Міністерство освіти і науки України

Чернівецький національний університет

імені Юрія Федьковича

Інститут фізико-технічних та комп'ютерних наук
(повна назва інституту/факультету)

Кафедра математичних проблем управління і кібернетики
(повна назва кафедри)

Уніфікована розробка під Xamarin.NET на прикладі створення ПЗ «Записник»

Дипломний проєкт

Рівень вищої освіти – перший (бакалаврський)

Виконав:
студент 4 курсу, групи 441
спеціальності
122 – Комп'ютерні науки та інформаційні технології
(шифр і назва спеціальності)
Бужак А.В.
(прізвище та ініціали)
Керівник к.ф.-м.н., доцент Лазорик В.В.
(науковий ступінь, вчене звання прізвище та ініціали)

До захисту допущено.

Протокол засідання кафедри № _____

від "____" _____ 2020 р.

Зав. кафедри_____ д. ф-м.н. **Дрінь Я.М.**

Чернівці–2020

АНОТАЦІЯ

У даній роботі проводились дослідження пов’язані з кроплатформною розробкою застосунків на платмормі Xamarin.NET.

На основі проведених досліджень був розроблений програмний продукт, центральним функціональним призначенням якого стало керування та обмін колекціями даних типу «Нотатка».

Розроблений додаток для розв'язання поставленої задачі по керуванню нотатками, в результаті випробувань показав свою повну працездатність. Відповідає всім функціональним вимогам, вимогам до інтерфейсу, а також вимогам по структурі та загальній архітектурі проекту.

Програмний продукт було реалізовано засобами середовища Microsoft Visual Studio 2017, мовою C# на платформі Xamarin та з використанням інстурментів Entity Framework та NUnit для трьох операційних систем: Windows, Android та iOS.

***Ключові слова**: Xamarin; кросплатформа.*

ANNOTATION

In this work, conducted research related to cross-platform application development on the Xamarin.NET platform.

Based on the research, a software product was developed, the central functional purpose of which was the management and exchange of data collections such as "Note".

Developed an application to solve the problem of notes management, as a result of tests showed its full efficiency. Meets all functional requirements, interface requirements, as well as requirements for the structure and general architecture of the project.

The software was implemented using Microsoft Visual Studio 2017, C# language on the Xamarin platform and using the Entity Framework and NUnit tools for three operating systems: Windows, Android and iOS.

Keywords: *Xamarin; cross-platform.*

ЗМІСТ

ТЕХНІЧНА ЧАСТИНА

1. ТЕХНІЧНЕ ЗАВДАННЯ

1.1. Підстави для розробки

Чимало програмних продуктів сьогодні потребують одночасного випуску рішення для декількох поширених операційних систем, як то: Windows, Android чи iOS. Оскільки, така ідея не лише задовольняє більшу частину користувачів, а й пришвидшує перебіг поширення продукту – це зумовлює формування інфраструктури цього програмного забезпечення, а інфраструктура, в свою чергу – запорука успішності проекту [1, 250 с.].

Зазвичай, великі компанії надають перевагу делегуванню розробки проекту під окрему платформу – окремій команді. Відповідно – вони будуть дотримуватися однієї ідеології, проте підходи, методи й засоби реалізації можуть суттєво відрізнятися, що в подальшому призводить до ускладнення підтримки й розробки конкретної програми [2]. Та й менеджмент дотримання цієї єдиної ідеології проекту для розподілених платформ також призводить до збільшення витрат грошових ресурсів.

Суттєво відрізняється принцип уніфікованої розробки, коли програмний продукт розробляється один раз але працює на різних платформах. Причому, можливості функціонування програмного забезбечення не повинні обмежуватися модулями що можуть працювати однакового на різних платформах, натомість має бути можливість перевизначення окремих модулів, якщо особливість їх роботи потребує певної специфіки при виконанні на конкретній платформі. Огляд одного з таких підходів уніфікованої розробки, а саме платформи Xamarin, і представлено в даній роботі на прикладі створення програми «Записник».

Розробка програмного забезпечення виконується на підставі рішення засідання кафедри МПУіК про затвердження тем курсових робіт та дипломних проєктів (протокол № *2* від “*06*” *вересня* 20*19* року).

1.2. Призначення розробки

Програмне забезпечення «Записник» може використовуватись як передовий інструмент керування інформативними даними на локальному пристрої, або в мережі. Рівні мережевого поширення роботи програми (глобальна мережа Інтернет чи корпоративна мережа) – залежать від під'єднання до конкретної бази даних, що у зручний спосіб доступно в панелі налаштувань програмного рішення.

1.3. Аргументація теми та її актуальність

Чи не вся інформація сучасного світу потребує певного аналізу та обробки. Досить зручно постійно мати під рукою можливість занотувати ту чи іншу інформацію, ідею або ж вибірку корисних даних а згодом у зручний спосіб переглянути їх та відредагувати. Дані для запису можуть бути різних видів: текст, веб-посилання, зображення, аудіо, відео або файл. Чимало програм існує для нотування даних лише одного із перелічених видів, або лише кількох із них. Натомість, ПЗ «Записник» об'єднує роботу всіх цих видів даних в один гнучкий інструмент, до того ж надає можливість керованої синхронізації та роботи на трьох ОС: Windows, Android та iOS. Особливість обраної платформи Xamarin забезпечує простоту підтримки і розгортання оновлень на всіх трьох операційних системах.

2. ВИМОГИ ДО РОБОТИ

2.1. Стадії та етапи розробки

Таблиця 2.1.1.

№ п/п	Назва етапів дипломного проєкту	Термін виконання етапів
1	Одержання технічного завдання	16.09.19 р.
2	Аналіз поставленого завдання	14.10.19 р.
3	Ознайомлення з літературою	31.12.19 р.
4	Розробка загальної концепції ПЗ	20.01.20 р.
5	Розробка основного алгоритму	31.01.20 р.
6	Реалізація функціональної частини	29.02.20 р.
7	Реалізація інтерфейсу	16.03.20 р.
8	Реалізація допоміжного забезпечення	31.03.20 р.
9	Тестування та налагодження програми	30.04.20 р.
10	Оформлення програмної документації	31.05.20 р.
11	Представлення готового проєкту	01.06.20 р.
12	Захист дипломного проєкту	згідно з розкладом

2.2. Аналіз вимог до програмного забезпечення

2.2.1. Функціональні вимоги

До програмного забезпечення висуваються наступні вимоги:

- можливість вносити, змінювати і поширювати дані формату «Нотатка», такі що складаються із частин – напередвизначених типів;

- напередвизначені типи частин нотатки мають являти собою представлення (зрозуміле для людини, і прийнятне для поширення, збереження і редагування) таких типів даних: текст, веб-посилання, файл (в т.ч. зображення, аудіо і відео);

- можливість внесення голосового запису як частини нотатки типу аудіо;

- можливість внесення відео запису як частини нотатки типу відео;

- можливість групування нотаток за тегами;

Програмний продукт повинен працювати на трьох операційних системах – Windows, Android та iOS, і мати єдиний візуальний інтерфейс користувача.

2.2.2. Вимоги до складу та параметрів технічних засобів

Мінімальні вимоги до апаратного забезпечення пристрою для коректної роботи розробленого програмного продукту:

Таблиця 2.2.2.1

ОС	Мінімальна версія
Windows	8.1
Android	7.0
iOS	11.2

Необхідною вимогою для коректної роботи розробленого програмного забезпечення на Windows є підтримка UWP (Universal Windows Platform). Для запуску версії win-32 необхідна наявність встановленого ".NET Framework 4.7 Runtime".

2.2.3. Вимоги до вхідних та вихідних даних

Вхідними даними для розробленого програмного продукту є дані типу «Нотатка» внесені шлях візуального інтерфейсу користувача, або завантажені із файлу JSON відповідної структури (описаної в коді програми).

Вихідними даними є повний варіант даних експортований у форматі JSON або ж інші варіанти збереження (PDF, TXT, Word, Excel, аудіо, відео, файли, картинки) із врахуванням неможливості повної передачі інформації для деяких випадків, наприклад – при варіанті експорту картинками або TXT, вихідні дані не будуть містити прикріплених аудіо чи відео файлів і т.д.

2.2.4. Вимоги до інтерфейсу

Інтерфейс програмного продукту повинен чітко розподіляти групи для керування даними: керування завантаженням, керування збереженням, представлення та редагування. Розміщення, відображення або підпис візуальних елементів керування не має суперечити логічній приналежності їх поля застосування [3].

2.2.5. Вимоги до тестування програмного забезпечення

Для тестування програмного забезпечення необхідно виконати наступні дії:

1. Запустити ПЗ «Записник».
2. Шляхом виконання «Файл => Завантажити з => JSON» завантажити json-файл, що відповідає структурі представлення даних, описаній у коді програми.

Після успішного завантаження у головній панелі програми повинен оновитися список «Нотаток» або «Питань-Відповідей» відповідно до вмісту завантажуваного файлу і встановлених налаштувань у панелі «Налаштування».

2.3. Вимоги до програмної документації

Програмне забезпечення постачається разом із супроводжувальною документацією, до складу якої входить:

1. Технічне завдання.
2. Особливості використання.
3. Функціональна специфікація.
4. Технічна специфікація.
5. Опис особливостей роботи розгалужених модулів (специфічних для конкретної ОС).

ВИСНОВКИ ДО ТЕХНІЧНОЇ ЧАСТИНИ

Під час виконання технічної частини курсової роботи мною було розроблено базову структуру модулів програмного забезпечення «Записник», визначено цільові функціональні можливості програми, спроектовано формат вхідних та вихідних даних.

Визначена необхідність вибору відкритого інструменту для роботи з базами даних, для поширення вмісту програми.

ТЕОРЕТИЧНА ЧАСТИНА

3. ОГЛЯД ТЕМИ, СУЧАСНІ ПІДХОДИ ДО РЕАЛІЗАЦІЇ, РОЗШИРЕНА ПОСТАНОВКА ЗАДАЧІ

3.1. Призначення та область застосування

Розроблений програмний продукт призначений для зручного зберігання і використання інформації, дозволяє керувати даними виду «Нотатка» чи «Питання-Відповідь» у зручній формі. Може бути задіяний у домашньому використання та корпоративному, завдяки функції синхронізації, що легко налаштовується. Розробнику програмного забезпечення часто зручно мати під рукою такий інструмент як «Записник», для швидкого пошуку занотованих набутих знань.

3.2. Постановка задачі, огляд літератури, підходи до розв'язування

Програмний продукт «Записник» необхідно розробити одразу для декількох операційних систем – Windows, Android, iOS.

Якщо використати нативні підходи для розробки під кожну ОС, наприклад Java Kotlin для Android [5], та Swift для iOS [6], тоді ресурси небхідні для розробки різко зростають, а задача узагальнення спільної логіки функціонування сильно ускладнюється.

Отож, це задача із класу кросплатформної розробки. Подібні задачі досліджувались таким автором як Чарльз Пітцольд [1]. Зокрема, найперспективнішою платформою для цього він вважає Xamarin [1, с.135].

Розроблена програма «Записник» створена на платформі Xamarin, та використовує всі її засоби для побудови максимально уніфікованої архітектури. Але, при цьому, платформа Xamarin дозволяє перевизначати окремі межі модулів, які вимагаю особливої поведінки залежно від ОС, що їх виконує. Частини візуального інтерфейсу користувача можуть бути позиційно розподілені відповідно до платформи завдяки дотриманню принципів MVVM [4]. Це буде корисно у розроблюваній програмі, адже

можна визначити відмінний візуальний інтерфейс для компілювання на win32, а узагальнений залишити для UWP, Android та iOS.

3.3. Опис основних алгоритмів

Розроблене програмне забезпечення сконструйовано з використанням шаблону проектування MVVM [4]. Розподілення на представлення (View), бізнес-логіку (Model) та контролер (ViewModel) для їх зв'язування – дозволяє зручним чином відокремити компоненти, та уникнути великої кількості дублювання коду, або складної ієрархії класів. Завдяки такому розподілу представлення для win32 та мобільних систем може бути зроблено окремими класами, обидва з яких будуть підходити під єдиний візуальний інтерфейс, котрим керуватиме контролер. Бізнес-логіка являє собою окремі класи із віртуальними методами для перевизначення специфічних місць, а ці класи, в свою чергу, успадковують базовий, що зберігає поведінку яка є однаковою для усіх платформ.

3.4. Опис інструментальних засобів

Розробка програмного продукту здійснювалась на персональному комп'ютері наступної конфігурації:

1. Процесор – Intel Core i5 2520M
2. ОЗП – 2x GoodRam 4GB DDR3 1333MHz
3. Відеоадаптери:

- інтегрований – Intel HD Graphics 3000.

4. Накопичувач – GoodRam SSD CX300 240 GB.

Комп'ютер працює під управлінням операційної системи Windows 10 Enterprise LTSC 1809 build 17763.1217 RU x64.

3.5. Загальна структура розробленого програмного продукту

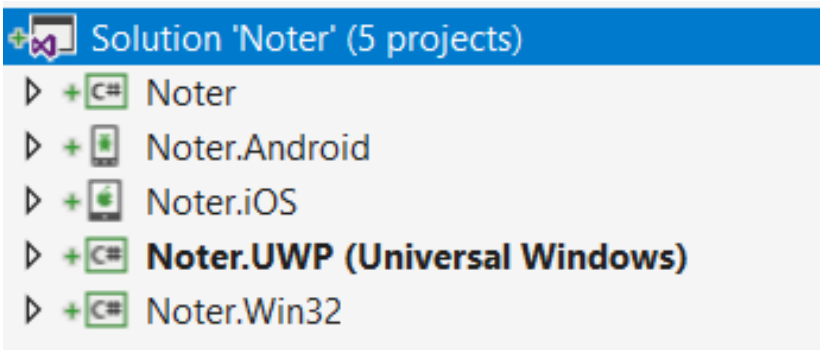

Рис. 3.5.1. Загальна структура рішення

Розроблена програмна система має наступну структуру:

- збірка «Noter» – реалізовує спільну бізнес-логіку, ресурси, представлення та базову архітектуру;
- збірка «Noter.Android» – реалізовує специфіку бізнес-логіки для ОС Android;
- збірка «Noter.iOS» – реалізовує специфіку бізнес-логіки для iOS;
- збірка «Noter.UWP» – реалізовує специфіку бізнес-логіки для ОС Windows на платформі UWP;
- збірка «Noter.Win32» – реалізовує специфіку бізнес-логіки та представлення для ОС Windows на платформі win32 (тип проекту – WPF).

ВИСНОВКИ ДО ТЕОРЕТИЧНОЇ ЧАСТИНИ

Під час роботи над теоретичною частиною мною було розроблено на базі платформи Xamarin.NET комплекс єдиних програмних застосунків «Записник» окремо для трьох операційних систем (Windows, Android та iOS), із єдиною програмною бібліотекою функціональної логіки, та єдиним джерелом візуального інтерфейсу користувача.

Модулі, що потребували особливого виконання на конкретній платформі, були виділені для окремого перевизначення із дотриманням принципів об’ктно орієнтованого програмування та зі збереженням спільної логіки модуля без дублювання коду. Архітектура ієрархії класів була створена згідно шаблону проектування MVVM [4].

Архітектура функціонування, розроблена у технічній частині, не змінила свого вмісту, лише тепер її суть виконує окрема збірка програми.

ПРАКТИЧНА ЧАСТИНА

4. КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1. Загальна структура апаратно-програмного комплексу

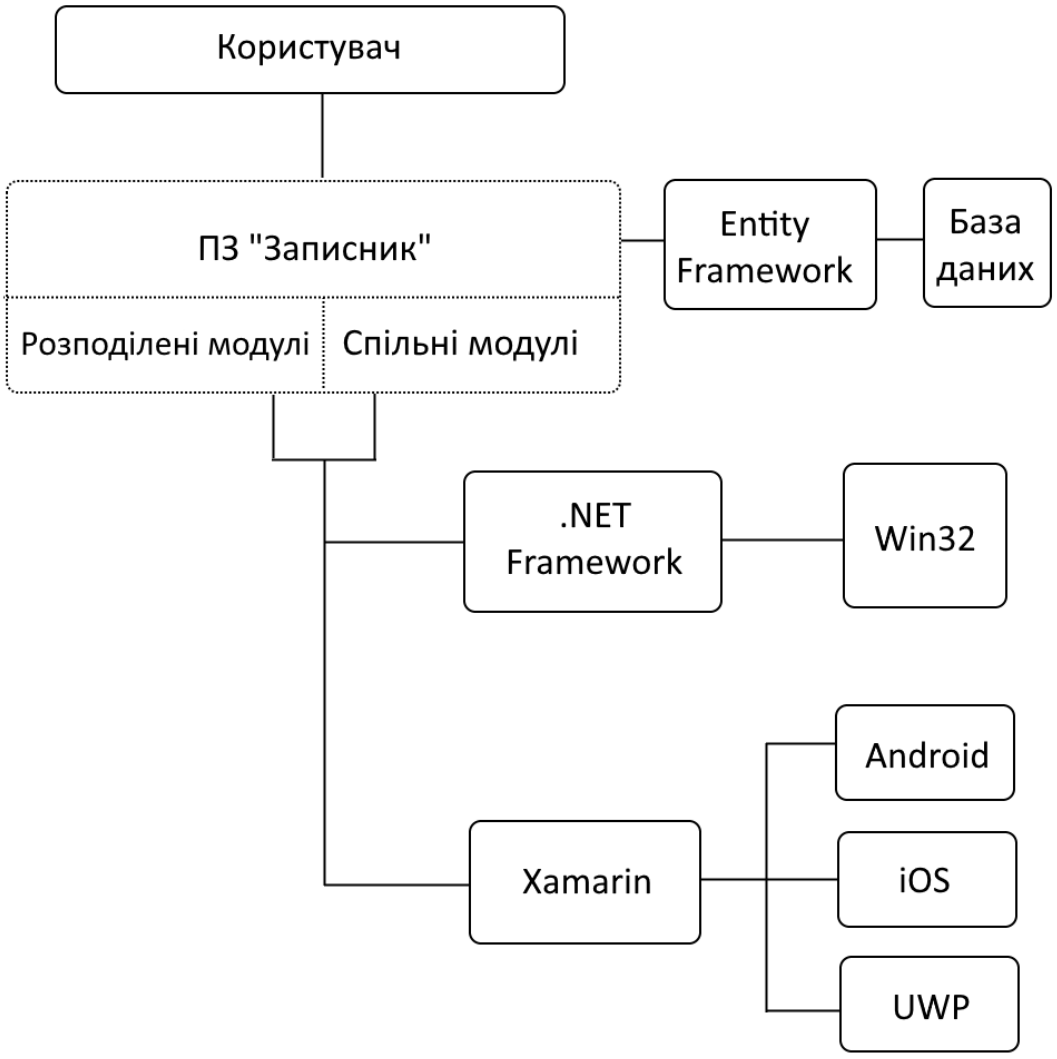

Рис. 4.1.1. Загальна схема роботи програми

Для розв’язання задачі розроблено кросплатформне програмне забезпечення мовою програмування C# у середовищі розробки Microsoft Visual Studio 2017. Мобільні застосунки розроблені із засобами платформи Xamarin. Застосунок для платформи Win32 використовує ті ж ресурси, однак працює на під .NET Framework.

Взаємодія користувача із продуктом здійснюється через застосунок скомпільований для конкретної платформи.

Для взаємодії програми з базами даних різних типів, використано інструмент Entity Framework [7].

Схематично процеси взаємодії програмного комплексу зображено на рис. 4.1.1.

4.2. Функціональні можливості системи

Програмне забезпечення має такі функціональні можливості:

- взаємодія із даними типу нотатка (запис, або питання і відповідні йому відповіді, що складаються із різнотипних частин);
- включення до нотатки таких частин як: текст, веб-посилання, зображення, аудіо, відео, файл;
- редагування, завантаження й збереження нотатох;
- різноманітні формати для збереження нотаток: json, txt, pdf, зображення, аудіо, відео, excel, word;
- різноманітні формати для завантаження нотаток: json, txt, розпізнавання зображень, запис мікрофону, запис веб-камери;
- поширення нотаток до бази даних;
- синхронізація із базою даних;
- теги для групування нотаток;
- групування нотаток у віртуальні папки;
- пошук по ключовим словам;
- підсвічування співпадінь у фільтруванні пошуку;
- темна і світла теми;
- зміна мови інтерфейсу.

4.3. Користувацький інтерфейс

Інтерфейс програмного продукту є однаковим для мобільних платформ (Android, iOS, UWP) і відрізняється для Win32, однак керуються вони всі одним і тим же контролером. Загальні кнопки керування для мобільних

платформ винесено у бокове меню ліворуч, а для Win32 – вони знаходяться у верхньому головному меню.

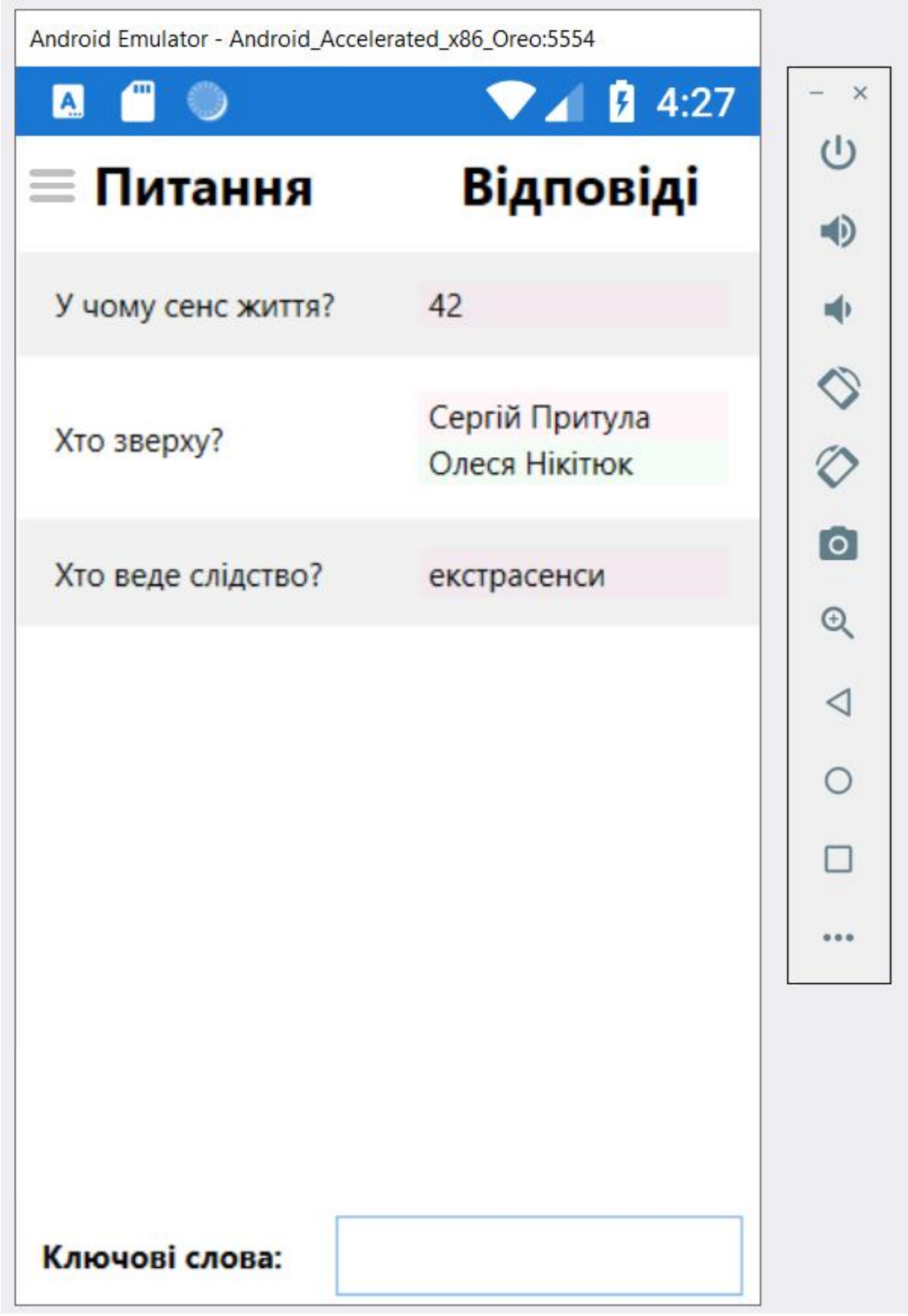

Рис. 4.3.1. Android. Головна панель

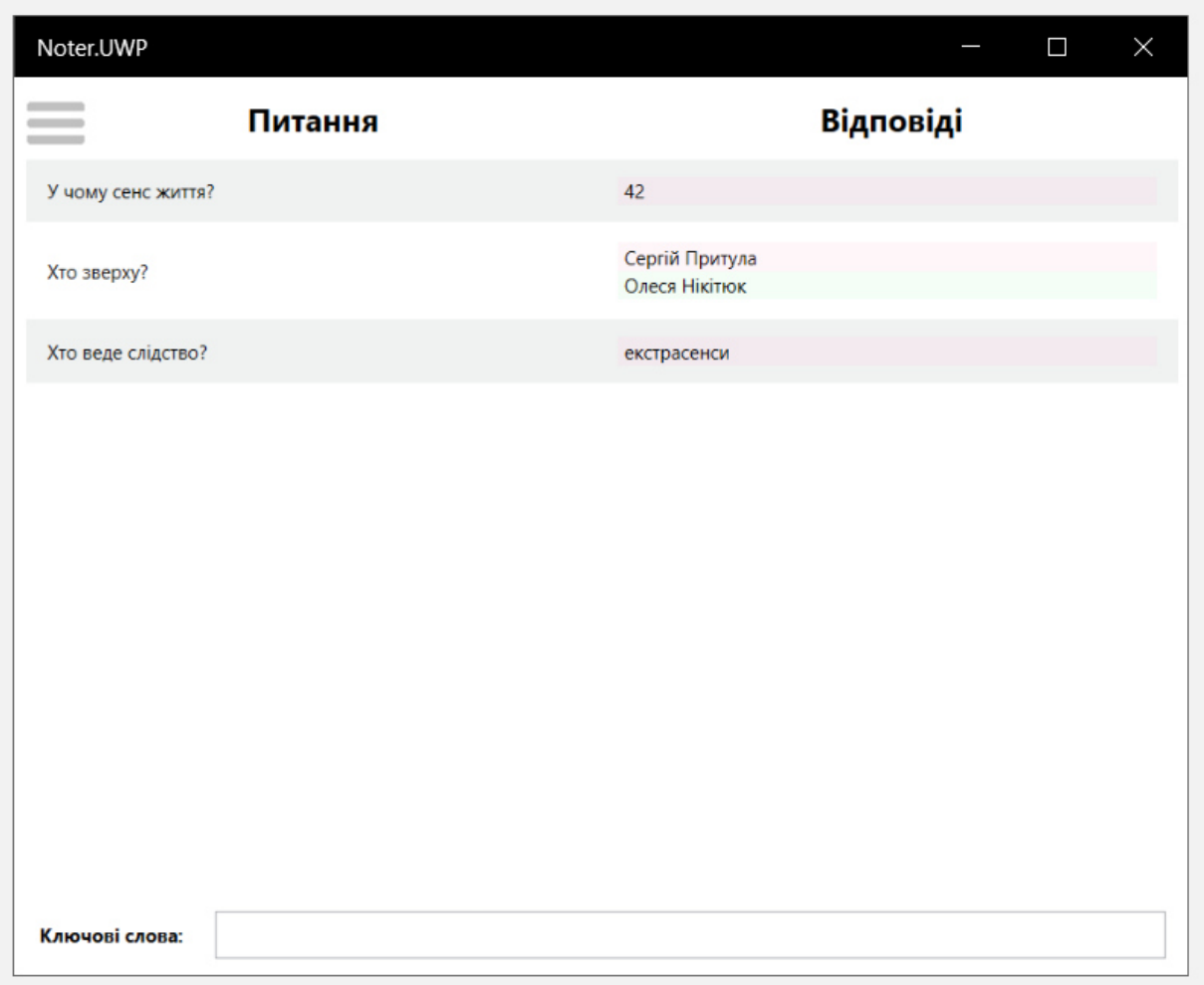

Рис. 4.3.2. UWP. Головна панель

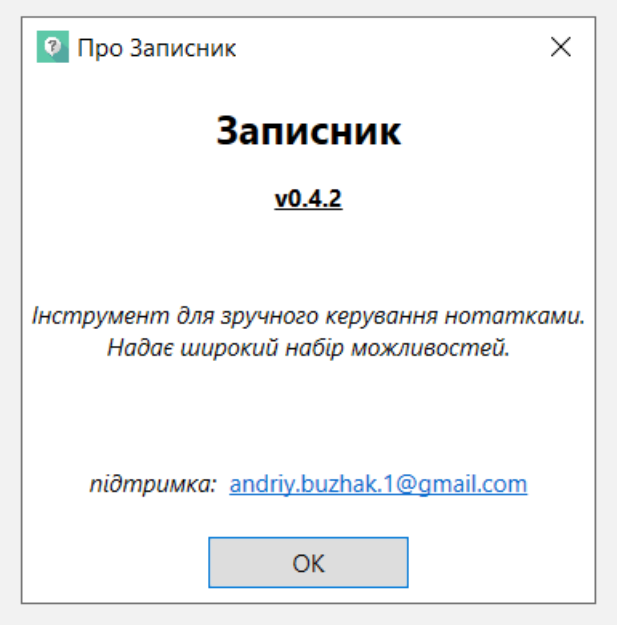

Рис. 4.3.3. Win32. Вікно про програму

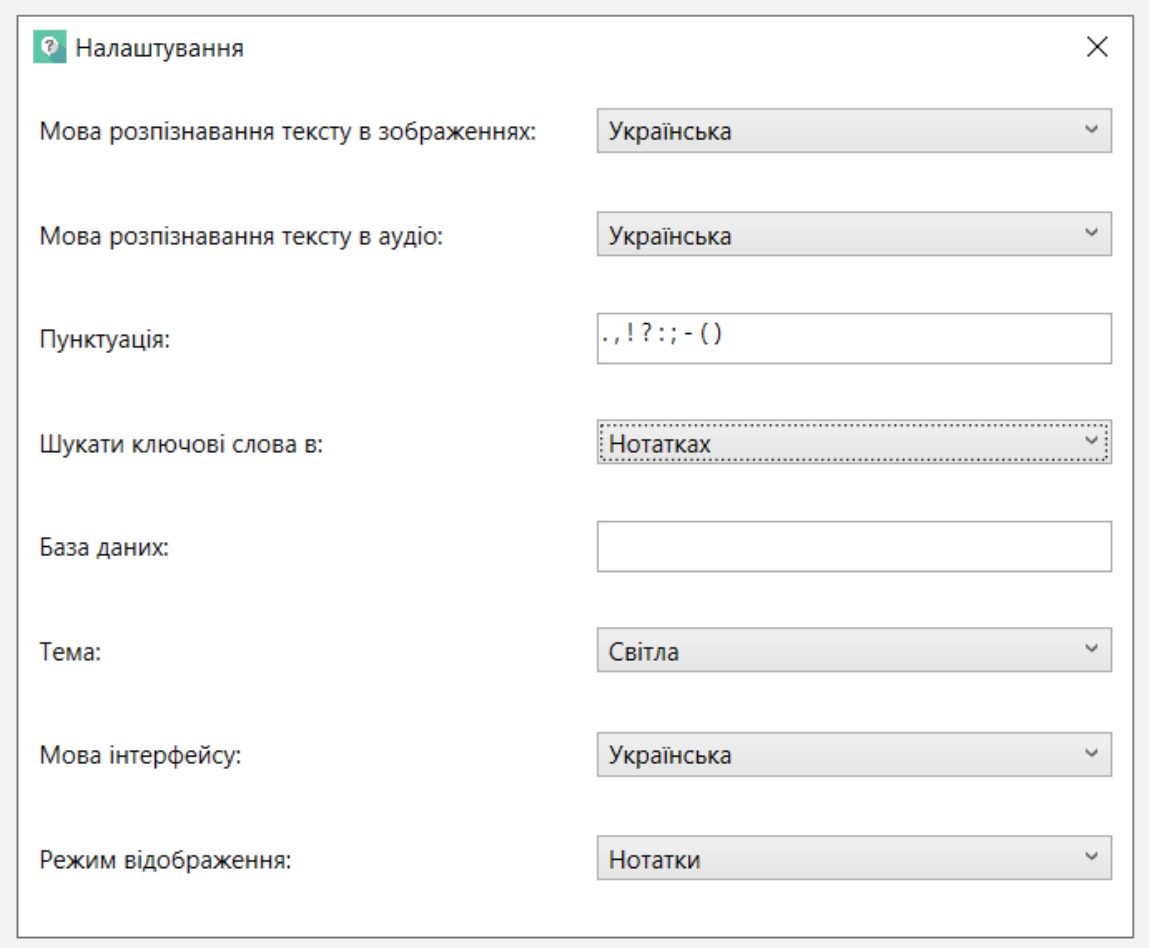

Рис. 4.3.4. Win32. Вікно налаштувань

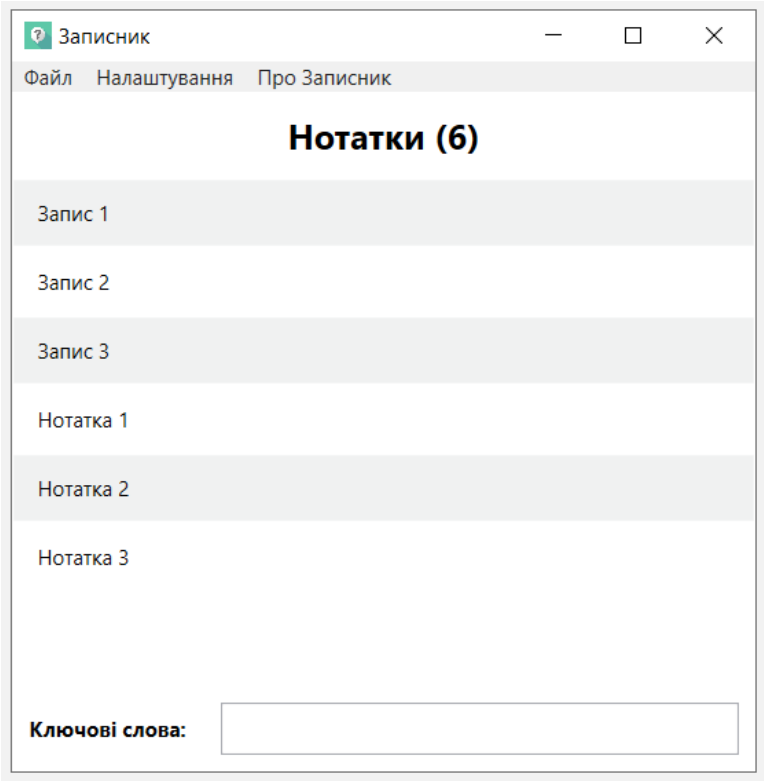

Рис. 4.3.5. Win32. Головне вікно

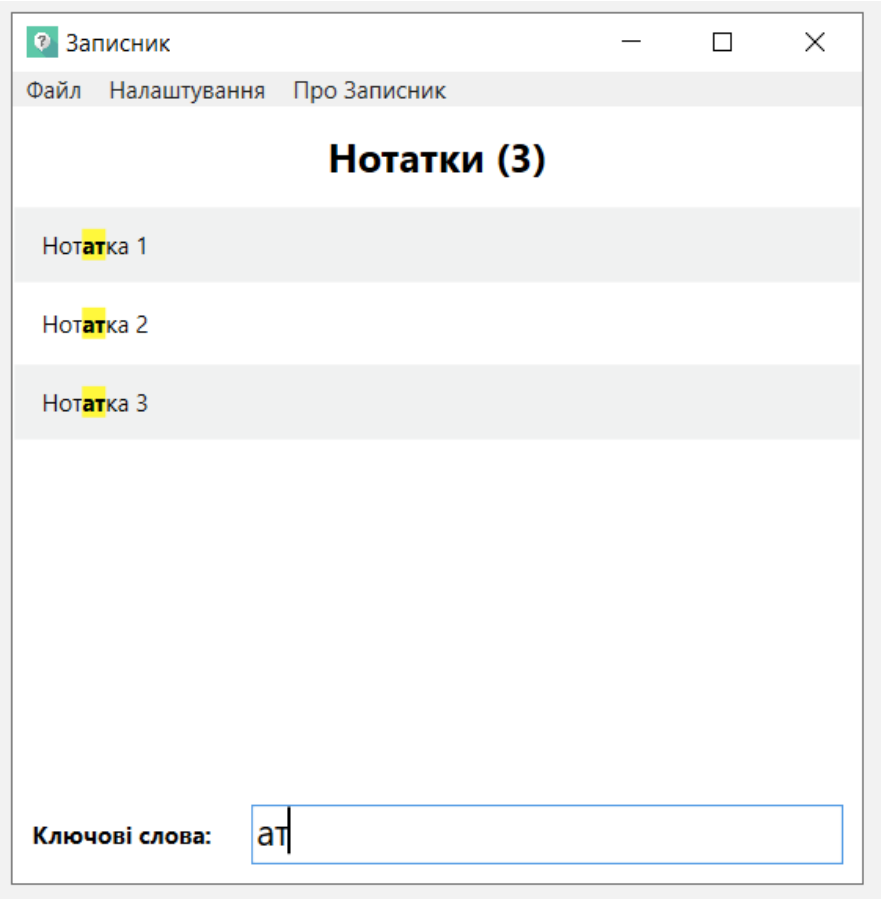

Рис. 4.3.6. Win32. Фільтрування за ключовими словами

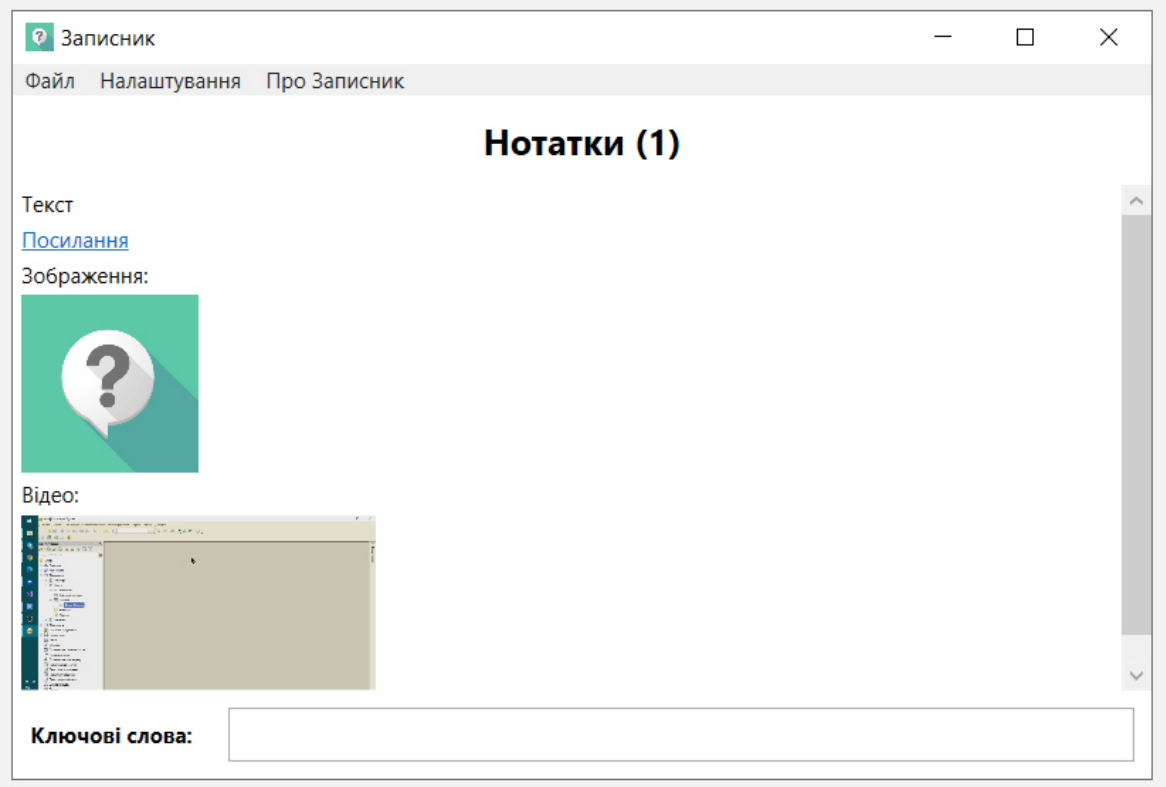

Рис. 4.3.7. Win32. Нотатка з різноманітними компонентами

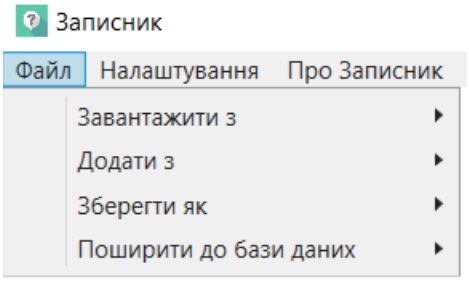

Рис. 4.3.8. Win32. Пункт меню «Файл»

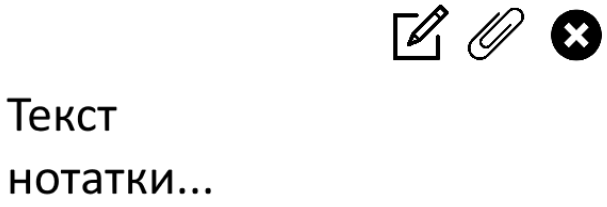

Рис. 4.3.9. Кнопки редагування обраної нотатки

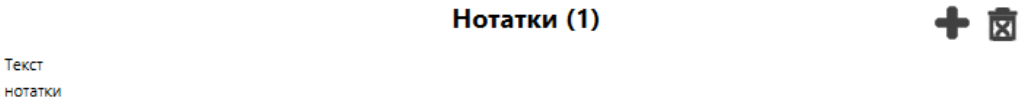

Рис. 4.3.10. Кнопка додавання нової нотатки, та видалення всіх нотаток (з’являються при наведенні на заголовок)

4.4. Опис класів та програмних модулів

Як зазначалося в розділі 3.5, розроблене програмне забезпечення складається із 5 збірок. Головна збірка – Noter.

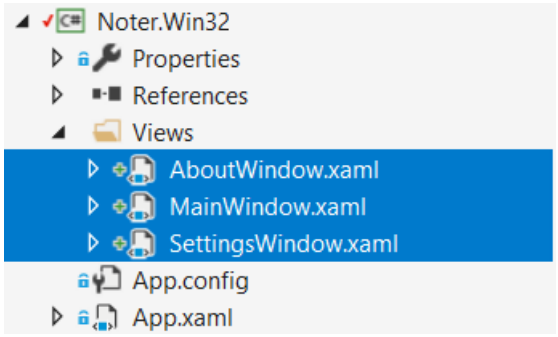

Рис. 4.4.1. Класи Noter.Win32

Збірка «Noter.Win32» лише перевизначає представлення (Views [4]), рис. 4.4.1.

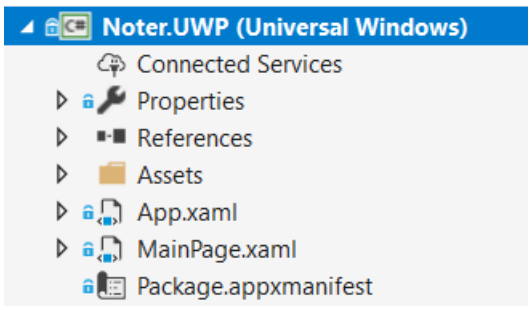

Рис. 4.4.2. Класи Noter.UWP

Збірка «Noter.UWP» не містить жодних компонентів (рис. 4.4.2), окрім створених за замовчуванням, оскільки вона працює на .NET Core, як і цільова збірка для Xamarin.

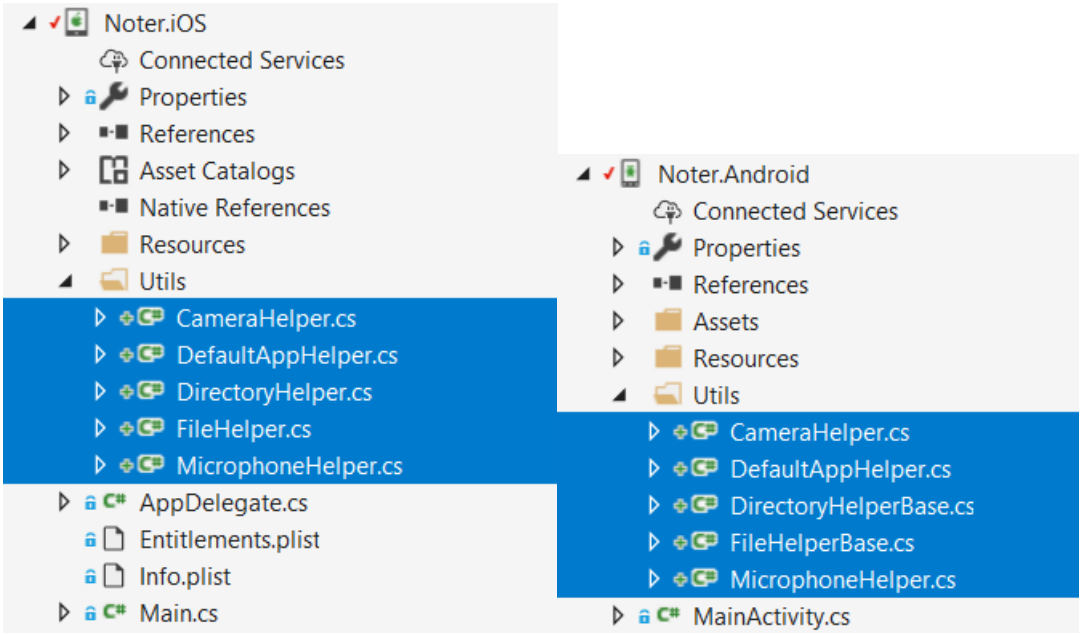

Рис. 4.4.3 Класи Noter.iOS та Noter.Android

Збірки «Noter.iOS» та «Noter.Android» перевизначають лише класи CameraHelper, DefaultAppHelper, DirectoryHelper, FileHelper та MicrophoneHelper (рис. 4.4.3), оскільки вони вимагають специфічного виконання в цих операційних системах.

Ці класи-помічники є нащадками базових класів збірки «Noter». Базові помічники є абстрактними класами і визначені для платформи .NET Core (рис. 4.4.4).

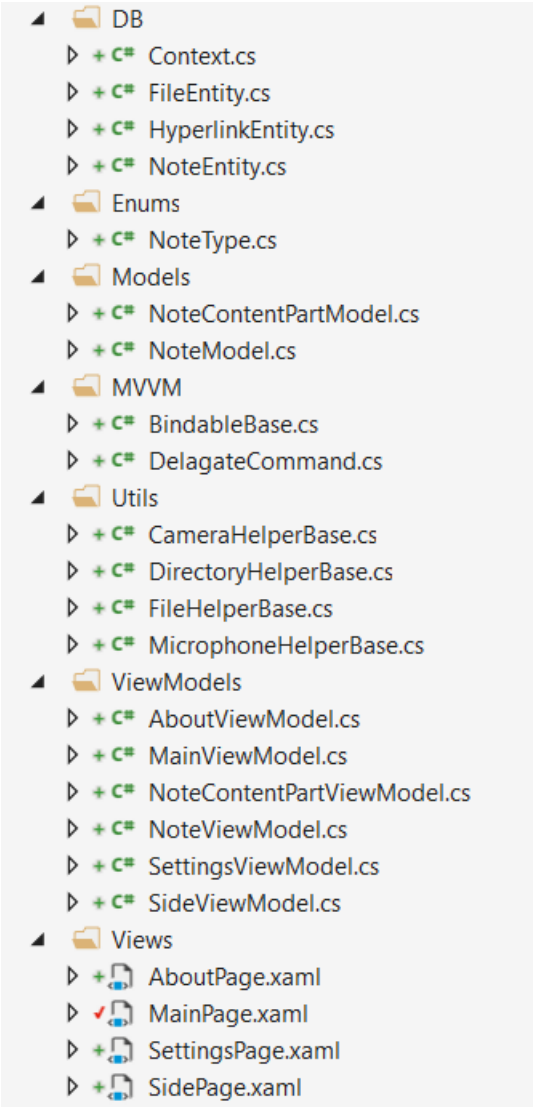

Рис. 4.4.4. Класи збірки Noter

Папка «Views» містить представлення всіх візуальних панелей: головна панель, панель налаштувань, панель про програму та бокова панель.

Папка «ViewModels» містить контролери перелічених представлень, а також два контролери для нотатки, та частини вмісту нотатки – NoteViewModel та NoteContentPartViewModel відповідно.

Моделі NoteModel та NoteContentPartModel використовують в контролерах для керуванням даними у пам’яті, для серіалізації даних у формат JSON, та для представлення сутностей бази даних у класі «DB.NoteEntity».

Папка «MVVM» містить реалізації інтерфейсів «System.ComponentModel.INotifyProperyChanged» та «System.Windows.Input.ICommand».

Папка «DB» містить класи опису сутностей бази даних, де клас Context – представлення колекцій усіх даних у базі.

4.5. Методика та результати випробувань

Варто зауважити, що загалом завантаження даних до програми може відбуватися за двома режимами: додавання або заміни (про що свідчать пункти меню на рис. 4.3.8). Отож, будь яка комбінація завантаження із довільного із доступних форматів повинна призводити до очікуваного результату – список нотаток у пам'яті програми повинен оновлюватися відповідно до виконаної дії завантаження (додавання чи перезапису).

Збереження даних, у довільний із доступних форматів, жодним чином не повинно: змінювати відображення нотаток у програмі, впливати на подальші завантаження, впливати на подальші збереження або на взаємодію із базою даних.

Збережений файл, вважається непошкодженим, якщо його можна завантажити назад до програми. Повноту і цілісність даних такого сценарію варто тестувати у форматі JSON.

За час проведення тестувань, мною було виявлено чимало сценаріїв що так чи інакше призводили до некоректної роботи програми, або неможливості її роботи. Усі ці випадки враховані у коді програмного забезпечення та виправлені або оброблені відповідним чином.

Таким чином, мною було проведене тестування роботи розробленого програмного продукту. За результатами випробувань програмне забезпечення показало свою працездатність.

ВИСНОВКИ ДО ПРАКТИЧНОЇ ЧАСТИНИ

У ході роботи над практичною частиною було завершено створення програмного забезпечення «Записник» для операційних систем Windows, Android та iOS.

У розробленому продукті реалізовані:

– засоби зручного маніпулювання колекціями даних типу «Нотатка»;

– швидкий пошук і фільтрування нотаток по заданим ключовим та візуальна підсвітка співпадінь;

– інструменту запису голосу, відео до нотаток;

– розпізнавання тексту у зображеннях;

– спеціалізоване відображення зображень, аудіо та відео у нотатках.

Програмний продукт розроблений мовою програмування C# із використанням середовища розробки Microsoft Visual Studio та платформи Xamarin. Для роботи з базами даних було використано інструмент Entity Framework. Для розпізнавання тексту в зображеннях була використана відкрита бібліотека класів «Tesseract.Net.SDK»[8]. Виокремлення тексту із аудіо файлів здійснювалось засобами відкритої бібліотеки «Plugin.SpeechRecognition»[9].

Перевірка роботи програмного продукту здійснювалася шляхом ручного тестування проблемних місць, а також автоматизованим тестування через написання модульних тестів з використанням інструменту NUnit[10].

Розроблений програмний продукт відповідає поставленим вимогам та продемонстрував свою працездатність.

ЗАГАЛЬНІ ВИСНОВКИ ДО РОБОТИ

У цій роботі мною проводилося дослідження проблеми кросплатформної розробки програмного забезпечення на прикладі створення програми «Записник».

Для конструювання ПЗ були використані такі алгоритми:

1) Паттерн проектування MVVM [4].

2) Дотримання принципів придатності модулів до тестування та засобів NUnit[10].

3) Методи проектування хорошого інтерфейсу користувача [3, с. 48].

У цілому, був розроблений програмний продукт для розв'язання задачі маніпулювання колекціями даних типу «Нотатка», із вмістом спеціалізованого представлення (зображення, відео, аудіо, веб-посилання), який за результатами випробувань показав свою працездатність. Розроблений програмний продукт можна використовувати для:

- нотування особистих ідей;
- увіковічення рішень деяких проблем, до трапляються досить рідко аби їх забути, але досить часто щоб зекономати час на пошуку занотувавши їх;
- для обміну відповідями на філософські питання людства – шляхом поширення і опублікування даних до публічних баз.

Використання продукту є поширюваним в перспективі, адже розроблено в узагальненій формі, та може конкретизуватися під специфічні конфігурації.

ВИКОРИСТАНА ЛІТЕРАТУРА

1. Charles Petzold. Creating Mobile Apps with Xamarin.Forms. Redmond, Washington 98052-6399 : Microsoft Press, 2016. 1680 с.
2. Нехотина В.С. Модель оценки ИТ-проектов. *Научные ведомости*. 2014. №8. URL: https://cyberleninka.ru/article/n/model-otsenki-it-proektov/viewer (дата звернення: 01.06.2020).
3. Головач В.В. Дизайн пользовательского интерфейса. Искусство мыть слона. 2010. 97 с. URL: http://salikhovilyas.ru/uploads/books/4fc69c328a1be.pdf (дата звернення: 01.06.2020).
4. Батюк З.В., Тарасов О.В. Використання шаблонів проектування MVVM при розробці застосувань на платформі Microsoft. Харківський національний економічний університет імені С. Кузнеця. 2015. 4 с. URL: http://www.hups.mil.gov.ua/periodic-app/article/12159/soi_2015_4_3.pdf (дата звернення: 01.06.2020).
5. Скин Джош, Гринхол Дэвид. Kotlin. Программирование для профессионалов. Москва : Питер, 2020. 464 с.
6. Усов В. Swift. Основы разработки приложений под iOS и macOS. Москва : Питер, 2017. 368 с.
7. Julia Lerman, Rowan Miller. Programming Entity Framework: Code First. CA 95472 : O'Reilly, 2011. 192 с.
8. Tesseract.Net.SDK. *nuget* : веб-сайт. URL: https://www.nuget.org/packages/Tesseract.Net.SDK (дата звернення: 01.06.2020).
9. Plugin.SpeechRecognition. *nuget* : веб-сайт. URL:

https://www.nuget.org/packages/Plugin.SpeechRecognition (дата звернення: 01.06.2020).

10. Введение в модульное тестирование для C# проектов в среде MonoDevelop. *Хабр* : веб-сайт. URL: https://habr.com/ru/post/181255 (дата звернення: 01.06.2020).

ДОДАТКИ

Додаток А
Код програми

```
// Notes.json sample.
[
  {
    "Question": "Q_1",
    "Answers": [
      "A_1"
    ]
  },
  {
    "Question": "Q_2",
    "Answers": [
      "A_2_1",
      "A_2_2"
    ]
  },
  {
    "Question": "Q_3",
    "Answers": [
      "A_3"
    ]
  }
]
```

```
// MainWindow.xaml.cs.
public partial class MainWindow : Window
    {
        public MainWindow()
        {
            InitializeComponent();
            DataContext = new MainViewModel();
        }
    }
```

```
// MainWindow.xaml.
<Window x:Class="Noter.Views.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:QAHelper"
        xmlns:WPF="clr-namespace:QAHelper.WPF"
        WindowStartupLocation="CenterScreen"
        ResizeMode="CanResize"
        MinWidth="300" MinHeight="250"
        Title="Записник" Height="450" Width="800">

    <Window.Icon>
        <BitmapImage UriSource="pack://application:,,,/Noter;component/Images/question.png"/>
    </Window.Icon>

    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="auto"/>
            <RowDefinition Height="50"/>
            <RowDefinition Height="*"/>
            <RowDefinition Height="50"/>
        </Grid.RowDefinitions>
        <Grid.ColumnDefinitions>
            <ColumnDefinition/>
            <ColumnDefinition/>
        </Grid.ColumnDefinitions>

        <Menu Grid.Row="0" Grid.Column="0" Grid.ColumnSpan="2">
            <MenuItem Header="Файл">
                <MenuItem Header="Завантажити з">
                    <MenuItem Header="JSON" Command="{Binding LoadQAItemsFromJsonCommand}"/>
                    <MenuItem Header="Images" Command="{Binding LoadQAItemsFromImagesCommand}"/>
                </MenuItem>
```

```
            <MenuItem Header="Додати з">
              <MenuItem Header="JSON" Command="{Binding AppendQAItemsFromJsonCommand}"/>
              <MenuItem Header="Images" Command="{Binding AppendQAItemsFromImagesCommand}"/>
            </MenuItem>
            <MenuItem Header="Зберегти як">
              <MenuItem Header="JSON" Command="{Binding SaveQAItemsIntoJsonCommand}"/>
              <!--<MenuItem Header="PDF" Command="{Binding SaveQAItemsIntoPdfCommand}"/>-->
              <!--<MenuItem Header="Excel" Command="{Binding SaveQAItemsIntoExcelCommand}"/>-->
              <!--<MenuItem Header="TXT" Command="{Binding SaveQAItemsIntoTxtCommand}"/>-->
            </MenuItem>
            <MenuItem Header="Поширити до бази даних">
              <MenuItem/>
            </MenuItem>
          </MenuItem>
          <MenuItem Header="Налаштування" Command="{Binding SettingsCommand}"/>
          <MenuItem Header="Про Записник" Command="{Binding AboutCommand}"/>
        </Menu>
        <TextBlock Grid.Row="1" Grid.Column="0" Grid.ColumnSpan="2" Text="Нотатки (1)" VerticalAlignment="Center"
HorizontalAlignment="Center" FontWeight="Bold" FontSize="20"/>
        <ScrollViewer Grid.Row="2" Grid.Column="0" Grid.ColumnSpan="2" ScrollViewer.HorizontalScrollBarVisibility="Disabled"
ScrollViewer.VerticalScrollBarVisibility="Auto">
          <ScrollViewer.Resources>
            <Style x:Key="AlternatingListViewItemStyle_1" TargetType="{x:Type ListViewItem}">
              <Style.Triggers>
                <Trigger Property="ItemsControl.AlternationIndex" Value="0">
                  <Setter Property="Background" Value="#77dee0e0"/>
                </Trigger>
                <Trigger Property="ItemsControl.AlternationIndex" Value="1">
                  <Setter Property="Background" Value="White"/>
                </Trigger>
              </Style.Triggers>
            </Style>
            <Style x:Key="AlternatingListViewItemStyle_2" TargetType="{x:Type ListViewItem}">
              <Setter Property="Focusable" Value="False"/>
              <Setter Property="Background" Value="Transparent"/>
              <Setter Property="BorderThickness" Value="0"/>
              <Style.Triggers>
                <Trigger Property="ItemsControl.AlternationIndex" Value="0">
                  <Setter Property="Background" Value="#20ffbada"/>
                </Trigger>
                <Trigger Property="ItemsControl.AlternationIndex" Value="1">
                  <Setter Property="Background" Value="#20abf5bf"/>
                </Trigger>
              </Style.Triggers>
            </Style>
          </ScrollViewer.Resources>

          <ListView Name="qaItemsList" HorizontalContentAlignment="Stretch" AlternationCount="2">
            <ListView.Template>
              <ControlTemplate>
                <ItemsPresenter/>
              </ControlTemplate>
            </ListView.Template>

            <ListViewItem>
              <TextBlock Text="Текст"/>
            </ListViewItem>
            <ListViewItem>
              <TextBlock Text="нотатки"/>
            </ListViewItem>
          </ListView>
        </ScrollViewer>
        <Grid Grid.Row="3" Grid.Column="0" Grid.ColumnSpan="2">
          <Grid.ColumnDefinitions>
            <ColumnDefinition Width="auto"/>
            <ColumnDefinition Width="*"/>
          </Grid.ColumnDefinitions>

          <TextBlock Grid.Column="0" FontWeight="Bold" VerticalAlignment="Center" Margin="10" Text="Ключові слова:">
            <TextBlock.ToolTip>
              Look up for questions or/and answers (according to Settings) with specified words,
              in specified order, separated with spacebar and punctuation specified in Settings.
            </TextBlock.ToolTip>
          </TextBlock>
          <TextBox Grid.Column="1" VerticalAlignment="Center" Margin="10,0,10,0" FontSize="18" Text="{Binding
QuestionSearchWordsString, Mode=OneWayToSource, UpdateSourceTrigger=PropertyChanged}" Height="30"/>
        </Grid>
      </Grid>
```

```
</Window>


// MVVM.BindableBase.
public abstract class BindableBase : INotifyPropertyChanged
    {
        public event PropertyChangedEventHandler PropertyChanged;

        /// <summary>
        /// This method is called by the Set accessor of each property.
        /// The CallerMemberName attribute that is applied to the optional propertyName
        /// parameter causes the property name of the caller to be substituted as an argument.
        /// </summary>
        /// <param name="propertyName">Name of changed property.</param>
        protected void RaisePropertyChanged([CallerMemberName] string propertyName = "")
        {
            PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));
        }
    }


// MVVM.DelegateCommand.
public class DelegateCommand : ICommand
    {
        public event EventHandler CanExecuteChanged;

        private readonly Action<object> _executeMethod;
        private readonly Func<object, bool> _canExecuteMethod;

        public DelegateCommand(Action executeMethod, Func<bool> canExecuteMethod = null)
            : this((obj) =>
            {
                if (executeMethod == null)
                {
                    return;
                }
                executeMethod();
            },
            (obj) => canExecuteMethod == null ? true : canExecuteMethod())
        { }

        public DelegateCommand(Action<object> executeMethod, Func<object, bool> canExecuteMethod = null)
        {
            _executeMethod = executeMethod;
            _canExecuteMethod = canExecuteMethod;

            if (_executeMethod == null)
            {
                _executeMethod = (obj) => { };
            }

            if (_canExecuteMethod == null)
            {
                _canExecuteMethod = (obj) => true;
            }
        }

        public bool CanExecute(object parameter)
        {
            return _canExecuteMethod(parameter);
        }

        public void Execute(object parameter)
        {
            _executeMethod(parameter);
        }

        public void RaiseCanExecuteChanged()
        {
            CanExecuteChanged?.Invoke(this, EventArgs.Empty);
        }
    }


// WPF.EnumToItemsSourceExtension.
public class EnumToItemsSourceExtension : MarkupExtension
    {
        private Type Type { get; }
```

```
        public EnumToItemsSourceExtension(Type type)
        {
            Type = type;
        }

        public override object ProvideValue(IServiceProvider serviceProvider)
        {
            if ((Type != null) && Type.IsEnum)
            {
                return Enum.GetValues(Type)
                    .OfType<object>()
                    .Select(e => new { Value = e, DisplayName = GetEnumValueDescription(e) });
            }
            return new object[0];
        }

        private string GetEnumValueDescription(object enumValue)
        {
            Type enumType = enumValue.GetType();
            string enumFieldName = enumValue.ToString();

            string description = enumType
                .GetField(enumFieldName)
                ?.GetCustomAttributes(typeof(DescriptionAttribute), false)
                .OfType<DescriptionAttribute>()
                .FirstOrDefault()
                ?.Description;

            return string.IsNullOrWhiteSpace(description) ? enumFieldName : description;
        }
    }


// WPF.HighlightableTextBlock.
public class HighlightableTextBlock : SelectableTextBlock
    {
        public static readonly DependencyProperty HighlightableTextPartsProperty = DependencyProperty.Register(
            "HighlightableTextParts",
            typeof(IList<Run>),
            typeof(HighlightableTextBlock),
            new FrameworkPropertyMetadata { PropertyChangedCallback = OnChanged });

        public IList<Run> HighlightableTextParts
        {
            get => (IList<Run>)GetValue(HighlightableTextPartsProperty);
            set
            {
                SetValue(HighlightableTextPartsProperty, value);
            }
        }

        private static void OnChanged(DependencyObject d, DependencyPropertyChangedEventArgs e)
        {
            if (d is HighlightableTextBlock textBlock)
            {
                textBlock.Inlines.Clear();
                textBlock.Inlines.AddRange(textBlock.HighlightableTextParts);
            }
        }
    }


// WPF.SelectableTextBlock.
public class SelectableTextBlock : TextBlock
    {
        static SelectableTextBlock()
        {
            FocusableProperty.OverrideMetadata(typeof(SelectableTextBlock), new FrameworkPropertyMetadata(true));
            TextEditorWrapper.RegisterCommandHandlers(typeof(SelectableTextBlock), true, true, true);

            // Remove the focus rectangle around the control.
            FocusVisualStyleProperty.OverrideMetadata(typeof(SelectableTextBlock), new FrameworkPropertyMetadata((object)null));
        }

        private readonly TextEditorWrapper _editor;

        public SelectableTextBlock()
        {
            _editor = TextEditorWrapper.CreateFor(this);
```

```
        }
    }


// WPF.TextEditorWrapper.
public class TextEditorWrapper
    {
        private static readonly Type TextEditorType = Type.GetType("System.Windows.Documents.TextEditor, PresentationFramework,
Version=4.0.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35");
        private static readonly PropertyInfo IsReadOnlyProp = TextEditorType.GetProperty("IsReadOnly", BindingFlags.Instance |
BindingFlags.NonPublic);
        private static readonly PropertyInfo TextViewProp = TextEditorType.GetProperty("TextView", BindingFlags.Instance |
BindingFlags.NonPublic);
        private static readonly MethodInfo RegisterMethod = TextEditorType.GetMethod("RegisterCommandHandlers",
            BindingFlags.Static | BindingFlags.NonPublic, null, new[] { typeof(Type), typeof(bool), typeof(bool), typeof(bool) }, null);

        private static readonly Type TextContainerType = Type.GetType("System.Windows.Documents.ITextContainer, PresentationFramework,
Version=4.0.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35");
        private static readonly PropertyInfo TextContainerTextViewProp = TextContainerType.GetProperty("TextView");

        private static readonly PropertyInfo TextContainerProp = typeof(TextBlock).GetProperty("TextContainer", BindingFlags.Instance |
BindingFlags.NonPublic);

        public static void RegisterCommandHandlers(Type controlType, bool acceptsRichContent, bool readOnly, bool registerEventListeners)
        {
            RegisterMethod.Invoke(null, new object[] { controlType, acceptsRichContent, readOnly, registerEventListeners });
        }

        public static TextEditorWrapper CreateFor(TextBlock tb)
        {
            var textContainer = TextContainerProp.GetValue(tb);

            var editor = new TextEditorWrapper(textContainer, tb, false);
            IsReadOnlyProp.SetValue(editor._editor, true);
            TextViewProp.SetValue(editor._editor, TextContainerTextViewProp.GetValue(textContainer));

            return editor;
        }

        private readonly object _editor;

        public TextEditorWrapper(object textContainer, FrameworkElement uiScope, bool isUndoEnabled)
        {
            _editor = Activator.CreateInstance(TextEditorType, BindingFlags.Instance | BindingFlags.NonPublic | BindingFlags.CreateInstance,
                null, new[] { textContainer, uiScope, isUndoEnabled }, null);
        }
    }


// Views.AboutWindow.xaml.
<Window x:Class="Noter.Win32.Views.AboutWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:QAHelper.Views"
        WindowStartupLocation="CenterScreen"
        ResizeMode="NoResize"
        Title="Про Записник" Height="300" Width="300">

    <Window.Icon>
        <BitmapImage UriSource="pack://application:,,,/Noter;component/Images/question.png"/>
    </Window.Icon>

    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="50"/>
            <RowDefinition Height="20"/>
            <RowDefinition Height="*"/>
            <RowDefinition Height="40"/>
            <RowDefinition Height="40"/>
        </Grid.RowDefinitions>

        <TextBlock Grid.Row="0" Text="Записник" HorizontalAlignment="Center" VerticalAlignment="Center" FontSize="20"
FontWeight="Bold"/>
        <TextBlock Grid.Row="1" Text="{Binding ToolVersion, Mode=OneTime}" HorizontalAlignment="Center" VerticalAlignment="Center"
FontWeight="Bold" TextDecorations="Underline"/>
        <TextBlock Grid.Row="2" Text="Інструмент для зручного керування нотатками. Надає широкий набір можливостей."
HorizontalAlignment="Center" VerticalAlignment="Center" TextWrapping="Wrap" TextAlignment="Center" FontStyle="Italic"/>
```

```
        <TextBlock Grid.Row="3" HorizontalAlignment="Center" VerticalAlignment="Center">
            <Run Text="підтримка: " FontStyle="Italic"/>
            <Hyperlink Command="{Binding SupportCommand}">
                <Run Text="{Binding SupportEmail, Mode=OneTime}"/>
            </Hyperlink>
        </TextBlock>
        <Button Grid.Row="4" Content="OK" Height="25" Width="100" IsDefault="True" IsCancel="True"/>
    </Grid>

</Window>


// Views.AboutWindow.xaml.cs.
public partial class AboutWindow : Window
    {
        public AboutWindow(AboutViewModel viewModel)
        {
            var windowInteropHelper = new WindowInteropHelper(this)
            {
                Owner = Process.GetCurrentProcess().MainWindowHandle
            };
            InitializeComponent();
            DataContext = viewModel;
        }
    }


// Views.SettingsWindow.xaml.
<Window x:Class="Noter.Win32.Views.SettingsWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:QAHelper.Views"
        xmlns:WPF="clr-namespace:QAHelper.WPF"
        xmlns:Enums="clr-namespace:QAHelper.Enums"
        WindowStartupLocation="CenterScreen"
        ResizeMode="NoResize"
        Title="Налаштування" Height="450" Width="550">

    <Window.Icon>
        <BitmapImage UriSource="pack://application:,,,/QAHelper;component/Images/question.png"/>
    </Window.Icon>

    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="50"/>
            <RowDefinition Height="50"/>
            <RowDefinition Height="50"/>
            <RowDefinition Height="50"/>
            <RowDefinition Height="50"/>
            <RowDefinition Height="50"/>
            <RowDefinition Height="50"/>
            <RowDefinition Height="50"/>
        </Grid.RowDefinitions>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="*"/>
            <ColumnDefinition Width="*"/>
        </Grid.ColumnDefinitions>

        <TextBlock Grid.Row="0" Grid.Column="0" Text="Мова розпізнавання тексту в зображеннях:" ToolTip="Uses on loading from
images." HorizontalAlignment="Left" VerticalAlignment="Center" Margin="10"/>
        <ComboBox Grid.Row="0" Grid.Column="1" HorizontalAlignment="Stretch" VerticalAlignment="Center" Margin="10">
            <ComboBox.Items>
                <ComboBoxItem Content="Англійська"/>
                <ComboBoxItem Content="Українська"/>
            </ComboBox.Items>
        </ComboBox>
        <TextBlock Grid.Row="1" Grid.Column="0" Text="Мова розпізнавання тексту в аудіо:" ToolTip="Uses on loading from images."
HorizontalAlignment="Left" VerticalAlignment="Center" Margin="10"/>
        <ComboBox Grid.Row="1" Grid.Column="1" HorizontalAlignment="Stretch" VerticalAlignment="Center" Margin="10">
            <ComboBox.Items>
                <ComboBoxItem Content="Англійська"/>
                <ComboBoxItem Content="Українська"/>
            </ComboBox.Items>
        </ComboBox>
        <TextBlock Grid.Row="2" Grid.Column="0" Text="Пунктуація:" ToolTip="Punctuation strings separated by spacebar."
VerticalAlignment="Center" Margin="10"/>
        <TextBox Grid.Row="2" Grid.Column="1" Text="{Binding Punctuation, Mode=TwoWay, Delay=10}" Height="25" Margin="10"/>
```

```
        <TextBlock Grid.Row="3" Grid.Column="0" Text="Шукати ключові слова в:" ToolTip="Uses in searching by key words."
HorizontalAlignment="Left" VerticalAlignment="Center" Margin="10"/>
        <ComboBox Grid.Row="3" Grid.Column="1" HorizontalAlignment="Stretch" VerticalAlignment="Center" Margin="10">
            <ComboBox.Items>
                <ComboBoxItem Content="Нотатках"/>
                <ComboBoxItem Content="Питаннях"/>
                <ComboBoxItem Content="Питаннях і Відповідях"/>
                <ComboBoxItem Content="Скрізь"/>
            </ComboBox.Items>
        </ComboBox>
        <TextBlock Grid.Row="4" Grid.Column="0" Text="База даних:" ToolTip="Punctuation strings separated by spacebar."
VerticalAlignment="Center" Margin="10"/>
        <TextBox Grid.Row="4" Grid.Column="1" Text="" Height="25" Margin="10"/>
        <TextBlock Grid.Row="5" Grid.Column="0" Text="Тема:" ToolTip="Uses in searching by key words." HorizontalAlignment="Left"
VerticalAlignment="Center" Margin="10"/>
        <ComboBox Grid.Row="5" Grid.Column="1" HorizontalAlignment="Stretch" VerticalAlignment="Center" Margin="10">
            <ComboBox.Items>
                <ComboBoxItem Content="Світла"/>
                <ComboBoxItem Content="Темна"/>
            </ComboBox.Items>
        </ComboBox>
        <TextBlock Grid.Row="6" Grid.Column="0" Text="Мова інтерфейсу:" ToolTip="Uses in searching by key words."
HorizontalAlignment="Left" VerticalAlignment="Center" Margin="10"/>
        <ComboBox Grid.Row="6" Grid.Column="1" HorizontalAlignment="Stretch" VerticalAlignment="Center" Margin="10">
            <ComboBox.Items>
                <ComboBoxItem Content="Англійська"/>
                <ComboBoxItem Content="Українська"/>
            </ComboBox.Items>
        </ComboBox>
        <TextBlock Grid.Row="7" Grid.Column="0" Text="Режим відображення:" ToolTip="Uses in searching by key words."
HorizontalAlignment="Left" VerticalAlignment="Center" Margin="10"/>
        <ComboBox Grid.Row="7" Grid.Column="1" HorizontalAlignment="Stretch" VerticalAlignment="Center" Margin="10">
            <ComboBox.Items>
                <ComboBoxItem Content="Нотатки"/>
                <ComboBoxItem Content="Питання й Відповіді"/>
                <ComboBoxItem Content="Все"/>
            </ComboBox.Items>
        </ComboBox>

        <Button IsCancel="True" Width="0" Height="0" HorizontalAlignment="Left" VerticalAlignment="Top" Background="Transparent"
BorderThickness="0"/>
    </Grid>

</Window>


// Views.SettingsWindow.xaml.cs.
public partial class SettingsWindow : Window
    {
        public SettingsWindow(SettingsViewModel viewModel)
        {
            var windowInteropHelper = new WindowInteropHelper(this)
            {
                Owner = Process.GetCurrentProcess().MainWindowHandle
            };
            InitializeComponent();
            DataContext = viewModel;
        }
    }

// ViewModels.AboutViewModel.cs.
public class AboutViewModel : BindableBase
    {
        public AboutViewModel()
        {
            SupportCommand = new DelegateCommand(SupportAction);
        }

        public string SupportEmail => "andriy.buzhak.1@gmail.com";

        public string ToolVersion
        {
            get
            {
                Version v = Assembly.GetExecutingAssembly().GetName().Version;
                return "v" + string.Join(".", v.Major, v.Minor, v.Build);
            }
        }
```

```
        public DelegateCommand SupportCommand { get; }

        private void SupportAction()
        {
            Process.Start(new Uri("mailto:" + SupportEmail + "?subject=QA Helper " + ToolVersion + " Support: <Please, specify subject of the
issue>&body=<Please, specify description of the issue>").AbsoluteUri);
        }
    }


// AnswerViewModel.cs.
public class AnswerViewModel : HighlightableTextViewModel
    {
        public AnswerViewModel(IList<Run> highlightableTextParts)
            : base(highlightableTextParts)
        { }
    }


// HighlightableTextViewModel.
public abstract class HighlightableTextViewModel
    {
        public HighlightableTextViewModel(IList<Run> highlightableTextParts)
        {
            HighlightableTextParts = highlightableTextParts;
        }

        public IList<Run> HighlightableTextParts { get; }
    }


// MainViewModel.cs.
public class MainViewModel : BindableBase
    {
        private ObservableCollection<QAItem> _qaItems = new ObservableCollection<QAItem>();
        private string _questionSearchWordsString = string.Empty;
        private readonly SettingsModel _settingsModel = new SettingsModel();
        private readonly Brush _highlightBrush = new SolidColorBrush(Color.FromArgb(250, 255, 248, 56));

        public MainViewModel()
        {
            LoadQAItemsFromJsonFile("Sample.json", false);
            AboutCommand = new DelegateCommand(AboutAction);
            SaveQAItemsIntoJsonCommand = new DelegateCommand(SaveQAItemsIntoJsonAction);
            LoadQAItemsFromJsonCommand = new DelegateCommand(() => LoadOrAppendQAItemsFromJson(false));
            AppendQAItemsFromJsonCommand = new DelegateCommand(() => LoadOrAppendQAItemsFromJson(true));
            LoadQAItemsFromImagesCommand = new DelegateCommand(() => LoadOrAppendQAItemsFromImages(false));
            AppendQAItemsFromImagesCommand = new DelegateCommand(() => LoadOrAppendQAItemsFromImages(true));
            SettingsCommand = new DelegateCommand(SettingsAction);
        }

        public int QuestionsNumber => QAItemsFiltered.Count;

        private List<string> SearchWordParts
        {
            get
            {
                string searchWordsStr = QuestionSearchWordsString;

                if (string.IsNullOrEmpty(searchWordsStr))
                {
                    return new List<string>();
                }

                foreach (char p in _settingsModel.Punctuation)
                {
                    searchWordsStr = searchWordsStr.Replace(p, ' ');
                }
                return searchWordsStr.Split(' ').Where(i => !string.IsNullOrWhiteSpace(i)).ToList();
            }
        }

        public List<QuestionViewModel> QAItemsFiltered => TryCatchWrapperMethod(() =>
        {
            List<QuestionViewModel> filteredItems = new List<QuestionViewModel>();

            IList<string> searchWordParts_origin = SearchWordParts;

            if (!searchWordParts_origin.Any())
```

```
        {
            return QAItems.Select(i => new QuestionViewModel(new List<Run> { new Run(i.Question) }, i.Answers.Select(a => new
AnswerViewModel(new List<Run> { new Run(a) })).ToList())).ToList();
        }

        foreach (QAItem item in QAItems)
        {
            bool passQuestion = false;
            bool passAnyAnswer = false;
            var qestionParts = new List<Run> { new Run(item.Question) };
            List<List<Run>> answersParts = item.Answers.Select(a => new List<Run> { new Run(a) }).ToList();

            // Search in Questions.
            if (_settingsModel.KeyWordsSearchType != Enums.KeyWordsSearchType.Answers)
            {
                if (SearchWordPartsInText(item.Question, searchWordParts_origin, out List<Run> qTextParts))
                {
                    qestionParts = qTextParts;
                    passQuestion = true;
                }
            }

            // Search in Answers.
            if (_settingsModel.KeyWordsSearchType != Enums.KeyWordsSearchType.Questions)
            {
                for (int i = 0; i < item.Answers.Count; i++)
                {
                    if (SearchWordPartsInText(item.Answers[i], searchWordParts_origin, out List<Run> answerParts))
                    {
                        passAnyAnswer = true;
                        answersParts[i] = answerParts;
                    }
                }
            }

            bool passed = false;
            switch (_settingsModel.KeyWordsSearchType)
            {
                case Enums.KeyWordsSearchType.Questions:
                    passed = passQuestion;
                    break;
                case Enums.KeyWordsSearchType.Answers:
                    passed = passAnyAnswer;
                    break;
                case Enums.KeyWordsSearchType.Both:
                    passed = passQuestion || passAnyAnswer;
                    break;
            }
            if (passed)
            {
                filteredItems.Add(new QuestionViewModel(qestionParts, answersParts.Select(a => new AnswerViewModel(a))));
            }
        }

        return filteredItems;
    }, new List<QuestionViewModel>());

    public ObservableCollection<QAItem> QAItems
    {
        get => _qaItems;
        set
        {
            _qaItems = value;
            RaisePropertyChanged(nameof(QAItems));
            RaisePropertyChanged(nameof(QAItemsFiltered));
            RaisePropertyChanged(nameof(QuestionsNumber));
        }
    }

    public string QuestionSearchWordsString
    {
        get => _questionSearchWordsString;
        set
        {
            _questionSearchWordsString = value;
            RaisePropertyChanged(nameof(QuestionSearchWordsString));
            RaisePropertyChanged(nameof(QAItemsFiltered));
            RaisePropertyChanged(nameof(QuestionsNumber));
        }
```

```
}

public DelegateCommand AboutCommand { get; }

public DelegateCommand SaveQAItemsIntoJsonCommand { get; }

public DelegateCommand LoadQAItemsFromJsonCommand { get; }

public DelegateCommand AppendQAItemsFromJsonCommand { get; }

public DelegateCommand LoadQAItemsFromImagesCommand { get; }

public DelegateCommand AppendQAItemsFromImagesCommand { get; }

public DelegateCommand SettingsCommand { get; }

private void TryCatchWrapperMethod(Action action)
{
    try
    {
        action();
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Error", MessageBoxButton.OK, MessageBoxImage.Error);
    }
}

private T TryCatchWrapperMethod<T>(Func<T> func, T defaultReturn)
{
    try
    {
        return func();
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Error", MessageBoxButton.OK, MessageBoxImage.Error);
        return defaultReturn;
    }
}

private void GroupAndAssignQAItems(IEnumerable<QAItem> items, bool append)
{
    QAItems = new ObservableCollection<QAItem>(items
        .Concat(append ? QAItems : new ObservableCollection<QAItem>())
        .GroupBy(i => i.Question, i => i.Answers)
        .Select(i => new QAItem
        {
            Question = i.Key,
            Answers = i.SelectMany(a => a)
                .Distinct()
                .Where(a => !string.IsNullOrWhiteSpace(a))
                .ToList()
        })
        .Where(i => !string.IsNullOrWhiteSpace(i.Question)));
}

private void LoadQAItemsFromJsonFile(string filePath, bool append)
{
    TryCatchWrapperMethod(() =>
    {
        GroupAndAssignQAItems(JsonConvert.DeserializeObject<IEnumerable<QAItem>>(File.ReadAllText(filePath)), append);
    });
}

private void AboutAction()
{
    new AboutWindow(new AboutViewModel()).ShowDialog();
}

private void SaveQAItemsIntoJsonAction()
{
    TryCatchWrapperMethod(() =>
    {
        var dialog = new SaveFileDialog
        {
            Filter = "JSON files (*.json)|*.json",
            FileName = "QA_Items"
        };
```

```
            if (dialog.ShowDialog() ?? false)
            {
                File.WriteAllText(dialog.FileName, JsonConvert.SerializeObject(QAItems, Formatting.Indented));
            }
        });
    }

    private void LoadOrAppendQAItemsFromJson(bool append)
    {
        TryCatchWrapperMethod(() =>
        {
            var dialog = new OpenFileDialog
            {
                Filter = "JSON files (*.json)|*.json"
            };

            if (dialog.ShowDialog() ?? false)
            {
                LoadQAItemsFromJsonFile(dialog.FileName, append);
            }
        });
    }

    private void LoadOrAppendQAItemsFromImages(bool append)
    {
        TryCatchWrapperMethod(() =>
        {
            using (var dialog = new System.Windows.Forms.FolderBrowserDialog())
            {
                dialog.Description = "Please, select folder with questions images.";
                if (dialog.ShowDialog() != System.Windows.Forms.DialogResult.OK)
                {
                    return;
                }
                string qFolder = dialog.SelectedPath;

                dialog.Description = "Please, select folder with answers images. They should be named same to questions. Multi answer should starts
with name of the question.";
                if (dialog.ShowDialog() != System.Windows.Forms.DialogResult.OK)
                {
                    return;
                }
                string aFolder = dialog.SelectedPath;

                var supportedExtensions = new string[] { ".jpg", ".png", ".bmp" };
                IEnumerable<string> qPaths = Directory
                    .GetFiles(qFolder)
                    .Where(p => supportedExtensions.Contains(Path.GetExtension(p)))
                    .Distinct();
                IEnumerable<string> aPaths = Directory
                    .GetFiles(aFolder)
                    .Where(p => supportedExtensions.Contains(Path.GetExtension(p)))
                    .Distinct();

                var qPath_to_qaItem = new Dictionary<string, QAItem>();
                using (OcrApi translator = OcrApi.Create())
                {
                    switch (_settingsModel.ImageRecognitionLanguage)
                    {
                        case Enums.RecognitionLanguage.EN:
                            translator.Init(Languages.English);
                            break;
                        case Enums.RecognitionLanguage.UA:
                            translator.Init(Languages.Ukrainian);
                            break;
                    }

                    foreach (string qPath in qPaths)
                    {
                        if (!qPath_to_qaItem.TryGetValue(qPath, out QAItem qaItem))
                        {
                            qaItem = new QAItem
                            {
                                Question = translator.GetTextFromImage(qPath)
                            };
                            qPath_to_qaItem.Add(qPath, qaItem);
                        }
```

```
                string qFileName = Path.GetFileNameWithoutExtension(qPath);

                foreach (string aPath in aPaths)
                {
                    string aFileName = Path.GetFileNameWithoutExtension(aPath);
                    if (aFileName.StartsWith(qFileName))
                    {
                        qaItem.Answers.Add(translator.GetTextFromImage(aPath));
                    }
                }
            }
        }

        GroupAndAssignQAItems(qPath_to_qaItem.Values, append);
    }
});
}

private void SettingsAction()
{
    new SettingsWindow(new SettingsViewModel(_settingsModel)).ShowDialog();
    RaisePropertyChanged(nameof(QAItemsFiltered));
}

private bool SearchWordPartsInText(string text, IList<string> origin_wordParts_toSearch, out List<Run> highlightableTextParts)
{
    highlightableTextParts = new List<Run>();
    char[] punctuation = _settingsModel.Punctuation.Concat(new char[] { ' ' }).ToArray();
    var currentText = new StringBuilder(text);
    IEnumerable<string> notFoundParts = origin_wordParts_toSearch;
    var currentTextPart = new StringBuilder();

    while (currentText.Length > 0 && notFoundParts.Any())
    {
        char f = currentText[0];

        if (punctuation.Any(p => p == f))
        {
            currentTextPart.Append(f);
            currentText.Remove(0, 1);
            continue;
        }
        else
        {
            if (currentTextPart.Length > 0)
            {
                highlightableTextParts.Add(new Run(currentTextPart.ToString()));
                currentTextPart.Clear();
            }

            int indexOfNextPunctuation = currentText.ToString().IndexOfAny(punctuation);
            indexOfNextPunctuation = indexOfNextPunctuation >= 0 ? indexOfNextPunctuation : currentText.Length;

            string currentWord = currentText.ToString().Substring(0, indexOfNextPunctuation);
            string currentSearchPart = notFoundParts.First();

            int indexOfMatchStart = currentWord.IndexOf(currentSearchPart, StringComparison.InvariantCultureIgnoreCase);
            if (indexOfMatchStart >= 0)
            {
                notFoundParts = notFoundParts.Skip(1);
                currentText.Remove(0, indexOfNextPunctuation);
                currentTextPart.Clear();

                string left = currentWord.Substring(0, indexOfMatchStart);
                string mid = currentWord.Substring(indexOfMatchStart, currentSearchPart.Length);
                string right = currentWord.Substring(indexOfMatchStart + currentSearchPart.Length);

                if (!string.IsNullOrEmpty(left))
                {
                    highlightableTextParts.Add(new Run(left));
                }

                if (!string.IsNullOrEmpty(mid))
                {
                    highlightableTextParts.Add(new Run(mid)
                    {
                        Background = _highlightBrush,
                        FontWeight = FontWeights.Bold
                    });
```

```
                }

                if (!string.IsNullOrEmpty(right))
                {
                    highlightableTextParts.Add(new Run(right));
                }
            }
            else
            {
                currentText.Remove(0, indexOfNextPunctuation);
                currentTextPart.Clear();
                highlightableTextParts.Add(new Run(currentWord));
            }
        }
    }

    if (currentTextPart.Length > 0)
    {
        highlightableTextParts.Add(new Run(currentTextPart.ToString()));
    }

    if (currentText.Length > 0)
    {
        highlightableTextParts.Add(new Run(currentText.ToString()));
    }

    if (!notFoundParts.Any())
    {
        if (highlightableTextParts.Any())
        {
            var part = new StringBuilder();
            var newParts = new List<Run>();
            string currentBackground = highlightableTextParts[0].Background?.ToString() ?? string.Empty;
            FontWeight currentFontWeight = highlightableTextParts[0].FontWeight;

            foreach (Run p in highlightableTextParts)
            {
                if ((p.Background?.ToString() ?? string.Empty) == currentBackground)
                {
                    part.Append(p.Text);
                }
                else
                {
                    newParts.Add(new Run(part.ToString()) { FontWeight = currentFontWeight, Background = _highlightBrush.ToString() ==
currentBackground ? _highlightBrush : Brushes.Transparent });
                    part.Clear();
                    part.Append(p.Text);
                    currentBackground = p.Background?.ToString() ?? string.Empty;
                    currentFontWeight = p.FontWeight;
                }
            }

            if (part.Length > 0)
            {
                newParts.Add(new Run(part.ToString()) { FontWeight = currentFontWeight, Background = _highlightBrush.ToString() ==
currentBackground ? _highlightBrush : Brushes.Transparent });
            }

            highlightableTextParts = newParts;
        }

        return true;
    }

    return false;
  }
}


// QuestionViewModel.cs.
public class QuestionViewModel : HighlightableTextViewModel
  {
    public QuestionViewModel(IList<Run> highlightableTextParts, IEnumerable<AnswerViewModel> answers)
      : base(highlightableTextParts)
    {
      Answers = answers;
    }

    public IEnumerable<AnswerViewModel> Answers { get; }
```

```
    }


// SettingsViewModel.cs.
public class SettingsViewModel : BindableBase
    {
        private readonly SettingsModel _model;

        public SettingsViewModel(SettingsModel model)
        {
            _model = model;
        }

        public RecognitionLanguage SelectedRecognitionLanguage
        {
            get => _model.ImageRecognitionLanguage;
            set
            {
                _model.ImageRecognitionLanguage = value;
                RaisePropertyChanged(nameof(SelectedRecognitionLanguage));
            }
        }

        public string Punctuation
        {
            get => string.Join(" ", _model.Punctuation);
            set
            {
                _model.Punctuation = value
                    .Split(' ')
                    .Where(p => !string.IsNullOrWhiteSpace(p))
                    .Distinct()
                    .SelectMany(s => s.ToCharArray())
                    .ToArray();

                RaisePropertyChanged(nameof(Punctuation));
            }
        }

        public KeyWordsSearchType KeyWordsSearchType
        {
            get => _model.KeyWordsSearchType;
            set
            {
                _model.KeyWordsSearchType = value;
                RaisePropertyChanged(nameof(KeyWordsSearchType));
            }
        }
    }


// Models.QAItems.cs.
public class QAItem
    {
        public string Question { get; set; } = string.Empty;

        public List<string> Answers { get; set; } = new List<string>();
    }


// SettingsModel.cs.
public class SettingsModel
    {
        public RecognitionLanguage ImageRecognitionLanguage { get; set; } = RecognitionLanguage.EN;

        public char[] Punctuation { get; set; } = new char[] { '.', ',', '!', '?', ':', ';', '-', '(', ')' };

        public KeyWordsSearchType KeyWordsSearchType { get; set; } = KeyWordsSearchType.Both;
    }


// Enums.KeyWordSearchType.cs.
public enum KeyWordsSearchType
    {
        [Description("Questions")]
        Questions,
        [Description("Answers")]
        Answers,
        [Description("Questions and Answers")]
```

```
        Both
    }


// Enums.RecognitionLanguage.cs.
public enum RecognitionLanguage
    {
        [Description("English")]
        EN,
        [Description("Ukrainian")]
        UA
    }
```