**SREDNJA ELEKTROTEHNIČKA ŠKOLA SARAJEVO**

**MATURSKI RAD**

**TEMA: Elektronsko plaćanje**

**Mentor: Učenik:**

**Adnan Delić dipl. el. ing. Faruk Zahiragić**

**Sarajevo, juni 2022.**

Sadržaj

[**1. Uvod** 3](#_Toc101646445)

[**1.1 Šta su aplikacijski programi?** 4](#_Toc101646446)

[**1.2 Primjena računarskih aplikacija** 4](#_Toc101646447)

[**2. Razvojno okruženje** 4](#_Toc101646448)

[**2.1 C# (C-sharp)** 5](#_Toc101646449)

[**2.2 Microsoft Visual Studio** 5](#_Toc101646450)

[**2.3 Microsoft SQL Server** 6](#_Toc101646451)

[**2.4 Crystal Reports** 7](#_Toc101646452)

[**3. Osnove objektno-orijentisanog programiranja** 7](#_Toc101646453)

[**3.1 Šta je objektno-orijentisano programiranje?** 8](#_Toc101646454)

[**3.2 Struktura objektno-orijentisanog programiranja** 8](#_Toc101646455)

[**3.3 Glavni principi OOP-a** 9](#_Toc101646456)

[**3.4 Prednosti OOP-a** 10](#_Toc101646457)

[**4. Aplikacija „Elektronsko plaćanje“** 11](#_Toc101646458)

[**4.1 Osnovne značajke aplikacije** 12](#_Toc101646459)

[**4.2 Funkcionalnosti aplikacije** 12](#_Toc101646460)

[**5. Programski kôd** 13](#_Toc101646461)

[**5.1 Početna forma** 14](#_Toc101646462)

[Slika 1. Izgled početne forme 14](#_Toc101646463)

[Slika 2. Kôd za otvaranje child forme 14](#_Toc101646464)

[**5.2 Forma proizvodi** 14](#_Toc101646465)

[Slika 3. Izgled forme proizvodi 15](#_Toc101646466)

[Slika 4. Kôd za popunjavanje DataGridView-a 16](#_Toc101646467)

[**5.3 Novi račun forma** 17](#_Toc101646468)

[Slika 5. Izgled forme za dodavanje novog računa 17](#_Toc101646469)

[Slika 6. Kôd za dodavanje artikla u DataGridView 18](#_Toc101646470)

[Slika 7. Kôd za spašavanje računa 19](#_Toc101646471)

[**5.4 Prikazivanje računa** 20](#_Toc101646472)

[Slika 8. Izgled forme za prikaz računa 20](#_Toc101646473)

[Slika 9. Kôd za filtriranje prikazanih računa 21](#_Toc101646474)

[Slika 10. Kôd za ažuriranje računa 21](#_Toc101646475)

[**5.5 Štampanje računa** 22](#_Toc101646476)

[Slika 11. Kôd za štampanje računa 22](#_Toc101646477)

[Slika 12. Kôd za štampanje izvještaja 22](#_Toc101646478)

[**ZAKLJUČAK** 23](#_Toc101646479)

[**MIŠLJENJE O RADU** 24](#_Toc101646480)

[**LITERATURA** 25](#_Toc101646481)

# **1. Uvod**

## **1.1 Šta su aplikacijski programi?**

Aplikacijski program (skraćeno aplikacija) je računarski program dizajniran za obavljanje specifičnog zadatka koji se ne odnosi na rad samog računara, koji obično koriste krajnji korisnici. Aplikacije mogu biti dio paketa sistemskih programa računara ili mogu biti zasebno objavljene i kodirane kao vlasničke aplikacije, aplikacije otvorenog kôda ili kao dio projekta.

Postoji mnogo različitih i alternativnih načina klasifikacije aplikativnog softvera. Sa pravne tačke gledišta, aplikativni softver se uglavnom klasifikuje sa pristupom crne kutije, u odnosu na prava krajnjih korisnika ili pretplatnika. Softverske aplikacije se također klasifikuju u odnosu na programski jezik na kojem je napisan ili izvršen izvorni kôd, kao i u odnosu na njihovu svrhu i rezultate.

## **1.2 Primjena računarskih aplikacija**

Računarske aplikacije imaju široku primjenu. Primjeri upotrebe računarskih aplikacija uključuju: softver za informacijske radnike, softver za zabavu, obrazovni softver, softver za infrastrukturu preduzeća, softver za simulaciju, softver za razvoj medija, softver za inžinjering proizvoda kao i softverski inžinjering uopćeno.

Aplikacije se često primjenjuju i kao softver koji pomaže radnicima da obave određeni zadatak. U takve svrhu se koriste računovodstvene aplikacije, aplikacije za upravljanje podacima, aplikacije za obradu teksta, kao i razni aplikacijski programi za upravljanje radnika, radnih aktivnosti i radnog toka (workflow software).

Kao softver za infrastrukturu preduzeća računarske aplikacije se primjenjuju u vidu sistema za upravljanje bazom podataka, softvera za upravljanje dokumentima, softvera za upravljanje digitalnim sredstvima, softvera za upravljanje poslovnim procesima, geografskih informacionih sistema, kao i aplikacija umjetne inteligencije za IT operacije.

# **2. Razvojno okruženje**

## **2.1 C# (C-sharp)**

C# je objektno-orijentisani programski jezik opće namjene. C# obuhvata imperativne, deklarativne, funkcionalne, generične, objektno-orijentisane i komponentno-orijentisane programske discipline. C# je dizajnirao Anders Hejlsberg iz Microsofta 2000. godine, a kasnije je odobren kao međunarodni standard od strane Ecme 2002. godine i ISO standarda 2003. Microsoft je predstavio C# zajedno sa .NET frameworkom i Visual Studiom, od kojih su oba bila zatvorenog izvora (closed-source). Microsoft ga je razvio kao nadogradnju na svoj C++, što se iz C# logoa može primjetiti kao kombinacija četiri „+“ znaka u jedan hash znak. Dok je C++ isključivo objektno-orijentisani programski jezik, C# je također i komponentno-orijentisani jezik, što ga čini fleksibilnijim i jednostavnijim za rad. Također C# je značajno moderniji programski jezik, što ga čini pristupačnim za sve generacije programera od početnika do softverskih stručnjaka.

## **2.2 Microsoft Visual Studio**

Microsoft Visual Studio je integrisano razvojno okruženje (IDE) koga pravi Microsoft. Koristi se za razvoj računarskih programa za Windows, web-stranica, aplikacija i usluga. Koristi Microsoftove platforme za razvoj poput aplikativnih programskih interfejsa (IDE) za Windows, Windows Forms, Windows Presentation Foundation, Windows Store i Microsoft Silverlight. Može proizvesti nativni kôd i upravljani kôd (eng. managed code).

Visual Studio sadrži uređivač izvornog kôda koji podržava IntelliSense (komponenta koja predlaže ostatak kôda) kao i refaktoriranje kôda. Integrisani program za otklanjanje grešaka (debugger) radi na nivou izvornog i mašinskog kôda. Program također sadrži alate poput dizajnera oblika koji se koristi za pravljenje aplikacija sa grafičkim korisničkim interfejsom, web-dizajnera, dizajnera klasa i dizajnera shema baza podataka.

Visual Studio podržava različite programske jezike i dozvoljava uređivaču kôda i debuggeru da podržava gotovo bilo koji programski jezik, pod uvjetom da usluga za taj servis postoji. Ugrađeni jezici su: C, C++ i C++/CLI (preko Visual C++), VB.NET (preko Visual Basic .NET-a), C# (preko Visual C#) i F# (počevši od programa Visual Studio 2010). Podrška za ostale programske jezike poput Pythona, Rubya i M-a kao i ostalih dostupna je instalacijom jezičkih servisa koji se mogu zasebno instalirati. Također podržava XML/XSLT, HTML/XHMTL, CSS i JavaScript.

Microsoft besplatno pruža izdanja „Express“ programa Visual Studio. Komercijalna izdanja programa Visual Studio kao i određenja prethodna izdanja, studentima su dostupna preko Microsoftovog programa DreamSpark.

## **2.3 Microsoft SQL Server**

Microsoft SQL Server je sistem za upravljanje relacionim bazama podataka koji je razvio Microsoft. Kao server baze podataka to je softverski proizvod sa primarnom funkcijom pohranjivanja i preuzimanja podataka po zahtijevu druge softverske aplikacije koje mogu raditi na istom računaru ili na drugom računaru unutar mreže (uključujući Internet). Microsoft plasira na tržište najmanje desetak različitih izdanja Microsoft SQL Servera, namijenjenih različitoj publici i radnim opterećenjima u rasponu od malih aplikacija za jednu mašinu do velikih aplikacija okrenutih prema Internetu sa mnogo istovremenih korisnika.

Primarni jezik za upite kod Microsoft SQL Servera je Transact SQL (T-SQL), što znači da osim osnovnih i klasičnih (SELECT tipa) SQL upita dozvoljava i složenije stvari poput mijenjanja programskog toka (IF naredba) i slično.

Važan alat u sklopu Microsoft SQL Servera je i SQL Server Managment Studio. SQL Server Managment Studio je GUI (grafički korisnički interfejs) alat uključen u SQL Server koji služi za konfigurisanje, upravljanje i administriranje svih komponenti unutar Microsoft SQL Servera.

## **2.4 Crystal Reports**

Crystal Reports je aplikacija za poslovnu inteligenciju koju SAP prodaje malim i srednjim preduzećima. Crystal Reports omogućuje kreiranje formatiranih izvještaja sa različitih izvora podataka.

Crystal Reports se primjenjuje za generiranje dokumenata dostupnih na više jezika, kreiranje kontrolnih tabli sa mapama, logotipovima, slikama, grafikonima i tabelama. Također se koristi za pravljenje specifičnih izvještaja potrebnih za korporativno poslovanje kao što su: fakture, pisma, izjave, izvještaji o radu, izvještaji o prodaji, izvještaji o ljudskim resursima, finansijski izvještaji, promotivne kampanje i slično.

# **3. Osnove objektno-orijentisanog programiranja**

## **3.1 Šta je objektno-orijentisano programiranje?**

Objektno-orijentisano programiranje (OOP) je model kompjuterskog programiranja koji organizuje dizajn softvera oko podataka ili objekata, a ne funkcija i logike. Objekat se može definisati kao polje podataka koje ima jedinstvene atribute i ponašanje.

OOP se fokusira na objekte kojima programeri žele da manipulišu, a ne na logiku potrebnu da njima manipuliše. Ovaj pristup programiranju je prikladan za velike, složene programe koji se aktivno ažuriraju ili održavaju. Ovo uključuje programe za proizvodnju i dizajn, kao i mobilne aplikacije; na primjer, OOP se može koristiti za proizvodnju softvera za simulaciju sistema.

Organizacija objektno orijentisanog programa takođe čini metod korisnim za kolaborativni razvoj, gde su projekti podijeljeni u grupe. Dodatne prednosti OOP-a uključuju ponovnu upotrebu kôda, skalabilnost i efikasnost.

Prvi korak u OOP-u je prikupljanje svih objekata kojima programer želi da manipuliše i identifikovanje njihovog međusobnog odnosa - vježba poznata kao modeliranje podataka.

Primjeri objekta mogu se kretati od fizičkih entiteta, kao što je ljudsko biće koje je opisano svojstvima kao što su ime i adresa, do malih kompjuterskih programa, kao što su widgeti.

Jednom kada je objekat poznat, on se označava klasom objekta koja definiše vrstu podataka koje sadrži i sve logičke sekvence koje mogu njime manipulisati. Svaki poseban logički niz poznat je kao metoda.

## **3.2 Struktura objektno-orijentisanog programiranja**

Struktura ili gradivni blokovi objektno-orijentisanog programiranja uključuju sljedeće:

**Klase** su korisnički definisani tipovi podataka koji djeluju kao nacrt za pojedinačne objekte, atribute i metode.

**Objekti** su instance klase kreirane sa specifično definisanim podacima. Objekti mogu odgovarati objektima iz stvarnog svijeta ili apstraktnom entitetu. Kada je klasa prvobitno definisana, opis je jedini objekat koji je definisan.

**Metode** su funkcije koje su definisane unutar klase koje opisuju ponašanja objekta. Svaka metoda sadržana u definicijama klase počinje referencom na objekt instance. Dodatno, potprogrami sadržani u objektu nazivaju se metodama instance. Programeri koriste metode za ponovnu upotrebu ili održavanje funkcionalnosti inkapsulirane unutar jednog po jednog objekta.

**Konstrukor** je posebna vrsta metode koja se poziva za kreiranje objekta. On priprema novi objekat za upotrebu, često prihvatajući argumente koje konstruktor koristi za postavljanje potrebnih varijabli člana.

**Atributi** su definisani u predlošku klase i predstavljaju stanje objekta. Objekti će imati podatke pohranjene u polju atributa. Atributi klase pripadaju samoj klasi.

## **3.3 Glavni principi OOP-a**

Objektno-orijentisano programiranje se zasniva na sljedećim principima:

**Enkapsulacija**. Ovaj princip kaže da su sve važne informacije sadržane unutar objekta i da su izložene samo odabrane informacije. Implementacija i stanje svakog objekta privatno se drže unutar definisane klase. Drugi objekti nemaju pristup ovoj klasi niti ovlaštenje za unošenje promjena. Oni mogu samo pozvati listu javnih funkcija ili metoda. Ova karakteristika skrivanja podataka pruža veću sigurnost programa i izbjegava nenamjerno oštećenje podataka.

**Apstrakcija**. Objekti otkrivaju samo unutrašnje mehanizme koji su relevantni za korištenje drugih objekata, skrivajući svaki nepotreban implementacijski kôd. Izvedena klasa može imati proširenu funkcionalnost. Ovaj koncept može pomoći programerima da lakše naprave dodatne promjene ili dodatke tokom vremena.

**Nasljedstvo**. Klase mogu ponovo koristiti kôd iz drugih klasa. Odnosi i podklase između objekata mogu se dodijeliti, omogućavajući programerima da ponovo koriste zajedničku logiku dok i dalje održavaju jedinstvenu hijerarhiju. Ovo svojstvo OOP-a forsira temeljitiju analizu podataka, smanjuje vrijeme razvoja i osigurava viši nivo tačnosti.

**Polimorfizam**. Objekti su dizajnirani da dijele ponašanje i mogu poprimiti više od jednog oblika. Program će odrediti koje je značenje ili upotreba potrebno za svako izvršavanje tog objekta iz nadređene klase, smanjujući potrebu za dupliranjem kôda. Zatim se kreira podređena klasa, koja proširuje funkcionalnost roditeljske klase. Polimorfizam omogućava različitim tipovima objekata da prođu kroz isti interfejs.

## **3.4 Prednosti OOP-a**

Prednosti OOP-a uključuju:

**Modularnost**. Enkapsulacija omogućava objektima da budu samostalni, što olakšava rješavanje problema i kolaborativni razvoj.

**Ponovna upotreba**. Kôd se može ponovo koristiti putem nasljeđivanja, što znači da tim ne mora pisati isti kôd više puta.

**Produktivnost**. Programeri mogu brže konstruisati nove programe upotrebom više biblioteka i kôda koji se može ponovo koristiti.

**Lahka nadogradnja i skalabilnost**. Programeri mogu samostalno implementirati sistemske funkcionalnosti.

**Opisi interfejsa**. Opisi eksternih sistema su jednostavni, zbog tehnika prenošenja poruka koje se koriste za komunikaciju objekata.

**Sigurnost**. Koristeći enkapsulaciju i apstrakciju, složeni kôd je skriven, održavanje softvera je lakše i internet protokoli su zaštićeni.

**Fleksibilnost**. Polimorfizam omogućava da se jedna funkcija prilagodi klasi u koju je smještena. Različiti objekti također mogu proći kroz isti interfejs.

# **4. Aplikacija „Elektronsko plaćanje“**

## **4.1 Osnovne značajke aplikacije**

Aplikacija Elektronsko plaćanje omogućava korisnicima da dinamički upravljaju podacima vezanim za njihovu trgovinu. Namijenjena je da olakša finansijsko poslovanje i analizu poslovanja trgovina koje koriste ovaj softver.

Elektronsko plaćanje je realizovano kao više-prozorska aplikacija. Pomoću svojstva IsMdiContainer (Multiple-document interface), početni prozor je trajno prikazan u pozadini, dok se individualni child containeri glavne Mdi Parent forme mogu nezavisno otvarati i zatvarati, bez uticaja na rad aplikacije.

## **4.2 Funkcionalnosti aplikacije**

Elektronsko plaćanje ima više funkcionalnosti.

Prva funkcionalnost je dodavanje i ažuriranje artikala u bazi podataka. Pomoću ove značajke korisnici mogu, bez naprednijeg znanja o bazama podataka, jednostavno dodavati i upravljati artiklima u bazi podataka.

Sljedeća funkcionalnost je kreiranje računa. Ova značajka omogućava korisnicima jednostavno dodavanje artikala na račun. Korisnik unosi željeni artikal i količinu istog, a cijena i ostali relevantni podaci se automatski unose na račun. Kada korisnik kreira račun moguće je isti i odštampati.

Treća funkcionalnost je prikazivanje i ažuriranje već napravljenih računa. Trgovci se često nađu u potrebi da pregledaju već izdate računa, nerijetko zbog greški prilikom unosa podataka na račun. Pomoću ove funkcionalnosti korisnici mogu veoma lahko steći uvid u već napravljene račune, te u slučaju eventualnih greški prilikom unosa, ažurirati odabrani račun.

Posljednja funkcionalnost je štampanje izvještaja prodaje. Pomoću ove funkcionalnosti korisnik može štampati određeni broj računa, filtriran na osnovu parametara kao što su broj računa, ili datum izdavanja računa. Ova funkcionalnost korisnicima omogućava uvid u obim poslovanja u određenom periodu, kao i u ukupnu zaradu u određenom periodu.

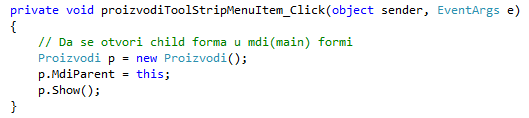
# **5. Programski kôd**

## **5.1 Početna forma**



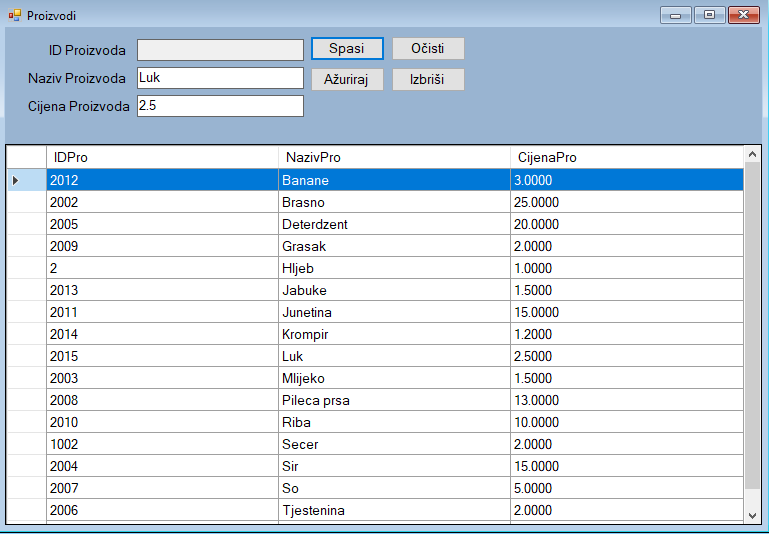
Slika 1. Izgled početne forme

Pomoću ToolStripMenua možemo otvarati pojedinačne child forme aplikacije.



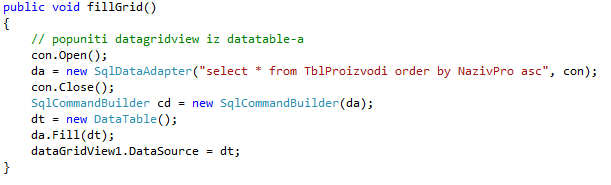
Slika 2. Kôd za otvaranje child forme

## **5.2 Forma proizvodi**



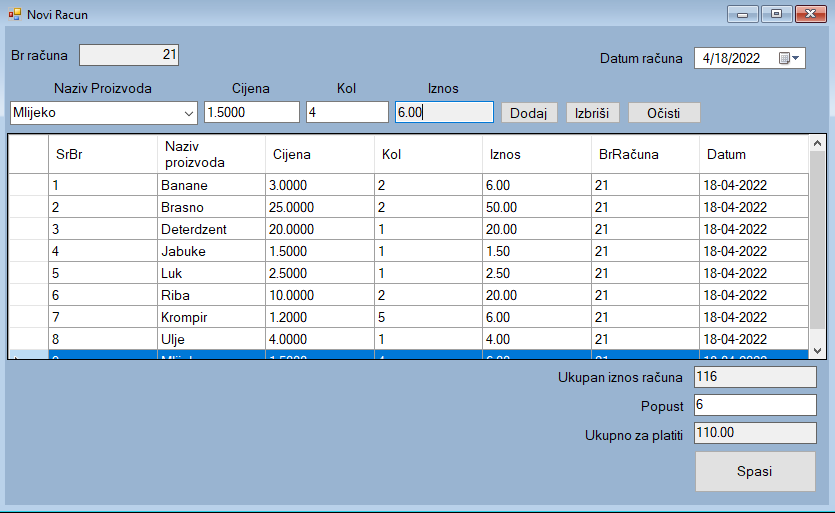
Slika 3. Izgled forme proizvodi

U formi proizvodi se nalazi prikaz pojedinačnih artikala u bazi podataka. Pri učitavanju forme inicijalizira se metoda fillGrid() koja popunjava DataGridView sa artiklima. Artikle dodajemo tako što navedemo validno ime i cijenu, a ID artikla se automatski generiše. Klikom na dugme „Spasi“ izvršava se Sql komanda koja pomoću INSERT upita dodaje artikal u tabelu proizvodi baze podataka. Odabirom artikla u DataGridView njegove informacije se prikazuju u textbox-ovima, te ih možemo po potrebi izmijenjivati. Slično kao i spasi, klikom na dugme „Ažuriraj“ izvršava se Sql komanda koja pomoću UPDATE upita ažurira odabrani artikal sa informacijama iz TextBoxova. Dugme „Izbriši“ izvršava DELETE upit, te briše odabrani artikal.



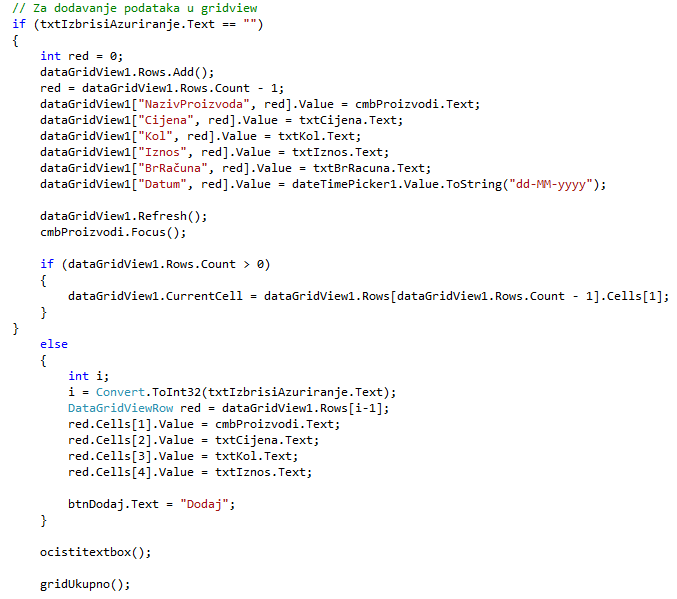
Slika 4. Kôd za popunjavanje DataGridView-a

## **5.3 Novi račun forma**



Slika 5. Izgled forme za dodavanje novog računa

U formi novi račun kreiramo račune. Tokom učitavanja forme ComboBox se popunjava sa artiklima iz tabele „TblProizvodi“ pomoću SELECT upita. Pri dodavanju artikala na račun potrebno je odabrati odgovarajući artikal iz polja ComboBox, te navesti željenu količinu artikla. Pritiskom na dugme „Dodaj“ DataGridView se ažurira sa unesenim podacima. Također možemo odabrati već uneseni artikal na račun, te ga ažurirati ili eventualno obrisati pritiskom na dugme „Izbriši“. Dugme „Očisti“ briše uneseni sadržaj iz TextBox-ova. Iznos računa se automatski računa, te možemo dodati i eventualni popust. Pritiskom na dugme „Spasi“ izvršava se INSERT upit u dvije tabele. U prvoj se dodaju opći podaci o računu tj. iznos, datum izdavanja, broj računa itd., dok se u drugu tabelu dodaju podaci iz svakog reda DataGridViewa. Nakon izvršavanja upita, prikazuje se forma „Štampaj račun“ koja prikazuje račun spreman za štampanje.

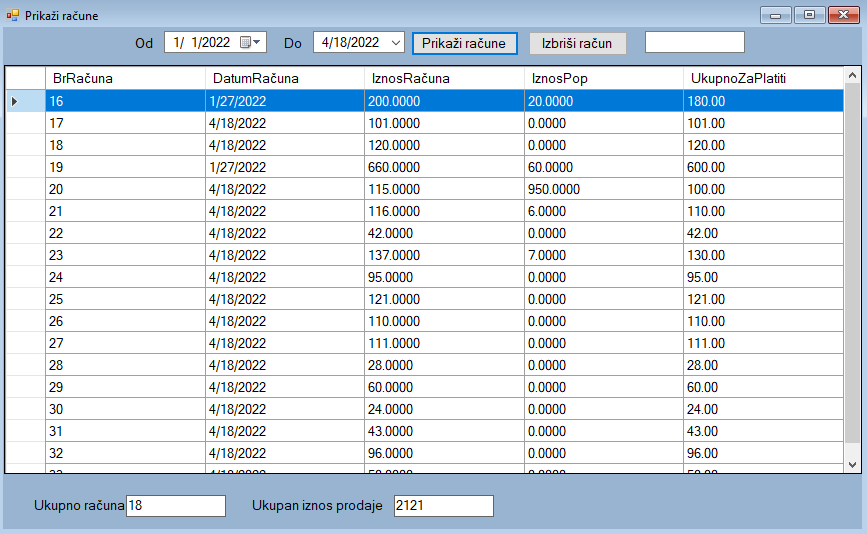


Slika 6. Kôd za dodavanje artikla u DataGridView



Slika 7. Kôd za spašavanje računa

## **5.4 Prikazivanje računa**



Slika 8. Izgled forme za prikaz računa

Pritiskom na opciju „Prikaži račun“ unutar ToolStripMenua prikazuje se forma „Prikaži račune“. Ova forma incijalno prikazuje sve do sada izdate račune. Prikazani računi se mogu filtrirati pomoću DateTimePicker polja, preko kojih filitriramo račune na osnovu datuma izdavanja. Račun se može izbrisati tako što se u TextBox unese broj računa. Pritiskom na dugme „Izbriši račun“ izvršava se DELETE upit u bazi podataka, te se navedeni račun briše. Račun možemo ažurirati tako što dvaput kliknemo na željeni račun. Tada se prikazuje forma „Ažuriraj račun“. Ova forma je funkcionalno slična formi „Novi račun“ sa bitnom razlikom da umjesto INSERT upita izvršava upit UPDATE u bazu podataka.

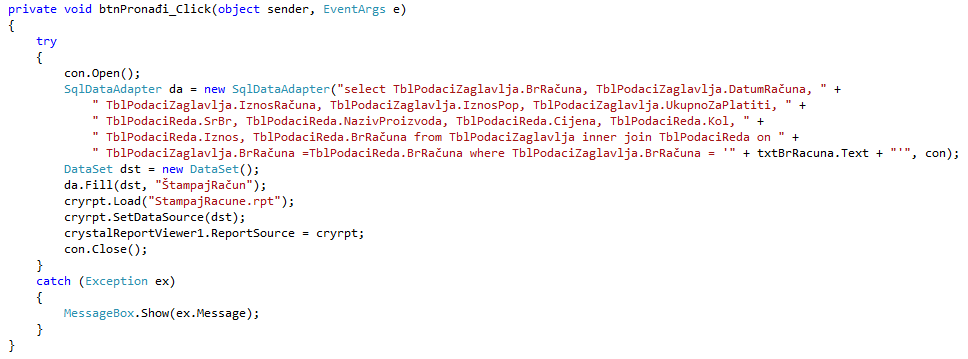


Slika 9. Kôd za filtriranje prikazanih računa



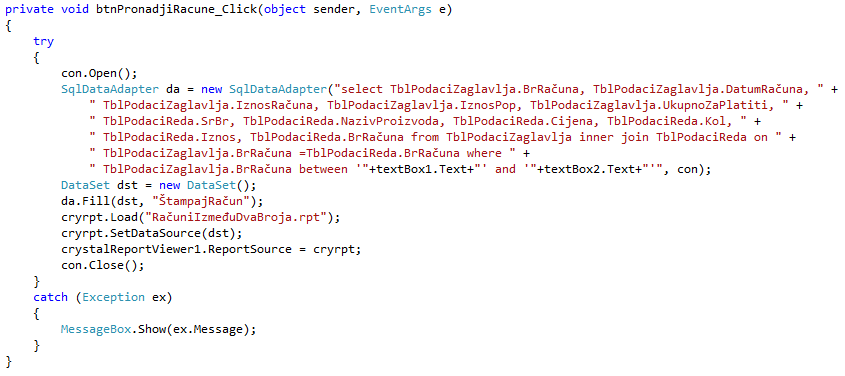
Slika 10. Kôd za ažuriranje računa

## **5.5 Štampanje računa**



Slika 11. Kôd za štampanje računa

U formi za štampanje računa možemo, na osnovu unesenog broja računa štampati račun. Računi se štampaju preko softvera Crystal Reports koji automatski generiše template za štampanje. Također možemo štampati više računa na jedan izvještaj prodaje. Izvještaji se mogu pisati tako što filtriramo račune na osnovu broja računa ili datuma izdavanja računa.



Slika 12. Kôd za štampanje izvještaja

# **ZAKLJUČAK**

Aplikacijski programi imaju široku primjenu u mnogim oblastima poslovnih aktivnosti. Razvojno okruženje C# i Visual Studio omogućava kreiranje raznovrsnih aplikacija za olakšavanje rada, koje, uparene sa SQL Serverom znatno olakšavaju upravljanje velikim količinama podataka. Korištenjem Objektno-orijentisanog programiranja naš kôd postaje jednostavniji i efikasniji, jer svaki segment naše aplikacije posmatramo kao jedan objekat vezan sa ostalim dijelovima aplikacije.

Aplikacija „Elektronsko plaćanje“ predstavlja samo jedan primjer korištenja navedenih tehnologija u svrhu olakšavanja upravljanja podacima jednog preduzeća. Pomoću svojstva IsMdiContainer omogućeno je istovremeno prikazivanje više prozora. Aplikacija omogućava dodavanje novih artikala u bazu podataka. Artikli se mogu kasnije dodavati na račun, te se kreirani račun može štampati. U formi „Prikaz računa“ dobijamo pristup svim izdatim računima, te tu možemo i ažurirati već izdati račun. Račune možemo i štampati tako što navedemo broj računa koji želimo štampati. Također možemo štampati i izvještaje prodaje tako što odaberemo više računa koje želimo štampati, filtriranih po datumu ili broju računa.

Ovakav tip aplikacije se često koristi u realnim uslovima za upravljanje podacima u mnogim trgovinama.

# **MIŠLJENJE O RADU**

**Potpis mentora:**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Adnan Delić dipl. el. ing.

# **LITERATURA**

<https://v2cloud.com/glossary/what-is-a-desktop-app>

<https://en.wikipedia.org/wiki/C_Sharp_(programming_language)>

<https://bs.wikipedia.org/wiki/Microsoft_Visual_Studio>

<https://en.wikipedia.org/wiki/Microsoft_SQL_Server>

<https://en.wikipedia.org/wiki/Crystal_Reports>

<https://www.techtarget.com/searchapparchitecture/definition/object-oriented-programming-OOP>