МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«Национальный исследовательский технологический университет «МИСиС»**

**(НИТУ «МИСиС»)**

Институт компьютерных наук (ИКН)

Кафедра Инженерной кибернетики

Домашнее задание

По дисциплине «Технологии программирования»

На тему «Разработка приложения “Записная книжка”»

Выполнил: студент 2 курса

гр. БИВТ-22-4 Иванов Никита

Руководитель: Куренкова Т. В.

ОГЛАВЛЕНИЕ

[1. Цель работы3](#_ЦЕЛЬ_РАБОТЫ)

[2. Основная часть4](#_ОСНОВНАЯ_ЧАСТЬ)

[• Техническое задание4](#_1._ТЕХНИЧЕСКОЕ_ЗАДАНИЕ)

[• Описание проекта5](#_2._ОПИСАНИЕ_ПРОЕКТА)

[• Техническое описание6](#_3._ТЕХНИЧЕСКОЕ_ОПИСАНИЕ)

[3. Заключение7](#_ЗАКЛЮЧЕНИЕ)

[4. Список использованных источников8](#_СПИСОК_ИСПОЛЬЗОВАННЫХ_ИСТОЧНИКОВ)

# ЦЕЛЬ РАБОТЫ

Разработать программу "Записная книжка" на языке программирования C# Windows Forms с использованием библиотек NuGet для сериализации и десериализации данных в JSON формат. Основное назначение программы "Записная книжка" - предоставление пользователю возможности сохранять, организовывать и управлять своими записями, заметками, контактами или другой информацией.

# ОСНОВНАЯ ЧАСТЬ

## 1. ТЕХНИЧЕСКОЕ ЗАДАНИЕ

1.1. ВВЕДЕНИЕ

Программа "Записная книжка" является актуальной в современном информационном обществе, где пользователи сталкиваются с необходимостью эффективного управления своими записями, контактами и другой информацией. В условиях повышенной цифровой активности и объема данных, представленных в различных форматах, существует потребность в удобном и надежном инструменте для организации и хранения личной информации. Таким образом, разработка программы "Записная книжка" направлена на обеспечение пользователей возможностью удобного и структурированного хранения своих записей, что делает данную программу востребованной и актуальной.

1.2. ТРЕБОВАНИЯ К ПРОГРАМНОМУ ОБЕСПЕЧЕНИЮ

1.2.1. Совместимость с системой Windows

1.2.2. Реализация проекта на языке C#

1.2.3. Использование библиотек NuGet для сериализации и десериализации данных в JSON формат.

1.3. ФУНКЦИОНАЛЬНЫЕ ВОЗМОЖНОСТИ

1.3.1. В программе должны быть реализованы следующие функции:

* Добавление новой записи.
* Просмотр списка всех записей.
* Поиск записей по заданным критериям (например, по ключевому слову или по дате).
* Редактирование существующей записи.
* Удаление выбранной записи.

1.3.2. Программа должна предоставлять пользователю понятный интерфейс.

1.3.3. Записи должны содержать следующие поля:

* Заголовок (название записи).
* Текст (содержание записи).
* Дата и время создания записи.

1.3.4. Программа должна предоставлять возможность сохранять и загружать записи из текстового файла.

1.4. ДИАГРАММА ВАРИАНТОВ ИСПОЛЬЗОВАНИЯ

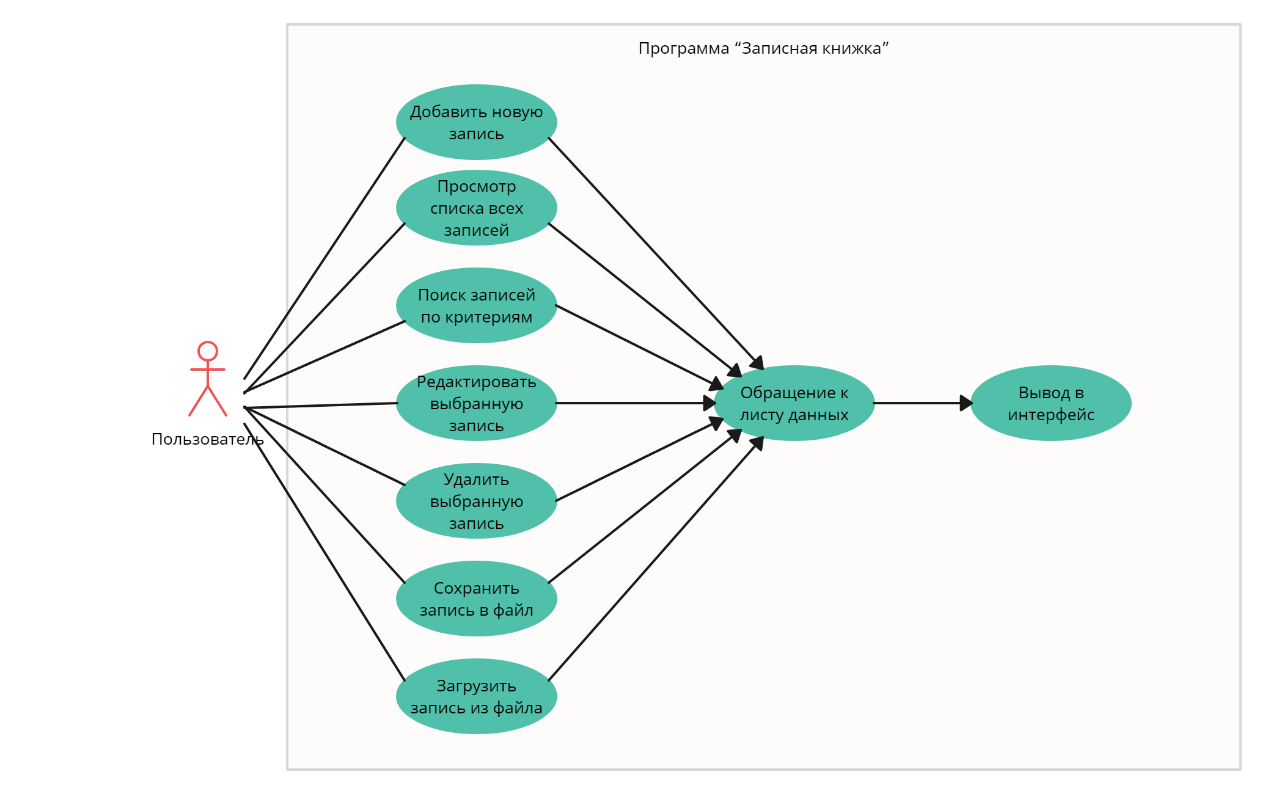


Рис. 1. Диаграмма вариантов использования.

## 2. ОПИСАНИЕ ПРОЕКТА

2.1. СОСТАВ ПРОЕКТА

Форма Windows Forms с названием MainForm. Это основная и единственная форма, с которой взаимодействует пользователь. На ней реализуются следующие элементы интерфейса:

* ListBox «notesListBox». Данный элемент используется для вывода информации из файла JSON формата.
* Label «label1» с надписью «Заголовок».
* Label «label2» с надписью «Содержимое».
* Label «label3» с надписью «Поиск по значению».
* TextBox «titleTextBox». Служит полем для ввода заголовка записи.
* TextBox «contentTextBox». Служит полем для ввода заголовка записи.
* TextBox «searchTextBox». Служит полем для ввода заголовка записи.
* Button «AddNoteButton». Данная кнопка добавляет записанные в «titleTextBox» и «contentTextBox» данные в лист записей.
* Button «EditNoteButton». Данная кнопка изменяет данные выбранной записи в «notesListBox» и заменяет их на записанные в «titleTextBox» и «contentTextBox» данные.
* Button «DeleteNoteButton». Данная кнопка удаляет данные выбранной записи в «notesListBox».
* Button «SearchButton». Данная кнопка производит поиск по записям, беря за фильтр поиска содержимое «searchTextBox». Результат поиска отображается в «notesListBox». Если «searchTextBox» будет пустым – кнопка просто обновит «notesListBox», выведя все доступные записи.
* Button «SaveToFileButton». Данная кнопка открывает диалоговое окно с пользователем, где можно сохранить записи в файл в любом удобном для пользователя расположении.
* Button «LoadFromFileButton». Данная кнопка открывает диалоговое окно с пользователем, где можно загрузить записи из файла.
* Chart «chart1». Это график, на котором наглядно можно увидеть сколько записей было создано в день.

2.2. ИНСТРУКЦИЯ ПО ИСПОЛЬЗОВАНИЮ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

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

Автоматически созданное описание

Рис.2. Окно приложения.

2.2.1. Вариант использования «Добавить запись»:

* Пользователь открывает программу.
* Вводит «Заголовок» будущей записи в указанное для этого поле («titleTextBox»).
* Вводит «Содержимое» будущей записи в указанное для этого поле («contentTextBox»).
* Нажимает кнопку «Добавить запись».
* Новая запись отобразится в поле «notesListBox».

2.2.2. Вариант использования «Редактировать запись»:

* Пользователь открывает программу.
* Выбирает запись, которую хочет изменить, в поле «notesListBox».
* Вводит «Заголовок» будущей записи в указанное для этого поле («titleTextBox»).
* Вводит «Содержимое» будущей записи в указанное для этого поле («contentTextBox»).
* Нажимает кнопку «Редактировать запись».
* Измененная запись отобразится в поле «notesListBox».

2.2.3. Вариант использования «Удалить запись»:

* Пользователь открывает программу.
* Выбирает запись, которую хочет удалить, в поле «notesListBox».
* Нажимает кнопку «Удалить запись».
* В поле «notesListBox» отобразится отсутствие выделенной ранее записи.

2.2.4. Вариант использования «Сохранить записи в файл»:

* Пользователь открывает программу.
* Нажимает кнопку «Сохранить в файл».
* Выбирает путь, куда сохранить файл.
* Выбирает название будущего файла.
* Нажимает кнопку «Сохранить».
* Файл с записями отобразится в директории, которую ранее выбрал пользователь.

2.2.5. Вариант использования «Загрузить записи из файла»:

* Пользователь открывает программу.
* Нажимает кнопку «Загрузить из файла».
* Выбирает путь, где расположен файл с записями в формате JSON.
* Выбирает файл с записями в формате JSON.
* Нажимает кнопку «Открыть».
* Записи из файла отобразятся в поле «notesListBox».

2.2.6. Вариант использования «Поиск записи по критерию»:

* Пользователь открывает программу.
* Вводит критерии для поиска в указанное для этого поле («searchTextBox»).
* Нажимает кнопку «Поиск».
* Отфильтрованные по критериям записи отобразятся в поле «notesListBox».

## 3. ТЕХНИЧЕСКОЕ ОПИСАНИЕ

3.1. СТРУКТУРА ПРОЕКТА

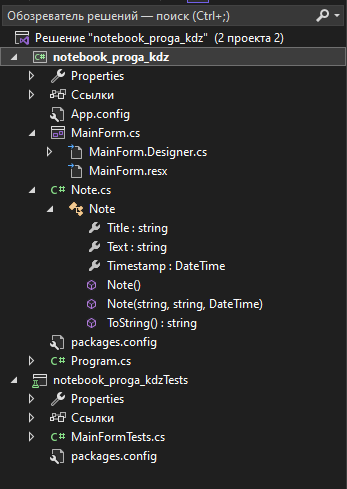


Рис.3. Структура проекта.

Данный проект включает в себя форму MainForm, класс Note и тесты MainFormTests. Также в проекте используется сторонняя библиотека для работы с JSON файлами Newtonsoft.Json.

3.2. ОПИСАНИЕ ДЕТАЛЕЙ РЕАЛИЗАЦИИ

Для взаимодействия с данными в программе реализован класс Note. Поля класса: Title, Text, Timestamp. В классе также реализовано переопределение метода ToString() в формат «Заголовок - Время создания - Содержание».

Основной элемент пользовательского интерфейса является объект Windows Forms «MainForm». На этой форме реализуются все элементы управления и некоторый функционал программы.

Методы MainForm для работы с данными: RefreshNotesList(), ClearFields(), LoadNotesFromFile(), SaveNotesToFile(), UpdateChartData(), AddNoteButton\_Click(), EditNoteButton\_Click(), DeleteNoteButton\_Click(), SearchButton\_Click(), SaveToFileButton\_Click(), LoadFromFileButton\_Click().

Метод RefreshNotesList() отвечает за обновление/заполнение данными в элементе интерфейса ListBox «notesListBox». Он вызывается каждый раз, когда добавляются/изменяются/удаляются данные.

Метод ClearFields() отвечает за очистку данных в элементе интерфейса ListBox «notesListBox». Он вызывается каждый раз, когда добавляются/изменяются/удаляются данные.

Метод LoadNotesFromFile() отвечает за загрузку и десериализацию данных из файла формата JSON в папке проекта. Вызывается каждый раз при заходе в приложение.

Метод SaveNotesToFile() отвечает за сохранение и сериализацию данных в файл формата JSON в папке проекта. Он вызывается каждый раз, когда добавляются/изменяются/удаляются данные.

Метод UpdateChartData() отвечает за обновление данных в элементе интерфейса Chart «chart1». Он вызывается каждый раз, когда добавляются/изменяются/удаляются данные.

Метод AddNoteButton\_Click() отвечает за добавление новой записи в List<note> notes. Вызывается при клике мышью по кнопке «Добавить запись».

Метод EditNoteButton\_Click() отвечает за редактирование старой записи в List<note> notes. Вызывается при клике мышью по кнопке «Редактировать запись».

Метод DeleteNoteButton\_Click() отвечает за удаление старой записи в List<note> notes. Вызывается при клике мышью по кнопке «Удалить запись».

Метод SearchButton\_Click() отвечает за поиск записи по заданному в TextBox «searchTextBox» критерию в List<note> notes. Вызывается при клике мышью по кнопке «Поиск».

Метод SaveToFileButton\_Click() отвечает за открытие диалогового окна с сохранением и сериализацией данных в JSON файл. Вызывается при клике мышью по кнопке «Сохранить в файл».

Метод LoadFromFileButton\_Click() отвечает за открытие диалогового окна с загрузкой и десериализацией данных из JSON файла. Вызывается при клике мышью по кнопке «Загрузить из файла».

Полный код MainForm:

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Windows.Forms;

using Newtonsoft.Json;

namespace notebook\_proga\_kdz

{

public partial class MainForm : Form

{

private List<Note> notes;

public MainForm()

{

InitializeComponent();

notes = new List<Note>();

}

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

{

LoadNotesFromFile();

RefreshNotesList();

UpdateChartData();

}

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

{

string title = titleTextBox.Text;

string text = contentTextBox.Text;

DateTime timestamp = DateTime.Now;

Note newNote = new Note(title, text, timestamp);

notes.Add(newNote);

ClearFields();

RefreshNotesList();

UpdateChartData();

SaveNotesToFile();

}

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

{

if (notesListBox.SelectedItem != null)

{

Note selectedNote = (Note)notesListBox.SelectedItem;

string newTitle = titleTextBox.Text;

string newText = contentTextBox.Text;

selectedNote.Title = newTitle;

selectedNote.Text = newText;

ClearFields();

RefreshNotesList();

UpdateChartData();

SaveNotesToFile();

}

}

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

{

if (notesListBox.SelectedItem != null)

{

Note selectedNote = (Note)notesListBox.SelectedItem;

notes.Remove(selectedNote);

ClearFields();

RefreshNotesList();

UpdateChartData();

SaveNotesToFile();

}

}

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

{

string keyword = searchTextBox.Text.ToLower();

IEnumerable<Note> searchResults = notes.Where(note =>

note.Title.ToLower().Contains(keyword) ||

note.Text.ToLower().Contains(keyword) ||

note.Timestamp.ToString().Contains(keyword));

RefreshNotesList(searchResults);

}

private void RefreshNotesList(IEnumerable<Note> noteList = null)

{

notesListBox.Items.Clear();

IEnumerable<Note> displayNotes = noteList ?? notes;

foreach (Note note in displayNotes)

{

notesListBox.Items.Add(note);

}

}

private void ClearFields()

{

titleTextBox.Text = "";

contentTextBox.Text = "";

}

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

{

using (SaveFileDialog saveFileDialog = new SaveFileDialog())

{

saveFileDialog.Filter = "JSON Files (\*.json)|\*.json";

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

{

string json = JsonConvert.SerializeObject(notes);

File.WriteAllText(saveFileDialog.FileName, json);

MessageBox.Show("Записи сохранены в файл.");

}

}

}

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

{

using (OpenFileDialog openFileDialog = new OpenFileDialog())

{

openFileDialog.Filter = "JSON Files (\*.json)|\*.json";

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

{

string json = File.ReadAllText(openFileDialog.FileName);

notes = JsonConvert.DeserializeObject<List<Note>>(json);

RefreshNotesList();

UpdateChartData();

MessageBox.Show("Записи загружены из файла.");

}

}

}

private void LoadNotesFromFile()

{

string projectPath = Path.GetDirectoryName(Application.ExecutablePath);

string filePath = Path.Combine(projectPath, "notes.json");

if (File.Exists(filePath))

{

string json = File.ReadAllText(filePath);

notes = JsonConvert.DeserializeObject<List<Note>>(json);

}

}

private void SaveNotesToFile()

{

string projectPath = Path.GetDirectoryName(Application.ExecutablePath);

string filePath = Path.Combine(projectPath, "notes.json");

if (File.Exists(filePath))

{

string json = JsonConvert.SerializeObject(notes);

File.WriteAllText(filePath, json);

}

else

{

var stream = File.Create(filePath);

stream.Close();

string json = JsonConvert.SerializeObject(notes);

File.WriteAllText(filePath, json);

}

}

private void UpdateChartData()

{

var data = notes.GroupBy(note => note.Timestamp.Date).Select(group => new { Date = group.Key, Count = group.Count() }).OrderBy(item => item.Date);

var xData = data.Select(item => item.Count).ToArray();

var yData = data.Select(item => item.Date).ToArray();

chart1.Series[0].Points.DataBindXY(yData, xData);

}

}

}

3.3. ДИАГРАММА КЛАССОВ

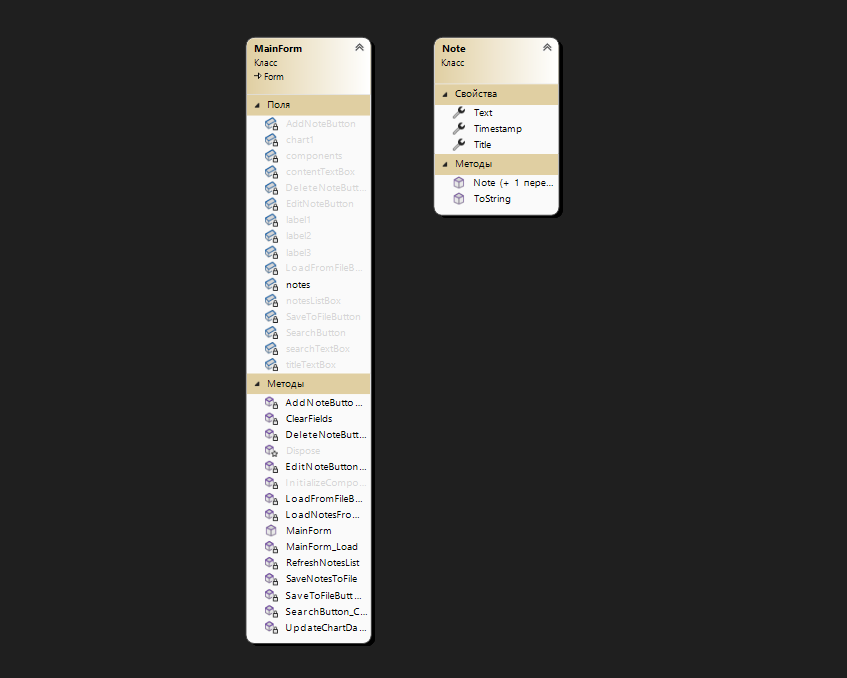


Рис.4. Диаграмма классов.

3.4. ТЕСТИРОВАНИЕ

Для тестирования программы реализуем тестовые методы: AddNoteTest() и DeleteNoteTest().

Тестовый метод AddNoteTest() проверяет добавление новой строки в List<Note> notes. На вход подаётся сам лист данных, новая запись и переменная для подсчета записей. Далее добавляется новая запись в лист данных и после метод сравнивает переменную для подсчета записей, увеличенную на 1, и количество записей в List<Note> notes.

Тестовый метод DeleteNoteTest() проверяет удаление существующей строки в List<Note> notes. На вход подаётся сам лист данных, новая запись и сразу добавляется в лист данных. Также на вход подаётся переменная для подсчета записей. Далее удаляется запись из листа данных, и после метод сравнивает переменную для подсчета записей, уменьшенную на 1, и количество записей в List<Note> notes.

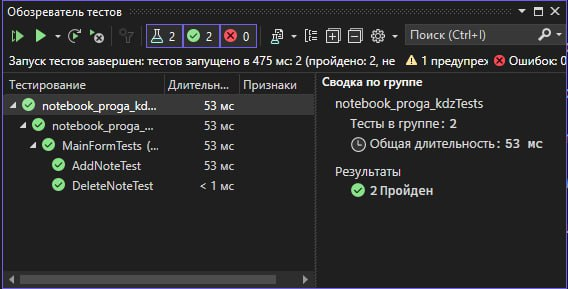


Рис.5. Проверка тестов.

3.5. СТОРОННИЕ БИБИЛИОТЕКИ, ИСПОЛЬЗОВАННЫЕ В РАБОТЕ

В данном проекте используется сторонняя библиотека Newtonsoft.Json полученная из диспетчера пакетов NuGet.

Библиотека Newtonsoft.Json, также известная как JSON.NET, является одной из самых популярных библиотек для работы с форматом JSON в платформе .NET. Она предоставляет мощные инструменты для сериализации и десериализации, а также для удобного чтения и записи JSON-данных.

Newtonsoft.Json поддерживает различные возможности, включая:

* Гибкую настройку процесса сериализации и десериализации с помощью атрибутов или конфигурационных параметров.
* Поддержку различных типов данных, включая пользовательские типы, коллекции, словари и др.
* Работу с JSON в различных сценариях, таких как веб-запросы, хранение настроек, обмен данными между приложениями и многое другое.
* Поддержку LINQ to JSON для удобной навигации и манипуляции с JSON-структурами.

Благодаря своей гибкости и производительности, Newtonsoft.Json широко используется для работы с JSON в приложениях на платформе .NET.

# ЗАКЛЮЧЕНИЕ

В ходе разработки программы "Записная книжка" была успешно достигнута поставленная цель, которая заключалась в создании инструмента для удобного управления и хранения личной информации. Использование языка программирования C# Windows Forms с библиотеками NuGet для сериализации и десериализации данных в JSON формат позволило создать эффективное приложение, способное удовлетворить потребности пользователей в организации записей, заметок и контактов.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

* Документация Windows Forms: <https://learn.microsoft.com/ru-ru/dotnet/desktop/winforms/?view=netdesktop-8.0>
* Документация Newtonsoft.Json: <https://www.newtonsoft.com/json/help/html/SerializingJSON.htm>
* Лекции по Технологиям программирования Куренковой Т. В.