КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет комп'ютерних наук та кібернетики
Кафедра теорії та технології програмування

Кваліфікаційна робота

на здобуття ступеня магістра

за спеціальністю 122 Комп'ютерні науки
на тему:

ВЕБЗАСТОСУНОК ДЛЯ ПУБЛІКАЦІЇ КВАЛІФІКАЦІЙНИХ РОБІТ

Виконав студент 2-го курсу магістратури
Юркевич Богдан Михайлович

(підпис)

Науковий керівник:
доцент, кандидат технічних наук
Ткаченко Олексій Миколайович

(підпис)

Засвідчую, що в цій курсовій роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент

(підпис)

Роботу розглянуто й допущено до захисту на засіданні кафедри теорії та технології програмування
«____» _____ 202_ р., протокол № ___
Завідувач кафедри
М. С. Нікітченко _____
(підпис)

Київ – 2023

РЕФЕРАТ

Кваліфікаційна робота складається з вступу, 3 розділів, висновку, списку використаних джерел (32 найменувань) та 4 додатків.

Робота містить 25 рисунка. Загальний обсяг роботи становить 60 сторінок, основний текст викладено на 46 сторінках.

ВЕБЗАСТОСУНОК, НАУКОМЕТРИЧНІ БАЗИ ДАНИХ, КВАЛІФІКАЦІЙНІ РОБОТИ, ПУБЛІКАЦІЯ, JAVA, MAVEN, MVC, SPRING FRAMEWORK, THYMELEAF, БАЗА ДАНИХ, MYSQL DATABASE.

Об’єктом дослідження є архіви наукових публікацій. Предметом дослідження є веб-орієнтовані системи формування архівів наукових публікацій.

Метою роботи є проектування та розробка вебзастосунку для публікації кваліфікаційних робіт.

Методи розроблення: комп’ютерне моделювання, комп’ютерне програмування, низхідний метод розробки, аналіз та синтез. Інструменти розроблення: інтегроване середовище розробки IntelliJ Idea, Spring Framework, мова програмування Java, Thymeleaf, Hibernate та реляційна СУБД MySQL .

Результати роботи: в роботі було проаналізовано різні системи для публікації наукових робіт, була спроектована та розроблена універсальна система для публікації студентських кваліфікаційних робіт, була спроектована база даних, після цього був спроектований та розроблений веб-застосунок. Це рішення дозволяє користувачам ділитися, публікувати та читати наукові роботи.

ЗМІСТ

ВСТУП

Актуальність дослідження. Нині публікація наукових робіт є важливою в галузі науки та досліджень. Публікації наукових робіт допомагають поширювати нові дослідження та відкриття, що є важливим для розвитку науки та технологій. Зокрема, це важливо і для майбутніх науковців та дослідників. Публікації дозволяють авторам отримувати зворотний зв'язок від колег з усього світу та вдосконалювати свої дослідження. В епоху цифрових технологій фізичні видання стали менш актуальними та мають такі проблеми, як швидкість публікації, доступність, інтерактивність, зворотній зв’язок.

Веб-орієнтовані наукові видання вирішують ці проблеми, але для цього потрібна єдина відкрита архівна база. Певним наближенням до такої бази є сукупність ресурсів Національної бібліотеки України імені В. І. Вернадського. Багато наукових публікацій розміщено на корпоративних репозиторіях. Це призводить до того, що далеко не у кожного є доступ до них. Об’єднана база даних помогла би студентам та іншим зацікавленим особам покращити і пришвидшити наукову діяльність, а також надала би змогу поширювати свої праці. Це було б покращенням не тільки наукового розвитку, але й навчального процесу.

Найбільш популярні іноземні веб-орієнтовані системи для публікації наукових робіт, що містять великі об’єми даних у власних базах даних, є зазвичай платними. Окрім того, дані сервіси погано пристосовані для використання українськими користувачами, оскільки їх інтерфейс та роботи, що в них містяться, є іншомовними. Також в даних системах складний процес публікації наукових робіт, що унеможливлює публікацію наукових матеріалів недосвідченими дослідниками чи простими студентами. Хоча і ретельний добір покращує загальну якість наукових матеріалів, що перебувають в сховищі, проте система, що описана в даній роботі, орієнтована на максимальну доступність як

процесу публікації, так і освоєння та пошуку потрібних матеріалів звичайними користувачами, де основними вимогами будуть дотримання авторських прав та дотримання наукового стилю.

Об'єкт дослідження: архіви наукових публікацій.

Предмет дослідження: веб-орієнтовані системи формування архівів наукових публікацій.

Фокус об'єкта зосереджено на студентських наукових публікаціях.

Мета дослідження. Створити веб-застосунок для формування бази даних наукових публікацій студентів.

Щоб досягнути мети потрібно виконати наступні **завдання**:

1. Дослідити існуючі системи публікації наукових праць;
2. Проаналізувати теоретичні засади створення вебзастосунку для публікації робіт;
3. Сформулювати вимоги до програмної системи;
4. Спроектувати архітектуру системи та базу даних;
5. Розглянути та обрати програмні засоби та технології для створення веб-застосунку;
6. Розробити веб-застосунок для формування архіву публікацій наукових праць;
7. Спроектувати тести та провести тестування створеної системи.

Методи й засоби розроблення. Перед веб-застосунок сайту було проаналізовано теоретичні засади створення веб-застосунку для публікації наукових робіт та вже існуючі схожі системи. Існуючі роботи з даної тематики та нові напрацювання дали можливість набагато краще розібратись в темі і виконати поставлені задачі. Застосунок розроблявся за методом низхідної розробки.

Основою за стосунку є мова програмування Java. Також було використано такі засоби як Spring Framework, Thymeleaf, Hibernate. Під час розробки проекту використовувалося інтегроване середовище IntelliJ Idea. Дане середовище має зручний інтерфейс та добре взаємодія з програмними технологіями, що використовуються в роботі.

Можливі сфери застосування. Розроблений програмний продукт може застосовуватись як офіційними, навчальними чи науковими організаціями (наприклад структурними підрозділами університету), так і широким колом користувачів як архів публікацій наукових робіт (наукові спільноти, академічна сфера).

РОЗДІЛ 1 ТЕОРЕТИЧНІ ЗАСАДИ СТВОРЕННЯ ВЕБЗАСТОСУНКУ ДЛЯ ПУБЛІКАЦІЇ КВАЛІФІКАЦІЙНИХ РОБІТ

1.1 Сутність та класифікація наукових робіт

Наукова робота – це самостійно виконане наукове дослідження тієї чи іншої проблеми, яке відповідає науковим принципам, має певну структуру, містить результат і власні висновки [1]. Ключовими критеріями для класифікації наукових робіт є структура та якість, адже наявність певних результатів і висновків та дотримання наукових принципів у подачі теми є спільними характеристиками для всіх робіт, що можна назвати науковими.

Іншим визначенням наукової роботи є визначення через публікацію в науковому виданні. В Україні наукове фахове видання — це періодичне або продовжуване видання (у тому числі — електронне), внесене до затвердженого Департаментом атестації кадрів МОН України переліку видань, у яких можуть публікуватися результати наукових досліджень на здобуття наукових ступенів доктора та кандидата наук.

Якісні характеристики роботи можна оцінювати різними способами. Це може бути оцінювання рецензентами або простими людьми. У випадку перевірки саме наукових праць рецензентами є люди, що вже мають досвід різного характеру у публікування робіт. Тобто це можуть бути як науковці з однієї сторони, так і редактори журналів чи бібліотекарі з іншої. За оцінюванням якості вмісту та структури роботи і визначається якісний критерій. Якщо роботу перевіряють прості читачі робіт, то тут головними критеріями є кількість оцінок, тобто популярність праці, та їх величина. Фактично це можна назвати рейтинговою системою на основі відгуків.

Ще одним критерієм якісної роботи є відсутність плагіату. Плагіат – це репрезентація виразів, думок чи ідей іншого автора як своїх [2]. Плагіат часто карається законом як фальсифікація чи шахрайство [3]. Зазвичай нормальним рівнем плагіату для роботи є не більше 5%, а рівень цитування не більше 20% від всього тексту (інколи критерії жорсткіші). Якщо плагіат перевищую 15%, то таку роботу вже не можна вважати аутентичною. [4]

Окрім якості, наукову роботу можна класифікувати за структурою. Найчастіше можна виділити такі види описових наукових робіт: реферат, тези, стаття, презентація.

Реферат — короткий виклад змісту документа чи його частини, що містить основні фактичні відомості та висновки, необхідні для початкового ознайомлення з документом [5].

Тези – це лаконічно сформульовані ключові аспекти, головні думки, ідеї та концепції будь-якої наукової праці [6]. Даний тип наукової роботи є більш демонстративним та таким, що ознайомлює читачів з більш розширеною версією роботи, наприклад, перед конференцією.

Стаття — науковий або публіцистичний твір невеликого розміру в збірці, журналі, газеті тощо. Вона є результатом розумового процесу, в якому поєднуються аналіз, структурування, формулювання та висловлення думок [7]. З означень можна зрозуміти, що всі наведені типи робіт є описовими, лаконічними і є створені для розгляду більш великих і фундаментальних праць.

На відміну від вищезгаданих видів робіт, презентація спирається більше на візуальні типи подачі інформації, ніж на текстові. Тобто в презентаціях часто використовують таблиці, графіки, ілюстрації, тощо. . В презентаціях основою є не текст, а усна розповідь доповідача. Проте добре створена презентація може якісно висвітлювати тему і без конспекту автора тому її теж можна вважати науковою роботою.

Щодо великих наукових праць, ними є книги, підручники, дисертації, тощо. Більшість таких праць мають за мету отримання вищого наукового ступеня автором або є навчальними чи науково-популярними матеріалами. Такі роботи часто пишуться декількома авторами та розкривають тему на дуже детальному рівні. Хоча система, що описана в даній роботі, і не спрямована на публікацію таких типів робіт, адже розповсюдженням таких робіт займаються видавничі заклади, проте вони є теж допустимими, якщо публікацію бажатиме законний автор.

У розробленій системі приймаються всі типи наукових робіт незалежно від структури. Щодо якісної перевірки, передбачене рецензування адміністраторами майбутнього вебзастосунку, а також рейтингова система на основі оцінок користувачів. Також планується інтеграція з засобами перевірки на плагіат.

1.2 Класифікація студентських робіт

Існують різні типи студентських робіт. Студентські наукові праці - це академічні роботи, які студенти виконують в процесі навчання в університеті або вищому навчальному закладі. Їх мета полягає в тому, щоб допомогти студентам розвинути дослідницькі навички та внести свій внесок у розвиток знань у різних галузях науки. Серед типів студентських наукових праць можна виділити бакалаврські роботи, наукові студентські звіти, конференційні тези та студентські статті. Кожен з цих типів має свої особливості та вимоги до написання, проте загальна мета їх полягає в тому, щоб допомогти студентам розвинути дослідницькі навички та здобути практичний досвід у своїй галузі.

Основною і найбільш значимою є дипломна робота бакалавра - це остаточна письмова робота, яку має написати студент, щоб отримати бакалаврський ступінь. Це завдання дається студентові на останньому році навчання і полягає в

проведенні дослідження на конкретну тему та підготовці детального звіту про результати дослідження. Дипломна робота бакалавра має на меті перевірити здатність студента застосовувати знання і навички, які він отримав протягом свого навчання, для вирішення конкретної проблеми чи виконання конкретного проекту. Зазвичай дипломна робота бакалавра складається зі вступу, теоретичної частини, практичної частини, висновків та списку використаних джерел. Дипломна робота магістра - це робота, яку студент пише в кінці свого магістерського навчання. Вона є продовженням дипломної роботи бакалавра та зазвичай вимагає більш глибокого дослідження теми, більш складного аналізу результатів і висновків.

Звіт наукових студентських робіт - це документ, що описує хід та результати проведення наукового дослідження студентом певної спеціальності. Цей документ може бути складений на основі виконання бакалаврської або магістерської дисертації, а також на підставі виконання наукових досліджень під керівництвом викладача.

Звіт наукових студентських робіт містить в собі вступ, де описується основна тема дослідження та його актуальність, теоретичну та практичну частини, де описуються методи дослідження, отримані результати та їх аналіз, висновки, де формулюються загальні висновки з проведеного дослідження та перспективи подальших досліджень. В звіті також мають бути вказані всі використані джерела та література.

Звіти наукових студентських робіт є важливим елементом наукової діяльності студентів, оскільки вони дозволяють їм продемонструвати свої наукові здібності та підготуватися до подальшої наукової роботи.

Ще одним прикладом студентських робіт є тези конференцій - це короткі резюме, які подаються на наукові конференції з метою отримання можливості представити свої дослідження. Вони зазвичай містять опис дослідження, його методику, результати та висновки. Студенти часто подають тези на конференції для представлення своїх досліджень та побудови відносин з колегами в галузі. Однак, менше досвідчені студенти повинні звернути увагу на підготовку та викладення своїх досліджень на конференції.

Щодо бакалаврських та магістерських робіт, їх можна розглядати як вирішення конкретної проблеми або завершення проекту. Бакалаврські роботи зазвичай зосереджені на розв'язанні конкретної проблеми, тоді як магістерські роботи зазвичай охоплюють більш широкі теми та потребують більш глибокого дослідження. Студенти працюють над цими роботами протягом тривалого періоду під керівництвом науковців, які надають їм необхідну підтримку та поради.

Іншим прикладом є студентські статті, які пишуться з метою публікації досліджень у наукових журналах. Основна мета таких статей полягає в тому, щоб поділитися результатами дослідження з іншими учасниками галузі та отримати відгуки щодо своєї роботи. Такі статті можуть допомогти студентам збільшити свою експертизу в певній галузі, а також збільшити їхні шанси на успіх у майбутньому.

Звіти про наукові студентські роботи мають велике значення в підготовці до подальших наукових досліджень та демонструванні наукових здібностей. Їх можна розглядати як звіти, які містять докладні відомості про дослідження, їхні методи та результати. Такі звіти допомагають студентам зрозуміти, як правильно структурувати та викладати інформацію в наукових текстах, а також розвивати навички наукового дослідження.

Студентські статті - це наукові тексти, які зазвичай пишуться студентами вищих навчальних закладів. Вони повинні відповідати науковим стандартам і містити результати дослідження, методику, аналіз та висновки. Це може бути частиною курсової роботи, проекту, наукового дослідження або публікації статті в науковому журналі. У студентських статтях зазвичай досліджуються питання, що стосуються конкретної галузі знань і можуть бути корисними для подальшого розвитку науки та технологій.

1.3 Огляд існуючих систем для публікації наукових робіт

В Україні більшість сайтів, які створені для публікації наукових праць, працюють на рівні локальних товариств, що займаються науковою діяльністю. Такими товариствами є зазвичай навчальні заклади.

У своїй базі даних навчальні заклади зазвичай публікують роботи, що потрібні для підвищення наукового ступеня. Це, наприклад, бакалаврські, дисертаційні, магістерські роботи. Зазвичай, щоб опублікувати статтю в такому закладі потрібно бути працівником даної організації або ж бути студентом. Це і є основним недоліком даних систем.

Одним з прикладів таких систем є Elsevier [8] - це один з найбільших видавництв наукової літератури в світі, засноване в 1880 році в Нідерландах. Компанія публікує більше 500 000 статей на рік в понад 2 500 журналах з різних наукових галузей, таких як медицина, біологія, фізика та ін.

Останнім часом Elsevier знаходиться під критикою за свою бізнес-модель, яка полягає в тому, що компанія продовжує збільшувати ціни на підписку на наукові журнали. Це створює проблему для університетів та наукових інститутів, які не можуть дозволити собі платити за доступ до всієї необхідної наукової

літератури. Це також призвело до того, що деякі науковці відмовилися від публікації в журналах Elsevier.

Однак, Elsevier також робить кроки для поліпшення доступності своєї літератури, такі як розширення доступу до відкритих даних та підписка на open-access журнали. Крім того, компанія розробляє нові технології, такі як штучний інтелект та аналітику даних, щоб полегшити науковий процес та підвищити якість досліджень.

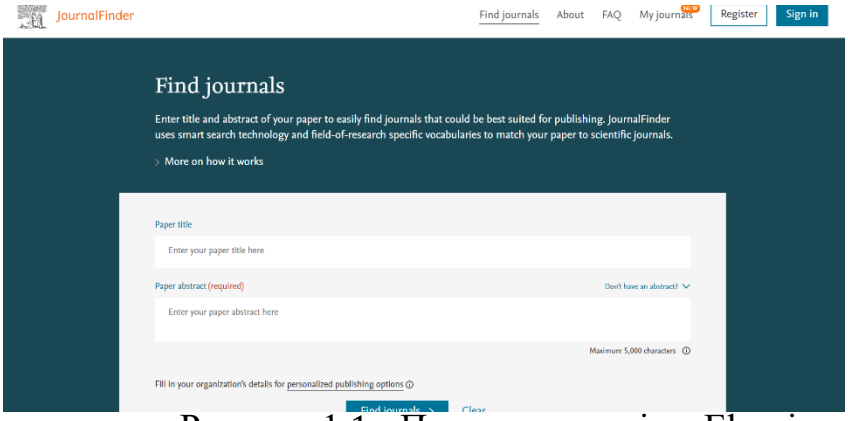

Рисунок 1.1 - Пошук журналів у Elsevier

Іншим популярним прикладом є Springer [9]. Глобальна видавнича компанія, заснована в Німеччині в 1842 році. Компанія видає понад 3000 журналів і 13 000 книг щорічно в різних наукових галузях, включаючи техніку, медицину та соціальні науки. Springer десятиліттями займає передову позицію в галузі наукових публікацій, і її публікації широко визнані як високоякісні ресурси для дослідників і вчених у всьому світі.

Однак Springer також зазнав критики за свою цінову політику, через яку університетам і науковим установам було важко отримати доступ до необхідної літератури. Деякі вчені навіть відмовилися публікувати інформацію в журналах Springer, посилаючись на стурбованість бізнес-практикою компанії.

Незважаючи на ці проблеми, Springer вживає заходів для покращення доступності та доступності. Компанія розробила ряд варіантів відкритого доступу, включаючи Springer Open і BioMed Central, які надають безкоштовний доступ до великої кількості статей і журналів. Springer також запустив низку ініціатив, спрямованих на покращення наукового спілкування та співпраці, включаючи програму Springer Nature SharedIt, яка дозволяє авторам ділитися своїми опублікованими статтями з колегами та однолітками.

Окрім видавничої діяльності, Springer також бере активну участь у розробці нових технологій та інструментів для полегшення наукового процесу. Компанія інвестувала значні кошти в штучний інтелект і аналітику даних і працює над розробкою нових інструментів і послуг, які допоможуть дослідникам і науковцям ефективніше аналізувати та інтерпретувати дані.

Загалом Springer залишається провідним видавцем у науковому співтоваристві, і його внесок у розвиток наукових знань широко визнаний. Хоча компанія стикається з проблемами, пов’язаними з ціноутворенням і доступністю, вона вживає заходів для вирішення цих проблем і покращення загальної якості та доступності своїх публікацій.

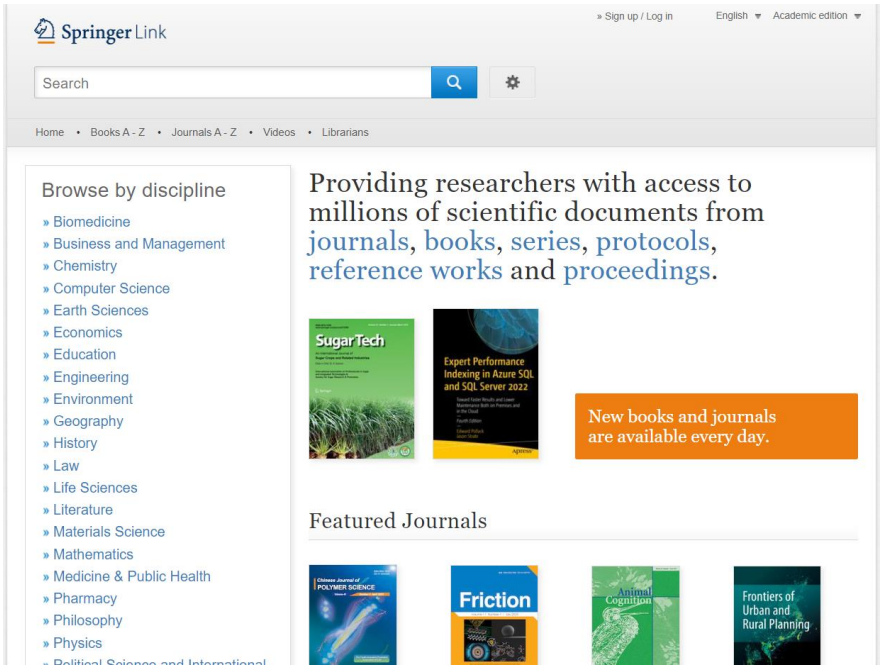

Рисунок 1.2 - Пошук видань у Springer

Недоліками цих систем є потреба у платній підписці, і навіть вона не доступна для всіх країн, зокрема України. Безплатним схожим варіантом є Google Scholar [9]. Це безкоштовний веб-пошуковий сервіс, що надає можливість пошуку наукових публікацій, таких як статті, книги, реферати, дисертації та інші видання. Він був запущений у 2004 році і відтоді став важливим інструментом для науковців та дослідників у всьому світі.

Основна перевага Google Scholar полягає в тому, що він забезпечує широкий доступ до наукових публікацій з усього світу. Крім того, він пропонує велику кількість функцій, що допомагають знайти необхідну інформацію. Наприклад, він здатний автоматично збирати цитування певної публікації, що дає можливість прослідкувати, як використовується ця інформація в інших дослідженнях.

Незважаючи на ці переваги, Google Scholar також має свої обмеження. Наприклад, він не є повноцінною базою даних та може пропускати деякі

публікації. Крім того, сервіс не завжди дозволяє отримати повний текст публікації, що може ускладнювати дослідницьку роботу.

Незважаючи на ці обмеження, Google Scholar є важливим інструментом для науковців та дослідників, оскільки надає доступ до великої кількості наукових публікацій та допомагає знайти необхідну інформацію. Крім того, він постійно розвивається та додавати нові функції, що робить його ще більш корисним для дослідників та науковців.

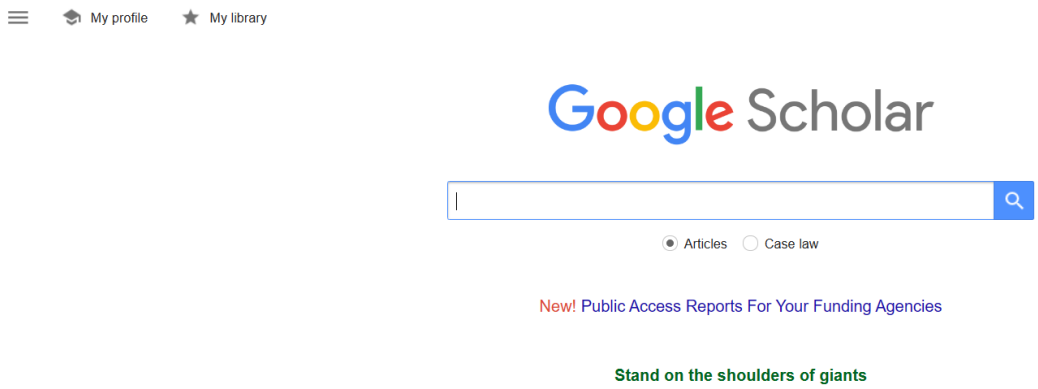

Рисунок 1.3 - Головна сторінка Google Scholar

Окрім таких міжнародних агрегаторів наукових видань які орієнтовані на іноземну аудиторію є також українські системи. Це Українська наукова періодика [10], різноманітні архіви на базі різних університетів, тощо.

Українська наукова періодика - це система наукових видань, які публікують наукові статті, дослідження, рецензії, огляди та інші наукові матеріали українських та зарубіжних авторів, які присвячені різним галузям науки. Українська наукова періодика є важливим інструментом для розвитку науки та її популяризації.

Українська наукова періодика має багато видань різного наукового рівня, що відповідають вимогам наукового світу. До таких видань відносяться наукові журнали, збірники наукових праць, монографії та інші наукові видання. Кожен з них має свої вимоги до форматування та оформлення рукописів, до кількості джерел, до наукового рівня статті, до рівня оригінальності дослідження та інших вимог, які потрібно дотримуватися.

Одним з основних видань Української наукової періодики є наукові журнали, які публікуються на різних мовах. Найбільш поширеними з них є журнали, що входять до баз даних Scopus [11] та Web of Science [12], такі як "Ukrainian Journal of Physical Optics", "Problems of Atomic Science and Technology", "Ukrainian Journal of Ecology", "Ukrainian Journal of Chemistry" та інші. Ці видання відомі своїм високим науковим рівнем, а також тим, що дотримуються високих стандартів редакційної практики та етики публікацій.

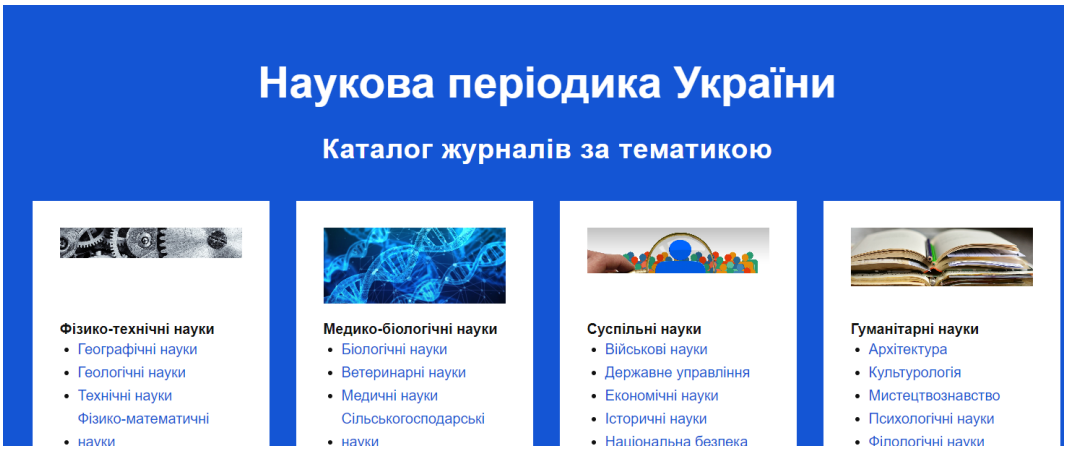

Рисунок 1.4 - Наукова періодика України

Щоб опублікувати свою працю в одній з цих систем для початку потрібно знайти видання з яким можливо домовитися про публікацію з редакцією і виконати багато формальної роботи для самої публікації. Тому це створює великий поріг входження для публікацій робіт.

Щодо самих видань в Україні існує багато вузько направлених видань таких як Науковий вісник Національного університету "Львівська політехніка", Український біохімічний журнал, "Вісник Київського національного університету імені Тараса Шевченка. Геологія", Український фізичний журнал. Також можна згадати про найбільшу бібліотеку України – бібліотеку ім. В.І. Вернадського. Сьогодні вона є науковим центром, який забезпечує доступ до наукової інформації та здійснює науково-інформаційні послуги для вчених та наукових установ України. Бібліотека також забезпечує доступ до своїх ресурсів за допомогою електронного каталогу та інших інформаційних систем.

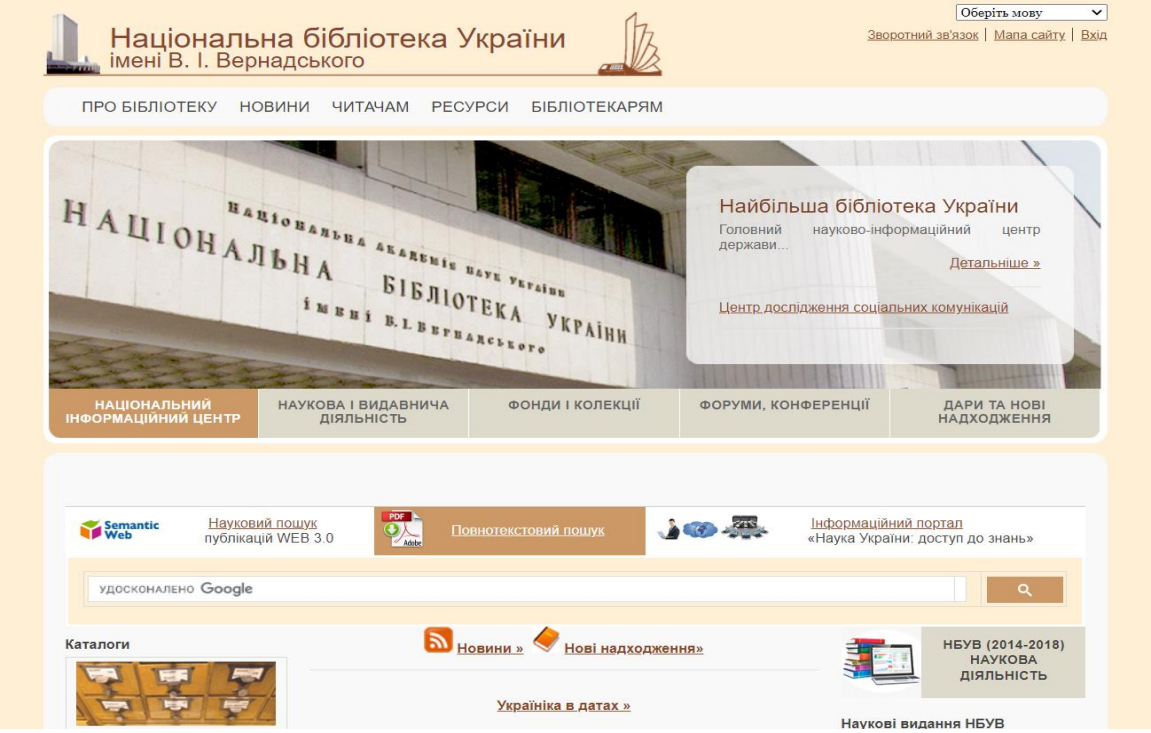

Рисунок 1.5 – Бібліотека ім. Вернадського

Отже, в Інтернеті вже існує досить багато систем для публікації наукових робіт. Слід зазначити, що система буде доступною для більш широкого користувача на противагу спеціалізованим виданням які орієнтовані на одну певну галузь та є спрямованою на україномовну аудиторію.

1.4 Бази даних для індексації наукових публікацій

Бази даних для індексації наукових публікацій - це електронні ресурси, які містять інформацію про наукові публікації, зокрема, назву публікації, авторів, рік публікації, журнал або видавництво, в якому публікація була опублікована, а також бібліографічні посилання на інші роботи, на які посилається ця публікація.

Ці бази даних використовуються для індексації наукових публікацій з метою забезпечення швидкого та ефективного доступу до цих даних. Наприклад, деякі з найбільш відомих баз даних для індексації наукових публікацій включають Scopus, Web of Science, Google Scholar та інші.

Такі бази даних мають велике значення для наукової спільноти, оскільки вони дозволяють дослідникам швидко знаходити публікації, які стосуються їх досліджень, а також використовувати ці дані для аналізу тенденцій та розвитку наукових дисциплін. Крім того, вони є важливим інструментом для визнання наукових досягнень, оскільки більшість наукових видавництв вимагають, щоб публікації були індексовані в певних базах даних, щоб вони могли бути визнані як наукові досягнення.

Бази даних для індексації наукових публікацій є важливим інструментом для визнання наукових досягнень та відображення впливу досліджень на наукову спільноту. Однак, деякі дослідники вважають, що використання наукометричних показників повинно бути здійснене з обережністю, оскільки вони не враховують всіх аспектів наукових досягнень та можуть призвести до спотворення цінності наукових досліджень.

Наукометричні бази даних використовують алгоритми індексації для організації та пошуку даних про наукові публікації, цитування та наукометричні

показники. Один з найпоширеніших алгоритмів індексації у наукометричних базах даних - це індексація за допомогою ключових слів (keyword indexing).

За цим методом, кожна наукова публікація індексується з використанням ключових слів, які надає автор публікації. Ключові слова - це терміни, які найкраще описують тему публікації, і вони використовуються для створення індексу даних.

Ключові слова можуть бути введені автором самостійно або надані редакторами наукового журналу. Після цього, ці ключові слова стають доступними для пошуку в базі даних. Користувачі можуть шукати публікації за ключовими словами або їх комбінаціями, що дозволяє швидко знайти наукові публікації за певними темами або проблемами.

Інші методи індексації, які також використовуються в наукометричних базах даних, включають індексацію за авторами, інституціями, рубриками журналів та темами наукових дисциплін. Однак, індексація за ключовими словами є основним методом індексації у багатьох наукометричних базах даних, оскільки вона дозволяє швидко та ефективно знаходити наукові публікації за певними темами або проблемами.

РОЗДІЛ 2. ОГЛЯД ТЕХНОЛОГІЙ ДЛЯ СТВОРЕННЯ ВЕБЗАСТОСУНКУ

2.1 Мова програмування Java та збірка проекту

Одним з важливих етапів створення програми є вибір актуальних технологій для її розробки. Даний розділ посвячений опису всіх технологій, що використовуються для системи.

Для серверної частини програми була обрана мову програмування Java. Java — це високорівнева, заснована на класах, об'єктно-орієнтована мова програмування, яка розроблена так, щоб якомога менше залежати від реалізації. Це мова програмування загального призначення, призначена для того, щоб програмісти могли запускати написаний програмний код в будь-якому місці. Це означає, що скомпільований код Java може працювати на всіх платформах, які підтримують Java, без необхідності перекомпіляції [15].

Основними плюсами даної мови програмування є простота і С-подібність, структурованість, ієрархічність та хороша типізація. Завдяки цьому є більшою читабельність програмного коду. Будь-котра людина, що вивчала колись будь-яку С-подібну мову (C++, C#, тощо) швидко вивчить синтаксис цієї мови.

Ще одним важливим плюсом мови програмування Java це автоматичне управління пам’яттю, який також використовується в мові програмування Swift, і програма збирача сміття, яка автоматично обробляє виділення та звільнення пам’яті.

Також для покращення чистоти коду використовується Lombok [16]. Ця бібліотека дозволяє не описувати в нових класах конструктори та методи для

отримання та встановлення значень атрибутів. Замість цього потрібно написати певні анотації, що значно зменшує обсяг коду.

Як і будь-яка мова програмування високого рівня, Java має проблеми з продуктивністю через рівень компіляції та абстракції віртуальної машини. Однак це не єдина причина поганої швидкості Java, яку часто за це критикують. Збірник сміття може займати більше 20 відсотків часу процесора. Погана конфігурація кешування також може призвести до надмірного використання пам'яті та збору сміття. Існують також тупикові блокування потоків, які трапляються, коли кілька потоків намагаються отримати доступ до одного ресурсу. Проте кожну з цих проблем можна запобігти за допомогою вмілого планування програмного коду [17].

За збірку проекту відповідає Maven. Apache Maven — це інструмент для управління та розуміння програмного проекту. Базуючись на концепції об'єктної моделі проекту (POM), Maven може керувати збіркою проекту, його звітністю та документацією [18]. Для того, щоб в проекті можна було використовувати нову бібліотеку, потрібно всього лиш добавити залежність в файл pom.xml.

2.2 Технології Spring Framework

В основі веб-застосунку було використано Spring Framework [19]. Фреймворк Spring створює програмування на мові Java швидшим, простішим та безпечнішим. Даний програмний каркас має багато переваг. Він надає гнучкі бібліотеки, які є популярними і актуальними. Для параметрів конфігурації розробник може вибрати Java анотації або XML. Функції інверсії контролю і ін'єкції залежностей створюють основу для широкого набору функціональних можливостей. Він забезпечує декларативну підтримку кешування, перевірки, транзакцій і форматування. Також даний фреймворк забезпечує відокремлення

конфігурації від логіки програми. Окрім цього Spring надає стандартні для галузі схеми аутентифікації та авторизації.

Основною властивістю даного фреймворку є інверсія управління. Інверсія управління використовується для збільшення модульності програми та її розширення. Інверсія контролю несе важливу ідею, що код для повторного використання та код для конкретної проблеми розробляються незалежно, навіть якщо вони працюють разом у додатку. Тобто між програмними класами слабка залежність і зміна одних класів не впливатиме і не ламатиме інші ієрархічні структури.

Мінусами Spring Framework є складність у використанні через велику кількість залежностей з іншими технологіями та часте використання дещо застарілого XML підходу до конфігурацій.

Для захисту програми використовується фреймворківська утиліта Spring Security. Дана структура забезпечує автентифікацію та контроль доступу до веб-застосунку. Вона захищає від різноманітних атак таких як, фіксація сесії, клікджекінг, міжсайтові запити, тощо [20]. Можна легко налаштовувати доступ до сторінок, до яких потрібна авторизація.

Важливою перевагою Spring є підтримка шаблону MVC. MVC Pattern розшифровується як Model-View-Controller Pattern. Цей шаблон використовується для розділення різних задач програми. Модель – це простий об’єкт або JAVA POJO, що має на меті представляти собою певні дані. Він також може мати логіку оновлення контролера, якщо його дані змінюються. Контролер містить бізнес-логіку програми. Він керує потоком даних в об’єкт моделі та оновлює представлення (View) щоразу, коли дані змінюються. Він розділяє представлення і модель. Представлення являє собою візуалізацію даних, які містить модель [21]. Схема шаблону зображена на рисунку 1.

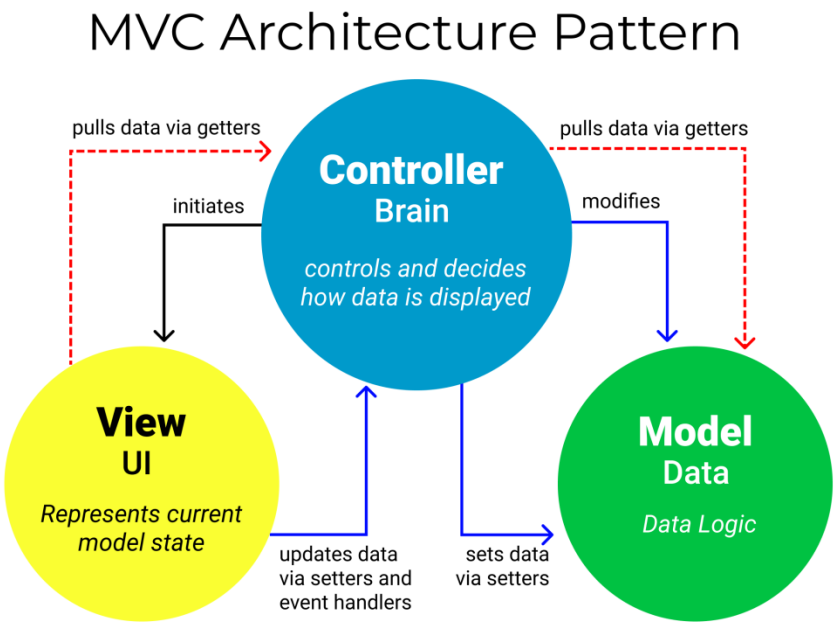

Рисунок 2.1 - Схема шаблону MVC [22]

Для зв’язку програмного коду з базою даних було використано технології Spring та JPA Hibernate [23]. Java Persistence API (JPA) — це специфікація для відображення об’єктів Java на таблиці бази даних. Spring Data JPA — це абстракція, яка полегшує роботу з JPA. Зокрема, Spring Data JPA надає набір інтерфейсів для легкого створення репозиторіїв доступу до даних. Hibernate — це провайдер JPA та ORM (Object-Relational Mapping), який відображає об’єкти Java у таблиці реляційної бази даних. Використовуючи Spring Data JPA, можна усунути багато стандартного коду, який пов’язаний з керуванням зв'язків між базою даних і програмою.

Ще однією перевагою Spring Framework є безліч бібліотек для додаткового функціоналу. Це, наприклад, утиліта для автоматичного надсилання листів на пошту користувача, підготовлені класи для ефективного передавання та зберігання файлів на сервері та бібліотека для шифрування паролів користувачів.

Отже, Spring Framework є технологією, що надає безліч бібліотек та засобів як для ефективної реалізації функціоналу веб-застосунку, так і для побудови грамотної структури проекту для полегшення майбутньої підтримки програмного засобу.

2.3 Технології для розробки клієнтського інтерфейсу

Для відображення інтерфейсу буде використовуватись мова розмітки HTML. HTML, або мова гіпертекстової розмітки, дозволяє створювати та структурувати розділи, абзаци та посилання за допомогою елементів, тегів та атрибутів. Розробники використовують HTML-код, щоб розробити відображення елементів веб-сторінки у браузері, такі як текст, гіперпосилання та медіафайли.

Для полегшення роботи з передачею елементів від контролера до веб сторінок і навпаки , було використано Thymeleaf [24]. Thymeleaf — це сучасний серверний механізм шаблонів Java для веб- та автономних середовищ. Шаблони HTML, написані на Thymeleaf, все ще виглядають і працюють як HTML, дозволяючи фактичним шаблонам, які запускаються у вашій програмі, продовжувати працювати як корисні артефакти дизайну. Thymeleaf допомагає як у передачі даних з контролерів до представлення, так і в локалізації тексту на веб-сайті. Окрім цього Thymeleaf надає можливість перевіряти права користувачів вже в фронтовій частині програми. Це полегшує розробку веб-застосунку в плані розділення функціоналу адміністратора та простого користувача та дає додаткову перевірку прав, що збільшує безпеку програмного застосунку.

Для покращення зовнішнього вигляду вебзастосунку було використано Bootstrap. Bootstrap – це потужний інтерфейсний інструментарій, що володіє великим набором вже готових рішень щодо зміни вигляду веб-сайту шляхом додавання до HTML тегів заготовлених бібліотекою CSS класів [25]. Для того,

щоб використовувати дану утиліту потрібно додати посилання в шапку HTML документу.

Окрім Bootstrap класів в програмі використовуються і не бібліотечний CSS. Каскадні таблиці стилів (англійською Cascading Style Sheets) – це мова стилів, що використовується для опису вигляду документу HTML [26]. Даний інструмент дозволяє розділяти контент та його презентацію у веб-застосунку. Це надає більшу гнучкість та простоту при розробці представлення програмного продукту.

Описані засоби подачі клієнтського інтерфейсу є найбільш перевіреними та поширеними, а найголовніше вони якісно розділяють функціонал та вигляд сайту, що дозволяє більш поступово та грамотно розробляти застосунок.

2.4 Технології для взаємодії з базою даних

Для розробки бази даних використовується MySQL [27]. MySQL - це реляційна база даних, що організовує дані в одну або кілька таблиць, в яких дані можуть бути пов’язані один з одним; ці відносини допомагають структурувати дані. SQL — це мова, яку програмісти використовують для створення, зміни та вилучення даних з реляційної бази даних, а також для керування доступом користувачів до бази даних.

MySQL Workbench — його краще використовувати для отримання інформації чи дрібних змінах через візуальний інструмент для архітекторів баз даних, розробників і адміністраторів баз даних. Даний інструмент використовується для моделювання даних, адміністрування та конфігурації сервера, адміністрування сервера, резервного копіювання, тощо. Даний інструмент добре підходить для візуалізації схеми бази даних та для керування даними [28].

Також для адміністрування бази даних проекту, на рівні розробки, широко використовувався MySQL Command Line Client. Даний інструмент є швидшим і

простішим, ніж MySQL Workbench. Для того, щоб адмініструвати базу даних достатньо ввести пароль до серверу і обрати базу даних. Після цього за допомогою SQL запитів можна взаємодіяти з табличними даними. При інтерактивному використанні результати запиту представлені у форматі ASCII-таблиці. При неінтерактивному використанні (наприклад, як фільтр) результат представлений у форматі, розділеному табуляцією. Формат виводу можна змінити за допомогою параметрів команди.

Обрані програмні засоби для адміністрування бази даних мають хорошу взаємодію з мовою програмування Java та середовищем IntelliJ Idea.

2.5 Технології для програмного тестування застосунку

Основою програмного тестування веб-застосунку є модульне тестування. У комп'ютерному програмуванні модульне тестування — це метод тестування програмного забезпечення, за допомогою якого конкретні окремі одиниці вихідного коду — набори одного чи кількох модулів комп'ютерної програми разом із пов'язаними керуючими даними — перевіряються, щоб визначити, чи придатні вони для використання при певних даних. Мета модульного тестування — виокремити кожну частину програмного застосунку та показати, що окремі модулі правильні [29].

Даний підхід до тестування має декілька переваг. Оскільки таке тестування передбачається на ранньому етапі розробки програми, творець тестів повинен чітко продумувати входи, виходи та умови помилок тестів і, таким чином, більш чітко визначити бажану поведінку пристрою. Також програмний код може бути занадто складним для модульного тестування, якщо він погано написаний, тому модульне тестування змушує розробників краще структурувати функції та об'єкти.

Основним недоліком модульного тестування (як і будь-якого іншого програмного тестування) є те, що воно не вловить кожну помилку в програмі, оскільки не може оцінити кожен шлях виконання в програми, крім найтривіальніших. Ця проблема є надмножиною проблеми зупинки, яка є нерозв'язною. Також модульне тестування не виявить помилок інтеграції з іншими програмними засобами чи бібліотеками. Для доведення того, що програмний модуль не має проблемної поведінки, потрібно використовувати формальні методи тестування.

Основною утилітою для запуску модульних тестів є JUnit. Дана платформа служить основою для запуску тестових фреймворків на JVM [30]. JUnit надає безліч анотацій для полегшення написання тестів, а також бібліотек, що передбачають різноманітні перевірки даних. Також вона має хорошу взаємодію з середовищем розробки програмного забезпечення IntelliJ Idea.

Для більш коректного модульного тестування використовується фреймворк Mockito. Mockito дозволяє зручно створювати замінники реальних об'єктів для тестування [31]. Заміна дозволяє зосередитись на тестуванні конкретного єдиного модуля, а його інтеграції замокати, тобто замінити об'єктам поведінкою, що передбачається.

Обрані технології є сучасними і широко розповсюдженими. Це дає змогу покращувати проект в майбутньому без використання застарілих бібліотек та утиліт. Також дані технології дозволяють сконцентруватися на розробці і продумуванні саме бізнес-логіки програми, а не на технічних деталях.

РОЗДІЛ 3. ВЕБ-ЗАСТОСУНОК ДЛЯ ПУБЛІКАЦІЇ НАУКОВИХ РОБІТ

3.1 Вимоги до системи

Система повинна мати такі риси та виконувати такі функції:

1. Надійність та безпека. Додаток має забезпечувати захист персональних даних користувачів, а також відповідний рівень захисту від хакерських атак та вірусів;
2. Зручність та простота використання. Додаток має мати простий та зрозумілий інтерфейс, що дозволить користувачам легко та швидко публікувати свої наукові роботи;
3. Функціональність. Додаток має мати можливість публікації наукових робіт у форматі PDF та Word, а також забезпечувати можливість редагування та видалення робіт;
4. Авторські права. Додаток має забезпечувати захист авторських прав користувачів на їх наукові роботи та відповідну систему ліцензування;
5. Пошук та фільтрація. Додаток має мати можливість пошуку та фільтрації наукових робіт за різними критеріями, такими як автор, тема, ключові слова тощо;
6. Підтримка відгуків та рецензії. Додаток має забезпечувати можливість користувачам залишати відгуки та рецензії на наукові роботи, що дозволить покращити якість робіт та забезпечити зворотний зв'язок між авторами та читачами.

Рисунок 3.1 - Діаграма Use Case

Проект можна умовно поділити на декілька шарів. Ними є репозиторії, сервіси, контролери та відображення сайту. Кожний рівень має свою роль та функціонал в проекті. Основними сутностями, які обробляє програма, є класи-домени та DTO (Data Transfer Object).

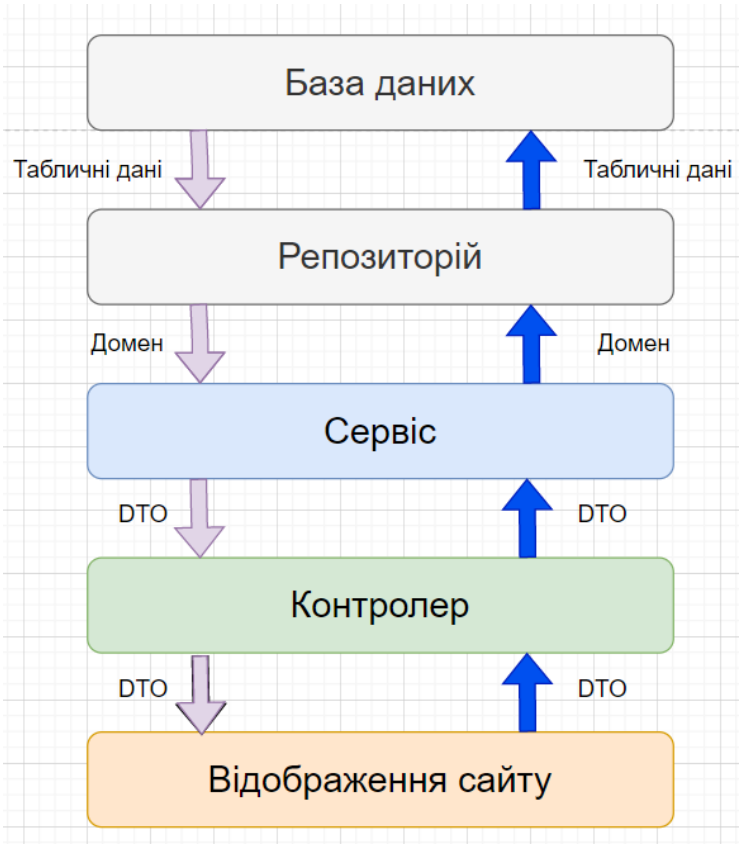

Рисунок 3.2 - Архітектура проекту та потоки даних

Клас-домени представляють дані, що зберігаються в базу даних. По суті кожний такий клас репрезентує таблицю, майже кожний атрибут якого відповідає певному атрибуту таблиці. Також є атрибути, що відповідають за зв'язки між таблицями. Наприклад, якщо зв'язок між таблицями типу "багато до багатьох", то в обох класах-доменах, буде присутній атрибут, що відповідає за даний зв'язок у вигляді списку чи множини об'єктів. Це дає хорошу гнучкість програмі в плані написання якісного програмного коду (див. додаток А) .

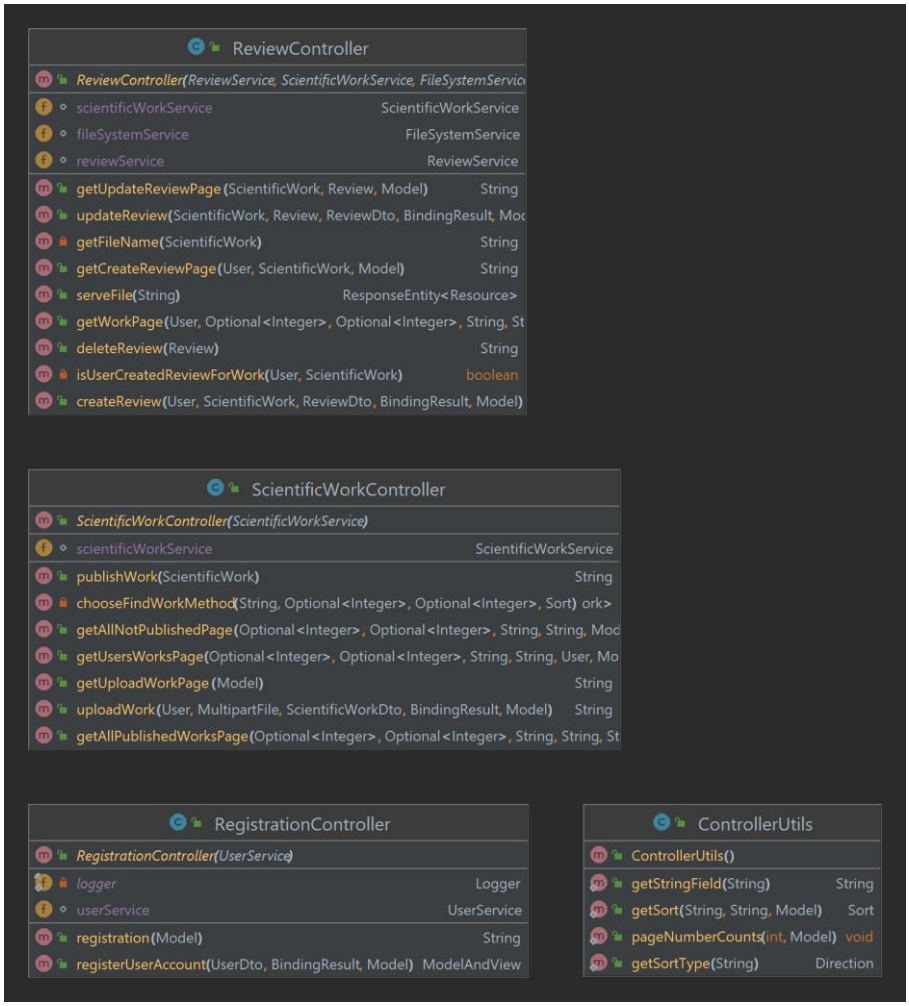

Рисунок 3.3 – Діаграма класів контролера

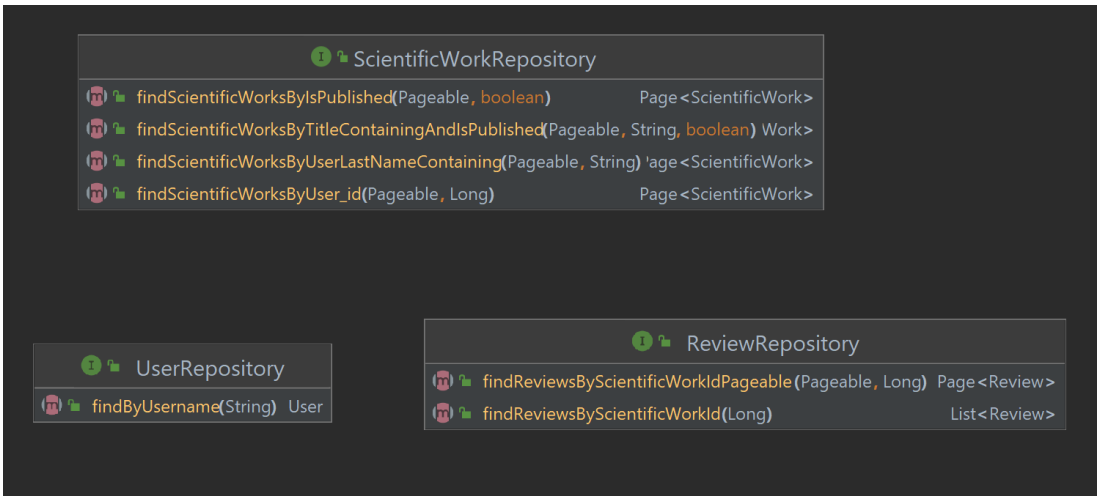

Рисунок 3.4 – Діаграма класів репозиторію

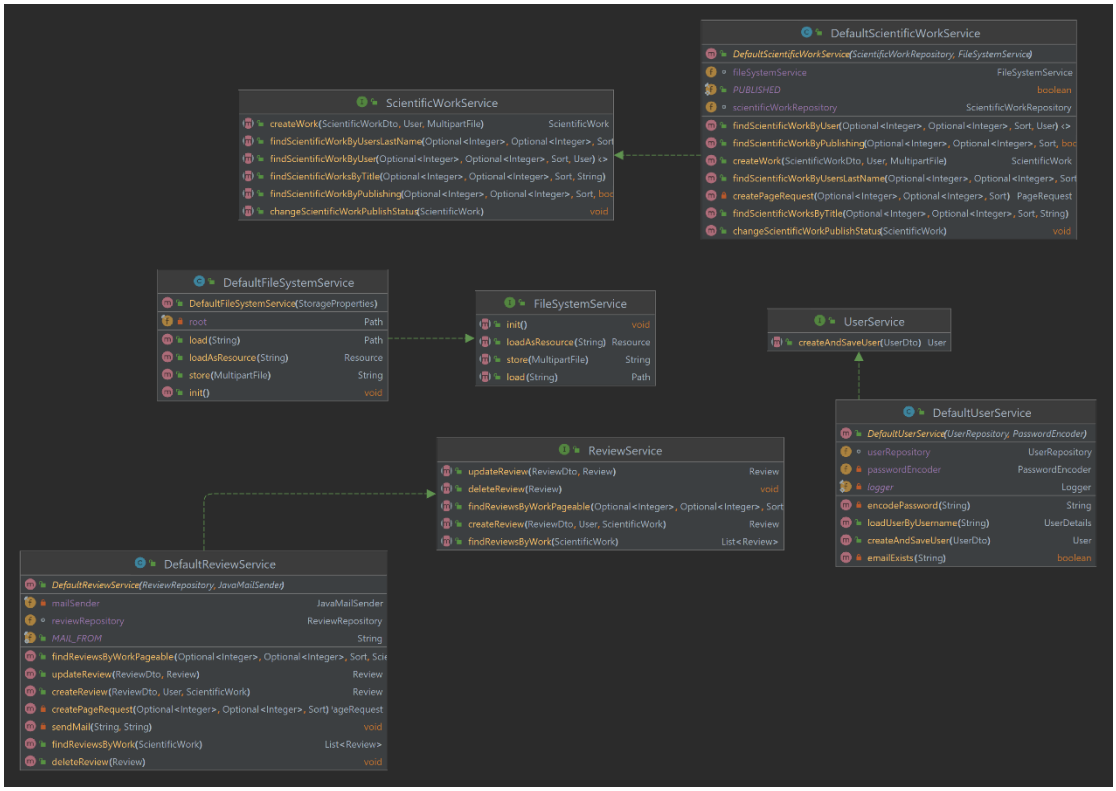

Рисунок 3.5 – Діаграма класів шару сервіс

3.2 Структура проекту веб-застосунку

Першим етапом створення веб-застосунку є розробка бази даних. Для роботи зі схемою бази даних використовувалось СУБД MySQL Workbench. Було створено 4 таблиці: user, user_role, scientific_work та review.

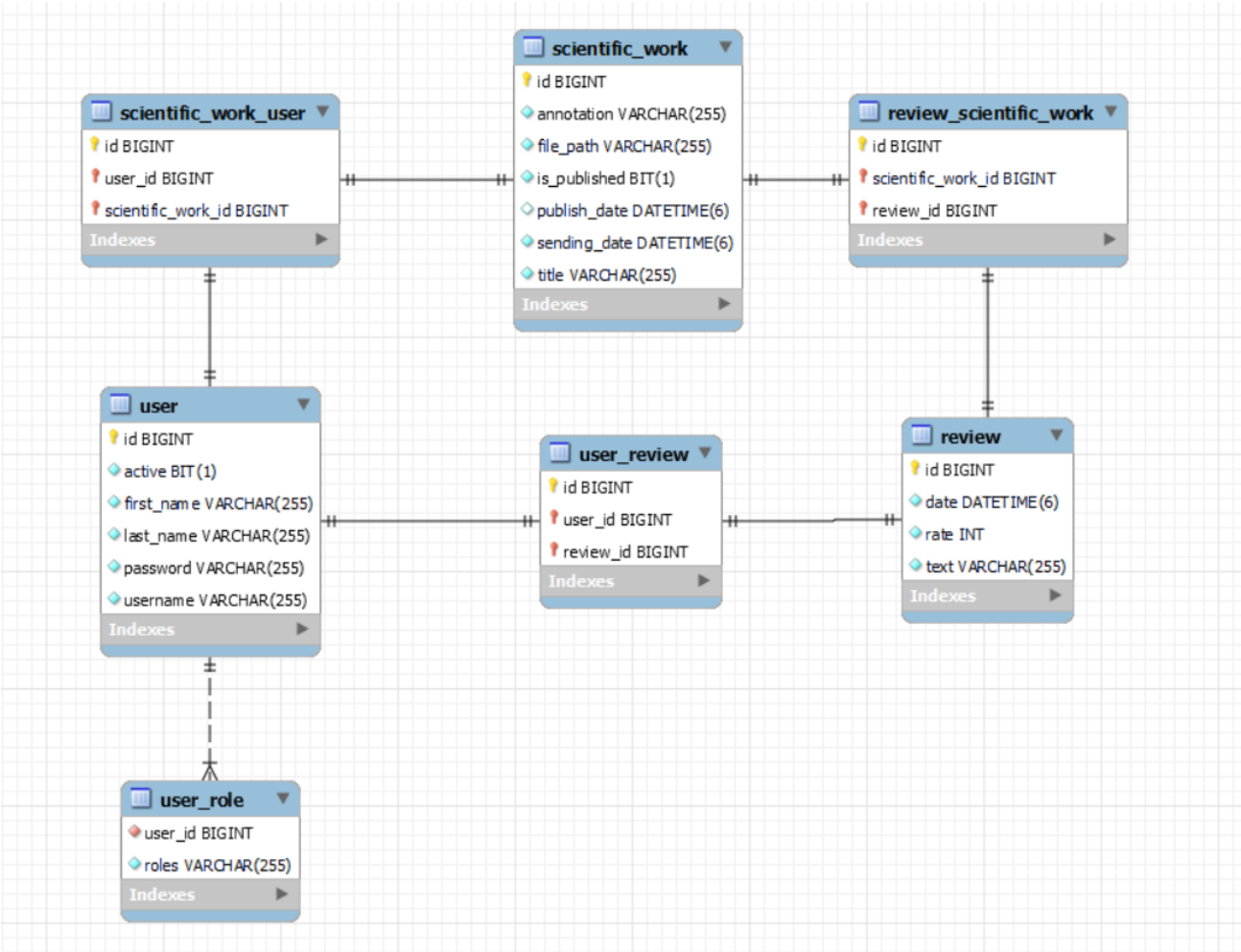

Рисунок 3.6 - Схема бази даних веб-застосунку

Таблиця user виконує дві ролі в застосунку: репрезентація самого користувача та його облікового запису. За відображення особистої інформації

користувача відповідають атрибути імені та прізвища користувача, а для репрезентації облікового запису - адреса електронної пошти та пароль.

У веб-застосунку користувачі матимуть дві ролі: адміністратор та простий користувач. Для їх розподілення є таблиця user_role. В ній є два атрибути: ключ користувача та його роль.

За зберігання наукових робіт відповідає таблиця scientific_work. Вона зберігає такі дані: назва роботи, анотація (короткий опис), шлях до файлу наукової роботи на сервері та користувач, що надіслав дану роботу. Також зберігаються дати надсилання роботи та її публікації адміністратором.

Останньою таблицею є review. В ній зберігаються дані про огляди робіт користувачами – текст огляду та оцінка роботи. Огляд робиться одним користувачем для однієї роботи, тому він містить зовнішні ключи цих таблиць.

Слід зазначити, що для ідентифікації кожного запису всіх таблиці є ідентифікатор. Ідентифікатор є послідовним числом, що генерується СУБД. Хоч він і не відображає жодної інформації про запис, проте такий ключ є незмінним та продуктивним для пошуку по таблиці.

Найважливішим етапом створення веб-застосунку є написання програмного коду. Для полегшення роботи з кодом використовувалось інтегроване середовище IntelliJ Idea. Воно має хорошу підтримку мови програмування Java та легко взаємодіє з системою керування версіями програми git.

DTO - це об’єкт, що переносить дані між процесами. Основною причиною їх використання є те, що зв’язок між процесами зазвичай здійснюється за допомогою віддалених інтерфейсів, де кожний виклик є дорогою операцією в плані пам’яті і часу [32]. Для зменшення кількості таких взаємодій використовують DTO, що зберігає в собі лиш потрібні дані, а не дані цілого

класу-домену. Тобто передаються тільки ті атрибути, з якими користувач буде взаємодіяти, (див. додаток Б).

Також в об’єктах DTO і відбувається основна валідація на серверному рівні. Валідація відбувається завдяки бібліотечним та написаним анотаціям. Наприклад, в програмному застосунку, що описаний в цій роботі, представлені анотації для перевірки на правильність будови адреси поштової скриньки та перевірки правильності паролю при його підтвердженні.

Репозиторій собою являє маркер-інтерфейс, що перехоплює типи, з якими потрібно працювати. Цей рівень забезпечує безпосередню роботу з базою даних. У якості аргументів типу приймається клас домену, яким потрібно керувати, а також тип ідентифікатора класу домену. Кожний метод CRUD («створення, читання, оновлення і вилучення») інтерфейсу є запитом до бази даних. Тобто запит формується з імені методу або за допомогою окремо створеного запиту в анотації до методу. Таке легке рішення взаємодії з базою даних можливе завдяки механізму Spring Data.

```
public interface UserRepository extends JpaRepository<User, Long> {
    2 usages
    User findByUsername(String username);
}
```

Рисунок 3.7 - Репозиторій користувачів

Сервісний рівень є проміжним рівнем у проекті, що забезпечую зв’язок між репозиторієм та контролером. У ньому містить основна бізнес-логіка застосунку. Даний рівень отримує дані з репозиторію у вигляді об’єктів класів-доменів, обробляє їх потрібним чином і передає їх в вигляді вже DTO в контролер. Також передача даних і відбувається в протилежному напрямку, від контролера до

репозиторію. Важливою частиною такого функціоналу є валідація даних перед їх збереженням в базу даних. Основний функціонал, який реалізовано в сервісах:

- зберігання та опрацювання файлів наукових робіт;
- опрацювання оглядів перед їх зберіганням, зміною чи видаленням (див. додаток В);
- опрацювання облікових записів користувачів перед їх зберіганням чи зміною;
- опрацювання наукових робіт перед їх зберіганням, зміною чи видаленням;
- формування та надсилання листів сповіщень;
- шифрування паролів користувачів;
- пагінація та сортування даних.

Контролер виконує декілька функцій в програмі. Головними обов'язками такого типу класів є перехоплення запитів та їх перетворення на внутрішню структуру даних та надсилання даних з серверу до інтерфейсу клієнту і навпаки, отримання даних від представлення та їх передача в сервіс для подальшої обробки та збереження в базу даних. (див. додаток Г).

Представлення – це інтерфейс користувача. За допомогою нього відвідувач вебзастосунку взаємодіє з програмою. На цьому рівні відбувається первинна валідація даних, що вводить клієнт в різних веб-формах. Також в представленні відображаються дані , що надходять з серверу.

Окрім загальної схеми структури програми, є ще допоміжні класи конфігурацій. Дані класи потрібні для додаткових налаштувань системи таких, як безпека, локалізація, конфігурація Thymeleaf, тощо. Наприклад, в налаштуваннях безпеки описано, які сторінки можуть відвідати користувачі без авторизація та аутентифікації, а в конфігураціях шифрування паролю обирається алгоритм шифрування.

Під час написання кожного рівня програми відбувалося модульне тестування. Кожний тест передбачає собою перевірку кожного публічного методу з конкретними вхідними та вихідними даними. Наприклад, на рисунку 3.4 представлено програмний код перевірки того, чи буде викликатися метод репозиторію на знаходження наукової роботи при виклику відповідного методу в сервісі.

```
    @Test
    public void shouldFindReviewsByWork(){
        reviewService.findReviewsByWork(scientificWork);
        verify(reviewRepository).findReviewsByScientificWorkId(WORK_ID);
    }
}
```

Рис 3.8. Приклад модульного тесту

Отже, проект є структурованим та багаторівневим. Структурованість дає проекту відкритість до доповнень і зрозумілість. Багаторівневість забезпечує хорошу валідацію даних, що зберігаються в базу даних і менше пов’язує структури між собою, що надає можливість подальшого накопичення функціоналу.

3.3 Використання системи

Перш, ніж отримати доступ до повного функціоналу сайту, користувач повинен авторизуватися. Для того щоб це зробити, він повинен ввести свій логін, тобто адресу електронної пошти, та пароль свого облікового запису.

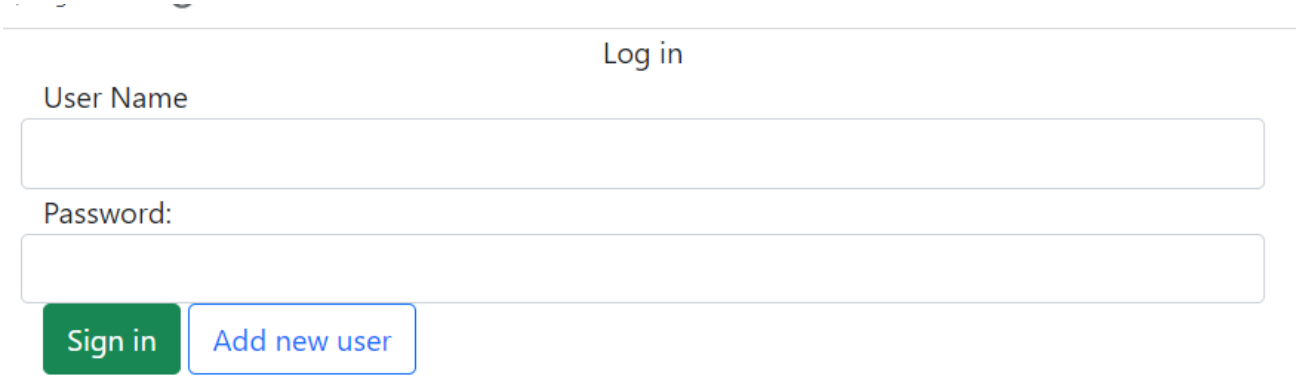

Рисунок 3.9 - Форма авторизації користувача.

Якщо користувача з введеними логіном та паролем немає в базі даних, то користувач не пройде авторизацію і доступу до веб-застосунку він не матиме.

Для створення нового облікового запису користувач повинен перейти на сторінку реєстрації. На цій сторінці користувач повинен ввести своє ім’я, прізвище, адресу поштової скриньки, якої ще немає в базі даних, та пароль.

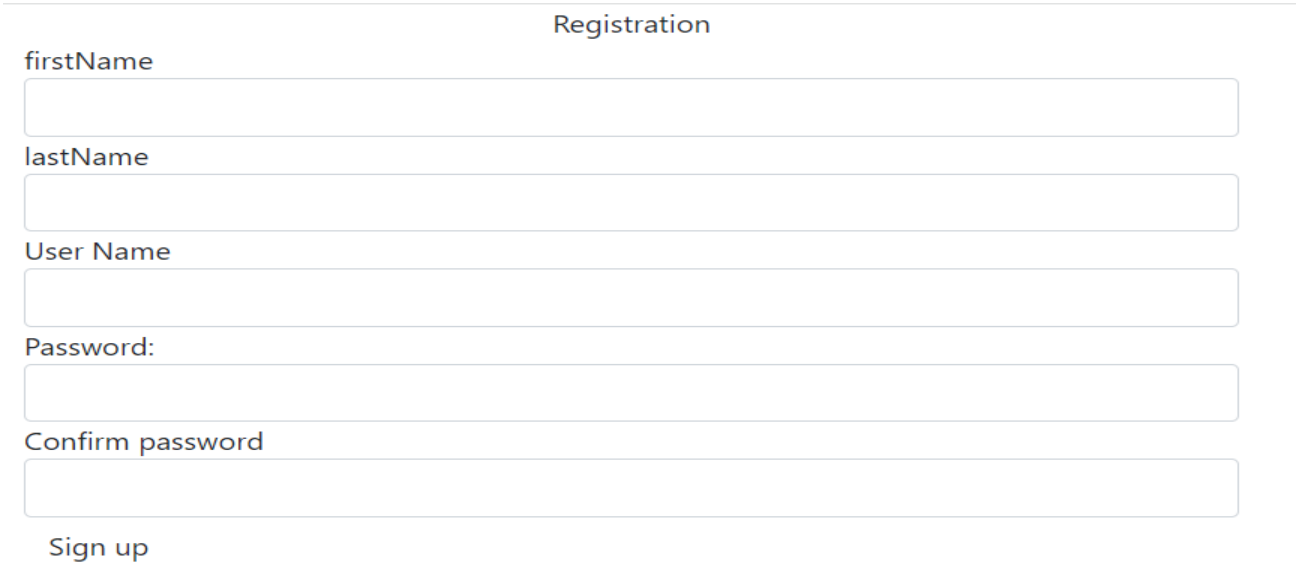

Рисунок 3.10 - Форма реєстрації користувача

В формі реєстрації, як і в інших формах, працює валідація. Ім’я і прізвище повинне бути відповідної довжини, а пошта відповідати правильному шаблону. Також реєстрація не відбудеться якщо користувач не зможе двічі повторити однаковий пароль. Це збільшує шанс того, що користувач одразу ж не забуде власний пароль для входу.

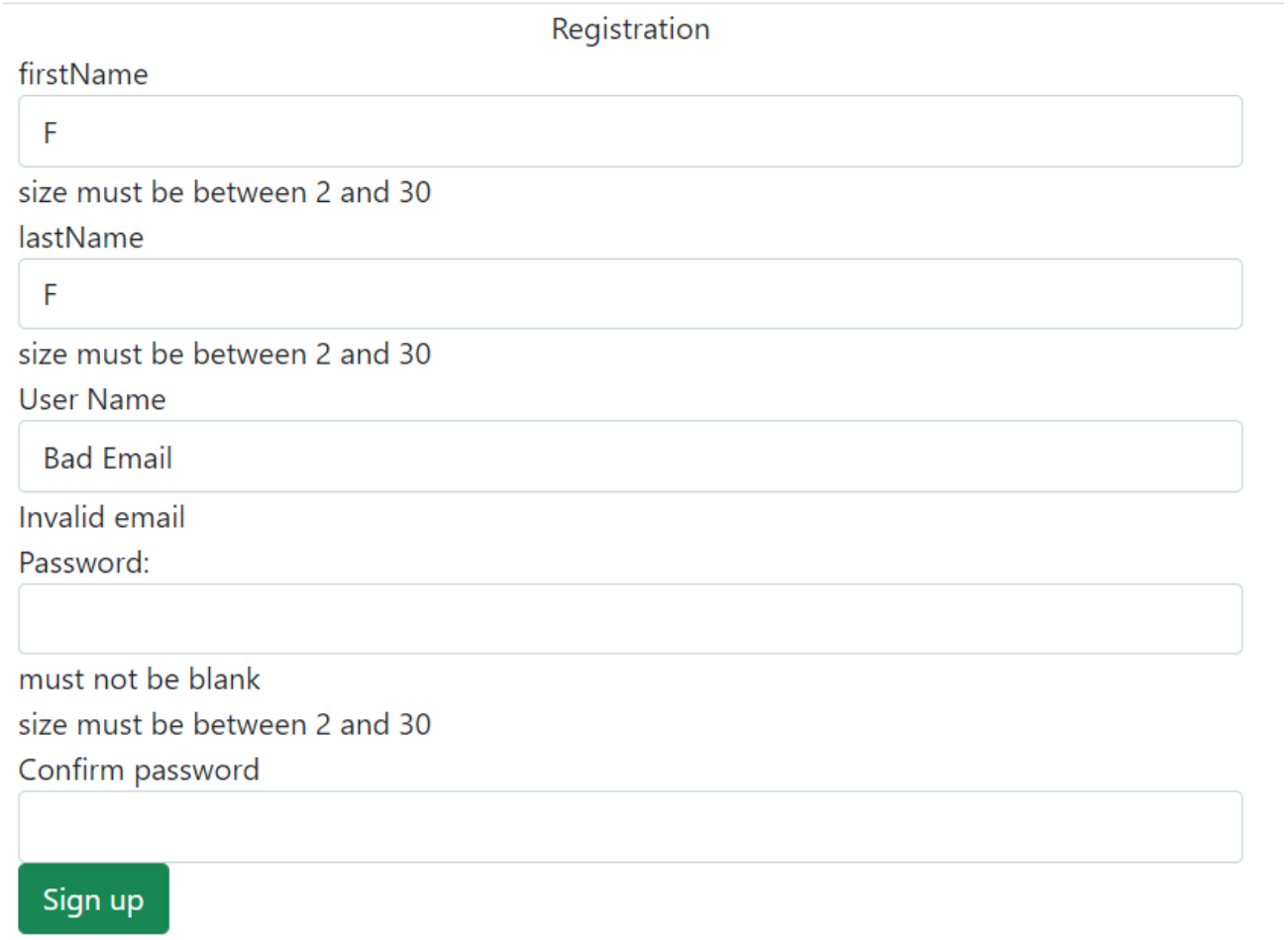

Рисунок 3.11 - Форма реєстрації користувача при неправильних даних

Як тільки користувач зареєструється, в рядку навігації веб-застосунку буде відображено основні сторінки, на які користувач може перейти. Також на сайті працює локалізація. Сторінки можна переглядати як на англійській, так і на українській мовах.

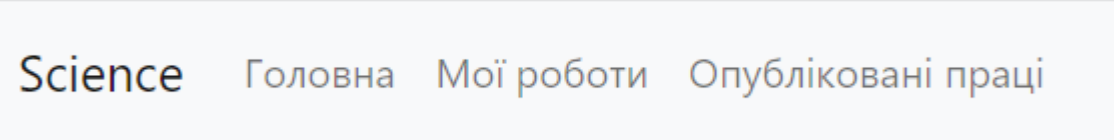

Рисунок 3.12 - Рядок навігації веб-застосунку

На сторінці «Мої роботи» можна завантажити свою наукову роботу. Для цього потрібно ввести назву своїй наукової роботи, анотацію або короткий опис та завантажити сам файл роботи. Файли наукових робіт зберігаються на серверному комп’ютері.

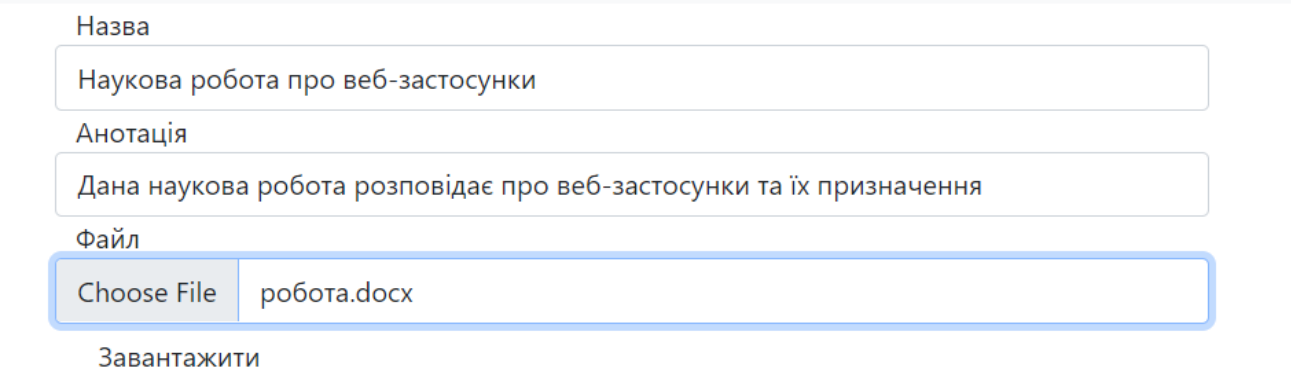

Рисунок 3.13 - Форма завантаження наукової роботи

Є можливість переглядати всі свої наукові роботи, що були відправлення на публікацію. Також на сторінці є можливість відсортовувати їх по даті відправлення на перевірку. Можна побачити, що на сторінці також працює пагінація. Вона потрібна для того, щоб розбивати контент на різні сторінки, щоб він не відображався весь одним пластом на сторінці.

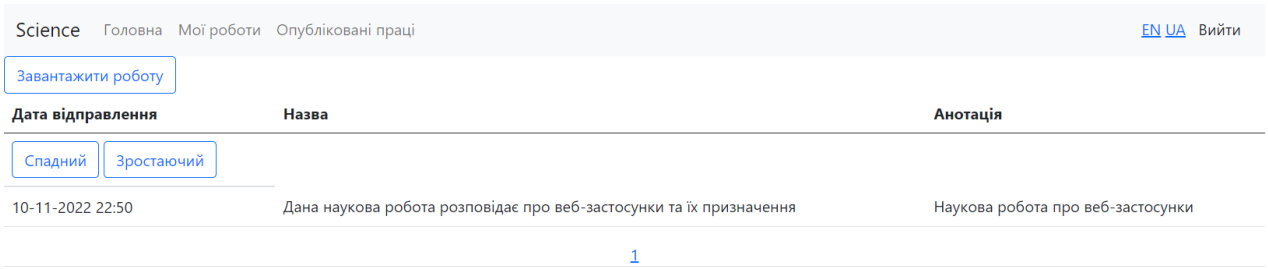

Рисунок 3.14 - Сторінка перегляду своїх робіт

Перш ніж бути опублікованою, робота повинна пройти перевірку адміністратору сайту. До сторінки з новими роботами мають доступ тільки користувачі з роллю адміністратора. Передивляючись файл роботи, його назву та короткий опис , він робить висновок чи робота валідна і після цього вже публікує коректні роботи.

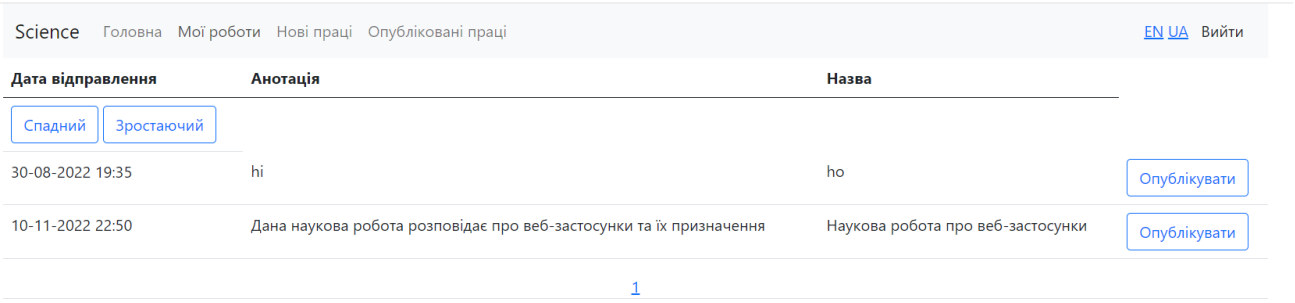

Рисунок. 3.15 - Сторінка з надісланими роботами до адміністратора

Якщо адміністратор вважає роботу валідною, то він її публікує. Перейшовши на сторінку опублікованих праць, можна побачити інформацію про них, а саме їх дату відправлення, анотацію та назву. До кожної праці можна переглянути детальнішу інформацію про неї. Адміністратор також має можливість скасувати публікацію роботи, тому кнопку, що реалізує цей функціонал, бачать користувачі з відповідною роллю.

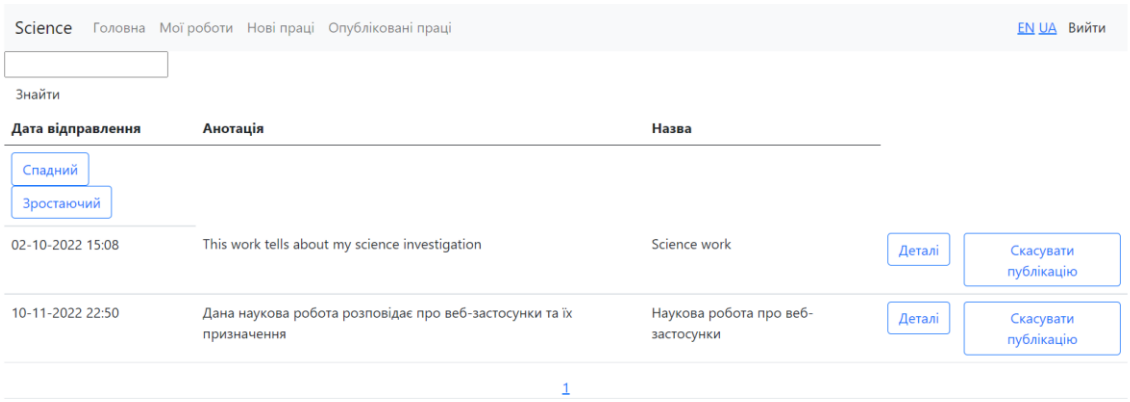

Рисунок 3.16 - Сторінка перегляду опублікованих праць

Також на даній сторінці є можливість пошуку роботи по її назві. Не потрібно писати всю назву роботи, достатньо ввести ключове слово теми, що вас цікавить.

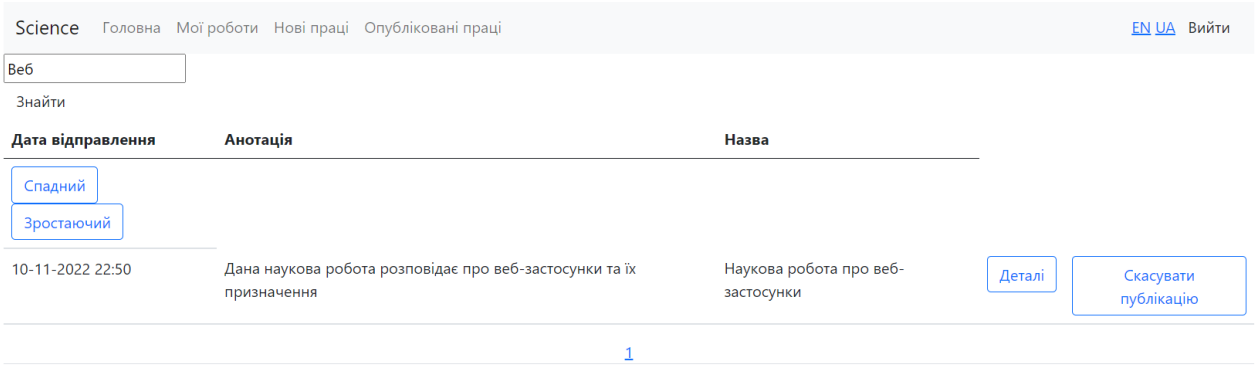

Рисунок 3.17 - Демонстрація роботи фільтра робіт по назві

В деталях роботи можна побачити головну інформацію про роботу, а також огляди на неї. Кожний користувач може робити тільки один огляд на одну роботу. Також користувачі можуть редагувати свої огляди з часом або видаляти їх. Є можливість завантажити файл роботи для подальшого ознайомлення з нею.

Щоб написати огляд на роботу користувач повинен заповнити поля тексту огляду та оцінки від одного до п’яти.

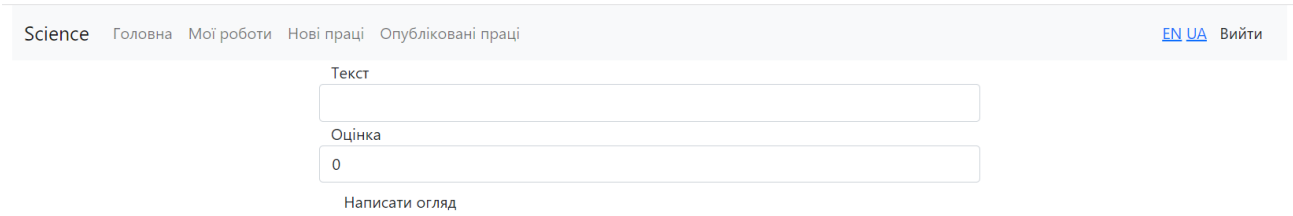

Рисунок 3.18 - Форма для написання огляду на роботу

Найкраще цей потік показує наступна схема:

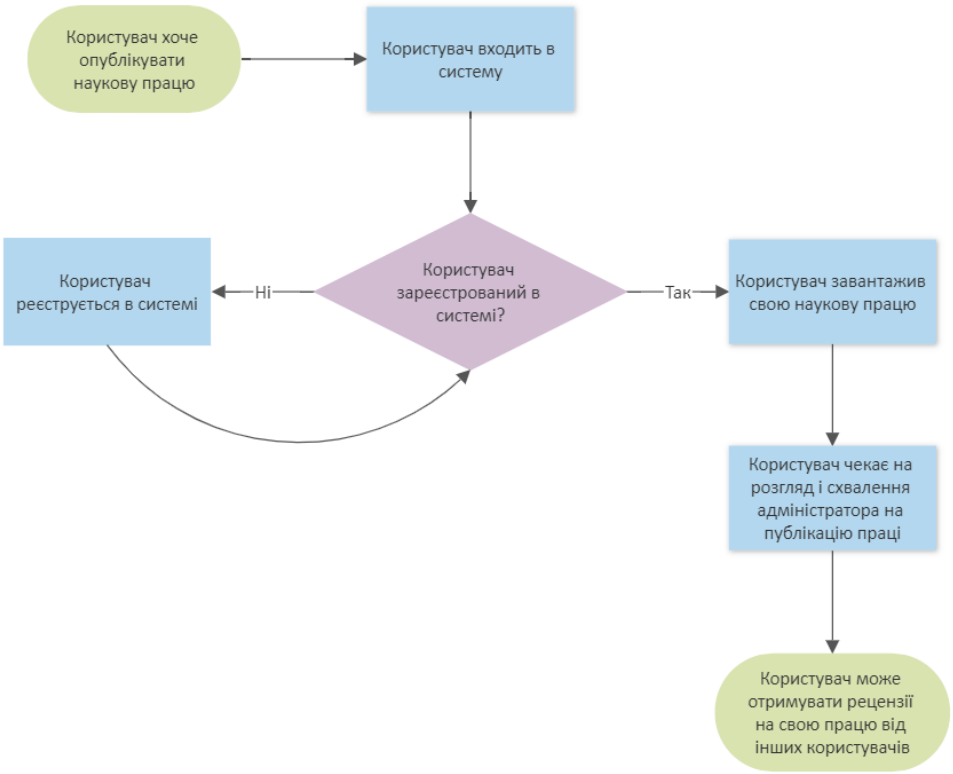

Схема 1 – потік публікації наукової праці

Коли користувач опубліковує свій огляд, то на поштову скриньку власника наукової роботи, на яку було написано рецензію, приходить лист-сповіщення.

Для зв’язку з розробником сайту в нижній частині сайту є дані про те як з ним зв’язатися. Дана інформація є на кожній сторінці веб-застосунку. В перспективах можливе створення окремої сторінки про розробників.

Якщо на сайті стається певний збій чи дії користувача приводять до певних проблем, то користувач побачить сторінку помилки. Це дозволяє не показувати користувачу розгортку стеку помилки, чим могли б скористатися зловмисники.

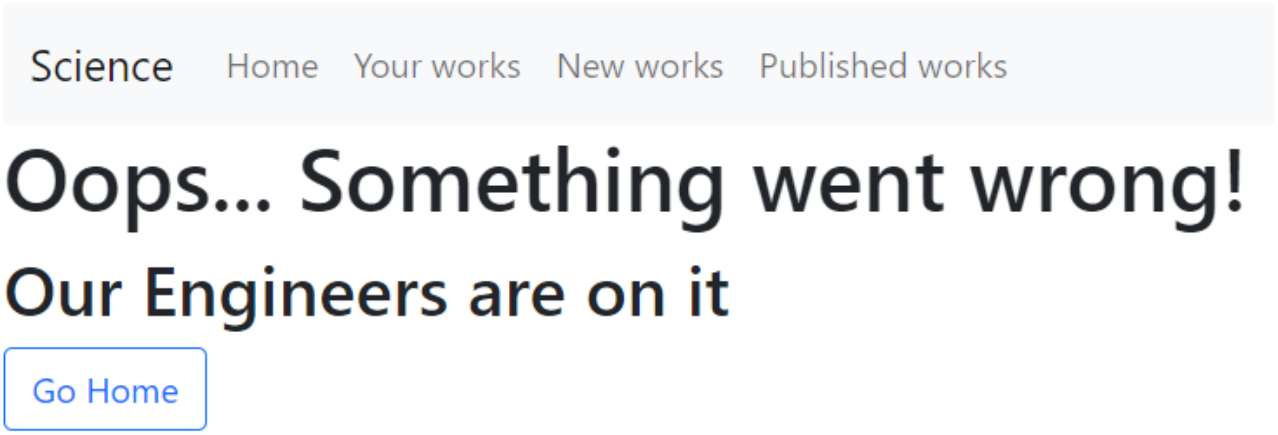

Рисунок 3.19 - Сторінка помилки

Отже, основний функціонал веб-застосунку реалізовано. Веб-застосунок є захищеним, оскільки потрібна реєстрація та авторизація, щоб могти користуватись головним його функціоналом. Користувачі можуть відправляти і переглядати наукові роботи на сайті. Якість робіт підтримується двома способами: верифікація зі сторони адміністратора та огляди зі сторони інших користувачів. Веб-застосунок відкритий до розширення, можлива реалізація додаткового функціоналу в майбутньому.

ВИСНОВКИ

Досліджено предметну область, здійснено огляд популярних наукометричних баз. Розглянуто різні типи наукових видань та види наукових робіт студентів. Проаналізовано існуючі системи для публікації наукових матеріалів, як українські, так і іноземні.

Проаналізовано доцільність та засади створення веб-застосунку архіву для публікацій наукових праць і змодельовано таку систему засобами UML.

Спроектовано базу даних та архітектуру застосунку. Структура програми є багаторівневою, що покращує захист та верифікацію даних, а також робить програмний код відкритим до покращень та розширення.

Розглянуто та обрано технології для створення програмної частини системи. Розроблено веб-застосунок для публікації наукових видань. Реалізовано інтерфейс та функціонал, що включає завантаження наукових робіт, їх публікація та перегляд, можливість написання оглядів на роботи та їх коригування. Доступ до застосунку вимагає реєстрацію та верифікацію нових користувачів. Адміністратор має можливість також керувати даними із СУБД.

Всі модулі програмного застосунку були модульно протестовані.

До перспективних напрямів можна віднести розширення прав адміністраторів, розширення функціоналу пошуку та перегляду наукових публікацій.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Наукова робота та її значення у житті студента: основні види науково-дослідної роботи. 2016. URL: https://www.nusta.edu.ua/2019/09/25 наукова-робота-та-її-значення-у-житті-с /.
2. Custom Essay Writers, Freelancers, and Other Paid Third Parties / Newton, Philip M., Lang, Christopher., 2016.
3. Основні види. Терміни та визначення. — К., 1995; Шейко В.М., Кушнаренко Н.М. Організація науково-дослідницької діяльності. — К., 2006.
4. Як писати тези: що таке тези та їх приклади. 2017. URL: https://naurok.ua/student/blog/yak-pisati-tezi-scho-take-tezi-i-h-prikladi.
5. Словник української мови: Том дев'ятий С / Редкол. І. К. Білодід та ін., редактори тому: І. С. Назарова, О. П. Петровська, Л. Г. Скрипник, Л. А. Юрчук — К.:"Наукова думка", 1978 (с.:670)
6. Науковий вісник Мукачівського державного університету. Серія Економіка. 2014. URL: https://economics-msu.com.ua/uk
7. Elsevier | An information analytics company. URL: https://www.elsevier.com/en-gb
8. Springer – International Publisher Science, Technology, Medicine. URL: https://www.springer.com/gp
9. Google Academy. URL: https://scholar.google.com/
10. Наукова періодика України. URL: http://journals.uran.ua/
11. Scopus Preview. 2004. URL: https://www.scopus.com/home.uri
12. Reuters, Thomson Overview - Web of Science" (Overview of coverage gleaned from promotional language.) Science. 2010. URL: https://clarivate.com/webofsciencegroup/solutions/web-of-science/
13. About Crossref. URL: https://www.crossref.org/services/crossmark/

14. About DOAJ. URL: https://doaj.org/about/
15. Java. URL: https://www.oracle.com/java/technologies/introduction-to-java.html
16. Lombok. URL: https://projectlombok.org/
17. The Good and the Bad of Java Programming. 2018. URL: https://www.altexsoft.com/blog/engineering/pros-and-cons-of-java-programming/
18. Apache Maven. URL: https://maven.apache.org/
19. Rob Harrop. Pro Spring 3 / Rob Harrop, Clarence Ho., 2012. – 880 c.
20. Spring Security. URL: https://spring.io/projects/spring-security
21. Buschmann, Frank Pattern-Oriented Software Architecture 1996. - 350 c.
22. Hibernate . URL: https://hibernate.org/
23. What is MVC Design Pattern. URL: https://www.educba.com/what-is-mvc-design-pattern/
24. ThymeLeaf . URL: https://www.thymeleaf.org/
25. Get started with Bootstrap. URL: https://getbootstrap.com/docs/5.2/getting-started/introduction/
26. CSS — Styling the web. URL: https://developer.mozilla.org/en-US/docs/Learn/CSS
27. MySQL. URL: https://dev.mysql.com/doc/
28. MySQL Workbench URL: https://www.mysql.com/products/workbench/
29. Automated Defect Prevention: Best Practices in Software Management / Dorota Huizinga, Adam Kolawa., 2007. – 454 c.
30. JUnit 5 User Guide. URL: https://junit.org/junit5/docs/current/user-guide/
31. Mockito. URL: https://site.mockito.org/
32. Data Transfer Object : URL: https://martinfowler.com/eaaCatalog/dataTransferObject.html

ДОДАТКИ

Додаток А. Приклад програмного коду класу-домену

```
import lombok.Getter;
import lombok.Setter;

import javax.persistence.*;
import javax.validation.constraints.NotNull;
import java.time.LocalDateTime;

@Entity
@Getter
@Setter
@Table(name = "review")
public class Review {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;

    @NotNull
    @Column(nullable = false)
    private String text;

    @NotNull
    @Column(nullable = false)
    private int rate;
```

```
    @NotNull
    @Column(nullable = false)
    private LocalDateTime date;

    @ManyToOne(fetch = FetchType.EAGER)
    @JoinColumn(name="user_id", nullable=false)
    private User user;

    @ManyToOne(fetch = FetchType.EAGER)
    @JoinColumn(name="scientific_work_id", nullable=false)
    private ScientificWork work;
}
```

Додаток Б. Приклад програмного коду Data Transfer Object

```
import lombok.Getter;
import lombok.Setter;

import javax.validation.constraints.Max;
import javax.validation.constraints.Min;
import javax.validation.constraints.NotBlank;

@Getter
@Setter
public class ReviewDto {

    @NotBlank
    private String text;

    @Min(1)
    @Max(5)
    private int rate;

    public ReviewDto(){}

    public ReviewDto( @NotBlank String text, @Min(1) @Max(5) int rate){
        this.rate = rate;
        this.text = text;
    }
}
```

Додаток В. Приклад програмного коду сервісу.

```
import lombok.AllArgsConstructor;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.PageRequest;
import org.springframework.data.domain.Sort;
import org.springframework.mail.SimpleMailMessage;
import org.springframework.mail.javamail.JavaMailSender;
import org.springframework.stereotype.Service;

import java.text.MessageFormat;
import java.time.LocalDateTime;
import java.util.List;
import java.util.Optional;

@AllArgsConstructor
@Service
public class DefaultReviewService implements ReviewService {

public static final String MAIL_FROM = "noreply@gmail.com";

ReviewRepository reviewRepository;
private final JavaMailSender mailSender;

@Override
```

```
public Review createReview(ReviewDto reviewDto, User user, ScientificWork scientificWork) {
Review review = new Review();
review.setDate(LocalDateTime.now());
review.setRate(reviewDto.getRate());
review.setText(reviewDto.getText());
review.setUser(user);
review.setWork(scientificWork);
sendMail(scientificWork.getUser().getUsername(),
MessageFormat.format("User {0} left review for your work {1}",
user.getUsername(),
scientificWork.getTitle()));
return reviewRepository.save(review);
}

@Override
public Review updateReview(ReviewDto reviewDto, Review review) {
review.setText(reviewDto.getText());
review.setRate(reviewDto.getRate());
return reviewRepository.save(review);
}

@Override
public void deleteReview(Review review) {
reviewRepository.delete(review);
}
```

```
@Override
public Page<Review> findReviewsByWorkPageable(Optional<Integer> page,
Optional<Integer> size,
Sort sort, ScientificWork scientificWork) {
return
reviewRepository.findReviewsByScientificWorkIdPageable(createPageRequest(page,
size, sort),
scientificWork.getId());
}

@Override
public List<Review> findReviewsByWork(ScientificWork scientificWork) {
return
reviewRepository.findReviewsByScientificWorkId(scientificWork.getId());
}

private PageRequest createPageRequest(Optional<Integer> page,
Optional<Integer> size, Sort sort) {
return sort == null ?
PageRequest.of(page.orElse(DEFAULT_PAGE_NUMBER) - 1,
size.orElse(DEFAULT_SIZE)) :
PageRequest.of(page.orElse(DEFAULT_PAGE_NUMBER) - 1,
size.orElse(DEFAULT_SIZE), sort);
}

private void sendMail(String emailTo, String messageText){
SimpleMailMessage message = new SimpleMailMessage();
```

```
message.setFrom(MAIL_FROM);
message.setTo(emailTo);
message.setSubject("New review for your scientific work!");
message.setText(messageText);

mailSender.send(message);
}
}
```

Додаток Г. Приклад програмного коду контролеру.

```
import lombok.AllArgsConstructor;
import org.springframework.core.io.Resource;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Sort;
import org.springframework.http.HttpHeaders;
import org.springframework.http.ResponseEntity;
import org.springframework.security.access.prepost.PreAuthorize;
import org.springframework.security.core.annotation.AuthenticationPrincipal;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.*;

import javax.validation.Valid;
import java.nio.file.Paths;
import java.util.Optional;

@Controller
@AllArgsConstructor
public class ReviewController {

    ReviewService reviewService;
    ScientificWorkService scientificWorkService;
    FileSystemService fileSystemService;
```

```
        @GetMapping("/published-works/{work}")
        public String getWorkPage(@AuthenticationPrincipal User user,
                                  @RequestParam("page") Optional<Integer> page,
                                  @RequestParam("size") Optional<Integer> size,
                                  @RequestParam(value = "sort", required = false) String
sortBy,
                                  @RequestParam(value = "nameBy", required = false) String
nameBy,
                                  @PathVariable("work") ScientificWork scientificWork,
                                  Model model){
            model.addAttribute("work",scientificWork);
            model.addAttribute("userId", user.getId());
            model.addAttribute("reviewCreated",
isUserCreatedReviewForWork(user,scientificWork));
            model.addAttribute("file", getFileName(scientificWork));
            Sort sort = ControllerUtils.getSort(sortBy , nameBy , model);
            Page<Review>                    reviews                    =
reviewService.findReviewsByWorkPageable(page, size, sort, scientificWork);
            int totalPages = reviews.getTotalPages();
            ControllerUtils.pageNumberCounts(totalPages , model);
            model.addAttribute("reviews", reviews);
            return "workReviews";
        }

        @GetMapping("/published-works/{work}/create-review")
        public String getCreateReviewPage(@AuthenticationPrincipal User user,
                                          @PathVariable("work") ScientificWork scientificWork,
```

```
                                Model model){
        model.addAttribute("work",scientificWork.getId());
        ReviewDto reviewDto = new ReviewDto();
        model.addAttribute("review", reviewDto);
        return isUserCreatedReviewForWork(user, scientificWork) ?
"redirect:/published-works/{work}" : "createReview";
    }

    @PostMapping("/published-works/{work}/create-review")
    public String createReview(@AuthenticationPrincipal User user,
                               @PathVariable("work") ScientificWork scientificWork,
                               @Valid ReviewDto reviewDto,
                               BindingResult bindingResult,
                               Model model){
        if(bindingResult.hasErrors()){
            model.addAttribute("work",scientificWork.getId());
            model.addAttribute("review", reviewDto);
            return "createReview";
        }
        reviewService.createReview(reviewDto,user,scientificWork);
        return "redirect:/published-works/{work}";
    }

    @PreAuthorize("#review.getUser().getUsername()                    ==
authentication.principal.username")
    @GetMapping("/published-works/{work}/update-review/{review}")
```

```
        public String getUpdateReviewPage(@PathVariable("work") ScientificWork
scientificWork,
                                          @PathVariable("review") Review review,
                                          Model model){
            model.addAttribute("work",scientificWork.getId());
            ReviewDto reviewDto = new ReviewDto(review.getText(),
review.getRate());
            model.addAttribute("review", reviewDto);
            model.addAttribute("reviewId", review.getId());
            return "updateReview";
        }
        @PreAuthorize("#review.getUser().getUsername()                    ==
authentication.principal.username")
        @PostMapping("/published-works/{work}/update-review/{review}")
        public String updateReview(@PathVariable("work") ScientificWork
scientificWork,
                                   @PathVariable("review") Review review,
                                   @Valid ReviewDto reviewDto,
                                   BindingResult bindingResult,
                                   Model model){
            if(bindingResult.hasErrors()){
                model.addAttribute("work",scientificWork.getId());
                model.addAttribute("review", reviewDto);
                model.addAttribute("reviewId", review.getId());
                return "updateReview";
            }
            reviewService.updateReview(reviewDto, review);
```

```
            return "redirect:/published-works/{work}";
        }
        @PreAuthorize("#review.getUser().getUsername()                    ==
authentication.principal.username")
        @GetMapping("/published-works/{work}/delete-review/{review}")
        public String deleteReview(@PathVariable("review") Review review){
            reviewService.deleteReview(review);
            return "redirect:/published-works/{work}";
        }
        @GetMapping("/published-works/{work}/{filename:.+}")
        @ResponseBody
        public ResponseEntity<Resource> serveFile(@PathVariable String filename)
{
            Resource file = fileSystemService.loadAsResource(filename);
            return
ResponseEntity.ok().header(HttpHeaders.CONTENT_DISPOSITION,
                    "attachment; filename=\"" + file.getFilename() + "\"").body(file);
        }
        private boolean isUserCreatedReviewForWork(User user, ScientificWork
work){
            return reviewService.findReviewsByWork(work).stream()
                    .anyMatch(review -> review.getUser().getId().equals(user.getId()));
        }
        private String getFileName(ScientificWork scientificWork){
            return Paths.get(scientificWork.getFilePath()).getFileName().toString();
        }
    }
```