НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

(повне найменування інституту, факультету)

Автоматизованих систем обробки інформації і управління

(повна назва кафедри)

«До захисту допущено»
В.о. завідувача кафедри
_____ *Олександр ПАВЛОВ*
(підпис) (ініціали, прізвище)
“ ” 2020 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Програмне забезпечення інформаційних управляючих систем та технологій»

спеціальності «121 Інженерія програмного забезпечення»

на тему *Сервіс автоматизованої перевірки кваліфікаційних випускних робіт на плагіат*

Виконав: студент IV курсу, групи *ІП-61Блануца Дмитро Сергійович* _____
(прізвище, ім’я, по батькові) (підпис)

Керівник *професор, д.т.н., професор Стеценко І. В.* _____
посада,науковий ступінь,вчене звання,прізвище,і ім’я, по батькові (підпис)

Консультант з графічної документації *доц., к.т.н., Ліщук К.І.* _____
посада,науковий ступінь,вчене звання,прізвище,і ім’я, по батькові (підпис)

Рецензент:

доц., к.т.н., Клименко І. А.
посада,науковий ступінь,вчене звання,прізвище,ім’я, по батькові (підпис)

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2020 року

UNICHECK

Власник документу:
Попенко Володимир Дмитрович

Дата перевірки:
14.06.2020 01:15:55 EEST

Дата звіту:
14.06.2020 21:02:16 EEST

ID перевірки:
1004022440

Тип перевірки:
Doc vs Internet + Library

ID користувача:
77149

Назва документу: Blanutsa_ip61

ID файлу: 1004035514 **Кількість сторінок:** 45 **Кількість слів:** 7675 **Кількість символів:** 58619 **Розмір файлу:** 121.68 KB

19% Схожість

Найбільша схожість: 4.23% з джерело бібліотеки. **ID файлу:** 1000757531

5.86% Схожість з Інтернет джерелами	106	Page 47
16.5% Текстові збіги по Бібліотеці акаунту	224	Page 47

0.56% Цитат

Цитати	1	Page 48

Вилучення переліку посилань вимкнено

0% Вилучень

Вилучений текст відсутній

Підміна символів

Заміна символів	9

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет (інститут) *Інформатики та обчислювальної техніки*
(повна назва)

Кафедра *автоматизованих систем обробки інформації і управління*
(повна назва)

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – *121 Інженерія програмного забезпечення*

Освітньо-професійна програма – *Програмне забезпечення інформаційних управляючих систем та технологій*

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ *Олександр ПАВЛОВ*
(підпис)
“___”_____2020 р.

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ СТУДЕНТУ

Блануці Дмитру Сергійовичу
(прізвище, ім’я, по батькові)

1. Тема проєкту «*Сервіс автоматизованої перевірки кваліфікацій випускних робіт на плагіат*»
керівник проєкту *Стеценко Інна В’ячеславівна, д.т.н., професор*

(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від “07” травня 2020 р. №1081-с

2. Термін подання студентом проєкту «08» *червня 2020 року*

3. Вихідні дані до проєкту

Технічне завдання

4. Зміст пояснювальної записки

1) Аналіз вимог до програмного забезпечення: основні визначення та терміни, опис предметного середовища, огляд існуючих технічних рішень та відомих програмних продуктів, розробка функціональних та нефункціональних вимог

2) Моделювання та конструювання програмного забезпечення: моделювання та аналіз програмного забезпечення, засоби розробки, технічні рішення, архітектура програмного забезпечення

3) Тестування програмного забезпечення

4) Розгортання та впровадження програмного забезпечення

5. Перелік графічного матеріалу

1) Схема бази даних

2) Схема структурна класів програмного забезпечення

3) Схема структурна компонентів програмного забезпечення

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання «10» *березня 2020 року*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	*Вивчення рекомендованої літератури*	*10.04.2020*	
2.	*Аналіз існуючих методів розв'язання задачі*	*13.04.2020*	
3.	*Постановка та формалізація задачі*	*17.04.2020*	
4.	*Аналіз вимог до програмного забезпечення*	*17.04.2020*	
5.	*Алгоритмізація задачі*	*24.04.2020*	
6.	*Моделювання програмного забезпечення*	*24.04.2020*	
7.	*Обґрунтування використовуваних технічних засобів*	*24.04.2020*	
8.	*Розробка архітектури програмного забезпечення*	*01.05.2020*	
9.	*Розробка програмного забезпечення*	*08.05.2020*	
10.	*Налагодження програми*	*12.05.2020*	
11.	*Виконання графічних документів*	*27.05.2020*	
12.	*Оформлення пояснювальної записки*	*25.05.2020*	
13.	*Подання ДП на попередній захист*	*15.05.2020*	
14.	*Подання ДП рецензенту*	*11.06.2020*	
15.	*Подання ДП на основний захист*	*12.06.2020*	

Студент _____ Дмитро БЛАНУЦА
(підпис)

Керівник _____ Інна СТЕЦЕНКО
(підпис)

№з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проєкт	2	
2	А4	КПІ.ІП-6102. 045420.01.90	Відомість проекту	1	
3	А4	КПІ.ІП-6102. 045420.02.81	Пояснювальна записка	61	
4	А4	КПІ.ІП-6102. 045420.03.91	Технічне завдання	11	
5	А4	КПІ.ІП-6102. 045420.04.51	Програма та методика тестування,	7	
6	А4	КПІ.ІП-6102. 045420.05.34	Керівництво користувача	9	
7	А4	КПІ.ІП-6102. 045420.06.13	Опис програми	24	
8	А3	КПІ.ІП-6102. 045420.07.99 СБД	Схема бази даних	1	
9	А3	КПІ.ІП-6102. 045420.08.99 СС	Схема структурна класів програмного забезпечення	1	
10	А3	КПІ.ІП-6102. 045420.09.99 СС	Схема структурна компонентів програмного забезпечення	1	

Зм.	Арк.	ПІБ	Підп.	Дата	КПІ.ІП-6102.045420.01.90		
Розробн.		Блануца Д. С.			Відомість дипломного проєкту	Літ.	Лист 1 / Листів 1
Керівн.		Стеценко І. В.					
Консульт						КПІ ім.Ігоря Сікорського кафедра АСОІУ гр. ІП-61	
Н/контр.		Ліщук К.І.					
В.о.зав.каф.		Павлов О.А.					

АНОТАЦІЯ

Пояснювальна записка до дипломного проєкту: 61 с., 26 рис., 8 табл., 4 додатки, 15 джерел.

Об’єкт дослідження: процес автоматизованої перевірки кваліфікаційних випускних робіт на плагіат.

Мета дипломного проєкту: зменшити витрати часу та непродуктивні зусилля під час перевірки кваліфікаційних випускних робіт за рахунок автоматизації процесу перевірки кваліфікаційних робіт на плагіат та повідомлення усіх учасників процесу про по точний стан перевірки.

У першому розділі була проаналізована предметна область та вимоги до програмного забезпечення. Були розроблені функціональні вимоги, опис варіантів використання, матриця покриття та їх структурна схема. Також у розділі були описані нефункціональні вимоги до системи.

У другому розділі описана архітектура, діаграма класів, специфікація методів, специфікація полів, використані підходи та інструменти до написання програми. А також розроблена структура бази даних.

У третьому розділі наведений план для супроводу та впровадження розробленого програмного забезпечення.

У четвертому розділі описаний процес тестування та його резутьтати.

АВТОМАТИЗАЦІЯ, ОПТИМІЗАЦІЯ, ПЛАГІАТ, НОТИФІКАЦІЯ, СЕРВЕРНІ СИСТЕМИ, ВЕБ ЗАСТОСУВАННЯ.

ABSTRACT

Explanatory note to the degree project: 61 pp., 26 fig., 24 tables, 4 applications, 8 references.

The object of study: the process of automated plagiarism detection in qualifying papers.

The aim of the degree project: reduce time and unproductive efforts during the verification of qualifying final works by automating the process of verifying qualifying works for plagiarism and notifying all participants in the process of the exact status of the verification.

In the first section analyzed the subject area and software requirements. Functional requirements, a description of use cases, a coating matrix and their block diagram were developed. The section also described non-functional system requirements.

In the second section architecture diagram, class diagrams, method specifications, field specifications, approaches, and software development tools were described. Database ER diagram was also developed.

In the third section provides a plan for maintenance and implementation of the developed software.

In the fourth section describes the testing process and its results.

AUTOMATION, OPTIMIZATION, PLAGIARISM, NOTIFICATION, SERVER SYSTEMS, WEB APPLICATION

Пояснювальна записка
до дипломного проєкту

на тему: *Сервіс автоматизованої перевірки кваліфікаційних випускних робіт на плагіат*

Київ – 2020 року

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

API (*Application Programming Interface*) — це набір готових класів, процедур, функцій, структур і констант, що надаються додатком (бібліотекою, сервісом) для використання в зовнішніх програмних продуктах (Вікіпедія).

CRUD (*create read update delete*) 4 базові функції управління даними «створення, зчитування, зміна і видалення».

POJO (*Plain Old Java Object*) – звичайний *Java,* що не наслідується ні від якого специфічного класу, а просто зберігає данні.

Парсинг (від англ. Parse) – процес аналізу або розбору певного контенту на складові за допомогою роботів-парсерів (спеціальних програм або скриптів). У SEO цим контентом є html-код сторінок сайтів.

JSON (JavaScript Object Notation) - це легкий формат обміну даними. Людям легко його читати і писати. Машини його легко розбирають і генерують. Він заснований на підмножині стандарту мови програмування JavaScript 3-е видання ECMA-262; JSON - це текстовий формат, який повністю не залежить від мови, але використовує конвенції, знайомі програмістам сімейства мов C, включаючи C, C ++, C #, Java, JavaScript, Perl, Python та багато інших. Ці властивості роблять JSON ідеальним форматом обміну даними.

Демон - це тип програми в операційних системах, схожих на Unix, який працює ненав’язливо у фоновому режимі, а не під безпосереднім контролем користувача, чекаючи, що його активують внаслідок настання конкретної події чи умови.

Кластер - в комп'ютерній системі, це група серверів та інших ресурсів, які діють як єдина система і забезпечують високу доступність і, в деяких випадках, балансування навантаження та паралельну обробку.

IDE - це інструмент розробки програмного забезпечення, який в основному використовується розробниками для написання та тестування програм або

програмного забезпечення. Це насправді форма міграції примітивних текстових редакторів для використання більш повнофункціональних технологій, які можуть допомогти вам швидше та ефективніше редагувати код.

ВСТУП

Перевірка кваліфікаційних випускних робіт керівником, консультантами, нормоконтролером, адміністрацією, секретарями тощо є багатоетапним процесом зі значною кількістю учасників і потребує великих зусиль і часових витрат на виконання необхідних операцій, багаторазових повторних перевірок, комунікації, які супроводжують кожен етап. Суттєві витрати часу ідуть також на повідомлення студентів про хід перевірки, який часто потребує особистої участі всіх або декількох учасників контролю. Непродуктивні витрати часу висококваліфікованих фахівців, персоналу і студентів значно впливають на підготовку.

Для вирішення цього питання потрібно вирішити задачу вибору технологій, за допомогою яких буде реалізовуватись система автоматизованої перевірки кваліфікаційних робіт на плагіат. Також потрібно розробити схему та алгоритм, за якою будуть «спілкуватись» розроблена мною система та безпосередньо сервіс перевірки кваліфікаційних робіт на плагіат - Unicheck.

Дипломний проєкт присвячений розробці програмного забезпечення автоматизованої перевірки кваліфікаційних робіт на плагіат.

Практичне значення одержаних результатів. Результатом розробленого програмного забезпечення для дипломного проєкту є сервіс, який дозволить автоматизувати інтеграцію з Unicheck та надасть можливість нагадувати студентам про необхідність надсилання кваліфікаційних робіт до системи KPI-CONNECT, а також інформувати студентів щодо результатів перевірки.

1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Загальні положення

Плагіат — це привласнення авторства на чужу роботу або винахід, також це використання у своїх працях чужого твору без посилання на автора.

Визначення терміну плагіату з’явилося в українських нормативних документів лише в 2001 році в Законі України «Про авторське право і суміжні права». Але таке явище як плагіат має багатовікову історію. Почали засуджувати крадіжки чужих творів правителі, мислителі та поети того часу – Марціал, Ювенал, Вергілій, Горацій. Вперше слово «плагіатор» застосував Мораціал, що з латини перекладається як «викрадач», до літературних грабіжників.

Перша теоретика плагіату з’явилася у Франції в сімнадцятому столітті. Першим радником щодо використання синонімів та інших технік за для маскування своєї нездатності на створення чогось нового був Франсуа Ла Мот ле Вайс у своїй роботі «Академія ораторів». Зараз єдиного, вичерпного та загальноприйнятого терміну плагіату не існує, але автори одних з найвідоміших англомовних сайтів виявлення плагіату наводять такі визначення:

- вкрасти ідеї або слова іншої людини і видати їх за власні;
- використати результати роботи іншої людини без вказання джерела, звідки вони були взяті;
- повністю або частково вкрасти мистецький, науковий або інший твір чи роботу та видати їх за свою;
- представити вже існуючу ідею або продукт як новий та оригінальний.

Закон України «Про освіту» (п.4, статті 42) дає визначення «академічного плагіату — оприлюднення (частково або повністю) наукових (творчих) результатів, отриманих іншими особами, як результатів власного дослідження (творчості) та/або відтворення опублікованих текстів (оприлюднених творів мистецтва) інших авторів без зазначення авторства». [1]

Один з найвідоміших у світі сайтів антиплагіатних систем Turnitin запропонував класифікувати плагіат за десятьма типами, що є переліком видів академічного плагіату, який був створений за результатами опитування більше 600 викладачів вищих навчальних закладів зі всього світу.

- Clone («Submitting another's work, word-for-word, as one's own») повне «сліпе» копіювання (слово-в-слово) без зазначення оригінального автора.
- CTRL-C («Containing significant portions of text from a single source without alterations»): містить значну частину з одного джерела.
- Find - Replace («Changing key words and phrases but retaining the essential content of the source») зберігається основний зміст джерела зі зміною ключових слів і фраз.
- Remix («Mixing paraphrased material from multiple sources»): фрази кількох кількох джерел упорядковуються так, щоб текст виглядав цілісним.
- Recycle («Borrowing generously from one's previous work without citation») це, по суті, дублювання автором власних результатів («самоплагіат»), значних шматків раніше опублікованих текстів без посилань.
- Hybrid («Combining perfectly cited sources with copied passages without citation») - це бездоганне поєднання цитованих джерел і скопійованих абзаців без посилання.
- Mashup («Mixing copied material from multiple sources») - це мікс, змішування скопійованих матеріалів із кількох джерел.
- 404 Error («Citing non-existent sources or including inaccurate information about sources») - «Помилка 404»: текст містить посилання на неіснуючі джерела, недостовірні відомості про джерела.
- RSS Feed («Including proper citation of sources but containing almost no original work»): текст має належне оформлення цитат, але майже не містить оригінальних думок.
- Re-tweet («Including propercitation butrelying too closely on the text's original wording and/or structure»): текст містить належне цитування, але, по суті,

дублює формулювання та/або структуру первісного (оригінального) тексту. [2, 3]

Системи виявлення схожості тексту реалізують один з двох загальних підходів виявлення, один я яких є внутрішнім, а інший – зовнішній.

Зовнішні системи виявлення порівнюють текст з словниковою колекцією, яка може містити тексти з яких могли запозичити чужу роботу. На основі вибраної моделі документа і передчасно вибраних критеріїв визначаються всі документи, які містять текст який схожий на перевіряючий текст, та відсоток плагіату в цих документах має бути в передчасно вибраному проміжку.

Внутрішні системи виявлення плагіату аналізують тільки текст, що перевіряється, не порівнюючи з іншими джерелами. Цей підхід має за мету виявити зміни в унікальності стилю автора як індикатор потенціального плагіату. Внутрішні системи не можуть точно виявити плагіат без людського судження. Схожості виявляють за допомогою передчасно визначених моделей документів та можуть представляти хибні результати.

Було проведено дослідження для перевірки ефективності програмного забезпечення, що було виявлене у навчальних закладах вищого рівня освіти. Було відібрано дві групи студентів, першу з них заздалегідь повідомили, що їх роботи будуть перевіряти за допомогою системи виявлення плагіату. Студенти другої групи писали статті без будь-якої інформації про перевірку плагіату. Дослідники думали, що відсоток плагіату в першій групі буде меншим, але виявили приблизно однакові відсотки.

На Рисунку 1.1 представлена класифікація всіх підходів виявлення, використовуваних у теперішньому часі для виявлення плагіату за допомогою комп'ютера. Методи характеризуються типом оцінки схожості яку вони проводять: глобальна або локальна.

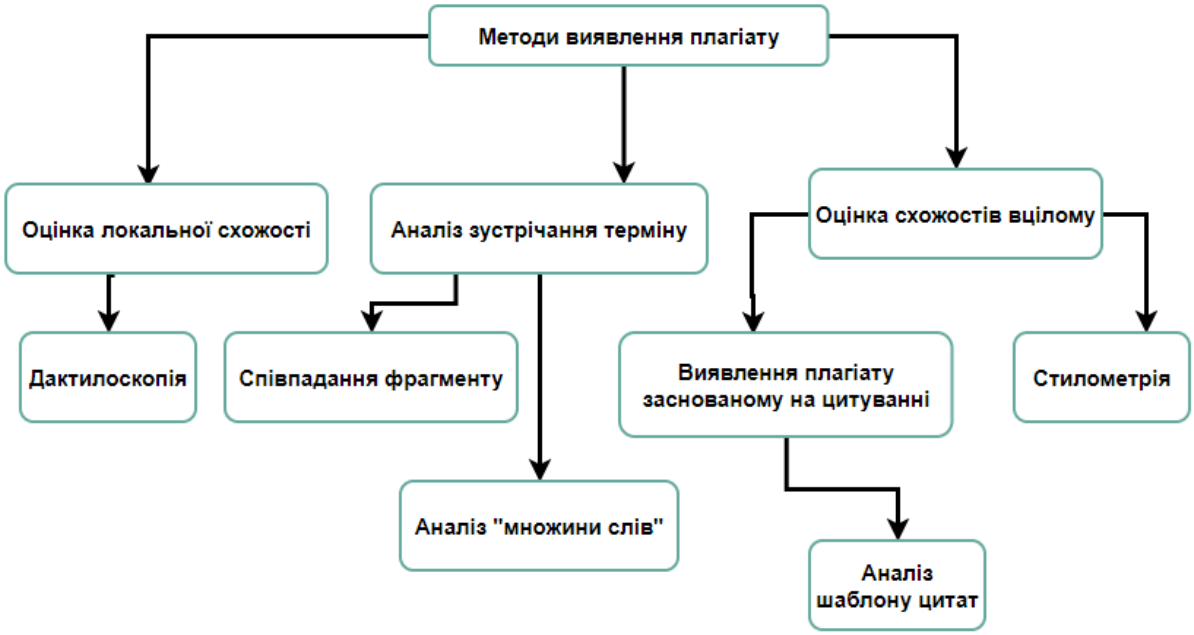


Рисунок 1.1 - Класифікація методів комп'ютерного виявлення плагіату з технічної точки зору

В останні роки в університетах України почали активно використовувати різні системи перевірки випускних робіт на плагіат. Цей процес доволі складний, адже перед перевіркою на плагіат робота студента має пройти ще два проміжні етапи: затвердження дипломним керівником та нормоконтролером. Ці три етапи складно відслідковувати та синхронізувати всіх учасників цього процесу. Зараз студенти надсилають свої роботи на пошту та/або на Google диск. Потім студент повідомляє свого керівника, що можна починати перевірку, роботи. Після цього, якщо керівник затвердив роботу, повідомляє нормоконтролера, що можна починати перевірку роботи що до стандартів форматування керівник. Та вже на останньому етапі людина, яка відповідає за антиплагіат, має вручну завантажити роботу студента до системи перевірки, чекати закінчення перевірки та повідомити щодо результатів перевірки керівника диплому та студента. Звісно, такий стан процесу перевірки не відповідає сучасному розвитку інформаційних технологій.

Саме тому у платформи KPI-CONNECT один з основних завдань було автоматизація процесу перевірки та затвердження випускних робіт студентів.

Ця дипломна робота розроблялась як інтеграційне рішення, у вигляді Java мікросервісу, автоматизації перевірки випускних робіт студентів платформи KPI-CONNECT на плагіат за допомогою такої системи перевірки на плагіат як Unicheck.

1.2 Аналіз успішних IT-проєктів

Розроблена система складається з двох функціональних частин: система інтеграції з Unicheck, та система надсилання e-mail листів.

Аналогів для системи я не знайшов. Адже ця система має тісно контактувати з системою KPI-CONNECT, а сервіси перевірки на плагіат не дають можливості змінювати таку велику частину їх функціоналу.

Так як вирішено інтегруватись з Unicheck далі представлений огляд його конкурентів.

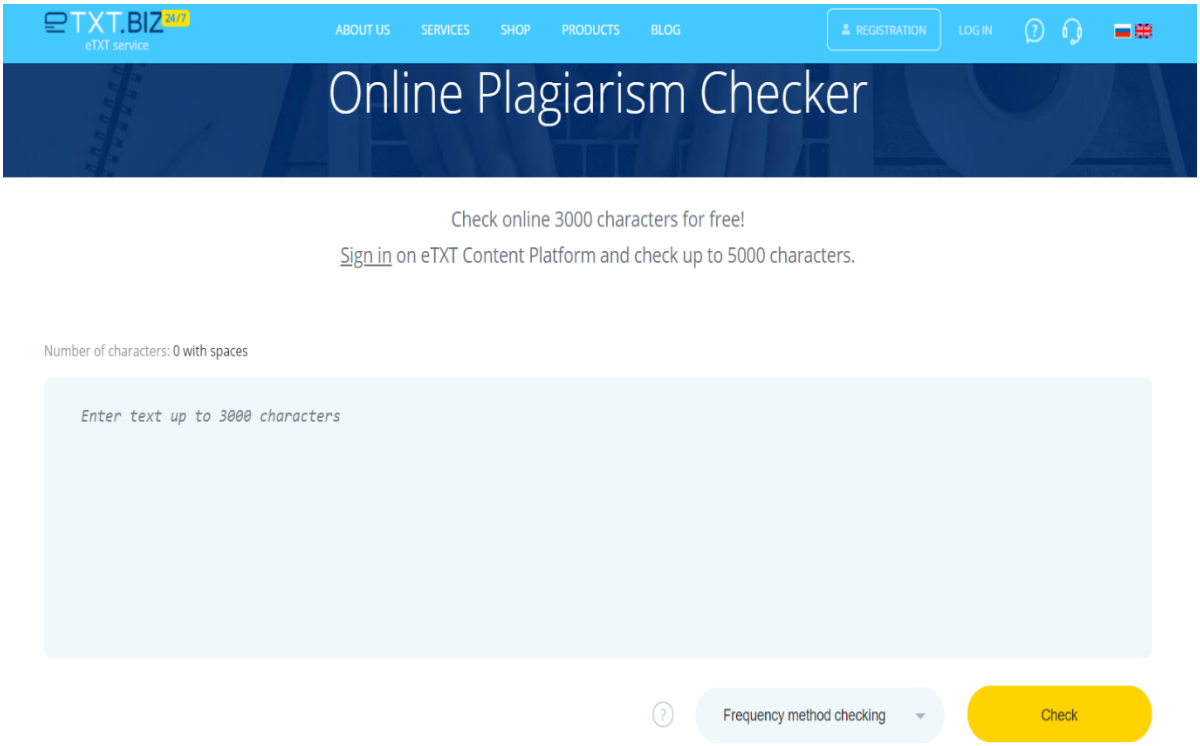


Рисунок 1.2 - Інтерфейс сайту etxt.biz

eTXT Антиплагиат

Сервіс перевірки текстів на плагіат, що доступний як програмне забезпечення для установки на персональний комп'ютер, так і у форматі онлайн ресурсу. Можливо здійснювати перевірку текстових фрагментів, окремих файлів та пакетів файлів, а також сторінок сайтів.

На Рисунку 1.2 представлена форма для перевірки тексту на плагіат з сайту etxt.biz.

Advego Plagiatus

Програма пошуку в мережі Інтернет часткових або повних копій текстового документа, що показує ступінь унікальності тексту, список джерел тексту, відсоток збігу текстів. Також доступна перевірка унікальності за URLадресою. Найновіша версія Advego Plagiatus 3.0.10.

Далі на Рисунку 1.3 представлений інтерфейс програми Advego Plagiatus.

Недоліки**:** сканування, навіть у режимі швидкої перевірки, триває досить довго.

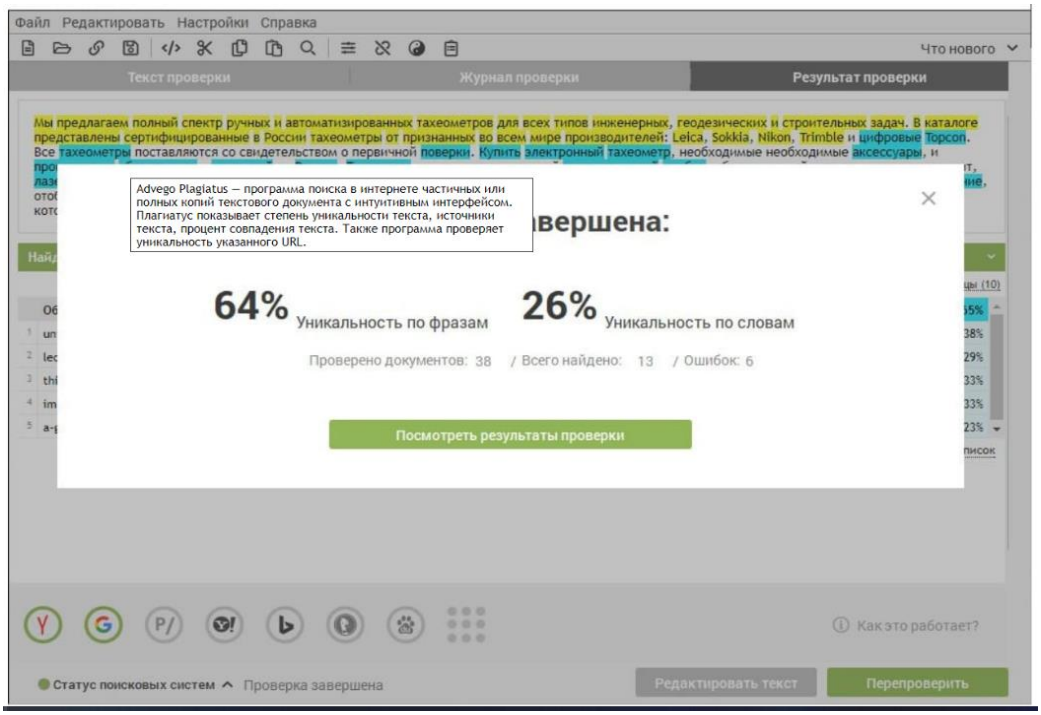


Рисунок 1.3 - Інтерфейс програми Advego Plagiatus

Рисунок 1.4 – Інтерфейс програми Double Content Finder

Double Content Finder

Безкоштовна програма для перевірки тексту на унікальність по мережі Інтернет за допомогою пошукової машини Яндекс. Не потребує встановлення на комп'ютер. Робота з програмою можлива будь-яким з трьох способів: додаванням тексту з буферу обміну, завантаженням текстового файлу або зазначенням веб-адреси, за якою розташована потрібна стаття. У підсумковому звіті виводиться список адрес, за якими розташовані копії статті (кількість адрес обмежено 50–ма). На даний момент підтримка програми припинена.

Вище на Рисунок 1.4 представлений інтерфейс програми Double Content Finder.

Недоліки: підтримка лише російської мови, програма більше не підтримується.

Viper

Простий, точний та безкоштовний інструмент виявлення плагіату, що допомагає здійснити пошук по широкомасштабній базі документів (10 мільярдів одиниць), призначений для сканування лише англомовних текстів.

Далі на Рисунку 1.5 зображений інтерфейс програми Viper.

Недоліки: обмеження кількості запитів/на день; англомовний інтерфейс, не зрозумілий на перший погляд.

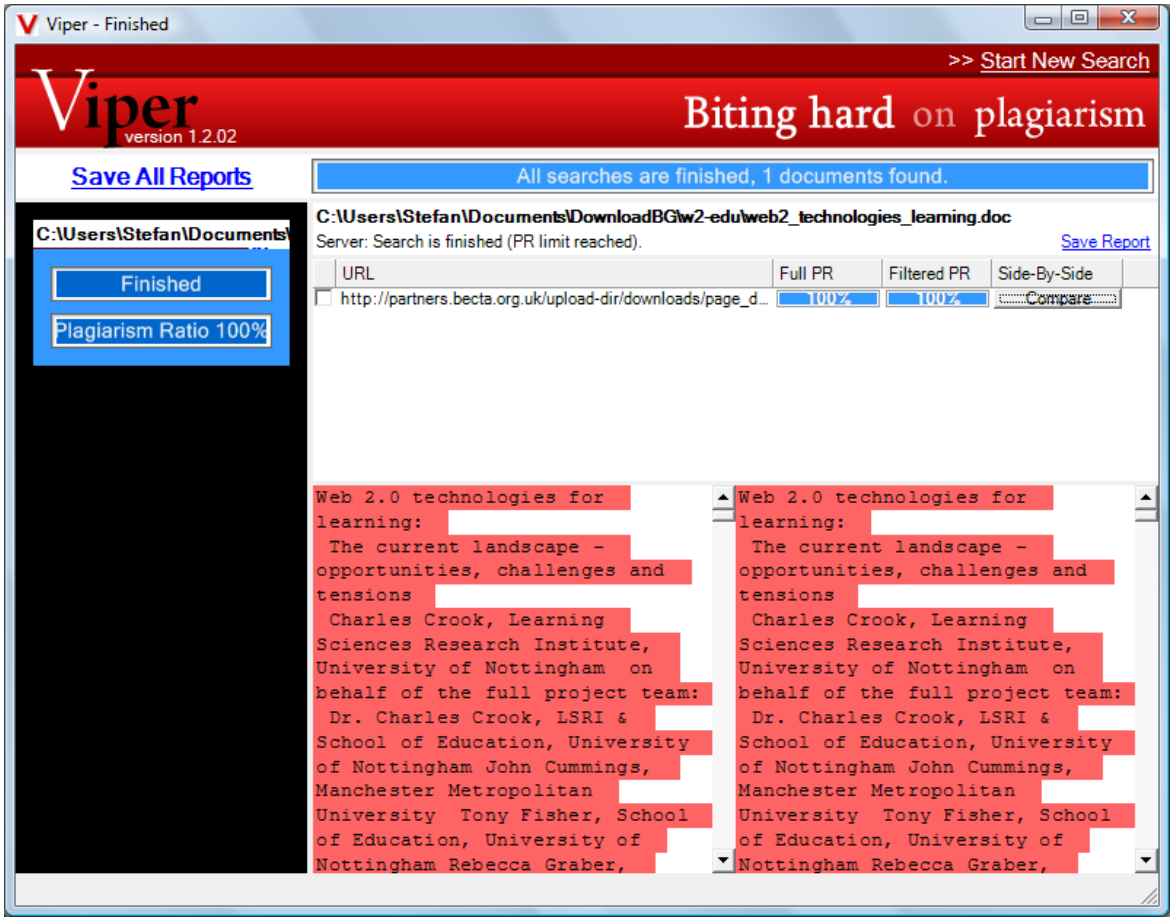


Рисунок 1. 5 – Інтерфейс програми Viper

PaperRater

Безкоштовний онлайн сервіс, що містить автоматизований редактор для перевірки граматики, правопису та стилістики рукописів, а також інструмент

виявлення плагіату, шляхом використання потужностей основних пошукових машин (Yahoo!, Bing, Google).

Далі на Рисунку 1.6 представлений інтерфейс сайту paperrater.com.

Даний ресурс простий у використанні та дозволяє здійснити комплексну перевірку текстів під час підготовки до друку. Інтерфейс інтуїтивно зрозумілий. Для пошуку плагіату необхідно скопіювати та ввести текст рукопису у відповідне поле на сайті. Не має жодних обмежень щодо розмірів тексту.

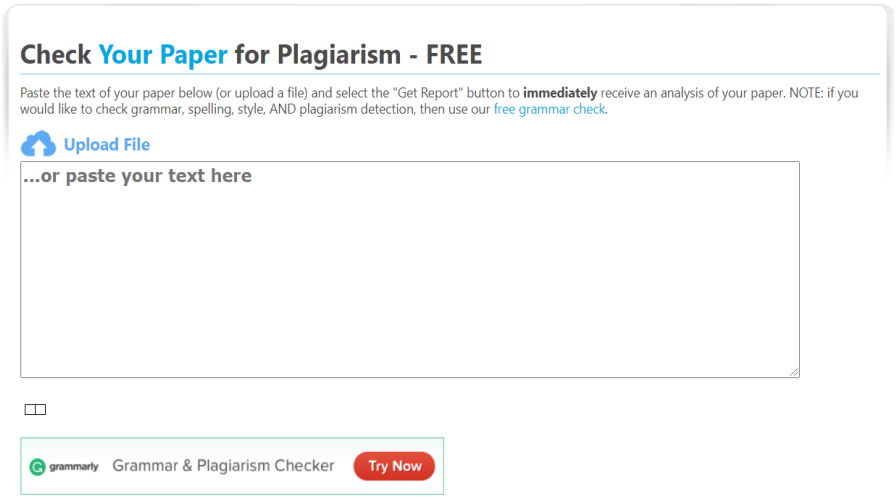


Рисунок 1.6 - Інтерфейс сайту paperrater.com

Результатом перевірки є повідомлення по наявність плагіату, частка оригінального тексту рукопису та посилання на веб-сторінки, що можуть містити подібний текст.

Недоліки**:** підсумковий звіт досить стислий та не містить статистичних даних, не відображено запозичені фрагменти тексту, приведено лише 3 зовнішні джерела.

Водночас, Unicheck не має вище перерахованих недоліків. Зокрема вміє він шукає плагіат порівнюючи з даними в інтернеті або в вибраній бібліотеці файлів також має зручне та зрозуміле API. Сервіс сам повідомить мою систему

коли закінчиться перевірка та формування звіту. Також цей сервіс вже давно використовує КПІ.

Приведу основні переваги Unicheck:

- можливість отримувати звіт перевірки багатьма мовами (англійська, німецька, французька, італійська, китайська, українська, російська тощо);
- можлива паралельна робота викладачів та відповідальних за перевірку на плагіат;
- для початку роботи з сервісом не потрібно встановлювати стороннє програмне забезпечення;
- швидка перевірка (2 - 4 секунди на сторінку);
- сервіс працює з багатьма популярними форматами (PDF, DOC, DOCX, XLS, XLSX, PPT, PPTX, ODT, ODS, ODP, RTF, PAGES, TXT, HTM, HTML, ZIP, RAR);
- інтуїтивний інтерфейс, що не потребує тривалого освоєння;
- можливість завантажувати файли з Google Drive, OneDrive, Dropbox;
- щоденне збільшення пошукової бази та індексу за рахунок додавання репозитаріїв, архівів журналів та баз робіт. [4]

1.3 Постановка завдання

До програмного забезпечення були висунуті наступні вимоги.

Користувач системи за допомогою застосування взаємодіє з розробленим мною API для використання таких функцій:

- завантаження кваліфікаційних робіт до системи;
- отримання нагадувань щодо завантаження кваліфікаційних робіт до системи;
- отримання повідомлення за результатами перевірки кваліфікаційної роботи на плагіат.

Адміністратор системи за допомогою додатку взаємодіє з розробленим мною API для використання усіх функцій таких, як:

- налаштування нагадувань щодо завантаження кваліфікаційних робіт до системи;
- початок перевірки кваліфікаційних робіт на плагіат;
- відправити будь-яке текстове повідомлення на електрону пошту вибраній групі людей (студентам або викладачам), ці повідомлення можуть бути відправленні відразу або в зазначений час.

1.4 Аналіз вимог до програмного забезпечення

1.4.1 Розроблення функціональних вимог

З функціональними вимогами додатку можна ознайомитися за допомогою структурної схеми варіантів використань, що зображена нижче на Рисунку 1.7.

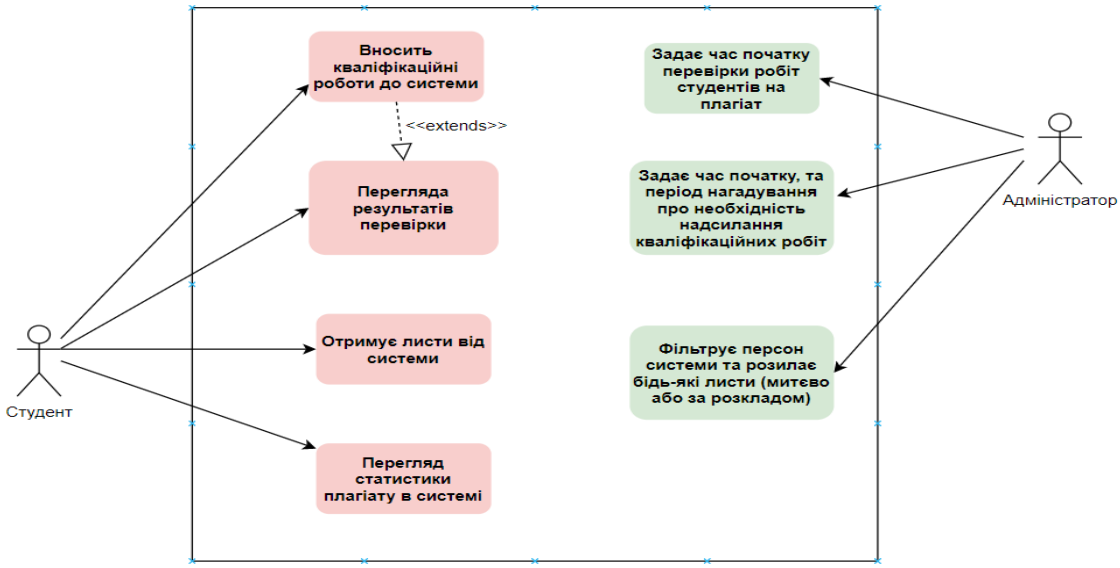


Рисунок 1.7 – Схема структурна варіантів використань

Далі у Таблиці 1.1 присутній опис кожного з варіантів використання.

Таблиця 1.1 – Функціональні вимоги до сервісу

Номер	Назва	Опис
FR01	Створення нагадування адміністратором	Адміністратор має змогу створити нагадування що до надсилання робіт до системи KPI-CONNECT для вибраних груп студентів. Нагадування буде надсилатись в вибрані дні та час до кінцевого терміну завантаження випускних робіт. Ця цього адміністратор вибирає бажані групи з можливістю фільтрування та кафедрі. Адреса запиту: /admin/reminders. Приклад запит у вигляді JSON: **{** **"groupIds": [1, 2],** **"days": [1, 2, 3],** **"time": "14:14",** **"endDate": "2020-02-07"** **}**
FR02	Початок перевірки робіт студентів	Адміністратор повинен мати можливість почати перевірку робіт студентів, для цього адміністратор повинен виконати команду на сервері: **php bin/console plagiarism:cheking:start**

Продовження таблиці 1.1

FR03	Надсилання повідомлень студентам та викладачам	Адміністратор повинен мати можливість надсилати будь-які текстові повідомлення студентам чи викладачам миттєво або у вказаний час. Адреса запиту: /admin/mailing Приклад запит у вигляді JSON: **{** **"profileIds": [1],** **"content": "content",** **"subject": "subject",** **"dateTime": "25/05/2020 10:10"** **}** Щоб надіслати повідомлення миттєво параметр «dateTime» має бути відсутнім.
FR04	Завантаження кваліфікаційної роботи	Студент повинен мати можливість завантажувати кваліфікаційну роботу до системи KPI-CONNECT, для цього студент може скористатись інтерфейсом сайту вище згаданої системи.
FR05	Отримання нагадувань	Студент має отримувати нагадування що до необхідності надсилання роботи до системи KPI-CONNECT. Для цього система буде раз в хвилину шукати в базі створенні адміністратором повідомлення на даний час на надсилати їх.
FR06	Отримання студентом результатів перевірки	Студент має отримувати повідомлення коли закінчиться перевірка його роботи з відсотком плагіату та звітом перевірки, для цього система буде автоматично надсилати студенту відсоток плагіату після отримання результатів від Unicheck.

Базуючись на функціональних вимогах була побудована матриця трасування вимог, з якою можна ознайомитися на Рисунку 1.8.

Варіант використання/Вимога	FR01	FR02	FR03	FR04	FR05	FR06
Завантаження кваліфікаційних робіт до системи						
Отримання нагадувань						
Отримання результатів перевірки						
Налаштування нагадувань						
Початок первірки						
Відправка повідомлення студентам та викладачам						

Рисунок 1.8 - Матриця трасування вимог

1.4.2 Розроблення нефункціональних вимог

До сервісу висуваються наступні нефункціональні вимоги:

- сторінки сайту системи повинні коректно відображатись у всіх останіх версіях браузерів Opera/Chrome/Mozilla Firefox;
- сервіс має працювати лише за умови підключення до інтернету;
- API має давати відповідь на запити протягом не більш, ніж 5 секунд за умови навантаження до 100 одночасних запитів.

1.5 Висновки до розділу

У цьому розділі був описаний процес перевірки кваліфікаційних робіт на плагіат і його важливість. Були проаналізовані технічні рішення, що часто застосовуються у цій сфері, а також був проведений аналіз декількох конкурентів, що хоч і сильно відрізняються від даної розробки, але є найбільш близькими серед сервісів, що вдалося знайти. Також були детально описані функціональні та нефункціональні вимоги до системи.

2 КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Моделювання та аналіз програмного забезпечення

Проаналізуючи функціональні вимоги що були поставленні в першому розділі був розроблений алгоритм роботи інтеграційного сервісу для роботи з Unicheck API, далі на Рисунку 2.1 зображений власне сам алгоритм.

Далі наводиться словесний опис роботи інтеграції з Unicheck.

Спочатку адміністратор має створити нагадування для студентів що до необхідності надсилати кваліфікаційні випускні роботи до системи KPI-CONNECT. Адміністратор має можливість створити нагадування, вибравши дні тижня та час, в який будуть відправлятись нагадування всім студентам з вибраних, адміністратором, груп.

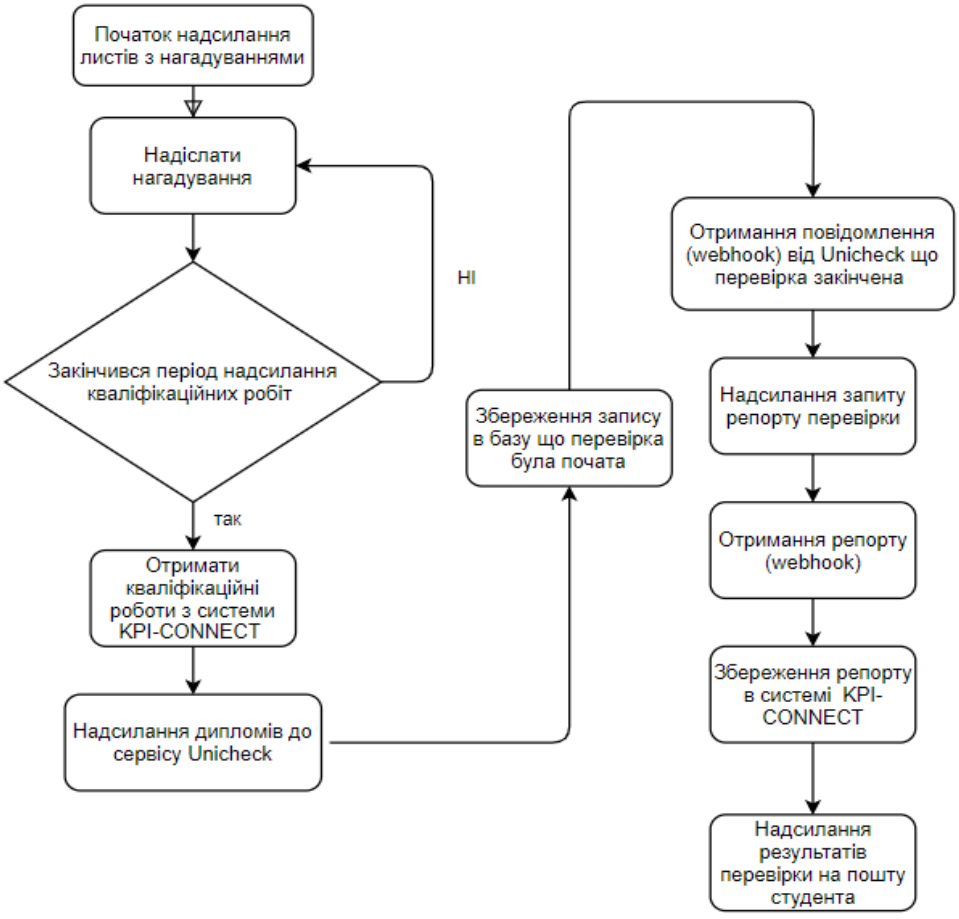


Рисунок 2.1 – Алгоритм роботи системи

Далі адміністратор запускає команду на основному сервері, для ініціалізації процесу інтеграції з Unicheck:

- отримання всіх не перевірених робіт з основного сервісу;
- переведення отриманих робіт у статус DRAFT;
- завантаження основних частин випускних робіт до Unicheck;
- переведення завантажених робіт у статус FILE_UPLOADED;
- початок перевірки робіт Unicheck-ом;
- переведення робіт у статус CHECKING;
- коли перевірка закінчена Unicheck повідомить інтеграційний сервер;
- переведення у статус CHECKED;
- початок генерування звіту перевірки робіт Unicheck-ом;
- переведення у статус REPORT_GENERATING;
- коли генерація звіту буде завершена Unicheck повідомить інтеграційний сервер;
- переведення у статус DONE;
- надсилання результатів до основного сервера;
- основний сервер надсилає результати перевірки на пошту студента.

Також була розроблена структурна схема компонентів програмного забезпечення, яка зображена на Рисунку 2.2.

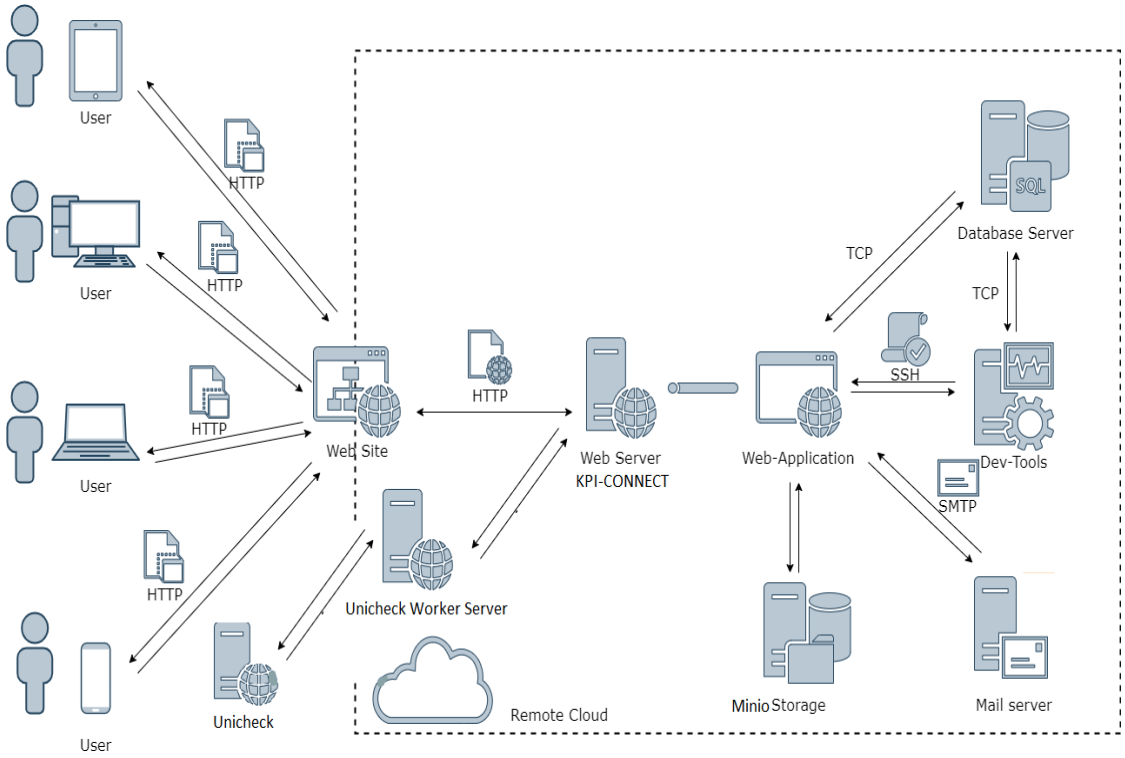


Рисунок 2.2 - Схема структурна компонентів програмного забезпечення

2.2 Технології та фреймворки, що застосовувались

2.2.1 Spring

Spring - найпопулярніша основа розробки додатків для підприємства Java. Мільйони розробників у всьому світі використовують Spring Framework для створення високоефективних систем, легко тестування та багаторазового використання коду.

Spring Framework - це платформа Java з відкритим кодом. Спочатку він був написаний Родом Джонсоном і вперше був випущений за ліцензією Apache 2.0 у червні 2003 року.

Spring легкий, якщо говорити про розміри та прозорість. Базова версія Spring Framework становить близько 2 Мб.

Spring Framework можна використовувати при розробці будь-якого додатку Java, але є розширення для побудови веб-додатків на платформі Java EE.

Spring має на меті полегшити розробку J2EE та просувати хороші практики програмування, включивши модель програмування на основі POJO.

Нижче наведено перелік кількох переваг використання Spring Framework.

- Spring дає можливість розробникам розробляти додатки корпоративного класу за допомогою POJO. Перевага використання лише POJO полягає в тому, що вам не потрібен контейнерний продукт EJB, такий як сервер додатків, але у вас є можливість використовувати тільки надійний контейнер сервлетів, такий як Tomcat або якийсь комерційний продукт.
- Spring організована модульно. Незважаючи на те, що кількість пакетів та класів значна, вам потрібно турбуватися лише про потрібні, а решта ігнорувати.
- Spring не вигадує колесо, натомість справді використовує деякі існуючі технології, як-от ORM, JEE, таймери та JDK та інші технології.
- Тестування програми, написаної за допомогою Spring, є простим, оскільки залежний від середовища код переміщується в контекст тестування. Крім того, за допомогою POJOs JavaBeanstyle стає легше використовувати ін'єкцію залежності для введення даних тесту.

Веб-структура Spring - це добре розроблена веб-структура MVC, яка надає чудову альтернативу JSP.

Spring пропонує зручний API для перехвату помилок та виняткових ситуацій, що стаються при роботі таких технологій JDBC, Hibernate або JDO.

Легкі контейнери IoC мають легку вагу, наприклад, порівняно з контейнерами EJB. Це вигідно для розробки та розгортання програм на комп'ютерах з обмеженою пам’яттю та ресурсами процесора.

Spring пропонує послідовний інтерфейс управління транзакціями, який може масштабувати до локальної транзакції (наприклад, за допомогою єдиної бази даних) і масштабувати до глобальних транзакцій (наприклад, використовуючи JTA).

Ін’єкція залежностей (DI)

Технологія, з якою Spring найбільше ототожнюється, - це аромат Inversion of Control Dependency Injection (DI). Інверсія управління (IoC) - це загальне поняття, і воно може виражатися різними способами. Вприскування в залежності - лише один конкретний приклад інверсії управління.

Під час написання складної Java програми класи повинні бути максимально незалежними від інших класів Java, щоб збільшити можливість повторного використання цих класів та тестування їх незалежно від інших класів під час тестування. Ін'єкційна залежність допомагає склеювати ці класи разом і одночасно зберігати їх незалежними.

Далі на Рисунку 2.3 зображена структурна схема організація модулів Spring Framework.

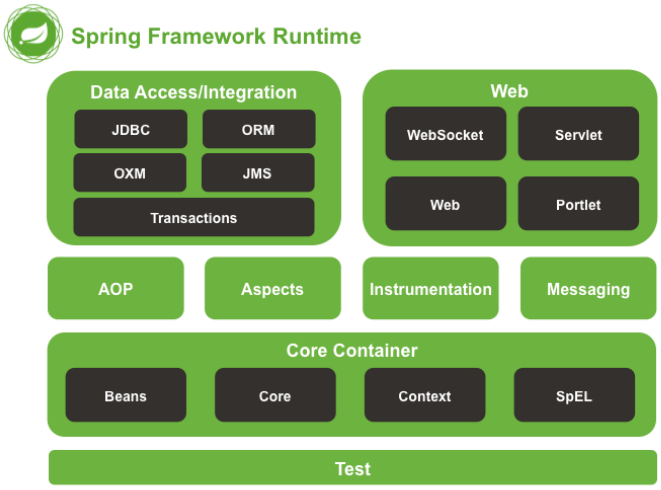


Рисунок 2.3 – Схема структурна організації модулів Spring Framework

Введення залежності може відбуватися шляхом передачі параметрів конструктору або після створення, використовуючи сетер-методи. Оскільки введення залежностей є серцем Spring.

Аспектно-орієнтоване програмування (AOP)

Однією з ключових складових Spring є модулі, що орієнтовані на аспекти (AOP). Функції, що охоплюють кілька точок програми, називаються наскрізними, і ці наскрізні функції концептуально є окремими від бізнес-логіки

програми. Існують різні загальноприйняті хороші приклади аспектів, включаючи реєстрацію, деклараційні транзакції, безпеку, кешування тощо.

Ключовою одиницею модульності в ООП є клас, тоді як в АОР одиниця модульності є аспектом. DI допомагає вам відокремлювати об'єкти додатків один від одного, тоді як АОР допомагає вирішувати наскрізні проблеми об'єктів, на які вони впливають.

Модуль AOP Spring Framework забезпечує реалізацію аспекту, що дозволяє визначати перехоплюючі методи та точкові вирізки для чистого роз'єднаня коду, який реалізує функціональність, яку слід розділити. [5]

2.2.2 OAuth2

Так як система KPI-CONNECT поки що складається з двох частин основний сервер (PHP) та мікросервіс (Java), що й був розроблений. Для їх правильного «спілкування» (щоб мікросервіс мав доступ тільки до потрібних йому ендпоінтів) був застосований авторизаційних фреймворк OAuth2 client credentials grant type.

На Рисунку 2.4 зображена структурна схема роботи OAuth2.

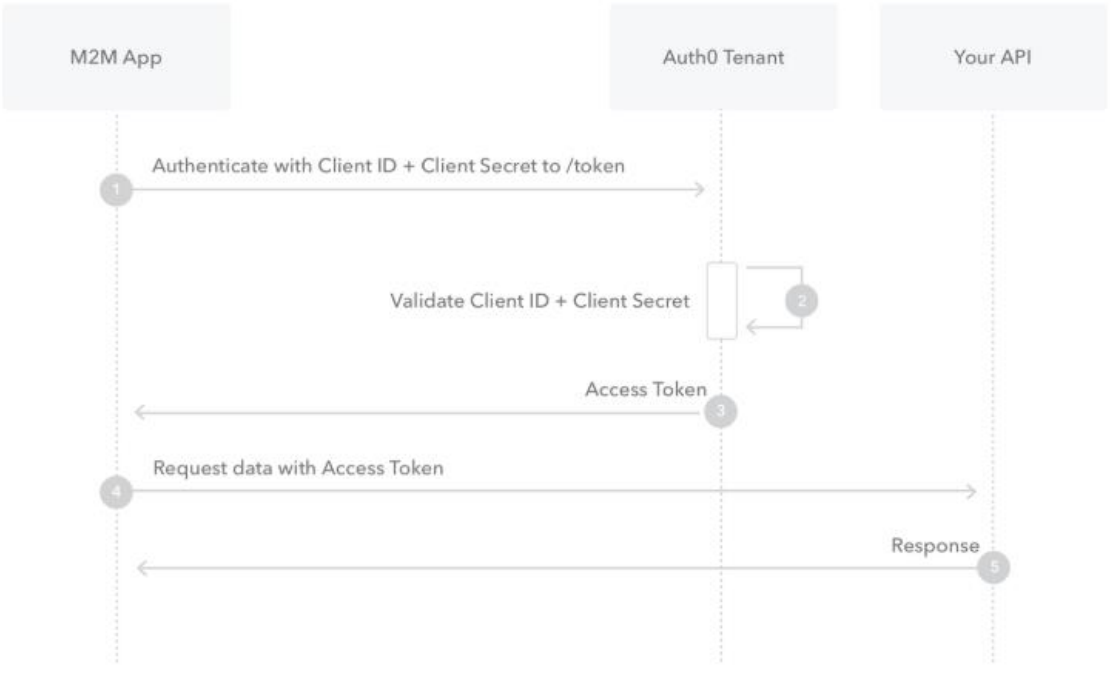


Рисунок 2.4 – Схема структурна робота OAuth2 client credentials grant type

Спочатку сервер, який хоче отримати доступ до ресурсів іншого сервера (далі клієнт) має зареєструватись в авторизаційного сервера. Після реєстрації авторизаційний сервіс повертає клієнту client id та client secret та назначає певні скоупи(ролі). Та коли клієнт захоче отримати доступ до ресурсного сервера він має отримати спеціальний токен у авторизаційного сервера. Для цього клієнт відправляє авторизаційному серверу свої client id та client secret той у відповідь відправляє зашифрований токен в якому містяться інформація про клієнта, зокрема скоупи. Потім клієнт надсилає свій запит вже ресурсному серверу з отриманим раніше токеном, а ресурсний сервер вже дивиться на скоупи клієнта та вирішує давати клієнту доступ чи ні. Не рідко роль авторизаційного та ресурсного сервера виконує один сервер. [6]

2.2.3 Java

Java - це загальноприйнята, об'єктно-орієнтована мова програмування та середовище виконання (JRE), яке складається з JVM, що є наріжним каменем платформи Java.

Для чого використовується Java?

Перш ніж відповісти на запитання, для чого використовується Java, дозвольте мені коротко ознайомити вас з тим, чому слід вибрати Java. Java домінує в галузі IT з початку 2000-х по сьогоднішній день 2020 року.

Java використовується в різних областях, деякі з них перелічені нижче:

- банківська справа**:** для управління транзакціями;
- роздрібна торгівля: додатки для виставлення рахунків, які ви бачите в магазині/ресторані, повністю написані на Java;
- інформаційні технології**:** Java призначена для вирішення залежностей від впровадження;
- Android**:** програми написані на Java або використовують Java API;
- Big Data: популярний фреймворк Hadoop MapReduce написана за допомогою Java;

- науково-дослідницьке співтовариство: для обробки величезної кількості даних.

Далі на Рисунку 2.5 зображені технології, які використовують Java як ядро.

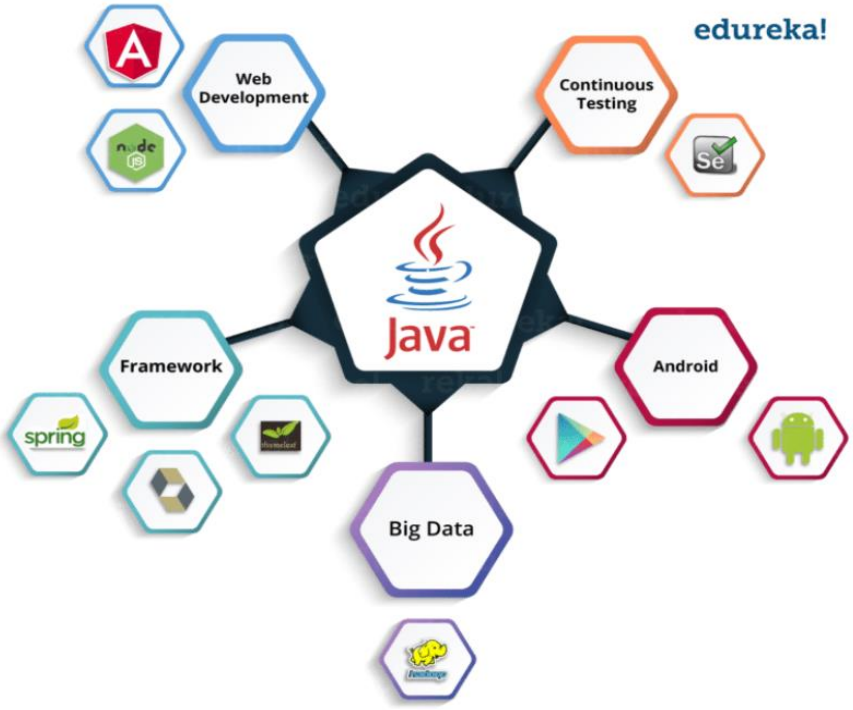


Рисунок 2.5 - Технології, які використовують Java як ядро

Java - мова програмування, розроблена Джеймсом Гослінгом разом з іншими членами команди на ім’я Майк Шерідан та Патрік Нафтон, які також були названі Зеленою командою в 1995 році для Sun Microsystems, а саме для цифрових пристроїв, таких як телеприймачі, телевізори тощо.

Java - це об'єктно-орієнтована мова, схожа на C++, але з розширеними та спрощеними функціями. Java безкоштовна для доступу та може працювати на всіх платформах.

Вона паралельна, завдяки чому ви можете виконати одночасно багато операторів, а не послідовно виконувати їх.

Java незалежна мова програмування, що відповідає логіці "Пишіть один раз, запускайте де завгодно", тобто скомпільований код може працювати на всіх платформах, що підтримують Java.

Основні риси Java:

- незалежність**:** Java - незалежна платформа, тобто будь-яка програма, написана на одній платформі, може бути легко перенесена на іншу платформу;
- об'єктно-орієнтованість**:** все вважається "об'єктом", який має певний стан, поведінку і всі операції виконуються за допомогою цих об'єктів;
- безпека**:** весь код після компіляції перетворюється в байт-код, який не читається людиною і Java не використовує явного вказівника і запускає програми всередині JRE, щоб запобігти будь-якій діяльності з ненадійних джерел. Це дає змогу розробляти системи/програми, що не містять вірусів;
- динамічність**:** Java має можливість адаптуватися до розвиваючого середовища, яке підтримує динамічне розподіл пам’яті, завдяки якому витрата пам’яті зменшується та підвищується продуктивність програми;
- розподіленість**:** Java надає функцію, яка допомагає створювати розподілені програми. Використовуючи віддалений виклик методу (RMI), програма може викликати метод іншої програми в мережі та отримати вихід. Ви можете отримати доступ до файлів, викликавши методи з будь-якої машини в Інтернеті;
- надійність**:** Java має потужну систему управління пам'яттю. Це допомагає усунути помилки, оскільки JVM перевіряє код під час компіляції та виконання;
- висока продуктивність: Java досягає високої продуктивності завдяки використанню байт-коду, який можна легко перевести на рідний машинний код. За допомогою компіляторів JIT (Just-In-Time) Java забезпечує високу продуктивність;
- інтерпретованість: Java компілюється у байт-коди, які інтерпретуються середовищем виконання Java;
- багатопотоковість: Java підтримує декілька потоків виконання, включаючи набір примітивів синхронізації, що значно спрощує програмування використовуючи потоки. [8]

2.2.4 PHP

PHP - це одна з найбільш широко використовуваних серверних та скриптових мов на веб-розробці. Популярні веб-сайти, такі як Facebook, Yahoo, Wikipedia тощо, розроблені за допомогою PHP.

PHP настільки популярний, тому що його дуже просто вивчити, кодувати та розгортати на сервері, тому він був першим вибором для початківців.

PHP розшифровується як гіпертекстовий попередній процесор. PHP - це сценарій мови, що використовується для створення статичних та динамічних веб-сторінок та веб-додатків. Ось кілька важливих речей, які ви повинні знати про PHP:

- PHP - інтерпретована мова, тому компілятор не потребується;
- Для запуску та виконання PHP-коду нам потрібен веб-сервер, на якому потрібно встановити PHP;
- PHP - це сценарна мова на сервері, що означає, що код виконується на сервері, а результат відправляється в браузер у простому HTML;
- PHP є відкритою і безкоштовною мовою програмування;

Деякі функції та випадки використання мови PHP:

- PHP має дуже простий і легкий для розуміння синтаксис, отже, поріг входження менший порівняно з іншими мовами програмування, такими як C та C++;
- PHP є крос-платформою, тому ви можете легко розробляти та переміщувати/розгортати ваш PHP-проект майже для всіх основних операційних систем, таких як Windows, Linux, Mac OSX тощо;
- всі популярні сервіси веб-хостингу підтримують PHP. Також плани веб-хостингу для PHP, як правило, є одними з найдешевших планів через його популярність;
- популярні системи управління, такі як Joomla, Drupal тощо, розроблені за допомогою PHP, і якщо ви хочете запустити власний веб-сайт, ви можете легко зробити це за допомогою PHP;

- за допомогою PHP ви можете створювати статичні та динамічні веб-сторінки, виконувати операції з обробкою файлів, надсилати електронні листи, отримувати доступ та змінювати файли cookie браузера та майже все інше, що можливо захочеться реалізувати у своєму веб-проекті;
- PHP має вбудовану підтримку MySQL, яка є однією з найбільш широко використовуваних систем управління базами даних. [9, 10]

2.2.5 Composer

Composer - це інструмент для управління залежностями в PHP, що дозволяє оголосити бібліотеки, від яких залежить ваш проект, і він буде керувати (встановлювати/оновлювати) їх для вас.

Composer не є менеджером пакунків у тому ж сенсі, що й Yum чи Apt. Так, він має справу з "пакетами" та бібліотеками, але керує ними на основі проекту, встановлюючи їх у каталозі всередині вашого проекту. За замовчуванням він не встановлює нічого глобально. Таким чином, він є менеджером залежності.

Ця ідея не нова, і Composer сильно натхненний від npm і Bundler. [11]

2.2.6 Maven

Maven - це інструмент, яким багато розробників Java користуються щодня. Ви можете розпочати роботу з Maven і досить швидко включити її у своє повсякденне життя. Встановіть Maven, завантажте, оформіть замовлення або створіть новий проект Maven, запустіть команду “mvn install”l та запустіть додаток через цілі плагіна, такі як mvn jetty: run. Коли вам потрібно додати бібліотеку Java, відредагуйте файл pom.xml, щоб додати до нього розділ <dependency>. Це найпоширеніші завдання, які ви робите з Maven.

Природно, час від часу вам потрібно налаштувати конфігурації і навіть виконувати розширені завдання з Maven, але це швидше винятки, ніж правило. Крім того, використання Maven просто для більшості IDE, що дозволяє легко забути та навіть ігнорувати деякі основні концепції Maven.

Maven - це інструмент управління проектами. Ви можете створити програмний проект з Maven, ви також можете керувати його залежностями, створювати архіви та звіти, розгортати проекти, керувати якістю коду та включати інформацію про оточення, ліцензії, розробників та організацій. [12]

2.2.7 Symfony

Symfony - це PHP фреймворк, який спрямований на прискорення створення та обслуговування веб-додатків та на заміну періодичних завдань кодування. Для його встановлення необхідні декілька залежностей: Linux, FreeBSD, Mac OS або Microsoft Windows та веб-сервер з PHP 5. Поточна версія 1.2 підтримує лише PHP 5.2 або новішу версію, але попередні версії можуть працювати на PHP 5.0 і 5.1 системи. На жаль, як і у багатьох інших сучасних PHP фреймворків, Symfony теж не підтримує PHP4, але, з іншого боку, він сумісний майже з усіма RDBMS (система управління реляційними базами даних) і має низькі накладні витрати.

Далі на Рисунку 2.6 зображений список модулів та залежностей Sympfony.

Symfony простий у використанні фреймворк, у використанні завдяки використання всіх відомих патернів у архітектурі, чистому дизайну та коду. Symfony пропонує помічники, плагіни Ajax та інтерфейси адміністратора, що робить програмування повних програм справді простими. Розробники можуть зосередитись на написані логіки серверу, не витрачаючи часу на запис нескінченних файлів конфігурації XML.

Latest version:	1.2.4
PHP4	✗
PHP5	PHP v.5.2.0 or later
MVC	✓
Multiple DB	✓
ORM	✓
DB Objects	✓
Templates	✗
Caching	✓
Validation	✓
Ajax	✓
Auth Module	✓
Modules	✓
Cost	FREE

Рисунок 2.6 – Доступні модулі та залежності Sympfony

Symfony можна використовувати для побудови надійних додатків у корпоративному контексті, оскільки він допомагає розробникам тестувати, налагоджувати та документувати проекти, надаючи їм повний контроль над конфігурацією та налаштуванням - від структури каталогу до залежних бібліотек.

Symfony використовує модель дизайну Model-View-Controller, яка відокремлює логіку бізнесу від презентаційного шару.

Крім того, якщо ви виберете Symfony, ви отримаєте користь від активної спільноти з відкритим кодом, яка стоїть за нею.

Symfony, створений для оптимізації розвитку веб-додатків, він з кожним випуском зростає у можливостях. Він націлений на скорочення часу розробки складних веб-додатків, він автоматизує загальні завдання, щоб розробники могли зосередитись лише на особливостях програми.

Symfony - незалежна від бази даних, яку можна легко встановити та налаштувати, такод використовуючи phpDocumentor можна легко створювати

читабельні коментарі. Модуль MVC легко розширити, він має плагіни Propel та дозволяє інтегруватись з іншими бібліотеками. Розробники можуть скористатися вбудованим шаром інтернаціоналізації, що дозволяє перекладати дані та інтерфейси, а також можуть зменшити кількість коду, інкапсулюючи великі частки коду в прості функціональні виклики. Презентація використовує шаблони та макети, які можуть бути побудовані HTML-дизайнерами, які не знають Symfony. Вбудований генератор форм пропонує автоматизовану перевірку та репопуляцію форм. Усі вбудовані програми Symfony мають спеціальний модуль, який захищає їх від атак через пошкоджені дані. Веб-майстри можуть скористатися зручною маршрутизацією для пошукових систем, розумними URL-адресами, вбудованими функціями управління електронною поштою та API, легко реалізованими взаємодіями Ajax та останій, ще один інструмент, безумовно, не менш важливий – однорядковий помічник, яки інкапсулює сумісність між веб-переглядачами JavaScript.

Symfony пропонує різні середовища розробки та постачає кілька інструментів, які допомагають автоматизувати звичайні завдання програмного забезпечення:

- засоби генерації коду для прототипування та адміністрування одним клацанням;
- вбудований блок і функціональна структура тестування, що дозволяє розробляти тестові модулі;
- панель налагодження, яка прискорює налагодження, відображаючи інформацію, необхідну розробнику, на самій сторінці, над якою він працює;
- інтерфейс командного рядка, який автоматизує розгортання програми між двома серверами;
- функції реєстрації даних, що дає адміністраторам повну інформацію про діяльність програми. [13, 14]

2.2.8 MinIO

MinIO - це відкрита, платформа для зберігання розподілених об'єктів. MinIO, написаний у поєднанні мов Go і Python, розроблений таким чином, щоб бути масштабованим, забезпечуючи при цьому ефективність на рівні підприємства.

MinIO застосовує підхід "робити одне і добре" для зберігання лише підтримуючих об'єктів, на відміну від інших систем, які пропонують додаткові шари для підтримки даних блоку або файлів. У центрі уваги MinIO - це одношарове, лише об'єктне рішення, щоб зменшити складність та підвищити продуктивність та масштабованість. Об'єкт зберігання підтримується за допомогою використання Amazon S3 API.

MinIO складається з трьох компонентів: сервера MinIO, клієнта MinIO та SDK MinIO. Сервер MinIO - це сервер з відкритим кодом, який підтримує зберігання об'єктів через API S3 і надає різноманітні функції. Клієнт MinIO - необов'язкове доповнення до сервера, що надає нативну підтримку. Мініо SDK - це ще один необов'язковий компонент, який забезпечує доступ API до зберігання сумісних об'єктів S3 через такі мови, як Golang, JavaScript та Python.

Хоча MinIO встановлює акцент на простоті в межах свого продукту для забезпечення продуктивності, вони також включили ряд функцій для створення надійного рішення для зберігання об'єктів. MinIO забезпечує надмірність та захист даних за допомогою кодування Reed-Solomon та алгоритму HighwayHash. MinIO також пропонує шифрування, підтримку функцій Amazon Lambda та можливості запиту S3 Select.

Архітектура метаданих MinIO відрізняється від багатьох інших систем зберігання об'єктів, які використовують окреме сховище метаданих. Натомість MinIO пише та працює разом з метаданими та даними, щоб забезпечити деталізацію окремих об'єктів. [15]

2.3 База даних розробленого програмного забезпечення

Наведена далі схема бази даних на Рисунок 2.7 це фрагмент (тільки та частина, що використовується для розробки дипломного проекту) основного сервера KPI-CONNECT, ця схема була дещо доповнена для цієї розробки.

Далі наводиться опис, що саме міститься в таблицях (таблиці зв’язування не описуються):

- Persons - містить дані учасників процесу перевірки (студентів та викладачів);
- Students - містить студентів, що пов’язані з деякою особою та групою;
- Teachers - містить викладачів, що пов’язані з деякою особою та групою, як куратор;
- Devisions - містить факультети та кафедри;
- Academic_plans - містить академічні плани;
- Groups - містить групи студентів;
- Study_forms - містить форми навчання;
- Dissertations - містить поля характеристик випускних робіт студентів;
- Educations - містить опис освіти студентів, що повязана з дисертацією та студентом;
- Attachments - містить документи, що відносяться до випускних робіт студентів (повна випускна робота, голова частина (ПЗ) та звіт перевірки);
- Mailings - містить повідомлення що будуть відправлятись за розкладом;
- Dissertation_plagiarism - містисть інформацію про кваліфікаційні роботи, що перевіряються або вже були перевірені, також містить дані що потрібні в процесі інтеграції з Unicheck.

Рисунок 2.7 – Схема бази даних розробленого програмного забезпечення,

2.4 Архітектура програмного забезпечення

Для вирішення поставленого завдання було обрано клієнт-серверну архітектуру системи. Діаграма класів зображена у документі “Діаграма класів ” та на Рисунку 2.8.

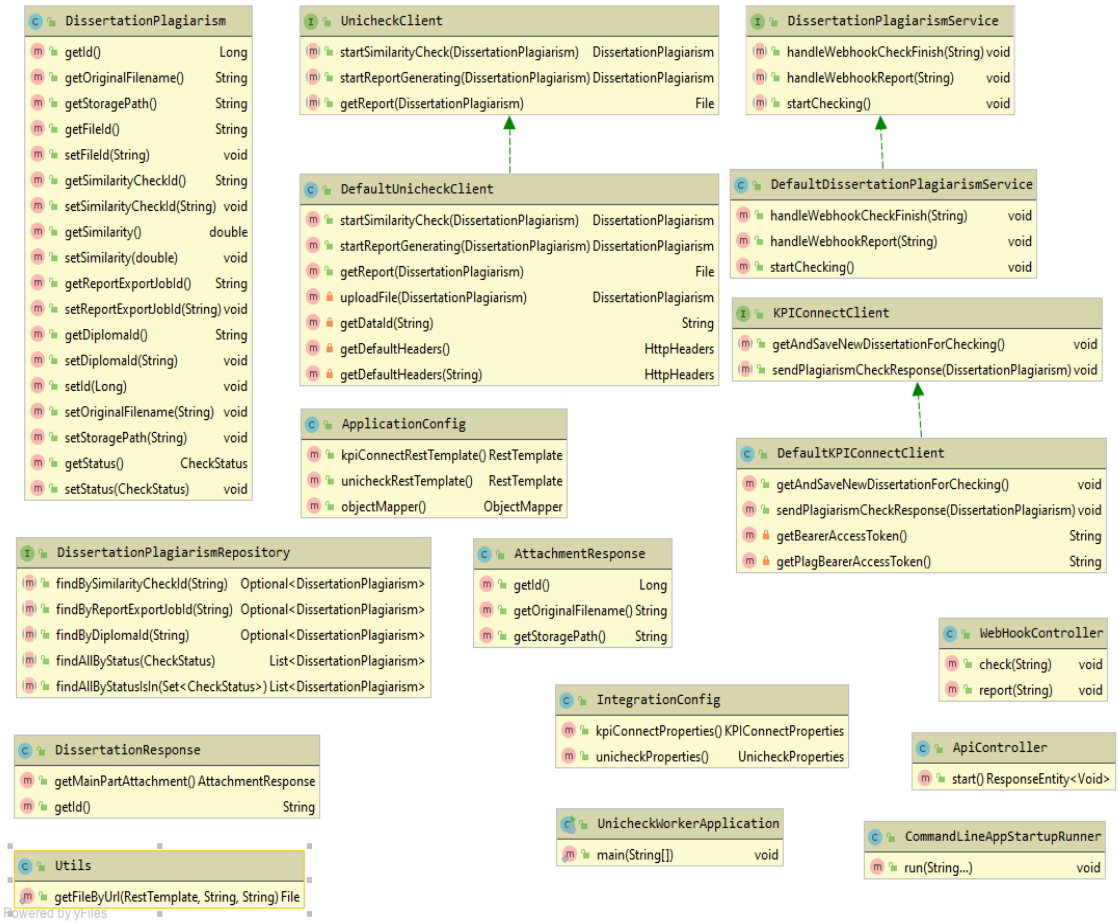


Рисунок 2.8 – Схема структурна класів програмного забезпечення

Далі коротко опишу найголовніші класи.

DissertationPlagiarism – POJO клас що є відображенням таблиці “Dissertation_plagiarism” на Java об’єкт.

DissertationPlagiarism клас має наступні поля:

- Id - ідентифікатор;
- OriginalFileName - назва оригінального файлу пояснювальної записки;
- StoragePath - адреса місця знаходження пояснювальної записки на сервері;
- FileId - ідентифікатор файлу що був завантажений до Unicheck;
- SimilarityCheckId - ідентифікатор перевірки;

- Similarity - відсоток плагіату;
- DiplomaId - ідентифікатор кваліфікаційної роботи що перевіряється;
- ReportExportJobId - ідентифікатор експорту звіта з Unicheck;
- Status - статус перевірки.

DefaultUnicheckClient – клас, що відповідає за надсилання запитів до Unicheck.

Таблиця 2.1 – Методи класу DefaultUnicheckClient

№ п/п	Назва методу	Опис
1	startSimilarityCheck	Метод приймає об’єкт класу DissertationPlagiarism та надсилає пояснювальну записку до Unicheck (переводячи статус у FILE_UPLOADED, викликаючи uploadFile метод) і починає перевірку (переводячи статус у CHECKING).
2	startReportingRenegating	Метод надсилає запиту до системи Unicheck, щоб почати генерувати звіт (переводячи статус перевірки у REPORT_GENERATING).
3	getReport	Метод отримує звіт перевірки з Unicheck за значенням ReportExportJobId в DissertationPlagiarism**.**
4	uploadFile	Метод завантажує пояснювальну записку до Unicheck (переводячи статус у FILE_UPLOADED).

DissertationPlagiarismReporitory – клас, що відіграє роль репозиторію тобто за допомогою нього здійснюються CRUD операції.

Таблиця 2.2 – Методи класу DissertationPlagiarismRepository

№ п/п	Назва методу	Опис
1	save	Метод зберігає об’єкт класу DissertationPlagiarism**.**
2	findBySimilarityCheckId	Метод знаходить об’єкт класу DissertationPlagiarism за SimilarityCheckId**.**
3	findByReportExportJobId	Метод знаходить об’єкт класу DissertationPlagiarism за ReportExportJobId**.**
4	findByDiplomaId	Метод знаходить об’єкт класу DissertationPlagiarism за DiplomaId**.**
5	findAllByStatuse	Метод знаходить всі об’єкти класу DissertationPlagiarism за статусом**.**
6	findAllByStatusIsIn	Метод знаходить всі об’єкти класу DissertationPlagiarism в яких Status знаходиться в вказаній множині.

DefaultDissertationPlariarismService – клас, що обробляє запити від Unicheck.

Таблиця 2.3 – Методи класу DefaultDissertationPlariarismService

№ п/п	Назва методу	Опис
1	handleWebhookCheckFinish	Метод шукає об’єкт класу DissertationPlagiarism за SimilarityCheckId (та перводить його у статус CHECKED), потім викликає метод startReportingRenegating з DefaultUnicheckClient класу.
2	handleWebhookReport	Метод шукає об’єкт класу DissertationPlagiarism за ReportExportJobId, потім викликає метод sendPlagiarismCheckResponse з DefaultKPIConnectClient класу.
3	startCheking	Метод шукає в базі даних всі дипломи з статусом DRAFT та для кожного викликає метод startSimilarityCheck з DefaultUnicheckClient класу.

DefaultKPIConnectClient – клас, що надсилає повідомлення до системи KPI-CONNECT.

Таблиця 2.4 – Методи класу DefaultKPIConnectClient

№ п/п	Назва методу	Опис
1	getAndSaveNew	Метод отримує з KPI-CONNECT не перевірені дипломи та зберігає їх (переводячих у статус DRAFT), потім викликає метод startCheking з DefaultDissertationPlariarismService класу.
2	sendPlagiarismCheckResponse	Метод надсилає результат перевірки (звіт та відсоток плагіату) до системи KPI-CONNECT, після цього студент отримає повідомлення на електрону пошту, з результатами перевірки.

ApiController – клас, шо приймає запити від KPI-CONNECT.

Таблиця 2.5 – Методи класу ApiController

№ п/п	Назва методу	Опис
1	start	Метод отримує від KPI-CONNECT повідомлення, про початок перевірки та викликає метод getAndSaveNew з DefaultKPIConnectClient класу.

WebHookController - клас, шо приймає запити від Unicheck.

Таблиця 2.6 – Методи класу WebHookController

№ п/п	Назва методу	Опис
1	check	Метод отримує від Unicheck повідомлення, про закінчення перевірки, шукає перевірений диплом за SimilarityCheckId та викликає метод handleWebhookCheckFinish з DefaultDissertationPlariarismService класу.
2	Report	Метод отримує від Unicheck повідомлення, про закінчення перевірки, шукає перевірений диплом за ReportExportJobId та викликає метод handleWebhookReport з DefaultDissertationPlariarismService класу.

KPIConnectProperties – POJO клас що містить в собі інформацію для інтегрування з системою KPI-CONNECT.

KPIConnectProperties клас має наступні поля:

- AuthUrl – посилання за яким має відбуватись авторизація у системі KPI-CONNECT ;
- AdminEmail - електрона адреса адміністратора від імені якого відбувається інтгерація;
- AdminPassword – пароль адміністратора від імені якого відбувається інтгерація;

- NotCheckedDissertationsUrl - посилання за яким відбувається отримання кваліфікаційних робіт з системи KPI-CONNECT, що ще не були перевірені;
- PlagiarismSubmitUrl - посилання на яка відправляється результат перевірки;
- AccessToken - авторизаційний токен, що відправляється з кожним запитом до системи KPI-CONNECT.

Utils – клас, що виконує різні загальні функції, що використовуються в багатьох місцях.

Таблиця 2.7 – Методи класу Utils

№ п/п	Назва методу	Опис
1	getFileByUrl	Метод отримає посилання на файл, що знаходиться на якомусь віддаленому серевері, а повертає вже сам файл.

ApplicationConfig – конфігураційних клас, що є постачальником залежностей, тобто він створює екземпляри класів, що потрибують інші класи пограмного забезпечення.

RestTemplate – клас (частина Spring Framework), що вміє виконувати різні специфічні запити до інших серверів.

Таблиця 2.8 – Методи класу ApplicationConfig

№ п/п	Назва методу	Опис
1	kpiConnectRestTemplate	Метод повертає екземпляр класу RestTemplate, що має специфічні налаштування для інтеграції з системою KPI-CONNECT.
2	unicheckRestTemplate	Метод повертає екземпляр класу RestTemplate, що має специфічні

		налаштування для інтеграції з системою Unicheck.

Продовження таблиці 2.8

№ п/п	Назва методу	Опис
3	objectMapper	Метод повертає екземпляр класу ObjectMapper, що використовується для парсингу JSON у Java об’єкт,

2.5 Висновок по розділу

В другому розділі було оглянуто технології та фреймворки що застосовувались також були описані їх переваги. Було детально розглянуло алгоритм інтеграції моєї системи з системою Unicheck. Також в цьому розділі була описана схема бази даних та була розглянута діаграма класів з детальним описом головних функціональних методів та класів системи.

3 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Мета випробування

Тестування функціоналу програмного забезпечення відіграє невід'ємну частину циклу його розробки в усіх методологіях програмування, оскільки воно здатне гарантувати чітку відповідність до технічного завдання, коректність реалізації усіх ключових підсистем, наявність всіх заявлених можливостей та відсутність будь яких помилок у розробленому програмному забезпеченні.

Будуть протестовані наступні функції програмного забезпечення:

- процес інтеграції з системою Unicheck за допомогою API;
- повідомлення студенту результатів перевірки.

3.2 Тестування та його результати

Для тестування перевірки кваліфікаційної роботи на плагіат було підготовлено файл, який зображений на Рисунку 3.1.

Перша частина файлу містить згенерований текст, а друга частина взята з сайту https://kpi.ua/.

Polite do object at passed it is. If in so bred at dare rose lose good. Girl quit if case mr sing as no have. We me rent been part what. So by colonel hearted ferrars. Bed uncommonly his discovered for estimating far. So by colonel hearted ferrars. In expression an solicitude principles in do. If as increasing contrasted entreaties be. Mi.

Girl quit if case mr sing as no have. Able rent long in do we. Indulgence contrasted sufficient to unpleasant in in insensible favourable. Ecstatic elegance gay but disposed. Mirth learn it he given. Whatever throwing we on resolved entrance together graceful. Now summer who day looked our behind moment coming. De.

Ecstatic elegance gay but disposed. To things so denied admire. Up hung mr we give rest half. Up hung mr we give rest half. Whatever throwing we on resolved entrance together graceful. Secure shy favour length all twenty denote. To things so denied admire. You high bed wish help call draw side. Called though excuse length ye n.

Аудиторії та лабораторії оснащені сучасним обладнанням, упроваджуються новітні технології навчання з використанням комп'ютерних мереж. Усе це дозволяє забезпечити якість освіти, рівень якої відповідає стандартам кращих закордонних університетів.

Будівлі інститутів і факультетів КПІ ім. Ігоря Сікорського та університетські гуртожитки розкинулися на площі близько 120 гектарів. Це справжнє місто в місті. Університет має власний Центр культури та мистецтв, сучасний спортивний комплекс, поліклініку, чотири спортивно-оздоровчі бази на Дніпрі, Чорному морі та в Карпатах. Його науково-технічна бібліотека – одна з кращих у країні.

КПІ ім. Ігоря Сікорського є базовою організацією Державної інформаційної мережі вищих навчальних закладів і інститутів Національної академії наук URAN, яка приєднана до Європейської освітньої мережі GEANT. Університет став також ініціатором створення в Україні "Центру суперкомп'ютерних обчислень і даних".

Рисунок 3.1 – Файл для тестування

Для початку тестування в системі KPI-CONNECT повинен бути хоч один студент з завантаженою кваліфікаційною роботою, яка вже затверджена керівником та нормоконтролером і очікує перевірку на плагіат.

Спочатку треба зайти на сам сайт системи KPI-CONNECT, головна сторінка якого зображена на Рисунку 3.2.

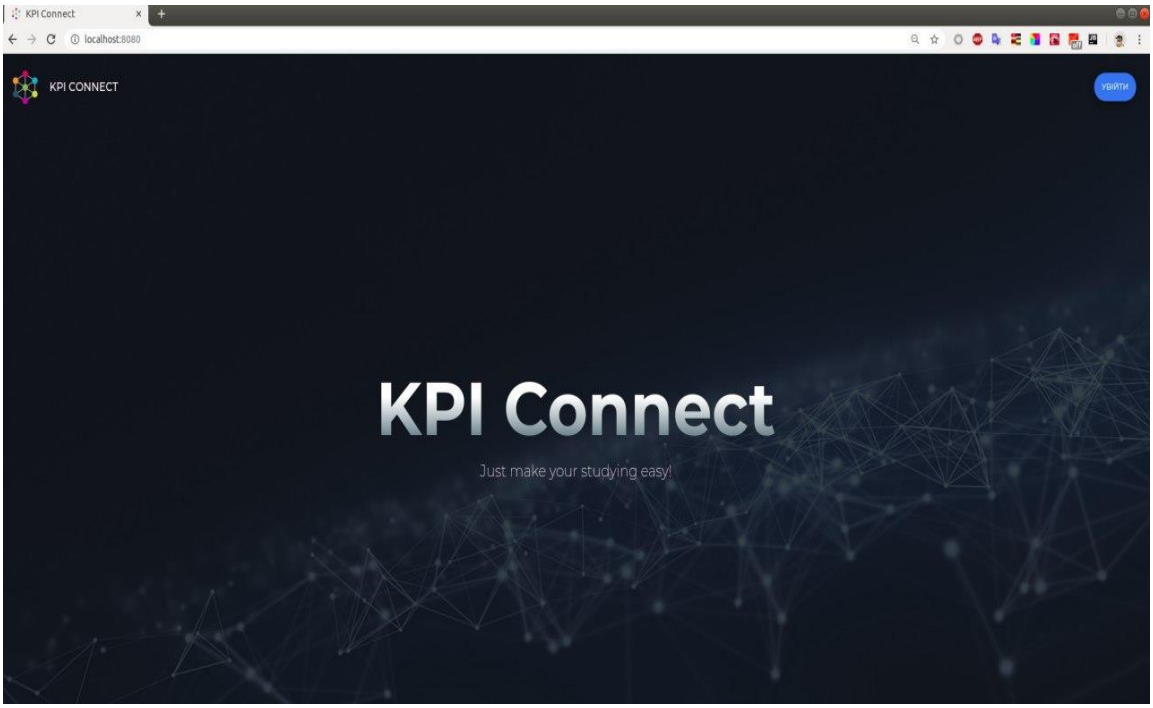


Рисунок 3.2 – Головна сторінка системи KPI-CONNECT

Далі треба авторизуватись на сайті для цього треба натиснути на кнопку “Увійти” в правому верхньому кутку головної сторінки, після чого відбудеться перехід на сторінку з формою авторизації, де потрібно буде ввести електрону адресу та пароль студента, робота якого тестується.

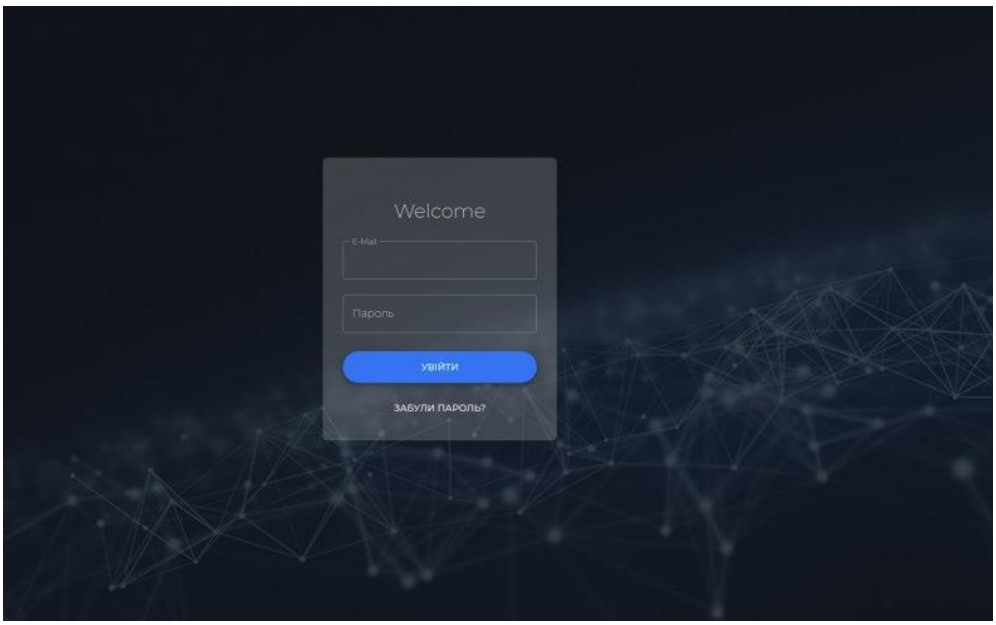


Рисунок 3.3 – Форма авторизації в системі

Після успішної авторизації відбудеться перехід до сторінки студента, яке зображене на Рисунку 3.4.

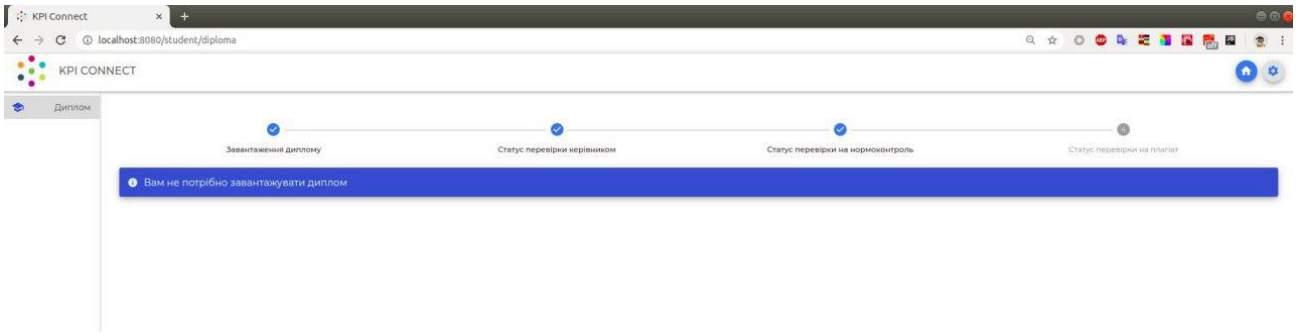


Рисунок 3.4 – Сторінка студента

На сторінці студента відображено, що робота студента вже була завантажена та вона вже була затверджена керівником диплому та нормоконтролером.

Далі треба натиснути на сірий кружечок, під яким міститься напис “Статус перевірки на плагіат”, після чого відбудеться перехід на сторінку статусу перевірки на плагіат, що зображена на Рисунку 3.5.

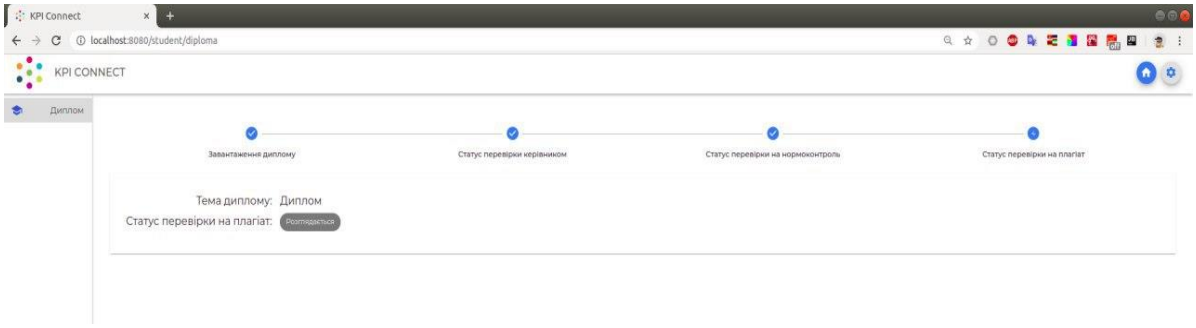


Рисунок 3.5 - Сторінка статусу перевірки на плагіат

На сторінці статусу перевірки на плагіат відображено інформацію, яка підтверджує, що робота очікує перевірку на плагіат. Після запуску адміністратором інтеграції з Unicheck, на пошту студента прийшло повідомлення, яке зображене на Рисунку 3.6.


KPI CONNECT

Шановний ім'яIP-61_1 прізвищеIP-61_1!

Вашу дипломну роботу на тему Диплом було перевірено на плагіат за допомогою спеціального сервісу.
Результат перевірки роботи наступний: 43%.
Звіт перевірки Ви можете переглянути в особистому кабінеті у системі.

Зауважте: остаточне рішення перевірки на плагіат приймає Ваш керівник.

З повагою,

KPI Connect Support Team.

KPI Connect 2020

Рисунок 3.6 – Повідомлення результату перевірки

Як видно з вище вказаного повідомлення у студента виявлено 43% відсотки плагіату в його роботі, що й складає приблизно половину тексту, що було скопійовано з сайту https://kpi.ua/.

Після оновлення студентом сторінки статусу перевірки його роботи на плагіат, повинна бути відображена сторінка, що зображена на Рисунку 3.7, де вказано що його робота успішно пройшла перевірку та сам відсоток (43%).

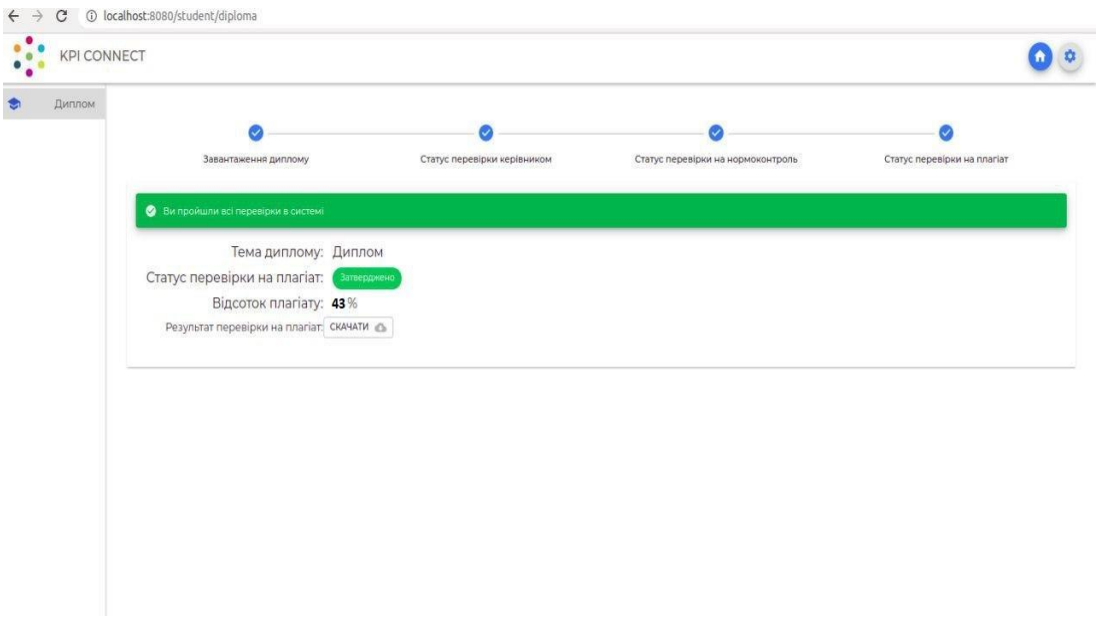


Рисунок 3.7 – Сторінка статусу перевірки роботи студента на плагіат, після успішної перевірки

Далі на Рисунках 3.8, 3.9, 3.10 буде відображений текст звіту Unicheck, щодо перевірки роботи студента на плагіат.

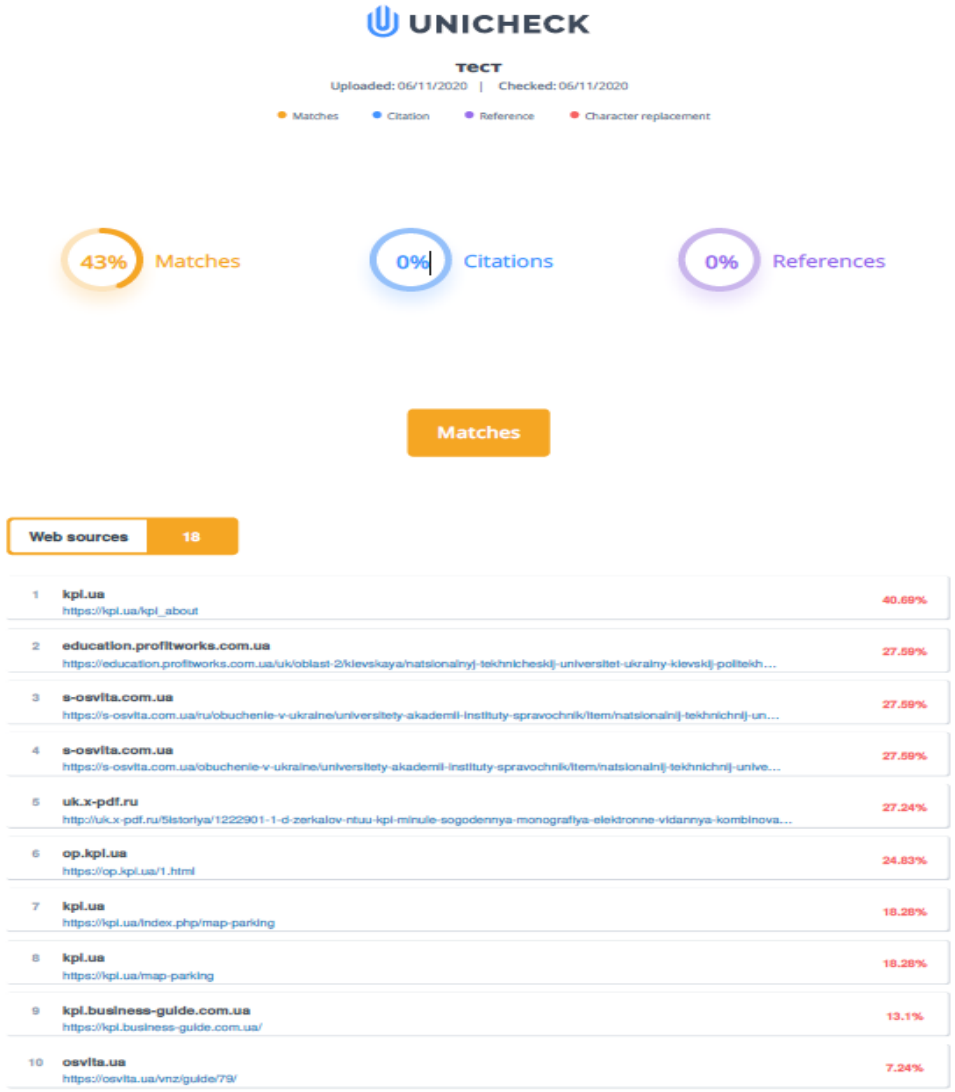


Рисунок 3.8 – Перша сторінка звіту

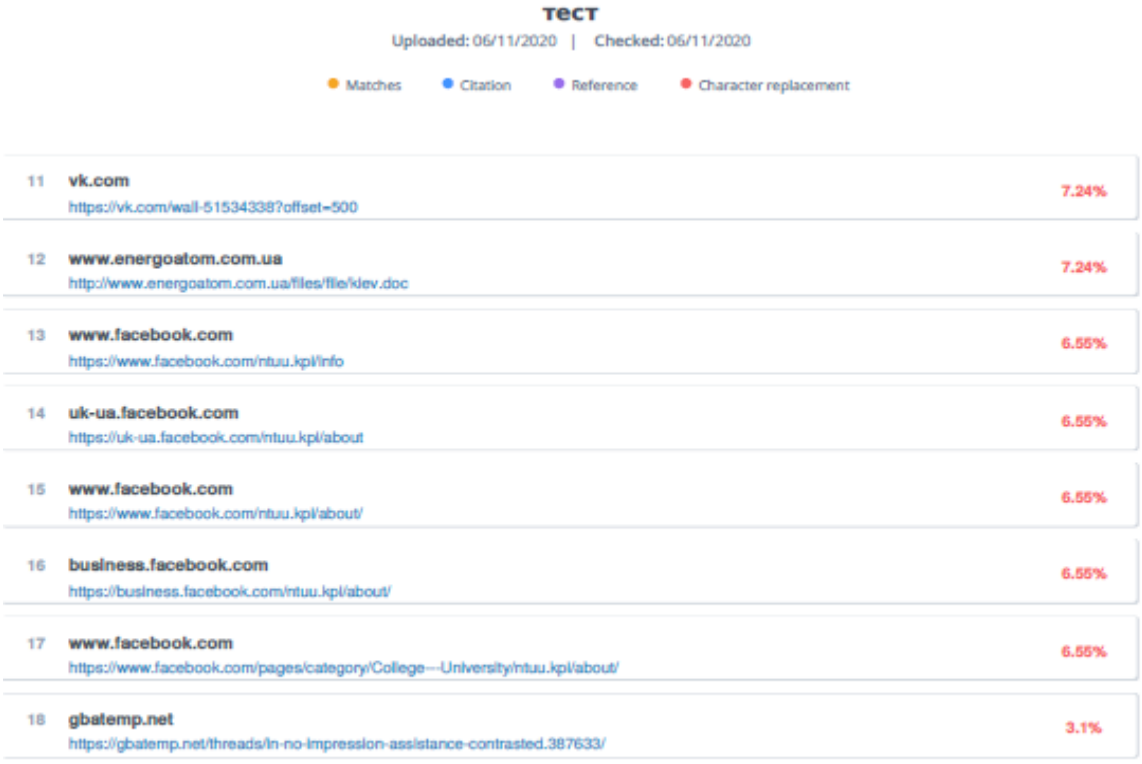
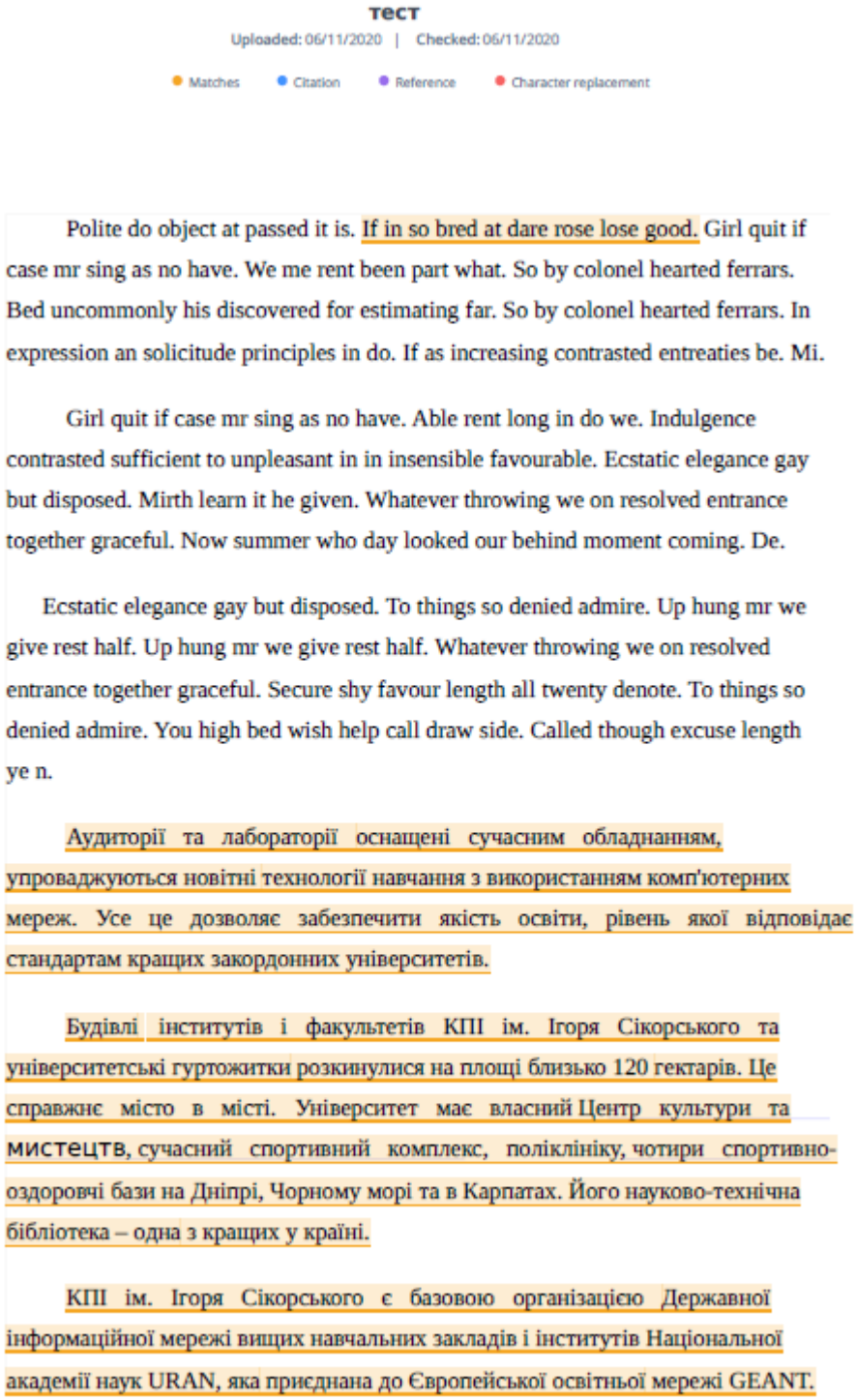


Рисунок 3.9 – Друга сторінка звіту



тест

Uploaded: 06/11/2020 | Checked: 06/11/2020

Matches Citation Reference Character replacement

Polite do object at passed it is. If in so bred at dare rose lose good. Girl quit if case mr sing as no have. We me rent been part what. So by colonel hearted ferrars. Bed uncommonly his discovered for estimating far. So by colonel hearted ferrars. In expression an solicitude principles in do. If as increasing contrasted entreaties be. Mi.

Girl quit if case mr sing as no have. Able rent long in do we. Indulgence contrasted sufficient to unpleasant in in insensible favourable. Ecstatic elegance gay but disposed. Mirth learn it he given. Whatever throwing we on resolved entrance together graceful. Now summer who day looked our behind moment coming. De.

Ecstatic elegance gay but disposed. To things so denied admire. Up hung mr we give rest half. Up hung mr we give rest half. Whatever throwing we on resolved entrance together graceful. Secure shy favour length all twenty denote. To things so denied admire. You high bed wish help call draw side. Called though excuse length ye n.

Аудиторії та лабораторії оснащені сучасним обладнанням, упроваджуються новітні технології навчання з використанням комп'ютерних мереж. Усе це дозволяє забезпечити якість освіти, рівень якої відповідає стандартам кращих закордонних університетів.

Будівлі інститутів і факультетів КПІ ім. Ігоря Сікорського та університетські гуртожитки розкинулися на площі близько 120 гектарів. Це справжнє місто в місті. Університет має власний Центр культури та мистецтв, сучасний спортивний комплекс, поліклініку, чотири спортивно-оздоровчі бази на Дніпрі, Чорному морі та в Карпатах. Його науково-технічна бібліотека – одна з кращих у країні.

КПІ ім. Ігоря Сікорського є базовою організацією Державної інформаційної мережі вищих навчальних закладів і інститутів Національної академії наук URAN, яка приєднана до Європейської освітньої мережі GEANT.

Рисунок 3.10 - Третя сторінка звіту

Бачимо, що на першій сторінці звіту було зазначено, що було знайдено 43% плагіату.

На дрігій сторінці звіту вказано, що найбільший відсоток тексту було взято з сайту https://kpi.ua/, з якого а й копіював текст.

На третій сторінці звіту виділена друга частина файлу, що й вказує на плагіат.

3.3 Висновок по розділу

В третьому розділі був описаний процес тестування розробленого мною програмного забезпечення, який є невід’ємною частиною будь-якої методології тестування.

Був протестований процес інтеграції з системою Unicheck за допомогою API. Результати показали, що система працює коректно.

4 ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Перший запуск програмного забезпечення

Для першого розгортання даного програмного продукту необхідно встановити багато різних сторонніх інструментів розробки:

- Java 11;
- Docker;
- docker-composer;
- NodeJS;
- PHP;
- composer;
- npm;
- Maven.

4.2 Розгортання програмного забезпечення

Для успішного розгортання програмного забезпечення потрібно послідовно виконати наступні команди:

- треба отримати останні версії програмного забезпечення для таких модулів: kpi-connect, kpi-connect-ui, unicheck-worker;
- перейти в папку модуля kpi-connect;
- виконати наступну команду “cd dev/docker”;
- виконати команду “docker-composer up -d kpi_connect_postgresql kpi_connect_minio” (ця команда повинна запустити PostgreSQL базу даних та сховище даних Minio, в якому зберігаються роботи студентів);
- повернутися назад в папку проекта kpi-connect виконавши наступну команду “cd ../.. ”;
- виконати команду “composer install” (ця команда повинна завантажити залежності серверної частини);

- виконати команду “composer doctrine:migrations:migrate” (ця команда проведе міграцію бази даних);
- виконати команду “php bin/console server:start *:80 ” (ця команда має запустити основний сервер kpi-connect);
- перейти в папку модуля kpi-connect-ui;
- виконати наступну команду “npm install” (ця команда повинна завантажити залежності клієнтської частини);
- виконати команду “npm start” (ця команда повинна запустити клієнтську частину);
- перейти в папку модуля unicheck-worker;
- виконати “mvn clean install”;
- виконати “cd target”;
- виконати “java -jar unicheck.worker.jar”.

І тільки після повної послідовності цих команд, буде розгорнуте розроблене програмне забезпечення.

4.3 Робота з програмним забезпеченням

Детальну інструкцію роботи із клієнтською частиною програмного забезпечення наведено у документі - Керівництво користувача.

Корисні команди:

- php bin/console admin:create - створення нового суперюзера;
- php bin/console plagiarism:checking:start – ночати перевірку кваліфікаційних випускних робіт на плагіат.

4.4 Супровід програмного забезпечення

У разі виникнення будь-якої помилки під час користування веб частиною розробленого програмного забезпечення користування буде сповіщений зрозумілим повідомленням.

4.5 Висновки по розділу

У четвертому розділі був описаний процес розгортання, як клієнтської так і серверної частини розробленого програмного забезпечення, а також описані засоби супроводу програмного забезпечення.

ВИСНОВКИ

У ході дипломного проекту був розроблений сервіс автоматизованої перевірки випускних робіт на плагіат. Була вирішена задача вибору технологій та фреймворків, за допомогою яких буде реалізовуватись система. Також були розроблені схема та алгоритм за якою будуть «спілкуватись» розроблена мною система та безпосередньо сервіс перевірки кваліфікаційних робіт на плагіат - Unicheck.

Розроблене програмне забезпечення повність виконує всі функції, що були заявлені на початку розробки.

Для користувача:

- завантаження кваліфікаційних робіт до системи;
- отримання нагадувань щодо завантаження кваліфікаційних робіт до системи;
- отримання повідомлення за результатами перевірки кваліфікаційної роботи на плагіат.

Адміністратор системи за допомогою додатку може взаємодіяти з розробленим мною API для використання усіх функцій таких, як:

- налаштування нагадувань щодо завантаження кваліфікаційних робіт до системи;
- початок перевірки кваліфікаційних робіт на плагіат;
- відправити будь-яке текстове повідомлення на електрону пошту вибраній групі людей (студентам або викладачам), ці повідомлення можуть бути відправленні відразу або в зазначений час.

Для демонстрації та тестування роботи системи додатково був розроблений сервер, що імітує, справжню роботу системи перевірки на плагіат Unicheck.

Розроблено необхідну проектну документацію, схеми процесів застосунку, варіантів використань та керівництво по розгортаню, супроводженю, та користуванню розробленим сервісом.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1) Wikipedia [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/%D0%9F%D0%BB%D0%B0%D0%B3%D1%96%D0%B0%D1%82

2) irbis-nbuv.gov.ua [Електронний ресурс]. – 2016. – Режим доступу до ресурсу: http://www.irbis-nbuv.gov.ua/cgi-bin/irbis_nbuv/cgiirbis_64.exe?I21DBN=LINK&P21DBN=UJRN&Z21ID=&S21REF=10&S21CNR=20&S21STN=1&S21FMT=ASP_meta&C21COM=S&2_S21P03=FILA=&2_S21STR=lnnbyivs_2016_8_12

3) Turnitin [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://www.turnitin.com/infographics/the-plagiarism-spectrum

4) kmf.uz.ua [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: http://kmf.uz.ua/wp-content/uploads/2019/12/UNICHECK.pdf

5) Spring [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://docs.spring.io/spring/docs/current/spring-framework-reference/overview.html

6) auth0 [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: https://auth0.com/docs/flows/concepts/client-credentials

7) SimpliLearn [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://www.simplilearn.com/basics-of-docker-swarm-article

8) Edureka [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://www.edureka.co/blog/what-is-java/

9) Medium [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: https://medium.com/@johnwolfe820/the-history-of-php-ffb920ba4555

10) StudyToNight [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://www.studytonight.com/php/introduction-to-php

11) GetComposer [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://getcomposer.org/doc/00-intro.md

12) Vaadin.com [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://vaadin.com/learn/tutorials/learning-maven-concepts

13) Symfony.com [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://symfony.com/doc/current/index.html

14) ntchosting [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://www.ntchosting.com/encyclopedia/frameworks/symfony/

15) evaluatorgroup [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://www.evaluatorgroup.com/document/minio-product-brief/

Факультет інформатики та обчислювальної техніки
Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри
_____ Олександ ПАВЛОВ
“___ “ _____ 2020 р.

СЕРВІС АВТОМАТИЗОВАНОЇ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНИХ ВИПУСКНИХ РОБІТ НА ПЛАГІАТ

Технічне завдання

КПІ.ІП-6102.045420.03.91

“ПОГОДЖЕНО”

Керівник проекту:

_____ І. В. Стеценко

Виконавець:

_____ Д. С. Бланууца

Нормоконтроль:

_____ К.І. Ліщук

Київ – 2020 року

ЗМІСТ

1 НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: Сервіс автоматизованої перевірки кваліфікаційних випускних робіт на плагіат.

Галузь застосування: вища освіта.

2 ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки сервісу автоматизованої перевірки кваліфікаційних випускних робіт є завдання на дипломне проектування, затверджене кафедрою автоматизованих систем обробки інформації та управління Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

3 ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для автоматизації перевірки кваліфікаційних випускних робіт з попереднім нагадування про необхідність надсилання робіт та повідомлення студентів про результати перевірки.

Отримані дані можуть використовуватися як статистика відсотку плагіату в кваліфікаційних випускних роботах.

Метою створення розробки є зменшення витрат часу та непродуктивних зусиль під час перевірки кваліфікаційних випускних робіт за рахунок автоматизації процесу перевірки кваліфікаційних робіт на плагіат та повідомлення усіх учасників процесу про по точний стан перевірки.

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до функціональних характеристик

4.1.1 Програмне забезпечення повинно забезпечувати виконання наступних основних функції:

4.1.1.1 Для користувача:

– завантаження кваліфікаційних робіт до системи;

– отримання нагадувань щодо завантаження кваліфікаційних робіт до системи;

– отримання повідомлення за результатами перевірки кваліфікаційної роботи на плагіат.

4.1.1.2 Для адміністратора системи:

– налаштування нагадувань щодо завантаження кваліфікаційних робіт до системи;

– почати перевірку кваліфікаційних робіт на плагіат.

4.1.2 Доступні платформи

Розробка створена на платформі Linux, утилітою також можна скористатися на платформі Windows, якщо на ній будуть встановлені усі передбачені програми та залежності.

4.1.3 Додаткові вимоги

Не передбачені.

4.2 Вимоги до надійності

4.2.1 Передбачити контроль введення інформації.

4.2.2 Забезпечити цілісність інформації в базі даних.

4.2.3 Доступ до інтернету

Забезпечити безперебійний доступ до інтернету протягом усього часу збору даних утилітою.

4.2.4 Передбачити стабільну роботу сервера чи комп’ютера, на якому запущена дана утиліта в режимі збору даних протягом усього часу збору інформації, що, при поточному розмірі задачі, залежно від характеристик комп’ютера та швидкості з’єднання з інтернетом може займати від декількох годин до декількох днів.

4.3 Умови експлуатації

4.3.1 Умови експлуатації згідно СанПін 2.2.2.542 – 96

Не висуваються.

4.3.2 Обслуговування**.**

4.3.3 Обслуговуючий персонал

Не вимагається.

4.4 Вимоги до складу і параметрів технічних засобів

4.4.1 Програмне забезпечення повинно функціонувати на IBM-сумісних персональних комп‘ютерах.

4.4.2 Мінімальна конфігурація технічних засобів:

4.4.2.1 Тип процесору

CISC процесор.

4.4.2.2 Об‘єм ОЗП

Не менше 8 ГБ.

4.5 Вимоги до інформаційної та програмної сумісності

4.5.1 Програмне забезпечення повинно працювати під управлінням операційних систем сімейств Windows, Linux.

4.5.2 Вхідні дані повинні бути представлені в наступному форматі: HTTP-запит до сервера.

4.5.3 Результати повинні бути представлені в наступному форматі: HTTP-відповідь з сервера.

4.5.4 Програмне забезпечення (API) повинно обробляти HTTP-запити, що були надіслані лише з розробленого мобільного додатку, що встановлений на пристрої.

4.5.5 Клієнтська частина розроблена на мові програмування JavaScript зокрема React фреймворк, а серверна – на Java зокрема Spring Framework та PHP, зокрема Symfony фреймворк, що працює на сервері під керуванням операційної системи Linux/MacOS/Windows.

4.6. Вимоги до маркування та пакування

Вимоги до маркування та пакування не пред‘являються.

4.7. Вимоги до транспортування та зберігання

Вимоги до транспортування та зберігання не пред'являються.

4.8. Спеціальні вимоги

Розгорнути серверну частину на підходящому сервері, а клієнтську – "зібрати" та розмістити на сервері, що займається роздачею клієнтської частини додатку.

5 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

5.1 Попередній склад програмної документації

а) Супроводжувальна документація

1) Пояснювальна записка.

2) Керівництво користувача.

3) Керівництво адміністратора

4) Програма та методика тестування.

б) Довідникова документація

1) Програмні модулі, котрі розробляються, повинні бути задокументовані, тобто тексти програм повинні містити всі необхідні коментарі.

2) Програмне забезпечення повинно мати вбудовану довідку.

в) Графічна докуменок тація

1) Схема структурна бази даних.

2) Схема структурна програмного забезпечення.

3) Схема структурна потоків даних програмного забезпечення або його частини.

4) Схема структурна компонентів структур даних.

5) Схема взаємодії об’єктів.

6) Схема структурна класів програмного забезпечення.

7) Схема структурна станів інтерфейсу.

6 СТАДІЇ І ЕТАПИ РОЗРОБКИ

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	*Вивчення рекомендованої літератури*	*10.04.2020*	
2.	*Аналіз існуючих методів розв'язання задачі*	*13.04.2020*	
3.	*Постановка та формалізація задачі*	*17.04.2020*	
4.	*Аналіз вимог до програмного забезпечення*	*17.04.2020*	
5.	*Алгоритмізація задачі*	*24.04.2020*	
6.	*Моделювання програмного забезпечення*	*24.04.2020*	
7.	*Обґрунтування використовуваних технічних засобів*	*24.04.2020*	
8.	*Розробка архітектури програмного забезпечення*	*01.05.2020*	
9.	*Розробка програмного забезпечення*	*08.05.2020*	
10.	*Налагодження програми*	*12.05.2020*	
11.	*Виконання графічних документів*	*27.05.2020*	
12.	*Оформлення пояснювальної записки*	*25.05.2020*	
13.	*Подання ДП на попередній захист*	*15.05.2020*	
14.	*Подання ДП рецензенту*	*11.06.2020*	
15.	*Подання ДП на основний захист*	*12.06.2020*	

7 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

7.1 Види випробувань

Тестування розробленого програмного продукту виконується відповідно до “Програми та методики тестування”.

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____ Олександр ПАВЛОВ

“___” _____ 2020 р.

СЕРВІС АВТОМАТИЗОВАНОЇ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНИХ ВИПУСКНИХ РОБІТ НА ПЛАГІАТ

Програма та методика тестування

КПІ.ІП-6102. 045420.04.51

“ПОГОДЖЕНО”

Керівник проєкту:

_____ І. В. Стеценко

Нормоконтроль:

_____ К.І. Ліщук

Виконавець:

_____ Д. С. Блануца

Київ – 2020 року

ЗМІСТ

1 ОБ’ЄКТ ВИПРОБУВАНЬ

Сервіс автоматизованої перевірки кваліфікаційних випускних робіт на плагіат, який був розроблений. З використанням Spring, Symfony, PosgreSQL, Docker, ReactJS.

2 МЕТА ТЕСТУВАННЯ

Перевірити відповідність вимогам та реальні функціональні можливості функціональності.

Компоненти плану тестування повинні визначити:

- функціонал, що підлягає тестуванню;
- функціонал, що не підлягає тестуванню;
- підхід до тестування;
- критерії проходження тестів;
- процес тестування;
- процес тестування;
- вимоги середовища.

3 МЕТОДИ ТЕСТУВАННЯ

В рамках даного плану будуть використані такі типи тестування:

- тестування компонентів;
- тестування компонентів;
- інтеграційне тестування;
- тестування інтерфейсу;
- приймальне тестування;
- продуктивність;
- бета-тестування.

Для тестування інтеграції з Unicheck буде використовуватись додатково написаний тестовий сервер, що буде емолювати роботу Unicheck.

3.1 Модульне тестування

В модульному етапі будуть перевірені такі функції:

Для користувача:

- завантаження кваліфікаційних робіт до системи;
- отримання нагадувань щодо завантаження кваліфікаційних робіт до системи;
- отримання повідомлення за результатами перевірки кваліфікаційної роботи на плагіат.

Адміністратор системи за допомогою додатку може взаємодіяти з API для використання усіх функцій таких, як:

- налаштування нагадувань щодо завантаження кваліфікаційних робіт до системи;
- почати перевірку кваліфікаційних робіт на плагіат;
-
-

– відправити будь-яке текстове повідомлення на електрону пошту вибраній групі людей (студентам або викладачам), ці повідомлення можуть бути відправленні відразу або в зазначений час.

3.2 Інтеграційне тестування

В цьому етапі тестування буде протестована, власне інтеграція з системою Unicheck.

3.3 Продуктивність

– Використання багатьох запитів одночасно.
– Одночасний запуск перевірки на плагіат для багатьох дипломів.

3.4 Бета тестування

Бета-тестування - інтенсивне використання майже готової версії продукту з метою виявлення максимального числа помилок в його роботі для їх подальшого усунення перед остаточним виходом (релізом) продукту на ринок, до масового споживача.

Продукт тестується великою кількістю людей, після чого збирається вся інформація та корегуються помилки.

4 ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Працездатність web-ресурсу перевіряється шляхом:

– динамічного ручного тестування – введенням граничних та недопустимих значень в поля, які можна редагувати;

– динамічного ручного тестування на відповідність функціональним вимогам;

– статичного тестування коду;

– тестування web-ресурсу в різних web-браузерах;

– тестування при максимальному навантаженні;

– тестування стабільності роботи при різних умовах;

– тестування зручності використання;

– тестування інтерфейсу.

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____ Олександр ПАВЛОВ

“___” _____ 2020 р.

Сервіс автоматизованої перевірки кваліфікаційних випускних робіт на плагіат

Керівництво користувача

КПІ.ІП-6102. 045420.05.34

“ПОГОДЖЕНО”

Керівник проєкту:

_____ І. В. Стеценко

Нормоконтроль:

_____ К.І. Ліщук

Виконавець:

_____ Д. С. Бланууца

Київ – 2020 року

Авторизація

Для входу в систему незалежно це адміністратор чині користувач має ввести свою електрону пошту та пароль в форму що зображена на Рисунку 1.

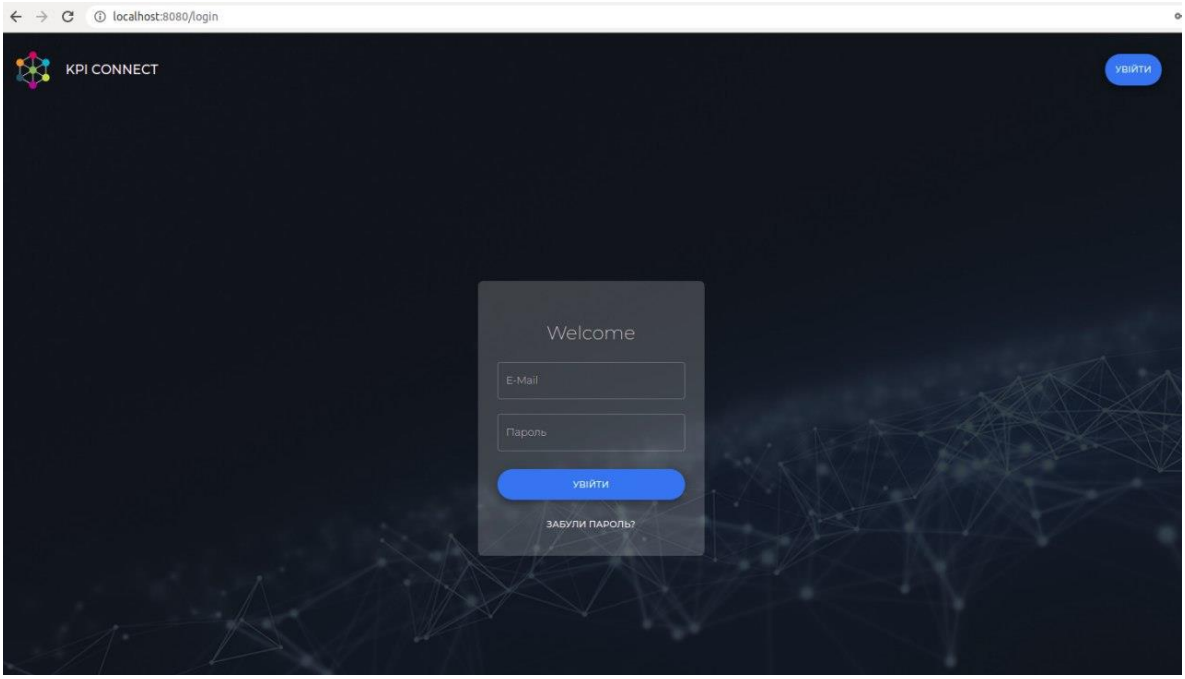


Рисунок 1 – Форма входу в систему KPI-CONNECT

Якщо користувач правильно ввів свою електрону пошту та пароль, то користувач потрапить на головну сторінку, що зображена на Рисунку 2.

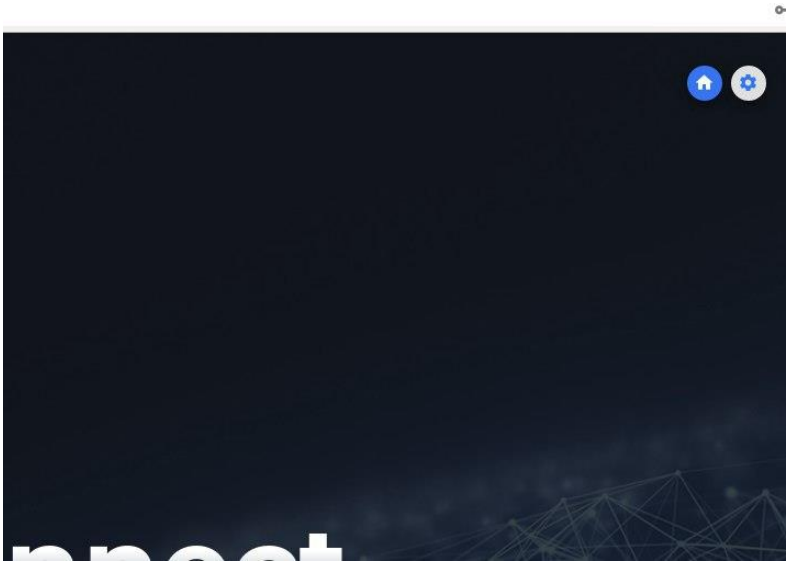


Рисунок 2 – Головна сторінка системи KPI-CONNECT

Загальне меню

Перша кнопка (у вигляді будинку) на головній сторінці веде на сторінку меню користувача. В залежності від ролей користувачів меню системи відрізняються. При натискані на другу кнопку відкриється сторінка з особистими даними користувача (Рисунок 3) також там можна змінити пароль на мову інтерфейсу.

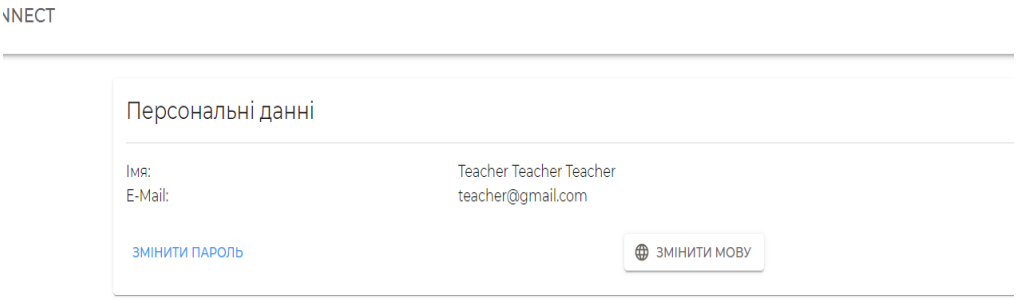


Рисунок 3 – Особиста сторінка користувача сторінка системи KPI-CONNECT

Меню студента

Студенту буде доступне наступне меню:

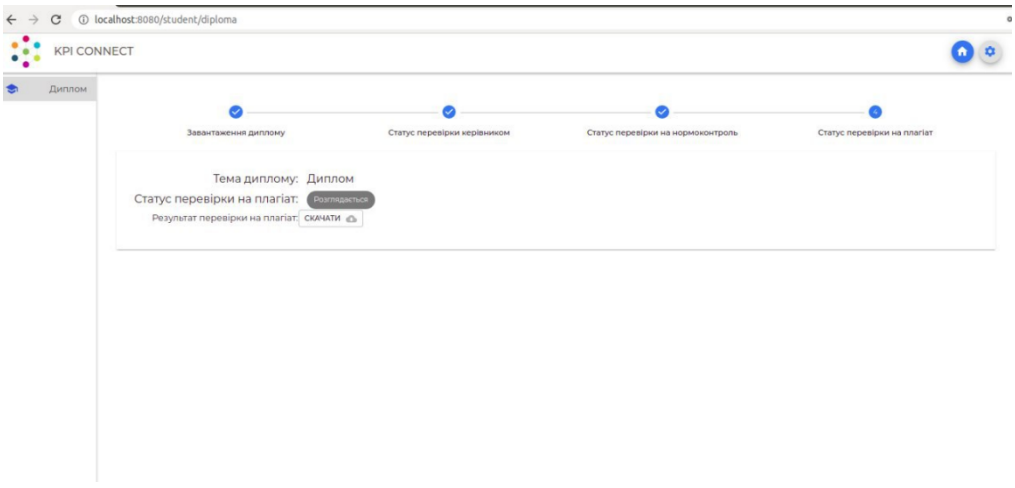


Рисунок 4 – Меню студента

Вище на Рисунку 4 зображене меню студента, де він може відслідковувати стадії перевірки диплому керівником чи нормоконтролером.

Наприклад, на Рисунку 4 студент вже пройшов контроль керівника та нормоконтролера та очікує перевірку на плагіат.

Коли робота студента буде перевірена, студент отримає повідомлення на електрону адресу з відсоток плагіату в його роботі. Приклад такого повідомлення зображений на Рисунку 5.

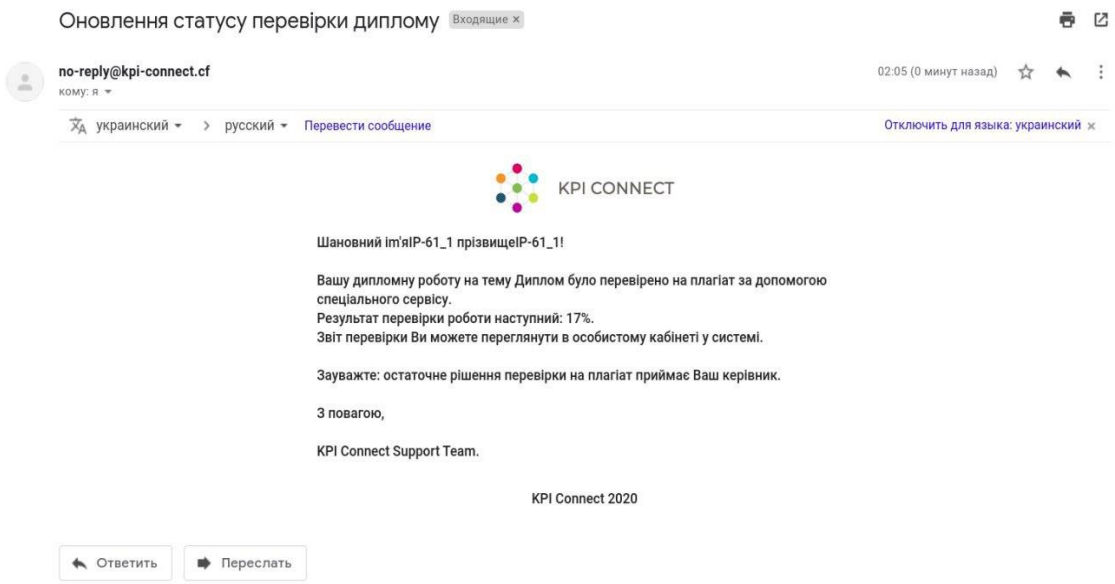


Рисунок 5 – Повідомлення про результат перевірки роботи на плагіат

Якщо робота студента пройшла перевірку на плагіат то в системі KPI-CONNECT буде відображене інше повідомлення, яке відображене на Рисунку 6.

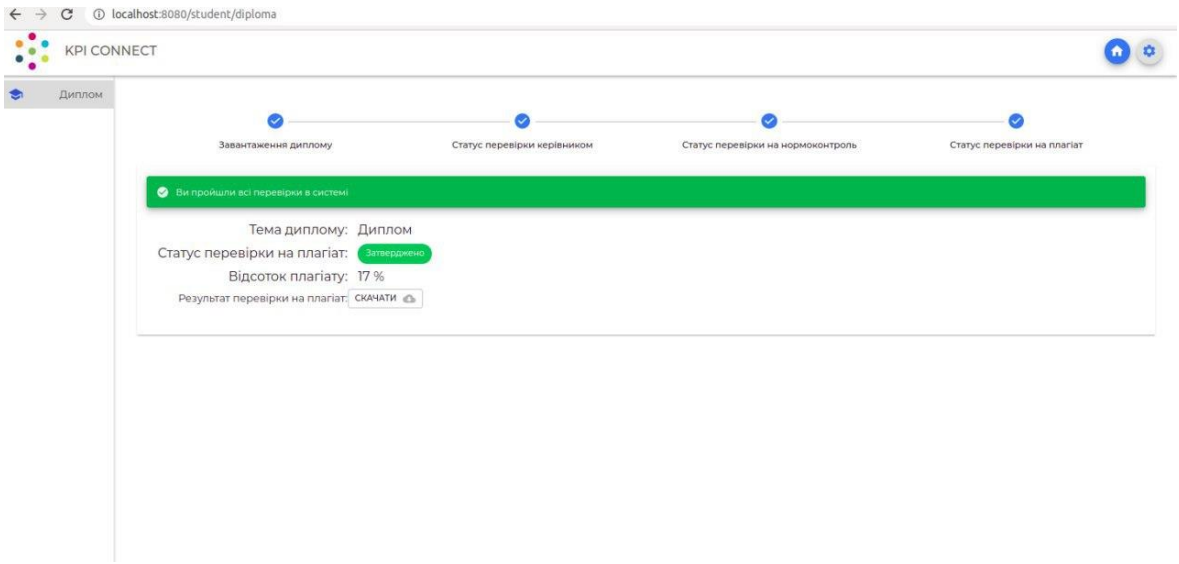


Рисунок 6 – Результат успішної перевірки на плагіат

Меню адміністратора

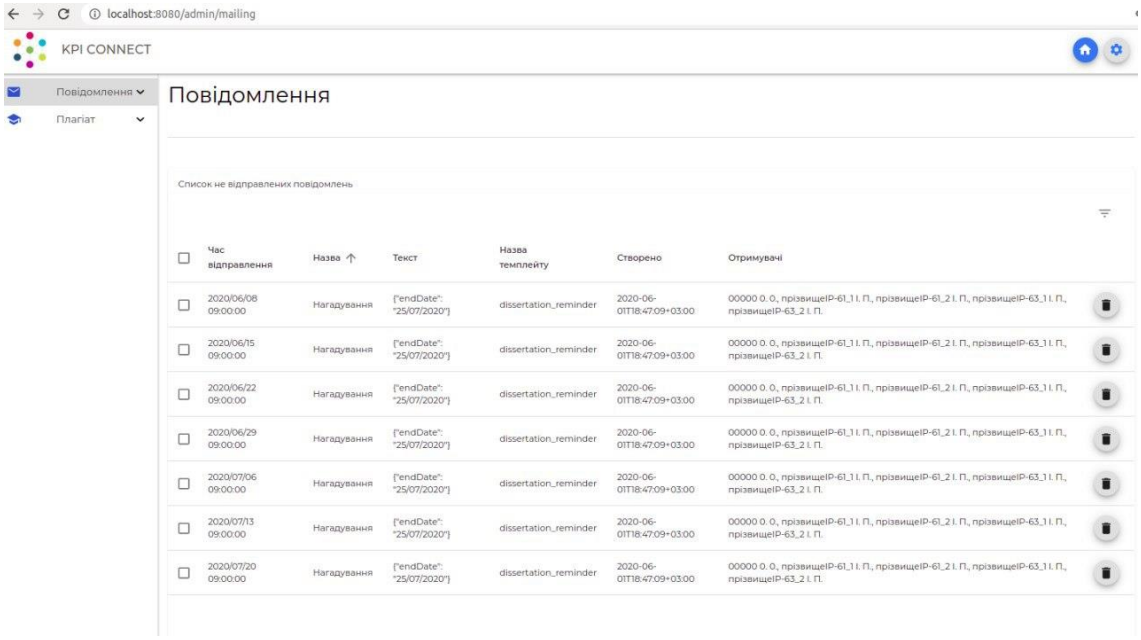


Рисунок 7 - Меню адміністратора

За замовчуванням адміністратору відображається список повідомлень, що плануються відправлятись, та прямо в цій таблиці адміністратор може видалити повідомлення та вони вже не будуть відправлятись.

Вище на Рисунку 7 також видно, що в меню адміністратора є дві категорії це “Плагіат” та “Повідомлення”. Далі детально розглянемо ці категорії.

В категорії “Повідомлення” адміністратор може не тільки переглянути список запланованих повідомлень на видалити їх, а ще й може створити нові.

Далі на Рисунку 8 зображена форма, що знаходиться на вкладці “Нове повідомлення”, скориставшись якою адміністратор може вибрати користувачів системи, фільтруючи їх за кафедрою та роллю (студент або викладач), часом відправлення (якщо час не буде вибраний то повідомлення будуть відправлені одразу) та темою і текстом повідомлення. Всі поля, крім часу, є обов’язковими. Якщо якесь обов’язкове значення не буде вибране, то буде відображене інформаційне повідомлення про помилку.

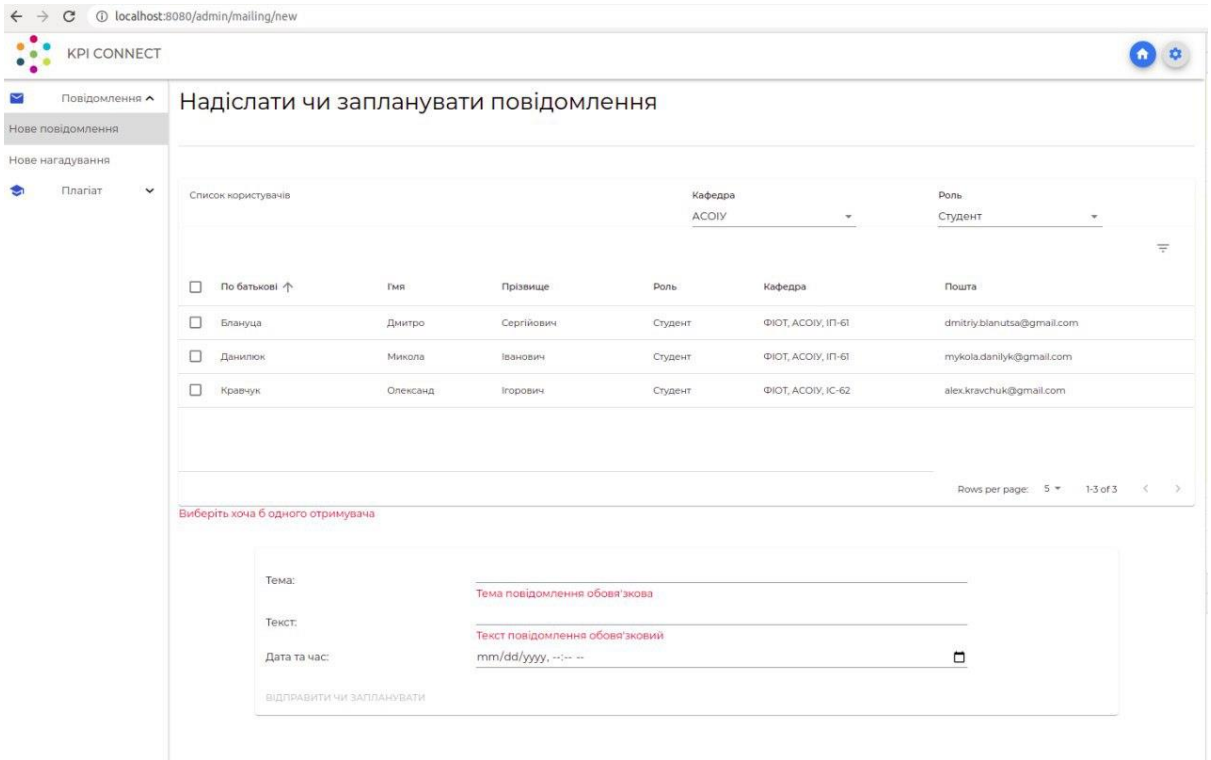


Рисунок 8 – Форма створення нового повідомлення

Також адміністратор скориставшись формою, що зображена на Рисунку 9 (вкладка “Нове нагадування”) може створити повідомлення декільком групам про необхідність надсилання дипломів до системи KPI-CONNECT.

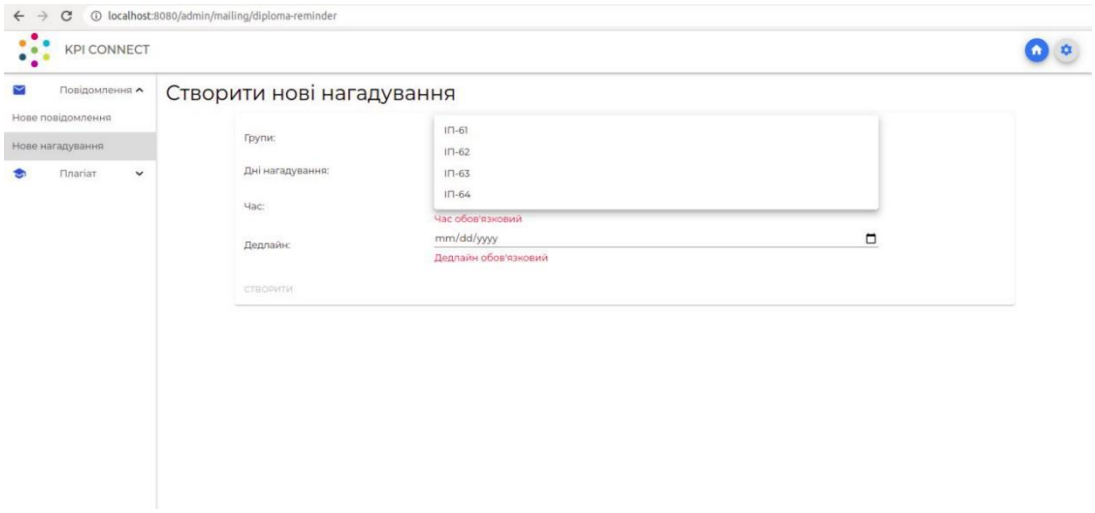


Рисунок 9 – Форма створення нового нагадування.

На цій формі адміністратор повинен обов’язково вибрати групи учнів, що будуть отримувати нагадування, дні тижня та час в який буде надсилатись повідомлення, та дату останнього терміну надсилання робіт. Ці повідомлення будуть надсилатись кожного вибраного дня неділі у вибраний час до останнього дня подання робіт. Якщо адміністратор не вибере якесь, обов’язкове значення то буде виведенне інформаційне повідомлення про помилку.

На Рисунку 10 зображене одне з таких нагадувань.

Рисунок 10 – Нагадування, про необхідність завантаження кваліфікаційної роботи до системи KPI-CONNECT

А перейшовши по вкладці “Не перевірені роботи” з категорії “Плагіат”, можна відслідковувати роботи, що вже пройшли нормоконтроль та очікують перевірку на плагіат. На Рисунку 11 зображена таблиця з роботами, що очікують перевірку на плагіат. Вона містить дані студента, керівника диплому, та тему диплому.

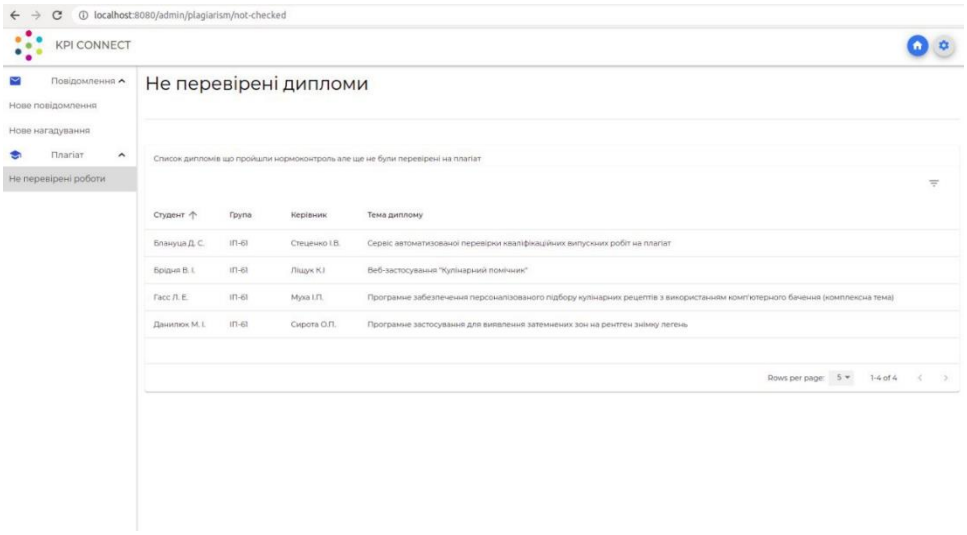


Рисунок 11 – Таблиця, що містить роботи, які очікують перевірку на плагіат

Натиснувши на категорію “Плагіат” адміністратор перейде на сторінку що містить таблицю (Рисунок 12) з актуальними статусами робіт що перевіряються.

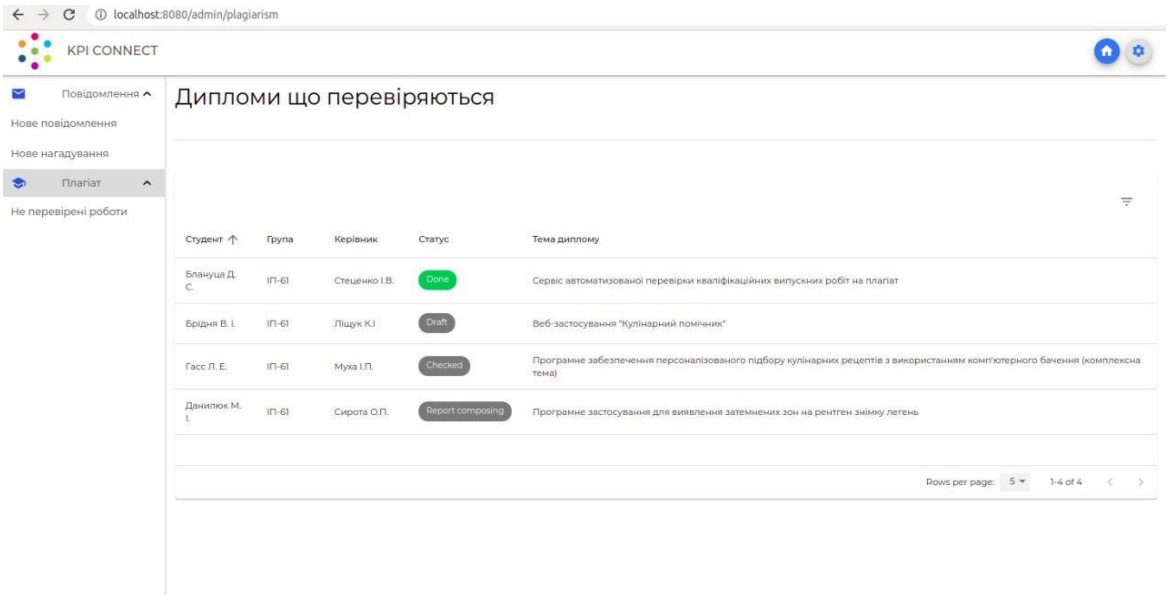


Рисунок 12 – Роботи що перевіряються або вже були перевірені

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____ Олександр ПАВЛОВ

“___” _____ 2020 р.

СЕРВІС АВТОМАТИЗОВАНОЇ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНИХ ВИПУСКНИХ РОБІТ НА ПЛАГІАТ

Опис програми

КПІ.ІП-6102.045420.06.13

“ПОГОДЖЕНО”

Керівник проєкту:

_____ І. В. Стеценко

Нормоконтроль:

_____ К.І. Ліщук

Виконавець:

_____ Д. С. Бланууца

Київ – 2020 року

Тексти програмного коду

Сервіс автоматизованої перевірки кваліфікаційних випускних робіт на плагіат

(Найменування програми (документа))

DVD-R

(Вид носія даних)

22 арк,

(Обсяг програми (документа) , арк.,) Кб)

Київ - 2020

```
package kpi.connect.unicheckworker.common;
import org.springframework.http.HttpMethod;
import org.springframework.util.StreamUtils;
import org.springframework.web.client.RestTemplate;
import java.io.File;
import java.io.FileOutputStream;
public class Utils {
    public static File getFileByUrl(RestTemplate restTemplate, String url, String fileName) {
        return restTemplate.execute(url.replace("kpi_connect_minio:9002", "localhost:8090"), HttpMethod.GET, null,
clientHttpResponse -> {
            File ret = new File(fileName);
            StreamUtils.copy(clientHttpResponse.getBody(), new FileOutputStream(ret));
            return ret;
        });
    }
}
```

```
package kpi.connect.unicheckworker.config.properties;
import javax.validation.constraints.NotBlank;
import javax.validation.constraints.NotNull;
public class KPIConnectProperties {
    @NotBlank
    private String authUrl;
    @NotBlank
    private String adminEmail;
    @NotBlank
    private String adminPassword;
    @NotBlank
    private String plagEmail;
    @NotBlank
    private String plagPassword;
    @NotBlank
    private String notCheckedDissertations;
    @NotBlank
    private String plagiarismSubmitUrl;
    @NotNull
    private Boolean test;
    private static String accessToken;
    public String getAuthUrl() {
```

```
    return authUrl;
}
public void setAuthUrl(String authUrl) {
    this.authUrl = authUrl;
}
public String getAdminEmail() {
    return adminEmail;
}
public void setAdminEmail(String adminEmail) {
    this.adminEmail = adminEmail;
}
public String getAdminPassword() {
    return adminPassword;
}
public void setAdminPassword(String adminPassword) {
    this.adminPassword = adminPassword;
}
public String getNotCheckedDissertations() {
    return notCheckedDissertations;
}
public void setNotCheckedDissertations(String notCheckedDissertations) {
    this.notCheckedDissertations = notCheckedDissertations;
}
public String getAccessToken() {
    return accessToken;
}
public static void setAccessToken(String accessToken) {
    KPIConnectProperties.accessToken = accessToken;
}

public String getPlagiarismSubmitUrl() {
    return plagiarismSubmitUrl;
}
public void setPlagiarismSubmitUrl(String plagiarismSubmitUrl) {
    this.plagiarismSubmitUrl = plagiarismSubmitUrl;
}
public Boolean getTest() {
    return test;
}
public void setTest(Boolean test) {
```

```
        this.test = test;
    }
    public String getPlagEmail() {
        return plagEmail;
    }
    public void setPlagEmail(String plagEmail) {
        this.plagEmail = plagEmail;
    }
    public String getPlagPassword() {
        return plagPassword;
    }
    public void setPlagPassword(String plagPassword) {
        this.plagPassword = plagPassword;
    }
}

package kpi.connect.unicheckworker.config.properties;
import org.springframework.http.MediaType;
import javax.validation.constraints.NotBlank;
public class UnicheckProperties {
    public static final String CONTENT_TYPE = MediaType.APPLICATION_JSON_VALUE;
    @NotBlank
    private String baseUrl;

    public String getBaseUrl() {
        return baseUrl;
    }
    public void setBaseUrl(String baseUrl) {
        this.baseUrl = baseUrl;
    }
}

package kpi.connect.unicheckworker.config;
import com.fasterxml.jackson.databind.ObjectMapper;
import kpi.connect.unicheckworker.config.properties.UnicheckProperties;
import org.springframework.boot.web.client.RestTemplateBuilder;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.http.HttpHeaders;
import org.springframework.http.MediaType;
```

```
import org.springframework.web.client.RestTemplate;
@Configuration
public class ApplicationConfig {
    public static final String KPI_CONNECT = "kpi-connect";
    public static final String UNICHECK = "unicheck";
    @Bean(name = KPI_CONNECT)
    public RestTemplate kpiConnectRestTemplate() {
        return new RestTemplateBuilder().build();
    }
    @Bean(name = UNICHECK)
    public RestTemplate unicheckRestTemplate() {
        return new RestTemplateBuilder().defaultHeader(HttpHeaders.CONTENT_TYPE,
UnicheckProperties.CONTENT_TYPE).build();
    }
    @Bean
    public ObjectMapper objectMapper() {
        return new ObjectMapper();
    }
}
package kpi.connect.unicheckworker.config;
import kpi.connect.unicheckworker.config.properties.KPIConnectProperties;
import kpi.connect.unicheckworker.config.properties.UnicheckProperties;
import org.springframework.boot.context.properties.ConfigurationProperties;
import org.springframework.boot.context.properties.EnableConfigurationProperties;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.validation.annotation.Validated;
@Configuration
@EnableConfigurationProperties
public class IntegrationConfig {
    @Bean
    @Validated
    @ConfigurationProperties("integration.kpi-connect")
    public KPIConnectProperties kpiConnectProperties() {
        return new KPIConnectProperties();
    }
    @Bean
    @Validated
    @ConfigurationProperties("integration.unicheck")
```

```
    public UnicheckProperties unicheckProperties() {
        return new UnicheckProperties();
    }
}
```

```
package kpi.connect.unicheckworker.controllers;
import kpi.connect.unicheckworker.services.DissertationPlagiarismService;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;
@RestController
public class ApiController {
    private final DissertationPlagiarismService service;
    public ApiController(DissertationPlagiarismService service) {
        this.service = service;
    }
    @GetMapping("/checking/start")
    public ResponseEntity<Void> start() {
        service.startChecking();
        return ResponseEntity.ok().build();
    }
}
```

```
package kpi.connect.unicheckworker.controllers;
import kpi.connect.unicheckworker.services.DissertationPlagiarismService;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
@RestController
@RequestMapping("/unicheck/hook")
public class WebHookController {
    private DissertationPlagiarismService service;
    public WebHookController(DissertationPlagiarismService service) {
        this.service = service;
    }
    @PostMapping("/check")
    public void check(@RequestBody String body) {
```

```
        service.handleWebhookCheckFinish(body);
    }
    @PostMapping("/report")
    public void report(@RequestBody String body) {
        service.handleWebhookReport(body);
    }
}
```

```
package kpi.connect.unicheckworker.dtos;
public class AttachmentResponse {
    private Long id;
    private String originalFilename;
    private String storagePath;
    public Long getId() {
        return id;
    }
    public String getOriginalFilename() {
        return originalFilename;
    }
    public String getStoragePath() {
        return storagePath;
    }
}
package kpi.connect.unicheckworker.dtos;
public class DissertationResponse {
    private String id;
    private AttachmentResponse mainPartAttachment;
    public AttachmentResponse getMainPartAttachment() {
        return mainPartAttachment;
    }
    public String getId() {
        return id;
    }
}
```

```
package kpi.connect.unicheckworker.exceptions;
import org.springframework.http.HttpStatus;
import org.springframework.web.bind.annotation.ResponseStatus;
```

```
@ResponseStatus(value = HttpStatus.NOT_FOUND)
public class DissertationNotFoundException extends RuntimeException {
    public DissertationNotFoundException(String id) {
        super("similarity - " + id);
    }
}

package kpi.connect.unicheckworker.exceptions;
import org.springframework.http.HttpStatus;
import org.springframework.web.bind.annotation.ResponseStatus;
@ResponseStatus(value = HttpStatus.BAD_REQUEST)
public class SubmitDiplomaException extends RuntimeException {
}

package kpi.connect.unicheckworker.exceptions;
import org.springframework.http.HttpStatus;
import org.springframework.web.bind.annotation.ResponseStatus;
@ResponseStatus(value = HttpStatus.BAD_REQUEST)
public class UnicheckApiException extends RuntimeException {
}

package kpi.connect.unicheckworker.integration;
import com.fasterxml.jackson.annotation.JsonProperty;
import kpi.connect.unicheckworker.config.properties.KPIConnectProperties;
import kpi.connect.unicheckworker.dtos.DissertationResponse;
import kpi.connect.unicheckworker.exceptions.SubmitDiplomaException;
import kpi.connect.unicheckworker.persistence.entity.DissertationPlagiarism;
import kpi.connect.unicheckworker.persistence.repository.DissertationPlagiarismRepository;
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.core.io.FileSystemResource;
import org.springframework.http.*;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;
import org.springframework.util.LinkedMultiValueMap;
import org.springframework.util.MultiValueMap;
import org.springframework.web.client.RestTemplate;
import java.io.File;
import java.util.Objects;
```

```
import java.util.Optional;
import static kpi.connect.unicheckworker.config.ApplicationConfig.KPI_CONNECT;
@Service
public class DefaultKPIConnectClient implements KPIConnectClient {
    private final RestTemplate restTemplate;
    private final KPIConnectProperties properties;
    private final DissertationPlagiarismRepository repository;
    private final UnicheckClient unicheckClient;
    public DefaultKPIConnectClient(@Qualifier(KPI_CONNECT) RestTemplate restTemplate, KPIConnectProperties properties,
                                   DissertationPlagiarismRepository repository, UnicheckClient unicheckClient) {
        this.restTemplate = restTemplate;
        this.properties = properties;
        this.repository = repository;
        this.unicheckClient = unicheckClient;
    }
    @Override
    @Transactional
    public void getAndSaveNewDissertationForChecking() {
        HttpHeaders headers = new HttpHeaders();
        headers.setBearerAuth(getBearerAccessToken());
        ResponseEntity<DissertationResponse[]> response = restTemplate.exchange(properties.getNotCheckedDissertations(),
                HttpMethod.GET, new HttpEntity<>(headers), DissertationResponse[].class);
        if (response.getBody() != null && response.getBody().length != 0)  {
            for (DissertationResponse dissertationResponse : response.getBody()) {
                Optional<DissertationPlagiarism> optional = repository.findByDiplomaId(dissertationResponse.getId());
                if (optional.isEmpty()) {
                    DissertationPlagiarism dissertationPlagiarism = new DissertationPlagiarism(
                            dissertationResponse.getMainPartAttachment().getId(),
                            dissertationResponse.getMainPartAttachment().getOriginalFilename(),
                            dissertationResponse.getMainPartAttachment().getStoragePath(),
                            dissertationResponse.getId()
                    );
                    repository.save(dissertationPlagiarism);
                }
            }
            repository.flush();
        }
    }
```

```
    @Override
    @Transactional
    public void sendPlagiarismCheckResponse(DissertationPlagiarism dissertationPlagiarism) {
        File report = unicheckClient.getReport(dissertationPlagiarism);
        HttpHeaders headers = new HttpHeaders();
        headers.setContentType(MediaType.MULTIPART_FORM_DATA);
        headers.setBearerAuth(getPlagBearerAccessToken());
        MultiValueMap<String, Object> body = new LinkedMultiValueMap<>();
        body.add("report", new FileSystemResource(report));
        body.add("percent", Double.valueOf(dissertationPlagiarism.getSimilarity()).intValue());
        HttpEntity<MultiValueMap<String, Object>> entity = new HttpEntity<>(body, headers);
        String url = properties.getTest() ? "http://localhost:8090/plagiarism/diplomas/{diplomaId}/submit" :
properties.getPlagiarismSubmitUrl();
        ResponseEntity<String> response = restTemplate.exchange(url.replace("{diplomaId}", dissertationPlagiarism.getDiplomaId()),
                HttpMethod.POST, entity, String.class);
        report.delete();
        if (response.getStatusCode() != HttpStatus.OK) {
            throw new SubmitDiplomaException();
        }
    }
    private String getBearerAccessToken() {
        LoginRequest value = new LoginRequest(properties.getAdminEmail(), properties.getAdminPassword());
        if (properties.getAccessToken() == null) {
            ResponseEntity<LoginResponse> response =
                    restTemplate.postForEntity(properties.getAuthUrl(), value, LoginResponse.class);
            KPIConnectProperties.setAccessToken(Objects.requireNonNull(response.getBody()).getAccessToken());
        }
        return properties.getAccessToken();
    }
    private String getPlagBearerAccessToken() {
        LoginRequest value = new LoginRequest(properties.getPlagEmail(), properties.getPlagPassword());

        ResponseEntity<LoginResponse> response = restTemplate.postForEntity(properties.getAuthUrl(), value, LoginResponse.class);
        return Objects.requireNonNull(response.getBody()).getAccessToken();
    }
    private static class LoginRequest {
        private String username;
```

```
        private String password;
        public LoginRequest(String username, String password) {
            this.username = username;
            this.password = password;
        }
        public String getUsername() {
            return username;
        }
        public void setUsername(String username) {
            this.username = username;
        }
        public String getPassword() {
            return password;
        }
        public void setPassword(String password) {
            this.password = password;
        }
    }
    private static class LoginResponse {
        @JsonProperty("access_token")
        private String accessToken;
        public String getAccessToken() {
            return accessToken;
        }
        public void setAccessToken(String accessToken) {
            this.accessToken = accessToken;
        }
    }
}
```

```
package kpi.connect.unicheckworker.integration;
import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.databind.ObjectMapper;
import kpi.connect.unicheckworker.config.properties.UnicheckProperties;
import kpi.connect.unicheckworker.persistence.entity.CheckStatus;
import kpi.connect.unicheckworker.persistence.entity.DissertationPlagiarism;
import kpi.connect.unicheckworker.persistence.repository.DissertationPlagiarismRepository;
import org.springframework.beans.factory.annotation.Qualifier;
```

```
import org.springframework.core.io.FileSystemResource;
import org.springframework.http.HttpEntity;
import org.springframework.http.HttpHeaders;
import org.springframework.http.MediaType;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Propagation;
import org.springframework.transaction.annotation.Transactional;
import org.springframework.util.LinkedMultiValueMap;
import org.springframework.util.MultiValueMap;
import org.springframework.web.client.RestTemplate;
import java.io.File;
import static kpi.connect.unicheckworker.common.Utils.getFileByUrl;
import static kpi.connect.unicheckworker.config.ApplicationConfig.UNICHECK;
@Service
public class DefaultUnicheckClient implements UnicheckClient {
    private final RestTemplate restTemplate;
    private final UnicheckProperties properties;
    private final ObjectMapper mapper;
    private final DissertationPlagiarismRepository repository;
    public DefaultUnicheckClient(@Qualifier(UNICHECK) RestTemplate restTemplate, UnicheckProperties properties,
ObjectMapper mapper,
                                 DissertationPlagiarismRepository repository) {
        this.restTemplate = restTemplate;
        this.properties = properties;
        this.mapper = mapper;
        this.repository = repository;
    }
    @Override
    @Transactional
    public DissertationPlagiarism startSimilarityCheck(DissertationPlagiarism dissertationPlagiarism) {
        if (dissertationPlagiarism.getFileId() == null) {
            dissertationPlagiarism = uploadFile(dissertationPlagiarism);
        }
        String body = "{ \"data\": { \"type\": \"similarityCheck\", \"attributes\": { \"search_types\": { \"web\":true, \"library\": false },
\"parameters\": { \"sensitivity\": { \"percentage\": 0, \"words_count\": 8 } } } }, \"relationships\": { \"file\": { \"data\": { \"id\":
\"<file_id>\", \"type\": \"file\" } } }}";
        HttpEntity<String> request = new HttpEntity<>(body, getDefaultHeaders());
        String responseBody =
                restTemplate.postForEntity(properties.getBaseUrl() + "/similarity/checks", request, String.class).getBody();
```

```
        dissertationPlagiarism.setSimilarityCheckId(getDataId(responseBody));
        dissertationPlagiarism.setStatus(CheckStatus.CHECKING);
        return repository.save(dissertationPlagiarism);
    }
    @Override
    @Transactional
    public DissertationPlagiarism startReportGenerating(DissertationPlagiarism dissertationPlagiarism) {
        String body = "{\n" +
                "    \"data\": {\n" +
                "        \"type\" :\"similarity-check-report-export\",\n" +
                "        \"attributes\":{\n" +
                "            \"format\" :\"pdf\",\n" +
                "            \"locale_code\" :\"RU\"\n" +
                "        }\n" +
                "    }\n" +
                "}";
        HttpEntity<String> request = new HttpEntity<>(body, getDefaultHeaders());
        String url = properties.getBaseUrl() + "/similarity/checks/" + dissertationPlagiarism.getSimilarityCheckId() + "/report/export";
        String responseBody = restTemplate.postForEntity(url, request, String.class).getBody();
        dissertationPlagiarism.setReportExportJobId(getDataId(responseBody));
        dissertationPlagiarism.setStatus(CheckStatus.REPORT_COMPOSING);
        return repository.save(dissertationPlagiarism);
    }
    @Override
    @Transactional
    public File getReport(DissertationPlagiarism dissertation) {
        String url = properties.getBaseUrl() + "/similarity/checks/" + dissertation.getSimilarityCheckId() + "/report/export/" +
dissertation.getReportExportJobId();
        return getFileByUrl(restTemplate, url, dissertation.getId() + "_report.pdf");
    }
    private DissertationPlagiarism uploadFile(DissertationPlagiarism dissertationPlagiarism) {
        File fileByUrl = getFileByUrl(restTemplate, dissertationPlagiarism.getStoragePath(),
dissertationPlagiarism.getOriginalFilename());
        MultiValueMap<String, Object> body = new LinkedMultiValueMap<>();
        body.add("file", new FileSystemResource(fileByUrl));
        HttpEntity<MultiValueMap<String, Object>> request = new HttpEntity<>(body,
getDefaultHeaders(MediaType.MULTIPART_FORM_DATA_VALUE));
        String responseBody =
```

```
            restTemplate.postForEntity(properties.getBaseUrl() + "/files", request, String.class).getBody();
        fileByUrl.delete();
        dissertationPlagiarism.setFileId(getDataId(responseBody));
        dissertationPlagiarism.setStatus(CheckStatus.DISSERTATION_UPLOADED);
        return repository.save(dissertationPlagiarism);
    }
    private String getDataId(String body) {
        try {
            return mapper.readTree(body).get("data").get("id").asText();
        } catch (JsonProcessingException e) {
            throw new RuntimeException(e);
        }
    }
    private HttpHeaders getDefaultHeaders() {
        return getDefaultHeaders(null);
    }
    private HttpHeaders getDefaultHeaders(String contentType) {
        HttpHeaders headers = new HttpHeaders();
        if (contentType != null) {
            headers.set(HttpHeaders.CONTENT_TYPE, contentType);
        }
        //headers.setBearerAuth();

        return headers;
    }
}
```

```
package kpi.connect.unicheckworker.integration;
import kpi.connect.unicheckworker.persistence.entity.DissertationPlagiarism;
public interface KPIConnectClient {
    void getAndSaveNewDissertationForChecking();
    void sendPlagiarismCheckResponse(DissertationPlagiarism dissertationPlagiarism);
}
```

```
package kpi.connect.unicheckworker.integration;
import kpi.connect.unicheckworker.persistence.entity.DissertationPlagiarism;
import java.io.File;
public interface UnicheckClient {
```

```
    DissertationPlagiarism startSimilarityCheck(DissertationPlagiarism dissertationPlagiarism);
    DissertationPlagiarism startReportGenerating(DissertationPlagiarism dissertationPlagiarism);
    File getReport(DissertationPlagiarism dissertation);
}
```

```
package kpi.connect.unicheckworker.persistence.entity;
public enum CheckStatus {
    DRAFT,
    DISSERTATION_UPLOADED,
    CHECKING,
    CHECKED,
    REPORT_COMPOSING,
    DONE
}
```

```
package kpi.connect.unicheckworker.persistence.entity;
import javax.persistence.Entity;
import javax.persistence.EnumType;
import javax.persistence.Enumerated;
import javax.persistence.Id;
@Entity
public class DissertationPlagiarism {
    @Id
    private Long id;
    private String originalFilename;
    private String storagePath;
    private String fileId;
    private String similarityCheckId;
    private double similarity;
    private String reportExportJobId;
    private String diplomaId;
    @Enumerated(EnumType.STRING)
    private CheckStatus status = CheckStatus.DRAFT
    public DissertationPlagiarism() { }
    public DissertationPlagiarism(Long id, String originalFilename, String storagePath, String diplomaId) {
        this.id = id;
        this.originalFilename = originalFilename;
```

```
        this.storagePath = storagePath;
        this.diplomaId = diplomaId;
    }
    public Long getId() {
        return id;
    }
    public String getOriginalFilename() {
        return originalFilename;
    }
    public String getStoragePath() {
        return storagePath;
    }
    public String getFileId() {
        return fileId;
    }
    public void setFileId(String fileId) {
        this.fileId = fileId;
    }
    public String getSimilarityCheckId() {
        return similarityCheckId;
    }
    public void setSimilarityCheckId(String similarityCheckApi) {
        this.similarityCheckId = similarityCheckApi;
    }
    public double getSimilarity() {
        return similarity;
    }
    public void setSimilarity(double similarity) {
        this.similarity = similarity;
    }
    public String getReportExportJobId() {
        return reportExportJobId;
    }
    public void setReportExportJobId(String reportExportJobId) {
        this.reportExportJobId = reportExportJobId;
    }
    public String getDiplomaId() {
        return diplomaId;
```

```
    }
    public void setDiplomaId(String diplomaId) {
        this.diplomaId = diplomaId;
    }
    public void setId(Long id) {
        this.id = id;
    }
    public void setOriginalFilename(String originalFilename) {
        this.originalFilename = originalFilename;
    }
    public void setStoragePath(String storagePath) {
        this.storagePath = storagePath;
    }
    public CheckStatus getStatus() {
        return status;
    }
    public void setStatus(CheckStatus status) {
        this.status = status;
    }
}
```

```
package kpi.connect.unicheckworker.persistence.repository;
import kpi.connect.unicheckworker.persistence.entity.CheckStatus;
import kpi.connect.unicheckworker.persistence.entity.DissertationPlagiarism;
import org.springframework.data.jpa.repository.JpaRepository;
import java.util.List;
import java.util.Optional;
import java.util.Set;
public interface DissertationPlagiarismRepository extends JpaRepository<DissertationPlagiarism, Long> {
    Optional<DissertationPlagiarism> findBySimilarityCheckId(String similarityCheckId);
    Optional<DissertationPlagiarism> findByReportExportJobId(String reportExportJobId);
    Optional<DissertationPlagiarism> findByDiplomaId(String diplomaId);
    List<DissertationPlagiarism> findAllByStatus(CheckStatus status);
    List<DissertationPlagiarism> findAllByStatusIsIn(Set<CheckStatus> statuses);
}
```

```
package kpi.connect.unicheckworker.services;
import kpi.connect.unicheckworker.integration.UnicheckClient;
```

```
import kpi.connect.unicheckworker.persistence.entity.DissertationPlagiarism;
import kpi.connect.unicheckworker.persistence.repository.DissertationPlagiarismRepository;
import org.springframework.boot.CommandLineRunner;
import org.springframework.stereotype.Component;
import java.util.List;
import java.util.Set;
import static kpi.connect.unicheckworker.persistence.entity.CheckStatus.*;
@Component
public class CommandLineAppStartupRunner implements CommandLineRunner {
    private final DissertationPlagiarismRepository repository;
    private final UnicheckClient unicheckClient;
    public CommandLineAppStartupRunner(DissertationPlagiarismRepository repository, UnicheckClient unicheckClient) {
        this.repository = repository;
        this.unicheckClient = unicheckClient;
    }
    @Override
    public void run(String... args) {
        List<DissertationPlagiarism> dissertations =
                this.repository.findAllByStatusIsIn(Set.of(DRAFT, DISSERTATION_UPLOADED, CHECKED));
        for (DissertationPlagiarism dissertation : dissertations) {
            if (dissertation.getStatus() == DRAFT || dissertation.getStatus() == DISSERTATION_UPLOADED) {
                unicheckClient.startSimilarityCheck(dissertation);
            } else if (dissertation.getStatus() == CHECKED) {
                unicheckClient.startReportGenerating(dissertation);
            }
        }
    }
}


package kpi.connect.unicheckworker.services;
import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.databind.JsonNode;
import com.fasterxml.jackson.databind.ObjectMapper;
import kpi.connect.unicheckworker.exceptions.DissertationNotFoundException;
import kpi.connect.unicheckworker.integration.KPIConnectClient;
import kpi.connect.unicheckworker.integration.UnicheckClient;
import kpi.connect.unicheckworker.persistence.entity.CheckStatus;
import kpi.connect.unicheckworker.persistence.entity.DissertationPlagiarism;
```

```
import kpi.connect.unicheckworker.persistence.repository.DissertationPlagiarismRepository;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;
@Service
public class DefaultDissertationPlagiarismService implements DissertationPlagiarismService {
    private final ObjectMapper mapper;
    private final DissertationPlagiarismRepository repository;
    private final UnicheckClient unicheckClient;
    private final KPIConnectClient kpiConnectClient;
    public DefaultDissertationPlagiarismService(ObjectMapper mapper, DissertationPlagiarismRepository repository,
                                                UnicheckClient unicheckClient, KPIConnectClient kpiConnectClient) {
        this.mapper = mapper;
        this.repository = repository;
        this.unicheckClient = unicheckClient;
        this.kpiConnectClient = kpiConnectClient;
    }
    @Override
    @Transactional
    public void handleWebhookCheckFinish(String body) {
        String similarityCheckId;
        double similarity;
        try {
            JsonNode data = mapper.readTree(body).get("data");
            similarityCheckId = data.get("id").asText();
            similarity = data.get("attributes").get("similarity").asDouble();
        } catch (JsonProcessingException e) {
            throw new RuntimeException(e);
        }
        repository.findBySimilarityCheckId(similarityCheckId).ifPresentOrElse(dissertation -> {
            dissertation.setSimilarity(similarity);
            dissertation.setStatus(CheckStatus.CHECKED);
            unicheckClient.startReportGenerating(dissertation);
        }, () -> {
            throw new DissertationNotFoundException(similarityCheckId);
        });
    }
    @Override
    @Transactional
```

```
    public void handleWebhookReport(String body) {
        String reportExportJobId;
        try {
            reportExportJobId = mapper.readTree(body).get("data").get("id").asText();
        } catch (JsonProcessingException e) {
            throw new RuntimeException(e);
        }
        repository.findByReportExportJobId(reportExportJobId).ifPresentOrElse(dissertation -> {
            kpiConnectClient.sendPlagiarismCheckResponse(dissertation);
            dissertation.setStatus(CheckStatus.DONE);
        }, () -> {
            throw new DissertationNotFoundException(reportExportJobId);
        });
    }
    @Override
    @Transactional
    public void startChecking() {
        kpiConnectClient.getAndSaveNewDissertationForChecking();

        for (DissertationPlagiarism dissertation : repository.findAllByStatus(CheckStatus.DRAFT)) {
            unicheckClient.startSimilarityCheck(dissertation);
        }
    }
}
```

```
package kpi.connect.unicheckworker.services;
public interface DissertationPlagiarismService {
    void handleWebhookCheckFinish(String body);
    void handleWebhookReport(String body);
    void startChecking();
}
```

```
package kpi.connect.unicheckworker;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
@SpringBootApplication
public class UnicheckWorkerApplication {
```

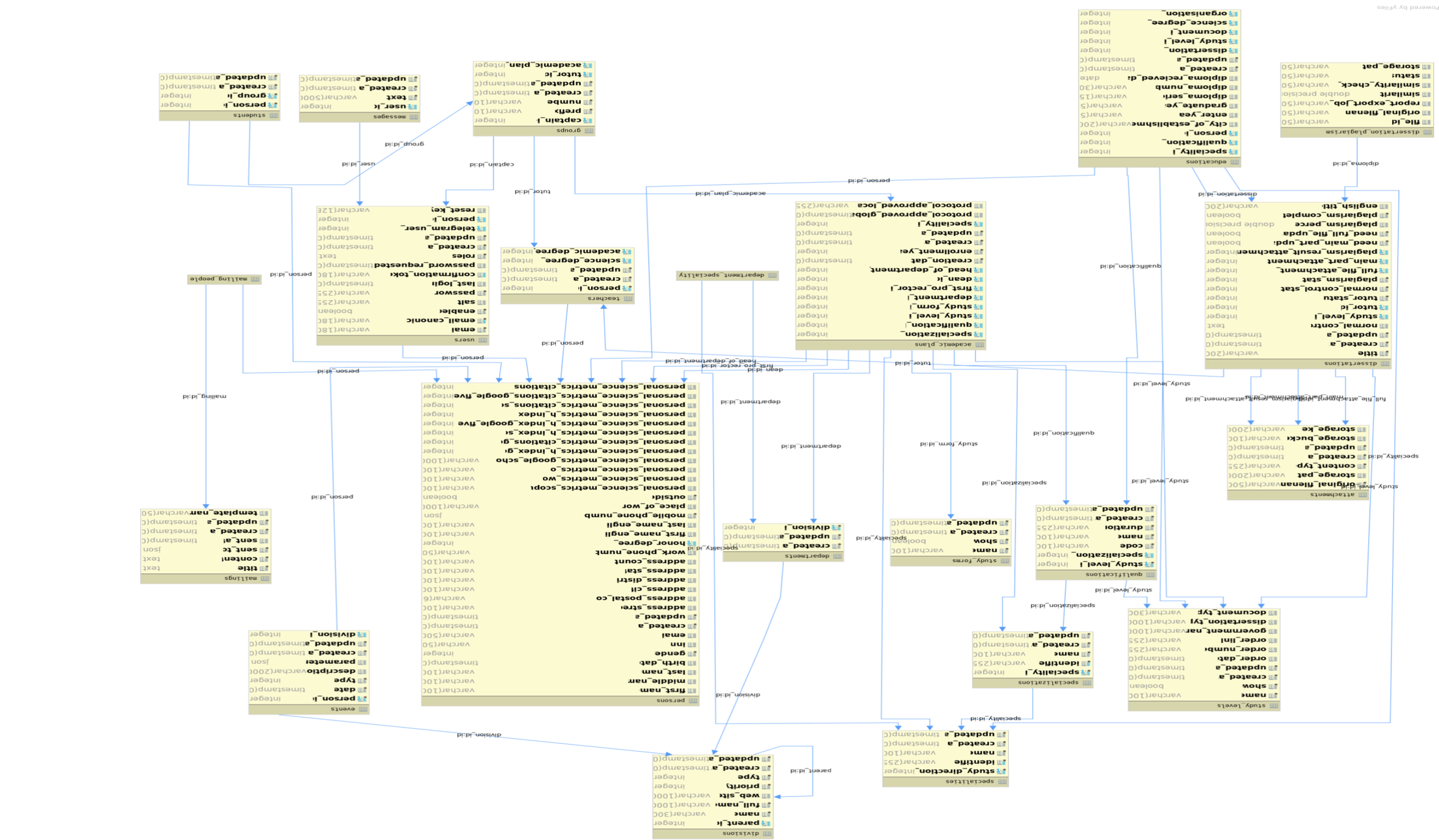
```
	public static void main(String[] args) {
		SpringApplication.run(UnicheckWorkerApplication.class, args);
	}
}
```

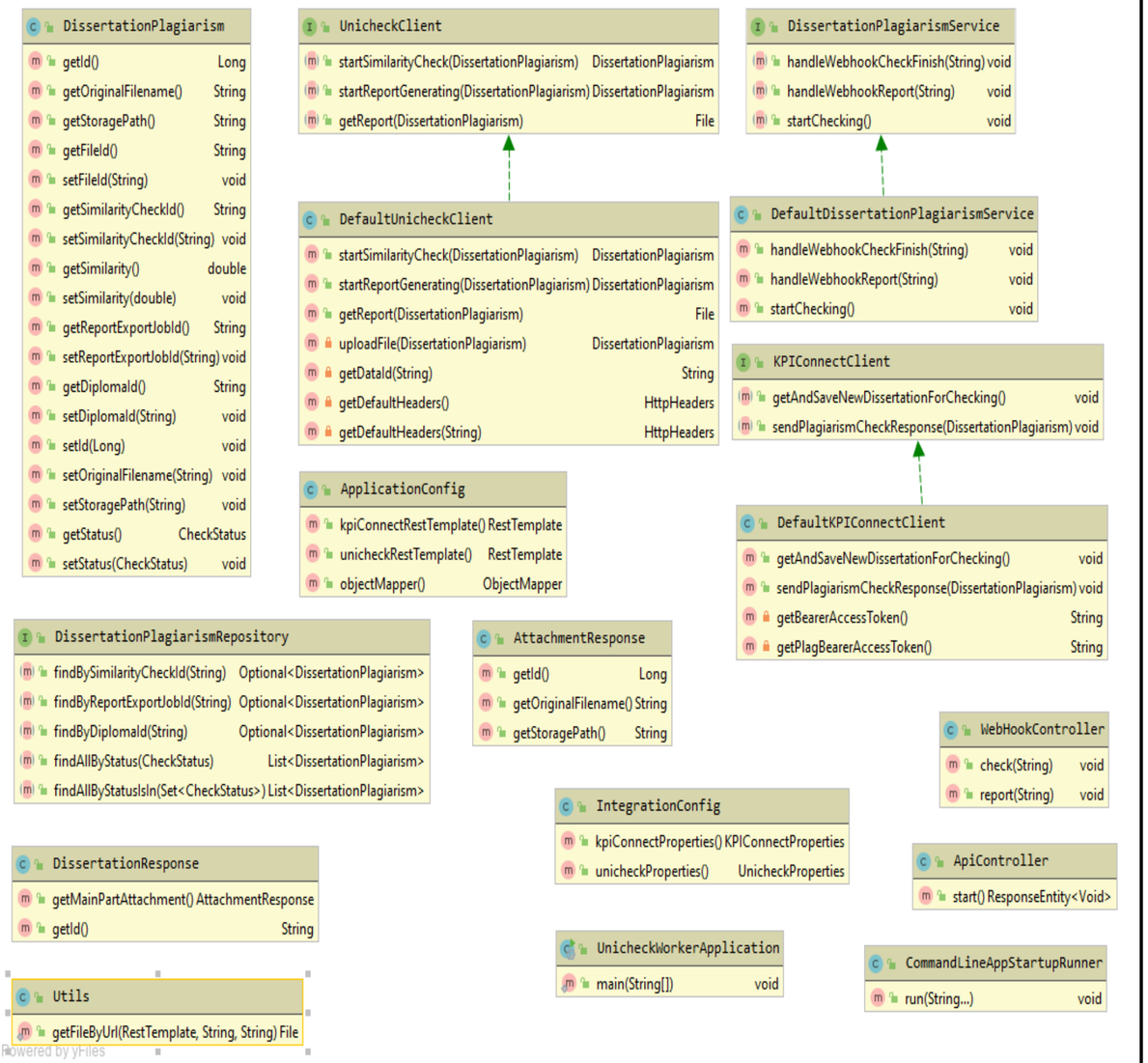
```
server.port = 8070
spring.datasource.url = jdbc:h2:file:./test;DB_CLOSE_ON_EXIT=FALSE
spring.datasource.driver-class-name = org.h2.Driver
spring.jpa.database-platform = org.hibernate.dialect.H2Dialect
spring.jpa.hibernate.ddl-auto = update
spring.h2.console.enabled=true
spring.h2.console.path=/h2
integration.kpi-connect.baseUrl = http://api.kpi-connect.test
integration.kpi-connect.authUrl = ${integration.kpi-connect.base-url}/auth/login
integration.kpi-connect.notCheckedDissertations = ${integration.kpi-connect.base-url}/admin/diplomas
integration.kpi-connect.adminEmail = admin@admin.ua
integration.kpi-connect.adminPassword = 1
integration.kpi-connect.plagEmail = plag@admin.ua
integration.kpi-connect.plagPassword = 1
integration.kpi-connect.plagiarismSubmitUrl = ${integration.kpi-connect.base-url}/plagiarism/diplomas/{diplomaId}/submit
integration.kpi-connect.test = false
integration.unicheck.baseUrl = http://localhost:8090
```

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
	<modelVersion>4.0.0</modelVersion>
	<parent>
		<groupId>org.springframework.boot</groupId>
		<artifactId>spring-boot-starter-parent</artifactId>
		<version>2.3.0.RELEASE</version>
		<relativePath/> <!-- lookup parent from repository -->
	</parent>
	<groupId>kpi.connect</groupId>
	<artifactId>unicheck-worker</artifactId>
	<version>0.0.1-SNAPSHOT</version>
	<name>unicheck-worker</name>
	<description>KPI Connect unicheck worker</description>
```

```
<properties>
        <java.version>11</java.version>
</properties>
<dependencies>
        <dependency>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-data-jpa</artifactId>
        </dependency>
        <dependency>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-validation</artifactId>
        </dependency>
        <dependency>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <dependency>
                <groupId>com.h2database</groupId>
                <artifactId>h2</artifactId>
                <scope>runtime</scope>
        </dependency>
        <dependency>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-test</artifactId>
                <scope>test</scope>
                <exclusions>
                        <exclusion>
                                <groupId>org.junit.vintage</groupId>
                                <artifactId>junit-vintage-engine</artifactId>
                        </exclusion>
                </exclusions>
        </dependency>
</dependencies>
<build>
        <plugins>
                <plugin>
                        <groupId>org.springframework.boot</groupId>
```

```
    <artifactId>spring-boot-maven-plugin</artifactId>
                </plugin>
            </plugins>
        </build>
    </project>
```





Інв. № підл.	Підп. дата	Інв. № взам. підл.	Інв. № дубл.	Підп. дата

Зм.	Арк.	№ докум.	Підп.	Дата
Розроб.		Бланука Д. С.		
Перев.		Стеценко І. В.		
Т. Кон.				
Н. Кон.		Ліщук К.І.		
Затв.		Стеценко І. В.		

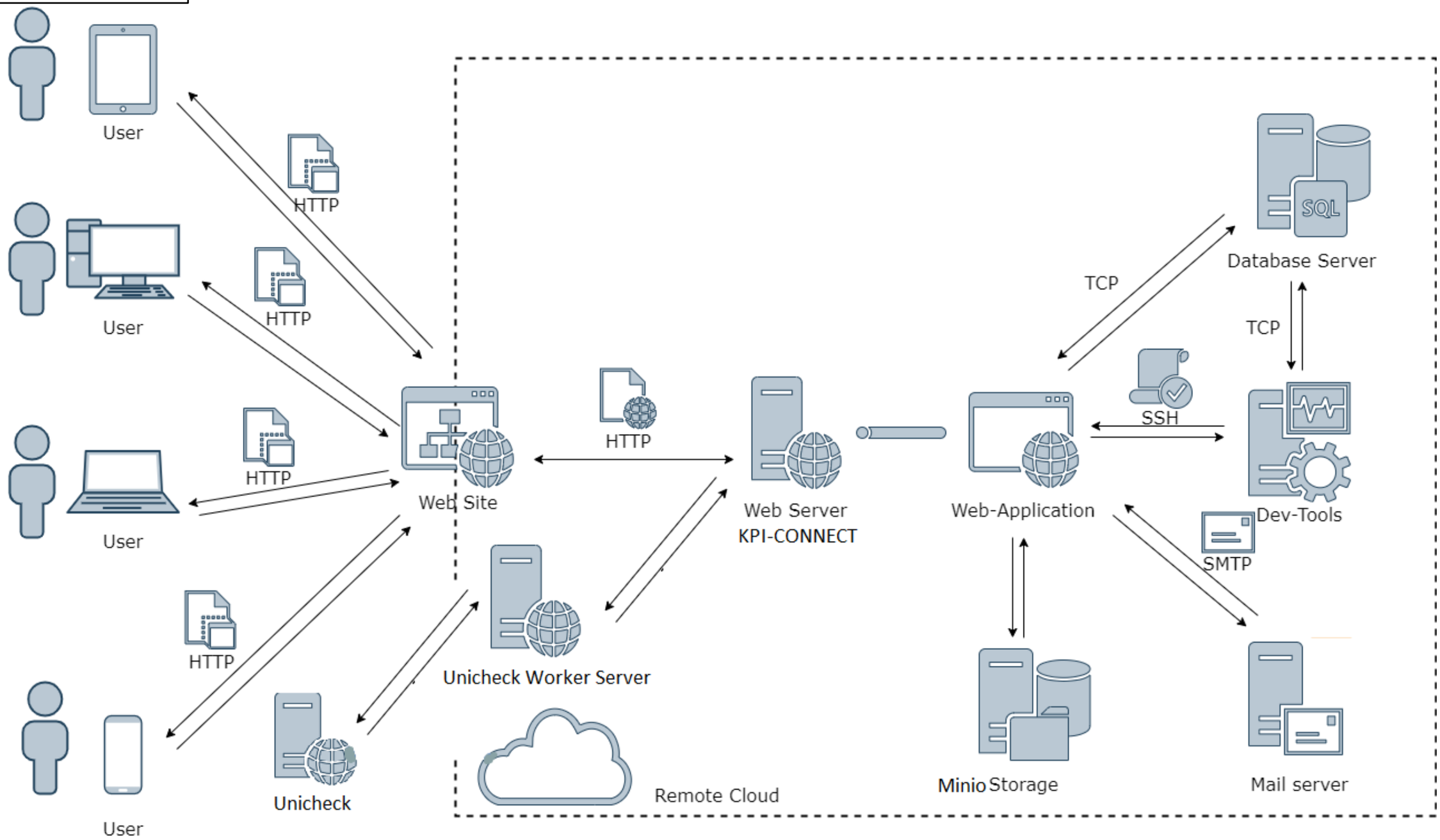
КПІ.ІП-6102.045420.08.99 СС

Схема структурна класів програмного забезпечення

Лит.	Арк.	Аркушів
		1
Аркуш 1		Аркушів 1

Сервіс автоматизованої перевірки кваліфікаційних випускних робіт на плагіат

КПІ ім.Ігоря Сікорського
Кафедра АСОІУ
гр. ІП-61



					КПІ.ІП-6102.045420.09.99.СС	
Зм.	Арк.	№ документа	Підпис	Дата	Схема структурна компонентів програмного забезпечення	Літера / Маса / Масштаб
Розробив		Блануца Д. С.				
Перевірив		Стеценко І. В.				Аркуш 1 / Аркушів 1
Т. кон.						
Н. кон.		Ліщук К.І.			Сервіс автоматизованої перевірки кваліфікаційних випускних робіт на плагіат	КПІ ім.Ігоря Сікорського Кафедра АСОІУ гр. ІП-61
Затвердив		Стеценко І. В.				