Міністерство освіти і науки України

Запорізький національний технічний університет

кафедра програмних засобів

ЗВІТ

З лабораторної роботи №5

З дисципліни «Якість програмного забезпечення та тестування» з теми:

**«ПРОВЕДЕННЯ UNIT-ТЕСТІВ»**

Виконав:

студент групи КНТ-115 Ю.О. Лукашенко

Прийняла:

доцент, к.т.н Н.О.Миронова

2018

1. Мета роботи

Вивчити можливості спеціального модулю для unit-тестів та реалізувати тести із застосуванням мови С# та Visual Studio.

1. Хід роботи

У якості проекту, для якого проводиться тестування, було взято проект SunNotes – програму для управління нотатками.

Основний фунціонал програми SunNotes представляє собою:

1. Надання інструментарію для формування нотаток – користувач може задавати для них текст, назву, теги, колір, прив’язувати дату.
2. Додавання, редагування, видалення нотаток. Реалізація даних вимог передбачає наявності бази даних.
3. Виконання пошуку нотаток за датою та за порядковим номером
4. Надання можливості архівації нотаток та відновлення їх з архіву.

Інтерфейс програми зображений на рис. 1.1-1.3

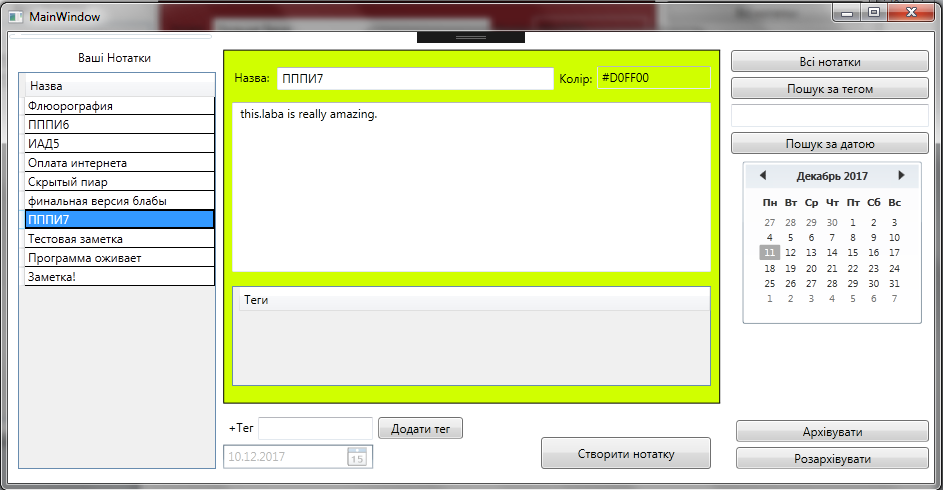


Рисунок 1.1 – Інтерфейс головного вікна програми

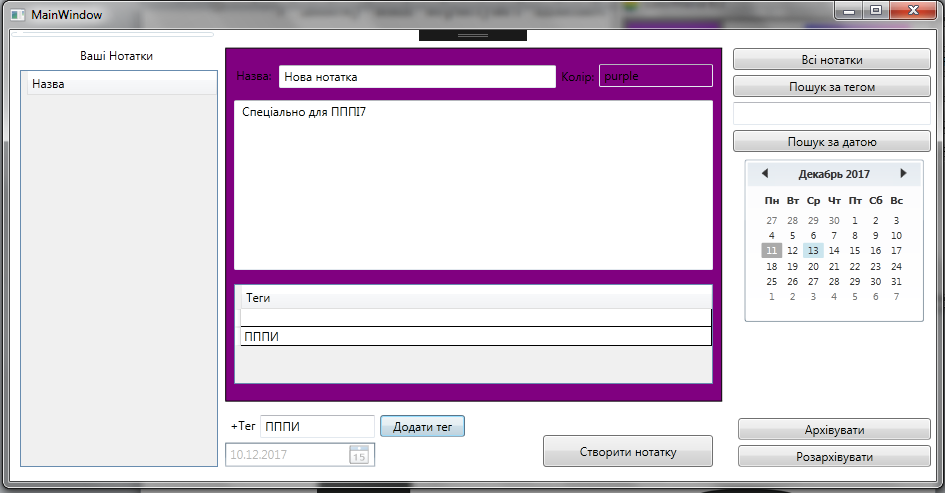


Рисунок 1.2 – Створення нової нотатки

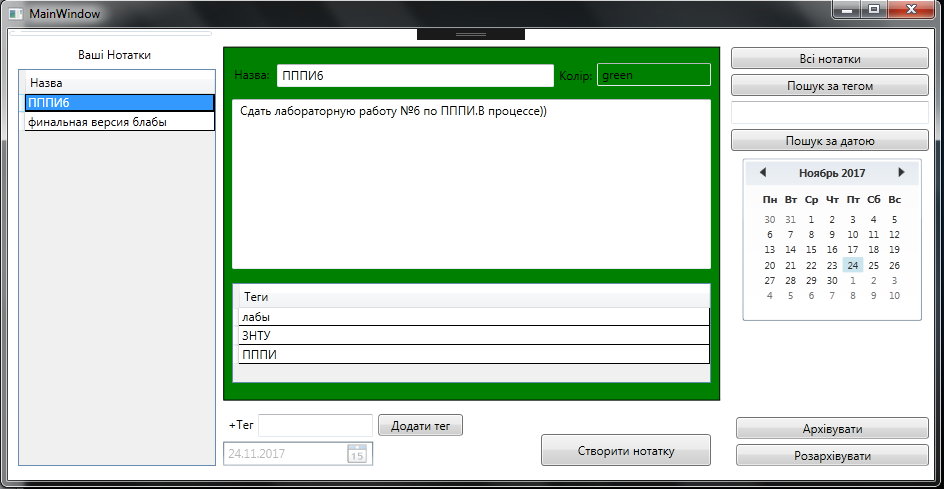


Рисунок 1.3 – Пошук за датою

1. Тестування функцій програми та бази даних за допомогою модульних тестів

Для реалізації модульного тестування було створено додатковий проект

SunNotes.Test, в якому були створенні тести для перевірки тих чи інших функцій. Кожна функція була перевірена як на позитивний та негативний сценарії.

### 5.3.1 Лістинг тестів бізнес-логіки

using System;

using System.Collections.Generic;

using ConsoleApp1;

using Microsoft.VisualStudio.TestTools.UnitTesting;

namespace SunNotes.Test

{

[TestClass]

public class UnitTests

{

[TestMethod]

public void Test\_IsParametersNote\_Null\_True()

{

//налаштування

Note test = new Note();

test.Num = 1;

test.Title = null;

test.Text = "Текст нотатки. Більше одного речення. А може двох. Навіть трьох.";

test.Tags = "Тег1 тег2";

test.Color = "FFAAFF";

test.Datatime = DateTime.Now;

bool expected = true;

//дії

User us = new User();

bool actual = us.IsParametersNoteNull(test);

//перевірка

Assert.AreEqual(expected, actual);

}

[TestMethod]

public void Test\_IsParametersNote\_Null\_False()

{

//налаштування

Note test = new Note();

test.Num = 2;

test.Title = "Назва нотатки";

test.Text = "Текст нотатки. Більше одного речення. А може двох. Навіть трьох.";

test.Tags = "Тег1 тег2";

test.Color = "FFAAFF";

test.Datatime = DateTime.Now;

bool expected = false;

//дії

User us = new User();

bool actual = us.IsParametersNoteNull(test);

//перевірка

Assert.AreEqual(expected, actual);

}

[TestMethod]

public void Test\_FindDate\_True()

{

//налаштування

List<Note> lst = new List<Note>();

Note test1 = new Note();

test1.Datatime = DateTime.Today;

lst.Add(test1);

Note test2 = new Note();

test2.Datatime = DateTime.MaxValue;

lst.Add(test2);

bool expected = true;

//дії

User us = new User(lst);

bool actual = us.FindDate(DateTime.Now);

//перевірка

Assert.AreEqual(expected, actual);

}

[TestMethod]

public void Test\_FindDate\_False()

{

//налаштування

List<Note> lst = new List<Note>();

Note test1 = new Note();

test1.Datatime = DateTime.MinValue;

lst.Add(test1);

Note test2 = new Note();

test2.Datatime = DateTime.MaxValue;

lst.Add(test2);

bool expected = false;

//дії

User us = new User(lst);

bool actual = us.FindDate(DateTime.Now);

//перевірка

Assert.AreEqual(expected, actual);

}

}

}

Всі тести були корректно виконані, про що свідчить рис. 1.4.

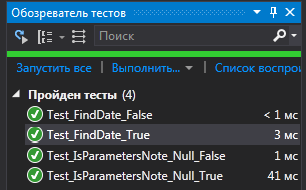
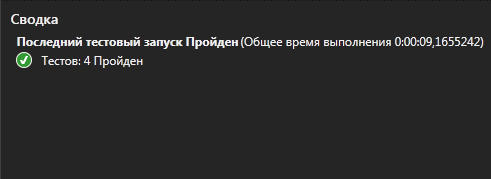
 

Рисунок 1.4 – Модульні тести бізнес-логіки

Для тестування бази даних засобами платформи .Net було використано SQL Server Unit Test. Дані тести мають за основу тестування запитів до бази даних, за допомогою T-SQL транзакцій. За допомогою конструкції:

BEGIN TRANSACTION TestTransaction

-----------SOME SQL TEST CODE-----

ROLLBACK TRANSACTION TestTransaction

дозволяє тестувати базу даних без якихось змін для самої бази. Тобто при запиті на додавання даних відбувається вставка, потім перевірка, а потом команда ROLLBACK повертає базу до початкового стану, чим «відкочує» проведені зміни.

### 5.3.2 Лістинг тестів SQL запитів до бази даних

// BaseTestAddNote

BEGIN TRANSACTION TestTransaction

INSERT INTO [Notte] VALUES ('Заметка тест', 'Текст тестовой заметки', 'Больше тегов', 'FFFFFF', getdate());

SELECT \* FROM [Notte] WHERE [Notte].[Title] = 'Заметка тест';

ROLLBACK TRANSACTION TestTransaction

// BaseTestExtendedAdd

BEGIN TRANSACTION TestTransaction

DECLARE @Id AS INT, @Title AS NCHAR (32),

@Text AS NVARCHAR(MAX), @Tags AS NCHAR (32), @Color AS NCHAR (10),

@Date AS DATE;

SELECT @Title = 'Заметка тест',

@Text = 'Текст тестовой заметки',

@Tags = 'Больше тегов',

@Color = 'FFFFFF',

@Date = getdate();

INSERT INTO [Notte] ([Notte].Title, [Notte].[Text], [Notte].Tags, [Notte].[Color], [Notte].[Date])

VALUES (@Title, @Text, @Tags, @Color,@Date);

ROLLBACK TRANSACTION TestTransaction

// BaseTestFindNoteByDate

DECLARE @Id AS INT, @Date AS Date;

SELECT @Id = 1,

@Date = getdate();

SELECT \* FROM [Notte] WHERE [Notte].Date = @Date;

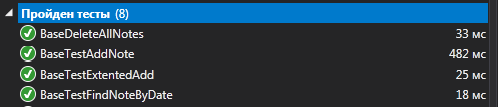
// BaseDeleteAllNotes

BEGIN TRANSACTION TestTransaction

TRUNCATE TABLE [Notte];

ROLLBACK TRANSACTION TestTransaction

Всі тести були корректно виконані, про що свідчить рис. 1.5.



## Рисунок 1.5 – Модульні тести бази даних

1. Висновки

В процесі роботи було виконано модульне тестування, яке дозволяє зменшити кількість ручних перевірок деяких частин коду шляхом автоматизації та створення Unit Test-ів, що в свою чергу прискорює розробку програмного продукту. Тестування було виконано у двох видах, а саме тестування бізнес-логіки програми, програмного коду та тестування запитів до бази даних.