Міністерство освіти і науки України

Український державний університет науки і технологій

Факультет «Комп’ютерні технології і системи»

Кафедра «Комп’ютерні інформаційні технології»

Пояснювальна записка

до кваліфікаційної роботи

ОС Бакалавр

на тему «Розробка програми для формування Word документів за шаблонами»

за освітньою програмою «12 Інженерія програмного забезпечення»

зі спеціальності «121 Інженерія програмного забезпечення»

Виконала: студентка групи ПЗ2012: \_\_\_\_\_\_\_ /Вероніка БАГНЮК/

(підпис)

Керівник: \_\_\_\_\_\_\_ /Олександр ЖЕВАГО/

(підпис)

Нормоконтролер: \_\_\_\_\_\_\_ /Світлана ВОЛКОВА /

(підпис)

Засвідчую, що у цій роботі немає

запозичень з праць інших авторів

без відповідних посилань.

Студент

(підпис)

Дніпро – 2024

Ministry of Education and Science of Ukraine

Ukrainian State University of Science and Technologies

Faculty «Computer technologies and systems»

Department «Computer information technology»

Explanatory Note

to Bachelor's Degree

on the topic: «Development of a program for creating Word documents based on templates»

according to educational curriculum «12 Software engineering»

in the Speciality: «121 Software engineering»

Done by the student of the group PZ2012: \_\_\_\_\_\_\_ /Veronika BAHNIUK/

(підпис)

Scientific Supervisor: \_\_\_\_\_\_\_ /Oleksandr SZEVAHO/

(підпис)

Normative controller: \_\_\_\_\_\_\_ /Svitlana VOLKOVA/

(підпис)

Dnipro –2024

Міністерство освіти і науки України

Український державний університет науки і технологій

Факультет: «Комп'ютерні технології і системи»

Кафедра: «Комп'ютерні інформаційні технології»

Рівень вищої освіти: бакалавр

Освітня програма: «12 Інженерія програмного забезпечення»

Спеціальність: «121 Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ

Завідувач кафедри КІТ

/Вадим ГОРЯЧКІН/

(підпис)

Дата

**ЗАВДАННЯ**

на кваліфікаційну роботу Бакалавр

студенту Багнюк Вероніка Валеріївна

1. Тема роботи: «Розробка програми для формування Word документів за шаблонами»
2. Керівник роботи: Жеваго Олександр Олександрович
3. Вхідні дані до роботи:
4. Зміст пояснювальної записки:
   1. Вступ;
   2. Збір та аналіз вимог;
   3. Проєктування;
   4. Розробка програми
   5. Тестування та налагодження;
   6. Аналіз та висновки;
   7. Література.
5. Перелік демонстраційного матеріалу:
   1. Доповідь;
   2. Презентація;
   3. Демонстраційне відео.

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |  |
| --- | --- | --- | --- |
| №  п/п | Назва етапів кваліфікаційної роботи | Строк виконання етапів роботи | Примітка |
| 1 | Вступ | 01.03.2024 – 03.03.2024 |  |
| 2 | Збір та аналіз вимог | 04.03.2024 – 10.03.2024 |  |
| 3 | Проєктування (зовнішнє проєктування) | 11.03.2024 – 24.03.2024 |  |
| 4 | Проєктування (внутрішнє проєктування) | 25.03.2024 – 10.03.2024 |  |
| 5 | Розробка програми |  |  |
| 6 | Тестування та налагодження |  |  |
| 7 | Аналіз та висновки |  |  |
| 8 | Подання кваліфікаційної роботи до кафедри |  |  |
| 9 | Захист кваліфікаційної роботи на засіданні Екзаменаційної комісії |  |  |

Студент Вероніка БАГНЮК

(підпис) (Ім’я ПРІЗВИЩЕ)

Керівник роботи: Олександр ЖЕВАГО

(підпис) (Ім’я ПРІЗВИЩЕ)

**РЕФЕРАТ**

Пояснювальна записка до кваліфікаційної роботи бакалавра:

(рівень освіти)

00 с., 0 рис., 0 табл., 0 додатки, 0 джерел

Об’єкт розробки — програма для формування Word документів за шаблонами.

Мета роботи — розробка програми для поліпшення та спрощення процесу створення Word документів на основі шаблонів.

Методи дослідження — аналіз стану предметної області та аналогів створюваного програмного забезпечення, дослідження програмних інструментів та технологій, пов’язаних з роботою з Word документами.

Результати роботи — настільний додаток для операційної системи Windows для формування Word документів за шаблоном. Розроблений додаток може забезпечити часткову автоматизацію ведення документації на основі шаблону, який містить закладки-розмітки для вставлення звичайного тексту, таблиці або зображення. Наприклад, щоб сформувати та заповнити такі документи як силабус або робочу програму.

Ключові слова: WORD ДОКУМЕНТ, ШАБЛОН, ЗАКЛАДКИ, КОМАНДИ ПЕРЕТВОРЕННЯ, НАСТІЛЬНИЙ ДОДАТОК

**ЗМІСТ**

[**ВСТУП** 7](#_Toc167290064)

[**1 ЗБІР ТА АНАЛІЗ ВИМОГ** 8](#_Toc167290065)

[1.1 Опис аналогів 8](#_Toc167290066)

[1.2 Постановка задачі 12](#_Toc167290067)

[**2 ПРОЄКТУВАННЯ** 13](#_Toc167290068)

[2.1 Зовнішнє проєктування 13](#_Toc167290069)

[2.1.1 Функціональне та експлуатаційне призначення 13](#_Toc167290070)

[2.1.2 Функціональні вимоги 13](#_Toc167290071)

[2.1.3 Вхідні дані 15](#_Toc167290072)

[2.1.4 Вихідні дані 17](#_Toc167290073)

[2.2 Внутрішнє проєктування 18](#_Toc167290074)

[2.2.1 Проєктування архітектури програми 18](#_Toc167290075)

[2.2.2 Проєктування інтерфейсу користувача 36](#_Toc167290076)

[2.2.3 Проєктування бази даних 45](#_Toc167290077)

[**2** **РОЗРОБКА ПРОГРАМИ** 46](#_Toc167290078)

[**3** **ТЕСТУВАННЯ ТА НАЛАГОДЖЕННЯ** 47](#_Toc167290079)

[**5** **АНАЛІЗ ТА ВИСНОВКИ** 48](#_Toc167290080)

[**БІБЛІОГРАФІЧНИЙ СПИСОК** 49](#_Toc167290081)

[**ДОДАТОК A** 50](#_Toc167290082)

[**ДОДАТОК Б** 68](#_Toc167290083)

[**ДОДАТОК В** 72](#_Toc167290084)

[**ДОДАТОК Г** 128](#_Toc167290085)

# **ВСТУП**

**Актуальність роботи.** Для роботи організації, підприємства або установи будь-якого виду діяльності важливе місце в управлінні займає діловодство. Діловодство — сукупність процесів, що забезпечують документування управлінської інформації і організацію роботи із службовими документами [1]. Від ступеня правильності оформлення, заповнення конкретних документів та загалом забезпечення якості й своєчасності створення документації залежить успішність результату управління вище згаданих організаційних структур.

У сучасному світі для створення документів широко використовують текстовий процесор Microsoft Office Word, який є частиною офісного пакету Microsoft Office. Хоча він має досить великий набір інструментів, які дозволяють маніпулювати текстом і редагувати його, створювати таблиці, вставляти зображення, математичні формули тощо, їх використання (навіть з достатнім рівнем компетенції) для об’ємних документів вимагає багато витрат часу.

Отже, існує потреба автоматизації формування документів за певним шаблоном. Вирішення цього завдання пришвидшить процес документування, наддасть змогу зосередитися на суті наповненості документа, полегшивши виконання повторюваних задач.

**Мета роботи.** Метою є розробка програми, яка дозволить формувати Word документи (файли формату .docx) на основі шаблону — іншого документу, де містяться закладки для позначення розташування даних певного типу: тексту, таблиці або зображення — у вихідному документі. Програма надає можливість формування документу як шляхом застосування одного шаблону, так і за допомогою команд перетворення одного шаблону на інший.

**Експлуатаційне призначення.** Програма буде корисна для часткової автоматизації ведення документації діловодами. Таким чином вдасться заощадити час, усунувши необхідність вручну створювати кожен документ з нуля, зменшити кількість помилок, забезпечити дотримання правил стандартизації документа відповідного типу.

# **1 ЗБІР ТА АНАЛІЗ ВИМОГ**

## **Опис аналогів**

**Microsoft Office Word** (або Word) – текстовий процесор, розроблений компанією Microsoft, є частиною офісного пакету Microsoft Office. Має широкий набір вбудованих шаблонів та надає можливість користувачеві створювати власні (рисунок 1.1).

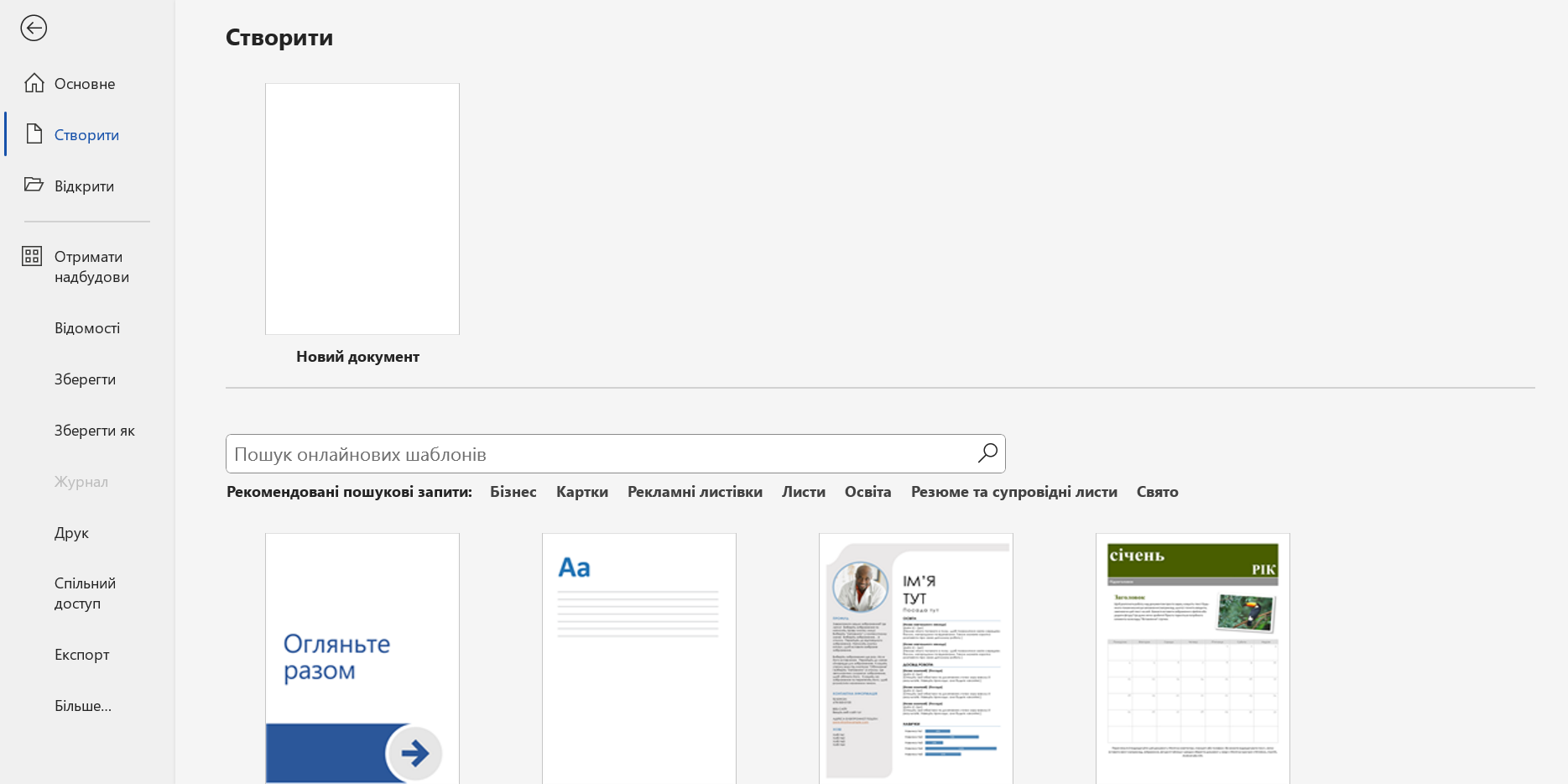


Рисунок 1.1 — Меню з шаблонами Word

Переваги:

* зручний інтерфейс;
* велика кількість вбудованих шаблонів на різну тематику;
* створення шаблону на основі документа з будь-якими підтримуваними структурними елементами.

Недоліки:

* заповнення шаблону вручну;
* відсутність перетворення одного шаблону на інший.

**Apryse WebViewer** (або WebViewer) — бібліотека JavaSript на стороні клієнта, яка дозволяє переглядати, коментувати, редагувати PDF-файли, документи Office та файли інших форматів [2]. Apryse WebViewer виконує візуалізацію документа без залучення сервера. Сервер використовується для розміщення файлів та документів JavaSript, однак є можливість завантаження локальних файлів з комп’ютера користувача.

Існує демонстраційна онлайн-версія (рисунок 1.2).

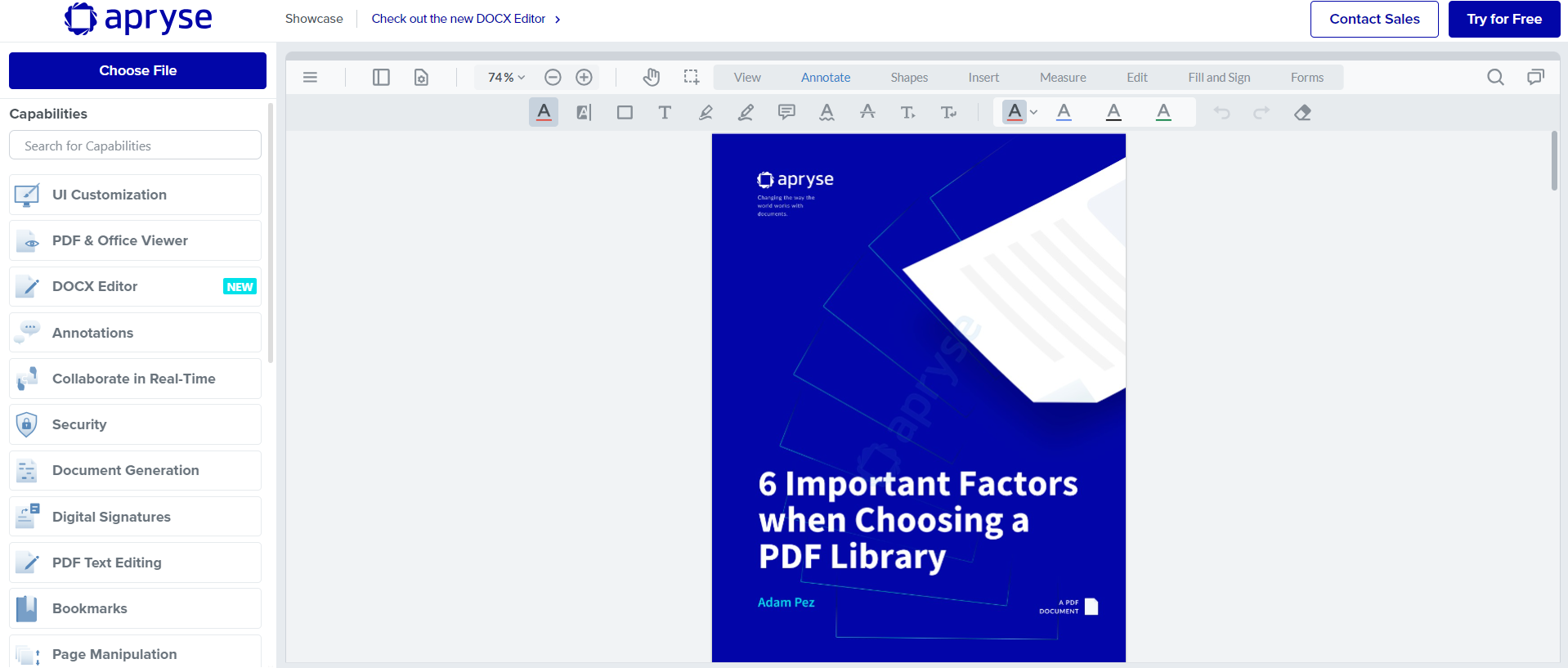


Рисунок 1.2 – Демонстраційна версія Apryse WebViewer

Вона підтримує формування документів за шаблоном з вмістом трьох типів: текст та зображення, динамічні таблиці та додатковий вміст (рисунок 1.3 та рисунок 1.5).

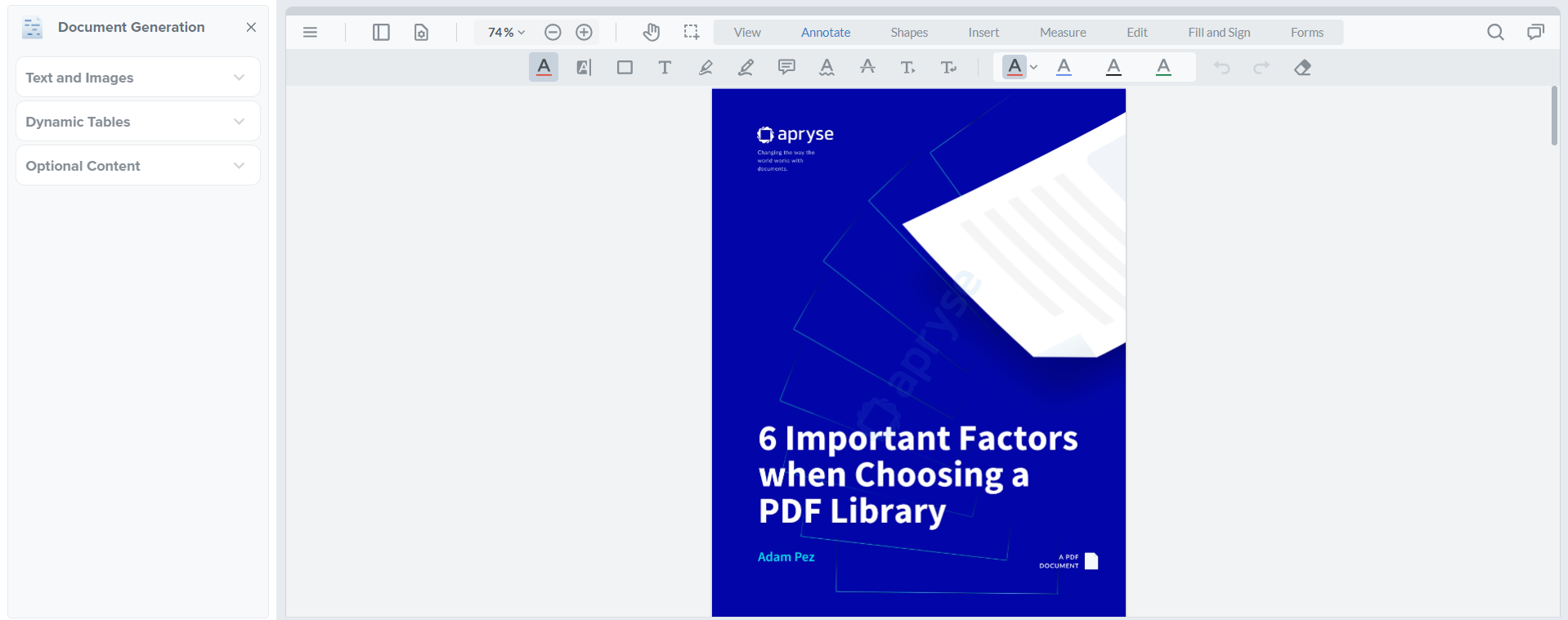


Рисунок 1.3 – Модуль формування документів за шаблоном Apryse WebViewer

Для варіанту текст та зображення підтримується заповнення вбудованого шаблону або завантаження власного файлу, що відповідає формату документів PDF, Microsoft Office (doc, docx, xlsx, pptx) або зображень (jpg, png, tif, bmp) (рисунок 1.4).

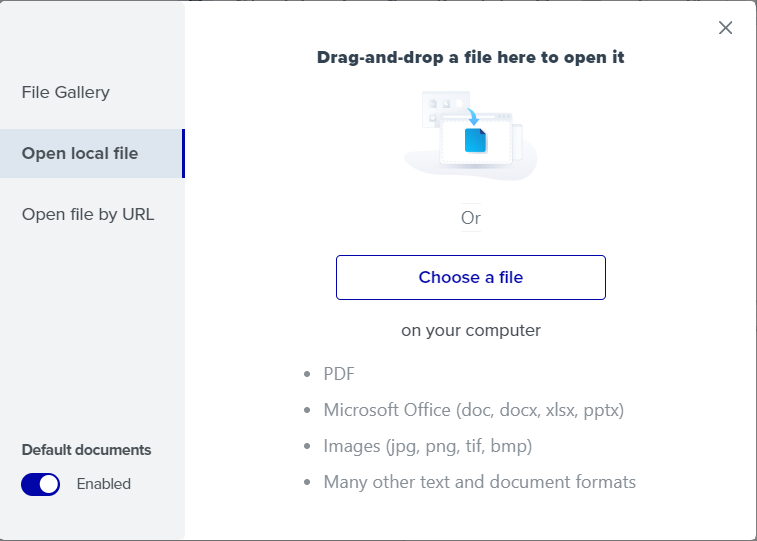


Рисунок 1.4 – Формати шаблонів для формування документа

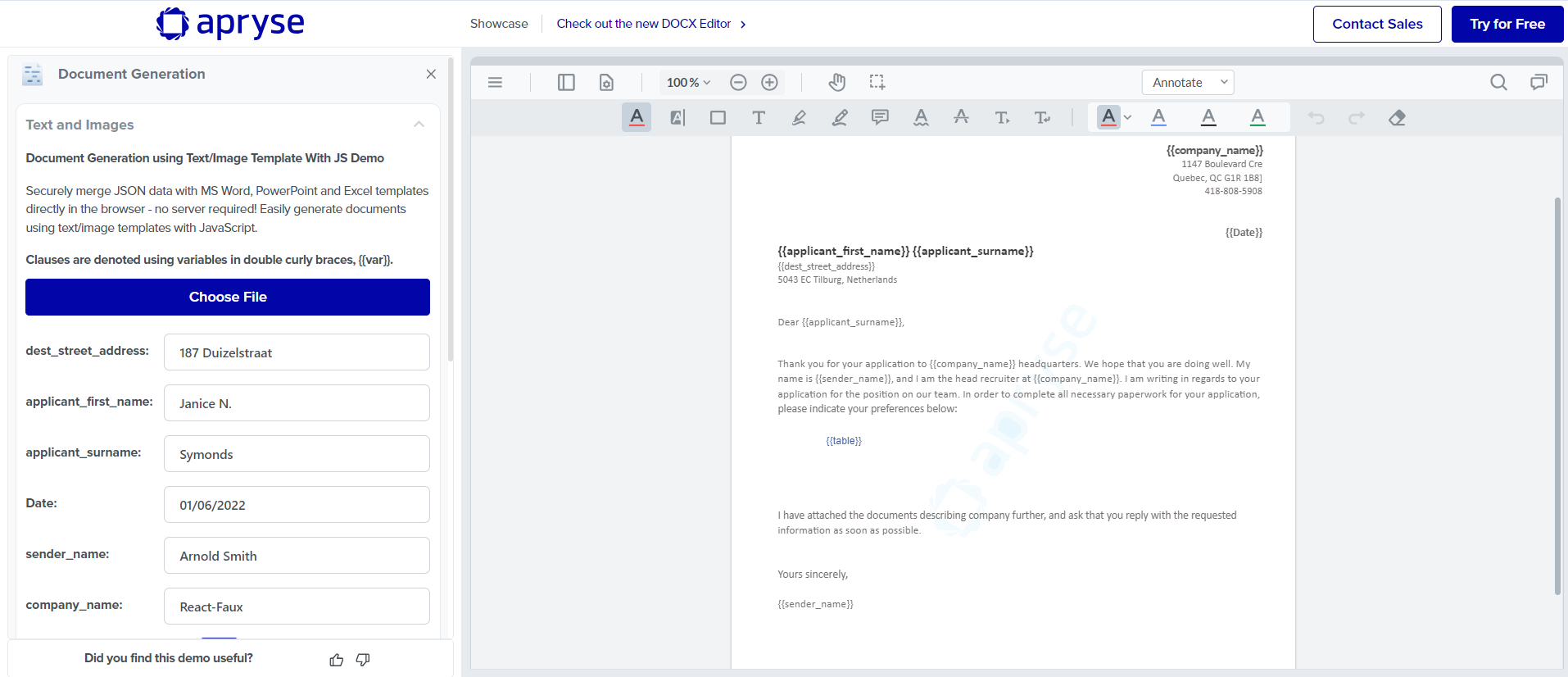


Рисунок 1.5 — Шаблон з категорією вмісту текст та зображення

Переваги:

* підтримка тегів (маркерів) JSON для вхідних даних, що заповнюються;
* динамічне редагування та заповнення шаблону;
* можливість експортувати у декількох форматах: pdf, png, Microsoft Office (doc, docx, xlsx, pptx).

Недоліки:

* неможна завантажувати користувацькі шаблони з таблицями для заповнення;
* неможливо завантажити для збереження більше одного користувацького шаблону;
* неможна завантажити зображення як дані для заповнення.

**Formstack Documents** — це онлайн-платформа для створення документів, яка дозволяє створювати документи кількох типів із динамічним вмістом [3]. Пропонує на вибір три варіанти налаштування шаблону: завантажити з файлу, вибрати з бібліотеки або створити шаблон нуля (рисунок 1.6). На рисунку 1.7 зображено приклад оформлення шаблону.

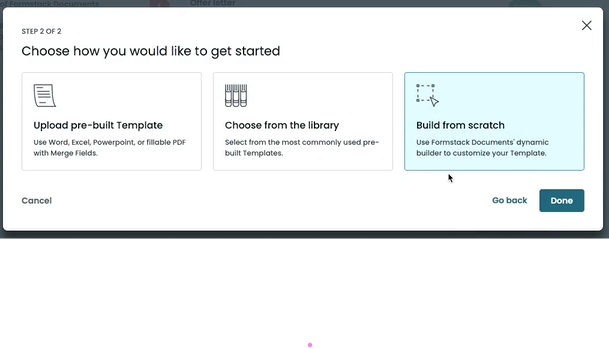


Рисунок 1.6 — Налаштування шаблону Formstack Documents [4]

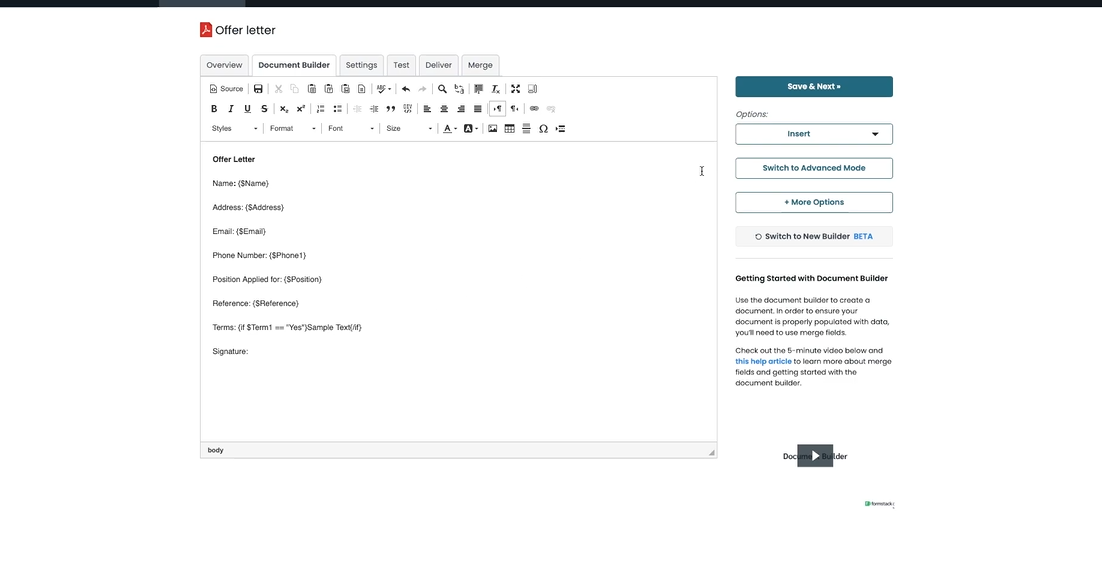


Рисунок 1.7 – Приклад створеного шаблону для Formstack Documents [4]

Переваги:

* динамічне створення шаблону;
* підтримка інтерактивних тегів для даних шаблону;
* можливість надіслати заповнений документ через різні сервіси (наприклад, електронну пошту).

Недоліки:

* комерційна модель поширення;
* відсутність можливості перенесення даних з одного шаблону в інший.

## **Постановка задачі**

Настільний додаток під операційну систему Windows, розроблений на тему «Розробка програми для формування Word документів за шаблонами», має надавати наступні можливості користувачеві:

* завантажувати до програми файли формату .docx, які розмічені закладками та виступають у ролі шаблонів;
* експортувати з програми заповнені шаблони у вигляді файлів формату .docx;
* на основі завантаженого шаблону (.docx файл) створювати заповнений документ, шляхом введення даних для знайдених закладок;
* створювати команди для перетворення одного документа на інший, шляхом завантаження заповненого документа та обрання шаблону вихідного документу.

# **2 ПРОЄКТУВАННЯ**

## **2.1 Зовнішнє проєктування**

### **2.1.1 Функціональне та експлуатаційне призначення**

Функціональне призначення — формування Word документів за шаблонами.

Експлуатаційне призначення — часткова автоматизація введення документації діловодами, що зменшить часові витрати на розробку та заповнення документів.

### **2.1.2 Функціональні вимоги**

Функціональні вимоги наступні:

1. до модуля «Шаблони»:
2. забезпечення можливості додавати (завантажувати) шаблони до програми та відображення інформації про них, а саме: назву, шлях до файлу;
3. забезпечення можливості видаляти шаблони з програми;
4. забезпечення можливості редагувати шаблони, а саме: їхню назву, тип даних закладок — та зберігати ці зміни;
5. забезпечення можливості пошуку шаблону за назвою;
6. забезпечення можливості перегляду закладок вибраного шаблону, а саме: їхньої назви та типу даних (за замовчування – текстовий);
7. до модуля «Команди»:
8. забезпечення можливості створення команд перетворення одного документа в інший, шляхом введення назви команди, задання налаштувань шляхом вибору вхідного документа та вихідного шаблону;
9. забезпечення можливості перегляду команд перетворення одного документа в інший, а саме відображення назви та налаштування команди;
10. забезпечення можливості видалення команд перетворення одного документа в інший;
11. забезпечення можливості редагування команд перетворення одного документа в інший, назви та налаштування команди;

с) до модуля «Генерація»:

1. забезпечення можливості вибору команди для формування Word документу за шаблоном;
2. забезпечення можливості вибрати шаблон зі збережених (в модулі «Шаблони») або завантажити новий;
3. забезпечення можливості відображення закладок шаблону для формування документа;
4. забезпечення можливості зміни типу даних закладок та задання даних відповідно типу (текст, таблиця або зображення);
5. забезпечення можливості введення назви та вибору шляху для експорту вихідного документа;
6. забезпечення можливості сформувати Word документ за шаблоном.

Специфікація функціональних вимог представлена у вигляді діаграми прецедентів на рисунку 2.1.

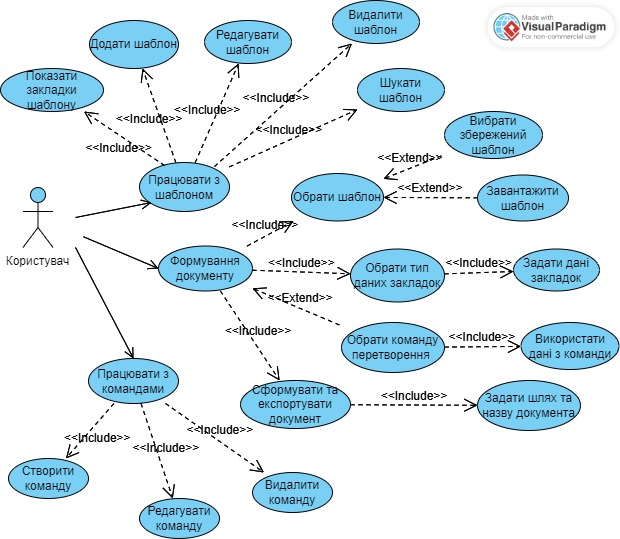


Рисунок 2.1 – Діаграма прецедентів

### **2.1.3 Вхідні дані**

Для модуля шаблонів вхідними даними є:

* шаблон, який представлено у вигляді Word документа (файл формату .docx), що містить закладки. На рисунку 2.2. наведено приклад вмісту шаблону з закладками (показано як символ вертикальної риси з горизонтальною маленькою рисою вгорі та знизу), а на рисунку 2.3. – імена закладок в порядку розташування;
* нова назва шаблону довжиною до 100 символів, яка містить наприкінці позначення формату .docx;
* тип закладки, представлений текстовим рядком з трьох варіантів на вибір: текст, зображення, таблиця.

Для модуля генерації вхідними даними є:

* назва сформованого Word документа, яка представлена у вигляді рядка тексту довжиною до 100 символів та який має наприкінці позначення формату .docx (наприклад, Template.docx) та не може містити наступних символів: \ / : \* ? " < > | ;
* шлях для збереження сформованого Word документа, який представлений у вигляді рядка тексту довжиною до 260 символів, що має починатися з імені пристрою і двокрапки, а після мають йти назви папок, розділені символом \. Наприклад, C:\Documents\Template.docx;
* вхідний шаблон (зі списку збережених або завантажений окремо);
* дані для закладок вхідного шаблону: текст довжиною до 5 тис. символів, зображення (формату .jpg, .png, .gif), таблиця розміром n x m (де n – кількість рядків (1-8), m – кількість рядків (1-8)).
* дані команди (зі списку збережених команд);

Для модуля команд вхідними даними є:

* назва команди, представлена текстовим рядком довжиною до 300 символів;
* вхідний заповнений документ з закладками у вигляді Word документа (файл формату .docx), де закладки позначають об’єкт одного з наступних типів: текст, зображення або таблиця. На рисунку 2.4. наведено приклад вмісту заповненого документа з закладками (показано як квадратні дужки ([ ]), всередині яких є відповідний вміст);
* вихідний шаблон (зі списку збережених);
* налаштування команди, а саме: вихідна закладка шаблону та вхідна закладка документа для вставки у шаблон;

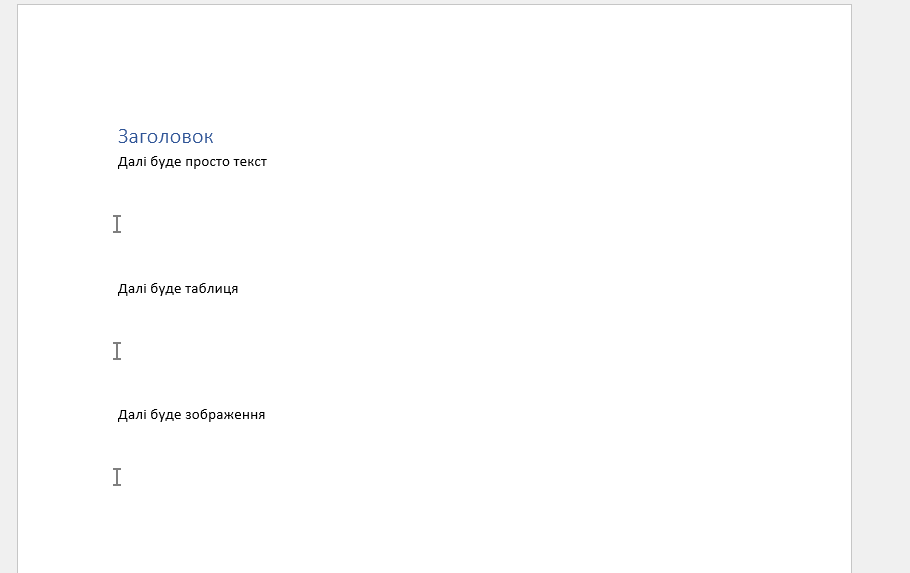


Рисунок 2.2 – Приклад вмісту шаблону з закладками

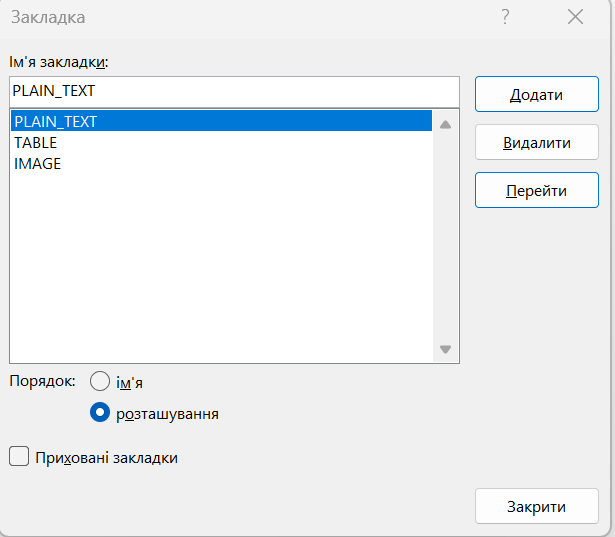


Рисунок 2.3 – Закладки в порядку розташування (див. рис.2.2)

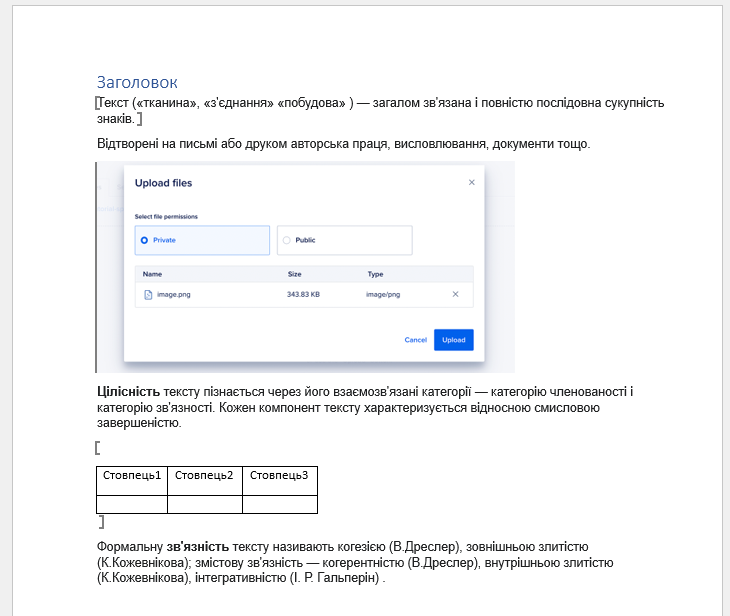


Рисунок 2.4 – Приклад заповненого документа

Примітка: зважаючи на обмеження безкоштовної версії бібліотеки, яка використовувалася для роботи з Word документами, рекомендовано використовувати документи зміст яких не перевищує 500 абзаців та 25 таблиць.

### **2.1.4 Вихідні дані**

Вихідними даними для модуля шаблонів є:

* шаблон, який зберігає наступну інформацію: назву, шлях, зміст та закладки.

Вихідними даними для модуля команд є:

* команда перетворення одного документа на інший, яка містить наступну інформацію: назву, вхідний документ, вихідний шаблон та налаштування команди.

Вихідними даними для модуля генерації є:

* документ формату .docx, який має введену користувачем назву та вибране місце розташування, а також містить задані користувачем дані (текст, таблиця або зображення) на місці відповідних закладок (див. рис.2.4).

## **2.2 Внутрішнє проєктування**

### **2.2.1 Проєктування архітектури програми**

Розроблена програма має багаторівневу архітектуру, де можливо виділити наступні три рівні: рівень представлення (англ. Presentation Layer), рівень бізнес-логіки (англ. Business Layer) та рівень даних (англ. Data Access Layer).

Рівень представлення реалізовує інтерфейс з усіма компонентами, необхідними для забезпечення взаємодії користувача з програмою [5]. Він відображає інформацію та отримує її.

На рівні бізнес-логіки інформація, зібрана на рівні презентації, обробляється – іноді проти іншої інформації на рівні даних – за допомогою бізнес-логіки, конкретного набору бізнес-правил [5].

Рівень даних зберігає та керує інформацією, яку обробляє програма. Працює з реляційною базою даних, виконуючи GRUD (англ. create read update delete; укр. створити читати оновити видалити) операції. Не залежить від рівня представлення та логіки.

На рисунку 2.5. наведено схематичне зображення багаторівневої архітектури.

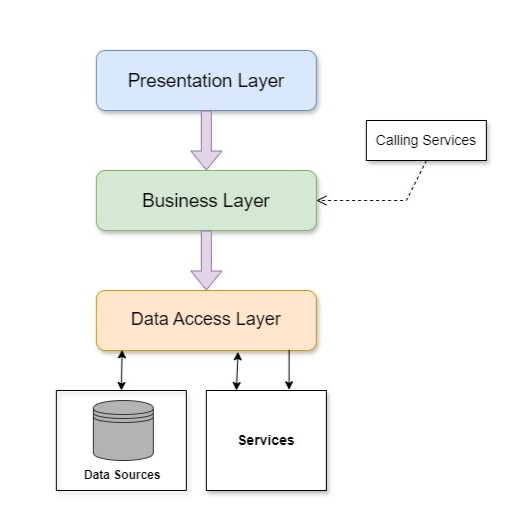


Рисунок 2.5 ­­­­– Схематичне зображення трирівненової архітектури

Для реалізації наведеної архітектури в середовищі розробки виділено п’ять проєктів, а саме:

* Domain – містить моделі предметної області та інтерфейси для рівня доступу до даних;
* Persistence – містить реалізації інтерфейсів, які описані в проєкті Domain;
* Core – містить логіку програми (сервіси для доступу до даних, збережених у сховищі бази даних);
* Presentation – містить елементи патерну MVP (Model-View-Presenter);
* WinFormsUI – містить інтерфейс програми у вигляді Windows Forms компонентів.

На рисунку 2.6 представлено схему взаємодії проєктів та вказано їхню відповідність трирівневій архітектурі.

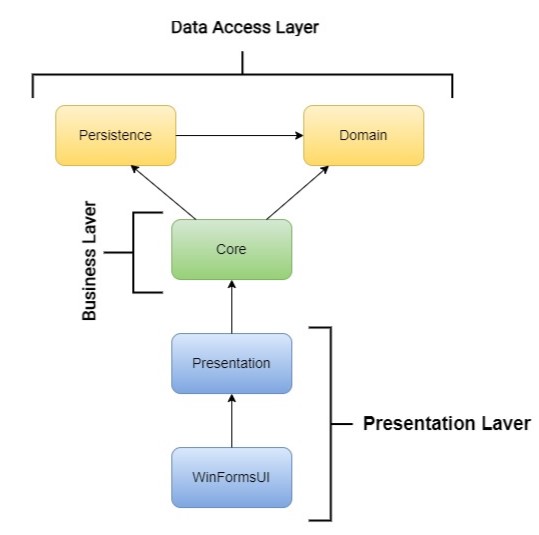


Рисунок 2.6 – Відповідність проєктів трирівневій архітектурі

Патерн MVP (Model-View-Presenter), який використовується для реалізації проєкту Presentation, це архітектурний шаблон, що відокремлює інтерфейс (View) користувача від бізнес-логіки (Model) і вводить третій компонент (Presenter), який є посередником між інтерфейсом та бізнес-логікою.

На рисунку 2.7. наведено схематичне зображення взаємодії елементів MVP.

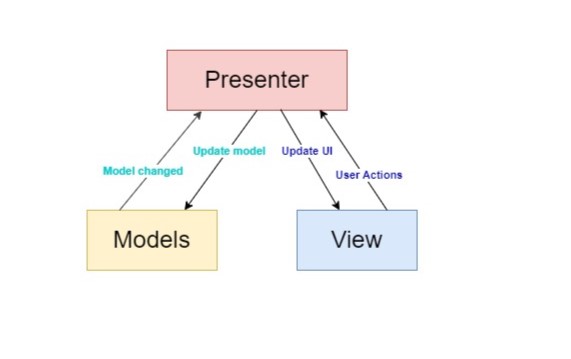


Рисунок 2.7 – Схема взаємодії компонентів MVP

В MVP ролі компонентів розподілені наступним чином:

* Модель (Model) – визначає дані, що використовуються для відображення або виконання дій в користувацькому інтерфейсі. Вона не залежить від представлення та пред’явника;
* Представлення (View) – інтерфейс для відображення даних з моделі і направлення команд користувача (події) до пред’явника для виконання дій на основі переданих даних;
* Пред’явник (Presenter) – реалізує взаємодію між моделлю і представленням. Він отримує дані від моделі для передачі їх до представлення.

Також у програмі реалізовано Dependency Injection (DI) (укр. Впровадження залежностей). DI – це шаблон проєктування, який забезпечує незалежність класу від його залежностей. Це досягається шляхом відокремлення використання об’єкта від його створення [6].

DI має чотири основні ролі:

* Клієнт (Client) – використовує послуги, надані іншим модулем або класом. Не створює та не займається самостійним управлінням залежностей, але покладається на об’єкти ззовні для забезпечення цих залежностей;
* Сервіс або служба (Service) – забезпечує функціональність чи послугу, яка необхідна клієнту. Має незалежність від клієнта, зосереджена на виконанні конкретних функцій;
* Ін’єктор (Injector) – створює екземпляри сервісів та впроваджує їх до клієнта, забезпечуючи його необхідними послугами під час виконання;
* Інтерфейс (Interface) – визначає методи, які повинна реалізувати служба. Клієнти взаємодіють з цим інтерфейсом замість конкретних реалізацій.

Впровадження залежностей має декілька основних переваг:

* слабкий зв’язок, який передбачає спрощення або розширення функціональності без необхідності змін в існуючому коді;
* збільшення гнучкості, оскільки клієнт може взаємодіяти з будь-якими реалізаціями інтерфейсу;
* можливість повторного використання залежностей в іншому коді.

На рисунку 2.8 наведено спрощене схематичне зображення взаємодії компонентів DI.

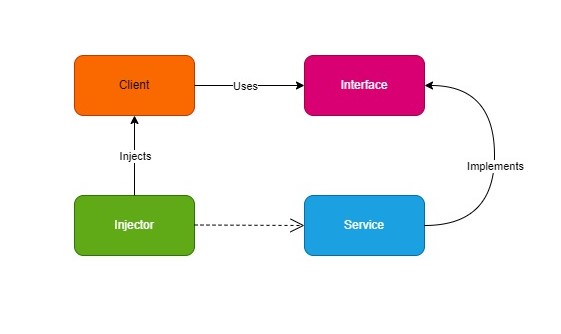


Рисунок 2.8 – Схематичне зображення патерну DI

**Моделювання системи.** Моделювання системи поділяється на кілька етапів, а саме:

1. моделювання словника системи;

Для проєкту Domain:

Ідентифікація сутностей: шаблон, команда, інтерфейс сховища шаблонів, інтерфейс сховища команд.

Ідентифікація обов’язків:

* шаблон – збереження інформації про шаблон;
* команда – збереження інформації про команду перетворення одного документа на інший;
* інтерфейс сховища шаблонів – містить набір методів для реалізації GRUD операцій для сховища шаблонів;
* інтерфейс сховища команд – містить набір методів для реалізації GRUD операцій для сховища команд.

Представлення атрибутів та операцій:

Атрибути та операції, які необхідні для виконання обов’язків кожної з сутностей проєкту Domain, наведено в таблиці 2.1.

Таблиця 2.1 – Сутності, атрибути та методи для проєкту Domain

|  |  |  |
| --- | --- | --- |
| Сутність | Атрибути | Методи |
|  | 4 | 1 |
| Шаблон (Template) | FileName – назва файлу, розміром до 100 символів, тип даних string;  FilePath – шлях до файлу, розміром до 260 символів, тип даних string;  FileContent – зміст файлу, тип даних масив byte;  FileBookmarks – словник закладок (ключ – назва закладки, а значення – тип), тип даних ключа та значення string. | конструктор – ініціалізація атрибутів |
| Команда (Command) | CommandName – назва команди, розміром до 300 символів, тип даних string;  CommandSetting – словник налаштувань (ключ – назва закладки шаблона, значення – назва закладки вхідного документа), тип даних ключа та значення string;  InputTemplate – вхідний документ, тип даних Template;  OutputTemplate – вихідний шаблон, тип даних Template. | конструктор – ініціалізація атрибутів |
|  | 1 | 5 |
| Інтерфейс сховища шаблонів  (ITemplateRepository) | – | Create – створити шаблон;  Get – отримати шаблон;  Update – оновити шаблон;  Save – зберегти шаблон;  Delete – видалити шаблон; |
| Інтерфейс сховища команд  (ICommandRepository) | – | Create – створити команду;  Get – отримати команду;  Update – оновити команду;  Save – зберегти команду;  Delete – видалити команду; |

Для проєкту Persistence:

Ідентифікація сутностей: сховище шаблонів у базі даних, сховище команд у базі даних.

Ідентифікація обов’язків:

* сховище шаблонів у базі даних – реалізація GRUD операцій над сховищем шаблонів (інтерфейс ITemplateRepository);
* сховище команд у базі даних – реалізація GRUD операцій над сховищем команд (інтерфейс ICommandRepository);

Представлення атрибутів та операцій:

Атрибути та операції, які необхідні для виконання обов’язків кожної з сутностей проєкту Persistence, наведено в таблиці 2.2.

Таблиця 2.2 – Сутності, атрибути та методи для проєкту Persistence

|  |  |  |
| --- | --- | --- |
| Сутність | Атрибути | Методи |
|  | 1 | 6 |
| Сховище шаблонів у базі даних (InDatabaseTemplateRepository) | connection – підключення до бази даних SQLite, тип даних SQLiteConnection, модифікатор доступу private; | CreateConnection – створити підключення до бази даних, модифікатор доступу private;  Create – створити шаблон (в пам’яті);  Delete – видалити шаблон з таблиці бази даних;  Get – отримати шаблон з таблиці бази даних;  Save – зберегти (вставити) шаблон у таблицю бази даних;  Update – оновити шаблон у таблиці бази даних; |
| Сховище команд у базі даних (InDatabaseCommandRepository) | connection – підключення до бази даних SQLite, тип даних SQLiteConnection, модифікатор доступу private; | CreateConnection – створити підключення до бази даних, модифікатор доступу private;  Create – створити команду(в пам’яті);  Delete – видалити команду з таблиці бази даних;  Get – отримати команду з таблиці бази даних;  Save – зберегти (вставити) команду у таблицю бази даних;  Update – оновити команду у таблиці бази даних; |

Для проєкту Core:

Ідентифікація сутностей: сервіс (служба) для шаблону, сервіс (служба) для команд. Також додатково зазначимо розширення, яке реєструє залежності між інтерфейсами сховищ та їхніми реалізаціями.

Ідентифікація обов’язків:

* сервіс (служба) для шаблону – реалізує функції доступу до таблиці шаблонів у базі даних;
* сервіс (служба) для команд – реалізує функції доступу до таблиці команд у базі даних;
* розширення – реєструє залежності між інтерфейсами сховищ шаблонів і команд та їхніми реалізаціями з доступом до бази даних.

Представлення атрибутів та операцій:

Атрибути та операції, які необхідні для виконання обов’язків кожної з сутностей проєкту Core, наведено в таблиці 2.3.

Таблиця 2.3 – Сутності, атрибути та методи для проєкту Core

|  |  |  |
| --- | --- | --- |
| Сутність | Атрибути | Методи |
|  | 1 | 7 |
| Сервіс для шаблону  (TemplateService) | templateRepository – сховище шаблонів, тип даних ITemplateRepository, модифікатор доступу private. | конструктор – для ініціалізації атрибута;  GetAllTemplate – отримати всі шаблони;  GetTemplate – отримати конкретний шаблон;  CreateTemplate – створити шаблон;  UpdateTemplate – оновити шаблон;  DeleteTemplate – видалити шаблон;  SaveTemplate – зберегти шаблон. |
| Сервіс для команд  (CommandSevice) | commandRepository – сховище команд, тип даних ICommandRepository, модифікатор доступу private. | конструктор – для ініціалізації атрибута;  GetAllCommand – отримати всі команди;  GetCommand – отримати конкретну команду;  CreateCommand – створити команду; |

Продовження таблиці 2.3

|  |  |  |
| --- | --- | --- |
| Сутність | Атрибути | Методи |
|  | 1 | 7 |
|  |  | DeleteCommand – видалити команду;  SaveCommand – зберегти команду. |
|  | – | 1 |
| Extensions | – | AddInMemoryRepositories – реєстрація залежностей між інтерфейсами сховищ та їх реалізаціями |

Для проєкту Presentation:

Ідентифікація сутностей: модель представлення шаблонів, модель представлення команд, інтерфейс представлення, інтерфейс головного представлення, інтерфейс пред’явника, інтерфейс головного пред’явника, головний пред’явник. Також додатково зазначимо розширення, яке використовує розширення Core, додає сервіси (служби) та реєструє інтерфейс пред’явника та його реалізацію.

Ідентифікація обов’язків:

* модель представлення шаблонів (TemplateViewModel) – визначає дані шаблона для відображення та дій в інтерфейсі користувача;
* модель представлення команд (CommandViewModel) – визначає дані команди для відображення та дій в інтерфейсі користувача;
* інтерфейс представлення (IView) – містить методи для показу і закриття представлення;
* інтерфейс головного представлення (IMainView) – містить методи для відображення даних, атрибути та події для дій користувача;
* інтерфейс пред’явника (IPresenter) – містить метод запуску пред’явника;
* інтерфейс головного пред’явника (IMainPresenter) – інтерфейс-посередник між IPresenter та MainPresenter;
* головний пред’явник (MainPresenter) – керує взаємодією між представленням та сервісами (службами).

Представлення атрибутів та операцій:

Атрибути та операції, які необхідні для виконання обов’язків кожної з сутностей проєкту Presentation, наведено в таблиці 2.4.

Таблиця 2.4 – Сутності, атрибути та методи для проєкту Presentation

|  |  |  |
| --- | --- | --- |
| Сутність | Атрибути | Методи |
|  | 4 | – |
| модель представлення шаблонів (TemplateViewModel) | FileName – назва файлу, розміром до 100 символів, тип даних string;  FilePath – шлях до файлу, розміром до 260 символів, тип даних string;  ContentFile – зміст файлу, тип даних масив byte;  BookmarksFile – словник закладок (ключ – назва закладки, а значення – тип), тип даних ключа та значення string. | – |
| модель представлення команд (CommandViewModel) | NameCommand – назва команди, розміром до 300 символів, тип даних string;  CommandSetting – словник налаштувань (ключ – назва закладки шаблона, значення – назва закладки вхідного документа), тип даних ключа та значення string;  InputTemplate – вхідний документ, тип даних Template;  OutputTemplate – вихідний шаблон, тип даних Template | – |
|  | – | 2 |
| інтерфейс представлення (IView) | – | Show – показати представлення  Close – закрити представлення |
|  | 12 | 3 |
| Інтерфейс головного представлення (IMainView) | viewTemplate – представлення моделі шаблона;  viewNameTemplate – представлення назви шаблона;  SaveTemplate – подія збереження шаблона;  DeleteTemplate – подія видалення шаблона; | SetCommandsList – задати список команд;  SetTemplateList – задати список шаблонів;  SetBookmarksDictionary – задати словник закладок шаблона. |

Продовження таблиці 2.4

|  |  |  |
| --- | --- | --- |
| Сутність | Атрибути | Методи |
|  |  |  |
|  | UpdateTemplate – подія оновлення шаблона;  GetTemplate – подія отримання шаблона;  viewCommand – представлення моделі команди;  viewNameCommand – представлення назви команди;  SaveCommand – подія збереження команди;  DeleteCommand – подія видалення команди;  GetCommand – подія отримання команди;  UpdateCommand – подія оновлення команди; |  |
|  | – | 1 |
| Інтерфейс пред’явника (IPresenter) | – | Run – запуск пред’явника |
|  | 3 | 10 |
| Головний пред’явник (MainPresenter) | mainView – головне представлення;  commandService – сервіс команд;  templateService – сервіс шаблонів; | конструктор – ініціалізація атрибутів;  SaveCommand – подія збереження команди;  DeleteCommand – подія видалення команди;  GetCommand – подія отримання команди;  UpdateCommand – подія оновлення команди;  SaveTemplate – подія збереження шаблона;  DeleteTemplate – подія видалення шаблона;  UpdateTemplate – подія оновлення шаблона;  GetTemplate – подія отримання шаблона;  Run – запуск пред’явника. |

Для проєкту WinFormsUI:

Ідентифікація сутностей: спеціальне вікно повідомлень, програма, форма повідомлення OK, форма повідомлення YesNo, головна форма, форма редагування шаблону, форма редагування команд, форма текстових даних, форма даних зображення, форма табличних даних, табличні дані. Також додатково зазначимо розширення, яке реєструє представлення для головної форми.

Ідентифікація обов’язків:

* спеціальне вікно повідомлень (CustomMessageBox) – діалогове вікно, яке обробляє повідомлення типу «ОК» та «YesNo»;
* програма (Program) – містить вхідну точку програми;
* форма повідомлення OK (FormOK) – форма, яка показує користувачу повідомлення;
* форма повідомлення YesNo (FormYesNo) – форма, яка отримує відповідь «так» або «ні» на запит з повідомлення;
* головна форма (FormMain) – форма, яка містить всі основні елементи керування інтерфейсом та обробляє події до них;
* форма редагування шаблону (FormEditTemplate) – форма, що містить елементи інтерфейсу для редагування шаблона та обробляє події до них;
* форма редагування команди (FormEditCommand) – форма, яка містить елементи інтерфейсу для редагування команд та обробляє події до них;
* форма текстових даних (FormTextData) – форма, що має елементи керування інтерфейсу для введення та збереження тексту;
* форма даних зображення (FormImageData) – форма, яка має елементи керування інтерфейсу для завантаження та збереження зображення;
* форма табличних даних (FormTableData) – форма, що містить елементи керування інтерфейсу для створення, заповнення та збереження таблиці;
* табличні дані (TableData) – клас для збереження даних таблиці, отриманої з FormTableData;
* розширення (extensions) – реєструє представлення для головної форми.

Представлення атрибутів та операцій:

Атрибути та операції, які необхідні для виконання обов’язків кожної з сутностей проєкту WinFormUI, наведено в табл.2.5.

Таблиця 2.5 – Сутності, атрибути та методи для проєкту WinFormsUI

|  |  |  |
| --- | --- | --- |
| Сутність | Атрибути | Методи |
|  | – | 1 |
| спеціальне вікно повідомлень (CustomMessageBox) | – | Show – показати вікно повідомлень. |
|  |  | 2 |
| програма (Program) | – | Main – точка входу в програму;  CreateHostBuilder – створити HostBuilder. |
|  | 1 | – |
| форма повідомлення OK (FormOK) | Message – отримує та задає повідомлення, тип даних string | конструктор форми |
| форма повідомлення YesNo (FormYesNo) |
|  | 6+ | 7+ |
| головна форма (FormMain) | IndexRowCommandTable – індекс рядка таблиці команд, тип даних int;  IndexRowTemplateTable – індекс рядка таблиці шаблонів, тип даних int;  cmbGenTypeDataBookmarks – стовпець типу даних закладки, тип даних DataGridViewComboBoxColumn;  cmdInputDoc – представлення вхідного документа команди;  genInputTemplate – представлення вхідного шаблона для генерації документа;  bookmarksData – словник даних закладок (ключ – назва закладки, значення – дані закладки), тип ключа string, тип даних object;  + атрибути IMainView | конструктор форми;  SetCommandsList – задання списку команд  SetTemplateList – задання списку шаблонів  checkNameTemplate – перевірка назви шаблона  checkIsSameTextBody – перевірка чи закладка в одному текстовому тілі;  GetBookmarksFromDoc – отримати закладки з документа;  CreateTableForWord – створити таблицю для Word;  + обробники подій елементів керування на формі. |
|  | 4 |  |
| форма редагування шаблона (FormEditTemplate) | oldNameTemplate – стара назва шаблону, тип string;  newNameTemplate – нова назва шаблону, тип string; | конструктор форми;  обробники подій елементів керування інтерфейсу; |

|  |  |  |
| --- | --- | --- |
| Сутність | Атрибути | Методи |
|  | 4 |  |
|  | newBookmarksTemplate – нові налаштування закладок шаблона;  SavedChanges – прапорець збережених змін. |  |

Продовження таблиці 2.5

1. моделювання розподілу обов’язків у системі;

Наведемо множини класів та інтерфейсів, які працюють разом, щоб достягнути певної поведінки:

* Template, ITemplateRepository – використання класу Template для опису методів доступу до сховища шаблонів;
* Command, ICommandRepository – використання класу Command для опису методів доступу до сховища команд;
* TemplateService, ITemplateRepostitory – використання інтерфейсу ITemplateRepository для реалізації GRUD операцій над сховищем шаблонів;
* CommandService, ICommandRepository – використання інтерфейсу ICommandRepository для реалізації GRUD операцій над сховищем команд;
* InDatabaseTemplateRepostitory, ITemplateRepostitory – реалізація методів інтерфейсу сховища шаблонів;
* InDatabaseCommandsRepository, ICommadRepository – реалізація методів інтерфейсу сховища команд;
* IMainView, IView – наслідування IView;
* IMainPresenter, IPresenter – наслідування IPresenter;
* MainPresenter, IMainPresenter – реалізація IMainPresenter.

1. моделювання непрограмних сутностей:

Моделювання непрограмних сутностей для цієї програми не потрібне, оскільки апаратні особливості не використовуються. Всі інші сутності представлені у вигляді класів.

1. моделювання примітивних типів, простих залежностей, наслідування та структурних зв’язків:

У таблиці 2.6 наведено моделювання різних зв’язків.

Таблиця 2.6 – Моделювання залежностей

|  |  |  |
| --- | --- | --- |
| **Клас, який зв’язується** | **Клас, з яким зв’язуються** | **Тип зв’язку** |
| *Для проєктів Domain, Persistence, Core* | | |
| Command | Template | Асоціація |
| ITemplateRepository | Template | Залежність |
| ICommandRepository | Template |
| Command |
| InDatabaseTemplateRepository | ITemplateRepository | Реалізація |
| InDatabaseCommandsRepository | ICommandRepository |
| TemplateService | ITemplateRepository | Асоціація |
| CommandService | ICommandRepository |
| *Для проєкту Presentation* | | |
| CommandViewModel | TemplateViewModel | Асоціація |
| IMainPresenter | IPresenter | Узагальнення |
| IMainView | IView | Узагальнення |
| TemplateViewModel | Асоціація |
| CommandViewModel |
| MainPresenter | IMainView | Асоціація |
| IMainPresenter | Реалізація |
| CommandService | Асоціація |
| TemplateService | Асоціація |
| *Для проєкту WinFormsUI* | | |
| CustomMessageBox | FormOK | Асоціація |
| FormYesNo |
| FormTableData | TableData | Асоціація |
| FormMain | FormTableData | Асоціація |
| FormTextData |
| FormImageData |
| CustomMessageBox |
| FormEditTemplate |
| FormEditCommand |
| IMainView | Реалізація |

На рисунку 2.9 представлено діаграму класів для проєктів Domain, Persistence та Core.

На рисунку 2.10 представлено діаграму класів для проєкту Presentation (асоціацію MainPresenter з TemplateService та CommandService прибрано, щоб уникнути перевантаження).

На рисунку 2.11 представлено діаграму класів для проєкту WinFormsUI (реалізацію MainForm з IMainView прибрано, щоб уникнути перевантаження).

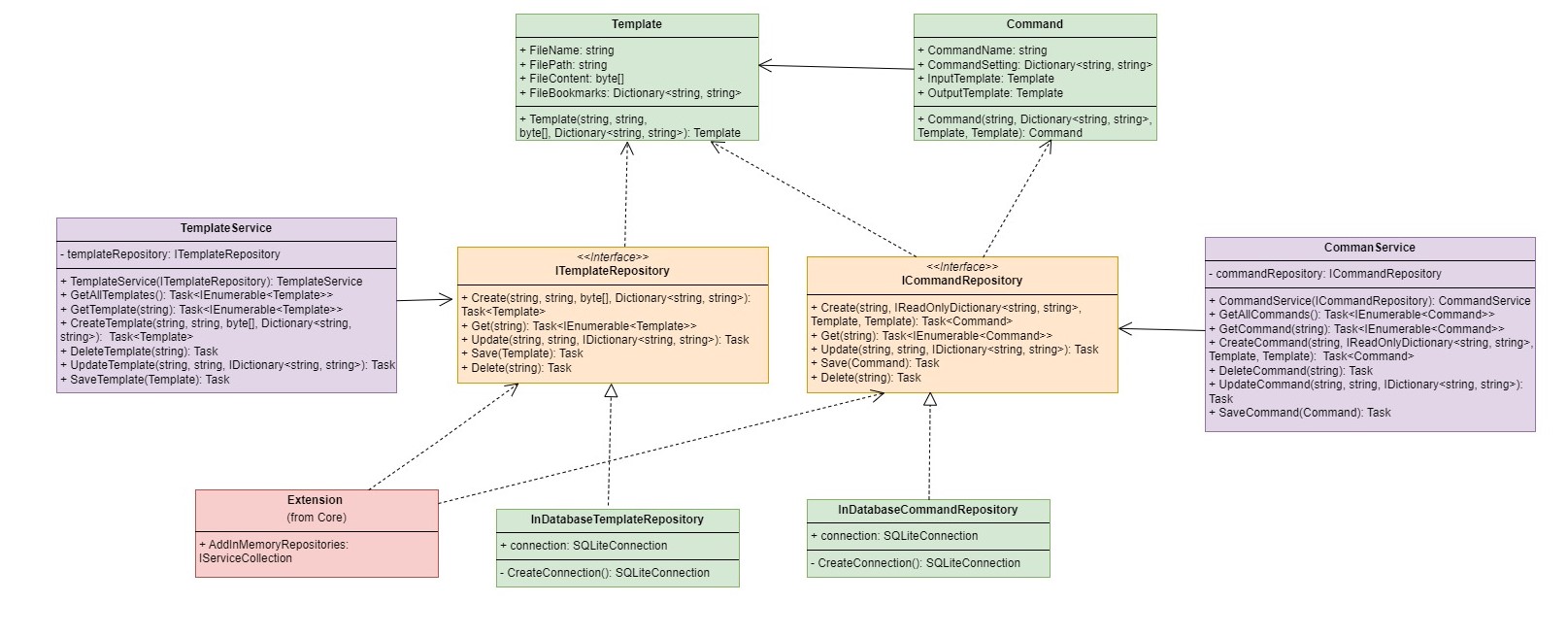


Рисунок 2.9 – Діаграма класів для проєктів Domain, Persistence та Core

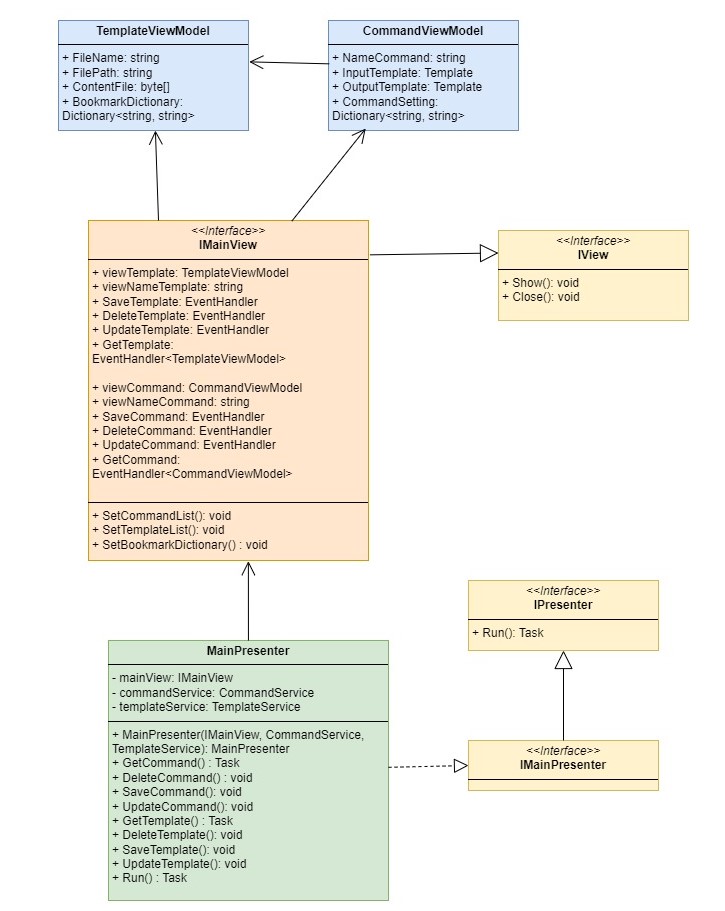


Рисунок 2.10 – Діаграма класів для проєкту Presentation

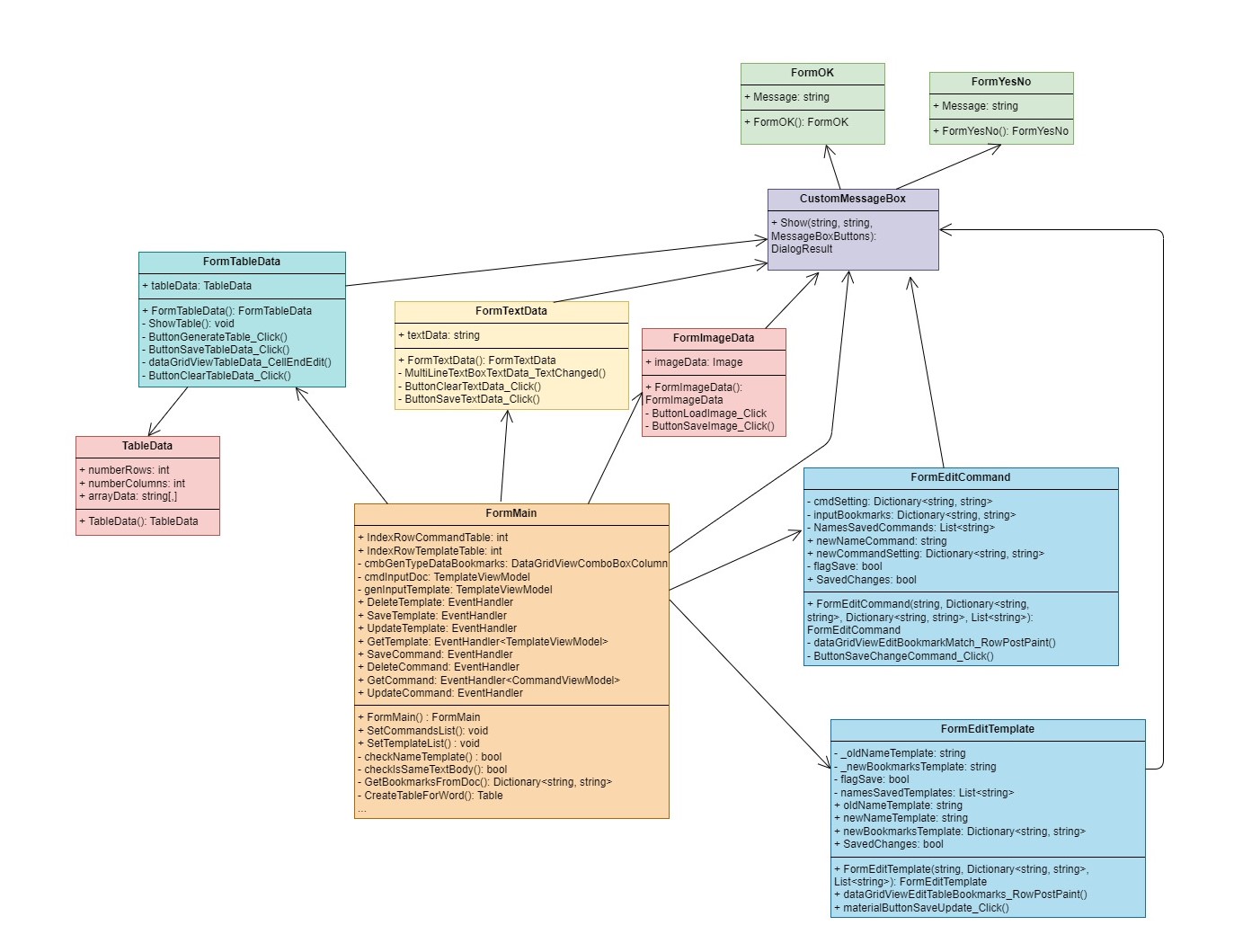


Рисунок 2.11 – Діаграма класів для проєкту WinFormUI

### **2.2.2 Проєктування інтерфейсу користувача**

**Сценарій діалогу.** Сценарій діалогу представлений у вигляді діаграм діяльності, де виділено чотири доріжки: користувач – користувач програми; форма – представляє елементи інтерфейсу, через які виконується взаємодія; контролер – посередник між інтерфейсом та бізнес-логікою програми; база даних – організовує роботи з даними.

Для модуля шаблонів виділено наступні прецеденти:

1. Прецедент: Додавання шаблону;

Ініціатор та належність завершення потоку керування: Користувач.

Перелік об'єктів з поясненнями та їх станами:

* вікно вибору файлу – вікно ОС Windows, яке дозволяє вибрати файл;
* шаблон – Word документ, який розмічений закладками;
* ознака завершення вибору файлу – визначає, чи підтверджено вибір файлу.

Діаграма діяльності для цього прецеденту наведена на рисунку 2.12.

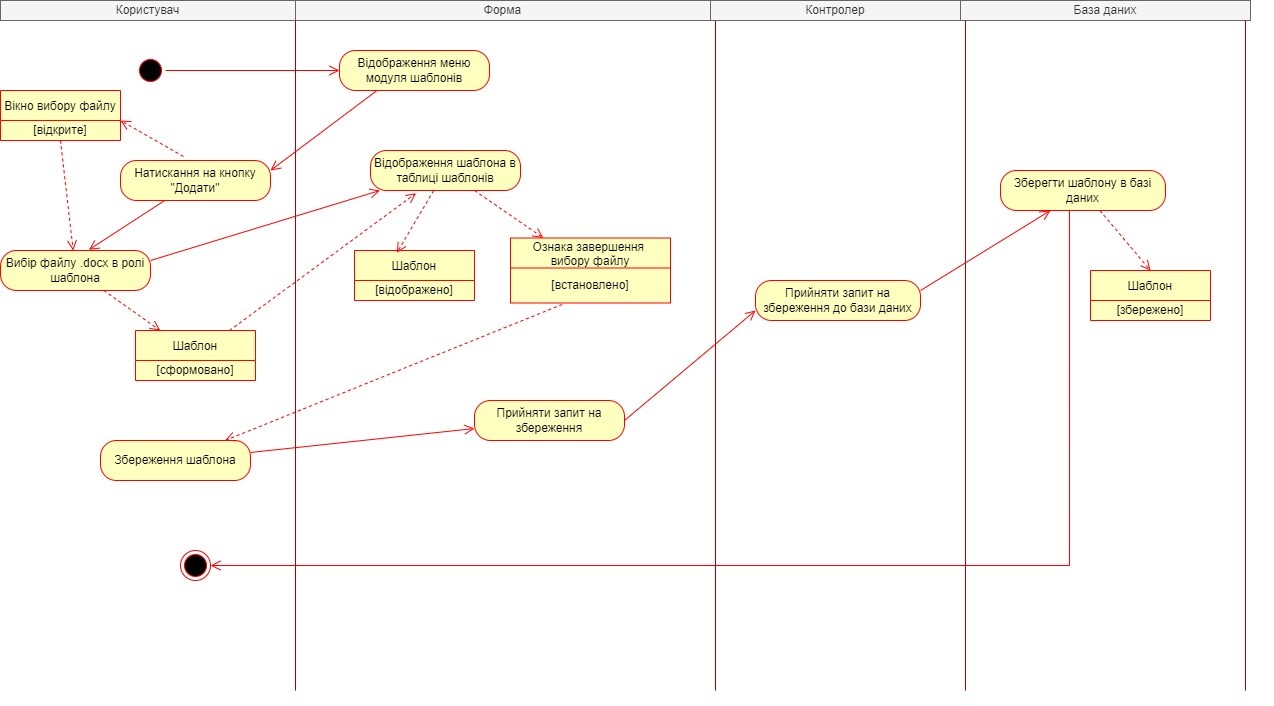


Рисунок 2.12 – Діаграма діяльності для прецеденту «Додавання шаблону» (модуль шаблонів)

1. Прецедент: Редагування шаблону;

Ініціатор та належність завершення потоку керування: Користувач.

Перелік об'єктів з поясненнями та їх станами:

* шаблон – вибраний шаблон зі збережених (Word документ);
* форма редагування шаблону – містить елементи інтерфейсу для редагування даних вибраного шаблону;
* нові дані шаблону – оновлені дані шаблону;
* ознака завершення введення – визначає чи користувач завершив редагувати дані шаблону.

Діаграма діяльності для цього прецеденту наведена на рисунку 2.13.

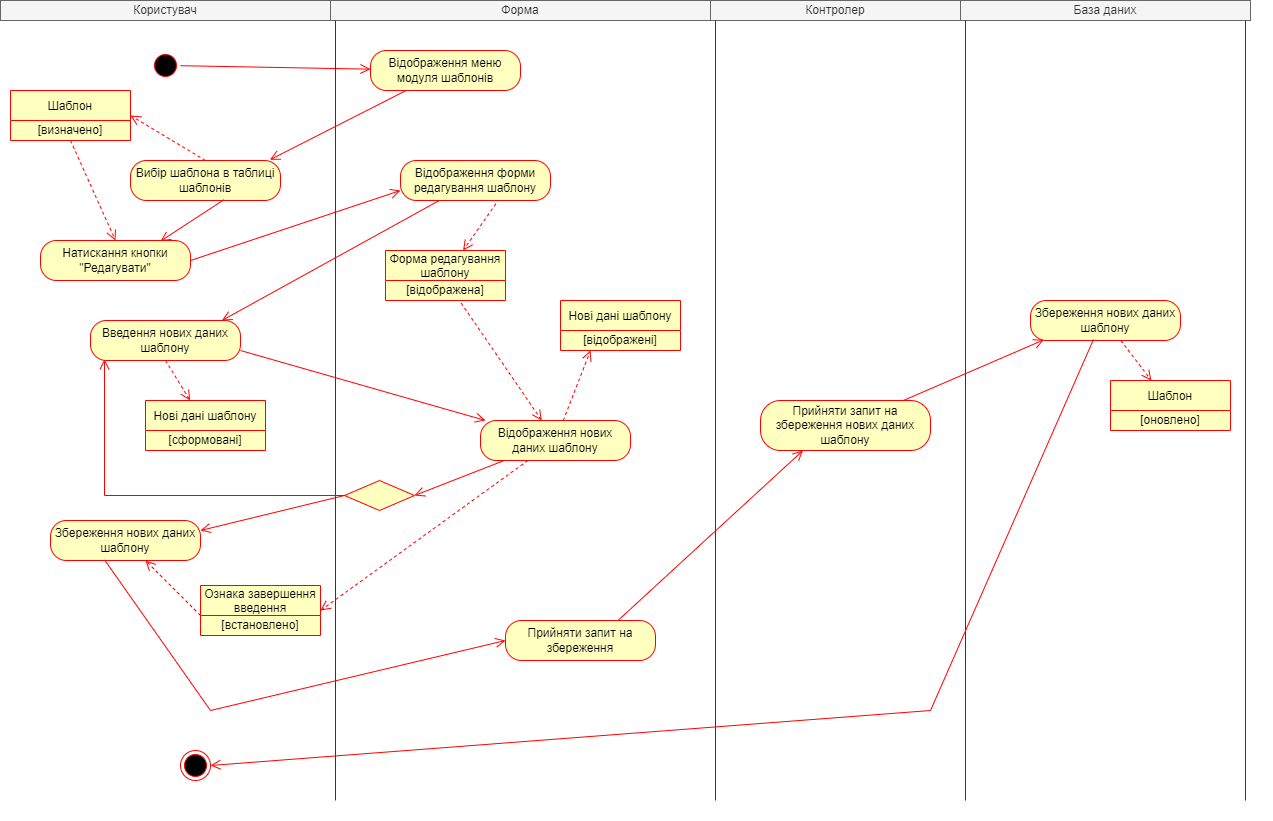


Рисунок 2.13 – Діаграма діяльності для прецеденту «Редагування шаблону» (модуль шаблонів)

1. Прецедент: Видалення шаблону

Ініціатор та належність завершення потоку керування: Користувач.

Перелік об'єктів з поясненнями та їх станами:

* шаблон – вибраний шаблон зі збережених (Word документ);
* форма ТакНі – діалогове вікно, яке приймає відповідь (так або ні) від користувача для підтвердження видалення шаблону;

На рисунку 2.14 приведено діаграму діяльності для цього прецеденту.

Для модуля команд виділено наступні прецеденти:

1. Прецедент: Створення команди.

Ініціатор та належність завершення потоку керування: Користувач.

Перелік об’єктів з поясненнями та їх станами:

* вхідні дані команди – дані, необхідні для створення команди;
* ознака завершення введення – визначає чи завершилося введення вхідних даних команди;
* ознака завершення налаштування – визначає чи завершилося налаштування параметрів конвертування;
* команда – команда для перетворення одного документа на інший.

На рисунку 2.15 наведено діаграму діяльності для цього прецеденту.

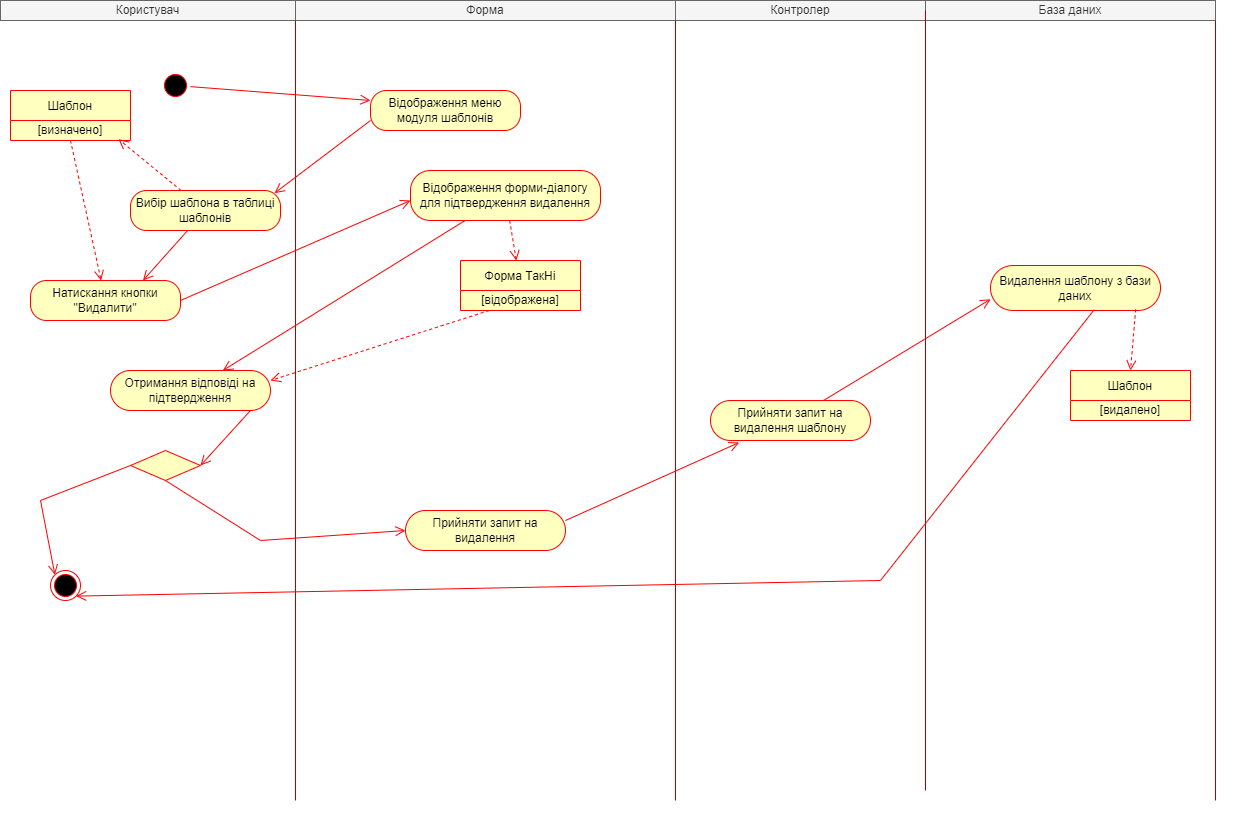


Рисунок 2.14 – Діаграма діяльності для прецеденту «Видалення шаблону» (модуль шаблонів)

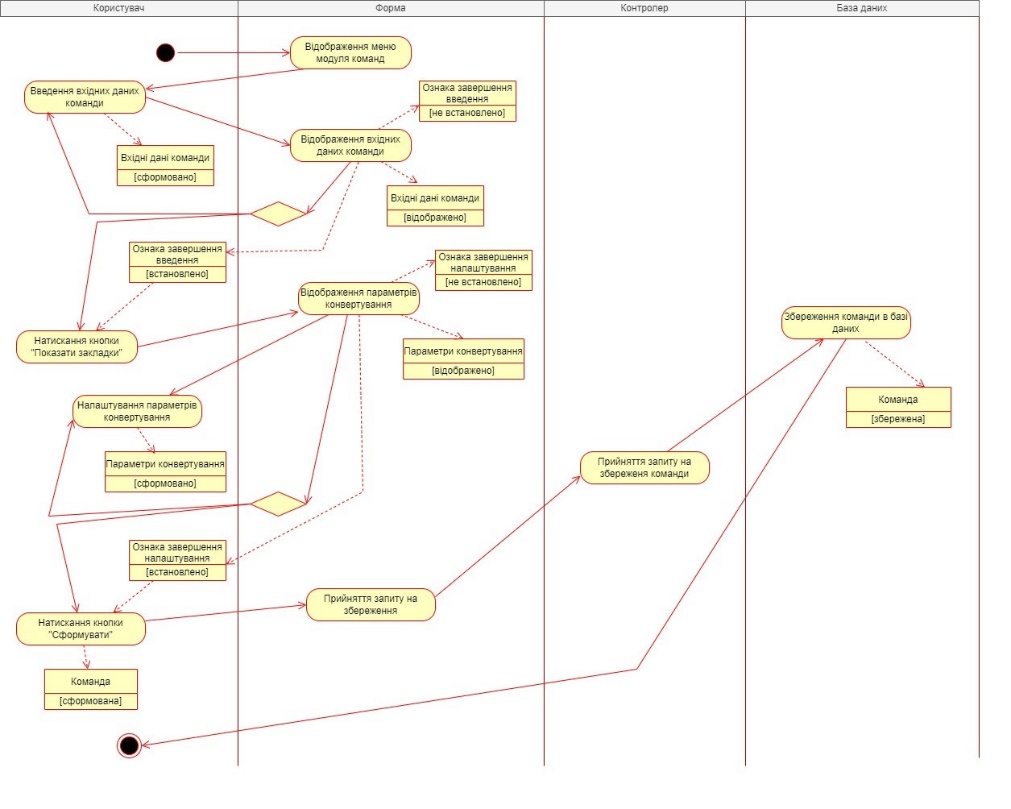


Рисунок 2.15 – Діаграма діяльності для прецеденту «Створення команди» (модуль команд)

Прецеденти «Редагування команди» та «Видалення команди» відбуваються аналогічно відповідним сценаріям для шаблонів, зображених на діаграмах діяльності на рисунках 2.13 – 2.14.

Для модуля генерації виділено наступні прецеденти:

1. Прецедент: Формування документа з використанням даних команди.

Ініціатор та належність завершення потоку керування: Користувач.

Перелік об’єктів з поясненнями та їх станами:

* дані вихідного документа – назва та шлях вихідного документа;
* назва документа – користувацька назва документа;
* дані команди – дані збереженої команди;
* вихідний документ – сформований документ з використанням даних команди.

На рисунку 2.16 наведено діаграму діяльності для цього прецеденту.

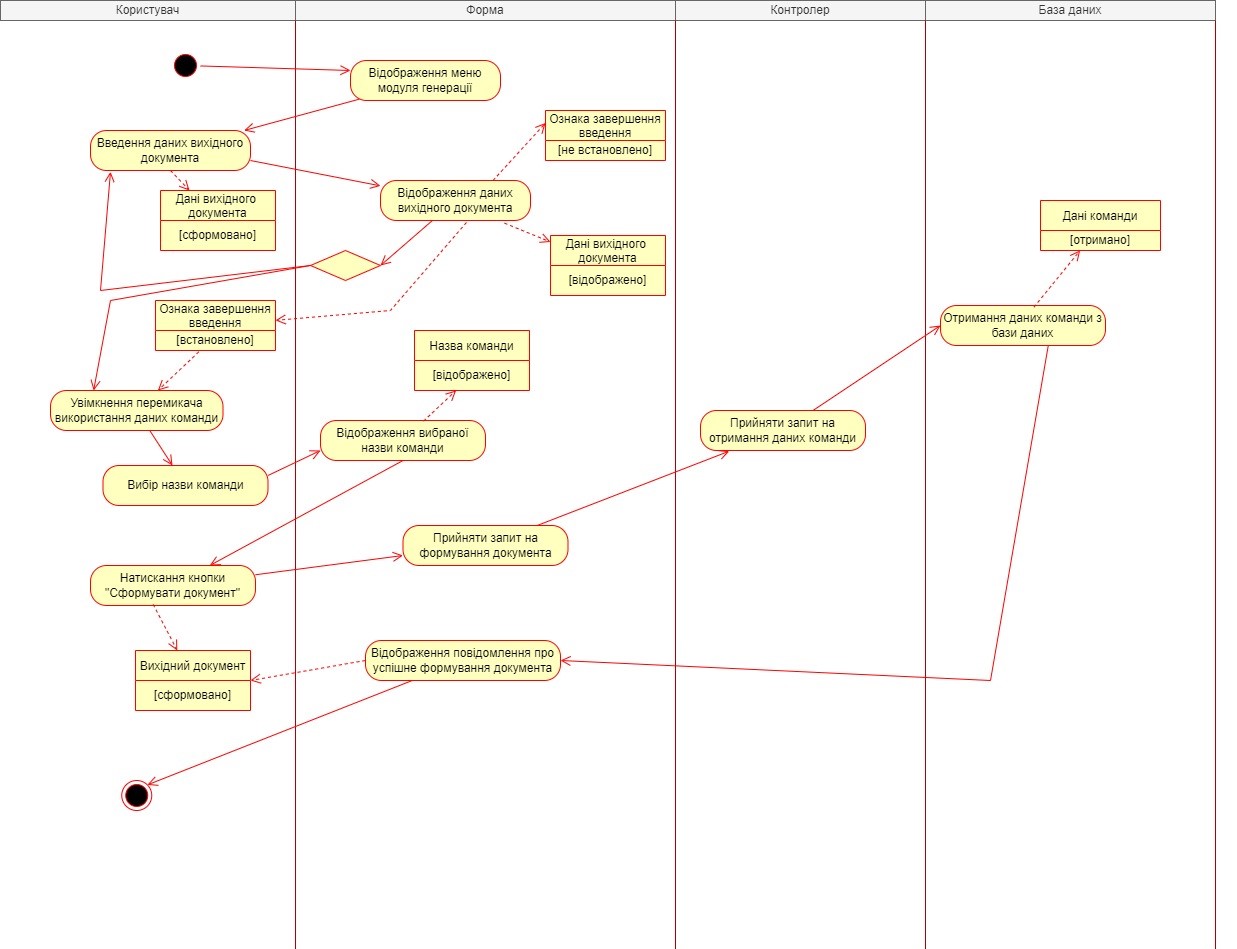


Рисунок 2.16 – Діаграма діяльності для прецеденту «Формування документа з використанням даних команди»

1. Прецедент: Формування документа на основі шаблона.

Ініціатор та належність завершення потоку керування: Користувач.

Перелік об’єктів з поясненнями та їх станами:

* шаблон – вибраний зі збережених (або завантажений шаблон);
* закладки шаблону – закладки, які містить шаблон;
* дані вихідного документа – назва та шлях вихідного документа;
* форма дані – форма для введення даних для закладок;
* дані шаблона – дані шаблона для формування документа;
* вихідний документ – – сформований документ на основі шаблона.

На рисунку 2.17 наведено діаграму діяльності для цього прецеденту.

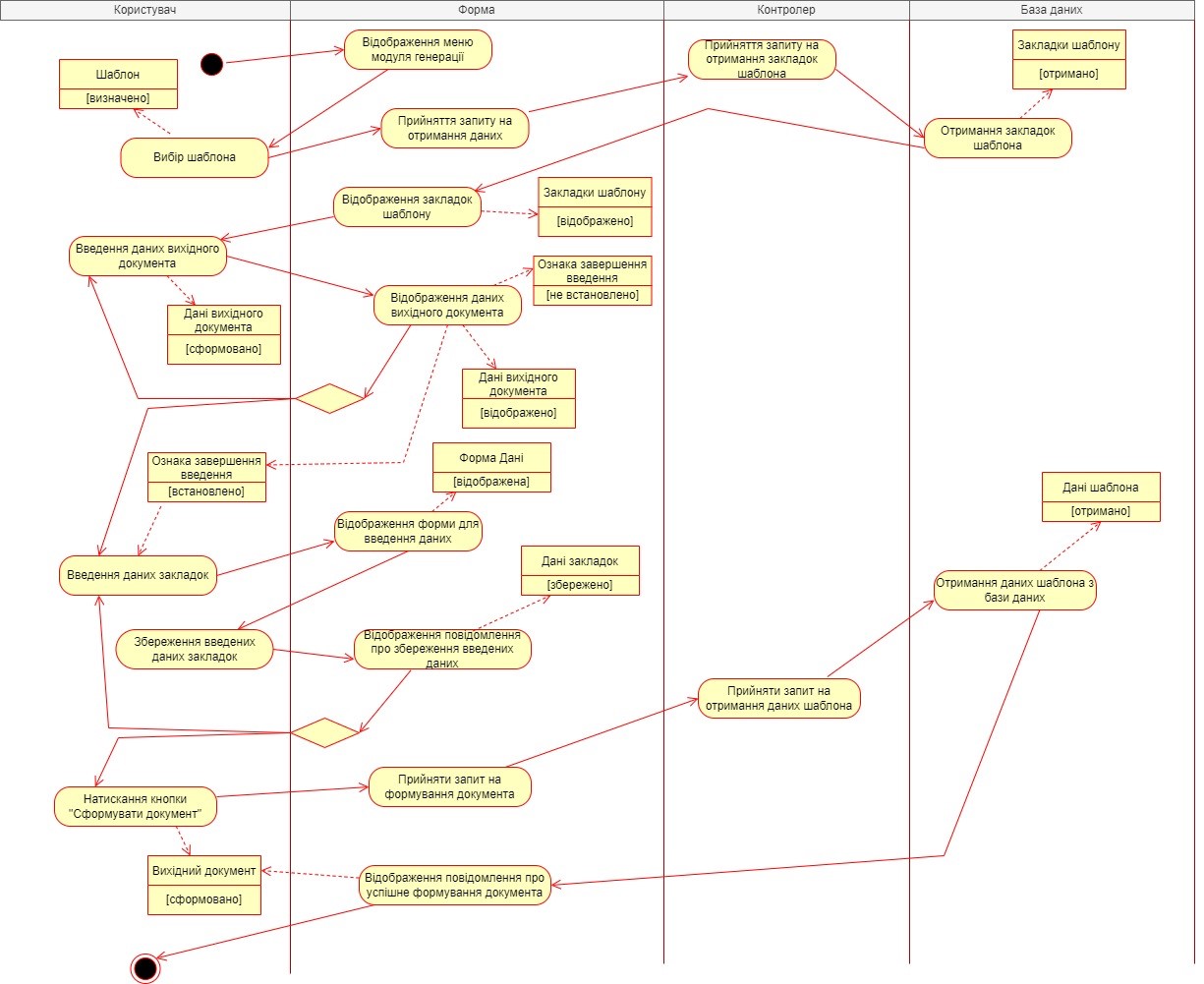


Рисунок 2.16 – Діаграма діяльності для прецеденту «Формування документа на основі шаблона»

**Структура діалогу.** У якості структури діалогу вибрано діалог на основі екранних форм. Головна форма має складатися з трьох вкладок: шаблони, генератор та команди.

Вкладка «Шаблони» має містити наступні елементи інтерфейсу:

* панель з кнопками «Додати», «Редагувати», «Читати» та полем «Знайти шаблон»;
* таблицю шаблонів зі стовпцями: «Номер», «Назва», «Шлях до файлу» та «Видалити» (кнопка-піктограма);
* таблицю закладок зі стовпцями: «Номер», «Назва закладки» та «Тип закладки».

Форма редагування шаблону має містити:

* поле «Назва шаблону»;
* таблицю зі стовпцями: «Номер», «Назва закладки», «Тип закладки» (випадний список з трьох елементів: текст, зображення, таблиця);
* кнопку «Зберегти оновлення».

Вкладка «Команди» має містити такі елементи інтерфейсу:

* поле «Назва команди», «Вхідний документ», «Вихідний шаблон» (випадний список);
* кнопку «Показати закладки»;
* таблицю для параметрів конвертування зі стовпцями: «Номер», «Вихідна закладка (шаблон)», «Вихідна закладка (документ)» (випадний список);
* панель з кнопками «Сформувати», «Редагувати»;
* таблицю команд зі стовпцями: «Номер», «Назва команди», «Вхідний шаблон», «Вихідний шаблон», «Видалити» (кнопка-піктограма).

Форма редагування команди має містити:

* поле «Назва команди»;
* таблицю налаштувань зі стовпцями «Номер», «Вихідна закладка (шаблон)», «Вихідна закладка (документ)» (випадний список);
* кнопка «Зберегти зміни».

Вкладка «Генератор» має містити:

* елемент з вибором вкладок «Збережені шаблони», «Завантажити шаблон»;
* поле «Назва вихідного документа», «Шлях для збереження» та кнопку «Вибрати папку»;
* перемикач «використати дані команди» та випадний список з назвами команд;
* кнопка «Сформувати документ»;
* таблиця закладок шаблона зі стовпцями: «Номер», «Закладка», «Тип даних» (випадний список), «Дані» (кнопка-піктограма).

Форма введення текстових даних має містити:

* багаторядкове поле для тексту;
* кнопки «Зберегти текст» та «Очистити текст».

Форма введення табличних даних має містити:

* числові лічильники «Кількість рядків» та «Кількість стовпців»;
* кнопки «Створити», «Зберегти», «Очистити»;
* таблицю із заданою розмінністю за допомогою числових лічильників.

Форма задання зображення має містити:

* контейнер для зображення;
* кнопки «Завантажити зображення», «Зберегти зображення».

**Перелік повідомлень для користувача.** Для інформування користувача передбачено ряд основних повідомлень, які представлені у таблиці 2.7.

Таблиця 2.7 – Повідомлення для користувача

|  |  |  |
| --- | --- | --- |
| **Текст** | **Ситуація** | **Рекомендовані дії** |
| Шаблон не містить закладок! Додайте закладки до шаблону. | Додавання шаблону (до модуля шаблонів, генерації або команд), який не містить заладок | Додати до шаблону хоча б одну закладку |
| Ви впевнені, що хочете видалити шаблон? Видалення скасувати неможливо. | Видалення шаблону з модуля шаблонів | Натиснути кнопку «Так» або «Ні» |

Продовження таблиці 2.7

|  |  |  |
| --- | --- | --- |
| **Текст** | **Ситуація** | **Рекомендовані дії** |
| Шаблон має бути з унікальним іменем | Додавання шаблону до модуля шаблонів з вже існуючою назвою | Змінити назву шаблону на іншу |
| Шаблон з назвою [введена\_назва] НЕ знайдено. | Пошук шаблона з неіснуючою назвою | Перевірити правильність написання назви для пошуку або додати шаблон з потрібною назвою |
| Перевірте ведення назви шаблону! Назва шаблону [введена\_назва] має містити .docx наприкінці. | Пошук шаблона з назвою, яка не містить .docx наприкінці | Додати до назви .docx |
| Шаблони для пошуку відсутні! Додайте шаблони. | Пошук шаблона за відсутності доданих шаблонів | Додати хоча б один шаблон |
| Назва шаблона не має співпадати з вже завантаженим шаблоном! Змініть назву одного з файлів. | Завантаження шаблону до модуля генерації, який має назву, яка співпадає зі вже збереженим шаблоном | Змінити назву одного з шаблонів |
| Документ не сформовано! Можливо, не вказаний шлях для збереження, або некоректна назва вихідного документа чи не показані закладки шаблона. | При спробі сформувати документ не вказані дані для вихідного документа | Перевірити запропоновані можливі місця проблем та виправити |
| Завантажте шаблон або виберіть зі списку збережених! | При спробі сформувати документ на основі шаблона не вибрано шаблон | Завантажити шаблон або до модуля генерації, або до модуля шаблонів |
| Задайте дані для всіх закладок шаблона! | При спробі сформувати документ на основі шаблона не були задані дані для всіх закладок | Ввести дані для закладок шаблона відповідно вибраного типу |
| Увага! Попередньо задані дані для закладок поточного шаблону буде видалено! | Зміна шаблону для формування документа | Натиснути кнопку «Так» або «Ні» |
| Спочатку завантажте заповнений шаблон. | При показі закладок у модулі команд не завантажено вхідний документ | Завантажити вхідний документ |
| Задайте налаштування команди! | При спробі створити команду не задані налаштування | Натиснути кнопку «Показати закладки» |
| Команда з такою назвою вже існує! | При спробі створити команду з вже існуючою назвою | Змінити назву нової команди або існуючої |

Продовження таблиці 2.7

|  |  |  |
| --- | --- | --- |
| **Текст** | **Ситуація** | **Рекомендовані дії** |
| Назви вхідної та вихідної закладки не мають співпадати! Або виберіть закладку з іншою назвою,  або змініть назву закладки в [назва\_вхідного\_документа] та завантажте його знову. | При спробі створення команди з налаштуваннями, які мають однакові назви вхідної та вихідної закладки | Виконати запропоновані у повідомленні дії |
| Ви впевнені, що хочете видалити команду? Видалення скасувати неможливо. | При спробі видалити команду | Натиснути кнопку «Так» або «Ні» |

### **2.2.3 Проєктування бази даних**

В програмі передбачено використання бази даних SQLite. SQLite – це бібліотека мовою C, яка реалізує невелику, швидку, самодостатню, високонадійну, повнофункціональну систему баз даних SQL [7].

Для програми розроблено дві таблиці (Templates та Commands), опис яких наведено у таблицях 2.8 – 2.9.

Таблиця 2.8 – Таблиця «Templates» (шаблони)

|  |  |  |  |
| --- | --- | --- | --- |
| Назва поля | Опис поля | Тип поля | Ключ |
| Id | Унікальний ідентифікатор шаблону | bigint | + |
| Name | Назва шаблону | text | - |
| Path | Шлях до файлу | text | - |
| Content | Зміст шаблону | blob | - |
| Bookmarks | Закладки (у форматі JSON) | text | - |

Таблиця 2.9 – Таблиця «Commands» (команди)

|  |  |  |  |
| --- | --- | --- | --- |
| Назва поля | Опис поля | Тип поля | Ключ |
| Id | Унікальний ідентифікатор команди | bigint | + |
| Name | Назва команди | text | - |
| InputDoc | Вхідний документ (у форматі JSON) | text | - |
| OutputTemplate | Вихідний документ (у форматі JSON) | text | - |
| SettingBookmarks | Налаштування команди (у форматі JSON) | text | - |

# **РОЗРОБКА ПРОГРАМИ**

# **ТЕСТУВАННЯ ТА НАЛАГОДЖЕННЯ**

# **АНАЛІЗ ТА ВИСНОВКИ**

# **БІБЛІОГРАФІЧНИЙ СПИСОК**

1. Про Національний архівний фонд та архівні установи: Закон України від 24.12.1993 № 3814-XII. URL: <https://zakon.rada.gov.ua/laws/show/3814-12#Text>
2. Overview of JavaScript PDF, DOCX WebViewer. Apryse Documentation: веб-сайт. URL: <https://docs.apryse.com/documentation/web/guides/overview/>
3. Formstack Documents. Zoho: вебсайт. URL: <https://help.zoho.com/portal/en/kb/crm/extensions/sales/articles/formstack-documents-24-2-2020-1#Installing_the_extension>
4. Getting Started with Formstack Documents Webinar. YouTube: вебсайт. URL: <https://www.youtube.com/watch?v=WTWdiuyCve0&list=PLe02B-M0zIcYY1SXIbNR8T5ziJMx7w55o&index=94>
5. What is three-tier architecture? IBM: веб-сайт. URL: <https://www.ibm.com/topics/three-tier-architecture>
6. Thorben. SOLID Design Principles Explained: The Single Responsibility Principle. Stackify: веб-сайт. URL: <https://stackify.com/solid-design-principles/>
7. What Is SQLite? SQLite: веб-сайт. URL: https://www.sqlite.org/

# **ДОДАТОК A**

**Технічне завдання**

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖУЮ

Проректор Українського

державного університету

науки і технологій

Анатолій РАДКЕВИЧ

ПРОГРАМА ДЛЯ ФОРМУВАННЯ   
WORD ДОКУМЕНТІВ ЗА ШАБЛОНАМИ

Технічне завдання

ЛИСТ ЗАТВЕРДЖЕННЯ

44165850.01365-01-ЛЗ

Завідувач кафедри КІТ

Вадим ГОРЯЧКІН

Керівник розробки

Олександр ЖЕВАГО

Виконавець

Вероніка БАГНЮК

Нормоконтролер

Світлана ВОЛКОВА

2024

ЗАТВЕРДЖЕНО

44165850.01365-01-ЛЗ

ПРОГРАМА ДЛЯ ФОРМУВАННЯ  
WORD ДОКУМЕНТІВ ЗА ШАБЛОНОМ

Технічне завдання

44165850.01365-01

Листів 16

202444165850.01365-01

2

ЗМІСТ

1. Введення3
2. Підстава для розробки4
3. Призначення розробки5
4. Вимоги до програмного продукту6
   1. Вимоги до функціональних характеристик6
   2. Вимоги до надійності10
   3. Умови експлуатації10
   4. Вимоги до складу і параметрів технічних засобів11
   5. Вимоги до інформаційної і програмної сумісності11
   6. Вимоги до маркування і упаковки12
   7. Вимоги до транспортування і зберігання12
5. Вимоги до програмної документації13
6. Стадії та етапи розробки14
7. Порядок контролю і приймання15
8. Бібліографічнийсписок16

44165850.01365-01

3

1. ВВЕДЕННЯ

Назва програмного продукту: Програма для формування Word документів за шаблоном.

Основна термінологія:

* Шаблон — Word документ, розмічений закладками;
* Команда перетворення (або команда) — механізм, який дозволяє перетворювати один документ на інший;
* Word документ — файл формату .docx, створений за допомогою використання текстового процесора Microsoft Office Word;

Причиною виникнення розробки є нижча вартість у порівнянні з аналогами та недостатня їхня функціональність для перетворення одних документів на інші.

Область застосування: програмний продукт, що розробляється, може використовуватися для часткової автоматизації ведення документації як користувачами для особистих потреб, так і для робочих документів підприємств та установ.

44165850.01365-01

4

1. ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки є наказ від 27.12.2023 року №1173ст виконуючого обов’язки ректора Українського державного університету науки і технологій «Про призначення керівників та затвердження тем бакалаврських робіт» за спеціальністю 121 Інженерія програмного забезпечення факультету «Комп’ютерних технологій і систем» по кафедрі «Комп’ютерні інформаційні технології».

Тема дипломної роботи — «Розробка програми для формування Word документів за шаблонами». Керівник — Жеваго О.О.

44165850.01365-01

5

1. ПРИЗНАЧЕННЯ РОЗРОБКИ

Функціональне призначення: формування Word документів за шаблонами, що включає можливість створення команд для перетворення одного документа в інший.

Експлуатаційне призначення: впровадження розробки дозволить частково автоматизувати ведення документації діловодами, за рахунок чого скоротяться часові витрати на створення робочих документів та зменшиться кількість помилок при їхньому заповненні.

44165850.01365-01

6

4. ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

4.1. Вимоги до функціональних характеристик

Вимоги до функціональних характеристик наступні:

* до модуля шаблонів:
* завантаження Word документів (файли формату .docx), розмічених закладками, які представляють шаблони. Шаблони завантажуються через діалогове вікно, яке пропонує відкрити Word файл (рис.1). Завантажені шаблони представлені у вигляді таблиці зі стовпцями: номер, назва, шлях до файлу, видалити (містить кнопку-піктограму для видалення) (рис.2);
* показ закладок завантажених шаблонів у вигляді таблиці зі стовпцями: номер, назва закладки, тип закладки (рис.3);
* редагування шаблонів, а саме введення нової назви у текстове поле та задання типу закладки через випадний список відповідного стовпця таблиці зі стовпцями: номер, назва закладки, тип закладки (див. рис.3);
* пошук шаблона в таблиці шаблонів за назвою, шляхом введення назви у відповідне текстове поле з піктограмою (рис.4);
* до модуля команд:
* створення команд для перетворення одного документа на інший, шляхом задання назви команди, завантаження заповненого документа (через діалогове вікно, яке дозволяє відкрити файл (див. рис.1)), вибору вихідного шаблону зі списку збережених шаблонів та налаштування команди, а саме якими закладками вихідного шаблона замінити закладки вхідного документа. Таблиця налаштувань команди має наступні стовпці: номер, вихідна закладка, вхідна закладка (є випадним список) (рис.5). Команди представлені у вигляді таблиці зі стовпцями: номер, назва команди, вхідний шаблон (містить назву), вихідний шаблон (містить назву), видалити (містить піктограму-кнопку для видалення) (рис.6)

44165850.01365-01

7

* редагування команди, а саме введення нової назви та задання налаштувань. Нова назва вводиться в текстове поле. Налаштування команди представлене у вигляді таблиці (див. рис.5);
* видалення команди шляхом натискання відповідної піктограми у стовпці видалити (див. рис.6).
* до модуля генерації (формування) документа:
* вибір шаблона для формування документа: зі списку збережених або завантажити новий шаблон. Вибір необхідного варіанта відбувається шляхом натискання на відповідну вкладки елемента управління (рис.7) та подальшого вибору зі збережених шаблонів або завантаження нового;
* можливість вибрати використовувати дані з команди шляхом увімкнення перемикача (рис.8);
* введення назви вихідного документа в текстове поле та вибір папки для збереження через відповідне вікно (рис.9). Вибраний шлях показується у текстовому полі;
* введення даних закладок шаблону залежно від типу. Закладки представлені у вигляді таблиці зі стовпцями: номер, закладка (містить назву), тип даних (спадний список з трьох елементів: текст, таблиця, зображення), дані (містить кнопку-піктограму для задання даних відповідного типу) (рис.10);
* формування та експортування з програми сформованих документів у вигляді .docx файлів згідно попередньо визначених налаштувань.

44165850.01365-01

8

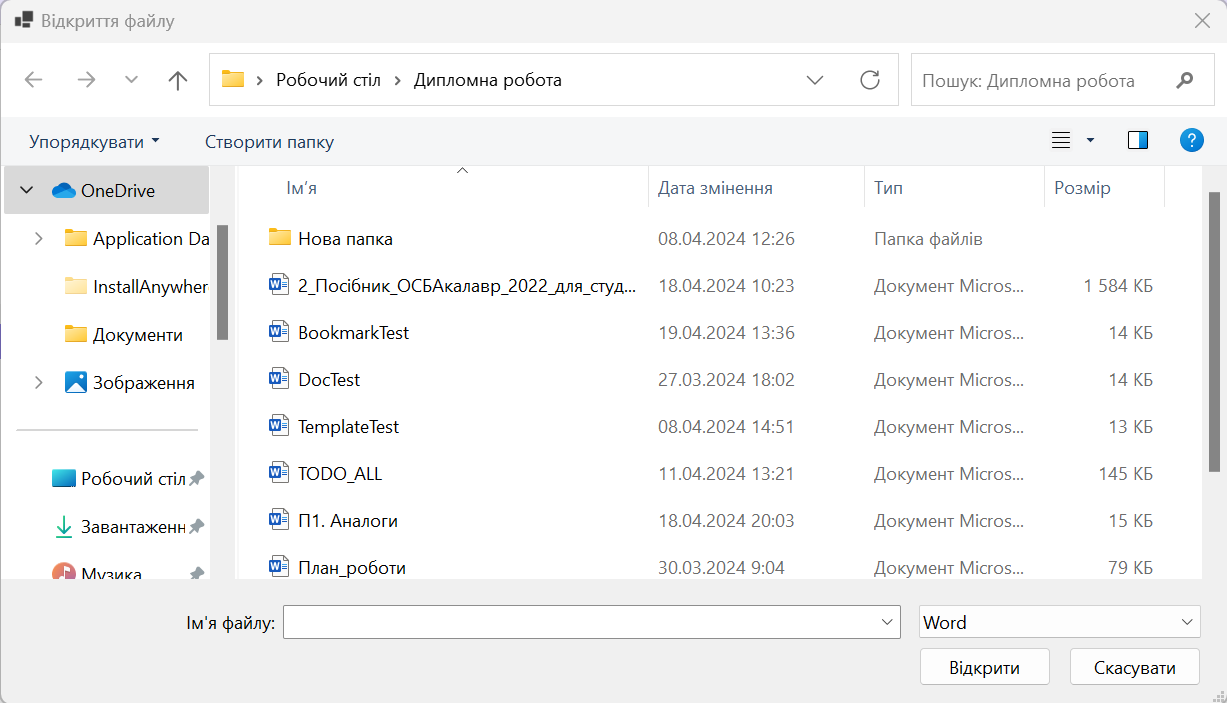


Рис.1. Приклад діалогового вікна відкриття файлу

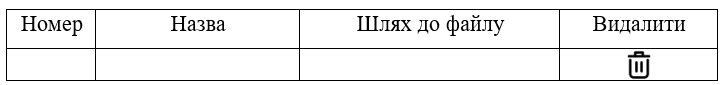


Рис.2. Макет для таблиці представлення шаблонів

|  |  |  |
| --- | --- | --- |
| Номер | Назва закладки | Тип закладки |

Рис.3. Макет для таблиці представлення закладок



Рис.4. Макет текстового поля для введення назви шаблону для пошуку

|  |  |  |
| --- | --- | --- |
| Номер | Вихідна закладка (шаблон) | Вхідна закладка (документ) |

Рис.5. Макет таблиці для налаштування команди

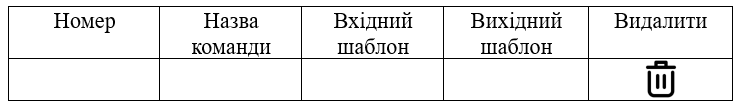


Рис.6. Макет таблиці для представлення команди

44165850.01365-01

9

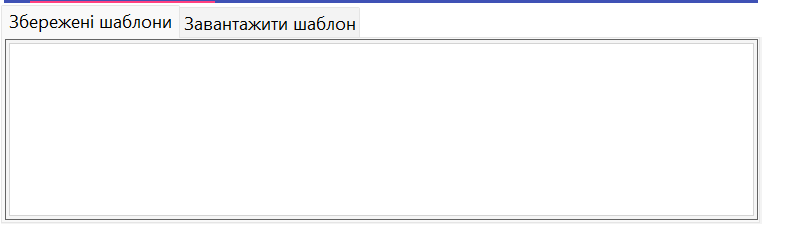


Рис.7. Макет елемента керування вкладками для вибору шаблона

Знімок екрана перемикачів увімкнено та вимкнено.

Рис.8. Макет перемикача для використання даних команди

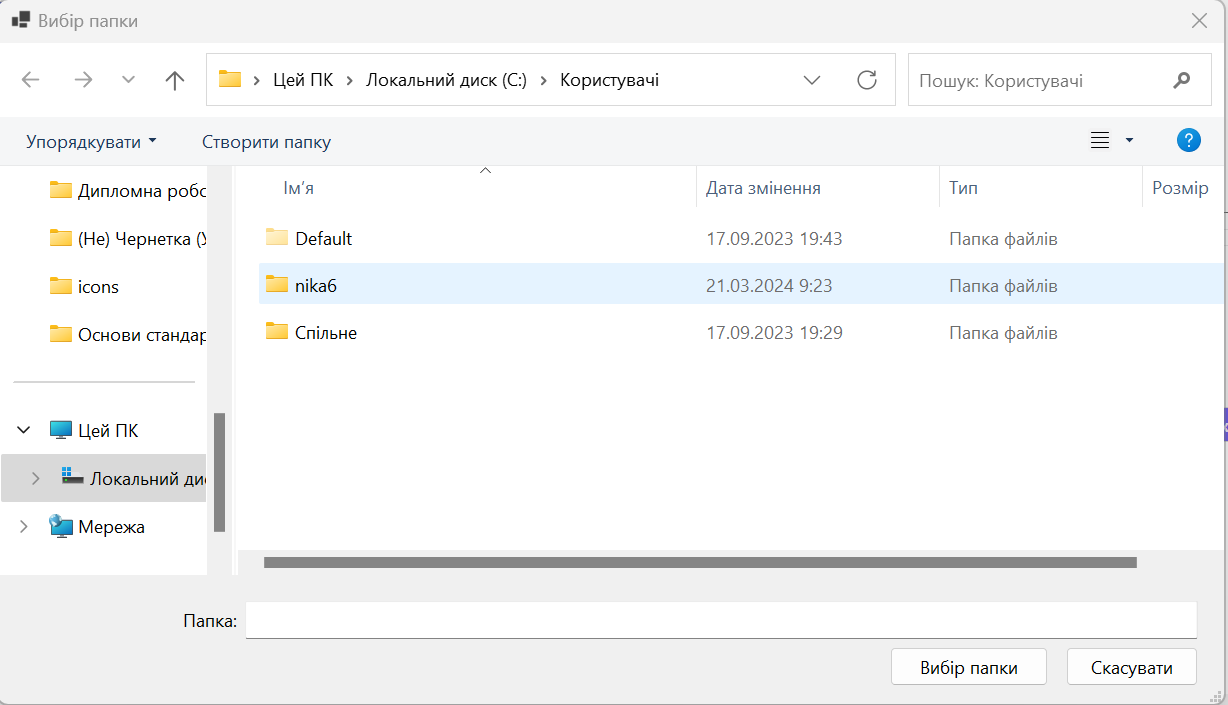


Рис.9. Приклад діалогового вікна для вибору папки

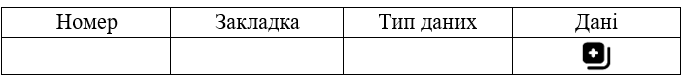


Рис.10. Макет таблиці закладок для формування документу за шаблоном

44165850.01365-01

10

4.2. Вимоги до надійності

Вимоги до надійності наступні:

* наявність архівної копії тексту програми на зовнішньому носії;
* наявність резервної копії бази даних на зовнішньому носії;
* забезпечення стійкого функціонування;
* контроль вхідної і вихідної інформації;
* захист від копіювання.

4.3. Умови експлуатації

Умови експлуатації мають відповідати вимогам Державних санітарних правил і норм роботи з візуальними дисплейними терміналами електронно-обчислювальних машин (ДСанПІН 3.3.2.007-98) (табл.2).

Таблиця 2. Норми мікроклімату

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Пора року | Категорія робіт | Температура повітря, град. С не більше | Відносна вологість повітря, % | Швидкість руху повітря, м/с |
| Холодна | легка-1а | 22-24 | 40-60 | 0,1 |
|  | легка-1б | 21-23 | 40-60 | 0,1 |
| Тепла | легка-1а | 23-25 | 40-60 | 0,1 |
|  | легка-1б | 22-24 | 40-60 | 0,1 |

Для сталого функціонування програмного забезпечення необхідні наступні умови експлуатації: температура навколишнього повітря 21-25 C°; відносна вологість 40-60 %. Працювати з програмою може людина, яка має навички роботи з персональним комп’ютером та ознайомлена з керівництвом користувача.

44165850.01365-01

11

4.4. Вимоги до складу і параметрів технічних засобів

Для експлуатації програмного продукту необхідні наступні технічні засоби: комп’ютер, монітор, комп’ютерна миша, клавіатура.

Комп’ютер повинен мати такі мінімальні характеристики:

* процесор: Intel(R) Celeron(R) CPU (1.8 – 2 ГГц);
* пам’ять: RAM 8 ГБ ;
* графічне ядро: Inlel (R) HD Graphics;
* порт: USB-порт.

Монітор повинен мати наступні параметри екрану:

* роздільна здатність – 1920 x 1080;
* формат кольору – RGB;
* глибина кольору – 6-біт.

Комп’ютерна миша повинна мати такі характеристики:

* тип датчика: оптичний;
* сумісність з ОС: Windows;
* кількість кнопок: 2;
* тип підключення: дротовий (USB-кабель).

Комп’ютерна клавіатура повинна мати такі характеристики:

* форма: повнорозмірна;
* розкладка: українська / англійська;
* сумісність з ОС: Windows.

4.5. Вимоги до інформаційної і програмної сумісності

Програмний продукт розробляється для операційної системи Windows 10/11.

Середовище розробки Microsoft Visual Studio Community 2022. MS VS Community 2022 — безкоштовне IDE, яке надає широкий функціонал для розробки настільних додатків.

Мова програмування: C#.

Додаткові пакети:

44165850.01365-01

12

* MaterialSkin.2 (для графічного інтерфейсу);
* Microsoft-WindowsAPICodePack-Shell (для вікна вибору папки);
* Microsoft.Extensions.DependencyInjection, Microsoft.Extensions.Hosting (для впровадження залежностей);
* FreeSpire.Doc (для роботи з Word документами);
* System.Data.SQLite (для роботи з SQLite);
* Newtonsoft.Json (для серіалізації закладок документу, щоб зберігати їх у базі даних).

Додаткові розширення:

* SQLite and SQL Server Compact ToolBox (створення бази даних).

4.6. Вимоги до маркування і упаковки

Упаковка програмного продукту, включаючи документацію, має бути захищена від механічних та кліматичних пошкоджень. На титульній стороні упаковці повинна бути вказана назва продукту, розробник та його юридична адреса (рис.4).

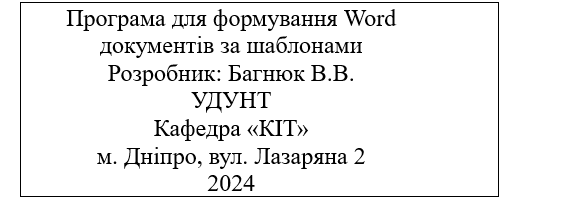


Рис.4. Приклад маркування упаковки

4.7. Вимоги до транспортування і зберігання

Транспортування має забезпечувати збереження програмного продукту, його цілісність і запобігати несанкціонованому доступу. Програмний продукт міститься на USB-флеш-накопичувачі (флешці). Флешка зберігається в упаковці у вигляді картонної коробки розмірами 50 х 20 х 15 мм з відповідним маркуванням (див. підрозділ 4.6. Вимоги до маркування та упаковки).

Місце зберігання має бути сухим, мати кімнатну температуру (див. підрозділ 4.3. Умови експлуатації), запобігати тривалому потраплянню прямого сонячного світла безпосередньо на флешку з програмним продуктом

44165850.01365-01

13

1. ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

Програмна документація повинна складатися з наступних документів:

* специфікація;
* текст програми;
* керівництво користувача.

Вся документація до програмного продукту повинна задовольняти вимоги до програмної документації [1].

44165850.01365-01

14

1. СТАДІЇ ТА ЕТАПИ РОЗРОБКИ

Стадії та етапи розробки наведені в табл.3.

Таблиця 3. Стадії та етапи розробки

|  |  |  |
| --- | --- | --- |
| Стадія | Зміст робіт | Терміни виконання |
| Технічне завдання | Постановка завдання, збір початкових матеріалів, вибір і обґрунтування критеріїв ефективності і якості. Попередній вибір методів рішення задач. Визначення вимог до технічних засобів. Узгодження і затвердження технічного завдання. | 19.02.24 – 10.03.24 |
| Робочий проект | Програмування та налагодження програми | 11.03.24 – 28.04.24 |
| Тестування програми | 29.04.24 – 26.05.24 |
| Розробка, узгодження і затвердження програмної документації | 27.05.24 – 16.06.24 |

44165850.01365-01

15

1. ПОРЯДОК КОНТРОЛЮ І ПРИЙМАННЯ

Контроль за виконанням роботи здійснює керівник розробки Жеваго О.О.

Прийом програмного продукту здійснюється уповноваженою комісією.

44165850.01365-01

16

1. БІБЛІОГРАФІЧНИЙ СПИСОК

1. Івченко, Ю.М. Основи стандартизації програмних систем [Текст]: методичні вказівки до дипломного проектування та лабораторних робіт / уклад.: Ю. М. Івченко, В. І. Шинкаренко, В. Г. Івченко; Дніпропетр. нац. ун-т залізн. трансп. ім. акад. В. Лазаряна. – Д.: Вид-во Дніпропетр. нац. ун-ту залізн. трансп. ім. акад. В. Лазаряна, 2009. - 38 с.

# **ДОДАТОК Б**

**Специфікація**

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖУЮ

Проректор Українського

державного університету

науки і технологій

Анатолій РАДКЕВИЧ

ПРОГРАМА ДЛЯ ФОРМУВАННЯ   
WORD ДОКУМЕНТІВ ЗА ШАБЛОНАМИ

Специфікація

ЛИСТ ЗАТВЕРДЖЕННЯ

44165850.01365-01-ЛЗ

Завідувач кафедри КІТ

Вадим ГОРЯЧКІН

Керівник розробки

Олександр ЖЕВАГО

Виконавець

Вероніка БАГНЮК

Нормоконтролер

Світлана ВОЛКОВА

2024

**ДОДАТОК Б**

**Специфікація**

ЗАТВЕРДЖЕНО

44165850.01365-01-ЛЗ

ПРОГРАМА ДЛЯ ФОРМУВАННЯ  
WORD ДОКУМЕНТІВ ЗА ШАБЛОНОМ

Специфікація

44165850.01365-01

Листів 2

2024

44165850.01365-01

3

|  |  |  |
| --- | --- | --- |
| Позначення | Найменування | Примітка |
|  | Документація |  |
| 44165850.01365-01-ЛЗ | Лист затвердження |  |
| 44165850.01365-01 12 01-ЛЗ | Лист затвердження |  |
| 44165850.01365-01 12 01 | Текст програми |  |
| 44165850.01365-01 І3 01-ЛЗ | Лист затвердження |  |
| 44165850.01365-01 І3 01 | Керівництво користувача |  |

# **ДОДАТОК В**

**Текст програми**

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖУЮ

Проректор Українського

державного університету

науки і технологій

Анатолій РАДКЕВИЧ

ПРОГРАМА ДЛЯ ФОРМУВАННЯ WORD   
ДОКУМЕНТІВ ЗА ШАБЛОНАМИ

Текст програми

ЛИСТ ЗАТВЕРДЖЕННЯ

44165850.01365-01 12 01-ЛЗ

Завідувач кафедри КІТ

Вадим ГОРЯЧКІН

Керівник розробки

Олександр ЖЕВАГО

Виконавець

Вероніка БАГНЮК

Нормоконтролер

Світлана ВОЛКОВА

2024

ЗАТВЕРДЖЕНО

44165850.01365-01 12 01-ЛЗ

ІНТЕРНЕТ-МАГАЗИН

ПОБУТОВОЇ ТЕХНІКИ «TECHNICSHUB»

Текст програми

44165850.01365-01 12 01

Листів 53

2024

44165850.01365-01 12 01

2

АНОТАЦІЯ

Документ 44165850.01365-01 12 01 «Програма для формування Word документів за шаблонами. Текст програми» входить до складу програмної документації на комплекс з п'яти проєктів, що реалізовують створення Word документів за шаблонами.

У даному документі представлений текст програм. Програми в проєктах написані на мові C#. Об'єм пам'яті, що займають програми комплексу, складає 105 Мб. Конфігурація комп'ютера стандартна. Комплекс функціонує в середовищі операційних систем Windows 10 та наступних версіях.

44165850.01365-01 12 01

3

ЗМІСТ

1. Схема взаємодії проєктів програми4
2. Текст програми5
   1. Проєкт Domain5
   2. Проєкт Persistence7
   3. Проєкт Core14
   4. Проєкт Presentation17
   5. Проєкт WinFormsUI23

44165850.01365-01 12 01

4

1 CХЕМА ВЗАЄМОДІЇ ПРОЄКТІВ ПРОГРАМИ

На рис. 1 приведена схема взаємодії проєктів програми.

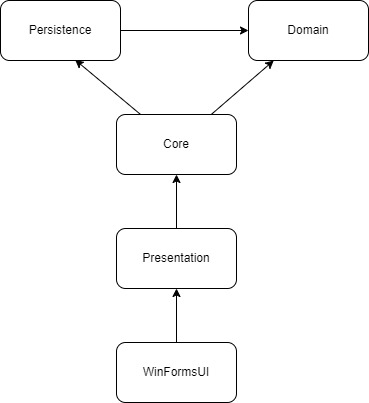


Рис.1. Схема взаємодії проєктів програми

Згідно з рис.1 взаємодія проєктів відбувається наступним чином:

* WinFormsUI взаємодіє з Presentation;
* Presentation взаємодіє з Core;
* Core взаємодіє з Persistence та Domain;
* Persistence взаємодіє з Domain.

44165850.01365-01 12 01

5

1. ТЕКСТ ПРОГРАМИ
   1. **Проєкт Domain**

Загальний опис змісту: містить моделі предметної області та інтерфейси для рівня доступу до даних. Розробник: Багнюк В.В.

Файл Template.cs

namespace Domain

{

/// <summary>

/// Шаблон

/// </summary>

public class Template

{

/// <summary>

/// Конструктор шаблона

/// </summary>

/// <param name="fileName">назва файлу</param>

/// <param name="fileContent">зміст файлу</param>

/// <exception cref="ArgumentNullException"></exception>

public Template(string fileName, string filePath, byte[] fileContent, Dictionary<string, string> fileBookmarks)

{

if (fileContent == null || !fileContent.Any())

{

throw new ArgumentNullException(nameof(fileContent));

}

this.FileName = fileName ?? throw new ArgumentNullException(nameof(fileName));

this.FileContent = fileContent;

this.FilePath = filePath ?? throw new ArgumentNullException(nameof(filePath));

this.FileBookmarks = fileBookmarks;

}

/// <summary>

/// Назва файлу

/// </summary>

public string FileName { get; }

/// <summary>

/// Шлях до файлу

/// </summary>

public string FilePath { get; }

/// <summary>

/// Зміст файлу

/// </summary>

public byte[] FileContent { get; }

/// <summary>

/// Закладки файлу

/// (ключ - назва, значення - тип)

/// </summary>

public Dictionary<string, string> FileBookmarks { get; }

}

}

44165850.01365-01 12 01

6

Файл Command.cs

namespace Domain

{

/// <summary>

/// Команда

/// </summary>

public class Command

{

/// <summary>

/// Конструктор команди

/// </summary>

/// <param name="commandName">назва команди</param>

/// <param name="commandSetting">налаштування команди</param>

/// <param name="inputTemplate">вхідний шаблон</param>

/// <param name="outputTemplate">вихідний шаблон</param>

/// <exception cref="ArgumentNullException">виключення</exception>

public Command(string commandName,

IReadOnlyDictionary<string, string> commandSetting,

Template inputTemplate,

Template outputTemplate)

{

this.CommandName = commandName ?? throw new ArgumentNullException(nameof(commandName));

this.CommandSetting = commandSetting ?? throw new ArgumentNullException(nameof(commandSetting));

this.InputTemplate = inputTemplate ?? throw new ArgumentNullException(nameof(inputTemplate));

this.OutputTemplate = outputTemplate ?? throw new ArgumentNullException(nameof(outputTemplate));

}

/// <summary>

/// Налаштування команди

/// </summary>

public IReadOnlyDictionary<string, string> CommandSetting { get; }

/// <summary>

/// Назва команди

/// </summary>

public string CommandName { get; }

/// <summary>

/// Вхідий шаблон

/// </summary>

public Template InputTemplate { get; }

/// <summary>

/// Вихідний шаблон

/// </summary>

public Template OutputTemplate { get; }

}

}

44165850.01365-01 12 01

7

Файл ITemplateRepository.cs

namespace Domain

{

/// <summary>

/// Інтерфейс репозиторію шаблону

/// </summary>

public interface ITemplateRepository

{

Task<Template> Create(string fileName, string filePath, byte[] fileContent, Dictionary<string, string> fileBookmarks);

Task<IEnumerable<Template>> Get(string name);

Task Update(string oldName,

string newName,

IDictionary<string, string> newBookmarks);

Task Save(Template template);

Task Delete(string name);

}

}

Файл ICommandsRepository.cs

namespace Domain

{

/// <summary>

/// Інтерфейс репозиторію команд

/// </summary>

public interface ICommandRepository

{

Task<Command> Create(string commandName,

IReadOnlyDictionary<string, string> commandSetting,

Template inputTemplate,

Template outputTemplate);

Task<IEnumerable<Command>> Get(string cmdName);

Task Update(string oldName,

string newName,

IDictionary<string, string> newSetting);

Task Save(Command command);

Task Delete(string cmdName);

}

}

* 1. **Проєкт Persistence**

Загальний опис змісту: містить реалізації інтерфейсів, описаних у проєкті Domain. Розробник: Багнюк В.В.

44165850.01365-01 12 01

8

Файл InDatabaseTemplateRepository.cs

using Domain;

using Newtonsoft.Json;

using System.Data.SQLite;

using Template = Domain.Template;

namespace Persistence

{

/// <summary>

/// Репозиторій шаблонів у базі даних

/// </summary>

public class InDatabaseTemplateRepository : ITemplateRepository

{

private SQLiteConnection connection;

/// <summary>

/// Створення підключення до

/// бази даних SQLite

/// </summary>

/// <returns>підключення до бази даних sqlite</returns>

/// <exception cref="Exception">виключення</exception>

private SQLiteConnection CreateConnection()

{

SQLiteConnection conn = new SQLiteConnection(@"Data Source=..\..\..\..\repos.db; Version = 3;");

try

{

conn.Open();

}

catch (Exception ex)

{

throw new Exception(ex.Message);

}

return conn;

}

/// <summary>

/// Створення шаблону

/// (в пам'яті)

/// </summary>

/// <param name="fileName">назва файлу</param>

/// <param name="filePath">шлях до файлу</param>

/// <param name="fileContent">зміст файлу</param>

/// <param name="fileBookmarks">закладки файлу</param>

/// <returns>успішність виконання операції з результатом</returns>

/// <exception cref="ArgumentNullException">виключення</exception>

public Task<Template> Create(string fileName, string filePath, byte[] fileContent, Dictionary<string, string> fileBookmarks)

{

Template template;

if (string.IsNullOrEmpty(fileName) || fileContent == null)

throw new ArgumentNullException();

else

template = new Template(fileName, filePath, fileContent, fileBookmarks);

return Task.FromResult(template);

}

44165850.01365-01 12 01

9

/// <summary>

/// Видалення шаблону з бази даних

/// </summary>

/// <param name="name">назва шаблону</param>

/// <returns>успішність виконання операціії</returns>

public Task Delete(string name)

{

connection = CreateConnection();

var command = connection.CreateCommand();

command.CommandText = @"DELETE FROM Templates WHERE Name=@name";

command.Parameters.AddWithValue("name", name);

command.ExecuteNonQuery();

command.Connection.Close();

connection.Close();

return Task.CompletedTask;

}

/// <summary>

/// Отримання шаблону з бази даних

/// </summary>

/// <param name="name">назва шаблону</param>

/// <returns>успішність виконання операції з результатом</returns>

public Task<IEnumerable<Template>> Get(string name)

{

connection = CreateConnection();

var command = connection.CreateCommand();

byte[] content = new byte[0];

if (string.IsNullOrEmpty(name))

{

command.CommandText = @"SELECT \* FROM Templates";

}

else

{

command.CommandText = @"SELECT \* FROM Templates WHERE Name=@name";

command.Parameters.AddWithValue("@name", name);

}

using (var reader = command.ExecuteReader())

{

List<Template> templates = new List<Template>();

while (reader.Read())

{

var \_name = reader.GetString(1);

var path = reader.GetString(2);

content = (byte[])reader["Content"];

Dictionary<string, string> bookmarks = JsonConvert.DeserializeObject<Dictionary<string, string>>(reader.GetString(4));

templates.Add(new Template(\_name, path, content, bookmarks));

}

connection.Close();

return Task.FromResult<IEnumerable<Template>>(templates);

}

}

44165850.01365-01 12 01

10

/// <summary>

/// Збереження шаблону до бази даних

/// </summary>

/// <param name="template">шаблон</param>

/// <returns>успішність виконання операції</returns>

public Task Save(Template template)

{

connection = CreateConnection();

string insertQuery = "INSERT INTO Templates ('Name','Path','Content','Bookmarks') VALUES(@name,@path,@content,@bookmarks)";

SQLiteCommand sqlite\_cmd = new SQLiteCommand(insertQuery, connection);

string json\_bookmarks = JsonConvert.SerializeObject(template.FileBookmarks);

sqlite\_cmd.Parameters.AddWithValue("name", template.FileName);

sqlite\_cmd.Parameters.AddWithValue("path", template.FilePath);

sqlite\_cmd.Parameters.AddWithValue("content", template.FileContent);

sqlite\_cmd.Parameters.AddWithValue("bookmarks", json\_bookmarks);

var insertResult = sqlite\_cmd.ExecuteNonQuery();

sqlite\_cmd.Connection.Close();

connection.Close();

return Task.FromResult(insertResult);

}

/// <summary>

/// Оновлення шаблону в базі данхи

/// </summary>

/// <param name="oldName">назва шаблону для оновлення</param>

/// <param name="newName">нова назва шаблону</param>

/// <param name="newBookmarks">оновлені закладки</param>

/// <returns>успішність виконання операції</returns>

public Task Update(string oldName, string newName, IDictionary<string, string> newBookmarks)

{

connection = CreateConnection();

var command = connection.CreateCommand();

string json\_bookmarks = JsonConvert.SerializeObject(newBookmarks);

command.CommandText = @"UPDATE Templates SET Name=@newname, Bookmarks=@newbookmarks WHERE Name=@oldname";

command.Parameters.AddWithValue("newname", newName);

command.Parameters.AddWithValue("newbookmarks", json\_bookmarks);

command.Parameters.AddWithValue("oldname", oldName);

\_ = command.ExecuteNonQuery();

command.Connection.Close();

connection.Close();

return Task.CompletedTask;

}

}

}

44165850.01365-01 12 01

11

Файл InDatabaseCommandsRepository.cs

using Domain;

using Newtonsoft.Json;

using System.Data.SQLite;

using Template = Domain.Template;

namespace Persistence

{

/// <summary>

/// Репозиторій команд у базі даних

/// </summary>

public class InDatabaseCommandsRepository : ICommandRepository

{

private SQLiteConnection connection;

/// <summary>

/// Створення підключення до

/// бази даних SQLite

/// </summary>

/// <returns>підключення до бази даних sqlite</returns>

/// <exception cref="Exception">виключення</exception>

private SQLiteConnection CreateConnection()

{

SQLiteConnection conn = new SQLiteConnection(@"Data Source=..\..\..\..\repos.db; Version = 3;");

try

{

conn.Open();

}

catch (Exception ex)

{

throw new Exception(ex.Message);

}

return conn;

}

/// <summary>

/// Створення команди

/// (в пам'яті)

/// </summary>

/// <param name="commandName">назва команди</param>

/// <param name="commandSetting">налаштування команди ()</param>

/// <param name="inputTemplate">вхідний документ</param>

/// <param name="outputTemplate">вихідний шаблон</param>

/// <returns>успішність виконання операції з результатом</returns>

/// <exception cref="ArgumentNullException">виключення</exception>

public Task<Command> Create(string commandName,

IReadOnlyDictionary<string, string> commandSetting,

Template inputTemplate,

Template outputTemplate)

{

Command command;

if (string.IsNullOrEmpty(commandName) || commandSetting == null || inputTemplate == null || outputTemplate == null)

throw new ArgumentNullException();

else

command = new Command(commandName, commandSetting, inputTemplate, outputTemplate);

return Task.FromResult(command);

}

44165850.01365-01 12 01

12

/// <summary>

/// Видалення команди з бази даних

/// </summary>

/// <param name="cmdName">назва команди</param>

/// <returns>успішність виконання операції</returns>

public Task Delete(string cmdName)

{

connection = CreateConnection();

var command = connection.CreateCommand();

command.CommandText = @"DELETE FROM Commands WHERE Name=@name";

command.Parameters.AddWithValue("name", cmdName);

command.ExecuteNonQuery();

command.Connection.Close();

connection.Close();

return Task.CompletedTask;

}

/// <summary>

/// Отримання команди з бази даних

/// </summary>

/// <param name="cmdName">назва команди</param>

/// <returns>успішність виконання операції з результатом</returns>

public Task<IEnumerable<Command>> Get(string cmdName)

{

connection = CreateConnection();

var command = connection.CreateCommand();

if (string.IsNullOrEmpty(cmdName))

{

command.CommandText = @"SELECT \* FROM Commands";

}

else

{

command.CommandText = @"SELECT \* FROM Commands WHERE Name=@name";

command.Parameters.AddWithValue("@name", cmdName);

}

using (var reader = command.ExecuteReader())

{

List<Command> commands = new List<Command>();

while (reader.Read())

{

var name = reader.GetString(1);

var input\_document = JsonConvert.DeserializeObject<Template>(reader["InputDoc"].ToString());

var output\_template = JsonConvert.DeserializeObject<Template>(reader["OutputTemplate"].ToString());

var setting\_bookmarks = JsonConvert.DeserializeObject<Dictionary<string, string>>(reader.GetString(4));

commands.Add(new Command(name, setting\_bookmarks, input\_document, output\_template));

}

connection.Close();

return Task.FromResult<IEnumerable<Command>>(commands);

}

}

44165850.01365-01 12 01

13

/// <summary>

/// Збереження команди в базі даних

/// </summary>

/// <param name="command">команда</param>

/// <returns>успішність виконання операції з результатом</returns>

public Task Save(Command command)

{

string json\_inputdoc = JsonConvert.SerializeObject(command.InputTemplate);

string json\_outputTemplate = JsonConvert.SerializeObject(command.OutputTemplate);

string json\_setting = JsonConvert.SerializeObject(command.CommandSetting);

connection = CreateConnection();

string insertQuery = "INSERT INTO Commands ('Name','InputDoc','OutputTemplate','SettingBookmarks') " +

"VALUES(@name,@inputdoc,@outtemplate,@setting)";

SQLiteCommand sqlite\_cmd = new SQLiteCommand(insertQuery, connection);

sqlite\_cmd.Parameters.AddWithValue("@name", command.CommandName);

sqlite\_cmd.Parameters.AddWithValue("@inputdoc", json\_inputdoc);

sqlite\_cmd.Parameters.AddWithValue("@outtemplate", json\_outputTemplate);

sqlite\_cmd.Parameters.AddWithValue("@setting", json\_setting);

var insertResult = sqlite\_cmd.ExecuteNonQuery();

sqlite\_cmd.Connection.Close();

connection.Close();

return Task.FromResult(insertResult);

}

/// <summary>

/// Оновлення команди в базі даних

/// </summary>

/// <param name="oldName">назва команди для оновлення</param>

/// <param name="newName">нова назва команди</param>

/// <param name="newSetting">нові налаштування команди</param>

/// <returns>успішність виконання операції</returns>

public Task Update(string oldName, string newName, IDictionary<string, string> newSetting)

{

connection = CreateConnection();

string json\_bookmarks = JsonConvert.SerializeObject(newSetting);

var command = connection.CreateCommand();

command.CommandText = @"UPDATE Commands SET Name=@newname, SettingBookmarks=@newbookmarks WHERE Name=@oldname";

command.Parameters.AddWithValue("newname", newName);

command.Parameters.AddWithValue("newbookmarks", json\_bookmarks);

command.Parameters.AddWithValue("oldname", oldName);

\_ = command.ExecuteNonQuery();

command.Connection.Close();

connection.Close();

return Task.CompletedTask;

}

}

}

44165850.01365-01 12 01

14

* 1. **Проєкт Core.**

Загальний опис змісту: містить логіку програми (сервіси для доступу даних у базі даних). Розробник: Багнюк В.В.

Файл TemplateService.cs

using Domain;

namespace Core.Services

{ /// <summary>

/// Сервіс для шаблону

/// </summary>

public class TemplateService

{

private readonly ITemplateRepository templateRepository;

public TemplateService(ITemplateRepository templateRepository)

{

this.templateRepository = templateRepository;

}

/// <summary>

/// Отримати всі шаблони

/// </summary>

/// <returns>успішність виконання операції з результатом</returns>

public Task<IEnumerable<Template>> GetAllTemplates()

{

return templateRepository.Get(string.Empty);

}

/// <summary>

/// Отримати шаблон

/// </summary>

/// <param name="templateName">шаблон</param>

/// <returns>успішність виконання операції з результатом</returns>

public Task<IEnumerable<Template>> GetTemplate(string templateName)

{

return templateRepository.Get(templateName);

}

/// <summary>

/// Створити шаблон

/// </summary>

/// <param name="fileName">назва шаблону</param>

/// <param name="filePath">шлях до шаблону</param>

/// <param name="fileContent">зміст шаблону</param>

/// <returns>успішність виконання операції з результатом</returns>

public Task<Template> CreateTemplate(string fileName,

string filePath,

byte[] fileContent,

Dictionary<string, string> fileBookmarks)

{

return templateRepository.Create(fileName, filePath, fileContent, fileBookmarks);

}

44165850.01365-01 12 01

15

/// <summary>

/// Оновлення шаблону

/// </summary>

/// <param name="oldTemplate">стариий шаблон</param>

/// <param name="newTemplate">новий шаблон</param>

/// <returns>успішність виконання операції</returns>

public Task UpdateTemplate(string oldName,

string newName,

IDictionary<string, string> newBookmarks)

{

//return templateRepository.Update(oldTemplate, newTemplate);

return templateRepository.Update(oldName, newName, newBookmarks);

}

/// <summary>

/// Видалення шаблону

/// </summary>

/// <param name="fileName">назва шаблону</param>

/// <returns>успішність виконання операції</returns>

public Task DeleteTemplate(string fileName)

{

return templateRepository.Delete(fileName);

}

/// <summary>

/// Збереження шаблону

/// </summary>

/// <param name="template">шаблон</param>

/// <returns>успішність виконання операції</returns>

public Task Save(Template template)

{

return templateRepository.Save(template);

}

}

}

Файл CommandService.cs

using Domain;

namespace Core.Services

{

/// <summary>

/// Сервіс для команд

/// </summary>

public class CommandService

{

private readonly ICommandRepository commandRepository;

public CommandService(ICommandRepository commandRepository)

{

this.commandRepository = commandRepository;

}

/// <summary>

/// Отримати всі команди

/// </summary>

/// <returns>успішність виконання операції</returns>

public Task<IEnumerable<Command>> GetAllCommands()

{

return commandRepository.Get(string.Empty);

}

44165850.01365-01 12 01

16

/// <summary>

/// Отримати команду

/// </summary>

/// <param name="commandName">назва команди</param>

/// <returns>успішність виконання операції</returns>

public Task<IEnumerable<Command>> GetCommand(string commandName)

{

return commandRepository.Get(commandName);

}

/// <summary>

/// Створення команди

/// </summary>

/// <param name="commandName">назва команди</param>

/// <param name="commandSetting">налаштування команди</param>

/// <param name="inputTemplate">вхідний шаблон</param>

/// <param name="outputTemplate">вихідний шаблон</param>

/// <returns>успішність виконання операції</returns>

public Task<Command> CreateCommand(string commandName,

IReadOnlyDictionary<string, string> commandSetting,

Template inputTemplate,

Template outputTemplate)

{

return commandRepository.Create(commandName, commandSetting, inputTemplate, outputTemplate);

}

/// <summary>

/// Видалення команди

/// </summary>

/// <param name="commandName">назва команди</param>

/// <returns>успішність виконання операції</returns>

public Task DeleteCommand(string commandName)

{

return commandRepository.Delete(commandName);

}

/// <summary>

/// Оновлення команди

/// </summary>

/// <param name="oldCommand">стара команда</param>

/// <param name="newCommand">нова команда</param>

/// <returns>успішність виконання операції</returns>

public Task UpdateCommand(string oldName,

string newName,

IDictionary<string, string> newSetting)

{

return commandRepository.Update(oldName, newName, newSetting);

}

/// <summary>

/// Зберегти команду

/// </summary>

/// <param name="command">команда</param>

/// <returns>успішність виконання операції</returns>

public Task SaveCommand(Command command)

{

return commandRepository.Save(command);

}

}

}

44165850.01365-01 12 01

17

Файл Extensions.cs

using Domain;

using Microsoft.Extensions.DependencyInjection;

using Persistence;

namespace Core

{

/// <summary>

/// Розширення

/// </summary>

public static class Extensions

{

/// <summary>

/// Додати до пам'яті репозиторії

/// </summary>

/// <param name="services"></param>

/// <returns></returns>

public static IServiceCollection AddInMemoryRepositories(this IServiceCollection services)

{

services.AddScoped<ITemplateRepository, InDatabaseTemplateRepository>();

services.AddScoped<ICommandRepository, InDatabaseCommandsRepository>();

return services;

}

}

}

* 1. **Проєкт Presentation**

Загальний опис змісту: містить елементи партерну MVP (Model-View-Presenter). Розробник: Багнюк В.В.

Файл TemplateViewModel.cs

namespace Presentation.ViewModels

{ /// <summary>

/// Модель представлення шаблона

/// </summary>

public class TemplateViewModel

{

public string FileName { get; set; }

public string FilePath { get; set; }

public byte[] ContentFile { get; set; }

public Dictionary<string, string> BookmarksFile { get; set; }

}

}

Файл CommandViewModel.cs

namespace Presentation.ViewModels

{ /// <summary>

/// Модель представлення команди

/// </summary>

public class CommandViewModel

{

public string NameCommand { get; set; }

public TemplateViewModel InputTemplate { get; set; }

public TemplateViewModel OutputTemplate { get; set; }

public Dictionary<string, string> CommandSetting { get; set; }

44165850.01365-01 12 01

18

}

}

Файл IView.cs

namespace Presentation.Views

{ /// <summary>

/// Інтерфейс представлення

/// </summary>

public interface IView

{

void Show();

void Close();

}

}

Файл IMainView.cs

using Presentation.ViewModels;

namespace Presentation.Views

{

public interface IMainView : IView

{

public TemplateViewModel viewTemplate { get; set; }

public string viewNameTemplate { get; set; }

public event EventHandler? SaveTemplate;

public event EventHandler? DeleteTemplate;

public event EventHandler? UpdateTemplate;

public event EventHandler<TemplateViewModel> GetTemplate;

public CommandViewModel viewCommand { get; set; }

public string viewNameCommand { get; set; }

public event EventHandler? SaveCommand;

public event EventHandler? DeleteCommand;

public event EventHandler<CommandViewModel>? GetCommand;

public event EventHandler? UpdateCommand;

void SetCommandsList(IEnumerable<CommandViewModel> commands);

void SetTemplateList(IEnumerable<TemplateViewModel> templates);

void SetBookmarksDictionary(IDictionary<string, string> dictionaryBookmarks);

}

}

Файл IPresenter.cs

namespace Presentation.Presenters

{ /// <summary>

/// Інтерфейс представлення

/// </summary>

public interface IPresenter

{

Task Run();

}

}

44165850.01365-01 12 01

19

Файл IMainPresenter.cs

namespace Presentation.Presenters

{

public interface IMainPresenter : IPresenter

{

}

}

Файл MainPresenter.cs

using Core.Services;

using Presentation.ViewModels;

using Presentation.Views;

namespace Presentation.Presenters

{

public class MainPresenter : IMainPresenter

{

private readonly IMainView mainView;

private readonly CommandService commandService;

private readonly TemplateService templateService;

public MainPresenter(IMainView mainView, CommandService commandService, TemplateService templateService)

{

this.mainView = mainView;

this.commandService = commandService;

this.templateService = templateService;

this.mainView.DeleteTemplate += (s, e) => RemoveTemplate(s, e);

this.mainView.SaveTemplate += (s, e) => \_ = SaveTemplate(s, e);

this.mainView.UpdateTemplate += (s, e) => UpdateTemplate(s, e, this.mainView.viewNameTemplate, this.mainView.viewTemplate);

this.mainView.GetTemplate += (s, e) => \_ = GetTemplate(s, this.mainView.viewTemplate);

this.mainView.SaveCommand += (s, e) => \_ = SaveCommand(s, e);

this.mainView.DeleteCommand += (s, e) => DeleteCommad(s, e);

this.mainView.GetCommand += (s, e) => \_ = GetCommand(s, this.mainView.viewCommand);

this.mainView.UpdateCommand += (s, e) => UpdateCommand(s, e, this.mainView.viewNameCommand, this.mainView.viewCommand);

}

/// <summary>

/// Подія "Отримати команду"

/// </summary>

public async Task GetCommand(object sender, CommandViewModel cvm)

{

// отримати команду за назвою

var cmd = await this.commandService.GetCommand(cvm.NameCommand);

// отримати вхідний документ команди

TemplateViewModel inputTmp = new TemplateViewModel();

inputTmp.FileName = cmd.First().InputTemplate.FileName;

inputTmp.FilePath = cmd.First().InputTemplate.FilePath;

inputTmp.ContentFile = cmd.First().InputTemplate.FileContent;

inputTmp.BookmarksFile = cmd.First().InputTemplate.FileBookmarks;

// отримати вихідний шаблон команди

TemplateViewModel outputTmp = new TemplateViewModel();

outputTmp.FileName = cmd.First().OutputTemplate.FileName;

outputTmp.FilePath = cmd.First().OutputTemplate.FilePath;

outputTmp.ContentFile = cmd.First().OutputTemplate.FileContent;

44165850.01365-01 12 01

20

outputTmp.BookmarksFile = cmd.First().OutputTemplate.FileBookmarks;

cvm.InputTemplate = inputTmp;

cvm.OutputTemplate = outputTmp;

cvm.CommandSetting = (Dictionary<string, string>)cmd.First().CommandSetting;

}

/// <summary>

/// Подія "Видалити команду"

/// </summary>

public void DeleteCommad(object sender, EventArgs e)

{

this.commandService.DeleteCommand(this.mainView.viewNameCommand);

}

/// <summary>

/// Подія "Зберегти команду"

/// </summary>

public async Task SaveCommand(object sender, EventArgs e)

{

var inputTmp = await templateService.CreateTemplate(mainView.viewCommand.InputTemplate.FileName,

mainView.viewCommand.InputTemplate.FilePath,

mainView.viewCommand.InputTemplate.ContentFile,

mainView.viewCommand.InputTemplate.BookmarksFile);

var outputTmp = await templateService.GetTemplate(mainView.viewCommand.OutputTemplate.FileName);

var cmd = await commandService.CreateCommand(this.mainView.viewCommand.NameCommand, this.mainView.viewCommand.CommandSetting,

inputTmp, outputTmp.Last());

\_ = this.commandService.SaveCommand(cmd);

}

/// <summary>

/// Подія "Оновити команду"

/// </summary>

public void UpdateCommand(object sender, EventArgs e, string oldName, CommandViewModel newCommand)

{

this.commandService.UpdateCommand(oldName, newCommand.NameCommand, newCommand.CommandSetting);

}

/// <summary>

/// Подія "Отримати шаблон"

/// </summary>

public async Task GetTemplate(object sender, TemplateViewModel e)

{

var tmp = await templateService.GetTemplate(e.FileName);

e.FilePath = tmp.First().FilePath;

e.ContentFile = tmp.First().FileContent;

e.BookmarksFile = tmp.First().FileBookmarks;

}

44165850.01365-01 12 01

21

/// <summary>

/// Подія "Оновити шаблон"

/// </summary>

public void UpdateTemplate(object? sender, EventArgs e, string oldName, TemplateViewModel newTemplate)

{

this.templateService.UpdateTemplate(oldName, newTemplate.FileName, newTemplate.BookmarksFile);

}

/// <summary>

/// Подія "Видалити шаблон"

/// </summary>

public void RemoveTemplate(object sender, EventArgs e)

{

this.templateService.DeleteTemplate(this.mainView.viewNameTemplate);

}

/// <summary>

/// Подія "Зберегти шаблон"

/// </summary>

public async Task SaveTemplate(object sender, EventArgs e)

{

var template = await this.templateService.CreateTemplate(this.mainView.viewTemplate.FileName,

this.mainView.viewTemplate.FilePath, this.mainView.viewTemplate.ContentFile,

this.mainView.viewTemplate.BookmarksFile);

\_ = templateService.Save(template);

}

public async Task Run()

{

var templates = await templateService.GetAllTemplates();

if (templates.Count() != 0)

{

var viewModelTemplates = templates.Select(t => new TemplateViewModel

{

FileName = t.FileName,

FilePath = Path.GetFullPath(t.FilePath),

BookmarksFile = t.FileBookmarks,

ContentFile = t.FileContent

}).ToList();

mainView.SetTemplateList(viewModelTemplates);

}

var commands = await commandService.GetAllCommands();

if(commands.Count() != 0)

{

var viewModelCommands = commands.Select(c => new CommandViewModel

{ NameCommand = c.CommandName, // назва команди

InputTemplate = new TemplateViewModel() // вхідний документ

{ FileName = c.InputTemplate.FileName,

FilePath = c.InputTemplate.FilePath,

ContentFile = c.InputTemplate.FileContent,

BookmarksFile= c.InputTemplate.FileBookmarks,

},

OutputTemplate = new TemplateViewModel() // вихідний шаблон

{

FileName = c.OutputTemplate.FileName,

44165850.01365-01 12 01

22

FilePath = c.OutputTemplate.FilePath,

ContentFile = c.OutputTemplate.FileContent,

BookmarksFile = c.OutputTemplate.FileBookmarks,

},

CommandSetting = (Dictionary<string, string>)c.CommandSetting // налаштування

}).ToList();

mainView.SetCommandsList(viewModelCommands);

}

mainView.Show();

}

}

}

Файл Extensions.cs

using Core;

using Core.Services;

using Microsoft.Extensions.DependencyInjection;

using Presentation.Presenters;

namespace Presentation

{

public static class Extensions

{

/// <summary>

/// Використати сервіси Core

/// </summary>

/// <param name="services">інтерфейс колекції сервісів</param>

public static void UseCore(this IServiceCollection services)

{

services.AddInMemoryRepositories();

AddServices(services);

AddPresenters(services);

}

/// <summary>

/// Додати сервіс

/// </summary>

/// <param name="services">інтерфейс колекції сервісів</param>

private static void AddServices(IServiceCollection services)

{

services.AddScoped<TemplateService>();

services.AddScoped<CommandService>();

}

/// <summary>

/// Додати перезентер

/// </summary>

/// <param name="services">інтерфейс колекції сервісів<</param>

private static void AddPresenters(IServiceCollection services)

{

services.AddScoped<IMainPresenter, MainPresenter>();

}

}

}

44165850.01365-01 12 01

23

* 1. **Проєкт WinFormsUI**

Загальний опис змісту: містить інтерфейс програми у вигляді WindowsForms компонентів. Розробник: Багнюк В.В.

Файл Program.cs

using Microsoft.Extensions.DependencyInjection;

using Microsoft.Extensions.Hosting;

using Presentation;

using Presentation.Presenters;

namespace WinFormsUI

{

internal static class Program

{

/// <summary>

/// The main entry point for the application.

/// </summary>

[STAThread]

static void Main()

{

ApplicationConfiguration.Initialize();

var host = CreateHostBuilder().Build();

var mainPresenter = host.Services.GetRequiredService<IMainPresenter>();

mainPresenter.Run();

Application.Run();

}

static IHostBuilder CreateHostBuilder()

{

return Host.CreateDefaultBuilder().ConfigureServices((\_, services) =>

{

services.AddViews();

services.UseCore();

});

}

}

}

Файл Extensions.cs

using Microsoft.Extensions.DependencyInjection;

using Presentation.Views;

namespace WinFormsUI

{ /// <summary>

/// Розширення

/// </summary>

public static class Extensions

{ /// <summary>

/// Додати представлення

/// </summary>

/// <param name="services"></param>

public static void AddViews(this IServiceCollection services)

{

services.AddScoped<IMainView, FormMain>();

}

}

}

44165850.01365-01 12 01

24

Файл CustomMessageBox.cs

namespace WinFormsUI

{

public static class CustomMessageBox

{

/// <summary>

/// Показати вікно повідомлення

/// </summary>

/// <param name="message">текст повідомлення</param>

/// <param name="caption">заголовок вікна</param>

/// <param name="buttons"> MessageBoxButtons кнопки (YesNo або OK)</param>

/// <returns></returns>

public static DialogResult Show(string message, string caption, MessageBoxButtons buttons)

{

DialogResult dialogResult = DialogResult.None;

switch (buttons)

{

case MessageBoxButtons.YesNo:

using (FormYesNo formYesNo = new FormYesNo())

{

formYesNo.Text = caption;

formYesNo.Message = message;

dialogResult = formYesNo.ShowDialog();

}

break;

case MessageBoxButtons.OK:

using (FormOK formOK = new FormOK())

{

formOK.Text = caption;

formOK.Message = message;

dialogResult = formOK.ShowDialog();

}

break;

}

return dialogResult;

}

}

}

Файл FormOK.cs

using MaterialSkin;

using MaterialSkin.Controls;

namespace WinFormsUI

{

public partial class FormOK : MaterialForm

{

public FormOK()

{

InitializeComponent();

var materialSkinManager = MaterialSkinManager.Instance;

materialSkinManager.AddFormToManage(this);

materialSkinManager.Theme = MaterialSkinManager.Themes.LIGHT;

}

public string Message

{

get { return LabelTextInfoMessage.Text; }

set { LabelTextInfoMessage.Text = value; }

}

}

}

44165850.01365-01 12 01

25

Файл FormYesNo.cs

using MaterialSkin;

using MaterialSkin.Controls;

namespace WinFormsUI

{

public partial class FormYesNo : MaterialForm

{

public FormYesNo()

{

InitializeComponent();

var materialSkinManager = MaterialSkinManager.Instance;

materialSkinManager.AddFormToManage(this);

materialSkinManager.Theme = MaterialSkinManager.Themes.LIGHT;

}

public string Message

{

get { return LabelTextMessage.Text; }

set { LabelTextMessage.Text = value; }

}

}

}

Файл FormTextData.cs

using MaterialSkin;

using MaterialSkin.Controls;

namespace WinFormsUI

{

public partial class FormTextData : MaterialForm

{

public string textData { get; set; }

public FormTextData()

{

InitializeComponent();

var materialSkinManager = MaterialSkinManager.Instance;

materialSkinManager.AddFormToManage(this);

materialSkinManager.Theme = MaterialSkinManager.Themes.LIGHT;

this.materialLabelMaxLengthSymbol.Text += " " + this.MultiLineTextBoxTextData.MaxLength.ToString();

}

private void MultiLineTextBoxTextData\_TextChanged(object sender, EventArgs e)

{

this.materialLabelCurrentCountSymbols.Text = this.MultiLineTextBoxTextData.Text.Length.ToString();

}

44165850.01365-01 12 01

26

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

{

if (this.MultiLineTextBoxTextData.Text.Length > 0)

this.MultiLineTextBoxTextData.Clear();

else

CustomMessageBox.Show("Поле для введення тексту вже очищене!", "Очищення", MessageBoxButtons.OK);

}

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

{

if (this.MultiLineTextBoxTextData.Text.Length > 0)

{

textData = this.MultiLineTextBoxTextData.Text;

CustomMessageBox.Show("Введений текст успішно збережено!", "Збереження", MessageBoxButtons.OK);

}

else

{

CustomMessageBox.Show("Для збереження заповніть текстове поле!", "Збереження", MessageBoxButtons.OK);

return;

}

this.Close();

}

}

}

Файл FormImageData.cs

using MaterialSkin;

using MaterialSkin.Controls;

namespace WinFormsUI

{

public partial class FormImageData : MaterialForm

{

public Image imageData { get; set; }

public FormImageData()

{

InitializeComponent();

var materialSkinManager = MaterialSkinManager.Instance;

materialSkinManager.AddFormToManage(this);

materialSkinManager.Theme = MaterialSkinManager.Themes.LIGHT;

}

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

{

OpenFileDialog open = new OpenFileDialog();

open.Filter = "Image Files(\*.jpg; \*.jpeg; \*.gif; \*.bmp)|\*.jpg; \*.jpeg; \*.gif; \*.bmp)";

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

{

pictureBoxImageData.Image = new Bitmap(open.FileName);

pictureBoxImageData.Tag = "imageData";

}

}

44165850.01365-01 12 01

27

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

{

if ((string)pictureBoxImageData.Tag != "imageData")

{

CustomMessageBox.Show("Завантажте зображення!", "Збереження зображення", MessageBoxButtons.OK);

return;

}

else

{

this.imageData = pictureBoxImageData.Image;

CustomMessageBox.Show("Зображення збережене успішно!", "Збереження зображення", MessageBoxButtons.OK);

}

this.Close();

}

}

}

Файл TableData.cs

namespace WinFormsUI

{

/// <summary>

/// Дані таблиці

/// (для форми FormTableData)

/// </summary>

public class TableData

{

public int numberRows { get; set; }

public int numberColumns { get; set; }

public string[,] arrayData { get; set; }

public TableData(int rows, int cols)

{

this.numberRows = rows;

this.numberColumns = cols;

this.arrayData = new string[rows, cols];

}

}

}

Файл FormTableData.cs

using MaterialSkin;

using MaterialSkin.Controls;

namespace WinFormsUI

{

public partial class FormTableData : MaterialForm

{

public TableData tableData { get; set; }

public FormTableData()

{

InitializeComponent();

var materialSkinManager = MaterialSkinManager.Instance;

materialSkinManager.AddFormToManage(this);

materialSkinManager.Theme = MaterialSkinManager.Themes.LIGHT;

this.tableData = new TableData(0, 0);

}

private void ShowTable()

{

44165850.01365-01 12 01

28

// виведення номерів стовпців таблиці і заборона сортування їх змісту

for (int i = 0; i < this.tableData.numberColumns; i++)

{

this.dataGridViewTableData.Columns[i].HeaderCell.Value = "C" + Convert.ToString(i + 1);

this.dataGridViewTableData.Columns[i].SortMode = DataGridViewColumnSortMode.NotSortable;

this.dataGridViewTableData.Columns[i].AutoSizeMode = DataGridViewAutoSizeColumnMode.Fill;

}

for (int i = 0; i < this.tableData.numberRows; i++)

{

this.dataGridViewTableData.Rows[i].HeaderCell.Value = Convert.ToString(i + 1);

for (int j = 0; j < this.tableData.numberColumns; ++j)

this.dataGridViewTableData.Rows[i].Cells[j].Value = Convert.ToString(this.tableData.arrayData[i, j]);

}

}

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

{

if (this.tableData.arrayData != null)

{

Array.Clear(this.tableData.arrayData, 0, this.tableData.arrayData.Length);

}

int numberRows = Convert.ToInt32(numericUpDownNumberRows.Value);

int numberColumns = Convert.ToInt32(numericUpDownNumberColumns.Value);

this.tableData = new TableData(numberRows, numberColumns);

this.dataGridViewTableData.RowCount = this.tableData.numberRows;

this.dataGridViewTableData.ColumnCount = this.tableData.numberColumns;

//заповнення таблиці

for (int i = 0; i < this.tableData.numberRows; i++)

for (int j = 0; j < this.tableData.numberColumns; j++)

this.tableData.arrayData[i, j] = "Text" + (i + j).ToString();

ShowTable();

}

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

{

if (this.dataGridViewTableData.Rows.Count == 0)

{

CustomMessageBox.Show("Створіть таблицю та заповніть її даними!", "Збереження даних таблиці", MessageBoxButtons.OK);

return;

}

else

{

CustomMessageBox.Show("Дані таблиці успішно збережені!", "Збереження даних таблиці", MessageBoxButtons.OK);

}

this.Close();

}

44165850.01365-01 12 01

29

private void dataGridViewTableData\_CellEndEdit(object sender, DataGridViewCellEventArgs e)

{

// індекси елемента матриці для поточной клітки таблиці

int i = this.dataGridViewTableData.CurrentCell.RowIndex;

int j = this.dataGridViewTableData.CurrentCell.ColumnIndex;

if (dataGridViewTableData.CurrentCell.Value == null)

{

CustomMessageBox.Show("Значення клітинки таблиці не має бути порожнім!", "Значення клітинок таблиці", MessageBoxButtons.OK);

}

else

{

string value = dataGridViewTableData.CurrentCell.Value.ToString() ?? throw new ArgumentNullException();

this.tableData.arrayData[i, j] = value;

}

}

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

{

if (this.dataGridViewTableData.Rows.Count == 0)

{

CustomMessageBox.Show("Таблиця для очищення не створена!", "Очищення", MessageBoxButtons.OK);

return;

}

for (int i = 0; i < this.tableData.numberRows; i++)

for (int j = 0; j < this.tableData.numberColumns; j++)

this.tableData.arrayData[i, j] = "";

ShowTable();

}

}

}

Файл FormEditTemplate.cs

using MaterialSkin;

using MaterialSkin.Controls;

namespace WinFormsUI

{

public partial class FormEditTemplate : MaterialForm

{

private string \_oldNameTemplate;

private Dictionary<string, string> \_newBookmarksTemplate;

private List<string> namesSavedTemplates;

public string oldNameTemplate { get => \_oldNameTemplate; set => \_oldNameTemplate = value; }

public string newNameTemplate { get => TextBoxTemplateName.Text; set => TextBoxTemplateName.Text = value; }

public Dictionary<string, string> newBookmarksTemplate { get => \_newBookmarksTemplate; set => \_newBookmarksTemplate = value; }

private bool flagSave;

public bool SavedChanges { get => flagSave; }

public FormEditTemplate(string templateName, Dictionary<string, string> templateBookmarks, List<string> namesSavedTemplates)

44165850.01365-01 12 01

30

{

InitializeComponent();

var materialSkinManager = MaterialSkinManager.Instance;

materialSkinManager.AddFormToManage(this);

materialSkinManager.Theme = MaterialSkinManager.Themes.LIGHT;

this.TextBoxTemplateName.Text = templateName;

oldNameTemplate = TextBoxTemplateName.Text;

flagSave = false;

DataGridViewComboBoxColumn theColumn = (DataGridViewComboBoxColumn)this.dataGridViewEditTableBookmarks.Columns[2];

theColumn.Items.Add("Текст");

theColumn.Items.Add("Зображення");

theColumn.Items.Add("Таблиця");

theColumn.DefaultCellStyle.NullValue = "Текст";

foreach (var itemBookmarks in templateBookmarks)

{

this.dataGridViewEditTableBookmarks.Rows.Add(0, itemBookmarks.Key);

}

this.namesSavedTemplates = namesSavedTemplates;

}

private void dataGridViewEditTableBookmarks\_RowPostPaint(object sender, DataGridViewRowPostPaintEventArgs e)

{

this.dataGridViewEditTableBookmarks.Rows[e.RowIndex].Cells["EditRowNumbers"].Value = (e.RowIndex + 1).ToString();

}

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

{

if (string.IsNullOrEmpty(this.TextBoxTemplateName.Text)

|| Path.GetExtension(this.TextBoxTemplateName.Text) != ".docx")

{

CustomMessageBox.Show("Перевірте ведення назви шаблону! Назва шаблону "

+ this.TextBoxTemplateName.Text + " має містити.docx наприкінці.",

"Редагування шаблону", MessageBoxButtons.OK);

return;

}

else

{

foreach (var name in namesSavedTemplates)

{

if (name == this.TextBoxTemplateName.Text)

{

CustomMessageBox.Show("Назва шаблону "

+ this.TextBoxTemplateName.Text + " немає співпадати зі вже збереженими шаблонами!",

"Редагування шаблону", MessageBoxButtons.OK);

return;

}

}

}

\_newBookmarksTemplate = new Dictionary<string, string>();

44165850.01365-01 12 01

31

for (int indexRow = 0; indexRow < this.dataGridViewEditTableBookmarks.Rows.Count; indexRow++)

{

string NameBookmark = dataGridViewEditTableBookmarks.Rows[indexRow].Cells[1].FormattedValue.ToString();

string SelectedType = dataGridViewEditTableBookmarks.Rows[indexRow].Cells[2].FormattedValue.ToString();

\_newBookmarksTemplate.Add(NameBookmark, SelectedType);

}

flagSave = true;

this.Close();

}

}

}

Файл FormEditCommand.cs

using MaterialSkin;

using MaterialSkin.Controls;

namespace WinFormsUI

{

public partial class FormEditCommand : MaterialForm

{

private Dictionary<string, string> newCmdSetting;

private Dictionary<string, string> inputBookmarks;

private List<string> NamesSavedCommands;

public string newNameCommand { get => this.TextBoxNewNameCommand.Text; set => this.TextBoxNewNameCommand.Text = value; }

public Dictionary<string, string> newCommandSetting { get => newCmdSetting; set => this.newCmdSetting = value; }

private bool flagSave;

public bool SavedChanges { get => flagSave; }

public FormEditCommand(string oldCmdName, Dictionary<string, string> inputDocBookmarks, Dictionary<string, string> oldCmdSetting,

List<string> namesSavedCommands)

{

InitializeComponent();

var materialSkinManager = MaterialSkinManager.Instance;

materialSkinManager.AddFormToManage(this);

materialSkinManager.Theme = MaterialSkinManager.Themes.LIGHT;

this.TextBoxNewNameCommand.Text = oldCmdName;

this.inputBookmarks = inputDocBookmarks;

this.flagSave = false;

this.NamesSavedCommands = namesSavedCommands;

// стовпець - вхідні закладки (документ)

DataGridViewComboBoxColumn cmbColBookmarks = (DataGridViewComboBoxColumn)this.dataGridViewEditBookmarkMatch.Columns[2];

// всі назви вхідних закладок (документ)

foreach (var inputBookmark in inputBookmarks.Keys)

{

cmbColBookmarks.Items.Add(inputBookmark);

}

44165850.01365-01 12 01

32

int j = 0;

foreach (var oldSetting in oldCmdSetting) // налаштування команди: key - вихід, value - вхід

{ // назва вибраної вхідної закладки (документ)

string defaultValue = oldSetting.Value;

this.dataGridViewEditBookmarkMatch.Rows.Add(0, oldSetting.Key); // назва вихідної закладки (шаблон)

this.dataGridViewEditBookmarkMatch.Rows[j].Cells[2].Value = defaultValue;

CmdEditInputBookmark.DefaultCellStyle.NullValue = defaultValue;

j++;

}

}

private void dataGridViewEditBookmarkMatch\_RowPostPaint(object sender, DataGridViewRowPostPaintEventArgs e)

{

this.dataGridViewEditBookmarkMatch.Rows[e.RowIndex].Cells["CmdEditNumberRow"].Value = (e.RowIndex + 1).ToString();

}

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

{

if (string.IsNullOrEmpty(this.TextBoxNewNameCommand.Text))

{

CustomMessageBox.Show("Перевірте ведену назву команди! Поле назви команди не має бути порожнім.",

"Редагування команди", MessageBoxButtons.OK);

return;

}

else

{

foreach (var name in NamesSavedCommands)

{

if (name == this.TextBoxNewNameCommand.Text)

{

CustomMessageBox.Show("Назва команди "

+ this.TextBoxNewNameCommand.Text + " немає співпадати зі вже збереженими командами!",

"Редагування команди", MessageBoxButtons.OK);

return;

}

}

this.newCmdSetting = new Dictionary<string, string>();

for (int indexRow = 0; indexRow < this.dataGridViewEditBookmarkMatch.Rows.Count; indexRow++)

{

string outputBoookmark = this.dataGridViewEditBookmarkMatch.Rows[indexRow].Cells[1].Value.ToString();

string SelectedInputBookmark = this.dataGridViewEditBookmarkMatch.Rows[indexRow].Cells[2].FormattedValue.ToString();

newCmdSetting.Add(outputBoookmark, SelectedInputBookmark);

}

flagSave = true;

this.Close();

}

}

}

}

44165850.01365-01 12 01

33

Файл FormMain.cs

using MaterialSkin;

using MaterialSkin.Controls;

using Microsoft.WindowsAPICodePack.Dialogs;

using Presentation.ViewModels;

using Presentation.Views;

using Spire.Doc;

using Spire.Doc.Documents;

namespace WinFormsUI

{

public partial class FormMain : MaterialForm, IMainView

{

static public int IndexRowCommandTable = 0; // індекс рядка таблиці шаблонів

static public int IndexRowTemplateTable = 0; // індекс рядка таблиці команд

private DataGridViewComboBoxColumn cmbGenTypeDataBookmarks; // тип даних закладки [модуль генерації]

private TemplateViewModel cmdInputDoc; // вхідний документ [модуль команд]

private TemplateViewModel genInputTemplate; // вхідний шаблон [модуль генерації]

private Dictionary<string, object> bookmarksData { get; set; } // дані закладок [модуль генерації]

// події для шаблонів

public event EventHandler? DeleteTemplate;

public event EventHandler? SaveTemplate;

public event EventHandler? UpdateTemplate;

public event EventHandler<TemplateViewModel> GetTemplate;

// події для команд

public event EventHandler? SaveCommand;

public event EventHandler? DeleteCommand;

public event EventHandler<CommandViewModel>? GetCommand;

public event EventHandler? UpdateCommand;

// поля для представлення

private string templateName;

private string commandName;

private TemplateViewModel templateViewModel;

private CommandViewModel commandViewModel;

public string viewNameTemplate { get => templateName; set => templateName = value; }

public TemplateViewModel viewTemplate { get => templateViewModel; set => templateViewModel = value; }

public CommandViewModel viewCommand { get => commandViewModel; set => commandViewModel = value; }

public string viewNameCommand { get => commandName; set => commandName = value; }

public FormMain()

{

InitializeComponent();

var materialSkinManager = MaterialSkinManager.Instance;

materialSkinManager.AddFormToManage(this);

materialSkinManager.Theme = MaterialSkinManager.Themes.LIGHT;

this.WindowState = FormWindowState.Maximized;

// [Модуль Генерації]

// додати у стовпець таблиці закладок типи даних

44165850.01365-01 12 01

34

cmbGenTypeDataBookmarks = (DataGridViewComboBoxColumn)this.dataGridViewGenSettingBookmarks.Columns[2];

cmbGenTypeDataBookmarks.Items.Add("Текст");

cmbGenTypeDataBookmarks.Items.Add("Зображення");

cmbGenTypeDataBookmarks.Items.Add("Таблиця");

this.TextBoxCmdInputDocument.ReadOnly = true;

this.bookmarksData = new Dictionary<string, object>();

this.ComboBoxGenCommandList.Enabled = false;

// [Модуль Команд]

CmdInputBookmark = (DataGridViewComboBoxColumn)this.dataGridViewCmdBookmarkMatch.Columns[2];

// представлення вхідного документа для команди

this.cmdInputDoc = new TemplateViewModel();

}

/// <summary>

/// Задати список команд

/// </summary>

/// <param name="commands">список команд</param>

public void SetCommandsList(IEnumerable<CommandViewModel> commands)

{

// materialComboBoxCmdList.Items.Add(commands.First().Name);

this.dataGridViewTableCommand.Rows.Insert(IndexRowCommandTable, 0, commands.Last().NameCommand,

commands.Last().InputTemplate.FileName, commands.Last().OutputTemplate.FileName);

IndexRowCommandTable++;

// генератор

this.ComboBoxGenCommandList.Items.Add(commands.Last().NameCommand);

this.ComboBoxGenCommandList.SelectedIndex = 0;

}

/// <summary>

/// Задати список шабонів

/// </summary>

/// <param name="templates">список шаблонів</param>

public void SetTemplateList(IEnumerable<TemplateViewModel> templates)

{

if (this.dataGridViewTableTemplate.Rows.Count == 0) // якщо шаблони не завантажені до таблиці

{

int counter = 0; // лічильник шаблонів в пам'яті

while (templates.Count() > counter)

{

// вставити рядок з інформацією про шаблон

dataGridViewTableTemplate.Rows.Insert(IndexRowTemplateTable, 0, templates.ElementAt(counter).FileName,

templates.ElementAt(counter).FilePath);

// додати списку збережених шаблонів в модулі "Генератор"

MaterialListBoxItem listBoxItem = new MaterialListBoxItem();

listBoxItem.Text = templates.ElementAt(counter).FileName;

ListBoxGenSavedTemplates.Items.Add(listBoxItem);

ListBoxGenSavedTemplates.SelectedIndex = 0;

// вихідний шаблон до модуля команд

this.ComboBoxCmdOutputTemplate.Items.Add(templates.ElementAt(counter).FileName);

this.ComboBoxCmdOutputTemplate.SelectedIndex = 0;

44165850.01365-01 12 01

35

IndexRowTemplateTable++; // збільшення індекса рядка таблиці шаблонів

counter++;

}

}

else

{

// вставка рядка до таблиці з інформацією про шаблон

dataGridViewTableTemplate.Rows.Insert(IndexRowTemplateTable, 0, templates.Last().FileName,

templates.Last().FilePath);

IndexRowTemplateTable++;

this.ComboBoxCmdOutputTemplate.Items.Add(templates.Last().FileName);

// генератор

MaterialListBoxItem listBoxItem = new MaterialListBoxItem();

listBoxItem.Text = templates.Last().FileName;

ListBoxGenSavedTemplates.Items.Add(listBoxItem);

ListBoxGenSavedTemplates.SelectedIndex = 0;

}

}

/// <summary>

/// [Модуль шаблонів]

/// Показати закладки шаблону

/// </summary>

/// <param name="dictionaryBookmarks">словник закладок

/// (ключ: ім'я закладки, значення: тип закладки)</param>

public void SetBookmarksDictionary(IDictionary<string, string> dictionaryBookmarks)

{ // очишення та оновлення таблиці закладок

this.dataGridViewTableBookmarks.Rows.Clear();

this.dataGridViewTableBookmarks.Refresh();

foreach (var itemBookmarks in dictionaryBookmarks)

{ // додаваня рядків до таблиці закладок

this.dataGridViewTableBookmarks.Rows.Add(0, itemBookmarks.Key, itemBookmarks.Value);

}

}

/// <summary>

/// Перевірка назви шаблону

/// на коректність

/// </summary>

/// <param name="NameTemplate">назва шаблону</param>

/// <returns>true - якщо назва коректна, або false - якщо назва некоректна</returns>

public bool checkNameTemplate(string NameTemplate)

{

if (string.IsNullOrEmpty(NameTemplate) || Path.GetExtension(NameTemplate) != ".docx")

{

return false;

}

return true;

}

// перевірка чи знаходиться початок і кінець закладки

// в одному TextBody (для таблиць)

private bool checkIsSameTextBody(Document doc)

{

foreach (Bookmark bookmark in doc.Bookmarks)

44165850.01365-01 12 01

36

{

var start = bookmark.BookmarkStart.OwnerParagraph.OwnerTextBody;

var end = bookmark.BookmarkEnd.OwnerParagraph.OwnerTextBody;

if (!object.Equals(start, end))

return false;

}

return true;

}

/// <summary>

/// Отримати словник закладок з документу

/// (ключ - назва закладки, значення - тип даних (текст за замовчуванням))

/// </summary>

/// <param name="pathFile">шлях до файлу</param>

/// <returns></returns>

private Dictionary<string, string> GetBookmarksFromDoc(string pathFile)

{

Document doc = new Document(pathFile);

if (doc.Bookmarks.Count == 0)

{

CustomMessageBox.Show("Шаблон не містить закладок! Додайте закладки до шаблону.", "Повідомлення", MessageBoxButtons.OK);

return null;

}

// словник закладок

Dictionary<string, string> dictionaryBookmarks = new Dictionary<string, string>();

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

{

dictionaryBookmarks.Add(doc.Bookmarks[i].Name, "Текст");

}

return dictionaryBookmarks;

}

/// <summary>

/// [Модуль шаблонів]

/// Видалення шаблону

/// </summary>

private void dataGridViewTableTemplate\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{ // назва стовпця таблиці шаблонів

string columnTableTemplate = dataGridViewTableTemplate.Columns[e.ColumnIndex].Name;

if (columnTableTemplate == "RemoveTemplate")

{

if (this.dataGridViewTableTemplate.Rows[e.RowIndex].Cells[0].Value == null)

{

CustomMessageBox.Show("Спочатку додайте шаблон до поточного рядка або виберіть інший зі списку.",

"Видалення шаблону", MessageBoxButtons.OK);

return;

}

DialogResult dialogResult = CustomMessageBox.Show("Ви впевнені, що хочете видалити шаблон? Видалення скасувати неможливо.",

"Видалення шаблону", MessageBoxButtons.YesNo);

if (dialogResult == DialogResult.Yes) // якщо видалити шаблон

44165850.01365-01 12 01

37

{

if (dataGridViewTableTemplate.SelectedRows.Count == 0) // якщо шаблон не вибрано

{

CustomMessageBox.Show("Для видалення виберіть шаблон зі списку.", "Видалення шаблону", MessageBoxButtons.OK);

}

else

{ // якщо шаблон вибрано

int index = dataGridViewTableTemplate.CurrentCell.RowIndex;

templateName = dataGridViewTableTemplate.Rows[index].Cells[1].Value.ToString() ?? throw new ArgumentNullException();

DeleteTemplate?.Invoke(sender, e);

this.dataGridViewTableTemplate.Rows.RemoveAt(index);

IndexRowTemplateTable--;

if (this.dataGridViewTableBookmarks.Rows.Count > 0)

{

this.dataGridViewTableBookmarks.Rows.Clear();

this.dataGridViewTableBookmarks.Refresh();

}

this.ListBoxGenSavedTemplates.Items.RemoveAt(index);

this.dataGridViewGenSettingBookmarks.Rows.Clear();

this.dataGridViewGenSettingBookmarks.Refresh();

this.ComboBoxCmdOutputTemplate.Items.RemoveAt(index);

}

}

}

}

/// <summary>

/// [Модуль шаблонів]

/// Додавання шаблонів

/// </summary>

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

{

// відкрити діалогове вікно з вибором файлу .docx

OpenFileDialog ofd = new OpenFileDialog

{

Filter = "Word|\*.docx"

};

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

{

FileInfo fileInfo = new FileInfo(ofd.FileName); // назва файлу

foreach (DataGridViewRow row in this.dataGridViewTableTemplate.Rows)

{

string columnValue = row.Cells[1].Value.ToString() ?? throw new ArgumentNullException(); ;

if (columnValue == fileInfo.Name)

{

CustomMessageBox.Show("Шаблон має бути з унікальним іменем.", "Повідомлення", MessageBoxButtons.OK);

return;

}

}

templateViewModel = new TemplateViewModel();

44165850.01365-01 12 01

38

templateViewModel.FileName = fileInfo.Name;

templateViewModel.FilePath = fileInfo.FullName;

templateViewModel.ContentFile = File.ReadAllBytes(fileInfo.FullName);

templateViewModel.BookmarksFile = GetBookmarksFromDoc(fileInfo.FullName);

// збереження шаблона в пам'яті

SaveTemplate?.Invoke(sender, e);

// додати шаблон до списку для показу

List<TemplateViewModel> templates = new List<TemplateViewModel>

{

templateViewModel // Template

};

this.SetTemplateList(templates);

}

}

/// <summary>

/// [Модуль шаблонів]

/// Редагування шаблона

/// </summary>

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

{

if (dataGridViewTableTemplate.SelectedRows.Count == 0)

{

CustomMessageBox.Show("Для редагування виберіть шаблон зі списку.", "Повідомлення", MessageBoxButtons.OK);

}

else

{

int indexTmp = dataGridViewTableTemplate.CurrentCell.RowIndex; // індекс вибраного шаблону

// назва шаблона та шлях до файлу

string templateName = this.dataGridViewTableTemplate.Rows[indexTmp].Cells[1].Value.ToString() ?? throw new ArgumentNullException();

string templatePath = this.dataGridViewTableTemplate.Rows[indexTmp].Cells[2].Value.ToString() ?? throw new ArgumentNullException();

List<string> namesTemplates = new List<string>();

foreach (DataGridViewRow row in this.dataGridViewTableTemplate.Rows)

{

string name = row.Cells[1].Value.ToString() ?? throw new ArgumentNullException();

if (name != templateName)

namesTemplates.Add(name);

}

// форма для редагування

FormEditTemplate formEditTemplate = new FormEditTemplate(templateName,

this.GetBookmarksFromDoc(templatePath), namesTemplates);

formEditTemplate.ShowDialog();

if (!formEditTemplate.SavedChanges)

return;

this.templateName = formEditTemplate.oldNameTemplate;

templateViewModel = new TemplateViewModel();

templateViewModel.FileName = formEditTemplate.newNameTemplate;

templateViewModel.BookmarksFile = formEditTemplate.newBookmarksTemplate;

// оновлення шаблону в пам'яті

UpdateTemplate?.Invoke(sender, e);

44165850.01365-01 12 01

39

this.dataGridViewTableTemplate.Rows[indexTmp].Cells[1].Value = formEditTemplate.newNameTemplate;

this.SetBookmarksDictionary(formEditTemplate.newBookmarksTemplate);

this.ListBoxGenSavedTemplates.Items[indexTmp].Text = formEditTemplate.newNameTemplate;

this.ComboBoxCmdOutputTemplate.Items[indexTmp] = formEditTemplate.newNameTemplate;

}

}

/// <summary>

/// [Модуль шаблонів]

/// Читання шаблонів (відображення закладок)

/// </summary>

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

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

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

{

if (dataGridViewTableTemplate.SelectedRows.Count == 0)

{

CustomMessageBox.Show("Для редагування виберіть шаблон зі списку.", "Повідомлення", MessageBoxButtons.OK);

}

else

{

int index = dataGridViewTableTemplate.CurrentCell.RowIndex;

string nameTemplate = this.dataGridViewTableTemplate.Rows[index].Cells[1].Value.ToString() ?? throw new ArgumentNullException();

this.templateViewModel = new TemplateViewModel();

templateViewModel.FileName = nameTemplate;

GetTemplate?.Invoke(sender, viewTemplate);

if (viewTemplate != null)

this.SetBookmarksDictionary(viewTemplate.BookmarksFile);

else

CustomMessageBox.Show("Шаблон не знайдено.", "Повідомлення", MessageBoxButtons.OK);

}

}

/// <summary>

/// [Модуль шаблонів]

/// Пошук шаблона за назвою

/// </summary>

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

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

private void TextBoxSearchTemplate\_TrailingIconClick(object sender, EventArgs e)

{

string nameTemplate = this.TextBoxSearchTemplate.Text;

if (!checkNameTemplate(nameTemplate))

{

CustomMessageBox.Show("Перевірте ведення назви шаблону! Назва шаблону " + nameTemplate + " має містити .docx наприкінці.",

"Пошук шаблону", MessageBoxButtons.OK);

return;

}

else

44165850.01365-01 12 01

40

{

if (this.dataGridViewTableTemplate.Rows.Count == 0)

{ // якщо доданих шаблонів немає

CustomMessageBox.Show("Шаблони для пошуку відсутні! Додайте шаблони.", "Пошук шаблону", MessageBoxButtons.OK);

return;

}

int rowCount = dataGridViewTableTemplate.RowCount - 1;

int i = 0; // лічильник рядків

while (i <= rowCount)

{

if (dataGridViewTableTemplate.Rows[i].Cells["NameFile"].Value.ToString() == nameTemplate)

{ // виділити знайдений шаблон

dataGridViewTableTemplate.CurrentCell = dataGridViewTableTemplate.Rows[i].Cells[0];

return;

}

i++;

}

CustomMessageBox.Show("Шаблон з назвою " + nameTemplate + " НЕ знайдено!", "Пошук шаблону", MessageBoxButtons.OK);

}

}

private void dataGridViewTableTemplate\_RowPostPaint(object sender, DataGridViewRowPostPaintEventArgs e)

{ // нумерація рядків таблиці шаблонів

this.dataGridViewTableTemplate.Rows[e.RowIndex].Cells["NumberRows"].Value = (e.RowIndex + 1).ToString();

}

private void dataGridViewTableBookmarks\_RowPostPaint(object sender, DataGridViewRowPostPaintEventArgs e)

{ // нумерація рядків таблиці закладок

this.dataGridViewTableBookmarks.Rows[e.RowIndex].Cells["ColumnNumber"].Value = (e.RowIndex + 1).ToString();

}

/// <summary>

/// [Модуль генерації]

/// Вибір шляху для збереження

/// сформованого документа

/// </summary>

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

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

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

{

CommonOpenFileDialog dialog = new CommonOpenFileDialog();

dialog.InitialDirectory = "C:\\Users";

dialog.IsFolderPicker = true;

if (dialog.ShowDialog() == CommonFileDialogResult.Ok)

{

this.TextBoxGenPathSaveDocument.Text = dialog.FileName;

}

}

private void dataGridViewGenSettingBookmarks\_RowPostPaint(object sender, DataGridViewRowPostPaintEventArgs e)

44165850.01365-01 12 01

41

{

this.dataGridViewGenSettingBookmarks.Rows[e.RowIndex].Cells["GenNumberRows"].Value = (e.RowIndex + 1).ToString();

}

/// <summary>

/// [Модуль генерації]

/// Завантаження шаблона

/// </summary>

private void TextBoxGenUploadTemplate\_TrailingIconClick(object sender, EventArgs e)

{

// відкрити діалогове вікно з вибором файлу .docx

OpenFileDialog ofd = new OpenFileDialog

{

Filter = "Word|\*.docx"

};

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

{

// інформація про файл

FileInfo fileInfo = new FileInfo(ofd.FileName);

if (ListBoxGenSavedTemplates.Items.Count > 0)

{

MaterialListBoxItem item = new MaterialListBoxItem();

item.Text = ofd.FileName;

var isContains = ListBoxGenSavedTemplates.Items.Contains(item);

if (isContains)

{

CustomMessageBox.Show("Назва шаблона не має співпадати з вже завантаженим шаблоном! Змініть назву одного з файлів.",

"Завантаження шаблона", MessageBoxButtons.OK);

return;

}

}

this.TextBoxGenUploadTemplate.Text = fileInfo.Name;

Document doc = new Document(fileInfo.FullName);

if (doc.Bookmarks.Count == 0)

{ // якщо в документі немає закладок

CustomMessageBox.Show("Шаблон не містить закладок! Додайте закладки до шаблону.",

"Завантаження шаблона", MessageBoxButtons.OK);

return;

}

// словник закладок

Dictionary<string, string> dictionaryBookmarks = new Dictionary<string, string>();

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

{

dictionaryBookmarks.Add(doc.Bookmarks[i].Name, "Текст");

}

genInputTemplate = new TemplateViewModel();

genInputTemplate.FileName = fileInfo.Name;

genInputTemplate.BookmarksFile = dictionaryBookmarks;

genInputTemplate.FilePath = fileInfo.FullName;

genInputTemplate.ContentFile = File.ReadAllBytes(fileInfo.FullName);

44165850.01365-01 12 01

42

if (this.bookmarksData.Count > 0)

this.bookmarksData.Clear();

this.dataGridViewGenSettingBookmarks.Rows.Clear();

this.dataGridViewGenSettingBookmarks.Refresh();

int indexRow = 0;

foreach (var item in genInputTemplate.BookmarksFile)

{

string val = item.Value;

this.dataGridViewGenSettingBookmarks.Rows.Add(0, item.Key);

this.dataGridViewGenSettingBookmarks.Rows[indexRow].Cells[2].Value = val;

cmbGenTypeDataBookmarks.DefaultCellStyle.NullValue = val;

indexRow++;

}

}

}

/// <summary>

/// [Модуль генерації]

/// Задання даних для закладок шаблона

/// залежно від типу (текст, зображення, таблиця)

/// </summary>

private void dataGridViewGenSettingBookmarks\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

string columnTableBookmark = this.dataGridViewGenSettingBookmarks.Columns[e.ColumnIndex].Name;

if (columnTableBookmark == "GenEnterData")

{

string typeData = dataGridViewGenSettingBookmarks.Rows[e.RowIndex].Cells[2].FormattedValue.ToString()

?? throw new ArgumentNullException();

if (typeData == "Текст")

{

string nameBookmark = dataGridViewGenSettingBookmarks.Rows[e.RowIndex].Cells[1].FormattedValue.ToString()

?? throw new ArgumentNullException();

FormTextData formTextData = new FormTextData();

formTextData.ShowDialog();

if (this.bookmarksData.ContainsKey(nameBookmark))

{

this.bookmarksData[nameBookmark] = formTextData.textData;

} else

this.bookmarksData.Add(nameBookmark, formTextData.textData);

}

else if (typeData == "Зображення")

{

string nameBookmark = dataGridViewGenSettingBookmarks.Rows[e.RowIndex].Cells[1].FormattedValue.ToString()

?? throw new ArgumentNullException();

FormImageData formImageData = new FormImageData();

formImageData.ShowDialog();

if (this.bookmarksData.ContainsKey(nameBookmark))

{

this.bookmarksData[nameBookmark] = formImageData.imageData;

} else

this.bookmarksData.Add(nameBookmark, formImageData.imageData);

}

44165850.01365-01 12 01

43

else if (typeData == "Таблиця")

{

string nameBookmark = dataGridViewGenSettingBookmarks.Rows[e.RowIndex].Cells[1].FormattedValue.ToString()

?? throw new ArgumentNullException(); ;

FormTableData formTableData = new FormTableData();

formTableData.ShowDialog();

if (this.bookmarksData.ContainsKey(nameBookmark))

{

this.bookmarksData[nameBookmark] = formTableData.tableData;

} else

this.bookmarksData.Add(nameBookmark, formTableData.tableData);

}

}

}

/// <summary>

/// [Модуль генерації]

/// Створення таблиці для

/// вставки у Word документ

/// </summary>

/// <param name="document">документ</param>

/// <param name="tableInsert">дані таблиці для заповнення</param>

/// <returns>таблиця</returns>

Table CreateTableForWord(Document document, TableData tableInsert)

{

Table table = new Table(document, true);

table.ResetCells(tableInsert.numberRows, tableInsert.numberColumns);

for (int i = 0; i < tableInsert.numberRows; i++)

{

TableRow row = table.Rows[i];

for (int j = 0; j < tableInsert.numberColumns; j++)

{

TableCell cell = row.Cells[j];

Paragraph para = cell.AddParagraph();

para.AppendText(tableInsert.arrayData[i, j]);

}

}

return table;

}

/// <summary>

/// [Модуль генерації]

/// Створення документа на основі шаблона

/// </summary>

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

{

string pathFile = this.TextBoxGenPathSaveDocument.Text.ToString(); // шлях для збрежеження

string nameFile = this.TextBoxGenNameOutputDocument.Text; // назва вихідного файлу

if (string.IsNullOrEmpty(pathFile) || (checkNameTemplate(nameFile) == false)

|| (this.dataGridViewGenSettingBookmarks.Rows.Count == 0))

{

CustomMessageBox.Show("Документ не сформовано! Можливо, не вказаний шлях для збереження, " +

"або некорекна назва вихідного документа чи не показані закладки шаблона.", "Формування документа", MessageBoxButtons.OK);

return;

44165850.01365-01 12 01

44

}

else

{

if (SwitchGenUseCommand.Checked)

{ // формування документа з використанням даних команди

int indexSelectedCommand = ComboBoxGenCommandList.SelectedIndex;

string nameSelectedCommand = ComboBoxGenCommandList.Items[indexSelectedCommand].ToString()

?? throw new ArgumentNullException();

this.commandViewModel = new CommandViewModel();

commandViewModel.NameCommand = nameSelectedCommand;

this.GetCommand?.Invoke(sender, viewCommand);

CustomMessageBox.Show("Назва: " + viewCommand.NameCommand + ", Вхідний документ: " +

viewCommand.InputTemplate.FileName + ", Вихідний шаблон: " + viewCommand.OutputTemplate.FileName,

"Вибрана команда", MessageBoxButtons.OK);

// вхідний документ

Document inputDocument = new Document();

MemoryStream streamInput = new MemoryStream(viewCommand.InputTemplate.ContentFile);

inputDocument.LoadFromStream(streamInput, FileFormat.Docx);

// вихідний шаблон

Document outputTemplate = new Document();

MemoryStream streamOutput = new MemoryStream(viewCommand.OutputTemplate.ContentFile);

outputTemplate.LoadFromStream(streamOutput, FileFormat.Docx);

// навігатори по закладкам вхідного документа та вихідного шаблона відповідно

BookmarksNavigator navigatorInput = new BookmarksNavigator(inputDocument);

BookmarksNavigator navigatorOutput = new BookmarksNavigator(outputTemplate);

foreach (var setting in viewCommand.CommandSetting)

{

navigatorInput.MoveToBookmark(setting.Value); // до вхідної закладки

TextBodyPart inTextBodyPart = navigatorInput.GetBookmarkContent();

navigatorOutput.MoveToBookmark(setting.Key); // до вихідної закладки

navigatorOutput.ReplaceBookmarkContent(inTextBodyPart); // замінити

}

string fullpath = pathFile + @"\" + nameFile;

outputTemplate.SaveToFile(fullpath, FileFormat.Docx);

CustomMessageBox.Show("Документ успішно створено! Шлях: " + fullpath,

"Формування документа", MessageBoxButtons.OK);

}

else

{

44165850.01365-01 12 01

45

Document inputTemplate = new Document();

if (materialTabControlGenSelectTemplate.SelectedTab == tabPageGenSaveTemplate)

{

//CustomMessageBox.Show("Ви вибрали шаблон зі списку збережених шаблонів", "Вибір шаблона", MessageBoxButtons.OK);

if (this.bookmarksData.Count != this.dataGridViewGenSettingBookmarks.Rows.Count)

{

CustomMessageBox.Show("Задайте дані для всіх закладок шаблона!",

"Формування документа", MessageBoxButtons.OK);

return;

}

int indexSelectedTemplate = ListBoxGenSavedTemplates.SelectedIndex;

this.templateViewModel = new TemplateViewModel();

templateViewModel.FileName = ListBoxGenSavedTemplates.Items[indexSelectedTemplate].Text;

this.GetTemplate.Invoke(sender, viewTemplate);

// bookmarksData: key - закладка, value - дані

MemoryStream streamInput = new MemoryStream(viewTemplate.ContentFile);

inputTemplate.LoadFromStream(streamInput, FileFormat.Docx);

}

else if (materialTabControlGenSelectTemplate.SelectedTab == tabPageGenLoadTemplate)

{

// CustomMessageBox.Show("Ви вибрали завантажений шаблон", "Вибір шаблона", MessageBoxButtons.OK);

if (string.IsNullOrEmpty(genInputTemplate.FileName))

{

CustomMessageBox.Show("Завантажте шаблон або виберіть зі списку збережених!",

"Вибір шаблона", MessageBoxButtons.OK);

return;

}

else if (this.bookmarksData.Count != this.dataGridViewGenSettingBookmarks.Rows.Count)

{

CustomMessageBox.Show("Задайте дані для всіх закладок шаблона!",

"Формування документа", MessageBoxButtons.OK);

return;

}

MemoryStream streamInput = new MemoryStream(genInputTemplate.ContentFile);

inputTemplate.LoadFromStream(streamInput, FileFormat.Docx);

}

BookmarksNavigator navigatorInput = new BookmarksNavigator(inputTemplate);

foreach (var item in bookmarksData)

{

navigatorInput.MoveToBookmark(item.Key);

44165850.01365-01 12 01

46

if (item.Value.GetType() == typeof(string)) // якщо дані текст

{

if (navigatorInput.GetBookmarkContent() != null)

{

navigatorInput.ReplaceBookmarkContent(item.Value.ToString(), true);

}

else

navigatorInput.InsertText(item.Value.ToString());

}

else if (item.Value.GetType() == typeof(System.Drawing.Bitmap)) // якщо дані зображення

{

Section section0 = inputTemplate.AddSection();

Paragraph paragraph = section0.AddParagraph();

paragraph.AppendPicture((System.Drawing.Image)item.Value);

// navigatorInput.InsertParagraph(paragraph);

TextBodyPart textBodyPart = new TextBodyPart(inputTemplate);

textBodyPart.BodyItems.Add(paragraph);

navigatorInput.ReplaceBookmarkContent(textBodyPart);

}

else if (item.Value.GetType() == typeof(TableData)) // якщо дані таблиця

{

Table table = CreateTableForWord(inputTemplate, (TableData)item.Value);

TextBodyPart part = new TextBodyPart(inputTemplate);

part.BodyItems.Add(table);

// navigatorInput.InsertTextBodyPart(part);

navigatorInput.ReplaceBookmarkContent(part);

}

}

string fullpath = pathFile + @"\" + nameFile;

inputTemplate.SaveToFile(fullpath, FileFormat.Docx);

CustomMessageBox.Show("Документ успішно створено! Шлях: " + fullpath,

"Формування документа", MessageBoxButtons.OK);

}

}

}

/// <summary>

/// [Модуль генерації]

/// TabControl вибору шаблона

/// серед збережених або завантажити

/// </summary>

private void TabControlGenSelectTemplate\_SelectedIndexChanged(object sender, EventArgs e)

{

if (materialTabControlGenSelectTemplate.SelectedTab == tabPageGenSaveTemplate)

{

if (this.dataGridViewTableTemplate.Rows.Count > 0)

{

this.dataGridViewGenSettingBookmarks.Refresh();

this.ListBoxGenSavedTemplates\_SelectedValueChanged(sender, null);

44165850.01365-01 12 01

47

}

}

else if (materialTabControlGenSelectTemplate.SelectedTab == tabPageGenLoadTemplate)

{

if (bookmarksData.Count > 0)

{

var result = CustomMessageBox.Show("Увага! Попередньо задані дані для закладок поточного шаблону буде видалено!" +

"Продовжити?", "Попередження", MessageBoxButtons.YesNo);

if (result == DialogResult.Yes)

this.bookmarksData.Clear();

else

return;

}

this.dataGridViewGenSettingBookmarks.Rows.Clear();

this.TextBoxGenUploadTemplate.Text = string.Empty;

}

}

/// <summary>

/// [Модуль генерації]

/// Список збережених шаблонів

/// </summary>

private void ListBoxGenSavedTemplates\_SelectedValueChanged(object sender, MaterialListBoxItem selectedItem)

{

int indexSelectedTemplate = ListBoxGenSavedTemplates.SelectedIndex;

if (bookmarksData.Count > 0)

{

var result = CustomMessageBox.Show("Увага! Попередньо задані дані для закладок поточного шаблону буде видалено!" +

" Продовжити?", "Попередження", MessageBoxButtons.YesNo);

if (result == DialogResult.Yes)

{

this.bookmarksData.Clear();

}

else

{

if (indexSelectedTemplate > 0)

ListBoxGenSavedTemplates.SelectedIndex = indexSelectedTemplate - 1;

return;

}

}

this.dataGridViewGenSettingBookmarks.Rows.Clear();

this.dataGridViewGenSettingBookmarks.Refresh();

this.templateViewModel = new TemplateViewModel();

templateViewModel.FileName = ListBoxGenSavedTemplates.Items[indexSelectedTemplate].Text;

// отримати шаблон з бази даних

this.GetTemplate.Invoke(sender, viewTemplate);

int indexRow = 0;

foreach (var item in viewTemplate.BookmarksFile)

{

string val = item.Value.ToString();

this.dataGridViewGenSettingBookmarks.Rows.Add(0, item.Key);

44165850.01365-01 12 01

48

this.dataGridViewGenSettingBookmarks.Rows[indexRow].Cells[2].Value = val;

cmbGenTypeDataBookmarks.DefaultCellStyle.NullValue = val;

indexRow++;44165850.01365-01 12 01

}

}

/// <summary>

/// [Модуль команд]

/// завантажити вхідний документ

/// </summary>

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

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

private void TextBoxCmdInputDocument\_TrailingIconClick(object sender, EventArgs e)

{

// відкрити діалогове вікно з вибором файлу .docx

OpenFileDialog ofd = new OpenFileDialog

{

Filter = "Word|\*.docx"

};

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

{

// інформація про файл

FileInfo fileInfo = new FileInfo(ofd.FileName);

if (ComboBoxCmdOutputTemplate.Items.Count > 0)

{

var isContains = ComboBoxCmdOutputTemplate.Items.Contains(ofd.FileName);

if (isContains)

{

CustomMessageBox.Show("Назва документа не має співпадати з вже завантаженим шаблоном! Змініть назву одного з файлів.",

"Завантаження документа", MessageBoxButtons.OK);

return;

}

}

this.TextBoxCmdInputDocument.Text = fileInfo.Name;

Document doc = new Document(fileInfo.FullName);

if (doc.Bookmarks.Count == 0)

{ // якщо в документі немає закладок

CustomMessageBox.Show("Шаблон не містить закладок! Додайте закладки до шаблону.", "Завантаження документа", MessageBoxButtons.OK);

return;

}

if (!checkIsSameTextBody(doc))

{

CustomMessageBox.Show("Початок і кінець закладки має бути в одному текстовому тілі! " +

"Перевірте закладку до таблиці: початок має бути на один рядок вище таблиці, а кінець - нижче.",

"Завантаження документа",

MessageBoxButtons.OK);

this.TextBoxCmdInputDocument.Text = "";

return;

}

44165850.01365-01 12 01

49

// словник закладок

Dictionary<string, string> dictionaryBookmarks = new Dictionary<string, string>();

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

{

dictionaryBookmarks.Add(doc.Bookmarks[i].Name, "Текст");

}

cmdInputDoc.FileName = fileInfo.Name;

cmdInputDoc.BookmarksFile = dictionaryBookmarks;

cmdInputDoc.FilePath = fileInfo.FullName;

cmdInputDoc.ContentFile = File.ReadAllBytes(fileInfo.FullName);

}

}

/// <summary>

/// [Модуль команд]

/// Показ закладок (налаштувань команди)

/// </summary>

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

{

if (this.dataGridViewCmdBookmarkMatch.Rows.Count > 0)

{

DialogResult dialogResult = CustomMessageBox.Show("Закладки вже показані." +

" Якщо ви не створили з наявними параметрами конвертування команду, то вони будуть видалені. Продовжити?",

"Показ закладок", MessageBoxButtons.YesNo);

if (dialogResult == DialogResult.Yes)

{

CmdInputBookmark.Items.Clear();

this.dataGridViewCmdBookmarkMatch.Rows.Clear();

this.dataGridViewCmdBookmarkMatch.Refresh();

}

else

return;

}

else

{

int indexOutputTemplate = ComboBoxCmdOutputTemplate.SelectedIndex;

if (TextBoxCmdInputDocument.Text.Length == 0)

{

CustomMessageBox.Show("Спочатку завантажте заповнений шаблон!", "Показ закладок", MessageBoxButtons.OK);

return;

}

this.templateViewModel = new TemplateViewModel();

string nameSelectedTemplate = ComboBoxCmdOutputTemplate.Items[indexOutputTemplate].ToString()

?? throw new ArgumentNullException();

templateViewModel.FileName = nameSelectedTemplate;

GetTemplate?.Invoke(sender, viewTemplate);

int countOutput = viewTemplate.BookmarksFile.Count;

int countInput = cmdInputDoc.BookmarksFile.Count();

if (countOutput != countInput)

{

CustomMessageBox.Show("Кількість закладок у вхідному документі та вихідному шаблоні має бути однакова! " +

"Вхідний документ має " + countInput.ToString() + " закладки(ок), а вихідний шаблон " + countOutput.ToString(),

44165850.01365-01 12 01

50

"Показ закладок", MessageBoxButtons.OK);

return;

}

DataGridViewComboBoxColumn inputBookmarks = (DataGridViewComboBoxColumn)this.dataGridViewCmdBookmarkMatch.Columns[2];

foreach (var input in cmdInputDoc.BookmarksFile.Keys)

{

inputBookmarks.Items.Add(input);

inputBookmarks.DefaultCellStyle.NullValue = inputBookmarks.Items[0].ToString();

}

foreach (var output in viewTemplate.BookmarksFile.Keys)

{

this.dataGridViewCmdBookmarkMatch.Rows.Add(0, output);

}

}

}

private void dataGridViewCmdBookmarkMatch\_RowPostPaint(object sender, DataGridViewRowPostPaintEventArgs e)

{

this.dataGridViewCmdBookmarkMatch.Rows[e.RowIndex].Cells["CmdNumberRow"].Value = (e.RowIndex + 1).ToString();

}

private void dataGridViewTableCommand\_RowPostPaint(object sender, DataGridViewRowPostPaintEventArgs e)

{

this.dataGridViewTableCommand.Rows[e.RowIndex].Cells["CmdNumberRowCommand"].Value = (e.RowIndex + 1).ToString();

}

/// <summary>

/// [Модуль команд]

/// Створення команди

/// </summary>

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

{

string commandName = this.TextBoxCmdCommandName.Text;

int indexOutputTemplate = ComboBoxCmdOutputTemplate.SelectedIndex;

if(this.dataGridViewCmdBookmarkMatch.Rows.Count == 0) {

CustomMessageBox.Show("Задайте налаштування команди!", "Створення команди", MessageBoxButtons.OK);

return;

}

if(this.dataGridViewTableBookmarks.Rows.Count > 0)

{

foreach (DataGridViewRow row in this.dataGridViewTableCommand.Rows)

{

string name = row.Cells[1].Value.ToString() ?? throw new ArgumentNullException();

if (commandName == name)

{

CustomMessageBox.Show("Команда з такою назвою вже існує!", "Створення команди", MessageBoxButtons.OK);

return;

}

44165850.01365-01 12 01

51

}

}

Dictionary<string, string> cmdSetting = new Dictionary<string, string>();

for (int indexRow = 0; indexRow < this.dataGridViewCmdBookmarkMatch.Rows.Count; indexRow++)

{

string OutputBookmark = this.dataGridViewCmdBookmarkMatch.Rows[indexRow].Cells[1].Value.ToString() ?? throw new ArgumentNullException();

string InputBookmark = this.dataGridViewCmdBookmarkMatch.Rows[indexRow].Cells[2].FormattedValue.ToString() ?? throw new ArgumentNullException();

if (string.Equals(OutputBookmark, InputBookmark))

{

CustomMessageBox.Show("Назви вхідної та вихідної закладки не мають співпадати! Або виберіть закладку з іншою назвою, " +

" або змініть назву закладки в " + this.TextBoxCmdInputDocument.Text + " та завантажте його знову.",

"Створення команди", MessageBoxButtons.OK);

return;

}

cmdSetting.Add(OutputBookmark, InputBookmark);

}

this.templateViewModel = new TemplateViewModel();

string nameSelectedTemplate = ComboBoxCmdOutputTemplate.Items[indexOutputTemplate].ToString() ?? throw new ArgumentNullException();

templateViewModel.FileName = nameSelectedTemplate;

GetTemplate?.Invoke(sender, viewTemplate);

this.commandViewModel = new CommandViewModel();

commandViewModel.NameCommand = commandName;

commandViewModel.InputTemplate = cmdInputDoc;

commandViewModel.OutputTemplate = viewTemplate;

commandViewModel.CommandSetting = cmdSetting;

SaveCommand?.Invoke(sender, e);

List<CommandViewModel> command = new List<CommandViewModel>()

{ viewCommand };

this.SetCommandsList(command);

}

private void dataGridViewTableCommand\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

// індекс стовпця таблиці команд

string columnTableCommand = this.dataGridViewTableCommand.Columns[e.ColumnIndex].Name;

if (columnTableCommand == "CmdDeleteCommand")

{

DialogResult dialogResult = CustomMessageBox.Show("Ви впевнені, що хочете видалити команду? Видалення скасувати неможливо.",

"Видалення команди", MessageBoxButtons.YesNo);

if (dialogResult == DialogResult.Yes) // якщо видалити команду

{

44165850.01365-01 12 01

52

if (this.dataGridViewTableCommand.SelectedRows.Count == 0) //

{

CustomMessageBox.Show("Для видалення виберіть команду зі списку.", "Видалення команди", MessageBoxButtons.OK);

}

else

{ // видалення

int index = this.dataGridViewTableCommand.CurrentCell.RowIndex;

viewNameCommand = this.dataGridViewTableCommand.Rows[index].Cells[1].Value.ToString() ?? throw new ArgumentNullException();

commandViewModel = new CommandViewModel();

commandViewModel.NameCommand = viewNameCommand;

GetCommand?.Invoke(sender, viewCommand);

DeleteCommand?.Invoke(sender, e);

// видалення з таблиці

this.dataGridViewTableCommand.Rows.RemoveAt(index);

IndexRowCommandTable--;

// видалення зі списку команд у модулі генерації

this.ComboBoxGenCommandList.Items.RemoveAt(index);

this.ComboBoxGenCommandList.SelectedIndex = 0;

}

}

}

}

/// <summary>

/// [Модуль генерації]

/// Перемикач використання даних команди

/// </summary>

private void SwitchGenUseCommand\_CheckedChanged(object sender, EventArgs e)

{

if (this.SwitchGenUseCommand.Checked)

this.ComboBoxGenCommandList.Enabled = true;

else

this.ComboBoxGenCommandList.Enabled = false;

}

/// <summary>

/// [Модуль команд]

/// Редагування команди

/// </summary>

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

{

if (this.dataGridViewTableCommand.SelectedRows.Count == 0)

{

CustomMessageBox.Show("Для редагування виберіть команду з таблиці.", "Редагування команди", MessageBoxButtons.OK);

}

else

{

int indexCmd = this.dataGridViewTableCommand.CurrentCell.RowIndex; // індекс вибраної команди

44165850.01365-01 12 01

53

this.commandName = this.dataGridViewTableCommand.Rows[indexCmd].Cells[1].Value.ToString() ?? throw new ArgumentNullException();

commandViewModel = new CommandViewModel();

commandViewModel.NameCommand = commandName;

GetCommand?.Invoke(sender, viewCommand);

List<string> namesCommand = new List<string>();

foreach (DataGridViewRow row in this.dataGridViewTableCommand.Rows)

{

string name = row.Cells[1].Value.ToString() ?? throw new ArgumentNullException();

if (viewCommand.NameCommand != name)

{

namesCommand.Add(name);

}

}

FormEditCommand formEditCommand = new FormEditCommand(viewCommand.NameCommand, viewCommand.InputTemplate.BookmarksFile,

viewCommand.CommandSetting, namesCommand);

formEditCommand.ShowDialog();

if (!formEditCommand.SavedChanges)

return;

commandViewModel = new CommandViewModel();

commandViewModel.NameCommand = formEditCommand.newNameCommand;

commandViewModel.CommandSetting = formEditCommand.newCommandSetting;

UpdateCommand?.Invoke(sender, e);

this.dataGridViewTableCommand.Rows[indexCmd].Cells[1].Value = viewCommand.NameCommand;

this.ComboBoxGenCommandList.Items[indexCmd] = viewCommand.NameCommand;

}

}

}

}

# **ДОДАТОК Г**

**Керівництво користувача**

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖУЮ

Проректор Українського

державного університету

науки і технологій

Анатолій РАДКЕВИЧ

ПРОГРАМА ДЛЯ ФОРМУВАННЯ WORD   
ДОКУМЕНТІВ ЗА ШАБОНАМИ

Керівництво користувача

ЛИСТ ЗАТВЕРДЖЕННЯ

44165850.01365-01 I3 01-ЛЗ

Завідувач кафедри КІТ

Вадим ГОРЯЧКІН

Керівник розробки

Олександр ЖЕВАГО

Виконавець

Вероніка БАГНЮК

Нормоконтролер

Світлана ВОЛКОВА

2024

ЗАТВЕРДЖЕНО

44165850.01365-01 I3 01-ЛЗ

ПРОГРАМА ДЛЯ ФОРМУВАННЯ WORD   
ДОКУМЕНТІВ ЗА ШАБОНАМИ

Керівництво користувача

Листів 14

2024

44165850.01365-01 I3 01

2

АНОТАЦІЯ

Документ 44165850.01365-01 I3 01-ЛЗ «Програма для формування Word документів за шаблонами. Керівництво користувача» входить до складу програмної документації на комплекс з п'яти проєктів, що реалізовують створення Word документів за шаблонами.

У даному документі наведені рекомендації з користування програмою. Програми в проєктах написані на мові C#. Об'єм пам'яті, що займають програми комплексу, складає 0,98 МБ. Конфігурація комп'ютера стандартна. Комплекс функціонує в середовищі операційних систем Windows 10 та наступних версіях.

44165850.01365-01 I3 0

3

ЗМІСТ

1. Введення4
2. Призначення та умови застосування5
3. Підготовка до роботи6
4. Опис операцій7
5. Аварійні ситуації13

44165850.01365-01 I3 0

4

1. ВВЕДЕННЯ

Програмний продукт, розроблений за темою «Розробка програми для формування Word документів за шаблонами», може бути застосований для створення документів у достатньо широкому переліку різноманітних областей, таких як:

* навчально-методичні матеріали: вчителі та викладачі можуть розробляти методичні вказівки, робочі програми, силабуси тощо;
* технічна документація: розробники та тестувальники можуть створювати проектну або іншу програмну документацію;
* особисті документи: окремі особи отримають можливість формувати документи за шаблонами для особистого використання, наприклад, резюме, візитівка, запрошення на проведення події тощо.

В загальному випадку можливий будь-який випадок застосування, якщо він потребує створення документа із заздалегідь визначеним форматуванням та вмістом.

Програма надає можливість створення Word документа, базуючись на шаблоні — іншому документі, який розмічений закладками (спеціальними позначками з унікальним ім’ям, що дозволяють швидко перейти до позначених ними місць). За допомогою програмного функціоналу їм задається тип даних: текст, зображення або таблиця. Також наявна можливість створювати команди перетворення одного документа на інший, шляхом завантаження заповненого документа із закладками та вибору вихідного шаблона.

Робота з програмою вимагає від користувача базових навичок користування персональним комп’ютером (прості маніпуляції в операційній системі Windows) та роботи з програмним забезпеченням для обробки текстів, а саме текстовим процесором Microsoft Office Word.

44165850.01365-01 I3 0

5

1. ПРИЗНАЧЕННЯ ТА УМОВИ ЗАСТОСУВАННЯ

Функціональне призначення програми наступне: формування Word документів за шаблонами як шляхом введення даних (тексту, зображення або таблиці) для закладок шаблона, так і використанням даних команд, створених завантаженням вхідного заповненого документа й вибором вихідного шаблона, що визначають які елементи документа перенести до шаблона.

Експлуатаційне призначення програми наступне: впровадження розробки забезпечить часткову автоматизацію ведення документації, що дозволить зменшити часові витрати на формування й заповнення документів, скоротивши кількість помилок при роботі.

Програма розроблена для використання в операційній системі Windows 10 та вище. Для експлуатації необхідний персональний комп’ютер з наступними мінімальними характеристиками:

* процесор: Intel(R) Celeron(R) CPU (1.8 – 2 ГГц);
* пам’ять: RAM 8 ГБ ;
* графічне ядро: Inlel (R) HD Graphics;
* порт: USB-порт.

44165850.01365-01 I3 0

6

1. ПІДГОТОВКА ДО РОБОТИ

Для користування програмним продуктом необхідно:

* мати папку «DocumentGenerator» з файлами програми;
* Запустити файл WinFormsUI.exe за приблизним шляхом DocumentGenerator\WinFormsUI\bin\Debug\net6.0-windows\WinFormsUI.exe.

44165850.01365-01 I3 0

7

1. ОПИС ОПЕРАЦІЙ

Програма поділена на три модулі: шаблони, генератор та команди. При запуску програми за замовчуванням відкривається модуль шаблонів (рис.1). Через натискання на символ з трьох горизонтальних рисок у лівому верхньому куті відкривається меню, через яке можна перейти до вкладок інших двох модулів (рис.2).

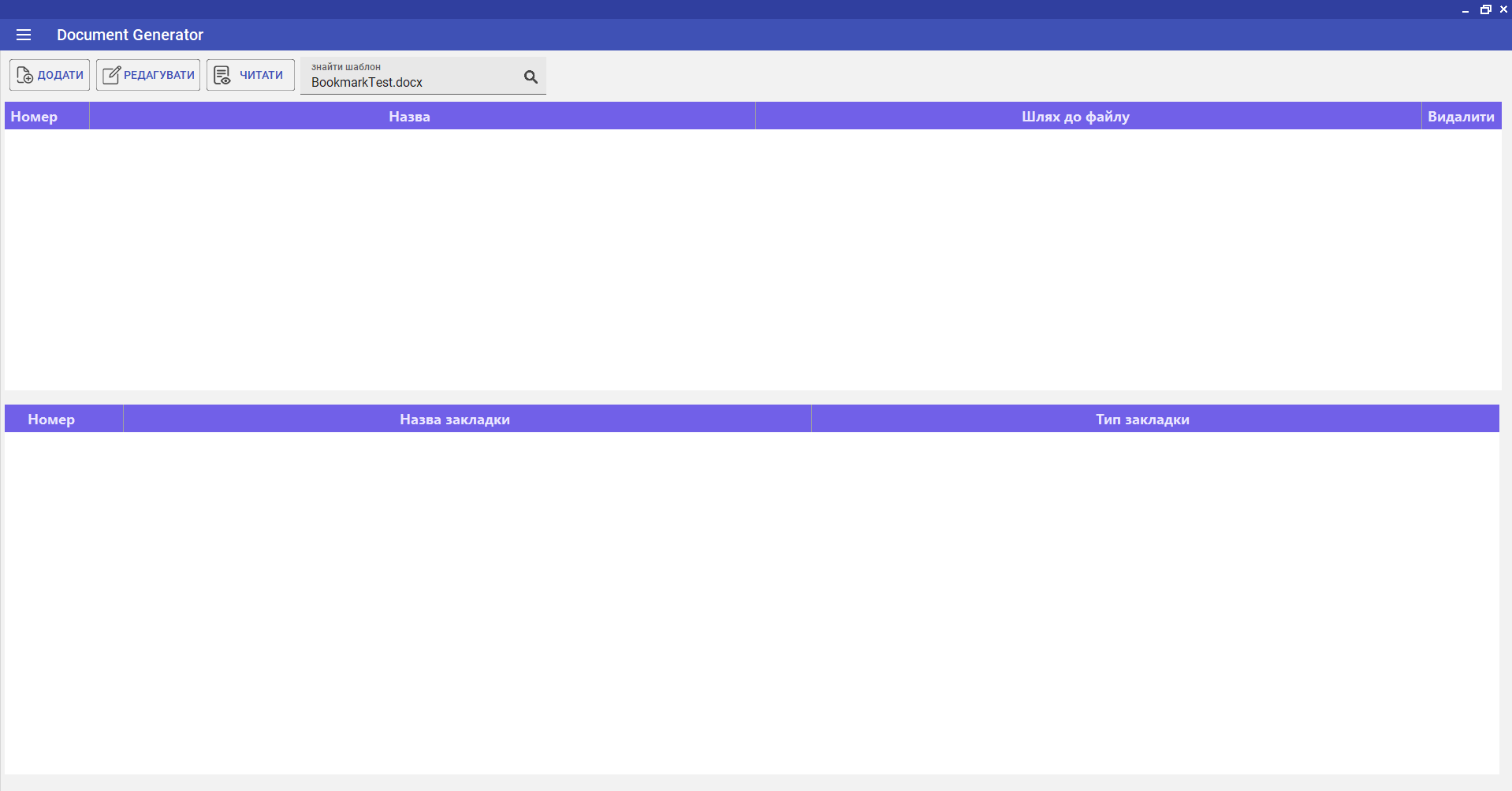


Рис.1. Модуль шаблонів при першому запуску

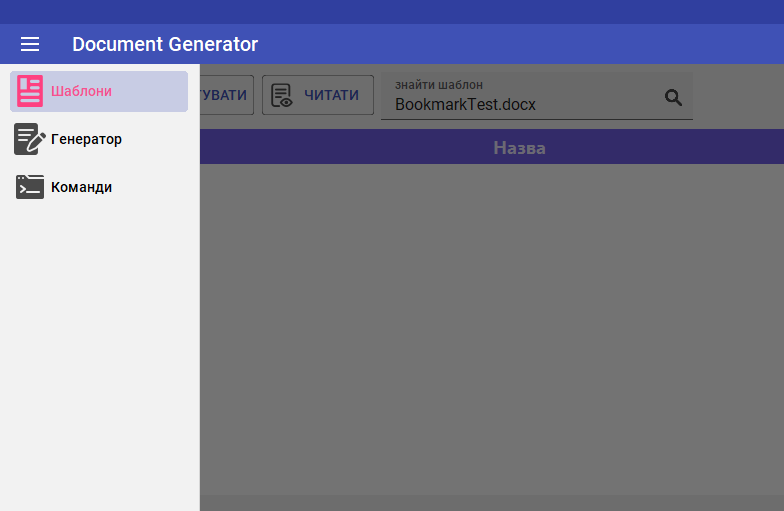


Рис.2. Головне меню

44165850.01365-01 I3 0

8

**Вкладка «Шаблони».** У модулі шаблонів доступі наступні функції: додавання, редагування, читання (перегляд закладок та їхніх типів) та пошук (за назвою) шаблонів. Для додавання шаблону необхідно натиснути на кнопку «Додати» (див. рис.1), після чого за допомогою відповідного вікна (рис.3) вибирається Word документ та натискається кнопка «Відкрити».

Примітка. Рекомендовано використовувати документи, які містять не більше 500 абзаців та 25 таблиць, інакше решта змісту не збережеться у програмі.

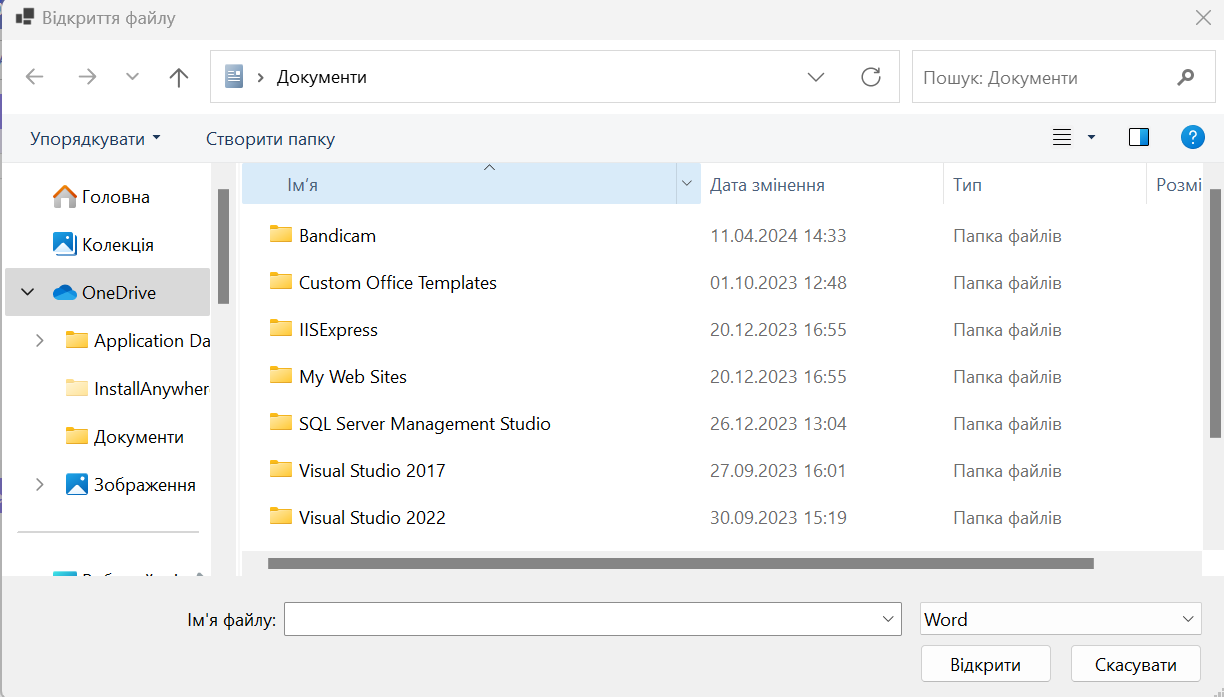


Рис.3. Приклад вікна відкриття Word файлу

В результаті попередньої дії отримаємо доданий шаблон. Для перегляду закладок шаблона вибирається шаблон з таблиці (шляхом натискання кнопкою миші) та натискається кнопка «Читати». На рис.4 наведений приклад доданого шаблона та перегляд його закладок.

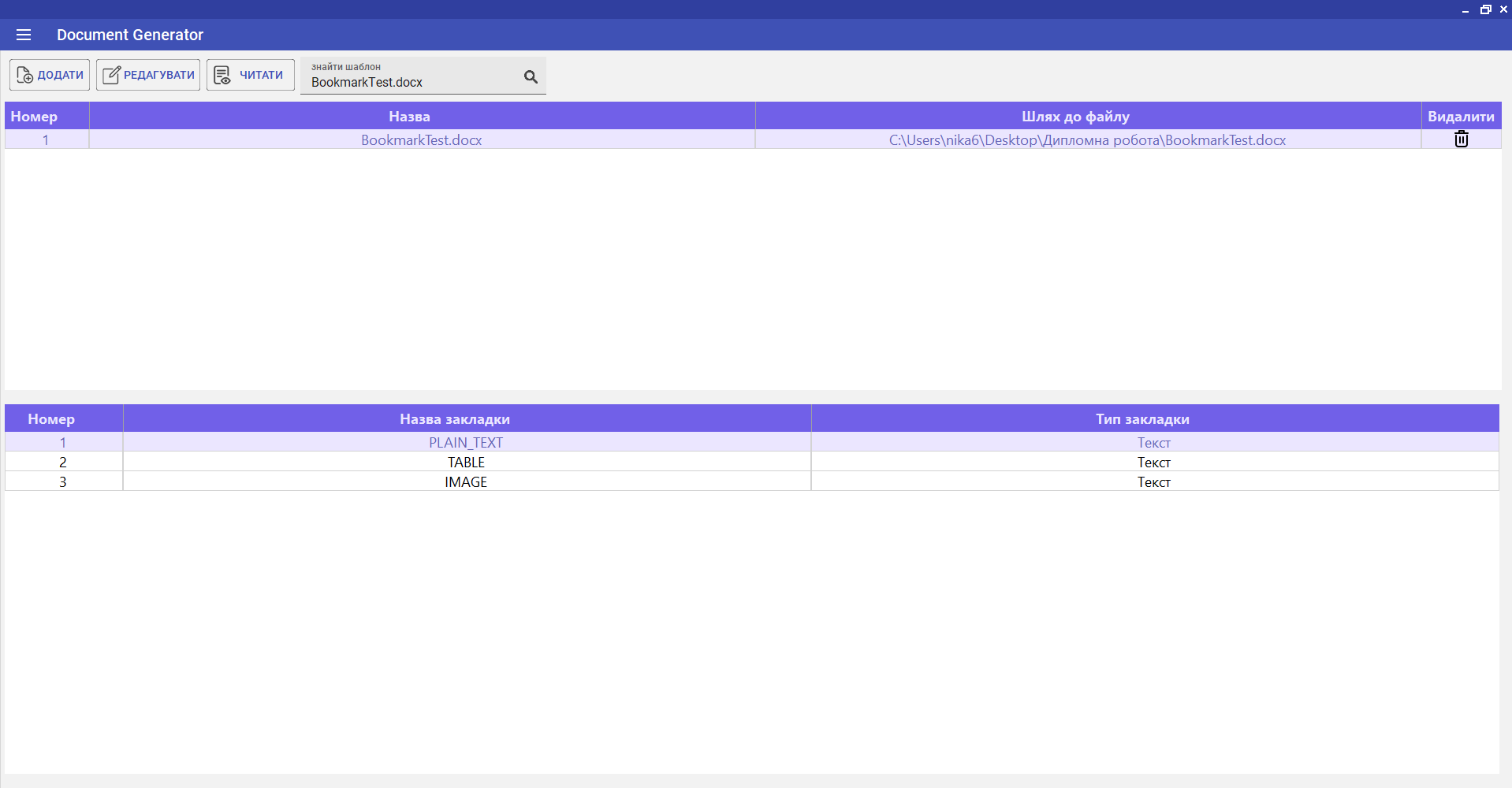


Рис.4. Доданий шаблона та перегляд його закладок

44165850.01365-01 I3 0

9

Для редагування шаблону вибирається шаблон та натискається кнопка «Редагувати» (див. рис.4). У вікні редагування можна задати нову назву шаблона та налаштувати тип закладок (рис.5). Для збереження змін натискається кнопка «Зберегти оновлення».

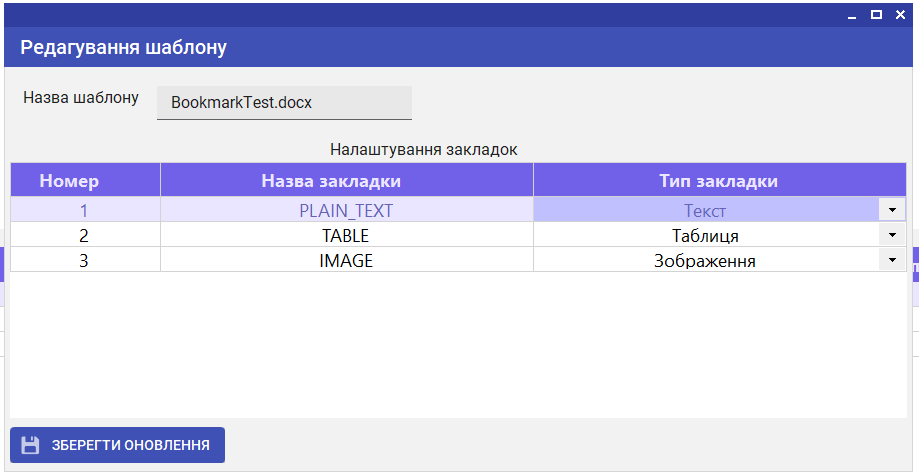


Рис.5. Редагування шаблона

Для видалення шаблону натискається відповідна піктограма у стовпці «Видалити» на потрібному рядку (див. рис.4).

Для пошуку шаблона необхідно ввести назву шаблона у відповідне поле з назвою «знайти шаблон» та натиснути піктограму лупи. В результаті рядок з потрібним шаблона виділяється світлим кольором в таблиці шаблонів (див. рис.4).

**Вкладка «Команди».** Щоб перейти до модуля команд необхідно натиснути на вкладку «Команди» у головному меню (див. рис.2). Модуль команд дозволяє створювати, редагувати та видаляти команди.

Для створення команди необхідно виконати наступні дії:

* заповнити поля блоку «Налаштування команди», а саме: ввести назву команди в поле «назва команди»; завантажити вхідний документ, шляхом натискання на відповідну піктограму поля «вхідний документ»; вибрати вихідний шаблон зі списку шаблонів у відповідному полі;

44165850.01365-01 I3 0

10

* натиснути кнопку «Показати закладки» після попереднього кроку та налаштувати заміну закладок відповідної таблиці у блоці «Параметри конвертування»;
* натиснути кнопку «Створити».

В результаті виконання цих кроків отримаємо доданий шаблон до таблиці. На рис.6. наведено приклад заповнення даних команди та її створення.

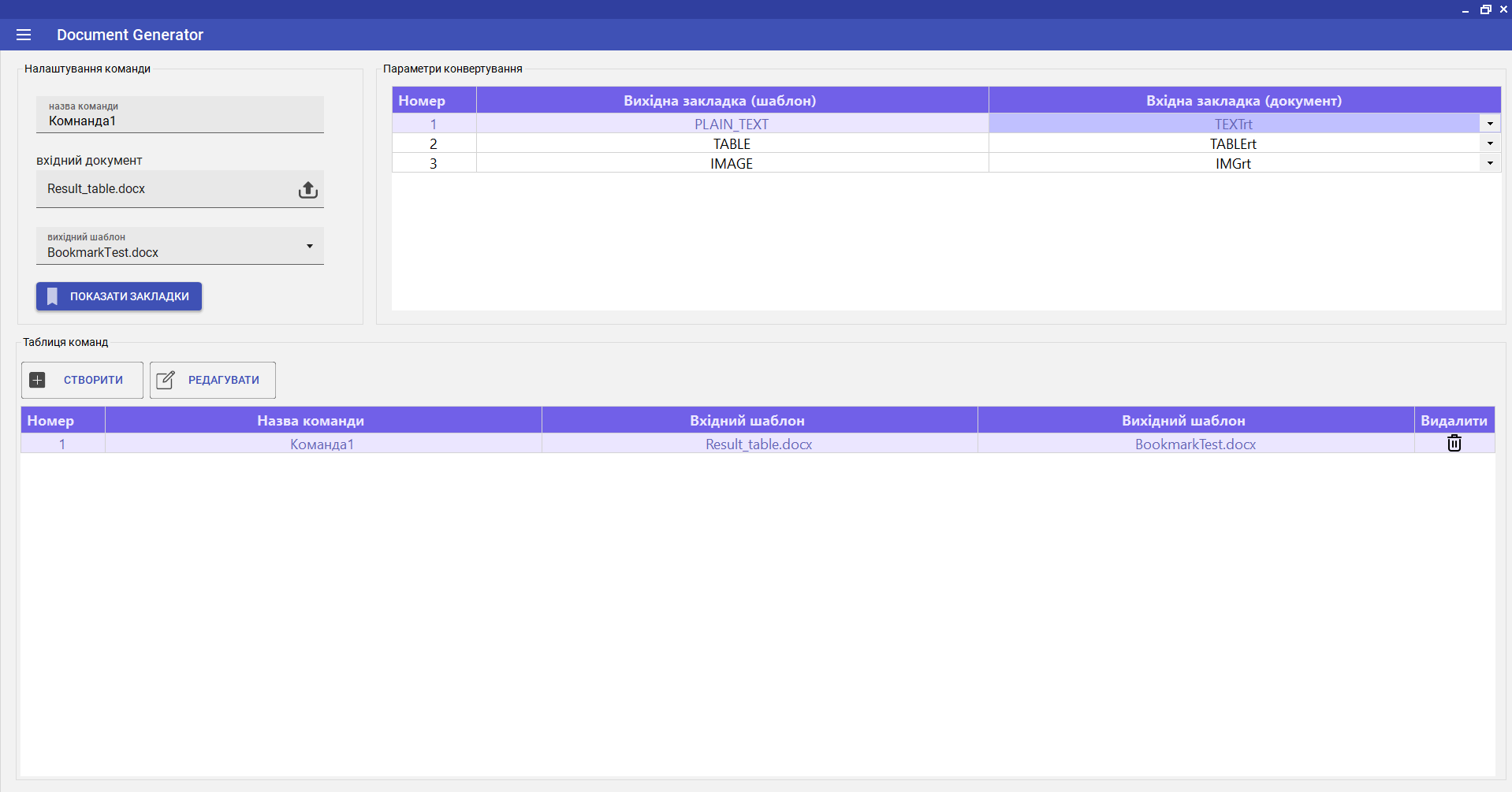


Рис.6. Приклад заповнення даних для команди та її створення

Для редагування команди необхідно вибрати шаблон з таблиці та натиснути кнопку «Редагувати». У вікні редагування можна задати нову назву команди та налаштувати заміну закладок (рис.7). Для збереження змін натискається кнопка «Зберегти зміни».

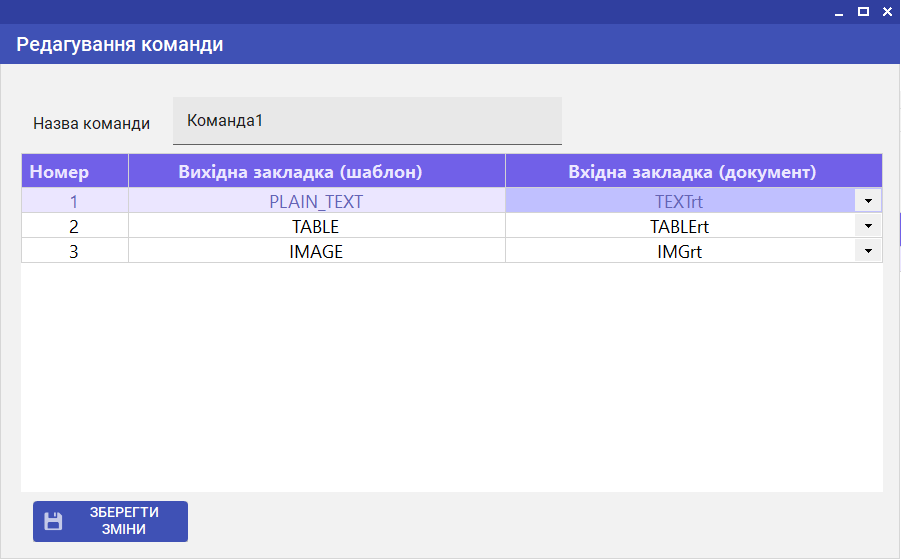


Рис.7. Приклад редагування команди

44165850.01365-01 I3 0

11

Для видалення команди натискається відповідна піктограма у стовпці «Видалити» на потрібному рядку.

**Вкладка «Генератор».** Щоб перейти до модуля генерації необхідно натиснути на вкладку «Генератор» у головному меню (див. рис.2). Модуль генерації дозволяє сформувати документ за збереженим чи завантаженим шаблоном, шляхом задання закладкам даних відповідного типу, або використати дані команди.

Для формування документа за збереженим шаблоном необхідно виконати наступні дії:

1. у блоці «Вибір шаблона» натиснути вкладку «Збережені шаблони» та вибрати у списку потрібний. Або натиснути «Завантажити шаблон» і завантажити новий шаблон, шляхом натискання на піктограму відповідного поля;
2. у блоці «Задання даних» в таблиці вибрати тип даних та ввести дані, натиснувши піктограму у стовпці «Дані» потрібного рядка. Дані необхідно ввести для всіх закладок шаблона;
3. у блоці «Налаштування» задати назву вихідного документа та шлях для його забережня у відповідних полях. Для вибору шляху необхідно натиснути кнопку «Вибрати папку»;
4. у блоці «Задання даних» натиснути кнопку «Сформувати документ».

Для формування документа з використанням даних команди необхідно у блоці «Задання команд» увімкнути перемикач «використати дані команди» і вибрати потрібну команду зі списку, та повторити дії 3 – 4 попереднього кроку.

На рис.8 наведено приклад для модуля генератора (генерації), де вибрано збережений шаблона, вказано вихідну назву документа та шлях для його збереження.

На рис.9 наведено приклад для модуля генератора (генерації), де вибрано використати дані команди та показана вкладка «Завантажити шаблон».

44165850.01365-01 I3 0

12

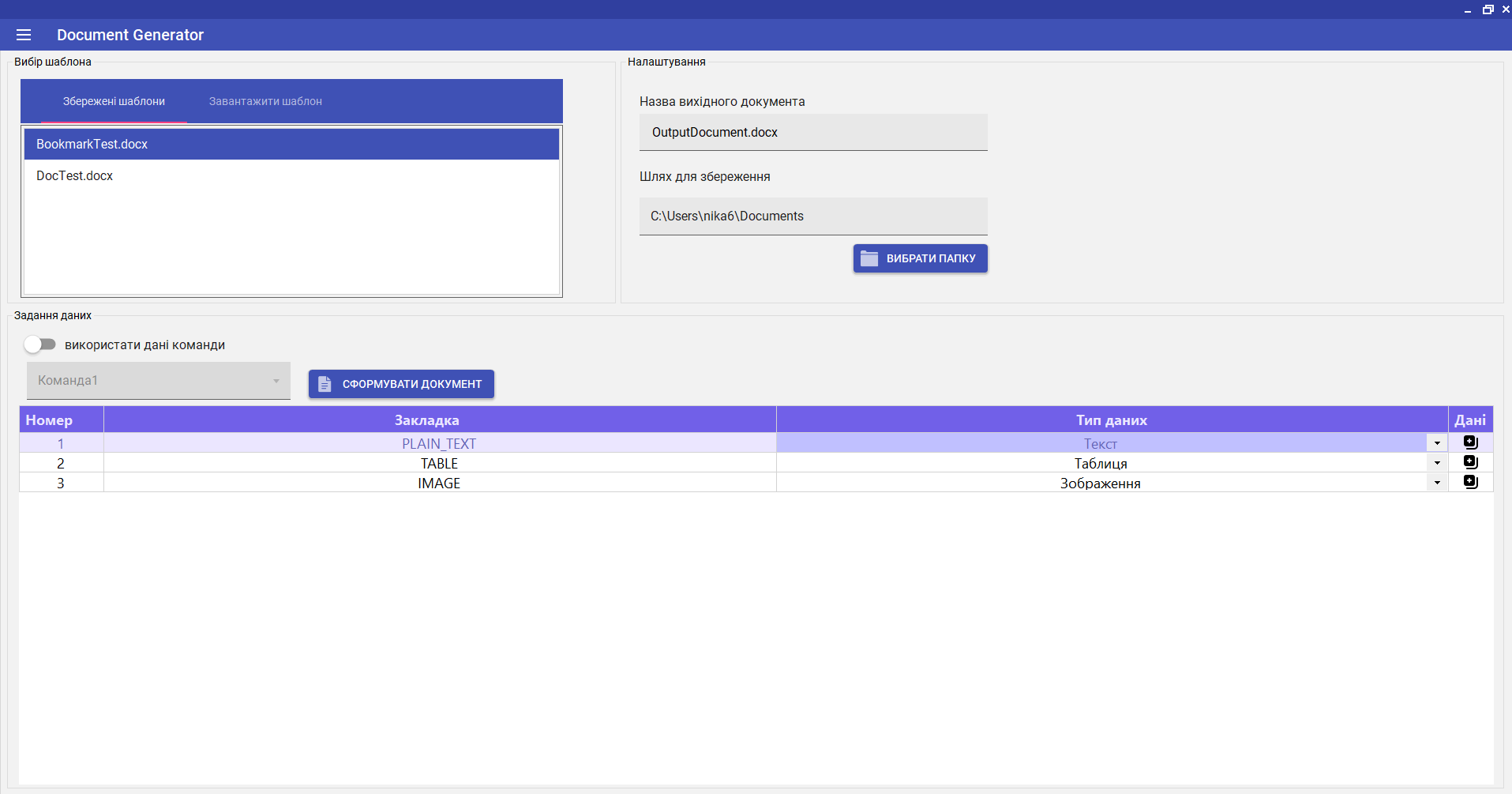


Рис.8. Приклад №1 для модуля генерацій для формування документа

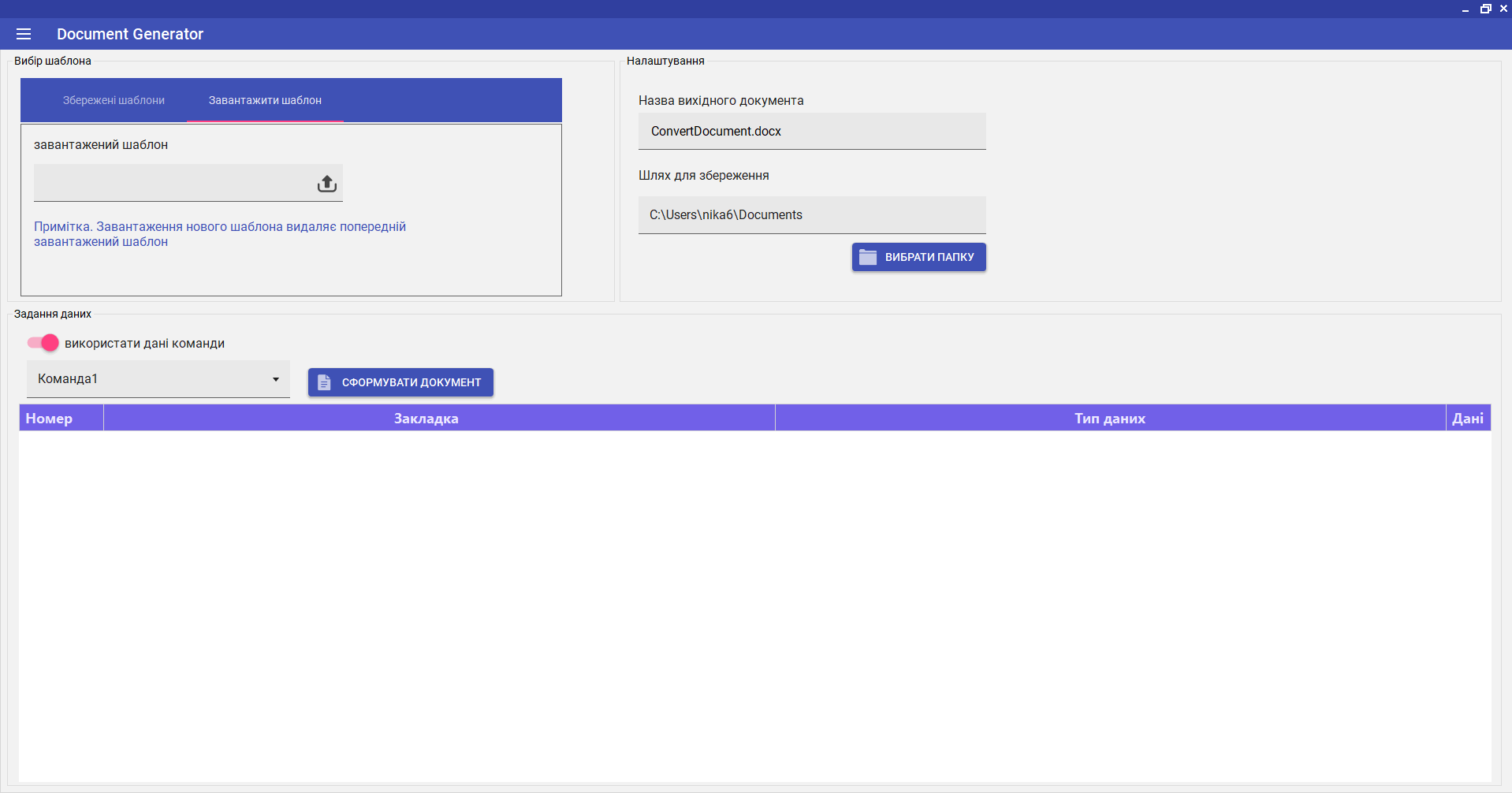


Рис.9. Приклад №2 для модуля генерацій для формування документа

44165850.01365-01 I3 0

13

1. АВАРІЙНІ СИТУАЦІЇ

Якщо під час роботи програми сталася аварійна ситуація, рекомендується перезавантажити програму для відновлення її працездатності.