Міністерство освіти та науки України
Харківський національний університет радіоелектроніки

Факультет *Комп'ютерних наук*
(повна назва)
Кафедра *Системотехніки*
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

Рівень вищої освіти другий (магістерський)
(рівень вищої освіти)

Розробка та дослідження компонентів системи управління документообігом з цифровими документами.
(тема)

Виконав:
студент *2* курсу, групи *СПРм-20-1*
Стаднік М. М.
(прізвище, ініціали)
спеціальності *122 – Комп'ютерні науки*
(код і повна назва спеціальності)
Тип програми *освітньо-професійна*
(освітньо-професійна або освітньо-наукова)
Освітня програма *Системне проектування*
(повна назва освітньої програми)
Керівник *проф. каф. СТ Мінухін С.В.*
(посада, прізвище, ініціали)

Допускається до захисту
Зав. кафедри системотехніки _____ *Гребеннік І.В.*
(підпис) (прізвище, ініціали)

2021 р.

Харківський національний університет радіоелектроніки

Факультет *Комп'ютерних наук*
(повна назва)

Кафедра *Системотехніки*
(повна назва)

Рівень вищої освіти другий (магістерський)

Спеціальність *122 – Комп'ютерні науки*
(код і повна назва)

Тип програми *освітньо-професійна*
(освітньо-професійна або освітньо-наукова)

Освітня програма *Системне проектування*
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)

«_____»_____
2021р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Стадніку *Михайлу Михайловичу*
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка та дослідження компонентів системи управління документообігом з цифровими документами
затверджена наказом по університету від 08.11 2021 р. № 1516СТ
2. Термін подання студентом роботи до екзаменаційної комісії 18 грудня 2021р
3. Вихідні дані до роботи: Перелік використовуваних програмних засобів: ОС Windows 10, Visual Studio. Технічне забезпечення: комп'ютер зі встановленим Visual Studio.
4. Перелік питань, що потрібно опрацювати в роботі 4. Зміст пояснювальної записки (перелік питань, що потрібно розробити) 4.1 Вступ. 4.2 Аналіз предметної області. 4.2.1 Процес розробки веб додатків. 4.2.2 Історія про адміністративні послуги. 4.2.3 Види послуг. 4.2.4 Постановка задачі. 4.3 Дослідження існуючих методів генерації файлів. 4.3.1 Парсинг файлів 4.3.2 Захист даних від шахраїв. 4.4 Розробка модифікованого методу генерації файлу. 4.4.1 Файл конфигурації. 4.4.2 Вибір інструмента реалізації. 4.4.3 Реалізація модифікованого методу. 4.5 Висновок.

5.Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів,

комп'ютерних ілюстрацій (слайдів) 2.9 Фінальний результат алгоритму генерації файлу3.2 Згенерований файл 3.2 Пул об'єктів в сервисах 3.3 Ініціалізація IdentityServer4 3.4 Запит під час генерації файлу 3.5 Генерація файлу з використанням Aspose.Words

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

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
1	Аналіз вихідних даних та літератури за темою магістерської кваліфікаційної роботи	12.11.2021	Виконано
2	Системний аналіз предметної області	14.11.2021	Виконано
3	Постановка мети та задач дослідження	20.11.2021	Виконано
5	Розробка програмного засобу	03.12.2021	Виконано
6	Оформлення пояснювальної записки та презентації	11.12.2021	Виконано
7	Подання кваліфікаційної роботи до екзаменаційної комісії	16.12.2021	Виконано

Дата видачі завдання 08 листопада 2021 р.

Студент _____ Стаднік М.М.
(підпис) (прізвище, ініціали)

Керівник роботи _____ проф. каф. СТ Мінухін С.В.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка до магістерської кваліфікаційної роботи: 88 с., 30 рис., 4 додатки, 11 джерел інформації

ASPNET Core, КОМПОНЕНТИ ДОКУМЕНТООБІГУ, ЦИФРОВІ ДОКУМЕНТИ, ПРОЦЕДУРНА ГЕНЕРАЦІЯ ФАЙЛІВ, Aspose.Words, Aspose.Cells

Об'єкт дослідження – процеси автоматичної генерації файлів або документів як система документообігу.

Предмет дослідження – методи та засоби автоматичної генерації файлів у системі документообігу.

Мета роботи – дослідити та розробити модифікований метод автоматичної генерації файлів у системах документообігу.

Методи дослідження – аналіз існуючих компонентів документообігу методів автоматичної генерації файлів. Визначено підхід до вирішення проблеми автоматичної генерації файлів в веб додатках. Виконано порівняння існуючих алгоритмів генерації, проведено порівняльний аналіз їх показників, виділені найефективніші алгоритми. Запропоновано модифікацію алгоритмів генерації яка задовольняє поставленій задачі.

Галузь застосування – програмний засіб використовується для автоматизаціх процесів докуменообігу в веб додатках.

ABSTACT

Master’s Thesis: 88 pages, 30 figures, 4 appendices, 11 title.

ASPNET Core, DOCUMENT COMPONENTS, DIGITAL DOCUMENTS, PROCEDURAL LEVEL GENERATION, Aspose.Words, Aspose.Cells

The object of study - processes of automatic generation of files or documents as a document management system.

The subject of research - methods and means of automatic generation of files in the document management system.

The purpose of the work is investigate and develop a modified method of automatic file generation in document management systems

Research methods - analysis of existing components of document management methods of automatic file generation. An approach to solving the problem of automatic file generation in web applications is defined. The comparison of existing generation algorithms is made, the comparative analysis of their indicators is carried out, the most effective algorithms are allocated. A modification of generation algorithms that satisfies the task is proposed.

Scope - software is used to automate document management processes in web applications.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

БД – база даних;

СКБД – система керування базами даних;

ПЗ – програмне забезпечення;

SDLC – життєвий цикл розробки програмного забезпечення;

HTTP – протокол передачі гіпертексту;

SRS – специфікація вимог програмного забезпечення;

HTML – мова гіпертекстової розмітки;

IDE – інтегроване середовище розробки;

API – програмний інтерфейс програми;

DOM – об'єктна модель документа;

GUI – графічний інтерфейс користувача;

TCP – протокол керування передаванням;

IP – інтернет протокол;

RC – віддалений доступ;

RS – сервер для взаємодії клієнта із базою даних;

ВСТУП

Сьогодні не можливо увити життя без використання інтернету. Інтернет охоплює багато речей та продовжує далі стрімкий розвиток. З початку це був пристрій з передачі даних. Він міг зв’язати декілька комп’ютерів або створити локальну мережу. Далі поява протоколу World Wide Web (www) відкрило багато можливостей не тільки дослідникам та вченим, але и звичайним людям. Веб сторінки почали стрімкий розвиток. З’являлись нові технічні рішення, поліпшувалась швидкість передачі даних.

Так як інтернет став неймовірно популярний, якась частина ідентифікації людини тепер в електронному вигляді. Україна – перша країна, яка вирішила використовувати замість паперового паспорту електронний на офіційному рівні. Електронні документи дедалі більше впроваджуються в наше життя. Вже давно є електронні черги, де запис відбувається онлайн. Всі ці ідеї сприяли покращенню в різних центрах надання адміністративних послуг. Тепер будь-який громадянин може планувати свій час гнучко та зручно для вирішення його адміністративної проблеми, оформити необхідний документ онлайн без потреб стояти у величезних чергах для того, щоб отримати відповідь, яка можливо і не вирішить проблему. Поки що система з електронними паспортами лише починає розширюватись у світі. Це пов'язано з тим, що системи хоч і працюють, але інші країни поки не знають, до чого це може призвести. Наприклад, якщо дані хтось вкраде, зламає обліковий запис та інше.

Метою магістерської кваліфікаційної роботи є аналіз засобів розробки компонентів документообігу із цифровими документами у додатках, з'ясувати структуру їхньої роботи, зробити свій варіант додатку із документообігом.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис предметної області

Сфера пов'язана з програмуванням зробила великий стрибок у розвитку за останні 15 років. Інтернет став невід'ємною частиною життям усього світу, щодня гігабайти даних завантажуються у мережу. За невелику кількість років було створено багато різних сервісів, таких як доставляння їжі, різні величезні платформи для продажу якихось речей (машини, будинки і т.д.), купівля квитків на потяги або літаки та багато інших сервісів.

З появою сервісів був розвиток архітектури побудов додатків, бази даних, з'явилися платні хмарні сервери такі як Amazon, Azure і т.д. У зв'язку з розвитком усіх цих технологій фахівці почали замислюватися до якої галузі можна застосувати ці знання. Якось міністр цифрової трансформації Михайло Федоров оголосив про запуск сайту та сервісів для отримання державних послуг онлайн. Міністр сказав, що перші послуги стануть доступними вже до кінця 2019 року, і до 2022 року вже всі державні послуги можна буде отримувати онлайн [1].

Іншими словами, зробити процес електронним та автоматичним. До того ж, якщо подумати, це скоротить корупцію, оскільки взаємодія людини із держслужбовцями буде мінімальною. Головне завдання проекту – створення єдиного ресурсу для отримання всіх державних послуг на смартфоні.

Програмні фахівці вирішили об'єднати державу з громадянином та надавати різний спектр послуг:

- Зберігання всієї необхідної інформації для власної ідентифікації: електронний паспорт, права водія, студентський квиток і т.д;
- Можливість використовувати веб-портал для отримання державних послуг: оформлення ФОП, підпис документів онлайн, реєстрація прописки для дитини і т.д;
- Створення та консультація з різних договорів, документів для освоєння якогось бізнесу;

Незважаючи на те, що в Україні популярно лаяти все державне, у «Дія» прогрес є. Поступово, але електронні документи входять у вжиток, їх вже приймають в аеропортах (що трапилося не відразу), і на пошті (теж шлях не з легких), автомобілісти можуть залишити вдома не тільки права та техпаспорт, а й страховий поліс.

1.1.1 Основні види адміністративних послуг

Оскільки послуг справді може бути дуже багато, тому що перенести всі державні послуги, пов'язані з громадянином важко, можна перерахувати основні.

Однією з найважливіших послуг є можливість заповнення різних документів у центрі адміністративних послуг (ЦНАП). Завдяки тому, що ЦНАП буде перенесено до веб-порталу, багато документів, які потрібні для ведення бізнесу або реєстрація ФОП більше не треба оформляти у паперовому вигляді, оскільки це можна буде легко оформити онлайн.

Друга послуга дозволяє користувачам мати такі чутливі дані, як паспорт, права водія, довідки про щеплення і т.д. Ця послуга має сильний зв'язок з оформленням документів, оскільки не потрібно більше переносити паспортні дані у паперовому вигляді до документів. Дані будуть підвантажуватись з бази даних, і відразу заповнювати необхідні поля у формі, що дозволить більше часу приділити на якісь важливіші моменти.

Однак існують проблеми, пов'язані з перенесенням даних у цифрофой вигляд. Це связанно з тим, що дані у друкованому вигляді хаотично розкидані у всьому ЦНАП. Причина цього в самих громадянах, коли вони заповнювали документи, могли помилитися під час заповнення форми, і тепер потрібно вручну перебирати інформацію стосовно громадян. Це найбільш тимчасова витрата при розробці програми, так як у будь-якої країни багато користувачів, і для кожного потрібно підвантажувати якусь унікальну інформацію, що дозволить ідентифікувати громадянина.

1.1.2 Держава у смартфоні

Завдяки тому, що смартфони популярні в сучасному суспільстві, за допомогою додатків можна зберігати різні дані щодо особистої чи публічної інформації. Тепер можна ідентифікувати людину, пред'явивши QR код для сканування. За допомогою технології QR коду, можна зберігати різну інформацію таку як інформацію про студента або права водія.

На жаль, незважаючи на зручність технології в ній є вагомі мінуси. Для того щоб дані були актуальні, потрібна постійна підтримка бездротової мережі (Wi-Fi), отже в пунктах, що не мають зв'язку, дані будуть не актуальні.

Другий недолік - це підтримка актуальних даних, тому дані зберігаються локально і оновлюється залежно від статусу мережі. До цього рішення дійшли розробники програми "Дія", через це в цьому додатку з'явилася критична вразливість, адже в сучасному світі зламати смартфон не складно, а взяти дані з локального сховища ще простіше. Для вирішення цієї проблеми потрібно зберігати дані тільки на сервері, у разі недоступності мережі користувач не зможе увійти до програми та отримати доступ до необхідних даних. Тоді залишиться лише один недолік – це бездротова мережа.

1.1.3 Можливість інтеграції

Так як додаток працює з унікальними даними громадянина потрібно передбачити обов'язкову інтеграцію з іншими сервісами, які запитують у паперовому вигляді такі дані як паспорт, права водія і т.д.

Інтеграція має бути обов'язковою, причина в цьому не лише у зручності, швидкої перевірки та надання необхідних документів, а й у безпеці. Для прикладу візьмемо програму "Дія", за останні роки роботи цієї програми у нього з'явилася купа клонів, які копіювали абсолютно весь дизайн і функціонал, при цьому дані та сертифікати були підробленими, через те, що система досі не інтегрована, навіть зараз все більше клонів з'являються світ.

Безумовно, держава карає підробників, але всі випадки покарань були зроблені через дурість і не обережність підробників. Варто задуматися скільки зараз існує таких підроблених додатків, на які досі не звернули увагу через те, що

немає інтеграції.

Для того щоб інтегрувати програму, потрібно спланувати та надати клієнтам публічне API.

API (програмний інтерфейс програми, інтерфейс прикладного програмування) - опис способів (набір класів, процедур, функцій, структур або констант), якими одна комп'ютерна програма може взаємодіяти з іншою програмою. Зазвичай входить до опису будь-якого інтернет-протоколу програмного каркасу або стандарту викликів функцій операційної системи. Часто реалізується окремою програмною бібліотекою чи сервісом операційної системи. Використовується програмістами під час написання різноманітних додатків.

Інтеграції додатка до інших додатків вважається однією з найскладніших завдань, оскільки успіх додатка у інтеграціях. Чим більше користувачів у додатку, тим більша у нього популярність, відповідно різні сервіси приділяють більше уваги інтегрування у свою програму.

Наприклад взяти Facebook, Twitter для інтеграції вони створили окремі акаунти розробників, у яких без підтвердження своєї особистості більшість функціонала заблокована. Для розблокування облікового запису потрібно надати свої фотографії або паспортні дані, залежить від політики компанії.

Це один з кращих способів на даний момент, так як це відразу ж захищає додаток від непередбачених інтеграції, наприклад, зроблених знову ж таки для створення клонів.

1.1.4 Веб-портал

Також однією з важливих частин є веб-портал. Веб-портал для використання різних адміністративних послуг, це можна зробити і в смартфоні, але це незручно особливо, коли користувач працює з великими формами, також легше провести валідацію полів, ідентифікувати користувача, додати спливаючі підказки для пояснення форм. Потрібно також враховувати, що веб-портал можуть відкрити через якийсь браузер на смартфоні, отже, потрібно масштабувати в залежності від розміру та якості екрану смартфона.

Також веб-портал вважається безпечнішим за смартфон, оскільки він тісно пов'язаний з сервером, а сервер залежить від стану бездротової мережі. Авторизацію можна зробити за допомогою однієї з найсильніших архітектур для ідентифікації користувача JWT Refresh Token.

JSON Web Token (JWT) – це відкритий стандарт (RFC 7519) для створення токенів доступу, заснований на форматі JSON. Як правило, використовується для передачі даних для аутентифікації в клієнт-серверних програмах. Токени створюються сервером, підписуються секретним ключем і передаються клієнту, який надалі використовує цей токен для підтвердження своєї особистості [2] (рисунок 1.1).

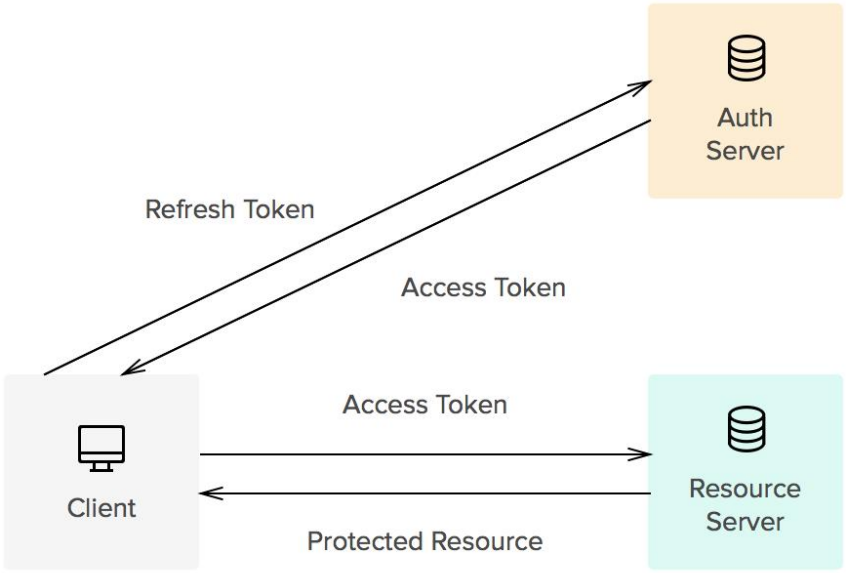


Рисунок 1.1 – Приклад взаємодії клієнта та JWT токенів

Ця архітектура дозволяє користувачеві зберігати в локальному сховищі зашифрований Refresh Token, який потрібен для того, щоб сервер обробив і дістав клієнту за допомогою алгоритму RSA-256 Access Token, і якщо Refresh Token правильний, то після того, як сервер віддасть клієнту Access Token, він зможе заходити будь-які ресурси, відправляти необхідні йому форми, отримувати

необхідну інформацію. Раніше використовували тільки Access Token, але з приходом такої вразливості як "Man in the middle", архітектурі були потрібні зміни.

"Man in the middle" ще називають як атака посередника, або атака "людина посередині" - вид атаки у криптографії та комп'ютерній безпеці, коли зловмисник таємно ретранслює і при необхідності змінює зв'язок між двома сторонами, які вважають, що вони безпосередньо спілкуються одна з одною. Є методом компрометації каналу зв'язку, у якому зломщик, підключившись до каналу між контрагентами, здійснює втручання у протокол передачі, видаляючи чи спотворюючи інформацію (рисунок 1.2) [3].

Рис. 1.2 – Наглядна схема атаки «людина посередині»

Одним із прикладів атак типу «людина посередині» є активне прослуховування, при якому зловмисник встановлює незалежні зв'язки з жертвами та передає повідомлення між ними. Тим самим він змушує жертв повірити, що вони розмовляють безпосередньо один з одним через приватний зв'язок, фактично вся розмова управляється зловмисником. Зловмисник повинен вміти перехоплювати всі повідомлення, що передаються між двома жертвами, а також вводити нові. Найчастіше це досить просто: наприклад, зловмисник може поводитися як «людина посередині» у межах діапазону прийому бездротової точки доступу.

Оскільки захищає Refresh Token, справа в тому, що публічний токен має термін придатності. Проміжок може бути різним від п'яти хвилин до місяця. Так от, якщо зловмисник зможе отримати доступ до ресурсу, то лише на певний проміжок. Але що буде, якщо хакер забере не тільки публічний токен, але й токен для оновлення, можна не турбуватися оскільки токен для оновлення зберігається в зашифрованому вигляді, а ключ від нього зберігається в базі даних у свою чергу база даних захищена сервером.

Варто відзначити, що Refresh Token теж має свій термін придатності, але зазвичай набагато довше, ніж публічний.

1.2 Опис сучасних проблем

У цьому пункті буде описано, чому поточні аналоги додатків роботи з державою мають помилки, які проблеми їм варто вирішити, і які рішення будуть прийняті для усунення цих проблем.

Найкращим прикладом у світі буде застосування "Дія", тому що аналогів йому не існує на даний момент, а якщо є то вони сирі, недопрацьовані. У світі вже існують електронні паспортні дані та інші документи, що засвідчують особу, але жодна країна поки що не пробувала оцифровувати адміністративні документи або заповнити їх онлайн. «Україна – третя в Європі та десята країна у світі, в якій є електронні документи у смартфоні», – сказав міністр цифрової трансформації Михайло Федоров. Але це далеко від правди. Як уже й описувалося вище, ідея об'єднати всі адміністративні послуги, а також збереження унікальної інформації про користувача для роботи з державними послугами онлайн неймовірно амбітна, у майбутньому це гарантовано буде затребувано. Це допоможе вирішити масу проблем, наприклад, пов'язаної зі втратою та відновленням паспорта, веденням свого бізнесу тощо. Очевидно, головний плюс - це зручність. Не потрібно більше носити купу різних папірців у папці, яку можна десь втратити чи забути, досить лише наявність смартфона.

Головною проблемою інших схожих додатків є безпека даних, підробки, а також способи структурування інформації. Кожен конкурент має своє бачення на

безпеку і на жаль ідеального способу захист не існує. Багато розробників можуть лише доповнити якісь існуючі засоби захисту, існуючими методами, наприклад: Google Authentificator, PGP-ключ тощо.

Нажаль самі користувачі не готові до того, щоб дотримуватися правил їхньої ж безпеки. Для того, щоб уникнути зломів і дати необхідну інформацію користувачеві потрібно зробити курс цифрової грамотності для звичайного громадянина. Користувач зобов'язаний пройти цифровий курс перед завантаженням програми, інакше сам користувач буде вразливим, а це, на жаль, ніяк не виправити.

1.2.1 Підвищена таємничість

Оскільки додаток працює з чутливими даними громадян, щоб йому можна було довіряти, його код має бути відкритим. Таким чином його можна переглядати та вивчати, тому незалежні фахівці зможуть перевірити твердження Мінцифри про безпеку персональних даних.

Наприклад, документація державних цифрових послуг Сінгапуру доступна на одному з найбільших веб-сервісів для спільної розробки програмного забезпечення GitHub, там є українська система електронних публічних закупівель Prozorro і навіть вихідний код ядра Linux. Але ніде у відкритому доступі немає документації та технічного опису програми «Дія».

До того ж код програми пройшов через процедуру обфускації, тобто заплутування, що ускладнює аналіз та розуміння алгоритму роботи програми. За словами Шона Таунсенда, заплутування коду є досить поширеною практикою, але у випадку з державним додатком варто було б навпаки все опублікувати і пояснити, чому прийняті саме такі технічні рішення.

1.2.2 Зберігати паролі та ключі у смартфоні ненадійно

Крім цього, шахраї успішно викрадають SIM-картки, а потім отримують доступ до рахунків у банку, до пошти, паспорта та даних ФОП. Це відбувається так: абоненту дзвонять багато разів, домагаються щоб він самостійно

передзвонив. Потім замовляють у оператора послугу відновлення SIM-картки, коли для ідентифікації абонента його просять назвати кілька останніх вхідних або вихідних номерів. Також іноді зловмисники можуть перерахувати незначні суми коштів на мобільний рахунок своєї жертви, аби точно знати дату останнього поповнення рахунку. Це глобальна проблема. Американський Bitcoin-підприємець Майкл Терпін кілька разів втрачав заощадження, бо вкрали мобільний номер. Навіть після того, як він замовив у свого оператора VIP-статус із посиленою безпекою, у нього знову вкрали номер та зняли з рахунків криптовалюту, яка коштувала на той момент 23 млн доларів.

Крадіжки через інтернет-банкінг взагалі стали буденністю. У месенджері Telegram з'явилися два боти, які шукають персональні дані українських користувачів, у тому числі серію та номер паспорта, прописку та ІПН, за номером телефону за гроші. Боти, швидше за все, беруть дані з вкраденої бази користувачів «ПриватБанку»

1.2.3 Паролі, підписи, ключі ідентифікують пристрій, а не людину

Наприкінці серпня 2016 року в системі е-декларування з'явилася підроблена декларація, заповнена невідомою особою від імені члена Нацагентства щодо запобігання корупції Руслана Рябошапки про те, що він нібито отримав 25 мільйонів гривень від фіктивної компанії. Це не злам: фальшива електронна декларація була підписана справжнім електронним ключем, створеним держпідприємством «Українські спеціальні системи». Винні у підробці досі не знайшлися, а Державна служба спеціального зв'язку та захисту інформації (ДССЗЗІ) і зовсім не побачила потреби у перевірці центру сертифікації ключів ДП «УСС». Двері не бачать, хто вставляє ключ. Комп'ютер не бачить, хто вводить пароль. Ваш ключ – це не ви. Якщо хтось дістанеться вашого телефону, то зможе отримати цифровий підпис на ваше ім'я в «ПриватБанку» в режимі онлайн. І використовувати її з іншого комп'ютера без вашого відома.

1.2.4 Державні реєстри працюють погано, проте їх поєднують

Сам голова Мінцифри Михайло Федоров визнає, що технічний стан державних реєстрів є жахливим. Водночас, стверджує, що відомство працює над наведенням порядку.

Співрозмовник Заборони Роман Хіміч пояснює, що низька якість реєстрів є наслідком складного комплексу проблем, на які Мінцифри не впливає. «Про проблеми з реєстрами персональних даних громадян відомо щонайменше років десять. Йдеться про наявність безлічі хибних даних у межах окремо взятих реєстрів та, що особливо важливо, невідповідності між персональними даними громадян у різних реєстрах. Це ускладнює чи неможливе масу, здавалося б, тривіальних завдань на кшталт запиту даних про громадянина самими чиновниками. Тому у випадку з додатком «Дія» ми маємо саме ситуацію цифровізації безладдя, наслідком чого буде вже цифрове безладдя» [4].

1.3 Аналіз існуючих методів здобуття знань

Для того, щоб програми працювали стабільно, потрібно отримати необхідні дані, наприклад: список адміністративних послуг, поля для паспортних даних, поля для водійських прав та інше. На щастя, проект "Дія" у відкритому доступі завантажила файл з усіма доступними послугами в даний момент, щоб його використовувати потрібно придумати алгоритм для парсингу файлу.

Парсинг - це процес автоматичного збору даних та їх структурування. На даний момент, є багато різних видів парсингу, від якихось точкових сайтів до великих файлів з баз даних. У цій дипломній роботі використовуватиметься парсинг для файлів з розширенням Excel.

Для паспортних даних буде використано базові поля, які використовуються у паперовому вигляді. Так як додаток це лише макет за основу полів для паспортних або водія можна взяти з інтернету.

1.3.1 Види парсингу

Відповідно до розмежування між бізнес-аналітикою та бізнес-розвідкою, відповідні аналітичні методи можна розділити на різні загальні категорії. Існує

поділ на ці п'ять категорій:

- Описовий парсинг;
- Дослідницький парсинг;
- Діагностичний парсинг;
- Прогностичний парсинг;
- Наказує парсинг;

Описовий парсинг, також відомий як описовий аналіз даних, зосереджений даних минулих років. Він організовує та структурує емпіричні дані. Наприклад, у ньому міститься така інформація, як обсяг продажу за останній квартал або тип та кількість запитів на обслуговування. Для отримання таких результатів описовий аналіз може отримати дані з різних джерел та узагальнити, систематизувати та структурувати інформацію. Описовий аналіз часто комбінується з іншими методами аналізу.

Метою дослідницького аналізу даних є пошук зв'язків у даних та генерація гіпотез. До проведення цього виду парсингу існують обмежені знання про взаємозв'язок даних та змінних. Типовою сферою застосування для аналізу розвідувальних даних є видобуток даних. Виявлення кореляцій за допомогою аналізу розвідувальних даних дозволяє зробити висновки про причини процесів.

Аналіз діагностичних даних стосується саме питання «Чому щось трапилося?». Порівнюючи історичні та інші дані, виявляючи закономірності та виявляючи взаємини, він знаходить причини чи взаємини. За допомогою аналізу діагностичних даних організації можуть вирішувати конкретні проблеми з виявлення їх корінних причин.

Прогнозний парсинг, також відомий як прогностичний аналіз, дозволяє зазирнути у майбутнє. Щоб зробити правильний прогноз, під час аналізу прогностичних даних використовуються результати описаних раніше методів описового, дослідного чи діагностичного аналізу, а також алгоритмів та методів штучного інтелекту (ШІ) та машинного навчання (МН). Пошук кореляцій, причин та тимчасових тенденцій робить майбутні тенденції передбачуваними. Імовірність та точність прогнозування багато в чому залежить від якості даних,

закономірностей, знайдених кореляцій та тенденцій, а також від інтелекту алгоритмів. Наприклад, можна передбачити майбутні продажі чи поведінку покупців.

Наказує аналіз даних є найбільш складною і дорогою категорією аналізу. Вони використовуються результати, які стосуються категорій аналізаторів. Використовуються права ML та AI, нейронні права, права та правила ведення бізнесу [5].

1.3.2 Парсинг Excel файлів за допомогою бібліотеки Aspose.Cells

Aspose.Cells for .NET дає вам повний контроль за установками сторінки і дозволяє вам маніпулювати широким колом опцій по відображенню робочих аркушів Excel, таких як перегляд розривів сторінок і рівня наближення для робочого аркуша, контроль видимості даних з використанням панелі "заморозки", установка опцій орієнтації сторінки, розтягування, розміру сторінки, верхній та нижній колонтитули та області друку тощо.

Бібліотека пропонує широкий діапазон параметрів безпеки, включаючи підтримку захисту XLSX файлу Excel 2007, захист контенту, об'єктів та сценаріїв робочого листа та можливість приховувати та показувати робочі листи.

Також бібліотека дарує вам можливість маніпулювати рядками та стовпцями різними способами. Ви можете легко керувати висотою рядка та шириною колонки, якщо потрібно. Ви зможете встановлювати автоматичний вибір розміру висоти/ширини комірки відповідно до контенту, вставляти та видаляти, приховувати та показувати, групувати та розгруповувати рядки та колонки.

Бібліотека має абсолютний контроль того, як ви відображатимете дані відповідно до того, які пропонуються розширені можливості форматування. Ви можете застосовувати індивідуальне форматування до робочих аркушів, рядків, стовпців та комірок. Ви також можете додавати збагачений форматуванням текст в одиночний осередок або застосовувати різне оформлення рамок, заднього фону та шрифтів. До комірок також може застосовуватися умовне форматування [6].

1.3.3 Постановка задачі

Система буде складатися із кількох основних компонентів. Першою частиною буде веб-портал, який реалізує можливість вибору адміністративної послуги, категорії. У веб-портал буде додано інформація про користувача, а також коротка інформація про категорію, можливість подати заяву, та заповнення форми.

Друга частина – це мобільна програма в якому відображатиметься інформація про користувача, також можна буде проводити операції з відновлення втрачених документів. Тут потрібно зробити акцент на безпеці, тому що мобільний додаток зламати найпростіше порівняно з веб-порталом. Не зберігати дані в локальному сховищі, підписувати користувача до облікового запису не тільки за номером телефону, а й за MAC-адресою ідентифікатора смартфона.

MAC-адреса (Media Access Control – нагляд за доступом до середовища, фізична адреса пристрою) – унікальний ідентифікатор, який присвоюється кожній одиниці активного обладнання або деяким їх інтерфейсам у комп'ютерних мережах Ethernet.

Необхідно визначити спосіб парсингу документів, зазначених вище. Необхідно розробити алгоритм із використанням інструментів Aspose.Cells для визначення вхідних даних та відображати дані на моделі. Модель може використовуватися для обробки інформації за вхідними даними, які можуть приходити з файлів для заповнення бази даних.

Таким чином, можна виділити чотири основні критерії для розробки алгоритму:

– Парсинг файлів із інструментами Aspose.Cells.

– Відобразити спарсені дані на моделі.

– Створити алгоритми генерації файлів у таких форматах, як pdf, docx

– Додати надійний захист даних при авторизації користувача

– Записати дані з моделі до бази даних.

– Використовувати дані у веб-портали.

За критеріями можна скласти загальну характеристику документів, в яку

можуть входити різні форми з веб-додатка, шаблони різних юридичних документів, зображення, валідацію вхідних даних з різних форм, генерація документів після валідації вхідних даних.

Таким чином, для дослідження та розробки компонентів документообігу у цій роботі ставляться такі завдання:

1. Провести порівняльний аналіз існуючих методів для автоматизації компонентів документообігу. Необхідно порівняти існуючі методи, виділені після ознайомлення з наявними джерелами інформації. При порівнянні слід відзначити слабкі і сильні сторони, яких у розроблюваному методі хотілося б позбутися, або реалізувати. Необхідно розглянути такі критерії, як складність реалізації, вимогливість до апаратних ресурсів, обсяг даних, який можна обробляти за короткий час.

2. Розробити метод, що враховує всі зазначені раніше недоліки та плюси рішень, призначених для вирішення цього питання. Необхідно реалізувати у методі найбільш вдале поєднання плюсів за аналогічними рішеннями задля досягнення максимальної ефективності.

3. Провести дослідження ефективності розроблених методів генерації файлів для компонентів документообігу із цифровими документами. Необхідно оцінити шляхи можливого розвитку у майбутньому розроблених методів. Провести порівняльний аналіз із подібними методами. Розглянути та визначити більш зручні області для використання методу.

4. Спроектувати інформаційну систему, що реалізує розроблений метод, що дозволяє повною мірою використати його потенціал. Визначити, до яких сфер метод у даній інформаційній системі буде більш практично використовувати та визначити, чи може спроектована інформаційна система в майбутньому масштабуватися до більшої пошукової системи для різних веб-додатків.

1.4 Розробка архітектури та компонентів системи

1.4.1 Мікросервіси

В першу чергу потрібно приділити увагу архітектурі через величезну

кількість сутностей, які в майбутньому допомагатимуть розширювати систему. Для програмної системи було вирішено взяти мікросервісну архітектуру, яка дозволить розширювати функціонал, а також додавати новий, не порушуючи принципів ООП і не знижуючи продуктивності. Також мікросервіси дозволять створювати зручний ланцюжок сутностей передачі необхідних даних пов'язані з різними правовими галузями. Окремі послуги надзвичайно прості і можуть розгортатися незалежно один від одного, а це означає, що для того, щоб змінити якийсь рядок у коді, не потрібно прийняття рішень на верхньому рівні. Тим самим внесення дрібних змін більше не забирає зайвого часу. Невеликі команди, які працюють над окремими сервісами, можуть також об'єднуватися в багатофункціональні колективи з Agile-методології.

При розробці додатків з монолітної архітектури зазвичай використовують велику реляційну базу даних, єдину для всього додатка, навіть якщо для якихось окремих процесів існують простіші та зручніші рішення. Це робить всю архітектуру менш ефективною та громіздкою. Що стосується мікросервісами кожен із новачків може мати власний стек, модель і базу даних, оптимізовані для конкретного процесу.

Зрозуміло, не буває ідеальних рішень, і мікросервіси теж мають слабкі місця. Головний недолік мікросервісів криється в самій їх природі: розподілена система з безліччю незалежних елементів складніша як в організаційному, так і в архітектурному плані. Ускладнюється управління командами розробки та розгортання, і тут не обійтися без методологій Agile та DevOps. Розподілений доступ до сервісів означає збільшення мережних затримок та потенційних збоїв, з чим можна боротися шляхом асинхронності та скорочення кількості викликів.

Не варто забувати і про необхідність забезпечувати узгодженість програми: через децентралізацію та модульність можливе виникнення неконсистентності даних, що також призводить до збоїв і недоступності програми в цілому. В цьому випадку варто шукати компроміси між доступністю сервісів та консистентністю.

1.4.2 Мова програмування

Як мову програмування було обрано C# оскільки він належить до платформі .NET. Ця технологія була обрана через такі основні риси:

- Підтримка кількох мов;
- Кросплатформність;
- Потужна бібліотека класів;
- Різноманітність технологій;
- Продуктивність;

На сьогоднішній момент мова програмування C# одна з найпотужніших мов, що швидко розвиваються і затребуваних в IT-галузі. Зараз на ньому пишуться різні програми: від невеликих десктопних програм до великих веб-порталів і веб-сервісів, що обслуговують щодня мільйони користувачів.

C# вже не молода мова і, як і вся платформа .NET, вже пройшов великий шлях. Перша версія мови вийшла разом із релізом Microsoft Visual Studio .NET у лютому 2002 року. Поточною версією мови є версія C# 10.0, яка вийшла 9 листопада 2021 разом із релізом .NET 6. C# є мовою із Сі-подібним синтаксисом і близький у цьому відношенні до C++ та Java. Тому, якщо ви знайомі з однією з цих мов, то опанувати C# буде легше.

C# є об'єктно-орієнтованим і в цьому плані багато перейняв у Java та C++. Наприклад, C# підтримує поліморфізм, успадкування, навантаження операторів, статичну типізацію. Об'єктно-орієнтований підхід дозволяє вирішити задачі по побудові великих, але в той же час гнучких, масштабованих і додатків, що розширюються. І C# продовжує активно розвиватися і з кожною новою версією з'являється все більше цікавих функціональностей.

1.4.3 База даних

Бази даних – це сховища, які містять великі масиви інформації. Відправляти запити, отримувати інформацію їх можна з допомогою СУБД – систем управління. Популярна СУБД, розроблена компанією Microsoft - SQL Server. MSSQLServer дозволить зберігати та масштабувати величезну кількість даних.

Також не потрібно хвилюватися, що ця технологія застаріє, адже оновлення виходять щороку.

SQL Server підтримує віддзеркалення та кластеризацію баз даних.Кластер сервера SQL – це сукупність однаково конфігурованих серверів; така схема допомагає розподілити робоче навантаження між кількома серверами. Усі сервери мають одне віртуальне ім'я, і дані розподіляються за IP-адресами машин кластера протягом робочого циклу. Також у разі відмови або збою на одному із серверів кластера доступний автоматичний перенесення навантаження на інший сервер.

У SQL Server 2014 вбудована підтримка .NET Framework. Завдяки цьому процедури БД, що зберігаються, можуть бути написані будь-якою мовою платформи .NET, використовуючи повний набір бібліотек, доступних для .NET Framework, включаючи Common Type System (система поводження з типами даних у Microsoft .NET Framework). Однак, на відміну від інших процесів, .NET Framework, будучи базовою системою для SQL Server 2014, виділяє додаткову пам'ять та вибудовує засоби управління SQL Server замість того, щоб використовувати вбудовані засоби Windows. Це підвищує продуктивність порівняно із загальними алгоритмами Windows, оскільки алгоритми розподілу ресурсів спеціально налаштовані для використання у структурах SQL Server.

Основні переваги:

- Масштабування системи. Взаємодіяти з нею можна як на простих ноутбуках, так і на комп'ютері з потужним процесором, який здатний обробляти великий обсяг запитів.
- Розмір сторінок – до 8 Кб. Дані отримуються швидко, а складну інформацію зручніше зберігати. Система обробляє транзакції в інтерактивному режимі, є динамічне блокування.
- Автоматизація рутинних адміністративних завдань. Наприклад, управління блокуваннями та пам'яттю, редагування розмірів файлів. У програмі продумано налаштування, можна створювати профілі користувачів.
- Зручний пошук. Його можна здійснювати за фразами, словами, текстом чи створювати ключові індекси.

- Підтримка роботи з іншими рішеннями Microsoft, у тому числі з Excel, Access.

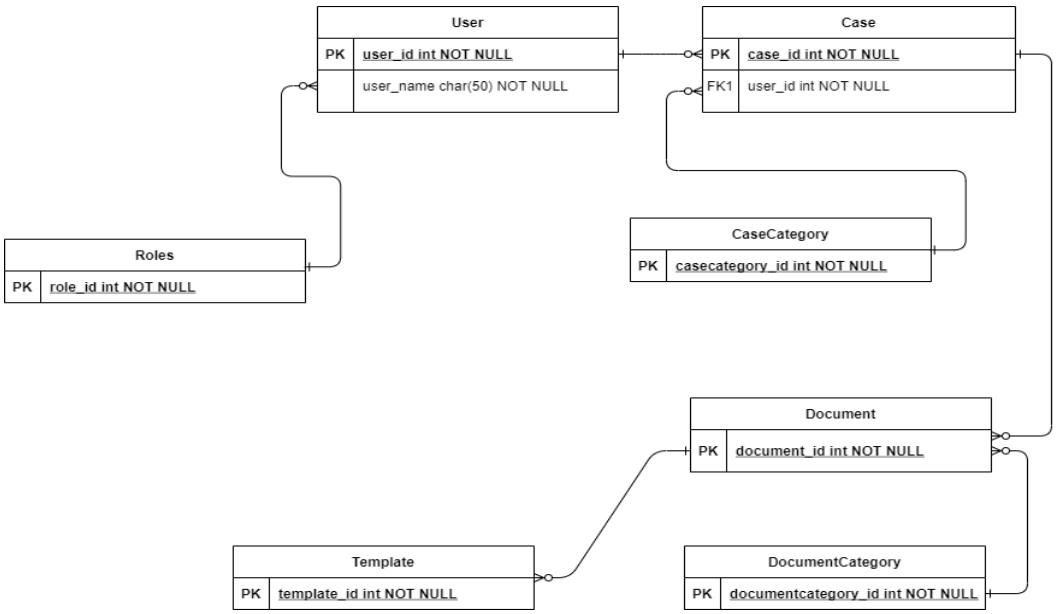


Рисунок 1.3 – Діаграма бази даних

1.4.4 Зовнішня частина додатку із бібліотекою JQuery

jQuery - це популярна Javascript бібліотека, здатна істотно спростити життя веб-розробнику. Бібліотека jQuery містить функціонал, корисний максимально широкого кола завдань. Тим не менш, розробниками бібліотеки не ставилося завдання поєднання в jQuery функцій, які підійшли б усюди, оскільки це призвело б до великого коду, більша частина якого не затребувана. Тому була реалізована архітектура компактного універсального ядра бібліотеки та плагінів. Це дозволяє зібрати для ресурсу саме той JavaScript-функціонал, який на ньому був затребуваний

Бібліотека jQuery в першу чергу забезпечує несуперечливу роботу програмного коду у всіх типах браузерів, вирішуючи такі складні проблеми JavaScript, як очікування на завантаження сторінки перед тим, як виконувати будь-які операції.

На той випадок, якщо в бібліотеці виявиться нестача функціональності, розробники передбачили простий, але дуже ефективний метод її розширення.

Багато програмістів-початківців jQuery виявляють цю гнучкість на практиці, розширюючи можливості jQuery в перший же день.

Щоб привнести динамічну функціональність на будь-яку інтернет-сторінку, доводиться слідувати одному і тому ж шаблону: спочатку відбирається елемент або група елементів, а потім над ними виконуються деякі дії, наприклад, приховувати або показувати елементи, що цікавлять нас, додавати до них клас CSS, створювати анімаційні ефекти або змінювати атрибути. Зі звичайним JavaScript для вирішення кожної із завдань потрібно десятки рядків програмного коду. Автор jQuery розробив свою бібліотеку саме для того, щоб зробити найбільш спільні завдання тривіальними. Наприклад, щоб створити таблицю з різним кольором фону для парних та непарних рядків, дизайнеру потрібно написати до 10 рядків коду мовою JavaScript, а ось з використанням jQuery цей ефект досягається з використанням не більше одного рядка.

1.4.5 Вимоги до ресурсів

Для того щоб підтримувати технологію завжди в оптимальному стані, потрібно враховувати такі критерії:

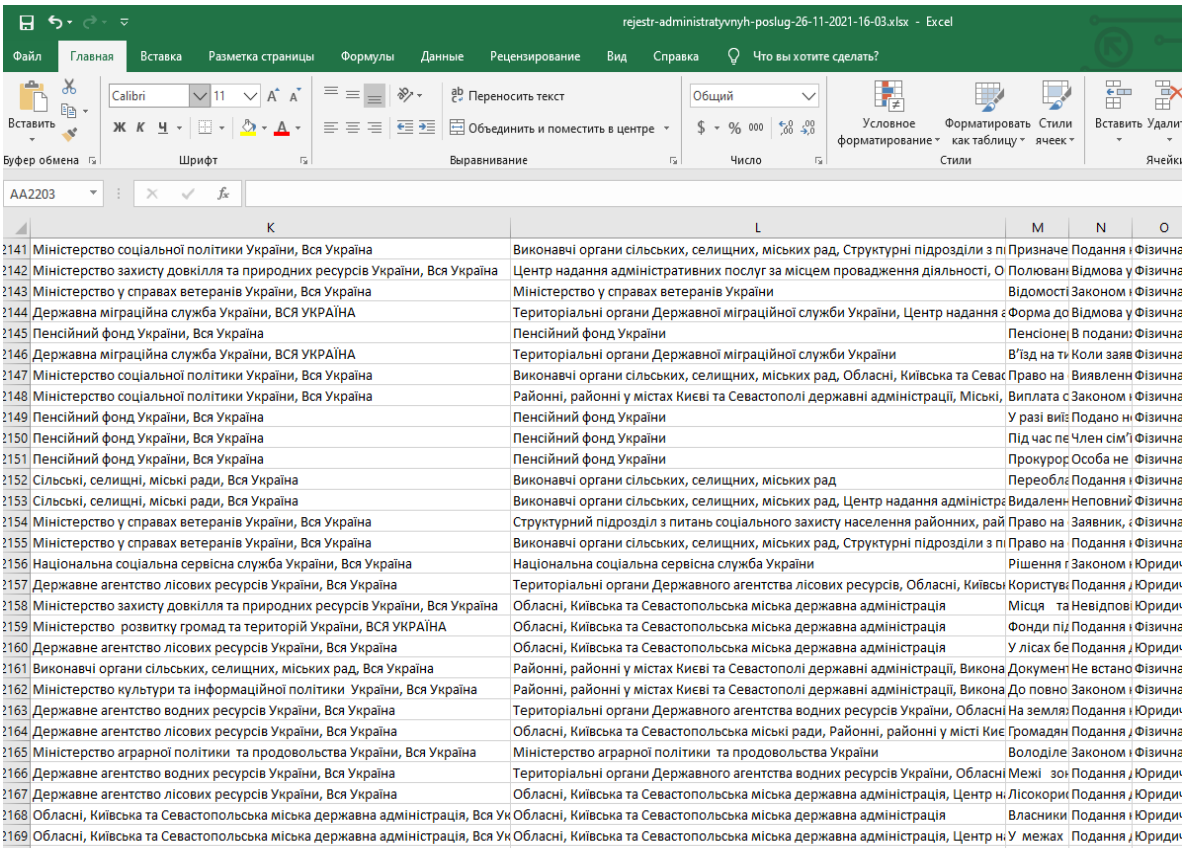
- Можливість збільшувати сховище файлів;
- Підтримувати веб-додаток у хмарі без збоїв;
- Підтримувати оптимальні дані на всіх мікросервісах;

Для уникнення порушень умов необхідно виділити гроші на підтримку серверів. Наприклад, можна купити вже готові хмарні сервери.

2 ДОСЛІДЖЕННЯ ІСНУЮЧИХ МЕТОДІВ ПАРСИНГУ ТА ГЕНЕРАЦІЇ ДОКУМЕНТІВ

2.1 Парсинг початкових даних

Перед початком необхідно заповнити базу даних потрібними адміністративними послугами. Щоб це зробити, потрібно вивчення адміністративних державних органів, але на даний момент у відкритому доступі є необхідні файли, які зберігають більшу частину існуючих послуг (рисунок 2.1).



	K	L	M	N	O
2141	Міністерство соціальної політики України, Вся Україна	Виконавчі органи сільських, селищних, міських рад, Структурні підрозділи з п	Призначе	Подання	Фізична
2142	Міністерство захисту довкілля та природних ресурсів України, Вся Україна	Центр надання адміністративних послуг за місцем провадження діяльності, О	Полювані	Відмова у	Фізична
2143	Міністерство у справах ветеранів України, Вся Україна	Міністерство у справах ветеранів України	Відомості	Законом	Фізична
2144	Державна міграційна служба України, ВСЯ УКРАЇНА	Територіальні органи Державної міграційної служби України, Центр надання а	Форма до	Відмова у	Фізична
2145	Пенсійний фонд України, Вся Україна	Пенсійний фонд України	Пенсіоне	В подани	Фізична
2146	Державна міграційна служба України, ВСЯ УКРАЇНА	Територіальні органи Державної міграційної служби України	В'їзд на т	Коли заяв	Фізична
2147	Міністерство соціальної політики України, Вся Україна	Виконавчі органи сільських, селищних, міських рад, Обласні, Київська та Севас	Право на	Виявлення	Фізична
2148	Міністерство соціальної політики України, Вся Україна	Районні, районні у містах Києві та Севастополі державні адміністрації, Міські,	Виплата с	Законом	Фізична
2149	Пенсійний фонд України, Вся Україна	Пенсійний фонд України	У разі вия	Подано н	Фізична
2150	Пенсійний фонд України, Вся Україна	Пенсійний фонд України	Під час пе	Член сім'ї	Фізична
2151	Пенсійний фонд України, Вся Україна	Пенсійний фонд України	Прокурор	Особа не	Фізична
2152	Сільські, селищні, міські ради, Вся Україна	Виконавчі органи сільських, селищних, міських рад	Переобла	Подання	Фізична
2153	Сільські, селищні, міські ради, Вся Україна	Виконавчі органи сільських, селищних, міських рад, Центр надання адміністра	Видаленн	Неповний	Фізична
2154	Міністерство у справах ветеранів України, Вся Україна	Структурний підрозділ з питань соціального захисту населення районних, рай	Право на	Заявник,	Фізична
2155	Міністерство у справах ветеранів України, Вся Україна	Виконавчі органи сільських, селищних, міських рад, Структурні підрозділи з п	Право на	Подання	Фізична
2156	Національна соціальна сервісна служба України, Вся Україна	Національна соціальна сервісна служба України	Рішення г	Законом	Юридич
2157	Державне агентство лісових ресурсів України, Вся Україна	Територіальні органи Державного агентства лісових ресурсів, Обласні, Київсь	Користува	Подання	Юридич
2158	Міністерство захисту довкілля та природних ресурсів України, Вся Україна	Обласні, Київська та Севастопольська міська державна адміністрація	Місця та	Невідпов	Юридич
2159	Міністерство розвитку громад та територій України, ВСЯ УКРАЇНА	Обласні, Київська та Севастопольська міська державна адміністрація	Фонди пі	Подання	Фізична
2160	Державне агентство лісових ресурсів України, Вся Україна	Обласні, Київська та Севастопольська міська державна адміністрація	У лісах бе	Подання	Юридич
2161	Виконавчі органи сільських, селищних, міських рад, Вся Україна	Районні, районні у містах Києві та Севастополі державні адміністрації, Викона	Документ	Не встано	Фізична
2162	Міністерство культури та інформаційної політики України, Вся Україна	Районні, районні у містах Києві та Севастополі державні адміністрації, Викона	До повно	Законом	Фізична
2163	Державне агентство водних ресурсів України, Вся Україна	Територіальні органи Державного агентства водних ресурсів України, Обласні	На земля	Подання	Юридич
2164	Державне агентство лісових ресурсів України, Вся Україна	Обласні, Київська та Севастопольська міські ради, Районні, районні у місті Кие	Громадян	Подання	Фізична
2165	Міністерство аграрної політики та продовольства України, Вся Україна	Міністерство аграрної політики та продовольства України	Володіле	Законом	Фізична
2166	Державне агентство водних ресурсів України, Вся Україна	Територіальні органи Державного агентства водних ресурсів України, Обласні	Межі зон	Подання	Юридич
2167	Державне агентство лісових ресурсів України, Вся Україна	Обласні, Київська та Севастопольська міська державна адміністрація, Центр на	Лісокори	Подання	Юридич
2168	Обласні, Київська та Севастопольська міська державна адміністрація, Вся Ук	Обласні, Київська та Севастопольська міська державна адміністрація	Власники	Подання	Фізична
2169	Обласні, Київська та Севастопольська міська державна адміністрація, Вся Ук	Обласні, Київська та Севастопольська міська державна адміністрація, Центр на	У межах	Подання	Юридич

Рисунок 2.1 – Файл із адміністративними послугами

Для того, щоб заповнити базу даних, потрібно розробити алгоритм парсингу. Одна з важливих речей при парсингу, щоб вхідні дані були строго структуровані під модель, яка в результаті передасть всю інформацію в базу даних.

Можна використати кілька методів, щоб заповнити базу даних необхідними вхідними даними. Один з них парсить файли з вже структурованими даними, які можна зручно відобразити на моделі, щоб потім використовувати її по всьому додатку. Також більшість баз даних має свої вбудовані інструменти для парсингу, наприклад Microsoft SQL Management Studio - це утиліта з Microsoft SQL Server 2005 та пізніших версій для конфігурування, управління та адміністрування всіх компонентів Microsoft SQL Server. Ця утиліта використовується для заповнення таблиць даними, які вже структуровані у самому файлі. Сама утиліта генерує запит для заповнення таблиць, який користувач не бачить. Цей запит досить оптимізований, тому що він може працювати дуже швидко, як з великою кількістю даних, так і з невеликою. Це означає, що всередині утиліти підбирається алгоритм вставки, який підганяється під кількість даних, що входять в базу.

2.2 Парсинг методом утиліти MSSQL

Утиліта включає скриптовий редактор та графічну програму, яка працює з об'єктами та налаштуваннями сервера. Також у MSSQL є інтеграція з різними файлами, наприклад з розширенням xslx, які як вище було описано, будуть використовуватися в парсингу (рисунок 2.2).

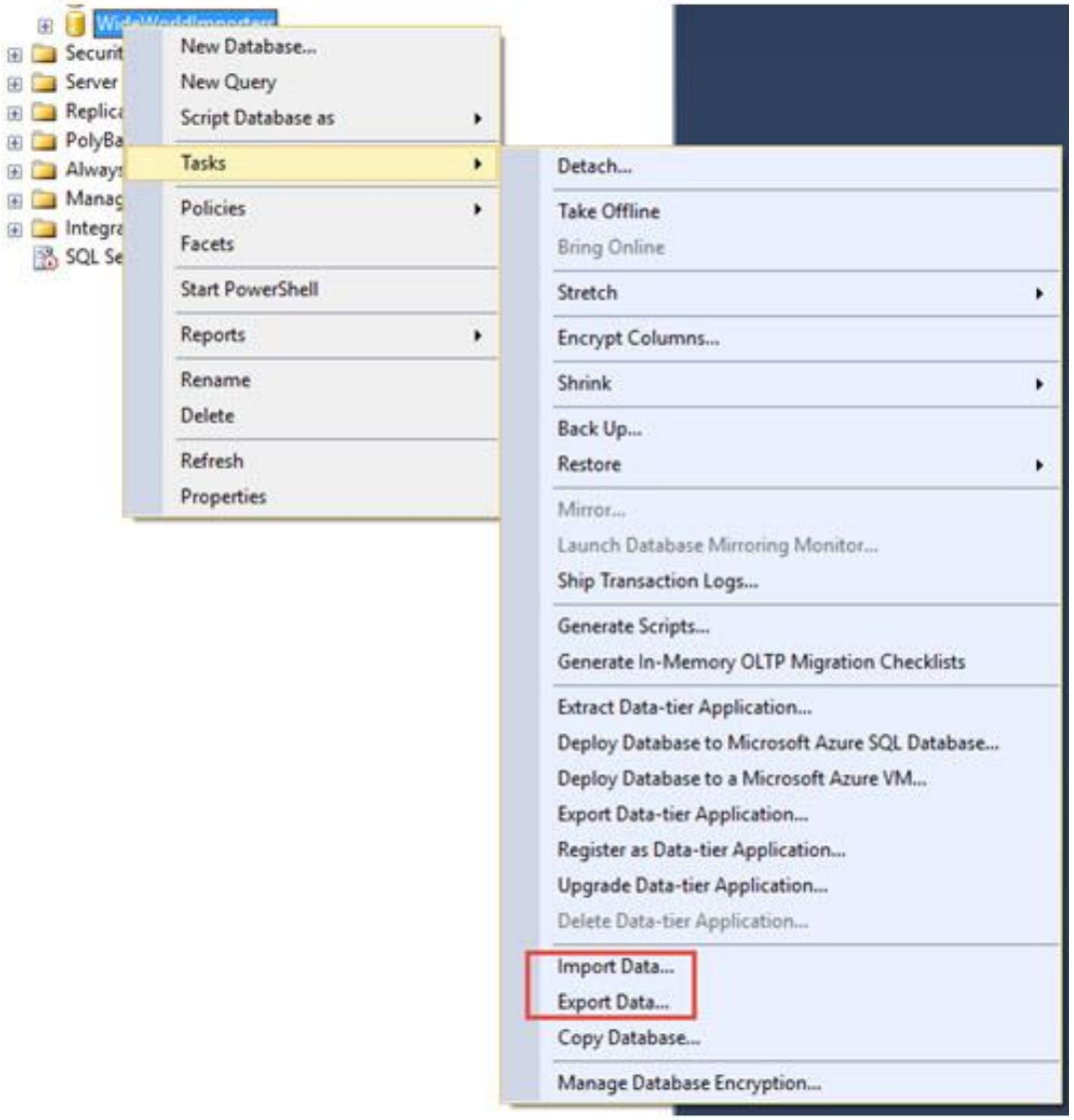


Рисунок 2.2 – Функції імпорту або експорту даних

Імпортувати дані безпосередньо з файлів Excel, дотримуючись інструкцій на сторінках майстра імпорту та експорту SQL Server. У разі потреби збережіть установки у вигляді пакета служб SQL Server Integration Services (SSIS), доступного для налаштування та багаторазового застосування в майбутньому (рисунок 2.3).

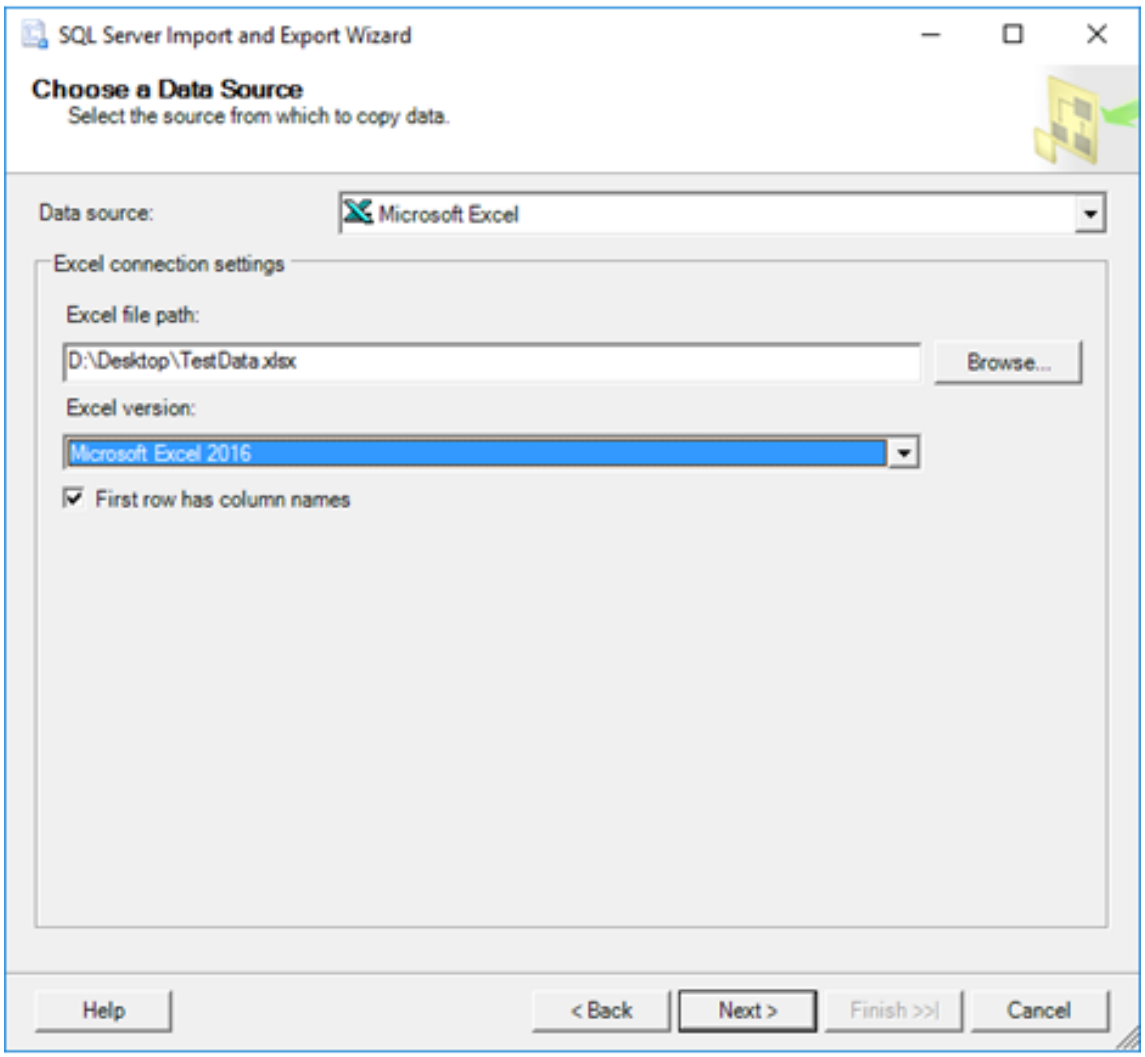


Рисунок 2.3 – Вікно із вибором джерела для імпорту даних

Для не структурованих даних у MSSQL є окремий спосіб імпорту даних. Потрібно зробити приблизно ті ж дії, що і минулого разу, але вибрати іншу функцію (рисунок 2.4). Це дозволить відкрити вікно у якому буду декілька шляхів описуючих, які файли імпортувати, як назвати таблицю на виході, які будуть колонки та інше.

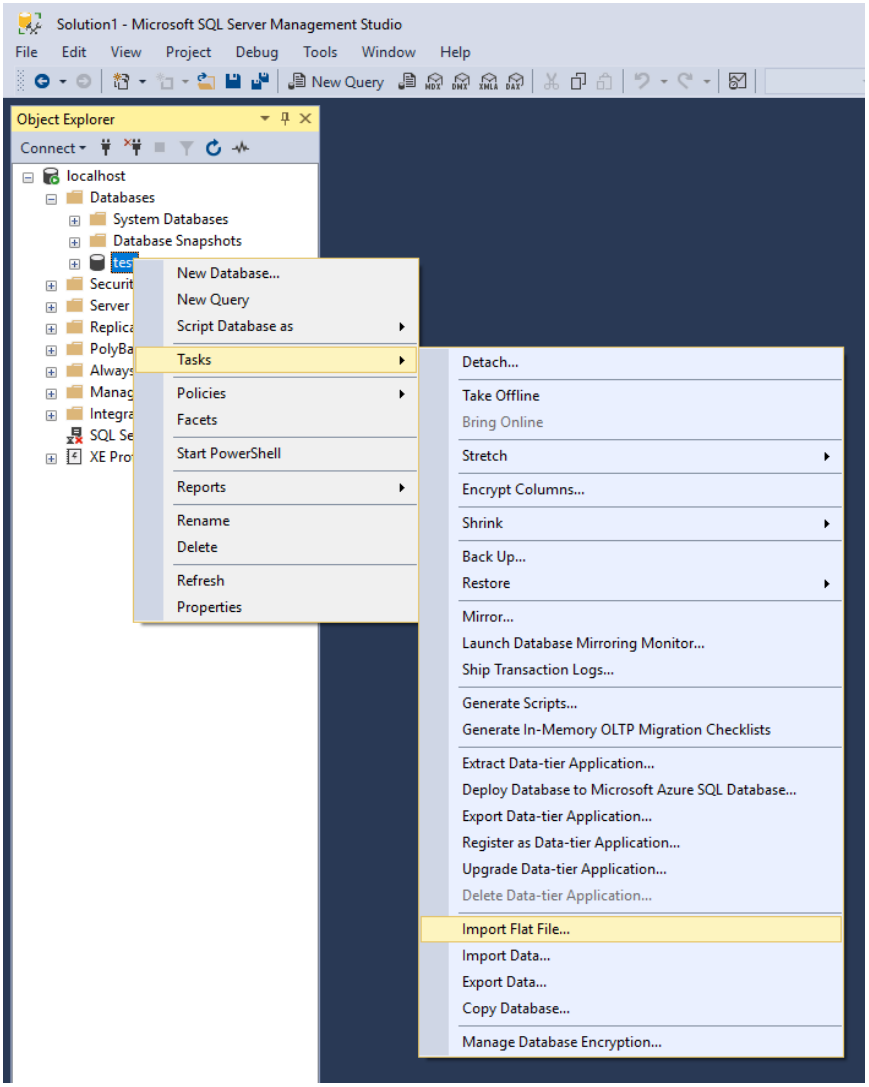


Рисунок 2.4 – Функція для інтегрування не структурованих даних

Адже далі відкриється вікно у якому треба визначити, який файл треба імпортувати (рисунок 2.5). Треба також звернути увагу на те що, імпортувати дані можна не тільки із розширенням xlsx, а й інші типи файлів, які працюють із середовищем Access.

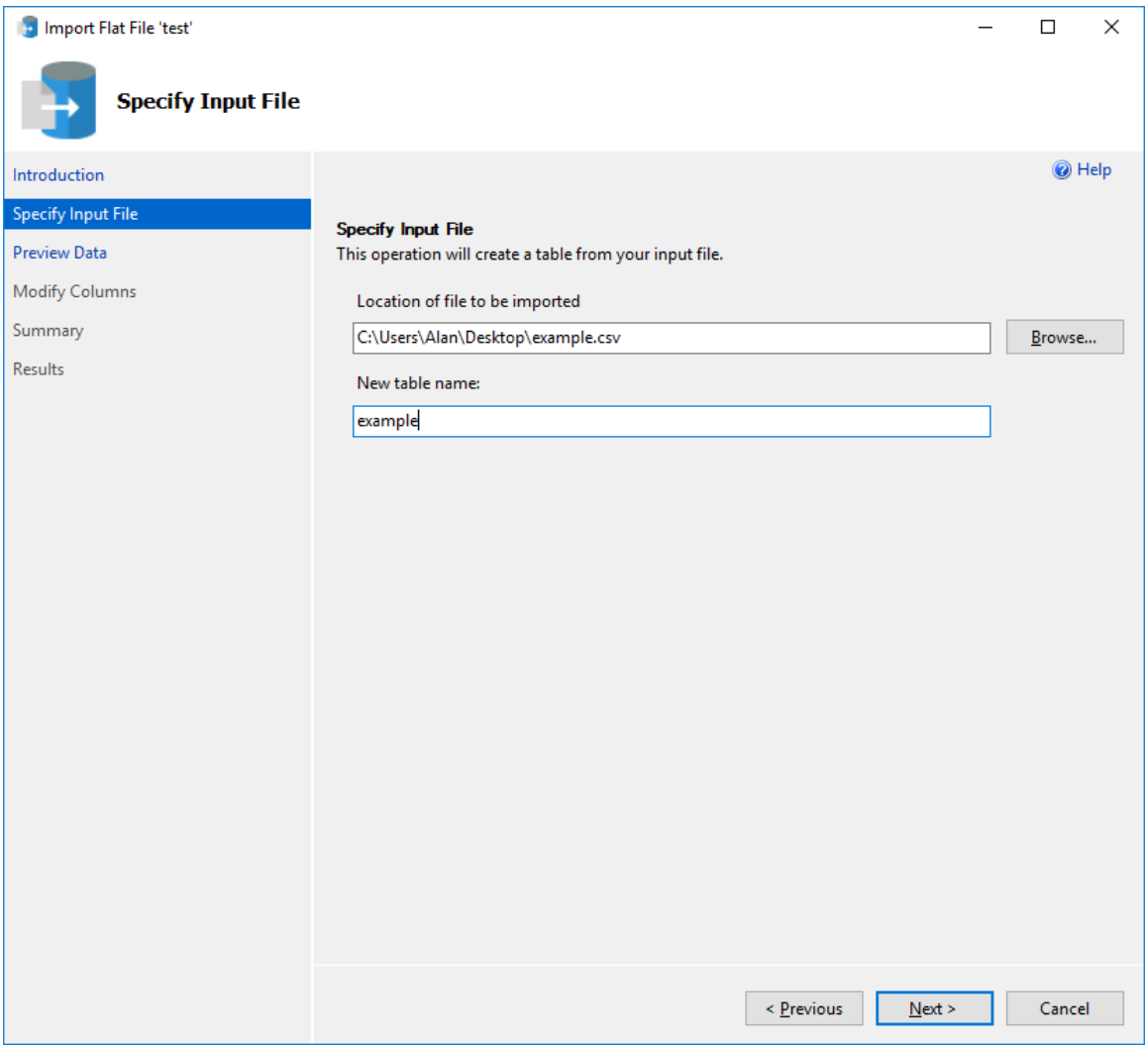


Рисунок 2.5 – Вибір файлу для імпортування даних

Після того як файл потрібно перевірити чи правильно сформувалася таблиця, перенеслися дані, а також у цієї утиліти є мінус, а саме це стосується самої бази даних від Microsoft (рисунок 2.6). Буває так, що вхідні дані мають тип, який на жаль не використовується або не підтримується базою даних, в результаті цього слід, що слід уважно вибирати базу даних при імпорті даних, або додавати файли, які мають структуровані дані із підтримуваними типами з бази даних.

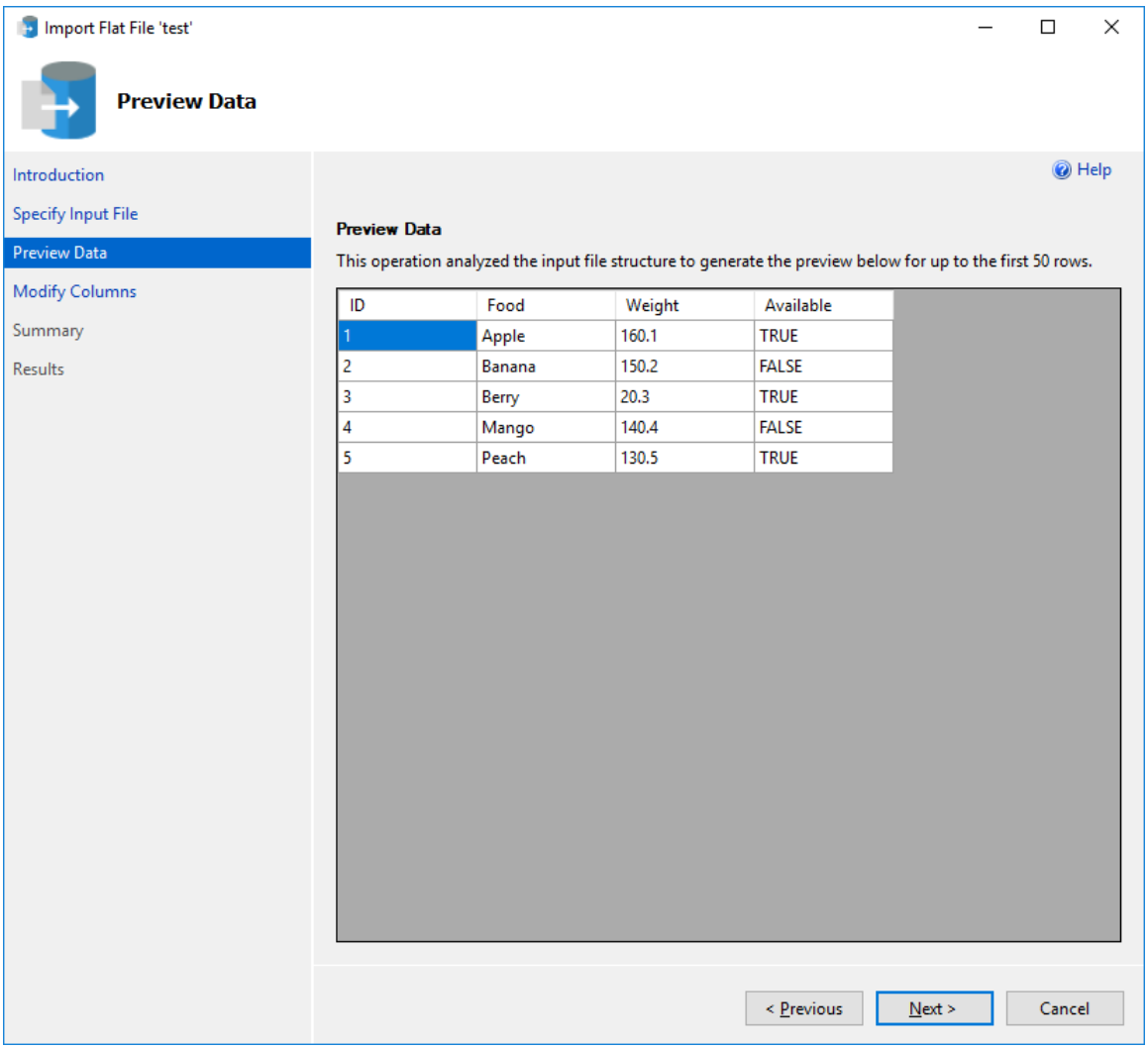


Рисунок 2.6 – Перевірка не структурованих даних

Якщо файл наблизився до етапу, коли потрібно налаштувати правильно колонки, а це вибрати тип, якісь буває колонки зайві (рисунок 2.7), слід поставитися до цього етапу серйозно, адже після цього буде йти коротка інформація про те, які дані будуть занесені, з якими типами, і змінити вже не можна, тут потрібно виявити особливу увагу.

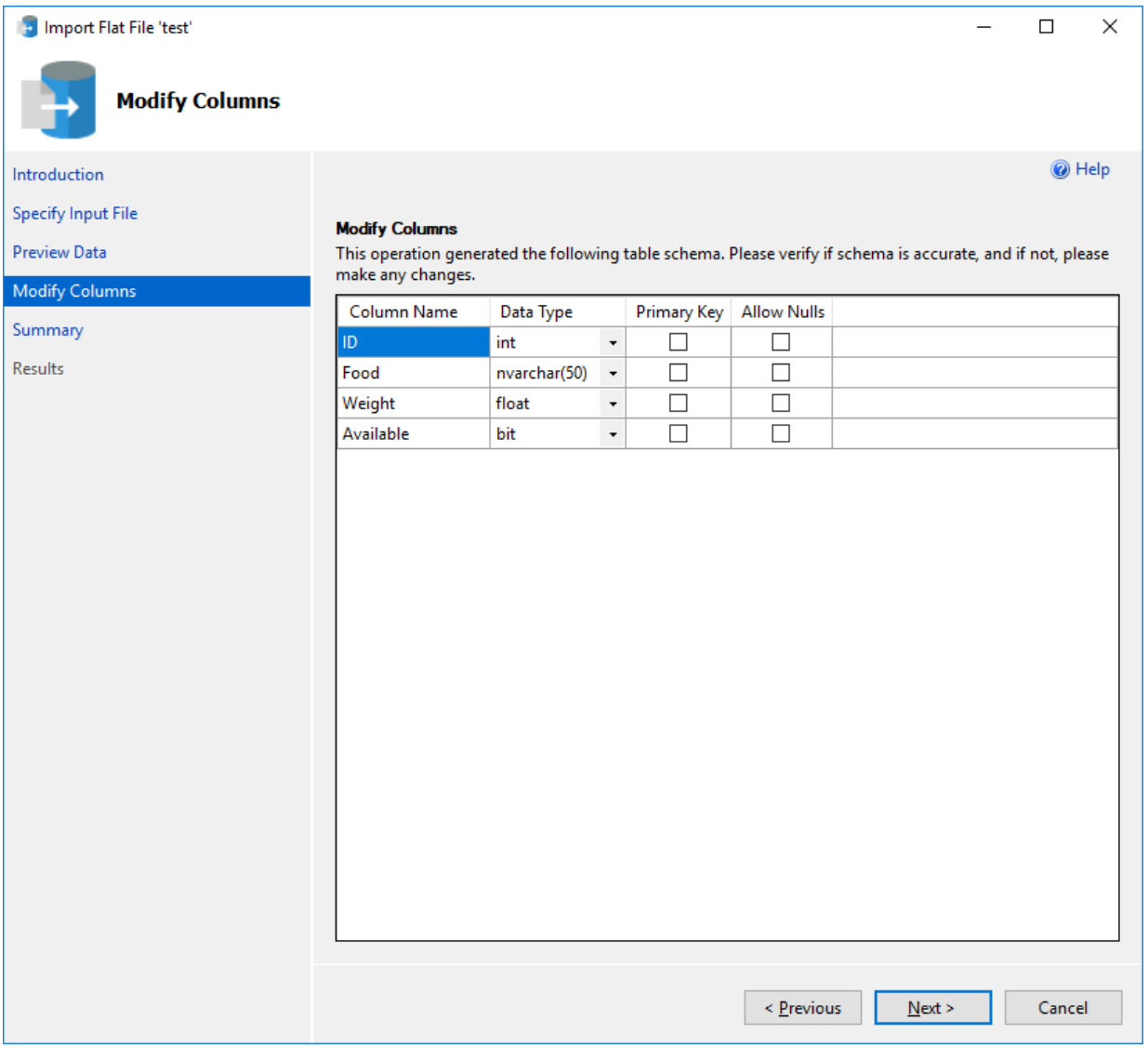


Рисунок 2.7 – Редагування нових колонок із вхідних даних

Це сторінка зведення, де відображається поточна конфігурація. Якщо виникли проблеми, можна повернутись до попередніх сторінок майстра. В іншому випадку натисніть кнопку "Готово", щоб розпочати імпорт (рисунок 2.8).

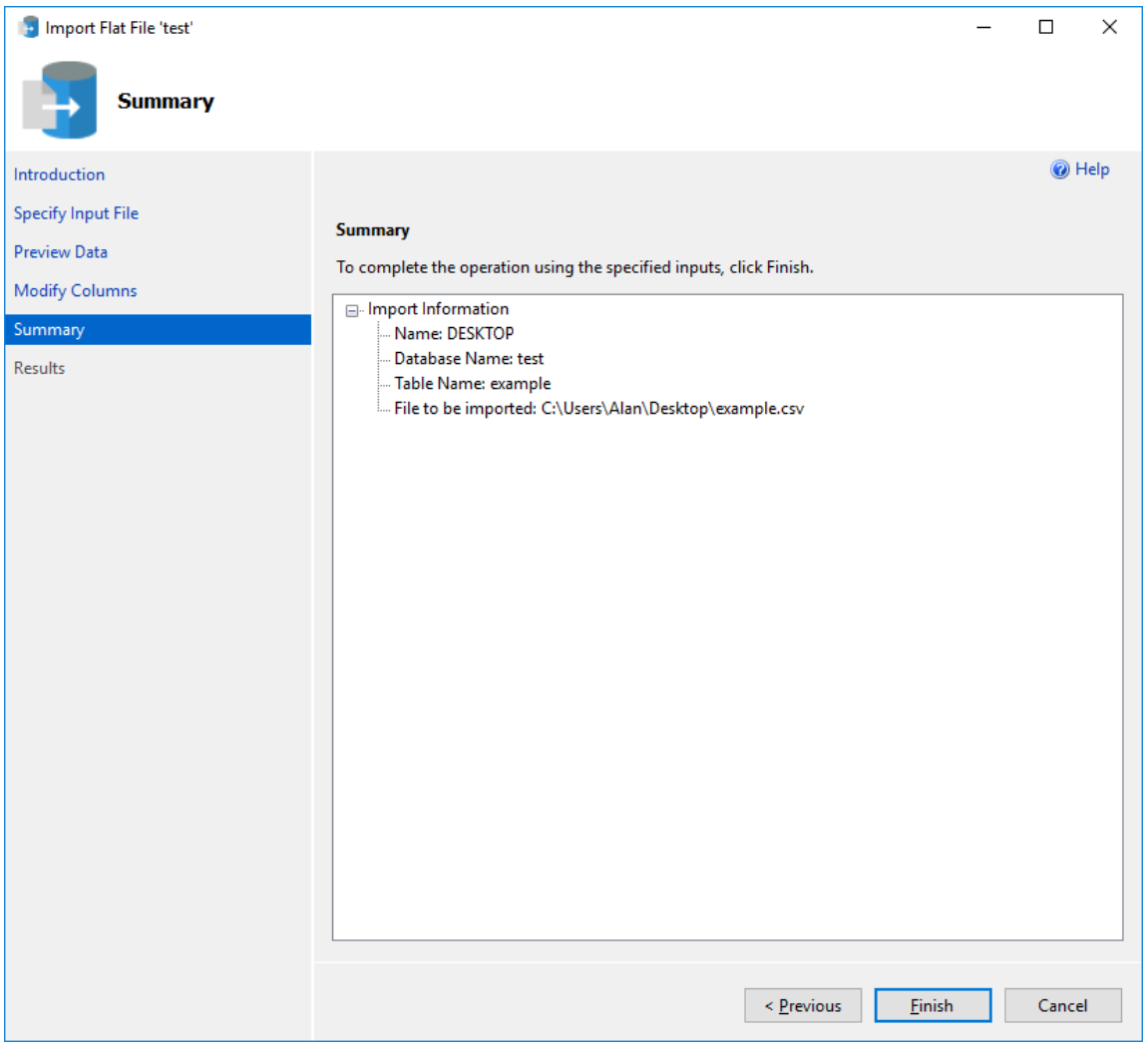


Рисунок 2.8 – Сторінка зведення інформацію щодо вхідних даних

То навіщо тоді створювати свій метод для парсингу, якщо є готовий? Якщо пояснити просто, дані потрібно фільтрувати перед тим, як відправити в базу, також дані повинні бути структуровані під таблиці, щоб не порушувати правила нормалізації. Використовуючи метод з автоматичною вставкою даних за допомогою MSSQL, дані будуть додані, але в хаотичному порядку, порушуючи структуру бази даних в цілому. Метод з утилітою слід використовувати у тому випадку, коли вхідні дані структуровані таким чином, щоб не порушувати основні принципи бази даних при формуванні структури таблиць [7].

В інших випадках, якщо вхідних даних багато, їх потрібно приводити до певної моделі через спеціально створені методи парсинг.

2.3 Метод генерації файлів

Методи генерації залежать від типу файлів, це пов'язано з тим, що найпоширеніший формат, як aslx, pdf, csv і т.д. Мають свої бібліотеки взаємодії, але це не одна проблема, крім кожного формату потрібно писати окремий алгоритм генерації. Варто уважно віднести до вибору бібліотеки для написання алгоритму. В першу чергу при виборі потрібно задаватися питанням, чи бібліотека вирішує поточне завдання пов'язане з генерацією файлу, завдання можуть бути такі як, вставка якихось форм, таблиць та інших елементів. Чи дозволяє бібліотека гнучко налаштовувати ці елементи.

2.3.1 Метод генерації документу із форматом docx за допомогою бібліотеки Microsoft.Office.Interop.Word

Ця бібліотека призначена для керування Word документами. Також один з плюсів цієї бібліотеки, що вона розроблена розробниками компанії Microsoft, що означає бібліотека буде підтримуватися і оновлюватися. Для додатків, які повинні підтримуватися роками, дуже важливо підтримувати оновлюваність, оскільки, якщо бібліотека застаріє, то доведеться переписувати величезну кількість коду.

Для генерації документа слід використовувати кілька основних функцій бібліотеки, але щоб їх використовувати перед цим потрібно створити об'єкт, який керуватиме станом файлу. Перший об'єкт, що відповідає за управління файлами з форматом docx, називається Interop.Word.Application, при його створенні відкривається можливість додавати файли. Цей об'єкт, щось на зразок звичного всіма Word, лише його механізм, функції відповідають за створення файлу, але не дозволить його гнучко налаштовувати.

Другий об'єкт Interop.Word.Document його можна отримати тільки після того, як створено об'єкт Application. Суть цього об'єкта - налаштування внутрішнього контенту файлу, такі як розмір шрифту, колір, межі, секції. Також цей об'єкт виконує одну з найважливіших функцій, має метод, який зберігає файл. У цьому об'єкті можна створювати різні форми, таблиці або параграфи. Наприклад, можна отримати об'єкт Interop.Word.Table, Interop.Word.Paragraph,

вони дозволять гнучко настроювати таблиці та інші елементи файлу docx.

Також слід враховувати, що після виконання всіх необхідних маніпуляцій проведених слід закривати підключення, що відкривається під час створення файлу. Це пов'язано з тим, що ресурси, які модифікують файл залишаються в пам'яті, навіть після всіх модифікацій. Може призвести до переповнення пам'яті, що вплине на швидкість виконання всього функціоналу. Нижче наведено код для генерації файлу [9].

```
Microsoft.Office.Interop.Word.Application winword = new Microsoft.Office.Interop.Word
.Application();
            winword.ShowAnimation = false;
            winword.Visible = false;
            object missing = System.Reflection.Missing.Value;

            Microsoft.Office.Interop.Word.Document document = winword.Documents.Add(r
ef missing, ref missing, ref missing, ref missing);

            foreach (Microsoft.Office.Interop.Word.Section section in document.Sectio
ns)
            {
                Microsoft.Office.Interop.Word.Range headerRange = section.Headers[Mic
rosoft.Office.Interop.Word.WdHeaderFooterIndex.wdHeaderFooterPrimary].Range;
                headerRange.Fields.Add(headerRange, Microsoft.Office.Interop.Word.WdF
ieldType.wdFieldPage);
                headerRange.ParagraphFormat.Alignment = Microsoft.Office.Interop.Word
.WdParagraphAlignment.wdAlignParagraphCenter;
                headerRange.Font.ColorIndex = Microsoft.Office.Interop.Word.WdColorIn
dex.wdBlue;
                headerRange.Font.Size = 30;
                section.Borders.Enable = 1;
                section.Borders.OutsideLineStyle = WdLineStyle.wdLineStyleSingle;
                section.Borders.OutsideLineWidth = WdLineWidth.wdLineWidth300pt;
                section.Borders.OutsideColor = WdColor.wdColorBlack;

            }

            document.Content.SetRange(0, 0);
            Microsoft.Office.Interop.Word.Paragraph para1 = document.Content.Paragrap
```

```
hs.Add(ref missing);
            object styleHeading1 = "Heading 1";
            para1.Range.set_Style(ref styleHeading1);
            para1.Range.Text = "Exam Dates";
            para1.Range.InsertParagraphAfter();

            Microsoft.Office.Interop.Word.Table firstTable = document.Tables.Add(para
1.Range, 1, 2, ref missing, ref missing);

            firstTable.Borders.Enable = 1;
            int intRow = 2;
            Object beforeRow = Type.Missing;
            string[] dates;
            while (reader.Read())
            {
                dates = reader[1].ToString().Split(' ');
                firstTable.Rows.Add(ref beforeRow);
                firstTable.Cell(intRow, 1).Range.Text = reader[0].ToString();
                firstTable.Cell(intRow, 2).Range.Text = dates[0];

                intRow += 1;
            }

            object filename = Environment.GetFolderPath(Environment.SpecialFolder.Des
ktop) + "\\Attachments\\Form.docx";
           document.SaveAs2(ref filename);
            ((Microsoft.Office.Interop.Word._Document)document).Close(ref missing, re
f missing, ref missing);
            ((Microsoft.Office.Interop.Word._Application)winword).Quit(ref missing, r
ef missing, ref missing);
```

2.3.2 Метод генерації документу із форматом docx за допомогою бібліотеки Aspose.Words

Aspose.Words - це кросплатформова бібліотека класів, яка дозволяє вашим програмам виконувати широкий спектр завдань з обробки документів.

Використання Aspose.Words у вашому проекті дає вам такі переваги:

– Багатий набір функцій;
– Незалежність від платформи;
– Незалежність від сторонніх додатків;
– Продуктивність і масштабованість;
– Мінімальна крива навчання.

На наступній діаграмі показано основні функції Aspose.Words і те, як вони пов’язані один з одним (рисунок 2.9).

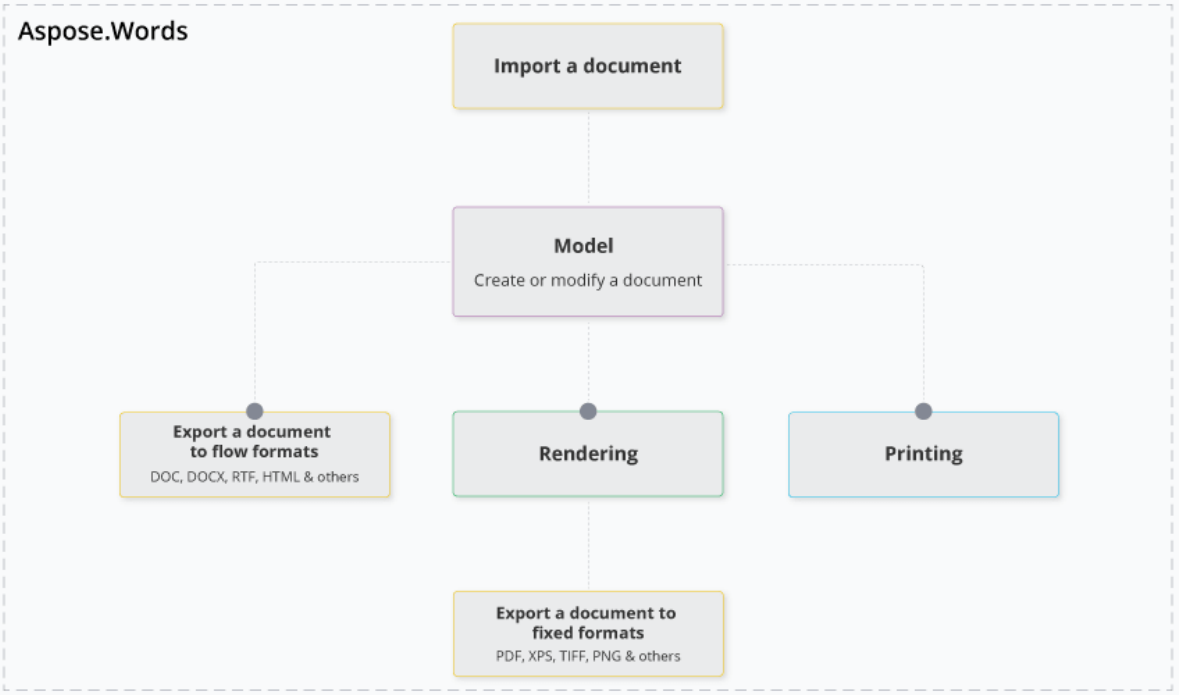


Рисунок 2.9 – Діаграма функцій та їх залежність Aspose.Words

Aspose.Words надає користувачам широкий спектр функцій. Користувачі можуть виконувати величезну кількість завдань, пов’язаних з документами – від простого перетворення документів з одного формату в інший і модифікації цих документів під час процесу перетворення до бізнес-завдань, таких як створення структурованих та візуально привабливих документів або автоматизація звітності.

Сучасні формати та стандарти документів є складними, і коли вам потрібно перетворення документів або інші функції обробки документів у вашому проекті, єдиним практичним рішенням часто є використання стороннього компонента, який реалізує потрібну функціональність. Але використання стороннього

компонента завжди несе певний ризик. Одним з унікальних ризиків при обробці документів є питання про те, наскільки повністю і правильно бібліотека реалізує певний формат або стандарт документа.

Aspose прагне забезпечити найбільш повну та точну реалізацію форматів і стандартів документів. Команда Aspose.Words демонструє свою прихильність до взаємодії, надаючи детальні примітки щодо впровадження підтримуваних форматів документів на кількох платформах [6].

2.4 Методи авторизації OAuth 2.0 та OpenID Connect

OAuth 2.0 — протокол авторизації, який дозволяє видати одному сервісу права доступу до ресурсу користувача на іншому сервісі. Протокол видає необхідний доступ до додатка логіна та пароля, а також дозволяє видавати обмежений набір прав, а не все відразу.

OpenID — відкритий стандарт децентралізованої системи аутентифікації, що надає користувацьку можливість створити єдиний учетний запис для аутентифікації на множини не пов'язаних із іншим інтернет-ресурсом, використовуючи послуги третіх осіб.

Третье покоління OpenID-технології, яке представляє собою аутентифікаційну надбудову над протоколом авторизації OAuth 2.0. OpenID Connect дозволяє інтернет-ресурсам перевірити особистість користувача на основі аутентифікації, виконаної авторизаційним сервером. Для роботи використовується RESTful API, описаний в специфікаціях. Також у OpenID Connect визначені додаткові механізми для надпочечного шифрування та цифрових підписів. Стандарт дозволяє використовувати додаткові можливості, такі як управління сесіями та виявлення OpenID-провайдерів.

OpenID призначений для аутентифікації — це є для того, щоб зрозуміти, що цей конкретний користувач є тем, ким представляється. Наприклад, за допомогою додатків OpenID некий сервіс може зрозуміти, що зайшов користувач, це саме він. При наступній аутентифікації сервіс зможе його дізнатися і зрозуміти, що, це той самий користувач, що і в попередньому разі.

OAuth — це протокол авторизації, він дозволяє видавати права на дії, які сам сервіс зможе виробляти від особи користувача. При цьому користувач після авторизації може взагалі не брати участь у процесі виконання дій, наприклад, сервіс зможе самостійно заливати фотографії в обліковому записі користувача [10].

Як і перша версія, OAuth 2.0 заснований на використанні базових веб-технологій: HTTP-запитах, редиректах тощо. для браузерів.

Ключова відмінність від OAuth 1.0 – простота. У новій версії немає громіздких схем підпису, скорочено кількість запитів, необхідних авторизації.

Загальна схема роботи програми, що використовує OAuth, така:

– отримання авторизації
– звернення до захищених ресурсів

Результатом авторизації є access token - якийсь ключ (зазвичай просто набір символів), пред'явлення якого є перепусткою до захищених ресурсів. Звертання до них у найпростішому випадку відбувається за HTTPS із зазначенням у заголовках або як один з параметрів отриманого access token'a.

У протоколі описано кілька варіантів авторизації, що підходять для різних ситуацій:

– авторизація для додатків, що мають серверну частину;
– авторизація для повністю клієнтських програм;
– авторизація за логіном та паролем;
– відновлення попередньої авторизації.

Стандарт визначає такі ролі:

– Resource Owner - користувач, який заходить на сайт і дає дозвіл використовувати свої закриті дані з Соцмережі.
– Client (він же сайт) - додаток або інтернет-сайт, яким користується користувач і який взаємодіє з Authorization Server і Resource Server для отримання закритих даних користувача.
– Authorization Server - сервер який перевіряє логін/пароль користувача, він же Соцмережа.

– Resource Server - зберігає закриту інформацію користувача, яку можна отримати за допомогою API. Authorization Server та Resource Server можуть бути поєднані в одну систему.

Тепер сам процес. Деталі конкретних реалізацій можуть відрізнятися, але загальна логіка завжди буде така (рисунок 2.10):

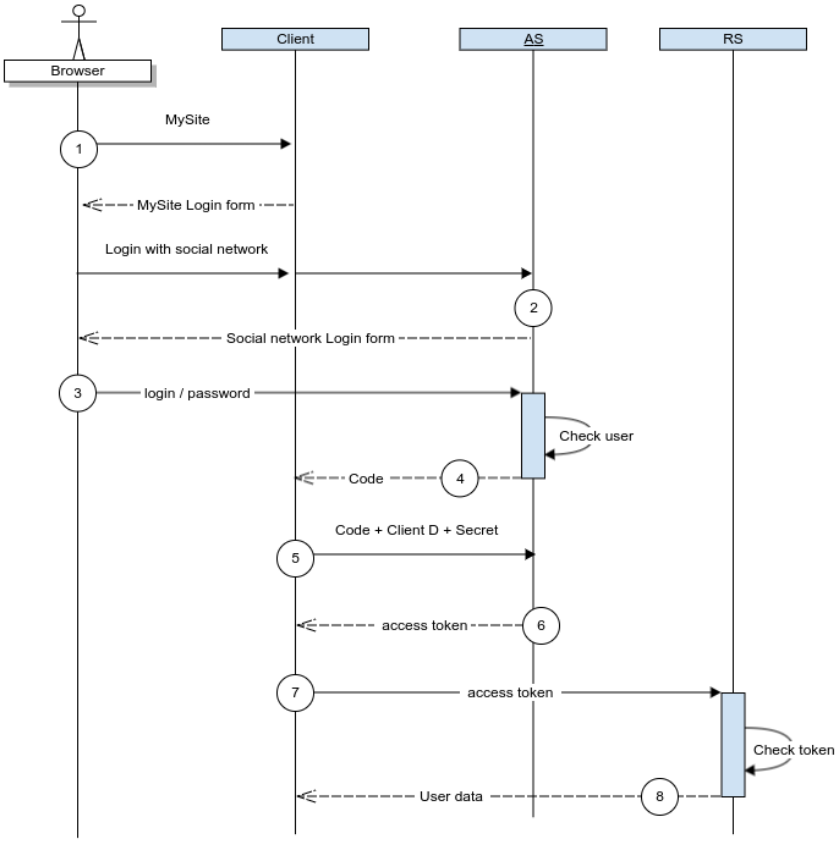


Рисунок 2.10 – Діаграма послідовності авторизації OAuth 2.0

Один із компонентів документообігу з цифровими файлами має мати змогу ідентифікувати юзера шляхом OAuth 2.0. Ця діаграма описує такі пункти:

1. Resource Owner (користувач, який хоче увійти до аккаунту) заходить на Client (додаток, один із компонентів документообігу), вибирає опцію "увійти в аккаунт ", сайт перенаправляє користувача в форму авторизації на Authorization Server (окремий сервіс який перевіряє сесію користувача та поля на формі, також зберігає різну інформацію щодо прав користувача).

2. Authorization Server перевіряє, чи є у користувача активна сесія (тут

перевіряється, чи авторизований користувач зараз у даний час) і, якщо ні, то показує форму для логіну.

3. Resource Owner (користувач) вводить свої логін/пароль і підтверджує, що певні закриті дані можуть бути використані з Client додатку, наприклад, ім'я користувача, паспортні дані або e-mail адресу.

4. Authorization Server перевіряє користувача та перенаправляє на адресу на Client (додаток на якому користувач використовую форму).

5. Callback з результатом аутентифікації та Authorization Code (це зашифрований рядок, в якому використовувалися закриті дані, такі як ім'я, прізвище та інші дані).

6. У відповідь Client (веб-додадок, один із компонентів документообігу) посилає “Authorization Code”, Client ID та Client Secret на Authorization Server.

7. Authorization Server перевіряє надіслані дані та формує access token у форматі JWT (JSON Web Token), підписаний своїм приватним ключем, JWT формується завдяки алгоритму SHA-256. У цьому ж JWT може міститися і "refresh token", за допомогою якого можливе відновлення сесії після закінчення.

8. Після цього Client може запросити закриту інформацію користувача за допомогою виклику Resource Server (основний сервер через який можна використати різні послуги та отримати чутливі дані такі як паспортні дані і так далі), який передається access token.

9. Resource Server перевіряє “access token” (наприклад, використовуючи відкритий ключ Authorization Server) та надає доступ до даних користувача і дає змогу використовувати різні послуги на додатку [11].

3 РОЗРОБКА МОДИФІКОВАНИХ КОМПОНЕНТІВ ДОКУМЕНТООБІГУ ІЗ ГЕНЕРАЦІЄЮ ЦИФРОВИХ ДОКУМЕНТІВ

3.1 Послідовність виконання запиту

Для відображення та розуміння всієї послідовності потрібно зробити діаграму (рисунок 3.1)

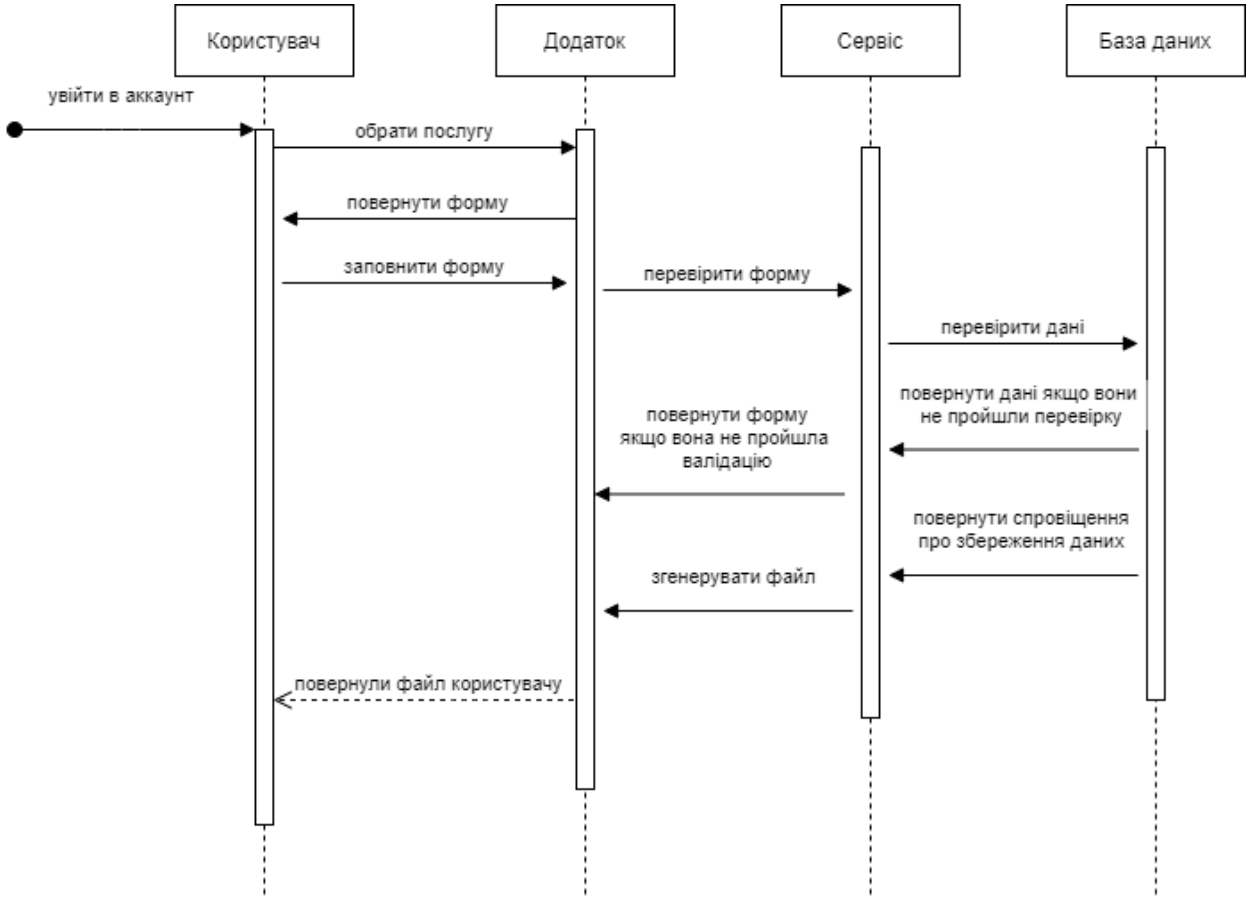


Рисунок 3.1 – Діаграма послідовности запиту

Діаграма послідовності поділяється на такі кроки описує звичайного користувача, який хоче використати послугу:

1. Користувачу треба обрати адміністративни послугу. Послуга може бути із різними компонентами документообігу такі як адміністративний центр тощо.

2. Після заповнення форми додаток відправить її в сервіс із послугами, який перевірить чи поля заповнені правильно. У разі провалу перевірки

користувач побачить, де він помилився і що потрібно змінити.

3. Після перевірки форми та всіх вхідних даних, база даних вставляє їх у таблицю та зберігає їх, дані можуть бути використовані у різних компонентах документообігу, це залежить від послуги в який компонент відправиться дані. У разі невдачі відправиться відповідь користувача, що щось пішло не так. Відповідь база даних надсилає як статус.

4. Після того, як сервіс отримав позитивну відповідь із бази даних, він запускає алгоритм генерації файлу та відправляє її додатку.

5. Додаток отримує згенерований файл який може бути цифровим документом та відображає його на екрані користувача. Користувач може вирішити завантажити документ чи ні.

Після розгляду діаграми послідовності компонентів документообігу, потрібен метод авторизації та генерації цифрового документу. Також додати можливість впровадження даних за допомогою парсингу, через те, що додавання даних користувачів та послуг в ручну займе велику кількість часу і уповільнить процеси розробки та оновлення програми.

3.2 Вибір інструмента реалізації

Для розробки програми було обрано платформу ASP.NET Core. Для створення додатка із компонентами документообігу в яких будуть генеруватися цифрові документи було обрано платформу ASP.NET. Для відповіді обґрунтування вибору відзначимо відмінні риси обраної платформи. Технологія ASP.NET є розвитком Active Server Page (ASP). Ця технологія є універсальною платформою для розробки веб-додатків корпоративного рівня. ASP.NET пропонує нову модель програмування та інфраструктуру, які дозволяють розробляти захищені та масштабовані рішення. Рішення реалізовано за допомогою патерну MVC 5. Концепція патерну (шаблону) MVC (model - view - controller) передбачає поділ програми на три компоненти:

– Контролер (controller) представляє клас, що забезпечує зв'язок між користувачем та системою, представленням та сховищем даних. Він отримує дані,

що вводяться користувачем, і обробляє їх. І в залежності від результатів обробки надсилає користувачеві певний висновок, наприклад, у вигляді уявлення.

– Подання (view) – це власне візуальна частина або інтерфейс програми користувача. Як правило, html-сторінка, яку користувач бачить зайшовши на сайт.

– Модель (model) є класом, що описує логіку використовуваних даних.

При такому підході модель є незалежним компонентом – будь-які зміни контролера або уявлення не зачіпають модель. Контролер и уявлення є щодо незалежними компонентами, і часто їх можна змінювати незалежно друг від друга.

Завдяки цьому реалізується концепція розподілу відповідальності, у зв'язку з чим легше побудувати роботу над окремими компонентами. Крім того, внаслідок цього додаток має кращу тестованість. І якщо нам, припустимо, важлива візуальна частина чи фронтенд, ми можемо тестувати уявлення незалежно від контролера. Або ми можемо зосередитися на бекенді та тестувати контролер. Конкретні реалізації та визначення даного патерну можуть відрізнятися, але через свою гнучкість і простоту він став дуже популярним останнім часом, особливо у сфері веб-розробки.

ASP.NET має такі функціональні можливості:

1. Простота розгортання. Розгортання ASP.NET додатків виконується шляхом копіювання файлів програми у спеціальну папку на веб-сервері. Перезапуск web-сервера не вимагається;

2. Засоби безпеки. Розробник ASP.NET може використовувати у своєму додатку будь-яку з пропонованих типових схем авторизації та аутентифікації користувачів;

3. Підтримка національних мов. ASP.NET використовує Unicode та розробники мають можливість застосовувати у своїх проекта національні алфавіти;

4. Висока продуктивність. ASP.NET має справу зі скомпілованимкодом. Завдяки цьому ASP.NET отримує можливість ефективно використовувати різні механізми оптимізації коду;

5. Підтримка мобільних пристроїв. ASP.NET підтримується будь-яким браузером, запущеним будь-якому пристрої (заява Microsoft);

6. Можливості налагодження. ASP.NET забезпечує можливість трасування та налагодження коду програм;

7. Інтеграція з .NET Framework. ASP.NET є частиною платформи .NET Framework. Розробники можуть використовувати можливості, що надаються цією платформою під час створення додатків;

ASP.NET містить безліч готових елементів керування, застосовуючи які, можна швидко створювати інтерактивні веб-програми. В загальному, можливості ASP.NET обмежені лише нашою уявою.

3.2.1 Розробка модифікованого методу генерації файлів із засобами Aspose.Words

Після розгляду та порівняння бібліотек у пунктах 2.2.1 та 2.2.2 вирішено було використовувати інструменти Aspose.Words тому що вона дозволяє швидко та ефективно конвертувати файли у той формат, який потрібен для зручного відображення користувачеві.

Об'єктна модель документа Aspose.Words (надалі називається DOM) є відображенням документів Word у пам'яті. DOM Aspose.Words може програмно читати, маніпулювати та змінювати вміст і формат документів Word. Розуміння структури та відповідних типів DOM є основою для гнучкого програмування за допомогою Aspose.Words, що дуже важливо. Наступний приклад документа Word та його структура показані нижче (рисунок 3.2):

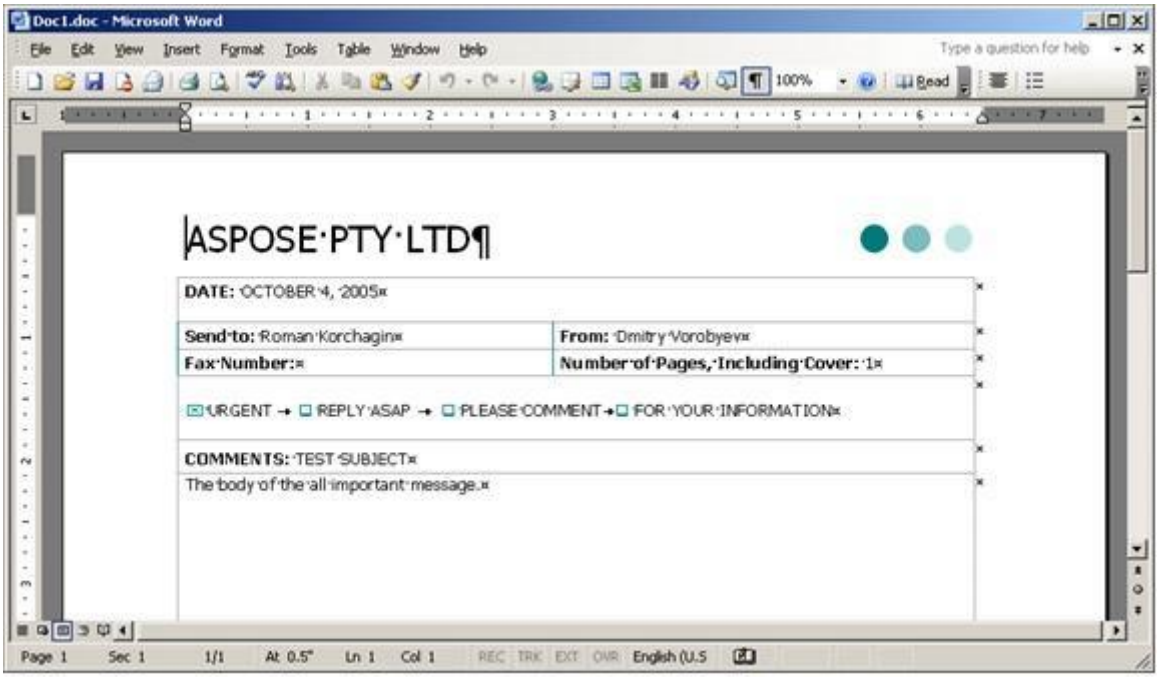


Рисунок 3.2 – Згенерований файл Aspose.Words

Коли вищезгаданий документ читається DOM Aspose.Words, створюється об'єкт дерева з наступною структурою (рисунок 3.3):

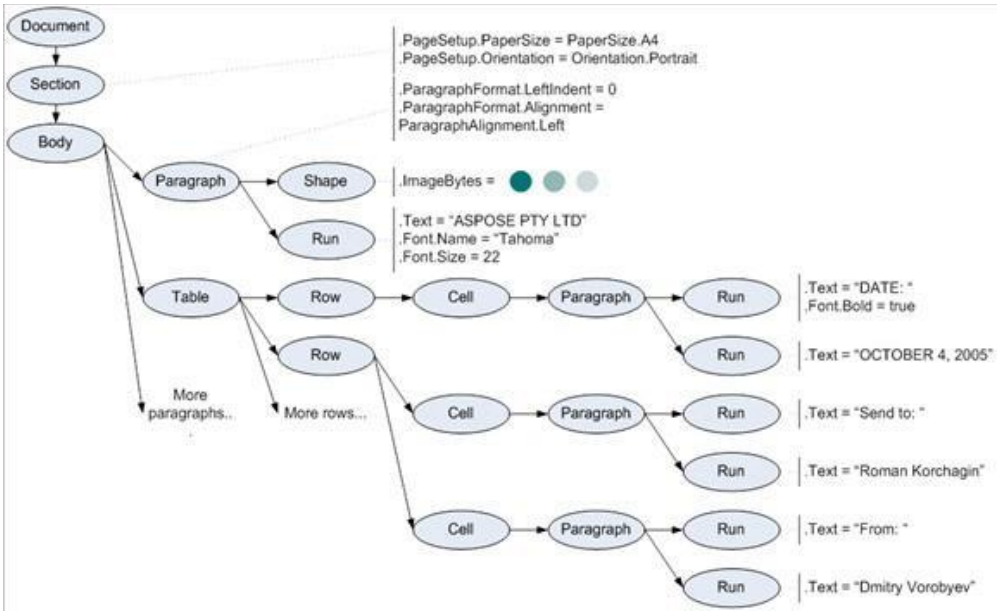


Рисунок 3.3 – Дерево генерації файла

Зі структури, наведеної вище, та відповідного документа Word можна побачити структуру пов'язаних об'єктів у DOM. За допомогою цих базових концепцій можна працювати з документом Word у самому процесі. Document,

Section, Paragraph, Table, Shape, Run та інші еліпси малюнку є об'єктами Aspose.Words. Ці об'єкти мають ієрархічну деревоподібну структуру. Коментарі на малюнку також показують, що об'єкти цих деревах об'єктів документа мають атрибутів [6].

DOM в Aspose.Words має такі характеристики:

1. Всі класи вузлів зрештою успадковуються від класу Node, який є базовим типом Aspose.WordsDOM.

2. Вузли можуть містити (вкладені) інші вузли, такі як Section та Paragraph, успадковані від класу CompositeNode, а також джерело класу CompositeNode та клас Node.

Коли Aspose.Words читає документ Word у пам'ять, різні типи елементів документа замінюються різними типами об'єктів. Текст, абзац, таблиця та розділ кожного текстового поля є об'єктами вузла, навіть сам документ є вузлом Aspose.Words визначає клас для кожного типу вузла документа.

Нижче наведена діаграма класів UML, що показує взаємозв'язок між різними типами вузлів DOM. Назва абстрактного класу виділено курсивом. Зверніть увагу, що Aspose.Words DOM також включає деякі невузлові типи, такі як Style, PageSetup, Font і т. д., які не показані на цьому рисунку (рисунок 3.4).

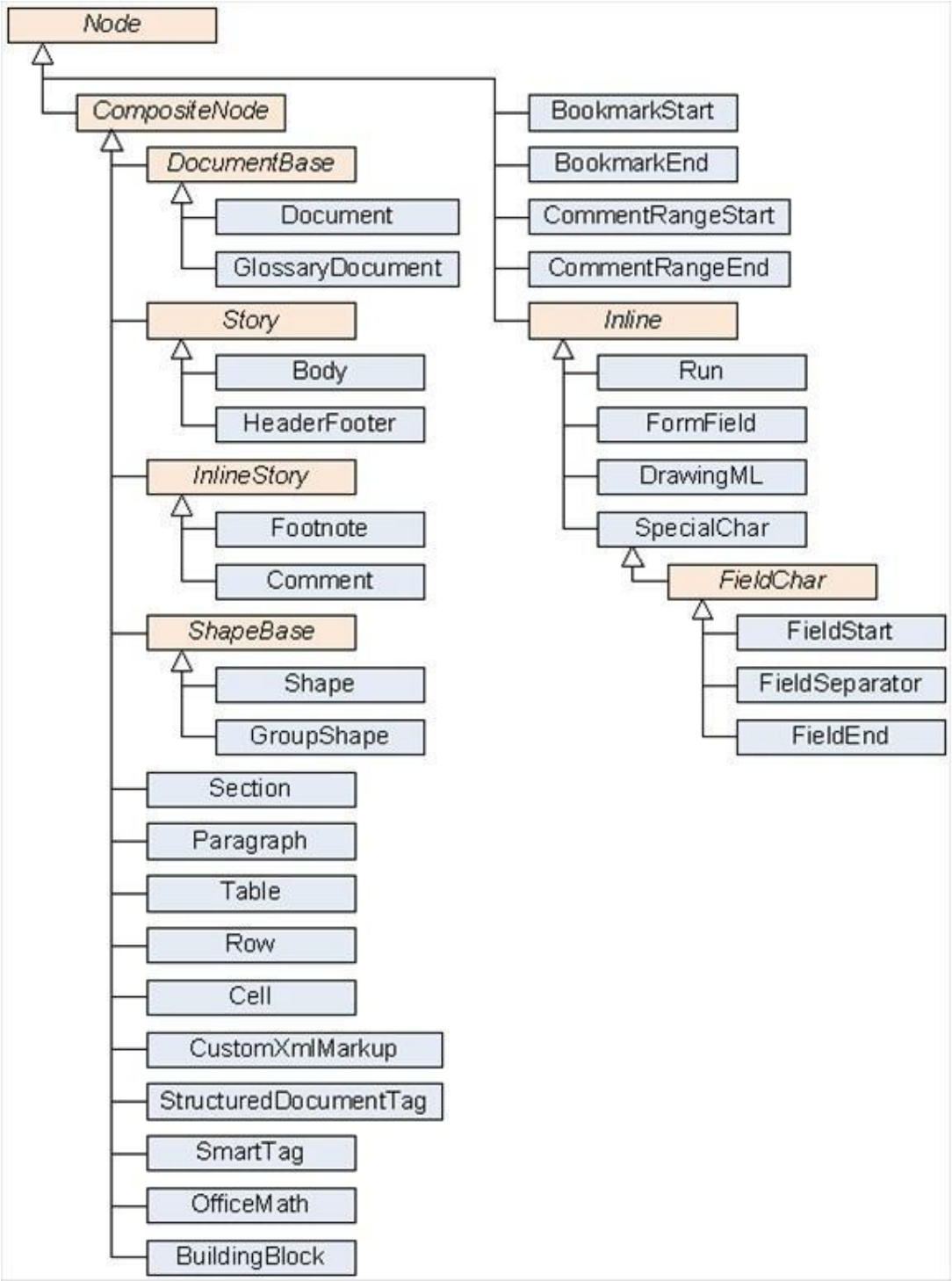


Рисунок 3.4 – Діаграма класів UML Aspose.Word

До батьківського вузла можна отримати доступ через властивість ParentNode класу Node, тому зручно отримати батьківський вузол. Об'єктна модель документа складається з великої кількості об'єктів, та їх взаємозв'язок така:

1. Клас Node є базовим класом усіх класів вузлів;

2. Клас CompositeNode є базовим класом для складових вузлів;

3. У класі Node немає інтерфейсу для управління дочірніми вузлами, а метод управління дочірніми вузлами з'являється лише у CompositeNode; [6].

3.2.2 Розробка модифікованого методу парсингу файлів is засобами Aspose.Cells

NuGet - це найпростіший спосіб завантажити та встановити Aspose.Cells для .NET (рисунок 3.5).

1. Відкрийте Microsoft Visual Studio та диспетчер пакетів NuGet.
2. Виконайте пошук «aspose.cells», щоб знайти потрібний Aspose.Cells для .NET.
3. Натисніть "Встановити", Aspose.Cells for .NET буде завантажено і на нього посилатиметься у вашому проекті.

Aspose.Cells повна свобода та гнучкість, щоб надати ваші електронні таблиці всім видам візуальних ефектів. API дозволяє використовувати різні шрифти з атрибутами, стилями осередків (вирівнювання, відступи, повороти, межі, затінювання та фарбування, захист, обтікання та стиснення тексту) та всі типи форматів чисел [8].

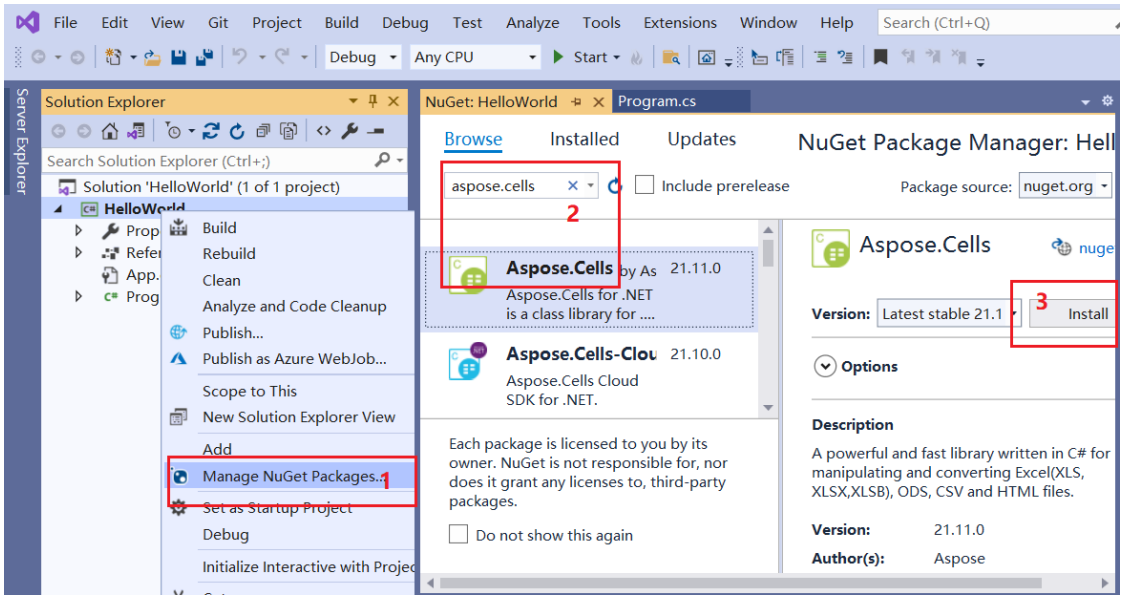


Рисунок 3.5 – Установка Aspose.Cells завдяки NuGet

Так само діаграми та графіки візуально привабливі та можуть надати вашій електронній таблиці професійний вигляд. Aspose.Cells для .NET надає повний набір API для створення всіх стандартних та настроюваних типів діаграм та управління ними. Крім того, клітини можна додавати різні типи об'єктів малювання, наприклад коментарі, зображення, об'єкти OLE, фігури та елементи управління.

Aspose.Cells для .NET надає елементи керування графічним інтерфейсом для веб- та настільних програм. Кінцевий користувач може використовувати ці елементи управління для відкриття, збереження та редагування файлів Excel, імпорту та експорту даних, управління форматуванням та формулами та перетворення між декількома підтримуваними форматами електронних таблиць прямо з інтерфейсу програми.

Aspose.Cells надає клас Workbook, який представляє файл Excel. Клас Workbook містить колекцію WorksheetCollection, яка забезпечує доступ до кожного листа у файлі Excel. Робочий лист представлений класом Worksheet. Клас Worksheet надає колекцію Cells, яка представляє всі осередки на аркуші.

Ми можемо використовувати колекцію Cells для доступу до осередків на аркуші. Aspose.Cells надає три основні підходи до доступу до осередків на аркуші:

1. Використовуючи ім'я комірки.
2. Використання індексу рядка та стовпця осередку.
3. Використання індексу комірки у колекції Cells.

Розробники можуть отримати доступ до будь-якої конкретної комірки, передавши її ім'я в колекцію Cells класу Worksheet як індекс.

Якщо ви створюєте порожній робочий лист на початку, кількість комірок у колекції дорівнює нулю. Коли ви використовуєте цей підхід для доступу до осередку, він перевіряє, чи існує ця осередок у колекції чи ні. Якщо так, він повертає об'єкт комірки в колекції, інакше створює новий об'єкт Cell, додає об'єкт у колекцію Cells, а потім повертає об'єкт. Цей підхід - найпростіший спосіб отримати доступ до осередку, якщо ви знайомі з Microsoft Excel, але він

найповільніший у порівнянні з іншими підходами.

3.2.2.1 Смарт маркери

Інтелектуальні маркери використовують для того, щоб Aspose.Cells знав, яку інформацію розмістити в електронній таблиці конструктора Microsoft Excel. Інтелектуальні маркери дозволяють створювати шаблони, що містять лише певну інформацію та форматування.

Електронні таблиці конструктора – це стандартні файли Excel, що містять візуальне форматування, формули та інтелектуальні маркери. Вони можуть містити інтелектуальні маркери, які посилаються на одне або кілька джерел даних, таких як інформація з проекту та інформація для зв'язаних контактів. Розумні маркери записуються в осередки, у яких ви хочете отримати інформацію.

Всі розумні маркери починаються з & =. Приклад маркера даних є & = Party.FullName. Якщо маркер даних призводить до більш ніж одного елемента, наприклад, до повного рядка, то наступні рядки автоматично переміщуються донизу, щоб звільнити місце для нової інформації. Таким чином, проміжні підсумки та підсумки можуть бути поміщені в рядок одразу після маркера даних для виконання обчислень на основі вставлених даних. Використовуйте динамічні формули, щоб робити розрахунки за вставленими рядками.

Інтелектуальні маркери складаються з частин джерела даних та імені поля для більшої інформації. Спеціальна інформація також може передаватися зі змінними та масивами змінних. Змінні завжди заповнюють лише одну комірку, тоді як масиви змінних можуть заповнювати кілька. Використовуйте лише один маркер даних на комірку. Невикористані смарт-маркери видаляються.

Смарт-маркер може також містити параметри. Параметри дозволяють змінювати спосіб розміщення інформації. Вони додаються в кінець розумного маркера у круглих дужках у вигляді списку, розділеного комами.

3.2.2.2 Таблиці

Однією з переваг електронних таблиць є те, що вони дозволяють

створювати різні типи списків, наприклад, списки телефонів, списки завдань, списки транзакцій, активів або зобов'язань. Декілька користувачів можуть працювати разом, щоб використовувати, створювати та підтримувати різні списки. Aspose.Cells підтримує створення списків та керування ними.

Перетворення списку даних на реальний об'єкт списку дає чимало переваг:

– Нові рядки та стовпці додаються автоматично.

– Рядок підсумків унизу списку можна легко додати для відображення SUM, AVERAGE, COUNT і т.д.

– Стовпці, додані праворуч, автоматично включаються до об'єкту List.

– Діаграми на основі рядків та стовпців розгортатимуться автоматично.

– Іменовані діапазони, надані рядкам і стовпцям, будуть розширені автоматично.

– Список захищений від випадкового видалення рядків та стовпців.

Aspose.Cells надає клас Workbook, який представляє файл Microsoft Excel. Клас Workbook містить колекцію Worksheets, яка забезпечує доступ до кожного аркуша у файлі Excel.

Робочий лист представлений класом Worksheet. Клас Worksheet надає широкий спектр властивостей та методів для керування робочим листом. Щоб створити ListObject на аркуші, використовуйте властивість колекції ListObjects класу Worksheet. Кожен ListObject, власне, є об'єктом класу ListObjectCollection, який додатково надає метод Add додавання об'єкта List і вказівки діапазону осередків для списку.

Згідно з вказаним діапазоном осередків об'єкт List створюється Aspose.Cells. Використовуйте атрибути (наприклад, ShowTotals, ListColumns тощо) Класу ListObject для керування списком.

У наведеному нижче прикладі ми створили той самий об'єкт ListObject за допомогою API Aspose.Cells, який ми створили за допомогою Microsoft Excel у попередньому розділі.

3.2.2.3 Плагин

Плагін Aspose Visual Studio - чудовий інструмент для швидкого завантаження та вивчення прикладів Aspose.Cells для .NET API. Це заощаджує багато часу та зусиль, надаючи дуже простий варіант безперешкодного вибору, завантаження та відкриття останніх прикладів проектів, не залишаючи Visual Studio.

Ця версія надає такі можливості:

– Підтримує Visual Studio 2010, Visual Studio 2012 та Visual Studio 2013.

– Легко запустити з меню File> Aspose у Visual Studio

– Економить час та скорочує час навчання.

– Дозволяє вибрати та відкрити останні приклади Aspose.Cells для .NET API, розміщені на Github.com.

– Дозволити вам відкрити проект прикладів C # або VB.NET

– Вибраний проект автоматично відкривається у Visual Studio з усіма необхідними посиланнями, щоб дати вам готове до запуску та дослідження середовище.

– Автоматично завантажує новітні бібліотеки API з NuGet та приклади з GitHub.

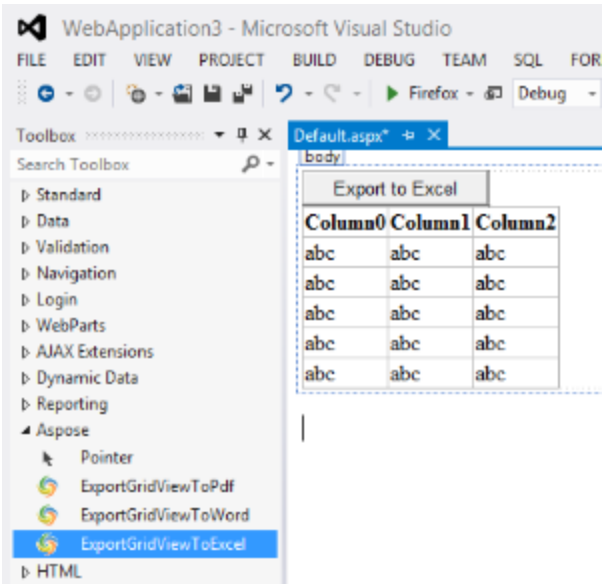


Рисунок 3.6 – Плагін Aspose.Cells для Visual Studio середовища

3.2.3 Розробка модифікованого методу авторизації файлів із засобами Identity Server

IdentityServer – це популярна платформа OSS OpenID Connect та OAuth для ASP.NET Core. Він дозволяє вашому веб-сайту ASP.NET Core виступати як постачальник OpenID і сервер авторизації OAuth, пропонуючи як єдиний вхід (SSO), так і захист API з коробки. IdentityServer дбає про підтримку протоколу, але автентифікація користувача залежить від вас, розробника.

Після випуску IdentityServer4 IdentityServer3 перейшов у режим обслуговування, і було випущено лише виправлення безпеки. Однак у 2019 році Microsoft припинила підтримку бібліотек OWIN, на які покладався IdentityServer3, і в результаті безкоштовна підтримка IdentityServer3 припинилася.

Якщо ви все ще використовуєте IdentityServer3, документація рекомендує вам якомога швидше перейти на останню версію IdentityServer.

IdentityServer3 та IdentityServer4 в основному сумісні. Зазвичай вони обидва реалізують одні й ті ж специфікації, вони є провайдерами OpenID, проте їх внутрішній пристрій майже повністю відрізняється. На щастя, при інтеграції з використанням OpenID Connect або OAuth, у випадку IdentityServer, ви не інтегруєтеся в реалізацію, а швидше інтегруєте з використанням специфікацій OpenID Connect або OAuth. На мій досвід, IdentityServer – одна із найкращих технологій для реалізації цих специфікацій.

IdentityServer розроблений для роботи як самостійний компонент, чого було важко досягти за допомогою ASP.NET 4.x через те, що MVC, як і раніше, тісно пов'язаний з IIS і System.Web. Це спричинило те, що IdentityServer3 мав внутрішній механізм перегляду, обслуговуваний компонентом katana. Тепер, коли IdentityServer4 працює на ASP.NET Core, ви можете використовувати будь-яку технологію інтерфейсу користувача, яку хочете, і розміщувати IdentityServer в будь-якому місці, де може працювати ASP.NET Core. Це також означає, що тепер ви можете інтегруватись з існуючими формами або системами входу в систему, що дозволяє виконувати оновлення на місці.

IdentityServer IUserService, який використовувався для інтеграції вашого сховища користувача, також зник, його замінила нова абстракція сховища користувача у формі IProfileService і IResourceOwnerPasswordValidator. Тепер ви

повинні самостійно продати аутентифікацію користувача (і це добре).

З 2021 року IdentityServer перейшов на комерційну ліцензію і тепер відомий як Duende IdentityServer (різновид IdentityServer v5). Duende IdentityServer, як і раніше, є OSS, але тепер для отримання ліцензії більшості організацій потрібно придбати ліцензію у Duende.

IdentityServer4, як і раніше, підтримуватиметься протягом усього життєвого циклу .NET Core 3.1, який закінчується у грудні 2022 року. Однак для нових версій .NET вам слід використовувати Duende IdentityServer.

На момент написання кодові бази IdentityServer4 та Duende IdentityServer в основному збігалися, але згодом це зміниться, особливо у міру того, як .NET випускає більше LTS-версій. В даний час нові функції Duende включають управління ключами, динамічне завантаження зовнішніх постачальників посвідчень, індикатори ресурсів OAuth та доступ до клієнтської бібліотеки BFF.

Ви збираєтеся створювати свій IdentityServer як порожню веб-програму без будь-яких залежностей MVC або Razor або будь-якої аутентифікації. Ви можете створити це за допомогою команди dotnet new CMD.

Потім перейдіть до свого класу Startup, де ви можете почати реєструвати залежності та підключати свій конвеєр.

У методі ConfigureServices додайте наступне, щоб зареєструвати мінімально необхідні залежності для IdentityServer (рисунок. 3.7):

```
services.AddIdentityServer()
    .AddInMemoryClients(new List<Client>())
    .AddInMemoryIdentityResources(new List<IdentityResource>())
    .AddInMemoryApiResources(new List<ApiResource>())
    .AddInMemoryApiScopes(new List<ApiScope>())
    .AddTestUsers(new List<TestUser>())
    .AddDeveloperSigningCredential();
```

Рисунок 3.7 – Реєстрація сервісу у методі ConfigureServices

За допомогою наведеного вище коду ви зареєстрували IdentityServer у

своєму контейнері DI за допомогою AddIdentityServer, використовували демонстраційний сертифікат підпису за допомогою AddDeveloperSigningCredential та використовували енергозалежні сховища у пам'яті для ваших клієнтів, ресурсів та користувачів. Вам не потрібно викликати AddDeveloperSigningCredential під час використання Duende IdentityServer, оскільки IdentityServer керуватиме ключами підпису за вас.

Використовуючи AddIdentityServer, ви також змушуєте всі згенеровані тимчасові дані зберігатись у пам'яті. Незабаром ви додасте реальних клієнтів, ресурсів та користувачів.

3.3 Реалізація модифікованих методів

За допомогою Visual Studio потрібно розпочати створення проекту та реалізацію модифікованих алгоритмів(рисунок 3.8).

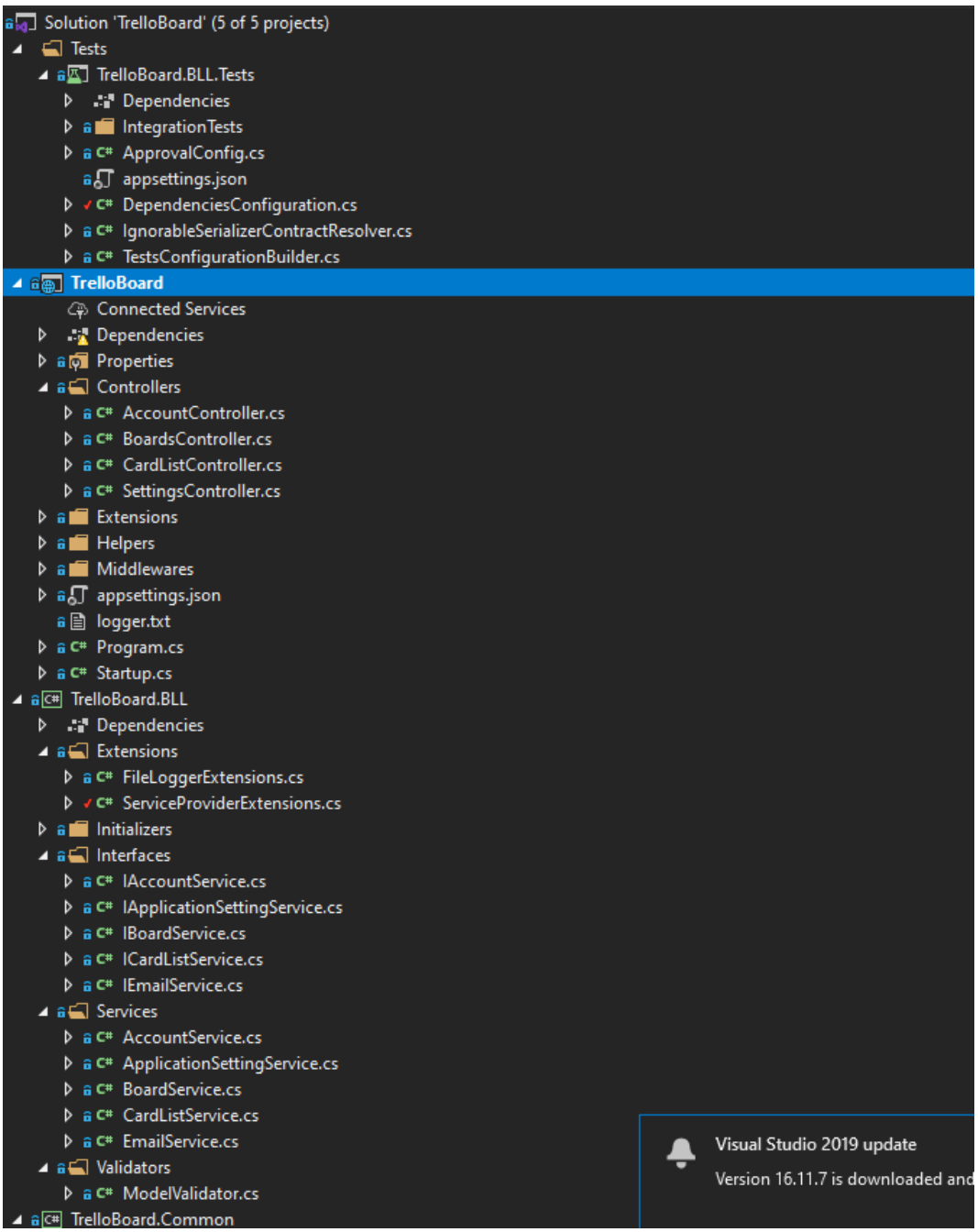
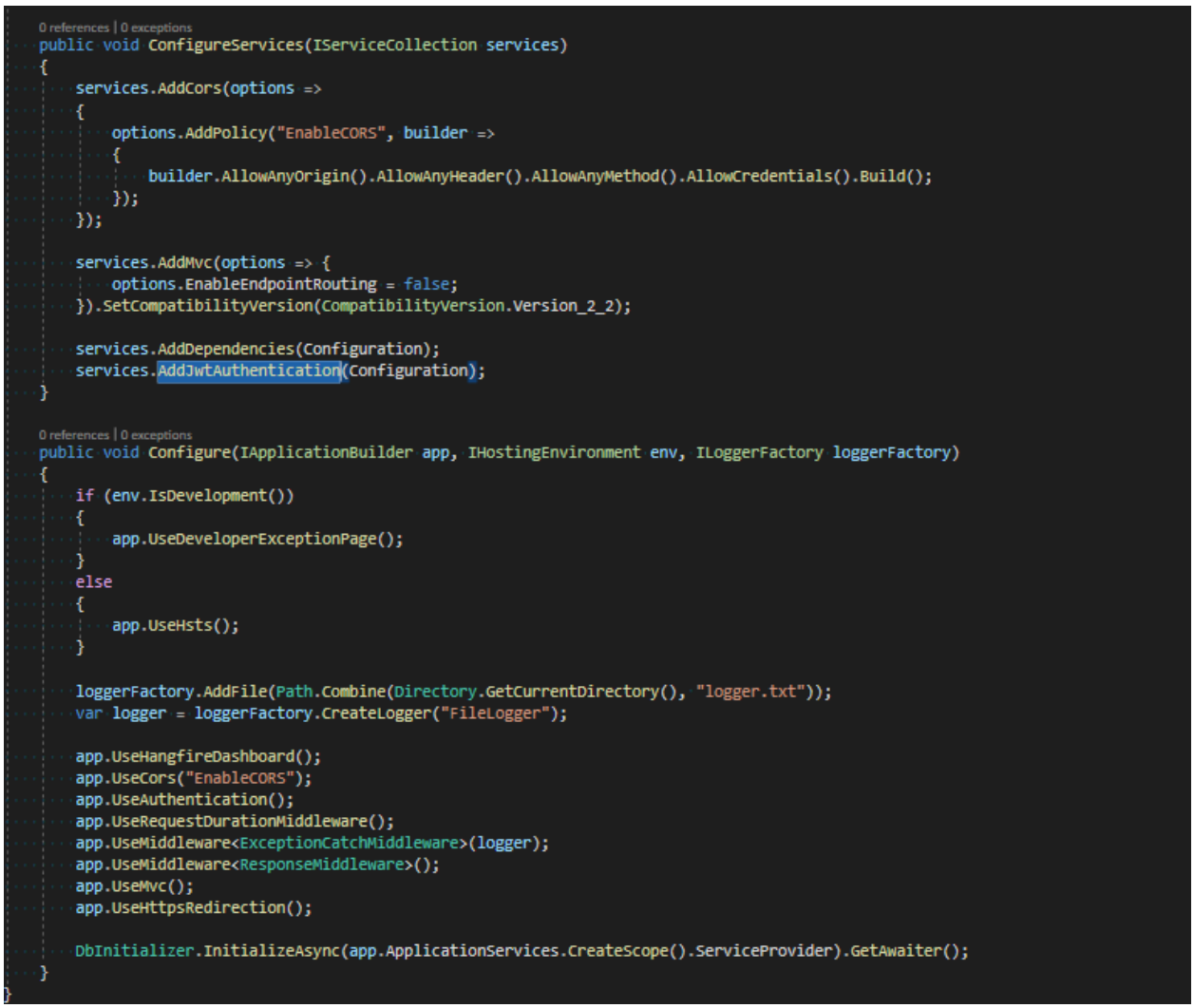


Рисунок 3.8 – Створення проекту та наповнення його контентом

Для того щоб використовувати IdentityServer4, потрібно, як було зазначено вище, налаштувати його у двох конфігураційних методах, які були згадані в пункті 3.2.3 (рисунок 3.9).



```
0 references | 0 exceptions
public void ConfigureServices(IServiceCollection services)
{
    services.AddCors(options =>
    {
        options.AddPolicy("EnableCORS", builder =>
        {
            builder.AllowAnyOrigin().AllowAnyHeader().AllowAnyMethod().AllowCredentials().Build();
        });
    });

    services.AddMvc(options => {
        options.EnableEndpointRouting = false;
    }).SetCompatibilityVersion(CompatibilityVersion.Version_2_2);

    services.AddDependencies(Configuration);
    services.AddJwtAuthentication(Configuration);
}

0 references | 0 exceptions
public void Configure(IApplicationBuilder app, IHostingEnvironment env, ILoggerFactory loggerFactory)
{
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
    }
    else
    {
        app.UseHsts();
    }

    loggerFactory.AddFile(Path.Combine(Directory.GetCurrentDirectory(), "logger.txt"));
    var logger = loggerFactory.CreateLogger("FileLogger");

    app.UseHangfireDashboard();
    app.UseCors("EnableCORS");
    app.UseAuthentication();
    app.UseRequestDurationMiddleware();
    app.UseMiddleware<ExceptionCatchMiddleware>(logger);
    app.UseMiddleware<ResponseMiddleware>();
    app.UseMvc();
    app.UseHttpsRedirection();

    DbInitializer.InitializeAsync(app.ApplicationServices.CreateScope().ServiceProvider).GetAwaiter();
}
}
```

Рисунок 3.9 – Додавання IdentityServer4 та других сервісів

Реалізація методу генерація JWT токену для OAuth 2.0 за допомогою IdentityServer4 (рисунок 3.10).

```
1 reference | 0 exceptions
private async Task<object> GenerateJwtTokenAsync(User user)
{
    var roles = await _userManager.GetRolesAsync(user);

    var claims = new List<Claim>
        {
            new Claim(JwtRegisteredClaimNames.Sub, user.Email),
            new Claim(JwtRegisteredClaimNames.Jti, Guid.NewGuid().ToString()),
            new Claim(JwtRegisteredClaimNames.Sid, user.Id.ToString()),
            new Claim(JwtRegisteredClaimNames.Typ, string.Join(", ", roles))
        };

        var key = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(_configuration["JwtKey"]));
        var creds = new SigningCredentials(key, SecurityAlgorithms.HmacSha256);
        var expires = DateTime.Now.AddDays(Convert.ToDouble(_configuration["JwtExpireDays"]));

        var token = new JwtSecurityToken(
            _configuration["JwtIssuer"],
            _configuration["JwtIssuer"],
            claims,
            expires: expires,
            signingCredentials: creds
        );

        return new { AccessToken = new JwtSecurityTokenHandler().WriteToken(token) };
}
```

Рисунок 3.10 – Генерація JWT токену за допомогаю IdentityServer4

Так як потрібно зробити оптимізоване заповнення бази даних, я розробив метод для обробки файлів з xslx форматом. Завдяки цьому методу відбувається обробка та структурування вхідних даних.

Для парсингу даних був створений модифікований метод парсингу (рисунок 3.11)

```
public static string ExportExcel(DataTable dt, string filename, List<string> listHeader)
        {
            try
            {
                                Aspose.Cells.Workbook workbook = new Aspose.Cells.Workbook (); // Рабоч
                                Aspose.Cells.Worksheet sheet = workbook.Worksheets [0]; // Рабочий лист
                                Aspose.Cells.Cells cells = sheet.Cells; // Ячейка
                Style style;
                for (int j = 0; j < dt.Columns.Count; j++)
                {
                    cells[0, j].PutValue(listHeader[j]);

                    style = cells[0, j].GetStyle();
                    style.BackgroundColor = Color.Blue;
                    style.ForegroundColor = System.Drawing.Color.FromArgb(153, 204, 0);
                    style.Pattern = BackgroundType.Solid;
                    cells[0, j].SetStyle(style);
                }
                for (int i = 0; i < dt.Rows.Count; i++)
                {
                    for (int j = 0; j < dt.Columns.Count; j++)
                    {
                        cells[i + 1, j].PutValue(dt.Rows[i][j].ToString());
                    }
                }

                sheet.AutoFitColumns();
                cells.SetRowHeight(0, 30);
                workbook.Save(filename);

                return "";
            }
            catch (Exception ex)
            {
                return ex.ToString();
            }
        }
```

Рисунок 3.12 – Модифікований метод парсингу даних

За допомогою бібліотеки Aspose.Words, можна генерувати різні файли на основі інформації з бази даних. Це дуже гнучка бібліотека, яка реалізую не тільки взаємодію між Word та кодом, але й конвертацію у різні формати (рисунок 3.13).

```
public void FileGeneration ()
    {
        // ДОСТУП ДО ДАНИХ
        Aspose.Words.Document doc = new Aspose.Words.Document();
        DocumentBuilder builder = new DocumentBuilder(doc);

        builder.InsertCell();
        builder.CellFormat.VerticalMerge = CellMerge.First;
        builder.Write("Text in merged cells.");

        builder.InsertCell();
        builder.CellFormat.VerticalMerge = CellMerge.None;
        builder.Write("Text in one cell");
        builder.EndRow();

        builder.InsertCell();
        builder.CellFormat.HorizontalMerge = CellMerge.First;
        builder.Write("Text in merged cells.");

        // ГЕНЕРАЦІЯ ФАЙЛА
        builder.InsertCell();
        builder.CellFormat.HorizontalMerge = CellMerge.Previous;
        builder.EndRow();

        builder.InsertCell();
        builder.CellFormat.HorizontalMerge = CellMerge.None;
        builder.Write("Text in one cell.");

        builder.InsertCell();
        builder.Write("Text in another cell.");
        builder.EndRow();
        builder.EndTable();

        // ВІДПРАВЛЕННЯ ФАЙЛУ
        doc.MailMerge.Execute(dataReader);
        dataReader.Close();
        conn.Close();

        doc.Save(MyDir + "MailMerge.ExecuteDataReader Out.doc");
    }
```

Рисунок 3.13 – Модифікований метод генерації файлу

В данному випадку платформа повертається файл кожен раз коли форма відправляється до сервісу. При поверненні файлу користувачу ми можемо конвертити файл у різні формати та будувати нові структури файлів. Таким чином зникає необхідність постійного створення нових сервісів, замість цього одного генератора, який довзоляє використовувати різні способи конвертації та генерації файлу.

Даний спосіб дозволяє сильно підняти продуктивність додатку, адже відпала потреба в ресурсозатратних сервісах та операціях створення та конвертації файлу. Особливо ефективним даний модифікований метод буде при потребі генерації відносно великих форм.

Для відображення зовнішньої частини було створено проект із гнучким налаштуванням структури DOM у HTML (рисунок 3.14).

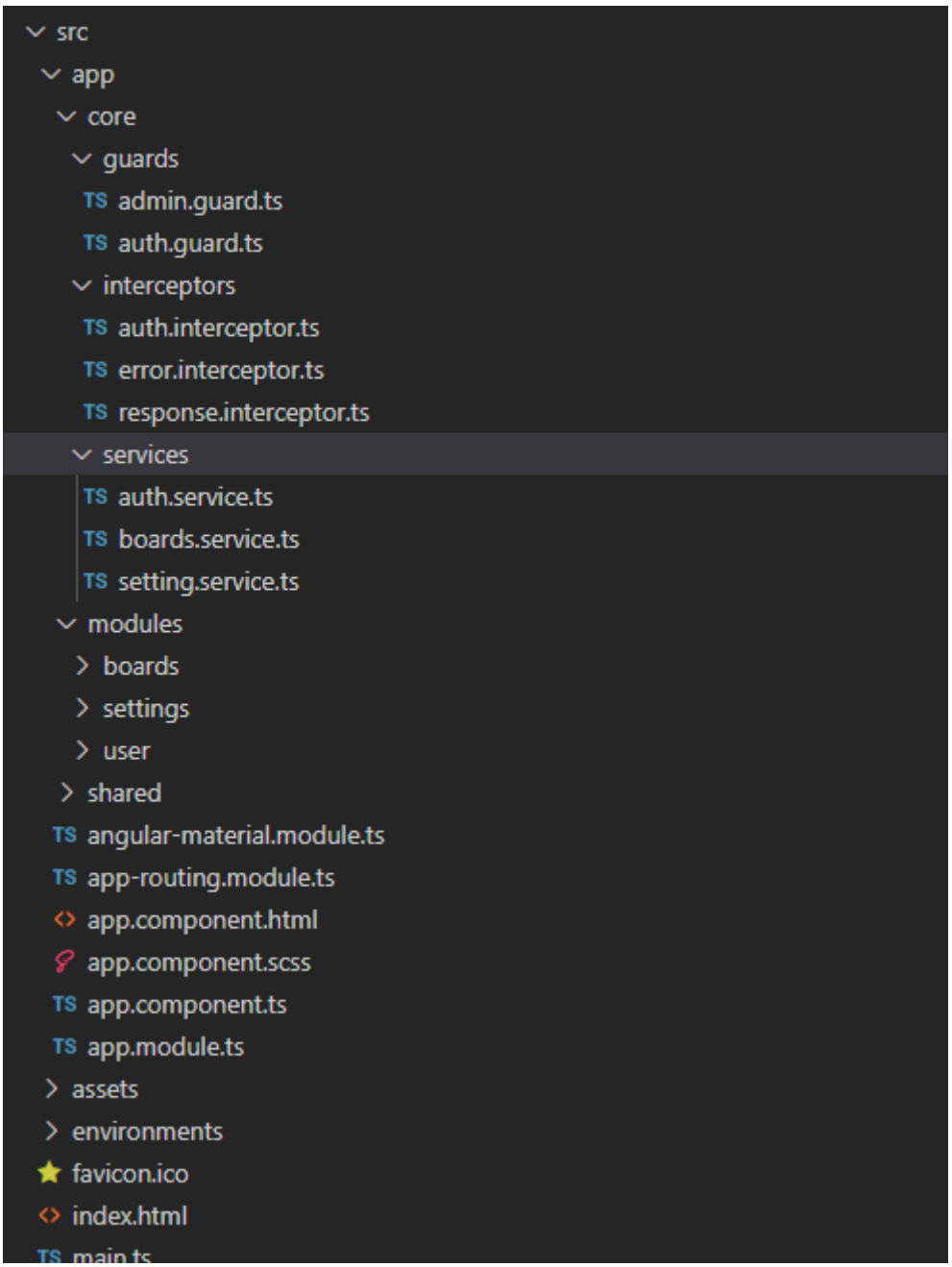


Рисунок 3.14 – Проект для зовнішньої частини

ВИСНОВОК

В магістерський кваліфікаційній роботі було досліджено основні існуючі додатки документообігу із цифровими документами,методи генерації цифрових документів, парсингу та конвертації файлів. Була проаналізована предметна область і поставлена завдання. В існуючих додатках та компонентах документообігу було виявлено недоліки і в результаті розробки модифікованого додатку документообігу та алгоритмів генерації цифрових документів він був виправлений. Розроблений модифікований додаток документообігу базується на існуючих бібліотеках, але дозволяє виконувати задачу генерації цифрових документів більш ефективно завдяки своїй оптимізації використання ресурсів, таких як операції на конвертацію форматів. Дані методи не є універсальним рішенням через можливі складності в реалізації для більш складних рівнів, але добре себе показує в ситуаціях, коли треба сформувати швидке рішення до проекту. Тепер метод для документообігу став оптимізованим у генерації завдяки швидкій конвертації, що дозволить ефективно генерувати документи з різними форматами.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Вікіпедія.Дія як сервіс збереження чутливих даних та виконання послуг.URL:https://uk.wikipedia.org/wiki/%D0%94%D1%96%D1%8F_(%D1%81%D0%B5%D1%80%D0%B2%D1%96%D1%81) (дата звернення 29.10.2021)

2. JWT як безпечний токен. URL: https://vc.ru/dev/106534-jwt-kak-bezopasnyy-sposob-autentifikacii-i-peredachi-dannyh(дата звернення 29.10.2021)

3. Википедия. Атака посредника. URL:https://ru.wikipedia.org/wiki/%D0%90%D1%82%D0%B0%D0%BA%D0%B0_%D0%BF%D0%BE%D1%81%D1%80%D0%B5%D0%B4%D0%BD%D0%B8%D0%BA%D0%B0(дата звернення 29.10.2021)

4. Как работает «Дия» и что это такое. URL: https://zaborona.com/ru/opasnost-gosudarstva-v-smartfone/

5. Что такое парсинг. URL: https://blog.ringostat.com/ru/parsing-dannyh-s-saytov-chto-eto-i-zachem-on-nuzhen/(дата звернення 29.10.2021)

6. Введение в компонент Aspose.Words. URL: https://russianblogs.com/article/6008237183/

7. Мастер импорта неструктурированных файлов в SQL. URL: https://docs.microsoft.com/ru-ru/sql/relational-databases/importexport/import-flat-file-wizard?view=sql-server-ver15(дата звернення 29.10.2021)

8. Функції та можливості Aspose.Cells. URL: https://docs.aspose.com/cells/net/your-first-aspose-cells-application-hello-world/#code-sample-creating-a-new-workbook (дата звернення 29.10.2021)

9. Microsoft.Office.Interop.Word. URL: https://docs.microsoft.com/ru-ru/dotnet/csharp/programming-guide/interop/how-to-access-office-onterop-objects (дата звернення 29.10.2021)

10. OAuth 2.0 простым и понятным языком. URL https://habr.com/ru/company/vk/blog/115163/ (дата звернення 29.10.2021)

11. Википедия. OpenId. URL: https://ru.wikipedia.org/wiki/OpenID (дата звернення 29.10.2021)