# Realizacija baze podataka i razvoj programa u .NET razvojnom okruženju

**Osman, Martin**

VELEUČILIŠTE U ŠIBENIKU

ODJEL MENADŽMENTA

PREDDIPLOMSKI STRUČNI STUDIJ MENADŽMENT

**Martin Osman**

# REALIZACIJA BAZE PODATAKA I RAZVOJ PROGRAMA U .NET RAZVOJNOM OKRUŽENJU

**Završni rad**

Šibenik, srpanj 2018.

VELEUČILIŠTE U ŠIBENIKU

ODJEL MENADŽMENTA

PREDDIPLOMSKI STRUČNI STUDIJ MENADŽMENT

# REALIZACIJA BAZE PODATAKA I RAZVOJ PROGRAMA U .NET RAZVOJNOM OKRUŽENJU

**Završni rad**

**Kolegij:** Baze podataka

**Mentor:** mr. sc. Ivan Livaja pred.

**Student:** Martin Osman

**Broj indeksa:** 15398153

Šibenik, srpanj 2018.

# SADRŽAJ

TEMELJNA DOKUMENTACIJSKA KARTICA

Veleučilište u Šibeniku
Odjel Menadžmenta
Preddiplomski stručni studij Menadžment

Završni rad

# REALIZACIJA BAZE PODATAKA I RAZVOJ PROGRAMA U .NET RAZVOJNOM OKRUŽENJU

MARTIN OSMAN
Siščani 128, 43240 Čazma;
martin_osman95@hotmail.com

## Sažetak rada

Tema završnog rada je izrada i opis razvoja Windows aplikacije u .NET razvojnom okruženju u svrhu korištenja iste na predavanjima Veleučilišta u Šibeniku.

Cilj je prikazati i opisati proces razvoja Windows aplikacije na na vlastitom primjeru, mogućnosti .NET tehnologije, te manipulacija bazama podataka kao i unos, pretraga i pohrana podataka u istima preko .NET grafičkog sučelja.

Završni rad u dogovoru sa mentorom nije izrađen klasičnom metodom, već je napravljen kao *tutorial* da bi mogao služiti praktično na predavanjima kao pomoć studentima za lakše razumijevanje gradiva. Izrada praktičnog dijela rada je detaljno opisana, potkrepljena primjerima kôda. U radu sam koristio Visual studio razvojno okruženje te Microsoft SQL server management studio. Pod stavkom „Dokumentacija“, opisane su osnovne mogućnosti i način uporabe programa za različitu vrstu korisnika.

Prvi korak u razvoju aplikacije je ideja. Nakon toga dolazi korak u kojem se detaljno analizira što aplikacija treba raditi i na koji način, te kako ona treba izgledati. Nakon dobre razrade ideje, dolazi dio u kojem je potrebno dio sa papira prebaciti u kôd. Ukoliko je plan dobro napravljen, dobro odrađena logika programa, i ukoliko se posjeduje dovoljno znanja, taj dio nije previše zahtjevan.

Windows aplikacija ima implementirano adminstratorsko sučelje kako bi se ograničio rad neautoriziranim osobama. Omogućuje se korisniku sa administratorskim ovlastima dodavati/brisati korisničke račune. Svi ostali korisnici kao i admin, imaju mogućnost unosa artikala u sustav, unosa proizvoda kao i normativa za svaki prizvod.

(48 stranica / 49 slika / 6 literaturnih navoda / jezik izvornika: hrvatski)

Rad je pohranjen u: Knjižnici Veleučilišta u Šibeniku

Ključne riječi: *Windows aplikacija, .NET, Visual Studio, SQL baze podataka*

Mentor: mr. sc. Ivan Livaja pred.

Rad je prihvaćen za obranu:

BASIC DOCUMENTATION CARD

Polytechnic of Šibenik
Department of Management
Professional Undergraduate Studies of Management

Final paper

# DATABASE REALIZATION AND PROGRAM DEVELOPMENT IN .NET DEVELOPING ENVIRONMENT

MARTIN OSMAN
Siščani 128, 43240 Čazma;
martin_osman95@hotmail.com

**Abstract**

Theme of the final assignment is development and description of development Windows application in .NET develop environment with purpose to use application in lectures.

The goal is to show and describe process of developing windows application on my own example, possibilities of .NET technology, and database manipulation like a input, search and storage data over .NET graphic user interface.

This paper in agreement with my mentor, is not made typically. Like a tutorial, it will help in lectures to my colleagues for easier understand and easier mastering C# and SQL curriculum.

Making a practical part is detailed described, supported by code examples in this paper. In development I used Microsoft Visual Studio environment and Microsoft SQL Server Management Studio. Under item 'Dokumentacija' are described basic features and '*how-to-use*' guide.

First step in application development is idea. After good idea, next step is detail analysis of application features and logic. If is this step made well, with a little knowledge in programming, next step, implementation, is not to demanding.

(48 pages / 49 figures / 6 references / original in Croatian language)
Paper deposited in: Library of Polytechnic in Šibenik
Keywords: *Windows application,.NET, Visual studio, SQL database*
Mentor: mr. sc. Ivan Livaja pred.
Paper accepted:

# 1. UVOD

Rapidnim rastom broja korisnika interneta, olakšanjem pristupa različitim alatima, rastom propusnosti internet, broj ljudi koji su informatički pismeni je naglo porastao. Velikom količinom različitih *how-to-do* stranica i videa, korisnici postaju sve zainteresiraniji te sami uče nove stvari te se usavršavaju u već poznatima. Osamnaest godina, kako se prati broj korisnika internet, taj broj je porastao ( na svjetskoj razini u odnosu na 2008. godinu) za nevjerojatnih 1052%.[1] O uključenosti interneta I društvenih mreža u sve sfere što malog čovjeka što velikih multinacionalnih kompanija govori iskustvena činjenica koja nam sama govori da je nemoguće zamisliti dan bez interneta ili mobitela. Sjednite u kafić i osvrnite se oko sebe. Vidjeti će te da velika većina ljudi stalno koristio mobitele. Prošećite ulicom, vidjeti ćete istu stvar. Istina je da tehnologija u jednu ruku olakšava život, no učinila nas je ovisnima. Ali to je nešto protiv čega se nemožemo boriti već prihvatiti to kao takvo i iz toga izvući maksimalnu korist. Uzmite u obzir da su internetske tehnologije, web, windows i aplikativni razvoj branše koje će u budućnosti imati i još veći razvoj nego dosada. Developeri postaju jedni od najtraženijih zanimanja, a uz to su i najbolje plaćeni.

U ovom radu će se prikazati primjer izrade 'Stock managmenta' za restoransko i poslovanje kafića. Biti će prikazan detaljan put izrade aplikacije, od ideje, do faze korištenja. Ovim radom želim pokazati svim studentima koliko se puno stvari može napraviti uz malo truda i puno volje.

U prvom dijelu će biti prikazana tehnologija izrade same aplikacije. Pod time se podrazumijevaju i definicije korištenih alata.

U drugom dijelu ćemo prikazati faze razvoja aplikacije. Dio koji će ujedino biti i najopširniji, pošto je u njemu prikazan moj put razvoja aplikacije. Svaka faza je detaljno objašnjena i potkrjepljena primjerima iz prakse i *screenshoot-ovima* dijelova kôda.

U drugome dijelu paralelno uz primjere kôd biti će I objašnjena logika programa, I način na koji radi.

Zaključak ću iskoristiti za riječ zahvale.

[1] Internet world stats, https://www.internetworldstats.com/stats.htm, 08.07.2018.

# 2. TEHNOLOGIJA IZRADE

Windows aplikacije se koriste već dugi niz godina, točnije od pojave grafičkog korisničkog sučelja (GUI). U početku su one bile jednostavne i obavljale primarne zadatke, do sve kompleksnijih aplikacija pa čak i igrica. Današnja tehonolgija je toliko napredovala, I toliko se radi na razvoju windows aplikacija da se trendovi mijenjaju skoro svakodnevno. U vremenu sve bržeg načina života, gdje ljudi žive ubrzano, i gdje nedostaje vremena, postoji niz aplikacija koje ljudima olakšavaju život. Upotrebom raznih senzora aplikacije mogu 'komunicirati' sa okolinom i sa korisnikom te tako mu olakšati svakodnevne brige. Primjenu aplikacija je razna, a ovaj završni rad je poslužio kao primjer izrade aplikacije za restoransko poslovanje. U nastavku su prikazani alati i tehnologije izrade završnog rada.

## 2.1. Mirosoft Visual Studio

'Microsoft Visual Studio je integrirano razvojno okruženje (IDE) koga pravi Microsoft. Koristi se za razvoj programa za Windows, web-stranica, aplikacija i usluga. Koristi Microsoftove platforme za razvoj poput aplikativnih programskih interface-a (API) 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. Integrirani 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 s grafičkom korisničkim interfejsom, veb-dizajnera, dizajnera klasa i dizajnera shema baza podataka.

Visual Studio podržava različite programske jezike i dozvoljava uređivaču kôda i debuggeru da podržava (u različitoj mjeri) gotovo bilo koji programski jezik, pod uvjetom da usluga za taj jezik 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 M-a, Pythona, i Rubyja kao i ostalih dostupna je instalacijom jezičkih servisa koji se mogu zasebno instalirati.'[2]

[2] Wikipedia, https://bs.wikipedia.org/wiki/Microsoft_Visual_Studio,08.07.2018.

Značajke Microsoft Visual Studija:

- Izgradite pametnije aplikacije brže
- Pronađite i lakše otklonite *bug-ove*
- Integriran cloud
- Učinkovito surađivanje
- Kvalitetne mobilne aplikacije
- Nadogradite svoj jezik
- Dostavite software brže[3]

Jedan od trenutno najboljih alata na tržištu za razvoj Windows aplikacija dolazi u nekoliko verzija. Zadnja verzija 2017. Godine, dolazi u 3 grupe:

- Enterprise 2017
- Professional 2017
- Community 2017

[3] VisualStudioMicrosoft, https://visualstudio.microsoft.com/vs/whatsnew/, 08.07.2018.

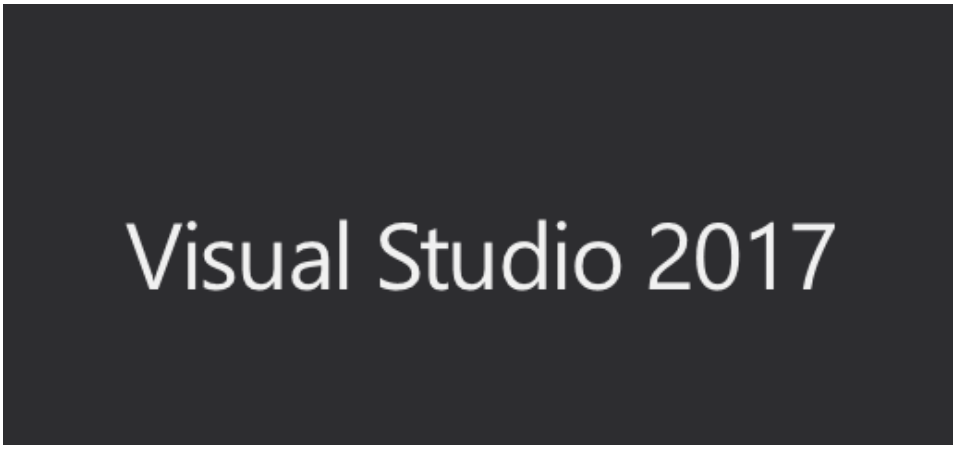
Slika 1: Pregled značajki pojedine verzije Microsoft Visual Studija

| Supported Features | Visual Studio Community | Visual Studio Professional | Visual Studio Enterprise |
|---|---|---|---|
| ⊖ Supported Usage Scenarios | ●●●○ | ●●●● | ●●●● |
| Individual Developers | ● | ● | ● |
| Classroom Learning | ● | ● | ● |
| Academic Research | ● | ● | ● |
| Contributing to Open Source Projects | ● | ● | ● |
| Non-enterprise organizations [1] for up to 5 users | ● | ● | ● |
| Enterprise | | ● | ● |
| Development Platform Support [2] | ●●●● | ●●●● | ●●●● |
| ⊕ Integrated Development Environment | ●●●○ | ●●●○ | ●●●● |
| ⊕ Advanced Debugging and Diagnostics | ●●○○ | ●●○○ | ●●●● |
| ⊕ Testing Tools | ●○○○ | ●○○○ | ●●●● |
| ⊕ Cross-platform Development | ●●○○ | ●●○○ | ●●●● |
| ⊕ Collaboration Tools and Features | ●●●● | ●●●● | ●●●● |

Izvor: https://visualstudio.microsoft.com/vs/compare/

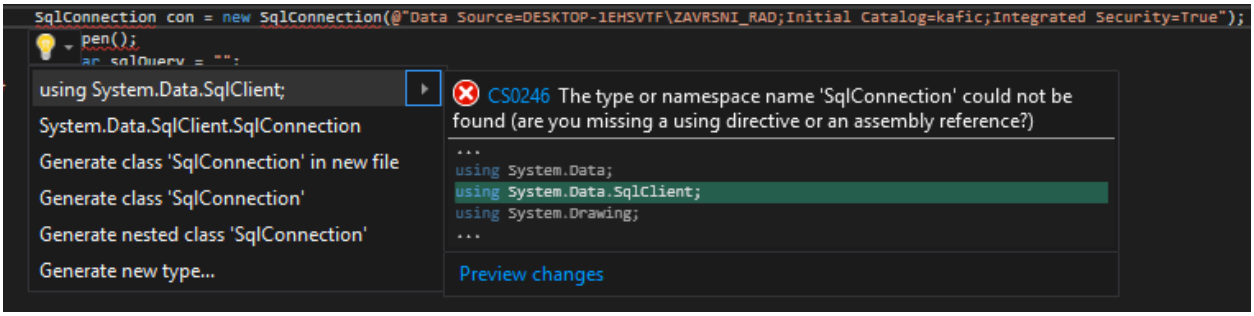
Za mene kao studenta i osobu koja se vidi u svijetu računala i programiranja, nema boljeg alata za izradu windows aplikacija. Program je jednostavan za korištenje, kôd je jednostavan i pregledan. *Debugger* brzo prikazuje i nudi pomoć pri otklanjanju problema. Program pri pisanju sam nudi završetak riječi.

Slika 2: Microsoft Visual Studio startup



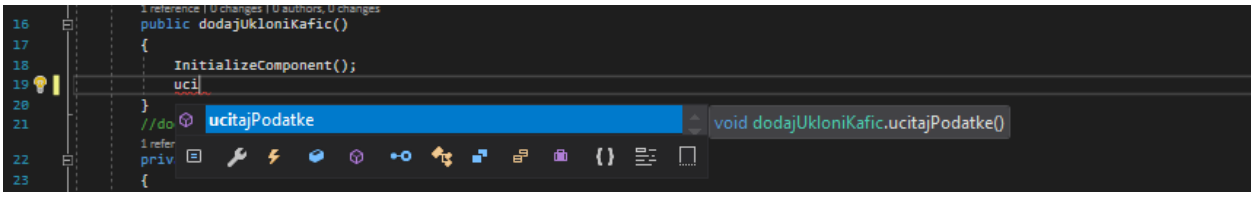

Izvor: izrada autora

Slika 3: Prikaz ponuđenih rješenja za problem




Izvor: izrada autora

Slika 4: Dovršavanje riječi




Izvor: izrada autora

### 2.2. Microsoft SQL Management Studio & Microsoft SQL server

'Microsoft SQL Server je relacijska baza podataka kojoj je primarni jezik za upite 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. Transact SQL nastao je kao plod suradnje između Microsofta i Sybasea. SQL server je baza podataka koja se smjestila na prag između manjih i srednjih baza.'[4]

'SQL Server Management Studio (SSMS) je integrirano okruženje za upravljanje bilo kojom SQL ifrastrukturom. Koristio se za pristup, uređivanje, vođenje, administritanje I razvoj svih komponenti kao što su: SQL Server, Azure SQL Database, i SQL Data Warehouse. SSMS pruža jedan cjelovit program koji kombinira široku skupinu grafičkih alata s brojim editorima za

[4] Wikipedia, https://hr.wikipedia.org/wiki/Microsoft_SQL_Server, 08.07.2018.

uređivanje skripti kako bi  omogućio pristup SQL Serveru programerima i administratorima baza podataka.'[5]

Da bi se razumio rad i način korištenja SSMS potrebno je određeno znanje o T-SQL-u koje smo stekli pohađanjem kolegija Uvoda u baze podataka kao i kolegija Baze podataka. Kada se shvati način i princip rada T-SQL-a, posao postaje lak.

Slika 5: SQL Management Studio Startup

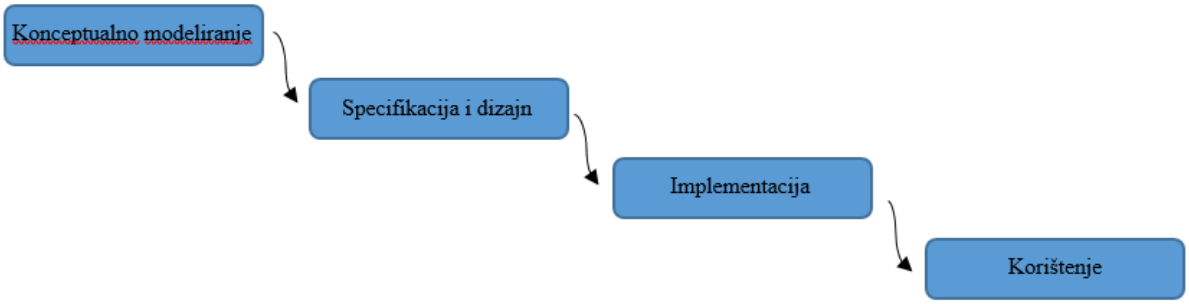

Izvor: izrada autora

[5] Microsoft, https://docs.microsoft.com/en-us/sql/ssms/sql-server-management-studio-ssms?view=sql-server-2017, 08.07.2018.

# 3. FAZE RAZVOJA WINDOWS APLIKACIJE

Prema Elliotu (2004) najstariji formalizirani pristup razvoju je SDLC pristup (eng. Systems Development Life Cycle)[6]

Slika 6 : SDLC dijagram




Izvor: izrada autora

Kao I svaki pravi projekt, I ovaj je prošao svaku fazu. U nastavku rada će biti prikazana i detaljno objašnjena svaka faza.

## 3.1. Konceptualno modeliranje

U fazi konceptualnog modeliranja sudjelovalo je nekoliko osoba. Kao idejni osnivač sam zatražio pomoć u razvoja ideje od mentora mr. sc. Ivan Livaja pred. i Jasmine Sladoljev, univ.spec.oec., pred., kako bi program sadržavao sve važne značajke koje su potrebne za rad u nastavi. Ova faza prikupljanja podataka je trajala oko dva tjedna sa svoj popratnom dokumentacijom. Ova faza je prilično zamorna jer se nevidi konkretan rezultat, a do implementacije slijedi još jedan korak.

[6] G. Elliott, Global business information technology: an integrated systems approach. Harlow, England; New York: Pearson Addison Wesley, 2004.

‘Konceptualno znači neovisno o implementaciji. Konceptualni model podataka daje cjelovit i neredundantan opis podataka informacijskog sustava.

APSTRAKCIJA-to je uočavanje nužnog, glavnog i bitnog a ispuštanje sporednog ili nebitnog.

Vrste apstrakcije

1. klasifikacija
2. generalizacija
3. agregacija

Klasifikacija je vrsta apstrakcije u kojoj se entiteti klasificiraju, opisuju i grupiraju u klase/razrede, tipove prema zajedničkim obilježjima npr. Student (JMBG, Ime, DatumRođenja)

Generalizacija je vrsta apstrakcije u kojoj se entiteti niže razine spajaju entitetom više razine.

Agregacija je formiranje novog pojma višeg stupnja na temelju odnosa postojećih pojmova.

Postoje dvije vrste agregacije:

- agregacija jednostavnih atributa èime se opisuje entitet npr. (šifra, ime, adresa, telefon) osoba
- agregacija entiteta u noviji entitet, npr. rezerviranje sobe za određeni datum predstavlja odnos entiteta osoba, soba datum koji se opisuje agregatnim vezom rezervacija.

  npr. Osoba ( ),Soba ( ),Datum ( ),Rezervacija (šifra osobe, šifra sobe, datum)’[7]

[7] Tečajevi, http://tecajevi.freeservers.com/iskoncept.htm, 08.07.2018.

### 3.2. Specifikacija i dizajn

Sljedeća faza, vrlo bitna u razvoju aplikacije je faza specifikacije I dizajna.

‘Uloga dizajn faze u modelima razvoja:

- Pojašnjenje ulaznih zahtjeva
- Rano otkrivanje pogrešaka
- Planiranje i organizacija programskog rješenja

Osnovna svrha faze dizajna je modeliranje sustava odnosno podjela sustava na module.

Osnovno pitanje koje se postavlja: kako možemo razložiti sustav u manje nezavisne cjeline tako da svaka cjelina ima manju razumnu kompleksnost a da se zajedničkim djelovanjem manjih cjelina ostvaruju funkcionalnosti sustava i učinkovito zadovoljava korisničkim zahtjevima?

Osnovni izlaz faze dizajna je dokument kojim je arhitektura sustava dovoljno razložena da omogući samostalan rad programeru bez interakcije s drugim programerom ili s osobom koja je specificirala zahtjeve

Ova faza je najkreativnija faza razvojnog procesa programskog proizvoda’[8]

### 3.3. Implementacija

‘Instalacija, odnosno implementacija rješenja faze su koje predstavljaju operativnu i tehničku provedbu svega onog što je dizajnom zamišljeno i projektnim planom definirano. Pod pojmom instalacija, podrazumijevamo fizičko postavljanje opreme i softvera, na mjesto i u oblik, kako je projektnim planom predviđeno.

Implementacija je dovođenje opreme i softvera u funkciju kako je predviđeno njenim karakteristikama i projektnim planom, tj. da radi ono za što je predviđena.

U ovoj je fazi važno reći da, kako se općenito povećava stupanj zrelosti IT industrije, relativno je malo tvrtki kod kojih imamo priliku graditi IT sustav ispočetka, tako da pojam „integracija“ dolazi

[8] dr. sc. Tihana Galinac Grbac, UniRi, http://www.riteh.uniri.hr/zav_katd_sluz/zr/nastava/proginz/materijali/Dizajn%20programskog%20proizvoda.pdf, 08.07.2018

sve više do izražaja. Naime, pojam integracija ima više značenja, a ovdje ga koristimo u oba njegova za IT industriju važna značenja:

- Integracija kao postupak spajanja raznorodnih tehnologija kako bi harmonizirano zajedno ispunjavale tražene funkcije i
- Integracija kao spajanje novih sustava ili elemenata s postojećim IT sustavima te integriranje i proširivanje njihovih funkcija.

Sve su komponente u ovoj fazi krucijalno važne za dobivanje traženog rezultata. Stoga je njihova provedba pod posebno strogim nadzorom. Taj je nadzor definiran projektnom metodologijom kojom se pokrivaju sve faze do primopredaje gotovog, funkcionalnog rješenja.

Ukratko, može se reći da je potrebno voditi računa o slijedećim koracima:

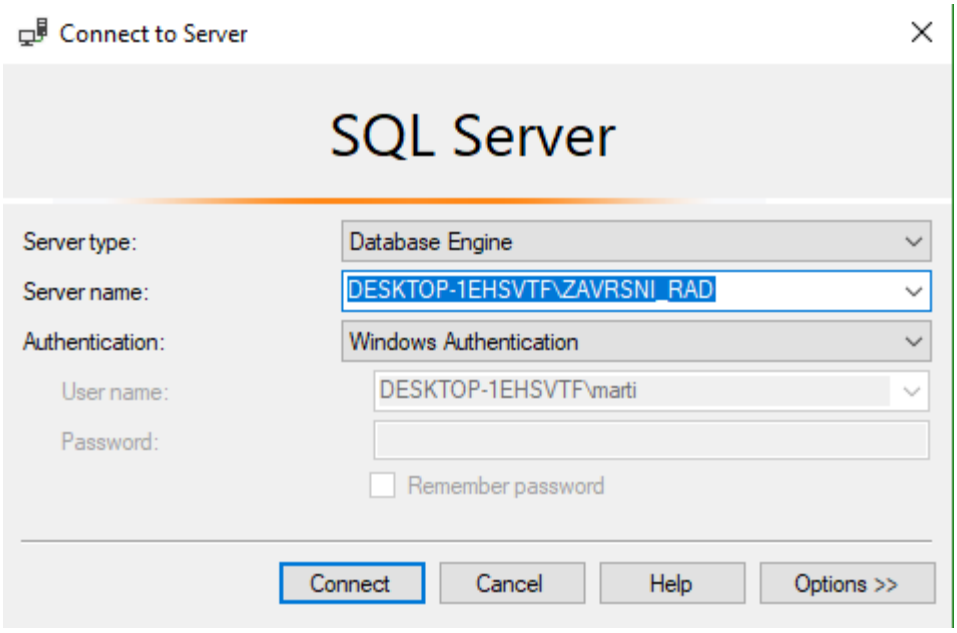
- Plan implementacije – definiraju se terminski i operativni elementi implementacije. Potrebna je koordinacija resursa kako izvođača tako i naručitelja, čiji predstavnici moraju biti članovi projektnog tima
- Testiranje – količina testiranja i sveobuhvatnost njegova dosega daju garanciju da će rješenje od prvog trenutka primopredaje raditi ono što je predviđeno i kako je predviđeno.
- Primopredaja – dokument kojim obje strane, korisnik i izvođač, konstatiraju da je posao obavljen u ugovorenim parametrima, kako količinski tako i funkcionalno.
- Dokumentacija izvedenog stanja – dokumentacija koju korisnik može samostalno koristiti tijekom životnog vijeka rješenja radi kontrole njegovih funkcija, ali isto tako i kao podlogu budućih promjena i nadogradnji.
- Operativni priručnik – dokumentacija koja služi za operativno upravljanje sustavom i njegovim nadzorom.'[9]

No dosta bi bilo suhoparne teorije. Idemo vidjeti kako to izgleda u praksi. Za motivaciju su vam dovoljne male stvari, budite sretni čim dobro odradite I najmanji dio kôda. A sada, vežite se, polijećemo.

[9] EuroComputerSystems, http://www.ecs.hr/it-rjesenja-i-usluge/instalacija-implementacija-integracija/, 08.07.2018.

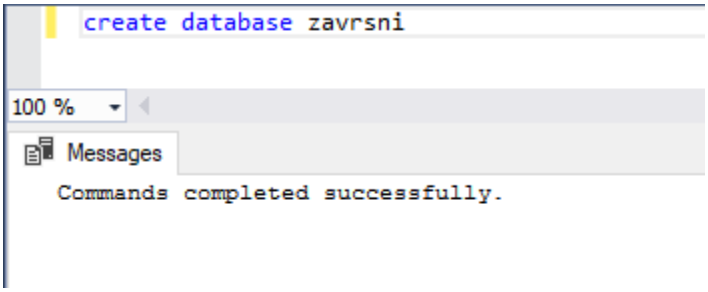
**Korak 1**: Logiranje na lokalni server i kreiranje baza podataka potrebnih za rad programa

Slika 7: Logiranje na bazu podataka



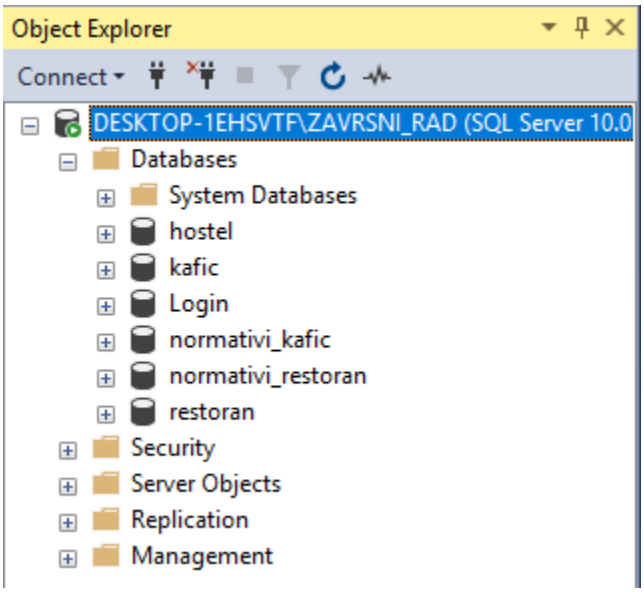

Izvor: izrada autora

Slika 8: Funkcija za stvaranje baze podataka




Izvor: izrada autora

Slika 9: Konačni prikaz baza podataka potrebih za aplikaciju




Izvor: izrada autora

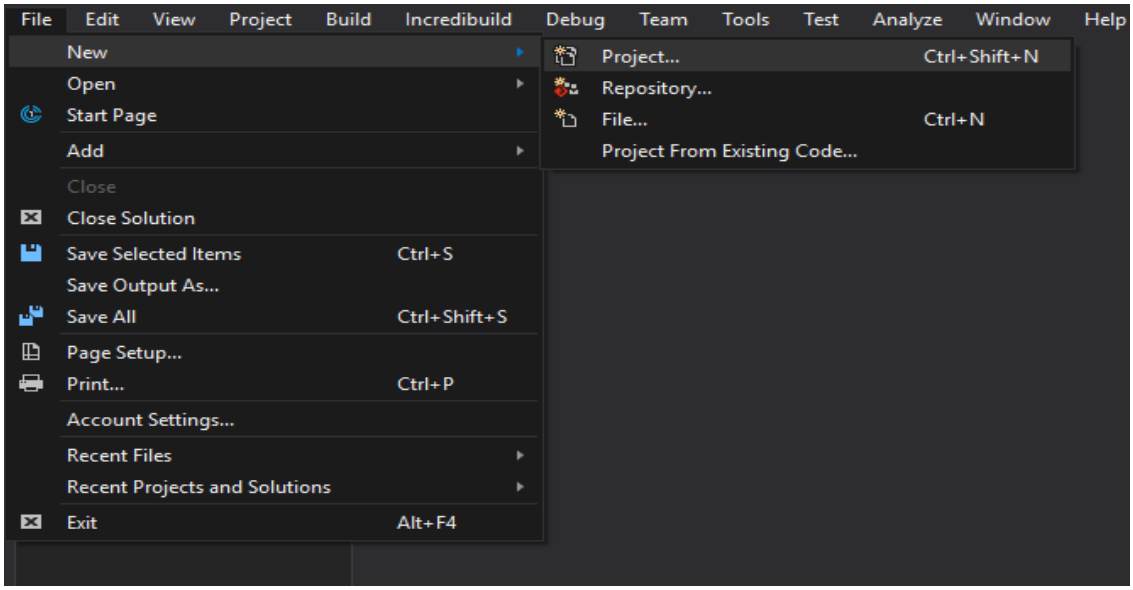
**Korak 2:** stvaranje novog projekta u Microsoft Visual Studiju

'Kada kreirate aplikaciju, website, plug-in, ili sl. u Visual Studiju, prvo što radite je *projekt.* U logičkom smislu, projekt sadrži sve izvorne kôdove, ikone, slike, datoteke s podatcima i slično koje su kompajlirane ( izgrađene ) u .exe, knjižnice ili web stranice.

Projekt je definiran u XML datoteci sa eksenzijom .vbproj, .csproj ili .vcxproj. datoteka sadrži vritualnu hijerarhiju datoteka, puteve do njih u projektu te sadrži '*build settings'*.

U Visual Studiju, projekt je definiran u '*Solution Exploreru'* gdje je prikazan sadržaj projekta i postavke. Kada kompajliramo ( izgradimo ) projekt, MSBuild mašina stvara .exe datoteku koja se može pokrenuti.

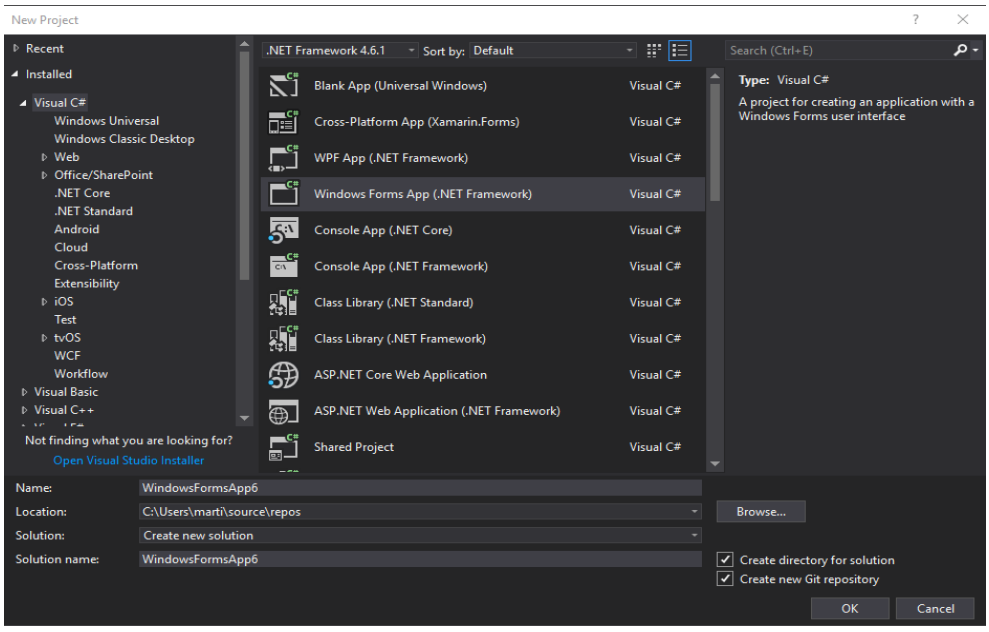
Slika 10:  Prikaz otvaranja novog projekta u Visual Studiju




Izvor: izrada autora

Neovisno o vašim potrebama, i da li je to web ili windows aplikacija, ovo je prvi korak pri stvaranju projekta. Nakon toga slijedi sljedeći korak u kojem odabirete što je Vama potrebno.

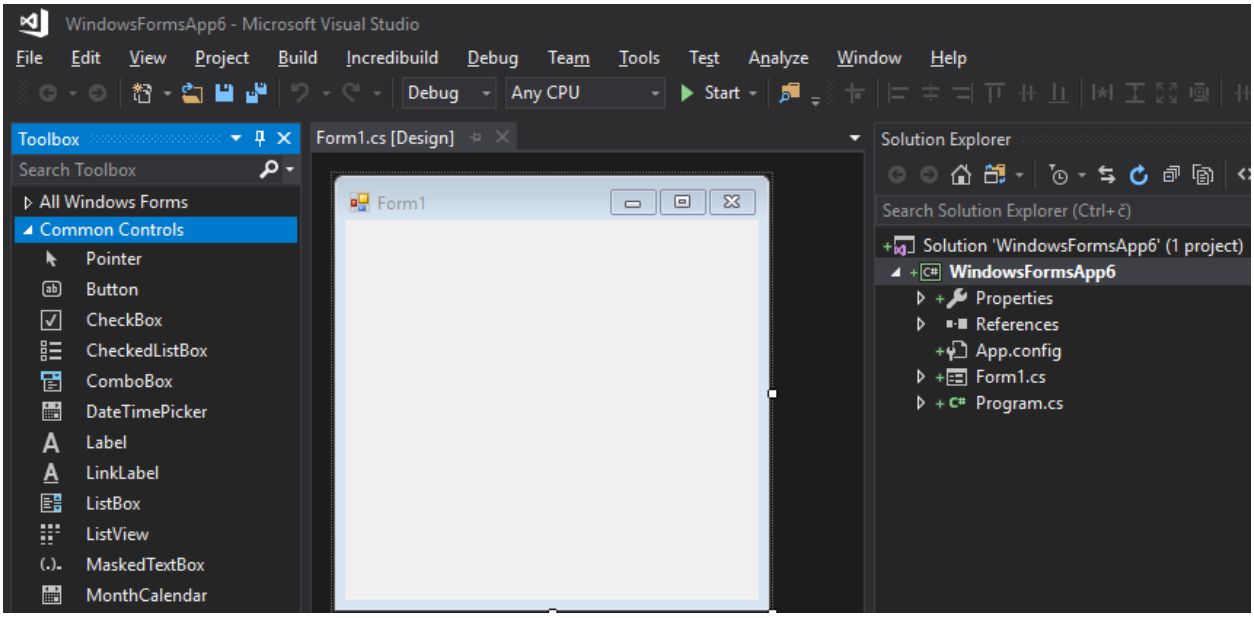
Slika 11: Odabir Windows forme




Izvor: izrada autora

Nakon odabira imena, lokacije gdje želite spremiti Vaše rješenje i pod kojim imenom, krećete sa ‘ozbiljnijim’ radom.

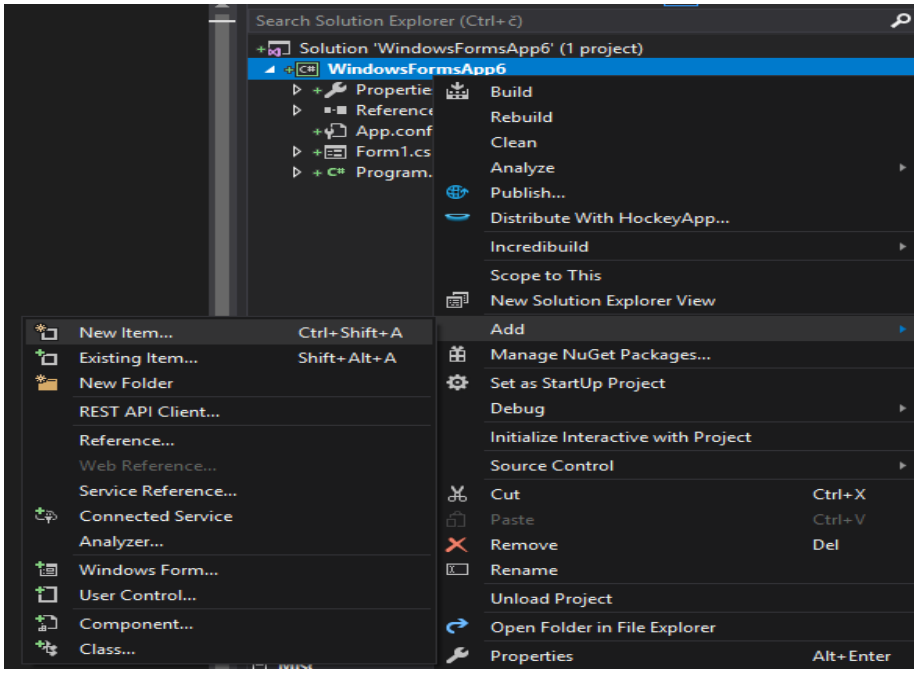
Slika 12: Prazan projekat




Izvor: izrada autora

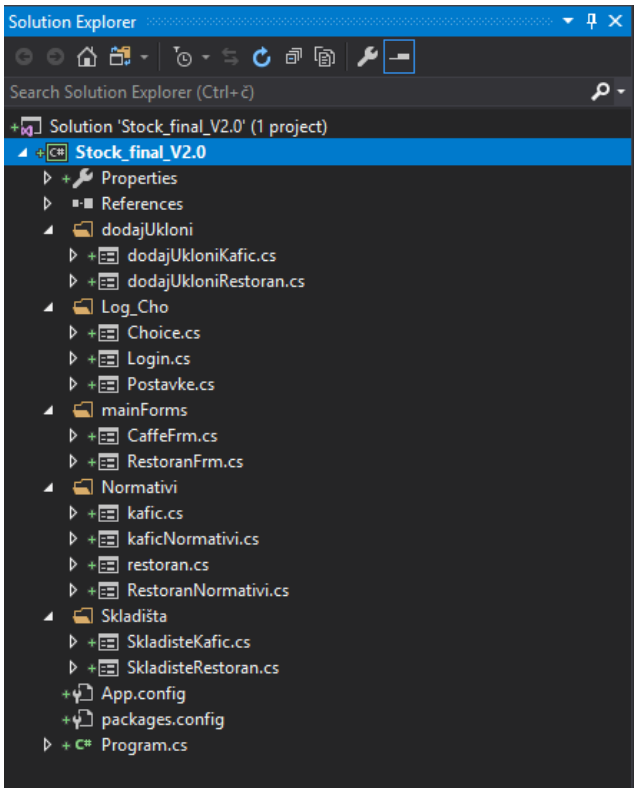
Na slici 12 je prikaz početnog stanja nakon kreiranja dokumenta. Sa lijeve strane su prikazane funkcije i kontrole koje možemo ubaciti u pojedinu formu. U središnjem dijelu je grafički editor koji u stvarnom vremenu prikazuje kako će program izgledati ukoliko se pokrene bez grešaka. S desne strane je prikazan gore navedeni ‘*Solution Explorer’* u kojemu se dodaju nove forme.

Slika 13: Dodavanje druge nove forme u projekt




Izvor: izrada autora

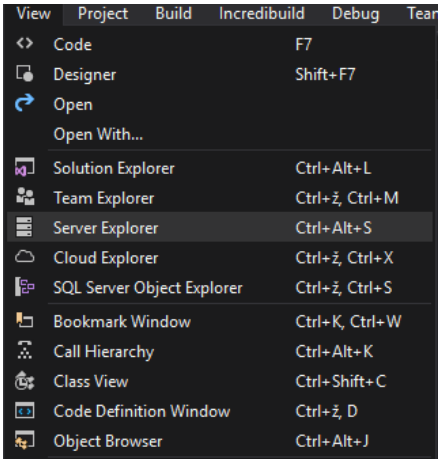
Slika 14: Konačan izgled '*Solution explorer-a'* nakon završenog projekta




Izvor: izrada autora

Nakon kreiranih formi, vrijeme je za dodavanje lokalnih baza podataka u projekt. Prvi korak je odabrati prozor pod nazivom '*server explorer'*.

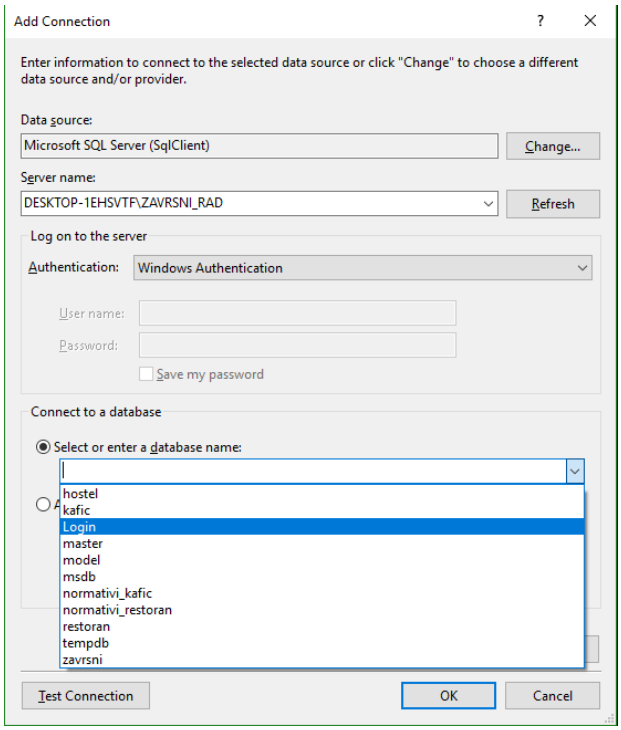
Slika 15: Odabir prozora '*server explorer'*




Izvor: izrada autora

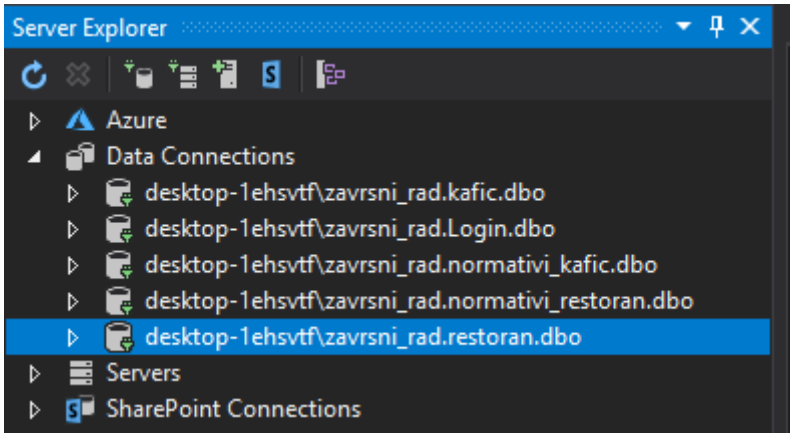
Nakon toga nam sa lijeve strane izbacuje prozor na kojem odabiremo gumb '*connect to database'*. U sljedećem koraku odabiremo naš '*server name'*.

Slika 16: Spajanje sa lokalnim serverom




Izvor: izrada autora

Slika 17: Prikaz svih potrebnih konekcija

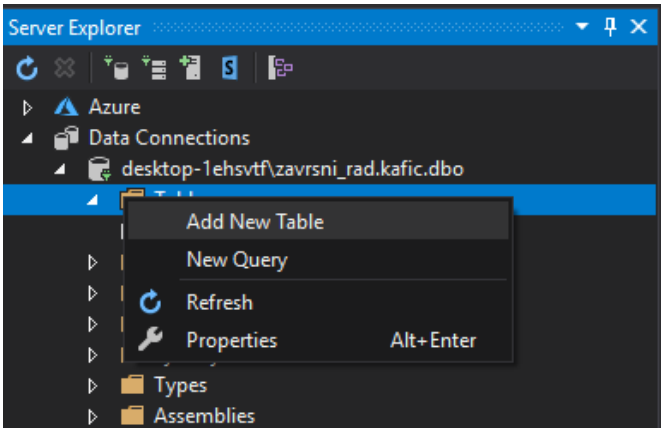



Izvor: izrada autora

Nakon što smo spojili sve baze koje su nam potrebne, nekoliko stvari ima  koje su bitne. Bitno je dobro definirati tablice i tipove podataka u njima. Ukoliko taj dio nismo dobro napravili u fazi specifikacije i dizajna, postoji velika šansa da će nam kasnije izbacivati logičke greške iako Vam je kôd u potpunosti sintaktički točan. Idemo pogledati kako se definiraju tablice i polja u tablicama. ( iako se ovaj korak može odraditi u SMSS, preporučam preko Visual Studija, lakše je i ne treba Vam znanje T-SQL-a.)

Da bi dodali novu tablicu u pojedinu bazu, potreban nam je slijedeći korak.

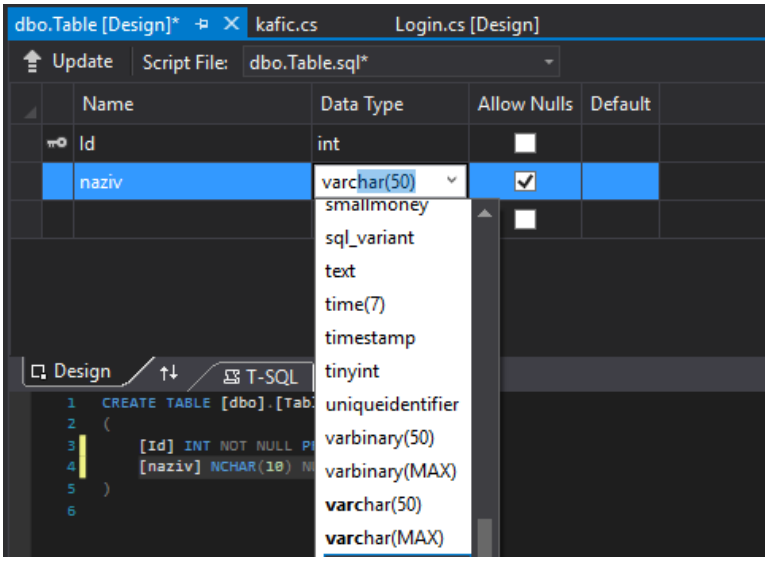
Slika 18: Dodavanje nove tablice u bazu podataka




Izvor: izrada autora

Nakon što smo dodali tablicu, potrebno je i imenovati pojedine stupce te definirati tipove podataka u njima. To ćemo učiniti na slijedeći način. Prvo unesemo naziv stupca, a nakon toga odabiremo '*Data type'*. Dok smo završili sa radom, dovoljno je pritisnuti gumb '*Update'*.

Slika 19: Dodavanje naziva stupcima




Izvor: izrada autora

Nakon kreiranih svih tablica, odrađen je 'pozadinski dio' posla te sad idemo na dio koji se 'vidi'. Kako su knjižnice po '*Default-u'* uključene, nećemo ih posebno opisivati.

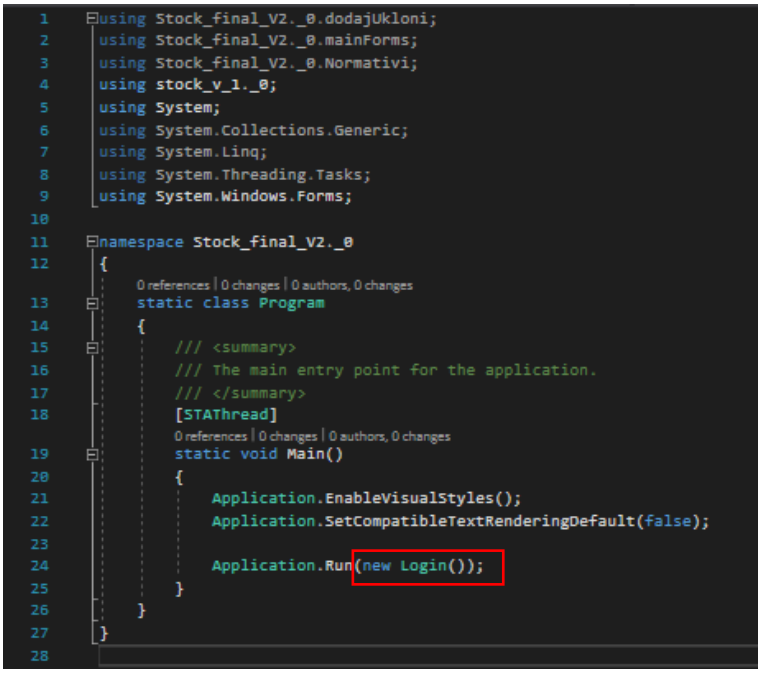
Slika 20: Knjižnice

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

Izvor: izrada autora

I zadnja forma koju će mo objasniti prije neko što se bacimo na kôd je forma '*Program.cs'* koju sadrži svaki projekt. Nećemo ulaziti u detalje samo će mo objasniti najbitniju značajku ove forme. U njoj postavljamo formu koju će '*Debugger'* prvu otvarati nakon kompajliranja programa.

Slika 21: Prikaz '*Program.cs'* forme



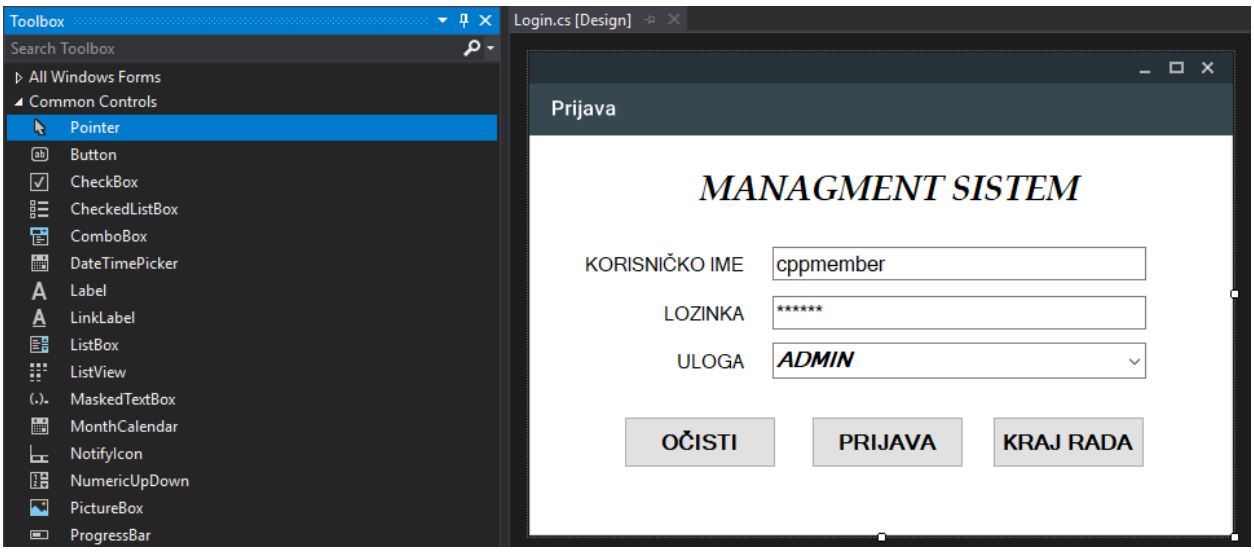
```
using Stock_final_V2._0.dodajUkloni;
using Stock_final_V2._0.mainForms;
using Stock_final_V2._0.Normativi;
using stock_v_1._0;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Stock_final_V2._0
{
    0 references | 0 changes | 0 authors, 0 changes
    static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        0 references | 0 changes | 0 authors, 0 changes
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);

            Application.Run(new Login());
        }
    }
}
```

Izvor: izrada autora

Nakon riječi 'NEW' dodajemo ime forme koju želimo da se prva pokreće. Praktično je kada imate nekoliko naslijeđenih formi, među kojima je upisivanje podataka poput korisničkog imena I lozinke I da nebi taj korak morali stalnno ponavljati prilikom testiranja programa, možemo ga ovim putem jednostavno zaobići.

Krećemo sa uređenjem formi. Prva nam je na redu LOGIN forma, pa počnimo.

Slika 22: Kreiranje LOGIN forme



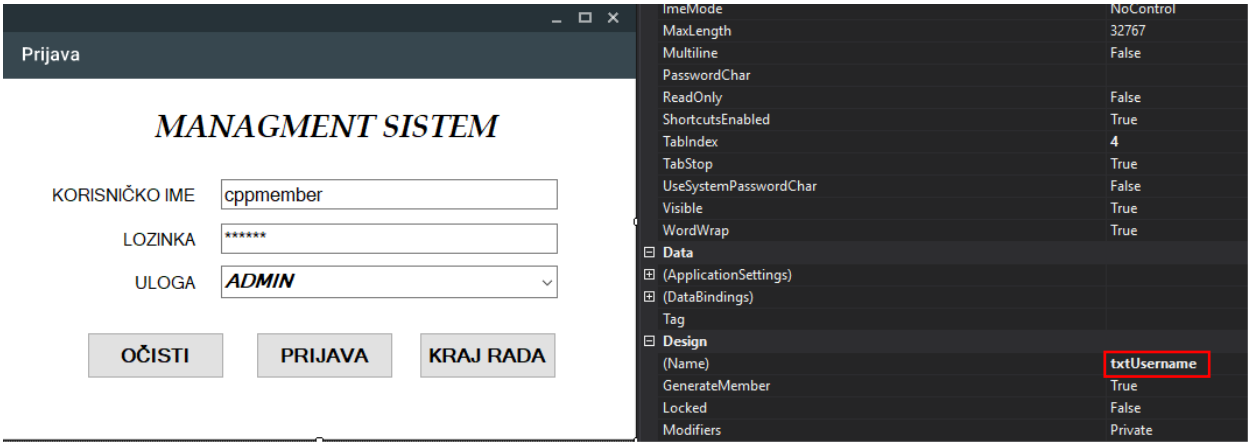

Izvor: izrada autora

Nakon dodavanja određenih kontrola, kao što su '*TextBox'*, '*label'* , '*button'* i '*ComboBox'* dobije se rezultat kao na slici 22. To je moja vizija LOGIN forme, te ju svatko može slagati po svojoj želji.

Logika je da korisnik unosi svoje podatke, dok se korisnik može ulogom prijaviti kao admin ili gost. Ukoliko se prijavi kao gost, zabranjena mu je forma postavke gdje se dodaju i brišu korisnički računi i postavljaju lozinke.

Još jedna napomena koja će Vam olakšati život prilikom povezivanja grafičkog sučelja, kontrola i kôda. '*TextBox-ovima'* dajte logična imena. Npr. Ukoliko imamo '*TextBox'* uz korisničko ime, dat ćemo mu naziv '*txtIme'* ili ako sve pišemo na engleskom biti će '*txtName'*. Praksa je još da se I prvo slovo piše malim slovom, a pošto riječi moraju biti spojene, svaka sljedeća riječ je velikim slovom kao što vidimo na ovom primjeru.

Slika 23: Prikaz davanja imena kontrolama




Izvor: izrada autora

Sada prelazimo na kôd login forme. Nakon prikazanog kôda biti će komentiran i objašnjen svaki dio kôda.

Slika 24: Kôd login forme

```
        }
        // pokretanjem programa se briše svaki podatak iz textBox-ova
        1 reference | 0 changes | 0 authors, 0 changes
        private void Login_Load(object sender, EventArgs e)
        {
            obrišiPodatke();
        }
        //Gumbom obriši čistimo podatke iz textBox-ova
        1 reference | 0 changes | 0 authors, 0 changes
        private void btnClear_Click(object sender, EventArgs e)
        {
            obrišiPodatke();
        }
        // Izlazimo iz aplikacije i gasimo je
        1 reference | 0 changes | 0 authors, 0 changes
        private void btnExit_Click(object sender, EventArgs e)
        {
            Application.Exit();

        }
        //nakon klika na gumb "PRIJAVA" vrši se provjera unesenih podataka sa bazom podataka "Login" u kojoj su pohranjeni podatci te uloga koju korisnički račun ima
        1 reference | 0 changes | 0 authors, 0 changes
        private void btnLogin_Click(object sender, EventArgs e)
        {
            SqlConnection con = new SqlConnection(@"Data Source=DESKTOP-1EHSVTF\ZAVRSNI_RAD;Initial Catalog=Login;Integrated Security=True");
            SqlDataAdapter sda = new SqlDataAdapter(@"SELECT * FROM[dbo].[Login] where Username = '"+txtUsername.Text+"' and Password = '"+txtPassword.Text+"' and Role='"+txtRole.Text+"'",con);
            DataTable dt = new DataTable();
            sda.Fill(dt);
            if (dt.Rows.Count==1)
            {
                Choice mF = new Choice(txtRole.Text);
                mF.Show();
                this.Hide();
            }
            else
            {
                MessageBox.Show("Pogrešan Username ili Password!","ERROR",MessageBoxButtons.OK,MessageBoxIcon.Error);
                obrišiPodatke();
            }

        }

        //funkcija za brisanje podataka iz textBox-ova
        3 references | 0 changes | 0 authors, 0 changes
        private void obrišiPodatke()
        {
            txtPassword.Clear();
            txtUsername.Clear();
            txtRole.Text = "";
            txtUsername.Focus();
        }
```

Izvor: izrada autora

Kôd se izvodi od početka prema kraju, tako da pripazite gdje stavljate koji dio kôda, jer Npr. Ukoliko definirate neku varijablu iza njezine upotrebe, '*Debugger'* će Vam izbaciti pogrešku iako ste vi teoretski dobro napisali ( osim u slučaju pozivanja funkcija gdje se preskaču dijelovi kôda).

Primjer funkcije kada se preskače dio kôda imamo u dijelu kada program poziva funkciju '*obrisiPodatke();'* koji brišu podatke iz '*TextBox-ova'*. Nakon izvedene funkcije program se vraća nazad za dio kôda gdje je funkcija pozvana.

'*btnLogin_Click'* prikazuje što se događa kada pritisnemo gumb 'Prijava'. U ovome dijelu kako I logika nalaže, povezujemo se sa bazom gdje provjeravamo podatke koji su uneseni, odnosno postoji li račun sa zadanim imenom, lozinkom i ulogom. Ponoviti ćemo dio kôda sa povezivanjem sa bazom da Vam pojednostavnimo prikaz.

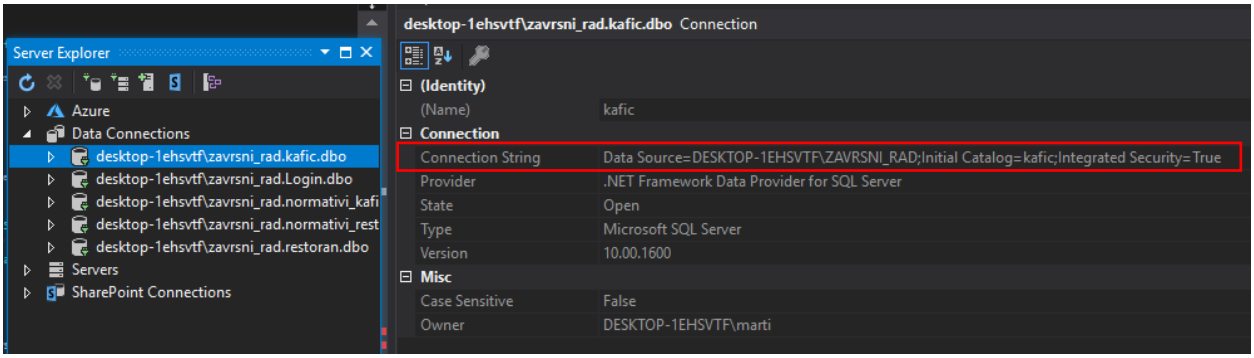
Slika 25: Ponovljeni prikaz dijela koda

```
SqlConnection con = new SqlConnection(@"Data Source=DESKTOP-1EHSVTF\ZAVRSNI_RAD;Initial Catalog=Login;Integrated Security=True");
SqlDataAdapter sda = new SqlDataAdapter(@"SELECT * FROM[dbo].[Login] where Username = '"+txtUsername.Text+"' and Password = '"+txtPassword.Text+"' and Role='"+txtRole.Text+"'",con);
DataTable dt = new DataTable();
sda.Fill(dt);
if (dt.Rows.Count==1)
{
   Choice mf = new Choice(txtRole.Text);
   mf.Show();
    this.Hide();
}
else
{
    MessageBox.Show("Pogrešan Username ili Password!","ERROR",MessageBoxButtons.OK,MessageBoxIcon.Error);
    obrišiPodatke();
}
```

Izvor: izrada autora

Pozivanje baze se odrađuje na gore prikazani način gdje otvaramo novi '*SqlConnection'*. Budite oprezni, da bi vam funkcija radila, morate uključiti 2 dodatne knjižnice. Dio u zagradi predstavlja '*Connection string'* kojeg ima svaka baza podataka koja je uključena u program te služi kao 'put' koji govori gdje se nalazi baza s kojom program mora komunicirati.

Slika 26: '*Connection string*'




Izvor: izrada autora

Slika 27: Dodavanje knjižnica potrebnih za povezivanje s bazom podataka

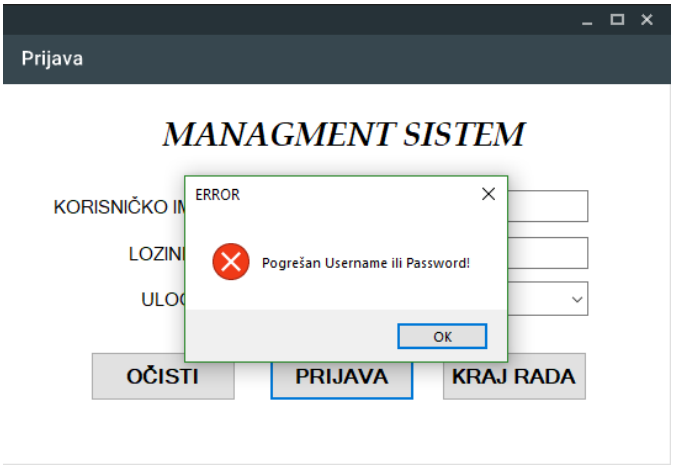
```
5   using System.Data;
6   using System.Data.SqlClient;
```

Izvor: izrada autora

'*SqlDataAdapter*' je naredba u koju upisujemo što želimo da naš program 'radi' sa bazom podataka. Kao što vidimo, ovdje želimo samo da program provjeri postoji li zapis koji ima '*Username*', '*Password*', i '*Role*' kao što smo mi unijeli.

Ukoliko zapis postoji, otvoriti će nam se forma '*Choice*'. Dijelom kôda koji kaže 'New Choice( txtRole.Text);' prosljeđujemo u sljedeću formu ulogu korisnika. Ukoliko ne postoji ili ste unijeli nepotpune podatke, program će izbaciti grešku da takav zapis ne postoji u bazi.

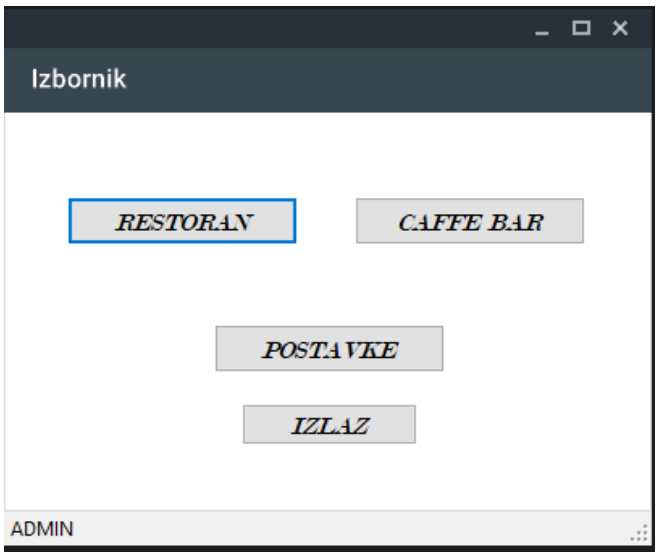
Slika 28: Primjer pogrešnog unosa podataka




Izvor: izrada autora

U nastavku je prikazano još nekoliko grafičkih rješenja formi.

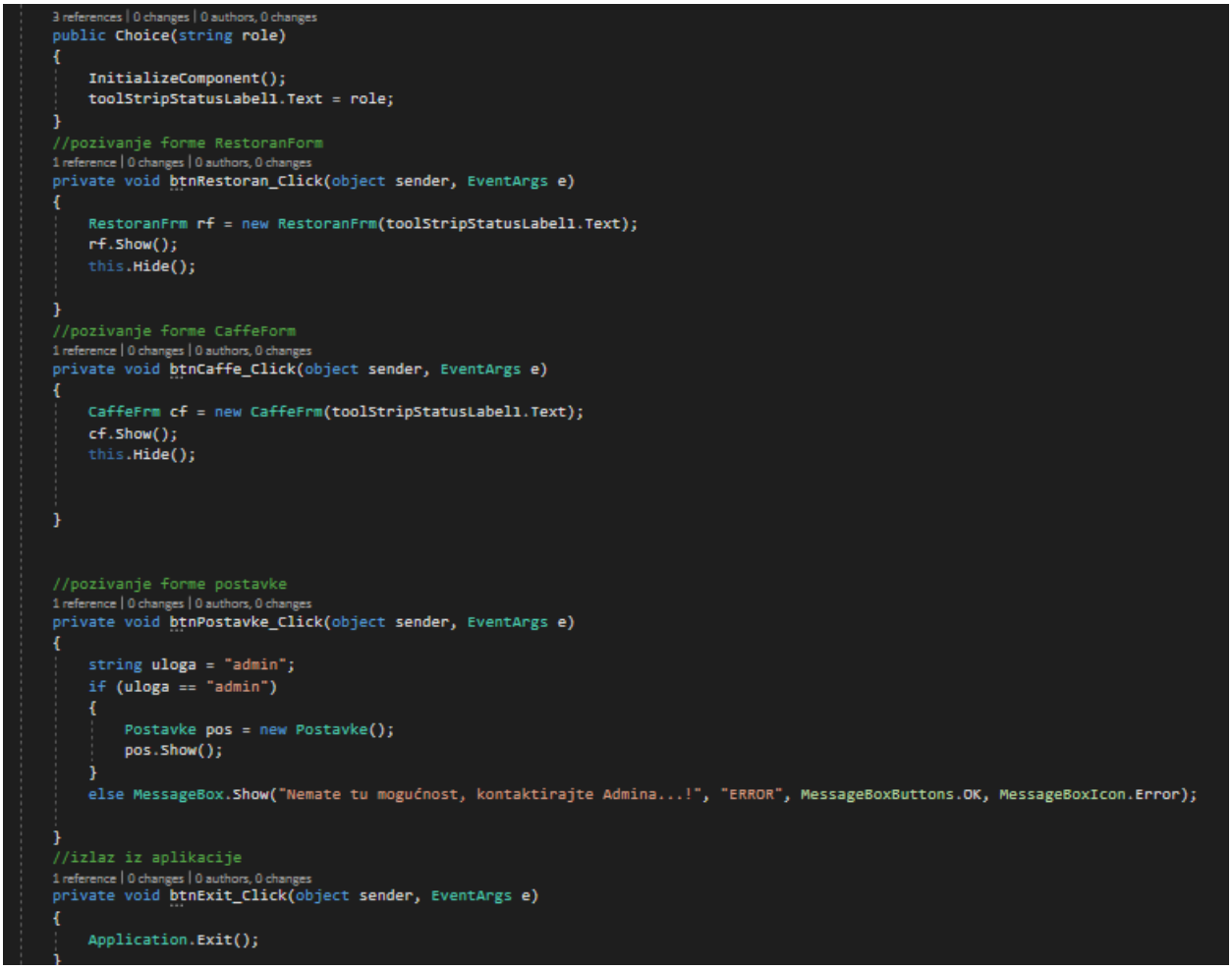
Slika 29: Prikaz forme odabira područja




Izvor: izrada autora

U ovoj formi nema ništa previše kompliciranо te je usputna forma koja usmjerava naš daljnji odabir. Jedina logika koja nam je ovdje bitna da korisnik koji nema administratorske ovlasti nemože otvoriti ‘Postavke’ .

Slika 30: Kôd forme ‘*Choice’*

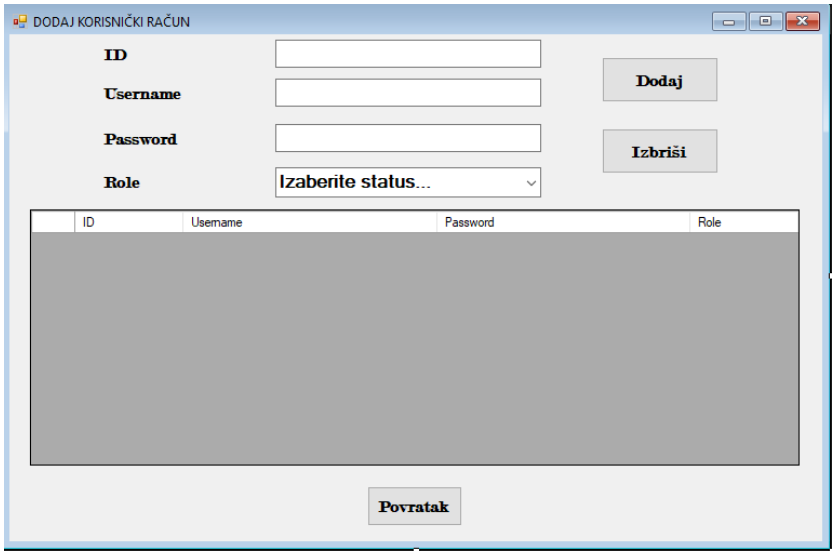


```
3 references | 0 changes | 0 authors, 0 changes
public Choice(string role)
{
    InitializeComponent();
    toolStripStatusLabel1.Text = role;
}
//pozivanje forme RestoranForm
1 reference | 0 changes | 0 authors, 0 changes
private void btnRestoran_Click(object sender, EventArgs e)
{
    RestoranFrm rf = new RestoranFrm(toolStripStatusLabel1.Text);
    rf.Show();
    this.Hide();

}
//pozivanje forme CaffeForm
1 reference | 0 changes | 0 authors, 0 changes
private void btnCaffe_Click(object sender, EventArgs e)
{
    CaffeFrm cf = new CaffeFrm(toolStripStatusLabel1.Text);
    cf.Show();
    this.Hide();


}


//pozivanje forme postavke
1 reference | 0 changes | 0 authors, 0 changes
private void btnPostavke_Click(object sender, EventArgs e)
{
    string uloga = "admin";
    if (uloga == "admin")
    {
        Postavke pos = new Postavke();
        pos.Show();
    }
    else MessageBox.Show("Nemate tu mogućnost, kontaktirajte Admina...!", "ERROR", MessageBoxButtons.OK, MessageBoxIcon.Error);

}
//izlaz iz aplikacije
1 reference | 0 changes | 0 authors, 0 changes
private void btnExit_Click(object sender, EventArgs e)
{
    Application.Exit();
}
```

Izvor: izrada autora

Novi dio kôda nam je dio gdje provjeravamo ima li korisnik administratorske ovlasti da uđe u postavke i dodaje korisničke račune. Kako smo stavili javno da se vidi je li trenutno prijavljen admin ili gost, dovoljno je da ‘posudimo’ string i provjerimo tko koristi aplikaciju.

Slika 31: Forma ‘Postavke’




Izvor: izrada autora

Slika 32: Kôd forme ‘Postavke’

```
private void btnDodaj_Click(object sender, EventArgs e)
{
    if (txtID.Text == "" || txtpassword.Text == "" || txtRole.Text == "" || txtUsername.Text == "" )
    {

        MessageBox.Show("NIste popunili sva polja!..!", "ERROR!", MessageBoxButtons.OK, MessageBoxIcon.Error);

    }
    else
    {
        SqlConnection con = new SqlConnection(@"Data Source=DESKTOP-1EHSVTF\ZAVRSNI_RAD;Initial Catalog=Login;Integrated Security=True");
        con.Open();
        var sqlQuery = "";
        if (PostojiLiZapis(con, txtID.Text)) // provjera postojanja zapisa
        {
            //ukoliko zapis postoji, ažurira se
            sqlQuery = @"UPDATE [dbo].[login] set [username] = '" + txtUsername.Text + "',[password] = '" + txtpassword.Text + "',[role] ='" + txtRole.Text + "'   WHERE [id] = '" + txtID.Text + "'";
        }
        else
        {
            //ukoliko zapis ne postoji, dodaje se
            sqlQuery = @"INSERT INTO [dbo].[login] ([id]  ,[username] ,[password],[role]) VALUES ('" + txtID.Text + "','" + txtUsername.Text + "','" + txtpassword.Text + "','" + txtRole.Text + "')";
            brisi();
        }
        SqlCommand com = new SqlCommand(sqlQuery, con);
        com.ExecuteNonQuery();
        con.Close();
        ucitavanjePodataka();
    }
}
```

Izvor: izrada autora

U ovome dijelu objasniti ćemo što se dogodi kada korisnik upiše podatke te pritisne gumb ‘Dodaj’. Ukoliko korisnik ne unese podatke, pojaviti će se greška da nisu uneseni svi podatci. Ukoliko su svi podatci uneseni, otvoriti će se konekcija s bazom, te će funkcija ‘*postojiLiZapis();’* provjeriti postoji li zapis sa tim podatcima u bazi.

Slika 33: Funkcija '*postojiLiZapis();*'

```
//funkcija za provjeru računa
2 references | 0 changes | 0 authors, 0 changes
private bool PostojiLiZapis(SqlConnection con, string idproizvoda)
{
    SqlDataAdapter sda = new SqlDataAdapter("select 1 from [dbo].[login] where [id]='" + idproizvoda + "' ", con);
    DataTable dt = new DataTable();
    sda.Fill(dt);
    if (dt.Rows.Count > 0) return true;
    else return false;
}
```

Izvor: izrada autora

Ukoliko račun postoji, funkcija vraća '*true'* te program ažurira navedene podatke u bazi, ukoliko podatci ne postoje, program ih nanovo unosi. Potrebno je dijelove unutar zagrada ( u kôdu crvene boje ) ispravno napisati jer, kao što sam već jednom spomenuo, '*Debugger'* neće izbacivati sintaktičku grešku, već logičku koju je kasnije teže pronaći i otkloniti.

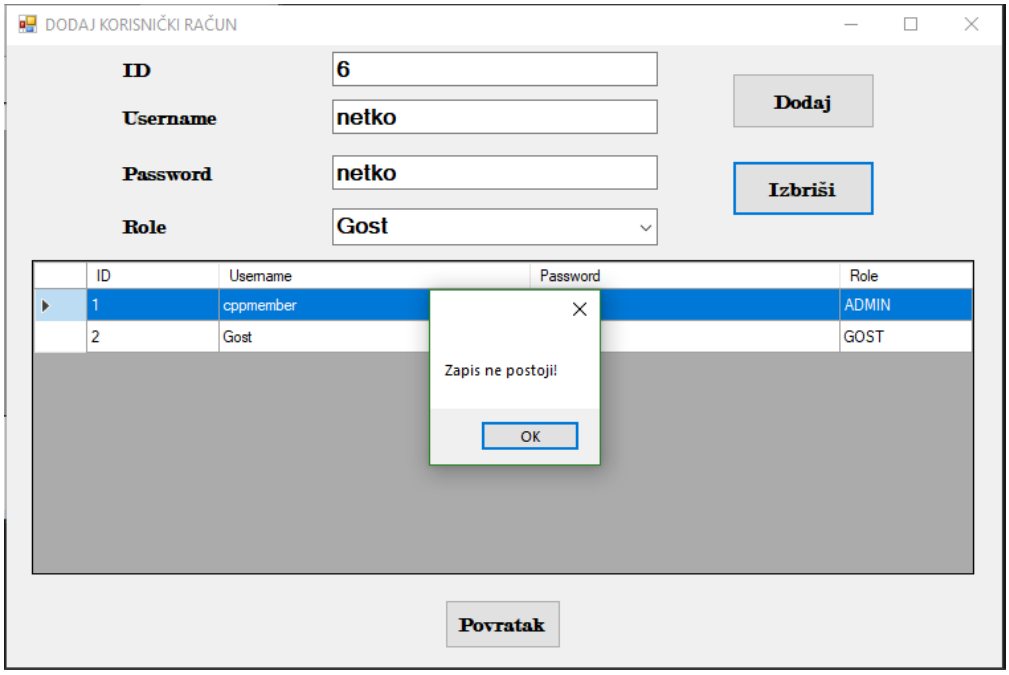
Slika 34: Brisnje korisničkih računa

```
//brisanje računa
1 reference | 0 changes | 0 authors, 0 changes
private void btnIzbrisi_Click(object sender, EventArgs e)
{
    SqlConnection con = new SqlConnection(@"Data Source=DESKTOP-1EHSVTF\ZAVRSNI_RAD;Initial Catalog=Login;Integrated Security=True");
    con.Open();
    var sqlQuery = "";
    SqlCommand com = new SqlCommand(sqlQuery, con);
    if (PostojiLiZapis(con, txtID.Text))
    {
        sqlQuery = @"DELETE FROM [dbo].[login] WHERE [id] = '" + txtID.Text + "'";
        SqlCommand cmd = new SqlCommand(sqlQuery, con);
        cmd.ExecuteNonQuery();
        con.Close();
        brisi();
    }
    else
    {
        MessageBox.Show("Zapis ne postoji!");
    }
    ucitavanjePodataka();
}
```

Izvor: izrada autora

Ista je logika kao i kod dodavanja računa. Ukoliko zapis postoji briše se, ukoliko ne postoji, program prikazuje poruku.

Slika 35: Prikaz greške prilikom brisanja računa




Izvor: izrada autora

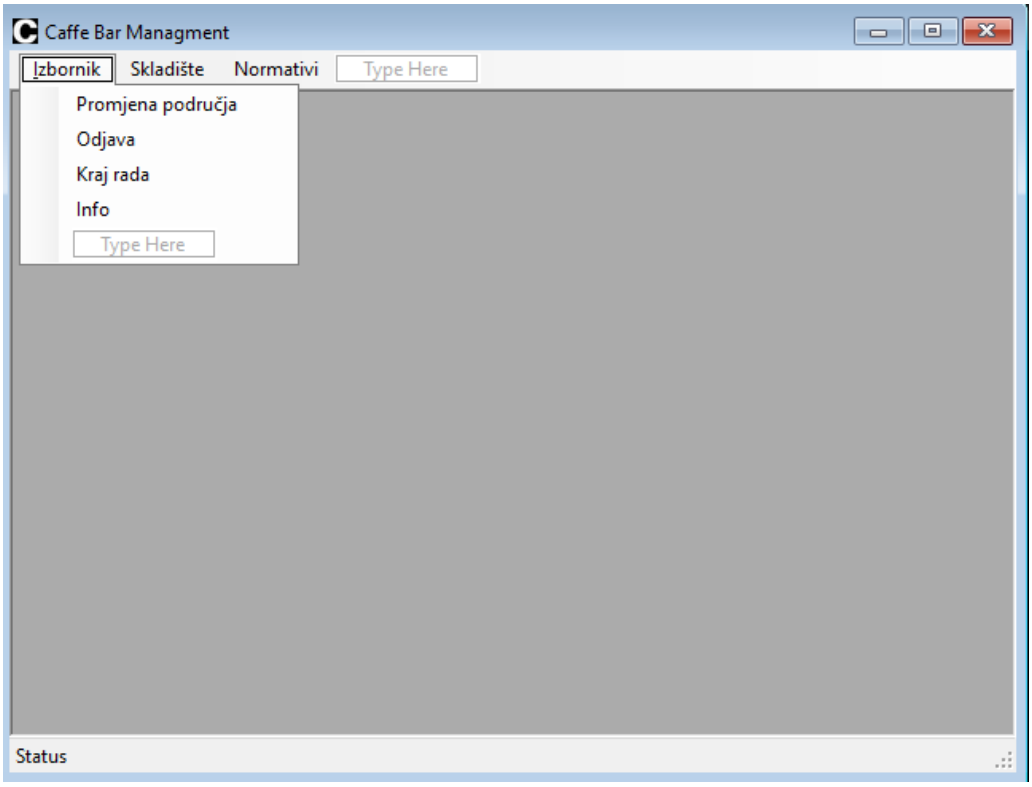
U ovom kôdu postoji još jedan zanimljivi dio koji ubacuje selektirane podatke iz '*DataGridWiew-a'* u već postojeće '*TextBox-ove'*.

Slika 36: Ubacivanje podataka u '*TextBox-ove'*

```
//učitavanje podataka u DataGridWiew
1 reference | 0 changes | 0 authors, 0 changes
private void dataGrid_MouseDoubleClick(object sender, MouseEventArgs e)
{
    txtRole.Text = "";
    txtID.Text = dataGrid.SelectedRows[0].Cells[0].Value.ToString();
    txtUsername.Text = dataGrid.SelectedRows[0].Cells[1].Value.ToString();
    txtpassword.Text = dataGrid.SelectedRows[0].Cells[2].Value.ToString();
    txtRole.SelectedText = dataGrid.SelectedRows[0].Cells[3].Value.ToString();
}
```

Izvor: izrada autora

Slika 37: Glavni izbornik



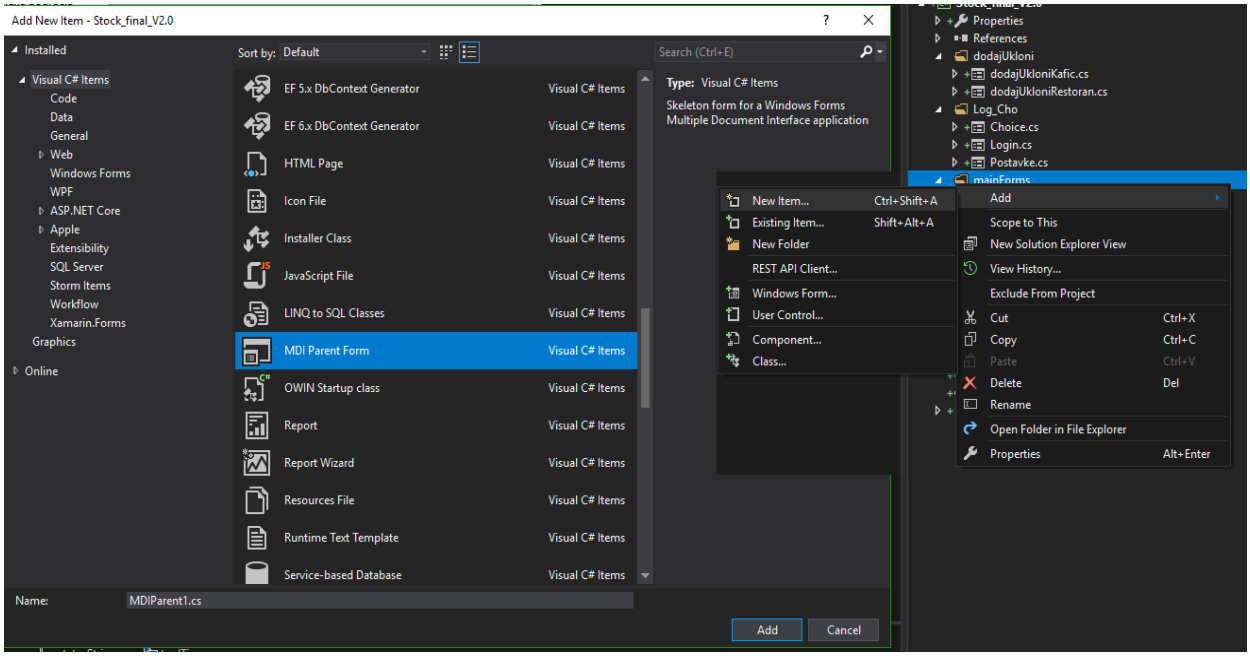

Izvor: izrada autora

U glavnoj formi imamo nekoliko opcija. U padajućem izborniku nam se nudi mogućnosti izbora što želimo da program radi. U kartici ćemo dodavati artikle u skladište, a u kartici normativi će mo dodavati proizvode i normative.

Sada ćemo prikazati kako se dodaje MDI parent forma.

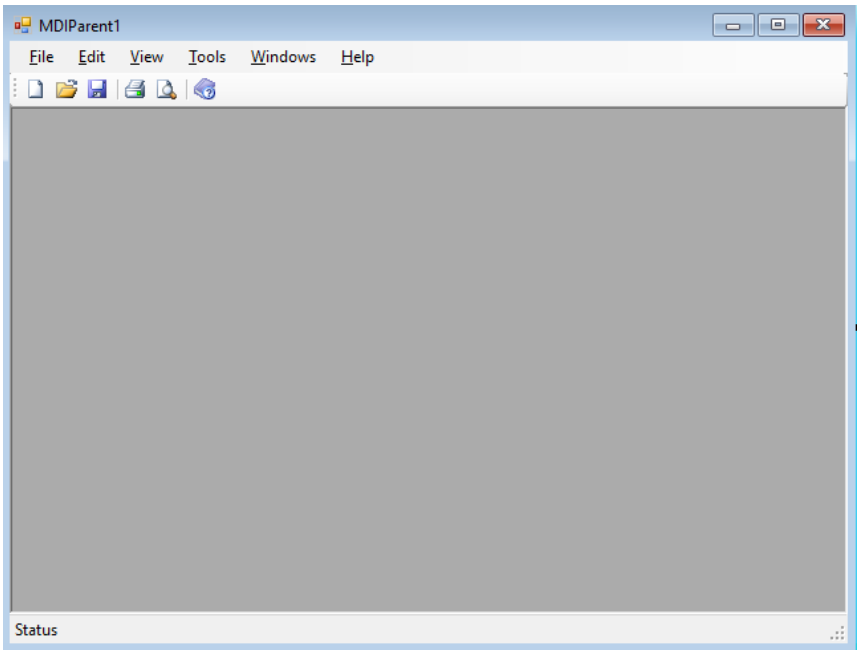
MDI parent forma gotovi je predložak nam daje ogromnu količinu mogućnosti koje Vam trenutno i neće trebati, ali ukloniti će mo višak. Služi kao bazni primjer forme u kojoj se nalazi sav potreban kôd za otvaranje, spremanje, brisanje datoteka i još mnogo drugih funkcija koje možete sami istražiti. Forma je vrlo pristupačna jer  nam je nadređena u svakom pogledu naspram drugih formi, te će uvijek biti u pozadini i pružati izgled profesionalnog programa.

Slika 38: Dodavanje MDI parent forme




Izvor: izrada autora

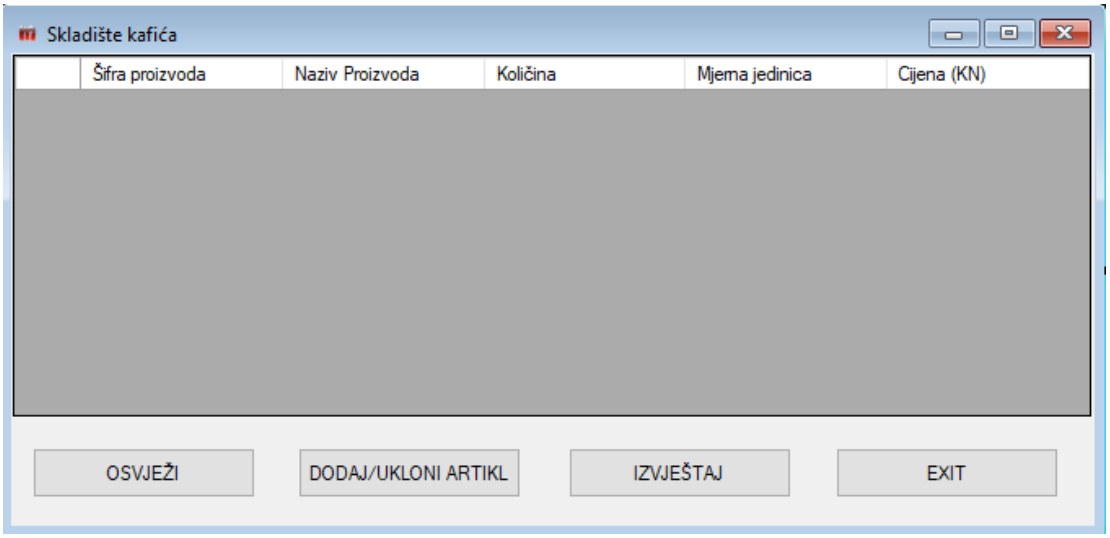
Slika 39: Nova MDI parent forma




Izvor: izrada autora

Ono što nam nije potrebno, označite, i samo pritisnite tipku '*Delete*'. Nakon što smo maknuli sve nepotrebno, dobiti će mo izgled slike 37.

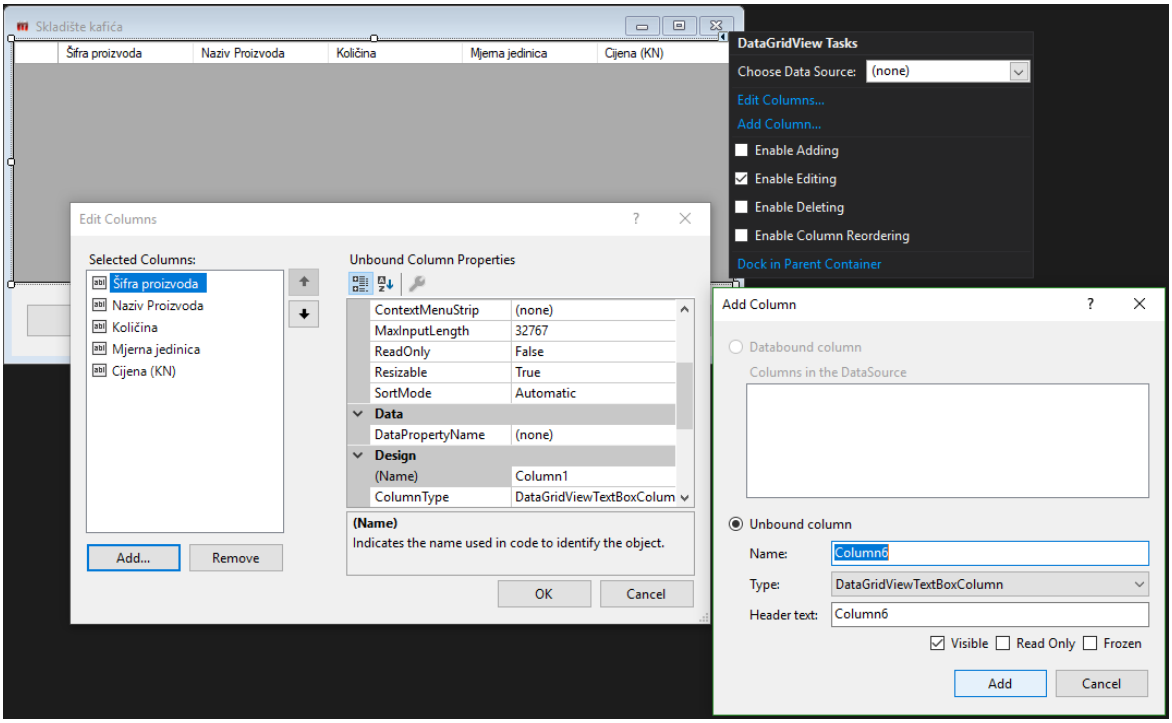
Slika 40: Izgled skladišta




Izvor: izrada autora

Sivi dio u formi predstavlja '*DataGridWiew'*, kontrola koja nam omogućuje prikaz podataka iz baza podataka. Prilikom unosa naziva stupaca potrebno je paziti da se unese pravilan raspored naziva kao što smo to radili prilikom unosa da bi nam svaki podataka u tablici bio na mjestu.

Slika 41: Dodavanje naziva stupaca u '*DataGridWiew'*




Izvor: izrada autora

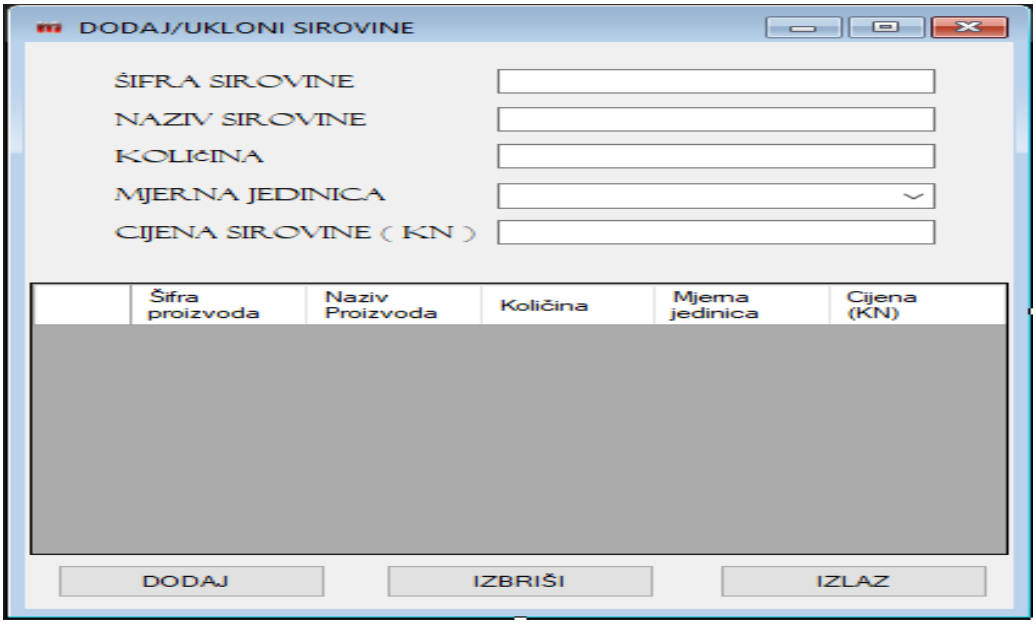
Slika 42: Kôd forme ‘Skladište kafića’

```
namespace Stock_final_V2._0.mainForms
{
    5 references | 0 changes | 0 authors, 0 changes
    public partial class SkladisteKafic : Form
    {
        1 reference | 0 changes | 0 authors, 0 changes
        public SkladisteKafic()
        {
            InitializeComponent();
            ucitajPodatke();
        }
        2 references | 0 changes | 0 authors, 0 changes
        public void ucitajPodatke()
        {
            SqlConnection con = new SqlConnection(@"Data Source=DESKTOP-1EHSVTF\ZAVRSNI_RAD;Initial Catalog=kafic;Integrated Security=True");
            SqlDataAdapter sda = new SqlDataAdapter("select*from [dbo].[Kafic]", con);
            DataTable dt = new DataTable();
            sda.Fill(dt);
            dataGrid.Rows.Clear();
            foreach (DataRow item in dt.Rows)
            {
                int n = dataGrid.Rows.Add();
                dataGrid.Rows[n].Cells[0].Value = item["Sifra"].ToString();
                dataGrid.Rows[n].Cells[1].Value = item["naziv"].ToString();
                dataGrid.Rows[n].Cells[2].Value = item["kolicina"].ToString();
                dataGrid.Rows[n].Cells[3].Value = item["mjernaJedinica"].ToString();
                dataGrid.Rows[n].Cells[4].Value = item["cijena"].ToString();
            }


        }

        1 reference | 0 changes | 0 authors, 0 changes
        private void btnRefresh_Click(object sender, EventArgs e)
        {
            ucitajPodatke();
        }

        1 reference | 0 changes | 0 authors, 0 changes
        private void btnExit_Click(object sender, EventArgs e)
        {
            this.Close();
        }

        1 reference | 0 changes | 0 authors, 0 changes
        private void btnAddDelete_Click(object sender, EventArgs e)
        {
            dodajUkloniKafic dk = new dodajUkloniKafic();
            dk.Show();

        }
```

Izvor: izrada autora

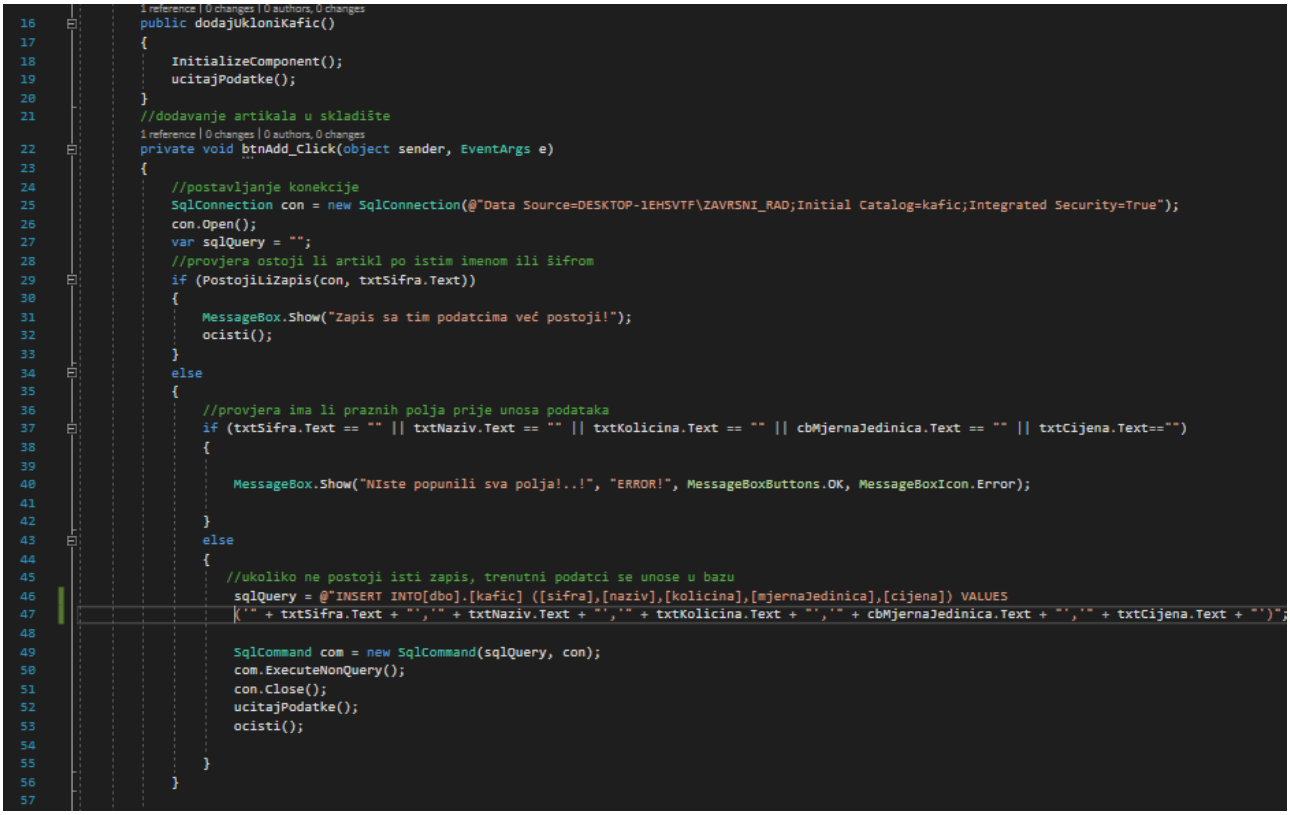
U ovome dijelu je novo samo učitavanje podataka u ‘*DataGridWiew’*. Svaki stupac ‘*DataGridWiew-a’* ima svoj broj odnosno index, te dodajemo podatak po podatak u njega.

Slika 43: Unos artikala u skladište




Izvor: izrada autora

Slika 44: Kôd forme ‘Dodaj/ukloni artikl’

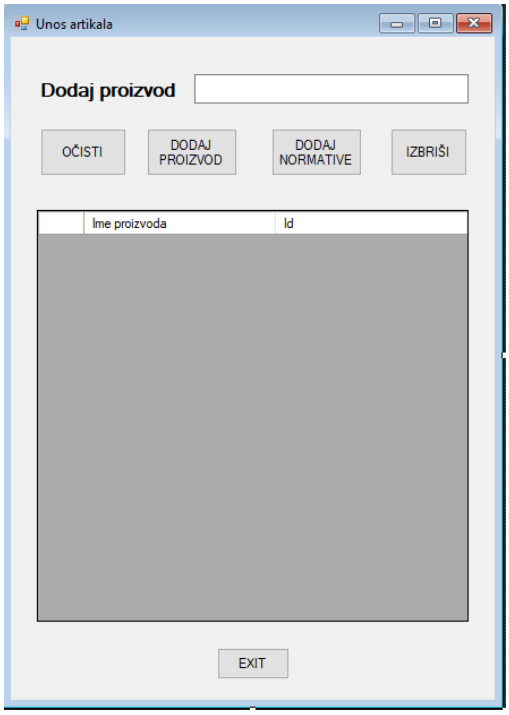


```
1 reference | 0 changes | 0 authors, 0 changes
public dodajUkloniKafic()
{
    InitializeComponent();
    ucitajPodatke();
}
//dodavanje artikala u skladište
1 reference | 0 changes | 0 authors, 0 changes
private void btnAdd_Click(object sender, EventArgs e)
{
    //postavljanje konekcije
    SqlConnection con = new SqlConnection(@"Data Source=DESKTOP-1EHSVTF\ZAVRSNI_RAD;Initial Catalog=kafic;Integrated Security=True");
    con.Open();
    var sqlQuery = "";
    //provjera ostoji li artikl po istim imenom ili šifrom
    if (PostojiLiZapis(con, txtSifra.Text))
    {
        MessageBox.Show("Zapis sa tim podatcima već postoji!");
        ocisti();
    }
    else
    {
        //provjera ima li praznih polja prije unosa podataka
        if (txtSifra.Text == "" || txtNaziv.Text == "" || txtKolicina.Text == "" || cbMjernaJedinica.Text == "" || txtCijena.Text=="")
        {

            MessageBox.Show("NIste popunili sva polja!..!", "ERROR!", MessageBoxButtons.OK, MessageBoxIcon.Error);


        }
        else
        {
           //ukoliko ne postoji isti zapis, trenutni podatci se unose u bazu
            sqlQuery = @"INSERT INTO[dbo].[kafic] ([sifra],[naziv],[kolicina],[mjernaJedinica],[cijena]) VALUES
            ('" + txtSifra.Text + "','" + txtNaziv.Text + "','" + txtKolicina.Text + "','" + cbMjernaJedinica.Text + "','" + txtCijena.Text + "')";

            SqlCommand com = new SqlCommand(sqlQuery, con);
            com.ExecuteNonQuery();
            con.Close();
            ucitajPodatke();
            ocisti();

        }
    }
```

Izvor: izrada autora

Kako smo dodali artikle u skladište, vrijeme je da unesemo i gotove proizvode koji se sastoje od nekoliko artikala. Ako razmislimo malo, doći će mo do zaključka da je teško napraviti naziv proizvoda a da u njega možemo dodavati podatke o proizvodima. Jedino rješenje koje nam preostaje je da taj proizvod koji će mo unesti, bude zapravo ime tablice koja će sadržavati ista imena kao i tablice u kojima su spremljeni artikli. Idemo pogledati kako to napraviti.

Slika 45: Grafičko rješenje forme ‘Unos artikala’




Izvor: izrada autora

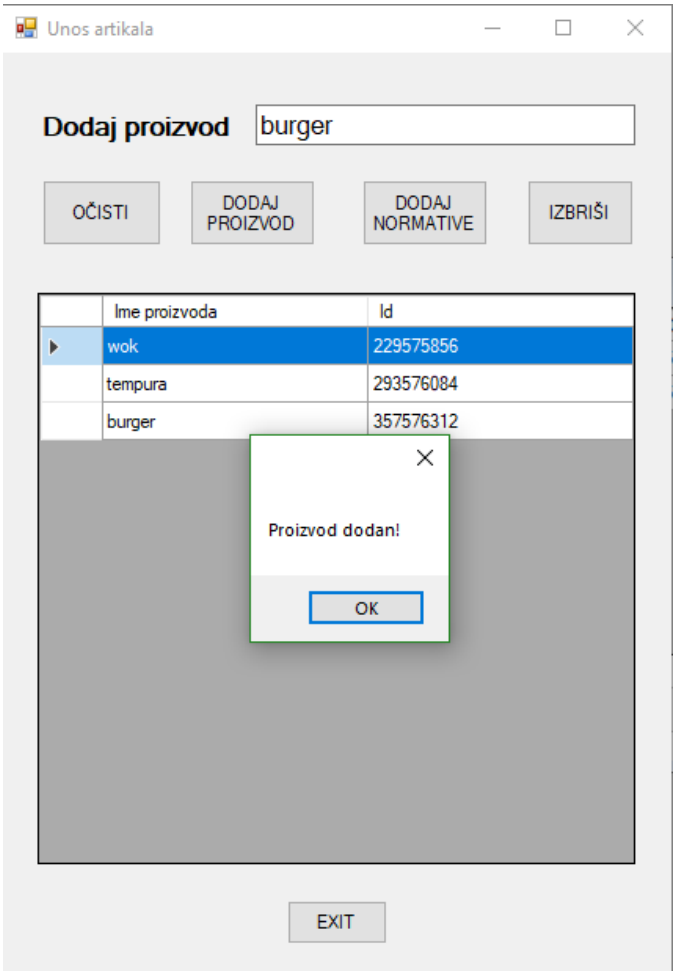
Slika 46: Kreiranje tablice preko Visual studija

```
1 reference | 0 changes | 0 authors, 0 changes
private void btnAdd_Click(object sender, EventArgs e)
{
    try
    {
        SqlConnection conn = new SqlConnection(@"Data Source=DESKTOP-1EH5VTF\ZAVR5NI_RAD;Initial Catalog=normativi_kafic;Integrated Security=True");
        SqlCommand cmd = new SqlCommand("" + kreirajTablicu(txtProizvod.Text) + " (  [sifra][int] PRIMARY KEY, " +
                                                                                    "[naziv][varchar](50), " +
                                                                                    "[Kolicina][varchar](50)," +
                                                                                    "[mjernaJednica][varchar](50), " +
                                                                                    "[cijena][varchar](50) )", conn);
        conn.Open();
        cmd.ExecuteNonQuery();
        MessageBox.Show("Proizvod dodan!");
        conn.Close();
        txtProizvod.Clear();
        ucitajPodatke();
    }
    catch (Exception ex)
    {
        MessageBox.Show("exception occured while creating table:" + ex.Message + "\t" + ex.GetType());
    }
}
1 reference | 0 changes | 0 authors, 0 changes
static string kreirajTablicu(string name)
{
    string tbName = name;
    string query = "CREATE table " + tbName + "";
    return query;
}
```

Izvor: izrada autora

Opet napominjem, bitno je paziti da se dobro unesu nazivi i tipovi podataka kako kasnije '*Debugger'* nebi izbacivao pogreške.

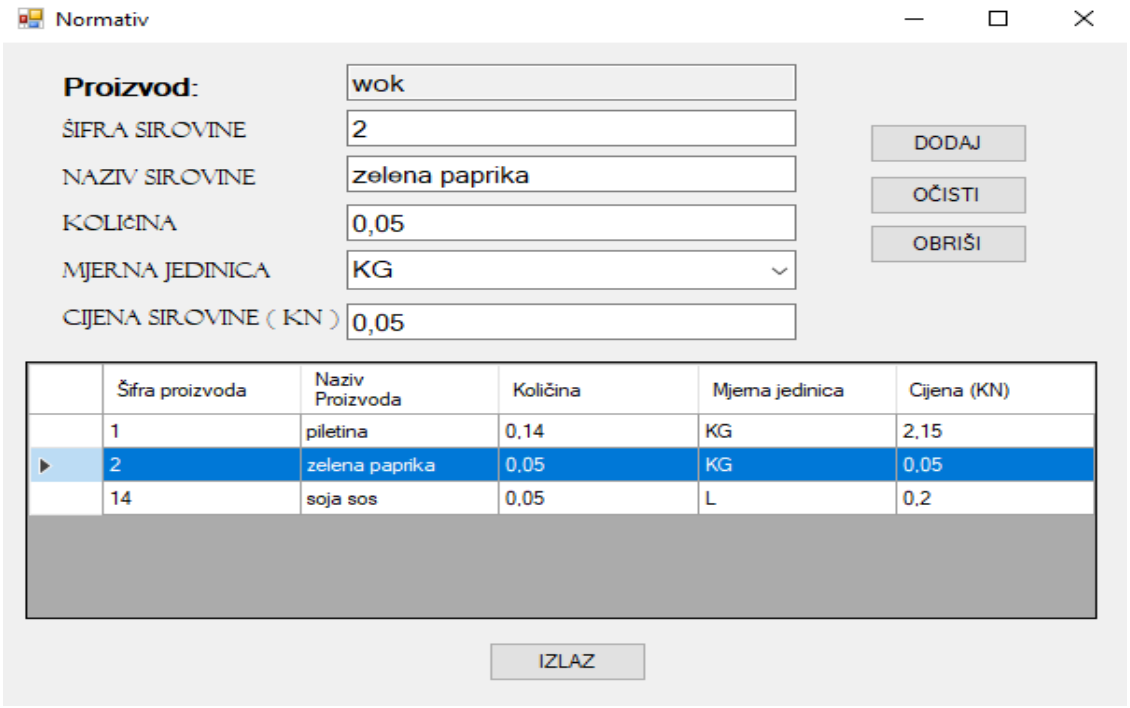
Slika 47: Prikaz unesenog proizvoda




Izvor: izrada autora

Sada kada smo kreirali tablicu, vrijeme je da u nju unesemo normative.

Slika 48: Forma unosa normativa




Izvor: izrada autora

Slika 49: Kôd za dodavanje normativa

```
1 reference | 0 changes | 0 authors, 0 changes
private void btnAdd_Click(object sender, EventArgs e)
{
    if (txtID.Text == " " || txtKolicina.Text == "" || txtNaziv.Text == "" || txtMjerna.Text == "" || txtCijena.Text == "")
    {
        MessageBox.Show("Niste unijeli sve podatke!", "ERROR", MessageBoxButtons.OK, MessageBoxIcon.Error);
        Brisanje();
    }
    else
    {
        SqlConnection con = new SqlConnection(@"Data Source=DESKTOP-1EHSVTF\ZAVRSNI_RAD;Initial Catalog=normativi_kafic;Integrated Security=True");
        con.Open();
        var sqlQuery = "";
        if (PostojiLiZapis(con, txtRoot.Text,txtID.Text))
        {
            sqlQuery = @"" + updateData(txtRoot.Text) + " set [Sifra] = '" + txtID.Text + "'," +
                                                        "[naziv] = '" + txtNaziv.Text + "'," +
                                                        "[Kolicina] ='" + txtKolicina.Text + "'," +
                                                        "[mjernaJednica] = '" + txtMjerna.Text + "'," +
                                                        "[cijena] = '" + txtMjerna.Text + "'   WHERE [sifra] = '" + txtID.Text + "'";
        }
        else
        {
            sqlQuery = @"" + addData(txtRoot.Text) + " ([sifra]  ,[naziv] ,[Kolicina] ,[mjernaJednica] ,[cijena] ) " +
                "VALUES ('" + txtID.Text + "','" + txtNaziv.Text + "','" + txtKolicina.Text + "','" + txtMjerna.Text + "','" + txtCijena.Text + "')";
        }
        SqlCommand com = new SqlCommand(sqlQuery, con);
        com.ExecuteNonQuery();
        con.Close();
        ucitajPodatke();
        Brisanje();
    }
}
```

Izvor: izrada autora

Logika je ista. Ukoliko nešto od popunjenih polja nije upisano, izbaciti će grešku, te tražiti korisnika da ponovo unese podatke. Ukoliko je sve dobro upisano, a podatci postoje, polje će se ažurirati, a ukoliko ne postoji, program će unijeti upisane podatke u bazu.

### 3.4. Korištenje

Nakon izrade aplikacije slijedi detaljno testiranje te ispravak ukoliko je potrebno. Treba razumijeti da se u fazi testiranja nemoraju nužno naći svi '*Bugovi'* te da uvijek postoji mogućnosti da će i dostavljena verzija biti problematična odnosno da će u njoj postojati neki problem u kôdu. Zato uz odabir poduzeća koji će odraditi izradu aplikacije, bitan faktor je i održavanje te responzivnost odnosno odaziv poduzeća na prijavljeni kvar kao i brzina otklanjanja problema.

Nakon detaljnog testiranja aplikacija ide u upotrebu. Kako ovo nije komercijalna aplikacija i ne ide u daljnju distribuciju, ovdje ćemo završiti sa tutorijalom.

# 4. ZAKLJUČAK

Zaključak bih želio iskoristiti prvo da se zahvalim mentoru i svima koji su bili uvijek dostupni pri pisanju završnog rada, kao i tokom mojeg studiranja. Svima onima odvažnima bih htio preporučiti da se upuste u programiranja te da iskreno uživaju. Garantiram Vam da nema boljeg osjećaja nego kada vam kôd radi te kada se vidi rezultat Vašeg rada i studiranja. Biti će poteškoća, uspona i padova. Provesti ćete vremena i vremena tražeći neku logičku grešku, možda ćete biti i nervozni, ali nemojte odustati. U današnje vrijeme ima dovoljno literature i sve se može pronaći na internetu ili u knjižnici. Isto tako sam siguran da nema boljeg osjećaja kada nakon nekoliko dana traženja problema u kôdu pronađete i otklonite problem te program proradi.

Za mene je ovo bilo veliko i zanimljivo iskustvo. Radeći na završnom radu stekao sam dodatno iskustvo koje dijelim sa ostalim kolegama. Rad me i motivirao za nastavak školovanja i pronalazak posla u IT sektoru. Na fakultetu su svi profesori visoko obrazovani i motivirani, te im se u bilo kojem trenutku možete obratiti za pomoć.

# POPIS SLIKA

# LITERATURA

## KNJIGE, PUBLIKACIJE I ČASOPISI


1. G. ELLIOTT
   Global business information technology: an integrated systems approach. Harlow, England; New York: Pearson Addison Wesley, 2004.


## INTERNETSKI IZVORI:


1. WIKIPEDIJA
   https://hr.wikipedia.org/wiki/Microsoft_SQL_Server, 08.07.2018.
2. MICROSOFT
   https://docs.microsoft.com/en-us/sql/ssms/sql-server-management-studio-ssms?view=sql-server-2017, 08.07.2018.
3. TEČAJEVI
   http://tecajevi.freeservers.com/iskoncept.htm, 08.07.2018.
4. DR. SC. TIHANA GALINAC GRBAC, UNIRI
   http://www.riteh.uniri.hr/zav_katd_sluz/zr/nastava/proginz/materijali/Dizajn%20programskog%20proizvoda.pdf, 08.07.2018
5. EuroComputerSystems
   http://www.ecs.hr/it-rjesenja-i-usluge/instalacija-implementacija-integracija/, 08.07.2018.