Aplikacija za trgovinu rabljene robe

Kevilj, Andro

Sveučilište u Zagrebu

Ekonomski fakultet

Integrirani preddiplomski i diplomski sveučilišni studij

Poslovna ekonomija - smjer Menadžerska informatika

APLIKACIJA ZA TRGOVINU RABLJENE ROBE

Diplomski rad

Andro Kevilj

Zagreb, srpanj 2023.

Sveučilište u Zagrebu

Ekonomski fakultet

Integrirani preddiplomski i diplomski sveučilišni studij

Poslovna ekonomija - smjer Menadžerska informatika

APLIKACIJA ZA TRGOVINU RABLJENE ROBE

SOFTWARE FOR SECOND HAND BUSINESS

Diplomski rad

Student: Andro Kevilj

JMBAG studenta: 0067489345

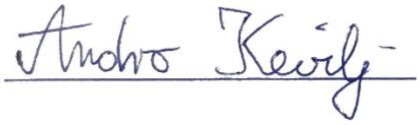
Mentor: Prof. dr. sc. Mirjana Pejić Bach

Zagreb, srpanj 2023.

IZJAVA O AKADEMSKOJ ČESTITOSTI

Izjavljujem i svojim potpisom potvrđujem da je prijava teme diplomskog rada isključivo rezultat mog vlastitog rada koji se temelji na mojim istraživanjima i oslanja se na objavljenu literaturu, a što pokazuju korištene bilješke i bibliografija.

Izjavljujem da nijedan dio prijave teme nije napisan na nedozvoljen način, odnosno da je prepisan iz ne citiranog izvora te da nijedan dio rada / prijave teme ne krši bilo čija autorska prava. Izjavljujem, također, da nijedan dio prijave teme nije iskorišten za bilo koji drugi rad u bilo kojoj drugoj visokoškolskoj, znanstvenoj ili obrazovnoj ustanovi**.**



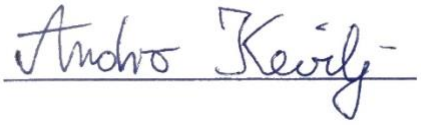
Andro Kevilj

Zagreb, 6. srpnja 2023.

STATEMENT ON THE ACADEMIC INTEGRITY

I hereby declare and confirm by my signature that the final thesis is the sole result of my own work based on my research and relies on the published literature, as shown in the listed notes and bibliography.

I declare that no part of the thesis has been written in an unauthorized manner, i.e., it is not transcribed from the non-cited work, and that no part of the thesis infringes any of the copyrights. I also declare that no part of the thesis has been used for any other work in any other higher education, scientific or educational institution.



Andro Kevilj

Zagreb, July 6th 2023

Sadržaj

1. Uvod

Trgovina rabljenom robom vrsta je poslovanja koja se bavi kupovinom i prodajom robe koja je već korištena od strane drugih osoba, a ta roba može biti odjeća, obuća, nakit, knjige, igračke, elektronika i druge stvari koje imaju neku vrijednost i potražnju na tržištu. Trgovina rabljenom robom može imati različite motive i ciljeve, kao što su zarada, održivost, humanitarnost ili zadovoljavanje specifičnih potreba kupaca. Roba za prodaju može se nabavljati na različite načine, kao što su otkup, donacija, zamjena ili komisijska prodaja. Osim toga, komisijska prodaja je način prodaje robe u kojem trgovina preuzima robu od vlasnika koji želi prodati svoju robu, ali ne prima novac unaprijed, već samo dio prodajne cijene nakon što se roba proda, a taj način prodaje omogućuje trgovini da ima veći izbor robe i manji rizik od gubitka novca, a vlasniku robe da ostvari neki prihod od svoje robe koju više ne koristi.

1.1. Predmet i cilj rada

Predmet ovog rada je izrada aplikacije za trgovinu rabljenom robom koja se bavi komisijskom prodajom. Cilj rada je prikazati proces razvoja aplikacije od analize problema i zahtjeva do dokumentiranja aplikacije i prikaza njenog rada. Aplikacija je namijenjena za korištenje u trgovini koja ima problem praćenja i evidentiranja stanja robe i vlasnika robe u trgovini i treba omogućiti trgovini da na jednostavan način unosi, mijenja i briše podatke o robi i vlasnicima robe u bazi podataka, te da evidentira prodaju robe i izračunava profit trgovine.

Izvori podataka za izradu aplikacije su sljedeći:

- Podaci o robi i vlasnicima robe koji se nalaze u trgovini ili koji se dostavljaju trgovini od strane vlasnika robe.
- Podaci o prodaji robe koji se bilježe prilikom svake transakcije u trgovini.
- Podaci o tržištu rabljene robe koji se mogu pronaći na internetu ili u literaturi.

Metode prikupljanja podataka za izradu aplikacije su sljedeće:

- Intervju sa naručiteljem aplikacije koji je vlasnik trgovine rabljenom robom.
- Istraživanje postojećih aplikacija za trgovinu rabljenom robom koje se mogu pronaći na internetu ili u literaturi.
- Testiranje aplikacije sa stvarnim podacima iz trgovine.

1.2. Sadržaj i struktura rada

Rad se sastoji od osam poglavlja, od kojih svako obrađuje određenu temu vezanu za izradu i predstavljanje aplikacije za trgovinu rabljenom robom koja se bavi komisijskom prodajom. U prvom poglavlju je izložen uvod u rad, u kojem je opisan predmet i cilj rada, te sadržaj i struktura rada. Zatim, u drugom poglavlju je opisan problem koji se rješava aplikacijom, a utvrđeni su i zahtjevi naručitelja aplikacije. U trećem poglavlju je opisana prodaja i maloprodaja kao poslovni procesi koji su relevantni za aplikaciju, gdje je definiran pojam i vrste prodaje i maloprodaje, opisan razvoj prodaje i maloprodaje kroz povijest, navedeni trendovi prodaje i maloprodaje u suvremenom svijetu, te objašnjen koncept komisijske prodaje kao specifičnog načina prodaje rabljene robe. U sljedećem poglavlju je oblikovan problem u obliku dijagrama koji prikazuju funkcionalnost i strukturu aplikacije, gdje su izrađeni dijagram korištenja, dijagram aktivnosti, dijagram slijeda i dijagram klasa. Sljedeće poglavlje opisuje baze podataka, koja su bitne za pohranu i obradu podataka u aplikaciji. U poglavlju su definirani pojam i vrste baza podataka, opisano je modeliranje i dizajniranje baza podataka, navedene su prednosti i nedostaci korištenja baza podataka u

razvoju aplikacija, te je predstavljen Microsoft SQL Server kao sustav za upravljanje bazama podataka koji se koristi u aplikaciji. U šestom poglavlju je opisana tema programskih jezika koja je bitna za izradu logike i sučelja aplikacije, gdje je definiran pojam i vrste programskih jezika, opisana klasifikacija i usporedba programskih jezika, navedene prednosti i nedostaci korištenja različitih programskih jezika u razvoju aplikacija, te predstavljen C# kao programski jezik koji se koristi u aplikaciji. Zatim, u sljedećem, najopširnijem poglavlju dokumentirana je aplikacija koja je izrađena kao rješenje problema. U istom je poglavlju opisan opis funkcija aplikacije koje omogućuju operacije sa vlasnicima robe, operacije sa robom, košaricu i evidentiranje prodaje robe, pretragu vlasnika u sustavu, prikaz tablice iz baze podataka i operacije sa bazom podataka. Također, opisana je baza podataka i modeli podataka koji predstavljaju strukturu podataka o robi, vlasnicima robe i trgovini. Nadalje, opisano je korisničko sučelje i programski kod koji omogućuju interakciju sa aplikacijom. Na kraju poglavlja je prikazan rad aplikacije na primjerima korištenja. Konačno, rad završava zaključkom, u kojem je sažeto što je postignuto radom, koje su moguća proširenja aplikacije u budućnosti te koji su mogući problemi u izradi projekta.

2. Opis problema

2.1. Definiranje i analiza problema

Trgovina rabljenom robom vrsta je poslovanja koja se bavi kupovinom i prodajom robe koja je već korištena od strane drugih osoba, te ima poteškoća s praćenjem, evidentiranjem spremanja stanja robe i vlasnika robe u trgovini. Ovo je problem jer trgovina mora znati koliko robe ima na raspolaganju za prodaju, koja je roba dostupna, koja je roba naplaćena, koja je roba vraćena vlasniku ili odbačena, te koliko duguje vlasnicima robe koji su ostavili svoju robu na prodaju u trgovini. Također, trgovina mora znati podatke o vlasnicima robe, kao što su ime, prezime i provizija koju dobivaju od prodaje svoje robe. Ako trgovina ne prati i ne evidentira stanje robe i vlasnika robe u trgovini, može doći do gubitka novca, nezadovoljstva kupaca i vlasnika robe, te loše reputacije trgovine.

Trgovina, stoga, traži rješenje kako na jednostavan način, brzo i efikasno pronaći stanje određene robe i vlasnika u određenom trenutku, dodavati, mijenjati i brisati robu i vlasnike. Trgovina traži aplikaciju sa jednostavnim korisničkim sučeljem u kojoj će moći evidentirati sve promjene u stanju robe i vlasnicima. Ta bi aplikacija trebala omogućiti trgovini da na jednostavan način unosi novu robu i njene karakteristike u bazu podataka. Karakteristike robe su: naziv, opis, cijena, dostupnost i naplaćenost robe. Također, trebala bi omogućiti promjenu svih karakteristika robe. Osim robe, aplikacija bi trebala omogućiti unos novih vlasnika robe koji su odlučili ostaviti robu na prodaju u trgovini u određenu proviziju ukoliko do prodaje dođe. Konkretno, trebala bi biti u mogućnosti unijeti ime, prezime i proviziju vlasnika u bazu podataka, kao i izmjenu podataka o vlasniku u bilo kojem trenutku. Osim toga, potrebno je evidentirati kada dođe do plaćanja duga pojedinom vlasniku (njegovog dijela prodaje robe koja ovisi o proviziji pojedinog vlasnika). Konačno, aplikacija bi trebala imati sučelje u kojemu će korisnik na jednostavan način evidentirati prodaju određene robe, konkretno, potrebno je sučelje u kojemu će korisnik aplikacije moći unositi dostupnu robu u košaricu i na kraju evidentirati prodaju robe koja je u košarici, a u skladu s time, potreban je prikaz trenutnog neto profita trgovine i trenutno stanje iznosa u blagajni. Aplikaciju trebaju moći koristiti osoblje na blagajni koje će evidentirati prodaju i u skladištu za unos nove robe i evidenciju o stanju robe.

2.2. Utvrđivanje zahtjeva

Zahtjevi naručitelja aplikacije su slijedeći:

- Potrebna je aplikacija koja će pratiti stanje robe u trgovini. Aplikacija treba omogućiti korisniku da na jednostavan način pomoću korisničkog sučelja unese novu robu i njene karakteristike u bazu podataka. Karakteristike robe su: naziv, opis, cijena, dostupnost i naplaćenost robe. Također treba omogućiti promjenu svih karakteristika robe.
- Osim robe, aplikacija treba omogućiti unos novih vlasnika robe koji su odlučili ostaviti robu na prodaju u trgovini u određenu proviziju ukoliko do prodaje dođe. Treba biti u mogućnosti unijeti ime, prezime i proviziju vlasnika u bazu podataka, kao i izmjenu podataka o vlasniku u bilo kojem trenutku. Treba biti u mogućnosti evidentirati kada dođe do plaćanja duga pojedinom vlasniku (njegovog dijela prodaje robe koja ovisi o proviziji pojedinog vlasnika).
- Aplikacija treba imati sučelje u kojemu će korisnik na jednostavan način evidentirati prodaju određene robe. Potrebno je sučelje u kojemu će korisnik aplikacije moći unositi dostupnu robu u košaricu i na kraju evidentirati prodaju robe koja je u košarici.
- Potreban je prikaz trenutnog neto profita trgovine i trenutno stanje iznosa u blagajni.

- Aplikaciju treba moći koristiti osoblje na blagajni koje će evidentirati prodaju i u skladištu za unos nove robe i evidenciju o stanju robe.

Svrha aplikacije je da trgovini koja se bavi nabavom i prodajom rabljene robe omogući praćenje i evidentiranje sve robe koju trgovina nabavi i proda putem aplikacije sa jednostavnim korisničkim sučeljem koje će omogućiti jednostavno praćenje nabave, prodaje i dostupnosti robe, vlasnika i mijenjanje podataka o istim.

Cilj aplikacije je omogućiti unos robe, njegovih karakteristika (naziva, opisa, cijene, dostupnosti, naplaćenosti i vlasnika robe), mijenjanje karakteristika, pregledavanje robe u sustavu, bilježenje prodaje robe. Također omogućiti unos vlasnika robe u sustav, njegovo ime, prezime i proviziju i mijenjanje istih podataka o vlasniku za vlasnike koji su već u sustavu.

3. Prodaja i maloprodaja

3.1. Definiranje prodaje i maloprodaje

Prodaja i maloprodaja su važni poslovni procesi koji se odnose na razmjenu robe ili usluga za novac ili drugu vrijednost. Prodaja je širi pojam koji obuhvaća sve aktivnosti koje uključuju ponudu, pregovaranje, sklapanje i izvršenje ugovora o kupoprodaji robe ili usluga između prodavatelja i kupca (Šamanović, 2009), dok je maloprodaja uži pojam koji se odnosi na prodaju robe ili usluga krajnjim potrošačima za osobnu upotrebu ili potrošnju (Brčić-Stipčević i Renko, 2007). Maloprodaja se može obavljati u različitim oblicima, kao što su prodavaonice, tržnice, kiosci, automati, internetske trgovine, telefonske narudžbe i druge vrste prodaje izvan prodavaonica (Segetlija i Lamza Maronić, 1999).

Prodaja i maloprodaja značajne su za gospodarski razvoj i društveni napredak jer omogućuju protok robe i novca između proizvođača i potrošača, stvaraju dodanu vrijednost i profit, zadovoljavaju potrebe i želje kupaca, potiču konkurenciju i inovacije, te stvaraju radna mjesta i porezne prihode. Prodaja i maloprodaja su, također, podložne različitim zakonskim i regulatornim okvirima koji utječu na njihovo poslovanje i zaštitu prava sudionika na tržištu. U Republici Hrvatskoj, prodaja i maloprodaja se uređuju Zakonom o trgovini (Narodne novine, br. 87/08-30/14) i Pravilnikom o minimalno tehničkim uvjetima za poslovne prostorije u kojima se obavlja trgovina i posredovanje u trgovini i uvjetima za prodaju robe izvan prodavaonica (Narodne novine, br. 66/2009).

3.2. Razvoj prodaje i maloprodaje

Razvoj prodaje i maloprodaje prati povijest ljudskog društva i njegovih ekonomskih i kulturnih promjena, odnosno, prodaja i maloprodaja su se razvijale u skladu s potrebama i željama ljudi, tehnološkim napretkom, tržišnim uvjetima, zakonskim regulativama i društvenim trendovima. Razvoj prodaje i maloprodaje se može podijeliti u nekoliko faza, koje su:

- razmjena dobara,
- plaćanje novcem,
- prodaja u prodavaonicama te
- prodaja izvan prodavaonica.

Razmjena dobara je najstariji oblik prodaje i maloprodaje koji se temelji na izravnoj razmjeni robe ili usluga između dvije strane bez korištenja novca ili drugog posrednika. Ovaj oblik prodaje i maloprodaje je bio ograničen zbog nedostatka zajedničke mjere vrijednosti, teškoće pronalaženja partnera za razmjenu, neusklađenosti ponude i potražnje, te problema s kvalitetom i količinom robe ili usluge (Šamanović, 2009).

Prodaja uz plaćanje novcem je oblik prodaje i maloprodaje koji se pojavio s uvođenjem novca kao općeg sredstva plaćanja i mjere vrijednosti. Ovaj oblik prodaje i maloprodaje omogućio je lakšu i bržu razmjenu robe ili usluga između strana, te razvoj trgovine na daljinu i međunarodne trgovine. Korištenje novca potaknulo je i razvoj financijskih institucija, tržišta kapitala, kreditiranja i osiguranja (Šamanović, 2009).

Prodaja u prodavaonicama oblik je prodaje i maloprodaje koji se razvio s nastankom gradova i urbanizacije, a temelji se na prodaji robe ili usluga u posebnim poslovnim prostorima koji su namijenjeni za tu svrhu. Ovaj oblik prodaje i maloprodaje je omogućio veći izbor robe i usluga za kupce, te veću konkurenciju među prodavateljima. Ova faza razvoja prodaje potaknula je i razvoj različitih vrsta prodavaonica, kao što su trgovine na veliko, trgovine na malo, specijalizirane trgovine, robne kuće, supermarketi, hipermarketi i druge (Brčić-Stipčević i Renko, 2007).

Prodaja izvan prodavaonica posljednji je oblik prodaje i maloprodaje koji se pojavio s razvojem novih tehnologija komunikacije i prijevoza te se temelji na prodaji robe ili usluga izvan fizičkih poslovnih prostora, putem telefona, pošte, interneta ili drugih kanala. Značajno za ovaj oblik prodaje jest to što je omogućio veću dostupnost robe i usluga za kupce, te niže troškove za prodavatelje. Uz ovaj oblik prodaje razvijale su se i nove vrste prodaje, kao što su direktna prodaja, mrežna prodaja, kataloška prodaja, elektronička trgovina, mobilna trgovina i druge (Segetlija i Lamza Maronić, 1999).

3.3. Trendovi prodaje i maloprodaje

Prodaja i maloprodaja dinamični su procesi koji se stalno mijenjaju i prilagođavaju novim uvjetima i potrebama tržišta te su također odraz društvenih vrijednosti, navika, preferencija i ponašanja kupaca i prodavatelja, što ih čini važnim dijelom gospodarstva i kulture svakog društva.

Trendovi prodaje i maloprodaje odnose se na promjene u načinu, obliku, opsegu i kvaliteti prodaje i maloprodaje koje se događaju u određenom vremenskom razdoblju, a rezultat su različitih čimbenika, kao što su potrebe i želje kupaca, tehnološki napredak, tržišni uvjeti, zakonske regulative i društveni trendovi. Oni utječu na konkurentnost, profitabilnost i održivost trgovaca na malo, te na zadovoljstvo i lojalnost kupaca. Neki od najvažnijih trendova prodaje i maloprodaje su sljedeći:

- porast konkurencije.
- pojava velikih trgovačkih centara,
- propadanje maloprodaje,
- porast ulaganju u tehnologiju te
- globalizacija.

Porast konkurencije ne obuhvaća samo konkurenciju u djelatnosti prodaje, već se trgovci na malo moraju suočiti s konkurencijom iz drugih sektora, kao što su usluge, zabava, putovanja i druge. Također, moraju se suočiti i s konkurencijom iz drugih zemalja, posebno s globalnim divovima e-trgovine, kao što su Amazon, Alibaba i drugi. Trgovci na malo, stoga, moraju biti inovativni, fleksibilni i prilagodljivi kako bi zadržali svoje kupce i privukli nove (Šamanović, 2009).

Pojava i porast velikih trgovačkih centara znači porast broja prodavaonica koje imaju veliku površinu prodajnog prostora, veliki asortiman robe, niske cijene i visoku učinkovitost. To uključuje hipermarkete, megastoreove, outlet centre i druge. Trgovački centri imaju prednost u ekonomiji obujma, pregovaračkoj moći, logistici i marketingu te privlače kupce koji traže povoljnu kupovinu velikog broja proizvoda na jednom mjestu (Brčić-Stipčević i Renko, 2007).

Propadanje maloprodaja na sredini tržišta odnosi se na one maloprodaje koje nemaju jasnu diferencijaciju ni po cijeni ni po kvaliteti robe ili usluge. Maloprodaje na sredini tržišta pod pritiskom su s dvije strane: s jedne strane od trgovačkih centara koji nude niže cijene i veći izbor, a s druge strane od specijaliziranih maloprodaja koje nude višu kvalitetu i bolju uslugu. Stoga, maloprodaje na sredini tržišta moraju pronaći svoju nišu i svoju vrijednost kako bi opstale na tržištu (Brčić-Stipčević i Renko, 2007).

Tehnologija je ključni čimbenik koji utječe na prodaju i maloprodaju u svim aspektima te je povećano ulaganje u razvoj tehnologije očekivano. Ona omogućuje bolje upravljanje podacima, procesima, zalihama, logistikom, financijama i drugim resursima, kao i bolju komunikaciju, interakciju i personalizaciju sa kupcima. Tehnologija također omogućuje nove oblike prodaje i maloprodaje, kao što su e-trgovina, m-

trgovina, i slično. Trgovci na malo, stoga, moraju pratiti tehnološke trendove i ulagati u tehnologiju kako bi poboljšali svoju konkurentnost i profitabilnost (Segetlija i Lamza Maronić, 1999).

Globalizacija je proces koji povezuje ljude, proizvode, usluge, ideje i informacije širom svijeta pa utječe i na prodaju i maloprodaju na više načina. Konkretno, globalizacija omogućuje trgovcima na malo da prošire svoje tržište i svoju ponudu na međunarodnoj razini, no, osim toga, globalizacija donosi nove izazove i prijetnje za trgovce na malo, kao što su različiti zakonski i regulatorni okviri, kulturne i jezične razlike, politička i ekonomska nestabilnost i druge (Segetlija i Lamza Maronić, 1999).

3.4. Komisijska prodaja

Jedan od specifičnih načina prodaje i maloprodaje jest komisijska prodaja. To je način prodaje robe u kojem trgovina preuzima robu od vlasnika koji želi prodati svoju robu, ali ne prima novac unaprijed, već samo dio prodajne cijene nakon što se roba proda. Ovaj način prodaje omogućuje trgovini da ima veći izbor robe i manji rizik od gubitka novca, a vlasniku robe da ostvari neki prihod od svoje robe koju više ne koristi. Komisijska se prodaja najčešće primjenjuje na rabljenu robu koja ima neku vrijednost i potražnju na tržištu, kao što je odjeća, obuća, nakit, knjige, igračke, elektronika i druge stvari. Tržište rabljene robe je u porastu zbog različitih čimbenika, kao što su ekonomska kriza, ekološka svijest, humanitarnost ili potraga za jedinstvenim stilom (Slakoper, 2007).

Komisijska prodaja se temelji na komisijskom ugovoru koji sklapaju trgovina (komisionar) i vlasnik robe (komitent). Komisijski ugovor je ugovor kojim se komisionar obvezuje da će za naknadu (proviziju) prodati robu koju mu je povjerio komitent ili da će obaviti neku drugu pravnu radnju u svoje ime, a za račun komitenta (Zakon o obveznim odnosima). Komisijski ugovor mora biti sklopljen u pisanom obliku i mora sadržavati podatke o strankama, robi, cijeni, proviziji i drugim bitnim odredbama. Prava i obveze strana komisijskog ugovora su regulirana Zakonom o obveznim odnosima i drugim propisima (Slakoper, 2007).

Računovodstvo prodaje robe u komisiji je poseban način evidentiranja prometa robe i novca između trgovine i vlasnika robe, a temelji se na dvojnom knjigovodstvu koje omogućuje praćenje stanja robe i vlasnika robe u trgovini, te izračunavanje profita trgovine i provizije vlasnika robe. Računovodstvo prodaje robe u komisiji se sastoji od nekoliko koraka, kao što su: unos robe i vlasnika robe u evidenciju, određivanje cijene robe i provizije vlasnika robe, izlaganje robe na prodaju, prodaja robe kupcima, isplata vlasniku robe njegovog dijela prodajne cijene, te zatvaranje evidencije za robu koja je prodana ili vraćena vlasniku (Trampus, 2007).

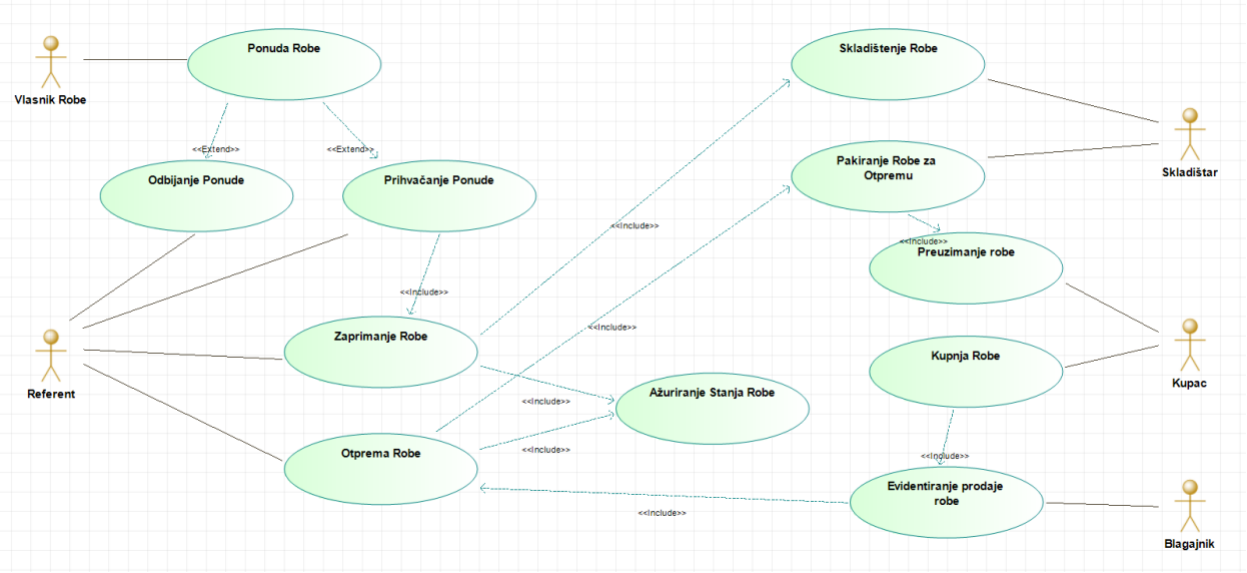
4. Oblikovanje problema

4.1. Dijagram korištenja

Dijagram korištenja je vrsta dijagrama koji se koristi u razvoju softvera za prikazivanje funkcionalnosti i interakcije sustava s njegovim korisnicima, a sastoji se od nekoliko elemenata, kao što su: akteri, slučajevi korištenja, veze i granice sustava. Akteri su vanjski entiteti koji koriste sustav ili na njega utječu, a mogu biti ljudi, organizacije ili drugi sustavi. Slučajevi korištenja su skupovi aktivnosti koje sustav obavlja u suradnji s akterima kako bi postigao neki cilj. Zatim, veze su linije koje povezuju aktere i slučajeve korištenja, a mogu biti različitih vrsta, kao što su: asocijacija, uključivanje, proširenje ili generalizacija. Granice sustava su pravokutnici koji označavaju opseg i kontekst sustava (Sommerville, 2011).

U ovom se radu koristi dijagram korištenja za prikazivanje funkcionalnosti i interakcije aplikacije za trgovinu rabljenom robom koja se bavi komisijskom prodajom. Aplikacija je namijenjena za evidenciju i praćenje stanja robe i vlasnika robe u trgovini, te za obavljanje prodaje robe kupcima. Aplikacija ima pet aktera koji sudjeluju u njenom korištenju: referenta, skladištara, blagajnika, kupca i vlasnika robe. Pritom je referent osoba koja radi u trgovini i koja je zadužena za prihvaćanje ili odbijanje ponude vlasnika robe za prodaju njegove robe u trgovini. Također, referent zaprima robu od vlasnika robe i unosi podatke o robi i vlasniku robe u aplikaciju, a osim toga, otprema robu koja je prodana kupcu i ažurira stanje robe u aplikaciji. Skladištar je osoba koja radi u skladištu trgovine i koja je zadužena za skladištenje robe koja je zaprimljena od vlasnika robe. Osim toga, on pakira robu koja je prodana kupcu i predaje ju kupcu. Zatim, slijedi blagajnik, koji radi na blagajni trgovine i koji je zadužen za evidentiranje prodaje robe kupcima. Blagajnik prima novac od kupaca i isplaćuje vlasnicima robe njihov dio prodajne cijene. Izvan prodavaonice sudjeluju kupac i vlasnik robe, a kupac je osoba koja posjeti trgovinu i koja želi kupiti robu koja je na prodaju, a blagajniku plaća robu koju je kupio te preuzima robu koju je kupio od skladištara. Posljednji je sudionik vlasnik robe koji ima svoju robu koju želi prodati u trgovini. Vlasnik robe može svoju robu ponuditi za prodaju referentu koji će odlučiti hoće li prihvatiti ili odbiti ponudu, a prihvati li referent ponudu, vlasnik robe predaje svoju robu referentu i prima potvrdu o zaprimanju robe. Vlasnik robe također prima novac od blagajnika nakon što se njegova roba proda.

Dijagram korištenja (Slika 1. Dijagram korištenja) prikazuje pet sudionika: referenta, skladištara, blagajnika, kupca i vlasnika robe, kao i osam slučajeva korištenja koji opisuju glavne funkcionalnosti aplikacije: ponuditi robu za prodaju, odbiti ponudu, prihvatiti ponudu, zaprimiti robu, skladištiti robu, kupiti robu, evidentirati prodaju i otpremiti robu. Tim su dijagramom prikazane i veze između aktera i slučajeva korištenja, kao što su: asocijacije koje pokazuju tko koristi koji slučaj korištenja, uključivanja koja pokazuju da se jedan slučaj korištenja sastoji od drugog slučaja korištenja, proširenja koja pokazuju da se jedan slučaj korištenja može proširiti drugim slučajem korištenja pod određenim uvjetima, te generalizacije koje pokazuju da su neki akteri podvrste drugih aktera.



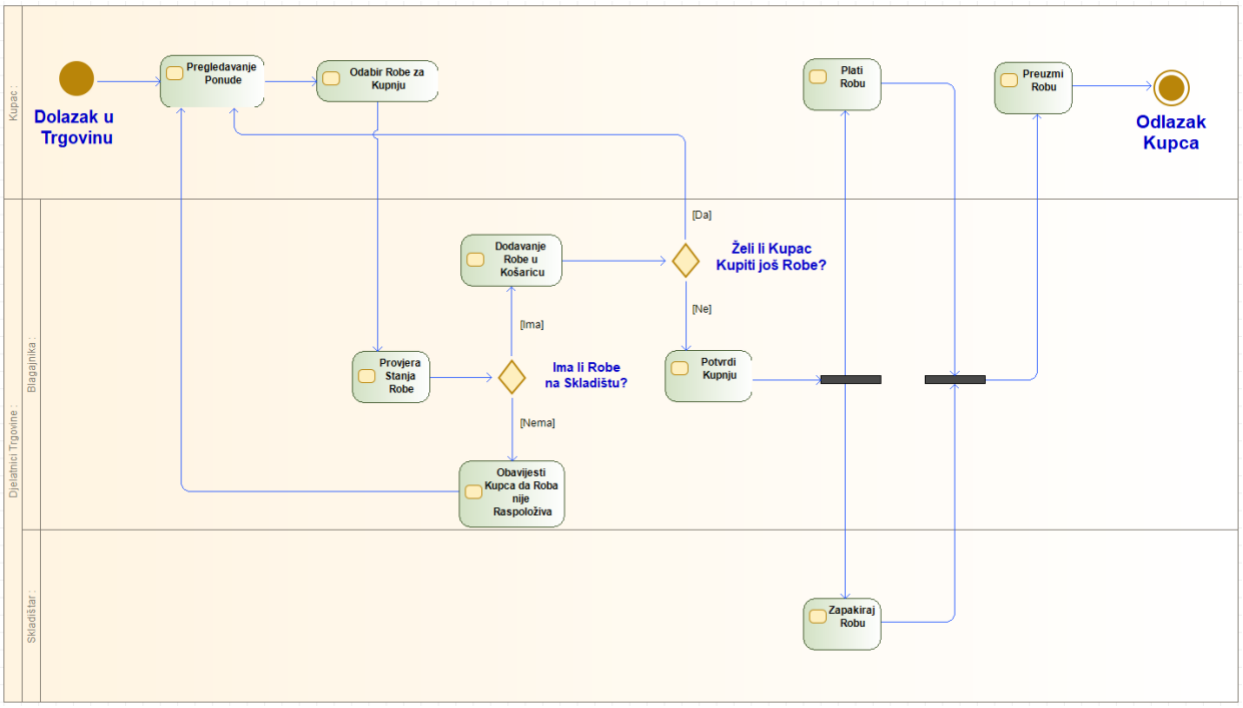
Slika 1. Dijagram korištenja

4.2. Dijagram aktivnosti

Dijagram aktivnosti je vrsta dijagrama koji se koristi u razvoju softvera za prikazivanje toka kontrole i podataka između različitih aktivnosti u sustavu, a čini ga nekoliko elemenata, kao što su: čvorovi, bridovi, particije i objekti. Čvorovi su grafički simboli koji predstavljaju aktivnosti, događaje, odluke ili krajnje točke u sustavu. Bridovi su linije koje povezuju čvorove i pokazuju tok kontrole ili podataka između njih. Zatim, particije su vertikalne linije koje dijele dijagram na segmente i pokazuju odgovornosti različitih aktera ili komponenata u sustavu. Konačno, objekti su pravokutnici koji predstavljaju podatke ili entitete koji se koriste ili stvaraju u sustavu (Sommerville, 2011). U ovom je radu korišten dijagram aktivnosti za prikazivanje toka kontrole i podataka između različitih aktivnosti u aplikaciji za trgovinu rabljenom robom koja se bavi komisijskom prodajom. Aplikacija je namijenjena za evidenciju i praćenje stanja robe i vlasnika robe u trgovini, te za obavljanje prodaje robe kupcima, a ima tri aktera koji sudjeluju u njenom korištenju: kupca, blagajnika i skladištara.

Dijagram aktivnosti (Slika 2. Dijagram aktivnosti) prikazuje tri sudionika: kupca, blagajnika i skladištara, a prikazuje i dvanaest čvorova koji opisuju glavne aktivnosti u aplikaciji: doći u trgovinu, pregledati ponudu, odabrati robu za kupnju, provjeriti stanje robe, obavijestiti kupca, dodati robu u košaricu, potvrditi kupnju, platiti robu, pakirati robu, preuzeti robu i otići iz trgovine. Dijagram aktivnosti također prikazuje bridove koji pokazuju tok kontrole i podataka između čvorova, kao što su: kontrolni tok koji pokazuje redoslijed izvršavanja aktivnosti, tok objekta koji pokazuje prijenos podataka između aktivnosti, tok signala koji pokazuje slanje ili primanje signala između aktivnosti, te tok uvjeta koji pokazuje ishod odluke na temelju nekog uvjeta. Dijagram aktivnosti također prikazuje particije koje pokazuju odgovornosti kupca, blagajnika i skladištara u sustavu. Osim toga, prikazani su i objekti koji predstavljaju podatke o robi i košarici u sustavu.

Proces kupnje počinje kada kupac dođe u trgovinu. Kupac pregledava ponudu i odabire robu za kupnju. Blagajnik zatim provjerava stanje robe. Ukoliko nema robe na skladištu, obavještava kupca koji dalje pregledava ponudu, odabire robu i zatim blagajnik opet provjerava stanje odabrane robe. Ukoliko ovaj put ima robe na skladištu, blagajnik dodaje robu u košaricu i pita kupca želi li kupiti još robe. Ako kupac želi kupiti još robe proces se ponavlja sve dok kupac nije zadovoljan sa svojom košaricom. Zatim blagajnik potvrđuje kupnju, kupac plaća robu, a skladištar pakira robu. Na kraju kupac preuzima robu i odlazi iz trgovine.



Slika 2. Dijagram aktivnosti

4.3. Dijagram slijeda

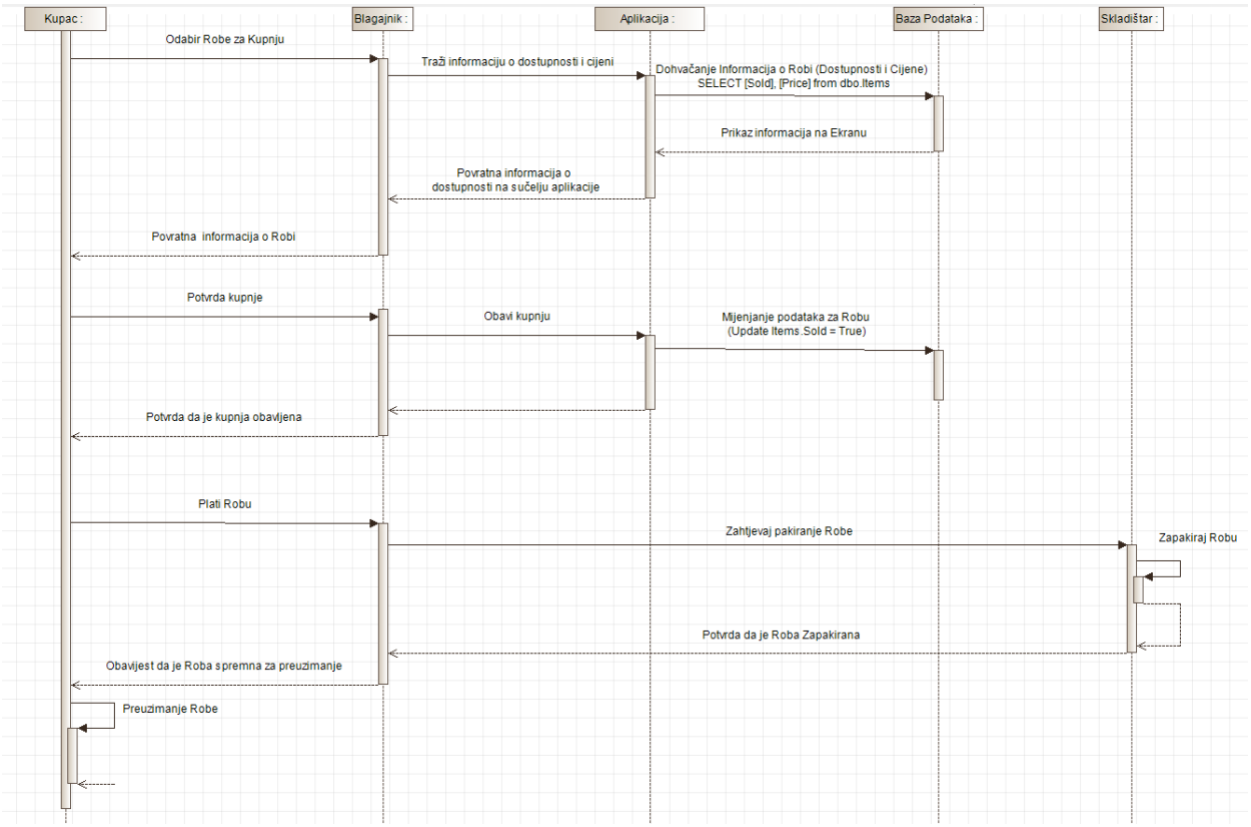
kupac, blagajnik, aplikacija, baza podataka i skladištar. Slijed počinje kada kupac odabire robu za kupnju. Blagajnik zatim traži informaciju o dostupnosti i cijeni od aplikacije. Aplikacija dohvaća informaciju iz baze podataka putem SQL naredbe SELECT. Baza podataka prikazuje stanje robe na sučelju aplikacije. Blagajnik čita podatke i daje kupcu povratnu informaciju o robi. Kupac zatim potvrđuje kupnju, referent obavlja prodaju putem aplikacije, a aplikacija mijenja podatke, dostupnost robe (UPDATE) u bazi podataka. Kupac zatim plaća robu, a blagajnik zahtijeva pakiranje robe od skladištara. Skladištar pakira robu i obavještava referenta da je roba spremna za preuzimanje. Kupac na kraju preuzima robu i odlazi iz trgovine.

Dijagram slijeda vrsta je dijagrama koji se koristi u razvoju softvera za prikazivanje interakcije između objekata u sustavu u određenom vremenskom redoslijedu, a sastoji se od nekoliko elemenata, kao što su: objekti, poruke, aktivacijske trake i vremenske oznake. Detaljnije, objekti su entiteti koji sudjeluju u interakciji i mogu biti ljudi, organizacije, komponente ili drugi sustavi. Nakon toga, poruke su komunikacije koje se šalju ili primaju između objekata i mogu biti sinkrone, asinkrone ili povratne, a aktivacijske trake su uspravne linije koje pokazuju životni ciklus objekta i njegovu aktivnost u sustavu. Posljednji element, vremenske oznake su horizontalne linije koje pokazuju redoslijed i trajanje poruka u sustavu (Sommerville, 2011).

U ovom su radu dijagramom slijeda prikazane interakcije između objekata u aplikaciji za trgovinu rabljenom robom koja se bavi komisijskom prodajom. Aplikacija je namijenjena za evidenciju i praćenje stanja robe i vlasnika robe u trgovini, te za obavljanje prodaje robe kupcima. Aplikacija ima pet objekata koji sudjeluju u njenom korištenju: kupca, blagajnika, aplikaciju, bazu podataka i skladištara.

Dijagram slijeda (Slika 3. Dijagram Slijeda - Kupac) prikazuje slučaj u kojemu sudjeluje 5 sudionika: kupac, blagajnik, aplikacija, baza podataka i skladištar. Dijagram slijeda također prikazuje dvanaest poruka koje opisuju komunikaciju između objekata u sustavu: odabrati robu za kupnju, tražiti informaciju o dostupnosti i cijeni, dohvaćati informaciju iz baze podataka, prikazivati stanje robe na sučelju aplikacije, čitati podatke i davati povratnu informaciju o robi, potvrđivati kupnju, plaćati robu, mijenjati podatke u bazi podataka, zahtijevati pakiranje robe, pakirati robu, obavještavati da je roba spremna za preuzimanje i preuzimati robu. Dijagram slijeda također prikazuje aktivacijske trake koje pokazuju kada i koliko dugo su objekti aktivni u sustavu. Dijagram slijeda također prikazuje vremenske oznake koje pokazuju redoslijed i trajanje poruka u sustavu.

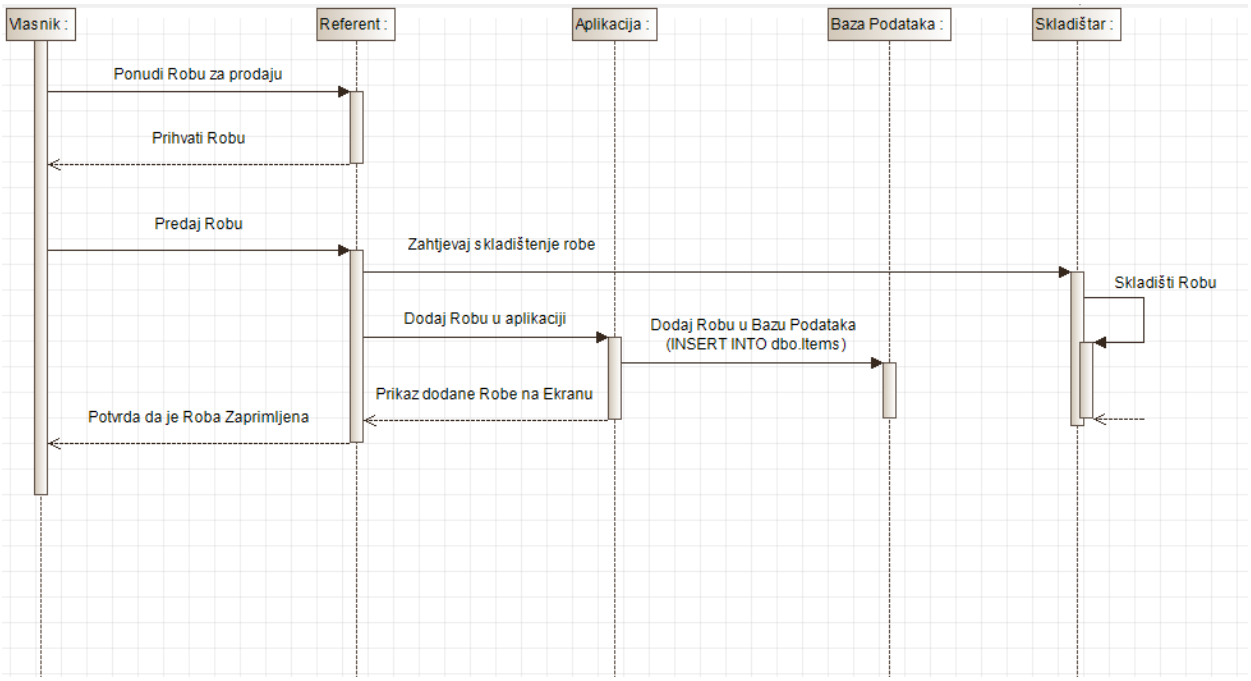
Proces kupnje započinje kada kupac odabire robu za kupnju. Blagajnik, zatim, traži informaciju o dostupnosti i cijeni od aplikacije. Aplikacija dohvaća informaciju iz baze podataka, putem SQL naredbe SELECT, a baza podataka prikazuje stanje robe na sučelju aplikacije. Blagajnik čita dobivene podatke i daje kupcu povratnu informaciju o robi. Kupac zatim potvrđuje kupnju, blagajnik obavlja prodaju putem aplikacije, a aplikacija mijenja podatke o dostupnosti robe (UPDATE) u bazi podataka. Kupac zatim plaća robu, a blagajnik zahtijeva pakiranje robe od skladištara. Skladištar pakira robu i obavještava blagajnika da je roba spremna za preuzimanje. Kupac na kraju preuzima robu i odlazi iz trgovine.



Slika 3. Dijagram Slijeda - Kupac

Dijagram slijeda (Slika 4. Dijagram slijeda - Vlasnik) prikazuje slučaj u kojemu sudjeluje pet objekata: vlasnik robe, referent, aplikacija, baza podataka i skladištar. Dijagram slijeda, također, prikazuje deset poruka koje opisuju komunikaciju između objekata u sustavu: nuditi robu za prodaju, prihvaćati robu, predavati robu, zahtijevati skladištenje robe, zaprimati robu u aplikaciji, unositi robu u bazu podataka, potvrđivati da je roba spremljena u bazi podataka, prikazivati robu na ekranu aplikacije, potvrđivati da je roba zaprimljena i primati potvrdu o zaprimanju robe. Dijagram slijeda prikazuje i aktivacijske trake koje pokazuju kada i koliko dugo su objekti aktivni u sustavu. Osim toga, prikazane su i vremenske oznake koje pokazuju redoslijed i trajanje poruka u sustavu.

Proces počinje kada vlasnik nudi svoju robu za prodaju u trgovini. Referent prihvaća robu, ako je zadovoljan njenom kvalitetom i cijenom. Vlasnik predaje svoju robu referentu i prima potvrdu o zaprimanju robe. Zatim, referent zahtjeva od skladištara skladištenje robe na sigurno mjesto. Skladištar preuzima robu od referenta i odlaže ju u skladište. Nakon toga, referent zaprima robu u aplikaciji i unosi podatke o robi i vlasniku robe u aplikaciju. Aplikacija unosi robu u bazu podataka putem SQL naredbe INSERT. Baza podataka potvrđuje da je roba spremljena u bazi podataka i vraća podatke o robi aplikaciji. Aplikacija zatim prikazuje robu na ekranu aplikacije i omogućuje referentu da pregleda i izmijeni podatke o robi ako je potrebno. Referent potvrđuje da je roba zaprimljena i obavještava vlasnika robe da je njegova roba uspješno zaprimljena i spremljena u trgovini.



Slika 4. Dijagram slijeda - Vlasnik

4.4. Dijagram klasa

U slučaju prikazanom ovim radom postoji pet klasa: Trgovina, Roba, Vlasnik, Kupac i Košarica.

Trgovina ima sljedeća svojstva:

- IdTrgovine kao integer (cijeli broj) koji predstavlja identifikacijski broj trgovine
- Naziv kao string (tekst) koji predstavlja naziv trgovine
- NetoProfit kao double (decimalan broj) koji predstavlja neto profit trgovine
- IznosBlagajne kao double (decimalan broj) koji predstavlja iznos u blagajni

Vlasnik ima sljedeća svojstva:

- IdVlasnika kao integer (cijeli broj) koji predstavlja identifikacijski broj vlasnika
- Ime kao string (tekst) koji predstavlja ime vlasnika
- Prezime kao string (tekst) koji predstavlja prezime vlasnika
- Provizija kao double (decimalan broj) koji predstavlja proviziju vlasnika
- Nepodmireni Iznos kao double (decimalan broj) koji predstavlja dug trgovine prema vlasniku

Vlasnika možemo dodavati putem metode: DodajVlasnika(), možemo izmjenjivati podatke o vlasniku putem metode IzmjeniVlasnika() i brisati vlasnika putem metode: IzbrišiVlasnika()

Roba ima sljedeća svojstva:

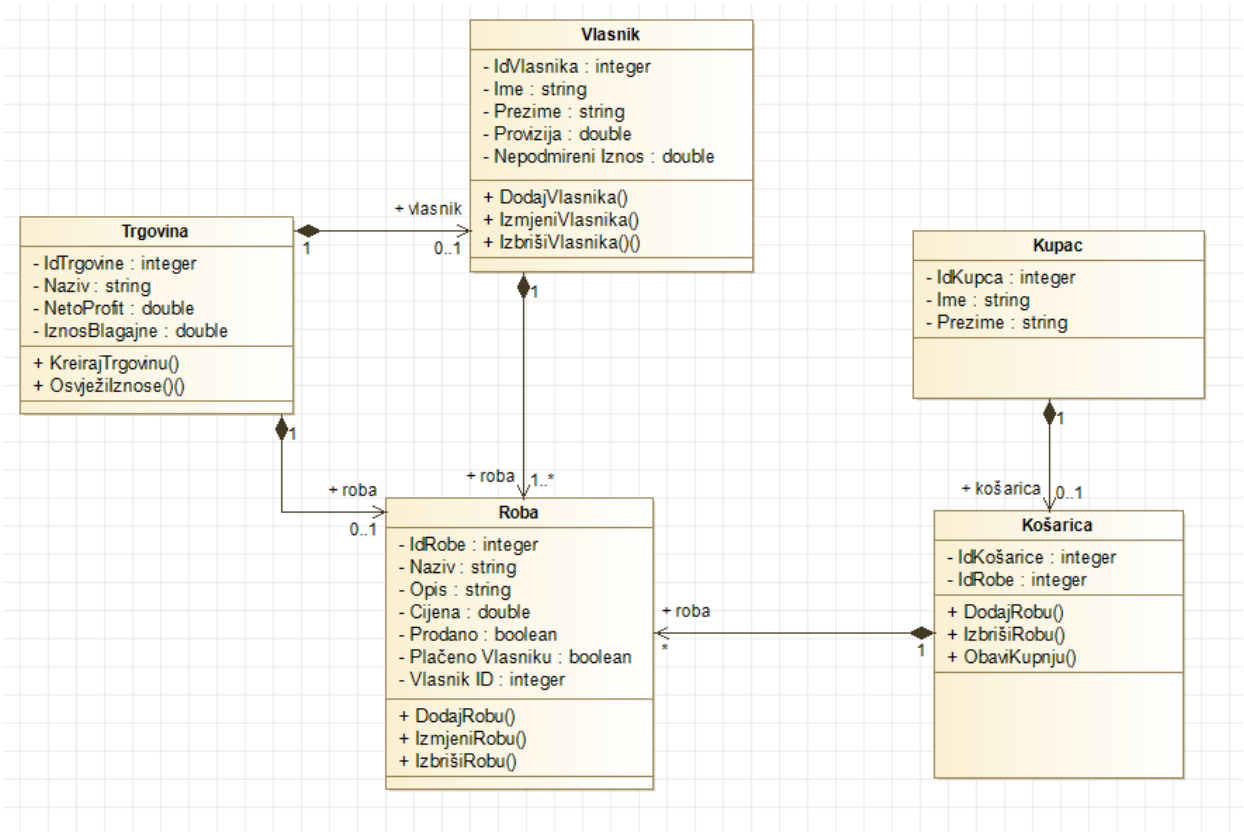
- IdRobe kao integer (cijeli broj) koji predstavlja identifikacijski broj robe
- Naziv kao string (tekst) koji predstavlja naziv robe
- Opis kao string (tekst) koji predstavlja opis robe
- Cijena kao double (decimalan broj) koji predstavlja cijenu robe
- Prodano kao boolean (je ili nije) koja predstavlja je li roba prodana (dostupna) ili nije
- Plaćeno Vlasniku kao boolean (je ili nije) koji nam govori je li plaćeno vlasniku robe ili nije
- IDVlasnika kao integer (cijeli broj) koji je primarni ključ u tablici Vlasnika i govori nam kojem vlasniku roba pripada

Robu možemo dodavati putem metode: DodajRobu(), možemo izmjenjivati podatke o robe putem metode: IzmjeniRobu() i brisati robu putem metode: IzbrišiRobu()

Košarica ima sljedeća svojstva:

- IdKošarice kao integer (cijeli broj) koji predstavlja identifikacijski broj košarice
- IdRobe kao integer (cijeli broj) koji je primarni ključ u tablici Robe i govori nam koja roba se nalazi u košarici

U košaricu možemo dodavati robu putem metode DodajRobu(), možemo brisati robu iz košarice putem metode: IzbrišiRobu(), i možemo potvrditi kupnju robe u košarici putem metode: Naplata()



Slika 5. Dijagram Klasa

5. Baze podataka

5.1. Pojam i vrste baza podataka

Pojam i vrste baza podataka važni su koncepti u području informatike i informacijskih sustava, a predstavljaju baze podataka koji su organizirani i pohranjeni na način koji omogućuje njihovo jednostavno i učinkovito pretraživanje, ažuriranje, analizu i obradu (Maleković i Rabuzin, 2016). Podaci su osnovne jedinice informacija koje predstavljaju neku činjenicu, pojavu ili događaj u stvarnom svijetu (Kramberger, Duk i Kovačević, 2018). Podaci se mogu klasificirati prema različitim kriterijima, kao što su: vrsta, struktura, format, kvaliteta, vrijednost i druge karakteristike (Cheng, Liu i Yao, 2017). Baze podataka se mogu podijeliti u različite vrste prema različitim kriterijima, kao što su: model podataka, način pristupa, lokacija pohrane, distribucija i druge karakteristike (Elmasri i Navathe, 2015). Model podataka apstraktna je reprezentacija strukture i odnosa podataka u bazi podataka koja određuje način organizacije, manipulacije i upravljanja podacima (Abiteboul, Hull i Vianu, 1995). Način pristupa je način komunikacije s bazom podataka koji omogućuje definiranje, dohvaćanje, ažuriranje i brisanje podataka u bazi podataka (Melton, 2016). Lokacija pohrane je mjesto gdje se podaci fizički nalaze i mogu biti na jednom ili više uređaja ili medija (Gunjal, 2003). Distribucija je način raspodjele podataka između više lokacija ili sustava koji omogućuje veću dostupnost, pouzdanost i performanse baze podataka (Hoffer, Prescott i McFadden, 2007).

Prema modelu podataka, baze podataka se mogu podijeliti u tri glavne vrste: hijerarhijske, mrežne i relacijske baze podataka (Darwen, 2010). Hijerarhijske baze podataka su one koje organiziraju podatke u stablastu strukturu gdje svaki čvor ima jednog roditelja i nula ili više djece. Mrežne baze podataka su one koje organiziraju podatke u grafu gdje svaki čvor može imati više roditelja i više djece. Zatim, relacijske baze podataka su one koje organiziraju podatke u tablice gdje svaki redak predstavlja jedan zapis ili entitet, a svaki stupac predstavlja jedno svojstvo ili atribut. Te su baze podataka najčešće korištene vrste baza podataka zbog svoje jednostavnosti, fleksibilnosti i efikasnosti (Darwen, 2010).

Prema načinu pristupa, baze podataka mogu se podijeliti na dvije glavne vrste: navigacijske i deklarativne baze podataka (Abiteboul i sur., 1995). Navigacijske baze podataka su one koje zahtijevaju od korisnika da navede putanju ili redoslijed pristupa do željenih podataka u bazi podataka, a deklarativne baze podataka su one koje omogućuju korisniku da navede samo željeni rezultat ili uvjet pristupa bez navođenja putanje ili redoslijeda pristupa te su one pogodnije za korisnike koji nisu upoznati s unutarnjom strukturom ili organizacijom baze podataka (Abiteboul i sur., 1995).

Prema lokaciji pohrane, baze podataka se mogu podijeliti u dvije glavne vrste: centralizirane i decentralizirane baze podataka (Gunjal, 2003). Centralizirane baze podataka su one koje pohranjuju sve podatke na jednom mjestu ili uređaju, dok su decentralizirane baze podataka one koje pohranjuju podatke na više mjesta ili uređaja. Decentralizirane baze podataka se mogu dalje podijeliti u distribuirane i replicirane baze podataka. Distribuirane baze podataka pohranjuju različite dijelove podataka na različitim mjestima ili uređajima, a replicirane baze podataka pohranjuju iste ili slične kopije podataka na različitim mjestima ili uređajima (Gunjal, 2003).

Prema distribuciji, baze podataka se mogu podijeliti u dvije glavne vrste: homogene i heterogene baze podataka (Hoffer i sur., 2007). Homogene baze podataka koriste isti model podataka, način pristupa, lokaciju pohrane i softver za upravljanje bazom podataka na svim mjestima ili sustavima koji sudjeluju u distribuciji, dok heterogene baze podataka koriste različite modele podataka, načine pristupa, lokacije

pohrane i softvere za upravljanje bazom podataka na različitim mjestima ili sustavima koji sudjeluju u distribuciji. Heterogene baze podataka su složenije i zahtjevnije za integraciju i koordinaciju od homogenih baza podataka (Hoffer i sur., 2007).

5.2. Modeliranje i dizajniranje baza podataka

Modeliranje i dizajniranje baza podataka važni su procesi u razvoju informacijskih sustava koji omogućuju efikasno i kvalitetno upravljanje podacima. Modeliranje i dizajniranje baza podataka uključuju različite faze i tehnike koje se primjenjuju na različitim razinama apstrakcije i detalja (Kovačević, 2018).

Modeliranje baza podataka je proces u kojem se definira struktura, odnosi i ograničenja podataka u bazi podataka pomoću određenog modela podataka (Abiteboul, Hull i Vianu, 1995). Model podataka je skup pravila i konvencija koji opisuju način predstavljanja i manipulacije podacima u bazi podataka (Darwen, 2010), a više je različitih modela podataka koji se koriste za modeliranje baza podataka, poput hijerarhijskog, mrežnog, relacijskog, objektnog, objektno-relacijskog, XML, NoSQL i drugih modela (Elmasri i Navathe, 2015). Svaki model podataka ima svoje prednosti i nedostatke, te se odabire prema potrebama i zahtjevima informacijskog sustava.

Dizajniranje baza podataka proces je u kojem se implementira model baze podataka na određenom sustavu za upravljanje bazom podataka pomoću određenog jezika za definiciju podataka (Hoffer, Prescott i McFadden, 2007). Sustav za upravljanje bazom podataka je softver koji omogućuje stvaranje, održavanje i korištenje baze podataka (Maleković i Rabuzin, 2016). Jezik za definiciju podataka je jezik koji omogućuje definiranje strukture, odnosa i ograničenja podataka u bazi podataka (Melton, 2016). Postoje različiti sustavi za upravljanje bazom podataka, kao i jezici za definiciju podataka koji se koriste za dizajniranje baza podataka, kao što su: Oracle, MySQL, PostgreSQL, SQL Server, MongoDB, Cassandra, Neo4j i dr. (Varga, 2012). Svaki sustav za upravljanje bazom podataka i jezik za definiciju podataka ima svoje karakteristike i mogućnosti, te se odabire prema performansama i funkcionalnostima informacijskog sustava.

Modeliranje i dizajniranje baza podataka uglavnom se provode u tri glavne faze: konceptualna, logička i fizička (Kramberger, Duk i Kovačević, 2018). Pritom je konceptualna faza ona faza u kojoj se analiziraju potrebe i zahtjevi korisnika informacijskog sustava te se izrađuje konceptualni model baze podataka koji prikazuje glavne entitete, atribute i veze u bazi podataka na visokoj razini apstrakcije. Konceptualni model baze podataka prikazuje se pomoću entitetsko-relacijskog dijagrama ili unificiranog modela klasa (Korbar, 2010). Slijedi logička faza u kojoj se transformira konceptualni model baze podataka u logički model baze podataka koji prikazuje strukturu, odnose i ograničenja podataka u bazi podataka na srednjoj razini apstrakcije. Logički se model baze podataka obično prikazuje pomoću relacijske sheme ili objektnog modela (Gunjal, 2003). Posljednja, odnosno fizička faza je faza u kojoj se implementira logički model baze podataka u fizički model baze podataka koji prikazuje pohranu, indeksiranje i pristup podacima u bazi podataka na niskoj razini apstrakcije. Fizički model baze podataka uglavnom se prikazuje pomoću SQL naredbi ili drugih jezika za definiciju podataka (Cheng, Liu i Yao, 2017).

Modeliranje i dizajniranje baza podataka složeni su i zahtjevni procesi koji zahtijevaju dobro poznavanje teorije i prakse baza podataka, kao i vještine analize, sinteze, komunikacije i kreativnosti. Modeliranje i dizajniranje baza podataka imaju veliki utjecaj na funkcionalnost, kvalitetu, sigurnost i efikasnost informacijskog sustava, te stoga zahtijevaju pažljivo planiranje, provjeru i testiranje (Manager, 2012).

5.3. Prednosti i nedostaci korištenja baza podataka u razvoju aplikacija

Prednosti i nedostaci korištenja baza podataka u razvoju podataka važna su tema koja se tiče svih koji se bave obradom, analizom i vizualizacijom podataka. Baze podataka su skupovi podataka koji su organizirani i pohranjeni na način koji omogućuje njihovo jednostavno i učinkovito pretraživanje, ažuriranje, analizu i obradu (Maleković i Rabuzin, 2016). Razvoj podataka je proces u kojem se prikupljaju, transformiraju, integriraju, modeliraju i vizualiziraju podaci radi stvaranja vrijednih informacija i znanja za donošenje odluka (Cheng, Liu i Yao, 2017).

Korištenje baza podataka u razvoju podataka ima brojne prednosti, ali i neke nedostatke. Prednosti korištenja baza podataka u razvoju podataka su:

- povećana produktivnost i efikasnost,
- poboljšana kvaliteta i pouzdanost te
- povećana sigurnost i zaštita.

Baze podataka omogućuju brz i jednostavan pristup velikim količinama podataka iz različitih izvora i formata, što olakšava procese prikupljanja, transformacije, integracije i modeliranja podataka, smanjujući vrijeme i napor potreban za razvoj podataka. Također, baze podataka pružaju različite alate i funkcije za analizu i obradu podataka, kao što su SQL upiti, agregacijske funkcije, statističke funkcije, strojno učenje i dr. (Melton, 2016). Konačno, baze podataka omogućuju stvaranje složenih i kvalitetnih informacija i znanja iz podataka. Osim toga, baze podataka nude mehanizme za osiguranje kvalitete i pouzdanosti podataka. Oni provode ograničenja podataka, kao što su jedinstvene vrijednosti, referentni integritet, domene vrijednosti i dr. (Abiteboul i sur., 1995) što sprječava nedosljednosti, pogreške i duplikate u podacima. Također, baze podataka omogućuju provjeru valjanosti i točnosti podataka pomoću različitih metoda, kao što su validacija ulaza, provjera pravopisa, provjera ispravnosti formata i dr., a to jamči da podaci pohranjeni u bazi podataka ostaju točni, relevantni i ažurni (Hoffer i sur., 2007). Zatim, baze podataka nude mehanizme za osiguranje sigurnosti i zaštite podataka, koji omogućuju kontrolu pristupa podacima pomoću različitih metoda, kao što su autentifikacija korisnika, autorizacija korisnika, enkripcija podataka, revizija aktivnosti i dr. (Elmasri i Navathe, 2015), a to sprječava neovlašteni pristup ili zlouporabu podataka od strane neovlaštenih osoba ili entiteta. Također, baze podataka omogućuju sigurnosno kopiranje i obnavljanje podataka pomoću različitih metoda, kao što su puno sigurnosno kopiranje, diferencijalno sigurnosno kopiranje, inkrementalno sigurnosno kopiranje, transakcijsko sigurnosno kopiranje i dr. (Korbar, 2010).

Nedostaci korištenja baza podataka u razvoju podataka su:

- visoki troškovi i složenost,
- ograničena fleksibilnosti skalabilnost te
- potencijalni rizici i izazovi.

Baze podataka zahtijevaju visoke troškove i složenost za njihovu implementaciju i održavanje, konkretno, one zahtijevaju nabavu i instalaciju hardvera i softvera, kao što su serveri, diskovi, mrežni uređaji, sustavi za upravljanje bazom podataka, alate za razvoj podataka i dr. (Varga, 2012). Također, zahtijevaju zapošljavanje i obuku kvalificiranog osoblja, poput administratora baza podataka, programera baza podataka, analitičara podataka i dr. (Kovačević, 2018). Sve to, u konačnici, predstavlja značajan financijski i ljudski resurs za tvrtke i organizacije koje se bave razvojem podataka.

Unatoč mnogim prednostima i mogućnostima korištenja, baze podataka mogu biti ograničene u pogledu fleksibilnosti i skalabilnosti za prilagodbu promjenjivim potrebama i zahtjevima razvoja podataka. Baze podataka može biti teško promijeniti ili proširiti nakon što su implementirane, jer to potencijalno predstavlja značajne promjene u strukturi, odnosima i ograničenjima podataka (Darwen, 2010). Osim toga, proces skaliranja ili distribucije baza podataka na više lokacija ili sustava može biti izrazito složeno, jer to može zahtijevati dodatne resurse i koordinaciju (Gunjal, 2003).

Posljednje, baze podataka mogu biti izložene potencijalnim rizicima i izazovima koji mogu utjecati na njihovu funkcionalnost i kvalitetu, jer su podložne napadima ili prijetnjama od strane hakera, virusa, crva, trojanaca ili drugih zlonamjernih programa koji mogu ukrasti, oštetiti ili uništiti podatke (Cheng i sur., 2017). Osim napada, baze podataka mogu biti pogođene kvarovima ili katastrofama, kao što su požari, poplave, potresi ili nestanci struje koji mogu uzrokovati gubitak ili nedostupnost podataka (Kramberger i sur., 2018). Još je jedna prijetnja, potencijalno suočavanje s pravnim ili etičkim pitanjima, kao što su privatnost, sigurnost, vlasništvo ili regulacija podataka koji mogu ograničiti ili zabraniti korištenje ili dijeljenje podataka (Manager, 2012).

5.4. Microsoft SQL baze podataka

Microsoft SQL baze podataka vrsta su relacijskih baza podataka koje koriste Microsoft SQL Server kao sustav za upravljanje bazom podataka. Microsoft SQL Server je softver koji omogućuje stvaranje, održavanje i korištenje baza podataka na Windows platformi. Microsoft SQL baze podataka koriste Transact-SQL (T-SQL) kao jezik za definiciju, manipulaciju i upravljanje podacima u bazi podataka (Melton, 2016).

Brojne su mogućnosti i prednosti Microsoft SQL baza podataka koje ih čine popularnim izborom za razvoj podataka i informacijskih sustava. Neke od tih mogućnosti i prednosti su:

- jednostavno korištenje i integracija,
- visoka učinkovitost,
- visoka kvaliteta i pouzdanost te
- visoka sigurnost i zaštita.

Microsoft SQL baze podataka nude korisničko sučelje i alate koji iznimno olakšavaju rad s bazom podataka, kao što su SQL Server Management Studio, Visual Studio, SQL Server Integration Services, SQL Server Reporting Services i dr. Također, Microsoft SQL baze podataka se lako mogu integrirati s drugim Microsoft proizvodima i tehnologijama, kao što su Windows, Office, .NET Framework, Azure i dr. (Varga, 2012). Zatim, Microsoft SQL baze podataka omogućuju brz i jednostavan pristup velikim količinama podataka iz različitih izvora i formata, te pružaju različite funkcije i mogućnosti za analizu i obradu podataka, kao što su agregacijske funkcije, statističke funkcije, strojno učenje, data mining i dr. (Melton, 2016). Visoka kvaliteta i pouzdanost zajamčeni su jer Microsoft SQL baze podataka nude mehanizme za osiguranje kvalitete i pouzdanosti podataka, te provode ograničenja podataka, kao što su jedinstvene vrijednosti, referentni integritet, domene vrijednosti i dr. (Abiteboul i sur., 1995). Ove mogućnosti sprječavaju nedosljednosti, pogreške i duplikate u podacima. Također, Microsoft SQL baze podataka omogućuju provjeru valjanosti i točnosti podataka pomoću različitih metoda, kao što su validacija ulaza, provjera pravopisa, provjera ispravnosti formata i dr. (Hoffer i sur., 2007). Zatim, s namjerom osiguravanja sigurnosti i zaštite podataka, Microsoft SQL baze podataka nude mehanizme za osiguranje sigurnosti i zaštite podataka, koji omogućuju kontrolu pristupa podacima pomoću različitih metoda, kao

što su autentifikacija korisnika, autorizacija korisnika, enkripcija podataka, revizija aktivnosti i dr. (Elmasri i Navathe, 2015). Sve to sprječava neovlašteni pristup ili zlouporabu podataka od strane neovlaštenih osoba ili entiteta. Osim toga, Microsoft SQL baze podataka omogućuju sigurnosno kopiranje i obnavljanje podataka pomoću različitih metoda, kao što su puno sigurnosno kopiranje, diferencijalno sigurnosno kopiranje, inkrementalno sigurnosno kopiranje, transakcijsko sigurnosno kopiranje i dr. (Korbar, 2010).

6. Programski jezici

6.1. Pojam i vrste programskih jezika

Pojam i vrste programskih jezika važni su koncepti u području računarstva i informatike. Programski jezik je skup simbola i pravila kojima se opisuje postupak računanja, odnosno algoritam ili program koji se može izvršiti na računalu (Lovreničić i Konecki, 2017). Uloga programskog jezika jest da služi kao sredstvo komunikacije između čovjeka i računala, te da omogući stvaranje različitih vrsta softvera, aplikacija i informacijskih sustava. Postoji mnogo vrsta programskih jezika koji se razlikuju po svojoj sintaksi, semantici, paradigmi, namjeni i mogućnostima. Vrste programskih jezika se mogu klasificirati prema različitim kriterijima, kao što su:

- razina apstrakcije,
- generacija te
- paradigma.

Razina apstrakcije je kriterij koji određuje koliko je programski jezik blizak strojnom jeziku (niski) ili ljudskom jeziku (visoki). Niski programski jezici su brži i precizniji, ali teži za učenje i pisanje. Primjeri niskih programskih jezika su asembler i C. Visoki programski jezici su lakši za učenje i pisanje, ali sporiji i manje efikasni, to su primjerice Python i Java (Kovačević i Stojanović, 2022).

Generacija je kriterij koji određuje koliko je programski jezik razvijen u povijesti računarstva. Prva generacija su strojni jezici koji se sastoje od binarnih kodova. Zatim, druga generacija su asembleri koji koriste simboličke oznake za naredbe. Slijedi treća generacija programskih jezika, koju predstavljaju viši programski jezici koji koriste riječi i znakove slične engleskom jeziku. Četvrta su generacija specijalizirani programski jezici koji omogućuju brzo rješavanje određenih problema, a posljednja, peta generacija programskih jezika odnosi se na inteligentne programske jezike koji koriste logiku i umjetnu inteligenciju (Lovrenčić i sur., 2009).

Paradigma je kriterij koji određuje način na koji se programski jezik konceptualizira i strukturira. Postoje različite programske paradigme koje utječu na način pisanja i izvršavanja programa, a neke od najpoznatijih paradigmi su: imperativna, deklarativna, funkcionalna, objektno orijentirana, logička, proceduralna i dr. (Kovačević i Stojanović, 2022).

6.2. Klasifikacija i usporedba programskih jezika

Tablica ispod (Tablica 1. Usporedba programskih jezika C, C++, C#, Java i Python) prikazuje usporedbu programskih jezika C, C++, C#, Java i Python, koji su svi bazirani na sintaksi jezika C, ali imaju različite osobine i mogućnosti.

Tablica 1. Usporedba programskih jezika C, C++, C#, Java i Python

Kriterij	C	C++	C#	Java	Python
Sintaksa	Ima jednostavnu i strogu sintaksu koja se sastoji od ključnih riječi, operatora, identifikatora i separatora. Sintaksa se temelji na blokovima koda koji su označeni vitičastim zagradama.	Ima složenu i bogatu sintaksu koja proširuje sintaksu jezika C s novim značajkama kao što su klase, predlošci, iznimke, preopterećenje operatora i sl. Sintaksa također podržava više paradigmi programiranja.	Ima jednostavnu i elegantnu sintaksu koja se temelji na sintaksi jezika C++, ali je pojednostavljuje i čini konzistentnijom. Sintaksa također uključuje nove značajke kao što su delegati, događaji, atributi, lambda izrazi i sl.	Ima jednostavnu i jasnu sintaksu koja se temelji na sintaksi jezika C++, ali je čini strožom i sigurnijom. Sintaksa također uključuje nove značajke kao što su sučelja, unutarnje klase, generički tipovi i sl.	Vrlo jednostavna i čitljiva sintaksa koja se ne oslanja na vitičaste zagrade ili točke-zareze za označavanje blokova koda ili naredbi. Također podržava više paradigmi programiranja i ima dinamičko tipiziranje podataka.
Paradigma	Podržava imperativnu i proceduralnu paradigmu programiranja, koja se temelji na naredbama koje mijenjaju stanje programa. Također podržava niskorazinsko programiranje koje omogućuje direktan pristup memoriji i hardveru.	Podržava više paradigmi programiranja, kao što su imperativna, proceduralna, objektno orijentirana, generička, funkcionalna i metaprogramiranje. Također podržava niskorazinsko programiranje koje omogućuje direktan pristup memoriji i hardveru.	Podržava objektno orijentiranu paradigmu programiranja, koja se temelji na objektima koji imaju svoje podatke i ponašanje. Također podržava generičku, funkcionalnu, deklarativnu i asinkronu paradigmu programiranja. Ne podržava niskorazinsko programiranje koje omogućuje direktan pristup memoriji i hardveru.	Podržava objektno orijentiranu paradigmu programiranja, koja se temelji na objektima koji imaju svoje podatke i ponašanje. Također podržava generičku i funkcionalnu paradigmu programiranja. Ne podržava niskorazinsko programiranje koje omogućuje direktan pristup memoriji i hardveru.	Podržava više paradigmi programiranja, poput imperativne, proceduralne, objektno orijentirane, generičke, funkcionalne i deklarativne. Podržava dinamičko tipiziranje podataka i refleksiju. Ne podržava niskorazinsko programiranje koje omogućuje direktan pristup memoriji i hardveru.
Namjena	Koristi se za razvoj operacijskih sustava, ugradbenih sustava, kompajlera, jezgrenih biblioteka i drugih aplikacija koje zahtijevaju visoke performanse i kontrolu nad resursima.	Koristi se za razvoj operacijskih sustava, ugradbenih sustava, kompajlera, jezgrenih biblioteka, grafičkih sučelja, igara i drugih aplikacija koje zahtijevaju visoke performanse i kontrolu nad resursima.	Koristi se za razvoj web aplikacija, stolnih aplikacija, mobilnih aplikacija, igara i drugih aplikacija koje rade u .NET Framework-u.	Koristi se za razvoj web aplikacija, stolnih aplikacija, mobilnih aplikacija, igara i drugih aplikacija koje rade na više platformi.	Koristi se za razvoj web aplikacija, stolnih aplikacija, mobilnih aplikacija, igara, znanstvenih i matematičkih aplikacija i drugih aplikacija koje zahtijevaju jednostavnost i fleksibilnost.
Brzina	Ima vrlo visoku brzinu izvođenja jer se prevodi u strojni kôd koji se izravno izvodi na procesoru. Međutim, brzina izvođenja ovisi o kvaliteti koda i kompajlera.	Ima vrlo visoku brzinu izvođenja jer se prevodi u strojni kôd koji se izravno izvodi na procesoru. Međutim, brzina izvođenja ovisi o kvaliteti koda i kompajlera.	Visoka brzina izvođenja jer se prevodi u međujezik koji se izvodi u virtualnom stroju koji optimizira kôd. Međutim, brzina izvođenja ovisi o kvaliteti virtualnog stroja i operacijskog sustava.	Visoka brzina izvođenja jer se prevodi u međujezik koji se izvodi u virtualnom stroju koji optimizira kôd. Međutim, brzina izvođenja ovisi o kvaliteti virtualnog stroja i operacijskog sustava.	Ima nisku brzinu izvođenja jer se interpretira u virtualnom stroju koji ne optimizira kôd. Međutim, brzina izvođenja ovisi o kvaliteti interpretera i operacijskog sustava.
Popularnost	Ima vrlo visoku brzinu izvođenja jer se prevodi u strojni kôd koji se izravno izvodi na procesoru. Međutim, brzina izvođenja ovisi o kvaliteti koda i kompajlera.	Ima vrlo visoku brzinu izvođenja jer se prevodi u strojni kôd koji se izravno izvodi na procesoru. Međutim, brzina izvođenja ovisi o kvaliteti koda i kompajlera.	Ima visoku brzinu izvođenja jer se prevodi u međujezik koji se izvodi u virtualnom stroju koji optimizira kôd. Međutim, brzina izvođenja ovisi o kvaliteti virtualnog stroja i operacijskog sustava.	Ima visoku brzinu izvođenja jer se prevodi u međujezik koji se izvodi u virtualnom stroju koji optimizira kôd. Međutim, brzina izvođenja ovisi o kvaliteti virtualnog stroja i operacijskog sustava.	Python ima vrlo visoku popularnost među programerima zbog svoje jednostavnosti, fleksibilnosti i raznolikosti primjene. Python se koristi u mnogim industrijama i područjima, kao što su web razvoj, znanost o podacima, umjetna inteligencija, robotika, digitalna obrada slike i druge.

6.3. Prednosti i nedostaci korištenja različitih programskih jezika u razvoju aplikacija

Prednosti i nedostaci korištenja različitih programskih jezika u razvoju aplikacija bitna su tema koja se tiče svih koji se bave programiranjem i razvojem softvera. Programski jezici su skupovi simbola i pravila kojima se opisuje postupak računanja, odnosno algoritam ili program koji se može izvršiti na računalu (Lovreničić i Konecki, 2017), a razlikuju po svojoj sintaksi, semantici, paradigmi, namjeni i mogućnostima, te imaju različite prednosti i nedostatke za razvoj aplikacija.

Prednosti korištenja različitih programskih jezika u razvoju aplikacija su:

- veća izbirljivost i prilagodljivost te
- veća kreativnost i inovativnost.

Veća izbirljivost i prilagodljivost predstavlja mogućnost izbora onog programskog jezika koji najbolje odgovara potrebama i zahtjevima aplikacije. Primjerice, ako se želi razviti aplikacija koja radi na više platformi, može se koristiti multiplatformski programski jezik kao što je Java ili C#. Želi li se razviti aplikacija koja ima visoke performanse i kontrolu nad hardverom, može se koristiti niski programski jezik kao što je C ili C++. Ako se želi razviti aplikacija koja ima složenu logiku i umjetnu inteligenciju, može se koristiti logički ili funkcionalni programski jezik kao što je Prolog ili Haskell (Kovačević i Stojanović, 2022).

Veća kreativnost i inovativnost opisuje mogućnost iskorištavanja raznolikosti i bogatstva programskih jezika za stvaranje kreativnih i inovativnih rješenja. Na primjer, ako se želi razviti aplikacija koja ima interaktivno i atraktivno sučelje, može se koristiti grafički ili multimedijski programski jezik kao što je HTML ili Flash. Želi li se, pak, razviti aplikacija koja ima dinamičko i fleksibilno ponašanje, može se koristiti skriptni ili interpretirani programski jezik kao što je Python ili JavaScript. Za razvoj aplikacije koja ima modularnu i ponovno upotrebljivu strukturu, može se koristiti objektno orijentirani ili komponentni programski jezik kao što je C++ ili Java (Lovrenčić i sur., 2009).

Nedostaci korištenja različitih programskih jezika u razvoju aplikacija su:

- veća složenost i težina te
- veći troškovi i rizici.

Veća složenost i težina opisuje suočavanje s većom složenošću i težinom u učenju i pisanju programa. Konkretno, ako se želi koristiti više programskih jezika u istoj aplikaciji, može se stvoriti problem kompatibilnosti i komunikacije između njih. Ako se želi koristiti novi ili rijetki programski jezik, može se dogoditi da nedostaje dokumentacije ili podrške za njega. Ipak, ako se želi koristiti specifičan ili ograničen programski jezik, može se ograničiti mogućnosti i funkcionalnosti aplikacije (Lovrenčić i sur., 2009).

Korištenjem različitih programskih jezika može se povećati troškovi i rizici u razvoju i održavanju aplikacije. Na primjer, ako se želi koristiti skupi ili licencirani programski jezik, može se povećati financijski trošak za nabavu i korištenje istog. Korištenjem nestabilnoga ili nesigurnoga programskoga jezika, može se povećati tehnički rizik za pojavu grešaka ili sigurnosnih problema u aplikaciji, a korištenjem zastarjelog ili neprilagođenog programskog jezika, može se povećati operativni trošak za ažuriranje ili prilagodbu aplikacije (Lovrenčić i sur., 2009).

6.4. C# programski jezik

C# (C sharp) programski je jezik koji je razvio Microsoft i koji radi u .NET Framework-u. Taj se programski jezik koristi za razvoj web aplikacija, stolnih aplikacija, mobilnih aplikacija, igara i još mnogo toga. C# je potpuno objektno orijentiran programski jezik, što znači da svi elementi unutar njega predstavljaju objekt, a objekt predstavlja strukturu koja sadrži podatke i metode koje ih obrađuju. C# se po sintaksi oslanja na mnoge druge programske

jezike, najviše na C++ i Javu, dok je sam jezik je pažljivo dizajniran s ciljem biti što jednostavniji i moderniji te iskoristiti najbolje značajke drugih jezika i popraviti nedostatke koji su se pokazali u njima. Važan primjer su generički tipovi, čija je specifikacija u C# bolje razrađena nego u Javi, što omogućuje bolje optimizirani strojni kôd, veću sigurnost pri definiciji tipova (lakše otkrivanje nekonzistentnosti u vrijeme prevođenja) te strogo poštivanje kovarijančnosti i kontravarijančnosti.

Ovo je programski jezik koji cilja na izvršno okruženje CLI i time na okvir .NET te njegove inačice kao što je Mono. Zbog toga se programi u C# izvode u posebnom izvršnom okruženju i programski kôd u tom jeziku smatra se upravljanim kôdom. Dijelove programa napisane u C# lako je povezivati s dijelovima koji su napisani u bilo kojem drugom programskom jeziku usklađenom s CLI-jem, a s postojanjem prijenosnih implementacija CLI-ja (npr. Mono ili .NET Core) programi napisani u C# imaju i vrlo dobru prijenosivost među različitim platformama. Zbog točne specifikacije programskog jezika i međujezika CIL, program napisan u C# može se izravno prevesti i izvesti na drugim okruženjima, bez potrebe za mijenjanjem programa radi prijenosivosti. Također, C# je prvi izbor za razvijanje igara u igraćem pogonu Unity. Zbog spomenutih značajki, jasne i razrađene sintakse, velike izražajnosti (mogućnost korištenja različitih paradigmi) C# je jedan od najpopularnijih programskih jezika.

Taj programski jezik ima mnoge prednosti u odnosu na druge programske jezike, koje ga čine pogodnim za razvoj različitih vrsta aplikacija. Neke od tih prednosti su:

- visoki programski jezik,
- korištenje u .NET Framework-u,
- potpuno objektno orijentiran programski jezik,
- moderan i inovativan programski jezik te
- omogućuje brz i jednostavan razvoj igara.

C# je visoki programski jezik, što znači da je lakši za učenje i pisanje od niskih programskih jezika kao što su C ili C++, te ima jasnu i razumljivu sintaksu koja se oslanja na mnoge druge popularne programske jezike, kao što su C++, Java i Visual Basic. C#, osim toga, nudi automatsko upravljanje memorijom, što smanjuje mogućnost grešaka i curenja memorije.

Zatim, C# se koristi u .NET Framework-u, koji je skup tehnologija i alata koje omogućuju razvoj web, stolnih, mobilnih i igraćih aplikacija na Windows platformi. .NET Framework nudi bogate biblioteke osnovnih klasa, koje sadrže stotine funkcija i mogućnosti za rad s datotekama, bazama podataka, mrežama, grafikom, sigurnošću i još mnogo toga. Prednost .NET Framework-a jest ta što podržava više programskih jezika koji su usklađeni s CLI-jem, a to omogućuje lako povezivanje i komunikaciju između različitih dijelova aplikacije.

Kao što je već spomenuto ranije u tekstu, C# je potpuno objektno orijentiran programski jezik, što znači da se svi elementi unutar njega predstavljaju kao objekti. Objektno orijentirano programiranje omogućuje modularnost, ponovnu upotrebu, nasljeđivanje, polimorfizam i enkapsulaciju koda, što poboljšava čitljivost, organizaciju i održivost programa.

C# je moderan i inovativan programski jezik koji se stalno razvija i prilagođava novim trendovima i potrebama u programiranju. C# podržava više paradigmi programiranja, kao što su imperativna, deklarativna, funkcionalna, generička, dinamička i asinkrona. C# također nudi napredne značajke kao što su LINQ, lambda izrazi, anonimne metode, delegati, događaji, atributi, refleksija i još mnogo toga što ga čini vrlo popularnim te je on i prvi izbor za razvoj igara u igraćem pogonu Unity. Unity je jedan od najpopularnijih alata za izradu igara za različite platforme i uređaje.

Unity podržava C# kao glavni programski jezik za pisanje skripti koje kontroliraju ponašanje objekata u igri. Zaključno, C# omogućuje brz i jednostavan razvoj igara uz korištenje bogatih grafičkih i zvučnih efekata.

7. Dokumentiranje aplikacije

Praćenje i evidentiranje velike količine robe u trgovinama uvijek je bio zahtjevan zadatak. Trgovine obično imaju veliku količinu robe za koju je teško pratiti stanje, evidentirati izlaze i ulaze, pratiti količinu, broj prodane i nabavljene robe.

Svrha aplikacije je da trgovini koja se bavi nabavom i prodajom rabljene robe i/ili komisijskom prodajom omogući praćenje i evidentiranje sve robe koju trgovina nabavi i proda putem aplikacije sa jednostavnim korisničkim sučeljem koje će omogućiti jednostavno praćenje nabave, prodaje i dostupnost robe, vlasnika i mijenjanje podataka o istim. Cilj aplikacije je omogućiti bilježenje prodaje i nabave robe, unos robe i njegovih karakteristika u bazu podataka (naziva, opisa, cijene, dostupnosti, naplaćenosti i vlasnika robe), mijenjanje karakteristika robe, pregledavanje robe u sustavu. Također omogućiti unos vlasnika u sustav, njegovo ime, prezime i proviziju i mijenjanje istih podataka o vlasniku za vlasnike koji su već u sustavu.

7.1. Opis funkcija aplikacije

Aplikacija ima sljedeće funkcije:

1. Operacije sa vlasnicima
2. Operacije sa robom
3. Košarica, stavljanje dostupne robe u košaricu i evidentiranje prodaje
4. Pretragu vlasnika u sustavu
5. Prikaz tablice iz baze podataka
6. Operacije sa bazom podataka

Detaljan opis funkcija slijedi ispod.

7.1.1. Operacije sa vlasnicima

Aplikacija omogućava sljedeće funkcije vezane sa vlasnicima:

1. Unos novog vlasnika i podataka o vlasniku:
    a. Ime
    b. Prezime
    c. Provizija
2. Uređivanje postojećeg vlasnika i podataka o vlasniku:
    a. Ime
    b. Prezime
    c. Provizija
3. Plaćanje duga vlasniku
4. Brisanje vlasnika iz sustava

7.1.2. Operacije sa robom

Aplikacija omogućava sljedeće funkcije vezane sa robom:

1. Unos nove robe i podataka o robi:
    a. Naziv
    b. Opis
    c. Cijena
    d. Prodano (je ili nije)
    e. Plaćeno vlasniku (je ili nije)
2. Uređivanje postojeće robe i podataka o robi:
    a. Naziv
    b. Opis
    c. Cijena
    d. Prodano (je ili nije)
    e. Plaćeno vlasniku (je ili nije)
3. Brisanje robe iz sustava

7.1.3. Košarica i evidentiranje prodaje

Aplikacija ima funkcionalnost dodavanja dostupne robe u košaricu i micanje robe iz košarice, prikaz ukupnog stanja košarice i gumb za obavljanje tj. evidentiranje prodaje robe koja se nalazi u košarici.

Nakon obavljanja prodaje robe, ukupni neto profit i iznos u blagajni automatski se osvježava i prikazuje na sučelju. Roba koja je u tom trenutku prodana miče se iz popisa dostupne robe, košarica se ispražnjuje i omogućava se obavljanje nove prodaje, tj. ponovni unos robe u košaricu za neku novu transakciju.

7.1.4. Pretraga vlasnika u sustavu

Aplikacija ima funkcionalnost pretrage vlasnika u sustavu, na način da se korisniku omogućava unos imena ili prezimena vlasnika kojeg želi pronaći. Dok korisnik unosi tekst u polje za unos imena/prezimena, popis vlasnika koji odgovaraju unosu teksta automatski se osvježuje sa svakim unosom i izmjenom slova u polje za tekst, tako da popis vlasnika odgovara imenu/prezimenu koji se u trenutku pisanja nalazi u polju za unos teksta.

U trenutku kada je korisnik zadovoljan sa pretragom, kada je našao vlasnika kojeg je tražio, označavanjem tog vlasnika osvježuje se popis robe koja pripada tom vlasniku.

Korisnik također može označiti bilo koju robu koja pripada vlasniku i tu tom trenutku na sučelju prikazuju se sve informacije o robi koju je korisnik označio.

7.1.5. Prikaz tablice iz baze podataka

Aplikacija omogućava korisniku da u posebnom sučelju ima prikaz svih podataka koji se trenutno nalaze u bazi podataka, prikazujući podatke u tablici.

Postoji prikaz tablice vlasnika, koji prikazuje u obliku tablice sve vlasnike u bazi podataka. U redovima tablice se nalazi svaki vlasnik, a svaki stupac predstavlja podatak o vlasnicima. Stupci odgovaraju stupcima kako je postavljeno u bazi podataka. Postoje stupci koji predstavlja Identifikacijski broj vlasnika, ime, prezime, proviziju, dug vlasniku i puno ime vlasnika. Korisnik u ovom sučelju može pretraživati vlasnike unosom imena/prezimena u polje za unos teksta, na taj način da sa svakim unosom novog slova popis(tablica) se osvježuje i prikazuje samo vlasnike koji u svom imenu ili prezimenu sadržavaju tekst koji se nalazi u polju za unos teksta.

Također postoji i prikaz tablice robe, koji prikazuje u obliku tablice svu robu koja se nalazi u bazi podataka. U redovima tablice se nalazi svaka roba, a svaki stupac predstavlja podatke o robi. Stupci odgovaraju stupcima kako je postavljeno u bazi podataka. Postoje stupci koji predstavlja identifikacijski broj robe, naziv, opis, cijenu, prodano (je ili nije), plaćeno (je ili nije) i pripadajući identifikacijski broj vlasnika svake robe (koja je primarni ključ u tablici vlasnika). Korisnik također može pretraživati robu po nazivu robe, tako da unosom teksta u polje za unos teksta unese naziv robe koju želi pronaći. Sa svakim unosom novog slova popis(tablica) se osvježuje.

Osim standardnog pregleda podataka u bazi podataka, korisnik ima sučelje u kojemu može svojom vlastitom SQL naredbom odraditi bilo koju SQL operaciju koju unese. Može koristiti naredbe: SELECT, UPDATE, INSERT, CREATE i ostale SQL naredbe, po želji. Korisnik na ovaj način može dobiti prikaz podataka u obliku tablice točno onakav kakav želi, koji odgovara SQL naredbi koju je unio. Također može mijenjati podatke i unositi nove podatke pomoću SQL naredbi UPDATE i INSERT.

7.1.6. Operacije sa bazom podataka

Aplikacija omogućava korisniku da pritiskom na gumb napravi rezervnu kopiju baze podataka koja će sadržavati sve podatke kao i trenutna baza podataka u trenutku pritiska na gumb.

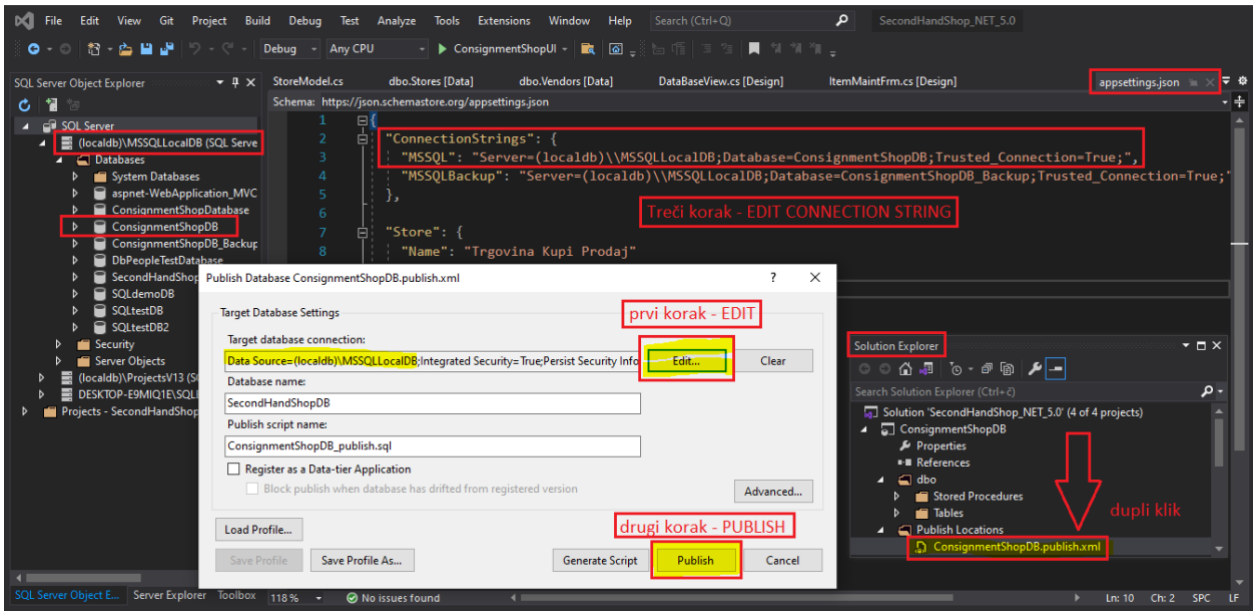
Ova funkcionalnost omogućava korisniku da na sigurnom mjestu ima rezervnu kopiju svih podataka koje su u sustavu, ukoliko dođe do bilo kakvih nepredviđenih događaja koji mogu uzrokovati oštećenje ili brisanje podataka iz trenutne baze podataka sa kojom korisnik radi.

Također omogućava korisniku da napravi kopiju svih podataka ukoliko nije siguran hoće li neka njegova radnja oštetiti ili obrisati podatke na način na koji on to nije predvidio.

7.2. Baza podataka i Modeli Podataka

Baza podataka koja se koristi za aplikaciju je MSSQL lokalna baza podataka koja se nalazi na lokalnom računalu sa koje se pokreće aplikacija. Kreacija baze podataka napravljena je u Microsoft Visual Studio programu. SQL naredbe pohranjene su u bazi podataka kao Stored Procedures. SQL naredbe nisu napisane u plain tektsu u programskom kodu što dodatno osigurava sustav od mogućnosti SQL injekcije.

Ukoliko aplikacija ne radi na drugom računalu, tj. ako se aplikacija ne želi spojiti na bazu podataka, to je zato jer na lokalnom računalu sa koje se pokreće aplikacija ne postoji baza podataka. Nova baza podataka se može kreirati putem Microsoft Visual Studio programa. Baza podataka će biti prazna, neće imati nikakve podatke u sebi ali će zadržati strukturu, relacije tj. tablice kakve su inače i biti će spremna za popunjavanje sa podacima putem aplikacije. To se može riješiti na taj način da se na vašem računalu u Visual Studiju otvori solucija sa programskim kodom i svim datotekama. Zatim je potrebno u Solution Exploreru pronaći datoteku „ConsignmentShopDB.publish.xml“. (Slika 6 Promjena Baze Podataka u Visual Studiju). Duplim klikom na tu datoteku otvara se novi prozor koji omogućuje da odaberemo SQL server (lokalni ili na mreži) i kreiramo novu bazu podataka. Kada smo odabrali server i dali ime svojoj bazi podataka stisnemo gumb „Publish“. Slijedeći korak je promjena ConnectionStringa u „appsettings.json“ datoteci.



Slika 6 Promjena Baze Podataka u Visual Studiju

ConnectionString () koji se nalazi u programskom kodu u datoteci „appsettings.json „ određuje na koju se bazu podataka aplikacija spaja. Ukoliko se lokacija baze podataka promijeni treba konfigurirati ConnectionString u datoteci „appsettings.json“. Treba promijeniti poboldano:

Server=**(localdb)\\MSSQLLocalDB**;Database=ConsignmentShopDB;Trusted_Connection=True; (umjesto boldanog treba unijeti PATH do baze podataka)

Server=(localdb)\\MSSQLLocalDB;Database=**ConsignmentShopDB**;Trusted_Connection=True; (umjesto boldanog treba unijeti ime baze podataka)

Baza podataka bi nakon ovih postupka trebala biti kreirana i spremna da je aplikacija koristi.

7.2.1. Model Roba u Bazi Podataka

SQL kôd za kreaciju tablice Robe koji je izvršen u Microsoft Visual Studio programu (Slika 7. SQL kod za kreaciju tablice Roba)

```
CREATE TABLE [dbo].[Items] (
    [Id]                 INT             IDENTITY (1, 1) NOT NULL,
    [Name]               NVARCHAR (200)  NOT NULL,
    [Description]        NVARCHAR (2000) NULL,
    [Price]              MONEY           NOT NULL,
    [Sold]               BIT             NOT NULL,
    [OwnerId]            INT             NOT NULL,
    [PaymentDistributed] BIT             NOT NULL,
    PRIMARY KEY CLUSTERED ([Id] ASC),
    CONSTRAINT [FK_Items_Vendors] FOREIGN KEY ([OwnerId])
    REFERENCES [dbo].[Vendors] ([Id])
);
```

Slika 7. SQL kod za kreaciju tablice Roba

Struktura Tablice

	Name	Data Type	Allow Nulls
🔑	Id	int	☐
	Name	nvarchar(200)	☐
	Description	nvarchar(2000)	☑
	Price	money	☐
	Sold	bit	☐
	OwnerId	int	☐
	PaymentDistributed	bit	☐

Slika 8. Struktura tablice Roba

Primjer Podataka u tablici

	Id	Name	Description	Price	Sold	Owne...	Paym...
▷	1	Harry Potter 1	a book about wizard, part 1	15.0000	True	2	False
	2	Harry Potter 2	a book about wizard, part 2	18.0000	False	1	False
	1002	Harry Potter 3	third book about a wizard, part 3	23.0000	False	2	False
	1003	Python Intro	A book about programming languag...	27.0000	False	1	False
	4002	C# for Beginners	Introduction to C# programming lan...	33.0000	False	4003	False
	4003	C# Advanced	Advanced C# programming Book	41.0000	False	4003	False
	4004	JavaScript for Beginners	Introduction to JavaScrip programmi...	18.0000	False	4002	False
	4005	JavaScript Advanced	Advanced Book about Javascript prog...	26.0000	False	4002	False
	4006	JavasScript Project Tutorial	Tutorial for making a JavaScript project	7.0000	True	4002	False

Slika 9. Primjer podataka u tablici Roba

Model Robe (kao klasa) u programskom kodu aplikacije

```
75 references
public class ItemModel
{
    5 references
    public int Id { get; set; }
    22 references
    public string Name { get; set; }
    15 references
    public string Description { get; set; }
    21 references
    public decimal Price { get; set; }
    14 references
    public bool Sold { get; set; }
    15 references
    public bool PaymentDistributed { get; set; }
    3 references
    public int OwnerId { get; set; }
    27 references
    public VendorModel Owner { get; set; }
    0 references
    public string Display
    {
        get => $"{ Name } - ${Price:f2}";
    }
}
```

Slika 10. Model Roba u programskom kodu

7.2.2. Model Vlasnici u Bazi Podataka

SQL kod za kreaciju tablice koji je izvršen u Microsoft Visual Studio programu (Slika 11. SQL kod za kreaciju tablice Vlasnici)

```
CREATE TABLE [dbo].[Vendors] (
    [Id]             INT             IDENTITY (1, 1) NOT NULL,
    [FirstName]      NVARCHAR (100)  NOT NULL,
    [LastName]       NVARCHAR (150)  NOT NULL,
    [CommissionRate] FLOAT (53)      NOT NULL,
    [PaymentDue]     MONEY           NOT NULL,
    PRIMARY KEY CLUSTERED ([Id] ASC)
);
```

Slika 11. SQL kod za kreaciju tablice Vlasnici

Struktura tablice Vlasnici

	Name	Data Type	Allow Nulls
🔑	Id	int	☐
	FirstName	nvarchar(100)	☐
	LastName	nvarchar(150)	☐
	CommissionRate	float	☐
	PaymentDue	money	☐

Slika 12. Struktura tablice Vlasnici

Primjer Podataka u tablici Vlasnici

	Id	FirstName	LastName	CommissionR...	PaymentDue
	1	Marko	Markić	0.6	0.0000
	2	Ivo	Ivić	0.7	10.5000
	4	Prazni	Praznić	0.4	0.0000
	1002	Tomislav	Jelić	0.4	0.0000
	2002	Darko	Darkić	0.65	9.7500
	2003	Pero	Perić	0.35	90.6500
	4002	Darko	Perić	0.45	3.1500
	4003	Pero	Darkić	0.5	0.0000
	6002	Tvrtko	Žarić	0.3	0.0000

Slika 13. Primjer podataka u tablici Vlasnici

Model Vlasnici (kao klasa) u programskom kodu aplikacije

```
71 references
public class VendorModel
{
    14 references
    public int Id { get; set; }
    11 references
    public string FirstName { get; set; }
    16 references
    public string LastName { get; set; }
    12 references
    public double CommissionRate { get; set; }
    13 references
    public decimal PaymentDue { get; set; }
```

Slika 14. Model Vlasnici u programskom kodu

7.2.3. Model Trgovine u Bazi Podataka

SQL kod za kreiranje tablice Trgovine koji je izvršen u Microsoft Visual Studio programu (Slika 15. SQL kod za kreiranje tablice Trgovine)

```
CREATE TABLE [dbo].[Stores] (
    [Id]          INT             IDENTITY (1, 1) NOT NULL,
    [Name]        NVARCHAR (100)  NOT NULL,
    [StoreBank]   MONEY           NOT NULL,
    [StoreProfit] MONEY           NOT NULL,
    PRIMARY KEY CLUSTERED ([Id] ASC)
);
```

Slika 15. SQL kod za kreiranje tablice Trgovine

Struktura tablice Trgovine

	Name	Data Type	Allow Nulls
🔑	Id	int	☐
	Name	nvarchar(100)	☐
	StoreBank	money	☐
	StoreProfit	money	☐

Slika 16. Struktura tablice Trgovine

Prikaz podataka u tablici Trgovine

	Id	Name	StoreBank	StoreProfit
	1	Trgovina Dubrava	0.0000	0.0000
	2	Trgovina Kupi Prodaj	142.0000	81.8500

Slika 17. Podaci u tablici Trgovina

Model Trgovine (kao klasa) u programskom kodu aplikacije

```
18 references
public class StoreModel
{
    3 references
    public int Id { get; set; }

    8 references
    public string Name { get; set; }

    10 references
    public decimal StoreBank { get; set; }

    8 references
    public decimal StoreProfit { get; set; }
}
```

Slika 18. Model Trgovina u programskom kodu

7.3. Korisnička sučelja i programski kod

Aplikacija ima sljedeća sučelja:

1. Početno sučelje
2. Sučelje za operacije sa vlasnicima
3. Sučelje za operacije sa robom
4. Sučelje sa dostupnom robom i košaricom za obavljanje prodaje robe
5. Sučelje za pretragu vlasnika u sustavu
6. Sučelje koje prikazuje sve podatke bazi podataka
7. Sučelje koje omogućava operacije sa bazom podataka

7.3.1. Početno sučelje

Početno sučelje koje se otvara i prikazuje korisniku nakon pokretanja aplikacije iz koje korisnik može pristupiti ostalim sučeljima pritiskom na određeni gumb.

7.3.1.1. Sučelje

Slika 19. Sučelje – Početno

Početno sučelje sadrži gumbove koji služe za otvaranje svih sučelja koji su na raspolaganju u aplikaciji.

Pritiskom na gumb „Prodaja“ otvara se sučelje u kojemu korisnik može zabilježiti prodaju robe na način na dodaje određenu robu u košaricu i nakon toga obavlja prodaju.

Pritiskom na gumb „Roba“ otvara se sučelje u kojemu korisnik može dodavati novu robu, brisati ili mijenjati podatke o postojećoj robi.

Pritiskom na gumb „Vlasnici“ otvara se sučelje u kojemu korisnik može dodavati nove vlasnike, brisati ili mijenjati podatke o postojećim vlasnicima.

Pritiskom na gumb „Baza Podataka“ otvara se sučelje u kojemu korisnik ima pregled nad svim podacima u tabličnom prikazu onako kako oni izgledaju u bazi podataka, može pretraživati robu ili vlasnike i također ima mogućnost izvršavanja SQL koda.

Pritiskom na gumb „Sigurnosna Kopija“ otvara se sučelje u kojemu korisnik može izraditi sigurnosnu kopiju baze podataka i razne druge operacije sa bazom podataka.

Pritiskom na gumb „Pretraga po vlasniku“ otvara se sučelje u kojemu korisnik može pretraživati robu po imenu ili prezimenu vlasnika.

Pritiskom na gumb „Pretraga po robi“ otvara se sučelje u kojemu korisnik može pretraživati direktno robu po nazivu robe

Kod koji se pokreće pritiskom na gumbe izgleda ovako:

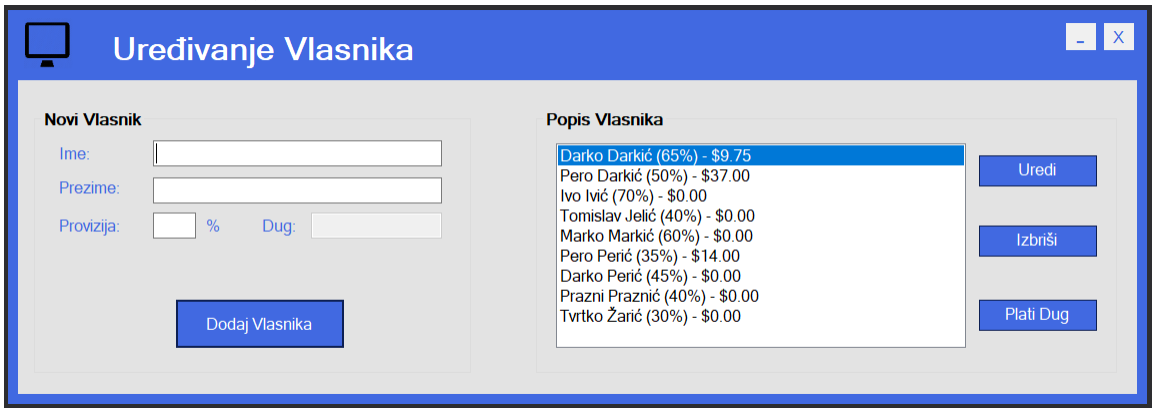
```
1 reference
private void btnShoppingCart_Click(object sender, EventArgs e)
{
    ConsignmentShop form = new ConsignmentShop();
    form.Show();
}

1 reference
private void btnItemMaintenance_Click(object sender, EventArgs e)
{
    ItemMaintFrm form = new ItemMaintFrm();
    form.Show();
}

1 reference
private void btnVendorMaintenance_Click(object sender, EventArgs e)
{
    VendorMaintFrm form = new VendorMaintFrm();
    form.Show();
}

1 reference
private void btnDbView_Click(object sender, EventArgs e)
{
    DataBaseView form = new DataBaseView();
    form.Show();
}
```

Slika 20. Kod - Gumbi na Početnoj

7.3.2. Sučelje za operacije sa vlasnicima

Sučelje u kojemu korisnik može unositi, uređivati i brisati vlasnika i podatke o vlasniku iz baze podataka.

7.3.2.1. Sučelje



Slika 21. Sučelje - Operacije sa Vlasnicima

Programski kôd koji se izvršava kod pokretanja sučelja izvršava dvije metode:

1. InitializeComponent() – metoda koja inicijalizira komponente forme
2. UpdateVendors() – metoda koja dohvaća vlasnike iz baze podataka

```
namespace ConsignmentShopUI
{
    6 references
    public partial class VendorMaintFrm : Form
    {
        private readonly BindingList<VendorModel> vendors = new BindingList<VendorModel>();
        private bool editing = false;
        private VendorModel editingVendor = null;

        private readonly IVendorData vendorData = new VendorData(dataAccess: GlobalConfig.Connection);


        2 references
        public VendorMaintFrm()
        {
            InitializeComponent();

            UpdateVendors();
        }
```

Slika 22. Kôd - Učitavanje Sučelja

Metoda UpdateVendors() dohvaća sve vlasnike i podatke o vlasnicima iz baze podataka i sprema ih u Listu Klase VendorModel koju smo nazvali allVendors (List VendorModel allVendors)

```
5 references
private async Task UpdateVendors()
{
    vendors.Clear();

    var List<VendorModel> allVendors = await vendorData.LoadAllVendors();
    allVendors = allVendors.OrderBy(keySelector: VendorModel x => x.LastName).ToList();

    foreach (var VendorModel v in allVendors)
    {
        vendors.Add(item: v);
    }

    listBoxVendors.DataSource = vendors;
    listBoxVendors.DisplayMember = "Display";
    listBoxVendors.ValueMember = "Display";

    vendors.ResetBindings();
}
```

Slika 23. Kôd - Učitavanje Vlasnika

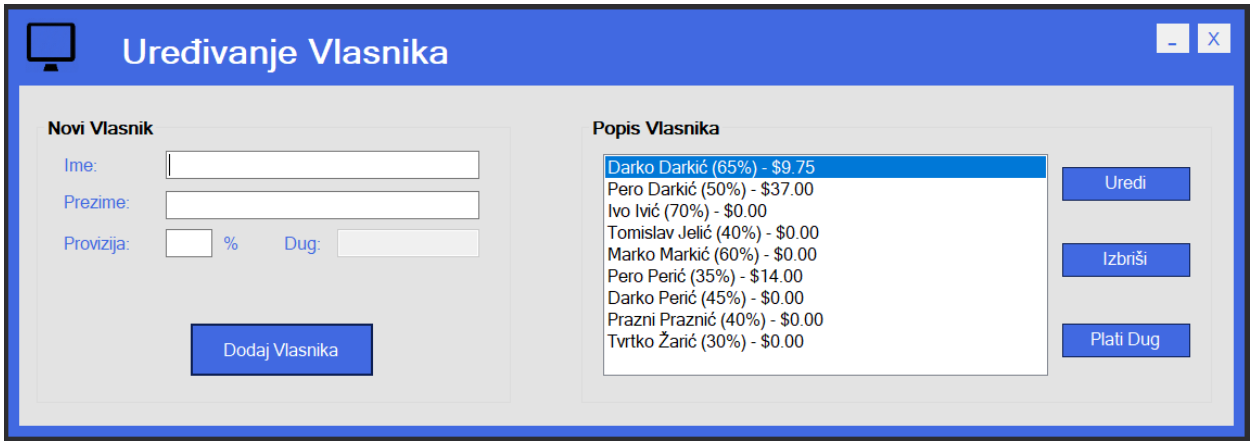
7.3.2.2. Gumb „Dodaj Vlasnika“

Kod pritiska na gumb „Dodaj Vlasnika“ u sučelju za uređivanje vlasnika pokreće se metoda btnAddVendor_Click() koja dodaje vlasnika u bazu podataka i dodjeljuje mu Ime, Prezime i Proviziju ovisno o tome što je korisnik unio u polje za unos teksta.

```
1 reference
private async void btnAddVendor_Click(object sender, System.EventArgs e)
{
    VendorModel output = null;

    if (!ValidateData())
    {
        return;
    }

    if (editing)
    {
        editingVendor.FirstName = textBoxFirstName.Text;
        editingVendor.LastName = textBoxLastName.Text;
        editingVendor.CommissionRate = double.Parse(s: textBoxCommison.Text) / 100;

        btnAddVendor.Text = "Add Vendor";
        btnEdit.Enabled = true;
        editing = false;

        output = editingVendor;

        textBoxCommison.Enabled = true;

        vendorData.UpdateVendor(vendor: output);
    }
    else
    {
        output = new VendorModel()
        {
            FirstName = textBoxFirstName.Text,
            LastName = textBoxLastName.Text,
            CommissionRate = double.Parse(s: textBoxCommison.Text) / 100
        };

        await vendorData.CreateVendor(vendor: output);
    }

    UpdateVendors();

    ClearVendorTextBoxes();
}
```

7.3.2.3. Gumb „Uredi Označenog“

Kod pritiska na gumb „Uredi označenog“ trenutni podaci o označenom vlasniku dohvaćaju se i kopiraju u polje za unos teksta i tada je korisnik u mogućnosti izmijeniti podatke o vlasniku tako da unese nove podatke u polje za unos teksta.



Slika 24. Sučelje - Uređivanje Vlasnika

```
private void btnEdit_Click(object sender, System.EventArgs e)
{
    VendorModel selectedVendor = (VendorModel)listBoxVendors.SelectedItem;
    editingVendor = selectedVendor;

    if (selectedVendor == null)
    {
        return;
    }

    if (editingVendor.PaymentDue > 0)
    {
        textBoxCommison.Enabled = false;
    }

    editing = true;

    PopulateVendorTextBoxes();

    UpdateVendors();

    btnAddVendor.Text = "Uredi Vlasnika";
    btnEdit.Enabled = false;
}
```

Slika 25. Kôd – Gumb „Uredi Označenog“

7.3.2.4. Gumb „Uredi Vlasnika“

Kada je korisnik promijenio podatke i zadovoljan je sa promjenom pritiskom na gumb „Uredi Vlasnika“ (to je isti gumb koji se prije zvao „Dodaj Vlasnika“, samo je sada promijenjen tekst gumba u „Uredi Vlasnika“) mijenja podatke o vlasniku i sprema promjene u bazi podataka. U tom slučaju se izvršava sljedeći kôd.

```
if (editing)
{
    editingVendor.FirstName = textBoxFirstName.Text;
    editingVendor.LastName = textBoxLastName.Text;
    editingVendor.CommissionRate = double.Parse(s: textBoxCommison.Text) / 100;

    btnAddVendor.Text = "Add Vendor";
    btnEdit.Enabled = true;
    editing = false;

    output = editingVendor;

    textBoxCommison.Enabled = true;

    vendorData.UpdateVendor(vendor: output);
}
```

Slika 26. Kôd - Gumb "Uredi Vlasnika"

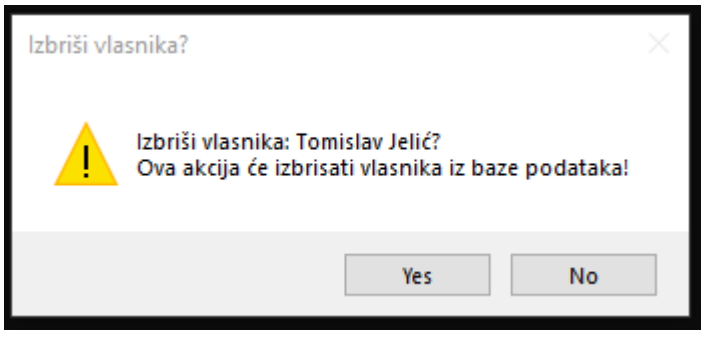
7.3.2.5. Gumb „Izbriši Označenog“

Kod pritiska na gumb „Izbriši Označenog“ briše se označeni vlasnik iz baze podataka.

```
private async void btnItemDelete_Click(object sender, System.EventArgs e)
{
    VendorModel selectedVendor = (VendorModel)listBoxVendors.SelectedItem;

    if (selectedVendor == null)
    {
        return;
    }

    var DialogResult result = MessageBox.Show(text: $"Izbriši vlasnika: {selectedVendor.FullName}?" +
                                $"\nOva akcija će izbrisati vlasnika iz baze podataka!",
                                caption: "Izbriši vlasnika?",
                                buttons: MessageBoxButtons.YesNo,
                                icon: MessageBoxIcon.Warning);

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

    try
    {
        await VendorHelper.RemoveVendor(vendor: selectedVendor);
    }
    catch (InvalidOperationException ex)
    {
        MessageBox.Show(text: ex.Message);
    }

    UpdateVendors();
}
```

Slika 27. Kôd - Gumb "Izbriši Označenog"

Prije nego li se vlasnik uistinu izbriše iz baze podataka, korisniku iskače prozor upozorenja koji zahtjeva potvrdu korisnika da uistinu želi izbrisati vlasnika iz baze podataka.



Slika 28. Skočni Prozor - Izbriši Vlasnika?

7.3.2.6. Gumb „Plati Dug Vlasniku“

Kod pritiska na gumb „Plati Dug Vlasniku“ mijenjaju se podaci u bazi podataka za označenog vlasnika i pripadajuću mu prodanu robu i također za blagajnu trgovine.

```
private async void btnPayVendor_Click(object sender, System.EventArgs e)
{
    VendorModel selectedVendor = (VendorModel)listBoxVendors.SelectedItem;

    if (selectedVendor == null)
    {
        return;
    }

    try
    {
        await VendorHelper.PayVendor(vendor: selectedVendor);
    }
    catch (InvalidOperationException)
    {
        MessageBox.Show(text: "Nemožete si priuštiti platiti vlasniku",
            caption: "Nedovoljno novaca",
            buttons: MessageBoxButtons.OK,
            icon: MessageBoxIcon.Error);
    }

    UpdateVendors();
}
```

Slika 29. Kôd - Gumb "Plati dug Vlasniku"

Metoda PayVendor() koja se izvršava za označenog vlasnika mijenja podatke u bazi podataka. Dug prema vlasniku smanjuje se na nula, PaymentDistributed za robu postaje true, a za trgovinu blagajna se smanjuje za određeni iznos.

```
public async static Task PayVendor(VendorModel vendor)
{
    if (vendor == null)
    {
        throw new ArgumentNullException(paramName: nameof(vendor), message: "Vlasnik mora biti odabran! Vendor cannot be null.");
    }

    IVendorData vendorData = new VendorData(dataAccess: GlobalConfig.Connection);
    IItemData itemData = new ItemData(dataAccess: GlobalConfig.Connection);
    IStoreData storeData = new StoreData(dataAccess: GlobalConfig.Connection);

    string storeName = GlobalConfig.Configuration.GetSection(key: "Store:Name").Value;
    StoreModel store = await storeData.LoadStore(name: storeName);

    var List<ItemModel> itemsOwnedByVendor = await itemData.LoadSoldItemsByVendor(vendor: vendor);

    foreach (ItemModel item in itemsOwnedByVendor)
    {
        if (!item.PaymentDistributed)
        {
            decimal amountOwed = (decimal)item.Owner.CommissionRate * item.Price;

            if (store.StoreBank >= amountOwed)
            {
                store.StoreBank -= amountOwed;

                vendor.PaymentDue -= amountOwed;

                item.PaymentDistributed = true;
            }
            else
            {
                throw new InvalidOperationException(message: "Blagajna trgovine nema dovoljno novaca da plati vlasniku!");
            }
        }

        await itemData.UpdateItem(item: item);
        await vendorData.UpdateVendor(vendor: vendor);
    }

    storeData.UpdateStore(store: store);
}
```

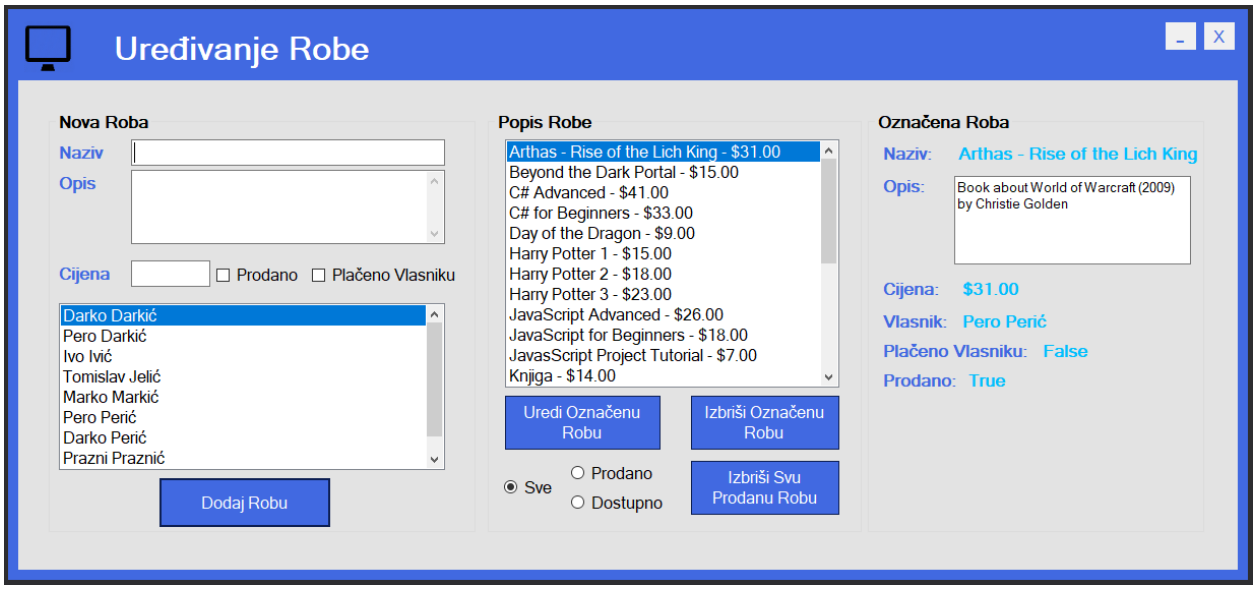
Slika 30. Kôd - Plati Vlasniku

Ukoliko u blagajni nema dovoljno novaca za platiti vlasniku korisniku iskače prozor upozorenja.

7.3.3. Sučelje za operacije sa robom

Sučelje u kojemu korisnik može unositi, uređivati i brisati robu i podatke o robi iz baze podataka.

7.3.3.1. Sučelje



Slika 31. Sučelje – Operacije sa robom

Kod učitavanja sučelja (Slika 31. Sučelje – Operacije sa robom) pozivaju se dvije metode. Metoda `UpdateItems()` i metoda `UpdateVendors()`

```
private async void ItemMaintFrm_Load(object sender, System.EventArgs e)
{
    await UpdateItems();
    await UpdateVendors();
}
```

Slika 32. Kôd - Učitavanje sučelja operacija sa robom

Metoda `UpdateItems()` dohvaća robu iz baze podataka, ovisno o tome želi li korisnik prikazati svu robu, prodanu robu ili dostupnu robu, i prikazuje ju u sučelju

```
7 references
private async Task UpdateItems()
{
    items.Clear();
    List<ItemModel> currentItems = new List<ItemModel>();

    if (radioButtonAll.Checked)
    {
        currentItems = await itemData.LoadAllItems();
        currentItems = currentItems.OrderBy(keySelector: ItemModel x => x.Name).ToList();

        foreach (var ItemModel item in currentItems)
        {
            items.Add(item: item);
        }
    }
    else if (radioButtonSold.Checked)
    {
        currentItems = await itemData.LoadSoldItems();
        currentItems = currentItems.OrderBy(keySelector: ItemModel x => x.Name).ToList();

        foreach (var ItemModel item in currentItems)
        {
            items.Add(item: item);
        }
    }
    else if (radioButtonUnsold.Checked)
    {
        currentItems = await itemData.LoadUnsoldItems();
        currentItems = currentItems.OrderBy(keySelector: ItemModel x => x.Name).ToList();

        foreach (var ItemModel item in currentItems)
        {
            items.Add(item: item);
        }
    }

    allItemsListBox.DataSource = items;
    allItemsListBox.DisplayMember = "Display";
    allItemsListBox.ValueMember = "Display";

    items.ResetBindings();
}
```

Slika 33. Kôd - Osvježavanje Robe

Ako je korisnik označio Radio Button za prikazivanje sve robe, izvršava se metoda LoadAllItems() koja dohvaća svu robu iz baze podataka izvršenjem SQL naredbe koja je spremljena u bazi podataka kao Stored Procedure (dbo.spItems_GetAll)

```
6 references
public async Task<List<ItemModel>> LoadAllItems()
{
    var List<ItemModel> allItems = await dataAccess.LoadData<ItemModel, dynamic>
        (storedProcedure: "dbo.spItems_GetAll", parameters: new { });

    await AssignOwner(allItems: allItems);

    return allItems;
}
```

Slika 34. Kôd - Učitavanje Robe

SQL naredba za dohvaćanje sve robe iz baze podataka izgleda ovako:

```
1   CREATE PROCEDURE [dbo].[spItems_GetAll]
2   AS
3   begin
4
5       set nocount on;
6
7       select [Id], [Name], [Description], [Price], [Sold], [OwnerId], [PaymentDistributed]
8       from dbo.Items;
9
10  end
```

Slika 35. SQL - Dohvaćanje sve Robe

Ako je korisnik označio Radio Button za prikazivanje prodane robe, izvršava se metoda LoadSoldItems() koja dohvaća svu prodanu robu iz baze podataka izvršenjem SQL naredbe koja je spremljena kao Stored Procedure (dbo.spItems_Sold)

```
3 references
public async Task<List<ItemModel>> LoadSoldItems()
{
    var List<ItemModel> soldItems = await dataAccess.LoadData<ItemModel, dynamic>
        (storedProcedure: "dbo.spItems_GetSold", parameters: new { });

    await AssignOwner(allItems: soldItems);

    return soldItems;
}
```

Slika 36. Kôd - Učitavanje prodane Robe

SQL naredba koja dohvaća svu prodanu robu izgleda ovako:

```sql
CREATE PROCEDURE [dbo].[spItems_GetSold]
AS
begin

    set nocount on;

    select [Id], [Name], [Description], [Price], [Sold], [OwnerId], [PaymentDistributed]
    from Items
    where Sold = 1;

end
```

Slika 37. SQL - Dohvaćanje prodane Robe

Ako je korisnik označio Radio Button za prikazivanje dostupne robe, izvršava se metoda LoadUnsoldtems() dohvaća svu dostupnu robu iz baze podataka izvršenjem SQL naredbe koja je spremljena kao Stored Procedure (dbo.spItems_GetUnsold).

```
3 references
public async Task<List<ItemModel>> LoadUnsoldItems()
{
    var List<ItemModel> unsoldItems = await dataAccess.LoadData<ItemModel, dynamic>
        (storedProcedure: "dbo.spItems_GetUnsold", parameters: new { });

    await AssignOwner(allItems: unsoldItems);

    return unsoldItems;
}
```

Slika 38. Kôd - Učitavanje dostupne Robe

SQL naredba koja iz baze podataka dohvaća svu dostupnu robu izgleda ovako:

```sql
CREATE PROCEDURE [dbo].[spItems_GetUnsold]
AS
begin

    set nocount on;

    select [Id], [Name], [Description], [Price], [Sold], [OwnerId], [PaymentDistributed]
    from Items
    where sold = 0;

end
```

Slika 39. SQL - Dohvaćanje dostupne Robe

Druga metoda koja se izvršava učitavanjem sučelja je Metoda UpdateVendors() dohvaća vlasnike iz baze podataka i prikazuje ih u sučelju kako bi korisnik mogao dodijeliti kojem vlasniku roba pripada kod unosa nove robe.

```
private async Task UpdateVendors()
{
    vendors.Clear();

    var List<VendorModel> allVendors = await vendorData.LoadAllVendors();
    allVendors = allVendors.OrderBy(keySelector: VendorModel x => x.LastName).ToList();

    foreach (var VendorModel vendor in allVendors)
    {
        vendors.Add(item: vendor);
    }

    listBoxVendors.DataSource = vendors;
    listBoxVendors.DisplayMember = "FullName";
    listBoxVendors.ValueMember = "FullName";
}
```

Slika 40. Kôd - Osvježavanje Vlasnika

Metoda LoadAllVendors() dohvaća sve vlasnike iz baze podataka izvršenjem SQL naredbe koja je spremljena u bazi podataka kao Stored Procedure (dbo.spVendors_GetAll“)

```
9 references
public Task<List<VendorModel>> LoadAllVendors()
{
    return dataAccess.LoadData<VendorModel, dynamic>
        (storedProcedure: "dbo.spVendors_GetAll", parameters: new { });
}
```

Slika 41. Kôd - Učitavanje svih Vlasnika

SQL naredba koja dohvaća sve vlasnike iz baze podataka izgleda ovako:

```sql
CREATE PROCEDURE [dbo].[spVendors_GetAll]
AS
begin

    set nocount on;

    select [Id], [FirstName], [LastName], [CommissionRate], [PaymentDue]
    from Vendors;

end

```

Slika 42. SQL - Dohvaćanje svih Vlasnika

7.3.3.2. Gumb „Dodaj Robu“

Nakon što korisnik ispuni podatke o robi koju želi dodati (naziv, cijena i opis robe, je li roba prodana i je li plaćeno vlasniku i odabir vlasnika robe), roba se dodaje u bazu podataka pritiskom na gumb „Dodaj Robu“. U tom slučaju nova roba se dodaje u bazu podataka.

```
private async void btmAddItem_Click(object sender, System.EventArgs e)
{
    ItemModel output = null;

    if (!validateData())
    {
        return;
    }

    if(editing)
    {
        editingItem.Name = textBoxName.Text;
        editingItem.Price = decimal.Parse(s: textBoxPrice.Text);
        editingItem.Description = textBoxDesc.Text;
        editingItem.Owner = (VendorModel)listBoxVendors.SelectedItem;
        editingItem.OwnerId = editingItem.Owner.Id;
        editingItem.Sold = checkBoxSold.Checked;
        editingItem.PaymentDistributed = checkBoxVendorPaid.Checked;

        itemData.UpdateItem(item: editingItem);

        btnAddItem.Text = "Dodaj Robu";
        btnEdit.Enabled = true;
        editing = false;

        output = editingItem;
    }
    else
    {
        output = new ItemModel()
        {
            Name = textBoxName.Text,
            Price = decimal.Parse(s: textBoxPrice.Text),
            Description = textBoxDesc.Text,
            Owner = (VendorModel)listBoxVendors.SelectedItem,
            Sold = checkBoxSold.Checked
        };

        await itemData.CreateItem(item: output);
    }

    UpdateItems();

    ClearItemInput();
}
```

Slika 43. Kôd - Gumb "Dodaj Robu"

Izvršava se metoda CreateItem(output) koja sprema robu u bazu podataka izvršenjem SQL naredbe koja je spremljena u bazi podataka kao Stored Procedure („dbo.spItems_Insert)

```
2 references
public async Task<int> CreateItem(ItemModel item)
{
    DynamicParameters p = new DynamicParameters();

    p.Add(name: "Name", value: item.Name);
    p.Add(name: "Description", value: item.Description);
    p.Add(name: "Price", value: item.Price);
    p.Add(name: "Sold", value: item.Sold);
    p.Add(name: "OwnerId", value: item.Owner.Id);
    p.Add(name: "PaymentDistributed", value: item.PaymentDistributed);
    p.Add(name: "Id", value: 0, dbType: DbType.Int32, direction: ParameterDirection.Output);

    await dataAccess.SaveData(storedProcedure: "dbo.spItems_Insert", parameters: p);

    return p.Get<int>(name: "Id");
}
```

Slika 44. Kôd - Kreiraj novu Robu

SQL naredba za umetanje robe u bazu podataka izgleda ovako:

```
1  CREATE PROCEDURE [dbo].[spItems_Insert]
2      @Name nvarchar(200),
3      @Description nvarchar(2000),
4      @Price money,
5      @Sold bit,
6      @OwnerId int,
7      @PaymentDistributed bit,
8      @Id int = 0 output
9  AS
10 begin
11
12     set nocount on;
13
14     insert into dbo.Items (Name, Description, Price, Sold, OwnerId, PaymentDistributed)
15     values (@Name, @Description, @Price, @Sold, @OwnerId, @PaymentDistributed);
16
17     select @Id = SCOPE_IDENTITY();
18
19 end
```

Slika 45. SQL - Unesi novu Robu

7.3.3.3. Gumb „Uredi Označenu“

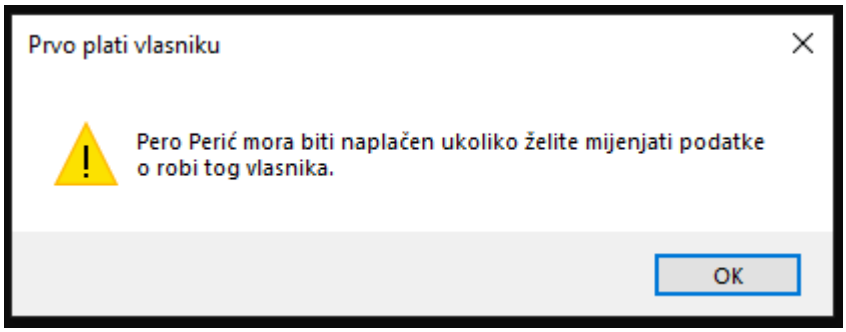
Pritiskom na gumb „Uredi Označenu“ izvršava se metoda btnEdit_Click() pa se omogućava korisniku da uredi podatke o robi prije nego li potvrdi promjene.

U tom slučaju polja za unos teksta se automatski popunjavaju sa podacima o označenoj robi koju korisnik želi promijeniti i omogućava se korisniku da unosom teksta u odgovarajuća polja promijeni podatke o robi.

```
private void btnEdit_Click(object sender, System.EventArgs e)
{
    ItemModel selectedItem = (ItemModel)allItemsListBox.SelectedItem;
    editingItem = selectedItem;

    if(selectedItem == null)
    {
        return;
    }

    if (selectedItem.Sold && !selectedItem.PaymentDistributed)
    {
        MessageBox.Show(text: $"{selectedItem.Owner.FullName} mora biti naplačen " +
            $"ukoliko želite mijenjati podatke o robi tog vlasnika.",
            caption: "Prvo plati vlasniku",
            buttons: MessageBoxButtons.OK,
            icon: MessageBoxIcon.Warning);
        return;
    }

    editing = true;

    PopulateItemTextBoxes();

    UpdateItems();

    btnAddItem.Text = "Uredi Robu";
    btnEdit.Enabled = false;
}
```

Slika 46. Kôd - Gumb "Uredi Označenu" Robu

Ukoliko korisnik odabere robu koja pripada vlasniku kome nije plaćen iznos provizije, iskače prozor upozorenja koji kaže da nije moguće mijenjati podatke o robi prije nego li je vlasnik naplaćen.



Slika 47. Skočni Prozor - Prvo plati Vlasniku

Ako je za korisnik odabrao robu koja pripada vlasniku prema kojemu nema nikakvih obveza, nakon što je zadovoljan sa promjenom podataka o robi pritiskom na gumb „Uredi Robu“ mijenja podatke o robi u bazi podataka.

U tom slučaju izvršava se kod sljedeći kod.

```
if(editing)
{
    editingItem.Name = textBoxName.Text;
    editingItem.Price = decimal.Parse(s: textBoxPrice.Text);
    editingItem.Description = textBoxDesc.Text;
    editingItem.Owner = (VendorModel)listBoxVendors.SelectedItem;
    editingItem.OwnerId = editingItem.Owner.Id;
    editingItem.Sold = checkBoxSold.Checked;
    editingItem.PaymentDistributed = checkBoxVendorPaid.Checked;

    itemData.UpdateItem(item: editingItem);

    btnAddItem.Text = "Dodaj Robu";
    btnEdit.Enabled = true;
    editing = false;

    output = editingItem;
}
```

Slika 48. Kôd - Uredi Označenu Robu

7.3.3.4. Gumb „Izbriši Označenu“

Pritiskom na gumb „Izbriši Označenu“, označena roba briše se iz baze podataka.

```
private void btnItemDelete_Click(object sender, System.EventArgs e)
{
    ItemModel selectedItem = (ItemModel)allItemsListBox.SelectedItem;

    if(selectedItem == null)
    {
        return;
    }

    if(selectedItem.Sold && !selectedItem.PaymentDistributed)
    {
        MessageBox.Show(text: $"{selectedItem.Owner.FullName} mora biti naplačen " +
            $"ukoliko želite mijenjati podatke o robi tog vlasnika.",
            caption: "Prvo plati vlasniku",
            buttons: MessageBoxButtons.OK,
            icon: MessageBoxIcon.Warning);

        UpdateItems();

        return;
    }

    itemData.RemoveItem(item: selectedItem);

    UpdateItems();
}
```

Slika 49. Kôd - Gumb "Izbriši Robu"

Izvršava se metoda RemoveItem(selectedItem) koja izvršava SQL naredbu spremljenu kao Stored Procedure ("dbo.spItems_Delete") u bazi podataka.

```
public Task RemoveItem(ItemModel item)
{
    return dataAccess.SaveData(storedProcedure: "dbo.spItems_Delete", parameters: new { Id = item.Id });
}
```

Slika 50. Kôd - Brisanje Robe

SQL naredba koja briše robu iz baze podataka izgleda ovako:

```
1   CREATE PROCEDURE [dbo].[spItems_Delete]
2       @Id int
3   AS
4   begin
5
6   set nocount on;
7
8   delete from Items where Id = @Id;
9
10  end
11
```

Slika 51. SQL - Brisanje Robe

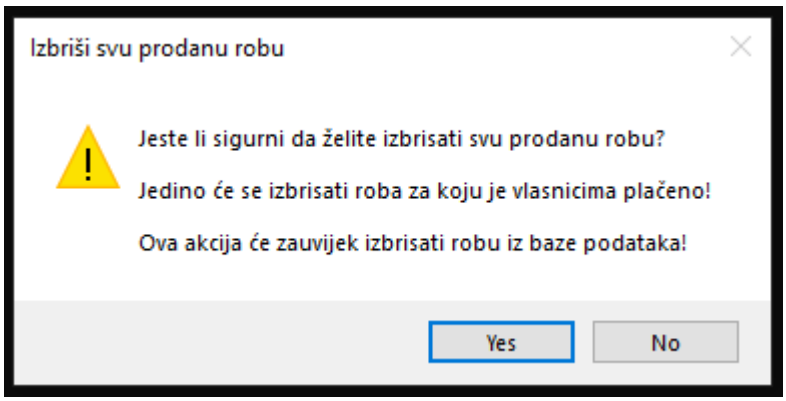
7.3.3.5. Gumb „Izbriši Prodanu Robu“

Pritiskom na gumb „Izbriši Prodanu Robu“ korisnik briše svu prodanu robu iz baze podataka.

```
private async void btnDeleteSold_Click(object sender, System.EventArgs e)
{
    var DialogResult result = MessageBox.Show(text: "Jeste li sigurni da želite izbrisati svu prodanu robu?" +
        "\n\nJedino će se izbrisati roba za koju je vlasnicima plačeno!" +
        "\n\nOva akcija će zauvijek izbrisati robu iz baze podataka!",
        caption: "Izbriši svu prodanu robu",
        buttons: MessageBoxButtons.YesNo,
        icon: MessageBoxIcon.Warning);

    if (result == DialogResult.No)
    {
        return;
    }
    else
    {
        var List<ItemModel> soldItems = await itemData.LoadSoldItems();

        foreach(var ItemModel item in soldItems)
        {
            if(item.PaymentDistributed)
            {
                await itemData.RemoveItem(item: item);
            }
            else
            {
                //MessageBox.Show($"Unable to delete {item.Name}. The vendor must be paid!");
            }
        }
        UpdateItems();
    }
}
```

Slika 52. Kôd - Gumb "Izbriši Prodanu Robu"

Prije nego li se roba uistinu izbriše iz baze podataka korisniku iskače prozor upozorenja.

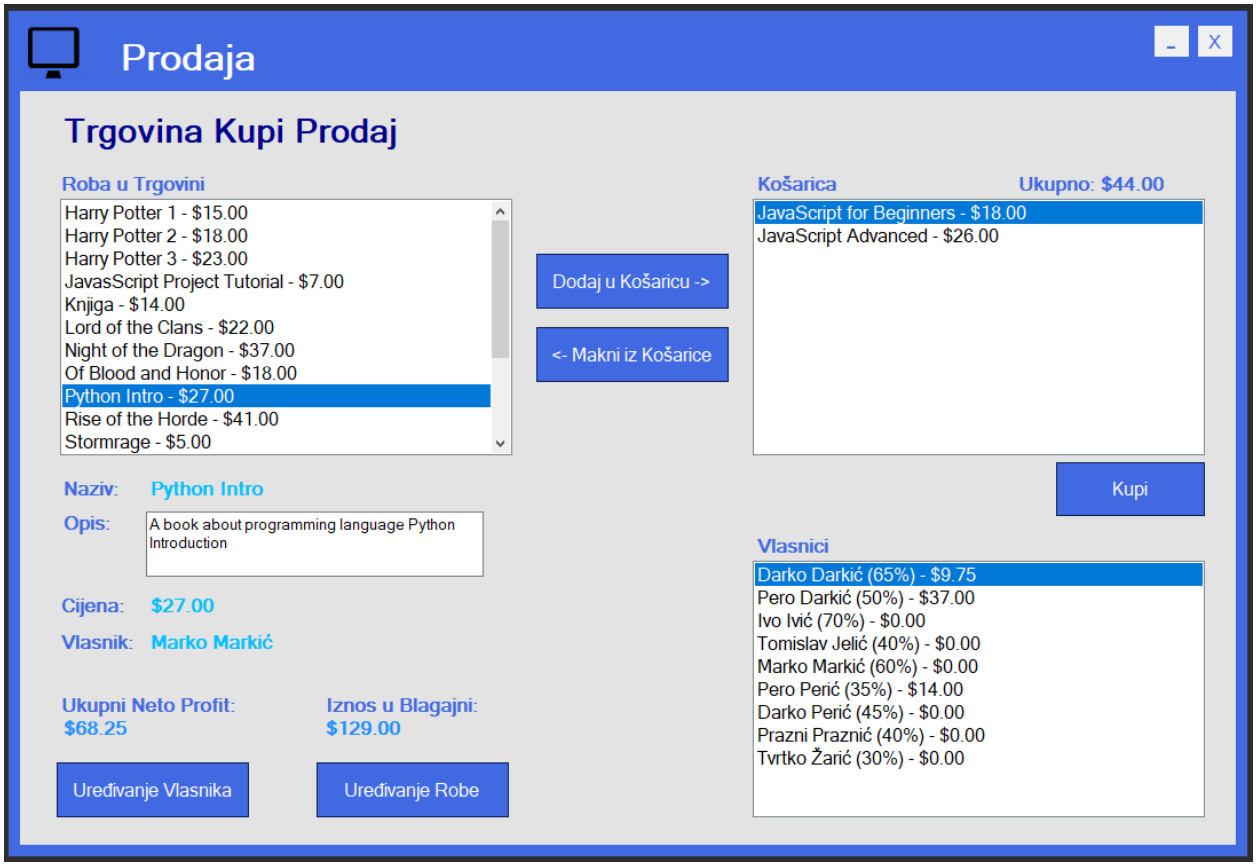


Slika 53. Skočni Prozor - Izbriši svu prodanu Robu?

7.3.4. Sučelje sa dostupnom robom za prodaju i košaricom za obavljanje prodaje robe

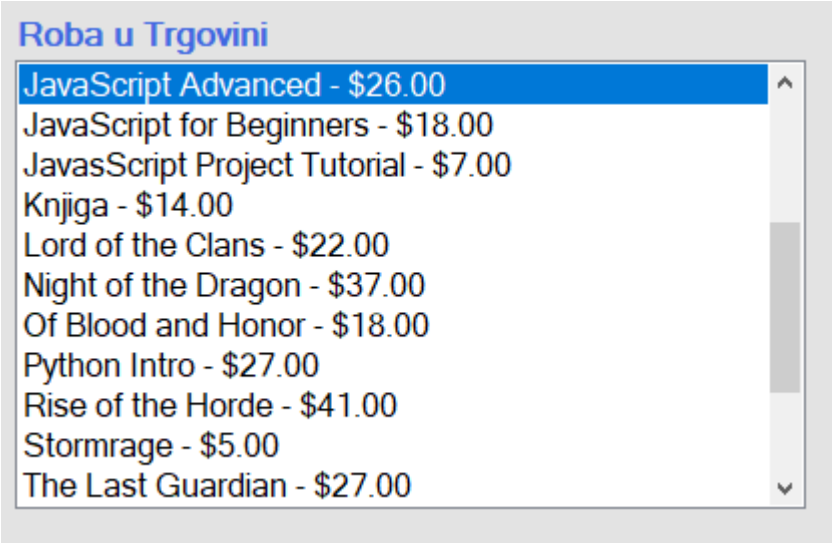
Sučelje u kojemu korisnik može dodavati dostupnu robu u košaricu i obaviti prodaju robe

7.3.4.1. Sučelje



Slika 54. Sučelje - Košarica i Prodaja Robe

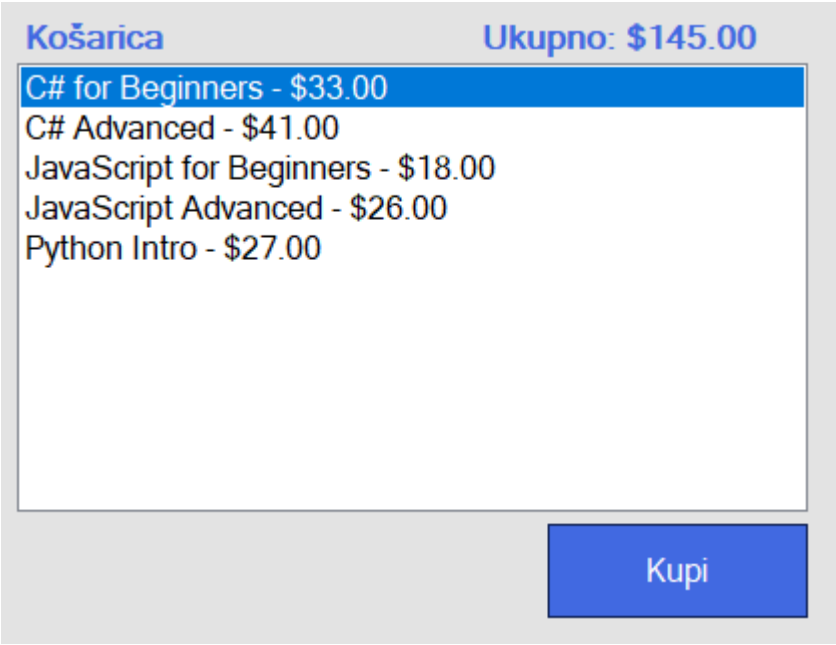
U gornjem lijevom kutu u sučelju nalazi se popis sve dostupne robe u trgovini koja je za prodaju.



Slika 55. Sučelje - Popis dostupne Robe za prodaju

U gornjem desnom kutu u sučelju nalazi se popis robe koja je trenutno u košarici. Iznad košarice je trenutni ukupni iznos koji odgovara zbroju cijena sve robe koja je trenutno u košarici.

Pritiskom na gumb „Kupi“ obavlja se prodaja robe koja je trenutno u košarici. Košarice se zatim isprazni, a sva roba koja je prethodno bila u košarici nestaje iz košarice i također više nije dostupna u popisu robe u trgovini na lijevoj strani sučelja. Također se u donjem lijevom kutu povećava ukupni neto profit (cijena robe*(1-provizija vlasnika)) i trenutno stanje iznosa u blagajni (neto profit + iznos koji još nije podmiren vlasnicima), a u donjem desnom kutu se povećava nepodmireni iznos za sve vlasnike robe koja je prodana (jer još uvijek nismo podmirili vlasniku njegov dio (proviziju). To moramo manualno obaviti u aplikaciji.

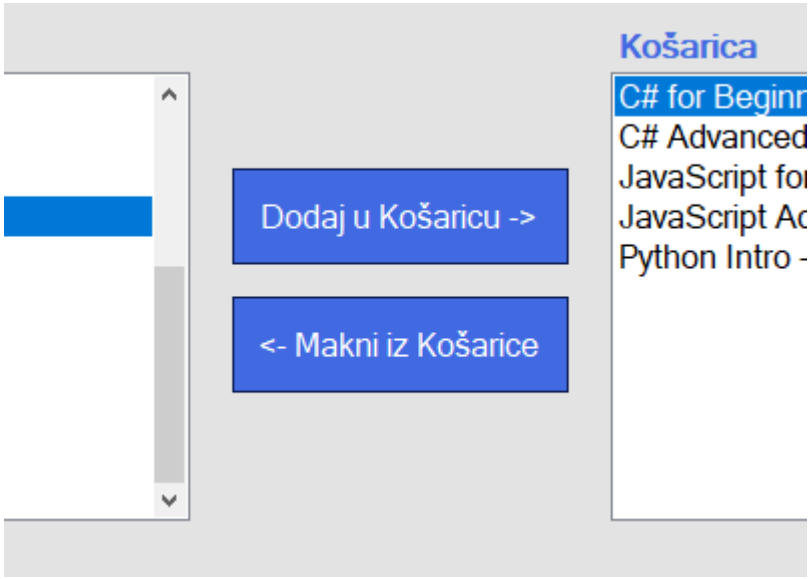


Slika 56. Sučelje - Popis Robe u Košarici

Roba se može unositi i micati iz košarice pritiskom na jedan od dva gumba („Dodaj u Košaricu“ i „Makni iz Košarice“)

Pritiskom na gumb „Dodaj u Košaricu“, označena roba sa lijeve strane prelazi u desnu stranu koja predstavlja Košaricu. Iznad popisa robe koja je u košarici piše trenutni Ukupni iznos za košaricu. Sa svakim novim dodavanjem robe u košaricu, Iznos Ukupno se povećava za iznos cijene robe.

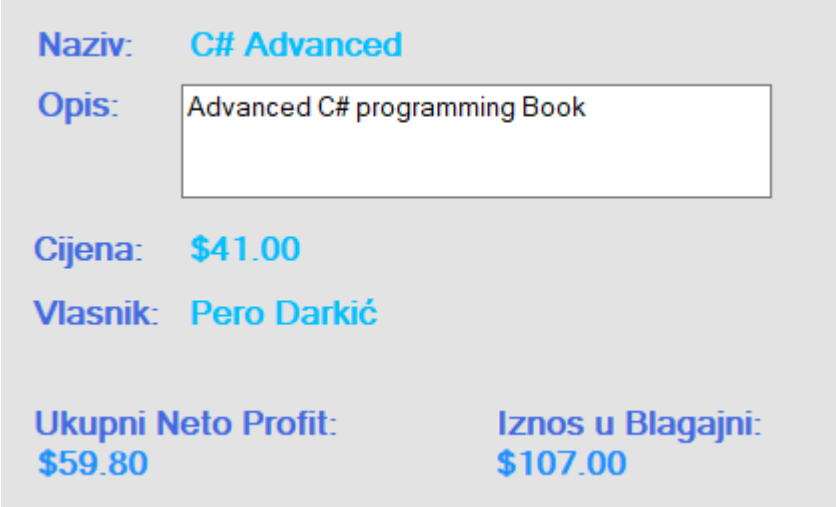
Pritiskom na gumb „Makni iz Košarice“, označena roba u košarici prelazi natrag u lijevi popis „Roba u Trgovini“ a Ukupni iznos košarice se smanjuje za cijenu robe koja je maknuta iz košarice



Slika 57. Sučelje - Gumbi za operacije sa Košaricom

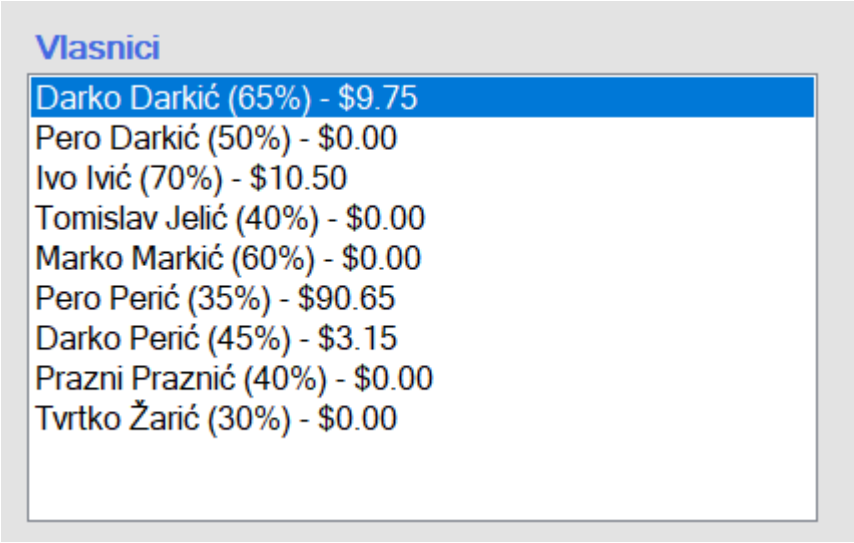
U donjem lijevom kutu u sučelju nalaze se podaci o trenutno označenoj robi.

Također ispod podataka o robi nalazi se trenutno stanje Ukupnog Neto Profita za trgovinu i trenutni Iznos u Blagajni trgovine.



Slika 58. Sučelje - Podaci o Robi, Profit i Iznos u Blagajni

U donjem desnom kutu nalazi se popis svih vlasnika, ime i prezime vlasnika, iznos provizije i iznos koji je nepodmiren prema vlasniku (iznos koji još uvijek nismo podmirili za svakom vlasnika)



Slika 59. Sučelje - Popis Vlasnika

Programski kôd koji se izvršava kod pokretanja sučelja izvršava tri metode.

1. SetupStore() – metoda koja iz baze podataka dohvaća sve podatke o trgovini
2. SetupData() – metoda koja iz baze podataka dohvaća podatke o svim vlasnicima i robi i osvježuje prikaz Neto Profita i Iznosa u Blagajni u sučelju
3. UpdateTotal() – metoda koja prikazuje Ukupni Iznos Košarice (koji je u trenutku učitavanja metode = 0)

```
private async void ConsignmentShop_Load(object sender, EventArgs e)
{
    await SetupStore();

    await SetupData();

    UpdateTotal();
}
```

Slika 60. Kôd - Učitavanje Sučelja Košarice

1. SetupStore() metoda iz baze podataka dohvaća sve podatke o trgovini

```
private async Task SetupStore()
{
    string storeName = GlobalConfig.Configuration.GetSection(key: "Store:Name").Value;
    store = await storeData.LoadStore(name: storeName);
}
```

Slika 61. Kôd - Postavljanje Trgovine

a. LoadStore() metoda izvršava SQL naredbu koja je spremljena u bazi podataka kao Stored Procedure (sbo.SpStores_Get)

```
8 references
public async Task<StoreModel> LoadStore(string name)
{
    var List<StoreModel> rows = await dataAccess.LoadData<StoreModel, dynamic>
        (storedProcedure: "dbo.SpStores_Get", parameters: new { Name = name });

    if (!rows.Any())
    {
        StoreModel store = new StoreModel { Name = name, StoreBank = 0, StoreProfit = 0 };

        await CreateStore(store: store);

        return store;
    }

    if (rows.Count > 1)
    {
        throw new Exception(message: $"Trgovina imena: {name} već postoji.");
    }

    return rows.First();
}
```

Slika 62. Kôd - Učitavanje Trgovine

SQL kod koji dohvaća podatke o trgovini iz baze podataka izgleda ovako:

```
1   CREATE PROCEDURE [dbo].[spStores_Get]
2       @Name nvarchar(100)
3   AS
4   begin
5
6       set nocount on;
7
8       select [Id], [Name], [StoreBank], [StoreProfit]
9       from Stores
10      where [Name] = @Name;
11
12  end
```

Slika 63. SQL - Dohvaćanje Trgovine

2. SetupData() – metoda iz baze podataka dohvaća podatke o svim vlasnicima i robi, prikazuje ih u sučelju i osvježuje prikaz Neto Profita i Iznosa u Blagajni u sučelju

```
private async Task SetupData()
{
    shoppingCartListBox.DataSource = shoppingCart;
    shoppingCartListBox.DisplayMember = "Display";
    shoppingCartListBox.ValueMember = "Display";

    lblStoreName.Text = store.Name;

    await UpdateVendors();
    await UpdateItems();
    UpdateBankData();
}
```

Slika 64. Kôd - Postavljanje Podataka

a. UpdateVendors() i UpdateItems() metode smo već opisali i objasnili u prethodnim formama jer se te metode više puta ponavljaju u aplikaciji
b. UpdateBankData() metoda dohvaća podatke o trgovini iz baze podataka i prikazuje na sučelju Ukupni Neto Profit i Iznos u Blagajni.

```
3 references
private async Task UpdateBankData()
{
    store = await storeData.LoadStore(name: store.Name);

    storeProfitValue.Text = $"${ store.StoreProfit:f2}";
    lblStoreBankValue.Text = $"${ store.StoreBank:f2}";
}
```

Slika 65. Kôd - Osvježavanje Iznosa u Trgovini

3. UpdateTotal() – metoda prikazuje Ukupni Iznos Košarice (koji je u trenutku učitavanja metode = 0), a sa svakim unosom robe u košaricu on se povećava za iznos cijene robe koja je dodana u košaricu.

```
4 references
private void UpdateTotal()
{
    decimal total = 0;

    foreach (var ItemModel item in shoppingCart)
    {
        total += item.Price;
    }

    lblTotal.Text = $"Ukupno: ${total:f2}";
}
```

Slika 66. Kôd - Osvježavanje Ukupnog Iznosa Košarice

7.3.4.2. Gumb „Dodaj u Košaricu“

Pritiskom na gumb „Dodaj u Košaricu“, označena roba sa lijeve strane prelazi u desnu stranu koja predstavlja Košaricu. Iznad popisa robe koja je u košarici piše trenutni Ukupni iznos za košaricu. Sa svakim novim dodavanjem robe u košaricu, Iznos Ukupno se povećava za iznos cijene robe.

```
private void addToCart_Click(object sender, EventArgs e)
{
    ItemModel selectedItem = (ItemModel)itemsListbox.SelectedItem;

    if(selectedItem == null)
    {
        return;
    }

    items.Remove(item: selectedItem); // Remove from available items
    shoppingCart.Add(item: selectedItem); // Add to shopping cart

    itemsListbox_SelectedIndexChanged(sender: this, e: EventArgs.Empty);

    UpdateTotal();
}
```

Slika 67.Kôd - Gumb "Dodaj u Košaricu"

itemsListbox_SelectedIndexChanged() metoda se u ovom slučaju izvršava jer se kod pritiska na gumb „Dodaj u Košaricu“ trenutno označena roba miče iz popisa dostupne robe pa je potrebno osvježiti prikaz podataka o automatski novo-označenoj robi.

```
2 references
private void itemsListbox_SelectedIndexChanged(object sender, EventArgs e)
{
    ItemModel selectedItem = (ItemModel)itemsListbox.SelectedItem;

    if(selectedItem == null)
    {
        ClearItemLabels();
        return;
    }

    lblNameValue.Text = $"{selectedItem.Name}";
    textBoxSelectedDesc.Text = $"{selectedItem.Description}";
    lblPriceValue.Text = $"${selectedItem.Price:F2}";
    lblVendorValue.Text = $"{selectedItem.Owner.FullName}";
}
```

Slika 68. Kôd - Event Promjena Indeksa u Popisu Robe

7.3.4.3. Gumb „Makni iz Košarice“

Pritiskom na gumb „Makni iz Košarice“, označena roba u košarici prelazi natrag u lijevi popis „Roba u Trgovini“ a Ukupni iznos košarice se smanjuje za cijenu robe koja je maknuta iz košarice

```
private void btnRemove_Click(object sender, EventArgs e)
{
    ItemModel selectedItem = (ItemModel)shoppingCartListBox.SelectedItem;

    if(selectedItem == null)
    {
        return;
    }

    items.Add(item: selectedItem);
    shoppingCart.Remove(item: selectedItem);

    UpdateTotal();
}
```

Slika 69. Kôd - Gumb "Makni iz Košarice"

7.3.4.4. Gumb „Kupi“

Pritiskom na gumb „Kupi“ obavlja se prodaja robe koja je trenutno u košarici. Košarice se zatim isprazni, a sva roba koja je prethodno bila u košarici nestaje iz košarice i također više nije dostupna u popisu robe u trgovini na lijevoj strani sučelja. Također se u donjem lijevom kutu povećava ukupni neto profit (cijena robe*(1-provizija vlasnika)) i trenutno stanje iznosa u blagajni (neto profit + iznos koji još nije podmiren vlasnicima), a u donjem desnom kutu se povećava nepodmireni iznos za sve vlasnike robe koja je prodana (jer još uvijek nismo podmirili vlasniku njegov dio (proviziju). To moramo manualno obaviti u aplikaciji.

Također se u bazi podataka osvježuju svi podaci o pripadajućim vlasnicima i robi.

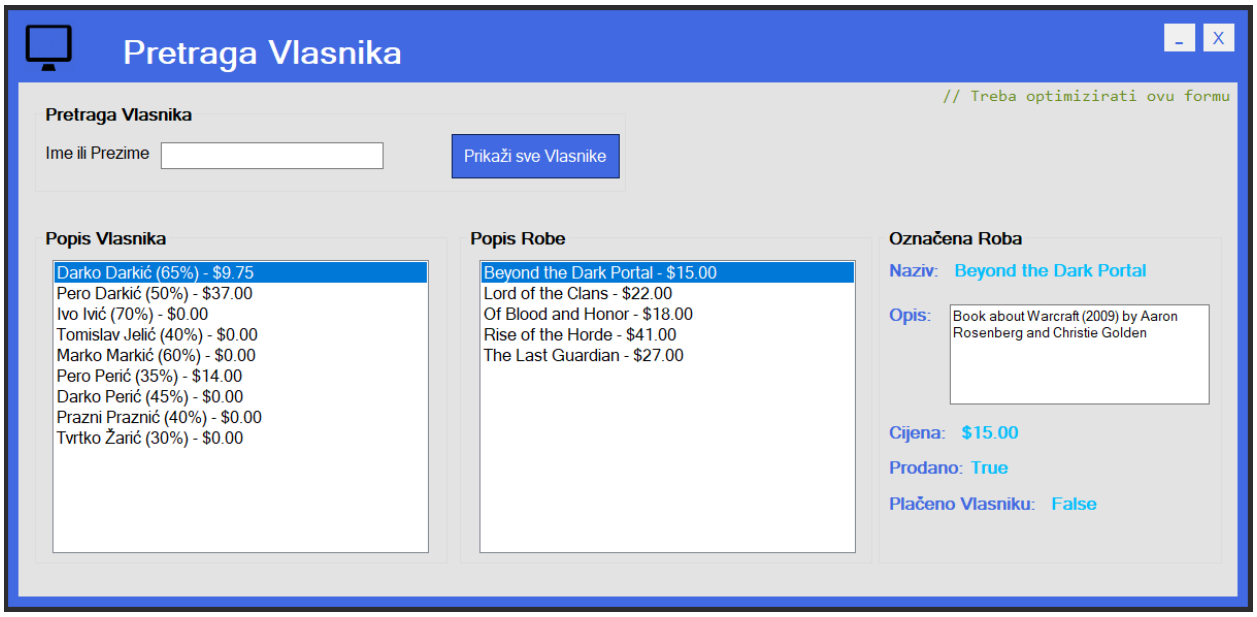
```
private async void makePurchase_Click(object sender, EventArgs e)
{
    await ItemHelper.PurchaseItems(shoppingCart: shoppingCart.ToList());

    shoppingCart.Clear();

    UpdateVendors();
    UpdateItems();
    UpdateBankData();
    UpdateTotal();

    ClearItemLabels();
}
```

Slika 70. Kôd - Gumb "Kupi"

7.3.5. Sučelje za pretragu po vlasniku

Sučelje u kojemu korisnik može pretraživati robu po imenu ili prezimenu vlasnika robe.

7.3.5.1. Sučelje



Korisnik pretraživati robu po imenu ili prezimenu vlasnika robe na način da unese tekst u polje za unos teksta. Popis vlasnika se automatski osvježava sa svakim novim unosom teksta u polje, tako da korisnik ne mora stiskati gumb kojim će potvrditi pretragu.

Nakon što kupac unese ime ili prezime vlasnika kojeg je tražio i zadovoljan je sa rezultatom, pritiskom na tog vlasnika osvježava se Popis Robe za tog vlasnika u sredini sučelja.

Korisnik nadalje može izabrati točno određenu robu u vlasništvu vlasnika kojeg je pretraživao i da desne strane sučelja će mu se prikazati informacije označenoj robi.

Korisnik također može prikazati sve vlasnike pritiskom na gumb „Prikaži sve Vlasnike“.

```
1 reference
private void txtSearchBox_TextChanged(object sender, EventArgs e)
{
    vendors.Clear();

    if (txtSearchBox.Text == string.Empty)
    {
        foreach (VendorModel vendor in allVendors)
        {
            vendors.Add(item: vendor);
        }
    }

    else
    {
        foreach (VendorModel vendor in allVendors)
        {
            if (vendor.FullName.ToUpper().Contains(value: txtSearchBox.Text.ToUpper()))
            {
                vendors.Add(item: vendor);
            }
        }
        listBoxVendors_SelectedIndexChanged(sender: this, e: EventArgs.Empty);
    }
}
```

Slika 71. Kod - Polje za unos imena/prezimena Vlasnika

Ovo je kod koji se izvršava svaki put kada se promijeni tekst u polju za unos teksta u kojemu kupac može pretraživati robu po imenu ili prezimenu vlasnika.

Sa svakim novim unosom teksta program uzima tekst koji se nalazi u polju za unos teksta i prolazi kroz svakog vlasnika koji se nalazi u sustavu. Ukoliko bilo koji vlasnik sadrži u svom imenu ili prezimenu tekst koji je korisnik unio u polje za unos teksta, program tog vlasnika prikazuje na sučelju.

Ukoliko korisnik izbriše sav tekst iz polja za unos teksta, na sučelju se prikazuju svi vlasnici. Isto to korisnik može postići pritiskom na gumb „Prikaži sve vlasnike“ u kojemu program automatski briše sav tekst iz polja za unos teksta i prikazuje sve vlasnike.

7.3.6. Sučelje prikaza baze podataka

Sučelje koje prikazuje sve podatke koji se trenutno nalaze u bazi podataka, prikazujući podatke u tablici.

7.3.6.1. Sučelje – Prikaz Vlasnika

Kod početnog otvaranja sučelja korisniku se prikazuje tablica iz baze podataka koja prikazuje sve vlasnike i informacije o vlasnicima.

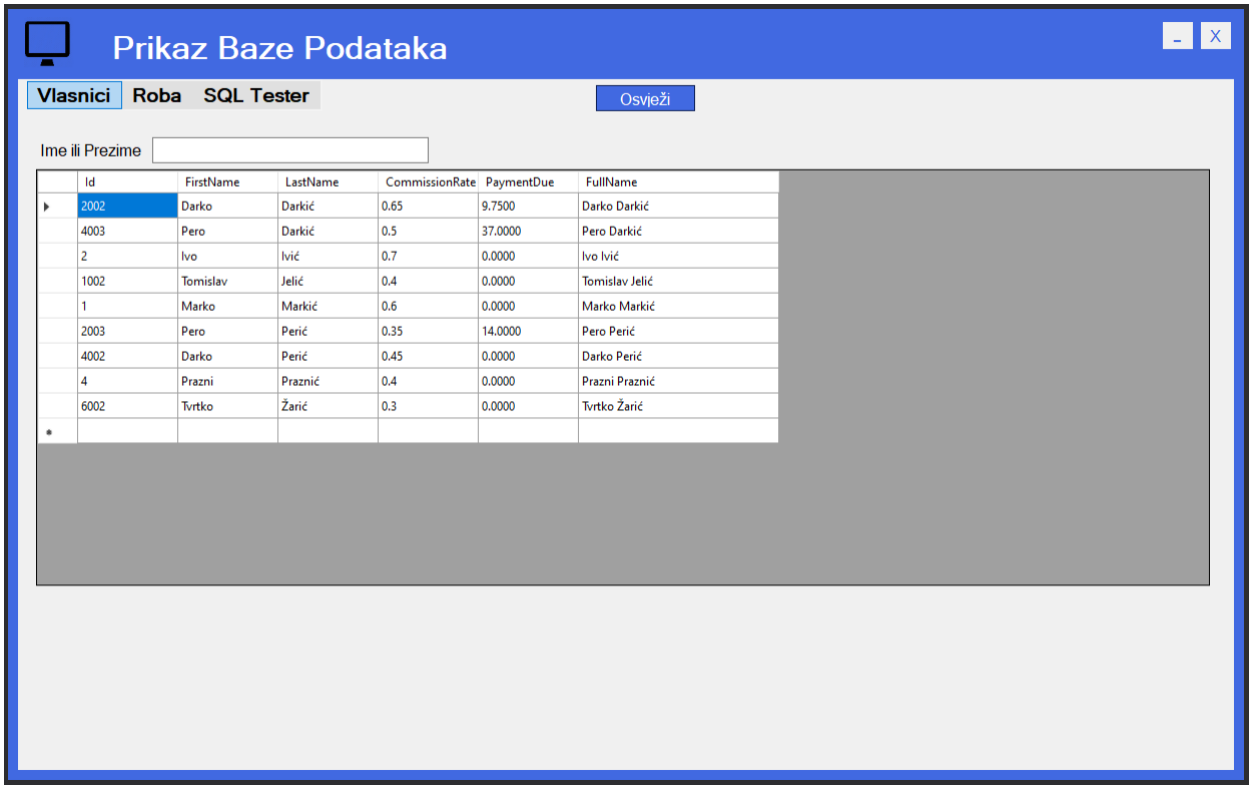
Na vrhu sučelja postoji Menu Strip koji omogućuje korisniku da izabere jedan od 3 moguća prikaza.

Korisnik može izabrati prikaz vlasnika, koji je aktiviran odmah na početku otvaranja sučelja.

Korisnik može izabrati prikaz Robe pritiskom na „Roba“ u Menu Strip-u.

Također može izabrati „SQL Tester“ koji će prikazati prozor u kojemu korisnik može unositi SQL kod koji želi izvršiti nad bazom podataka.

Ovako izgleda sučelje (Slika 72. Sučelje - Baza Podataka – Vlasnici) kod početnog pokretanja sučelja u kojemu je automatski označen prikaz vlasnika iz Menu Strip-a.



Slika 72. Sučelje - Baza Podataka – Vlasnici

U redovima tablice se nalazi svaki vlasnik, a svaki stupac predstavlja podatak o vlasnicima. Stupci odgovaraju stupcima kako je postavljeno u bazi podataka. Postoji stupac koji predstavlja Identifikacijski broj vlasnika, ime, prezime, proviziju, dug vlasniku i puno ime vlasnika. Korisnik u ovom sučelju može pretraživati vlasnike unosom imena/prezimena u polje za unos teksta, na taj način da sa svakim unosom novog slova popis(tablica) se osvježuje i prikazuje samo vlasnike koji u svom imenu ili prezimenu sadržavaju tekst koji se nalazi u polju za unos teksta.

Kod (Slika 73. Kod - Učitavanje i prikaz Vlasnika) koji omogućava prikaz vlasnika na sučelju dohvaća sve vlasnike iz baze podataka, sprema ih u listu i tu listu vlasnika prikazuje u obliku tablice na sučelju.

```
2 references
private async Task UpdateVendorsDbView()
{
    vendors.Clear();

    allVendors = await vendorData.LoadAllVendorsDbView();
    allVendors = allVendors.OrderBy(keySelector: VendorModelDbView x => x.LastName).ToList();

    foreach (var VendorModelDbView vendor in allVendors)
    {
        vendors.Add(item: vendor);
    }
    grdVlasniciTable.DataSource = null;
    grdVlasniciTable.Rows.Clear();

    grdVlasniciTable.AutoGenerateColumns = false;
    grdVlasniciTable.DataSource = vendors;
    IdVlColumn.DataPropertyName = "Id";
    FirstNameVlColumn.DataPropertyName = "FirstName";
    LastNameVlColumn.DataPropertyName = "LastName";
    CommissionRateVlColumn.DataPropertyName = "CommissionRate";
    PaymentDueVlColumn.DataPropertyName = "PaymentDue";
    FullNameVlColumn.DataPropertyName = "FullName";

    vendors.ResetBindings();
}
```

Slika 73. Kod - Učitavanje i prikaz Vlasnika

Korisnik također može pretraživati vlasnike unosom imena ili prezimena vlasnika kojeg želi naći u polje za unos teksta. (Slika 74. Kôd - Pretraga Vlasnika - Prikaz Baze Podataka)

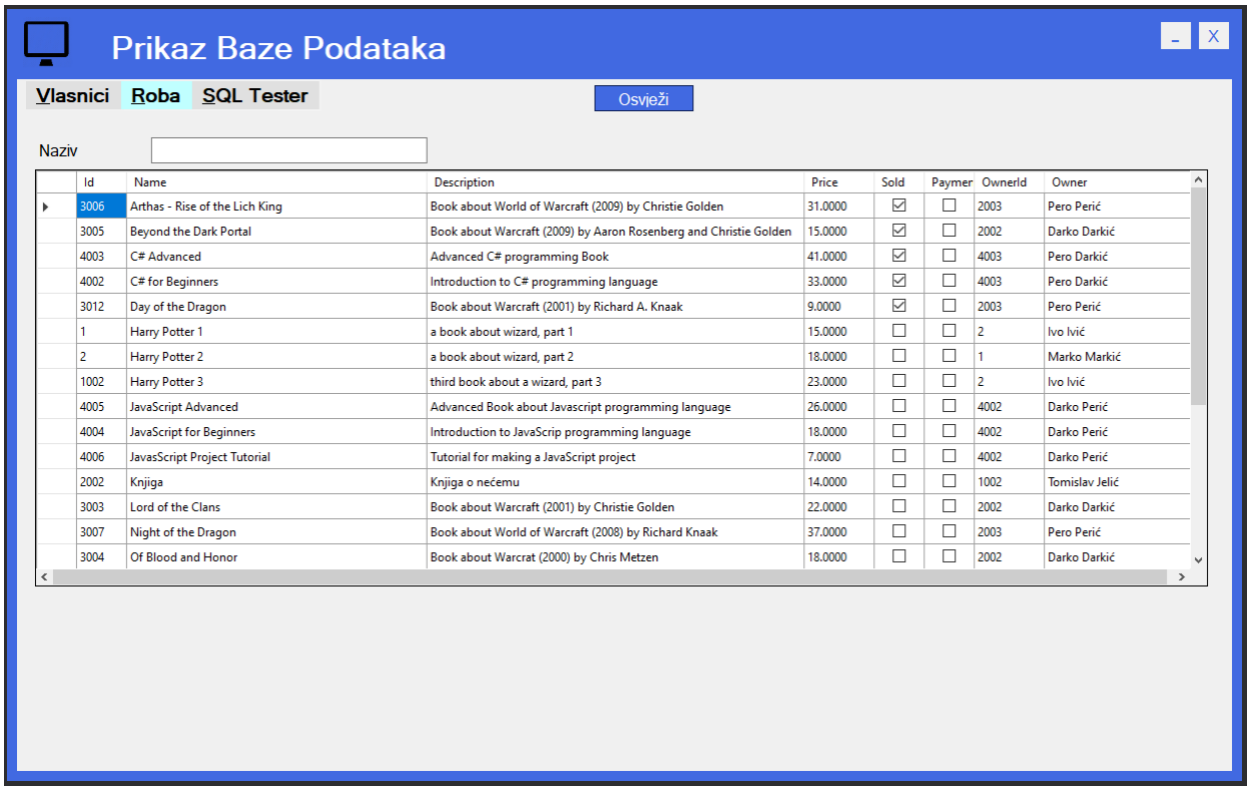
Kod koji se izvršava uspoređuje tekst koji je korisnik unio sa popisom svih vlasnika. Ukoliko bilo koji od vlasnika u svom imenu ili prezimenu sadrži tekst koji je korisnik unio, program te vlasnike prikazuje na sučelju.

```
1 reference
private void txtVlasniciSearchBox_TextChanged(object sender, EventArgs e)
{
    vendors.Clear();
    listVendorsDbSearched.Clear();

    foreach (var VendorModelDbView vendor in allVendors)
    {
        if (vendor.FullName.ToUpper().Contains(value: txtVlasniciSearchBox.Text.ToUpper()))
        {
            listVendorsDbSearched.Add(item: vendor);
        }
    }

    foreach (var VendorModelDbView vendor in listVendorsDbSearched)
    {
        vendors.Add(item: vendor);
    }
}
```

Slika 74. Kôd - Pretraga Vlasnika - Prikaz Baze Podataka

7.3.6.2. Sučelje – prikaz Robe

Ukoliko je korisnik na Menu Strip-u odabrao prikaz robe pritiskom na „Roba“ sučelje (Slika 75. Sučelje - Baza Podataka – Roba) mijenja tablicu koju prikazuje korisniku pa mu u ovom slučaju tablicu koja sadrži svu robu koja se nalazi u bazi podataka i informacije o robi. U redovima tablice se nalazi svaka roba, a svaki stupac predstavlja podatke o robi. Stupci odgovaraju stupcima kako je postavljeno u bazi podataka. Postoji stupac koji predstavlja identifikacijski broj robe, naziv, opis, cijenu, prodano (je ili nije), plaćeno (je ili nije) i pripadajući identifikacijski broj vlasnika svake robe (koja je primarni ključ u tablici vlasnika). Korisnik također može pretraživati robu po nazivu robe, tako da unosom teksta u polje za unos teksta unese naziv robe koju želi pronaći. Sa svakim unosom novog slova popis(tablica) se osvježuje.



Id	Name	Description	Price	Sold	Paymer	OwnerId	Owner
3006	Arthas - Rise of the Lich King	Book about World of Warcraft (2009) by Christie Golden	31.0000	☑	☐	2003	Pero Perić
3005	Beyond the Dark Portal	Book about Warcraft (2009) by Aaron Rosenberg and Christie Golden	15.0000	☑	☐	2002	Darko Darkić
4003	C# Advanced	Advanced C# programming Book	41.0000	☑	☐	4003	Pero Darkić
4002	C# for Beginners	Introduction to C# programming language	33.0000	☑	☐	4003	Pero Darkić
3012	Day of the Dragon	Book about Warcraft (2001) by Richard A. Knaak	9.0000	☑	☐	2003	Pero Perić
1	Harry Potter 1	a book about wizard, part 1	15.0000	☐	☐	2	Ivo Ivić
2	Harry Potter 2	a book about wizard, part 2	18.0000	☐	☐	1	Marko Markić
1002	Harry Potter 3	third book about a wizard, part 3	23.0000	☐	☐	2	Ivo Ivić
4005	JavaScript Advanced	Advanced Book about Javascript programming language	26.0000	☐	☐	4002	Darko Perić
4004	JavaScript for Beginners	Introduction to JavaScrip programming language	18.0000	☐	☐	4002	Darko Perić
4006	JavasScript Project Tutorial	Tutorial for making a JavaScript project	7.0000	☐	☐	4002	Darko Perić
2002	Knjiga	Knjiga o nećemu	14.0000	☐	☐	1002	Tomislav Jelić
3003	Lord of the Clans	Book about Warcraft (2001) by Christie Golden	22.0000	☐	☐	2002	Darko Darkić
3007	Night of the Dragon	Book about World of Warcraft (2008) by Richard Knaak	37.0000	☐	☐	2003	Pero Perić
3004	Of Blood and Honor	Book about Warcrat (2000) by Chris Metzen	18.0000	☐	☐	2002	Darko Darkić

Slika 75. Sučelje - Baza Podataka – Roba

Kod (Slika 76. Kôd - učitavanje i prikaz Robe) koji omogućava prikaz robe na sučelju dohvaća svu robu iz baze podataka, sprema ih u listu i tu listu robe prikazuje u obliku tablice na sučelju.

```
2 references
private async Task UpdateItemsDbView()
{
    itemsDbView.Clear();
    listItemsDbView.Clear();

    currentItems = await itemData.LoadAllItems();
    currentItems = currentItems.OrderBy(keySelector: ItemModel x => x.Name).ToList();

    foreach (var ItemModel item in currentItems)
    {
        listItemsDbView.Add(item: new ItemModelDbView
        {
            Id = item.Id,
            Owner = item.Owner.FullName,
            Description = item.Description,
            Name = item.Name,
            OwnerId = item.OwnerId,
            PaymentDistributed = item.PaymentDistributed,
            Price = item.Price,
            Sold = item.Sold
        });
    }
    foreach (var ItemModelDbView itemDb in listItemsDbView)
    {
        itemsDbView.Add(item: itemDb);
    }
    grdRobaTable.DataSource = null;
    grdRobaTable.Rows.Clear();
    grdRobaTable.AutoGenerateColumns = false;
    grdRobaTable.DataSource = itemsDbView;
    columnOwner.DataPropertyName = "Id";
    columnId.DataPropertyName = "Id";
    columnName.DataPropertyName = "Name";
    columnDescription.DataPropertyName = "Description";
    columnPrice.DataPropertyName = "Price";
    columnSold.DataPropertyName = "Sold";
    columnPaymentDistributed.DataPropertyName = "PaymentDistributed";
    columnOwnerId.DataPropertyName = "OwnerId";
    columnOwner.DataPropertyName = "Owner";
    itemsDbView.ResetBindings();
}
```

Slika 76. Kôd - učitavanje i prikaz Robe

Korisnik također može pretraživati robu unosom naziva robe koju želi naći u polje za unos teksta. (Slika 77. Kôd - Pretraga Robe - Prikaz Baze Podataka)

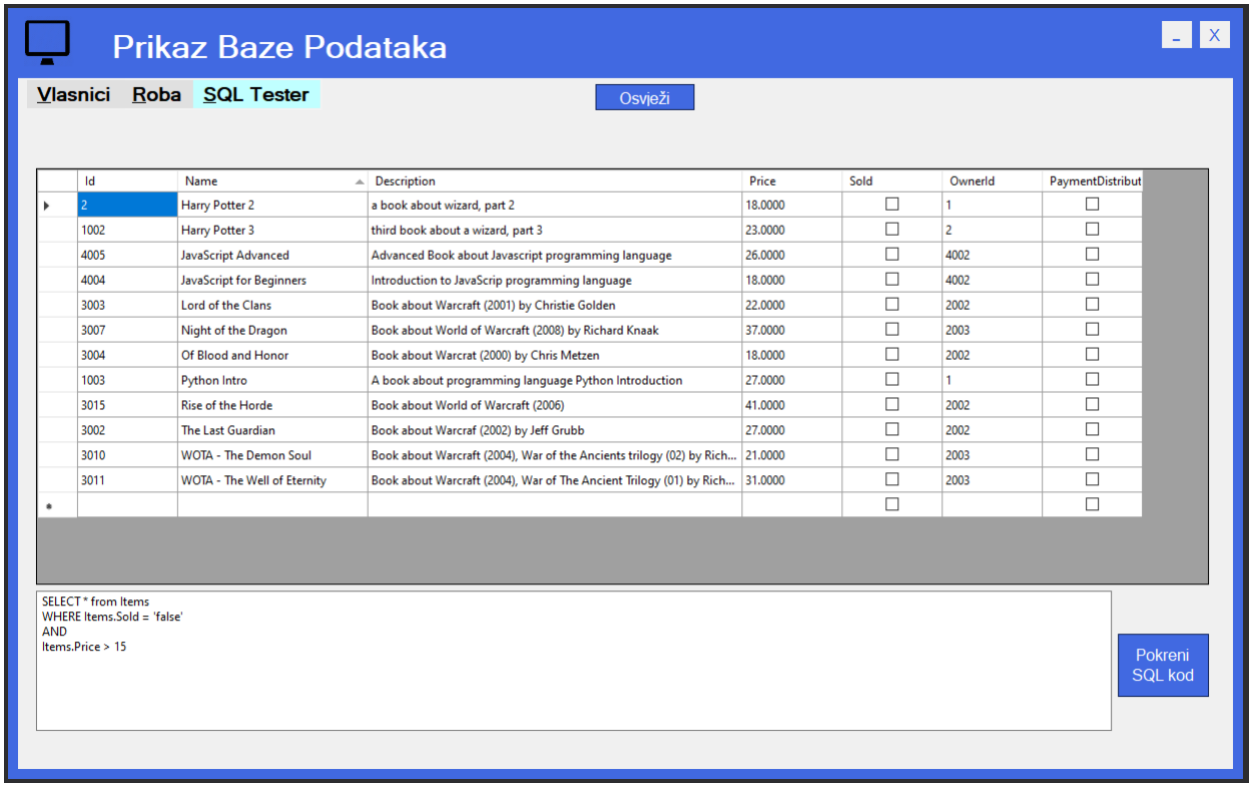
Kod koji se izvršava uspoređuje tekst koji je korisnik unio sa popisom sve robe. Ukoliko bilo roba u svom nazivu sadrži tekst koji je korisnik unio, program tu robu prikazuje na sučelju.

```
1 reference
private void txtRobaSearchBox_TextChanged(object sender, EventArgs e)
{
    itemsDbView.Clear();

    foreach (var ItemModelDbView item in listItemsDbView)
    {
        if (item.Name.ToUpper().Contains(value: txtRobaSearchBox.Text.ToUpper()))
        {
            itemsDbView.Add(item: item);
        }
    }
}
```

Slika 77. Kôd - Pretraga Robe - Prikaz Baze Podataka

7.3.6.3. Sučelje – SQL Tester

Osim standardnog pregleda podataka u bazi podataka, korisnik ima sučelje (Slika 78. Sučelje - SQL Tester) u kojemu može svojom vlastitom SQL naredbom odraditi bilo koju SQL operaciju koju unese. Može koristiti naredbe: SELECT, UPDATE, INSERT, CREATE i ostale SQL naredbe, po želji. Korisnik na ovaj način može dobiti prikaz podataka u obliku tablice točno onakav kakav želi, koji odgovara SQL naredbi koju je unio. Također može mijenjati podatke i unositi nove podatke pomoću SQL naredbi UPDATE i INSERT. Ovo sučelje će se korisniku prikazati pritiskom na „SQL Tester“ u Menu Strip-u.

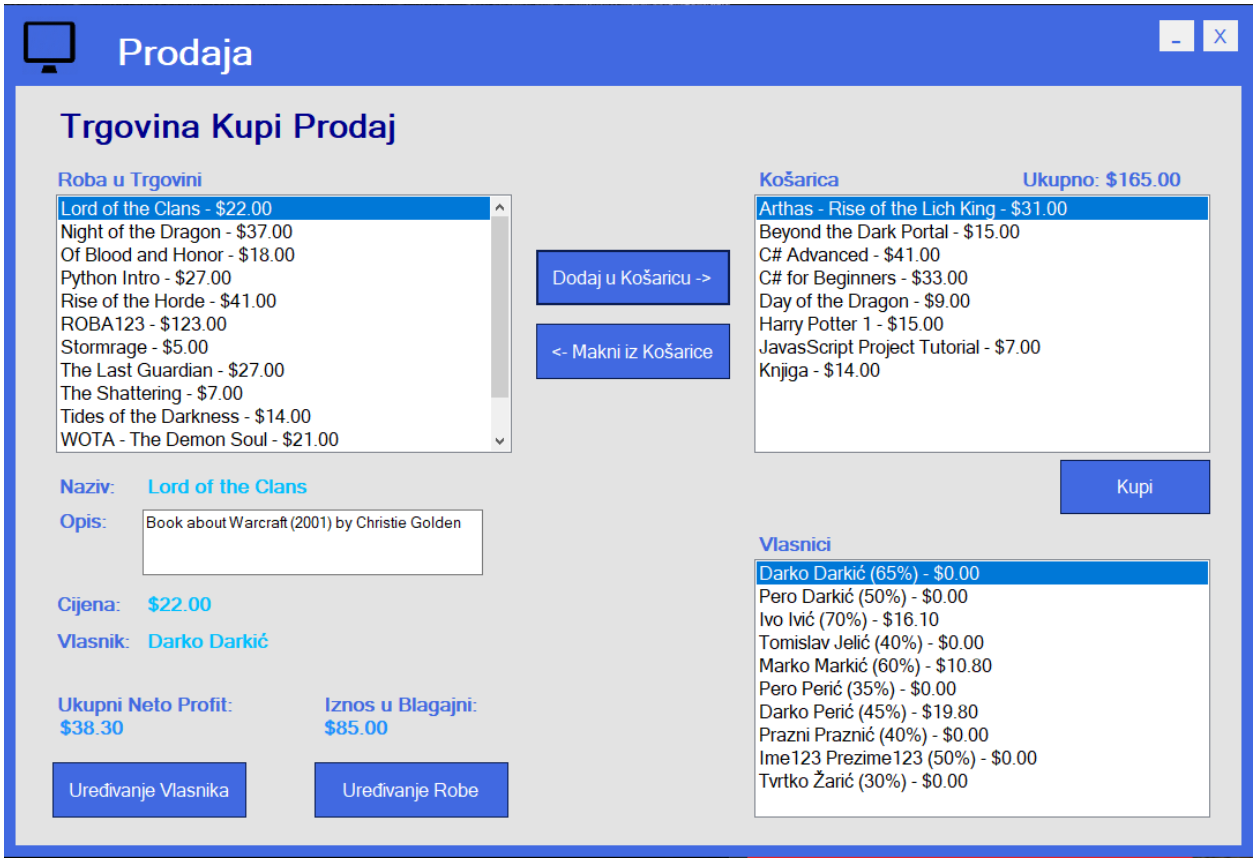


Slika 78. Sučelje - SQL Tester

7.4. Prikaz rada aplikacije

Detaljan prikaz svih funkcionalnosti sučelja u aplikaciji možemo pročitati u sekciji 4.3.(Korisnička sučelja i programski kod) pa da ne ponavljam detaljno sve funkcionalnosti, slijedi ukratko prikaz rada aplikacije sa temeljnim, glavnim funkcionalnostima.

U formi za prodaju dodajemo robu iz trgovine u košaricu i pritiskom na gumb „Kupi“ potvrđujemo kupnju robe. Ukupni neto profit, Iznos u blagajni i dug vlasnicima se povećava i prikazuje na sučelju. Košarica se zatim isprazhnjuje a prodana roba više nije dostupna za prodaju. (Slika 79. Prikaz rada aplikacije – Prodaja)



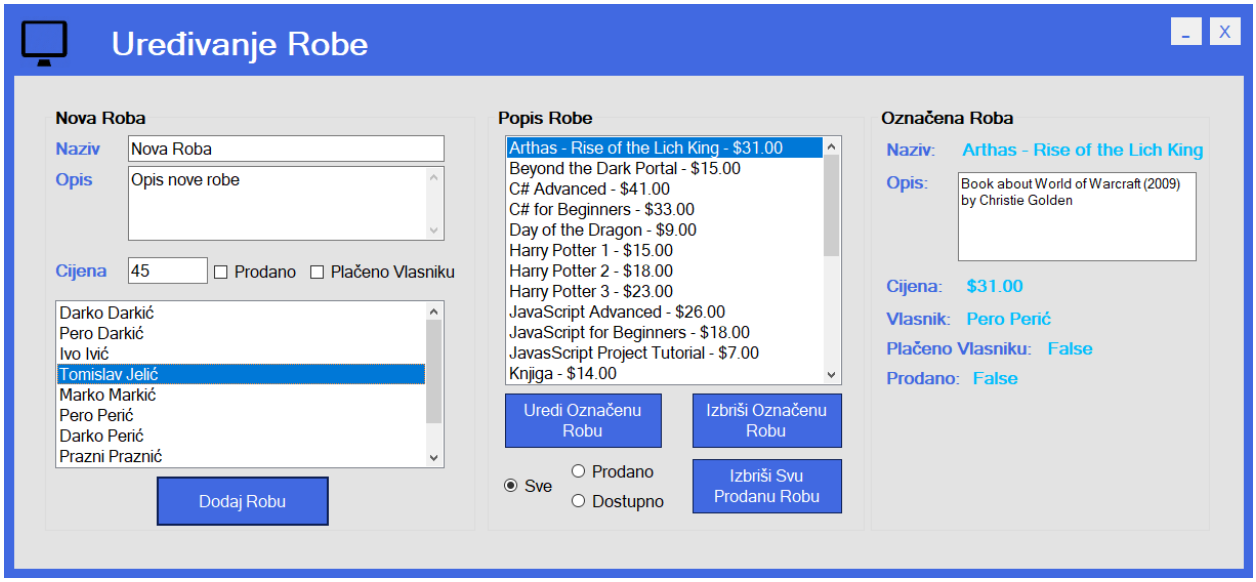
Slika 79. Prikaz rada aplikacije – Prodaja

U formi za uređivanje Robe možemo dodavati novu robu tako da robi damo naziv, opis i cijenu i pritiskom na gumb „Dodaj Robu“ roba je dodana u bazi podataka.

Robu možemo i uređivati tako da odaberemo robu iz popisa robe i kliknemo na gumb uredi. Kada smo zadovoljni za izmjenama kliknemo na gumb Uredi Robu.

Robu možemo brisati tako da je označimo i kliknemo na gumb Izbriši označenu robu. Možemo također automatski izbrisati svu prodanu robu iz baze podataka pritiskom na gumb Izbriši Svu Prodanu Robu.

Označena roba sa desne strane ima popis svih karakteristika. (Slika 80. Prikaz rada aplikacije - Uređivanje i dodavanje robe)



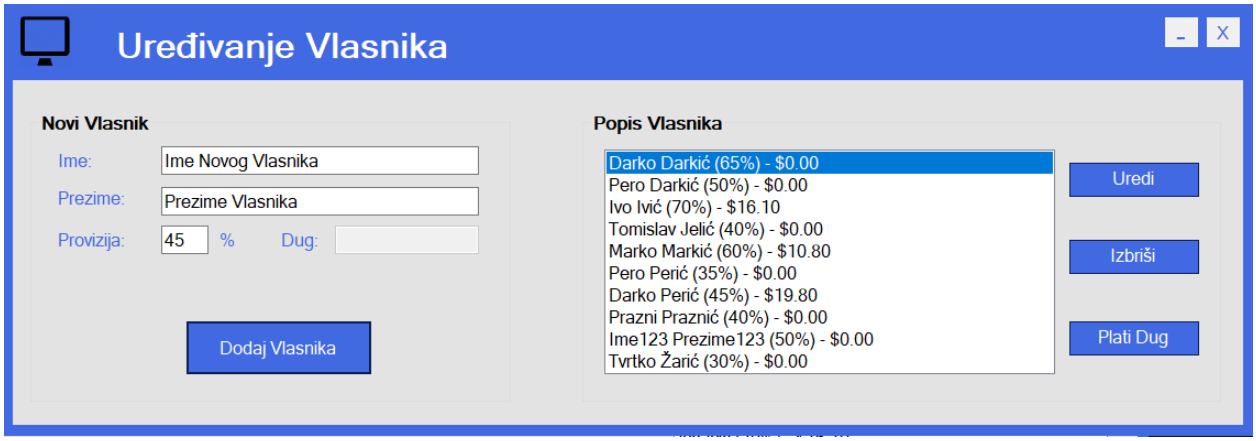
Slika 80. Prikaz rada aplikacije - Uređivanje i dodavanje robe

U formi za uređivanje Vlasnika možemo dodavati vlasnika tako da upišemo vlasnikovo ime, prezime i proviziju i pritiskom na gumb Dodaj Vlasnika, vlasnik je dodan u bazu podataka i možemo ga pronaći u popisu vlasnika na istom sučelju.

Vlasnike možemo uređivati tako da označimo vlasnika i pritisnemo gumb Uredi. Nakon što smo zadovoljni za izmjenama pritiskom na gumb Uredi Vlasnika spremamo izmjene u bazi podataka.

Vlasnike možemo brisati tako da označimo vlasnika i pritisnemo gumb Izbriši.

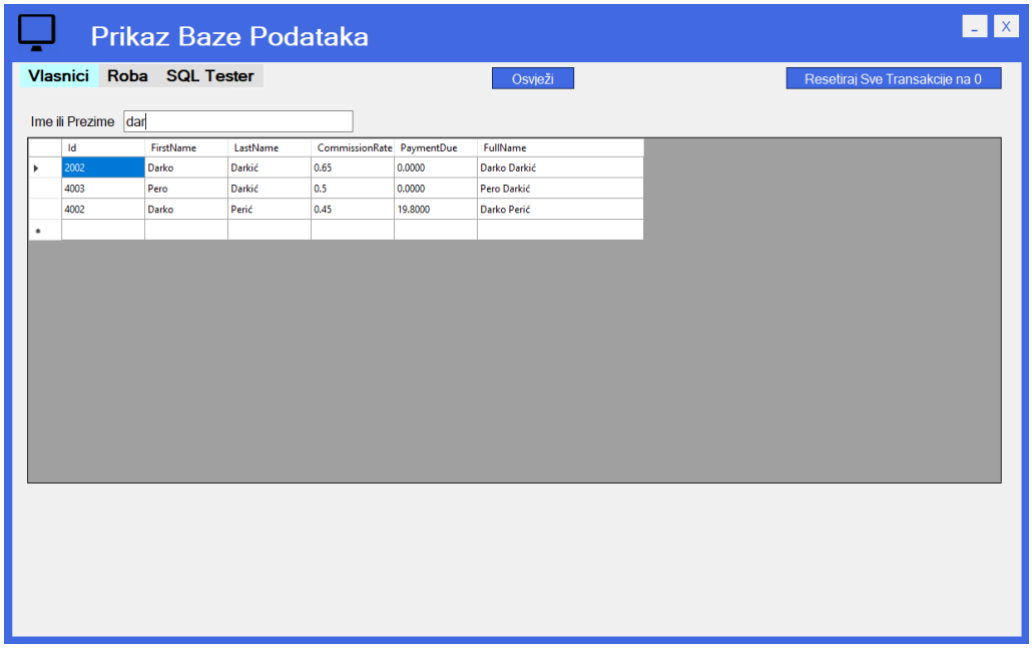
Možemo evidentirati da je dug vlasniku plaćen pritiskom na gumb Plati Dug.



Slika 81. Prikaz rada aplikacije - Uređivanje i dodavanje Vlasnika

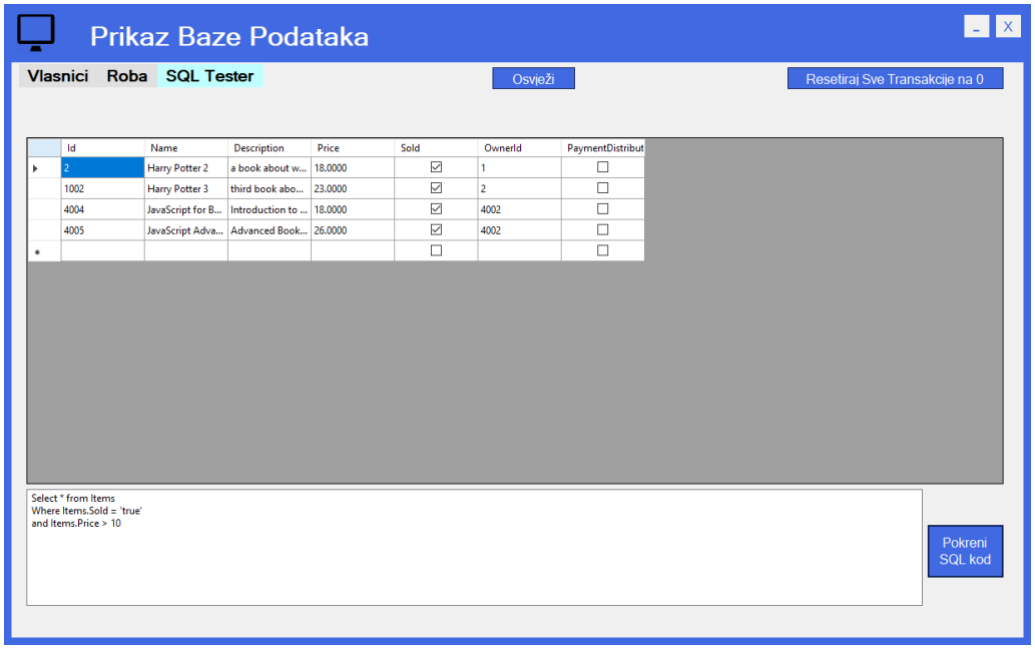
U formi Prikaz Baze podataka imamo prikaz podataka iz baze podataka u obliku tablice. Imamo prikaz tablice Vlasnika, Robe i formu za izvršavanje SQL koda

Vlasnike i robu možemo pretraživati unosom imena/prezimena vlasnika ili naziva robe u polje za unos teksta.



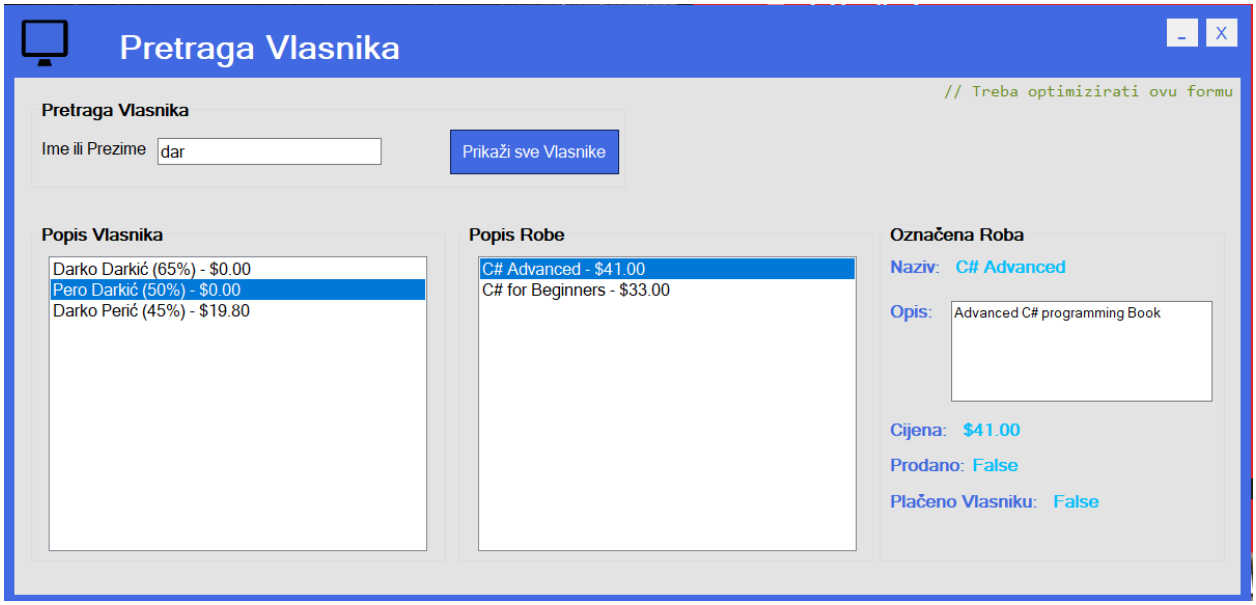
Slika 82. Prikaz rada aplikacije - prikaz baze podatka

SQL kod možemo izvršiti unosom SQL koda u polje za unos teksta i pritiskom na gumb Pokreni SQL kod izvršavamo SQL kod.



Slika 83. Prikaz rada aplikacije - SQL tester

U formi za pretragu vlasnika možemo pretraživati robu po vlasnicima tako da u polje za unos teksta unesemo ime ili prezime vlasnika čiju robu želimo vidjeti u popisu robe. Robu možemo označiti i prikazat će nam se informacije u robi sa desne strane sučelja. Također možemo prikazati sve vlasnike pritiskom na gumb „Prikaži sve vlasnike“.



Slika 84. Prikaz rada aplikacije - Pretraga vlasnika

8. Zaključak

Rad pridonosi razumijevanju važnosti prodaje i maloprodaje kao gospodarskih aktivnosti, kao i trendova i izazova koji utječu na njihov razvoj. Rad također pridonosi razumijevanju osnovnih pojmova i vrsta baza podataka i programskih jezika, kao i njihovih prednosti i nedostataka u razvoju aplikacija. Rad posebno ističe značajke i mogućnosti Microsoft SQL baze podataka i C# programskog jezika, koji su korišteni za izradu aplikacije. U izradi rada naišlo se na nekoliko problema, koji su uglavnom vezani za korištenje lokalne baze podataka, koja nije na serveru koji je mrežno povezan ili na oblaku, već se nalazi na lokalnom kompjuteru sa kojeg se pokreće aplikacija. To potencijalno može ugroziti sigurnost podataka u bazi podataka, kao i dovesti do nedosljednosti i neažurnosti podataka ako se aplikacija distribuira na više računala u trgovini. Stoga, preporuča se da se baza podataka preseli na samostalni server koji će biti dodatno osiguran od bilo kakvih vanjskih prijetnji, kao i da se izvorni kod aplikacije prilagodi da se spaja na taj server.

8.1. Moguća proširenja

U ovom diplomskom radu razvijena je aplikacija za komisijsku prodaju u trgovini rabljene robe, koja omogućuje evidenciju vlasnika i robe, prodaju robe, izdavanje računa i izvještaja, te pregled i pretraživanje podataka. Aplikacija je izrađena u programskom jeziku C# i koristi bazu podataka Microsoft SQL Server. Aplikacija je namijenjena za korištenje na stolnim računalima u trgovini i skladištu. Aplikacija ima nekoliko funkcionalnosti koje olakšavaju i ubrzavaju rad s podacima, kao što su autentifikacija i autorizacija korisnika, unos podataka pomoću barkod čitača, filtriranje i sortiranje podataka, te izvoz podataka u Excel datoteke. Aplikacija također ima grafičko sučelje koje je jednostavno i intuitivno za korištenje. Uza sve značajke razvijene aplikacije, postoji nekoliko mogućnosti i načina poboljšanja i proširenja aplikacije, poput uvođenja obavijesti (notifikacija), sigurnosna kopija baze podataka, te izrada web aplikacije.

Uvođenje obavijesti ili notifikacija je funkcionalnost koja bi omogućila da se korisnici obavijeste o promjenama u stanju robe, kao što su prodaja, rezervacija ili povrat. Na primjer, kada blagajnik na svom računalu evidentira prodaju robe, u skladištu na računalu se pojavi notifikacija da je roba prodana, što bi bio znak da se roba može zapakirati i dati kupcu ako želi preuzeti. Zaključno, ova bi funkcionalnost poboljšala komunikaciju i koordinaciju između zaposlenika u trgovini i skladištu. Sigurnosna kopija baze podataka omogućila bi da se svi trenutni podaci u bazi podataka pohrane u sigurnosnu kopiju baze podataka, čime bi se osiguralo da se nesmetano obavljaju razne operacije s bazom podataka, jer ako dođe do pogreške korisnika koji obavlja ili testira razne operacije, korisnik ima sigurnosnu kopiju baze podataka na koju može računati. Također, ako dođe do bilo kakvih drugih problema ili oštećenja podataka, uvijek je korisno i poželjno imati sigurnosnu kopiju važnih i povjerljivih podataka kao što su podaci u bazi podataka.

Izrada web aplikacije pružila bi mogućnost da se aplikacija koristi kao web aplikacija, čime bi se omogućio pristup bazi podataka s bilo koje lokacije i računala na kojem prethodno nije instalirana aplikacija. Iako bi to značilo obvezan pristup internetu i bazu podataka koja je izložena na internetu, što u slučaju samo stolne aplikacije ne mora biti nužno. Sveukupno, razvijena aplikacija predstavlja koristan alat za upravljanje komisijskom prodajom u trgovini rabljene robe, koji olakšava i ubrzava rad s podacima, te pruža pregledan i jednostavan način praćenja stanja robe i financija. Aplikacija se može nadograditi i prilagoditi različitim potrebama i zahtjevima korisnika, te tako postati još bolja i kvalitetnija.

8.2. Mogući problemi u izradi projekta

Ova aplikacija koristi lokalnu bazu podataka, tj. bazu podataka koja nije na serveru koji je mrežno povezan ili na oblaku, već se nalazi na lokalnom kompjuteru sa kojeg se pokreće aplikacija. To potencijalno može ugroziti sigurnost podataka u bazi podataka, jer bi neovlašteni korisnici mogli pristupiti ili oštetiti podatke ako imaju fizički pristup računalu. Stoga,

moguće je unaprjeđenje aplikacije da se baza podataka preseli na samostalni server (server koji je mrežno povezan sa infrastrukturom ili server na oblaku kod jednog od pružatelja takvih servisa) koji će biti dodatno osiguran od bilo kakvih vanjskih prijetnji. Ipak, to bi poboljšanje značilo i dodatne financijske izdatke.

Problem je, također, i u tome što kod distribucije ove aplikacije na razna računala u trgovini, svako bi računalo koje koristi ovu aplikaciju imalo svoju lokalnu bazu podataka, što znači da će se bilo koja operacija koju korisnik izvede na svom računalu pohraniti podatke samo na svojoj lokalnoj bazi podataka, što bi značilo da korisnici aplikacije ne dijele jedno te istu bazu podataka i one nisu sinkronizirane, već svaka operira zasebno, pa bi korisnici vidjeli samo one promjene koje su oni napravili, a ne i drugi korisnici na svojim računalima. Konačno, to bi dovelo do nedosljednosti i neažurnosti podataka, što bi moglo utjecati na točnost i pouzdanost aplikacije. Također, to bi otežalo backup i oporavak podataka u slučaju gubitka ili kvara računala.

S namjerom rješavanja ovih problema trebalo bi osigurati poseban server za bazu podataka na koji bi se sva računala sa kojih se aplikacija pokreće spojili na jedan server sa bazom podataka, i u skladu sa time podesiti izvorni kod aplikacije da se spaja na taj server. Time bi se postigla centralizacija i sinkronizacija podataka, što bi omogućilo bolju kontrolu i zaštitu podataka, kao i lakše backup i oporavak podataka. Također, to bi poboljšalo performanse i skalabilnost aplikacije, jer bi se smanjilo opterećenje pojedinačnih računala.

9. Popis korištene literature

1. Abiteboul S., Hull R., Vianu V. (1995). Foundations of Databases. Addison-Wesley, Reading MA.
2. Brčić-Stipčević, V., & Renko, S. (2007). Čimbenici utjecaja na izbor maloprodajnih oblika. Zbornik Ekonomskog fakulteta u Zagrebu, 5(1), 387-401.
3. Cheng, L., Liu, F. i Yao, D. (2017) Enterprise data breach: causes, challenges, prevention, and future directions. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 7(5).
4. Darwen, H. (2010). An Introduction to Relational Database Theory. Ventus Publishing ApS.
5. Elsmasri, R., Navathe, S. (2015). Fundamentals of Danabase Systems. 7th Edition. Boston: Pearson.
6. Gunjal, B. (2003). Database System: Concepts and Design, V Proceedings of 24th IASLIC-SIG. Sundargarh: National instutute of Technology Rourkela, str.1-20.
7. Hoffer, J., Prescott, M., McFadden, F. (2007). Modern database management. 8th Edition, New Jersey: Prentice-Hall.
8. Narodne novine. (2008). Zakon o trgovini. https://narodne-novine.nn.hr/clanci/sluzbeni/2008_07_87_2499.html
9. Narodne novine. (2009). Pravilnik o minimalno tehničkim uvjetima za poslovne prostorije u kojima se obavlja trgovina i posredovanje u trgovini i uvjetima za prodaju robe izvan prodavaonica. https://narodne-novine.nn.hr/clanci/sluzbeni/2009_06_66_1632.html
10. Korbar, D. (2010). Administriranje baza podataka. Zagreb: Algebra d.o.o.
11. Kovačević, Ž. (2018). Modeliranje, implementacija i administracija baza podataka. Zagreb: Tehničko veleučilište u Zagrebu.
12. Kovačević, Ž. i Stojanović, A. (2022). Noviteti standarda C++20. Polytechnic and design, 10 (1), 75-83.
13. Kramberger, T., Duk, S., Kovačević, R. (2018). Baze podataka. Zagreb: Priručnici tehničkog veleučilišta u Zagrebu.
14. Lovreničić A., Konecki M. (2017), „Programiranje u 14 lekcija“, Sveučilište u Zagrebu Fakultet organizacije i informatike, 224-225
15. Lovrenčić A., Konecki M., Orehovački T. (2009.): 1957-2007: 50 Years of Higher Order Programming Languages, Zagreb: FOI, Journal of Information and Organizational Sciences
16. Manager R. (2012). Baze podataka. Zagreb: Element.
17. Maleković, M., Rabuzin K. (2016). Uvod u baze podataka. Varaždin: Fakultet organizacije i informatike, udžbenik.
18. Melton, J. (2016). Database Language SQL, Handbook on Architectures of Information Systems, Berlin: Springer, str. 103–128.
19. Segetlija, Z., & Lamza Maronić, M. (1999). Suvremene informatičke tehnologije i razvitak maloprodaje. Ekonomski vjesnik, 12(1-2), 129-134.
20. Slakoper, Z. (2007). Prava i obveze strana komisijskog ugovora (I. dio). Hrvatska pravna revija, 7(10), 17-34.
21. Sommerville, I. (2011). Software engineering (9th ed.). Pearson Education.
22. Šamanović, J. (2009). Prodaja, distribucija, logistika. Ekonomski fakultet Split.
23. Trampus, Z. (2007). Računovodstvo prodaje robe u komisiji. Računovodstvo porezi u praksi, 16(11), 29-41.
24. Varga, M. (2012). Upravljanje podacima. Zagreb: Element.

10. Popis Ilustracija