**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, lipanj 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, lipanj 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, lipanj 2019.

**SADRŽAJ**

[SAŽETAK 1](#_Toc15754202)

[SUMMARY 2](#_Toc15754203)

[1. UVOD 3](#_Toc15754204)

[2. TEHNOLOGIJE 4](#_Toc15754205)

[2.1. ASP .NET MVC 4](#_Toc15754206)

[2.2. Entity Framework 6 4](#_Toc15754207)

[2.3. Language Integrated Query (LINQ) 6](#_Toc15754208)

[2.4. JavaScript Object Notation (JSON) 7](#_Toc15754209)

[2.5. Windows forme 7](#_Toc15754210)

[3. APLIKACIJA 8](#_Toc15754211)

[3.1. Baza podataka 8](#_Toc15754212)

[3.2. *Desktop* aplikacija 10](#_Toc15754213)

[3.2.1. Struktura *desktop* aplikacije 10](#_Toc15754214)

[3.2.3. Forma PoslužiteljKlijent 12](#_Toc15754215)

[3.2.4. Forma Poslužitelj 13](#_Toc15754216)

[3.2.5. Forma Klijent 17](#_Toc15754217)

[3.2.6. Forma Lista Poslužitelja 19](#_Toc15754218)

[3.2.7. Forma ListaIgrača 20](#_Toc15754219)

[3.2.8. Forma Vrsta Igre 21](#_Toc15754220)

[3.2.9. Forma IgraTri 21](#_Toc15754222)

[3.2.10. Forma IgraČetiri 22](#_Toc15754223)

[3.2.11. Višekorisnička logika 24](#_Toc15754224)

[3.2.12. Forma PravilaIgre 25](#_Toc15754225)

[3.2.13. Forma IgraGotova 25](#_Toc15754226)

[3.2.14. Klasa Špil 25](#_Toc15754227)

[3.2.15. Klasa Igrač 26](#_Toc15754228)

[3.3. *Web* aplikacija 27](#_Toc15754229)

[3.3.1. Struktura *web* aplikacije 27](#_Toc15754230)

[3.3.2. Prava pristupa 27](#_Toc15754231)

[3.3.3. Forma za registraciju 28](#_Toc15754232)

[3.3.4. Forma za prijavu 29](#_Toc15754233)

[3.3.5. Prikaz Moji rezultati 30](#_Toc15754234)

[3.3.6. Account Manager prikaz 30](#_Toc15754235)

[3.3.7. Forma Lista igrača 31](#_Toc15754236)

[3.3.8. Prikaz Najbolji igrači 31](#_Toc15754237)

[4. ZAKLJUČAK 33](#_Toc15754238)

[5. LITERATURA 34](#_Toc15754239)

[6. POPIS SLIKA I ISPISA 36](#_Toc15754240)

# 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 consist 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 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 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. 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 server and online players at that time. Web application help 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 je završnoga rada 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 višekrisnička 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 upućuje na daljnji razvoj aplikacije u budućnosti. Peto poglavlje završnog rada sadrži popis literature korištene prilikom pisanja 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 serverskoj 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 kôd izrada *web* aplikacija [1]. Osnovna ideja MVC arhitekture jest razdvajanje kôda na tri cjeline [2]. MVC (model, pregled, kontroler) je razvojni okvir koji se koristi za izradu *web* aplikacija otvorenoga kôda. ASP .NET MVC arhitektura 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 opisuje 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 je baziran na kontrolorima“ [3]. ASP .NET 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 razvojna okolina (engl. *Integrated* *Development* *Environment*), služi za pisanje raznih vrsta aplikacija za operativni sustav Windows. Tvorac razvojnoga okruženja Visual Studio je tvrtka Microsoft. Razvojno okruženje Visual Studio trenutno dolazi u tri varijateta:

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

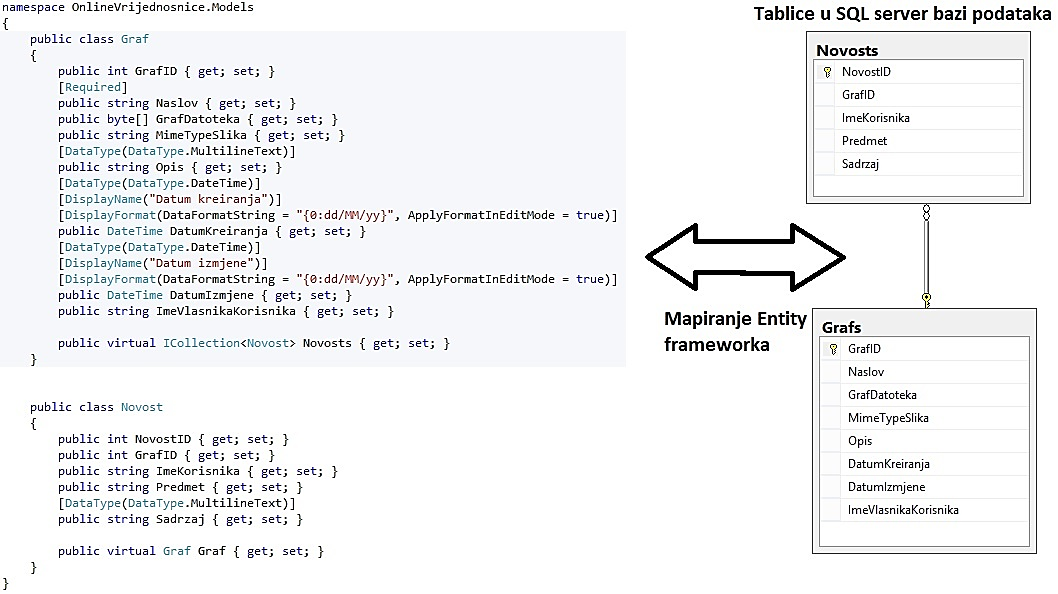
. Visual Studio Community je moguće besplatno preuzeti na službenim stranicama Microsoft- a. Instalacija razvojnog okruženja Visual Studio Community zahtijeva posjedovanje besplatnoga Microsoft naloga, te da bi se isntalacija uspiješno izvršila potrebna je stalna internet veza. Visual Studio omogućava integraciju različitih tehnologija koje ne moraju biti razvijene od strane Microsoft kompanije.

## 2.3. Entity Framework 6

Razvojni okvir Entity (engl. *Entity Framework*) 6 je okvir objektno-relacijskog mapiranja (skraćenica ORM) otvorenog kôda za ADO.NET. ORM pristup značajno olakšava razvoj podatkovno orijentiranih aplikacija. Od verzije 6 razvojni okvir Entity je odvojen od .NET okvira [4]. Razvojni okvir Entity je postao jedna od vodećih tehnologija koja se koristi za pristup podatcima [5]. U verziji 6 razvojnog okvira Entity dodani su novi pristupi:

* baza prva (engl. *database* *first*)
* kôd prvi (engl. *code first*).

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 koda C# programskog jezika i SQL baze podataka se naziva mapiranje (engl. *mapping*) . Razvojni okvir entity mapira relacijske tabilice u objekte koje C# razumije. Na slici 1. prikazano je mapiranje u EF-u.



**Slika 1:** Mapiranje u EF-u

Postoji više načina da se napravi ORM 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 [5].

Kao most (engl. *bridge*) između *entity* klasa i baze podataka koristi se kontekstna klasa (engl. *dbcontext*). Izgled PlayersEntities1 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: PlayersEntities1 klasa

## 2.4. 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 [6]. 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 lambda izrazi (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.

## 2.5. JavaScript Object Notation (JSON)

JavaScript Object Notation (skraćenica JSON) je tekstualni format čija je namjena prijenos podataka u formatu koji je čitljiv i ljudima i strojevima [10]. 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 [10].

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

## 2.6. Windows forme

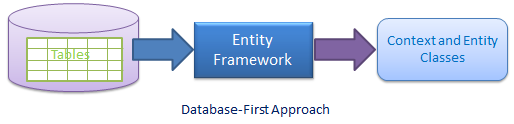
Windows forma je alat koji se koristi za izradu Windows aplikacija. Te su forme učinkovit i jednostavan način komunikacije koja se ostvaruje između korisnika i programa. Window forma je prozor koji sadrži kontrole za prikaz, unos i manipulaciju podatcima [10]. Za izradu Windows formi potrebno je unutar 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 [9]. 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 [10].

# 3. APLIKACIJA

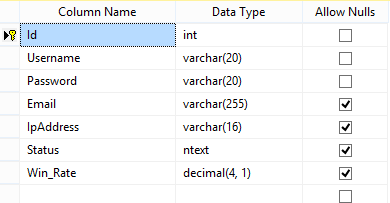
## 3.1. Baza podataka

Prilikom korištenja EF prvo što trebamo napraviti je odlučiti se na koji način ćemo stvoriti bazu podataka. Za izradu baze podataka izbor se sveo baza prva pristup. Na temelju izrađene baze EF automatski generira potrebne modele. Baza podataka koristi se za pohranjivanje podataka registriranih korisnika te za pohranjivanje ishoda igre. Iz ER (engl. *Entity Relationship*) dijagrama vidljivo je da se model podataka sastoji od pet entiteta. Na slici 2. prikazana je prva baza modela EF-ka.

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

Slika 2: Baza prva pristup

Baza podataka izrađena je pomoću Microsoft SQL Server Management Studio 2017 alata. Nakon izrade baze podataka slijedi izrada pojedinih tablica. Primjer izrađene tablice prikazan je na slici 3.



**Slika 3:** Tablica Korisnik

Na slici 4. 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 4: Relacijski model baze podataka

Podatci o korisniku (engl. *user*), korisničkim ulogama (engl. *role*) te igri (engl. *game*) pohranjeni su u bazi podataka Igrači (engl. *players*). Baza podataka kreirana je pomoću alata Microsoft SQL Express Server, a postavljena je na Microsoft Azure Sql poslužitelj. Za manipulaciju podatcima koji se nalaze u bazi podataka treba povezati aplikaciju s bazom. DBContext upravlja vezom s bazom te se po potrebi spaja ili odspaja s baze.PlayersEntities1 klasa izvedena je iz DBContext klase. Konstruktor (engl. *constructor*) PlayersEntities1 klasa 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 serverom 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 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"

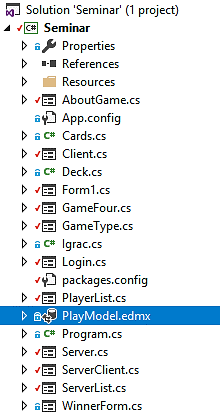
Ispis 3: *String* za konekciju s Microsoft Azure Sql poslužiteljem

## 3.2. Desktop aplikacija

U praktičnom dijelu rada kroz *desktop* aplikaciju demonstrirana je upotreba Windows formi, EF-a , JSON i tcp tehnologije. Aplikacija sadrži jedanaest formi te tri klase.

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

Struktura *desktop* aplikacije prikazana je na slici 5.

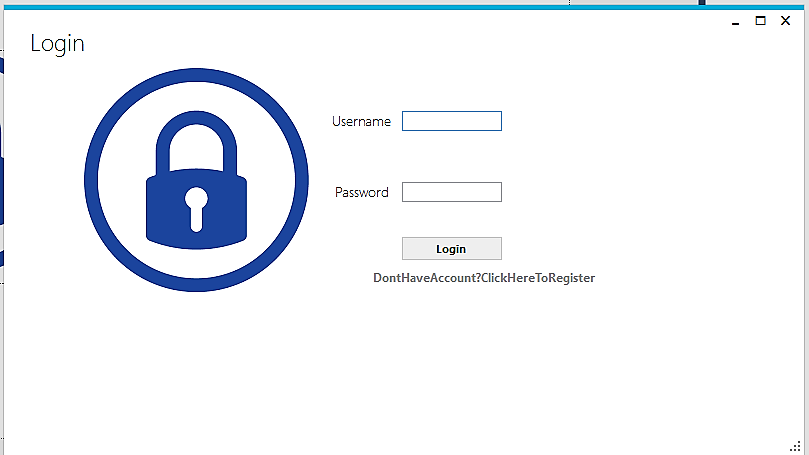


Slika 5: Struktura *desktop* aplikacije

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

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

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 6.

.

Slika 6: Forma za prijavu

### **3.2.3. Forma PoslužiteljKlijent**

Forma PoslužiteljKlijent sadrži glavni izbornik. Izbornik u toj formi omogućuje navigaciju korisniku. Izgled forme ovisi o pravima pristupa pojedinog korisnika. Postoje dvije razine prava pristupa *desktop* 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

PoslužiteljKlijent forma posjeduje izbornik (engl. *menu*) koji sadrži opcije za sljedeće:

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

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 7. prikazana je forma PoslužiteljKlijent s administratorskim pravima.

****

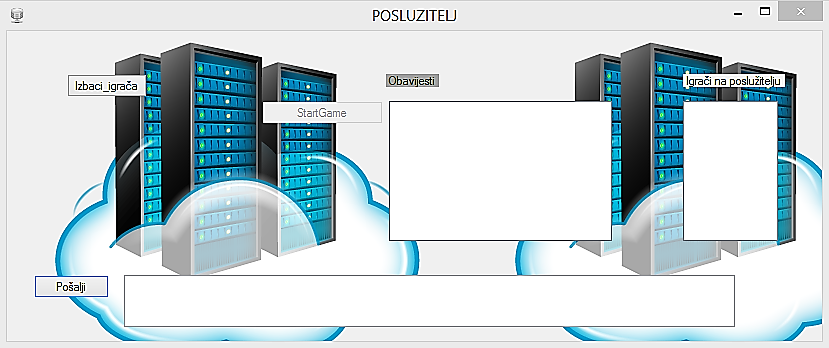
Slika 7: Forma PoslužiteljKlijent s administratorskim pravima

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 tablici igra (engl. *game*), tj. uklanjaju se sve igre iz baze podataka i svi trenutno pokrenuti poslužitelji.

### **3.2.4. Forma Poslužitelj**

Forma Poslužitelj prikazana je na slici 8.



Slika 8: Forma Poslužitelj

Pokretanjem forme Poslužitelj kreira se poslužitelj (engl. *server*), a pokretanjem poslužitelja kreira se nova igra. Ako se prilikom kreiranja poslužitelja ne promjene postavke u formi PostavkeIgre, kreirat će se igra s unaprijed zadanim vrijednostima:

* broj igrača: dva igrača
* broj karata: tri karte.

Nakon pokretanja poslužitelj na priključu (engl. *portu*) 7000 sluša zahtjeve klijenta za spajanje. Lista Igrači na poslužitelju sadrži popis korisnika koji su trenutno spojeni na poslužitelj. Lista klijenata spojenih na pristupnik osvježava se prilikom povezivanja klijenta na poslužitelj ili odspajanja s poslužitelja. Poslužitelj može odabranog klijenta odspojiti s mreže. Klikom na dugme *Izbaci igrača* klijent se preusmjerava na formu ListaIgrača te se uklanja iz liste „igrači na poslužitelju“. U listi obavijesti nalaze se sve poruke koje je poslao klijent.

Klikom na dugme *send* šalje se poruka svim klijentima spojenim na poslužitelj. Poruka sadrži tekst koji se nalazi u tekstualnom polju (engl. *TextBox*). Kako bi se omogućila komunikacija s klijentom obavlja se pretplata (engl. *subscribe*) na događaj (engl. *event*) DataReceived iz SimpleTcpServer klase. Kôd za pretplatu na događaje SimpleTcpServer klase prikazana 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. klijent se preplaćuje na događaje *ClientConnected* i *ClientDisconnected* koji se nalaze u klasi SimpleTcpServer. 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. Ako dođe do odspajanja klijenta s poslužitelja, poslužitelj će biti obaviješten o izlasku klijenta porukom.

Maksimalni broj klijenata koji mogu pristupiti poslužitelju postavlja se u formi Vrsta Igre, a po zadanoj vrijednosti postavljen je na jednog klijenta. Nakon spajanja ili odspajanja klijenta na poslužitelja osvježava se broj klijenata na poslužitelju. Dok klijent ne aktivira događaj klikom na dugme *Ready* onemogućeno je dugme *Start* te poslužitelj ne može pokrenuti igru.

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

JsonObject (MemberSerialization.OptIn) izvan deklaracije klase omogućuje da se korištenjem [JsonProperty] iznad deklaracije atributa odaberu atributi klase koja se želi serijalizirati.

Na ispisu 5. prikazana je serijalizacija objekta IgraTri te slanje serijaliziranog teksta klijentima spojenim na poslužitelj. Nakon slanja JSON stringa klijentima koji su spojeni na poslužitelj, izvršava se pretplata na sljedeće događaje.

json = JsonConvert.SerializeObject(f);

tcpServer.BroadcastLine(json);

Ispis 5: Kôd JSON serijalizacija

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

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

Nakon aktivacije događaja poslužitelj obavještava klijenta o završetku igre te se poziva metoda NewGame. Za provjeru ishoda igre koristi se metoda NewGame. Dok ne dođe do završetka igre prikazuje se *dialog result*, gdje se igraču nudi mogućnost nastavka igre.

Ako klijent ili poslužitelj odgovore *No*, klijent će se odspojiti s poslužitelja. Smatra se da je poražen onaj tko je prvi odgovorio s *No*. 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: Kod za uklanjanje klijenta sa servera

Ako su, pak, klijent ili poslužitelj odgovorili *Yes*, pokreće se nova igra. Metoda OnFormUnsub poziva se prilikom ponovnog pokretanja igre, a služi za odjavu forme Poslužitelj s pretplate svih događaja. Ako se ne izvrši odjava s događaja, prilikom pokretanja nove igre dolazi do povećanja pretplate na sve događaje, što u konačnici rezultira neispravnim radom same aplikacije.

### **3.2.5. Forma Klijent**

Forma Klijent prikazana je na slici 9.



Slika 9: 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. Klikom na dugme *Send* klijent šalje poruku poslužitelju, a poruka sadrži tekst koji se nalazi u tekstualnom polju. Aktivacijom dugmeta *Ready* klijent omogućuje poslužitelju da pokrene igru. U listi poruke nalaze se poruke razmijenjene s poslužiteljem. Kako bi komunikacija s poslužiteljem bila moguća klijent se mora pretplatiti na događaj iz SimpleTcpClient klase. Kôd Client pretplata na DataReceived događaj prikazan je na ispisu 8.

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

Ispis 8: Kôd Client pretplata na DataReceived događaj

U formi Klijent obavlja se deserijalizacija objekta tipa IgraTri ili IgraČetiri, ovisno o vrste igre koju je poslužitelj kreirao. String je prethodno serijalizirao poslužitelj te je poslan klijentu korištenjem tcp protokola. 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. Kod 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;

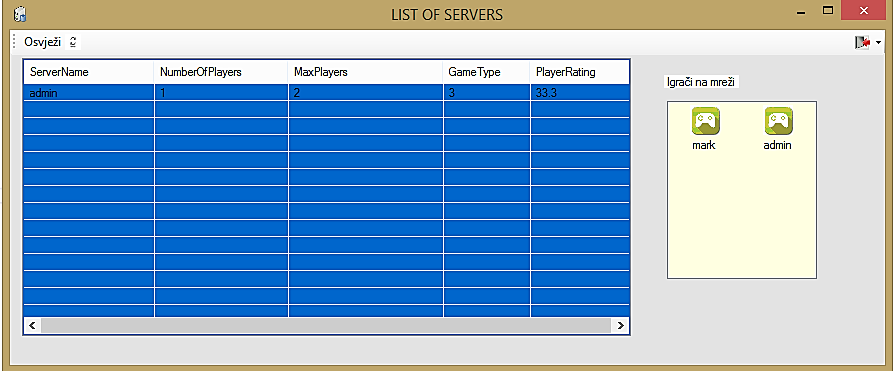
g.FormExiting -= this.OnFormClosing;

g.Close();}}

Ispis 10: Kod za odjavu s događaja

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

Forma ListaPoslužitelja prikazana je na slici 10.

****

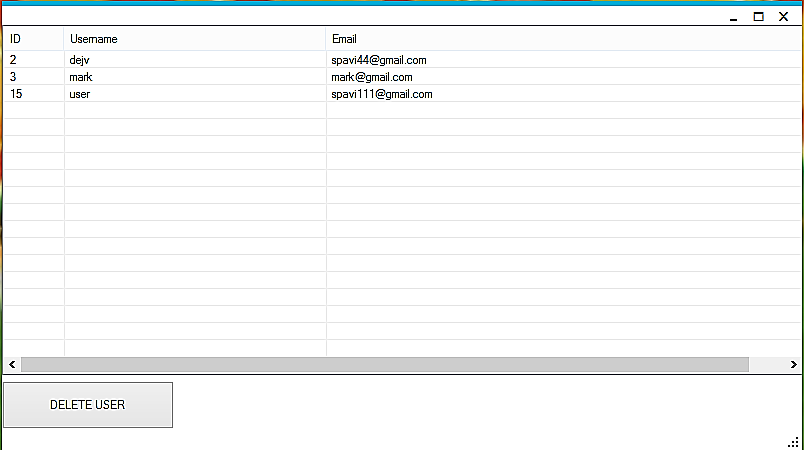
Slika 10: Forma ListaPoslužitelja

Lista „Popis poslužitelja“ sadrži kreirane poslužitelje koji su dostupni korisniku. Prikazani su detaljni podatci o imenu poslužitelja, podatci o vrsti igre koja će biti kreirana na tom poslužitelju te rejting korisnika koji je kreirao poslužitelja. U listi „Igrači na mreži“ nalaze se igrači 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 neuspješno povezuje 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, može doći 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. Ta forma omogućuje administratoru brisanje željenih korisnika iz baze podataka. Forma ListaIgrača prikazana je na slici 11.

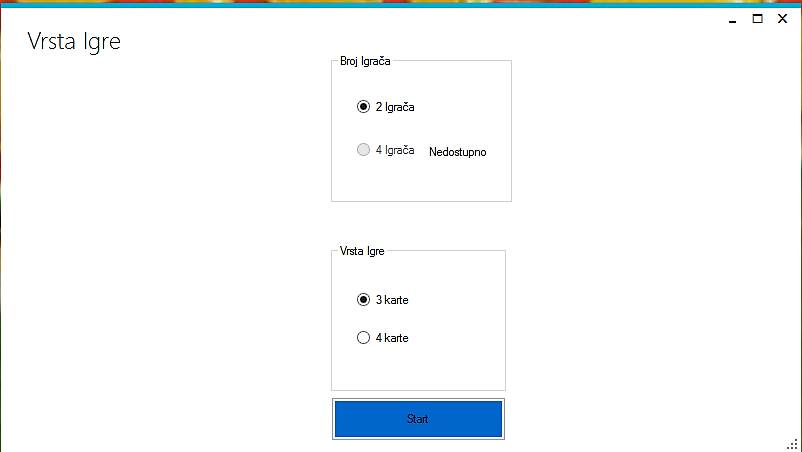


Slika 11: Forma ListaIgrača

### **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 promijeni, koriste se zadane 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 12.



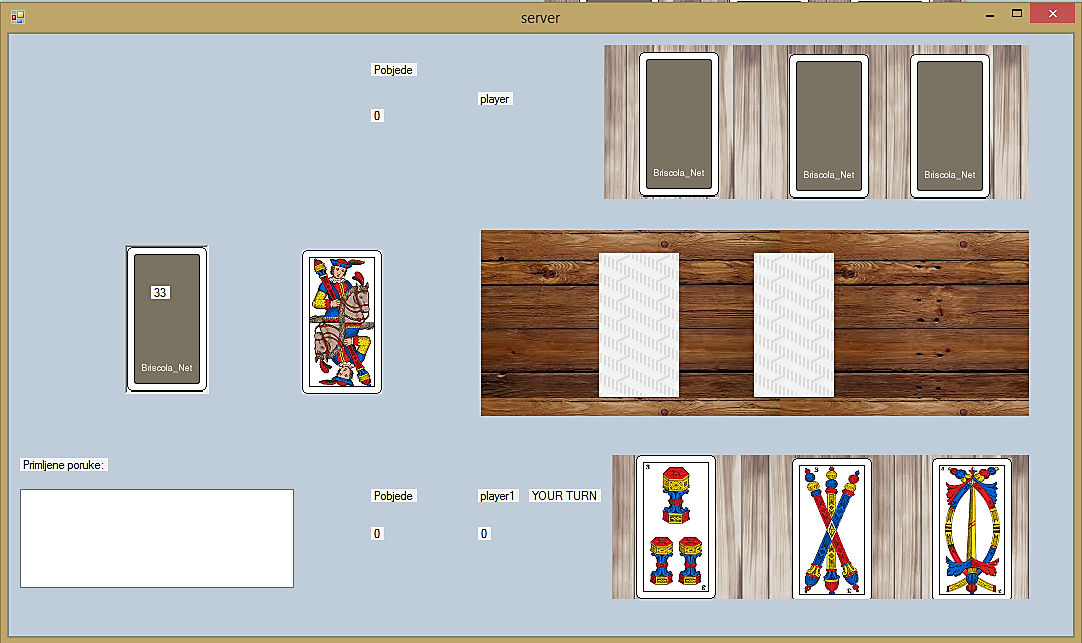
Slika 12: Forma VrstaIgre

### **3.2.9. Forma IgraTri**

U formi IgraTri odvija se igra briškula. Forma IgraTri sadrži okvire za slike (engl. *Picturebox*) koje sadrže odgovarajuću sliku za pojedinu karte. Panel Igrač1 sadrži karte igrača1. Za svaku kartu igrača1 postavljena je odgovarajuća slika u okviru za slike. Panel Igrač2 sadrži karte igrača2. Za svaku kartu igrača2 postavljena je pozadinska slika u okvir za slike. U okviru za slike *Igra* postavljena je karta u koju se igra. Okvir za slike *Špil* sadrži pozadinsku sliku s brojem preostalih karata. Lista *Received Messages* sadrži sve poruke koje je poslao klijent ili poslužitelj. 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.

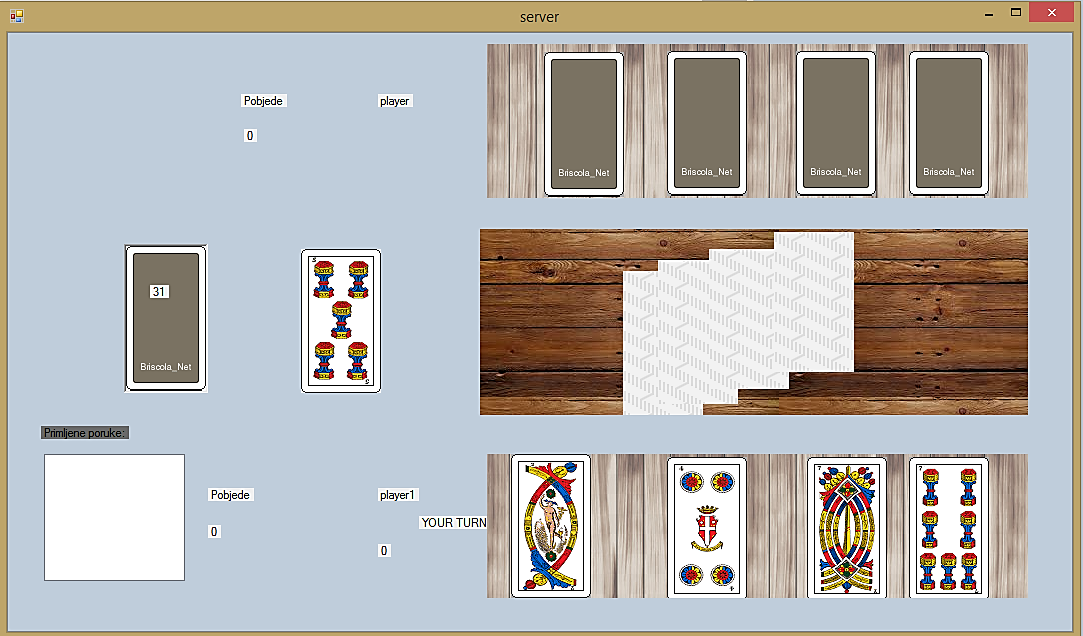
Forma IgraTri prikazana je na slici 13.



Slika 13: Forma IgraTri

### **3.2.10. Forma IgraČetiri**

Kako bi došlo do pokretanja forme IgrČetiri prilikom kreiranja poslužitelja u formi VrstaIgre u radijskom dugmetu VrstaIgre treba odabrati igra četiri karte. Forma IgraČetiri prikazana je na slici 14.



Slika 14: Forma IgraČetiri

Forma IgraČetiri sadrži okvire za slike koji sadrže odgovarajuću sliku za pojedinu kartu. Panel Igrač1 sadrži karte igrača1. Za svaku kartu igrača1 postavljena je odgovarajuća slika u okviru za slike. Panel Igrač2 sadrži karte igrača2. Za svaku kartu igrača2 postavljena je pozadinska slika u svaki okvir za sliku. U panelu Igrač1 i Igrač2 nalaze se po četiri okvira za sliku.

U okviru za sliku *Igra* postavljena je karta u koju se igra. Okvir za sliku *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 (engl. *labels*) 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.

Glavna razlika između formi IgraČetiri i IgraTri je u implementaciji metode *pokupi()*.Kako bi se odredio pobjednik ruke treba napraviti usporedbu između četiri karte, za razliku od IgraTri gdje su se uspoređivale samo dvije karte.

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

U formama IgraTri i IgraČetiri implementirana je vg logika. Klikom na jedan od okvira za sliku (engl. *Picturebox*): *Picturebox1*, *Picturebox2*, *Picturebox3*, *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 okvira za sliku u kojem se aktivirao događaj, tj. na koji je igrač1 kliknuo mišom. U formi Klijent aktivira se događaj ClientMove te se u formi IgračTri 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");}} |

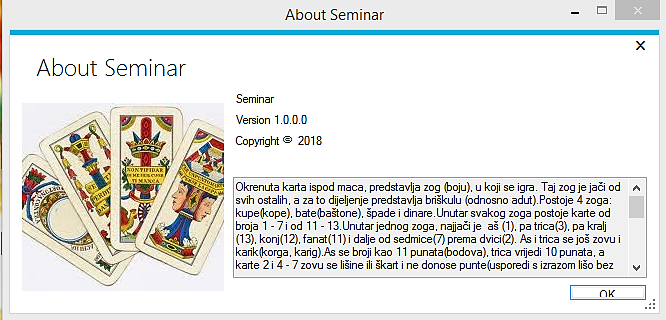
Ispis 11: Kôd OnMoveMade metoda

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 odigranoj ruci 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 listi pravila igre. Forma PravilaIgre prikazana je na slici 15.



**Slika 15:** Forma PravilaIgre

### **3.2.13. Forma IgraGotova**

Prikazuje se nakon završetka igre. Sadrži okvir za sliku, 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 Vrsta i Vrijednost, što omogućava jednostavniju izradu karte, a time i samoga špila.

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

Kako bi karte bile nasumično izmiješane prilikom svakog pokretanja nove igre u Špil klasi je implementirana metoda shuffle .

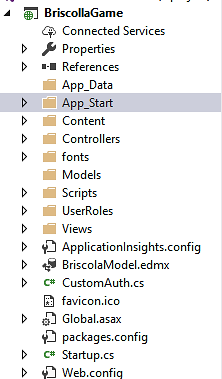
### **3.2.15. Klasa Igrač**

Klasa *Igrač* sadrži atribute ime, punti, red (engl. *turn*) te ruku u kojoj se nalaze karte.

## 3.3. Web aplikacija

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

Struktura *web* aplikacije prikazana je na slici 16.



Slika 16: 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. Neulogirani korisnik se pri pokretanju stranice preusmjerava na formu za prijave te može pristupiti stranici za registraciju, početnoj stranici i pravilima igre. Odgovarajuće akcije predznačene su atributom [AllowAnonymous], čime je omogućen pristup neautoriziranim korisnicima.

*Igračev pristup*

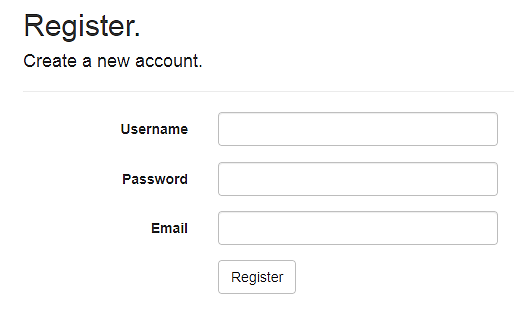
Igrač može pristupiti stranici Najbolji Igrači, Moja Statistika, Lista Igrača, Pravila Igre i Početna stranica. Za pristup metodama koje su predznačene atributom [CustomAuth(Roles="Player,Admin") , igrač mora biti autoriziran s ulogom Player ili Administrator.

*Administratorski pristup*

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

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

Forma za registraciju prikazana je na slici 17.



Slika 17: Forma za registraciju

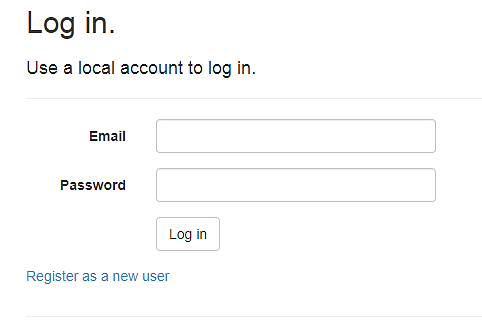
Nakon uspješne registracije korisnik može pristupi *web* i *desktop* aplikaciji. Forma za registraciju sastoji se od polja za korisničko ime, lozinku i e-poštu. Formu se može jednostavno proširiti s nekim od dodatnih polja, kao sto su ponovni unos lozinke. Nakon klika na dugme *Register* provjerava se jesu li zadovoljeni uvjeti za sva tekstualna polja. Uvjeti za uspješnu registriraciju korisnika jesu sljedeći:

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

Ako je registracija neuspješna i ako nisu zadovoljeni istaknuti uvjeti, posjetitelj će biti preusmjeren na formu za registraciju te će se ispisati odgovarajuća poruka za polje koje nije zadovoljilo navedene kriterije.

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

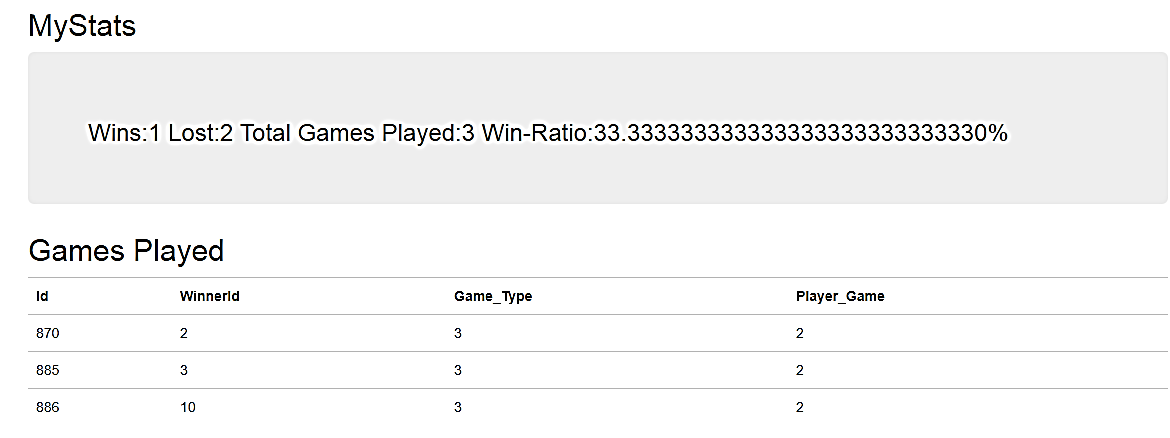
Forma za prijavu korisnika u *web* aplikaciju sastoji se od tekstualnih polja *Email* i dugmadi *Password*, *Log* *in* te poveznice *Register*. Prilikom pokretanja aplikacije posjetitelj se automatski preusmjerava na formu za prijavu ako nije prijavljen u aplikaciju. Također, posjetitelji koji ne posjeduju račun mogu preko poveznice pristupiti formi za registraciju. Forma za prijavu prikazana je na slici 18.



Slika 18: Forma za prijavu

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

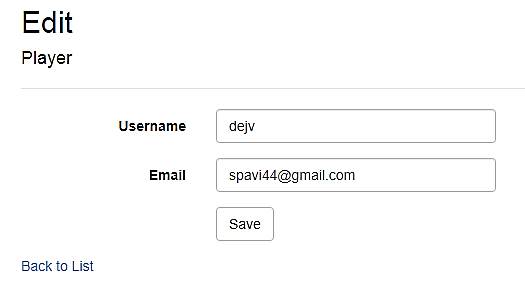
Moji rezultati prikazuje broj odigranih partija, broj pobjeda i poraza te ukupni postotak pobjeda trenutno prijavljenog korisnika. U listi *Odigrane igre* sadržana je kompletna povijest igara u kojima je korisnik sudjelovao te ishod pojedine igre. Moji rezultati prikazani su na slici 19.

****

Slika 19: Prikaz Moji rezultati

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

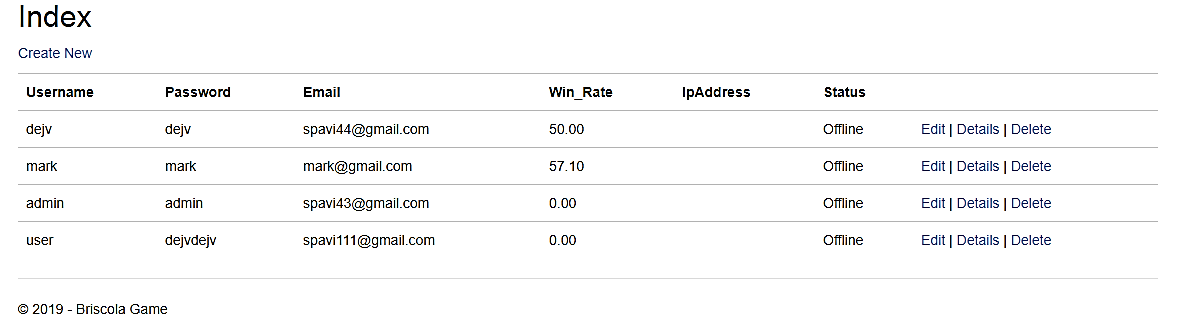
Logirani korisnici klikom na korisničko ime koje se nalazi u navigaciji otvaraju stranicu Account Manager koja im omogućava promjenu korisničkog imena i e-pošte. Klikom na dugme *Spremi* promjene se pohranjuju u bazu podataka. Account Manager prikazan je na slici 20.



Slika 20: Prikaz Account Manager

### **3.3.7**. **Forma Lista igrača**

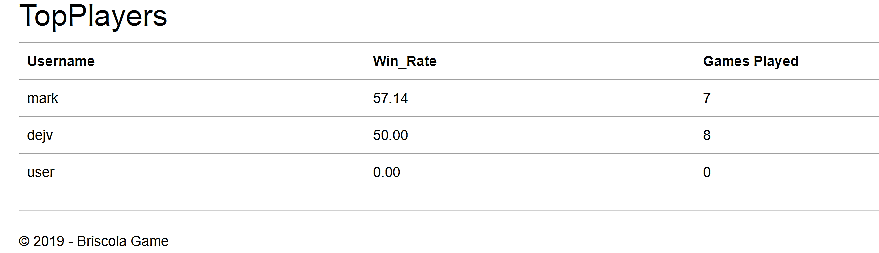
Forma Lista Igrača ima postavljena prava pristupa [Authorize(“Admin”)]. „Authorize“ se koristi kako bi se metodi kazalo da klijent mora biti autoriziran s ulogom „Admin“ kako bi mogao pristupiti. U formi Lista igrača sadržan je popis svih korisnika, s povjerljivim informacijama kao što su lozinka i IP adresa. Prilikom pokušaja pristupa bez administratorskih prava ispisuje se poruka o nedovoljnoj razini korisničkih prava. Lista igrača pokraj podataka svakog korisnika sadrži poveznice za promjenu podataka i brisanje korisnika. Preko forme Player List administrator može kreirati nove korisnike, brisati postojeće, pregledati detaljne informacije o korisniku te promijeniti korisnikov profil. Lista igrača prikazana je na slici 21.



Slika 21: Prikaz Lista igrača

### **3.3.8**. **Prikaz Najbolji igrači**

Najbolji igrači sadrži listu najbolje rangiranih igrača prema povijesti odigranih igara. Lista sadrži pet igrača, uz uvjet da ima najmanje pet korisničkih računa u bazi podataka. Ako je broj korisnika u bazi manji, lista će sadržavati sve korisnike koji se nalaze u bazi podataka. Najbolji Igrači prikazan je na slici 22.



Slika 22: 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. *Desktop* aplikacija izrađena je korištenjem Windows formi, dok je *web* aplikacija izrađena korištenjem aplikacijskog okvira ASP .NET MVC 5. *Desktop* i *web* aplikacija u pozadini koriste bazu podataka „Player“ koja se nalazi na Microsoft Azure SQL poslužitelju.

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.

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. Izbor programskog jezika sveo se na C# zbog njegove jednostavnosti te velike lepeze mogućnosti. 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 itd., 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. 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%20(11.%207.%202019.))
5. ENTITY FRAMEWORK za data-orijentisane aplikacije, <https://www.helloworld.rs/blog/ENTITY-FRAMEWORK-za-data-orijentisane-aplikacije/289> (posjećeno 25. 6. 2019.)
6. 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.))
7. Štefanac, M.: *Web* sustav za upravljanje dokumentima temeljem polustrukturiranih baza podataka (Diplomski rad), Fakultet organizacije i informatike Varaždin, Varaždin, 2016, [https://bib.irb.hr/datoteka/835099.1-mirstefan.pdf (posjećeno 21. 7. 2019.)](https://bib.irb.hr/datoteka/835099.1-mirstefan.pdf%20(posjećeno%2021.%207.%202019.))
8. JSON, [https://www.webprogramiranje.org/json/ (posjećeno 20. 7. 2019.)](https://www.webprogramiranje.org/json/%20(20.%207.%202019.))
9. 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.)
10. 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.))
11. Json Creation Part 1, <https://www.softwaretestinghelp.com/create-json-objects-using-c/> (posjećeno 24. 6. 2019.)
12. Socket programing in C#, <https://www.geeksforgeeks.org/socket-programming-in-c-sharp/> (posjećeno 26. 6. 2019.)
13. Socket programing in C#, <https://www.c-sharpcorner.com/article/socket-programming-in-C-Sharp/> (posjećeno 12. 6. 2019.)
14. Microsoft docs, <https://docs.microsoft.com/en-us/previousversions/visualstudio/visual-studio-2008/dd30h2yb(v%3dvs.90)> (posjećeno 14. 7. 2019.)
15. Entity framework, <https://arianscorner.wordpress.com/2014/04/04/designing-a-many-to-many-relationship-with-additional-fields-using-entity-framework/> (posjećeno 15. 7. 2019.)
16. Asp Net MVC Tutorials, <https://www.tutorialsteacher.com/mvc/asp.net-mvc-tutorials> (posjećeno 10. 7. 2019.)
17. Entity Framework Tutorial, <https://www.entityframeworktutorial.net/> 9 (posjećeno 20. 7. 2019.)

# 6. POPIS SLIKA I ISPISA

**Popis slika:**

Slika 1:Mapiranje u EF-u........................................................................................................5

[Slika 2: Prva baza modela EF-ka 8](#_Toc15816104)

[Slika 3: Tablica Korisnik 8](#_Toc15816105)

[Slika 4: Relacijski model baze podataka 9](#_Toc15816106)

[Slika 5: Struktura *desktop* aplikacije 11](#_Toc15816107)

[Slika 6: Forma za prijavu 12](#_Toc15816108)

[Slika 7: Forma PoslužiteljKlijent s administratorskim pravima 13](#_Toc15816109)

[Slika 8: Forma Poslužitelj 14](#_Toc15816110)

[Slika 9: Forma Klijent 17](#_Toc15816111)

[Slika 10: Forma ListaPoslužitelja 19](#_Toc15816112)

[Slika 11: Forma ListaIgrača 20](#_Toc15816113)

[Slika 12: Forma VrstaIgre 21](#_Toc15816114)

[Slika 13: Forma IgraTri 22](#_Toc15816115)

[Slika 14: Forma IgraČetiri 23](#_Toc15816116)

[Slika 15: Forma PravilaIgre 25](#_Toc15816117)

[Slika 16: Struktura *web* aplikacije 27](#_Toc15816118)

[Slika 17: Forma za registraciju 28](#_Toc15816119)

[Slika 18: Forma za prijavu 29](#_Toc15816120)

[Slika 19: Prikaz Moji rezultati 30](#_Toc15816121)

[Slika 20: Prikaz Account Manager 31](#_Toc15816122)

[Slika 21: Prikaz Lista igrača 31](#_Toc15816123)

[Slika 22: Prikaz Najbolji Igrači 32](#_Toc15816124)

**Popis ispisa:**

[Ispis 1: Kontekstna klasa 6](#_Toc15816142)

[Ispis 2: PlayersEntities1 konstruktor 10](#_Toc15816143)

[Ispis 3: String za konekciju s Microsoft Azure Sql serverom 10](#_Toc15816144)

[Ispis 4: Kôd za pretplatu na događaje SimpleTcpServer klase 15](#_Toc15816145)

[Ispis 5: Kôd JSON serijalizacija 16](#_Toc15816146)

[Ispis 6: Kôd pretplata na događaje 16](#_Toc15816147)

[Ispis 7: Kod za uklanjanje klijenta sa servera 16](#_Toc15816148)

[Ispis 8: Kôd Client pretplata na DataReceived događaj 17](#_Toc15816149)

[Ispis 9: Kod deserijalizacija objekta 18](#_Toc15816150)

[Ispis 10: Kod za odjavu s događaja 19](#_Toc15816151)

[Ispis 11: Kôd OnMoveMade metoda 24](#_Toc15816152)

[Ispis 12: Kôd pobrojane vrijednosti Vrsta i Vrijednost 26](#_Toc15816153)

[Ispis 13: Kôd setCards metoda 26](#_Toc15816154)

[Ispis 14: Kôd Upit LINQ-a 32](#_Toc15816155)