НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

Кафедра автоматизованих систем обробки інформації і управління

До захисту допущено:
В.о. завідувача кафедри
_____ *Олександр ПАВЛОВ*
(підпис) (вл.ім'я, прізвище)
"___"_____2021 р.

Дипломний проєкт
на здобуття ступеня бакалавра

**за освітньо-професійною програмою «Програмне забезпечення комп'ютеризованих систем»
спеціальності 121 «Інженерія програмного забезпечення»**

на тему: «*Інформаційна система підтримки дипломного проектування (комплексна тема). Система зберігання документів дипломного проектування. Індивідуальна частина №3*»

Виконав: студент *IV* курсу, групи *ІП-71*
Румянцев Олексій Васильович _____
(прізвище, ім'я, по батькові) (підпис)

Керівник *ас. Очеретяний Олександр Костянтинович* _____
(посада, науковий ступінь, вчене звання, прізвище, ім'я, по батькові) (підпис)

Консультант з графічної документації *доц., к.т.н., Ліщук Катерина Ігорівна* _____
(посада, науковий ступінь, вчене звання, прізвище, ім'я, по батькові) (підпис)

Рецензент _____ _____
(посада, науковий ступінь, вчене звання, прізвище, ім'я, по батькові) (підпис)

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент (-ка) _____
(підпис)

Київ – 2021 року

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет (інститут) *інформатики та обчислювальної техніки*
(повна назва)

Кафедра *автоматизованих систем обробки інформації і управління*
(повна назва)

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 121 «Інженерія програмного забезпечення»
Освітньо-професійна програма «Програмне забезпечення комп’ютеризованих систем»

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ *Олександр ПАВЛОВ*
(підпис) (вл.ім’я, прізвище)
“___”_____2021 р.

ЗАВДАННЯ
на дипломний проєкт студенту

Румянцеву Олексію Васильовичу
(прізвище, ім’я, по батькові)

1. Тема проєкту «*Інформаційна система підтримки дипломного проектування (комплексна тема). Система зберігання документів дипломного проектування. Індивідуальна частина №1*»
керівник проєкту *Очеретяний Олександр Костянтинович, ас.*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від “*11*”*травня* 2021 р. №1139-с
2. Термін подання студентом проєкту “*08*”*червня* 2021 року
3. Вихідні дані до проєкту
Технічне завдання

4. Зміст пояснювальної записки
1. Аналіз вимог до програмного забезпечення: основні визначення та терміни, опис предметного середовища, огляд існуючих рішень, постановка задачі
2. Моделювання та конструювання програмного забезпечення: моделювання та аналіз програмного забезпечення, засоби розробки, технічні рішення, архітектура програмного забезпечення
3. Аналіз якості та тестування програмного забезпечення
4. Розгортання та впровадження програмного забезпечення

5. Перелік графічного матеріалу

1. Схема структурна діяльності

2. Схема структурна класів програмного забезпечення

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання «*14*» *березня 2021 року*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	*Вивчення рекомендованої літератури*	*01.02.2021*	
2.	*Аналіз існуючих методів розв'язання задачі*	*10.02.2021*	
3.	*Постановка та формалізація задачі*	*14.02.2021*	
4.	*Розробка інформаційного забезпечення*	*17.02.2021*	
5.	*Алгоритмізація задачі*	*22.02.2021*	
6.	*Обґрунтування використовуваних технічних засобів*	*01.03.2021*	
7.	*Розробка програмного забезпечення*	*15.03.2021*	
8.	*Налагодження програми*	*15.04.2021*	
9.	*Виконання графічних документів*	*28.04.2021*	
10.	*Оформлення пояснювальної записки*	*01.05.2021*	
11.	*Подання ДП на попередній захист*	*17.05.2021*	
12.	*Подання ДП на основний захист*	*31.05.2021*	
13.	*Подання ДП рецензенту*	*08.06.2021*	

Студент Олексій РУМЯНЦЕВ

Керівник Олександр ОЧЕРЕТЯНИЙ

АНОТАЦІЯ

Структура та обсяг роботи. Пояснювальна записка дипломного проєкту містить 69 сторінок, 7 рисунків, 19 таблиць, 2 додатки, 13 джерел.

Дана дипломна робота присвячена розробці модулю для створення типових документів за наведеними шаблонами. Даний програмний продукт значно прискорить роботу із документами, адже створення будь-якого документу значно простіше, якщо є відповідний шаблон, особливо якщо даний процес шаблонізації буде автоматизований.

Головною метою розробки є автоматизація процесу створення шаблонів та генерація документів на їх основі.

У розділі аналізу вимог до програмного забезпечення був проведений детальний аналіз предметної області, існуючих програмних продуктів та підходів до вирішення поставленої задачі. Також були розглянуті основні сценарії використання розроблюваної системи та проаналізовані відповідні вимоги та рішення поставленої задачі.

У розділі моделювання та конструювання програмного забезпечення здійснено огляд архітектури розробленого програмного забезпечення та проведено аналіз безпеки даних та процесу обміну даними між клієнтом та сервером та автентифікація.

У розділі впровадження та супроводу програмного забезпечення був визначений процес розгортання модулів створеного програмного забезпечення.

У розділі аналіз якості та тестування програмного забезпечення був розроблений план тестування, також був описаний та проведений процес тестування.

КЛЮЧОВІ СЛОВА: АВТОМАТИЧНА ГЕНЕРАЦІЯ ДОКУМЕНТІВ, ШАБЛОНИ, ДОКУМЕНТООБІГ, КВАЛІФІКАЦІЙНА РОБОТА, АВТОМАТИЗАЦІЯ.

ABSTRACT

Structure and scope of work. The explanatory note of the diploma project consists of 69 pages, contains 7 figures, 19 tables, 2 appendicies, 13 sources.

This thesis is devoted to the development of module to create standard documents for these templates. This software product significantly improves the work with documents, cause creating any document is significantly easier if there is a suitable template, especially if this template can be filled automatically.

The main development service is the automation of the process of creating templates and generating documents based on them.

In the section of analysis of software requirements, a detailed analysis of the subject area, existing software products and approaches to solving the tasks was conducted. The main scenarios of using the developed system were also considered and the answers to the requirements and solutions of the problem were analyzed.

The software modeling and design section provides an overview of the software software architecture and security data analysis and data exchange process between clients and servers and authorization.

In the section of software implementation and maintenance, the process of deployment of the created software modules was defined.

In the section of software quality analysis and testing, a developed testing plan was developed, and the testing process was described and conducted.

KEY WORDS: AUTOMATIC GENERATION OF DOCUMENTS, TEMPLATES, DOCUMENT FLOW, QUALIFICATION WORK, AUTOMATION.

Пояснювальна записка
до дипломного проєкту

на тему: «*Інформаційна система підтримки дипломного проектування (комплексна тема). Система зберігання документів дипломного проектування. Індивідуальна частина №1*»

Київ – 2021 року

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

- Шаблон (Template) – шаблон типового документу із зазначеними індикаторами підстановки для динамічних даних.
- Макет документу – розмітка документу (розмір та шрифт тексту, встановлення відступів, інтервалів між рядками та абзацами, вирівнювання тексту тощо).
- Nest.js – прогресивний Node.js фреймворк для створення ефективних, надійних та масштабованих серверних додатків.
- DTO (Data Transfer Object) – об'єкт, який переносить дані між процесами.
- Docker – набір продуктів платформи як послуги (PaaS), які використовують віртуалізацію на рівні ОС для доставки програмного забезпечення в пакетах, званих контейнерами.
- OAuth2 – стандартний протокол для авторизації.

ВСТУП

У багатьох державних та приватних організаціях, установах, які пов’язані з сферами керування та звітності, великий обсяг функціональних обов’язків працівників складають різноманітні операції із документами – створення, редагування, перегляд та перевірка, аналіз та збір інформації тощо.

На даний момент відцифровування документів та трансформація традиційного документообігу в системи, які є більш автоматизованими та зручними в користуванні, значно спрощує роботу з документами, мінімізує витрачений час на пошук документів та обробку відповідних даних. Проте створення самих документів все ще є достатньо важким для регулювання процесом, так як передбачає багато помилок під час виконання даної задачі. Це створює необхідність додаткових перевірок, надлишково витрачених ресурсів та організаційних заходів. Більшість створених документів може бути приведена до одного «знаменника», так як ці документи створюються на базі вже існуючої та повторюваної інформації.

При коректному використанні сучасних технологій процес створення таких документів може бути автоматизований із метою мінімізації людського втручання. Для реалізації такої оптимізації необхідно формалізувати вхідні та вихідні документи та визначити правила і методи перетворення даних, що дозволить генерувати шаблонні документи без втручання людини та із мінімальною кількістю помилок.

Система освіти – одна із сфер, яка так чи інакше пов’язана із системами звітності та документообігу. Зокрема це стосується вищих навчальних закладів, які щорічно випускають тисячі студентів за певним освітньо-кваліфікаційним рівнем. Однією із складових процесу успішного закінчення вищого навчального є написання студентом дипломної роботи. Даний процес є затратним з точки зору ресурсів, та вимагає чіткого дотримання правил оформлення всіх супутніх документів на всіх стадіях роботи, тому тема роботи є актуальною.

Мета цієї дипломної роботи – автоматизація процесу створення шаблонів та генерація документів на їх основі.

Даний програмний продукт значно прискорить роботу із документами, адже створення будь-якого документу значно простіше, якщо є відповідний шаблон, особливо якщо даний процес шаблонізації буде автоматизований.

Задачі дипломної роботи:

- аналіз систем генерації шаблонів та визначення їх недоліків;
- генерація документів на основі шаблону та введених даних користувача;
- організація сховища отриманих вихідних документів та вхідних шаблонів;
- реалізація системи автоматичної звітності про роботу з шаблоном та взаємодію з документами створеними в рамках системи.

Цілі дипломної роботи:

- автоматизація системи документообігу;
- мінімізація помилок при створенні шаблонних документів;
- створення централізованої системи збереження дипломних робіт та супутніх файлів;
- спрощення перевірки документів учасниками процесу написання дипломних робіт;
- уніфікація системи документообігу та її компонентів.

1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Загальні положення

Документи виступають інструментом формального ведення справ у різних сферах життєдіяльності – законодавство, освіта, наука, взаємовідносини у фінансовій сфері, відносини робітника і роботодавця тощо. Окрім основної функції – зберігання та передавання інформації в часі і просторі – документ також виконує деякі вторинні функції: управлінські, правові, комунікативні тощо. Таким чином, документ є базовою конструкцією, без якої неможливо уявити функціонування багатьох систем у світі.

У документів є життєвий цикл, який визначається документообігом. Процес документообігу відрізняється у різних установах, проте можна визначити основні етапи, які є спільними: прийом документів, перевірка, обробка та реєстрація документів, відправка документів в архів. Централізоване збереження та впорядкування архівованих документів є одним із ключових процесів документообігу.

За структурою розрізняють стандартні та нестандартні документи. Стандартні документи мають чітко визначену структуру, заповнюються згідно з певним порядком та регламентуються певними правилами. Структура нестандартних документів варіюється від ситуації, тому її неможливо формалізувати.

У процесі написання дипломного проєкту учасники процесу працюють з великою кількістю саме структурних документів, які мають визначену форму, таких як: титульний лист до пояснювальної записки, технічного завдання та графічних матеріалів, лист завдання та календарний план, наліпка на диплом та інші.

До кожного з цих документів існує безліч вимог та правил щодо розмітки та форматування тексту. Регламент визначає:

- розмір та шрифт тексту;

- відступи абзаців;
- інтервали між рядками;
- вирівнювання тексту;
- вирівнювання, шрифт та розмір заголовків;
- правила оформлення списків, таблиць, діаграм, рисунків та інших комплексних структур у тексті тощо.

Отже регламент описує розміщення абсолютно всіх символів у документі, що дозволяє максимально формалізувати процес побудови того чи іншого документу, а також мінімізувати ймовірність неоднозначного трактування вимог.

Попри чітко визначені правила оформлення даних документів існує ряд проблем, з якими можуть стикнутись учасники процесу написання дипломних робіт.

Однією з таких проблем є людський фактор. Виходячи з особливостей людського мислення та сприйняття, а також зважаючи на велику кількість студентів, які вимушені власноруч збирати та заповнювати відповідні документи, можна стверджувати, що існує досить велика ймовірність того, що робота з оформлення та перередагування текстів буде більш ресурсозатратною, аніж робота з побудови змісту даних текстів.

Ще однією з можливих проблем є питання версійності шаблонів та прикладів заповнення документів. Так як перелік необхідних документів та правила їх оформлення можуть змінюватись з року в рік, студенти та їх дипломні керівники мають прикласти додаткових зусиль, щоб знайти релевантні версії для поточного часу.

Остання з найбільш розповсюджених проблем документообігу при процесі написання дипломних робіт є втрата частини документів. Так як відсутнє централізоване збереження і впорядкування архівованих документів (один із ключових процесів документообігу), то документи можна легко переплутати чи загубити зовсім, що призводить до додаткових дій учасників

процесу.

Деякі з перерахованих проблем частково можуть бути вирішеними за допомогою документів-зразків. Проте відсутність єдиного еталонного документа, або хоча б централізованого зберігання усіх подібних зразків призводить до фактичного розмиття рамок стандарту, адже як правило зразками виступають роботи студентів попередніх років, які в силу людського фактору особи, яка перевіряла дані роботи, також можуть містити в собі порушення правил оформлення документу.

Завдяки вищезазначеним фактам має місце наступна твердження - найбільша кількість помилок при перевірці кваліфікаційних робіт полягає у порушенні студентами вимог та правил щодо розмітки та форматування тексту документів.

Вирішенням цієї проблеми в контексті даного дипломного проєкту є створення системи автоматизованої генерації стандартних документів на основі шаблонів.

1.2 Змістовний опис і аналіз предметної області

Під шаблоном у термінології розроблюваного продукту розуміється текстовий документ, який містить спеціальні текстові конструкції – індикатори підстановки, тобто змінні.

Процес створення шаблону називається шаблонізацією. Даний процес зображений на діаграмі діяльності у графічних матеріалах і містить наступні етапи:

а) створення файлу у форматі DOCX;

б) написання загальної частини шаблону, тобто того, що спільне для всіх користувачів даного шаблону, наприклад: назви заголовків, назви полів для заповнення та інше;

в) для відображення інформації, яка є індивідуальною для кожного користувача шаблону, проводиться розташування індикаторів підстановки у

відповідних місцях, наприклад: назва факультету, код групи, ім’я студента та керівника, назва теми, тощо;

г) побудова макету документу, тобто налаштування розміру та шрифту тексту, встановлення відступів, інтервалів між рядками та абзацами, вирівнювання тексту. Важливо виконати цей крок не лише для загальної частини, але й для індикаторів підстановки.

В наш час документи-зразки складових дипломних робіт вже представлені у вигляді файлів формату DOCX. Зважаючи на даний факт, можна стверджувати, що формування документів-шаблонів з документів-зразків є досить легкою задачею. Адже зразок вже містить у собі загальну частину та макет. Вся складність побудови шаблону в даному випадку полягає в визначенні індивідуальної частини у документі-зразку та заміна даної частини на ряд індикаторів підстановки, а також відповідне форматування непостійних даних.

Така система шаблонізації дозволяє нівелювати помилки при створенні документів та зменшує кількість людей, які мають перевірити даний документ, що дозволяє генерувати готові документи на базі шаблонів та наданих даних автоматично, без участі людини.

Всі шаблони та готові документи зберігатимуться далі централізовано, що є частковим втіленням системи електронного документообігу. Створювати та модифікувати та видаляти шаблони можуть тільки користувачі, наділені відповідними правами. Всі документи організовані в ієрархічну структуру з наданням відповідних прав. Таке рішення підвищує рівень безпеки для збережених файлів.

Таке рішення мінімізує кількість потенційних проблем, які пов’язані із генерацією та збереженням документів та шаблонів, а також їх версіонуванням.

1.3 Аналіз успішних ІТ-проєктів

Існує достатньо багато систем електронного документообігу та генерації документів за шаблонами, проте зазвичай це різні системи, які вимагають додаткової інтеграції між собою. Також майже всі продукти, які надають необхідний функціонал, є платними або надають не достатню багатоплатформність, що не є зручним для організації написання дипломних робіт. Проте дані програмні продукти надають можливість проаналізувати та порівняти необхідний функціонал для розробки модулю збереження та генерації шаблонів типових документів.

У даній роботі розглянуті тільки найбільш розповсюджені програмні рішення, так як на даний момент існує безліч продуктів відповідного призначення як на українському, так і на зарубіжному ринках.

Одними з найбільш розповсюджених та цікавих з точки зору аналізу та порівняння програмних продуктів, які реалізують системи електронного документообігу, є:

- M.E.Doc;
- XpertDoc;
- Legito.

M.E.Doc – це програмне забезпечення для автоматизації процесів звітності та електронного документообігу в цілому [1]. Даний програмний продукт представлений як веб-додаток, що дозволяє користувачам на різних системах використовувати його. Також M.E.Doc має приємний та зрозумілий інтерфейс (рисунок 1.1).

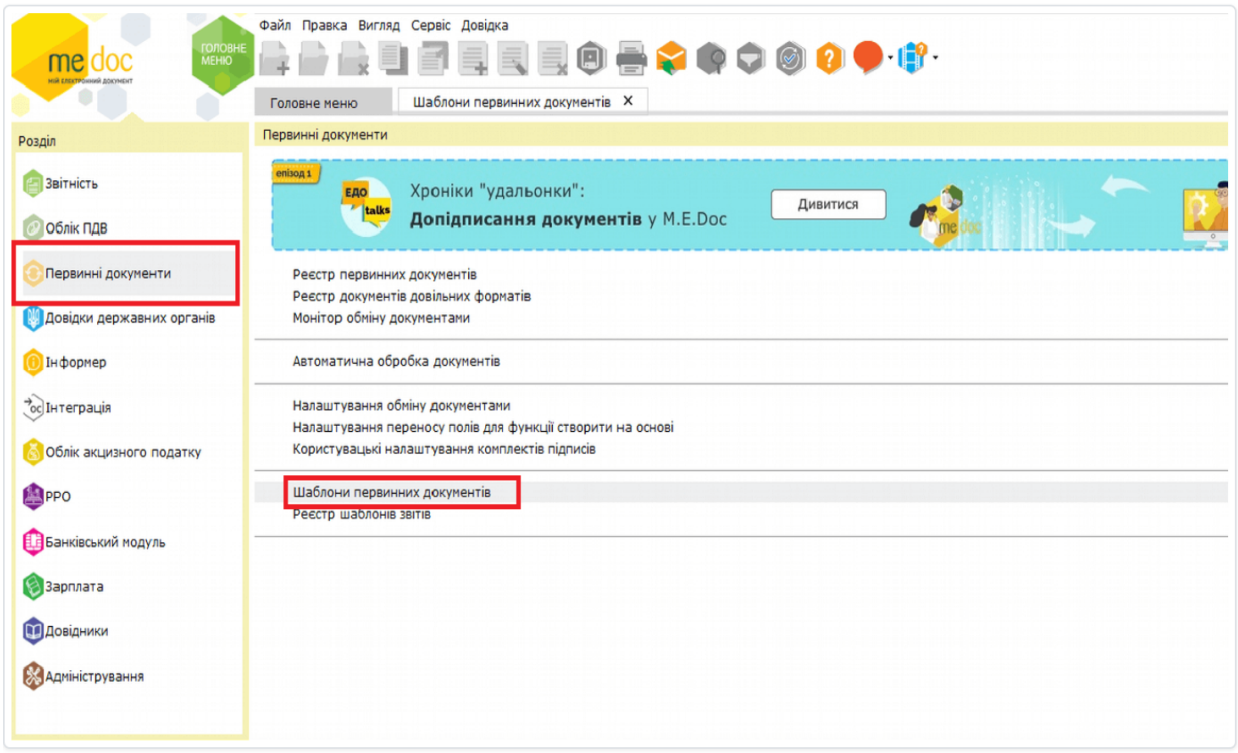

Рисунок 1.1 – Інтерфейс M.E.Doc

Незважаючи на ці переваги, даний програмний продукт більше націлений на можливість ведення документування в обмеженому переліку сфер (держава, бізнес), що звужує коло можливих створених шаблонів та документів. Хоч M.E.Doc також надає можливість конфігурувати власні шаблони, все ж більший фокус в системі наданий роботі із первинними документами, які зав'язані на певну сферу життя.

Наступний програмний продукт, який заслуговує уваги – XpertDoc [2]. Дана система надає наступний функціонал:

- створення документів за заданими шаблонами;
- автоматична генерація документів на основі наданих даних;
- централізоване зберігання документів;
- версіонування документів та шаблонів;
- надання можливості експортування документів у різні формати;
- надсилання документів багатьма способами (електронна пошта, месенджери, веб-портали);

- керування документами (підтвердження, відхилення змін);
- пошук по документах;
- електронні підписи тощо.

Попри вищезазначений необхідний функціонал, даний програмний продукт є значно складним з точки зору інтеграції, так як представлений програмний інтерфейсом (API), що значно підвищує поріг входу для звичайного користувача та потребує додаткових ресурсів на конфігурацію середовища.

Legito – система автоматизованого документообігу, яка надає широкий функціонал для користувачів: керування документами (підтвердження, відхилення) та шаблонами, версіонування, аналіз документів, централізоване збереження документів тощо [3]. Даний програмний продукт є багатоплатформний та надає користувачам приємний інтерфейс для роботи та створення документів (рисунок 1.2). Зважаючи на весь наданий функціонал та зручність користування та інтеграцій, Legito виступає одним із найдорожчих рішень сьогодні, тому не є вибором для широкого кола користувачів.

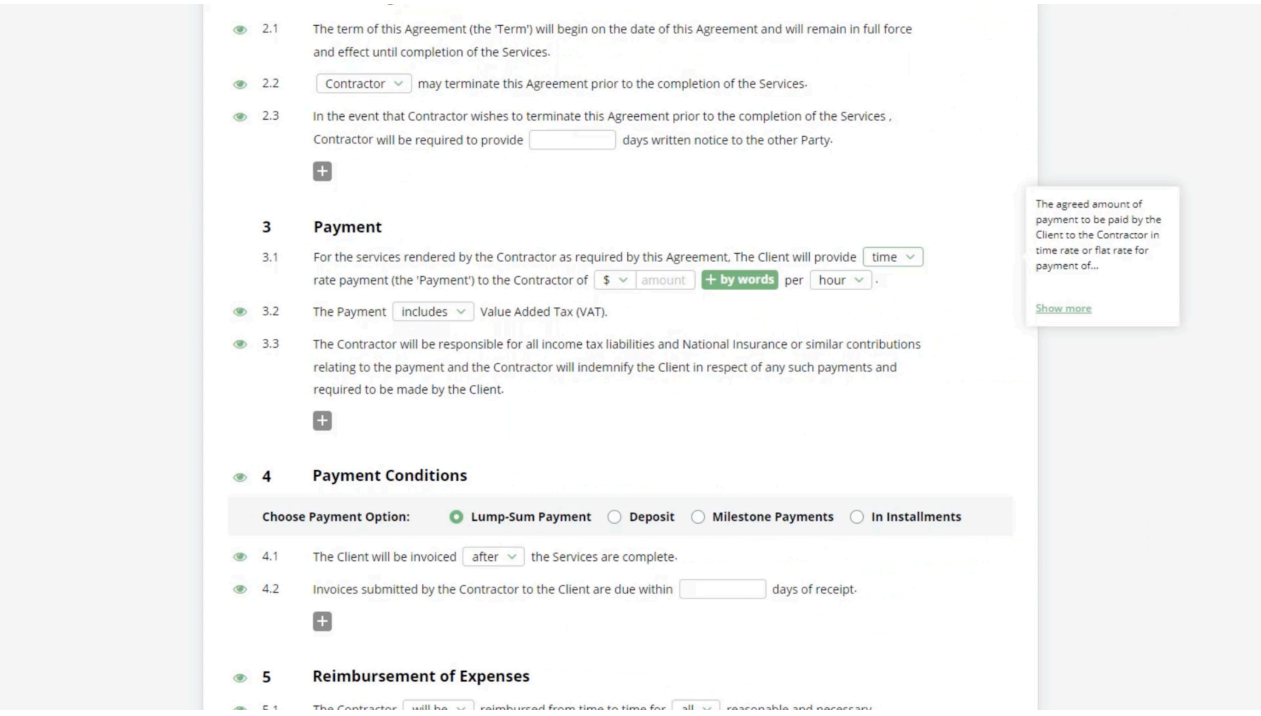

Рисунок 1.2 – Заповнення документа на базі шаблона у Legito

Виходячи з наведеного опису кількох програмних рішень, можна побачити, що існує багато систем, які надають функціонал для вирішення проблем, висвітлених в даній дипломній роботі. Але більшість існуючих систем мають ряд проблем, які ускладнюють їх інтеграцію в процес написання дипломних робіт та в поточну систему освіти в цілому – ціна, питання інтеграції та багатоплатформеність, недостатньо широкий функціонал.

1.4 Аналіз вимог до програмного забезпечення

Для визначення функціональної моделі розроблюваної системи зручно використовувати діаграму прецедентів (Use Case Diagrams), яка описує поведінку системи та допомагає сформулювати вимоги. Діаграма прецедентів для користувача представлена на рисунку 1.3.

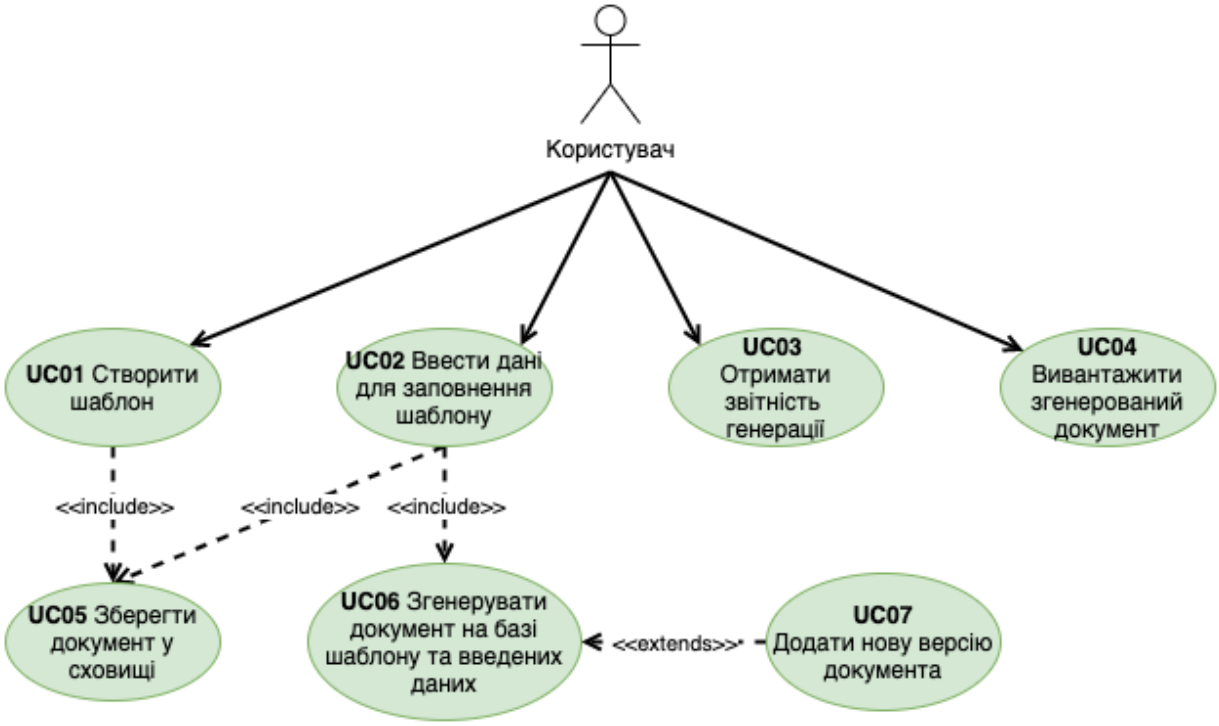

Рисунок 1.3 – Діаграма прецедентів модулю генерації документів

Всі визначені варіанти використання модулю генерації документів наведені у таблицях 1.1 – 1.7.

Таблиця 1.1 – Варіант використання UC01

Ідентифікатор	UC01
Назва	Створити шаблон
Опис	Користувач має можливість створити шаблон документу у системі
Учасники	Користувач
Передумови	-
Післяумови	Створений шаблон документу, за яким в подальшому може бути згенерований документ із використанням динамічних даних на місці шаблонних заготовок. Даний шаблон збережений у централізованому сховищі документів. Користувачу повертається ідентифікатор створеного шаблону
Основний сценарій	Користувач імпортує файл-шаблон із заготовками (placeholders) для місць, які повинні бути замінені динамічними даними в готовому згенерованому документі
Альтернативний сценарій	-
Сценарій-виключення	Формат шаблону не валідний. Користувачу повертається помилка із повідомленням, що формат шаблону не відповідає нормам

Таблиця 1.2 – Варіант використання UC02

Ідентифікатор	UC02
Назва	Ввести дані для заповнення шаблону
Опис	Користувач має можливість надіслати динамічні дані для заповнення шаблону
Учасники	Користувач
Передумови	Існує відповідний попередньо створений шаблон
Післяумови	Створена нова версія згенерованого за шаблоном документу. Користувачу повертається згенерований документ
Основний сценарій	Користувач надсилає динамічні дані у вигляді назв полей в шаблоні та відповідних значень, шляхом підстановки даних у відповідні поля шаблону генерується документ. Створюється перша версія даного документу поточного користувача. Згенерований документ зберігається у сховищі файлів, також зберігаються метадані про файл. Користувачу повертається згенерований файл

Продовження таблиці 1.2

Альтернативний сценарій	Користувач надсилає динамічні дані у вигляді назв полей в шаблоні та відповідних значень, шляхом підстановки даних у відповідні поля шаблону генерується документ. Існує попередня версія документу у поточного користувача, тому створюється запис про нову версію документа. Згенерована нова версія документу зберігається у сховищі файлів, також зберігаються метадані про нову версію файла. Користувачу повертається згенерований файл
Сценарій-виключення	Не існує відповідного шаблону. Користувачу повертається помилка із повідомленням, що відповідного шаблону не існує в системі

Таблиця 1.3 – Варіант використання UC03

Ідентифікатор	UC03
Назва	Отримати звітність генерації
Опис	Користувач має можливість отримати звітність по згенерованих документах
Учасники	Користувач
Передумови	-

Продовження таблиці 1.3

Післяумови	Отримана звітність по згенерованих користувачем документах (час генерації, версія, ідентифікатор файлу та шаблону)
Основний сценарій	Із бази даних системи формується статистика по користувачу – згенеровані документи та їх час генерації, версія, ідентифікатор файлу та шаблону. Користувачу повертається зібрана статистика
Альтернативний сценарій	Користувач не має жодного згенерованого файлу. Повертається пуста відповідь
Сценарій-виключення	-

Таблиця 1.4 – Варіант використання UC04

Ідентифікатор	UC04
Назва	Вивантажити згенерований документ
Опис	Користувач має можливість скачати згенерований документ
Учасники	Користувач
Передумови	Існує відповідний попередньо згенерований документ

Продовження таблиці 1.4

Післяумови	Згенерований документ відповідної версії вивантажено із централізованого сховища документів
Основний сценарій	Користувач надає ідентифікатор документу та версію, яку необхідно отримати. Користувачу надається файл за відповідним ідентифікатором та версією із централізованого сховища документів
Альтернативний сценарій	Користувач надає ідентифікатор документ, але не надає версію документу. Користувачу надається файл за відповідним ідентифікатором та останньою версією із централізованого сховища документів
Сценарій-виключення	Не існує відповідного документу. Користувачу повертається помилка із повідомленням, що відповідного документу не існує в системі

Таблиця 1.5 – Варіант використання UC05

Ідентифікатор	UC05
Назва	Зберегти документ у сховищі
Опис	Користувач має можливість зберегти документ у сховищі
Учасники	Користувач
Передумови	-

Продовження таблиці 1.5

Післяумови	Документ збережений у централізованому сховищі документів
Основний сценарій	Необхідно зберегти згенерований документ користувача. Якщо папки користувача не існує – вона створюється. Файл зберігається у папку користувача.
Альтернативний сценарій	Необхідно зберегти шаблон. Файл шаблону зберігається у папку шаблонів.
Сценарій-виключення	-

Таблиця 1.6 – Варіант використання UC06

Ідентифікатор	UC06
Назва	Згенерувати документ на базі шаблону та введених даних
Опис	Користувач має можливість згенерувати документ на базі шаблону та введених даних
Учасники	Користувач
Передумови	Існує відповідний попередньо створений шаблон
Післяумови	Згенерований документ за шаблоном

Продовження таблиці 1.6

Основний сценарій	Відповідний шаблон вивантажується із сховища документів. Шаблон зчитується, із шаблону відділяються індикатори підстановки. На їх місце підставляються динамічні дані. Створений документ записується у файл.
Альтернативний сценарій	-
Сценарій-виключення	Не існує відповідного шаблону. Користувачу повертається помилка із повідомленням, що відповідного шаблону не існує в системі

Таблиця 1.7 – Варіант використання UC07

Ідентифікатор	UC07
Назва	Додати нову версію документа
Опис	Користувач має можливість створити нову версію документа
Учасники	Користувач
Передумови	1) Існує відповідний попередньо створений шаблон 2) Існує попередня версія документу

Продовження таблиці 1.7

Післяумови	Збережені метадані про нову версію згенерованого за шаблоном документу
Основний сценарій	Існує попередня версія документу у поточного користувача. Створюється запис про нову версію документа. Зберігаються метадані про нову версію файла.
Альтернативний сценарій	Не існує попередньої версії документу. Створюється запис про першу версію документа. Зберігаються метадані про першу версію файла.
Сценарій-виключення	Не існує відповідного шаблону. Користувачу повертається помилка із повідомленням, що відповідного шаблону не існує в системі

1.4.1 Розроблення функціональних вимог

Відповідно до описаних вище варіантів використання розроблюваної системи можна сформулювати функціональні вимоги, результат наведений в таблиці 1.8.

Таблиця 1.8 – Опис функціональних вимог

Ідентифікатор	Назва	Опис
REQ01	Збереження документів у централізованому сховищі документів	Система надає можливість збереження документів у централізованому сховищі документів та надає користувачеві відповідний ідентифікатор збереженого документу
REQ02	Скачування документу із централізованого сховища документів	Система надає користувачу можливість вивантаження документу за відповідним ідентифікатором документу
REQ03	Створення шаблону	Система надає можливість створення шаблону. Для цього необхідно імпортувати файл із заготовками (placeholders) для динамічних даних. Система повинна зберегти шаблон у централізованому сховищі документів та надавати ідентифікатор збереженого шаблону користувачу

Продовження таблиці 1.8

REQ04	Генерація документу	Система надає користувачу можливість генерації документу за попередньо створеним шаблоном та наданих даних. Система повинна підставляти динамічні дані у заготовки в шаблонах, зберігати згенеровані документи та надавати користувачеві ідентифікатор збереженого документу
REQ05	Версіонування документів	Система зберігає інформацію про поточну версію документа та попередні існуючі версії. Всі версії документів наявні у централізованому сховищі документів та можуть бути надані користувачеві за потреби

Відповідність функціональних вимог до визначених варіантів використання системи представлене через матриц. трасування для модулю генерації документів за шаблонами представлена рисунку 1.4.

	UC01	UC02	UC03	UC04	UC05	UC06	UC07
REQ01	■	■			■		
REQ02				■		■	
REQ03	■						
REQ04		■				■	
REQ05		■	■				■

Рисунок 1.4 – Матриця трасування

1.4.2 Розроблення нефункціональних вимог

Розроблюваний модуль генерації документів за шаблонами автоматизованої системи централізованої підготовки дипломних робіт представлений серверною частиною. Серверна частина надає програмний інтерфейс (API) для можливості інтегрування в інші системи.

Сервер даного модуля повинен надавати REST API [4] для роботи із шаблонами та документами. Сервер працює із централізованим сховищем даних, який є інтегрованою та легко замінимою частиною, що дозволяє використовувати будь-який із можливих сервісів для сховища даних.

API представлено кінцевими точками, які використовують формат FormData для передачі файлів та JSON для передачі інших даних. Усі супутні дані про документи зберігаються у базі даних із урахуванням захисту чутливих даних як на рівні застосунку, так на рівні бази даних. Контроль модифікації документів здійснюється шляхом порівняння хешів файлів, що дозволяє уникнути компрометування файлів у процесі роботи системи. Змінні оточення не є доступними всім користувачам, так як зберігаються у спеціальних файлах оточення, до яких є доступ тільки при розгортанні. Розгортання системи відбувається із використанням контейнеризації за допомогою Docker [5], що дозволяє проводити розгортання на будь-якій платформі.

1.4.3 Постановка комплексу завдань модулю

Головне призначення даного модулю автоматизованої системи централізованої підготовки дипломних робіт - це спрощення підготовки та написання документації, яка стосується процесу написання дипломних робіт, шляхом автоматизації створення відповідних типових документів.

Мета цієї дипломної роботи – автоматизація створення шаблонних документів та їх генерація на базі отриманих даних, формалізація правил перетворення вхідної інформації у вихідні документи із використанням метаданих для побудови шаблонів, збереження відповідних файлів та спрощення ведення звітності по згенерованих даних.

Розроблюваний модуль має вирішувати наступні поставлені задачі:

- створення системи формалізації правил перетворення даних;
- реалізація перетворення отриманої інформації у вихідні документи із використанням метаданих;
- організація отриманих вихідних документів та вхідних шаблонів у єдину базу файлів;
- реалізація безпечного сховища файлів;
- реалізація системи автоматичної звітності.

1.5 Висновки до розділу

У даному розділі були сформульовані основні функціональні та нефункціональні вимоги до розроблюваного модулю. Для того, щоб вивести необхідні вимоги були розглянуті основні сценарії використання модулю, проведений детальний аналіз та були поставлені задачі.

Важливою складовою цього розділу був аналіз існуючих підходів до вирішення поставленої задачі та відповідних програмних продуктів на ринку. Використовуючи зібрані дані можна зробити висновок, що існує достатньо багато програмних продуктів для часткового вирішення поставлених задачі.

Проте поставлені вимоги та задачі не можуть бути повністю вирішені за допомогою існуючих програмних продуктів. Тому система, яка розв’язувала вище описані задачі, була би затребувана серед вже існуючих аналогів на ринку.

Отже, аналіз предметної області та існуючих рішень показав актуальність розробки модулю збереження та генерації типових документів автоматизованої системи централізованої підготовки дипломних робіт.

2 МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Моделювання та аналіз програмного забезпечення

Для подальшої коректної розробки програмного забезпечення коректної необхідно провести його аналіз та моделювання. На даному етапі можна визначити процеси, які будуть відповідати визначеним раніше вимогам. Для наочної ілюстрації бізнес-процесів було використано BPMN. BPMN - це система умовних позначень, яка відображає процеси у розроблюваній системі та дозволяє всім бізнес-користувачам розуміти основні бізнес-процеси, які передбачені системою.

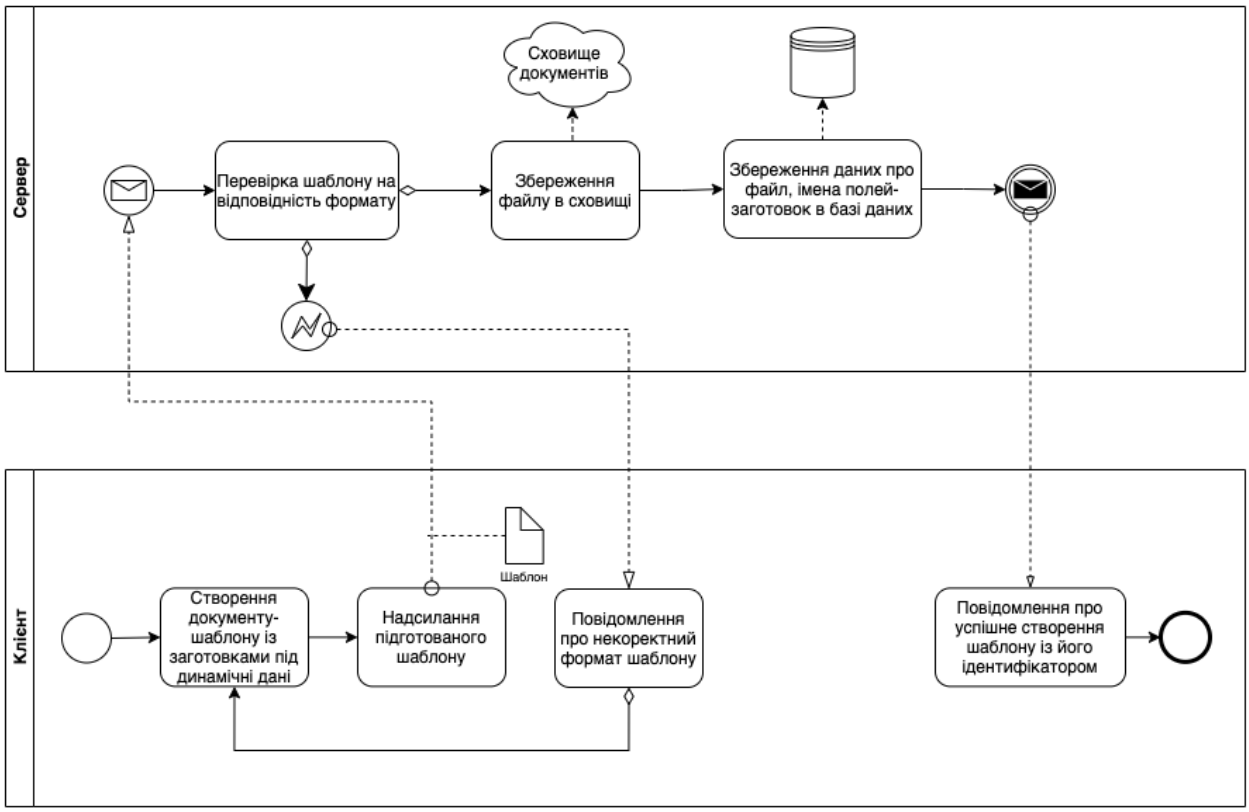

Рисунок 2.1 – Створення шаблону

Опис процесу створення шаблону:

- користувач створює документ-шаблон із статичними даними зазначеними полями-заготовками для їх подальшої підміни динамічними даними;
- користувач надсилає даний документ на сервер;
- шаблон перевіряється на відповідність формату;
- шаблон зберігається у централізованому сховищі даних;
- дані про шаблон (ідентифікатор, поля-заготовки) зберігаються у базі даних системи;
- у випадку некоректного формату шаблона користувачеві повертається змістовна помилка;
- у разі успішної перевірки та збереження шаблона користувачу повертається ідентифікатор шаблона у системі.

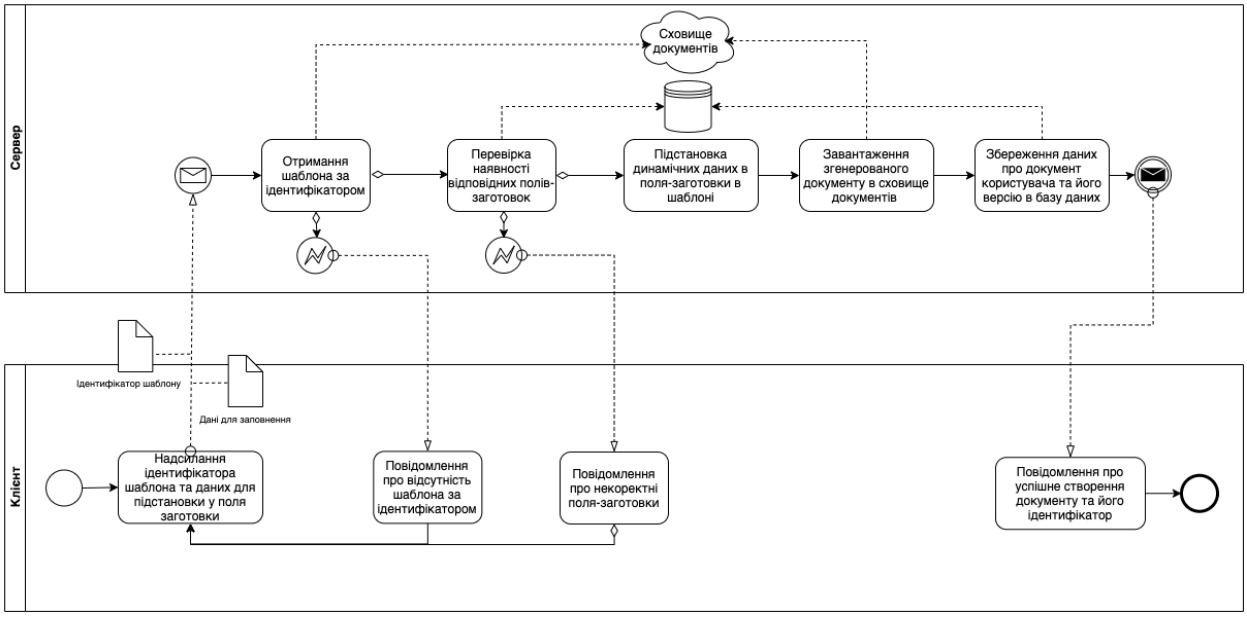

Рисунок 2.2 – Генерація документу за шаблоном

Опис процесу генерації документу за шаблоном:

- користувач надсилає на сервер ідентифікатор шаблону, на базі якого потрібно згенерувати документ, та динамічні дані для заповнення полів-заготовок;
- відбувається перевірка того, що відповідний шаблон існує в системі;
- у випадку відсутності відповідного шаблона у сховищі документів користувачеві повертається змістовна помилка;
- відбувається перевірка на відповідність та наявність полей-заготовок у шаблоні;
- у випадку не коректних полей-заготовок користувачеві повертається змістовна помилка;
- динамічні дані, отримані від користувача, підставляються у попередньо створений шаблон;
- згенерований документ зберігається у сховищі документів;
- дані про документ (ідентифікатор, користувач, версія) зберігаються у базі даних системи;
- у разі успішного завантаження документу у сховище документів та збереження даних про згенерований документ користувачу повертається ідентифікатор документа у системі.

2.2 Архітектура програмного забезпечення

2.2.1 Опис використаних рішень

Для розробки було обрано середовище WebStorm IDE, що полегшило розробку модулю. Так як для написання модулю була обрана мова TypeScript, середовище WebStorm IDE проводило аналіз коду під час розробки, що пришвидшило розробку та покращило автодоповнення.

Для реалізації даного модулю була вибрана програмна платформа Node.js [6], заснована на движку V8, яка дозволяє писати на мові JavaScript високопродуктивні мережеві застосунки. У даному випадку була вибрана мова TypeScript, яка компілюється у JavaScript код. Таке рішення дозволило перевірку типів під час компіляції. Для написання сервера розглядались декілька популярних фреймворків – Express, Koa.js, Fastify та Nest.js (абстракція поверх існуючих фреймворків Express/Fastify). В кінцевому результаті був вибраний Nest.js із використанням Express, так як даний фреймворк надає можливість розроблювати високоефективні масштабовані застосунки без накладних витрат, а також надає можливість приміняти найкращі архітектурні рішення при розробці застосунків.

Nest.js комбінує об’єктно-орієнтований, функціональний та функціонально-реактивний підходи програмування [7], що дає можливість писати код для різних потреб. Були використані такі патерни програмування - прототип, одинак, декоратор, репозиторій, ланцюжок обов’язків та впровадження залежності. Також були використані наступні принципи – інверсія управління та модульний підхід розробки. Весь реалізований функціонал був логічно розділений на модулі, що робить систему модульною.

У якості сховища даних була використана SQL база даних PostgreSQL. Для інтеграції вибраної бази даних була використана бібліотека Prisma, яка надає зручний функціонал для роботи із драйвером бази даних і також забезпечує міграції.

2.2.2 Опис сховища даних

У даному модулі є два типи сховища даних – сховище документів для збереження файлів користувачів та база даних для збереження супутніх даних про документи.

У якості сховища документів був вибраний Google Drive. Даний модуль є агностичним з точки зору сховища документів - в будь-який момент поточне

сховище документів може бути легко замінене на будь-яку існуючу альтернативу. Проте Google Drive надає зрозуміле API для роботи [8], та має найбільш оптимальні обмеження по об’єму безкоштовного сховища – 15 Гігабайт, що дає комфортне підґрунтя для першої стадії роботи програмного продукту.

Всі файли організовані в ієрархічну структуру. Для шаблонів створена окрема папка, файли користувачів також зберігаються у відповідних папках користувачів. Назви файлу генеруються унікальні для уникнення колізій.

Наступним сховищем даних виступає база даних. Під час проєктування модулю була розроблена її схема (рисунок 2.3).

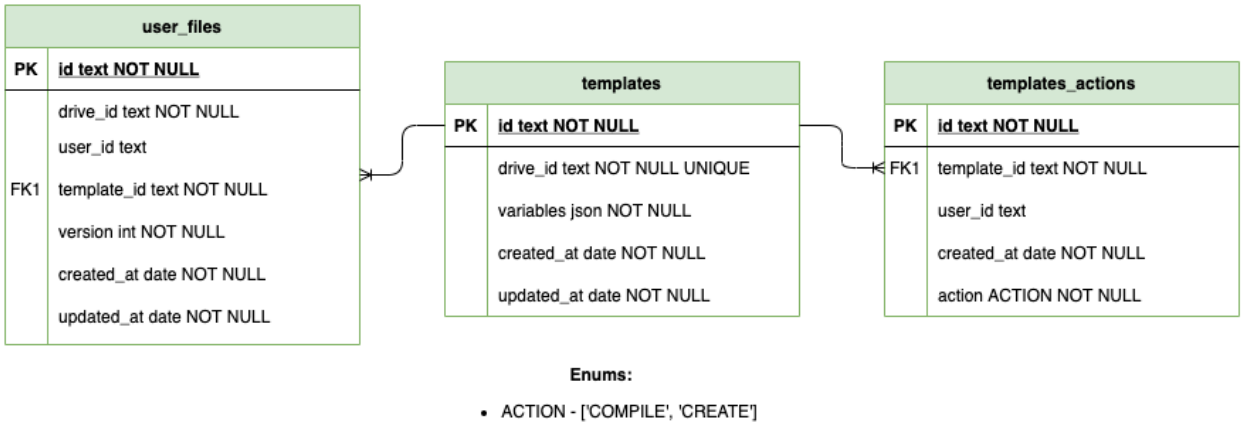

Рисунок 2.3 – Схема бази даних

При проєктуванні бази даних були створені перелічувані типи, які наведено у таблиці 2.1.

Таблиця 2.1 – Перелічувані типи даних

Назва типу	Можливі значення	Опис
ACTION	'CREATE', 'COMPILE'	Види дій над шаблонами

Шаблони у даному модулі представлені через таблицю templates. Дана таблиця містить інформацію про ідентифікатори шаблона у системі та у сховищі, також перелік індикаторів підстановки та дані про дату створення.

Таблиця 2.2 – Опис таблиці templates

Назва поля	Тип	Первинний ключ	Обов'язковий	Опис
id	Text	Так	Так	Унікальний ідентифікатор шаблона
drive_id	Text	Ні	Так	Унікальний ідентифікатор шаблона у сховищі даних
variables	Json	Ні	Так	Назви полів індикаторів підстановки
created_at	Date	Ні	Так	Дата створення шаблона
updated_at	Date	Ні	Так	Дата останнього оновлення шаблона

Для фіксування операцій над шаблоном була створена таблиця templates_actions, яка містить інформацію про дію над конкретним шаблоном та зв'язана із таблицею шаблонів через зовнішній ключ.

Таблиця 2.3 – Опис таблиці templates_actions

Назва поля	Тип	Первинний ключ	Обов'язковий	Опис
id	Text	Так	Так	Унікальний ідентифікатор операції
template_id	Text	Ні	Так	Ідентифікатор шаблону, зовнішній ключ
user_id	Text	Ні	Ні	Ідентифікатор користувача

Продовження таблиці 2.3

created_at	Date	Ні	Так	Дата виконання операції
action	ACTION	Ні	Так	Тип операції

Метадані про згенеровані файли зберігаються у таблиці user_files.

Таблиця 2.4 – Опис таблиці user_files

Назва поля	Тип	Первинний ключ	Обов'язковий	Опис
id	Text	Так	Так	Унікальний ідентифікатор файлу
template_id	Text	Ні	Так	Ідентифікатор шаблону, зовнішній ключ
user_id	Text	Ні	Ні	Ідентифікатор користувача
drive_id	Text	Ні	Так	Ідентифікатор файлу у сховищі даних
version	Integer	Ні	Так	Версія файлу
created_at	Date	Ні	Так	Дата створення файлу
updated_at	Date	Ні	Так	Дата останнього оновлення файлу

2.2.3 Опис модулів коду

Основні концепти в архітектурі застосунків, написаних на Nest.js -– модулі та провайдери. Модулі частіше за все розділяються з урахуванням реальних сутностей в бізнес-логіці застосунку. Модулі інкапсулюють в собі

провайдерів – сервіси, репозиторії, фабрики, утиліти тощо, які тісно пов’язані між собою набором можливостей та сферою дії. Структурна схема класів розроблюваного програмного забезпечення представлена в графічних матеріалах. Система збереження та генерації типових документів складається з модулів, представлених в таблиці 2.5:

Таблиця 2.5 – Опис модулів системи

Назва модулю	Опис модулю
AppModule	Кореневий модуль системи, є відправною точкою, яку Nest.js використовує для побудови графу застосунку - внутрішньої структури даних, яку Nest.js використовує для забезпечення взаємозв’язків та залежностей модулів та провайдерів
TemplatesModule	Функціональний модуль, який інкапсулює логіку роботи із шаблонами – завантаження шаблона, генерація документу за шаблоном тощо. Функціональні модулі дозволяють організовувати код, що відповідає певній функції, підтримуючи код упорядкованим та встановлюючи чіткі межі
GoogleApisModule	Функціональний модуль, який інкапсулює логіку роботи із Google API – авторизація, створення папок, файлів тощо
FileTransformModule	Функціональний модуль, який відповідає за перетворення буферів у потік даних та навпаки

Продовження таблиці 2.5

DbModule	Динамічний модуль для роботи з базою даних – встановлення та знищення з’єднання
ConfigModule	Динамічний модуль для конфігурування роботи інших модулів системи

Більш детальний опис компонентів (провайдерів) модулів застосунку наведений у таблиці 2.6.

Таблиця 2.6 – Опис компонентів модулів системи

Назва модулю	Назва провайдера	Тип провайдера	Призначення провайдера
AppModule	AppController	Контролер	Містить основну точку входу застосунку – перевірка статусу сервера
	AppService	Сервіс	Сервіс, який віддає контролеру статус сервера
TemplatesModule	TemplatesController	Контролер	Описує точки входу, які відповідають за шаблони – завантаження шаблону в систему, генерація документу за шаблоном, отримання шаблону

Продовження таблиці 2.6

	TemplatesService	Сервіс	Сервіс, який містить методи для завантаження шаблону в систему, генерація документу за шаблоном, отримання шаблону, генерації DOCX файлу та розбір DOCX на індикатори підстановки
GoogleApisModule	GoogleApisBase	Сервіс	Сервіс, який містить методи для авторизації в сховище документів використовуючи OAuth2
	DriveService	Сервіс	Сервіс, який авторизується під час своєї ініціалізації до сховища документів, а також містить методи для створення папки/файлу, отримання файлу/файлів та видалення файлу із сховища документів
FileTransformModule	FileTransformService	Сервіс	Сервіс, який містить методи для перетворення буферу даних в потік даних і навпаки
DbModule	DbService	Сервіс	Сервіс, який під’єднується до бази даних під час своєї ініціалізації та від’єднується під час деініціалізації

Продовження таблиці 2.6

ConfigModule	ConfigService	Сервіс	Сервіс, який віддає іншим модулям в системі об’єкти конфігурацій (з’єднання з базою даних, сховищем документів, змінні оточення тощо)

В таблиці 2.7 наведений детальна специфікація методів провайдерів.

Таблиця 2.7 – Опис методів провайдерів

Назва провайдера	Назва метода	Аргументи	Призначення
AppController	getHealthCheck	-	Описує точку входу для отримання поточного статусу сервера
AppService	getHealthCheck	-	Віддає контролеру статус сервера
TemplatesController	uploadTemplate	File: Express.Multer.File	Описує точку входу для завантаження шаблону в систему
	compileTemplate	compileData: CompileTemplateDto, res: Response	Описує точку входу для генерації документу за шаблоном

Продовження таблиці 2.7

	getTemplate	templateId: string	Описує точку входу для отримання шаблону із сховища документів
TemplatesService	uploadTemplate	File: Express.Multer.File	Завантаження шаблону в систему
	compileTemplate	compileData: CompileTemplateDto	Генерація документу за шаблоном
	getTemplate	templateId: string	Отримання шаблону із сховища документів
	extractVariablesFromDOCX	data: Buffer	Розбір DOCX на індикатори підстановки
	compileDOCXTemplate	data: Buffer, values: Record<string, string>	Генерація DOCX файлу на базі шаблону та індикаторів підстановки

Продовження таблиці 2.7

GoogleApisBase	authorize	scope: string[]	Авторизація в сховище документів використовуючи OAuth2
DriveService	onModuleInit	-	Викликається під час ініціалізації модулю – відбувається авторизація та збереження OAuth2 клієнта
	createFolder	{ name }: { name: string }	Створення папки в сховищі документів
	createFile	{ name, body, mimeType, filderId }: { name: string, body: Stream, mimeType ?: string, folderId?: string }	Створення файлу в сховищі документів у вказаній папці

Продовження таблиці 2.7

	getFile	{ fileId }	Отримання файлу із сховища документів
	listFiles	{ folderId }: { folderId?: string }	Отримання списку файлів в сховищі у вказаній папці
	deleteFile	{ fileId }	Видалення файлу із сховища документів
FileTransformService	bufferToStream	buffer:Buffer | Uint8Array	Перетворення буферу даних в потік даних
	streamToBuffer	stream: Stream	Перетворення потоку даних в буфер даних
DbService	onModuleInit	-	Викликається під час ініціалізації модулю – відбувається під’єднання до бази даних
	onModuleDestroy	-	Викликається під час деініціалізації модулю – відбувається від’єднання від бази даних

Опис всі об’єктів трансформації даних (DTO) модулів застосунку наведений у таблиці 2.8.

Таблиця 2.8 – Опис об’єктів трансформації даних системи

Назва DTO	Поля DTO	Опис полей
CompileTemplateDTO	userId: string	Ідентифікатор користувача
	templateId: string	Ідентифікатор шаблону
	values: Record<string, string>	Об'єкт із індикаторами підстановки
AppResponseDTO	data?: T	Дані, які віддаються клієнту у відповідь
	error?: E	Помилка

В результаті модуль надає RestAPI із всіма необхідними для функціонування точками входу (таблиця 2.9).

Таблиця 2.9 – Опис точок входу модулюЯ

Шлях точки входу	HTTP метод	Призначення
/health	GET	Отримати поточний статус сервера
/templates	POST	Завантажити шаблон у систему
/templates/compile	POST	Згенерувати документ за шаблоном та динамічними даними
/templates/:templateId	GET	Отримати шаблон та його статистику

Продовження таблиці 2.9

/documents	POST	Завантаження документу у сховище документів
/documents	GET	Вивантаження документу з сховища документів

2.3 Аналіз безпеки даних

Для забезпечення цілісності файлів та захисту від несанкціонованих правок використовується один із алгоритмів перевірки цілісності файлів - створення хешу збереженого у сховищі файлу та порівняння цього хешу з хешем файлу при збереженні. Також дану перевірку можна зробити, порівнюючи два файли посимвольно, але такий метод повільніший на великих файлах та він може пропустити систематичні пошкодження, які можуть трапитися з обома файлами. Для того, щоб не виникало колізій при утворені хешів файлів, необхідно використовувати криптографічно сильні хеш-функції, які дають мало колізій. Як відомо, MD5 та хеш-функції сімейства SHA1 мають високий рівень колізій, тому була вибрана хеш-функція SHA256 [9].

Авторизація сервера у сховищі даних відбувається за допомогою OAuth2 клієнта, який дозволяє отримати токен доступу, оновити його та повторно спробувати запит. Даний спосіб авторизації вважається рекомендованим, так як наприклад JWT авторизація має певні недоліки в порівнянні з OAuth2.

Для гарантування безпеки вразливих даних необхідно використовувати шифрування та захищені з’єднання. Тому сервер та сховище документів спілкуються, використовуючи безпечне HTTPS з’єднання. Звернення до сервера модулю також здійснюється по захищеному HTTPS з’єднанню.

2.4 Висновки до розділу

Даний розділ описує основні етапи моделювання та конструювання розроблюваного модулю збереження та генерації типових документів. Для повного розуміння бізнес-процесів були реалізовані BPMN діаграми. На базі вимог та необхідного кінцевого набору функціоналу були вибрані технології та інструменти розробки, які дають можливість розроблювати програмний продукт ефективно та досягнути поставлених задач.

Також були описані класи та модулі програмного забезпечення, описана структура сховища документів та представлена схема бази даних із необхідними таблицями, перелічуваними значеннями тощо.

Не менш важливим пунктом у даному розділі був аналіз безпеки даних, який показав достатній рівень безпеки розроблюваного модулю. Були описані процеси автентифікації сервера у сховищі даних і процес перевірки консистентності документів для запобігання їх компрометування. Були пояснені вибрані підходи та їх переваги.

3 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Аналіз якості програмного забезпечення

Тестування програмного забезпечення – важлива частина розробки програмного забезпечення, оскільки всі ми допускаємо помилки. Деякі з цих помилок є не важливими, але деякі з них є небезпечними.

Наявність тестування важлива через ряд наступних причин:

- тестування програмного забезпечення насправді потрібно, щоб вказати на дефекти та помилки, допущені на етапах розробки;
- тестування необхідне для ефективної роботи програмного забезпечення чи продукту;
- програма не повинна спричиняти будь-яких збоїв, оскільки це може привести до проблем в майбутньому або на пізніх стадіях розробки;
- тестування необхідне для того, щоб доставляти високоякісний продукт, що вимагає менших витрат на технічне обслуговування, і витікає в більш точні, послідовні та надійні результати.

Тестування серверної частини може проходити багатьма способами. Найпоширеніші з них є наступні:

- модульне тестування (англ. Unit testing) – це метод тестування, при якому незалежно тестуються всі модулі (найменші частини коду, які можуть бути протестовані) програми [10];
- інтеграційне тестування (англ. Integration testing) – це тестування, яке відбувається на рівень вище від модульного, передбачаючи тестування зв’язки модулів та їх роботи разом [11].

Також важливими видами тестів є статичний аналіз коду [12] та тестування на покриття коду [13]:

- статичний аналіз коду виконується без запуску програми та дозволяє відловити помилки на кшталт помилок програмування, вад, порушень стилю тощо;
- тестування на покриття коду дозволяє перевірити, чи тестування проходить по всіх можливих гілках виконання.

Проходження всіх вищезазначених тестів повинно відбуватись перед кожним розгортанням сервера. Після закінчення тестування потрібно генерувати звіт із наступними показниками:

- список знайдених проблем;
- опис критичних проблем;
- висновок про результати тестування – продовження розгортання чи зупинка процесу розгортання через невдало пройдене тестування.

3.2 Опис процесу тестування

Тестування виконувалось на сервері із наступними характеристиками,:

- 4 ГБ оперативної пам'яті;
- 10 ГБ постійного сховища;
- 8 процессорів.

В ході тестування було виконане модульне й інтеграційне тестування та статичний аналіз коду. Під час виявлення помилок, вони були детально задокументовані у GitHub Issues або виправлені, якщо це критично впливало на необхідний функціонал.

3.2.1 Документація функціональних тестів

Таблиця 3.1 – Опис тестових кейсів

Ідентифікатор тестового кейсу	Компонент	Опис
AUTH-1	Авторизація	Перевірити можливість авторизації модулю в Google API з коректними даними для авторизації
AUTH-2	Авторизація	Перевірити неможливість авторизації модулю в Google API з коректними даними для авторизації
TEM-1	Шаблон	Перевірити можливість завантаження шаблона в систему
TEM-2	Шаблон	Перевірити можливість розбору шаблона на спільні дані та на індикатори підстановки
TEM-3	Шаблон	Перевірити можливість розбору шаблона на спільні дані та на індикатори підстановки, якщо у шаблоні немає жодного індикатора підстановки
TEM-4	Шаблон	Перевірити неможливість розбору шаблона на спільні дані та на індикатори підстановки, якщо у шаблоні неправильний формат
TEM-5	Шаблон	Перевірити неможливість завантаження шаблона в систему, якщо у шаблоні неправильний формат

Продовження таблиці 3.1

ТЕМ-6	Шаблон	Перевірити можливість генерації документу за шаблоном та динамічними даними
ТЕМ-7	Шаблон	Перевірити неможливість генерації документу за шаблоном та динамічними даними, якщо у відповідного шаблона не існує
ТЕМ-8	Шаблон	Перевірити неможливість генерації документу за шаблоном та динамічними даними, якщо передані не всі динамічні дані
ТЕМ-9	Шаблон	Перевірити можливість генерації документу за шаблоном та динамічними даними, якщо шаблон не містив жодного індикатора підстановки
ТЕМ-10	Шаблон	Перевірити можливість отримання шаблону із сховища даних
ТЕМ-11	Шаблон	Перевірити неможливість отримання шаблону із сховища даних, якщо відповідного шаблону не існує
ТЕМ-12	Шаблон	Перевірити можливість отримання статистики шаблону із сховища даних
ТЕМ-13	Шаблон	Перевірити неможливість отримання статистики шаблону із сховища даних, якщо відповідного шаблону не існує

Продовження таблиці 3.1

ТЕМ-14	Шаблон	Перевірити можливість отримання статистики шаблону із сховища даних, якщо по шаблону не виконувалось жодних операцій
ST-1	Сховище	Перевірити можливість завантаження документу у сховище даних
ST-2	Сховище	Перевірити неможливість завантаження документу у сховище даних, якщо розмір файлу перевищує максимальний розмір дозволеного файлу
ST-3	Сховище	Перевірити можливість вивантаження документу з сховища даних
ST-4	Сховище	Перевірити неможливість вивантаження документу з сховища даних, якщо відповідного документу не існує
ST-5	Сховище	Перевірити можливість створення папки у сховищі даних
ST-6	Сховище	Перевірити неможливість створення папки у сховищі даних, якщо папка з даним ім’ям вже існує на цьому рівні
ST-6	Сховище	Перевірити неможливість створення файлу у сховищі даних, якщо файл з даним ім’ям вже існує на цьому рівні
VER-1	Версіонування	Перевірити можливість створення нової версії файлу, якщо минула версія файлу існувала у системі

Продовження таблиці 3.1

VER-2	Версіонування	Перевірити можливість створення нової версії файлу, якщо минула версія файлу не існувала у системі

3.2.2 Звіт тестування

У таблиці 3.2 наведено звіт про результати тестування, що включає в себе всі тестові кейси, описані в таблиці 3.1.

Таблиця 3.2 – Звіт тестування

Ідентифікатор тесту	Очікуваний результат	Фактичний результат	Статус тесту
AUTH-1	Модуль може авторизуватись в GoogleAPI використовуючи правильні дані	Модуль може авторизуватись в GoogleAPI використовуючи правильні дані	Пройдено
AUTH-2	Модуль не може авторизуватись в GoogleAPI використовуючи не правильні дані	Модуль не може авторизуватись в GoogleAPI використовуючи не правильні дані	Пройдено

Продовження таблиці 3.2

ТЕМ-1	Користувач може завантажити шаблон у правильному форматі у систему	Користувач може завантажити шаблон у правильному форматі у систему	Пройдено
ТЕМ-2	Модуль може розібрати шаблон на спільні дані та на індикатори підстановки	Модуль може розібрати шаблон на спільні дані та на індикатори підстановки	Пройдено
ТЕМ-3	Модуль може розібрати шаблон на спільні дані та на індикатори підстановки, якщо у шаблоні немає жодного індикатора підстановки	Модуль може розібрати шаблон на спільні дані та на індикатори підстановки, якщо у шаблоні немає жодного індикатора підстановки	Пройдено

Продовження таблиці 3.2

ТЕМ-4	Модуль не може розібрати шаблон на спільні дані та на індикатори підстановки, якщо у шаблоні неправильний формат	Модуль не може розібрати шаблон на спільні дані та на індикатори підстановки, якщо у шаблоні неправильний формат	Пройдено
ТЕМ-5	Користувач не може завантажити шаблон у систему, якщо у шаблоні неправильний формат	Користувач не може завантажити шаблон у систему, якщо у шаблоні неправильний формат	Пройдено
ТЕМ-6	Користувач може згенерувати документ за шаблоном та динамічними даними	Користувач може згенерувати документ за шаблоном та динамічними даними	Пройдено

Продовження таблиці 3.2

ТЕМ-7	Користувач не може згенерувати документ за шаблоном та динамічними даними, якщо у відповідного шаблона не існує	Користувач не може згенерувати документ за шаблоном та динамічними даними, якщо у відповідного шаблона не існує	Пройдено
ТЕМ-8	Користувач не може згенерувати документ за шаблоном та динамічними даними, якщо передані не всі динамічні дані	Користувач не може згенерувати документ за шаблоном та динамічними даними, якщо передані не всі динамічні дані	Пройдено

Продовження таблиці 3.2

ТЕМ-9	Користувач може згенерувати документ за шаблоном та динамічними даними, якщо шаблон не містив жодного індикатора підстановки	Користувач може згенерувати документ за шаблоном та динамічними даними, якщо шаблон не містив жодного індикатора підстановки	Пройдено
ТЕМ-10	Користувач може отримати шаблон із сховища даних	Користувач може отримати шаблону із сховища даних	Пройдено
ТЕМ-11	Користувач не може отримати шаблон із сховища даних, якщо відповідного шаблону не існує	Користувач не може отримати шаблон із сховища даних, якщо відповідного шаблону не існує	Пройдено

Продовження таблиці 3.2

ТЕМ-12	Користувач може отримати статистику шаблону із сховища даних	Користувач може отримати статистику шаблону із сховища даних	Пройдено
ТЕМ-13	Користувач не може отримати статистику шаблону із сховища даних, якщо відповідного шаблону не існує	Користувач не може отримати статистику шаблону із сховища даних, якщо відповідного шаблону не існує	Пройдено
ТЕМ-14	Користувач може отримати статистику шаблону із сховища даних, якщо по шаблону не виконувалось жодних операцій	Користувач може отримати статистику шаблону із сховища даних, якщо по шаблону не виконувалось жодних операцій	Пройдено
ST-1	Модуль може завантажити документ у сховище даних	Модуль може завантажити документ у сховище даних	Пройдено

Продовження таблиці 3.2

ST-2	Модуль не може завантажити документ у сховище даних, якщо розмір файлу перевищує максимальний розмір дозволеного файлу	Модуль не може завантажити документ у сховище даних, якщо розмір файлу перевищує максимальний розмір дозволеного файлу о файлу	Пройдено
ST-3	Модуль може вивантажити документ з сховища даних	Модуль може вивантажити документ з сховища даних	Пройдено
ST-4	Модуль не може вивантажити документ з сховища даних, якщо відповідного документу не існує	Модуль не може вивантажити документ з сховища даних, якщо відповідного документу не існує	Пройдено
ST-5	Модуль може створити папку у сховищі даних	Модуль може створити папку у сховищі даних	Пройдено

Продовження таблиці 3.2

ST-6	Модуль не може створити папку у сховищі даних, якщо папка з даним ім’ям вже існує на цьому рівні	Модуль не може створити папку у сховищі даних, якщо папка з даним ім’ям вже існує на цьому рівні	Пройдено
ST-6	Модуль не може створити файл у сховищі даних, якщо файл з даним ім’ям вже існує на цьому рівні	Модуль не може створити файл у сховищі даних, якщо файл з даним ім’ям вже існує на цьому рівні	Пройдено
VER-1	Модуль може створити нову версію файлу, якщо минула версія файлу існувала у системі	Модуль може створити нову версію файлу, якщо минула версія файлу існувала у системі	Пройдено

Продовження таблиці 3.2

VER-2	Модуль може створити нову версію файлу, якщо минула версія файлу не існувала у системі	Модуль може створити нову версію файлу, якщо минула версія файлу не існувала у системі	Пройдено

3.3 Знайдені недоліки та висновки

Під час тестування розроблюваного програмного забезпечення було проведено аналіз якості даногопрограмного забезпечення.

Було визначено найважливіші етапи в тестуванні серверних застосувань. На базі цього були проведені модульне й інтеграційне тестування та статичний аналіз коду.

Були виявлені загальні критерії тестування і та розроблений детальний план тестування згідно сценаріям використання розроблюваної системи.

У результаті тестування було розроблено план тестування і виявлено, що серверний застосунок задовільняє всі вимоги, та готовий до розгортання в робочому середовищі.

4 ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Розгортання програмного забезпечення

Розгортання модулю збереження та генерації типових документів може відбуватись наступним чином - або за допомогою утиліт контейнеризації Docker [4], або вручну.

Docker є зручнішим варіантом, так як даний крок не вимагає ніякої попередньої підготовки системи. Для розгортання в такому випадку необхідно скачати код модулю та запустити всі необхідні контейнери за допомогою команди: «docker-compose up -d».

Для запуску програмного забезпечення вручну необхідно проробити наступні кроки:

а) встановити на сервер систему контролю версій git, середовище виконання Node.js (v14.17.0), базу даних PostgreSQL (v12.7);

б) завантажити код модулю на сервер;

в) законфігурувати змінні оточення для підключення до сховища документів та бази даних;

г) створити базу даних;

д) запустити модуль (npm start).

4.2 Робота з програмним забезпеченням

Після запуску модуль буде продовжувати працювати безперебійно в автономному режимі. Деталі подальшої роботи представлені у документі КПІ.ІП-7104.7126.045430.02.34 Керівництво користувача.

4.3 Висновки до розділу

У цьому розділі були описані способи розгортання модулю створення подій та узгодження документів.

ВИСНОВКИ

В рамках роботи над індивідуальною частиною дипломного проєкту був створена система зберігання документів дипломного проектування. Даний модуль інтегрується в модуль створення подій та узгодження документів та є частиною цілої системи.

Спочатку був проведений детальний аналіз предметної області та розглянуті основні сценарії використання розроблюваного модулю системи, також були розглянуті існуючі рішення на ринку та підходи до вирішення задачі. На базі цих даних були виведені відповідні вимоги та рішення поставленої задачі.

Була розроблена архітектура програмного забезпечення та на базі цього був вибраний технологічний стек модулю. Також був проведений аналіз безпеки даних на всіх рівнях застосунку.

Для підвищення якості розроблюваного програмного забезпечення було описане і проведене детальне тестування модулю, що дозволило знайти проблеми в роботі та виправити їх.

Останнім кроком було розглянуто впровадження та розгортання системи для її подальшого використання.

ПЕРЕЛІК ПОСИЛАНЬ

1) M. E. Doc [Електронний ресурс] – Режим доступу до ресурсу: https://medoc.ua/.

2) XpertDoc [Електронний ресурс] – Режим доступу до ресурсу: https://www.xpertdoc.com/en/.

3) Legito [Електронний ресурс] – Режим доступу до ресурсу: https://www.legito.com/UA/en.

4) Rest API Wikipedia [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Representational_state_transfer.

5) Docker Docs [Електронний ресурс] – Режим доступу до ресурсу: https://docs.docker.com/.

6) Node.js Docs [Електронний ресурс] – Режим доступу до ресурсу: https://nodejs.org/en/docs/.

7) Nest.js docs [Електронний ресурс] – Режим доступу до ресурсу: https://docs.nestjs.com/.

8) Google Drive API Docs [Електронний ресурс] – Режим доступу до ресурсу: https://developers.google.com/drive.

9) OWASP Hashing best practices [Електронний ресурс] – Режим доступу до ресурсу: https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html.

10) Модульне тестування [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D1%83%D0%BB%D1%8C%D0%BD%D0%B5_%D1%82%D0%B5%D1%81%D1%82%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F.

11) Інтеграційне тестування [Електронний ресурс] – Режим доступу до ресурсу:

https://uk.wikipedia.org/wiki/%D0%86%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%B0%D1%86%D1%96%D0%B9%D0%BD%D0%B5_%D1%82%D0%B5%D1%81%D1%82%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F#:~:text=%D0%86%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%B0%D1%86%D1%96%D0%B9%D0%BD%D0%B5%20%D1%82%D0%B5%D1%81%D1%82%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F%20(%D0%B0%D0%BD%D0%B3%D0%BB.,%D0%BF%D0%B5%D1%80%D0%B5%D0%B4%20%D0%B2%D0%B5%D1%80%D0%B8%D1%84%D1%96%D0%BA%D0%B0%D1%86%D1%96%D1%94%D1%8E%20%D1%82%D0%B0%20%D0%B2%D0%B0%D0%BB%D1%96%D0%B4%D0%B0%D1%86%D1%96%D1%94%D1%8E%20%D0%9F%D0%97..

12) Статичний аналіз коду [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/%D0%A1%D1%82%D0%B0%D1%82%D0%B8%D1%87%D0%BD%D0%B8%D0%B9_%D0%B0%D0%BD%D0%B0%D0%BB%D1%96%D0%B7_%D0%BA%D0%BE%D0%B4%D1%83.

13) Покриття коду ВІкіпедія [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BA%D1%80%D0%B8%D1%82%D1%82%D1%8F_%D0%BA%D0%BE%D0%B4%D1%83#:~:text=code%20coverage%2C%20tests%20coverage)%20%E2%80%94,%D0%B4%D0%BB%D1%8F%20%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D1%87%D0%BD%D0%BE%D0%B3%D0%BE%20%D1%82%D0%B5%D1%81%D1%82%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F%20%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE%20%D0%B7%D0%B0%D0%B1%D0%B5%D0%B7%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%BD%D1%8F..