APLIKACIJA BRIŠKULA

Pavić, Josip

Undergraduate thesis / Završni rad

2019

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Split / Sveučilište u Splitu**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:228:623165

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2025-03-01**

Repository / Repozitorij:

Repository of University Department of Professional Studies

zir.nsk.hr

UNIVERSITY OF SPLIT

dabar
DIGITALNI AKADEMSKI ARHIVI I REPOZITORIJI

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

APLIKACIJA BRIŠKULA

Split, rujan 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

APLIKACIJA BRIŠKULA

Split, rujan 2019.

SVEUČILIŠTE U SPLITU

SVEUČILIŠNI ODJEL ZA STRUČNE STUDIJE

Preddiplomski stručni studij Informacijske tehnologije

Predmet: Programiranje na internetu

Z A V R Š N I R A D

Kandidat: Josip Pavić

Naslov rada: Aplikacija *Briškula*

Mentor: Marina Rodić, predavač

Split, rujan 2019.

SADRŽAJ

SAŽETAK

U ovom završnom radu opisana je izrada aplikacije *Briškula*. Aplikacija je bazirana na višekorisničkoj logici (engl. *multiplayer*) , a namijenjena je isključivo u svrhu zabave. Projekt se sastoji od *desktop* aplikacije i *web* aplikacije. *Desktop* i *web* aplikacija koriste istu bazu podataka. Baza podataka izrađena je u SQL Management Studio 2017 te je postavljena na Microsoft Azure SQL poslužitelj (engl. *server*). *Desktop* aplikacija omogućuje korisnicima međusobno nadmetanje na računalima povezanima u lokalnu mrežu. *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 sinhronizirana 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. Korisnicima je također omogućeno lakše uređivanje profila putem *web* aplikacije.

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

SUMMARY

Briscola Application

This thesis describes the development of the *Briscola* application. The application is based on multiplayer logic and is intended for entertainment purposes only. The project consists of a desktop application and a web application. A desktop application and web application uses the same database in the background. This database was developed in SQL Studio 2017 environment and it was migrated to the Microsoft Azure SQL server. The desktop application enables users to compete with each other on the computers connected to the local network. The desktop application enables the user to create the server or join an existing one. The game itself is held in the desktop application. It is created in a way that two players can compete against each other and the final result is stored in a database. The server can create a game type with three or four cards. The application is synchronized with the database that enables the user to have insight into the list of all active servers and active players. Web application helps the administrator to have easier management and access to the database data since it is not necessary to have the desktop application installed on your personal computer. Users also have an easier time managing their profiles.

Keywords: C#, desktop application, web application

1. UVOD

Cilj ovog završnog rada je izrada aplikacije koja će služiti za zabavu korisnika. U tu svrhu izrađene su *desktop* i *web* aplikacija.

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

Završni rad sadrži četiri poglavlja. Nakon uvoda, u drugom poglavlju završnog rada analizirane su tehnologije koje su se koristile pri izradi *desktop* i *web* aplikacije. U trećem poglavlju rada analiziran je model podataka te su prikazane relacije između entiteta. Kroz primjere i slike prikazan je razvoj *desktop* i *web* aplikacije. Četvrto poglavlje rada jest zaključak u kojem se sažeto iznosi sve što je istaknuto u radu te se donose određene spoznaje o korištenim tehnologijama tijekom izrade aplikacije i mogućnosti razvoja aplikacije u budućnosti.

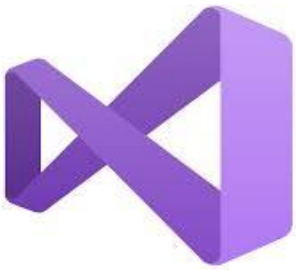
2. TEHNOLOGIJE

2.1. ASP.NET MVC

ASP *(engl. Active Server Pages*) je tehnologija razvojnog okvira (engl. *framework*) .NET koja se koristi za razvoj dinamičkih i interaktivnih *web* stranica te za razvoj *web* usluga (engl. *services*) . Stranice ASP izvršavaju se na poslužiteljskoj strani te generiraju kôd u HTML (engl. *HyperText Markup Language*) ili XML (engl. *Extensible Markup Language*) formatu koji se šalje *desktop* ili mobilnim preglednicima. MVC je predložak (engl. *template*) koji se koristi pri izradi *web* aplikacija. Kôdna arhitektura MVC, iako razvijana prvenstveno za *desktop* aplikacije, danas je popularnija kod izrada *web* aplikacija [1]. Osnovna ideja arhitekture MVC jest razdvajanje kôda na tri cjeline: *Model*, Pogled (engl. *View*) i Upravitelj (engl. *Controller*) [2]. U sloju modela najčešće je implementirana poslovna logika, točnije to je skup klasa koje opisuju podatke poslovne logike s kojom se radi. Model enkapsulira (zaštita podataka u klasi) podatke koji se čuvaju u bazi podataka. Pogled omogućuje prikaz podataka te se najčešće koristi za prikaz podataka iz modela. Upravitelj je, pak, veza između modela i pogleda. Čita ulazne podatke od korisnika te ih prosljeđuje modelu. Nakon komunikacije s modelom upravitelj odlučuje koji će se pogled prikazati krajnjem korisniku. MVC za čuvanje podataka koristi metode koje se nalaze u upravitelju.

2.2. Razvojno okruženje Microsoft Visual Studio Community

Visual Studio je programsko razvojno okruženje, služi za pisanje raznih vrsta aplikacija za operativni sustav Windows [4]. Na slici 1. prikazan je Visual Studio logo.



Slika 1: Visual Studio logo

Tvorac Visual Studio je tvrtka Microsoft, a trenutno dolazi u tri varijante:

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

Visual Studio Community je moguće besplatno preuzeti na službenim stranicama Microsoft kompanije. Instalacija zahtijeva posjedovanje besplatnoga Microsoft naloga, a uvjet za uspješnu instalaciju je stalna internet veza. Visual Studio omogućuje integraciju različitih tehnologija koje ne moraju biti razvijene od strane Microsoft kompanije [4]. Visual Studio je jednostavan za korištenje. Pri pisanju kôda nudi se završetak riječi što značajno olakšava posao prilikom izrade aplikacije.

2.3. SQL Server Management Studio

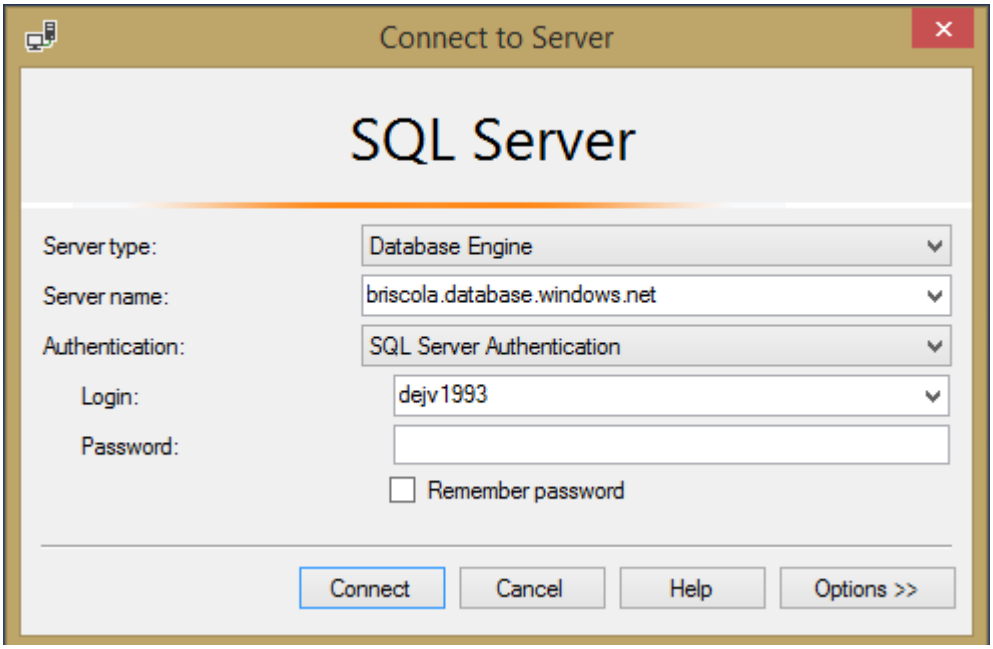
SQL Server Management Studio je integrirano razvojno okruženje za upravljanje SQL (engl. *Structured Query Language*) poslužiteljem. Instalacijski paket je moguće besplatno preuzeti sa službenih stranica Microsoft kompanije. Na slici 2. prikazan je SQL Server Management Studio logo.

Slika 2: SQL Server Management Studio logo

Uz pomoć Management Studio administrator može upravljati skoro svim aspektima SQL poslužitelja. Management Studio omogućuje kreiranje i upravljanje bazom podataka, prilagođavanje sigurnosti, izradu korisničkih naloga za pristup SQL poslužitelju [5]. Kreiranjem baze podataka kreiraju se minimalno dvije datoteke. Prva datoteka s ekstenzijom

„.MDF“ sadrži dodatke i objekte, dok datoteka s ekstenzijom „.LDF“ predstavlja dnevnik transakcija [6]. Pokretanjem Management Studio otvara se forma za prijavu na SQL poslužitelj.

Na slici 3. prikazana je forma za prijavu na SQL poslužitelj.



Slika 3: Forma za prijavu na SQL poslužitelj

Komponente forme za prijavu na SQL poslužitelj:

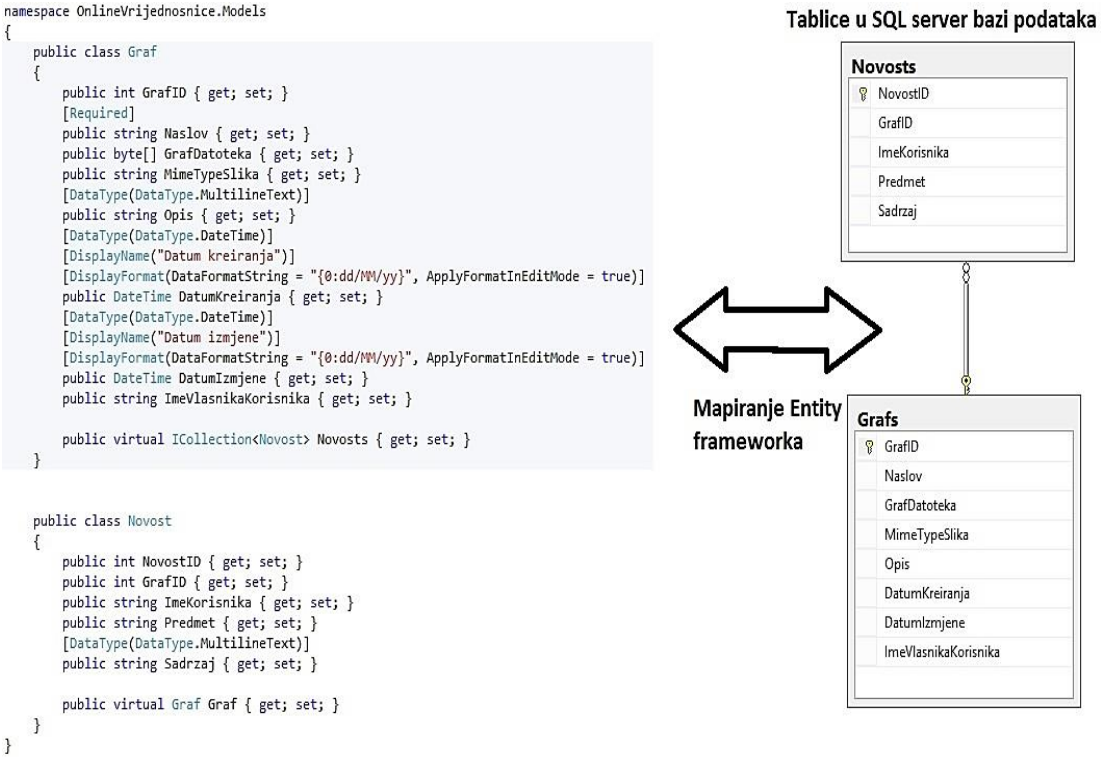
- Vrsta poslužitelja (engl. *Server type*) – odabire se poslužitelj na koji se želimo povezati
- Ime poslužitelja - ime poslužitelja na koji se želimo povezati
- Autentifikacija (engl. *authentication*) - odabire se vrsta naloga za prijavu na SQL poslužitelj
- Prijava (engl. *login*) - unos u ovo polje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

2.4. Razvojni okvir Entity

Razvojni okvir Entity je okvir objektno relacijskog mapiranja otvorenog kôda za ADO (engl. *ActiveX Data Objects*) . Objektno relacijsko mapiranje (engl. *object relational*

mapping) značajno olakšava razvoj podatkovno orijentiranih aplikacija [8]. Entity može automatski generirati veliki dio kôda, čime se programeru značajno štedi vrijeme.

Entity omogućuje rad s podacima u obliku objekta umjesto s tabličnim podatcima. Entity mapira relacijske tablice u objekte koje C# razumije [9]. Prijenos podataka između kôda programskog jezika C# i baze podataka se naziva mapiranje (engl. *mapping*). Na slici 4. prikazano je mapiranje u razvojnom okviru Entity.



Slika 4: Mapiranje u razvojnom okviru Entity

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

- Baza prva (engl. *database first*) – generira sve potrebne objekte u kôdu na temelju modela baze podataka.
- Kôd prvi (engl. *code first*) – generira sve potrebne bazne objekte na osnovi postojećeg kôda. Omogućuje kreiranje klasa bez grafičkog korisničkog sučelja (engl. *Graphical User Interface*). Taj se pristup preporučuje koristiti ako ne postoji baza podataka.

- Model prvi (engl. *model first*) – definira entitete i modele na temelju kojih će se generirati baza podataka i klase [4].

Kao most između *entity* klasa i baze podataka koristi se posebna klasa koja predstavlja kontekst baze podataka (engl. *dbcontext*) , a naziva se kontekstna klasa. `DtSet` sadrži objekt koji će Entity generirati na osnovu baze. Izgled kontekstne klase prikazan je na ispisu 1.

```
public partial class PlayersEntities1 : DbContext{
public virtual DbSet<Game> Games { get; set; }
public virtual DbSet<Player> Players { get; set; }}
public virtual DbSet<PlayerRole> PlayerRoles { get; set; }
public virtual DbSet<Role> Roles { get; set; }}
```

Ispis 1: Kontekstna klasa

2.5. LINQ

LINQ (engl. *Language Integrated Query*) je dodatak programskim jezicima .NET tehnologije koji sadrži izraze za upite (engl. *query expressions*) [9]. LINQ unificira način pristupa i pretrage podataka. Omogućuje pristupanje podatcima različitog tipa korištenjem iste sintakse. Uvođenje LINQ-a omogućuje jednostavnije pretraživanja podataka u bazi, čime se smanjuje količina potrebnoga kôda te povećava čitljivost kôda [10]. LINQ sadrži oko pedeset upita operatora pomoću kojih se značajno smanjuje vrijeme sortiranja, filtriranja te grupiranja podataka. Neki od najčešće korištenih operatora jesu sljedeći:

- sortiranje
- filtriranje
- grupiranje
- spajanje
- pretvorba.

Operatori koji se koriste u metodama standardnih upita nazivaju se izrazi *lambda* (engl. *Lambda expressions*). Ti izrazi omogućuju definiranje operatora upita kao metoda. LINQ se može koristiti za sve vrste podataka koje su izvedeni iz sučelja (engl. *interface*) `IEnumreable` [10].

2.6. JSON

JSON (engl. *JavaScript Object Notation*) je tekstualni format čija je namjena prijenos podataka u formatu koji je čitljiv i ljudima i strojevima [11].

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 (engl. *java script*) parser. Objekt u JSON formatu je oblika ključ : par, a nalazi se unutar vitičastih zagrada [11].

JSON ne ovisi o programskom jeziku, a njegova je najveća primjena u *web* aplikacijama. Objekt IgraTri u JSON formatu prikazan je na ispisu 2.

```
{"i":0,"d":{"karte":[{"v":0,"vr":0},{"v":0,"vr":1},
{"v":0,"vr":2},{"v":0,"vr":3},{"v":0,"vr":4},
{"v":0,"vr":5},{"v":0,"vr":6},
{"v":0,"vr":7},{"v":0,"vr":8},{"v":0,"vr":9},
{"v":1,"vr":0},{"v":1,"vr":1},
{"v":1,"vr":2},{"v":1,"vr":3},{"v":1,"vr":4},
{"v":1,"vr":5},{"v":1,"vr":6},{"v":1,"vr":7},{"v":1,"vr":8},
"p2":{"name":"player2","punti":0,
"turn":false,"ruka":[{"v":0,"vr":3},{"v":0,"vr":4},
{"v":0,"vr":5},null]},"c1":null,"c2":null,"c3":null}
```

Ispis 2: Objekt IgraTri u JSON formatu

2.7. Windows forme

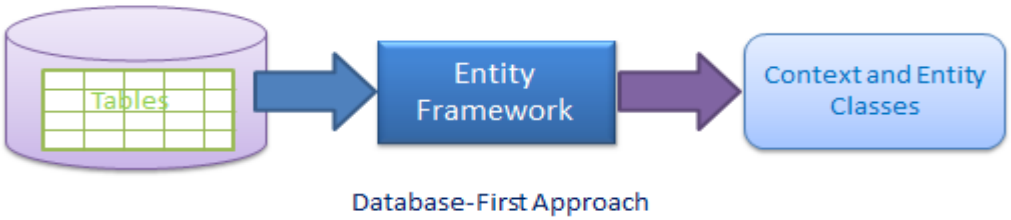
Windows forma je alat koji se koristi za izradu Windows aplikacija. Te su forme učinkovit i jednostavan način komunikacije koja se ostvaruje između korisnika i programa. Window forma je prozor koji sadrži kontrole za prikaz, unos i manipulaciju podatcima [12]. Za izradu Windows formi potrebno je unutar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

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

3. APLIKACIJA

3.1. Baza podataka

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



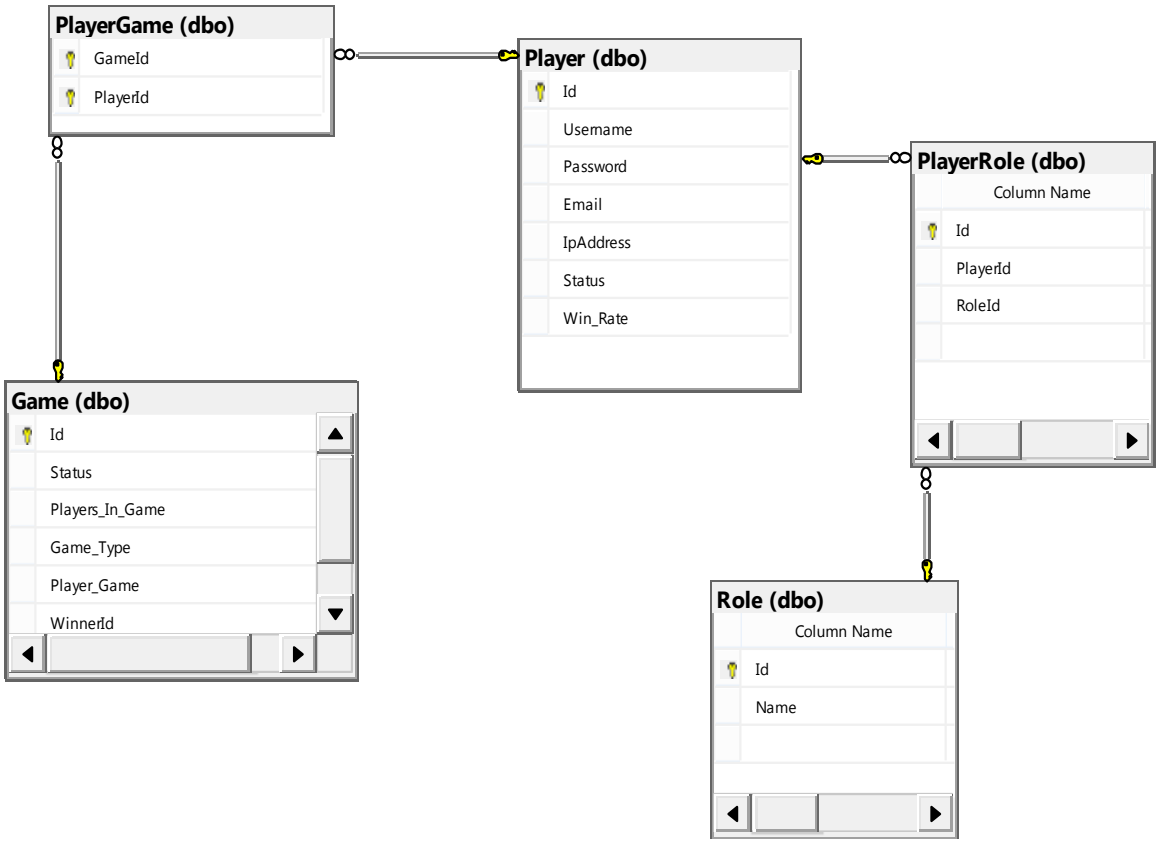
Slika 5: Pristup baza prva

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

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 6: Tablica Korisnik

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



Slika 7: Relacijski model baze podataka

Za manipulaciju podatcima koji se nalaze u bazi podataka treba povezati aplikaciju s bazom. Kontekstna klasa upravlja vezom s bazom te se po potrebi spaja ili odspaja s baze. PlayersEntities1 klasa izvedena je iz kontekstne klase. Konstruktor PlayersEntities1 klase kao parametar može primiti ime baze na koju se spaja ili *string* na vezu s bazom. PlayersEntities1 konstruktor prikazan je na ispisu 3.

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

Ispis 3: PlayersEntities1 konstruktor

Za ostvarivanje veze s Microsoft Azure SQL poslužiteljem treba podesiti *string* na vezu s bazom. *String* se kreira automatski te je podešen na vezu s bazom podataka koja se nalazi na lokalnom (engl. *localhost*) računalu. Za ostvarivanje veze s Microsoft Azure SQL poslužiteljem potrebno je izmijeniti postojeći *string*. Na ispisu 4. prikazan je kôd ispravno podešenog *stringa* za povezivanje s Microsoft Azure SQL poslužiteljem.

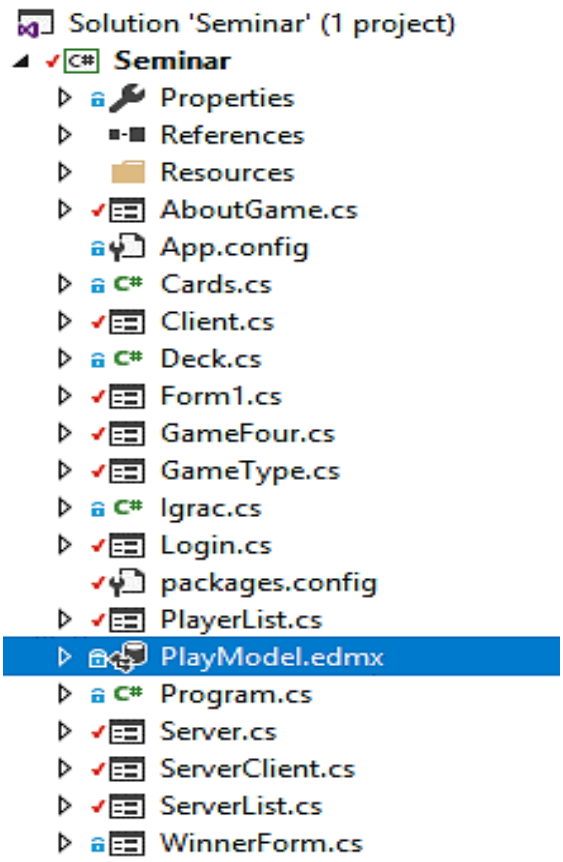
```
metadata=res://*/PlayModel.csdl|res://*/PlayModel.ssdl|res://*/PlayModel.msl;

                provider=System.Data.SqlClient;provider connection
  string="datasource=briscola.database.windows.net;initial catalog=Players;

      user id=željeno_ime;password=željena_šifra;

      MultipleActiveResultSets=True;

      App=EntityFramework"
```

Ispis 4: String za konekciju s Microsoft Azure SQL poslužiteljem

3.2. Desktop aplikacija

3.2.1. Struktura *desktop* aplikacije

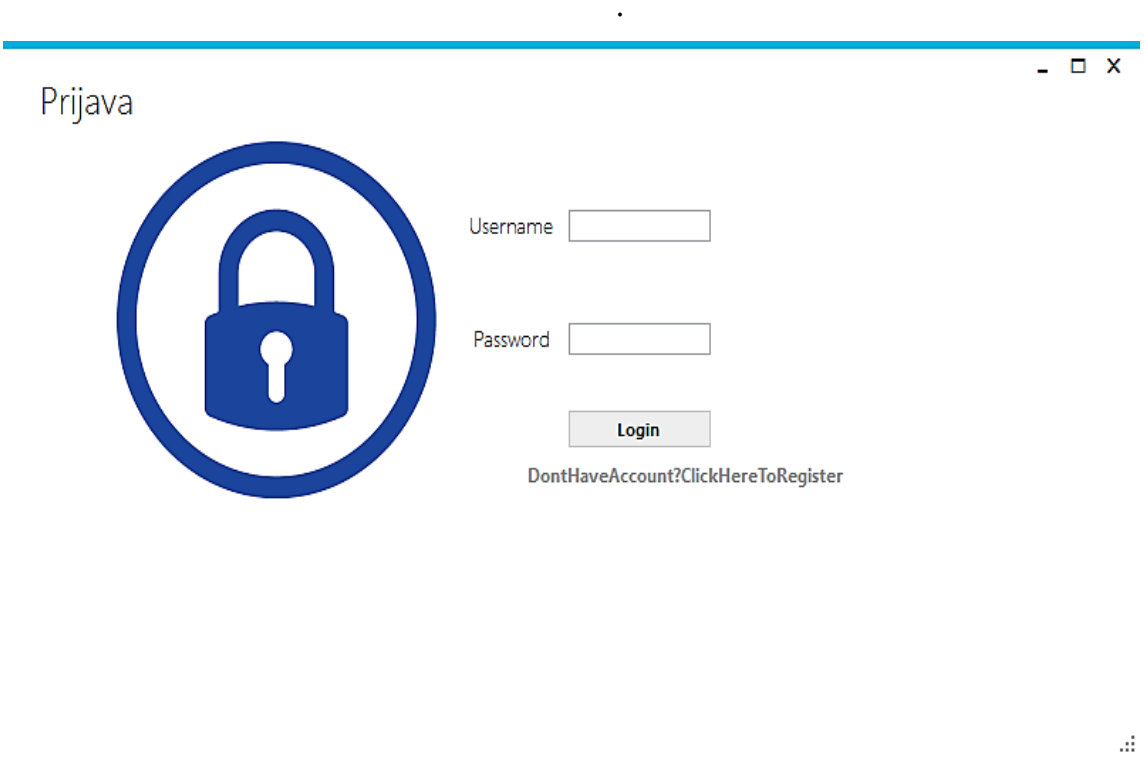
Struktura *desktop* aplikacije prikazana je na slici 8.



Slika 8: Struktura *desktop* aplikacije

3.2.2. Forma za prijavu

Pokretanjem aplikacije prikazuje se forma za prijavu koja omogućuje prijavu za postojeće korisnike. Forma za prijavu sastoji se od dva tekstualna polja (engl. *textbox*), dugmeta (engl. *button*) te poveznice (engl. *link*) . Klik na poveznicu za registraciju odvodi korisnika na *web* aplikaciju Briškula. Nakon uspješnog logiranja forma prelazi u pozadinu. Korisnik je usmjeren na glavni izbornik, odnosno otvara se forma PosluziteljKlijent. Status korisnika mijenja se iz izvanmrežno (engl. *offline*) u mrežno (engl. *online*). Forma za prijavu prikazana je na slici 9.



Slika 9: Forma za prijavu

3.2.3. Forma PosluziteljKlijent

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

- dugme *Poslužitelj*

- dugme *Vrsta Igre*
- dugme *Lista Poslužitelja*
- dugme *Pravila Igre*

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

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

Klikom miša na neku ponuđenu opciju otvara se odabrana forma. Forma PosluziteljKlijent prelazi u pozadinu te njezine kontrole više nisu vidljive korisniku sve dok se trenutna forma ne zatvori. Na slici 10. prikazana je forma PosluziteljKlijent s administratorskim pravima.

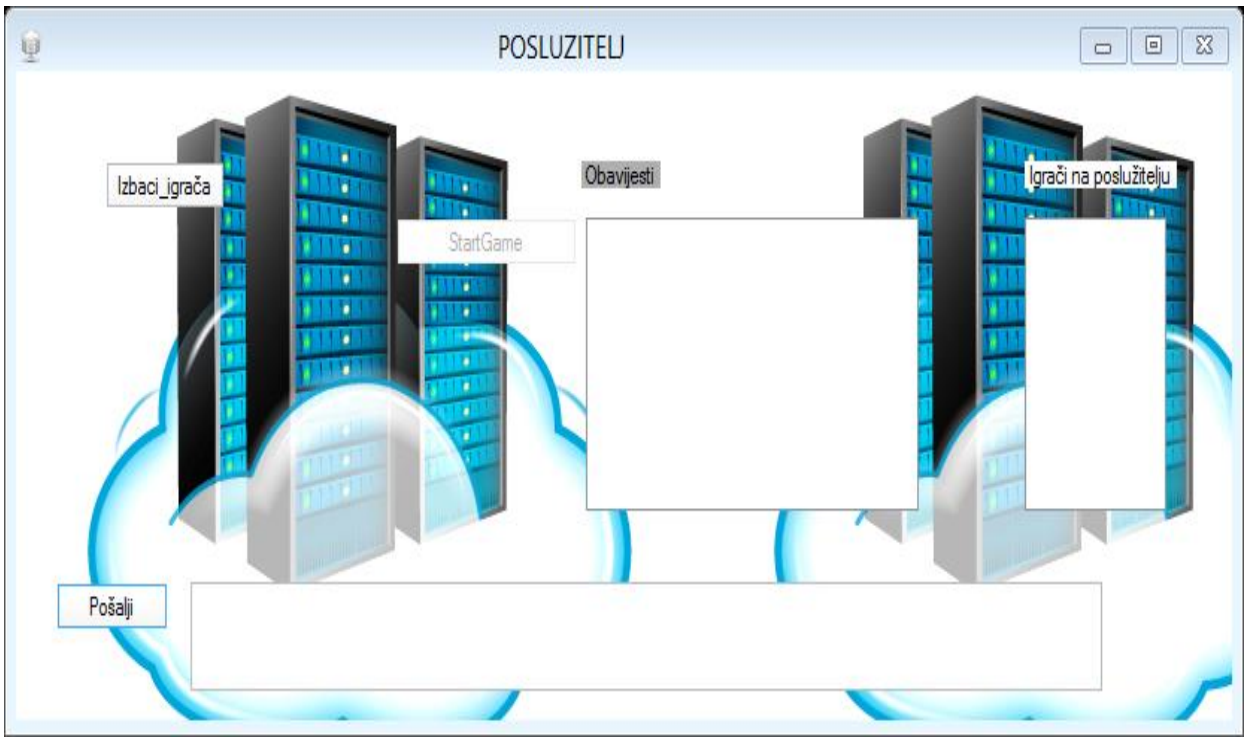
Slika 10: Forma PosluziteljKlijent s administratorskim pravima

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

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

3.2.4. Forma Poslužitelj

Pokretanjem forme Poslužitelj pokreće se poslužitelj. Nakon pokretanja poslužitelj na priključku (engl. *portu*) 7000 sluša zahtjeve klijenta za spajanje. Lista *Igrači na poslužitelju* sadrži popis korisnika koji su trenutno spojeni na poslužitelj. Forma Poslužitelj prikazana je na slici 11.



Slika 11: Forma Poslužitelj

Klikom na dugme *Izbaci igrača* klijent se preusmjerava na formu ListaIgrača te se uklanja iz liste *Igrači na poslužitelju*. Na ispisu 5. prikazan je kôd za uklanjanje klijenta s poslužitelja.

```
OnFormUnsub("Form");
 tcpServer.BroadcastLine("kicked");
Client_Disconnect(null, tcpClient);
```

Ispis 5: Kôd za uklanjanje klijenta s poslužitelja

Kako bi se omogućila komunikacija s klijentom obavlja se pretplata (engl. *subscribe*) na događaj `DataReceived` iz SimpleTcpServer klase. Klikom na dugme *Pošalji* šalje se poruka svim klijentima spojenim na poslužitelj. Poruka sadrži tekst koji se nalazi u tekstualnom polju. U listi *Obavijesti* nalaze se sve poruke poslane od strane klijenta. Kôd za pretplatu na događaje SimpleTcpServer klase prikazan je na ispisu 6.

```
tcpServer = new SimpleTcpServer();

tcpServer.Delimiter = 0x13;

tcpServer.StringEncoder = Encoding.ASCII;

tcpServer.DataReceived += Server_Recived;

tcpServer.ClientConnected += Client_Connected;

tcpServer.ClientDisconnected += Client_Disconnect;
```

Ispis 6: Kôd za pretplatu na događaje SimpleTcpServer klase

U prikazanom kôdu na ispisu 6. vrši se pretplata na događaje `ClientConnected` i `ClientDisconnected` SimpleTcpServer klase. Kako bi poslužitelj bio obaviješten o spajanju klijenta vrši se pretplata na događaj `ClientConnected`. Za obavijest o odspajanju klijenta s poslužitelja vrši se pretplata na događaj `ClientDisconnected`.

Prilikom pokretanja igre na poslužiteljevoj strani izvršava se serijalizacija (pretvorba objekta u niz bajtova) objekta tipa IgraTri ili IgraČetiri, ovisno o vrsti igre. Serijalizirani objekt poslužitelj šalje svim klijentima na mreži. Na ispisu 7. prikazan je kôd JSON serijalizacija.

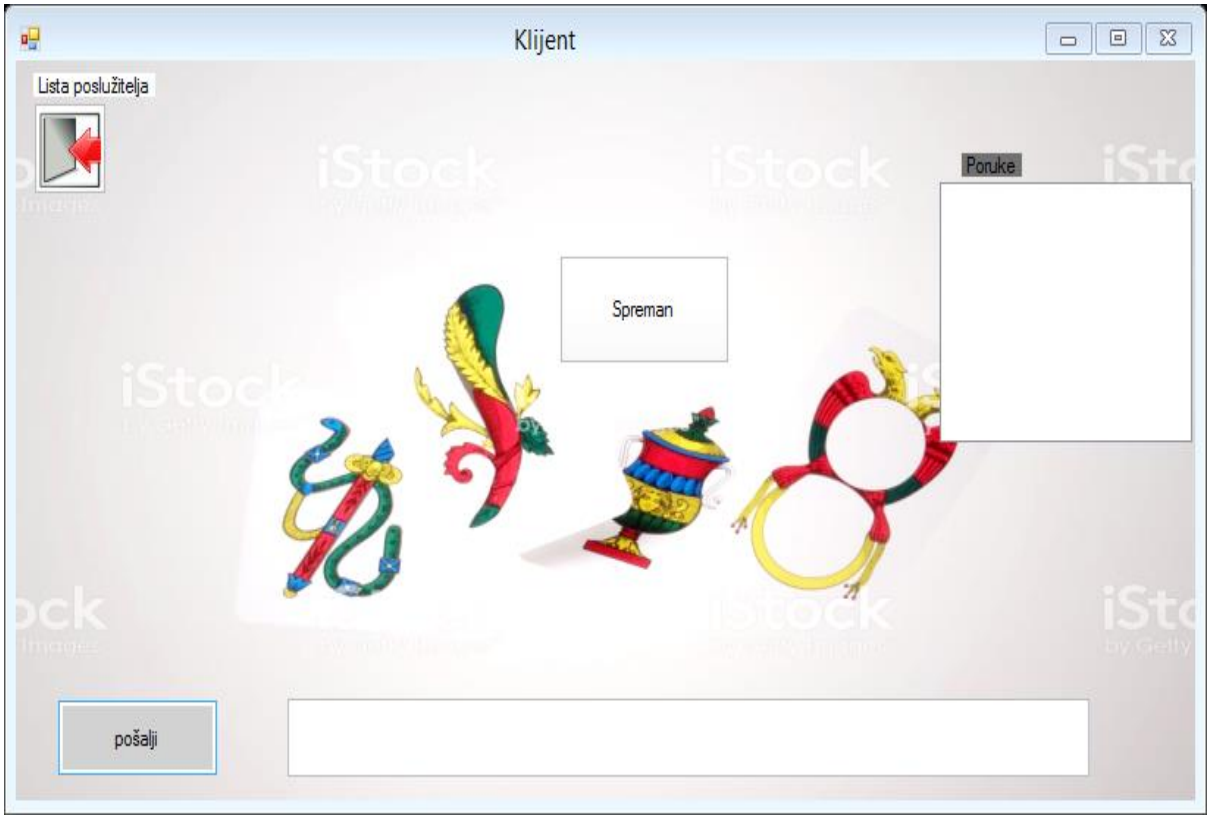
```
json= JsonConvert.SerializeObject(f);

tcpServer.BroadcastLine(json);
```

Ispis 7: Kôd JSON serijalizacija

3.2.5. Forma Klijent

Nakon uspješnoga povezivanja s poslužiteljem otvara se forma Klijent. Klikom na dugme *Ready* klijent omogućuje poslužitelju pokretanje igre.

Klijent može napustiti poslužitelja pritiskom na dugme *Return*. Klikom na to dugme zatvara se forma Klijent te iz pozadine na površinu prelazi forma Lista Poslužitelja. Klikom na dugme *Send* klijent šalje poruku poslužitelju, a poruka sadrži tekst koji se nalazi u tekstualnom polju. U listi Poruke nalaze se poruke razmijenjene s poslužiteljem. Forma Klijent prikazana je na slici 12.



Slika 12: Forma Klijent

Kako bi komunikacija s poslužiteljem bila moguća klijent se mora pretplatiti na događaj iz SimpleTcpClient klase. Kôd pretplata na `DataReceived` događaj prikazan je na ispisu 8.

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

Ispis 8: Kôd pretplata na DataReceived događaj

U formi Klijent obavlja se deserijalizacija (engl. *deserialize*) *stringa* u objekt tipa IgraTri ili IgraČetiri. Deserijalizaciju je nužno izvršiti unutar `BeginInvoke (MethodInvoker)` funkcije, u suprotnom dolazi do greške tijekom izvođenja (engl. runtime error). 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>(g);
   Thread.Sleep(1);}
   catch (Exception e){
   MessageBox.Show(e.Message.ToString();}}
```

Ispis 9: Kôd deserijalizacija objekta

Po završetku igre poziva se metoda `OnFormUnsub`. U metodi se vrši odjava s pretplaćenih događaja, kako pri pokretanju nove igre nebi dolazilo do povećanja pretplate na isti događaj. Kôd za odjavu s događaja prikazan je na ispisu 10.

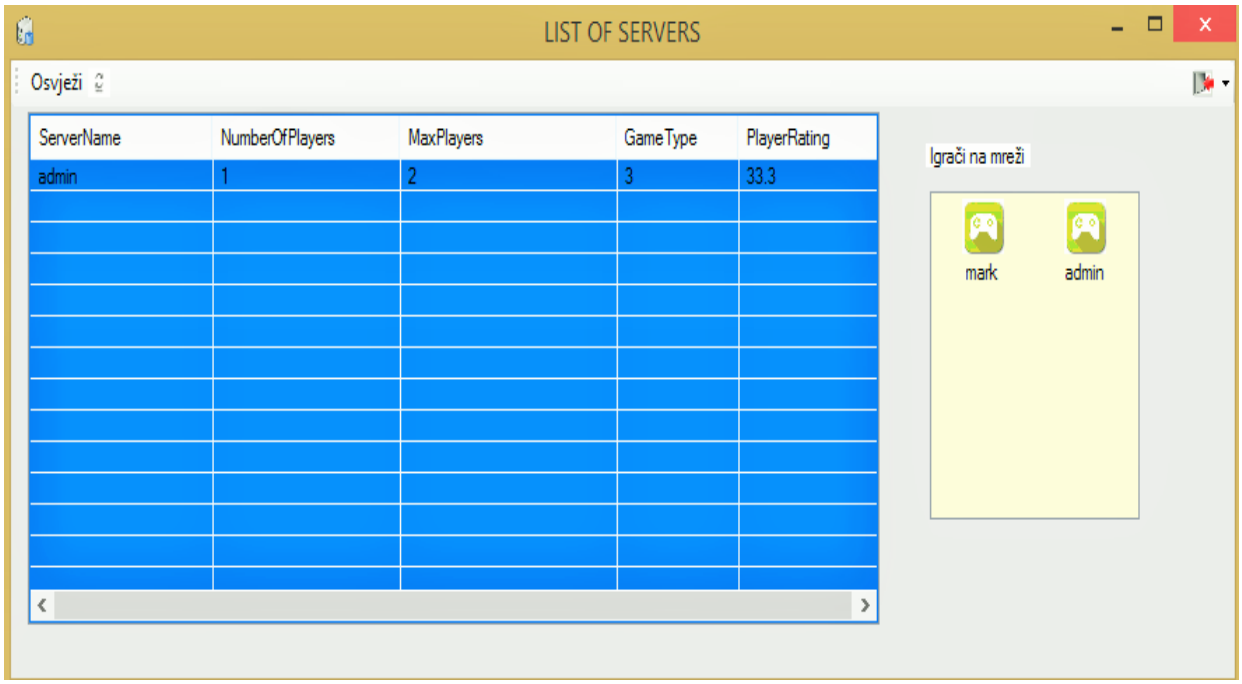
```
public void OnFormUnsub(string s)
{
if (gameType == 3)
 {
      this.MoveMade -= f.OnMoveMade;
      f.GameEnded -= this.OnGameEnded;
      f.MouseClicked -= this.OnMOuseClicked;
      f.FormExiting -= this.OnFormClosing;
      f.Close();
 else{
      this.MoveMade -= g.OnMoveMade;
      g.GameEnded -= this.OnGameEnded;
      g.MouseClicked -= this.OnMOuseClicked;
      g.FormUnsub -= this.OnFormUnsub;
      g.Close();}
}
```

Ispis 10: Kôd za odjavu s događaja

3.2.6. Forma Lista Poslužitelja

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

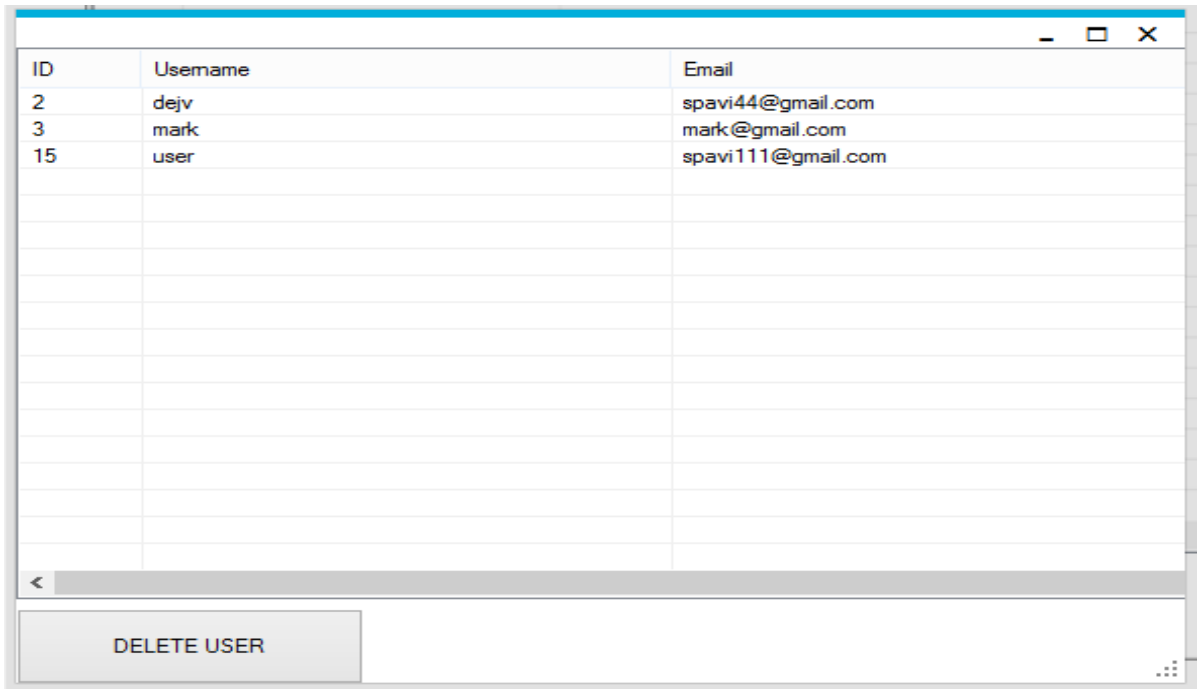
Korisnik odabire željeni poslužitelj te dvoklikom miša pokušava pristupiti poslužitelju. Ako se klijent uspješno poveže na poslužitelja otvara se forma Klijent, dok forma Lista Poslužitelja prelazi u pozadinu i njezine kontrole više nisu vidljive korisniku. Ako je poslužitelj popunjen ili ako postoje tehnički problemi, dolazi do neuspješnog povezivanja. Forma Lista Poslužitelja prikazana je na slici 13.



Slika 13: Forma Lista Poslužitelja

3.2.7. Forma ListaIgrača

Formi ListaIgrača moguće je pristupiti samo s ulogom administratora. U listi *Korisnici* nalaze se svi korisnici sa svojim osnovnim informacijama. Klikom na dugme *Delete User* iz baze se uklanja odabrani korisnik. Forma ListaIgrača prikazana je na slici 14.

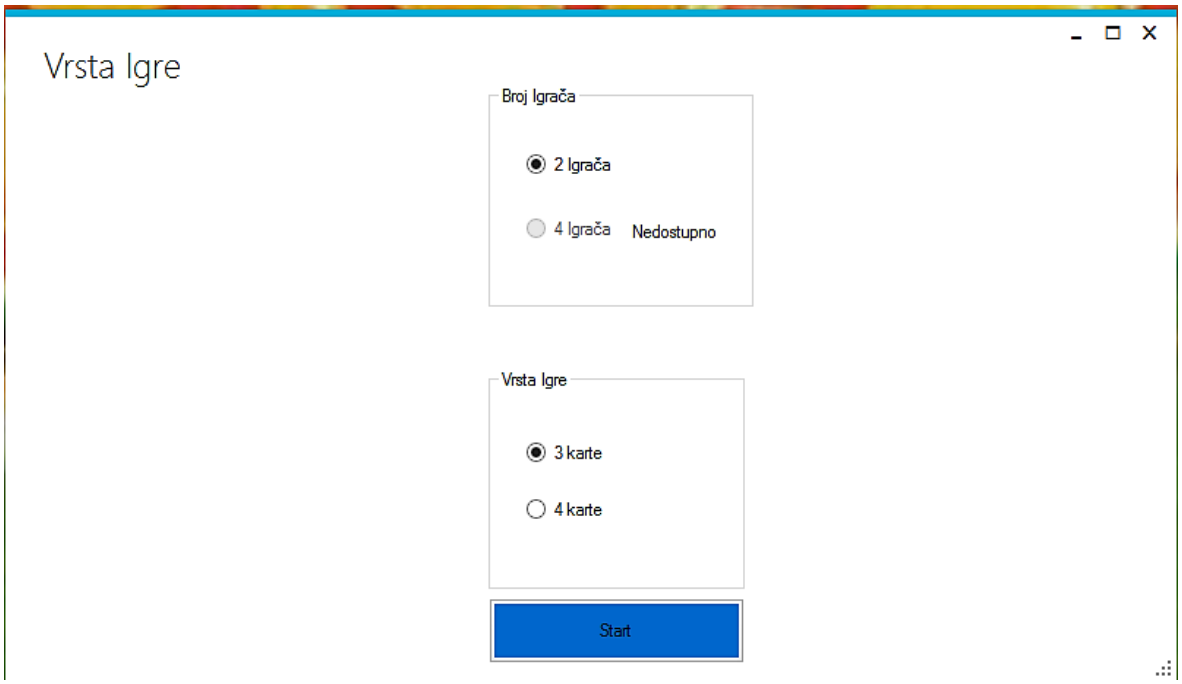


Slika 14: Forma ListaIgrača

3.2.8. Forma Vrsta Igre

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

Radijsko dugme *Vrsta Igre* sadrži mogućnost odabira igre s tri ili s četiri karte. Ako se radijska dugmad ne promijene, koriste se podrazumijevane postavke: dva igrača i tri karte. Klikom miša na dugme *Pokreni* pokreće se forma Poslužitelj, dok forma Vrsta Igre prelazi u pozadinu. Forma Vrsta Igre prikazana je na slici 15.



Slika 15: Forma Vrsta Igre

3.2.9. Cilj igre

Briškula igra se realizira kroz najmanje tri dvoboja. Svaki dvoboj može završiti pobjedom, porazom ili neriješenim ishodom. Cilj samog dvoboja je ostvariti više od 60 punata, čime se ostvaruje pobjeda. Pobjednikom igre se smatra onaj igrač koji:

- prvi ostvari tri pobjede
- ako protivnik napusti igru za vrijeme trajanja iste

3.2.10. Višekorisnička logika

Višekorisnička logika je implementirana za dva moguća slučaja: poslužitelj baca kartu, klijent baca kartu.

Poslužitelj baca kartu

Poslužitelj strana

Forma Poslužitelj pretplaćena je na događaj `MouseClicked` forme IgraTri ili IgraČetiri ovisno o vrsti igre. Klikom na neku od karata okvira (engl. *panel*) *Karte1*, aktivira se događaj `MouseClicked`. Karta se postavlja u kontrolu za prikaz slike (engl. *pictureBox*) koja se nalazi u okviru *table*. Aktivacijom događaja poslužitelj šalje poruku klijentu. Poruka sadrži ime kontrole za prikaz slike nad kojom je izvršena akcija (engl. *action*) *Click*.

Klijent strana

Forma IgraTri / IgraČetiri pretplaćena je na događaj `ClientMove` forme Klijent. Nakon što klijent primi poruku, aktivira se događaj `ClientMove`. Aktivacijom događaja, nad kartom okvira *Karte2* izvršava se akcija *Click*. U metodi `OnMoveMade` je implementirana logika za odabir karte nad kojom će se izvršiti akcija *Click*. Karta se postavlja u kontrolu za prikaz slike okvira *table*. Metoda `OnMoveMade` nalazi se na ispisu 11.

```
public void OnMoveMade(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: OnMoveMade metoda

Klijent baca kartu

Za slučaj *Klijent baca kartu* primijenjena je analogna logika iz slučaja *Poslužitelj baca kartu.*

Opisani slučajevi višekorisničke logike su primijenjeni na forme IgraTri i IgraČetiri.

3.2.11. Dodjela punata i karata

U formama u kojima se odvija *Briškula* implementirana je metoda `pokupi` koja se poziva pri završetku svake odigrane ruke. U metodi `pokupi` implementirana je logika za određivanje pobjednika ruke na temelju odigranih karata. Broj odigranih karata unutar jedne ruke jednak je broju kontrola za prikaz slika unutar okvira *table*. Na temelju ishoda usporedbe, povećava se broj punata pobjednika ruke. Kod forme IgraTri se uspoređuje karta klijenta s kartom poslužitelja, u ovisnosti o adutskoj karti. Kod forme IgraČetiri vrši se usporedba između dvije karte poslužitelja i dvije karte klijenta, u ovisnosti o adutskoj karti.

Postoji više kriterija na temelju kojih se vrši usporedba odigranih karata:

- *Poslužitelj i klijent su odigrali karte adutske boje*
  - pobjednik ruke je igrač koji je odigrao kartu veće vrijednosti
- *Jedan od igrača odigrao kartu adutske boje*
  - pobjednik je igrač koji je odigrao kartu iste boje kao adutska karta
- *Klijent i poslužitelj su odigrali karte iste boje, različite od adutske*
  - pobjednik ruke je igrač koji je odigrao kartu veće vrijednosti
- *Klijent i poslužitelj su odigrali karte različite boje, te različite od adutske karte*
  - pobjednik ruke je igrač koji je prvi odigrao kartu

Na temelju slike 16. implementirana je logika za usporedbu karata.

Karte poredane po vrijednosti
As
Tri
Kralj
Konj
Fanat
Sedam
Šest
Pet
Četiri
Dva

Slika 16: Karte

Broj punata koji se dodjeljuju pobjedniku, sadrži ukupnu vrijednost odigranih karata. Za IgraTri to je vrijednost dvije karte, dok je kod IgraČetiri to vrijednost četiri karte. Vrijednost svih karata je implementirana u metodi `punti`. Na slici 17. prikazana je vrijednost karata na temelju koje je implementirana metoda `punti`.

Karta	Vrijednost
As	11
Trica	10
Kralj	4
Konj	3
Fanat	2
7,6,5,4,2	0

Slika 17: Vrijednost karata

Pobjedniku ruke se prvom dodjeljuje nova karta, a nakon njega poraženom. Tim redoslijedom se dodjeljuje onoliko karata koliko je odigrano u prethodnoj ruci. Prethodno opisana radnja se ponavlja dok jedan od igrača ne ostvari dovoljan broj punata za pobjedu ili dok se ne potroše sve karte špila.

3.2.12. Pohranjivanje ishoda igre

Logika za pohranjivanje ishoda igre implementirana je u formi Poslužitelj.

Mogući ishodi igre:

- *Klijent napušta igru*
  - pobjednik igre je poslužitelj
- *Poslužitelj napušta igru*
  - pobjednik igre je klijent
- *Igrač je ostvario tri pobjede*
  - pobjednik je igrač koji je ostvario tri pobjede

Na ispisu 12. prikazan je kôd za pohranu ishoda *klijent napušta igru.*

```
if(Game_Running){
        using (PlayersEntities1 players = new PlayersEntities1()){
            Game game = players.Games.Find(this.GameId);
         Player player1 = players.Players.FirstOrDefault
                    (r => r.Username == this.logedUser);
         Player player2 = players.Players.FirstOrDefault
                    (r => r.Username ==this.clientUsername);
               game.WinnerId = player1.Id;
               game.Status = "Closed";
               player1.Games.Add(game);
               player2.Games.Add(game);
               players.SaveChanges(); }
               this.Game_Running = false;
               OnFormUnsub("Form"); }
```

Ispis 12: Pohrana ishoda igre

3.2.13. Poslužitelj napušta igru

Poslužitelj strana

Poslužitelj zatvara forumu u kojoj se odvija igra, te šalje poruku klijentu o izlasku iz igre. Forma Poslužitelj prelazi na površinu. Klijent je uklonjen iz liste *Igrači na poslužitelju*, te se novi klijent može povezati s poslužiteljem.

Klijent strana

Klijent je odspojen s poslužitelja. Forma u kojoj se odvija igra se zatvara zajedno s formom Klijent. Iz pozadine na površinu prelazi forma Lista Poslužitelja.

3.2.14. Klijent napušta igru

Klijent strana

Klijent zatvara formu u kojoj se odvija igra, te se automatski odspaja s poslužitelja. Forma Klijent se zatvara, te iz pozadine na površinu prelazi forma Lista Poslužitelja.

Poslužitelj strana

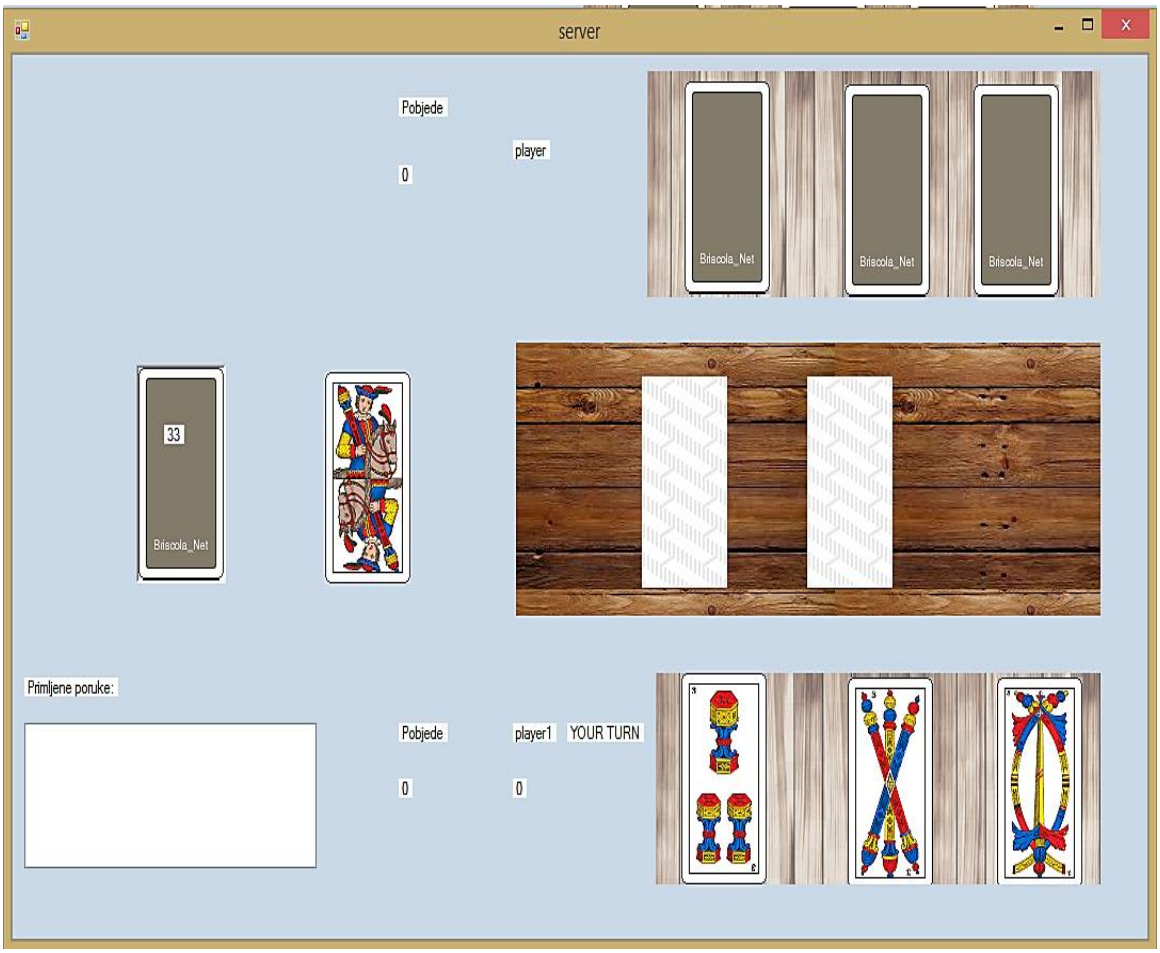
Izlaskom klijenta iz igre aktivira se događaj `ClientDisconnected`. Forma u kojoj se odvija igra se zatvara, te iz pozadine na površinu prelazi forma Poslužitelj. Klijent je uklonjen iz liste *Igrači na poslužitelju*, te se novi klijent može povezati s poslužiteljem.

3.2.13. Forma IgraTri

U formi IgraTri odvija se igra briškula. Okvir *table* sadrži dvije kontrole za prikaz slike, u koje se postavljaju odigrane karte za vrijeme trajanja igre. Okvir *Karte1* sadrži tri kontrole za prikaz slike koje predstavljaju karte trenutno prijavljenog korisnika. Okvir *Karte2* sadrži tri kontrole za prikaz slike koji predstavljaju karte protivnika. U kontrolu za prikaz slike *Igra* postavlja se adutska karta. Kontrola za prikaz slike *Špil* predstavlja špil karata. Forma sadrži naslove (engl. *labels*) koji omogućuju korisniku lakše praćenje tijeka igre. Neki od važnijih naslova jesu sljedeći naslovi:

- *pobjede* – pokazuje broj pobjeda poslužitelja i klijenta
- *punti* – prikazuje broj punata, punti protivnika se ne prikazuju
- *YOUR TURN* – nalazi se pokraj igrača kojeg je red baciti kartu

Forma IgraTri prikazana je na slici 18.

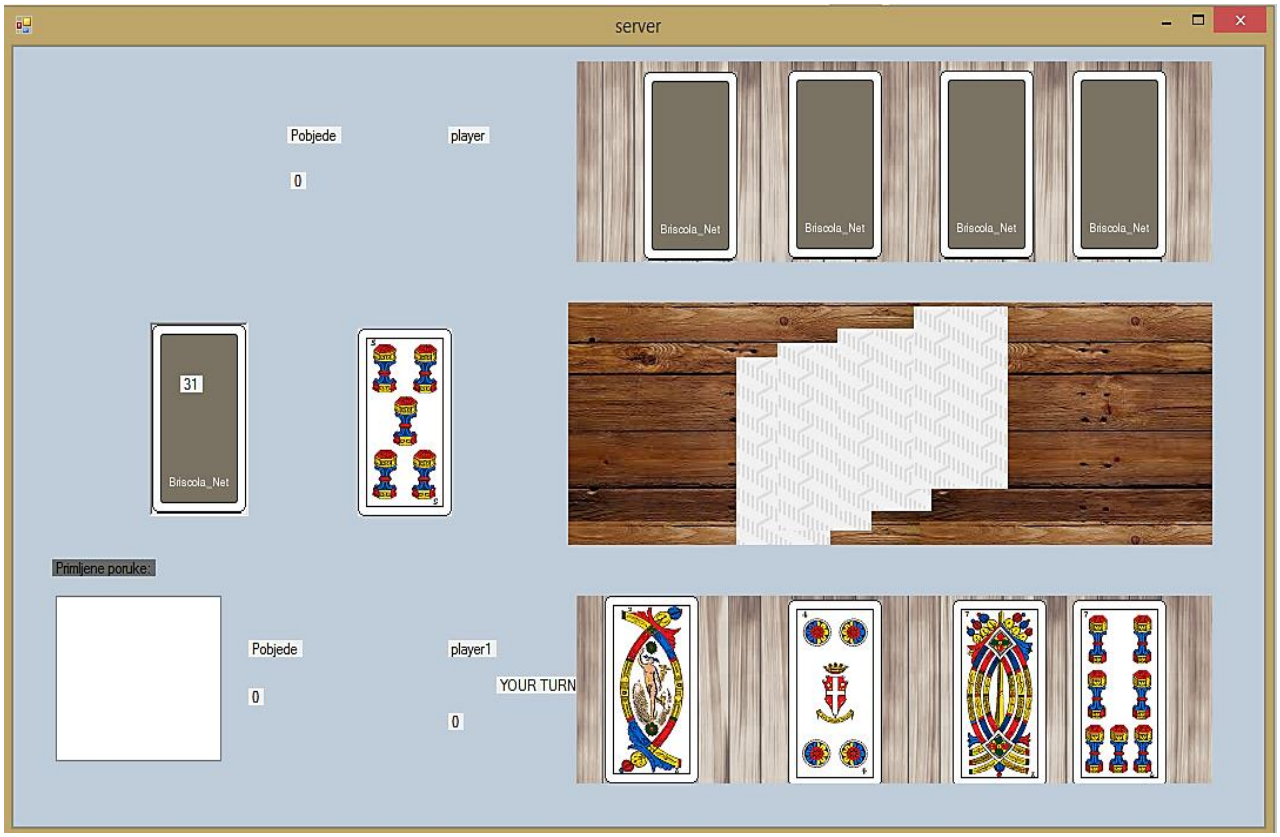


Slika 18: Forma IgraTri

3.2.14. Forma IgraČetiri

Forma IgraČetiri ima istu funkciju kao i forma IgraTri. U formi IgraČetiri se odvija igra u kojoj svaki igrač za vrijeme trajanje jedne ruke odigrava dvije karte. Povećanjem broja odigranih karata u ruci, povećava se broj mogućih ishoda koje je potrebno implementirati u logici same igre. Glavna razlika u pozadinskom kôdu odnosu na formu IgraTri je u implementaciji metoda zaduženih za dodjelu karata i punata.

Osim razlike u pozadinskom kôdu postoje razlike i u grafičkom sučelju. Okvir *table* sadrži četiri kontrole za prikaz slike u koje se pohranjuju odigrane karte za vrijeme trajanja igre. Okvir *Karte1* sadrži četiri kontrole za prikaz slike koji predstavljaju karte trenutno prijavljenog korisnika. Okvir *Karte2* sadrži četiri kontrole za prikaz slike u koje predstavljaju karte protivnika. Forma IgraČetiri prikazana je na slici 19.



Slika 19: Forma IgraČetiri

3.2.15. Forma AboutGame

Forma AboutGame sadrži informacije o nazivu aplikacije, verziji, autoru aplikacije te pravilima igre. Forma AboutGame prikazana je na slici 20.

Slika 20: Forma AboutGame

3.2.16. Forma IgraGotova

Prikazuje se nakon završetka igre. Sadrži kontrolu za prikaz slike koja se mijenja ovisno o ishodu igre.

3.2.17. Klasa Špil

Klasa Špil koristi se za izradu špila karata. Svaka karta opisana je atributima vrijednost i vrsta. Klasa Špil sadrži pobrojane vrijednosti (engl. *enum*) *Vrsta* i *Vrijednost*, što omogućuje jednostavniju izradu karte, a time i samoga špila. Na ispisu 13. prikazana je upotreba pobrojanih vrijednost *Vrsta* i *Vrijednost*.

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

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

Ispis 13: Kôd pobrojane vrijednosti Vrsta i Vrijednost

Metoda `setCards` služi za izradu špila karata. Na ispisu 14. 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 14: setCards metoda

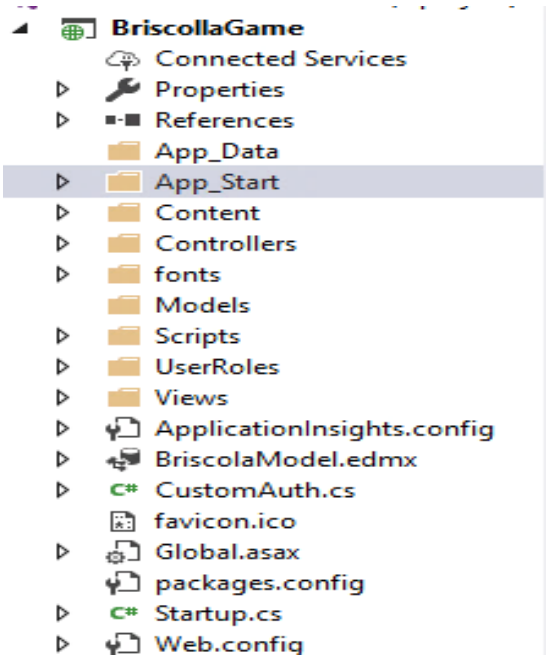
3.2.18. Klasa Igrač

Klasa *Igrač* sadrži atribute *ime*, *punti*, *turn*, *karte*.

3.3. Web aplikacija

3.3.1. Struktura *web* aplikacije

Struktura *web* aplikacije prikazana je na slici 21.



Slika 21: Struktura *web* aplikacije

3.3.2. Prava pristupa

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

Neprijavljeni korisnik pristup

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

Igračev pristup

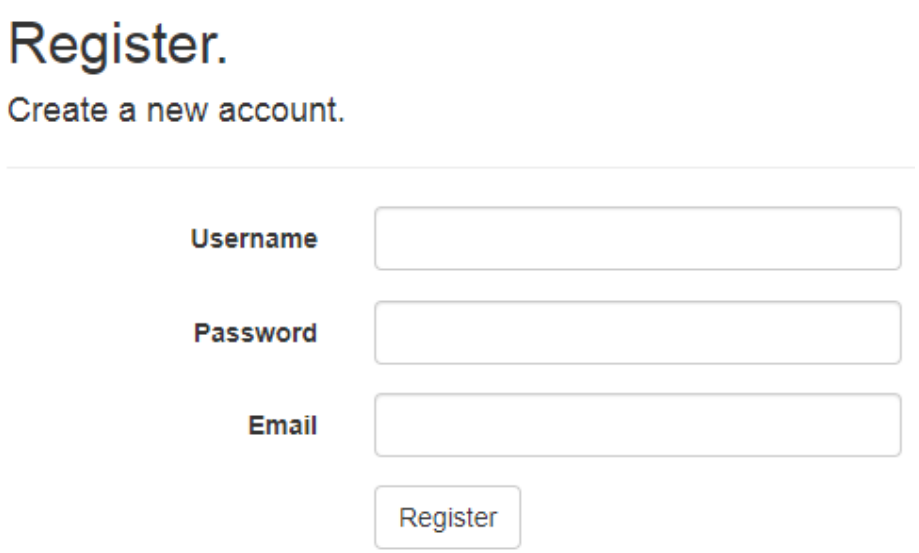
Korisnik može pristupiti onim prikazima predznačenim atributima: `[AllowAnonymous]`, `[CustomAuth(Roles="Player")]`.

Administratorski pristup

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

3.3.3. Prikaz za registraciju

Prikaz za registraciju prikazan je na slici 22.



Slika 22: Prikaz za registraciju

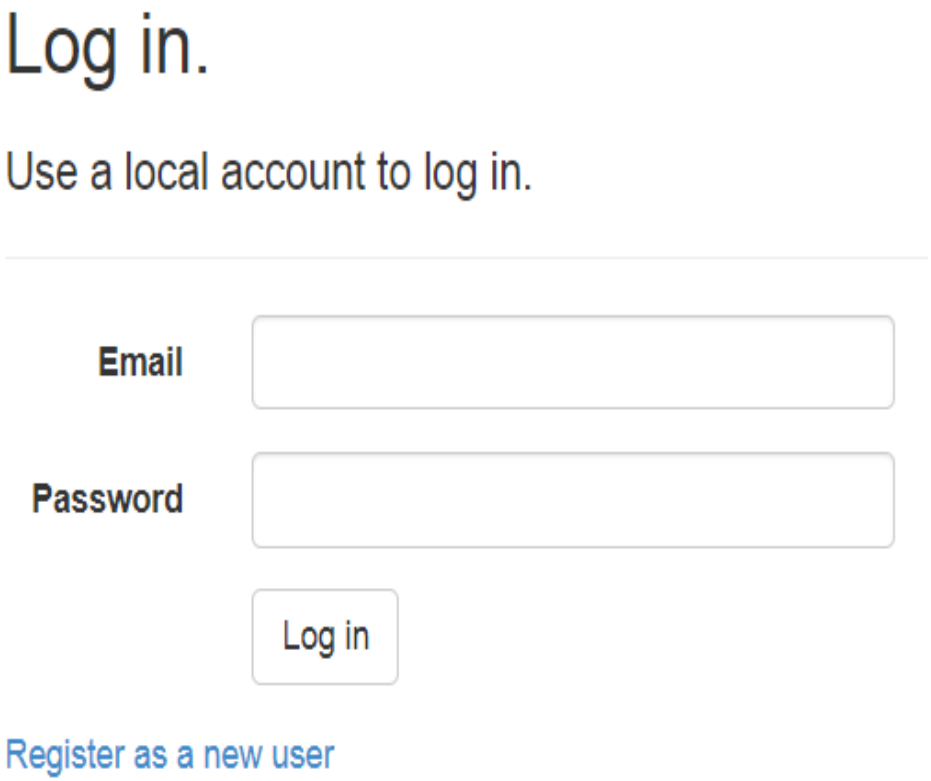
Prikaz za registraciju sastoji se od tekstualnih polja *Username*, *Password* i *Email*. Nakon klika na dugme *Register* provjerava se jesu li zadovoljeni uvjeti za sva tekstualna polja. Uvjeti za uspješnu registraciju korisnika jesu sljedeći:

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

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

3.3.4. Prikaz za prijavu

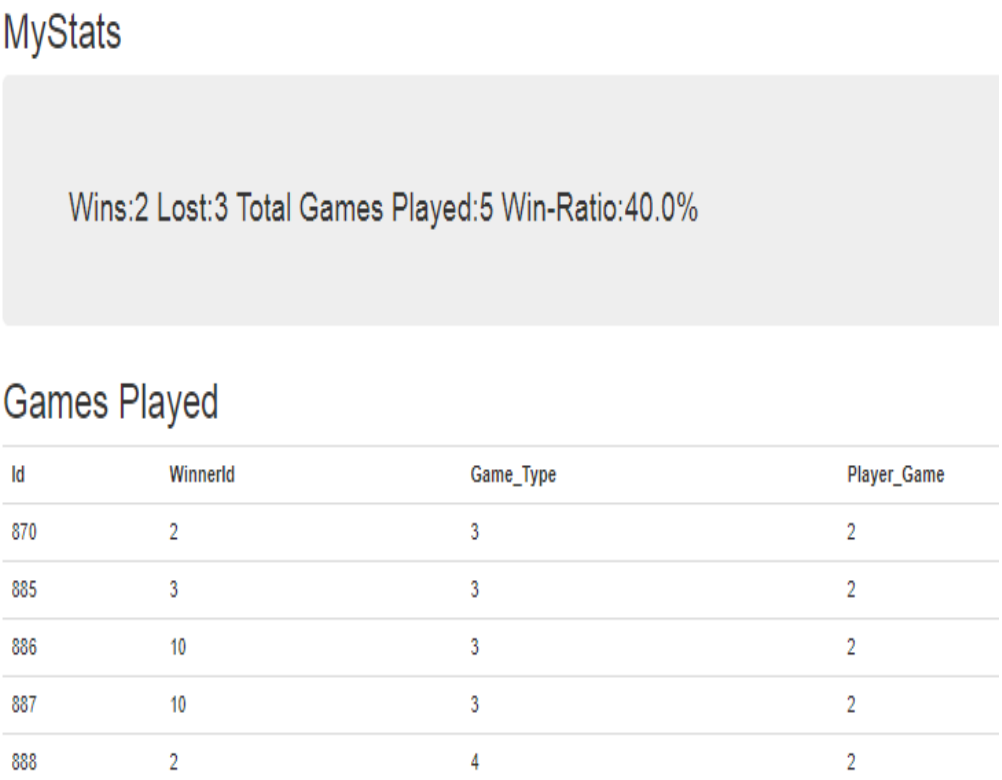
Prilikom pokretanja aplikacije posjetitelj se preusmjerava na formu za prijavu. Prikaz za prijavu korisnika sastoji se od tekstualnih polja *Email* i *Password*, dugmeta *Log in* te poveznice *Register*. Posjetitelji koji ne posjeduju račun mogu preko poveznice pristupiti prikazu za registraciju. Prikaz za prijavu prikazan je na slici 23.



Slika 23: Prikaz za prijavu

3.3.5. Prikaz MyStats

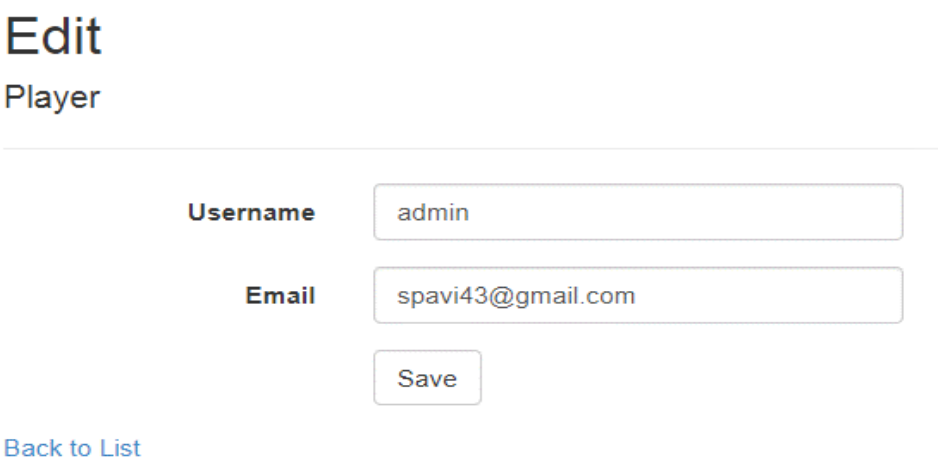
MyStats prikazuje broj odigranih partija, broj pobjeda i poraza te ukupni postotak pobjeda trenutno prijavljenog korisnika. U listi *Games Played* sadržana je kompletna povijest igara u kojima je korisnik sudjelovao te ishod pojedine igre. MyStats prikazan je na slici 24.



Slika 24: Prikaz MyStats

3.3.6. Prikaz Account Manager

Account Manager prijavljenim korisnicima omogućuje promjenu korisničkog imena i elektroničke pošte. Klikom na dugme *Save* promjene se pohranjuju u bazu podataka. Account Manager prikazan je na slici 25.



Slika 25: Prikaz Account Manager

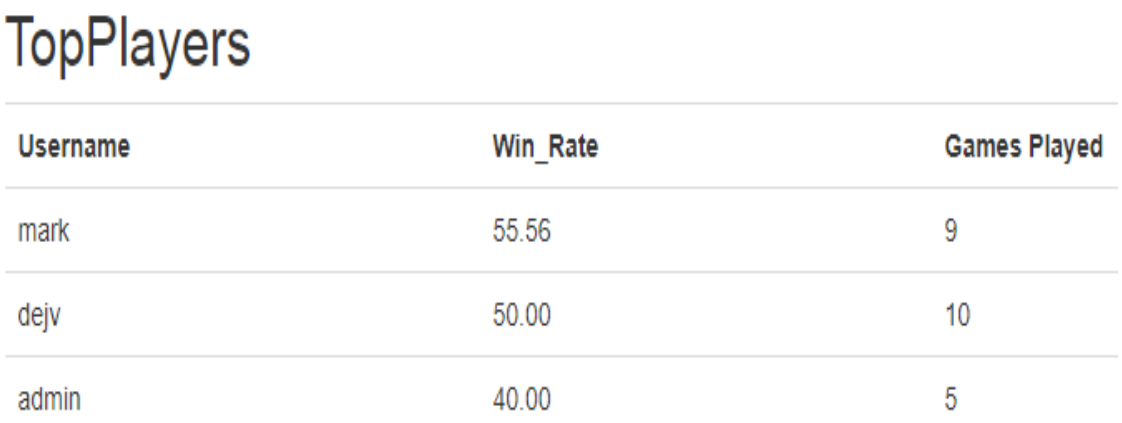
3.3.7. Prikaz Lista igrača

U prikazu Lista igrača sadržan je popis svih korisnika, s povjerljivim informacijama. Lista igrača pokraj podataka svakog korisnika sadrži poveznice *Edit, Details* i *Delete*. Preko prikaza Lista igrača administrator može kreirati nove korisnike, brisati postojeće, pregledati detaljne informacije o korisniku te promijeniti korisnikov profil. Lista igrača prikazana je na slici 26.

Slika 26: Prikaz Lista igrača

3.3.8. Prikaz TopPlayers

Prikaz TopPlayers sadrži listu najbolje rangiranih igrača prema povijesti odigranih igara. Prikaz TopPlayers prikazan je na slici 27.



Slika 27: Prikaz TopPlayers

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 15.

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

Ispis 15: Kôd Upit LINQ-a

4. ZAKLJUČAK

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

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

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

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

5. LITERATURA

[1] Zbornik Veleučilišta u Rijeci, 2014, vol. 2, no. 1, https://www.veleri.hr/files/datoteke/knjige/digi/2014_06_01_Veleuciliste%20Zbornik.pdf (posjećeno 15. 7. 2019.)

[2] Smijulj, A. i Meštrović, A.: Izgradnja MVC modularnog radnog okvira, Zbornik Veleučilišta u Rijeci, 2014, vol. 2, no. 1, str. 215-232, https://hrcak.srce.hr/file/190412 (posjećeno 28. 6. 2019.)

[3] Uvod u ASP .Net MVC, http://www.manuelradovanovic.com/2017/09/uvod-u-asp-net-mvc.html (posjećeno 25. 6. 2019.)

[4] Instalacija razvojnog okruženja, https://petlja.org/biblioteka/r/lekcije/prirucnik-cpp/instalacija-razvojnog-okruzenja (posjećeno 25.6.2019.)

[5] SQL Server Management Studio, https://www.guru99.com/sql-server-management-studio.html#2 (posjećeno 24.6.2019.)

[6] Modeliranje, implementacija i administracija baza podataka, https://bib.irb.hr/datoteka/979191.Modeliranje_implementacija_i_administracija_baza_podataka.pdf (posjećeno 27.7.2019)

[7] MS SQL SERVER (2018) - Osnove upravljanja serverskim nalozima, https://programirajmozajedno.wordpress.com/2016/09/13/ms-sql-server-logins/ (posjećeno 21.7.2019)

[8] Kratofil, D.: Izrada *web* aplikacije u programskom jeziku C# (Završni rad), Fakultet elektrotehnike, računarstva i informacijskih tehnologija Sveučilišta Josipa Jurja Strossmayera u Osijeku, Osijek, 2017, https://urn.nsk.hr/urn:nbn:hr:200:999545 (posjećeno 11. 7. 2019.)

[9] Rad s bazom podataka, https://www.scribd.com/document/335756231/RPA-P5-Rad-s-Bazom-Podataka (posjećeno 20.6.2019)

[10] Detaljni pregled LINQ – Integrirani SQL upiti u .NET programskom jeziku, https://bhrnjica.net/2010/10/26/detaljni-pregled-linq-integrirani-sql-upiti-u-net-programskom-jeziku/ (posjećeno 15. 7. 2019.)

[11] From zero to hero in Json with C#, https://www.c-sharpcorner.com/article/from-zero-to-hero-in-json-with-c-shar/ (posjećeno 25. 4. 2019.)

[12] Lovrić, V: Windows forme u C#-u (Završni rad), Fakultet elektrotehnike, računarstva i informacijskih tehnologija Sveučilišta Josipa Jurja Strossmayera u Osijeku, Osijek, 2016, https://urn.nsk.hr/urn:nbn:hr:200:800058 (posjećeno 17. 7. 2019.)

6. POPIS SLIKA I ISPISA

Popis slika:

Popis ispisa: