Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук

Кафедра Програмної інженерії
(повна назва)

АТЕСТАЦІЙНА РОБОТА
Пояснювальна записка

другий (магістерський)
(рівень вищої освіти)

Дослідження методів обробки даних у файлах docx для тестування контенту на виявлення помилок щодо вимог до оформлення тексту
(тема)

Виконав: студент 2 курсу, групи ІПЗм-18-3

спеціальність 121

Інженерія програмного забезпечення
(код і повна назва спеціальності)

Освітньо-наукової програми

Інженерія програмного забезпечення
(повна назва освітньої програми)

Іорданов Сергій Олексійович
(прізвище, ініціали)

Керівник доц. Ревенчук І. А.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ проф. Дудар З.В.
(підпис) (прізвище, ініціали)

2020 р.Харківський національний університет радіоелектроніки

Факультет Комп’ютерних наук

Кафедра Програмної інженерії

Рівень вищої освіти другий (магістерський)

Спеціальність 121 - Інженерія програмного забезпечення

Освітньо-наукова програма Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ

Зав. кафедри _____
(підпис)

«___»_____ 2020 р.

ЗАВДАННЯ

НА АТЕСТАЦІЙНУ РОБОТУ

студентові Іорданову Сергію Олексійовичу

1. Тема роботи: Дослідження методів обробки даних у файлах docx для тестування контенту на виявлення помилок щодо вимог до оформлення тексту
затверджена наказом по університету від 27.03.2020 № 473 .
2. Термін подання студентом роботи до екзаменаційної комісії
18.04.2020 р.
3. Вихідні дані до роботи електронні ресурси за обраною тематикою, мінімальні вимоги до функціональності програми, загальні вимоги до архітектури системи.
4. Перелік питань, що потрібно опрацювати в роботі Аналіз предметної галузі, дослідження методів обробки даних у документах формату .docx, опис програмної системи, опис можливостей використання досліджених методів та розробленої системи.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) Архітектура програмної системи, дизайн компоненту перевірки форматування, інтерфейс користувача, приклади реалізації алгоритмів, висновки, слайди презентації.

6. Консультанти розділів роботи

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

	№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка*
		Аналіз предметної галузі і постановка задачі	02.04.2020	
		Дослідження методів обробки даних у документах формату .docx	08.04.2020	
		Розробка специфікації ПЗ	10.04.2020	
		Кодування, тестування і налагодження програми	30.04.2020	
		Підготовка пояснювальної записки	04.05.2020	
		Підготовка презентації та доповіді	11.05.2020	
		Нормоконтроль, Рецензування	12.05.2020	
		Попередній захист	13.05.2020	
		Занесення роботи в електронний архів	14.05.2020	
		Допуск до захисту у зав. кафедри	15.05.2020	
* заповнюється вручну після виконання чергового пункту				

Дата видачі завдання 30.03.2020 р.

Студент _____
(підпис)

Керівник роботи _____ к. т. н. доц. Ревенчук І. А
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ / ABSTRACT

Пояснювальна записка до атестаційної роботи: 39 с., 14 рис., 1 табл., 4 додатки, 10 джерел(о/а).

ЕЛЕКТРОННІ ДОКУМЕНТИ, ДОКУМЕНТООБІГ, АНАЛІЗ СТРУКТУР ДАНИХ, ПЕРЕВІРКА ФОРМАТУВАННЯ, MICROSOFT WORD, DOCX, PDF, .NET

Об’єктом дослідження є методи обробки даних у файлах docx з метою перевірки документу на відповідність вимогам щодо його оформлення.

Метою роботи є аналіз методів роботи з електронними документами у форматі docx та можливості комбінування їх з методами обробки інших типів документів для перевірки форматування (оформлення) документу.

В роботі проведено аналіз предметної галузі, здійснено дослідження основних методів, моделей та алгоритмів аналізу відповідності електронного документу до вимог оформлення. В результаті аналізу виявлено декілька найбільш ефективних методів та досліджено можливість їх поєднання в процесі програмної реалізації.

ELECTRONIC DOCUMENTS, DATA STRUCTURES ANALYSIS, FORMATTING VALIDATION, MICROSOFT WORD, DOCX, PDF, .NET

The object of the study is the methods of processing the data in docx files in order to verify the document for compliance with its design requirements.

The purpose of this work is to analyze the methods of working with electronic documents in docx format and the possibility of combining them with methods of processing other types of documents to check the formatting (layout) of the document.

The article analyzes the subject area, researches the basic methods, models and algorithms for analyzing the compliance of an electronic document to the requirements of design. The analysis revealed several of the most effective methods and the possibility of combining during the software implementation was investigated.

ЗМІСТ

ВСТУП

Документообіг, як процес, притаманний майже будь-якій організації. Процес поділяється на декілька кроків і найважливішими принципами процесу є прямий та швидкий процес переходи з однієї стадії до іншої та мінімізація повернень документа в обрубку до попередньої стадії. Найбільш вразливою стадією з найбільшою кількістю повернень є стадія перевірки змісту та оформлення документів. Для перевірки змісту часто необхідна участь людини у процесі, а у випадках, коли людину можна замінити використовуються вже розроблені та поширені засоби OCR.

Значно більшою проблемою є перевірка оформлення документа відносно певних правил. Ця проблема є найбільш актуальною для великих документів, що мають бід собою творчу складову та все ж таки мають відповідати чітко прописаним та задокументованим правилам оформлення, як то наукові роботи, реферати, атестаційні роботи, звіти у різних галузях та різної направленості.

Документи що потрапляють до стадії перевірки правил оформлення у переважній більшості випадків обробляються електронно у форматі Microsoft Office – docx. Через особливості своєї структури, документи такого типу мають багато проблем у питані можливості верифікації його на відповідність правилам оформлення. На сьогоднішній день на ринку немає продукту що дозволив би автоматизувати процес перевірки документа щодо параметрів оформлення і для цього все ще необхідна кропітка, рутинна робота людини, що призводить до сповільнення процесу документообігу та помилок, що виникають через людський фактор.

Метою роботи є аналіз методів роботи з електронними документами у форматі docx та можливості комбінування їх з методами обробки інших типів документів для перевірки форматування (оформлення) документу. Необхідно провести аналіз

предметної галузі, моделей та алгоритмів аналізу відповідності електронного документу до вимог оформлення.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ І ПОСТАНОВКА ЗАДАЧІ

Предметною галуззю дослідження є документообіг у будь-яких державних чи приватних підприємствах, так само як і між фізичними особами. Більш конкретизованою частиною цієї галузі, у рамках якої проводиться дослідження, можна назвати перевірку документів на відповідність певним стандартам їх оформлення.

1.1 Електронний та стандартний документообіг

Документообіг являє собою повний процес життєвого циклу документа від його оформлення до архівації чи утилізації. Цей процес повністю покриває роботу певної організації чи підприємства з будь-якими документами тому поділяється на декілька фаз та типів.

У першу чергу треба звернути увагу на те, що документообіг може бути електронним чи стандартним (рис. 1.1).

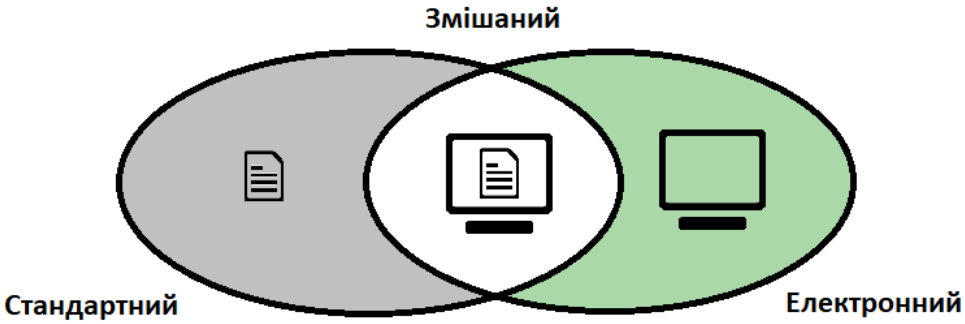

Рисунок 1.1 – Види документообігу

Це залежить від того, чи потрапляє документ під час якогось з етапів свого життєвого циклу на папір. Якщо так, то документообіг цього документу є стандартним. Історично, стандартний документообіг вважається найпопулярнішим, бо у цілому не мав альтернатив. Насьогодні ж існує тенденція переведення якнайбільшої кількості документів в електронний вигляд. Процес переходу до електронного документообігу не є тривіальним, особливо для великих організацій або ж для цілих країн. Саме через це наразі найчастіше зустрічається змішаний тип документообігу, коли частина документів залишаються в електронному вигляді, а інша частина має зберігатися на паперових носіях [1].

Документи, що приймають участь у процесі документообігу можна поділити на три великі групи [2]:

- вхідні – документи, що потрапляють до організації із зовнішнього світу. Наприклад, резюме, що кандидат на посаду присилає до компанії, чи накладна, що поступає до компанії з певним товаром;
- вихідні – документи, що організація відправляє до іншої організації, людини, держави тощо. Прикладом може бути квитанція про оплату якоїсь послуги, що видає клієнту компанія;
- внутрішні – документи, що приймають участь у документообігу всередині організації (наприклад, заява співробітника про звільнення з компанії).

За формою, документообіг також поділяють на централізований, децентралізований та змішаний [3]:

- централізованих – за принципом акумуляції усіх документів в одному спеціалізованому підрозділі;
- децентралізований – за принципом розподіли документообігу між декількома підрозділами;
- змішаний – за принципом поєднання обох форм.

Далі класифікація документів змінюється в залежності від структури чи організації, що ними оперує. Та існує кілька типів документів, що приймають участь у документообігу майже будь-якої організації. До них можна віднести управлінські, фінансові, архівні, складські, кадрові, технологічні та виробничі [4]. Деякі можуть бути відсутні через специфіку діяльності організації або ж навпаки, можуть буди додані інші типи. Схематично типові види документів у документообігу зображено на рисунку 1.2.

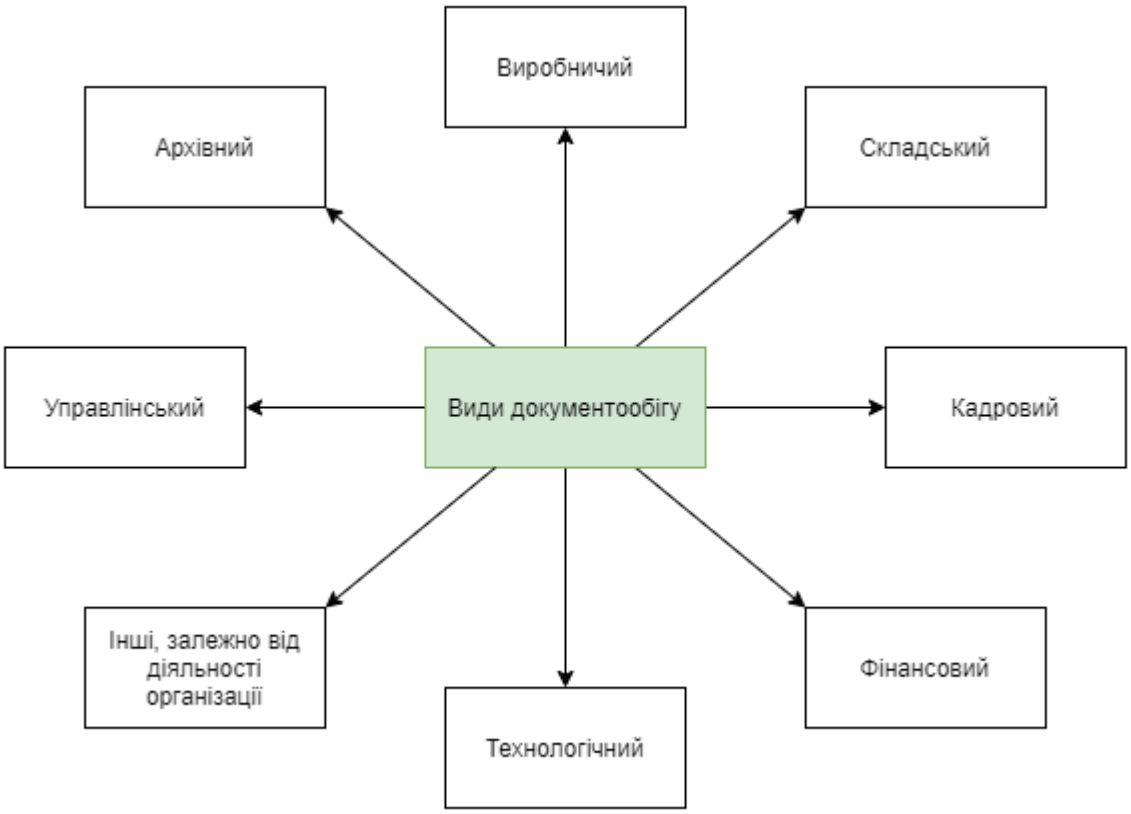

Рисунок 1.2

Життєвий цикл документу також змінюється в залежності від його типу, організації, що ним оперує та процесу документообігу в організації. Але, як і з типами документів, є загальний процес, який проходить документ і він працює за

принципом конвеєра. Відрізняється опрацювання вхідних, вихідних та внутрішніх документів.

Для вхідних документів життєвий цикл виглядає наступним чином:

- надходження документа до організації;
- первинна обробка та перевірка на відповідність стандартам;
- реєстрація документа у відповідному реєстрі;
- передача документа відповідальному управлінню;
- (опціонально) створення внутрішніх документів на основі документа що надійшов;
- передача документа на виконання;
- архівація чи утилізація.

Для вихідних документів найчастіше притаманні наступні кроки обробки:

- формування та оформлення документа;
- перевірка змісту;
- перевірка оформлення;
- підпис уповноваженим співробітником;
- створення копії для зберігання в організації;
- реєстрація документа у відповідному реєстрі вихідних документів;
- відправка документа отримувачу.

Щодо внутрішніх документів, тут виділити чіткі кроку дещо важче але до процесу роботи з внутрішніми документами можна віднести наступні кроки:

- формування та оформлення документа;
- (за необхідністю) перевірка змісту та оформлення;
- підпис уповноваженим співробітником;
- реєстрація документа у відповідному реєстрі внутрішніх документів;
- (за необхідністю) передача документа на виконання;
- архівація чи утилізація.

У цілому, організація документообігу побудована за принципом конвеєру. Що веде до наступних правил обробки документу. Документ має переходити від одної стадії до іншої послідовно та без зайвих посередників. Протягом пересування документа по «конвеєру» обробки необхідно мінімізувати кількість повернень його до попередніх кроків.

1.2 Проблема перевірки форматування

Висновком пункту 1.1 цієї роботи стали два важливих принципи, що є основою побудови ефективного документообігу в організації.

Перший полягає в необхідності побудови «конвеєру» обробки вхідних, вихідних та внутрішніх документів та організації ефективного переходу документів з одної стадії до наступної на шляху їх обробки. Цю проблему в загальному випадку вирішують наступними методами: чітким вказанням послідовності роботи з усіма типами документів, автоматизацією рутинних кроків та переводи максимальної кількості документів в електронний вигляд, що істотно спрощує роботу з ними та ведення аудиту.

Другий принцип - це необхідність мінімізувати кількість повернень документу до попередньої стадії його обробки. Можна виділити наступні основні етапи використання документів:

- формування документа;
- перевірка змісту;
- перевірка оформлення;
- підпис, реєстрація та архівація.

Найбільш ймовірним є повернення документу на попередні стадії на етапах перевірки змісту та оформлення документу. Проблема також полягає у рутинності та затратах часу, що необхідні на проходження цих двох стадій.

Вирішення проблеми повернення документа на цих двох стадіях полягає у максимальній автоматизації процесу з мінімальним залученням людини до нього. На етапі перевірки змісту це не завжди є можливим але існують випадки у яких зміст документу може бути перевірений автоматично. Для прикладу можна привести автоматичну перевірку рахунків, що приходять до компанії від постачальників сировини. Система сканує рахунок, методами розпізнавання контенту (OCR) та структурує отримані данні. Далі вона звертається до внутрішніх систем компанії, де зберігається інформація про закази компанії та отримані товари на складі. Виконавши порівняння отриманих даних (процес зветься Three Way Matching[5]) система може сказати, чи правильний рахунок вислав постачальник та, якщо дані збігаються, автоматично провести оплату за рахунком. Часто такі системи створюються специфічно під компанію та існують і глобальні рішення, як то частина функціоналу ERP системи Microsoft Dynamics, що відповідає за Three Way Matching.

Зовсім інша ситуація з перевіркою форматування. Цей процес не є складним у випадку, якщо документ складається з однієї-двох сторінок або ж являє собою шаблон з декількома полями для заповнення. Проблема тут виникає, коли документ не має строго визначеного шаблону та має певні чіткі правила для його оформлення. Найяскравішими прикладами такого типу документів є наукові праці, студентські атестаційні роботи, дисертації тощо. Оформлення такого типу документів чітко описано у ДСТУ 3008-2015 «Звіти у сфері науки і техніки» [6]. Ситуація також стає складнішою через великі об’єми таких документів.

Наразі не існує системи, що за заданими параметрами оформлення могла б перевірити відповідність документа цим правилам. Відсутність такої системи на ринку зумовлена певними труднощами роботи з форматом у якому зберігається майже уся документація на етапі перевірки оформлення. Мова йде про формат docx,

що є основним для роботи з документами з використанням текстового редактора Microsoft Word.

1.3 Постановка задачі

У наш час існує багато установ, що працюють з постійним потоком документів, що мають відповідати певним строго визначеним вимогам форматування та оформлення. Більшість з таких організацій стикаються з проблемою перевірки оформлення документів, що є рутинною роботою та займає значну частину робочого часу спеціалістів.

У переважаючій більшості випадків уся документація у момент написання та перевірки зберігається у форматі docx (у деяких випадках у застарілому doc). Таким чином, для автоматизації стадії перевірки оформлення документа необхідно розробити механізм роботи зі структурою файлів у форматі docx та навчитися проводити перевірку цієї структури на відповідність заданим правилам.

Основним завданням даної роботи є дослідження методів обробки даних у файлах docx для тестування контенту на виявлення помилок щодо невідповідностей оформлення тексту. Для підтвердження доцільності методу необхідно розробити систему, що буде використовувати його для перевірки оформлення певного типу документів.

У якості предметної галузі для перевірки систему було вирішено обрати процес нормоконтролю у вищих навчальних закладах, а саме перевірку відповідності атестаційних робот магістерського та бакалаврського рівня на відповідність ДСТУ 3008-2015 «Звіти у сфері науки і техніки» [7].

Результатом роботи має стати система, що використовує досліджений метод обробки електронних документів та дозволяє отримавши на вхід документ у форматі docx на виході вивести список помилок оформлення, що були знайдені протягом аналізу документа на правильність оформлення.

2 ДОСЛІДЖЕННЯ МЕТОДІВ ОБРОБКИ ДАНИХ У ДОКУМЕНТАХ ФОРМАТУ DOCX

У даному розділі наведено аналіз та дослідження методів обробки даних у документах формату docx, а також можливість комбінування методів та використання декількох форматів файлів для реалізації можливості перевірки відповідності документа до вимог оформлення.

2.1 Дослідження можливостей формату docx

У першу чергу для аналізу методів роботи з документами у форматі docx потрібно зрозуміти як працює docx, що він являє собою структурно та які особливості має. Також необхідно провести дослідження можливостей використання структурних елементів docx для аналізу відповідності документа до правил оформлення.

Приклад структури файлів, що архівовані у форматі docx наведено на рисунку 2.1.

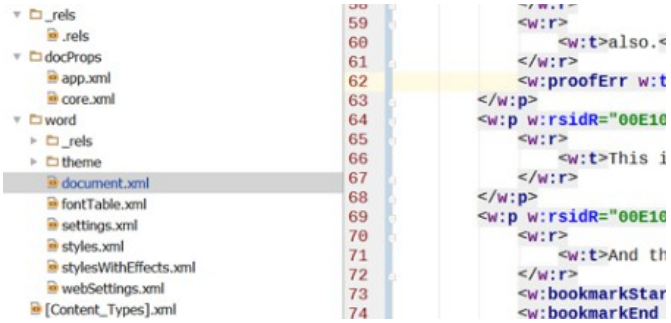

Рисунок 2.1 – Структура XML файлів формату docx

Формат docx являє собою набір XML файлів зібраних у ZIP архів.

2.1.1 Аналіз структурних особливостей OpenXML

Формат кожного файлу в архіві описано у специфікацій «Office Open XML» описаній стандартом ECMA-376 [7]. За специфікацією Office Open XML (OpenXML) - це запропонований відкритий стандарт для обробки текстових документів, презентацій та електронних таблиць який може вільно реалізовуватися декількома додатками на декількох платформах. Його створення приносить користь організаціям, які мають намір впроваджувати програми, здатні використовувати формат, комерційні та урядові організації, які закуповують таке програмне забезпечення, та освітяни чи автори, які викладають формат. Зрештою, всі користувачі користуються перевагами стандарту XML для своїх документів, включаючи стабільність, збереження, сумісність та постійну еволюцію [7].

Навіть при створенні документа з одним словом, процес збереження в Microsoft Word створює теми за замовчуванням, властивості документа, таблиці шрифтів тощо у форматі XML.

У стандарті описано шість властивостей, що характеризують Office Open XML та на які слід спиратися при його досліджені:

- "Інтероперабельність" описує, як незалежність OpenXML від патентованих форматів, функцій та часу виконання оточення, що спрощує використання його у додатках різного типу;
- "Інтернаціоналізація" описує кілька репрезентативних способів, за допомогою яких OpenXML підтримує кожну основну мовну групу;
- "Низький бар'єр для прийняття розробника", "Компактність" та "Модульність" перераховують конкретні способи, якими OpenXML уникає

або усуває практичні перешкоди для впровадження різними сторонами: крива навчання, мінімальний набір функцій та виконання;

- "Міграція з високою точністю" описує, як OpenXML виконує мету захисту для збереження інформації, в тому числі повний намір творця оригіналу в існуючих та нових документах;
- "Інтеграція з бізнес-даними" описує, як OpenXML включає в себе службову інформацію в спеціальні схеми для можливості інтегрувати та використовувати повторно інформацію між додатками та інформаційними системами;
- «Місце для інновацій» описує, як OpenXML готується до майбутнього, визначаючи подальші механізми розширення та забезпечення сумісності між додатками з різними наборами функцій.

Open Packaging Conventions (OPC) забезпечують спосіб зберігання декількох типів вмісту (наприклад, XML, зображень та метаданих) у контейнер, такий як ZIP-архів, щоб повністю представляти документ [8].

Рекомендована реалізація для OPC використовує формат архіву ZIP. Можна перевірити структуру будь-якого файлу OpenXML за допомогою будь-якого переглядача ZIP. За логікою, документ OpenXML - це пакет OPC.

Пакет - це плоска колекція частин. Кожна частина має нечутливу до регістру назву частини, яка складається з косої послідовності імен сегментів, таких як «/pres/slides/slide1.xml». Кожна частина також має тип вмісту. Фізично ZIP-архів - це один пакет, кожен ZIP-файл в архіві – одна частина, і імена шляхів в архіві ZIP безпосередньо відповідають назвам частин. У реалізації ZIP «/[Content_Types].xml» дозволяє споживачеві визначати тип вмісту кожної частини в пакеті. Частина пакету, що відповідає за відношення називається «_rels/.rels». Приклад змісту «_rels/.rels» наведено на рис. 2.2.

Пакети можуть містити явні зв’язки з іншими частинами пакету, а також із зовнішніми ресурсами. Кожне явне відношення має ідентифікатор, який дозволяє

вмісту частини посилатися на нього і тип, який дозволяє додатку вирішити, як його обробити. Частина, що описує відношення для певної частини пакету OpenXML (наприклад «/a/b/c.xml») зберігається під назвою «/a/b/_rels/c.xml.rels».

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Relationships xmlns="http://schemas.openxmlformats.org/package/2006/relationships">
   <Relationship Id="rId1"
Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/officeDocument"
                 Target="word/document.xml"/>
</Relationships>
```

Рисунок 2.2 – Зміст «_rels/.rels» із посиланням на документ

Щоб відкрити пакет, програма повинна проаналізувати частину з відношеннями та слідувати зв’язкам відповідного типу. Усі інші частини документа OpenXML містять OpenXML, спеціальний XML або вміст довільного типу, наприклад мультимедіа об’єкти. Здатність частини зберігати власні XML - це особливо потужний механізм вбудовування бізнес-даних та метаданих.

2.1.2 WordprocessingML як реалізація OpenXML для Microsoft Word

WordprocessingML є реалізацією стандарту OpenXML та описує документи формату docx. Документ WordprocessingML складається із збірки історій (stories). Історії поділяють на основний документ, словниковий документ, піддокумент, заголовок, колонтитул, коментар, фрейм, текстове поле, виноска та кінцева примітка.

Єдина необхідна історія – це основний документ. Його тип – http://schemas.openxmlformats.org/officeDocument/2006/relationships/officeDocument. Типовий шлях від кореня до листа в дереві XML міститиме такі елементи XML:

– document – кореневий елемент основного документа;
– body – тіло. Може містити кілька абзаців. Також може містити властивості розділу, зазначені в sectPr елементі;
– p – параграф (рис.2.3). Може містити один або кілька запусків (run). Також може містити властивості абзацу, зазначені в pPr елементі, який, в свою чергу, може містити властивості run за замовчуванням зазначені в rPr елементі;
– r –run. Може містити кілька типів запущеного вмісту, насамперед текстові діапазони. Також може містити властивості (rPr). Run - це фундаментальна концепція OpenXML. Він є суміжним фрагментом тексту з однаковими властивостями; run не містить додаткової розмітки тексту. У цьому відношенні, OpenXML суттєво відрізняється від форматів, що дозволяють довільно вкладати властивості, такі як HTML;
– t – текстовий діапазон. Містить довільну кількість тексту без форматування, розривів рядків, таблиць, графіки або інших нетекстових матеріалів. Форматування тексту успадковується від властивостей запуску та властивостей абзацу.

```
<w:p>
   <w:pPr>
       <w:pStyle w:val="Heading1"/>
   </w:pPr>
   <w:r>
       <w:t>My heading 1</w:t>
   </w:r>
</w:p>
```

Рисунок 2.3 – простий приклад структури параграфу у WordprocessingML

Для параграфа з рисунку 2.3 задано стиль «Heading1». Параметри для цього стилю задаються окремо у файлі «styles.xml» [9].

2.2 Основні проблеми аналізу форматування docx файлів

Провівши аналіз структури стандарту OpenXML та його реалізації для документів Microsoft Word (WordprocessingML) можна зробити важливі висновки, щодо особливостей обробки формату docx з метою виявлення невідповідності до заданих правил оформлення.

У першу чергу слід звернути увагу на те, що WordprocessingML являє собою послідовність параграфів, таблиць та фігур. Розділення на сторінки ніяк не задано в описі документа. Відомий лише розмір сторінки, а відображення документу, розбитого по сторінках залишається на плечах додатку, що читає документ. Цей функціонал, що його використовую Microsoft Word відсутній у відкритому доступі і тому неможливо відтворити розбивку документа по сторінках, працюючи зі структурою WordprocessingML.

Від описаного вище обмеження насправді залежить досить багато правил форматування, як то кількість сторінок у документі, кількість тексту на сторінці, відповідність розміщення тексту на сторінках та інформації, вказаній у змісті документа тощо. Це обмеження є концептуальною особливістю стандарту, що стає очевидно з аналізу, проведеного у розділі 2.1.2. Для того, щоб обійти це обмеження засобів OpenXML вже не достатньо. Методи, що можуть бути використані для вирішення цієї проблеми описані більш детально у розділі 2.3 цієї роботи. Також на рисунку 2.4 зображена різниця між структурою документа у XML та її відображення у Microsoft Word.

Також важливо звернути увагу на зберігання інформації про заголовки та колонтитули. Особливість роботи з ними логічно випливають з відсутністю структуризації документа по сторінках. У зв’язку з цим, дані, що можна отримати з

XML репрезентації документа не мають інформації про кількість колонтитулів, їх порядок, та зміст кожного з них. Колонтитули задаються декларативно для кожної секції документу, як загальні стилі чи налаштування документа. Це значно ускладнює валідацію змісту колонтитулів на сторінках документу.

```
<w:r>
    <w:rPr>
        <w:lang w:val="uk-UA" />
    </w:rPr>
    <w:t xml:space="preserve">репрезентації документа не мають інформації про кількість
    колонтитулів, їх порядок, та зміст кожного з них. Колонтитули задаються
    декларативно для кожної секції документу, як загальні стилі чи налаштування
    документа. Це значно ускладнює </w:t>
</w:r>
<w:proofErr w:type="spellStart" />
<w:r>
    <w:rPr>
        <w:lang w:val="uk-UA" />
    </w:rPr>
    <w:t>валідацію</w:t>
</w:r>
<w:proofErr w:type="spellEnd" />
<w:r>
    <w:rPr>
        <w:lang w:val="uk-UA" />
    </w:rPr>
    <w:t xml:space="preserve"> змісту колонтитулів на сторінках документу.</w:t>
</w:r>
```

структуризації документа по сторінках. У зв'язку з цим, дані, що можна отримати з XML репрезентації документа не мають інформації про кількість колонтитулів, їх порядок, та зміст кожного з них. Колонтитули задаються декларативно для кожної

22

секції документу, як загальні стилі чи налаштування документа. Це значно ускладнює валідацію змісту колонтитулів на сторінках документу.

Рисунок 2.4 – Різниця між структурним зберіганням та відображенням документа стосовно розділення документа на сторінки

Не меншою складністю є велику кількість рівнів задання стилізації для елементів документа. Починаючи від специфічних стилів елемента і закінчуючи стилями секцій та усього документа в цілому. Та з вирішенням цієї проблеми допомагає глибокий аналіз специфікації WordprocessingML та OpenXML у цілому для побудови ієрархічних моделей прийняття рішень щодо необхідної стилізації та її перевірки.

Як видно з проведеного дослідження та аналізу проблем стандарту можна зробити логічний висновок щодо відсутності додатків для вирішення проблеми перевірки відповідності документа правилам оформлення. Структурні особливості формату не надають можливості для реалізації перевірки найбільш істотних параметрів оформлення документа. Та за результатами подальшого дослідження було виявлено методи, що можуть вирішити проблему обмежень стандарту. Ці методи більш детально описані у наступному розділі.

2.3 Аналіз методів перевірки документу до правил оформлення

Метою аналізу було знайти методи, що дозволять обійти обмеження, що були виявлені у форматі docx та провести повний аналіз відповідності документа до заданих правил форматування.

З найбільш істотних виявлених проблем було виділено велику кількість рівнів задання стилізації для елементів документа що обмежувало можливості аналізу, неможливість розбиття документу по сторінках через особливості структури WordprocessingML та декларативне задання колонтитулів, що ускладнює валідацію змісту колонтитулів на сторінках документу.

Для вирішення першої проблеми зі складною ієрархічною структурою стилізації було проведено ретельний аналіз існуючих додатків, що дозволяють

побудувати спрощену об’єктну модель та денормалізувати ієрархічну структуру до біль плоскої, що дозволило б реалізувати механізм перевірки правильності шрифтів, форматування тексту та стилів заголовків.

Було проаналізовано три найбільш популярні бібліотеки для роботи зі структурою WordprocessingML: Aspose.Words, OpenXML SDK, Spire.Doc.

OpenXML SDK побудований на API System.IO.Packaging і забезпечує строго типізовані класи для маніпулювання документами, які відповідають специфікаціям файлів Office Open XML [10]. Специфікація форматів файлів Office Open XML – це відкритий міжнародний стандарт ECMA-376, друге видання та ISO / IEC 29500. Ця бібліотека не додає ніяких можливостей для роботи з Office Open XML та вона переносить усю його структуру на об’єктну модель, якою можна легко маніпулювати у .NET застосунках. Тому доречно буде використовувати OpenXML SDK як базу для реалізації додатку для перевірки документа на відповідність правилам оформлення. Окремою перевагою є те, що бібліотека розповсюджується за ліцензією MIT і може бути використана безкоштовно.

Aspose.Words та Spire.Doc працюють з OpenXML додають додаткові можливості. А саме – спрощують роботу з текстом та надають методи для автоматичного виявлення стилів параграфа з їх ієрархічної структури. Після проведення аналізу документації та роботи з демонстраційними версіями цих бібліотек було з’ясовано, що Aspose.Words краще працює зі створенням нових документів, редагуванням та конвертацією документів у той час, як Spire.Doc надає більш потужні інструменти для роботи зі структурою документа, стилями та їх взаємозв’язком.

Саме тому методами Spire.Doc можна спростити роботу ієрархічною структурою стилів. Спрощена модель визначення стилів документа наведена на рисунку 2.5.

Spire.Doc для .NET - це професійна бібліотека Word .NET, спеціально розроблена для розробників для створення, читання, запису, перетворення та друку

файлів документів Word з будь-якої платформи .NET із швидкою та якісною продуктивністю. Як незалежний API .NET Word, Spire.Doc для .NET не потрібно встановлювати Microsoft Word ні на розробці, ні на цільових системах. Spire.Doc для .NET - це перевірений надійний MS Word API для .NET, який дозволяє виконувати багато завдань з обробки документів Word. Він підтримує C #, VB.NET, ASP.NET і ASP.NET MVC.

Рисунок 2.5 – Спрощена модель визначення стилів тексту.

Розглядаючи більш істотні проблеми – відсутність розбиття на сторінки та декларативні колонтитули, що були досліджені раніше, було зроблено наступні висновки. По-перше, отримати необхідну інформацію з формату WordprocessingML не є можливим і для вирішення цієї проблеми необхідно звернути увагу на інші формати документів, структура яких дозволяє отримати необхідну інформацію. По-друге, необхідно розробити механізм для поєднання аналізу одного документа, збереженого у різних форматах. Механізм поєднання не складно розробити засобами .NET з використанням інтерфейсів та спільних бібліотек [11].

Для вибору формату документа, що містив би інформацію про розбиття на сторінки та зміст колонтитулів необхідно було дослідити можливості точної конвертації документів docx до форматів, що таку інформацію містять. Найбільш

привабливим форматом виявився PDF. PDF є відкритим стандартом, який підтримує Міжнародна організація зі стандартизації (ISO) [12] [13].

Таким чином, методом вирішення найбільших проблем було виявлено конвертацію до PDF та подальшу його обробку паралельно з docx для аналізу сторінок документа та колонтитулів з нумерацією.

3 ОПИС ПРОГРАМНОЇ СИСТЕМИ, ЩО РЕАЛІЗУЄ ДОСЛІДЖЕНІ МЕТОДИ

3.1 Функціональні вимоги та обмеження системи

На основі ДСТУ 3008-2015 «Звіти у сфері науки і техніки» було сформовано вимоги для перевірки форматування у демонстраційній версії системи, що охоплює найбільшу кількість можливих правил оформлення і є основою для подальшого вже механічного її розширення та адаптації цих правил. Детальну інформацію щодо функціональних вимог до системи надано у додатку А «Специфікація вимог до програмного забезпечення»

Для коректного функціонування системи, що охоплює усі заданні функціональні вимоги, є необхідним ввести деякі обмеження. Найбільш істотними є обмеження, що накладаються на вхідних документ. Детально усі обмеження описано у таблиці 3.1.

Таблиця 3.1 – Обмеження щодо формати вхідного документа

Вимога	Функціонал, на який впливатиме недотримання вимоги	Рівень впливу на результуючу якість аналізу документа
Документ має бути у форматі docx.	Робота системи протестована з форматом docx. Можлива робота з деякими версіями doc.	Критичний
Кожна секція документа (наприклад, титульний аркуш, реферат або пункт основної частини) має завершуватися розривом сторінки.	Перевірка розміщення та форматування секцій документа.	Високий
Автоматичний функціонал має бути використано для оформлення змісту.	Перевірка форматування змісту та його відповідності реальному змісту документа.	Середній

Кінець таблиці 3.1

Вимога	Функціонал, на який впливатиме недотримання вимоги	Рівень впливу на результуючу якість аналізу документа
Стиль тексту «Заголовок 1» (Heading 1) має бути використано для оформлення заголовків документа	Перевірка форматування заголовків та їх розміщення у документі	Середній
Стиль тексту «Заголовок 2» (Heading 2) має бути використано для оформлення підзаголовків документа	Перевірка форматування підзаголовків та їх розміщення у документі	Середній
Посилання на малюнки у тексті мають відповідати одному з форматів: - (рис. 1) - (рисунок 1) - (див. рис. 1) - ... на рисунку 1	Перевірка посилань на малюнки у тексті	Низький
Усі формули мають бути додані до документу за допомогою стандартних інструментів Microsoft Word (Insert Equation)	Перевірка оформлення формул	Низький
Відступи параграфу (лівий, правий) мають бути задані як 0	Перевірка відступів (може бути налаштована на використання значення відмінного від 0)	Низький
Інтервали параграфу (до, після) мають бути задані як 0	Перевірка інтервалів (може бути налаштована на використання значення відмінного від 0)	Низький

Задані обмеження мають загалом рекомендаційний характер та не призводять до падіння, чи неправильного функціонування системи. Та необхідно зазначити, що вони можуть деяким чином впливати на якість отриманих результатів у наслідок

використання обраних методів аналізу документів. Детально вплив описано у таблиці 3.1.

3.2 Архітектура та дизайн компонентів системи

Функціональна частина системи являє собою набір бібліотек, написаних з використанням фреймворку .NET Core. Бібліотеки розроблено для максимального спрощення використання їх у різних типах систем. Наразі використовуються консольний додаток для тестування, а також клієнт-серверна архітектура для інтеграції системи [14].

Систему розгорнуто у хмарному середовищі Microsoft Azure. Для забезпечення роботи системи на даному етапі, використовуються наступні сервіси Azure:

- Web Apps – використовуються для розгортання серверної та клієнтської частин системи, забезпечують можливості горизонтального та вертикального масштабування, налаштування політики CORS, захисту від найбільш поширених загроз для серверних застосунків (DDOS, XSS, SQL Injections, тощо).
- Application Insights – використовується для моніторингу загального стану системи, реалізовано моніторинг та нотифікації щодо навантаження системи, виключних ситуацій, використання пам’яті та процесорного часу.
- Log Analytics – використовується для проведення аналізу причин виникнення виключних ситуацій, тенденцій системи та типової поведінки користувачів.

Для системи було обрано клієнт-серверну архітектуру комп'ютерної мережі, в якій декілька клієнтів (віддалені процесори) запитують та отримують послугу від централізованого сервера [15]. Клієнтські комп’ютери надають інтерфейс, що дозволяє користувачеві комп'ютера запитувати послуги сервера та відображати результати, які сервер повертає. Сервери чекають надходження запитів від клієнтів, а потім відповідають на них [16].

Детально інформацію щодо розгортання системи та основних фізичних компонентів, що приймають участь у функціонуванні системи зображено на діаграмі розгортання на рисунку 3.1.

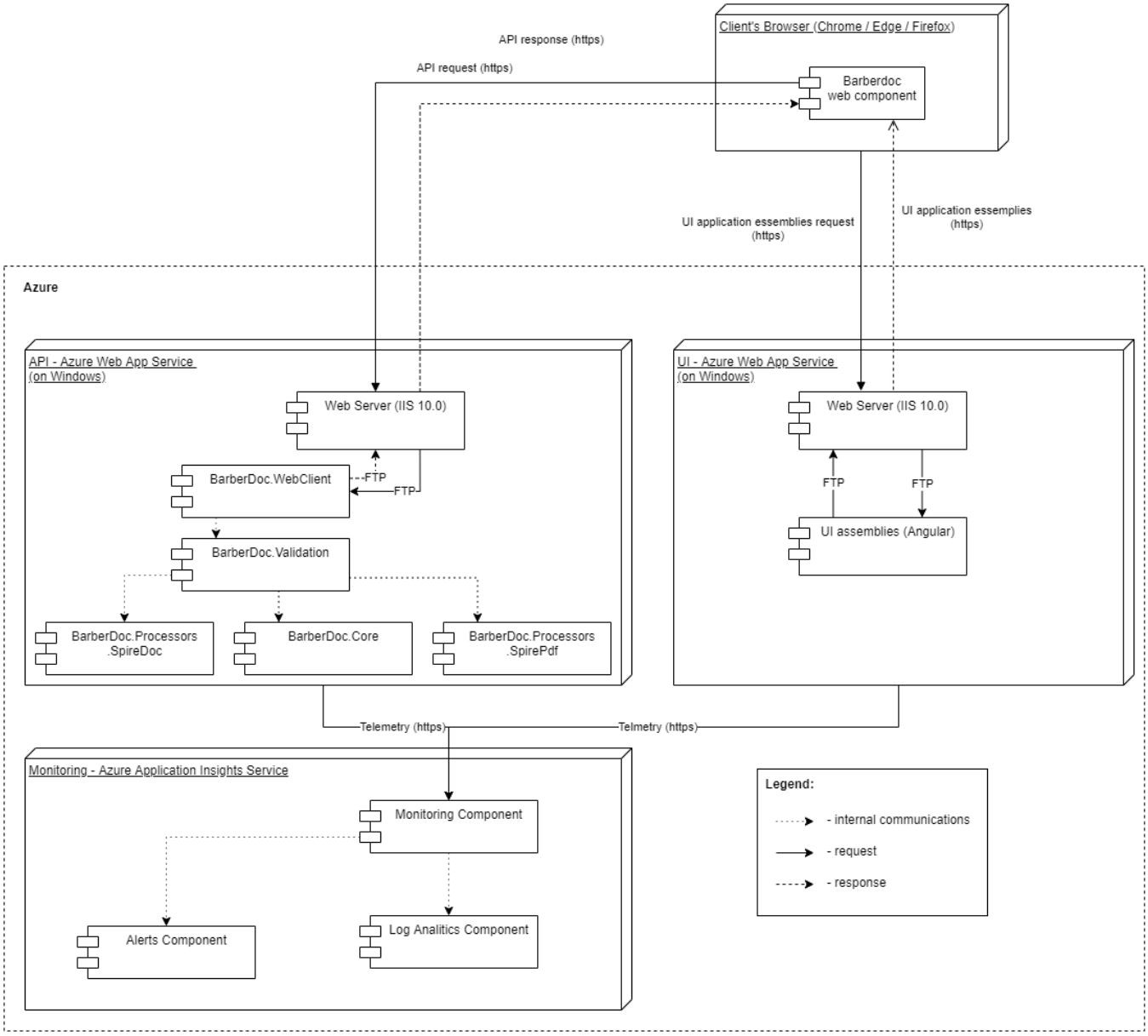

Рисунок 3.1 – Діаграма розгортання системи

Взаємодію з системою та типові шляхи застосування системи зображено на діаграмі варіантів використання на рисунку 3.2. У процесі приймають участь два актори (студент та експерт нормоконтролю) та розроблена система. На рисунку 3.2 не зображено процес аутентифікації та авторизації акторів у системи, бо ця поведінка є типовою для подібних систем [17].

Студент має можливість використовувати веб-клієнт системи для попередньої перевірки своєї дипломної роботи. Експерт нормоконтролю використовує той самий веб-клієнт для отримання результатів перевірки та аналізу результатів. Веб-клієнт передає завантажений документ серверу валідації. Сервер валідації використовує процесори документа для перетворення документа на модель для аналізу. Валідатори форматування перевіряють отриману модель на помилки форматування та повертають клієнту список знайдених помилок форматування з їх описанням.

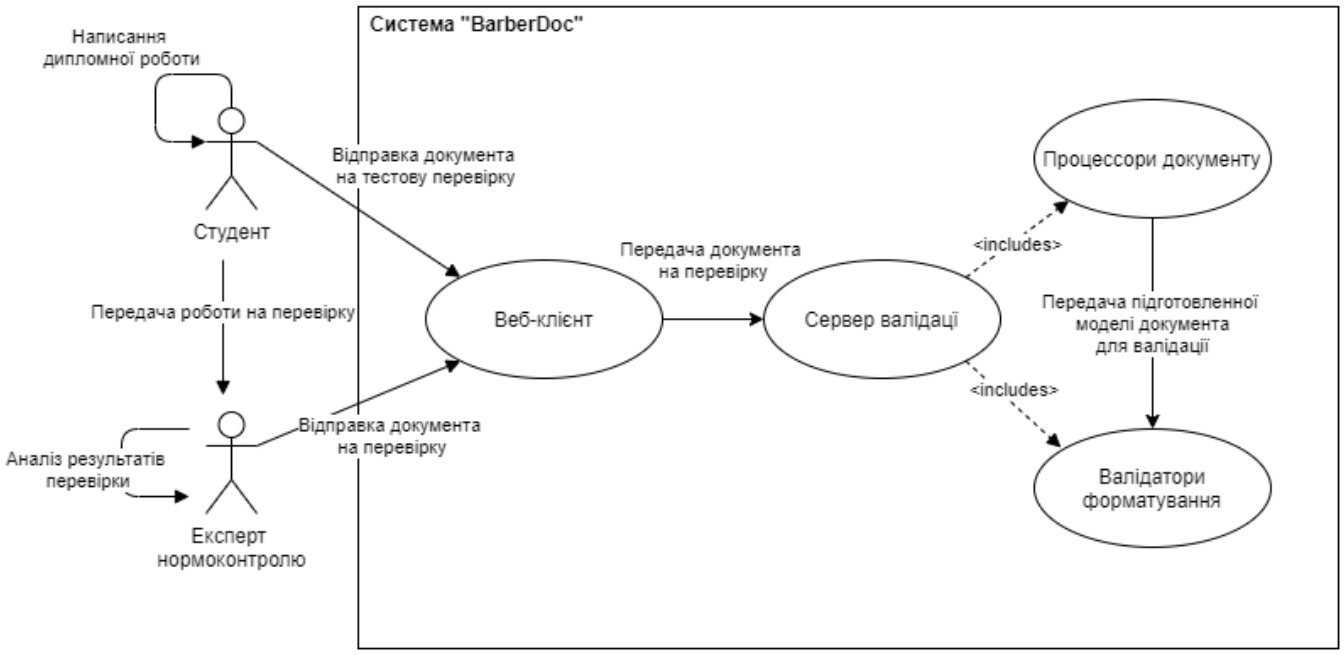

Рисунок 3.2 – Діаграма варіантів використання

Детальний дизайн бібліотеки валідаторів форматування зображено на діаграмі класів на рисунку 3.3.

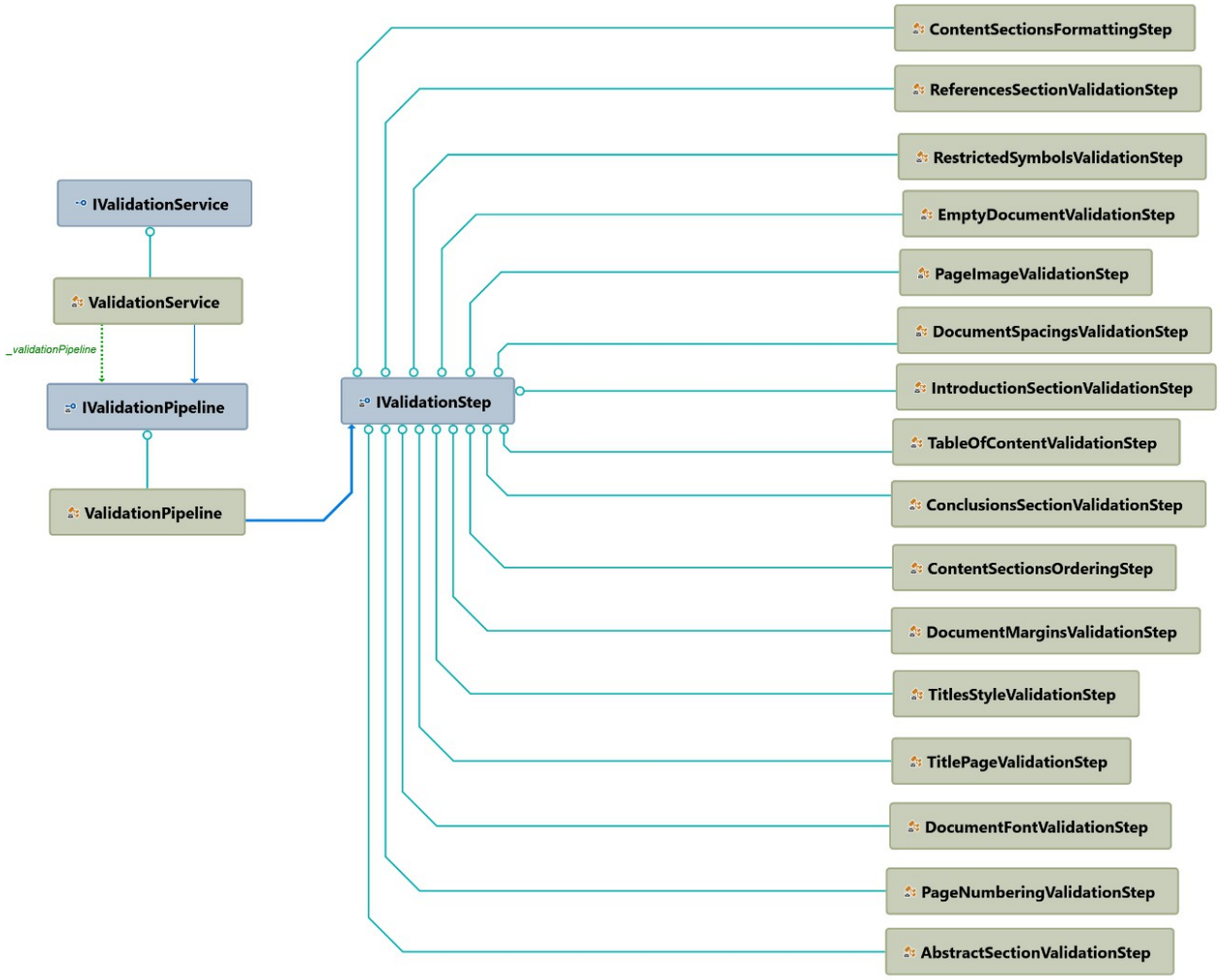

Рисунок 3.3 – Діаграма класів бібліотеки валідаторів форматування

Важливим аспектом є виділення окремих валідаторів, що незалежно працюють з моделлю документа та можуть буду додані до системи без жодних змін у ній. Такий підхід робить систему модульною та дозволяє змінювати її структуру та поведінку в залежності від конфігурації правил форматування [18] [19]. Наприклад, якщо користувач вирішить відключити перевірку форматування таблиці, то модулі, що відповідають за таку валідацію не будуть використовуватися у роботі системи.

Це рішення дозволяє легко адаптувати систему до роботи з різними наборами правил форматування, а також додавати нові [20]. Апробацію даного методу наведено у додатку Б.

3.3 Реалізація окремих компонентів перевірки форматування

Далі приведено декілька найбільш показових прикладів валідаторів форматування, що реалізують досліджені методи обробки документа з метою перевірки відповідності оформлення до певних правил на прикладі перевірки атестаційних дипломних робіт та автоматизації процесу нормоконтролю.

Для демонстрації роботи з docx документом є доречним розглянути перевірку форматування секцій документа (вступ, висновки, реферат, аналіз предметної галузі, тощо). З функціональних вимог до системи можна виділити наступні, що стосуються перевірки форматування змістових секцій документа:

- перевірка розміщення секції (відокремлення розривами сторінки, назва наступної секції);
- перевірка стилів заголовку (текст розміщено в центрі, великими літерами);
- заголовок відокремлений від основного тексту двома пустими рядками;
- шрифт та абзацні відступи відповідають заданим.

Першочерговим завданням є виділення секції з повної моделі документа. Для цього використовується ім’я секції та розриви сторінок. Для пошуку імен нетипових секцій, як то «Висновки», «Вступ», «Реферат», використовується наступний алгоритм:

а) отримати усі параграфи документа між вже відомими секціями – «Вступ» та «Висновки»;

б) пошук усіх розривів сторінок в отриманому масиві параграфів;

в) для кожного отриманого елемента масиву (параграфа з розривом сторінки n) виконати наступне:

   1) перевірити, чи є текст у параграфі n після елементу розриву сторінки;
   2) якщо текст знайдено – це і є заголовок секції;
   3) якщо текст не знайдено – взяти текст із наступного параграфа у масиві;
   4) додати отриманий текст до результуючого масиву заголовків;

г) з отриманого масиву заголовків вилучити «Висновки».

На рисунку 3.4 надано програмну реалізацію алгоритму.

```
// Ініціалізація масиву заголовкив
var titles = new List<string>();

// Задання першої секції "ВСТУП"
var sectionTitle = introductionSectionTitle; // "ВСТУП"

// Цикл по секціях документу до моменту отримання секції з заголовком "ВИСНОВКИ"
do
{
    // Отримання останього параграфу поточної секції
    if (GetSectionItems(document, sectionTitle).LastOrDefault() is Paragraph lastParagraph)
    {
        // Отримання назви наступної секції с останього або наступного параграфа
        sectionTitle = GetNextSectionTitle(lastParagraph);

        // Збереження отриманої назви секції
        titles.Add(sectionTitle);
    }
    else
    {
        // Припинення циклу, якщо досягнуто кінця документа
        break;
    }
}
while (!EqualsIgnoringCase(sectionTitle, conclusionsSectionTitle)); // "ВИСНОВКИ"

// Вилучення секції "ВИСНОВКИ" та повернення результату роботи алгоритму
return titles.Where(title => !EqualsIgnoringCase(title, conclusionsSectionTitle));
```

Рисунок 3.4 – Реалізація алгоритму пошуку заголовків секцій документа

Отриманий список заголовків формує чергу на валідацію кожної окремої секції. Для реалізації функціональних вимог необхідно виділити та проаналізувати три складові секції документа: перший та останній параграфи і масив елементів секції між ними.

Перший параграф використовується для перевірки стилю заголовку, відступів після заголовку та відокремлення секції від попередньої. Реалізація перевірки стилів заголовку складається з кількох кроків, що зображенні на рисунку 3.5.

```
// Метод перевірки вирівнюрання тексту. Вирівнювання задається у налаштуваннях та може бути задане як:
// По лівому краю / У центрі / По правому краю / Розтягнути
1 reference | Serhii Iordanov, 12 days ago | 1 author, 2 changes | 0 exceptions
private void ValidateAlignment(ParagraphStyle style, ICollection<ValidationError> errors)
{
    if (Enum.TryParse(_validationSettings.TextAlignment, out HorizontalAlignment alignment))
    {
        if (style.ParagraphFormat.HorizontalAlignment != alignment)
        {
            errors.Add(ValidationError.CreateError(GetType(),
                $"Style {style.Name} must have '{alignment.ToString()}' alignment."));
        }
    }
    else... // Повернути помилку налаштування (значення вирювнювання задано невірно)
}

// Перевірка на використання списку у якості заголовку
1 reference | Serhii Iordanov, 20 days ago | 1 author, 1 change | 0 exceptions
private void ValidateListFormatting(ParagraphStyle style, ICollection<ValidationError> errors)
{
    if (_validationSettings.UsesListFormatting && style.ListFormat.ListType == ListType.NoList)
    {
        errors.Add(ValidationError.CreateError(GetType(), $"Style '{style.Name}' must use list formatting."));
    }
    else if (!_validationSettings.UsesListFormatting && style.ListFormat.ListType != ListType.NoList)
    {
        errors.Add(ValidationError.CreateError(GetType(), $"Style '{style.Name}' must not use list formatting."));
    }
}

// Перевірка, чи записаний заголовок великими літерами
1 reference | Serhii Iordanov, 20 days ago | 1 author, 1 change | 0 exceptions
private void ValidateCaps(ParagraphStyle style, ICollection<ValidationError> errors)
{
    if (_validationSettings.AllCaps != style.CharacterFormat.AllCaps)
    {
        var message = _validationSettings.AllCaps ? $"Style '{style.Name}' must set text to be in uppercase."
            : $"Style '{style.Name}' must not set text to be in uppercase.";

        errors.Add(ValidationError.CreateError(GetType(), message));
    }
}
```

Рисунок 3.5 – Реалізація перевірки стилів заголовків секцій документа

Останній параграф використовується для перевірки розриву сторінки, а також заголовку наступної секції для збереження заданої послідовності. Складові елементи секції необхідні для валідації шрифтів, абзацних відступів та інтервалів. На рисунку 3.6 наведено реалізацію алгоритму перевірки відповідності шрифту кожного параграфа секції до встановлених значень.

```
// Задання масиву помилок
var errors = new List<ValidationError>();

// Перевірка відповідності назви шрифта (наприклад: Times New Roman)
var isValidFontFamily = requiredFont.FontFamily == paragraph.BreakCharacterFormat.FontName;
// Перевірка розміру шрифта (використовується допустима похибка рівна 0.00001)
var isValidFontSize = Math.Abs(paragraph.BreakCharacterFormat.FontSize - requiredFont.Size) < SizeInfelicity;
// Перевірка кольору шрифта
var isValidFontColor = ColorRgb.ConvertToRgb(paragraph.BreakCharacterFormat.TextColor).Equals(requiredFont.ColorRgb);

// Висновок щодо правильності шрифта у параграфі
var isValidFont = isValidFontFamily && isValidFontSize && isValidFontColor;

// Формування помилки у разі неправильності шрифта
if (paragraph.Text.Length > 0 && !isValidFont)
{
    var errorType = isTableParagraph ? ErrorType.Warning : ErrorType.Error;

    var message =
        $"Invalid font: {paragraph.BreakCharacterFormat.FontName}" +
        $" {paragraph.BreakCharacterFormat.FontSize}" +
        $" {ColorRgb.ConvertToRgb(paragraph.BreakCharacterFormat.TextColor)}." +
        paragraph.GetTextSubstring(_settings.ErrorMessageNumberOfChars, isTableParagraph);

    errors.Add(new ValidationError(errorType, typeof(TContext), message));
}
```

Рисунок 3.6 – Перевірка відповідності шрифту параграфа до встановлених значень

Після виконання усіх описаних алгоритмів для кожної секції необхідно скомбінувати та встановити пріоритети отриманим помилкам форматування. Також важливим є те, що для кожного алгоритму використовується окремий компонент

валідації, що надає змогу додавати, видаляти та замінювати будь які частини перевірки іншими.

3.4 Інтерфейс користувача та використання системи

Для впровадження системи було створено веб-застосунок з використанням технологій ASP.NET Core та Angular. Angular було використано для створення клієнтської частини застосунку. Наразі застосунок надає можливість завантажити роботу у форматі docx. Також є можливість завантажити документ у форматі PDF для комбінованої та біль детальної перевірки форматування методами, що описані раніше у цій роботі. На рисунку 3.7 приведено вигляд інтерфейсу користувача.

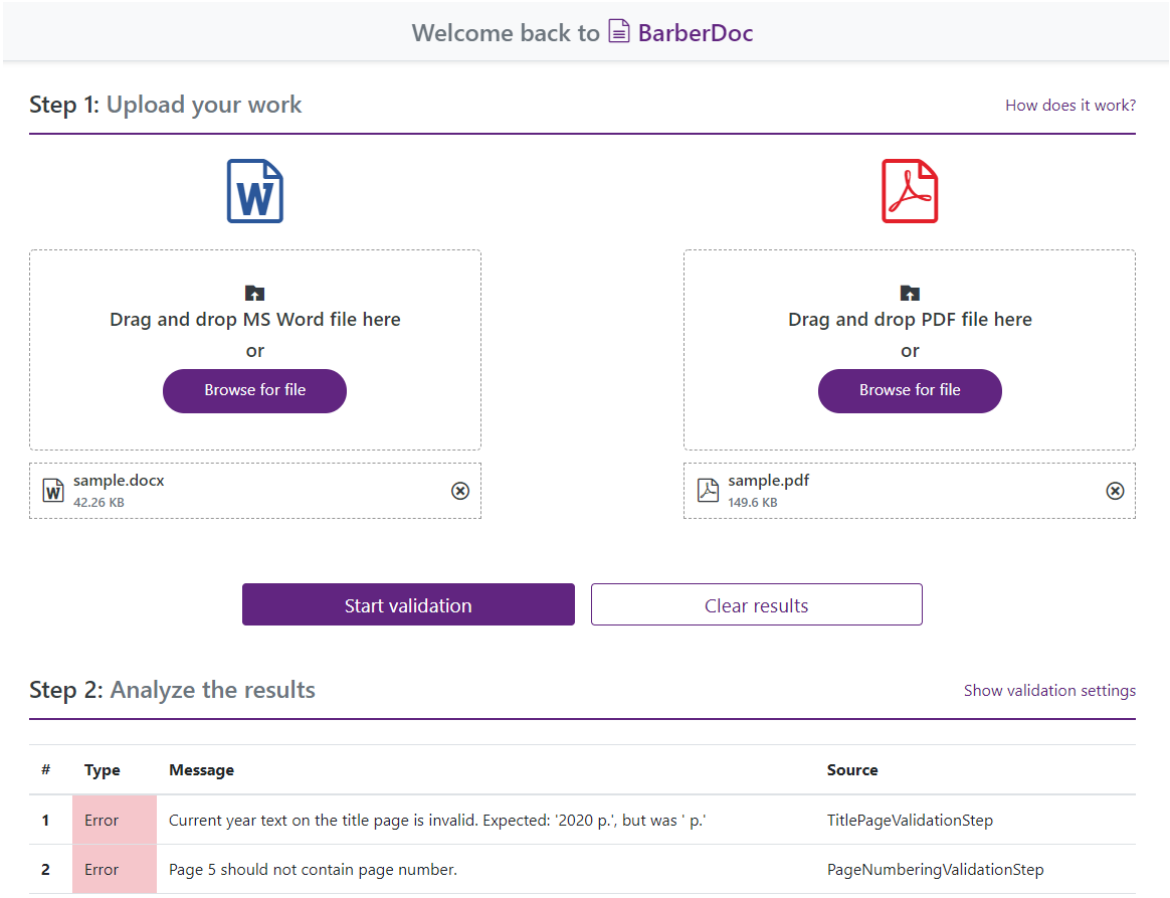

Рисунок 3.7 – Інтерфейс користувача застосунку BarberDoc

Інтерфейс побудовано з урахуванням основних практик створення інтерфейсів користувача та принципів UI/UX. Серед основних з використаних принципів слід виділити наступні:

- принцип KISS. Інтерфейс повинен бути простий і зрозумілий, завдання повинні вирішуватися мінімальним числом дій, все повинно бути зрозуміло і очевидно;
- співвідношення сигнал / шум. У кожному інтерфейсі є важливі елементи (сигнали) і незначні або навіть безглузді для певної частини системи (шум), природно, потрібно концентруватися на сигналах і уникати шуму;
- звичні елементи управління. У будь-якому сучасному інтерфейсі є багато елементів управління, буде краще використовувати звичні елементи і візуальні образи;
- принцип угруповання. Інформацію на сторінці бажано створити декілька логічних блоки (групи), так користувачеві легше орієнтуватися.

Окрім безпосередньо завантаження своєї роботи та перевірки форматування інтерфейс надає користувачеві декілька додаткових можливостей, що спрощують роботи з системою та роблять її більш інтуїтивною:

- перегляд результатів проведеної перевірки форматування у вигляді таблиці, відсортованої за пріоритетами помилок;
- можливість очистити таблицю помилок;
- перегляд заданих налаштувань перевірки форматування, що надалі буде перетворено у інтерфейс конфігурування цих налаштувань;
- видалення обраного файлу (docx чи PDF);
- перегляд сторінки з поясненнями щодо роботи застосунку та описом необхідних дій для перевірки документа.

4 ОПИС МОЖЛИВОСТЕЙ ВИКОРИСТАННЯ ОТРИМАНИХ РЕЗУЛЬТАТІВ

4.1 Використання системи автоматизації нормоконтролю

Розроблена система для автоматизації процесу нормоконтролю значно спрощує процес перевірки атестаційних робіт студентів першого та другого рівнів освіти на відповідність вимогам форматування пояснювальної записки. Завдяки гнучким архітектурним рішенням, систему легко розширити новими правилами перевірки та змінити існуючі, що дозволяє своєчасно налаштовувати систему для роботи зі змінами у вимогах форматування.

Не зважаючи на те, що першочергово систему було створено для автоматизації процесу нормоконтролю у Харківському національному університеті радіоелектроніки, вже на даному етапі реалізації систему легко адаптувати для різних вищих навчальних закладів, враховуючи особливості форматування атестаційних робіт у них.

Важливим також є питання ліцензування програмного забезпечення, що використовується системою. Для повноцінного використання усіх можливостей системи необхідним є використання ліцензованої версії програмного забезпечення Spire.Doc для .NETs.

Реалізуючи у собі досліджені методи роботи з електронними документами у форматах docx та PDF система є відправною точкою для побудови більш універсальних рішень, що можуть бути використані будь-яким підприємством чи організацією для автоматизації частини електронного документообігу, що пов’язана з перевіркою форматування документа.

4.2 Використання досліджених методів та розширення системи перевірки форматування

За результатами дослідження методів роботи з документами формату docx з метою перевірки відповідності документа правилам форматування, а також реалізації демонстраційної системи автоматизації нормоконтролю, що побудована на досліджених методах можна побачити подальші перспективи використання досліджених методів.

За допомогою досліджених методів отримання інформації о розміщенні тексту по сторінках зі структури docx документа, а також конвертації цієї структури у спрощенні програмні моделі з’являється можливість створення уніфікованих програмних засобів для перевірки документа на відповідність правилам форматування. Наразі ринок програмного забезпечення не має аналогів подібним системам, у той час як потреба в повній автоматизації електронного документообігу спостерігається вже тривалий час.

Згідно з результатами аналізу предметної галузі було виявлено, що однією з найважливіших задач у процесі документообігу в будь-якій організації є мінімізація кількості повернень документа на стадіях його опрацювання. Найбільше таких повернень підпадає на стадії перевірки змісту та форматування. Система автоматизації перевірки форматування значно зменшує кількість повернень, адже документ може бути перевірено автоматично без участі працівників та контролерів.

Створення подібної системи потребує чималих людських та технологічних ресурсів для обробки усіх можливих варіантів змін правил форматування. Та базуючись на досліджених методах ця задача стає більш рутинною працею і може бути виконана за прогнозований проміжок часу.

ВИСНОВКИ

Метою роботи був аналіз методів роботи з електронними документами у форматі docx та можливості комбінування їх з методами обробки інших типів документів для перевірки форматування (оформлення) документу.

Результатами проведених досліджень та програмної реалізації стали перевірені методи аналізу docx документів за допомогою комбінування різних форматів та міграції складних ієрархічних моделей до спрощених об’єктних репрезентацій. Створено додаток-прототип для автоматизації процесу нормоконтролю атестаційних робіт першого (бакалаврського) та другого (магістерського) рівнів. У рамках додатку реалізовано методи, досліджені у процесі аналізу та доведена ефективність більшості з них. До найбільш успішних та виправданих методів слід віднести спрощення ієрархічних структур за допомогою бібліотеки Spire.Doc та комбінування аналізу документа у різних форматах, а саме – docx та PDF, а також конвертацію документів у відповідні формати.

Серверну частину системи побудовано з використанням язика програмування C# на платформі .NET, базуючись на крос-платформному фреймворці ASP.NET Core, що надає можливість розвернути серверну частину на таких операційних системах як Windows, Unix та MacOS. Серверну частину було розгорнуто на віртуальній машині з операційною системою Windows у хмарному середовищі Azure.

Розроблена система надає механізми для подальшого розширення та використання для автоматизації стадії перевірки оформлення документів у будь-якій галузі, що і є завданням для подальшого розвитку продукту.

В роботі проведено аналіз предметної галузі, здійснено дослідження основних методів, моделей та алгоритмів аналізу відповідності електронного документу до

вимог оформлення. В результаті аналізу виявлено декілька найбільш ефективних методів та досліджено можливість їх поєднання в процесі програмної реалізації.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. William B. Green. Introduction to Electronic Document Management Systems. Academic Press, 1993. – 250 с.
2. Michael J. D. Sutton. Document Management for the Enterprise: Principles, Techniques, and Applications. Wiley; 1 edition, 1996. – 400 с.
3. Основи документообігу [Електронний ресурс] / KtoNaNovenkogo. – режим доступу: https://ktonanovenkogo.ru/voprosy-i-otvety/dokumentooborot-chto-ehto-takoe.html – Загол. з екрану.
4. Види документообігу [Електронний ресурс] / Sekretariat – режим доступу: https://www.sekretariat.ru/article/210652-qqq-16-m9-vidy-dokumentooborota – Загол. з екрану.
5. Three Way Matching [Електронний ресурс] / Purchasecontrol – режим доступу: https://www.purchasecontrol.com/uk/blog/what-is-3-way-matching/ – Загол. з екрану.
6. ДСТУ 3008-2015 [Електронний ресурс] / KNMU – режим доступу: http://www.knmu.kharkov.ua/attachments/3659_3008-2015.PDF – Загол. з екрану.
7. ECMA (ECMA-376) [Електронний ресурс] / Ecma International – режим доступу: http://www.ecma-international.org/publications/standards/Ecma-376.htm – Загол. з екрану.
8. Wouter van Vugt. Open XML Explained. Microsoft, 2007. – 129 с.
9. How to work with docx [Електронний ресурс] / Toptal – режим доступу: https://www.toptal.com/xml/an-informal-introduction-to-docx – Загол. з екрану.
10. OpenXML SDK documentation [Електронний ресурс] / Microsoft Docs – режим доступу: https://docs.microsoft.com/en-us/office/open-xml/open-xml-sdk – Загол. з екрану.

11. A. Mendes, Z.Dudar, V. Kauk, T. Shatovska, I. Revenchuk, A. Chupryna, D. Fedasyuk, V. Yakovina, I. Lyutak. Technopreneurship in Ukraine. How to Boost Enterpreneurial Competence Development in the Ukrainian IT Industry. //- Linnaueus University Copycenter, 2016.- 160p. ISBN: 978-91-88357-68-7
12. ISO/IEC 29500-1:2016 [Електронний ресурс] – режим доступу: https://www.iso.org/standard/71691.html (дата звернення: 20.03.2020)
13. ISO/IEC 14496-22:2007 [Електронний ресурс] – режим доступу: https://www.iso.org/standard/43466.html (дата звернення: 20.03.2020)
14. Bohdan Sus, Ilona Revenchuk, Nataliia Tmienova,Vira VialkovaDevelopment of Virtual Laboratory Works for Technical and Computer Sciences .- ICIST 2019.- Vilnius, Lithuania, October 10–12, 2019.- pp 383-394
15. І.А. Ревенчук, К.В. Перцьова, О.І. Маренич. Програмна реалізація кластеризації пошукових запитів.- Біоніка інтелекту.-№.-2019.-С.7-14
16. І.А. Ревенчук. Математична модель агрегації даних в соціальних медіа.-Сб. наук. праць "Математичне та комп'ютерне моделювання. Серія: Техн. Науки" за Міжнар. - Кам'янець-Подільський.- 2017.- С. 197-203
17. Bondarenko M. F., Dudar Z. V., Revenchuk I.A. Information Systems and Technologies Used in Distance Form of Education at the University.- Informational and Communication Technologies Technologies. - 2012.- P.485-490. ISBN: 978-1-61470-050-0
18. Spire.Doc for .NET [Електронний ресурс] – режим доступу: https://www.e-iceblue.com/Introduce/word-for-net-introduce.html#.XnTuM6gzaUk (дата звернення: 20.03.2020)
19. Spire.Pdf for .NET [Електронний ресурс] – режим доступу: https://www.e-iceblue.com/Introduce/pdf-for-net-introduce.html#.XnTuiagzaUk (дата звернення: 20.03.2020)
20. Home of PDFsharp and MigraDoc Foundation [Електронний ресурс] – режим доступу: http://www.pdfsharp.net/ (дата звернення: 20.03.2020)