Міністерство освіти і науки України
Рівненський державний гуманітарний університет
Кафедра інформаційних технологій та моделювання

Кваліфікаційна робота

за освітнім ступенем «бакалавр»

на тему:

ІНФОРМАЦІЙНА СИСТЕМА ПІДТРИМКИ ЕЛЕКТРОННОГО ДОКУМЕНТООБІГУ ЗАКЛАДУ ВИЩОЇ ОСВІТИ. МОДУЛЬ «ДИПЛОМ»

Виконав:
здобувач IV курсу
групи ІПЗ-41
спеціальності 121 «Інженерія програмного забезпечення»
Євген ГАФИЧ

Науковий керівник:
кандидат педагогічних наук, доцент
Сергій ПЕТРЕНКО

Рівне – 2025

ЗМІСТ

СПИСОК ТЕРМІНІВ ТА УМОВНИХ ПОЗНАЧЕНЬ

API — application programming interface

HTTP — hypertext Transfer Protocol

IDE — integrated development environment

JSON — JavaScript object notation

REST — representational State Transfer

URL — uniform resource locator

ЕДО — електронний документообіг

ЄДЕБО — єдина державна електронна база з питань освіти

КЕП — кваліфікований електронний підпис

ООП — об'єктно-орієнтоване програмування

ОС — операційна система

ПЗ — програмне забезпечення

УЕП — удосконалений електронний підпис

ВСТУП

Сучасний розвиток цифрових технологій впливає на всі сфери життя, включаючи освіту та управління академічною документацією. Зростаючий обсяг інформації, необхідність в автоматизації процесів та забезпечення надійного зберігання й обміну даними висувають нові, високі, вимоги до систем електронного документообігу. У зв'язку з цим стає актуальним створення нових, сучасних рішень, які матимуть здатність забезпечити ефективну підтримку академічних процесів, зокрема і автоматичне створення дипломів і додатків до них, що відповідають стандартам.

Актуальність дослідження зумовлена вдосконалення процесів управлінням академічною документацією та автоматизації створення і обробки дипломів та додатків до них. Оформлення цих документів – це довгий процес, що передбачає збір, перевірку та впорядкування великого обсягу даних. В умовах ручної обробки це може призвести до виникнення помилок, затримку у підготовці документів та в потребі повторної перевірки. Крім того, такий підхід потребує значного часу або участі кількох працівників, що не завжди є доцільно.

Використання сучасних інформаційних технологій для автоматизації цього процесу дозволяє мінімізувати людський фактор, забезпечити точність перенесення даних, пришвидшити підготовку та полегшити контроль за їх відповідністю встановленим стандартам.

Мета дослідження полягає у розробці системи підтримки електронного документообігу, яка автоматизує процес створення дипломів та додатків до нього, забезпечить їх зберігання у надійному сховищі у зручному форматі, використовуючи сучасні технології та інструменти.

Для реалізації поставленої мети передбачено такі **завдання:**

1. дослідити поняття електронного документообігу;
2. проаналізувати існуючі системи, для автоматичного створення документів;
3. визначити вимоги до системи;

4. розробити архітектури системи;
5. розробити програмний додаток;
6. провести аналіз ефективності.

Об'єктом дослідження є інформаційна система електронного документообігу закладу вищої освіти, яка охоплює процеси створення, обробки та зберігання документів, зокрема дипломів та додатків до них.

Предметом дослідження є методи, засоби та програмні технології автоматизації формування та обліку дипломів і додатків у рамках модуля інформаційної системи електронного документообігу.

Методи дослідження:

- аналіз наукової літератури і стандартів;
- об’єктно–орієнтований аналіз;
- моделювання системи.

Практичне значення дослідження полягає в оцінці того, як отримані результати та висновки можуть бути використані в реальному житті для вирішення практичних завдань чи проблем. У даному випадку, створена система підтримки електронного документообігу демонструє можливість автоматизації процесу створення та зберігання дипломів та додатків до них. Використання цієї системи може значно полегшити роботу закладів вищої освіти, скоротити час на підготовку документів, мінімізувати людський фактор та забезпечити відповідність сучасним стандартам.

Результати дослідження можуть бути застосовані для подальшої розробки та впровадження подібних систем у навчальних закладах, державних установах та приватних організаціях, що мають потребу в автоматизації процесів документування. Це може підвищити ефективність, знизити витрати та сприяти оптимізації робочих процесів.

Апробація і впровадження результатів дослідження. Основні теоретичні результати дослідження обговорювалися на XVIII Всеукраїнської науково–

практичної конференції здобувачів вищої освіти і молодих учених «НАУКА, ОСВІТА, СУСПІЛЬСТВО ОЧИМА МОЛОДИХ» (м. Рівне, 14 травня 2025 р.), звітній науковій конференції викладачів, співробітників і здобувачів вищої освіти Рівненського державного гуманітарного університету за 2025 рік (м. Рівне, 15 травня 2025 року).

Структура роботи. Кваліфікаційна робота складається з вступу, трьох розділів, висновків та списку використаних джерел. Загальний обсяг роботи становить 40 сторінок. У роботі використано 13 рисунків. Список джерел налічує 21 найменування.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Інформаційні системи та електронний документообіг: основні поняття та аналіз сучасних рішень

Інформаційна система — це комунікаційна система, яка здійснює або в якій відбуваються інформаційні процеси: збирання, пошук, зберігання, передавання й оброблення інформації.

У інформаційні системи можуть відбуватись одночасно один, два чи кілька процесів. Опрацювання інформації залежить від змісту вхідної інформації, але під час самого опрацювання інформація не осмислюється, а лише перетворюється згідно з попередньо розробленими алгоритмами [1].

У сучасних умовах інформаційні системи допомагають закладам вищої освіти ефективно управляти великими обсягами інформації, що покращує комунікацію та продуктивність.

Інформаційна технологія — це сукупність методів, виробничих процесів і програмно-технічних засобів, об’єднаних у технологічний ланцюжок, що забезпечує збір, обробку, зберігання, поширення й відображення інформації з метою зниження трудомісткості процесів використання інформаційного ресурсу, а також підвищення його надійності й оперативності [2, с. 36].

У сучасних умовах інформаційні технології відіграють ключову роль у забезпеченні ефективної діяльності закладів. Їхнє впровадження дозволяє оптимізувати робочі процеси та підвищити продуктивність праці.

Інформаційні системи офісної автоматизації застосовують працівники середньої кваліфікації: бухгалтери, секретарі, клерки. Основними завданнями таких систем є обробка даних, підвищення ефективності рутинної роботи тощо. Інформаційні системи офісної автоматизації з’єднують працівників інформаційної сфери в різних регіонах і допомагають підтримувати зв’язок з покупцями, замовниками, іншими відповідними організаціями. Їхня діяльність в основному

охоплює управління потоками документації, комунікацію, складання розкладів і т.д [2, с. 112].

Системи автоматизації процесів стають необхідним інструментом для взаємодії між учасниками інформаційного середовища. Вони дозволяють забезпечити контроль за виконанням поточних завдань та сприяють мінімізації людського фактора.

Один із процесів інформаційної системи є створення документів. У інформаційній технології обробки даних необхідно розробляти документи для керівництва і працівників підприємства, а також зовнішніх партнерів. При цьому документи можуть створюватись як за запитом або у зв'язку із проведеною підприємством операцією, так і періодично [2, с. 120].

Автоматизоване створення документів забезпечить стандартизоване оформлення, зменшить витрати часу та знизить відсоток помилок. Тому автоматизація цього процесу є одним з головних напрямків розвитку інформаційних технологій.

Різні інформаційні системи використовуються для управління, навчання, адміністративної роботи, а також для підтримки наукових досліджень.

Документообіг — це рух документів в компанії від їхнього створення (чи отримання від контрагента) до передачі в архів.

Електронний документообіг — це, відповідно, такий же процес, тільки з документами в електронному вигляді.

Кожна установа має свої процеси роботи з документами, але зазвичай, до них належить створення чи одержання документа, обробка та редагування, узгодження та підписання, надсилання контрагенту чи передача на виконання, архівування. Залежно від виду документа, компанії, налаштованих процесів ці етапи можуть доповнюватись та відрізнятись. Всі процеси, які відбуваються з паперовими документами, можуть бути і в електронному форматі, завдяки чому оптимізуються, стають швидшими та дешевшими. Системи електронного документообігу ж дозволяють бути впевненими в захисті документів, відслідковувати всі дії, які з ними відбувались (бачити, хто, коли і що саме робив з документом), підписувати

документи за допомогою КЕП, УЕП чи хмарних підписів, а також забезпечують повноцінну організацію роботи з документами [3].

Електронний документообіг дозволяє реалізувати обробку документів у цифровому форматі, що значно спрощує процес створення, редагування та передачі інформації в закладах. Він передбачає використання спеціалізованих програмних рішень для забезпечення швидкого та безпечного обміну документами між учасниками цих процесів. Завдяки цьому значно скорочується витрати на паперові матеріали та фізичне зберігання, також мінімізується ризики втрати чи пошкодження документів. Упровадження електронного документообігу у вищих навчальних закладах дозволяє автоматизувати процеси видачі дипломів, зберігання оцінок та іншої документації, що значно полегшує управлінську діяльність.

Сучасні системи підтримки електронного документообігу використовуються для автоматизації процесів створення, зберігання та обробки даних, управління доступом та інтеграції з іншими системами. Системи ЕДО можуть використовуватись в освітніх установах для ведення студентських записів, видачі дипломів та додатків до них. Основними характеристиками таких систем повинне бути забезпечення надійного захисту інформації, підтримка стандартизованих форматів файлів, а також можливість інтеграції з зовнішніми сервісами через API.

При **аналізі існуючих рішень** можна відзначити їхню архітектуру, яка часто базується на клієнт–серверній моделі або мікросервісному підході. Мікросервіси забезпечують масштабованість та гнучкість системи, дозволяючи окремим компонентам взаємодіяти через чітко визначені інтерфейси. Використання REST для обміну даними забезпечує зручний доступ до різних функцій системи через HTTP–запити.

1.2. Визначення вимог до системи

Функціональні вимоги (functional requirements) визначають функціональність ПЗ, яку розробники повинні забезпечити, щоб користувачі змогли виконати свої завдання в межах бізнес-вимог. Інколи їх називають вимоги

поведінки (behavioral requirements), вони містять положення з традиційним «повинна». Наприклад, «Система повинна по електронній пошті відправляти користувачу підтвердження замовлення» [4].

Функціональні вимоги описують, що розробнику необхідно реалізувати. є ключовим елементом у розробці будь-якої інформаційної системи. Для підтримки електронного документообігу в освітньому закладі функціональні вимоги включають:

1. Збір та обробку даних про студентів та освітніх програм з використанням зовнішніх API

Однією з основних функціональних вимог є збір та обробка даних про студентів та освітніх програм, на яких навчаються студенти. Система повинна мати можливість інтеграціями з зовнішніми сервісами для отримання актуальної інформації. Це передбачає реалізацію захищеного доступу до даних за допомогою запитів, що відповідають стандартам REST. Використання API дозволяє системі автоматично отримувати дані, такі як особисті дані студентів, їхні оцінки та кваліфікації, що підвищує точність і швидкість обробки інформації.

2. Автоматизоване створення документів, таких як дипломи та додатки до них

Ця функціональна вимога включає використання шаблонів документів, що містять ключові слова, які у процесі обробки заміняться на відповідні дані. Такий підхід не лише економить час на створення документів вручну, але й мінімізує ризик помилок, пов’язаних з людським фактором.

3. Конвертація документів у зручний формат

Однією з вимог до системи є можливість конвертації створених документів у PDF-формат для зручності зберігання, однакового відображення на різних пристроях та операційних системах та подальшого використання. Конвертація має бути автоматизованою та без встрати якості документу, що є важливим для підтримки стандартів документації.

4. Зберігання створених документів у захищеному хмарному сховищі

Для забезпечення безпеки та зручності доступу до документів, система

передбачає збереження всіх створених файлів у хмарному сховищі. Вибір такого підходу зумовлений необхідністю гарантувати надійність зберігання, конфіденційність та можливість віддаленого доступу до них. Хмарні технології, такі як Google Drive, пропонують можливість інтеграції за допомогою API, що дозволяє автоматизувати процес завантаження та управлінням файлами.

Додаткові функціональні вимоги

5. Управління доступом

Система повинна бути захищена від можливості створення дипломів та додатків звичайними користувачами. Це передбачає, що скористатись цією функцію мають право тільки авторизовані користувачі. Управління доступом забезпечує підвищення рівня безпеки та кондиційності інформації.

6. Логування та моніторинг

Для забезпечення надійності та стабільної роботи системи необхідно впровадити механізми логування подій. Це дозволить адміністратором системи відстежувати процеси створення документів, доступу до них та інших дій у системі. Моніторинг системи допоможе швидко виявляти та усувати будь-які неполадки або збої в роботі.

Нефункціональні вимоги або NFR (non-functional requirements) – це набір специфікацій, які описують робочі можливості та обмеження системи та намагаються покращити її функціональність. Це в основному вимоги, які визначають, наскільки добре працюватиме продукт якщо врахувати, наприклад, швидкість, безпеку, надійність, цілісність даних тощо. Структура стандартів ISO/IEC 25000 визначає нефункціональні вимоги як вимоги до якості системи та якості програмного забезпечення [6].

Нефункціональні вимоги є не менш важливими, ніж функціональні, оскільки вони визначають загальну якість системи та її здатність відповідати очікування користувачів. Для системи підтримки електронного документообігу в освітньому закладі нефункціональні вимоги включають:

1. Продуктивність

Однією з ключових нефункціональних вимог системи є висока

продуктивність. Система повинна забезпечувати швидкий відгук на запити користувачів, навіть при значному обсязі даних, що обробляється одночасно. Це особливо важливо під час пікових навантажень, наприклад при масовому створенні документів у кінці навчального року. Вимога щодо продуктивності включає мінімізацію часу обробки запитів, швидкість створення документів та передачу даних до зовнішніх сервісів. Забезпечення стабільної роботи системи сприяє покращенню користувацького досвіду та зниженню рівня затримок у процесах обробки даних.

2. Масштабованість

Система повинна бути здатна до масштабування, щоб відповідати зростаючим потребам навчального закладу. Це означає, що система повинна легко адаптуватись до збільшення кількості користувачів, зростання обсягу обролюваних даних та інтеграції з новими зновнішніми джерелами даних. Масштабованість передбачає можливість розширення апаратних ресурсів або збільшення обчислювальної потужності без порушення роботи системи. Це важливо для забезпечення стабільної роботи системи про розширенні її функціоналу та збільшенні навантаження.

3. Портативність

Портативність системи включає її здатність працювати на різних операційних системах, що забезпечує гнучкість у використанні та розгортанні мікросервісів. Це дозволяє запускати мікросервіс як на Windows, так і на Linux, macOS та інших операційних системах без необхідності значних змін у коді чи конфігурації. Така портативність забезпечує можливість обирати платформу відповідно до потреб та наявних ресурсів навчального закладу. Завдяки підтримці багатьох ОС система може інтегруватися у різні IT-інфраструктури, що розширює можливості її використання і полегшує процес розгортання та підтримки.

1.3 Технологічні основи розробки інформаційної системи

Мова програмування — це штучна мова для написання команд, виконуваних обчислювальною машиною. Мова програмування складається з фіксованого словника і сукупності правил (синтаксису) написання команд. Оскільки мова програмування незрозуміла для обчислювальної машини, має бути спеціальна програма, яка перекладала б символи цієї мови мовою машинних команд. Така програма перекладу символів, або, простіше, транслятор (від англійського слова translation - переклад), була створена на початку 50-х років XX століття американською програмісткою, контрадміралом морських сил США Грейс Хопер [6].

Мови програмування можна поділити на різні покоління, від низькорівневих, таких як асемблер, до високорівневих, як C++, C#, Python, Java та інші. Вони відрізняються своє структурою, синтаксисом, можливостями та сферою застосування. Високорівневі мови програмування зазвичай легкі у використанні та зрозуміліші для людини, оскільки вони ближчі до природної мови та приховуються деталі роботи з апаратною частиною комп’ютера.

Сучасні мови програмування часто будуються на концепції об’єктно–орієнтованого програмування, які допомагають розробникам створювати більш структуровані та зрозумілі програми. Мова програмування також обирається залежно від специфічних потреб у проєкту, таких як продуктивність, простота підтримки та доступність бібліотек для вирішення певних задач.

Кожна мова програмування має свої переваги та недоліки, які впливають на її вибір для конкретних завдань. Наприклад C# є популярною мовою програмування для розробки вебзастосунків та мікросервісів завдяки, завдяки своїй інтеграції з платформою .NET, що надає широкий набір інструментів та бібліотек для швидкого створення програмного забезпечення.

Об’єктно–орієнтоване програмування дозволяє полегшити повторне використання коду. Для того щоб мова програмування була об'єктно-орієнтованою, вона має використовувати класи й об'єкти, а також реалізовувати фундаментальні

принципи об'єктно-орієнтованого підходу, головними з яких є абстракція, інкапсуляція, ієрархія та поліморфізм [7].

Концепції ООП дозволяють створювати модульний код, що легко підтримується, тестується та розширюється. Завдяки таким принципам, розробники можуть організовувати програмний код так, щоб кожен об’єкт відповідав за певну функціональність, зменшуючи взаємозалежність між компонентами системи. Це сприяє створенню більш зрозумілих і гнучких програмних архітектур, які можуть ефективно розвиватися та масштабуватись.

Ще однією важливою властивістю ООП є підтримка механізму обробки подій, які змінюють атрибути об'єктів і моделюють їх взаємодію у предметній області. Переміщаючись по ієрархії класів від більш загальних понять предметної області до більш конкретних і навпаки, програміст отримує можливість змінювати ступінь абстрактності погляду на модельований ним реальний світ. Використання раніше розроблених (можливо, іншими колективами програмістів) бібліотек об'єктів і методів дозволяє значно заощадити трудовитрати при виробництві програмного забезпечення, особливо типового. Об'єкти, класи і методи можуть бути поліморфними, що робить реалізоване програмне забезпечення більш гнучким і універсальним [8, с. 19].

Сучасні інструменти для програмування, такі як середовища Visual Studio, Visual Studio Code чи Intellij IDEA, надають розробника зручні можливості для написання ООП–коду, полегшуючи роботу над великими проєктами.

REST — це архітектурний стиль для розподілених гіпермедійних систем, який Рой Філдінг вперше представив у 2000 році в своїй знаменитій дисертації. З того часу він став одним із найбільш широко використовуваних підходів для створення вебінтерфейсів API (інтерфейсів прикладного програмування) [9].

REST — це не протокол чи стандарт, це архітектурний стиль, а точніше — один з наявних типів синхронної комунікації вебсервісів. На етапі розробки розробники API можуть реалізувати REST різними способами [9].

REST став основою для створення веб–служб, які відрізняються своєю простотою та масштабованістю. Основний принцип роботи REST полягає в тому,

що клієнт відправляє запит на сервер, а сервер відповідає відповідними даними, не зберігаючи інформацію про стан клієнта між запитами. Це дозволяє розробляти легкі та ефективні системи, де взаємодія є статичною та базується на чітких правилах обробки запитів. Серед головних характеристик REST є використання стандартних HTTP методів, таких як GET, POST, PUT та DELETE, що забезпечує виконання основних операцій над ресурсами. Завдяки своїй гнучкості та підтримці різних форматів даних, REST став стандартом у створенні API для веб–додатків. JSON, як найбільш популярний формат для передачі даних, надає зручний способі взаємодії між клієнтом і сервером завдяки його легкості у читанні, серіалізації та десеріалізації.

REST API широко застосовуються для інтеграції різних систем і створення сервісів, які обмінюються даними у реальному часі, сприяючи побудові масштабованих і зручних у використанні архітектур.

Інтерфейс програмування застосунків (API) — це набір правил і протоколів, які дозволяють взаємодіяти між різним програмним забезпеченням навіть у режимі реального часу . Цей інструмент дозволяє одній програмі використовувати функціональність іншої, не знаючи внутрішньої реалізації останньої, шляхом обміну даними між програмами. API визначає, як різні компоненти програмного забезпечення повинні взаємодіяти один з одним, які дані збирати, а які ігнорувати . Цей процес може включати набір інструкцій, протоколів зв'язку, форматів даних та інших елементів, що сприяють обміну інформацією між програмами та можуть підвищити зручність їх використання [10].

Прикладний програмний інтерфейс відіграє ключову роль у забезпеченні взаємодії між різними програмними компонентами та системами. Завдяки використанню API розробники можуть інтегрувати свої додатки з іншими сервісами та платформами, що значно підвищує функціональність та ефективність програмного забезпечення. API надає можливість створювати програмні продукти, які можуть обмінюватись даними та виконувати спільні функції без необхідності глибокого розуміння внутрішньої структури кожного окремого модуля або системи.

Сучасні API часто будуються за допомогою стандартів REST, SOAP або GraphQL, що забезпечує надійну та масштабовану взаємодію між системами. Використання API дозволяє зменшити обсяг коду, прискорює процес розробки та підтримки програмного забезпечення, а також забезпечують безпеку передачі даних завдяки використанню механізмів аутентифікації та авторизації, таких як OAuth або JWT.

API можуть бути, як публічними, що надають доступ до широкого кола користувачів, так і приватними, обмеженими лише для внутрішнього користування. Вони є невід'ємною частиною багатьох сучасних додатків та служб, оскільки забезпечують можливість взаємодії з різними сервісами та полегшують інтеграцію в багатокомпонентні системи.

Мікросервіси — це архітектурний підхід, коли єдиний додаток будується як сукупність невеликих, самодостатніх, незалежних, не тісно зв'язаних сервісів, що спілкуються між собою за допомогою легких механізмів як то HTTP, gRPC, AMQP. Ці сервіси побудовані навколо бізнес-потреб (кожен відповідальний за конкретний процес) та розгортаються незалежно з використанням повністю автоматизованого середовища. Існує абсолютний мінімум централізованого управління цими сервісами. Самі по собі сервіси можуть бути написані на різних мовах і використовувати різні технології зберігання даних [11].

Сучасні мікросервіси будуються на високому рівні незалежності один від одного, можуть незалежно масштабуватись, один від одного, зазвичай, містять не багато бізнес-логіки, що зменшує кількість конфліктів, дозволяють швидко залучати нових розробників, мають можливість швидко і просто додавати новий функціонал у систему.

Контролер (controller**)** — це клас, який реалізує операції, визначені API застосунку. Він реалізує бізнес-логіку застосунку та діє як місток між HTTP/REST API та моделями домену/бази даних. До controller класу та його членів додаються декорації для зіставлення операцій API застосунку з відповідними операціями контролера. А controller працює лише з обробленими вхідними даними та абстракціями серверних служб/баз даних [12].

Контролери обробляють вхідний HTTP-запит. Коли клієнт надсилає запит на сервер, цей запит спочатку проходить через конвеєр обробки запитів. Після того, як запит проходить конвеєр обробки запитів (тобто компонент проміжного програмного забезпечення зареєстровано в конвеєрі), він потрапляє до контролера. Усередині контролера є багато методів (так званих методів дій), які обробляють вхідні HTTP-запити. Метод дії всередині виконує бізнес-логіку та готує відповідь, яка надсилається назад клієнту, який спочатку її запитував [13].

РОЗДІЛ 2
ВИБІР ТЕХНОЛОГІЧНОГО СТЕКА ТА ЙОГО ОБҐРУНТУВАННЯ

2.1 Вибір мови програмування та інтегрованого середовища розробки

Для реалізації мікросервісу, що підтримує електронний обіг документів, обрано **мову програмування** C#.

Мова C# (вимовляється "Сі-шарп") – це багатопарадигмова об'єктно-орієнтована та компонентно-орієнтована мова програмування зі строгою типизацією, розроблена для платформи .NET Framework [14, с. 9].

Цілі, поставлені при розробці мови C#, були такими:

- C# має бути простою, сучасною, об'єктно-орієнтованою мовою програмування.
- Мова має підтримувати безпечні принципи програмування, такі як строга перевірка типів, перевірка меж масиву, виявлення спроб використання неініціалізованих змінних, і автоматичне прибирання сміття.
- Можливість розробки програмних компонентів для розподілених систем.
- Мова має підтримувати переносимість коду.
- Підтримка національних мовних та інших особливостей має бути простою [9, с. 10].

C# є потужною та сучасною мовою програмування, яка ідеально підходить для створення веб-сервісів та мікросервісів завдяки інтеграції з платформою .NET. Вона підтримує об’єктно–орієнтоване програмування та забезпечує зручні засоби для побудови безпечних і масштабованих рішень. Ця мова також має багатий набір бібліотек і фреймворків для роботи з HTTP-запитами, обробками даних і реалізацію асинхроних операцій, що є критично важливим для мікросервісу.

Для розробки мікросервісу на мові програмування C# було обрано **інтегроване середовище розробки** Visual Studio.

Microsoft Visual Studio — це інтегроване середовище розробки програмного забезпечення. Воно складається із сукупності програмних продуктів для розробки та налагодження застосунків різних видів, у тому числі консольних програм, застосунків з графічним інтерфейсом користувача, web-сайтів, web-застосунків, web-служб тощо. Visual Studio використовує такі платформи для ПЗ, як Windows API, Windows Forms, Windows Presentation Foundation, Windows Store і Microsoft Silverlight. Середовище може компілювати як керований, так і некерований код [14, с. 11].

IDE забезпечує повний набір інструментів для створення, налагодження та розгортання системи. Воно підтримує розробку проєктів на базі ASP.NET Core або .NET 6/7/8/9/10, що забезпечує створення високопродуктивних мікросервісів із можливістю кросплатформної роботи. Середовище включає вбудовані інструменти для інтеграції з різними сервісами, роботу з API та зручне управління пакетами NuGet, що значно спрощує підключення зовнішніх бібліотек.

Visual Studio має зручний редактор для написання коду програми (рис 2.1), який здійснює автоматичне форматування тексту та підтримує IntelliSense – компонент, розроблений Microsoft для автоматизованого введення коду. При розробці програми він надає підказки. Це дозволяє швидко вибрати потрібну функцію, метод, змінну чи тип, а також переглянути переліки доступних полів, властивостей та меодів, разом з короткими описами їх параметрів або ж підключити потрібний простір імен. Вікно IntelliSense можна також викликати примусова, якщо натиснути комбінацію клавіш <Ctrl> + <Space>. Ще один зручний механізм, який надає редактор Visual Studio – рефакторинг коду. Він полягає у зміні внутрішньої будови програми для полегшеного розуміння коду без зміни її зовнішньої поведінки [14, с. 14-15].

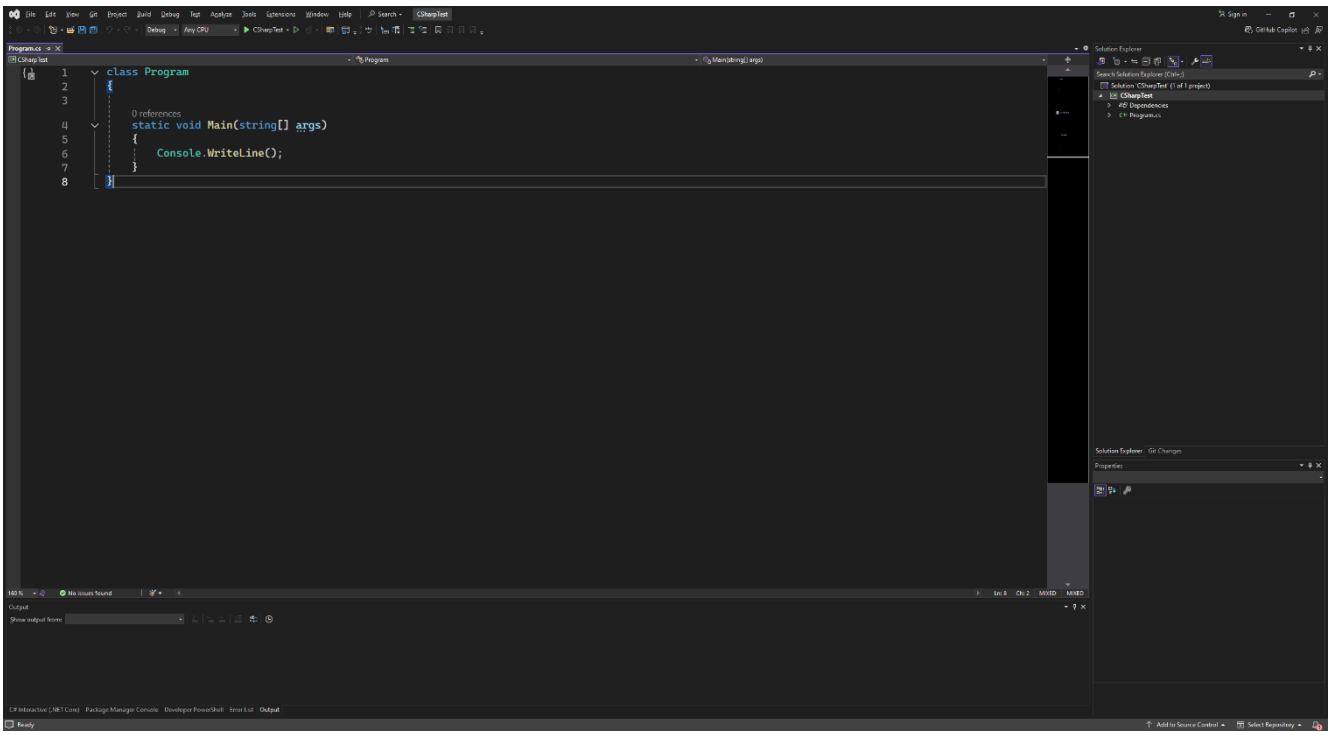

Рисунок. 2.1. Середовище розробки Visual Studio 2022

Visual Studio також підтримує роботу з різними версіями .NET, надає потужний інструментарій для написання та тестування коду, а також дозволяє швидко налагоджувати додатки завдяки вбудованому відлагоджувачу та інтеграції з системами контролю версій.

2.2 Фреймворки, бібліотеки та засоби для конвертації документів

Для створення мікросервісу використовується **фреймворк** ASP.NET Core. Це сучасний, кросплатформений і високопродуктивний фреймворк, який дозволяє створювати сервіси та мікросервіси будь-якої складності. ASP.NET Core є частиною платформи .NET й пропонує всі необхідні інструменти для розробки API, що підтримують стандарти безпеки та забезпечує швидкість обробки запитів та підтримує ОС Windows, Linux та macOS.

ASP.NET Core включає базові бібліотеки, які забезпечують широкий функціонал для розробки додатків. До цих бібліотек належать Microsoft.AspNetCore.Mvc, яка надає засоби для створення контролерів та обробки запитів, Microsoft.Extensions.DependencyInjection для налаштування впровадження

залежностей, а також Microsoft.AspNetCore.Routing для роботи з маршрутизацією. Ці бібліотеки забезпечують легкість у налаштування маршрутизації, обробці HTTP-запитів та конфігурації сервісів, що дозволяє створювати гнучі та масштабовані рішення.

Фреймворк підтримує middleware–компоненти, що легко інтегруються для реалізації аутентифікації, авторизації, логування та обробки помилок. Це дозволяє розробникам будувати безпечні API з повним контролем над усіма етапами обробки запиту.

Фреймворк також легко інтегрується з іншими бібліотеками. Завдяки вбудованим можливостям роботи з JSON та XML, ASP.NET Core спрощує передачу та обробку даних між сервісами, роблячи його ідеальним для створення ефективних і сучасних мікросервісів.

Для роботи з електронними документами у проєкті використовується **бібліотека** NPOI та **засіб конвертації документів** LibreOffice. Для взаємодії з хмарним сховищем Google Drive використовується **бібліотека** Google.Apis.Drive.v3.

NPOI — це безкоштовна бібліотека з відкритим вихідним кодом, з ліцензією Apache-2.0, для роботи з офісними документами формату Microsoft Office, такими як docx, xlsx і т.д., на платформі .NET. Вона є портом Java-бібліотеки Apache POI, що забезпечує можливість створення, читання та модифікації файлів без потреби в додаткових компонентах Microsoft Office. Ця бібліотека може працювати з шаблонами документів, змінювати ключові слова на відповідні дані, створювати таблиці та стилізувати текст.

LibreOffice — це потужний офісний пакет із відкритим кодом, з ліцензією Mozilla Public License Version 2.0, який надає широкий спектр інструментів для роботи з документами, включаючи текстові документи, електронні таблиці та презентації. У цьому проєкті LibreOffice використовується як засіб для конвертації документів із різних форматів. Вона використовується у фоновому режимі, забезпечуючи надійну і якісну конвертацію з підтримкою форматування, зображень та інших елементів документа. Використання LibreOffice дозволяє

уникнути використання платних ліцензійних інструментів та забезпечує кросплатформну підтримку, що робить його ефективним і гнучким рішенням для автоматизації процесів конвертації документів у системах електронного документообігу.

Google.Apis.Drive.v3 — це безкоштовна бібліотека з відкритим вихідним кодом, з ліцензією Apache-2.0, для роботи з хмарним сховищем Google Drive. Вона надає можливість взаємодіяти з файлами та папками, зокрема завантажувати різні типи файлів.

РОЗДІЛ 3
СТВОРЕННЯ СИСТЕМИ

3.1 Архітектура системи підтримки електронного документообігу

Система підтримки електронного документообігу має на меті автоматизацію процесів створення, обробки та зберігання документів, що стосуються академічної діяльності студентів, а саме дипломів та додатків до них. Основною архітектури цієї система є взаємодія кількох мікросервісів, для обміну даними про студента, його отриманні бали та інформація про освітню програму, на якій студент навчався. Ці сервіси обмінюються даними через API та забезпечують автоматизовану обробку інформації. Збереження документів здійснюється на зовнішньому хмарному сховищі, такому як Google Drive. Інформація, яка має приходити з зовнішніх сервісів:

- серія, номер диплома, дата
- номер додатку диплома, дата
- прізвище та ім’я, дата народження, стать студента
- ступінь вищої освіти
- спеціальність
- спеціалізація
- професійна кваліфікація
- галузь знань
- мова навчання/оцінювання
- рівень кваліфікації
- вимога для вступу
- форма здобуття освіти

- програмні результати навчання
- накопичені індивідуальні кредити та отримані бали/оцінки
- класифікація кваліфікації
- доступ до подальшого навчання
- інформація про орган акредитація та сертифікат

DiplomaAPI – назва системи, яка розробляється в рамках кваліфікаційної роботи.

Основні етапи створення диплома та додатку до нього.

3.1.1 Створення диплома (рис. 3.1):

1. Користувач надсилає запит на створення диплома: Користувач, зокрема працівник деканату, ректор або інша відповідальна особа, відправляє запит через інтерфейс користувача до мікросервісу для створення диплома.

2. Запит на отримання інформації: DiplomaAPI надсилає запит до мікросервісу для отимання інформації: серію та реєстраційний номер диплома, ім’я та прізвище студента, дата закінчення навчання, освітню програму, найменування органу акредитації, ступінь освіти, галузь знань, спеціальність, професійну кваліфікацію, додаткову інформацію, посаду, ім’я та прізвище керівника, або іншої уповноваженої особи, дату видачі.

3. Отримує інформацію: мікросервіс повертає необхідну інформацію.

4. Створення диплома: DiplomaAPI генерує диплом, заповнюючи поля інформацією та конвертує у PDF формат.

5. Збереження диплома в Google Drive: DiplomaAPI знаходить потрібну папку в хмарному сховищі Google Drive та відправляє запит на збереження цього документа в цю папку.

6. Отримання підтвердження зберігання: GoogleDriveAPI відправляє підтвердження збереження файлу в хмарному сховищі.

7. Повертання посилання користувачу: DiplomaAPI повертає користувачу посилання на папку в хмарному сховищі у форматі URL.

Рисунок. 3.1. Архітектура системи створення диплома

3.1.2. Створення додатку до диплома (рис. 3.2):

1. Користувач надсилає запит на створення додатку до диплома: відповідальна особа відправляє запит до DiplomaAPI для створення додатку до диплома.

2. Запит на отримання інформації про студента: DiplomaAPI відправляє запит до сервісу для отримання інформації про: код картки фізичної особи в ЄДЕБО, дату народження, документ про освіту, що був підставою для вступу.

3. Запита на отримання інформації додаткової інформації про студента серію, номер, дату диплома та номер і дату додатку до диплома, уповноваженої особи закладу: DiplomaAPI відправляє запит до сервісу для отримання інформації про: серію, реєстраційний номер та дату видачі диплома, реєстраційний номер, дату видачі додатку до диплома, прізвище, ім’я, особливі досягнення та відзнаки студента, ім’я та прізвище, посаду керівника або уповноваженої особи.

4. Запит на отримання інформації про оцінки студента та випускників цієї спеціальності за певну кількість років: DiplomaAPI відправляє запит до сервісу для отримання інформації про оцінки студента, здобуті впродовж навчання, а також оцінки здобуті випускниками цієї ж освітньої програми впродовж

декількох років або року.

5. Запит на отримання інформації про освітню програму: DiplomaAPI відправляє запит на отримання інформації про ступінь вищої освіти, спеціальність, спеціалізація, професійну кваліфікацію, основну галузь знань, рівень кваліфікації, тривалість освітньої програми, вимоги для вступу, програмні результати, доступ до подальшого навчання в межах освітньої програми.

6. Отримання інформації від цих сервісів: DiplomaAPI чекає на отримання всієї інформації від сервісів.

7. Створення додатку до диплома: DiplomaAPI генерує додаток до диплома, заповнюючи поля інформацією і конвертує у PDF формат.

8. Збереження додатку до диплома в Google Drive: Diploma API відправляє запит на збереження додатку в хмарному сховищі, у потрібній папці.

9. Отримання підтвердження зберігання: GoogleDriveAPI повертає підтвердження зберігання документа.

10. Повертання посилання на документ користувачу: DiplomaAPI повертає користувачу посилання на папку в хмарному сховищі у форматі URL.

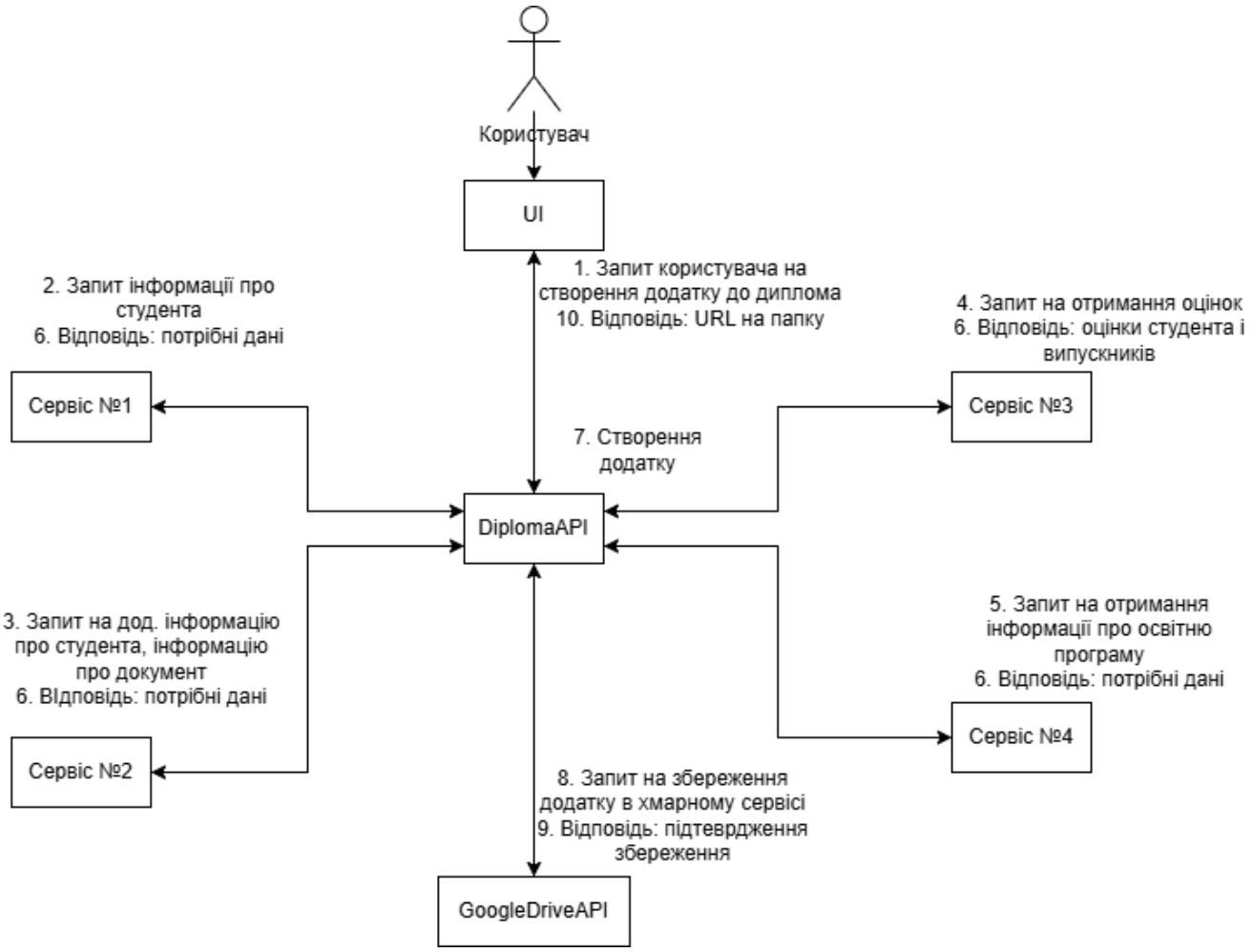

Рисунок. 3.2. Архітектура системи створення додатку до диплома

Архітектура система підтримує гнучке масштабування та дозволяє інтегрувати нові сервіси за потреби.

3.2 Проєктування та створення системи електронного документообігу

3.2.1 Проєктування моделей, сервісів та контролерів системи електронного документообігу

Під час проєктування та створення системи електронного документообігу важливо визначити, які саме дані будуть використовуватись та як ці дані будуть передаватись між компонентами системи. Ці **моделі** використовуються для абстракції диплома та додатку до нього, а також для отримання даних з інших сервісів:

- DiplomaModel, StudentName, InformationEducationProgramme, Capacity – моделі, які повторюють структуру диплома;
- DiplomaSupplementModel, InformationIdentifyingHolderQualification, InformationIdentifyingQualification, InformationLevelDurationQualification, InformationProgrammeCompletedResultsObtained, StudentMark, GradeDistributionTable, InformationAcademicProfessionalRightsFunctionQualification, AdditionalInformation – моделі, які повторюють структуру додатку до диплома;
- Diploma – модель для заповнення полів диплома;
- DiplomaSupplement – модель для заповнення полів додатку до диплома.

Сервіси, які взаємодіють через API з іншими системами, для отримання та надсилання даних:

- GoogleDriveService – сервіс для взаємодії з хмарним сховищем Google Drive, шукає потрібну папку для зберігання документа, а також зберігає документ, детальніша інформація в пунктах 3.1.1.5 та 3.1.2.8;
- StudentService – сервіс для отримання даних про студента зазначених в пункті 3.1.2.2;
- StudentDocumentService – сервіс для отримання додаткових даних про студента та інформацію про документ зазначених в пунктах 3.1.1.2 та 3.1.2.3;
- StudentEvaluationsService – сервіс для отримання даних про оцінки студента та оцінки випускників цієї спеціальності зазначених в пункті 3.1.2.4;
- EducationProgrammeService – сервіс для отримання даних про освітню програму студента зазначених в пунктах 3.1.2.5.

Контролери, які отримують запит на генерування і зберігання диплома чи додатку до нього:

- DiplomaController – контролер, який отримує запит з системи для генерування диплому та зберігання його в хмарному сховищі;
- DiplomaSupplementController – контролер, який отримує запит з системи для генерування додатку до диплома та зберігання його в хмарному сховищі.

3.2.2 Створення системи електронного документообігу

У процесі розробки було прийнято рішення використовувати шаблон документа у форматі docx, що містить ключові слова {Series}, {Number}, {FirstNameUkraine} тощо. Ці ключові слова надалі замінюються відповідними даними отримані з сервісів.

Було створено два окремі шаблони: шаблон диплома (рис. 3.4) та шаблон додатку до диплома (рис. 3.3). Обидва документи були структуровані згідно з наказом Міністерства освіти і науки України.

УКРАЇНА

UKRAINE

ДОДАТОК ДО ДИПЛОМА /
DIPLOMA SUPPLEMENT

Диплом/Diploma {SeriesDiploma} № {NumberDiploma} від/on {DateIssueDiploma}
Додаток/Supplement № {NumberDiplomaSupplement} від/on {DateIssueDiplomaSupplement}
(без диплома недійсний) / (not valid without the diploma)

1. ІНФОРМАЦІЯ ПРО ОСОБУ, ЯКІЙ ПРИСВОЄНО КВАЛІФІКАЦІЮ	1. INFORMATION IDENTIFYING THE HOLDER OF THE QUALIFICATION
1.1 Прізвище {LastNameUkraine}	1.1 Last name(s) {LastNameEnglish}
1.2 Ім’я {NameUkraine}	1.2 First name(s) {NameEnglish}

1.3 Дата народження (дд/мм/рррр) / 1.3 Date of birth (dd/mm/yyyy)
{DateOfBirth}

1.4 Код картки фізичної особи в Єдиній державній електронній базі з питань освіти /
1.4 Personal ID in Unified State Electronic Database on Education
{PersonalIdUSEDE}

Рисунок. 3.3. Частина шаблону додатку до диплома

УКРАЇНА
UKRAINE

ДИПЛОМ БАКАЛАВРА

{Series} № {Number}

{FirstNameUkraine}
{LastNameUkraine}

закінчи{Gender} у {CompletedYear} році
Рівненський державний гуманітарний університет

Освітня програма
{EducationProgrammeUkraine}
акредитована **{AccreditedByUkraine}**
здобу{Gender} кваліфікацію:
ступінь вищої освіти **бакалавр**
галузь знань (галузі знань)
{FieldStudyUkraine}
спеціальність (спеціальності)
{ProgrammeSubjectAreaUkraine}
спеціалізація **{SpecializationUkraine}**
професійна кваліфікація (у разі присвоєння)

{ProfessionalQualificationUkraine}

Додаткова інформація
{AdditionalInformationUkraine}

{HeadPositionUkraine}

BACHELOR'S DIPLOMA

{Series} № {Number}

{FirstNameEnglish}
{LastNameEnglish}

in {CompletedYear} completed the full course of
Rivne State University of the Humanities

Educational Programme
{EducationProgrammeEnglish}
accredited by **{AccreditedByEnglish}**
obtained qualification:
Bachelor`s Degree
Field(s) of Study
{FieldStudyEnglish}
Programme Subject Area(s)
{ProgrammeSubjectAreaEnglish}
Specialization **{SpecializationEnglish}**
Professional Qualification (in case of its awarding)
{ProfessionalQualificationEnglish}

Additional information
{AdditionalInformationEnglish}

{NameSurnamePositionUkranian} /
{NameSurnamePositionEnglish}

Рисунок. 3.4. Шаблон диплома

Для заміни ключових слів використовується бібліотека NPOI, вона відкриває потрібний шаблон за допомогою класу XWPFDocument, що дозволяє здійснити заміну вмісту документа. Заміну ключових слів реалізовано через метод FindAndReplaceText (рис. 3.5), який надає бібліотека, вона замінює ключові слова на дані отримані з сервісів. Для пошуку та заповнення таблиці, було реалізовано три методи, FindTable, AddRow та ReplaceCellTextPreservingStyle. Метод FindTable – використовується для пошуку таблиці за ключовим словом у ній. Метод AddRow – дозволяє додати рядок і зберегти властивості попереднього рядка. Метод

ReplaceCellTextPreservingStyle – дозволяє замінити текст в рядку, зберігаючи попередній стиль тексту. Процес заповнення шаблону включає отримання даних за допомогою сервісів: EducationProgrammeService, StudentDocumentService, StudentEvaluationsService та StudentService.

```
1 reference
private void SetEducationalProgrammeUkraine(string educationalProgramme)
{
    _document.FindAndReplaceText("{EducationProgrammeUkraine}", educationalProgramme);
}
1 reference
private void SetAccreditedByUkraine(string accreditedBy)
{
    _document.FindAndReplaceText("{AccreditedByUkraine}", accreditedBy);
}
```

Рисунок. 3.5. Заміна ключових слів методом FindAndReplaceText

При базовому тестуванні заміни слів, було виявлено, що базовий метод бібліотеки FindAndReplaceText перетворює символ перенесення на новий рядок за допомогою елементу <w:cr/>. Цей елемент не підтримує LibreOffice, яка відповідає за конвертування документа в інші формати. Було вирішено модернізувати метод бібліотеки NPOI. Було створено новий клас CustomXWPFDocument, який наслідується від XWPFDocument та модернізовано метод FindAndReplaceTextInParagraph, який використовувався методом FindAndReplaceText (рис 3.6).

Було реалізовано клас ConverterDocuments, який дозволяє перетворення документів формату docx в формат pdf за допомогою запуску зовнішнього процесу. Для цього використовується LibreOffice, який забезпечує конвертацію без необхідності взаємодії з графічним інтерфейсом.

Було реалізовано сервіси: GoogleDriveService EducationProgrammeService, StudentDocumentService, StudentEvaluationsService та StudentService, які відповідають за отримання та відправлення даних.

```
var replacedText = combinedText.Replace(oldValue, newValue);
var baseRun = paragraph.Runs[firstIndex.Value];

for (int i = lastIndex.Value; i >= firstIndex.Value; i--)
{
    paragraph.RemoveRun(i);
}

string[] lines = replacedText.Split('\n');

for (int i = lastIndex.Value; i >= firstIndex.Value; i--)
{
    paragraph.RemoveRun(i);
}


for (int i = 0; i < lines.Length; i++)
{
    var run = paragraph.CreateRun();

    SetStyle(run, baseRun);

    var text = run.GetCTR().AddNewT();
    text.Value = lines[i];
    text.space = "preserve";

    if (i < lines.Length - 1)
    {
        var br = run.GetCTR().AddNewBr();
        br.type = ST_BrType.textWrapping;
    }
}
```

Рисунок. 3.6. Частина модернізованого методу FindAndReplaceTextInParagraph

3.3 Тестування системи електронного документообігу

Для перевірки працездатності системи електронного документообігу було проведено тестування за допомогою інструментів Swagger.

Swagger — це незалежна від мови програмування специфікація для опису REST API. Вона дозволяє як комп'ютерам, так і людям розуміти можливості REST API без необхідності прямого доступу до його вихідного коду. Основні цілі Swagger полягають у наступному:

- мінімізувати обсяг роботи, необхідної для з'єднання незалежних сервісів;
- зменшити час, потрібний для точного документування сервісу [15].

Це потужний інструмент, який складається з кількох основних компонентів, кожен з яких відіграє важливу роль у документуванні та тестуванні API [16].

- Один із ключових компонентів Swagger – це Swagger Editor. Цей інструмент надає розробникам зручне середовище для створення та редагування специфікації OpenAPI, яка описує структуру та функціональність API. За допомогою Swagger Editor розробники можуть легко визначати доступні ендпоінти, параметри запитів, формати відповідей та інші ключові аспекти API [16].
- Ще одним важливим компонентом Swagger є Swagger UI. Цей інструмент дає змогу візуалізувати створену специфікацію OpenAPI у вигляді інтерактивної документації. Swagger UI надає зручний інтерфейс для дослідження API, надсилання запитів і перегляду відповідей прямо в браузері. Така документація робить процес взаємодії з API простішим і прозорішим для розробників та інших зацікавлених сторін [16].
- Swagger Codegen – ще один важливий компонент Swagger, який допомагає автоматизувати процес створення клієнтських бібліотек, серверних стабів та інших компонентів API на основі специфікації OpenAPI. Цей інструмент значно прискорює розробку та забезпечує узгодженість між різними частинами API [16].

  Swagger інтегрується за допомогою бібліотеки Swashbuckle, яка йде у стандартному пакеті фреймворку ASP.NET Core. Для тестування використовувались тестові дані, які мали повернути сервіси описані в пункті

3.2.1 та 3.2.2.

Процедура тестування складалась з наступних етапів:

1. Відправлення запиту через Swagger для створення диплома (рис 3.7):

- відправлявся запит на створення диплома за допомогою інтерфейсу Swagger. У запиті вказувався необхідний параметр, такий як StudentId;
- запит відправлся до контролеру Diploma, який відповідає за генерування диплома та зберігання в хмарному сховищі.

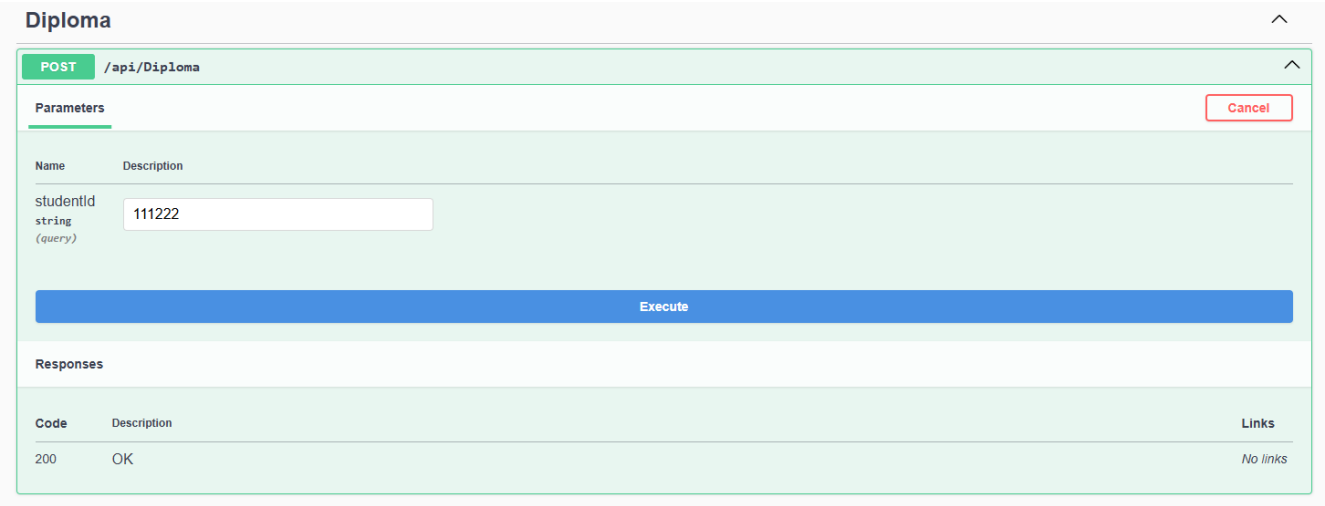

Рисунок. 3.7. Відправлення запиту для створення диплома

2. Перевірка відповіді на запит генерації диплома:

- після відправлення запиту система повертала посилання в хмарному сховищі файлу у форматі pdf, що містив заповнений шаблон документа (рис. 3.8). Було перевірено, чи всі ключові слова у шаблоні було коректно замінені на відповідні дані (рис. 3.9);

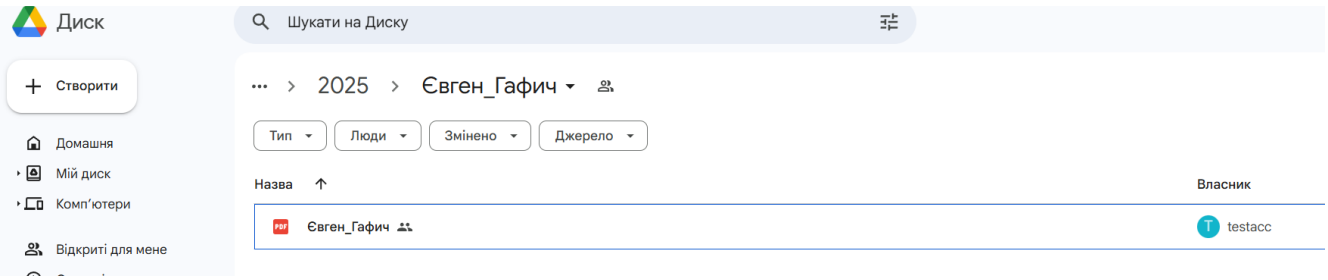

Рисунок. 3.8. Згенерований диплом у хмарному сховищі

- додатково перевірялось, чи файл відповідав очікуваному форматуванню та змісту.

УКРАЇНА
UKRAINE

ДИПЛОМ БАКАЛАВРА	**BACHELOR'S DIPLOMA**
ТС № 111222	ТС № 111222
Євген **Гафич**	**Yevhen** **Hafych**
закінчив у 2025 році **Рівненський державний гуманітарний університет**	in 2025 completed the full course of **Rivne State University of the Humanities**
Освітня програма **Інженерія програмного забезпечення** акредитована **Міністерство освіти і науки України** здобув кваліфікацію: ступінь вищої освіти **бакалавр** галузь знань (галузі знань) **Інформаційні технології**	Educational Programme **Software Engineering** accredited by **Ministry of Education and Science of Ukraine** obtained qualification: **Bachelor`s** Degree Field(s) of Study **Information Technology**

Рисунок. 3.9. Згенерований диплом у форматі pdf

3. Відправлення запиту через Swagger для створення додатку до диплома (рис. 3.10):

- відправлявся запит на створення додатку до диплома з інтерфейсу Swagger. У запиті вказувався необхідний параметр, такий як рік випуску диплома і додатку;
- запит відправлявся до контролеру DiplomaSupplement, який відповідає за генерування додатку до диплома та зберігання в хмарному сховищі.

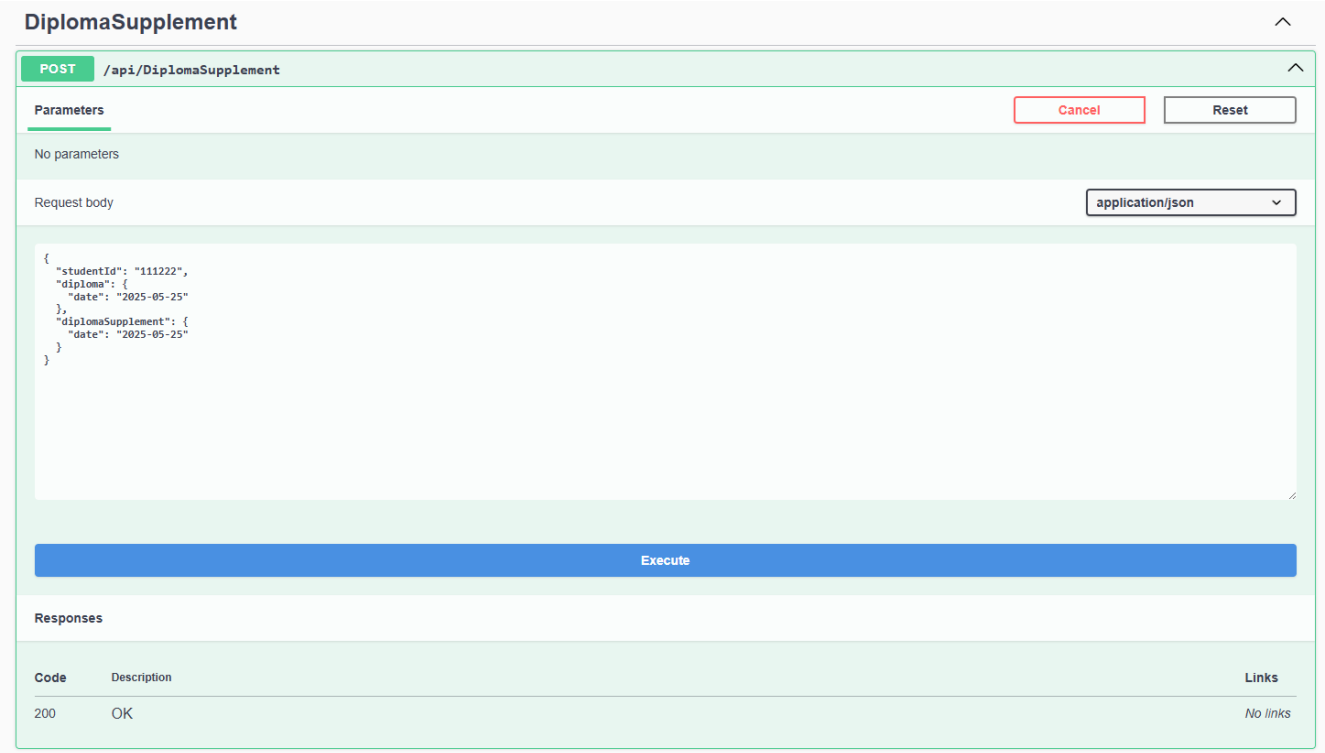

Рисунок. 3.10. Відправлення запиту на генерування додатку до диплома

4. Перевірка відповіді на запит генерації додатку до диплома:

- після запиту, система очікувано повертала посилання на документ в хмарному сховищі у форматі pdf, який був заповнений в потрібних полях (рис. 3.11). Було перевірено, чи всі ключові слова були замінені на відповідні дані (рис. 3.12);

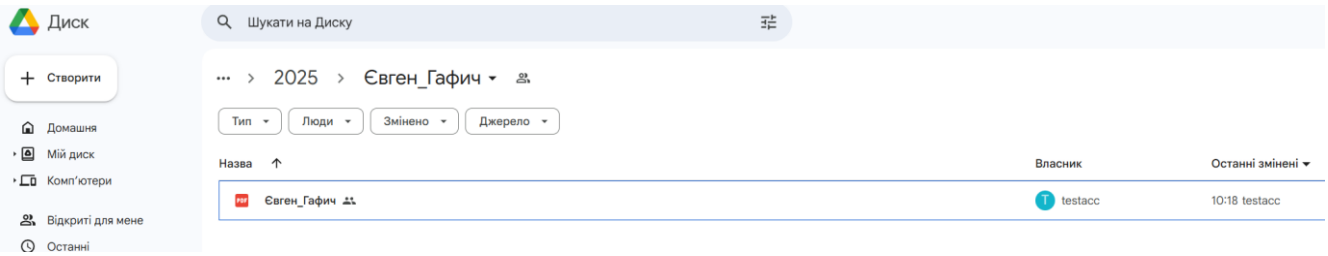

Рисунок. 3.11. Згенерований додаток до диплома у хмарному сховищі

- додатково, була перевірка чи файл відповідав форматуванню та змісту.

УКРАЇНА

UKRAINE

ДОДАТОК ДО ДИПЛОМА /
DIPLOMA SUPPLEMENT

Диплом/Diploma ТС № 111222 від/on 25/05/2025
Додаток/Supplement № 222333 від/on 25/05/2025
(без диплома недійсний) / (not valid without the diploma)

1. ІНФОРМАЦІЯ ПРО ОСОБУ, ЯКІЙ ПРИСВОЄНО КВАЛІФІКАЦІЮ	1. INFORMATION IDENTIFYING THE HOLDER OF THE QUALIFICATION
1.1 Прізвище	1.1 Last name(s)
Гафич	Hafych
1.2 Ім'я	1.2 First name(s)
Євген	Yevhen
1.3 Дата народження (дд/мм/рррр) / 1.3 Date of birth (dd/mm/yyyy)	
01/01/2000	
1.4 Код картки фізичної особи в Єдиній державній електронній базі з питань освіти / 1.4 Personal ID in Unified State Electronic Database on Education	
12211221	
2. ІНФОРМАЦІЯ ПРО ПРИСВОЄНУ КВАЛІФІКАЦІЮ	2. INFORMATION IDENTIFYING THE QUALIFICATION
2.1 Назва кваліфікації та присвоєний ступінь	2.1 Name of qualification and (if applicable) title conferred
Бакалавр з інженерії програмного забезпечення	Bachelor of Science in Software Engineering
2.1.1 Ступінь вищої освіти	2.1.1 Degree
Бакалавр	Bachelor
2.1.2 Спеціальність	2.1.2 Programme Subject Area
121 Інженерія програмного забезпечення (код та найменування)	121 Software Engineering (code and name)
2.1.3 Спеціалізація	2.1.3 Specialization
Освітня програма: Інженерія програмного забезпечення	Educational Program: Software Engineering

Рисунок. 3.12. Згенерований додаток до диплома в форматі pdf

Результати тестування

Тестування показало, що після відправлення запиту через Swagger документ генерувався у форматі pdf з правильно заповненими даними. Заміна ключових слів у шаблоні на реальні дані відбувалась коректно, що підтверджує тестування. Документи успішно завантажувались у хмарне сховище Google Drive, у потрібну

папку.

Конвертація файлу з docx формату до pdf за допомогою LibreOffice пройшла, також, успішно. Отриманий pdf документ відповідав усім вимогам форматування та змісту.

ВИСНОВКИ

У процесі виконання кваліфікаційної роботи було розроблено та протестовано систему підтримки електронного документообігу, призначеної для автоматизації створення дипломів та додатків до них. Основна увага приділялась архітектурі системи, використанню сучасних технологій та інструментів, забезпеченню коректності обробки даних та створення документів у зручному форматі.

У рамках дослідження було досягнуто наступних результатів:

1. досліджено поняття електронного документообігу;
2. проведено аналіз та вибір технологій для реалізації: мова програмування c#, фреймворк ASP.NET Core, бібліотеки NPOI для роботи з шаблонами формату docx і Google.Apis.Drive.v3 для роботи з хмарним сховищем та засіб LibreOffice для конвертації документів у зручний формат;
3. розроблено архітектуру системи на основі мікросервісів, що забезпечує гнучкість та можливість масштабування;
4. створено абстракті моделі, які повторюють шаблон документа;
5. створено сервіси для взаємодії з іншими мікросервісами для отримання даних про студентів, їх оцінки, освітню спеціальність та завантаження документів у хмарне сховище;
6. проведено тестування системи, що підтвердило правильність обробки запитів, генерації документів, конвертації у інший формат та завантаження у хмарне сховище.

Результати роботи демонструють, що система здатна ефективно виконувати завдання, пов’язані з генерацією дипломів та додатків до них, підтримувати інтеграцію з іншими сервісами для отримання даних, зберігати документи у відповідному форматі та завантажувати у хмарне сховище.

Подальші напрямки розвитку включають: забезпечення перевірки отриманих даних з мікросервісів у разі некоректності чи відсутності повідомляти користувача з зазначенням поля у якому виникла помилка, логування системи,

вдосконалення системи з точки зору безпеки та оптимізацію процесу обробки даних для підвищення продуктивності.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Інформаційні системи, їх види. Апаратне та програмне забезпечення інформаційної системи. Київські учнівські олімпіади з інформаційних технологій і вивчення інформатики. URL: https://www.kievoit.ippo.kubg.edu.ua/kievoit/2013/95/95.html (дата звернення: 06.05.2025).
2. ІНФОРМАЦІЙНІ СИСТЕМИ І ТЕХНОЛОГІЇ / П. М. Павленко та ін. URL: https://files.znu.edu.ua/files/Bibliobooks/Inshi71/0052123.pdf (дата звернення: 07.05.2025).
3. Електронний документообіг: системи, види, особливості впровадження та як він працює в Україні. InBase. URL: https://inbase.com.ua/elektronnyj-dokumentoobig-systemy-vydy-osoblyvosti-vprovadzhennya-ta-yak-vin-praczyuye-v-ukrayini/ (дата звернення: 10.05.2025).
4. Козак О. Л. Аналіз вимог до програмного забезпечення. URL: https://dspace.wunu.edu.ua/retrieve/14135/FCIT_kKN_sPZS_dAVPZ_%20LEC.pdf (дата звернення: 08.05.2025).
5. Краснов О. В. Сучасні мови програмування. Поняття об’єкта в мові програмування, його властивостей і методів. СУЧАСНА ІНФОРМАТИКА (сила практики в теорії). URL: https://alextexnok.blogspot.com/p/blog-page_537.html (дата звернення: 11.05.2025).
6. Визначення нефункціональних вимог. *Онлайн-курси від компанії QATestLab | Головна сторінка*. URL: https://training.qatestlab.com/blog/technical-articles/non-functional-requirements-examples-types-approaches/ (дата звернення: 23.05.2025).
7. Щербаков О. В., Парфьонов Ю. Е., Федорченко В. М. ОСНОВИ ОБ'ЄКТНО-ОРІЄНТОВАНОГО ПРОГРАМУВАННЯ. URL: https://library.megu.edu.ua:9443/jspui/handle/123456789/4125 (дата звернення: 12.05.2025).

8. Дегтярьова Л. М., Гроза П. М., Сомов С. В. Навчальний посібник з дисципліни «Технології розробки програмного забезпечення» для студентів спеціальності 123 «Комп'ютерна інженерія». Полтава : ПолтНТУ, 2017. 212 с. URL: https://reposit.nupp.edu.ua/handle/PoltNTU/4461 (дата звернення: 11.05.2025).
9. Вівчарик В. Повний огляд REST: нюанси, поради, приклади. URL: https://dou.ua/forums/topic/50364/ (дата звернення: 12.05.2025).
10. Co to jest API? Wszystko o interfejsie programowania aplikacji - Szkoła IT Coders Lab. Kursy programowania – szkolenia, kursy IT (informatyczne) online, bootcamp programistyczny – Szkoła Programowania Coders Lab. URL: https://coderslab.pl/pl/blog/co-to-jest-api-wszystko-o-interfejsie-programowania-aplikacji (data dostępu: 14.05.2025).
11. Zmerzlyi I. Мікросервісна архітектура. Medium. URL: https://medium.com/@IvanZmerzlyi/microservices-architecture-461687045b3d (дата звернення: 15.05.2025).
12. Controller | LoopBack Documentation. LoopBack. URL: https://loopback.io/doc/en/lb4/Controller.html (date of access: 15.05.2025).
13. Controllers in ASP.NET core MVC. Dot Net Tutorials. URL: https://dotnettutorials.net/lesson/controllers-asp-net-core-mvc/ (date of access: 13.05.2025).
14. Коноваленко І. В. Програмування мовою C# 6.0. Тернопіль : ТНТУ, 2016. 227 с.
15. ASP.NET Core web API documentation with Swagger / OpenAPI. URL: https://learn.microsoft.com/en-us/aspnet/core/tutorials/web-api-help-pages-using-swagger?view=aspnetcore-8.0&viewFallbackFrom=aspnetcore-10.0 (date of access: 23.06.2025).
16. Що таке Swagger: Тестування API. FoxmindEd. URL: https://foxminded.ua/shcho-tak-swagger/ (дата звернення: 23.05.2025).

17. МОН України. Форми документів про вищу освіту. URL: https://mon.gov.ua/osvita-2/vishcha-osvita-ta-osvita-doroslikh/formi-dokumentiv-pro-vishchu-osvitu (дата звернення: 16.05.2025).
18. Richards M., Ford N. Fundamentals of Software Architecture: An Engineering Approach. O'Reilly Media, 2020. 432 p.
19. Newman S. Building Microservices. O'Reilly Media, Incorporated, 2015. 280 p.
20. Getting Started with NPOI. GitHub. URL: https://github.com/nissl-lab/npoi/wiki/Getting-Started-with-NPOI (date of access: 17.05.2025).
21. API design guide | Cloud API Design Guide | Google Cloud. Google Cloud. URL: https://cloud.google.com/apis/design (date of access: 22.05.2025).