Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

«МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ» (МОСКОВСКИЙ ПОЛИТЕХ)

КУРСОВОЙ ПРОЕКТ

По курсу «Разработка мобильных приложений»

ТЕМА

**«Календарь Windows»**

Выполнил Солдатов Денис Александрович

*(ФИО)*

Группа 221–329

Проверил Денис Васильев

*(ФИО)*

Москва, 2025

[**Цель:** 3](#_Toc195484351)

[**Функциональность проекта** 3](#_Toc195484352)

[**Реализация в коде** 4](#_Toc195484353)

[**Демонстрация проекта** 46](#_Toc195484354)

[**Заключение** 52](#_Toc195484355)

# **Цель:**

Целью курсовой работы является разработка безопасного календарного приложения на C# под Windows, которое обеспечивает большую часть функционала современного календаря. Программа реализована в виде трех интерфейсов: WinForms, WPF и консольного приложения.

Задачи проекта:  
- Реализовать интерфейс календаря и отображение списка событий.  
- Обеспечить взаимодействие с пользователем в разных графических оболочках и консоли используя собственную библиотеку классов.  
- Добавить возможность шифрования и дешифрования данных при сохранении и загрузке.  
- Добавить защиту паролем и функциональность сброса через контрольные вопросы.

# **Функциональность проекта**

Приложение поддерживает следующие функции:  
- Добавление, удаление, редактирование и отображение событий.  
- Хранение событий в зашифрованном виде с использованием AES (Rfc2898DeriveBytes).  
- Настройка шифрования, включая включение/отключение и смену пароля.  
- Защита паролем при запуске и возможность сброса пароля по контрольным вопросам.  
- Автоматическое уведомление пользователя при наступлении события.  
- Кроссплатформенная реализация (WinForms, WPF, Console).

**Архитектура и реализация**

Приложение реализовано с использованием паттернов разделения логики:  
- Библиотека CalendarLibrary содержит бизнес-логику, модели, шифрование и хранилище.  
- Интерфейсы WinForms, WPF и Console используют общую логику библиотеки.  
- Для хранения используется формат JSON, зашифрованный по алгоритму AES (CBC).

-Все функции разделены по классам с соответствующими именованиями.

# **Реализация в коде**

Листинг 1: Calendar Console. Program.cs.

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using CalendarLibrary;

namespace CalendarConsole

{

class Program

{

static void Main()

{

Console.OutputEncoding = Encoding.UTF8;

Console.InputEncoding = Encoding.UTF8;

Console.Title = "Консольный календарь";

Console.WriteLine("Добро пожаловать в консольный календарь!");

var settings = SettingsStorage.LoadSettings();

if (settings.EncryptionEnabled && !string.IsNullOrEmpty(settings.EncryptionPassword))

{

bool authenticated = false;

while (!authenticated)

{

Console.Write("Введите пароль (или введите RESET для сброса): ");

var input = Console.ReadLine();

if (input == "RESET")

{

if (TryResetPassword())

{

settings = SettingsStorage.LoadSettings();

authenticated = true;

}

else

{

Console.WriteLine("Сброс не удался. Повторите попытку.");

}

}

else if (input == settings.EncryptionPassword)

{

authenticated = true;

}

else

{

Console.WriteLine("Неверный пароль.");

}

}

}

var calendar = new CalendarManager();

try

{

var loadedEvents = DataStorage.LoadEvents(settings.EncryptionEnabled, settings.EncryptionPassword);

foreach (var ev in loadedEvents)

calendar.AddEvent(ev);

}

catch (Exception ex)

{

Console.WriteLine("Ошибка загрузки событий: " + ex.Message);

}

while (true)

{

Console.WriteLine("\nМеню:");

Console.WriteLine("1 - Добавить событие");

Console.WriteLine("2 - Показать события");

Console.WriteLine("3 - Удалить событие");

Console.WriteLine("4 - Изменить настройки шифрования");

Console.WriteLine("5 - Выход");

Console.Write("Выбор: ");

string choice = Console.ReadLine();

switch (choice)

{

case "1":

AddEvent(calendar);

break;

case "2":

ShowEvents(calendar);

break;

case "3":

DeleteEvent(calendar);

break;

case "4":

ChangeEncryptionSettings();

break;

case "5":

SaveAndExit(calendar, settings);

return;

default:

Console.WriteLine("Неверный выбор.");

break;

}

}

}

static void AddEvent(CalendarManager calendar)

{

Console.Write("Дата (гггг-мм-дд): ");

string input = Console.ReadLine();

if (!DateTime.TryParse(input, out DateTime date))

{

Console.WriteLine("Неверный формат.");

return;

}

Console.Write("Заголовок: ");

string title = Console.ReadLine();

Console.Write("Описание: ");

string desc = Console.ReadLine();

calendar.AddEvent(new CalendarEvent(date, title, desc));

Console.WriteLine("Событие добавлено.");

}

static void ShowEvents(CalendarManager calendar)

{

var events = calendar.GetEvents().ToList();

if (events.Count == 0)

{

Console.WriteLine("Событий нет.");

return;

}

Console.WriteLine("Список событий:");

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

{

Console.WriteLine($"{i + 1}. {events[i].ToDetailedString()}");

}

}

static void DeleteEvent(CalendarManager calendar)

{

var events = calendar.GetEvents().ToList();

if (!events.Any())

{

Console.WriteLine("Нет событий для удаления.");

return;

}

ShowEvents(calendar);

Console.Write("Введите номер события для удаления: ");

if (int.TryParse(Console.ReadLine(), out int index) && index > 0 && index <= events.Count)

{

calendar.RemoveEvent(events[index - 1]);

Console.WriteLine("Событие удалено.");

}

else

{

Console.WriteLine("Неверный индекс.");

}

}

static void SaveAndExit(CalendarManager calendar, EncryptionSettings settings)

{

try

{

DataStorage.SaveEvents(calendar.GetEvents(), settings.EncryptionEnabled, settings.EncryptionPassword);

Console.WriteLine("События сохранены. До свидания!");

}

catch (Exception ex)

{

Console.WriteLine("Ошибка при сохранении: " + ex.Message);

}

}

static bool TryResetPassword()

{

var questions = SecurityQuestionsStorage.LoadQuestions();

Console.WriteLine("Для сброса пароля ответьте на контрольные вопросы:");

foreach (var q in questions)

{

Console.Write(q.Question + " ");

if (!Console.ReadLine().Trim().Equals(q.Answer, StringComparison.OrdinalIgnoreCase))

return false;

}

Console.Write("Введите новый пароль: ");

string newPwd = Console.ReadLine();

if (string.IsNullOrWhiteSpace(newPwd) || newPwd.Length < 6)

{

Console.WriteLine("Пароль должен быть не менее 6 символов.");

return false;

}

var newSettings = new EncryptionSettings

{

EncryptionEnabled = true,

EncryptionPassword = newPwd

};

SettingsStorage.SaveSettings(newSettings);

return true;

}

static void ChangeEncryptionSettings()

{

Console.Write("Включить шифрование? (да/нет): ");

string enable = Console.ReadLine().ToLower();

if (enable == "да")

{

Console.Write("Введите новый пароль: ");

string pwd = Console.ReadLine();

if (pwd.Length < 6)

{

Console.WriteLine("Пароль слишком короткий.");

return;

}

var newSettings = new EncryptionSettings

{

EncryptionEnabled = true,

EncryptionPassword = pwd

};

SettingsStorage.SaveSettings(newSettings);

Console.WriteLine("Шифрование включено.");

}

else

{

Console.WriteLine("Ответьте на контрольные вопросы для отключения шифрования:");

if (TryResetPassword())

{

var newSettings = new EncryptionSettings

{

EncryptionEnabled = false,

EncryptionPassword = string.Empty

};

SettingsStorage.SaveSettings(newSettings);

Console.WriteLine("Шифрование отключено.");

}

else

{

Console.WriteLine("Неверные ответы.");

}

}

}

}

}

**Calendar Library-собственная библиотека классов реализующая основы приложений.**

Листинг 2 CalendarEvent.cs

using System;

using System.Collections.Generic;

namespace CalendarLibrary

{

public class CalendarEvent

{

public DateTime Date { get; set; }

public string Title { get; set; }

public string Description { get; set; }

public string Geolocation { get; set; }

public List<string> Attachments { get; set; }

public bool Notified { get; set; }

public CalendarEvent(DateTime date, string title, string description, string geolocation = "", List<string> attachments = null)

{

Date = date;

Title = title;

Description = description;

Geolocation = geolocation;

Attachments = attachments ?? new List<string>();

Notified = false;

}

public override string ToString()

{

return $"{Date.ToString("g")} - {Title}";

}

public string ToDetailedString()

{

string attachmentsStr = (Attachments != null && Attachments.Count > 0) ? string.Join(", ", Attachments) : "Нет";

return $"Дата: {Date.ToString("f")}\nЗаголовок: {Title}\nОписание: {Description}\nМестоположение: {Geolocation}\nВложения: {attachmentsStr}";

}

}

}

Листинг 3 Calendar Manager.cs

using System;

using System.Collections.Generic;

using System.Threading;

namespace CalendarLibrary

{

public class CalendarManager

{

private List<CalendarEvent> \_events;

private Dictionary<CalendarEvent, Timer> \_eventTimers;

public event Action<CalendarEvent> EventTimeReached;

public CalendarManager()

{

\_events = new List<CalendarEvent>();

\_eventTimers = new Dictionary<CalendarEvent, Timer>();

}

public void AddEvent(CalendarEvent ev)

{

\_events.Add(ev);

if (ev.Date > DateTime.Now)

{

CreateTimerForEvent(ev);

}

else if (!ev.Notified)

{

ev.Notified = true;

OnEventTimeReached(ev);

}

}

private void CreateTimerForEvent(CalendarEvent ev)

{

double ms = (ev.Date - DateTime.Now).TotalMilliseconds;

if (ms < 0)

ms = 0;

int dueTime = ms > int.MaxValue ? int.MaxValue : (int)ms;

Timer timer = new Timer((state) =>

{

ev.Notified = true;

OnEventTimeReached(ev);

RemoveTimer(ev);

}, null, dueTime, Timeout.Infinite);

\_eventTimers[ev] = timer;

}

private void OnEventTimeReached(CalendarEvent ev)

{

EventTimeReached?.Invoke(ev);

}

private void RemoveTimer(CalendarEvent ev)

{

if (\_eventTimers.ContainsKey(ev))

{

\_eventTimers[ev].Dispose();

\_eventTimers.Remove(ev);

}

}

public void RemoveEvent(CalendarEvent ev)

{

\_events.Remove(ev);

if (\_eventTimers.ContainsKey(ev))

{

\_eventTimers[ev].Dispose();

\_eventTimers.Remove(ev);

}

}

public void EditEvent(CalendarEvent oldEvent, CalendarEvent newEvent)

{

RemoveEvent(oldEvent);

AddEvent(newEvent);

}

public IEnumerable<CalendarEvent> GetEvents()

{

return \_events;

}

}

}

Листинг 4: DataStorage.cs

using System;

using System.Collections.Generic;

using System.IO;

using System.Security.Cryptography;

using System.Text;

using System.Text.Json;

namespace CalendarLibrary

{

public static class DataStorage

{

private static string filePath = Path.Combine(AppDomain.CurrentDomain.BaseDirectory, "events.dat");

private static string logFilePath = Path.Combine(AppDomain.CurrentDomain.BaseDirectory, "error.log");

public static void SaveEvents(IEnumerable<CalendarEvent> events, bool encryptionEnabled, string password)

{

var options = new JsonSerializerOptions { WriteIndented = true };

string json = JsonSerializer.Serialize(events, options);

if (encryptionEnabled)

{

try

{

json = EncryptString(json, password);

}

catch (Exception ex)

{

LogError("Ошибка шифрования: " + ex.Message);

}

}

File.WriteAllText(filePath, json, Encoding.UTF8);

}

public static List<CalendarEvent> LoadEvents(bool encryptionEnabled, string password)

{

if (!File.Exists(filePath))

return new List<CalendarEvent>();

string content = File.ReadAllText(filePath, Encoding.UTF8);

if (encryptionEnabled)

{

if (!IsBase64String(content))

{

LogError("Данные файла не являются корректной строкой Base-64. Файл, возможно, не зашифрован или поврежден.");

ResetFile();

return new List<CalendarEvent>();

}

try

{

content = DecryptString(content, password);

}

catch (Exception ex)

{

LogError("Ошибка дешифрования файла событий: " + ex.Message);

ResetFile();

return new List<CalendarEvent>();

}

}

try

{

return JsonSerializer.Deserialize<List<CalendarEvent>>(content);

}

catch (Exception ex)

{

LogError("Ошибка десериализации JSON: " + ex.Message);

return new List<CalendarEvent>();

}

}

#region Шифрование / Дешифрование

private static string EncryptString(string plainText, string password)

{

byte[] salt = GenerateRandomBytes(16);

byte[] iv = GenerateRandomBytes(16);

string encrypted;

using (var keyDerivationFunction = new Rfc2898DeriveBytes(password, salt, 10000, HashAlgorithmName.SHA256))

{

byte[] key = keyDerivationFunction.GetBytes(32); // 256 бит

using (Aes aes = Aes.Create())

{

aes.Key = key;

aes.IV = iv;

aes.Mode = CipherMode.CBC;

using (MemoryStream ms = new MemoryStream())

{

ms.Write(salt, 0, salt.Length);

ms.Write(iv, 0, iv.Length);

using (CryptoStream cs = new CryptoStream(ms, aes.CreateEncryptor(), CryptoStreamMode.Write))

{

using (StreamWriter sw = new StreamWriter(cs))

{

sw.Write(plainText);

}

}

encrypted = Convert.ToBase64String(ms.ToArray());

}

}

}

return encrypted;

}

private static string DecryptString(string cipherText, string password)

{

byte[] fullCipher;

try

{

fullCipher = Convert.FromBase64String(cipherText);

}

catch (FormatException ex)

{

throw new Exception("Ошибка: данные файла не являются корректной строкой Base-64. Возможно, файл не зашифрован или поврежден.", ex);

}

byte[] salt = new byte[16];

byte[] iv = new byte[16];

Array.Copy(fullCipher, 0, salt, 0, salt.Length);

Array.Copy(fullCipher, salt.Length, iv, 0, iv.Length);

byte[] cipher = new byte[fullCipher.Length - salt.Length - iv.Length];

Array.Copy(fullCipher, salt.Length + iv.Length, cipher, 0, cipher.Length);

string decrypted;

using (var keyDerivationFunction = new Rfc2898DeriveBytes(password, salt, 10000, HashAlgorithmName.SHA256))

{

byte[] key = keyDerivationFunction.GetBytes(32);

using (Aes aes = Aes.Create())

{

aes.Key = key;

aes.IV = iv;

aes.Mode = CipherMode.CBC;

using (MemoryStream ms = new MemoryStream(cipher))

{

using (CryptoStream cs = new CryptoStream(ms, aes.CreateDecryptor(), CryptoStreamMode.Read))

{

using (StreamReader reader = new StreamReader(cs))

{

decrypted = reader.ReadToEnd();

}

}

}

}

}

return decrypted;

}

private static byte[] GenerateRandomBytes(int count)

{

byte[] bytes = new byte[count];

using (RNGCryptoServiceProvider rng = new RNGCryptoServiceProvider())

{

rng.GetBytes(bytes);

}

return bytes;

}

#endregion

#region Вспомогательные методы обработки ошибок

private static void LogError(string message)

{

try

{

string logEntry = $"{DateTime.Now}: {message}{Environment.NewLine}";

File.AppendAllText(Path.Combine(AppDomain.CurrentDomain.BaseDirectory, "error.log"), logEntry, Encoding.UTF8);

}

catch

{

}

}

private static bool IsBase64String(string s)

{

if (string.IsNullOrWhiteSpace(s))

return false;

s = s.Trim();

if (s.Length % 4 != 0)

return false;

try

{

Convert.FromBase64String(s);

return true;

}

catch

{

return false;

}

}

private static void ResetFile()

{

try

{

File.Delete(filePath);

}

catch (Exception ex)

{

LogError("Не удалось сбросить файл событий: " + ex.Message);

}

}

#endregion

}

}

Листинг 5 EncryptionSettings.cs

using System;

namespace CalendarLibrary

{

public class EncryptionSettings

{

public bool EncryptionEnabled { get; set; } = false;

public string EncryptionPassword { get; set; } = string.Empty;

}

}

Листинг 6: SecurityQuestionsStorage.cs

using System;

using System.Collections.Generic;

using System.IO;

using System.Text;

using System.Text.Json;

namespace CalendarLibrary

{

public class SecurityQuestion

{

public string Question { get; set; }

public string Answer { get; set; }

}

public static class SecurityQuestionsStorage

{

private static readonly string questionsFilePath = Path.Combine(AppDomain.CurrentDomain.BaseDirectory, "securityQuestions.json");

public static List<SecurityQuestion> LoadQuestions()

{

if (!File.Exists(questionsFilePath))

{

var defaultQuestions = new List<SecurityQuestion>

{

new SecurityQuestion { Question = "Как зовут вашего питомца?", Answer = "Fluffy" },

new SecurityQuestion { Question = "В каком городе вы родились?", Answer = "Moscow" }

};

SaveQuestions(defaultQuestions);

return defaultQuestions;

}

string json = File.ReadAllText(questionsFilePath, Encoding.UTF8);

try

{

return JsonSerializer.Deserialize<List<SecurityQuestion>>(json);

}

catch

{

return new List<SecurityQuestion>();

}

}

public static void SaveQuestions(List<SecurityQuestion> questions)

{

string json = JsonSerializer.Serialize(questions, new JsonSerializerOptions { WriteIndented = true });

File.WriteAllText(questionsFilePath, json, Encoding.UTF8);

}

}

}

Листинг 7 :SettingsStorage.cs

using System;

using System.IO;

using System.Text;

using System.Text.Json;

namespace CalendarLibrary

{

public static class SettingsStorage

{

private static readonly string SettingsFilePath = Path.Combine(AppDomain.CurrentDomain.BaseDirectory, "encryptionSettings.json");

public static EncryptionSettings LoadSettings()

{

if (!File.Exists(SettingsFilePath))

return new EncryptionSettings();

string json = File.ReadAllText(SettingsFilePath, Encoding.UTF8);

try

{

return JsonSerializer.Deserialize<EncryptionSettings>(json);

}

catch

{

return new EncryptionSettings();

}

}

public static void SaveSettings(EncryptionSettings settings)

{

string json = JsonSerializer.Serialize(settings, new JsonSerializerOptions { WriteIndented = true });

File.WriteAllText(SettingsFilePath, json, Encoding.UTF8);

}

}

}

Листинг 8 EncryptionSettingsForm.cs:

using System;

using System.Windows.Forms;

using CalendarLibrary;

namespace CalendarWinForms

{

public partial class EncryptionSettingsForm : Form

{

public EncryptionSettingsForm()

{

InitializeComponent();

var settings = SettingsStorage.LoadSettings();

chkEnableEncryption.Checked = settings.EncryptionEnabled;

tbNewPassword.Text = "";

tbConfirmPassword.Text = "";

}

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

{

bool enabled = chkEnableEncryption.Checked;

string pwd = tbNewPassword.Text;

string pwdConfirm = tbConfirmPassword.Text;

if (enabled)

{

if (string.IsNullOrWhiteSpace(pwd))

{

MessageBox.Show("Пароль не может быть пустым.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Warning);

return;

}

if (pwd.Length < 6)

{

MessageBox.Show("Пароль должен содержать не менее 6 символов.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Warning);

return;

}

if (pwd != pwdConfirm)

{

MessageBox.Show("Пароль и подтверждение не совпадают.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Warning);

return;

}

}

var newSettings = new EncryptionSettings

{

EncryptionEnabled = enabled,

EncryptionPassword = enabled ? pwd : string.Empty

};

SettingsStorage.SaveSettings(newSettings);

MessageBox.Show("Настройки шифрования сохранены.", "Информация", MessageBoxButtons.OK, MessageBoxIcon.Information);

DialogResult = DialogResult.OK;

Close();

}

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

{

DialogResult = DialogResult.Cancel;

Close();

}

private void EncryptionSettingsForm\_Load(object sender, EventArgs e)

{

}

}

}

Листинг 9: EventEditorForm.cs

using System;

using System.Collections.Generic;

using System.Windows.Forms;

using CalendarLibrary;

using System.Device.Location;

namespace CalendarWinForms

{

public partial class EventEditorForm : Form

{

public CalendarEvent CalendarEvent { get; private set; }

public EventEditorForm()

{

InitializeComponent();

CalendarEvent = new CalendarEvent(DateTime.Now, "", "", "");

}

public void SetEvent(CalendarEvent ev)

{

CalendarEvent = ev;

dtpDate.Value = ev.Date;

txtTitle.Text = ev.Title;

txtDescription.Text = ev.Description;

txtGeolocation.Text = ev.Geolocation;

lbAttachments.Items.Clear();

foreach (var file in ev.Attachments)

{

lbAttachments.Items.Add(file);

}

}

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

{

using (OpenFileDialog openFile = new OpenFileDialog())

{

if (openFile.ShowDialog() == DialogResult.OK)

{

string filePath = openFile.FileName;

lbAttachments.Items.Add(filePath);

}

}

}

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

{

if (lbAttachments.SelectedItem != null)

{

lbAttachments.Items.Remove(lbAttachments.SelectedItem);

}

}

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

{

if (string.IsNullOrWhiteSpace(txtTitle.Text))

{

MessageBox.Show("Заголовок не может быть пустым.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

var attachments = new List<string>();

foreach (var item in lbAttachments.Items)

{

attachments.Add(item.ToString());

}

CalendarEvent = new CalendarEvent(

dtpDate.Value,

txtTitle.Text.Trim(),

txtDescription.Text.Trim(),

txtGeolocation.Text.Trim(),

attachments

);

DialogResult = DialogResult.OK;

Close();

}

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

{

try

{

GeoCoordinateWatcher watcher = new GeoCoordinateWatcher(GeoPositionAccuracy.Default);

bool started = watcher.TryStart(false, TimeSpan.FromSeconds(5));

if (started)

{

var coord = watcher.Position.Location;

if (coord.IsUnknown)

{

MessageBox.Show("Не удалось определить текущее местоположение.", "Информация", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

else

{

txtGeolocation.Text = $"Lat: {coord.Latitude:F6}, Lon: {coord.Longitude:F6}";

}

}

else

{

MessageBox.Show("Не удалось запустить определение местоположения.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

catch (Exception ex)

{

MessageBox.Show("Ошибка при получении местоположения: " + ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

}

Листинг 10: MainForm.cs

using System;

using System.Collections.Generic;

using System.Windows.Forms;

using CalendarLibrary;

using System.Device.Location;

namespace CalendarWinForms

{

public partial class EventEditorForm : Form

{

public CalendarEvent CalendarEvent { get; private set; }

public EventEditorForm()

{

InitializeComponent();

CalendarEvent = new CalendarEvent(DateTime.Now, "", "", "");

}

public void SetEvent(CalendarEvent ev)

{

CalendarEvent = ev;

dtpDate.Value = ev.Date;

txtTitle.Text = ev.Title;

txtDescription.Text = ev.Description;

txtGeolocation.Text = ev.Geolocation;

lbAttachments.Items.Clear();

foreach (var file in ev.Attachments)

{

lbAttachments.Items.Add(file);

}

}

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

{

using (OpenFileDialog openFile = new OpenFileDialog())

{

if (openFile.ShowDialog() == DialogResult.OK)

{

string filePath = openFile.FileName;

lbAttachments.Items.Add(filePath);

}

}

}

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

{

if (lbAttachments.SelectedItem != null)

{

lbAttachments.Items.Remove(lbAttachments.SelectedItem);

}

}

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

{

if (string.IsNullOrWhiteSpace(txtTitle.Text))

{

MessageBox.Show("Заголовок не может быть пустым.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

var attachments = new List<string>();

foreach (var item in lbAttachments.Items)

{

attachments.Add(item.ToString());

}

CalendarEvent = new CalendarEvent(

dtpDate.Value,

txtTitle.Text.Trim(),

txtDescription.Text.Trim(),

txtGeolocation.Text.Trim(),

attachments

);

DialogResult = DialogResult.OK;

Close();

}

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

{

try

{

GeoCoordinateWatcher watcher = new GeoCoordinateWatcher(GeoPositionAccuracy.Default);

bool started = watcher.TryStart(false, TimeSpan.FromSeconds(5));

if (started)

{

var coord = watcher.Position.Location;

if (coord.IsUnknown)

{

MessageBox.Show("Не удалось определить текущее местоположение.", "Информация", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

else

{

txtGeolocation.Text = $"Lat: {coord.Latitude:F6}, Lon: {coord.Longitude:F6}";

}

}

else

{

MessageBox.Show("Не удалось запустить определение местоположения.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

catch (Exception ex)

{

MessageBox.Show("Ошибка при получении местоположения: " + ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

}

Листинг 11: EventEditorForm.cs

using System;

using System.Collections.Generic;

using System.Windows.Forms;

using CalendarLibrary;

using System.Device.Location;

namespace CalendarWinForms

{

public partial class EventEditorForm : Form

{

public CalendarEvent CalendarEvent { get; private set; }

public EventEditorForm()

{

InitializeComponent();

CalendarEvent = new CalendarEvent(DateTime.Now, "", "", "");

}

public void SetEvent(CalendarEvent ev)

{

CalendarEvent = ev;

dtpDate.Value = ev.Date;

txtTitle.Text = ev.Title;

txtDescription.Text = ev.Description;

txtGeolocation.Text = ev.Geolocation;

lbAttachments.Items.Clear();

foreach (var file in ev.Attachments)

{

lbAttachments.Items.Add(file);

}

}

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

{

using (OpenFileDialog openFile = new OpenFileDialog())

{

if (openFile.ShowDialog() == DialogResult.OK)

{

string filePath = openFile.FileName;

lbAttachments.Items.Add(filePath);

}

}

}

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

{

if (lbAttachments.SelectedItem != null)

{

lbAttachments.Items.Remove(lbAttachments.SelectedItem);

}

}

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

{

if (string.IsNullOrWhiteSpace(txtTitle.Text))

{

MessageBox.Show("Заголовок не может быть пустым.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

var attachments = new List<string>();

foreach (var item in lbAttachments.Items)

{

attachments.Add(item.ToString());

}

CalendarEvent = new CalendarEvent(

dtpDate.Value,

txtTitle.Text.Trim(),

txtDescription.Text.Trim(),

txtGeolocation.Text.Trim(),

attachments

);

DialogResult = DialogResult.OK;

Close();

}

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

{

try

{

GeoCoordinateWatcher watcher = new GeoCoordinateWatcher(GeoPositionAccuracy.Default);

bool started = watcher.TryStart(false, TimeSpan.FromSeconds(5));

if (started)

{

var coord = watcher.Position.Location;

if (coord.IsUnknown)

{

MessageBox.Show("Не удалось определить текущее местоположение.", "Информация", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

else

{

txtGeolocation.Text = $"Lat: {coord.Latitude:F6}, Lon: {coord.Longitude:F6}";

}

}

else

{

MessageBox.Show("Не удалось запустить определение местоположения.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

catch (Exception ex)

{

MessageBox.Show("Ошибка при получении местоположения: " + ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

}

Листинг 12: PasswordEntryForm.cs

using System;

using System.Windows.Forms;

using CalendarLibrary;

namespace CalendarWinForms

{

public partial class PasswordEntryForm : Form

{

public string StoredPassword { get; set; }

public PasswordEntryForm()

{

InitializeComponent();

}

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

{

string inputPassword = tbPassword.Text.Trim();

if (inputPassword == StoredPassword)

{

DialogResult = DialogResult.OK;

Close();

}

else

{

MessageBox.Show("Неверный пароль. Попробуйте ещё раз.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

tbPassword.Clear();

tbPassword.Focus();

}

}

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

{

DialogResult = DialogResult.Cancel;

Close();

}

private void PasswordEntryForm\_Load(object sender, EventArgs e)

{

}

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

{

using (var resetForm = new ResetPasswordForm())

{

if (resetForm.ShowDialog() == DialogResult.OK)

{

var settings = SettingsStorage.LoadSettings();

StoredPassword = settings.EncryptionPassword;

MessageBox.Show("Пароль успешно сброшен.", "Информация", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

}

}

}

Листинг 13: Program.cs

using System;

using System.Windows.Forms;

using CalendarLibrary;

namespace CalendarWinForms

{

static class Program

{

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

var encryptionSettings = SettingsStorage.LoadSettings();

if (encryptionSettings.EncryptionEnabled && !string.IsNullOrEmpty(encryptionSettings.EncryptionPassword))

{

using (var passForm = new PasswordEntryForm())

{

passForm.StoredPassword = encryptionSettings.EncryptionPassword;

if (passForm.ShowDialog() != DialogResult.OK)

return;

}

}

Application.Run(new MainForm());

}

}

}

Листинг 14: ResetPasswordForm.cs

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Linq;

using System.Windows.Forms;

using CalendarLibrary;

namespace CalendarWinForms

{

public partial class ResetPasswordForm : Form

{

private List<SecurityQuestion> securityQuestions;

private Dictionary<SecurityQuestion, TextBox> answerControls;

public ResetPasswordForm()

{

InitializeComponent();

answerControls = new Dictionary<SecurityQuestion, TextBox>();

securityQuestions = SecurityQuestionsStorage.LoadQuestions();

PopulateQuestions();

}

private void PopulateQuestions()

{

panelQuestions.Controls.Clear();

answerControls.Clear();

int yOffset = 10;

foreach (var question in securityQuestions)

{

Label lblQuestion = new Label();

lblQuestion.AutoSize = true;

lblQuestion.Location = new Point(10, yOffset);

lblQuestion.Text = question.Question;

panelQuestions.Controls.Add(lblQuestion);

yOffset += lblQuestion.Height + 5;

TextBox tbAnswer = new TextBox();

tbAnswer.Location = new Point(10, yOffset);

tbAnswer.Width = panelQuestions.Width - 20;

panelQuestions.Controls.Add(tbAnswer);

answerControls[question] = tbAnswer;

yOffset += tbAnswer.Height + 15;

}

}

private bool ValidateControlQuestions()

{

foreach (var pair in answerControls)

{

SecurityQuestion question = pair.Key;

string userAnswer = pair.Value.Text.Trim();

if (string.IsNullOrEmpty(userAnswer))

{

MessageBox.Show($"Ответ на вопрос \"{question.Question}\" не должен быть пустым.",

"Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Warning);

return false;

}

if (!userAnswer.Equals(question.Answer, StringComparison.OrdinalIgnoreCase))

{

MessageBox.Show($"Неверный ответ на вопрос: \"{question.Question}\"",

"Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return false;

}

}

return true;

}

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

{

if (!ValidateControlQuestions())

return;

string newPassword = tbNewPassword.Text.Trim();

string confirmPassword = tbConfirmPassword.Text.Trim();

if (string.IsNullOrEmpty(newPassword))

{

MessageBox.Show("Новый пароль не может быть пустым.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Warning);

return;

}

if (newPassword.Length < 6)

{

MessageBox.Show("Пароль должен содержать не менее 6 символов.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Warning);

return;

}

if (!newPassword.Equals(confirmPassword))

{

MessageBox.Show("Пароль и подтверждение не совпадают.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Warning);

return;

}

var newSettings = new EncryptionSettings

{

EncryptionEnabled = true,

EncryptionPassword = newPassword

};

SettingsStorage.SaveSettings(newSettings);

MessageBox.Show("Пароль успешно сброшен.", "Информация", MessageBoxButtons.OK, MessageBoxIcon.Information);

DialogResult = DialogResult.OK;

Close();

}

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

{

if (!ValidateControlQuestions())

return;

DialogResult dr = MessageBox.Show("Вы уверены, что хотите отключить шифрование данных событий?\nДанные будут сохранены в открытом виде.",

"Подтверждение", MessageBoxButtons.YesNo, MessageBoxIcon.Question);

if (dr != DialogResult.Yes)

return;

var newSettings = new EncryptionSettings

{

EncryptionEnabled = false,

EncryptionPassword = string.Empty

};

SettingsStorage.SaveSettings(newSettings);

MessageBox.Show("Шифрование отключено.", "Информация", MessageBoxButtons.OK, MessageBoxIcon.Information);

DialogResult = DialogResult.OK;

Close();

}

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

{

DialogResult = DialogResult.Cancel;

Close();

}

private void ResetPasswordForm\_Load(object sender, EventArgs e)

{

}

}

}

CalendarWPF

Листинг 15:MainWindow.xaml

<Window x:Class="CalendarWPF.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

Title="Календарь WPF" Height="400" Width="600"

WindowStartupLocation="CenterScreen">

<Grid Margin="10">

<Grid.RowDefinitions>

<RowDefinition Height="\*"/>

<RowDefinition Height="Auto"/>

</Grid.RowDefinitions>

<ListBox x:Name="lbEvents" Grid.Row="0"

MouseDoubleClick="lbEvents\_MouseDoubleClick"

Margin="0,0,0,10"/>

<StackPanel Grid.Row="1" Orientation="Horizontal" HorizontalAlignment="Center">

<Button x:Name="btnAddEvent" Content="Добавить событие" Margin="5" Click="btnAddEvent\_Click"/>

<Button x:Name="btnRemoveEvent" Content="Удалить событие" Margin="5" Click="btnRemoveEvent\_Click"/>

<ComboBox x:Name="cbDisplayFormat" Margin="5" Width="150" SelectionChanged="cbDisplayFormat\_SelectionChanged"/>

</StackPanel>

</Grid>

</Window>

Листинг 15: MainWindow.xaml.cs

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Input;

using CalendarLibrary;

namespace CalendarWPF

{

public partial class MainWindow : Window

{

private CalendarManager calendar;

private EncryptionSettings encryptionSettings;

private System.Windows.Threading.DispatcherTimer notificationTimer;

private string eventsFile = Path.Combine(AppDomain.CurrentDomain.BaseDirectory, "events.dat");

public MainWindow()

{

InitializeComponent();

encryptionSettings = SettingsStorage.LoadSettings();

if (!encryptionSettings.EncryptionEnabled && string.IsNullOrEmpty(encryptionSettings.EncryptionPassword))

{

MessageBoxResult dr = MessageBox.Show("Вы хотите включить шифрование данных событий? Если да, настройте пароль.",

"Настройка шифрования", MessageBoxButton.YesNo, MessageBoxImage.Question);

if (dr == MessageBoxResult.Yes)

{

var frm = new EncryptionSettingsWindow();

if (frm.ShowDialog() == true)

{

encryptionSettings = SettingsStorage.LoadSettings();

}

}

}

calendar = new CalendarManager();

List<CalendarEvent> loadedEvents = DataStorage.LoadEvents(encryptionSettings.EncryptionEnabled, encryptionSettings.EncryptionPassword);

foreach (var ev in loadedEvents)

{

calendar.AddEvent(ev);

}

cbDisplayFormat.Items.Add("Краткий формат");

cbDisplayFormat.Items.Add("Подробный формат");

cbDisplayFormat.SelectedIndex = 0;

notificationTimer = new System.Windows.Threading.DispatcherTimer();

notificationTimer.Interval = TimeSpan.FromMinutes(1);

notificationTimer.Tick += NotificationTimer\_Tick;

notificationTimer.Start();

RefreshEventList();

}

private void NotificationTimer\_Tick(object sender, EventArgs e)

{

DateTime now = DateTime.Now;

DateTime nextCheck = now.Add(notificationTimer.Interval);

foreach (var ev in calendar.GetEvents())

{

if (!ev.Notified && ev.Date >= now && ev.Date <= nextCheck)

{

ev.Notified = true;

MessageBox.Show($"Событие \"{ev.Title}\" наступило!", "Уведомление", MessageBoxButton.OK, MessageBoxImage.Information);

}

}

}

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

{

var editor = new EventEditorWindow();

if (editor.ShowDialog() == true)

{

calendar.AddEvent(editor.CalendarEvent);

RefreshEventList();

}

}

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

{

if (lbEvents.SelectedItem is CalendarEvent selectedEvent)

{

calendar.RemoveEvent(selectedEvent);

RefreshEventList();

}

else

{

MessageBox.Show("Выберите событие для удаления.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Warning);

}

}

private void lbEvents\_MouseDoubleClick(object sender, MouseButtonEventArgs e)

{

if (lbEvents.SelectedItem is CalendarEvent ev)

{

var editor = new EventEditorWindow();

editor.SetEvent(ev);

if (editor.ShowDialog() == true)

{

ev.Date = editor.CalendarEvent.Date;

ev.Title = editor.CalendarEvent.Title;

ev.Description = editor.CalendarEvent.Description;

ev.Geolocation = editor.CalendarEvent.Geolocation;

ev.Attachments = editor.CalendarEvent.Attachments;

ev.Notified = false;

RefreshEventList();

}

}

}

private void cbDisplayFormat\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

RefreshEventList();

}

private string FormatEvent(CalendarEvent ev)

{

return cbDisplayFormat.SelectedIndex == 1 ? ev.ToDetailedString() : ev.ToString();

}

private void RefreshEventList()

{

lbEvents.ItemsSource = calendar.GetEvents().ToList();

}

}

}

Листинг 16: PasswordEntryWindow.xaml

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Input;

using CalendarLibrary;

namespace CalendarWPF

{

public partial class MainWindow : Window

{

private CalendarManager calendar;

private EncryptionSettings encryptionSettings;

private System.Windows.Threading.DispatcherTimer notificationTimer;

private string eventsFile = Path.Combine(AppDomain.CurrentDomain.BaseDirectory, "events.dat");

public MainWindow()

{

InitializeComponent();

encryptionSettings = SettingsStorage.LoadSettings();

if (!encryptionSettings.EncryptionEnabled && string.IsNullOrEmpty(encryptionSettings.EncryptionPassword))

{

MessageBoxResult dr = MessageBox.Show("Вы хотите включить шифрование данных событий? Если да, настройте пароль.",

"Настройка шифрования", MessageBoxButton.YesNo, MessageBoxImage.Question);

if (dr == MessageBoxResult.Yes)

{

var frm = new EncryptionSettingsWindow();

if (frm.ShowDialog() == true)

{

encryptionSettings = SettingsStorage.LoadSettings();

}

}

}

calendar = new CalendarManager();

List<CalendarEvent> loadedEvents = DataStorage.LoadEvents(encryptionSettings.EncryptionEnabled, encryptionSettings.EncryptionPassword);

foreach (var ev in loadedEvents)

{

calendar.AddEvent(ev);

}

cbDisplayFormat.Items.Add("Краткий формат");

cbDisplayFormat.Items.Add("Подробный формат");

cbDisplayFormat.SelectedIndex = 0;

notificationTimer = new System.Windows.Threading.DispatcherTimer();

notificationTimer.Interval = TimeSpan.FromMinutes(1);

notificationTimer.Tick += NotificationTimer\_Tick;

notificationTimer.Start();

RefreshEventList();

}

private void NotificationTimer\_Tick(object sender, EventArgs e)

{

DateTime now = DateTime.Now;

DateTime nextCheck = now.Add(notificationTimer.Interval);

foreach (var ev in calendar.GetEvents())

{

if (!ev.Notified && ev.Date >= now && ev.Date <= nextCheck)

{

ev.Notified = true;

MessageBox.Show($"Событие \"{ev.Title}\" наступило!", "Уведомление", MessageBoxButton.OK, MessageBoxImage.Information);

}

}

}

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

{

var editor = new EventEditorWindow();

if (editor.ShowDialog() == true)

{

calendar.AddEvent(editor.CalendarEvent);

RefreshEventList();

}

}

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

{

if (lbEvents.SelectedItem is CalendarEvent selectedEvent)

{

calendar.RemoveEvent(selectedEvent);

RefreshEventList();

}

else

{

MessageBox.Show("Выберите событие для удаления.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Warning);

}

}

private void lbEvents\_MouseDoubleClick(object sender, MouseButtonEventArgs e)

{

if (lbEvents.SelectedItem is CalendarEvent ev)

{

var editor = new EventEditorWindow();

editor.SetEvent(ev);

if (editor.ShowDialog() == true)

{

ev.Date = editor.CalendarEvent.Date;

ev.Title = editor.CalendarEvent.Title;

ev.Description = editor.CalendarEvent.Description;

ev.Geolocation = editor.CalendarEvent.Geolocation;

ev.Attachments = editor.CalendarEvent.Attachments;

ev.Notified = false;

RefreshEventList();

}

}

}

private void cbDisplayFormat\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

RefreshEventList();

}

private string FormatEvent(CalendarEvent ev)

{

return cbDisplayFormat.SelectedIndex == 1 ? ev.ToDetailedString() : ev.ToString();

}

private void RefreshEventList()

{

lbEvents.ItemsSource = calendar.GetEvents().ToList();

}

}

}

Листинг 17 PasswordEntryWindow.xaml.cs:

using System.Windows;

using CalendarLibrary;

namespace CalendarWPF

{

public partial class PasswordEntryWindow : Window

{

public string StoredPassword { get; set; }

public PasswordEntryWindow()

{

InitializeComponent();

}

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

{

string inputPassword = pbPassword.Password.Trim();

if (inputPassword == StoredPassword)

{

DialogResult = true;

Close();

}

else

{

MessageBox.Show("Неверный пароль. Попробуйте ещё раз.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

pbPassword.Clear();

pbPassword.Focus();

}

}

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

{

DialogResult = false;

Close();

}

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

{

var resetWindow = new ResetPasswordWindow();

if (resetWindow.ShowDialog() == true)

{

var settings = SettingsStorage.LoadSettings();

StoredPassword = settings.EncryptionPassword;

MessageBox.Show("Пароль успешно сброшен.", "Информация", MessageBoxButton.OK, MessageBoxImage.Information);

}

}

}

}

Листинг 18: ResetPasswordWindow.xaml

<Window x:Class="CalendarWPF.ResetPasswordWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

Title="Сброс пароля" Height="450" Width="500"

WindowStartupLocation="CenterScreen">

<Grid Margin="10">

<Grid.RowDefinitions>

<RowDefinition Height="Auto"/>

<RowDefinition Height="Auto"/>

<RowDefinition Height="Auto"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="Auto"/>

</Grid.RowDefinitions>

<ScrollViewer Grid.Row="0" Height="200">

<StackPanel x:Name="stackPanelQuestions" />

</ScrollViewer>

<StackPanel Orientation="Horizontal" Grid.Row="1" Margin="0,10,0,0">

<Label Content="Новый пароль:" Width="120" VerticalAlignment="Center"/>

<PasswordBox x:Name="pbNewPassword" Width="300"/>

</StackPanel>

<StackPanel Orientation="Horizontal" Grid.Row="2" Margin="0,10,0,0">

<Label Content="Подтвердите пароль:" Width="120" VerticalAlignment="Center"/>

<PasswordBox x:Name="pbConfirmPassword" Width="300"/>

</StackPanel>

<StackPanel Orientation="Horizontal" Grid.Row="4" HorizontalAlignment="Right" Margin="0,10,0,0">

<Button Content="Сбросить пароль" Width="140" Margin="5" Click="btnReset\_Click"/>

<Button Content="Отключить шифрование" Width="180" Margin="5" Click="btnDisableEncryption\_Click"/>

<Button Content="Отмена" Width="100" Margin="5" Click="btnCancel\_Click"/>

</StackPanel>

</Grid>

</Window>

Листинг 18: ResetPasswordWindow.xaml.cs

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Windows;

using System.Windows.Controls;

using CalendarLibrary;

namespace CalendarWPF

{

public partial class ResetPasswordWindow : Window

{

private List<SecurityQuestion> securityQuestions;

private Dictionary<SecurityQuestion, TextBox> answerControls;

public ResetPasswordWindow()

{

InitializeComponent();

answerControls = new Dictionary<SecurityQuestion, TextBox>();

securityQuestions = SecurityQuestionsStorage.LoadQuestions();

PopulateQuestions();

}

private void PopulateQuestions()

{

stackPanelQuestions.Children.Clear();

answerControls.Clear();

foreach (var question in securityQuestions)

{

var tbQuestion = new TextBlock

{

Text = question.Question,

Margin = new Thickness(5),

TextWrapping = TextWrapping.Wrap

};

stackPanelQuestions.Children.Add(tbQuestion);

var tbAnswer = new TextBox

{

Margin = new Thickness(5),

Width = 400

};

stackPanelQuestions.Children.Add(tbAnswer);

answerControls[question] = tbAnswer;

}

}

private bool ValidateControlQuestions()

{

foreach (var pair in answerControls)

{

SecurityQuestion question = pair.Key;

string userAnswer = pair.Value.Text.Trim();

if (string.IsNullOrEmpty(userAnswer))

{

MessageBox.Show($"Ответ на вопрос \"{question.Question}\" не может быть пустым.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Warning);

return false;

}

if (!userAnswer.Equals(question.Answer, StringComparison.OrdinalIgnoreCase))

{

MessageBox.Show($"Неверный ответ на вопрос: \"{question.Question}\"", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return false;

}

}

return true;

}

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

{

if (!ValidateControlQuestions())

return;

string newPassword = pbNewPassword.Password.Trim();

string confirmPassword = pbConfirmPassword.Password.Trim();

if (string.IsNullOrEmpty(newPassword))

{

MessageBox.Show("Новый пароль не может быть пустым.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Warning);

return;

}

if (newPassword.Length < 6)

{

MessageBox.Show("Пароль должен содержать не менее 6 символов.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Warning);

return;

}

if (!newPassword.Equals(confirmPassword))

{

MessageBox.Show("Пароль и подтверждение не совпадают.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Warning);

return;

}

var newSettings = new EncryptionSettings

{

EncryptionEnabled = true,

EncryptionPassword = newPassword

};

SettingsStorage.SaveSettings(newSettings);

MessageBox.Show("Пароль успешно сброшен.", "Информация", MessageBoxButton.OK, MessageBoxImage.Information);

DialogResult = true;

Close();

}

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

{

if (!ValidateControlQuestions())

return;

MessageBoxResult dr = MessageBox.Show("Вы уверены, что хотите отключить шифрование данных событий?\nДанные будут сохранены в открытом виде.",

"Подтверждение", MessageBoxButton.YesNo, MessageBoxImage.Question);

if (dr != MessageBoxResult.Yes)

return;

var newSettings = new EncryptionSettings

{

EncryptionEnabled = false,

EncryptionPassword = string.Empty

};

SettingsStorage.SaveSettings(newSettings);

MessageBox.Show("Шифрование отключено.", "Информация", MessageBoxButton.OK, MessageBoxImage.Information);

DialogResult = true;

Close();

}

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

{

DialogResult = false;

Close();

}

}

}

# **Демонстрация проекта**

CalendarConsole:

Изображение выглядит как текст, снимок экрана, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Calendar Winforms:

Изображение выглядит как текст, снимок экрана, Шрифт, число

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана, программное обеспечение, число

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана, программное обеспечение, компьютер

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана, Шрифт, Операционная система

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Контент, сгенерированный ИИ, может содержать ошибки.

Calendar WPF:

Изображение выглядит как текст, снимок экрана, Шрифт, число

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана, мультимедиа, Операционная система

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

# **Заключение**

В рамках курсовой работы разработано защищенное кроссплатформенное приложение-календарь. Приложение обеспечивает шифрование данных, защиту паролем, удобный графический и консольный интерфейс. Проект может быть легко расширен, портирован на мобильные платформы через .NET MAUI или Xamarin, и интегрирован с облачными сервисами хранения.

Ссылка на репозиторий: https://github.com/denrus2003/Calendar