МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХЕРСОНСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
Факультет комп'ютерних наук, фізики та математики
Кафедра інформатики, програмної інженерії та економічної кібернетики

ПРОЄКТУВАННЯ ТА РОЗРОБЛЕННЯ СЕРВІСНОЇ АРХІТЕКТУРИ УПРАВЛІННЯ БІЗНЕС-ПРОЦЕСАМИ УНІВЕРСИТЕТУ. ГЕНЕРАЦІЯ PDF ДОКУМЕНТІВ ЗАСОБАМИ СИСТЕМИ

Кваліфікаційна робота (проєкт)
на здобуття ступеня вищої освіти «магістр»

Виконав: здобувач 2 курсу, 241М групи
Спеціальності 121 Інженерія програмного забезпечення
Освітньо-професійної (наукової) програми
«Інженерія програмного забезпечення»
другого (магістерського) рівня вищої освіти
Коломієць Олег Едуардович
Керівник: кандидат фізико-математичних наук,
доктор педагогічних наук, професор
Співаковський Олександр Володимирович
Рецензент: Senior Software Engineer (EPAM)
Кожевніков Дмитро Ігорович

Херсон – 2021

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ 4
ВСТУП 5
РОЗДІЛ 1. Аналіз використаних інструментів 7
1.1 Електронні документи 7
1.1.1 Document management system. 7
1.1.2 Формати електронних документів. 7
1.1.2.1 DOCX. 8
1.1.2.2 PDF. 8
1.1.2.3 ODT. 9
1.1.2.4 RTF. 9
1.1.2.5 XML. 10
1.1.2.6 HTML. 10
1.2 Інструменти розробки системи 11
1.2.1 Python. 12
1.2.2 Django. 12
1.2.3 PostgreSQL. 13
1.2.4 Javascript. 13
1.2.5 React. 13
1.3 Аналіз інструментів розробки мікросервісу 14
1.3.1 Інструменти формування електронних документів. 14
1.3.1.1 React. 14
1.3.1.2 React-pdf/renderer. 15
1.3.1.3 jsPDF. 15
1.3.1.4 PyPDF. 15
1.3.1.5 xhtml2pdf. 16
1.3.1.6 PDFKit. 16
1.3.1.7 pyppeteer 16
1.3.1.8 WeasyPrint. 17

1.3.1.9 ReportLab. 17
1.3.1.10 Crystal Reports. 17
1.3.2 Інструменти побудови діаграм. 19
1.3.2.1 matplotlib. 19
1.3.2.2 Seaborn. 20
1.3.2.3 ggplot. 20
1.3.2.4 Bokeh. 20
1.3.2.5 Plotly. 21
1.3.2.6 pygal. 21
1.3.2.7 Leather. 22
РОЗДІЛ 2. Проєктування архітектури мікросервісу 24
2.1 Архітектура системи 24
2.2 Алгоритм взаємодії компонентів мікросервісу 27
2.3 Архітектура компоненту обробки запитів 28
РОЗДІЛ 3. Реалізація мікросервісу 31
3.1 Реалізація backend частини 31
3.1.1 API. 31
3.1.2 Chromium backend. 34
3.1.3 Менеджер запитів. 37
3.1.4 Шаблони. 38
3.2 Реалізація frontend частини 38
3.3 Документація 41
ВИСНОВКИ 42
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 43
ДОДАТКИ 46
Додаток А 46
Додаток Б 47
Додаток В 48
Додаток Г 54

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

1) **API** – прикладний програмний інтерфейс (Application Programming Interface);
2) **DMS** – система керування документами (Document Management System);
3) **XML** – розширювана мова розмітки (Extensible Markup Language);
4) **PDF** – портативний формат файлу (Portable Document Format);
5) **HTML** – мова розмітки гіпертексту (HyperText Markup Language);
6) **DBMS** – системи керування базами даних (Database Management Systems);
7) **URL** – уніфікований локатор ресурсів (Uniform Resource Locator).

ВСТУП

Уніфікація процесів управління системою, з метою оптимального керування нею, її оптимізації, є надзвичайно важливим кроком для поліпшення ефективності системи. Будь-яке підприємство прагне до скупчення всіх необхідних засобів керування, та їх стандартизації, оскільки таким чином зменшується залежність від інструментів третіх сторін, зменшується час простою працівників, зокрема через спрощення комунікації між ними. Крім того власна система управління дає змогу налагодити процеси так, як того потребує підприємство, а не так як передбачали розробники сторонніх інструментів. Також важливу роль в цьому питанні відіграє зменшення затрат на підтримку такої системи, оскільки придбання надійних інструментів витрачає великі кошти з бюджету підприємства.

Дана робота є продовженням робіт магістрантів 2019-2020 років навчання зі створення єдиної системи управління бізнес-процесами Херсонського державного університету. В даній роботі буде запропонована та розроблена архітектура сервісу для формування електронних документів як частини функціонала системи.

Документи можуть містити різну інформацію, однак їх головне призначення одне – спрощувати комунікацію: між працівниками, студентами та викладачами і т. д. В цій роботі увагу буде приділено створенню документів, що спрощують процес навчання студента і, як сервіс, доповнюють існуючий функціонал системи: генерація розкладу, залікової книжки, тощо.

Актуальність: полягає в необхідності уніфікації підходів до управління електронними ресурсами, прискоренні об’єднання різнорівневих ресурсів навчального закладу в єдиний портал та забезпеченні учасників освітнього процесу доступом до персоніфікованої інформації.

Об'єкт дослідження: системи управління університетами, системи керування документами.

Предмет дослідження: сервісна архітектура управління бізнес-процесами університету, мікросервіс формування електронних документів.

Мета: реалізація мікросервісу формування електронних документів та API для взаємодії з ним.

Завдання:

1) розглянути існуючі засоби для формування електронних документів;
2) обґрунтувати використані технології при проєктуванні мікросервісу;
3) на основі проведеного аналізу розробити вимоги щодо можливостей мікросервісу формування електронних документів;
4) відповідно до створених вимог розробити архітектуру мікросервісу;
5) реалізувати API мікросервісу;
6) розробити документацію до API.

Методи дослідження:

1) аналіз;
2) порівняння;
3) узагальнення;
4) моделювання.

Гіпотеза: очікується, що спроєктований продукт, визначений вимогами кваліфікаційної роботи, буде придатний до використання учасниками освітнього процесу в закладі вищої освіти.

Структура кваліфікаційної роботи: робота складається зі вступу, трьох розділів, висновків, списку використаних джерел та додатків.

РОЗДІЛ 1
АНАЛІЗ ВИКОРИСТАНИХ ІНСТРУМЕНТІВ

В даному розділі представлені загальні теоретичні відомості відповідно до теми кваліфікаційної роботи, інформація про засоби що були використані при розробці системи, а також аналіз існуючих засобів для втілення мікросервісу формування електронних документів.

1.1 Електронні документи

1.1.1 Document management system.

Document management system – це система, яка використовується для отримання, відстеження, керування і зберігання документів [38]. У разі керування цифровими документами такі системи засновані на комп'ютерних програмах. Цей термін частково перегукується з концепціями систем керування вмістом. Він часто розглядається як компонент систем керування корпоративним вмістом і пов'язаний з керуванням цифровими активами, візуалізацією документів, системами робочих процесів і системами керування записами.

Хоча, на даному етапі, мікросервіс не є повноцінною DMS, деякі її властивості йому присутні: створення, зберігання, завантаження, контроль версій файлів, і забезпечуються переліком прийнятих програмних рішень, і використаних програмних засобів, при розробці системи.

1.1.2 Формати електронних документів.

Існує багато форматів електронних документів, нижче будуть приведені та проаналізовані ті з них, використання яких є доцільним згідно до поставленої мети кваліфікаційної роботи, а саме:

1) DOCX;

2) PDF;

3) ODT;

4) RTF;

5) XML,

6) HTML.

1.1.2.1 DOCX.

Office Open XML – один з найпопулярніших і поширених форматів документів – це формат, що спочатку був властивий лише Microsoft Word [24].

Файл DOCX може містити текст з розширеним форматуванням, а також таблиці, зображення та інші об'єкти. Це робить його надійним форматом файлу для різних цілей, що вимагають гнучкого способу форматування документу. На його основі можуть бути створені накладні, контракти, звіти, та інші типи документів, що містять графіки і таблиці.

Файли цього формату можуть бути відредаговані автором та одержувачем.

Багато програм здатні відкривати файли DOCX, в залежності від їх версії, однак в деяких програмах може змінитися форматування або зображення і об’єкти можуть відображатися по-іншому.

1.1.2.2 PDF.

Portable Document Format – це формат файлу для відображення документів, включаючи форматування тексту та зображення, незалежно від програмного забезпечення, обладнання та операційних систем [35]. На основі мови PostScript кожен файл PDF інкапсулює повний опис документу з фіксованим макетом, включаючи текст, шрифти, векторну графіку, растрові зображення і іншу інформацію, необхідну для відображення. Іноді файли складаються лише з зображень і

відсканованих сторінок замість основного текстового файлу, також можливі їх суміші.

Файли даного формату можуть бути редаговані лише у спеціалізованих середовищах, що має як переваги так і недоліки.

PDF є найпоширенішим форматом документу у Web, що не дивно, оскільки його незалежність відносно платформи і можливість конвертації в інші формати, робить його ідеальним для обміну інформацією та друку

1.1.2.3 ODT.

Файли OpenDocument Text схожі на файли DOCX. Проте, хоча останні є властивими для Microsoft Word, файли ODT, його формат, поширюється за відкритою ліцензією і, таким чином, є властивим для багатьох різних програм обробки тексту з відкритим початковим кодом, таких як OpenOffice і Libre Office [20].

Документи з розширенням ODT можуть містити розширене форматування, об'єкти OLE, таблиці, графіки та зображення.

Файли ODT можуть редагуватися як автором, так і одержувачем.

Формат також підтримується іншими програмами такими як Microsoft Word, що робить його дуже універсальним, однак форматування і загальне відображення можуть бути зміщені або іншим чином відрізнятися від вихідного форматування при відкритті за допомогою програмного забезпечення, яке не створювало документ.

1.1.2.4 RTF.

RTF – це розширення файлу для документів у форматі Rich Text Format [37].

Даний формат також був розроблений Microsoft, але може бути створений з використанням майже будь-якої програми обробки тексту, такої як Microsoft NotePad або OpenOffice Writer.

Файли RTF можуть містити базове форматування тексту та не можуть містити зображення, відео та інші файли, тільки текст. Таким чином, файл можна використовувати для всіх потреб, які не включають медіа, а тільки форматований текст, наприклад: листи, контракти, інвойси.

Файли RTF також можуть бути редаговані.

1.1.2.5 XML.

Extensible Markup Language – це мова розмітки, яка визначає набір правил для кодування документів в форматі, який зручний для читання людиною і комп'ютером [16]. Документи XML є текстовим форматом з підтримкою Unicode для різних мов. Формати на основі XML стали стандартом для багатьох офісних інструментів, включаючи Microsoft Office, OpenOffice і LibreOffice, Хоча формат є дуже гнучким, він і його розширення часто піддаються критиці за багатослівність і складність, що робить його менш привабливим для використання.

1.1.2.6 HTML.

HTML – це мова гіпертекстової розмітки, документи на основі якої можна відображати в веб-браузерах [19]. Веб-браузери при необхідності можуть зберігати веб-сторінки в форматі HTML.

Хоча HTML – це абревіатура, які найчастіше асоціюються з веб-сторінками, документи також можна зберігати в цьому форматі. Більшість текстових редакторів можуть зберігати форматований текст, що містить зображення, аудіо та інші об'єкти в форматі HTML. Їх можна відкривати в будь-якому веб-браузері, і вони обробляються як веб-сайти, з вбудованими медіа та вихідними посиланнями.

Вибираючи формат створення електронних документів, нами були зважені наступні критерії оцінювання:

- поширеність;
- можливість широкого форматування;
- незалежність від програмних засобів, платформи.

Деталі порівняння наведено в таблиці 1.

Таблиця 1

Порівняння форматів документів

\Критерій Формат\	Поширеність	Широке форматування	Незалежність
DOCX	поширений	+	-
PDF	поширений	+	+
ODT	поширений	+	-
RTF	непоширений	-	+
XML	непоширений	+	-
HTML	непоширений	+	+

В вищенаведеній таблиці, в стовпці “широке форматування” плюсом позначені ті формати в яких можлива вставка інших об’єктів, крім тексту, в стовпці незалежність плюсом позначені ті формати які не залежать від програмних та апаратних засобів.

Як результат аналізу, формат PDF був обраний нами, як формат створюваних електронних документів.

1.2 Інструменти розробки системи

Програмні засоби, що були попередньо обрані при розробці системи, напряму впливають на можливий спектр інструментів які можуть бути використані при розробці мікросервісу, тому є важливим зробити на них наголос.

1.2.1 Python.

Python – це популярна, потужна мова програмування, що має ефективні високорівневі структури даних і ефективний підхід до об'єктно-орієнтованого програмування [8]. Вона мультиплатформна, інтерпретована, з динамічною типізацією і простим синтаксисом, що робить її ідеальною для написання скриптів та швидкої розробки застосунків.

Сфери використання Python включають в себе:

- веб-розробка на стороні сервера;
- написання системних скриптів;
- проведення математичних обчислень.

У випадку веб-розробки ключові особливості Python разом з її потужними сторонніми бібліотеками, які дозволяють нам встановлювати зв'язки з іншими застосунками, такими як система керування базами даних і працювати з великим обсягом даних, робить її ідеальною для швидкого прототипування, розробки і тестування складних систем.

1.2.2 Django.

Django – це безкоштовний веб-фреймворк з відкритим початковим кодом на основі Python, який слідує архітектурному шаблону “модель – шаблон – вигляд” [39]. Основне його завдання – спростити створення складних веб-сайтів на основі баз даних.

Фреймворк підкреслює можливість повторного використання і можливості підключення компонентів, низькорівневе з’єднання і швидкий темп розробки. Django також надає додатковий адміністративний інтерфейс створення, зчитування, оновлення та видалення, який генерується динамічно на основі аналізу моделей і налаштовується за допомогою адміністративних моделей.

Django бере на себе більшу частину турбот веб-розробки, дозволяючи нам сфокусуватись на розробці застосунку.

1.2.3 PostgreSQL.

PostgreSQL – це потужна система управління об'єктно-реляційними базами даних з відкритим початковим кодом, яка використовує і розширює мову SQL в поєднанні з великою кількістю функцій, які безпечно зберігають і масштабують найскладніші робочі навантаження [5].

PostgreSQL заслужила міцну репутацію завдяки своїй перевіреній архітектурі, надійності, цілісності даних, великому набору функцій, розширюваності, вона працює в усіх основних операційних системах.

Django підтримує кілька систем керування базами даних, історично Postgres заручилася найбільшою підтримкою як найбільш просунута і швидкозростаюча система, тому була обрана в ролі основної DBMS.

1.2.4 Javascript.

JavaScript – це інтерпретована або оперативно-скомпільована мова програмування [1]. Хоча вона найбільш відома як скриптова мова для веб-сторінок, вона також використовується в багатьох середовищах, не пов'язаних з браузером, наприклад Node.js. JavaScript заснована на прототипах, багатопарадигмальна, однопоточна, динамічна мова, що підтримує об'єктно-орієнтований, імперативний і декларативний підхід. Стандартом для JavaScript є специфікація мови ECMAScript.

1.2.5 React.

React – це бібліотека JavaScript для створення користувацьких інтерфейсів [9]. React спрощує створення інтерактивних інтерфейсів за рахунок розробки простих виглядів, для кожного компоненту в застосунку. Потім React ефективно оновлює і відображає тільки потрібні компоненти при зміні їх стану.

Інкапсульовані компоненти, які керують своїм станом, можна скомпонувати для створення складних інтерфейсів.

Оскільки логіка компонентів написана на JavaScript, а не на шаблонах, можливо легко передавати дані через застосунок і зберігати стан поза об'єктною моделлю документа.

1.3 Аналіз інструментів розробки мікросервісу

1.3.1 Інструменти формування електронних документів.

В процесі аналізу предметної області, нами було виділено перелік існуючих програмних засобів для створення електронних документів, а саме:

1) React;
2) React-pdf/renderer;
3) jsPDF;
4) PyPDF;
5) xhtml2pdf;
6) PDFKit (Python);
7) pyppeteer;
8) WeasyPrint;
9) ReportLab;
10) Crystal Reports.

Детальніший аналіз надано нижче.

1.3.1.1 React.

Хоча React, як бібліотека для створення інтерфейсу, за замовчуванням не надає інструментів для створення PDF, використовуючи “components” та JavaScript XML розширення можливо швидко створювати шаблони і маніпулюючи вікном браузера конвертувати їх у PDF формат.

Плюсами такого підходу є відсутність будь-яких окремих залежностей та простота у використанні.

1.3.1.2 React-pdf/renderer.

React-pdf експортує набір примітивів React, які дозволяють дуже легко перетворювати об’єкти в документ [10]. Вона також має API для їх стилізації з використанням властивостей CSS і макету Flexbox. React-pdf дозволяє створювати документ в різних середовищах: веб і сервер.

Набір примітивів є плюсом в тих випадках, коли необхідно створювати більш прості документи, у випадку коли потрібно створити документ зі складного шаблону, дана бібліотека може не надати бажаної гнучкості.

1.3.1.3 jsPDF.

jsPDF – це бібліотека JavaScript, провідне HTML5 рішення для створення PDF-файлів [2].

Хоча й має високу гнучкість за рахунок створення PDF з нуля, за допомогою JavaScript примітивів, в той же час це створює труднощі при створенні шаблонів, а для розширеного функціоналу необхідні окремі залежності.

1.3.1.4 PyPDF.

PyPDF – це бібліотека для роботи з PDF на чистому Python, здатна розділяти, об'єднувати, обрізати і перетворювати сторінки файлів PDF [7]. Вона також може додавати визначені користувачем дані, параметри перегляду і паролі, може витягувати текст і метадані з PDF-файлів, а також об'єднувати файли цілком. Будучи pure-Python, вона працює на будь-якій платформі без будь-яких залежностей від зовнішніх бібліотек. Тому це корисний інструмент для веб-сайтів, які керують PDF-файлами або маніпулюють ними.

Даний інструмент добре підходить для тих випадків коли необхідно працювати вже з існуючими документами, у нашому випадку вона не надає бажаної гнучкості для їх створення.

1.3.1.5 xhtml2pdf.

xhtml2pdf – це бібліотека Python, конвертер HTML в PDF з використанням ReportLab Toolkit, html5lib і PyPDF2 [40]. Вона підтримує HTML5, CSS 2.1 і частину CSS 3. Вона повністю написана на чистому Python, тому не залежить від платформи.

Основна перевага цього інструменту полягає в тому, що користувач з такими навичками роботи в Інтернеті, як HTML і CSS, може дуже швидко створювати шаблони PDF, не вивчаючи нові технології, однак суттєвим мінусом є відсутність повної підтримки CSS 3.

1.3.1.6 PDFKit.

PDFKit – це оболонка Python для утиліти wkhtmltopdf для перетворення HTML в PDF за допомогою рушія Webkit [30]. Це адаптована версія бібліотеки Ruby PDFKit.

Хоча й проста у використанні, є труднощі при роботі із локальними файлами, через застарілу версію C бібліотеки та рушія.

1.3.1.7 pyppeteer.

pyppeteer – це Python порт JavaScript бібліотеки puppeteer автоматизації “headless” Chrome/Chromium браузера [27]. Використовуючи інструменти вікна браузера, можна легко створювати PDF документи на основі шаблонів.

Суттєвим плюсом такого підходу є надійність роботи та повна підтримка останніх стандартів веб-сторінок, з мінусів – необхідність завантаження браузера та збільшення навантаження на систему.

1.3.1.8 WeasyPrint.

WeasyPrint – це розумне рішення, що допомагає веб-розробникам створювати документи PDF з HTML шаблонів [34].

Суттєвий мінус даного інструменту полягає у складності роботи на Windows системах, та наявності багатьох залежностей.

1.3.1.9 ReportLab.

ReportLab – це інструмент для створення PDF з використанням різноманітних технологій, включаючи JSON та XML [31].

Може бути використаний через URL посилання або як Python бібліотека. Використовує власну мову розмітки Report Markup Language для створення шаблонів. В обох випадках головним мінусом є велика вартість інструменту.

1.3.1.10 Crystal Reports.

Crystal Reports – це сукупність програмних рішень, що допомагають створювати PDF звіти на основі інформації з баз даних [32].

Головним мінусом є велика вартість інструментів, а також складність в обслуговуванні.

На основі проведеного аналізу нами було визначено перелік критеріїв для порівняння вищезазначених засобів:

- ціна;
- сторона роботи;
- мова розмітки;
- залежності;
- ефективність;
- підтримка.

В таблиці 2 наведено порівняння за визначеними критеріями.

Таблиця 2

Порівняння інструментів формування

\Критерій Інструмент\	Ціна	Сторона	Мова розмітки	Залеж-ності	Ефект-ивність	Під-тримка
React	-	front	JSX, HTML	-	medium	+
React-pdf	-	front	JSX	+	fast	+
jsPDF	-	front	JS	+	fast	+
PyPDF	-	back	Python	-	fast	-
xhtml2pdf	-	back	HTML	+	medium	-
PDFKit	-	back	HTML	+	fast	-
pyppeteer	-	back	HTML	-	medium	+
WeasyPrint	-	back	HTML	+	medium	+
ReportLab	+	back	RML	+	medium	+
Crystal Reports	+	back	-	+	medium	+

В вищенаведеній таблиці, в стовпці “Ціна” плюсом позначені ті інструменти, що є платними, в стовпці “Залежності” плюсом позначені ті інструменти які потребують встановлення сторонніх залежностей, в стовпці “Підтримка” плюсом позначені ті інструменти Git репозиторії яких є активними в публікації оновлень.

Головна увага приділялась наступним критеріям: ціна, мова розмітки, залежності.

У результаті порівняння нами було обрано два інструменти: React – для фронтенду, та pyppeteer – для бекенду. Обидва інструменти є безкоштовними і простими в роботі.

Використовуючи два інструменти, можливо зменшити навантаження на систему, в тих випадках коли створення документу не потребує додаткових запитів до серверу, та посилити захист системи, працюючи з чутливою інформацією на стороні серверу.

1.3.2 Інструменти побудови діаграм.

Окрім інструментів для формування електронних документів, нами був обраній та проаналізований список інструментів для створення діаграм та графіків на бекенд частині застосунку, для тих випадків коли документ повинен містити візуалізацію оброблених даних. До списку увійшли наступні Python бібліотеки:

1) matplotlib;
2) Seaborn;
3) ggplot;
4) Bokeh;
5) Plotly;
6) pygal;
7) Leather.

1.3.2.1 matplotlib.

matplotlib – це одна з найстаріших бібліотек візуалізації даних Python [23]. Незважаючи на те, що їй більше 15 років, вона все ще залишається найбільш використовуваною бібліотекою для побудови графіків на Python. Вона була розроблений під впливом MATLAB – пропрієтарної мови програмування.

Оскільки matplotlib була першою бібліотекою візуалізації даних Python, багато інших бібліотеки побудовані на її основі або призначені для роботи в тандемі з нею під час аналізу. Деякі бібліотеки, такі як pandas і Seaborn, є оболонками над нею, вони дозволяють отримати доступ до ряду методів її пишучи меншу кількість коду.

Хоча matplotlib гарна для аналізу даних, вона не дуже корисна для швидкого і простого створення діаграм якості публікації. matplotlib є дуже потужною бібліотекою, але водночас досить складною в роботі. Крім того дана бібліотека довгий час піддається критиці за стандартні стилі, що є досить застарілими.

1.3.2.2 Seaborn.

Seaborn використовує можливості matplotlib для створення гарних діаграм в кілька рядків коду [11]. Ключова відмінність – це стилі і колірні палітри за замовчуванням, які покликані бути більш естетичними і сучасними

Оскільки Seaborn побудована поверх matplotlib, необхідно знати останню, щоб налаштувати параметри Seaborn.

1.3.2.3 ggplot.

Бібліотека ggplot заснована на ggplot2, системі побудови графіків R і концепціях з ”The Grammar of Graphics” [17]. ggplot працює інакше, ніж matplotlib: вона дозволяє накладати компоненти на “пласти” для створення кінцевого графіку. Наприклад можливо почати з осей, потім додати точки, потім лінію і т. д.

ggplot не призначена для створення сильно налаштованих графіків, вона жертвує складністю заради більш простого методу побудови графіків. ggplot тісно інтегрований з pandas, тому при використанні ggplot найкраще зберігати дані в структурах визначених в pandas.

Хоча “The Grammar of Graphics” хвалять як інтуїтивно зрозумілий метод побудови графіків, досвідченим користувачам matplotlib може знадобитися час, щоб пристосуватися до нового підходу.

1.3.2.4 Bokeh.

Як і ggplot, Bokeh заснована на ”The Grammar of Graphics”, але, на відміну від ggplot, розроблена на Python, а не перенесена з R [12]. Її

ключова перевага полягає в здатності створювати інтерактивні, готові до роботи в Web графіки, які можна легко передавати як JSON-об’єкти або HTML-документи. Bokeh також підтримує потокову передачу даних в реальному часі.

Bokeh надає три інтерфейсу з різними рівнями керування для різних типів користувачів. Найвищий рівень призначений для швидкого створення діаграм. Він включає методи для створення добре відомих діаграм, таких як стовпчикові діаграми, діаграми розмаху, гістограми. Середній рівень має ту ж специфіку, що і matplotlib, і дозволяє керувати основними будівельними блоками кожної діаграми (наприклад, точками на точковій діаграмі). Найнижчий рівень орієнтований на розробників та інженерів-програмістів. Він не має попередньо встановлених значень за замовчуванням і вимагає визначення кожного елементу діаграми.

1.3.2.5 Plotly.

Plotly відома як онлайн-платформа для візуалізації даних, але також існує бібліотека візуалізації на Python [4]. Як і Bokeh, сильною стороною Plotly є створення інтерактивних графіків, але вона пропонує деякі діаграми, яких не має в більшості бібліотек, такі як контурні графіки, дендрограми і 3D-діаграми.

1.3.2.6 pygal.

Подібно Bokeh і Plotly, pygal пропонує інтерактивні графіки, які можна вбудувати в веб-браузер [26]. Її головна відмінність – можливість виводити діаграми в форматі SVG, такий підхід можливий при невеликих об’ємах даних, що дозволяє масштабувати діаграми без втрати якості зображення. В pygal кожен тип діаграми упакований в метод та вбудовані стилі приємні на вигляд, тому легко створити гарну діаграму за допомогою кількох рядків коду.

1.3.2.7 Leather.

Leather – це бібліотека візуалізації з допомогою якої можливо дуже швидко та просто створювати графіки, жертвуючи при цьому якістю [3]. Вона призначена для роботи з усіма типами даних і створює діаграми у формати SVG. Оскільки ця бібліотека відносно нова, частина документації ще недопрацьована.

Кожна з приведених бібліотек має свої переваги та недоліки. В таблиці 3 наведено порівняння за наступними критеріями:

- гнучкість;
- формати збереження діаграм;
- простота використання;
- залежності;
- підтримка.

Таблиця 3

Порівняння бібліотек візуалізації

\Критерій Бібліотека\	Гнуч-кість	Формати збережен-ня діаграм	Простота використ-тання	Залежно-сті	Підтримка
matplotlib	велика	PNG, PDF, SVG	низька	+	+
Seaborn	велика	PNG, PDF, SVG	середня	+	-
ggplot	середня	PNG, PDF, SVG	середня	+	-

Bokeh	велика	HTML	висока	+	+
Plotly	велика	PNG, JPEG, WEBP, SVG, PDF, HTML	середня	+	+
pygal	середня	SVG	висока	-	-
Leather	мала	SVG	висока	+	-

В вищенаведеній таблиці, в стовпці “Залежності” плюсом позначені ті бібліотеки які для роботи потребують додаткові залежності, в стовпці “Підтримка” плюсом позначені ті бібліотеки Git репозиторії яких є активними в публікації оновлень.

В подальшому при виборі інструменту необхідно звернути найбільшу увагу на наступні критерії: формати збереження діаграм, простота використання.

На даному етапі нами були проаналізовані та вибрані необхідні інструменти для розробки мікросервісу формування електронних документів.

На основі проведеного аналізу предметної області та вирішивши завдання 1, 2 є можливим описати вимоги до мікросервісу формування електронних документів:

1) мікросервіс повинен доповнювати наступний перелік мікросервісів:

- “Відділ кадрів”;
- “Розклад навчальних занять”;
- “Електронна залікова книжка”.

2) мікросервіс повинен бути побудований на основі інструментів визначених в процесі аналізу, а саме:

- React;

- pyppeteer.

3) архітектура мікросервісу повинна передбачати можливість збільшення кількості можливих запитів, а тому повинна бути ефективною в їх збереженні та обробці.

РОЗДІЛ 2
ПРОЄКТУВАННЯ АРХІТЕКТУРИ МІКРОСЕРВІСУ

При аналізі архітектури системи було визначено множину мікросервісів та їх компонентів. Релевантними до теми даної кваліфікаційної роботи є наступні з них: “Відділ кадрів”, “Розклад навчальних занять”, “Електронна залікова книжка”.

Оскільки функціонал мікросервісу формування електронних документів базується на роботі з типами даних, що були спроєктовані при розробці вищезазначених мікросервісів, нижче буде надано стислий опис основних моделей бази даних.

2.1 Архітектура системи

При розробці мікросервісу були використані наступні моделі БД:

- Person – особа, основа для системи автентифікації, спрощено має вигляд:

```
1  class Person()
2      user = models.OneToOneField()
3      surname = models.CharField()
4      name = models.CharField()
5      patronymic = models.CharField()
6      sex = models.PositiveIntegerField()
7      image = models.ImageField()
```

Модель необхідна для можливості збереження файлів створених документів, оскільки необхідно вказувати власника файлу;

• StudentGroup – група студентів, спрощено має вигляд:

```
1  class StudentGroup():
2      name = models.CharField()
3      parent = models.ForeignKey()
4      students = models.ManyToManyField()
5      is_archived = models.BooleanField()
6      is_subgroup = models.BooleanField()
7      education_start = models.DateField()
8      education_end = models.DateField()
9      educational_programme = models.ForeignKey()
10     faculty = models.ForeignKey()
11     grade = models.PositiveIntegerField()
12     course = models.PositiveIntegerField()
13     education_form = models.PositiveIntegerField()
```

Дана модель необхідна для формування розкладу факультету, так як містить інформацію про групи;

• Schedule – розклад навчальних занять, спрощено має вигляд:

```
1  class Schedule():
2      name = models.CharField()
3      description = models.TextField()
4      is_draft = models.BooleanField()
5      is_published = models.BooleanField()
6      start_date = models.DateField()
7      end_date = models.DateField()
```

Модель розкладу пов’язана з множиною навчальних занять які вона об’єднує;

• Lesson – навчальне заняття, спрощено має вигляд:

```
1  class Lesson():
2      schedule = models.ForeignKey()
```

```
3     type = models.IntegerField()
4     subject = models.ForeignKey()
5     teacher = models.ForeignKey()
6     classroom = models.ForeignKey()
7     description = models.CharField()
8     date = models.DateField()
9     time_begin = models.TimeField()
10    duration = models.DurationField()
11    link = models.URLField()
12    online_id = models.CharField()
13    online_code = models.CharField()
14    participants = models.ManyToManyField()
```

Множина навчальних занять становить розклад;

- Recordbook – залікова книжка студента, спрощено має вигляд:

```
1  class Recordbook():
2      id_record_book = models.CharField()
3      student = models.ForeignKey()
4      speciality = models.ForeignKey()
5      entry_date = models.DateField()
```

Дана модель необхідна для створення документу що містить інформацію щодо залікової книжки студента, включаючи зроблені в ній записи.

Крім того при розробці мікросервісу було спроєктовано клас-модель БД, що необхідна для збереження документів створених користувачами, спрощено вона має наступний вигляд:

```
1  class Report():
2      creator = models.ForeignKey()
3      title = models.CharField()
4      description = models.CharField()
5      file = models.FileField()
```

Тут creator – власник, створювач файлу, title – заголовок документу, description – опис документу, file – локація файлу в системі, місце його зберігання.

Також був створений “серіалайзер” на основі отриманої моделі, з використанням інструментів Django REST, що необхідний для зручної передачі даних між сторонами веб-застосунку.

2.2 Алгоритм взаємодії компонентів мікросервісу

Беручи до уваги обрані інструменти для розробки мікросервісу був розроблений наступний алгоритм створення електронних документів:

1) на основі отриманого запиту вибрати дані з БД;
2) обробити отримані дані;
3) використовуючи розроблений шаблон сформувати HTML документ на основі оброблених даних;
4) отриманий документ відкрити у вікні браузера;
5) використовуючи функціонал браузера зробити конвертацію в PDF.

На рисунку 2.1 наведена діаграма послідовності розробленого алгоритму.

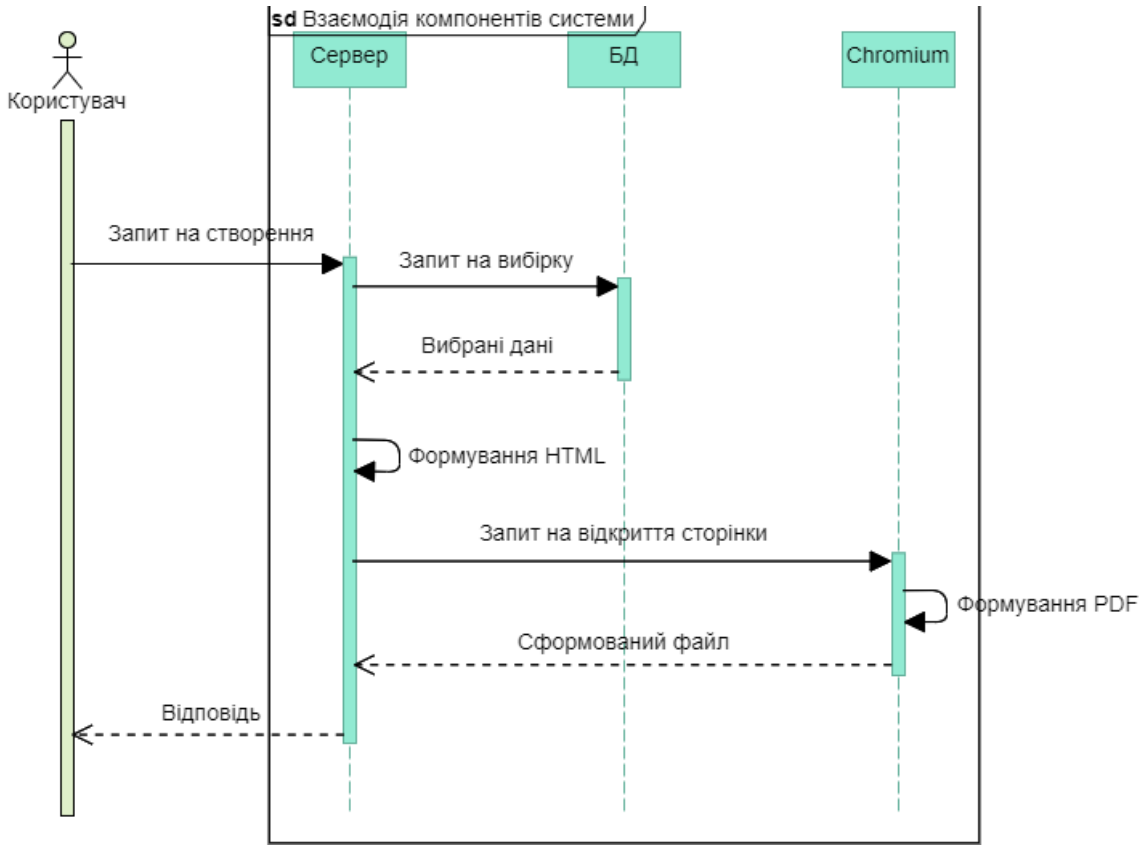

Рисунок 2.1 – Алгоритм взаємодії

Хоча API для взаємодії з програмними засобами різниться залежно від сторони, на якій створюється документ, загалом алгоритм залишається незмінним.

2.3 Архітектура компоненту обробки запитів

Під час проєктування мікросервісу постало питання щодо того, яким чином організувати збереження переліку можливих запитів на сервер, до бази даних та функцій обробки даних. Зазвичай API для взаємодії з системою представляють у вигляді окремих функцій або класів, що об’єднують споріднені функції, приклад такої архітектури зображено на рисунку 2.2. Причому кожній функції у відповідність стоїть URL адреса. Проте такий підхід може швидко стати неефективним при зростанні кількості можливих запитів, оскільки їх імплементації будуть розкидані по різним частинам системи, що суттєво погіршить можливість підтримки такої системи, “забруднить” її API.

```python
class ContactViewSet(viewsets.ModelViewSet):
    queryset = Contact.objects.all()
    serializer_class = ContactSerializer
    authentication_classes = [JSONWebTokenAuthentication]
    permission_classes = [IsAuthenticated]

    def get_queryset(self):
        user = self.request.user
        return Contact.objects.all().filter(person__user=user).order_by('-type')

    def create(self, request, **kwargs):
        data = request.data
        contact = Contact.objects.create(person=self.request.user.person, **data)
        return Response(data, status=status.HTTP_201_CREATED)
```

Рисунок 2.2 – Class-based API

Нами було запропоновано рішення на основі об’єктно-орієнтованої парадигми, використовуючи такі її властивості як інкапсуляція та наслідування.

Оскільки система заснована на ролях користувачів, було прийнято рішення розробки архітектури, що базується на ролях, приклад наведено на рисунку 2.3.

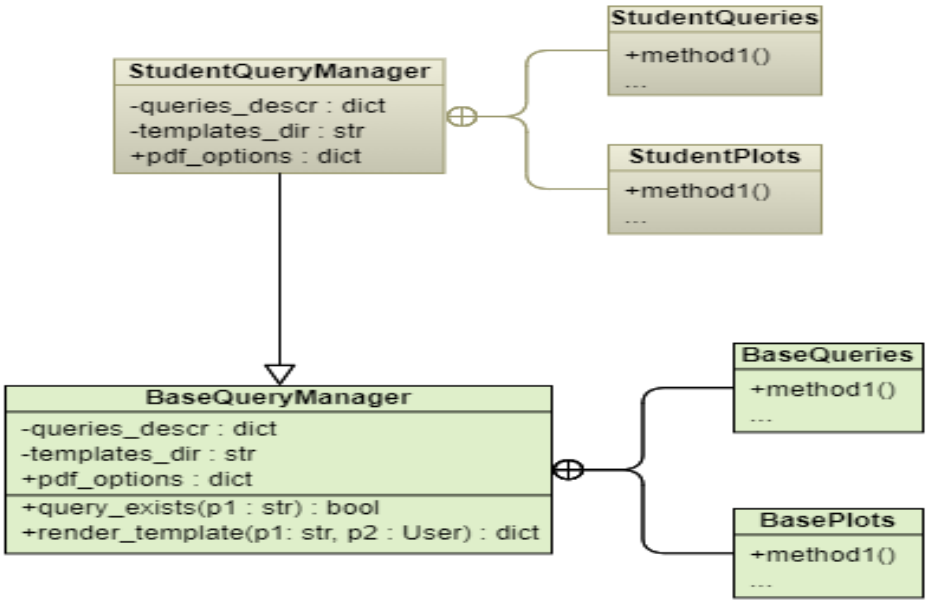

Рисунок 2.3 – Менеджери запитів

На основі кожної ролі формується окремий файл, що містить відповідний менеджер запитів.

Кожен менеджер включає в себе список можливих запитів, директорію шаблонів, налаштування документу та 2 внутрішні класи, що містять функції для роботи з базою даних та функції побудови діаграм. Список запитів зберігається у вигляді “словнику”, приклад зображений на рисунку 2.4.

Коли сервер отримує запит на формування електронного документу, він вибирає необхідний менеджер на основі ролі користувача і робить усі необхідні перевірки.

У результаті такого підходу ми отримали архітектуру яка може бути легко налаштована, та підтримка якої не займає багато часу.

```
queries_descr = {
    'contacts': {
        'query': BaseQueries.contacts,
        'template': templates_dir + 'contacts.html',
        'options': pdf_options,
    },
    'profile': {
        'query': BaseQueries.profile,
        'template': templates_dir + 'profile.html',
        'options': pdf_options,
    },
    'query3': {
        'query': BaseQueries.query3,
        'plot': BasePlots.query3,
        'template': templates_dir + 'query3.html',
        'options': pdf_options,
    },
```

Рисунок 2.4 – Словник запитів

РОЗДІЛ 3
РЕАЛІЗАЦІЯ МІКРОСЕРВІСУ

В даному розділі буде описано розроблений функціонал мікросервісу на прикладі таких документів як розклад навчальних занять та залікова книжка.

3.1 Реалізація backend частини

3.1.1 API.

Створення мікросервісу починається зі створення Django-застосунку. Для цього у командному рядку необхідно перейти в директорію з обраним проєктом Django і ввести наступну команду:

python manage.py startapp pdf apps/pdf

Ця команда створить застосунок Django зі стандартною архітектурою, приклад зображено на рисунку 3.1.

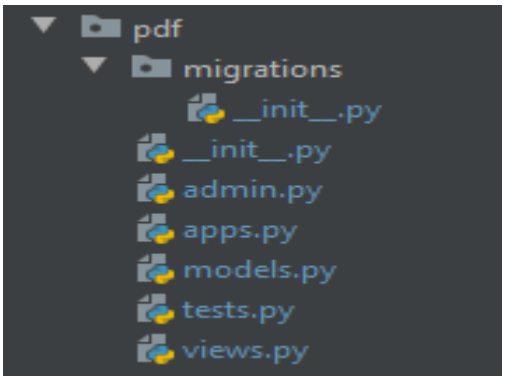

Рисунок 3.1 – Структура застосунку

Отриманий застосунок необхідно зареєструвати в списку застосунків, що знаходиться в налаштуваннях проєкту, в файлі *config/settings.py*. В список *INSTALLED_APPS* необхідно додати *'apps.pdf'*.

Наступним кроком є створення моделі БД, для цього у папці застосунку створюється папка *models*, з файлами *__init__.py* та *report.py*.

Розміщуючи моделі в окремій папці ми можемо покращити структуру проєкту. В *report.py* вписується модель БД, атрибути якої засновані на типах, що визначені Django, а в *__init__.py* отримана модель імпортується, що необхідно для компонування, таким чином полегшується звернення до отриманої моделі. Загальний опис моделі був наданий у другому розділі. Після опису моделі необхідно внести зміни в БД виконавши наступні команди у командному рядку:

python manage.py makemigrations

python manage.py migrate

При необхідності створюються серіалайзери, що розміщуються відповідно у власній папці *serializers*, вони необхідні для спрощення роботи з фреймворком Django REST і використовуються для конвертації моделей в JSON-об’єкти.

Наступним кроком є створення “виглядів” –до функцій, що будуть поєднані з відповідними URL та виконуватись при запиті на відповідну адресу. Вони містять основну логіку обробки запитів.

У нашому випадку структура проєкту передбачає створення виглядів у *api* застосунку, оскільки таким чином поліпшується структура проєкту. В *api* застосунку необхідно створити папку *pdf* з файлами *urls.py* та *views.py*. Вигляди, що будуть створені поміщуються у файл *views.py*, тому аналогічний файл у *pdf* застосунку можна видалити. Файл *urls.py* відповідає за маршрутизацію – компонування виглядів з URL адресами.

Як зазначалось раніше створення виглядів можливе двома способами: у вигляді функцій та у вигляді класів. Обидва шляхи мають свої переваги, але оскільки архітектурою мікросервісу передбачено створення єдиного API ендпоінту, для прикладу буде продемонстрований спосіб на основі функцій.

Оскільки для функціоналу мікросервісу необхідно працювати з бібліотекою pyppeteer, потрібно розробити низку функцій комунікації з

Chromium-рушієм. Наразі увага буде приділена архітектурі єдиного ендпоінту, усі допоміжні функції будуть описані далі.

У *views.py* створюємо наступну функцію:

```
1  @api_view(['GET'])
2  @permission_classes([IsAuthenticated])
3  @authentication_classes([JSONWebTokenAuthentication])
4      def get_default_pdf(request, kind):
       """
       View for all default queries (with predetermined results) for pdfs.
       """
5          if chromium_available():
6              manager = get_manager(request.query_param('role'))
7              if manager.kind_exists(kind):
8              html = manager.render_template(kind, request.user)
9              pdf = asyncrun(render_html_to_pdf(html,
   os.environ[settings.CHROMIUM_ENDPOINT],
   manager.queries_descr[kind]['options']))
10             if pdf is not None:
11                 return FileResponse(pdf, as_attachment='True',
   filename=f'{ kind }.pdf', status=status.HTTP_200_OK)
12             else:
13                 return Response({'message': 'Could not create pdf. Internal
   service error.'},
   status=status.HTTP_500_INTERNAL_SERVER_ERROR)
14         else:
15             return Response({'message': 'Could not create pdf. Service
   unavailable.'}, status=status.HTTP_503_SERVICE_UNAVAILABLE)
```

Декоратори *api_view*, *permission_classes* та *authentication_classes* визначені в Django REST і слугують для більш тонкої автентифікації користувача.

В тілі функції можна побачити такі функції як *chromium_available* та *render_html_to_pdf*, дані функції пов’язані з роботою з Chromium-рушієм.

Важливим моментом є вибір менеджеру запитів на основі ролі користувача в рядку №6: функція get_manager повертає необхідний менеджер, що пов’язаний з відповідною роллю, використовуючи словник, що пов’язує їх разом.

В тілі функції виконуються перевірки умов використовуючи *if-else* блоки, зокрема: чи працює рушій, чи існує запитаний ендпоінт, якщо не сталось помилок під час виконання функції та усі перевірки успішно пройдено, то користувачу повертається сформований файл.

3.1.2 Chromium backend.

Для роботи з Chromium-рушієм було реалізовано наступний перелік функцій:

1) run_chromium – для запуску рушія:

```
1  async def run_chromium():
       """
       Launch chromium and return it's endpoint.
       """
2      options = {
   'autoClose': False,
   'handleSIGINT': False,
   'handleSIGTERM': False,
   'handleSIGHUP': False,
   'logLevel': 'ERROR',
   'args': ['--disable-logging'],
       }
3      try:
4          try:
```

```
5            browser = await wait_for(launch(options), timeout=60)
6        except TimeoutError:
7            logger.error('Could not launch chromium. Timeout.')
8            return None
9    except BrowserError:
10       logger.error('Could not launch chromium. Internal
pyppeteer browser error occurred.')
11       return None
12   else:
13       return browser.wsEndpoint
```

Функція виконує запуск рушія і в успішному випадку повертає його ендпоінт;

2) kill_chromium – для припинення роботи рушія:

```
1    def kill_chromium(endpoint):
         """
         Kill chromium process.
         """
2        async def run():
3            try:
4                try:
5                    browser = await
wait_for(connect(browserWSEndpoint=endpoint,
logLevel='ERROR'), timeout=60)
6                    await browser.close()
7                except TimeoutError:
8                    logger.error('Could not kill chromium. Timeout.')
9            except BrowserError:
10               logger.error('Could not kill chromium. Internal
pyppeteer browser error occurred.')
```

11	*asyncrun(run())*

Функція завершує процес браузера за визначеним ендпоінтом;

3) chromium_available – для перевірки наявності бекенду:

1	*def chromium_available():*
	""" *Check if chromium is available.* *"""*
2	*if os.environ.get(settings.CHROMIUM_ENDPOINT, False):*
3	*return True*
4	*else:*
5	*return False*

Функція, перевіряє чи працює бекенд браузера;

4) render_html_to_pdf – для конвертації HTML в PDF:

Деталі імплементації зазначено в додатку А. Функція конвертує HTML в PDF керуючи вікном браузера.

Крім того була перевизначена функція запуску проєкту Django для можливості запуску рушія разом з проєктом, використовуючи командний рядок. Деталі імплементації зазначено в додатку Б.

При запуску рушія його ендпоінт *CHROMIUM_ENDPOINT* зберігається як змінна середовища та в файлі налаштувань *settings.py*. При першій роботі з pyppeteer Chromium автоматично завантажується на локальну систему, цим процесом можна керувати використовуючи змінні середовища: *PYPPETEER_HOME, PYPPETEER_DOWNLOAD_HOST ma PYPPETEER_CHROMIUM_REVISION.*

Всі функції для роботи з браузером було поміщено в файл *utils.py pdf* застосунку.

3.1.3 Менеджер запитів.

Розроблена архітектура менеджерів була описана в другому розділі, для наочності, нижче буде описана функція з *StudentQueryManager*, що формує дані для залікової книжки студента:

```
class StudentQueryManager:
    """
    Wrapper class to store dict of names of possible queries,
    with corresponding templates and methods.
    """
    class StudentQueries:
        @classmethod
        def recordbook(cls, user):
            """
            Returns all records of student's recordbook
            """
            r_book = Recordbook.objects.filter(student__person__user=user).first()
            records = r_book.recordbookrecord_set.all().annotate(semester=F('subject__semester'), hours_amount=F('subject__hours_amount'))
            studied_semesters = {(r.semester, SEMESTER_CHOICES[r.semester]) for r in records}
            studied_semesters = sorted(list(studied_semesters), key=itemgetter(1))
            return {'r_book': r_book, 'records': records, 'semesters': studied_semesters,}
```

3.1.4 Шаблони.

Шаблони, що використовуються для створення електронних документів, на бекенді зберігаються у папці *templates* як зображено на рисунку 3.2.

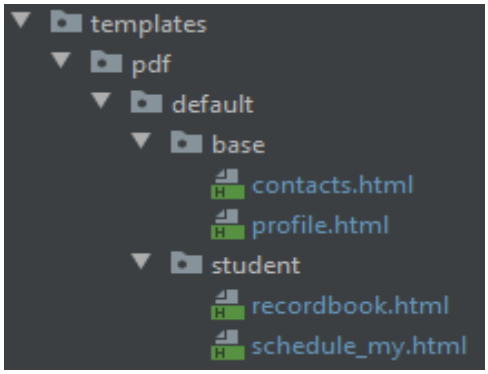

Рисунок 3.2 – Зберігання шаблонів

Для створення шаблонів на backend використовуються шаблонізатори, такі як DTE, Jinja та інші. Вони декларують набір примітивів за допомогою яких можливе динамічне створення сторінок. Для можливості роботи Django з шаблонами необхідно зробити налаштування об’єкту *TEMPLATES* в файлі *settings.py*.

3.2 Реалізація frontend частини

Як зазначалось раніше формування документів на стороні користувача приводить до зменшення кількості запитів до серверу, крім того таким чином можливо зменшити навантаження на сервер делегуючи процес формування документу фронтенд частині застосунку, в тих випадках коли його формування займає багато ресурсів. В даній кваліфікаційній роботі саме таким чином було обрано формувати розклад факультету.

Першим чином створюється React-компонент, він одночасно відіграє роль шаблонізатора та вигляду. В папці *components* застосунку створюємо папку *FacultySchedule* з наступними файлами: *index.js*,

FacultyScheduleCSS.js. *index.js* буде містити сам компонент та допоміжні функції, а *FacultyScheduleCSS.js* – стилі форматування шаблону.

Спрощено компонент має наступний вигляд:

```
1  const FacultySchedule = ({data}) => {
2      data processing…
3      render ( … )};
```

Як можна побачити головною відмінністю від backend функціоналу є інкапсуляція функцій обробки даних разом з шаблонами, що для них призначені. Деталі імплементації зазначені в додатку В.

Окрім компоненту було необхідним реалізувати наступний список функцій:

1) fetchFacultySchedule – запит до серверу для отримання даних розкладу:

```
1  const fetchFacultySchedule = async (schedule_name) => {
2      try {
3          const { data } = await Api.get(`api/${endpoint}`,
{'schedule_name': schedule_name});
4          return data;}
5      catch (error) {
6          openNotification('error', 'Сталася помилка', 'Помилка у
отриманні розкладу факультету');};};
```

Функція робить запит на отримання даних і виводить помилку у разі невдачі;

2) printData – для виведення інформації, для друку:

```
1  const printData = (data) => {
2      const w = window.open();
3      const closeWindowAfterPrintScript = `
       <script type="text/javascript">
           window.onafterprint = window.close;
```

```
</script>`
w.document.write(data + closeWindowAfterPrintScript);
w.print();};
```

Функція виводить необхідну інформацію у нове вікно і визиває діалогове вікно друку;

3) printSchedule – для інкапсуляції вищезазначеного компоненту та функцій:

```
const printSchedule = async (schedule_name) => {
    const schedule_data = await fetchFacultySchedule(schedule_name);
    printData(ReactDOMServer.renderToStaticMarkup(<FacultySchedule data={schedule_data}/>)); };
```

На рисунку 3.3 зображено приклад частини сторінки сформованого PDF документу.

Курс		6
Спеціальність		121 Інженерія програмного забезпечення
Освітня програма		Інженерія програмного забезпечення
Група		241М
Середа	1	ОСНОВИ BigData (ПР.) доц. Г.Кравцов Ауд. 501
	2	ОСНОВИ BigData (ПР.) доц. Г.Кравцов Ауд. 501
	3	ФОРМАЛЬНІ МЕТОДИ ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ (ПР.) викл. М.Полторацький Ауд. 512
	4	

Рисунок 3.3 – Фрагмент розкладу

3.3 Документація

Документація backend компонентів мікросервісу здійснюється автоматично оскільки система використовує бібліотеку sphinx, що створює документацію у форматі RST, на основі початкового коду, разом з коментарями. Для документації frontend частини використовуються коментарі де це є необхідним.

ВИСНОВКИ

У процесі виконання кваліфікаційної роботи були вирішені всі поставлені завдання, а саме:

1) розглянуті існуючі засоби для формування електронних документів;

2) обґрунтовано використані технології при проєктуванні мікросервісу;

3) на основі проведеного аналізу розроблено вимоги щодо можливостей мікросервісу формування електронних документів;

4) відповідно до створених вимог розроблено архітектуру мікросервіу;

5) реалізований API мікросервісу;

6) створена документація до API.

Досягнення поставленої мети і завдань було здійснено ретельним аналізом предметної області, з вивченням відповідних літературних джерел, аналізом існуючих архітектурних рішень та засобів формування документів як на backend так і frontend частинах застосунку, спираючись на досвід попередніх розробників системи.

Було обґрунтовано використання обраних технологій, та наведено порівняння між ними. Тому вважаємо, що обрані інструменти є найліпшими для вирішення поставленої мети.

Як результат виконання кваліфікаційної роботи був розроблений мікросервіс формування електронних документів, як частини функціоналу системи управління бізнес-процесами університету.

Даний сервіс відповідно до вимог доповнює існуючі мікросервіси системи:

- “Відділ кадрів”;
- “Розклад навчальних занять”;
- “Електронна залікова книжка”.

Розроблене програмне забезпечення повинно спрощувати комунікацію між учасниками освітнього процесу, та покращувати ефективність керівництва системою університету.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1) About JavaScript [Електронний ресурс]. – URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript

2) About jspdf [Електронний ресурс]. – URL: https://parall.ax/products/jspdf

3) About Leather [Електронний ресурс]. – URL: https://leather.readthedocs.io/en/0.3.3/about.html

4) About Plotly [Електронний ресурс]. – URL: https://plotly.com/

5) About PostgreSQL [Електронний ресурс]. – URL: https://www.postgresql.org/about/

6) About psycopg [Електронний ресурс]. – URL: https://www.psycopg.org/

7) About PyPDF4 [Електронний ресурс]. – URL: https://github.com/claird/PyPDF4

8) About Python [Електронний ресурс]. – URL: https://www.python.org/about/

9) About React [Електронний ресурс]. – URL: https://reactjs.org/

10) About React-pdf [Електронний ресурс]. – URL: https://react-pdf.org/

11) An introduction to seaborn [Електронний ресурс]. – URL: https://seaborn.pydata.org/introduction.html

12) Bokeh’s documentation [Електронний ресурс]. – URL: https://docs.bokeh.org/en/latest/

13) Developing with Docker [Електронний ресурс]. – URL: https://www.docker.com/why-docker

14) Django REST framework [Електронний ресурс]. – URL: https://www.django-rest-framework.org/

15) ECMAScript specification [Електронний ресурс]. – URL: https://www.ecma-international.org/publications-and-standards/standards/?order=last-change

16) Explaining XML [Електронний ресурс]. – URL: https://hbr.org/2000/07/explaining-xml

17) ggplot’s project description [Електронний ресурс]. – URL: https://pypi.org/project/ggplot/

18) How to Generate a PDF with JavaScript [Електронний ресурс]. – URL: https://pspdfkit.com/blog/2019/html-to-pdf-in-javascript/

19) HTML: HyperText Markup Language [Електронний ресурс]. – URL: https://developer.mozilla.org/en-US/docs/Web/HTML

20) Information technology — Open Document Format for Office Applications (OpenDocument) v1.2 — Part 1: OpenDocument Schema [Електронний ресурс]. – URL: https://www.iso.org/standard/66363.html

21) Introduction to Mastering Django [Електронний ресурс]. – URL: https://djangobook.com/mdj2-introduction/

22) Learn how to create beautiful and insightful charts with Python [Електронний ресурс]. – URL: https://towardsdatascience.com/plotting-with-python-c2561b8c0f1f

23) Matplotlib: Visualization with Python [Електронний ресурс]. – URL: https://matplotlib.org/stable/index.html

24) Open XML Formats and file name extensions [Електронний ресурс]. – URL: https://support.microsoft.com/en-us/office/open-xml-formats-and-file-name-extensions-5200d93c-3449-4380-8e11-31ef14555b18

25) PDF Report generator [Електронний ресурс]. – URL: https://www.pythonguis.com/examples/python-pdf-report-generator/

26) Pygal – Sexy python charting [Електронний ресурс]. – URL: http://www.pygal.org/en/stable/index.html

27) pyppeteer’s documentation [Електронний ресурс]. – URL: https://pyppeteer.github.io/pyppeteer/

28) Python Design Patterns: For Sleek And Fashionable Code [Електронний ресурс]. – URL: https://www.toptal.com/python/python-design-patterns

29) Python Notes for Professionals book [Електронний ресурс]. – URL: https://books.goalkicker.com/PythonBook/

30) Python PDFKit: HTML to PDF wrapper [Електронний ресурс]. – URL: https://github.com/JazzCore/python-pdfkit

31) ReportLab Open Source [Електронний ресурс]. – URL: https://www.reportlab.com/dev/opensource/

32) SAP Crystal solutions [Електронний ресурс]. – URL: https://www.crystalreports.com/faq/

33) Types of UML Diagrams [Електронний ресурс]. – URL: https://www.lucidchart.com/blog/types-of-UML-diagrams

34) WeasyPrint The Awesome Document Factory [Електронний ресурс]. – URL: https://doc.courtbouillon.org/weasyprint/stable/

35) What is a PDF? [Електронний ресурс]. – URL: https://pdf.abbyy.com/learning-center/what-is-pdf/

36) What is a Relational Database (RDBMS)? [Електронний ресурс]. – URL: https://www.oracle.com/database/what-is-a-relational-database/

37) What is a RTF file? [Електронний ресурс]. – URL: https://docs.fileformat.com/word-processing/rtf/

38) What is Document Management (DMS)? [Електронний ресурс]. – URL: https://www.aiim.org/what-is-document-imaging

39) Why Django? [Електронний ресурс]. – URL: https://www.djangoproject.com/start/overview/

40) xhtml2pdf’s documentation [Електронний ресурс]. – URL: https://xhtml2pdf.readthedocs.io/en/latest/

ДОДАТКИ

Додаток А

```
async def render_html_to_pdf(html, endpoint, options=None):
    """
    Renders HTML to PDF using Chromium.
    On first run downloads latest version of Chromium if it's not found on your
system.
    For details visit https://pyppeteer.github.io/pyppeteer/reference.html .
    """
    try:
        try:
            browser = await wait_for(connect(browserWSEndpoint=endpoint,
logLevel='ERROR'), timeout=5)
        except TimeoutError:
            logger.error('Could not connect to chromium. Timeout.')
            return None
        else:
            try:
                page = await browser.newPage()
                await page.setContent(html)
                pdf = BytesIO()
                pdf.write(await page.pdf(options))
                pdf.seek(0)
                await page.close()
            except PageError:
                logger.error('Internal pyppeteer page error occurred.')
                return None
            else:
                await browser.disconnect()
                return pdf
    except BrowserError:
        logger.error('Could not connect to chromium. Internal pyppeteer browser
error occurred.')
        return None
```

Додаток Б

```
class Command(RunServerCommand):
    def add_arguments(self, parser):
        super(Command, self).add_arguments(parser)
        parser.add_argument(
            '--with-chromium', action='store_true', dest='chromium',
            help='run server with chromium',
        )

    def run(self, **options):
        """
        Overrides runserver command to run&kill chromium.

        Note: The functions registered via atexit module are not called when
the program is killed by a signal
        not handled by Python, when a Python fatal internal error is detected,
or when os._exit() is called.
        """
        if os.environ.get('RUN_MAIN') != 'true':
            if options.get('chromium', False):
                endpoint = asyncrun(run_chromium())
                if endpoint is not None:
                    os.environ[settings.CHROMIUM_ENDPOINT] = endpoint
                    atexit.register(kill_chromium,
os.environ[settings.CHROMIUM_ENDPOINT])
            else:
                logger.warning('Starting server without chromium.')
        super(Command, self).run(**options)
```

Додаток В

```
const FacultySchedule = ({ data }) => {
  const lessonStartTimes = {};
  for (const time of BELL_SCHEDULE) {
    // convert array to dictionary-like object
    lessonStartTimes[time] = {};
  }
  const paperWidth = 247;
  const maxWidth = 70;
  const pageCapacity = 4; // max number of groups displayed per page
  // break down groups into different pages
  let courses = {};
  for (const group of data.groups) {
    if (!(group.course in courses)) {
      courses[group.course] = { 0: [], 1: [] };
    }
  }
  for (const group of data.groups) {
    if (!group.parent_id) {
      courses[group.course][0].length < pageCapacity
        ? courses[group.course][0].push(group)
        : courses[group.course][1].push(group);
    }
  }

  // calc lesson container width if number of groups < pageCapacity
  let lessonContainerWidth = Math.floor(paperWidth / pageCapacity);
  if (lessonContainerWidth > maxWidth) {
    lessonContainerWidth = maxWidth;
  }
  const subgroupLessonContainerWidth =
    (lessonContainerWidth / 2).toString() + 'mm';
  lessonContainerWidth = lessonContainerWidth.toString() + 'mm';

  // break down lessons into following hierarchy: course -> page -> day ->
group -> time -> lesson
  moment.locale('uk');
  let lessonsByCourseDateGroup = {};
  for (const course in courses) {
    lessonsByCourseDateGroup[course] = {};
    for (const page in courses[course]) {
      if (courses[course][page].length) {
        lessonsByCourseDateGroup[course][page] = {};
        for (const dayName of moment.weekdays().slice(1, 6)) {
          lessonsByCourseDateGroup[course][page][dayName] = {};
        }
      }
    }
  }
  for (const dayName of moment.weekdays().slice(1, 6)) {
    for (const group of data.groups) {
      if (!group.parent_id) {
        // skip groups which have parent
        Object.keys(lessonsByCourseDateGroup[group.course][0][dayName]).length
<
        pageCapacity
          ? (lessonsByCourseDateGroup[group.course][0][dayName][group.name] =
              JSON.parse(JSON.stringify(lessonStartTimes))) // *deep* cloning
          : (lessonsByCourseDateGroup[group.course][1][dayName][group.name] =
              JSON.parse(JSON.stringify(lessonStartTimes)));
      }
```

```
  }
}
for (const lesson of data.lessons) {
  for (const group of lesson.group) {
    let lessonGroupName = group;
    let groupCourse;
    let parentId;
    for (let i = 0, l = data.groups.length; i < l; i++) {
      if (data.groups[i].name == lessonGroupName) {
        groupCourse = data.groups[i].course;
        if (data.groups[i].parent_id) {
          parentId = data.groups[i].parent_id;
        }
        break;
      }
    }
    if (parentId) {
      // if group has parent write lesson into corresponding parent group
      for (let i = 0, l = data.groups.length; i < l; i++) {
        if (data.groups[i].id == parentId) {
          lessonGroupName = data.groups[i].name;
          parentId = null;
          break;
        }
      }
    }
    lessonGroupName in
    lessonsByCourseDateGroup[groupCourse][0][
      moment(lesson.date).format('dddd')
    ]
      ? (lessonsByCourseDateGroup[groupCourse][0][
          moment(lesson.date).format('dddd')
        ][lessonGroupName][
          moment(lesson.time_begin, 'HH:mm:ss').format('HH:mm')
        ][lesson.id] = lesson)
      : (lessonsByCourseDateGroup[groupCourse][1][
          moment(lesson.date).format('dddd')
        ][lessonGroupName][
          moment(lesson.time_begin, 'HH:mm:ss').format('HH:mm')
        ][lesson.id] = lesson);
  }
}
return (
  <body>
    <style>{stylesheet}</style>
    <div className="curtain">
      {Object.keys(courses).map(courseNumber => (
        <>
          {Object.keys(courses[courseNumber]).map(
            page =>
              !!courses[courseNumber][page].length && (
                <table className="course-tb">
                  <tr>
                    <td colSpan="2" className="b">
                      <div className="leg flx">Курс</div>
                    </td>
                    <td colSpan="100" className="b">
                      <div className="leg-course f-big flx">
                        {courseNumber}
                      </div>
                    </td>
                  </tr>
```

```
<tr>
  <td colSpan="2">
    <table className="tb-full">
      <tr>
        <td className="b">
          <div className="leg flx">Спеціальність</div>
        </td>
      </tr>
      <tr>
        <td className="b">
          <div className="leg flx">Освітня програма</div>
        </td>
      </tr>
      <tr>
        <td className="b">
          <div className="leg flx">Група</div>
        </td>
      </tr>
    </table>
  </td>
  {Object.keys(courses[courseNumber][page]).map(group => (
    <td>
      <table className="tb-full">
        <tr>
          <td className="b">
            <div
              className="leg-data flx"
              style={{ width: lessonContainerWidth }}
            >
              {
                courses[courseNumber][page][group]
                  .speciality
              }
            </div>
          </td>
        </tr>
        <tr>
          <td className="b">
            <div
              className="leg-data flx"
              style={{ width: lessonContainerWidth }}
            >
              {
                courses[courseNumber][page][group]
                  .educational_programme_name
              }
            </div>
          </td>
        </tr>
        <tr>
          <td className="b">
            <div
              className="leg-data flx"
              style={{ width: lessonContainerWidth }}
            >
              {courses[courseNumber][page][group].name}
            </div>
          </td>
        </tr>
      </table>
    </td>
  ))}
```

```
</tr>
{Object.keys(
  lessonsByCourseDateGroup[courseNumber][page],
).map(day => (
  <tr>
    <td colSpan="2">
      <table className="tb-full">
        <tr>
          <td className="leg-col b">
            <div className="wday flx">
              {day.toUpperCase()}
            </div>
          </td>
          <td>
            <table className="tb-full f-big sm-col">
              <tr>
                <td className="b">
                  <div className="les-num flx">1</div>
                </td>
              </tr>
              <tr>
                <td className="b">
                  <div className="les-num flx">2</div>
                </td>
              </tr>
              <tr>
                <td className="b">
                  <div className="les-num flx">3</div>
                </td>
              </tr>
              <tr>
                <td className="b">
                  <div className="les-num flx">4</div>
                </td>
              </tr>
              <tr>
                <td className="b">
                  <div className="les-num flx">5</div>
                </td>
              </tr>
              <tr>
                <td className="b">
                  <div className="les-num flx">6</div>
                </td>
              </tr>
            </table>
          </td>
        </tr>
      </table>
    </td>
    {Object.keys(
      lessonsByCourseDateGroup[courseNumber][page][day],
    ).map(group => (
      <td>
        <table className="tb-full">
          {Object.keys(
            lessonsByCourseDateGroup[courseNumber][page][
              day
            ][group],
          ).map(lessons => (
            <tr>
              <td className="b">
```

```
                                {Object.keys(
                                  lessonsByCourseDateGroup[courseNumber][
                                    page
                                  ][day][group][lessons],
                                ).length ? (
                                  <>
                                    {Object.keys(
                                      lessonsByCourseDateGroup[
                                        courseNumber
                                      ][page][day][group][lessons],
                                    ).map(lesId => (
                                      <div
                                        className="les-data f-sm flx"
                                        style={{
                                          width:
                                            Object.keys(
                                              lessonsByCourseDateGroup[
                                                courseNumber
                                              ][page][day][group][lessons],
                                            ).length > 1
                                              ?
subgroupLessonContainerWidth
                                              : lessonContainerWidth,
                                        }}
                                      >
                                        {lessonsByCourseDateGroup[
                                          courseNumber
                                        ][page][day][group][lessons][lesId]
                                          ?.id ? (
                                          <>
                                            {
                                              lessonsByCourseDateGroup[
                                                courseNumber
                                              ][page][day][group][lessons][
                                                lesId
                                              ].subject_name
                                            }
                                            <br />
                                            {
                                              lessonsByCourseDateGroup[
                                                courseNumber
                                              ][page][day][group][lessons][
                                                lesId
                                              ].teacher_name
                                            }
                                            <br />
                                            Ауд.{' '}
                                            {
                                              lessonsByCourseDateGroup[
                                                courseNumber
                                              ][page][day][group][lessons][
                                                lesId
                                              ].classroom_code.split(
                                                '-',
                                              )?.[1]
                                            }
                                            <br />
                                            {lessonsByCourseDateGroup[
                                              courseNumber
                                            ][page][day][group][lessons][
                                              lesId
                                            ]?.online_id ? (
```

```
                                                    <>
                                                      {lessonsByCourseDateGroup[
                                                        courseNumber
                                                      ][page][day][group][
                                                        lessons
                                                      ][lesId]?.link ? (
                                                        <>
                                                          {
lessonsByCourseDateGroup[
                                                              courseNumber
                                                            ][page][day][group][
                                                              lessons
                                                            ][lesId].link
                                                          }{' '}
                                                          <br />
                                                        </>
                                                      ) : null}
                                                      Ідентиф.:{' '}
                                                      {
                                                        lessonsByCourseDateGroup[
                                                          courseNumber
                                                        ][page][day][group][
                                                          lessons
                                                        ][lesId].online_id
                                                      }{' '}
                                                      Пароль:{' '}
                                                      {
                                                        lessonsByCourseDateGroup[
                                                          courseNumber
                                                        ][page][day][group][
                                                          lessons
                                                        ][lesId]?.online_code
                                                      }
                                                    </>
                                                  ) : null}
                                                </>
                                              ) : null}
                                            </div>
                                          ))}
                                        </>
                                      ) : (
                                        <div
                                          className="les-data f-sm flx"
                                          style={{ width: lessonContainerWidth }}
                                        ></div>
                                      )}
                                    </td>
                                  </tr>
                                ))}
                              </table>
                            </td>
                          ))}
                        </tr>
                      ))}
                    </table>
                  ),
                )}
              </>
            ))}
          </div>
        </body>);};
```

Додаток Г

КОДЕКС АКАДЕМІЧНОЇ ДОБРОЧЕСНОСТІ
ЗДОБУВАЧА ВИЩОЇ ОСВІТИ ХЕРСОНСЬКОГО
ДЕРЖАВНОГО УНІВЕРСИТЕТУ

Я, Коломієць Олег Едуардович, учасник(ця) освітнього процесу Херсонського державного університету, **УСВІДОМЛЮЮ**, що академічна доброчесність – це фундаментальна етична цінність усієї академічної спільноти світу.

ЗАЯВЛЯЮ, що у своїй освітній і науковій діяльності **ЗОБОВ'ЯЗУЮСЯ**:

– дотримуватися:

- вимог законодавства України та внутрішніх нормативних документів університету, зокрема Статуту Університету;
- принципів та правил академічної доброчесності;
- нульової толерантності до академічного плагіату;
- моральних норм та правил етичної поведінки;
- толерантного ставлення до інших;
- дотримуватися високого рівня культури спілкування;

надавати згоду на:

- безпосередню перевірку курсових, кваліфікаційних робіт тощо на ознаки наявності академічного плагіату за допомогою спеціалізованих програмних продуктів;
- оброблення, збереження й розміщення кваліфікаційних робіт у відкритому доступі в інституційному репозитарії;
- використання робіт для перевірки на ознаки наявності академічного плагіату в інших роботах виключно з метою виявлення можливих ознак академічного плагіату;

– самостійно виконувати навчальні завдання, завдання поточного й підсумкового контролю результатів навчання;

– надавати достовірну інформацію щодо результатів власної навчальної (наукової, творчої) діяльності, використаних методик досліджень та джерел інформації;

– не використовувати результати досліджень інших авторів без використання покликань на їхню роботу;

– своєю діяльністю сприяти збереженню та примноженню традицій університету, формуванню його позитивного іміджу;

– не чинити правопорушень і не сприяти їхньому скоєнню іншими особами;

– підтримувати атмосферу довіри, взаємної відповідальності та співпраці в освітньому середовищі;

– поважати честь, гідність та особисту недоторканність особи, незважаючи на її стать, вік, матеріальний стан, соціальне становище, расову належність, релігійні й політичні переконання;

– не дискримінувати людей на підставі академічного статусу, а також за національною, расовою, статевою чи іншою належністю;

– відповідально ставитися до своїх обов'язків, вчасно та сумлінно виконувати необхідні навчальні та науково-дослідницькі завдання;

– запобігати виникненню у своїй діяльності конфлікту інтересів, зокрема не використовувати службових і родинних зв'язків з метою отримання нечесної переваги в навчальній, науковій і трудовій діяльності;

– не брати участі в будь-якій діяльності, пов'язаній із обманом, нечесністю, списуванням, фабрикацією;

– не підроблювати документи;

– не поширювати неправдиву та компрометуючу інформацію про інших здобувачів вищої освіти, викладачів і співробітників;

– не отримувати і не пропонувати винагород за несправедливе отримання будь-яких переваг або здійснення впливу на зміну отриманої академічної оцінки;

– не залякувати й не проявляти агресії та насильства проти інших, сексуальні домагання;

– не завдавати шкоди матеріальним цінностям, матеріально-технічній базі університету та особистій власності інших студентів та/або працівників;

– не використовувати без дозволу ректорату (деканату) символіки університету в заходах, не пов'язаних з діяльністю університету;

– не здійснювати і не заохочувати будь-яких спроб, спрямованих на те, щоб за допомогою нечесних і негідних методів досягати власних корисних цілей;

– не завдавати загрози власному здоров'ю або безпеці іншим студентам та/або працівникам.

УСВІДОМЛЮЮ, що відповідно до чинного законодавства у разі недотримання Кодексу академічної доброчесності буду нести академічну та/або інші види відповідальності й до мене можуть бути застосовані заходи дисциплінарного характеру за порушення принципів академічної доброчесності.

"07" вересня 2020 р. _____ Олег Коломієць
(дата) (підпис) (ім'я, прізвище)