II faza

Projekat: igra Uno

Predmet: Arhitektura i projektovanje softvera

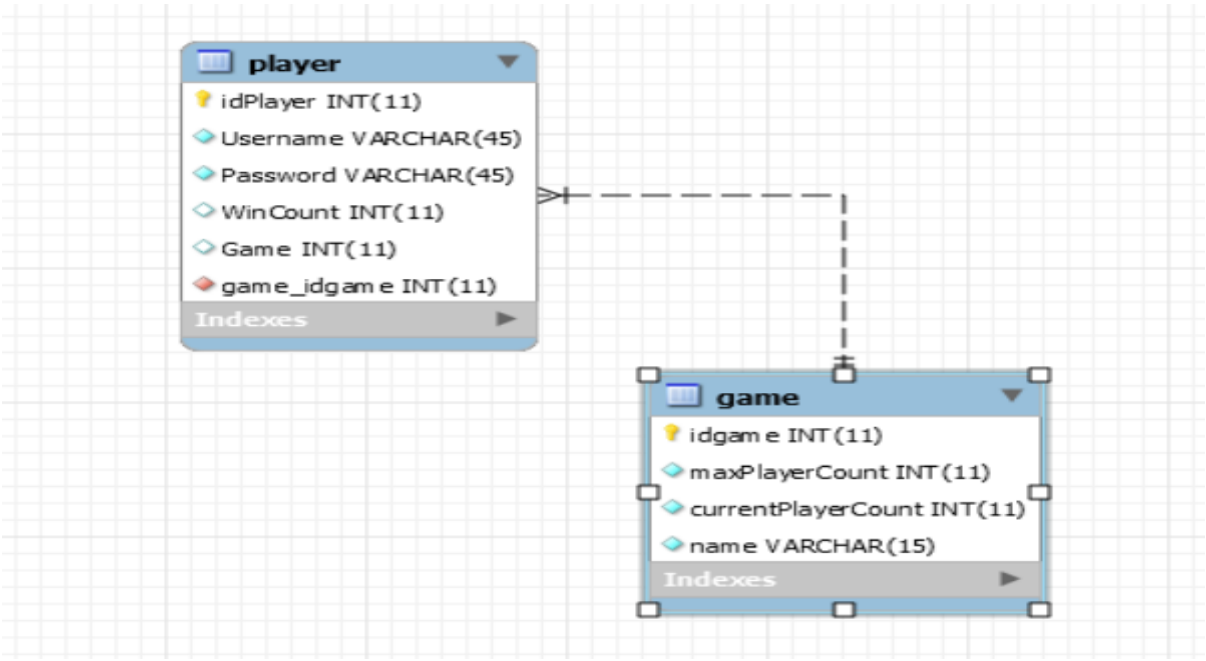
Nikola Popović, 15826

Dejan Randjelović, 15843

Sadrzaj

1. Struktura baze podataka

Ispod je prikazana veza izmedju entiteta u bazi podataka.



2. Klase

Ispod su prikazane klase koje odgovaraju entitetima u bazi podataka.

Player

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace UnoTest.Entities
8  {
9      class Player
10     {
11         public virtual int id { get; set; }
12         public virtual string username { get; set; }
13         public virtual string password { get; set; }
14         public virtual int winCount { get; set; }
15         public virtual Game game { get; set; }
16     }
17 }
18
```

Game

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace UnoTest.Entities
{
    class Game
    {
        public virtual int id { get; set; }
        public virtual int maxPlayerCount { get; set; }
        public virtual int currentPlayerCount { get; set; }
        public virtual string name { get; set; }
        public virtual IList<Player> players { get; set; }
        public Game()
        {
            players = new List<Player>();
        }
    }
}
```

3. Mapiranje klasa

PlayerMapping

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using FluentNHibernate.Mapping;

namespace UnoTest.EntitiesMapping
{
    class PlayerMapping : ClassMap<Entities.Player>
    {
        public PlayerMapping()
        {
            Table("PLAYER");
            Id(x => x.id, "IDPLAYER").GeneratedBy.TriggerIdentity();
            Map(x => x.username, "USERNAME");
            Map(x => x.password, "PASSWORD");
            Map(x => x.winCount, "WINCOUNT");
            References(x => x.game).Column("GAME").Not.LazyLoad();
        }
    }
}
```

GameMapping

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using FluentNHibernate.Mapping;

namespace UnoTest.EntitiesMapping
{
    class GameMapping : ClassMap<Entities.Game>
    {
        public GameMapping()
        {
            Table("GAME");
            Id(x => x.id, "IDGAME").GeneratedBy.TriggerIdentity();
            Map(x => x.maxPlayerCount, "MAXPLAYERCOUNT");
            Map(x => x.currentPlayerCount, "CURRENTPLAYERCOUNT");
            Map(x => x.name, "NAME");
            HasMany(x => x.players).KeyColumn("GAME").Not.LazyLoad().Inverse().Cascade.All();
        }
    }
}
```

4. Funkcije

Dodavanje novog igraca u sistem

```
try
{

    ISession s = DataLayer.GetSession();

    Entities.Player p = new Entities.Player();
    p.username = txtUsername.Text;
    p.password = txtPassword.Text;
    p.winCount = 0;
    s.Save(p);
    s.Flush();
    IQuery q = s.CreateQuery("from Player");

    IList<Player> py = q.List<Player>();
    listPlayers.Items.Clear();
    foreach (Player x in py)
    {
        ListViewItem item = new ListViewItem(new string[] {  x.id.ToString() });
        item.Tag = x;
        listPlayers.Items.Add(item);

        item = new ListViewItem(new string[] { x.username.ToString() });
        item.Tag = x;
        listPlayers.Items.Add(item);

        item = new ListViewItem(new string[] { x.winCount.ToString() });
        item.Tag = x;
        listPlayers.Items.Add(item);
    }
    listPlayers.Refresh();


    s.Close();
```

Dodavanje nove igre u sistem

```
private void btnAddGame_Click(object sender, EventArgs e)
{


    try
    {
        ISession s = DataLayer.GetSession();
        IQuery q = s.CreateQuery("from Game");
        string name = txtName.Text;
        Entities.Game g = new Entities.Game();
        g.name = name;
        s.Save(g);
        s.Flush();

        IList<Game> gm = q.List<Game>();

        listGames.Items.Clear();
        foreach (Game x in gm)
        {
            ListViewItem item = new ListViewItem(new string[] { x.id.ToString() });
            item.Tag = x;
            listGames.Items.Add(item);

            item = new ListViewItem(new string[] { x.name.ToString() });
            item.Tag = x;
            listGames.Items.Add(item);
        }
        listGames.Refresh();
        s.Close();
    }
```

Brisanje svih igraca

```
private void btnClearPlayers_Click(object sender, EventArgs e)
{


    try
    {
        ISession s = DataLayer.GetSession();
        IQuery q = s.CreateQuery("from Player");
        IList<Player> py = q.List<Player>();
        listPlayers.Items.Clear();

        //brise se objekat iz baze ali ne i instanca objekta u memroiji
        //s.Delete(gm);
        s.Delete("from Player");

        s.Flush();
        s.Close();
        listPlayers.Refresh();
    }
    catch (Exception ec)
    {
        MessageBox.Show(ec.Message);
    }
}
```

Brisanje svih partija

```
private void btnDeleteGames_Click(object sender, EventArgs e)
{

    try
    {
        ISession s = DataLayer.GetSession();
        IQuery q = s.CreateQuery("from Game");
        IList<Game> gm = q.List<Game>();
        listGames.Items.Clear();

        //brise se objekat iz baze ali ne i instanca objekta u memroiji
        //s.Delete(gm);
        s.Delete("from Game");

        s.Flush();
        s.Close();

        listGames.Refresh();

    }
```

Predstavljanje igraca i igara u sistemu pri ucitavanju forme

```
UnoTest.Form1
private void Form1_Load(object sender, EventArgs e)
{
    ISession s = DataLayer.GetSession();
    IQuery q = s.CreateQuery("from Game");
    IList<Game> gm = q.List<Game>();
    listGames.Items.Clear();
    foreach (Game x in gm)
    {
        ListViewItem item = new ListViewItem(new string[] { x.id.ToString() });
        item.Tag = x;
        listGames.Items.Add(item);

        item = new ListViewItem(new string[] { x.name.ToString() });
        item.Tag = x;
        listGames.Items.Add(item);
    }
    listGames.Refresh();
    q = s.CreateQuery("from Player");
    IList<Player> py = q.List<Player>();
    listPlayers.Items.Clear();
    foreach (Player x in py)
    {
        ListViewItem item = new ListViewItem(new string[] { x.id.ToString() });
        item.Tag = x;
        listPlayers.Items.Add(item);

        item = new ListViewItem(new string[] { x.username.ToString() });
        item.Tag = x;
        listPlayers.Items.Add(item);

        item = new ListViewItem(new string[] { x.winCount.ToString() });
        item.Tag = x;
        listPlayers.Items.Add(item);
    }
    listPlayers.Refresh();
    s.Close();
```

Azuriranje sifre igraca sa zadatim username-om.

```
private void btnUpdatePlayer_Click(object sender, EventArgs e)
{

    try
    {
        ISession s = DataLayer.GetSession();
        string username = txtUsername.Text;
        IQuery q = s.CreateQuery("from Player p where p.username=:username");
        q.SetString("username", username);
        Player p = q.UniqueResult<Player>();
        s.Close();

        //objekat se modifikuje potpuno nezavisno od sesije
        p.password = tbxNewPassword.Text;

        //otvara se nova sesija
        ISession s1 = DataLayer.GetSession();

        //poziva se Update i objekat se povezuje sa novom sesijom
        s1.Update(p);

        s1.Flush();
        s1.Close();

        listPlayers.Refresh();
    }
    catch (Exception ec)
    {
        MessageBox.Show(ec.Message);
    }
}
```