

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова
Навчально-науковий інститут комп'ютерних наук та управління проектами
Кафедра інформаційних управляючих систем та технологій

"Допущений до захисту"

Завідувач кафедри ІУСТ

к.т.н, доц. Михелєв І.Л.

" ____ " _____ 2024 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти "бакалавр"

на тему: «Розробка веб-сайту для управління музичним архівом»

Виконала: студентка групи 4142

_____ *Терещенко А. С.*

(підпис)

Керівник роботи:

Старший викладач _____

(посада, науковий ступінь вчене звання)

_____ *Павленко А. Ю.*

(підпис)

м. Миколаїв – 2024 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова
Навчально-науковий інститут комп'ютерних наук та управління проектами
Кафедра інформаційних управляючих систем та технологій
Спеціальність 122 "Комп'ютерні науки"
Освітня програма "Комп'ютерні науки"

"ЗАТВЕРДЖУЮ"

Гарант освітньої програми

к.т.н, доц. Гайда А.Ю.

"____" _____ 2024 р.

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ на здобуття ступеня вищої освіти "бакалавр"

Студентці Терещенко Аллі Сергіївні

(прізвище, ім'я, по батькові)

1. Тема роботи: Розробка веб-сайту для управління музичним архівом

Керівник роботи Павленко Альона Юріївна

Затверджені наказом ректора № 243-уч від "18" березня 2024 року.

2. Термін подання роботи: 20 червня 2024 р.

3. Вихідні дані до проекту (роботи) ДСТУ щодо обробки інформації, літературні джерела, технічна документація на існуючі аналоги систем,

4. Перелік питань, що належать до розробки (найменування розділів)

Аналіз предметної області, постановка задачі, розробка проекту інформаційної підсистеми, реалізація проєкту підсистеми, охорона праці.

5. Перелік презентаційних матеріалів титульна сторінка, мета та актуальність роботи, схема запропонованої концепції, діаграма варіантів використання, логічна модель даних, фізична модель даних, діаграма компонентів, діаграма класів, діаграма переходу станів, діаграма розгортання системи, результати розробки, висновки.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	Павленко А. Ю. старший викладач	05.02.2024	22.02.2024
2	Павленко А. Ю. старший викладач	22.02.2024	21.03.2024
3	Павленко А. Ю. старший викладач	21.03.2024	25.04.2024
4	Павленко А. Ю. старший викладач	25.04.2024	07.06.2024

7. Дата видачі завдання 05 лютого 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів роботи	Примітка
1	Аналіз предметної галузі	05.02.2024	
2	Постановка задачі	29.02.2024	
3	Розробка концепції	21.03.2024	
4	Розробка проекту веб-сайту	11.04.2024	
5	Реалізація проекту	23.05.2024	
6	Розробка питань з охорони праці	05.06.2024	
7	Розробка графічних матеріалів	12.06.2024	
8	Подання роботи на перевірку на плагіат	10.06.2024	
9	Подання роботи рецензенту	15.06.2024	
10	Подання роботи на захист	20.06.2024	

Студент

(підпис)

Терещенко А. С.

(ПІБ)

Керівник роботи

(підпис)

Павленко А. Ю.

(ПІБ)

АННОТАЦІЯ

Тема кваліфікаційної роботи «Розробка веб-сайту для управління музичним архівом».

Актуальність теми полягає в необхідності ефективного організування та зберігання великого обсягу музичного контенту. Користувачі потребують зручних інструментів для швидкого пошуку, сортування та управління музичними колекціями, що забезпечить економію часу і підвищить продуктивність.

Метою кваліфікаційної роботи є підвищення продуктивності роботи музичної установи за рахунок розробки веб-сайту. Веб-сайт забезпечить зручний та ефективний спосіб каталогізації, зберігання та управління інформацією про музичні твори. Також дасть змогу користувачам створення, редагування та організацію музичних колекцій.

Результатом роботи є створений багатофункціональний та зручний в використанні веб-сайт, який задовольнить потреби користувачів в ефективному управлінні музичним архівом. Зможе забезпечити високу продуктивність та стабільність системи.

Готовий програмний продукт буде корисний як для приватних музичних студій, радіостанцій та культурних установ, так і для окремих користувачів, які хочуть організувати особисті музичні колекції.

Кваліфікаційна робота виконана на 79 сторінках, містить 35 рисунків, 7 таблиць та 3 додатки з технічним завданням, текстом програми та інструкцією користувача, також список використаних джерел.

Робота виконана українською мовою.

ABSTRACT

The topic of the qualification work is ‘Development of a website for managing a music archive’.

The relevance of the topic lies in the need to effectively organise and store a large amount of music content. Users need convenient tools to quickly search, sort and manage music collections, which will save time and increase productivity.

The aim of the qualification work is to increase the productivity of a music institution by developing a website. The website will provide a convenient and efficient way to catalogue, store and manage information about musical works. It will also allow users to create, edit and organise music collections.

The result of the work is a multifunctional and easy-to-use website that will meet the needs of users in the effective management of a music archive. It will ensure high performance and stability of the system.

The finished software product will be useful both for private music studios, radio stations and cultural institutions, as well as for individual users who want to organise personal music collections.

The qualification work consists of 79 pages, 35 figures, 7 tables and 3 appendices with the terms of reference, the program text and user manual, as well as a list of references.

Work is executed in Ukrainian language.

ЗМІСТ

ВСТУП

РОЗДІЛ I. АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ. ПОСТАНОВКА ЗАДАЧІ

- 1.1 Дослідження та аналіз предметної галузі
- 1.2 Аналіз існуючих рішень
- 1.3 Визначення проблем
- 1.4 Постановка задачі
- 1.5 Розробка концепцій рішення проблем

РОЗДІЛ II. РОЗРОБКА ПРОЕКТУ ІНФОРМАЦІЙНОЇ ПІДСИСТЕМИ

- 2.1 Загальносистемні рішення
 - 2.1.1 Загальний опис системи
 - 2.1.2 Розробка діаграми варіантів використання
- 2.2 Рішення з інформаційного забезпечення
 - 2.2.1 Визначення сутностей інформаційної підсистеми
 - 2.2.2 Розробка логічної моделі даних
 - 2.2.3 Розробка фізичної моделі даних
- 2.3 Рішення з технічного забезпечення
 - 2.3.1 Вимоги до технічного забезпечення
 - 2.3.2 Склад технічного забезпечення
- 2.4 Рішення з програмного забезпечення
 - 2.4.1 Розробка схеми структури інформаційної підсистеми
 - 2.4.2 Розробка діаграми компонентів
 - 2.4.3 Розробка діаграми переходів станів
 - 2.4.4 Розробка проекту інтерфейсу інформаційної підсистеми
 - 2.4.5 Розробка діаграми розгортання підсистеми

РОЗДІЛ III. РЕАЛІЗАЦІЯ ПРОЕКТУ ПІДСИСТЕМИ

- 3.1 Аргументація вибору засобів розробки
- 3.2 Розробка структури програмного забезпечення
- 3.3 Результати розробки програмного забезпечення

РОЗДІЛ IV. ОХОРОНА ПРАЦІ

4.1 Загальні питання охорони праці

4.2 Структура управління питаннями охорони праці на підприємстві

4.3 Шкідливі та небезпечні фактори, що супроводжують працю програміста

4.4 Ергономічні вимоги до робочого місця при роботі за комп'ютером

4.5 Нормативні вимоги до робочої площі на одного працюючого за комп'ютером

ВИСНОВОК

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

ДОДАТОК А –Технічне завдання

ДОДАТОК Б – Програмний код

ДОДАТОК В – Інструкція користувача

ВСТУП

На сьогоднішній день музика є важливою частиною життя людей. Більшість з нас любить займатись своїми справами, працюючи або відпочиваючи вмикати улюблені пісні та мелодії. У всіх мелодії викликають різні емоції та настрій. На просторах Інтернету зараз можна знайти безліч нових та цікавих пісень, різні альбоми та жанри, популярні підбірки мелодій. При цьому вибір може бути складним, адже не вся музика підходить під різний настрій та може не відповідати умовам пошуку.

Саме тому було створено вже багато музичних архівів, але не всі з них можуть бути доступними та зручними у використанні. Також більшість музичних архівів було створено без використання авторських прав, без вказування всієї інформації про створення мелодії, що унеможливорює їх використання в на офіційних платформах чи стрімінгових сервісах.

У зв'язку з цим розробка інформаційної системи для зберігання повної інформації про музичні твори є актуальним завданням, яке може полегшити життя багатьом користувачам та зберегти їхній час, зробивши пошук бажаного більш зручним та швидшим.

Актуальність теми полягає в необхідності ефективного організування та зберігання великого обсягу музичного контенту.

Метою кваліфікаційної роботи є підвищення ефективності роботи музичної установи за рахунок розробки веб-сайту, що надасть змогу зручно та швидко знаходити потрібні пісні, додавати їх до альбомів, плейлистів, а також здійснювати всі необхідні операції з управління колекціями.

РОЗДІЛ І. АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ. ПОСТАНОВКА ЗАДАЧІ

1.1 Дослідження та аналіз предметної галузі

Веб-сайти на сьогоднішній день користуються шаленою популярністю. Зараз веб-сайтами користуються не лише великі компанії, а й ВНЗ, невеликі підприємства, клуби та звичайні люди.

В умовах високої конкуренції та зі стрімким розвитком новітніх технологій організаціям необхідно створювати різні інформаційні системи з потрібними інструментами для успішного ведення бізнесу в нинішніх реаліях.

У цифрову епоху й музична індустрія зазнає значних змін. Перехід від традиційних аналогових носіїв до цифрових форматів допоміг збільшити доступність, зручність і потужність обробки музичного вмісту. Однак це також створює проблеми для ефективного зберігання, організації та керування цими великими обсягами цифрових даних.

Пошук інформації про музику передбачає перегляд та організацію величезних музичних колекцій та відповідності інформації конкретним запитам. Це особливо актуально, якщо враховувати незлічену кількість музичної інформації та її різноманіття.

Розробка інформаційної системи для зберігання повної інформації про музичні твори та виконавців є актуальним завданням, яке може полегшити життя багатьом користувачам та зберегти їхній час, зробивши пошук бажаного більш зручним та швидшим.

Актуальність теми полягає в необхідності ефективного організування та зберігання великого обсягу музичного контенту.

Веб-сайт «Музичний архів» призначений для збереження музичних творів, альбомів, плейлистів. Він допомагає організовувати та розподіляти музичні твори за жанрами, виконавцями, альбомами та плейлистами, виконавцями [1]. Наразі подібні ресурси користуються великою популярністю серед людей, які люблять слухати музику, та прагнуть відпочити від буденності.

Перевагами музичного архіву є :

- Масштабність. Архіви можуть включати в себе необмежену кількість інформації про музичні твори;
- Збереження. Забезпечення безпеки даних та каталогізація музичного архіву в електронному вигляді;
- Автоматизація. Надання доступу до інформації для культурних та розважальних цілей;
- Індексація. Передбачає належну класифікацію та індексацію кожного треку, щоб забезпечити швидкий доступ до необхідної інформації.

Музичний архів відіграє важливу роль у збереженні та вивченні музичної культури. Музичні архіви є джерелом інформації для музикантів, композиторів, звукорежисерів та журналістів [2].

Характерними завданнями при створенні музичного архіву є :

- Обсяг і різноманітність комплектування архівів музичними творами.
- Забезпечення збереження повної інформації про музичні твори (виконавець, жанр, композитор, альбом до якого належить, та плейлист до якого входить тощо).
- Можливість керування інформацією (додавання, оновлення, редагування, видалення).

1.2 Аналіз існуючих рішень

У повсякденному житті роль плесрів, радіо та телебачення поступово зменшується, оскільки такі джерела інформації стають все менш ефективними. Найкращими та найсучаснішими ресурсами для отримання інформації є всесвітня мережа Інтернет, яка об'єднує величезну кількість веб-сайтів та додатків, створюючи таким чином головне інформаційне середовище.

Необхідність реалізації веб-ресурсу є пріоритетною для людей, що займаються комерційними послугами та розповсюдженням інформації.

Оскільки існує велике різноманіття інформаційних систем для управління музичними архівами, що призначені для різних потреб користувачів у різних галузях, включаючи аудіо-індустрію, медіа-компанії, культурні установи, радіостанції, особисті та колективні музичні архіви.

Для кращого розуміння роботи музичних платформ важливо розглянути існуючі рішення в цій сфері.

Media Asset Management системи — системи, що спеціалізуються на управлінні різними видами сучасних медіа, включаючи аудіо- та відеофайли. Вони надають можливості каталогізації, організації, пошуку, індексації та зберігання аудіофайлів [3]. Прикладом таких систем є: Dalet Galaxy(Рисунок 1.1), Avid Interplay(Рисунок 1.2), MediaBeacon тощо.

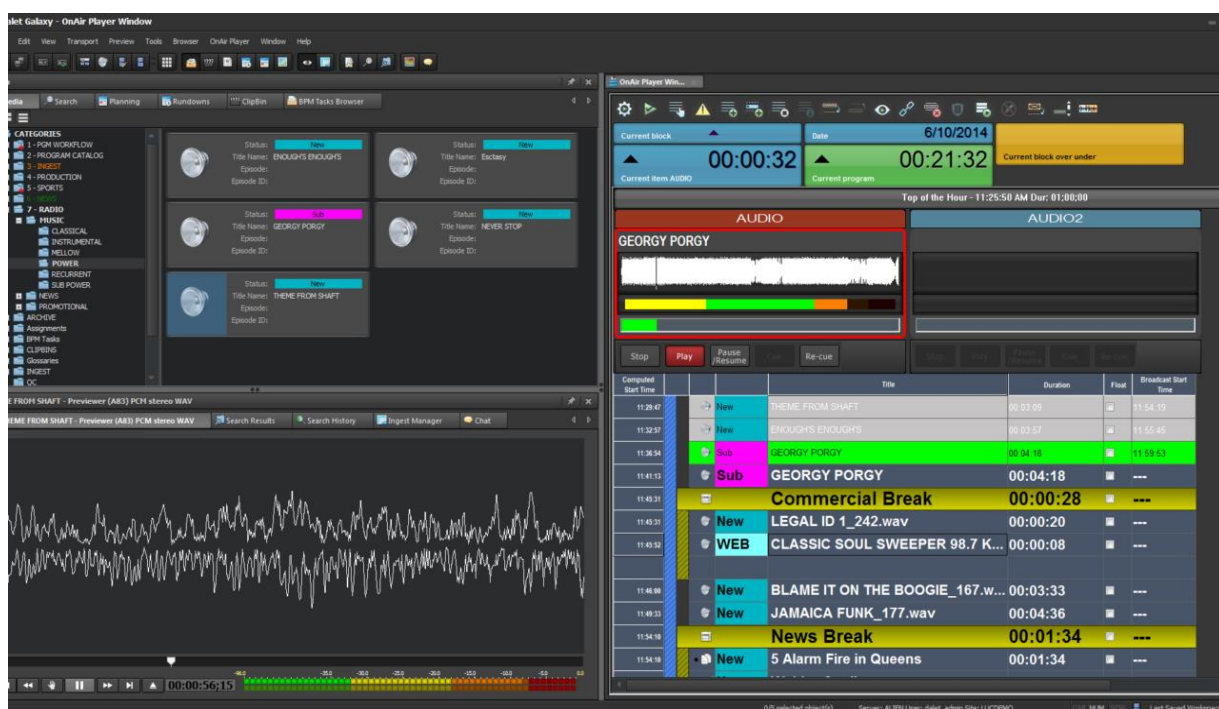


Рисунок 1.1 – Dalet Galaxy

Системи дають змогу керування всіма медіа, включаючи музичні, що дозволяє швидко каталогізувати, організовувати та використовувати музичні архіви [4]. Використовуючи індексацію користувачі зможуть швидко знаходити потрібні файли, мають змогу користуватися пошуком за ключовими

словами та фільтрами з різними параметрами; системи дозволяють багатьом користувачам одночасно працювати з медіа-файлами, що надає можливість спільній роботі та співпраці на музичних архівах.

Недоліками систем є: високі витрати на використання, інтегрування з іншими системами може вимагати додаткового програмного забезпечення та налаштування.

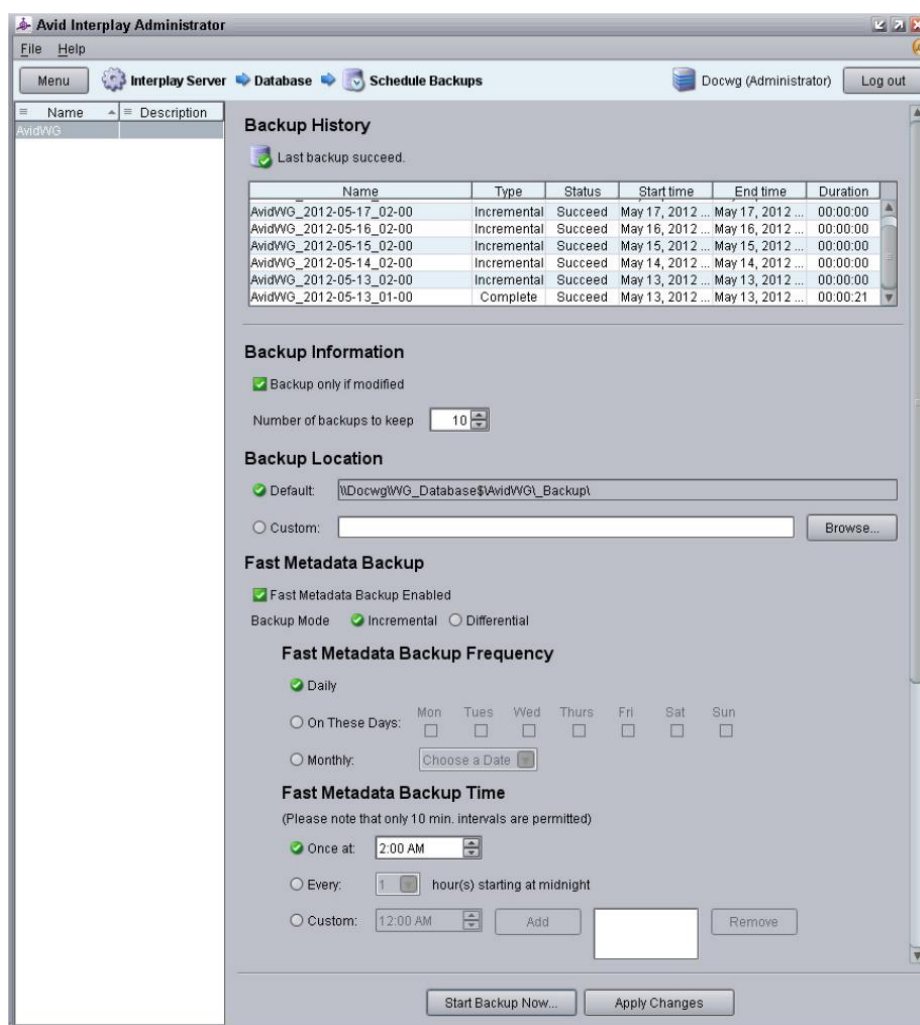


Рисунок 1.2 – Avid Interplay

Цифрові аудіо-архівні системи — системи, які спеціалізуються на збереженні аудіо-файлів у відповідності до стандартів довгострокової архівації. Зазвичай вони мають вбудовані функції для забезпечення цілісності даних та можливості резервного копіювання [5]. Яскравими прикладами таких систем є: Artefact, Memnon(Рисунок 1.3), Rosetta(Рисунок 1.4);



Рисунок 1.3 – Цифрова аудіоархівна система Memnon

Home / Data Management: Publishing Configuration

Filter: All in All Go

Add Configuration 1 - 8 of 8 configurations

	Active	Name	Description	Update Date				
1	✓	getty		14/03/2016 16:44:44	Update	Duplicate	Delete	Sync Configuration
2	✓	Getty apos		14/03/2016 14:24:48	Update	Duplicate	Delete	Sync Configuration
3	✓	nfs		09/06/2014 04:02:58	Update	Duplicate	Delete	Sync Configuration
4	✓	Primo Publish Conf	Primo Publish Conf	25/02/2016 13:19:48	Update	Duplicate	Delete	Sync Configuration
5	✓	Rina Test	Rina Test	29/11/2016 09:30:12	Update	Duplicate	Delete	Sync Configuration
6	✓	South Africa		13/03/2013 13:42:42	Update	Duplicate	Delete	Sync Configuration
7	✓	test 24 May	test	30/10/2016 17:32:12	Update	Duplicate	Delete	Sync Configuration
8	✓	test test		30/06/2016 17:23:14	Update	Duplicate	Delete	Sync Configuration

1 - 8 of 8 configurations

Рисунок 1.4 – Налаштування конфігурацій на платформі Rosetta

Перевагами цих систем є: легкий доступ та пошук інформації, надійний захист завдяки резервному копіюванню, легке копіювання та розповсюдження інформації

Недоліками таких систем є: потреби в постійній технічній підтримці, розгортання та своєчасна підтримка систем потребує значних витрат на спеціалізоване обладнання та програмне забезпечення для його використання.

Системи управління контентом — системи що є важливим інструментом для створення, використання та управління веб- та цифровим контентом. Деякі з CMS можна адаптувати для управління музичним контентом, що дозволить користувачам завантажувати, організовувати та відтворювати аудіофайли. Найпопулярнішими приладами систем є WordPress [6], Joomla(Рисунок 1.5), або Drupal, мають інтуїтивно простий інтерфейс, що дозволяє користувачам без професійних навичок легко створювати та редагувати потрібний контент.

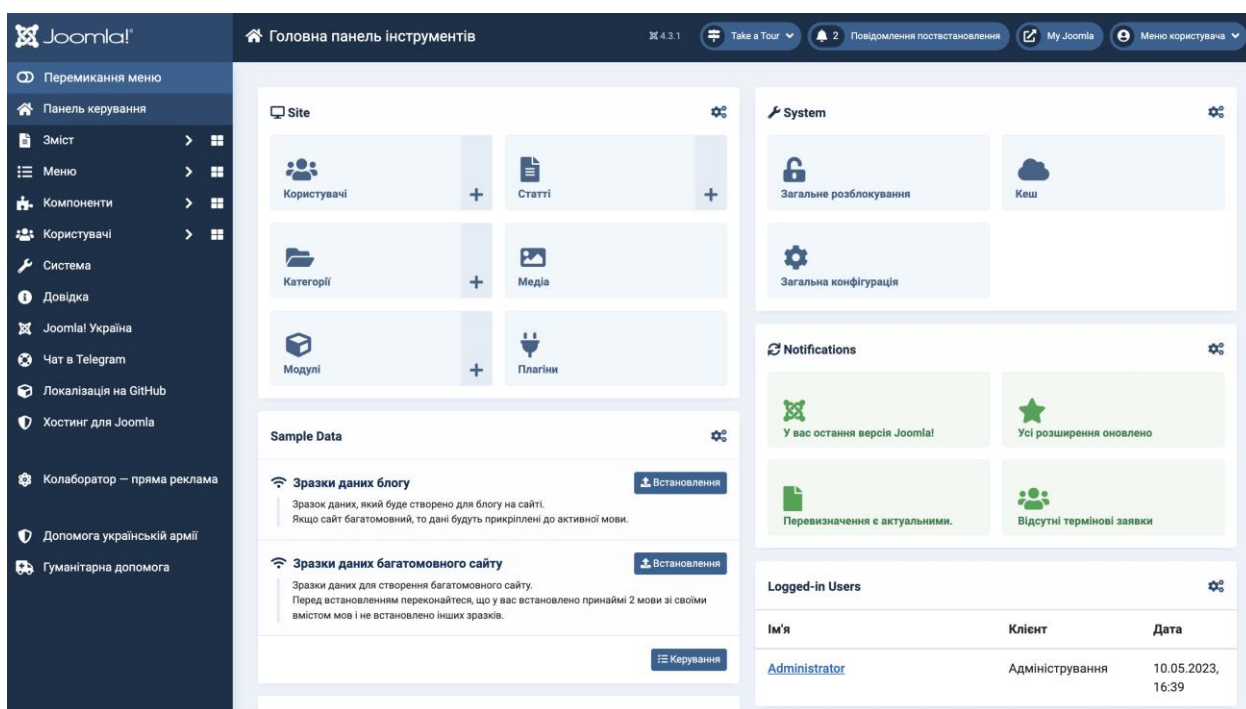


Рисунок 1.5 – CMS система Joomla

Перевагами цих систем вказують: легкість використання, швидке реагування.

Недоліками таких систем є: використання додаткових компонентів, постійна підтримка та оновлення програмного та апаратного забезпечення.

Отже, нині системи керування контентом дозволяють легко та без особливих знань та навичок керувати сайтами, корегувати їх роботу. Дають

додавати чи редагувати всю потрібну інформацію, також створювати особисті шаблони та використовувати з переліку вже наявні.

Існує також незліченна кількість спеціалізованих платформ, що призначені саме для управління музичним контентом, такі як SoundCloud та Spotify (Рисунок 1.6) для спільного доступу до музики, SoundDrop та Billboard World Albums (Рисунок 1.7) для продажу та дистрибуції музичних записів, або системи управління музичними бібліотеками, які розробляються спеціально для культурних установ або радіостанцій та телемереж.

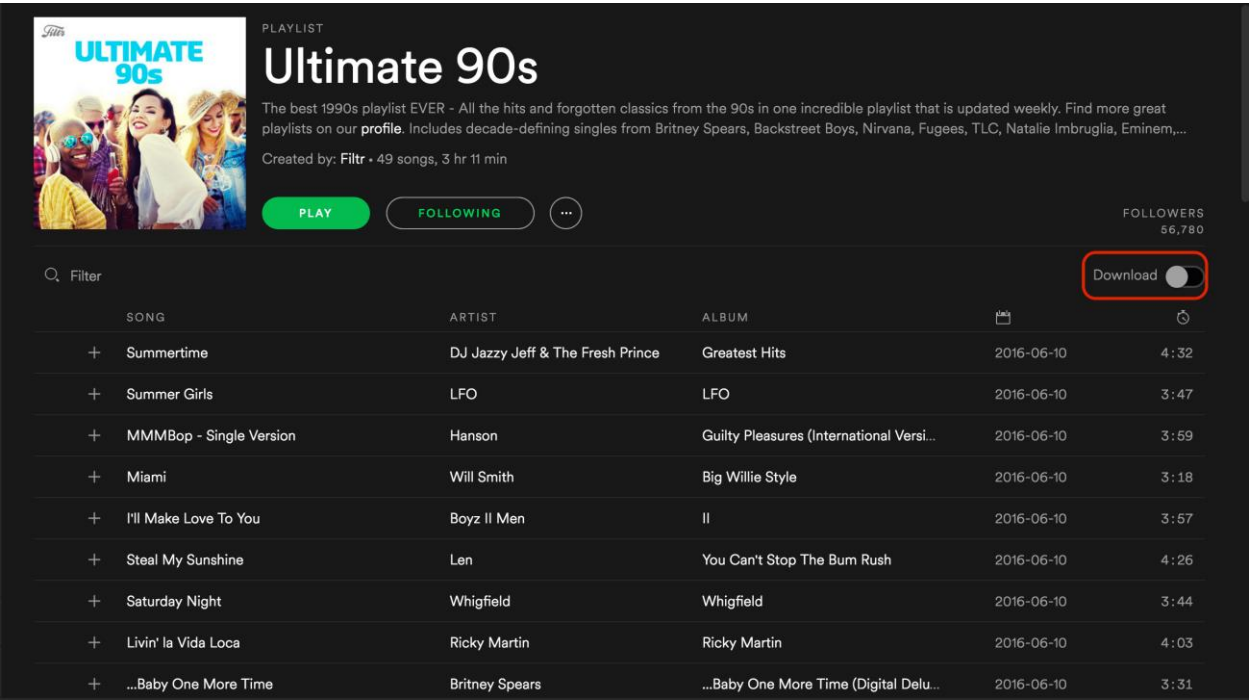


Рисунок 1.6 – Платформа Spotify

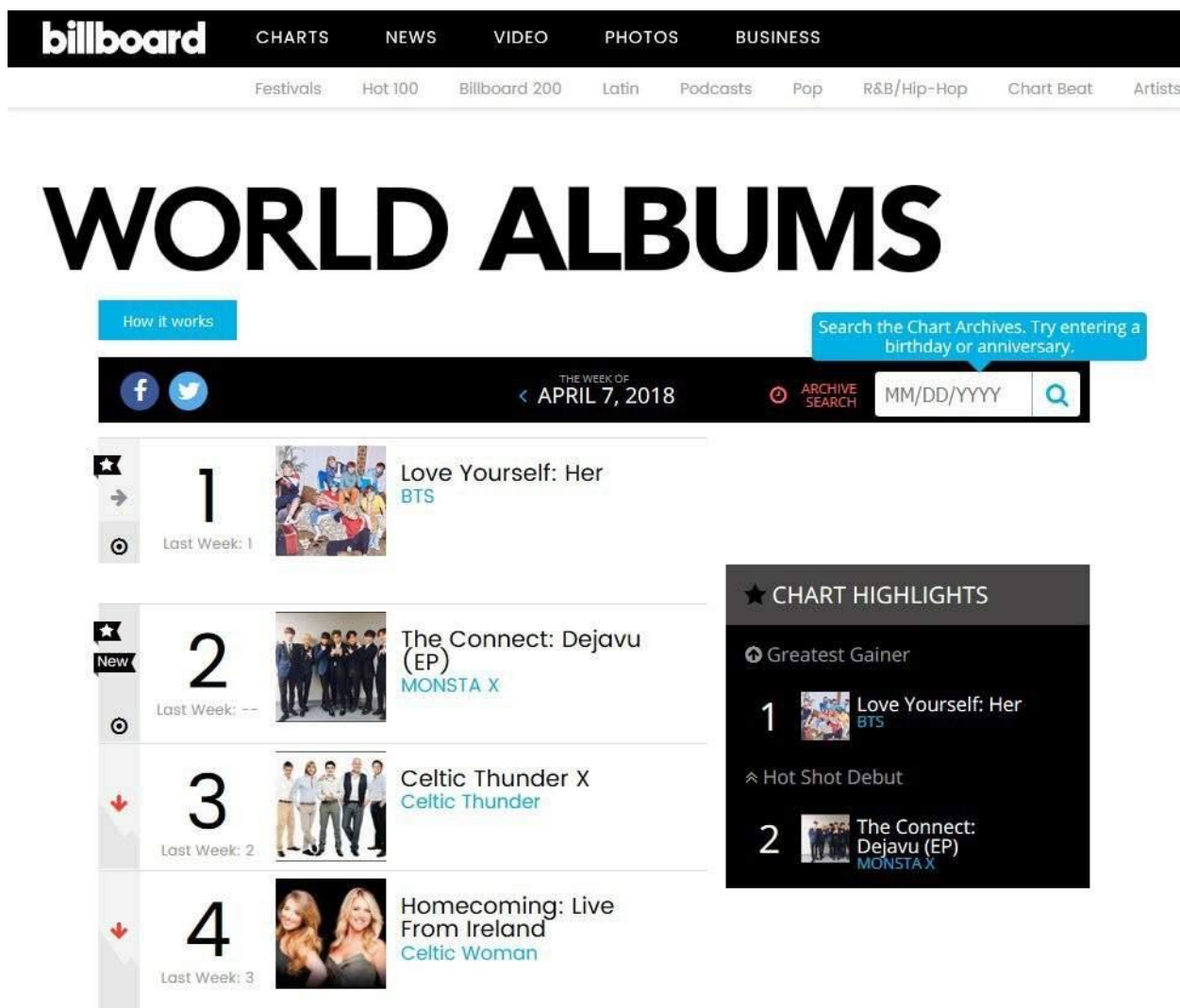


Рисунок 1.7 – Приклад роботи програми Billboard World Albums.

Одними з переваг музичних платформ є: фокусування на музичному контенті, можливість зручного прослуховування та знаходження новинок в музиці, аналітика та звітність

Недоліками таких систем зазвичай є: обмежений функціонал, висока вартість розробки та низька конкурентність.

На основі проведеного аналізу існуючих систем стає зрозуміло, що існує багато різноманітних підходів до розробки інформаційних систем для управління музичними архівами. Також можна визначити ключові елементи, котрі потрібно інтегрувати в розробку нової системи.

Виявлено, що індустрія музичних архівів використовує широке різноманіття програмних продуктів та платформ, від систем управління

різним контентом до вузькоспеціалізованих платформ та цифрових музичних архівів.

Тому має сенс почати розробку інформаційної системи сайту для управління музичним архівом. Основними перевагами якої буде:

- Інтуїтивно зрозумілий інтерфейс;
- Швидкість та простота у використанні;
- Безкоштовність;
- Можливість інтеграції з іншими системами.

1.3 Визначення проблем

У процесі розробки інформаційної систем сайту для управління музичним архівом можуть виникати різноманітні проблеми, які потрібно вирішити для швидкого та ефективного функціонування системи. Для ефективної розробки інформаційної системи сайту потрібно враховувати такі проблеми:

- Музичні архіви включають величезні обсяги важливої інформації, яка потребує створення потужних систем зберігання та обробки даних;
- Ефективне управління музичними архівами повинне передбачати належну класифікацію та індексацію кожного треку, щоб забезпечити максимально швидкий доступ до необхідної інформації;
- Забезпечення доступності та швидкості, для редагування або оновлення наявної інформації вимагає оптимізації роботи системи.

1.4 Постановка задачі

Існує незліченна кількість веб-застосунків, що пропонують різноманітні можливості для управління музичним контентом. Їх було створено з метою полегшити та спростити доступ до музичних даних для користувачів.

Однією з головних незручностей є обмежений функціонал, так званий ‘базовий’ функціонал: додання треку, перегляд альбомів та пошук потрібних треків. Проте вони часто не забезпечують повного спектру можливостей для організацій та керування великими музичними колекціями.

Враховуючи ці обмеження, постає необхідність розробки нового веб-сайту, який забезпечить повний спектр можливостей для управління музичним контентом та задовольнить різноманітні потреби користувачів.

Основними завданнями які потрібно вирішити є:

- 1) Проаналізувати данні після проведеного дослідження та вимоги щодо функціональності системи та інтерфейсу.
- 2) Створити архітектурний план системи, що включає в себе базу даних, серверну та клієнтську частину веб-застосунку.
- 3) Реалізувати основні функції веб-застосунку:
 - відображення даних;
 - перегляд сторінок альбомів, виконавців, треків та плейлист
 - пошук та каталогізація контенту;
 - додання, редагування та оновлення інформації про треки;
 - видалення непотрібної інформації;
 - можливість створення, редагування та видалення плейлистів, управління альбомами;
 - можливість додання інформації про нових виконавців.
- 4) Створити зручний та інтуїтивно зрозумілий інтерфейс користувача, що відповідає вимогам мінімалістичному дизайну та потребам користувачів.

- 5) Провести тестування додатку на різних етапах розробки, виявляти та виправляти недоліки.
- 6) Забезпечити сумісність з різними браузерями.
- 7) Після впровадження системи забезпечити стабільну роботу застосунку та надати користувачам необхідну підтримку, а також здійснювати регулярні оновлення та вдосконалення.

Результатом виконання поставлених завдань повинен стати веб-додаток для управління музичним архівом.

Технічне завдання на розробку інформаційної системи представлено в додатку А.

1.5 Розробка концепцій рішення проблем

На рисунку 1.8 зображено модель концепції системи сайту Spotify.

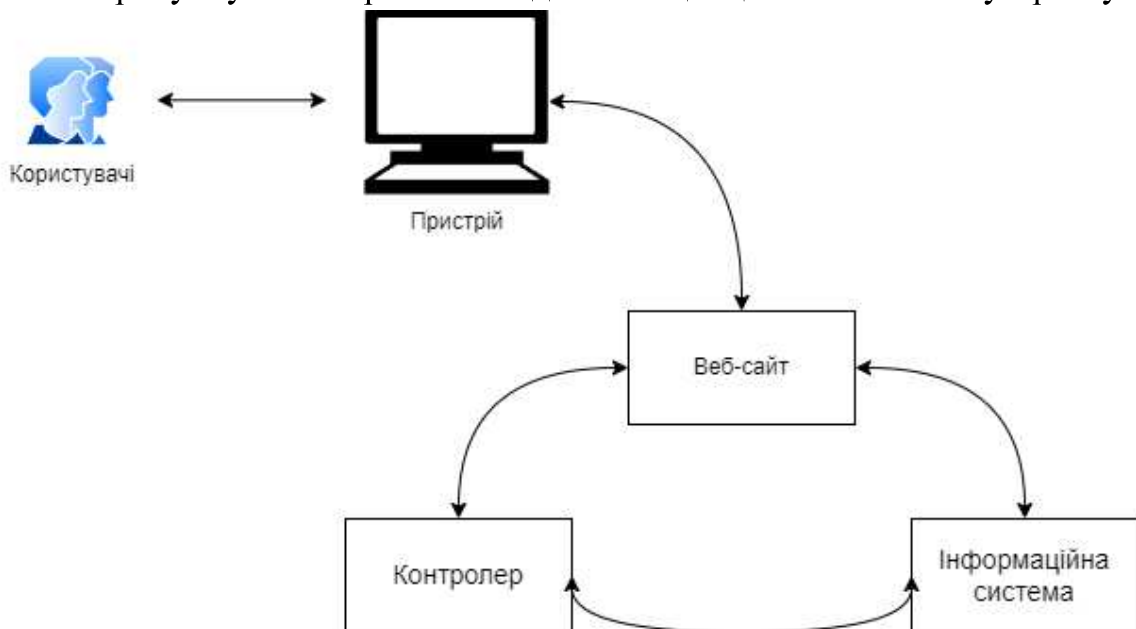


Рисунок 1.8 – Модель концепції сайту Spotify

На основі сформованих вимог розроблено модель концепції інформаційної системи сайту, яку зображено на рисунку 1.9.

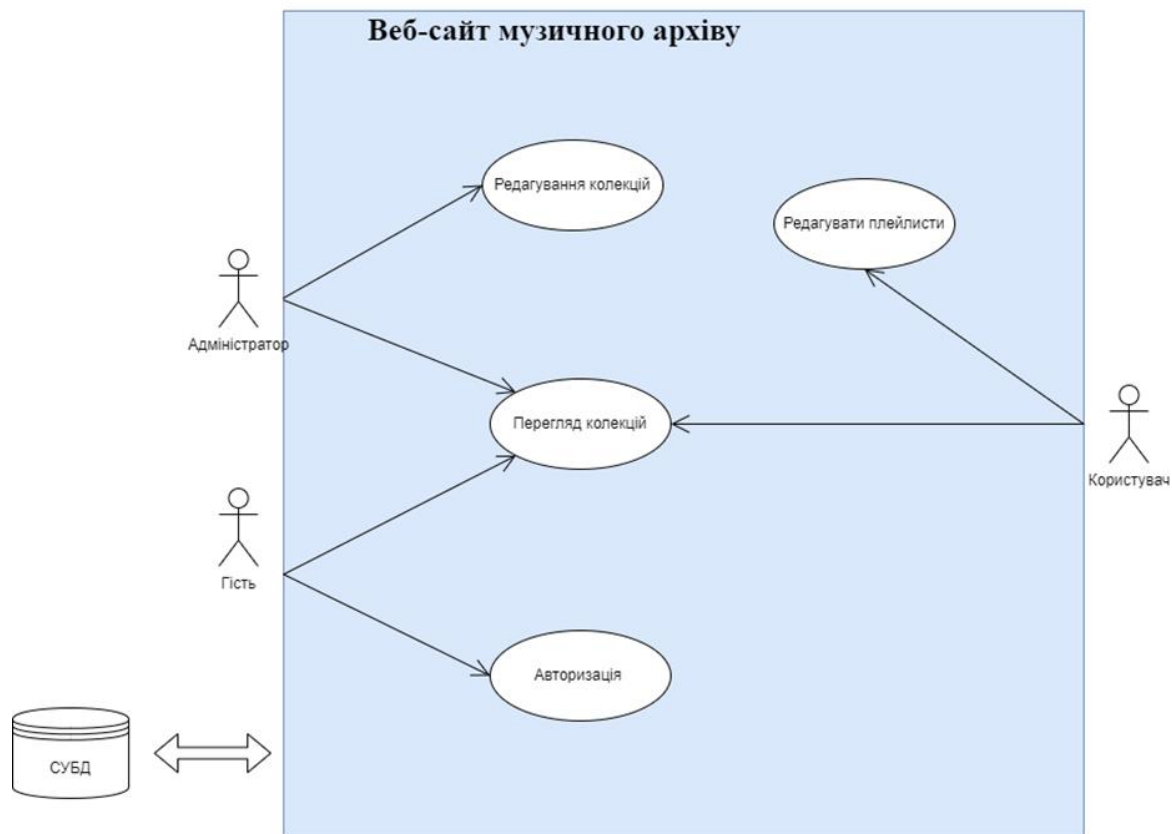


Рисунок 1.19 – Модель концепції сайту для управління музичним архівом

- Формування інформації про музичні записи з різних джерел. Дані можуть бути отримані з різних джерел, таких як музичні сервіси, медіа-холдинги, різні музичні лейбли. Після збору данні потрібно впорядкувати та систематизувати, класифікувати за жанрами, виконавцями та роками.
- Автоматичне визначення метаданих пісень (назва, виконавець, жанр, рік випуску тощо). Програмні засоби можуть звертатися до онлайн джерел, таких як музичні сервіси або бази даних, для отримання метаданих про пісні за їх звуковими характеристиками;
- Збереження інформації та відповідних метаданих в базі даних. Вся інформація про музичні записи та відповідні метадані можуть бути

збережені в одній базі даних, що спрощує доступ до неї та управління.

- Створення альбомів та плейлистів для організації даних за різними критеріями (жанр, виконавець, альбом, рік тощо). Користувач може створювати власні музичні альбоми, об'єднуючи пісні за певними критеріями, наприклад, за жанром, виконавцем, альбомом або роком випуску. Це дозволяє створювати персоналізовані колекції музики для перегляду за власними уподобаннями.
- Фільтрація музичних записів за різними критеріями (рік випуску, тривалість, рейтинг тощо). База даних дозволяє швидко знаходити та витягувати необхідну інформацію про музичні записи за допомогою запитів та фільтрів.
- Можливість редагувати та оновлювати інформацію про музичні записи (зміна назви, виконавця, жанру тощо). Це дозволяє активно керувати змістом музичних колекцій.

Внутрішня структура сайту представлена на рисунку 1.10.



Рисунок 1.10 – Внутрішня структура сайту

Структурування веб-сайту допомагає розподілити сторінки сайту як зручності під час використання чи розробки, так і для пошуку потрібних даних. Багато сайтів не мають чіткої структури, яка спрямовувала б

користувачів до потрібної інформації. Наявність чіткої структури сайту також дозволяє краще зрозуміти основні аспекти сайту.

Система управління музичним архівом призначена для формування, зберігання та управління музичним творами та відповідними метаданими. Вона дозволяє користувачам легко знаходити, переглядати та організовувати свою музичну колекцію.

На основі спроектованої концепції виконується розробка проєкту інформаційної системи сайту для управління музичним архівом, яка дозволить досягти поставленої мети.

РОЗДІЛ II. РОЗРОБКА ПРОЕКТУ ІНФОРМАЦІЙНОЇ ПІДСИСТЕМИ

2.1 Загальносистемні рішення

2.1.1 Загальний опис системи

Система управління музичним архівом - це інформаційна система, яка дозволяє користувачам ефективно збирати, організовувати, зберігати та надавати доступ до їхнього музичного контенту.

Ця система розроблена з метою спростити та поліпшити процеси управління музичним архівом для зручності та задоволення потреб користувачів. Вона забезпечує можливість додавання нових музичних записів, організацію їх у альбоми та плейлисти, оновлення та редагування інформації.

Інформація, що міститься в цьому веб-застосунку, повинна бути представлена у зручній для користувачів формі та відповідати зазначеній структурі. Користувачам повинні мати можливість додавати, редагувати, оновлювати та видаляти інформацію.

В діаграмі варіантів використання важливі головні елементи взаємодії для повного відображення та більш чіткого розуміння взаємозв'язків між різними частинами системи що створюється.

Веб-застосунок повинен містити розділи для надання інформації про:

- загальна інформація про альбоми;
- повна інформація про альбоми;
- загальна інформація про музичні треки;
- повна інформація про альбоми, можливість оновлення/редагування/видалення інформації;
- інформація про плейлисти;
- загальна інформація про виконавців;
- повна інформація про виконавців, можливість оновлення/редагування/видалення інформації;
- публікація нових музичних треків,
- створення нових альбомів;

- створення плейлистів;
- додавання інформації про нових виконавців.

2.1.2 Розробка діаграми варіантів використання

Діаграми варіантів використання описують різні сценарії співдії між користувачами та системою. В свою чергу це допомагає зрозуміти, як працює система в різних ситуаціях.

Діаграма варіантів використання — це один з багатьох типів діаграм, що використовуються в моделюванні системи за методологією Unified Modeling Language (UML) для моделювання програмних систем [7]. Така діаграма показує кілька акторів та варіанти використання, що пов'язані один з одним через створення асоціацій. Варіанти використання та дійові особи на діаграмі варіантів використання показують, що саме робить система та як її використовують учасники. Діаграма станів не описує те, як внутрішньо працює система.

Вже створена діаграма варіантів використання включає в себе такі ролі:

- актор — цю роль виконують гість, користувач та адміністратор;
- граничні межі системи — прямокутник, в якому вказуються всі можливі дії всередині системи.

Наявна діаграма станів системи має три ролі:

- Гість може переглядати інформацію про треки, альбоми та плейлисти.
- Користувач має можливість перейти на головну сторінку та ознайомитись із доступними йому розділами. Шляхом переходу між ними, Користувач може ознайомитись із переліком створених альбомів, музичними треками, плейстами та іншою інформацією пов'язаною з музичними треками, а також створювати плейлисти за настроєм та тематикою. Користувач може шукати музичні

твори за різними критеріями та застосовувати фільтри до результатів пошуку.

- Адміністратор має більший спектр можливостей. У нього є доступ до перегляду й редагування всіх таблиць системи, з метою виправлення помилок або внесення коректив. Може створювати нові альбоми та плейлисти, додавати нових виконавців. Додавати перелік жанрів та тематик плейлистів, вносити дані про нові студії звукозапису.

Елементи взаємодії зображені в діаграмі варіантів використання важливі для показу та розуміння взаємозв'язків між різними частинами системи.

Діаграма варіантів використання зображено на рисунку 2.1.

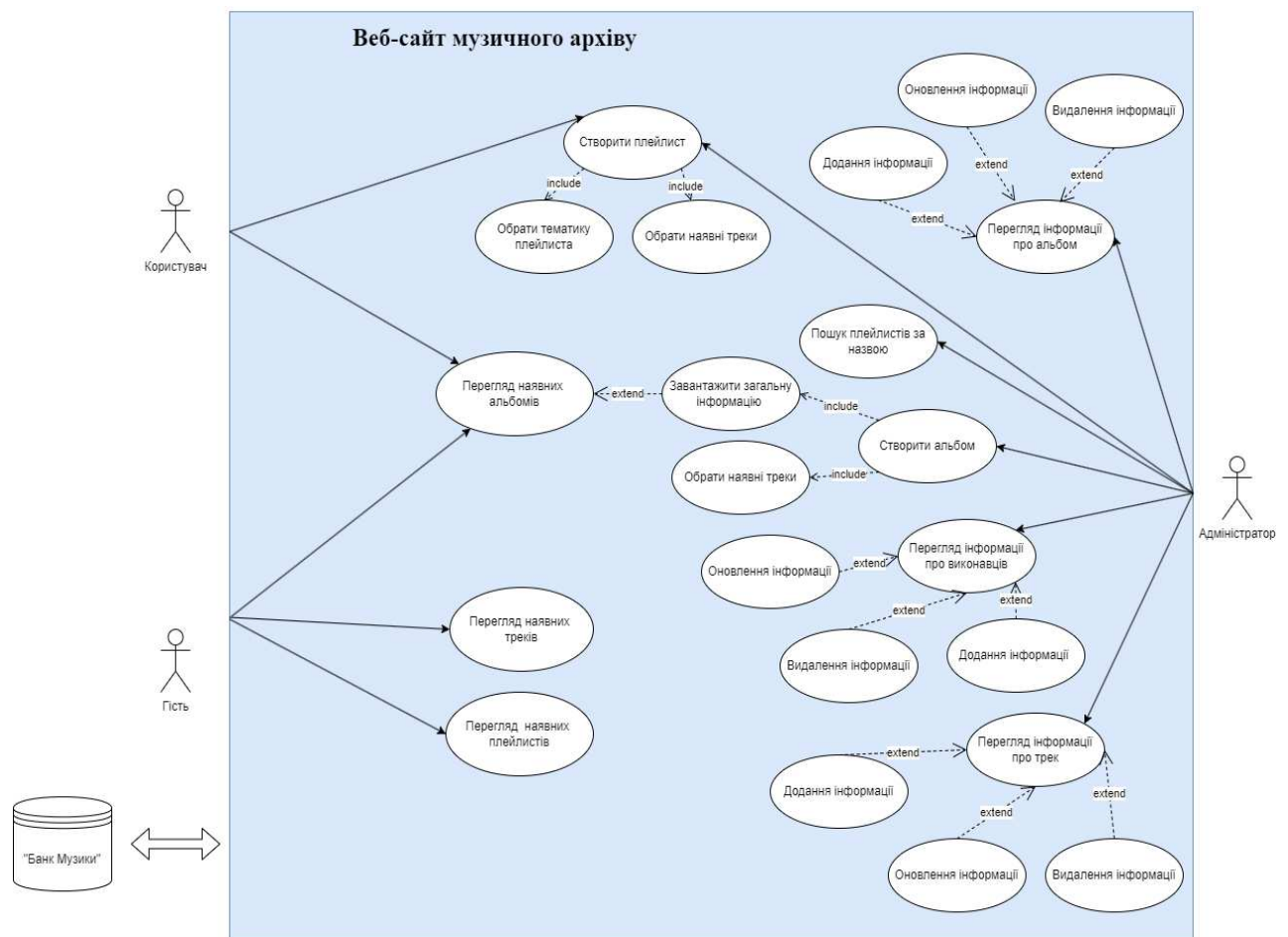


Рисунок 2.1 — Діаграма варіантів використання

Вони представляють зв'язки між акторами та варіантами використання:

- використовує (extend) — актор розпочинає основні дії;

- включає (include) — прецедент включає дії з іншого прецеденту;

Діаграми варіантів використання крім того дозволяють:

- визначати функції, які повинна виконувати система;
- зрозуміти, як саме користувачі співдіють із системою та якими є їх ролі;
- опрацювати різні сценарії співдії між користувачами та системою, описати їх порядок;
- зазначити, на які проблеми натрапляють користувачі та як їх може уникнути система;
- визначити функції системи, що мають найвищий пріоритет серед користувачів системи.

2.2 Рішення з інформаційного забезпечення

2.2.1 Визначення сутностей інформаційної підсистеми

Для ефективного функціонування інформаційної підсистеми важливо чітко визначити сутності, які входять до її складу та представляють ключові об'єкти та концепції, що використовуються в цій системі.

Основні сутності для інформаційної підсистеми:

- Музичний запис: Представляє інформацію, яка додається до архіву користувача. Складається з метаданих, такі як назва пісні, виконавець, альбом, жанр та тривалість.
- Альбом: Групує музичні записи за виконавцями. Кожен альбом може містити декілька музичних записів.
- Плейлист: Представляє собою список музичних записів, які обрані користувачем для перегляду та каталогізації. Користувач може створювати різні плейлисти для різних ситуацій або настроїв.
- Виконавець: Представляє виконавця системи, який має доступ до управління музичним архівом. Включає в себе інформацію про

ім'я та прізвище, країну походження, рік народження та інші персональні дані.

- Жанр: Категорія, до якої відноситься музичний запис або альбом (наприклад, рок, поп, класика тощо).
- Студія звукозапису: Представляє список студій звукозапису, на якій проводився запис. Включає інформацію, таку як назву, виконавця.

2.2.2 Розробка логічної моделі даних

Одним із ключових аспектів при проєктуванні системи є розробка логічної моделі бази даних. Створення структури даних є метою процесу, який вдосконалює та оптимізує обробку та зберігання інформації, гарантуючи надійність та швидкість системи.

При розробці логічної моделі бази даних використовуються різні форми нормалізації щоб покращити структуру даних та забезпечити мінімізацію дублювання та невідповідність використаних даних. У цьому проєкті використовуються перша, друга та третя форми нормалізації (1NF, 2NF та 3NF) [8].

Форми нормалізації поділяються на:

- перша нормальна форма (1NF) - вимагає, аби кожна колонка в таблиці містила лише атомарне значення, а саме кожне з полів повинне містити лише одне значення певного типу.
- друга нормальна форма (2NF) - потребує, щоб всі атрибути, які не є частиною первинного ключа, повністю залежать від цього ключа. Це допомагає запобігти повторенню даних та гарантує, що зміни внесені в одному місці відбуваються в одній таблиці.
- третя нормальна форма (3NF) - вимагає, аби атрибути, котрі не є ключами, залежали тільки від первинного ключа, не відносячись до інших

неключових атрибутів. Це сприяє мінімізації дублювання даних та забезпечує цілісність даних [8].

Логічну модель даних зображено на рисунку 2.2.

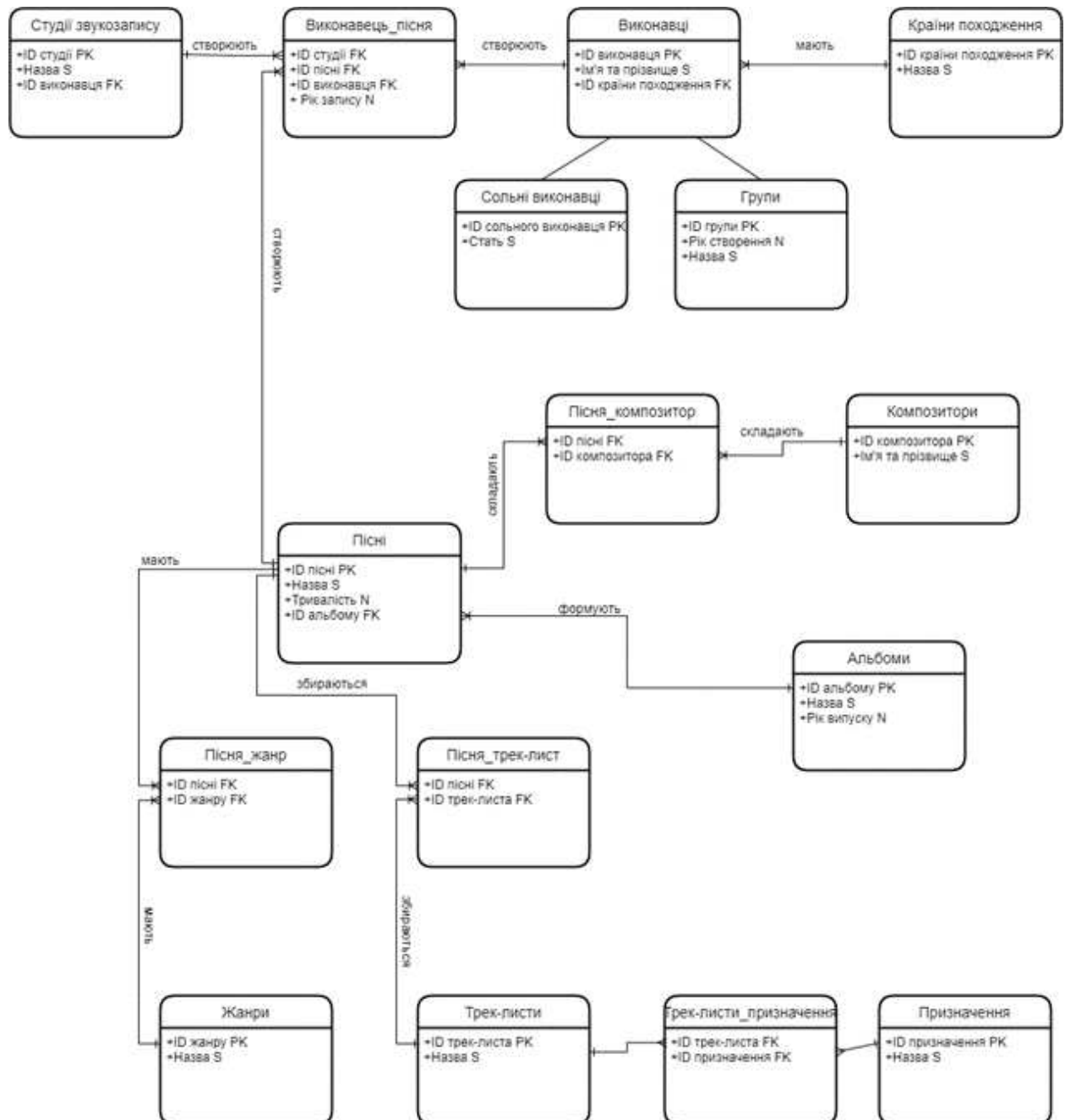


Рисунок 2.2 – Логічна модель даних

Аналіз на нормальні форми:

1. Атрибут Ім'я та прізвище (з табл. Виконавці) не використовується для сортування або пошуку по частинам. Тому перебуває у 1NF, тому відношення є простими.

2. Відношення перебуває в 2NF, тому що воно вже перебуває у 1NF, де всі неключові атрибути зв'язків функціонально повністю залежать саме від первинного ключа.

3. Відношення знаходиться у 3NF, тому що воно вже перебуває в 2NF, і всі неключові атрибути нетранзитивно залежать від ключа.

Після аналізу відомо, що спроектована база даних знаходиться у 3 посиленій нормальній формі Бойса-Кодда.

Результатом проектування бази даних є створення логічної моделі, яка відображає розподіл даних та оптимізує співдію між іншими елементами системи.

Нормалізація сприяє створенню оптимальної структури даних, що в свою чергу забезпечує ефективну роботу системи, гарантує надійність та цілісність даних. Важливою частиною процесу розробки програмного продукту та системи є розробка схеми бази даних, адже воно зазначає провідні способи опрацювання та збереження інформації.

2.2.3 Розробка фізичної моделі даних

Фізична модель даних — модель, що визначає, як дані розташовані в середовищі збереження, а також як здійснюється доступ до цих даних, їх підтримка на фізичному рівні. На відміну від логічної моделі даних, вона детально описує структурування бази даних, використовуючи чітко визначені характеристики та специфікацій для обраної системи управління.

Фізична модель повністю побудована на основі логічної й строго відповідає структурі: таблиці - сутність, а стовпцю — атрибут.

В Таблиці 2.1 визначено відповідність сутностей логічної моделі таблицям фізичної моделі.

Таблиця 2.1 — Відповідність сутностей таблицям

Сутність	Таблиця
Музичний запис	Track
Альбом	Album
Плейлист	Playlist
Виконавець	Artist
Жанр	Genre
Студія звукозапису	Record_label

З зазначеній вище інформації зрозуміло, що фізична модель даних деталізує реалізацію бази даних у конкретній СУБД. При цьому кожен стовпець повинен відповідати типу даних, що зберігати.

Фізичну модель даних зображено на рисунку 2.3.

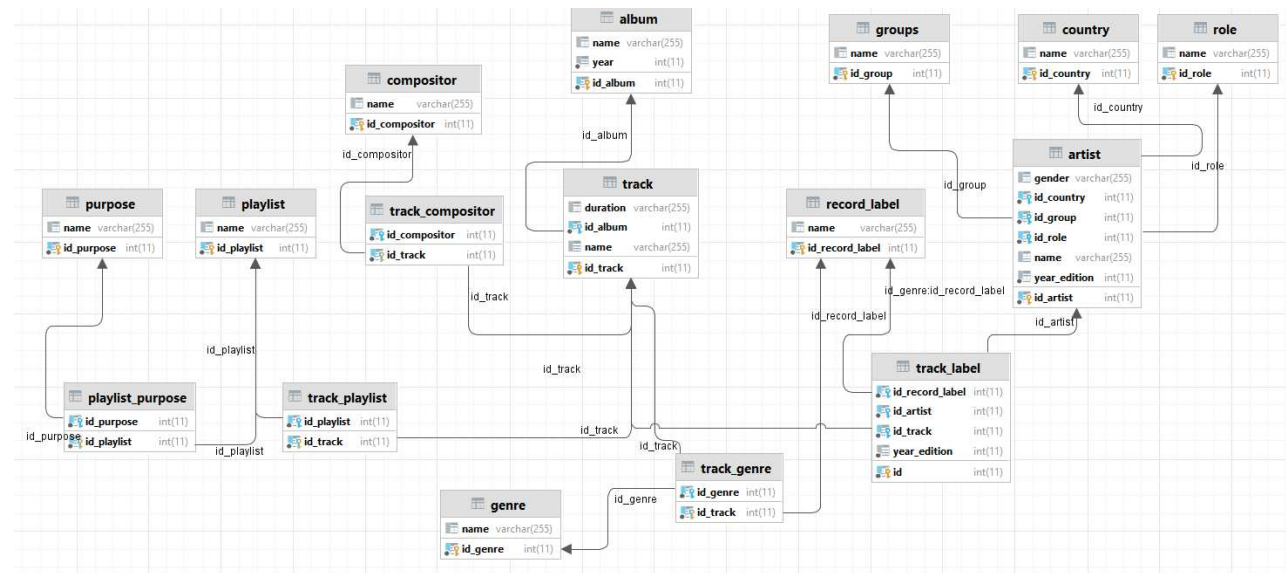


Рисунок 2.3 – Фізична модель даних

У таблицях. 2.2, 2.3, 2.4, 2.5, 2.6, 2.7 наведено стовпці для всіх таблиць системи, Album, Artist, Track і Playlist, Genre, Record_label відповідно з вказанням типу даних та виду ключа.

Таблиця 2.2 – Таблиця Album

Ім'я стовпця	Тип	Обов'язкове поле	Вид ключа
Id_album	Int	NOT NULL	Первісний ключ
name	Varchar	NULL	
Year	Int	NOT NULL	

Таблиця 2.3 – Таблиця Artist

Ім'я стовпця	Тип	Обов'язкове поле	Вид ключа
Id_artist	Int	NOT NULL	Первісний ключ
name	Varchar	NOT NULL	
gender	Varchar	NOT NULL	
year_edition	Int	NOT NULL	
id_country	Int	NOT NULL	Зовнішній ключ
id_group	Int	NOT NULL	Зовнішній ключ
id_role	Int	NOT NULL	Зовнішній ключ

Таблиця 2.4 – Таблиця Track

Ім'я стовпця	Тип	Обов'язкове поле	Вид ключа
Id_track	Int	NOT NULL	Первісний ключ
name	Varchar	NULL	
id_album	Int	NOT NULL	Зовнішній ключ
duration	Varchar	NULL	

Таблиця 2.5 – Таблиця Playlist

Ім'я стовпця	Тип	Обов'язкове поле	Вид ключа
Id_playlist	Int	NOT NULL	Первісний ключ
name	Varchar	NULL	

Таблиця 2.6 – Таблиця Genre

Ім'я стовпця	Тип	Обов'язкове поле	Вид ключа
Id_genre	Int	NOT NULL	Первісний ключ
name	Varchar	NULL	

Таблиця 2.7 – Таблиця Record_label

Ім'я стовпця	Тип	Обов'язкове поле	Вид ключа
id_record_label	Int	NOT NULL	Первісний ключ
name	Varchar	NULL	

2.3 Рішення з технічного забезпечення

2.3.1 Вимоги до технічного забезпечення

До системних вимог комп'ютера, на якому заплановано застосування даної інформаційної системи, що дозволить зручно та швидко взаємодіяти зі створеним веб-сайтом:

- операційна система macOS чи Microsoft Windows;
- частота процесора не менше 2,10 ГГц;
- оперативна пам'ять не менше 8 Гб ;
- жорсткий диск від 1 Гб;
- звукова карта 7.1.

2.3.2 Склад технічного забезпечення

Користувачу слід володіти пристроєм підключеним до мережі Інтернет, ним може бути комп'ютер чи ноутбук з операційною системою Windows, MacOS. На якому має бути встановлено будь-який браузер що підтримує JavaScript. Також має бути сервер, на якому планується подальше розгортання інформаційної системи.

2.4 Рішення з програмного забезпечення

2.4.1 Розробка схеми структури інформаційної підсистеми

На рисунку 2.4 представлено схему взаємодії програмних модулів.

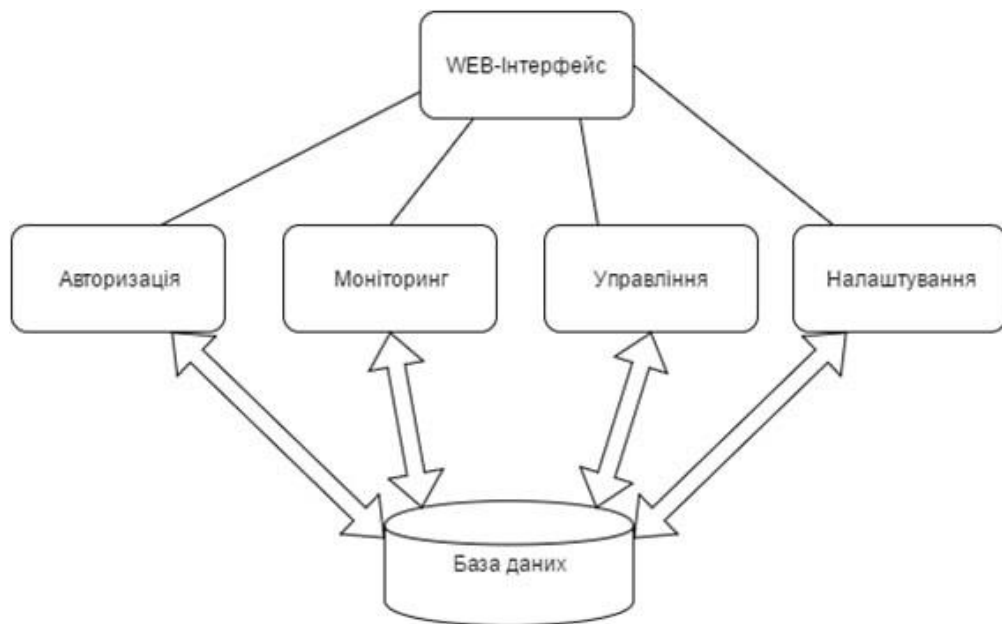


Рисунок 2.4 – Схема взаємодії програмних модулів

З виконання пошукового запиту користувачем починається взаємодія між програмними модулями. Дані запити обробляються сервером та виконують пошук відповідей у базі даних. Якщо відповідь знаходиться в базі — то вона відразу ж надсилається користувачеві назад. Отримана відповідь надсилається до графічного інтерфейсу користувача.

2.4.2 Розробка діаграми компонентів

Діаграма компонентів є важливим інструментом при розробці веб-сайтів. Використовується для зображення структури системи та взаємозв'язків між компонентами, відображає компоненти системи, їх залежності та інтерфейси.

Компоненти — це функціональні частини або модулі системи. Кожен виконує свої функції та має внутрішню реалізацію. Залежності між компонентами вказують на те, як компоненти взаємодіють між собою.

Найчастіше, коли один компонент використовує інший при виконанні певних дій чи отриманні потрібної інформації. Інтерфейси також визначають спосіб співдії з основними компонентами, але включають методи, параметри та повернені значення.

Діаграма допомагає краще спланувати архітектуру системи, виявити проблеми та залежності, а також полегшує налагодження та роботу системи. Вона є корисним інструментом в ході комунікації між розробниками та сприяє ефективній роботі веб-сайту.

На рисунку 2.5 зображено діаграму компонентів системи.

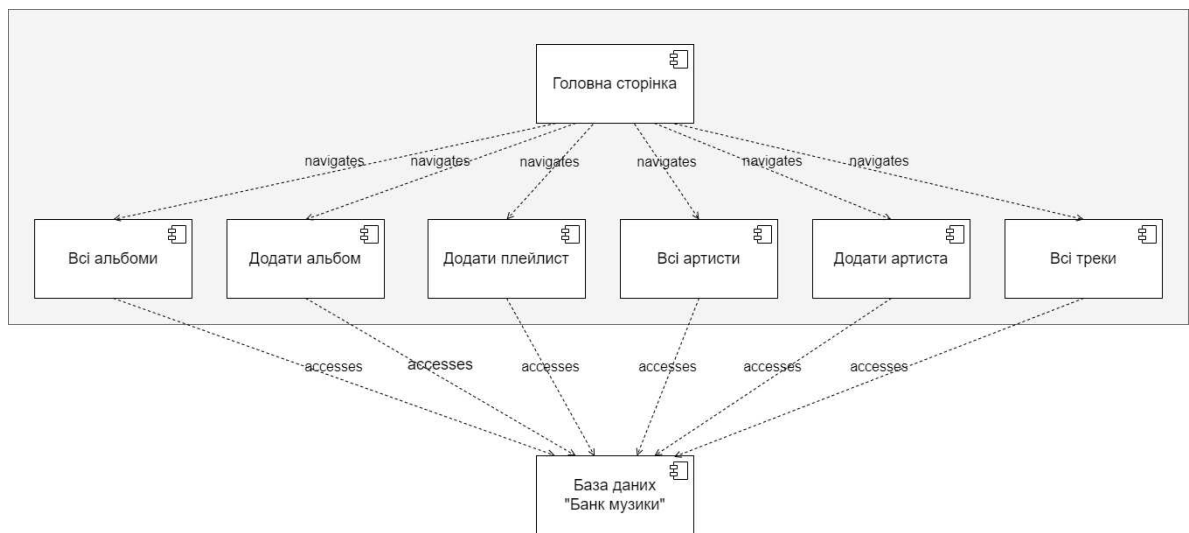


Рисунок 2.5 – Діаграма компонентів

2.4.3 Розробка діаграми переходів станів

Діаграма станів – зображує різні стани, через які проходить об’єкт системи пі час своєї роботи, відображає його поведінку. На цій діаграмі зображено два основних стани – початковий і кінцевий. Початковий стан показується чорною крапкою невеликого діаметру: він зображує стан об’єкта в момент його створення.

Чорною точкою в червоному крузі позначено кінцевий стан: безпосередньо перед зникненням він відповідає за стан об’єкта. Показаний на діаграмі станів може бути один й тільки один початковий стан.

Діаграма переходів станів для гостя зображено на рисунку 2.6.



Рисунок 2.6 – Діаграма переходів станів для гостя

Гостю на головній сторінці сайту потрібно натиснути кнопку “Всі плейлисти”, для того щоб переглянути список всіх плейлистів сайту.

На рисунку 2.7 зображено діаграму переходів станів для користувача.



Рисунок 2.7 – Діаграма переходів станів для користувача

Користувач відкривши сайт може натиснути на кнопку “Реєстрація”, після успішної реєстрації стане доступна вкладка “Всі плейлисти”, натиснувши на неї відкриється список всіх плейлистів. Навпроти кожного плейлиста є дві кнопки “Додати трек” та “Видалити трек”. Обравши потрібну кнопку ми повертаємось до перегляду оновленого списку плейлистів.

Діаграму переходів станів для адміністратора зображено на рисунку 2.8.

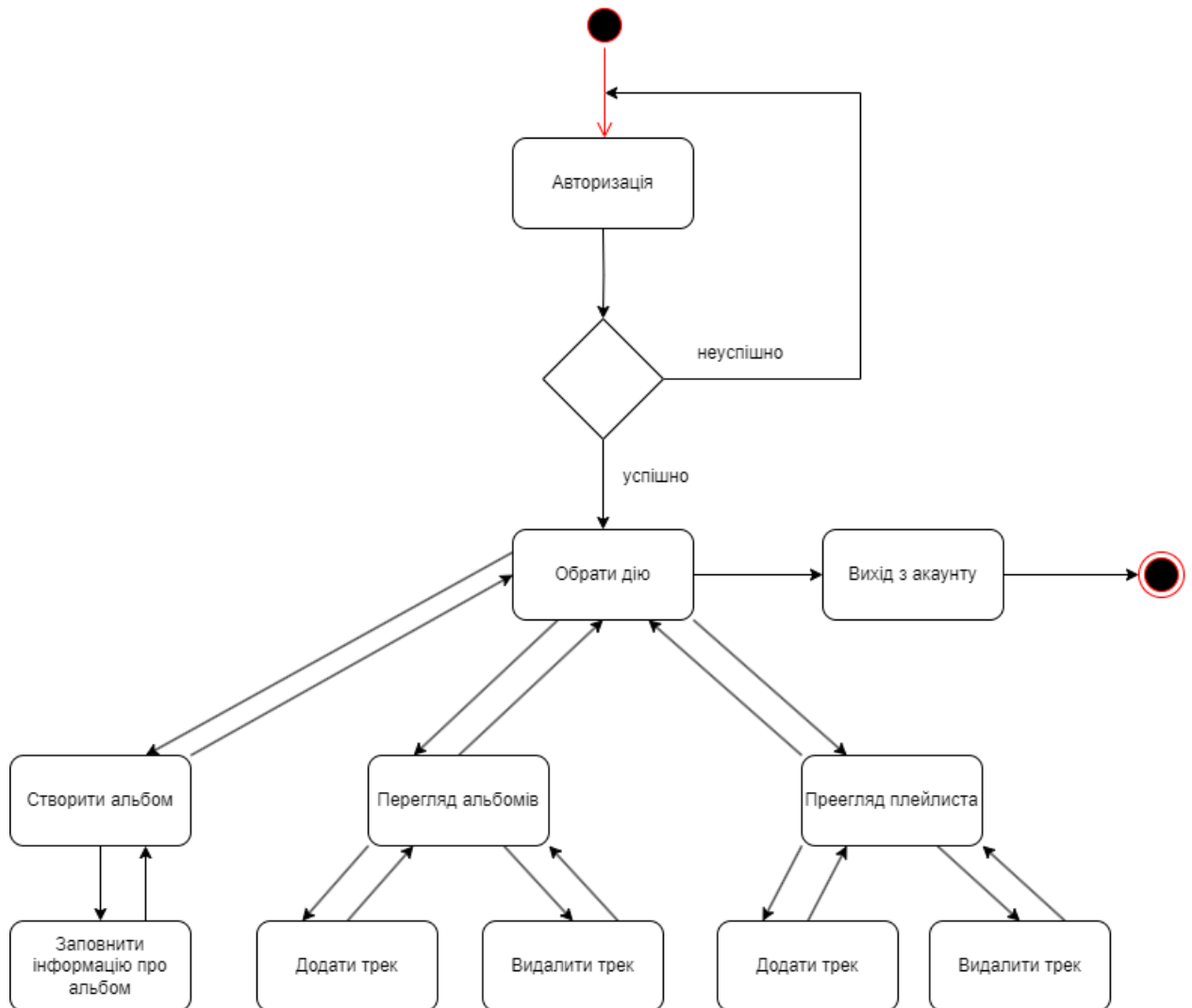


Рисунок 2.8 – Діаграма переходів станів для адміністратора

Адміністратор зайшовши на сайт обирає кнопку “Авторизація”, після успішної авторизації з’являються вкладки, такі як “Додати альбом”, “Всі альбоми” та “Всі плейлисти”. Натиснувши на “Додати альбом”, потрібно заповнити форму даними про альбом та зберегти дані. Після натискання кнопки “Всі альбоми” з’являється перелік всіх альбомів сайту, навпроти

кожного альбому є кнопки “Додати трек” та “Видалити трек”. Обравши потрібну кнопку повертаємось до перегляду оновленого списку альбомів. Перейшовши на вкладку “Всі плейлисти”, та обравши потрібний плейлист бачимо кнопки “Додати трек” та “Видалити трек”. Обравши потрібну кнопку повертаємось до перегляду оновленого списку плейлистів.

2.4.4 Розробка проекту інтерфейсу інформаційної підсистеми

Інтерфейс інформаційної підсистеми складається з невеликої кількості елементів інтерфейсу та з використання обмеженої кількості кольорів дизайну інтерфейсу. Тому відображення таких сторінок можна забезпечити за допомогою стандартних засобів та бібліотек. Створено файл style.css та підключено необхідні бібліотеки, обрано стиль кожного елементу підсистеми. Таким чином, внесенні зміни в код можуть змінити вигляд елементів всієї системи.

Інтерфейс реєстрації в додатку зображено на рисунку 2.9.

Рисунок 2.9 – Інтерфейс сторінки реєстрації

Вигляд розробленого інтерфейсу сторінки для показу всіх треків в плейлисті зображено на рисунку 2.10.

All Tracks In Playlist			
ID Track	Name Track	Duration	Action
1	Future	4:15 min	<button>Видалити</button>
11	Wild Flover	4:34 min	<button>Видалити</button>

Рисунок 2.10 – Інтерфейс сторінки для всіх треків в плейлисті

Найчастіше для додання нової інформації або редагування наявної використовуються спеціальні форми для заповнення. Кожне поле підписується для запобігання некоректного вводу даних.

На рисунку 2.11 зображено інтерфейс редагування інформації про трек.

Update Forms For Tracks	
Name:	Future
Album:	Jack in the Box
Artist:	Agust D
Label:	BigHit
Duration:	4:15 min
Year Edition:	2021
Update track	

Рисунок 2.11 – Інтерфейс сторінки для редагування інформації про трек

Інтерфейс для створення нового альбому зображено на рисунку 2.12.

The image shows a web form titled "Create Album" on a brown background. It has two input fields: "Name:" with the text "Hope World" and "Year:" with the text "2018". Below the fields is a blue button labeled "Create".

Create Album

Name: Hope World

Year: 2018

Create

Рисунок 2.12 – Інтерфейс сторінки створення нового альбому

Для розкриття можливостей перегляду інформації також використовуються активні кнопки дії, такі як видалення, додання треків.

Розміщення активних кнопок зображено рисунку 2.13.

The image shows a table titled "All Artists" on a brown background. The table has 7 columns: ID, Name, Gender, Country, Role, Year of birth, and Action. There are 7 rows of data, each with a "Видалити" (Delete) button and a "Додати трек" (Add track) button in the Action column.

ID	Name	Gender	Country	Role	Year of birth	Action
1	Agust D	male	SK	rapper	1993	Видалити Додати трек
2	Jimin	male	SK	sub	1995	Видалити Додати трек
3	J-Hope	male	SK	rapper	1994	Видалити Додати трек
4	JK	Male	SK	vokal	1997	Видалити Додати трек
5	RM	Male	SK	rapper	1994	Видалити Додати трек
6	V	Male	SK	sub	1995	Видалити Додати трек
7	Jin	Male	SK	vokal	1992	Видалити Додати трек

Рисунок 2.13 – Інтерфейс сторінки всі артисти з активними кнопками

2.4.5 Розробка діаграми розгортання підсистеми

Діаграма розгортання використовується для візуального зображення фізичного розташування компонентів системи або програмного забезпечення

на різних вузлах або серверах. Також надає зручну та зрозумілу модель для відображення взаємодії компонентів системи та їх фізичне розташування.

При розробці допомагає описати архітектору системи та компонентів, котрі використовуються для повноцінного функціонування веб-сайту. Показує, які компоненти розташовано на сервері, девайсі користувача чи на інших вузлах.

Відображає такі елементи:

- демонструє з'єднання між компонентами системи;
- пристрої, якими користуються для доступу до веб-сайту;
- демонструє, як дані передаються між серверам та користувачами;
- вказує, де розташовані різні компоненти системи.

Діаграма розгортання також допомагає краще зрозуміти фізичну архітектуру системи та її компонентів. Надає можливість виявити потенційні проблеми, які можуть виникнути при розгортанні системи, недостатність ресурсів чи з'єднання з мережею. Вона зі свого боку служить основою для планування розгортання системи, визначення основних вимог обладнання, налаштування серверів та модулів.

Діаграма розгортання системи зображено на рисунку 2.14.

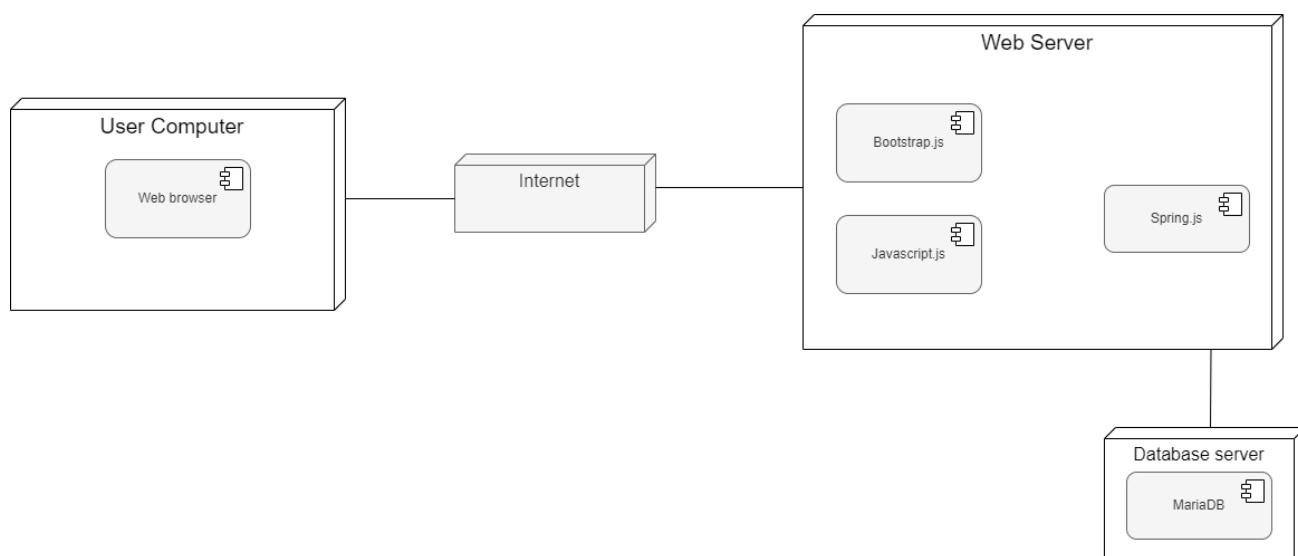


Рисунок 2.14 – Діаграма розгортання системи

РОЗДІЛ III. РЕАЛІЗАЦІЯ ПРОЕКТУ ПІДСИСТЕМИ

3.1 Аргументація вибору засобів розробки

Важливою частиною розробки інформаційної системи є правильний вибір технологій, які використовуються при розробці. Адже при правильному виборі мов програмування, бібліотек, баз даних та інших засобів розробки можна досягти кращого функціонування системи.

Середовищем розробки є IntelliJIDEA, що має дуже зручний інтерфейс та надає безліч можливостей при розробці програм та взаємодію з плагінами.

Для розробки Backend використано мову програмування Java, вона не лише кросплатформна, а й включає ряд бібліотек, які можуть допомогти спростити процес розробки, дозволяє зробити архітектуру проекту більш чіткою і зрозумілою, що є перевагою при розширенні проекту [9].

Перевагами використання мови Java є:

- можливість запускати Java-додатки на будь-якій платформі, що підтримує JVM, без необхідності переробляти код;
- підтримка принципів об'єктно-орієнтованого програмування (ООП), що сприяє організації та логічному підходу до розробки програм;
- вбудовані механізми безпеки, такі як управління доступом і захист пам'яті;
- можливість створювати високопродуктивні додатки, які можуть легко масштабуватися;
- велика екосистема бібліотек та фреймворків, таких як Spring, Apache, які спрощують розробку додатків та скорочують час на реалізацію проєктів.

Всі ці переваги роблять мову програмування Java хорошим вибором для розробки різноманітних веб-сайтів та додатків, від особистих сайтів до великих корпоративних систем, забезпечуючи надійність, безпеку та високу продуктивність.

Спеціально розроблені бібліотеки — фреймворки часто використовуються для полегшення створення архітектури проєкту та використання шаблонів дизайну. Програмний фреймворк — це набір заздалегідь розроблених рішень, які можна використовувати при розробці програмних продуктів, що містять готові модулі архітектури та логіки побудови.

Фреймворк Spring надає повний контроль на структуру додатку, тому можна зосередитися на реалізації проєкту, не турбуючись про архітектуру системи [10]. Технологія Spring дозволяє розробляти всі види додатків мовою Java та підходить для створення як веб-додатків так і JEE-додатків [11].

Для розробки клієнтської частини веб-сайту для управління музичним архівом обрано мови розмітки та стилізації HTML, CSS, JavaScript та Bootstrap, оскільки вони є найпопулярнішими мовами для веб-розробки і досить прості в використанні.

Мова розмітки HTML складається з серії коротких кодів, і які розробник веб-сайту нормалізує у спеціальний файл. Текст зберігається у вигляді HTML-файлу і переглядається за допомогою браузера. Браузер сканує файл, інтерпретує текст у видимій формі і відображає сторінку так, як запланував розробник [12].

Перевагами цієї мови є: легкість вивчення та використання, сумісність веб-сторінок у різних браузерах та пристроях, підтримка усіма браузерами, дозволяє легко створювати гіперпосилання на інші веб-сторінки, підтримка мультимедійних елементів та доступність.

CSS (Cascading Style Sheets - каскадні таблиці стилів) — це мова таблиць стилів. Вона використовується для визначення способу представлення елементів HTML на веб-сторінці, включаючи дизайн, верстку та різноманітність пристроїв з різним розширенням екрану. CSS керує версткою різних веб-сторінок одночасно [13, 14].

JavaScript — одна з найпопулярніших мов програмування в Інтернеті. Вона дозволяє створювати динамічні та інтерактивні веб-сторінки. Існує

незліченна популярних інструментів для веб-розробки, заснованих на JavaScript [15].

Переваги використання JavaScript:

- можливість створювати інтерактивні елементи на веб-сторінках, такі як спливаючі вікна, динамічні фони, анімації та багато іншого;
- підтримка усіма сучасними веб-браузерами;
- висока швидкість виконання скриптів та реагування на дії користувача;
- простий синтаксис;
- підтримка серверного програмування та кросплатформеність.

Всі ці переваги роблять JavaScript незамінним інструментом для створення сучасних веб-сайтів та додатків, що відповідають вимогам користувачів та розробників.

Bootstrap – це фреймворк з відкритим вихідним кодом, над яким працюють мільйони розробників, створюючи найкращі та передові функції. Він також включає підтримку нових браузерів, а також нових версій браузерів. Веб-дизайнери та розробники часто використовують Bootstrap у своїх проектах. Він використовується для створення адаптивних веб-дизайнів, які виглядають однаково на всіх розмірах екранів (смартфонів, планшетів, ноутбуків чи ПК) [16].

Базу даних розроблено за допомогою СУБД MariaDB — система, що дозволяє вільно та надійно працювати та адмініструвати реляційні бази даних [17]. Основними перевагами використання є:

- швидке виконання запитів і обробка великих обсягів даних;
- підтримка різних механізмів зберігання даних (наприклад, InnoDB, Aria);
- відкрите програмне забезпечення;
- розширені функції безпеки, покращений механізм аутентифікації, шифрування даних та управління доступом до бази даних;
- можливість резервного копіювання та відновлення даних;

- підтримка багатьох інструментів для управління базами даних, включаючи графічні інтерфейси (найпопулярніший phpMyAdmin).

phpMyAdmin – це веб-додаток з відкритим вихідним кодом, написаний мовою програмування PHP. Він надає зручний веб-інтерфейс для керування базами даних MySQL або MariaDB. Dodatok дозволяє керувати базами даних з різних веб-браузерів, виконувати запити на мові SQL, отримувати та редагувати дані зберігаючи в табличному форматі. Завдяки простому інтерфейсу можна керувати базами даних MySQL на іншому комп'ютері без необхідності вводити SQL-команд. Значна кількість українських провайдерів використовують цю програму як панель для управління базами даних, пропонуючи надати можливість адміністрування базами даних [18].

3.2 Розробка структури програмного забезпечення

Головна сторінка веб-застосунку для управління музичним архівом для різних типів користувачів відрізняється. Для користувача, що має права адміністратора, є кнопки та форми для редагування всієї інформації про треки, альбоми, плейлисти, додання нових артистів чи груп, студій звукозапису. Має доступ до списку всіх користувачів системи, може переглядати та редагувати їх параметри.

Для зареєстрованого користувача наявний список всіх треків плейлистів, альбомів. За потреби він може створювати власні плейлисти за настроєм та тематикою.

Гість, що не зареєстрований в системі має обмежені можливості. Він може лише переглядати створені колекції або зареєструватися в системі для збільшення можливостей.

Інтерфейс меню навігації забезпечує користувачам наглядне та інтуїтивно зрозуміле представлення структурованої інформації, яка розміщена в системі. Забезпечує швидкий і зручний перехід до функціональних

можливостей. Навігаційні елементи забезпечують однозначне розуміння змісту сторінки. Всі елементи підписані відповідно до змісту.

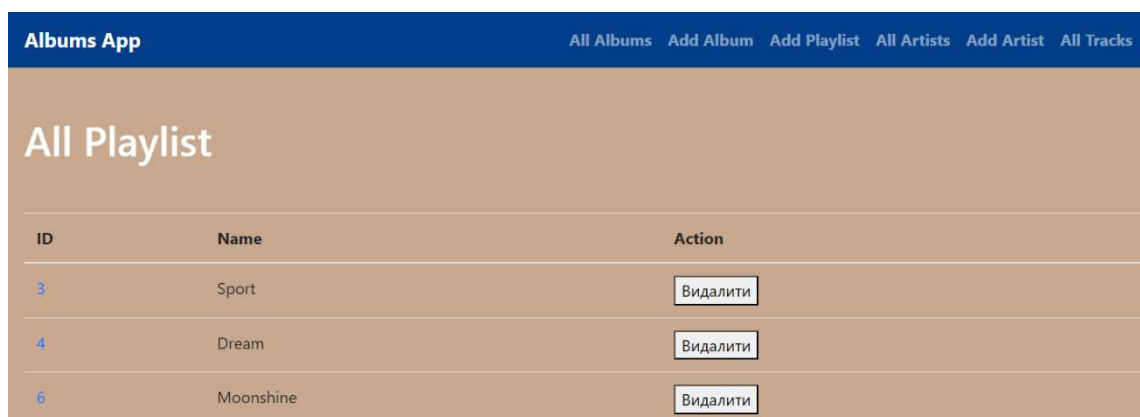
Для забезпечення інтуїтивно зрозумілого та простого дизайну, проєкт виконано в одному стилі. Використано сучасні кольорові гами з переважаючими коричневими, синіми та білими відтінками. Обрано шрифти, які комфортні та зручні для читання. Всі елементи веб-застосунку розташовано на певній відстані.

Текст програми представлено в додатку Б.

3.3 Результати розробки програмного забезпечення

У результаті виконання кваліфікаційної роботи реалізовано заплановану мету проєкту. Розроблено основний функціонал: головну сторінку, сторінки реєстрації та авторизації, створення та редагування колекцій, редагування інформації про виконавців, студії звукозапису.

При першому вході на сайт одразу з'являється головна сторінка з основними вкладками навігації, які зображено на рисунку 3.1.



The screenshot shows the 'Albums App' interface. At the top is a dark blue navigation bar with the app name on the left and links to 'All Albums', 'Add Album', 'Add Playlist', 'All Artists', 'Add Artist', and 'All Tracks' on the right. Below the navigation bar is a large tan-colored section titled 'All Playlist'. Inside this section is a table with three columns: 'ID', 'Name', and 'Action'. The table contains three rows of data, each with a blue 'ID' value and a 'Видалити' (Delete) button in the 'Action' column.

ID	Name	Action
3	Sport	Видалити
4	Dream	Видалити
6	Moonshine	Видалити

Рисунок 3.1 – Головна сторінка сайту

Для входу на сайт з'являється вікно для швидкої реєстрації чи авторизації, яке зображено на рисунку 3.2.

Реєстрація

Ім'я По-батькові Прізвище

Електронна пошта

Номер телефону Пароль

☐ Показати пароль

Реєстрація

Рисунок 3.2 – Сторінка реєстрації

Після натискання на будь-який плейлист відкривається сторінка про Всі треки в плейлисті, яку зображено на рисунку 3.3.

Albums App

All Albums

Add Album

Add Playlist

All Artists

Add Artist

All Tracks

All Tracks In Playlist

ID Track	Name Track	Duration	Action
3	on the street	5 min	Видалити
8	3D (feat. Jack Harlow)	3:50 min	Видалити

Рисунок 3.3 – Сторінка всіх треків плейлиста

При переході на вкладку Всі альбоми, де є перелік всіх музичних альбомів на сайті.

Сторінку Всі альбоми зображено на рисунку 3.4.

All Albums			
ID	Name	Year	Action
2	FACE	2023	Видалити
3	Jack in the Box	2022	Видалити

Рисунок 3.4 – Всі альбоми

Обравши трек та натиснувши на нього відчиняється сторінка інформації про обраний трек.

Сторінку інформації про трек відображено на рисунку 3.5.

Albums App						
All Albums Add Album Add Playlist All Artists Add Artist All Tracks						
Track						
ID	Name	Duration	Album	Artist	Label	Action
8	3D (feat. Jack Harlow)	3:50 min	Golden	JK	BigHit	Видалити

Рисунок 3.5 – Сторінка інформації про трек

Нижче розташована форма для додання треку в потрібний альбом.

Форму для додання треку в альбом зображено на рисунку 3.6.

Add to playlist

Playlist:

Sport

Add Track to Playlist

Рисунок 3.6 – Форма для додання треку до плейлиста

Внизу сторінки розташована форма для редагування відомостей про трек, яку зображено на рисунку 3.7.

Update Forms For Tracks

Name:

3D (feat. Jack Harlow)

Album:

Golden

Artist:

JK

Label:

BigHit

Duration:

3:50 min

Year Edition:

2023

Update track

Рисунок 3.7 – Форма для редагування треку

Для адміністратора системи доступна сторінка Створення альбому. Сторінка додання альбому зображена на рисунку 3.8.

The image shows a web form titled "Create Album" on a light brown background. The form contains two text input fields: "Name:" and "Year:". Below the "Year:" field is a blue button labeled "Create".

Create Album

Name:

Year:

Create

Рисунок 3.8 – Створити альбом

Також доступна сторінка для додання нового виконавця та групи. Для заповнення даних використовується форма з підписаними полями для зручності у використанні.

На рисунку 3.9 зображено сторінку додання виконавця.

The image shows a web form titled "Create Artist" on a light brown background. The form contains several fields: "Name:" (text input), "Gender:" (dropdown menu with "Male" selected), "Country:" (dropdown menu with "SK" selected), "Role:" (dropdown menu with "vokal" selected), "group:" (dropdown menu with "BP" selected), and "Year:" (text input). A blue button labeled "Create" is located at the bottom left of the form.

Create Artist

Name:

Gender:

Country:

Role:

group:

Year:

Create

Рисунок 3.9 – Сторінка додання артиста

Сторінку з переліком всіх виконавців зображено на рисунку 3.10.

All Artists							
ID	Name	Gender	Country	Role	Year of birth	Action	
1	Agust D	male	SK	rapper	1993	Видалити	Додати трек
2	Jimin	male	SK	sub	1995	Видалити	Додати трек
3	J-Hope	male	SK	rapper	1994	Видалити	Додати трек
4	JK	Male	SK	vokal	1997	Видалити	Додати трек
5	RM	Male	SK	rapper	1994	Видалити	Додати трек
6	V	Male	SK	sub	1995	Видалити	Додати трек
7	Jin	Male	SK	vokal	1992	Видалити	Додати трек
8	Suga	Male	SK	vokal	1994	Видалити	Додати трек

Рисунок 3.10 – Всі артисти

Сторінку переліку треків зображено на рисунку 3.11.

All Tracks			
ID	Name	Duration	Album
1	Future	4:15 min	Jack in the Box
2	Into	4 min	Jack in the Box
3	on the street	5 min	Jack in the Box
4	Set Me Free	3 min	FACE
5	Fack-off	4 min	FACE
6	D-Day	3 min	D-Day
7	Haegeum	3 min	D-Day
8	3D (feat. Jack Harlow)	3:50 min	Golden
9	Super Tuna	1:13 min	슈퍼 참치
10	Love Me Again	3:18 min	Layover
11	Wild Flover	4:34 min	Indigo

Рисунок 3.11– Всі треки

Інструкцію користувача розроблено і представлено в додатку В.

РОЗДІЛ IV. ОХОРОНА ПРАЦІ

4.1 Загальні питання охорони праці

Охорона праці - це система правових і соціально-економічних, організаційно-технічних і санітарно-гігієнічних заходів, спрямованих на забезпечення безпечних умов праці.

На підприємствах загальне керівництво охороною праці покладено на керівників і головного інженера, а безпосередній контроль і організацію цієї роботи здійснюють головний інженер та його заступник по охороні праці.

Основні функції відділу охорони праці:

- контроль за дотриманням керівниками цехів, відділів та інших структурних підрозділів чинного законодавства, стандартів і нормативних актів з охорони праці;
- виконання приписів наглядових органів та наказів по підприємству;
- розробка заходів щодо створення здорових та безпечних умов праці в цехах, відділах і на об'єктах, що будуються;
- організація проведення вступних інструктажів для нових працівників;
- контролю своєчасності та якості проведення інструктажів на робочих місцях;
- участь у роботі комісій з розслідування причин нещасних випадків та аварій, перевірка будівельних проектів та перевірка знань технічних та кваліфікованих робітників щодо управління гігієною на підприємстві;
- розгляд проектів будівництва, капітального ремонту цехів та обладнання.
- прийняття рішень про капітальний ремонт обладнання;
- сертифікація гігієнічного та технічного стану робочих місць;
- реєстрація потерпілих від нещасних випадків на виробництві; аналіз нещасних випадків на виробництві;
- підготовка звітів про нещасні випадки на виробництві;
- реалізація заходів щодо поліпшення умов праці;

- організація зібрань та стендів з охорони праці.

Покращення умов праці сприяє:

- збільшенню продуктивності виробництва;
- збереженню та швидкому відновленню високого рівня працездатності людини:

- зменшенню кількості нещасних випадків на виробництві;
- зменшенню кількості професійних захворювань.

4.2 Структура управління питаннями охорони праці на підприємстві

Охорона праці на підприємствах є важливим питанням для забезпечення безпеки та здоров'я працівників на підприємстві. Для того, щоб ефективно управляти цими питаннями, підприємства повинні мати відповідні системи управління, які б забезпечували впровадження та дотримання стандартів і правил охорони праці та промислової безпеки.

Основними компонентами системи управління охороною праці на підприємствах є наступні:

1) Комітет (або відділ) з охорони праці: комітет (або відділ) з охорони праці - це організаційна частина, до складу якої входять представники керівництва, профспілок та експерти з охорони праці. Комітет відповідає за обговорення та вирішення питань охорони праці на підприємстві, розгляд звітів з безпеки, аналіз нещасних випадків та розробку заходів щодо їх запобігання.

2) Безпосередня відповідальність за дотримання правил охорони праці полягає на керівник кожного відділу або дільниці несе відповідальність за забезпечення безпеки та здоров'я працівників у своєму відділі. Він перевіряє дотримання вимог охорони праці, забезпечує необхідним обладнанням та засобами захисту, організовує інформаційні зустрічі та контролює дотримання правил безпеки.

3) Робоча група з охорони праці — це група працівників, які спеціалізуються на охороні праці в певному секторі або виробничому процесі. Вона проводить аналіз ризиків, розробляє та впроваджує профілактичні заходи, а також надає консультації та рекомендації з питань безпеки.

4) Офіцери з охорони праці або інженери з охорони праці — це фахівці, відповідальні за організацію та контроль за дотриманням законодавства про охорону праці на підприємстві. Вони проводять аналіз ризиків, розробляють програми та плани з охорони праці, навчають працівників, проводять перевірку знань та аудит.

5) Комітет з розслідування нещасних випадків — комітет створюється, коли на підприємстві стається серйозний нещасний випадок або інцидент. Комітет з розслідування нещасних випадків розслідує обставини нещасного випадку, визначає причини та формулює рекомендації щодо запобігання нещасним випадкам у майбутньому.

4.3 Шкідливі та небезпечні фактори, що супроводжують працю програміста

Попри те, що робота програміста зазвичай пов'язана з високим рівнем комфорту та відносною безпекою порівняно з іншими професіями, існують деякі шкідливі та небезпечні фактори, що можуть впливати на здоров'я програміста.

Основними з них є:

- підвищений ризик ураження електричним струмом;
- підвищений ризик виникнення пожежі;
- шкідливе електромагнітне випромінювання;
- недостатнє освітлення.

Електрика - одна з найпоширеніших професійних небезпек. Вплив електричного струму на живі тканини є багатограним та унікальним.

Електричний струм, проходячи через тіло людини, чинить термічну, електролітичну, механічну та біологічну дію.

Термічна дія електричного струму проявляється у вигляді опіків на окремих ділянках тіла та високих температур нагрівання органів на шляху проходження струму і викликає значні функціональні розлади. Електролітична дія проявляється у розкладанні органічних рідин, у тому числі крові, та порушенні їх фізико-хімічного складу.

Механічна дія електричного струму викликає розшарування і розрив тканин організму в результаті електродинамічного ефекту, що призводить до миттєвого утворення з рідини вибухонебезпечних парів. Біологічна дія електричного струму проявляється в стимуляції та руйнуванні біологічних тканин організму і порушенні внутрішніх біологічних процесів.

Допустимий струм - це струм, який людина може відпустити з електричного кола. Його величина залежить від швидкості проходження струму через тіло людини. Він становить 2 мА, якщо час впливу більше 10 секунд, і 6 мА, якщо час впливу менше однієї секунди. Струм, при якому потерпілий не може вийти з активної зони, називається струмом утримання і становить від 10 до 20 мА.

Змінний струм є більш небезпечним, ніж постійний, а постійний струм не є безпечнішим при високій напрузі.

Більшість пожеж у житлових та офісних приміщеннях виникають через порушення технічного режиму. На жаль, це трапляється настільки часто, що держава розробила спеціальний документ, який описує основи протипожежного захисту. Це стандарти: ДСТУ 12.1.004-76 "Пожежна безпека" і ДСТУ 12.1.010-76 "Вибухобезпека".

Заходи пожежної безпеки поділяються на організаційні, технічні, системні та експлуатаційні.

Організаційні заходи включають в себе правильну експлуатацію техніки і транспорту, належне утримання будівель і територій, навчання робітників і службовців правилам пожежної безпеки, організацію добровільних пожежних

дружин, пожежно-технічних комісій, видання наказів щодо поліпшення пожежної безпеки.

Технічні заходи включають дотримання правил проектування будівель, монтажу електропроводки та обладнання, опалення, вентиляції, освітлення та розміщення обладнання.

Нормативні заходи включають заборону куріння у невстановлених місцях та використання вогню, наприклад, зварювання, у пожежонебезпечних зонах.

Експлуатаційні заходи включають своєчасний профілактичний огляд, ремонт і випробування технологічного обладнання.

Шкідливий вплив електромагнітного випромінювання від комп'ютерних мереж та системного обладнання є результатом взаємодії людини як біологічного об'єкта з навколишнім середовищем. Докладаються зусилля для нейтралізації цих ефектів, наприклад, за допомогою спеціальних екранів і чохлів для моніторів, а також шляхом організації робочого простору таким чином, щоб співробітники не знаходилися за екранами. Ергономічні вимоги також слід враховувати при проектуванні інтер'єрів приміщень. Зокрема, необхідно переконатися, що клавіатура знаходиться в правильному положенні відносно рук працівника, а монітор - на правильній відстані від очей (не менше 80 см).

Основним завданням промислового освітлення є підтримання на робочих місцях рівнів освітленості, що відповідають характеру зорової роботи. Збільшення освітленості робочої поверхні підвищує яскравість об'єкта, покращує видимість, прискорює ідентифікацію деталей і впливає на продуктивність праці.

При налаштуванні промислового освітлення яскравість робочої поверхні і навколишніх предметів повинна бути рівною. Переведення погляду з яскравої ділянки на тьмяну змушує око перелаштовуватися, що призводить до втоми очей, а це знижує продуктивність праці. Для підвищення рівномірності природного освітлення у великих приміщеннях застосовують

композиційне освітлення. Світлі стелі, стіни та обладнання забезпечують рівномірну яскравість в полі зору працівника.

Виробниче освітлення повинно запобігати утворенню жорстких тіней в полі зору працівника. Жорсткі тіні спотворюють розмір і форму предметів, підвищуючи втому і знижуючи продуктивність праці. Рухомі тіні особливо шкідливі і можуть призвести до травм.

4.4 Ергономічні вимоги до робочого місця при роботі за комп'ютером

Ергономічні вимоги в організації робочого місця відіграють важливу роль у забезпеченні комфорту, безпеки та ефективності працівників. Правильно організоване робоче середовище сприяє покращенню загального самопочуття, запобіганню травм та захворювань, а також підвищує продуктивність праці.

Однією з найважливіших ергономічних вимог є правильне розташування робочих інструментів та обладнання.

Монітор комп'ютера слід розташовувати на рівні очей, приблизно на відстані 50-70 см від очей оператора, щоб зменшити навантаження на очі та шию, а клавіатуру та мишу слід розміщувати так, щоб зап'ястя були прямими та рівними. Миша повинна бути розміщена поруч з клавіатурою і на тій же висоті, щоб уникнути тиску на зап'ястя і кисті рук.

Інші ергономічні вимоги включають регульовані стільці з підлокітниками, щоб працівники могли підтримувати зручну позу під час тривалої роботи за комп'ютером. Стільці повинні регулюватися за висотою, кутом нахилу та підлокітниками. Правильно підібраний стілець допомагає запобігти перенапруженню спини та шиї. Також важливо, щоб на робочому столі було достатньо місця для всіх необхідних предметів і він був достатньо великим, щоб не обмежувати свободу рухів працівника.

Особливу увагу слід також приділяти освітленню робочих місць. Робочі місця повинні бути освітлені достатнім природним і штучним світлом, а інтенсивність світла повинна регулюватися.

Промислове освітлення не повинно відкидати різкі тіні в поле зору оператора. Щільні тіні спотворюють розмір і форму об'єктів, підвищуючи втому і знижуючи продуктивність. Рухомі тіні особливо шкідливі і можуть призвести до травм.

Для поліпшення видимості об'єктів, що знаходяться в полі зору працівника, бажано, щоб на них не було відблисків, викликаних прямим або відбитим сонячним світлом. Відблиски - це збільшення яскравості світлої поверхні, що викликає порушення зору (сліпоту), тобто зниження видимості об'єктів. Відблиски можна зменшити, зменшивши яскравість джерела світла або вибравши відповідний тип світильника.

Коливання освітленості на робочому місці, наприклад, через різкі перепади напруги в мережі, викликають надмірну адаптацію очей, що призводить до значної втоми.

Промислове освітлення повинно забезпечувати необхідний спектральний склад світлового потоку. Освітлювальні установки повинні бути зручними, простими в експлуатації, довговічними, відповідати естетичним вимогам і вимогам електробезпеки, а також не повинні бути причиною вибухів і пожеж.

Штучне освітлення використовується в темний або світлий час доби, коли природне освітлення неприйнятне або недостатнє за нормальних умов освітлення.

Периферійне обладнання, таке як принтери та сканери, має бути розміщене в межах легкої досяжності.

Слід також звернути увагу на робочий стіл, який повинен бути достатньо великим, щоб вмістити все необхідне обладнання та документи, залишаючи вільний простір для роботи за комп'ютером.

Інші ергономічні вимоги включають наявність спеціальної підставки для монітора, яка дозволяє регулювати висоту і кут нахилу монітора для комфортного перегляду. Також важливо, щоб працівники мали можливість робити регулярні перерви від монітора для розминки і розтяжки, що запобігає перенапруженню м'язів і знижує втому.

Всі ці міркування можна врахувати, щоб створити робоче середовище, яке сприятиме здоров'ю, комфорту та продуктивності працівників. Ергономічні вимоги є важливим фактором для будь-якого робочого місця, яке хоче досягти оптимальних результатів і забезпечити комфорт для співробітників.

4.5 Нормативні вимоги до робочої площі на одного працюючого за комп'ютером

З розвитком сучасних технологій та використанням комп'ютерів у всіх сферах життя, створення комфортних та ефективних умов праці стало важливим викликом. Одним з найважливіших питань є правильна організація робочого простору працівників.

Законодавчі вимоги до робочого простору на одного користувача комп'ютера є важливим аспектом забезпечення здорових і безпечних умов праці. Вони покликані забезпечити комфортне та ергономічне робоче середовище, підвищити продуктивність праці та знизити ризик отримання травм і захворювань.

Основні нормативні вимоги до робочого простору на одного користувача комп'ютера такі:

- Згідно з основними стандартами та рекомендаціями, мінімальні розміри робочого столу повинні забезпечувати достатній простір для комфортного розміщення комп'ютерного обладнання (монітор, клавіатура, миша, додаткові монітори та принтери), а також додаткових інструментів і

документів. Загалом, рекомендується, щоб робочі місця працівників були не менше 120 см завдовжки та 80 см завширшки.

- Робочі місця повинні забезпечувати достатній простір для вільного переміщення працівників до комп'ютера. Це означає, що між робочим столом та іншими меблями має бути достатньо місця для комфортного пересування.

- Важливо враховувати ергономічні принципи при структуруванні робочого місця. Це включає в себе правильне розміщення комп'ютерного обладнання, необхідну підтримку спини і ніг, яку забезпечують стільці, а також можливість регулювати висоту столів і стільців, щоб забезпечити зручну позу для працівника.

- Робочі місця повинні мати належне освітлення та вентиляцію для забезпечення комфортних і безпечних умов праці.

Ще однією важливою частиною нормативних вимог є забезпечення електробезпеки на робочому місці. Це включає такі аспекти, як правильне розташування електричних розеток і кабелів, використання безпечного електрообладнання та перевірка високої якості джерел живлення. Крім того, важливо забезпечити належне заземлення для захисту від потенційної небезпеки ураження електричним струмом.

Важливо також враховувати індивідуальні потреби працівників, наприклад, регульовану висоту столів і стільців, наявність підставки для ніг, додаткове освітлення та наявність фільтрів проти блисків на екранах комп'ютерів. Такий підхід допомагає підтримувати високий рівень комфорту та продуктивності праці, а також знижує ризик травматизму та професійних захворювань.

ВИСНОВОК

У даній кваліфікаційній роботі розглянуто комплекс питань, пов'язаних зі створення ефективної та зручної системи для управління музичними колекціями.

Метою роботи було підвищення продуктивності роботи музичної установи за рахунок розробки веб-сайту. Веб-сайт, забезпечить користувачам можливість організації та каталогізації музичних колекцій за різними критеріями, зручний та авторизований доступ для даних та надійне зберігання музичних записів та метаданих.

В ході роботи було проведено огляд існуючих системи управління різними видами архівів, визначено їх переваги та недоліки. Це в свою чергу надало можливість сформулювати вимоги для проєкту, врахувати бажаний функціонал та вимоги користувачів.

В результаті проєктування системи розроблено архітектуру веб-сайту та базу даних для зберігання інформації та метаданих. Це забезпечило основу для реалізації всіх функціональних можливостей системи.

Реалізовано основний функціонал системи, включаючи визначення метаданих музичних творів, створення та управління колекціями, а також інтерфейс користувача з можливістю реєстрації на сайті.

Програма веб-сайту розроблена мовами Java та JavaScript, з використанням сучасних технологій, таких як мова розмітки HTML, таблиця стилів CSS, також фреймворків та бібліотек для серверної та клієнтської частини. Для зберігання даних використано СУБД MariaDB.

Результатом виконаної кваліфікаційної роботи, є веб-сайт, що відповідає поставленим задачам та є ефективним інструментом для управління музичним архівом. Він дозволяє користувачам організовувати свої музичні колекції, швидко знаходити необхідну інформацію та забезпечує надійне зберігання даних.

Розроблений веб-сайт може значно полегшити роботу користувачів, стрімінгових сервісів, радіостанцій, культурних установ, приватних установ

чи медіа-платформ. Подальший розвиток проєкту може включати збільшення функціоналу, інтеграцію з іншими сервісами чи платформами, покращення інтерфейсу користувача для ще більшого розширення можливостей системи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Мазур О. Л. Музичні радіозаписи як об'єкти архівного зберігання. *Бібліотекознавство. Документознавство. Інформологія*. 2020. № 3. С. 40–46.
2. Бугаєва, О. В. "Музично-теоретична бібліотека імені К. Стеценка в архівних документах Музичного товариства імені М. Леонтовича." *Рукописна та книжкова спадщина України* 8 (2003): 208-215.
3. Everything you need to know about digital asset management. URL: <https://www.veritone.com/blog/media-asset-management-mam/> (дата звернення: 15.02.2024).
4. Юрченко, Ольга, and Леся Халецька. "ISO 15836: 2003:«Інформація та документація. Набір елементів метаданих Дублінське ядро»: створення та практики застосування." *Society. Document. Communication. Соціум. Документ. Комунікація* 15 (2022): 349-382. URL: <https://doi.org/10.31470/2518-7600-2022-15-349-382>
5. Understanding digital audio. URL: <https://www.sony.ua/electronics/support/understanding-digital-audio> (дата звернення: 16.03.2024).
6. WordPress. URL: <https://astwellsoft.com/ru/blog/cms/wordpress.html> (дата звернення: 16.03.2024).
7. Entity Relationship Diagram (ERD). URL: <https://www.smart-draw.com/entityrelationship-diagram/> (дата звернення: 16.03.2024).
8. H. K. Al-Masree, Extracting Entity Relationship Diagram (ERD) From Relational Database Schem. *International Journal of Database Theory and Application*. 8, 15–26 (2015).
9. Коли та чому Java використовується для розробки? URL: <https://wezom.com.ua/ua/blog/kogda-i-pochemu-java-ispolzuesja-dlja-razrabotki-prilozhenij> (дата звернення: 21.03.2024).
10. Офіційний сайт Spring Framework. URL: <https://spring.io/> (дата звернення: 05.04.2024).

11. Java EE — Spring Framework. URL: <https://javarush.com/ua/groups/posts/uk.3546.jsho-potrбно-znati-pro-spring-storja-pojavi-kljuhov-modul-porvnjannja-z-java-ee> (дата звернення: 07.04.2024).
12. Структура HTML. URL: <http://weblib.com.ua/blog/struktura-html> (дата звернення: 08.04.2024).
13. Pfaffenberger HTML, XHTML and CSS. User's Bible / Pfaffenberger 45 et al. - М.: Williams; Edition 3, 2015. - 752 p.
14. Тіттел, Ед HTML, XHTML і CSS для користувачів / Ед Тіттел , Джефф Ноубл. - М.: Діалектика, 2013. - 400 с.
15. Крокфорд Дуглас. JavaScript. Сильні сторони, 2016. — 176 с.
16. Debolt HTML and CSS. Sharing / Debolt, Virginia. - М.: NT Press, 2013.
17. Документація MySQL: веб-сайт. URL: <https://dev.mysql.com/doc/> (дата звернення: 13.05.2024).
18. Документація phpMyAdmin: веб-сайт. URL: <https://www.phpmyadmin.net/docs/> (дата звернення: 15.04.2024).

ДОДАТОК А –Технічне завдання

Вступ

Завданням даної дипломної роботи є розробка інформаційної системи для управління музичним архівом. В таблиці 1 представлено загальні відомості про систему.

Таблиця 1 — Загальні відомості про систему

Повна назва системи	Веб-сайт для управління музичним архівом
Умовне позначення системи	Веб-сайт
Розроблювач	Терещенко Алла Сергіївна
Замовник	Кафедра інформаційних управляючих систем та технологій

1. Підстави для розробки

Веб-сайт розробляється на підставі завдання для дипломного проектування за спеціальністю 122 – «Комп'ютерні науки».

2. Призначення розробки

Об'єкт автоматизації: музична установа.

Основними завданнями музичної установи:

- організація та каталогізація музичних колекцій;
- додання супутньої інформації про треки;
- створення альбомів та плейлистів;
- управління музичними колекціями користувачів.

3. Мета створення системи

Основною метою створення веб-сайту для управління музичним архівом є підвищення продуктивності роботи музичної установи за рахунок розробки веб-сайту. Легкість та зручність у використанні, інтуїтивно зрозумілий інтерфейс, наявність інструментів управління колекціями та даними.

4. Вимоги до програми

4.1 Вимоги до функціональних характеристик

ІС, що буде спроектована, повинна виконувати наступні завдання:

- здійснювати вхід користувача в систему після коректного введення даних входу;
- виконувати операції додавання, редагування, видалення та перегляду даних;
- додавання записів у всі наявні колекції та створення нових;
- виконувати наступні запити:
 - Перегляд даних про альбоми (які і скільки альбомів наявні);
 - Перегляд даних про плейлисти (список плейлистів та кнопки управління);
 - Перегляд даних про трек (в даному запиті буде проглядатися, вся інформація про трек);
 - Перегляд даних про виконавців;
 - Перегляд списку користувачів системи;
 - Перегляд списку треків в плейлисті;
 - Перегляд списку треків в альбомі;
 - Пошук потрібного треку;
 - Пошук альбому за назвою.

Для введення (редагування) даних потрібно створити діалогові форми, які дозволять користувачеві вводити (редагувати) інформацію. Для зручності роботи користувачів необхідно розробити відповідні інтерфейси користувача.

4.2 Вимоги до надійності

При проектуванні повинні бути використані типові проектні рішення там, де це можливо. На стадії технічного проектування повинні прийматися рішення, що зменшують трудомісткість експлуатації системи.

4.3 Умови експлуатації

Веб-сайт розроблено на мові програмування Java та JavaScript в якості СУБД використано MariaDB.

4.4 Вимоги до інформаційної та програмної сумісності

Клієнтський комп'ютер повинен використовувати операційні системи (ОС) Microsoft Windows /8/10/11 .

Вимоги для роботи веб-додатку:

Наявність стабільного підключення до глобальної мережі «Інтернет» на робочому місці клієнта / співробітника / тестового користувача зі швидкістю не менше 256 Кбіт / сек (рекомендується 1 Мбіт / с і більше).

Вимоги до сервера:

- Процесор з частотою 2.4 Гц або більш потужний;
- Для підвищення швидкодії рекомендується 2-х процесорна платформа;
- Оперативна пам'ять 1GB, рекомендується 4Gb.

4.5 Вимоги до технічного забезпечення

Технічні вимоги до клієнтських комп'ютерів визначаються вимогами операційної систем.

5. Вимоги до програмної документації

Усі регламентуючі документи є елементами веб-сайту, які безпосередньо входять до цього. Регламент системи описує правила її ефективного використання. В таблиці 2 наведено склад документів.

Таблиця 2 — Склад документів організаційного забезпечення

№	Назва	Стадія
1	Опис організаційної структури ППГІА	Технічне проектування
2	Керівництво користувача	Реалізація системи

6. Стадії та етапи розробки

В таблиці 3 представлено стадії та етапи по створенню веб-сайту.

Таблиця 3 — Стадії та етапи робіт по створенню веб-сайту

№	Стадія	Документи
1	Аналіз предметної області	Звіт з описом предметної області
2	Системне проектування	Технічне завдання
3	Ескізне проектування	Ескізний проект
4	Технічне проектування	Технічний проект
5	Робоче проектування	Робочий проект

ДОДАТОК Б – Програмний код

Повний код проєкту можна переглянути за посиланням:

<https://github.com/Alla183/CourseWork>

Текст програми для управління музичним архівом представлено на лістингах 1— 10.

Лістинг 1 — Головна сторінка. Album.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>All Albums</title>
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/boo
tstrap.min.css"
        integrity="sha384-
ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T
" crossorigin="anonymous">
    <link rel="stylesheet" href="/css/style.css"
type="text/css">
</head>
<body>
<nav class="navbar navbar-expand-lg navbar-dark">
    <div class="container">
        <a class="navbar-brand" href="/">Albums App</a>
        <button class="navbar-toggler" type="button" data-
toggle="collapse" data-target="#navbarNav"
            aria-controls="navbarNav" aria-expanded="false"
aria-label="Toggle navigation">
            <span class="navbar-toggler-icon"></span>
        </button>
        <div class="collapse navbar-collapse" id="navbarNav">
            <ul class="navbar-nav ml-auto">
                <li class="nav-item">
                    <a class="nav-link" href="/albums">All
Albums</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="/create">Add
Album</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="/createpl">Add
Playlist</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="/artists">All
Artists</a>
```

```

        </li>
        <li class="nav-item">
            <a class="nav-link" href="/createar">Add
Artist</a>
        </li>
        <li class="nav-item">
            <a class="nav-link" href="/tracks">All
Tracks</a>
        </li>
    </ul>
</div>
</div>
</nav>
<div class="container">
    <h1 class="my-5">All Albums</h1>
    <table class="table">
        <thead>
            <tr>
                <th>ID</th>
                <th>Name</th>
                <th>Year</th>
                <th>Action</th>
            </tr>
        </thead>
        <tbody>
            <tr th:each="album : ${albums}">
                <td><a
th:href="@{/album_tracks/{id} (id=${album.id})}"
th:text="${album.id}"></a></td>
                <td th:text="${album.name}"></td>
                <td th:text="${album.year}"></td>
                <td>
                    <form
th:action="@{/deletea/{id} (id=${album.id})}" method="post">
                        <input type="hidden" name="id_album"
th:value="${album.id}" />
                        <button type="submit">Видалити</button>
                    </form>
                </td>
            </tr>
        </tbody>
    </table>
</div>
<script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"
    integrity="sha384-
q8i/X+965Dz00rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo
"
    crossorigin="anonymous"></script>
<script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd
/popper.min.js"
    integrity="sha384-
U02eT0CpHqdsSJQ6hJty5KVphtPhzWj9WO1clHTMGa3JDZwrnQq4sF86dIHNSLr01

```



```

"
    crossorigin="anonymous"></script>
<script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/boots
trap.min.js"
    integrity="sha384-
JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEaFf/nJGzIxFDsf4x0xIM+B07jRM
"
    crossorigin="anonymous"></script>
</body>
</html>

```

Лістинг 2 — Album.java

```

package com.example.coursework.models;

import jakarta.persistence.*;
import lombok.Getter;
import lombok.Setter;

import java.util.LinkedHashSet;
import java.util.Set;

@Getter
@Setter
@Entity
@Table(name = "album")
public class Album {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id_album", nullable = false)
    private Integer id;

    @Column(name = "name")
    private String name;

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

    @OneToMany(mappedBy = "idAlbum")
    private Set<Track> tracks = new LinkedHashSet<>();
}

```

Лістинг 3 — AlbumController.java

```

package com.example.coursework.controllers;

import com.example.coursework.models.Album;
import com.example.coursework.services.AlbumService;
import lombok.RequiredArgsConstructor;
import org.springframework.stereotype.Controller;

```

```

import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;

import java.util.List;

@Controller
@RequiredArgsConstructor
public class AlbumController {

    private final AlbumService albumService;

    @GetMapping("/albums")
    public String getAllAlbums(Model model) {
        List<Album> albums = albumService.getAllAlbums();
        model.addAttribute("albums", albums);
        return "albums";
    }

    @GetMapping("/create")
    public String createAlbumForm(Model model) {
        model.addAttribute("album", new Album());
        return "createAlbum";
    }

    @PostMapping("/createa")
    public String createAlbum(@ModelAttribute Album album) {
        albumService.createAlbum(album);
        return "redirect:/albums";
    }

    @PostMapping("/deletea/{id}")
    public String deleteAlbum(@PathVariable int id) {
        albumService.deleteAlbum(id);
        return "redirect:/albums";
    }

    @GetMapping("/album_tracks/{id}")
    public String getAlbumTrack(@PathVariable int id, Model
model) {
        Album album = albumService.getAlbumById(id);
        model.addAttribute("album", album);
        return "album_tracks";
    }
}

```

Лістинг 4 — AlbumService.java

```

package com.example.coursework.services;

import com.example.coursework.models.Album;
import com.example.coursework.repositories.AlbumRepository;

```

```

import lombok.RequiredArgsConstructor;
import org.springframework.stereotype.Service;

import java.util.List;
@Service
@RequiredArgsConstructor
public class AlbumService {
    private final AlbumRepository albumRepository;

    public List<Album> getAllAlbums() {
        return albumRepository.findAll();
    }

    public Album getAlbumById(int id){
        return albumRepository.findById(id).orElse(null);
    }

    public void createAlbum(Album album){
        albumRepository.save(album);
    }
    public void deleteAlbum(int idAlbum){
        albumRepository.deleteById(idAlbum);
    }
}

```

Лістинг 5 — AlbumRepository.java

```

package com.example.coursework.repositories;

import com.example.coursework.models.Album;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface AlbumRepository extends JpaRepository<Album,
Integer> {
}

```

Лістинг 6 — TrackController.java

```

package com.example.coursework.controllers;

import com.example.coursework.models.*;
import com.example.coursework.services.*;
import lombok.RequiredArgsConstructor;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.*;

import java.util.*;

```

```

@Controller
@RequiredArgsConstructor
public class TrackController {
    private final TrackService trackService;
    private final AlbumService albumService;
    private final CompositorService compositorService;
    private final GenreService genreService;
    private final LabelService labelService;
    private final TrackLabelService trackLabelService;
    private final PlaylistService playlistService;
    private final ArtistService artistService;

    @GetMapping("/tracks")
    public String getAllTracks(Model model) {
        List<Track> tracks = null;
        tracks = trackService.getAllTracks();
        model.addAttribute("tracks", tracks);
        return "tracks";
    }

    @GetMapping("/track/{id}")
    public String getTrackById(@PathVariable int id, Model
model) {
        Track track = trackService.getTrackById(id);

        model.addAttribute("track", track);

model.addAttribute("playlists",playlistService.getAllPlaylist());
;

model.addAttribute("albums",albumService.getAllAlbums());

model.addAttribute("artists",artistService.getAllArtists());
        model.addAttribute("labels",labelService.getAll());

        return "track_info";
    }

    @GetMapping("/createTrack/{artistId}")
    public String createTrackForm(Model model, @PathVariable int
artistId) {
        model.addAttribute("track", new Track());

model.addAttribute("albums",albumService.getAllAlbums());

model.addAttribute("compositors",compositorService.getAll());
        model.addAttribute("genres",genreService.getAll());
        model.addAttribute("labels",labelService.getAll());

        model.addAttribute("artistId",artistId);
        return "createTrack";
    }

    @PostMapping("/createt/{artistId}")
    public String createAlbum(@ModelAttribute Track track,

```

```

@RequestParam("compositor") int compositorId,
                                @RequestParam("genre") int
genreId, @RequestParam("label") int labelId, @PathVariable int
artistId,
                                @RequestParam("year_edition") int
yearEdition) {

    int trackId = trackService.createTrack(track);
    track.setId(trackId);
    trackLabelService.save(new TrackLabel(0, new
RecordLabel(labelId), new Artist(artistId), track, yearEdition));
    return "redirect:/tracks";
}

@PostMapping("/update-tracks")
public String updateTrack(@ModelAttribute("track") Track
track, Model model) {

    Album album =
albumService.getAlbumById(track.getIdAlbum().getId());
    if (album != null) {
        track.setIdAlbum(album);
    }
    trackService.updateTrack(track);
    TrackLabel trackLabel = track.getTrackLabel();
    trackLabel.setIdTrack(track);
    trackLabelService.save(trackLabel);
    return "redirect:/track/" + track.getId();
}

@PostMapping("/deletet/{id}")
public String deleteTrack(@PathVariable int id) {
    trackService.deleteTrack(id);
    return "redirect:/tracks";
}
}

```

Лістинг 7 — Track.java

```

package com.example.coursework.models;

import jakarta.persistence.*;
import lombok.Getter;
import lombok.Setter;

import java.util.LinkedHashSet;
import java.util.Set;

@Getter
@Setter
@Entity
@Table(name = "track")

```

```

public class Track {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id_track", nullable = false)
    private Integer id;

    @Column(name = "duration")
    private String duration;

    @ManyToOne(fetch = FetchType.LAZY, optional = false)
    @JoinColumn(name = "id_album", nullable = false)
    private Album idAlbum;

    @Column(name = "name")
    private String name;

    @ManyToMany(mappedBy = "tracks", cascade =
CascadeType.PERSIST)
    private Set<Compositor> composers = new LinkedHashSet<>();

    @ManyToMany(mappedBy = "tracks", cascade =
{CascadeType.DETACH, CascadeType.MERGE, CascadeType.PERSIST,
CascadeType.REFRESH})
    private Set<Genre> genres = new LinkedHashSet<>();

    @ManyToMany(mappedBy = "tracks")
    private Set<Playlist> playlists = new LinkedHashSet<>();

    @OneToOne(mappedBy = "idTrack")
    private TrackLabel trackLabel;

    public void addGenre(Genre genre) {
        this.genres.add(genre);
        genre.addTrack(this);
    }

    public void addCompositor(Compositor compositor) {
        this.composers.add(compositor);
        compositor.addTrack(this);
    }
}

```

Лістинг 8 — TrackService.java

```

package com.example.coursework.services;

import com.example.coursework.models.Track;
import com.example.coursework.repositories.TrackRepository;
import lombok.RequiredArgsConstructor;
import org.springframework.stereotype.Service;

import java.util.List;

```

```

@Service
@RequiredArgsConstructor
public class TrackService {
    private final TrackRepository trackRepository;

    public List<Track> getAllTracks(){
        return trackRepository.findAll();
    }

    public Track getTrackById(int id){
        return trackRepository.findById(id).orElse(null);
    }

    public int createTrack(Track track){
        return trackRepository.save(track).getId();
    }

    public void deleteTrack(int idTrack){
        trackRepository.deleteById(idTrack);
    }

    public void updateTrack(Track track){
        trackRepository.updateTrack(track.getDuration(), track.getIdAlbum(), track.getName(), track.getId());
    }
}

```

Лістинг 9 — Сторінка “Всі треки”. Tracks.html

```

<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>All Tracks</title>
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/boo
tstrap.min.css"
        integrity="sha384-
ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T
" crossorigin="anonymous">
    <link rel="stylesheet" href="/css/style.css"
type="text/css">
</head>
<body>
<nav class="navbar navbar-expand-lg navbar-dark">
    <div class="container">
        <a class="navbar-brand" href="/">Albums App</a>
        <button class="navbar-toggler" type="button" data-
toggle="collapse" data-target="#navbarNav"
            aria-controls="navbarNav" aria-expanded="false"
aria-label="Toggle navigation">
            <span class="navbar-toggler-icon"></span>
        </button>
    </div>
</nav>

```

```

        <div class="collapse navbar-collapse" id="navbarNav">
            <ul class="navbar-nav ml-auto">
                <!--
                    <li class="nav-item">-->
                <!--
                    <a class="nav-link"
href="/allTrackPlaylist">All Playlist</a>-->
                <!--
                    </li>-->
                <li class="nav-item">
                    <a class="nav-link" href="/albums">All
Albums</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="/create">Add
Album</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="/createpl">Add
Playlist</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="/artists">All
Artists</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="/createar">Add
Artist</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="/tracks">All
Tracks</a>
                </li>
            </ul>
        </div>
    </div>
</nav>
<div class="container">
    <h1 class="my-5">All Tracks</h1>
    <table class="table">
        <thead>
            <tr>
                <th>ID</th>
                <th>Name</th>
                <th>Duration</th>
                <th>Album</th>
            </tr>
        </thead>
        <tbody>
            <tr th:each="track : ${tracks}">
                <td><a th:href="@{/track/{id} (id=${track.id})}"
th:text="${track.id}"></a></td>
                <td th:text="${track.name}"></td>
                <td th:text="${track.duration}"></td>
                <td th:text="${track.idAlbum.name}"></td>
            </tr>
        </tbody>
    </table>

```



```

        </tbody>
    </table>
</div>
<script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"
        integrity="sha384-
q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo
"
        crossorigin="anonymous"></script>
<script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd
/popper.min.js"
        integrity="sha384-
U02eT0CpHqdSJQ6hJty5KVphtPhzWj9WO1clHTMGa3JDZwrnQq4sF86dIHNSLr01
"
        crossorigin="anonymous"></script>
<script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/boots
trap.min.js"
        integrity="sha384-
JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEaFf/nJGzIxFDs4x0xIM+B07jRM
"
        crossorigin="anonymous"></script>
</body>
</html>

```

Лістинг 10 — TrackRepository.java

```

package com.example.coursework.repositories;

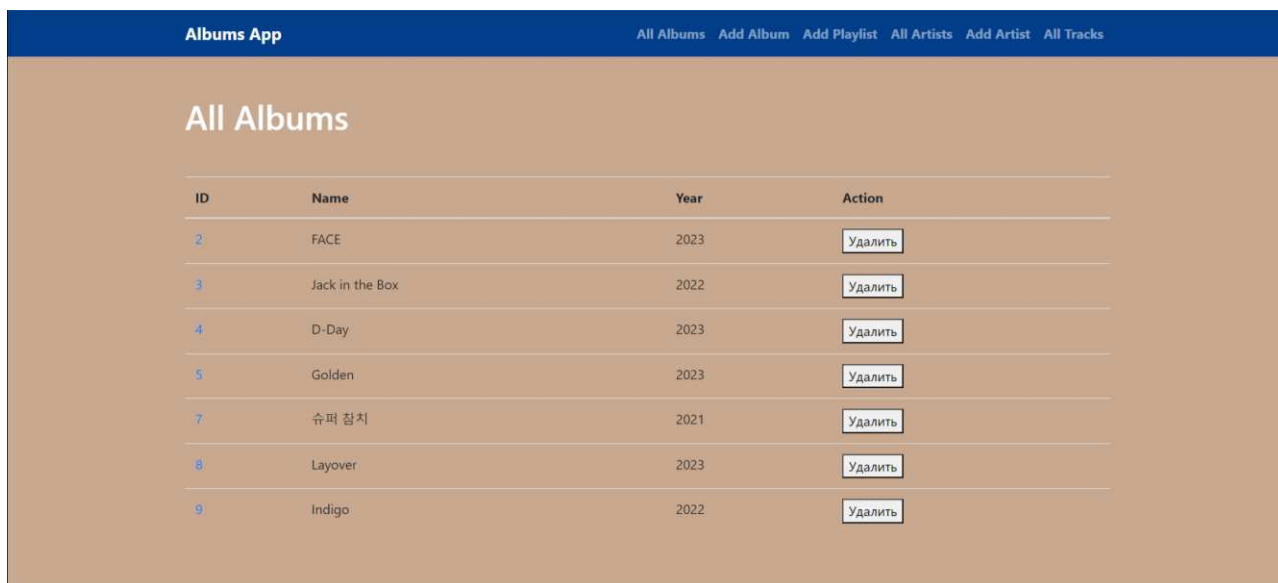
import com.example.coursework.models.Album;
import com.example.coursework.models.Track;
import lombok.RequiredArgsConstructor;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Modifying;
import org.springframework.data.jpa.repository.Query;
import org.springframework.stereotype.Repository;
import org.springframework.transaction.annotation.Transactional;

@Repository
public interface TrackRepository extends JpaRepository<Track,
Integer> {
    @Transactional
    @Modifying
    @Query("update Track t set t.duration = ?1, t.idAlbum = ?2,
t.name = ?3 where t.id = ?4")
    void updateTrack(String duration, Album idAlbum, String name
, Integer id);
}

```

ДОДАТОК В – Інструкція користувача

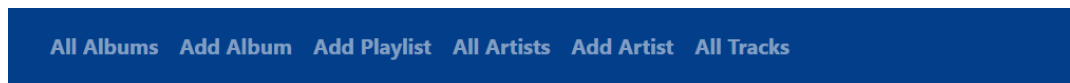
При відкритті сайту одразу опиняємось на головній сторінці (рис. В-1).



ID	Name	Year	Action
2	FACE	2023	Удалить
3	Jack in the Box	2022	Удалить
4	D-Day	2023	Удалить
5	Golden	2023	Удалить
7	슈퍼 참치	2021	Удалить
8	Layover	2023	Удалить
9	Indigo	2022	Удалить

Рисунок В-1 — Сторінка з усіма альбомами

В правому верхньому куті знаходяться сторінки навігації (рис.В-2).



All Albums	Add Album	Add Playlist	All Artists	Add Artist	All Tracks
------------	-----------	--------------	-------------	------------	------------

Рисунок В-2 — Вкладки веб-сайту

Натиснувши, на вкладку “Всі артисти”, яку представлено на рисунку В-3, знаходиться список всіх артистів та кнопки для видалення інформації про них або додавання нових пісень.

Albums App

All AlbumsAdd AlbumAdd PlaylistAll ArtistsAdd ArtistAll Tracks

All Artists

ID	Name	Gender	Country	Role	Year of birth	Action
1	Agust D	male	SK	rapper	1993	Удалить Добавить трек
2	Jimin	male	SK	sub	1995	Удалить Добавить трек
3	J-Hope	male	SK	rapper	1994	Удалить Добавить трек

Рисунок В-3 — Сторінка з усіма артистами

Натиснувши кнопку “Додати пісню” (рис. В-4), вносимо в відповідну форму всю інформацію: назву трека, альбом до якого він входить, композитора, студію звукозапису, жанр, тривалість звучання, рік випуску; та натиснувши кнопку зберігає всю інформації та одразу бачимо нову додану пісню.



Create Track

Name: Closer to You

Album: Golden

Compositor: Blood Pop

Genre: K-pop

Label: BigHit

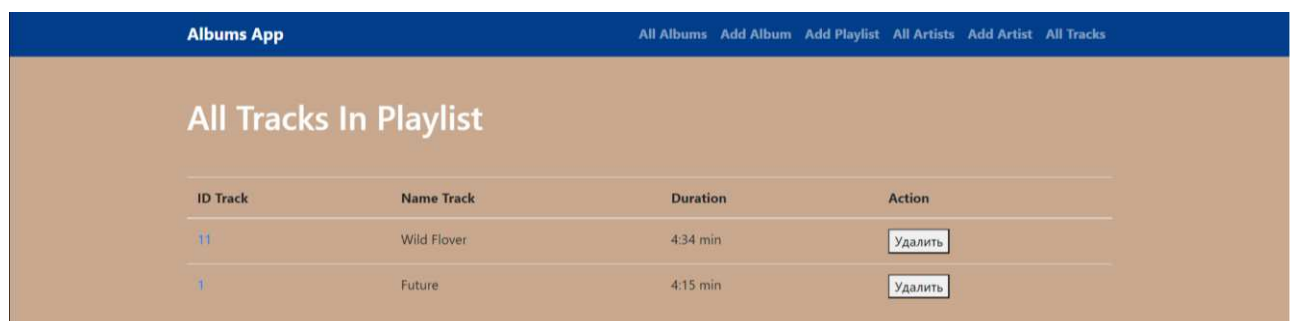
Duration: 2:51 min

Year Edition: 2023

Create

Рисунок В-4 — Сторінка для додання нового трека

Відкривши вкладку “Всі пісні плейлиста” (рис. В-5), зможемо побачити всі наявні плейлисти, натиснувши на потрібний плейлист побачимо всі пісні, що входять до нього та всю інформацію при них.



ID Track	Name Track	Duration	Action
11	Wild Flower	4:34 min	Удалить
1	Future	4:15 min	Удалить

Рисунок В-5 — Сторінка з усіма піснями плейлиста

При редагуванні потрібної інформації система сама вставить попередні дані, потрібно тільки замінити їх на потрібну інформацію.