**SVEUČILIŠTE U SPLITU**

**SVEUČILIŠNI ODJEL ZA STRUČNE STUDIJE**

Preddiplomski stručni studij Informacijske tehnologije

**JOSIP PAVIĆ**

**Z A V R Š N I R A D**

**IZRADA DESKTOP I WEB APLIKACIJE KORIŠTENJEM C#**

Split, kolovoz 2019.

**SVEUČILIŠTE U SPLITU**

**SVEUČILIŠNI ODJEL ZA STRUČNE STUDIJE**

Preddiplomski stručni studij Informacijske tehnologije

**JOSIP PAVIĆ**

**Z A V R Š N I R A D**

**IZRADA DESKTOP I WEB APLIKACIJE KORIŠTENJEM C#**

Split, kolovoz 2019.

**SVEUČILIŠTE U SPLITU**

**SVEUČILIŠNI ODJEL ZA STRUČNE STUDIJE**

Preddiplomski stručni studij Informacijske tehnologije

**Predmet**: Programiranje u C#

**Z A V R Š N I R A D**

**Kandidat:** Josip Pavić

**Naslov** **rada:** Izrada *desktop* i *web* aplikacije korištenjem C#

**Mentor:** Marina Rodić, predavač

Split, kolovoz 2019.

**SADRŽAJ**

[SAŽETAK 1](#_Toc17067256)

[SUMMARY 2](#_Toc17067257)

[1. UVOD 3](#_Toc17067258)

[2. TEHNOLOGIJE 4](#_Toc17067259)

[2.1. ASP .NET MVC 4](#_Toc17067260)

[2.2. Razvojno okruženje Microsoft Visual Studio Community 4](#_Toc17067261)

[2.3. SQL Server Management Studio 5](#_Toc17067262)

[2.4. Entity Framework 6 7](#_Toc17067263)

[2.5. Language Integrated Query (LINQ) 8](#_Toc17067264)

[2.6. JavaScript Object Notation (JSON) 9](#_Toc17067265)

[2.7. Windows forme 9](#_Toc17067266)

[3. APLIKACIJA 10](#_Toc17067267)

[3.1. Baza podataka 10](#_Toc17067268)

[3.2. Desktop aplikacija 12](#_Toc17067269)

[**3.2.1. Struktura *desktop* aplikacije** 12](#_Toc17067270)

[**3.2.2. Forma za prijavu** 13](#_Toc17067271)

[**3.2.3. Forma PoslužiteljKlijent** 14](#_Toc17067272)

[**3.2.4. Forma Poslužitelj** 15](#_Toc17067273)

[**3.2.5. Forma Klijent** 19](#_Toc17067274)

[**3.2.6. Forma Lista Poslužitelja** 21](#_Toc17067275)

[**3.2.7. Forma ListaIgrača** 22](#_Toc17067276)

[**3.2.8. Forma Vrsta Igre** 23](#_Toc17067277)

[**3.2.9. Forma IgraTri** 23](#_Toc17067278)

[**3.2.10. Forma IgraČetiri** 24](#_Toc17067279)

[**3.2.11. Višekorisnička** **logika** 26](#_Toc17067280)

[**3.2.12. Forma PravilaIgre** 27](#_Toc17067281)

[**3.2.13. Forma IgraGotova** 27](#_Toc17067282)

[**3.2.14. Klasa Špil** 27](#_Toc17067283)

[**3.2.15. Klasa Igrač** 28](#_Toc17067284)

[3.3. Web aplikacija 29](#_Toc17067285)

[**3.3.1. Struktura *web* aplikacije** 29](#_Toc17067286)

[**3.3.2. Prava pristupa** 29](#_Toc17067287)

[**3.3.3. Forma za registraciju** 30](#_Toc17067288)

[**3.3.4**. **Forma za prijavu** 31](#_Toc17067289)

[**3.3.5**. **Prikaz Moji rezultati** 32](#_Toc17067290)

[**3.3.6**. **Account Manager prikaz** 32](#_Toc17067291)

[**3.3.7**. **Forma Lista igrača** 33](#_Toc17067292)

[**3.3.8**. **Prikaz Najbolji igrači** 33](#_Toc17067293)

[4. ZAKLJUČAK 35](#_Toc17067294)

[5. LITERATURA 36](#_Toc17067295)

[6. POPIS SLIKA I ISPISA 38](#_Toc17067296)

# SAŽETAK

Projekt se sastoji od *desktop* aplikacije i *web* aplikacije. *Desktop* i *web* aplikacija koriste istu bazu podataka. Baza podataka izrađena je u Microsoft SQL Management Studiju 2017 te je postavljena na Microsoft Azure SQL server. *Desktop* aplikacija omogućuje korisnicima međusobno nadmetanje na računalima povezanima u lokalnu mrežu, dok se za igru preko globalne mreže korisnik treba sam pobrinuti. *Desktop* aplikacija omogućuje korisniku da kreira poslužitelja ili da pristupi nekom od postojećih poslužitelja kao klijent. Sama igra odvija se u *desktop* aplikaciji. Igra je realizirana tako da omogućuje sudjelovanje dva korisnika, a ishod igre pohranjuje se u bazu podataka. Poslužitelj može kreirati igru s tri ili četiri karte. Aplikacija je sinkronizirana s bazom podataka, pa korisnik može vidjeti listu igrača koji su trenutno prijavljeni u aplikaciji. Osim toga, korisnik može vidjeti i sve poslužitelje koji su trenutno pokrenuti. *Web* aplikacija omogućuje administratoru jednostavnije upravljanje i lakši pristup bazi podataka jer nije potrebno imati instaliranu *desktop* aplikaciju. Korisniku je također omogućen lakši pregled i uređivanje profila putem *web* aplikacije.

**Ključne riječi:** C #, *desktop* aplikacija, *web* aplikacija

# SUMMARY

**Creating desktop and web application using C #**

The project consists of desktop application and web application. Desktop application and web application use the same database. This database is made in Microsoft SQL Studio 2017 programme and is set on the Microsoft Azure SQL server. Desktop application enables it's users to compete between each other on the computers connected to the local network, while for playing the game on the global network user need to make a setup on his router. Desktop application enables the user to create the server or to have an access to the already existing one. The game itself is held in the desktop application. Is is created in a way that two players play against each other and the final result is stored in a database. The server can create a game type with three or four cards.The application is synchronized with the database what enables the user to have insight into the list of all active servers and online players at that time. Web application helps the administrator to have an easier management and access to the database data, since it is not necessary to have the desktop application installed on your personal computer. The user also has easier view and can more easily edit his profile via web application.

**Keywords:** C #, desktop application, web application

# 1. UVOD

Cilj ovog završnog rada ja izrada aplikacije koja će služiti za zabavu korisnika. U tu svrhu izrađene su *desktop* i *web* aplikacija. Tema projekta je višekorisnička kartaška igra briškula.

Za razvoj *desktop* aplikacija korištene su Windows forme, dok je *web* aplikacija razvijana u MVC arhitekturi. Programski jezik korišten za izradu *desktop* i *web* aplikacije je C#, dok je za razvojno okruženje korišten Microsoft Visual Studio 2017. Kroz *desktop* aplikaciju opisana je izrada aplikacije u kojoj je implementirana višekorisnička logika. *Desktop* aplikaciju je moguće instalirati na računala s operativnim sustavom Windows. Za korištenje *desktop* i *web* aplikacije potrebno je imati pristup internetu.

Završni rad sadrži šest poglavlja. Prvo je poglavlje rada uvod u kojem se analiziraju tema rada, njegov cilj i struktura. U drugom poglavlju završnog rada analizirane su tehnologije koje su se koristile pri izradi *desktop* i *web* aplikacije. U trećem poglavlju rada analiziran je model podataka te su prikazane relacije između entiteta. Kroz primjere i slike prikazan je razvoj *desktop* i *web* aplikacije. Četvrto poglavlje rada jest zaključak u kojem se sažeto iznosi sve što je istaknuto u radu te se donose određene spoznaje o korištenim tehnologijama tijekom izrade aplikacije i mogućnosti razvoja aplikacije u budućnosti. Peto poglavlje završnog rada sadrži popis literature korištene prilikom izradi rada. Šesto poglavlje rada sadrži popis slika uvrštenih u završni rad.

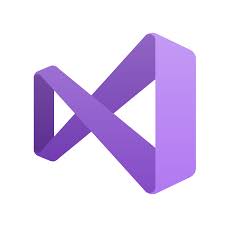
# 2. TEHNOLOGIJE

## 2.1. ASP .NET MVC

Tehnologija Microsoft .NET okvira jest tehnologija koja se koristi za razvoj dinamičkih i interaktivnih *web* stranica te za razvoj *web* servisa. ASP stranice izvršavaju se na poslužiteljskoj strani te generiraju kôd u HTML, XML formatu koji se šalje *desktop* ili mobilnim preglednicima. MVC kôdna arhitektura, koja je iako razvijana prvenstveno za *desktop* aplikacije danas je popularnija kod izrada *web* aplikacija [1]. Osnovna ideja MVC arhitekture jest razdvajanje kôda na tri cjeline: *Model*, Pogled (engl. *view*) i Kontroler (engl. *controller*) [2]. MVC je razvojni okvir koji se koristi za izradu *web* aplikacija otvorenoga kôda. Arhitektura MVC sastoji se od tri međusobno zavisne komponente, a to su model, pogled i upravitelj. U sloju modela najčešće je implementirana poslovna logika, točnije to je skup klasa koje opisuju podatke poslovne logike s kojom se radi. Model enkapsulira podatke koji se čuvaju u bazi podataka. Pogled omogućuje prikaz podataka te se najčešće koristi za prikaz podataka iz modela. Upravitelj je, pak, veza između modela i pogleda. Čita ulazne podatke od korisnika te ih prosljeđuje modelu. Nakon komunikacije s modelom upravitelj odlučuje koji će se pogled prikazati krajnjem korisniku. MVC za čuvanje podataka koristi metode koje se nalaze u kontroloru.

## 2.2. Razvojno okruženje Microsoft Visual Studio Community

Razvojno okruženje Visual Studio je programsko razvojno okruženje (engl. *Integrated* *Development* *Environment*), služi za pisanje raznih vrsta aplikacija za operativni sustav Windows [4].



**Slika 1:** Visual Studio logo

Tvorac razvojnoga okruženja Visual Studio je tvrtka Microsoft. Razvojno okruženje Visual Studio trenutno dolazi u tri varijeteta:

* Visual Studio Community
* Visual Studio Professional
* Visual Studio Enterprise

Visual Studio Community je moguće besplatno preuzeti na službenim stranicama Microsoft kompanije. Instalacija razvojnog okruženja Visual Studio Community zahtijeva posjedovanje besplatnoga Microsoft naloga, te da bi se instalacija uspješno izvršila potrebna je stalna internet veza. Visual Studio omogućava integraciju različitih tehnologija koje ne moraju biti razvijene od strane Microsoft kompanije [4]. Razvojno okruženje Visual Studio je jednostavno za korištenje. Pri pisanju kôda nudi se završetak riječi što značajno olakšava posao prilikom izrade aplikacije.

## 2.3. SQL Server Management Studio

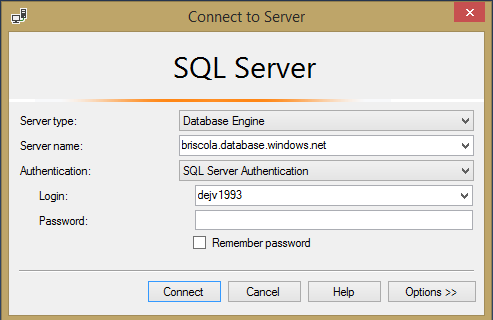
SQL Server Management Studio je integrirano razvojno okruženje za upravljanje SQL poslužiteljem (engl. *server*) . SQL Server Management Studio je moguće besplatno preuzeti sa službenih stranica Microsoft kompanije.



**Slika2:** SQL Server Management Studio logo

Uz pomoć SQL Server Management Studia administrator može upravljati sa skoro svim aspektima SQL poslužitelja. SQL Server Menagement Studio omogućava kreiranje i upravljanje bazom podataka, prilagođavanje sigurnosti, izradu korisničkih naloga za pristup SQL poslužitelju [5]. Kreiranjem baze podataka u SQL Server Management Studiju kreiraju se minimalno dvije datoteke. Prva datoteka s ekstenzijom „ .MDF“ sadrži dodatke i objekte, dok datoteka s ekstenzijom „ .LDF“ predstavlja dnevnik transakcija [6]. Na slici 3. prikazana je forma za prijavu na SQL poslužitelj.

Pokretanjem Management Studia otvara se forma za prijavu na SQL poslužitelj.



**Slika 3:** Forma za prijavu na SQL poslužitelj

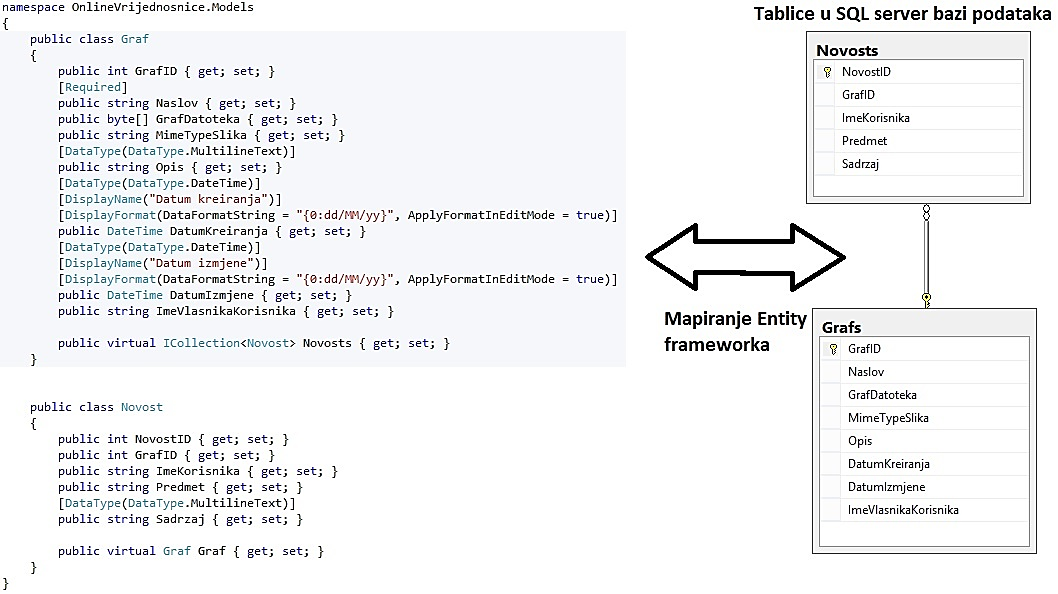
Komponente forme za prijavu na SQL poslužitelj:

* Vrsta poslužitelja (engl. *Server Type*) – odabiremo poslužitelj na koji se želimo povezati
* Ime poslužitelja (engl. *Server Name*) – ime poslužitelja na koji se želimo povezati
* Autentifikacija (engl. *Authetication*) - odabire se vrsta naloga za prijavu na SQL poslužitelj
* Prijava (engl. *Login*) – unos u ovo polje je omogućen samo ako je kao način prijave izabran SQL poslužitelj nalog
* Lozinka (engl. *Password*) - unos u ovo polje je omogućeno samo ako je kao način prijave izabran SQL poslužitelj nalog [7]

## 2.4. Entity Framework 6

Razvojni okvir Entity (engl. *Entity Framework*) 6 je okvir objektno relacijskog mapiranja otvorenog kôda za ADO.NET. Objektno relacijsko mapiranje značajno olakšava razvoj podatkovno orijentiranih aplikacija. Od verzije 6 razvojni okvir Entity je odvojen od .NET okvira [8].

Razvojni okvir Entity omogućuje programerima da umjesto s tabličnim podatcima rade s podatcima u obliku objekta. Isto tako, razvojni okvir Entity omogućuje jednostavnu sinkronizaciju modela s bazom podataka. Razvojni okvir Entity može automatski generirati veliki dio kôda, čime se programeru značajno štedi vrijeme. Prijenos podataka između kôda C# programskog jezika i SQL baze podataka se naziva mapiranje (engl. *mapping*) . Razvojni okvir entity mapira relacijske tablice u objekte koje C# razumije [9]. Na slici 4. prikazano je mapiranje u razvojnom okviru Entity.



**Slika 4:** Mapiranje u razvojnom okviru Entity

Postoji više načina da se napravi objektno relacijsko mapiranje, odnosno postoje tri različita razvojna tijeka rada (engl. *development* *workflow*) :

* Baza prva – generira sve potrebne objekte u kôdu na temelju modela baze podataka.
* Kôd prvi – generira sve potrebne bazne objekte na osnovi postojećeg kôda. Omogućuje kreiranje klasa bez grafičkog korisničkog sučelja (engl. *Graphical* *User* *Interface*) ili .edmx fajla. Taj se pristup preporučuje koristiti ako ne postoji baza podataka.
* Model prvi – definira entitete i modele na temelju kojih će se generirati baza podataka i klase [4].

Kao most (engl. *bridge*) između *entity* klasa i baze podataka koristi se kontekstna klasa (engl. *dbcontext*) . Izgled kontekstne klase prikazan je na ispisu 1.

public partial class PlayersEntities1 : DbContext{

public virtual DbSet<Game> Games { get; set; }

public virtual DbSet<Player> Players { get; set; }}

public virtual DbSet<PlayerRole> PlayerRoles { get; set; }

public virtual DbSet<Role> Roles { get; set; }}

Ispis 1: Kontekstna klasa

## 2.5. Language Integrated Query (LINQ)

Language Integrated Query (skraćenica LINQ) je prvi put predstavljen u .NET 3.5 verziji te u Visual Studio 2008. LINQ unificira način pristupa i pretrage podataka. Omogućuje pristupanje podatcima različitog tipa korištenjem iste sitakste. Uvođenje LINQ-a omogućuje jednostavnije pretraživanja podataka u bazi, čime se smanjuje količina potrebnoga kôda te povećava čitljivost kôda [10]. LINQ sadrži oko pedeset upita operatora pomoću kojih se značajno smanjuje vrijeme sortiranja, filtriranja te grupiranja podataka. Neki od najčešće korištenih operatora jesu sljedeći:

* sortiranje
* filtriranje
* grupiranje
* spajanje
* pretvorba.

Operatori koji se koriste u metodama standardnih upita nazivaju se izrazi lambda (engl. *Lambda expressions*). Ti izrazi omogućavaju da se operatori upita definiraju kao metode te povezuju korištenjem točka oznake (engl. *dot notation*) . LINQ se može koristiti za sve vrste podataka koje su izvedeni iz IEnumreable sučelja [10].

## 2.6. JavaScript Object Notation (JSON)

JSON (engl. *JavaScript Object Notation*) je tekstualni format čija je namjena prijenos podataka u formatu koji je čitljiv i ljudima i strojevima [11]. Ekstenzija koja se koristi za JSON datoteku je oblika „.json“ .

Zbog svojih prednosti nad XML-om JSON sve češće postaje prvi izbor. XML koristi oznake, pa je teži za pisanje i čitanje. Prednost JSON-a je što za parsiranje koristi js parser. Objekt u JSON formatu je oblika ključ: par, a nalazi se unutar vitičastih zagrada [11].

JSON ne ovisi o programskom jeziku, a njegova je najveća primjena u *web* aplikacijama.

## 2.7. Windows forme

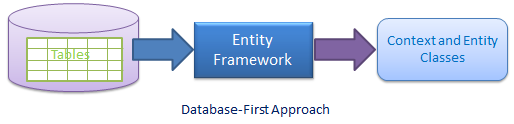
Windows forma je alat koji se koristi za izradu Windows aplikacija. Te su forme učinkovit i jednostavan način komunikacije koja se ostvaruje između korisnika i programa. Window forma je prozor koji sadrži kontrole za prikaz, unos i manipulaciju podatcima [12]. Za izradu Windows formi potrebno je unutar Visual Studia odabrati *File->New* projekt te u izborniku odabrati Windows formu. Upiše se željeno ime te se pritiskom tipke *OK* kreira nova Windows forma.

Alati (engl. *Toolbox*) sadrže skup gotovih Windows kontrola [12]. Osim toga, alati sadrže kontrole koje se mogu dodati Windows formama, a prikazane su samo one kontrole koje se mogu koristiti za trenutni dizajn. Dizajn prikazuje obrazac forme sa svim elementima koji se nalaze u formi. Svojstva (engl. *Properties*) omogućuju da se postavke podese za željeni element. Dijele se na devet kategorija, a to su izgled (engl. *appearance*), ponašanje (engl. *behavior*), podatci (engl. *Data*), pristupačnost (engl. *accessibility*), dizajn (engl. *design*), fokus (engl. *focus*), predložak (engl. *layout*), stil prozora (engl. *windows* *style*) te različito (engl. *Misc*). Istraživač rješenja (engl. *Solution* *Explorer*) koristi se za prikaz strukture projekta [12].

# 3. APLIKACIJA

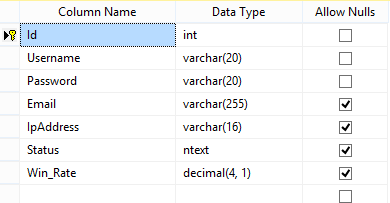
## 3.1. Baza podataka

Baza podataka izrađena je pomoću Microsoft SQL Server Management Studio 2017 alata. Baza podataka koristi se za pohranjivanje podataka registriranih korisnika te za pohranjivanje ishoda igre. Na temelju izrađene baze razvojni okvir Entity automatski generira potrebne modele. Na slici 5. prikazan je baza prva pristup.

[](https://www.entityframeworktutorial.net/images/EF5/databasefirst.png)

Slika 5: Baza prva pristup

Nakon izrade baze podataka slijedi izrada pojedinih tablica. Primjer izrađene tablice prikazan je na slici 6.



**Slika 6:** Tablica Korisnik

Na slici 7. prikazan je *EER* model baze podataka koja sadrži tablice s odgovarajućim relacijama između pojedinih tablica. Relacije između tablica važne su za lakšu manipulaciju samim podatcima te pravilan rad baze podataka.

****

Slika 7: Relacijski model baze podataka

Za manipulaciju podatcima koji se nalaze u bazi podataka treba povezati aplikaciju s bazom. Kontekstna klasa upravlja vezom s bazom te se po potrebi spaja ili odspaja s baze. PlayersEntities1 klasa izvedena je iz kontekstne klase. Konstruktor (engl. *constructor*) PlayersEntities1 klase kao parametar može primiti ime baze na koju se spaja ili *string* za vezu s bazom. PlayersEntities1 konstruktor prikazan je na ispisu 2.

public partial class PlayersEntities1 : DbContext{

public PlayersEntities1(): base("name=PlayersEntities1")

{}}

Ispis 2: PlayersEntities1 konstruktor

Za ostvarivanje veze s Microsoft Azure SQL poslužiteljem treba podesiti *string* za vezu s bazom. *String* za vezu s bazom kreira se automatski te je podešen za vezu s bazom podataka koja se nalazi na lokalnom računalu. Na ispisu 3. prikazan je kôd ispravno podešenog *stringa* za povezivanje s Microsoft Azure SQL poslužiteljem.

metadata=res://\*/PlayModel.csdl|res://\*/PlayModel.ssdl|res://\*/PlayModel.msl;

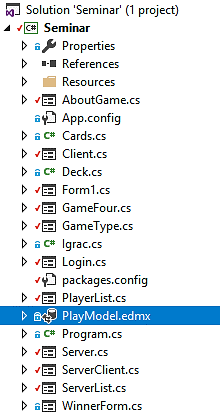
provider=System.Data.SqlClient;provider connection string="data source=briscola.database.windows.net;initial catalog=Players;user id=željeno\_korisničko ime;password=željena\_šifra;MultipleActiveResultSets=True;App=EntityFramework"

Ispis 3: *String* za konekciju s Microsoft Azure SQL poslužiteljem

## 3.2. Desktop aplikacija

### **3.2.1. Struktura *desktop* aplikacije**

Struktura *desktop* aplikacije prikazana je na slici 8.

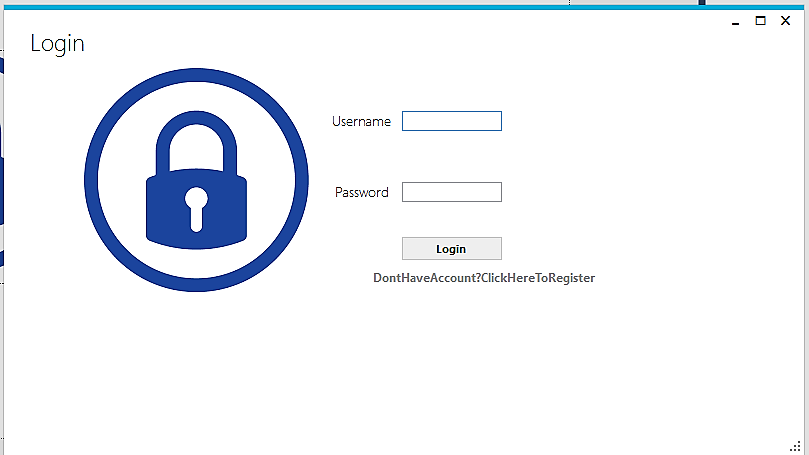


Slika 8: Struktura *desktop* aplikacije

### **3.2.2. Forma za prijavu**

Pokretanjem aplikacije prikazuje se forma za prijavu koja omogućuje prijavu za postojeće korisnike. Forma za prijavu sastoji se od dva tekstualna polja (engl. *textbox*), dugmeta (engl. *button*) *Login* te poveznice (engl. *link*). Klik na poveznicu za registraciju odvodi korisnika na *web* aplikaciju Briškula.

Nakon uspješnog logiranja forma prelazi u pozadinu. Korisnik je usmjeren na glavni izbornik, odnosno otvara se forma PoslužiteljKlijent. Status korisnika mijenja se iz *offline* u o*nline*. Forma za prijavu prikazana je na slici 9.

.

Slika 9: Forma za prijavu

### **3.2.3. Forma PoslužiteljKlijent**

Forma PoslužiteljKlijent sadrži glavni izbornik aplikacije. Izgled izbornika ovisi o pravima pristupa pojedinog korisnika. Postoje dvije razine prava pristupa aplikaciji, a to su administratorski i korisnički pristup. Kod korisničkog pristupa formi PoslužiteljKlijent izbornik sadrži sljedeće opcije:

* dugme Poslužitelj
* dugme Vrsta Igre
* dugme Lista Poslužitelja
* dugme Pravila Igre

Kod administratorskog pristupa izbornik forme PoslužiteljKlijent sadrži sve opcije koje sadrži korisnički pristup uz dodatne mogućnosti:

* dugme lista igrača – klikom miša pokreće se forma lista igrača
* dugme resetiranje baze – klikom miša na to dugme resetiraju se vrijednosti u listi „igra“ , tj. uklanjaju se sve igre iz baze podataka i svi trenutno pokrenuti poslužitelji.

Klikom miša na neku ponuđenu opciju otvara se odabrana forma. Forma PoslužiteljKlijent prelazi u pozadinu te njezine kontrole više nisu vidljive korisniku sve dok se trenutna forma ne zatvori. Na slici 11. prikazana je forma PoslužiteljKlijent s administratorskim pravima.

****

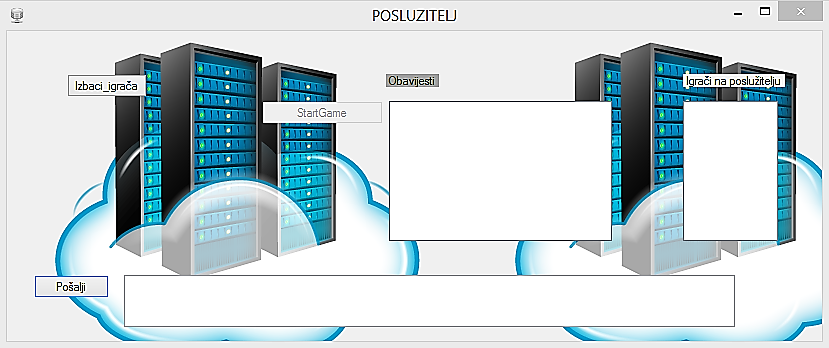
Slika 11: Forma PoslužiteljKlijent s administratorskim pravima

PoslužiteljKlijent forma posjeduje izborničku traku (engl. *menu bar*) koji sadrži opcije za sljedeće:

* Moja statistika – prikazuje se statistika trenutno prijavljenoga korisnika.
* Najbolji igrači – prikazuje listu pet najboljih korisnika sortiranih po omjeru pobjeda/poraza te po broju odigranih igara. Ako u bazi podataka ne postoji toliko korisničkih računa, prikazat će se svi korisnici.
* Briškula Web – klikom miša otvara se početna stranica *web* aplikacije „Briškula“ koja je detaljnije analizirana u poglavlju 3.3.

### **3.2.4. Forma Poslužitelj**

Forma Poslužitelj prikazana je na slici 12.



Slika 12: Forma Poslužitelj

Pokretanjem forme Poslužitelj pokreće se poslužitelj. Nakon pokretanja poslužitelj na priključku (engl. *portu*) 7000 sluša zahtjeve klijenta za spajanje. Lista „Igrači na poslužitelju“ sadrži popis korisnika koji su trenutno spojeni na poslužitelj.

Klikom na dugme *Izbaci igrača* klijent se preusmjerava na formu ListaIgrača te se uklanja iz liste „Igrači na poslužitelju“. Na ispisu 7. prikazan je kôd koji se će izvršiti u formi Poslužitelj kako bi se klijent uspješno odspojio s poslužitelja.

OnFormUnsub("Form");

tcpServer.BroadcastLine("kicked");

Client\_Disconnect(null, tcpClient);

Ispis 7: Kôd za uklanjanje klijenta sa poslužitelja

Kako bi se omogućila komunikacija s klijentom obavlja se pretplata (engl. *subscribe*) na događaj (engl. *event*) DataReceived iz SimpleTcpServer klase. Klikom na dugme *Pošalji* šalje se poruka svim klijentima spojenim na poslužitelj. Poruka sadrži tekst koji se nalazi u tekstualnom polju. U listi „Obavijesti“ nalaze se sve poruke poslane od strane klijenta.

Kôd za pretplatu na događaje SimpleTcpServer klase prikazan je na ispisu 4.

tcpServer = new SimpleTcpServer();

tcpServer.Delimiter = 0x13;

tcpServer.StringEncoder = Encoding.ASCII;

tcpServer.DataReceived += Server\_Recived;

tcpServer.ClientConnected += Client\_Connected;

tcpServer.ClientDisconnected += Client\_Disconnect;

Ispis 4: Kôd za pretplatu na događaje SimpleTcpServer klase

U prikazanom kôdu na ispisu 4. vrši se pretplata na događaje *ClientConnected* i *ClientDisconnected* SimpleTcpServer klase. Kako bi poslužitelj bio obaviješten o spajanju klijenta vrši se pretplata na događaj ClientConnected. Za obavijest o odspajanju klijenta s poslužitelja vrši se pretplata na događaj ClientDisconnected.

Prilikom pokretanja igre na poslužiteljevoj strani izvršava se serijalizacija (engl. *serialize*) objekta tipa IgraTri ili IgraČetiri, ovisno o vrsti igre. Serijalizirani objekt poslužitelj šalje svim klijentima na mreži.

Na ispisu 5. prikazan je kôd JSON serijalizacija.

json = JsonConvert.SerializeObject(f);

tcpServer.BroadcastLine(json);

Ispis 5: Kôd JSON serijalizacija

Kako bi dobio obavijest o završetku igre poslužitelj se pretplaćuje na događaj GameEnded forme IgraTri ili IgraČetiri, ovisno o vrsti igre koja je pokrenuta. Na ispisu 6. prikazan je kôd  pretplata na događaje.

f.MouseClicked += this.OnMOuseClicked;

f.GameEnded += this.OnGameEnded;

this.MoveMade += f.OnMoveMade;

f.FormUnsub += this.OnFormUnsub;

f.FormExiting += this.OnFormClosing;

tcpServer.DataReceived += f.GameRecived;

Ispis 6: Kôd pretplata na događaje

### **3.2.5. Forma Klijent**

Forma Klijent prikazana je na slici 13.



Slika 13: Forma Klijent

Nakon uspješnoga povezivanja s poslužiteljem otvara se forma Klijent. Klijent može napustiti poslužitelja pritiskom na dugme *Return*. Klikom na to dugme zatvara se forma Klijent te iz pozadine na površinu prelazi forma Lista Servera. Kako bi komunikacija s poslužiteljem bila moguća klijent se mora pretplatiti na događaj iz SimpleTcpClient klase. Kôd pretplata na DataReceived događaj prikazan je na ispisu 8.

|  |
| --- |
| client=new SimpleTcpClient();  client.StringEncoder = Encoding.UTF8;  client.DataReceived += Client\_Received; |

Ispis 8: Kôd pretplata na DataReceived događaj

Klikom na dugme *Send* klijent šalje poruku poslužitelju, a poruka sadrži tekst koji se nalazi u tekstualnom polju. U listi „poruke“ nalaze se poruke razmijenjene s poslužiteljem. Klikom na dugme *Ready* klijent omogućuje poslužitelju da pokrene igru.

U formi Klijent obavlja se deserijalizacija *stringa* u objekt tipa IgraTri ili IgraČetiri. Kôd deserijalizacija objekta prikazan je na ispisu 9.

|  |
| --- |
| BeginInvoke((MethodInvoker) {  string s = start.Substring(0, start.Length - 1);  try{  if (gameType == 3){  f = JsonConvert.DeserializeObject<Form1>(s);}  else {  g = JsonConvert.DeserializeObject<GameFour;  Thread.Sleep(1);}  catch (Exception e){  MessageBox.Show(e.Message.ToString();}} |

Ispis 9: Kôd deserijalizacija objekta

Deserijalizaciju je nužno izvršiti unutar *BeginInvoke (MethodInvoker*) funkcije, u suprotnom dolazi do greške tijekom izvođenja (engl. *runtime* *error*).

Po završetku igre poziva se metoda OnFormUnsub. U metodi se vrši odjava s pretplaćenih događaja. Bez odjave s pretplate pri svakom pokretanju nove igre dolazilo bi da povećanja pretplate na isti događaj. Kôd za odjavu s događaja prikazan je na ispisu 10.

public void OnFormUnsub(string s){

if (gameType == 3) {

this.MoveMade -= f.OnMoveMade;

f.GameEnded -= this.OnGameEnded;

f.MouseClicked -= this.OnMOuseClicked;

f.FormUnsub -= this.OnFormUnsub;

f.FormExiting -= this.OnFormClosing;

f.Close(); else{

this.MoveMade -= g.OnMoveMade;

g.GameEnded -= this.OnGameEnded;

g.MouseClicked -= this.OnMOuseClicked;

g.FormUnsub -= this.OnFormUnsub;

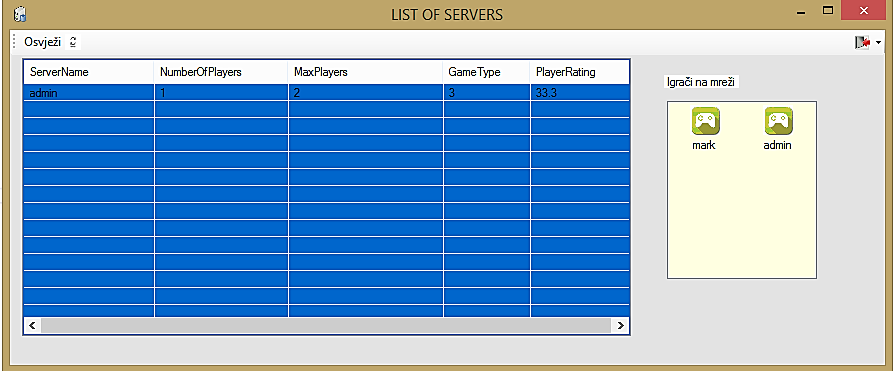
g.FormExiting -= this.OnFormClosing;

g.Close();}}

Ispis 10: Kôd za odjavu s događaja

### **3.2.6. Forma Lista Poslužitelja**

Forma ListaPoslužitelja prikazana je na slici 14.

****

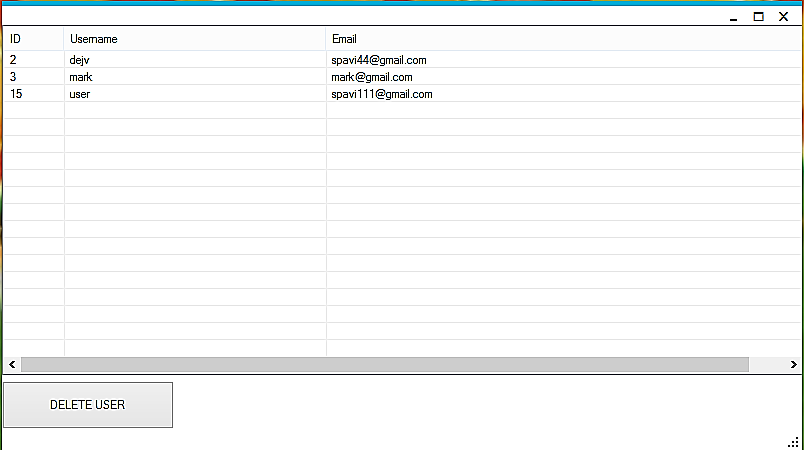
Slika 14: Forma ListaPoslužitelja

Lista „Popis poslužitelja“ sadrži kreirane poslužitelje koji su dostupni korisniku. Prikazani su detaljni podatci o imenu poslužitelja, podatci o vrsti igre koja će biti pokrenuta na tom poslužitelju te rejting korisnika koji je kreirao poslužitelja. U listi „Igrači na mreži“ nalaze se korisnici koji su trenutno prijavljeni u aplikaciji. Dugme *Refresh* dodano je u formu kako bi liste „Igrači na mreži“ i „Popis poslužitelja“ bile sinkronizirane s bazom.

Korisnik odabire željeni poslužitelj te dvoklikom miša pokušava pristupiti poslužitelju. Ako se klijent uspješno poveže na poslužitelja otvara se forma Klijent, dok forma ListaPoslužitelja prelazi u pozadinu i njezine kontrole više nisu vidljive korisniku. Ako je poslužitelj popunjen ili ako postoje tehnički problemi, dolazi do neuspješnog povezivanja.

### **3.2.7. Forma ListaIgrača**

Formi ListaIgrača moguće je pristupiti samo s ulogom administratora. U listi „Korisnici“ nalaze se svi korisnici sa svojim osnovnim informacijama. Klikom na dugme *Delete User* iz baze se uklanja odabrani korisnik. Forma ListaIgrača prikazana je na slici 15.

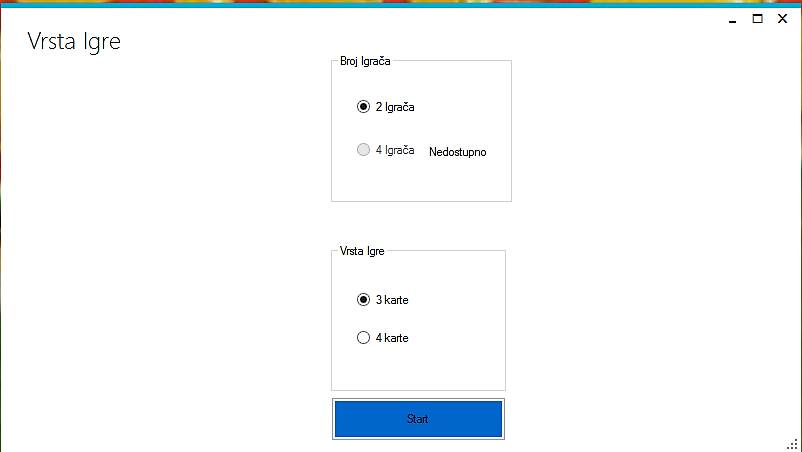


Slika 15: Forma ListaIgrača

### **3.2.8. Forma Vrsta Igre**

Vrsta Igre omogućuje da se prilikom izrade poslužitelja namjeste željene postavke igre koja će se pokretati na tom poslužitelju. Radijsko dugme (engl. *radio button*) *Broj* *igrača* sadrži mogućnost odabira maksimalnoga broja korisnika koji mogu pristupiti poslužitelju. Trenutno se ne može mijenjati, odnosno igra se može odvijati samo između dva korisnika.

Radijsko dugme *Vrsta Igre* sadrži mogućnost odabira igre s tri ili s četiri karte. Ako se radijska dugmad ne promijene, koriste se podrazumijevane postavke: dva igrača i tri karte. Klikom miša na dugme *Pokreni* pokreće se forma Poslužitelj, dok forma VrstaIgre prelazi u pozadinu. Forma Vrsta Igre prikazana je na slici 16.



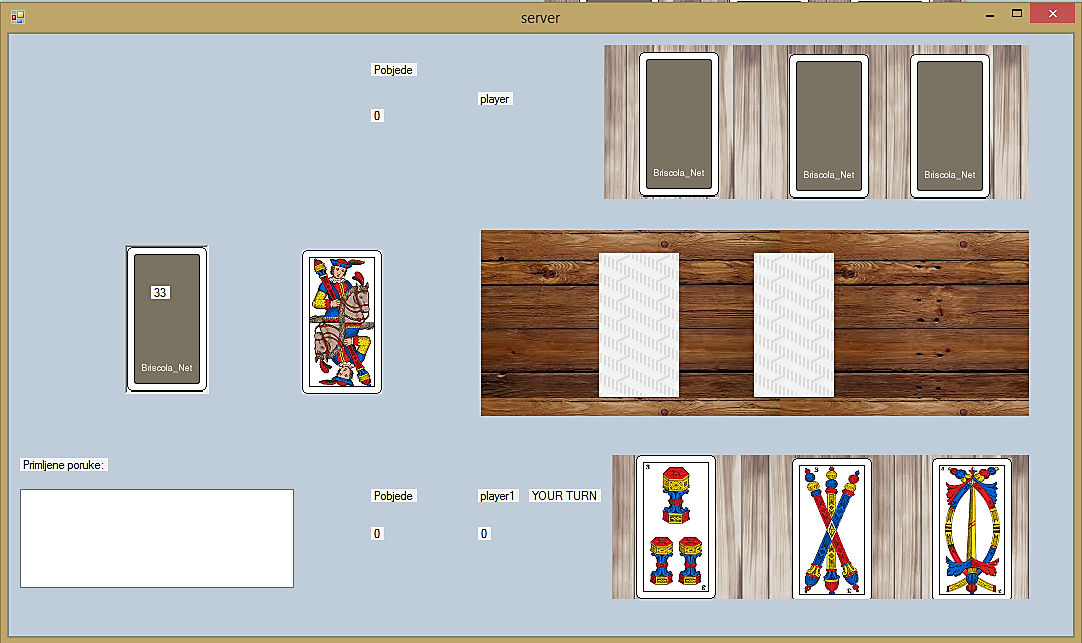
Slika 16: Forma VrstaIgre

### **3.2.9. Forma IgraTri**

U formi IgraTri odvija se igra briškula. Forma IgraTri sadrži *Picturebox- ove* koje sadrže odgovarajuću sliku za pojedinu karte. *Panel* Igrač1 sadrži karte „igrača1“. Za svaku kartu „igrača1“ postavljena je odgovarajuća slika u *Picturebox - u* . *Panel* Igrač2 sadrži karte „igrača2“. Za svaku kartu „igrača2“ postavljena je pozadinska slika u okvir za slike. U okviru za slike *Igra* postavljena je adutska karta. *Picturebox* *Špil* sadrži pozadinsku sliku s brojem preostalih karata. Forma sadrži naslove (engl. *labels*) koji omogućuju korisniku lakše praćenje tijeka igre. Neki od važnijih naslova jesu sljedeći naslovi:

* pobjede – pokazuje broj pobjeda „igrača1“ i „igrača2“
* punti – prikazuje broj punata „igrača1“ , dok punti protivnika nisu vidljivi
* baci kartu - pokazuje kojeg je igrača red baciti kartu.

Forma IgraTri prikazana je na slici 17.



Slika 17: Forma IgraTri

### **3.2.10. Forma IgraČetiri**

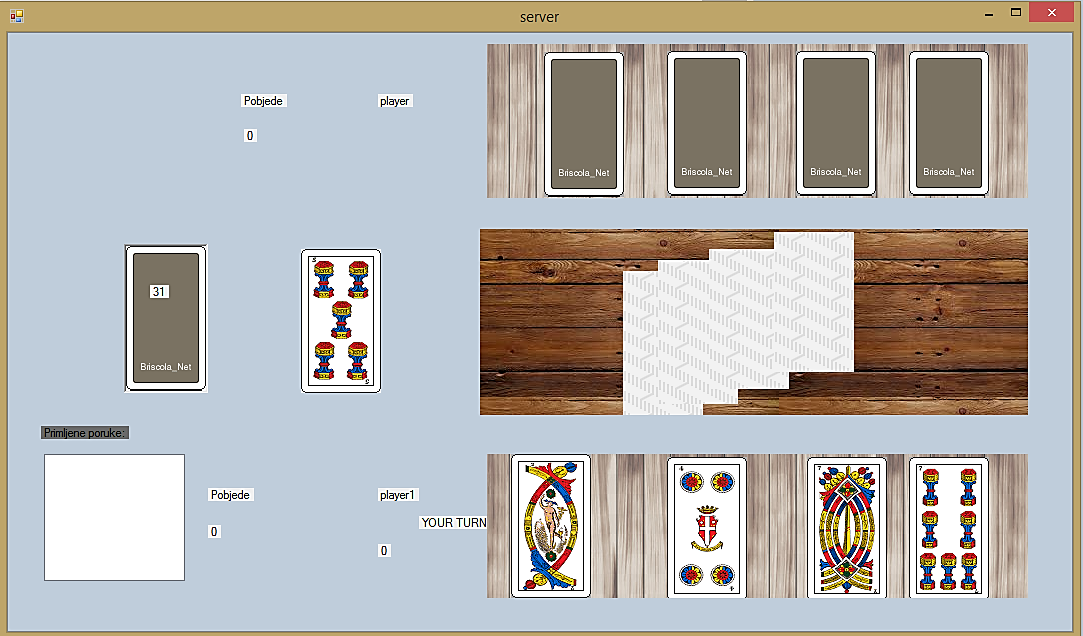
Forma IgraČetiri sadrži *PictureBox* - ove koji sadrže odgovarajuću sliku za pojedinu kartu. *Panel* Igrač1 sadrži karte „igrača1“. *Panel* Igrač2 sadrži karte „igrača2“. Za svaku kartu „igrača2“ postavljena je pozadinska slika u svaki *PictureBox*. U *panelu* Igrač1 i Igrač2 nalaze se po *PictureBox -* a.

U *PictureBox* - u *Igra* postavljena je adutska karta. *PictureBox* *Špil* sadrži pozadinsku sliku s brojem preostalih karata. Lista „Receieved Messages“ sadrži sve poruke od klijenta ili poslužitelja. U toj vrsti igre svaki igrač započinje igru s četiri karte. U svakoj ruci igrač treba odigrati po dvije karte te nakon završetka ruke dobiva nove dvije karte iz špila.

Forma sadrži naslove koji omogućavaju korisniku lakše praćenje tijeka igre. Neki od važnijih naslova jesu sljedeći naslovi:

* pobjede – pokazuje broj pobjeda „igrača1“ i „igrača2“
* punti – prikazuje broj punata „igrača1“.
* red igranja – pokazuje kojeg je igrača red baciti kartu.

Forma IgraČetiri prikazana je na slici 18.



Slika 18: Forma IgraČetiri

### **3.2.11. Višekorisnička** **logika**

U formama IgraTri i IgraČetiri implementirana je višekorisnička logika. Klikom na *Picturebox1*, *Picturebox2*, *Picturebox3* ili *Picturebox4* aktivira se događaj MouseClicked te se izvršava metoda OnMouseClicked. Ako je poslužitelj aktivirao događaj MouseClicked, šalje se poruka svim klijentima na mreži. Poruka sadrži ime *Picturebox - a* u kojem se aktivirao događaj, tj. na koji je igrač kliknuo mišom. U formi Klijent aktivira se događaj ClientMove te se u formi IgraTri izvršava metoda OnClientMove. Implementacija metode OnClientMove u formi IgraTri prikazana je na ispisu 11. Prikazan je kôd gdje se vidi koji okvir za sliku će se aktivirati na klijentovoj strani ovisi o okviru za sliku koji je aktiviran na poslužiteljevoj strani.

|  |
| --- |
| public void OnClientMove (string s) {  if (s.Equals("pictureBox1")){  pictureBox5\_Click(this, EventArgs.Empty);  MessageBox.Show("card thrown"); }  else if (s.Equals("pictureBox2")){  pictureBox6\_Click(this, EventArgs.Empty);  MessageBox.Show("card thrown");}  else if (s.Equals("pictureBox3")){  pictureBox7\_Click(this, EventArgs.Empty);  MessageBox.Show("card thrown");}} |

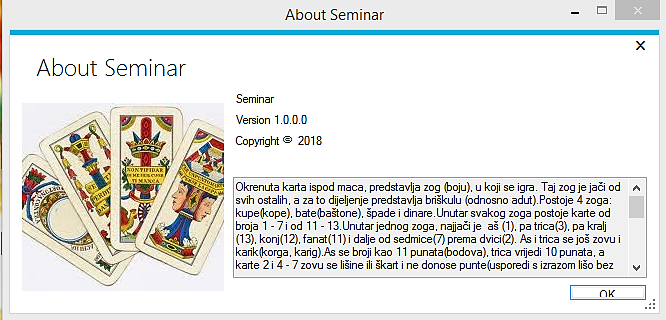
Ispis 11: Kôd OnMoveMade metoda

Ako se događaj MouseClicked aktivirao na klijentovoj strani, šalje se poruka poslužitelju. U formi Poslužitelj aktivira se događaj MoveMade te se u formi IgračTri izvršava metoda OnMoveMade. Za računanje broja bodova, poslužitelja i klijenta, odnosno „igrača1“ i „igrača2“ implementirana je metoda Punti. Metoda kao parametar prima karte koje se nalaze u panelu *Desk* te računa njihovu ukupnu vrijednost.

U metodi Pokupi implementirana su pravila igre te pravila za dijeljenje novih karata igračima. Kada dođe do završetka igre aktivira se događaj GameEnded, izvršava se metoda OnGameEnded implementirana u formi Poslužitelj. Preko OnGameEnded metode poslužitelj obavještava klijenta o ishodu igre te se poziva NewGame metoda.

### **3.2.12. Forma PravilaIgre**

PravilaIgre Forma sadrži informacije o nazivu aplikacije, verziji, autoru aplikacije te pravilima igre. Forma PravilaIgre prikazana je na slici 19.



**Slika 19:** Forma PravilaIgre

### **3.2.13. Forma IgraGotova**

Prikazuje se nakon završetka igre. Sadrži *Picturebox*, a slika se mijenja ovisno o ishodu igre.

### **3.2.14. Klasa Špil**

Klasa Špil koristi se za izradu špila karata. Svaka karta opisana je atributima vrijednost i vrsta. Klasa Špil koristi pobrojane vrijednosti (engl. *enum*) *Vrsta* i *Vrijednost*, što omogućava jednostavniju izradu karte, a time i samoga špila. Na ispisu 12. prikazana je upotreba pobrojanih vrijednost *Vrsta* i *Vrijednost*.

public enum Vrsta { Kupe, Spade, Dinari, Bastoni };

public enum Vrijednost { dva, četri, pet, šest, sedam, fanat, konj, kralj,tri,As };

Ispis 12: Kôd pobrojane vrijednosti Vrsta i Vrijednost

Metoda setCards služi za izradu špila karata koji se koristi u igri. Na ispisu 13. prikazana je implementacija setCards metode.

public void setCards(){

int i = 0;

foreach (Vrsta v in Enum.GetValues(typeof(Vrsta))){

foreach (Vrijednost vr in Enum.GetValues(typeof(Vrijednost)){

string s = v.ToString() + vr.ToString(

this.karte[i] = new Cards(v, vr);

i++;}}}

Ispis 13: Kôd setCards metoda

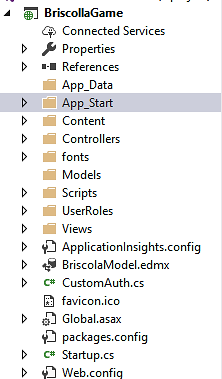
### **3.2.15. Klasa Igrač**

Klasa *Igrač* sadrži atribute *ime*, *punti*, *red*, *ruku*.

## 3.3. Web aplikacija

### **3.3.1. Struktura *web* aplikacije**

Struktura *web* aplikacije prikazana je na slici 20.



Slika 20: Struktura *web* aplikacije

### **3.3.2. Prava pristupa**

Za prijavu korisnika i registraciju korisnika u *web* aplikaciju izrađen je prilagođeni (engl. *custom*) autentifikacijski sustav. Prava pristupa podijeljena su na tri uloge: kao neprijavljeni posjetitelj, kao igrač (engl. *player*) te kao administrator.

*Neprijavljeni korisnik pristup*

Neprijavljeni korisnik se pri pokretanju *web* aplikacije preusmjerava na formu za prijavu. Prikazi kojima neprijavljeni korisnik može pristupiti predznačeni su atributom [AllowAnonymous].

*Igračev pristup*

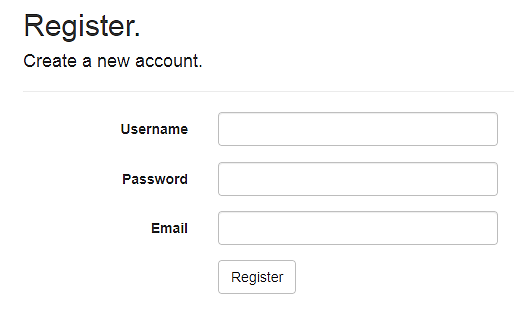
Korisnik može pristupiti onim prikazima predznačenim atributima: [AllowAnonymous], [CustomAuth(Roles="Player")].

*Administratorski pristup*

Administratorskim pristupom korisnik dobiva potpunu kontrolu nad aplikacijom. Administrator može pristupiti svim prikazima dok, korisnici s manjom razinom pristupa ne mogu pristupiti prikazima koje su predznačene atributom [CustomAuth(Roles = "Admin")].

### **3.3.3. Forma za registraciju**

Forma za registraciju prikazana je na slici 21.



Slika 21: Forma za registraciju

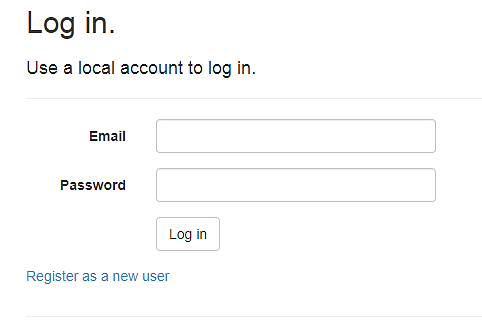
Forma za registraciju sastoji se od polja za korisničko ime, lozinku i e-poštu. Nakon klika na dugme *Register* provjerava se jesu li zadovoljeni uvjeti za sva tekstualna polja. Uvjeti za uspješnu registriraciju korisnika jesu sljedeći:

* lozinka minimalne veličine šest znakova
* ispravna elektronička pošta
* nepostojeće korisničko ime i elektronička pošta u bazi podataka.

Nakon uspješne registracije korisnik može pristupi *web* i *desktop* aplikaciji. Ako je registracija neuspješna i ako nisu zadovoljeni istaknuti uvjeti, ispisati će se odgovarajuća poruka za tekstualna polja koje nisu zadovoljila navedene kriterije.

### **3.3.4**. **Forma za prijavu**

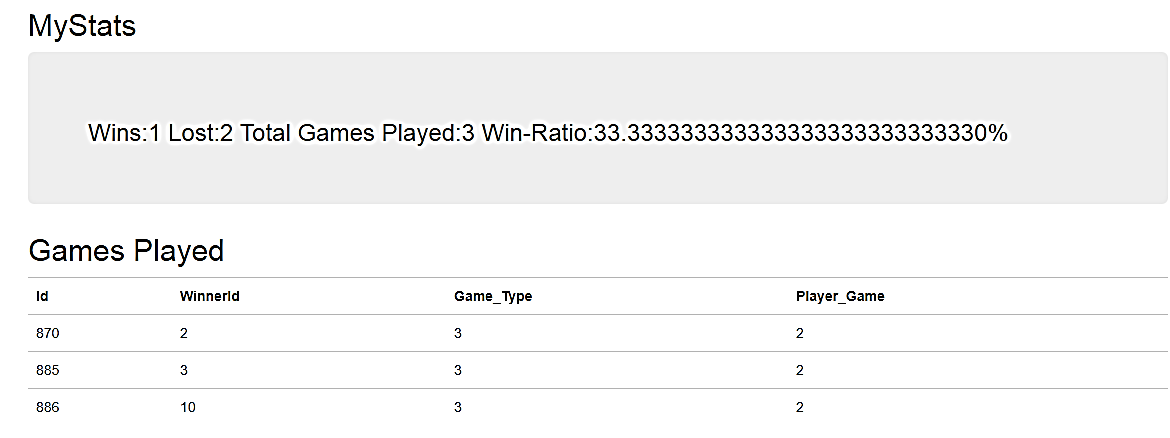
Prilikom pokretanja aplikacije posjetitelj se preusmjerava na formu za prijavu. Forma za prijavu korisnika sastoji se od tekstualnih polja *Email* i *Password*, dugmeta *Log* *in* te poveznice *Register*. Posjetitelji koji ne posjeduju račun mogu preko poveznice pristupiti formi za registraciju. Forma za prijavu prikazana je na slici 22.



Slika 22: Forma za prijavu

### **3.3.5**. **Prikaz Moji rezultati**

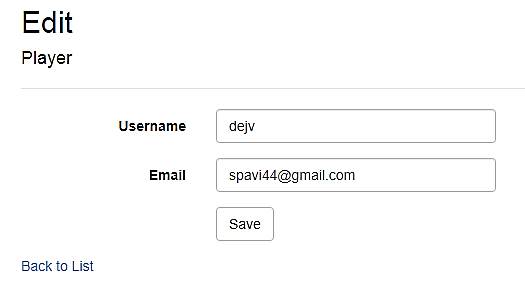
MyStats prikazuje broj odigranih partija, broj pobjeda i poraza te ukupni postotak pobjeda trenutno prijavljenog korisnika. U listi „Games Played“ sadržana je kompletna povijest igara u kojima je korisnik sudjelovao te ishod pojedine igre. Moji rezultati prikazani su na slici 23.

****

Slika 23: Prikaz Moji rezultati

### **3.3.6**. **Account Manager prikaz**

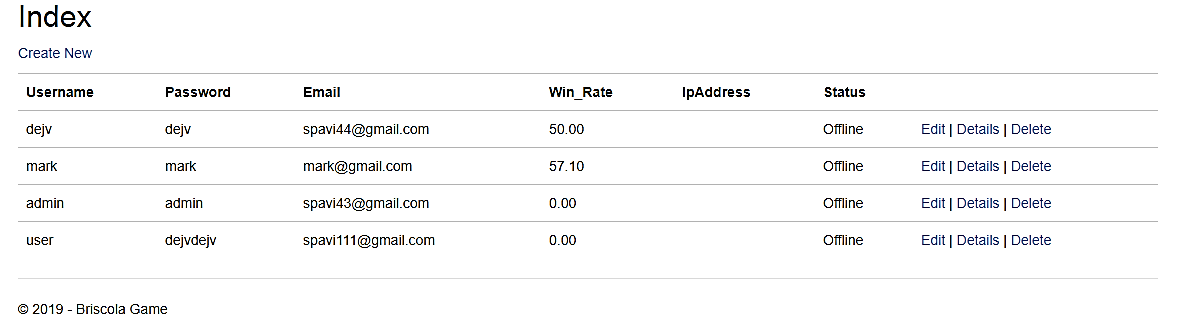
Account Manager prijavljenim korisnicima omogućava promjenu korisničkog imena i elektroničke pošte. Klikom na dugme *Spremi* promjene se pohranjuju u bazu podataka. Account Manager prikazan je na slici 24.



Slika 24: Prikaz Account Manager

### **3.3.7**. **Forma Lista igrača**

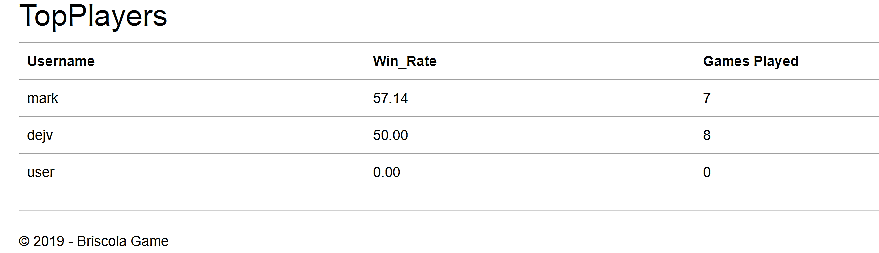
U formi Lista igrača sadržan je popis svih korisnika, s povjerljivim informacijama. Lista igrača pokraj podataka svakog korisnika sadrži poveznice *Edit*, *Details* i *Delete*. Preko forme Lista igrača administrator može kreirati nove korisnike, brisati postojeće, pregledati detaljne informacije o korisniku te promijeniti korisnikov profil. Lista igrača prikazana je na slici 25.



Slika 25: Prikaz Lista igrača

### **3.3.8**. **Prikaz TopPlayers**

Forma TopPlayers sadrži listu najbolje rangiranih igrača prema povijesti odigranih igara. Prikaz Najbolji Igrači prikazan je na slici 26.



Slika 26: Prikaz Najbolji Igrači

Kriteriji rangiranja su prema postotku pobjeda te po broju odigranih igara. Taj kriterij onemogućuje da igrač koji ima jednu pobjedu uz jednu odigranu igru bude bolje rangiran od igrača koji ima manji postotak pobjeda, ali mnogo veći broj odigranih igara. Primjer upita LINQ-a prikazan je na ispisu 14.

var p = players.Players.OrderBy(r=>r.Games.Count).OrderByDescending(r=> r.Win\_Rate).Take(3).ToList();

Ispis 14: Kôd Upit LINQ-a

# 4. ZAKLJUČAK

Tema je završnoga rada izrada višekorisničke kartaške igre u *desktop* verziji s pripadnom *web* aplikacijom. Ideja je bila izrada aplikacije koja se koristi za zabavu korisnika.

Prilikom izrade aplikacije stekao se uvid u prednosti i nedostatke pojedinih tehnologija. Glavna prednost izrade aplikacija korištenjem Microsoft tehnologija jest opsežna dokumentacija, što je omogućilo uspješnu izradu aplikacije te svladavanje svih problema prilikom izrade.

Izbor programskog jezika sveo se na C# zbog njegove jednostavnosti te velike lepeze mogućnosti. C# kombiniran s Windows formama koje su dio .NET tehnologije predstavlja razvojno okruženje koje omogućuje razvoj bilo kojeg tipa aplikacija, čime se pokrivaju svi zahtjevi tržišta. Windows forme zbog jednostavnog implementiranja međusobne komunikacije predstavljaju idealan način izrade aplikacija u kojima je implementirana višekorisnička logika.

Razvoj aplikacije se nastavlja. Kako bi aplikacija dosegla svoj puni potencijal potrebno je uložiti još puno rada u njezino poboljšanje. U budućnosti će *desktop* aplikacija dobiti nove funkcionalnosti poput mogućnosti prisustvovanja do četiri korisnika u igri, poboljšat će se način komunikacije između igrača, dok će *web* aplikacija dobiti funkcionalnosti koje će omogućiti pokretanje igre iz *web* preglednika.

# 5. LITERATURA

1. Zbornik Veleučilišta u Rijeci, 2014, vol. 2, no. 1,  [<https://www.veleri.hr/files/datoteke/knjige/digi/2014_06_01_Veleuciliste%20Zbornik.pdf> (posjećeno 15. 7. 2019.)](https://www.veleri.hr/files/datoteke/knjige/digi/2014_06_01_Veleuciliste%20Zbornik.pdf%20(posjećeno%2015.%207.%202019.))
2. Smijulj, A. i Meštrović, A.: Izgradnja MVC modularnog radnog okvira, Zbornik Veleučilišta u Rijeci, 2014, vol. 2, no. 1, str. 215-232, <https://hrcak.srce.hr/file/190412> (posjećeno 28. 6. 2019.)
3. Uvod u ASP .Net MVC, <http://www.manuelradovanovic.com/2017/09/uvod-u-asp-net-mvc.html> (posjećeno 25. 6. 2019.)
4. Instalacija razvojnog okruženja, <https://petlja.org/biblioteka/r/lekcije/prirucnik-cpp/instalacija-razvojnog-okruzenja> (posjećeno 25.6.2019.)
5. SQL Server Management Studio, <https://www.guru99.com/sql-server-management-studio.html#2> (posjećeno 24.6.2019.)
6. Modeliranje, implementacija i administracija baza podataka, <https://bib.irb.hr/datoteka/979191.Modeliranje_implementacija_i_administracija_baza_podataka.pdf> (posjećeno 27.7.2019)
7. MS SQL SERVER (2018) - Osnove upravljanja serverskim nalozima, <https://programirajmozajedno.wordpress.com/2016/09/13/ms-sql-server-logins/> (posjećeno 21.7.2019)
8. Kratofil, D.: Izrada *web* aplikacije u programskom jeziku C# (Završni rad), Fakultet elektrotehnike, računarstva i informacijskih tehnologija Sveučilišta Josipa Jurja Strossmayera u Osijeku, Osijek, 2017, [https://urn.nsk.hr/urn:nbn:hr:200:999545 (posjećeno 11. 7. 2019.)](https://urn.nsk.hr/urn:nbn:hr:200:999545)
9. Rad s bazom podataka, <https://www.scribd.com/document/335756231/RPA-P5-Rad-s-Bazom-Podataka> (posjećeno 20.6.2019)
10. Detaljni pregled LINQ – Integrirani SQL upiti u .NET programskom jeziku, [https://bhrnjica.net/2010/10/26/detaljni-pregled-linq-integrirani-sql-upiti-u-net-programskom-jeziku/ (posjećeno 15. 7. 2019.)](https://bhrnjica.net/2010/10/26/detaljni-pregled-linq-integrirani-sql-upiti-u-net-programskom-jeziku/%20(15.%207.%202019.))
11. From zero to hero in Json with C#, <https://www.c-sharpcorner.com/article/from-zero-to-hero-in-json-with-c-shar/> (posjećeno 25. 4. 2019.)
12. Lovrić, V: Windows forme u C#-u (Završni rad), Fakultet elektrotehnike, računarstva i informacijskih tehnologija Sveučilišta Josipa Jurja Strossmayera u Osijeku, Osijek, 2016, [https://urn.nsk.hr/urn:nbn:hr:200:800058 (posjećeno 17. 7. 2019.)](https://urn.nsk.hr/urn:nbn:hr:200:800058%20(posjećeno%2017.%207.%202019.))

# 6. POPIS SLIKA I ISPISA

**Popis slika:**

Slika 1: Visual Studio logo......................................................................................................5

Slika 2: SQL Server Management Studio logo.......................................................................6

Slika 3: Forma za prijavu na SQL poslužitelj..........................................................................7

Slika 4: Mapiranje u razvojnom okviru Entity........................................................................7

[Slika 5: Baza prva pristup 10](#_Toc15816104)

[Slika 6: Tablica Korisnik 10](#_Toc15816105)

[Slika 7: Relacijski model baze podataka 12](#_Toc15816106)

[Slika 8: Struktura *desktop* aplikacije 13](#_Toc15816107)

[Slika 9: Forma za prijavu 14](#_Toc15816108)

[Slika 10: Forma PoslužiteljKlijent s administratorskim pravima 15](#_Toc15816109)

[Slika 11: Forma Poslužitelj 16](#_Toc15816110)

[Slika 12: Forma Klijent 19](#_Toc15816111)

[Slika 13: Forma ListaPoslužitelja 21](#_Toc15816112)

[Slika 14: Forma ListaIgrača 22](#_Toc15816113)

[Slika 15: Forma VrstaIgre 23](#_Toc15816114)

[Slika 16: Forma IgraTri 24](#_Toc15816115)

[Slika 17: Forma IgraČetiri 25](#_Toc15816116)

[Slika 18: Forma PravilaIgre 27](#_Toc15816117)

[Slika 19: Struktura *web* aplikacije 29](#_Toc15816118)

[Slika 20: Forma za registraciju 30](#_Toc15816119)

[Slika 21: Forma za prijavu 31](#_Toc15816120)

[Slika 22: Prikaz Moji rezultati 32](#_Toc15816121)

[Slika 23: Prikaz Account Manager 32](#_Toc15816122)

[Slika 24: Prikaz Lista igrača 33](#_Toc15816123)

[Slika 25: Prikaz Najbolji Igrači 34](#_Toc15816124)

**Popis ispisa:**

[Ispis 1: Kontekstna klasa 8](#_Toc15816142)

[Ispis 2: PlayersEntities1 konstruktor 12](#_Toc15816143)

[Ispis 3: String za konekciju s Microsoft Azure SQL serverom 12](#_Toc15816144)

[Ispis 4: Kôd za pretplatu na događaje SimpleTcpServer klase 17](#_Toc15816145)

[Ispis 5: Kôd JSON serijalizacija 18](#_Toc15816146)

[Ispis 6: Kôd pretplata na događaje 18](#_Toc15816147)

[Ispis 7: Kôd za uklanjanje klijenta sa poslužitelja 18](#_Toc15816148)

[Ispis 8: Kôd Client pretplata na DataReceived događaj 19](#_Toc15816149)

[Ispis 9: Kôd deserijalizacija objekta 20](#_Toc15816150)

[Ispis 10: Kôd za odjavu s događaja 21](#_Toc15816151)

[Ispis 11: Kôd OnMoveMade metoda 26](#_Toc15816152)

[Ispis 12: Kôd pobrojane vrijednosti Vrsta i Vrijednost 28](#_Toc15816153)

[Ispis 13: Kôd setCards metoda 28](#_Toc15816154)

[Ispis 14: Kôd Upit LINQ-a 34](#_Toc15816155)