Izrada poslovnih aplikacija koristeći programski jezik C#

Miličević, Tea

Master's thesis / Diplomski rad

2020

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, University of Zagreb, Faculty of Humanities and Social Sciences / Sveučilište u Zagrebu, Filozofski fakultet**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:131:469678

SVEUČILIŠTE U ZAGREBU
FILOZOFSKI FAKULTET
ODSJEK ZA INFORMACIJSKE I KOMUNIKACIJSKE ZNANOSTI
SMJER ISTRAŽIVAČKA INFORMATIKA
Ak. god. 2019./2020.

Tea Miličević

Izrada poslovnih aplikacija koristeći programski jezik C#

Diplomski rad

Mentor: dr. sc. Ivan Dunđer

Zagreb, rujan 2020.

Izjava o akademskoj čestitosti

Izjavljujem i svojim potpisom potvrđujem da je ovaj rad rezultat mog vlastitog rada koji se temelji na istraživanjima te objavljenoj i citiranoj literaturi. Izjavljujem da nijedan dio rada nije napisan na nedozvoljen način, odnosno da je prepisan iz necitiranog rada, te da nijedan dio rada ne krši bilo čija autorska prava. Također izjavljujem da nijedan dio rada nije korišten za bilo koji drugi rad u bilo kojoj drugoj visokoškolskoj, znanstvenoj ili obrazovnoj ustanovi.

Tea Miličević
(potpis)

Sadržaj

1. Uvod

Cilj ovog diplomskog rada je na razumljiv način predstaviti proces izrade poslovne aplikacije. U posljednjem desetljeću stavio se naglasak na cjelokupni životni ciklus izrade aplikacije, a ne samo na njezino programiranje. Jedan od niza autora koji su se prvotno počeli zalagati za kvalitetnu pripremu prije samog kodiranja bio je inženjer Joel Spolsky. Spolsky (2000) navodi kako je neplaniranje izrade projekta, prije implementacije rješenja, najveći nepotrebni rizik. Sukladno tome, i uzevši u obzir suvremeni pristup izradi aplikacija, cilj autora ovog rada je predstaviti faze procesa izrade poslovne aplikacije za potrebe fiktivnog poduzeća.

Ponajprije će se u radu definirati temeljni pojmovi koji će biti polazišna točka za razumijevanje sintakse cjelokupnog rada. Za potrebe prijevoda pojmova koristit će se rječnik Stranice GNU tima za hrvatske prijevode[1]. Nadalje, bit će definiran opseg pojma poslovne aplikacije jer je bitno razumjeti o kakvoj vrsti aplikacije će se raditi u ovom radu. Tako je, najjednostavnije objašnjeno, poslovna aplikacija vrsta aplikacije koja se bavi automatizacijom određenog poslovnog procesa. Nadalje, objasnit će se odnos pojmova aplikacija i softver. Potrebno je ukazati na to da razlika između navedenih pojmova postoji i da je shvaćanje iste jedan od glavnih koraka razumijevanja cjelokupnog procesa projektnog razvoja.

Proces izrade aplikacije počinje razgovorom s korisnikom. Za potrebe ovog rada autor je oformio izmišljeno poduzeće koje dolazi sa zahtjevom za izradu aplikacije. U drugom poglavlju rada će se objasniti proces analize korisničkih zahtjeva, a u sljedećem poglavlju bit će riječ o izradi plana aplikacije na temelju analize korisničkih zahtjeva. Objasnit će se što će aplikacija raditi i koja metodologija je odabrana za izradu aplikacije. Kako je naglasak rada na praktičnom primjeru, autor će ugrubo objasniti teorijsku podlogu po kojoj će razvijati aplikaciju te zbog opsega rada neće ulaziti u detaljnija razmatranja.

U sljedećem poglavlju u radu će biti objašnjene funkcionalne i nefunkcionalne, odnosno, tehničke specifikacije aplikacije. Objasnit će se značajke koje će aplikacija imati te načine na koji će korisnici moći koristiti aplikaciju. Šesto poglavlje odnosit će se na izradu baze podataka. Baza će biti oformljena u razvojnom okruženju SQL Server Management Studio. U sljedećem poglavlju će se aplikacija programirati u programskom jeziku C# koristeći razvojno okruženje Visual Studio. U završnom poglavlju bit će riječ o testiranju i

[1] GNU rječnik, https://www.gnu.org/server/standards/translations/hr/?fbclid=IwAR2MBmA4vQatI24vJg_uXJVNeiYPAZo7oFUUe8_XBNITF8lojp-DmNdwsUo

mogućnostima daljnjeg razvoja aplikacije, odnosno održavanja. Ovo poglavlje bit će ujedno i završna cjelina ciklusa postupne izrade aplikacije.

2. Temeljni pojmovi

Namjena ovog poglavlja je predstaviti temeljne pojmove koji su potrebni za razumijevanje općenitog procesa izrade poslovne aplikacije. Termin poslovna aplikacija (engl. *business application*) se definira kao skup komponenti koje pružaju poslovnu funkcionalnost, koja se može koristiti interno, eksterno ili s drugim poslovnim aplikacijama[2]. Blaće (2015) nudi sličnu definiciju poslovne aplikacije, kao vrste računalne aplikacije čija je svrha obavljanje određenog poslovnog procesa. Kako bi se termin kvalitetnije razumio potrebno je dati objašnjenje pojma aplikacija. Hrvatska mrežna enciklopedija definira aplikaciju kao skup uputa koje omogućuju izvršenje određenog zadatka[3]. Aplikacija rađena za ovaj rad pripada *stand-alone* vrsti. Stand-alone aplikacije su računalni programi koji rade na osobnim računalima ili one koje pokreću mobilni uređaji, a uključuju svu potrebnu funkcionalnost i nije potrebno spajanje na mrežu za rad s istima (Sommerville, 2011). Primjer takvih aplikacija su uredske, poslovne, aplikacije.

Aplikacija se razvija unutar softvera. Naime, softver (programska podrška) ne uključuje samo aplikaciju (program) već i svu povezanu dokumentaciju, procedure, funkcije, pakete i drugo koji su povezani s računalnim sustavom[4]. Glavne karakteristike softvera prema Sommerville (2016) su:

1. prihvatljivost (engl. *acceptability*) – softver mora biti jasan korisnicima i kompatibilan s operacijskim sustavom na kojem će se pokretati;

2. pouzdanost (engl. *dependability*) – softver mora biti osiguran da će ga koristiti samo zaposlenici poduzeća za koji je napravljen (engl. *login credentials*);

3. efikasnost (engl. *efficiency*) – softver ne smije trošiti previše memorije, mora biti intuitivan i brz;

4. mogućnost održavanja (engl. *maintainability*) – mora biti napravljen tako da može evoluirati i mijenjati s obzirom na potrebe korisnika.

Sommerville (2016) dijeli softvere prema načinu razvoja, a razlikuje dvije vrste: generičke i prilagođene (naručene). Generički softver (engl. *generic*) je softver koji je razvijen od strane neke organizacije i „pušten“ na otvorenom tržištu bilo kojem kupcu koji je u mogućnosti kupiti

---

[2] IBM, https://www.ibm.com/support/knowledgecenter/en/SSPLFC_7.3.0/com.ibm.taddm.doc_7.3/UserGuide/c_cmdb_business_apps.html
[3] Hrvatska enciklopedija, https://www.enciklopedija.hr/Natuknica.aspx?ID=3306
[4] Merriam-Webster dictionary, https://www.merriam-webster.com/dictionary/software

ga. Primjerice, sustav za bilježenje zubarskih kartona. Prilagođeni softver (engl. *customized* ili *bespoke*) je softver koji je napravljen za određenog kupca i odgovara na njegove želje i potrebe.

Općenito, softverski projekt može uključivati ali nije ograničen na koncepciju, dizajn, razvoj, testiranje, implementaciju i održavanje (Hecksel, 2004). Navedeni postupak razvoja softvera, odnosno aplikacije, naziva se softverski proces. Prema Sommerville (2016), softverski proces je niz aktivnosti koje vode do proizvodnje softverskog proizvoda. Postoje četiri temeljne aktivnosti od kojih se sastoji svaki softverski proces (Sommerville, 2016):

1. specifikacija – proces koji se sastoji od komunikacije između klijenta i razvojnog tima o značajkama i izgledu aplikacije;
2. razvoj – implementacija softvera, način na koji je on dizajniran i programiran;
3. validacija – razgovor s klijentom ispunjava li softver njihova očekivanja;
4. evolucija – sve aktivnosti koje uključuju modificiranje softvera da odgovara razvoju tehnologije.

Potrebno je naglasiti da navedene faze nisu ograničene jer je svaki projekt drugačiji te se sami aspekti mogu preklapati jedni s drugima. U slučaju softverskog projekta ovog rada faze će biti sljedeće: planiranje, analiza korisničkih zahtjeva, izrada specifikacije, implementacija, tj. programiranje te testiranje i održavanje. Svaka od navedenih faza činit će pojedino poglavlje ovog rada dok će najveća pažnja biti usmjerena na programiranje aplikacije.

3. Analiza korisničkih zahtjeva

Kreirano je imaginarno poduzeće koje se bavi cipelama (tzv. klijent) za čiju će se poslovnu potrebu izraditi aplikacija „Trgovina“. Prilikom sastanka s klijentom, tj. naručiteljem, isti je iznio svoje želje i potrebe koje su formirane u obliku korisničkog zahtjeva (engl. *application request*). Zahtjev klijenta se odnosi na sve usmene i pismene popise funkcionalnosti i značajki koje aplikacija mora imati. Klijent je podnio zahtjev za aplikaciju čija će svrha biti olakšanje načina vođenja financijskih dokumenata u poduzeću (ponuda, narudžbenica i računa). Dakle, cilj nije stavljen na skladišno poslovanje poduzeća već na financijsko, s mogućnošću proširenja aplikacije, no o tome u ovome radu, zbog opsega, neće biti riječ. Razgovor s korisnikom je interaktivan i iterativan proces u kojemu se:

- istražuje što klijent želi,
- te želje se analiziraju i bilježe,
- odlučuje se razlika između potreba i želja klijenta (ono što aplikacija mora imati i ono što bi bilo „simpatično“ da aplikacija ima),
- provjerava mogućnost izvođenja,
- pregovara način izvođenja,
- provjerava dvosmislenost klijenta,
- formiraju se značajke aplikacije (engl. *feature*),
- zaključno, potvrdom od strane korisnika, počinje faza implementacije (Sommerville, 2016).

Potrebno je naglasiti da prilikom razgovora s klijentom klijent ne mora nužno biti osoba tehničke pozadine te značajke koje se klijentu „podrazumijevaju“ ne znači nužno da su definirane razgovorom. Neki od mogućih problema kod kojih može doći kod razgovora s klijentom su (Sommerville, 2016): problemi u razumijevanju, često mijenjanje zahtjeva, dvosmisleni zahtjevi, mogućnost konflikta, mogućnost izostavljanja „očitih informacija“, mogućnost da klijent ne posjeduje znanje o tome što mu je potrebno i slično. Zato je bitno da zahtjevi budu jasni, nedvosmisleni i potpuni. Kako bi zahtjev bio potpun, i obje strane dogovora bila sigurne u izgled i značajke softvera, preporučuje se odlazak na radno mjesto i uviđanje problema koji klijent aplikacijom želi riješiti. Prilikom odlaska u poduzeće potrebno je uvidjeti sljedeće stavke: kako trenutno izgleda i funkcionira poslovni proces, kako se i na koji način vode podaci u poduzeću (od kojih će se većinom kreirati baza podataka za aplikaciju) te koliko zaposlenika sudjeluje u poslovnom procesu.

4. Planiranje aplikacije

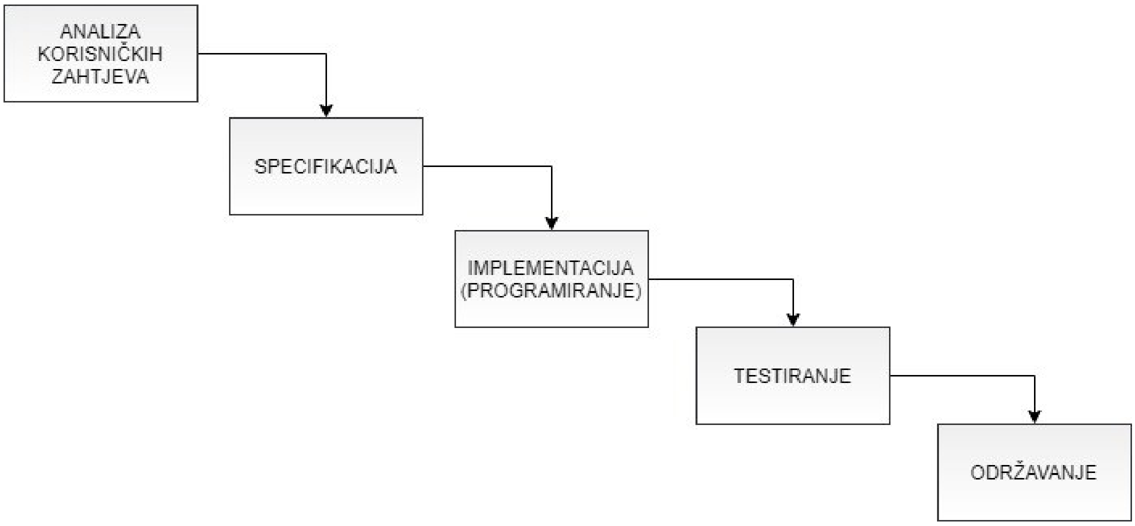
4.1. Opis aplikacije

Kao rješenje problema klijenta kreirana je aplikacija naziva „Trgovina“. Aplikacija će biti jednostavna, intuitivna i u potpunosti će odgovarati zahtjevima korisnika. Prilikom ulaska u aplikaciju na temelju korisničkog imena davat će se ovlasti nad određenim dijelom aplikacije. Administrator će moći dodavati korisnike i uloge korisnicima. Korisnici neće imati uvid u navedene stavke. U aplikaciji će biti omogućeno dodavanje ponuda partnerima, izrađivanje i pregled narudžbenica, kreiranje ulaznih i izlaznih računa i pregled izvješća.

4.2. Metodologija izrade aplikacije

U području razvoja softvera oformila se želja za formiranjem temeljnih standarda ili disciplina koji su zajednički svim projektima, a mogu se predvidjeti tako da će ih svaki projekt sadržavati, bez obzira na to što na veličinu i kompleksnost projekta ovisi niz faktora, poput utjecaja ljudi koji traže proizvod ili utjecaja ljudi koji proizvod formiraju (Hecksel, 2004). Zbog takvog načina razmišljanja oformljene su takozvane metodologije softvera (engl. *methodologies for software development*). Prema Heckselu (2004) metodologija je konstrukcija koja uključuje popis uloga, vještina, timskog djelovanja, aktivnosti, tehnika, standarda, rezultata i općenite kulture organizacije koja uključuje razvoj softvera. Drugim riječima, metodologija je opis procesa razvoja softvera. Postoje dvije glavne podjele metodologija (Hecksel, 2004): teške (engl. *heavyweight*) i lake (engl. *lightweight*). Teške metodologije uključuju sve one koje zahtijevaju velike količine dokumentacije i tijek projekta je većinom ne-iterativan i razvija se prema planu (Hecksel, 2004). Primjer takve metodologije je model vodopada. Lake metodologije se odnose na to da je tijek razvoja projekta iterativan, poput primjerice XP modela (engl. *eXtreme Programming model*) koji uključuje razvoj dijela aplikacije, revizije i testiranja te ponavljanje navedenog obrasca dok projekt ne bude završen (Hecksel, 2004).

Prema Sommerville (2016) ne postoji univerzalni procesni model koji je odgovarajući za svaku izradu softvera. Odabir modela ovisi o tipu softvera, tehnologijama izrade i korisnikovim zahtjevima i potrebama. Za potrebe ovog rada odabran je model vodopada (slika 1). Model vodopada je najpoznatiji pristup procesu izrade softvera. On uključuje procesne aktivnosti analize i definiranja zahtjeva, dizajna softvera, specifikacije, razvoja odnosno implementacije, integracije, validacije i evolucije i predstavlja ih kao odvojene faze procesa (Sommerville, 2016).



Slika 1. Prikaz primjene modela vodopada na primjeru izrade aplikacije .

5. Specifikacija

Specifikacija je jasan i razumljiv dokument koji opisuje na koji način softver treba raditi te koje su njegove funkcionalne i nefunkcionalne karakteristike (Beciric, 2017). Pojednostavljeno, specifikacije su dokumenti koji sadrže obilježja aplikacija. Neki autori, poput Levensona (1996) tvrde da je loša ili nepostojeća specifikacija, uz nepotrebnu kompleksnost koda i manjak testiranja, jedna od glavnih karakteristika lošeg projekta. Joel Spolsky (2000) navodi razloge pisanja specifikacije: daje uvid u dizajn programa, moguće ju je izmjenjivati dok potpuno ne odgovara potrebama korisnika, čuva vrijeme na „nepotrebnu“ komunikaciju odnosno nesporazume, olakšava kreiranje rasporeda i smanjuje vrijeme trajanja projekta. Specifikacije se pišu uvijek prije programiranja. Naime, na početku projekta korisnički zahtjevi konstruirani su na prirodnom i neformalnom jeziku te se trebaju „pretočiti“ u dizajn. Ako se taj korak preskoči vrlo je teško pisati programski kod i iterirati ga da odgovara izgledom i funkcijom korisniku (Spolsky, 2000). Spolsky tvrdi da se nitko ne osjeća loše ako obriše paragraf u Wordu prilikom izrade specifikacije, no stvar je drugačija kada je riječ o brisanju koda u programu. Prema McGuire (2016) cilj pisanja specifikacije je razjasniti koje će biti značajke programa, postaviti temelje što će finalni proizvod biti i odgovarati na pitanja kako, na koji način, gdje i zašto će se raditi. Ona treba sadržavati: sažetak što će aplikacija raditi, dokumentirati sve korisnike aplikacije, dokumentirati sve procese kroz koje korisnik treba prolaziti u aplikaciji, izraditi žičane okvire (engl. *wireframe*) svih ekrana u aplikaciji i popisati sve značajke aplikacije.

Specifikacije se prema Spolskyju dijele na dvije vrste: funkcionalne i tehničke. Funkcionalna specifikacija opisuje način na koji aplikacija radi sa stajališta korisnika (Spolsky, 2000). Ne opisuje načine na koje je ona implementirana već samo njene značajke, ekrane, izbornike i slično. Tehnička specifikacija opisuje način na koji je ona implementirana (Spolsky, 2000). Opisuje strukture podataka, modele relacijskih baza podataka, izbor tehnologija, algoritme i drugo.

5.1. Funkcionalna specifikacija

Funkcionalna specifikacija je formalni dokument koji služi za detaljan opis mogućnosti proizvoda, značajki i izgleda proizvoda te načina na koji izgleda interakcija korisnika s aplikacijom (Rouse, 2007). Ona je smjernica i kontinuirana referentna točka programerima koji pišu kod (Rouse, 2007). Karakteristike funkcionalne aplikacije su (Rouse, 2007): daje softverskim inženjerima točan vremenski okvir cijelog postupka, omogućuje transparentnost među svim sudionicima projekta, pomaže članovima ne-tehničkog projekta da razumiju različite procese testiranja i razvoja te pomaže u određivanju funkcionalnosti aplikacije. Pojednostavljeno, funkcionalna specifikacija je popis značajki sustava i načina na koji on funkcionira, a mogu je pročitati svi članovi projektnog tima.

U odnosu na definirano, popisat će se popis značajki aplikacije „Trgovina“:

1. Prilikom ulaska u aplikaciju tražit će se unos korisničkog imena i zaporke. Ako se podaci ne unesu ili su netočni neće biti moguće ući u aplikaciju. Ako su podaci točni prikazat će se glavni ekran aplikacije.
2. Prilikom ulaska u aplikaciju u donjem desnom kutu nalazit će se pismeni prikaz korisnika koji je pristupio aplikaciji.
3. Aplikacija će biti napravljena tako da zauzima cijeli ekran (engl. *full screen*).
4. Ulaskom i navigiranjem kroz aplikaciju grafičko korisničko sučelje (engl. *GUI – graphical user interface*) će biti sive boje kako se ne bi odvajale komponente.
5. Postoje dvije uloge korisnika aplikacije prilikom prijave: korisnik i administrator. Administrator je zaposlenik poduzeća, voditelj odjela u poduzeću i nadređeni radnicima (korisnicima). Korisnik je osoba koja je zaposlenik poduzeća, nadređena mu je osoba voditelj odjela (administrator) i nema sve ovlasti u aplikaciji. Korisnik nema uvid i pristup ažuriranju podataka drugih korisnika kao ni njihovim ulogama u aplikaciji. On može raditi s ponudama, narudžbenicama, računima i izvješćima. Korisnik pristupa aplikaciji s KOR akreditacijom. Administrator ima sve ovlasti u aplikaciji, on može sve što može korisnik te uz to može uređivati korisnike i njihove uloge u aplikaciji. Administrator pristupa aplikaciji s ADMIN akreditacijom.
6. Ako je korisnik pristupio aplikaciji s KOR akreditacijom, bit će vidljive sljedeće stavke:
    - Datoteka – Izlaz
    - Matični podaci – Artikli, Partneri, Poslovne jedinice, Grupe artikala, Jedinice mjere, Vrste dokumenata, Porezne grupe, Način plaćanja, Države, Mjesta
    - Ulazni dokumenti – Narudžbenice, Ulazni račun, Knjiga ulaznih računa
    - Izlazni dokumenti – Ponude, Račun otpremnica, Knjiga izlaznih računa
    - Izvještaji – Stanje partnera, Knjiga ulaznih računa, Knjiga izlaznih računa
    - Pomoć – Upute za upotrebu, O aplikaciji
7. Ako je korisnik pristupio s ADMIN akreditacijom, bit će vidljive sljedeće stavke:
    - Datoteka – Postavke (Korisnici, Korisnici uloge, Uloge), Izlaz

- Matični podaci – Artikli, Partneri, Poslovne jedinice, Grupe artikala, Jedinice mjere, Vrste dokumenata, Porezne grupe, Način plaćanja, Države, Mjesta
- Ulazni dokumenti – Narudžbenice, Ulazni račun, Knjiga ulaznih računa
- Izlazni dokumenti - Ponude, Račun otpremnica, Knjiga izlaznih računa
- Izvještaji - Stanje partnera, Knjiga ulaznih računa, Knjiga izlaznih računa
- Pomoć – Upute za upotrebu, O aplikaciji

5.2. Tehnička specifikacija

Tehnička specifikacija je dokument koji opisuje tehničke zahtjeve koje proizvod, postupak ili usluga mora ispuniti kako bi bili u skladu s funkcionalnom specifikacijom[5]. Korisne strane pisanja tehničke specifikacije su (Cooper, 2020): mogućnost predviđanja potencijalnih problema prije pisanja koda, pospješuje komunikaciju svih ljudi u timu koji razvijaju aplikaciju, ubrzava izradu aplikacije, potiče "brain storming" i slično. Tehnička specifikacija sadrži podatke o tome na koji način će aplikacija (softver) funkcionirati s hardverom, koje tehnologije će se koristiti za izradu aplikacije, na kojem operativnom sustavu će biti razvijena, na koji način će biti osiguran ulaz u aplikaciju i slično.

U odnosu na opisano, definirat će se popis značajki aplikacije „Trgovina":

1. Ulazak u aplikaciju bit će moguć jedino uz korisničko ime i valjanu zaporku. Tako će se osigurati sigurnost aplikacije.
2. Podržana platforma za pokretanje aplikacije bit će Windows (od verzije 7 do najnovije verzije).
3. Aplikacija neće raditi na webu ili mobilnim uređajima.
4. Baza podataka za aplikaciju bit će kreirana u razvojnom integriranom okruženju MS SQL Management Studio.
5. Kod za aplikaciju će biti napisan u programskom jeziku C# koristeći integrirano razvojno okruženje Visual Studio.

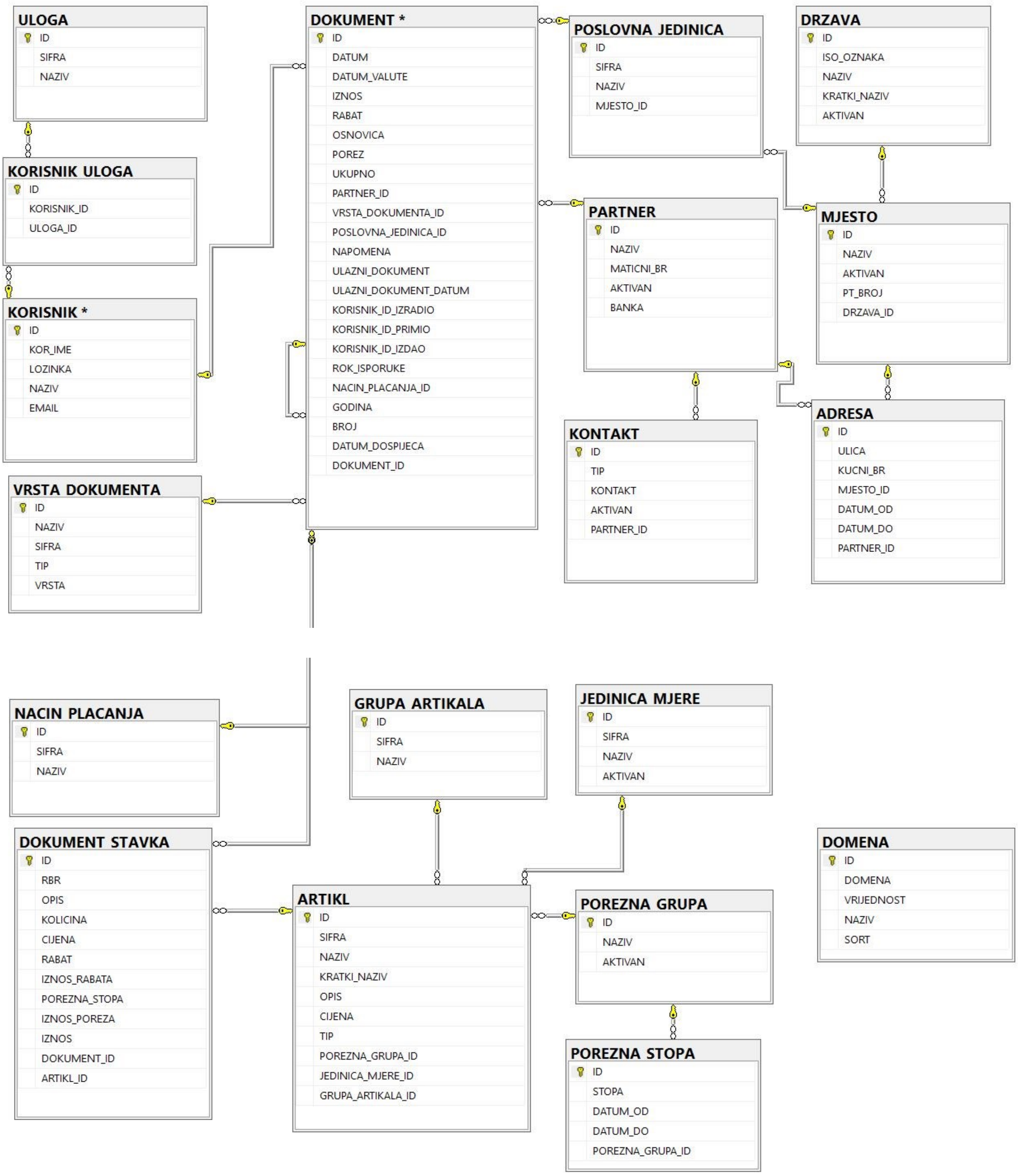
[5] IADC, https://www.iadclexicon.org/technical-specification/

6. Baza podataka

Baza podataka će sadržavati sljedeće tablice:

- **Adresa** (ID, ulica, kucni_br, mjesto_ID, datum_od, datum_do, partner_ID)
- **Artikl** (ID, sifra, naziv, kratki_naziv, opis, cijena, tip (usluga/artikl), porezna_grupa_ID, jedinica_mjere_ID, grupa_artikala_ID)
- **Dokument** (ID, datum, datum_valute, iznos, rabat, osnovica, porez, ukupno, partner_ID, vrsta_dokumenta_ID, napomena, ulazni_dokument, ulazni_dokument_datum, korisnik_ID_izradio, korisnik_ID_primio, korisnik_ID_izdao, rok_isporuke, nacin_placanja_ID, godina, broj, datum_dospijeca, dokument_ID)
- **Dokument_stavka** (ID, rbr, opis, kolicina, cijena, rabat, iznos_rabata, porezna_stopa, iznos_poreza, iznos, dokument_ID, artikl_ID)
- **Domena** (ID, domena, vrijednost, naziv, sort)
- **Drzava** (ID, iso_oznaka, naziv, kratki_naziv, aktivan)
- **Grupa_artikala** (ID, sifra, naziv)
- **Jedinica_mjere** (ID, sifra naziv, aktivan)
- **Kontakt** (ID, tip (e-mail, telefon, mobitel), kontakt, aktivan, partner_ID)
- **Korisnik** (ID, kor_ime, lozinka, naziv, email)
- **Korisnik_uloga** (ID, korisnik_ID, uloga_ID)
- **Mjesto** (ID, naziv, aktivan, pt_broj, drzava_ID)
- **Nacin_placanja** (ID, sifra, naziv)
- **Partner** (ID, naziv, maticni_br, aktivan, banka)
- **Porezna_grupa** (ID, naziv, aktivan)
- **Porezna_stopa** (ID, stopa, datum_od, datum_do, porezna_grupa_ID)
- **Poslovna_jedinica** (ID, sifra, naziv, mjesto_ID)
- **Uloga** (ID, sifra, naziv)
- **Vrsta_dokumenta** (ID, naziv, sifra, tip (ulazni/izlazni), vrsta)

Na slici 2. prikazan je dijagram baze podataka.



Slika 2. Dijagram baze podataka.

7. Programiranje u Visual Studiju

7.1. Karakteristike objektno orijentiranog programiranja

Objektno orijentirano programiranje je naziv za pristup pri razvijanju softvera koji se temelji na objektima i odnosima među tim objektima (Clark, 2013). Koncept objektno orijentiranog jezika za razvoj softvera se počeo primjenjivati sredinom 1960-tih godina, a do sredine 1980-tih godina objektno orijentirana paradigma bila je središte razvoja poslovnih softvera. Kako bi bio u korak s promjenama u svijetu razvoja softvera, Microsoft je 2002. godine objavio .NET Framework i programski jezik C#. Razlog tome je bio taj što su se poslovni softveri razvijali u jezicima poput C-a i Pascala koji su proceduralno orijentirani jezici, odnosno, program napisan u istima se razvija linearno, jedna operacija za drugom (Clark, 2011). Prema Clark (2011) navedena vrsta programiranja je dobro funkcionirala za razvoj manjih poslovnih softvera koji su se sastojali od nekoliko stotina linija koda, ali kako su programi postali veći tako je postalo teško upravljati istima i otklanjati greške. Naime, upravo je navedeno razlog prihvaćanja objektno orijentirane paradigme kao polazišne točke pri programiranju poslovnih aplikacija te razlog zašto je odabran C# kao programski jezik za izradu aplikacije ovog rada. Prema Clarku (2011) prednosti objektno orijentiranog programiranja uključuju:

- intuitivniji prijelaz s modela poslovne analize u model implementacije softvera;
- mogućnost kvalitetnijeg i bržeg održavanja i provođenja promjena u programima;
- mogućnost učinkovitijeg kreiranja informacijskih sustava pomoću timskog procesa, omogućujući specijalistima za određeno područje za rad na tom dijelu softvera;
- mogućnost ponovne upotrebe koda u drugim programima i kupnje komponenti napisanih u od strane drugih developera za povećavanje funkcionalnosti;
- poboljšana integracija s modernim operativnim sustavima;
- mogućnost stvaranja intuitivnijeg grafičko-korisničkog sučelja.

Nadalje, bitno je naglasiti karakteristike, odnosno, temeljne principe objektno orijentirane paradigme (Clark, 2011):

- Objektno orijentirana paradigma bazira se na objektima. Objekti su strukture koje uključuju podatke i postupke za rad s tim podacima. Oni su instance klasa. Klasa je tip podatka koji definira predložak kojim se opisuju zajednička svojstva svih objekata koji su proizašli iz navedene klase. Ona se sastoji od atributa (svojstva) i ponašanja (engl. *behaviour*).

- Svojstvo apstrakcije omogućuje da objekti uključuju samo one informacije koje su relevantne u kontekstu aplikacije. Primjerice, prilikom izrade aplikacije za dostavu kreira se objekt „proizvod“ koji sadržava atribute težina i veličina. Boja tog proizvoda ne bi bila potrebna informacija i s toga se ne bi definirala.
- Svojstvo enkapsulacije omogućuje ne-davanje direktnog pristupa podacima, drugim riječima, skrivanje podataka. Ako želimo pristupiti podacima potrebno je izvršiti interakciju s objektom koji je zadužen za podatke. Drugim riječima, to je postupak ili mehanizam pomoću kojeg se kombiniraju podaci, s kojima će se manipulirati, u jednu cjelinu (Essay Sauce, 2019). Navedeno svojstvo omogućuje pouzdanost i sigurnost, lakše održavanje i olakšava proces uklanjanja grešaka (engl. *debugging*).
- Svojstvo polimorfizma omogućuje da dva različita objekta daju odgovore na istu poruku na jedinstvene načine.
- Sposobnost nasljeđivanja omogućuje klasifikaciju objekata ovisno o njihovim zajedničkim karakteristikama ili funkcijama. Drugim riječima, to je kombinacija općih karakteristika u jednoj nadređenoj klasi (engl. *parent class*) čija će svojstva naslijediti podklase (engl. *child class*). Svaki put kad se promijeni neko od svojstava nadređene klase, promijenit će se isto svojstvo u svim podređenim klasama te klase. Navedeno svojstvo omogućuje lakši rad s objektima i intuitivnost.
- Svojstvo agregacije omogućuje da se objekt sastoji od kompozicije drugih objekata koji rade zajedno. Primjerice, objekt „auto“ sastoji se od objekata „volan“, „motor“, „karoserija“ i drugo. Navedeno svojstvo omogućuje točnu implementaciju i modeliranje poslovnih procesa.

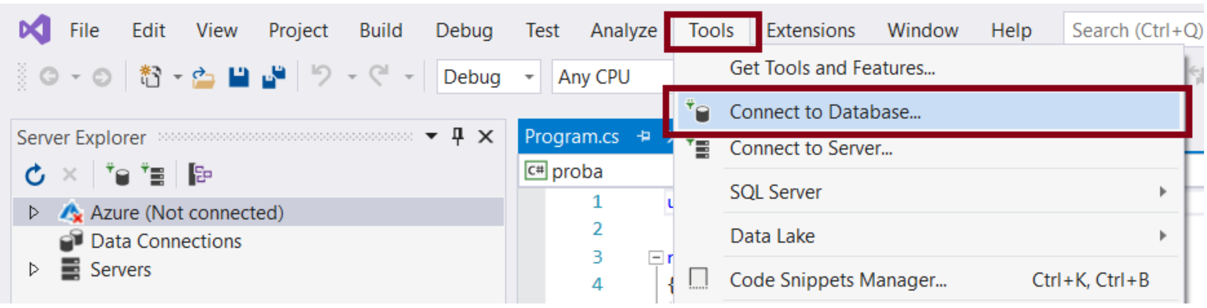
7.2. O programskom jeziku C#

1980-tih godina većina aplikacija koje su radile na operativnom sustavu Windows bile su napisane u programskom jeziku C++. Kako je C++ prvenstveno težak jezik, zbog potrebe programera da vodi računa o memoriji i sigurnosti podataka, razvijen je programski jezik Java. Java je omogućila programeru da se koncentrira na poslovnu logiku i programer se više nije morao nositi s problemom memorije, provjere sigurnosti, uništavanja nepotrebnih objekata i slično (Clark, 2011). Kako je Microsoft uvidio popularizaciju Jave shvatio je potrebu izrade jezika pomoću kojeg će se olakšati rad programerima koji razvijaju softvere za desktop i web okruženja. Zato je napravljen .NET Framework koji se bavio funkcionalnostima poput memorije i sigurnosti. Biblioteke klasa (engl. *class libraries*) .NET Frameworka bile su pisane u novom jeziku naziva C# koji je kreirao Anders Hejlsberg. Kako je Hejlsberg prije razvoja

.NET Frameworka radio s jezicima poput Turbo Pascala i Delphija, C# je rađen na njihovom primjeru, a cilj je bio uklopiti sintaksu sličnu programskom jeziku C kako bi se jezik svidio C++ i Java programerima (Clark, 2011). C# u odnosu na konkurenciju tako omogućuje null tipove podataka, enumeraciju, lambda izraze, izravan pristup memoriji, generičke metode, povećanu sigurnost, iteratore, LINQ i razne druge karakteristike. Prema Microsoft Docs[6] jednostavniji je od programskih jezika C i C++, a fleksibilniji od programskog jezika Java.

7.3. Spajanje na bazu

Nakon kreiranja baze u MS SQL Management Studiju potrebno je povezati bazu s alatom Visual Studio (verzija 2019). Prvi korak je odabir stavke *Connect to Database* unutar alatne trake (stavka Tools, vidi slika 3).

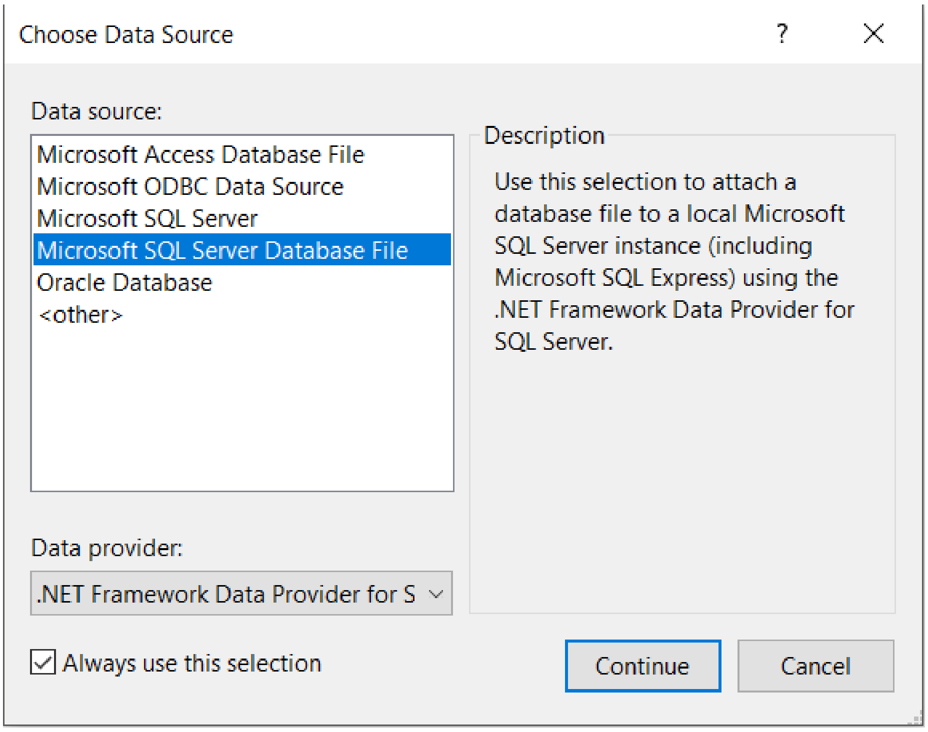


Slika 3. Prikaz prvog koraka.

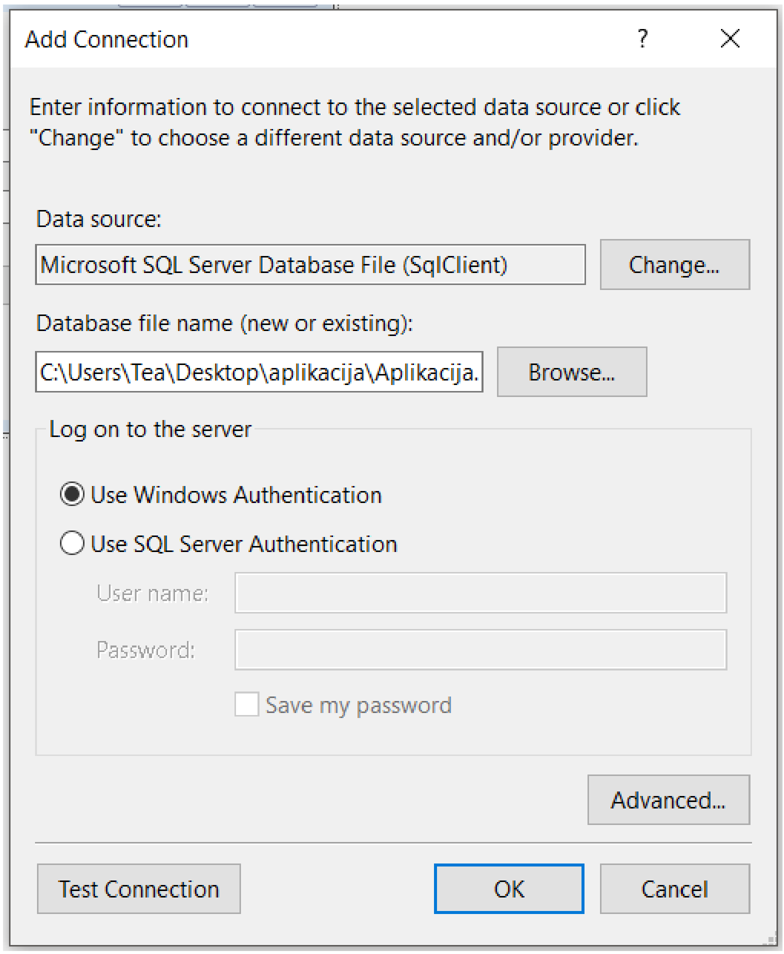
Nakon što se otvori dijaloški okvir potrebno je odabrati *Data source* s kojim će se Visual Studio povezati. Microsoft Docs definira *Data Source* kao skup objekata koji pohranjuju podatke iz baze podataka u memoriju i podržavaju praćenje promjena da bi se omogućili operacije kreiranje, čitanja, ažuriranja i brisanja nad tim podacima bez potrebe da se uvijek povezuje s bazom podataka[7]. Nadalje, za potrebe aplikacije odabran je Microsoft SQL Server Database File. Klikom na OK otvara se okvir (slika 4). Klikom na *Browse* moguće je odabrati bazu (slika 5). Klikom na OK baza je spojena.

[6] https://docs.microsoft.com/en-us/dotnet/csharp/getting-started/introduction-to-the-csharp-language-and-the-net-framework

[7] https://docs.microsoft.com/en-us/visualstudio/data-tools/create-and-configure-datasets-in-visual-studio?view=vs-2019



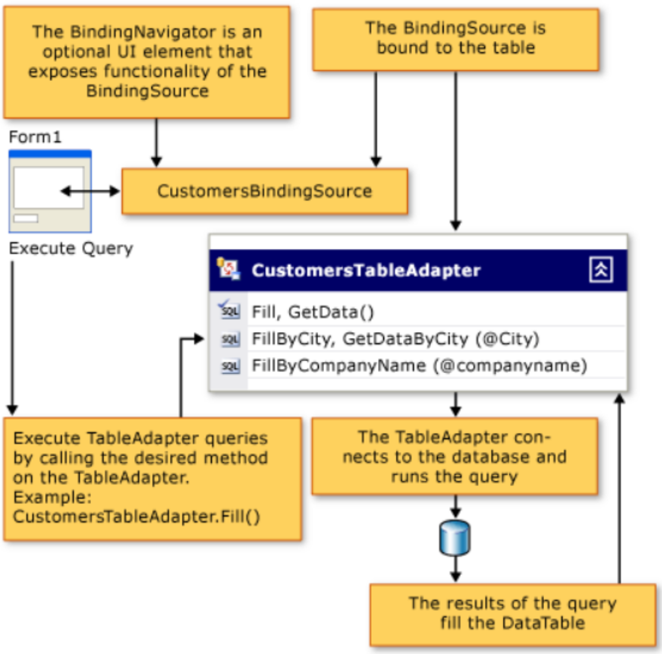
Slika 4. Prikaz drugog koraka.



Slika 5. Prikaz trećeg koraka.

7.4. Dohvaćanje podataka iz baze podataka

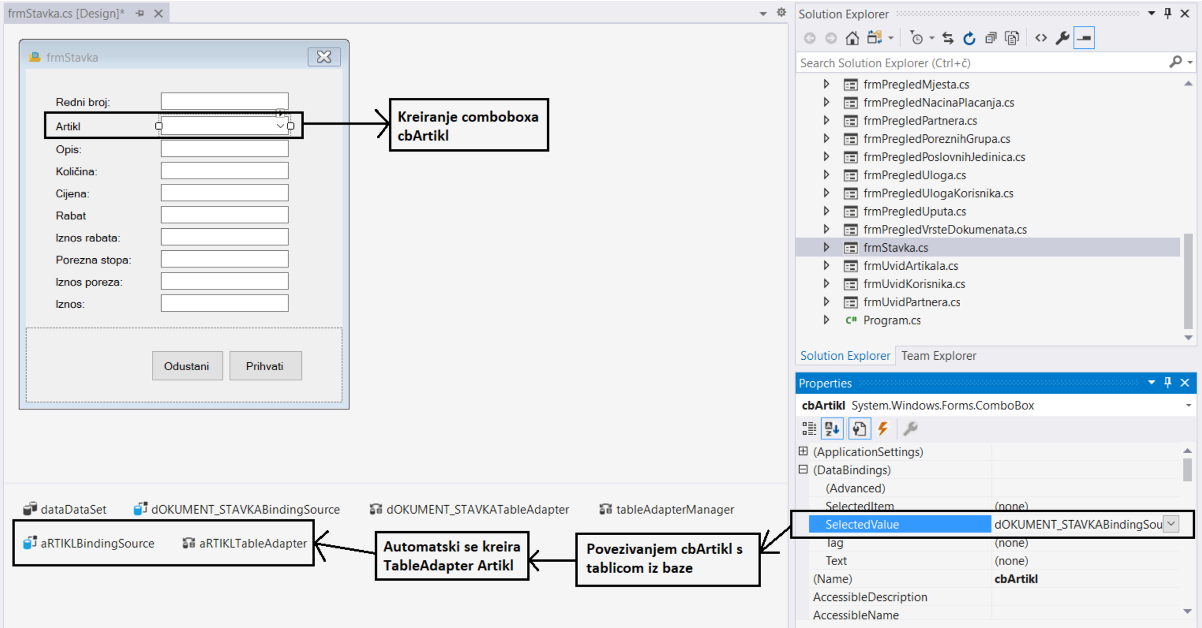
Za dohvaćanje podataka iz baze i komunikaciju baze s aplikacijom, u C#-u koriste se TableAdapteri. TableAdapteri se spajaju na bazu podataka, pokreću upite ili pohranjene procedure i vraćaju novu tablicu podataka ili ispunjavaju postojeću s vraćenim ažuriranim podacima, omogućavaju dodavanja, ažuriranja i brisanja u bazi podataka te izdavanje globalnih naredbi koje nisu povezane s bilo kojom određenom tablicom[8]. Na TableAdapteru se može pokrenuti neograničeni broj upita dok god su vraćeni podaci u skladu sa shemom tablice s kojom je isti povezan. Na slici 6 prikazan je dijagram koji objašnjava način na koji TableAdapteri vrše interakciju s bazom i drugim objektima u memoriji.



Slika 6. Prikaz interakcije TableAdaptera s bazom[9]

Na sljedećoj slici (slika 7) prikazan je postupak kreiranja TableAdaptera u radu. Povučeni su podaci iz tablice DOKUMENT_STAVKA i automatski je kreiran dOKUMENT_STAVKATableAdapter i dOKUMENT_STAVKABindingSource. Kako je Artikl strani ključ na tablicu ARTIKL kreiran je combobox cbArtikl koji je povezan s tablicom DOKUMENT_STAVKA. Automatski se kreirao TableAdapterArtikl.

[8] Microsoft Docs, https://docs.microsoft.com/en-us/visualstudio/data-tools/create-and-configure-tableadapters?view=vs-2019
[9] Microsoft Docs, https://docs.microsoft.com/en-us/visualstudio/data-tools/fill-datasets-by-using-tableadapters?view=vs-2019



Slika 7. Prikaz TableAdaptera Artikl.

7.5. Dodavanje formi

Sve forme unutar aplikacije podliježu određenim osnovnim koracima. Prvo je kreiranje formi. Forma se kreira klikom na *Add – New Form* unutar *Solution Explorera*. Prema Microsoft Docsu[10] forme su temeljne jedinice aplikacije, tj. prazne ploče koje programeri poboljšavaju kontrolama za stvaranje korisničkog sučelja i koda za manipuliranje podacima. Nakon kreiranja, formama se dodjeljuju svojstva koristeći *Application Settings*. Sljedeći korak je dodavanje podataka (ako je potrebno) iz baze što je omogućeno "*drag and drop*" opcijom putem *DataGridView* stavke (unutar *DataSources* komponente). *DataSources* je univerzalni naziv za komponentu unutar razvojnih sučelja koja služi za povezivanje baze s aplikacijom. Nakon toga formi se dodaju određena svojstva, piše se kod za kvalitetniju integraciju svojstava, dodaju se WPF funkcije (primanje poruke od tipkovnice i miša) te se proces iterira dok forma ne bude završena. U kodu forme se povezuju s korisničkim sučeljem pomoću metode ShowDialog(), primjerice:

```
private void korisnikaToolStripMenuItem_Click(object sender, EventArgs e)
        {
            frmPregledKorisnika fForma = new frmPregledKorisnika();
            fForma.ShowDialog();
        }
```

[10] https://docs.microsoft.com/en-us/dotnet/framework/winforms/

8. Forme

8.1. Prijava u aplikaciju

Prilikom ulaska u aplikaciju otvara se forma prijave (engl. *login form*). Forma omogućuje unos dva podatka: *Korisničko ime i Lozinka*. Podaci se unose u tekstualno polje (engl. *textbox*). *Korisničko ime* je vidljivo, no *Lozinka* će biti prekrivena sa znakom zvjezdice (engl. *hidden password*). Nakon što korisnik unese podatke i klikom na „Prihvati" aplikacija se povezuje s bazom (metoda SpojiSeNaBazu()) i poziva se funkcija USER_LOGIN (Prilog 1):

```
private string SpojiSeNaBazu()
        {
            using (SqlConnection conn = new
SqlConnection(Properties.Settings.Default.DataConnectionString))
            using (SqlCommand cmd = new SqlCommand("SELECT dbo.USER_LOGIN('" +
tbUserName.Text +"','"+ tbPassword.Text + "')", conn))
            {
                conn.Open();
                var result = cmd.ExecuteScalar();
                conn.Close();

                return result.ToString();
            }
        }
```

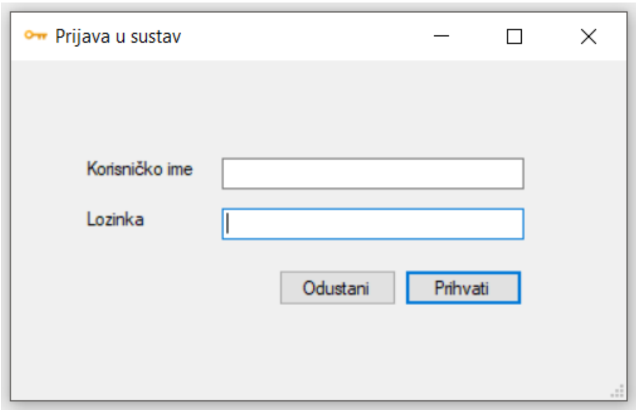
```
CREATE   FUNCTION [dbo].[USER_LOGIN]
(
       @KorIme  nvarchar(100),
       @Lozinka  nchar(60)
)
RETURNS INT
AS
BEGIN
DECLARE @KorisnikID int
SELECT @KorisnikID = ID
       FROM KORISNIK
       WHERE UPPER(KOR_IME) = UPPER(@KorIme)
       AND LOZINKA= @Lozinka;
RETURN @KorisnikID;
END
```

Funkcija prima ulazne parametre „Korisničko ime" i „Lozinka" te provjerava u bazi točnost podataka. Unosom valjanih korisničkih podataka i klikom na *Prihvati* ili pritiskom tipke *Enter*, korisnik ulazi u aplikaciju (Prilog 1):

```
private void btnOK_Click(object sender, EventArgs e)
        {
            string vID = SpojiSeNaBazu();
            if (string.IsNullOrEmpty(vID))
            {
                //ukoliko se podaci koje je korisnik unio NE slažu s onima u bazi,
javlja se poruka greške

                MessageBox.Show("Prijava nije uspjela!", "Trgovina");
```

```
            this.DialogResult = DialogResult.None;
            tbPassword.Text = "";
        }
        else
        {
            fID = Int32.Parse(vID);
            this.DialogResult = DialogResult.OK;
            Close();
        }
    }
```
S druge strane, klikom na *Odustani* zatvara se forma prijave u aplikaciju.

Prvi korak prilikom izrade bilo koje forme u aplikaciji je dodavanje nove stavke (*Add, New Item*) – Windows Form. Sve forme u aplikaciji počinju s kraticom „frm“, primjerice frmLogin. Navedeno svojstvo, koje opisuje na koji način će se određena stavka imenovati i biti dosljedna tom pravilu, se naziva konvencija imenovanja (engl. *naming convention*). Prema Microsoft Docs[11] konvencija imenovanja je skupni naziv za sva pravila za koja bi bilo poželjno da ih se korisnik pridržava, a uključuju izbor riječi, smjernice za korištenje kratica i akronima te preporuke za izbjegavanje korištenja imena specifičnih za jezik. Dodaju se labele (engl. *label*), tekstualna polja za unos podataka (kratica „tb“) i gumbi (kratica „btn“), a pod *Properties* se stavlja odgovarajući tekst. Sljedeće što je potrebno je definirati koju akciju će obavljati gumbi. Pod *DialogResult* u svojstvima kod gumba *Odustani* stavljamo *Cancel*. Kod gumba *Prihvati* potrebno je napisati metodu koja će omogućiti spajanje na bazu i provjeru podataka koje je korisnik unio (slika 8).



Slika 8. Izgled forme prijave.

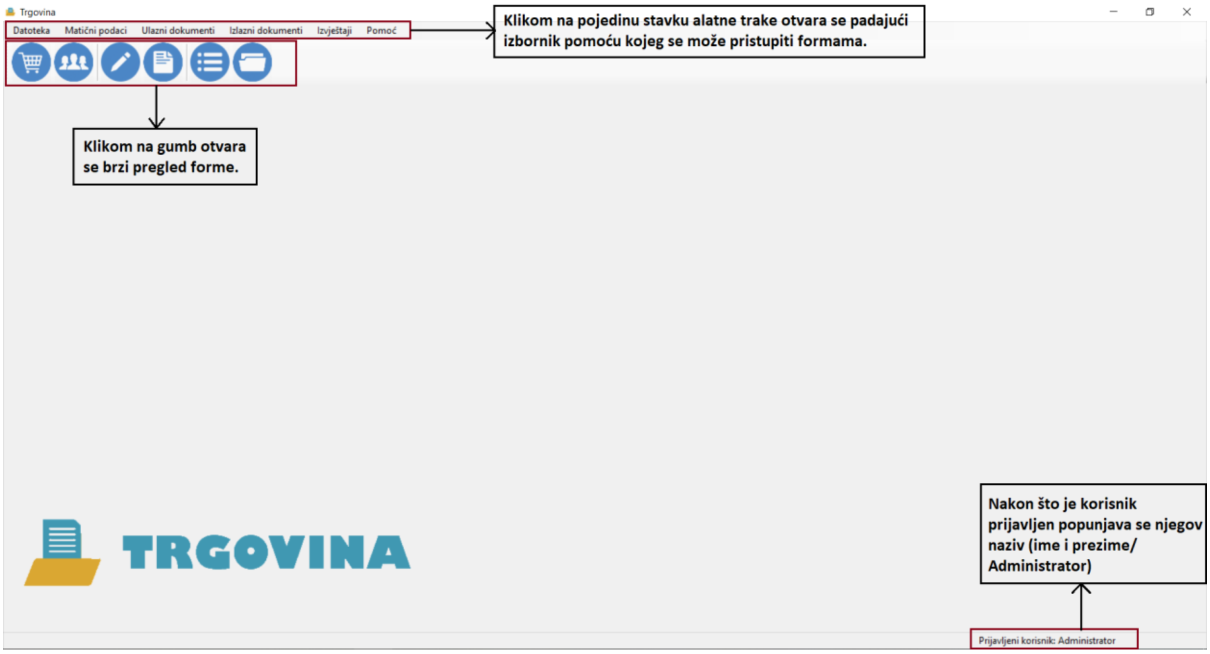
[11] https://docs.microsoft.com/en-us/dotnet/standard/design-guidelines/general-naming-conventions

8.2. Glavni ekran

Nakon što je korisnik pristupio aplikaciji vidljivi su mu glavni izbornik, odnosno izbornička traka (engl. *menu bar*) na kojima su redom izlistani: Datoteka, Matični podaci, Ulazni dokumenti, Izlazni dokumenti, Izvještaji i Pomoć. Ispod glavnog izbornika nalaze se ikone, odnosno gumbi (engl. *button*), koji omogućuju brzi uvid u određene stavke aplikacije. Ukoliko se osoba prijavi kao Administrator ima uvid u cijelu aplikaciju. Ako se prijavi kao Korisnik omogućeno mu je vidjeti samo pojedine dijelove aplikacije (slika 9). Kako bi se ovaj korisnički uvjet mogao ispuniti bilo je potrebno kreirati metodu SakrijMenu() koja će omogućiti spajanje na bazu i provjeru razine ovlasti koju korisnici imaju u veznoj tablici UlogaKorisnik (Prilog 2):

```
private void SakrijMenu()
        {
            int admin = 0;
            using (SqlConnection conn = new SqlConnection(Properties.Settings.Default.DataConnectionString))
            using (SqlCommand cmd = new SqlCommand("SELECT dbo.IS_ADMIN('" + UserID.ToString() + "')", conn))
            {
                conn.Open();
                var result = cmd.ExecuteScalar();
                conn.Close();
                string s = result.ToString();
                if (!string.IsNullOrEmpty(s))
                {
                    admin = int.Parse(s);
                }
            }
            if (admin != 1)
            {
                postavkeToolStripMenuItem.Visible = false;
                toolStripMenuItem1.Visible = false;
            }
        }
```

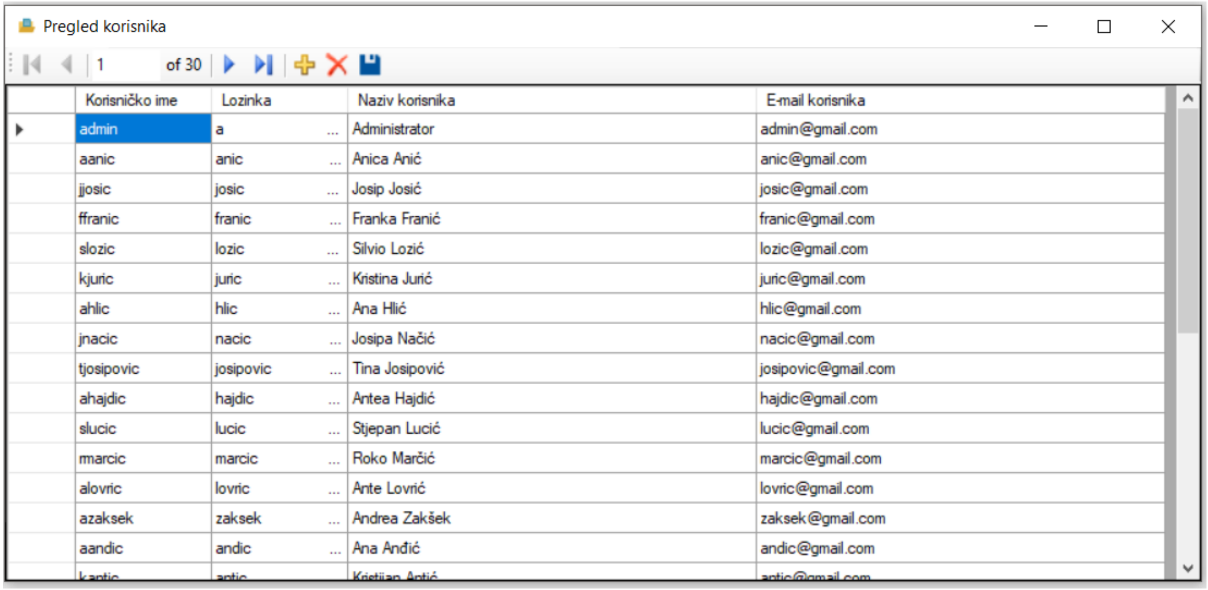
Pod stavkom Datoteka na izborničkoj traci korisnik može izaći iz aplikacije.



Slika 9. Izgled glavnog ekrana aplikacije.

8.3. Korisnici

Unutar aplikacije pod glavni izbornik (Datoteka) moguće je dobiti uvid u popis korisnika i uloga koje imaju (slika 10). Sve forme (Korisnici, Korisnici uloge, Uloge) kreirane su na isti način. Za dodavanje mreže podataka (engl. *data grid*) korišten je objekt *Data Source*. Klikom na tablicu KORISNIK otvara nam se padajući izbornik gdje je moguće odabrati polje *DataGridView* koje se načinom "*drag and drop*" povuče na formu. Uređivanjem kolona (engl. *Edit Colums*) omogućeno je prikazivanje i sakrivanje kolona iz tablice. U formi Pregled uloga korisnika korišten je tip kolone *DataGridViewComboBoxColumn* koji omogućuje povezivanje (engl. *join*) kolona s podacima u tablici (Prilog 3).


Pregled korisnika

1 of 30

	Korisničko ime	Lozinka	Naziv korisnika	E-mail korisnika
▸	admin	a ...	Administrator	admin@gmail.com
	aanic	anic ...	Anica Anić	anic@gmail.com
	jjosic	josic ...	Josip Josić	josic@gmail.com
	ffranic	franic ...	Franka Franić	franic@gmail.com
	slozic	lozic ...	Silvio Lozić	lozic@gmail.com
	kjuric	juric ...	Kristina Jurić	juric@gmail.com
	ahlic	hlic ...	Ana Hlić	hlic@gmail.com
	jnacic	nacic ...	Josipa Načić	nacic@gmail.com
	tjosipovic	josipovic ...	Tina Josipović	josipovic@gmail.com
	ahajdic	hajdic ...	Antea Hajdić	hajdic@gmail.com
	slucic	lucic ...	Stjepan Lucić	lucic@gmail.com
	rmarcic	marcic ...	Roko Marčić	marcic@gmail.com
	alovric	lovric ...	Ante Lovrić	lovric@gmail.com
	azaksek	zaksek ...	Andrea Zakšek	zaksek@gmail.com
	aandic	andic ...	Ana Anđić	andic@gmail.com
	kantic	antic	Kristijan Antić	antic@gmail.com

Slika 10. Izgled popisnih tablica aplikacije.

8.4. Matični podaci

U matičnim podacima korisnici mogu pristupiti pregledu: artikala, partnera, poslovnih jedinica, grupama artikala, jedinicama mjere, vrstama dokumenata, poreznim grupama, načinom plaćanja, državama i mjestima. Sve forme kreirane su na isti način koji je objašnjen u ranijem poglavlju.

8.5. Ulazni i izlazni dokumenti

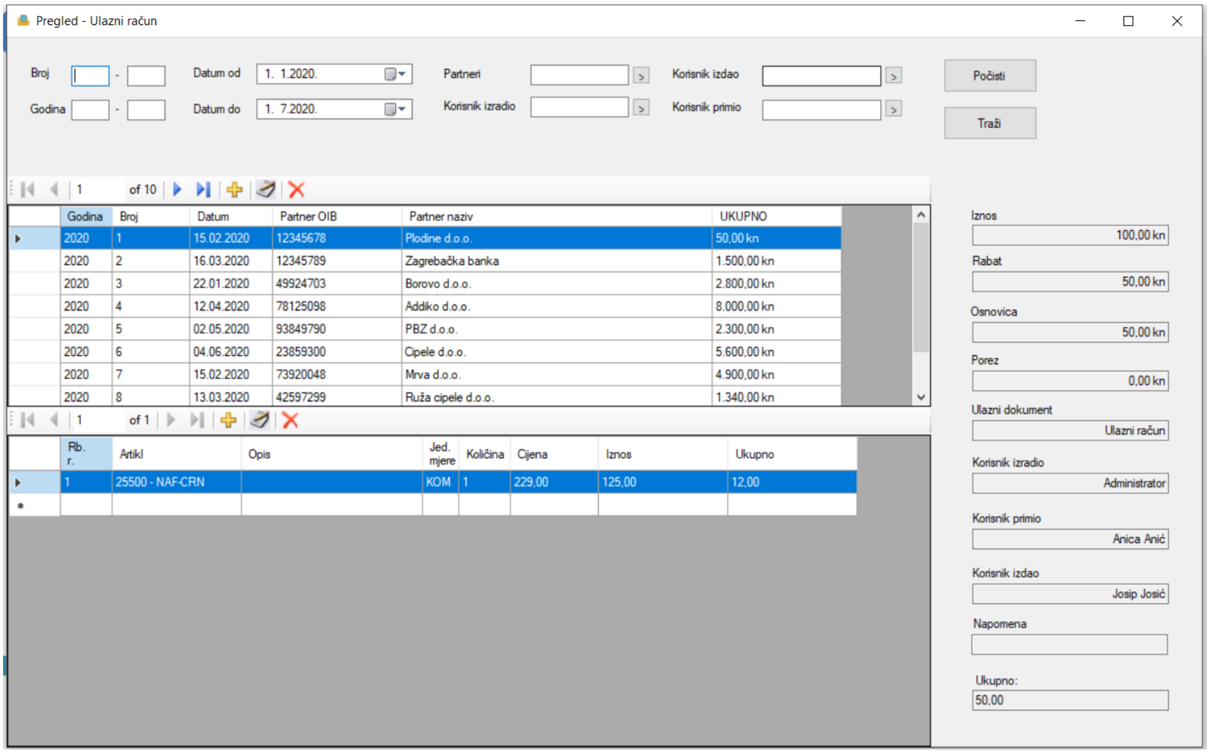
Prilikom odabira polja Narudžbenice, Ulazni računi, Ponude i Računa otpremnice, otvara se forma frmPregledDokumenata (slika 11)(Prilog 4). Forma je za pregled svih dokumenata poduzeća ista, a njene funkcije se mijenjaju ovisno o vrsti dokumenta. Forma se sastoji od četiri ploče (engl. *panel*). Gornja ploča omogućuje upis podataka po kojima korisnik može pretraživati dokumente. Može ih pretraživati po sljedećim stavkama: broj od – do, godina od – do, datum od – do, partneri, korisnik (izradio / izdao / primio). Klikom na gumb „Počisti" otvara se metoda koja stavlja vrijednosti svih polja na prazno. Kako je za datum valute odabrana klasa *DateTimePicker* postavljeno je da datum *od* bude postavljen na prvi dan u trenutnom mjesecu, dok je datum *do* postavljen da pokazuje trenutni datum. Klikom na gumb Traži omogućeno je pretraživanje prema datim parametrima od strane korisnika. Kako bi se ta akcija mogla ispuniti potrebno je bilo kreirati metodu koja će učitavati upisane podatke. Navedena metoda Ucitaj() tako filtrira bazu i traži unesene podatke za prikaz:

```
private void Ucitaj()
        {
```

```
string Filter = "VRSTA_DOKUMENTA_ID = " +
FVrsta_Dokumenta_ID.ToString();

//slaganje filtera broj od

// ! je znak negacije, znači ukoliko string NIJE null ili prazan

if (!string.IsNullOrEmpty(tbBrojOd.Text))

{

   Filter = Filter + " AND BROJ >= " + tbBrojOd.Text;

}

//slaganje filtera broj do

if (!string.IsNullOrEmpty(tbBrojDo.Text))

{

   Filter = Filter + " AND BROJ <= " + tbBrojDo.Text;

}

//slaganje filtera godina do

if (!string.IsNullOrEmpty(tbGodinaDo.Text))

{

   Filter = Filter + " AND GODINA <= " + tbGodinaDo.Text;

}

//slaganje filtera godina od

if (!string.IsNullOrEmpty(tbGodinaOd.Text))

{

   Filter = Filter + " AND GODINA >= " + tbGodinaOd.Text;

}

//slaganje filtera tbPartneri

if (FPartnerID != -1)

{

   Filter = Filter + " AND PARTNER_ID = " + FPartnerID.ToString();

}


//slaganje filtera datum valute od

if (!string.IsNullOrEmpty(pDatumValuteOd.ToString()))
```

```
{
    Filter = Filter + " AND DATUM_D >='" +
pDatumValuteOd.Value.ToString("dd.MM.yyyy") + "'";
}
//slaganje filtera datum valute do
if (!string.IsNullOrEmpty(pDatumValuteDo.ToString()))
{
    Filter = Filter + " AND DATUM_D <= '" +
pDatumValuteDo.Value.ToString("dd.MM.yyyy") + "'";
}
//slaganje filtera Izradio
if (FKorisnikIzradioID != -1)
{
    Filter = Filter + " AND KORISNIK_ID_IZRADIO = " +
FKorisnikIzradioID.ToString();
}
//slaganje filtera Izdao
if (FKorisnikIzdaoID != -1)
{
    Filter = Filter + " AND KORISNIK_ID_IZDAO = " +
FKorisnikIzdaoID.ToString();
}
//slaganje filtera Primio
if (FKorisnikPrimioID != -1)
{
    Filter = Filter + " AND KORISNIK_ID_PRIMIO =  " +
FKorisnikPrimioID.ToString();
vRACUNBindingSource.Filter = Filter;
this.vRACUNTableAdapter.Fill(this.dataDataSet.vRACUN);
}
```

Sljedeća ploča sadržava mrežu podataka računa i stavki artikla. Klikom na određeni dokument automatski se pune podaci koji se nalaze u desnoj ploči i sadrže podatke o računu (iznos, rabat, osnovica, porez, ulazni dokument, korisnik izdao, korisnik primio, korisnik izradio, napomena). Računi, odnosno dokumenti, su prikazani na pločama koji se nalaze u sredini forme i daju uvid u: broj dokumenta, godini u kojoj je izrađen, datumu, OIB-u i nazivu partnera te ukupnom iznosu. U ploči ispod tablice Dokumenti nalazi se tablica Stavke artikla (redni broj, artikl, opis, jedinica mjere, količina, cijena, iznos, ukupno).



Slika 11. Prikaz forme za pregled dokumenata.

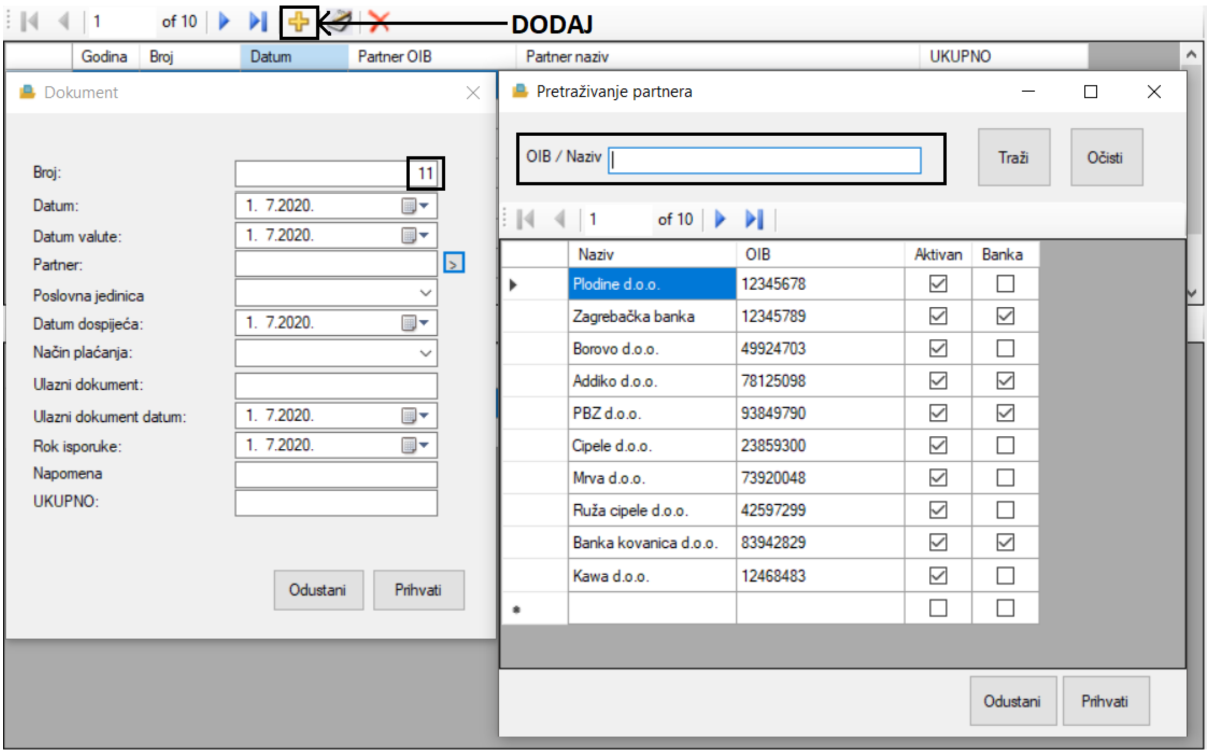
8.5.1. Pregled računa

Dodan je *BindingNavigator* koji omogućuje navigaciju po računima. Klikom na gumb „Dodaj" otvara se forma Dokument (slika 12). Broj je dodan automatski i ovisi o broju prošlog dokumenta (uvećan za 1). Za navedeno je kreirana funkcija NEXT_BROJ koja prima parametar ID vrste dokumenta i vraća vrijednost broja dokumenta po vrsti:

```
CREATE   FUNCTION [dbo].[NEXT_BROJ]
(
       @VrstaDokumentaID int
)
RETURNS int
AS
BEGIN
    DECLARE @Broj int;
       DECLARE @RET int;

       select @Broj = max(d.broj)
       from DOKUMENT d
```

```
    where d.VRSTA_DOKUMENTA_ID = @VrstaDokumentaID
    and d.GODINA = YEAR(GETDATE());

       SET @RET = ISNULL(@Broj,0) + 1;

       RETURN @RET;
END
```

Klikom na gumb Partner otvara se forma Pretraživanje partnera. Klikom na gumb „Traži“ (ili pritiskom tipke *Enter*) pretražuje se opisani partner. Klikom na gumb „Očisti“ briše se upisani upit.



Slika 12. Prikaz izgleda formi klikom na gumb Dodaj.

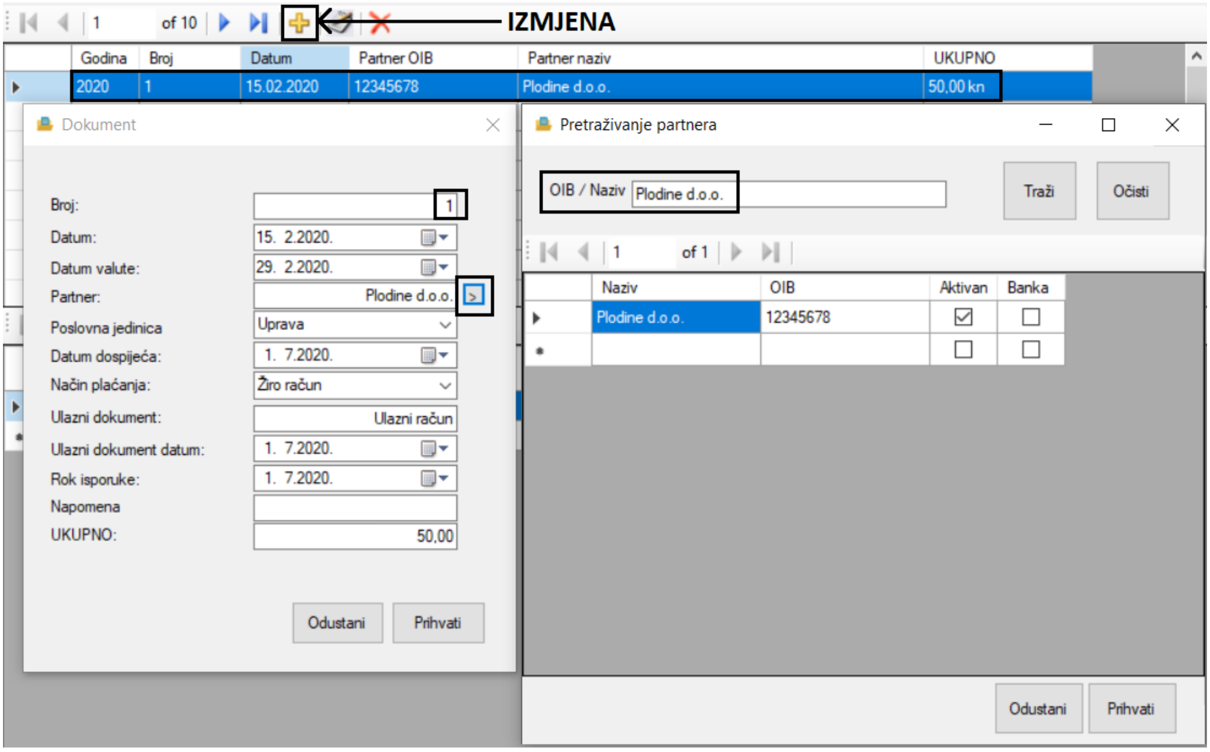
Nadalje, klikom na gumb „Izmijeni“ moguće je izmijeniti odabrani račun. Otvara se forma Dokument u kojoj se automatski pune podaci (slika 13). Klikom na gumb otvara se forma Pretraživanje partnera. Zato što je odabran već postojeći dokument naziv partnera se puni automatski (Prilog 5):

```
private void PotraziPartnera(object sender)
        {
            int max = 2;
            int max2 = 0;
            int l = tbPartneri.TextLength;

            //ako se metoda poziva s gumba zanemari se dužina teksta u kontroli za
naziv partnera
            if (sender is Button)
            {
                max = -1;
```

```
                max2 = -1;
            }

            //ukoliko su uneseni znakovi
            if (l > max2)
            {
                //ukoliko je dužina unesenog partnera dulja od 3 znaka

                if (l > max)
                {
                    string naziv;
                    frmUvidPartnera fForma = new frmUvidPartnera();
                    FPartnerID = fForma.Prikazi(this, tbPartneri.Text, out naziv);
                    if (FPartnerID == -1)
                    {
                        tbPartneri.Text = "";
                    }
                    else
                    {
                        //ispiši naziv
                        tbPartneri.Text = naziv;
                    }
                }
                else
                {
                    //provjeri je li duljina unešenog dulja od 3 znaka, a ako nije
javlja se poruka greške
                    MessageBox.Show("Za pretraživanje unesite bar 3 znaka!", "Info");

                }
            }

        }
```

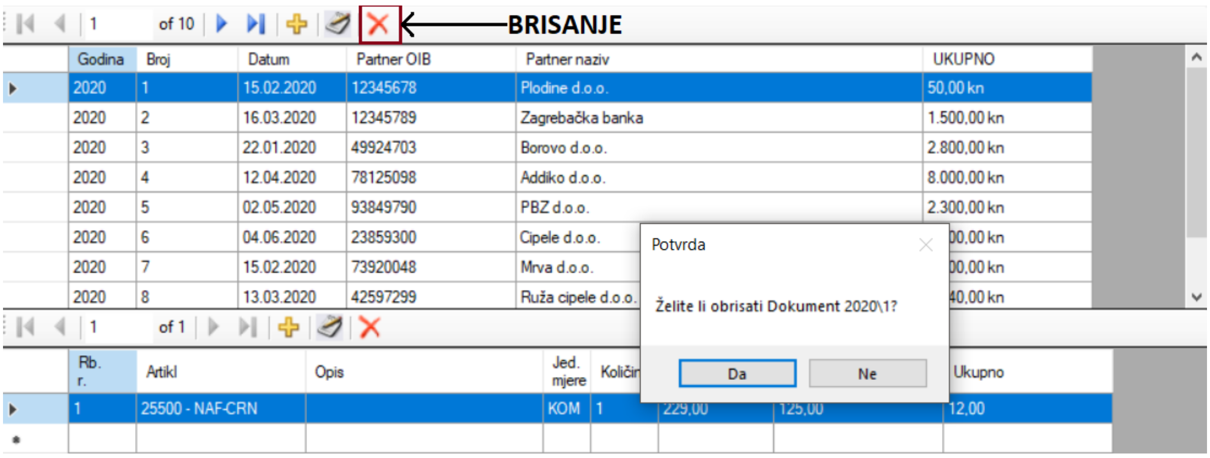
Za to je kreirana funkcija GET_PARTNER koja prima ulazni parametar ID partnera i vraća naziv partnera:

```
CREATE FUNCTION [dbo].[GET_PARTNER]
(
       @PartnerID int

)
RETURNS nvarchar(200)
AS
BEGIN
DECLARE @Naziv nvarchar(200)

       SELECT @Naziv = NAZIV
       FROM PARTNER
       WHERE ID = @PartnerID;

       RETURN @Naziv;
END
```



Slika 13. Prikaz izgleda formi klikom na gumb Izmijeni.

Nadalje, klikom na gumb „Brisanje“ javlja se poruka koja traži potvrdu (slika 14).

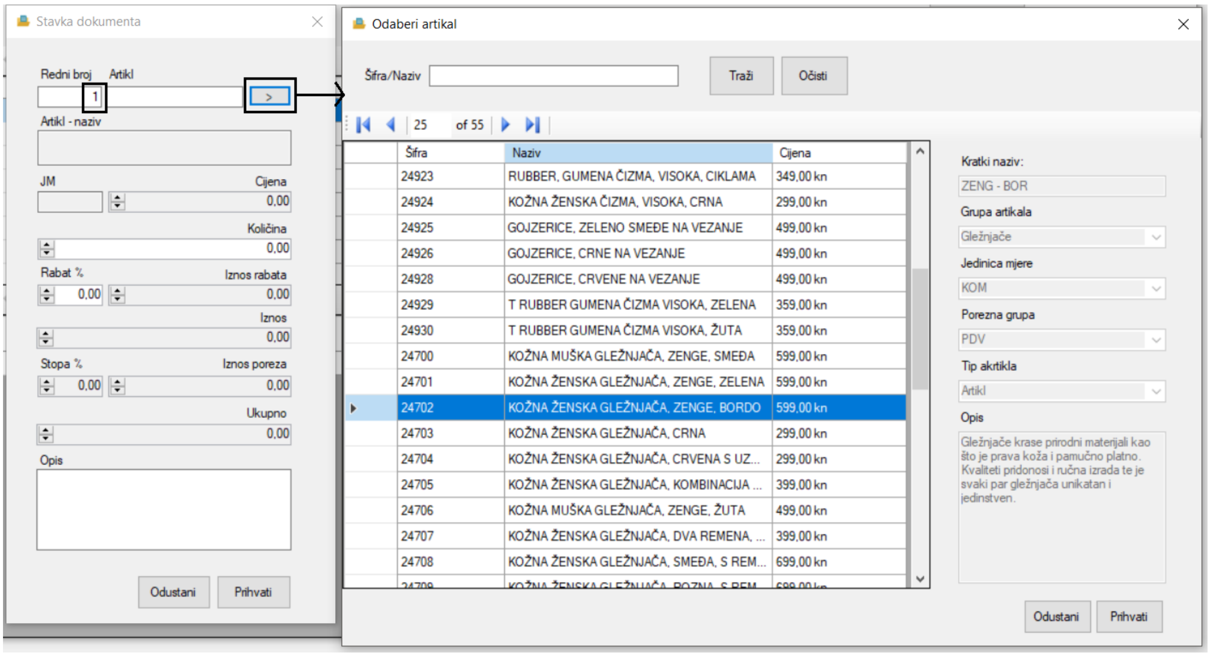


Slika 14. Prikaz brisanja dokumenta.

8.5.2. Pregled stavki

Stavke računa vidljive su u donjoj ploči na prikazu računa. Iz tablice DOKUMENT_STAVKA je povučen *DataGridView* i s time smo dobili uvid u: redni broj dokumenta, artikl, opis artikla, jedinicu mjere, količinu, cijenu i sveukupan iznos. Dodan je *BindingNavigator* koji omogućuje navigaciju po stavkama artikla. Klikom na gumb „Dodaj“ otvara se forma Stavka dokumenta (slika 15). Broj je dodan automatski i ovisi o broju prošlog dokumenta (uvećan za 1), a za to je kreirana funkcija NEXT_STAVKA:

```
CREATE   FUNCTION [dbo].[NEXT_STAVKA]
(
       @DokumentaID int
)
RETURNS int
AS
BEGIN
    DECLARE @Broj int;
       DECLARE @RET int;

       select @Broj = max(d.RBR)
       from DOKUMENT_STAVKA d
    where d.DOKUMENT_ID = @DokumentaID;

       SET @RET = ISNULL(@Broj,0) + 1;

       RETURN @RET;
END
```

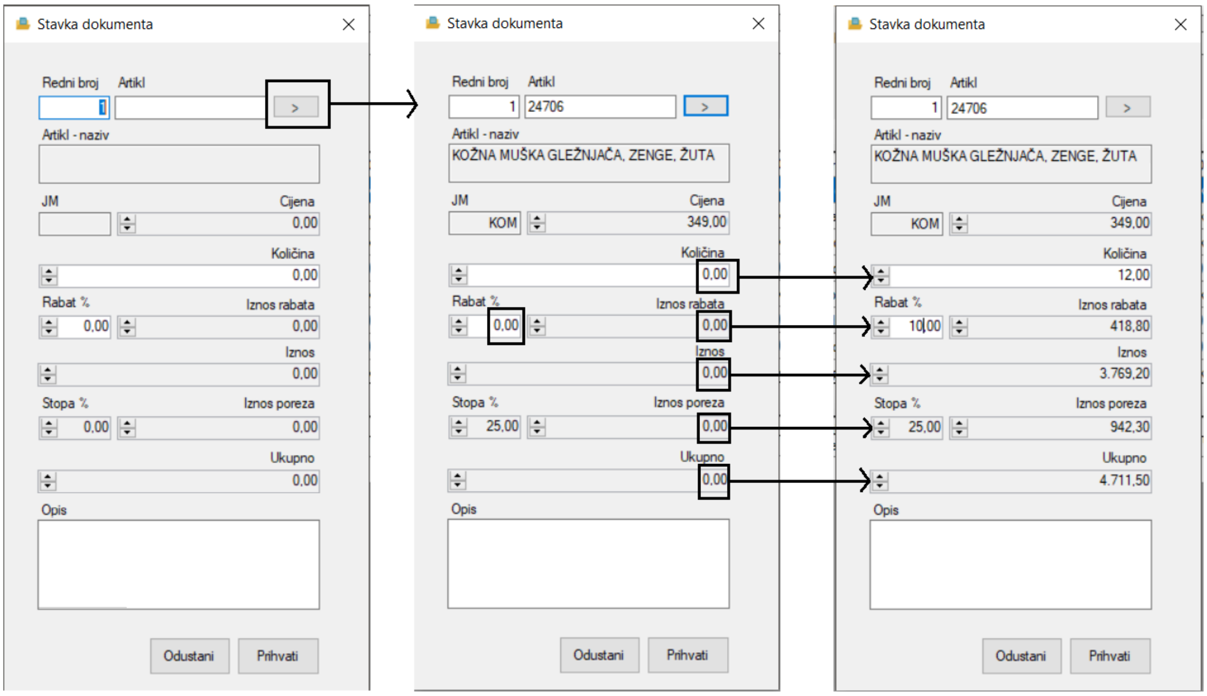


Slika 15. Prikaz izgleda formi klikom na gumb „Dodaj“ .

Klikom na gumb „Artikl“ otvara se forma koja omogućuje Odabir artikla. Odabirom artikla automatski se popunjavaju podaci vezani uz artikl (slika 16). Navedeno je omogućeno pomoću metode Izračunaj() koja računa iznos (Prilog 6):

```
private void Izracunaj()
{
    //početne vrijednosti pomoćnih varijabli stavljene su na 0
    decimal kolicina = 0;
    decimal cijena = 0;
    decimal rabat = 0;
    decimal rabat_iznos = 0;
    decimal porez = 0;
```

```
    decimal porez_iznos = 0;
    decimal iznos = 0;
    decimal ukupno = 0;
    try
    {
        kolicina = nbKolicina.Value;
        cijena = nbCijena.Value;
        rabat = nbRabat.Value;
        porez = nbPorez.Value;

        decimal ms = kolicina * cijena;
        rabat_iznos = (rabat / 100) * ms;
        iznos = ms - rabat_iznos;
        decimal pi = 1 + (porez / 100);
        ukupno = (pi * iznos);
        porez_iznos = ukupno - iznos;
    }
    catch (System.Exception)
    {
        rabat_iznos = 0;
        porez_iznos = 0;
        iznos = 0;
        ukupno = 0;
    }
    nbIznosRabata.Value = rabat_iznos;
    nbIznosPoreza.Value = porez_iznos;
    nbIznos.Value = iznos;
    nbUkupno.Value = ukupno;
}
```

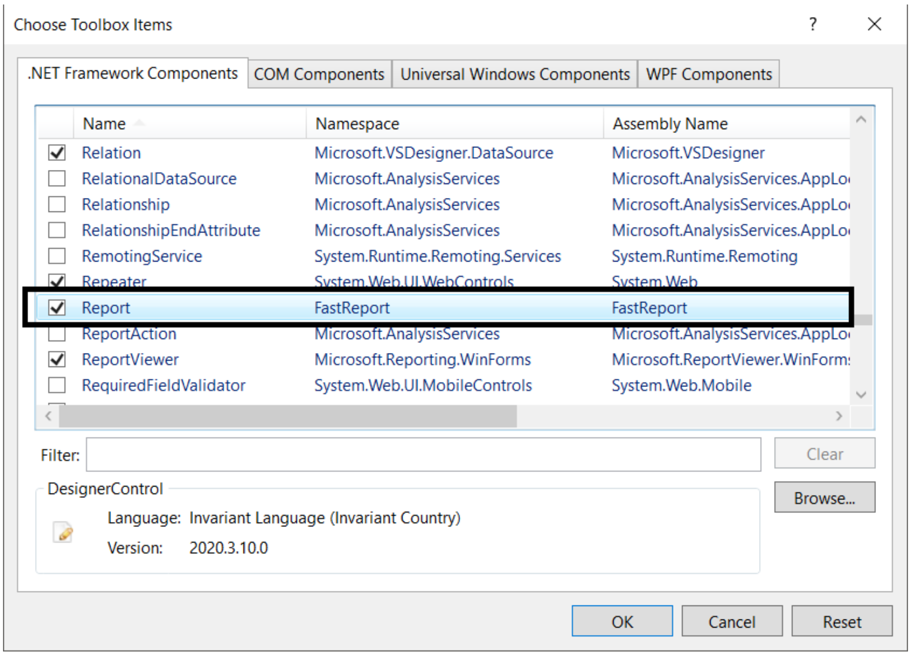


Slika 16. Prikaz automatskog popunjavanja vrijednosti odabirom artikla.

8.6. Izvještaji

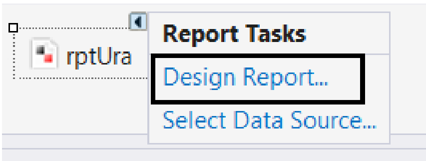
Za izradu izvještaja koristio se alat za generiranje izvještaja *.NET Fast Reports*. Na službenoj stranici skinuta je najnovija verzija[12]. Izvještajni podsustav omogućuje korisniku da pošalje upit bazi podataka sustava i da distribuira izvještaje (Cheng, 1985). Aplikacija tako izrađuje tradicionalno izvješće u obliku obrasca, a podaci se automatski pune u zato predviđene forme. Nadalje, u Visual Studiju u komponenti *Toolbox – General – Choose Items – Browse* odabire se datoteka Fast.Report.dll dostupna unutar instalirane FastReport datoteke. Nakon toga dobije se sljedeći zaslon (slika 17):

[12] Fast Reports, https://www.fast-report.com/en/download/fast-report-net/



Slika 17. Prikaz FastReport komponente unutar *Toolbox Itemsa*.

Na formi glavnog zaslona dodaju se sljedeće komponente iz *Toolboxa*: *DataDataSet* i *Report*. Kako bi se izvještaj mogao kreirati (i kasnije uređivati) potrebno je na novo dodanoj komponenti *Report* odabrati *Design report*… (slika 18):



Slika 18. Prikaz svojstva Design Report.

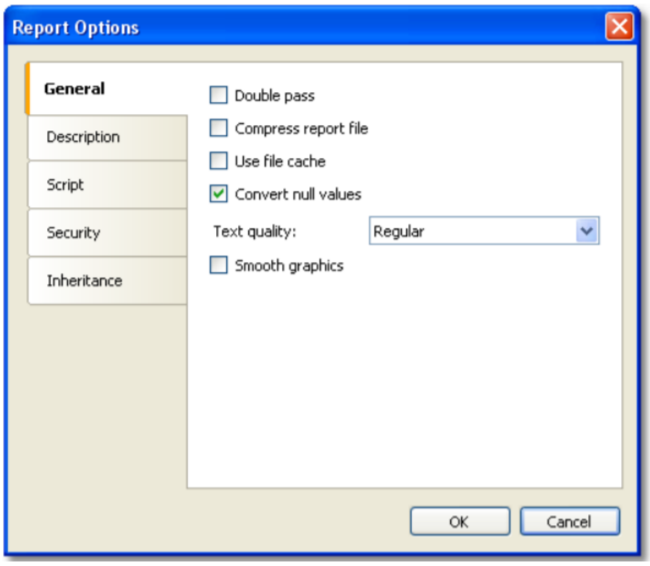
Nadalje se otvara pomoć pri kreiranju izvješća. Na kartici „Općenito" mogu se kontrolirati sljedeći parametri izvješća[13] (slika 19):

- parametar „Double pass" omogućuje prikazivanje dva izvješća što može biti potrebno kada se koristi sistemska varijabla „ukupan broj stranica";
- parametar „Stisni datoteku izvješća" omogućuje spremanje izvješća u komprimiranom obliku za što se upotrebljava zip algoritam;
- parametar „Upotreba pred memorije datoteke" omogućava uštedu memorije pri stvaranju izvještaja (koristi se kad izvješće ima puno stranica);
- kontrole „Pretvori nulte vrijednosti" pretvaraju stupac podataka s null vrijednosti u zadanu vrijednost (0, prazan niz, netočno – ovisno o vrsti podataka stupca);

[13] https://www.fast-report.com/public_download/FRNetUserManual-en.pdf

- parametar „Kvaliteta teksta“ omogućava odabir načina prikazivanja teksta u izvješću;
- parametar „Glatka grafika“ omogućava uključenje glatkog načina pri grafičkom crtanju predmeti (linija, granica, slika).

Ostale dostupne kartice nisu obavezne. Na kartici „Opis“ može se dodati opis izvješća. Na kartici „Skripta“ može se dodati jezik skripte za izvješće (podržava hrvatski jezik). Na kartici „Sigurnost“ može se unijeti lozinka koja će biti zatražena prilikom otvaranja izvješća (izvještaj sa zaporkom sprema se u kodiranom obliku i vraćanje bez zaporke nije moguće).



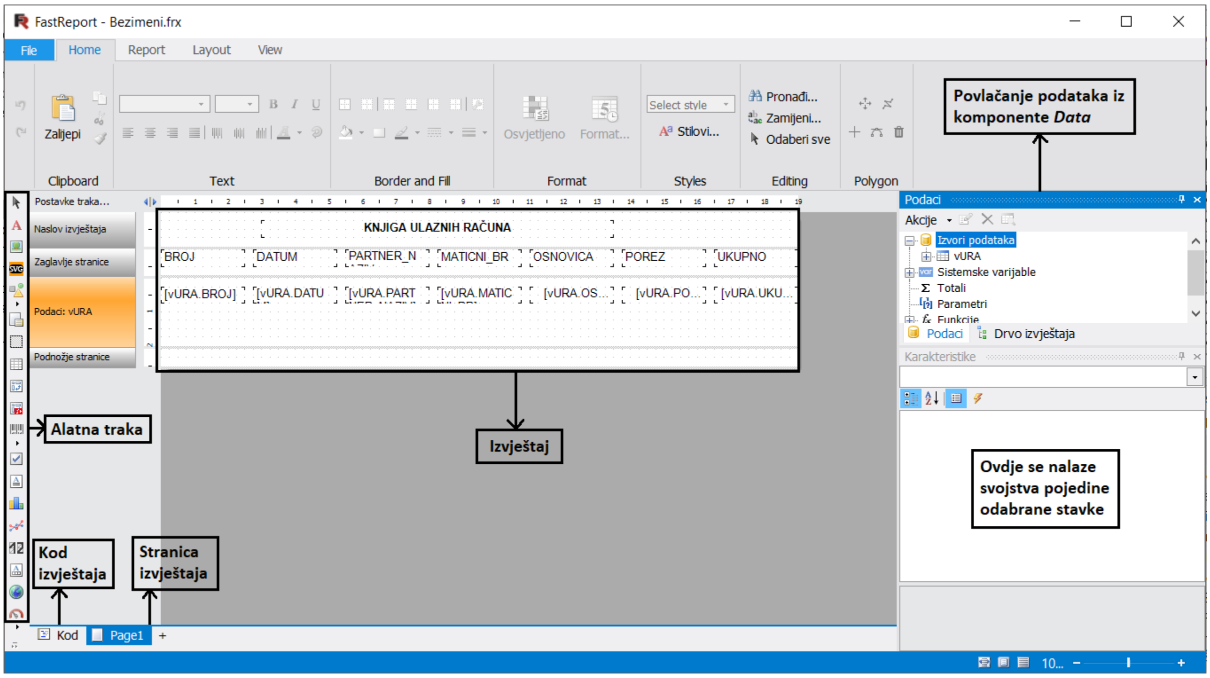
Slika 19. Prikaz kartica svojstava izvješća[14]

Sljedeće je potrebno povezati izvještaj s podacima iz baze. Za to se koristi metoda `rptUra.Show();`. Izvještaj se kreira odabirom tablica koje su potrebne za prikaz. Za potrebe izrade knjige ulaznih računa (rptUra) kreiran je pogled:

```
CREATE VIEW vURA AS
SELECT
D.BROJ AS BROJ,
D.DATUM AS DATUM,
P.NAZIV AS PARTNER_NAZIV,
P.MATICNI_BR ,
D.OSNOVICA ,
D.POREZ ,
D.UKUPNO
FROM
DOKUMENT D,
PARTNER P,
VRSTA_DOKUMENTA VD
WHERE D.PARTNER_ID = P.ID
AND D.VRSTA_DOKUMENTA_ID = VD.ID
AND VD.SIFRA = 'URA'
```

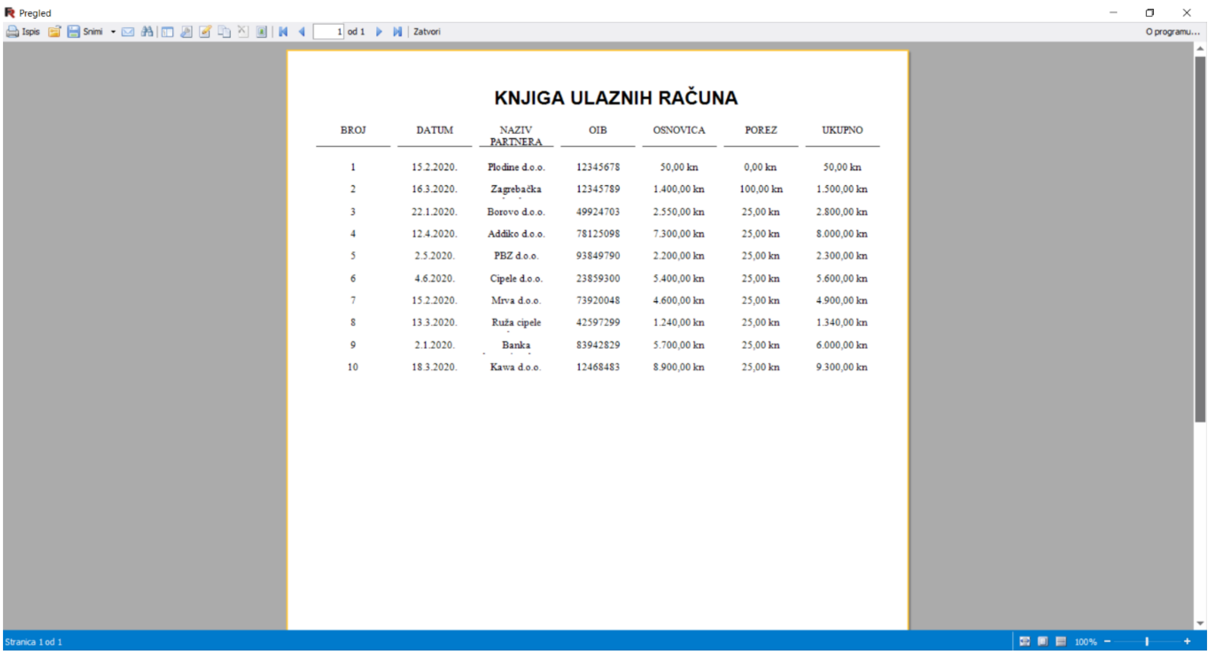
Nakon toga otvara se *Report Designer* (slika 20).

[14] https://www.fast-report.com/public_download/FRNetUserManual-en.pdf



Slika 20. Prikaz *Report designera*.

Nakon zatvaranja prozora izvještaj se automatski sprema. Nakon klika na određeni izvještaj (pod Izvještaji u aplikaciji) otvara se pregled izvještaja (slika 21)

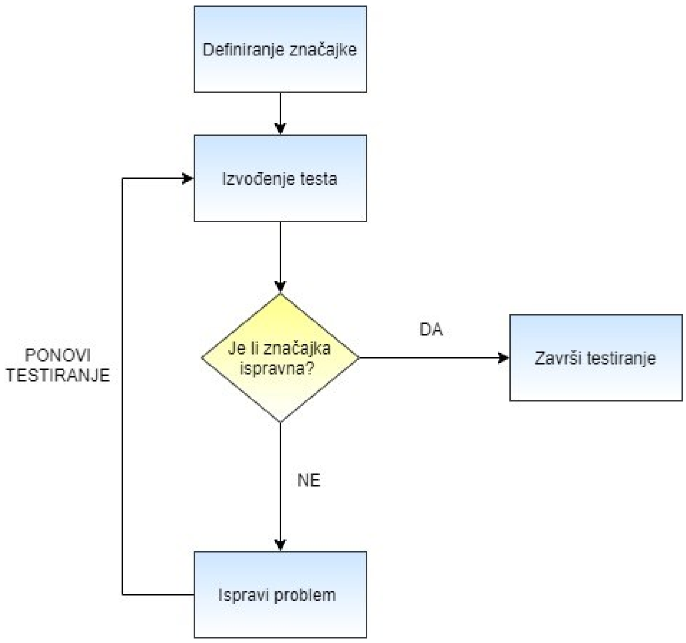


Slika 21. Prikaz izvještaja.

9. Testiranje i održavanje

9.1. Testiranje

Cilj testiranja je uvidjeti da program radi ono što treba i otkriti nedostatke programa, prije nego se isti stavi u upotrebu. Prema Sommerville (2016) postoje dvije vrste testiranja: validacijsko testiranje i traženje nedostataka. Validacijsko testiranje ili testiranje valjanosti je provjeravanje hoće li sustav nastupiti pravilno prilikom izvršenja određenih akcija, odnosno, hoće li pravilno odgovoriti na korisnički zahtjev. Testiranje nedostataka je različita vrsta testiranja od validacijskog i fokusira se na mane sustava, drugim riječima, kvarove (engl. *bug*) i popravljanje tih kvarova (engl. *bugfix*). Sommerville (2016) tvrdi da se oba načina testiranja mogu vrlo lako ispreplesti te je korisno raditi kombinaciju oba za kvalitetno i uspješno testiranje. Na slici 22. nalazi se prikaz načina po kojima će se voditi testiranje funkcionalnosti aplikacije.



Slika 22. Prikaz načela testiranja značajki aplikacije.

Kako bi se dao uvid u to jesu li sve značajke aplikacije koje je korisnik tražio u skladu s izgledom i funkcijom aplikacije, izraditi će se tablica za testiranje (Tablica 1). Lijevi stupac tablice sadrži popis značajki aplikacije dok se desni stupac tablice odnosi na ispravnost (+) ili neispravnost (-) pojedine značajke.

Tablica 1. Prikaz testiranja značajki aplikacije .

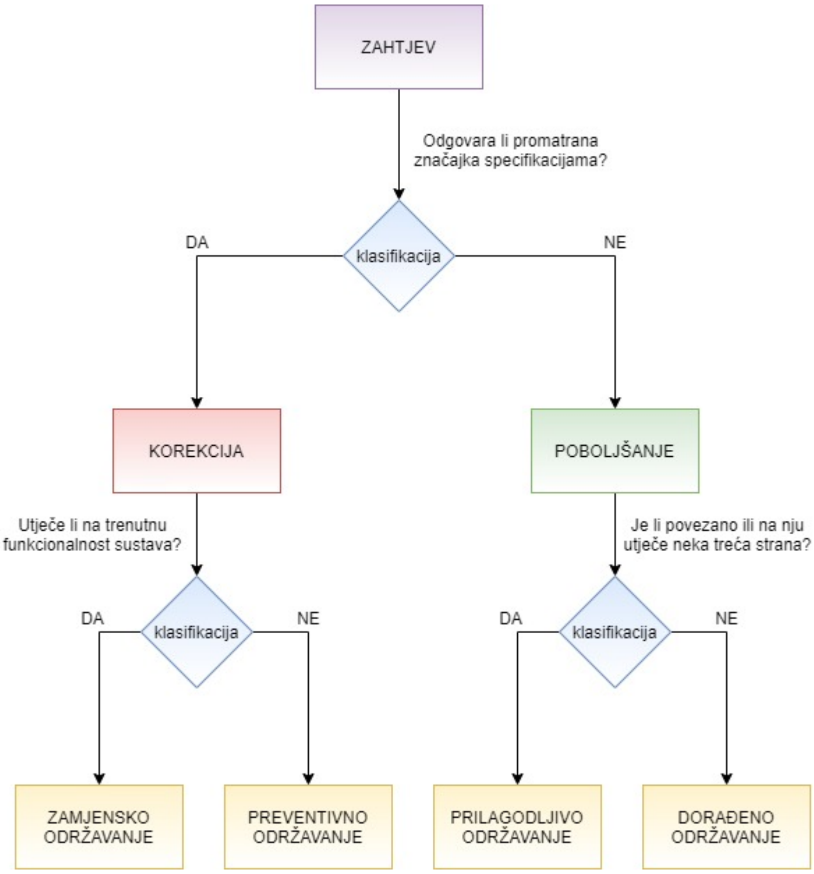
FORMA	ZNAČAJKA /FUNKCIONALNOST	ISPRAVNOST
Ekran prijave	Ikona	+
	Spojeno s bazom	+
	Provjera podataka	+
	Poruka greške	+
	Naziv forme	+
Glavni ekran	Ikona	+
	Otvaranje forme klikom na ikonu Korisnik	+
	Otvaranje forme klikom na ikonu Artikl	+
	Otvaranje forme klikom na ikonu Partner	+
	Otvaranje forme klikom na ikonu Izvještaj	+
	Izlaz iz aplikacije klikom na Izlaz	+
	Otvaranje svih formi unutar polja Datotetka	+
	Otvaranje svih formi unutar polja Matični podaci	+
	Otvaranje svih formi unutar polja Ulazni dokumenti	+
	Otvaranje svih formi unutar polja Izlazni dokumenti	+
	Otvaranje svih formi unutar polja Izvještaji	+
	Otvaranje svih formi unutar polja Pomoć	+
	Odgovarajući prijavljeni korisnik	+
Forme Pregled: korisnika, uloga, uloga korisnika, artikala, matičnih podataka, ulaznih i izlazih	Ikona	+
	Naziv forme	+
	Centrirano	+
	Prikaz svih podataka	+

dokumenata, izvještaja i pomoći	Omogućeni padajući izbornici	+
	Povezano s bazom	+
	Omogućeno brisanje	+
	Omogućeno dodavanje	+
Pregled dokumenata (narudžbenica, ulazni račun, ponude, račun otpremnica)	Ikona	+
	Gumb Traži	+
	Gumb Počisti	+
	Tekstualni okvir Broj	+
	Tekstualni okvir Godina	+
	Gumb Partneri	+
	Gumb Izradio	+
	Gumb Primio	+
	Gumb Izdao	+
	Automatsko popunjavanje forme (desno)	+
	Alatna traka za CRUD Dokumenata	+
	Odabirom računa popunjavaju se podaci u formi (desno)	+
	Alatna traka za CRUD Stavki Artikla	+
Dodavanje dokumenta	Ikona	+
	Automatsko popunjavanje Broja dokumenta	+
	Tab prebacuje na sljedeće polje	+
	Gumb Partner	+
	Automatsko punjenje partnera odabirom na formi Pretraživanje partnera	+
	Padajući izbornik odabira načina plaćanja	+
	Padajući izbornik odabira poslovna jedinica	+
	Klikom na Prihvati zatvara se forma	+

	Klikom na Prihvati automatski se popunjavaju podaci u formu (desno)	+
	Klikom na Odustani izlazi se iz forme	+
	Gumb Dodaj	+
	Gumb Izmijeni	+
	Gumb Brisanje	+
Dodavanje stavke artikla	Ikona	+
	Automatsko punjenje rednog broja	+
	Gumb Artikl otvara formu	+
	Klikom na Prihvati povlače se podaci o artiklu	+
	Klikom na Odustani izlazi se iz forme	+
	Padajući izbornik Poslovna jedinica	+
	Padajući izbornik Način plaćanja	+
	Unosom podataka automatski se ažuriraju drugi podaci	+

9.2. Održavanje

Održavanje softvera bavi se izmjenom softvera nakon isporuke i ulaska u službu korisnika (Bennet et al., 2002). Faza održavanja počinje nakon što je softver pozitivno prošao sva testiranja i uručen je korisniku na korištenje. Iako zbog potrebnog opsega ovog rada autor neće ulaziti u detalje procesa održavanja softvera, bitno je istaknuti navedenu fazu procesa zbog niza istraživanja koja su potvrdila da je održavanje softvera jedan od ključnih elemenata životnog ciklusa softvera i stoga je održavanje od ogromne industrijske i komercijalne važnosti (Bennet et al., 2002). Prema IEEE standardima (2006) postoje četiri glavne kategorije za održavanje softvera: zamjensko, preventivno, prilagodljivo i dorađeno (slika 23). Uloga zamjenskog održavanja je provesti promjene u sustavu kako bi ga prilagodio izvornim zahtjevima i specifikacijama (Michas, 2020). Preventivno održavanje je tip održavanja koji se obavlja kako bi se spriječila greška. Prilagodljivo održavanje podrazumijeva sve promjene u sustavu koje nisu bile dio njegovih originalnih zahtjeva i specifikacija, dok s druge strane dorađeno održavanje uključuje sve novo dodatne značajke koje su utvrđene kao dio novonastalih zahtjeva korisnika (Michas, 2020).



Slika 23. Identificiranje vrste održavanja potrebne za obradu zahtjeva za modifikacijom (Michas, 2020).

10. Zaključak

U ovom radu objašnjen je životni ciklus razvoja poslovne aplikacije. Iako je pojam razvoja aplikacije širok, temeljni principi razvoja bilo koje vrste aplikacije su jednaki. Ciklus se tako sastoji od procesa analize korisničkih zahtjeva, planiranja aplikacije (dizajna korisničkog sučelja, značajki, funkcionalnosti), implementacije (programiranja), testiranja i održavanja. Obrazloženo je da se zahtjev klijenta odnosi na sve usmene i pismene popise funkcionalnosti i značajke koje aplikacija mora imati. Naglašeno je da korisnici nisu nužno osobe tehničke pozadine i da je iterirani razgovor s korisnikom i visoko razumijevanje njegovih potreba i želja jedna od glavnih polaznih točaka pri izradi bilo koje vrste aplikacije. Na temelju korisnikovih želja kreira se specifikacija koja objašnjava kako aplikacija treba izgledati, koje će biti njene funkcionalnosti, treba li biti složena ili jednostavna i drugo. Opisana je izrada baze podataka i objašnjen model podataka. Na temelju svih potrebnih i ranije obrazloženih značajki opisana je izrada aplikacije koristeći C#. Objašnjena je svaka stavka aplikacije i priložen kod koji stoji iza pojedine funkcionalnosti. Ukratko se opisala i sljedeća stavka razvoja, testiranje i održavanje, zbog potrebe zaokruživanja cijelog ciklusa izrade aplikacije.

11. Literatura

1. Beciric, Dejan. (2017). *Funkcionalna specifikacija softvera, detaljno*. DM Spot portal. Pristupljeno 9. lipnja 2020, s https://www.majkic.net/novosti/nauka-i-tehnologija/813-funkcionalna-specifikacija-softvera-detaljno
2. Bennet, K. H., Rajlich, V.T., Wilde, N. (2002). *Software Evolution and the Staged Model of the Software Lifecycle*. Advances in Computers 56, 1 – 54. Pristupljeno 25. lipnja 2020., s http://www.sciencedirect.com/science/article/pii/S0065245802800031
3. Blaće, Dubravko. (2015). *Što su poslovne aplikacije?* EVision informacijski sustavi. Pristupljeno 13. lipnja 2020., s https://www.evision.hr/hr/Novosti/Stranice/sto-su-poslovne-aplikacije.aspx
4. Cheng, Wei-Tih. (1985). *An integrated business application architecture.* Proceedings of the international conference on APL: APL and the future (str. 52- 58). Seattle: Association for Computing Machinery. Pristupljeno 10. lipnja 2020., s https://doi.org/10.1145/17701.255339
5. Clark, Dan. (2013). *Beginning C# Object-Oriented Programming*. Apress.
6. Cooper, Zara. (2020.) *A practical guide to writing technical specs*. Stack Overflow Blog. Pristupljeno 10. lipnja 2020., s https://stackoverflow.blog/2020/04/06/a-practical-guide-to-writing-technical-specs/
7. Essay Sauce. (2019). *Main Characteristics And Features Of Object Oriented Programming*. Pristupljeno 10. lipnja 2020., s https://www.essaysauce.com/computer-science-essays/main-characteristics-and-features-of-object-oriented-programming/
8. Fast Reports. (2020). Pristupljeno 2. srpnja 2020., s https://www.fast-report.com/en/download/fast-report-net/
9. FastReport.Net User's Manual. (2020). Pristupljeno 2. srpnja 2020., s https://www.fast-report.com/public_download/FRNetUserManual-en.pdf
10. GNU. *Stranica GNU tima za hrvatske prijevode*. Pristupljeno 25. svibnja 2020., s https://www.gnu.org/server/standards/translations/hr/?fbclid=IwAR2MBmA4vQatI24vJg_uXJVNeiYPAZo7oFUUe8_XBNITF8lojp-DmNdwsUo
11. Hecksel, David. (2004). *System and method for software methodology evaluation and selection.* Sun Microsystems Inc. Pristupljeno 12. lipnja 2020., s https://patents.google.com/patent/US20040243968A1/en
12. IADC. (2013). *Technical Specification*. IADC Lexicon. Pristupljeno 14. lipnja 2020., s https://www.iadclexicon.org/technical-specification/

13. IEEE. (2006). *14764-2006 - ISO/IEC/IEEE International Standard for Software Engineering - Software Life Cycle Processes – Maintenance*. Pristupljeno 25. lipnja 2020., s https://standards.ieee.org/content/ieee-standards/en/standard/14764-2006.html
14. IBM. (2014). *Business Applications*. Pristupljeno 13. lipnja 2020., s www.ibm.com/support/knowledgecenter/en/ssplfc_7.3.0/com.ibm.taddm.doc_7.3/userguide/c_cmdb_business_apps.html
15. Levenson, Nancy G. (1996). *The Role of Software in Spacecraft Accidents*. Pristupljeno 15. lipnja 2020., s http://sunnyday.mit.edu/papers/jsr.pdf
16. McGuire, Nate. (2016). *How to Build a Software Specification Document*. San Francisco: Top Digital Agency. Pristupljeno 13. lipnja 2020., s https://mayvenstudios.com/blog/how-to-build-a-software-specification-document
17. Michas, Nassos. (2020). *How to Plan for Software Maintenance.* Medium. Pristupljeno 25. lipnja 2020., s https://medium.com/swlh/types-of-software-maintenance-2b0503848b43
18. Microsoft Docs. (2008). *General Naming Conventions - Framework Design Guidelines.* Pristupljeno 20. lipnja 2020., s https://docs.microsoft.com/en-us/dotnet/standard/design-guidelines/general-naming-conventions
19. Microsoft Docs. (2015). *Introduction to the C# Language and the .NET Framework*. Pristupljeno 22. lipnja 2020., s https://docs.microsoft.com/en-us/dotnet/csharp/getting-started/introduction-to-the-csharp-language-and-the-net-framework
20. Microsoft Docs. (2016). Fill datasets by using TableAdapters. Pristupljeno, 28. lipnja 2020., s https://docs.microsoft.com/en-us/visualstudio/data-tools/fill-datasets-by-using-tableadapters?view=vs-2019
21. Mierriam – Webster dictionary. *Software*. Pristupljeno 19. lipnja 2020., s https://www.merriam-webster.com/dictionary/software
22. Rouse, Margaret. (2007). *Functional specification. Search Software Quality @* Tech Target Network. Pristupljeno 24. lipnja 2020., s https://searchsoftwarequality.techtarget.com/definition/functional-specification
23. Sommerville, Ian. (2011). *Software engineering*. Boston: Pearson.
24. Spolsky, Joel. (2000). *Painless Functional Specifications – Part 1: Why Bother?* Joel on Software. Pristupljeno 22. lipnja 2020., s https://www.joelonsoftware.com/2000/10/02/painless-functional-specifications-part-1-why-bother/

25. Hrvatska enciklopedija. *Aplikacija*. Leksikografski zavod Miroslav Krleža. Pristupljeno 22. lipnja 2020., s https://www.enciklopedija.hr/Natuknica.aspx?ID=3306

Popis slika

Popis tablica

Prilozi

Prilog 1 – Kod forme prijave (frmLogin)

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

namespace Trgovina
{
    public partial class frmLogin : Form
    {

        private int fID;

        public frmLogin()
        {
            InitializeComponent();

        }

        //metoda koja prikazuje formu
        public int Prikazi()
        {

            fID = 0;

            var res = ShowDialog();

            return fID;
        }

        // metoda koja omogućuje spajanje na bazu i poziva funkciju USER_LOGIN pomoću
koje se provjerava podudaranost unešenih i podataka u bazi
        private string SpojiSeNaBazu()
        {

            using (SqlConnection conn = new
SqlConnection(Properties.Settings.Default.DataConnectionString))
            using (SqlCommand cmd = new SqlCommand("SELECT dbo.USER_LOGIN('" +
tbUserName.Text +"','"+ tbPassword.Text + "')", conn))
            {
                conn.Open();
                var result = cmd.ExecuteScalar();
                conn.Close();

                return result.ToString();
            }

        }

        // metoda koja omogućuje spajanje na bazu i provjeru podataka koji je korisnik
unio
```

```
        // ukoliko je ID (vID) korisnika null ili prazan pojavljuje se poruka "Prijava
nije uspjela!"
        // ukoliko se podaci koje je korisnik unio slažu s onima u bazi, prozor
prijave se zatvara
        private void btnOK_Click(object sender, EventArgs e)
        {
            string vID = SpojiSeNaBazu();
            if (string.IsNullOrEmpty(vID))
            {
                //ukoliko se podaci koje je korisnik unio NE slažu s onima u bazi,
javlja se poruka greške

                MessageBox.Show("Prijava nije uspjela!", "Trgovina");
                this.DialogResult = DialogResult.None;
                tbPassword.Text = "";

            }
            else
            {
                fID = Int32.Parse(vID);
                this.DialogResult = DialogResult.OK;
                Close();
            }
        }

        private void frmLogin_Shown(object sender, EventArgs e)
        {
            /*
            // za testiranje
            tbUserName.Text = "ADMIN";
            tbPassword.Text = "a";
            */
        }

        private void frmLogin_Load(object sender, EventArgs e)
        {

        }

    }
}
```

Prilog 2 – Kod forme glavnog ekrana (frmMain)

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

namespace Trgovina
{
    public partial class frmMain : Form
    {
        public int UserID ;

        public frmMain()
        {
            InitializeComponent();
        }

        private void frmMain_Shown(object sender, EventArgs e)
        {

            // postavi veličinu tooltipa

            double sirina = (80  * this.Width ) / 100;
            int sirinaINT = Convert.ToInt32(sirina);

            toolStripStatusLabel1.Width = sirinaINT;
            toolStripStatusLabel2.Text = "Prijavljeni korisnik: ";
            toolStripStatusLabel2.AutoSize = true;

            // prijava na bazu
            PorukaToolTip("Prijava u aplikaciju ...");
            frmLogin flogin = new frmLogin();
            UserID = flogin.Prikazi();
            if (UserID == 0) { Application.Exit(); }
```

```
            PorukaToolTip("");
            IspisiKorisnikToolTip();

            //sakrij izbornik ako nije administrator
            SakrijMenu();
        }

        // metoda koja omogućuje spajanje na bazu i provjerava korisnički ID te sakriva izbornik
        private void SakrijMenu()
        {

            int admin = 0;
            using (SqlConnection conn = new SqlConnection(Properties.Settings.Default.DataConnectionString))
            using (SqlCommand cmd = new SqlCommand("SELECT dbo.IS_ADMIN('" + UserID.ToString() + "')", conn))
            {
                conn.Open();
                var result = cmd.ExecuteScalar();
                conn.Close();

                string s = result.ToString();

                if (!string.IsNullOrEmpty(s))
                {
                    admin = int.Parse(s);
                }
            }

           // MessageBox.Show(admin.ToString() ,"admin");

            if (admin != 1)
            {
                postavkeToolStripMenuItem.Visible = false;
                toolStripMenuItem1.Visible = false;
            }
        }

        // metoda koja ispisuje (u donjem desnom uglu) je li prijavljena osoba na aplikaciju Admin ili Korisnik
```

```
        private void IspisiKorisnikToolTip()
        {
            using (SqlConnection conn = new SqlConnection(Properties.Settings.Default.DataConnectionString))
            using (SqlCommand cmd = new SqlCommand("SELECT dbo.GET_USER('" + UserID.ToString() + "')", conn))
            {
                conn.Open();
                var result = cmd.ExecuteScalar();
                conn.Close();
                toolStripStatusLabel2.Text = "Prijavljeni korisnik: " + result.ToString();
            }
        }

        // metoda koja ispisuje poruku na ToolStrip
        public void PorukaToolTip(string poruka)
        {
            this.toolStripStatusLabel1.Text = poruka;
        }

        // klikom na polje 'Izlaz' aplikacija se zatvara
        private void izlazToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Application.Exit();
        }

        // klikom na polje 'Korisnik' otvara se forma frmPregledKorisnika
        private void korisnikaToolStripMenuItem_Click(object sender, EventArgs e)
        {
            frmPregledKorisnika fForma = new frmPregledKorisnika();
            fForma.ShowDialog();
        }

        // klikom na polje 'Uloga korisnika' otvara se forma frmUlogaKorisnika
        private void korisniciUlogeToolStripMenuItem_Click(object sender, EventArgs e)
        {
            frmPregledUlogaKorisnika fForma = new frmPregledUlogaKorisnika();
            fForma.ShowDialog();
        }

        // klikom na polje 'Pregled Uloga' otvara se forma frmPregledUloga
```

```
        private void ulogeToolStripMenuItem_Click(object sender, EventArgs e)
        {
            frmPregledUloga fForma = new frmPregledUloga();
            fForma.ShowDialog();
        }

        // klikom na polje 'Vrste dokumenata' otvara se forma frmPregledVrsteDokumenata
        private void vrsteDokumenataToolStripMenuItem_Click(object sender, EventArgs e)
        {
            frmPregledVrsteDokumenata fForma = new frmPregledVrsteDokumenata();
            fForma.ShowDialog();
        }

        // klikom na polje 'Poslovne jedinice' otvara se forma frmPregledPoslovnihJedinica
        private void poslovneJediniceToolStripMenuItem_Click(object sender, EventArgs e)
        {
            frmPregledPoslovnihJedinica fForma = new frmPregledPoslovnihJedinica();
            fForma.ShowDialog();
        }

        // klikom na polje 'Države' otvara se forma frmDrzava
        private void drzaveToolStripMenuItem_Click(object sender, EventArgs e)
        {
            frmPregledDrzava fForma = new frmPregledDrzava();
            fForma.ShowDialog();
        }

        // klikom na polje 'Grupe artikala' otvara se forma frmPregledGrupeArtikala
        private void grupeArtikalaToolStripMenuItem_Click(object sender, EventArgs e)
        {
            frmPregledGrupeArtikala fForma = new frmPregledGrupeArtikala();
            fForma.ShowDialog();
        }

        // klikom na polje 'Jedinice mjere' otvara se forma frmPregledJedinicaMjere
        private void jediniceMjereToolStripMenuItem_Click(object sender, EventArgs e)
        {
            frmPregledJedinicaMjere fForma = new frmPregledJedinicaMjere();
            fForma.ShowDialog();
```

```
}

// klikom na polje 'Mjesta' otvara se forma frmPregledMjesta
private void mjestaToolStripMenuItem_Click(object sender, EventArgs e)
{
    frmPregledMjesta fForma = new frmPregledMjesta();
    fForma.ShowDialog();
}

// klikom na polje 'Nacin placanja' otvara se forma frmPregledNacinaPlacanja
private void načinPlaćanjaToolStripMenuItem_Click(object sender, EventArgs e)
{
    frmPregledNacinaPlacanja fForma = new frmPregledNacinaPlacanja();
    fForma.ShowDialog();
}

// klikom na polje 'Porezne grupe' otvara se forma frmPregledPoreznihGrupa
// ona sadrži pregled poreznih grupa i poreznih stopa
private void porezneGrupeToolStripMenuItem_Click(object sender, EventArgs e)
{
    frmPregledPoreznihGrupa fForma = new frmPregledPoreznihGrupa();
    fForma.ShowDialog();
}

// klikom na polje 'Artikli' otvara se forma frmPregledArtikala
private void artikliToolStripMenuItem_Click(object sender, EventArgs e)
{
    frmPregledArtikala fForma = new frmPregledArtikala();
    fForma.ShowDialog();
}

// klikom na polje 'Partneri' otvara se forma frmPregledPartnera
private void partneriToolStripMenuItem_Click(object sender, EventArgs e)
{
    frmPregledPartnera fForma = new frmPregledPartnera();
    fForma.ShowDialog();
}

// klikom na polje 'Narudžbenice' otvara se forma frmPregledDokumenata
private void narudžbeniceToolStripMenuItem_Click(object sender, EventArgs e)
{
```

```
    frmPregledDokumenata fForma = new frmPregledDokumenata("NAR", UserID);
    fForma.ShowDialog();
}

// klikom na polje 'Ulazni računi' otvara se forma frmPregledDokumenata
private void ulazniRačuniToolStripMenuItem_Click(object sender, EventArgs e)
{
    frmPregledDokumenata fForma = new frmPregledDokumenata("URA", UserID);
    fForma.ShowDialog();
}

// klikom na polje 'Ponude' otvara se forma frmPregledDokumenata
private void ponudeToolStripMenuItem_Click(object sender, EventArgs e)
{
    frmPregledDokumenata fForma = new frmPregledDokumenata("PON", UserID);
    fForma.ShowDialog();
}

// klikom na polje 'Otpremnica' otvara se forma frmPregledDokumenata
private void računOtpremnicaToolStripMenuItem_Click(object sender, EventArgs e)
{
    frmPregledDokumenata fForma = new frmPregledDokumenata("IRA", UserID);
    fForma.ShowDialog();
}

private void frmMain_Load(object sender, EventArgs e)
{


}

// klikom na ikonu (gumb) Artikl otvara se forma frmPregledArtikala
private void toolStripButton6_Click(object sender, EventArgs e)
{
    frmPregledArtikala fForma = new frmPregledArtikala();
    fForma.ShowDialog();
}


// klikom na ikonu (gumb) Partneri otvara se forma frmPregledPartnera
private void toolStripButton2_Click_1(object sender, EventArgs e)
{
```

```
    frmPregledPartnera fForma = new frmPregledPartnera();
    fForma.ShowDialog();
}

// klikom na polje 'Upute za upotrebu' otvara se forma frmPregledUputa
private void uputeZaUpotrebuToolStripMenuItem_Click(object sender, EventArgs e)
{
    frmPregledUputa fForma = new frmPregledUputa();
    fForma.ShowDialog();
}

// klikom na polje 'O aplikaciji' otvara se forma frmPregledAplikacije
private void oAplikacijiToolStripMenuItem_Click(object sender, EventArgs e)
{
    frmPregledAplikacije fForma = new frmPregledAplikacije();
    fForma.ShowDialog();
}

// klikom na ikonu Ulazni račun otvara se forma Pregled dokumenata
private void toolStripButton3_Click(object sender, EventArgs e)
{
    frmPregledDokumenata fForma = new frmPregledDokumenata("URA", UserID);
    fForma.ShowDialog();
}

//klikom na gumb Narudzbenica otvara se forma Pregled dokumenata
private void btnNarudzbenica_Click(object sender, EventArgs e)
{
    frmPregledDokumenata fForma = new frmPregledDokumenata("NAR", UserID);
    fForma.ShowDialog();
}

//klikom na gumb Ponuda otvara se forma Pregled dokumenata
private void btnPonuda_Click(object sender, EventArgs e)
{
    frmPregledDokumenata fForma = new frmPregledDokumenata("PON", UserID);
    fForma.ShowDialog();
}

//klikom na gumb Racun otvara se forma Pregled dokumenata
private void btnRacun_Click(object sender, EventArgs e)
```

```
        {
            frmPregledDokumenata fForma = new frmPregledDokumenata("IRA", UserID);
            fForma.ShowDialog();
        }

        private    void    knjigaUlaznihRačunaToolStripMenuItem_Click(object    sender,
EventArgs e)
        {
            rptUra.Show();
        }

        private    void    knjigaUlaznihRačunaToolStripMenuItem1_Click(object    sender,
EventArgs e)
        {
            rptUra.Show();
        }

        private    void    knjigaIzlaznihRačunaToolStripMenuItem_Click(object    sender,
EventArgs e)
        {
            rptIra.Show();
        }

        private    void    knjigaIzlaznihRačunaToolStripMenuItem1_Click(object    sender,
EventArgs e)
        {
            rptIra.Show();
        }

        private void stanjePartneraToolStripMenuItem_Click(object sender, EventArgs e)
        {
            rptStanje.Show();
        }
    }
}
```

Prilog 3 – Kod forme korisnici (frmPregledKorisnika)

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

namespace Trgovina
{
    public partial class frmPregledKorisnika : Form
    {
        public frmPregledKorisnika()
        {
            InitializeComponent();
        }

        private void kORISNIKBindingNavigatorSaveItem_Click(object sender, EventArgs 
e)
        {
            this.Validate();
            this.kORISNIKBindingSource.EndEdit();
            this.tableAdapterManager.UpdateAll(this.dataDataSet);

        }

        private void frmPregledKorisnika_Load(object sender, EventArgs e)
        {
            // TODO: This line of code loads data into the 'dataDataSet.KORISNIK' 
table. You can move, or remove it, as needed.
            this.kORISNIKTableAdapter.Fill(this.dataDataSet.KORISNIK);

        }
    }
}
```

Prilog 4 – Kod forme pregleda dokumenata (frmPregledDokumenata)

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

namespace Trgovina
{
    public partial class frmPregledDokumenata : Form
    {
        //definiranje varijabli
        private int FUserID;
        private int FVrsta_Dokumenta_ID;
        private string FVrsta_Dokumenta_NAZIV = "";
        private int FPartnerID = -1;
        private int FKorisnikIzdaoID = -1;
        private int FKorisnikIzradioID = -1;
        private int FKorisnikPrimioID = -1;

        // definiranje polja partner, korisnik (izdao, primio, izradio) na vrijednost -1
        public frmPregledDokumenata()
        {
            InitializeComponent();
            FPartnerID = -1;
```

```
            FKorisnikIzdaoID = -1;
            FKorisnikIzradioID = -1;
            FKorisnikPrimioID = -1;
            FUserID = 0;
        }

        public frmPregledDokumenata(string pVrstaDokumenta, int pUserID)
        {
            InitializeComponent();
            // definiranje polja partner, korisnik (izdao, primio, izradio) na vrijednost -1
            FPartnerID = -1;
            FKorisnikIzdaoID = -1;
            FKorisnikIzradioID = -1;
            FKorisnikPrimioID = -1;
            FUserID = pUserID;

            using (SqlConnection conn = new
SqlConnection(Properties.Settings.Default.DataConnectionString))
            {
                //pozivanje funkcije koja vraća ID vrste dokumenta
                using (SqlCommand cmd = new SqlCommand("SELECT
dbo.GET_VRSTA_DOKUMENTA_ID('" + pVrstaDokumenta + "')", conn))
                {
                    conn.Open();
                    var result = cmd.ExecuteScalar();
                    conn.Close();
                    FVrsta_Dokumenta_ID = (int)result;
                }

                //pozivanje naziva unutar tablice VRSTA_DOKUMENTA za naziv forme
```

```
            using (SqlCommand cmd2 = new SqlCommand("SELECT NAZIV FROM
DBO.VRSTA_DOKUMENTA WHERE ID = " + FVrsta_Dokumenta_ID.ToString(), conn))
            {
                conn.Open();

                SqlDataReader reader = cmd2.ExecuteReader();

                if (reader.HasRows)
                {
                    while (reader.Read())
                    {
                        FVrsta_Dokumenta_NAZIV = reader.GetString(0);
                    }
                }
                else
                {
                    MessageBox.Show("Greška kod dohvata podataka o vrsti
dokumenta!","Trgovina");
                }
                reader.Close();
                conn.Close();
            }
        }

        //naziv forme
        this.Text = "Pregled - " + FVrsta_Dokumenta_NAZIV;
    }

    // dohvaća podatke za stavku dokumenta
    private void refreshStavke()
    {
```

```
            int dokumentID = 0;

            if (vRACUNDataGridView.Rows.Count > 0)
            {
                if
(!String.IsNullOrEmpty(Convert.ToString(vRACUNDataGridView.CurrentRow.Cells[0].Val
ue)))
                {

                    dokumentID = (int)vRACUNDataGridView.CurrentRow.Cells[0].Value;
                }
            }
            //filtriraj po ID-u dokumenta
            vStavkeBindingSource.Filter = "DOKUMENT_ID = " + dokumentID.ToString();
            this.vStavkeTableAdapter.Fill(this.dataDataSet.vStavke);
        }

        private void frmPregledDokumenata_Load(object sender, EventArgs e)
        {
            // TODO: This line of code loads data into the 'dataDataSet.DOKUMENT' table. You
can move, or remove it, as needed.
            this.dOKUMENTTableAdapter.Fill(this.dataDataSet.DOKUMENT);

            //podaci koji se popunjavaju nakon što se digne forma
            pDatumValuteOd.Value = new DateTime(DateTime.Now.Year, 1, 1);
            pDatumValuteDo.Value = DateTime.Now;
            vRACUNBindingSource.Filter = "VRSTA_DOKUMENTA_ID = " +
FVrsta_Dokumenta_ID.ToString();

            this.vRACUNTableAdapter.Fill(this.dataDataSet.vRACUN);
        }
```

```
        // metoda koja filtrira bazu podataka i traži unesene podatke
        private void Ucitaj()
        {
            string Filter = "VRSTA_DOKUMENTA_ID = " +
FVrsta_Dokumenta_ID.ToString();
            //slaganje filtera broj od
            // ! je znak negacije, znači ukoliko string NIJE null ili prazan
            if (!string.IsNullOrEmpty(tbBrojOd.Text))
            {
                Filter = Filter + " AND BROJ >= " + tbBrojOd.Text;
            }
            //slaganje filtera broj do
            if (!string.IsNullOrEmpty(tbBrojDo.Text))
            {
                Filter = Filter + " AND BROJ <= " + tbBrojDo.Text;
            }
            //slaganje filtera godina do
            if (!string.IsNullOrEmpty(tbGodinaDo.Text))
            {
                Filter = Filter + " AND GODINA <= " + tbGodinaDo.Text;
            }
            //slaganje filtera godina od
            if (!string.IsNullOrEmpty(tbGodinaOd.Text))
            {
                Filter = Filter + " AND GODINA >= " + tbGodinaOd.Text;
            }
            //slaganje filtera tbPartneri

            if (FPartnerID != -1)
            {
                Filter = Filter + " AND PARTNER_ID = " + FPartnerID.ToString();
```

```
            }


            //slaganje filtera datum valute od
            if (!string.IsNullOrEmpty(pDatumValuteOd.ToString()))
            {
                Filter = Filter + " AND DATUM_D >='" + 
pDatumValuteOd.Value.ToString("dd.MM.yyyy") + "'";
            }
            //slaganje filtera datum valute do
            if (!string.IsNullOrEmpty(pDatumValuteDo.ToString()))
            {
                Filter = Filter + " AND DATUM_D <= '" + 
pDatumValuteDo.Value.ToString("dd.MM.yyyy") + "'";
            }


            //slaganje filtera Izradio
            if (FKorisnikIzradioID != -1)
            {
                Filter = Filter + " AND KORISNIK_ID_IZRADIO = " + 
FKorisnikIzradioID.ToString();
            }
            //slaganje filtera Izdao
            if (FKorisnikIzdaoID != -1)
            {
                Filter = Filter + " AND KORISNIK_ID_IZDAO = " + 
FKorisnikIzdaoID.ToString();
            }
            //slaganje filtera Primio
            if (FKorisnikPrimioID != -1)
            {
                Filter = Filter + " AND KORISNIK_ID_PRIMIO =  " + 
FKorisnikPrimioID.ToString();
```

```
        }
        //MessageBox.Show(Filter, "filter");
        vRACUNBindingSource.Filter = Filter;

        this.vRACUNTableAdapter.Fill(this.dataDataSet.vRACUN);
    }

    //metoda koja ažurira dokument
    private void refreshDokument()
    {
        int dok = 0;

        if (vRACUNDataGridView.Rows.Count > 0)
        {
            if
(!String.IsNullOrEmpty(vRACUNDataGridView.CurrentRow.Cells[0].Value.ToString()))
            {

                dok = (int)vRACUNDataGridView.CurrentRow.Cells[0].Value;
            }
        }

        if (dok > 0)
        {

            SqlConnection cnn = new
SqlConnection(Properties.Settings.Default.DataConnectionString);
            cnn.Open();

            if (cnn.State == ConnectionState.Open)
            {
```

```
            try
            {
                SqlCommand cmd = new SqlCommand("dbo.REFRESH_DOKUMENT",
cnn);
                cmd.CommandType = CommandType.StoredProcedure;
                cmd.Parameters.Add(new SqlParameter("@ID", dok.ToString()));
                //SqlDataReader rdr = cmd.ExecuteReader();
                int rowAffected = cmd.ExecuteNonQuery();


            }
            catch (SqlException ex)
            {
                StringBuilder errorMessages = new StringBuilder();
                for (int i = 0; i < ex.Errors.Count; i++)
                {
                    errorMessages.Append("Index #" + i + "\n" +
                        "Message: " + ex.Errors[i].Message + "\n" +
                        "LineNumber: " + ex.Errors[i].LineNumber + "\n" +
                        "Source: " + ex.Errors[i].Source + "\n" +
                        "Procedure: " + ex.Errors[i].Procedure + "\n");
                }
                MessageBox.Show(errorMessages.ToString(), "Greška");
            }

        }
    }
}

    // klikom na gumb Traži poziva se metoda Ucitaj (filtrira bazu podataka i traži unesene
podatke)
    private void btnTrazi_Click(object sender, EventArgs e)
```

```
{
    Ucitaj();
}

//postavljanje svih polja na "prazne" vrijednosti klikom na Počisti
//string tipovi podataka su postavljeni na ""
//int tipovi podataka su postavljeni na -1
//datumi su postavljeni na trenutni datum
private void btnPocisti_Click(object sender, EventArgs e)
{
    tbBrojOd.Text = "";
    tbBrojDo.Text = "";
    tbGodinaOd.Text = "";
    tbGodinaDo.Text = "";
    tbPartneri.Text = "";
    tbIzdao.Text = "";
    tbIzradio.Text = "";
    tbPrimio.Text = "";
    pDatumValuteOd.Value = new DateTime(DateTime.Now.Year, 1, 1);
    pDatumValuteDo.Value = DateTime.Now;
    FPartnerID = -1;
    FKorisnikIzdaoID = -1;
    FKorisnikIzradioID = -1;
    FKorisnikPrimioID = -1;
}

private void PotraziPartnera(object sender)
{
    int max = 2;
    int max2 = 0;
```

```
int l = tbPartneri.TextLength;

//ako se metoda poziva s gumba zanemari se dužina teksta u kontroli za naziv partnera
if (sender is Button)
{
   max = -1;
   max2 = -1;
}

if (l > max2)
{
   //ukoliko je dužina unesenog dulja od 3 znaka
   if (l > max)
   {
      string naziv;
      frmUvidPartnera fForma = new frmUvidPartnera();
      FPartnerID = fForma.Prikazi(this, tbPartneri.Text, out naziv);
      if (FPartnerID == -1)
      {
         tbPartneri.Text = "";
      }
      else
      {
         //ispiši naziv
         tbPartneri.Text = naziv;
      }
   }
   else
   {
      //ako dužina teksta nije veća od dva javlja se poruka greške
```

```
                MessageBox.Show("Za pretraživanje unesite bar 3 znaka!", "Info");
            }
        }
    }

    //klikom na gumb kojim se pretražuje partner poziva se funkcija PotraziPartnera
    private void button1_Click(object sender, EventArgs e)
    {
        PotraziPartnera(sender);
    }

    private void PotraziIzdao(object sender)
    {
        int max = 2;
        int max2 = 0;
        int l = tbIzdao.TextLength;

        //ako se metoda poziva s gumba zanemari se dužina teksta u kontroli za naziv
korisnika
        if (sender is Button)
        {
            max = -1;
            max2 = -1;
        }

        //ako je prazan vraća poruku da je za pretraživanje potrebno bar 3 znaka

        if (l > max2)
        {
            //ukoliko je dužina unesenog dulja od 3 znaka
            if (l > max)
```

```
        {
            string naziv;
            frmUvidKorisnika fForma = new frmUvidKorisnika();
            FKorisnikIzdaoID = fForma.Prikazi(this, tbIzdao.Text, out naziv);
            if (FKorisnikIzdaoID == -1)
            {
                tbIzdao.Text = "";
            }
            else
            {
                //ispiši naziv
                tbIzdao.Text = naziv;
            }
        }
        else
        {
            MessageBox.Show("Za pretraživanje unesite bar 3 znaka!", "Info");
        }
    }
}

private void PotraziIzradio(object sender)
{
    int max = 2;
    int max2 = 0;
    int l = tbIzradio.TextLength;

    if (sender is Button)
    {
        max = -1;
```

```
        max2 = -1;
    }

    if (l > max2)
    {
        if (l > max)
        {
            string naziv;
            frmUvidKorisnika fForma = new frmUvidKorisnika();
            FKorisnikIzradioID = fForma.Prikazi(this, tbIzradio.Text, out naziv);
            if (FKorisnikIzradioID == -1)
            {
                tbIzradio.Text = "";
            }
            else
            {
                tbIzradio.Text = naziv;
            }
        }
        else
        {
            MessageBox.Show("Za pretraživanje unesite bar 3 znaka!", "Info");
        }
    }
}

        private void PotraziPrimio(object sender)
        {
            int max = 2;
            int max2 = 0;
```

```
int l = tbPrimio.TextLength;

if (sender is Button)
{
    max = -1;
    max2 = -1;
}

if (l > max2)
{
    if (l > max)
    {
        string naziv;
        frmUvidKorisnika fForma = new frmUvidKorisnika();
        FKorisnikPrimioID = fForma.Prikazi(this, tbPrimio.Text, out naziv);
        if (FKorisnikPrimioID == -1)
        {
            tbPrimio.Text = "";
        }
        else
        {
            tbPrimio.Text = naziv;
        }
    }
    else
    {
        MessageBox.Show("Za pretraživanje unesite bar 3 znaka!", "Info");

    }
}
```

```
    }

private void button2_Click(object sender, EventArgs e)
{
  PotraziIzradio(sender);
}

private void button3_Click(object sender, EventArgs e)
{
  PotraziIzdao(sender);
}

private void button4_Click(object sender, EventArgs e)
{
  PotraziPrimio(sender);
}

private void tbPartneri_KeyPress(object sender, KeyPressEventArgs e)
{
  if (e.KeyChar == Convert.ToChar(Keys.Return))
  {
    PotraziPartnera(sender);
  }
}

private void tbPartneri_Leave(object sender, EventArgs e)
{
  PotraziPartnera(sender);
}
```

```
private void frmPregledDokumenata_Shown(object sender, EventArgs e)
{
    refreshStavke();
}

//metoda pozoviDokument koja omogućuje kreiranje novog dokumenta
private void pozoviDokuemnt(int pTip)
{
    //kad proslijedimo ID=0 znači da dodaje novi zapis jer u tablici ne postoji zapis s ID-
jem 0
    int dokumentID = 0;

    if (pTip == 1)  //kad je 1 onda je izmjieni
    {
        //sve veće od 0 je izmijeni
        if (vRACUNDataGridView.Rows.Count > 0)
        {
            if
(!String.IsNullOrEmpty(vRACUNDataGridView.CurrentRow.Cells[0].Value.ToString()))
            {

                dokumentID = (int)vRACUNDataGridView.CurrentRow.Cells[0].Value;
            }
        }
    }
    //pozivanje forme frmDokument
    frmDokument fForma = new frmDokument();
    fForma.Prikazi(this, dokumentID, FVrsta_Dokumenta_ID, FUserID);
}

//klikom na znak + poziva se:
```

```
//metoda PozoviDokument (omogućuje kreiranje novog dokumenta) i
//Ucitaj (metoda koja filtrira bazu podataka i traži unesene podatke)
private void tsbDodajDokument_Click(object sender, EventArgs e)
{
   pozoviDokuemnt(0);
   Ucitaj();
}

//metoda koja se poziva klikom na gumb Obriši u navigacijskoj traci
private void tsbObrisiDokument_Click(object sender, EventArgs e)
{
   int k = 0;
   string god = "";
   string br = "";
   //u lokalne varijable k,god i br dohvaćene su vrijednosti iz grida
   try
   {
      k = (int)vRACUNDataGridView.CurrentRow.Cells[0].Value;
      god = vRACUNDataGridView.CurrentRow.Cells[1].Value.ToString();
      br = vRACUNDataGridView.CurrentRow.Cells[2].Value.ToString();
   }

   catch
   {
      k = 0;
   }

   if (k > 0)
   {
```

```
            if (MessageBox.Show($"Želite li obrisati Dokument {god}\\{br}?", "Potvrda",
MessageBoxButtons.YesNo) == DialogResult.Yes)
            {

                SqlConnection cnn = new
SqlConnection(Properties.Settings.Default.DataConnectionString);
                cnn.Open();

                if (cnn.State == ConnectionState.Open)
                {
                    try //ako korisnik odabere Yes poziva se funkcija OBRISI_DOKUMENT
                    {
                        //pozivanje funkcije OBRISI_DOKUMENT koja briše dokument po ID-u
                        SqlCommand cmd = new SqlCommand("dbo.OBRISI_DOKUMENT",
cnn);
                        cmd.CommandType = CommandType.StoredProcedure;
                        cmd.Parameters.Add(new SqlParameter("@ID", k.ToString()));
                        //SqlDataReader rdr = cmd.ExecuteReader();
                        int rowAffected = cmd.ExecuteNonQuery();

                    }
                    catch (SqlException ex)
                    {
                        StringBuilder errorMessages = new StringBuilder();
                        for (int i = 0; i < ex.Errors.Count; i++)
                        {
                            errorMessages.Append("Index #" + i + "\n" +
                                "Message: " + ex.Errors[i].Message + "\n" +
                                "LineNumber: " + ex.Errors[i].LineNumber + "\n" +
                                "Source: " + ex.Errors[i].Source + "\n" +
                                "Procedure: " + ex.Errors[i].Procedure + "\n");
```

```
            }
            MessageBox.Show(errorMessages.ToString(), "Greška");
          }



          cnn.Close();
          Ucitaj(); //metoda koja filtrira bazu podataka i traži unesene podatke
        }
      }
    }
  }
```

//dvostrukim klikom tablicu poziva se metoda PozoviDokument koja omogućuje kreiranje novog dokumenta

```
  private void vRACUNDataGridView_CellMouseDoubleClick(object sender, DataGridViewCellMouseEventArgs e)
  {
    pozoviDokuemnt(1);
  }

  private void vRACUNDataGridView_SelectionChanged(object sender, EventArgs e)
  {
    refreshStavke();
  }
```

//klikom na ikonu Izmijeni poziva se metoda PozoviDokument koja omogućuje kreiranje novog dokumenta

```
  private void tsbDokumentEdit_Click(object sender, EventArgs e)
  {
      pozoviDokuemnt(1);  //kad je 1 izmijeni, kad je 0 je dodaj
```

```
        }

        private void pozoviStavku(int pTip)
        {
            int dokumentID = 0; //ako je 0 onda dodaj novu

            try
            {
                if (vRACUNDataGridView.Rows.Count > 0)
                {
                    if
(!String.IsNullOrEmpty(Convert.ToString(vRACUNDataGridView.CurrentRow.Cells[0].Val
ue)))
                    {

                        dokumentID = (int)vRACUNDataGridView.CurrentRow.Cells[0].Value;
                    }
                }
            }

            catch
            {
                dokumentID = 0;
            }

            if (dokumentID != 0)    //ako ID dokumenta nije 0
            {
                int stavkaID = 0;

                try
                {
```

```
            if (pTip == 1)  //ako je 1 (i više) onda izmijeni
            {
                if (vStavkeDataGridView.Rows.Count > 0)
                {
                    if
(!String.IsNullOrEmpty(vStavkeDataGridView.CurrentRow.Cells[0].Value.ToString()))
                    {

                        stavkaID = (int)vStavkeDataGridView.CurrentRow.Cells[0].Value;
                    }
                }
            }
        }

        catch
        {
            stavkaID = 0;
        }

        if (((pTip == 1)  && (stavkaID > 0)) || (pTip == 0))
        {
            frmStavka fForma = new frmStavka();
            fForma.Prikazi(this, stavkaID, dokumentID);

            Ucitaj(); //metoda koja filtrira bazu podataka i traži unesene podatke
            refreshStavke();    //dohvaća podatke za stavku dokumenta
        }
    }
}

private void tsbDokumentStavkaAdd_Click(object sender, EventArgs e)
```

```
        {
            pozoviStavku(0);    //ako je 0 dodaj
        }

        private void vStavkeDataGridView_CellMouseDoubleClick(object sender,
DataGridViewCellMouseEventArgs e)
        {
            pozoviStavku(1);    //ako je 1 izmijeni
        }

        private void tsbDokumentStavkaEdit_Click(object sender, EventArgs e)
        {
            pozoviStavku(1);    //ako je 1 izmijeni
        }

        private void tsbDokumentStavkaDelete_Click_1(object sender, EventArgs e)
        {
            //u lokalne varijable k dohvaćena je vrijednosti iz grida

            int k = 0;

            try
            {
                if (vStavkeDataGridView.Rows.Count > 0)
                {
                    if
(!String.IsNullOrEmpty(vStavkeDataGridView.CurrentRow.Cells[0].Value.ToString()))
                    {

                        k = (int)vStavkeDataGridView.CurrentRow.Cells[0].Value;
                    }
```

```
            }
        }

        catch
        {
            k = 0;
        }

        if (k > 0)
        {

            if (MessageBox.Show($"Želite li obrisati stavku?", "Potvrda",
MessageBoxButtons.YesNo) == DialogResult.Yes)
            {

                SqlConnection cnn = new
SqlConnection(Properties.Settings.Default.DataConnectionString);
                cnn.Open();

                if (cnn.State == ConnectionState.Open)
                {
                    try //ako korisnik odabere Yes poziva se funkcija OBRISI_STAVKU
                    {
                        //pozivanje funkcije OBRISI_STAVKU koja briše dokument po ID-u
                        SqlCommand cmd = new SqlCommand("dbo.OBRISI_STAVKU", cnn);
                        cmd.CommandType = CommandType.StoredProcedure;
                        cmd.Parameters.Add(new SqlParameter("@ID", k.ToString()));
                        //SqlDataReader rdr = cmd.ExecuteReader();
                        int rowAffected = cmd.ExecuteNonQuery();

                    }
```

```
            catch (SqlException ex)
            {
                StringBuilder errorMessages = new StringBuilder();
                for (int i = 0; i < ex.Errors.Count; i++)
                {
                    errorMessages.Append("Index #" + i + "\n" +
                        "Message: " + ex.Errors[i].Message + "\n" +
                        "LineNumber: " + ex.Errors[i].LineNumber + "\n" +
                        "Source: " + ex.Errors[i].Source + "\n" +
                        "Procedure: " + ex.Errors[i].Procedure + "\n");
                }
                MessageBox.Show(errorMessages.ToString(), "Greška");
            }
            cnn.Close();
            refreshDokument();
            Ucitaj(); //metoda koja filtrira bazu podataka i traži unesene podatke
        }
    }
  }
}

private void tbIzradio_KeyPress(object sender, KeyPressEventArgs e)
{
    if (e.KeyChar == Convert.ToChar(Keys.Return))
    {
        PotraziIzradio(sender);
    }
}

private void tbIzradio_Leave(object sender, EventArgs e)
```

```
        {
            PotraziIzradio(sender);
        }

        private void tbIzdao_KeyPress(object sender, KeyPressEventArgs e)
        {
            if (e.KeyChar == Convert.ToChar(Keys.Return))
            {
                PotraziIzdao(sender);
            }
        }
        private void tbIzdao_Leave(object sender, EventArgs e)
        {
            PotraziIzdao(sender);
        }

        private void tbPrimio_KeyPress(object sender, KeyPressEventArgs e)
        {
            if (e.KeyChar == Convert.ToChar(Keys.Return))
            {
                PotraziPrimio(sender);
            }
        }

        private void tbPrimio_Leave(object sender, EventArgs e)
        {
            PotraziPrimio(sender);
        }
    }
}
```

Prilog 5 – Kod forme frmDokument

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

namespace Trgovina
{
    public partial class frmDokument : Form
    {
        // definiranje polja partner, dokument i id vrste dokumenta na vrijednost -1
        private int FDokumentID = -1;
        private int FPartnerID = -1;
        private int FVrsta_Dokumenta_ID = 0;
        private int FUserID;



        public frmDokument()
        {
            //kada se uđe u formu,partner i dokument će biti prazni
            FDokumentID = -1;
            FPartnerID = -1;
            FUserID = 0;
            InitializeComponent();
        }

        //metoda koja prikazuje formu
        public int Prikazi(object sender, int pID, int pVrsta_Dokumenta_ID, int
pUserId)
        {

            FVrsta_Dokumenta_ID = pVrsta_Dokumenta_ID;
            FDokumentID = pID;
            FUserID = pUserId;
            FPartnerID = -1;
            ShowDialog(sender as Form);
            return FDokumentID;
        }


        private void dOKUMENTBindingNavigatorSaveItem_Click(object sender, EventArgs
e)
        {
            this.Validate();
            this.dOKUMENTBindingSource.EndEdit();
            this.tableAdapterManager.UpdateAll(this.dataDataSet);

        }

        //metoda kojom se poziva funkcija NEXT_BROJ koja vraća broj povećan za jedan
od prehodnog ID-a
        private string nextBroj()
        {
```

```
            using (SqlConnection conn = new
SqlConnection(Properties.Settings.Default.DataConnectionString))
            using (SqlCommand cmd = new SqlCommand("SELECT dbo.NEXT_BROJ('" +
FVrsta_Dokumenta_ID.ToString() + "')", conn))
            {
                conn.Open();
                var result = cmd.ExecuteScalar();
                conn.Close();

                return  result.ToString();
            }
        }

        private void frmDokument_Load(object sender, EventArgs e)
        {



            // TODO: This line of code loads data into the
'dataDataSet.NACIN_PLACANJA' table. You can move, or remove it, as needed.
            this.nACIN_PLACANJATableAdapter.Fill(this.dataDataSet.NACIN_PLACANJA);
            // TODO: This line of code loads data into the
'dataDataSet.POSLOVNA_JEDINICA' table. You can move, or remove it, as needed.

this.pOSLOVNA_JEDINICATableAdapter.Fill(this.dataDataSet.POSLOVNA_JEDINICA);
            // TODO: This line of code loads data into the
'dataDataSet.VRSTA_DOKUMENTA' table. You can move, or remove it, as needed.
            this.vRSTA_DOKUMENTATableAdapter.Fill(this.dataDataSet.VRSTA_DOKUMENTA);
            // TODO: This line of code loads data into the 'dataDataSet.DOKUMENT'
table. You can move, or remove it, as needed.

            //filtriram dokument
            dOKUMENTBindingSource.Filter = "ID = " + FDokumentID.ToString();


            this.dOKUMENTTableAdapter.Fill(this.dataDataSet.DOKUMENT);

            //ako je ID nula=
            //ukoliko je riječ o novom dokumentu, neka godina bude 2020 i nek se
automatski postavi broj dokumenta
            if (FDokumentID == 0)
            {
                this.dOKUMENTBindingSource.AddNew();
                gODINATextBox.Text = "2020";
                dATUMDateTimePicker.Value = DateTime.Now;
                bROJTextBox.Text = nextBroj();
                // logirani korisnik


                kORISNIK_ID_IZRADIOTextBox.Text = FUserID.ToString();
                kORISNIK_ID_PRIMIOTextBox.Text = FUserID.ToString();
                kORISNIK_ID_IZDAOTextBox.Text = FUserID.ToString();




            }
            //ukoliko ID nije nula=
            //naziv partnera se ne puni automatski
            else
            {
                FPartnerID = Int32.Parse(pARTNER_IDTextBox.Text);
```

```
                string nazivPartnera = "";

                using (SqlConnection conn = new
SqlConnection(Properties.Settings.Default.DataConnectionString))
                using (SqlCommand cmd = new SqlCommand("SELECT dbo.GET_PARTNER('" +
FPartnerID.ToString() + "')", conn))
                {
                    conn.Open();
                    var result = cmd.ExecuteScalar();
                    conn.Close();

                    nazivPartnera = result.ToString();
                }

                tbPartneri.Text = nazivPartnera;
            }

        }

        //metoda se poziva klikom na gumb pokraj tekstualnog okvira Partneri
        private void PotraziPartnera(object sender)
        {
            int max = 2;
            int max2 = 0;
            int l = tbPartneri.TextLength;

            //ako se metoda poziva s gumba zanemari se dužina teksta u kontroli za
naziv partnera
            if (sender is Button)
            {
                max = -1;
                max2 = -1;
            }

            //ukoliko su uneseni znakovi
            if (l > max2)
            {
                //ukoliko je dužina unesenog partnera dulja od 3 znaka

                if (l > max)
                {
                    string naziv;
                    frmUvidPartnera fForma = new frmUvidPartnera();
                    FPartnerID = fForma.Prikazi(this, tbPartneri.Text, out naziv);
                    if (FPartnerID == -1)
                    {
                        tbPartneri.Text = "";
                    }
                    else
                    {
                        //ispiši naziv
                        tbPartneri.Text = naziv;
                    }
                }
                else
                {
                    //provjeri je li duljina unešenog dulja od 3 znaka, a ako nije
javlja se poruka greške
                    MessageBox.Show("Za pretraživanje unesite bar 3 znaka!", "Info");

                }
            }
```

```
        }

        //klikom na gumb pokraj tekstualnog okvira Partner poziva se metoda 
PotraziPartnera
        //ukoliko se unesu minimalno 3 znaka moguće je automatski pronaći partnera
        //ukoliko se klikne na gumb, tekstualni okvir ostaje prazan, a puni se tek 
odabirom partnera u novootvorenoj formi
        private void button1_Click(object sender, EventArgs e)
        {
                PotraziPartnera(sender);
        }


        //klikom na Prihvati napusi se partner
        private void btnPrihvati_Click(object sender, EventArgs e)
        {

          //  MessageBox.Show(cbPoslovnaJedinica.Text, "cbPoslovnaJedinica.Text");
            if ((FPartnerID != -1) && 
(!String.IsNullOrEmpty(cbPoslovnaJedinica.Text))) //ukoliko je zadovoljen uvjet
            {
                pARTNER_IDTextBox.Text = FPartnerID.ToString();
                vRSTA_DOKUMENTA_IDTextBox.Text = FVrsta_Dokumenta_ID.ToString();
                //onda on napuni partnera


                this.Validate();
                this.dOKUMENTBindingSource.EndEdit();
                this.tableAdapterManager.UpdateAll(this.dataDataSet);
                Close();
            }
        }

        //klikom na Odustani zatvara se forma
        private void btnOdustani_Click(object sender, EventArgs e)
        {
            Close();
        }

        private void frmDokument_Shown(object sender, EventArgs e)
        {
            bROJTextBox.Focus();
        }

    }
}
```

Prilog 6 – Kod forme frmStavka

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

namespace Trgovina
{
    public partial class frmStavka : Form
    {

        int FStavkaID = -1;
        int FArtiklID = -1;
        int FDokumentID = -1;

        public frmStavka()
        {
            FStavkaID = -1;
            FArtiklID = -1;
            FDokumentID = -1;
            InitializeComponent();
        }



        //metoda koja računa iznose
        private void Izracunaj()
        {
            //početne vrijednosti pomoćnih varijabli stavljene su na 0
            decimal kolicina = 0;
            decimal cijena = 0;
            decimal rabat = 0;
            decimal rabat_iznos = 0;
            decimal porez = 0;
            decimal porez_iznos = 0;
```

```
    decimal iznos = 0;
    decimal ukupno = 0;
    try
    {
        kolicina = nbKolicina.Value;
        cijena = nbCijena.Value;
        rabat = nbRabat.Value;
        porez = nbPorez.Value;

        decimal ms = kolicina * cijena;
        rabat_iznos = (rabat / 100) * ms;
        iznos = ms - rabat_iznos;
        decimal pi = 1 + (porez / 100);
        ukupno = (pi * iznos);
        porez_iznos = ukupno - iznos;
    }
    catch (System.Exception)
    {
        rabat_iznos = 0;
        porez_iznos = 0;
        iznos = 0;
        ukupno = 0;
    }

    nbIznosRabata.Value = rabat_iznos;
    nbIznosPoreza.Value = porez_iznos;
    nbIznos.Value = iznos;
    nbUkupno.Value = ukupno;

}

public int Prikazi(object sender, int pID, int pDokumentID)
{
    FStavkaID = pID;
    FArtiklID = -1;
    FDokumentID = pDokumentID;
    ShowDialog(sender as Form);
    return FStavkaID;
}

private void frmStavka_Load(object sender, EventArgs e)
```

```
        {
            // TODO: This line of code loads data into the 'dataDataSet.ARTIKL' table. You can move, or remove it, as needed.
            this.aRTIKLTableAdapter.Fill(this.dataDataSet.ARTIKL);
            // TODO: This line of code loads data into the 'dataDataSet.DOKUMENT_STAVKA' table. You can move, or remove it, as needed.

            dOKUMENT_STAVKABindingSource.Filter = "ID = " + FStavkaID.ToString();
            this.dOKUMENT_STAVKATableAdapter.Fill(this.dataDataSet.DOKUMENT_STAVKA);

            if (FStavkaID == 0)
            {
                this.dOKUMENT_STAVKABindingSource.AddNew();
                dOKUMENT_IDTextBox.Text = FDokumentID.ToString();
                tbRedniBroj.Text = nextRBR();
            }
            else
            {
                FArtiklID = Int32.Parse(aRTIKL_IDTextBox.Text);
                string nazivArtikla = "";

                //spaja se na bazu, poziva funkciju GET_ARTIKL_N koja vraća naziv artikla
                using (SqlConnection conn = new SqlConnection(Properties.Settings.Default.DataConnectionString))
                using (SqlCommand cmd = new SqlCommand("SELECT dbo.GET_ARTIKL_N('" + FArtiklID.ToString() + "')", conn))
                {
                    conn.Open();
                    var result = cmd.ExecuteScalar();
                    conn.Close();

                    nazivArtikla = result.ToString();
                }

                tbArtiklNaziv.Text = nazivArtikla;

                string sifraArtikla = "";

                //spaja se na bazu, poziva funkciju GET_ARTIKL_S koja vraća šifru artikla
                using (SqlConnection conn = new SqlConnection(Properties.Settings.Default.DataConnectionString))
```

```
                using (SqlCommand cmd = new SqlCommand("SELECT dbo.GET_ARTIKL_S('" + FArtiklID.ToString() + "')", conn))
                {
                    conn.Open();
                    var result = cmd.ExecuteScalar();
                    conn.Close();

                    sifraArtikla = result.ToString();
                }
                tbArtikl.Text = sifraArtikla;

                string jmArtikla = "";

                //spaja se na bazu, poziva funkciju GET_ARTIKL_JM koja vraća naziv jedinice mjere
                using (SqlConnection conn = new SqlConnection(Properties.Settings.Default.DataConnectionString))
                using (SqlCommand cmd = new SqlCommand("SELECT dbo.GET_ARTIKL_JM('" + FArtiklID.ToString() + "')", conn))
                {
                    conn.Open();
                    var result = cmd.ExecuteScalar();
                    conn.Close();

                    jmArtikla = result.ToString();
                }
                tbJM.Text = jmArtikla;

            }
        }
        //metoda koja omogućuje spajanje na bazu i pozivanje funkcije NEXT_STAVKA koja vraća broj posljednje unešene stavke + 1
        private string nextRBR()
        {
            using (SqlConnection conn = new SqlConnection(Properties.Settings.Default.DataConnectionString))
            using (SqlCommand cmd = new SqlCommand("SELECT dbo.NEXT_STAVKA('" + dOKUMENT_IDTextBox.Text + "')", conn))
            {
                conn.Open();
                var result = cmd.ExecuteScalar();
                conn.Close();
```

```
                return result.ToString();
            }
        }

        private void textBox1_TextChanged(object sender, EventArgs e)
        {
            frmUvidArtikala fForma = new frmUvidArtikala();
            fForma.ShowDialog();
        }

        //metoda koja poziva funkciju POREZNASTOPA_STAVKA koja se povezuje s poveznom grupom i vraća poveznu stopu na današnji datum
        private decimal DohvatiStopu(int pGrupaID)
        {

            decimal res= 0;
            using (SqlConnection conn = new SqlConnection(Properties.Settings.Default.DataConnectionString))
            using (SqlCommand cmd = new SqlCommand("SELECT dbo.POREZNASTOPA_STAVKA('" + pGrupaID.ToString() + "')", conn))
            {
                conn.Open();
                var result = cmd.ExecuteScalar();
                conn.Close();

                res = decimal.Parse(result.ToString());
            }

            return res;
        }

        //klikom na gumb Odustani zatvara se forma
        private void btnOdustani_Click(object sender, EventArgs e)
        {

            Close();
        }

        //metoda koja provjerava je li korisnik unio više od dva znaka
        //ukoliko je puni tekstualna polja s varijablama u kojima su podaci o stavkama
        private void PotraziArtikl(object sender)
        {
```

```
            int max = 2;
            int max2 = 0;
            int l = tbArtikl.TextLength;

            if (sender is Button)
            {
                max = -1;
                max2 = -1;
            }



            if (l > max2)
            {
                if (l > max)        //ukoliko je korisnik unio više od 2 znaka
                {
                    string sifra;
                    string naziv;
                    string opis;
                    string jm;
                    int pg;
                    decimal cijena;

                    frmUvidArtikala fForma = new frmUvidArtikala();
                    //object sender, string pNaziv, out string pArtiklSifra, out string
pArtiklNaziv, out string pArtiklOpis, out string pArtiklJM, out int pArtiklPG, out
double pCijena)
                    FArtiklID = fForma.Prikazi(this, tbArtikl.Text, out sifra , out
naziv , out opis, out jm, out pg, out cijena);
                    if (FArtiklID == -1)
                    {
                        tbArtikl.Text = "";
                        tbArtiklNaziv.Text = "";
                        tbJM.Text = "";
                        nbCijena.Value = 0;
                        nbPorez.Value = 0;
                        aRTIKL_IDTextBox.Text = "";
                    }
                    else
                    {
                        /*
                         MessageBox.Show(sifra, "sifra");
                         MessageBox.Show(naziv, "naziv");
```

```
                     MessageBox.Show(opis, "opis");
                     MessageBox.Show(jm, "jm");
                     MessageBox.Show(pg.ToString(), "pg");
                     MessageBox.Show(cijena.ToString(), "cijena");
                     */
                    aRTIKL_IDTextBox.Text = FArtiklID.ToString();
                    tbArtikl.Text = sifra;
                    tbArtiklNaziv.Text = naziv;
                    tbJM.Text = jm;
                    nbCijena.Value = cijena;

                    if (pg != -1)
                    {
                        nbPorez.Value = DohvatiStopu(pg);
                    }
                    else
                    {
                        nbPorez.Value = 0;
                    }
                }
                Izracunaj();     //metoda koja računa stavke na temelju dostupnih
podataka
            }
            else
            {
                MessageBox.Show("Za pretraživanje unesite bar 3 znaka!", "Info");
            }
        }
    }

    //klikom na gumb pokraj tekstualnog okvira Artikl poziva se metoda PotraziArtikl
    private void button1_Click(object sender, EventArgs e)
    {
        PotraziArtikl(sender); //metoda koja puni tekstualna polja s varijablama u
kojima su podaci o stavkama
    }

    private void tbArtikl_Leave(object sender, EventArgs e)
    {
        PotraziArtikl(sender);
    }
```

```
        private void tbArtikl_KeyPress(object sender, KeyPressEventArgs e)
        {
            if (e.KeyChar == Convert.ToChar(Keys.Return))
            {
                PotraziArtikl(sender);
            }
        }

        //metoda koja ažurira dokument
        private void refreshDokument()
        {
            if (FDokumentID > 0)
            {

                SqlConnection cnn = new
SqlConnection(Properties.Settings.Default.DataConnectionString);
                cnn.Open();

                if (cnn.State == ConnectionState.Open)
                {
                    try
                    {
                        SqlCommand cmd = new SqlCommand("dbo.REFRESH_DOKUMENT", cnn);
                        cmd.CommandType = CommandType.StoredProcedure;
                        cmd.Parameters.Add(new SqlParameter("@ID",
FDokumentID.ToString()));
                        //SqlDataReader rdr = cmd.ExecuteReader();
                        int rowAffected = cmd.ExecuteNonQuery();

                    }
                    catch (SqlException ex)
                    {
                        StringBuilder errorMessages = new StringBuilder();
                        for (int i = 0; i < ex.Errors.Count; i++)
                        {
                            errorMessages.Append("Index #" + i + "\n" +
                                "Message: " + ex.Errors[i].Message + "\n" +
                                "LineNumber: " + ex.Errors[i].LineNumber + "\n" +
                                "Source: " + ex.Errors[i].Source + "\n" +
                                "Procedure: " + ex.Errors[i].Procedure + "\n");
```

```
                }
                MessageBox.Show(errorMessages.ToString(), "Greška");
            }
        }
    }
}

private void nbKolicina_ValueChanged(object sender, EventArgs e)
{
    Izracunaj();
}

private void nbRabat_ValueChanged(object sender, EventArgs e)
{
    Izracunaj();
}

private void btnPrihvati_Click(object sender, EventArgs e)
{

    if (FArtiklID != -1)
    {
        Validate();

        dOKUMENT_STAVKABindingSource.EndEdit();
        tableAdapterManager.UpdateAll(dataDataSet);

        dataDataSet.AcceptChanges();

        refreshDokument(); //nakon prihvati poziva se metoda refreshDokument
koja ažurira dokument

        Close();
    }
}
    }
}
```

Izrada poslovnih aplikacija koristeći programski jezik C#

Sažetak

Cilj ovog rada je objasniti mogućnosti razvoja poslovnih aplikacija u suvremenom poslovnom okruženju. Aplikacija će se koristiti za izrade ponuda kupcima. Najprije će biti kreirana baza podataka koja će sadržavati potrebne tablice s podacima poput korisnika, partnera, proizvoda, ponude i drugog. Razvoj aplikacije biti će obavljen u razvojnom okruženju Visual Studio koristeći C# programski jezik. Naime, aplikacija će omogućiti dodavanje ponude partnerima, izrađivanje i pregled narudžbenice, kreiranje ulaznih i izlaznih računa. Rad će biti potkrijepljen prilozima. Ovaj rad može pomoći svima onima koji žele istraživati o mogućnostima razvoja aplikacija koristeći C# programski jezik, od analize korisničkih zahtjeva, izrade specifikacije, izrade baze podataka, izrade aplikacije i izrade izvještaja.

Ključne riječi:

C# programski jezik, baza podataka, aplikacija, ponuda

Business application development using C# programming language

Summary

The aim of this paper is to explain the possibilities of application development in a modern business environment. The aim of the application is to be used to create offers to customers. First, a database will be created that will contain the necessary tables with information's about users (or customers), products and offers of others. Application will be developed in the integrated development environment Visual Studio using the C # programming language. Accordingly, the application will add partner offers, create and review purchase orders, create input and output computers. The paper will be supported by attachments and pictures. This paper can help anyone who wants to explore the possibility of developing an application in C # programming language, from analyzing user requirements, creating specifications, creating databases, creating applications, and creating reports.

Key words:

C# programming language, database, application, offer