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](#_Toc167270470)

[**1 ЗБІР ТА АНАЛІЗ ВИМОГ** 8](#_Toc167270471)

[1.1 Опис аналогів 8](#_Toc167270472)

[1.2 Постановка задачі 12](#_Toc167270473)

[**2 ПРОЄКТУВАННЯ** 13](#_Toc167270474)

[2.1 Зовнішнє проєктування 13](#_Toc167270475)

[2.1.1 Функціональне та експлуатаційне призначення 13](#_Toc167270476)

[2.1.2 Функціональні вимоги 13](#_Toc167270477)

[2.1.3 Вхідні дані 15](#_Toc167270478)

[2.1.4 Вихідні дані 17](#_Toc167270479)

[2.2 Внутрішнє проєктування 18](#_Toc167270480)

[2.2.1 Проєктування архітектури програми 18](#_Toc167270481)

[2.2.2 Проєктування інтерфейсу користувача 35](#_Toc167270482)

[2.2.3 Проєктування бази даних 44](#_Toc167270483)

[**2** **РОЗРОБКА ПРОГРАМИ** 45](#_Toc167270484)

[**3** **ТЕСТУВАННЯ ТА НАЛАГОДЖЕННЯ** 46](#_Toc167270485)

[**4** **АНАЛІЗ ТА ВИСНОВКИ** 47](#_Toc167270486)

[**БІБЛІОГРАФІЧНИЙ СПИСОК** 48](#_Toc167270487)

[**ДОДАТКИ** 49](#_Toc167270488)

# **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).

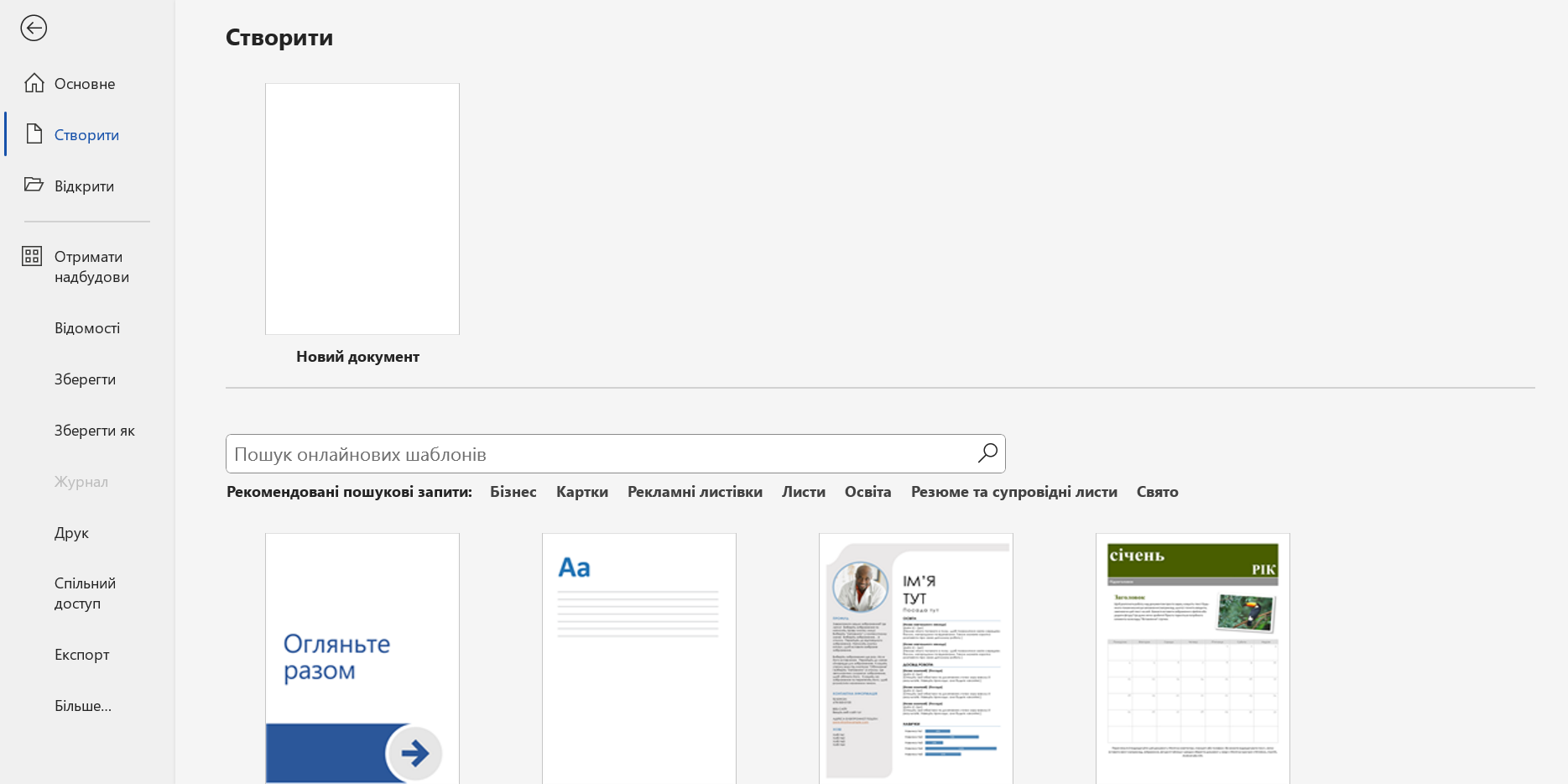


Рисунок 1.1 — Меню з шаблонами Word

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).

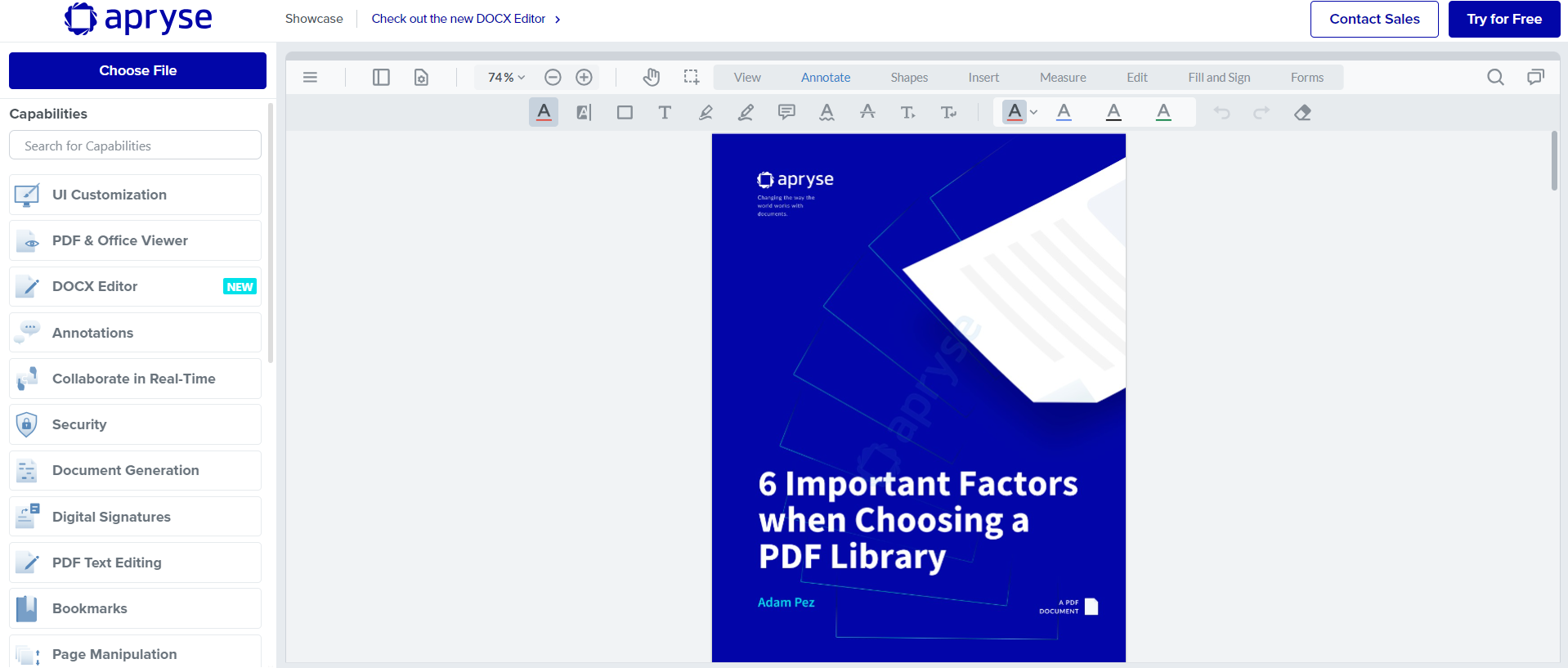


Рисунок 1.2 – Демонстраційна версія Apryse WebViewer

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).

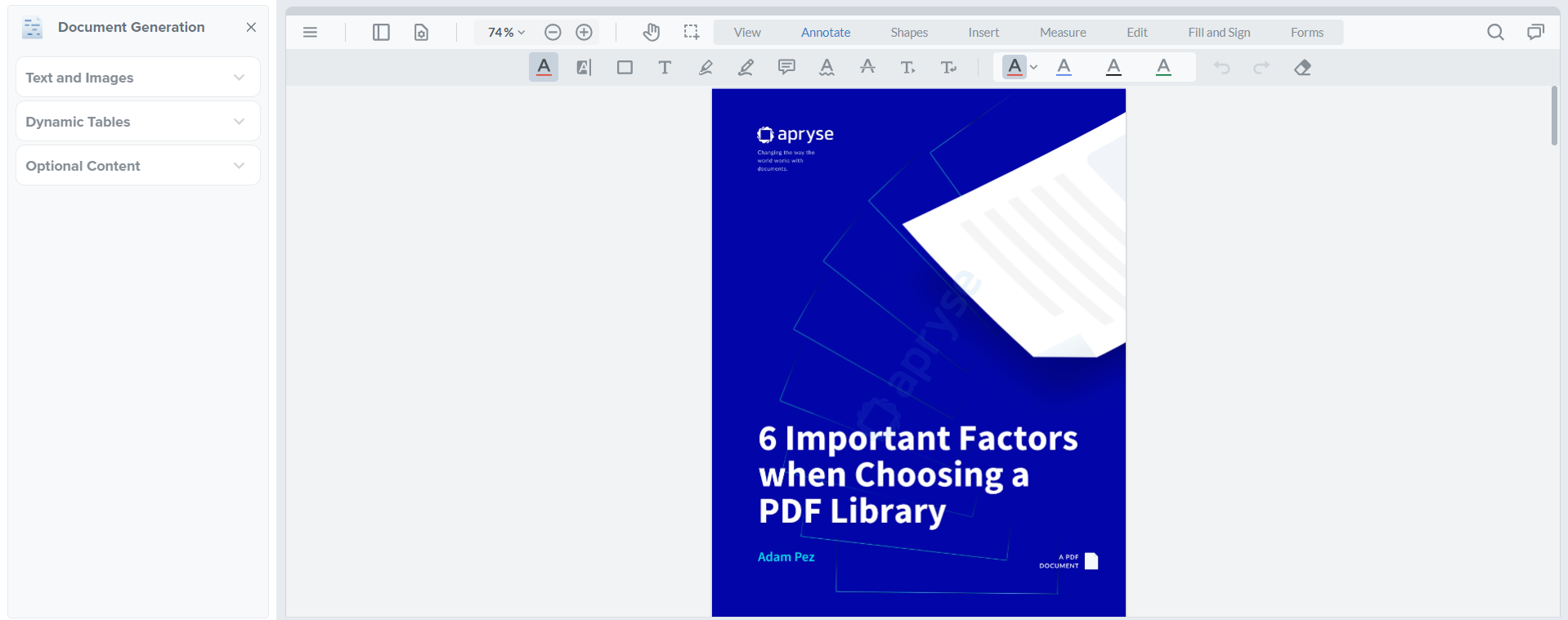


Рисунок 1.3 – Модуль формування документів за шаблоном Apryse WebViewer

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).

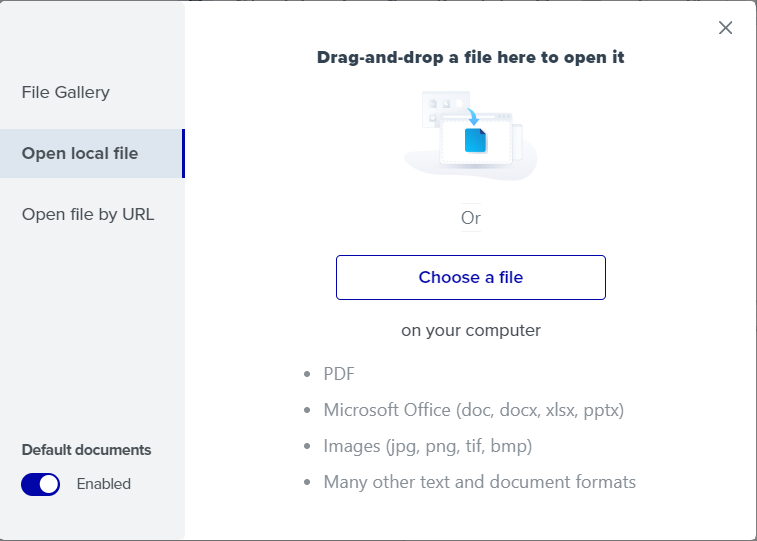


Рисунок 1.4 – Формати шаблонів для формування документа

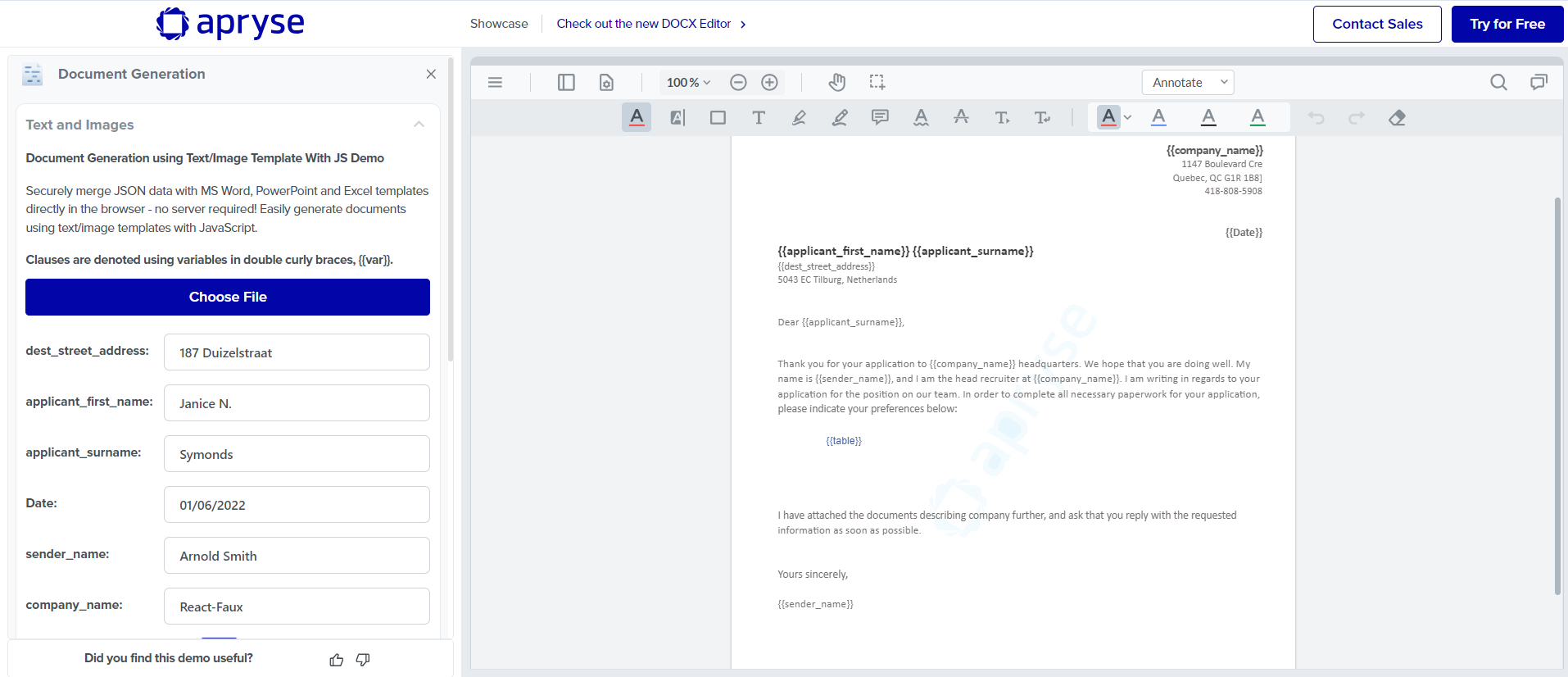


Рисунок 1.5 — Шаблон з категорією вмісту текст та зображення

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.

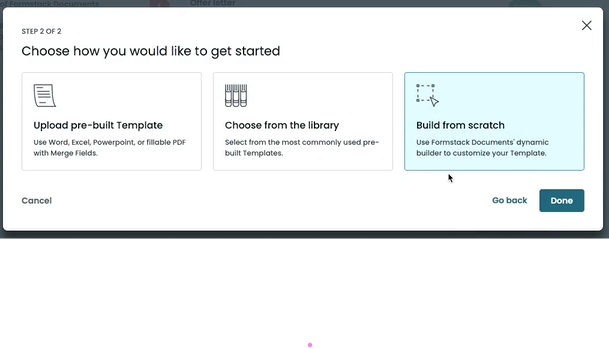


Рисунок 1.6 — Налаштування шаблону Formstack Documents [4]

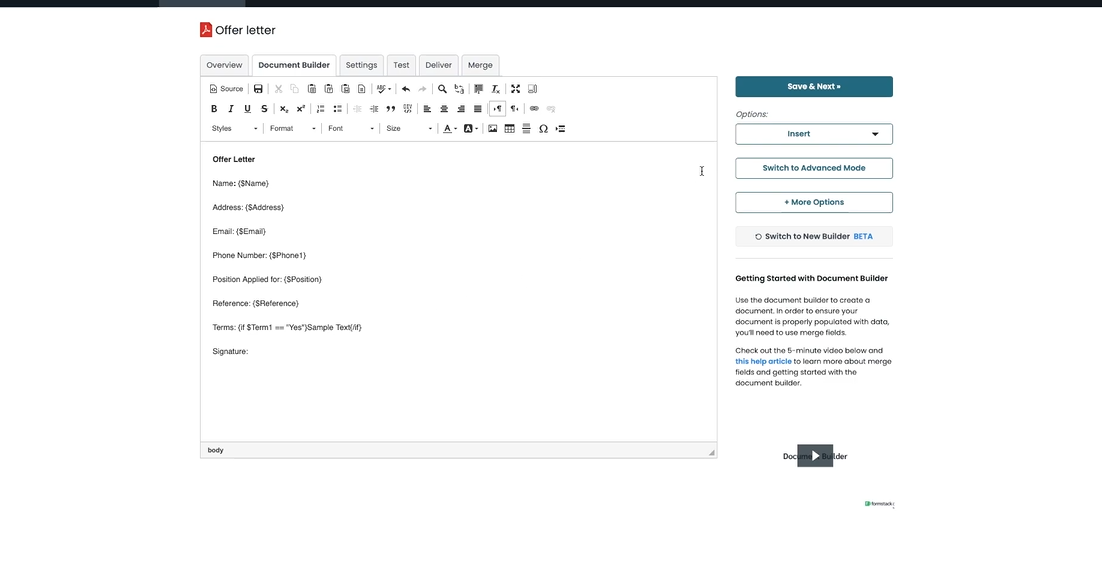


Рисунок 1.7 – Приклад створеного шаблону для Formstack Documents [4]

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.

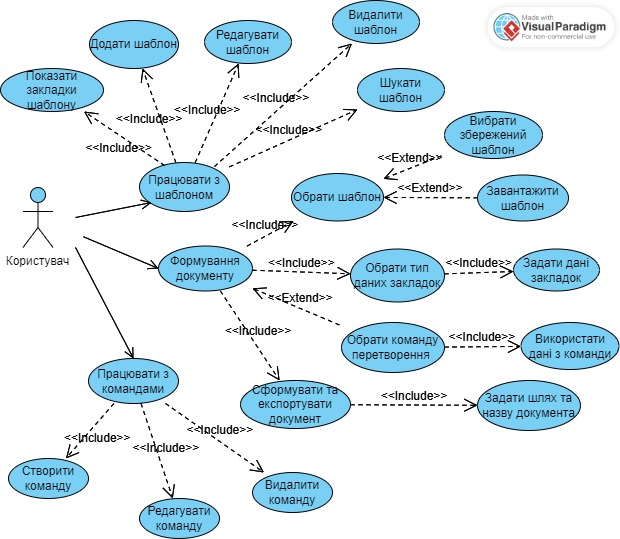


Рисунок 2.1 – Діаграма прецедентів

### **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;

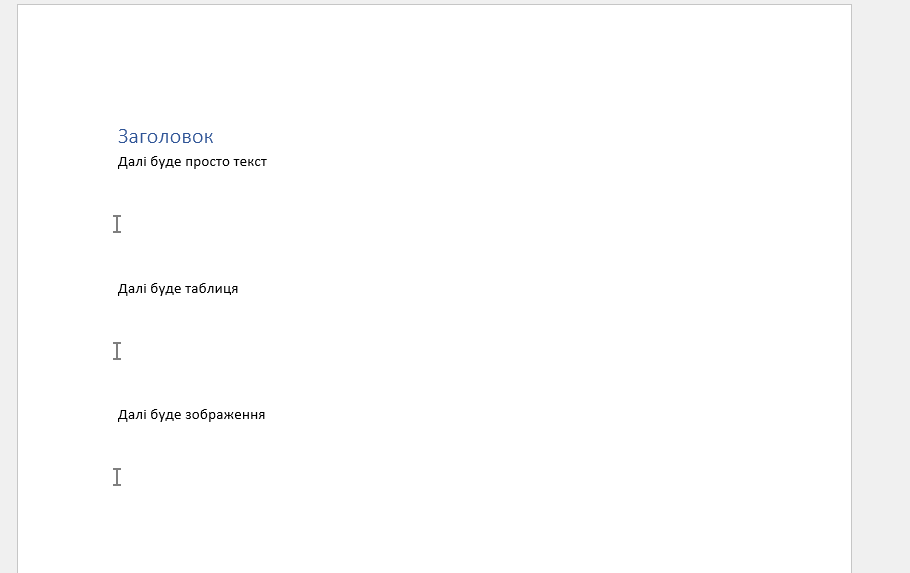


Рисунок 2.2 – Приклад вмісту шаблону з закладками

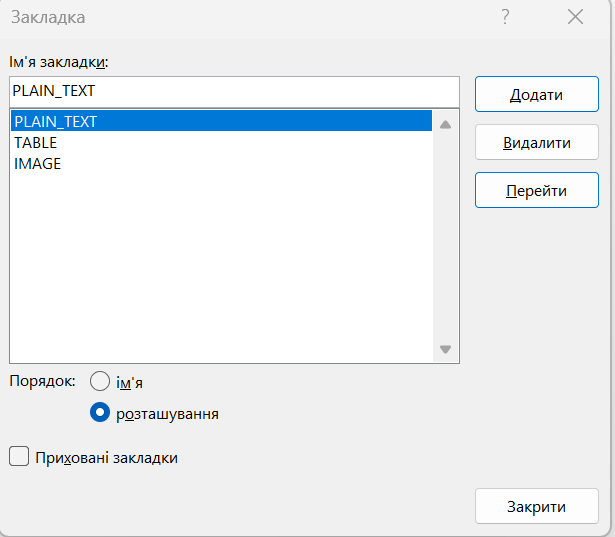


Рисунок 2.3 – Закладки в порядку розташування (див. рис.2.2)

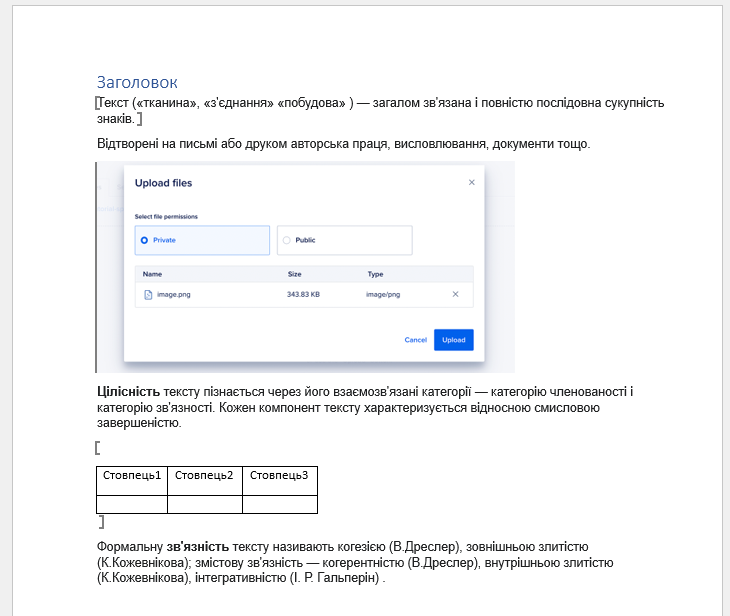


Рисунок 2.4 – Приклад заповненого документа

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.

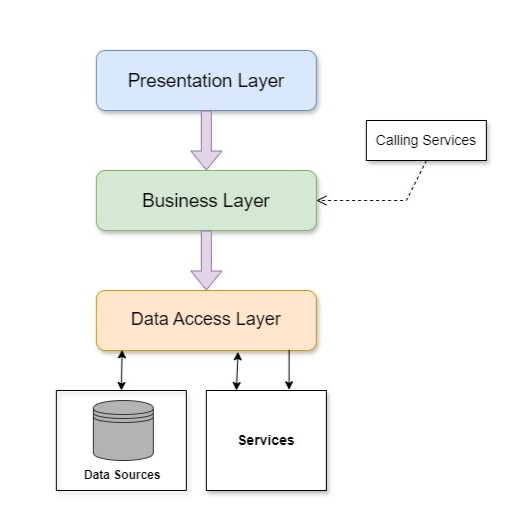


Рисунок 2.5 ­­­­– Схематичне зображення трирівненової архітектури

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.

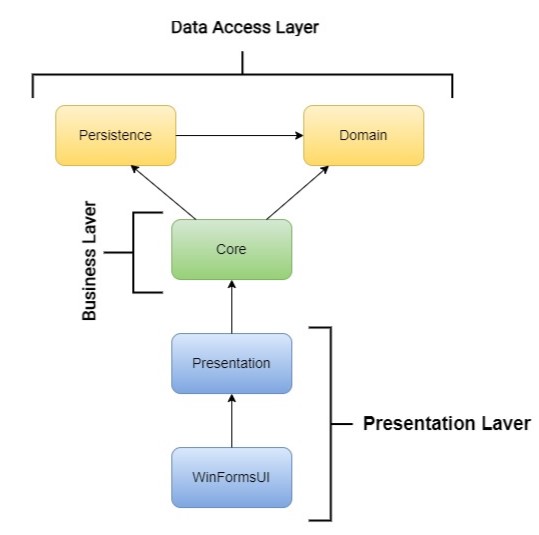


Рисунок 2.6 – Відповідність проєктів трирівневій архітектурі

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.

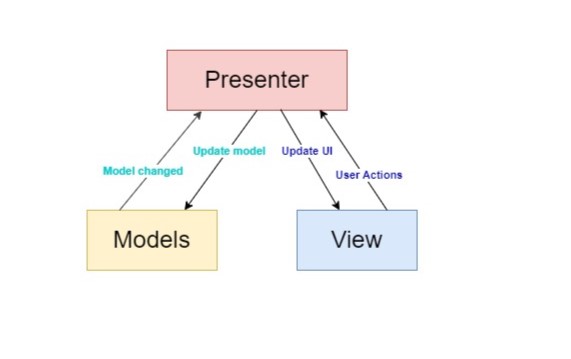


Рисунок 2.7 – Схема взаємодії компонентів MVP

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.

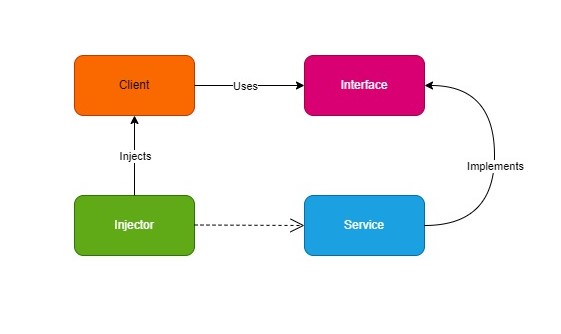


Рисунок 2.8 – Схематичне зображення патерну DI

**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).

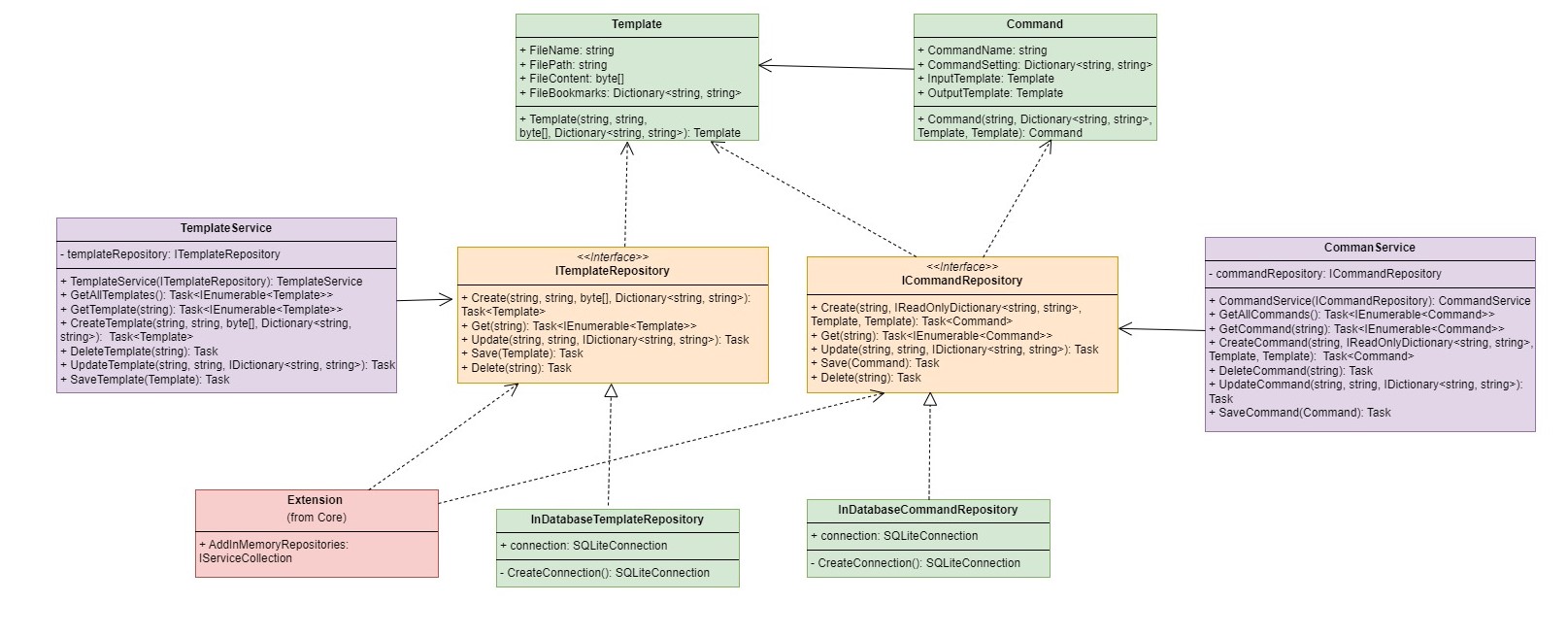


Рисунок 2.9 – Діаграма класів для проєктів Domain, Persistence та Core

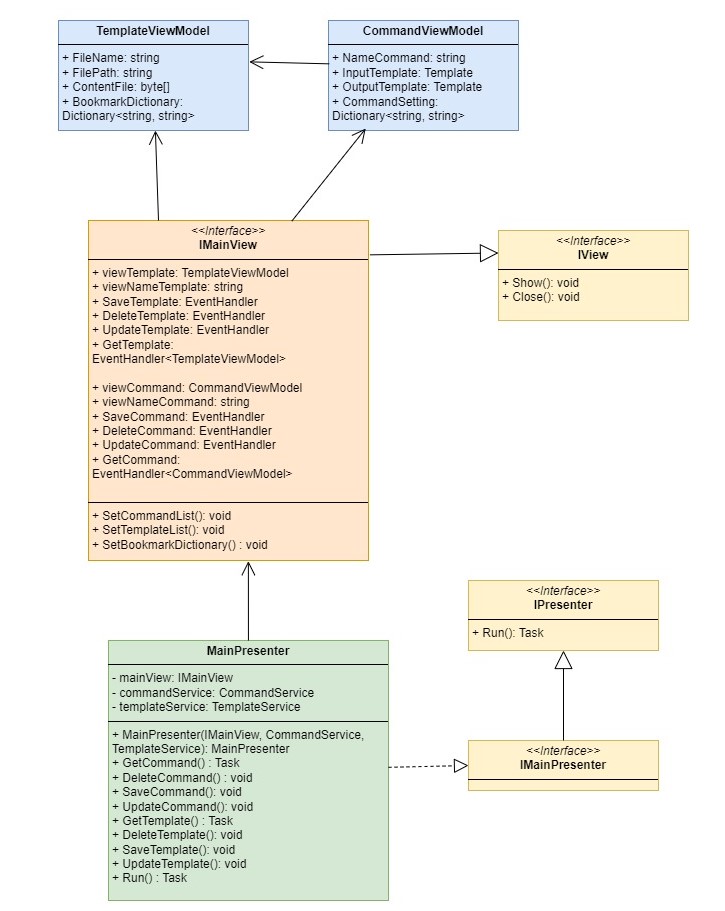


Рисунок 2.10 – Діаграма класів для проєкту Presentation

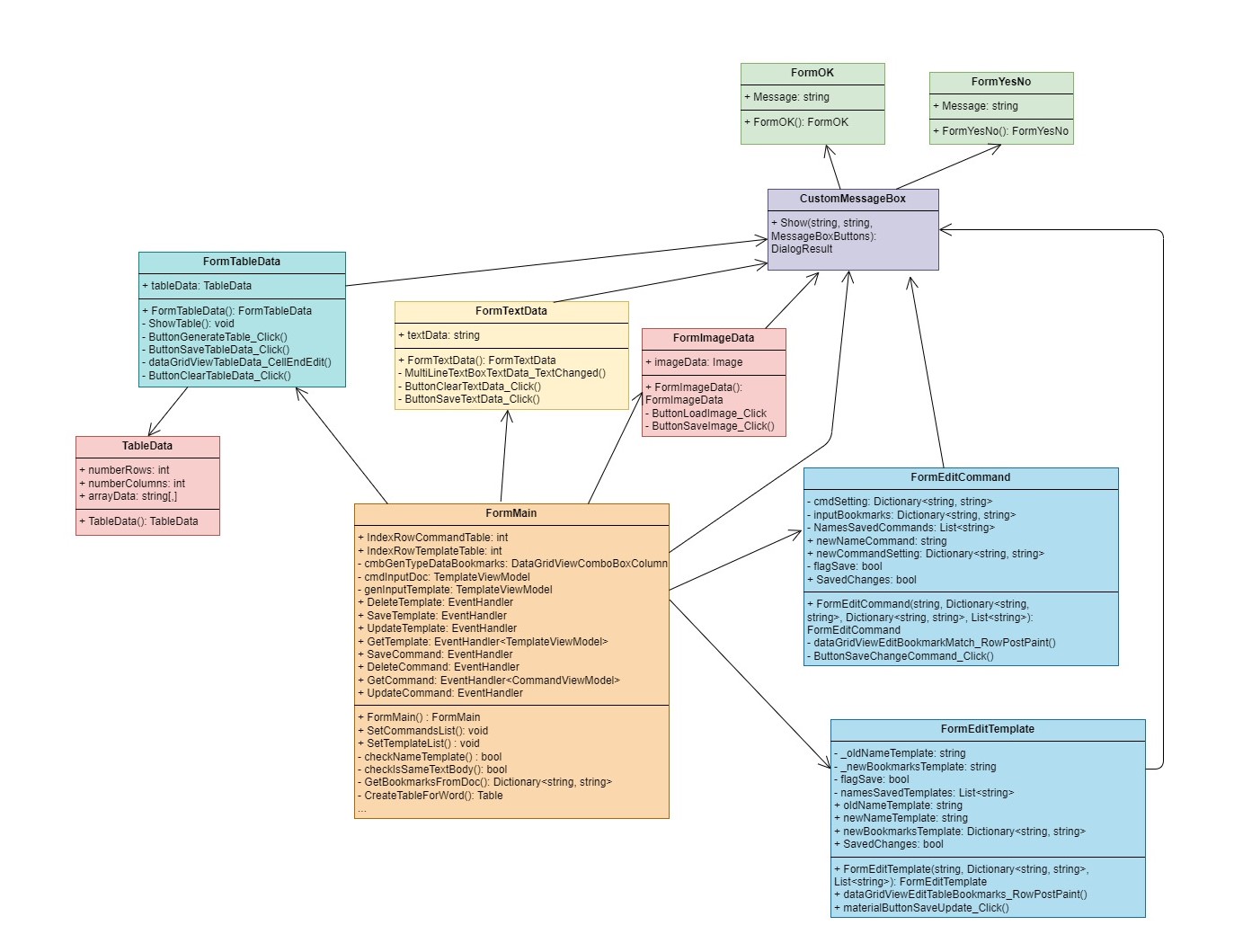


Рисунок 2.11 – Діаграма класів для проєкту WinFormUI

### **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.

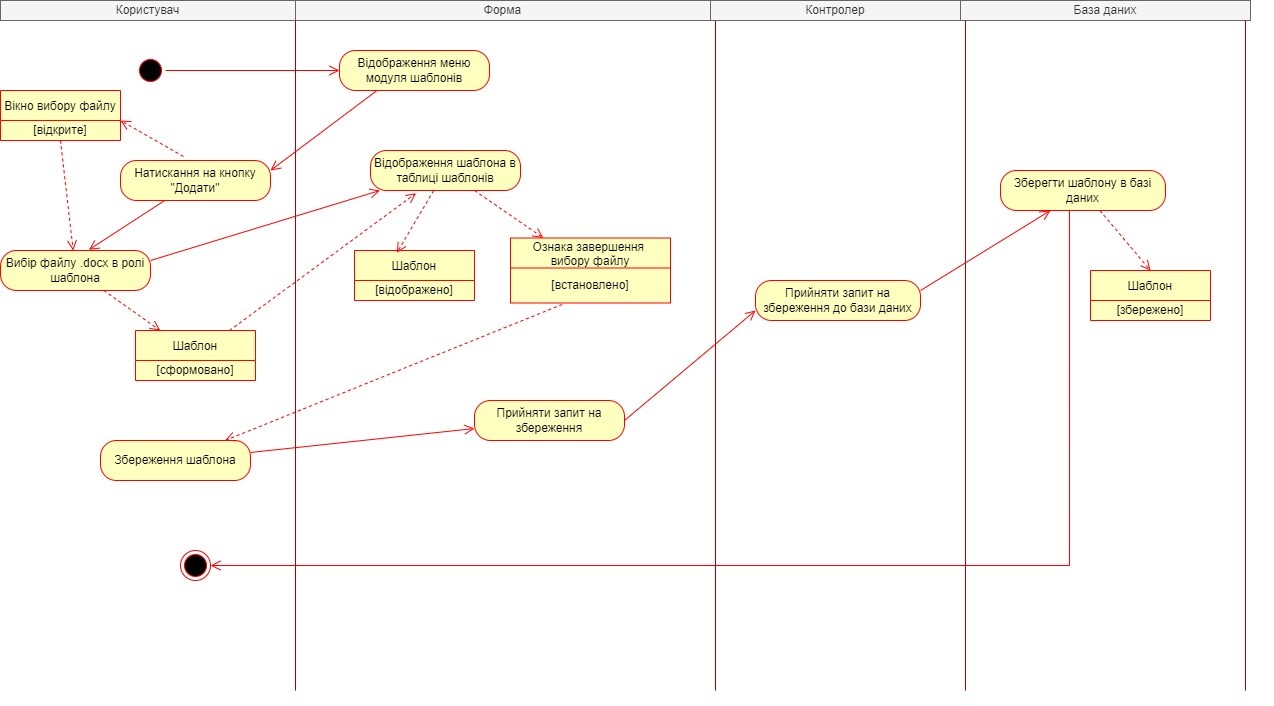


Рисунок 2.12 – Діаграма діяльності для прецеденту «Додавання шаблону» (модуль шаблонів)

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.

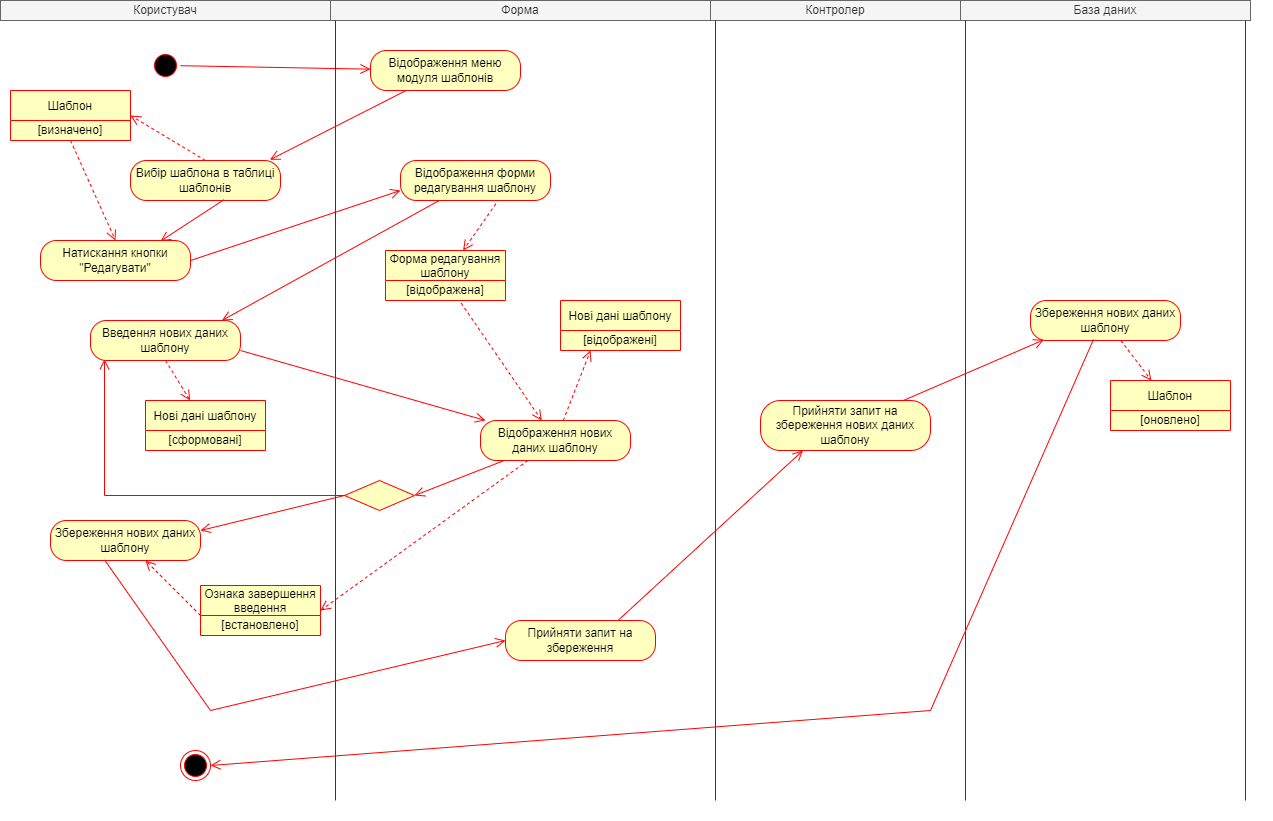


Рисунок 2.13 – Діаграма діяльності для прецеденту «Редагування шаблону» (модуль шаблонів)

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.

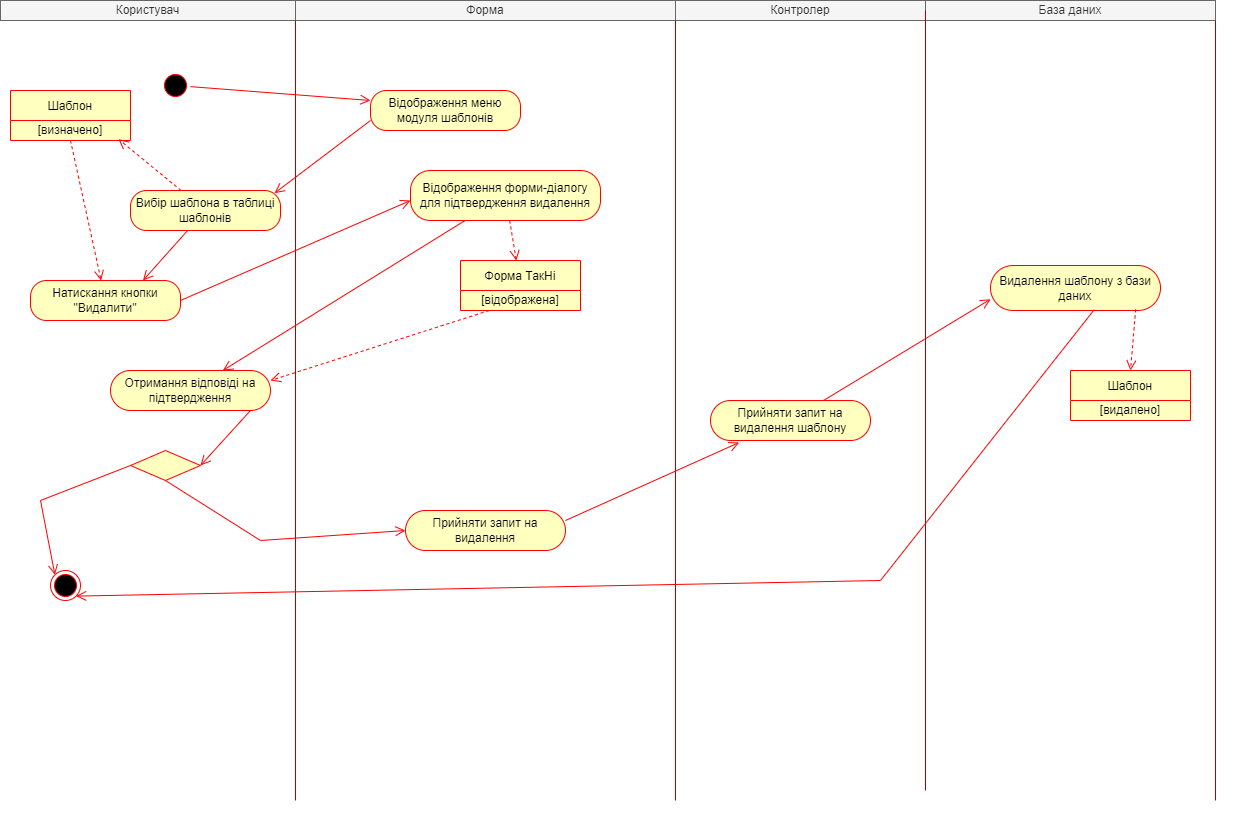


Рисунок 2.14 – Діаграма діяльності для прецеденту «Видалення шаблону» (модуль шаблонів)

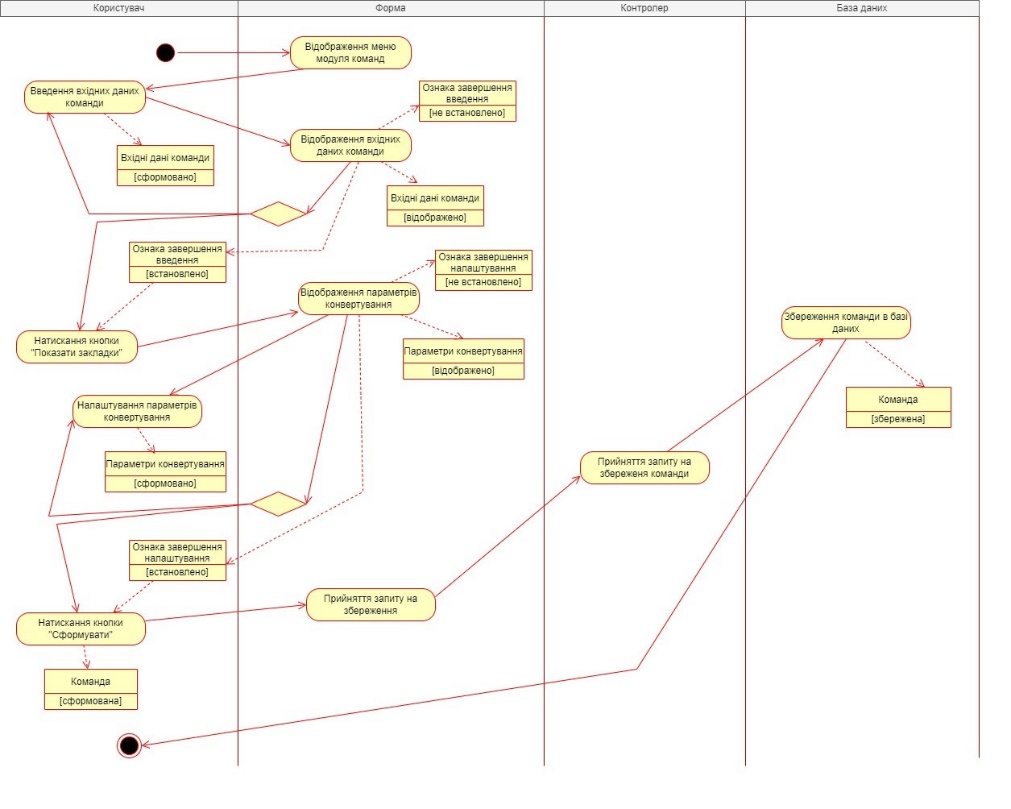


Рисунок 2.15 – Діаграма діяльності для прецеденту «Створення команди» (модуль команд)

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.

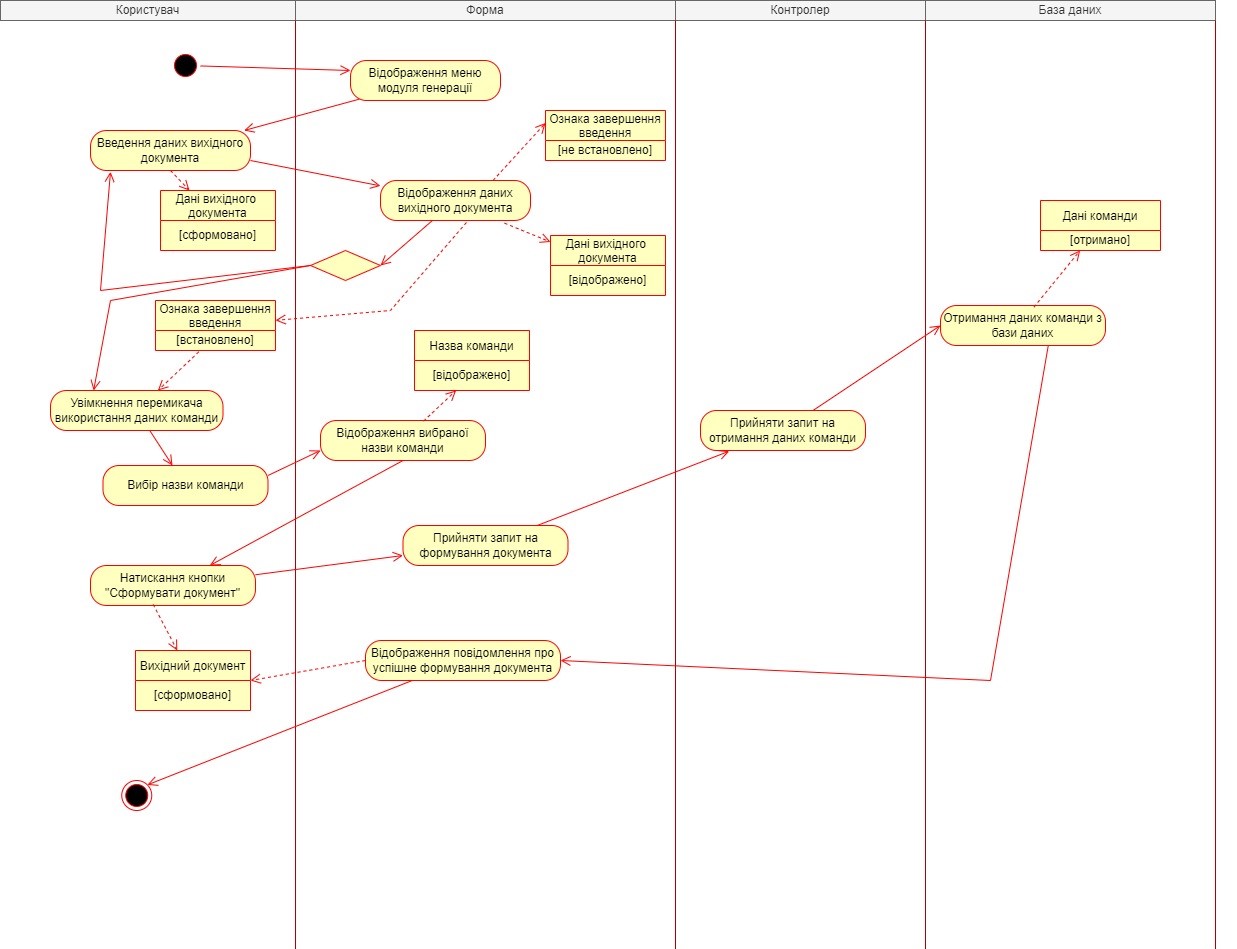


Рисунок 2.16 – Діаграма діяльності для прецеденту «Формування документа з використанням даних команди»

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.

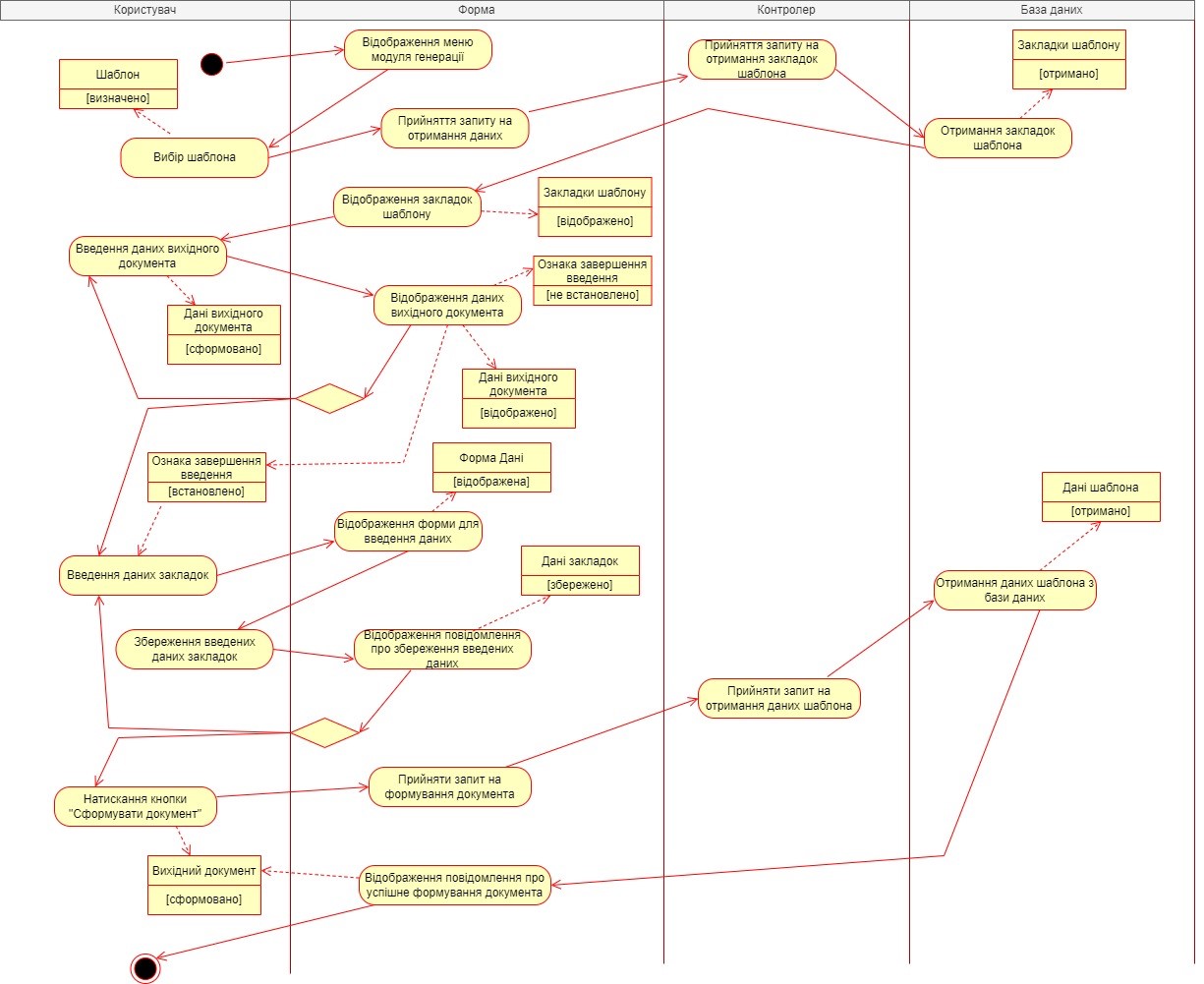


Рисунок 2.16 – Діаграма діяльності для прецеденту «Формування документа на основі шаблона»

**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/

# **ДОДАТКИ**