МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Кафедра комп'ютеризованих систем управління

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри

_____Олександр ЛИТВИНЕНКО

«___» _____2023 р.

КВАЛІФІКАЦІЙНА РОБОТА

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ЗДОБУВАЧА ВИЩОЇ ОСВІТИ
СТУПЕНЯ «МАГІСТР»

Тема: Програмний модуль зберігання *MP3* файлів на основі мікросервісної архітектури

Виконавець: Богдан БОЙКО

Керівник: Галина РОСІНСЬКА

Нормоконтролер: Євгеній ТУПОТА

Київ 2023

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет комп'ютерних наук та технологій

Кафедра комп'ютеризованих систем управління

Спеціальність 123 «Комп'ютерна інженерія»

Освітньо професійна програма «Системне програмування»

Форма навчання денна

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____Олександр ЛИТВИНЕНКО

«_____»_____2023 р.

ЗАВДАННЯ

на виконання кваліфікаційної роботи

Бойка Богдана Миколайовича

1.Тема роботи: «Програмний модуль зберігання *MP3* файлів на основі мікросервісної архітектури»
затверджена наказом ректора від «28» серпня 2023 року № 1494 /ст.

2. Термін виконання роботи: з 02.10.2023 до 31.12.2023

3. Вихідні дані до проєкту (роботи): постановка задачі до виконання роботи, мова програмування *Python.*

4. Зміст пояснювальної записки (перелік питань, що підлягають розробці):

1) аналіз методів розробки програмного модулю зберігання *MP*3 файлів на основі мікросервісної архітектури;
2) моделювання роботи модуля фільтрації музичної системи;
3) описання розробленого модуля зберігання *MP*3 файлів.

5. Перелік обов'язкового графічного матеріалу:

1) Дерево функцій програмного модуля;
2) Зв'язки між таблицями програмного модуля;
3) Вікна роботи програмного модуля;
4) схема алгоритму попередньої обробки налаштувань для роботи фільтру;
5) схема алгоритму обробки списку видачі.

6. Календарний план-графік

№ п/п	Етапи виконання кваліфікаційної роботи	Термін виконання етапів	Примітка
1	Провести аналіз літератури за темою кваліфікаційної роботи та аналіз існуючих систем	02.10.2023-12.10.2023	
2	Аналіз існуючих технологій	13.10.2023-29.10.2023	
3	Зробити вибір компонентів системи	30.10.2023-06.11.2023	
4	Розробити структуру програмного модулю	07.10.2023-14.11.2023	
5	Розробити програмні засоби для роботи з клієнтами через месенджер *Telegram*	15.10.2023-26.11.2023	
6	Провести налаштування програмних засобів на сервері	27.10.2023-01.12.2023	
7	Написати пояснювальну записку	02.12.2023-14.12.2023	
8	Підготувати презентацію	15.12.2023-17.12.2023	
9	Оформити супрводжувальну документацію	18.12.2023 19.12.2023	

7. Дата видачі завдання «02» жовтня 2023 р.

Керівник кваліфікаційної роботи_____Галина РОСІНСЬКА
(підпис)

Завдання прийняв до виконання_____Богдан БОЙКО
(підпис студента)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи «Програмний модуль зберігання *MP3* файлів на основі мікросервісної архітектури» складається з: 80 с., 22 рис., 8 таблиць, 26 літературних джерел, 1 додаток.

АУДІОФАЙЛ, КОДУВАННЯ, МІКРОСЕРВІСИ

Об'єкт дослідження – процес обробки та зберігання аудіофайлів у форматі *MP*3 на віддалених серверах.

Предмет дослідження – програмний модуль для зберігання та обміну *MP*3 файлами на основі мікросервісної архітектури.

Мета дослідження – розробка програмного модуля для зберігання та обміну *MP*3 файлами на основі мікросервісної архітектури, а також проведення наукового дослідження, спрямованого на вдосконалення обробки *MP*3 файлів та оптимізацію системи.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

API – Application Programming Interface

ANN – Artificial Neural Networks

ARM – Advanced Risc Machine

CMN – Collaborative Memory Networks

FISSA – Fusing Item Similarity Models with Self-Attention

NCF – Neural Collaborative Filtering

ВСТУП

Сучасний світ зазнає надзвичайних змін завдяки стрімкому розвитку інформаційних технологій. Однією з ключових галузей в цьому процесі є розробка програмних систем, які спрощують та полегшують роботу користувачів у багатьох аспектах їхнього повсякденного життя. Однак із зростанням обсягу та різноманітності цифрових даних з'являються нові виклики та проблеми, пов'язані з їх обробкою, зберіганням та обміном.

Однією з таких проблем є обробка та зберігання аудіофайлів у форматі *MP*3. Музичні файли *MP*3 широко використовуються у всьому світі і представляють собою важливу частину цифрового контенту. Проте обробка та зберігання великого обсягу *MP*3 файлів може бути складною задачею, особливо якщо це відбувається на віддалених серверах або у розподілених середовищах. Ця проблема стає особливо актуальною в контексті зростання об'ємів цифрових аудіофайлів та потреби в їхньому зберіганні та обміні, наприклад, у сферах музичного маркетингу, розповсюдження музичного контенту або музичних платформ для користувачів.

Кваліфікаційна робота присвячена розробці програмного модуля для зберігання та обміну *MP*3 файлами на основі мікросервісної архітектури. Вирішення цієї проблеми вимагає глибокого розуміння архітектурних аспектів, вибору технологій та оптимізації обробки *MP*3 файлів. Однак ця робота також має науковий характер, оскільки вона включає в себе наукове дослідження у сфері обробки та зберігання аудіофайлів, а також дослідження ефективності мікросервісної архітектури для вирішення даної задачі.

Метою роботи є розробка програмного модуля для зберігання та обміну *MP*3 файлами на основі мікросервісної архітектури, а також проведення наукового дослідження, спрямованого на вдосконалення обробки *MP*3 файлів та оптимізацію системи.

Для досягнення цієї мети були визначені наступні напрями дослідження:

– принципи обробки *MP*3 файлів на віддалених серверах та аналіз існуючих рішень у цій області;

– вимоги до програмного модуля для зберігання та обміну *MP*3 файлами;

– мікросервісна архітектура програмного модуля та опис компонентів системи на базі мікросервісної архітектури;

– технології для реалізації програмного модуля та обґрунтування цього вибору;

– інтеграція програмного модуля з месенджером *Telegram* та іншими сторонніми сервісами для обміну *MP*3 файлами;

– оптимізація обробки *MP*3 файлів на сервері для підвищення продуктивності та якості обробки.

Об'єктом дослідження є процес розробки програмного забезпечення для обробки та зберігання аудіофайлів у форматі *MP*3 на віддалених серверах.

Предметом дослідження є програмний модуль для зберігання та обміну *MP*3 файлами на основі мікросервісної архітектури, а також методи оптимізації обробки *MP*3 файлів.

Основними задачами дослідження є:

– розробка програмного модуля для зберігання та обміну *MP*3 файлами на основі мікросервісної архітектури;

– проведення наукового дослідження з питань обробки та зберігання аудіофайлів у форматі *MP*3;

– вивчення та аналіз існуючих підходів та рішень у сфері обробки *MP*3 файлів та їхнього зберігання;

– визначення основних вимог до програмного модуля та розробка дерева функцій;

– вибір та обґрунтування технологій для реалізації програмного модуля;

– інтеграція програмного модуля з месенджером *Telegram* та іншими сторонніми сервісами для забезпечення обміну *MP*3 файлами;

– оптимізація процесу обробки *MP*3 файлів на сервері з метою підвищення продуктивності та якості обробки.

Сучасні технології та підходи можуть бути використані в розробці програмного модуля для зберігання та обміну *MP*3 файлами. Важливо вибрати ті технології, які найкраще відповідають вашим потребам та вимогам проекту, з урахуванням ефективності, масштабованості та безпеки системи.

Наукова новизна роботи полягає в декількох ключових аспектах, які роблять це дослідження унікальним і цінним для області обробки та зберігання аудіофайлів у форматі *MP*3:

1. Використання мікросервісної архітектури для розробки програмного модуля, що дозволяє побудувати систему як набір невеликих незалежних компонентів, що спрощує розгортання, масштабування та підтримку.

2. Оптимізація процесу обробки *MP*3 файлів на сервері може допомогти знайти ефективні та швидкі способи обробки аудіоданих, що є ключовим аспектом в розробці аналогічних систем.

3. Інтеграція програмного модуля з месенджерами, зокрема з *Telegram,* є ще однією новизною, яка може знайти застосування у побудові нових медійних сервісів та платформ для обміну аудіофайлами, що може бути особливо актуальним у сучасному цифровому світі.

Апробація результатів була проведена на науковій практичній конференції “Сучасні тенденції розвитку системного програмування” (м. Київ, 23-24 грудня 2023 р.). За результатами опубліковано тезу доповіді: Мікульський В.В. Тестування зіткнень між об'єктами в 3*D* симуляторі // Тези доповідей наук.-практ. конф. “Сучасні тенденції розвитку системного програмування” (23-24 листопада 2023 р.). – К.: НАУ, 2023. – С. 43-44.

РОЗДІЛ 1
АНАЛІЗ МЕТОДІВ РОЗРОБКИ ПРОГРАМНОГО МОДУЛЮ ЗБЕРІГАННЯ *MP3* ФАЙЛІВ НА ОСНОВІ МІКРОСЕРВІСНОЇ АРХІТЕКТУРИ

1.1. Принципи обробки *MP*3 файлів на віддалених серверах

В добу цифрових технологій аудіофайли у форматі *MP*3 стали одним із основних способів зберігання та обміну аудіоданими. Для розуміння проблематики обробки та зберігання *MP*3 файлів на віддалених серверах, важливо спочатку розглянути основні принципи цього формату, його структуру та кодування.

1.1.1. Розгляд структури *MP*3 файлів та їхнього кодування

*MP*3 (*MPEG*-1 *Audio Layer* 3) – це аудіокодек, розроблений для стиснення аудіоданих з мінімальною втратою якості. Файли *MP*3 мають складну структуру, що включає в себе заголовок файлу, фрейми з аудіоданими та заголовки фреймів:

– заголовок файлу: у заголовку файлу *MP*3 міститься мета-інформація про аудіофайл, така як назва, виконавець, бітрейт, частота дискретизації, тощо. Ця інформація допомагає програвачам та програмам коректно відтворювати аудіо та відображати його власникам;

– фрейми з аудіоданими: є основними блоками даних у файлі *MP*3. Кожен фрейм містить певну кількість аудіоданих та метадані, необхідні для декодування та відтворення звуку. Структура фрейму включає в себе заголовок, де зберігається інформація про бітрейт, режим стиснення, наявність захисту від помилок, тощо;

– заголовки фреймів: Кожен фрейм має власний заголовок, що містить ключову інформацію для декодування аудіоданих. Завдяки цьому, програвачі можуть правильно інтерпретувати дані та відтворювати аудіо.

Кодування *MP*3 полягає у видаленні непотрібної інформації та акустичних складових, які людське вухо не сприймає. Це дозволяє зменшити розмір файлу, несуттєво втративши якість звуку. Для розуміння обробки *MP*3 файлів на

віддалених серверах, важливо мати чітке уявлення про цей процес кодування та структуру файлів.

1.1.2. Оцінка основних етапів обробки *MP*3 файлів на серверах

Обробка *MP*3 файлів на віддалених серверах включає в себе кілька основних етапів:

– завантаження файлу: перший етап – це завантаження *MP*3 файлу на сервер. Це може бути здійснене користувачем або автоматично через інші джерела, такі як месенджери або інші сервіси;

– розпакування та дешифрування: одержаний файл може бути зашифрованим або стиснутим. На сервері він розпаковується та дешифрується до початкового *MP*3 формату для подальшої обробки;

– обробка аудіо: на цьому етапі можуть проводитися різні операції з аудіоданими, такі як фільтрація шуму, зміна частоти дискретизації, зменшення бітрейту тощо. Основна задача – підготовка аудіо для подальшого зберігання або відтворення;

– зберігання: після обробки, аудіофайл може бути збережений на сервері. Тут важливо визначити стратегію зберігання, де враховується доступність, швидкість та надійність;

– відправлення користувачу або іншим сервісам: Збережені аудіодані можуть бути відправлені користувачу або іншим сервісам, таким як месенджери, для подальшого використання або обміну.

1.1.3. Визначення вимог до обробки та зберігання *MP*3 даних

Важливо визначити вимоги до обробки та зберігання *MP*3 даних на сервері, оскільки це визначить параметри системи та необхідність використання певних технологій. Деякі з ключових вимог можуть включати:

– безпека: збереження інформації про користувачів та їхні файли повинно бути захищеним від несанкціонованого доступу та атак;

– швидкодія: оперативність обробки та зберігання *MP*3 файлів має бути на достатньо високому рівні для забезпечення задоволення користувачів;

– Масштабованість: система повинна бути готовою до масштабування для обробки великої кількості аудіофайлів;

– Надійність: збереження та обробка аудіоданих повинна бути надійною, щоб уникнути втрати файлів чи їхніх пошкоджень.

– Інтеграція з іншими системами: Якщо система має взаємодіяти з іншими сервісами, вимоги до інтеграції також повинні бути визначені.

1.2. Огляд літератури та існуючих програмних рішень

Розглянемо літературу та існуючі програмні рішення, що стосуються обробки та зберігання аудіофайлів у форматі *MP*3 на віддалених серверах. Огляд літератури та аналіз існуючих програмних продуктів є важливим етапом для зрозуміння поточного стану галузі та ідентифікації можливих варіантів для розробки власного програмного модуля.

1.2.1. Огляд літератури з тематики обробки та зберігання аудіофайлів

Обробка та зберігання аудіофайлів, зокрема у форматі *MP*3, є актуальною проблемою у сучасному інформаційному суспільстві [1-5]. Ця проблематика залучає багато дослідників та розробників [6, 7, 8]. Переглянемо основні теми та напрямки літератури, пов'язані з цією областю.

– способи стиснення аудіоданих: в літературі активно досліджуються методи стиснення аудіоданих у форматі *MP*3 та інших аудіоформатах [9, 10]. Роботи з цієї області розглядають алгоритми стиснення, такі як *MPEG* або *AAC*, і їхні впливи на якість звуку та розмір файлу;

– методи обробки аудіо: література також містить дослідження з обробки аудіоданих [11, 12, 13], які можуть включати в себе фільтрацію, еквалізацію, підсилення та інші методи обробки для покращення якості аудіо;

– зберігання аудіо на серверах: деякі дослідження [14, 15, 16] спрямовані на проблеми зберігання аудіофайлів на серверах, зокрема в хмарних сервісах. Вони розглядають аспекти надійності, швидкодії та масштабованості систем зберігання;

– інтеграція з месенджерами та соціальними мережами [17, 18, 19, 20]: у світі, де обмін мультимедійними даними через месенджери та соціальні мережі стає все популярніше, література розглядає можливості інтеграції аудіофайлів з цими платформами;

– аналіз споживачів аудіофайлів: деякі дослідження [21, 22, 23] аналізують споживачів аудіофайлів, їхні уподобання та звички щодо зберігання та обміну музичними даними.

1.2.2. Аналіз існуючих програмних рішень для зберігання та обміну *MP*3 файлами

На ринку існують різноманітні програмні рішення для зберігання та обміну аудіофайлами у форматі *MP*3. Деякі з них є популярними та вже завоювали довіру користувачів. Розглянемо декілька прикладів існуючих програмних рішень для зберігання та обміну *MP*3 файлами та побудуємо порівняльну таблицю, визначаючи їхні переваги та недоліки:

1. *OneDrive*:

Переваги:

– Інтеграція зі суітами офісних додатків (*Microsoft Office*).

– Зручний доступ до файлів з різних пристроїв.

– Підтримка автоматичної синхронізації файлів.

Недоліки:

– Обмежений безкоштовний обсяг зберігання.

– Відсутність спеціалізованих функцій для музики та аудіофайлів.

2. *iCloud*:

Переваги:

– Інтеграція з екосистемою *Apple*.

– Автоматична синхронізація медіафайлів між пристроями.

– Великий безкоштовний обсяг зберігання для музики та фотографій.

Недоліки:

– Обмежена підтримка для немузичних аудіофайлів.

– Відсутність широкого доступу до системи файлів.

3. *Amazon Drive*:

Переваги:

– Інтеграція з *Amazon Music*.

– Зручний доступ до медіафайлів з пристроїв.

– Розширені можливості для сортування та організації музичної бібліотеки.

Недоліки:

– Обмежений безкоштовний обсяг зберігання.

– Можливість обмеженого доступу до деяких файлів поза *Amazon*-екосистемою.

4. *Box*:

Переваги:

– Підтримка обміну великими аудіофайлами.

– Висока безпека та контроль доступу.

– Інтеграція з популярними інструментами для спільної роботи.

Недоліки:

– Обмежений обсяг безкоштовного зберігання.

– Відсутність розвинутої функціональності для відтворення аудіо в самому сервісі.

5. *Mega*:

– Переваги:

– Великий безкоштовний обсяг зберігання.

– Захист даних завдяки шифруванню.

– Зручний інтерфейс для завантаження та обміну файлами.

– Недоліки:

– Відсутність інтеграції з більшістю музичних додатків.

– Можливість обмеженого доступу до медіафайлів у безкоштовному варіанті.

Табл. 1.1 надає загальну інформацію щодо переваг та недоліків існуючих програмних рішень для зберігання та обміну *MP*3 файлами. Вибір конкретного рішення повинен базуватися на конкретних потребах та вимогах вашого проекту.

Таблиця 1.1

Порівняльна таблиця використання хмарних сервісів для обробки *mp*3 файлів

Параметр	*OneDrive*	*iCloud*	*Amazon Drive*	*Box*	*Mega*
Безкоштовний обсяг	Обмежений	Обширний	Обмежений	Обмежений	Великий
Інтеграція з іншими сервісами	Так	Так	Так	Так	Ні
Підтримка музики	Обмежена	Обмежена	Так	Ні	Обмежена
Швидкодія	Добра	Добра	Добра	Добра	Добра
Захист даних	Так	Так	Так	Так	Так

Існують спеціалізовані засоби для обробки *MP*3 файлів, які не є хмарними. Ці засоби можуть бути встановлені на локальних серверах або комп'ютерах і надають розширені можливості для обробки та редагування аудіофайлів у форматі *MP*3. Ось деякі приклади таких програм та бібліотек:

1. *Audacity*: *Audacity* є відкритим і безкоштовним аудіоредактором, який підтримує формат *MP*3. Він дозволяє вам записувати, редагувати, змінювати швидкість відтворення та багато інших операцій над аудіофайлами у форматі *MP*3.

2. *Adobe Audition*: *Adobe Audition* – це професійний аудіоредактор від *Adobe*, який також підтримує *MP*3. Він має багато функцій для обробки аудіофайлів, включаючи шумозменшення, еквалізацію, зміну темпу та багато інших.

3. *Reaper*: *Reaper* – це цифрова аудіостанція для обробки та створення музики. Вона також підтримує *MP*3 і має розширені можливості для зведення аудіо.

4. *FFmpeg*: *FFmpeg* – це мультимедійна бібліотека та набір утиліт для обробки аудіо та відео. Вона може бути використана для конвертації та обробки *MP*3 файлів з командного рядка.

5. *MP*3*tag*: *MP*3*tag* – це безкоштовна програма для редагування метаданих у *MP*3 файлах. Вона дозволяє вам змінювати інформацію про виконавця, назву треку, альбому та інше без втрати якості аудіо.

Ці програми та бібліотеки надають різні функціональні можливості для обробки *MP*3 файлів на локальних серверах або комп'ютерах. Вибір конкретного засобу залежить від ваших конкретних потреб і завдань (табл. 1.2).

Таблиця 1.2

Порівняльна таблиця спеціалізованих програм для обробки *MP*3 файлів

Програма	Переваги	Недоліки
Audacity	– Безкоштовна та відкрита програма.	– Інтерфейс може бути складним для новачків.
	– Широкий спектр функцій для обробки аудіо.	– Відсутність деяких професійних функцій порівняно з *Adobe Audition* або *Reaper*.
	– Підтримка різних платформ (*Windows, macOS, Linux*).	
Adobe Audition	– Професійний аудіоредактор з багатофункціональним набором інструментів.	– Платна програма (пробна версія доступна).
	– Висока якість обробки та обробка високоякісного аудіо.	– Вимагає більше ресурсів комп'ютера порівняно з *Audacity*.
	– Інтеграція з іншими програмами *Adobe Creative Cloud*.	– Має великий вивчений поріг вхід.
Reaper	– Дуже гнучкий та налаштовуваний аудіоредактор.	– Деякі можливості можуть здаватися складними для новачків.
	– Підтримка *VST* плагінів для розширення функціональності.	– Треба купувати ліцензію для повного функціоналу.
	– Сприяє створенню музики та зведенню аудіо.	
FFmpeg	– Безкоштовний та відкритий інструмент для командного рядка.	– Вимагає знань командного рядка для використання.
	– Великий вибір функцій для обробки та конвертації аудіо.	– Не має графічного інтерфейсу, що може бути незручним для користувачів без досвіду в командному рядку.
	– Підтримує багато аудіоформатів, включаючи *MP*3.	
*MP*3*tag*	– Простий та зручний інтерфейс для редагування метаданих *MP*3.	– Спеціалізується на редагуванні метаданих, не надає функціоналу для обробки звуку.
	– Швидка та ефективна робота з великими колекціями *MP*3 файлів.	
	– Безкоштовний та легкий у використанні.	

Вибір програми для обробки *MP*3 файлів залежить від ваших потреб та рівня досвіду. *Audacity* і *MP*3*tag* корисні для користувачів, які шукають безкоштовні та легкі у використанні інструменти, *Adobe Audition* і *Reaper* підходять для професіоналів, які потребують високоякісну обробку та створення

музики. *FFmpeg* є потужним інструментом для технічних спеціалістів, які шукають автоматизовану обробку через командний рядок.

*MP*3 *audio steganalysis* – це область цифрового аналізу, яка вивчає методи виявлення прихованих повідомлень або даних в аудіофайлах у форматі *MP*3. Стеганографія – це наука про приховування інформації в інших носіях даних так, щоб ця інформація залишалася непомітною для очей або вух. Стеганаліз – це наука про виявлення такої прихованої інформації.

*MP*3 *audio steganalysis* включає в себе розробку алгоритмів та методів для виявлення прихованих даних в аудіофайлах *MP*3. Ця область стала актуальною через потенційне використання стеганографії для незаконних цілей, таких як прихована передача конфіденційної інформації або авторських прав порушень.

Основні завдання *MP*3 *audio steganalysis* включають:

1. Розробка методів для виявлення прихованих повідомлень в аудіофайлах *MP*3.

2. Вивчення характеристик стеганографічних алгоритмів та методів їхньої інтеграції з аудіоданими.

3. Аналіз параметрів аудіофайлів *MP*3 з метою виявлення аномалій або змін, що можуть свідчити про наявність прихованої інформації.

4. Розробка алгоритмів для виявлення різних типів стеганографічних атак та відновлення оригінальних аудіоданих.

5. Підвищення надійності та ефективності методів стеганалізу в контексті аудіофайлів *MP*3.

1.3. Визначення основних задач та вимог до програмного модуля

1.3.1. Визначення функціональних вимог до програмного модуля

Визначення функціональних вимог до програмного модуля є ключовим етапом в розробці будь-якої програмної системи. Цей етап передбачає визначення того, які функції та можливості повинен надавати програмний модуль для досягнення поставлених завдань і вирішення проблеми.

Основними функціональними вимогами до програмного модуля зберігання *MP*3 файлів на основі мікросервісної архітектури можуть бути:

1. Завантаження *MP*3 файлів: Можливість завантаження *MP*3 файлів на сервер за допомогою веб-інтерфейсу або *API*.

2. Зберігання і каталогізація: Автоматичне зберігання завантажених *MP*3 файлів на сервері з можливістю їхньої каталогізації за різними параметрами (назва, виконавець, жанр, рік тощо).

3. Пошук і фільтрація: Можливість швидкого пошуку та фільтрації *MP*3 файлів за різними критеріями.

4. Відтворення і потокова передача: Підтримка відтворення *MP*3 файлів на сервері та їх потокової передачі на клієнтські пристрої.

5. Інтеграція з *Telegram*: Можливість обміну *MP*3 файлами через месенджер *Telegram,* включаючи відправку та отримання файлів.

6. Автоматична обробка та оптимізація: Автоматична обробка *MP*3 файлів на сервері, включаючи можливість оптимізації якості аудіо, видалення метаданих, стискання тощо.

7. Система доступу та безпеки: Реалізація системи авторизації, контролю доступу та захисту даних користувачів.

8. Інтеграція з іншими сервісами: Можливість інтеграції з іншими популярними сервісами, такими як *Dropbox, Google Drive* тощо, для зручного обміну файлами.

1.3.2. Створення дерева функцій для програмного модуля

Дерево функцій (*Functional Decomposition*) – це методологія розкладання складних систем або програмних модулів на менші функціональні частини, що допомагає легше розуміти і організовувати роботу над проектом. У випадку програмного модуля зберігання *MP*3 файлів, дерево функцій може мати наступний вигляд:

1. Завантаження та завантаження файлів:

– завантаження *mp*3 файлів;

– завантаження інших типів файлів;

2. Зберігання і каталогізація:

– зберігання *mp*3 файлів;

– каталогізація за жанром;

– каталогізація за роком;

3. Пошук та фільтрація:

– пошук за назвою;

– пошук за виконавцем;

– фільтрація за жанром;

– фільтрація за роком;

4. Відтворення та потокова передача:

– відтворення *mp*3 файлів;

– потокова передача на клієнтські пристрої;

5. Інтеграція з *Telegram*:

– відправка *mp*3 файлів через *Telegram*;

– отримання *mp*3 файлів через *Telegram*;

6. Автоматична обробка та оптимізація:

– обробка якості аудіо;

– видалення метаданих;

– стискання аудіо;

7. Система доступу та безпеки:

– авторизація користувачів;

– контроль доступу;

– захист даних;

8. Інтеграція з іншими сервісами:

– інтеграція з *Dropbox*;

– інтеграція з *Google Drive.*

Це дерево функцій допомагає розкласти функціональні можливості програмного модуля на окремі компоненти, що можуть бути реалізовані окремо. Кожен вузол дерева функцій представляє собою окрему функціональну частину модуля (рис. 1.1).

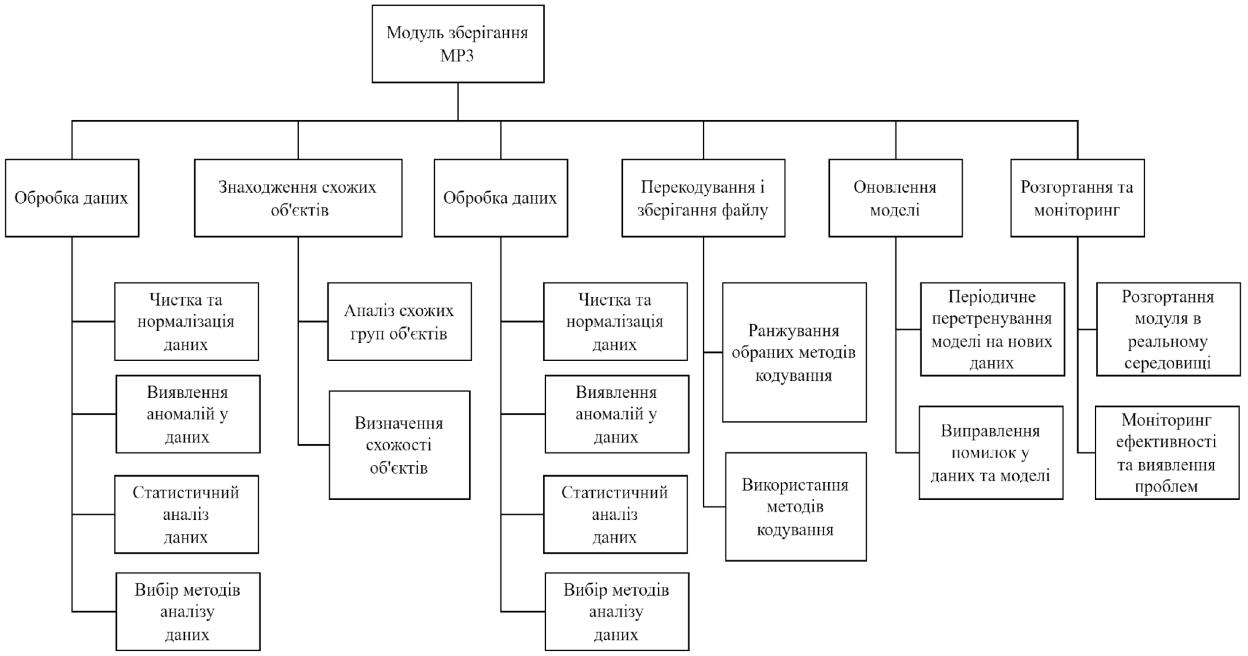

Рис 1.1. Дерево функцій

Після визначення функціональних вимог та створення дерева функцій, важливо встановити пріоритети для задач і функцій модуля. Це допомагає визначити, які функції будуть реалізовані в першу чергу, а які можуть бути реалізовані пізніше або за умови наявності додаткового часу та ресурсів.

Пріоритети можуть бути встановлені на основі важливості функцій для користувачів, вимог замовника, термінів виконання проекту та інших факторів. Наприклад, функції, пов'язані з завантаженням і зберіганням *MP*3 файлів, можуть мати високий пріоритет, оскільки вони становлять основну функціональність модуля, тоді як деякі опціональні функції можуть мати менший пріоритет і бути реалізовані пізніше.

1.4. Обгрунтування використання мікросервісної архітектури для розробки програмного модуля

Мікросервісна архітектура – це архітектурний підхід до розробки програмного забезпечення, при якому програмний модуль розбивається на невеликі незалежні сервіси, які працюють разом, взаємодіючи через *API*. Кожен

мікросервіс відповідає за виконання обмеженого набору функціональних завдань і може бути розроблений та впроваджений окремо.

Головні концепції мікросервісної архітектури включають:

– Декомпозиція на сервіси: Поділ програми на невеликі сервіси, кожен з яких відповідає за певний функціонал. Це дозволяє зменшити складність кожного сервісу і покращити його можливості для масштабування.

– Незалежність і автономія: Кожен мікросервіс може бути розроблений, тестований та впроваджений незалежно від інших сервісів. Це забезпечує високу автономію розвитку та розгортання.

– Комунікація через *API*: Сервіси взаємодіють між собою за допомогою *API,* що дозволяє їм обмінюватися даними та функціональністю.

– Самостійне масштабування: Кожен сервіс може бути масштабований окремо, що дозволяє ефективно використовувати ресурси і забезпечувати високу доступність.

Обгрунтування вибору мікросервісної архітектури для розробки програмного модуля зберігання *MP*3 файлів є ключовим етапом при розробці будь-якої програмної системи. Основні аргументи та переваги такого вибору включають:

– Модульність: мікросервісна архітектура дозволяє розбити систему на окремі сервіси, які можуть бути розроблені та підтримані незалежно. У вас є можливість працювати над окремими функціональними блоками, що спрощує розробку та управління проектом.

– Масштабованість: завдяки незалежному масштабуванню кожного сервісу, ви можете ефективно реагувати на змінні потреби у ресурсах. Це дозволяє забезпечити високий рівень продуктивності та доступності.

– Гнучкість: мікросервісна архітектура надає гнучкість вибору технологій для кожного сервісу. Ви можете використовувати різні мови програмування та технології в залежності від потреб кожного модуля.

– Інтеграція: мікросервіси легше інтегруються з іншими системами та сервісами. У вашому випадку, це може бути інтеграція з *Telegram* та іншими сторонніми сервісами для обміну файлами.

– Споживач-постачальник: кожен сервіс може виступати як споживач інших сервісів та постачальник своїх послуг. Це забезпечує гармонійну взаємодію між компонентами системи.

Хоча мікросервісна архітектура має багато переваг, вона також може викликати певні труднощі та вимагати уваги до деяких аспектів. Ось деякі з переваг та можливих викликів, які варто врахувати при розробці програмного модуля:

Переваги:

– Модульність та гнучкість: можливість незалежної розробки та масштабування кожного сервісу.

– Ізоляція помилок: помилки в одному сервісі не впливають на решту системи.

– Швидкість розгортання та вдосконалення: Можливість швидко внести зміни та вдосконалення в окремі сервіси без впливу на решту системи.

– Паралельний розвиток: Різні команди можуть працювати над різними сервісами паралельно.

Можливі виклики:

– Складність управління: збільшується складність управління багатьма сервісами та їхньою координацією.

– Комунікація між сервісами: необхідно добре спроектувати та підтримувати механізми комунікації між сервісами через *API*.

– Спроблеми з тестуванням та відлагодженням: важливо мати ефективні засоби тестування та відлагодження, оскільки окремі сервіси можуть взаємодіяти в складних способах.

– Забезпечення безпеки: важливо забезпечити безпеку кожного сервісу та правильно керувати доступом.

Загалом, обґрунтування вибору мікросервісної архітектури для розробки програмного модуля зберігання *MP*3 файлів підкреслює переваги гнучкості, масштабованості та незалежності окремих компонентів системи. Однак це також вимагає уваги до управління та координації, а також правильного проектування механізмів комунікації та безпеки.

1.5. Висновки до розділу

В межах розділу проведено вивчення принципів обробки *MP*3 файлів на віддалених серверах, огляд літератури та існуючих програмних рішень у сфері зберігання та обміну *MP*3 файлами, визначення основних задач та вимог до програмного модуля, а також обґрунтування вибору мікросервісної архітектури для розробки програмного модуля зберігання *MP*3 файлів. Проаналізовано такі ключові аспекти:

– проаналізовано принципи обробки *MP*3 файлів на віддалених серверах, включаючи їхню структуру та кодування, визначено вимоги до обробки та зберігання *MP*3 даних;

– проведено огляд літератури та існуючих програмних рішень, що стосуються обробки та зберігання аудіофайлів, зокрема *MP*3. Ретельно вивчено існуючі програмні рішення для зберігання та обміну *MP*3 файлами, а також виділено їх переваги та недоліки;

– визначено основні задачі та вимоги до програмного модуля за допомогою створення дерева функцій, визначено функціональні вимоги та встановлені пріоритети задач та функцій модуля;

– проведено ретельний аналіз концепції мікросервісної архітектури та обґрунтовано її вибір для розробки програмного модуля зберігання *MP*3 файлів. Вказано переваги такого підходу та визначено можливі виклики під час реалізації мікросервісної архітектури.

РОЗДІЛ 2
МОДЕЛЮВАННЯ РОБОТИ МОДУЛЯ ОБРОБКИ МУЗИЧНИХ ФАЙЛІВ В РЕКОМЕНДАЦІЙНІЙ СИСТЕМІ

Моделювання роботи модуля обробки музичних файлів в системі рекомендацій вимагає глибокого розуміння як особливостей формату *MP*3, так і алгоритмів, які використовуються для аналізу та генерації рекомендацій. *MP*3 – це формат стиснення аудіо, який дозволяє зменшити розмір звукових файлів без суттєвої втрати якості. Робота з цим форматом включає в себе різні етапи, починаючи з обробки метаданих та закінчуючи складними алгоритмами фільтрації.

2.1. Основні алгоритми для обробки *mp*3-файлів

Екстракція метаданих: Для коректної роботи рекомендаційних систем необхідно вилучити метадані з кожного *MP*3-файлу. Це включає інформацію про трек, артиста, альбом, рік випуску, жанр, і тривалість. Метадані можуть бути використані для організації музики, уточнення критеріїв пошуку та управління плейлистами.

Аналіз аудіо: Для більш глибокого аналізу контенту, система може використовувати алгоритми, що розглядають спектральні характеристики аудіо, такі як темп, тональність, енергія, та багато іншого. Ці дані можуть допомогти системі рекомендацій розпізнавати подібні музичні шаблони та пропонувати користувачам пісні зі схожими аудіо-характеристиками.

Колаборативна фільтрація: Це популярний алгоритм у системах рекомендацій, який використовує оцінки та вподобання інших користувачів для прогнозування інтересів поточного користувача. Він може бути реалізований через "*user-to-user*" або "*item-to-item*" підходи.

Контент-базована фільтрація: На відміну від колаборативної фільтрації, контент-базова фільтрація аналізує властивості самих пісень для генерації

рекомендацій. Якщо користувач високо оцінив певні треки, система шукатиме інші з подібними характеристиками.

Гібридні системи: Багато сучасних систем рекомендацій поєднують колаборативну та контент-базовану фільтрацію для покращення точності рекомендацій.

Кластеризація: Алгоритми кластеризації можуть групувати пісні за жанрами чи іншими аудіо-характеристиками, що дозволяє виявити приховані зв'язки між треками та рекомендувати їх на основі музичних особливостей.

Машинне навчання: Застосування моделей машинного навчання, таких як нейронні мережі, може допомогти у виявленні складних патернів у вподобаннях користувачів та в музичному контенті.

Ранжування на основі правил (*Rule-based Ranking*): Використання предвизначених правил (наприклад, новизна треку або популярність серед певної демографії) для ранжування рекомендацій.

Оцінка настрою (*Sentiment Analysis*): Аналіз текстів пісень або коментарів користувачів може допомогти визначити емоційний вміст, що може вплинути на рекомендації.

Підбір на основі контексту (*Context-aware Matching*): Врахування контексту користувача, такого як час доби або погода, може покращити відповідність рекомендацій поточному стану та потребам користувача.

Застосування цих алгоритмів у моделюванні роботи модуля обробки музичних файлів дозволяє системі рекомендацій не лише пропонувати пісні, які мають велику ймовірність сподобатися користувачу, але й виявляти нові музичні тренди, підтримувати динаміку змін у музичних перевагах та адаптуватися до постійно змінюваних вимог та інтересів аудиторії.

Екстракція метаданих з *MP*3-файлів є важливою частиною роботи над рекомендаційними системами, оскільки ці дані допомагають системі краще розуміти та організовувати музичний контент. Основні метадані, які можуть бути вилучені з *MP*3-файлів, включають наступне:

1. Назва треку: Інформація про назву пісні або треку, яка допомагає ідентифікувати конкретний музичний запис.

2. Артист: Ім'я або назва виконавця, який записав трек. Це дозволяє ідентифікувати хто виконує музику і може бути використано для пошуку і рекомендацій на основі виконавця.

3. Альбом: Назва альбому, в якому включений трек. Ця інформація допомагає в організації музики за альбомами і може бути використана для рекомендацій на основі альбому.

4. Рік випуску: Рік, коли трек був випущений. Ця інформація може бути важливою для рекомендацій старих або нових треків, а також для організації музики за роками.

5. Жанр: Категорія або стиль музики, до якої може бути віднесений трек. Ця інформація допомагає в класифікації музики та рекомендаціях за жанром.

6. Тривалість: Кількість часу, який триває трек у форматі годин, хвилин і секунд. Ця інформація може бути використана для обчислення загальної тривалості плейлисту або для рекомендацій треків з певною тривалістю.

7. Інші метадані: Додаткові дані можуть включати номер треку на альбомі, текст пісні, обкладинку альбому, бітрейт (якість аудіо), та іншу інформацію, яка може бути корисною для рекомендацій та організації музики.

Процес екстракції метаданих може бути автоматизованим і зазвичай виконується за допомогою програм або бібліотек для обробки аудіофайлів, таких як "*eyeD*3", "*mutagen*", "*pydub*" та інших. Ці інструменти дозволяють зчитувати інформацію з заголовків *MP*3-файлів та зберігати її у вигляді метаданих у вигляді текстових полів або об'єктів, що можна використовувати в рекомендаційних системах.

Після того як метадані вилучені, вони можуть бути використані для покращення організації музики, для побудови фільтрів та критеріїв пошуку, а також для створення рекомендацій, які враховують характеристики треків та інтереси користувачів.

Аналіз аудіо - це важлива складова роботи систем рекомендацій, яка дозволяє глибоко розуміти музичний контент і надавати користувачам більш точні та персоналізовані рекомендації. Для цього використовуються алгоритми обробки сигналу та аналізу аудіо. Основні аспекти аналізу аудіо включають наступне:

1. Спектральний аналіз: Спектральний аналіз аудіосигналу дозволяє розкласти звуковий сигнал на його складові частоти. Ця інформація дуже корисна для визначення акцентів у музиці та характеристик, таких як темп, тональність і інші акустичні особливості.

2. Темп: Аналіз темпу визначає швидкість музичного руху, тобто кількість ударів на хвилину (*BPM*). Ця інформація допомагає системі визначити, наскільки енергійною чи спокійною є музика, і може бути використана для рекомендацій схожих треків за темпом.

3. Тональність: Аналіз тональності визначає основну ноту або тон, який переважає в музичному матеріалі. Це може бути корисно для рекомендацій треків зі схожою тональністю або атмосферою.

4. Енергія: Вимірювання енергії в аудіосигналі може допомогти визначити, наскільки інтенсивною або емоційною є музика. Треки з високою енергією можуть рекомендуватися для активних ситуацій, тоді як треки з меншою енергією можуть бути відзначені для відпочинку.

5. Інші аудіо-характеристики: Додаткові характеристики можуть включати аналіз ритму, гучність, акустичні особливості (наприклад, наявність інструментів або вокалу) та інші параметри, які допомагають визначити структуру та характер музики.

За допомогою аналізу аудіо система рекомендацій може розпізнавати подібні музичні шаблони та структури в треках і рекомендувати користувачам пісні зі схожими аудіо-характеристиками. Наприклад, якщо користувач вподобав пісню зі швидким темпом і високою енергією, система може запропонувати інші треки з подібними аудіо-характеристиками для подальшого відтворення. Цей підхід допомагає створити більш деталізовані та зв'язані рекомендації для користувачів, враховуючи не тільки метадані, але й сам звуковий вміст музики.

2.2. Використання модуля обробки *mp*3 файлів в системі рекомендацій

Моделювання роботи модуля обробки музичних файлів є критичним етапом у створенні ефективної системи рекомендацій. Цей процес включає в себе не тільки технічну сторону, а й розробку бізнес-логіки, яка визначатиме, як система буде взаємодіяти з користувачами та як вона оброблятиме дані. У цьому контексті, моделювання бізнес-процесів та архітектури модуля є фундаментальним для досягнення головної мети системи – надання персоналізованих музичних рекомендацій.

Бізнес-процеси модуля обробки *mp*3 файлів включають в себе наступні ключові дії:

– Збір даних: Система збирає *mp*3 файли з різних джерел, включаючи виконавців, лейбли та користувачів. Важливо зазначити, що цей процес має відповідати авторським правам та іншим юридичним вимогам.

– Екстракція та обробка метаданих: Автоматизовані інструменти екстрагують інформацію про треки, як-от назву, артиста, альбом, жанр і тривалість. Ці дані є необхідними для подальшого аналізу та класифікації контенту.

– Аналіз аудіосигналів: Використовуючи *DSP* (*Digital Signal Processing*) алгоритми, система аналізує аудіосигнали для визначення характеристик, які не можна отримати з метаданих.

– Рекомендації: На основі отриманих даних та поведінки користувачів система генерує персоналізовані музичні рекомендації.

– Зворотний зв'язок від користувачів: Система аналізує зворотний зв'язок, отриманий від користувачів, для покращення точності рекомендацій.

Архітектура модуля обробки *mp*3 файлів включає в себе наступні компоненти:

– Інтерфейс користувача (*UI*): Дозволяє користувачам взаємодіяти з системою, включаючи завантаження треків, налаштування фільтрів та перегляд рекомендацій.

– Сервер прикладів (*Application Server*): Обробляє запити від *UI,* управляє бізнес-логікою та координує роботу інших компонентів системи.

– Модуль обробки даних (*Data Processing Module*): Виконує всі операції з аналізу аудіо та екстракції метаданих.

– База даних (*Database*): Зберігає всі необхідні дані, включаючи метадані, інформацію про користувачів та їх вподобання, а також історію взаємодії з системою.

– Система рекомендацій (*Recommendation Engine*): Використовує алгоритми машинного навчання та статистичний аналіз для генерації рекомендацій на основі зібраних даних.

– *API* сторонніх сервісів (*Third-party Services API*): Забезпечує інтеграцію з зовнішніми музичними платформами та соціальними мережами для збору додаткових даних або надання доступу до музичного контенту.

Для успішного моделювання бізнес-процесів і архітектури, необхідно враховувати як технічні так і комерційні аспекти роботи системи. Технічні аспекти включають вибір найкращих технологій та алгоритмів, тоді як комерційні аспекти охоплюють забезпечення відповідності продукту ринковим потребам та створення вартості для користувачів та власників контенту. Сучасні підходи до розробки рекомендаційних систем часто включають *Agile*-методології, які сприяють гнучкості та адаптивності процесу розробки, та *DevOps* практики для забезпечення неперервної інтеграції та доставки (*CI*/*CD*), що дозволяє швидко впроваджувати нововведення та реагувати на зміни на ринку.

Використання модуля обробки *mp*3 файлів в системі рекомендацій є ключовим компонентом для створення ефективної та вдосконаленої платформи музичних рекомендацій. Цей модуль відіграє критичну роль у виявленні, аналізі та розумінні музичних треків, щоб забезпечити користувачам персоналізовані та

задоволені музичні рекомендації. Нижче наведено докладний огляд теоретичних аспектів використання модуля обробки *mp*3 файлів у системі рекомендацій:

1. Аналіз та обробка аудіофайлів: Модуль обробки *mp*3 файлів використовується для аналізу аудіофайлів у форматі *MP*3. Це включає в себе розшифрування звукових даних, визначення параметрів аудіо, таких як бітрейт, частота дискретизації і стерео-доріжки. Крім того, він виконує аналіз спектральних характеристик для визначення звукових особливостей треку, таких як темп, ключова тональність та інші акустичні параметри.

2. Виявлення музичних характеристик: Модуль обробки *mp*3 файлів визначає різні музичні характеристики, такі як ритм, тимбральні особливості, інтенсивність, гармонічну структуру тощо. Ці характеристики допомагають відокремити музичні жанри, розпізнавати настрій пісень і краще розуміти смак користувача.

3. Мелодійна та гармонічна аналіз: Модуль може виявляти основні мелодійні та гармонічні структури в музиці, що дозволяє системі рекомендацій розуміти, які музичні елементи приваблюють користувача. Це може бути корисним для пошуку пісень або артистів зі схожими музичними особливостями.

4. Аналіз текстової інформації: Модуль також може аналізувати текстову інформацію, пов'язану з музичними треками, таку як назва пісні, виконавець, тексти пісень та метадані. Ця інформація може бути використана для покращення рекомендацій, зокрема для збагачення змісту рекомендацій та пошуку музичних пісень за ключовими словами.

5. Застосування машинного навчання: Використання модуля обробки *mp*3 файлів також може включати в себе машинне навчання для аналізу та вивчення користувальницьких уподобань. Модель може навчатися розпізнавати патерни у виборі музики користувачів і робити рекомендації на основі цих знань.

6. Система рейтингів і зворотного зв'язку: Для поліпшення точності рекомендацій можна використовувати систему збору даних про рейтинги користувачів та зворотний зв'язок. Модуль обробки *mp*3 файлів може допомагати в аналізі даних з оцінками треків користувачами.

7. Персоналізовані рекомендації: За допомогою зібраних даних і аналізу музичних характеристик модуль може генерувати персоналізовані рекомендації для кожного користувача. Це дозволяє забезпечити більш індивідуалізований підхід до музичних рекомендацій.

Використання модуля обробки *mp*3 файлів в системі рекомендацій відкриває безмежні можливості для покращення музичного досвіду користувачів, забезпечуючи їх більш точними, розуміючими та персоналізованими рекомендаціями. Теоретичні аспекти цього підходу допомагають побудувати ефективну систему рекомендацій, яка задовольняє потреби різних користувачів у світі музики.

Інтеграція модуля обробки *mp*3 файлів в систему рекомендацій відкриває широкий спектр можливостей для створення більш точних, цікавих та персоналізованих рекомендацій для користувачів. Ось деякі можливості інтеграції цього модуля:

1. Поліпшення аналізу смаку користувача: Модуль обробки *mp*3 файлів дозволяє системі рекомендацій краще розуміти смак користувача, аналізуючи їхні вибори у музиці. Наприклад, він може виявити, які жанри, інструменти або артисти найчастіше прослуховує користувач, і враховувати це в рекомендаціях.

2. Пошук музики за аналогією: Модуль може допомогти користувачам знаходити нову музику, подібну до тих, які їм вже сподобалися. За допомогою аналізу музичних характеристик і порівняння з вже відомими треками, система може запропонувати аналогічну музику, яка може бути цікавою.

3. Динамічні рекомендації: Модуль може враховувати зміни в музичних уподобаннях користувача з часом. Це дозволяє системі надавати рекомендації, які відповідають поточному стану і настрою користувача.

4. Рекомендації на основі аналізу текстів пісень: Якщо модуль обробки *mp*3 файлів аналізує тексти пісень, система може враховувати текстовий контент пісень при генерації рекомендацій. Наприклад, вона може запропонувати пісні зі схожою тематикою або текстовими змістами.

5. Мікросервісна архітектура: Інтеграція модуля обробки *mp*3 файлів через мікросервіси дозволяє легко масштабувати та оновлювати функціональність обробки музичних файлів без впливу на інші компоненти системи рекомендацій.

6. Модельний навчання і покращення алгоритмів рекомендацій: Дані, зібрані модулем обробки *mp*3 файлів, можуть використовуватися для навчання машинного навчання та покращення алгоритмів рекомендацій. Це може підвищити точність та ефективність системи рекомендацій з часом.

7. Контент-базовані рекомендації: Інтеграція модуля обробки *mp*3 файлів дозволяє створювати контент-базовані рекомендації, які враховують акустичні та музичні особливості треків, а не тільки історію перегляду користувача.

Інтеграція модуля обробки *mp*3 файлів в систему рекомендацій робить її більш потужною та інтелектуальною, дозволяючи користувачам отримувати більш цікаві та персоналізовані музичні рекомендації. Можливості цієї інтеграції безмежні і дозволяють постійно покращувати музичний досвід користувачів.

Системи рекомендацій музичних файлів є важливою частиною сучасної музичної індустрії та стрімінгових платформ. Вони допомагають користувачам знаходити нову музику, яка відповідає їхнім смакам і уподобанням. Існують різні типи систем рекомендацій музичних файлів, включаючи:

1. Системи рекомендацій на основі контенту:

– Акустичні рекомендації: Вони використовують аналіз акустичних особливостей музики, таких як ритм, темп, інструментальний склад тощо, для знаходження схожих треків.

– Текстові рекомендації: Вони аналізують тексти пісень та використовують текстовий контент для знаходження схожих за змістом треків.

– Мелодійний аналіз: Використовують аналіз мелодійних структур, а також гармонійні та музичні особливості для рекомендацій.

2. Системи рекомендацій на основі колаборативного фільтрування:

– Фільтрування за схожістю користувачів: Вони порівнюють користувачів зі схожими музичними уподобаннями і рекомендують музику, яка сподобалася іншим користувачам зі схожим смаком.

– Фільтрування за схожістю об'єктів: Вони порівнюють музичні об'єкти (пісні, альбоми) за споживачами та рекомендують об'єкти, які схожі на ті, що користувач вже споживав.

3. Системи гібридного рекомендаційного підходу:

– Комбіновані системи: Вони поєднують підходи контентного та колаборативного фільтрування для надання більш точних рекомендацій.

– Модельні гібриди: Вони використовують машинне навчання для створення моделей, які комбінують різні підходи для рекомендацій.

4. Системи рекомендацій на основі змістового фільтрування:

– Рекомендації на основі історії слухання: Вони враховують історію слухання користувача та рекомендують музику, схожу на ту, яку користувач вже слухав.

– Рекомендації на основі рейтингів і відгуків: Вони використовують інформацію про рейтинги та відгуки користувачів для рекомендацій.

5. Системи рекомендацій на основі внутрішньої моделі: Вони будують модель інтересів користувача та використовують її для рекомендацій на основі внутрішнього розуміння музичних характеристик.

6. Системи рекомендацій з використанням метаданих: Вони враховують метадані музичних треків, такі як жанр, виконавець, рік видання і т. д., для рекомендацій.

7. Системи рекомендацій на основі спільної активності: Вони аналізують спільну активність користувачів, таку як створення плейлистів або спільні рейтинги, для рекомендацій.

Кожен з цих типів систем рекомендацій має свої переваги та недоліки і може бути використаний залежно від конкретних потреб і можливостей платформи музичної рекомендації. Більшість сучасних систем рекомендацій музичних файлів використовують комбінацію різних підходів для надання користувачам найкращого музичного досвіду.

Системи рекомендацій на основі контенту використовують властивості самого контенту (музичного матеріалу) для створення рекомендацій. Цей підхід

може бути особливо корисним, коли немає достатньої історії слухання користувача або коли користувачі шукають нову музику, схожу на ту, яку вони вже полюбили. Важливі підкатегорії систем рекомендацій на основі контенту включають:

1. Акустичні рекомендації:

– Аналіз ритму: Системи можуть аналізувати ритм та темп музики, визначаючи, чи є трек енергійним, спокійним або танцювальним. Користувачам рекомендуються треки зі схожими ритмічними характеристиками.

– Інструментальний склад: Визначення інструментального складу пісні, такого як використані музичні інструменти, дозволяє рекомендувати треки з подібним музичним звучанням.

2. Текстові рекомендації:

– Аналіз текстів пісень: Системи можуть аналізувати тексти пісень і шукати спільні слова, теми або ключові слова. Наприклад, якщо користувач полюбив пісні з текстами про море або подорожі, система може рекомендувати треки зі схожою тематикою.

– Використання семантики: Деякі системи навіть можуть застосовувати аналіз семантики слів для знаходження пісень з подібним змістом, навіть якщо вони не мають точно співпадаючих слів.

3. Мелодійний аналіз:

– Аналіз мелодійних структур: Ця підкатегорія використовує аналіз мелодій, включаючи визначення мелодійних ліній та гармонійних особливостей. Наприклад, якщо користувач полюбив трек з характерною мелодією на гітарі, система може рекомендувати інші пісні з подібним мелодійним структурами.

Ці методи дозволяють створювати рекомендації на основі конкретних аспектів музичного контенту, а не тільки на основі споживацьких патернів користувачів або метаданих. Однак важливо враховувати, що аналіз музичного контенту може бути складним завданням, і точність рекомендацій залежить від якості аналізу музики і доступу до адекватних акустичних і текстових даних. На

сьогоднішній день багато музичних платформ поєднують цей підхід з іншими методами, щоб надавати користувачам більш точні та різноманітні рекомендації.

Системи рекомендацій на основі колаборативного фільтрування (*Collaborative Filtering*) – це один з найпоширеніших підходів до створення рекомендаційних систем, який базується на ідейці спільної активності та схожості між користувачами і об'єктами (у нашому випадку, музичними треками). Цей підхід включає дві головні підкатегорії:

1. Фільтрування за схожістю користувачів:

– Порівняння користувачів: В цьому підході система аналізує історію споживання музики користувачів і порівнює їхні уподобання. Користувачі, які мають схожі смаки та споживали схожу музику, вважаються "схожими" користувачами.

– Рекомендація на основі користувачів: Після визначення схожості користувачів, система може рекомендувати музику, яка була сподобалася іншим користувачам з аналогічними смаками. Наприклад, якщо користувач *A* і користувач *B* мають подібний смак і користувач *A* вподобав трек *X,* то користувачу *B* може бути запропоновано трек *X*.

2. Фільтрування за схожістю об'єктів:

– Порівняння об'єктів (музичних треків): У цьому підході система порівнює музичні треки між собою на основі того, які користувачі споживали ці треки. Треки, які часто споживаються однаковими користувачами, вважаються "схожими" об'єктами.

– Рекомендація на основі об'єктів: Після визначення схожості між музичними треками, система може рекомендувати користувачам треки, які схожі на ті, які вони вже споживали. Наприклад, якщо користувач слухав трек *Y* і трек *Z* має схожий акустичний профіль з треком *Y,* то трек *Z* може бути рекомендований користувачеві.

Обидва підходи до колаборативного фільтрування вимагають матриці взаємодії користувач-об'єкт (наприклад, матриці рейтингів, де кожен рядок представляє користувача, а кожен стовпчик - об'єкт). Вони можуть бути

реалізовані за допомогою різних алгоритмів, таких як *Singular Value Decomposition* (*SVD*), *Matrix Factorization,* або методів на основі згорткових нейронних мереж.

Важливою перевагою колаборативного фільтрування є здатність робити рекомендації навіть у випадку, коли немає додаткової інформації про музичний контент або користувачів. Він покликаний знайти схожості, які можуть бути невидимими за іншими методами. Однак він також стикається з викликами, такими як холодний старт для нових користувачів і об'єктів, а також проблемами розріджених даних, коли матриця взаємодії має багато пропущених значень.

Системи гібридного рекомендаційного підходу - це способи поєднання різних методів рекомендацій, таких як контентне фільтрування та колаборативне фільтрування, для досягнення більш точних та різноманітних рекомендацій. Цей підхід дозволяє вирішити деякі з важливих проблем інших методів і використовувати їх переваги. Важливими підкатегоріями гібридних систем рекомендацій є:

1. Комбіновані системи:

– Поєднання контенту і колаборативного фільтрування: Комбіновані системи використовують як контентний, так і колаборативний підхід. Наприклад, вони можуть враховувати акустичні характеристики музичних треків, а також історію слухання користувача та його схожість з іншими користувачами. Такий підхід дозволяє покращити якість рекомендацій, особливо в умовах обмеженої історії споживання або обмеженої інформації про користувача.

2. Модельні гібриди:

– Використання машинного навчання: Модельні гібриди використовують методи машинного навчання для створення моделей, які комбінують різні підходи до рекомендацій. Ці моделі можуть враховувати багато факторів, таких як акустичні характеристики, тексти пісень, історія слухання, а також інші додаткові дані. Модель може навчитися, як найкраще комбінувати ці фактори для надання найкращих рекомендацій.

Гібридні системи рекомендацій дозволяють подолати деякі обмеження і недоліки інших методів. Наприклад, вони можуть бути корисні при холодному

старті, коли немає досвіду користувачів або інформації про об'єкти. Крім того, вони можуть покращити якість рекомендацій за рахунок використання різноманітних джерел інформації та методів аналізу. Однак створення та оптимізація гібридних систем може бути складнішим завданням, ніж інших підходів, і вимагає більше обчислювальних ресурсів та експертного знання для розробки та налаштування моделей.

Системи рекомендацій на основі змістового фільтрування (*Content-Based Filtering*) використовують інформацію про вміст або характеристики об'єктів, таких як музичні треки, для надання рекомендацій користувачам. Цей підхід базується на теорії, що якщо користувач споживав певні об'єкти і вони мали певні характеристики, то інші об'єкти з аналогічними характеристиками також можуть сподобатися користувачу. Два основних підходи до систем рекомендацій на основі змістового фільтрування:

1. Рекомендації на основі історії слухання:

– Аналіз історії слухання користувача: Система вивчає, які музичні треки користувач вже слухав або споживав і які характеристики (наприклад, жанр, виконавець, акустичні особливості) були присутні в цих треках.

– Рекомендації на основі подібності характеристик: Система потім шукає інші музичні треки зі схожими характеристиками та рекомендує їх користувачу. Наприклад, якщо користувач часто слухав рок-музику, система може рекомендувати інші треки у жанрі року.

2. Рекомендації на основі рейтингів і відгуків:

– Збір рейтингів та відгуків користувачів: Система може використовувати рейтинги, які користувачі виставляють музичним трекам, або відгуки, які вони пишуть про них.

– Рекомендації на основі подібності до популярних треків або відгуків: Система може рекомендувати користувачеві треки, які отримали високі рейтинги або позитивні відгуки від інших користувачів, і вважаються популярними.

Перевагою систем рекомендацій на основі змістового фільтрування є те, що вони можуть надавати рекомендації навіть для нових користувачів, які ще не мали

історії слухання або не залишали рейтингів та відгуків. Вони також можуть бути корисними для рекомендацій у випадку, коли користувач шукає треки з певними характеристиками, які він зараз вподобав би послухати. Однак ці системи також можуть бути обмежені, оскільки вони враховують тільки характеристики об'єктів і можуть не враховувати смаки користувачів, які можуть змінюватися з часом.

Системи рекомендацій на основі внутрішньої моделі (*Model-Based Recommendation Systems*) є іншим підходом до створення рекомендацій, які будують модель інтересів користувача та об'єктів рекомендації для подальшого розрахунку рекомендацій на основі цих моделей. Основна ідея полягає в тому, щоб мати внутрішню модель, яка репрезентує як користувачів, так і об'єкти (музичні треки) в просторі ознак чи характеристик. Основні аспекти цих систем:

1. Побудова моделей інтересів користувача:

– Внутрішнє представлення користувачів: Система створює внутрішнє представлення кожного користувача, яке відображає його інтереси і уподобання щодо музики. Це представлення може бути вектором чи матрицею, де кожна компонента відображає ступінь інтересу користувача до певних музичних характеристик або об'єктів.

2. Побудова моделей об'єктів:

– Внутрішнє представлення об'єктів: Система також створює внутрішнє представлення музичних треків, альбомів або виконавців. Це представлення може містити інформацію про характеристики музичного контенту, такі як жанр, темп, інструментальний склад тощо.

3. Розрахунок рекомендацій на основі моделей:

– Прогнозування інтересів: Система використовує побудовані моделі для прогнозування інтересів користувача до різних об'єктів. На підставі цих прогнозів розраховуються рекомендації. Наприклад, можна визначити, які музичні треки найбільше відповідають інтересам конкретного користувача, основуючись на його моделі інтересів та моделі об'єктів.

Переваги систем рекомендацій на основі внутрішньої моделі включають здатність до рекомендацій навіть для нових користувачів або об'єктів, оскільки

моделі можуть бути побудовані на основі наявних даних без необхідності знання історії слухання чи рейтингів. Крім того, ці системи можуть враховувати складні та неочевидні взаємозв'язки між музичними характеристиками та інтересами користувачів.

Однак створення та оптимізація цих моделей може бути вимогливим завданням і вимагати великої кількості даних та обчислювальних ресурсів. Також важливо підтримувати та оновлювати моделі, оскільки інтереси користувачів можуть змінюватися з часом.

Системи рекомендацій з використанням метаданих:

– Використання метаданих музичних треків: Ці системи аналізують метадані, що описують музичні треки, для створення рекомендацій. Метадані можуть включати такі характеристики, як жанр, виконавець, альбом, рік видання, тривалість, темп, мова і багато інших. Система враховує ці характеристики та зіставляє їх із смаками та історією слухання користувача для рекомендацій.

– Жанрова рекомендація: Однією з популярних підкатегорій цих систем є рекомендація на основі жанрів. Система може рекомендувати музику в тому ж жанрі, який користувач слухав раніше, або пропонувати схожі жанри, які можуть сподобатися.

– Рекомендація на основі виконавців і альбомів: Системи також можуть бути налаштовані для рекомендацій на основі уподобань до конкретних виконавців або альбомів. Якщо користувач зазвичай слухає певного виконавця, система може рекомендувати інші треки цього виконавця або схожих артистів.

7. Системи рекомендацій на основі спільної активності:

– Аналіз спільної активності користувачів: Ці системи аналізують дані про активність користувачів, такі як створені плейлисти, спільні рейтинги, додавання треків до улюблених, а також рейтинги пісень і відгуки. Вони виявляють спільну активність між користувачами, яка може вказувати на схожість смаків.

– Рекомендація на основі активності інших користувачів: Система може рекомендувати користувачеві музику, яка сподобалася іншим користувачам зі схожими смаками. Наприклад, якщо два користувачі створили плейлисти із

схожими треками, то система може рекомендувати треки з одного плейлиста іншому користувачеві.

Ці системи рекомендацій використовують важливу інформацію про музичний контент (метадані) та активність користувачів для створення рекомендацій, які можуть бути більш зрозумілими і легкими для використання для користувачів. Вони дозволяють враховувати більше контексту та уподобань користувачів для створення більш персоналізованих рекомендацій.

Популярні *LTR*-алгоритми (рис. 2.1).

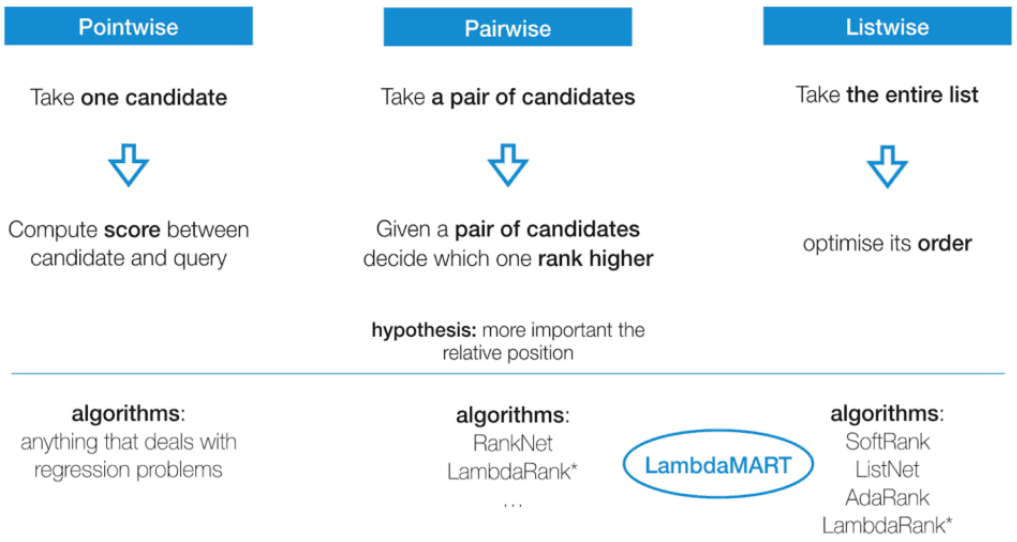

Рис. 2.1. Популярні *LTR*-алгоритми

LTR (*Learning to Rank*) є підходом в машинному навчанні, який використовується для вирішення задач ранжування. Він зосереджується на створенні моделей, які можуть передбачати оптимальний порядок серед набору елементів. *LTR* застосовується в різноманітних областях, включаючи пошукові системи, персоналізовані рекомендації та рекламні технології. Ось декілька популярних *LTR*-алгоритмів:

1. *Pointwise LTR Algorithms*:

– Ці алгоритми розглядають ранжування як задачу регресії або класифікації на рівні окремих об'єктів. Моделі, такі як логістична регресія або *SVM* (метод

опорних векторів), використовуються для прогнозування індивідуальної "важливості" або "релевантності" кожного елемента.

2. *Pairwise LTR Algorithms*:

– Парні алгоритми фокусуються на правильному порівнянні пар об'єктів. Вони намагаються мінімізувати кількість помилково розташованих пар в ранжуванні. *RankNet* та *RankBoost* є прикладами таких алгоритмів, які тренують модель, щоб точно визначити, який об'єкт з пари є більш релевантним.

3. *Listwise LTR Algorithms*:

– Лістові алгоритми розглядають задачу ранжування у її цілісності, оптимізуючи порядок всього списку об'єктів. Вони використовують такі метрики, як *NDCG* (*Normalized Discounted Cumulative Gain*), для оптимізації порядку цілого списку. *LambdaMART,* який є поєднанням *LambdaRank* та *MART* (*Multiple Additive Regression Trees*), є одним з найбільш використовуваних лістових алгоритмів в індустрії.

4. *Neural LTR Algorithms*:

– Це відносно новий клас алгоритмів, який використовує нейронні мережі для ранжування. Ці моделі можуть автоматично виявляти складні взаємозалежності між ознаками та ефективно вирішувати задачі ранжування. Прикладами можуть бути *RankBrain* від *Google* або нейронні мережі, що використовуються в системах рекомендацій *Netflix*.

Використання *LTR*-алгоритмів у системах рекомендацій дозволяє більш точно ранжувати музичні файли, враховуючи не тільки вподобання користувачів, але й контекстуальні фактори, що можуть впливати на музичні переваги. Це може включати час доби, емоційний стан користувача, соціальні взаємодії (наприклад, пісні, які слухають друзі), і навіть зовнішні події, які можуть вплинути на популярність певного жанру або артиста.

На рис. 2.2-2.5 представлено кілька архітектур глибинного навчання, що застосовуються в системах рекомендацій. Ці моделі демонструють різні підходи до обробки та аналізу великих наборів даних з метою надання персоналізованих

рекомендацій користувачам. Давайте розглянемо особливості кожної з цих архітектур:

1. *Wide & Deep Models*: Ця архітектура поєднує в собі "широкі" лінійні моделі та "глибокі" нейронні мережі для забезпечення як широкого охоплення функцій, так і глибокого аналізу властивостей користувачів і товарів. Широкі моделі ефективні для запам'ятовування різноманітних взаємодій, тоді як глибокі моделі використовуються для узагальнення невідомих взаємодій зі складними абстрактними функціями.

2. *Logistic Loss & ReLU Networks*: Модель використовує *ReLU* (*rectified linear unit*) активаційні функції в глибоких шарах нейронної мережі для виявлення нелінійних взаємозв'язків між функціями. *Logistic loss* використовується як функція втрат для класифікаційних завдань, де модель прогнозує ймовірність того, що користувач вибере певний товар або послугу.

3. *Embedding & Attention Mechanisms*: Системи рекомендацій часто використовують вбудовування (*embeddings*) для перетворення категоріальних даних у високовимірний простір, де можна виконувати векторні обчислення. Механізми уваги (*attention mechanisms*) в моделях, як *SASRec,* дозволяють моделі зосереджуватися на найважливіших частинах вхідних даних, наприклад, на останніх взаємодіях користувача.

4. *Sequence-Aware Recommender Systems*: Моделі, як *FISSA,* призначені для врахування послідовності дій користувачів. Вони моделюють взаємодії як послідовність, що дозволяє виявляти шаблони поведінки та зрозуміти динаміку інтересів користувачів.

5. *Collaborative Memory Networks*: Архітектура *CMN* використовує спільну пам'ять для узагальнення інформації про взаємодії між користувачами та товари, використовуючи її для генерації рекомендацій.

Кожна з цих моделей використовує унікальний набір алгоритмів та математичних операцій для обробки даних, враховуючи історичні дані, переваги користувачів та інші фактори. Вони можуть включати техніки машинного навчання, статистичного аналізу, інтелектуального аналізу даних, та

використовувати різноманітні джерела даних для створення комплексної картини інтересів користувачів.

Усі ці моделі розроблені з метою покращення точності та релевантності рекомендацій, що надаються користувачам, і використовують складні обчислювальні архітектури для обробки великих наборів даних в режимі реального часу. Завдяки глибокому навчанню та інноваційним підходам у сфері штучного інтелекту ці системи постійно еволюціонують, відкриваючи нові можливості для персоналізації користувацького досвіду.

Комбінована модель рекомендацій дозволяє отримувати якісні та персоналізовані рекомендації додатків для користувачів магазину *Google Play* (рис. 2.2).

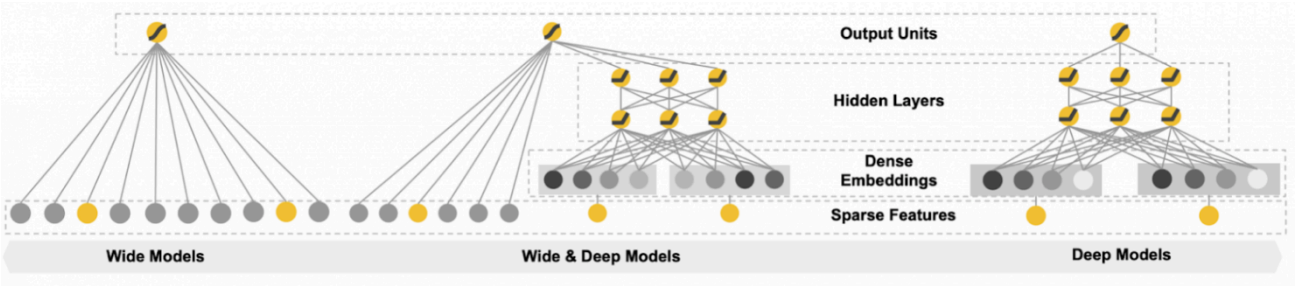

Рис. 2.2. Модель рекомендацій *Google Play*

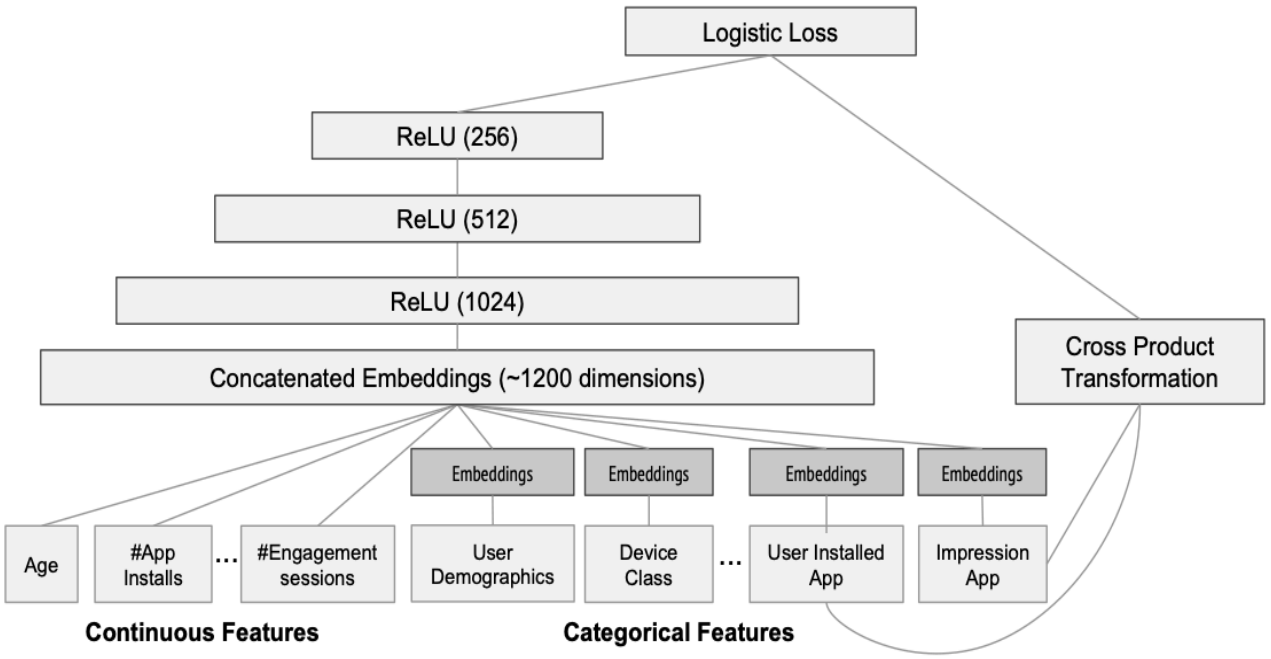

Рис. 2.3. Модель *Deep Component*

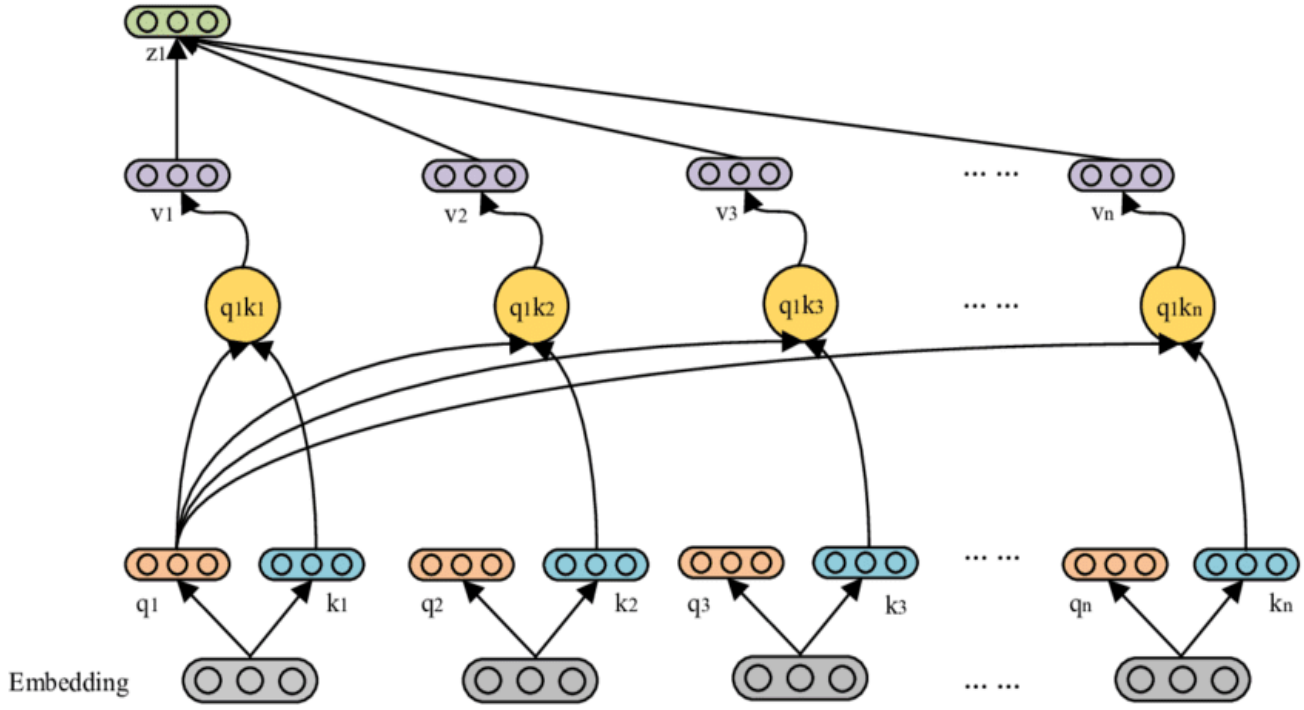

Рис. 2.4. Візуалізація роботи алгоритму *SASRec* [24]

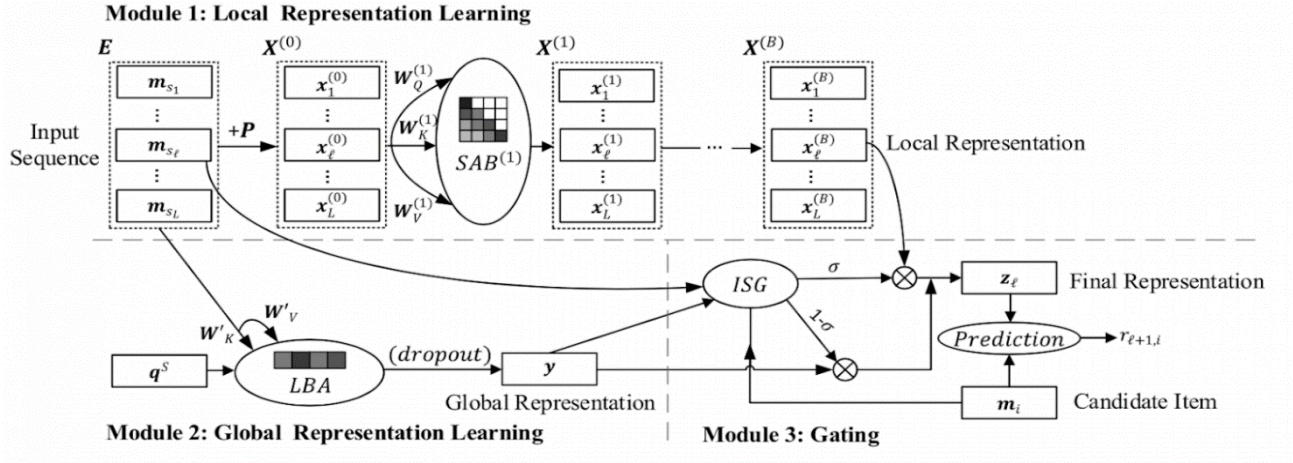

Рис. 2.5. Модель *FISSA*

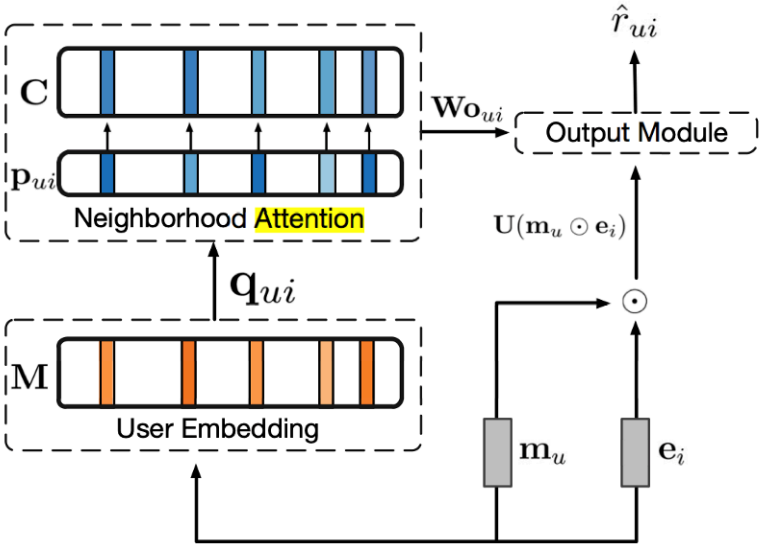

Рис. 2.6. Модель *CMN*

Багато рекомендаційних моделей так чи інакше приходять до ідеї гібридних моделей. Ймовірно, причина в тому, що хороша рекомендаційна система повинна підходити великому числу різних у поведінці користувачів і вирішувати одночасно кілька завдань (які вже перераховувалися в першій частині).

Дерево функцій модуля обробки *mp*3 файлів в системі рекомендацій (рис. 2.7).

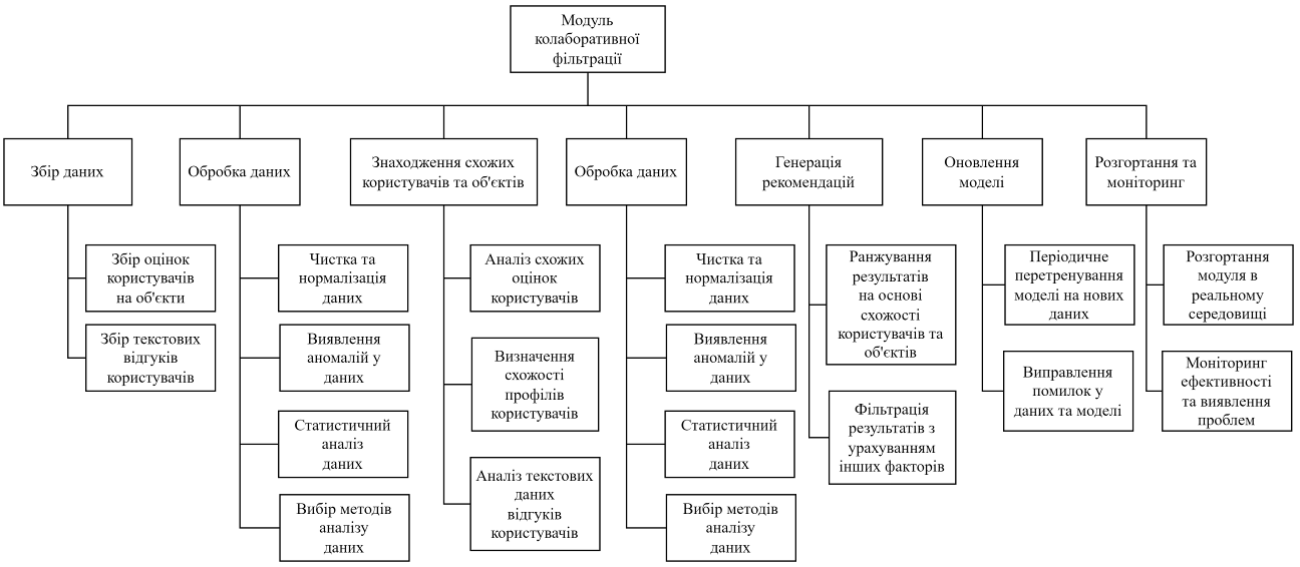

Рис. 2.7. Дерево функцій модуля колаборативної фільтрації

Загальний опис діаграми роботи колаборативної системи рекомендацій у форматі *IDEF*0 для описаного функціоналу (рис. 2.8):

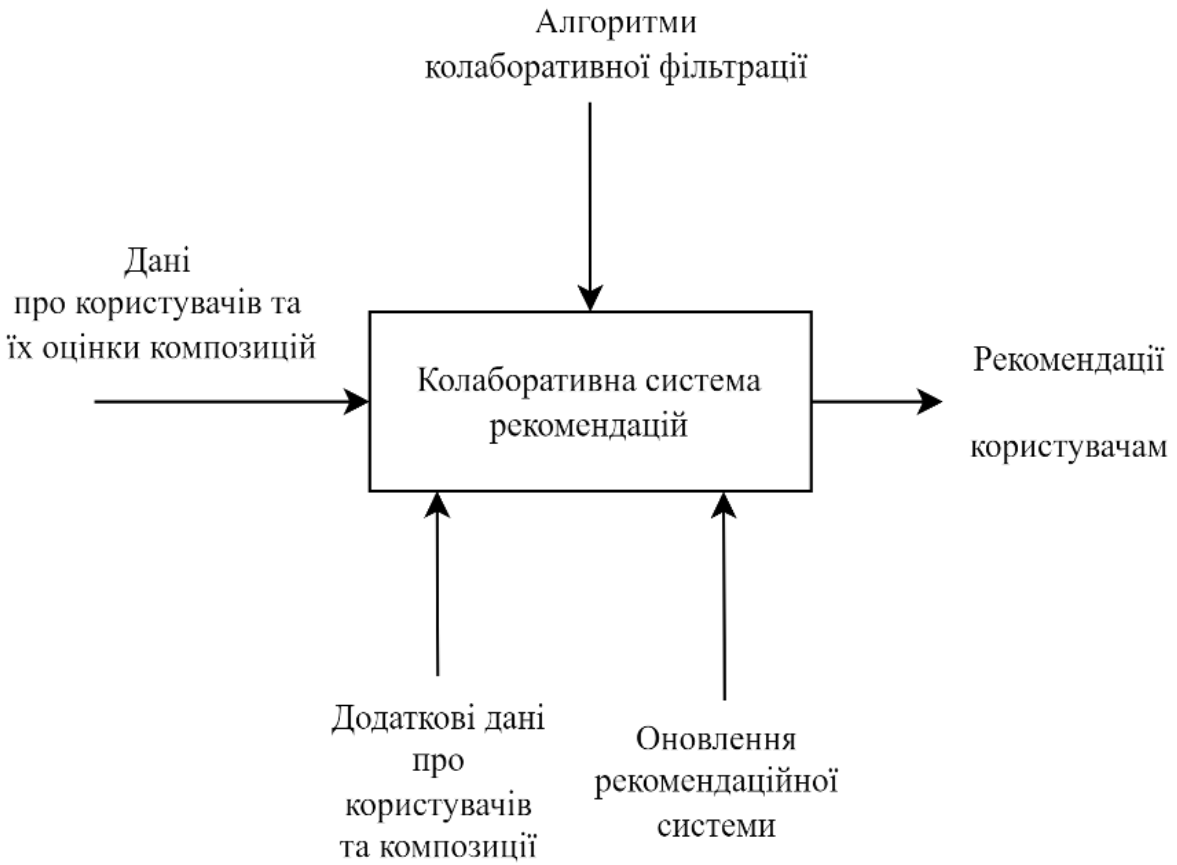

Рис. 2.8. Діаграма роботи програмного модуля обробки *mp*3-файлів в системі рекомендацій

Для подальшої деталізації та декомпозиції функції *A*1 (рис. 2.9).

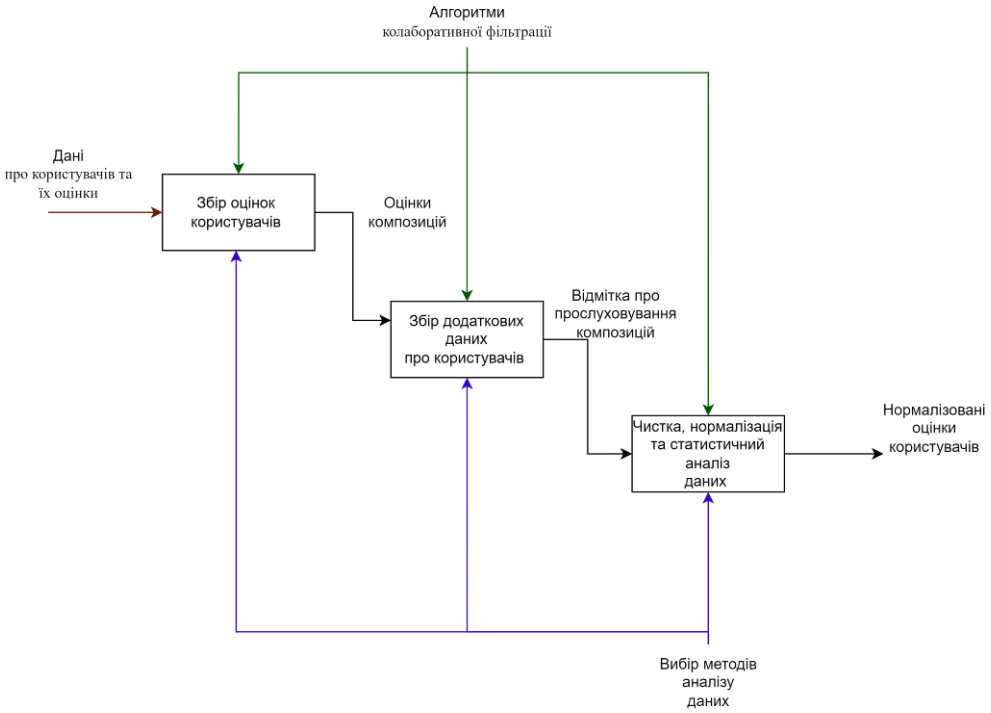

Рис. 2.9. Деталізація роботи програмного модуля обробки *mp*3-файлів в системі рекомендацій

Для подальшої деталізації та декомпозиції функції «*A*2. Знаходження схожих користувачів та об'єктів» можна використати діаграму рівня *A*1 (рис. 2.10).

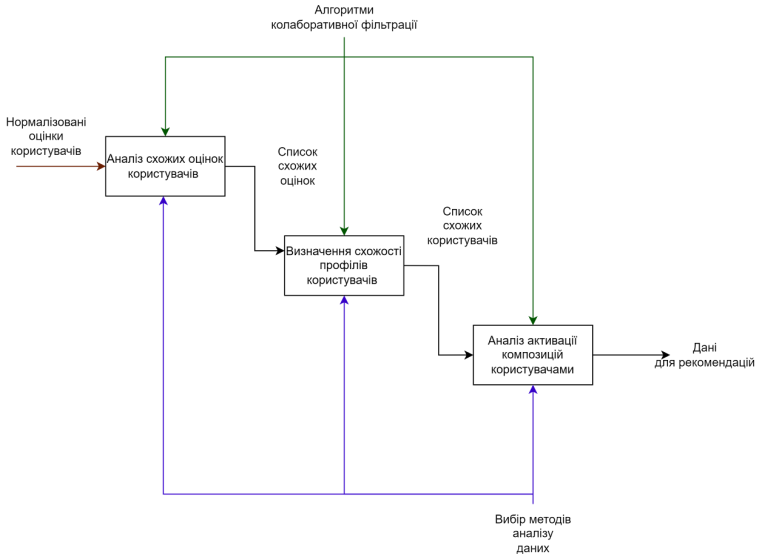

Рис. 2.10. Деталізація роботи програмного модуля обробки *mp*3-файлів в системі рекомендацій

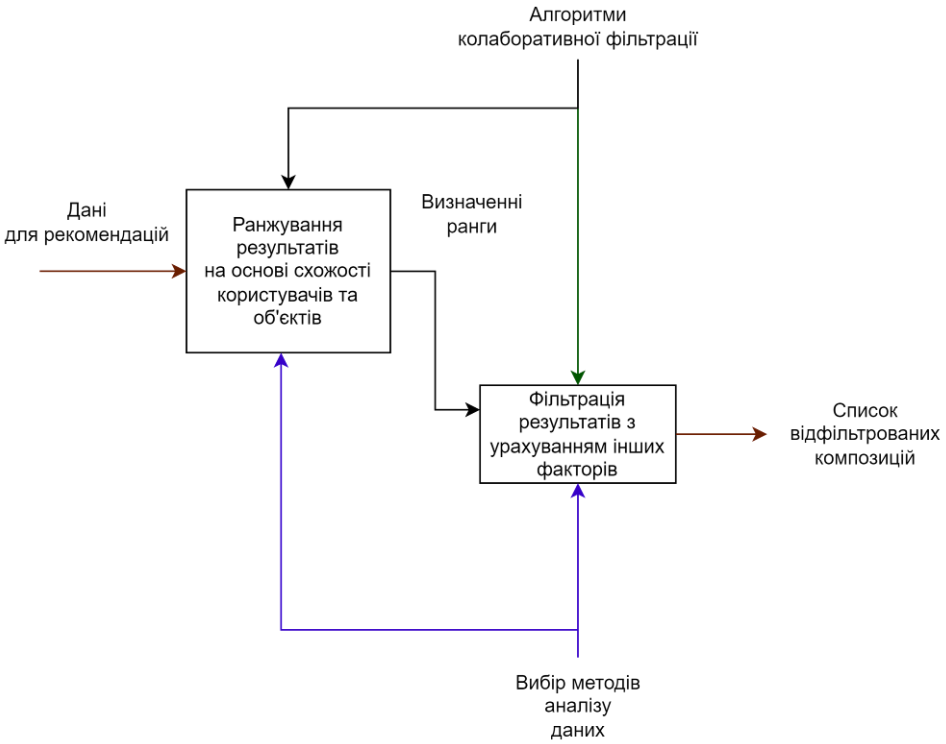

Рис. 2.11. Деталізація роботи програмного модуля обробки *mp*3-файлів в системі рекомендацій

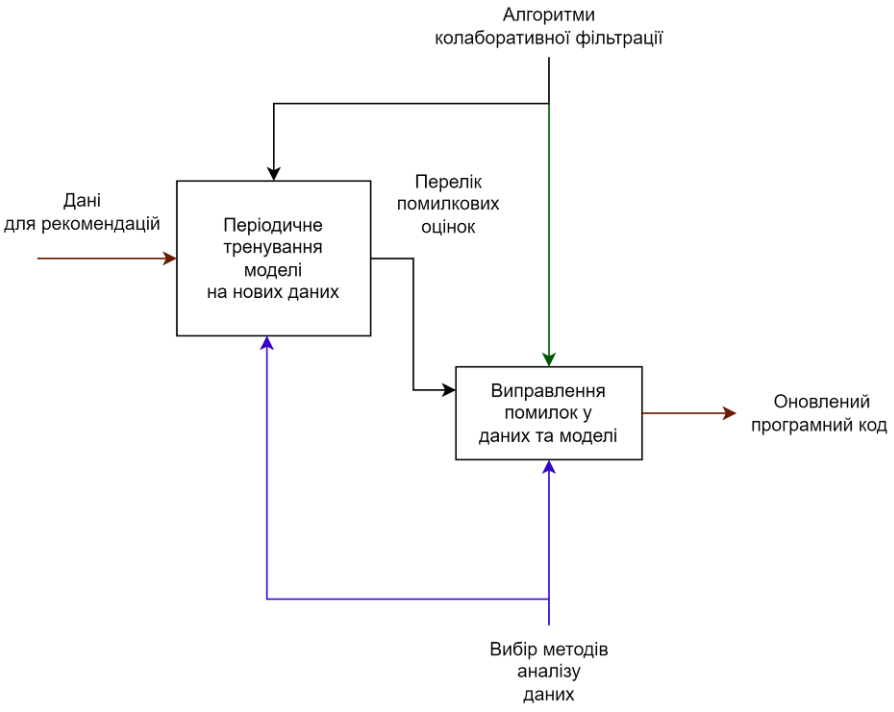

Рис. 2.12. Деталізація роботи програмного модуля обробки *mp*3-файлів в системі рекомендацій

Ці функції можна представити на діаграмі використання, яка відображатиме можливість користувачів здійснювати певні дії в системі (рис. 2.13).

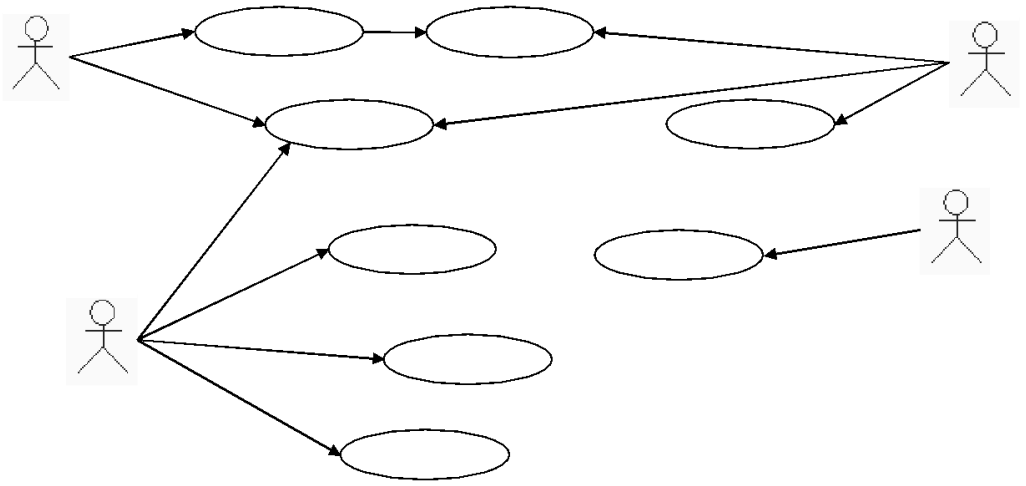

Рис. 2.13. Діаграма використання

2.3. Проєктування бази даних для підтримки роботи системи рекомендацій

Проєктування бази даних для підтримки роботи системи рекомендацій є важливим завданням, яке вимагає ретельного планування та розуміння бізнес-процесів системи. Основна мета проєктування полягає в тому, щоб створити структуру, яка буде ефективною для зберігання, видачі, обробки та аналізу даних. Оптимізована база даних дозволяє системі швидко реагувати на запити користувачів і пропонувати актуальні рекомендації. Нижче наведено основні аспекти проєктування бази даних для системи рекомендацій.

Першим кроком у проєктуванні бази даних є визначення ключових сутностей, які будуть представлені в системі. Для музичної системи рекомендацій, такими сутностями можуть бути:

– Користувачі (*Users*): Інформація про користувачів, включаючи їх переваги, історію слухань, рейтинги пісень та особисті дані.

– Треки (*Tracks*): Дані про музичні композиції, включаючи назви, виконавців, альбоми, жанри та інші метадані.

– Альбоми (*Albums*) та Виконавці (*Artists*): Інформація про альбоми та артистів, до яких належать треки.

– Плейлисти (*Playlists*): Списки відтворення, які користувачі створюють з улюблених треків.

– Рекомендації (*Recommendations*): Записи рекомендацій, що генеруються системою для кожного користувача.

Після визначення сутностей розробляється схема бази даних, що включає в себе таблиці та зв'язки між ними. Сучасні системи рекомендацій часто використовують реляційні бази даних через їх здатність ефективно обробляти запити з великою кількістю взаємозв'язків, а також нереляційні бази даних для гнучкості та масштабування.

Вибір конкретної СУБД залежить від багатьох факторів, включаючи об'єм даних, необхідну швидкість обробки, гнучкість схеми даних, а також вимоги безпеки і масштабування. Популярними варіантами для систем рекомендацій є *PostgreSQL, MySQL, MongoDB, Cassandra,* і *Redis*.

Нормалізація даних допомагає уникнути дублювання і забезпечує цілісність даних. В той же час, в системах з великим об'ємом запитів та даних часто використовують денормалізацію для оптимізації швидкості читання. Рішення про нормалізацію або денормалізацію має базуватися на конкретних потребах системи.

Ефективна індексація даних може значно покращити продуктивність системи, особливо коли йдеться про швидке видачу рекомендацій. Оптимізація запитів включає в себе аналіз планів запитів, вибір правильних типів індексів та оптимізацію схеми.

Забезпечення безпеки даних включає в себе захист від несанкціонованого доступу, шифрування даних, регулярне створення резервних копій та інші заходи.

Це особливо важливо для систем, які зберігають особисту інформацію користувачів.

Проєктування бази даних для системи рекомендацій є складним процесом, який вимагає глибокого розуміння як самої системи, так і потреб її користувачів. Правильно спроєктована база даних є ключовою до створення швидкої, надійної та масштабованої системи рекомендацій.

2.4. Висновки до розділу

У цьому розділі було здійснено деталізоване моделювання та структурування бізнес-процесів, які становлять основу програмного модуля обробки *mp*3 файлів для системи рекомендацій. Процес проєктування розглядав не лише логічну та фізичну архітектуру модуля, але й глибинну інтеграцію з іншими компонентами системи, що сприяє вирішенню завдань персоналізації контенту.

Для забезпечення ефективності рекомендацій було розглянуто ряд передових алгоритмів, які підтримують сучасні тенденції в машинному навчанні та штучному інтелекті. Особливу увагу було приділено таким методам, як *Learning to Rank* (*LTR*), що покращують точність ранжування елементів відповідно до користувацьких переваг; *Sequential Models,* які використовують послідовність взаємодій користувача для виявлення потенційно привабливих позицій; *Variational Autoencoders* (*VAE*), що здатні генерувати нові дані на основі вихідних, забезпечуючи розширення персоналізованого вибору; та *Reinforcement Learning* (*RL*), який адаптує рекомендації на основі неперервної взаємодії з користувачем.

Були детально розглянуті ключові моделі оцінки вхідних змінних, зокрема *Wide Component* для ефективного запам'ятовування простих взаємозв'язків та *Deep Component* для узагальнення складних неявних залежностей. Це дозволило створити багатошарові моделі, здатні враховувати різноманітність даних та забезпечувати більш глибинні та надійні рекомендації.

Презентовано приклади реалізації передових рекомендаційних алгоритмів. *Self-Attentive Sequential Recommendation* (*SASRec*) використовує механізми уваги для зосередження на найважливіших частинах користувацької історії; *Fusing Item Similarity Models with Self-Attention* (*FISSA*) інтегрує моделі схожості елементів з механізмами уваги для виокремлення більш тонких взаємозв'язків; *Collaborative Memory Networks* (*CMN*) забезпечують гнучке врахування спільних інтересів користувачів; а *Collaborative Variational Autoencoder* (*CVAE*) застосовує глибинні генеративні моделі для виявлення потенційних інтересів користувача за межами його явно виражених переваг.

РОЗДІЛ 3
ОПИСАННЯ РОЗРОБЛЕНОГО МОДУЛЯ ЗБЕРІГАННЯ *MP*3 ФАЙЛІВ

3.1. Вибір програмних і технічних компонентів для розробки програмного модуля

Для розгортання модуля треба описати фізичну архітектуру системи та її компонентів, а також їх взаємодію під час роботи. Розгортання прогармного модуля обробки *mp*3 файлів в системі рекомендацій включає наступні компоненти (рис. 3.1):

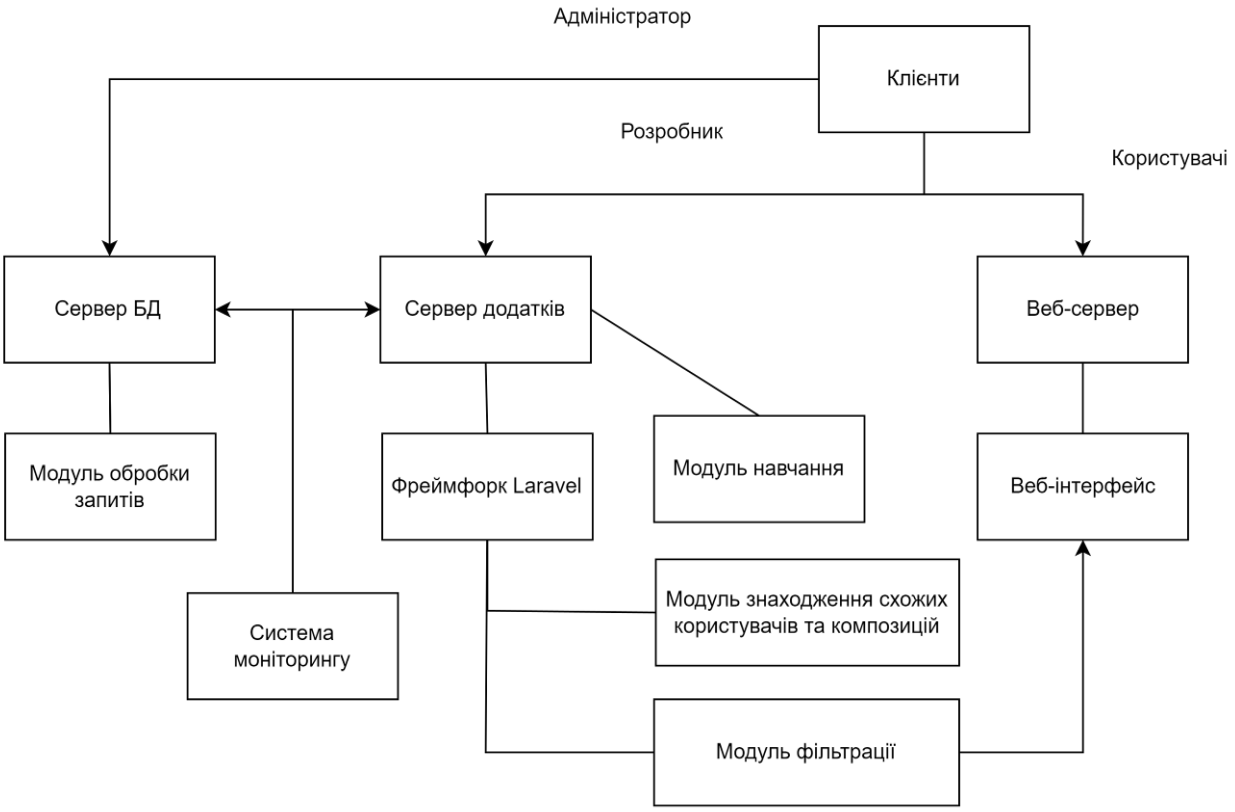

Рис. 3.1. Архітектура системи рекомендацій

Система використовує мікросервісну архітектуру для рекомендаційних сервісів і включає в себе такі компоненти:

– Сервер бази даних (Сервер БД): Зберігає всю інформацію, необхідну для роботи системи, включаючи метадані *mp*3 файлів та результати їх обробки.

– Сервер подачі: Відповідає за доставку *mp*3 файлів до відповідних модулів для обробки та виконання запитів користувачів.

– Фреймворк *Laravel*: Ймовірно використовується для розробки веб-додатків, що є частиною системи, і може слугувати як основа для сервера подачі та інших веб-орієнтованих компонентів.

– Модуль обробки заявів: Можливо, це компонент, який виконує первинну обробку *mp*3 файлів, таку як транскодування, нормалізацію аудіо чи видалення шумів.

– Система моніторингу: Слідкує за статусом інших модулів та системи в цілому, а також може відстежувати продуктивність та виявляти збої.

– Модуль навчання: Це може бути частина системи машинного навчання, яка аналізує *mp*3 файли та генерує рекомендації на основі навчальних даних.

– Модуль знаходження схожих конструкцій та композицій: Цей модуль може аналізувати музичні твори для визначення схожості між різними треками, що може допомогти у формуванні рекомендацій.

– Модуль фільтрації: Може використовуватися для відсіювання несуттєвих або небажаних *mp*3 файлів перед їх обробкою іншими модулями.

– Веб-сервер та Веб-інтерфейс: Ці компоненти забезпечують користувачам доступ до системи через інтернет, де вони можуть взаємодіяти з рекомендаційним сервісом.

Розгортання програмного модуля обробки *mp*3 файлів в системі рекомендацій починається з встановлення і конфігурації сервера бази даних, який буде зберігати не тільки самі *mp*3 файли, але і результати їх обробки, які будуть використовуватися для генерування рекомендацій. Сервер подачі має бути налаштований для ефективного розподілу завдань між різними модулями обробки та моніторингу. Фреймворк *Laravel* може бути використаний для створення веб-додатку, який дозволить користувачам інтерактивно користуватися сервісом.

Компоненти, які відповідають за машинне навчання, потребують налаштування моделей та алгоритмів, які будуть використовуватися для аналізу *mp*3 файлів та формування рекомендацій. Модулі, які відповідають за знаходження

схожостей і фільтрацію, повинні бути інтегровані таким чином, щоб вони могли співпрацювати з іншими частинами системи для забезпечення точності та релевантності рекомендацій.

Веб-сервер та веб-інтерфейс мають бути налаштовані для безпечного та зручного доступу користувачів до сервісу. В кінцевому підсумку, цілісна інтеграція всіх цих компонентів забезпечить створення надійної системи рекомендацій на базі мікросервісів.

Взаємодія між сервісами *ms-resource-service, ms-song-service, ms-resource-processor* та *storage-service* в системі збереження та обробки *MP*3 файлів може бути наступною:

– *ms-resource-service*: цей сервіс відповідає за управління ресурсами, такими як *MP*3 файли та інші аудіофайли, які завантажуються користувачами. Він надає *API* для завантаження, збереження та отримання аудіофайлів;

– *ms-song-service*: Сервіс *ms-song-service* відповідає за обробку метаданих аудіофайлів, таких як інформація про виконавця, назва пісні, жанр і т. д. Після завантаження аудіофайлу до *ms-resource-service, ms-song-service* отримує доступ до нього для аналізу та обробки метаданих;

– *ms-resource-processor*: Цей сервіс служить для обробки самого аудіофайлу (наприклад, конвертації в інший формат, стискання, покращення якості тощо). Він може бути інтегрований з *ms-song-service* для отримання додаткових даних про аудіофайл та подальшої обробки;

– *storage-service*: Сервіс зберігання (*storage-service*) відповідає за збереження аудіофайлів та доступ до них. Він може використовувати різні типи сховищ, такі як файлове сховище, обліковий запис хмари тощо. Після завантаження аудіофайлів *ms-resource-service* може передати їх для збереження в *storage-service*.

Розглянемо конкретну взаємодію між цими сервісами:

– користувач завантажує *MP*3 файл через інтерфейс *ms-resource-service,* який передає цей файл для збереження в *storage-service*;

– після завершення збереження, *ms-resource-service* надсилає повідомлення до *ms-song-service* про наявність нового аудіофайлу для обробки метаданих;

– *ms-song-service* обробляє аудіофайл і отримує необхідну інформацію про назву пісні, виконавця та інші метадані;

– *ms-song-service* надсилає оновлену інформацію назад до *ms-resource-service,* де вона може бути пов'язана з відповідним аудіофайлом.

У випадку, якщо потрібна обробка аудіофайлу (наприклад, конвертація в інший формат), *ms-resource-processor* бере файл з *storage-service,* обробляє його і повертає назад в сховище.

Така взаємодія дозволяє системі ефективно керувати завантаженням, обробкою та збереженням *MP*3 файлів, а також забезпечує коректну роботу з метаданими та обробку самого аудіофайлу, якщо це необхідно.

Завантаження *MP*3 файлу користувачем відбувається за наступними правилами:

– користувач взаємодіє з інтерфейсом *ms-resource-service* для завантаження *MP*3 файлу;

– *ms-resource-service* приймає файл від користувача та генерує унікальний ідентифікатор для цього файлу;

– Файл зберігається в системі зберігання (*storage-service*) під цим ідентифікатором.

Після збереження *MP*3 файлу в системі зберігання, *ms-resource-service* надсилає повідомлення до *ms-song-service* з ідентифікатором завантаженого файлу. Тоді *ms-song-service* приймає це повідомлення і починає аналіз *MP*3 файлу для отримання метаданих, таких як назва пісні, виконавець, жанр і т.д. Після аналізу, *ms-song-service* зберігає отримані метадані та повертає їх до *ms-resource-service*.

Сервіс *ms-resource-service* отримує метадані від *ms-song-service* та пов'язує їх з відповідним *MP*3 файлом за допомогою унікального ідентифікатора. Тепер інформація про файл та його метадані зберігаються в системі та доступні для подальшого використання.

Якщо необхідно провести обробку самого аудіофайлу (наприклад, конвертацію в інший формат або зменшення розміру), *ms-resource-processor* взаємодіє з *storage-service* для отримання аудіофайлу за його ідентифікатором *ms-resource-processor* обробляє аудіофайл та повертає його до *storage-service*.

Після обробки аудіофайлу, *ms-resource-processor* зберігає оброблений файл в системі зберігання (*storage-service*) під тим самим ідентифікатором.

Ця взаємодія дозволяє системі ефективно керувати завантаженням, збереженням, аналізом метаданих та, за необхідності, обробкою аудіофайлів *MP*3. При цьому інформація про файли та їх метадані зберігаються відокремлено, що дозволяє ефективно керувати та використовувати дані в системі.

Під час взаємодії сервісів *ms-resource-service, ms-song-service, ms-resource-processor* та *storage-service* в системі збереження та обробки *MP*3 файлів можуть виникнути різні проблеми та виклики. Розглянемо їх більш детально:

– споживання ресурсів: при обробці та збереженні великої кількості *MP*3 файлів може виникнути проблема надмірного споживання ресурсів серверів, таких як обсяг дискового простору та обчислювальна потужність. Важливо враховувати це при плануванні архітектури системи та масштабуванні;

– завантаження і мережева пропускна здатність: велика кількість користувачів, які завантажують або обробляють *MP*3 файли одночасно, може спричинити високе навантаження на мережу та сервери. Необхідно розглядати питання оптимізації мережевого обсягу та завантаження;

– помилки при завантаженні та обробці файлів: Під час завантаження або обробки *MP*3 файлів можуть виникати різні помилки, такі як недійсні файли, відсутність прав доступу, недостатньо ресурсів для обробки тощо. Система повинна бути стійкою до таких помилок і надавати відповідні повідомлення користувачам.

– керування мікросервісами: кількість мікросервісів у системі може зростати, що робить керування та моніторинг системи складнішим завданням.

Необхідно мати налагоджену систему моніторингу та логування для виявлення та вирішення проблем у сервісах.

– забезпечення безпеки: система повинна бути відповідно захищеною від атак та несанкціонованого доступу, особливо в умовах обробки користувацьких файлів. Це включає в себе валідацію та аутентифікацію користувачів, захист від -ін'єкцій та інших вразливостей.

– масштабованість: з плином часу кількість користувачів та *MP*3 файлів може зростати, що вимагатиме масштабування системи. Потрібно мати план масштабування, який дозволить збільшувати потужність серверів та ресурси відповідно до потреб системи.

– збереження та резервне копіювання даних: для забезпечення надійності та доступності даних важливо мати механізми резервного копіювання та відновлення. Необхідно розглянути питання збереження даних на різних носіях та в різних географічних регіонах.

– витрати на обслуговування та інфраструктуру: витрати на обслуговування та інфраструктуру системи, включаючи вартість хмарних послуг, обладнання серверів та оплату персоналу, який відповідає за підтримку системи.

Розуміння цих потенційних проблем та їх вирішення на етапі проектування та розробки системи допоможе забезпечити її стабільну роботу та надійність у процесі експлуатації.

Основні режими роботи модуля зберігання та обробки *MP*3 файлів можуть включати наступні:

1) Завантаження та збереження *MP*3 файлів, коли модуль може працювати у режимі завантаження *MP*3 файлів користувачами. В цьому режимі він приймає файли від користувачів, генерує унікальні ідентифікатори для них та зберігає файли на сервері або у системі збереження.

2) Аналіз метаданих *MP*3 файлів: модуль може аналізувати метадані *MP*3 файлів, такі як назва пісні, виконавець, жанр, тривалість тощо. В цьому режимі він виконує завдання з вилучення цих даних з аудіофайлів та їх подальшого збереження.

3) Обробка аудіофайлів: модуль може включати функціональність з обробки аудіофайлів, таку як конвертація в інший формат, зменшення розміру, стискіння, редагування аудіотреків тощо.

4) Інтеграція з іншими сервісами: модуль може працювати у режимі інтеграції з іншими сервісами, такими як месенджери, хмарні сховища або інші програмні системи. Це дозволяє обмінюватися *MP*3 файлами та метаданими з іншими додатками.

5) Система збереження та резервне копіювання: модуль може працювати у режимі системи збереження, де він забезпечує зберігання та доступ до *MP*3 файлів і метаданих. Також, він може реалізувати механізми резервного копіювання та відновлення даних.

6) Обслуговування запитів користувачів: Модуль може приймати запити від користувачів на отримання, редагування, видалення або обробку *MP*3 файлів та метаданих. В цьому режимі він надає *API* для взаємодії з клієнтськими додатками.

Модуль може працювати у режимі оптимізації обробки та збереження *MP*3 файлів з метою підвищення продуктивності та масштабованості системи.

3.2. Налаштування бази даних системи

Запити щодо колаборативної фільтрації можуть використовувати дані з таблиці *songables* та *userables,* які містять інформацію про зв'язок між користувачами та піснями, що їх вони вже слухали.

Основні запити до системи обробки *MP*3 файлів:

На рис. 3.2 показано інтерфейс налаштування бази даних системи:

1. Схема бази даних: Схема включає в себе таблиці, які мають відношення до музичної бібліотеки, такі як *authors, genres, songs,* і також таблиці, які пов'язані з користувацькими аспектами, такі як *users, userables,* та системні таблиці як *migrations, failed_jobs,* та *password_resets*. Це вказує на те, що база даних зберігає інформацію як про музичний контент, так і про користувачів і взаємодії з системою.

2. Загальні налаштування: Вказано, що для з'єднання з сервером бази даних використовується кодування *utf*8*mb*4*_unicode_ci*. Це кодування підтримує повний діапазон символів *Unicode,* що є важливим для міжнародної підтримки, включаючи емодзі. Вибір *utf*8*mb*4*_unicode_ci* також забезпечує, що рядки порівнюються без урахування регістру, що може бути корисним для запитів пошуку та фільтрації.

3. Налаштування вигляду: Користувацький інтерфейс налаштовано на українську мову, що свідчить про локалізацію системи для україномовних користувачів. Тема інтерфейсу встановлена як *pmahomme,* яка може бути однією з тем *phpMyAdmin,* популярного інструменту для управління *My* базами даних через веб-браузер.

На основі цього можна зробити висновок, що база даних правильно налаштована для зберігання різноманітного музичного контенту та для обслуговування користувачів з різних країн. Кодування *utf*8*mb*4 дозволяє зберігати велику кількість унікальних символів, що робить базу даних готовою до роботи з міжнародним контентом. Також, локалізація інтерфейсу забезпечує кращу доступність та зручність для українських користувачів.

Показано частину веб-інтерфейсу для налаштувань бази даних, який є частиною інструменту адміністрування баз даних. Детальний огляд налаштувань виглядає так:

1. Таблиці бази даних:

– *authors*: Містить інформацію про авторів музики, можливо включаючи імена, біографії та пов'язані твори.

– *genres*: Зберігає різні музичні жанри, які можуть бути використані для класифікації пісень.

– *songs*: Основна таблиця, де зберігаються дані про пісні, включаючи назви, тривалість, жанр, автора та інші метадані.

– *songables*: Може бути прикладом поліморфної асоціації, де пісні можуть мати багато відношень з різними іншими сутностями.

– *users*: Включає дані користувачів системи, такі як імена, електронні адреси, паролі тощо.

– *userables*: Ще одна таблиця поліморфних відносин, яка може вказувати на відношення користувачів до інших сутностей в базі даних.

– Системні таблиці, такі як *migrations*, *failed_jobs*, та *password_resets*, вказують на структуру, що використовується в *Laravel*, і служать для відстеження змін у структурі бази даних, завдань, які не вдалося виконати, та забезпечення можливості відновлення паролів.

2. Загальні налаштування:

– Кодування *utf8mb4_unicode_ci* для з'єднання з сервером бази даних гарантує, що текстові дані можуть включати символи у повному розмаїтті *Unicode*, що особливо важливо для систем, які містять музичні та культурні дані з усього світу.

– Опція "Дорадчі налаштування" може вказувати на можливість системи автоматично оптимізувати запити або налаштування сервера для поліпшення продуктивності.

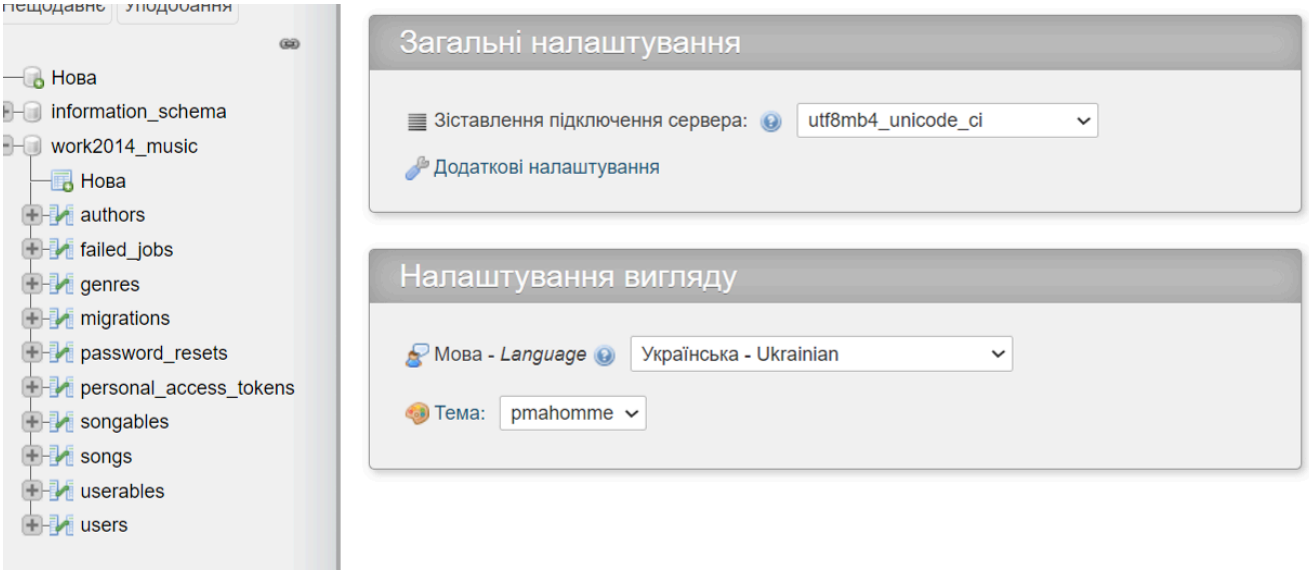

Рис. 3.2. Вікно управління БД

Основні запити до системи обробки *MP*3 файлів можуть бути різноманітними, в залежності від функціоналу системи. Нижче наведені декілька

прикладів запитів, які можуть бути використані в такій системі, разом із поясненням їх функцій:

Запит для отримання списку всіх пісень:

*SELECT * FROM songs*;

Цей запит вибирає всі записи з таблиці *songs*. Він може бути використаний для відображення усього каталогу музики в системі.

Запит для пошуку пісні за назвою:

*SELECT * FROM songs WHERE title LIKE* '%*love*%';

Цей запит шукає пісні, назва яких містить слово "*love*". Відсоткові знаки (%) використовуються як символи підстановки для здійснення пошуку з частковою відповідністю.

Запит для вибірки пісень певного жанру:

*SELECT * FROM songs WHERE genre_id* = (*SELECT id FROM genres WHERE name* = '*Rock*');

Цей запит спочатку знаходить ідентифікатор жанру "*Rock*" у таблиці *genres,* а потім вибирає всі пісні цього жанру з таблиці *songs*.

Запит для додавання нової пісні:

INSERT INTO songs (*title, author_id, genre_id, length*) *VALUES* ('*New Song*', 1, 3, '3:30');

Цей запит додає нову пісню з назвою "*New Song*", автором з ідентифікатором 1, жанром з ідентифікатором 3, і тривалістю 3 хвилини 30 секунд.

Запит для оновлення інформації про пісню:

UPDATE songs SET title = '*Updated Title*' *WHERE id* = 5;

Цей запит оновлює назву пісні з ідентифікатором 5 на "*Updated Title*".

Запит для видалення пісні:

DELETE FROM songs WHERE id = 5;

Цей запит видаляє пісню з ідентифікатором 5 з бази даних.

Запит для знаходження всіх пісень певного автора:

SELECT s. FROM songs s JOIN authors a ON s.author_id = a.id WHERE a.name* = '*Artist Name*';

Цей запит з'єднує таблиці *songs* і *authors* для вибору пісень, написаних автором з ім'ям "*Artist Name*".

3.3. Основні режими роботи програмного модуля

Схема алгоритму попередньої обробки налаштувань на рис. 3.5.

На схемі алгоритму попередньої обробки налаштувань, яку ви надіслали, представлено послідовність кроків, які виконуються у процесі введення та обробки замовлення користувачем. Ось опис цих кроків:

1. Початок: Стартова точка алгоритму.

2. Введення замову: Користувач починає процес, вносячи інформацію для замовлення.

3. Обробка форми: Система перевіряє дані, введені користувачем, на предмет помилок або неповноти.

4. *new_check*?: Перевірка, чи замовлення є новим.

– Якщо так, переходимо до *add_new*.

– Якщо ні, переходимо до наступної перевірки.

5. *add_new*: Додаємо нове замовлення до системи, що включає в себе виконавців, які виступають у обрані дати.

6. *genre_list*?: Перевірка, чи користувач обрав жанр.

– Якщо так, переходимо до *add_genre*.

– Якщо ні, переходимо до наступної перевірки.

7. *add_genre* (*genre_list*): Додаємо замовлення виконавців, що відповідають обраному жанру зі списку *genre_list*.

8. *year_list*?: Перевірка, чи користувач обрав епоху/рік.

– Якщо так, переходимо до *add_year*.

– Якщо ні, переходимо до виконання запиту.

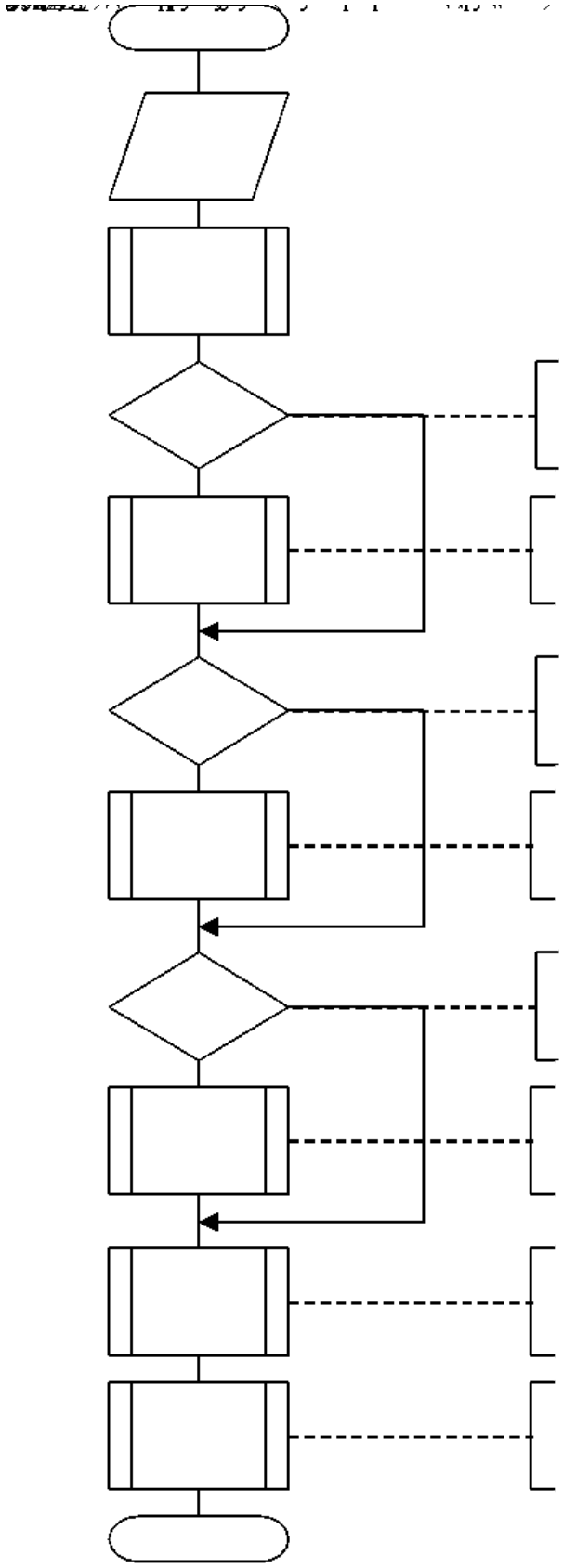

Рис. 3.5. Схема алгоритму попередньої обробки налаштувань для роботи фільтру

9. *add_auth* (*year_list*): Додаємо до замовлення виконавців, які відповідають обраному року зі списку *year_list*.

10. *query_go*: Виконуємо запит на основі введених критеріїв.

11. *refresh*(): Оновлюємо сторінку з результатами запиту.

12. Кінець: Завершення алгоритму.

Загалом, ця схема представляє логіку обробки замовлень у системі, яка дозволяє користувачам вибирати нові замовлення чи додавати до існуючих замовлень виконавців на основі їх жанру та епохи. Система виконує запити та оновлює інформацію на екрані користувача відповідно до вибору, зробленого користувачем.

Схема алгоритму обробки списку видачі на рис. 3.6.

На зображенні представлено схему алгоритму обробки списку видачі, який використовується для обробки замовлень користувачів у системі. Ось кроки цього алгоритму:

1. Початок: Вхідна точка алгоритму, початок процесу.

2. *id_user*: Ідентифікація користувача, можливо за допомогою унікального ідентифікатора користувача.

3. Отримання даних за *id_user*: Витягнення даних користувача з бази даних або іншого джерела даних.

4. *no_estm*?: Перевірка на наявність неоцінених треків.

– Якщо немає неоцінених треків, переходимо до наступної перевірки.

– Якщо є неоцінені треки, виконуємо процедуру виведення форм оцінювання випадкових треків без оцінки.

5. *no_g_estm*?: Перевірка на наявність неоцінених жанрів.

– Якщо немає неоцінених жанрів, переходимо до вибірки таблиці.

– Якщо є неоцінені жанри, виконуємо процедуру виведення форм оцінювання випадкових жанрів без оцінки.

6. *corel_f_u* (*id_user*): Вибірка таблиці кореляційних оцінок обраного користувача за кожен оцінений трек з оцінками інших користувачів.

7. *filtr_c*(0.7): Фільтр за рівнем кореляції >0.7.

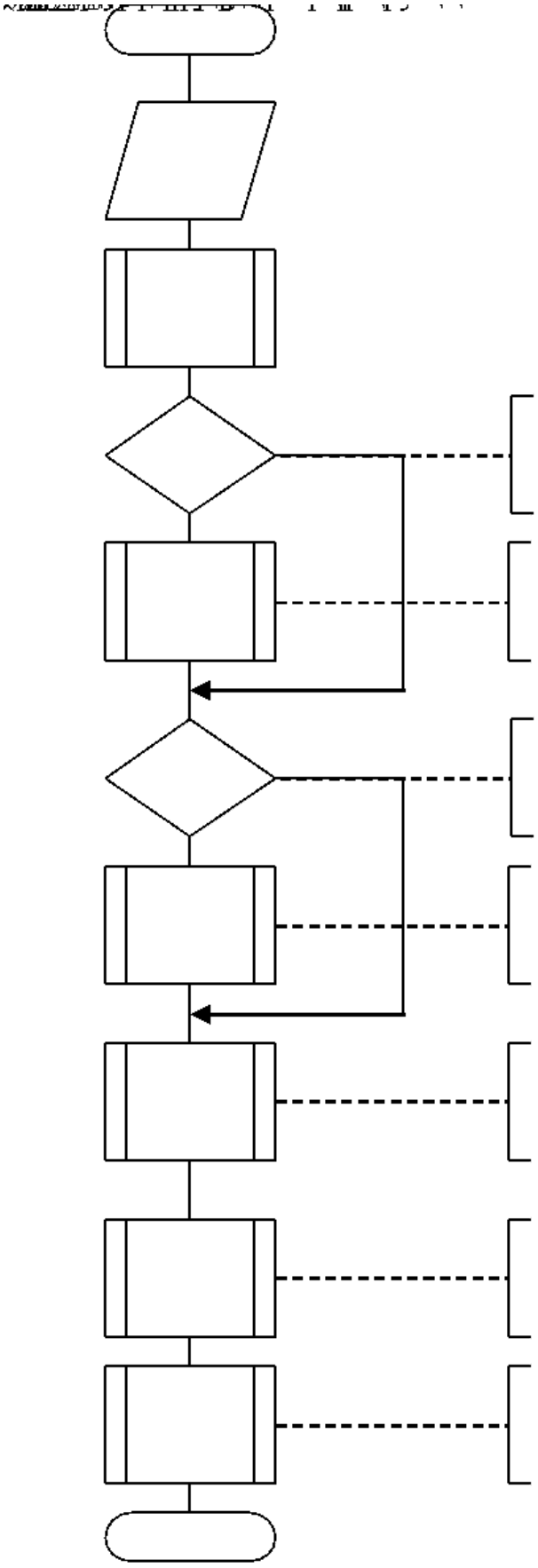

Рис. 3.6. Схема алгоритму обробки списку видачі

8. *refresh*(): Оновлення сторінки з композиціями з відповідними функцією *filtr_c*(0.7).

9. Кінець: Завершальний етап алгоритму.

Судячи з схеми, алгоритм виконує відбір і оцінювання музичних треків та жанрів відповідно до вподобань користувача. Після оцінювання треків і жанрів система використовує кореляцію між оцінками користувача і оцінками інших користувачів для фільтрації і видачі рекомендацій. Фільтр за рівнем кореляції дозволяє відсіювати тільки ті треки, які мають високу схожість з вподобаннями користувача. Завершується процес оновленням списку видачі для користувача.

Для прослуховування та перегляду кліпів з урахуванням фільтрації і обробки *mp*3 файлів необхідно зареєструватись (рис. 3.7).

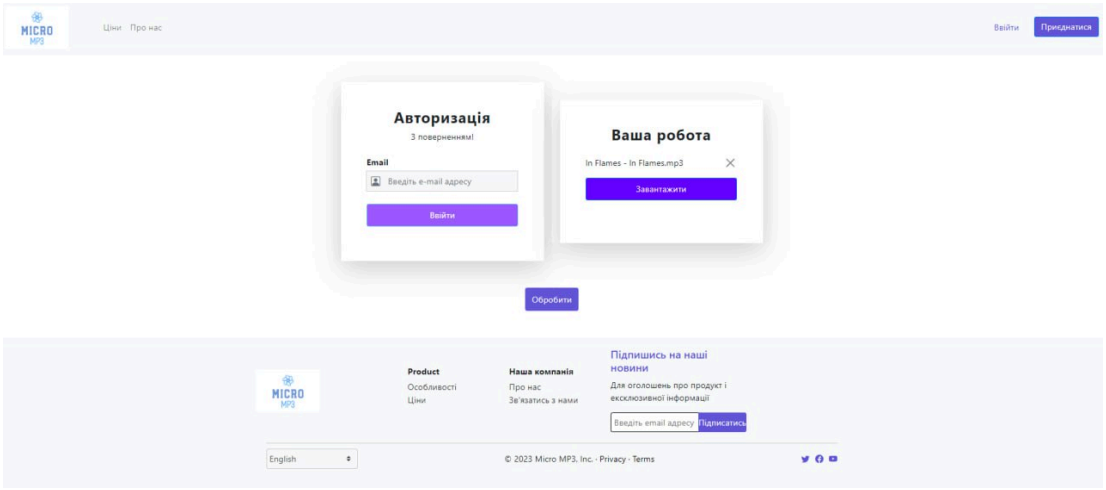

Рис. 3.7. Вікно авторизації користувача

На зображенні відображено веб-інтерфейс, який, схоже, пропонує користувачам можливість авторизації для доступу до певних функцій. Ось детальний опис того, як і чому реєстрація і вхід в систему є необхідними для прослуховування та перегляду кліпів:

1. Персоналізація досвіду користувача:

– Реєстрація дозволяє системі зберігати персональні вподобання та історію переглядів користувача, що сприяє кращій персоналізації рекомендацій та забезпечує більш відповідний досвід прослуховування.

2. Збереження налаштувань фільтрації:

– Під час прослуховування музики або перегляду кліпів користувачі можуть встановлювати фільтри, наприклад, за жанром, епохою, мовою чи іншими критеріями. Зареєстровані користувачі можуть зберігати ці налаштування для майбутнього використання.

3. Обробка та якість відтворення:

– Системи часто пропонують різні рівні якості відтворення, від базової до високої дефініції, залежно від статусу користувача. Зареєстровані користувачі можуть мати можливість вибирати вищу якість відтворення або отримати доступ до додаткових опцій обробки звуку.

4. Синхронізація між пристроями:

– Коли користувачі зареєстровані, вони можуть синхронізувати свої списки відтворення, налаштування фільтрації та вподобання між різними пристроями, що дозволяє легко переходити між мобільним, планшетом та настільним комп'ютером.

5. Захист контенту:

– Авторизація також служить засобом захисту авторських прав, обмежуючи доступ до контенту лише для зареєстрованих користувачів, що може бути вимогою ліцензійних угод.

6. Збір аналітики:

– Зареєстровані користувачі дозволяють платформі краще зрозуміти попит на певні жанри чи артистів, що може впливати на майбутні рекомендації та стратегії контенту.

У контексті зображення, користувач має можливість ввести свою електронну пошту для отримання доступу до функціоналу сайту. Це стандартна практика для веб-сайтів, які пропонують персоналізований контент і хочуть забезпечити кращий користувацький досвід, зберігаючи при цьому захист контенту та дотримуючись норм авторського права.

Завжди можна переглянути список оброблених файлів (рис. 3.8).

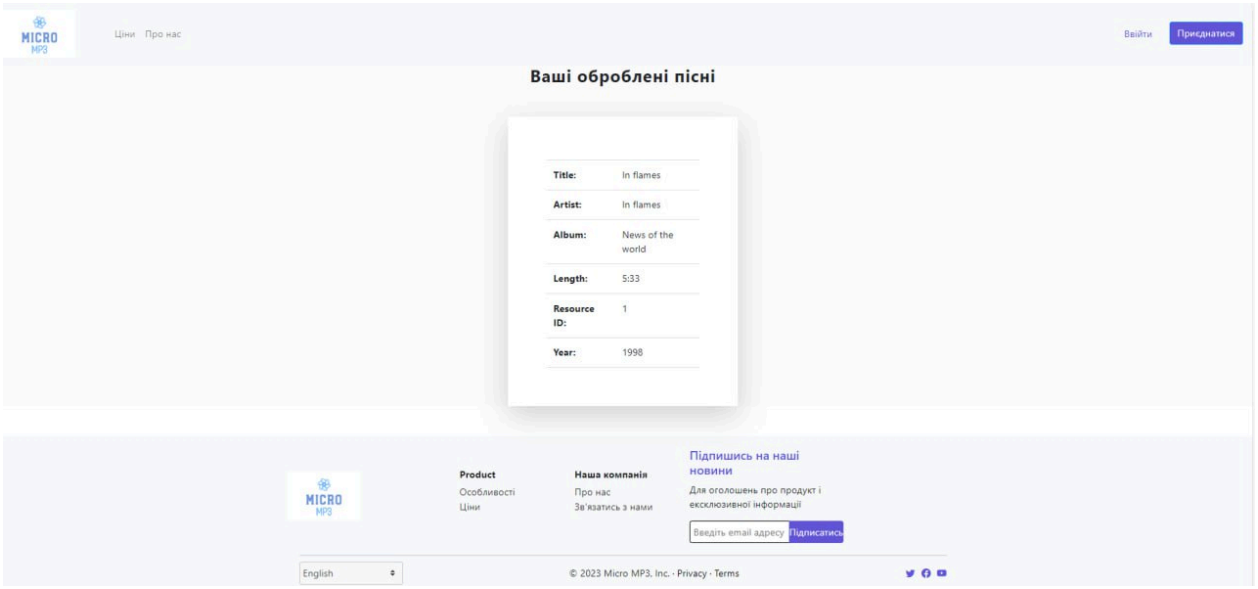

Рис. 3.8. Вікно обробки файлів

На зображенні відображено веб-інтерфейс, де користувачі можуть переглядати список оброблених файлів. З контексту можна зробити припущення про функціональність цього інтерфейсу:

1. Список оброблених файлів:

– Це вікно зазвичай містить інформацію про кожен з оброблених *MP*3 файлів, включаючи такі деталі, як назва треку, артист, альбом, тривалість, жанр та рік випуску.

2. Функціональність:

– Користувачі можуть використовувати цей список для організації своїх музичних колекцій, вибору пісень для прослуховування, або для здійснення додаткових дій, таких як завантаження, поділитися з іншими чи видалення файлів.

3. Переваги:

– Зручний доступ до обраних файлів підвищує продуктивність користувача та покращує загальний досвід використання платформи.

– Відображення повних метаданих кожного треку дозволяє користувачам легко ідентифікувати та вибирати потрібні треки.

4. Інтеграція:

– Цей список може інтегруватися з іншими функціями платформи, такими як персоналізовані рекомендації, плейлисти та історія прослуховування.

На зображенні також присутній логотип компанії або продукту, футер з вибором мови та посилання на правила та умови, що є типовими елементами веб-сайтів і вказують на професійний рівень розробки.

Робота додаткового модуля фільтрації передбачає ранжування музичних кліпів у відповідності до вподобань користувача та вподобань користувачів, які зробили подібний вибір (рис. 3.9).

Робота додаткового модуля фільтрації в системі обробки музичних кліпів може бути досить складною та включати в себе декілька ключових елементів, щоб ефективно ранжувати кліпи відповідно до вподобань користувача. Ось детальний опис процесу:

1. Збір даних про вподобання користувачів:

– Система використовує алгоритми машинного навчання та аналітику даних для збору та аналізу інформації про переваги користувачів, які включають історію переглядів, оцінки, частоту прослуховування та вибір жанрів.

2. Профілювання користувачів:

– Кожен користувач має унікальний профіль, який враховує його вподобання. Це дозволяє системі визначати, які кліпи ймовірніше сподобаються конкретному користувачу.

3. Ранжування за схожістю вподобань:

– Модуль фільтрації аналізує схожість вподобань між користувачами, використовуючи методи, такі як колаборативна фільтрація. Якщо один користувач подібно оцінює певний набір кліпів, як і інший, система може рекомендувати цьому користувачу інші кліпи, які сподобались другому користувачу.

4. Персоналізоване ранжування:

– На основі зібраних даних та профілювання користувачів модуль фільтрації формує персоналізований рейтинг кліпів. Кліпи, які мають вищу ймовірність сподобатися користувачу, будуть відображатися вище в списку рекомендацій.

5. Адаптивність та навчання:

– Модуль фільтрації постійно оновлюється та адаптується, враховуючи нові дані, що надходять, такі як зміни в поведінці користувачів чи нові оцінки, що дозволяє підтримувати актуальність рекомендацій.

6. Оцінка та зворотний зв'язок:

– Користувачі можуть оцінювати рекомендації, надаючи зворотний зв'язок, який додатково використовується для вдосконалення алгоритмів ранжування.

Цей процес передбачає постійне самовдосконалення системи, щоб забезпечити максимально релевантний та персоналізований досвід користувачів. Важливим аспектом є захист приватності користувачів та забезпечення, що всі зібрані дані використовуються етично та відповідно до встановлених правил та нормативів.

Також в системі реалізовано модуль програвання кліпів з сторонніх джерел (рис. 3.10).

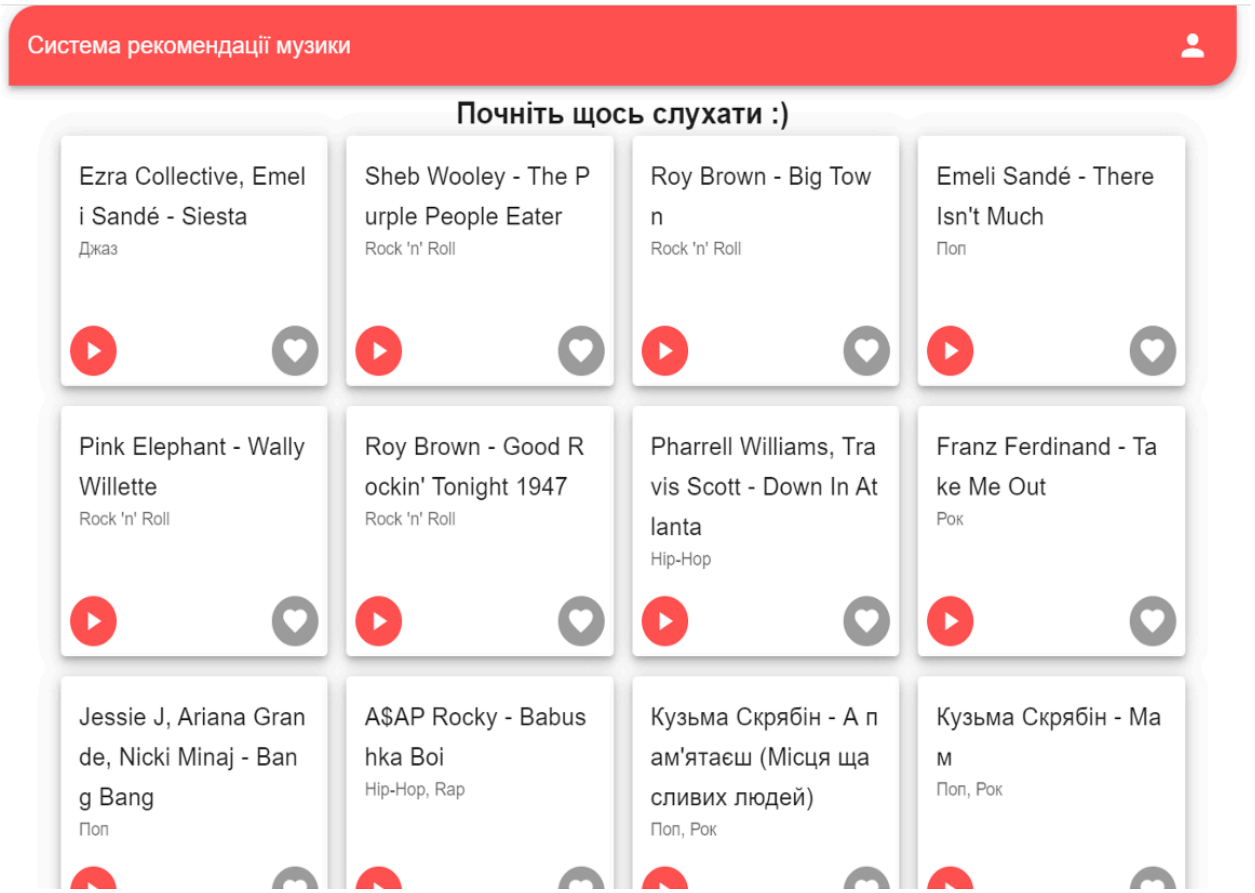

Рис. 3.9. Рекомендовані треки

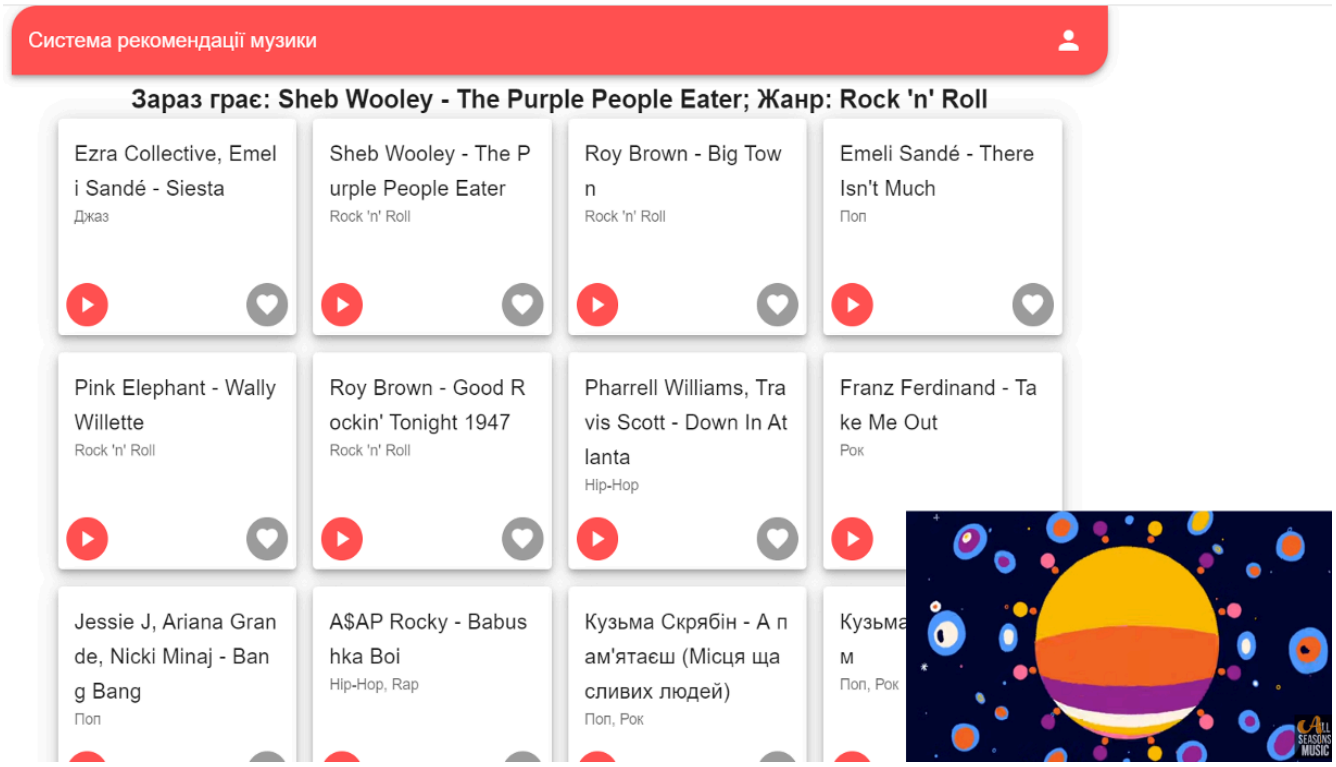

Рис. 3.10. Програвання треку

Модуль програвання кліпів зі сторонніх джерел у системі обробки музичного контенту дозволяє користувачам відтворювати відео або музичні кліпи, які зберігаються на інших платформах чи веб-сайтах. Ось детальний опис, як цей модуль може працювати:

1. Інтеграція з *API* сторонніх сервісів:

– Система може використовувати програмні інтерфейси (*API*) сторонніх сервісів, таких як *YouTube, Vimeo, SoundCloud* тощо, для отримання доступу до музичних кліпів.

– Через *API* система може запитувати метадані кліпів, отримувати *URL*-адреси потокового передавання та інтегрувати їх у свій інтерфейс для програвання.

2. Вбудований медіаплеєр:

– Модуль включає вбудований медіаплеєр, який підтримує різні формати відео та аудіо.

– Медіаплеєр може мати стандартні функції, такі як відтворення, пауза, прокрутка, регулювання гучності та зміна якості відео.

3. Кешування та буферизація:

– Для забезпечення плавного відтворення без перервань, система може кешувати дані відео та використовувати буферизацію для мінімізації затримок при низькій швидкості інтернету.

4. Адаптивний поток:

– Модуль може підтримувати адаптивне потокове передавання, що автоматично адаптує якість відео в залежності від швидкості інтернет-з'єднання користувача.

5. Ліцензування та авторські права:

– Система повинна дотримуватися правил ліцензування та авторських прав при використанні контенту зі сторонніх джерел, включаючи відображення необхідних авторських повідомлень та лінків на оригінальний контент.

6. Безпека та приватність:

– Всі запити до сторонніх *API* мають бути забезпечені, використовуючи шифрування, а доступ до сторонніх ресурсів має відповідати стандартам безпеки та приватності.

7. Юзабіліті та інтерфейс користувача:

– Інтерфейс користувача для модуля програвання повинен бути інтуїтивно зрозумілим та легким у використанні, забезпечуючи користувачам зручний доступ до всіх функцій програвання.

Цей модуль значно розширює можливості системи, надаючи користувачам доступ до широкого спектру музичного контенту, який може не бути безпосередньо збережений в локальній базі даних системи, і при цьому забезпечуючи високий рівень сервісу та зручності.

3.4. Тестування роботи модуля

При розробці та тестуванні модуля програвання кліпів на *Java* можуть бути застосовані різні стратегії тестування, які забезпечують його надійність та ефективність. Ось детальний опис потенційних тестів, які можуть бути виконані:

1. Юніт-тести (*Unit Testing*):

– Проводиться тестування окремих компонентів модуля, таких як методи для запитування даних з *API,* функції обробки відеопотоку та алгоритми буферизації.

– Використання фреймворків для юніт-тестування, таких як *JUnit,* дозволяє автоматизувати цей процес і забезпечити, що окремі функції працюють як очікується.

2. Інтеграційні тести (*Integration Testing*):

– Перевірка взаємодії між компонентами модуля, наприклад, як *API*-клієнт взаємодіє з медіаплеєром.

– Можливе використання інструментів, таких як *Mockito,* для мокування зовнішніх сервісів і переконання в тому, що модуль коректно обробляє реальні сценарії взаємодії.

3. Функціональні тести (*Functional Testing*):

– Оцінка роботи всього модуля з точки зору кінцевого користувача, щоб переконатися, що всі користувацькі сценарії виконуються правильно.

– Можуть бути використані інструменти автоматизації тестування інтерфейсу, такі як *Selenium,* для імітації користувацьких дій.

4. Навантажувальні тести (*Load Testing*):

– Симуляція високого навантаження на модуль для перевірки його продуктивності та масштабованості.

– Використання інструментів як *Apache JMeter* дозволяє виміряти час відгуку системи та її здатність обробляти багатокористувацькі запити.

5. Стрес-тести (*Stress Testing*):

– Випробування модуля за умовами, які перевищують нормальні операційні вимоги, щоб визначити його точку відмови.

– Це допомагає виявити потенційні проблеми в обробці помилок та відмовостійкості.

6. Тести безпеки (*Security Testing*):

– Аналіз модуля на предмет вразливостей, таких як *SQL* ін'єкції, *Cross-Site Scripting* (*XSS*), або небезпечні помилки в обробці *API*.

– Інструменти, такі як *OWASP ZAP* або *Fortify,* можуть бути використані для автоматизованого виявлення вразливостей.

7. Тести сумісності (*Compatibility Testing*):

– Переконання, що модуль працює на всіх підтримуваних платформах та браузерах.

– Може включати тестування на різних операційних системах, версіях браузерів та типах пристроїв.

8. Тести користувацького інтерфейсу (*User Interface Testing*):

– Перевірка елементів інтерфейсу на предмет відповідності макетам дизайну, правильності відображення та інтуїтивності використання.

Було розроблено низку автотестів, що наведено в Додатку А.

3.5. Висновки до розділу

У цьому розділі представлено детальний опис та аналіз розробленого модуля фільтрації для системи музичних рекомендацій. Описуються вибрані програмні інструменти та технічні рішення, що були задіяні при створенні модуля, розглядається структура бази даних, яка лежить в основі системи, та детально описуються алгоритми колаборативної фільтрації, що використовуються для генерування рекомендацій. Також надається огляд ключових режимів роботи розробленого модуля та обговорюються результати його тестування. Завдяки проведеному тестуванню підтверджено функціональність модуля і його здатність надавати рекомендації високої якості, що вказує на успішну реалізацію методів колаборативної фільтрації у системі.

ВИСНОВКИ

У кваліфікаційній роботі була проведена розробка та дослідження програмного модуля для зберігання та обробки *MP*3 файлів на основі мікросервісної архітектури з використанням системи *Telegram* для обміну файлами. Робота включала в себе аналіз проблематики, огляд існуючих рішень, визначення основних задач, обґрунтування архітектурних рішень та наукове дослідження в обраній області.

Проблема збереження та обробки аудіофайлів, зокрема *MP*3, є актуальною в сучасному інформаційному суспільстві. Забезпечення безпеки та доступності цих файлів відіграє важливу роль в багатьох сферах, від розваг до бізнесу.

Робота містить наукову складову, оскільки вона включає в себе наукове дослідження щодо оптимізації обробки *MP*3 файлів та аналізу сучасних технологій у цій галузі.

Метою дослідження була розробка програмного модуля, який забезпечує збереження та обробку *MP*3 файлів на основі мікросервісної архітектури та інтеграцію з *Telegram*. Результатом є функціонально працездатний модуль та проведене наукове дослідження.

Основні досягнення:

– розроблено програмний модуль, який дозволяє зберігати та обробляти *MP*3 файли на віддалених серверах з використанням мікросервісної архітектури.

– виконано інтеграцію з *Telegram* для забезпечення обміну *MP*3 файлами між користувачами.

– проведено наукове дослідження щодо оптимізації обробки *MP*3 файлів та оцінено його результати.

Перспективи подальших досліджень та розробок: на основі результатів цієї роботи відкриваються нові можливості для подальших досліджень і розробок в галузі збереження та обробки аудіофайлів. Можливі напрямки подальших досліджень включають в себе покращення алгоритмів обробки, розширення

функціональності програмного модуля, підвищення безпеки та збільшення ефективності роботи.

Загальною важливістю цієї кваліфікаційної роботи є те, що вона сприяє розвитку сучасних технологій у галузі обробки та збереження аудіофайлів, розширюючи можливості їхнього використання в різних галузях, включаючи розваги, медицину, освіту та інші.

Слід підкреслити важливість отриманих результатів та їхній потенціал у різних сферах. Основні висновки та результати цієї роботи варто взяти до уваги:

1. Збереження та обробка *MP*3 файлів: Розроблений програмний модуль відкриває можливості для збереження та обробки *MP*3 файлів на віддалених серверах. Це особливо корисно в контексті медіа-проектів, де потрібно забезпечити надійне та ефективне збереження музичних аудіотреків.

2. Мікросервісна архітектура: використання мікросервісної архітектури дозволяє підвищити масштабованість та флексібільність розробленого модуля. Цей підхід дає можливість швидко реагувати на зміни та вимоги користувачів.

3. Інтеграція з *Telegram*: інтеграція з популярною месенджерною системою *Telegram* розширює можливості використання розробленого модуля. Користувачі можуть легко обмінюватися аудіофайлами, що додає зручності у комунікації та спільній роботі.

4. Наукове дослідження: робота включає в себе наукову складову, оскільки проводить аналіз та дослідження важливих аспектів обробки *MP*3 файлів та оптимізації процесів. Це відкриває можливості для подальших досліджень у цій області.

5. Потенціал для розвитку: результати цієї роботи можуть бути використані як основа для подальших розробок та вдосконалення програмного модуля. Покращення алгоритмів обробки, розширення функціональності та підвищення безпеки є потенційними напрямками розвитку.

Робота відкриває нові горизонти в області збереження та обробки аудіофайлів, а також демонструє значущість використання мікросервісної архітектури та інтеграції з сучасними месенджерами.

СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Бойченко С.В., Іванченко О.В. Положення про дипломні роботи (проєкти) випускників Національного авіаційного університету. – К.: НАУ, 2017. – 63 с.

2. ДСТУ 3008-95. Документація. Звіти у сфері науки і техніки. Структура і правила оформлення. – 39 с.

3. *R. Wazirali, R. Ahmad, A. Al-Amayreh, M. Al-Madi, and A. Khalifeh, Secure Watermarking Schemes and Their Approaches in the IoT Technology*: *An Overview, Electronics,* 14 (10), 2021, *pp*. 144-152.

4. *Q. Liu, A. Sung, M. Qiao, Temporal derivative-based spectrum and Mel–cepstrum audio steganalysis, IEEE Transactions on Information Forensics and Security,* 4 (3), 2009, *pp*. 359–368.

5. *A. Zgank, IoT-Based Bee Swarm Activity Acoustic Classification Using Deep Neural Networks, Sensors* (*Basel*), *vol*. 21, *no*. 3, 2021.

6. *Q. Liu, A. Sung, M. Qiao, Z. Chen, B. Ribeiro, An improved approach to steganalysis of JPEG images, Information Sciences,* 180 (9), 2010, *pp*. 1643–1655.

7. *R. R. Ginanjar, S. Bhardwaj, D.-S. Kim, and J.-M. Lee,* "*Rounding Modulation for Transparent Data-Hiding Scheme in High-Quality Audio File,*" *in* 2019 *IEEE International Conference on Consumer Electronics* (*ICCE*), 2019, *pp*. 1–2.

8. *I. Avcibas, Audio steganalysis with content-independent distortion measures, IEEE Signal Processing Letters,* 13 (2), 2006, *pp*. 92–95.

9. *R. Böhme, A. Westfeld, Statistical characterisation of MP*3 *encoders for steganalysis, in*: *Proceedings of the Workshop on Multimedia and Security,* 2004, *pp*. 25–34.

10. *J. Fridrich, J. Kodovsky´ , V. Holub, M. Goljan, Steganalysis of content-adaptive steganography in spatial domain, Information Hiding, Lecture Notes in Computer Science,* 2011, *pp*. 58-64.

11. *S. Geetha, N. Ishwarya, N. Kamaraj, Evolving decision tree rule based system for audio stego anomalies detection based on Hausdorff distance statistics, Information Sciences,* 180 (13), 2010, *pp*. 2540–2559.

12. *I. Guyon, J. Weston, S. Barnhill, V. Vapnik, Gene selection for cancer classification using support vector machines, Machine Learning* 46 (1–3) 2002, *pp*. 389–422.

13. *J. Harmsen, W. Pearlman, Steganalysis of additive noise modelable information hiding, Proceedings of SPIE Electronic Imaging, Security, Steganography, and Watermarking of Multimedia Contents,* 5020, 2003, *pp*. 131–142.

14. *I. Inza, B. Sierra, R. Blanco, P. Larranaga, Gene selection by sequential search wrapper approaches in microarray cancer class prediction, Journal of Intelligent and Fuzzy Systems,* 12 (1), 2002, *pp*. 25–33.

15. *M. Johnson, S. Lyu, H. Farid, Steganalysis of recorded speech, Proceedings of SPIE,* 5681, 2005, *pp*. 664–672.

16. *M. Kharrazi, T.H. Sencar, N. Memon, Improving steganalysis by fusion techniques*: *a case study with image steganography, LNCS Transactions on Data Hiding and Multimedia Security,* 4300, 2006, *pp*. 123–137.

17. *C. Kraetzer, J. Dittmann, Pros and cons of Mel–cepstrum based audio steganalysis using SVM classification, Lecture Notes in Computer Science* 4567, 2008, *pp*. 359–377.

18. *Q. Liu, A. Sung, Z. Chen, J. Xu, Feature mining and pattern classification for steganalysis of LSB matching steganography in grayscale images, Pattern Recognition,* 41 (1), 2008, *pp*. 56–66.

19. *R. R. Ginanjar, D. S. Kim, and C. B. Moon,* "*Enhanced digital audio watermarking using genetic algorithm,*" *in* 2018 *IEEE International Conference on Consumer Electronics – Asia* (*ICCE-Asia*), 2018, *pp*. 206– 212.

20. *H. Karajeh, T. Khatib, L. Rajab, and M. Maqableh,* "*A Robust Digital Audio Watermarking Scheme Based on DWT and Schur Decomposition,*" *Multimedia Tools Appl., vol*. 78, *no*. 13, 2019, *p*. 18395–18418. *https*://*doi.org*/10.1007/*s*11042-019-7214-3

21. *L. Novamizanti, G. Budiman, and B. A. Wibowo,* "*Optimasi Sistem Penyembunyian Data pada Audio menggunakan Sub-band Stasioner dan Manipulasi Rata-rata Statistik,*" *ELKOMIKA*: *Jurnal Teknik Energi Elektrik, Teknik Telekomunikasi, & Teknik Elektronika,* 2018*, pp*. 302-308.

22. *S. C. Kushwaha, P. Das, and M. Chakraborty,* "*Multiple Watermarking on Digital Audio Based on DWT Technique,*" *in* 2015 *International Conference on Communications and Signal Processing* (*ICCSP*)*,* 2015*, pp*. 303–307.

23. *Q. Liu, A. Sung, B. Ribeiro, M. Wei, Z. Chen, J. Xu, Image complexity and feature mining for steganalysis of least significant bit matching steganography, Information Sciences,* 178 (1)*,* 2008*, pp*. 21–36.

24. *Q. Liu, A. Sung, M. Qiao, Neighboring joint density based JPEG steganalysis, ACM Transactions on Intelligent Systems and Technology,* 2 (2)*,* 2011*, http*://*dx.doi.org*/10.1145/1899412.1899420.

25. *Y. Lin and W. Abdulla, Audio Watermark*: *A Comprehensive Foundation Using MATLAB*. *Springer,* 01 2015.

26. Бойко Б.М. Програмний модуль зберігання *mp*3 файлів на основі мікросервісної архітектури // Тези доповідей наук.-практ. конф. "Сучасні тенденції розвитку системного програмування" (23-24 листопада 2023 р.). – К.: НАУ, 2023. – С. 43-44.

Програмний код тестування модуля обробки *MP*3 файлів

```
// 1. Тестування відтворення музичного файлу

@Test

public void testPlayAudio() {

    AudioPlayer player = new AudioPlayer();

    player.loadAudio("AudioId123");

    player.play();

    assertTrue(player.isPlaying());

}


// 2. Тестування паузи музичного файлу

@Test

public void testPauseAudio() {

    AudioPlayer player = new AudioPlayer();

    player.loadAudio("AudioId123");

    player.play();

    player.pause();

    assertFalse(player.isPlaying());
```

```
}

// 3. Тестування зупинки музичного файлу
@Test
public void testStopAudio() {
    AudioPlayer player = new AudioPlayer();
    player.loadAudio("AudioId123");
    player.play();
    player.stop();
    assertEquals(player.getCurrentTime(), 0);
}

// 4. Тестування перемотування музичного файлу
@Test
public void testRewindAudio() {
    AudioPlayer player = new AudioPlayer();
    player.loadAudio("AudioId123");
    player.play();
    // Припустимо, що музичного файлу вже програло 10 секунд
```

```
    player.rewind(5);
    assertEquals(player.getCurrentTime(), 5);
}

// 5. Тестування вибору якості музичного файлу
@Test
public void testChangeAudioQuality() {
    AudioPlayer player = new AudioPlayer();
    player.loadAudio("AudioId123");
    player.setQuality(AudioQuality.HD);
    assertEquals(player.getCurrentQuality(), AudioQuality.HD);
}

// 6. Тестування відображення субтитрів
@Test
public void testSubtitlesDisplay() {
    AudioPlayer player = new AudioPlayer();
    player.loadAudio("AudioId123");
    player.enableSubtitles(true);
```

```
    assertTrue(player.isSubtitlesEnabled());
}

// 7. Тестування повідомлення про помилку при завантаженні музичного файлу
@Test(expected = AudioLoadException.class)
public void testAudioLoadError() {
    AudioPlayer player = new AudioPlayer();
    player.loadAudio("invalidAudioId");
}

// 8. Тестування тривалості музичного файлу
@Test
public void testAudioDuration() {
    AudioPlayer player = new AudioPlayer();
    player.loadAudio("AudioId123");
    assertEquals(player.getDuration(), 120); // Припустимо, що тривалість - 120 секунд
}

// 9. Тестування гучності звуку
```

```
@Test
public void testVolumeControl() {
    AudioPlayer player = new AudioPlayer();
    player.setVolume(50);
    assertEquals(player.getVolume(), 50);
}

// 10. Тестування збереження стану музичного файлу
@Test
public void testAudioStateSave() {
    AudioPlayer player = new AudioPlayer();
    player.loadAudio("AudioId123");
    player.play();
    // Симулюємо збереження стану після програвання музичного файлу
    AudioState state = player.saveState();
    assertNotNull(state);
}
```