НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”
ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ
Кафедра автоматизованих систем обробки інформації і управління

«**До захисту допущено**»
В.о. завідувача кафедри
_____ __*О.А.Павлов*__
(підпис) (ініціали, прізвище)
“___” _____ 2019 р.

Дипломний проект

на здобуття ступеня бакалавра

з напряму підготовки *6.050103 «Програмна інженерія»*

спеціальність *«Програмне забезпечення систем»*

на тему: *Програмне забезпечення агрегації наукових публікацій*

Виконав: студент 4 курсу, групи ІП-52

Тарасюк Марія Андріївна _____
(прізвище, ім’я, по батькові) (підпис)

Керівник *старший викладач Халус О.А.* _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Консультант з графічної документації *доц. к.т.н. Ліщук К.І.* _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент *доц. каф. АУТС, к.т.н., доц. Писаренко А.В.* _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

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

Київ – 2019 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”
ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ
Кафедра автоматизованих систем обробки інформації і управління

«**До захисту допущено**»
В.о. завідувача кафедри
_____ __*О.А.Павлов*__
(підпис) (ініціали, прізвище)
“___” _____ 2019 р.

Дипломний проект

на здобуття ступеня бакалавра

з напряму підготовки *6.050103 «Програмна інженерія»*

спеціальність *«Програмне забезпечення систем»*

на тему: *Програмне забезпечення агрегації наукових публікацій*

Виконав: студент 4 курсу, групи ІП-52

Тарасюк Марія Андріївна _____
(прізвище, ім’я, по батькові) (підпис)

Керівник *ст. викладач Халус О.А.* _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Консультант з графічної документації *доц. к.т.н. Ліщук К.І.* _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент _____ _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

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

Київ – 2019 року

Національний технічний університет України
“Київський політехнічний інститут ім. І.Сікорського”

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

Кафедра автоматизованих систем обробки інформації і управління
(повна назва)

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки 6.050103 «Програмна інженерія»

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ *О.А.Павлов*
(підпис) (ініціали, прізвище)
“___” _____ 2019 р.

ЗАВДАННЯ
НА ДИПЛОМНИЙ ПРОЕКТ СТУДЕНТУ

Тарасюк Марії Андріївни
(прізвище, ім’я, по батькові)

1. Тема проекту *Програмне забезпечення агрегації наукових публікацій*

керівник проекту Халус Олена Андріївна, ст. викладач
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджена наказом по університету від «23» квітня 2019 р. № 1181-с

2. Термін подання студентом проекту «3» червня *2019 року*

3. Вихідні дані до проекту

Технічне завдання

4. Зміст пояснювальної записки

1.Аналіз вимог до програмного забезпечення
2. Моделювання та конструювання програмного забезпечення
3. Аналіз якості та тестування програмного забезпечення
4. Впровадження та супровід програмного забезпечення

5. Перелік графічного матеріалу (із зазначенням обов'язкових кресленників, плакатів, презентацій тощо)

1.Схема стуктурна варіантів використання

2.Схема бази даних

3.Схема структурна компонентів

6. Консультанти розділів проекту

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

7. Дата видачі завдання «15» лютого *2019 року*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів проекту	Примітка
1	*Вивчення предметної області*	15.02.2019	
2	*Аналіз існуючих методів розв'язання задачі*	23.03.2019	
3	*Постановка та формалізація задачі*	25.03.2019	
4	*Аналіз вимог до програмного забезпечення*	03.04.2019	
5	*Моделювання програмного забезпечення*	10.04.2019	
6	*Оформлення пояснювальної записки*	21.05.2019	
7	*Подання ДП на попередній захист*	28.05.2019	
8	*Подання ДП рецензенту*	03.05.2019	
9	*Подання ДП на основний захист*	08.06.2019	

Студент _____ Тарасюк М.А.
(підпис)

Керівник проекту _____ Халус О.А.
(підпис)

ВІДОМІСТЬ ДИПЛОМНОГО ПРОЕКТУ

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проект	2	
2	А4	КПІ.ІП-5220.045440.01.81	Програмне забезпечення агрегації наукових публікацій. Пояснювальна записка	56	
3	А4	КПІ.ІП-5220.045440.02.91	Програмне забезпечення агрегації наукових публікацій. Технічне завдання	12	
4	А4	КПІ.ІП-5220.045440.03.51	Програма та методика тестування.	10	
5	А4	КПІ.ІП-5220.045440.04.34	Керівництво користувача.	5	
6	А3	КПІ.ІП-5220.045440.06.99	Схема структурна варіантів використання	1	
7	А3	КПІ.ІП-5220.045440.06.99	Схема бази даних	1	
8	А3	КПІ.ІП-5220.045440.06.99	Схема струкрурна компонентів	1	

АНОТАЦІЯ

Пояснювальна записка дипломного проекту складається з трьох розділів, містить 5 таблиць та 11 джерел – загалом 56 сторінок.

Об`єкт дослідження: Електронні наукові бібліотеки, як системи агрегації даних.

Мета дипломного проекту: узагальнення отриманих під час навчання знань процесом створення системи агрегації наукових публікацій.

В першому розділі було виконано аналіз предметної області, сформульовано вимоги до розроблюваної системи, проведено аналіз існуючих рішень.

У другому розділі було спроектовано структуру модуля для системи агрегації наукових публікацій. Описано процес тестування.

У третьому розділі описано розгортання та впровадження веб застосунку, а також наведено схему структурну розгортання.

У додатках наведено: опис програми, схема використання програмного забезпечення.

КЛЮЧОВІ СЛОВА: ЕЛЕКТРОННА НАУКОВА БІБЛІОТЕКА, ,МІКРОСЕРВІС, РОЗРОБКА, ОБРОБКА ДАНИХ**.**

ABSTRACT

The explanatory note of the diploma project consists of three sections, contains 5 tables and 11 sources - a total of 49 pages.

Object of research: Electronic scientific libraries as data aggregation systems.

The purpose of the diploma project: generalization of the knowledge obtained during the study process by creating a system of aggregation of scientific publications.

In the first section, the analysis of the subject area was made, the requirements for the developed system were formulated, the analysis of existing decisions was made.

In the second section, the structure of the module for the system of aggregation of scientific publications was designed. The testing process is described.

The third section describes the deployment and implementation of the Web application, as well as the structured deployment scheme.

The appendixes contain: description of the program, the scheme of use of the software.

KEYWORDS: ELECTRONIC SCIENTIFIC LIBRARY, MICROSERVICE, DEVELOPMENT, DATA PROCESSING.

ВМЕСТО ЭТОГО ЛИСТА ВСТАВИТЬ ЛИСТ ТИТУЛА ПОЯСНИТЕЛЬНОЙ ЗАПИСКИ

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ 10

ВСТУП 11

АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 13

1.1 Загальні положення 13

1.2 Змістовний опис і аналіз предметної області 13

1.3 Аналіз успішних IT-проектів 14

1.4 Аналіз вимог до програмного забезпечення 18

1.4.1 Розроблення функціональних вимог 18

1.4.2 Розроблення нефункціональних вимог 22

1.4.3 Постановка комплексу завдань модулю 23

1.5 Висновки по розділу 23

МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 24

2.1 Моделювання та аналіз програмного забезпечення 24

2.2 Архітектура програмного забезпечення 26

2.3 Конструювання програмного забезпечення 30

2.4 Аналіз безпеки даних 36

2.5 Висновки по розділу 40

АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 42

3.1 Аналіз якості ПЗ 42

3.2 Опис процесів тестування 46

3.3 Опис контрольного прикладу 48

ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 50

4.1 Розгортання програмного забезпечення 50

4.2 Робота з програмним забезпеченням 51

ВИСНОВКИ 52

ПЕРЕЛІК ПОСИЛАНЬ.....53

ДОДАТОК А ТЕКСТИ ПРОГРАМНОГО КОДУ.....54

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

DB (Database) - база даних

SQL (Structured Query Language) - мова структурованих запитів

IDE (Integrated Development Environment) - інтегроване середовище розробки

JSON (JavaScript Object Notation) - запис об’єктів JavaScript

ВСТУП

На сьогоднішній день жодна людина не обходиться без інтернету. Часто постає питання пошуку певної інформації і неодноразово ми зіштовхуємося з проблемами доступу до матеріалів. Особливо яскраво така проблема виражається під час пошуку наукових публікацій. Для студентів та вчених велике значення має пошук пошук, потрібної їм публікації в найкоротші терміни. Оптимальним рішенням такої проблеми є використання електронних бібліотек для пошуку матеріалів для написання рефератів, курсових проектів, дипломних та наукових робіт.

Світ наукових публікацій зараз охоплює 28000 рецензованих журналів, 9 мільйонів дослідників і порядка 2,5 млн. винаходів в рік. Для сучасного дослідника дуже важлива можливість знаходження значних зв’язків між минулими і теперешніми дослідженнями. Щоденно тисячі наукових і дослідницьких установ за допомогою глобальної павутини знаходять нові підключення в спільну справу дослідження.

Електронні наукові бібліотеки можна назвати посередником між студентами та вченими, що займаються збором наукової інформації. У багатьох користувачів такими бібліотеками виникають проблеми з отриманням повної інформації з баз даних

Основною ціллю є створення системи агрегації наукових публікацій для студентів, університетів та науковців.

Мета розробки - узагальнення отриманих під час навчання знань процесом створення системи агрегації наукових публікацій.

Призначення розробки - використання проекту університетами, студентами та дослідниками для швидкого пошуку інформації різної наукової тематики для написання дисертацій, дипломних та наукових робіт.

Задачі:

– скачування pdf документа;
– можливість перегляду усіх скачаних документів;
– пошук статей за ключовими словами;
– систематизація, швидкий пошук по базі даних.

Цілі:

– створити онлайн бібліотеку наукових публікацій;
– надати користувачеві інтерфейс для пошуку наукових публікацій за ключовими словами;
– надати можливість розширеного пошуку;
– створити базу даних, де міститимуться публікації усіх університетів, для зручного використання її у майбутньому.

АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Загальні положення

Програмний продукт повинен бути написаний на мові, що дозволить запустити програму на операційній системі Linux. За рахунок великих об’ємів даних, для швидкого та зручного пошуку повинен використовуватися ElasticSearch, який проіндексує усі дані. Для передачі задач у чергу повинен використовуватися RabbitMq. Програмний продукт повинен підтримувати HTTP запити по протоколу Rest, для надання команд про запуск процесів.

Для забезпечення гнучкості та масштабування програма повинна бути побудована на основі мікросервісної архітектури. Кожен окремий компонент повинен відповідати певній задачі, що спрощує будь-які зміни продукту у майбутньому.

1.2 Змістовний опис і аналіз предметної області

Розвиток інформаційних технологій і комп’ютерної техніки послужило початком розвитку суспільства, в якому велика кількість працівників займаються пошуком, збором, збереженням, обробкою і наданням інформації. З’явилося багато аналогів: електронна бібліотека, цифрова бібліотека, віртуальна бібліотека. Основною задачею бібліотеки є залучення молоді до читання і появи в бібліотечній діяльності інформаційних технологій. Нові технології не заміняють традиційних книг, а навпаки дозволяють залучати нових читачів і стимулюють до науково-дослідницької діяльності. Можна сказати, що електронна бібліотека - інформаційна система, яка містить фонд електронних документів, який формується в залежності з заданими критеріями, призначений для загального використання.

Для студентів та вчених велике значення має пошук пошук, потрібної їм публікації в найкоротші терміни. Оптимальним рішенням такої проблеми є використання електронних бібліотек для пошуку матеріалів для написання рефератів, курсових проектів, дипломних та наукових робіт.

1.3 Аналіз успішних ІТ-проектів

Серед онлайн бібліотек та пошукових систем на просторах інтернету було знайдено наступних конкурентів: Архів TarraNova, Альдебаран, Електронна бібліотека української літератури університета Торонто, Google Play та App Store, Internet Archive, портал національної бібліотеки України імені Вернадського, Чтиво.

Якщо говорити про Архів TarraNova (рисунок 1.1), то це архів перекладів та авторських текстів. Відомих книг тут дуже мало, але знайти щось корисне можна. Керівництво просить називати себе архівом, а не бібліотекою, саме через те, що всі тексти розміщені офіційно і зі згодою автора. На відміну від цього проекта, мій додаток містить значно більше наукових статей, оскільки співпрацює не на пряму з авторами, а з університетами, які вже мають дозвіл на публікацію від різних науковців[1].

Рисунок 1.1 - Інтерфейс TarraNova

Альдебаран (рисунок 1.2), - літературний сервіс з великою кількістю колекцій творів. На сайті можна не тільки читати без регістрації онлайн, а й скачати книгу в будь-якому зручному для себе форматі. Перед скачуванням є функція ознайомитися з коротким змістом та можливість прочитати початок твору. У моєму сервісі є в наявності не тільки літературні твори, а й наукові публікації. Також надається можливість скачування документа в pdf форматі, короткий перегляд змісту. Окрім цього система знаходить подібні до змістового наповнення статті за ключовими словами, що збільшує вибір для читання[2].

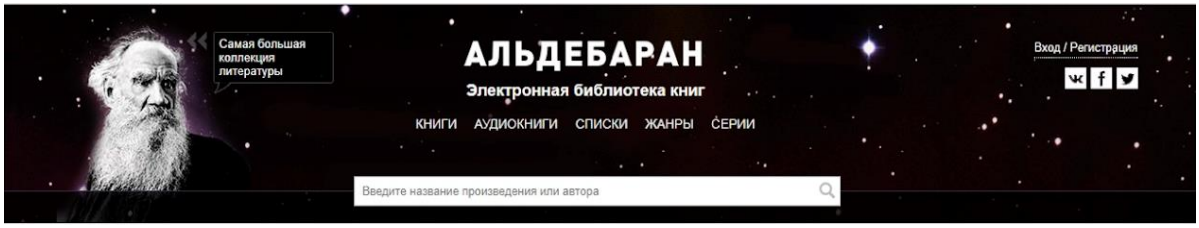


Рисунок 1.2 - Інтерфейс Альдебаран

Електронна бібліотека української літератури університета Торонто (рисунок 1.3), - особистий проект професора університета Максима Тарнавського. Бібліотека була створена для забезпечення вільного та безкоштовного доступа до електронних текстів всім зацікавленим читачам, особливо студентам за межами України, де доступ україномовних текстів обмежений. Мінусом цієї системи є те, що скачувати документи не є можливим, лише перегляд їх онлайн. Окрім цього, пошук потрібної статті відбувається довго по міркам комп'ютерного часу. Моя електронна бібліотека надає можливість скачати документ для зручного перегляду на будь-якому пристрої. Також за допомогою використання elasticsearch відбувається швидкий пошук через індексацію усіх статей під час їхнього унаслідування системою[3].

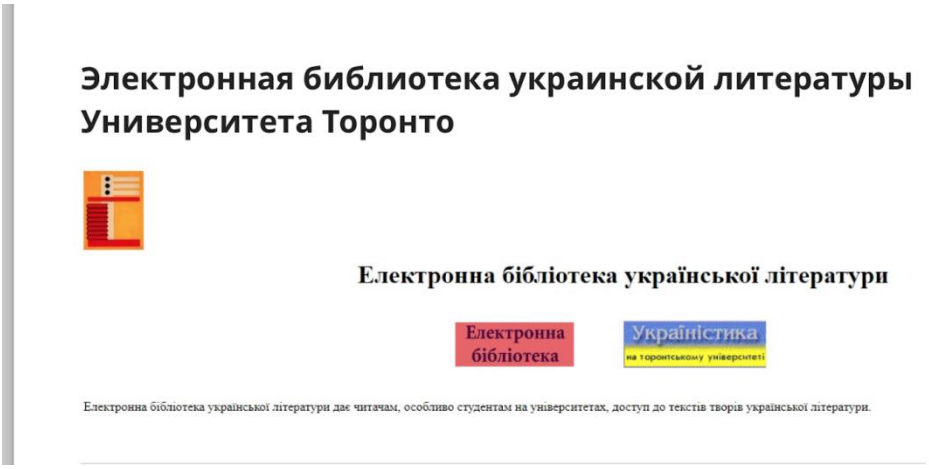


Рисунок 1.3 - Інтерфейс електронної бібліотеки української літератури університету Торонто

Google Play та App Store (рисунок 1.4), - в офіційних магазинах є як платні так і безкоштовні книги. Щоб отримати доступ до усіх матеріалів наявних у цій системі потрібно оформити платну підписку. Моя електронна бібліотека створена з метою спрощення роботи усім науковцям та студентам, саме через це доступ до усіх статей є безкоштовний для будь-якого користувача[4].

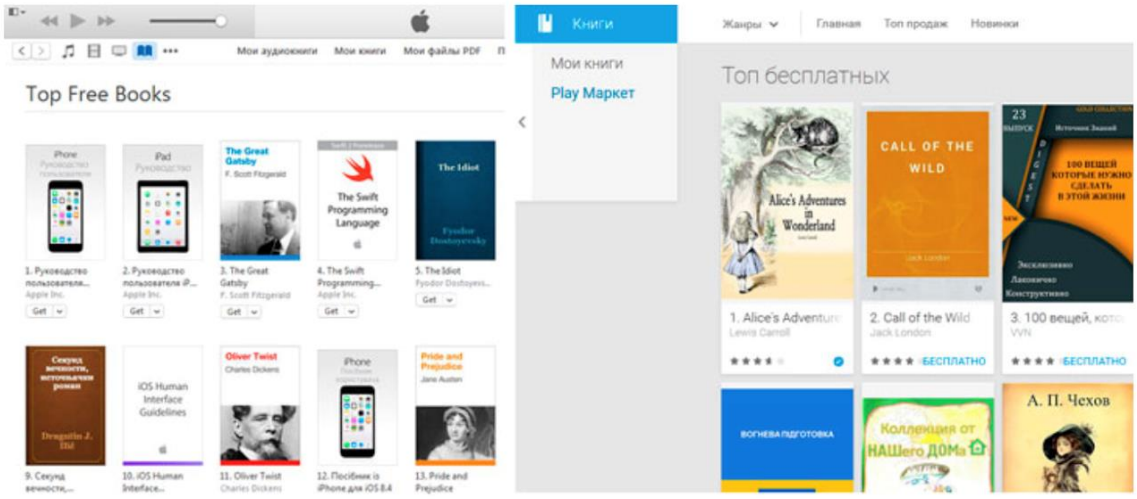


Рисунок 1.4 - Інтерфейс Google Play та App Store

Internet Archive (рисунок 1.5), - найбільша англомовна база безкоштовних цифрових книг. Портал зібрав більше 3 млн текстів, мільйон відео та аудіокниг.

Дана система підтримує тільки англомовні книжки та публікації в той час як моя система дозволяє знаходити книжки будь-якою мовою[5].

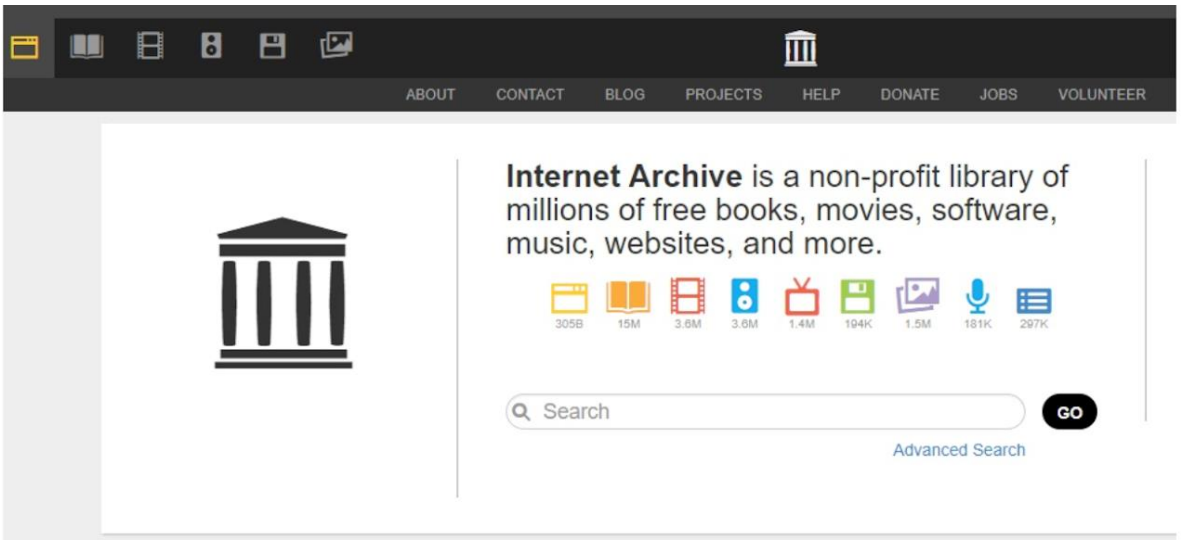


Рисунок 1.5 - Інтерфейс Internet Archive

Портал національної бібліотеки України імені Вернадського (рисунок 1.6), - відома бібліотека України з найбільшою базою даних наукової літератури. Послуги, які надає дана сторінка - онлайн-пошук документів та інформаційних ресурсів, доступ до електронних текстових інформаційних ресурсів, розміщених у вільному інтернет-доступі, а також отримання справочно-консультаційної інформації[6].

Рисунок 1.6 - Інтерфейс порталу національної бібліотеки України імені Вернадського

Чтиво (рисунок 1.7), - бібліотека книг як для рядового читача так і для людини закоханої в літературу. Всі твори публікуються лише з дозволу авторів,

а сам архів постійно наповнюється. Для кожного автора можуть відображатися три види творів: особисті твори, написані автором; твори,які цей автор перевів на українську мову, а також твори зв'язані з цим автором. Тексти зберігаються в різних форматах: fb2, * .epub, * .txt, * .htm, * .rtf, * .doc, * .pdf и * .djvu[7].

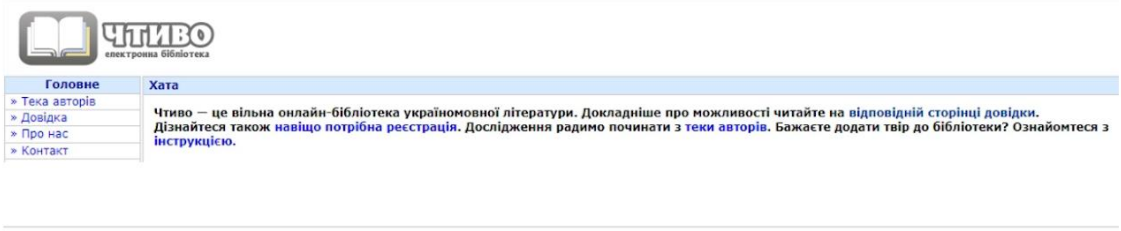


Рисунок 1.7 - Інтерфейс Чтиво

1.4 Аналіз вимог до програмного забезпечення

1.4.1 Розроблення функціональних вимог

Необхідно розробити програмний продукт, що буде давати можливість обробки електронних наукових робіт. Схему варіантів використання зображено нижче (рисунок 1.8). В системі передбачено наступні варіанти використання:

– збір ресурсів, які містять наукові роботи;

– отримання документів з зовнішнього ресурсу;

– збереження документу;

– перетворення документу - для забезпечення автоматизованої обробки документів, необхідно реалізувати перетворення роботи в оптимальний для обробки формат (наприклад xml);

– систематизація і пошук документів - необхідно забезпечити узгоджену структуру збереження наукових робіт та швидкий пошук в заданій структурі;

– перегляд документів - можливість переглянути вміст документу в системі.

Таблиця 1.1 - Варіант використання UC01

Назва	Збір ресурсів, які містять наукові роботи
Опис	систематизація та збереження зовнішніх ресурсів, для подальшої обробки в системі
Учасники	Адміністратор, система
Передумови	Наявність списку зовнішніх ресурсів у адміністратора
Постумови	Список зовнішніх ресурсів збережений у системі
Основний сценарій	Адміністратор отримує список ресурсів Адміністратор отримує доступ до бази даних Ресурс збережені в сховищі
Розширення сценаріїв	Для забезпечення простоти та коректності роботи необхідно реалізувати програмний інтерфейс для додання ресурсів

Таблиця 1.2 - Варіант використання UC02

Назва	Отримання документів з зовнішнього ресурсу
Опис	Забезпечення завантаження всіх існуючих наукових робіт з публічного ресурсу
Учасники	Система
Передумови	Наявність списку зовнішніх ресурсів у базі даних
Постумови	Документи завантажені
Основний сценарій	Системи читає список зовнішніх ресурсів Система перевіряє зовнішні ресурси на наявність нових документів Система завантажує документи
Розширення сценаріїв	

Таблиця 1.3 - Варіант використання UC03

Назва	Збереження документу
Опис	Забезпечення збереження документів для подальшої обробки в системі
Учасники	Система
Передумови	Завантажені документи
Постумови	Документи збережені
Основний сценарій	Системи отримує доступ до бази даних Система зберігає документи до бази даних
Розширення сценаріїв	

Таблиця 1.4 - Варіант використання UC04

Назва	Перетворення документу
Опис	Забезпечення автоматизованої обробки документів, перетворення роботи в оптимальний для обробки формат (наприклад xml)
Учасники	Система
Передумови	Завантажені документи
Постумови	Документи оброблені
Основний сценарій	Системи отримує список необроблених документів Система виконує перетворення документу в інший формат Система зберігає файли до бази даних

Таблиця 1.5 - Варіант використання UC05

Назва	Систематизація і пошук документів
Опис	Забезпечення узгодженої структури збереження наукових робіт та швидкий пошук в заданій структурі
Учасники	Система, користувач
Передумови	Завантажені документи
Постумови	Документ знайдений
Основний сценарій	Система виконує індексування документів Користувач надсилає запит на пошук Система виконує пошук по заданому критерію Користувач отримує результат пошуку

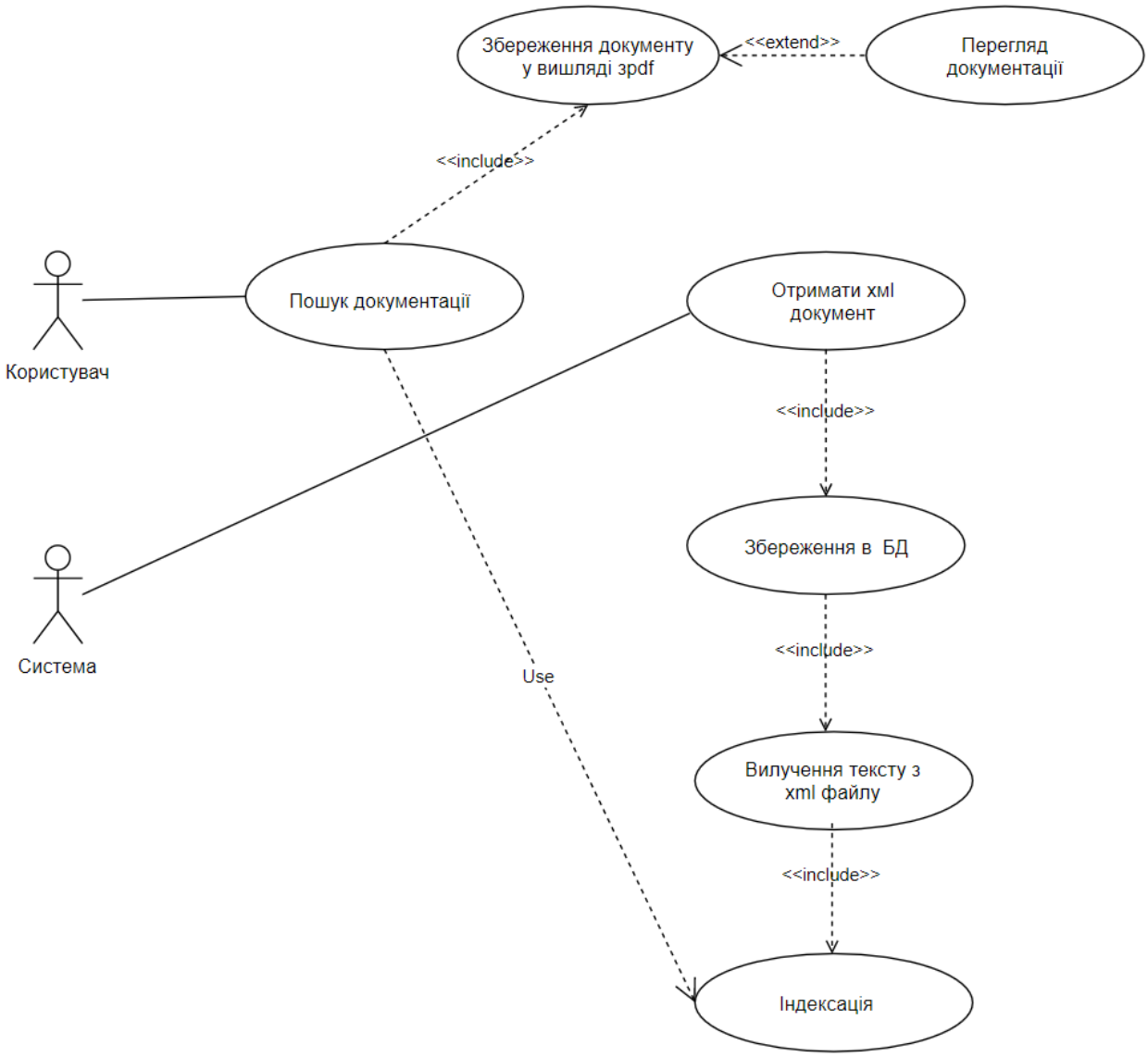


Рисунок 1.8 - Схема структурна варіантів використання

1.4.2 Розроблення нефункціональних вимог

– розробка повинна проводитись на мові Java з використанням фреймворку ElasticSearch для забезпечення маштабованого пошуку;

– повинен бути створений програмний інтерфейс, для роботи з системою за допомогою REST технологій;

– повинне бути налаштоване регулярне резервне копіювання бази документів.

1.4.3 Постановка комплексу завдань модулю

Отже, розроблювана система повинна виконувати наступні завдання:

– регулярний збір та структурування наукових робіт з вказаних публічний ресурсів (репозиторіїв);

– швидкий пошук по документам.

1.5 Висновки по розділу

В даному розділі було розглянуто поняття електронної бібліотеки наукових робіт. Також було проведено аналіз існуючих ресурсів отримання електронних документів. Як висновок, існує багато програмного забезпечення, яке призначене для агрегації наукових робіт, але деякі методи та властивості в них відсутні. Також жоден з розглянутих ресурсів не може надати повної бібліотеки наукових робіт.

Необхідна розробка нового програмного забезпечення, яке реалізує відсутні методі і створить можливості для інтеграції з системою розподіленої обробки даних.

МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Моделювання та аналіз програмного забезпечення

Розроблена система працює під операційними системами на базі ядра MacOs. Використання під операційними системами Windows та Linux можливе, але не було протестовано.

Програмна система повинна мати Web-сервіс для зручного користування нею. Отже необхідні наступні компоненти:

– серверне середовище;
– база даних;
– фреймворк для створення Web додатку.

В якості серверного середовища буде використана мова програмування Java. Для спрощення створення застосунку також буде використаний фреймворк Spring Boot[8]. Переваги Spring Boot:

– легко використовується для розвитку програми на основі Spring з Java або Groovy Spring;
– мінімізує час розвитку і піднімає продуктивність;
– уникає написання багатьох кодів прототипу (boilerplate), Annotations і конфігурації XML;
– легко дозволяє вам взаємодіяти з додатками Spring Boot с екологічними системами Spring як Spring JDBC, Spring ORM, Spring Data, Spring Security і т.д;
– слід підходу "Принципи конфігурації за замовчуванням" щоб мінімізувати час і старання, вкладені для розвитку додатків;

– забезпечує вбудований Server (Embedded HTTP servers) як Tomcat, Jetty .... щоб швидко і легко розвивати і тестувати веб-додатки;

– надає інструменти CLI (Command Line Interface) для розвитку і тестування додатків Spring Boot (Java або Groovy) з командних рядків (command prompt) дуже легко і швидко;

– забезпечує багато плагінів для швидкого розвитку та тестування програми Spring Boot використовуючи інструменти Build, як Maven і Gradle;

– пропонує багато плагінів для легкої роботи з контейнерами вбудованими базами даних (embedded database) і базами даних зберігаються в пам'яті (in- memory Databases).

Для забезпечення горизонтального маштабованого пошуку буде використовуватись фреймворк ElasticSearch[9]. Переваги:

– легка маштабованість на великих об’ємах даних;

– реплікація даних;

– висока швидкість роботи;

– зручне пошукове ядро;

– наявність добре продокументованого API.

В якості СУБД буде використана MySQL[10]. MySQL – вільна система керування реляційними базами даних. MySQL був розроблений компанією «TcX» для підвищення швидкодії обробки великих баз даних. Переваги:

– простота у встановленні та використанні;

– підтримується необмежена кількість користувачів, що одночасно працюють із DB;

– кількість рядків у таблицях може досягати 50 млн.;

– висока швидкість виконання команд;

– наявність простої і ефективної системи безпеки.

2.2 Архітектура програмного забезпечення

Архітектура програмного забезпечення – спосіб структурування програмної або обчислювальної системи, абстракція елементів системи на певній фазі її роботи. Система може складатись з кількох рівнів абстракції, і мати багато фаз роботи, кожна з яких може мати окрему архітектуру [5].

Архітектура програмного забезпечення включає в себе:

– вибір структурних елементів, за допомогою яких створена система, а також їх поведінку при взаємодії;

– поєднання вибраних структурних компонентів та їх поведінки у більші системи;

– архітектурний стиль.

Найпоширенішими прикладами архітектурних моделей і стилів є архітектура класної дошки, клієнт-серверна архітектура, розподілені обчислення, “peer-to-peer” архітектура.

Для розробки даного дипломного проекту було вирішено використати клієнт-серверну архітектуру, так як її було визначено як найоптимальнішу для поставленої задачі.

«Клієнт-сервер» – обчислювальна або мережева архітектура, в якій задачі або мережеве навантаження розподілені між постачальниками послуг, або серверами, та замовниками послуг, або клієнтами.

Ця архітектура включає в себе такі основні складові (дивись рисунок 2.1):

– набір серверів. Сервери надають інформацію або інші послуги програмам, що до них звертаються.

– набір клієнтів. Клієнти звертаються до серверів та використовують надані ними дані.

– мережа. Забезпечує взаємодію між клієнтами і серверами[12].

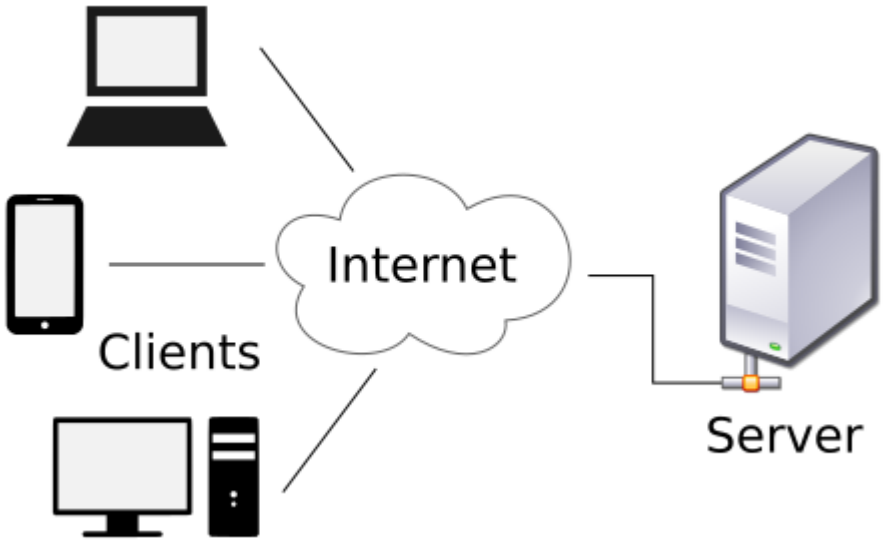


Рисунок 2.1 - Клієнт-серверна архітектура

Фактично клієнт та сервер – це програмне забезпечення. Зазвичай ці програми знаходяться на різних обчислювальних машинах та взаємодіють між собою за допомогою мережевих протоколів, проте вони також можуть розміщуватись і на одній машині, в цьому випадку сервер називають локальним.

Клієнтські програми відправляють програмам-серверам запити та отримують їх ресурси у вигляді даних (наприклад, завантаження файлів за допомогою HTTP, FTP, робота з базами даних) або сервісних функцій (робота з електронною поштою, перегляд веб-сторінок у всесвітній павутині).

Клієнт-серверна модель визначається перш за все розподілом обов’язків між клієнтом та сервером. Виділяють три групи функції, що орієнтовані на розв'язання різноманітних підзадач:

– функції вводу та відображення даних (забезпечують взаємодію з користувачем);

– прикладні функції (реалізують алгоритм рішення конкретної задачі);

– функції керування ресурсами (забезпечують доступ до необхідних ресурсів).

До основних переваг клієнт-серверної архітектури відносять:

– відсутність дублювання коду програми-сервера програмами-клієнтами;

– зниження вимог до комп’ютерів з програмами клієнтами, так як всі обчислення виконуються на сервері;

– всі дані зберігаються на сервері, який захищений набагато краще більшості клієнтів. На сервері простіше реалізувати організацію контролю повноважень для того, щоб надавати доступ до даних лише клієнтами, що мають відповідні права доступу.

Проте така модель має й недоліки:

– у випадку відмови серверу вся обчислювальна система може виявитись непрацездатною;

– щоб підтримувати роботу такої системи, потрібен окремий спеціаліст – системний адміністратор;

– вартість обладнання є досить високою.

Одним з ефективних різновидів архітектури «клієнт-сервер» є багаторівнева архітектура. В такій моделі функція обробки даних винесена на

один або кілька окремих серверів, що дозволяє розділити функції зберігання, обробки і представлення даних для більш ефективного використання можливостей серверів та клієнтів. Дана архітектура є значно складнішою в реалізації, проте вона забезпечує високі ступінь гнучкості та масштабування, безпеку і продуктивність.

Схему архітектурних компонентів зображено за наступному рисунку (рисунок 1.8).

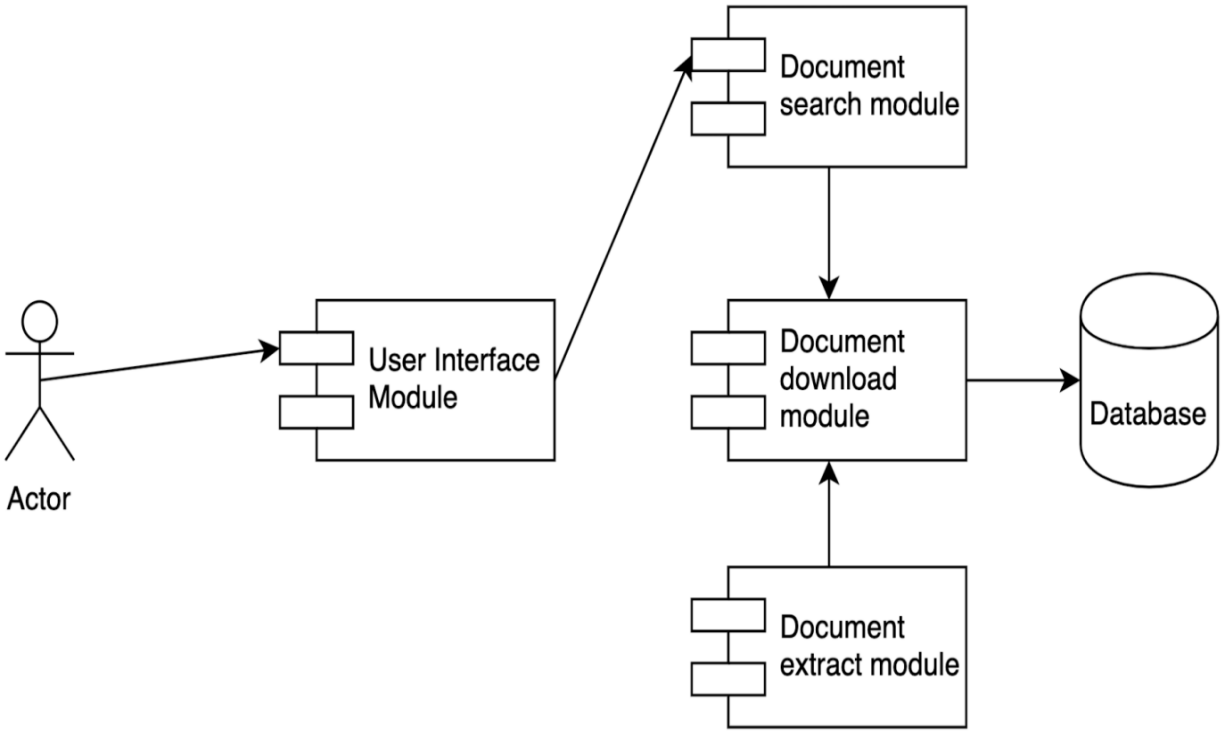


Рисунок 2.2 - Схема архітектурних компонентів

2.3 Конструювання програмного забезпечення

Таблиця 2.1 – Опис інтерфейсів системи

Інтерфейс	Опис
FilesystemDAO	Інтерфейс, який використовується для роботи з файлами
RepositoriesDAO	Інтерфейс, який використовується для роботи з репозиторіями
RepositoryDocumentDAO	Інтерфейс, який створений для роботи зі скачаними pdf документами

Таблиця 2.2 – Опис класів(структур) системи

Клас(структура)	Опис
QueueWorker	Абстрактний клас, що містить головну логіку усіх воркерів
MetadataDownloadWorker	Клас, що відповідає за скачування xml файла
ExtractMetadataWorker	Клас, що забезпечує наповнення бази даних з попередньо-скачаного xml файла
DocumentDownloadWorker	Клас, що забезпечує скачування pdf документа на локальну машину або на сервер
IndexItemWorker	Клас, що забезпечує індексацію усіх документів, що дозволяє здійснювати швидкий пошук серед наявних
TextExtractWorker	Клас, що забезпечує вилучення тексту в текстовий документ з pdf документа

Продовження таблиці 2.2

HardDiskFilesystemDAO	Клас, що використовується для роботи з файлами
OaiPmhDownloaderService	Клас, що містить логіку роботи з OAI-PMH endpoint, які надаються сторонніми сервісами для інтеграції з їхніми системами і міграції їхніх даних
MySQLRepositoriesDAO	Клас, створений для роботи з репозиторіями
OaiPmhMetadataRawWriteService	Клас, який працює OAI-PMH endpoint
WorkerStatus	Клас, який працює зі станами задачі, що запущена на виконання в даний момент або була запущенаа у минулому
FilesystemConfiguration	Клас, що містить конфігурацію та всі налаштування з файлами
ExtractMetadataFactoryService	Клас, фабрика куди надходять усі задачі по вилученню даних в БД, працює з усіма поставленими задачами.
DownloadMetadataFactoryService	Клас, що визначає яке API нам надала інша сторона, та обирає, який сервіс надалі працюватиме
DownloadMetadataTaskItemStatus	Клас, який працює зі статусами виконання скачування pdf документа
MySQLRepositoryDocumentDAO	Клас, який працює зі скачаними pdf документами
TextExtractionApplicationConfiguration	Клас, який містить конфігурацію та налаштування для діставання тексту

Продовження таблиці 2.2

ItemIndexConfiguration	Клас, що містить містить конфігурацію та налаштування для індексації усіх скачаних документів
QueueInfoService	Клас, який містить інформацію про стан усіх запущених на виконання задач
Worker	Абстрактний клас, що задає поведінку усім воркерам
WorkerProgress	Клас, що відображає прогрес кожного запущеног

Таблиця 2.3 – Опис методів класів та інтерфейсів системи

Клас/Інтерфейс	Метод	Опис
MetadataDownloadWorker	collectData()	Збирає дані необхідні для скачування xml
MetadataDownloadWorker	collectStatistics()	Збирає дані після того як xml документ було скачано
MetadataDownloadWorker	process()	Запускає процес скачування xml документу
ExtractMetadataWorker	collectData()	Збирає дані необхідні для вилучення даних до бази даних
ExtractMetadataWorker	collectStatistics()	Збирає статистику після того, як усі дані були вигружені до бази даних
ExtractMetadataWorker	process()	Запускає процес записування даних до бази даних

Продовження таблиці 2.3.

DocumentDownloadWorker	collectData()	Збирає необхідні дані для скачування pdf документа
DocumentDownloadWorker	collectStatistics()	Збирає статистику після того, як pdf документ був скачаний
DocumentDownloadWorker	process()	Запускає процес скачування pdf документа
FilesystemDAO	makeDirectory(String path)	Створює новий файл або папку за вказаним шляхом path
FilesystemDAO	compress(File file)	Зжимає вказаний файл з метою економії пам’яті
FilesystemDAO	deleteFile(File file)	Видаляє вказаний файл
FilesystemDAO	moveFile(String srcPath, String dstPath)	Переміщає файл за вказаним шляхом у місце, яке відповідає новому шляху
FilesystemDAO	createSymbolicLink()	Створення ярлика задля легкого пошуку у системі
FilesystemDAO	getMetadataPath()	Знаходження шляху за яким має зберігатися той чи інший документ

Продовження таблиці 2.3.

OaiPmhDownloaderService	downloadMetadata(OutputStream outputStream, DownloadMetadataTaskItemStatus downloadMetadataTaskItemStatus)	Скачує метадані для певного репозиторія
oaiPmhMetadataRawWriteService	harvest()	Скачує xml файл за певним алгоритмом в спеціально виділене місце
IndexItemWorker	process(TaskDescription taskDescription)	Індексує усі наявні документи для швидкого пошуку у системі
IndexItemWorker	taskReceived()	Збирає дані необхідні для індексації документів
TextExtractWorker	taskReceived()	Збирає дані необхідні для вилучення тексту з усіх наявних pdf документів до текстових документів
TextExtractWorker	process(TaskDescription taskDescription)	Вилучає текст з усіх наявних pdf документів до нових текстових документів
DownloadMetadataFactoryService	createDownloader(Integer repositoryId, Date fromDate)	Обирає який сервіс працюватиме з даним репозиторієм та визначає як саме скачуватиметься документ
MySQLRepositoryDocumentDAO	getRepositoryDocumentById(final Integer articleId)	Знаходження документа з певним айді для певного репозиторія

Продовження таблиці 2.3.

MySQLRepositoryDocumentDAO	getDocumentLanguage(Integer id)	Отримання інформацію про мову на якій написаний даний документ певного репозиторія
MySQLRepositoryDocumentDAO	addDocument(final Long updateId, final Integer repositoryId)	Додавання нового документа до усіх існуючих певного репозиторія
MySQLRepositoryDocumentDAO	getRepositoryDocumentsByRepositoryId(final Integer repositoryId, final Integer status)	Отримання усіх документів певного репозиторія
RabbitConfiguration	rabbitAdmin(ConnectionFactory connectionFactory)	Метод що нада\ можливість налаштувати конфігурацію та роботу з RabbitMQ сервісом
LanguageDAO	insertLanguageForDocument(Integer documentId, String language)	Встановлення та визначення мови наукової публікації.
QueueInfoService	getCountMessages(String queueName)	Отримання кількості повідомлень, які знаходяться в черзі на обробку та виконання якимось з існуючих воркерів.
ReportingWorker	generateReport(List<TaskItemStatus> results, boolean taskOverallSuccess)	Створення звіту про успішність або неуспішність виконання якихось з існуючих задач, а також збереження інформіції про помилку, якщо така була виявлена, в БД

2.4 Аналіз безпеки даних

Важливим етапом під час розробки проекту є включення в систему засобів безпеки. Всі загрози веб-застосунків можна розділити на три групи [19]:

– порушення конфіденційності. Викрадення конфіденційної інформації, зокрема даних клієнтів, партнерів.

– порушення цілісності. Поширення шкідливого програмного забезпечення і забороненого контенту.

– порушення доступності. Видалення вмісту, DoS та DDoS атаки.

Для того, щоб захистити сервіс перш за все потрібно включити в систему автентифікацію – процес перевірки справжності користувача. Також обов’язковим моментом є валідація усіх даних, що отримуються від користувача, адже будь-який користувач може виявитись зловмисником. Наступним необхідним кроком є реалізація захисту від SQL-ін’єкцій.

Автентифікація

В будь-якій системі автентицікації зазвичай можна виділити кілька елементів [20]:

– суб’єкт – той, хто буде проходити процедуру (авторизований користувач).

– характеристика суб’єкта – відмінна риса (таємний пароль).

– власник системи автентифікації – той, хто несе відповідальність та контролює роботу системи.

– механізм автентифікації – принцип роботи системи (програмне забезпечення, яке перевіряє пароль).

Механізм керування доступом – надає визначені права доступу суб’єкту (процес реєстрації, керування доступом).

Розглянемо автентифікацію за допомогою віртуальних приватних мереж, що реалізована в даному дипломному проекті.

Віртуальні приватні мережі (VPN) можуть надавати додатковий рівень безпеки та конфіденційності. Якщо ви працюєте в громадській мережі Wi-Fi і хочете уникнути цікавих очей, або ви турбуєтеся про конфіденційність в цілому, VPN може запропонувати багато переваг.

Коротко кажучи, VPN встановлює безпечне, зашифроване з'єднання між вашим пристроєм і приватним сервером, приховуючи ваш трафік від перегляду іншими. Звичайно, сам VPN може бачити ваш трафік, тому ви повинні вибрати VPN від компанії, якій ви довіряєте.

Основними перевагами VPN є:

– VPN - це найпростіше рішення у всіх випадках, коли необхідно створити або отримати доступ до мережі через економічну, ізольовану, безпечну приватну мережу через Інтернет.
– VPN дозволяє використовувати існуючу централізовану інфраструктуру мережевої безпеки, щоб забезпечити єдиний захист від кіберзагроз у мережевих пристроях компанії незалежно від місцезнаходження
– VPN забезпечує безпечний доступ до необхідних внутрішніх послуг для мобільної робочої сили, що підвищує їх продуктивність
– VPN знижує ризик безпеки, дозволяючи доступ до певних мережевих ресурсів тільки користувачам, які мають дозвіл, шифруючи дані і тим самим захищаючи від небезпечного доступу Wi-Fi, і

забезпечуючи безперервність централізованого управління уніфікованими загрозами.

VPN може використовувати одну з багатьох технологій, таких як безпека протоколу Інтернету (IPsec), захист транспортного шару (SSL / TLS), безпеку передачі дейтаграмного транспорту (DTLS), для безпечного підключення пристроїв або мереж через загальнодоступні мережі. розширити або сформувати приватну мережу.

Таку ж технологію, що використовується для створення віртуальної зв'язку між мережами, також можна використовувати для підключення пристроїв користувача до приватної мережі. Поширене використання VPN полягає в тому, щоб забезпечити віддаленим співробітникам безпечний доступ через Інтернет до IT-послуг своєї компанії. Працівники використовують клієнти VPN, встановлені на корпоративних ноутбуках або мобільних пристроях для підключення до сервера VPN, який присутній у приватній мережі компанії.

Випадок використання віддаленого доступу не обмежується доступом для співробітників. Будь-який підключений до Інтернету пристрій може використовувати VPN як частину приватної мережі. Пристрої можуть варіюватися від звичайних обчислювальних пристроїв, таких як ноутбуки, до спеціалізованих промислових датчиків або побутової електроніки, таких як смарт-телевізори.

Валідація даних

Валідація вхідних даних грає важливу роль при проектуванні системи [22]. Необхідно захистити систему від вхідних даних, які можуть нашкодити системі. Не дивлячись на те, що більшість користувачів не загрожують розробленому застосунку, завжди є ймовірність виникнення бажання у зловмисника зіпсувати

систему, який може зробити це вводячи шкідливі дані через форму або рядок адреси.

Валідацію варто виконувати в тому місці, де дані приходять в систему [23]. Зазвичай це користувацький інтерфейс, через який вводяться дані, файл з налаштуваннями програми чи даними для програми. У випадку такої перевірки гарантовано, що програма отримує лише коректні дані і може надалі використовувати їх без додаткових перевірок.

Існує кілька способів перевірити вхідну інформацію:

- посимвольна перевірка. Виконується в користувацькому інтерфейсі під час введення даних.
- перевірка окремих значень. Це перевірка значення в окремому полі, що може виконуватись як під час введення, так і після його завершення.
- сукупність введених значень. Спочатку дані передаються в програму, після чого подається деякий сигнал, який ініціює їх обробку. Цей метод передбачає перевірку не лише значень, але й зв’язків між ними.
- перевірка стану системи після обробки даних. Дані обробляються з можливістю повернутися до початкового стану. Цей механізм називається транзакційним і використовується в тому випадку, коли не вдається виконати валідація безпосередньо вхідних даних.

У даному дипломному проекті реалізовано валідацію даних введених користувачем, вона передбачає перевірку усіх введених даних після подання деякого сигналу, який зазвичай є натисканням користувача на кнопку.

Захист від SQL-ін’єкцій

SQL-ін’єкції – одні з найпоширеніших і найнебезпечніших вразливостей в питання безпеки [24]. Вони дають змогу зловмисникам отримати необмежений доступ до вашої системи.

Ризик SQL-ін’єкції виникає щоразу, коли програміст створює динамічний запит до бази даних, який містить дані введені користувачем. Надійним та ефективним способом попередження SQL-ін’єкцій є заміна динамічних запитів підготовленими виразами, запитами з параметрами або збереженими процедурами. У випадку неможливості такої заміни необхідно ретельно перевіряти дані, що містяться у запитах.

В мові програмування PHP, що використовується для розробки дипломного проекту, для безпечної роботи з базами даних передбачено інструмент зв’язку з сервером бази даних PDO. Дане розширення містить у своєму функціоналі підготовлені вирази, іменовані та неіменовані плейсхолдери. Наприклад, використовуючи клас PDO і його метод bindParam(), можна встановити сувору типізацію даних, які передаються до бази даних.

Також PDO використовує для різних баз даних їх власні драйвери, що дозволяє досягнути високої продуктивності [25]. PDO підтримує велику кількість драйверів баз даних, що є дуже зручним у тому випадку, якщо з’явиться необхідність змінити базу даних. Змін потребуватиме лише невеликий фрагмент коду, який реалізує підключення до бази даних.

2.5 Висновки по розділу

В другому розділі на основі сформульованих вимог до функціональності системи, що розробляється, спроектовано:

– схема даних;
– багатошарова модульна структура системи;
– середовище для створення та розгортання Web сервісу;
– життєвий цикл запиту до підсистеми.

Також був визначений склад програмної системи, розглянути та обрані інструменти та засоби для реалізації системи агрегації наукових робіт.

АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Аналіз якості ПЗ

Якість ПЗ - це набір деяких характеристик, що впливають на здатість відповідати вимогам замовника продукту, які він відзначив у вигляді вимог до продукту що розробляється. Виділяють два процеси забезпечення якості програмного продукту впродовж циклу розробки відповідно до всесвітніх та вітчизняних стандартів оцінки рівня якості програмного продукту:

– гарантія якості ПЗ - це результуєче після певних дій на кожній стадії циклу розробки з підтвердження й перевірки відповідності програмного продукту стандартам та процедурам, орієнтованим на досягнення якості;

– інженерія якості як процес надання продуктам ПЗ надійності, супроводження й інших характеристик якості.

Ці процеси вимагають:

– вплив стандартів і процедур, які виникають при вивченні програм;

– перегляд управління, перегляду та супроводу умов програмного забезпечення, а також усієї проектної документації (звіти, звіти, огляди);

– контроль за виконанням фізичних перевірок та перевірок;

– аналіз і контроль програмного забезпечення тестування (оновлення).

Функціональність являє собою набір властивостей, що визначають здатність програмного забезпечення виконувати в даному середовищі впорядковані послідовності дій для задоволення визначених користувачем властивостей, упорядкованих користувачем, відповідно до вимог обробки і загальносистемних засобів. Атрибути функціональності програмного забезпечення:

– функціональна повнота - атрибут, що показує ступінь достатності основних функцій для вирішення спеціальних завдань відповідно до призначення програмного забезпечення;

– правильність - це атрибут, який показує, наскільки добре й послідовно досягнуті результати;

– сумісність або сумісність - атрибути, що вказують на здатність програмного забезпечення взаємодіяти з іншими системами та середовищами;

– безпека - атрибути, що вказують на можливість запобігання несанкціонованому доступу до програм і даних;

– узгодженість - атрибут, що вказує на відповідність встановленим стандартам, конвенціям, правилам, законам і нормативним актам.

Надійність - це набір атрибутів, які вказують на здатність програмного забезпечення правильно конвертувати введені дані в результати. Зниження надійності програмного забезпечення зумовлено помилками у вимогах, розробці та виконанні.

Атрибути надійності програмного забезпечення:

– відмовостійкість - атрибути, що визначають частоту відмов через помилки в програмному забезпеченні;

– допуск помилок - атрибути, що вказують на можливість виконання функцій в аномальних умовах (апаратний збій, помилки даних і інтерфейси, порушення дій оператора і т.д.);

– поновлення - атрибути, які вказують на здатність програми перезапускатись для повторного виконання та відновлення даних після збоїв;

– послідовність - атрибут, що показує відповідність діючим стандартам, конвенціям, правилам, законам і нормам.

Деякі типи систем (реального часу, радіолокації, безпеки, зв'язку, медичного обладнання тощо) містять конкретні вимоги, що забезпечують високу надійність з такими атрибутами, як недопустимість помилок, безпека, безпека та зручність використання додатків, а також надійність як Головний критерій надійності.

Зручність застосування - сукупність атрибутів, що характеризують умови взаємодії користувача з програмним забезпеченням. Атрибути зручності використання програмного забезпечення:

– чіткість - визначено, наскільки зрозумілі логічні концепції програмного забезпечення та їх застосування;

– простота навчання - визначити, наскільки легко вивчити стан використання;

– ефективність - характеризується швидкістю реакції системи на дії користувача;

– узгодженість - визначається відповідність розробки вимогам існуючих стандартів, угод, правил, законів і правил;

Ефективність - це зв'язок між результатами використання програмного забезпечення та кількістю задіяних ресурсів (обладнання, матеріали, послуги обслуговуючого персоналу тощо).

Спостереження - це зусилля, які потрібно витратити на налаштування, вдосконалення та адаптацію програмного забезпечення у випадку зміни умов, вимог чи функціональних специфікацій.

Атрибути супроводу програмного забезпечення:

– аналіз - індикатор, що визначає необхідні зусилля для діагностики причин збоїв або визначення частин, які необхідно модифікувати;

– мінливість - індикатор, що визначає зусилля змінити, усунути помилки або внести зміни через помилки або нові особливості операційного середовища;

– стабільність - атрибут, що характеризує ймовірність модифікації;

– тестування - це атрибут, який характеризує спроби перевірки та перевірки.

Портативність - це здатність програмного забезпечення адаптуватися до роботи в разі зміни середовища виконання.

В рамках аналізу даного продукту буде виконано тестування наступного функціоналу:

– збір ресурсів, які містять наукові роботи;
– отримання документів з зовнішнього ресурсу;
– збереження документу;
– перетворення документу;
– систематизація і пошук документів.

3.2 Опис процесів тестування

Всі види тестування програмного забезпечення, залежно від переслідуваних цілей, можна умовно розділити на наступні групи:

- функціональні (Functional testing);
- нефункціональні (Non-functional testing);
- пов’язані зі змінами (Regression testing).

Функціональні тести базуються на функціях і особливостях, а також взаємодії з іншими системами, і можуть бути представлені на всіх рівнях тестування: компонентному або модульному (Component / Unit testing), інтеграційному (Integration testing), системному (System testing) і приймальному (Acceptance testing).

Функціональні види тестування розглядають зовнішню поведінку системи. Далі перераховані одні з найпоширеніших видів функціональних тестів:

- функціональне тестування (Functional testing);
- тестування безпеки (Security and Access Control Testing);
- тестування взаємодії (Interoperability Testing).

Нефункціональне тестування описує тести, необхідні для визначення характеристик програмного забезпечення, які можуть бути виміряні різними величинами. В цілому, це тестування того, “Як” система працює. Далі перераховані основні види нефункціональних тестів:

- тестування навантаження (Performance and Load Testing);
- стресове тестування (Stress Testing);
- тестування стабільності або надійності (Stability / Reliability Testing);
- об’ємне тестування (Volume Testing);

– тестування установки (Installation testing);
– тестування зручності користування (Usability Testing);
– тестування на відмову і відновлення (Failover and Recovery Testing);
– конфігураційне тестування (Configuration Testing).

Після проведення необхідних змін, таких як виправлення бага / дефекту, програмне забезпечення повинне бути перетестоване для підтвердження того факту, що проблема була дійсно вирішена. Нижче перераховані види тестування, які необхідно проводити після установки програмного забезпечення, для підтвердження працездатності програми або правильності здійсненого виправлення дефекту:

– димове тестування (Smoke Testing);
– регресійне тестування (Regression Testing);
– тестування збірки (Build Verification Test);
– санітарне тестування або перевірка узгодженості / справності (Sanity Testing).

3.3 Опис контрольного прикладу

Таблиця 3.1 - Матриця залежності між вимогами застосунку і варіантами використання

	REQ001 Збереження даних про ресурси	REQ002 Завантаження документу використовуючи апи ресурсу	REQ003 Збереження завантаженого документу в базі даних	REQ004 Перетворення pdf документів в тест	REQ005 Забезпечення пошуку документів
UC001 Збір ресурсів	■				
UC002 Отримання документів	■	■			■
UC003 Збереження документу		■	■	■	■
UC004 Перетворення документу				■	
UC005 Пошук по документам					■

Методом компонентного тестування перевіримо окремі частині API, такі як:

– оброблювачі шляхів API;

– методи доступу до базі даних;

– методи доступу до зовнішніх ресурсів.

Вхідними даними є набори параметрів на яких очікується певний результат, що є вихідними даними тестування.

Методом інтеграційного тестування будуть перевірені взаємодії між модулями системи, такі як:

– взаємодія сервера та бази даних;
– взаємодія сервера та черги подій;
– взаємодія сервера та зовнішніх ресурсів, які містять документи.

Вхідними даними є набори повідомлень, що будуть передані від одного компоненту системи до іншого відповідно до конкретного тесту.

Методом тестування продуктивності буде перевірена швидкодія системи.

Вхідними даними є набори даних, що покривають усі варіанти роботи системи в конкретному випадку.

Критерієм проходження є успішне виконання кожного пункту тесту. Якщо хоч один з пунктів не був виконаний - все тестування вважається проваленим.

ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Розгортання програмного забезпечення

Для роботи програмного забезпечення необхідні наступні технічні вимоги:

– пристрій під керівництвом операційної системи Windows/MacOSX/Linux;

– 512 МБ оперативної пам’яті;

– 256 МБ дискового простору;

– браузер з доступом до інтернету;

– Java Runtime Environment версії 8 або вище.

Для запуску продукту необхідно запустити виконуваний файл (фай л додається до диплому). Запуск виконується за допомогою команди: *ja va -jar [шлях]/application.jar*, де *[шлях]* - шлях до директорії, де розміщений виконуваний файл.

Для запуску додатку з програмного коду на пристрої, де запускає ться програма повинен бути встановлений Java Development Kit версії 8 або вище. Для створення виконуваного файлу потрібно виконати кома нду: *maven -Dmaven.multiModuleProjectDirectory=[шлях] clean install*, де *[шлях]* - шлях до директорії, в якій розміщений програмний код продукту

4.2 Робота з програмним забезпеченням

Інструкція для роботи з програмним забезпеченням наведена в додатку Є до дипломного проекту.

ВИСНОВКИ

Дипломний проект виконувався в декілька етапів, які були оформ лені як окремі розділи.

В першому розділі було проаналізовано проблематику пошукових систем наукових публікацій, було визначено слабкі сторони усіх існуюч их систем, які були враховано під час розробки програмного забезепече ння. Окрім цього було проведено розробку нефункціональних та функці ональних вимог до розробленої програми.

В другому розділі було створено та проведено моделювання розро блюваного сервіса, повністю створена архітектура проекта, а також опис ані класи відповідно до вище описаної архітектури. Крім цього було р озроблено десктопний додаток.

Даний проект надає ряд наступних переваг:

– можливість скачування всіх документів з певного репозиторія;

– швидкий пошук по всім доступним документам за ключовими сло вами або назвою репизиторія;

– можливість перегляду усіх скачаних документів;

– створено величну базу даних, яка містить наукові публікації усіх доступних університетів.

В третьому розділі було було проведено аналіз якості продукту і було проведено його тестування. Було встановлено, що система працю є коректно і відповідає усім описаним вимогам.

В четвертому розділі було описано процес розгортання додатку на пристрої, усі необхідні умови для запуску програми, а також наведена інструкція користувача з усіма описаними кроками, як користуватися даною системою.

Разом з цим було розроблено схему класів та проектну документа цію.

ПЕРЕЛІК ПОСИЛАНЬ

1) TarraNova [Електронний ресурс]. – 2019. Режим доступу до ресурсу: http://tarranova.lib.ru/.

2) Альдебаран[Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://aldebaran.ru/

3) Бібліотека української літератури [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: http://sites.utoronto.ca/elul/Main-Ukr.html.

4) Google store [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://play.google.com/store

5) Internet Archive [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://archive.org/.

6) Портал національної бібліотеки України імені Вернадського [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: http://www.nbuv.gov.ua/.

7) Чтиво [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: http://chtyvo.org.ua/authors/letter/21/%D0%A1.

8) UML. Use case [Електронний ресурс] – Режим доступу до ресурсу: https://msdn.microsoft.com/en-us/library/dd409432.aspx

9) SpringBoot [Електронний ресурс] – Режим доступу до ресурсу: https://spring.io/projects/spring-boot

10) Elasticsearch [Електронний ресурс] – Режим доступу до ресурсу: https://www.elastic.co/products/elasticsearch

11) MySql [Електронний ресурс] – Режим доступу до ресурсу: https://dev.mysql.com/doc/

12) Клієнт серверна архітектура [Електронний ресурс] – Режим доступу до ресурсу: https://www.w3schools.in/what-is-client-server-architecture/

ДОДАТОК А ТЕКСТИ ПРОГРАМНОГО КОДУ

Тексти програмного коду

Програмне забезпечення агрегації наукових публікацій

CD-ROM

(Вид носія даних)

10 арк, 1488 Кб

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

```
package uk.ac.core.metadatadownloadworker.worker;

import com.google.gson.Gson;

import java.io.*;
import java.nio.file.Files;
import java.nio.file.Path;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Date;
import java.util.List;

import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import uk.ac.core.common.model.task.TaskItem;
import uk.ac.core.common.model.task.TaskItemStatus;
import uk.ac.core.common.model.task.parameters.RepositoryTaskParameters;
import uk.ac.core.database.service.repositories.RepositoriesDAO;
import uk.ac.core.filesystem.services.FilesystemDAO;
import uk.ac.core.issueDetection.collectors.MetadataDownloadIssueCollectorService;
import
uk.ac.core.metadatadownloadworker.worker.metadata.DownloadMetadataFactoryService;
import uk.ac.core.metadatadownloadworker.worker.metadata.IDownloadMetadata;
import
uk.ac.core.metadatadownloadworker.worker.taskStatus.DownloadMetadataTaskItemStatus;
import uk.ac.core.worker.QueueWorker;

/**
 *
 * @author mc26486
 */
public class MetadataDownloadWorker extends QueueWorker {
```

```
        private final org.slf4j.Logger logger =
LoggerFactory.getLogger(MetadataDownloadWorker.class);

        OutputStream outputStream = null;

        private Integer repositoryId;
        private RepositoryTaskParameters repositoryTaskParameters;
        private Date fromDate;

        @Autowired
        DownloadMetadataFactoryService downloadMetadataFactoryService;

        @Autowired
        FilesystemDAO filesystemDAO;

        @Autowired
        RepositoriesDAO repositoriesDAO;

        @Autowired
        private                                  MetadataDownloadIssueCollectorService
metadataDownloadIssueCollectorService;

        public MetadataDownloadWorker() {
        }

        @Override
        public List<TaskItem> collectData() {
          this.repositoryTaskParameters                =                new
Gson().fromJson(this.currentWorkingTask.getTaskParameters(),
RepositoryTaskParameters.class);
          this.repositoryId = this.repositoryTaskParameters.getRepositoryId();
          this.fromDate = this.repositoryTaskParameters.getFromDate();
          logger.info("fromDate set to : " + this.fromDate);
```

```
            // Pass in Fake taskItems as its not really needed for this task
            List<TaskItem> items = new ArrayList<>();
            items.add(new TaskItem());
            return items;
        }

        @Override
        public List<TaskItemStatus> process(List<TaskItem> taskItems) {

            DownloadMetadataTaskItemStatus downloadMetadataTaskStatus = new
DownloadMetadataTaskItemStatus();
            workerStatus.setTaskStatus(downloadMetadataTaskStatus);
            try {
              if (fromDate != null) {// if is incremental, incremental directory must be created
        //          logger.info("fromDate != null");

filesystemDAO.makeDirectory(filesystemDAO.getIncrementalFolder(repositoryId));
              }
              File metadataFinalLocation =
filesystemDAO.createPathToNewMetadataXmlFile(repositoryId);

              String metadataPath = metadataFinalLocation.getAbsolutePath();
              if (fromDate != null) {
                metadataPath = filesystemDAO.getMetadataPath(repositoryId, fromDate,
fromDate);
              }
              String metadataPathPart = filesystemDAO.getMetadataPathPart(repositoryId,
fromDate, fromDate);

              IDownloadMetadata downloader =
downloadMetadataFactoryService.createDownloader(repositoryId, fromDate);

              filesystemDAO.deleteFile(metadataPathPart);
              logger.info("Storing metadata file to: " + metadataPathPart);
```

```
                outputStream = new FileOutputStream(metadataPathPart);
                outputStream.write("<?xml version=\"1.1\" encoding=\"UTF-
8\"?>\n".getBytes("UTF-8"));
                outputStream.write("<harvest>\n".getBytes("UTF-8"));

                // Now download the metadata
                downloader.downloadMetadata(outputStream, downloadMetadataTaskStatus);

                try {
                   outputStream.write("</harvest>\n".getBytes("UTF-8"));
                   outputStream.close();
                } catch (IOException ex) {
                   logger.error(ex.getMessage());
                }

        filesystemDAO.moveFile(metadataPathPart, metadataPath);

                // If the current harvest is not incremental
                if (fromDate == null) {
                   // Compress location of old metadata file
                   File path =
filesystemDAO.getLatestMetadataPath(repositoryId).getAbsoluteFile();
                   if(path.exists()){
                      String readSymLink = Files.readSymbolicLink(path.toPath()).toString();
                      try {
                         filesystemDAO.compress(new File(readSymLink));
                      } catch (Exception ex) {
                         logger.error(ex.getMessage(), ex);
                      }
                   }
                }

                filesystemDAO.createSymbolicLink(repositoryId, metadataPath);
```

```
            } catch (Exception ex) {
                logger.error(ex.getMessage(), ex);
                downloadMetadataTaskStatus.setSuccess(false);
            }
            return Arrays.asList(new TaskItemStatus[]{downloadMetadataTaskStatus});
        }

        @Override
        public void collectStatistics(List<TaskItemStatus> results) {
            DownloadMetadataTaskItemStatus downloadMetadataTaskItemStatus =
(DownloadMetadataTaskItemStatus) results.get(0);
            if (downloadMetadataTaskItemStatus.getOaiPMHIssueDescriptions().size() > 0) {
                for (String oaiPMHIssueDescription :
downloadMetadataTaskItemStatus.getOaiPMHIssueDescriptions()) {

metadataDownloadIssueCollectorService.collectDownloadIssue(oaiPMHIssueDescription);
                }
            }
        }
    }

    package uk.ac.core.metadatadownloadworker.worker.metadata;

    import java.io.File;
    import java.io.FileNotFoundException;
    import java.io.OutputStream;
    import java.text.SimpleDateFormat;
    import java.util.Date;
    import java.util.logging.Level;
    import java.util.logging.Logger;
    import javax.xml.parsers.ParserConfigurationException;
    import javax.xml.transform.TransformerException;
    import org.slf4j.LoggerFactory;
    import org.springframework.beans.factory.annotation.Autowired;
```

```
import org.springframework.http.ResponseEntity;
import org.springframework.stereotype.Service;
import org.springframework.web.client.RestTemplate;
import org.xml.sax.SAXException;
import uk.ac.core.common.exceptions.CHARSException;
import uk.ac.core.common.model.legacy.LegacyRepository;
import uk.ac.core.common.model.task.TaskItemStatus;
import uk.ac.core.database.service.repositories.RepositoriesDAO;
import uk.ac.core.filesystem.services.FilesystemDAO;
import
uk.ac.core.metadatadownloadworker.worker.metadata.oaipmh.OaiPmhMetadataRawWriteService
;
import
uk.ac.core.metadatadownloadworker.worker.taskStatus.DownloadMetadataTaskItemStatus;
import uk.ac.core.metadatadownloadworker.worker.util.CleanUrl;
import uk.ac.core.supervisor.client.SupervisorClient;

/**
 *
 * @author Tomas Korec
 */
@Service
public class OaiPmhDownloaderService implements IDownloadMetadata {

    private final org.slf4j.Logger logger =
LoggerFactory.getLogger(OaiPmhDownloaderService.class);

    Integer repositoryId;
    TaskItemStatus status;
    LegacyRepository repository;
    @Autowired
    RepositoriesDAO repositoryDAO;
    @Autowired
    OaiPmhMetadataRawWriteService oaiPmhMetadataRawWriteService;
```

```
@Autowired
FilesystemDAO filesystemDAO;
@Autowired
SupervisorClient supervisorClient;

Date fromDate;
private final String FROM_DATE_FORMAT = "yyyy-MM-dd";

public void init(Integer repositoryId) {
   this.repositoryId = repositoryId;
   this.repository = repositoryDAO.getRepositoryById(String.valueOf(repositoryId));
}

public void init(Integer repositoryId, Date fromDate) {
   this.fromDate = fromDate;
   this.init(repositoryId);
}

@Override
public void downloadMetadata(OutputStream outputStream,
DownloadMetadataTaskItemStatus downloadMetadataTaskItemStatus) throws Exception {

   try {

      String from = null;
      if (fromDate != null) {
         // format in OAI-PMH way, i.e. convert it to yyyy-MM-dd format
         SimpleDateFormat fromDateFormatter = new
SimpleDateFormat(FROM_DATE_FORMAT);
         from = fromDateFormatter.format(fromDate);
         logger.info("fromDate: " + fromDate + " formatted to: " + from + " and passed as
from parameter for incremental harvesting");
      }
```

```
                checkRepositoryUri(repository);

                // This block performs the actual metadata download
                oaiPmhMetadataRawWriteService.harvest(
                    new CleanUrl(repository.getUri()).toString(),
                    from, // From
                    null, // Until
                    repository.getMetadataFormat(),
                    null, // SetSpec
                    outputStream,
                    repositoryId,
                    downloadMetadataTaskItemStatus
                );

                // if it is less than 100 bytes then it is probably not what we want
                Boolean success = (new File(filesystemDAO.getMetadataPathPart(repositoryId,
fromDate, fromDate)).length() > 100);

downloadMetadataTaskItemStatus.setSuccess(downloadMetadataTaskItemStatus.isSuccess() &&
success);
                if (!success) {
                   logger.info("Metadata of repository " + repository.getId()
                        + " is probably not done, less than 100 bytes was downloaded!");
                   throw new Exception("Harvest not Successful as metadata size was less than 100
bytes");
                } else {
                   logger.info("Metadata of repository " + repository.getId()
                        + " have been downloaded.");


            if ("rioxx".equals(repository.getMetadataFormat().toLowerCase())) {
                    try {
                       supervisorClient.sendRioxxComplianceRepositoryRequest(repositoryId);
                    } catch (CHARSException ex) {
```

```
Logger.getLogger(OaiPmhDownloaderService.class.getName()).log(Level.SEVERE, null, ex);
                        }
                    }

                }

                //delete incremental files
        //        this.filesystemDAO.deleteFile(filesystemDAO.getIncrementalFolder(repositoryId));
            } catch (FileNotFoundException | ParserConfigurationException | SAXException |
TransformerException | InterruptedException | NoSuchFieldException ex) {
                downloadMetadataTaskItemStatus.setSuccess(false);
                downloadMetadataTaskItemStatus.addOAIPMHIssue("Exception: " +
ex.getMessage());
                logger.error("Exception " + ex.getMessage(), ex);
                throw new Exception(ex.getMessage(), ex);
            }
        }


        private void checkRepositoryUri(LegacyRepository repository) {
            String repositoryUri = repository.getUri();
            if(!repositoryUri.startsWith("https") && repositoryUri.startsWith("http")) {
                RestTemplate restTemplate = new RestTemplate();
                ResponseEntity<String> response = restTemplate.getForEntity(repositoryUri,
String.class);
                if(response.getStatusCodeValue()/100 != 4) {
                    repository.setUri(repositoryUri.replace("http://", "https://"));
                    repositoryDAO.updateUri(repository.getId(), repository.getUri());
                }
            }
        }
```

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____ О.А. Павлов

“___” _____ 2019 р.

Програмне забезпечення агрегації наукових публікацій

Технічне завдання

КПІ.ІП-5220.045440-02-91

“ПОГОДЖЕНО”

Керівник проекту:

_____ О.А. Халус

Виконавець:

_____М.А. Тарасюк

Нормоконтроль:

_____ К.І. Ліщук

Київ – 2019 року

ЗМІСТ

1	**НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ**	**3**
2	**ПІДСТАВА ДЛЯ РОЗРОБКИ**	**4**
3	**ПРИЗНАЧЕННЯ РОЗРОБКИ**	**5**
4	**ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ**	**6**
5	**ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ**	**8**
6	**СТАДІЇ І ЕТАПИ РОЗРОБКИ**	**10**
7	**ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ**	**11**

1 НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: Програмне забезпечення агрегації наукових публікацій.

Галузь застосування: університети, науковці та наукові проекти, яким необхідний доступ до повної бази наукових робіт.

Наведене технічне завдання поширюється на розробку програмного забезпечення “Програмне забезпечення агрегації наукових публікацій”, котра використовується для збору та систематизації наукових робіт та призначена для використання університетами та іншими науковими проектами.

2 ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки “Програмне забезпечення агрегації наукових публікацій” є завдання на дипломне проектування, затверджене кафедрою автоматизованих систем обробки інформації і управління Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» (КПІ ім.Ігоря Сікорського).

3 ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для збору та систематизації наукових публікацій.

Метою розробки є створення програмного забезпечення, яке надає можливість збереження наукових публікації в одному місці та швидкий доступ до них.

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до функціональних характеристик

4.1.1 Програмне забезпечення повинно забезпечувати виконання наступних основних функції:

– отримання наукової роботи;

– збереження наукової роботи;

– систематизація наукових робіт;

– пошук по науковим роботам.

4.2 Вимоги до надійності

4.2.1 Передбачити контроль введення інформації.

4.2.2 Передбачити захист від некоректних дій користувача.

4.2.3 Забезпечити цілісність інформації в базі даних.

4.3 Вимоги до складу і параметрів технічних засобів

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

4.3.2 Мінімальна конфігурація технічних засобів:

Тип процесору Pentium.

Об‘єм ОЗП 32 Мб.

Доступ до інтернету

4.4 Вимоги до інформаційної та програмної сумісності

4.4.1 Програмне забезпечення повинно працювати під управлінням операційних систем сімейства WIN32 (Windows XP, Windows NT і т.д.) або Unix.

4.5 Вимоги до маркування та пакування

Вимоги до маркування та пакування не пред‘являються.

4.6 Вимоги до транспортування та зберігання

Вимоги до транспортування та зберігання не пред'являються.

4.7 Спеціальні вимоги

Згенерувати установчу версію програмного забезпечення.

5 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

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

5.2 Програмне забезпечення повинно мати довідникову систему

5.3 У склад супроводжувальної документації повинні входити наступні документи:

5.3.1 Пояснювальна записка не менше ніж на 50 аркушах формату А4 (без додатків 5.3.2 - 5.3.6).

5.3.2 Технічне завдання.

5.3.3 Керівництво користувача.

5.3.4 Керівництво системного програміста

5.3.5 Керівництво адміністратора

5.3.6 Програма та методика тестування

5.4 Графічна частина повинна бути виконана у форматі А3, котрі включаються у якості додатків до пояснювальної записки:

5.4.1 Схема структура інформаційної системи.

5.4.2 Схема структурна програмного забезпечення.

5.4.3 Схема функціональна програмного забезпечення.

5.4.4 Схема структура потоків даних програмного забезпечення або його частини.

5.4.5 Схема структурна компонентів структур даних

5.4.6 Схема структурна варіантів використання

5.4.7 Схема структурна концептуальної моделі предметного середовища

5.4.8 Схемы взаимодействия объектов, объектная декомпозиция.

5.4.9 Схема структурна компонент.

5.4.10 Схема структурна класів програмного забезпечення

5.4.11 Схема структурна станів інтерфейсу

5.4.12 Креслення вигляду екранних форм.

6 СТАДІЇ І ЕТАПИ РОЗРОБКИ

Назва етапу	Строк,	Звітність
Вивчення літератури за тематикою проекту		
Розробка технічного завдання		Технічне завдання
Аналіз вимог та уточнення специфікацій		Специфікації програмного забезпечення
Проектування структури програмного забезпечення, проектування компонентів		Схема структурна програмного забезпечення та специфікація компонентів (діаграма класів, схема алгоритму)
Програмна реалізація програмного забезпечення		Тексти програмного забезпечення
Тестування програмного забезпечення		Тести, результати тестування
Розробка матеріалів текстової частини проекту		Пояснювальна записка.
Розробка матеріалів графічної частини проекту		Графічний матеріал проекту
Оформлення технічної документації проекту		Технічна документація

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

_____ О.А. Павлов

“___” _____ 2019 р.

Програмне забезпечення агрегації наукових публікацій

Програма та методика тестування

КПІ.ІП-5220.045440.03.51

“ПОГОДЖЕНО”

Керівник проекту:

_____ О.А. Халус

Нормоконтроль:

_____ К.І. Ліщук

Виконавець:

_____М.А. Тарасюк

Київ – 2019 року

ЗМІСТ

1 АНАЛІЗ ЯКОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 3

2 ПІДХОДИ ДО ТЕСТУВАННЯ 6

3 МЕТА ТЕСТУВАННЯ 8

4 ПРОЦЕС ТЕСТУВАННЯ 9

5 ВИМОГИ ДО СЕРЕДОВИЩА 10

5.1 Апаратна частина 10

5.2 Програмна частина 10

5.3 Інструменти 10

1 АНАЛІЗ ЯКОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Якість ПЗ - це сукупність властивостей, що визначають спроможність задовольнити запити замовника, які він висловив у вигляді вимог до розроблень. Згідно з міжнародними та вітчизняними стандартами оцінки рівня якості виділяють два процеси забезпечення якості впродовж життєвого циклу програмного забезпечення:

1) гарантія якості ПЗ, що є результатом певних дій на кожній стадії ЖЦ з перевірки й підтвердження відповідності ПЗ стандартам та процедурам, орієнтованим на досягнення якості;

2) інженерія якості як процес надання продуктам ПЗ надійності, супроводження й інших характеристик якості.

Ці процеси потребують:

– оцінки стандартів і процедур, що виконуються при розробленні програм;

– ревізії управління, розроблення і забезпечення гарантії якості ПЗ, а також усієї проектної документації (звітів, графіків розроблення, повідомлень);

– контролю проведення формальних інспекцій та оглядів;

– аналізу і контролю проведення тестування (випробувань) ПЗ.

Функціональність - це сукупність властивостей, які визначають спроможність ПЗ виконувати в заданому середовищі упорядковану послідовність дій для задоволення споживчих властивостей, замовлених користувачем, відповідно до вимог обробки і загальносистемних засобів. Атрибути функціональності ПЗ:

– функціональна повнота - атрибут, який показує ступінь достатності основних функцій для вирішення спеціальних завдань відповідно до призначення ПЗ;

– правильність - атрибут, який показує, як забезпечується досягнення правильних та погоджених результатів;

– інтероперабельність або сумісність - атрибути, які вказують на спроможність ПЗ взаємодіяти з іншими системами і середовищами;

– захищеність - атрибути, які вказують на можливість запобігати несанкціонованому доступу до програм і даних;

– узгодженість - атрибут, який вказує на відповідність заданим стандартам, угодам, правилам, законам і розпорядженням.

Надійність - це множина атрибутів, які вказують на спроможність ПЗ коректно перетворювати вхідні дані на результати. Зниження надійності ПЗ відбувається внаслідок помилок у вимогах, проектуванні і виконанні.

Атрибути надійності ПЗ:

– безвідмовність - атрибути, які визначають частоту відмов внаслідок наявності помилок у ПЗ;

– стійкість до помилок - атрибути, які вказують на забезпечення спроможності виконувати функції в аномальних умовах (збої апаратури, помилки в даних та інтерфейсах, порушення в діях оператора тощо);

– відновлюваність - атрибути, які вказують на спроможність програми до перезапуску для повторного виконання й відновлення даних після відмов;

– узгодженість - атрибут, який показує відповідність діючим стандартам, угодам, правилам, законам і розпорядженням.

Деякі типи систем (реального часу, радарні, безпеки, комунікації, медичного устаткування тощо) містять особливі вимоги до забезпечення високої надійності з такими атрибутами, як недопустимість помилок, безпека, захищеність і зручність застосування, а також достовірність як основний критерій надійності.

Зручність застосування - це множина атрибутів, що характеризують умови взаємодії користувача з ПЗ. Атрибути зручності застосування ПЗ:

– зрозумілість - визначається, наскільки зрозумілі для розпізнавання логічні концепції ПЗ та умов їх застосування;

– легкість навчання - визначається, наскільки доступні (легкі) для вивчення умови використання;

– оперативність - характеризується швидкістю реакції системи на дії користувача;

– узгодженість - визначається відповідністю розробки вимогам діючих стандартів, угод, правил, законів і розпоряджень;

Ефективність - це зв'язок між результатами використання ПЗ та кількістю задіяних для цього ресурсів (апаратура, матеріали, послуги обслуговуючого персоналу тощо).

Супроводжуваність - зусилля, які необхідно витратити на коригування, вдосконалення й адаптацію ПЗ у разі зміни середовища, вимог або функціональних специфікацій.

Атрибути супроводжуваності ПЗ:

– аналізованість - показник, який визначає необхідні зусилля для діагностики причин відмов або ідентифікації частин, що потрібно модифікувати;

– змінюваність - показник, який визначає зусилля на модифікацію, усунення помилок або внесення змін у зв'язку з помилками чи новими можливостями середовища функціонування;

– стабільність - атрибут, що характеризує імовірність модифікації;

– тестованість- атрибут, що характеризує зусилля щодо проведення валідації та верифікації.

Переносність - це здатність ПЗ пристосовуватися до роботи у разі зміни середовища виконання.

2 ПІДХОДИ ДО ТЕСТУВАННЯ

Всі види тестування програмного забезпечення, залежно від переслідуваних цілей, можна умовно розділити на наступні групи:

- функціональні (Functional testing);
- нефункціональні (Non-functional testing);
- пов’язані зі змінами (Regression testing).

Функціональні тести базуються на функціях і особливостях, а також взаємодії з іншими системами, і можуть бути представлені на всіх рівнях тестування: компонентному або модульному (Component / Unit testing), інтеграційному (Integration testing), системному (System testing) і приймальному (Acceptance testing).

Функціональні види тестування розглядають зовнішню поведінку системи. Далі перераховані одні з найпоширеніших видів функціональних тестів:

- функціональне тестування (Functional testing);
- тестування безпеки (Security and Access Control Testing);
- тестування взаємодії (Interoperability Testing).

Нефункціональне тестування описує тести, необхідні для визначення характеристик програмного забезпечення, які можуть бути виміряні різними величинами. В цілому, це тестування того, “Як” система працює. Далі перераховані основні види нефункціональних тестів:

- тестування навантаження (Performance and Load Testing);
- стресове тестування (Stress Testing);
- тестування стабільності або надійності (Stability / Reliability Testing);
- об’ємне тестування (Volume Testing);
- тестування установки (Installation testing);
- тестування зручності користування (Usability Testing);
- тестування на відмову і відновлення (Failover and Recovery Testing);

– конфігураційне тестування (Configuration Testing).

Після проведення необхідних змін, таких як виправлення бага / дефекту, програмне забезпечення повинне бути перетестоване для підтвердження того факту, що проблема була дійсно вирішена. Нижче перераховані види тестування, які необхідно проводити після установки програмного забезпечення, для підтвердження працездатності програми або правильності здійсненого виправлення дефекту:

– димове тестування (Smoke Testing);

– регресійне тестування (Regression Testing);

– тестування збірки (Build Verification Test);

– санітарне тестування або перевірка узгодженості / справності (Sanity Testing).

3 МЕТА ТЕСТУВАННЯ

У процесі тестування має бути перевірено наступне:

– збір ресурсів, які містять наукові роботи;
– отримання документів з зовнішнього ресурсу;
– збереження документу;
– перетворення документу;
– систематизація і пошук документів.

4 ПРОЦЕС ТЕСТУВАННЯ

Методом компонентного тестування перевіримо окремі частині API, такі як:

– оброблювачі шляхів API;

– методи доступу до базі даних;

– методи доступу до зовнішніх ресурсів.

Вхідними даними є набори параметрів на яких очікується певний результат, що є вихідними даними тестування.

Методом інтеграційного тестування будуть перевірені взаємодії між модулями системи, такі як:

– взаємодія сервера та бази даних;

– взаємодія сервера та черги подій;

– взаємодія сервера та зовнішніх ресурсів, які містять документи.

Вхідними даними є набори повідомлень, що будуть передані від одного компоненту системи до іншого відповідно до конкретного тесту.

Методом тестування продуктивності буде перевірена швидкодія системи.

Вхідними даними є набори даних, що покривають усі варіанти роботи системи в конкретному випадку.

Критерієм проходження є успішне виконання кожного пункту тесту. Якщо хоч один з пунктів не був виконаний - все тестування вважається проваленим.

5 ВИМОГИ ДО СЕРЕДОВИЩА

5.1 Апаратна частина

Вимоги до апаратної частини співпадають з вимогами з технічного завдання.

5.2 Програмна частина

Для виконання тестування апаратна платформа повинна мати операційну систему на базі Linux або іншу unix-сумісну, зі встановленою JDK8.

5.3 Інструменти

Для виконання тестування використовувати наступні програмні інструменти:

- Postman;
- PageSpeed;
- JIRA;
- Zephyr.

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____ О.А. Павлов

“___” _____ 2019 р.

Програмне забезпечення агрегації наукових публікацій

Керівництво користувача

КПІ.ІП-5220.045440.04.34

“ПОГОДЖЕНО”

Керівник проекту:

_____ О.А. Халус

Нормоконтроль:

_____ К.І. Ліщук

Виконавець:

_____М.А. Тарасюк

Київ – 2019 року

ЗМІСТ

1 ІНСТРУКЦІЯ КОРИСТУВАЧА 3

1.1 Пошук документів 3

1.2 Розширений пошук 3

1.3 Перегляд документів 4

1.4 Завантаження документу 5

1 ІНСТРУКЦІЯ КОРИСТУВАЧА

1.1 Пошук документів

Для роботи з системою користувачу не потрібно бути зареєстрованим. Для пошуку документа потрібно зайти на головну сторінку вебсайту та ввести в поле для пошуку ключові слова.(Рисунок 1.1).

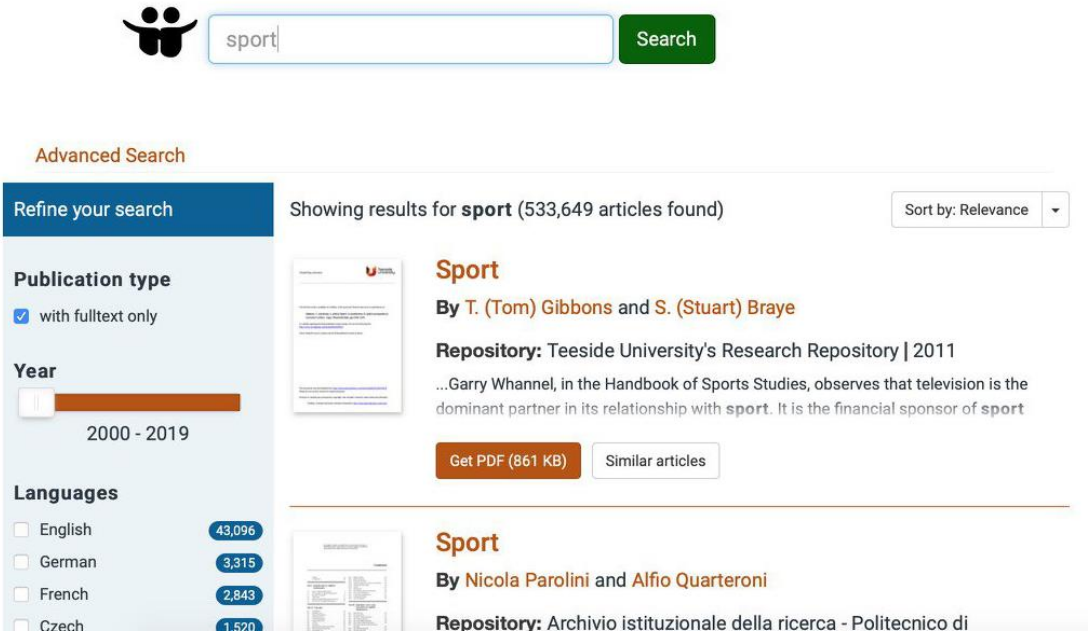


Рисунок 1.1 – Пошук документів

1.2 Розширений пошук

Окрім звичайного пошуку за ключовими словами можна здійснювати розширений. Наприклад пошук за іменем автора або роками написання публікації. Потрібно лише заповнити форму для вводу необхідних параметрів (Рисунок 1.2).

Рисунок 1.2 – Розширений пошук

1.3 Перегляд документів

Після того як був здійснений пошук по певним параметрам користувач бачить сторінку з результатами усіх підходящих статей. Усі слова за якими робився пошук виділені. Користувач має можливість переглянути усі документи, які були знайдені. (Рисунок 1.3).

Рисунок 1.3 – Перегляд документу

1.4 Завантаження документу

Після того як підходяща стаття була знайдена користуває має можливість скачати даний документ або перейти на сайт репозиторію, де був початково здобутий документ(Рисунок 1.4)

Рисунок 1.4 – Завантаження документу

ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ
Кафедра автоматизованих систем обробки інформації і управління

«ЗАТВЕРДЖЕНО»
В.о. завідувача кафедри
_____ І.П.Муха
(підпис) (ініціали, прізвище)
“___” _____ 2019 р.

«ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ АГРЕГАЦІЇ НАУКОВИХ ПУБЛІКАЦІЙ»
Графічний матеріал
КПІ.ІП-5220.045440.06.99

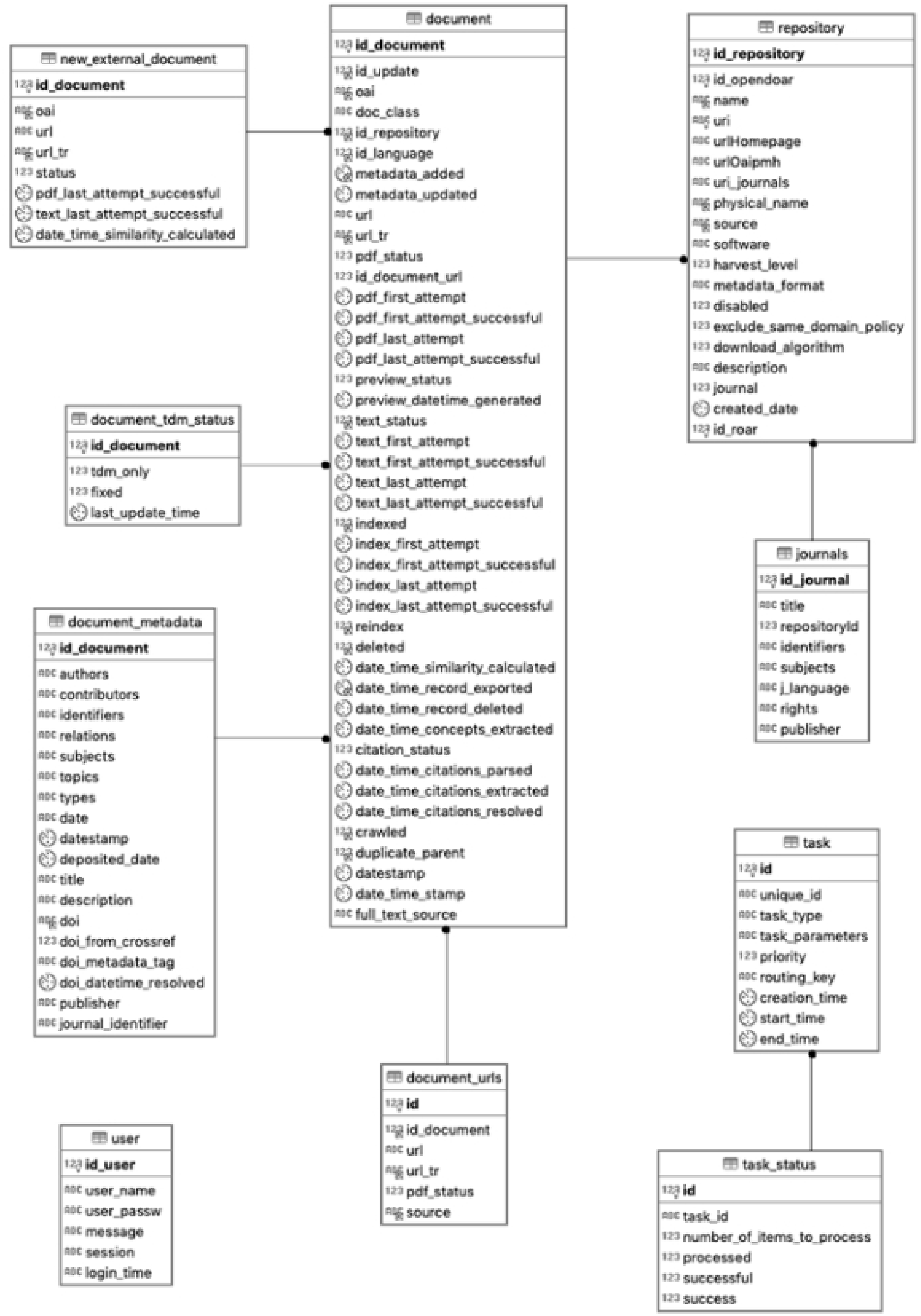
“ПОГОДЖЕНО”
Керівник проекту:
_____ Халус О.А.

Нормоконтроль:
_____ Ліщук К.І.

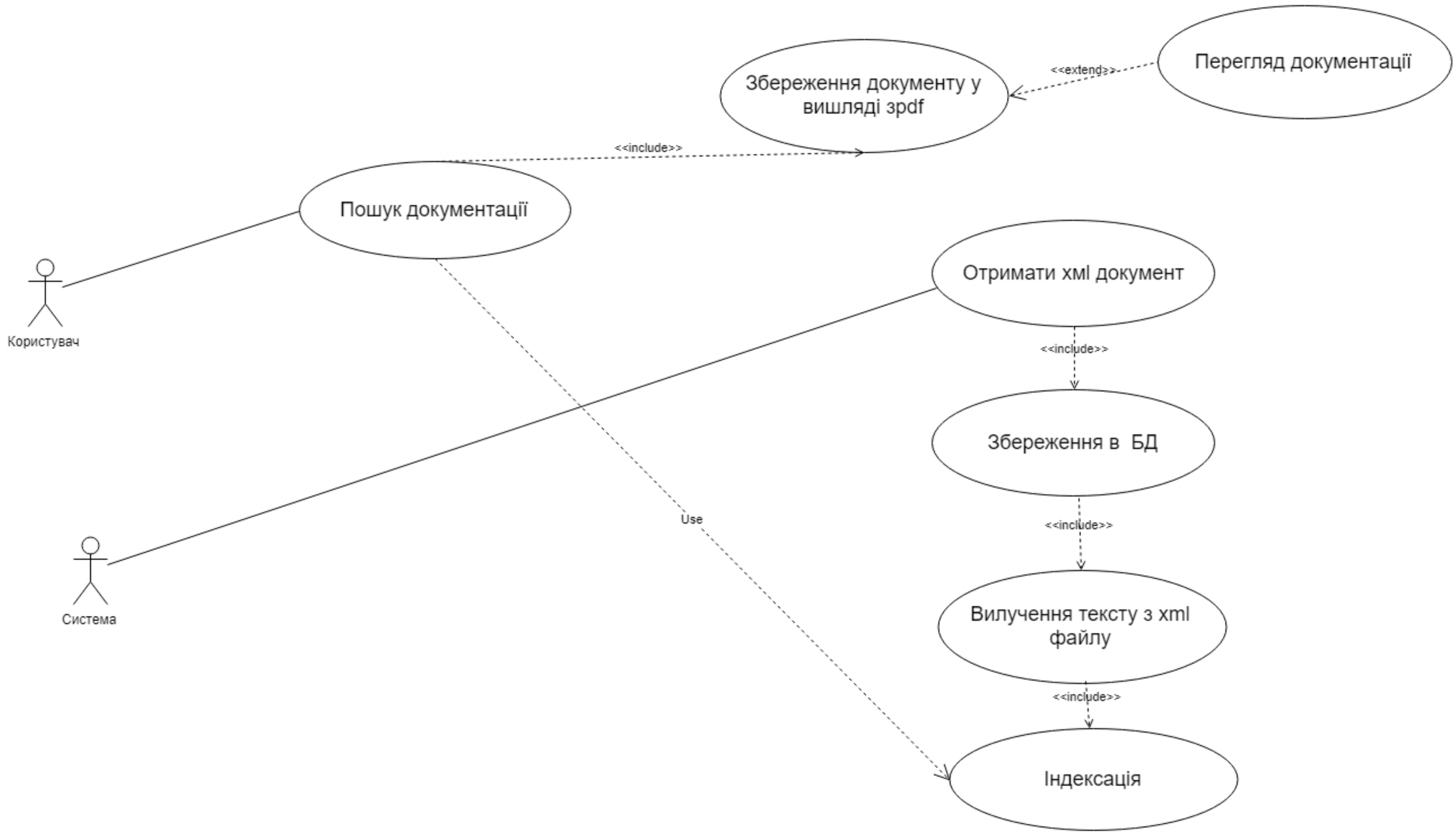
Виконавці:
_____ Тарасюк М.А.

КИЇВ – 2019 РОКУ

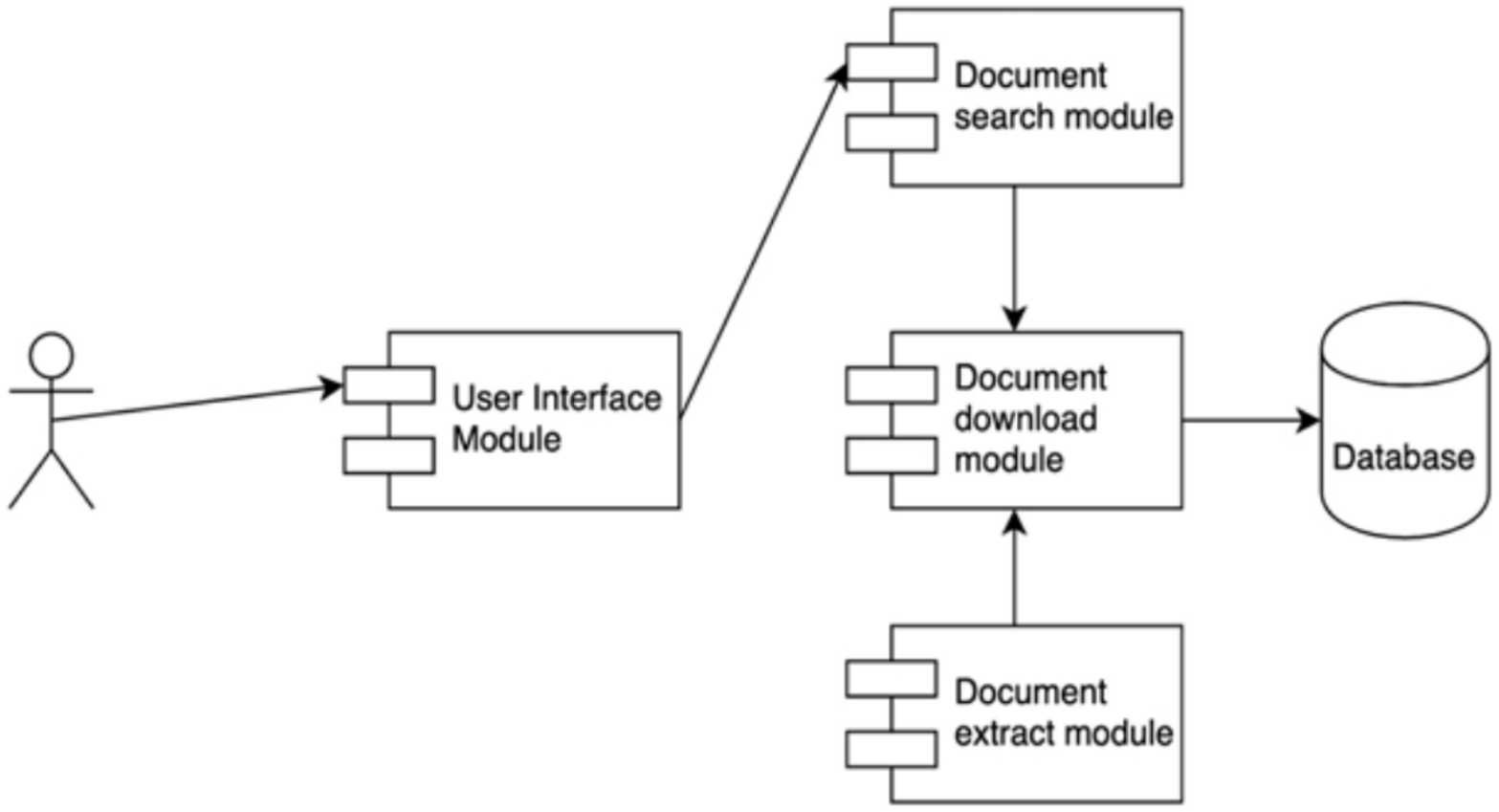
КПІ.ІП-5220.045440.06.99.СБД



					КПІ.ІП-5220.045440.06.99.СБД		
Зм.	Арк.	№ документа	Підпис	Дата			
Розробив		Тарасюк М.А.			Схема бази даних	Літера	Маса / Масштаб
Перевірив		Халус О.А					
Т. кон.						Аркуш 1	Аркушів 1
					Програмне забезпечення агрегації наукових публікацій	КПІ ім.Ігоря Сікорського Кафедра АСОІУ гр. ІП-52	
Н. кон.		Ліщук К.І.					
Затвердив		Халус О.А.					



					КПІ.ІП-5220.045440.06.99.СБД		
Зм.	Арк.	№ документа	Підпис	Дата	Схема структурна варіантів використання	Літера	Маса Масштаб
Розробив		Тарасюк М,А,					
Перевірив		Халус О.А.				Аркуш 1	Аркушів 1
Т. кон.							
Н. кон.		Ліщук К.І.			Програмне забезпечення агрегації наукових публікацій	КПІ ім.Ігоря Сікорського Кафедра АСОІУ гр. ІП-52	
Затвердив		Халус О.А.					



					КПІ.ІП-5220.045440.06.99.СБД	
Зм.	Арк.	№ документа	Підпис	Дата	Схема струкрурна компонентів	Літера / Маса / Масштаб
Розробив		Тарасюк М.А.				
Перевірив		Халус О.А.				Аркуш 1 / Аркушів 1
Т. кон.						
Н. кон.		Ліщук К.І.			Програмне забезпеченні наукових публікацій	КПІ ім.Ігоря Сікорського Кафедра АСОІУ гр. ІП-52
Затвердив		Халус О.А.				