МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Сумський державний університет

Факультет електроніки та інформаційних технологій

Кафедра комп'ютерних наук

«До захисту допущено»

В.о. завідувача кафедри

_____ Ігор ШЕЛЕХОВ

(підпис)

09 червня 2023 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня бакалавр

зі спеціальності 122 – Комп'ютерних наук,

освітньо- професійної програми «Інформатика»

на тему: «Інформаційне та програмне забезпечення шкільної онлайн бібліотека»

здобувача групи ІН-92 Остапенка Сергія Сергійовича

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

_____ Сергій ОСТАПЕНКО

(підпис)

(підпис)

Кандидат технічних наук Артем КОРОБОВ _____

(підпис)

Суми – 2023

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«Затверджую»
В.о. завідувача кафедри
_____ Ігор ШЕЛЕХОВ
(підпис)

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня бакалавра

зі спеціальності 122 - Комп'ютерних наук, освітньо-професійної програми «Інформатика»
здобувача групи ІН-92 Остапенка Сергія Сергійовича

1. Тема роботи: «Інформаційне та програмне забезпечення шкільної онлайн бібліотека» затверджую наказом по СумДУ від *«01» червня 2023 р. № 0475-VI*
2. Термін здачі здобувачем кваліфікаційної роботи *до 09 червня 2023 року*
3. Вхідні дані до кваліфікаційної роботи _____
4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) *1) Аналіз предметної області, формування та визначення мети дослідження. 2) Розгляд технологій, що використовуються для створення веб-додатків. 3) Розроблення веб-додатку для спільного користування автомобілями. 4) Аналіз отриманих результатів.*
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)_____
6. Консультанти до проєкту (роботи), із значенням розділів проєкту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання «____» _____ 20 ___ р.

Завдання прийняв до виконання _____ Керівник _____
(підпис) (підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Термін виконання	Примітка
1	*Аналіз предметної області, постановка й формування завдань дослідження*		
2	*Огляд технологій, що використовуються для створення веб-додатків*		
3	*Розробка веб-додатку для шкільної бібліотеки*		
4	*Аналіз отриманих результатів*		
5	*Оформлення пояснювальної записки до кваліфікаційної роботи*		

Здобувач вищої освіти _____ Керівник _____
(підпис) (підпис)

АНОТАЦІЯ

Записка: 40 стр., 20 рис., 8 використаних джерел.

Обґрунтування актуальності теми роботи – Тема кваліфікаційної роботи є актуальною, оскільки присвячена розв’язанню важливої практичної задачі створення веб-додатка для шкільної бібліотеки.

Об’єкт дослідження – процес автоматизації та поліпшення роботи шкільної бібліотеки.

Мета роботи – розробка веб-додатку та інформаційної системи для шкільної бібліотеки.

Методи дослідження – аналіз потреб користувачів, проектування інтерфейсів, розробка веб-додатків та тестування функціоналу.

Результати – розроблено інформаційну систему, яка дозволяє учням, вчителям та бібліотекарям зручно та ефективно користуватися шкільною бібліотекою. Система забезпечує доступ до каталогу книг, можливість резервування та видачі книг, а також підтримує електронне зберігання та доступ до книг у цифровому форматі.

ВЕБ ДОДАТОК, ІНФОРМАЦІЙНА СИСТЕМА, JAVA, SPRING, POSTGRESQL

Зміст

ВСТУП 5
1 АНАЛІТИЧНИЙ ОГЛЯД 6
1.1 Сучасний стан 6
1.2 Аналіз аналогічних проєктів 9
1.3 Постановка задачі 15
1.4 Технічне завдання 16
2 ВИБІР МЕТОДІВ РОЗВ'ЯЗАННЯ ЗАДАЧІ 17
2.1 Вибір стеку технологій 17
2.2 Формування технічного завдання на основі стеку технологій 22
3 ІНФОРМАЦІЙНЕ ТА ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ 24
3.1 Створення проекту 24
3.2 Архітектура проекту 26
3.3 Взаємодія з базою данних 29
3.4 Взаємодія з view 31
3.5 Розробка інтерфейсу проекту 34
3.6 Тестування 38
ВИСНОВКИ 39
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 40

ВСТУП

Актуальність. Тема кваліфікаційної роботи є актуальною, оскільки враховуючи світові тенденції, де все більше людей працюють та навчаються з дому, необхідність діджиталізації шкіл стає надзвичайно актуальною. Особливо в цьому контексті шкільні бібліотеки відіграють важливу роль у забезпеченні доступу до навчальних матеріалів. Розробка та впровадження онлайн-платформи для шкільної бібліотеки стає кроком уперед у цифровій трансформації освіти.

Об’єкт дослідження. процес автоматизації та поліпшення роботи шкільної бібліотеки.

Предмет дослідження. Методологія розробки веб-додатків для поліпшення роботи шкіл.

Гіпотеза. Діджиталізація шкільної бібліотеки передбачає використання інформаційної технології для автоматизації процесу надання доступу до шкільних навчальних матеріалів. За допомогою спеціального веб-додатка, шкільна бібліотека зможе полегшить взаємодію між учнями, вчителями та бібліотекарями, сприятиме швидкому пошуку та доступу до потрібної інформації, роблячи шкільну бібліотеку сучасною та ефективною у навчальному процесі.

Новизна. Розроблене програмне рішення для шкільної бібліотеки має потенціал покращити ефективність та функціональність інформаційної системи. Завдяки чому, внутрішні процеси бібліотеки можуть бути автоматизовані, що сприятиме швидкому та зручному доступу до шкільних матеріалів.

Структура. Дана робота складається зі вступу, аналітичного огляду, вибір методу розв’язання поставленої задачі, опису програмного забезпечення інформаційної системи, висновків, списку використаних джерел та додатків.

1 АНАЛІТИЧНИЙ ОГЛЯД

1.1 Сучасний стан

Розробка інформаційної системи є процесом створення комплексної програмної системи, яка забезпечує збір, обробку, зберігання та передачу інформації з метою оптимізації певних процесів в організації чи установі. Інформаційна система може бути розроблена для вирішення різних завдань, таких як автоматизація бізнес-процесів, керування ресурсами, обробка та аналіз даних, забезпечення комунікації та спільної роботи між користувачами.

При розробці інформаційної системи необхідно враховувати потреби та вимоги користувачів, а також особливості діяльності організації. Це включає в себе проектування архітектури системи, вибір технологій, розробку програмного забезпечення, тестування та впровадження системи.

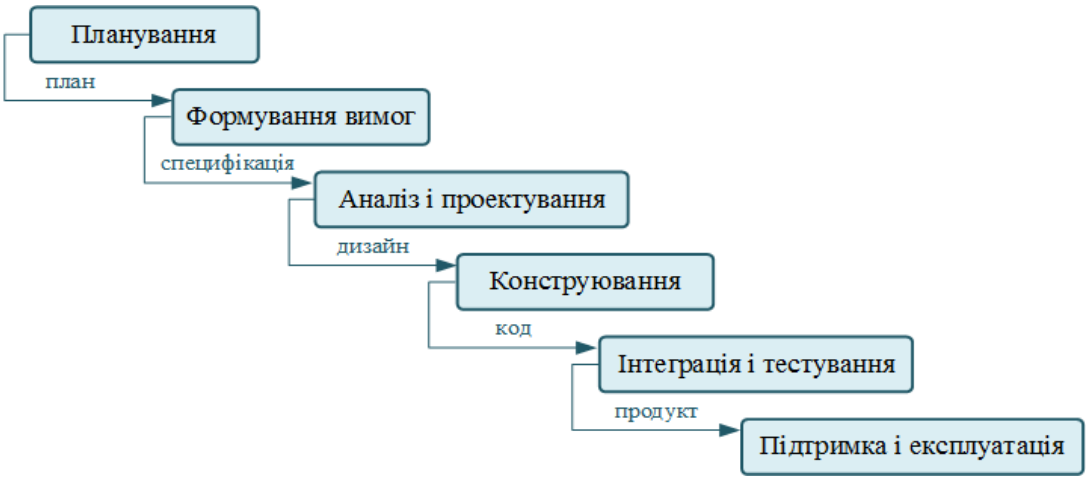

Рисунок 1.1 – Етапи проектування інформаційних систем

Розробка інформаційної системи є комплексним процесом, який включає в себе різні етапи та аспекти. Одним із ключових аспектів розробки є створення зручного та ефективного інтерфейсу, який дозволяє користувачам легко та зручно взаємодіяти з системою. Розробка зручного інтерфейсу полягає у визначенні та реалізації способів, за допомогою яких користувачі зможуть взаємодіяти з різними функціями та можливостями системи. Це може включати створення веб-

додатків, які дозволяють користувачам отримувати доступ до системи через веб-браузер, або мобільних додатків, які надають можливість використання системи на мобільних пристроях.

Важливим елементом розробки інформаційної системи є створення та керування базами даних. База даних є основою системи, де зберігається інформація, необхідна для функціонування та виконання завдань системи. Розробка баз даних включає визначення структури даних, встановлення зв'язків між ними та вибір оптимального методу зберігання та доступу до даних.

Крім того, розробка інформаційної системи може включати створення інших компонентів, які допомагають забезпечити її функціональність та продуктивність. Наприклад, це можуть бути алгоритми обробки даних, модулі аналітики, системи безпеки, механізми забезпечення взаємодії з іншими системами тощо. Розробка цих компонентів вимагає визначення вимог, проектування, реалізації та тестування.

В результаті розробки інформаційної системи досягається створення комплексного та збалансованого рішення, яке забезпечує зручну та ефективну роботу з інформацією. Розроблена інформаційна система сприяє автоматизації процесів, підвищенню продуктивності та забезпеченню якісного аналізу та обробки даних.

Розробка інформаційної системи для шкільної бібліотеки означає створення цифрового середовища, де учні, вчителі та бібліотекарі можуть спілкуватися, отримувати доступ до електронних ресурсів, проводити дистанційне навчання та спільні проекти. Це може включати розробку веб-порталу бібліотеки, створення електронного каталогу, публікацію електронних книг, журналів та інших матеріалів, а також організацію електронного обміну документами.

Результатом діджиталізації шкільної бібліотеки є полегшений доступ до актуальних джерел інформації, розширені можливості навчання та саморозвитку учнів, підвищена ефективність роботи вчителів та бібліотекарів. Впровадження

інформаційних технологій у бібліотеку стимулює інноваційність, активну взаємодію та співпрацю між всіма учасниками навчального процесу та сприяє підготовці учнів до сучасного цифрового світу.

Розробка інформаційної системи, яка буде дружня до користувача, має вирішити ряд завдань. По-перше, це полегшення доступу до каталогу книг та інших ресурсів, що містяться в шкільній бібліотеці. Користувачам повинно бути зручно шукати необхідну літературу, переглядати описи книг, обкладинки та відгуки інших користувачів.

Крім того, інформаційна система повинна надати можливість реєстрації користувачів і створення особистих кабінетів. Учні та вчителі зможуть створити свій особистий профіль, де зберігатимуться їхні позичені книги, а також користуватися додатковими функціями, які допоможуть їм у навчанні та дослідженнях.

Одним із ключових аспектів розробки інформаційної системи є створення зручного та інтуїтивно зрозумілого інтерфейсу. Користувачі мають мати можливість легко орієнтуватися в системі, швидко знаходити необхідну інформацію та взаємодіяти з ресурсами бібліотеки. Це може бути досягнуто шляхом розробки зрозумілих меню, пошукових функцій, а також інтеграції з іншими платформами та сервісами, що сприятимуть зручності використання.

Все це сприятиме поліпшенню якості обслуговування користувачів шкільної бібліотеки та створенню зручних умов для навчання та саморозвитку. Інформаційна система стане необхідним інструментом для учнів та вчителів, який допоможе їм отримувати доступ до потрібної літератури, взаємодіяти зі спільнотою читачів та розширювати свої знання.

1.2 Аналіз аналогічних проєктів

Після докладного аналізу сумських шкіл з метою оцінки їх рівня діджиталізації та доступу до інформаційних ресурсів, було виявлено, що більшість шкіл у місті не досягли високого рівня цифрової трансформації та не мають достатньо розвинутих веб-сайтів чи інших онлайн-ресурсів. Навіть базові елементи цифрової присутності, такі як веб-сайти, відсутні в багатьох школах.

Зазначено, що лише обмежена кількість шкіл вже має цифрові ресурси та електронні бібліотеки. Однак, ці ресурси не є широкодоступними або функціональними (рис. 1.2). Це означає, що учні шкіл не мають повноцінного доступу до літературних джерел та бібліотечних послуг, що обмежує їх можливості з навчання та самостійного дослідження.

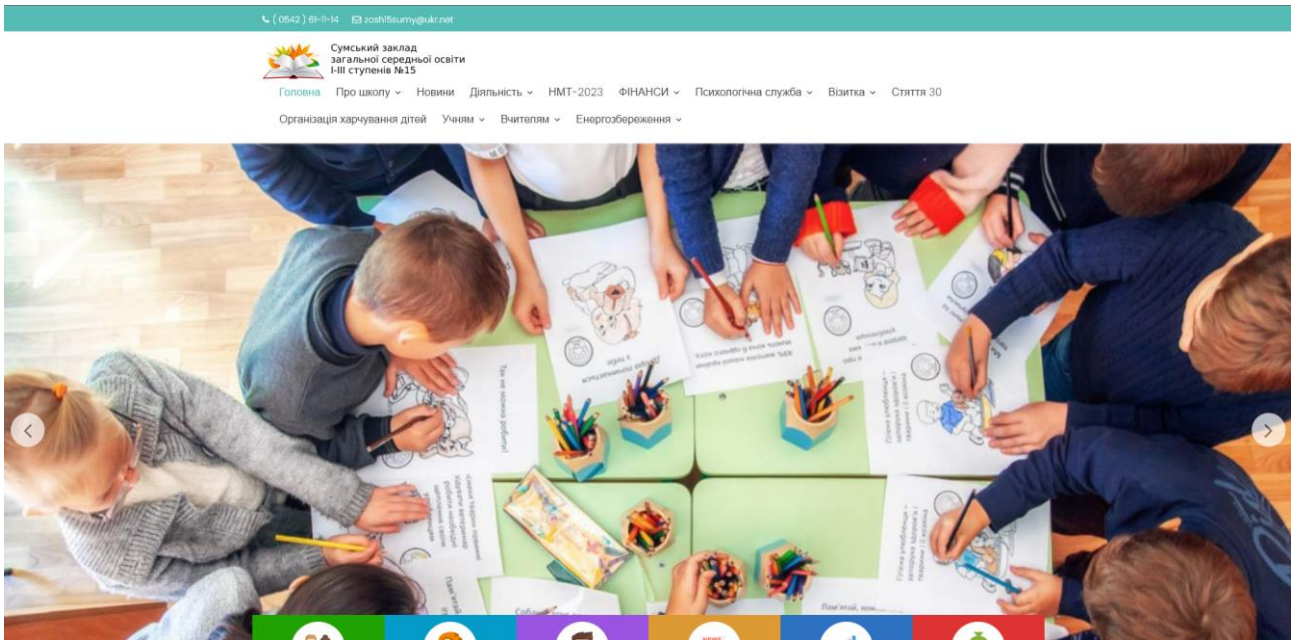

Рисунок 1.2 – Приклад сайту школи

Під час проведення аналізу сумських шкіл для оцінки рівня їх діджиталізації та доступу до інформаційних ресурсів, стало очевидним, що сумські університети значно випереджають школи у цьому питанні.

Університетські заклади особливо активно використовують широкий спектр цифрових ресурсів та розвинуті бібліотечні системи, що забезпечують студентам доступ до наукових джерел, журналів, електронних книг та інших інформаційних матеріалів.

Це свідчить про те, що університети в Сумах активно використовують інформаційні технології з метою покращення освітнього процесу та надання студентам широких можливостей для навчання, досліджень та саморозвитку. Завдяки діджиталізації, студенти можуть здійснювати пошук необхідних даних, проводити аналіз наукових робіт, спілкуватися з колегами та викладачами за допомогою електронних платформ, а також отримувати доступ до актуальної інформації в будь-який час із будь-якого пристрою.

Прикладом тому є університетська бібліотека СУМДУ (рис. 1.3), яка є важливим компонентом інформаційної інфраструктури університету.

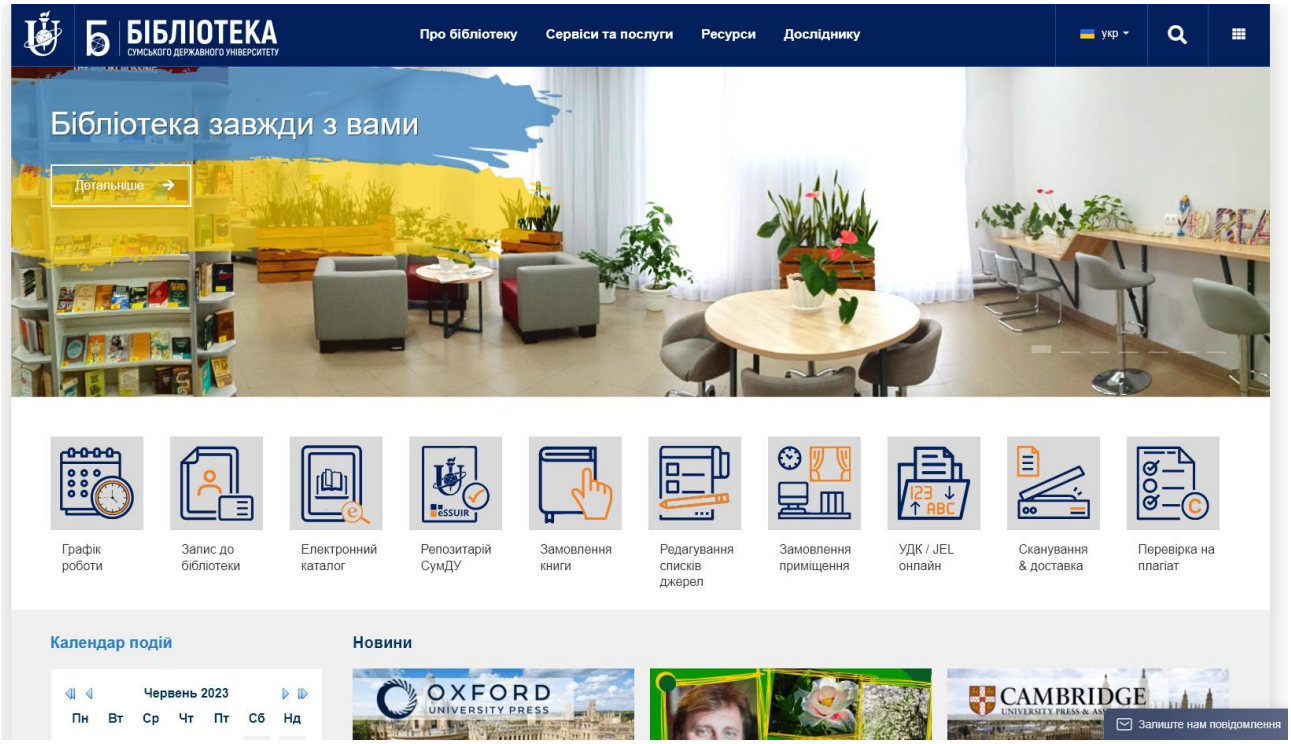

Рисунок 1.3 – Бібліотека СУМДУ

Вона надає студентам широкий доступ до багатогранних ресурсів, які підтримують академічні дослідження, навчання та самостійну роботу. Користувачі можуть скористатися електронним каталогом, в якому представлені різні види літератури, інтерактивними платформами для пошуку, перегляду та завантаження електронних ресурсів.

У контексті цього порівняння, школи у Сумах відстають у своїй діджиталізації. Широкий доступ до літературних ресурсів та бібліотечних послуг, які є важливими для навчання та саморозвитку учнів, є обмеженим або навіть відсутнім у багатьох школах. Незважаючи на те, що деякі школи можуть мати просторові бібліотеки, вони не забезпечують широкий доступ до цифрових ресурсів та інформаційних технологій, які б полегшили процес пошуку, отримання та взаємодії з інформацією для учнів та педагогічного персоналу.

Тому, на основі проведеного аналізу, виникає необхідність у розвитку інформаційної системи для шкільних бібліотек з метою забезпечення учням доступу до різноманітних літературних ресурсів та послуг. Така система буде сприяти поліпшенню якості освіти та розширенню можливостей навчання та саморозвитку учнів, а також сприятиме їхній підготовці до вимог сучасного світу, де діджиталізація та доступ до інформації відіграють все більш важливу роль.

У зв'язку з відсутністю предмету аналізу у вигляді шкільних бібліотек, було проведено аналіз простих онлайн бібліотек як альтернативного джерела доступу до літературних ресурсів. Під час цього аналізу було вивчено деякі онлайн платформи, які надають електронний доступ до книжок, журналів та інших джерел інформації.

При пошуку в інтернеті запиту «онлайн бібліотека» першим сайтом видається офіційний сайт-бібліотеку Свідків Ієгови (рис. 1.4). Це говорить про гостру відсутність схожих ресурсів.

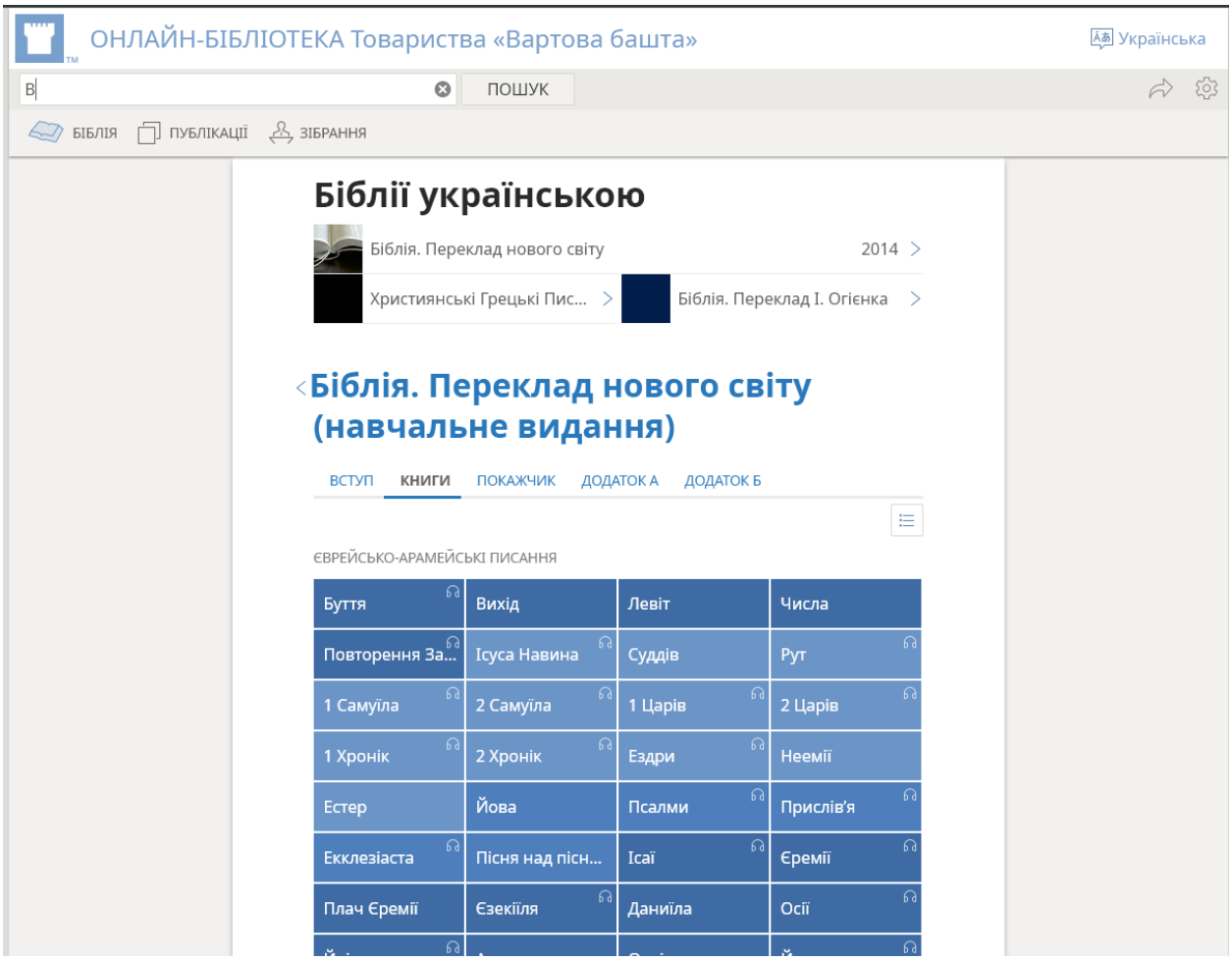

Рисунок 1.3 – Бібліотека СУМДУ

При проведенні аналізу веб-сайту було виявлено, що його дизайн викликає позитивні враження: чистий і естетичний, зрозумілий і зручний у використанні. Однак, при ближчому огляді було помічено обмежений вибір літературних ресурсів на цьому сайті. Незважаючи на те, що доступні аудіокниги, відсутність можливості завантаження матеріалів або читання їх онлайн виявилася значним недоліком.

Його обмежені можливості в частині доступу до літературних матеріалів обмежують користувачів. Відсутність можливості скачування або читання книг онлайн обмежує їх зручність та функціональність.

Це свідчить про те, що, хоча сайт може виконувати певні розважальні функції, він не задовольняє потреби у повноцінному доступі до літературних ресурсів.

Під час мого дослідження різних інтернет-ресурсів, я наткнувся на хороший приклад онлайн-бібліотеки (рис. 1.4).

Рисунок 1.4 – Хороший приклад сайту-бібліотеки

На рисунку 1.4 представлений зразок дизайну та присутній значний асортимент книг для прочитання. Однак, під час аналізу функціоналу сайту, представленого на рисунках 1.5 та 1.6, було виявлено його обмеженості. Наприклад, відсутня можливість скачувати книги, що може обмежити користувачів у зручному доступі до матеріалів. Крім того, важливо зазначити, що якість тексту книг іноді може бути низькою через те, що книги скановані, а не відформатовані електронно. Це може вплинути на зручність читання та загальне задоволення користувачів.

Рисунок 1.5 – Сторінка перегляду книги

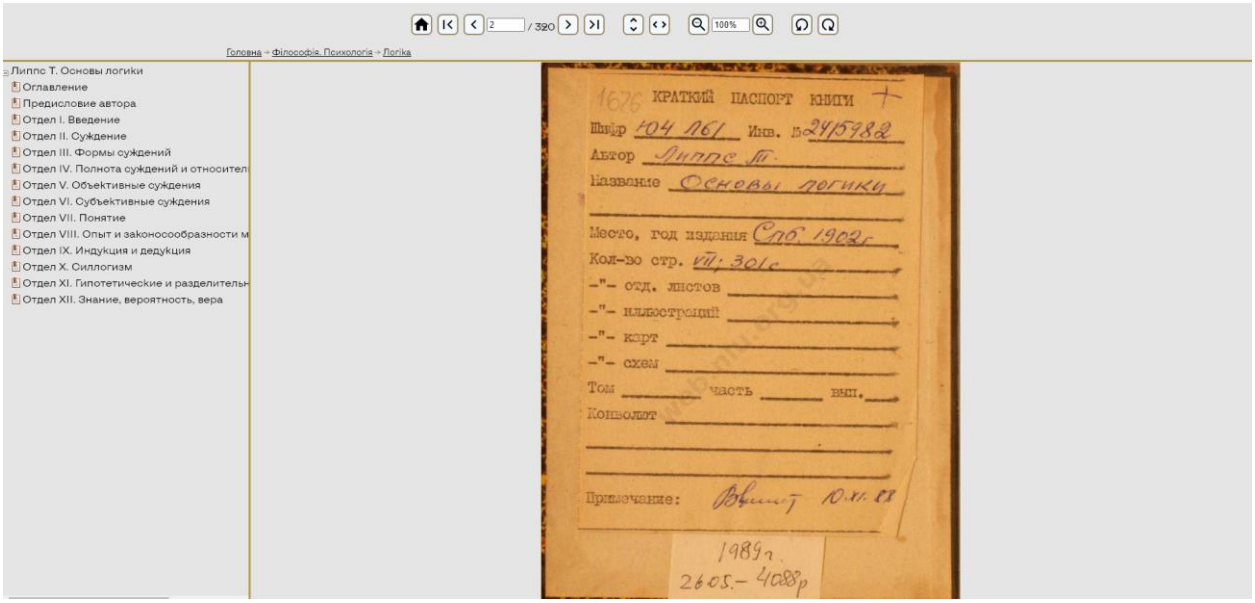

Рисунок 1.6 – Режим читання

1.3 Постановка задачі

Метою роботи є розробка веб-додатку та інформаційної системи для шкільної бібліотеки. Для досягнення цієї мети необхідно вирішити наступні задачі:

1) Провести провести аналіз різних інструментів розробки, що доступні, з метою вибору найбільш оптимальних. Огляд буде включати в себе оцінку функціональних можливостей, зручність використання, сумісність з вимогами проекту, наявність підтримки та розширюваності. Після цього здійснити вибір тих інструментів, які найкраще задовольнять потреби розробки бібліотечного веб-додатку, забезпечуючи його оптимальну роботу та ефективне функціонування.
2) Забезпечити розробку архітектури веб-додатку, що забезпечить ефективне функціонування бібліотечної системи, включаючи модулі для керування запитами, автоматичного оновлення інформації про книги та користувачів, контролю запозичень та повернень книг тощо.
3) Розробити базу даних для зберігання інформації, необхідної для ефективного управління бібліотекою, таку як каталог книг, інформація про користувачів, видача та повернення книг тощо.
4) Розробити інтуїтивно зрозумілий та зручний інтерфейс веб-додатку для користувачів, що дозволить їм легко шукати книги, бронювати, замовляти та отримувати інформацію про доступні ресурси бібліотеки.
5) Здійснити тестування веб-додатку з метою перевірки його функціональності, надійності та відповідності вимогам бібліотечної системи.

1.4 Технічне завдання

Читач реєструється в системі і далі має можливість:

- здійснювати пошук (за автором / назвою/классом).
- скачувати книгу з каталогу.
- оформляти замовлення на книгу з каталогу.
- переглядати власний профіль зі списком замовлених книг.
- змінювати інформацію про себе (класс / аватар).

Книга може бути присутньою в бібліотеці в одному або декількох екземплярах. Система веде облік доступного кількості книг.

Адміністратор системи володіє правами:

- додавання книги.
- видалення книги.
- редагування інформації про книгу.
- перегляд списку користувачів.
- видалення користувачів.

2 ВИБІР МЕТОДІВ РОЗВ’ЯЗАННЯ ЗАДАЧІ

2.1 Вибір стеку технологій

При розробці бекенду мого проекту, я прийняв рішення використовувати Java та Spring.

Java - це потужна інтерпретована, об'єктно-орієнтована мова програмування, розроблена у 1995 році компанією Sun Microsystems (зараз власністю Oracle Corporation). Вона просунулася до однієї з найпопулярніших мов програмування завдяки своїй надійності, переносимості та широкому спектру застосувань.

- Основні риси Java: Об'єктно-орієнтована: Java побудована на принципах об'єктно-орієнтованого програмування (ООП). Це означає, що програми побудовані з використанням об'єктів, які взаємодіють один з одним для виконання завдань. Це сприяє модульності, повторному використанню коду та полегшує розробку та підтримку додатків.
- Платформа Java: Java працює на власній віртуальній машині Java (Java Virtual Machine, JVM), яка дозволяє виконувати код Java на будь-якій підтримуваній платформі без необхідності перекомпіляції. Це робить Java крос-платформенною мовою, що дозволяє запускати додатки на різних операційних системах, таких як Windows, macOS та Linux.
- Надійність та безпека: Java була розроблена з фокусом на безпеку та надійність. Вона має вбудовану систему управління пам'яттю, що допомагає уникнути багатьох типових помилок, пов'язаних з управлінням пам'яттю. Крім того, Java має вбудовану систему обробки виключень, яка дозволяє контролювати та обробляти помилки в програмі.

Велика екосистема: Java має широку екосистему, яка включає в себе різноманітні фреймворки, бібліотеки та інструменти для розробки. Фреймворки, такі як Java Spring, Hibernate, Apache Struts та інші, спрощують розробку різноманітних додатків, включаючи веб-додатки, мобільні додатки та вбудовані системи.

Spring Framework є одним з найпопулярніших фреймворків для розробки додатків на мові програмування Java. Він надає різноманітні інструменти та бібліотеки для спрощення розробки бекенд-додатків, підвищення продуктивності та забезпечення доброї архітектури.

- Основні компоненти та особливості Spring Framework: Інверсія керування та контейнер виконання: Одна з ключових концепцій Spring Framework - це інверсія керування (Inversion of Control, IoC). Фреймворк бере на себе відповідальність за створення та управління об'єктами, а не розробник. Це забезпечує більшу модульність, легкість тестування та зменшення залежностей між компонентами. Контейнер виконання Spring здійснює управління життєвим циклом об'єктів, що забезпечує їх створення, налаштування та знищення.
- Внедрення залежностей (Dependency Injection, DI): Spring Framework підтримує внедрення залежностей, що дозволяє зв'язувати компоненти додатку між собою без жорстких залежностей. Це полегшує тестування, заміну компонентів та забезпечує більшу гнучкість.
- Модульність: Spring Framework побудований на принципах модульності. Він складається з різних модулів, які можна використовувати окремо або в поєднанні залежно від потреб проекту. Наприклад, модуль Spring MVC використовується для розробки веб-додатків, модуль Spring Data - для роботи з базами

даних, модуль Spring Security - для забезпечення безпеки та автентифікації.

- Підтримка транзакцій: Spring Framework надає механізми для керування транзакціями в базах даних. Використовуючи анотації або конфігурацію, можна легко визначити, які операції мають бути виконані в межах однієї транзакції, що забезпечує цілісність даних та уникнення помилок.
- Підтримка веб-розробки: Spring Framework має модуль Spring MVC, який є потужним фреймворком для розробки веб-додатків. Він надає механізми для обробки HTTP-запитів, маршрутизації, створення моделей та видів, валідації даних, кешування та інші функції, необхідні для розробки веб-додатків.
- Підтримка тестування: Spring Framework має вбудовану підтримку для написання тестів. Ви можете легко створювати юніт-тести та інтеграційні тести для перевірки функціональності вашого додатка. Spring Framework також надає можливості для імітації залежностей та контролю стану додатка під час тестування.

Підтримка безпеки: Spring Framework має модуль Spring Security, який надає функціональність для забезпечення безпеки додатків. Він дозволяє налаштовувати аутентифікацію, авторизацію, керування ролями та інші аспекти безпеки в вашому додатку.

Єдиний шаблон створення веб-додатків: Spring пропонує консистентний підхід до створення веб-додатків, використовуючи шаблон проектування MVC (Model-View-Controller). Це дозволяє розподілити функціональність додатку на логічно відокремлені компоненти, що полегшує розробку, тестування та підтримку, що значно полегшує процесс розробки в порівнянні з іншими фреймворками. Загалом, вибір Java та Spring для розробки бекенду сайту дозволить використовувати потужні інструменти та забезпечити швидкість, безпеку та надійність вашого додатку.

PostgreSQL є одною з популярних відкритих реляційних баз даних і має декілька переваг порівняно з іншими системами управління базами даних. Ось деякі з них:

- Надійність та цілісність даних: PostgreSQL відомий своєю високою надійністю та цілісністю даних. Він підтримує транзакції з високою ступенем безпеки і механізми відновлення після відмови, що забезпечують стабільну роботу бази даних навіть у разі аварій або перебоїв в роботі.
- Розширюваність та масштабованість: PostgreSQL надає різні можливості для масштабування бази даних. Він підтримує партиціонування, реплікацію та кластеризацію, що дозволяє розподілити навантаження та забезпечити високу доступність.
- Розширені можливості: PostgreSQL має багатий набір функцій та розширень. Він підтримує високий рівень стандартів SQL, включаючи складні запити, підзапити, з'єднання та інші операції. Крім того, PostgreSQL дозволяє користувачам створювати свої власні функції, типи даних та розширення.
- Підтримка географічних та геолокаційних даних: PostgreSQL має вбудовану підтримку для роботи з географічними та геолокаційними даними. Це робить його відмінним вибором для додатків, які працюють з картографією, геолокацією або великими обсягами географічних даних.
- Відкритість та активна спільнота: PostgreSQL є відкритим проектом з активною спільнотою розробників. Це означає, що ви можете розраховувати на підтримку, оновлення та вдосконалення бази даних, а також на доступ до багатої документації та ресурсів, що допоможуть вам у роботі з PostgreSQL.

Враховуючи ці переваги, PostgreSQL є часто обраною базою даних для розробки бекенд-додатків, особливо для проектів, де надійність, цілісність даних та розширюваність є важливими факторами.

Thymeleaf добре інтегрується з Java та фреймворками, такими як Spring. Ви можете використовувати Thymeleaf як шаблонний двигун для генерації HTML на сервері, що дозволяє вам використовувати Java-код та передавати дані з бекенду до фронтенду.

Thymeleaf має простий та зрозумілий синтаксис, що робить його легким для навчання і використання, особливо для розробників, які вже мають досвід роботи з HTML.

Bootstrap 5 - готовий набір стилів і компонентів: Bootstrap 5 надає велику кількість готових стилів, компонентів та шаблонів, що спрощує розробку інтерфейсу. Ви можете легко використовувати класи Bootstrap для створення респонсивного та привабливого дизайну свого веб-додатка. Bootstrap 5 добре підтримує респонсивний дизайн, що означає, що ваш веб-додаток буде добре виглядати та працювати на різних пристроях та розмірах екранів, включаючи мобільні пристрої. Але використання Bootstrap 5 може призвести до того, що ваш веб-додаток може виглядати схожим на багато інших веб-сайтів, які також використовують Bootstrap. Це може зробити його менш унікальним або вирізнятися з-поміж конкурентів.

2.2 Формування технічного завдання на основі стеку технологій

Вимоги до реалізації UI - частини:

1. Розробити користувацький інтерфейс веб-додатка, який буде забезпечувати зручну навігацію та взаємодію користувача з онлайн бібліотекою. Використовувати фреймворк Thymeleaf для побудови динамічних сторінок з використанням шаблонів та розширених можливостей управління даними.
2. Забезпечити зручну систему пошуку та фільтрації книг з використанням відповідних інтерактивних елементів.
3. Реалізувати можливість перегляду деталей про книги, включаючи обкладинку, опис, авторів та інші важливі атрибути.
4. Забезпечити можливість реєстрації та аутентифікації користувачів, що дозволить зберігати персональні налаштування та історію взаємодії з бібліотекою.
5. Забезпечити можливість додавання та видалення книг з особистого списку замовлених.
6. Розробити відповідні форми та механізми валідації даних при реєстрації, вході та редагуванні профілю користувача
7. Підтримувати роботу з кирилицею та багатомовність, дозволяючи користувачам використовувати сайт на різних мовах, включаючи українську та англійську.
8. Забезпечити коректну обробку помилок та виключних ситуацій, що можуть виникнути під час взаємодії з веб-додатком.
9. Застосовувати практики оформлення коду та стандарти оформлення в Thymeleaf, забезпечуючи читабельність та підтримку коду в майбутньому.
10. Забезпечити респонсивний дизайн, що дозволяє веб-додатку належним чином відображатися на різних пристроях та екранах різного розміру.
11. Впровадити можливість збереження та відновлення стану сесій користувачів для збереження контексту та покращення зручності використання.

Вимоги до реалізації серверної частини:

1. На основі сутностей предметної області створити класи, які їм відповідають.
2. Класи і методи повинні мати назви, що відображають їх функціональність, і повинні бути рознесені по пакетам.
3. Оформлення коду має відповідати Java Code Convention.
4. Інформацію щодо предметної області зберігати у реляційній базі даних (в якості СУБД рекомендується використовувати MySQL або PostgreSQL).
5. Для доступу до даних використовувати ORM фреймворк Spring Data JPA.
6. Застосунок має підтримувати роботу з кирилицею (бути багатомовним), в тому числі при зберіганні інформації в базі даних: a. повинна бути підтримка введення, виведення і зберігання інформації (в базі даних), записаної на різних мовах; b. в якості мов обрати мінімум дві: одна на основі кирилиці (українська), інша на основі латиниці (англійська).
7. Архітектура застосунка повинна відповідати шаблону MVC. Допускається використання ORM фреймворків(рекомендується використовувати Spring MVC).
8. При реалізації бізнес-логіки необхідно використовувати шаблони проектування: Команда, Стратегія, Фабрика, Будівельник, Сінглтон, Фронт-контролер, Спостерігач, Адаптер та ін. Використання шаблонів повинно бути обґрунтованим.
9. Реалізувати захист від повторної відправки даних на сервер при оновленні сторінки (реалізувати PRG).
10. При розробці використовувати сесії, фільтри, слухачі.
11. У застосунку повинні бути реалізовані аутентифікація і авторизація, розмежування прав доступу користувачів системи до компонентів програми. Шифрування паролів заохочується.
12. Впровадити у проект журнал подій із використанням бібліотеки log4j.
13. Всі поля введення повинні бути із валідацією даних.
14. Застосунок має коректно реагувати на помилки та виключні ситуації різного роду (кінцевий користувач не повинен бачити stack trace на стороні клієнта).

3 ІНФОРМАЦІЙНЕ ТА ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ

3.1 Створення проекту

Для успішної розробки проекту Spring Boot необхідно створити проект (рис. 3.1) та обрати всі необхідні компоненти (рис. 3.2). Для досягнення надійності та стабільності роботи системи, рекомендується встановити наступні програмні бібліотеки:

1. Lombok: Ця бібліотека надає додаткові аннотації та функції, що спрощують розробку, такі як автоматичне створення геттерів і сеттерів, конструкторів тощо.
2. Spring Web: Ця бібліотека надає функціональність для розробки веб-додатків, включаючи веб-контролери, обробку HTTP-запитів та відповідей.
3. Thymeleaf: Ця бібліотека забезпечує можливості для шаблонізації веб-сторінок, що дозволяє зручно відображати дані на стороні сервера.
4. Spring Security: Ця бібліотека надає засоби для забезпечення безпеки в додатку, включаючи автентифікацію, авторизацію та керування правами доступу.
5. Spring Data JPA: Ця бібліотека спрощує взаємодію з базами даних через Java Persistence API (JPA) і надає можливість легко виконувати операції з даними.
6. PostgreSQL Driver: Ця бібліотека є драйвером (розширенням) для Spring Boot, яке дозволяє взаємодіяти з базою даних PostgreSQL. Вона надає можливість з'єднуватися з базою даних PostgreSQL та виконувати різноманітні запити та операції, такі як створення, читання, оновлення та видалення даних.

7. Spring Validation: Ця бібліотека дозволяє проводити валідацію вхідних даних, перевіряти їх на відповідність певним правилам та обмеженням.

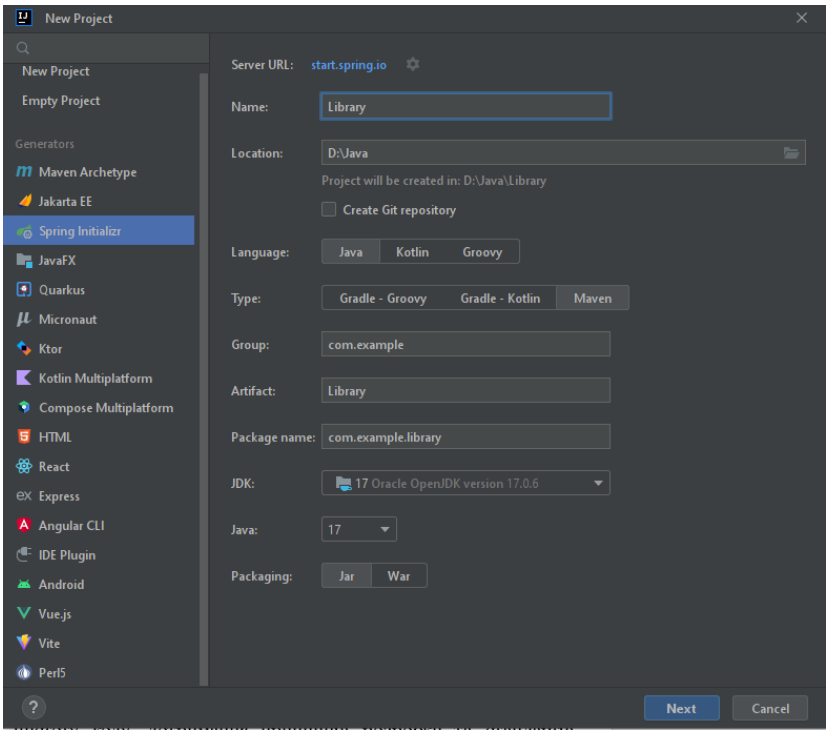

Рисунок 3.1 – Створення SpringBoot проекту

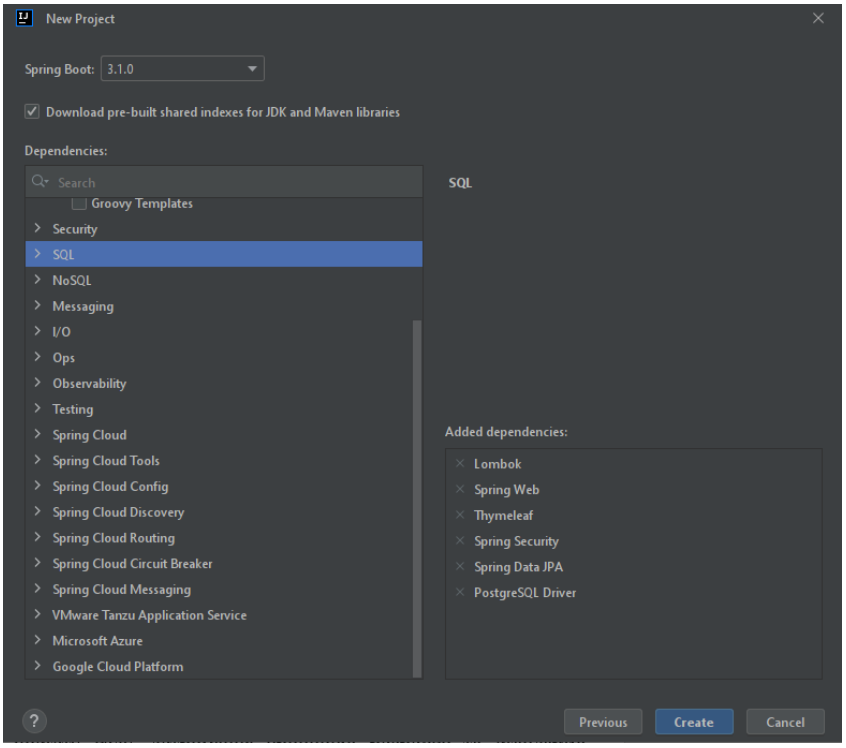

Рисунок 3.2 – Архітектура пакетів

3.2 Архітектура проекту

У розробці цього проекту буде використана архітектура, яка базується на шаблоні Model-View-Controller (MVC). Цей шаблон розділяє програму на три основні компоненти: модель (Model), представлення (View) та контролер (Controller), кожен з яких відповідає за свої функціональні обов'язки.

Модель представляє дані та бізнес-логіку додатку. Вона відповідає за обробку та зберігання інформації, а також взаємодію з базою даних.

Представлення відповідає за відображення інформації користувачеві. Це може бути веб-сторінка, інтерфейс користувача або будь-який інший спосіб візуалізації даних.

Контролер відповідає за обробку запитів користувача та взаємодію з моделлю і представленням. Він приймає вхідні дані від користувача, виконує необхідні дії та оновлює модель та представлення відповідно.

Застосування архітектури MVC дозволяє розділити логіку програми на окремі компоненти, що полегшує розробку, тестування та розширення проекту. Вона також покращує повторне використання коду та забезпечує більшу гнучкість і підтримку змін.

Завдяки архітектурі MVC в цьому проекті забезпечується чітке розмежування відповідальностей та забезпечується добра організація коду, що сприяє покращенню ефективності та підтримці проекту.

Структура папок проекту (рис. 3.3) виглядає наступним чином:

1. Папка "src": Ця папка є основною папкою проекту, яка містить весь вихідний код. Папка "src/main": В цій папці знаходяться основні файли проекту.
2. Папка "src/main/java": В цій папці містяться файли з Java-кодом проекту.
3. Папка "src/main/resources": Ця папка відповідає компоненту "View" або представленню. Вона містить ресурсні файли, такі як HTML-шаблони, CSS-стилі, JavaScript-файли та інші ресурси, які

використовуються для відображення даних користувачеві. Також ця папка виступає файловим сховищем нашого веб-додатку.

4. Папка "src/main/java/com/example/library/controllers": Ця папка відповідає компоненту "Controller" або контролеру. Вона містить класи, які обробляють запити користувача і взаємодіють з моделлю та представленням.
5. Папка "src/main/java/com/example/library/models": Ця папка відповідає компоненту "Model" або моделі. Вона містить класи, які представляють дані і бізнес-логіку проекту, а також забезпечують взаємодію з базою даних.
6. Інші папки та файли в папці "src/main/java": Вони також відповідають компоненту "Model" і містять класи, які допомагають управляти даними, конфігурацією, сервісами та іншими аспектами проекту.

Ця структура папок допомагає зберігати код організованим, легким для розуміння та підтримки. Вона також сприяє використанню принципів розробки, таких як розділення відповідальностей і модульність, що полегшує розширення та збереження стабільності проекту..

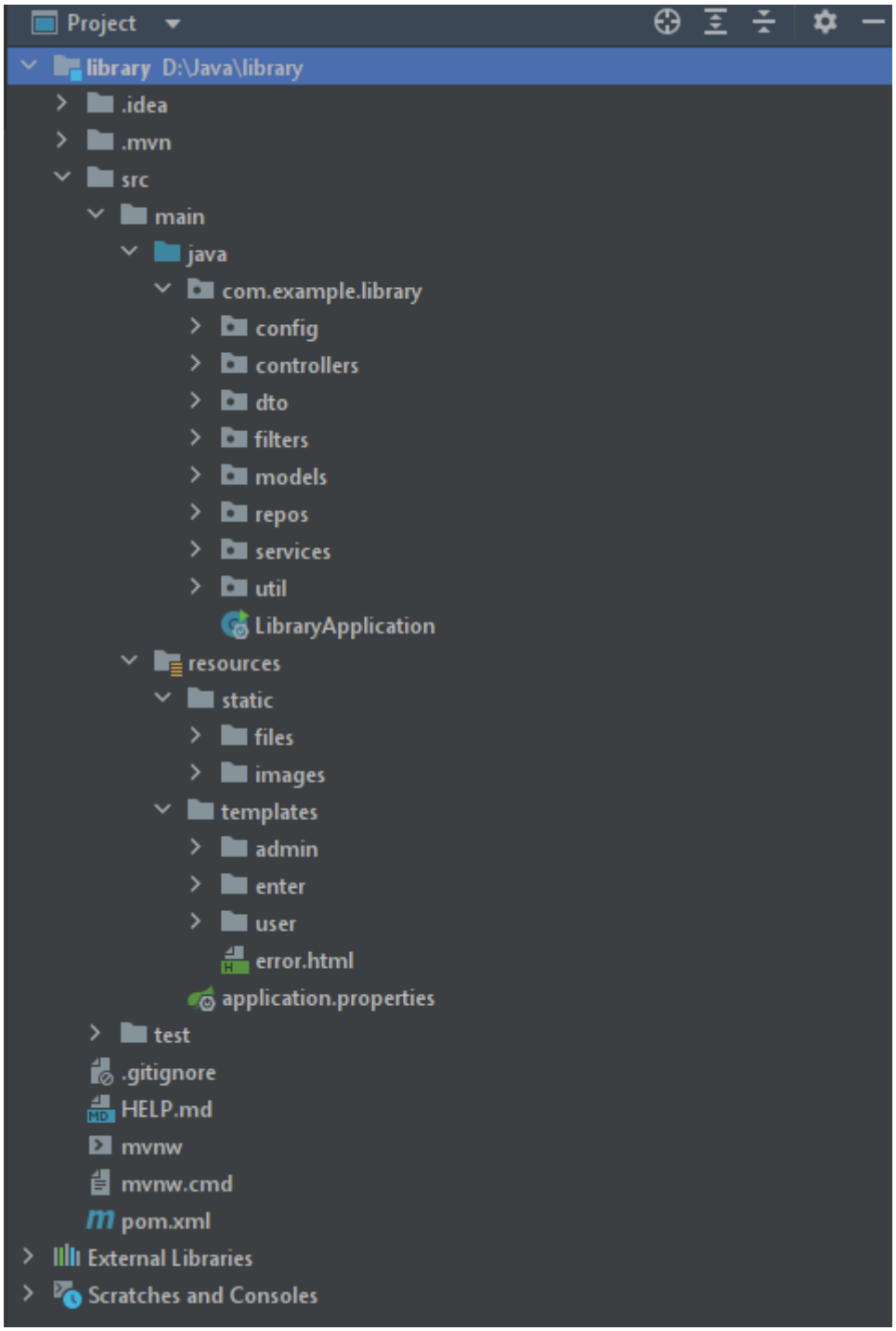

Рисунок 3.3 – Структура папок проекту

3.3 Взаємодія з базою данних

Для підключення до бази даних у нашому проекті використовується файл application.properties (рис. 3. 4), який знаходиться в папці "src/main/resources". У цьому файлі ми вказуємо налаштування для з'єднання з базою даних.

Також в цьому файлі вказані деякі інші налаштування для проекту.

```
spring.datasource.url=jdbc:postgresql://localhost:5432/library
spring.datasource.username=postgres
spring.datasource.password=
spring.jpa.hibernate.ddl-auto=update
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.PostgreSQLDialect
spring.jpa.properties.hibernate.show_sql=false

spring.servlet.multipart.max-file-size=20MB
spring.servlet.multipart.max-request-size=20MB

upload.path=src/main/resources/static/files/
image.path=src/main/resources/static/images/

server.error.include-stacktrace=always
```

Рисунок 3.4 – Файл налаштувань проекту

Для роботи з базою даних ми використовуємо моделі Spring Data JPA (рис. 3.5) . Моделі Spring Data JPA - це класи, які відображають сутності бази даних і виконують роль об'єктно-реляційного відображення (ORM). Вони використовуються для взаємодії з базою даних та виконання операцій зчитування, запису, оновлення та видалення даних.

Spring Data JPA надає спрощений спосіб визначення моделей за допомогою анотацій, які вказують на зв'язки між класами та таблицями бази даних. Це дозволяє автоматично створювати SQL-запити на основі методів, визначених у моделях, і виконувати їх за допомогою JPA-постачальника (наприклад, Hibernate). Моделі Spring Data JPA також можуть містити методи, які визначають власні запити до бази даних з використанням JPQL (Java Persistence Query

Language) або критеріїв JPA. Це дозволяє виконувати більш складні запити та налаштовувати вибірку даних відповідно до вимог додатку.

Окрім того, моделі Spring Data JPA підтримують різні анотації для визначення структури таблиць, відношень між сутностями, валідації даних та інших аспектів ORM. Вони дозволяють розширювати функціональність моделей та забезпечувати їх відповідність вимогам проекту та схеми бази даних.

Використання моделей Spring Data JPA спрощує розробку бекенду, оскільки вони автоматично забезпечують основні операції з базою даних і зменшують необхідність у написанні власного SQL-коду.

```java
@Entity
@Data
@NoArgsConstructor
public class Book implements BasicModel{
    no usages
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id", nullable = false)
    private Integer id;

    1 usage
    @Size(max = 255)
    @NotNull
    @Column(name = "name", nullable = false)
    private String name;

    1 usage
    @Size(max = 255)
    @NotNull
    @Column(name = "author", nullable = false)
    private String author;

    1 usage
    @Size(max = 1020)
    @NotNull
    @Column(name = "description")
    private String description;

    2 usages
    @Enumerated(EnumType.STRING)
    private EGrade grade;

    1 usage
    @Column(name = "amount")
    private Integer amount;

    3 usages
    @OneToMany(mappedBy = "book", cascade = CascadeType.ALL, orphanRemoval = true, fetch = FetchType.EAGER)
    private List<FileEntity> files = new ArrayList<>();
```

Рисунок 3.5 – Приклад моделі

3.4 Взаємодія з view

Взаємодія з view в Spring Boot відбувається через контролери та шаблонний двигун, такий як Thymeleaf.

Контролери (рис. 3.6) відповідають за обробку HTTP-запитів та встановлення відповідних моделей для передачі даних у view. Вони визначають методи, які будуть виконуватися при отриманні певного типу запиту на певний URL-шлях. Контролери можуть приймати параметри з запиту, обробляти їх і генерувати відповідь для клієнта.

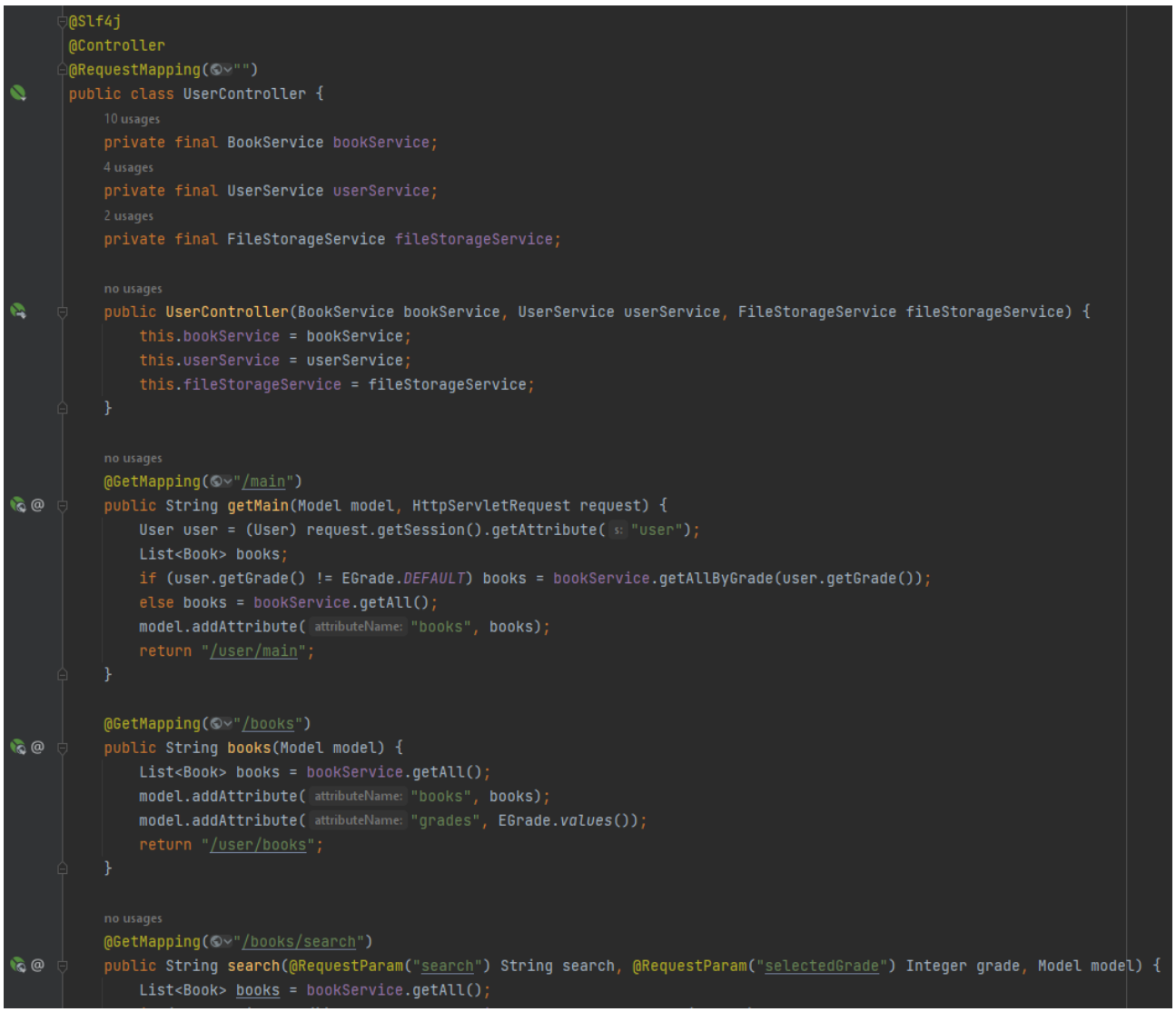

Рисунок 3.6 – Приклад контроллера

У проекті було реалізовано 4 контролери, які відповідають за різні аспекти функціональності системи:

1. Контролер авторизації (EnterController.java): Цей контролер відповідає за обробку запитів, пов'язаних з авторизацією користувачів. Він може мати методи для відображення сторінок авторизації, обробки введених даних, перевірки правильності логіна та пароля та встановлення сесії або токенів авторизації.
2. Контролер взаємодії з файлами (FileController.java): Цей контролер відповідає за обробку запитів, пов'язаних з завантаженням, збереженням та взаємодією з файлами. Він може мати методи для відображення сторінок завантаження файлів, обробки вибраних файлів, збереження їх у системі, відображення списку файлів, видалення файлів тощо.
3. Контролер адміністратора (AdminController.java): Цей контролер відповідає за обробку запитів, пов'язаних з адмініструванням системи. Він може мати методи для відображення сторінок управління користувачами, додавання та видалення користувачів, зміни їхніх ролей, перегляду статистики тощо.
4. Контролер користувача (UserController.java): Цей контролер відповідає за обробку запитів, пов'язаних з функціональністю, доступною звичайному користувачеві. Він може мати методи для відображення основних функцій системи, які доступні користувачу, таких як перегляд інформації, зміна налаштувань, взаємодія зі списками, коментування тощо.

Thymeleaf є шаблонним двигуном, що дозволяє вбудовувати код і динамічно генерувати HTML-сторінки на основі даних, переданих з контролера. Він використовує синтаксис, схожий на HTML, з додатковими атрибутами та можливістю виконувати вирази. За допомогою Thymeleaf можна виконувати

розмітку, управляти відображенням даних, використовувати умови, цикли та багато іншого.

Коли контролер обробляє запит, він може встановлювати потрібні дані у модель, яка потім передається в шаблон Thymeleaf. У шаблоні Thymeleaf можна використовувати спеціальні атрибути та вирази для відображення даних з моделі. Наприклад, можна використовувати th:text для встановлення текстового вмісту елементу або th:each для ітерації по списку та відображення кожного елементу.

Після обробки шаблону Thymeleaf згенерує HTML-сторінку (рис. 3.7) з врахуванням вбудованих виразів та атрибутів. Цю HTML-сторінку контролер повертає як відповідь на запит. Нарешті, клієнт отримує цю сторінку та може переглядати її у своєму веб-браузері.

```
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
  <title>Online library</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-1BmE4kW
</head>
<body>
<nav class="navbar navbar-expand-lg navbar-light bg-light">
  <div class="container px-4 px-lg-5">
    <div>
      <img alt="Image" th:src="@{/image/navbar_logo.jpg}"/>
      <a class="navbar-brand" href="/main">Library</a>
    </div>
    <div class="collapse navbar-collapse" id="navbarSupportedContent">
      <ul class="navbar-nav me-auto mb-2 mb-lg-0 ms-lg-4">
        <li class="nav-item"><a class="nav-link" aria-current="page" href="/profile">Profile</a></li>
        <li class="nav-item"><a class="nav-link" href="/books">Books</a></li>
        <li class="nav-item"><a class="nav-link" href="#!">About</a></li>
      </ul>
      <form th:action="@{/logout}" method="get">
        <input class="btn btn-outline-dark mt-auto" type="submit" value="Logout">
      </form>
    </div>
  </div>
</nav>
<header class="bg-dark py-3">
  <div class="container px-4 px-lg-5 my-5">
    <div class="text-center text-white">
      <h1 class="display-4 fw-bolder">Online library</h1>
      <p class="lead fw-normal text-white-50 mb-0">Sumy school for the mentally retarded library web site</p>
    </div>
  </div>
</header>
<section class="py-5">
  <div class="container px-7 px-lg-8 mt-8">
    <div class="row gx-4 gx-lg-5 row-cols-2 row-cols-md-3 row-cols-xl-4 justify-content-center">
      <table th:each="book : ${books}">
        <tr>
          <th style="...">
            <div class="col mb-5">
              <div class="card h-100">
```

Рисунок 3.7 – Приклад HTML-сторінки

3.5 Розробка інтерфейсу проекту

Кожен сучасний веб-сайт повинен мати механізм авторизації, який забезпечує захист конфіденційної інформації користувачів та контроль доступу до функціональності системи. Для цього, в проекті використовуються інтерфейси авторизації та реєстрації, які надають користувачам можливість ідентифікуватися та отримувати доступ до персоналізованого змісту і функцій. Інтерфейс авторизації включає наступні елементи: початкова сторінка (рис. 3.8), сторінка авторизації та сторінка реєстрації (рис. 3.9).

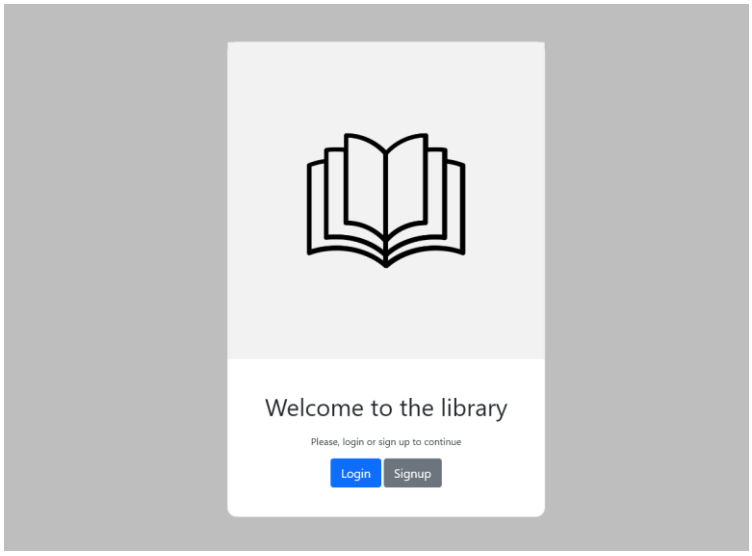

Рисунок 3.8 – Початкова сторінка

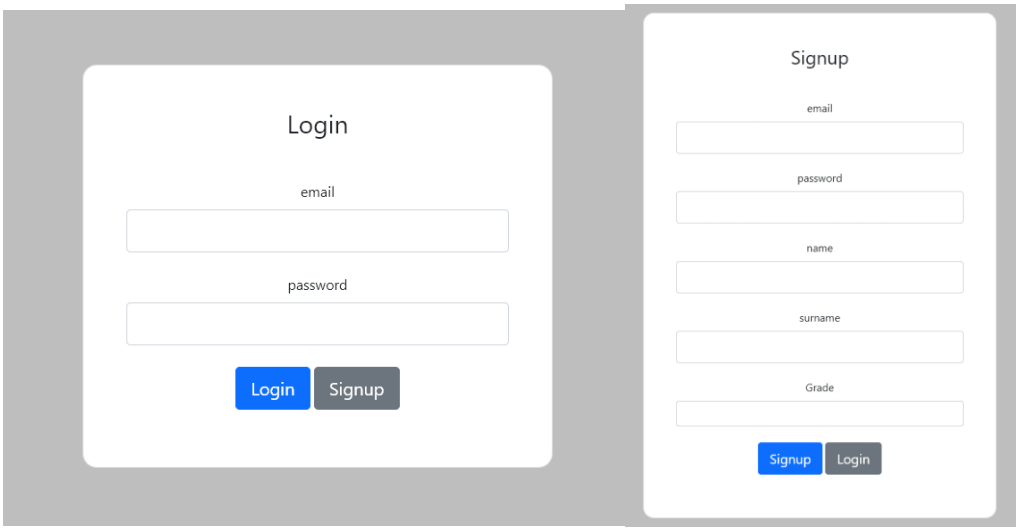

Рисунок 3.9 – Сторінки авторизації та реєстрації

Також була розроблена панель адміністратора (рис. 3.10), яка надає додаткові можливості для управління системою. Адміністративна панель дозволяє авторизованим адміністраторам виконувати такі дії:

1. Керування книжками: Адміністратор може додавати нові книжки до системи, редагувати існуючі книжки, а також видаляти книжки, які більше не актуальні або потребують виправлень. Це дозволяє підтримувати актуальну та зручну базу даних книжок.
2. Управління користувачами: Адміністратор може переглядати список зареєстрованих користувачів, переглядати їхні дані та здійснювати дії з управління користувачами, такі як блокування або видалення облікового запису.
3. Заходження на сайт від імені юзера: Адміністратор має можливість увійти на сайт від імені будь-якого користувача. Це дозволяє перевірити та відлагодити функціонал, доступний для різних типів користувачів та переконатися, що система працює належним чином для всіх ролей.

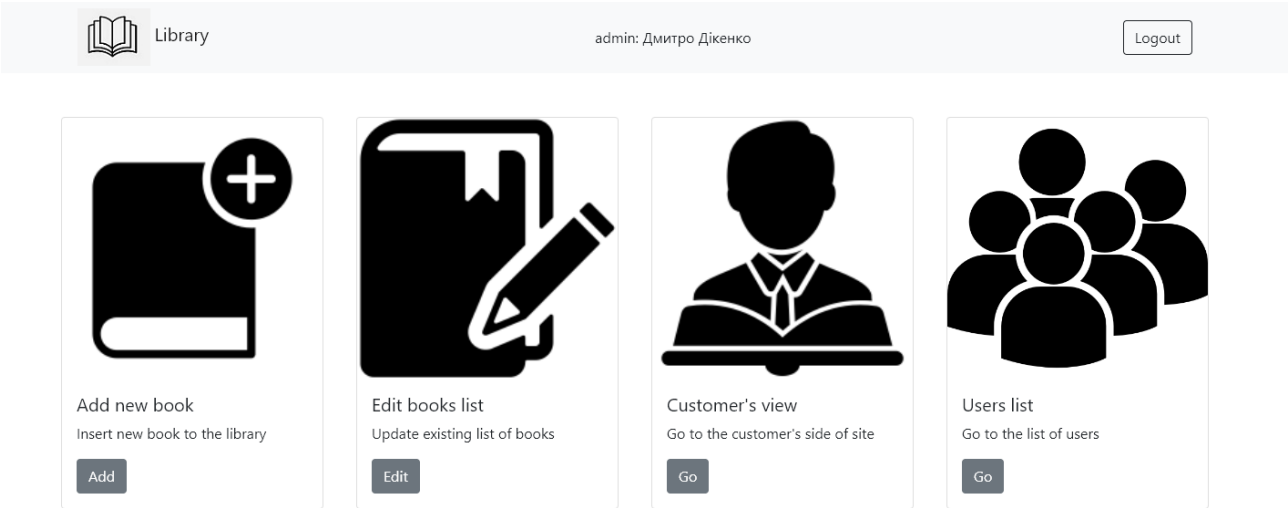

Рисунок 3.10 – Панель адміністратора

В рамках проекту була розроблена адаптивна головна сторінка (рис. 3.11), яка надає користувачам зручний і персоналізований досвід. На цій сторінці відображаються рекомендовані книги, враховуючи клас, в якому навчається користувач.

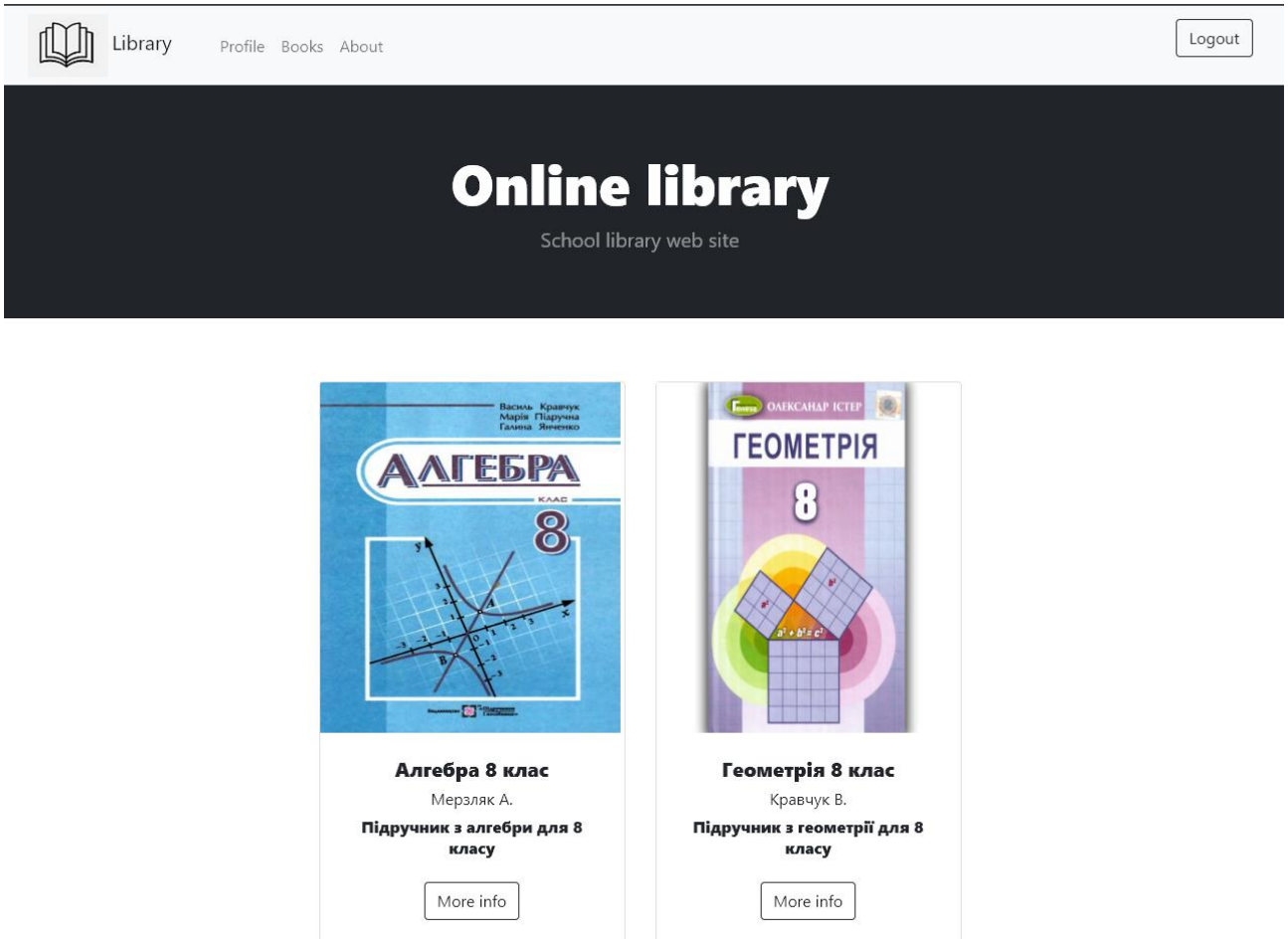

Рисунок 3.11 – Головна сторінка

Додатково до адаптивної головної сторінки, в проекті була реалізована сторінка, на якій відображаються всі наявні книги в бібліотеці (рис. 3.12). Ця сторінка надає користувачам можливість виконувати пошук за класом, назвою книги або ім'ям автора, для зручного та швидкого знаходження необхідних книжок. На цій сторінці користувачам пропонується перелік усіх доступних книг у бібліотеці. Кожна книга відображається зі своїм заголовком, автором, обкладинкою та додатковою інформацією.

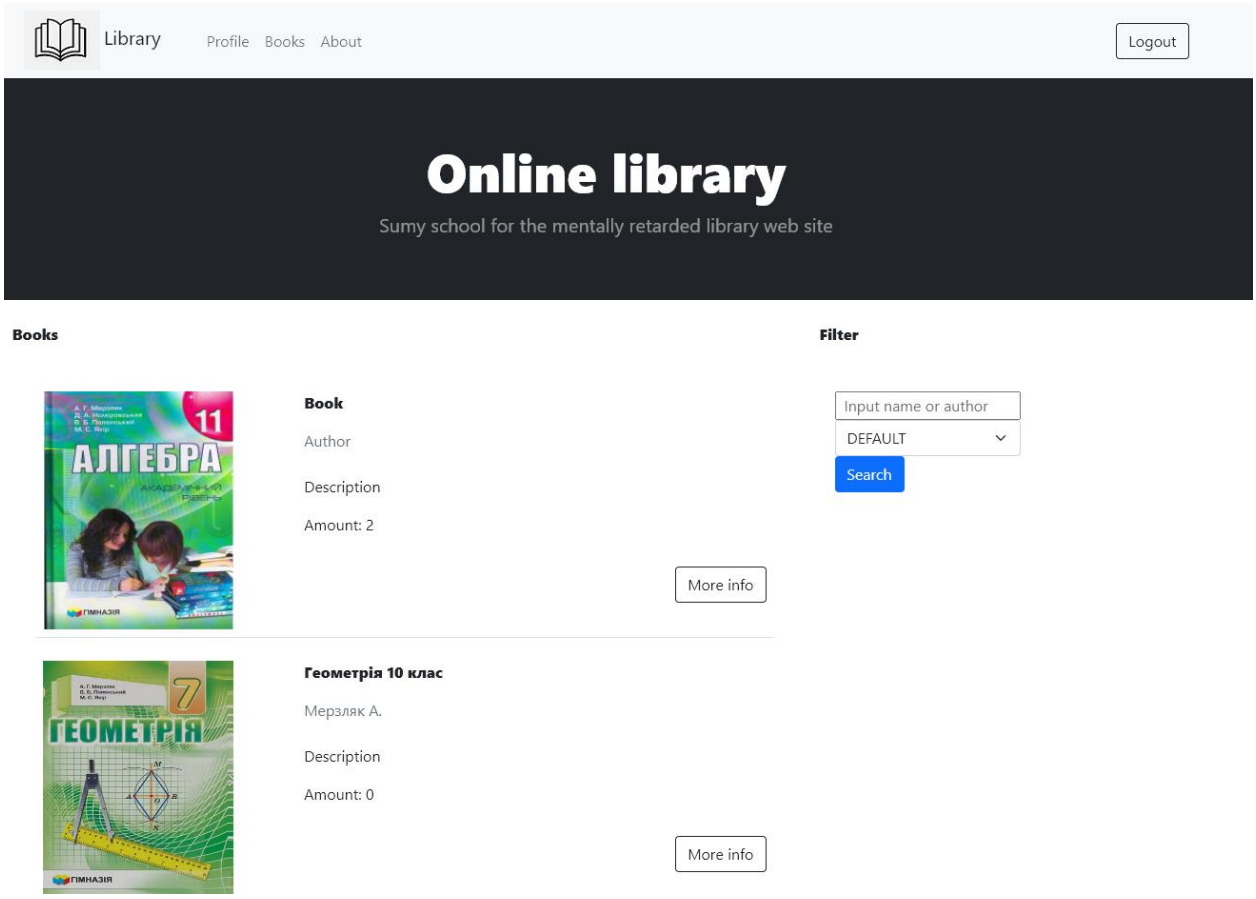

Рисунок 3.12 – Сторінка списку книг

У проекті також була розроблена сторінка профілю (рис. 3.13), на якій користувач може переглянути свою особисту інформацію та список книг, які він взяв у бібліотеці.

На сторінці профілю користувачу відображаються основні дані про нього, такі як ім'я, прізвище, електронна пошта та інші важливі відомості. Крім того, користувач має можливість змінювати свої особисті дані.

Окремий розділ на сторінці профілю присвячений списку книг, які користувач взяв у бібліотеці. Кожна книга представлена зі своїм заголовком, автором та іншою додатковою інформацією.

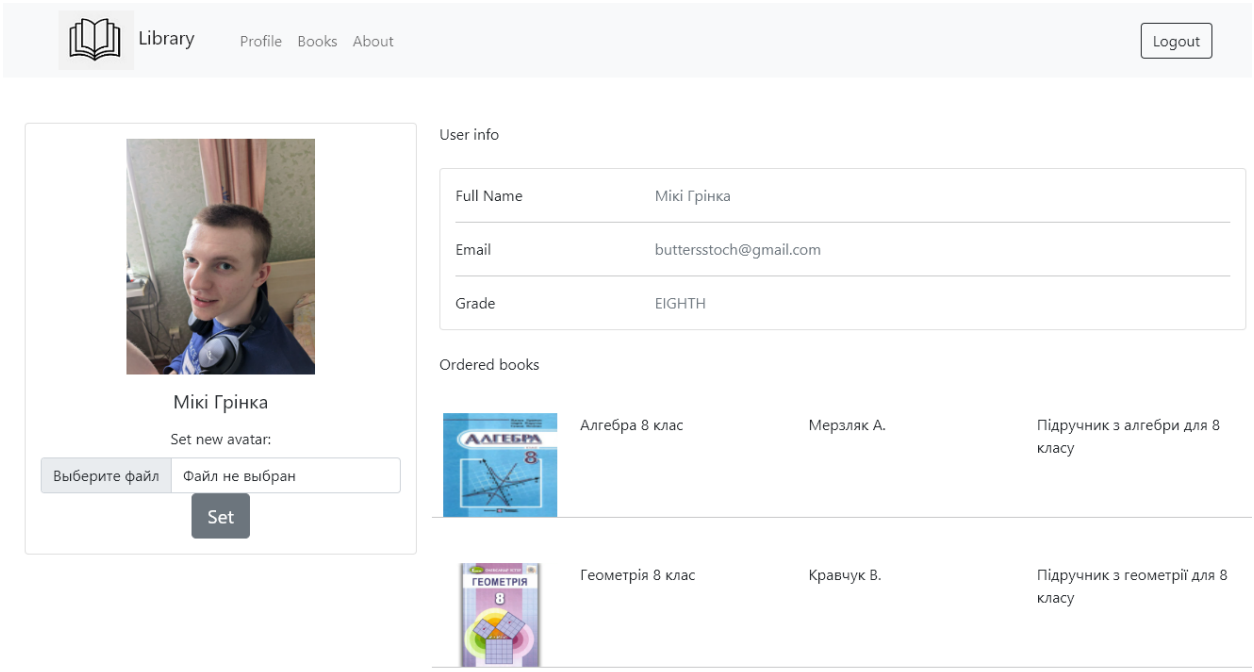

Рисунок 3.13 – Сторінка профіля користувача

3.6 Тестування

Проект та його окремі компоненти були успішно протестовані з метою забезпечення якості та надійності програмного продукту. У процесі тестування були використані різні методики, такі як модульні тести, інтеграційні тести та функціональні тести. Результати тестування були успішними, що свідчить про коректну роботу проекту та його відповідність функціональним та якісним вимогам. Під час тестування були виявлені та виправлені потенційні проблеми, що дозволило покращити якість та стабільність системи. Протестований функціонал включає різноманітні частини проекту, включаючи авторизацію та аутентифікацію користувачів, роботу з базою даних, взаємодію з файлами, адміністративну панель, а також функціонал, пов'язаний з користувачами та книгами. Кожна з цих складових була вичерпно протестована з метою переконатися в її правильному функціонуванні та відповідності вимогам проекту. Завдяки успішному процесу тестування ми забезпечили високу якість та надійність нашого проекту, що сприятиме задоволенню користувачів та успішному функціонуванню системи.

ВИСНОВКИ

У процесі виконання кваліфікаційної роботи було успішно розроблено веб-додаток для шкільної бібліотеки, який забезпечує зручний та ефективний доступ до ресурсів для учнів та вчителів. Проект пройшов крок за кроком, виконуючи різні завдання, щоб забезпечити його успішну реалізацію.

У першому етапі було проведено аналіз створення інформаційних систем, що дозволило зрозуміти основні вимоги та функціональність, які має мати шкільна бібліотека. Далі було ретельно досліджено і проаналізовано подібні сервіси та інтернет-платформи, з метою виявлення недоліків та визначення можливих покращень для нашого продукту.

Після цього було проведено детальний аналіз різних технологій, доступних для створення веб-додатків.

Весь проект та його окремі частини були ретельно протестовані, щоб забезпечити їх стабільну та надійну роботу.

Розроблений веб-додаток може бути легко розширений для включення додаткових функцій. Одним з напрямків розширення може бути додавання аудіокниг та відеоматеріалів. Це дозволить користувачам мати доступ до різноманітного контенту та розширити можливості навчання та розвитку. Крім того, є можливість покращити дизайн сайту, забезпечивши більш сучасний та привабливий інтерфейс. Це може включати оновлення кольорової схеми, використання більш естетичних елементів дизайну, поліпшення навігації та взаємодії з користувачем. До інших можливих розширень можуть належати додаткові функції, такі як можливість відстеження прогресу читання, обговорення книг між користувачами та багато іншого. Архітектура проекту забезпечує гнучкість та масштабованість, що дає можливість додавати новий функціонал без значних змін у вже існуючому коді.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Spring Documentation. [Electronic resource]. URL: https://spring.io/ (accessed: 20.04.2023).
2. Thymeleaf Documentation. [Electronic resource]. URL: https://www.thymeleaf.org/ (accessed: 02.05.2023).
3. Baeldung - Java, Spring, and Web Development tutorials. [Electronic resource]. URL: https://www.baeldung.com/ (accessed: 22.04.2023).
4. Walls, C. Spring in Action. // Manning Publications. 2016.
5. Greg L. Turnquist , Greg L. Turnquist. Learning Spring Boot 2.0 - Second Edition // Packt Publishing, 2017.
6. Claudio Eduardo de Oliveira, Greg L. Turnquist, Alex Antonov. Developing Java Applications with Spring and Spring Boot. // Packt Publishing, 2018.
7. Jon Duckett. HTML and CSS: Design and Build Websites // John Wiley & Sons, 2011.
8. Thomas Connolly, Carolyn Begg. Database Systems: A Practical Approach to Design, Implementation, and Management // Pearson, 2014.