BRZI RAZVOJ APLIKACIJA U PROGRAMSKOM JEZIKU C#

Kostanjevac, Jurica

VELEUČILIŠTE U ŠIBENIKU
ODJEL MENADŽMENTA
PREDDIPLOMSKI STRUČNI STUDIJ MENADŽMENT

Jurica Kostanjevac

BRZI RAZVOJ APLIKACIJA U PROGRAMSKOM JEZIKU C#

ZAVRŠNI RAD

ŠIBENIK, 2018.

VELEUČILIŠTE U ŠIBENIKU
ODJEL MENADŽMENTA
PREDDIPLOMSKI STRUČNI STUDIJ MENADŽMENT

BRZI RAZVOJ APLIKACIJA U PROGRAMSKOM JEZIKU C#

ZAVRŠNI RAD

Kolegij: Objektno orijentirano programiranje
Mentor: dr. sc. Frane Urem
Student: Jurica Kostanjevac
Matični broj studenta: 0248034312

ŠIBENIK, 2018.

Sadržaj

1. UVOD 1
2. C# 2
2.1. C# Općenito 2
2.2 Povijest razvoja C# jezika 4
3. .NET ARHITEKTURA 7
4. VISUAL STUDIO 9
4.1. Osnove rada u Visual Studiu 10
5. BAZA PODATAKA 14
5.1. Baza podataka općenito 14
5.2. DBMS 14
5.3. SQL 16
5.4. SQL Server Data Tools 17
6. IZRADA JEDNOSTAVNE APLIKACIJE 18
6.1. Prva forma pregled koda 19
6.2. Postavljanje veze i TextBox-ova 20
6.3. Login 21
6.4. Druga forma 22
6.5. Druga forma pregled koda 23
6.6. Analiza koda druge forme 24
6.7. Izrada baze podataka 25
7. ZAKLJUČAK 28
LITERATURA 29

TEMELJNA DOKUMENTACIJSKA KARTICA

Veleučilište u Šibeniku

Odjel Menadžmenta

Preddiplomski stručni studij Menadžment

Završni rad

BRZI RAZVOJ APLIKACIJA U PROGRAMSKOM JEZIKU C#

JURICA KOSTANJEVAC

Samobor Perkovčeva 79, 10430, jkostanj@vus.hr

Sa razvojem integriranih razvojnih okolina omogućen je brzi razvoj aplikacija u modernim objektno orijentiranim programskim jezicima. Istovremeno su takvi alati postali dostupni sve većem broju programera te je cijeli proces razvoja aplikacija postao brži i otporniji na programske greške. Jedan od najraširenijih objektno orijentiranih programskih jezika opće namjene je C# koji je dio .NET arhitekture i Visual Studio razvojne okoline. U ovom radu se opisuje evolucija C# programskog jezika te osnovna struktura .NET arhitekture kao i Visual Studio razvojne okoline. Također se prikazuje brzi razvoj klijentske aplikacije s grafičkim sučeljem koja pristupa bazi podataka. Svi dijelovi prikazanog rješenja su izrađeni u Visual Studio razvojnoj okolini.

(28 stranica / 12 slika / 0 tablica / 21 literarni navod / jezik izvornika: hrvatski)

Rad je pohranjen u: Knjižnici Veleučilišta u Šibeniku

Ključne riječi: C#, aplikacija, login, baza podataka, brzi razvoj

Mentor: dr.sc. Frane Urem, prof.v.š.

Rad je prihvaćen za obranu.

BASIC DOCUMENTATION CARD

Polytechnic of Šibenik

Final paper

Department of Management

Undergraduate Professional Study Program

RAPID APPLICATION DEVELOPMENT IN C# PROGRAMMING LANGUAGE

JURICA KOSTANJEVAC

Samobor Perkovčeva 79, 10430, jkostanj@vus.hr

With development of integrated development enviroments (IDE) was enabled rapid application development in modern object oriented programing languages. At the same time sutch tools have become available to an increasing number of programers, also the entire process of application development became faster and more resilient to programming errors. One of the most widespread object oriented programming languages for general use is C# wich is part of .NET architecture and Visual Studio development enviroment. In this paper is described the evolution of C# programming language, also basic structure of .NET architecture and Visual Studio development enviroment. Also is shown rapid development of client application with graphical interface that acceses a database.

(28 pages / 12 pictures / 0 tables / 21 references / original language: croatian)

Paper deposited in: Library of Polytechnic in Šibenik

Keywords: C#, application, login, database, rapid development

Supervisor: dr.sc. Frane Urem, prof.v.š.

Paper accepted

1. UVOD

U ovom radu govorit će se o brzom razvoju aplikacije u C# programskom jeziku uz pomoć Visual Studio razvojnog sučelja. Aplikacija je povezana s bazom podataka, na koju se spaja pomoću unosa korisničkog imena i lozinke. Rad se sastoji od dva djela: prvi je teoretski a drugi je praktični.

U prvom djelu opisuju se programski jezici ali i programi korišteni za izradu aplikacije. U ovom slučaju su korišteni programski jezici C# i SQL a projekt je rađen u Visual Studio razvojnom sučelju.

A u praktičnom djelu opisuje se izrada projekta počevši od dizajna formi i njihove uloge u interakciji s korisnikom nakon toga kod je prikazan i objašnjen . Budući da je projekt povezan sa SQL-om grafički će se prikazati tablice i opisati njihova izrada u Visual Studiu.

2. C#

2.1. C# Općenito

C# je objektno orijentirani programski jezik za brzi razvoj aplikacija (eng. *Rapid application development*) RAD. Ovaj programski jezik je razvijen u sklopu Microsoftove .NET inicijative koja je trebala ublažiti probleme s kompatibilnosti i omogućiti pokretanje aplikacija na različitim operativnim sustavima (OS), ali također i pružiti mogućnosti koje programerima nude gotova rješenja i funkcionalnosti da bi ubrzala i pojednostavila razvoj aplikacija svih vrsta i oblika.

C# je sličan jezicima C i C++ iz kojih je proizašao također ima jako velike sličnosti s Javom (naročito u ranijim verzijama) što je u vrijeme kada je C# izašao pobudilo dosta kontroverze ali također omogućilo lak prijelaz s jednog programskog jezika na drugi. „Zapravo C# spaja moć i efikasnost C++-a, jednostavni i čisti objektno orijentirani dizajn Jave sa pojednostavljenim korištenjem kao npr. u Visual Basic-u.“[1]

C# je programski jezik koji se povodi za događajima (eng. *Event driven*) u njemu se pišu programi koji reagiraju na događaje koje je napravio korisnik npr: klikovi mišem, pritiskom tipki na tipkovnici, dodirom na dodirnom ulaznom uređaju (*touchpad*) i ostalim gestama koje se rabe na računalima, pametnim telefonima i tabletima. Također C# omogućava asinhrono programiranje u kojem više zadataka može biti obavljeno u isto vrijeme što čini aplikacije više reaktivnima na korisnikovu interakciju s njima.

Kao objektno orijentirani programski jezik važne karakteristike C#-a su: enkapsulacija, nasljeđivanje i polimorfizam.

[1] Faraz Rasheed, Programmers Heaven: C# School First Edition, str.17

Enkapsulacija: Tradicionalni programski jezici imaju razdvojene procese od podataka, u objektno orijentiranim sustavima procesi i podaci su spojeni u objektima. To „objedinjavanje atributa koji opisuju podatke i metode koje nad takvim podacima mogu izvršiti određene postupke u jedan entitet je mehanizam koji se naziva apstrakcija stvarnog objekta“[2].

No znanje korisnika o tome kako je implementiran spoj podataka i procesa nije od ključne važnosti za korisnika, u objektno orijentiranom okruženju i dovoljno je da poznaje sučelje preko kojeg može utjecati na procese i podatke u objektu. Taj način skrivanja sakrivanja apstrakcije kako bi izložili funkcionalnost zove se enkapsulacija.

Nasljeđivanje: nasljeđivanje (eng: *inheritance*) je karakteristika objektno orijentiranih programskih jezika koja omogućuje da se definira bazna klasa koja pruža specifikacije funkcionalnosti (podataka i procesa) i da definira klase koje su proizašle iz bazne klase da naslijede tu funkcionalnost.

Na primjer imamo super klasu računalo koja definira sve nama bitne atribute koje ima računalo, kao pod klasu imamo prijenosno računalo koje nasljeđuje atribute i funkcije od super klase računalo ali ih preciznije definira u skladu sa našim potrebama.

Polimorfizam: Dolazi od grčke riječi polymorph što u slobodnom prijevodu znači „više oblika“ odnosno „jedno ime za više oblika“ u smislu da je nešto otvoreno za interpretaciju.

Tako u objektno orijentiranom programiranju polimorfizam je karakteristika koja omogućuje da imamo klase sa više implementacija sa istim imenom i da se upotrebljavaju u kontekstu objekta nad kojim se izvode.

[2] Frane Urem, Uvod u objektno orijentirano programiranje s primjerima, Šibenik, str.11

2.2 Povijest razvoja C# jezika

Kao što je već spomenuto C# je razvijen sa ciljem maksimalnog korištenja mogućnosti .NET arhitekture, najznačajnija osoba u njegovom razvoju je bio Anders Hejlsberg koji je u siječnju 1999. godine formirao tim kako bi razvili novi objektno orijentirani programski jezik pod nazivom Cool (*C-like Object Oriented Language*).

Microsoft je planirao zadržati ime „Cool" kao ime jezika ali od toga su odustali zbog autorskih prava nad sličnim nazivom i zaštitnim znakom. Do vremena kada je .NET projekt bio javno najavljen u srpnju 2000. Godine na profesionalnoj konferenciji developera (eng. *Professional Developers Conference*), jezik je bio preimenovan u C#.

Anders Hejlsberg je vodeći dizajner i glavni arhitekt C# jezika u Microsoftu, a prije toga je bio umješan u razvoju Turbo Pascal prevoditelja za MS DOS operacijski sustav i kasnije objektno orijentiranu izvedenicu Pascala pod nazivom Delphi. „U intervjuima i tehničkim papirima Hejlsberg je izjavio da greške u većini većih programskih jezika (npr. C++, Java, Delphi) vuku osnove u *Common Language Runtimeu* (CLR), koji je postavljen kao osnova (odnosno dio .NET arhitekture) C# jezika kako bi se izbjegli takvi problemi."[3]

James Gosling, koji je razvio objektno orijentirani jezik Java 1994. godine nazvao je C# „imitacijom" Jave također je izjavio da „C# je kao vrsta Jave sa pouzdanosti, produktivnosti i sigurnosti izbrisanima."[4] Klaus Kreft i Angelika Langer (autori C++ streams book-a) su izjavili preko bloga: „Java i C# su gotovo identični programski jezici. Dosadne repeticije kojima nedostaje inovacija" i da „C# je posudio puno toga od Jave i obrnuto". U srpnju 2000. godine Hejlsberg je rekao da „C# nije klon jave i puno je bliži C++-u u svom dizajnu".

Puno toga se promijenilo u proteklih 18 godina, C# i Java su evoluirali postepeno sve različitijim putevima. Izašle su četiri glavne verzije Jave dok je za C# izašlo šest verzija, u tih šest verzija C# je vidio puno inovacije.

[3] https://en.wikipedia.org/wiki/C_Sharp_(programming_language) (16.5.2018)
[4] https://www.cnet.com/news/why-microsofts-c-isnt/ (16.5.2018)

Kako bi bolje vidjeli razvoj C#-a ukratko ćemo proći koja nova obilježja su bila dodana u većim verzijama.

C# 1.0

Ako se pogleda prvu verziju C#-a stvarno je izgledala jako slično ranijim verzijama jave. Kako bi zadovoljili ciljeve ECMA, C# je trebao biti dizajniran kao jednostavni, moderan, objektno orijentirani jezik široke upotrebe.

Prva verzija C# iz današnje perspektive bi izgledala jako osnovno i bez značajki. Mogao se pisati kod i to je više-manje bilo sve. Radio je na .NET frameworku 1.0 i dolazio je uz Visual Studio .NET 2002.

C# 2.0

C# možda je počeo kao generični objektno orijentirani jezik ali sa izlaskom druge verzije se to brzo promijenilo. Dodano je dosta značajki koje su olakšale rad developerima.

Neke od važnijih promjena su: generični tipovi koji su predlošci za klase ili metode kako bi mogle raditi sa bilo kojim tipom podataka, djelomične klase sa kojima se klasa može odjeliti na više datoteka u istom projektu, anonimne metode koje nemaju ime sastoje se samo od koda, null-vrijednosti koje pomažu u zastupanju nedefiniranih vrijednosti, ineratore koji su metode koje olakšavaju pozivanje klasa prije nego se nastavi izvršavanje izvornog koda, i kovarijance i kontravarijance. Radio je na .NET frameworku 2.0 i dolazio je uz Visual Studio 2005.

C# 3.0

Sa trećom verzijom C# stvarno postaje moćan programski jezik (makar većina dodatnih značajki dolazi u verziji 3.5). „Najznačajnija nova funkcija je LINQ (*Language INtegrated Query*) što omogućava korisniku pisanje u stilu SQL-a[5]“. Također dolaze bitne funkcije kao:

Auto implementirana svojstva što omogućava deklariranje svojstava i omogućava kreiranje objekata, anonimni tipovi koji omogućavaju kreiranje tipova bez da ih se definira, metode ekstenzije koje omogućavaju dodavanje metoda već postojećem tipu bez potrebe za

[5] https://docs.microsoft.com/en-us/dotnet/csharp/linq/query-expression-basics (18.5.2018)

izmjenama tipa i lambda izrazi. Radio je na .NET frameworku 3.0 i dolazio je uz Visual Studio 2008.

C# 4.0

Iako se verzije C#-a 4.0 ne može mjeriti sa inovacijom koju je donijela verzija 3.0, razvijene su zanimljive nove značajke u ovoj verziji. Nove značajke su: dinamično vezivanje, što je mogućnost zaobilaženja provjera compilera gdje je primijenjena ova operacija; imenovani i opcionalni argumenti; „generične varijance i kontravarijance, one omogućavaju veću fleksibilnost u dodjeljivanju i korištenju generičnih tipova“[6] i ugrađeni interoperabilni tipovi podataka. Radio je na .NET frameworku 4.0 i dolazio je uz Visual Studio 2010.

C# 5.0

Verzija 5.0 je bila jako fokusirano razvijena verzija u kojoj je dodano par revolucionarnih značajki. Nove značajke su: „asinkroni članovi, koji poboljšavaju performanse cijelog programa na način da kompajler radi posao koji je inače programer trebao raditi a aplikacija zadržava logičnu strukturu kao u sinkronom kodu[7]“ i „*caller info*“ atributi, koji omogućavaju lakše praćenje informacija o pozivnoj metodi ili slično. Radio je na .NET frameworku 4.5 i dolazio je uz Visual Studio 2012.

C# 6.0

U ovoj verziji nema revolucionarnih novih značajki ali Microsoft se posvetio značajkama koje kod čine preglednijim i čitljivijim. Neke od novih značajki su: statički importi, filterom iznimki, inicijalizatorom svojstava, interpolacijom stringova i inicijalizator rječnika. Radio je na .NET frameworku 4.6 i dolazio je uz Visual Studio 2015.

C# 7.0

„Trenutno zadnja velika verzija C#-a odane su nove značajke koje su dodatno „ispolirale“ jezik[8].“ Neke od značajki su: „*out*“ varijabla, torke i dekonstrukciju, podudaranje obrazaca, lokalne funkcije. To je omogućilo pisanje još preglednijeg i urednijeg koda. Radio je na .NET frameworku 4.7 i dolazio je uz Visual Studio 2017.

[6] https://docs.microsoft.com/en-us/dotnet/standard/generics/covariance-and-contravariance (18.5.2018)
[7]https://docs.microsoft.com/en-us/previous-versions/visualstudio/visual-studio-2012/hh191443(v=vs.110) (18.5.2018)
[8] https://blog.ndepend.com/c-versions-look-language-history/ (18.5.2018)

3. .NET ARHITEKTURA

„Najjednostavnije rečeno, to je sustav koji nadograđuje mogućnosti samog operativnog sustava. Radi se o posebnoj infrastrukturi koja programerima nudi gotova rješenja i funkcionalnosti da bi ubrzala i pojednostavila razvoj aplikacija svih vrsta i oblika.[9]“

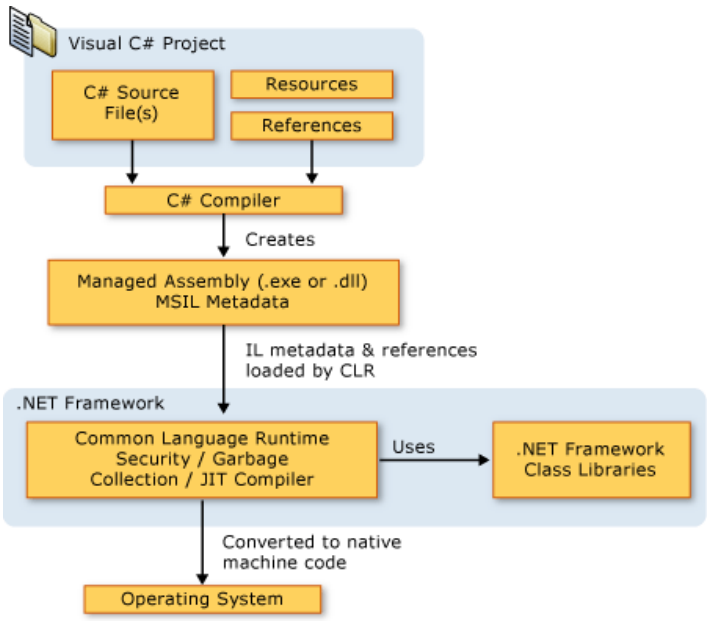
Najvažnije sastavnice .NET Frameworka su Common Language Runtime (CLR) i sjedinjeni set *class libraries*-a. Aplikacije za .NET platformu moguće je pisati u raznim programskim jezicima, ali zapravo CLR ne prepoznaje niti jedan taj jezik. CLR dobiva naredbe preko posrednog jezika koji se zove Microsoft Intermidiate Language (MSIL). Stoga je jasno da moraju postojati kompajleri koji programske jezike u kojima developeri pišu kod prevode u MSIL kako bi ga mogao razumjeti CLR. Takvi kompajleri zovu se IL kompajleri i dostupni su za velik broj programskih jezika, Microsoft je izdao kompajlere za pet jezika: C#, J#, C++, Visual Basic i JScript, dok su se ostali proizvođači potrudili oko mnogih drugih jezika.

Budući da se naredbe svih jezika koje podržava .NET prvo pretvaraju u MSIL postaje svejedno u kojemu od njih će se pisati aplikacije također zbog ove značajke proizlazi velika prednost .NET-a u obliku višejezičnog pisanja aplikacija. U ovakvom razvojnom okruženju više nije potrebno da programeri koji rade na nekom projektu svi poznaju isti programski jezik, važno je samo da je podrška za njihov jezik dostupna u .NET-u.

[9] https://hr.wikipedia.org/wiki/.NET_Framework (7.9.2018)

CLR radi tako da instrukcije koje su dobivene od MSIL-a u realnom vremenu prevode u izvorni strojni kod koji razumije računalo. Za taj zadatak služe JIT kompajleri (*Just In Time*). To prevođenje u izvorni strojni kod računala omogućuje .NET-u prelazak na druge operativne sustave kao što su Linux ili MacOS. „Mogućnosti koje CLR nudi su izuzetne, no same po sebi nisu dovoljno uporabljive iz ljudskog aspekta. Upravo zbog toga u .NET Frameworku postoje setovi klasa koje omogućavaju brzo i jednostavno korištenje mogućnosti koje CLR nudi[10]".

Slika 1: Shema .NET arhitekture



Izvor: https://docs.microsoft.com/en-us/dotnet/csharp/getting-started/media/netarchitecture.png

[10]https://docs.microsoft.com/en-us/dotnet/csharp/getting-started/introduction-to-the-csharp-language-and-the-net-framework (7.9.2018)

4. VISUAL STUDIO

Visual studio je integrirano razvojno sučelje (*Integrated Development Enviroment*, IDE) kojeg je razvio Microsoft. „To je program koji omogućuje razvoj web aplikacija, web usluga, mobilnih aplikacija i računalnih programa. Od verzije Visual Studio .NET 2002 pa nadalje moguće je višejezično pisanje programa koji mogu raditi na više operativnih sustava zbog uporabe moćne .NET arhitekture.“[11]

Visual Studio rabi Microsoftove razvojne platforme kao što su: Windows API, Windows Forms, Windows Presentation Foundation, Windows Store i Microsoft Silverlight. Preko Visual Studia moguće je dobiti izvorni strojni kod ili programski kod koji se izvršava preko CLR-a kada se piše program u njemu.

Microsoft Visual Studio je dostupan u sljedećim edicijama:

- **Community**- „Community edicija je izašla sa Visual Studiom 2015 kao nova besplatna verzija, sa sličnom funkcionalnosti kao Professional edicija. Prije te verzije jedina besplatna edicija Visual Studia je bila Express u kojoj je funkcionalnost bila ograničena. Za razliku od Express edicije Visual Studio Community podržava više jezika i pruža potporu za ekstenzije[12]“. Individualni developeri nemaju ograničenja pri uporabi ove edicije a organizacije koje se klasificiraju kao manji poduzetnici mogu imati do pet community kopija bez ikakvih ograničenja, iznimka su organizacije koje rabe ovu ediciju za edukaciju i akademska istraživanja.

- **Professiona**l- Od verzije Visual Studia 2010 početna komercionalna edicija je Professional. Ova edicija pruža integrirano razvojno okruženje za sve jezike koje podržava .NET arhitektura, ima potporu za Microsoft Development Network (MDN), podržava uređivanje XML koda i također podupire alate kao Server Explorer i njegovu integraciju sa Microsoft SQL Serverom. Ova edicija također podupire razvoj mobilnih aplikacija.

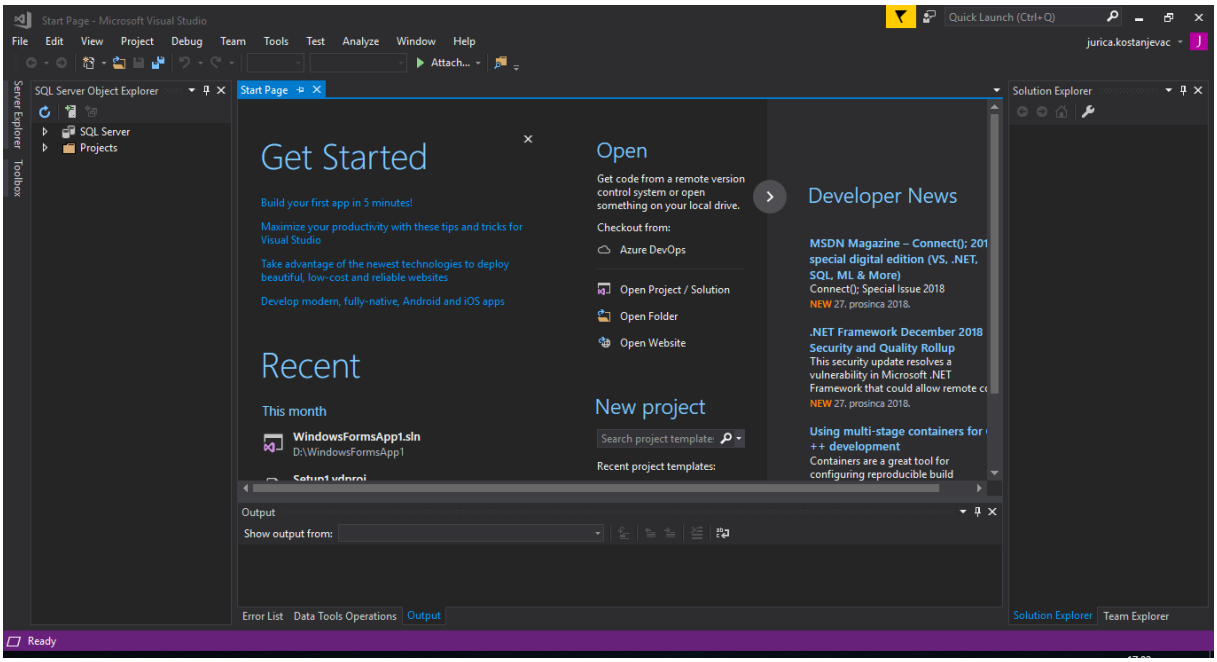
- **Enterprise**- Uz funkcije koje pruža Professional edicija, Enterprise pruža novi set alata za razvoj softvera, razvoj baza podataka, suradnju developera i alati za testiranje.

[11] https://bs.wikipedia.org/wiki/Microsoft_Visual_Studio (21.11.2018)
[12] https://en.wikipedia.org/wiki/Microsoft_Visual_Studio (21.11.2018)

4.1. Osnove rada u Visual Studiu

Kada se upali Visual Studio otvara se početna stranica na kojoj su prikazane novosti i moguća ažuriranja te projekti na kojima je nedavno radio korisnik koji se mogu otvoriti preko prečaca.

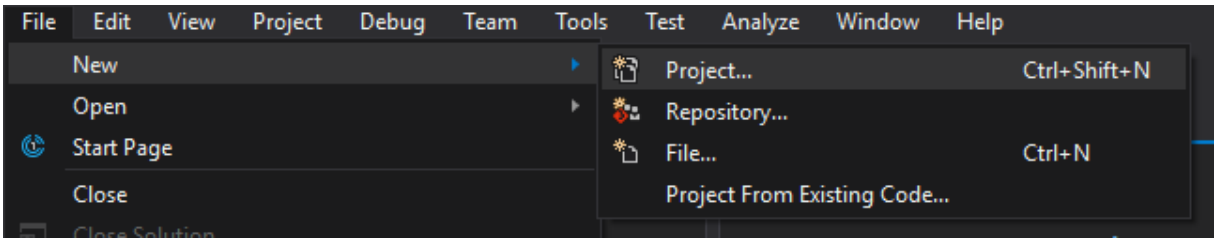
Slika 2: Početna stranica Visual Studia



Izvor: autorska slika

Novi projekt se otvara tako da se na alatnoj traci klikne na „*File*“ te u padajućem izborniku odabere „*New*“ i pod tim opcijama klikne „*Project*“.

Slika 3: Otvaranje novog projekta

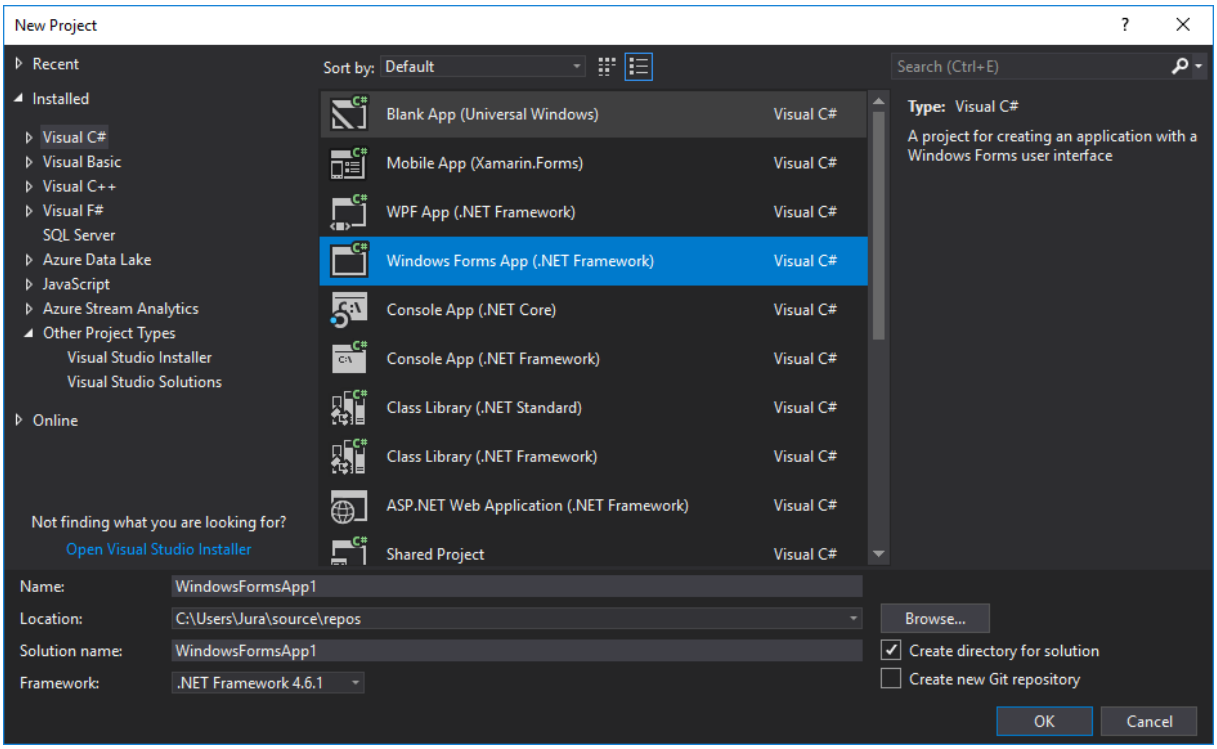


Izvor: autorska slika

„U ovom prozoru korisnik može odabrati u kojem će programskom jeziku pisati. Nakon što je jezik odabran pokaže se prozor sa mogućim predlošcima za odabrani jezik ovisno koji tip

projekta korisnik želi raditi u tom programskom jeziku.“[13] Predložak se sastoji od osnovnih datoteka i postavki potrebnih za neki projekt odabranog tipa. U kućici s desne strane nalazi se opis projekta odabranog tipa što pomaže pri odabiru. Na dnu prozora nalazi se polje gdje se može unijeti ime projekta, ime programa koji se radi i lokaciju spremanja. Nakon odabira lokacije i imena pritiskom na tipku „*OK*“ projekt se kreira, sprema i moguće je započeti rad na njemu.

Slika 4: Odabir vrste novog projekta

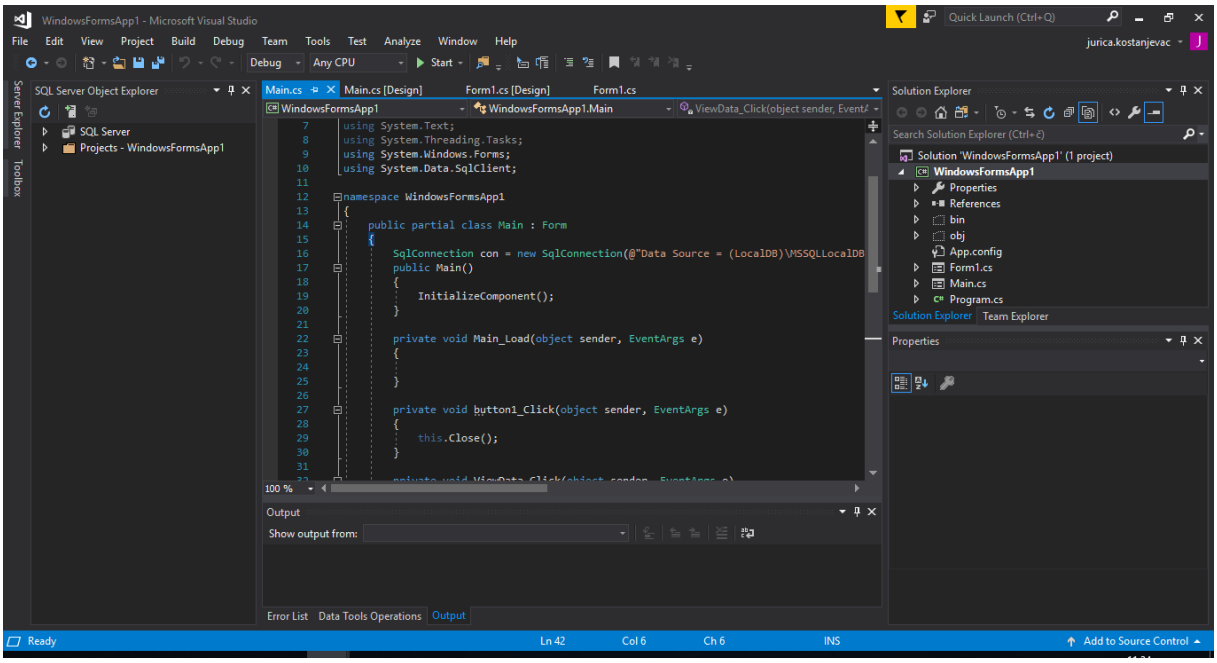


Izvor: autorska slika

[13] https://tutorials.visualstudio.com/cpp-console/create (4.12.2018)

„Nakon kreiranja novog projekta automatski se otvara novi prozor u kojem su pred korisnika predstavljeni svi alati i korisničko sučelje.“[14] Samo sučelje se sastoji od niza prozora koji prikazuju bitne informacije o projektu.

Slika 5: Sučelje Visual Studia



Izvor: autorska slika

Najvažniji prozori za rad u visual studiju su:

- Središnji prozor ili *Code Editor window* – „U tom prozoru će korisnik provoditi najviše vremena radeći na projektu u Visual Studiju, prikazuje sadržaj datoteke odabrane u Solution Exploreru izražen kao kod ili dizajn datoteke.“ [15]Tu se može uređivati kod ili dizajn korisničkog sučelja koje se radi, kao na primjer dodavanje tipki i polja za unos teksta.
- Solution Explorer – Dopušta korisniku da gleda i upravlja datotekama koda na kojemu radi. Također ovaj prozor je napravljen kako bi pomogao organizirati kod tako da se grupiraju datoteke u rješenja (*Solution*) i projekte.
- Izlazni prozor ili *Output Window* – To je prozor u kojem Visual Studio šalje notifikacije kao što su informacije o otklanjanu grešaka i poruke o postojećim greškama i gdje se nalaze također prikazuje upozorenja kompajlera i poruke

[14]https://visualstudiomagazine.com/blogs/tool-tracker/2018/08/controlling-default-window-layout.aspx (6.12.2018)
[15] https://docs.microsoft.com/hr-hr/visualstudio/get-started/visual-studio-ide?view=vs-2017 (6.12.2018)

vezane za status objavljivanja projekta. Svaka vrsta notifikacije ima svoju karticu u kojoj se prikazuje.

- Server Explorer – Prozor u kojemu se pregledavaju i uređuju veze sa bazama podataka i serverima. Preko Server Explorera moguće se spojiti na Microsoft Azure server ali također korisnik može kreirati i dodavati svoje konekcije i baze podataka zahvaljujući LINQ funkciji.
- Razvojni alati ili *Toolbox* – Ovom prozoru je moguće pristupiti jedino ako projekt ima grafičko korisničko sučelje

5. BAZA PODATAKA

5.1. Baza podataka općenito

Baze podataka su organizirane zbirke podataka u koje se pohranjuju i pristupaju podaci elektronički. Formalno „baza podataka“ je pojam koji se odnosi na set određenih podataka i način na koji je organiziran. Dizajneri baza podataka obično organiziraju podatke da modeliraju aspekte stvarnosti na način da potpomažu procesima kojima su potrebne te informacije. Na primjer baza podataka za parkiralište modelirana je na način koji potpomaže naći parkirališta sa slobodnim mjestima. Preko DBMS (*database management system*) programskog sustava omogućeno je korisniku kreiranje novih baza podataka kao i unošenje novih podataka te manipuliranje već postojećim podacima.

5.2. DBMS

Database management system omogućava krajnjim korisnicima da čitaju, upisuju, brišu i ažuriraju podatke u bazama podataka ali i postavljanje upita. DBMS zapravo služi kao sučelje između baze podataka i korisnika ili aplikacijskih programa, što osigurava da su podaci budu dosljedno organizirani i ostanu lagano dostupni.

DBMS upravlja s tri važne stvari: podatcima; database engineom koji omogućava pristup, spremanje i modifikaciju podataka; i shemom baze podataka koja definira logičku strukturu baze podataka.[16] Ta tri temeljna elementa pomažu pružiti sigurnost, integritet podataka i ujedinjene administrativne procedure.

DBMS je iznimno koristan jer omogućuje centralizirani pregled podataka kojima mogu pristupiti različiti korisnici na različitim lokacijama na kontrolirani način. Pomoću DBMS-a možemo ograničiti kojim podacima krajnji korisnik ima pristup, kao što i način pregledavanja podataka od strane korisnika tako da omogućuje više načina gledanja na jednu shemu baze podataka. Korisnici i software ne moraju biti svjesni gdje su podaci fizički pohranjeni ili na kojem tipu medija za pohranu su spremljeni zato jer svi upiti idu preko DBMS-a.

[16] https://searchsqlserver.techtarget.com/definition/database-management-system (3.3.2019)

Dok god programi rabe API (*application programming interface*) za bazu podataka koju pruža DBMS, developeri ne moraju modificirati programe ako je došlo do promjene u bazi podataka.

Od nastanka DBMS-a razvijeno je više sustava ovisno o modelu baze podataka.
Po tome se razlikuju:

1. **Rani modeli baza podataka**- Ovi modeli su bili popularni 1960-ima i 1970-ima, ali danas se mogu najprije naći na starim sustavima.[17] Okarakterizirani su primarno kao navigacijski sustavi sa jakim vezama između njihovih logičkih i fizičkih reprezentacija i nedostacima u neovisnosti podataka.
   1.1. **Hijerarhijski model**- „U hijerarhijskom modelu podaci su organizirani u strukturu nalik drvetu, što bi značilo da jedan entitet klasificiran kao „roditelj“ se grana na više zapisa podataka „djece“.“[18] Ovaj model omogućava jedan na više tip veze između dva entiteta. Hijerarhijske strukture su bile u širokoj upotrebi u ranim mainframe database management system-ima kao što je IMS (*Information Management System*) od IBM-a
   1.2. **Mrežni model**- Model napravljen kao nadogradnja već postojećeg hijerarhijskog modela sa ciljem rješavanja nedostataka u starom modelu. Budući da je hijerarhijski model rigidan promjene su bile ciljane na fleksibilnost sustava, tako da je omogućeno stvaranje veza više na više pomoću upisivanja više podataka na jedan entitet. [19]Ovaj model je bio jako popularan prije nego ga je zamijenio relacijski model
2. **Relacijski model**- Razvijen od strane E. F. Codd-a 1970. kao model koji bi učinio baze podataka manje ovisnima o bilo kojoj određenoj aplikaciji. Zasniva se na konceptu povezanih tablica, tablice se povezuju preko zajedničkih atributa. Danas naj raširenij model baza podataka.
3. **Objektni model**- U ovom modelu informacije su predstavljene u obliku objekata, kao u objektno orijentiranom programiranju. Objektne baze podataka se razlikuju od relacijskih baza podataka po tome što spajaju relacijski model koji je tablično orijentiran i objektno orijentiranu paradigmu. Što čini objektni model zapravo hibridnim modelom.

[17] https://en.wikipedia.org/wiki/Database_model#Early_data_models (3.3.2019)
[18] http://www.computerbusinessresearch.com/Home/database/hierarchical-database-model (4.3.2019)
[19] http://www.computerbusinessresearch.com/Home/database/network-database-model (4.3.2019)

5.3. SQL

SQL (*Structured Query Language*) je programski jezik dizajniran za upravljanje bazama podataka koje su u relational database-menagement systemu-u (RDBMS). [20]SQL je posebno koristan u radu sa strukturiranim podacima gdje se nalaze relacije između različitih entiteta ili varijabli podataka.

Također nudi dvije glavne prednosti nad starijim „čitaj/piši“ API softwareom (*application programming interface*) kao što su ISAM(*indexed sequential access method*) i VSAM (*Virtual Storage Access Method*): prvo, predstavio je koncept pristupa više zapisa sa jednom naredbom; drugo, maknuo je potrebu da se specificira put kako doći do zapisa odnosno nije potreban indeks.

„Originalno se bazirao na relacijskoj algebri i tuple relational calculusu-u, SQL se sastoi od više tipova izjava koje se mogu neformalno klasificirati kao podjezici, na primjer: DQL(*data query language*),DDL (*data definition language*),DCL (*data control language*), DML (*data manipulation language*).“ Krug djelovanja SQL-a uključuje upit podataka, manipulaciju podataka (insert, update, delete), definiranje podataka (stvaranje i modifikacija shema) i kontrola pristupa podacima. Makar je SQL često opisan kao deklarativan jezik (4GL) , također uključuje proceduralne elemente.

[20] https://www.tutorialspoint.com/sql/sql-overview.htm (17.3.2019)

5.4. SQL Server Data Tools

SSDT (*SQL Server Data Tools*) je set alata koji dopuštaju developerima aplikacija i baza podataka da vrše sav rad oko dizajniranja baza podataka za SQL Server i SQL Azure unutar Visual Studia. SSDT pruža bogato okruženje za razvoj baza podataka integrirano u Visual Studio, kao i alate koji se temelje na deklarativnom modelu koji se mogu rabiti za online i offline razvoj.[21]

Project system u SSDT-u omogućava developerima da imaju kopiju potpuno reprezentativnog izvornog koda (*source-code*) koja predstavlja online bazu podataka ali se rabi za offline development.

U oba scenarija, projektnom (*offline*) i povezanom (*online*), SSDT nudi dva vrijedna alata koja uvelike poboljšavaju produktivnost u razvoju baza podataka: Schema Compare i Table Designer.

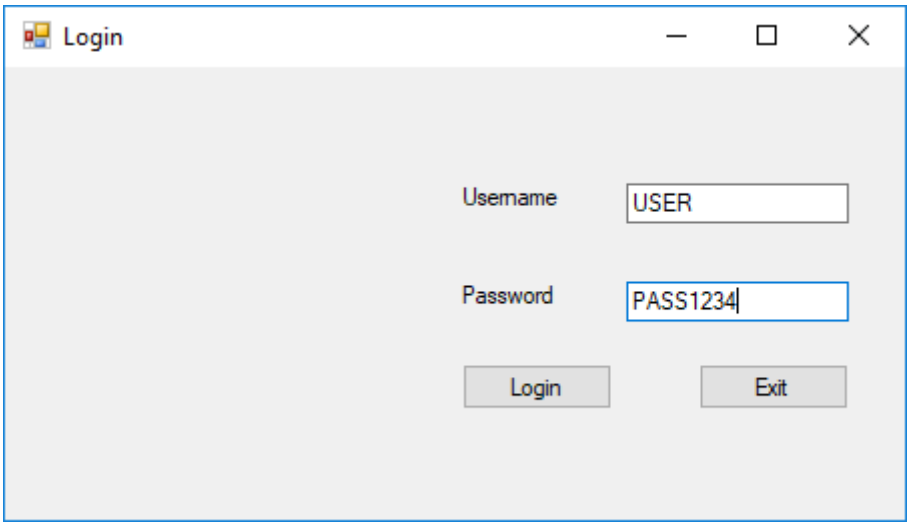
Developeri baza podataka mogu upotrijebiti Schema Compare alat kako bi usporedili i sinkronizirali sheme između online baza podataka i offline projekata. Ovaj alat omogućava da se vizualno usporede dvije sheme baze podataka, kao i mogućnost da se mijenja kod kako bi se promijenila baza podataka.

Table Designer u SSDT-u rabi se u prozoru zajedno sa prozorom kod-a, kako je tablica promijenjena u prozoru grafičkog dizajna, mijenja se i kod tablice u stvarnom vremenu. Ovaj alat omogućuje developerima da dizajniraju tablice preko dizajnera, skripte ili kombinacije oba načina i donosi sve objekte povezane sa tablicom u jedan radni prostor tako da je rad na jednostavnijim projektima ubrzan.

[21] https://devblogs.microsoft.com/ssdt/what-is-sql-server-data-tools-ssdt/ (11.4.2019)

6. IZRADA APLIKACIJE

Slika 6: Prva forma aplikacije



Izvor: autorska slika

U prvoj formi programa korisnik upisuje *username* (eng. korisničko ime) i *password* (eng. lozinku) u za to predviđene kućice, nakon unosa sa pritiskom na tipku „Login“ . Ako su korisničko ime i lozinka točni korisnik se spaja sa bazom podataka i otvara se novi prozor.

6.1. Prva forma pregled koda

```
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;
using System.Data.SqlClient;
using System.IO;

namespace WindowsFormsApp1
{
    public partial class Form1 : Form
    {
        // pretpostavka da baza podataka postoji
        // postavljanje nove veze s bazom podataka
        SqlConnection con = new SqlConnection(@"Data Source = (localdb)\MSSQLLocalDB;
        Initial Catalog = puci; Integrated Security = True");
        public Form1()
        {
            InitializeComponent();
        }

private void Form1_Load(object sender, EventArgs e)
        {
            // Postavljamo prazne text boxove
            UsernameText.Text = "";
            PasswordText.Text = "";
        }

        private void Login_Click(object sender, EventArgs e)
        {
       // Provjerava dali se uneseni podaci podudaraju s podacima koji se nalaze u
       tablici
            SqlDataAdapter sda = new SqlDataAdapter("Select count(*) from maca where
            Username='" + UsernameText.Text + "'and Pass= '" + PasswordText.Text +
            "'", con);
            DataTable dt = new DataTable();
            sda.Fill(dt);
            if (dt.Rows[0][0].ToString() == "1")
            {
                this.Hide();
                Main ss = new Main();
                ss.Show();
            }
            else
            {
                // Ako se ne podudaraju,
                // prikazuje da pristup nije dopušten
                MessageBox.Show("Access Denied");
            }
        }

        private void Exit_Click(object sender, EventArgs e)
        {
            // Zatvara prozor
            this.Close();
        }
```

6.2. Postavljanje veze i TextBox-ova

```
        // postavljanje nove veze s bazom podataka
         SqlConnection con = new SqlConnection(@"Data Source = (localdb)\MSSQLLocalDB;
         Initial Catalog = puci; Integrated Security = True");
         public Form1()
         {
             InitializeComponent();
         }


private void Form1_Load(object sender, EventArgs e)
         {
             // Postavljamo prazne text boxove
             UsernameText.Text = "";
             PasswordText.Text = "";
```

U početnom djelu koda deklarira se nova SQL veza (*SqlConnection*) pod nazivom „con“ koja predstavlja vezu sa lokalnom bazom podataka „puci“. Funkcijom „Form1_Load“ se postavljaju prazni tekstovi u Username i Password poljima (*TextBox*).

6.3. Login

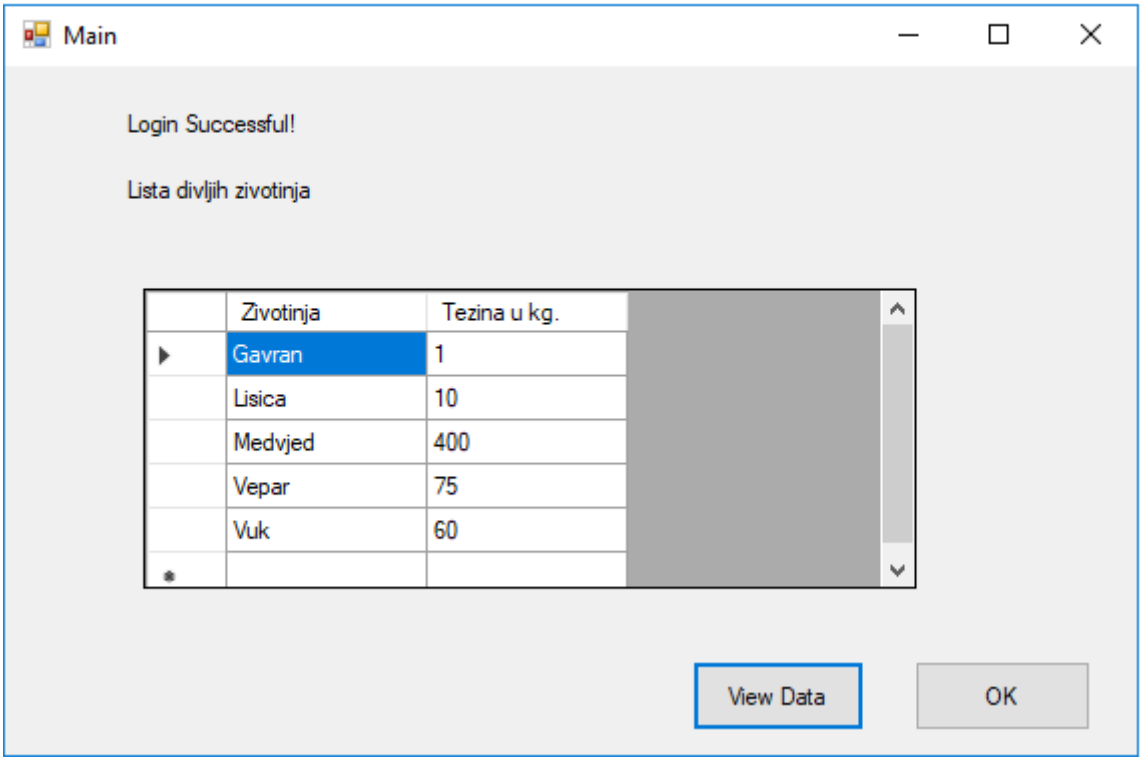
```
 private void Login_Click(object sender, EventArgs e)
 {
// Provjerava dali se uneseni podaci podudaraju s podacima koji se nalaze u
tablici
     SqlDataAdapter sda = new SqlDataAdapter("Select count(*) from maca where
     Username='" + UsernameText.Text + "'and Pass= '" + PasswordText.Text +
     "'", con);
     DataTable dt = new DataTable();
     sda.Fill(dt);
     if (dt.Rows[0][0].ToString() == "1")
     {
         this.Hide();
         Main ss = new Main();
         ss.Show();
     }
     else
     {
         // Ako se ne podudaraju,
         // prikazuje da pristup nije dopušten
         MessageBox.Show("Access Denied");
     }
 }

 private void Exit_Click(object sender, EventArgs e)
 {
     // Zatvara prozor
     this.Close();
 }
```

Funkcija Login_Click preko naredbe SqlDataAdapter sda ažurira tablicu „maca“ sa podacima unesenim u TextBox-ove UsernameText i PasswordText te nakon toga pomoću „if“ petlje provjerava dali se podudaraju novi podaci s podacima koji su se već nalazili u tablici. Ako se podudaraju prozor Form1 se skriva i otvara se prozor Main, u protivnom prikazuje sa MessageBox koji obavještava korisnika da mu pristup nije dopušten.
Funkcija Exit_Click zatvara prozor preko naredbe this.Close();

6.4. Druga forma

Slika 7: Druga forma aplikacije



Izvor: autorska slika

U drugoj formi programa otvara se nova veza sa bazom podataka u kojoj korisnik pritiskom na tipku „*View Data*“ može pogledati sadržaj tablice pohranjene na našoj bazi podataka. A pritiskom na tipku „OK“ zatvara se prozor.

6.5. Druga forma pregled koda

```
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;
using System.Data.SqlClient;

namespace WindowsFormsApp1
{
    public partial class Main : Form
    {
        // postavljanje nove veze s bazom podataka
        SqlConnection con = new SqlConnection(@"Data Source = (localdb)\MSSQLLocalDB;
        Initial Catalog = puci; Integrated Security=True");
        public Main()
        {
            InitializeComponent();
        }

        private void Main_Load(object sender, EventArgs e)
        {

        }

        private void exitButton_Click(object sender, EventArgs e)
        {
            this.Close();
        }

        private void showData_Click(object sender, EventArgs e)
        {
            //  otvaranje veze s bazom podataka
            con.Open();
            // postavljanje SQL upita
            String upit = "SELECT ID_Animal, Animal_Name, Animal_Pop FROM SecretInfo";
            // stvaranje objekta tipa SqlDataAdapter uz parametre upit i con
            SqlDataAdapter SDA = new SqlDataAdapter(upit, con);
            // stvaranje objekta tipa DataTable
            DataTable DT = new DataTable();
            // popunjavanje tablice sa traženim podacima
            SDA.Fill(DT);
            // prikaz podataka preko Data Grid View objekta
            PregledPodataka.DataSource = DT;
            // zatvaranje veze
            con.Close();
        }
    }
}
```

6.6. Analiza koda druge forme

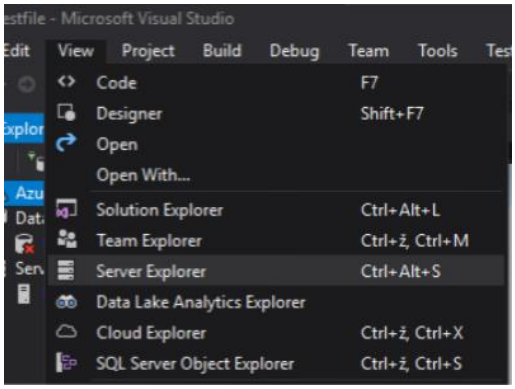
Kada se otvori prozor druge forme deklarira se nova SQL (SqlConnection) veza pod nazivom con koja postavlja vezu sa lokalnom bazom podataka „puci“. Klikom na tipku exitButton zatvara se prozor. Klikom na tipku showData otvata se veza sa bazom podataka (con.Open();) nakon toga postavlja se SQL upit pod nazivom „upit“ s kojim tražimo da nam se označe podaci iz tablice „SecretInfo“.

Kada su željeni podaci označeni stvara se novi objekt tipa SqlDataAdapter sa parametrima „upit“ i „con“. Stvara se nova tablica odnosno objekt tipa DataTable i naredbom „SDA.Fill(DT);“ tablica se popunjava sa prethodno označenim podacima iz tablice „SecretInfo“. Te na kraju prikazuju se podaci iz nove tablice korisniku preko objekta DataGridView naredbom „PregledPodataka.DataSource = DT;“ , nakon što su podaci prikazani zatvara se veza sa bazom podataka (con.Close();).

6.7. Izrada baze podataka

Kreiranje novih i spajanje na već postojeće baze podataka u Visual Studiju obavlja se preko Server Explorera. Otvara se klikom na alatnoj traci na „*View*“ te nakon toga na Server Explorer.

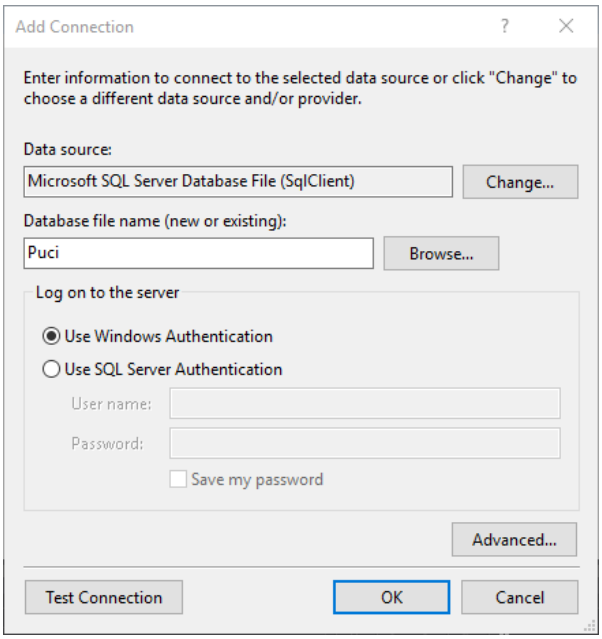
Slika 8: Otvaranje Server Explorera



Izvor: autorska slika

U Server Explorer-u desnim klikom na „Data Connections“ otvara se izbornik u kojem upravljamo vezama s bazama podataka, u izborniku kliknemo na „Add Connection“ čime se otvara novi prozor u kojemu se možemo povezati sa već postojećom bazom podataka ili možemo stvoriti novu. Za potrebe ove vježbe stvorena je nova baza podataka pod nazivom „Puci“. Kada se definira ime klikom na „OK“ baza podataka je kreirana.

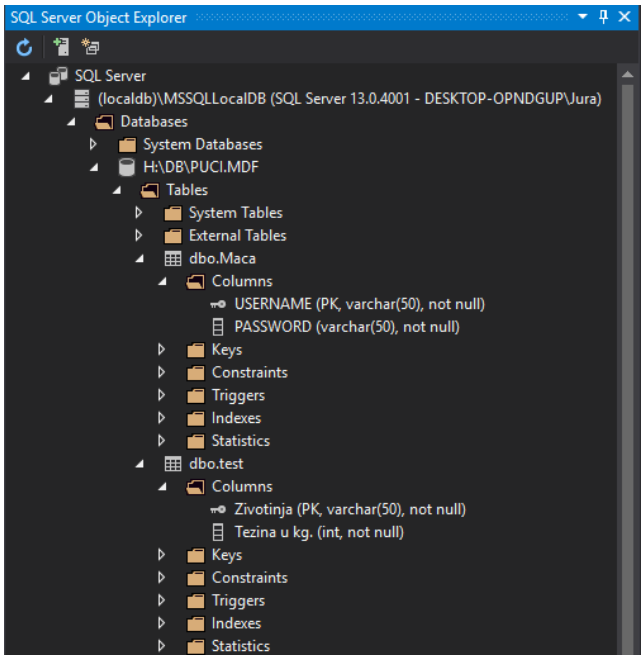
Slika 9: Dodavanje veze u Server Exploreru



Izvor: autorska slika

Budući da je novo kreirana baza podataka prazna a potrebna je tablica u kojoj će biti pohranjeno korisničko ime i lozinka. Tablica se kreira tako da se otvori „*SQL Server Object Explorer*“ te unutar odabrane baze podataka otvara se mapa „*Tables*“ te desnim klikom otvara se izbornik i odabire opcija „*Create New Table*“, u ovom slučaju tablica pod nazivom „Maca“.

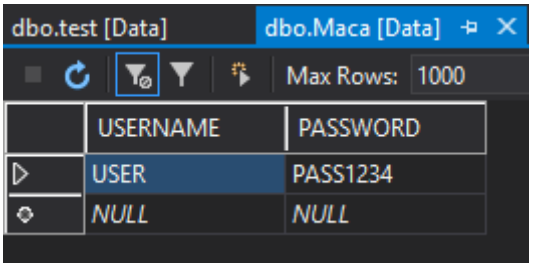
Slika 10: SQL Server Object Explorer



Izvor: autorska slika

U tablici postavljaju se stupci „USERNAME“ i „PASSWORD“. Kada tablica ima postavljene stupce potrebno ju je popuniti podacima. Desnim klikom na tablicu otvara se izbornik i odabirom na opciju „*View Data*“ otvara se grafički prikaz sadržaja tablice u glavnom prozoru i u nju se mogu unositi novi podaci ili izmijeniti stari. U ovoj vježbi postavljeno je korisničko ime (USERNAME) „USER“ a lozinka (PASSWORD) „PASS1234“.

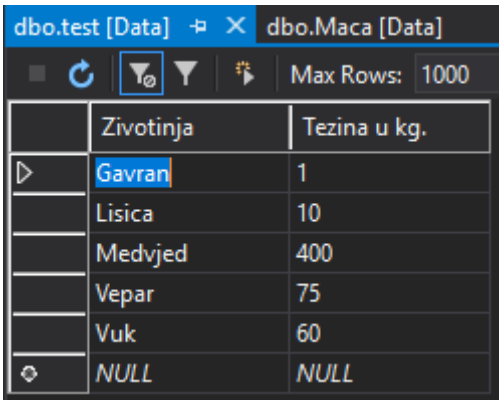
Slika 11: Tablica Maca



Izvor: autorska slika

Istim postupkom kreirana je još jedna tablica koja predstavlja podatke koji su jedino dostupni ako korisnik unese ispravno korisničko ime i lozinku. U ovom slučaju za primjer je postavljena lista divljih životinja i njihova težina. Te ogledna tablica izgleda ovako.

Slika 12: Tablica test



Izvor: autorska slika

7. ZAKLJUČAK

U teoretskom djelu rada vidimo kako se C# razvio od jako osnovnog objektno orijentiranog programskog jezika do stvarno moćnog jezika koji omogućava brzi razvoj aplikacija i pisanje jasnog, kratkog i efikasnog koda.

Također razvojem baza podataka i sa njima povezanih jezika kao što je SQL te razvojem Visual Studio integriranog razvojnog sučelja i njegovih alata za rad sa bazama podataka (*SQL Server Dana Tools*) moguće je raditi aplikaciju u C#-u i vrlo jednostavno ju spojiti s postojećom bazom podataka ili stvoriti vlastitu jednostavnu bazu podataka.

U praktičnom djelu korišten je program Visual Studio. Ova aplikacija predstavlja korisni koncept prijave putem korisničkog imena i lozinke koji je jako široko primjenjiv.

Glavna prednost ovog projekt je što je u cijelosti napravljen u Visual Studiu te ističe koliko je pisanje različitih dijelova koda u različitim programskim jezicima olakšano u modernim razvojnim sučeljima.

LITERATURA

1. Faraz Rasheed, Programmers Heaven: C# School First Edition
2. Frane Urem, Uvod u objektno orijentirano programiranje s primjerima, Šibenik
3. https://www.cnet.com (16.5.2018)
4. https://blog.ndepend.com(18.5.2018)
5. https://tutorials.visualstudio.com (4.12.2018)
6. https://visualstudiomagazine.com (6.12.2018)
7. https://docs.microsoft.com (6.12.2018)
8. https://searchsqlserver.techtarget.com(3.3.2019)
9. http://www.computerbusinessresearch.com (4.3.2019)
10. https://www.tutorialspoint.com (17.3.2019)
11. https://devblogs.microsoft.com (11.4.2019)