementaciju takve metode je potrebno dosta više vremena. Koristi se za prikaz liste vozila u **DataGrid** kontroli.

Statička metoda **UcitajVozilo(int id)** tipa **Vozilo** služi za učitavanje osobina klase **Vozilo** kreiranom objektu iste klase. Metoda se koristi u skoro svim ostalim modelima, veoma je korisna i zahvalna. Primer korišćenja:

```
Vozilo vozilo = Vozilo.UcitajVozilo(1);
```

Objektu vozilo će biti učitane sve osobine vozila iz baze sa **Id** brojem 1. Učitavanje se vrši izvršavanjem upita u bazi i povlačenjem podataka gde je parametar **@id** jednak koloni **VoziloID**. Nije potrebno navoditi koliko je ovo korisno, a najviše je u upotrebi za kreiranje objekata u okviru drugih modela, kao što je slučaj sa modelima **TipVozila** itd. u okviru ove klase. Preko ovog objekta vozilo možemo pristupiti i osobinama klase **Vlasnik**, **Marka**, **Model** itd.

Primer:

```
string imeVlasnika = vozilo.Vlasnik.ImeVlasnika;
```

Statička metoda **ListaNaloga(int id)** tipa **List<int>** služi da vrati listu sa brojevima svih radnih naloga koji su kreirani za ovo vozilo. Ta lista kasnije služi da bi se prolaskom kroz nju brisali radni nalozi jedan po jedan. Uistinu korisna metoda, a naizgled vrlo glupa.

6.4.1.2. Klasa Vlasnik

```
class Vlasnik : IData<Vlasnik>
{
    private static SqlConnection konekcija = Konekcija.KreirajKonekciju();

    #region privatne promenljive
    private int _id;
    private string _imeVlasnika;
    private string _prezimeVlasnika;
    private string _kontaktVlasnika;
    private string _jmbgVlasnika;
    private string _gradVlasnika;
    private string _brojLKVlasnika;
    private string _adresaVlasnika;
    #endregion

    #region getteri i setteri
    public int Id
    {
        get { return _id; }
        set { _id = value; }
    }
    public string ImeVlasnika
    {
        get { return _imeVlasnika; }
        set { _imeVlasnika = value; }
    }
    public string PrezimeVlasnika
    {
        get { return _prezimeVlasnika; }
        set { _prezimeVlasnika = value; }
    }
    public string KontaktVlasnika
    {
        get { return _kontaktVlasnika; }
        set { _kontaktVlasnika = value; }
    }
    public string JMBGVlasnika
    {
        get { return _jmbgVlasnika; }
        set { _jmbgVlasnika = value; }
    }
    public string GradVlasnika
    {
        get { return _gradVlasnika; }
        set { _gradVlasnika = value; }
    }
    public string BrojLKVlasnika
```

```

{
    get { return _brojLKVlasnika; }
    set { _brojLKVlasnika = value; }
}
public string AdresaVlasnika
{
    get { return _adresaVlasnika; }
    set { _adresaVlasnika = value; }
}
#endregion

#region konstruktor
public Vlasnik()
{

}
#endregion

#region metode
public void Sacuvaj()
{
    string sqlUpit = @"insert into tblVlasnik(ImeVlasnika, PrezimeVlasnika,
        KontaktVlasnika, JMBGVlasnika, AdresaVlasnika,
        GradVlasnika, BrojLKVlasnika)
        values (@ime, @prezime, @kontakt, @jmbg,
            @adresa, @grad, @brojLK)";

    try
    {
        konekcija.Open();
        SqlCommand cmd = new SqlCommand(sqlUpit, konekcija);
        cmd.Parameters.AddWithValue("@ime", ImeVlasnika);
        cmd.Parameters.AddWithValue("@prezime", PrezimeVlasnika);
        cmd.Parameters.AddWithValue("@kontakt", KontaktVlasnika);
        cmd.Parameters.AddWithValue("@jmbg", JMBGVlasnika);
        cmd.Parameters.AddWithValue("@adresa", AdresaVlasnika);
        cmd.Parameters.AddWithValue("@grad", GradVlasnika);
        cmd.Parameters.AddWithValue("@brojLK", BrojLKVlasnika);
        cmd.ExecuteNonQuery();
    }
    catch (Exception e)
    {
        MessageBox.Show($"Došlo je do greške: { e.Message }.
            Podaci nisu sačuvani.", "Greška",
            MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
    finally
    {
        if (konekcija != null)
        {
            konekcija.Close();
        }
    }
}

public void Obrisi()
{
    AutoServisData.Obrisi("tblVlasnik", "VlasnikID", Id);
}

public void Azuriraj(Vlasnik noviVlasnik)
{
    try
    {
        string sqlUpit = @"update tblVlasnik set ImeVlasnika=@ime,
            PrezimeVlasnika=@prezime, KontaktVlasnika=@kontakt,

```

```

        JMBGVlasnika=@jmbg,
        AdresaVlasnika=@adresa, GradVlasnika=@grad,
        BrojLKVlasnika=@brojLK where VlasnikID=@id;";

    konekcija.Open();
    SqlCommand cmd = new SqlCommand(sqlUpit, konekcija);
    cmd.Parameters.AddWithValue("@id", Id);
    cmd.Parameters.AddWithValue("@ime", noviVlasnik.ImeVlasnika);
    cmd.Parameters.AddWithValue("@prezime", noviVlasnik.PrezimeVlasnika);
    cmd.Parameters.AddWithValue("@kontakt", KontaktVlasnika);
    cmd.Parameters.AddWithValue("@jmbg", noviVlasnik.JMBGVlasnika);
    cmd.Parameters.AddWithValue("@adresa", noviVlasnik.AdresaVlasnika);
    cmd.Parameters.AddWithValue("@grad", noviVlasnik.GradVlasnika);
    cmd.Parameters.AddWithValue("@brojLK", noviVlasnik.BrojLKVlasnika);
    cmd.ExecuteNonQuery();
}
catch (Exception e)
{
    MessageBox.Show($"Došlo je do greške: { e.Message }.
        Podaci nisu ažurirani.", "Greška",
        MessageBoxButtons.OK, MessageBoxIcon.Error);
}
finally
{
    if (konekcija != null)
    {
        konekcija.Close();
    }
}
}
public bool PostojiDuplikat()
{
    try
    {
        string sqlUpit = @"select count(*) from tblVlasnik where
            ImeVlasnika=@ime AND JMBGVlasnika=@jmbg;";
        konekcija.Open();
        SqlCommand cmd = new SqlCommand(sqlUpit, konekcija);
        cmd.Parameters.AddWithValue("@ime", ImeVlasnika);
        cmd.Parameters.AddWithValue("@jmbg", JMBGVlasnika);
        int rez = Convert.ToInt32(cmd.ExecuteScalar());
        if (rez > 0)
        {
            return true;
        }
        else
        {
            return false;
        }
    }
    catch (Exception e)
    {
        MessageBox.Show($"Došlo je do greške prilikom provere
            duplikata u bazi: { e.Message }", "Greška",
            MessageBoxButtons.OK, MessageBoxIcon.Error);
        return false;
    }
    finally
    {
        if(konekcija != null)
        {
            konekcija.Close();
        }
    }
}

```

```

}
public static DataTable ListaVlasnika(string filter)
{
    string sqlUpit = @"select VlasnikID as 'ID', ImeVlasnika as 'Ime',
        PrezimeVlasnika as 'Prezime', KontaktVlasnika as 'Kontakt',
        JMBGVlasnika as 'JMBG', AdresaVlasnika as 'Adresa',
        GradVlasnika as 'Grad',
        BrojLKVlasnika as 'Broj LK'
    FROM tblVlasnik
    where ImeVlasnika like '%" + filter + @%"' or
        PrezimeVlasnika like '%" + filter + @%"' or
        KontaktVlasnika like '%" + filter + @%"' or
        JMBGVlasnika like '%" + filter + @%"' or
        AdresaVlasnika like '%" + filter + @%"' or
        GradVlasnika like '%" + filter + @%"' or
        BrojLKVlasnika like '%" + filter + @%"';";

    return AutoServisData.UcitajPodatke(sqlUpit);
}
public static Vlasnik UcitajVlasnika(int id)
{
    try
    {
        string sqlUpit = @"select * from tblVlasnik where VlasnikID=@id";
        Vlasnik vlasnik = new Vlasnik();
        konekcija.Open();
        SqlCommand cmd = new SqlCommand(sqlUpit, konekcija);
        cmd.Parameters.AddWithValue("@id", id);
        SqlDataReader citac = cmd.ExecuteReader();
        while (citac.Read())
        {
            vlasnik.Id = Convert.ToInt32(citac["VlasnikID"]);
            vlasnik.ImeVlasnika = citac["ImeVlasnika"].ToString();
            vlasnik.PrezimeVlasnika = citac["PrezimeVlasnika"].ToString();
            vlasnik.KontaktVlasnika = citac["KontaktVlasnika"].ToString();
            vlasnik.JMBGVlasnika = citac["JMBGVlasnika"].ToString();
            vlasnik.AdresaVlasnika = citac["AdresaVlasnika"].ToString();
            vlasnik.GradVlasnika = citac["GradVlasnika"].ToString();
            vlasnik.BrojLKVlasnika = citac["BrojLKVlasnika"].ToString();
        }
        return vlasnik;
    }
    catch (Exception e)
    {
        MessageBox.Show($"Došlo je do greške: { e.Message }.
            Podaci nisu učitani.", "Greška",
            MessageBoxButtons.OK, MessageBoxIcon.Error);
        return null;
    }
    finally
    {
        if (konekcija != null)
        {
            konekcija.Close();
        }
    }
}
public static List<int> ListaVozila(int id)
//vraća listu brojeva vozila za vlasnika sa prosledjenim id brojem
{
    try
    {
        string sqlUpit = @"select VoziloID from tblVozilo where VlasnikID=@id";
        List<int> list = new List<int>();
    }
}

```

```

        konekcija.Open();
        SqlCommand cmd = new SqlCommand(sqlUpit, konekcija);
        cmd.Parameters.AddWithValue("@id", id);
        SqlDataReader citac = cmd.ExecuteReader();
        while (citac.Read())
        {
            list.Add(Convert.ToInt32(citac["VoziloID"]));
        }
        return list;
    }
    catch (Exception e)
    {
        MessageBox.Show($"Došlo je do greške prilikom povlačenja
            vozila iz baze: { e.Message }", "Greška",
            MessageBoxButton.OK, MessageBoxImage.Error);
        return null;
    }
    finally
    {
        if (konekcija != null)
        {
            konekcija.Close();
        }
    }
}
#endregion
}

```

U modelu **Vlasnik** se nalaze sve metode analogne metodama u modelu **Vozilo**. Nisu potrebna dodatna objašnjenja.

6.4.1.3 Klasa Zaposleni

```

class Zaposleni : Interfaces.IData<Zaposleni>
{
    private static SqlConnection konekcija = Konekcija.KreirajKonekciju();

    #region privatne promenljive
    private int _id;
    private string _imeZaposlenog;
    private string _prezimeZaposlenog;
    private string _jmbgZaposlenog;
    private string _gradZaposlenog;
    private string _korisnickoIme;
    private string _lozinka;
    private string _brojLKZaposlenog;
    private string _adresaZaposlenog;
    #endregion

    #region getteri i setteri
    public int Id
    {
        get { return _id; }
        set { _id = value; }
    }
    public string ImeZaposlenog
    {
        get { return _imeZaposlenog; }
        set { _imeZaposlenog = value; }
    }
    public string PrezimeZaposlenog

```

```

{
    get { return _prezimeZaposlenog; }
    set { _prezimeZaposlenog = value; }
}
public string JMBGZaposlenog
{
    get { return _jmbgZaposlenog; }
    set { _jmbgZaposlenog = value; }
}
public string GradZaposlenog
{
    get { return _gradZaposlenog; }
    set { _gradZaposlenog = value; }
}
public string KorisnickoIme
{
    get { return _korisnickoIme; }
    set { _korisnickoIme = value; }
}
public string Lozinka
{
    get { return _lozinka; }
    set { _lozinka= value; }
}
public string BrojLKZaposlenog
{
    get { return _brojLKZaposlenog; }
    set { _brojLKZaposlenog = value; }
}
public string AdresaZaposlenog
{
    get { return _adresaZaposlenog; }
    set { _adresaZaposlenog = value; }
}
#endregion

#region konstruktor
public Zaposleni()
{
}
#endregion

#region metode
public void Obrisi()
{
    AutoServisData.Obrisi("tblZaposleni", "ZaposleniID", Id);
}
public void Sacuvaj()
{
    string sqlUpit = @"insert into tblZaposleni(ImeZaposlenog, PrezimeZaposlenog,
        JMBGZaposlenog, AdresaZaposlenog,
        GradZaposlenog, KorisnickoIme, Lozinka, BrojLKZaposlenog)
        values (@ime, @prezime, @jmbg, @adresa, @grad, @korisnickoIme,
        @lozinka, @brojLK)";

    try
    {
        konekcija.Open();
        SqlCommand cmd = new SqlCommand(sqlUpit, konekcija);
        cmd.Parameters.AddWithValue("@ime", ImeZaposlenog);
        cmd.Parameters.AddWithValue("@prezime", PrezimeZaposlenog);
        cmd.Parameters.AddWithValue("@jmbg", JMBGZaposlenog);
        cmd.Parameters.AddWithValue("@adresa", AdresaZaposlenog);
    }
}

```



```

        cmd.Parameters.AddWithValue("@grad", GradZaposlenog);
        cmd.Parameters.AddWithValue("@korisnickoIme", KorisnickoIme);
        cmd.Parameters.AddWithValue("@lozinka", Lozinka);
        cmd.Parameters.AddWithValue("@brojLK", BrojLKZaposlenog);
        cmd.ExecuteNonQuery();
    }
    catch (Exception e)
    {
        MessageBox.Show($"Došlo je do greške: { e.Message }.
                        Podaci nisu sačuvani.", "Greška",
                        MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
    finally
    {
        if (konekcija != null)
        {
            konekcija.Close();
        }
    }
}

public void Azuriraj(Zaposleni noviZaposleni)
{
    try
    {
        string sqlUpit = @"update tblZaposleni set ImeZaposlenog=@ime,
                        PrezimeZaposlenog=@prezime, JMBGZaposlenog=@jmbg,
                        AdresaZaposlenog=@adresa, GradZaposlenog=@grad,
                        KorisnickoIme=@korisnickoIme, Lozinka=@lozinka,
                        BrojLKZaposlenog=@brojLK where ZaposleniID=@id;";
        konekcija.Open();
        SqlCommand cmd = new SqlCommand(sqlUpit, konekcija);
        cmd.Parameters.AddWithValue("@id", Id);
        cmd.Parameters.AddWithValue("@ime", noviZaposleni.ImeZaposlenog);
        cmd.Parameters.AddWithValue("@prezime", noviZaposleni.PrezimeZaposlenog);
        cmd.Parameters.AddWithValue("@jmbg", noviZaposleni.JMBGZaposlenog);
        cmd.Parameters.AddWithValue("@adresa", noviZaposleni.AdresaZaposlenog);
        cmd.Parameters.AddWithValue("@grad", noviZaposleni.GradZaposlenog);
        cmd.Parameters.AddWithValue("@korisnickoIme", noviZaposleni.KorisnickoIme);
        cmd.Parameters.AddWithValue("@lozinka", noviZaposleni.Lozinka);
        cmd.Parameters.AddWithValue("@brojLK", noviZaposleni.BrojLKZaposlenog);
        cmd.ExecuteNonQuery();
    }
    catch (Exception e)
    {
        MessageBox.Show($"Došlo je do greške: { e.Message }.
                        Podaci nisu ažurirani.", "Greška",
                        MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
    finally
    {
        if (konekcija != null)
        {
            konekcija.Close();
        }
    }
}

public bool PostojiDuplikat()
{
    try
    {
        string sqlUpit = @"select count(*) from tblZaposleni where
                        ImeZaposlenog=@ime AND JMBGZaposlenog=@jmbg;";
        konekcija.Open();
        SqlCommand cmd = new SqlCommand(sqlUpit, konekcija);
    }

```

```

        cmd.Parameters.AddWithValue("@ime", ImeZaposlenog);
        cmd.Parameters.AddWithValue("@jmbg", JMBGZaposlenog);
        int rez = Convert.ToInt32(cmd.ExecuteScalar());
        if (rez > 0)
        {
            return true;
        }
        else
        {
            return false;
        }
    }
    catch (Exception e)
    {
        MessageBox.Show($"Došlo je do greške prilikom provere
                        duplikata u bazi: { e.Message }", "Greška",
        MessageBoxButtons.OK, MessageBoxIcon.Error);
        return false;
    }
    finally
    {
        if (konekcija != null)
        {
            konekcija.Close();
        }
    }
}

public static DataTable ListaZaposlenih(string filter)
{
    string sqlUpit = @"select ZaposleniID as 'ID', ImeZaposlenog as 'Ime',
                        PrezimeZaposlenog as 'Prezime',
                        JMBGZaposlenog as 'JMBG', AdresaZaposlenog as 'Adresa',
                        GradZaposlenog as 'Grad',
                        KorisnickoIme as 'Korisničko ime',
                        BrojLKZaposlenog as 'Broj LK'
                        FROM tblZaposleni
                        where ImeZaposlenog like '%" + filter + @"%' or
                        PrezimeZaposlenog like '%" + filter + @"%' or
                        JMBGZaposlenog like '%" + filter + @"%' or
                        AdresaZaposlenog like '%" + filter + @"%' or
                        GradZaposlenog like '%" + filter + @"%' or
                        KorisnickoIme like '%" + filter + @"%' or
                        BrojLKZaposlenog like '%" + filter + @"%'";

    return AutoServisData.UcitajPodatke(sqlUpit);
}

public static Zaposleni UcitajZaposlenog(int id)
{
    try
    {
        string sqlUpit = @"select * from tblZaposleni where ZaposleniID=@id";
        Zaposleni zaposleni = new Models.Zaposleni();
        konekcija.Open();
        SqlCommand cmd = new SqlCommand(sqlUpit, konekcija);
        cmd.Parameters.AddWithValue("@id", id);
        SqlDataReader citac = cmd.ExecuteReader();
        while (citac.Read())
        {
            zaposleni.Id = Convert.ToInt32(citac["ZaposleniID"]);
            zaposleni.ImeZaposlenog = citac["ImeZaposlenog"].ToString();
            zaposleni.PrezimeZaposlenog = citac["PrezimeZaposlenog"].ToString();
            zaposleni.JMBGZaposlenog = citac["JMBGZaposlenog"].ToString();
            zaposleni.AdresaZaposlenog = citac["AdresaZaposlenog"].ToString();
            zaposleni.GradZaposlenog = citac["GradZaposlenog"].ToString();
            zaposleni.KorisnickoIme = citac["KorisnickoIme"].ToString();
        }
    }
}

```

```

        zaposleni.Lozinka = citac["Lozinka"].ToString();
        zaposleni.BrojLKZaposlenog = citac["BrojLKZaposlenog"].ToString();

    }
    return zaposleni;
}
catch (Exception e)
{
    MessageBox.Show($"Došlo je do greške: { e.Message }. Podaci nisu učitani.",
        "Greška",
        MessageBoxButton.OK, MessageBoxImage.Error);
    return null;
}
finally
{
    if (konekcija != null)
    {
        konekcija.Close();
    }
}
}
#endregion
}

```

I u modelu **Zaposleni** se nalaze metode analogne prethodnim klasama. Razlika je jedino što u ovoj klasi ne postoji lista sa identifikacionim brojevima drugih objekata, jer prema inicijalnoj ideji, ukoliko korisnik proba da obriše objekat zaposlenog koji je povezan kao strani ključ na radni nalog (ili više radnih naloga), prijavice mu grešku, tjs. izbaciće **SqlException** u statičkoj metodi **Obrisi(string tabela, string idKolona, int idVrednost)** iz klase **AutoServisData**.

6.4.1.4. Klasa RadniNalog

```

class RadniNalog : IData<RadniNalog>
{
    private static SqlConnection konekcija = Konekcija.KreirajKonekciju();

    #region privatne promenljive
    private int _id;
    private DateTime _datumOtvaranja;
    private DateTime _datumZatvaranja;
    private Vozilo _vozilo;
    private Zaposleni _zaposleni;
    #endregion

    #region getteri i setteri
    public int Id
    {
        get { return _id; }
        set { _id = value; }
    }
    public DateTime DatumOtvaranja
    {
        get { return _datumOtvaranja; }
        set { _datumOtvaranja = value; }
    }
    public DateTime DatumZatvaranja

```

```

{
    get { return _datumZatvaranja; }
    set { _datumZatvaranja = value; }
}
public Vozilo Vozilo
{
    get { return _vozilo; }
    set { _vozilo = value; }
}
public Zaposleni Zaposleni
{
    get { return _zaposleni; }
    set { _zaposleni = value; }
}
#endregion

#region konstruktor
public RadniNalog()
{
}
#endregion

#region metode
public void Sacuvaj()
{
    string sqlUpit = @"insert into tblRadniNalog (DatumOtvaranja, DatumZatvaranja,
        VoziloID, ZaposleniID)
        values (@datumOtvaranja, @datumZatvaranja, @voziloId,
        @zaposleniId)";

    try
    {
        konekcija.Open();
        SqlCommand cmd = new SqlCommand(sqlUpit, konekcija);
        cmd.Parameters.AddWithValue("@datumOtvaranja", DatumOtvaranja);
        cmd.Parameters.AddWithValue("@datumZatvaranja", DatumZatvaranja);
        cmd.Parameters.AddWithValue("@voziloId", Vozilo.Id);
        cmd.Parameters.AddWithValue("@zaposleniId", Zaposleni.Id);
        cmd.ExecuteNonQuery();
    }
    catch (Exception e)
    {
        MessageBox.Show($"Došlo je do greške: { e.Message }. Podaci nisu sačuvani.",
            "Greška",
            MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
    finally
    {
        {
            if (konekcija != null)
            {
                konekcija.Close();
            }
        }
    }
}
public void Obrisi()
{
    AutoServisData.Obrisi("tblRadniNalog", "RadniNalogID", Id);
}
public void Azuriraj(RadniNalog noviRadniNalog)
{
    try
    {
        string sqlUpit = @"update tblRadniNalog set DatumOtvaranja=@datumOtvaranja,

```

```

        DatumZatvaranja=@datumZatvaranja,
        VoziloID=@voziloId, ZaposleniID=@zaposleniId
        where RadniNalogID=@id";

    konekcija.Open();
    SqlCommand cmd = new SqlCommand(sqlUpit, konekcija);
    cmd.Parameters.AddWithValue("@id", Id);
    cmd.Parameters.AddWithValue("@datumOtvaranja", noviRadniNalog.DatumOtvaranja);
    cmd.Parameters.AddWithValue("@datumZatvaranja", noviRadniNalog.DatumZatvaranja);
    cmd.Parameters.AddWithValue("@voziloId", noviRadniNalog.Vozilo.Id);
    cmd.Parameters.AddWithValue("@zaposleniId", noviRadniNalog.Zaposleni.Id);
    cmd.ExecuteNonQuery();
}
catch (Exception e)
{
    MessageBox.Show($"Došlo je do greške: { e.Message }.
                    Podaci nisu ažurirani.", "Greška",
    MessageBoxButtons.OK, MessageBoxIcon.Error);
}
finally
{
    if (konekcija != null)
    {
        konekcija.Close();
    }
}
}
public bool PostojiDuplikat()
{
    try
    {
        string sqlUpit = @"select count(*) from tblRadniNalog where
                        DatumOtvaranja=@datumOtvaranja AND
                        DatumZatvaranja=@datumZatvaranja AND
                        VoziloID=@voziloId AND ZaposleniID=@zaposleniId";

        konekcija.Open();
        SqlCommand cmd = new SqlCommand(sqlUpit, konekcija);
        cmd.Parameters.AddWithValue("@datumOtvaranja", DatumOtvaranja);
        cmd.Parameters.AddWithValue("@datumZatvaranja", DatumZatvaranja);
        cmd.Parameters.AddWithValue("@voziloId", Vozilo.Id);
        cmd.Parameters.AddWithValue("@zaposleniId", Zaposleni.Id);
        int rez = Convert.ToInt32(cmd.ExecuteScalar());
        if (rez > 0)
        {
            return true;
        }
        else
        {
            return false;
        }
    }
    catch (Exception e)
    {
        MessageBox.Show($"Došlo je do greške prilikom provere
                        duplikata u bazi: { e.Message }", "Greška",
        MessageBoxButtons.OK, MessageBoxIcon.Error);
        return false;
    }
    finally
    {
        if (konekcija != null)
        {
            konekcija.Close();
        }
    }
}

```

```

    }
}
public int BrojDelova()
{
    return AutoServisData.BrojRedovaUBazi("tblDelovi", "RadniNalogID", Id.ToString());
}
public int BrojRadova()
{
    return AutoServisData.BrojRedovaUBazi("tblIzvrzeniRadovi",
        "RadniNalogID", Id.ToString());
}
public double IznosDelova()
{
    if (BrojDelova() <= 0)
    {
        return 0;
    }

    try
    {
        string sqlUpit = @"select Sum(Kolicina*Cena) from tblDelovi
            where RadniNalogID=@id;";
        konekcija.Open();
        SqlCommand cmd = new SqlCommand(sqlUpit, konekcija);
        cmd.Parameters.AddWithValue("@id", Id);
        double rez = Convert.ToDouble(cmd.ExecuteScalar());
        return rez;
    }
    catch (Exception e)
    {
        MessageBox.Show($"Došlo je do greške prilikom računanja
            iznosa delova: { e.Message }", "Greška",
            MessageBoxButton.OK, MessageBoxImage.Error);
        return -1;
    }
    finally
    {
        if (konekcija != null)
        {
            konekcija.Close();
        }
    }
}
public double IznosRadova()
{
    if (BrojRadova() <= 0)
    {
        return 0;
    }

    try
    {
        string sqlUpit = @"select Sum(Kolicina*Cena)
            from tblIzvrzeniRadovi where RadniNalogID=@id;";
        konekcija.Open();
        SqlCommand cmd = new SqlCommand(sqlUpit, konekcija);
        cmd.Parameters.AddWithValue("@id", Id);
        double rez = Convert.ToDouble(cmd.ExecuteScalar());
        return rez;
    }
    catch (Exception e)
    {
        MessageBox.Show($"Došlo je do greške prilikom
            računanja iznosa radova: { e.Message }", "Greška",
            MessageBoxButton.OK, MessageBoxImage.Error);
    }
}

```

```

        return -1;
    }
    finally
    {
        if (konekcija != null)
        {
            konekcija.Close();
        }
    }
}

public static DataTable ListaNaloga(string filter)
{
    string sqlUpit = @"SELECT RadniNalogID as 'ID', CONVERT(VARCHAR(10),
        DatumOtvaranja, 103) as 'Datum otvaranja',
        CONVERT(VARCHAR(10), DatumZatvaranja, 103) as 'Datum zatvaranja',
        (NazivMarke + ' ' + NazivModela) as 'Vozilo',
        (ImeVlasnika + ' ' + PrezimeVlasnika) as 'Vlasnik',
        (ImeZaposlenog + ' ' + PrezimeZaposlenog) as 'Zaposleni'
    FROM tblRadniNalog join tblVozilo on
        tblRadniNalog.VoziloID=tblVozilo.VoziloID
    join tblZaposleni on
        tblRadniNalog.ZaposleniID=tblZaposleni.ZaposleniID
    join tblModel on tblVozilo.ModelID=tblModel.ModelID
    join tblMarka on tblModel.MarkaID=tblMarka.MarkaID
    join tblVlasnik on tblVozilo.VlasnikID=tblVlasnik.VlasnikID
    WHERE RadniNalogID like '%' + filter + @%"' or
        CONVERT(VARCHAR(10), DatumOtvaranja, 103) like '%' + filter + @%"'
        or CONVERT(VARCHAR(10), DatumZatvaranja, 103)
        like '%' + filter + @%"'
        or (NazivMarke + ' ' + NazivModela) like '%' + filter + @%"'
        or (ImeVlasnika + ' ' + PrezimeVlasnika) like '%' + filter + @%"'
        or (ImeZaposlenog + ' ' + PrezimeZaposlenog)
        like '%' + filter + @%"';";

    return AutoServisData.UcitajPodatke(sqlUpit);
}

public static RadniNalog UcitajNalog(int id)
{
    try
    {
        string sqlUpit = @"select * from tblRadniNalog where RadniNalogID=@id";
        RadniNalog nalog = new RadniNalog();
        konekcija.Open();
        SqlCommand cmd = new SqlCommand(sqlUpit, konekcija);
        cmd.Parameters.AddWithValue("@id", id);
        SqlDataReader citac = cmd.ExecuteReader();
        while (citac.Read())
        {
            nalog.Id = Convert.ToInt32(citac["RadniNalogID"]);
            nalog.DatumOtvaranja = Convert.ToDateTime(citac["DatumOtvaranja"]);
            nalog.DatumZatvaranja = Convert.ToDateTime(citac["DatumZatvaranja"]);

            nalog.Vozilo = Vozilo.UcitajVozilo(Convert.ToInt32(citac["VoziloID"]));
            nalog.Zaposleni = Zaposleni.UcitajZaposlenog(
                Convert.ToInt32(citac["ZaposleniID"]));
        }
        return nalog;
    }
    catch (Exception e)
    {
        MessageBox.Show($"Došlo je do greške: { e.Message }.
            Podaci nisu učitani.", "Greška",
            MessageBoxButtons.OK, MessageBoxIcon.Error);
        return null;
    }
}

```

```

    }
    finally
    {
        if (konekcija != null)
        {
            konekcija.Close();
        }
    }
}
public static void ObrisiSveRadneNaloge(int voziloId)
//brise sve radne naloge za odredjeno vozilo
{
    AutoServisData.Obrisi("tblRadniNalog", "VoziloID", voziloId);
}

#endregion
}

```

Pored standardnih metoda, koje su već opisane, u modelu **RadniNalog** definisane su i sledeće metode:

Metode **BrojDelova()** i **BrojRadova()** su tipa **int** i koriste se za utvrđivanje da li za radni nalog postoje u bazi uneti objekti modela **Radovi** i **Delovi** (tjs. njihove osobine), preciznije, da li za radni nalog postoje radovi i delovi. Metode vraćaju rezultat pozivanjem ranije objašnjene statičke metode **BrojRedovaUBazi(string imeTabele, string polje, string uslov)** iz klase **AutoServisData**.

Metode **IznosDelova()** i **IznosRadova()** su metode tipa **double**, koje vraćaju vrednost radova i delova referenciranih na radni nalog. Ukoliko za radni nalog ne postoje delovi i radovi (tjs. metode **BrojDelova()** i **BrojRadova()** vrate rezultat 0) rezultat i ovih metoda će biti 0.

Pored njih, u modelu **RadniNalog** definisana je i metoda **ObrisiSveRadneNaloge(int Voziloid)** koja se koristi za brisanje svih radnih naloga referenciranih na jedno vozilo. Metoda je statička i tipa void, a izvršava se pozivanjem metode **Obrisi(string tabela, string idKolona, int idVrednost)** iz klase **AutoServisData**.

6.4.1.5 Faktura

```

class Faktura : IData<Faktura>
{
    private static SqlConnection konekcija = Konekcija.KreirajKonekciju();

    #region privatne promenljive
    private int _id;
    private DateTime _datum;
    private DateTime _valuta;
    private int _brojFiskalnogRacuna;
    private RadniNalog _radniNalog;
    #endregion
}

```



```

#region getteri i setteri
public int Id
{
    get { return _id; }
    set { _id = value; }
}
public DateTime Datum
{
    get { return _datum; }
    set { _datum = value; }
}
public DateTime Valuta
{
    get { return _valuta; }
    set { _valuta = value; }
}
public int BrojFiskalnogRacuna
{
    get { return _brojFiskalnogRacuna; }
    set { _brojFiskalnogRacuna = value; }
}
public RadniNalog RadniNalog
{
    get { return _radniNalog; }
    set { _radniNalog = value; }
}
#endregion

#region konstruktor
public Faktura()
{

}
#endregion

#region metode
public void Sacuvaj()
{
    string sqlUpit = @"insert into tblFaktura(Datum, Valuta, BrojFiskalnogRacuna,
        RadniNalogID)
        values (@datum, @valuta, @brojFisk, @radniNalogId)";

    try
    {
        konekcija.Open();
        SqlCommand cmd = new SqlCommand(sqlUpit, konekcija);
        cmd.Parameters.AddWithValue("@datum", Datum);
        cmd.Parameters.AddWithValue("@valuta", Valuta);
        cmd.Parameters.AddWithValue("@brojFisk", BrojFiskalnogRacuna);
        cmd.Parameters.AddWithValue("@radniNalogId", RadniNalog.Id);
        cmd.ExecuteNonQuery();
    }
    catch (Exception e)
    {
        MessageBox.Show($"Došlo je do greške: { e.Message }.
            Podaci nisu sačuvani.", "Greška",
            MessageBoxButton.OK, MessageBoxImage.Error);
    }
    finally
    {
        if (konekcija != null)
        {
            konekcija.Close();
        }
    }
}

```

```

    }
}
public void Obrisi()
{
    AutoServisData.Obrisi("tblFaktura", "FakturaID", Id);
}
public void Azuriraj(Faktura novaFaktura)
{
    try
    {
        string sqlUpit = @"update tblFaktura set Datum=@datum,
                               Valuta=@valuta, BrojFiskalnogRacuna=@brojFisk,
                               RadniNalogID=@radniNalogId where FakturaID=@id";

        konekcija.Open();
        SqlCommand cmd = new SqlCommand(sqlUpit, konekcija);
        cmd.Parameters.AddWithValue("@id", Id);
        cmd.Parameters.AddWithValue("@datum", novaFaktura.Datum);
        cmd.Parameters.AddWithValue("@valuta", novaFaktura.Valuta);
        cmd.Parameters.AddWithValue("@brojFisk", novaFaktura.BrojFiskalnogRacuna);
        cmd.Parameters.AddWithValue("@radniNalogId", novaFaktura.RadniNalog.Id);
        cmd.ExecuteNonQuery();
    }
    catch (Exception e)
    {
        MessageBox.Show($"Došlo je do greške: { e.Message }.
                          Podaci nisu ažurirani.", "Greška",
                          MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
    finally
    {
        if (konekcija != null)
        {
            konekcija.Close();
        }
    }
}
public bool PostojiDuplikat()
{
    try
    {
        string sqlUpit = @"select count(*) from tblFaktura
                          where RadniNalogID=@radniNalogId or BrojFiskalnogRacuna=@brojFisk;";
        konekcija.Open();
        SqlCommand cmd = new SqlCommand(sqlUpit, konekcija);
        cmd.Parameters.AddWithValue("@radniNalogId", RadniNalog.Id);
        cmd.Parameters.AddWithValue("@brojFisk", BrojFiskalnogRacuna);
        int rez = Convert.ToInt32(cmd.ExecuteScalar());
        if (rez > 0)
        {
            return true;
        }
        else
        {
            return false;
        }
    }
    catch (Exception e)
    {
        MessageBox.Show($"Došlo je do greške prilikom provere
                          duplikata u bazi: { e.Message }", "Greška",
                          MessageBoxButtons.OK, MessageBoxIcon.Error);
        return false;
    }
}

```

```

        finally
        {
            if (konekcija != null)
            {
                konekcija.Close();
            }
        }
    }
}

public static DataTable ListaFaktura(string filter)
{
    string sqlUpit = @"SELECT tblFaktura.FakturaID as 'ID',
        CONVERT(VARCHAR(10), tblFaktura.Datum, 103) as 'Datum',
        CONVERT(VARCHAR(10), tblFaktura.Valuta, 103) as 'Valuta',
        tblFaktura.BrojFiskalnogRacuna as 'Fiskalni račun',
        tblRadniNalog.RadniNalogID as 'Radni nalog ID',
        tblMarka.NazivMarke + ' ' +
            tblmodel.NazivModela as 'Vozilo',
        tblVlasnik.ImeVlasnika + ' ' +
            tblVlasnik.PrezimeVlasnika as 'Vlasnik'
    FROM   tblFaktura join tblRadniNalog on
        tblFaktura.RadniNalogID=tblRadniNalog.RadniNalogID
        join tblVozilo on
            tblRadniNalog.VoziloID = tblVozilo.VoziloID
        join tblVlasnik on
            tblVozilo.VlasnikID = tblVlasnik.VlasnikID
        join tblModel on tblVozilo.ModelID = tblModel.ModelID
        join tblMarka on tblModel.MarkaID = tblMarka.MarkaID
    WHERE  tblFaktura.FakturaID like '%" + filter + @"%' or
        CONVERT(VARCHAR(10), tblFaktura.Datum, 103)
            like '%" + filter + @"%' or
        CONVERT(VARCHAR(10), tblFaktura.Valuta, 103)
            like '%" + filter + @"%' or
        tblFaktura.BrojFiskalnogRacuna
            like '%" + filter + @"%' or
        tblRadniNalog.RadniNalogID
            like '%" + filter + @"%' or
        (tblMarka.NazivMarke + ' ' + tblmodel.NazivModela)
            like '%" + filter + @"%' or
        (tblVlasnik.ImeVlasnika + ' ' + tblVlasnik.PrezimeVlasnika)
            like '%" + filter + @"%';";

    return AutoServisData.UcitajPodatke(sqlUpit);
}

public static Faktura UcitajFakturu(int id)
{
    try
    {
        string sqlUpit = @"select * from tblFaktura where FakturaID=@id";
        Faktura faktura = new Faktura();
        konekcija.Open();
        SqlCommand cmd = new SqlCommand(sqlUpit, konekcija);
        cmd.Parameters.AddWithValue("@id", id);
        SqlDataReader citac = cmd.ExecuteReader();
        while (citac.Read())
        {
            faktura.Id = Convert.ToInt32(citac["FakturaID"]);
            faktura.Datum = Convert.ToDateTime(citac["Datum"]);
            faktura.Valuta = Convert.ToDateTime(citac["Valuta"]);
            faktura.BrojFiskalnogRacuna = Convert.ToInt32(
                citac["BrojFiskalnogRacuna"]);
            faktura.RadniNalog = RadniNalog.UcitajNalog(
                Convert.ToInt32(citac["RadniNalogID"]));
        }
        return faktura;
    }
}

```

```

    }
    catch (Exception e)
    {
        MessageBox.Show($"Došlo je do greške: { e.Message }.
                          Podaci nisu učitani.", "Greška",
                          MessageBoxButton.OK, MessageBoxImage.Error);
        return null;
    }
    finally
    {
        if (konekcija != null)
        {
            konekcija.Close();
        }
    }
}

public static void ObrisiSveFakture(int radniNalogId)
//brise sve fakture za odredjen radni nalog
{
    AutoServisData.Obrisi("tblFaktura", "RadniNalogID", radniNalogId);
}
#endregion
}

```

Metode u modelu **Faktura** analogne su ranije predstavljanim modelima, tjs. klasama, uz napomenu da i **Faktura**, kao i model **RadniNalog**, poseduje metodu za brisanje svih fakture, međutim ovaj put se kao argument prosleđuje **ID** broj radnog naloga. Statička metoda tipa **void ObrisiSveFakture(int radniNalogId)** se izvršava pozivanjem metode **Obrisi(string tabela, string idKolona, int idVrednost)** iz klase **AutoServisData**.

6.4.1.6. Garancija

```

class Garancija : IData<Garancija>
{
    private static SqlConnection konekcija = Konekcija.KreirajKonekciju();

    #region privatne promenljive
    private int _id;
    private string _opis;
    private int _rokVazenja;
    private Faktura _faktura;
    #endregion

    #region getteri i setteri
    public int Id
    {
        get { return _id; }
        set { _id = value; }
    }
    public Faktura Faktura
    {
        get { return _faktura; }
        set { _faktura = value; }
    }
    public string Opis

```

```

{
    get { return _opis; }
    set { _opis = value; }
}
public int RokVazenja
{
    get { return _rokVazenja; }
    set { _rokVazenja = value; }
}
#endregion

#region konstruktor
public Garancija()
{

}
#endregion

#region metode
public void Sacuvaj()
{
    string sqlUpit = @"insert into tblGarancija(FakturaID, Opis, RokVazenja)
                      values (@fakturaId, @opis, @rokVazenja)";

    try
    {
        konekcija.Open();
        SqlCommand cmd = new SqlCommand(sqlUpit, konekcija);
        cmd.Parameters.AddWithValue("@fakturaId", Faktura.Id);
        cmd.Parameters.AddWithValue("@opis", Opis);
        cmd.Parameters.AddWithValue("@rokVazenja", RokVazenja);
        cmd.ExecuteNonQuery();
    }
    catch (Exception e)
    {
        MessageBox.Show($"Došlo je do greške: { e.Message }.
                          Podaci nisu sačuvani.", "Greška",
                          MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
    finally
    {
        if (konekcija != null)
        {
            konekcija.Close();
        }
    }
}

public void Obrisi()
{
    AutoServisData.Obrisi("tblGarancija", "GarancijaID", Id);
}

public void Azuriraj(Garancija novaGarancija)
{
    try
    {
        string sqlUpit = @"update tblGarancija set FakturaID=@fakturaId,
                      Opis=@opis, RokVazenja=@rokVazenja
                      where GarancijaID=@id";

        konekcija.Open();
        SqlCommand cmd = new SqlCommand(sqlUpit, konekcija);
        cmd.Parameters.AddWithValue("@id", Id);
        cmd.Parameters.AddWithValue("@fakturaId", novaGarancija.Faktura.Id);
    }
}

```

```

        cmd.Parameters.AddWithValue("@opis", novaGarancija.Opis);
        cmd.Parameters.AddWithValue("@rokVazenja", novaGarancija.RokVazenja);
        cmd.ExecuteNonQuery();
    }
    catch (Exception e)
    {
        MessageBox.Show($"Došlo je do greške: { e.Message }.
            Podaci nisu ažurirani.", "Greška",
            MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
    finally
    {
        if (konekcija != null)
        {
            konekcija.Close();
        }
    }
}

public bool PostojiDuplikat()
{
    try
    {
        string sqlUpit = @"select count(*) from tblGarancija
            where FakturaID=@fakturaId and Opis=@opis and RokVazenja=@rokVazenja;";
        konekcija.Open();
        SqlCommand cmd = new SqlCommand(sqlUpit, konekcija);
        cmd.Parameters.AddWithValue("@fakturaId", Faktura.Id);
        cmd.Parameters.AddWithValue("@opis", Opis);
        cmd.Parameters.AddWithValue("@rokVazenja", RokVazenja);
        int rez = Convert.ToInt32(cmd.ExecuteScalar());
        if (rez > 0)
        {
            return true;
        }
        else
        {
            return false;
        }
    }
    catch (Exception e)
    {
        MessageBox.Show($"Došlo je do greške prilikom provere
            duplikata u bazi: { e.Message }", "Greška",
            MessageBoxButtons.OK, MessageBoxIcon.Error);
        return false;
    }
    finally
    {
        if (konekcija != null)
        {
            konekcija.Close();
        }
    }
}

public static DataTable ListaGarancija(int id, string filter)
    //prikazuje listu garancija za odredjenu fakturu
{
    string sqlUpit = @"SELECT tblGarancija.GarancijaID as 'ID garancije',
        tblFaktura.FakturaID as 'ID fakture',
        tblVlasnik.ImeVlasnika + ' ' +
        tblVlasnik.PrezimeVlasnika as 'Kupac',
        tblGarancija.Opis,
        CONVERT(VARCHAR(10),
            (DATEADD(month, tblGarancija.RokVazenja, tblFaktura.Datum)), 103)
    ";
}

```

```

        as 'Važi do datuma:'
        FROM tblGarancija join tblFaktura on
        tblGarancija.FakturaID = tblFaktura.FakturaID
        join tblRadniNalog on
        tblFaktura.RadniNalogID = tblRadniNalog.RadniNalogID
        join tblVozilo on tblRadniNalog.VoziloID = tblVozilo.VoziloID
        join tblVlasnik on tblVozilo.VlasnikID = tblVlasnik.VlasnikID
        where tblFaktura.FakturaID=" + id + @" AND
        (tblGarancija.GarancijaID like '%"
+ filter + @"%" OR tblFaktura.FakturaID like '%" + filter + @"%"
        OR (tblVlasnik.ImeVlasnika + ' ' + tblVlasnik.PrezimeVlasnika)
        like '%" + filter + @"%"
        OR tblGarancija.Opis like '%" + filter + @"%"
        OR (DATEADD(month, tblGarancija.RokVazenja, tblFaktura.Datum))
        like '%" + filter + @"%"
        OR tblGarancija.RokVazenja like '%" + filter + @"%');";

    return AutoServisData.UcitajPodatke(sqlUpit);
}
public static Garancija UcitajGaranciju(int id)
{
    try
    {
        string sqlUpit = @"select * from tblGarancija where GarancijaID=@id";
        Garancija garancija = new Garancija();
        konekcija.Open();
        SqlCommand cmd = new SqlCommand(sqlUpit, konekcija);
        cmd.Parameters.AddWithValue("@id", id);
        SqlDataReader citac = cmd.ExecuteReader();
        while (citac.Read())
        {
            garancija.Id = Convert.ToInt32(citac["GarancijaID"]);
            garancija.Opis = citac["Opis"].ToString();
            garancija.RokVazenja = Convert.ToInt32(citac["RokVazenja"]);
            garancija.Faktura = Faktura.UcitajFakturu(
                Convert.ToInt32(citac["FakturaID"]));
        }
        return garancija;
    }
    catch (Exception e)
    {
        MessageBox.Show($"Došlo je do greške: { e.Message }.
        Podaci nisu učitani.", "Greška",
        MessageBoxButtons.OK, MessageBoxIcon.Error);
        return null;
    }
    finally
    {
        if (konekcija != null)
        {
            konekcija.Close();
        }
    }
}
public static void ObrisiSveGarancije(int fakturaId)
//brise sve garancije za odredjenu fakturu
{
    AutoServisData.Obrisi("tblGarancija", "FakturaID", fakturaId);
}

public static void ObrisiSveGarancijeZaRadniNalog(int radniNalogId)
//RADI!!!! NE DIRATI NISTA!!!
{
    try

```

```

{
    string sqlUpit = @"select tblGarancija.GarancijaID, tblGarancija.FakturaID,
                           tblGarancija.Opis, tblGarancija.RokVazenja
                           from tblGarancija join tblFaktura on
                           tblFaktura.FakturaID=tblGarancija.FakturaID
                           where tblRadniNalog.RadniNalogID=@id;";

    Garancija garancija;
    List<Garancija> listaGarancija = new List<Garancija>();

    konekcija.Open();
    SqlCommand cmd = new SqlCommand(sqlUpit, konekcija);
    cmd.Parameters.AddWithValue("@id", radniNalogId);
    SqlDataReader citac = cmd.ExecuteReader();
    while (citac.Read())
    {
        garancija = new Garancija();
        garancija.Id = Convert.ToInt32(citac["GarancijaID"]);
        garancija.Opis = citac["Opis"].ToString();
        garancija.RokVazenja = Convert.ToInt32(citac["RokVazenja"]);
        garancija.Faktura = Faktura.UcitajFakturu(
            Convert.ToInt32(citac["FakturaID"]));
        listaGarancija.Add(garancija);
    }

    foreach(Garancija tempGarancija in listaGarancija)
    {
        AutoServisData.Obrisi("tblGarancija", "GarancijaID", tempGarancija.Id);
    }
}
catch (Exception e)
{
    MessageBox.Show($"Došlo je do greške prilikom brisanja
                    garancije za radni nalog: { e.Message }.", "Greška",
    MessageBoxButtons.OK, MessageBoxIcon.Error);
}
finally
{
    if (konekcija != null)
    {
        konekcija.Close();
    }
}
}
#endregion
}

```

U modelu, tj. klasi **Garancija** definisana je, pored ostalih, i dodatna metoda **ObrisiSveGarancijeZaRadniNalog(int radniNalogId)** koja služi za brisanje svih garancija otvorenih za fakturu koja je kreirana za radni nalog sa prosleđenim **ID** brojem. Razlog zašto je ova metoda kreirana zapravo leži u problemu da kada korisnik želi da obriše vozilo, moraju prvo da se obrišu svi referencirani objekti, tj. redovi iz tabela **tblRadniNalog**, **tblNaruceniRadovi**, **tblIzvršeniRadovi**, **tblDelovi**, **tblFaktura** i **tblGarancija**. Metoda je statička, tipa **void**, i izvršava se tako što se prvo izvršava upit koji povlači sve garancije referencirane na radni nalog sa prosleđenim **ID** brojem dok se one kontinualno pune u listu tipa **<Garancija>**. Zatim se prolazi kroz listu pomoću

foreach petlje i svaki put se poziva metoda **AutoServisData.Obrisi(...)**. Možda i previše komplikovano rešenje, međutim pošto je funkcionisalo savršeno iz prvog pokušaja, bez ijedne izmenjene linije koda, autor se osetio slobodnim da sve ostavi kako je prvobitno zamišljeno.

6.4.1.7. NaruceniRadovi

```
class NaruceniRadovi : IData<NaruceniRadovi>
{
    private static SqlConnection konekcija = Konekcija.KreirajKonekciju();

    #region privatne promenljive
    private int _redniBroj;
    private string _opis;
    private RadniNalog _radniNalog;
    #endregion

    #region getteri i setteri
    public int RedniBroj
    {
        get { return _redniBroj; }
        set { _redniBroj = value; }
    }
    public string Opis
    {
        get { return _opis; }
        set { _opis = value; }
    }
    public RadniNalog RadniNalog
    {
        get { return _radniNalog; }
        set { _radniNalog = value; }
    }
    #endregion

    #region konstruktor
    public NaruceniRadovi()
    {

    }
    #endregion

    #region metode
    public void Sacuvaj()
    {
        string sqlUpit = @"insert into tblNaruceniRadovi (RedniBroj, Opis, RadniNalogID)
                           values (@redniBroj, @opis, @radniNalogId)";

        try
        {
            konekcija.Open();
            SqlCommand cmd = new SqlCommand(sqlUpit, konekcija);
            cmd.Parameters.AddWithValue("@redniBroj", RedniBroj);
            cmd.Parameters.AddWithValue("@opis", Opis);
            cmd.Parameters.AddWithValue("@radniNalogId", RadniNalog.Id);
            cmd.ExecuteNonQuery();
        }
        catch (Exception e)
        {
        }
    }
}
```

```

        MessageBox.Show($"Došlo je do greške: { e.Message }.
                        Podaci nisu sačuvani.", "Greška",
        MessageBoxButtons.OK, MessageBoxImage.Error);
    }
    finally
    {
        if (konekcija != null)
        {
            konekcija.Close();
        }
    }
}
public void Obrisi()
    //improvizacija metode obrisi :))))))
{
    AutoServisData.Obrisi("tblNaruceniRadovi", $"RedniBroj={ RedniBroj }
                        AND RadniNalogID", RadniNalog.Id);

    try
    {
        string sqlUpit = @"update tblNaruceniRadovi set
                        RedniBroj=RedniBroj-1 where RadniNalogID=@id AND RedniBroj>@redniBroj";

        konekcija.Open();
        SqlCommand cmd = new SqlCommand(sqlUpit, konekcija);
        cmd.Parameters.AddWithValue("@id", RadniNalog.Id);
        cmd.Parameters.AddWithValue("@redniBroj", RedniBroj);
        cmd.ExecuteNonQuery();
    }
    catch (Exception e)
    {
        MessageBox.Show($"Došlo je do greške prilikom ažuriranja
                        rednih brojeva: { e.Message }", "Greška",
        MessageBoxButtons.OK, MessageBoxImage.Error);
    }
    finally
    {
        if (konekcija != null)
        {
            konekcija.Close();
        }
    }
}
public void Azuriraj(NaruceniRadovi noviRadovi)
{
    try
    {
        string sqlUpit = @"update tblNaruceniRadovi set
                        Opis=@opis where RedniBroj=@redniBroj AND RadniNalogID=@id";

        konekcija.Open();
        SqlCommand cmd = new SqlCommand(sqlUpit, konekcija);
        cmd.Parameters.AddWithValue("@opis", noviRadovi.Opis);
        cmd.Parameters.AddWithValue("@redniBroj", RedniBroj);
        cmd.Parameters.AddWithValue("@id", RadniNalog.Id);
        cmd.ExecuteNonQuery();
    }
    catch (Exception e)
    {
        MessageBox.Show($"Došlo je do greške: { e.Message }.
                        Podaci nisu ažurirani.", "Greška",
        MessageBoxButtons.OK, MessageBoxImage.Error);
    }
    finally

```

```

        {
            if (konekcija != null)
            {
                konekcija.Close();
            }
        }
    }
}

public bool PostojiDuplikat()
{
    try
    {
        string sqlUpit = @"select count(*) from tblRadniNalog
                           where Opis=@opis AND RedniBroj=@redniBroj AND RadniNalogID=@id;";
        konekcija.Open();
        SqlCommand cmd = new SqlCommand(sqlUpit, konekcija);
        cmd.Parameters.AddWithValue("@opis", Opis);
        cmd.Parameters.AddWithValue("@redniBroj", RedniBroj);
        cmd.Parameters.AddWithValue("@id", RadniNalog.Id);
        int rez = Convert.ToInt32(cmd.ExecuteScalar());
        if (rez > 0)
        {
            return true;
        }
        else
        {
            return false;
        }
    }
    catch (Exception e)
    {
        MessageBox.Show($"Došlo je do greške prilikom provere
                        duplikata u bazi: { e.Message }", "Greška",
                        MessageBoxButton.OK, MessageBoxImage.Error);
        return false;
    }
}

public static DataTable ListaNarucenihRadova(int RadniNalogID)
    //i ovde moramo proslediti argument ID radnog naloga,
    //da bi prikazali samo radove za taj radni nalog
{
    string sqlUpit = $"Select RedniBroj as 'Rbr', Opis as 'Opis radova'
                     FROM tblNaruceniRadovi where RadniNalogID={ RadniNalogID }";
    return AutoServisData.UcitajPodatke(sqlUpit);
}

public static NaruceniRadovi UcitajNaruceneRadove(int redniBroj, int id)
    //vraca jednu stavku iz narucenih radova sa parametrima redniBroj i id
{
    try
    {
        string sqlUpit = @"select * from tblNaruceniRadovi
                           where RedniBroj=@redniBroj AND RadniNalogID=@id";
        NaruceniRadovi rad = new NaruceniRadovi();
        konekcija.Open();
        SqlCommand cmd = new SqlCommand(sqlUpit, konekcija);
        cmd.Parameters.AddWithValue("@redniBroj", redniBroj);
        cmd.Parameters.AddWithValue("@id", id);
        SqlDataReader citac = cmd.ExecuteReader();
        while (citac.Read())
        {
            rad.RedniBroj = Convert.ToInt32(citac["RedniBroj"]);
            rad.Opis = citac["Opis"].ToString();
            rad.RadniNalog = RadniNalog.UcitajNalog(
                Convert.ToInt32(citac["RadniNalogID"]));
        }
    }
}

```

```

        return rad;
    }
    catch (Exception e)
    {
        MessageBox.Show($"Došlo je do greške: { e.Message }.
        Podaci nisu učitani.", "Greška",
        MessageBoxButtons.OK, MessageBoxIcon.Error);
        return null;
    }
    finally
    {
        if (konekcija != null)
        {
            konekcija.Close();
        }
    }
}
public static void ObrisiSveNaruceneRadove(int radniNalogId)
    //brise sve narucene radove za odredjen radni nalog
{
    AutoServisData.Obrisi("tblNaruceniRadovi", "RadniNalogID", radniNalogId);
}
#endregion
}

```

Model, odnosno klasa **NaruceniRadovi** morala je u određenim segmentima biti modifikovana, tačnije moralo se odstupiti od naviknutog korišćenja nekih metoda. Razlog za ovo je inicijalna ideja da primarni ključ u tabelama **tblNaruceniRadovi**, **tblIzvršeniRadovi**, i **tblDelovi** čine kombinacija kolona **RedniBroj** i **RadniNalogID**. To je imalo dosta pozitivnih posledica, ali i negativnih kao u ovom slučaju.

Metoda **Obrisi()** je morala biti izmenjena jer je trebalo da se prosledi dva argumenta (**RadniNalogID** i **RedniBroj**). Problem je rešen jednostavnom, ali neoptimalnom improvizacijom statičke metode **AutoServisData.Obrisi(...)**. Nakon izvršavanja te linije, vrši se ažuriranje rednih brojeva u tabeli (**dekrementiraju** se redni brojevi), ali samo gde je redni broj veći od rednog broja obrisane linije i gde je, naravno, **RadniNalogID** jednak prosleđenom **ID** broju.

Statička metoda **ListaNarucenihRadova(int RadniNalogID)** sada prima kao argument i **ID** broj radnog naloga, da bi se učitali samo radovi za određen radni nalog.

Statička metoda **UcitajNaruceneRadove(int redniBroj, int id)** sada prima kao argument i redni broj da bi se učitao samo jedan rad sa jedinstvenom kombinacijom rednog broja i ID broja radnog naloga. Treba voditi računa jer je notacija ove metode čudna, pa se na prvi pogled čini da se učitava više radova, međutim objektu se dodaju osobine samo jednog naručenog rada. Neprirodnost naziva "UcitajNaruceniRad" imala je za posledicu da ostane gore navedeni naziv metode.

6.4.1.8. Delovi

```
class Delovi : IData<Delovi>
{
    private static SqlConnection konekcija = Konekcija.KreirajKonekciju();

    #region privatne promenljive
    private int _redniBroj;
    private string _sifra;
    private string _naziv;
    private string _jedinicaMere;
    private double _kolicina;
    private double _cena;
    private RadniNalog _radniNalog;
    #endregion

    #region getteri i setteri
    public int RedniBroj
    {
        get { return _redniBroj; }
        set { _redniBroj = value; }
    }
    public string Sifra
    {
        get { return _sifra; }
        set { _sifra = value; }
    }
    public string Naziv
    {
        get { return _naziv; }
        set { _naziv = value; }
    }
    public string JedinicaMere
    {
        get { return _jedinicaMere; }
        set { _jedinicaMere = value; }
    }
    public double Kolicina
    {
        get { return _kolicina; }
        set { _kolicina = value; }
    }
    public double Cena
    {
        get { return _cena; }
        set { _cena = value; }
    }
    public RadniNalog RadniNalog
    {
        get { return _radniNalog; }
        set { _radniNalog = value; }
    }
    #endregion

    #region konstruktor
    public Delovi()
    {
    }
    #endregion

    #region metode
    public void Sacuvaj()
```

```

{
    string sqlUpit = @"insert into tblDelovi(RadniBroj, Sifra, Naziv, JedinicaMere,
        Kolicina, Cena, RadniNalogID)
        values (@radniBroj, @sifra, @naziv, @jedinicaMere,
            @kolicina, @cena, @radniNalogId)";

    try
    {
        konekcija.Open();
        SqlCommand cmd = new SqlCommand(sqlUpit, konekcija);
        cmd.Parameters.AddWithValue("@radniBroj", RadniBroj);
        cmd.Parameters.AddWithValue("@sifra", Sifra);
        cmd.Parameters.AddWithValue("@naziv", Naziv);
        cmd.Parameters.AddWithValue("@jedinicaMere", JedinicaMere);
        cmd.Parameters.AddWithValue("@kolicina", Kolicina);
        cmd.Parameters.AddWithValue("@cena", Cena);
        cmd.Parameters.AddWithValue("@radniNalogId", RadniNalog.Id);
        cmd.ExecuteNonQuery();
    }
    catch (Exception e)
    {
        MessageBox.Show($"Došlo je do greške: { e.Message }.
            Podaci nisu sačuvani.", "Greška",
            MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
    finally
    {
        if (konekcija != null)
        {
            konekcija.Close();
        }
    }
}

public void Obrisi()
    //moramo da improvizujemo da bismo obrisali samo jedan deo za jedan radni nalog :)
{
    AutoServisData.Obrisi("tblDelovi", $"RadniNalogID={ RadniNalog.Id }
        AND RadniBroj", RadniBroj);

    try
    {
        string sqlUpit = @"update tblDelovi set
            RadniBroj=RadniBroj-1 where RadniNalogID=@id AND RadniBroj>@redniBroj";

        konekcija.Open();
        SqlCommand cmd = new SqlCommand(sqlUpit, konekcija);
        cmd.Parameters.AddWithValue("@id", RadniNalog.Id);
        cmd.Parameters.AddWithValue("@redniBroj", RadniBroj);
        cmd.ExecuteNonQuery();
    }
    catch (Exception e)
    {
        MessageBox.Show($"Došlo je do greške prilikom ažuriranja
            rednih brojeva: { e.Message }", "Greška",
            MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
    finally
    {
        if (konekcija != null)
        {
            konekcija.Close();
        }
    }
}
}

```

```

public void Azuriraj(Delovi noviDeo)
{
    try
    {
        string sqlUpit = @"update tblDelovi set Sifra=@sifra,
                                Naziv=@naziv, JedinicaMere=@jedinicaMere, Kolicina=@kolicina,
                                Cena=@cena where RadniNalogID=@radniNalogId
                                AND RedniBroj=@redniBroj;";

        konekcija.Open();
        SqlCommand cmd = new SqlCommand(sqlUpit, konekcija);
        cmd.Parameters.AddWithValue("@redniBroj", RedniBroj);
        cmd.Parameters.AddWithValue("@sifra", noviDeo.Sifra);
        cmd.Parameters.AddWithValue("@naziv", noviDeo.Naziv);
        cmd.Parameters.AddWithValue("@jedinicaMere", noviDeo.JedinicaMere);
        cmd.Parameters.AddWithValue("@kolicina", noviDeo.Kolicina);
        cmd.Parameters.AddWithValue("@cena", noviDeo.Cena);
        cmd.Parameters.AddWithValue("@radniNalogId", RadniNalog.Id);
        cmd.ExecuteNonQuery();
    }
    catch (Exception e)
    {
        MessageBox.Show($"Došlo je do greške: { e.Message }.
                        Podaci nisu ažurirani.", "Greška",
                        MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
    finally
    {
        if (konekcija != null)
        {
            konekcija.Close();
        }
    }
}

public bool PostojiDuplikat()
{
    try
    {
        string sqlUpit = @"select count(*) from tblDelovi where
                        Sifra=@sifra AND RedniBroj=@redniBroj AND RadniNalogID=@radniNalogId;";
        konekcija.Open();
        SqlCommand cmd = new SqlCommand(sqlUpit, konekcija);
        cmd.Parameters.AddWithValue("@redniBroj", RedniBroj);
        cmd.Parameters.AddWithValue("@sifra", Sifra);
        cmd.Parameters.AddWithValue("@radniNalogId", RadniNalog.Id);
        int rez = Convert.ToInt32(cmd.ExecuteScalar());
        if (rez > 0)
        {
            return true;
        }
        else
        {
            return false;
        }
    }
    catch (Exception e)
    {
        MessageBox.Show($"Došlo je do greške prilikom provere
                        duplikata u bazi: { e.Message }", "Greška",
                        MessageBoxButtons.OK, MessageBoxIcon.Error);
        return false;
    }
}

```

```

    }

    public static DataTable ListaDelova(int id)
    //i ovde moramo proslediti argument ID radnog naloga, da bi prikazali samo delove za ta
j radni nalog
    {
        string sqlUpit = $"Select RedniBroj as 'Rbr', Sifra as 'Šifra',
            Naziv as 'Naziv', JedinicaMere as 'Jmr',
            Cast(Kolicina as numeric(36,3)) as 'Količina',
            Cast(Cena as numeric(36,2)) as 'Cena'
            FROM tblDelovi where RadniNalogID={ id }";
        return AutoServisData.UcitajPodatke(sqlUpit);
    }

    public static Delovi UcitajDeo(int redniBroj, int id)
    //vraca jednu stavku iz narucenih delova sa parametrima redniBroj i id
    {
        try
        {
            string sqlUpit = @"select * from tblDelovi
                where RedniBroj=@redniBroj AND RadniNalogID=@id";
            Delovi deo = new Delovi();
            konekcija.Open();
            SqlCommand cmd = new SqlCommand(sqlUpit, konekcija);
            cmd.Parameters.AddWithValue("@redniBroj", redniBroj);
            cmd.Parameters.AddWithValue("@id", id);
            SqlDataReader citac = cmd.ExecuteReader();
            while (citac.Read())
            {
                deo.RedniBroj = Convert.ToInt32(citac["RedniBroj"]);
                deo.Sifra = citac["Sifra"].ToString();
                deo.Naziv = citac["Naziv"].ToString();
                deo.JedinicaMere = citac["JedinicaMere"].ToString();
                deo.Kolicina = Convert.ToDouble(citac["Kolicina"]);
                deo.Cena = Convert.ToDouble(citac["Cena"]);
                deo.RadniNalog = RadniNalog.UcitajNalog(
                    Convert.ToInt32(citac["RadniNalogID"]));
            }
            return deo;
        }
        catch (Exception e)
        {
            MessageBox.Show($"Došlo je do greške: { e.Message }.
                Podaci nisu učitani.", "Greška",
                MessageBoxButton.OK, MessageBoxImage.Error);
            return null;
        }
        finally
        {
            if (konekcija != null)
            {
                konekcija.Close();
            }
        }
    }

    public static void ObrisiSveDelove(int id)
    //brise sve delove za odredjen radni nalog
    {
        AutoServisData.Obrisi("tblDelovi", "RadniNalogID", id);
    }
    #endregion
}

```


U modelu **Delovi**, situacije u potpunosti analogna modelu **NaruceniRadovi**. Nisu potrebna dodatna objašnjenja.

6.4.1.9. IzvrсениRadovi

```
class IzvrсениRadovi : IData<IzvrсениRadovi>
{
    private static SqlConnection konekcija = Konekcija.KreirajKonekciju();

    #region privatne promenljive
    private int _redniBroj;
    private string _naziv;
    private string _jedinicaMere;
    private double _kolicina;
    private double _cena;
    private RadniNalog _radniNalog;
    #endregion

    #region getteri i setteri
    public int RedniBroj
    {
        get { return _redniBroj; }
        set { _redniBroj = value; }
    }
    public string Naziv
    {
        get { return _naziv; }
        set { _naziv = value; }
    }
    public string JedinicaMere
    {
        get { return _jedinicaMere; }
        set { _jedinicaMere = value; }
    }
    public double Kolicina
    {
        get { return _kolicina; }
        set { _kolicina = value; }
    }
    public double Cena
    {
        get { return _cena; }
        set { _cena = value; }
    }
    public RadniNalog RadniNalog
    {
        get { return _radniNalog; }
        set { _radniNalog = value; }
    }
    #endregion

    #region konstruktor
    public IzvrсениRadovi()
    {
    }
}
```

```

#endregion

#region metode
public void Sacuvaj()
{
    string sqlUpit = @"insert into tblIzvrzeniRadovi(RadniBroj, Naziv,
        JedinicaMere, Kolicina, Cena, RadniNalogID)
        values (@redniBroj, @naziv, @jedinicaMere, @kolicina,
        @cena, @radniNalogId)";

    try
    {
        konekcija.Open();
        SqlCommand cmd = new SqlCommand(sqlUpit, konekcija);
        cmd.Parameters.AddWithValue("@redniBroj", RadniBroj);
        cmd.Parameters.AddWithValue("@naziv", Naziv);
        cmd.Parameters.AddWithValue("@jedinicaMere", JedinicaMere);
        cmd.Parameters.AddWithValue("@kolicina", Kolicina);
        cmd.Parameters.AddWithValue("@cena", Cena);
        cmd.Parameters.AddWithValue("@radniNalogId", RadniNalog.Id);
        cmd.ExecuteNonQuery();
    }
    catch (Exception e)
    {
        MessageBox.Show($"Došlo je do greške: { e.Message }.
            Podaci nisu sačuvani.", "Greška",
            MessageBoxButton.OK, MessageBoxImage.Error);
    }
    finally
    {
        if (konekcija != null)
        {
            konekcija.Close();
        }
    }
}

public void Obrisi()
    //takodje improvizacija
{
    AutoServisData.Obrisi("tblIzvrzeniRadovi", $"RadniBroj={ RadniBroj }
        AND RadniNalogID", RadniNalog.Id);

    try
    {
        string sqlUpit = @"update tblIzvrzeniRadovi set
            RadniBroj=RadniBroj-1 where RadniNalogID=@id AND RadniBroj>@redniBroj";

        konekcija.Open();
        SqlCommand cmd = new SqlCommand(sqlUpit, konekcija);
        cmd.Parameters.AddWithValue("@id", RadniNalog.Id);
        cmd.Parameters.AddWithValue("@redniBroj", RadniBroj);
        cmd.ExecuteNonQuery();
    }
    catch (Exception e)
    {
        MessageBox.Show($"Došlo je do greške prilikom ažuriranja
            rednih brojeva: { e.Message }", "Greška",
            MessageBoxButton.OK, MessageBoxImage.Error);
    }
    finally
    {
        if (konekcija != null)
        {
            konekcija.Close();
        }
    }
}

```

```

    }
}
}
public void Azuriraj(IzvršeniRadovi noviRadovi)
{
    try
    {
        string sqlUpit = @"update tblIzvršeniRadovi set Naziv=@naziv,
                                JedinicaMere=@jedinicaMere, Kolicina=@kolicina,
                                Cena=@cena where RadniNalogID=@radniNalogId
                                AND RedniBroj=@redniBroj;";

        konekcija.Open();
        SqlCommand cmd = new SqlCommand(sqlUpit, konekcija);
        cmd.Parameters.AddWithValue("@redniBroj", RedniBroj);
        cmd.Parameters.AddWithValue("@naziv", noviRadovi.Naziv);
        cmd.Parameters.AddWithValue("@jedinicaMere", noviRadovi.JedinicaMere);
        cmd.Parameters.AddWithValue("@kolicina", noviRadovi.Kolicina);
        cmd.Parameters.AddWithValue("@cena", noviRadovi.Cena);
        cmd.Parameters.AddWithValue("@radniNalogId", RadniNalog.Id);
        cmd.ExecuteNonQuery();
    }
    catch (Exception e)
    {
        MessageBox.Show($"Došlo je do greške: { e.Message }.
                        Podaci nisu ažurirani.", "Greška",
                        MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
    finally
    {
        if (konekcija != null)
        {
            konekcija.Close();
        }
    }
}

public bool PostojiDuplikat()
{
    try
    {
        string sqlUpit = @"select count(*) from tblIzvršeniRadovi
                        where Naziv=@naziv AND RedniBroj=@redniBroj AND RadniNalogID=@radniNalogId;";
        konekcija.Open();
        SqlCommand cmd = new SqlCommand(sqlUpit, konekcija);
        cmd.Parameters.AddWithValue("@redniBroj", RedniBroj);
        cmd.Parameters.AddWithValue("@naziv", Naziv);
        cmd.Parameters.AddWithValue("@radniNalogId", RadniNalog.Id);
        int rez = Convert.ToInt32(cmd.ExecuteScalar());
        if (rez > 0)
        {
            return true;
        }
        else
        {
            return false;
        }
    }
    catch (Exception e)
    {
        MessageBox.Show($"Došlo je do greške prilikom provere
                        duplikata u bazi: { e.Message }", "Greška",
                        MessageBoxButtons.OK, MessageBoxIcon.Error);
        return false;
    }
}

```

```

    }
}

public static DataTable ListaIzvršenihRadova(int id)
{
    string sqlUpit = $"Select RedniBroj as 'Rbr', Naziv as 'Naziv',
        JedinicaMere as 'Jmr',
        Cast(Kolicina as numeric(36,3)) as 'Količina',
        Cast(Cena as numeric(36,2)) as 'Cena'
        FROM tblIzvršeniRadovi where RadniNalogID={ id }";
    return AutoServisData.UcitajPodatke(sqlUpit);
}

public static IzvršeniRadovi UcitajIzvršeneRadove(int redniBroj, int id)
{
    try
    {
        string sqlUpit = @"select * from tblIzvršeniRadovi
            where RedniBroj=@redniBroj AND RadniNalogID=@id";
        IzvršeniRadovi rad = new IzvršeniRadovi();
        konekcija.Open();
        SqlCommand cmd = new SqlCommand(sqlUpit, konekcija);
        cmd.Parameters.AddWithValue("@redniBroj", redniBroj);
        cmd.Parameters.AddWithValue("@id", id);
        SqlDataReader citac = cmd.ExecuteReader();
        while (citac.Read())
        {
            rad.RedniBroj = Convert.ToInt32(citac["RedniBroj"]);
            rad.Naziv = citac["Naziv"].ToString();
            rad.JedinicaMere = citac["JedinicaMere"].ToString();
            rad.Kolicina = Convert.ToDouble(citac["Kolicina"]);
            rad.Cena = Convert.ToDouble(citac["Cena"]);
            rad.RadniNalog = RadniNalog.UcitajNalog(
                Convert.ToInt32(citac["RadniNalogID"]));
        }
        return rad;
    }
    catch (Exception e)
    {
        MessageBox.Show($"Došlo je do greške: { e.Message }.
            Podaci nisu učitani.", "Greška",
            MessageBoxButton.OK, MessageBoxImage.Error);
        return null;
    }
    finally
    {
        if (konekcija != null)
        {
            konekcija.Close();
        }
    }
}

public static void ObrisiSveIzvršeneRadove(int id)
{
    AutoServisData.Obrisi("tblIzvršeniRadovi", "RadniNalogID", id);
}
#endregion
}

```

Model **IzvršeniRadovi** prati prethodno dva opisana modela. Nema izmena.

6.4.1.10. Marka

```
private static SqlConnection konekcija = Konekcija.KreirajKonekciju();

#region privatne promenljive
private int _id;
private string _nazivMarke;
#endregion

#region getteri i setteri
public int Id
{
    get { return _id; }
    set { _id = value; }
}
public string NazivMarke
{
    get { return _nazivMarke; }
    set { _nazivMarke = value; }
}
#endregion

#region konstruktor
public Marka()
{
}
#endregion

#region metode
public void Sacuvaj()
{
    string sqlUpit = @"insert into tblMarka(NazivMarke) values (@marka)";

    try
    {
        konekcija.Open();
        SqlCommand cmd = new SqlCommand(sqlUpit, konekcija);
        cmd.Parameters.AddWithValue("@marka", NazivMarke);
        cmd.ExecuteNonQuery();
    }
    catch (Exception e)
    {
        MessageBox.Show($"Došlo je do greške: { e.Message }.  
Podaci nisu sačuvani.", "Greška",
            MessageBoxButton.OK, MessageBoxImage.Error);
    }
    finally
    {
        if (konekcija != null)
        {
            konekcija.Close();
        }
    }
}
```

```

}
public void Obrisi()
{
    AutoServisData.Obrisi("tblMarka", "MarkaID", Id);
}
public void Azuriraj(Marka novaMarka)
{
    try
    {
        string sqlUpit = @"update tblMarka set NazivMarke=@marka where MarkaID=@id";
        konekcija.Open();
        SqlCommand cmd = new SqlCommand(sqlUpit, konekcija);
        cmd.Parameters.AddWithValue("@id", Id);
        cmd.Parameters.AddWithValue("@marka", novaMarka.NazivMarke);
        cmd.ExecuteNonQuery();
    }
    catch (Exception e)
    {
        MessageBox.Show($"Došlo je do greške: { e.Message }.
        Podaci nisu ažurirani.", "Greška",
        MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
    finally
    {
        if (konekcija != null)
        {
            konekcija.Close();
        }
    }
}
public bool PostojiDuplikat()
{
    try
    {
        string sqlUpit = @"select count(*) from tblMarka where NazivMarke=@marka;";
        konekcija.Open();
        SqlCommand cmd = new SqlCommand(sqlUpit, konekcija);
        cmd.Parameters.AddWithValue("@marka", NazivMarke);
        int rez = Convert.ToInt32(cmd.ExecuteScalar());
        if (rez > 0)
        {
            return true;
        }
        else
        {
            return false;
        }
    }
    catch (Exception e)
    {
        MessageBox.Show($"Došlo je do greške prilikom provere
        duplikata u bazi: { e.Message }", "Greška",
        MessageBoxButtons.OK, MessageBoxIcon.Error);
        return false;
    }
    finally
    {
        if (konekcija != null)
        {
            konekcija.Close();
        }
    }
}
public static DataTable ListaMarki(string filter)

```

```

{
    string sqlUpit = @"SELECT MarkaID as 'ID marke', NazivMarke as 'Naziv marke'
                        FROM tblMarka WHERE MarkaID like '%" + filter + @"%'
                        or NazivMarke like '%" + filter + @"'";
    return AutoServisData.UcitajPodatke(sqlUpit);
}
public static Marka UcitajMarku(int id)
{
    try
    {
        string sqlUpit = @"select * from tblMarka where MarkaID=@id;";
        Marka marka = new Marka();
        konekcija.Open();
        SqlCommand cmd = new SqlCommand(sqlUpit, konekcija);
        cmd.Parameters.AddWithValue("@id", id);
        SqlDataReader citac = cmd.ExecuteReader();
        while (citac.Read())
        {
            marka.Id = Convert.ToInt32(citac["MarkaID"]);
            marka.NazivMarke = citac["NazivMarke"].ToString();
        }
        return marka;
    }
    catch (Exception e)
    {
        MessageBox.Show($"Došlo je do greške: { e.Message }.
        Podaci nisu učitani.", "Greška",
        MessageBoxButtons.OK, MessageBoxIcon.Error);
        return null;
    }
    finally
    {
        if (konekcija != null)
        {
            konekcija.Close();
        }
    }
}
}
#endregion
}

```

Klasa **Marka** prati prvobitno opisanu ideju implementacije modela.

6.4.1.11. Model

```

class Model : IData<Model>
{
    private static SqlConnection konekcija = Konekcija.KreirajKonekciju();

    #region privatne promenljive
    private int _id;
    private string _nazivModela;
    private Marka _marka;
    #endregion

    #region getteri i setteri
    public int Id
    
```

```

{
    get { return _id; }
    set { _id = value; }
}
public string NazivModela
{
    get { return _nazivModela; }
    set { _nazivModela = value; }
}
public Marka Marka
{
    get { return _marka; }
    set { _marka = value; }
}
}
#endregion

#region konstruktor
public Model()
{
}
}
#endregion

#region metode
public void Sacuvaj()
{
    string sqlUpit = @"insert into tblModel(NazivModela,MarkaID) values
                        (@model,@markaId)";

    try
    {
        konekcija.Open();
        SqlCommand cmd = new SqlCommand(sqlUpit, konekcija);
        cmd.Parameters.AddWithValue("@model", NazivModela);
        cmd.Parameters.AddWithValue("@markaId", Marka.Id);
        cmd.ExecuteNonQuery();
    }
    catch (Exception e)
    {
        MessageBox.Show($"Došlo je do greške: { e.Message }.
                        Podaci nisu sačuvani.", "Greška",
                        MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
    finally
    {
        if (konekcija != null)
        {
            konekcija.Close();
        }
    }
}
public void Obrisi()
{
    AutoServisData.Obrisi("tblModel", "ModelID", Id);
}
public void Azuriraj(Model noviModel)
{
    try
    {
        string sqlUpit = @"update tblModel set NazivModela=@model,
                        MarkaID=@markaID where ModelID=@id";
        konekcija.Open();
        SqlCommand cmd = new SqlCommand(sqlUpit, konekcija);
        cmd.Parameters.AddWithValue("@id", Id);
    }
}

```



```

        cmd.Parameters.AddWithValue("@model", noviModel.NazivModela);
        cmd.Parameters.AddWithValue("@markaID", noviModel.Marka.Id);
        cmd.ExecuteNonQuery();
    }
    catch (Exception e)
    {
        MessageBox.Show($"Došlo je do greške: { e.Message }.
                           Podaci nisu ažurirani.", "Greška",
        MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
    finally
    {
        if (konekcija != null)
        {
            konekcija.Close();
        }
    }
}

public bool PostojiDuplikat()
{
    try
    {
        string sqlUpit = @"select count(*) from tblModel
                           where NazivModela=@model AND MarkaID=@markaID;";
        konekcija.Open();
        SqlCommand cmd = new SqlCommand(sqlUpit, konekcija);
        cmd.Parameters.AddWithValue("@model", NazivModela);
        cmd.Parameters.AddWithValue("@markaID", Marka.Id);
        int rez = Convert.ToInt32(cmd.ExecuteScalar());
        if (rez > 0)
        {
            return true;
        }
        else
        {
            return false;
        }
    }
    catch (Exception e)
    {
        MessageBox.Show($"Došlo je do greške prilikom provere
                           duplikata u bazi: { e.Message }", "Greška",
        MessageBoxButtons.OK, MessageBoxIcon.Error);
        return false;
    }
    finally
    {
        if (konekcija != null)
        {
            konekcija.Close();
        }
    }
}

public static DataTable ListaModela(string filter)
{
    string sqlUpit = @"SELECT tblModel.ModelID as 'ID',
                           tblMarka.NazivMarke as 'Marka', tblModel.NazivModela as 'Model'
                           FROM tblMarka join tblModel on
                           tblMarka.MarkaID = tblModel.MarkaID
                           WHERE tblMarka.MarkaID like '%" + filter + @"%' or
                           tblMarka.NazivMarke like '%" + filter + @"%'
                           OR tblModel.NazivModela like '%" + filter + @"%' or
                           (tblMarka.NazivMarke + ' ' + tblModel.NazivModela) like '%" + filter + @"%'";
}

```

```

ORDER BY tblModel.ModelID,tblMarka.NazivMarke,
        tblModel.NazivModela;";

return AutoServisData.UcitajPodatke(sqlUpit);
}
public static Model UcitajModel(int id)
{
    try
    {
        string sqlUpit = @"select * from tblModel where ModelID=@id;";
        Model model = new Model();
        konekcija.Open();
        SqlCommand cmd = new SqlCommand(sqlUpit, konekcija);
        cmd.Parameters.AddWithValue("@id", id);
        SqlDataReader citac = cmd.ExecuteReader();
        while (citac.Read())
        {
            model.Id = Convert.ToInt32(citac["ModelID"]);
            model.NazivModela = citac["NazivModela"].ToString();
            model.Marka = Marka.UcitajMarku(Convert.ToInt32(citac["MarkaID"]));
        }
        return model;
    }
    catch (Exception e)
    {
        MessageBox.Show($"Došlo je do greške: { e.Message }.
            Podaci nisu učitani.", "Greška",
            MessageBoxButtons.OK, MessageBoxIcon.Error);
        return null;
    }
    finally
    {
        if (konekcija != null)
        {
            konekcija.Close();
        }
    }
}
#endregion
}

```

I klasa **Model** prati u potpunosti prvobitno opisanu ideju implementacije modela kao reprezentacije klasa iz dijagrama klasa.

6.4.1.12. Gorivo

```

class Gorivo : IData<Gorivo>
{
    private static SqlConnection konekcija = Konekcija.KreirajKonekciju();

    #region privatne promenljive
    private int _id;
    private string _vrstaGoriva;
    #endregion

    #region getteri i setteri
    public int Id
    
```

```

{
    get { return _id; }
    set { _id = value; }
}

public string VrstaGoriva
{
    get { return _vrstaGoriva; }
    set { _vrstaGoriva = value; }
}
#endregion

#region konstruktor
public Gorivo()
{

}
#endregion

#region metode
public void Sacuvaj()
{
    string sqlUpit = @"insert into tblGorivo(VrstaGoriva) values (@vrstaGoriva)";

    try
    {
        konekcija.Open();
        SqlCommand cmd = new SqlCommand(sqlUpit, konekcija);
        cmd.Parameters.AddWithValue("@vrstaGoriva", VrstaGoriva);
        cmd.ExecuteNonQuery();
    }
    catch (Exception e)
    {
        MessageBox.Show($"Došlo je do greške: { e.Message }. Podaci nisu sačuvani.",
            "Greška",
            MessageBoxButtons.OK, MessageBoxIcon.Error);
    }

    finally
    {
        if (konekcija != null)
        {
            konekcija.Close();
        }
    }
}

public void Obrisi()
{
    AutoServisData.Obrisi("tblGorivo", "VrstaGorivaID", Id);
}

public void Azuriraj(Gorivo novaVrstaGoriva)
{
    try
    {
        string sqlUpit = @"update tblGorivo set VrstaGoriva=@vrstaGoriva
                           where VrstaGorivaID=@id";
        konekcija.Open();
        SqlCommand cmd = new SqlCommand(sqlUpit, konekcija);
        cmd.Parameters.AddWithValue("@id", Id);
        cmd.Parameters.AddWithValue("@vrstaGoriva", novaVrstaGoriva.VrstaGoriva);
        cmd.ExecuteNonQuery();
    }
    catch (Exception e)
    {

```

```

        MessageBox.Show($"Došlo je do greške: { e.Message }.
            Podaci nisu ažurirani.", "Greška",
            MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
    finally
    {
        if (konekcija != null)
        {
            konekcija.Close();
        }
    }
}
public bool PostojiDuplikat()
{
    try
    {
        string sqlUpit = @"select count(*) from tblGorivo where VrstaGoriva=@vrstaGoriva;";
        konekcija.Open();
        SqlCommand cmd = new SqlCommand(sqlUpit, konekcija);
        cmd.Parameters.AddWithValue("@vrstaGoriva", VrstaGoriva);
        int rez = Convert.ToInt32(cmd.ExecuteScalar());
        if (rez > 0)
        {
            return true;
        }
        else
        {
            return false;
        }
    }
    catch (Exception e)
    {
        MessageBox.Show($"Došlo je do greške prilikom provere
            duplikata u bazi: { e.Message }", "Greška",
            MessageBoxButtons.OK, MessageBoxIcon.Error);
        return false;
    }
}
public static DataTable ListaGoriva()
{
    string sqlUpit = @"select VrstaGorivaID as 'ID',
        VrstaGoriva as 'Vrsta goriva' from tblGorivo;";
    return AutoServisData.UcitajPodatke(sqlUpit);
}
public static Gorivo UcitajGorivo(int id)
{
    try
    {
        string sqlUpit = @"select * from tblGorivo where VrstaGorivaID=@id;";
        Gorivo gorivo = new Gorivo();
        konekcija.Open();
        SqlCommand cmd = new SqlCommand(sqlUpit, konekcija);
        cmd.Parameters.AddWithValue("@id", id);
        SqlDataReader citac = cmd.ExecuteReader();
        while (citac.Read())
        {
            gorivo.Id = Convert.ToInt32(citac["VrstaGorivaID"]);
            gorivo.VrstaGoriva = citac["VrstaGoriva"].ToString();
        }
        return gorivo;
    }
    catch (Exception e)
    {

```

```

        MessageBox.Show($"Došlo je do greške: { e.Message }.
                        Podaci nisu učitani.", "Greška",
        MessageBoxButtons.OK, MessageBoxIcon.Error);
        return null;
    }
    finally
    {
        if (konekcija != null)
        {
            konekcija.Close();
        }
    }
}
#endregion
}

```

Situacija je identična i u klasi **Gorivo**.

6.4.1.13. TipVozila

```

class TipVozila : IData<TipVozila>
{
    private static SqlConnection konekcija = Konekcija.KreirajKonekciju();

    #region privatne promenljive
    private int _id;
    private string _nazivTipaVozila;
    #endregion

    #region getteri i setteri
    public int Id
    {
        get { return _id; }
        set { _id = value; }
    }

    public string NazivTipaVozila
    {
        get { return _nazivTipaVozila; }
        set { _nazivTipaVozila = value; }
    }
    #endregion

    #region konstruktor
    public TipVozila()
    {
    }
    #endregion

    #region metode
    public void Sacuvaj()
    {
        string sqlUpit = @"insert into tblTipVozila(TipVozila) values (@tipVozila)";

        try
        {

```

```

        konekcija.Open();
        SqlCommand cmd = new SqlCommand(sqlUpit, konekcija);
        cmd.Parameters.AddWithValue("@tipVozila", NazivTipaVozila);
        cmd.ExecuteNonQuery();
    }
    catch (Exception e)
    {
        MessageBox.Show($"Došlo je do greške: { e.Message }.
                           Podaci nisu sačuvani.", "Greška",
        MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
    finally
    {
        if (konekcija != null)
        {
            konekcija.Close();
        }
    }
}
public void Obrisi()
{
    AutoServisData.Obrisi("tblTipVozila", "TipVozilaID", Id);
}
public void Azuriraj(TipVozila noviTipVozila)
{
    try
    {
        string sqlUpit = @"update tblTipVozila set TipVozila=@tipVozila
                           where TipVozilaID=@id";

        konekcija.Open();
        SqlCommand cmd = new SqlCommand(sqlUpit, konekcija);
        cmd.Parameters.AddWithValue("@id", Id);
        cmd.Parameters.AddWithValue("@tipVozila", noviTipVozila.NazivTipaVozila);
        cmd.ExecuteNonQuery();
    }
    catch (Exception e)
    {
        MessageBox.Show($"Došlo je do greške: { e.Message }.
                           Podaci nisu ažurirani.", "Greška",
        MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
    finally
    {
        if (konekcija != null)
        {
            konekcija.Close();
        }
    }
}
public bool PostojiDuplikat()
{
    try
    {
        string sqlUpit = @"select count(*) from tblTipVozila where TipVozila=@tipVozila;";
        konekcija.Open();
        SqlCommand cmd = new SqlCommand(sqlUpit, konekcija);
        cmd.Parameters.AddWithValue("@tipVozila", NazivTipaVozila);
        int rez = Convert.ToInt32(cmd.ExecuteScalar());
        if (rez > 0)
        {
            return true;
        }
        else
        {

```

```

        return false;
    }
}
catch (Exception e)
{
    MessageBox.Show($"Došlo je do greške prilikom provere
        duplikata u bazi: { e.Message }", "Greška",
        MessageBoxButtons.OK, MessageBoxIcon.Error);
    return false;
}
}
public static DataTable ListaTipovaVozila()
{
    string sqlUpit = @"select TipVozilaID as 'ID', TipVozila as 'Tip vozila'
        from tblTipVozila;";
    return AutoServisData.UcitajPodatke(sqlUpit);
}
public static TipVozila UcitajTipVozila(int id)
{
    try
    {
        string sqlUpit = @"select * from tblTipVozila where TipVozilaID=@id;";
        TipVozila tip = new TipVozila();
        konekcija.Open();
        SqlCommand cmd = new SqlCommand(sqlUpit, konekcija);
        cmd.Parameters.AddWithValue("@id", id);
        SqlDataReader citac = cmd.ExecuteReader();
        while (citac.Read())
        {
            tip.Id = Convert.ToInt32(citac["TipVozilaID"]);
            tip.NazivTipaVozila = citac["TipVozila"].ToString();
        }
        return tip;
    }
    catch (Exception e)
    {
        MessageBox.Show($"Došlo je do greške: { e.Message }.
            Podaci nisu učitani.", "Greška",
            MessageBoxButtons.OK, MessageBoxIcon.Error);
        return null;
    }
    finally
    {
        if (konekcija != null)
        {
            konekcija.Close();
        }
    }
}
}
#endregion
}

```

Takođe, situacija je identična i u klasi **TipVozila**.

6.1.5.Prozori (Windows)

Prozori (**Windows**) i stranice (**Pages**) koje se koriste kao korisnički interfejs u ovoj aplikaciji, smeštene su u folderu **Forms** u okviru projekta. Aplikacija raspolaže sa dva prozora **frmLogin.xaml** i **frmPocetna.xaml**. Početna forma služi kao **Parent** kontrola za stranice koje treba da budu prikazane na njoj. Pre nego što se na **gridu** prozora **frmPocetna.xaml** prikaže potrebna stranica, mora se zatvoriti stranica koja je prethodno bila otvorena.

Ovde će biti reči o kodu koji definiše ponašanje prozora, dok će dizajn korisničkog interfejsa biti objašnjen u sledećem poglavlju.

6.1.5.1. FrmLogin.xaml

```
public partial class frmLogin : Window
{
    public frmLogin()
    {
        InitializeComponent();
        lblPoruka.Content = "";
        txtKorisnickoIme.Text = "";
        psbLozinka.Password = "";
        txtKorisnickoIme.Focus();
    }

    private void btnOtkazi_Click(object sender, RoutedEventArgs e)
    {
        System.Windows.Application.Current.Shutdown();
    }

    private void btnUlogujSe_Click(object sender, RoutedEventArgs e)
    {
        try
        {
            if (string.IsNullOrEmpty(txtKorisnickoIme.Text))
            {
                lblPoruka.Content = "Korisničko ime nije uneto!";
                txtKorisnickoIme.Focus();
                return;
            }
            if (string.IsNullOrEmpty(psbLozinka.Password))
            {
                lblPoruka.Content = "Lozinka nije uneta!";
                psbLozinka.Focus();
                return;
            }

            string unetoKorisnickoIme = txtKorisnickoIme.Text;
            string unetaLozinka = psbLozinka.Password;

            if (AutoServisData.IspravnoKorisnickoIme(unetoKorisnickoIme) != true)
            {
                lblPoruka.Content = "Uneto korisničko ime ne postoji!";
                txtKorisnickoIme.Focus();
                return;
            }
        }
    }
}
```



```

        if (AutoServisData.IspravnaLozinka(unetoKorisnickoIme, unetaLozinka) != true)
        {
            lblPoruka.Content = "Lozinka nije ispravna!";
            psbLozinka.Focus();
            return;
        }

        lblPoruka.Content = "";
        frmPocetna pocetna = new frmPocetna();
        pocetna.Show();
        this.Close();
    }
    catch (Exception ex)
    {
        MessageBox.Show($"Došlo je do greške: { ex.Message }.
                        Podaci nisu učitani.", "Greška",
                        MessageBoxButton.OK, MessageBoxImage.Error);
    }
}
}

```

Klikom na dugme **btnOtkazi** zatvara se cela aplikacija. Klikom na dugme **btnUlogujSe** proverava se da li su uneti korisničko ime i lozinka i pozivaju se metode **AutoServisData.IspravnoKorisnickoIme(...)** i **AutoServisData.IspravnaLozinka(...)**. U slučaju bilo kakve neispravnosti iskače se iz koda. Ako su uneto korisničko ime i lozinka ispravni, otvara se glavni prozor aplikacije: **frmPocetna.xaml**

6.1.5.2. frmPocetna.xaml

```

public partial class frmPocetna : Window
{
    public frmPocetna()
    {
        InitializeComponent();
        tbNaslov.Text = "";
        lblDatum.Content = $"Datum: { DateTime.Now.ToShortDateString() }";
        lblBrojVozila.Content = $"Broj vozila u bazi:
        { AutoServisData.BrojRedovaUBazi("tblVozilo", "", "").ToString() }";
        lblNaloga.Content = $"Broj radnih naloga u bazi:
        { AutoServisData.BrojRedovaUBazi("tblRadniNalog", "", "").ToString() }";
        lblBrojFaktura.Content = $"Broj faktura u bazi:
        { AutoServisData.BrojRedovaUBazi("tblFaktura", "", "").ToString() }";
    }

    private void btnIzlaz_Click(object sender, RoutedEventArgs e)
    {
        //odjavljuje se
        frmLogin login = new frmLogin();
        this.Close();
        login.ShowDialog();
    }

    private void btnVlasnik_Click(object sender, RoutedEventArgs e)
    {
        pgVlasnici vlasnici = new pgVlasnici();
        var content = vlasnici.Content;
        vlasnici.Content = null;
    }
}

```

```

        grid.Children.Clear();
        grid.Children.Add((UIElement)content);
        vlasnici.txtID.Focus();
        tbNaslov.Text = "Vlasnici";
    }

    private void btnZaposleni_Click(object sender, RoutedEventArgs e)
    {
        pgZaposleni zaposleni = new pgZaposleni();
        var content = zaposleni.Content;
        zaposleni.Content = null;
        grid.Children.Clear();
        grid.Children.Add((UIElement)content);
        zaposleni.txtID.Focus();
        tbNaslov.Text = "Zaposleni";
    }

    private void btnMarka_Click(object sender, RoutedEventArgs e)
    {
        pgMarke marke = new pgMarke();
        var content = marke.Content;
        marke.Content = null;
        grid.Children.Clear();
        grid.Children.Add((UIElement)content);
        marke.txtIDMarke.Focus();
        tbNaslov.Text = "Marke i modeli";
    }

    private void btnRadniNalozi_Click(object sender, RoutedEventArgs e)
    {
        pgRadniNalozi nalozi = new pgRadniNalozi();
        var content = nalozi.Content;
        nalozi.Content = null;
        grid.Children.Clear();
        grid.Children.Add((UIElement)content);
        nalozi.txtID.Focus();
        tbNaslov.Text = "Radni nalozi";
    }

    private void btnFakture_Click(object sender, RoutedEventArgs e)
    {
        pgFakture fakture = new pgFakture();
        var content = fakture.Content;
        fakture.Content = null;
        grid.Children.Clear();
        grid.Children.Add((UIElement)content);
        fakture.txtID.Focus();
        tbNaslov.Text = "Fakture";
    }

    private void btnDelovi_Click(object sender, RoutedEventArgs e)
    {
        pgRadovi radovi = new pgRadovi();
        var content = radovi.Content;
        radovi.Content = null;
        grid.Children.Clear();
        grid.Children.Add((UIElement)content);
        tbNaslov.Text = "Delovi/Usuge";
    }

    private void btnGarancije_Click(object sender, RoutedEventArgs e)
    {
        pgGarancije garancije = new pgGarancije();

```

```

        var content = garancije.Content;
        garancije.Content = null;
        grid.Children.Clear();
        grid.Children.Add((UIElement)content);
        garancije.cmbFaktura.Focus();
        tbNaslov.Text = "Garancije";
    }

    private void btnVozila_Click(object sender, RoutedEventArgs e)
    {
        pgVozila vozila = new pgVozila();
        var content = vozila.Content;
        vozila.Content = null;
        grid.Children.Clear();
        grid.Children.Add((UIElement)content);
        vozila.txtID.Focus();
        tbNaslov.Text = "Vozila";
    }
}

```

U konstruktoru se čitaju vrednosti iz baze o broju naloga, faktura i vozila i prikazuju u kontrolama tipa **Label** na prozoru.

Klikom na ostala dugmad, deklarirše se i inicijalizuje stranica koja treba da bude prikazana. Čita se njen sadržaj i zatim dodaje na grid predviđen za to. Na otvorenoj stranici se fokusira određena kontrola, a naslov iznad grida dobija vrednost u zavisnosti od toga koja je stranica otvorena. Ukoliko je na gridu već bila prikazana druga stranica, grid se čisti komandom **grid.Children.Clear()** i tek onda se prikazuje sadržaj zatražene stranice.

6.1.6.Stranice (Pages)

Aplikacija raspolaže sa 8 stranica koje koriste neke od modela prema svojoj nameni.

6.1.6.1. pgMarke.xaml

```

public partial class pgMarke : Page
{
    string filter1;
    string filter2;
    public pgMarke()
    {
        InitializeComponent();
        filter1 = "";
        filter2 = "";
        tbPoruka1.Text = "";
        tbPoruka2.Text = "";
        UcitajMarke();
        UcitajListuMarki();
        UcitajListuModela();
    }
}

```

```

private void UcitajMarku()
{
    if (dgPregledMarke.Items.Count <= 0)
    {
        txtIDMarke.Text = "";
        txtNazivMarke.Text = "";
        return;
    }
    DataRowView red = (DataRowView)dgPregledMarke.SelectedItems[0];

    int id = Convert.ToInt32(red[0]);
    Marka marka = Marka.UcitajMarku(id);
    txtIDMarke.Text = marka.Id.ToString();
    txtNazivMarke.Text = marka.NazivMarke.ToString();
}
private void UcitajListuMarki()
{
    tbPoruka1.Text = "";
    dgPregledMarke.ItemsSource = Marka.ListaMarki(filter1).DefaultView;
    UcitajMarku();
}
private void UcitajModel()
{
    if (dgPregledModela.Items.Count <= 0)
    {
        txtIDModel.Text = "";
        txtNazivModela.Text = "";
        cmbMarka.Text = "";
        return;
    }
    DataRowView red = (DataRowView)dgPregledModela.SelectedItems[0];

    int id = Convert.ToInt32(red[0]);
    Model model = Model.UcitajModel(id);
    txtIDModel.Text = model.Id.ToString();
    txtNazivModela.Text = model.NazivModela.ToString();
    cmbMarka.SelectedValue = model.Marka.Id;
}
private void UcitajListuModela()
{
    tbPoruka2.Text = "";
    dgPregledModela.ItemsSource = Model.ListaModela(filter2).DefaultView;
    UcitajModel();
}
private void UcitajMarke()
{
    string sqlUpit = "select MarkaID, NazivMarke from tblMarka;";
    cmbMarka.ItemsSource = AutoServisData.UcitajPodatke(sqlUpit).DefaultView;
}
private void btnSacuvajMarka_Click(object sender, RoutedEventArgs e)
{
    if (String.IsNullOrEmpty(txtNazivMarke.Text))
    {
        tbPoruka1.Text = "Morate uneti naziv marke vozila.";
        return;
    }

    tbPoruka1.Text = "";

    Marka novaMarka = new Marka();

    novaMarka.NazivMarke = txtNazivMarke.Text;
}

```

```

        if (String.IsNullOrEmpty(txtIDMarke.Text) != true)
        {
            Marka staraMarka = Marka.UcitajMarku(Convert.ToInt32(txtIDMarke.Text));
            staraMarka.Azuriraj(novaMarka);
        }
        else
        {
            if (novaMarka.PostojiDuplikat())
            {
                tbPoruka1.Text = "Ova marka već postoji u bazi.  
Ne možete sačuvati duplikat.";
                return;
            }
            novaMarka.Sacuvaj();
        }
        UcitajListuMarki();
    }

    private void btnNovoMarka_Click(object sender, RoutedEventArgs e)
    {
        txtIDMarke.Text = "";
        txtNazivMarke.Text = "";
        txtNazivMarke.Focus();
    }

    private void btnNovoModel_Click(object sender, RoutedEventArgs e)
    {
        txtIDModel.Text = "";
        txtNazivModela.Text = "";
        cmbMarka.SelectedValue = null;
        cmbMarka.Focus();
    }

    private void dgPregledMarke_SelectionChanged(object sender,
        SelectionChangedEventArgs e)
    {
        UcitajMarku();
    }

    private void dgPregledModela_SelectionChanged(object sender,
        SelectionChangedEventArgs e)
    {
        UcitajModel();
    }

    private void btnObrisiMarka_Click(object sender, RoutedEventArgs e)
    {
        if (dgPregledMarke.Items.Count > 0)
        {
            DataRowView red = (DataRowView)dgPregledMarke.SelectedItems[0];
            int id = Convert.ToInt32(red[0]);

            try
            {
                MessageBoxResult rez = MessageBox.Show(@"Da li ste sigurni?",
                    "Upozorenje",
                    MessageBoxButton.YesNo,
                    MessageBoxImage.Question);

                if (rez != MessageBoxResult.Yes)
                {
                    return;
                }
            }
        }
    }

```

```

        Marka marka = Marka.UcitajMarku(id);
        marka.Obrisi();
        UcitajListuMarki();
    }
    catch (InvalidOperationException)
    {
        MessageBox.Show("Niste izabrali red.", "Greška",
            MessageBoxButton.OK, MessageBoxImage.Error);
    }
    catch (Exception ex)
    {
        MessageBox.Show($"Došlo je do greške prilikom pokušaja
            brisanja podataka: { ex.Message }.", "Greška",
            MessageBoxButton.OK, MessageBoxImage.Error);
    }
}

private void btnObrisiModel_Click(object sender, RoutedEventArgs e)
{
    if (dgPregledModela.Items.Count > 0)
    {
        DataRowView red = (DataRowView)dgPregledModela.SelectedItems[0];
        int id = Convert.ToInt32(red[0]);

        try
        {
            MessageBoxResult rez = MessageBox.Show(@"Da li ste sigurni?",
                "Upozorenje",
                MessageBoxButton.YesNo,
                MessageBoxImage.Question);

            if (rez != MessageBoxResult.Yes)
            {
                return;
            }

            Model model = Model.UcitajModel(id);
            model.Obrisi();
            UcitajListuModela();
        }
        catch (InvalidOperationException)
        {
            MessageBox.Show("Niste izabrali red.", "Greška",
                MessageBoxButton.OK, MessageBoxImage.Error);
        }
        catch (Exception ex)
        {
            MessageBox.Show($"Došlo je do greške prilikom pokušaja brisanja
                podataka: { ex.Message }.", "Greška",
                MessageBoxButton.OK, MessageBoxImage.Error);
        }
    }
}

private void btnSacuvajModel_Click(object sender, RoutedEventArgs e)
{
    if (String.IsNullOrEmpty(txtNazivModela.Text))
    {
        tbPoruka2.Text = "Morate uneti naziv modela vozila.";
        return;
    }
    if (cmbMarka.SelectedValue == null)
    {

```

```

        tbPoruka2.Text = "Morate izabrati marku vozila.";
        return;
    }
    tbPoruka2.Text = "";

    Model noviModel = new Model();

    noviModel.NazivModela = txtNazivModela.Text;
    noviModel.Marka = Marka.UcitajMarku(Convert.ToInt32(cmbMarka.SelectedValue));

    if (String.IsNullOrEmpty(txtIDModel.Text) != true)
    {
        Model stariModel = Model.UcitajModel(Convert.ToInt32(txtIDModel.Text));
        stariModel.Azuriraj(noviModel);
    }
    else
    {
        if (noviModel.PostojiDuplikat())
        {
            tbPoruka2.Text = "Ova marka već postoji u bazi.  
Ne možete sačuvati duplikat.";
            return;
        }
        noviModel.Sacuvaj();
    }
    UcitajListuModela();
}

private void Grid_KeyDown(object sender, KeyEventArgs e)
{
    if (e.Key == Key.F3)
    {
        txtPretragaMarka.Text = "";
        txtPretragaMarka.Focus();
    }
    if (e.Key == Key.F4)
    {
        txtPretragaModel.Text = "";
        txtPretragaModel.Focus();
    }
}

private void btnOcistiFilterMarka_Click(object sender, RoutedEventArgs e)
{
    filter1 = "";
    txtPretragaMarka.Text = "Pretraga (F3)";
    UcitajListuMarki();
}

private void btnFiltrirajMarka_Click(object sender, RoutedEventArgs e)
{
    filter1 = txtPretragaMarka.Text;
    UcitajListuMarki();
    txtPretragaMarka.Text = "Pretraga (F3)";
}

private void btnFiltrirajModel_Click(object sender, RoutedEventArgs e)
{
    filter2 = txtPretragaModel.Text;
    UcitajListuModela();
    txtPretragaModel.Text = "Pretraga (F4)";
}

```

```
private void btnOcistiFilterModel_Click(object sender, RoutedEventArgs e)
{
    filter2 = "";
    txtPretragaModel.Text = "Pretraga (F4)";
    UcitajListuModela();
}
}
```

U konstruktoru se učitavaju vrednosti iz tabela u **ComboBox** kontrolu, učitavaju se podaci iz tabela u obe **DataGrid** kontrole, i ukoliko obe **DataGrid** kontrole imaju bar po jedan red, prvi red se učitava u **TextBox** kontrole za prikaz korisniku.

Učitavanje objekata u DataGrid kontrolu se vrši prosleđivanjem argumenta **filter** za svaki **DataGrid** pregled, jer se tako omogućava filtriranje prikaza. Filter se menja unosom željenog pojma u **TextBox** kontrolu namenjenoj za pretragu na koju se fokusira pritiskom dugmeta **F3** ili **F4** na tastaturi. Nakon unosa pojma potrebno je pritisnuti dugme sa desne strane **TextBox** kontrole za pretragu, čime će se inicirati prosleđivanje argumenta **filter** prvo metodi za učitavanje liste u pregled, koja će taj argument proslediti metodi definisanoj u okviru klase nad kojom vršimo prikaz objekata (statička metoda iz modela koja vraća **DataTable** tip podataka). Ukoliko korisnik želi da isključi filter (postavi ga na **prazan string**) i ponovo učitava podatke u **DataGrid** kontrolu potrebno je da pritisne dugme za isključivanje filtera sa leve strane **TextBox** kontrole za pretragu. Time se string filter postavlja na prazan string i poziva metoda za učitavanje podataka u **DataGrid** kontrolu.

Učitavanje objekata u pregled pored DataGrid kontrole se vrši pozivom statičke metode **Ucitaj()** respektivno za svaki objekat svake klase. Dobijene vrednosti se prosleđuju kao stringovi **TextBox** kontrolama (ili **ComboBox** kontrolama).

Ukoliko se na **DataGrid** kontroli selektuje neki drugi objekat (tj. red), podaci će automatski biti učitani u kontrole sa strane.

Klikom na komandno dugme **btnNovo**, očistiće se polja za prikaz podataka i dozvoliće se unos novih podataka za čuvanje u bazu. Fokusiraće se polje koje treba prvo da bude uneto.

Klikom na komandna dugmad **btnSacuvaj** i za marke, i za modele, vrši se provera da li su unete neophodne vrednosti u **TextBox** kontrole (tj. da li su izabrane vrednosti iz padajućih lista ukoliko je to neophodno) i zatim se vrši provera da li je potrebno ažuriranje već postojećeg niza podataka u bazi ili je zahtevan unos novog objekta (tj. njegovih osobina) u bazu. Provera da li je potrebno ažuriranje vrši se tako što se proverava da li u **TextBox** kontroli predviđenoj za prikazivanje **ID** broja postoji unos ili je prazna. Kada se klikne na dugme **btnNovo**, prikaz u toj kontroli će nestati, i to je indikator da korisnik želi da sačuva novi objekat u bazu. Kada se, pak, promeni selektovana vrednost na **DataGrid** kontroli, **ID** broj će se zajedno sa ostalim osobinama objekta učitati u kontrole za prikaz korisniku. Ukoliko je zahtevano

ažuriranje podataka, nad objektom klase se poziva metoda **Azuriraj(obj noviObjekat)**, u suprotnom se proverava da li za objekat koji korisnik želi da sačuva u bazi već postoji duplikat. Ukoliko ne postoji, nad objektom klase se poziva metoda **Sacuvaj()**. Osobine objekta su sačuvane, a pregled u **DataGrid** kontroli se osvežava.

Brisanje se vrši tako što se klikom na komandno dugme **btnBrisi** prikazuje **MessageBox** sa pitanjem upućenim korisniku da li je siguran da želi da izbriše objekat (tj. njegove osobine) iz baze. Ovo pitanje sadrži dodatna upozorenja ako brisanje konkretnog objekta povlači brisanje objekata koji su referencirani na njega, što dolazi do izražaja na stranicama **pgVozila**, **pgRadniNalozi** itd. Kada korisnik potvrdi brisanje, deklariše se objekat kome se vrednost dodeljuje pozivanjem statičke metode **Ucitaj(...)** iz njegove klase, i njoj se prosleđuje argument **ID** broj iz **TextBox** kontrole za prikaz **ID** broja (ili na nekim stranicama iz kolone za prikaz **ID** broja u selektovanom redu na **DataGrid** kontroli). Zatim se nad tim objektom poziva metoda **Obrisi()** iz njegove klase i u **DataGrid** kontrolu se učitava osvežena lista podataka.

Pošto je i na ostalim stranicama princip izvršavanja **CRUD** operacija analogan opisanom, biće objašnjene samo situacije u kojima se neki događaji ili aktivnosti razlikuju od ovde navedenih.

6.1.6.2. pgZaposleni.xaml

```
public partial class pgZaposleni : Page
{
    string filter;
    public pgZaposleni()
    {
        InitializeComponent();
        filter = "";
        UcitajListuZaposlenih();
    }
    private void UcitajZaposlenog()
    {
        if (dgPregled.Items.Count <= 0)
        {
            txtID.Text = "";
            txtIme.Text = "";
            txtPrezime.Text = "";
            txtJMBG.Text = "";
            txtBRLK.Text = "";
            txtGrad.Text = "";
            txtAdresa.Text = "";
            txtKorisnickoIme.Text = "";
            psbLozinka.Password = "";
            tbBrojNaloga.Text = "";
            return;
        }

        DataRowView red = (DataRowView)dgPregled.SelectedItems[0];

        int id = Convert.ToInt32(red[0]);
```

```

Zaposleni zaposleni = Zaposleni.UcitajZaposlenog(id);
txtID.Text = zaposleni.Id.ToString();
txtIme.Text = zaposleni.ImeZaposlenog;
txtPrezime.Text = zaposleni.PrezimeZaposlenog;
txtJMBG.Text = zaposleni.JMBGZaposlenog;
txtBRLK.Text = zaposleni.BrojLKZaposlenog;
txtGrad.Text = zaposleni.GradZaposlenog;
txtAdresa.Text = zaposleni.AdresaZaposlenog;
txtKorisnickoIme.Text = zaposleni.KorisnickoIme;
psbLozinka.Password = zaposleni.Lozinka;
tbBrojNaloga.Text = $"{@"Broj radnih naloga: { AutoServisData.BrojRedovaUBazi(
    "tblRadniNalog", "ZaposleniID", zaposleni.Id.ToString()) }"}";
}
private void UcitajListuZaposlenih()
{
    tbPoruka.Text = "";
    dgPregled.ItemsSource = Zaposleni.ListaZaposlenih(filter).DefaultView;
    UcitajZaposlenog();
}
private void btnSacuvaj_Click(object sender, RoutedEventArgs e)
{
    #region validacija
    if (String.IsNullOrEmpty(txtIme.Text))
    {
        tbPoruka.Text = "Morate uneti ime zaposlenog.";
        return;
    }
    if (String.IsNullOrEmpty(txtPrezime.Text))
    {
        tbPoruka.Text = "Morate uneti prezime zaposlenog.";
        return;
    }
    if (String.IsNullOrEmpty(txtJMBG.Text))
    {
        tbPoruka.Text = "Morate uneti JMBG zaposlenog.";
        return;
    }
    if (String.IsNullOrEmpty(txtBRLK.Text))
    {
        tbPoruka.Text = "Morate uneti broj lične karte zaposlenog.";
        return;
    }
    if (String.IsNullOrEmpty(txtAdresa.Text))
    {
        tbPoruka.Text = "Morate uneti adresu zaposlenog.";
        return;
    }
    if (String.IsNullOrEmpty(txtGrad.Text))
    {
        tbPoruka.Text = "Morate uneti grad zaposlenog.";
        return;
    }
    if (String.IsNullOrEmpty(txtKorisnickoIme.Text))
    {
        tbPoruka.Text = "Morate uneti korisničko ime zaposlenog.";
        return;
    }
    if (String.IsNullOrEmpty(psbLozinka.Password))
    {
        tbPoruka.Text = "Morate uneti lozinku zaposlenog.";
        return;
    }
    #endregion
    tbPoruka.Text = "";
}

```

```

Zaposleni noviZaposleni = new Zaposleni();

try
{
    noviZaposleni.ImeZaposlenog = txtIme.Text;
    noviZaposleni.PrezimeZaposlenog = txtPrezime.Text;
    noviZaposleni.JMBGZaposlenog = txtJMBG.Text;
    noviZaposleni.BrojLKZaposlenog = txtBRLK.Text;
    noviZaposleni.GradZaposlenog = txtGrad.Text;
    noviZaposleni.AdresaZaposlenog = txtAdresa.Text;
    noviZaposleni.KorisnickoIme = txtKorisnickoIme.Text;
    noviZaposleni.Lozinka = psbLozinka.Password;
}
catch (Exception)
{
    tbPoruka.Text = "Niste uneli ispravne vrednosti.";
    return;
}

if (String.IsNullOrEmpty(txtID.Text) != true)
{
    Zaposleni stariZaposleni = Zaposleni.UcitajZaposlenog(
        Convert.ToInt32(txtID.Text));
    stariZaposleni.Azuriraj(noviZaposleni);
}
else
{
    if (noviZaposleni.PostojiDuplikat())
    {
        tbPoruka.Text = "Ovaj zaposleni već postoji u bazi.  
Ne možete sačuvati duplikat.";
        return;
    }
    noviZaposleni.Sacuvaj();
}
UcitajListuZaposlenih();
}

private void btnObrisi_Click(object sender, RoutedEventArgs e)
{
    if (dgPregled.Items.Count > 0)
    {
        DataRowView red = (DataRowView)dgPregled.SelectedItems[0];
        int id = Convert.ToInt32(red[0]);

        try
        {
            MessageBoxResult rez = MessageBox.Show(@"Da li ste sigurni?",
                "Upozorenje",
                MessageBoxButton.YesNo,
                MessageBoxImage.Question);

            if (rez != MessageBoxResult.Yes)
            {
                return;
            }

            Zaposleni zaposleni = Zaposleni.UcitajZaposlenog(id);
            zaposleni.Obrisi();
            UcitajListuZaposlenih();
        }
        catch (InvalidOperationException)
        {
            MessageBox.Show("Niste izabrali red.", "Greška",
                MessageBoxButton.OK, MessageBoxImage.Error);
        }
    }
}

```

```

    }
    catch (Exception ex)
    {
        MessageBox.Show($"Došlo je do greške prilikom pokušaja
                        brisanja podataka: { ex.Message }.", "Greška",
                        MessageBoxButton.OK, MessageBoxImage.Error);
    }
}

private void dgPregled_SelectionChanged(object sender, SelectionChangedEventArgs e)
{
    UcitajZaposlenog();
}

private void btnDodaj_Click(object sender, RoutedEventArgs e)
{
    txtID.Text = "";
    txtIme.Text = "";
    txtPrezime.Text = "";
    txtJMBG.Text = "";
    txtBRLK.Text = "";
    txtGrad.Text = "";
    txtAdresa.Text = "";
    txtKorisnickoIme.Text = "";
    psbLozinka.Password = "";
    txtIme.Focus();
}

private void btnFiltriraj_Click(object sender, RoutedEventArgs e)
{
    filter = txtPretraga.Text;
    UcitajListuZaposlenih();
    txtPretraga.Text = "Pretraga (F3)";
}

private void btnOcistiFilter_Click(object sender, RoutedEventArgs e)
{
    filter = "";
    txtPretraga.Text = "Pretraga (F3)";
    UcitajListuZaposlenih();
}

private void Grid_KeyDown(object sender, KeyEventArgs e)
{
    if (e.Key == Key.F3)
    {
        txtPretraga.Text = "";
        txtPretraga.Focus();
    }
}
}

```

Princip izvršavanja osnovnih operacija u okviru stranice **pgZaposleni.xaml** identičan je ranije opisanom. Nema potrebe za dodatnim objašnjenjima.

6.1.6.3. pgVlasnici.xaml

```
public partial class pgVlasnici : Page
{
    string filter = "";
    public pgVlasnici()
    {
        InitializeComponent();
        filter = "";
        UcitajListuVlasnika();
    }
    private void UcitajVlasnika()
    {
        if (dgPregled.Items.Count <= 0)
        {
            txtID.Text = "";
            txtIme.Text = "";
            txtPrezime.Text = "";
            txtJMBG.Text = "";
            txtBRLK.Text = "";
            txtGrad.Text = "";
            txtAdresa.Text = "";
            txtKontakt.Text = "";
            return;
        }

        DataRowView red = (DataRowView)dgPregled.SelectedItems[0];

        int id = Convert.ToInt32(red[0]);

        Vlasnik vlasnik = Vlasnik.UcitajVlasnika(id);
        txtID.Text = vlasnik.Id.ToString();
        txtIme.Text = vlasnik.ImeVlasnika;
        txtPrezime.Text = vlasnik.PrezimeVlasnika;
        txtJMBG.Text = vlasnik.JMBGVlasnika;
        txtBRLK.Text = vlasnik.BrojLKVlasnika;
        txtGrad.Text = vlasnik.GradVlasnika;
        txtAdresa.Text = vlasnik.AdresaVlasnika;
        txtKontakt.Text = vlasnik.KontaktVlasnika;
    }

    private void UcitajListuVlasnika()
    {
        tbPoruka.Text = "";
        dgPregled.ItemsSource = Vlasnik.ListaVlasnika(filter).DefaultView;
        UcitajVlasnika();
    }
    private void btnSacuvaj_Click(object sender, RoutedEventArgs e)
    {
        if (String.IsNullOrEmpty(txtIme.Text))
        {
            tbPoruka.Text = "Morate napisati ime vlasnika.";
            return;
        }
        if (String.IsNullOrEmpty(txtKontakt.Text))
        {
            tbPoruka.Text = "Morate napisati kontakt vlasnika.";
            return;
        }
        if (String.IsNullOrEmpty(txtGrad.Text))
        {
            tbPoruka.Text = "Morate napisati grad vlasnika.";
            return;
        }
    }
}
```

```

    }
    tbPoruka.Text = "";

    Vlasnik noviVlasnik = new Vlasnik();

    try
    {
        noviVlasnik.ImeVlasnika = txtIme.Text;
        noviVlasnik.PrezimeVlasnika = txtPrezime.Text;
        noviVlasnik.JMBGVlasnika = txtJMBG.Text;
        noviVlasnik.BrojLKVlasnika = txtBRLK.Text;
        noviVlasnik.GradVlasnika = txtGrad.Text;
        noviVlasnik.AdresaVlasnika = txtAdresa.Text;
        noviVlasnik.KontaktVlasnika = txtKontakt.Text;
    }
    catch (Exception)
    {
        tbPoruka.Text = "Niste uneli ispravne vrednosti.";
        return;
    }

    if (String.IsNullOrEmpty(txtID.Text) != true)
    {
        Vlasnik stariVlasnik = Vlasnik.UcitajVlasnika(Convert.ToInt32(txtID.Text));
        stariVlasnik.Azuriraj(noviVlasnik);
    } else
    {
        if (noviVlasnik.PostojiDuplikat())
        {
            tbPoruka.Text = "Ovaj vlasnik već postoji u bazi.  
Ne možete sačuvati duplikat.";
            return;
        }
        noviVlasnik.Sacuvaj();
    }
    UcitajListuVlasnika();
}

private void btnObrisi_Click(object sender, RoutedEventArgs e)
{
    if (dgPregled.Items.Count > 0)
    {
        DataRowView red = (DataRowView)dgPregled.SelectedItems[0];
        int id = Convert.ToInt32(red[0]);

        try
        {
            MessageBoxResult rez = MessageBox.Show(@"Da li ste sigurni? Biće  
obrisani i svi podaci  
povezani sa vlasnikom.",  
"Upozorenje",  
MessageBoxButton.YesNo,  
MessageBoxImage.Question);

            if (rez != MessageBoxResult.Yes)
            {
                return;
            }

            //mora ovako da bismo isli unazad i obrisali sve povezane podatke
            Vlasnik vlasnik = Vlasnik.UcitajVlasnika(id);
            foreach (int idVozila in Vlasnik.ListaVozila(id))
            {
                Vozilo vozilo = Vozilo.UcitajVozilo(idVozila);
                foreach (int idNaloga in Vozilo.ListaNaloga(idVozila))
            }
        }
    }
}

```

```

        {
            RadniNalog nalog = RadniNalog.UcitajNalog(idNaloga);
            Garancija.ObrisiSveGarancijeZaRadniNalog(idNaloga);
            Faktura.ObrisiSveFakture(idNaloga);
            NaruceniRadovi.ObrisiSveNaruceneRadove(idNaloga);
            Delovi.ObrisiSveDelove(idNaloga);
            IzvrzeniRadovi.ObrisiSveIzvrseneRadove(idNaloga);
            nalog.Obrisi();
        }
        vozilo.Obrisi();
    }
    vlasnik.Obrisi();
    UcitajListuVlasnika();
}
catch (InvalidOperationException)
{
    MessageBox.Show("Niste izabrali red.", "Greška",
        MessageBoxButton.OK, MessageBoxImage.Error);
}
catch (Exception ex)
{
    MessageBox.Show($"Došlo je do greške prilikom pokušaja
        brisanja podataka: { ex.Message }.", "Greška",
        MessageBoxButton.OK, MessageBoxImage.Error);
}
}
}

private void dgPregled_SelectionChanged(object sender, SelectionChangedEventArgs e)
{
    UcitajVlasnika();
}

private void btnDodaj_Click(object sender, RoutedEventArgs e)
{
    txtID.Text = "";
    txtIme.Text = "";
    txtPrezime.Text = "";
    txtJMBG.Text = "";
    txtBRLK.Text = "";
    txtGrad.Text = "";
    txtAdresa.Text = "";
    txtKontakt.Text = "";
    txtIme.Focus();
}

private void btnFiltriraj_Click(object sender, RoutedEventArgs e)
{
    filter = txtPretraga.Text;
    UcitajListuVlasnika();
    txtPretraga.Text = "Pretraga (F3)";
}

private void btnOcistiFilter_Click(object sender, RoutedEventArgs e)
{
    filter = "";
    txtPretraga.Text = "Pretraga (F3)";
    UcitajListuVlasnika();
}

private void Grid_KeyDown(object sender, KeyEventArgs e)
{
    if (e.Key == Key.F3)

```

```

    {
        txtPretraga.Text = "";
        txtPretraga.Focus();
    }
}

```

Jedina razlika u načinu izvršavanja operacija na ovoj stranici je što se brisanje izvodi tako što se prolazi kroz listu svih vozila za objekat klase **Vlasnik** koji se briše, pa se pre brisanja osobina samog objekta prvo brišu podaci iz tabela koje su referencirane na njega. Ostale operacije izvršavaju se analogno ranije opisanim.

6.1.6.4. pgVozila.xaml

```

public partial class pgVozila : Page
{
    string filter;
    public pgVozila()
    {
        InitializeComponent();
        filter = "";
        UcitajGoriva();
        UcitajVlasnike();
        UcitajTipove();
        UcitajMarke();
        UcitajListuVozila();
    }

    private void UcitajGoriva()
    {
        string sqlUpit = @"select VrstaGorivaID, VrstaGoriva as 'Gorivo' from tblGorivo;";
        cmbGorivo.ItemsSource = AutoServisData.UcitajPodatke(sqlUpit).DefaultView;
    }
    private void UcitajVlasnike()
    {
        string sqlUpit = @"select VlasnikID,
                                (ImeVlasnika + ' ' + PrezimeVlasnika) as 'Vlasnik'
                                From tblVlasnik;";
        cmbVlasnik.ItemsSource = AutoServisData.UcitajPodatke(sqlUpit).DefaultView;
    }
    private void UcitajTipove()
    {
        string sqlUpit = @"select TipVozilaID, TipVozila from tblTipVozila";
        cmbTipVozila.ItemsSource = AutoServisData.UcitajPodatke(sqlUpit).DefaultView;
    }
    private void UcitajMarke()
    {
        string sqlUpit = @"select ModelID, (NazivMarke + ' ' + NazivModela) as 'Vozilo'
                                From tblModel join tblMarka on tblModel.MarkaID=tblMarka.MarkaID;";
        cmbMarka.ItemsSource = AutoServisData.UcitajPodatke(sqlUpit).DefaultView;
    }
}

```



```

    }
    private void UcitajVozilo()
    {
        if (dgPregled.Items.Count <= 0)
        {
            txtID.Text = "";
            txtSnagaMotora.Text = "";
            txtBrojMotora.Text = "";
            txtGodinaProizvodnje.Text = "";
            txtZapreminaMotora.Text = "";
            txtRegOznaka.Text = "";
            txtBrojSasije.Text = "";
            cmbGorivo.SelectedValue = null;
            cmbVlasnik.SelectedValue = null;
            cmbTipVozila.SelectedValue = null;
            cmbMarka.SelectedValue = null;
            return;
        }
        DataRowView red = (DataRowView)dgPregled.SelectedItems[0];

        int id = Convert.ToInt32(red[0]);
        Vozilo vozilo = Vozilo.UcitajVozilo(id);
        txtID.Text = vozilo.Id.ToString();
        txtSnagaMotora.Text = vozilo.SnagaMotora.ToString();
        txtBrojMotora.Text = vozilo.BrojMotora;
        txtGodinaProizvodnje.Text = vozilo.GodinaProizvodnje.ToString();
        txtZapreminaMotora.Text = vozilo.ZapreminaMotora.ToString();
        txtRegOznaka.Text = vozilo.RegistarskaOznaka;
        txtBrojSasije.Text = vozilo.BrojSasije;
        cmbGorivo.SelectedValue = vozilo.VrstaGoriva.Id;
        cmbVlasnik.SelectedValue = vozilo.Vlasnik.Id;
        cmbTipVozila.SelectedValue = vozilo.TipVozila.Id;
        cmbMarka.SelectedValue = vozilo.Model.Id;
    }
    private void UcitajListuVozila()
    {
        tbPoruka.Text = "";
        dgPregled.ItemsSource = Vozilo.ListaVozila(filter).DefaultView;
        UcitajVozilo();
    }
    private void btnSacuvaj_Click(object sender, RoutedEventArgs e)
    {
        if (cmbGorivo.SelectedValue == null)
        {
            tbPoruka.Text = "Morate izabrati vrstu goriva.";
            return;
        }
        if (cmbVlasnik.SelectedValue == null)
        {
            tbPoruka.Text = "Morate izabrati vlasnika vozila.";
            return;
        }
        if (cmbTipVozila.SelectedValue == null)
        {
            tbPoruka.Text = "Morate izabrati tip vozila.";
            return;
        }
        if (cmbMarka.SelectedValue == null)
        {
            tbPoruka.Text = "Morate izabrati marku i model.";
            return;
        }
    }
    tbPoruka.Text = "";

```

```

Vozilo novoVozilo = new Vozilo();

try
{
    novoVozilo.BrojMotora = txtBrojMotora.Text;

    if (!String.IsNullOrEmpty(txtSnagaMotora.Text))
    {
        novoVozilo.SnagaMotora = Convert.ToInt32(txtSnagaMotora.Text);
    }
    if (!String.IsNullOrEmpty(txtGodinaProizvodnje.Text))
    {
        novoVozilo.GodinaProizvodnje = Convert.ToInt32(txtGodinaProizvodnje.Text);
    }
    if (!String.IsNullOrEmpty(txtZapreminaMotora.Text))
    {
        novoVozilo.ZapreminaMotora = Convert.ToInt32(txtZapreminaMotora.Text);
    }

    novoVozilo.RegistarskaOznaka = txtRegOznaka.Text;
    novoVozilo.BrojSasije = txtBrojSasije.Text;

    novoVozilo.TipVozila = TipVozila.UcitajTipVozila(
        Convert.ToInt32(cmbTipVozila.SelectedValue));
    novoVozilo.VrstaGoriva = Gorivo.UcitajGorivo(
        Convert.ToInt32(cmbGorivo.SelectedValue));
    novoVozilo.Vlasnik = Vlasnik.UcitajVlasnika(
        Convert.ToInt32(cmbVlasnik.SelectedValue));
    novoVozilo.Model = Model.UcitajModel(
        Convert.ToInt32(cmbMarka.SelectedValue));
}
catch (Exception)
{
    tbPoruka.Text = "Niste uneli ispravne vrednosti.";
    return;
}

if (String.IsNullOrEmpty(txtID.Text) != true)
{
    Vozilo staroVozilo = Vozilo.UcitajVozilo(Convert.ToInt32(txtID.Text));
    staroVozilo.Azuriraj(novoVozilo);
} else
{
    if (novoVozilo.PostojiDuplikat())
    {
        tbPoruka.Text = "Ovo vozilo već postoji u bazi.  
Ne možete sačuvati duplikat.";
        return;
    }
    novoVozilo.Sacuvaj();
}
UcitajListuVozila();
}

private void btnObrisi_Click(object sender, RoutedEventArgs e)
{
    if (dgPregled.Items.Count > 0)
    {
        DataRowView red = (DataRowView)dgPregled.SelectedItems[0];
        int id = Convert.ToInt32(red[0]);

        try
        {

```

```

MessageBoxResult rez = MessageBox.Show(@"Da li ste sigurni?
    Biće obrisani i svi podaci povezani sa ovim vozilom.",

    if (rez != MessageBoxResult.Yes)
    {
        return;
    }

    Vozilo vozilo = Vozilo.UcitajVozilo(id);

    foreach (int idNaloga in Vozilo.ListaNaloga(id))
    {

RadniNalog nalog = RadniNalog.UcitajNalog(idNaloga);

Garancija.ObrisiSveGarancijeZaRadniNalog(idNaloga);
        Faktura.ObrisiSveFakture(idNaloga);

NaruceniRadovi.ObrisiSveNaruceneRadove(idNaloga);
        Delovi.ObrisiSveDelove(idNaloga);

IzvrсениRadovi.ObrisiSveIzvršeneRadove(idNaloga);
        nalog.Obrisi();
    }

    vozilo.Obrisi();
    UcitajListuVozila();
}
catch (InvalidOperationException)
{
    MessageBox.Show("Niste izabrali red.", "Greška",
    MessageBoxButton.OK, MessageBoxImage.Error);
}
catch (Exception ex)
{
    MessageBox.Show($"Došlo je do greške prilikom pokušaja brisanja
        podataka: { ex.Message }.", "Greška",
        MessageBoxButton.OK, MessageBoxImage.Error);
}
}

private void dgPregled_SelectionChanged(object sender, SelectionChangedEventArgs e)
{
    UcitajVozilo();
}

private void btnDodaj_Click(object sender, RoutedEventArgs e)
{
    txtID.Text = "";
    txtSnagaMotora.Text = "";
    txtBrojMotora.Text = "";
    txtGodinaProizvodnje.Text = "";
    txtZapreminaMotora.Text = "";
    txtRegOznaka.Text = "";
    txtBrojSasije.Text = "";
    cmbGorivo.SelectedValue = null;
    cmbVlasnik.SelectedValue = null;
}

```

```

        cmbTipVozila.SelectedValue = null;
        cmbMarka.SelectedValue = null;
        txtSnagaMotora.Focus();
    }

private void Grid_KeyDown(object sender, KeyEventArgs e)
{
    if (e.Key == Key.F3)
    {
        txtPretraga.Text = "";
        txtPretraga.Focus();
    }
}

private void btnOcistiFilter_Click(object sender, RoutedEventArgs e)
{
    filter = "";
    txtPretraga.Text = "Pretraga (F3)";
    UcitajListuVozila();
}

private void btnFiltriraj_Click(object sender, RoutedEventArgs e)
{
    filter = txtPretraga.Text;
    UcitajListuVozila();
    txtPretraga.Text = "Pretraga (F3)";
}

private void btnProcitaj_Click(object sender, RoutedEventArgs e)
{
    eVehicleRegistrationCOM.Registration vd =
        new eVehicleRegistrationCOM.Registration();
    try {
        string reader = "";
        int procitano = 0;
        vd.Initialize();
        procitano = vd.GetReaderName(0, out reader);
        string citac = reader;

        if (String.IsNullOrEmpty(citac))
        {
            tbPoruka.Text = "Nema učitanih čitača.";
            return;
        }

        _VEHICLE_DATA vhData = new _VEHICLE_DATA();
        vd.Initialize();
        procitano = vd.SelectReader(citac);
        procitano = vd.ProcessNewCard();
        procitano = vd.ReadVehicleData(out vhData);

        txtSnagaMotora.Text = vhData.maximumNetPower;
        txtBrojMotora.Text = vhData.engineIDNumber;
        txtGodinaProizvodnje.Text = vhData.yearOfProduction;
        txtZapreminaMotora.Text = vhData.engineCapacity;
        txtRegOznaka.Text = vhData.registrationNumberOfVehicle;
        txtBrojSasiije.Text = vhData.vehicleIDNumber;
        try {
            cmbGorivo.Text = vhData.typeOfFuel;
        }
        catch (Exception) { }
        try
        {
            cmbTipVozila.Text = vhData.vehicleType;

```

```

    }
    catch (Exception) { }
}
catch (Exception)
{
    tbPoruka.Text = "Podaci sa saobraćajne dozvole nisu pročitani.
                    Proverite čitač.";
}
finally
{
    vd.Finalize();
}
}
}

```

Razlika u odnosu na prethodno opisane operacije je što u stranici **pgVozilo.xaml** imamo implementiranu funkcionalnost za čitanje podataka sa čipa saobraćajne dozvole i prikaz istih na kontrolama za prikaz vozila. Implementacija je izvršena korišćenjem eksterne biblioteke klasa namenjene za ovu upotrebu, naziva **eVehicleRegistrationCOM.dll**, iz koje je deklarisan objekat klase **Registration** (**eVehicleRegistrationCOM.Registration**) pomoću čijih metoda se pristupa instaliranim čitačima i čitanju podataka sa njihovih čipova. Ova dodatna funkcionalnost je dosta korisna za aplikaciju i deo je ideje autora od samog početka. Dodatne informacije o biblioteci za čitanje podataka sa čipa saobraćajne dozvole mogu se pronaći na sajtu MUP-a Republike Srbije.

Ostale operacije se izvršavaju analogno ranije opisanim na prethodnim stranicama.

6.1.6.5. pgRadniNalozi.xaml

```

public partial class pgRadniNalozi : Page
{
    string filter;
    public pgRadniNalozi()
    {
        InitializeComponent();
        filter = "";
        UcitajListuNaloga();
        UcitajVozila();
        UcitajZaposlene();
    }
    private void UcitajNalog()
    {
        if (dgPregled.Items.Count <= 0)
        {
            txtID.Text = "";
            dtDatumOtvaranja.SelectedDate = null;
            dtDatumZatvaranja.SelectedDate = null;
            cmbVozilo.Text = "";
            cmbZaposleni.Text = "";
            tbTip.Text = "";
            tbVlasnik.Text = "";
            tbGodinaProizvodnje.Text = "";
            tbGorivo.Text = "";
            tbBrojSasije.Text = "";
        }
    }
}

```

```

        tbSnagaMotora.Text = "";
        return;
    }
    DataRowView red = (DataRowView)dgPregled.SelectedItems[0];

    int id = Convert.ToInt32(red[0]);
    RadniNalog nalog = RadniNalog.UcitajNalog(id);
    cmbVozilo.SelectedValue = nalog.Vozilo.Id;
    cmbZaposleni.SelectedValue = nalog.Zaposleni.Id;
    txtID.Text = id.ToString();
    dtDatumOtvaranja.SelectedDate = nalog.DatumOtvaranja;
    dtDatumZatvaranja.SelectedDate = nalog.DatumZatvaranja;

    tbTip.Text = $"Tip vozila:
        { nalog.Vozilo.TipVozila.NazivTipaVozila }";
    tbVlasnik.Text = $"Vlasnik: { nalog.Vozilo.Vlasnik.ImeVlasnika }
        { nalog.Vozilo.Vlasnik.PrezimeVlasnika }";
    tbGodinaProizvodnje.Text = $"Godina proizvodnje:
        { nalog.Vozilo.GodinaProizvodnje.ToString() }";
    tbGorivo.Text = $"Pogonsko gorivo:
        { nalog.Vozilo.VrstaGoriva.VrstaGoriva }";
    tbBrojSasije.Text = $"Broj šasije: { nalog.Vozilo.BrojSasije }";
    tbSnagaMotora.Text = $"Snaga motora: { nalog.Vozilo.SnagaMotora }";
}
private void UcitajVozila()
{
    string sqlUpit = @"select VoziloID,
        (NazivMarke + ' ' + nazivmodela +
        ' - ' + ImeVlasnika + ' ' + PrezimeVlasnika) as Vozilo
        from tblVozilo join tblModel on
        tblVozilo.ModelID=tblModel.ModelID
        join tblMarka on
        tblModel.MarkaID=tblMarka.MarkaID
        join tblVlasnik on
        tblVozilo.VlasnikID=tblVlasnik.VlasnikID;";

    cmbVozilo.ItemsSource =
    AutoServisData.UcitajPodatke(sqlUpit).DefaultView;
}
private void UcitajZaposlene()
{
    string sqlUpit = @"select ZaposleniID,
        (ImeZaposlenog + ' ' + PrezimeZaposlenog)
        as 'Zaposleni' From tblZaposleni";
    cmbZaposleni.ItemsSource =
    AutoServisData.UcitajPodatke(sqlUpit).DefaultView;
}
private void UcitajListuNaloga()
{
    tbPoruka.Text = "";
    dgPregled.ItemsSource = RadniNalog.ListaNaloga(filter).DefaultView;
    UcitajNalog();
}
private void btnObrisi_Click(object sender, RoutedEventArgs e)
{
    if (dgPregled.Items.Count>0)
    {
        DataRowView red = (DataRowView)dgPregled.SelectedItems[0];
        int id = Convert.ToInt32(red[0]);

        try
        {
            MessageBoxResult rez =

```

```

        MessageBox.Show(@"Da li ste sigurni? Biće obrisani
                        i svi podaci povezani na radni nalog.",

                        if (rez != DialogResult.Yes)
                        {
                            return;
                        }

                        RadniNalog nalog = RadniNalog.UcitajNalog(id);
                        Garancija.ObrisiSveGarancijeZaRadniNalog(id);
                        Faktura.ObrisiSveFakture(id);
                        NaruceniRadovi.ObrisiSveNaruceneRadove(id);
                        Delovi.ObrisiSveDelove(id);
                        IzvrсениRadovi.ObrisiSveIzvršeneRadove(id);

                        nalog.Obrisi();
                        UcitajListuNaloga();
                    }
                catch (InvalidOperationException)
                {
                    MessageBox.Show("Niste izabrali red.",
                                    "Greška",
                                    MessageBoxButtons.OK,
                                    MessageBoxIcon.Error);
                }
                catch (Exception ex)
                {
                    MessageBox.Show($"Došlo je do greške prilikom
pokušaja brisanja podataka: { ex.Message }.", "Greška",
                                    MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
            }
        }

private void dgPregled_SelectionChanged(object sender, SelectionChangedEventArgs e)
{
    UcitajNalog();
}

private void cmbVozilo_SelectionChanged(object sender, SelectionChangedEventArgs e)
{
    if (cmbVozilo.SelectedValue != null)
    {
        int voziloId = Convert.ToInt32(cmbVozilo.SelectedValue.ToString());
        Vozilo vozilo = Vozilo.UcitajVozilo(voziloId);
        tbTip.Text = $"Tip vozila:
        { vozilo.TipVozila.NazivTipaVozila }";
        tbVlasnik.Text = $"Vlasnik:
        { vozilo.Vlasnik.ImeVlasnika }
        { vozilo.Vlasnik.PrezimeVlasnika }";
        tbGodinaProizvodnje.Text = $"Godina proizvodnje:
        { vozilo.GodinaProizvodnje.ToString() }";
        tbGorivo.Text = $"Pogonsko gorivo:
        { vozilo.VrstaGoriva.VrstaGoriva }";
        tbBrojSasije.Text = $"Broj šasije: { vozilo.BrojSasije }";
        tbSnagaMotora.Text = $"Snaga motora: { vozilo.SnagaMotora }";
    }
}

```

```

private void btnDodaj_Click(object sender, RoutedEventArgs e)
{
    txtID.Text = "";
    tbPoruka.Text = "";
    dtDatumOtvaranja.SelectedDate = null;
    dtDatumZatvaranja.SelectedDate = null;
    cmbVozilo.Text = "";
    cmbZaposleni.Text = "";
    tbTip.Text = "";
    tbVlasnik.Text = "";
    tbGodinaProizvodnje.Text = "";
    tbGorivo.Text = "";
    tbBrojSasije.Text = "";
    tbSnagaMotora.Text = "";
    dtDatumOtvaranja.Focus();
}

private void btnSacuvaj_Click(object sender, RoutedEventArgs e)
{
    if (dtDatumOtvaranja.SelectedDate == null)
    {
        tbPoruka.Text = "Morate izabrati datum otvaranja radnog naloga.";
        return;
    }

    if (dtDatumZatvaranja.SelectedDate == null)
    {
        tbPoruka.Text = "Morate izabrati datum zatvaranja radnog naloga.";
        return;
    }

    if (cmbVozilo.SelectedValue == null)
    {
        tbPoruka.Text = "Morate izabrati vozilo iz liste.";
        return;
    }

    if (cmbZaposleni.SelectedValue == null)
    {
        tbPoruka.Text = "Morate izabrati zaposlenog.";
        return;
    }

    tbPoruka.Text = "";
    RadniNalog noviNalog = new RadniNalog();
    noviNalog.DatumOtvaranja =
        Convert.ToDateTime(dtDatumOtvaranja.SelectedDate);
    noviNalog.DatumZatvaranja =
        Convert.ToDateTime(dtDatumZatvaranja.SelectedDate);

    noviNalog.Zaposleni =
        Zaposleni.UcitajZaposlenog(Convert.ToInt32(cmbZaposleni.SelectedValue));
    noviNalog.Vozilo =
        Vozilo.UcitajVozilo(Convert.ToInt32(cmbVozilo.SelectedValue));

    if (String.IsNullOrEmpty(txtID.Text) != true)
    {
        RadniNalog stariNalog =
            RadniNalog.UcitajNalog(Convert.ToInt32(txtID.Text));
        stariNalog.Azuriraj(noviNalog);
    } else
    {

```



```

        if (noviNalog.PostojiDuplikat())
        {
            tbPoruka.Text = "Ovaj radni nalog već postoji u bazi.  
Ne možete sačuvati duplikat.";
            return;
        }
        noviNalog.Sacuvaj();
    }
    UcitajListuNaloga();
}

private void btnFiltriraj_Click(object sender, RoutedEventArgs e)
{
    filter = txtPretraga.Text;
    UcitajListuNaloga();
    txtPretraga.Text = "Pretraga (F3)";
}

private void btnOcistiFilter_Click(object sender, RoutedEventArgs e)
{
    filter = "";
    txtPretraga.Text = "Pretraga (F3)";
    UcitajListuNaloga();
}

private void Grid_KeyDown(object sender, KeyEventArgs e)
{
    if (e.Key == Key.F3)
    {
        txtPretraga.Text = "";
        txtPretraga.Focus();
    }
}
}

```

Principi osnovnih operacija i na ovoj stranici ostaju isti, s razlikom što na ovoj stranici prvi put uvodimo pokazivanje podataka iz tabela koje su sa datom tabelom povezane preko stranog ključa. To se koristi i u narednim stranicama, a princip prikaza se sastoji u tome da se osobinama pristupa preko modela (klasa).

Tako npr. da bi se preko objekta klase **RadniNalog** dobio naziv marke vozila iz tog naloga dovoljno je samo ići unazad po klasama:

```

RadniNalog radniNalog = RadniNalog.UcitajNalog(1);

string markaVozila = radniNalog.Vozilo.Model.Marka.NazivMarke;

```

Tabela radni nalog ima strani ključ **VoziloID**, tabela vozilo ima strani ključ **ModelID**, tabela model ima strani ključ **MarkaID**, a tabela marka ima svoju kolonu **NazivMarke**. Međutim, kvalitetnim dizajniranjem modela (klasa) pristupanje je mnogo lakše i jednostavnije, što se vidi i u primeru iznad. Ovo važi za sve modele koje u bazi imaju respektivno povezane tabele.

6.1.6.6. pgFaktura.xaml

```
public partial class pgFaktura : Page
{
    string filter;
    public pgFaktura()
    {
        InitializeComponent();
        filter = "";
        UcitajRadneNaloge();
        UcitajListuFaktura();
    }

    private void UcitajListuFaktura()
    {
        tbPoruka.Text = "";
        dgPregled.ItemsSource = Faktura.ListaFaktura(filter).DefaultView;
        UcitajFakturu();
    }

    private void UcitajRadneNaloge()
    {
        string sqlUpit = @"select tblRadniNalog.RadniNalogID,
(Convert(varchar(10), tblRadniNalog.RadniNalogID) + ';' + tblMarka.NazivMarke + ' '
        + tblModel.NazivModela
        + '-' + tblVlasnik.ImeVlasnika) as 'Nalog'
        From tblRadniNalog join tblVozilo on
        tblRadniNalog.VoziloID=tblVozilo.VoziloID
        join tblVlasnik on
        tblVozilo.VlasnikID=tblVlasnik.VlasnikID
        join tblModel on
        tblVozilo.ModelID=tblModel.ModelID
        join tblMarka on
        tblModel.MarkaID=tblMarka.MarkaID;";

        cmbRadniNalog.ItemsSource = AutoServisData.UcitajPodatke(sqlUpit).DefaultView;
    }

    private void UcitajFakturu()
    {
        if (dgPregled.Items.Count <= 0)
        {
            txtID.Text = "";
            dtDatum.SelectedDate = null;
            dtValuta.SelectedDate = null;
            txtBrojFiskalnogRacuna.Text = "";
            cmbRadniNalog.Text = "";
            tbRadniNalogID.Text = "";
            tbVlasnik.Text = "";
            tbVozilo.Text = "";
            tbIznosDelova.Text = "";
            tbIznosRadova.Text = "";
            tbUkupno.Text = "";
            return;
        }
        DataRowView red = (DataRowView)dgPregled.SelectedItems[0];

        int id = Convert.ToInt32(red[0]);

        Faktura faktura = Faktura.UcitajFakturu(id);
        cmbRadniNalog.SelectedValue = faktura.RadniNalog.Id;
        txtID.Text = id.ToString();
        dtDatum.SelectedDate = faktura.Datum;
        dtValuta.SelectedDate = faktura.Valuta;
        txtBrojFiskalnogRacuna.Text = faktura.BrojFiskalnogRacuna.ToString();
    }
}
```

```

tbRadniNalogID.Text = $"ID radnog naloga: { faktura.RadniNalog.Id }";
tbVlasnik.Text = $"Vlasnik: { faktura.RadniNalog.Vozilo.Vlasnik.ImeVlasnika }
{ faktura.RadniNalog.Vozilo.Vlasnik.PrezimeVlasnika }";
tbVozilo.Text = $"Vozilo: { faktura.RadniNalog.Vozilo.Model.Marka.NazivMarke }
{ faktura.RadniNalog.Vozilo.Model.NazivModela }";
tbIznosDelova.Text = $"Iznos delova: {
    faktura.RadniNalog.IznosDelova().ToString("F2") }";
tbIznosRadova.Text = $"Iznos radova: {
    faktura.RadniNalog.IznosRadova().ToString("F2") }";
tbUkupno.Text = $"Ukupan iznos: { (faktura.RadniNalog.IznosRadova() +
    faktura.RadniNalog.IznosDelova()).ToString("F2") }";
dgRadovi.ItemsSource = IzvrсениRadovi.ListaIzvrсениhRadova(
    faktura.RadniNalog.Id).DefaultView;
dgDelovi.ItemsSource = Delovi.ListaDelova(
    faktura.RadniNalog.Id).DefaultView;
}
private void btnObrisi_Click(object sender, RoutedEventArgs e)
{
    if (dgPregled.Items.Count > 0)
    {
        DataRowView red = (DataRowView)dgPregled.SelectedItems[0];
        int id = Convert.ToInt32(red[0]);

        try
        {
            MessageBoxResult rez = MessageBox.Show(@"Da li ste sigurni?
                Biće obrisani i sve garancije povezane na fakturu.",
                "Upozorenje",
                MessageBoxButton.YesNo,
                MessageBoxImage.Question);

            if (rez != MessageBoxResult.Yes)
            {
                return;
            }

            Faktura faktura = Faktura.UcitajFakturu(id);
            Garancija.ObrisiSveGarancije(id);

            faktura.Obrisi();
            UcitajListuFaktura();
        }
        catch (InvalidOperationException)
        {
            MessageBox.Show("Niste izabrali red.", "Greška",
                MessageBoxButton.OK, MessageBoxImage.Error);
        }
        catch (Exception ex)
        {
            MessageBox.Show($"Došlo je do greške prilikom pokušaja brisanja
                podataka: { ex.Message }.", "Greška",
                MessageBoxButton.OK, MessageBoxImage.Error);
        }
    }
}

private void dgPregled_SelectionChanged(object sender, SelectionChangedEventArgs e)
{
    UcitajFakturu();
}
private void cmbRadniNalog_SelectionChanged(object sender, SelectionChangedEventArgs e)
{
    if (cmbRadniNalog.SelectedValue != null)

```

```

    {
        RadniNalog nalog = RadniNalog.UcitajNalog(Convert.ToInt32(
            cmbRadniNalog.SelectedValue));
        tbRadniNalogID.Text = $"ID radnog naloga: { nalog.Id }";
        tbVlasnik.Text = $"Vlasnik: { nalog.Vozilo.Vlasnik.ImeVlasnika }
            { nalog.Vozilo.Vlasnik.PrezimeVlasnika }";
        tbVozilo.Text = $"Vozilo: { nalog.Vozilo.Model.Marka.NazivMarke }
            { nalog.Vozilo.Model.NazivModela }";

        tbIznosDelova.Text = $"Iznos delova: { nalog.IznosDelova().ToString("F2") }";
        tbIznosRadova.Text = $"Iznos radova: { nalog.IznosRadova().ToString("F2") }";
        tbUkupno.Text = $"Ukupan iznos: { (nalog.IznosRadova() +
            nalog.IznosDelova()).ToString("F2") }";

        dgRadovi.ItemsSource = IzvrсениRadovi.ListaIzvrсениhRadova(
            nalog.Id).DefaultView;
        dgDelovi.ItemsSource = Delovi.ListaDelova(nalog.Id).DefaultView;
    }
}

private void btnDodaj_Click(object sender, RoutedEventArgs e)
{
    txtID.Text = "";
    dtDatum.SelectedDate = null;
    dtValuta.SelectedDate = null;
    txtBrojFiskalnogRacuna.Text = "";
    cmbRadniNalog.Text = "";
    tbRadniNalogID.Text = "";
    tbVlasnik.Text = "";
    tbVozilo.Text = "";
    tbIznosDelova.Text = "";
    tbIznosRadova.Text = "";
    tbUkupno.Text = "";
    dtDatum.Focus();
}

private void Grid_KeyDown(object sender, KeyEventArgs e)
{
    if (e.Key == Key.F3)
    {
        txtPretraga.Text = "";
        txtPretraga.Focus();
    }
}

private void btnSacuvaj_Click(object sender, RoutedEventArgs e)
{
    if (dtDatum.SelectedDate == null)
    {
        tbPoruka.Text = "Morate izabrati datum.";
        return;
    }
    if (dtValuta.SelectedDate == null)
    {
        tbPoruka.Text = "Morate izabrati valutu.";
        return;
    }
    if (String.IsNullOrEmpty(txtBrojFiskalnogRacuna.Text))
    {
        tbPoruka.Text = "Morate uneti broj fiskalnog računa.";
        return;
    }
    if (cmbRadniNalog.SelectedValue == null)
    {

```

```

        tbPoruka.Text = "Morate izabrati radni nalog.";
        return;
    }

    tbPoruka.Text = "";

    Faktura novaFaktura = new Faktura();

    try
    {
        novaFaktura.Datum = Convert.ToDateTime(dtDatum.SelectedDate);
        novaFaktura.Valuta = Convert.ToDateTime(dtValuta.SelectedDate);
        novaFaktura.BrojFiskalnogRacuna = Convert.ToInt32(txtBrojFiskalnogRacuna.Text);
        novaFaktura.RadniNalog = RadniNalog.UcitajNalog(Convert.ToInt32(
            cmbRadniNalog.SelectedValue));
    }
    catch (Exception)
    {
        tbPoruka.Text = "Niste uneli ispravne vrednosti.";
        return;
    }

    if ((novaFaktura.RadniNalog.BrojDelova() + novaFaktura.RadniNalog.BrojRadova()) <= 0)
    {
        tbPoruka.Text = "Ne možete sačuvati fakturu kojoj je iznos radova i delova 0.";
        return;
    }

    if (String.IsNullOrEmpty(txtID.Text) != true)
    {
        Faktura staraFaktura = Faktura.UcitajFakturu(Convert.ToInt32(txtID.Text));
        staraFaktura.Azuriraj(novaFaktura);
    } else
    {
        if (novaFaktura.PostojiDuplikat())
        {
            tbPoruka.Text = "Ova faktura već postoji u bazi.  
Ne možete sačuvati duplikat.";
            return;
        }
        novaFaktura.Sacuvaj();
    }
    UcitajListuFaktura();
}

private void btnFiltriraj_Click(object sender, RoutedEventArgs e)
{
    filter = txtPretraga.Text;
    UcitajListuFaktura();
    txtPretraga.Text = "Pretraga (F3)";
}

private void btnOcistiFilter_Click(object sender, RoutedEventArgs e)
{
    filter = "";
    txtPretraga.Text = "Pretraga (F3)";
    UcitajListuFaktura();
}
}

```

Principi izvršavanja osnovnih operacija i na ovoj stranici prate konzistentnost. Nisu potrebna dodatna objašnjenja.

6.1.6.7. pgRadovi.xaml

```
public partial class pgRadovi : Page
{
    int naruceniRedni;
    int izvrseniRedni;
    int deloviRedni;

    public pgRadovi()
    {
        InitializeComponent();
        cmbRadniNalog.SelectedIndex = 0;
        UcitajRadneNaloge();
        UcitajRadniNalog();
        UcitajListuNarucenih();
        UcitajListuIzvršenih();
        UcitajListuDelova();
    }
    private void UcitajRadneNaloge()
    {
        string sqlUpit = @"select tblRadniNalog.RadniNalogID, (Convert(varchar(10),
            tblRadniNalog.RadniNalogID) + ';' + tblMarka.NazivMarke + ' '
            + tblModel.NazivModela + ';' + tblVlasnik.ImeVlasnika + ' ' +
            tblVlasnik.PrezimeVlasnika) as 'Nalog'
            From tblRadniNalog join tblVozilo on
                tblRadniNalog.VoziloID=tblVozilo.VoziloID
                join tblVlasnik on
                    tblVozilo.VlasnikID=tblVlasnik.VlasnikID
                join tblModel on
                    tblVozilo.ModelID=tblModel.ModelID
                join tblMarka on
                    tblModel.MarkaID=tblMarka.MarkaID;";

        cmbRadniNalog.ItemsSource = AutoServisData.UcitajPodatke(sqlUpit).DefaultView;
    }
    private void UcitajRadniNalog()
    {
        if (cmbRadniNalog.SelectedValue != null)
        {
            RadniNalog nalog = RadniNalog.UcitajNalog(
                Convert.ToInt32(cmbRadniNalog.SelectedValue));
            tbRadniNalog.Text = $"{nalog.Id } { nalog.Vozilo:
                { nalog.Vozilo.Model.Marka.NazivMarke } { nalog.Vozilo.Model.NazivModela }"
                + $"{nalog.Vlasnik:
                { nalog.Vozilo.Vlasnik.ImeVlasnika } { nalog.Vozilo.Vlasnik.PrezimeVlasnika }";
            tbIzvršeni.Text = $"{nalog.IznosRadova().ToString("F2") }";
            tbDelovi.Text = $"{nalog.IznosDelova().ToString("F2") }";
        }
    }
    private void cmbRadniNalog_SelectionChanged(object sender, SelectionChangedEventArgs e)
    {
        if (cmbRadniNalog.SelectedValue != null)
        {
            UcitajListuNarucenih();
            UcitajListuIzvršenih();
            UcitajListuDelova();
        }
    }
}
```

```

#region naruceni
private void UcitajNaruceni()
{
    if (dgNaruceni.Items.Count <= 0 || cmbRadniNalog.SelectedValue == null)
    {
        txtOpisNarucenih.Text = "";
        return;
    }

    int id = Convert.ToInt32(cmbRadniNalog.SelectedValue);
    DataRowView red = (DataRowView)dgNaruceni.SelectedItems[0];
    int rbr = Convert.ToInt32(red[0]);
    NaruceniRadovi naruceni = NaruceniRadovi.UcitajNaruceneRadove(rbr, id);
    naruceniRedni = rbr;
    txtOpisNarucenih.Text = naruceni.Opis;
}
private void UcitajListuNarucenih()
{
    tbPoruka1.Text = "";
    if (cmbRadniNalog.SelectedValue == null)
    {
        return;
    }
    int id = Convert.ToInt32(cmbRadniNalog.SelectedValue);
    dgNaruceni.ItemsSource = NaruceniRadovi.ListaNarucenihRadova(id).DefaultView;
    UcitajNaruceni();
}
private void btnSacuvajNaruceni_Click(object sender, RoutedEventArgs e)
{
    if (cmbRadniNalog.SelectedValue == null)
    {
        tbPoruka1.Text = "Morate izabrati radni nalog.";
        return;
    }

    if (String.IsNullOrEmpty(txtOpisNarucenih.Text))
    {
        tbPoruka1.Text = "Morate uneti opis.";
        return;
    }

    NaruceniRadovi naruceni = new NaruceniRadovi();
    naruceni.RedniBroj = naruceniRedni;
    naruceni.Opis = txtOpisNarucenih.Text;
    naruceni.RadniNalog = RadniNalog.UcitajNalog(
        Convert.ToInt32(cmbRadniNalog.SelectedValue));

    bool azuriraj = false;

    foreach (DataRowView red in dgNaruceni.ItemsSource)
    {
        if (Convert.ToInt32(red[0]) == naruceni.RedniBroj)
        {
            azuriraj = true;
            break;
        }
    }

    if (azuriraj == true)
    {
        NaruceniRadovi stari = NaruceniRadovi.UcitajNaruceneRadove(naruceniRedni,
            Convert.ToInt32(cmbRadniNalog.SelectedValue));
        stari.Azuriraj(naruceni);
    }
}

```

```

        UcitajListuNarucenih();
        return;
    }

    naruceni.Sacuvaj();
    UcitajListuNarucenih();
}
private void dgNaruceni_SelectionChanged(object sender, SelectionChangedEventArgs e)
{
    UcitajNaruceni();
}
private void btnNovoNaruceni_Click(object sender, RoutedEventArgs e)
{
    naruceniRedni = dgNaruceni.Items.Count + 1;
    txtOpisNarucenih.Text = "";
    txtOpisNarucenih.Focus();
}
private void btnObrisiNaruceni_Click(object sender, RoutedEventArgs e)
{
    if (dgNaruceni.Items.Count > 0 && cmbRadniNalog.SelectedValue != null)
    {
        DataRowView red = (DataRowView)dgNaruceni.SelectedItems[0];
        int rbr = Convert.ToInt32(red[0]);

        try
        {
            MessageBoxResult rez = MessageBox.Show(@"Da li ste sigurni?",
                                                    "Upozorenje",
                                                    MessageBoxButton.YesNo,
                                                    MessageBoxImage.Question);

            if (rez != MessageBoxResult.Yes)
            {
                return;
            }

            NaruceniRadovi rad = NaruceniRadovi.UcitajNaruceneRadove(rbr,
                                                                    Convert.ToInt32(cmbRadniNalog.SelectedValue));
            rad.Obrisi();

            UcitajListuNarucenih();
        }
        catch (InvalidOperationException)
        {
            MessageBox.Show("Niste izabrali red.", "Greška",
                            MessageBoxButton.OK, MessageBoxImage.Error);
        }
        catch (Exception ex)
        {
            MessageBox.Show($"Došlo je do greške prilikom pokušaja
                            brisanja podataka: { ex.Message }.", "Greška",
                            MessageBoxButton.OK, MessageBoxImage.Error);
        }
    }
}
#endregion

#region izvrseni

private void UcitajIzvrzeni()
{
    if (dgIzvrzeni.Items.Count <= 0 || cmbRadniNalog.SelectedValue == null)
    {
        txtNazivIzvrzenih.Text = "";
    }
}

```



```

        txtKolicina.Text = "";
        txtCena.Text = "";
        txtJmr.Text = "";
        izvrseniRedni = 1;
        return;
    }

    int id = Convert.ToInt32(cmbRadniNalog.SelectedValue);
    DataRowView red = (DataRowView)dgIzvrzeni.SelectedItems[0];
    int rbr = Convert.ToInt32(red[0]);

    IzvrzeniRadovi izvrseni = IzvrzeniRadovi.UcitajIzvrzeneRadove(rbr, id);
    izvrseniRedni = rbr;
    txtNazivIzvrzenih.Text = izvrseni.Naziv;
    txtKolicina.Text = izvrseni.Kolicina.ToString("F3");
    txtJmr.Text = izvrseni.JedinicaMere;
    txtCena.Text = izvrseni.Cena.ToString("F2");
}
private void UcitajListuIzvrzenih()
{
    tbPoruka2.Text = "";
    if (cmbRadniNalog.SelectedValue == null)
    {
        return;
    }
    int id = Convert.ToInt32(cmbRadniNalog.SelectedValue);
    dgIzvrzeni.ItemsSource = IzvrzeniRadovi.ListaIzvrzenihRadova(id).DefaultView;
    UcitajIzvrzeni();
    UcitajRadniNalog();
}
private void btnSacuvajIzvrzeni_Click(object sender, RoutedEventArgs e)
{
    if (cmbRadniNalog.SelectedValue == null)
    {
        tbPoruka2.Text = "Morate izabrati radni nalog.";
        return;
    }

    if (String.IsNullOrEmpty(txtNazivIzvrzenih.Text))
    {
        tbPoruka2.Text = "Morate uneti naziv usluge.";
        return;
    }

    if (String.IsNullOrEmpty(txtKolicina.Text))
    {
        tbPoruka2.Text = "Morate uneti količinu.";
        return;
    }

    if (String.IsNullOrEmpty(txtJmr.Text))
    {
        tbPoruka2.Text = "Morate uneti jedinicu mere.";
        return;
    }

    if (String.IsNullOrEmpty(txtCena.Text))
    {
        tbPoruka2.Text = "Morate uneti cenu.";
        return;
    }

    IzvrzeniRadovi izvrseni = new IzvrzeniRadovi();

```

```

try {

    izvrseni.RedniBroj = izvrseniRedni;
    izvrseni.Naziv = txtNazivIzvrsekih.Text;
    izvrseni.Kolicina = Convert.ToDouble(txtKolicina.Text);
    izvrseni.Cena = Convert.ToDouble(txtCena.Text);
    izvrseni.JedinicaMere = txtJmr.Text;
    izvrseni.RadniNalog = RadniNalog.UcitajNalog(Convert.ToInt32(
                                                                    cmbRadniNalog.SelectedValue));
}
catch (Exception)
{
    tbPoruka2.Text = "Niste uneli ispravne vrednosti.";
    return;
}
bool azuriraj = false;

foreach (DataRowView red in dgIzvrsekih.ItemsSource)
{
    if (Convert.ToInt32(red[0]) == izvrseni.RedniBroj)
    {
        azuriraj = true;
        break;
    }
}

if (azuriraj == true)
{
    IzvrsekihRadovi stari = IzvrsekihRadovi.UcitajIzvrsekihRadove(izvrsekihRedni,
                                                                    Convert.ToInt32(cmbRadniNalog.SelectedValue));
    stari.Azuriraj(izvrsekih);
    UcitajListuIzvrsekih();
    return;
}

izvrsekih.Sacuvaj();
UcitajListuIzvrsekih();
}
private void dgIzvrsekih_SelectionChanged(object sender, SelectionChangedEventArgs e)
{
    UcitajIzvrsekih();
}
private void btnNovoIzvrsekih_Click(object sender, RoutedEventArgs e)
{
    izvrseniRedni = dgIzvrsekih.Items.Count + 1;
    txtNazivIzvrsekih.Text = "";
    txtKolicina.Text = "";
    txtCena.Text = "";
    txtJmr.Text = "";
    txtNazivIzvrsekih.Focus();
}
private void btnObrisiIzvrsekih_Click(object sender, RoutedEventArgs e)
{
    if (dgIzvrsekih.Items.Count > 0 && cmbRadniNalog.SelectedValue != null)
    {
        DataRowView red = (DataRowView)dgIzvrsekih.SelectedItems[0];
        int rbr = Convert.ToInt32(red[0]);

        try
        {
            MessageBoxResult rez = MessageBox.Show(@"Da li ste sigurni?",
                                                    "Upozorenje",
                                                    MessageBoxButton.YesNo,

```

```

        MessageBoxImage.Question);

        if (rez != DialogResult.Yes)
        {
            return;
        }

        IzvrсениRadovi rad = IzvrсениRadovi.UcitajIzvrсeneRadove(rbr,
            Convert.ToInt32(cmbRadniNalog.SelectedValue));
        rad.Obrisi();

        UcitajListuIzvrсenih();
    }
    catch (InvalidOperationException)
    {
        MessageBox.Show("Niste izabrali red.", "Greška",
            MessageBoxButtons.OK, MessageBoxImage.Error);
    }
    catch (Exception ex)
    {
        MessageBox.Show($"Došlo je do greške prilikom pokušaja brisanja
            podataka: { ex.Message }.", "Greška",
            MessageBoxButtons.OK, MessageBoxImage.Error);
    }
}

}

#endregion

#region delovi

private void UcitajDeo()
{
    if (dgDelovi.Items.Count <= 0 || cmbRadniNalog.SelectedValue == null)
    {
        txtNazivDela.Text = "";
        txtSifra.Text = "";
        txtKolicinaDelovi.Text = "";
        txtCenaDelovi.Text = "";
        txtJmrDelovi.Text = "";
        deloviRedni = 1;
        return;
    }

    int id = Convert.ToInt32(cmbRadniNalog.SelectedValue);
    DataRowView red = (DataRowView)dgDelovi.SelectedItems[0];
    int rbr = Convert.ToInt32(red[0]);

    Delovi deo = Delovi.UcitajDeo(rbr, id);
    deloviRedni = rbr;
    txtNazivDela.Text = deo.Naziv;
    txtSifra.Text = deo.Sifra;
    txtKolicinaDelovi.Text = deo.Kolicina.ToString("F3");
    txtCenaDelovi.Text = deo.Cena.ToString("F2");
    txtJmrDelovi.Text = deo.JedinicaMere;
}

private void UcitajListuDelova()
{
    tbPoruka3.Text = "";
    if (cmbRadniNalog.SelectedValue == null)
    {
        return;
    }

    int id = Convert.ToInt32(cmbRadniNalog.SelectedValue);

```

```

        dgDelovi.ItemsSource = Delovi.ListaDelova(id).DefaultView;
        UcitajDeo();
        UcitajRadniNalog();
    }
    private void btnSacuvajDelovi_Click(object sender, RoutedEventArgs e)
    {
        if (cmbRadniNalog.SelectedValue == null)
        {
            tbPoruka3.Text = "Morate izabrati radni nalog.";
            return;
        }
        if (String.IsNullOrEmpty(txtSifra.Text))
        {
            tbPoruka3.Text = "Morate uneti šifru.";
            return;
        }
        if (String.IsNullOrEmpty(txtNazivDela.Text))
        {
            tbPoruka3.Text = "Morate uneti naziv dela.";
            return;
        }
        if (String.IsNullOrEmpty(txtKolicinaDelovi.Text))
        {
            tbPoruka3.Text = "Morate uneti količinu.";
            return;
        }

        if (String.IsNullOrEmpty(txtJmrDelovi.Text))
        {
            tbPoruka3.Text = "Morate uneti jedinicu mere.";
            return;
        }

        if (String.IsNullOrEmpty(txtCenaDelovi.Text))
        {
            tbPoruka3.Text = "Morate uneti cenu.";
            return;
        }

        Delovi deo = new Delovi();

        try {

            deo.RedniBroj = deloviRedni;
            deo.Sifra = txtSifra.Text;
            deo.Naziv = txtNazivDela.Text;
            deo.Kolicina = Convert.ToDouble(txtKolicinaDelovi.Text);
            deo.Cena = Convert.ToDouble(txtCenaDelovi.Text);
            deo.JedinicaMere = txtJmrDelovi.Text;
            deo.RadniNalog = RadniNalog.UcitajNalog(Convert.ToInt32(cmbRadniNalog.SelectedValue));

        }
        catch (Exception)
        {
            tbPoruka3.Text = "Niste uneli ispravne vrednosti.";
            return;
        }

        bool azuriraj = false;

        foreach (DataRowView red in dgDelovi.ItemsSource)
        {
            if (Convert.ToInt32(red[0]) == deo.RedniBroj)
            {

```

```

        azuriraj = true;
        break;
    }
}

if (azuriraj == true)
{
    Delovi stari = Delovi.UcitajDeo(deloviRedni,
        Convert.ToInt32(cmbRadniNalog.SelectedValue));
    stari.Azuriraj(deo);
    UcitajListuDelova();
    return;
}

deo.Sacuvaj();
UcitajListuDelova();
}
private void dgDelovi_SelectionChanged(object sender, SelectionChangedEventArgs e)
{
    UcitajDeo();
}
private void btnNovoDelovi_Click(object sender, RoutedEventArgs e)
{
    deloviRedni = dgDelovi.Items.Count + 1;
    txtNazivDela.Text = "";
    txtSifra.Text = "";
    txtKolicinaDelovi.Text = "";
    txtCenaDelovi.Text = "";
    txtJmrDelovi.Text = "";
    txtSifra.Focus();
}
private void btnObrisiDelovi_Click(object sender, RoutedEventArgs e)
{
    if (dgDelovi.Items.Count > 0 && cmbRadniNalog.SelectedValue != null)
    {
        DataRowView red = (DataRowView)dgDelovi.SelectedItems[0];
        int rbr = Convert.ToInt32(red[0]);

        try
        {
            MessageBoxResult rez = MessageBox.Show(@"Da li ste sigurni?",
                "Upozorenje",
                MessageBoxButton.YesNo,
                MessageBoxImage.Question);

            if (rez != MessageBoxResult.Yes)
            {
                return;
            }

            Delovi deo = Delovi.UcitajDeo(rbr, Convert.ToInt32(
                cmbRadniNalog.SelectedValue));
            deo.Obrisi();
            UcitajListuDelova();
        }
        catch (InvalidOperationException)
        {
            MessageBox.Show("Niste izabrali red.", "Greška",
                MessageBoxButton.OK, MessageBoxImage.Error);
        }
        catch (Exception ex)
        {
            MessageBox.Show($"Došlo je do greške prilikom pokušaja brisanja
                podataka: { ex.Message }.", "Greška",
                MessageBoxButton.OK, MessageBoxImage.Error);
        }
    }
}

```

```

    }

    }

#endregion

}

```

Principi izvršavanja osnovnih operacija ostaju isti, iako se na ovoj stranici nalaze tri osnovna modela (**NaruceniRadovi**, **IzvrсениRadovi** i **Delovi**). Nisu potrebna dodatna objašnjenja.

6.1.6.8. pgGarancije.xaml

```

public partial class pgGarancije : Page
{
    string filter;
    public pgGarancije()
    {
        InitializeComponent();
        filter = "";
        cmbFaktura.SelectedIndex = 0;
        UcitajFakturu();
        UcitajListuGarancija();
    }

    private void UcitajFakturu()
    {
        int id = Convert.ToInt32(cmbFaktura.SelectedValue);
        Faktura faktura = Faktura.UcitajFakturu(id);
        tbFaktura.Text = $"{ID fakture: { faktura.Id }, Datum: { faktura.Datum.ToShortDateString() }, Valuta: { faktura.Datum.ToShortDateString() }}";
        tbRadniNalogID.Text = $"{ID radnog naloga: { faktura.RadniNalog.Id }}";
        tbVlasnik.Text = $"{Vlasnik: { faktura.RadniNalog.Vozilo.Vlasnik.ImeVlasnika } { faktura.RadniNalog.Vozilo.Vlasnik.PrezimeVlasnika } { faktura.RadniNalog.Vozilo.Model.NazivModela }}";
        tbIznosFakture.Text = $"{Iznos fakture: { (faktura.RadniNalog.IznosDelova() + faktura.RadniNalog.IznosRadova()).ToString("F2") }}";
    }
    private void UcitajFakturu()
    {
        string sqlUpit = @"select tblFaktura.FakturaID,
        (Convert(varchar(10), tblFaktura.FakturaID) +
        ' - ' +
        tblVlasnik.ImeVlasnik
        + ' ' +
        tblVlasnik.PrezimeVlasnika +
        ' ; ' + tblMarka.NazivMarke + ' ' +
        tblModel.NazivModela)
        as 'Faktura'
        from tblFaktura join tblRadniNalog on
        tblFaktura.RadniNalogID=tblRadniNalog.RadniNalogID

        join tblVozilo on

```

```

        tblRadniNalog.VoziloID=tblVozilo.VoziloID

    join tblVlasnik on
        tblVozilo.VlasnikID=tblVlasnik.VlasnikID

    join tblModel on
        tblVozilo.ModelID=tblModel.ModelID

    join tblMarka on
        tblModel.MarkaID=tblMarka.MarkaID;";

cmbFaktura.ItemsSource = AutoServisData.UcitajPodatke(sqlUpit).DefaultView;
}
private void UcitajGaranciju()
{
    if (dgPregled.Items.Count <= 0 || cmbFaktura.SelectedValue == null)
    {
        txtID.Text = "";
        txtOpis.Text = "";
        txtRokVazenja.Text = "";
        return;
    }

    DataRowView red = (DataRowView)dgPregled.SelectedItems[0];
    int garancijaID = Convert.ToInt32(red[0]);
    Garancija garancija = Garancija.UcitajGaranciju(garancijaID);
    txtID.Text = garancija.Id.ToString();
    txtOpis.Text = garancija.Opis.ToString();
    txtRokVazenja.Text = garancija.RokVazenja.ToString();
}
private void UcitajListuGarancija()
{
    tbPoruka.Text = "";
    if (cmbFaktura.SelectedValue == null)
    {
        return;
    }
    int id = Convert.ToInt32(cmbFaktura.SelectedValue);

    dgPregled.ItemsSource = Garancija.ListaGarancija(id,filter).DefaultView;
    UcitajGaranciju();
}
private void btnObrisi_Click(object sender, RoutedEventArgs e)
{
    if (dgPregled.Items.Count > 0 && cmbFaktura.SelectedValue != null)
    {
        DataRowView red = (DataRowView)dgPregled.SelectedItems[0];
        int rbr = Convert.ToInt32(red[0]);

        try
        {
            MessageBoxResult rez = MessageBox.Show(@"Da li ste sigurni?",

                if (rez != MessageBoxResult.Yes)
                {
                    return;
                }

            Garancija garancija = Garancija.UcitajGaranciju(Convert.ToInt32(
                onvert.ToInt32(txtID.Text)));

```

```

        garancija.Obrisi();

        UcitajListuGarancija();
    }
    catch (InvalidOperationException)
    {
        MessageBox.Show("Niste izabrali red.", "Greška",
            MessageBoxButton.OK, MessageBoxImage.Error);
    }
    catch (Exception ex)
    {
        MessageBox.Show($"Došlo je do greške prilikom pokušaja brisanja podataka:
            { ex.Message }.", "Greška",
            MessageBoxButton.OK, MessageBoxImage.Error);
    }
}

private void btnSacuvaj_Click(object sender, RoutedEventArgs e)
{
    if (cmbFaktura.SelectedValue == null)
    {
        tbPoruka.Text = "Morate izabrati fakturu.";
        return;
    }
    if (String.IsNullOrEmpty(txtOpis.Text))
    {
        tbPoruka.Text = "Morate uneti opis.";
        return;
    }
    if (String.IsNullOrEmpty(txtRokVazenja.Text))
    {
        tbPoruka.Text = "Morate uneti rok važenja garancije.";
        return;
    }
    tbPoruka.Text = "";
    Garancija novaGarancija = new Garancija();
    try {
        novaGarancija.Opis = txtOpis.Text;

        novaGarancija.RokVazenja = Convert.ToInt32(txtRokVazenja.Text);
        novaGarancija.Faktura = Faktura.UcitajFakturu(
            Convert.ToInt32(cmbFaktura.SelectedValue));
    }
    catch (Exception)
    {
        tbPoruka.Text = "Niste uneli ispravne vrednosti.";
        return;
    }

    if (String.IsNullOrEmpty(txtID.Text) != true)
    {
        Garancija staraGarancija = Garancija.UcitajGaranciju(
            Convert.ToInt32(txtID.Text));
        staraGarancija.Azuriraj(novaGarancija);
    } else
    {
        if (novaGarancija.PostojiDuplikat())
        {
            tbPoruka.Text = "Ova garancija već postoji u bazi.
                Ne možete sačuvati duplikat.";
        }
    }
}

```



```

        return;
    }
    novaGarancija.Sacuvaj();
}
UcitajListuGarancija();

}

private void dgPregled_SelectionChanged(object sender, SelectionChangedEventArgs e)
{
    UcitajGaranciju();
}

private void cmbFaktura_SelectionChanged(object sender, SelectionChangedEventArgs e)
{
    UcitajListuGarancija();
    UcitajFakturu();
}

private void btnFiltriraj_Click(object sender, RoutedEventArgs e)
{
    filter = txtPretraga.Text;
    UcitajListuGarancija();
    txtPretraga.Text = "Pretraga (F3)";
}

private void btnOcistiFilter_Click(object sender, RoutedEventArgs e)
{
    filter = "";
    txtPretraga.Text = "Pretraga (F3)";
    UcitajListuGarancija();
}

private void Grid_KeyDown(object sender, KeyEventArgs e)
{
    if (e.Key == Key.F3)
    {
        txtPretraga.Text = "";
        txtPretraga.Focus();
    }
}

private void btnDodaj_Click(object sender, RoutedEventArgs e)
{
    filter = "";
    UcitajListuGarancija();
    txtID.Text = "";
    txtOpis.Text = "";
    txtRokVazenja.Text = "";
    txtOpis.Focus();
}
}

```

Principi izvršavanja osnovnih operacija ostaju isti. Nisu potrebna dodatna objašnjenja.

6.2. Dizajn korisničkog interfejsa

Prvobitna ideja da korisnički dizajn bude spoj modernog, intuitivnog, i inženjerski prihvatljivog nije napustila misao autora, međutim ovo poslednje je moralo da izostane - naravno zbog nedostatka vremena. **XAML** kod je pisan hard-core, bez previše definisanja stilova, već su atributi dodavani fizički, na licu mesta, ne bi li se postigao optimalan izgled i lično zadovoljstvo izazvano usaglašenim širinama i dužinama pojedinih kontrola. Sve ovo rezultiralo je činjenicom da bi aplikacija morala da pretrpi znatne dizajnerske promene da bi profesionalni dizajner mogao da je prihvati kao više od malog studentskog projekta. Ipak, ako se to ostavi po strani, prosečnom korisniku će se više nego dopasti izgled prozora, i osećaj koji se dobija prolaskom kroz aplikaciju.

Jedini stilovi napisani u **App.xaml** fajlu se tiču **TextBox** i **PasswordBox** kontrola. Pri fokusu one menjaju **Background** boju u žutu. Zbog svega napisanog biće prikazan **XAML** kod fajla **App.xaml**, dok će od ostalih prozora i stranica biti prikazan fizički izgled, pa na čitaocu ostaje da proceni finalni rezultat.

6.2.1.App.xaml

```
<Application x:Class="AutoServis.App"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="clr-namespace:AutoServis"
    StartupUri="Forms\frmLogin.xaml">
    <Application.Resources>
        <Style TargetType="TextBox">
            <Setter Property="Template">
                <Setter.Value>
                    <ControlTemplate TargetType="{x:Type TextBox}">
                        <Border x:Name="border" BorderBrush="{TemplateBinding BorderBrush}" BorderThickness="{TemplateBinding BorderThickness}" Background="{TemplateBinding Background}" SnapsToDevicePixels="True">
                            <ScrollViewer x:Name="PART_ContentHost" Focusable="false" HorizontalScrollBarVisibility="Hidden" VerticalScrollBarVisibility="Hidden"/>
                        </Border>
                        <ControlTemplate.Triggers>
                            <Trigger Property="IsEnabled" Value="false">
                                <Setter Property="Opacity" TargetName="border" Value="0.3"/>
                            </Trigger>
                            <Trigger Property="IsMouseOver" Value="true">
                                <Setter Property="BorderBrush" TargetName="border" Value="#FF7EB4EA"/>
                            </Trigger>
                            <Trigger Property="IsFocused" Value="true">
                                <Setter Property="BorderBrush" TargetName="border" Value="black" />
                                <Setter Property="BorderThickness" TargetName="border" Value="1" />
                                <Setter Property="Background" TargetName="border" Value="yellow" />
                            </Trigger>
                        </ControlTemplate.Triggers>
                    </ControlTemplate>
                </Setter.Value>
            </Setter>
        </Style>
    </Application.Resources>
</Application>
```


```





        </ControlTemplate.Triggers>
    </ControlTemplate>
    </Setter.Value>
</Setter>
</Style>
<Style TargetType="PasswordBox">
    <Setter Property="Template">
        <Setter.Value>
            <ControlTemplate TargetType="{x:Type PasswordBox}">
                <Border x:Name="border" BorderBrush="{TemplateBinding BorderBrush}" BorderThickness="{TemplateBinding BorderThickness}" Background="{TemplateBinding Background}" SnapsToDevicePixels="True">
                    <ScrollViewer x:Name="PART_ContentHost" Focusable="false" HorizontalScrollBarVisibility="Hidden" VerticalScrollBarVisibility="Hidden"/>
                </Border>
                <ControlTemplate.Triggers>
                    <Trigger Property="IsEnabled" Value="false">
                        <Setter Property="Opacity" TargetName="border" Value="0.3"/>
                    </Trigger>
                    <Trigger Property="IsMouseOver" Value="true">
                        <Setter Property="BorderBrush" TargetName="border" Value="#FF7EB4EA"/>
                    </Trigger>
                    <Trigger Property="IsFocused" Value="true">
                        <Setter Property="BorderBrush" TargetName="border" Value="black"/>
                        <Setter Property="BorderThickness" TargetName="border" Value="1"/>
                        <Setter Property="Background" TargetName="border" Value="yellow"/>
                    </Trigger>
                </ControlTemplate.Triggers>
            </ControlTemplate>
        </Setter.Value>
    </Setter>
</Style>
</Application.Resources>
</Application>

```

6.2.2.frmLogin.xaml

Log In - Auto servis





Verzija 1.0.0.1

6.2.3.frmPocetna.xaml

Vozila

Radni nalozi

Delovi/Usluge

Fakture

Garancije

Vlasnici

Zaposleni

Marke

Odjava





Datum: 15/01/2020
Broj vozila u bazi: 2
Broj radnih naloga u bazi: 1
Broj faktura u bazi: 0

6.2.4.pgMarke.xaml

Vozila

Radni nalozi

Delovi/Usluge

Fakture

Garancije

Marke i modeli

Vlasnici

Zaposleni

Marke

Odjava

Marke

ID marke	Naziv marke
1	Alfa Romeo
2	Audi
3	BMW
4	Bugatti
5	Chevrolet
6	Chrysler
7	Citroen
8	Daewoo
9	Daihatsu

ID marke

1

Naziv marke

Alfa Romeo

Sačuvaj Nova marka Obriši

Pretraga (F3)

Modeli

ID	Marka	Model
3	Alfa Romeo	164
4	Alfa Romeo	Spider
5	Peugeot	1007
6	Peugeot	106
7	Peugeot	107
8	Peugeot	108
9	BMW	128i
10	BMW	135i
11	Chevrolet	1500

ID marke

4

Marka

Alfa Romeo

Naziv modela

Spider

Sačuvaj Novi model Obriši

Pretraga (F4)

6.2.5.pgZaposleni.xaml

Vozila

Radni nalozi

Delovi/Usluge

Fakture

Garancije

Zaposleni

Vlasnici

Zaposleni

Marke

Odjava

ID	Ime	Prezime	JMBG	Adresa	Grad	Korisničko ime	Broj LK
8	Jovan	Milosevic	2403999790074	Kolo srpskih sestara	Novi Sad	admin	123456789
9	Petar	Petrovic	1234567891012	Bulevar Oslobođenja 15	Novi Sad	pero.petrovic	32165478
10	Marko	Markovic	2101987654321	Vojvode Stepe 3	Beograd	marko.markovic	789456132

ID zaposlenog

8

Ime zaposlenog

Jovan

Prezime zaposlenog

Milosevic

JMBG zaposlenog

2403999790074

Broj lične karte

123456789

Grad zaposlenog

Novi Sad

Adresa stanovanja

Kolo srpskih sestara

Korisničko ime

admin

Lozinka

•••••

Broj radnih naloga: 0

Obriši

Pretraga (F3)

Sačuvaj Novi zaposleni

6.2.6.pgVlasnici.xaml

		Vozila	Radni nalozi	Delovi/Usluge	Fakture	Garancije	Vlasnici																																
<div>Vlasnici</div> <div>Zaposleni</div> <div>Marke</div>	<table border="1"> <thead> <tr> <th>ID</th> <th>Ime</th> <th>Prezime</th> <th>Kontakt</th> <th>JMBG</th> <th>Adresa</th> <th>Grad</th> <th>Broj LK</th> </tr> </thead> <tbody> <tr> <td>8</td> <td>Nikola</td> <td>Nikolic</td> <td>-</td> <td>0105987790044</td> <td>Bulevar Kralja Aleksandra 43</td> <td>Beograd</td> <td>018584</td> </tr> <tr> <td>9</td> <td>Djordje</td> <td>Djokovic</td> <td>0645985623</td> <td>2107965790042</td> <td>Bulevar Cara Lazara 40</td> <td>Novi Sad</td> <td>149357</td> </tr> <tr> <td colspan="8"></td> </tr> </tbody> </table>						ID	Ime	Prezime	Kontakt	JMBG	Adresa	Grad	Broj LK	8	Nikola	Nikolic	-	0105987790044	Bulevar Kralja Aleksandra 43	Beograd	018584	9	Djordje	Djokovic	0645985623	2107965790042	Bulevar Cara Lazara 40	Novi Sad	149357									<div>ID vlasnika</div> <div>8</div> <div>Ime vlasnika</div> <div>Nikola</div> <div>Prezime vlasnika</div> <div>Nikolic</div> <div>JMBG vlasnika</div> <div>0105987790044</div> <div>Broj lične karte</div> <div>018584</div> <div>Grad vlasnika</div> <div>Beograd</div> <div>Adresa stanovanja</div> <div>Bulevar Kralja Aleksandra 43</div> <div>Kontakt</div> <div>-</div>
	ID	Ime	Prezime	Kontakt	JMBG	Adresa	Grad	Broj LK																															
	8	Nikola	Nikolic	-	0105987790044	Bulevar Kralja Aleksandra 43	Beograd	018584																															
	9	Djordje	Djokovic	0645985623	2107965790042	Bulevar Cara Lazara 40	Novi Sad	149357																															
Odjava	Obriši	<div>✗</div> <div>Pretraga (F3)</div> <div>🔍</div>		Sačuvaj	Novi vlasnik																																		

6.2.7.pgVozila.xaml

		Vozila	Radni nalozi	Delovi/Usluge	Fakture	Garancije	Vozila																																
<div>Vlasnici</div> <div>Zaposleni</div> <div>Marke</div>	<table border="1"> <thead> <tr> <th>ID</th> <th>Vozilo</th> <th>Vlasnik</th> <th>Pogonsko gorivo</th> <th>Tip vozila</th> <th>Registarska oznaka</th> <th>Snaga motora</th> <th>Broj motora</th> </tr> </thead> <tbody> <tr> <td>9</td> <td>Peugeot 206 CC</td> <td>Nikola Nikolic</td> <td>Benzin</td> <td>Putnicko vozilo</td> <td>NS423RD</td> <td>180</td> <td>0156748ad1e</td> </tr> <tr> <td>10</td> <td>Audi A3</td> <td>Djordje Djokovic</td> <td>TNG</td> <td>Putnicko vozilo</td> <td>BG321DA</td> <td>130</td> <td>1515</td> </tr> <tr> <td colspan="8"></td> </tr> </tbody> </table>						ID	Vozilo	Vlasnik	Pogonsko gorivo	Tip vozila	Registarska oznaka	Snaga motora	Broj motora	9	Peugeot 206 CC	Nikola Nikolic	Benzin	Putnicko vozilo	NS423RD	180	0156748ad1e	10	Audi A3	Djordje Djokovic	TNG	Putnicko vozilo	BG321DA	130	1515									<div>ID vozila</div> <div>9</div> <div>Snaga motora</div> <div>180</div> <div>Broj motora</div> <div>0156748ad1e</div> <div>Godina proizvodnje</div> <div>2011</div> <div>Zapremina motora</div> <div>1987</div> <div>Registarska oznaka</div> <div>NS423RD</div> <div>Broj šasije</div> <div>VWVSFFAF151652</div> <div>Vrsta goriva</div> <div>Benzin</div> <div>Vlasnik</div> <div>Nikola Nikolic</div> <div>Tip vozila</div> <div>Putnicko vozilo</div> <div>Marka i model</div> <div>Peugeot 206 CC</div>
	ID	Vozilo	Vlasnik	Pogonsko gorivo	Tip vozila	Registarska oznaka	Snaga motora	Broj motora																															
	9	Peugeot 206 CC	Nikola Nikolic	Benzin	Putnicko vozilo	NS423RD	180	0156748ad1e																															
	10	Audi A3	Djordje Djokovic	TNG	Putnicko vozilo	BG321DA	130	1515																															
Odjava	Obriši	Pročitaj saobraćajnu	<div>✗</div> <div>Pretraga (F3)</div> <div>🔍</div>		Sačuvaj	Novo vozilo																																	

6.2.8.pgRadniNalozi.xaml

	Vozila	Radni nalozi	Delovi/Usluge	Fakture	Garancije	Radni nalozi																		
Vlasnici	<table border="1"> <thead> <tr> <th>ID</th> <th>Datum otvaranja</th> <th>Datum zatvaranja</th> <th>Vozilo</th> <th>Vlasnik</th> <th>Zaposleni</th> </tr> </thead> <tbody> <tr> <td>41</td> <td>15/01/2020</td> <td>15/01/2020</td> <td>Peugeot 206 CC</td> <td>Nikola Nikolic</td> <td>Jovan Milosevic</td> </tr> <tr> <td>42</td> <td>15/01/2020</td> <td>15/01/2020</td> <td>Audi A3</td> <td>Djordje Djokovic</td> <td>Petar Petrovic</td> </tr> </tbody> </table>					ID	Datum otvaranja	Datum zatvaranja	Vozilo	Vlasnik	Zaposleni	41	15/01/2020	15/01/2020	Peugeot 206 CC	Nikola Nikolic	Jovan Milosevic	42	15/01/2020	15/01/2020	Audi A3	Djordje Djokovic	Petar Petrovic	ID naloga <input type="text" value="41"/>
ID	Datum otvaranja	Datum zatvaranja	Vozilo	Vlasnik	Zaposleni																			
41	15/01/2020	15/01/2020	Peugeot 206 CC	Nikola Nikolic	Jovan Milosevic																			
42	15/01/2020	15/01/2020	Audi A3	Djordje Djokovic	Petar Petrovic																			
Zaposleni						Datum otvaranja <input type="text" value="15/01/2020"/>																		
Marke						Datum zatvaranja <input type="text" value="15/01/2020"/>																		
						Zaposleni <input type="text" value="Jovan Milosevic"/>																		
						Vozilo <input type="text" value="Peugeot 206 CC - Nikola Nikoli"/>																		
						Tip vozila: Putnicko vozilo Vlasnik: Nikola Nikolic Godina proizvodnje: 2011 Pogonsko gorivo: Benzin Broj šasije: VVVSFFAF151652 Snaga motora: 180																		
Odjava	<input type="button" value="Obriši"/> <input type="button" value="Pretraga (F3)"/> <input type="button" value="Sačuvaj"/> <input type="button" value="Novi nalog"/>																							

6.2.9.pgRadovi.xaml

	Vozila	Radni nalozi	Delovi/Usluge	Fakture	Garancije	Delovi/Usluge																																																																		
Vlasnici	41; Peugeot 206 CC - ID naloga: 41, Vozilo: Peugeot 206 CC, Vlasnik: Nikola Nikolic																																																																							
Zaposleni	Naručeni radovi Opis <input type="text" value="Mali servis"/>		Izvršeni radovi: 1540,00 Naziv izvršene usluge <input type="text" value="Zamena filtera ulja"/>		Ugrađeni delovi: 8580,00 Šifra Naziv <input type="text" value="D1A32"/> <input type="text" value="Filter ulja"/>																																																																			
Marke	Količina <input type="text" value="0,300"/> JMR SAT Cena <input type="text" value="1200,00"/>		Količina <input type="text" value="0,300"/> JMR SAT Cena <input type="text" value="1200,00"/>		Količina <input type="text" value="1,000"/> JMR KOM Cena <input type="text" value="850,00"/>																																																																			
	<table border="1"> <thead> <tr> <th>Rbr</th> <th>Opis radova</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>Mali servis</td> </tr> <tr> <td>2</td> <td>Provera akumulatora</td> </tr> </tbody> </table>		Rbr	Opis radova	1	Mali servis	2	Provera akumulatora	<table border="1"> <thead> <tr> <th>Rbr</th> <th>Naziv</th> <th>Jmr</th> <th>Količina</th> <th>Cena</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>Zamena filtera ulja</td> <td>SAT</td> <td>0.300</td> <td>1200.00</td> </tr> <tr> <td>2</td> <td>Zamena filtera goriva</td> <td>SAT</td> <td>0.200</td> <td>1200.00</td> </tr> <tr> <td>3</td> <td>Zamena filtera vazduha</td> <td>SAT</td> <td>0.200</td> <td>1200.00</td> </tr> <tr> <td>4</td> <td>Zamena ulja</td> <td>SAT</td> <td>0.500</td> <td>1200.00</td> </tr> <tr> <td>5</td> <td>Zamena akumulatora</td> <td>SAT</td> <td>0.100</td> <td>1000.00</td> </tr> </tbody> </table>		Rbr	Naziv	Jmr	Količina	Cena	1	Zamena filtera ulja	SAT	0.300	1200.00	2	Zamena filtera goriva	SAT	0.200	1200.00	3	Zamena filtera vazduha	SAT	0.200	1200.00	4	Zamena ulja	SAT	0.500	1200.00	5	Zamena akumulatora	SAT	0.100	1000.00	<table border="1"> <thead> <tr> <th>Rbr</th> <th>Šifra</th> <th>Naziv</th> <th>Jmr</th> <th>Količina</th> <th>Cena</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>D1A32</td> <td>Filter ulja</td> <td>KOM</td> <td>1.000</td> <td>850.00</td> </tr> <tr> <td>2</td> <td>OIU51</td> <td>Filter goriva</td> <td>KOM</td> <td>1.000</td> <td>600.00</td> </tr> <tr> <td>3</td> <td>45E6F</td> <td>Filter vazduha</td> <td>KOM</td> <td>1.000</td> <td>550.00</td> </tr> <tr> <td>4</td> <td>AKM23</td> <td>Akumulator</td> <td>KOM</td> <td>1.000</td> <td>6580.00</td> </tr> </tbody> </table>		Rbr	Šifra	Naziv	Jmr	Količina	Cena	1	D1A32	Filter ulja	KOM	1.000	850.00	2	OIU51	Filter goriva	KOM	1.000	600.00	3	45E6F	Filter vazduha	KOM	1.000	550.00	4	AKM23	Akumulator	KOM	1.000	6580.00
Rbr	Opis radova																																																																							
1	Mali servis																																																																							
2	Provera akumulatora																																																																							
Rbr	Naziv	Jmr	Količina	Cena																																																																				
1	Zamena filtera ulja	SAT	0.300	1200.00																																																																				
2	Zamena filtera goriva	SAT	0.200	1200.00																																																																				
3	Zamena filtera vazduha	SAT	0.200	1200.00																																																																				
4	Zamena ulja	SAT	0.500	1200.00																																																																				
5	Zamena akumulatora	SAT	0.100	1000.00																																																																				
Rbr	Šifra	Naziv	Jmr	Količina	Cena																																																																			
1	D1A32	Filter ulja	KOM	1.000	850.00																																																																			
2	OIU51	Filter goriva	KOM	1.000	600.00																																																																			
3	45E6F	Filter vazduha	KOM	1.000	550.00																																																																			
4	AKM23	Akumulator	KOM	1.000	6580.00																																																																			
Odjava	<input type="button" value="Sačuvaj"/> <input type="button" value="Novi"/> <input type="button" value="Obriši"/>		<input type="button" value="Sačuvaj"/> <input type="button" value="Novi"/> <input type="button" value="Obriši"/>		<input type="button" value="Sačuvaj"/> <input type="button" value="Novi"/> <input type="button" value="Obriši"/>																																																																			

6.2.10. pgFakture.xaml

		Vozila	Radni nalozi	Delovi/Usluge	Fakture	Garancije	Fakture																																																											
Vlasnici Zaposleni Marke	<table border="1"> <thead> <tr> <th>ID</th> <th>Datum</th> <th>Valuta</th> <th>Fiskalni račun</th> <th>Radni nalog ID</th> <th>Vozilo</th> <th>Vlasnik</th> </tr> </thead> <tbody> <tr> <td>10</td> <td>15/01/2020</td> <td>15/01/2020</td> <td>1354</td> <td>41</td> <td>Peugeot 206 CC</td> <td>Nikola Nikolic</td> </tr> <tr> <td>11</td> <td>15/01/2020</td> <td>15/01/2020</td> <td>9874</td> <td>42</td> <td>Audi A3</td> <td>Djordje Djokovic</td> </tr> </tbody> </table>						ID	Datum	Valuta	Fiskalni račun	Radni nalog ID	Vozilo	Vlasnik	10	15/01/2020	15/01/2020	1354	41	Peugeot 206 CC	Nikola Nikolic	11	15/01/2020	15/01/2020	9874	42	Audi A3	Djordje Djokovic	ID fakture <input type="text" value="10"/>																																						
	ID	Datum	Valuta	Fiskalni račun	Radni nalog ID	Vozilo	Vlasnik																																																											
	10	15/01/2020	15/01/2020	1354	41	Peugeot 206 CC	Nikola Nikolic																																																											
11	15/01/2020	15/01/2020	9874	42	Audi A3	Djordje Djokovic																																																												
						Datum <input type="text" value="15/01/2020"/>																																																												
						Valuta <input type="text" value="15/01/2020"/>																																																												
						Broj fiskalnog računa <input type="text" value="1354"/>																																																												
						Radni nalog <input type="text" value="41; Peugeot 206 CC-Nikola"/>																																																												
<div> <div>Izvršeni radovi</div> <table border="1"> <thead> <tr> <th>Rbr</th> <th>Naziv</th> <th>Jmr</th> <th>Količina</th> <th>Cena</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>Zamena filtera ulja</td> <td>SAT</td> <td>0.300</td> <td>1200.00</td> </tr> <tr> <td>2</td> <td>Zamena filtera goriva</td> <td>SAT</td> <td>0.200</td> <td>1200.00</td> </tr> <tr> <td>3</td> <td>Zamena filtera vazduha</td> <td>SAT</td> <td>0.200</td> <td>1200.00</td> </tr> <tr> <td>4</td> <td>Zamena ulja</td> <td>SAT</td> <td>0.500</td> <td>1200.00</td> </tr> <tr> <td>5</td> <td>Zamena akumulatora</td> <td>SAT</td> <td>0.100</td> <td>1000.00</td> </tr> </tbody> </table> </div> <div> <div>Ugrađeni delovi</div> <table border="1"> <thead> <tr> <th>Rbr</th> <th>Šifra</th> <th>Naziv</th> <th>Jmr</th> <th>Količina</th> <th>Cena</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>D1A32</td> <td>Filter ulja</td> <td>KOM</td> <td>1.000</td> <td>850.00</td> </tr> <tr> <td>2</td> <td>OIU51</td> <td>Filter goriva</td> <td>KOM</td> <td>1.000</td> <td>600.00</td> </tr> <tr> <td>3</td> <td>45E6F</td> <td>Filter vazduha</td> <td>KOM</td> <td>1.000</td> <td>550.00</td> </tr> <tr> <td>4</td> <td>AKM23</td> <td>Akumulator</td> <td>KOM</td> <td>1.000</td> <td>6580.00</td> </tr> </tbody> </table> </div>						Rbr	Naziv	Jmr	Količina	Cena	1	Zamena filtera ulja	SAT	0.300	1200.00	2	Zamena filtera goriva	SAT	0.200	1200.00	3	Zamena filtera vazduha	SAT	0.200	1200.00	4	Zamena ulja	SAT	0.500	1200.00	5	Zamena akumulatora	SAT	0.100	1000.00	Rbr	Šifra	Naziv	Jmr	Količina	Cena	1	D1A32	Filter ulja	KOM	1.000	850.00	2	OIU51	Filter goriva	KOM	1.000	600.00	3	45E6F	Filter vazduha	KOM	1.000	550.00	4	AKM23	Akumulator	KOM	1.000	6580.00	ID radnog naloga: 41 Vlasnik: Nikola Nikolic Vozilo: Peugeot 206 CC Iznos radova: 1540,00 Iznos delova: 8580,00 Ukupan iznos: 10120,00
Rbr	Naziv	Jmr	Količina	Cena																																																														
1	Zamena filtera ulja	SAT	0.300	1200.00																																																														
2	Zamena filtera goriva	SAT	0.200	1200.00																																																														
3	Zamena filtera vazduha	SAT	0.200	1200.00																																																														
4	Zamena ulja	SAT	0.500	1200.00																																																														
5	Zamena akumulatora	SAT	0.100	1000.00																																																														
Rbr	Šifra	Naziv	Jmr	Količina	Cena																																																													
1	D1A32	Filter ulja	KOM	1.000	850.00																																																													
2	OIU51	Filter goriva	KOM	1.000	600.00																																																													
3	45E6F	Filter vazduha	KOM	1.000	550.00																																																													
4	AKM23	Akumulator	KOM	1.000	6580.00																																																													
Odjava	Obriši		Pretraga (F3)		Sačuvaj Nova faktura																																																													

6.2.11. pgGarancije.xaml

		Vozila	Radni nalozi	Delovi/Usluge	Fakture	Garancije	Garancije															
Vlasnici Zaposleni Marke	<div> <input type="text" value="10 - Nikola Nikolic; Peugeot 206"/> <input type="text" value="ID fakture: 10, Datum: 15/01/2020, Valuta: 15/01/2020"/> </div>						ID garancije <input type="text" value="17"/>															
	<table border="1"> <thead> <tr> <th>ID garancije</th> <th>ID fakture</th> <th>Kupac</th> <th>Opis</th> <th>Važi do datuma:</th> </tr> </thead> <tbody> <tr> <td>17</td> <td>10</td> <td>Nikola Nikolic</td> <td>Garancija za akumulator marke Bosch 75AH</td> <td>15/01/2022</td> </tr> <tr> <td>18</td> <td>10</td> <td>Nikola Nikolic</td> <td>Garancija na izvršene usluge</td> <td>15/07/2020</td> </tr> </tbody> </table>						ID garancije	ID fakture	Kupac	Opis	Važi do datuma:	17	10	Nikola Nikolic	Garancija za akumulator marke Bosch 75AH	15/01/2022	18	10	Nikola Nikolic	Garancija na izvršene usluge	15/07/2020	Opis <input type="text" value="Garancija za akumulator marke Bosch 75AH"/>
	ID garancije	ID fakture	Kupac	Opis	Važi do datuma:																	
17	10	Nikola Nikolic	Garancija za akumulator marke Bosch 75AH	15/01/2022																		
18	10	Nikola Nikolic	Garancija na izvršene usluge	15/07/2020																		
						Rok važenja <input type="text" value="24"/>																
						ID radnog naloga: 41 Vlasnik: Nikola Nikolic Vozilo: Peugeot 206 CC Iznos fakture: 10120,00																
Odjava	Obriši		Pretraga (F3)		Sačuvaj Nova garancija																	

7. Testiranje aplikacije

Testiranje funkcionalnosti aplikacije je sprovedeno u više navrata pokušavajući da se stvori situacija da aplikacija reaguje neočekivano. Sve metode su grupisane u **Try-Catch** blokove, da bi se sprečilo bilo kakvo neželjeno dejstvo. Očekivani problemi mogu se pojaviti ukoliko **Microsoft SQL Server** procesi nisu pokrenuti pravilno, ili dođe do problema u vezi između aplikacije i baze. Što se tiče validacije, za sva **NOT NULL** polja iz tabela baze podataka koja se ne unesu u aplikaciji, prilikom klika na dugme **btnSacuvaj** će prijaviti grešku, dok validacija u smislu provere dužine unetih podataka nije implementirana. Prilikom **CRUD** operacija su u najvećoj meri korišćeni **parametrizovani** upiti (osim u par metoda, npr. za filtriranje se koristi obična **konkatenacija stringova**), pa se tu, očekivano, mogu javiti problemi (prilikom unosa **apostrofa** u polje za pretragu itd.). Kako ovo nije profesionalno testirana aplikacija (iako je i za to postojala iskrena želja), više od ovoga nije moguće zaključiti.

Jednostavna analiza koda:

Code Metrics Results						
Filter: None						
Hierarchy	Maintainability Index	Cyclomatic Complexity	Depth of Inheritance	Class Coupling	Lines of Code	
AutoServis2020 (Debug)	70	733	9	79	2.506	
AutoServis2020.Data	60	25	1	10	89	
AutoServis2020.Forms	63	301	9	69	1.153	
AutoServis2020.Models	71	398	1	24	1.248	
AutoServis2020	84	5	9	14	16	
AutoServis2020.Interfaces	100	4	0	0	0	

Jednostavnom analizom koda primećuje se da se pojedini segmenti mogu znatno optimizovati, međutim autor smatra da je napravljen prilično dobar kompromis između raspoloživog vremena, optimalnosti i kvaliteta koda i aplikacije celokupno.

8. Zaključak

Pored svih želja da se razvije moderna aplikacija u kojoj će grafički interfejs biti potpuno fizički izolovan od **back-end** dela i modela, i u kojoj će dominirati **MVVM** dizajnerski obrazac sa bogatim **XAML** stilovima i **DataBinding**-om na svim kontrolama, kreirana je aplikacija koja je na pola puta između **WindowsForms** projekta napisanog u **WPF** okruženju i zaista ozbiljne aplikacije koja može nastati iz ovoga. Sa druge strane, korisnički interfejs, iako pisan usput, zaista ostavlja jake impresije na prosečne korisnike, dok klase koje reprezentuju modele i metode u njima imaju moć da u svakom trenutku u minimalno linija koda **sačuvaju**, **ažuriraju**, **obrišu** ili **prikažu** objekat iz baze, tjs. njegove osobine definisane u samom modelu.

Krajnji rezultat zapravo jeste mala, ali jako sposobna aplikacija, koja može da zadovolji deo potreba prosečnog auto servisa kojima je namenjena. Uz sagledavanje potencijala ovakve aplikacije, činjenica je da se dodatnim ulaganjem, bez problema, može stvoriti aplikacija koja će zadovoljiti sve potrebe i probleme iznete u opisu poslovanja realnog sistema.