МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ІНЖЕНЕРНИЙ НАВЧАЛЬНО – НАУКОВИЙ ІНСТИТУТ
ім. Ю.М. Потебні
ЗАПОРІЗЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ

КАФЕДРА ЕЛЕКТРОНІКИ, ІНФОРМАЦІЙНИХ СИСТЕМ ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Кваліфікаційна робота

перший (бакалаврський)
(рівень вищої освіти)

на тему **Розробка Веб-застосунку засобами фреймворку Spring**

Виконав: студент 4 курсу, групи 6.1210-пзс-з
спеціальності 121 Інженерія програмного забезпечення
(код і назва спеціальності)
освітньої програми Програмне забезпечення систем
(код і назва освітньої програми)
О.С. Ноздрюхіна
(ініціали та прізвище)
Керівник к.ф.-м.н.,доцент, Г.П. Коломоєць
(посада, вчене звання, науковий ступінь, підпис, ініціали та прізвище)
Рецензент директор ТОВ «Дискус»
Р.О. Лютий
(посада, вчене звання, науковий ступінь, підпис, ініціали та прізвище)

Запоріжжя
2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ІНЖЕНЕРНИЙ НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ім. Ю.М. Потебні
ЗАПОРІЗЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ

Кафедра електроніки, інформаційних систем та програмного забезпечення
Рівень вищої освіти _____перший (бакалавський)_____
Спеціальність ___121_Інженерія програмного забезпечення_____
(код та назва)
Освітня програма ___Програмне забезпечення систем_____
(код та назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри_____Тетяна КРИТСЬКА
“_01_” __березня_____2024_року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Ноздрюхіній Ользі Сергіївні
(прізвище, ім’я, по батькові)

1.Тема роботи Розробка Веб-застосунку засобами фреймворку Spring
керівник роботи _____Коломоєць Генадій Павлович, к.ф.-м.н., доцент__
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом ЗНУ від ___26.12.2023 № 2212-с_____

2. Строк подання студентом кваліфікаційної роботи_____21.05.2024_____
3. Вихідні дані кваліфікаційної роботи бакалавра
- комплект нормативних документів ;
- технічне завдання до роботи.

4. Зміст розрахунково – пояснювальної записки (перелік питань, які потрібно розробити)
- огляд та збір літератури стосовно теми кваліфікаційної роботи;
- огляд та аналіз існуючих рішень та аналогів;
- аналіз сучасних технологій для створення клієнтської частини Веб застосунків;
- проєктування Веб-застосунку;
- програмна реалізація застосунку.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
слайдів презентації

6. Консультанти розділів бакалаврської роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата Завдання прийняв

7. Дата видачі завдання 01.03.2024

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської роботи	Строк виконання етапів магістерської роботи	Примітка
1	Аналіз предметної області	25.01 – 29.01.24	виконано
2	Формулювання основної задачі дипломної роботи та узгодження її з науковим керівником	01.02 – 06.02.24	виконано
3	Аналіз існуючих методів рішення	07.02 – 11.02.24	виконано
4	Огляд та збір літератури стосовно теми кваліфікаційної роботи	12.02 – 26.02.24	виконано
5	Огляд та аналіз існуючих рішень та ан алогів	27.02 – 29.02.24	виконано
6	Вивчення засобів Spring для розробки Веб-застосунку	01.03 – 08.03.24	виконано
7	Визначення функціоналу застосунку, що розроблятиметься з використанням засобів Spring	09.03 – 17.03.24	виконано
8	Узгодження подальших дій з науковим керівником	18.03.24	виконано
9	Проєктування бази даних та створення Maven-проєкту застосунку	19.03 – 27.03.24	виконано
10	Проєктування та конструювання застосунку із використанням засобів Spring	28.03 – 29.04.24	виконано
11	Програмна реалізація застосунку та його апробація	30.04 – 06.05.24	виконано
12	Оформлення звіту	07.05 – 24.05.24	виконано
13	Оформлення презентації. Отримання рецензій від опонентів	25.05 – 27.05.24	виконано

Студент _____ О.С. Ноздрюхіна
(підпис) (прізвище та ініціали)

Керівник роботи _____ Г.П. Коломоєць
(підпис) (прізвище та ініціали)

Нормоконтроль пройдено

Нормоконтролер _____ І.А. Скрипник
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Сторінок – 72

Рисунків – 20

Таблиць – 1

Лістинги – 17

Джерел – 13

Ноздрюхіна О.С. Розробка Веб-застосунку засобами фреймворку Spring: кваліфікаційна робота бакалавра спеціальності 121 «Інженерія програмного забезпечення» / наук. керівник Г. П. Коломоєць. Запоріжжя : ЗНУ, 2024. 72 с.

Метою роботи є вивчення засобів фреймворку Spring на прикладі розробки Веб-застосунку «Музичні альбоми».

У роботі були вивчені підходи до розробки та проєктування Веб-застосунків, із використанням фреймворку Spring, архітектурного шаблону MVC, Spring Boot для простоти розгортання та управління конфігурацією, Spring Data JPA для ефективної роботи з базами даних та Thymeleaf для створення динамічних Веб-сторінок. Створення Apache Maven для управління залежностями проєкту та автоматизацію збірки.

Такі підходи та відповідні засоби розробки використані при розробці Веб-застосунку «Музичні альбоми», який дозволяє керувати колекцією музичних записів користувача.

Ключові слова: *Веб-застосунок, сутність, репозитарій, контролер, HTML-шаблон, база даних*.

ABSTRACT

Pages – 72

Drawings – 20

Tables – 1

Listings – 17

Sources – 13

Nozdiukhina O.S. Web Application Development Using the Spring Framework: qualification thesis of the bachelor's specialty 121 "Software engineering" / Science. manager H. P. Kolomoets. Zaporizhzhia: ZNU, 2024. 72 p.

The purpose of the work is to study the tools of the Spring framework on the example of the development of the Web application «Music albums».

During the work, approaches to the development and design of Web applications were studied, using the Spring framework, the MVC architectural pattern, Spring Boot for ease of deployment and configuration management, Spring Data JPA for efficient work with databases, and Thymeleaf for creating dynamic web pages. Creation of Apache Maven for project dependency management and build automation.

Such approaches and corresponding development tools were used in the development of the «Music Albums» Web application, which allows you to manage the user's music collection.

Keywords: *Web application, entity, repository, controller, HTML template, database*.

ЗМІСТ

ВСТУП

Актуальність теми

У сучасному світі, де Веб-технології стрімко розвиваються, зростає попит на ефективні засоби для спрощеного проєктування та розробки Веб-застосунків. Зокрема, використання фреймворків, таких як Spring, стає все більш актуальним та популярним серед розробників.

Однією з основних переваг фреймворку Spring є його здатність спрощувати процес проєктування Веб-застосунків. Фреймворк забезпечує структуру, яка дозволяє розробникам не витрачати зайвий час на налаштування проєкту.

Spring пропонує широкий вибір інструментів, які роблять розробку Веб-застосунків швидшою та ефективною. Наприклад, Spring Boot дозволяє створювати застосунки з мінімальними зусиллями. Spring Data спрощує роботу з базами даних, надаючи прості механізми для роботи з репозиторіями. Spring Security має набір інструментів для хорошої реалізації безпеки застосунків.

Таким чином, використання фреймворку Spring для розробки та проєктування Веб-застосунків є актуальною та важливою темою для сучасних розробників. Фреймворк Spring пропонує рішення, які дозволяють зменшити час на розробку, підвищити якість коду та забезпечити високу продуктивність застосунків.

Мета дослідження

Метою роботи є вивчення засобів фреймворку Spring на прикладі розробки Веб-застосунку «Музичні альбоми».

Завдання дослідження

Вивчити способи та засоби розробки Веб-застосунків за допомогою фреймворку Spring, сформулювати функціональні вимоги до Веб-застосунку

«Музичні альбоми», розробити структуру бази даних для зберігання сутностей, виконати проєктування архітектури та інтерфейсу, створити об’єктну модель проєкту з необхідними залежностями за допомогою Spring Boot, розробити клієнтську частину з використанням HTML шаблонів та CSS стилів, розробити серверну частину з використанням моделей, репозиторіїв та контролерів Spring. Виконати розгортання застосунку у контейнері сервлетів.

Об’єкт дослідження

Об'єктом дослідження є підходи та засоби, які надає фреймворк Spring для розробки типових Веб-застосунків, що зберігають інформацію у базах даних.

Предмет дослідження

Предметом дослідження є процес розробки Веб-застосунку «Музичні альбоми» засобами фреймворку Spring.

Методи дослідження

Аналіз та визначення функціональних вимог, проєктування реляційної схеми даних, проєктування архітектури та Веб-інтерфейсу користувача застосунку.

Практичне значення одержаних результатів

Практичне значення одержаних результатів роботи полягає в розробці повноцінного Веб-застосунку з використанням засобів фреймворку Spring, який забезпечує зручний доступ користувачам до музичних альбомів а також може бути функціонально розширений при подальшому.

Апробація одержаних результатів

Результати дослідження були представлені на XVII науково-практичній конференції студентів, аспірантів, докторантів і молодих вчених Запорізького національного університету «Молода наука-2024» [13]**Помилка! Джерело посилання не знайдено.**.

Глосарій

Веб-застосунок (Web Application) - це програмний застосунок, який працює на Веб-сервері і користувач може відкрити його через будь-який Веб-браузер.

Spring Framework - це фреймворк для розробки на основі Java, який має широкий набір інструментів для створення Веб-застосунків.

MVC (Model-View-Controller) – це архітектурний патерн, який розділяє застосунок на три основні компоненти: модель (дані), подання (інтерфейс користувача) та контролер (логіка взаємодії).

Spring Data JPA - модуль Spring Framework, який забезпечує спрощення роботи з базами даних за допомогою JPA.

Thymeleaf - це сучасний шаблонний рушій Java на стороні сервера, який дозволяє генерувати HTML-сторінки з динамічними даними.

Spring Boot - проєкт Spring, що спрощує налаштування та розгортання Java-застосунку за допомогою попередньо налаштованого середовища.

Maven - система управління проєктами на базі Java, яка автоматизує завантаження залежностей, компіляцію та розгортання.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд літературних джерел

У сучасному світі розробка Веб-застосунків є важливою складовою інформаційних технологій. Був час, коли десктопні програми, встановлені на комп'ютері, використовувалися майже для всього. Але минули роки, і більшість застосунків тепер доступні через браузер. Завдяки цьому вони стали набагато зручнішими. Більше нічого не потрібно встановлювати: сьогодні ці програми доступні з будь-якого пристрою, підключеного до інтернету, і користувачі можуть легко отримати доступ до них в будь-який час, з будь-якого місця. Один із потужних інструментів для створення таких застосунків — це фреймворк Spring. Його популярність зумовлена високою гнучкістю, модульністю та підтримкою великої кількості додаткових бібліотек та інструментів. Spring надає розробникам змогу створювати масштабовані, надійні та зручні у використанні застосунки, що є важливим у конкурентному середовищі інформаційних технологій [1].

Основні аспекти розробки Веб-застосунків з використанням Spring:

1. Інверсія управління (IoC) та впровадження залежностей (DI): Spring забезпечує інверсію управління (IoC), що дозволяє розробникам керувати залежностями між об'єктами через зовнішні конфігурації, замість жорсткого кодування в самих класах. Використання IoC сприяє створенню більш гнучких та підтримуваних застосунків. Впровадження залежностей (DI) в Spring досягається за допомогою анотацій, конфігурацій XML або Java-бінів [1].

2. Модель-представлення-контролер (MVC): Spring MVC є одним з основних модулів фреймворку, який дозволяє розробляти Веб-застосунки за архітектурою MVC. Ця архітектура сприяє розділенню логіки програми на моделі, види та контролери, що робить код більш структурованим та підтримуваним. Контролери відповідають за обробку HTTP-запитів і

керування логікою застосунка моделі містять дані та бізнес-логіку, а види відповідають за представлення даних користувачеві [1].

3. Безпека: Spring Security — це потужний модуль для організації безпеки Веб-застосунків. Він забезпечує аутентифікацію та авторизацію користувачів, захист від CSRF (Cross-Site Request Forgery — міжсайтове підроблення запиту) та інші важливі аспекти безпеки. Конфігурація Spring Security може здійснюватися як за допомогою анотацій, так і через XML-файли конфігурації [1].

4. Доступ до даних: Spring Data JPA полегшує роботу з базами даних завдяки автоматизації створення запитів та управління транзакціями. Це значно скорочує кількість коду, необхідного для роботи з даними. Spring Data JPA використовує об'єктно- реляційне відображення (ORM) для зв'язку об'єктів Java з записами в базах даних [3].

5. Тестування: Spring забезпечує інструменти для інтеграційного та модульного тестування, що дозволяє перевіряти коректність роботи компонентів застосунку на різних рівнях. Для цього використовуються такі інструменти, як JUnit і Spring TestContext Framework, які дозволяють писати тести для компонентів Spring та запускати їх у спеціальному тестовому контексті [3].

1.2 Аналіз програмних продуктів – аналогів

Веб-застосунків з’являється все більше на цьому ринку, тому розглянемо деякі приклади Веб-застосунків для музичних альбомів. Discogs — корисний ресурс для музичних фанатів і колекціонерів записів. Він допомагає досліджувати дискографії виконавців і організовувати власні колекції [5].

Discogs об'єднує глобальну спільноту шанувальників музики та колекціонерів записів, допомагаючи обліковувати дані про альбоми, композиції та виконавців. Система містить детальну інформацію про музичні релізи, включаючи треклисти, обкладинки, дати випуску та інше [5].

Особливості системи включають:

- Альбоми: Інформація про студійні, живі альбоми та інші типи релізів.
- Композиції: Окремі треки з детальною інформацією про виконавців, авторів та тривалість.
- Виконавці: Біографічні дані, дискографія, фотографії та зв'язки з іншими музикантами.

Комерціалізація записів на Discogs здійснюється через маркетплейс, де користувачі можуть купувати та продавати музичні релізи. Платформа також забезпечує публічну доступність великої частини своєї бази даних, дозволяючи користувачам переглядати інформацію про релізи, композиції та виконавців без необхідності реєстрації. Однак, для використання додаткових функцій, таких як купівля, продаж та ведення власної колекції, необхідно створити обліковий запис [4] [5].

Інтерфейс застосунку можемо побачити на рисунку 1.

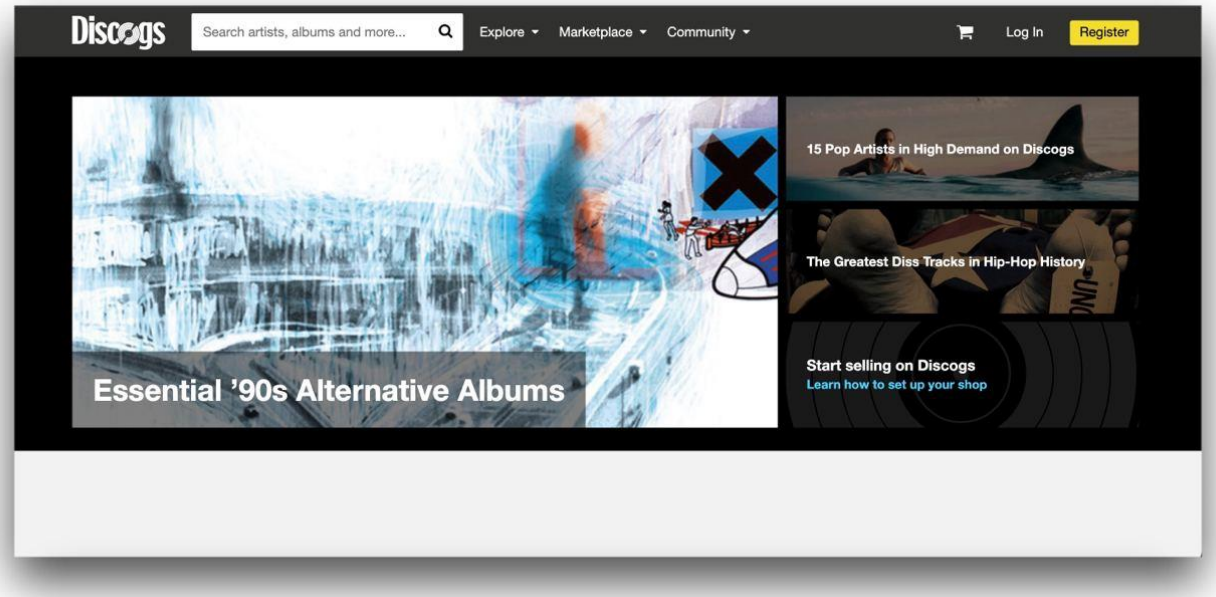

Рисунок *1– Інтерфейс Discogs*

Переваги:

- Discogs має одну з найбільших баз даних музичних записів, що охоплює різні жанри та формати.
- Користувачі можуть додавати і редагувати інформацію про альбоми, що дозволяє базі постійно оновлюватись.
- Надається можливість інтерактивного пошуку з різними фільтрами для точного знаходження записів.
- Можливість використання API для інтеграції з іншими застосунками та сервісами.

Недоліки:

- Для використання додаткових функцій, таких як купівля, продаж та ведення власної колекції, необхідно створювати обліковий запис.
- Не всі користувачі мають однакові права на редагування записів, що може обмежувати можливості внесення змін.

Rate Your Music (RYM) - RYM — це створена спільнотою база даних музики та фільмів, де ви можете оцінювати, переглядати, каталогізувати та відкривати нову музику, а також брати участь у створенні самої бази даних [6].

На сайті є можливість створювати власні списки улюблених альбомів, пісень чи виконавців. Підтримуються профілі, включаючи можливість завантажувати фотографії, створювати список друзів і спілкуватися за допомогою приватних повідомлень і вашої скриньки. Надається можливість отримувати автоматичні рекомендації щодо нової музики та фільмів на основі ваших особистих оцінок [6]. Інтерфейс Веб-застосунку можемо побачити на рисунку 2.

Переваги:

- Можливість залишати рецензії та оцінки альбомів.
- Використовується система рейтингів, яка допомагає у пошуку найпопулярніших записів.
- Інтерактивний та інтуїтивно зрозумілий інтерфейс пошуку.

Недоліки:

- У порівнянні з іншими сервісами не така велика база даних.
- Якісні і кількісні дані залежать від активності користувачів.

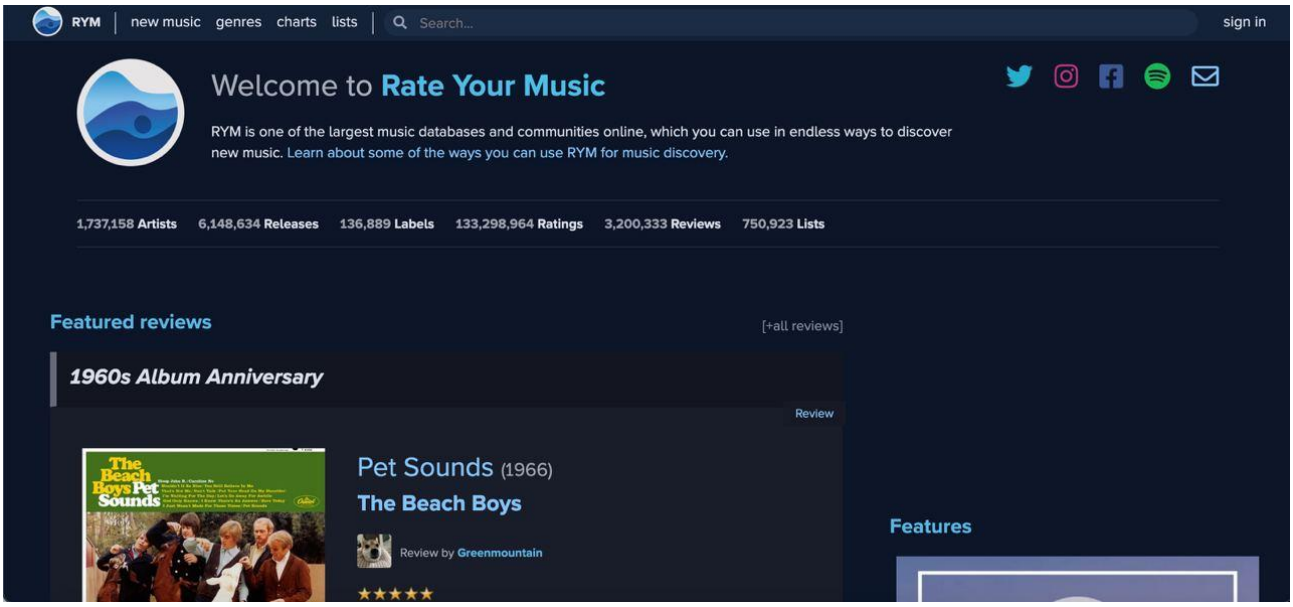

Рисунок *2– Інтерфейс Rate Your Music*

MusicBrainz - це музична база з відкритими даними. Платформа функціонує за принципом спільного редагування, де користувачі можуть додавати, змінювати та перевіряти дані. MusicBrainz містить інформацію про виконавців, альбоми, композиції та інші релізи, а також дати випуску та обкладинки альбомів [7].

Як і Вікіпедія, MusicBrainz підтримується глобальною спільнотою користувачів [8]. Інтерфейс Веб-застосунку можемо побачити на рисунку 3.

Переваги:

- MusicBrainz є відкритою базою даних, яку може використовувати кожен безкоштовно.
- Містить детальну інформацію про кожен альбом, включаючи такі дані як виконавець, назва альбому, дата випуску, список композицій та інше.
- Акаунти користувачів дозволяють редагувати та перевіряти інформацію про релізи, створювати та керувати плейлистами, вести особисті каталоги, брати участь у обговореннях, додавати мітки та оцінки.

Недоліки:

- Складність в навігації, так як платформа має багато вкладок і меню що ускладнює швидкий доступ до потрібної інформації.
- Кожна сторінка містить велику кількість інформації та опцій, що може перевантажувати користувача.

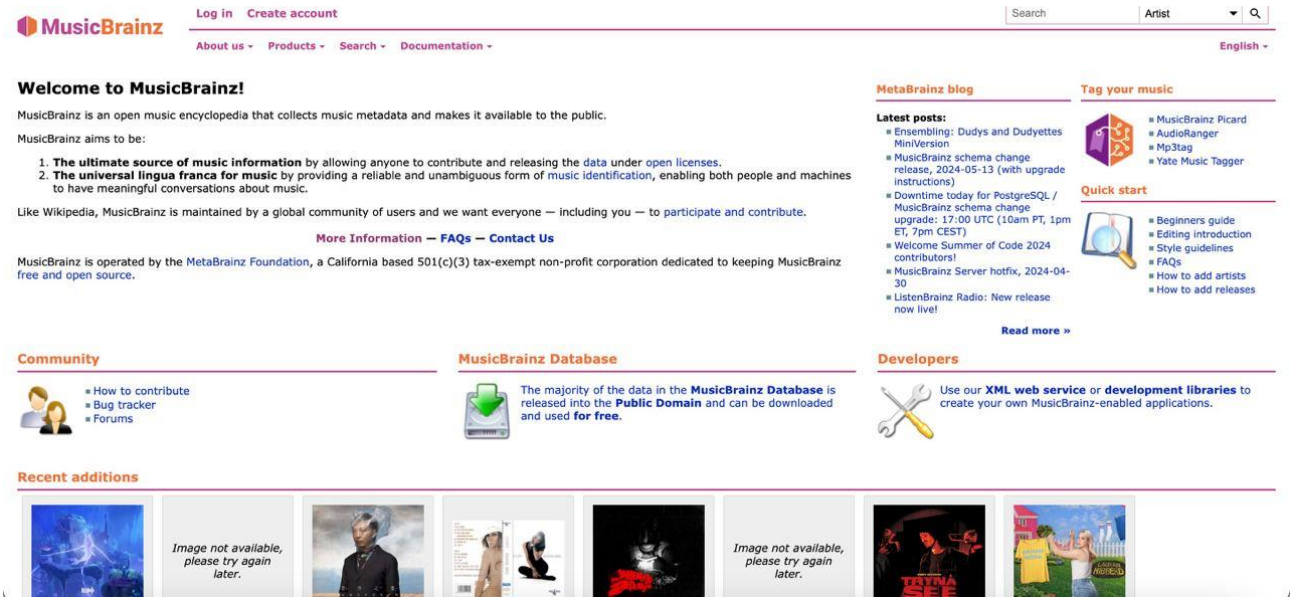

Рисунок *3 –Інтерфейс* MusicBrainz

1.3 Постановка завдання

Метою кваліфікаційної роботи є розробка Веб-застосунку «Музичні альбоми» засобами фреймворку Spring. Застосунок «Музичні альбоми» повинен надавати ефективні та зручні інструменти для роботи з музичними колекціями. Він має забезпечувати користувачам можливість додавати, редагувати та видаляти композиції з такими характеристиками, як назва, тривалість, жанр, виконавці композицій. Крім того, застосунок дозволяє формувати музичні альбоми з цих композицій. Для полегшення роботи з музичними колекціями, застосунок також повинен забезпечувати можливість пошуку по різних критеріях, таких як назва композиції, альбому, виконавець, група або жанр

Для досягнення поставленої мети необхідно вирішити наступні завдання:

1. Вивчити способи та засоби розробки Веб-застосунків що надає фреймворк Spring.
2. Сформулювати функціональні вимоги до Веб-застосунку «Музичні альбоми».
3. Розробити структуру бази даних для зберігання сутностей застосунку.
4. Виконати проєктування архітектури та інтерфейсу Веб-застосунку.
5. Розробити об'єктну модель проєкту з необхідними залежностями за допомогою Spring Boot.
6. Розробити клієнтську частину за допомогою шаблонів HTML та стилів CSS.
7. Розробити серверну частину застосунку за допомогою моделей, репозиторіїв та контролерів Spring.
8. Виконати розгортання застосунку у контейнері сервлетів.

2 ДОСЛІДЖЕННЯ ПРОГРАМНИХ ЗАСОБІВ РЕАЛІЗАЦІЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1 Огляд Java Веб-фреймворків Spring, GWT та Tapestry

Spring Framework забезпечує комплексну модель програмування та конфігурації для сучасних корпоративних програм на основі Java на будь-якій платформі розгортання. Ключовим елементом Spring є інфраструктурна підтримка на рівні застосунків: Spring зосереджується на «plumbing» корпоративних застосунків, щоб команди могли зосередитися на бізнес-логіці рівня застосунків без непотрібних зв’язків із певними середовищами розгортання [10].

Spring вважається безпечною, недорогою та гнучкою платформою, яка покращує ефективність кодування та скорочує загальний час розробки програми за рахунок ефективного використання системних ресурсів. Усуває виснажливу роботу з налаштуваннями.

Spring надає численні слабко пов’язані модулі або субфрейми, також відомі як шари, для покращення функціональності Веб-застосунку. Ці модулі, такі як Spring AOP, Spring Object Relational Mapping, Spring Web Flow і Spring Web model-view-controller (MVC), можна використовувати окремо або разом, забезпечуючи більшу гнучкість під час розробки [10].

Однією з основних рис Spring є її здатність виконувати ін’єкції залежностей, що є шаблоном програмування, який дозволяє розробникам створювати більш роз’єднані архітектури. Spring розуміє різні анотації Java, які розробник розміщує поверх класів, і може допомогти переконатися, що всі створені екземпляри мають належним чином заповнені залежності.

Ключовою перевагою Spring є те, що він усуває багато складнощів, пов’язаних із програмуванням на Java, і допомагає прискорити процеси розробки та тестування застосунків Ще одна перевага полягає в тому, що

Spring не змушує розробників успадковувати класи чи впроваджувати інтерфейси під час розробки.

За допомогою Spring розробники можуть легко писати код для використання API стійкості для зберігання та доступу до даних постійності в базі даних. Вони також можуть створювати Веб-застосунки, побудовані на архітектурі Spring Web MVC, Веб-фреймворку, побудованому на API Servlet з підтримкою різноманітних робочих процесів [10].

Переваги:

- Широкий вибір модулів: Spring пропонує різні модулі для різних потреб, наприклад Spring MVC, Spring Data, Spring Security і Spring Boot.
- Узгодженість: забезпечує узгоджену модель програмування та конфігурацію для різних модулів.
- Loose Coupling: сприяє слабкому зв’язку через ін’єкцію залежностей, роблячи програми більш модульними та легшими для тестування.
- Гнучка конфігурація: підтримує різні параметри конфігурації (наприклад, XML, анотації, конфігурація на основі Java).
- Spring Boot: спрощує налаштування та розробку застосунків Spring, надаючи готові конфігурації та угоди.
- Інтеграція: легко інтегрується з іншими технологіями та фреймворками, такими як Hibernate для ORM і Thymeleaf для шаблонів.
- Підтримка мікросервісів: Spring Cloud пропонує інструменти для створення архітектур мікросервісів, включаючи виявлення сервісів, автоматичні вимикачі та розподілене трасування.
- Висока продуктивність: оптимізовано для високої продуктивності та може легко обробляти великомасштабні програми.
- Spring Security: надає повний набір функцій безпеки для автентифікації, авторизації та захисту від поширених уразливостей.
- Зручне налаштування: широкі можливості налаштування відповідно до конкретних вимог безпеки.

Недоліки:

- Складність: широкий набір функцій Spring і різноманітні параметри конфігурації можуть бути надзвичайно важкими для новачків.
- Концепції та термінологія. Розуміння основних концепцій потребує значного навчання.
- Початкове налаштування: незважаючи на спрощення Spring Boot, початкове налаштування та налаштування програми Spring можуть бути складними та займати багато часу.
- Налаштування: для налаштування конфігурацій за замовчуванням може знадобитися глибоке знання різноманітних модулів Spring.
- Рівні абстракції: рівні гнучкості та абстракції у Spring можуть призвести до збільшення продуктивності, особливо у великих програмах.
- Потрібна оптимізація: вимагає ретельної оптимізації для забезпечення високої продуктивності та ефективного використання ресурсів.
- Шаблон: Програми Spring часто містять значну кількість шаблонного коду та файлів конфігурації.

Документація: для розуміння та підтримки конфігурації та налаштувань потрібна обширна документація.

GWT - це набір інструментів із відкритим вихідним кодом, випущений Google, щоб допомогти розробникам писати код Java і компілювати його в JavaScript для запуску в браузерах. Цей фреймворк для розробки Веб-застосунків містить низку унікальних функцій, таких як абстракція інтерфейсу користувача, сумісність із різними браузерами [9].

Основні модулі:

- GWT SDK: Містить компілятор та бібліотеки для створення клієнтських застосунків.
- GWT Designer: Інструмент для візуального проєктування інтерфейсів користувача.
- GWT-RPC: Протокол для забезпечення зв’язку між клієнтом та сервером.

Переваги:

- Дозволяє створювати програми, сумісні з різними браузерами, із автоматично створеним кодом JS.
- Включає файли JS, які можуть працювати з вихідними кодами Java основних браузерів.
- Хороша інтеграція з існуючими Java інструментами та середовищами розробки (Eclipse, IntelliJ IDEA).
- Не вимагає навичок фронтенду для створення зручних для користувача інтерфейсів.
- Підтримується командою досвідчених програмістів Google.

Недоліки:

- Може бути складно освоїти для початківців.
- Не підтримує деякі функції сучасних браузерів.
- Генерований JavaScript код може бути менш оптимізованим, що може впливати на продуктивність.
- Вимагає багато пам'яті для його запуску в режимі розробника, великий час компіляції.

Tapestry - дозволяє розробникам Java створювати динамічні, надійні та високомасштабовані Веб-застосунки. Він працює в будь-якому контейнері сервлетів або на сервері застосунків, оскільки покращує та розширює стандартний Java Servlet API. Будь-який сервер застосунків може запускати Tapestry, і він швидко інтегрується з усіма серверними частинами, такими як Spring, Hibernate тощо [9].

Основні модулі:

- Tapestry Core: Основні бібліотеки для розробки компонентів.
- Tapestry IOC: Модуль для інверсії керування, що забезпечує управління залежностями.
- Tapestry Hibernate: Інтеграція з Hibernate для роботи з базами даних.

Переваги:

- Забезпечує чітке розділення логіки та представлення через використання компонентів.
- Оптимізований для високої продуктивності та масштабованості.
- Відсутність залежності від інших бібліотек.
- Підтримка Ajax і чудові можливості звітування про винятки.

Недоліки:

- Менша кількість користувачів і ресурсів.
- Запуск кількох програм не підтримується.
- Неможливість динамічного додавання нових компонентів до існуючої сторінки
- Може вимагати більше часу на освоєння через меншу популярність серед розробників.

Таблиця 1 - Порівняння фреймворків Spring, GWT та Tapestry

	Spring	**GWT**	**Tapestry**
Крива навчання	Складна	Складна	Складна
Продуктивність	Відмінна	Висока	Відмінна
Екосистема	Обширна	Середня	Менша
Спільнота	Велика та активна	Середня	Менша
Гнучкість	Висока	Середня	Висока
Перевикористання компонентів	Високе	Високе	Високе
Підтримка мобільних пристроїв	Так	Так	Так
Інтеграція	Висока	Висока	Висока
Легкість налагодження	Середня	Висока	Середня
Підтримка мікросервісів	Так	Ні	Ні
Інверсія керування	Так	Так	Так
Підтримка документації	Висока	Середня	Висока
Підтримка оновлень	Часто	Рідко	Середня

Spring було обрано для розробки Веб-застосунку через:

1. Обширу екосистему:

- Spring Boot: Значно спрощує налаштування та конфігурацію застосунків, забезпечуючи швидкий старт розробки.

- Spring Cloud: Набір інструментів для створення мікросервісів, включаючи реєстрацію сервісів, балансування навантаження, розподілені конфігурації, тощо.
- Spring Data: Полегшує роботу з базами даних, підтримуючи різні типи баз даних та спрощуючи доступ до них.

2. Висока гнучкість:

- Конфігурація: Spring підтримує кілька способів конфігурації (анотації, XML, Java-based конфігурація), що дозволяє вибрати найзручніший для вашої команди.
- Модульність: Дозволяє використовувати тільки ті модулі, які необхідні для вашого проєкту, не перевантажуючи застосунок зайвими функціями.

3. Інтеграція:

- Легка інтеграція з іншими технологіями і сервісами, що дозволяє створювати гнучкі і потужні рішення.

4. Безпека:

- Spring Security: Потужний і гнучкий фреймворк для забезпечення безпеки застосунків, включаючи аутентифікацію, авторизацію та захист від поширених атак.

5. Широка підтримка та спільнота:

- Активна спільнота: Велика і активна спільнота розробників, що забезпечує доступ до численних ресурсів, таких як документація, форуми, приклади коду

6. Підходить для великомасштабних проєктів:

- Багато великих підприємств обирають Spring для своїх критично важливих застосунків через його надійність і масштабованість.
- Оптимізація: Оптимізований для високої продуктивності і може обробляти великі навантаження.

7. Інтеграція з популярними технологіями.

2.2 Огляд Spring

Фреймворк Spring - це прикладний фреймворк, який є частиною екосистеми Java. Прикладний фреймворк - це пакет типових функцій програмного забезпечення, що утворюють базову структуру для розробки програми. Прикладний фреймворк дозволяє витрачати менше зусиль при написанні програми, тому що не доводиться створювати весь код програми з нуля [2].

Фреймворк~~ом~~ Spring складається зекосистеми фреймворків. Як правило, коли говорять про фреймворк Spring, мають на увазі частину програмного функціоналу, яка включає наступне [2]:

Інверсію керування (IoC) - це принцип проєктування, який використовується для інверсії керування об’єктом об’єктами-залежностями. Spring реалізує IoC через ін’єкцію залежностей (DI), де об’єкти визначають свої залежності, а контейнер Spring впроваджує ці залежності під час створення компонентів [1]. Приклад ін’єкції залежності наведено в Лістингу 1.

Лістинг *1 Приклад ін’єкції залежності*

```
@Controller
public class AlbumsController {
    @Autowired
    private AlbumRepository albumRepository;
}
```

Аспектно-орієнтоване програмування (AOP) дозволяє відокремити наскрізні проблеми, такі як керування транзакціями та безпеку. Spring AOP надає спосіб динамічно додавати ці поведінки до коду без зміни фактичного коду. Це досягається за допомогою перехоплювачів методів і точкових скорочень, що забезпечує більш чистий і модульний код [2]. Приклад застосування аспектів наведено в Лістингу 2.

Лістинг *2 Приклад аспекту для логування у Spring*

```
@Aspect
@Component
public class LoggingAspect {
    @Before("execution(* com.example.service.*.*(..))")
    public void logBefore(JoinPoint joinPoint) {
        System.out.println(""+
joinPoint.getSignature().getName());
    }
}
```

Spring MVC — це фреймворк для створення Веб-застосунків з використанням шаблону проєктування Model-View-Controller. Він розділяє логіку застосунків на три взаємопов’язані компоненти, полегшуючи розробку Веб-застосунків, які можна масштабувати та підтримувати. Контролери обробляють вхідні запити та відповіді, моделі інкапсулюють бізнес-логіку, а представлення відповідають за рендеринг виводу для користувача [1]. Приклад використання команди наведено в Лістингу 3.

Лістинг *3 Приклад контролера у Spring MVC*

```
@Controller
public class AlbumsController {
@Autowired
private AlbumRepository albumRepository;

@GetMapping("/albums")
public String albums(@RequestParam(required = false) String
search, Model model) {
      Iterable<Album> albums;
      if (search != null && !search.isEmpty()) {
```

```
            albums =
albumRepository.findByAlbumNameContainingIgnoreCaseOrReleas
eYear(search, parseReleaseYear(search));
        } else {
            albums = albumRepository.findAll();
        }
        model.addAttribute("albums", albums);
        return "albums";
    }
```

Spring Security — це потужна структура аутентифікації та авторизації користувачів. Він надає комплексні послуги безпеки для застосунків Java, включаючи захист від поширених вразливостей, таких як CSRF (підробка запитів між сайтами, що дозволяє зловмиснику змусити користувача виконати небажані дії в застосунку), фіксація сесії (атака, коли зловмисник перехоплює або краде ідентифікатор сесії користувача) та клікджекінг (атака, що примушує користувача натискати на невидимий або прихований елемент на Веб-сторінці). Конфігурація може здійснюватися за допомогою анотацій або XML-файлів [3]. Приклад використання команди наведено в Лістингу 4.

Лістинг *4 Приклад конфігурації безпеки Spring Security*

```
@Configuration
@EnableWebSecurity
public class SecurityConfig extends
WebSecurityConfigurerAdapter {
    @Override
    protected void configure(HttpSecurity http) throws
Exception {
        http
   .authorizeRequests()
   .antMatchers("/admin/**").hasRole("ADMIN")
   .anyRequest().authenticated()
```

```
    .and()
    .formLogin().loginPage("/login").permitAll()
    .and()
    .logout().permitAll();
     }
}
```

Spring Data спрощує доступ до даних і маніпулювання ними, зменшуючи шаблонний код і покращуючи швидкість розробки. Він забезпечує узгоджену модель доступу до даних, включаючи підтримку реляційних баз даних (через Spring Data JPA), баз даних NoSQL і технологій Big Data. Цей модуль полегшує операції CRUD, розбиття на сторінки та генерацію динамічних запитів [2]. Приклад використання команди наведено в Лістингу 5.

Лістинг *5 Приклад репозиторію Spring Data JPA*

```
public interface UserRepository extends JpaRepository<User,
Long> {
List<User>findByLastName(String lastName);
}
```

Spring Boot базується на існуючому фреймворку Spring, спрощуючи процес створення готових програм. Він забезпечує типові параметри для різних конфігурацій, надаючи необхідні для кожної конфігурації залежності та узгоджуючи їх версії, тим самим зменшуючи потребу в шаблонному коді та дозволяючи розробникам зосередитися на написанні логіки програми. Spring Boot містить вбудовані сервери, що полегшує запуск окремих програм [10]. Приклад методу запуску застосунку наведено в Лістингу 6.

Лістинг *6 Приклад класу Spring Boot з метою запуску засосунку*

```
@SpringBootApplication
public class Application {
```

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

Підсумовуючи можливості, що надає фреймворк Spring, можна виділити такі його переваги:

- Модульність: модульна архітектура Spring дозволяє розробникам використовувати лише ті компоненти, які їм потрібні, сприяючи створенню більш легкої та ефективної програми.
- Масштабованість: Spring підтримує як невеликі, так і корпоративні програми, що робить його придатним для широкого спектру випадків використання.
- Велика спільнота: як один з найпопулярніших фреймворків Java, Spring має велику й активну спільноту, яка пропонує численні ресурси, навчальні посібники та сторонні бібліотеки.

2.3 Бази даних у Веб-застосунках

Бази даних відіграють ключову роль у Веб-застосунках, надаючи структурований спосіб зберігання, керування та отримання даних. Вони дозволяють програмам зберігати дані протягом сеансів, підтримують одночасний доступ і забезпечують цілісність і безпеку даних.

База даних — це організована колекція структурованої інформації, або даних, які зазвичай зберігаються в електронному форматі в комп’ютерній системі. Зазвичай базою даних керують за допомогою системи керування базами даних (СКБД). Дані, СКБД і програми, пов’язані з ними, разом називають системою баз даних, скорочено часто просто базою даних (БД). У найпоширеніших сучасних типах БД дані зазвичай представлено як рядки й стовпці в низці таблиць, щоб підвищити ефективність обробки та запитів щодо

даних. Після цього до даних можна легко отримати доступ, керувати ними, а також змінювати, оновлювати, контролювати та впорядковувати їх. Більшість баз даних використовують мову структурованих запитів (SQL) для запису й запитів даних [11].

Характеристики баз даних:

Зберігання даних: бази даних забезпечують структуроване середовище для зберігання даних, забезпечуючи ефективну організацію для легкого доступу та маніпулювання.

Отримання даних: ефективні можливості створення запитів дозволяють користувачам швидко отримувати дані, як правило, використовуючи SQL (мову структурованих запитів) для реляційних баз даних.

Управління даними: бази даних підтримують різні операції, включаючи вставку, оновлення, видалення та читання даних, які часто називають операціями CRUD (створення, читання, оновлення, видалення).

Цілісність даних: такі механізми, як обмеження, транзакції та нормалізація, забезпечують точність і послідовність даних.

Масштабованість: Бази даних можуть обробляти зростаючі обсяги даних і навантаження на користувачів, що досягається за допомогою вертикального масштабування (додавання додаткових ресурсів до існуючої бази даних) або горизонтального масштабування (розподіл бази даних на кількох серверах).

Безпека: Бази даних застосовують різні заходи безпеки для захисту даних, включаючи автентифікацію користувачів, контроль доступу, шифрування та аудит.

Контроль паралельності: Бази даних підтримують роботу кількох користувачів одночасно та гарантують, що одночасні транзакції не заважають одна одній, зберігаючи узгодженість даних.

Веб-застосунки можуть використовувати різні типи баз даних, кожна з яких має свої сильні сторони та випадки використання.

Основними типами є:

Реляційні бази даних (RDBMS): ці бази даних зберігають дані в таблицях із рядками та стовпцями та використовують мову структурованих запитів (SQL) для обробки даних. Загальні СКБД включають MySQL, PostgreSQL, Oracle і SQL Server.

Бази даних NoSQL: ці бази даних пропонують гнучкий дизайн схем і підходять для обробки великих обсягів неструктурованих або напівструктурованих даних. Типи баз даних NoSQL включають бази даних документів (наприклад, MongoDB), сховища ключ-значення (наприклад, Redis), сховища сімейства стовпців (наприклад, Cassandra) і бази даних графів (наприклад, Neo4j).

Бази даних у пам’яті: ці бази даних зберігають дані в пам’яті, щоб забезпечити надшвидкий доступ до даних, що корисно для кешування та аналітики в реальному часі. Приклади включають Redis і Memcached.

2.4 Інтеграція баз даних із Spring Framework

Фреймворк Spring забезпечує підтримку баз даних різних типів, спрощення розробки за допомогою абстракцій і інструментів Spring Data, Spring ORM і управління транзакціями. Використовуючи ці можливості, розробники можуть створювати надійні, масштабовані та безпечні Веб-застосунки з ефективними механізмами обробки даних. Використання MySQL Workbench додатково покращує процес розробки, надаючи потужні інструменти для проєктування баз даних і керування ними.

Використання Spring Framework надає численні переваги: Спрощений доступ до даних: Spring Data JPA забезпечує зручний інтерфейс для роботи з базами даних, дозволяючи виконувати CRUD-операції без написання громіздкого SQL-коду.

Об'єктно-реляційне відображення (ORM): Інтеграція з Hibernate дозволяє легко перетворювати Java-об'єкти в реляційні таблиці бази даних і навпаки, що робить роботу з даними більш інтуїтивною та зручною.

Управління транзакціями: Spring забезпечує як декларативне, так і програмне управління транзакціями, що дозволяє гарантувати цілісність і консистентність даних навіть при одночасному доступі багатьох користувачів.

Пул з'єднань: Використання пулінгу з'єднань, наприклад, через HikariCP, підвищує ефективність роботи з базою даних, зменшуючи накладні витрати на відкриття і закриття з'єднань [12].

Безпека і цілісність даних: Spring Security надає потужні механізми для автентифікації та авторизації, що забезпечує захист доступу до даних, а також підтримка обмежень та валідацій гарантує цілісність даних на рівні застосунку.

Завдяки цим особливостям Spring Framework забезпечує розробку надійних, масштабованих та безпечних Веб-застосунків, що робить його ідеальним вибором для проєкту.

3 РОЗРОБКА ВЕБ-ЗАСТОСУНКУ «МУЗИЧНІ АЛЬБОМИ» ЗАСОБАМИ ФРЕЙМВОРКУ SPRING

3.1 Опис предметної області

Розробка Веб-застосунку «Музичні альбоми» забезпечує зручне управління музичними колекціями. Облік музичних записів передбачає розробку таких сутностей:

1. Музичні копмозиції – можуть обліковуватися за назвою, тривалістю та жанром;
2. Альбоми - можуть обліковуватися за назвою і роком випуску;
3. Виконавці - можуть обліковуватися за ім’ям та прізвищем;
4. Групи - можуть обліковуватися за назвою;
5. Жанри - можуть обліковуватися за назвою;
6. Користувачі - можуть обліковуватися за ім’ям користувача та паролем.

Сутності та їх атрибути:

1) Музичні композиції:
   - song_id: Унікальний ідентифікатор композиції.
   - song_name: Назва композиції, яка використовується для ідентифікації та пошуку пісень.
   - duration_minutes: Тривалість композиції у хвилинах, що допомагає точно визначити загальну тривалість.
   - duration_seconds: Тривалість композиції у секундах, що дозволяє врахувати навіть короткі композиції.
2) Альбоми:
   - album_id: Унікальний ідентифікатор альбому.
   - album_name: Назва альбому, яка використовується для його ідентифікації та пошуку.
   - release_year: Рік випуску альбому, що дозволяє класифікувати альбом за роком випуску.

3) Виконавці:
    - artist_id: Унікальний ідентифікатор виконавця.
    - first_name: Ім'я виконавця, яке використовується для його ідентифікації.
    - last_name: Прізвище виконавця, яке використовується для його ідентифікації.
4) Групи:
    - group_id: Унікальний ідентифікатор групи.
    - group_name: Назва групи, яка використовується для її ідентифікації та пошуку.
5) Жанри (genre):
    - genre_id: Унікальний ідентифікатор жанру.
    - genre_name: Назва жанру, яка використовується для ідентифікації та пошуку жанру.
6) Користувачі:
    - id: Унікальний ідентифікатор користувача, що дозволяє однозначно ідентифікувати кожного користувача у системі.
    - username: Ім'я користувача для входу в систему, яке використовується для аутентифікації.
    - password: Пароль для аутентифікації користувача.

Система надає можливість виконувати основні операції з цими сутностями, такі як додавання, редагування, видалення та перегляд, що робить управління музичними колекціями зручним та ефективним для користувачів.

3.2 Функціональні вимоги до застосунку

Основні функціональні можливості застосунку:

1. Зберігання даних про музичні альбоми, включаючи назву, дату релізу та перелік пісень.

2. Зберігання даних про пісні, включаючи назву, тривалість, жанр та виконавців.
3. Зберігання даних про виконавців, включаючи ім'я та прізвище артиста або назву групи.
4. Пошук за назвою альбому, назвою пісні, жанром, ім'ям або прізвищем артиста, назвою групи.
5. Аутентифікація користувачів для забезпечення доступу тільки довірених користувачів.

3.3 Архітектура системи

Архітектура системи Веб-застосунку «Музичні альбоми» побудована на основі багаторівневої моделі, яка включає базу даних (Database), серверну частину (Backend) та клієнтську частину (Frontend). Такий підхід забезпечує чітке розділення обов'язків, спрощує масштабування та підтримку системи.

База даних реалізована на основі MySQL. Вона містить таблиці для зберігання даних про альбоми, пісні, жанри, артистів, групи та користувачів застосунку.

Серверна частина реалізована на основі фреймворку Spring і складається з таких компонентів:

- Моделі даних (Data Models): Моделі даних визначають структуру таблиць бази даних та використовуються для передачі даних між компонентами застосунку. Вони позначені анотаціями @Entity та @Table.
- Репозиторії (Repositories): Репозиторії забезпечують доступ до бази даних, використовуючи Spring Data JPA. Вони виконують CRUD-операції (створення, читання, оновлення, видалення) над об'єктами бази даних. Репозиторії позначені анотацією @Repository.
- Контролери (Controllers): Контролери обробляють HTTP-запити від клієнта, викликають відповідні методи репозиторіїв для доступу до

даних і повертають відповіді. Вони використовують анотації Spring MVC (@Controller, @RequestMapping) для маршрутизації запитів.

- Клієнтська частина реалізована за допомогою HTML, CSS та Thymeleaf для шаблонізації. Основні функції клієнтської частини включають:
- Інтерфейс користувача: Забезпечення зручного та зрозумілого інтерфейсу для взаємодії з системою.
- Відображення даних: Відображення списків даних, результатів пошуку та деталей окремих альбомів.
- Форми для введення даних: Форми для додавання, редагування та видалення альбомів, жанрів, пісень, артистів, груп а також форми для реєстрації та автентифікації користувачів.

Користувачі взаємодіють з клієнтською частиною через Веб-браузер: Вони заповнюють форми та відправляють запити до серверної частини потім контролери приймають запити таким чином, що викликають відповідні методи репозиторіїв для обробки запитів і доступу до бази даних. Далі репозиторії виконують необхідні CRUD-операції та повертають результати контролерам. Контролери відправляють їх клієнтській частині а вже потім вона відображає результати користувачам.

На рисунку 4 представлена структура проєкту, яка відображає шари архітектури.

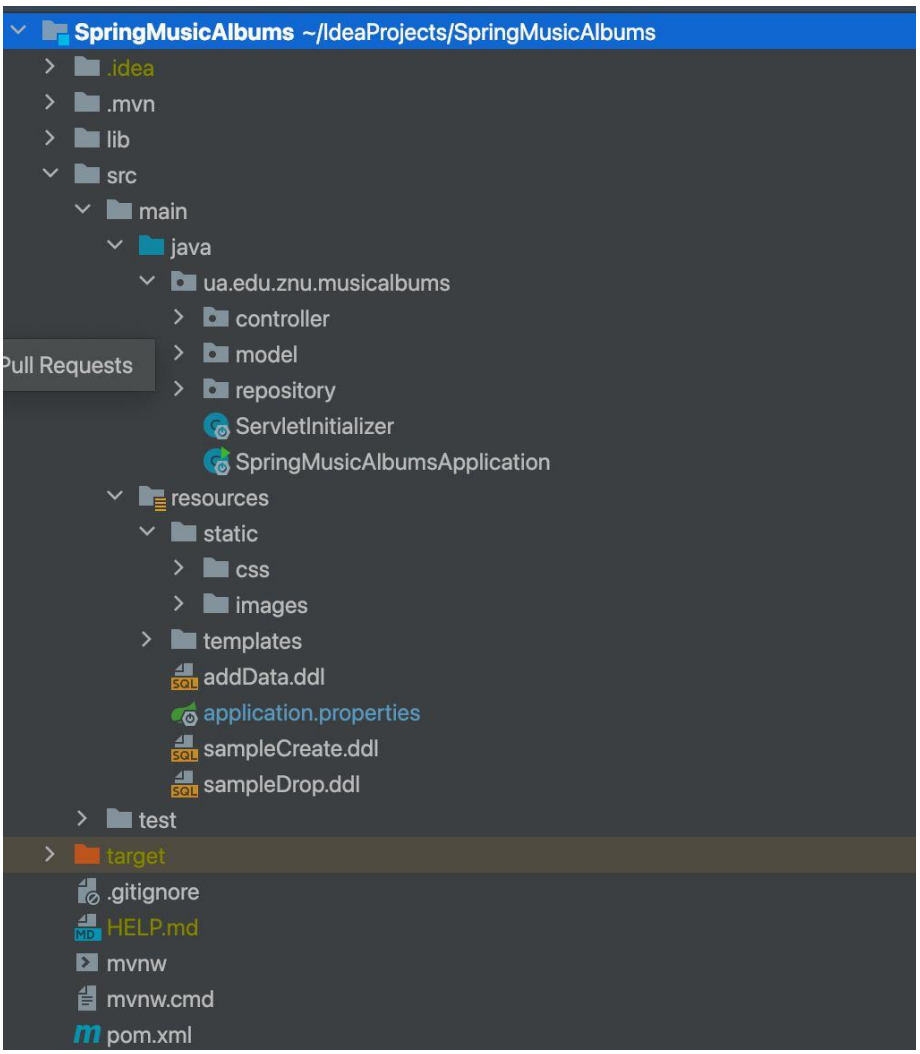

Рисунок *4 Структура проєкту*

3.4 Проєктування Веб-застосунку «Музичні альбоми»

Одним із етапів проєктування застосунку є створення діаграм використання (Use – Case) для визначення функціональності застосунка.

Для візуалізації функціональності та взаємодії між акторами (користувачами) і системою (Веб-застосунком «Музичні альбоми») була створена Use – Case діаграма.

Елементи Use – Case діаграми представленої на рисунку 5-6 включають:

1. Актори:

- Користувач: основний актор, який взаємодіє з застосунком.

2. Використані сценарії:

а) Сценарій для управління альбомами

- Вхід в систему: Користувач входить у систему, використовуючи свої облікові дані (наприклад, ім'я, електронну пошту, пароль тощо).
- Перегляд списку альбомів: Користувач може переглядати всі альбоми в таблиці.
- Додавання альбому: Користувач додає новий альбом до таблиці.
- Редагування альбому: Користувач редагує інформацію про існуючий альбом, змінюючи інформацію у формі.
- Пошук альбомів: Користувач здійснює пошук альбомів за назвою альбому або роком релізу.
- Видалення альбому: Користувач має кнопку видалення альбому, що дозволяє видалити його зі списку а також нашої бази даних.
- Вихід із системи: Користувач може натисну ти кнопку виходу, доступна в будь-який момент.

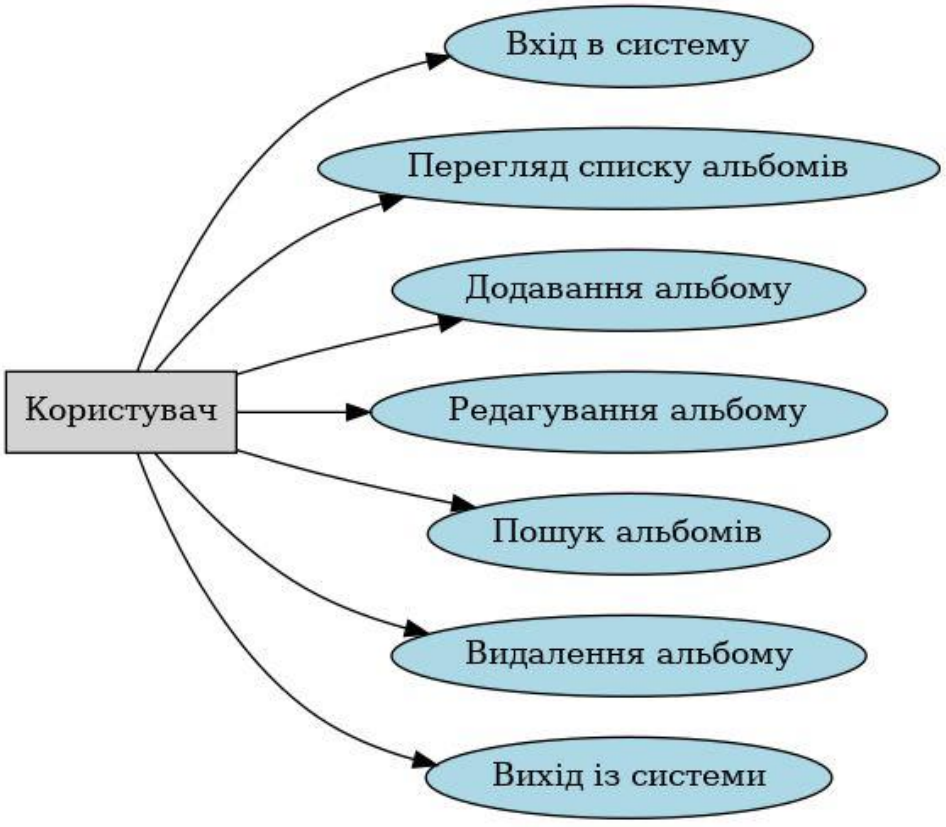

Рисунок 5 –*UseCase діаграма можливостей взаємодії користувача з застосунком*

b) Сценарій для управління артистами

- Реєстрація в системі: Користувач переходить на сторінку реєстрації і створює новий обліковий запис (вводить ім’я та пароль).
- Вхід в систему: Користувач входить у систему, використовуючи свої облікові дані (ім'я та пароль).
- Перегляд списку артистів: Користувач може переглядати всіх артистів в таблиці.
- Додавання артиста: Користувач додає нового артисту до таблиці.
- Редагування артиста: Користувач редагує інформацію про існуючого артиста в таблицю, змінюючи інформацію при занесенні даних в певну форму.
- Пошук артистів: Користувач здійснює пошук артистів за ім’ям або прізвищем артиста.

Рисунок 6 – *UseCase діаграма можливостей взаємодії користувача з застосунком*

Для опису взаємодії користувача з застосунком «Музичні альбоми» було створено діаграми послідовностей (sequence diagram), (Див. Рис.7-9). В них показано послідовні кроки, які відбуваються під час реєстрації, додання альбому та видалення альбому.

Рисунок *7 – Діаграма послідовності реєстрації*

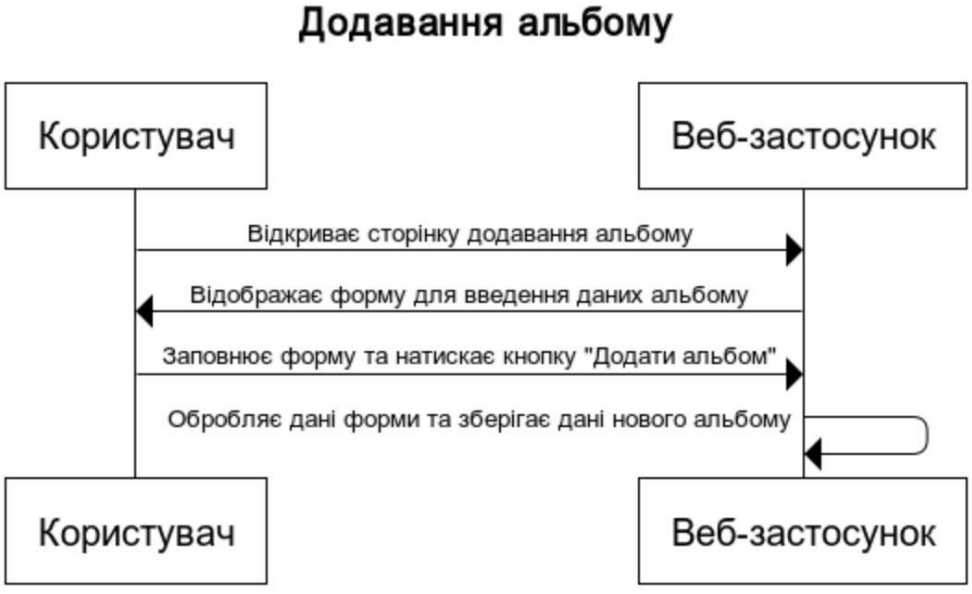

Рисунок *8 – Діаграма послідовності додання нового альбому*

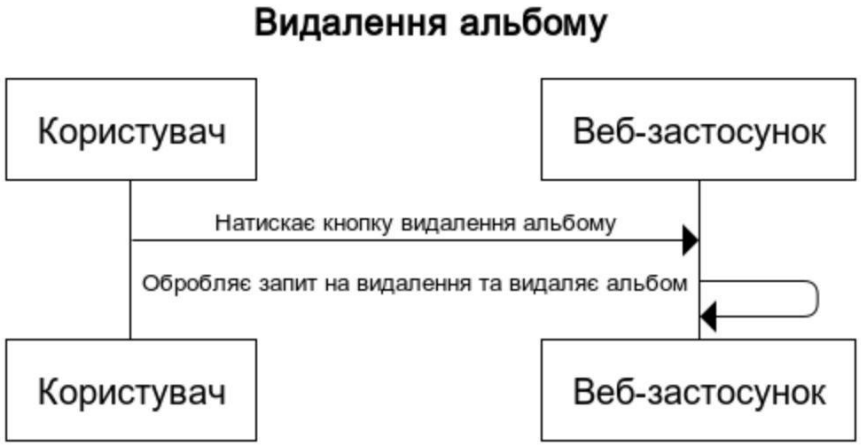

Рисунок 9 – *Діаграма послідовності видалення альбому*

3.4.1 Розробка бази даних для Веб-застосунку

База даних для Веб-застосунку «Музичні альбоми» містить таблиці які мають інформацію про альбоми, пісні, артистів, жанри, групи та користувачів.

Структура бази даних:

1. Таблиця "albums":
    a. поле "album_id": унікальний ідентифікатор;
    b. поле "album_name": ім’я альбому;
    c. поле "release_year": рік випуску альбому.
2. Таблиця "artists":
    a. поле "artist_id": унікальний ідентифікатор;
    b. поле "first_name": ім’я артиста;
    c. поле "last_name": прізвище артиста;
3. Таблиця "genres":
    a. поле "genre_id": унікальний ідентифікатор;
    b. поле "genre_name": назва жанру.
4. Таблиця "groups":
    a. поле "group_id": унікальний ідентифікатор;
    b. поле "group_name": назва групи.

5. Таблиця "songs":
    a. поле "song_id": унікальний ідентифікатор;
    b. поле "song_name": назва пісні;
    c. поле "duration_minutes": тривалість пісні в хвилинах;
    d. поле "duration_seconds": тривалість пісні в секундах.
6. Таблиця "users":
    a. поле "id": унікальний ідентифікатор користувача;
    b. поле "password": пароль користувача;
    c. поле "username": ім'я користувача.
7. Таблиця "album_artist_group":
    a. поле "id": унікальний ідентифікатор;
    b. поле "album_id": зовнішній ключ до таблиці “albums" ;
    c. поле "artist_id": зовнішній ключ до таблиці "artists" ;
    d. поле "group_id": зовнішній ключ до таблиці "groups" .
8. Таблиця "album_songs":
    a. поле "album_id": зовнішній ключ до таблиці "albums";
    b. поле "song_id": зовнішній ключ до таблиці "songs".

Взаємозв'язки між таблицями:

- Альбом може мати багато пісень а пісня може бути включена до декількох альбомів, тому таблиці albums та songs мають тип зв'язку «багато-до-багатьох». Таблиця album_songs зберігає зв'язок між альбомами та піснями.
- Альбом може бути випущений кількома артистами та/або групами, що складаються з артистів, тому створена зв'язкова таблиця album_artist_group, яка містить зовнішні ключі до таблиць albums, artists, та groups і реалізує зв’язок «багато-до-багатьох» між ними.
- Кожна пісня може належати до одного жанру, тому таблиці songs та genres пов’язані зв’язком типу «один-до-багатьох» .

На рисунку 10 представлена діаграма "сутність-зв'язок" бази даних застосунку.

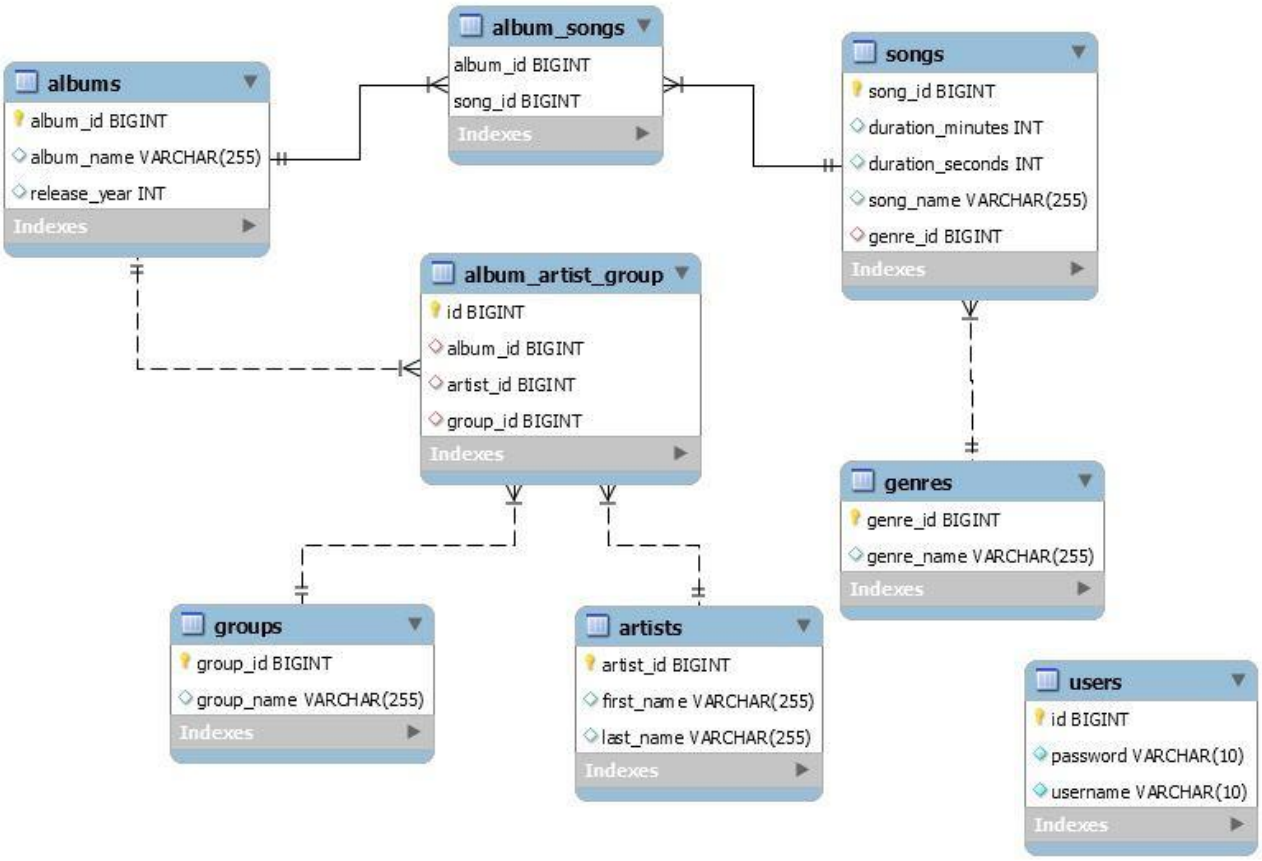

Рисунок *10 – Діаграма таблиць БД*

1. Були розроблені скрипти на мові SQL:

1) Для створення таблиць, організації їх зв’язків та цілісності даних - ці скрипти забезпечують створення необхідних таблиць та встановлюють зв’язки між ними за допомогою зовнішніх ключів.

Повний текст SQL-скриптів для створення таблиць та організації їх зв'язків наведено у Додатку А.

2) Для внесення тестових даних - ці скрипти дозволяють заповнити базу даних тестовими даними для перевірки її роботи.

Повний текст SQL-скриптів для внесення тестових даних наведено у Додатку Б.

Додавання пов'язаної таблиці та організація зв'язку наведено нижче у лістингах 7 і 8.

Лістинг *7 Лістинг коду класу Song*

```
import jakarta.persistence.*;
import lombok.Data;
import lombok.EqualsAndHashCode;
import lombok.ToString;
import java.util.LinkedHashSet;
import java.util.Set;

@Data
@Entity
@Table(name = "songs")
public class Song {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "song_id", nullable = false)
    @ToString.Exclude
    private Long id;

    @Column(name = "song_name", nullable = false)
    private String songName;

    @Column(name = "duration_minutes", nullable = false)
    private Integer durationMinutes;

    @Column(name = "duration_seconds")
    private Integer durationSeconds;

    @ToString.Exclude
    @EqualsAndHashCode.Exclude
@ManyToOne(cascade = {CascadeType.PERSIST,
CascadeType.MERGE})
@JoinColumn(name = "genre_id")
private Genre genre;
```

```
@ToString.Exclude
@EqualsAndHashCode.Exclude
@ManyToMany(mappedBy = "songs", cascade =
CascadeType.PERSIST)
    private Set<Album> albums = new LinkedHashSet<>();
}
```

Лістинг 8 *Лістинг коду класу Genre*

```
import java.util.LinkedHashSet;
import java.util.Set;
import lombok.*;
import jakarta.persistence.*;

@Data
@Entity
@Table(name = "genres")
public class Genre {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "genre_id", nullable = false)
    private Long id;
    @ToString.Exclude

    @Column(name = "genre_name", nullable = false)
    private String name;

    @ToString.Exclude
     @EqualsAndHashCode.Exclude
     @OneToMany(mappedBy = "genre", cascade =
     CascadeType.PERSIST, orphanRemoval = true)
      private Set<Song> songs = new LinkedHashSet<>();
}
```

У класі **Genre** використовується анотація **@OneToMany**, що вказує на зв'язок з багатьма об'єктами **Song**. Поле **songs** містить набір пісень, які відносяться до певного жанру.

У класі **Song** використовується анотація **@ManyToOne** та **@JoinColumn** що вказує на стовпець **genre_id**, який зберігає посилання на жанр, до якого відноситься пісня.

Також у класі Song використовується анотація **@ManyToMany** для зв'язку з класом **Album**, що вказує на можливий зв'язок пісні з кількома альбомами.

3.4.2 Розробка об’єктної моделі Maven для застосунку

Для розробки об’єктної моделі в Maven застосунку слід створити файл pom.xml, який є основним описом проєкту. Цей файл визначає залежності, плагіни, профілі та інші конфігурації, необхідні для збирання та управління проєктом. Нижче у лістингу наведено частину файлу pom.xml. Повний текст файлу pom.xml наведено у Додатку В.

Лістинг *9 Файл pom.xml*

```
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-
jpa</artifactId>
    </dependency>
<dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-
thymeleaf</artifactId>
   </dependency>
```

```
    <dependency>
        <groupId>com.mysql</groupId>
        <artifactId>mysql-connector-j</artifactId>
        <scope>runtime</scope>
    </dependency>
    <dependency>
        <groupId>org.projectlombok</groupId>
        <artifactId>lombok</artifactId>
        <optional>true</optional>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-devtools</artifactId>
        <scope>runtime</scope>
        <optional>true</optional>
    </dependency>
</dependencies>
```

- **spring-boot-starter-data-jpa:**

Забезпечує підтримку Java Persistence API (JPA) для роботи з базами даних, включаючи реалізацію Hibernate.

- **spring-boot-starter-thymeleaf**:

Забезпечує інтеграцію шаблонізатора Thymeleaf для створення динамічних HTML-сторінок у Spring Boot застосунку.

- **mysql-connector-j:**

Підключає Spring Boot застосунок до MySQL бази даних через JDBC.

- **lombok:**

Зменшує шаблонний код в Java за допомогою анотацій для автоматичного генерування методів getter, setter та інших.

- **spring-boot-devtools:**

Забезпечує інструменти для розробки, такі як автоматичне перезавантаження застосунку при зміні коду.

3.4.3 Розробка шару моделі застосунку

Модель є сутністю, яка зберігається в базі даних. Для прикладу розглянемо лістинг коду 10 для моделі артистів.

Лістинг *10 Лістинг коду класу Artist*

```
import jakarta.persistence.*;
import lombok.Data;
import lombok.ToString;

@Data
@Entity
@Table(name = "artists")
public class Artist {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "artist_id", nullable = false)
    @ToString.Exclude
    private Long id;
    @Column(name = "first_name")
    private String firstName;
    @Column(name = "last_name")
    private String lastName;
}
```

Клас **Artist** є сутністю JPA, яка відображається на таблицю **artists** у базі даних і використовує анотації з пакета jakarta.persistence для визначення мапінгу, а також бібліотеку Lombok для автоматичного створення гетерів, сетерів та інших методів. Поле **id** є первинним ключем, визначеним як

стовпець **artist_id**, який не може бути null. Поля **firstName** та **lastName** відповідають стовпцям **first_name** та **last_name** відповідно. Анотації **@Data**, **@Entity**, **@Table**, **@Id**, **@GeneratedValue** та **@Column** забезпечують відповідну поведінку та мапінг класу до таблиці в базі даних.

3.4.4 Розробка репозиторіїв для доступу до баз даних

Репозиторії забезпечують доступ до бази даних. Вони використовують JPA або інші інструменти для виконання CRUD-операцій (Create, Read, Update, Delete). Для прикладу розглянемо лістинг коду 11 для репозиторію артистів.

Лістинг *11 Лістинг коду класу ArtistRepository*

```
import org.springframework.data.repository.CrudRepository;
import ua.edu.znu.musicalbums.model.Artist;

import java.util.List;

public interface ArtistRepository extends
CrudRepository<Artist, Long> {
    List<Artist>
findByFirstNameContainingIgnoreCaseOrLastNameContainingIgno
reCase(String firstName, String lastName);
}
```

ArtistRepository розширює **CrudRepository** і використовується для роботи з базою даних артистів у Spring Data JPA. Додатково він містить метод **findByFirstNameContainingIgnoreCaseOrLastNameContainingIgnoreCase**, який знаходить всіх артистів.

3.4.5 Розробка шару контролерів застосунку

Контролери обробляють HTTP-запити, взаємодіють з репозиторіями для отримання або збереження даних та повертають відповідні представлення користувачеві. Для прикладу розглянемо нижче лістинг 12 коду для контролера артистів.

Лістинг *12 Лістинг коду класу ArtistsController*

```
importorg.springframework.beans.factory.annotation.Autowire
d;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import
org.springframework.web.bind.annotation.RequestParam;
import ua.edu.znu.musicalbums.model.Artist;
import ua.edu.znu.musicalbums.repository.ArtistRepository;
@Controller
public class ArtistsController {
    @Autowired
    private ArtistRepository artistRepository;
    @GetMapping("/artists")
    public String artists(@RequestParam(required = false)
String search, Model model) {
        Iterable<Artist> artists;
        if (search != null && !search.isEmpty()) {
            artists =
artistRepository.findByFirstNameContainingIgnoreCaseOrLastN
ameContainingIgnoreCase(search, search);
        } else {
            artists = artistRepository.findAll();
        }
```

```
        model.addAttribute("artists", artists);
        return "artists";
    }
    @PostMapping("/artistEditForm")
    public String artistEditForm(@RequestParam Long
artistId, Model model) {
        Artist artist =
artistRepository.findById(artistId).orElse(new Artist());
        model.addAttribute("artist", artist);
        return "artistedit";
    }
    @PostMapping("/artistEdit")
    public String artistEdit(@RequestParam Long artistId,
@RequestParam String artistFirstName,
                             @RequestParam String
artistLastName, Model model) {
        Artist artist =
artistRepository.findById(artistId).orElse(new Artist());
        artist.setFirstName(artistFirstName);
        artist.setLastName(artistLastName);
        artistRepository.save(artist);
        return "redirect:/artists";
    }
    @PostMapping("/artistRemove")
    public String artistRemove(@RequestParam Long artistId)
{

artistRepository.findById(artistId).ifPresent(artistReposit
ory::delete);
        return "redirect:/artists";
    }
    @GetMapping("/artistAddForm")
    public String artistAddForm() {
```

```
        return "artistadd";
    }
    @PostMapping("/artistAdd")
    public String artistAdd(@RequestParam String
artistFirstName,
                            @RequestParam String
artistLastName) {
        Artist artist = new Artist();
        artist.setFirstName(artistFirstName);
        artist.setLastName(artistLastName);
        artistRepository.save(artist);
        return "redirect:/artists";
    }
}
```

Код представляє контролер **ArtistsController**, який обробляє HTTP-запити, пов'язані з артистами, з використанням анотацій Spring MVC для маршрутів. Контролер ін'єктує залежність **artistRepository** за допомогою анотації **@Autowired** для взаємодії з базою даних артистів.

Метод **artists** обробляє GET-запит. Якщо параметр search не пустий, виконується пошук артистів за ім'ям або прізвищем, інакше повертається список всіх артистів. Результати додаються до моделі під атрибутом **artists** і повертається ім'я шаблону **artists** для відображення.

Метод **artistEditForm** обробляє POST-запит. Він отримує ідентифікатор a **artistId** і шукає його в репозиторії. Якщо артист не знайдений, створюється новий об'єкт **Artist**. Артист додається до моделі під атрибутом **artist**, і повертається ім'я шаблону **artistedit** для редагування.

Метод **artistEdit** обробляє POST-запит. Він отримує ідентифікатор артиста, ім'я та прізвище. Йде пошук артиста за ідентифікатором, і якщо він не знайдений, створюється новий об'єкт. Оновлюються ім'я та прізвище артиста.

Метод **artistRemove** обробляє POST-запит. Він отримує ідентифікатор артиста і видаляє його з репозиторію, якщо він існує.

Метод **artistAddForm** обробляє GET-запит і повертає ім'я шаблону artistadd для форми додавання артиста.

Метод **artistAdd** обробляє POST-запит. Він отримує ім'я та прізвище нового артиста, створює об'єкт **Artist**, зберігає його у репозиторії.

3.5 Розробка клієнтської частини за допомогою HTML-шаблонів з використанням рушія Thymeleaf та стилів CSS.

Клієнтська частина застосунку була розроблена з використанням HTML-шаблонів для відображення даних та CSS для їх стилізації. Шаблони HTML з використанням Thymeleaf забезпечили динамічне відображення даних на Веб-сторінках. CSS використовується для покращення зовнішнього вигляду Веб-сторінок, забезпечуючи привабливий дизайн та зручність використання. HTML-шаблони включають форми для додавання, редагування та видалення альбомів, а також таблиці для відображення списку альбомів, пісень, жанрів, артистів та груп.

Як приклад шаблону розглянемо лістинг 13 albums.html використовується для відображення списку альбомів та надання можливості пошуку альбомів за назвою або роком випуску.

Лістинг *13 Лістинг коду для шаблону albums.html*

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>Albums</title>
    <link rel="stylesheet" type="text/css" media="all"
href="../../css/home.css" th:href="@{/css/home.css}"/>
```

```
</head>
<body>
<section id="musicAlbum">
    <div class="container">
        <div th:insert="fragments/header :: header"></div>
        <form th:action="@{/albums}" method="get"
class="search-form">
            <input type="text" name="search"
placeholder="Search by albums..."class="search-input"/>
            <input type="submit"
value="Search"class="search-button"/>
        </form>
        <div class="tables">
            <table border="1">
                <caption>Albums</caption>
                <tr>
                    <th>Album id</th>
                    <th>Album name</th>
                    <th>Release Year</th>
                    <th>Edit album</th>
                <th>Remove album</th>
                </tr>
                <tr th:each="album : ${albums}">
                    <td th:text="${album.id}">albumId</td>
                    <td
th:text="${album.albumName}">albumName</td>
                    <td
th:text="${album.releaseYear}">releaseYear</td>
                    <td>
                        <form th:action="@{/albumEditForm}"
method="post">
                            <input type="hidden"
name="albumId" th:value="${album.id}"/>
```

```
                                <input type="submit"
value="Edit"/>
                            </form>
                        </td>
                        <td>
                            <form th:action="@{/albumRemove}"
method="post">
                                <input type="hidden"
name="albumId" th:value="${album.id}"/>
                                <input type="submit"
value="Remove"/>
                            </form>
                        </td>
                    </tr>
                </table>
                <a href="albumadd.html"
th:href="@{/albumAddForm}" class="add-link">Add album</a>
                <div th:insert="fragments/footer ::
footer"></div>
            </div>
</div>
</section>
</body>
</html>
```

Цей HTML-код використовує шаблонізатор Thymeleaf для створення Веб-сторінки, яка відображає список музичних альбомів з можливістю пошуку, редагування та видалення записів. Тег **<html>** визначає документ з використанням простору імен Thymeleaf. У блоці **<head>** містяться метадані, такі як кодування символів UTF-8, заголовок сторінки "**Albums**", і підключення CSS-файлу. Основний вміст сторінки розташований у блоці **<body>**, який містить розділ **<section>** з ідентифікатором musicAlbum. У

цьому розділі є контейнер ** ** з класом container, який містить заголовок та форму для пошуку альбомів з полем введення і кнопкою відправки. Контролер **AlbumsController** отримує запит на сторінку **/albums** і виконує пошук альбомів у базі даних, залежно від параметра пошуку. Список альбомів додається до об'єкта моделі як атрибут **«albums»**. Потім цей атрибут передається до HTML-шаблону **«albums.html»**. У шаблоні Thymeleaf теги, такі як **th:each**, використовуються для ітерації по списку альбомів і відображення даних у таблиці на сторінці.

Теги Thymeleaf, які дозволяють це зробити:

1. <tr th:each="album : ${albums}">:
    - Тег th:each дозволяє виконати ітерацію по списку альбомів, переданому від контролера як атрибут «albums». Для кожного елемента у списку створюється новий рядок таблиці, де album представляє поточний об'єкт у циклі.
2. <td th:text="${album.id}">albumId</td>:
    - Тег th:text дозволяє вставляти значення властивостей об'єкта альбому у відповідні клітинки таблиці. У цьому прикладі, значення album.id вставляється у відповідну клітинку таблиці.
3. <form th:action="@{/albumEditForm}" method="post">:
    - Тег th:action задає URL для обробки форми. У цьому випадку форма надсилає дані на /albumEditForm.
4. <input type="hidden" name="albumId" th:value="${album.id}"/>:
    - Тег th:value дозволяє встановлювати значення прихованого поля форми. У цьому випадку передається ідентифікатор альбому.
5. <input type="submit" value="Edit"/>:
    - Кнопка для надсилання форми редагування.
6. <form th:action="@{/albumRemove}" method="post">:
    - Вказує URL для видалення альбому.
7. <input type="hidden" name="albumId" th:value="${album.id}"/>:
    - Передає ідентифікатор альбому.

8. <input type="submit" value="Remove"/>:
    - Кнопка для надсилання форми видалення.
9. Add album:
    - Тег th:href використовується для встановлення динамічного URL для посилання. У цьому випадку, посилання веде на форму додавання нового альбому.
10. :
    - Тег th:insert використовується для включення фрагменту HTML-шаблону. У цьому випадку, вставляється фрагмент з іменем header з файлу fragments/header.

Одним із стилів, які використовуються у шаблоні альбомів, є стиль для таблиці. Розглянемо детальніше, як цей стиль визначається у CSS та як він застосовується у HTML-шаблоні який представлений на лістингу 14.

Лістинг *14 Лістинг коду для файлу стилів*

```
table {
    width: 100%;
    border-collapse: collapse;
    margin-top: 15px;
}
table caption {
    font-size: 20px;
    margin-bottom: 15px;
    color: #ffffff;
}
table th, table td {
    text-align: left;
    padding: 10px;
}
table th {
```

```
    background-color: rgb(7, 56, 38);
    font-family: 'Monserrat', sans-serif;
    color: rgb(255, 255, 255);
}
```

Пояснення CSS стилів:

1. table:
    - width: 100%; - встановлює ширину таблиці на 100% від ширини контейнера.
    - border-collapse: collapse; - зливає кордони таблиці та комірок в одинарні лінії.
    - margin-top: 15px; - додає відступ зверху таблиці.
2. table caption:
    - font-size: 20px; - встановлює розмір шрифту для заголовка таблиці.
    - margin-bottom: 15px; - додає відступ знизу заголовка таблиці.
    - color: #ffffff; - встановлює білий колір тексту заголовка таблиці.
3. table th, table td:
    - text-align: left; - вирівнює текст у комірках таблиці по лівому краю.
    - padding: 10px; - додає внутрішні відступи у комірках таблиці.
4. table th:
    - background-color: rgb(7, 56, 38); - встановлює темно-зелений колір фону для заголовків таблиці.
    - font-family: 'Monserrat', sans-serif; - застосовує шрифт 'Monserrat' до тексту заголовків таблиці.
    - color: rgb(255, 255, 255); - встановлює білий колір тексту заголовків таблиці.

3.6 Виконання розгортання застосунку у контейнері сервлетів

Останнім кроком було розгортання застосунку у контейнері сервлетів. Spring Boot автоматично налаштовує вбудований контейнер сервлетів (Tomcat за замовчуванням). Для цього достатньо запустити застосунок з відповідною командою, і він стає доступним для користувачів через еб-браузер.

- Запуск Spring Boot застосунку.

Головний клас **SpringMusicAlbumsApplication**, що запускає Spring Boot застосунок. Лістинг коду 15 класу **SpringMusicAlbumsApplication** наведено нижче.

Лістинг *15 Лістинг коду класу SpringMusicAlbumsApplication*

```
import jakarta.servlet.DispatcherType;
import lombok.extern.slf4j.Slf4j;
import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.SpringBootApplicatio
n;
import
org.springframework.boot.web.servlet.FilterRegistrationBean
;
import org.springframework.context.annotation.Bean;
import ua.edu.znu.musicalbums.controller.filter.AuthFilter;

@Slf4j
@SpringBootApplication
public class SpringMusicAlbumsApplication {

    public static void main(String[] args) {

SpringApplication.run(SpringMusicAlbumsApplication.class,
args);
```

```
    }
    /**
     * We register the AuthFilter as a Spring bean.
     *
     * @return AuthFilter registration bean
     */
    @Bean
    public FilterRegistrationBean<AuthFilter> authFilter()
{
        FilterRegistrationBean<AuthFilter> registration =
new FilterRegistrationBean<>();
        registration.setFilter(new AuthFilter());
        registration.addUrlPatterns("/home", "/logout",
"/albumassign",
                "/albums", "/albumEditForm", "/albumEdit",
"/albumRemove", "/albumAddForm", "/albumAdd",
"/genres", "/genreEditForm", "/genreEdit", "/genreRemove",
"/genreAddForm", "/genreAdd",
                "/artists", "/artistEditForm",
"/artistEdit", "/artistRemove", "/artistAddForm",
"/artistAdd",
                "/groups", "/groupEditForm", "/groupEdit",
"/groupRemove", "/groupAddForm", "/groupAdd",
                "/songs", "/songEditForm", "/songEdit",
"/songRemove", "/songAddForm", "/songAdd");

registration.setDispatcherTypes(DispatcherType.FORWARD,
DispatcherType.REQUEST);
        log.info("AuthFilter registered");
        return registration;
    }
}
```

AuthFilter перевіряє, що зі сторінками застосунку працює лише автентифікований користувач, шляхом перехоплення запитів до захищених сторінок. Коли запит надходить, фільтр спочатку намагається отримати поточну сесію користувача. Якщо сесія відсутня, користувача перенаправляють на сторінку входу. Якщо сесія існує, фільтр перевіряє наявність об'єкта користувача у сесії. Якщо користувача не знайдено, сесія анулюється, і користувач також перенаправляється на сторінку входу. Якщо користувач автентифікований, запит передається далі для обробки. Лістинг коду 16 класу **AuthFilter** наведено нижче.

Лістинг *16 Лістинг коду класу AuthFilter*

```
import jakarta.servlet.Filter;
import jakarta.servlet.FilterChain;
import jakarta.servlet.FilterConfig;
import jakarta.servlet.ServletException;
import jakarta.servlet.ServletRequest;
import jakarta.servlet.ServletResponse;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import jakarta.servlet.http.HttpSession;
import lombok.extern.slf4j.Slf4j;
import ua.edu.znu.musicalbums.model.User;
import java.io.IOException;
@Slf4j
public class AuthFilter implements Filter {
    @Override
    public void init(FilterConfig filterConfig) throws
ServletException {
        Filter.super.init(filterConfig);
    }
    @Override
```

```
    public void doFilter(ServletRequest request,
ServletResponse response, FilterChain chain) throws
IOException, ServletException {
        HttpServletRequest req = (HttpServletRequest)
request;
        HttpServletResponse resp = (HttpServletResponse)
response;
        resp.setContentType("text/html;charset=UTF-8");
        HttpSession session = req.getSession(false);
        if (session == null) {
            log.info("AuthFilter: session is null");
            resp.sendRedirect("/login");
        } else {
            String sessionID = session.getId();
            System.out.println("AuthFilter: sessionID = " +
sessionID);
            User user = (User)
session.getAttribute("user");
            if (user == null) {
                session.invalidate();
                log.info("AuthFilter: user is null");
                resp.sendRedirect("/login");
            } else {
                log.info("AuthFilter: user is {}",
user.getUsername());
                chain.doFilter(req, resp);
            }
        }
    }

    @Override
    public void destroy() {
        Filter.super.destroy();
```

```
    }
}
```

- Ініціалізація сервлету (клас ServletInitializer).

Забезпечує конфігурацію для розгортання Spring Boot застосунку у контейнері сервлетів. Лістинг коду 17 класу **ServletInitializer** наведено нижче.

Лістинг *17 Лістинг коду класу ServletInitialize*

```
import
org.springframework.boot.autoconfigure.SpringBootApplicatio
n;
import
org.springframework.boot.builder.SpringApplicationBuilder;
import
org.springframework.boot.web.servlet.support.SpringBootServ
letInitializer;
@SpringBootApplication
public class ServletInitializer extends
SpringBootServletInitializer {
    @Override
    protected SpringApplicationBuilder
configure(SpringApplicationBuilder application) {
        return
application.sources(SpringMusicAlbumsApplication.class);
    }
}
```

Клас **ServletInitializer** розширює **SpringBootServletInitializer** і використовується для налаштування Spring Boot застосунку при його розгортанні на зовнішньому Веб-сервері, такому як Tomcat. Він позначений анотацією **@SpringBootApplication** для автоматичної конфігурації. Метод

configure, перевизначений з **SpringBootServletInitializer**, налаштовує застосунок. Це дозволяє Spring Boot правильно ініціалізувати застосунок у контейнері сервлетів.

3.7 Інтерфейс Веб-застосунку «Музичні альбоми»

Інтерфейс Веб-застосунку є важливим елементом, який забезпечує взаємодію користувача з системою. Він включає в себе візуальні та функціональні компоненти, що дозволяють користувачу виконувати різноманітні дії.

Інтерфейс застосунку представлений на рисунках 11 – 20.

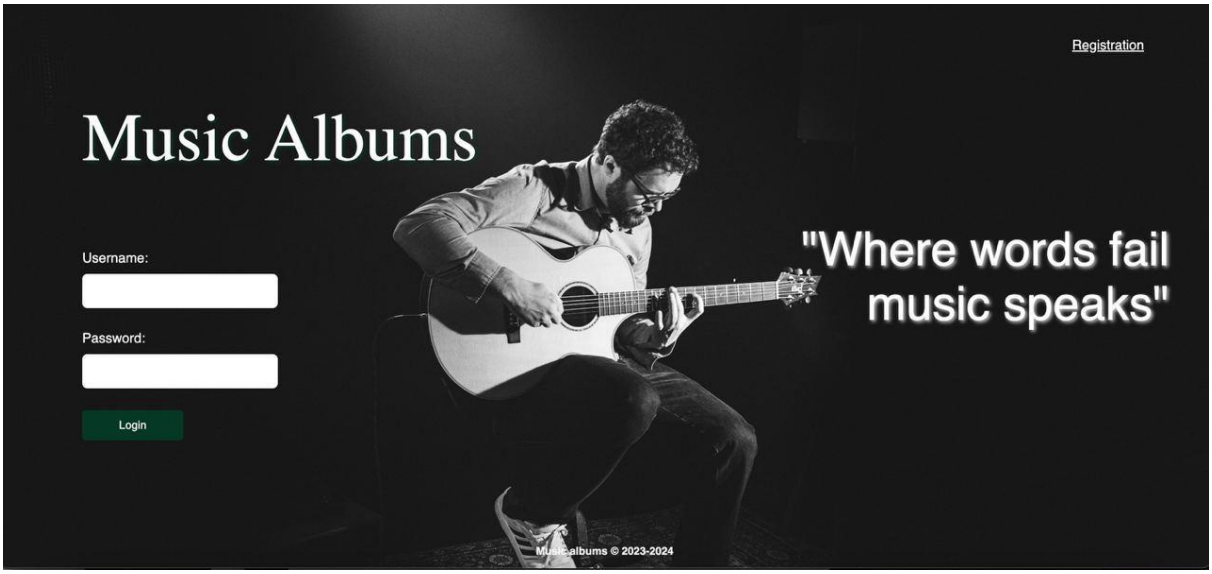

Рисунок *11– Перша сторінка застосунку «Музичні альбоми»*

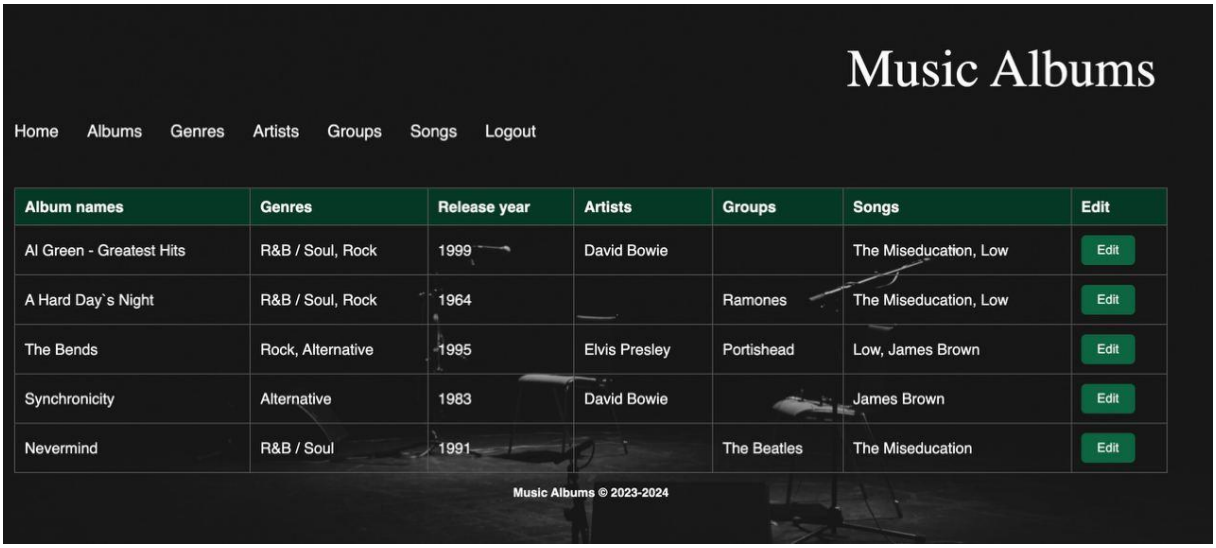

Рисунок *12 – Головна сторінка застосунку*

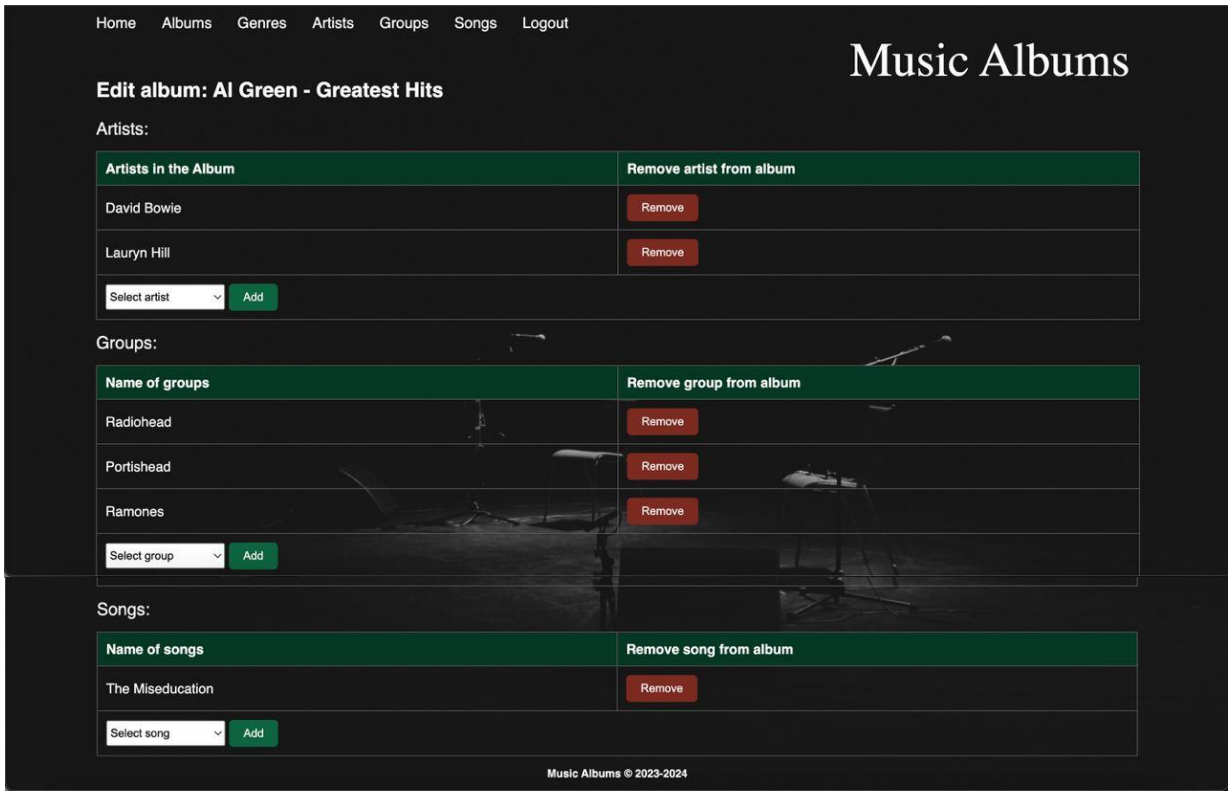

Рисунок *13 – Сторінка редагування даних музичного альбому*

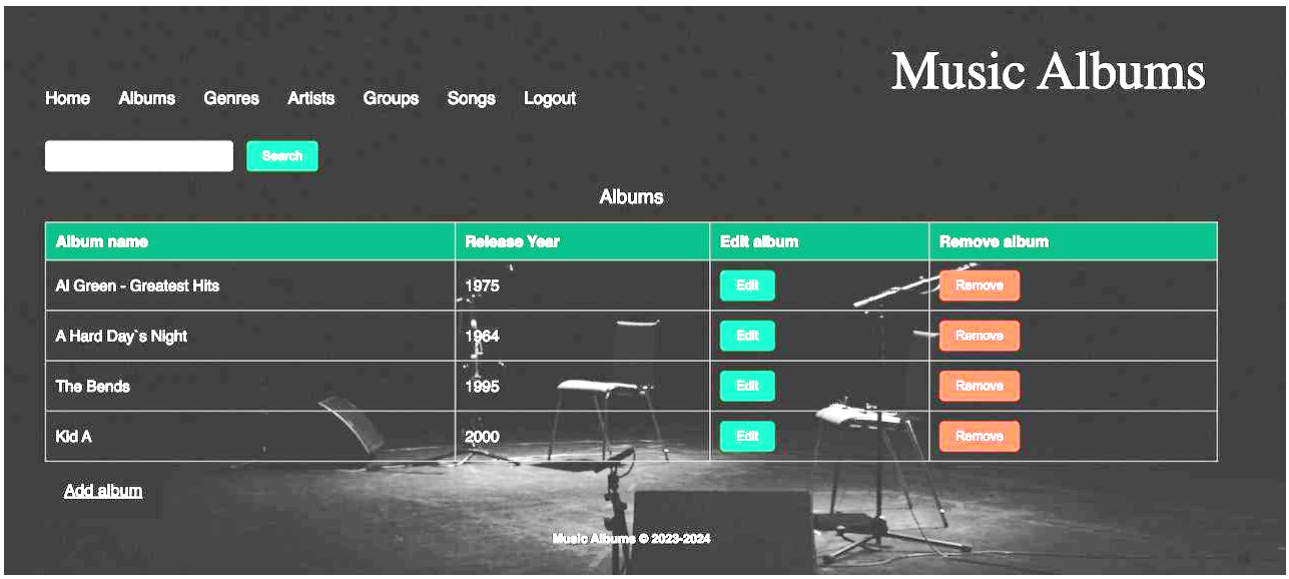

Рисунок *14* – *Сторінка довідника альбомів*

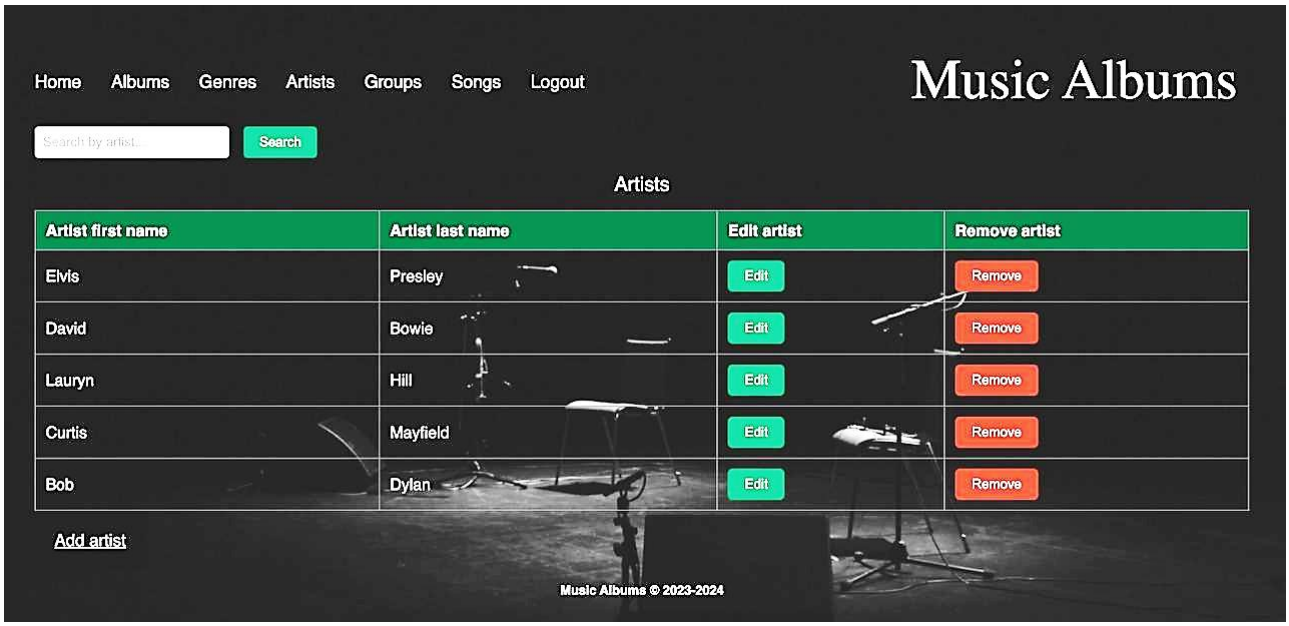

Рисунок *15* – *Сторінка довідника артистів*

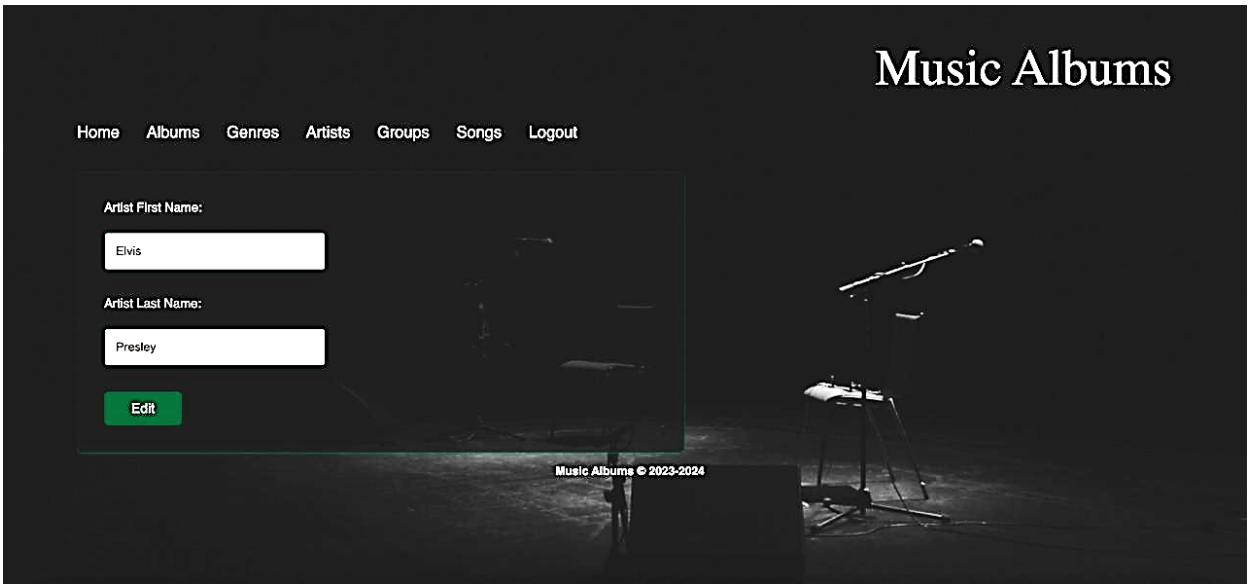

Рисунок *16* – *Форма для редагування артиста*

Рисунок *17*– *Форма для додання артиста*

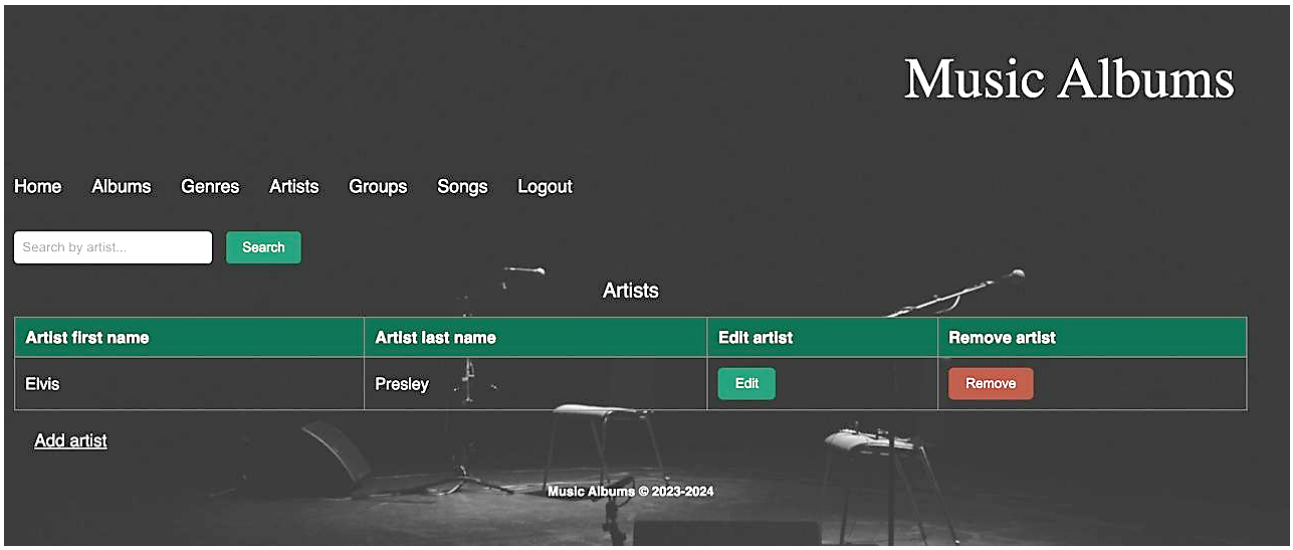

Рисунок *18* – *Відображається результату пошуку по імені артиста*

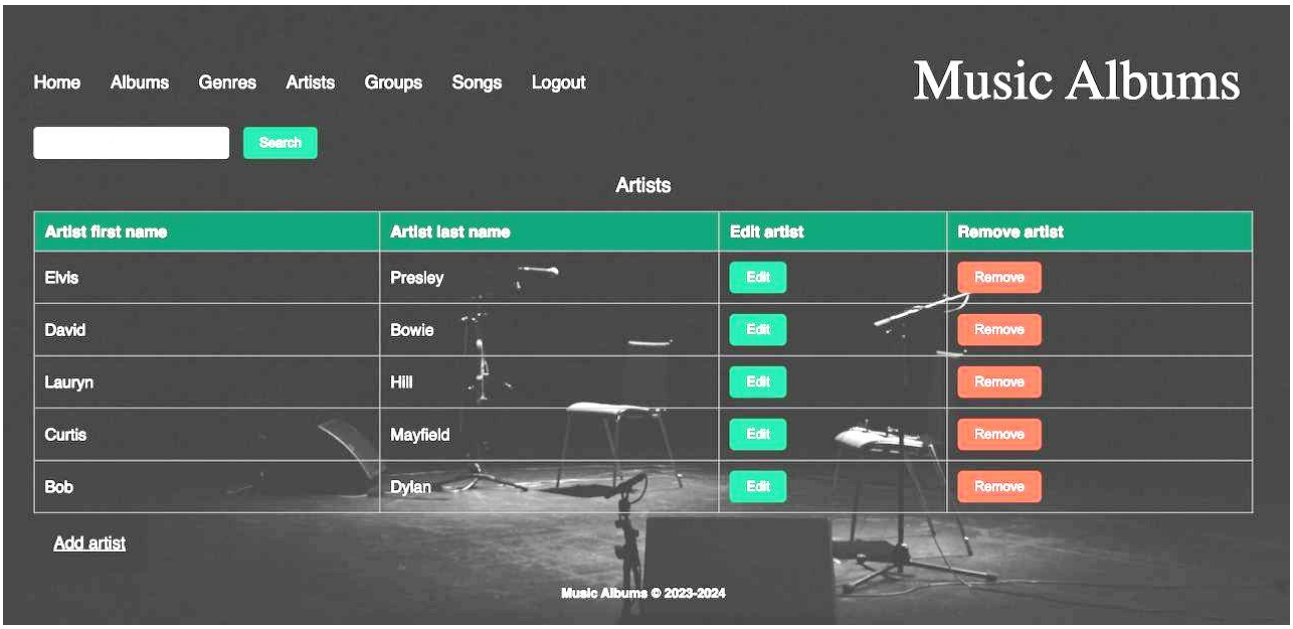

Рисунок *19* – *Довідник артистів перед видаленням*

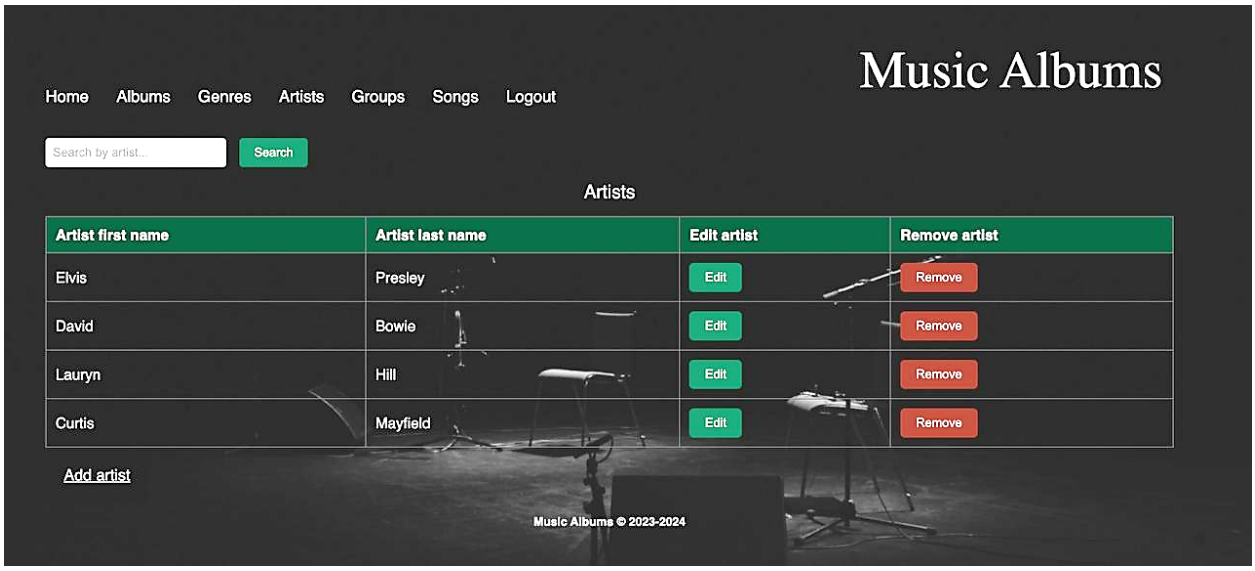

Рисунок *20 – Довідник артистів після видалення*

ВИСНОВКИ

1. Вивчені засоби фреймворку Spring для створення Веб-застосунків що зберігають інформацію у базах даних.
2. Виконані проєктування та розробка Веб-застосунку «Музичні альбоми», який дозволяє користувачам керувати даними колекцій музичних альбомів.
3. Використані засоби фреймворку Spring, які спрощують розробку Веб-застосунків, а також засоби шаблонного двигуна Thymeleaf для створення динамічного Веб-інтерфейсу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Walls, C. Spring in Action, Sixth Edition. Manning Publications Co, 2022. 520 с.
2. Spilca, L. Spring Start Here. Manning Publications Co, 2021. 731 с.
3. Hoeller, J. Spring Framework Reference Documentation. URL: https://docs.spring.io/spring-framework/docs/current/reference/html/ (дата звернення 20.04.2024).
4. Hartnett J. Discogs.com. The Charleston Advisor, V. 16, Number 4, Annual Reviews, 2015, p. 26-33: https://annurev.publisher.ingentaconnect.com/content/annurev/tca/2015/00000016/00000004/art00008 (дата звернення 05.05.2024).
5. Discogs: About Us. URL: https://www.discogs.com/about (дата звернення 07.05.2024).
6. Rate Your Music: Wiki. URL: https://rateyourmusic.com/wiki/ (дата звернення 07.05.2024).
7. MusicBrainz. URL: https://simple.wikipedia.org/wiki/MusicBrainz#:~:text=MusicBrainz (дата звернення 07.05.2024).
8. MusicBrainz: The Open Music Encyclopedia. URL: https://musicbrainz.org/ (дата звернення 07.05.2024).
9. 10 Best Java Frameworks for Web Development in 2024 URL: https://www.aimprosoft.com/blog/java-framework-for-web-development/ (дата звернення 09.05.2024).
10. Spring Framework Projects. URL: https://spring.io/projects/spring-framework (дата звернення 04.04.2024).
11. Oracle Database: What is Database?. URL: https://www.oracle.com/ua/database/what-is-database/ (дата звернення 20.04.2024).
12. HikariCP. HikariCP Documentation. URL: https://github.com/brettwooldridge/HikariCP (дата звернення 20.04.2024).

13. Ноздрюхіна О.С., Коломоєць Г.П. Дослідження способів додання бінів до контексту Spring. Збірник наукових праць студентів, аспірантів, докторантів і молодих вчених «Молода наука-2024» / Запорізький національний університет. Запоріжжя : ЗНУ, 2024. Т.5. С. 124

Декларація
академічної доброчесності
здобувача ступеня вищої освіти ЗНУ

Я, Ноздрюхіна Ольга Сергіївна, студентка 4 курсу, форми навчання заочної, Інженерного навчально-наукового інституту ім. Ю.М. Потебні ЗНУ, спеціальність 121 Інженерія програмного забезпечення освітньої програми "Програмне забезпечення систем", адреса електронної пошти ipz20bd-212@stu.zsea.edu.u, — підтверджую, що написана мною кваліфікаційна робота на тему «**Розробка Веб-застосунку засобами фреймворку Spring**» відповідає вимогам академічної доброчесності та не містить порушень, що визначені у ст.42 Закону України «Про освіту», зі змістом яких ознайомлений.

- заявляю, що надана мною для перевірки електронна версія роботи є ідентичною її друкованій версії;

- згодна на перевірку моєї роботи на відповідність критеріям академічної доброчесності у будь-який спосіб, у тому числі за допомогою інтернет-системи, а також на архівування моєї роботи в базі даних цієї системи.

Дата 27.05.2024 _____ Ноздрюхіна Ольга Сергіївна (студентка)
(підпис) (прізвище та ініціали)

Дата 27.05.2024 _____ Коломоєць Геннадій Павлович (керівник)
(підпис) (прізвище та ініціали)