Рівненський державний гуманітарний університет
Факультет математики та інформатики
Кафедра інформаційних технологій та моделювання

Кваліфікаційна робота

за освітнім ступенем «бакалавр»

на тему: «**Інформаційна система підтримки електронного документообігу закладу вищої освіти. Розроблення та верифікація освітньо-професійних програм, навчальних планів**»

Виконав:
Здобувач IV курсу
групи ІПЗ-41
спеціальності 121 Інженерія програмного забезпечення
Зінчук Валерій Олегович
Науковий керівник:
к.т.н., доцент **Сяський Володимир Андрійович**

м. Рівне – 2025 рік

Зміст

СПИСОК ТЕРМІНІВ ТА УМОВНИХ ПОЗНАЧЕНЬ

ЗВО – заклад вищої освіти

ОПП – освітньо-професійна програма

НП – навчальний план

ІНП – індивідуальний навчальний план

ОК – обов’язковий компонент

ВК – вибірковий компонент

ПРН – програмний результат навчання

ECTS – Європейська система трансферу та накопичення кредитів (European Credit Transfer and Accumulation System)

ЕЦП / КЕП – електронний цифровий підпис / кваліфікований електронний підпис

EDMS – система електронного документообігу (Electronic Document Management System)

JSON – формат обміну структурованими даними (JavaScript Object Notation)

PDF – формат електронного документа (Portable Document Format)

ORM – об’єктно-реляційне відображення (Object-Relational Mapping)

API – інтерфейс прикладного програмування (Application Programming Interface)

HTTP – протокол передавання гіпертексту (HyperText Transfer Protocol)

SQL – мова структурованих запитів (Structured Query Language)

CI/CD – безперервна інтеграція / безперервне розгортання (Continuous Integration / Continuous Deployment)

fitz – бібліотека PyMuPDF для обробки PDF-документів

Flask – веб-фреймворк на Python для створення серверних застосунків

Peewee – легка ORM-бібліотека для роботи з базами даних у Python

Aspose – бібліотека для професійної обробки документів, зокрема PDF

ВСТУП

У сучасних закладах вищої освіти дедалі більше уваги приділяється цифровій трансформації освітніх і адміністративних процесів. Одним із ключових напрямів цієї трансформації є впровадження автоматизованих інформаційних систем для підтримки електронного документообігу. Такі системи дозволяють суттєво скоротити витрати часу на підготовку, обробку та верифікацію навчальної документації, підвищити прозорість процедур та мінімізувати кількість помилок.

В умовах постійних змін у стандартах вищої освіти, появи нових компетентностей і програмних результатів навчання, особливого значення набуває автоматизація процесів формування та перевірки освітньо-професійних програм (ОПП) і навчальних планів (НП). Ці документи становлять основу організації навчального процесу й потребують регулярного оновлення, перевірки на відповідність чинним вимогам і зручного представлення для подальшого аналізу або експорту до інших систем.

Інформаційні системи, орієнтовані на підтримку електронного документообігу в ЗВО, повинні забезпечувати не лише зберігання та перегляд ОПП і НП, а й автоматичне формування індивідуальних навчальних планів (ІНП), відстеження навчальних досягнень студентів, а також інтеграцію з іншими підсистемами закладу. До основних функціональних можливостей таких рішень належить:

- створення й редагування освітніх програм і навчальних планів;
- автоматичне формування ІНП на основі затверджених НП;
- ведення обліку семестрового контролю;
- підтримка рейтингової оцінки результатів навчання.

Користувачами системи переважно є структурні підрозділи ЗВО — кафедри, деканати, навчальні відділи, а також гаранти освітніх програм, які

потребують зручного інструменту для внесення змін до документації, перевірки її повноти та актуальності. У зв’язку з цим особливої важливості набуває розроблення систем, які забезпечують централізований підхід до супроводу освітнього процесу, прозору взаємодію між учасниками та можливість інтеграції в загальну цифрову екосистему навчального закладу.

Актуальність дослідження

Необхідність цифровізації документообігу в ЗВО обумовлена великою кількістю навчальної документації, яка створюється, оновлюється й використовується протягом усього освітнього процесу. Більшість таких документів, зокрема ОПП і НП, досі готуються у форматі PDF або Word-файлів, що ускладнює їх автоматизовану обробку, пошук, перевірку та обмін між підрозділами. Автоматизовані інформаційні системи, здатні працювати з неструктурованими документами, дозволяють вирішити цю проблему та підвищити ефективність освітнього адміністрування.

Крім того, інтеграція таких систем у цифрове середовище ЗВО сприяє уніфікації підходів до зберігання та перевірки документації, створенню централізованої бази даних освітніх компонентів, а також забезпеченню доступу до актуальної інформації для гаранта програми, викладача чи студента.

Мета дослідження

Метою даної роботи є розроблення інформаційної підсистеми для автоматизації процесів створення, верифікації та структурування освітньо-професійних програм і навчальних планів, яка інтегрується до загальної системи електронного документообігу закладу вищої освіти.

Завдання дослідження

Для досягнення мети були поставлені такі завдання:

- провести огляд нормативної бази та наявних рішень у сфері автоматизації документообігу ЗВО;

– визначити вимоги до інформаційної підсистеми для підтримки ОПП і НП;

– реалізувати механізм обробки PDF-документів для вилучення освітніх компонентів;

– створити універсальний формат зберігання даних (словник/JSON);

– забезпечити доступ до обробленої інформації через API;

– протестувати систему та проаналізувати можливості її інтеграції з іншими підсистемами.

Об’єкт дослідження

Інформаційна система для підтримки електронного документообігу у ЗВО, яка забезпечує автоматизовану обробку, зберігання та взаємодію з навчальними документами.

Предмет дослідження

Процес розробки та впровадження підсистеми для автоматизації обробки ОПП і НП, включаючи методи парсингу, структурування, формування словника даних і побудову API для доступу до результатів.

Методи дослідження

Під час виконання роботи використовувались:

– аналіз чинних інформаційних систем у сфері освітнього документообігу;

– методи структурного та об’єктно-орієнтованого проєктування;

– застосування бібліотек PyMuPDF, Aspose.PDF, Flask, Peewee для побудови прототипу;

– моделювання структури даних, побудова API та перевірка функціональності за допомогою тестових документів.

Практичне значення дослідження

Результатом роботи стало створення робочого прототипу підсистеми, що дозволяє автоматично обробляти PDF-документи з навчальними планами, виявляти обов’язкові та вибіркові компоненти, структурувати їх у форматі словника, а також забезпечити взаємодію з іншими підсистемами ЗВО. Запропоноване рішення може бути легко адаптоване до конкретних потреб університету, забезпечуючи прозорість, масштабованість і спрощення процесів адміністрування навчального процесу.

РОЗДІЛ 1
ТЕОРЕТИЧНІ ВІДОМОСТІ

1.1 Понятійний апарат та нормативно-правова база забезпечення електронного документообігу в ЗВО

У сучасних умовах цифровізації освіти електронний документообіг (ЕДО) стає невід’ємною складовою функціонування закладів вищої освіти (ЗВО). Його впровадження спрямоване на підвищення ефективності управлінських процесів, оптимізацію обміну інформацією, зменшення витрат часу та паперових ресурсів, а також забезпечення прозорості й збереження даних. Для глибшого розуміння цієї сфери необхідно розглянути понятійний апарат та нормативно-правову базу, яка регламентує ЕДО в ЗВО.

Основні поняття

Електронний документообіг — це технологічний процес створення, обробки, зберігання, передавання та використання електронних документів за допомогою комп’ютерних інформаційних систем із дотриманням вимог чинного законодавства.

Електронний документ — документ, інформація в якому зафіксована у формі електронних даних, включаючи обов’язкові реквізити та електронний підпис (у разі потреби), що забезпечує його юридичну силу.

Електронний цифровий підпис (ЕЦП) або **кваліфікований електронний підпис (КЕП)** — вид електронного підпису, що створюється за допомогою засобів криптографічного захисту і має таку саму юридичну силу, як і власноручний підпис.

Інформаційна система електронного документообігу — сукупність програмних засобів, серверного та клієнтського обладнання, яка забезпечує створення, реєстрацію, зберігання, передачу, пошук та облік документів у цифровій формі.

У контексті ЗВО така система зазвичай охоплює внутрішній обіг наказів, заяв, розпоряджень, службових записок, звітів, навчальних планів та освітніх програм.

Нормативно-правове забезпечення електронного документообігу в ЗВО

В Україні правові засади впровадження електронного документообігу визначаються низкою нормативно-правових актів загального та спеціального призначення:

1. Закон України «Про електронні документи та електронний документообіг» (2003 р.) — основний закон, що визначає правовий статус електронного документа, порядок його створення, зберігання, передавання і підтвердження юридичної сили; [1]
2. Закон України «Про електронні довірчі послуги» (2017 р.) — встановлює вимоги до КЕП, кваліфікованих надавачів електронних довірчих послуг, а також регулює використання електронної ідентифікації; [2]
3. Закон України «Про інформацію», «Про захист інформації в інформаційно-телекомунікаційних системах», «Про доступ до публічної інформації» — регулюють питання доступу, захисту та збереження інформації, зокрема в електронній формі; [3][4]

4. Постанова Кабінету Міністрів України №55 від 17.01.2018 року «Про деякі питання документообігу в органах виконавчої влади» — окреслює механізми впровадження електронного документообігу в державних установах, включаючи освітні органи; [8]
5. Наказ Міністерства освіти і науки України №676 від 01.06.2013 року «Про затвердження Інструкції з діловодства у вищих навчальних закладах» — містить вимоги до обліку документів, у тому числі можливість ведення документації в електронній формі; [7]
6. Стандарти ISO/IEC 27001 та ДСТУ 2732:2004 — визначають вимоги до безпеки інформації та структури управління документацією; [5][6]
7. Указ Президента України «Про концепцію розвитку цифрової економіки та суспільства України» (2018 р.) — визначає стратегічний курс держави на цифрову трансформацію, зокрема в сфері освіти. [9]

У контексті ЗВО також важливе значення мають локальні нормативні акти, такі як положення про електронний документообіг, регламенти роботи внутрішніх інформаційних систем, інструкції для користувачів, внутрішні протоколи цифрової безпеки тощо.

Значення нормативної бази для ЗВО

Чітко сформована нормативно-правова база забезпечує правову визначеність та юридичну силу дій, здійснюваних в електронному середовищі. Це дозволяє ЗВО:

- гарантувати цілісність, достовірність і конфіденційність інформації;
- забезпечувати збереження цифрових архівів;
- організувати прозорий документообіг із викладачами, студентами та зовнішніми партнерами;
- впроваджувати інноваційні ІТ-рішення з урахуванням вимог держави.

Таким чином, нормативно-правове забезпечення є основою для ефективної цифрової трансформації управлінських процесів у закладах вищої освіти та для побудови довіри до електронного середовища документообігу.

1.2 Інформаційні системи підтримки електронного документообігу

У сучасних закладах вищої освіти (ЗВО) зростає потреба в ефективному управлінні освітньою, адміністративною та науковою документацією. З огляду на значні обсяги документації, а також необхідність дотримання нормативних вимог, автоматизація документообігу є стратегічним напрямом цифрової трансформації вищої освіти. Саме тому розробка та впровадження інформаційних систем підтримки електронного документообігу (ІС ЕДО) стає критично важливою.

Класифікація інформаційних систем ЕДО

ІС ЕДО можна класифікувати за кількома ознаками:

- за рівнем автоматизації: локальні (для окремих підрозділів) та інтегровані (єдині системи на рівні всього ЗВО);
- за сферою застосування: адміністративні, навчальні, кадрові, архівні;
- за типом реалізації: настільні програми, web-орієнтовані системи, хмарні платформи;
- за способом взаємодії: системи з ручним введенням даних, системи з автоматичним витягом інформації з документів (наприклад, із PDF).

У закладах освіти найбільш затребуваними є комплексні платформи, які дозволяють працювати з широким спектром документів — наказами, положеннями, навчальними планами, заявами, контрактами, звітами тощо.

Функціональні можливості ІС ЕДО у ЗВО

Сучасна ІС ЕДО у ЗВО зазвичай включає такі основні компоненти:

- модуль обліку вхідної та вихідної кореспонденції;
- модуль внутрішнього обігу документів (заяви, розпорядження, звіти);
- модуль зберігання і пошуку документів за реквізитами;
- підтримка кваліфікованого електронного підпису;
- контроль термінів виконання та етапів погодження;
- рольовий доступ до документів (кафедра, деканат, студент, гаранти програм);
- модуль звітності (наприклад, для акредитаційних процедур);
- інтеграція з іншими системами ЗВО (електронний журнал, LMS, кадрова система).

Ці модулі дають змогу скоротити час на опрацювання документів, мінімізувати помилки, автоматизувати рутинні дії, а також забезпечити прозорість усіх етапів документообігу.

Приклади використання ІС ЕДО в освіті

На практиці ЗВО використовують як готові рішення, так і власні розробки. Приклади типових систем:

- Мегаполіс.Документообіг — вітчизняна система, яку адаптували під потреби МОН, університетів і академій;
- Електронний університет — комплексне рішення, яке поєднує документообіг, студентський облік, бібліотечні сервіси та фінанси;
- Document.online, Інфодок, EDM.Soft — універсальні системи з можливістю адаптації під ЗВО;
- Moodle, Google Workspace for Education — платформи, які хоч і не є ІС ЕДО в повному розумінні, але мають інтегровані елементи документообігу (формування та зберігання звітів, заяв, результатів навчання).

Крім цього, деякі університети впроваджують **власні внутрішні системи**, створені на основі Python, Flask або Django, які адаптовані до унікальних вимог

структурних підрозділів. Основною перевагою таких систем є повна відповідність внутрішнім процесам і гнучкість до змін у навчальному та адміністративному середовищі.

Проблеми при впровадженні ІС ЕДО в ЗВО

Попри наявність ефективних технологій, у ЗВО залишаються такі типові проблеми:

- фрагментарність ІТ-середовища: різні підрозділи використовують несумісні системи;
- відсутність стандартизованої структури цифрових навчальних планів;
- низький рівень підготовки персоналу до роботи з електронними системами;
- складність перенесення архівних документів у цифрову форму;
- обмежені ресурси для закупівлі готових систем або створення власних;
- проблеми з інформаційною безпекою та захистом персональних даних.

Усі ці труднощі свідчать про те, що впровадження ІС ЕДО в ЗВО вимагає комплексного підходу — технічного, організаційного та нормативного.

Роль ІС ЕДО у цифровій трансформації ЗВО

Ефективна ІС ЕДО є основою цифрового середовища ЗВО, оскільки:

- формує єдиний простір зберігання та обміну інформацією;
- дозволяє автоматизувати супровід ОПП та НП;
- забезпечує прозорість та контроль управлінських дій;
- підвищує мобільність доступу до даних з будь-якої точки;
- зменшує залежність від паперових архівів;
- дозволяє швидко адаптуватися до змін в освітній політиці або вимогах акредитації.

Таким чином, інформаційні системи електронного документообігу є не просто засобом цифровізації, а базовим елементом сучасної освітньої

інфраструктури. Вони забезпечують гнучкість, масштабованість і ефективність в управлінні всіма процесами освітньої діяльності.

1.3 Розроблення та верифікація освітньо-професійних програм і навчальних планів спеціальностей

Розроблення освітньо-професійних програм (ОПП) та навчальних планів (НП) — це ключові процеси в діяльності будь-якого закладу вищої освіти, що безпосередньо впливають на якість підготовки фахівців та відповідність освітнього процесу державним і міжнародним стандартам.

Процес створення ОПП та НП вимагає не лише методичного і наукового підходу, а й дотримання низки нормативних вимог, що регламентуються Міністерством освіти і науки України, Національним агентством із забезпечення якості вищої освіти (НАЗЯВО), а також Національною рамкою кваліфікацій.

Особливості розроблення освітньо-професійних програм

Освітньо-професійна програма — це комплекс нормативних документів, що визначає:

- мету та завдання підготовки здобувача;
- обсяг, зміст і логіку навчального процесу;
- очікувані програмні результати навчання (ПРН);
- перелік навчальних компонентів (ОК і ВК);
- вимоги до оцінювання, кваліфікаційної роботи та випуску.

У розробленні ОПП беруть участь:

- гаранти освітніх програм;
- проектні групи (викладачі, науковці, представники ринку праці);
- кафедри та навчально-методичні відділи;
- зовнішні стейкхолдери (роботодавці, випускники, студенти).
- Основні документи, які регламентують зміст ОПП:

- Стандарти вищої освіти за відповідною спеціальністю;
- Національна рамка кваліфікацій (НРК);
- Методичні рекомендації МОН;
- Локальні положення ЗВО.

Важливим елементом структури ОПП є матриця відповідності ПРН і навчальних дисциплін, яка забезпечує логіку формування компетентностей.

Розроблення навчальних планів спеціальностей

Навчальний план — це структурований документ, що відображає реалізацію ОПП у вигляді переліку навчальних дисциплін, розподілу їх по семестрах, обсягів кредитів ECTS та форм контролю.

Його структура зазвичай включає:

- загальну інформацію (назва спеціальності, форма навчання, тривалість, мова);
- перелік обов’язкових компонентів (ОК);
- перелік вибіркових компонентів (ВК);
- розподіл дисциплін за семестрами;
- тип контролю (іспит, залік, курсовий проєкт тощо);
- індивідуальні навчальні траєкторії;
- інтеграцію з дуальною освітою або практиками.
- список:
- 240 кредитів — магістерський рівень (2+4 роки);
- 180 кредитів — бакалаврський рівень (3 роки);
- 60 кредитів — освітньо-наукові програми (PhD-1 рік).

Навчальний план формується в тісній координації з проектною групою ОПП та затверджується на рівні вченої ради ЗВО. Його зміст має узгоджуватись зі змінами в законодавстві, галузевих стандартах, потребах ринку праці та результатами зовнішнього моніторингу.

Автоматизація процесів створення та верифікації ОПП і НП

З огляду на складність структури та великі обсяги навчальної інформації, сучасні ЗВО поступово переходять до автоматизованого супроводу розробки й оновлення ОПП та НП. Це дозволяє:

- зменшити кількість помилок при формуванні документації;
- пришвидшити погодження між підрозділами;
- забезпечити актуальність ПРН відповідно до нових стандартів;
- генерувати шаблони НП з автоматичним розрахунком кредитів;
- перевіряти дублювання дисциплін або перевищення кредитного навантаження;
- формувати експорт у PDF, Word або JSON-форматах;
- зберігати навчальні документи в структурованому архіві.

Верифікація освітніх документів

Верифікація — це процедура перевірки відповідності ОПП і НП встановленим стандартам та внутрішнім вимогам. Вона передбачає:

- аналіз відповідності ПРН стандарту;
- оцінку логічної цілісності освітньої траєкторії;
- перевірку коректності обсягів кредитів;
- контроль за використанням національної термінології;
- погодження з роботодавцями та академічними радами.

Верифікація може здійснюватися як вручну (методичною комісією), так і автоматизовано — за допомогою програмних засобів, які здійснюють синтаксичний або семантичний аналіз документа.

На практиці використання інструментів на основі Python, регулярних виразів, парсерів PDF, JSON-моделей значно полегшує процедуру верифікації та дозволяє впроваджувати електронну підтримку під час акредитації.

РОЗДІЛ 2
ІНСТРУМЕНТИ ДЛЯ РЕАЛІЗАЦІЇ ЗАВДАННЯ

2.1 Життєвий цикл програмного забезпечення

Життєвий цикл програмного забезпечення (ПЗ) — це сукупність послідовних етапів, які охоплюють процес створення, впровадження, експлуатації та супроводу програмного продукту. Розуміння та управління життєвим циклом дозволяє досягти якості, передбачуваності та повторюваності результатів у розробці ПЗ.

Життєвий цикл ПЗ:

- аналіз потреб користувача;
- планування і техніко-економічне обґрунтування;
- проєктування архітектури системи;
- розробка і тестування;
- впровадження та підтримка;
- модернізація або завершення експлуатації.

На першому етапі — аналіз вимог — визначаються функціональні та нефункціональні потреби кінцевих користувачів. Цей етап є ключовим, оскільки від якості сформульованих вимог залежить подальша розробка.

Далі здійснюється архітектурне і технічне проєктування, де визначаються структура програми, вибір мови програмування, інтерфейсні компоненти, взаємодія між модулями, базою даних та API.

Етап реалізації включає безпосереднє написання коду та створення функціональних модулів. Після цього виконується **тестування**, яке може охоплювати юніт-тести, інтеграційне тестування, системне й регресійне тестування.

Впровадження означає встановлення програмного продукту в робоче середовище ЗВО, де він взаємодіє з користувачами. У подальшому відбувається

підтримка, що включає виправлення помилок, оновлення функціоналу та адаптацію до змін у нормативній базі.

Життєвий цикл ПЗ також враховує **етап завершення**, коли продукт виводиться з експлуатації, а дані переносяться до нової системи.

Таким чином, структурований підхід до управління життєвим циклом дає змогу контролювати якість, витрати та терміни реалізації, що є критично важливим для освітніх інформаційних систем.

2.2 Методологія розробки

Методологія розробки програмного забезпечення визначає загальний підхід до організації процесу створення, тестування і впровадження програмного продукту. Вибір методології залежить від складності системи, її цілей, ресурсів, часу і командної структури.

Основні підходи:

- Waterfall (каскадна модель) — класичний поетапний підхід, де кожен етап починається після завершення попереднього;
- Iterative & Incremental — модульна розробка із поступовим удосконаленням функціоналу;
- Agile (Scrum, Kanban) — гнучкий підхід, що передбачає регулярні ітерації (спринти), адаптивне планування і зворотній зв’язок;
- DevOps — поєднання розробки і експлуатації з акцентом на автоматизацію CI/CD, моніторинг і безперервне вдосконалення.

У проєкті з підтримки електронного документообігу доцільно обрати ітераційно-інкрементальний підхід у поєднанні з принципами Agile. Це дозволяє:

- швидко реагувати на зміни у вимогах;
- поступово розширювати функціональність;
- залучати користувача до перевірки результатів кожного етапу;
- вчасно виявляти та виправляти помилки.

Завдяки такому підходу, забезпечується гнучкість і контрольованість розробки без втрати загальної цілісності системи.

2.3 Функціональні вимоги

Функціональні вимоги описують, що саме повинна виконувати система — тобто набір функцій, які вона повинна реалізувати для задоволення потреб користувачів.

У системі підтримки електронного документообігу в ЗВО основними функціональними вимогами є:

- парсинг PDF-документів з освітніми програмами;
- витяг і обробка інформації про обов’язкові та вибіркові компоненти (ОК, ВК);
- зберігання структурованих даних у форматі JSON або базі даних;
- побудова REST API для доступу до даних;
- можливість фільтрації та пошуку компонентів за семестром, формою контролю тощо;
- інтерфейс завантаження та обробки PDF-файлів;
- генерація JSON-виводу з навчальних документів;
- логування дій користувача і журнал змін.

Крім цього, система має забезпечити коректну обробку різних шаблонів документів (від різних факультетів) та підтримку української мови в текстовому розпізнаванні.

Функціональні вимоги є основою для побудови архітектури, розробки API та реалізації контролерів у Flask. Їх дотримання гарантує, що система буде ефективно вирішувати поставлену задачу та відповідати очікуванням користувачів.

2.4 Нефункціональні вимоги

У даній системі до основних нефункціональних вимог належать:

- надійність — система повинна стабільно працювати з різними PDF-файлами без збоїв;
- продуктивність — обробка одного документа не повинна перевищувати 5–10 секунд;
- масштабованість — можливість додавання нових модулів (наприклад, аналіз ПР, формування ІНП);
- захист даних — підтримка авторизації, обмеження доступу, захист API;
- відповідність стандартам — відповідність форматів JSON до вимог обміну між ЗВО або МОН;
- портативність — можливість запуску на різних ОС (Linux, Windows) та середовищах (локально або у хмарі);
- зрозумілий інтерфейс — простота використання навіть без технічної підготовки;
- розширюваність — легко доповнювана структура словників ОК/ВК.

Також важливим є збереження сумісності з іншими платформами, наприклад LMS або цифровими кабінетами студентів. Всі ці вимоги враховувалися при проєктуванні структури Flask-додатку, виборі формату JSON та ORM-бібліотеки Peewee.

2.5 Конструювання інтегрованих комп’ютерних інформаційних систем

Інтегрована комп’ютерна інформаційна система (ІКІС) — це система, що поєднує в собі декілька підсистем, які взаємодіють між собою через єдину інформаційну платформу.

У контексті ЗВО така система може включати:

- модуль документообігу (ОПП, НП, заяви, розпорядження);
- навчальний модуль (відомості, оцінки, ІНП);
- кадровий модуль (викладацьке навантаження, структура кафедр);
- модуль статистики та аналітики (звіти для МОН, НАЗЯВО).
- Для побудови ІКІС важливо дотримуватись принципів:
- єдиного формату даних (наприклад, JSON);
- використання API для взаємодії між модулями;
- логічного поділу на мікросервіси, де кожен відповідає за окрему функцію;
- централізованої аутентифікації та ролей доступу;
- резервного копіювання і журналювання подій.

У проєкті система спроєктована з урахуванням можливості розширення до повноцінної ІКІС, де парсер PDF — лише один із модулів, що взаємодіє з базою навчальних програм, модулем верифікації або кабінетом адміністратора.

2.6 Конструювання веб-компонентів інтегрованих систем

Веб-компоненти — це елементи інтерфейсу або серверної частини, які реалізують окремі функції системи та доступні через браузер або API.

У даному проєкті реалізовані такі веб-компоненти:

- інтерфейс для завантаження PDF-документа;
- API-ендпоінти для отримання списку ОК, ВК, загальної інформації;
- фільтрація даних за ключовими полями (тип, кредит, семестр);
- інтерфейс перегляду JSON-виводу;
- веб-сторінка результату після обробки.

На базі Flask реалізовано серверну частину, де кожен маршрут (/OK, /BK, /main_info) відповідає за окрему функцію. Обробка запитів відбувається через HTTP-протокол з поверненням відповіді у форматі JSON. Для побудови

фронтенду можуть бути використані такі бібліотеки як Bootstrap або простий HTML+JS.

Важливі вимоги до веб-компонентів:

- швидкодія (відповідь API за 1–2 с);
- доступність (адаптація під різні екрани);
- безпека (перевірка типу файлу, захист від SQL-ін'єкцій);
- зрозумілий інтерфейс для кінцевого користувача.

РОЗДІЛ 3
КОНЦЕПТУАЛЬНА МОДЕЛЬ ІНФОРМАЦІЙНОЇ СИСТЕМИ РОЗРОБЛЕННЯ ТА ВЕРИФІКАЦІЇ ОСВІТНЬО-ПРОФЕСІЙНИХ ПРОГРАМ І НАВЧАЛЬНИХ ПЛАНІВ

3.1 Постановка та формалізація задачі

Основною метою проєкту є автоматизація процесу обробки освітньо-професійних програм і навчальних планів у ЗВО, зокрема їх оцифрування, перевірки та інтеграції в єдину інформаційну систему. Ця задача є актуальною у зв’язку з масштабністю документообігу, необхідністю забезпечення відповідності державним стандартам та створенням цифрового архіву програм.

Постановка задачі передбачає:

- автоматизований збір даних із PDF-документів ОПП і НП;
- визначення обов’язкових та вибіркових компонентів;
- структурування отриманих даних за фіксованими ключами;
- збереження інформації у форматі JSON та у базі даних;
- побудову інтерфейсу для взаємодії з користувачем і зовнішніми модулями;
- реалізацію API для отримання структурованих даних.

Формалізація задачі передбачає опис її у вигляді послідовності операцій, які виконує система:

- вхід: PDF-документ освітньої програми або навчального плану;
- обробка: виявлення та вилучення компонентів за ключовими словами;
- трансформація: поділ на ОК, ВК, ПР, формування структури JSON;
- вивід: доступ через веб-інтерфейс або API.

Кожна операція в системі має відповідати визначеним критеріям точності, стабільності, швидкодії та масштабованості. Крім того, задача передбачає забезпечення адаптивності до різних форматів документів, створених різними підрозділами університету.

3.2 Аналіз структури вхідних документів

Вхідними даними для інформаційної системи є PDF-документи, що містять текстовий або табличний опис освітньої програми. Структура таких документів умовно поділяється на блоки:

- загальні відомості (назва програми, спеціальність, освітній рівень);
- таблиця обов’язкових компонентів (ОК);
- таблиця вибіркових компонентів (ВК);
- опис програмних результатів навчання (ПР);
- додаткові елементи (перелік компетентностей, карта відповідностей, тощо).

Типовий запис у таблиці має вигляд:

ОК1 Алгоритми та структури даних 5 Іспит

Ключові особливості таких документів:

- відсутність структурованих тегів;
- текст і таблиці представлені як фрагменти, без чіткої розмітки;
- можливі варіації у назвах колонок, мовному оформленні, розділових символах;
- варіативність шаблонів залежно від факультету або року формування документа.

- Для ефективної обробки документів необхідно:
- ідентифікувати сторінки з корисним вмістом (наприклад, ті, що містять ОК або ВК);
- локалізувати таблиці;
- витягнути текст з кожного рядка, зберігаючи його позиційну прив’язку;
- розпізнати назву дисципліни, кредити, тип контролю тощо.

Аналіз вхідної структури дав змогу побудувати систему, здатну працювати з PDF-файлами без попереднього ручного редагування, а також адаптуватися до різних форматів подачі даних

3.3 Основні бази даних інформаційної системи

З метою ефективного зберігання та обробки структурованих даних, система використовує реляційну базу даних. Основною обраною технологією є SQLite у поєднанні з ORM-бібліотекою Peewee.

Основні таблиці бази даних:

- program — зберігає загальні дані про освітню програму;
- component — таблиця освітніх компонентів (ОК, ВК) з полями:
  - код компонента;
  - назва дисципліни;
  - кількість кредитів;
  - тип контролю;
  - тип (обов’язковий/вибірковий);
- user — базова таблиця користувачів;
- log — зберігає історію дій системи (дата, тип операції, результат).
- Усі таблиці мають зв’язки між собою за допомогою ключів:
- один програм має багато компонентів;
- лог містить записи, пов’язані з конкретними діями над програмами.

Завдяки ORM реалізується:

- зручна робота з базою через Python-код;
- автоматичне створення таблиць при запуску;
- перевірка цілісності зв’язків;
- фільтрація, сортування та об’єднання даних без використання SQL-запитів.

База даних є основою для збереження цифрових образів ОПП/НП, їх повторного використання, експорту, статистики та аналітики.

3.4 Компонента для оцифрування діючих ОПП і НП, їх верифікації та формування електронних образів

Цей компонент виконує функцію перетворення вже наявних освітніх програм у цифровий формат. Основні етапи його роботи:

- прийом документа у форматі PDF;
- визначення сторінок, що містять потрібну інформацію (ОК, ВК, ПР);
- витяг таблиць за допомогою бібліотек fitz та Aspose;
- парсинг змісту і побудова структури даних;
- збереження результату у форматі JSON;
- формування електронного образу програми в базі даних.

Вимоги до якості оцифрування:

- точність парсингу повинна перевищувати 90%;
- формат JSON має бути уніфікованим незалежно від структури документа;
- перед збереженням в базу даних виконується логічна перевірка.

Основні перевірки при верифікації документа:

- перевірка наявності ключових полів;
- відповідність кількості кредитів загальній сумі;
- уникнення дублікатів компонентів.

3.5 Компонента для розроблення нових ОПП і НП, їх верифікації та формування електронних образів

Компонент дозволяє створювати нові навчальні документи на основі вже збережених шаблонів або з нуля. Основні функції:

- формування нового JSON-документа з нуля;
- додавання обов’язкових і вибіркових компонентів вручну або через шаблон;
- редагування назв дисциплін, кількості кредитів, форм контролю;
- автоматична перевірка коректності структури;
- експорт у PDF, Word або JSON;
- інтеграція з інтерфейсом адміністратора.

Автоматичні перевірки, які виконує система під час створення нової ОПП або НП:

- перевірка дублікатів назв;
- перевірка відповідності кількості кредитів до меж;
- перевірка наявності ключових обов’язкових блоків.

Цей компонент дозволяє повністю оцифрувати процес проєктування освітньої програми, перевірити її правильність та одразу сформувати електронний образ, придатний для архівації або публікації.

3.6 Місце інформаційної підсистеми в структурі інтегрованої платформи забезпечення взаємодії з іншими підсистемами

Інформаційна підсистема з обробки ОПП та НП є частиною великої інтегрованої платформи цифрового супроводу освітнього процесу. Її основна роль — забезпечення:

- оцифрування, структурування та архівації програм;

- доступу до освітніх компонентів через API;
- автоматичної перевірки освітніх даних на відповідність вимогам.

У структурі платформи підсистема взаємодіє з:

- модулем індивідуальних навчальних планів (ІНП);
- модулем аналітики та статистики (побудова звітів для МОН);
- електронним журналом викладача (сумісність за кодами дисциплін);
- кабінетом адміністратора (додавання, редагування, контроль).
- Ключові аспекти технічної взаємодії між підсистемами:
- компонента передачі даних працює через REST API;
- всі сервіси використовують спільний формат JSON;
- реалізовано базову систему автентифікації для обмеження доступу.

Таким чином, інформаційна підсистема виконує функцію ядра для обробки освітніх даних і забезпечує структурну взаємодію між усіма компонентами освітньої цифрової інфраструктури ЗВО.

РОЗДІЛ 4
АРГУМЕНТАЦІЯ ТА ОБГРУНТУВАННЯ ВИБОРУ ІНСТРУМЕНТІВ РОЗРОБЛЕННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

4.1. Використання мови програмування Python, формату JSON, протоколу HTTP та мови запитів SQL

Python обрано як основну мову програмування для розробки інформаційної системи через його універсальність, простоту синтаксису та велику кількість готових бібліотек, що прискорюють створення прикладних рішень. Він дозволяє працювати з PDF-документами, базами даних, створювати web-сервери, реалізовувати API та працювати з форматами обміну даними.

Python добре підходить для обробки документів завдяки підтримці бібліотек, що дозволяють зчитувати вміст, аналізувати структуру сторінки,

витягати таблиці або текстові блоки. У цьому проєкті використовуються PyMuPDF (fitz) та Aspose.PDF, які забезпечують доступ до вмісту PDF без потреби в OCR.

Для зберігання результатів обробки було обрано формат JSON. Його структура ідеально підходить для представлення освітніх компонентів у вигляді масивів об’єктів, які можна легко серіалізувати, передавати або зберігати. JSON підтримується всіма сучасними web-системами і дозволяє забезпечити стандартизовану взаємодію між підсистемами. [19]

Комунікація між клієнтом і сервером здійснюється через HTTP-протокол, що дозволяє реалізувати REST API — стандарт для обміну даними між сервісами. Через API можна отримати дані про обов’язкові та вибіркові компоненти, загальну інформацію про програму або результати парсингу. [17]

Для зберігання даних застосовується SQL через вбудовану в Python базу даних SQLite. Це дозволяє створити просту, але стабільну модель даних, яка не потребує зовнішнього сервера і добре підходить для десктопних або локальних web-рішень.

У сукупності Python, JSON, HTTP і SQL формують базу технічної реалізації системи, яка дозволяє створити простий у підтримці, ефективний і сумісний інструмент для обробки навчальних програм.

4.2. Середовище розробки та інструменти

Розробка системи здійснювалась у середовищі **Visual Studio Code (VS Code)**. Це легке, багатофункціональне середовище з підтримкою багатьох мов програмування та великою кількістю плагінів. Воно дозволяє одночасно працювати з Python-кодом, файлами JSON, SQL-структурами та шаблонами HTML, що було особливо зручно при побудові веб-інтерфейсу.

Для роботи з віртуальними середовищами використовувався **venv**. Це дозволяє ізолювати залежності проєкту, щоб уникнути конфліктів між бібліотеками або версіями Python.

Основні плагіни та інструменти, які використовувались у VS Code:

- Python Extension (автодоповнення, інтегрований дебагер);
- SQLite Viewer (перегляд вмісту локальної бази даних);
- REST Client (тестування API без сторонніх інструментів);
- Git (контроль версій);
- Terminal (керування віртуальним середовищем та запуск скриптів).

Тестування web-компонентів проводилось локально через Flask Dev Server, який дозволяє швидко перевірити результат змін, не потребуючи додаткового налаштування сервера.

Також частина перевірки коду виконувалась у Jupyter Notebook — зручному середовищі для відлагодження фрагментів коду при роботі з PDF-документами або даними.

Обрана конфігурація середовища дозволила ефективно розробляти, тестувати та модифікувати проєкт без надмірного ускладнення налаштувань.

4.3. Огляд бібліотек fitz (PyMuPDF), Aspose.PDF for Python, Flask, peewee

Для реалізації проєкту були використані бібліотеки, які вирішують конкретні прикладні задачі, пов’язані з обробкою PDF, створенням API, побудовою веб-серверу та роботою з базами даних.

fitz (PyMuPDF) — це бібліотека для роботи з PDF-файлами. Вона дозволяє отримати доступ до сторінок документа, витягнути текст, координати блоків, здійснювати пошук по вмісту. Її основна перевага — швидкість та відсутність потреби в конвертації документа у зображення. [14]

Aspose.PDF for Python — потужна комерційна бібліотека з розширеними можливостями. У межах цього проєкту використовувалась для коректнішого витягу таблиць, коли fitz давав некоректні результати. Вона дозволяє працювати з табличною структурою, зберігати розмітку, що полегшує подальшу обробку. [13]

Flask — це легкий web-фреймворк на Python. Він дозволяє створити серверну частину системи, яка обробляє запити, передає дані в форматі JSON, запускає процедури обробки файлів, а також забезпечує доступ до результатів через браузер. [15]

Peewee — це ORM-бібліотека для взаємодії з базою даних SQLite. Вона забезпечує зв’язок між Python-класами і SQL-таблицями, дозволяє створювати таблиці, здійснювати вибірки, фільтрувати, додавати записи без написання SQL-запитів вручну. Завдяки цьому зменшується кількість помилок і спрощується підтримка коду. [16]

Ці інструменти взаємодіють між собою і дозволяють побудувати систему, яка виконує весь цикл: завантаження PDF → витяг інформації → формування структури → збереження → передача через API.

4.4. Мікросервісна архітектура

Система спроєктована з урахуванням принципів мікросервісної архітектури, хоча її реалізація обмежена локальним розгортанням. Основна ідея мікросервісів — поділ системи на окремі незалежні частини, кожна з яких виконує одну конкретну функцію.

У рамках даного проєкту мікросервісами умовно вважаються такі компоненти:

- парсер PDF-файлів (відповідає за вилучення тексту);
- сервіс обробки даних (перетворення у JSON, логіка розпізнавання);
- база даних з ORM (виконує зберігання і вибірки);
- API-сервер (передає дані за запитом);
- web-інтерфейс (відображення результатів користувачеві).

Кожен з модулів може бути протестований, змінений або розгорнутий окремо. Наприклад, замість Flask API можна реалізувати FastAPI без впливу на решту логіки.

Переваги такого підходу:

- простота оновлення окремих частин;
- ізоляція логіки і даних;
- легка масштабованість;
- спрощене тестування;
- зменшення впливу помилки одного модуля на всю систему.

Мікросервісний підхід особливо важливий для систем, які з часом мають бути розширені (наприклад, додавання нових форматів документів, інтеграція з LMS, додаткові API).

РОЗДІЛ 5
ОСОБЛИВОСТІ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ РОЗРОБЛЕННЯ ТА ВЕРИФІКАЦІЇ ОСВІТНЬО-ПРОФЕСІЙНИХ ПРОГРАМ І НАВЧАЛЬНИХ ПЛАНІВ

5.1 Аналіз структури вхідних документів

Першим етапом реалізації інформаційної системи стала робота з вхідними документами, які подаються у форматі PDF. Типовим прикладом таких документів є навчальні плани та освітньо-професійні програми (ОПП), що затверджуються у закладах вищої освіти на рівні факультетів або університетських методичних комісій. Ці документи містять чітко структуровану інформацію, що охоплює загальні відомості про програму, навчальні дисципліни, перелік обов’язкових та вибіркових компонентів, а також розподіл змісту по семестрах, обсяг кредитів та форми підсумкового контролю.

На початковому етапі було проведено візуальний та логічний аналіз структури документа, а також вивчено типове наповнення сторінок. Зокрема, було встановлено, що в основному текстовий вміст подається у табличній формі, часто без розмітки в структурованому вигляді, що створює складності при автоматичному обробленні. Було виявлено, що обов’язкові компоненти (ОК) зазвичай позначаються маркуванням на зразок "ОК1", "ОК2" тощо, тоді як вибіркові компоненти (ВК) — відповідно "ВК1", "ВК2" і так далі. Водночас у таблицях присутні такі важливі атрибути, як назва дисципліни, кількість кредитів та тип контролю (наприклад, "Іспит", "Залік").

Для прикладу, типовий запис виглядає наступним чином:

ОК1 Алгоритми та структури даних 5 Іспит

Таким чином, документ містить усю необхідну інформацію, однак вона не подається у вигляді, зручному для автоматизованої обробки, оскільки PDF-файли не мають чіткої семантичної структури, яка би дозволила просто ідентифікувати, наприклад, де починається таблиця, а де — просто текстовий фрагмент.

Послідовність дій програми для коректного оброблення документу:

- відкриває PDF-документ із навчальною програмою;
- виявляє сторінки, що містять необхідні компоненти (ОК, ВК);
- виділяє та витягує текстову інформацію з таблиць;
- перетворює її у структурований вигляд (словники, масиви);
- зберігає отриману інформацію у форматі JSON для подальшого використання;
- забезпечує доступ до цих даних через API (наприклад, запит на отримання списку ОК або ВК).

Ключовою проблемою, яка була виявлена при аналізі, стало те, що текстовий вміст PDF-документів не завжди зберігається як логічно пов’язані елементи — навіть рядки таблиці можуть бути представлені як окремі блоки тексту з різними координатами. У зв’язку з цим було прийнято рішення комбінувати кілька підходів: попередній відбір потрібних сторінок за ключовими

словами, а також подальший аналіз таблиць із використанням спеціалізованих бібліотек, таких як PyMuPDF (fitz) та Aspose.PDF.

Ці кроки дозволили сформувати основу для подальшої реалізації функціоналу обробки та структурування інформації, що є центральним завданням у створенні електронної системи супроводу навчальних планів. Зібрані дані використовуються не лише для формування відповіді на API-запити, але й можуть бути збережені у базі даних або передані до інших інформаційних підсистем (наприклад, до аналітичного модуля або внутрішнього архіву програм).

5.2 Парсинг PDF-документів

Використання бібліотеки fitz

На цьому етапі реалізації було здійснено попередню обробку PDF-документа з використанням бібліотеки fitz, яка є частиною пакету PyMuPDF. Основне завдання цього кроку — із повного документа виділити лише ті сторінки, які містять релевантну інформацію, тобто таблиці з обов’язковими (ОК) та вибірковими (ВК) компонентами освітньої програми. Це дозволяє суттєво зменшити обсяг аналізованих даних і сконцентрувати ресурси на змістовній частині.

Обробка документа виконувалась у кілька етапів (рис 3.1):

1. PDF-документ відкривається засобами fitz за допомогою методу fitz.open(). На цьому етапі відбувається ініціалізація об’єкта документа, який надає доступ до кожної сторінки та її вмісту.
2. Для кожної сторінки здійснюється зчитування тексту методом get_text(), після чого в текстовому вмісті виконується пошук за ключовими шаблонами за допомогою регулярних виразів. Зокрема, використовуються патерни на зразок ОК\d та ВК\d, які дозволяють виявити сторінки, на яких починається перелік компонентів навчального плану.

3. Сторінки, які не містять таких маркерів, вважаються другорядними (наприклад, титульна сторінка, пояснювальна записка, інструкції, додатки), тому підлягають видаленню. Для цього застосовується метод doc.delete_pages(start, end), який фізично вилучає зайві сторінки з документа перед подальшим аналізом.

```
doc = fitz.open("kn2022.pdf")
for i in range(doc.page_count):
    page_text = doc.load_page(i).get_text()
    if re.search(f"ОК\d", page_text) or re.search(f"ВК\d", page_text):
        pages.append(i)
doc.delete_pages(0, pages[0]-1)
doc.save('op.pdf')
```

Рисунок 5.1. Приклад пошуку потрібних сторінок

Такий підхід дозволяє залишити у фінальному файлі лише ті сторінки, де безпосередньо представлена таблична структура з освітніми компонентами. Це значно спрощує подальший парсинг, зменшує кількість помилкових спрацювань, пришвидшує обробку та підвищує точність виявлення навчальних дисциплін.

Окрім цього, збереження оновленого документа у новий файл (наприклад, op.pdf) дозволяє повторно використовувати очищений документ у наступних етапах — наприклад, при витягу таблиць за допомогою Aspose або при побудові загального профілю освітньої програми. Такий підхід також дозволяє візуально верифікувати якість попереднього очищення.

У процесі реалізації було враховано, що PDF-файли можуть мати різну структуру залежно від навчального закладу або року створення документа. Тому код побудований з урахуванням можливих варіацій, наприклад, наявності символів-роздільників, пропущених маркерів або змішаних форм представлення даних. У разі виявлення сторінки, де маркерів не знайдено, система її ігнорує, однак дозволяє масштабувати підхід — додавши нові шаблони у регулярні вирази.

Таким чином, використання бібліотеки fitz на цьому етапі дозволило ефективно сформувати з вихідного документа лише ту частину, яка підлягає

подальшому аналізу, що стало основою для автоматизованого структурування освітньої інформації.

Застосування Aspose.PDF для розпізнавання таблиць

Після попередньої обробки PDF-документа за допомогою бібліотеки fitz, наступним етапом стало розпізнавання табличних структур за допомогою бібліотеки Aspose.PDF. Цей етап є ключовим для коректного витягу інформації про освітні компоненти, оскільки саме у вигляді таблиць подається більшість важливих даних: назви дисциплін, кількість кредитів, форми контролю, тощо.

Бібліотека Aspose.PDF забезпечує високий рівень точності при роботі з таблицями у PDF-документах. В основі реалізації лежить клас TableAbsorber, який дозволяє здійснювати повноцінну ітерацію по структурі таблиці, зчитувати вміст кожної комірки та обробляти її як окрему одиницю даних. Це дає змогу автоматично витягнути значення з таблиць без потреби в ручному розборі позицій тексту, як це часто трапляється при використанні менш спеціалізованих бібліотек.

Процес обробки виконується поетапно. Спочатку реалізується ітерація по кожній сторінці документа, що була попередньо очищена від зайвих сторінок (наприклад, титульної або вступної частини). Далі на кожній сторінці викликається TableAbsorber, який визначає усі наявні таблиці. Для кожної знайденої таблиці виконується ітерація по її рядках. У кожному рядку — послідовне зчитування комірок. Вміст кожної комірки представлений у вигляді набору текстових фрагментів, які агрегуються у повноцінне текстове значення (рис 3.2).

Зібрані значення додаються до тимчасового списку, що відповідає одному рядку таблиці. Після цього виконується перевірка, чи перший елемент списку відповідає одному з ключових шаблонів, що позначають обов’язкові або вибіркові компоненти — відповідно, регулярні вирази ОК\d+ або ВК\d+. Якщо співпадіння є, то система інтерпретує рядок як валідний навчальний елемент та додає його у відповідний масив — OK_list або BK_list (рис 3.3).

```
for page in pdfDocument.pages:
    absorber = pdf.text.TableAbsorber()
    absorber.visit(page)
    for table in absorber.table_list:
        for row in table.row_list:
            l = []
            for cell in row.cell_list:
                tfc = cell.text_fragments
                txt = ""
                for fr in tfc:
                    if fr.text != ' ':
                        txt += fr.text
                l.append(txt)
```

Рисунок 5.2. Приклад витягу тексту з таблиці

```
if re.search(r'ОК\d', l[0]):
    OK_list.append({"id" : l[0], "name" : l[1], "credits" : l[2], "type" : l[3]})
elif re.search(r'ВК\d', l[0]):
    BK_list.append({"id" : l[0], "name" : l[1], "credits" : l[2], "type" : l[3]})
```

Рисунок 5.3. Приклад збереження даних

Кожен такий об’єкт зберігається у вигляді словника з чотирма основними ключами: ідентифікатор компонента (наприклад, ОК1), назва дисципліни, кількість кредитів (як правило, ціле число) та тип контролю (наприклад, залік, іспит, курсовий проєкт). Така структура дозволяє в подальшому конвертувати ці об’єкти у формат JSON або додавати до бази даних, що є зручною формою для роботи веб-додатків та інтеграції з іншими системами.

З технічної точки зору, значною перевагою Aspose є можливість зчитувати таблиці навіть у складних випадках — наприклад, коли одна комірка займає кілька рядків або містить внутрішні відступи. Крім того, бібліотека дозволяє працювати з українськомовними документами, не втрачаючи коректність відображення специфічних символів, що є критично важливим для обробки освітніх програм, затверджених у вітчизняних закладах вищої освіти.

Отримані на цьому етапі масиви структурованих даних формують ядро інформаційної моделі, яка може бути використана для подальшої аналітики, побудови інтерфейсів доступу, або інтеграції з іншими освітніми сервісами. При потребі ці дані можуть бути доповнені метаінформацією, збережені у базі даних або передані як JSON через API, що реалізований у наступному етапі.

5.3 Формування словнику даних

Наступним кроком після розпізнавання таблиць та витягу ключових елементів освітньої програми стало формування структури даних у вигляді словників, які можуть бути використані для обміну з іншими сервісами або модулями системи. Формат зберігання повинен бути уніфікованим, структурованим і передбачуваним, щоб забезпечити сумісність із зовнішніми інформаційними системами та API-інтерфейсами.

Для цього на основі попередньо зібраних даних з таблиць будуються словники з фіксованими ключами. Кожен елемент таблиці (рядок) перетворюється у словник, що описує окрему навчальну дисципліну. Визначено обов’язкові поля, які мають бути присутні в кожному словнику:

- id — унікальний ідентифікатор компонента, наприклад "ОК1" або "ВК2";
- name — назва дисципліни у форматі, в якому вона подана у PDF-документі;
- credits — кількість кредитів ECTS, що відповідають навчальному навантаженню;
- type — форма підсумкового контролю: іспит, залік, курсовий проєкт тощо.

Таким чином, один навчальний елемент представлено як словник, наприклад:

{"id": "ОК1", "name": "Алгоритми", "credits": "5", "type": "Іспит"}

У результаті створюються два основні масиви: один для обов’язкових компонентів (ОК), інший — для вибіркових компонентів (ВК). Кожен з цих масивів містить набір словників, які відповідають усім дисциплінам відповідної категорії.

Окрім компонентів ОК та ВК, до словникової структури додається також інформація про програмні результати навчання (ПР), яка витягується окремим модулем аналізу вмісту розділу "Очікувані результати навчання". Цей блок представляється у вигляді окремого підрозділу словника, що містить список результатів у текстовому вигляді.

Також до загальної структури включається загальна інформація про програму, яка зчитується окремо: спеціальність, мова викладання, кількість кредитів для здобуття ступеня, рівень акредитації, рівень освіти та інші поля. Уся ця інформація об’єднується у єдиний словник, що має наступну структуру:

- OK — список обов’язкових компонентів;
- ВК — список вибіркових компонентів;
- PR — перелік програмних результатів.

Сформований словник стає універсальним об’єктом обміну всередині системи. Його можна передавати до інших частин архітектури, зберігати у базі даних, використовувати для генерації динамічних HTML-сторінок або відповідей у форматі JSON в рамках реалізованого REST API.

Таке структурування даних забезпечує надійність та повторне використання — зміна формату в одному місці автоматично оновить його для всіх модулів. Окрім того, фіксована структура дозволяє проводити валідацію, тобто перевірку наявності всіх обов’язкових ключів перед збереженням чи передачею.

5.4 Створення API для доступу до даних

Наступним етапом стало створення простого REST API, що дозволяє звертатися до компонентів через HTTP-запити. Для цього використано Flask. Реалізовано три основні маршрути:

– /ОК/ — повертає всі обов’язкові компоненти (рис 3.4);
– /ВК/ — повертає всі вибіркові компоненти (рис 3.5);
– /main_info/ — повертає загальну інформацію про програму (рис 3.6).

Рисунок 5.4. Приклад відповіді з обов’язковими компонентами

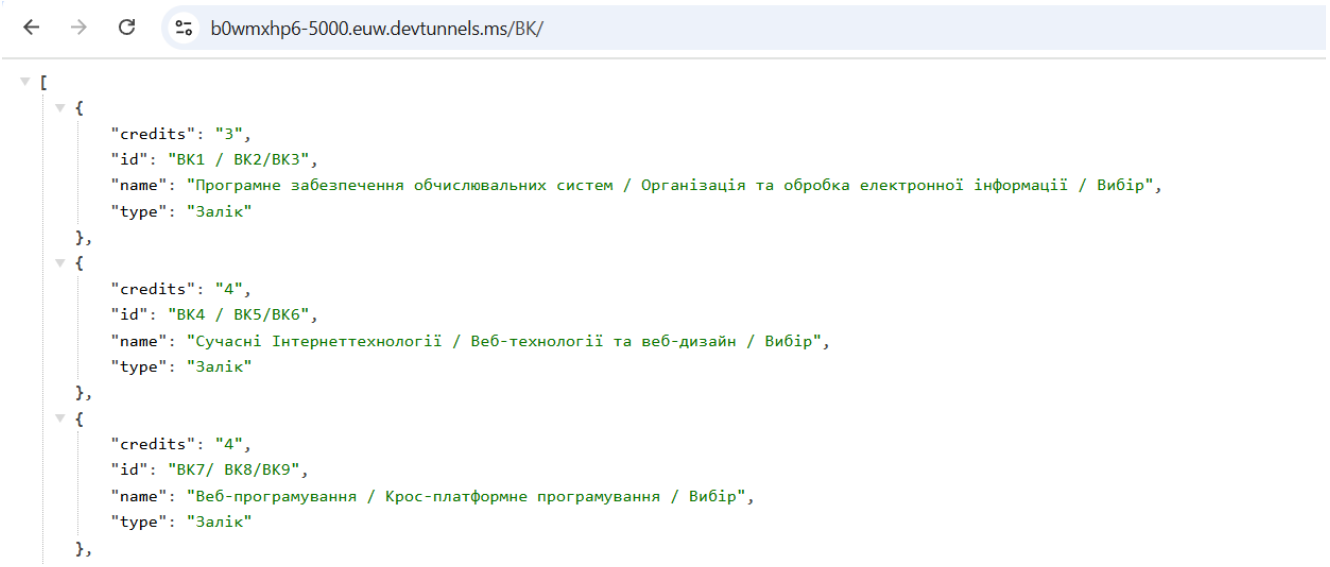

Рисунок 5.5. Приклад відповіді з вибірковими компонентами

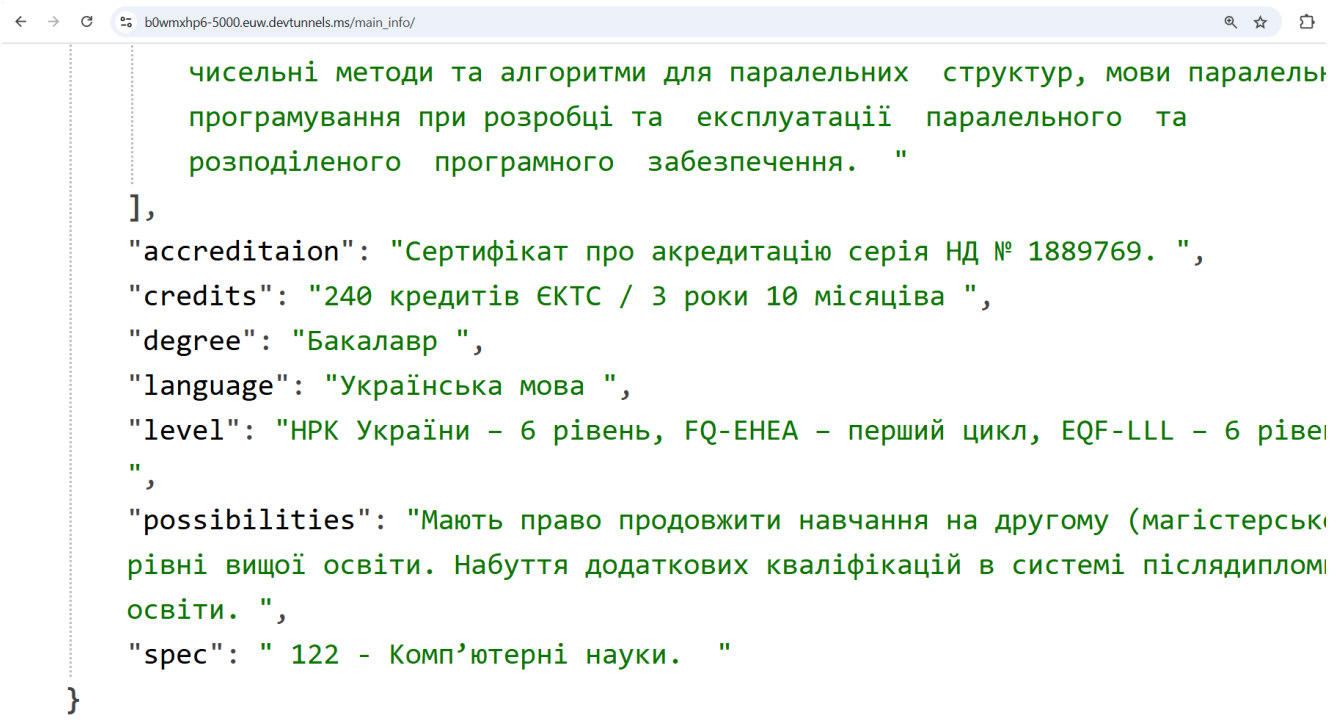

Рисунок 5.5. Приклад відповіді з основною інформацією

ВИСНОВОК

У результаті виконання кваліфікаційної роботи була спроєктована та реалізована інформаційна система підтримки електронного документообігу, призначена для автоматизації обробки освітньо-професійних програм і навчальних планів у закладах вищої освіти. Реалізоване рішення дозволяє виконувати парсинг PDF-документів, витягувати обов’язкові та вибіркові компоненти, формувати уніфіковану структуру даних, зберігати її у форматі JSON і надавати доступ через REST API.

Для побудови системи було обґрунтовано вибір програмного середовища та бібліотек: Python, Flask, PyMuPDF, Aspose.PDF, Peewee ORM і SQLite. Це дало змогу забезпечити ефективну реалізацію всіх етапів обробки — від завантаження документа до видачі структурованої відповіді.

Особливу увагу приділено формуванню словника даних, що забезпечує уніфіковане подання компонентів навчальних програм. Впровадження верифікації та попередньої фільтрації сторінок дозволило підвищити точність обробки документів та зменшити кількість помилок.

Практична значущість проєкту полягає у можливості інтеграції розробленої підсистеми до більшої інформаційної екосистеми ЗВО. Зокрема, вона може бути використана для формування індивідуальних навчальних планів, автоматизації подання документів до акредитаційних органів, або побудови аналітичної звітності.

Отримані результати можуть бути адаптовані до різних форматів навчальних документів, що створює основу для масштабування системи на рівні факультетів або навіть міжуніверситетських платформ. Перспективами подальшого розвитку є створення інтерфейсу редагування ОПП, підтримка експорту у формат Word, та інтеграція з сервісами цифрового підпису.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Закон України "Про електронні документи та електронний документообіг", 2003.
2. Закон України "Про електронні довірчі послуги", 2017.
3. Закон України "Про інформацію".
4. Закон України "Про захист інформації в інформаційно-телекомунікаційних системах".
5. ISO/IEC 27001:2013 — Information Security Management.
6. ДСТУ 2732:2004 — Уніфікована система документації.
7. Наказ МОН №676 від 01.06.2013 року "Про затвердження інструкції з діловодства у ВНЗ".
8. Постанова КМУ №55 від 17.01.2018 "Про деякі питання документообігу в органах виконавчої влади".
9. Указ Президента України "Про Концепцію розвитку цифрової економіки", 2018.
10. Національна рамка кваліфікацій України.
11. Методичні рекомендації МОН щодо розроблення ОПП.
12. Стандарт вищої освіти за спеціальністю 121 «Інженерія програмного забезпечення».
13. Aspose.PDF for Python documentation. https://docs.aspose.com/pdf/python-net/
14. Документація бібліотеки PyMuPDF. https://pymupdf.readthedocs.io
15. Flask Documentation. https://flask.palletsprojects.com
16. Peewee ORM Documentation. https://docs.peewee-orm.com
17. REST API Design Guidelines, Microsoft. https://learn.microsoft.com/en-us/azure/architecture/best-practices/api-design
18. SQLite Documentation. https://sqlite.org/docs.html
19. JSON Specification — ECMA-404. https://www.ecma-international.org/publications-and-standards/standards/ecma-404/

20. DevOps practices overview — Atlassian. https://www.atlassian.com/devops