Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра мультимедійних систем факультету інформатики

Створення Java-вебзастосунку із використанням Spring Boot

Текстова частина до курсової роботи

за спеціальністю „Інженерія програмного забезпечення” 121

Керівник курсової роботи
с.в. _Борозенний С.О.__
(прізвище та ініціали)
_____
(підпис)
“____” _____ 2022 р.

Виконав студент _____
___Кузнець І. І._____
(прізвище та ініціали)
“____” ___ _____ 2022 р.

Київ 2022

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра мультимедійних систем факультету інформатики

ЗАТВЕРДЖУЮ
Зав.кафедри мультимедійних систем,
доцент, к.ф-м.н.
_____О. П. Жежерун (підпис)
„____”_____2022 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на курсову роботу

студенту Кузнецю Іллі Ігоровичу факультету інформатики 3-го курсу

ТЕМА Створення Java-вебзастосунку із використанням Spring Boot

Зміст ТЧ до курсової роботи:

Індивідуальне завдання
Вступ
1 Архітектура застосунку
2 Використані технології та бібліотеки
3 Інструкція користувача, опис програми
Висновки
Список використаних джерел
Додатки (за необхідністю)

Дата видачі „___” _____ 2022 р. Борозенний О.С. _____ (підпис)

Завдання отримав _____ (підпис)

Тема: Створення Java-вебзастосунку із використанням Spring Boot

Календарний план виконання роботи:

№ п/п	Назва етапу виконання курсової роботи	Термін виконання етапу	Примітка
1	Отримання завдання на курсову роботу	26.10.2021	
2	Аналіз матеріалів за темою та розробка ідеї	Січень - Лютий 2022	
3	Розробка застосунку	Березень – Травень 2022	
4	Написання текстової частини до курсової роботи	02.06.2022 - 06.06.2022	
5	Коригування застосунку	02.06.2022 - 06.06.2022	
6	Захист курсової роботи	07.06.2022	

Кузнець І.І. _____

Борозенний О. С. _____

“_____” _____

Зміст

Анотація

Робота ставить за мету вивчення методів розробки Java-вебзастосунків із використанням фреймворку Spring Boot. Під час її виконання був створений сайт для поширення музики між користувачами, який реалізує ці методи.

Назва застосунку – “Dance Center”.

Вступ

Проблема

Досить поширеною незручністю для учнів, що займаються у танцювальних студіях, є постійна необхідність запитувати тренера про музичні композиції, які використовувалися на заняттях. Останнім доводиться надсилати аудіотреки через месенджери, що не дуже зручно. Саме тому було прийнято рішення розробки вебзастосунку, який міг би зробити обмін музикою більш комфортним для танцівників і не тільки.

Завдання

Основним завданням є використати фреймворк Spring Boot для створення вебсайту, на якому зареєстровані особи зможуть робити публікації із музичними композиціями та відстежувати інших користувачів, переглядати їхні публікації та зручно шукати треки та виконавців.

Основна частина

Розділ 1 : Архітектура застосунку

1.1. Архітектурний підхід

Model-View-Controller (MVC) – архітектурний підхід до проєктування та розробки програмного забезпечення, використаний для цього застосунку. Він зазвичай будується навколо бази даних та дає змогу розділити логіку програми на три незалежні рівні:

- **Model** – бізнес-логіка та взаємодія із базою даних.
- **View** – представлення інформації із моделі в користувацькому інтерфейсі.
- **Controller** – обробка та передавання вхідних даних із представлення до моделі.

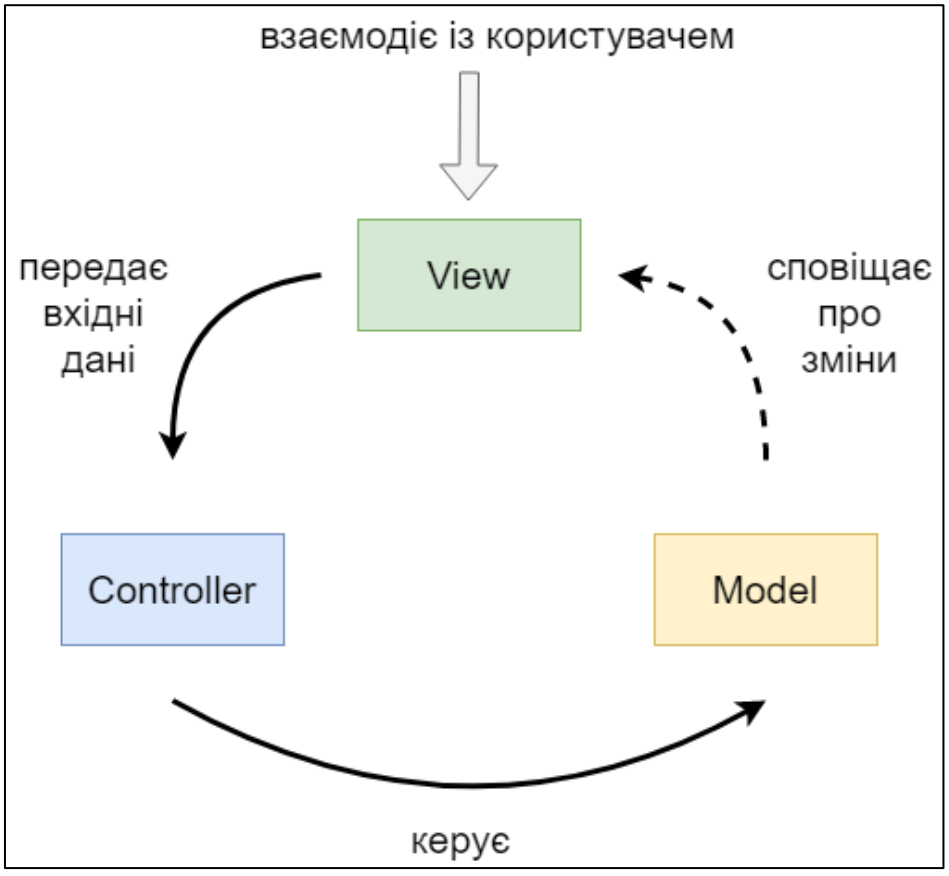

Рис. 1 Архітектурний підхід MVC

1.2. Модель

1.2.1. Сутності та репозиторії

Модель застосунку будується на таких сутностях:

- `AppUser` – користувач застосунку, містить ім'я, пароль, список ролей, список його публікацій, відстежуваних користувачів та тих, хто відстежують його.
- `Role` – роль користувача, містить назву.
- `Artist` – виконавець, містить ім'я та список пісень, які виконує (сам або із іншими виконавцями).
- `Song` – музична композиція, містить назву, покликання на аудіофайл, список виконавців та список публікацій із цією піснею.
- `Post` – публікація, містить текстовий опис, користувача, якому вона належить, дату створення та список її пісень.

Для кожної моделі існує свій репозиторій для збереження та зміни інформації у базі даних .

1.2.2. Сервіси

Було створено інтерфейси сервісів та класи-імплементації для них, оскільки це дозволить легко масштабувати програму у подальшому. Сервіси виконують валідацію та маніпулюють даним, а потім використовують методи репозиторіїв для збереження інформації у базі даних.

Для сутностей існують такі сервіси :

- `Artist` – інтерфейс та імплементація (`ArtistServiceInterface`, `ArtistService`).
- `Song` – інтерфейс та імплементація (`SongServiceInterface`, `SongService`).
- `Post` – інтерфейс та імплементація (`PostServiceInterface`, `PostService`).

- `AppUser` – інтерфейс та імплементація (`UserServiceInterface`, `UserService`), також містить у собі репозиторій та методи для ролі, оскільки користувач завжди йде разом із роллю. Окрім цього існує імплементація UserDetailsService (`UserDetailsServiceImpl`) для автентифікації.

1.3. Представлення

Для користувацького інтерфейсу було створено 13 HTML-шаблонів, які шаблонізатор використовує для генерації HTML-сторінок разом із даними моделі.

1.4. Контролер

Рівень представлення використовує методи контролера для передачі даних на сервер, далі у сервіси. Перед цим він форматує вхідну інформацію, проводить необхідну на цьому рівні валідацію, та обробляє помилки і повертає користувачеві сторінки із відповідними повідомленнями.

Програма містить такі контролери:

- `ArtistController`, `SongController`, `PostController`, `UserController` – для обробки та передавання вхідних даних, що стосуються відповідно виконавців, композицій, публікацій та користувачів.
- `MainController` – для обробки та передавання вхідних даних, що не стосуються будь-яких сервісів, або стосуються усіх.
- `AuthenticationController` – для обробки реєстрації та логіну.
- `MyErrorController` – для обробки помилок та виведення відповідних сторінок.

1.5. Конфігурації

Програма використовує класи конфігурації, які задають налаштування обробки ресурсів (`WebMvcConfig`) та автентифікації і авторизації (`WebSecurityConfig`).

1.6. Допоміжні класи

Також у програмі використовуються 2 незалежні класи, які виконують допоміжні функції або містять у собі статичні змінні, потрібні для різних частин програми: `Utils` та `Values` відповідно.

Розділ 2: Використані технології та бібліотеки

2.1. Spring Boot

- `Spring Boot Starter Data JPA` – стартер, який дозволяє використовувати Hibernate як провайдера для специфікацій Spring Data JPA (фреймворк, що працює із JPA та дає змогу легко створювати репозиторії, основані на JPA) .

- `Spring Boot Starter Web` – стартер для створення вебзастосунків (у тому числі RESTful) із використанням Spring MVC (фреймворк, який підтримує архітектурний підхід MVC та надає готові компоненти для роботи із ним).

- `Spring Boot Starter Validation` – стартер для використання Java Bean Validation із Hibernate валідатором.

- `Spring Boot Starter Security` – стартер, який агрегує залежності, необхідні для роботи Spring Security (фреймворк для автентифікації та авторизації).

- `Spring Boot Starter Thymeleaf` – стартер, який дозволяє використовувати шаблонізатор Thymeleaf для створення рівня представлення у MVC вебзастосунках.

- `Spring Boot Devtools` – модуль, що дозволяє автоматично перезапускати застосунок при змінюванні його файлів.

2.2. Інше

- PostgreSQL – використана система керування базами даних.
- JSON in Java (org.json) – бібліотека для обробки та парсингу JSON-даних.
- Lombok – бібліотека, яка дозволяє зменшити кількість шаблонного коду у програмі (гетери, сетери, конструктори і т.д.), замінивши його анотаціями.
- Jaudiotagger – бібліотека, яка надає можливість отримати аудіо теги із файлів (назва композиції, виконавці, альбом і т.д.).
- Bootstrap 5 – фреймворк для створення HTML-сторінок.
- JavaScript – використаний для валідації і відправлення запитів з клієнтської частини програми та для керуванням вмістом HTML-елементів.

Розділ 3 : Інструкція користувача, опис програми

3.1. Автентифікація, авторизація

При вході у застосунок користувачеві буде запропоновано увійти до свого акаунту або створити новий. Доступ до сторінок входу, реєстрації та помилок мають усі особи, що намагаються увійти на сайт, натомість інші сторінки можуть відвідати лише користувачі, що пройшли автентифікацію.

При успішній спробі входу особу буде перенаправлено на сторінку “Feed”, де знаходитимуться пости користувачів, на які вона підписана, а при невдалій буде виведено повідомлення про помилку. Також при автентифікації дістається роль користувача, яка впливатиме на те, які вкладки відображатимуться йому у меню навігації.

При виході з акаунту інвалідується HTTP сесія та очищується чинна автентифікація із `SecurityContext`.

Наведені вище функції були реалізовані за допомогою Spring Security, а саме імплементацій `WebSecurityConfigurerAdapter` (дозволяє створити особисту конфігурацію безпеки) та `UserDetailsService` (дозволяє отримати інформацію про користувача).

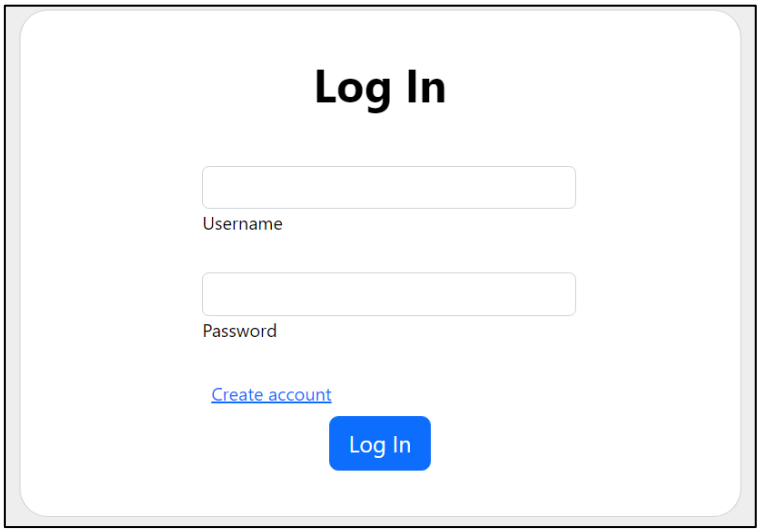

Рис. 2 Сторінка входу

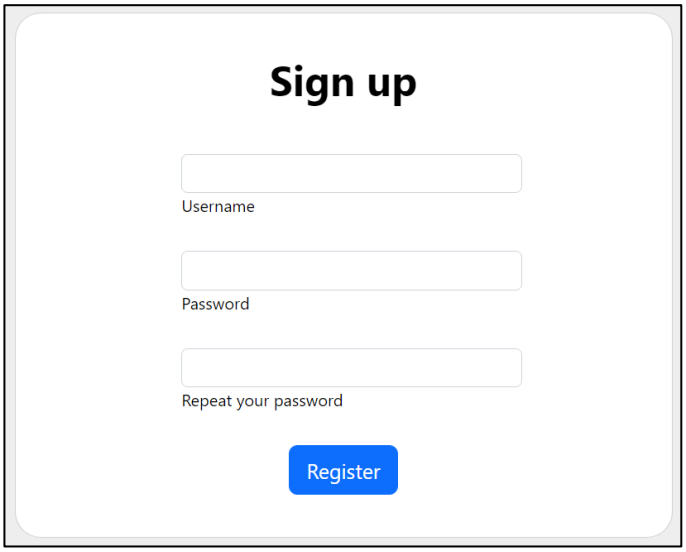

Рис. 3 Сторінка реєстрації

3.2. Меню навігації

Для меню навігації було створено HTML-шаблон, який містить `Thymeleaf Fragment`. Це дозволить без дуплікації коду вміщувати елемент “header” у кожну сторінку.

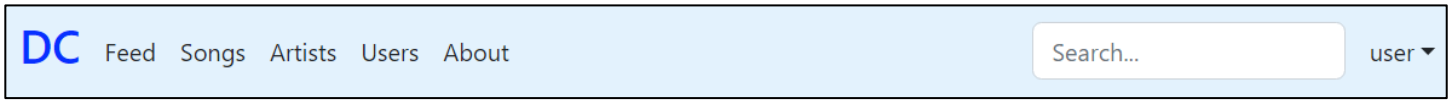

Рис. 4 Меню навігації користувача

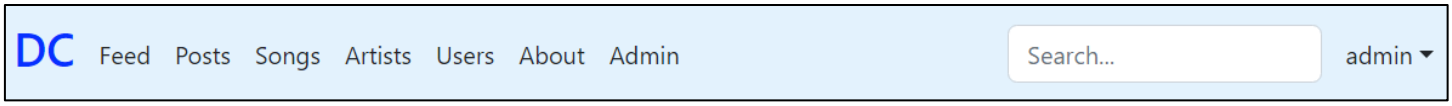

Рис. 5 Меню навігації адміністратора

Перед відтворенням будь-якої сторінки у методі контролера до моделі додається атрибут “currentUser”, що містить у собі об’єкт чинного автентифікованого користувача для цієї сесії та атрибут “isAdmin”, що вказує, чи має користувач роль адміністратора. У залежності від заданих атрибутів буде відображатися різна кількість вкладок у меню навігації (лише адміністратор може бачити публікації усіх користувачів та сторінку “Admin”) та ім’я акаунту у правому верхньому куті.

Меню навігації також містить поле для пошуку за текстом у назвах пісень або іменах користувачів чи виконавців.

При натисненні на ім’я користувача з’явиться список, що випадає, із кнопками створення нової публікації, переходу на свій профіль та виходу із акаунту.

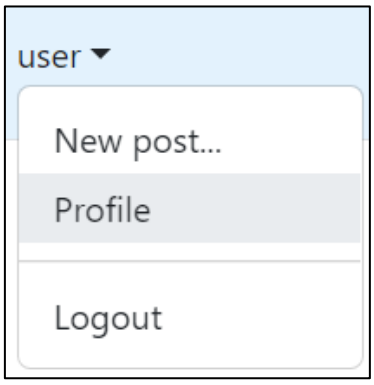

Рис. 6 Список, що випадає із меню навігації

3.3. Сторінка пісень

На сторінці “Songs” користувач може переглянути усі наявні у застосунку композиції. Для кожної пісні показується її назва (чорним кольором) та перелік виконавців (синім), якщо вони є.

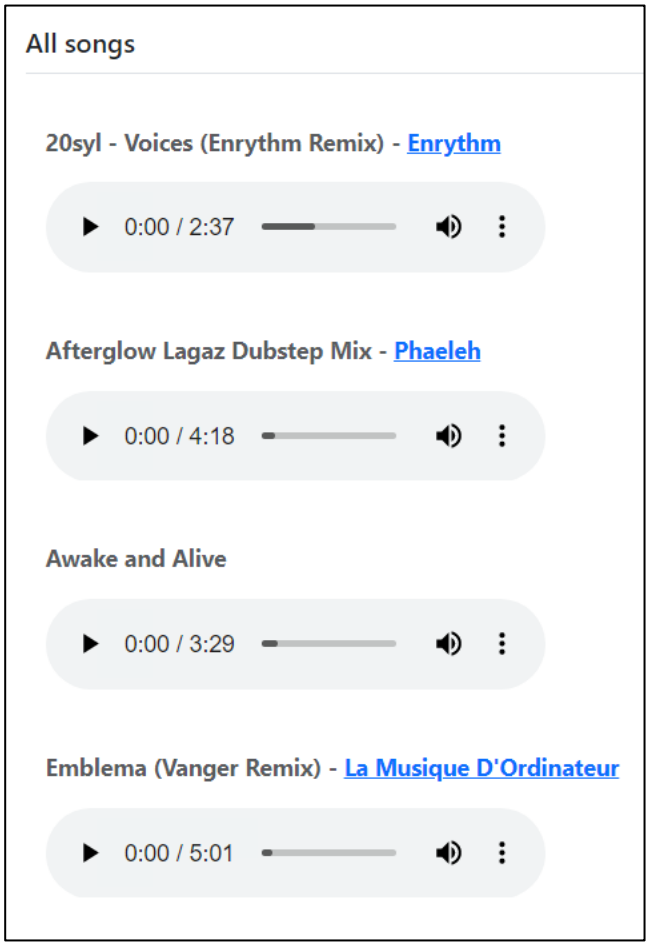

Рис. 7 Сторінка із усіма композиціями

Метод, що відповідає за адресу, за якою можна доступитися до сторінки пісень, перед поверненням відповідного представлення додає до моделі усі композиції із `SongService`, роблячи виклик функції `getAll()`.

```
@Override
public List<Song> getAll() {
    return songRepository.findAll(Sort.by("title"));
}
```

Рис. 8 Метод для отримання усіх пісень у SongService

Репозиторій пісень наслідує `JpaRepository`, що дає можливість робити сортування та використовувати багато функцій, що можуть бути корисними при подальшій розробці проєкту. У цьому застосунку відбувається сортування пісень за назвою, що є атрибутом “title” у базі даних.

3.4. Сторінка виконавців

Сторінка “Artists” містить усіх виконавців, наявних у застосунку. Для кожного виконавця відображається перелік усіх його пісень (рис. 9).

При натисканні на виконавця (ім’я у заголовку відповідного блоку або у стрічці із піснею) буде здійснено перехід на його окрему сторінку, що міститиме лише його композиції (рис. 10).

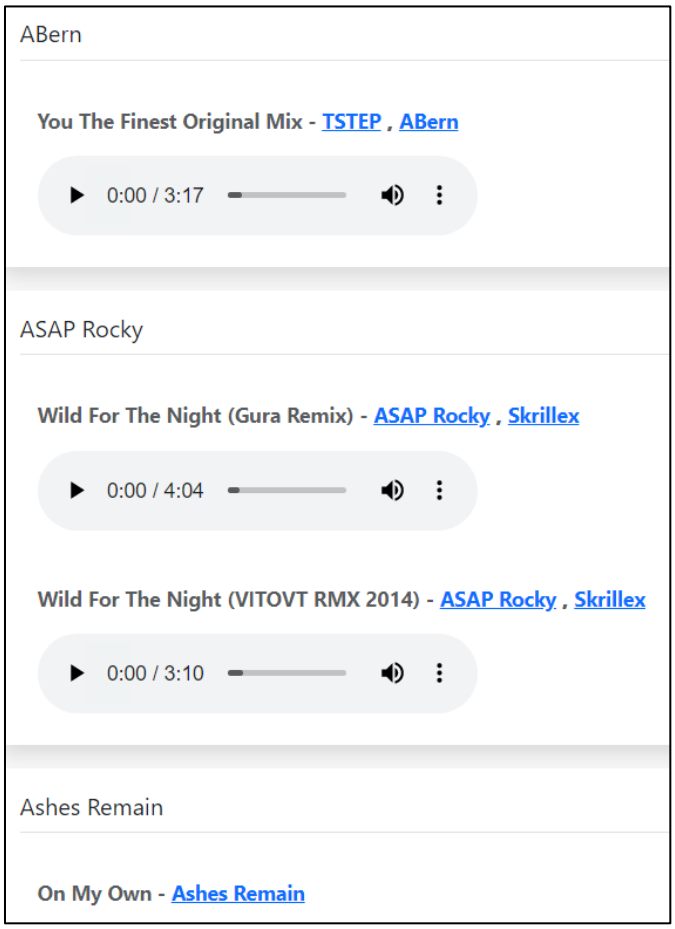

Рис. 9 Сторінка усіх виконавців

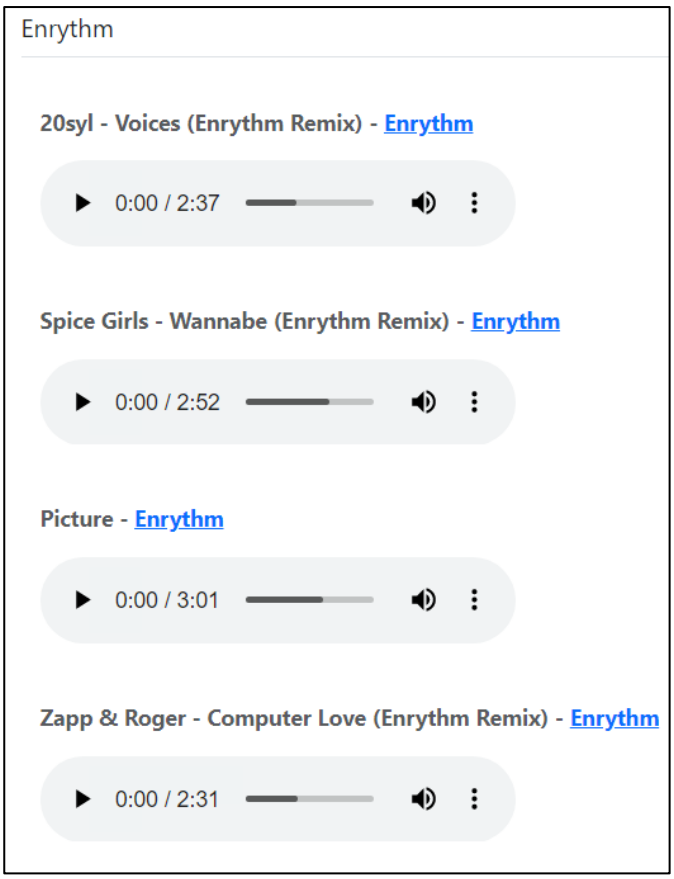

Рис. 10 Сторінка окремого виконавця

Завантаження виконавців до моделі та їхнє сортування відбувається аналогічно до пісень, за допомогою сервісу та репозиторію відповідно.

3.5. Сторінка публікацій

Сторінка “Feed” містить пости усіх користувачів, яких відстежує нинішній, відсортованих за датою створення так, що найновіші відображаються спочатку (ніби стрічка новин або постів у Твіттері чи Інстаграмі).

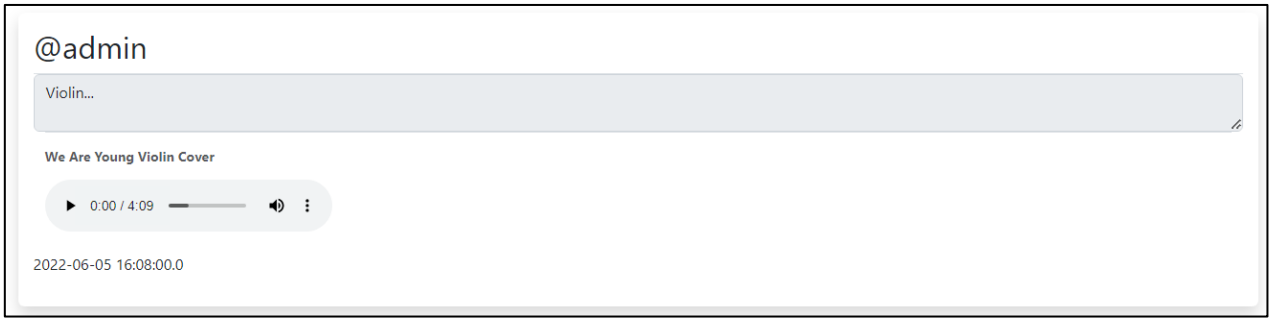

Рис. 11 Вигляд публікації

Кожна публікація містить у собі ім’я користувача у заголовку, якому передує символ ‘@’ для легшого розрізнення від іншого тексту. Під ім’ям розміщений опис публікації, усі її композиції та дата і час створення.

Покликання на сторінки виконавців присутні у кожній пісні, де вони згадуються, проте у наведеному прикладі (рис. 11) композиція не має жодного виконавця (оскільки користувач не завжди володіє повною інформацією про файл).

На відміну від пісень та виконавців, окрім сортування усіх наявних у базі даних відповідних об’єктів, існує ще сортування певної обраної групи публікацій (метод `sortPosts()` у `PostService`). Це дозволяє виводити до інтерфейсу відсортовані публікації окремого користувача (чи групи користувачів) на цій та інших сторінках.

```
public List<Post> sortPosts(List<Post> posts) {
    return postRepository.findByIdIn(posts.stream().map(Post::getId).collect(Collectors.toList()),
            Sort.by(Sort.Direction.DESC, ...properties: "date"));
}
```

Рис. 12 Метод для сортування публікацій у PostService

При натисканні на ім’я користувача, якому належить публікація, буде здійснено перехід на сторінку із його профілем.

3.6. Сторінка користувачів

Ця сторінка дозволяє переглядати усіх користувачів застосунку у короткому вигляді.

Рис. 13 Нинішній користувач

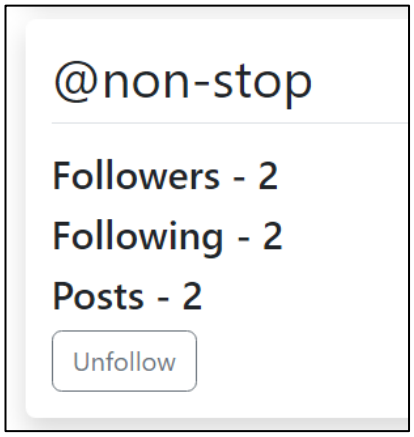

Рис. 14 Користувач, відстежуваний нинішнім

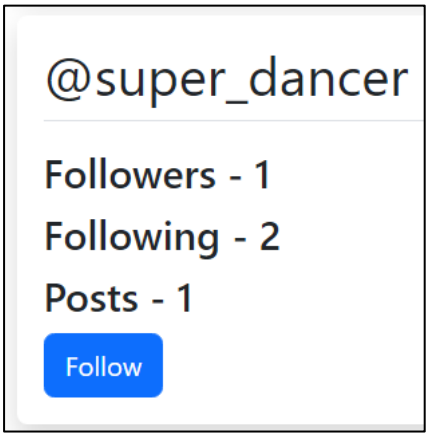

Рис. 15 Користувач, не відстежуваний нинішнім

Для кожного користувача вказана кількість людей, які його відстежують (Followers), яких відстежує він (Following) та кількість його публікацій (Posts). Кожен з цих написів є покликанням на сторінки, що містять відповідних користувачів або відповідні публікації.

Під цими покликаннями міститься кнопка, яка дозволяє відстежувати або не відстежувати певного користувача (але не себе, що очевидно, тому у блоці із @dubstep_fan (рис. 13) немає такої кнопки).

При натисканні на ім’я будь-якого користувача відбудеться перехід на сторінку його профілю.

3.7. Профіль користувача

Профіль користувача у повному вигляді містить усі складові короткого вигляду разом із публікаціями цього користувача, відсортованими за датою (на прикладі (рис. 16) @admin має лише одну публікацію). Усі покликання із короткого вигляду користувача також працюють.

Рис. 16 Профіль ненинішнього користувача

Якщо перейти на сторінку свого профілю, користувач не бачитиме кнопки відстежувати/не відстежувати, проте з’явиться інша – кнопка видалення публікації, що знаходиться у її правому нижньому куті.

Рис. 17 Профіль нинішнього користувача

3.8. Додавання публікації

3.8.1. Загальний опис

На рис. 6 було показано меню користувача, що випадає з меню навігації. Воно містить кнопку “New post…”, при натисканні на яку відбудеться перехід на сторінку створення публікації.

Рис. 18 Незаповнена сторінка додавання публікації

При створенні публікації користувач може ввести опис у поле “Description”. Розмір цього поля буде збільшуватись разом із кількістю введеного тексту за допомогою JavaScript. При додаванні нових композицій (після чого сторінка перезавантажується) текст опису не змінюється, оскільки передається через URL-змінну у метод контролера.

Для того, щоб додати нові композиції до публікації, необхідно натиснути на кнопку “Choose files” та обрати бажані аудіофайли. Тоді завантажені файли будуть зберігатися у значенні HTML-елементу input (далі `loadFileInput`).

Унизу блоку знаходиться кнопка додавання публікації “Add” та кнопка скидання введених даних “Reset”, яка очищує опис та видаляє завантажені на сервер композиції, що мали міститися у цій публікації.

3.8.2. Послідовність процесів

Для `loadFileInput` існує обробник події onChange у JavaScript файлі. Ця функція є асинхронною, оскільки подальші запити до серверу, що можуть зайняти деякий час, не повинні негативно впливати на процеси, що відбуваються у інтерфейсі, наприклад.

Коли користувач змінює вміст `loadFileInput`, на сервер відправляється запит на отримання тегових даних про обрані аудіофайли. Для цього використовується метод `getMultipleFileInfo()` із `SongController`, який викликається при відправленні POST-запиту на адресу "/songs/multipleFileInfo". Цей запит відправляється за допомогою виклику функції `fetch` із цією адресою у JavaScript, яка в разі помилки перенаправляє користувача на сторінку помилки отримання інформації про файли.

Для того, щоб надіслати будь-які файли на сервер, необхідно використати певний тип кодування тіла POST-запиту – “multipart/form-data”. Для згаданого вище запиту такий тип кодування не вказаний, оскільки сучасні браузери встановлюють його самостійно. Проте для того, щоб це відбулося, потрібно передати у тіло запиту об’єкт типу `FormData`.

Отже, при зміні `loadFileInput` створюється об’єкт `FormData`, у який додається багатозначний атрибут “files”, кожне значення якого містить у собі один аудіофайл. Це означає, що цей атрибут буде доданий до `FormData` стільки разів, скільки було завантажено файлів у `loadFileInput`.

При прийомі POST-запиту на "/songs/multipleFileInfo" сервер отримує список `MultipartFile`, які є надісланими користувачем аудіофайлами. Тоді для кожного файлу за допомогою методу `getSingleFileInfo()` із `SongService` дістаються тегові дані та додаються у список-результат.

Процес отримання цих даних складається із декількох частин. Спочатку із переданого як аргумент `MultipartFile` створюється звичайний Java `File`. Далі із цього файлу зчитується `AudioFile` із бібліотеки `Jauditagger`. Саме він міститиме у собі необхідні тегові дані.

У кінці процесу створюється HashMap, що міститиме пари назва атрибуту тегу – значення атрибуту. Ця мапа і повертається як результат методу.

Після успішного отримання інформації про завантажені у `loadFileInput` файли до минулої `FormData` додаються два атрибути: “titles” – список назв пісень та “artists” – список списків виконавців пісень (оскільки кожна пісня містить список виконавців). Значення цих атрибутів передаються, як два JSON-об’єкти.

Після цього на сервер відправляється новий POST-запит на адресу “/songs/multipleAdd”. Метод контролера `addMultipleSongs()` приймає передані через `FormData` списки назв, виконавців і файлів. Далі за допомогою бібліотеки `JSON in Java(org.json)` із вхідних аргументів парсяться Java-списки, елементи яких передаються для створення об’єктів композицій.

При додаванні композиції до репозиторію створюється Java `File` та зберігається у локальній файловій системі (при подальшому розвитку проєкту необхідно буде перейти до зберігання файлів на FTP-сервері чи в хмарі, наприклад). Об’єкт Song містить у собі поле “location”, що є шляхом до файлу у застосунку.

Кожен “location” розпочинається із префіксу “files/”. Тоді за допомогою `WebMvcConfig` (імплементує `WebMvcConfigurer`) можна визначити `resourceHandler`, який вказуватиме серверу, що за адресою “files/**” (усі адреси, що є вкладеними у “files/”) потрібно дістати файл із певної локації.

```
@Override
public void addResourceHandlers(ResourceHandlerRegistry registry) {
    registry.addResourceHandler( ...pathPatterns: "/files/**")
            .addResourceLocations("file://" + Values.UPLOAD_PATH + "/");
    registry.addResourceHandler( ...pathPatterns: "/js/**")
            .addResourceLocations("classpath:/js/");
    registry.addResourceHandler( ...pathPatterns: "/images/**")
            .addResourceLocations("classpath:/images/");
}
```

Рис. 19 Обробник ресурсів у WebMvcConfig

Стрічка “file://” вказує, що має використатись файловий протокол, а ще один ‘/’ у `Values.UPLOAD_PATH` - що шлях стане відносним до кореня нинішнього диску комп’ютера (із якого запускається застосунок).

Стрічка “classpath:” вказує, що шлях буде відносним до директорії “src/main/resources”, оскільки classpath має значення за замовчуванням.

Результатом додавання пісень до бази даних є список їхніх ID. Цей список перетворюється у стрічку та додається до URL-адреси, за якою буде здійснено перенаправлення. Тоді отримавши GET-запит за цією адресою, метод контролера прийме список ID доданих пісень та відобразить їх на сторінці додавання публікації.

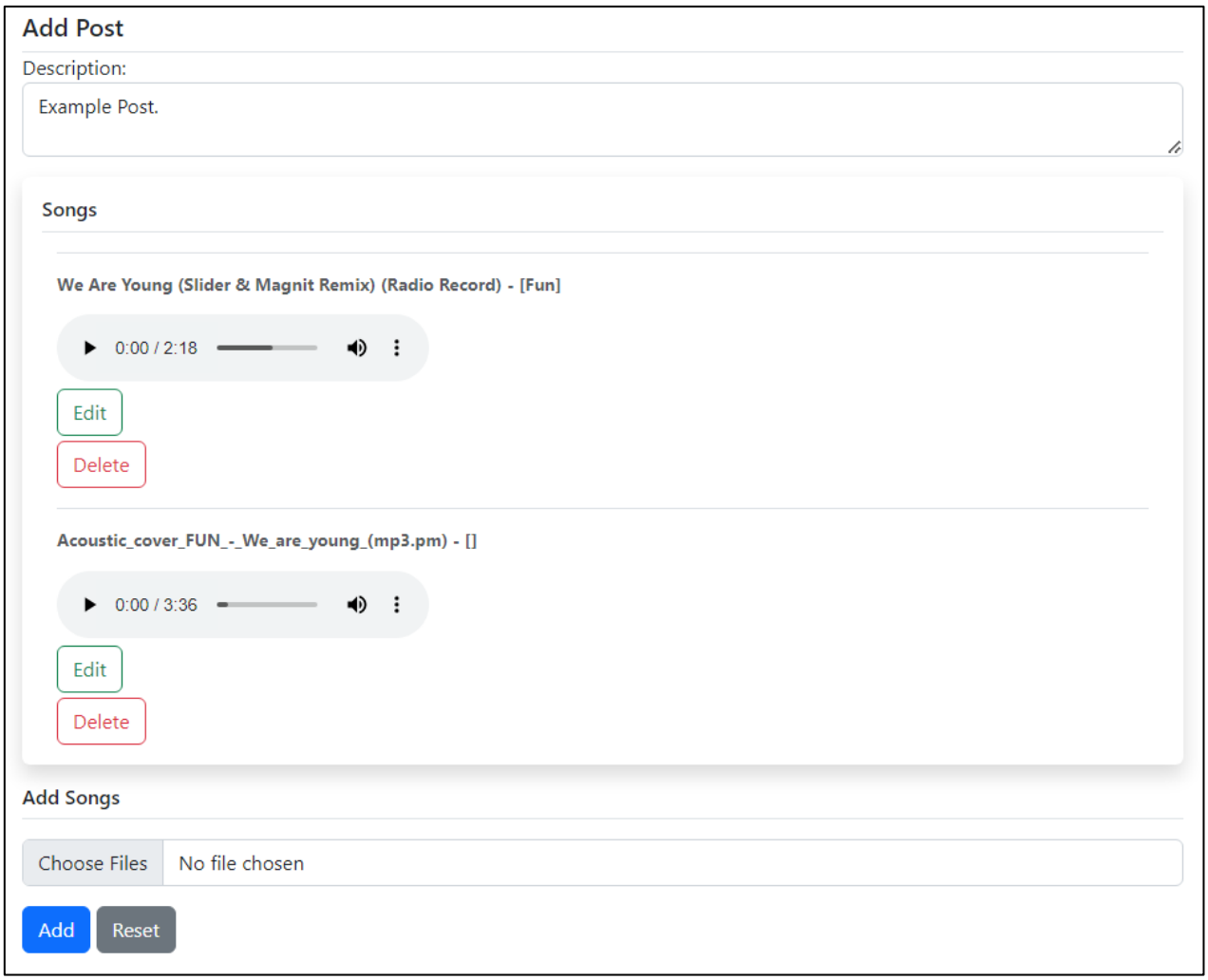

Рис. 20 Заповнена сторінка додавання публікації

Додавши якісь композиції, користувач зможе редагувати їх або видалити. Якщо пісня має зайвий текст у назві або її аудіофайл не містить тегових даних, а отже і виконавці не будуть додані до неї, то можна додати їх самостійно у модальному вікні редагування композиції.

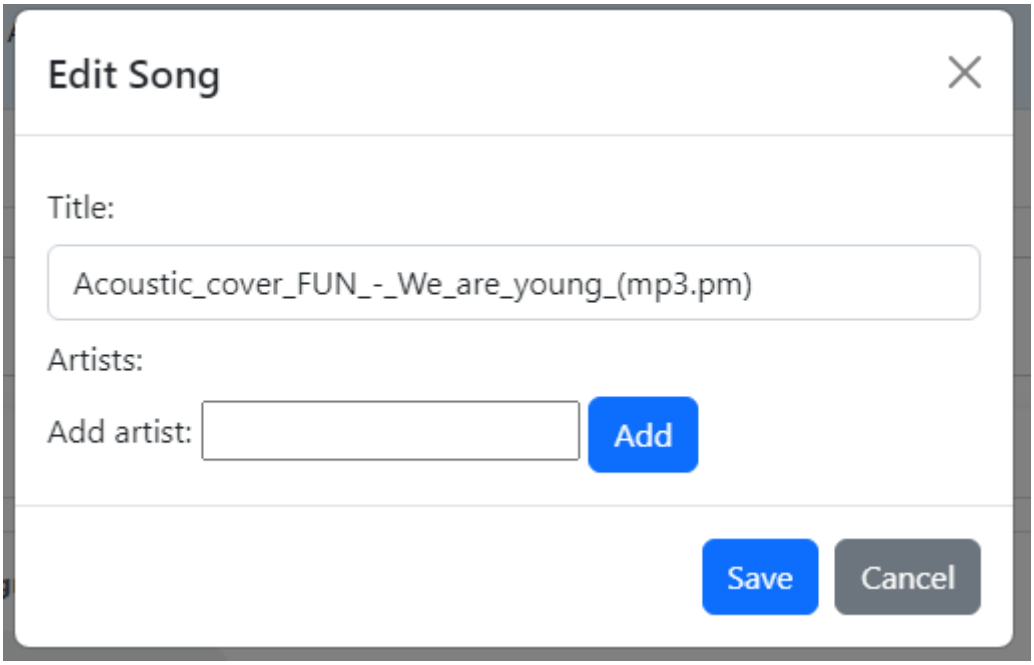

Рис. 21 Незмінена композиція

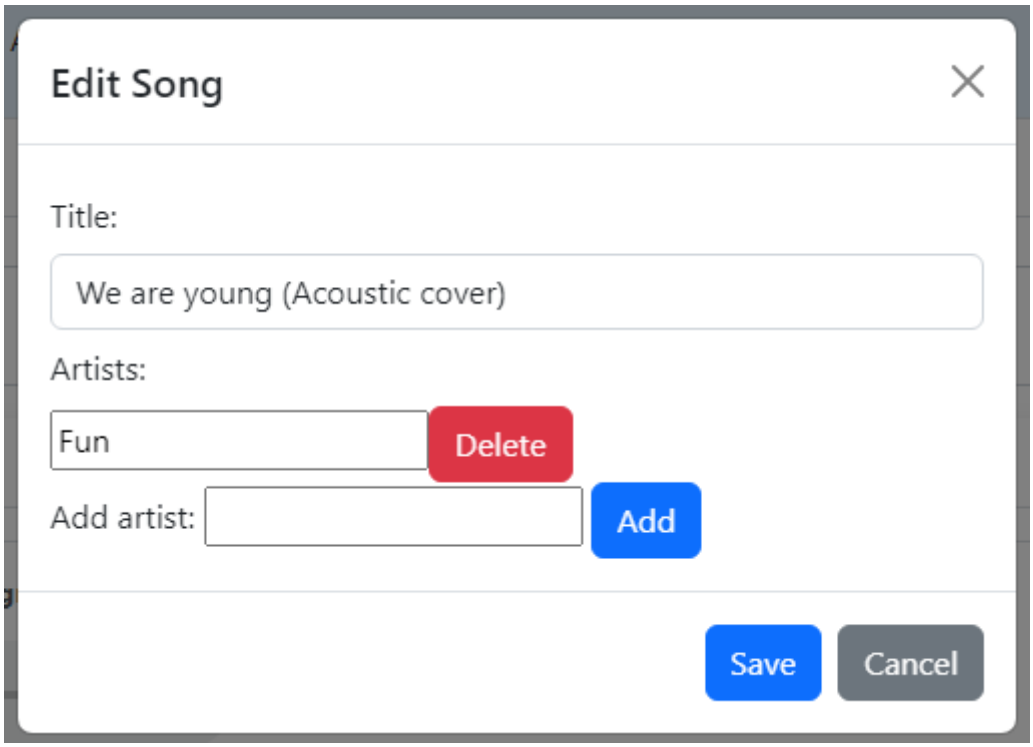

Рис. 22 Змінена композиція

Після натискання на кнопку “Save” буде надіслано POST-запит із новими значеннями назви і авторів та пісня буде оновлена у базі даних.

Після натискання на кнопку “Add” на сторінці додавання публікації буде надіслано POST-запит із значеннями опису та ID пісень, що містяться у ній, після чого публікація буде додана до бази даних. Далі відбудеться перенаправлення на сторінку із профілем нинішнього користувача.

3.9. Пошук по сайту

У меню навігації на кожній сторінці доступне поле для пошуку за текстом у назвах пісень або іменах користувачів чи виконавців.

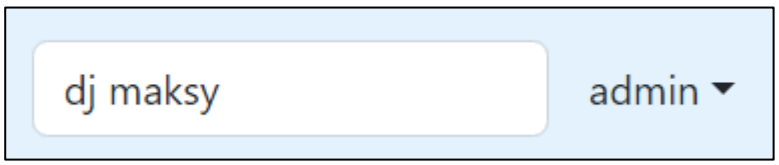

Рис. 23 Поле для пошуку із введеним запитом

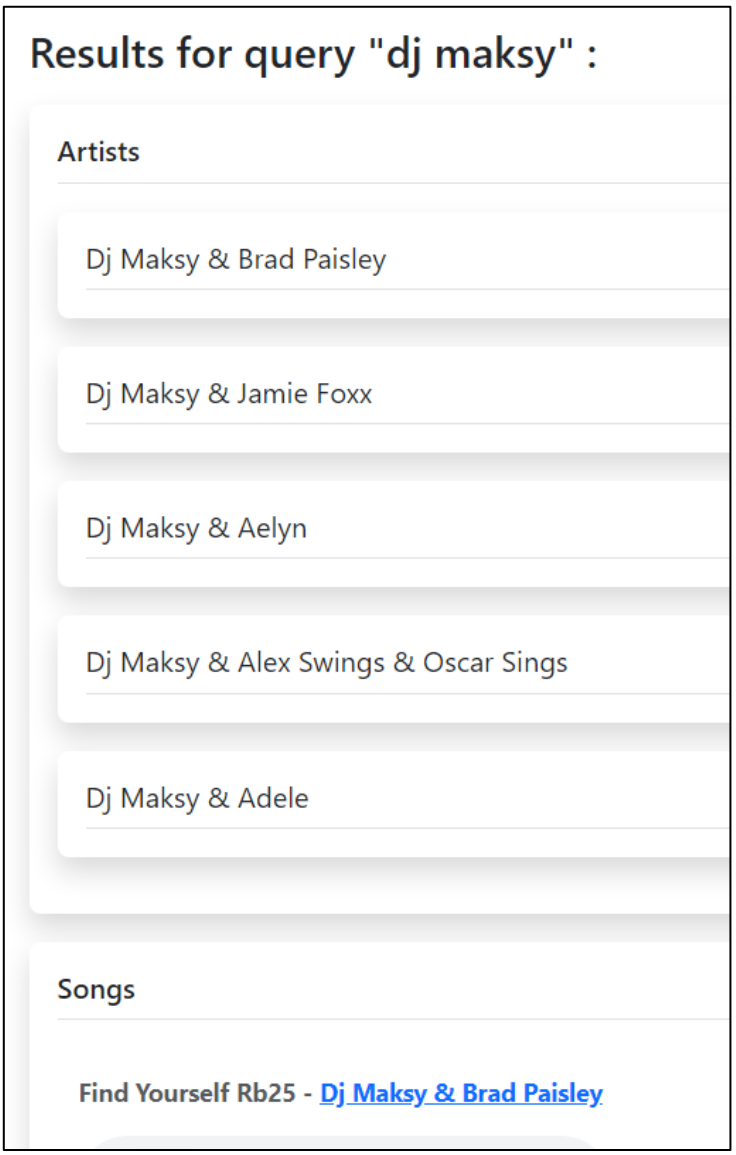

Рис. 24 Результат пошуку

У результаті пошуку буде відображено усіх користувачів та виконавців, що містять текст запиту, а також пісні, що містять цей текст у назві чи виконавцях. Натиснувши на виконавця, можна перейти на його сторінку.

Для запиту “dj maksy” не існує користувача, ім’я якого містило б цю стрічку. Перелік пісень за цим запитом складається із п’яти відповідних елементів, проте вони не поміщаються на скріншоті.

3.10. Можливості адміністратора

Адміністратор має можливість перейти на сторінку “Posts”, що міститиме усі публікації у застосунку, а також на сторінку “Admin”.

Сторінка “Admin” дає можливість користувачам, які мають роль адміністратора, видаляти публікації, композиції та виконавців.

Рис. 25 Таблиця із публікаціями для адміністратора

Таблиці для пісень у публікаціях, пісень загалом та виконавців виглядають аналогічно.

3.11. Інше

На сторінці “About” користувач зможе побачити назву застосунку та його гасло.

При натисканні на логотип сайту у меню навігації користувача буде перенаправлено на сторінку “Feed”.

Висновок

Провівши необхідні дослідження було опановано потрібні навички для створення вебзастосунку, визначеного у завданні. Виконання роботи допомогло покращити розуміння деяких архітектурних аспектів (пов’язаних із MVC), процесів, що відбуваються у Spring Security та при відправленні запитів у вебзастосунках, методів передачі файлових даних до серверу через клієнтський інтерфейс та багато іншого.

Вважаю, що створений застосунок відповідає завданню та має велике значення для вирішення вказаної проблеми.

Список використаних джерел

Рисунки

Рис 1: https://uk.wikipedia.org/wiki/%D0%A4%D0%B0%D0%B9%D0%BB:%D0%94%D1%96%D0%B0%D0%B3%D1%80%D0%B0%D0%BC%D0%B0_%D0%B2%D0%B7%D0%B0%D1%94%D0%BC%D0%BE%D0%B4%D1%96%D1%97_%D0%BC%D1%96%D0%B6_%D0%BA%D0%BE%D0%BC%D0%BF%D0%BE%D0%BD%D0%B5%D0%BD%D1%82%D0%B0%D0%BC%D0%B8_%D1%88%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD%D1%83_MVC.png

Література

1. MVC design pattern [Електронний ресурс]: https://www.geeksforgeeks.org/mvc-design-pattern/
2. Spring – MVC Framework [Електронний ресурс]: https://www.tutorialspoint.com/spring/spring_web_mvc_framework.htm
3. The top 5 software architecture patterns: How to make the right choice – Peter Wayner [Електронний ресурс]: https://techbeacon.com/app-dev-testing/top-5-software-architecture-patterns-how-make-right-choice
4. Spring Boot Starter Data JPA artifact in maven repository [Електронний ресурс]: https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-starter-data-jpa
5. Spring Boot Starter Web artifact in maven repository [Електронний ресурс]: https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-starter-web
6. Spring Boot Starter Validation artifact in maven repository [Електронний ресурс]: https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-starter-validation

7. Spring Boot Starter Thymeleaf artifact in maven repository [Електронний ресурс]: https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-starter-thymeleaf
8. Spring Data JPA documentation [Електронний ресурс]: https://spring.io/projects/spring-data-jpa
9. Getting Spring Security – Spring documentation [Електронний ресурс]: https://docs.spring.io/spring-security/reference/getting-spring-security.html#:~:text=Boot%20with%20Gradle-,Spring%20Boot%20provides%20a%20spring%2Dboot%2Dstarter%2Dsecurity%20starter,%3A%2F%2Fstart.spring.io
10. Spring Security Autoconfiguration – baeldung [Електронний ресурс]: https://www.baeldung.com/spring-boot-security-autoconfiguration
11. Spring Security: Authentication with a Database-backed UserDetailsService - Loredana Crusoveanu [Електронний ресурс]: https://www.baeldung.com/spring-security-authentication-with-a-database#:~:text=The%20UserDetailsService%20interface%20is%20used,about%20the%20user%20during%20authentication
12. Uploading files – Spring guides [Електронний ресурс]: https://spring.io/guides/gs/uploading-files/
13. File Upload with Spring MVC – baeldung [Електронний ресурс]: https://www.baeldung.com/spring-file-upload
14. How Spring Security Authentication works - Java Brains [Електронний відео-ресурс]: https://www.youtube.com/watch?v=caCJAJC41Rk&ab_channel=JavaBrains
15. Spring Boot and Spring Security with JWT including Access and Refresh Tokens – Amigoscode [Електронний відео-ресурс]: https://www.youtube.com/watch?v=VVn9OG9nfH0&t=4232s&ab_channel=Amigoscode
16. Bootstrap 5 – documentation [Електронний ресурс]: https://getbootstrap.com/docs/5.0/getting-started/introduction/

17. Display error messages in Spring login – answer by Yan Pak [Електронний ресурс]: https://stackoverflow.com/questions/13261794/display-error-messages-in-spring-login
18. Creating a textarea with auto-resize – answer by DreamTek [Електронний ресурс]: https://stackoverflow.com/questions/454202/creating-a-textarea-with-auto-resize