```
using Microsoft.Win32;
using MyNeuronNetworkWin.Model;
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Net;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using System.Drawing;
using MyNeuronNetworkWin.Models;
using Microsoft.VisualBasic;

namespace MyNeuronNetworkWin
{
    /// <summary>
    /// Логика взаимодействия для MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window
    {
        public NeuronNetworkBaseEntities1 nnentities;

        public MainWindow()
        {
            InitializeComponent();
            #region Подключение БД
            try
            {
                nnentities = new NeuronNetworkBaseEntities1();
                #endregion


                #region ListBox
                foreach (var name in nnentities.Info) actors_ListBox.Items.Add(name);
                #endregion


            }
            catch (Exception)
            {
                MessageBox.Show("Ошибка при подключении к БД! Обратитесь к разработчику",
"MyNeuronNetworkWin", MessageBoxButton.OK, MessageBoxImage.Error);
                Close();
            }
        }

        private void Button_Click(object sender, RoutedEventArgs e)
        {
            actors_ListBox.SelectedIndex = 0;
            link_TextBox.Text = "";
            name_TextBox.Text = "";
        }

        private void Button_Click_1(object sender, RoutedEventArgs e)
        {
            try
            {
                if (link_TextBox.Text != "")
                {
                    int neuronCount = 10; int width = 750; int height = 750;
```

```
                    ImageNeuronNetwork net = new ImageNeuronNetwork(neuronCount, width, height);
                    using (WebClient webClient = new WebClient())
                    {
                        webClient.DownloadFile(link_TextBox.Text, "digit.bmp");
                    }
                    var img = System.Drawing.Image.FromFile("digit.bmp");
                    var answer = net.getAnswer(img.ToInput(width, height));

                    bool check = false;

                    if (name_CheckBox.IsChecked == false)
                    {
                        foreach (var name in nnentities.Info)
                        {
                            if (name.IdNeuronNetwork == answer) //сравнивает существующий код в БД
со считываемым значением с изображения
                            {
                                var qustion = MessageBox.Show($"Это - {name.Name}. Правильно?",
"MyNeuronNetworkWin", MessageBoxButton.YesNo);

                                if (qustion == MessageBoxResult.Yes)
                                {
                                    check = true;
                                    MessageBox.Show("Нейросеть не ошибается! :)",
"MyNeuronNetworkWin", MessageBoxButton.OK, MessageBoxImage.Information);
                                }
                                else if (qustion == MessageBoxResult.No)
                                {
                                    check = true;
                                    MessageBox.Show("Без подсказки не справится :)",
"MyNeuronNetworkWin", MessageBoxButton.OK, MessageBoxImage.Information);
                                }
                                Close();
                                break;
                            }
                        }

                        if (check == false)
                        {
                            MessageBox.Show("Данных об этом актере нет", "MyNeuronNetworkWin",
MessageBoxButton.OK, MessageBoxImage.Information);
                        }
                    }
                    else
                    {
                        if (name_TextBox.Text != "")
                        {
                            bool checkA = true;

                            foreach (var name in nnentities.Info)
                            {
                                if (name.IdNeuronNetwork == answer) //сравнивает существующий код в
БД с тем кодом которая считает нейросеть
                                {
                                    var qustion = MessageBox.Show($"Это - {name.Name}. Правильно?",
"MyNeuronNetworkWin", MessageBoxButton.YesNo);

                                    if (qustion == MessageBoxResult.Yes)
                                    {
                                        check = true;
                                        MessageBox.Show("Нейросеть не ошибается! :)",
"MyNeuronNetworkWin", MessageBoxButton.OK, MessageBoxImage.Information);
                                        Close();
                                    }
                                    else if (qustion == MessageBoxResult.No)
                                    {
                                        checkA = true;
                                    }
                                    break;
```

```
                            }
                        }

                        if (check == false) //если данных об актере нет или нейросеть допустила
ошибку, то заносим данные в нейросеть (и базу данных)
                        {
                            int neu = -1;
                            foreach (var name in nnentities.Info)
                            {
                                if (name.Name == name_TextBox.Text)
                                {
                                    neu = int.Parse(name.IdNeuronNetwork.ToString());
                                    checkA = false;
                                    break;
                                }
                                else neu = int.Parse(name.IdNeuronNetwork.ToString());
                            }

                            if (checkA == true)
                            {
                                var add_info = new Info
                                {
                                    Name = name_TextBox.Text,
                                    IdNeuronNetwork = (neu + 1)
                                };

                                nnentities.Info.Add(add_info);
                                nnentities.SaveChanges();

                                NeuronAnswer(name_TextBox.Text, int.Parse((neu + 1).ToString()),
width, height, img, net);
                            }
                            else NeuronAnswer(name_TextBox.Text, int.Parse(neu.ToString()),
width, height, img, net);
                        }
                    }
                    else MessageBox.Show("Поле с именем и фамилией не заполнено",
"MyNeuronNetworkWin", MessageBoxButton.OK, MessageBoxImage.Warning);
                }
            }
            else MessageBox.Show("Поле с ссылкой не заполнено", "MyNeuronNetworkWin",
MessageBoxButton.OK, MessageBoxImage.Warning);
        }
        catch (Exception)
        {
            MessageBox.Show("Перезапустите программу и повторите запрос", "MyNeuronNetworkWin",
MessageBoxButton.OK, MessageBoxImage.Information);
            Close();
        }
    }

    public void NeuronAnswer(string name, int input, int width, int height, System.Drawing.Image
img, ImageNeuronNetwork net)
    {
        try
        {
            img.ToInput(width, height);
            net.Study(img.ToInput(width, height), input, 15);
            MessageBox.Show($"Хорошо, я запомнил что это - {name}", "MyNeuronNetworkWin",
MessageBoxButton.OK, MessageBoxImage.Information);
            net.JSSer();
            Close();
        }
        catch (Exception)
        {
            MessageBox.Show("Ошибка записью в нейросеть! Обратитесь к разработчику",
"MyNeuronNetworkWin", MessageBoxButton.OK, MessageBoxImage.Error);
            Close();
        }
```

```
        }

        public void WebSite(string link)
        {
            using (WebClient webClient = new WebClient())
            {
                webClient.DownloadFile(link, "digit.bmp");
            }
        }

        private void PasteLink_Click(object sender, RoutedEventArgs e)
        {
            if (Clipboard.ContainsText() == true) link_TextBox.Text = Clipboard.GetText();
        }

        private void Name_CheckBox_Checked(object sender, RoutedEventArgs e)
        {
            name_TextBox.IsEnabled = true;
        }

        private void Name_CheckBox_Unchecked(object sender, RoutedEventArgs e)
        {
            name_TextBox.IsEnabled = false; name_TextBox.Text = "";
        }

        private void Actors_ListBox_SelectionChanged(object sender, SelectionChangedEventArgs e)
        {
            if (name_TextBox.IsEnabled == true)
            {
                var selected_actors = actors_ListBox.SelectedItem as Info;

                if (selected_actors != null)
                {
                    name_TextBox.Text = selected_actors.Name;
                }
            }
        }
    }
}
```

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

namespace MyNeuronNetworkWin.Interface
{
    public interface INeuron
    {
        int[,] Weight { get; set; } //память нейрона

        int Handle(int[,] input); //сила нейрона

        void Study(int[,] input, int factor); //обучение нейрона
    }
}
```

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

namespace MyNeuronNetworkWin.Interface
{
    public interface INeuronNetwork
    {
        INeuron[] Neurons { get; set; } //массив интерфейсов

        int getAnswer(int[,] input); //возврат правильного ответа по мнению нейронной сети

        void Study(int[,] input, int correctAnswer, int factor); //обучение конкретного нейрона и
уверенность выбора
    }
}
```

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

namespace MyNeuronNetworkWin.Models
{
    public static class ImageEx
    {
        public static int[,] ToByte(this Image img) //преобразует изображение в двумерный массив в двоичном коде
        {
            int[,] mass = null;

            try
            {
                var bmp = new Bitmap(img);
                mass = new int[bmp.Width, bmp.Height];

                for (int y = 0; y < img.Height; y++)
                {
                    for (int x = 0; x < img.Width; x++)
                    {
                        var IsWhite = bmp.GetPixel(x, y).R >= 230 && bmp.GetPixel(x, y).G >= 230 && bmp.GetPixel(x, y).B >= 230;
                        mass[x, y] = IsWhite ? 0 : 1;
                    }
                }

            }
            catch(IndexOutOfRangeException)
            {
                MessageBox.Show("Было выбрано некорректное изображение. Попробуйте выбрать другое изображение. Перезапустите программу", "MyNeuronNetworkWin", MessageBoxButton.OK, MessageBoxImage.Error);
                MainWindow mw = new MainWindow();
                mw.Close();
            }
            return mass;
        }

        public static Image ScaleImage(this Image source, int width, int height) //увеличение изображение в размерах и "размещение по центру"
        {
            Image dest = new Bitmap(width, height);
            using (Graphics gr = Graphics.FromImage(dest))
            {
                gr.FillRectangle(Brushes.White, 0, 0, width, height); //очищаем экран
                gr.InterpolationMode = System.Drawing.Drawing2D.InterpolationMode.HighQualityBicubic;

                float srcwidth = source.Width;
                float srcheight = source.Height;
                float dstwidth = width;
                float dstheight = height;

                if (srcwidth <= dstwidth && srcheight <= dstheight) //исходное изображение меньше целевого
                {
                    int left = (width - source.Width) / 2;
                    int top = (height - source.Height) / 2;
                    gr.DrawImage(source, left, top, source.Width, source.Height);
                }
            }

            return dest;
```

```
        }

        public static int[,] ToInput(this Image sourse, int width, int height) //превращение
изображения в необходимый вид для нейронной сети
        {
            return sourse.ScaleImage(width, height).ToByte();
        }
    }
}




using MyNeuronNetworkWin.Interface;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace MyNeuronNetworkWin.Model
{
    public class ImageNeuron : INeuron
    {
        public int[,] Weight { get; set; }


        public int Handle(int[,] input) //сила
        {
            var power = 0;

            for (int y = 0; y < Weight.GetLength(1); y++)
            {
                for (int x = 0; x < Weight.GetLength(0); x++)
                {
                    power += Weight[x, y] * input[x, y];
                }
            }

            return power;
        }

        public void Study(int[,] input, int factor) //обучение
        {
            for (int y = 0; y < Weight.GetLength(1); y++)
            {
                for (int x = 0; x < Weight.GetLength(0); x++)
                {
                    Weight[x, y] += factor * input[x, y];
                }
            }
        }
    }
}
```

```
using MyNeuronNetworkWin.Interface;
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace MyNeuronNetworkWin.Model
{
    public class ImageNeuronNetwork : INeuronNetwork
    {
        const string NeuronsPath = "Neurons.json";

        public INeuron[] Neurons { get; set; }

        public ImageNeuronNetwork(int neuronCount, int width, int height)
        {
            if (!File.Exists(NeuronsPath)) //если файла нет, он будет создан и ему будут переданы
введенные параметры
            {
                Neurons = new ImageNeuron[neuronCount];

                for (int i = 0; i < Neurons.Length; i++)
                {
                    Neurons[i] = new ImageNeuron
                    {
                        Weight = new int[width, height]
                    };
                }
            }
            else
            {
                Neurons = JsonHelper.Deserialize<ImageNeuron[]>(NeuronsPath); //если нейрон уже
есть, то данные будут загружаться в одномерный массив
            }
        }

        public void JSSer()
        {
            JsonHelper.Serialize(Neurons, NeuronsPath);
        }

        public int getAnswer(int[,] input) //возврат ответа
        {
            var answers = new int[Neurons.Length];

            for (int i = 0; i < Neurons.Length; i++)
            {
                answers[i] = Neurons[i].Handle(input);
            }

            int maxIndex = 0;

            for (int i = 0; i < answers.Length; i++)
            {
                if (answers[i] > answers[maxIndex])
                {
                    maxIndex = i;
                }
            }

            return maxIndex;
        }

        public void Study(int[,] input, int correctAnswer, int factor) //запоминание к корректному
нейрону
        {
            Neurons[correctAnswer].Study(input, 15);
```

```
        }
    }
}
```

```
using Newtonsoft.Json;
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace MyNeuronNetworkWin.Model
{
    class JsonHelper
    {
        public static void Serialize<T>(T obj, string path) //записываем объект в файл
        {
            File.WriteAllText(path, JsonConvert.SerializeObject(obj));
        }

        public static T Deserialize<T>(string path) //загрузка файла
        {
            return JsonConvert.DeserializeObject<T>(File.ReadAllText(path));
        }
    }
}
```