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

Projekt se sastoji od desktop aplikacije i web aplikacije. Desktop i web aplikacija koriste istu bazu podataka.

Baza podataka je izrađena u Microsoft SQL Management Studio 2017, te je postavljena na Microsoft Azure Sql server. Aplikacija omogućava korisnicima međusobno nadmetanje na računalima povezanim u lokalnu mrežu, dok za igru preko globalne mreže se korisnik treba sam pobrinuti.

Desktop aplikacija omogućava korisniku da kreira poslužitelja ili da kao klijent pristupi nekom od postojećih poslužitelja. Sama igra se odvija u desktop aplikaciji.Igra je realizirana tako da omogućava sudjelovanje dva korisnika, te se ishod igre pohranjuje u bazu podataka. Poslužitelj može kreirati igru s tri ili četiri karte. Aplikacija je sinkoronizirana s bazom podataka, čime je omogućeno da korisnik ima pregled na listu igraća koja je trenutno prijavljena u aplikaciju. Također korisnik ima pregled i svih poslužitelja koji su trenutno pokrenuti.

Web aplikacija omogućava adminu 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.

SUMMARY

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. The 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 helt 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 admin 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 vie web application.

1. UVOD

Cilj ovoga završnoga rada bio je izrada aplikacije koja će služiti u svrhu zabave korisnika. U tu svrhu izrađena je desktop aplikacija i web aplikacija. Tema projekta je multiplayer kartaška igra briškula.

Rad je podijeljen u tri poglavlja. U prvome poglavlju je opisan zadatak završnoga rada. U drugom poglavlju su pobliže opisane tehnologije koje su se koristile pri izradi desktop i web aplikacije. U trećem dijelu prikazan je razvoj desktop aplikacije korištenjem windows formi, te razvoj web aplikacije u MVC arhitekturi. Desktop i web aplikacija su izrađene u C# programskom jeziku.

2. TEHNOLOGIJE

2.1. ASP .NET MVC

Tehnologija Microsoft .NET okvira koja se koristi za razvoj dinamičkih web stranica, interaktivnih web stranica, te web servisa.ASP stranice se izvršavaju na serverskoj strani te generiraju kôd u HTML, XML formatu koji se šalje desktop ili mobilnim preglednicima. MVC kodna arhitektura, koja je iako razvijana prvenstveno za desktop aplikacije danas je popularnija kôd izrada web aplikacija[1]. Osnovna ideja MVC arhitekture je razdvajanje koda na tri cjeline[2]. MVC(model, pregled, kontroler) je razvojni okvir koji se koristi za izradu web aplikacija otvorenoga koda, a baziran je na MVC obrascu. Potpuno je neovisan o platformi te je definiran na posebnom asembleru System.Web.Mvc.ASP. .NET MVC arhitektura se sastoji od tri međusobno zavisne komponente: Model- u sloju modela je najčešće implementirana poslovna logika, tj. to je skup klasa koje opisuje podatke poslovne logike s kojom se radi. Model enkapsulira podatke koji se čuvaju u bazi podataka.

Pogled- omogućava prikaz podataka, i najčešće se koristi za prikaz podataka iz modela.

Upravitelj- 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. Entity Framework 6

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

- baza prva(engl. „*database first*“)

- kôd prvi (engl. „*code first*“)

Entity framework aplikacija se može pokrenuti na bilo kojem računalu koje ima instaliran .NET okvir. EF omogućava programerima da rade s podacima u obliku objekta umjesto tablicama. EF omogućava jednostavnu sinkronizaciju modela s bazom podataka. EF ima mogućnost da automatski generira veliki dio koda čime se značajno štedi vrijeme programeru.

Postoji više načina da se napravi ORP mapiranje, tj. postoje tri različita razvojna tijeka rada(engl. „*development workflowa*“):

1. baza prva- generira sve potrebne objekte u kodu na osnovi modela baze podataka.

2. kôd prvi- generira sve potrebne bazne objekte na osnovi postojećeg koda. Omogućava kreiranje klasa bez GUI(engl. „*Graphical User Interface*“) dizajnera ili .edmx fajla. Korištenje ovoga pristupa se preproučiva ako ne postoji baza podataka.

3. Model prvi- definira entitete i modele na osnovi koji će se generirati baza podataka i klase[5].

Kao most između entity klasa i baze podataka koristi se kontekstna klasa(engl. „*dbcontext*“). Izgled kontekstne klase prikazan je na Slici 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; }}
```

Slika 1. Kontekstna klasa

2.3. LINQ(engl. „Language Integrated Query“)

Prvi puta je predstavljen u .NET 3.5 verziji te u Visual Studio 2008. Prednosti LINQ-a su što omogućava skupini .NET jezika(C# , F#, Visual Basic) jednostavniju izradu upita prema bazi podataka[6]. LINQ sadrži oko pedeset operatora upita pomoću kojih se značajno smanjuje vrijeme sortiranja, filtriranja, grupiranja podataka. Neki od najčešće korištenih operatora su:

- Sortiranje
- Filtriranje
- Grupiranje
- Spajanje
- Pretvorba

LINQ se može koristiti za sve vrste podataka koje su izvedeni iz IEnumreable sučelja. Uvođenje Linq-a omogućava jednostavnije pretraživanja podataka u bazi, čime se smanjuje količina potrebnoga koda te povećava čitljivost koda[7]. LINQ podržaje upite za različite tipove podataka kao što su relacijske baze, xml. LINQ sadrži dvije različite sintakse, a to su Query i Lambda. Za korištenje LINQ-a u C# potrebno je dodati System.Linq biblioteku.

2.4. JSON(engl. „JavaScript Object Notation“)

JSON-je tekstualni format čija je namjena prijenos podataka u formatu koji je čitljiv i ljudima i strojevima[8]. Ekstenzija koja se koristi za json datoteku je oblika .json.

Json zbog svojih prednosti nad xml-om sve više postaje prvi izbor. Xml koristi oznake zbog čega 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[9].

JSON ne ovisi o programskom jeziku,a njegova najveća primjena je u web aplikacijama. Json podržaje različite tipove podatak kao: tekst, logički podaci, broj, polja. Json ne može sadržavati funkciju.

2.5. Windows forme

Windows forma je alat koji se koriste za izradu Windows aplikacija.Windows forme su učinkovit i jednostavan način koji omogućava korisniku komunikaciju s programom. Window forma je prozor koji sadrži kontrole za prikaz, unos i manipulaciju podacima[9]. Za izradu windows formi potrebno je unutar Visual Studia odabrati File->New projekt te se u izborniku odabire windows forma,upiše se željeno ime te se pritiskom na uredu (engl. „*OK*“) gumb kreira nova windows form aplikacija.

Alati(engl. „*Toolbox*“) sadrže skup gotovih windows kontrola[9]. Alati sadrže kontrole koje se mogu dodati windows forms aplikacijama,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*“) nam omogućavaju da podesimo postavke za željeni element, a dijele se na devet kategorija: izgled(engl. „*Appearance*“), ponašanje(engl. „*Behavior*“), podaci(engl. „*Data*“), pristupačnost(engl. „*Accessibility*“), dizajn(engl. „*Design*“), fokus(engl. „*Focus*“), predložak(engl. „*Layout*“),

stil prozora(engl. „*Windows Style*“) i različito(engl. „*Misc*“). Istraživač rješenja(engl. „*Solution Explorer*“) koristi se za prikaz strukture projekta.

2.6. C#

C# je u potpunosti objektno-orijentiran programski jezik. C# podržava koncepte nasljeđivanja, enkapsulacije, polimorfrizma[10]. C# je tipiziran jezik, što znači da se mora obaviti deklaracija svakog stvorenog objekta, a prevoditelj će prijaviti pogrešku ako objektu nije pridružen odgovarajući tip podataka[11]. Najčešća primjena C# je kôd izrada Windows i web aplikacija. Svi tipovi podataka imaju određenu i nepromjenjivu veličinu npr.tip short se preslikava u int16. Varijablama je potrebno pridružiti vrijednost ili obaviti inicijalizaciju prije samog korištenja.

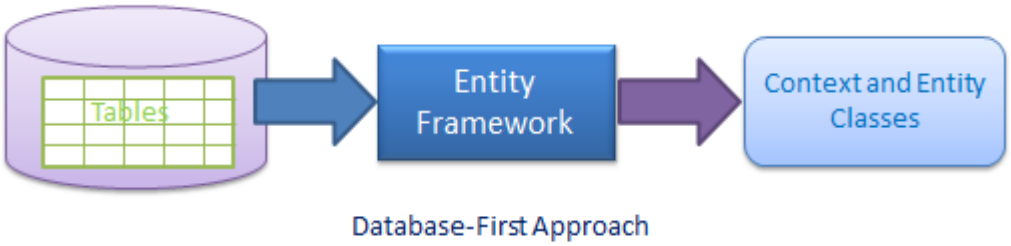
U C# vrši se podjela na dva tipa podataka: vrijednosne(bolean, char, integer, decimal, enum) i referentne (Klase, Sučelja, Delegati, Polja..)[9]. Osnovna razlika je u načinu pristupa, te što se vrijednosni tipovi pohranjuju na stogu, a referentni na heap-u. C# upravlja memorijom koristeći svoj sustav za skupljanje otpada(engl. „*Garbage Collector*“). Program se izvršava u CLR(engl. „*Common Language Runtime*“), što omogućava ponovno korištenje klase koja je napisana u nekom od .NET programskih jezika. C# tekstualne datoteke sadrže ekstenzije .cs .

3. APLIKACIJA

3.1. Baza podataka

Nakon završetka izrade same igre, tj. formi IgraTri i IgraČetiri unutar kojih se pokreće igra, bilo je potrebno osmisliti model i izgled baze podataka. Baza podataka se koristi za pohranjivanje podataka registriranih korisnika, te za pohranjivanje ishoda igre. Iz ER(skraćenica engl. „*Entity Relationship*“) dijagrama se vidi da se model podataka sastoji od ukupno pet entiteta. Relacije između tablica su prikazane na slici 4.

Za izradu baze podataka koristio je se baza prva pristup, kôd kojeg se prvo modelira baza podataka, na temelju koje će EF automatski generirati potrebne modele. Na slici 2. je prikazan baza prva model EF-ka.

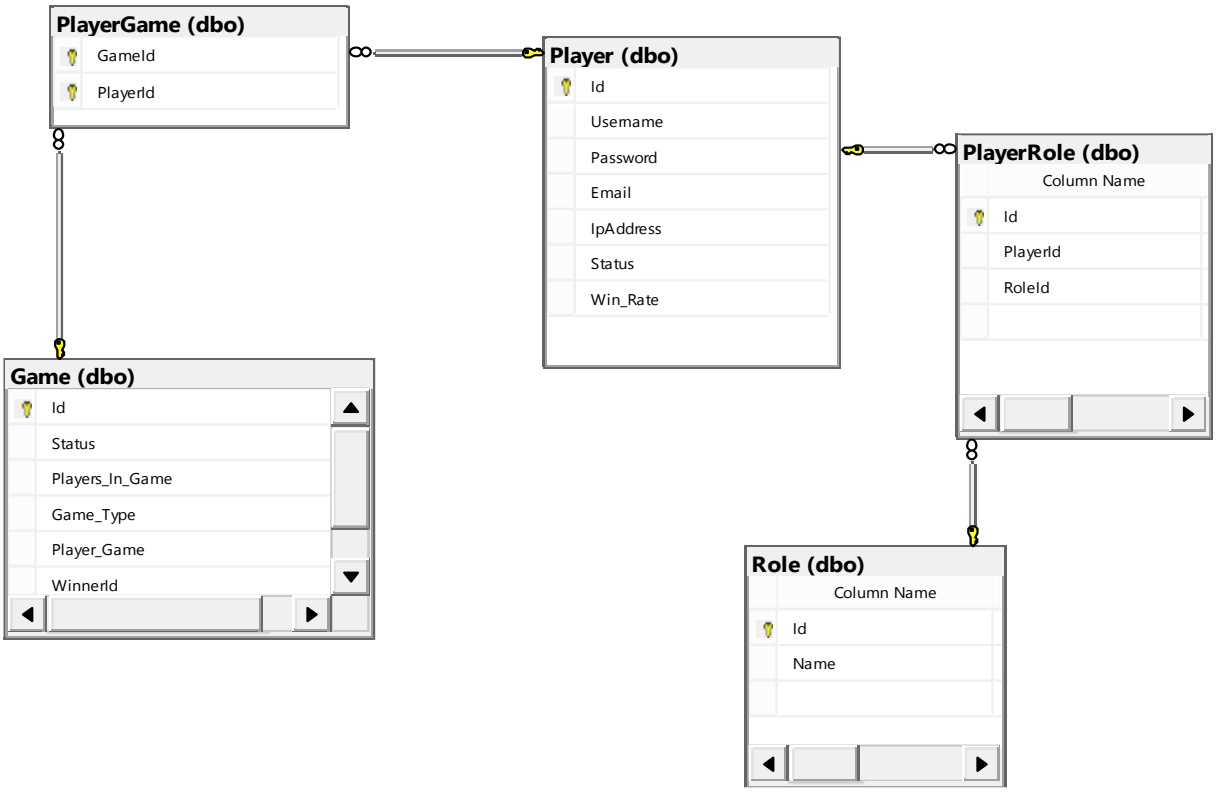


Slika 2. EF baza prva schema

Baza podataka je izrađena pomoću Microsoft SQL Server Management Studio 2017 alata. Nakon izrade baze podataka sljedeći korak je izrada pojedinih tablica. Primjer izrađene tablice je vidljiv na Slici 3.

	Column Name	Data Type	Allow Nulls
🔑	Id	int	☐
	Username	varchar(20)	☐
	Password	varchar(20)	☐
	Email	varchar(255)	☑
	IpAddress	varchar(16)	☑
	Status	ntext	☑
	Win_Rate	decimal(4, 1)	☑
			☐

Slika 3. Tablica Korisnik



Slika 4. Relacijski model baze podataka

Slika 4. prikazuje EER model baze podataka koja sadrži tablice s odgovarajućim relacijama između pojedinih tablica. Relacije između tablica važne su za lakšu manipulaciju samim podacima, te pravilan rad baze podataka.

Podaci o korisniku (engl. „*User*“),korisničkim ulogama(engl. „*role*“), igri(engl. „*game*“), pohranjeni su u bazi podataka Igrači(engl. „*Players*“). Baza podataka je kreirana pomoću alata Microsoft SQL Express Server a postavljena je na Microsoft Azure Sql poslužitelj. Da bi mogli manipulirati podacima koji se nalaze u bazi podataka potrebno je povezati aplikaciju s bazom. DBContext upravlja vezom s bazom, te se po potrebi spaja ili odspaja s baze. DBContext konstruktor(engl. „*constructor*“) kao parametar može primiti ime baze na koju se spaja ili string za vezu s bazom.

```
public partial class PlayersEntities1 : DbContext
{
    public PlayersEntities1(): base("name=PlayersEntities1")
 {}}
```

Slika 5.DBContext konstruktor

Da bi se ostvarila veza s Microsoft Azure SQL serverom potrebno je podesiti string za vezu s bazom. String za vezu s bazom se kreira automatski te je podešen za vezu s bazom podataka koja se nalazi na lokalnom računalu. Slika 6. prikazuje kod ispravno podešenog stringa za povezivanje s Microsoft azure serverom.

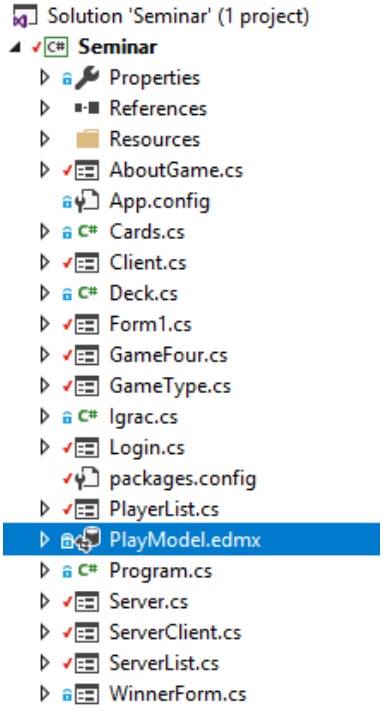
```
metadata=res://*/PlayModel.csdl|res://*/PlayModel.ssdl|res://*/PlayModel.msl;
provider=System.Data.SqlClient;provider connection string="data
source=briscola.database.windows.net;initial catalog=Players;user id=želejno_korisničko
ime;password=željena_šifra;MultipleActiveResultSets=True;App=EntityFramework"
```

Slika 6. String za konekciju s Microsoft Azure Sql serverom

3.2. Desktop aplikacija

U praktičnom dijelu rada kroz desktop aplikaciju demonstrirana je upotreba windows formi, entity okvira, json i tcp tehnologije. Aplikacija sadrži jedanaest formi te tri klase.

3.2.1. Struktura desktop aplikacije



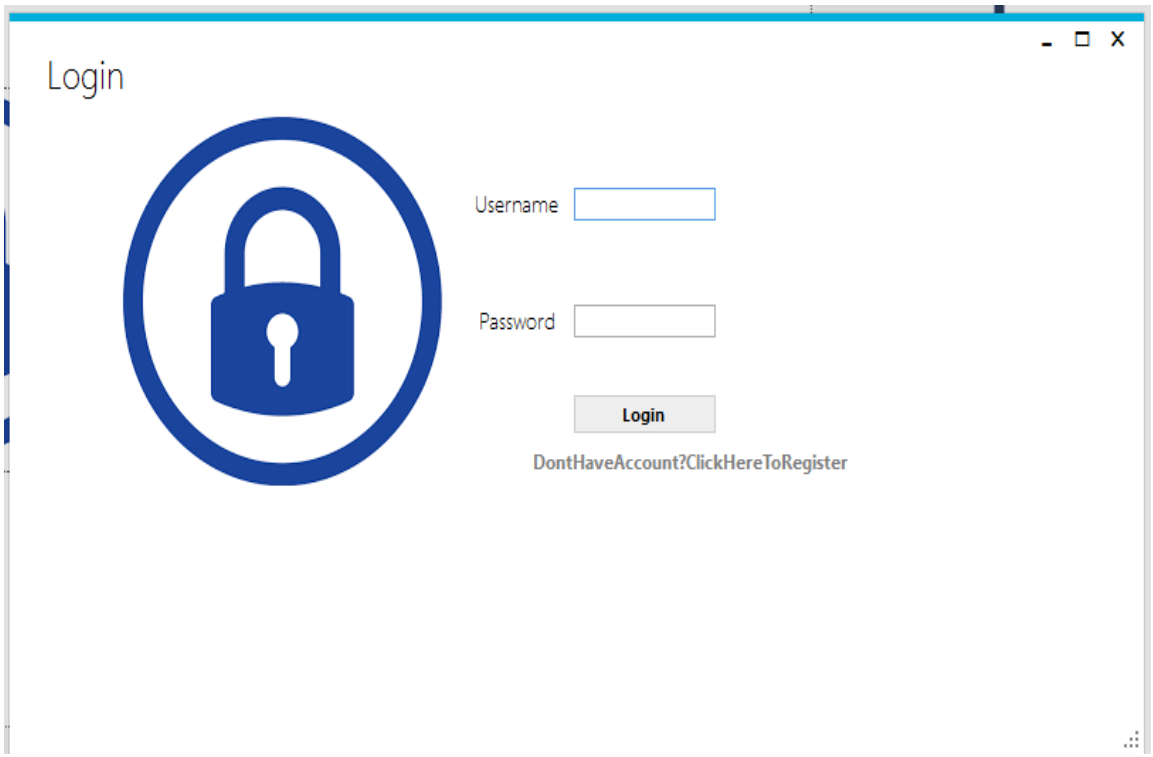
Slika 7. Struktura desktop aplikacije

3.2.2. Forma za prijavu

Pokretanjem aplikacije prikazuje se forma za prijavu (engl. „*Login*“) koja omogućuje prijavu za postojeće korisnike.Forma za prijavu se sastoji od dva tekstualna polja(engl. „*TextBox*“), gumba(engl. „*button*“), 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 PoslužiteljKlijent forma. Status korisnika se mijenja iz nije povezan na mrežu (engl. „*Offline*“) u na mreži(engl. „*Online*“) te se promjene pohranjuju u bazu podataka.

.



Slika 8. Forma za prijavu

3.2.3. PoslužiteljKlijent forma

PoslužiteljKlijent forma sadrži glavni izbornik.Izbornik u PoslužiteljKlijent formi omogućuje navigaciju korisniku. Izgled PoslužiteljKlijent forme je ovisan o pravima pristupa pojedinog korisnika. Postoje dvije razine prava pristupa desktop aplikaciji a to su admin pristup i korisnički pristup.

Kod korisničkog pristupa PoslužiteljKlijent formi izbornik sadrži sljedeće opcije:

- Poslužitelj gumb
- Vrsta Igre gumb
- Lista Poslužitelja gumb
- Pravila Igre gumb

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

- 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. U slučaju da u bazi podataka ne postoji toliko korisničkih računa, prikazat će se svi korisnici.

- Briškola Web- klikom misa otvara se početna stranica web aplikacije „Briškula“ koja je detaljnije opisana u poglavlju 3.3

Klikom miša na neki od ponuđenih opcija otvara se odabrana forma. PoslužiteljKlijent Forma prelazi u pozadinu te njene kontrole više nisu vidljive korisniku sve dok se trenutna forma ne zatvori.

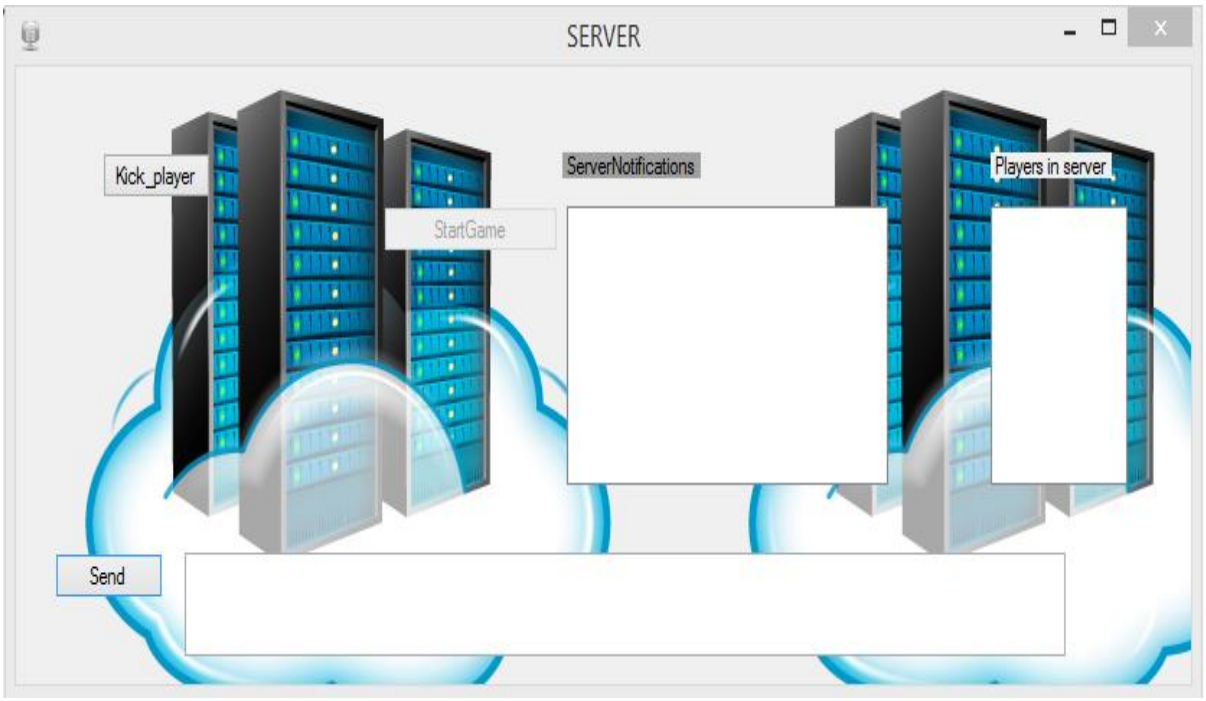
Slika 9. PoslužiteljKlijent forma sa admin pravima

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

- lista igraća gumb- klikom miša pokreće se forma lista igrača

- resetiranje baze gumb: klikom miša na gumb resetiraju se vrijednosti u tablici igra(engl. „*game*“), tj. uklanjaju se sve igre iz baze podataka, te se uklanjaju i svi trenutno pokrenuti poslužitelji.

3.2.4. Poslužitelj forma



Slika 10. Poslužitelj forma

Pokretanjem Poslužitelj forme korisnik kreira poslužitelj (engl. „*server*“). Dohvaća se ip adresa korisnika, zapisuje se u bazu podataka, te se promjene pohranjuju. Pokretanjem poslužitelja kreira se i nova igra, te se dodaje u bazu podataka. Ako se prilikom kreiranja poslužitelja ne promjene postavke u PostavkeIgre formi, kreirat će se igra s unaprijed zadanim vrijednostima:

- broj igrača: 2 igrača

- broj karata: 3 karte

Nakon starta poslužitelj na portu 7000 sluša zahtjeve klijenta za spajanje. Lista Igrači na poslužitelju sadrži popis korisnika koji se trenutno spojeni na poslužitelj. Lista klijenata spojenih na pristupnik se ažurira prilikom povezivanja klijenta na poslužitelj ili odspajanja sa poslužitelja. Poslužitelj ima mogućnost da odabranoga klijenta odspoji, klikom na gumb *Izbaci_igrača*, nakon čega će klijent biti preusmjeren na ListaIgrača formu, te će biti uklonjen iz liste igrači na poslužitelju te se promjene pohranjuju u bazu. U listi obavijesti se nalaze sve poruke poslane od strane klijenta.

Klikom na gumb pošalji (engl. „*send*“) šalje se poruka svim klijentima spojenim na poslužitelj. Poruka sadrži tekst koji se nalazi u tekstualnom polju (engl. „*TextBox*“). Da bi se omogućila komunikacija s klijentom, obavlja se pretplata (engl. „*subscribe*“) na događaj podaci primjeni (engl. „*DataReceived*“) iz SimpleTcpServer klase.

```
tcpServer = new SimpleTcpServer();

        tcpServer.Delimiter = 0x13;

        tcpServer.StringEncoder = Encoding.ASCII;

        tcpServer.DataReceived += Server_Recived;

        tcpServer.ClientConnected += Client_Connected;

        tcpServer.ClientDisconnected += Client_Disconnect;
```

Slika 11. Kôd pretplata na događaje SimpleTcpServer klase

U prikazanom kodu na slici 11. vrši se pretplata na događaje klijent povezan (engl. „ *Client Connected*“) te klijent odspojen (engl. „*ClientDisconnected*“) koje se nalaze u klasi SimpleTcpServer.Da bi poslužitelj bio obaviješten o spajanju klijenta na poslužitelj vrši se pretplata na *Client Connected* događaj. Za obavijest o odspajanju klijenta sa poslužitelja vrši se pretplata na *ClientDistonnected* događaj. Ako dođe do odspajanja klijenta s poslužitelja, poslužitelj će biti obaviješten o izlasku klijenta porukom.

Maksimalni broj klijenata koji mogu pristupiti poslužitelju se postavlja u Vrsta Igre formi, a po unaprijed zadanoj vrijednosti je postavljen na jednoga klijenta. Nakon spajanja ili odspajanja klijenta na poslužitelj, ažurira se broj klijenata na poslužitelju te se promjene zapisuju u bazu podataka te pohranjuju. Dok klijent ne klikne na spreman (engl. „*ready*“) gumb, *Start* gumb je onemogućen te poslužitelj ne može pokrenuti igru.

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

JsonObject(MemberSerialization.OptIn) izvan deklaracije klase nam omogućava da korištenjem [JsonProperty] iznad deklaracije atributa odabiremo atribute klase koje želimo serijalizirati.

```
json = JsonConvert.SerializeObject(f);

tcpServer.BroadcastLine(json);
```

Slika 12.Kod Json serializacija

Na slici 12. prikazana je serijalizacija objekta IgraTri, te slanje serijaliziranoga teksta klijentima spojenim na poslužitelj. Nakon slanja json stringa klijentima koji su spojeni na poslužitelj izvršava se pretplata na sljedeće 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;
```

, Slika 13.Kod Pretplata na događaje

Na slici 13. prikazan je kôd u kojem se u poslužitelj formi 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. Nakon aktivacije događaja(engl. „*eventa*“), poslužitelj obavještava klijenta o završetku igre, te se poziva metoda NewGame(s). Metoda NewGame(s) se koristi za provjeru ishoda igre te dok se ne dobije konačni pobjednik igre, prikazuje se dialog result gdje se nudi mogućnost nastavka igre.

U slučaju da je odgovor Ne (engl. „*No*”) od strane klijenta ili poslužitelja, klijent će biti odspojen s poslužitelja. Za poraženog smatra se onoga tko je prvi odgovorio Ne. Na slici 14. je prikazan kôd koji se treba izvršiti u Poslužitelj formi kako bi se klijent uspješno odspojio s poslužitelja.

```
OnFormUnsub("Form");

tcpServer.BroadcastLine("kicked");

Client_Disconnect(null, tcpClient);
```

Slika 14. Klijent uklonjen sa servera kôd

U slučaju da su klijent i poslužitelj odgovorili Da (engl. „*Yes*”), pokreče se nova igra. Metoda *OnFormUnsub* se poziva prilikom ponovnoga pokretanja igre a služi za odjavu Poslužitelj forme s pretplate svih događaja. Ukoliko se odjava s događaja ne bi izvršila, prilikom pokretanje nove igre, dolazi do povećanja pretplate na sve događaje. Što u konačnici rezultira neispravnim radom same aplikacije.

3.2.5. Klijent Forma

Slika 15. Klijent Forma

Nakon uspješnoga povezivanja s poslužiteljom otvara se Klijent forma. Klijent ima mogućnost napuštanja poslužitelja pritiskom na povratak(engl. „*return*“) gumb. Klikom na povratak gumb, client forma se zatvara, te se ponovno pokreće forma Lista Servera. Klikom na pošalji(engl. „*send*“) gumb, klijent šalje poruku poslužitelju, a poruka sadrži tekst koji se nalazi u tekstualnom prikazu. Kada je klijent spreman za pokretanje igre, klikne spreman(engl. „*ready*“) gumb te čeka na odgovor od strane poslužitelja. U listi poruke nalaze se poruke razmijenjene s poslužiteljem. Da bi komunikacija s poslužiteljom bila moguća klijent se mora pretplatiti na događaj iz SimpleTcpClient klase.

```
client=new SimpleTcpClient();
client.StringEncoder = Encoding.UTF8;
client.DataReceived += Client_Received;
```

Slika 16. Kôd Client pretplata na DataReceived događaj

U klijent form se obavlja deserijalizacija objekta tipa IgraTri ili IgraČetiri ovisno o vrste igre koju je poslužitelj kreirao. String je prethodno serijaliziran od strane poslužitelja te poslan klijentu korištenjem tcp protokola.

```
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();}}
```

Slika 17. Deserijalizacija objekta kôd

Deserijalizaciju je nužno izvršiti unutar „*BeginInvoke(MethodInvoker)*“ funkcije, u suprotnom dolazi do greške tijekom izvođenja(engl. „*runtime error*“). IgraTri ili IgraČetiri pokušat obaviti inicijalizaciju elemenata prije nego se završi čitava deserijalizacija stringa. Nakon deserijalizacije objekta potrebno je izvršiti pretplatu na događaje.

```
f.MouseClicked+=this.OnMOuseClicked;
this.ClientMove += f.OnClientMove;
client.DataReceived += f.GameRecved;
f.FormClosing += this.Game_Form_Closing;
```

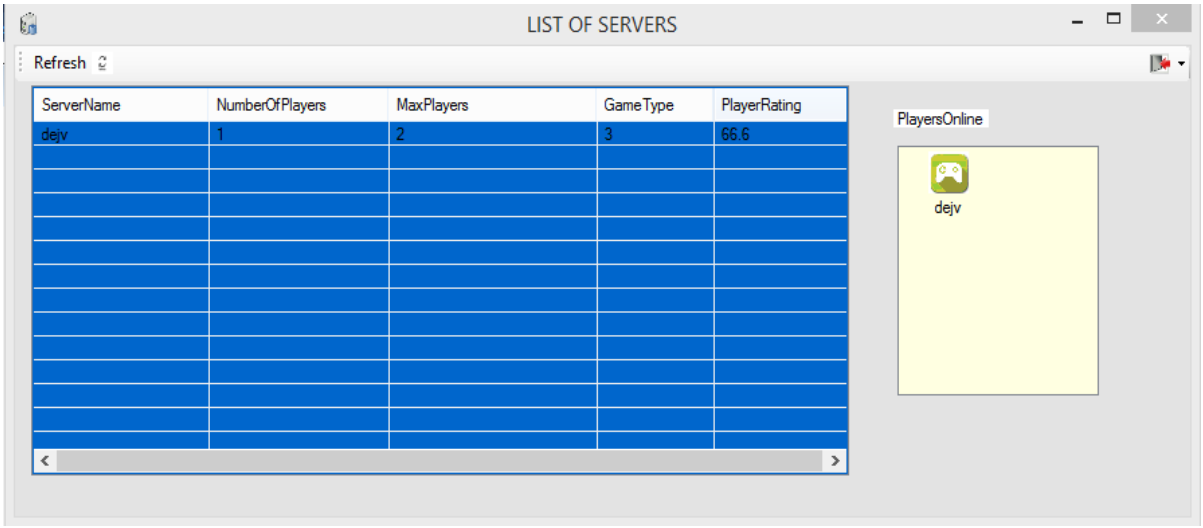
Slika 18. Kôd Pretplata na događaje u Klijent Formi

Po završetku igre poziva se metoda *OnFormUnsub*(). U metodi se vrši odjava s pretplaćenih događaja. Bez odjave sa pretplate pri svakom pokretanju nove igre dolazilo da povećanja pretplate na isti događaj.

```
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();}}
```

Slika 19.Odjava sa događaja kod

3.2.6. Lista Poslužitelja forma



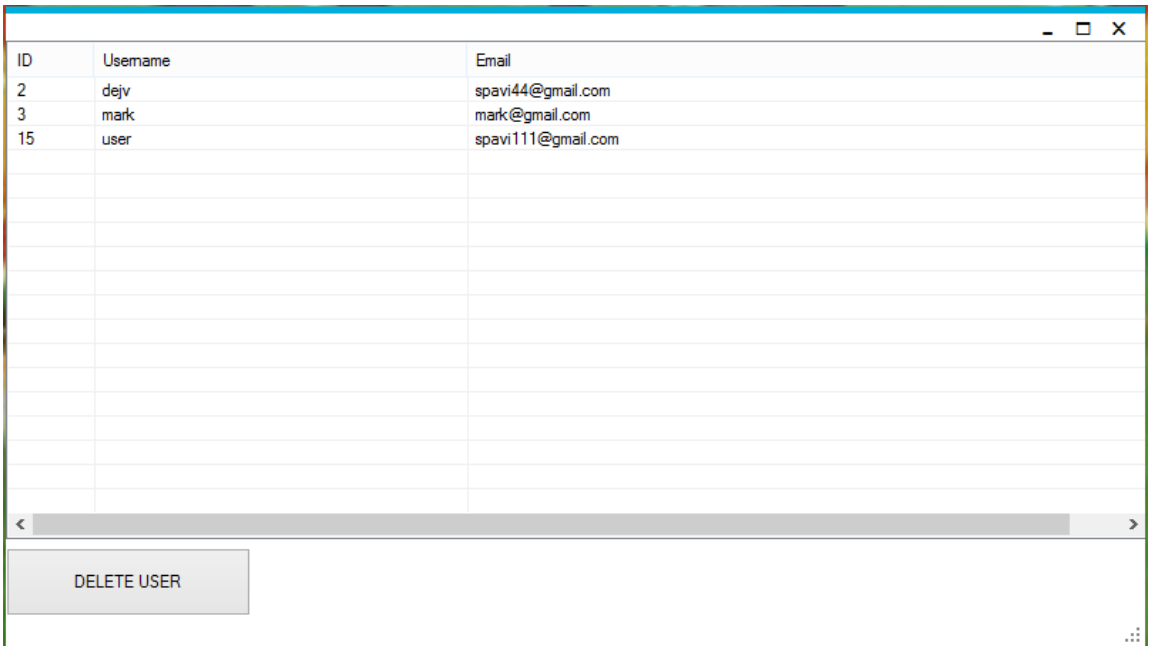
Slika 20. ListaPoslužitelja forma

Podaci su sadržani u listi popis poslužitelja. Lista sadrži kreirane poslužitelje koji su dostupni korisniku. Prikazani su detaljni podatci o imenu poslužitelja, podaci o vrsti igre koja će biti kreirana na tom poslužitelju,rejting korisnika koji je kreirao poslužitelja. U listi *Igrači na mreži* nalaze se igrači koji su trenutno prijavljeni u aplikaciji. Osvježi(engl. „*Refresh*“) gumb omogućava korisniku sinkronizaciju s bazom, tj. da vidi trenutno stanje u listi popis poslužitelja.

Korisnik odabire željeni poslužitelj te dvoklikom miša pokušava pristupiti poslužitelju. U slučaju uspješnoga povezivanja na poslužitelj otvara se Klijent forma, dok forma ListaPoslužitelja prelazi u pozadinu i njene kontrole više nisu vidljive korisniku. Do neuspješnoga povezivanja može doći u slučaju da je poslužitelj popunjen, tada će se pojaviti dialog s odgovarajućom porukom. Ako dođe do pogreške zbog problema s vezom, tj. klijent se ne može spojiti na poslužitelj, aplikacija će tada obavijestiti korisnika s porukom koja sadrži odgovarajuću iznimku.

3.2.7. Lista Igrača forma

Lista Igrača formi je moguće pristupiti samo s ulogom admina. U listi korisnici nalaze se svi korisnici s njihovim osnovnim informacijama. Klikom na gumb *Delete User* iz baze se uklanja odabrani korisnik. Ova forma omogućuje adminu brisanje željenih korisnika iz baze podataka.

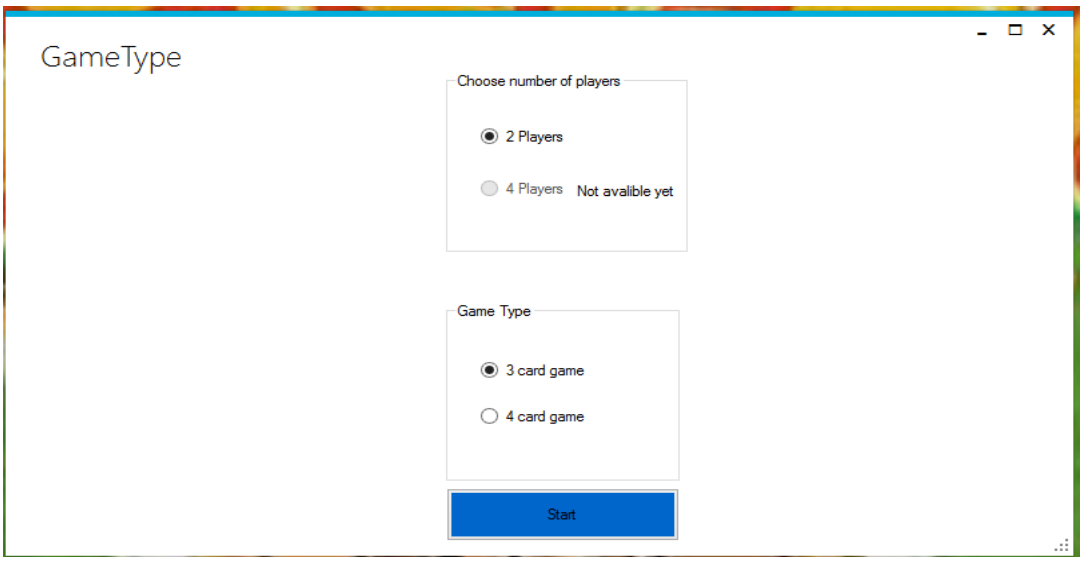


Slika 21. ListaIgrača forma

3.2.8. Vrsta Igre Forma

Vrsta Igre omogućuje da se prilikom izrade poslužitelja namjeste željene postavke igre koja će se pokretati na tome poslužitelju. Opcijski gumbi (eng. „*radio button*“) Broj igrača sadrže mogućnost odabira maksimalnoga broja korisnika koji mogu pristupiti poslužitelju. Trenutno se ne može mijenjati, odnosno igra se može odvijati samo između dva korisnika.

Opcijski gumbi Vrsta Igre sadrže mogućnost odabira igre s tri ili s četiri karte. U slučaju da se Opcijski gumbi ne promijene koriste se tvorničke unaprijed zadane postavke:2 igrača i 3 karte. Klikom miša na gumb Pokreni, pokreće se Poslužitelj forma dok forma VrstaIgre prelazi u pozadinu.



Slika 22. VrstaIgre forma

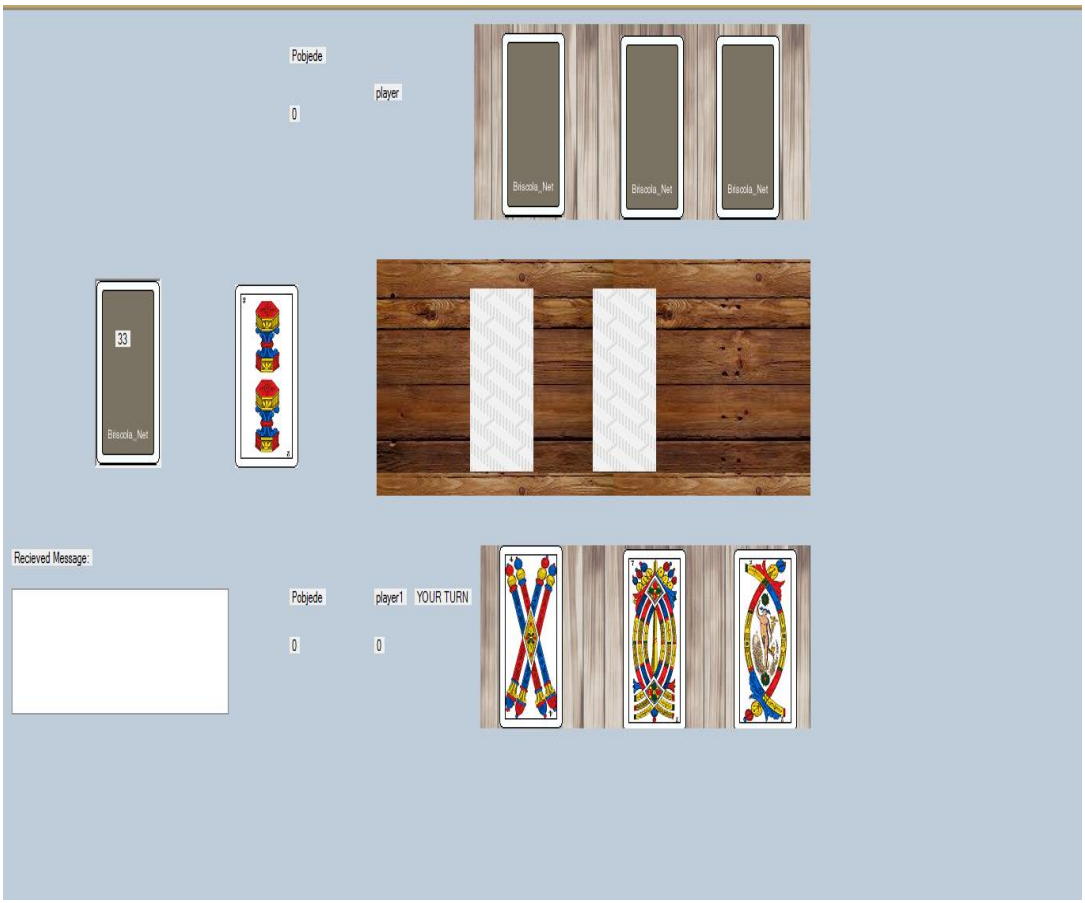
3.2.9. IgraTri Forma

U formi IgraTri se odvija 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 primljene od strane klijenta ili poslužitelja. Forma sadrži naslove(engl. „*labels*“) koji omogućavaju korisniku lakše praćenje tijeka igre. Neki od važnijih naslova su:

- Pobjede pokazuje broj pobjeda igrača1 i igrača2

- Punti prikazuje broj punata igrača1, dok punti protivnika nisu vidljivi

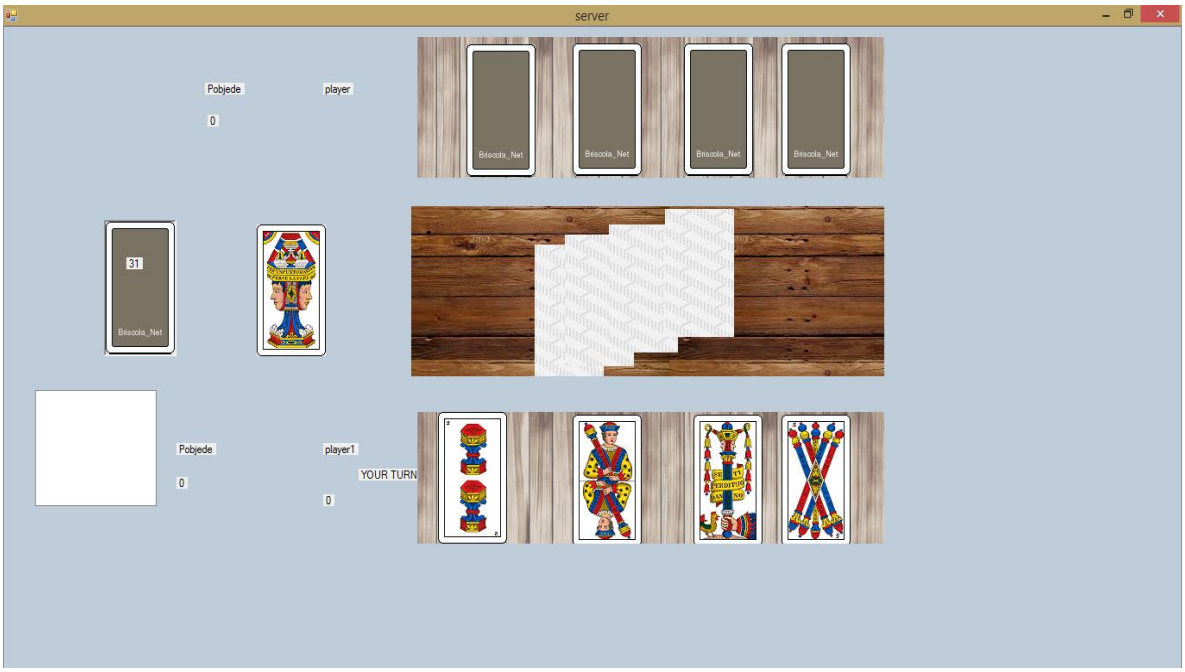
- Baci kartu pokazuje kojeg je igrača red da baci kartu



Slika 23. IgraTri forma

3.2.10. IgraČetiri Forma

Da bi došlo do pokretanja ove forme potrebno je prilikom kreiranja poslužitelja u VrstaIgre formi u opcijskom gumbu *VrstaIgre* odabrati 4 karte igra.



Slika 24. IgraČetiri Forma

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 se nalaze 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 primljene poruke(engl. „*Receieved Messages*“) sadrži sve poruke primljene od strane klijenta ili poslužitelja. U ovoj 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.

Formi sadrži naslove(engl. „*labels*“) koji omogućavaju korisniku lakše praćenje tijeka igre. Neki od važnijih naslova su:

- 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 implementacoji metode *pokupi()***.**Da bi se odredio pobjednik ruke potrebno napraviti usporedbu između 4 karte za razliku od IgraTri gdje su se uspoređivale samo dvije karte.

3.2.11. Multiplayer logika

U formama IgraTri i IgraČetiri je implementirana „multiplayer“ logika.

Klikom na jedan od okvira za sliku (engl. „*Picturebox*“): „*Picturebox1“*, „*Picturebox2“* , „*Picturebox3“*, „*Picturebox4“* aktivira se događaj *Mouse Clicked*, te se izvršava metoda *OnMouseClicked*. *OnMouseClicked* sadrži različitu implementaciju u Poslužitelj i Klijent formi. 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 Klijent formi se aktivira događaj ClientMove te se u formi IgračTri izvršava metoda `OnClientMove()`. Implementacija metode `OnClientMove()` u IgraTri formi je prikazana na Slici 25. Na slici je prikazan kôd, gdje se vidi koji okvir za sliku će se aktivirati na klijent strani u ovisnosti o okviru za sliku koji je aktiviran na poslužitelj 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");}}
```

Slika 25. Kôd OnMoveMade metoda

U slučaju da se događaj *MouseClicked* aktivirao na klijent strani, šalje se poruka poslužitelju. U Poslužitelj formi se aktivira događaj *MoveMade* te se u formi IgračTri izvršava metoda OnMoveMade. Za računanje broja punti poslužitelja i klijenta, odnosno igrača1 i igrača2 implementirana je metoda *Punti*. Metoda kao parametar prima karte koje se nalaze u odigranoj ruci,te računa ukupne punte ruke.

U metodi *Pokupi***()** su implementirana 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. Pravila Igre forma

PravilaIgre Forma sadrži informacije o nazivu aplikacije, verziji, autoru aplikacije, te listu pravila igre.

Slika 26. PravilaIgre forma

3.2.13. IgraGotova Forma

Prikazuje se nakon završetka igre.Sadrži okvir za sliku, a slika se mijenja ovisno o ishodu igre.

3.2.14. Klasa Špil

Klasa Špil se koristi za izradu špila karata.Svaka karta je opisana atributima vrijednost i vrsta.Klasa Špil koristi pobrojane vrijednosti Vrsta, 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 };
```

Slika 27. Kôd pobrojane vrijedonsti Vrsta i Vrijednost

Metoda *setCards()* služi za izradu špila karata koji se koristi u igri. Na Slici 29. 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++;}}}
```

Slika 28. Kôd setCartds() metoda

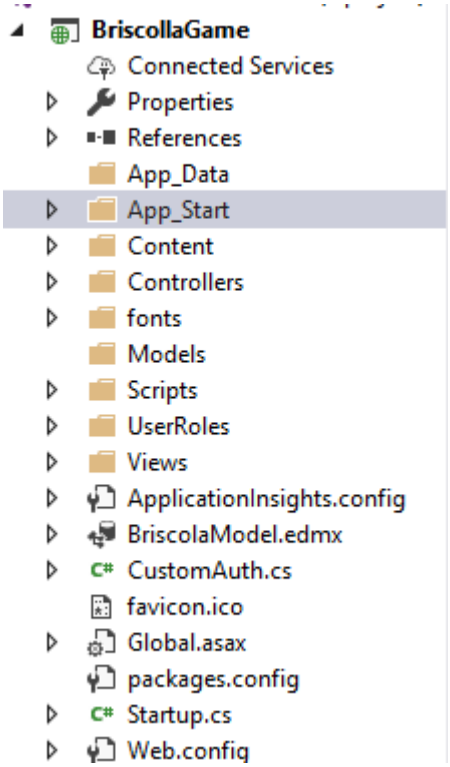
Da bi karte bile nasumično izmiješane prilikom svakog pokretanja nove igre u Špil klasa je implementirana metoda *shuffle*() .

3.2.15. Klasa Igrač

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

3.3. Web Aplikacija

3.3.1. Struktura web aplikacije



Slika 29. Web aplikacija struktura

3.3.2. Prava pristupa

Za prijavu korisnika i registraciju korisnika u web aplikaciju izrađen je prilagođeni(engl. „*custom*“) autentifikacijski sustav.Prava pristupa su podijeljena na tri uloge:kao ne prijavljeni posjetitelj, kao igrač(engl. „*player*“), te kao admin. Neulogirani korisnik se pri startu stranice preusmjerava na formu za prijave, te može pristupiti stranici za registraciju, početnoj stranici, te Pravila igre. Odgovarajuće akcije su predznačene atributom [AllowAnonymous], čime je omogućen pristup neautoriziranim korisnicima.

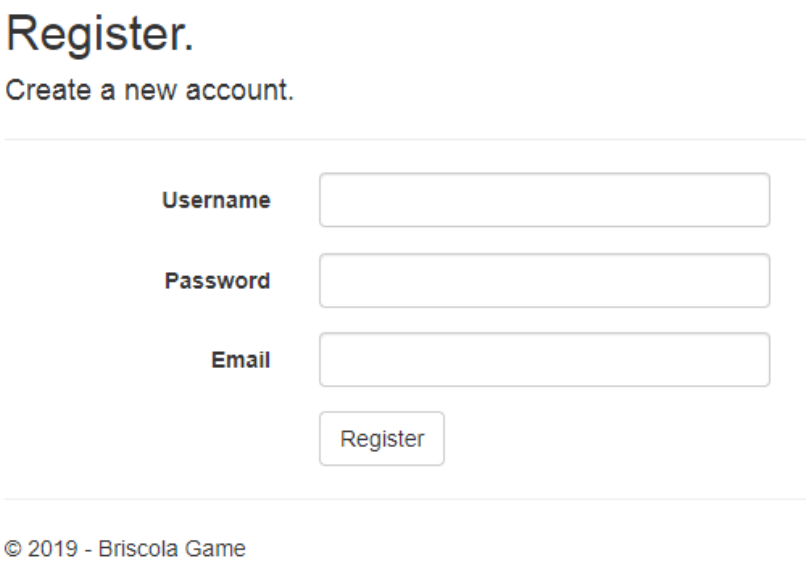
Igrač pristup

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

Admin pristup

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

3.3.3. Registracija forma



Slika 30. Registracija forma

Nakon uspješne registracije korisnik može pristupi web i desktop aplikaciji.Forma za registraciju sastoji se od polja za korisničko ime, lozinku, email. Formu je jednostavno proširiti s nekim od dodatnih polja kao sto su ponovni unos lozinke. Nakon klika na gumb registracija, provjerava se dali su zadovoljeni uvjeti za sva tekstualna polja.Uvjeti da bi se korisnik uspješno registrirao :

- lozinka minimalne veličine 6 znakova

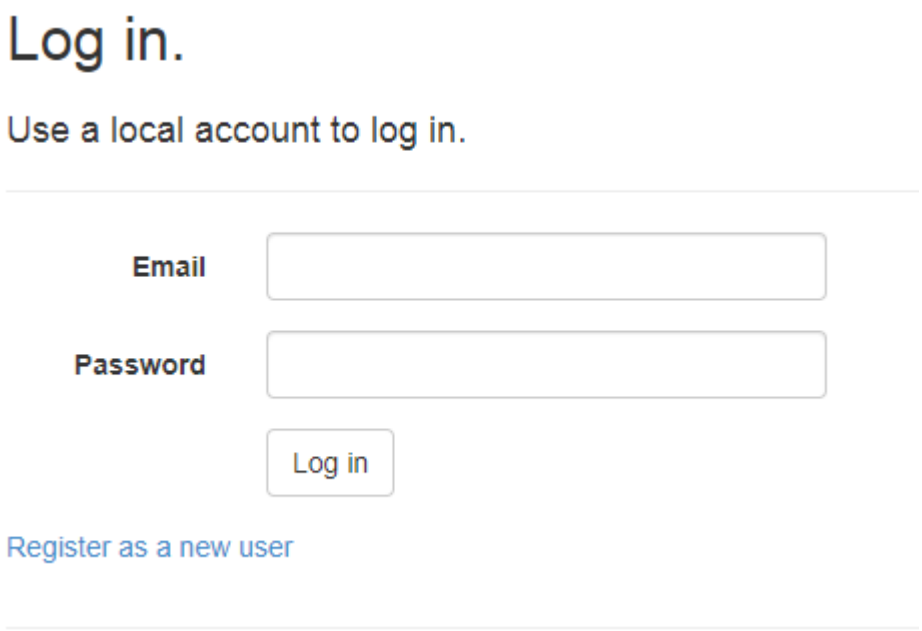
- ispravna email adresa

- nepostojeća email adresa i korisničko ime u bazi podataka.

U slučaju neuspješne registracije, odnosno da nisu zadovoljeni gore navedeni uvjeti, posjetitelj će biti preusmjeren na register formu, te će se ispisati odgovarajuća poruka, za polje koje nije zadovoljilo navedene kriterije.

3.3.4. Forma za prijavu

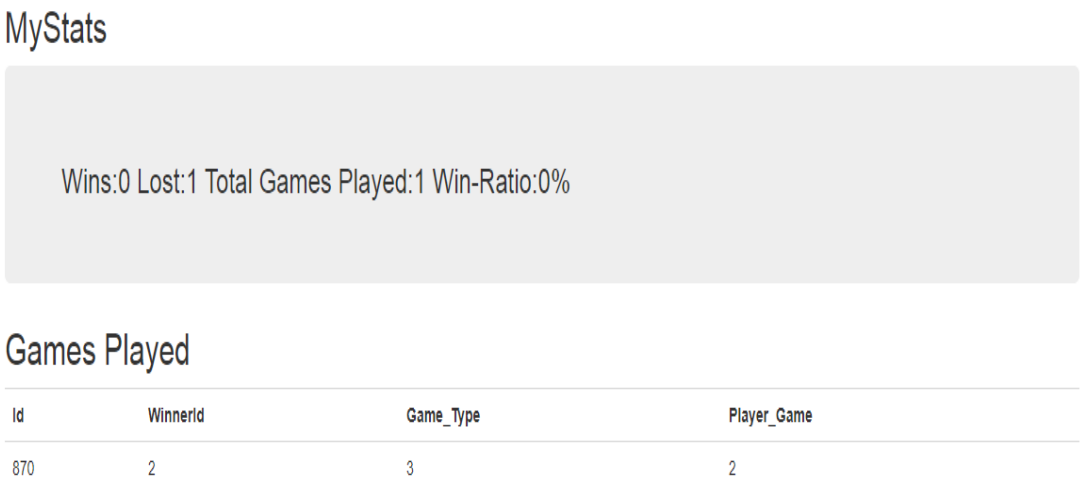
Forma za prijavu korisnika u web aplikaciju sastoji se od tekstualnih polja „*Email*“ i „*Password*“ , „*Log in*“) gumba, te linka „*Register*“.Prilikom startanja aplikacije, ako posjetitelj nije prijavljen u aplikaciju, automatski se preusmjerava na formu za prijavu. Također posjetitelj koji ne posjeduju račun imaju mogućnost preko linka pristupiti formi za registraciju.



Slika 32. Forma za prijavu

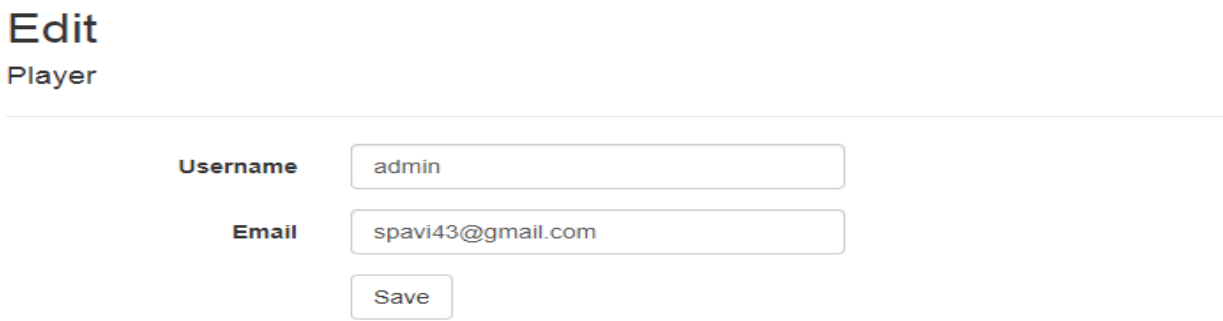
3.3.5. Moji rezultati prikaz

Moji rezultati prikazuje broj odigranih partija, broj pobjeda i poraza te ukupni postotak pobjeda, trenutno prijavljenog korisnika. U listi odigrane igre dijelu sadržana je kompletna povijest igara u kojima je korisnik sudjelovao te ishod pojedine igre.



Slika 32.Moji rezultati prikaz

3.3.6. Account Manager prikaz

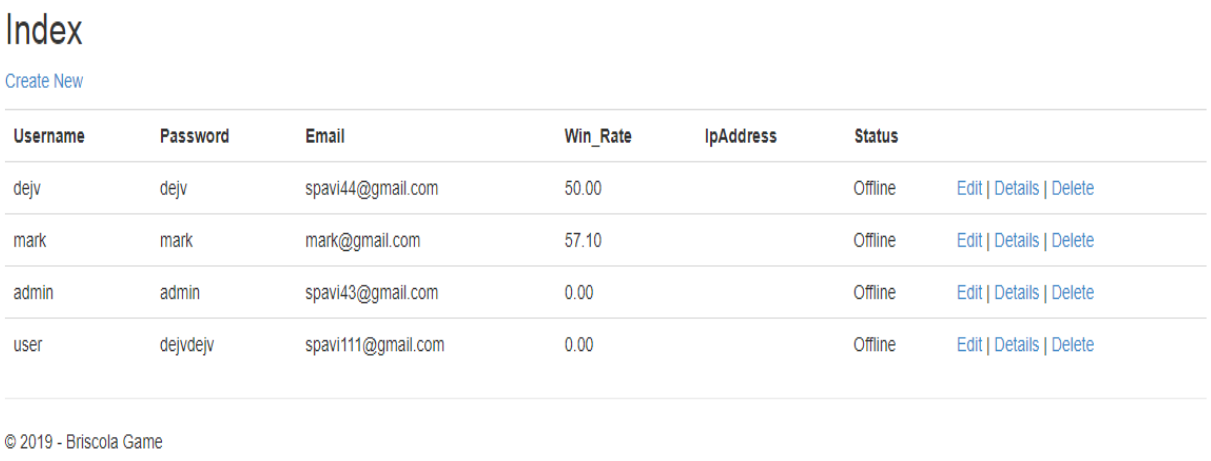


Slika 33. Account Manager prikaz

Logirani korisnici klikom na korisničko ime, koje se nalazi u navigaciji otvaraju Account Manager stranicu koji im omogućava promjenu korisničkog imena i email adrese. Klikom na gumb Spremi, promjene se pohranjuju u bazu podataka.

3.3.7. Lista igrača prikaz

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



Slika 34. Lista igrača prikaz

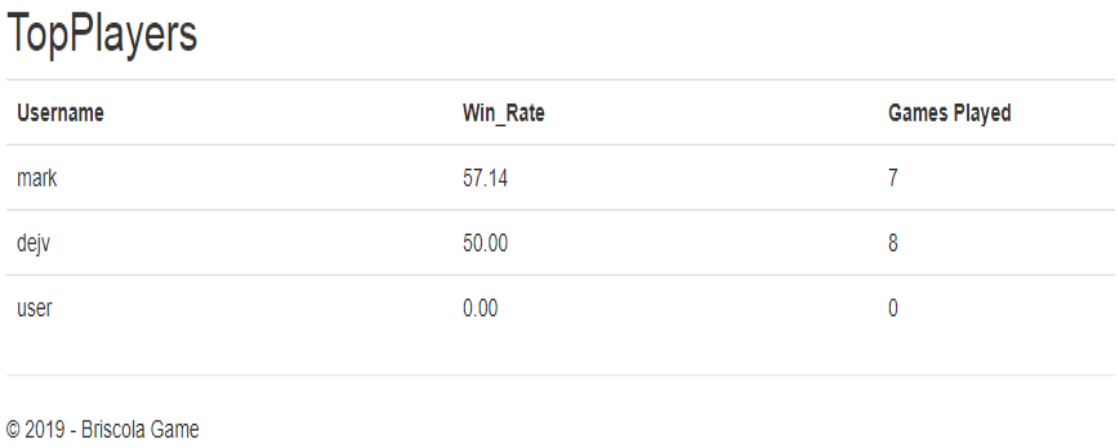
3.3.10. Najbolji igrači prikaz

Sadrži listu najbolje rangiranih igrača prema povijesti odigranih igara. Lista sadrži pet igrača, uz uvjet da ima najmanje pet korisničkih računa u bazi podataka. U slučaju manjeg broja korisnika u bazi, lista će sadržavati sve korisnike koji se nalaze u bazi podataka.

Kriteriji rangiranja su po postotku pobjeda te po broju odigranih igara. Ovaj kriterij onemogućuje da igrač koji ima jednu pobjedu uz jednu odigranu igru, bolje rangiran od igrača koji ima manji postotak pobjeda ali mnogo veći broj odigranih igara. Primjer upita Linq-a korištenjem lambda sintakse prikazan je na slici 35.

```
        var   p   =   players.Players.OrderBy(r=>r.Games.Count).OrderByDescending(r=>
r.Win_Rate).Take(3).ToList();
```

Slika 35. Kôd Upit Linq-a Lambda sintaksa



Slika 36. Najbolji Igrači prikaz

4. ZAKLJUČAK

Tema ovog završnoga rada bila je izrada multiplayer kartaške igre u desktop verziji sa pripadnom web aplikacijom. Ideja je bila izrada aplikacije koja se koristi u svrhu zabave korisnika. Desktop aplikacija je izrađena korištenjem windows formi. Web aplikacija je izrađena korištenjem aplikacijskog okvira ASP .NET MVC 5. Desktop aplikacija i web aplikacija u pozadini koriste bazu podataka Igrači(eng. „Player“) koja se nalazi na Microsoft Azure SQL poslužitelju. Za pristup bazi podataka te manipulaciju podataka koristi se EF 6.

U prvom dijelu rada objašnjena je izrada desktop aplikacije korištenjem windows formi. U drugom dijelu rada je objašnjena izrada web aplikacije korištenjem aplikacijskog okvira APS .NET MVC5. Glavni fokus je bio na izradi desktop aplikacije dok je web aplikacija dodana kako bi se administratoru omogućilo jednostavnije upravljanje bazom podataka, dok je korisniku omogućena izmjenu osobnih podataka, te pregled statistike igre. Web aplikacija omogućava jednostavniji pristup bazi time što nije potrebno imati desktop aplikaciju. Posjetitelju je time omogućen pristup bilo gdje u svijetu.

C# kombiniran s windows formama koje su dio .NET tehnologije predstavlja razvojno okruženje koje omogućava razvoj bilo kojega tipa aplikacija, čime se pokrivaju svi zahtjevi tržišta. Izbor programskog jezika sveo se na C# zbog njegove jednostavnosti te velike lepeze mogućnosti.

Razvoj aplikacije se nastavlja. U budućnosti će desktop aplikacija će dobiti nove funkcionalnosti poput mogućnost prisustvovanja do četiri korisnika u igri, te će biti dodana moći igranja protiv računala. Web aplikacija će dobiti funkcionalnosti koje će omogućiti pokretanje igre iz web preglednika.

5. LITERATURA

[1] Zbornik veleučilišta u Rijeci Vol.2 No.1 , https://www.veleri.hr/files/datoteke/knjige/digi/2014_06_01_Veleuciliste%20Zbornik.pdf(15.7.2019)

[2] Smijulj,A.: Izgradnja mvc modularnog radnog okvira, https://hrcak.srce.hr/file/190412(28.6.2019)

[3] Uvod u ASP .Net MVC, http://www.manuelradovanovic.com/2017/09/uvod-u-asp-net-mvc.html(25.6.2019)

[4] Kratofil, D. (2017). Izrada web aplikacije u programskom jeziku C# (Završni rad). Preuzeto s https://urn.nsk.hr/urn:nbn:hr:200:999545(11.7.2019)

[5] ENTITY FRAMEWORK za data-orijentisane aplikacije, https://www.helloworld.rs/blog/ENTITY-FRAMEWORK-za-data-orijentisane-aplikacije/289(25.6.2019)

[6] LINQ, https://www.geeksforgeeks.org/linq-language-integrated-query/(12.7.2019)

[7] 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/(15.7.2019)

[8] Štefanac,M. (2016). Web sustav za upravljanje dokumentima temeljem polustrukturiranih baza podataka(Diplomski rad), https://bib.irb.hr/datoteka/835099.1-mirstefan.pdf(21.7.2019)

[9] Json, https://www.webprogramiranje.org/json/(20.7.2019)

[10] Lovrić, V. (2016). Windows forme u C#-u (Završni rad). Preuzeto s https://urn.nsk.hr/urn:nbn:hr:200:800058(17.7.2019)

[11]Elektronički fakultet Osijek, http://www.etfos.unios.hr/~lukic/oop/Auditorne_vje%C5%BEbe_5.pdf(26.6.2019)

[12] Liberty, Jesse.: Programming C#, 4th Edition,O'Reilly Media 2009

[13] From zero to hero in Json with C#, https://www.c-sharpcorner.com/article/from-zero-to-hero-in-json-with-c-shar/(25.4.2019)

[14] Json Creation Part 1, https://www.softwaretestinghelp.com/create-json-objects-using-c/ (24.6.2019)

[15] Socket programing in C#, https://www.geeksforgeeks.org/socket-programming-in-c-sharp/ (26.6.2019)

[16] Socket programing in C#, https://www.c-sharpcorner.com/article/socket-programming-in-C-Sharp/ (12.6.2019)

[17] Microsoft docs, https://docs.microsoft.com/en-us/previousversions/visualstudio/visual-studio-2008/dd30h2yb(v%3dvs.90)

[18] Entity framework, https://arianscorner.wordpress.com/2014/04/04/designing-a-many-to-many-relationship-with-additional-fields-using-entity-framework/(15.7.2019)

[19] Asp Net MVC Tutorials, https://www.tutorialsteacher.com/mvc/asp.net-mvc-tutorials (10.7.2019)

[20] Entity Framework Tutorial, https://www.entityframeworktutorial.net/ 9 (20.7.2019)

6. POPIS SLIKA