COLEGIUL NAŢIONAL DE INFORMATICĂ

„GRIGORE MOISIL” BRAŞOV

LUCRARE PENTRU ATESTAREA COMPETENŢELOR PROFESIONALE

Șah

Elev

Shaikh Saud

Clasa

12 E

Profesor îndrumător

Șerban Manuela

Nițulescu Laura

Mai 2023

Cuprins

1. Motivarea alegerii temei

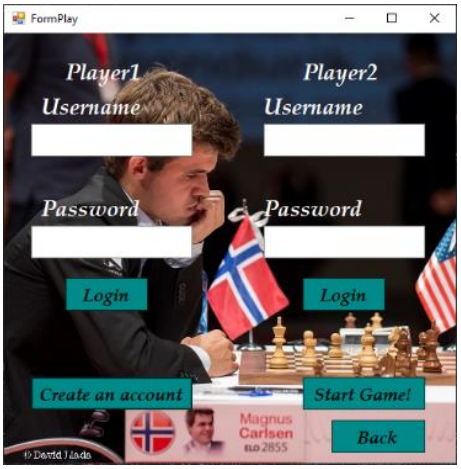
În perioada recentă, șahul a crescut în popularitate din cauza rețelelor de socializare promovând-ul tot mai mult. Având din trecut o pasiune pentru șah am recăpatat astfel un interes pentru acest joc și am decis să fac un șah dispus într-o aplicație de tip WindowsForms, având cunoștințele de C# și Sql Server reînnoite cu participarea la Olimpiada De Tehnologie a Informației – secțiunea C# și pregătirea pentru examene din aceste materii.

2. Prezentarea aplicației

Aplicația constituie un joc de șah care este jucat la același calculator de 2 persoane. La deschiderea aplicației, utilizatorul va fi întâmpinat cu o pagină principală care are 3 butoane: Play – te duce la organizarea unui joc, How to Play – reguli ale jocului, Exit – ieșire totală din aplicație(închiderea acesteia) .

*Toate butoanele aplicației se aprind schimbându-și culoarea când mousul este ținut peste ele.

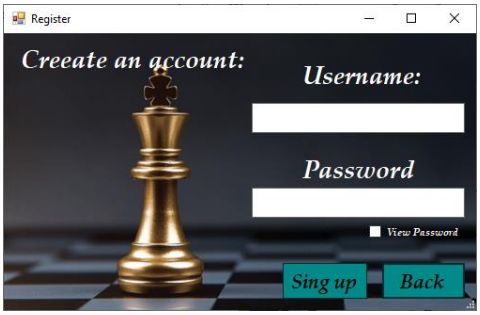
Organizarea unui joc:



Pentru a începe un joc, 2 utilizatori trebuie să fie conectați. Pentru a se conecta, aceștia trebuie să introducă numele și parola respectivă. În cazul în care nu au un cont sau vor să-și facă altul, aceștia pot să apese butonul „Create an account”. După ce 2 utilizatori sunt logați, butonul „Start game!” poate fi apăsat pentru a începe un joc nou. Butonul „Back” va duce înapoi la pagina principală. Utilizatorilor logați li se va afișa numele, numărul de meciuri jucate, numărul de meciuri câștigate și rata de câștig. Partea pe care va juca fiecare va fi decisă aleatoriu. Timpul pe care fiecare îl are de la start este de 30 de minute.

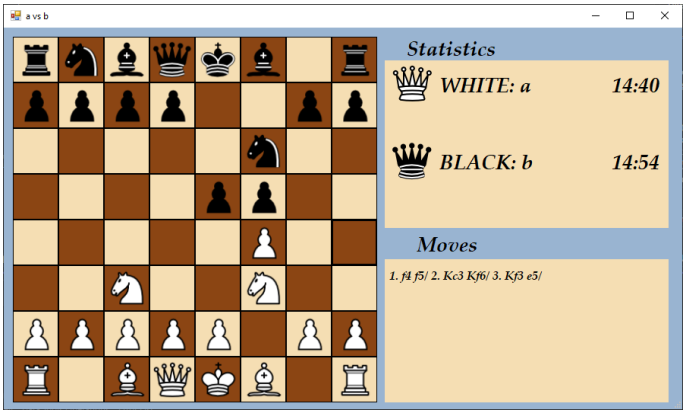
Înregistrare:

Formul dedicat înregistrării unui utilizator nou este simplu: acesta trebuie să introducă un nume unic și parolă. Casuța „View Passowrd” poate fi bifată pentru a arăta parola, iar butonul „Sing Up” va creea un utilizator nou cu datele introduse(dacă acestea sunt valide), și va arăta un



mesaj de succes. Butonul back închide înregistrarea și redeschide meniul cu organizarea unui joc.

Jocul în sine:



Acesta se va desfășura într-un form separat și funcționeaza ca un joc normal de șah. Titlul formului va fi redat de numele celor 2 jucători. La statistici se vor afișa numele jucătorilor, partea pe care joacă, timpul rămas și piesele luate. La mișcari vor fi afișate sub formă prescurtată mutările duse de aceștia. Jocul se termină odată ce timpul unui jucător s-a scurs sau și-a luat șah mat. Închiderea formului pe parcursul meciului de șah va anula meciul.

3. Detalii de implementare

La construcția aplicației am folosit C# și Windows Forms pentru logica jocului și interfață, și Sql Server pentru gestionarea unei baze de date.

La deschiderea aplicației se va realiza conexiunea la o bază de date care conține datele tuturor utilizatorilor. De asemenea un timer va fi pornit care cheamă garbage collector-ul din C# la câte o secundă să se ocupe de reducerea memoriei utilizate de program.

```
SqlConnection conn;
 public Form1()
 {
     InitializeComponent();
     deschidere_conexiune();
     panelLogare.Visible = false;
     panelP2.Visible = false;
     panelP1.Visible = false;
     Timer t1 = new Timer();
     t1.Interval = 1000;
     t1.Tick += new EventHandler(t1_tick);
     t1.Start();
     pictureBox1.Visible = false;
 }
 public void t1_tick(object sender, EventArgs e)
 {
     GC.Collect();
 }
 public void deschidere_conexiune()
 {
     conn = new SqlConnection(@"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=|DataDirectory|\Sah.mdf;Integrated
Security=True;Connect Timeout=30");
     conn.Open();
 }
```

Adăugarea unei noi înregistrări în baza de date: Se creează o comandă sql care verifică dacă utilizatorul este unic in baza de date și în cazul în care este va creea o nouă înregistrare.

```
private void buttonSignUp_Click(object sender, EventArgs e)
 {
     SqlCommand cmd = new SqlCommand("SELECT IdUtilizator FROM Utilizatori WHERE
NumeUtilizator = @Name", conn);
     cmd.Parameters.AddWithValue("@Name", textBoxNume.Text);
     var x = cmd.ExecuteScalar();
     if (textBoxParola.Text == "" || textBoxNume.Text == "")
     {
         MessageBox.Show("Please enter an username/password!");
         return;
     }
     if (x != null)
     {
         MessageBox.Show("Username already used!");
         textBoxNume.Clear();
         textBoxParola.Clear();
         return;
```

```
            }
            cmd = new SqlCommand("INSERT INTO Utilizatori VALUES(@Nume, @Parola, 0, 0)",
conn);
            cmd.Parameters.AddWithValue("@Nume", textBoxNume.Text);
            cmd.Parameters.AddWithValue("@Parola", textBoxParola.Text);
            cmd.ExecuteNonQuery();
            MessageBox.Show("A new account has been created.");
            textBoxNume.Clear();
            textBoxParola.Clear();
            checkBoxView.Checked = false;
        }
```

Setarea tablei de șah: Se folosește o matrice de butoane care este editată astfel încat să dea impresia unei table de șah și care va fi adăugată într-un panou în form.

```
public void set_board()
        {
            for(int i = 0; i < 8; i++)
            {
                for(int j = 0; j < 8; j++)
                {
                    chess_board[i, j] = new Button();
                    chess_board[i, j].Size = new Size(60, 60);
                    chess_board[i, j].Tag = get_piecetype(i, j);
                    chess_board[i, j].BackColor = get_backcolor(i, j);
                    chess_board[i, j].BackgroundImage = get_image(i, j);
                    chess_board[i, j].BackgroundImageLayout = ImageLayout.Stretch;
                    chess_board[i, j].Location = new Point(60*j, 60*i);
                    chess_board[i, j].FlatStyle = FlatStyle.Flat;
                    chess_board[i, j].Click += new EventHandler(move_click);
                    this.panelGame.Controls.Add(chess_board[i, j]);
                }
            }
            for(int i = 0; i < 8; i++)
            {
                did_first_move[0, i] = false;
                did_first_move[1, i] = false;
            }
        }
```

Logica din spatele mutărilor pieselor se află în spatele funcției move_click, iar piesele de șah au fiecare funcțiile lor respective care sunt apelate când piesa este selectată și un loc este ales. La fiecare mișcare se va verifica dacă regele poziției care nu mută se va afla in șah după mișcare.

```
public void move_click(object sender, EventArgs e)
        {
            Button b = (Button)sender;
            string new_tag = b.Tag.ToString();
            string my_tag = chess_board[selectedI, selectedJ].Tag.ToString();
            Point p = get_point(new_tag);
            int i = p.X - 1, j = p.Y - 1;
            if (is_selected == 1)
            {
                if (can_move(i, j))
                {
                    if(get_piecetype(i, j).Split(' ')[1] != "Space")
                    {
```

```
                    if (active_side == "White")
                        whiteTaken[blackLost++].BackgroundImage = get_image(i, j);
                    else
                        blackTaken[whiteLost++].BackgroundImage = get_image(i, j);
                }
                string acro = "";
                if (turn % 2 == 1)
                    labelTurns.Text += (turn + 1) / 2 + ".";
                if(active_piece != "Pawn")
                    acro += active_piece[0];
                labelTurns.Text += $" {acro}{(char)('a' + j)}{8 - i}";
                if(turn % 2 == 0)
                    labelTurns.Text += "/ ";
                if (turn % 8 == 0)
                    labelTurns.Text += '\n';
                b.BackgroundImage = chess_board[selectedI,
selectedJ].BackgroundImage;
                chess_board[selectedI, selectedJ].BackgroundImage = null;
                b.Tag = my_tag;
                chess_board[selectedI, selectedJ].Tag = "Neutral Space " +
selectedI + selectedJ;
                turn++;
            }
            else if (new_tag.Split(' ')[0] == active_side)
            {
                set_activepiece(i, j, new_tag);
                return;
            }
            is_selected = 0;
        }
        else
        {
            if (turn % 2 == 1 && new_tag.Split(' ')[0] == "Black" || turn % 2 == 0
&& new_tag.Split(' ')[0] == "White" || new_tag.Split(' ')[0] == "Neutral")
            {
                return;
            }
            else
            {
                is_selected = 1;
                set_activepiece(i, j, new_tag);
            }
        }
    }
```

Pionul merge doar în față cu câte o pătrățică(caz particular când e prima sa mișcare și se poate mișca două) sau în diagonală dacă poate ataca. Direcția acestuia va fi respectivă părții pe care se află, iar când ajunge în celălalt capăt al tablei se va putea alege ce piesă trebuie să devină.

```
bool check_pawn(int i, int j)
    {
        int k = (active_side == "White") ? (-1) : 1, z = (active_side ==
"White") ? (0) : (1);
        if (i == selectedI + 2 * k && j == selectedJ && is_space(i, j) &&
did_first_move[z, get_ord(selectedI, selectedJ)] == false)
        {
            did_first_move[z, get_ord(selectedI, selectedJ)] = true;
            return true;
        }
```

```
            if (i == selectedI + k && (j == selectedJ - 1 || j == selectedJ + 1)
&& !is_space(i, j))
            {
                did_first_move[z, get_ord(selectedI, selectedJ)] = true;
                return true;
            }
            if(i != selectedI+k || j != selectedJ)
                return false;
            if(i == selectedI+k && j == selectedJ && !is_space(i, j))
                return false;
            did_first_move[z, get_ord(selectedI, selectedJ)] = true;
            return true;
        }
```

Regele și Calul verifică daca pozițiile pe care se pot mișca sunt libere și valide. Turnul și nebunul verifică dacă pozițiile se află pe liniile/diagonalele respective și dacă se află înainte de primul obstacol întâmpinat. Regina folosește funcțiile pentru nebun si tură pentru a verifica dacă poziția pe care trebuie să fie pusă este validă.

```
bool check_rook(int i, int j)
        {
            if(i != selectedI && j != selectedJ)
            {
                return false;
            }
            int u = selectedI-1, d = selectedI+1, l = selectedJ-1, r = selectedJ+1;
            if(u > 0) for (; is_space(u, selectedJ) && u > 0; u--);
            if(d < 7) for (; is_space(d, selectedJ) && d < 7; d++);
            if(l > 0) for (; is_space(selectedI, l) && l > 0; l--);
            if(r < 7) for (; is_space(selectedI, r) && r < 7; r++);
            if (i < u || i > d)
            {
                return false;
            }
            if (j < l || j > r)
            {
                return false;
            }
            return true;
        }
bool check_bishop(int i, int j)
        {
            if (i + j != selectedI + selectedJ && i - j != selectedI - selectedJ)
                return false;
            int TLi = selectedI - 1, TLj = selectedJ - 1;
            int TRi = selectedI - 1, TRj = selectedJ + 1;
            int BLi = selectedI + 1, BLj = selectedJ - 1;
            int BRi = selectedI + 1, BRj = selectedJ + 1;
            if(TLi > 0 && TLj > 0) for (;is_space(TLi, TLj) && TLi > 0 && TLj > 0;
TLi--, TLj--);
            if(TRi > 0 && TRj < 7) for (;is_space(TRi, TRj) && TRi > 0 && TRj < 7;
TRi--, TRj++);
            if(BLi < 7 && BLj > 0) for (;is_space(BLi, BLj) && BLi < 7 && BLj > 0;
BLi++, BLj--);
            if(BRi < 7 && BRj < 7) for (;is_space(BRi, BRj) && BRi < 7 && BRj < 7;
BRi++, BRj++);
```

```
            if (i < TLi && j < TLj || i < TRi && j > TRj || i > BLi && j < BLj || i >
BRi && j > BRj)
                return false;
            return true;
        }
```

Timpul jucătorilor este gestionate prin 3 timere care se opresc și pornesc in funcițe de jucătorul care joacă. Toată logica din timpului se regasește în funcția set_time.

```
public void set_time()
        {
            timerWhite.Interval = 1000;
            timerWhite.Tick += new EventHandler(timerTime_tick);
            timerBlack.Interval = 1000;
            timerBlack.Tick += new EventHandler(timerTime_tick);
            timerTurns.Interval = 10;
            timerTurns.Tick += new EventHandler(timerTurns_tick);
            timerTurns.Start();
        }
        public void timerTime_tick(object sender, EventArgs e)
        {
            if(turn % 2 == 1)
            {
                timeWhite--;
                labelTimeWhite.Text = $"{timeWhite/60}:{timeWhite%60}";
            }
            else
            {
                timeBlack--;
                labelTimeBlack.Text = $"{timeBlack/60}:{timeBlack%60}";
            }
        }
        public void timerTurns_tick(object sender, EventArgs e)
        {
            if (turn % 2 == 1)
            {
                timerBlack.Stop();
                timerWhite.Start();
            }
            else
            {
                timerBlack.Start();
                timerWhite.Stop();
            }
        }
```

4. Posibilități de dezvoltare

O primă posibilitate de dezvoltare este adăugarea unui un AI de șah pe mai multe dificultăți cu care un utilizator se poate confrunta pentru a-și testa cunoștintele. AI-ul va fi implementat prin intermediul unui algoritm care găsește cea mai bună mișcare pentru dificultatea respectivă sau se va folosi un API prin care utilizatorul poate comunica cu un bot de șah, trimițând și primind mișcări.

O altă posibilitate de dezvoltare constă în mutarea jocului pe un server la care aplicația să fie constant conectată astfel încât 2 utilizatori să se poată loga și înfrunta de pe dispozitive diferite. Odată cu posibilitatea ca 2 utilizatori să poată să se înfrunte de la distanță, aplicația va avea și un sistem care va clasifica jucătorii după abilitățile și cunoștințele lor(similar cu Elo-ul de pe Chess.com).

Pe lângă toate acestea vreau să adaug și un sistem de minigame-uri pe care un utilizator le poate juca precum puzzle-uri și 4-player chess.

5. Resurse hardware și software necesare

- Calculator tip desktop sau laptop
- Procesor AMD sau Intel, Dual Core CPU
- Memorie RAM 2 GB minim
- 50 MB memorie liberă pe disc
- Sistem de operare Windows 10/8.1/8/7
- C# 4.0(.NET Framework 4.0) sau mai nou instalat pe calculator

6. Bibliografie

- Documentație C# de la Microsoft - https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/
- Documentație Windows Forms de la Microsoft - https://learn.microsoft.com/en-us/dotnet/desktop/winforms/?view=netdesktop-7.0
- Tutorial SQL W3Schools - https://www.w3schools.com/sql/default.asp
- Microsoft – Manual Programare Orientată pe Obiecte și Programare Vizuală in .Net Framework
- Joyce Farrell – Microsoft Visual C#: An Introduction to Object-Oriented Progranming