МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение высшего образования

**«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ СИСТЕМ И ТЕХНОЛОГИЙ

Кафедра «Информационные системы»

Дисциплина «Информационная безопасность»

Лабораторная работа № 1

Выполнил:

студент гр. ПИбд -41

Калядин М. С.

Проверил:

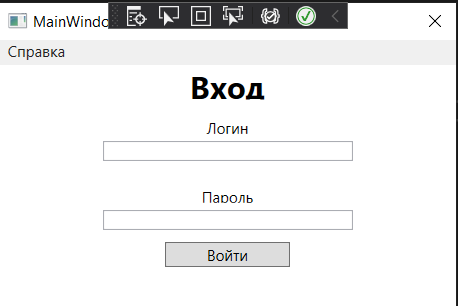
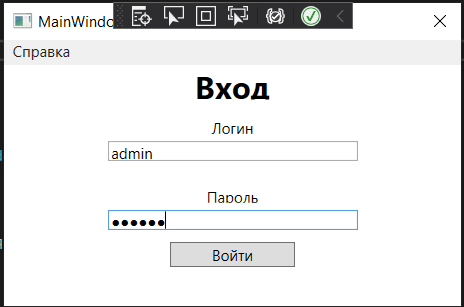
Мытарев П. В.

Ульяновск, 2021г.

Программа написана на языке C# на системе для построения клиентских приложений WPF.

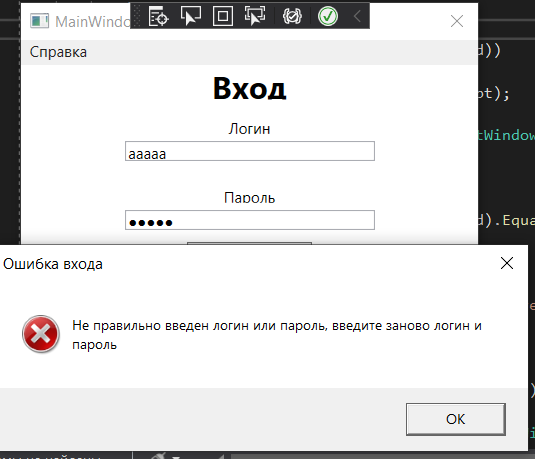
Ссылка на репозиторий - <https://github.com/MaximKalyadin/IB>

Скриншоты экранов

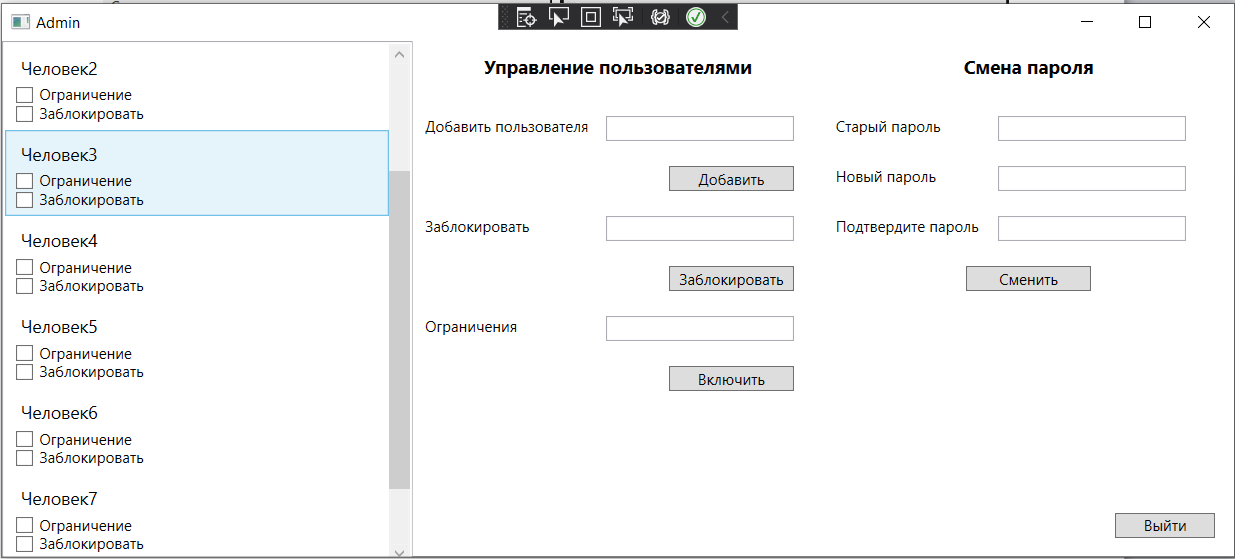
 

Экран входа, после того как пользователь введет сюда логин и пароль, то попадет на экран клиента либо на экран админки.

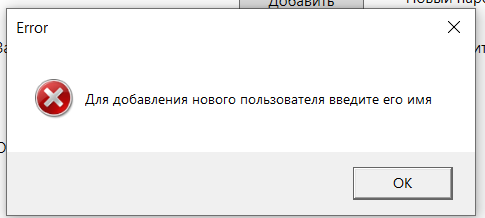
При неправильном наборе логина или пароля появиться сообщение:



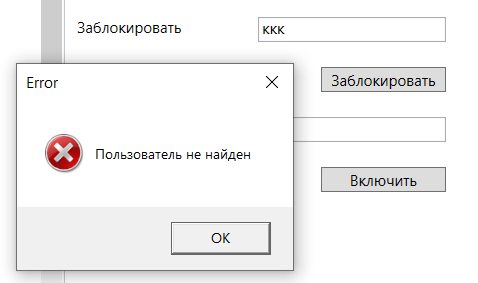
После трех неправильных попыток авторизации программа завершается.



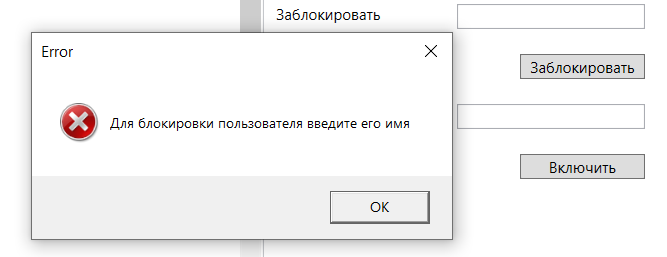
После правильного ввода пароля и логина попадаем на окно админки: мы видим слева список всех пользователей добавленных, посередине управление пользователями, то есть блокировка и ограничение на пользоватеей и их добавление. Справа смена пароля у администратора.



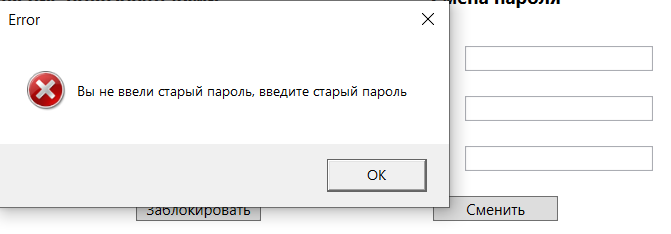
Сообщение при нажатие на кнопку добавить пользователя не введя имя пользователя.



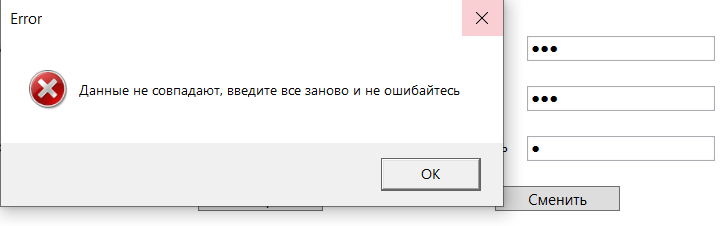
При нажатие на кнопку либо заблокировать либо включить ограничение, когда ввели не правильное имя пользователя.



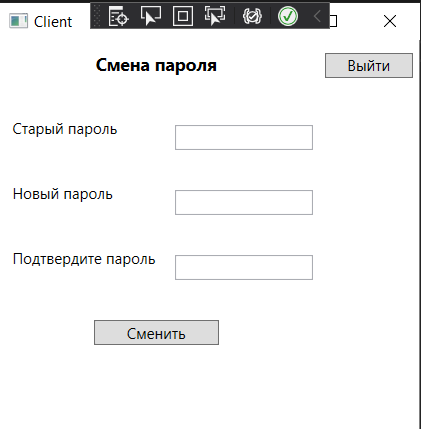
При нажатие на кнопку заблокировать либо включить ограничение не введя имя пользователя.



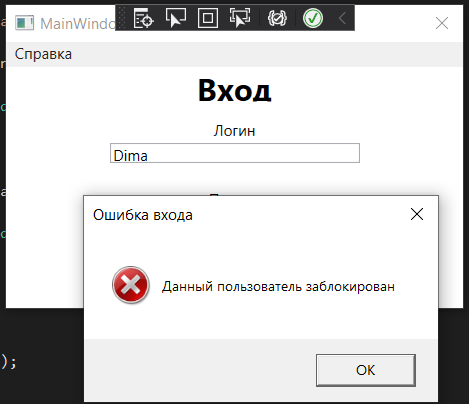
Сообщение при не вводе старого пароля.



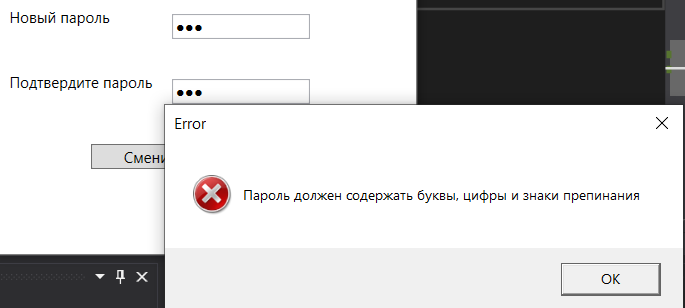
При не совпадении либо старого пароля, либо подтверждения пароля.



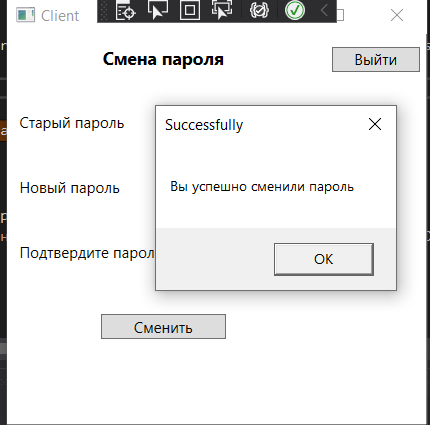
Окно пользователя, здесь показано лишь смена пароля и кнопка выйти. Такие же ошибки будут сыпаться как и на админке.



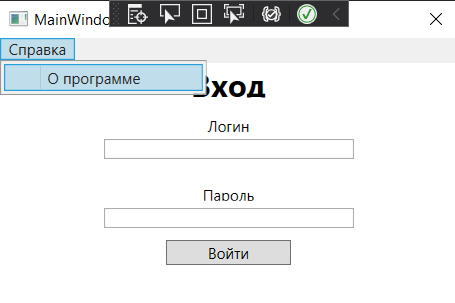
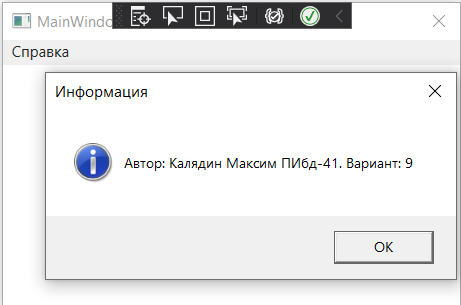
Сообщение при болкировки пользователя



При наложении ограничений на пользователя, а пользователь не правильно ввел новы пароль.

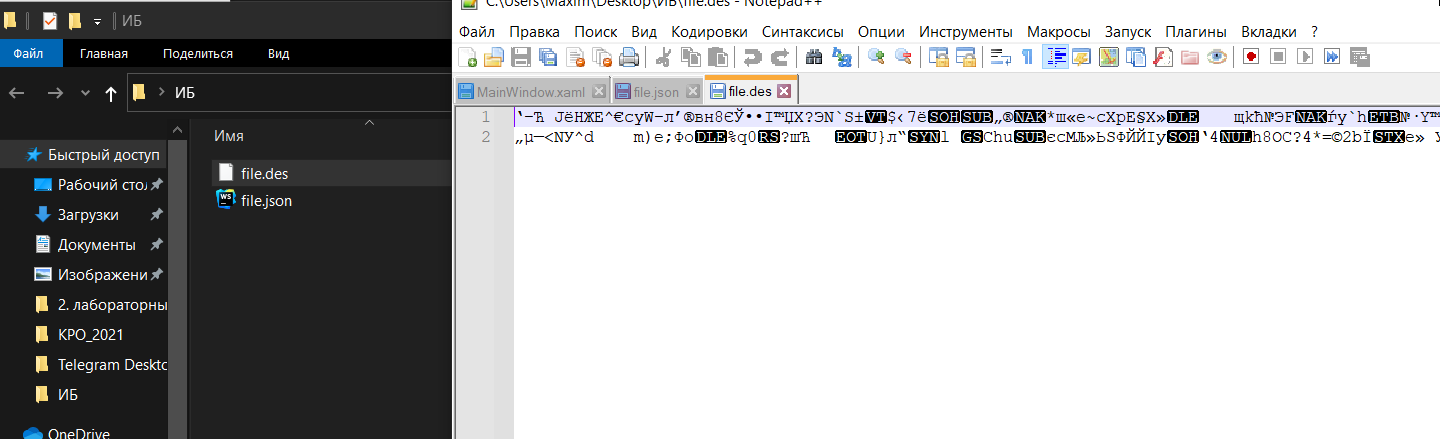


Успешная смена пароля.

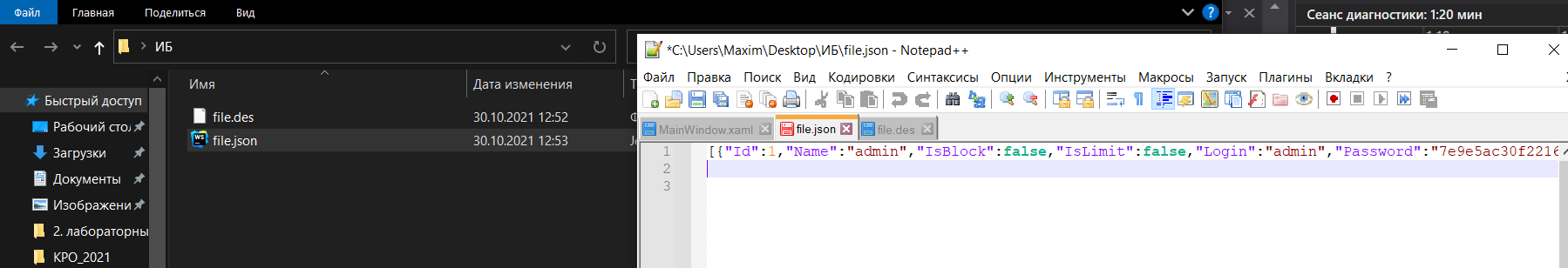
 

Информация об авторе.

Описание файлов



Представлен зашифрованный файл механизмом DES. Расположение его в отдельной папке.



Временный файл, где храняться все о пользователях и администраторе. Формат файла – JSON, так как с ним удобнее работать. Также храниться в отдельной папке. Пароли захешированы методом sha.

При первом открытие программы создается json файл. После входа этот json сразу шифруется и сохраниется в отдельной папке, после входа на админку или коиента файл расшифровывается во временный файл и помещается туда же где и лежит шифрованный файл. После закрытия временный файл удаляется а все данные шифруюся обратно в файл des.

Листинг кода.

Листинг окна входа

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

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 IB1.Models;

using IB1.Service;

using Newtonsoft.Json;

namespace IB1

{

/// <summary>

/// Логика взаимодействия для MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

private int IsChecked = 0;

private List<Client> clients = new List<Client>();

readonly string path = @"C:\Users\Maxim\Desktop\ИБ\file.json";

private readonly string pathEncrypt = @"C:\Users\Maxim\Desktop\ИБ\file.des";

public MainWindow()

{

InitializeComponent();

CreateOrUpdateFile();

}

/// <summary>

/// Шифрование и расшифрование файла. Создание временного файла после расшифрования.

/// </summary>

public void CreateOrUpdateFile()

{

if (File.Exists(pathEncrypt))

{

DES.DecryptFile(pathEncrypt, path);

string json = File.ReadAllText(path);

clients = JsonConvert.DeserializeObject<List<Client>>(json);

}

else

{

Client client = new Client()

{

Name = "admin",

Id = 1,

IsAdmin = true,

IsBlock = false,

IsLimit = false,

Login = "admin",

Password = ""

};

clients.Add(client);

string json = JsonConvert.SerializeObject(clients);

using (StreamWriter tw = new StreamWriter(path, true))

{

tw.WriteLine(json.ToString());

tw.Close();

}

}

}

/// <summary>

/// Кнопка для входа в программу

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void Button\_Click(object sender, RoutedEventArgs e)

{

if (login.Text.ToLower().Equals("admin"))

{

if (string.IsNullOrEmpty(clients[0].Password))

{

AdminWindow admin = new AdminWindow();

admin.Show();

Close();

}

else if (DES.ToSHA256(passw.Password).Equals(clients[0].Password))

{

AdminWindow admin = new AdminWindow();

admin.Show();

Close();

}

else

{

IncorrectLoginOrPassword();

}

}

else

{

bool temp = false;

foreach (var el in clients)

{

if (el.Name.Equals(login.Text))

{

temp = true;

if (string.IsNullOrEmpty(el.Password))

{

if (el.IsBlock)

{

MessageBox.Show("Данный пользователь заблокирован", "Ошибка входа", MessageBoxButton.OK, MessageBoxImage.Error);

}

else

{

DES.EncryptFile(path, pathEncrypt);

File.Delete(path);

ClientWindow client = new ClientWindow(el);

client.Show();

Close();

}

}

else if (DES.ToSHA256(passw.Password).Equals(el.Password))

{

if (el.IsBlock)

{

MessageBox.Show("Данный пользователь заблокирован", "Ошибка входа", MessageBoxButton.OK, MessageBoxImage.Error);

}

else

{

DES.EncryptFile(path, pathEncrypt);

File.Delete(path);

ClientWindow client = new ClientWindow(el);

client.Show();

Close();

}

}

else

{

IncorrectLoginOrPassword();

}

}

}

if (!temp)

{

IncorrectLoginOrPassword();

}

}

}

/// <summary>

/// Функция для показа своего рода ошибок

/// </summary>

public void IncorrectLoginOrPassword()

{

IsChecked++;

if (IsChecked == 3)

{

Close();

}

else

{

MessageBox.Show("Не правильно введен логин или пароль, введите заново логин и пароль", "Ошибка входа",

MessageBoxButton.OK, MessageBoxImage.Error);

login.Text = "";

passw.Password = "";

}

}

/// <summary>

/// Событие нажатие на вкладку справка->об авторе и вывод информации на экран

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void MenuItem\_Click(object sender, RoutedEventArgs e)

{

MessageBox.Show("Автоp: Калядин Максим ПИбд-41. Вариант: 9", "Информация", MessageBoxButton.OK , MessageBoxImage.Information);

}

/// <summary>

/// Событие обработки события закрытие окна, чтобы все дейсвия во временном файле сохранить, удалить файл и данные перенести в зашифрованный

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void Window\_Closed(object sender, EventArgs e)

{

DES.EncryptFile(path, pathEncrypt);

File.Delete(path);

}

}

}

Листинг окна клиента

using IB1.Models;

using IB1.Service;

using Newtonsoft.Json;

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Text.RegularExpressions;

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.Shapes;

namespace IB1

{

/// <summary>

/// Логика взаимодействия для ClientWindow.xaml

/// </summary>

public partial class ClientWindow : Window

{

private List<Client> clients = new List<Client>();

Client client = new Client();

/// <summary>

/// ограничение на выбор пользоателем пароля(регулярные выражения)

/// </summary>

string pattern = @"([a-zA-Z0-9а-яА-Я]\*[,;.-:!?]\*)+";

private readonly string path = @"C:\Users\Maxim\Desktop\ИБ\file.json";

private readonly string pathEncrypt = @"C:\Users\Maxim\Desktop\ИБ\file.des";

public ClientWindow(Client client)

{

InitializeComponent();

DeserializedJson();

this.client = client;

}

/// <summary>

/// Десериализация json (временного файла)

/// </summary>

private void DeserializedJson()

{

if (File.Exists(pathEncrypt))

{

DES.DecryptFile(pathEncrypt, path);

string json = File.ReadAllText(path);

clients = JsonConvert.DeserializeObject<List<Client>>(json);

}

}

/// <summary>

/// Событие кнопки смены пароля

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void Button\_Click(object sender, RoutedEventArgs e)

{

foreach (var el in clients)

{

if (el.Name.Equals(client.Name))

{

if (string.IsNullOrEmpty(old\_passw.Password))

{

if (!el.Password.Equals(old\_passw.Password))

{

Incorrect("Вы не ввели старый пароль, введите старый пароль");

}

}

if (string.IsNullOrEmpty(new\_passw.Password))

{

Incorrect("Для смены пароля введите новый пароль");

}

else if (string.IsNullOrEmpty(conf\_passw.Password))

{

Incorrect("Введите подтврждение пароля");

}

else if (!string.IsNullOrEmpty(new\_passw.Password) && new\_passw.Password.Equals(conf\_passw.Password))

{

if (el.IsLimit)

{

if (Regex.IsMatch(new\_passw.Password, pattern, RegexOptions.IgnoreCase))

{

if (el.Password.Equals(""))

{

el.Password = DES.ToSHA256(new\_passw.Password);

MessageBox.Show("Вы успешно сменили пароль", "Successfully", MessageBoxButton.OK, MessageBoxImage.None);

old\_passw.Password = "";

new\_passw.Password = "";

conf\_passw.Password = "";

}

else if (DES.ToSHA256(old\_passw.Password).Equals(el.Password))

{

el.Password = DES.ToSHA256(new\_passw.Password);

MessageBox.Show("Вы успешно сменили пароль", "Successfully", MessageBoxButton.OK, MessageBoxImage.None);

old\_passw.Password = "";

new\_passw.Password = "";

conf\_passw.Password = "";

}

else

{

Incorrect("Данные не совпадают, введите все заново и не ошибайтесь");

}

}

else

{

MessageBox.Show("Пароль должен содержать буквы, цифры и знаки препинания", "Error", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

else

{

if (el.Password.Equals(""))

{

el.Password = DES.ToSHA256(new\_passw.Password);

MessageBox.Show("Вы успешно сменили пароль", "Successfully", MessageBoxButton.OK, MessageBoxImage.None);

old\_passw.Password = "";

new\_passw.Password = "";

conf\_passw.Password = "";

}

else if (DES.ToSHA256(old\_passw.Password).Equals(el.Password))

{

el.Password = DES.ToSHA256(new\_passw.Password);

MessageBox.Show("Вы успешно сменили пароль", "Successfully", MessageBoxButton.OK, MessageBoxImage.None);

old\_passw.Password = "";

new\_passw.Password = "";

conf\_passw.Password = "";

}

else

{

Incorrect("Данные не совпадают, введите все заново и не ошибайтесь");

}

}

}

else

{

Incorrect("Данные не совпадают, введите все заново и не ошибайтесь");

}

}

}

}

/// <summary>

/// метод для вывода различного вывода сообщений об ошибках

/// </summary>

/// <param name="text">Сообщение ошибки</param>

private void Incorrect(string text)

{

MessageBox.Show(text, "Error", MessageBoxButton.OK, MessageBoxImage.Error);

old\_passw.Password = "";

new\_passw.Password = "";

conf\_passw.Password = "";

}

/// <summary>

/// событие выхода из окна клиента (нажатие на кнопки выйти)

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void Button\_Click\_1(object sender, RoutedEventArgs e)

{

MainWindow mainWindow = new MainWindow();

mainWindow.Show();

Close();

}

/// <summary>

/// событие закрытие окна клиента, чтобы все дейсвия во временном файле сохранить, удалить файл и данные перенести в зашифрованный

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void Window\_Closed(object sender, EventArgs e)

{

string json = JsonConvert.SerializeObject(clients);

File.Delete(path);

using (StreamWriter tw = new StreamWriter(path, true))

{

tw.WriteLine(json.ToString());

tw.Close();

}

DES.EncryptFile(path, pathEncrypt);

File.Delete(path);

}

}

}

Листинг окна администратора

using IB1.Models;

using System;

using System.Collections.Generic;

using System.Linq;

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.Shapes;

using Newtonsoft.Json;

using System.IO;

using IB1.Service;

namespace IB1

{

/// <summary>

/// Логика взаимодействия для AdminWindow.xaml

/// </summary>

public partial class AdminWindow : Window

{

private List<Client> clients = new List<Client>();

private readonly string path = @"C:\Users\Maxim\Desktop\ИБ\file.json";

private readonly string pathEncrypt = @"C:\Users\Maxim\Desktop\ИБ\file.des";

public AdminWindow()

{

InitializeComponent();

DeserializedJson();

}

/// <summary>

/// Десериализация json (временного файла)

/// </summary>

private void DeserializedJson()

{

if (File.Exists(pathEncrypt))

{

DES.DecryptFile(pathEncrypt, path);

string json = File.ReadAllText(path);

clients = JsonConvert.DeserializeObject<List<Client>>(json);

}

LoadData();

}

/// <summary>

/// метод отображение списка пользователей слева

/// </summary>

private void LoadData()

{

listbox.ItemsSource = null;

listbox.ItemsSource = clients;

}

/// <summary>

/// событие добавление нового пользователя

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void Button\_Click(object sender, RoutedEventArgs e)

{

if (string.IsNullOrEmpty(add\_client.Text))

{

Incorrect("Для добавления нового пользователя введите его имя");

}

else

{

clients.Add(new Client()

{

Id = clients.Count,

Name = add\_client.Text,

Login = add\_client.Text,

Password = "",

IsAdmin = false,

IsBlock = false,

IsLimit = false

});

add\_client.Text = "";

LoadData();

}

}

/// <summary>

/// событие блокировки пользователя

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void Button\_Click\_1(object sender, RoutedEventArgs e)

{

if (string.IsNullOrEmpty(block\_client.Text))

{

Incorrect("Для блокировки пользователя введите его имя");

}

else

{

bool temp = false;

foreach (var el in clients)

{

if (el.Name.Equals(block\_client.Text))

{

el.IsBlock = true;

temp = true;

block\_client.Text = "";

LoadData();

}

}

if (!temp)

{

Incorrect("Пользователь не найден");

}

}

}

/// <summary>

/// событие наложение ограничений пользователя

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void Button\_Click\_2(object sender, RoutedEventArgs e)

{

if (string.IsNullOrEmpty(add\_restrict.Text))

{

Incorrect("Для ввода ограничений пользователя введите его имя");

}

else

{

bool temp = false;

foreach (var el in clients)

{

if (el.Name.Equals(add\_restrict.Text))

{

el.IsLimit = true;

temp = true;

add\_restrict.Text = "";

LoadData();

}

}

if (!temp)

{

Incorrect("Пользователь не найден");

}

}

}

/// <summary>

/// событие смены пароля админа

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void Button\_Click\_3(object sender, RoutedEventArgs e)

{

if (string.IsNullOrEmpty(old\_passw.Password))

{

if (!clients[0].Password.Equals(old\_passw.Password))

{

Incorrect("Вы не ввели старый пароль, введите старый пароль");

}

}

if (string.IsNullOrEmpty(new\_passw.Password))

{

Incorrect("Для смены пароля введите новый пароль");

}

else if(string.IsNullOrEmpty(confirm\_passw.Password))

{

Incorrect("Введите подтврждение пароля");

}

else if(!string.IsNullOrEmpty(new\_passw.Password) && new\_passw.Password.Equals(confirm\_passw.Password))

{

if (clients[0].Password.Equals(""))

{

clients[0].Password = DES.ToSHA256(new\_passw.Password);

MessageBox.Show("Вы успешно сменили пароль", "Successfully", MessageBoxButton.OK, MessageBoxImage.None);

old\_passw.Password = "";

new\_passw.Password = "";

confirm\_passw.Password = "";

} else if (DES.ToSHA256(old\_passw.Password).Equals(clients[0].Password))

{

clients[0].Password = DES.ToSHA256(new\_passw.Password);

MessageBox.Show("Вы успешно сменили пароль", "Successfully", MessageBoxButton.OK, MessageBoxImage.None);

old\_passw.Password = "";

new\_passw.Password = "";

confirm\_passw.Password = "";

} else

{

Incorrect("Данные не совпадают, введите все заново и не ошибайтесь");

}

}

else

{

Incorrect("Данные не совпадают, введите все заново и не ошибайтесь");

}

}

/// <summary>

/// метод для вывода различного вывода сообщений об ошибках

/// </summary>

/// <param name="text">Сообщение ошибки</param>

private void Incorrect(string text)

{

MessageBox.Show(text, "Error", MessageBoxButton.OK, MessageBoxImage.Error);

old\_passw.Password = "";

new\_passw.Password = "";

confirm\_passw.Password = "";

}

/// <summary>

/// событие выхода из окна клиента (нажатие на кнопки выйти)

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void Button\_Click\_4(object sender, RoutedEventArgs e)

{

MainWindow mainWindow = new MainWindow();

mainWindow.Show();

Close();

}

/// <summary>

/// событие закрытие окна админа, чтобы все дейсвия во временном файле сохранить, удалить файл и данные перенести в зашифрованный

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void Window\_Closed(object sender, EventArgs e)

{

string json = JsonConvert.SerializeObject(clients);

using (StreamWriter tw = new StreamWriter(path, true))

{

tw.WriteLine(json.ToString());

tw.Close();

}

DES.EncryptFile(path, pathEncrypt);

File.Delete(path);

}

}

}

Листинг модели клиента/админа

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace IB1.Models

{

/// <summary>

/// Модель клиента

/// </summary>

public class Client

{

/// <summary>

/// Уникальный идентификационный номер

/// </summary>

public int Id { get; set; }

/// <summary>

/// Имя пользователя

/// </summary>

public string Name { get; set; }

/// <summary>

/// Флаг блокировки пользователя

/// </summary>

public bool IsBlock { get; set; }

/// <summary>

/// Флаг наложение ограничений на пользователя

/// </summary>

public bool IsLimit { get; set; }

/// <summary>

/// Логин пользователя/админа

/// </summary>

public string Login { get; set; }

/// <summary>

/// пароль пользователя/админа

/// </summary>

public string Password { get; set; }

/// <summary>

/// флаг, показывающий заход на форму под админом или нет

/// </summary>

public bool IsAdmin { get; set; }

}

}

Листинг кода шифрвания

Шифрование алгоритмом Des с использованием встроенных в c# технологий.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Security.Cryptography;

using System.IO;

using System.Windows;

namespace IB1.Service

{

public static class DES

{

/// <summary>

/// ключ для шифрованиия и разшифрования файлов

/// </summary>

static readonly string sKey = "adminMax";

/// <summary>

/// шифрование файла

/// </summary>

/// <param name="source">путь файла котого надо шифровать</param>

/// <param name="pathEncrypt">путь в какой файл зашифровать</param>

public static void EncryptFile(string source, string pathEncrypt)

{

FileStream fsInput = new FileStream(source, FileMode.Open, FileAccess.Read);

FileStream fsEncrypt = new FileStream(pathEncrypt, FileMode.Create, FileAccess.Write);

DESCryptoServiceProvider DES = new DESCryptoServiceProvider

{

Key = Encoding.ASCII.GetBytes(sKey),

IV = Encoding.ASCII.GetBytes(sKey)

};

ICryptoTransform desencrypt = DES.CreateEncryptor();

CryptoStream cryptoStream = new CryptoStream(fsEncrypt, desencrypt, CryptoStreamMode.Write);

byte[] bytearrayinput = new byte[fsInput.Length - 0];

fsInput.Read(bytearrayinput, 0, bytearrayinput.Length);

cryptoStream.Write(bytearrayinput, 0, bytearrayinput.Length);

cryptoStream.Close();

fsInput.Close();

fsEncrypt.Close();

}

/// <summary>

/// расшифрование файла

/// </summary>

/// <param name="source">путь файла котого надо расшифровать</param>

/// <param name="pathEncrypt">путь в какой файл расшифровать</param>

public static void DecryptFile(string source, string pathEncrypt)

{

FileStream fsInput = new FileStream(source, FileMode.Open, FileAccess.Read);

FileStream fsEncrypt = new FileStream(pathEncrypt, FileMode.Create, FileAccess.Write);

DESCryptoServiceProvider DES = new DESCryptoServiceProvider

{

Key = Encoding.ASCII.GetBytes(sKey),

IV = Encoding.ASCII.GetBytes(sKey)

};

ICryptoTransform desencrypt = DES.CreateDecryptor();

CryptoStream cryptoStream = new CryptoStream(fsEncrypt, desencrypt, CryptoStreamMode.Write);

byte[] bytearrayinput = new byte[fsInput.Length - 0];

fsInput.Read(bytearrayinput, 0, bytearrayinput.Length);

cryptoStream.Write(bytearrayinput, 0, bytearrayinput.Length);

cryptoStream.Close();

fsInput.Close();

fsEncrypt.Close();

}

/// <summary>

/// Хеширование паролей методом sha

/// </summary>

/// <param name="s">строка пароля</param>

/// <returns></returns>

public static string ToSHA256(string s)

{

var sha256 = SHA256.Create();

byte[] bytes = sha256.ComputeHash(Encoding.UTF8.GetBytes(s));

var sb = new StringBuilder();

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

{

sb.Append(bytes[i].ToString("x2"));

}

return sb.ToString();

}

}

}