**1.1 Класс MainWindow.cs**

using System;

using System.Windows.Forms;

using System.Runtime.Serialization;

using System.Runtime.Serialization.Formatters.Binary;

using System.IO;

using System.Collections.Generic;

namespace CSharp\_Analyzer\_of\_Keystroke\_Dynamics

{

public partial class MainWindow : Form

{

private KeyStrokeRecorder \_ksr;

private Account \_userInfo;

private Database \_db;

private List<KeyStrokeRecorder> \_kdrList = new List<KeyStrokeRecorder>();

private string \_username;

private string \_password;

private Account bestMatch = null;

private bool firstKey = true;

public MainWindow()

{

InitializeComponent();

//Check if the file exists

if (File.Exists("accounts.bin"))

{

//Read data from the file

IFormatter formatter = new BinaryFormatter();

Stream fs = new FileStream("accounts.bin", FileMode.Open, FileAccess.Read, FileShare.Read);

try

{

\_db = (Database)formatter.Deserialize(fs);

}

catch (Exception)

{

MessageBox.Show("Критическая ошибка в работе с файлом, вызванная его изменением пользователем.\nФайл с базой пользователей используется программой и не предназначен для открытия и/или изменения самим пользователем/другим ПО.\nДля устранения неполадки и в целях безопасности файл и все данные о пользователях будут удалены, а приложение перезапущено.", "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

fs.Close();

File.Delete("accounts.bin");

Application.Restart();

}

fs.Close();

}

else

{

//Create new database

\_db = new Database();

\_db.Error = 180;

}

error\_trackBar.Value = \_db.Error;

\_dg.Columns.Add("Key", "Клавиша");

\_dg.Columns.Add("PressTime", "Длительность");

\_dg.Columns.Add("Interval", "Интервал");

\_dg.Columns[0].Width = (int)(1.2 \* \_dg.Columns[0].Width);

\_dg.Columns[1].Width = (int)(1.6 \* \_dg.Columns[1].Width);

\_dg.Columns[2].Width = (int)(1.2 \* \_dg.Columns[2].Width);

TrackBarValue\_label.Text = error\_trackBar.Value.ToString();

}

private void Input\_richTextBox\_KeyDown(object sender, KeyEventArgs e)

{

if (e.Modifiers == Keys.Control)

{

MessageBox.Show("Нельзя копировать/вставлять текст", "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

else

{

if (firstKey)

{

firstKey = false;

\_ksr.Start();

}

//Filter keys

if (char.IsLetterOrDigit((char)e.KeyCode))

{

if (\_ksr.IsKeyPressed(e.KeyCode))

{

e.SuppressKeyPress = true;

}

else

{

\_ksr.KeyPressed(e.KeyCode);

}

}

}

}

private void Input\_richTextBox\_KeyUp(object sender, KeyEventArgs e)

{

//Filter keys

if (char.IsLetterOrDigit((char)e.KeyCode))

{

\_ksr.KeyReleased(e.KeyCode);

var ks = \_ksr.KeyStrokes[\_ksr.KeyStrokes.Count - 1];

}

}

private void Back\_button1\_Click(object sender, EventArgs e)

{

SetPassword\_richTextBox.Text = "";

SetUsername\_richTextBox.Text = "";

\_username = "";

\_password = "";

tablessControl.SelectedTab = Welcome\_tabPage;

}

private void Next\_button1\_Click(object sender, EventArgs e)

{

bool wrongUsername = false, wrongPassword = false, reserved = false;

string username = SetUsername\_richTextBox.Text;

string password = SetPassword\_richTextBox.Text;

//Check if username is already reserved

for (int i = 0; i < \_db.Accounts.Count; i++)

{

if (username == \_db.Accounts[i].Name)

{

reserved = true;

}

}

if (username == "")

{

wrongUsername = true;

}

if (password.Length<12)

{

wrongPassword = true;

}

foreach (var letter in password)

{

if (letter == ' ')

{

wrongPassword = true;

}

}

foreach (var letter in username)

{

if (!char.IsLetterOrDigit(letter) && letter !=' ')

{

wrongUsername = true;

}

}

if (!wrongUsername&&!wrongPassword&&!reserved)

{

\_password = password;

\_username = username;

SetPassword\_richTextBox.Text = "";

SetUsername\_richTextBox.Text = "";

\_ksr = new KeyStrokeRecorder();

firstKey = true;

tablessControl.SelectedTab = DynamicsRecord\_tabPage;

ActiveControl = Input\_richTextBox;

}

else if (wrongPassword && wrongUsername)

{

MessageBox.Show("Недопустимые пароль и имя пользователя", "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

SetPassword\_richTextBox.Text = "";

}

else if (wrongUsername)

{

MessageBox.Show("Недопустимое имя пользователя", "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

else if (wrongPassword)

{

MessageBox.Show("Пароль должен состоять из 12 или более символов и не содержать пробелов", "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

SetPassword\_richTextBox.Text = "";

}

else if (reserved)

{

MessageBox.Show("Пользователь с таким именем уже существует.", "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void Next\_button2\_Click(object sender, EventArgs e)

{

\_ksr.Stop();

if (Input\_richTextBox.Text==\_password)

{

//Add a record about the user and his keystroke dynamics to the list

\_kdrList.Add(\_ksr);

\_ksr = new KeyStrokeRecorder();

firstKey = true;

tablessControl.SelectedTab = KDR2\_tabPage;

ActiveControl = KDR2\_richTextBox;

}

else

{

MessageBox.Show("Пароли должны совпадать!", "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

//Clear the recieved data

\_ksr = new KeyStrokeRecorder();

firstKey = true;

ActiveControl = Input\_richTextBox;

}

Input\_richTextBox.Text = "";

}

private void Back\_button2\_Click(object sender, EventArgs e)

{

\_password = "";

\_username = "";

Input\_richTextBox.Text = "";

tablessControl.SelectedTab = setPassUsername\_tabPage;

ActiveControl = SetUsername\_richTextBox;

}

private void Login\_button\_Click(object sender, EventArgs e)

{

if (\_db.Accounts.Count>0)

{

tablessControl.SelectedTab = LogIn\_tabPage;

\_ksr = new KeyStrokeRecorder();

firstKey = true;

ActiveControl = LoginPassword\_richTextBox;

}

else

{

MessageBox.Show("Нет зарегистрированных пользователей!\nПожалуйста, создайте новую учетную запись!", "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void newUser\_button\_Click(object sender, EventArgs e)

{

tablessControl.SelectedTab = setPassUsername\_tabPage;

\_kdrList = new List<KeyStrokeRecorder>();

ActiveControl = SetUsername\_richTextBox;

}

private void Back\_button3\_Click(object sender, EventArgs e)

{

LoginPassword\_richTextBox.Text = "";

tablessControl.SelectedTab = Welcome\_tabPage;

}

private void MainWindow\_FormClosing(object sender, FormClosingEventArgs e)

{

IFormatter formatter = new BinaryFormatter();

Stream stream = new FileStream("accounts.bin", FileMode.Create, FileAccess.Write, FileShare.ReadWrite);

formatter.Serialize(stream, \_db);

stream.Close();

}

private void Next\_button3\_Click(object sender, EventArgs e)

{

\_dg.Rows.Clear();

List<Account> matchedAccounts = new List<Account>();

for (int i = 0; i < \_db.Accounts.Count; i++)

{

if (\_db.Accounts[i].Password == LoginPassword\_richTextBox.Text)

{

matchedAccounts.Add(\_db.Accounts[i]);

}

}

if (matchedAccounts.Count > 0)

{

Account tryLogIn = new Account("TryLogIn", LoginPassword\_richTextBox.Text, \_ksr.KeyStrokes);

double min = double.MaxValue;

for (int i = 0; i < matchedAccounts.Count; i++)

{

if (min > matchedAccounts[i].CompareTo(tryLogIn))

{

min = matchedAccounts[i].CompareTo(tryLogIn);

bestMatch = matchedAccounts[i];

}

}

if (min < \_db.Error) //Right?

{

LoginPassword\_richTextBox.Text = "";

\_dg.Rows.Add(bestMatch.Key(0), bestMatch.PressTime(0).ToString() + " мс", "-");

for (int i = 1; i < \_ksr.KeyStrokes.Count; i++)

{

\_dg.Rows.Add(bestMatch.Key(i), bestMatch.PressTime(i).ToString() + " мс", Math.Abs(bestMatch.Interval(i)).ToString() + " мс");

}

double APT = 0;

double AI = 0;

for (int i = 0; i < bestMatch.Password.Length; i++)

{

APT += bestMatch.PressTime(i);

}

for (int i = 1; i < bestMatch.Password.Length; i++)

{

AI += bestMatch.Interval(i);

}

AveragePresstime\_label.Text = (APT / bestMatch.Password.Length).ToString("F4") + " мс";

AverageInterval\_label.Text = (AI / (bestMatch.Password.Length - 1)).ToString("F4") + " мс";

InfoUsername\_label.Text = bestMatch.Name;

\_ksr.Stop();

tablessControl.SelectedTab = UserDetails\_tabPage;

}

else

{

MessageBox.Show("Доступ запрещен!", "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Stop);

LoginPassword\_richTextBox.Text = "";

tablessControl.SelectedTab = Welcome\_tabPage;

}

}

else

{

MessageBox.Show("Неверный пароль!\nПопробуйте еще раз.", "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

LoginPassword\_richTextBox.Text = "";

\_ksr = new KeyStrokeRecorder();

firstKey = true;

}

}

private void Logout\_button\_Click(object sender, EventArgs e)

{

tablessControl.SelectedTab = Welcome\_tabPage;

}

private void LoginPassword\_richTextBox\_KeyPress(object sender, KeyPressEventArgs e)

{

if (!char.IsLetterOrDigit(e.KeyChar))

{

MessageBox.Show("Вы сбились!\nПопробуйте еще раз.", "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

LoginPassword\_richTextBox.Text = "";

\_ksr = new KeyStrokeRecorder();

firstKey = true;

}

}

private void Delete\_button\_Click(object sender, EventArgs e)

{

tablessControl.SelectedTab = Welcome\_tabPage;

MessageBox.Show("Пользователь " + \_db.Accounts.Find(x => x.Name == InfoUsername\_label.Text).Name + " успешно удален", "Удаление пользователя", MessageBoxButtons.OK, MessageBoxIcon.Information);

\_db.Accounts.Remove(\_db.Accounts.Find(x => x.Name == InfoUsername\_label.Text));

}

private void Input\_richTextBox\_KeyPress(object sender, KeyPressEventArgs e)

{

if (!char.IsLetterOrDigit(e.KeyChar))

{

MessageBox.Show("Вы сбились!\nПопробуйте еще раз.", "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

Input\_richTextBox.Text = "";

\_ksr = new KeyStrokeRecorder();

firstKey = true;

}

}

private void NextKDR3\_button\_Click(object sender, EventArgs e)

{

\_ksr.Stop();

if (KDR3\_richTextBox.Text == \_password)

{

\_kdrList.Add(\_ksr);

List<KeyStroke> averageKDR = new List<KeyStroke>();

double pressed, released;

for (int i = 0; i < \_kdrList[0].KeyStrokes.Count; i++)

{

averageKDR.Add(new KeyStroke());

averageKDR[i].Key = \_kdrList[0].KeyStrokes[i].Key;

pressed = \_kdrList[0].KeyStrokes[i].Pressed.TotalMilliseconds + \_kdrList[1].KeyStrokes[i].Pressed.TotalMilliseconds + \_kdrList[2].KeyStrokes[i].Pressed.TotalMilliseconds;

released = \_kdrList[0].KeyStrokes[i].Released.TotalMilliseconds + \_kdrList[1].KeyStrokes[i].Released.TotalMilliseconds + \_kdrList[2].KeyStrokes[i].Released.TotalMilliseconds;

averageKDR[i].Pressed = TimeSpan.FromMilliseconds(pressed / 3);

averageKDR[i].Released = TimeSpan.FromMilliseconds(released / 3);

}

//Make a record about the user and his keystroke dynamics

\_userInfo = new Account(\_username, \_password, averageKDR);

\_db.Accounts.Add(\_userInfo);

MessageBox.Show("Новый пользователь успешно добавлен!", "Новый пользователь", MessageBoxButtons.OK, MessageBoxIcon.Information);

tablessControl.SelectedTab = Welcome\_tabPage;

}

else

{

MessageBox.Show("Пароли должны совпадать!", "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

//Clear the recieved data

\_ksr = new KeyStrokeRecorder();

firstKey = true;

ActiveControl = KDR3\_richTextBox;

}

KDR3\_richTextBox.Text = "";

}

private void KDR2\_richTextBox\_KeyPress(object sender, KeyPressEventArgs e)

{

if (!char.IsLetterOrDigit(e.KeyChar))

{

MessageBox.Show("Вы сбились!\nПопробуйте еще раз.", "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

KDR2\_richTextBox.Text = "";

\_ksr = new KeyStrokeRecorder();

firstKey = true;

}

}

private void NextKDR2\_button\_Click(object sender, EventArgs e)

{

\_ksr.Stop();

if (KDR2\_richTextBox.Text == \_password)

{

//Add a record about the user and his keystroke dynamics to the list

\_kdrList.Add(\_ksr);

\_ksr = new KeyStrokeRecorder();

firstKey = true;

tablessControl.SelectedTab = KDR3\_tabPage;

ActiveControl = KDR3\_richTextBox;

}

else

{

MessageBox.Show("Пароли должны совпадать!", "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

//Clear the recieved data

\_ksr = new KeyStrokeRecorder();

firstKey = true;

ActiveControl = KDR2\_richTextBox;

}

KDR2\_richTextBox.Text = "";

}

private void KDR3\_richTextBox\_KeyPress(object sender, KeyPressEventArgs e)

{

if (!char.IsLetterOrDigit(e.KeyChar))

{

MessageBox.Show("Вы сбились!\nПопробуйте еще раз.", "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

KDR3\_richTextBox.Text = "";

\_ksr = new KeyStrokeRecorder();

firstKey = true;

}

}

private void оПрограммеToolStripMenuItem1\_Click(object sender, EventArgs e)

{

MessageBox.Show("С# анализатор клавиатурного почерка\nВыполнил: Мариносян Никита Арамович\nстудент 1 курса\nНИУ ВШЭ\nnikita.marinosyan@gmail.com", "О программе", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

private void оПрограммеToolStripMenuItem\_Click(object sender, EventArgs e)

{

MessageBox.Show("Клавиатурный почерк – поведенческая биометрическая характеристика человека, обуславливающаяся особенностями его работы с клавиатурой, такими как время удержания нажатой клавиши, время паузы между нажатием клавиш и частота нажатия клавиш клавиатуры.", "Клавиатурный почерк", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

private void созданиеНовогоПользователяToolStripMenuItem\_Click(object sender, EventArgs e)

{

MessageBox.Show("Создание нового пользователя происходит в несколько этапов:\n 1.Выбор имени пользователя и пароля\n2. Запоминание клавиатурного почерка путем ввода пароля несколько раз", "Создание нового пользователя", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

private void входВСистемуToolStripMenuItem\_Click(object sender, EventArgs e)

{

MessageBox.Show("Для входа в систему требуется ввести пароль.\nЕго требуется вводить в том же темпе,\nчто и при создании учетной записи.", "Вход в систему", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

private void Settings\_button\_Click(object sender, EventArgs e)

{

tablessControl.SelectedTab = Preferences\_tabPage;

}

private void PreferencesOK\_button\_Click(object sender, EventArgs e)

{

\_db.Error = error\_trackBar.Value;

tablessControl.SelectedTab = UserDetails\_tabPage;

}

private void error\_trackBar\_Scroll(object sender, EventArgs e)

{

TrackBarValue\_label.Text = error\_trackBar.Value.ToString();

}

}

}

**1.2 Класс KeyStroke.cs**

using System;

using System.Windows.Forms;

namespace CSharp\_Analyzer\_of\_Keystroke\_Dynamics

{

/// <summary>

/// Performs the information about pressing and releasing time of the key

/// </summary>

[Serializable]

class KeyStroke

{

public Keys Key;

public TimeSpan Pressed; //Time when the key was pressed

public TimeSpan Released; //Time when the key was released

public TimeSpan PressTime //Duration of pressing the key

{

get { return Released - Pressed; }

}

}

}

**1.3 Класс KeyStrokeRecorder.cs**

using System.Collections.Generic;

using System.Diagnostics;

using System.Windows.Forms;

namespace CSharp\_Analyzer\_of\_Keystroke\_Dynamics

{

/// <summary>

/// Records keys being pressed and released

/// </summary>

class KeyStrokeRecorder

{

/// <summary>

/// Tracks the time

/// </summary>

private readonly Stopwatch \_sw = new Stopwatch();

/// <summary>

/// Contains user pressed keys and corresponding press (release) time

/// </summary>

private readonly Dictionary<Keys, KeyStroke> \_dict = new Dictionary<Keys, KeyStroke>();

public List<KeyStroke> KeyStrokes { get; set; }

public KeyStrokeRecorder()

{

KeyStrokes = new List<KeyStroke>();

}

/// <summary>

/// Returns true if the key hase been already pressed. Returns false if the key has not been pressed yet.

/// </summary>

/// <param name="k">Key</param>

/// <returns></returns>

public bool IsKeyPressed(Keys k)

{

return \_dict.ContainsKey(k);

}

/// <summary>

/// Adds pressed key in dictionary of the pressed keys

/// </summary>

/// <param name="k">Key</param>

public void KeyPressed(Keys k)

{

\_dict.Add(k, new KeyStroke { Pressed = \_sw.Elapsed, Key = k });

}

/// <summary>

/// Removes released key from the dictionary of the pressed keys

/// </summary>

/// <param name="k"></param>

public void KeyReleased(Keys k)

{

var ks = \_dict[k];

\_dict.Remove(k);

ks.Released = \_sw.Elapsed;

KeyStrokes.Add(ks);

}

/// <summary>

/// Starts the stopwatch

/// </summary>

public void Start()

{

\_sw.Restart();

}

/// <summary>

/// Stops the stopwatch

/// </summary>

public void Stop()

{

\_sw.Stop();

}

}

}

**1.4 Класс Account.cs**

using System;

using System.Collections.Generic;

using System.Windows.Forms;

namespace CSharp\_Analyzer\_of\_Keystroke\_Dynamics

{

[Serializable]

class Account

{

private string \_username;

private string \_password;

private List<KeyStroke> \_keystrokeDynamics;

public string Name

{

get

{

return \_username;

}

}

public string Password

{

get

{

return \_password;

}

}

public Account(string Username, string Password, List<KeyStroke> KeystrokeDynamics)

{

\_keystrokeDynamics = KeystrokeDynamics;

\_username = Username;

\_password = Password;

}

public double CompareTo(Account ac)

{

double epsilon = 0;

for (int i = 0; i < ac.\_keystrokeDynamics.Count; i++)

{

epsilon += Math.Pow((ac.\_keystrokeDynamics[i].PressTime.TotalMilliseconds - \_keystrokeDynamics[i].PressTime.TotalMilliseconds),2);

}

for (int i = 1; i < ac.\_keystrokeDynamics.Count; i++)

{

epsilon += Math.Pow((ac.Interval(i) - Interval(i)),2);

}

return Math.Sqrt(epsilon);

}

public double Interval(int index)

{

return Math.Abs(\_keystrokeDynamics[index - 1].Released.TotalMilliseconds - \_keystrokeDynamics[index].Pressed.TotalMilliseconds);

}

public double PressTime(int index)

{

return \_keystrokeDynamics[index].PressTime.TotalMilliseconds;

}

public Keys Key(int index)

{

return \_keystrokeDynamics[index].Key;

}

}

}

**1.5 Класс Database.cs**

using System;

using System.Collections.Generic;

namespace CSharp\_Analyzer\_of\_Keystroke\_Dynamics

{

[Serializable]

class Database

{

public List<Account> Accounts = new List<Account>();

public int Error;

}

}

**1.6 Класс TablessControl.cs**

using System;

using System.Windows.Forms;

namespace CSharp\_Analyzer\_of\_Keystroke\_Dynamics

{

class TablessControl : TabControl

{

protected override void WndProc(ref Message m)

{

// Hide tabs by trapping the TCM\_ADJUSTRECT message

if (m.Msg == 0x1328 && !DesignMode) m.Result = (IntPtr)1;

else base.WndProc(ref m);

}

}

}