

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіoeлектроніки,
Факультет комп'ютерних наук і технологій
(повне найменування інституту, назва факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

магістр

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДІВ
ЗНАХОДЖЕННЯ ПОДІБНОЇ МУЗИКИ
RESEARCH AND SOFTWARE IMPLEMENTATION OF METHODS FOR
SEARCHING SIMILAR MUSIC

Виконав: студент(ка) 2 курсу, групи КНТ-219м
Спеціальності 122 Комп'ютерні науки
(код і найменування спеціальності)

Освітня програма (спеціалізація)
Системи штучного інтелекту

Корнишов А.А.

(прізвище та ініціали)

Керівник Зайко Т.А.

(прізвище та ініціали)

Рецензент Скрупський С.Ю.

(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
 (повне найменування закладу вищої освіти)

Інститут, факультет ІРЕ, ФКНТ

Кафедра програмних засобів

Ступінь вищої освіти магістр

Спеціальність 122 Комп'ютерні науки

(код і найменування)

Освітня програма (спеціалізація) Системи штучного інтелекту

(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.

С.О. Субботін

“ ” 2020 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Корнишова Алекса Артуровича

(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація методів знаходження подібної музики

Research and Software Implementation of Methods for Searching Similar Music

керівник проєкту (роботи) Зайко Тетяна Анатоліївна, к.т.н., доцент

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “3” листопада 2020 року № 301

2. Строк подання студентом проєкту (роботи) 16 грудня 2020 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Реферат, Перелік скорочень та умовних познач, Вступ, 1. Аналіз проблеми та постановка завдань дослідження, 2. Матеріали і методи, 3. Опис програми, 4. Розробка програми, 5. Експлуатація та тестування програми, 6. Економіко-організаційна частина, 7. Охорона праці та безпека в надзвичайних ситуаціях, Висновки, Перелік джерел посилання, Додатки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5 Основна частина	Зайко Т.А., доцент		
Економіко-організаційна частина	Пожуєва Т.О., професор		
Охорона праці та безпека в надзвичайних ситуаціях	Коробко О.В., ст. викладач		
Нормоконтролер	Дейнега Л.Ю., ст. викладач		

7. Дата видачі завдання “ 1 ” вересня 2020 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи	1 тиждень	Завдання, ТЗ
2	Аналіз та дослідження предметної області	2-5 тижні	Розділи 1, 2
3	Розробка архітектури програми	6 тиждень	Розділ 3
4	Розробка програми	7-9 тижні	Розділ 4
5	Тестування та експериментальне дослідження програми	10 тиждень	Розділ 5
6	Розробка економіко-організаційної частини	11 тиждень	Розділ 6
7	Розробка заходів з охорони праці та безпеки в надзвичайних ситуаціях	11 тиждень	Розділ 7
8	Оформлення пояснювальної записки та документів до неї. Нормоконтроль та рецензування	12 тиждень	Додатки
9	Захист роботи	13 тиждень	

Студент(ка)

_____ Корнишов А.А.
(підпис) (прізвище та ініціали)

Керівник проєкту (роботи)

_____ Зайко Т.А.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра:
117 с., 36 рис., 7 табл., 3 дод., 33 джерела.

ПОШУК МУЗИКИ, ПОДІБНІСТЬ, КЛАСТЕРИЗАЦІЯ, СУМІШ
ГАУСІВСЬКИХ РОЗПОДІЛІВ, JAVA, MYSQL, TELEGRAM.

Об'єкт дослідження – процес пошуку музики.

Предмет дослідження – методи знаходження подібної музики.

Мета роботи – розробка програмного забезпечення знаходження подібної музики для реалізації пошуку нової музики користувачами.

Матеріали, методи та технічні засоби: мова програмування Java, об'єктно-орієнтоване програмування, система управління базою даних MySQL, месенджер Telegram.

Результати. Визначено функціональні вимоги до програми, прийнято рішення щодо вибору в якості мови програмування для реалізації програми мови Java, виконано проєктування бази даних на основі використання MySQL в якості системи управління базами даних та визначено структуру програмного забезпечення з побудовою відповідної структурної схеми. Реалізовано програмне забезпечення та описано прийняті рішення.

Висновки. Наукова новизна роботи в розробленому методі знаходження подібної музики, який використовує метод кластеризації на основі суміші гаусівських розподілів, визначаючи в якості набору ознак музичної композиції показники акустичності, танцювальності, енергійності, інструментальності, концертності, гучності, мовленнєвості, валентності, ритмічності, що дозволяє виконувати пошук композицій без використання характеристик сигналу з визначенням подібності за кластерами.

Галузь використання – програмні засоби музичних каталогів, програмне забезпечення програвання музики, рекомендаційні музичні системи.

ABSTRACT

Explanatory note to the diploma qualifying work of the master: 117 pages, 36 figures, 7 tables, 3 appendixes, 33 sources.

MUSIC SEARCHING, SIMILARITY, CLUSTERIZATION, GAUSSIAN MIXTURE MODEL, JAVA, MYSQL, TELEGRAM.

The object of research is the process of music searching.

The subject of research is methods of similar music searching.

The purpose of the work is to develop software for similar music searching to give users possibility to search for new music.

Materials, methods and technical means: Java programming language, object-oriented programming, MySQL database management system, Telegram messenger.

Results. The functional requirements for the software are determined, the decision on the choice of the Java language as a programming language for the sofytware implementation is made, the database is designed based on the usage of MySQL as a database management system and the software structure is determined with the appropriate structure diagram. The software is implemented and the made decisions are described.

Conclusions. Scientific novelty of the work is in the developed method of similar music searching, which uses the method of clustering based on a Gaussian mixture model, defining musical composition indicators of acoustics, dance, energy, instrumentality, concert, volume, speech, valence, rhythm as a set of features, which allows users to search compositions without the usage of signal characteristics with cluster similarity determination.

Field of use is software for music catalogs, music playback software, recommended music systems.

ЗМІСТ

	С.
Перелік скорочень та умовних познач.....	8
Вступ.....	9
1 Аналіз проблеми та постановка завдань дослідження	10
1.1 Принципи організації сучасного музичного сервісу	10
1.2 Огляд музичних сервісів	13
1.3 Висновки за розділом 1 та постановка завдань дослідження.....	18
2 Матеріали і методи.....	20
2.1 Методи підбору музики.....	20
2.2 Методи кластеризації.....	21
2.3 Метод кластеризації на основі Gaussian mixture model	27
2.4 Метод знаходження подібної музики	28
2.5 Висновки за розділом 2	30
3 Опис програми.....	31
3.1 Визначення функціональних вимог до програми.....	31
3.2 Вибір мови програмування	32
3.3 Визначення структури бази даних	35
3.4 Визначення структури програмного забезпечення	38
3.5 Висновки за розділом 3	39
4 Розробка програми	41
4.1 Опис логіки функціонування програми.....	41
4.2 Опис розроблених класів.....	42
4.3 Висновки за розділом 4	47
5 Експлуатація та тестування програми	48
5.1 Призначення програми	48
5.2 Умови запуску програми.....	48
5.3 Робота з програмою	48
6 Економіко-організаційна частина.....	59
6.1 Планування розробки програмного продукту.....	59

6.2	Визначення витрат на розробку програми	62
6.3	Розрахунок капітальних вкладень	68
6.4	Розрахунок експлуатаційних витрат	70
6.5	Визначення економічної ефективності програмного продукту	70
6.6	Техніко-економічні показники	72
7	Охорона праці та безпека в надзвичайних ситуаціях	73
7.1	Аналіз потенційних небезпек.....	73
7.2	Заходи по забезпеченню безпеки	73
7.3	Заходи по забезпеченню промислової санітарії та гігієни праці	74
7.4	Заходи по забезпеченню пожежної безпеки.....	77
7.5	Заходи по забезпеченню безпеки у надзвичайних ситуаціях	80
	Висновки	82
	Перелік джерел посилання	83
	Додаток А Технічне завдання	87
	Додаток Б Текст програми	92
	Додаток В Слайди презентації.....	110

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

ЕМ – expectation-maximization;

GMM – Gaussian mixture model;

ВДТ – візуальний дисплейний термінал;

ЕОМ – електронно-обчислювальна машина;

КТЗ – комплекс технічних засобів;

ОЦ – обчислювальний центр.

ВСТУП

Споживання музики продовжує зростати з кожним роком по всьому світу вибуховими темпами, сьогодні люди слухають музику всюди: на вулиці, в громадському транспорті, в машині, на роботі, вдома, музика вбудована в повсякденне життя людей в усьому світі, як ніколи раніше, а музична індустрія забезпечує музичний ринок продукцією самих різних музичних жанрів, стилів і напрямків [1].

Сучасна музична індустрія – це створення, виробництво і поширення музичних творів, при цьому музична індустрія не тільки орієнтується на попит, а й формує його, виходячи із необхідності забезпечення її розвитку та отримання прибутку [1].

З розвитком музичної індустрії загалом збільшуються обсяги фінансів, які заробляються індустрією. З іншого боку чим більше стає музичної продукції, тим складніше в ній орієнтуватися. Завжди актуальне завдання пошуку нової музики відповідно стає ще більш актуальним через збільшення масиву музики, в якому такий вибір можна зробити. Одночасно все складніше стає знаходити свій шлях до слухача новим виконавцям. Тому важливість створення програмного забезпечення, яке дозволить користувачам знаходити подібну музику на ту, яку вони прослуховують, стає все більшою з часом разом з розвитком інформаційних технологій зокрема, адже ті в свою чергу призводять до розвитку методів, які можна використовувати для знаходження музики.

Метою дипломної кваліфікаційної роботи магістра було визначено розробку програмного забезпечення знаходження подібної музики для реалізації пошуку нової музики користувачами.

1 АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

1.1 Принципи організації сучасного музичного сервісу

Стрімінг – це потокове онлайн-мовлення: до стрімінгових мультимедіа відносять музику, відео та інформацію, які користувач отримує безперервно від провайдера потокового мовлення, велика частина ринку стрімінга відноситься до сегменту кіберспорту: геймери показують в режимі онлайн, як вони проходять ігри, стримерами називають також авторів відеоблогів в жанрі музики, кулінарії, подорожей, розпакування та інших напрямках [2].

Стрімінгові (вони ж потокові) сервіси працюють за принципом передачі контенту від провайдера до користувача – весь контент вже завантажений на сторонньому сервері, кінцевому користувачу не потрібно нічого завантажувати для перегляду або прослуховування, контент транслюється в режимі реального часу, швидкість підвантаження безпосередньо залежить від швидкості інтернету користувача, але з нинішніми швидкостями інтернету можна без проблем прослуховувати музику та переглядати відео зі стрімінгових сервісів, при цьому перегляд контенту онлайн став відмінною заміною скачуванню файлів, не потрібно витрачати вільний простір на жорсткому диску, а відразу можна переглядати або слухати контент, у той же час популярні сервіси легалізують перегляд всього того, що бажає користувач, тому що користувач оплачує підписку на певний час, виходить те ж телебачення (або радіо), але з більш розширеною функціональністю і відсутністю запланованої програми передач з обов'язковими рекламними паузами [3].

Зараз у всьому світі відбувається достатньо активний ріст стрімінгу музики.

Разом з розвитком технологій змінюється культура прослуховування музики, від якої безпосередньо залежить вся музична індустрія – сьогодні музичний бізнес процвітає не за рахунок продажу композицій на фізичних

носіях або в цифровому форматі, а завдяки передплатам на стрімінгові платформи, багато в чому саме стрімінг визначив феноменальне зростання доходів від записаної в США музики – на 18% в першій половині 2019 року: щомісяця на стрімінгові сервіси підписується понад мільйон нових користувачів [4].

Стрімінгові сервіси як вид покупки музичного контенту набувають популярності з декількох причин:

- розвиток мережних технологій (в тому числі поява 4G, LTE) дає можливість користуватися інтернетом на високій швидкості, майже без обмежень за часом – в таких умовах завантажування музики стає кілька невиправданим;

- багато користувачів визнають, що прослуховувати музику онлайн зручніше при наявності швидкого інтернету, так як це дає необмежений доступ до різноманітних композицій, крім цього, багато сервісів пропонують слухачам готові плейлисти або підбирають музику на основі аналізу їх музичних смаків;

- більшість стрімінгових сервісів працюють на основі передплати, за яку користувач платить близько 10\$ на місяць, а часто ця цифра навіть менша, це дає йому право прослуховувати всі композиції, що знаходяться в базі програм [5].

Американська асоціація звукозаписних компаній підвела підсумки десятиліття, опублікувавши статистику музичних релізів з 2010 по 2019 роки, за цією статистикою:

- на ринку музики тепер домінують стрімінгові сервіси;

- на початку десятиліття лише 7 % доходів доводилося на сервіси, а кількість передплатників становила близько 1.5 мільйонів, в першу половину 2019 року ці ж показники зросли до 80 % і 61 мільйона передплатників відповідно;

– примітно, що в 2010 році найпопулярнішими були фізичні носії – на них припадало 52 % доходів, друге місце займали цифрові завантаження з 38 %, зараз же ці показники становлять по 9 % у кожного формату [6].

Одночасно за даними асоціації в 2015-2016 роках доходи від продажу музики в США демонстрували вперше стабільне зростання, що відбувалося вперше з кінця 90-х, коли CD-диски були на піку популярності, а про продаж музики через інтернет не було й мови, у той же час головним джерелом прибутку для музикантів протягом 2016 року стали платні підписки на стрімінгові сервіси – платформи для потокового відтворення музики (Spotify, Apple Music, Deezer): з \$3,4 млрд., зароблених на музиці в США, \$1,6 млрд були зароблені завдяки стрімінговим сервісам, в той час як за прямі скачування музики з інтернет-магазинів (iTunes, Google Play) користувачі заплатили \$1,2 млрд., отже, на фізичні носії (CD, вініл) американці витратили близько \$600 млн. [7].

Головними перевагами стрімінгових сервісів вважаються невелика вартість використання і універсальність, крім того, користувачі прислухаються до музичних рекомендацій за стилями і жанрами і слухають плейлисти, створені за допомогою аналізу їх переваг, розміщення музики на стрімінгових сервісах вигідно і музикантам, крім роялті за прослуховування, вони отримують нових слухачів [7].

Стрімінгові сервіси зараз мають величезний вплив на сучасну музичну індустрію, до прикладів, що підтверджують даний факт, належать:

– однією з головних змін, які спричинили за собою стрімінгові сервіси, є так зване потрекове прослуховування: сучасна тенденція така, що люди пвіддають перевагу альбомам над окремими композиціями виконавців, про це свідчить існування добірок під смаки слухача на всіх стрімінгових платформах;

– такі впливові видання в світі музики як Billboard і MetalHammer використовують в своїх рейтингах дані потокових сервісів: при складанні BillboardHot 100 тепер використовуються дані сервісу YouTube, а також

сервісів потокової музики, глобальна компанія-вимірювач Nielsen також використовує в своїх дослідженнях дані потокових сервісів, таких як YouTube і Yahoo! Music [5].

1.2 Огляд музичних сервісів

Відповідно для виявлення тенденцій, яким піддається розвиток сучасних музичних сервісів, потрібно проаналізувати існуючі програмні застосунки, що використовуються в цій сфері зокрема в розрізі знаходження подібної музики як двигуна втягування користувача в потік прослуховування музики, а значить і користування застосунком.

Найбільш популярним музичним сервісом на даний момент є Spotify (рис. 1.1), тому логічно розпочати огляд саме з нього.

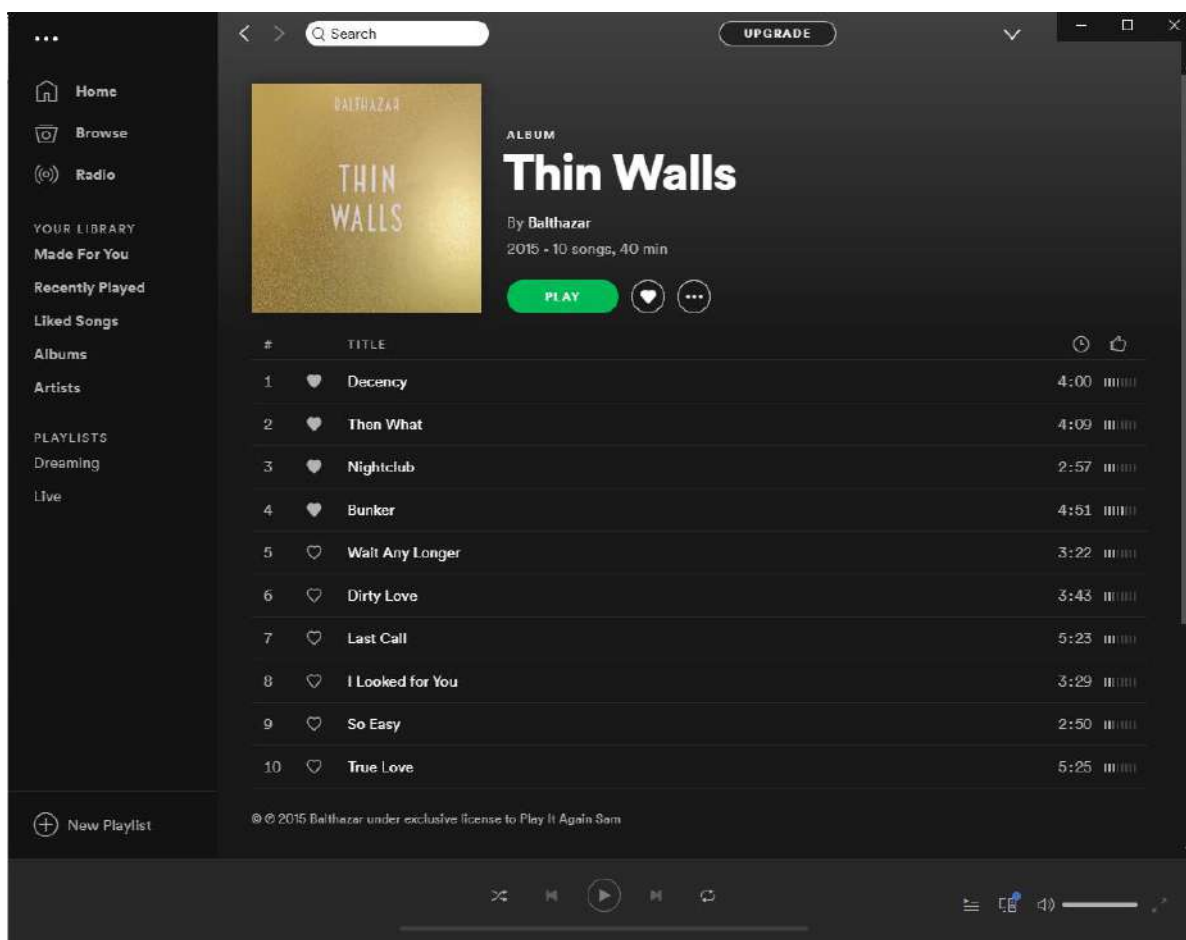


Рисунок 1.1 – Вигляд Windows-застосунку Spotify

Загалом до переваг сервісу Spotify належать:

- медіатека більш 50 мільйонів пісень;
- є сімейний тариф і студентська підписка;
- наявність для любителів бігу можливості Spotify Running;
- гарна якість звуку: в преміум-доступі 320 Кбіт/с у форматі Ogg Vorbis (вище, ніж MP3);
- дуже точні алгоритми підбору композицій за перевагами користувача;
- інтеграція з PlayStation, соцмережею Facebook, сервісами SoundHound і Shazam;
- сервіс спритно працює з будь-якими операційними системами і практично всіма пристроями: існує застосунок як для iOS, Android, так і для Windows;
- немає реклами в платній підписці взагалі;
- функціональність повноцінної музичної соцмережі;
- доступ до прослуховування музики онлайн в безкоштовному режимі;
- можливість зберігати треки на телефоні і слухати без інтернет-з'єднання;
- мільйони кураторських плейлистів, які створюються кожен день, і щотижневий Discover Weekly – ретельно опрацьована добірка нових виконавців за смаками користувача від музичних експертів;
- можна ділитися своєю колекцією з друзями, бачити, що слухають вони, і створювати спільні листи [8].

Окрім широко відомої можливості рекомендацій, які поновлюються раз на тиждень, сервіс ще надає можливість прослуховування подібної музики. Дана функціональність оновлюється постійно і відповідно не вимагає очікування. На рис. 1.2 продемонстровано приклад переліку подібних музичних альбомів на прослуханий користувачем альбом.

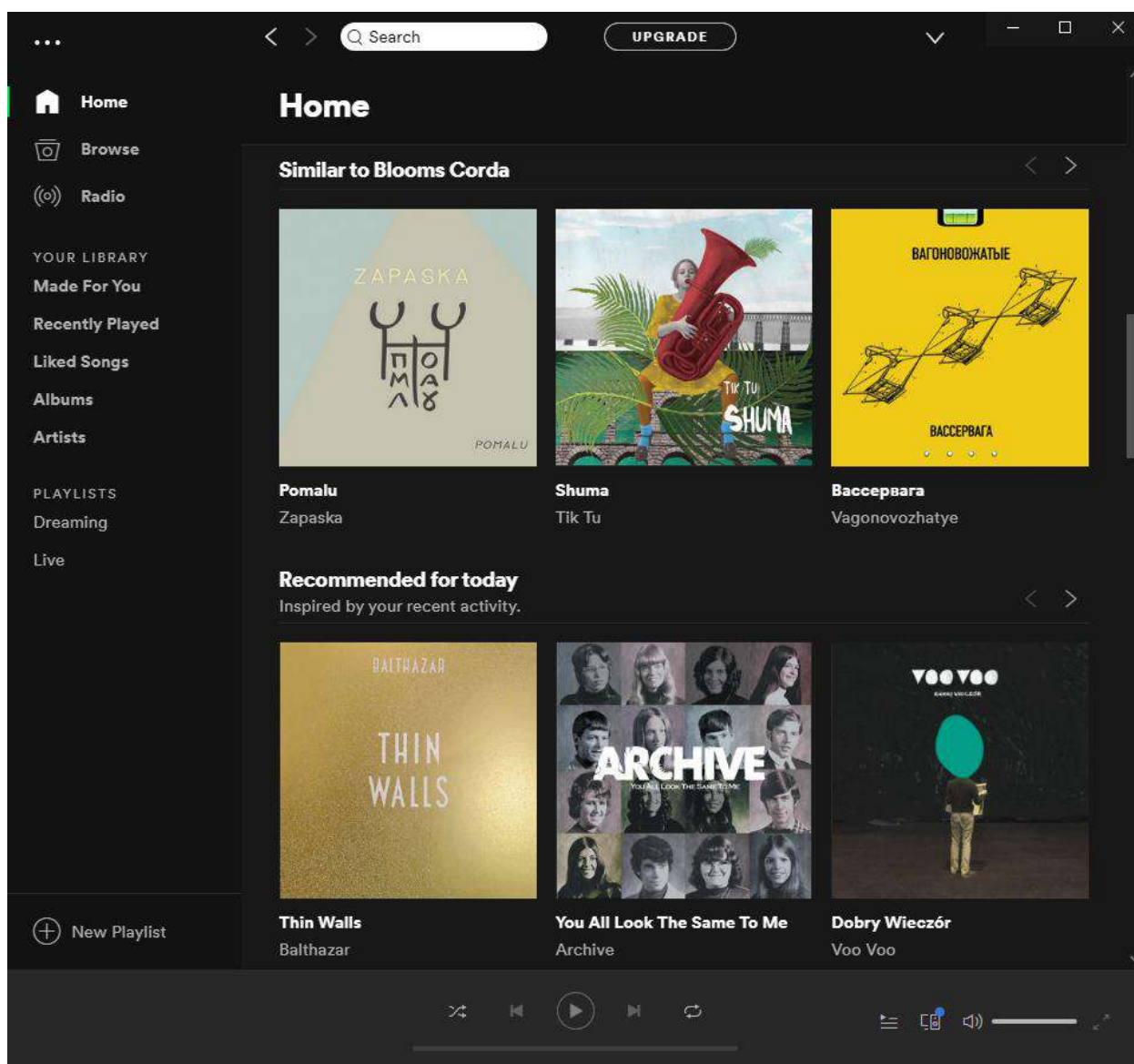


Рисунок 1.2 – Відображення подібної музики в застосунку Spotify

Сервіс Deezer (рис. 1.3) був створений у Франції як сервіс прослуховування музики на різних девайсах: зараз він працює на комп'ютерах, планшетах і смартфонах з операційними системами iOS, Windows, Android, крім того, скористатися сервісом можуть власники телевізорів з Apple TV, Smart TV і Android TV, а також смарт-годинниками на платформах Apple і Android (як Gear S3 Classic, наприклад) [9].

Позитивні сторони Deezer:

- багата фонотека – більш 53 мільйонів пісень та подкастів;
- достатньо якісний бітрейт – 320 Кбіт/с;
- невисоке споживання трафіку;

- вибір звуку в якості lossless – 1411 Кбіт/с в кодеку FLAC (тільки для пристроїв, що сприймають таку швидкість передачі);
- річна підписка по пільговій ціні;
- є бібліотека текстів пісень з інтерактивним рядком, як в караоке;
- опція Flow для любителів слухати персональні добірки незнайомих треків упереміш з власною колекцією;
- вибір плейлистів і створення своїх музичних списків;
- застосунок працює на всіх пристроях і операційних системах;
- немає реклами;
- можливість завантажувати музику і слухати без мережі;
- дуже зручний і зрозумілий інтерфейс як в вебверсії, так і в мобільному застосунку;
- достатня кількість кураторських і призначених для користувача плейлистів, які можна зберігати в своєму профілі або ділитися ними;
- інтеграція з соцмережами Facebook, Twitter і LastFM [8].

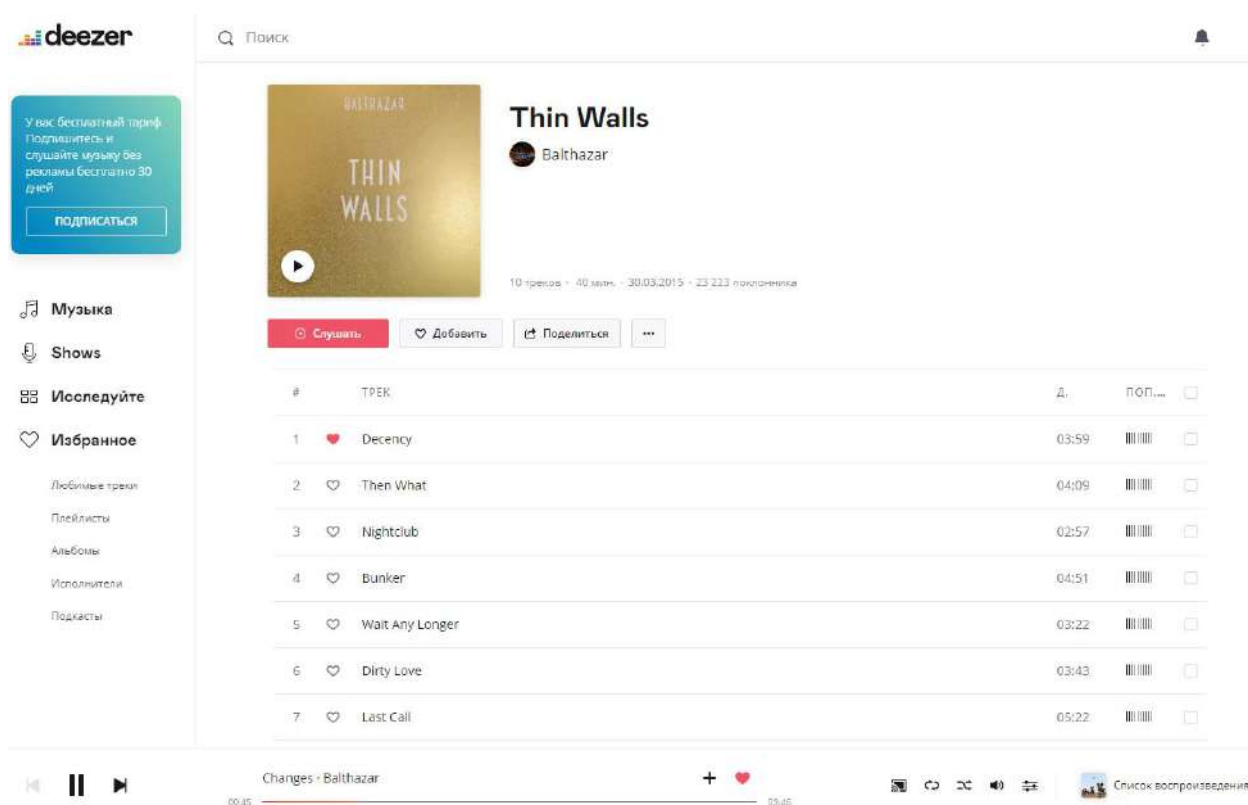


Рисунок 1.3 – Вигляд застосунку Deezer

Deezer такж дозволяє переглядати схожих виконавців (рис. 1.4), тобто загалом це можна назвати важливою особливістю сучасних музичних сервісів.

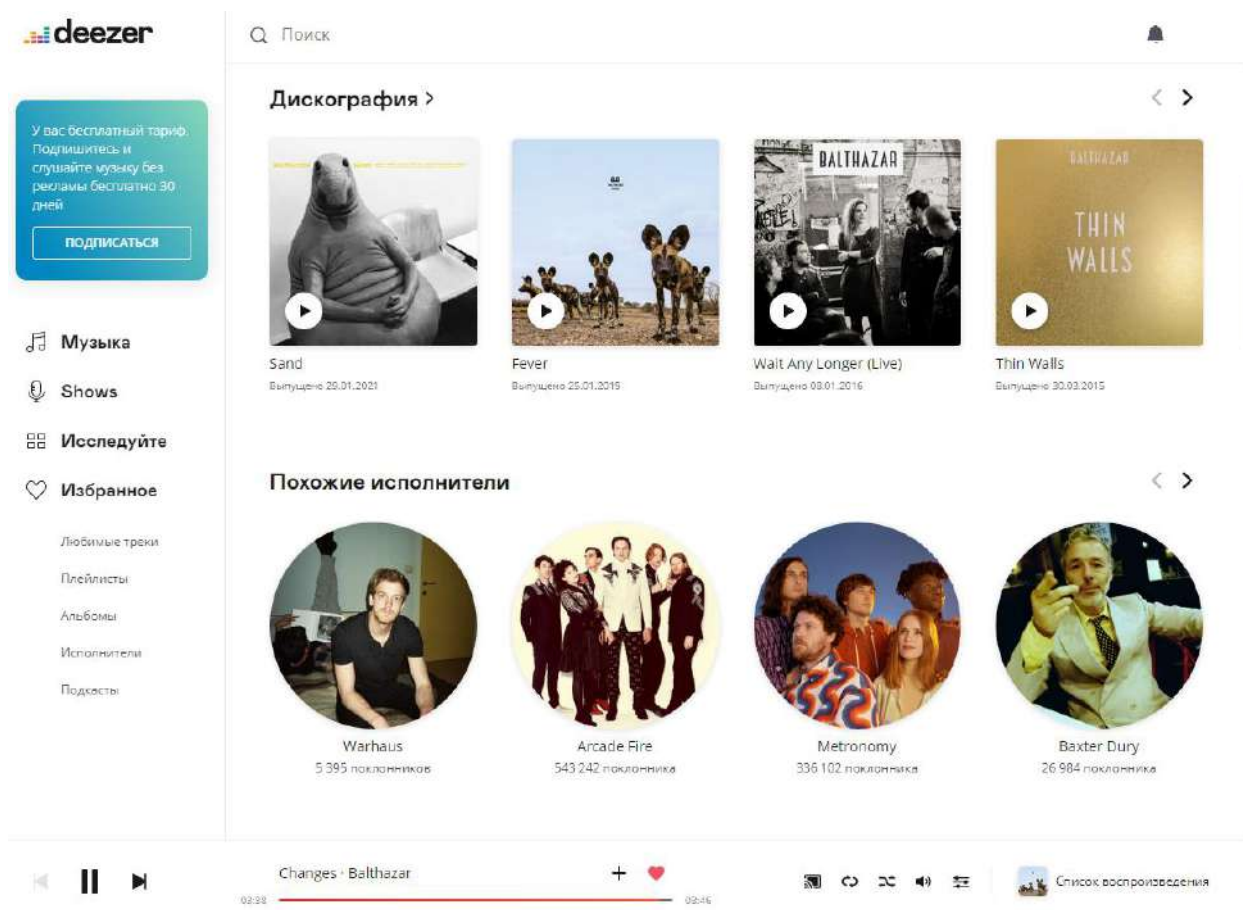


Рисунок 1.4 – Відображення подібної музики в застосунку Deezer

Choosic – це застосунок для iOS, який пропонує користувачу нову музику, дізнаючись про смаки користувача все більше по мірі користування застосунком:

– Choosic виконаний в стилі Tinder, спочатку потрібно вибрати улюблені музичні стилі, потім будуть пропонувати різні треки, під кожним є дві кнопки, натиснувши на іконку серця, користувач відправляє музику до списку вибраного, а застосунок враховує вибір в майбутньому;

– Choosic можна використовувати і як потокове радіо: всі пісні програваться повністю, і кількість пропусків, які можна зробити, не обмежена, хоча головною особливістю застосунку є функція пошуку;

– всі пісні, які сподобалися користувачу, зберігаються в окремий список – там їх можна послухати в будь-який час [10].

Music-Map (рис. 1.5) – це сервіс, який дозволяє виконувати пошук подібних виконавців: потрібно ввести назву улюбленого артиста і отримати список виконавців, які грають в тому ж жанрі, однак тут показуються відразу всі відомі сервісу артисти, а про ступінь схожості можна судити по близькості до зразка, в результаті створюється мапа близькості музичних виконавців [11].



Рисунок 1.5 – Вигляд вебзастосунку music-map.com

1.3 Висновки за розділом 1 та постановка завдань дослідження

У першому розділі розглянуто основні принципи організації сучасних музичних сервісів, виконано огляд особливостей сучасних музичних сервісів, серед яких розглянуто Spotify, Deezer, Choosic, Music-Map.

Виділено, що знаходження нової музики взагалі є актуальним напрямком і сучасні сервіси оснащені тими чи іншими функціями для виконання цієї діяльності. Однією з основних функцій є функція знаходження подібної музики.

Завданнями дослідження було визначено наступні завдання:

- дослідження методів знаходження подібної музики;
- проєктування програмного забезпечення;
- розробка програмного забезпечення;
- тестування та відлагодження програми;
- економічне обґрунтування доцільності розробки;
- забезпечення заходів охорони праці та безпеки в надзвичайних ситуаціях.

2 МАТЕРІАЛИ І МЕТОДИ

2.1 Методи підбору музики

У таких застосунках, як системи підбору (рекомендацій) на основі вмісту, визначення належного показника подібності між елементами має вирішальне значення для багатьох завдань, включаючи пошук та класифікацію найближчого сусіда, у деяких випадках уявлення про подібність може виникнути із знань предметної області, наприклад, косинусова подібність для моделей тексту, однак у більш складних, мультимедійних предметних областях, часто немає очевидного вибору міри подібності, швидше, перегляд різних аспектів даних може призвести до кількох різних і, мабуть, однаково справедливих понять подібності (наприклад, якщо корпус складається з музичних даних, кожна пісня або виконавець можуть бути представлені одночасно акустичними характеристиками (такими як ритм і тембр), семантичними ознаками (теги, тексти пісень) або соціальними особливостями (колаборативна фільтрація, огляди виконавців та біографії тощо)), хоча знання предметної області можуть бути включені, щоб наділити кожне уявлення внутрішньою геометрією і, отже, сенсом подібності – різні поняття подібності не можуть бути взаємопов'язаними (у таких випадках, як правило, не існує очевидного способу поєднання уявлень для формування єдиного простору подібності, який оптимально інтегрує різноманітні дані, без додаткової інформації, яка допомагає будувати міру подібності, ситуація може бути безнадійною), однак, якщо доступна якась побічна інформація, наприклад, маркери, надані людьми (експертами), її можна використовувати для формулювання алгоритму навчання, щоб оптимізувати міру подібності [12].

Саме визначення подібності між об'єктами часто в таких задачах є важливим етапом: незважаючи на те, що евклідова відстань при застосуванні до вихідних ознак забезпечує просту та математично зручну метрику, часто немає підстав вважати, що вона є неоптимальною для даного завдання, отже,

багато досліджень направлено на розробку алгоритмів для автоматичного вивчення метрик відстані в контрольованих налаштуваннях, за невеликим винятком, ці метричні алгоритми навчання відповідають одному і тому ж керівному принципу: кращі сусіди точки повинні знаходитись ближче, ніж її інші сусіди, відповідно до цього принципу, алгоритми метричного навчання часто оцінюються шляхом тестування точності міток, передбачених найближчими сусідами на витягнутах даних [12].

2.2 Методи кластеризації

Кластеризація (або кластерний аналіз) – це задача розбиття множини об'єктів на групи, які називаються кластерами: усередині кожної групи повинні виявитися «схожі» об'єкти, а об'єкти різних груп повинні бути якомога більш відмінні [13].

Загалом серед методів інтелектуального аналізу даних особливе місце займають класифікація і кластеризація:

- класифікація за відомого заздалегідь угруповання даних на підмножини (класи) встановлює закономірність, за якою дані групуються саме таким чином;

- кластеризація, ґрунтуючись на встановленому відношенні схожості елементів, встановлює підмножини (кластери), в які групуються вхідні дані: у подальшому може бути виконаний поглиблений аналіз кожної з виділених підмножин з урахуванням подібності та відмінності об'єктів, встановлених в ході кластеризації;

- завдання кластеризації полягає в розбитті вихідної вибірки, представленій матрицею «об'єкт-властивість», на групи (кластери) таким чином, щоб усередині кожного кластера перебували схожі один на одного об'єкти, а об'єкти різних кластерів істотно відрізнялися, якщо дані уявити як точки в багатовимірному просторі ознак, то задача кластеризації зводиться до відшукування компактних згустків точок [14].

Всі атрибути або ознаки об'єктів діляться на числові і категорійні:

- числові атрибути – це такі атрибути, які можуть бути впорядковані в просторі певним чином;

- категорійні – це такі атрибути, які не можуть бути впорядковані: наприклад, атрибут «вік» – числовий, а «колір» – категорійний, приписування атрибутам значень відбувається під час вимірювань обраним типом шкали, а це являє собою окрему задачу [15].

Кластеризація використовується в різноманітних застосунках, включаючи перераховані нижче:

- для сегментації замовників: можна кластеризувати своїх замовників на основі їх покупок і активності на вебсайті, що корисно для розуміння, хто замовники і чого вони потребують, так що можна адаптувати свої товари і маркетингові кампанії до кожного сегменту, так сегментація замовників може бути корисною в системах видачі рекомендацій для пропозиції вмісту, який сподобалося іншим користувачам в тому ж самому кластері;

- для аналізу даних: коли аналізується новий набір даних, може бути корисно запустити алгоритм кластеризації і потім проаналізувати кожен кластер окремо;

- як методики зниження розмірності: після кластеризації набору даних зазвичай з'являється можливість виміряти схожість кожного зразка з кожним кластером (схожість – це будь-яка міра того, наскільки добре зразок відповідає кластеру), потім вектор ознак кожного примірника може бути замінений вектором його схожості з кластерами, якщо є k кластерів, тоді цей вектор буде k -мірним, зазвичай він матиме набагато меншу розмірність, ніж вихідний вектор ознак, але здатний зберігати достатньо інформації для подальшої обробки;

- для виявлення аномалій (також званого виявленням викидів): будь-який зразок, який має низьку схожість з усіма кластерами, може бути аномалією, наприклад, якщо відбувається кластеризація користувачів вебсайту на основі їх поведінки, то можна виявляти користувачів з

незвичайною поведінкою, такою як нестандартна кількість запитів в секунду, виявлення аномалій особливо корисно при виявленні дефектів у виробництві або при виявленні шахрайства;

– для часткового навчання: якщо в розпорядженні є незначне число міток, то можна було б виконати кластеризацію і поширити мітки на всі зразки в одному і тому ж кластері, така методика здатна значно збільшити кількість міток, доступних для подальшого алгоритму навчання з учителем, і тим самим збільшити його ефективність;

– для пошукових механізмів: певні пошукові механізми дозволяють шукати зображення, які схожі на опорне зображення, для побудови такої системи спочатку застосовується алгоритм кластеризації до всіх зображень у базі даних, схожі зображення опиняться в одному кластері, потім, коли користувач надасть опорне зображення, доведеться лише скористатися навченою моделлю кластеризації для знаходження кластера цього зображення і просто повернути всі зображення зі знайденого кластера;

– для сегментування зображень: за рахунок кластеризації пікселів відповідно до їх кольорів і подальшої заміни кольору кожного пікселя усередненим кольором його кластера стає можливим значне скорочення кількості різних кольорів в зображенні, сегментування зображень застосовується в багатьох системах виявлення та відстеження об'єктів, тому що воно полегшує розпізнавання контурів кожного об'єкта [16].

Застосування кластерного аналізу припускає наступні етапи:

- відбір вибірки для кластеризації;
- визначення множини змінних, за якими будуть оцінюватися об'єкти у вибірці, тобто простору ознак;
- обчислення значень тієї чи іншої міри подібності (або відмінності) між об'єктами;
- застосування методу кластерного аналізу для створення груп схожих об'єктів;
- перевірка достовірності результатів кластерного рішення [14].

Усі методи кластеризації можна поділити на 2 загальні групи:

- ієрархічні;
- неієрархічні.

Неієрархічні алгоритми набули неабиякої популярності, з огляду на те, що в їх основі лежить та чи інша задача оптимізації, тобто групування вихідної множини об'єктів в кластери є рішенням деякої екстремальної задачі, розглянемо декілька найбільш популярних методів:

а) метод k -середніх:

1) ідея методу полягає в наступному – спочатку вибирається k довільних початкових центрів з множини всіх об'єктів, Далі всі об'єкти розбиваються на k груп, найбільш близьких до відповідного центру, на наступному кроці обчислюються центри знайдених кластерів, процедура повторюється ітераційно до тих пір, доки центри кластерів не стабілізуються;

2) метод k -середніх є спрощенням expectation-maximization (ЕМ)-алгоритму: головна відмінність в тому, що в ЕМ-алгоритмі кожен об'єкт розподіляється по всім кластерам з деякою вірогідністю, а в алгоритмі k -середніх кожен об'єкт жорстко приписується тільки до одного кластеру, друга відмінність в тому, що в методі k -середніх форма кластерів не налаштовується;

3) до недоліків цього методу, перш за все, потрібно віднести: наявність апріорної інформації про кількість кластерів, чутливість до ізольованих віддалених елементів, істотну залежність швидкості збіжності методу від початкового вибору центрів кластерів;

б) метод fuzzy c -means: для класичних методів кластеризації існує ефект, який характеризується тим, що декілька елементів вихідного набору даних відносяться до більше ніж одного кластеру, у цьому випадку використання нечітких алгоритмів дозволяє побудувати більш ефективні методи, ніж чіткі, нечітка кластеризація грає важливу роль у вирішенні проблем в областях розпізнавання образів і нечіткої ідентифікації моделі,

алгоритм fuzzy c-means більше підходить для даних, які більш-менш рівномірно розподілені навколо кластерних центрів, і дозволяє використовувати приналежність частини даних двом або більше кластерам;

в) кластеризація Гюстафсона-Кесселя: кластеризація даних за допомогою класичного методу k-середніх не дає задовільних результатів в силу ряду недоліків, пов'язаних, в першу чергу, з використанням евклідової метрики і ігноруванням статистичної залежності між ознаками, за якими здійснюється кластеризація, у разі використання вибірок, що мають несферичні класи, застосування класичного методу k-середніх призводить до невірних результатів (формуються зайві класи або зливаються апріорно різні класи), в зв'язку з тим, що в алгоритмі використовується сферична евклідова відстань, одним з методів усунення такого роду недоліків класичного методу k-середніх є використання методу кластеризації Гюстафсона-Кесселя – цей метод відрізняється від класичного методу використанням замість метрики Евкліда відстані Махалобіса, що дозволяє враховувати статистичну залежність між ознаками спостережень і при наявності цієї залежності поліпшити якість кластеризації;

г) ідея методу Density Based Spatial Clustering of Application with Noise заснована на гіпотезі, що елементи одного кластера формують область, щільність об'єктів всередині якої по деякому заданому порогу перевищує щільність за його межами, у кластері має бути присутня деяка «безперервність» даних, іншими словами, якщо точки лежать недалеко одна від одної, то і значення функції в них не повинні сильно відрізнятися;

д) expectation-maximization-алгоритм: алгоритм заснований на методиці ітеративного обчислення оцінок максимальної правдоподібності, в основі ідеї ЕМ-алгоритму лежить припущення, що досліджувана множина даних може бути змодельована за допомогою лінійної комбінації багатовимірних нормальних розподілів, а метою є оцінка параметрів розподілу, які максимізують логарифмічну функцію правдоподібності, використовувану в якості міри якості моделі, іншими словами,

передбачається, що дані в кожному кластері підкоряються певному закону розподілу, а саме, нормальному розподілу, з урахуванням цього припущення можна визначити параметри – математичне сподівання і дисперсію, які відповідають закону розподілу елементів в кластері, що найкращим чином відповідає спостережуваним даним:

1) серед переваг ЕМ-алгоритму можна виділити наступні: потужна статистична основа, лінійне збільшення складності при зростанні обсягу даних, стійкість до шумів і пропущених даних, можливість побудови бажаного числа кластерів, швидка збіжність при вдалій ініціалізації;

2) алгоритм має і ряд недоліків: по-перше, припущення про нормальність всіх вимірювань даних не завжди виконується, по-друге, при невдалій ініціалізації збіжність алгоритму може виявитися досить повільною, крім цього, алгоритм може зупинитися в локальному мінімумі і дати квазіоптимальне рішення [17]-[18].

Ієрархічна кластеризація – алгоритми кластеризації, що відносяться до цієї групи, називають також алгоритмами таксономії, вони будують не одне розбиття вибірки на непересічні класи, а систему вкладених розбиттів, результат таксономії зазвичай представляється у вигляді таксономічного дерева – дендрограми [17].

Якщо є n точок у d -вимірному просторі, то метою ієрархічної кластеризації є створення послідовності вкладених розділів, які можна зручно візуалізувати за допомогою дерева або ієрархії кластерів, кластери в ієрархіях варіюються від дрібнозернистого до крупнозернистого – найнижчий рівень дерева (листя) складається з кожної точки в своєму кластері, тоді як найвищий рівень (корінь) складається з усіх точок в одному кластері, обидва ці випадки можуть вважатися тривіальними кластерами, на якомусь проміжному рівні можна знайти необхідні значні кластери, якщо користувачеві необхідне задану кількість кластерів k , то можна вибрати

рівень, на якому є k кластерів, існує два основних алгоритмічних підходів до ієрархічної кластеризації:

- агломеративні стратегії працюють від низу до верху, починаючи з кожної з n точок в окремому кластері, вони намагаються об'єднати найбільш схожі пари кластерів, поки всі точки не стануть членами одного і того ж кластера;

- дивізімні стратегії роблять навпаки стосовно агломеративних стратегій, працюючи зверху вниз, починаючи з усіх точок в одному кластері, вони рекурсивно поділяють кластери, поки всі точки не будуть знаходитися в окремих кластерах [17].

2.3 Метод кластеризації на основі Gaussian mixture model

Gaussian mixture model (GMM) або суміш гаусівських розподілів – це імовірнісна модель, яка має на увазі, що всі елементи даних породжені сумішшю кінцевого числа гаусівських розподілів з невідомими параметрами, при цьому GMM розглядається як узагальнення кластеризації за допомогою методу k -середніх, відповідно ЕМ-алгоритм традиційно використовується для знаходження невідомих параметрів GMM, він заснований на максимізації правдоподібності, коли модель залежить від прихованих змінних [19].

GMM – це імовірнісний метод отримання нечіткої класифікації спостережень: розраховується ймовірність належності до кожного кластера, і класифікація зазвичай досягається шляхом віднесення кожного спостереження до найбільш вірогідного кластера, ці ймовірності також можуть бути використані для інтерпретації передбачуваних класифікацій, окрім того моделювання суміші дуже гнучке [20].

Навчання GMM дуже схоже на алгоритм методу k -середніх, воно проходить в два етапи, що відображають аналогічні етапи в методі k -середніх:

– розрахунок приналежностей: для цього вводиться новий символ γ , який означає приналежність k -ої гауссіани до генерації n -ої точки, таким чином, це просто відношення даної гауссіани до суми всіх гауссіан, відповідно якщо π_k буде великим, то дана гауссіана перекриє інші і стане приблизно дорівнювати одиниці, розрахунок відношення відбувається на основі формули (2.1):

$$\gamma_k^{(n)} = p(z^{(n)} | x) = \frac{\pi_k N(x^{(n)} | \mu_k, \Sigma_k)}{\sum_{j=1}^K \pi_j N(x^{(n)} | \mu_j, \Sigma_j)}, \quad (2.1)$$

де μ_k – середнє значення;

Σ_k – коваріація;

x – множина вхідних даних;

– перерахунок всіх параметрів гауссіан, тобто середнього значення, коваріації і π_k : проводиться він також дуже схоже на метод k -середніх – необхідно зважити вплив кожного прикладу на параметр через приналежність, якщо приналежність мала, то і x в загальному підрахунку буде мало означати, для цього використовуються формули (2.2)-(2.5):

$$\mu_k = \frac{1}{N_k} \sum_{n=1}^N \gamma_k^{(n)} x^{(n)}, \quad (2.2)$$

$$\Sigma_k = \frac{1}{N_k} \sum_{n=1}^N \gamma_k^{(n)} (x^{(n)} - \mu_k)(x^{(n)} - \mu_k)^T, \quad (2.3)$$

$$\pi_k = \frac{N_k}{N}, \quad (2.4)$$

$$N_k = \sum_{n=1}^N \gamma_k^{(n)} \quad [21]-[23]. \quad (2.5)$$

2.4 Метод знаходження подібної музики

На основі проведеного аналізу було запропоновано наступний метод знаходження подібної музики.

На першому етапі методу має бути сформовано вибірку даних. Вибірка складається з екземплярів, кожен з яких характеризує відповідну музичну композицію.

Кожен екземпляр характеризується наступними властивостями:

- акустичність;
- танцювальність;
- енергійність;
- інструментальність;
- концертність;
- гучність;
- мовленнєвість;
- валентність;
- ритмічність.

Значення кожного показника є оцінкою в діапазоні від 0 до 1.

На другому етапі застосовується модель GMM, в результаті чого визначаються кластери музичних композицій за вибіркою даних, сформованою за результатом першого етапу. Обрано саме модель GMM, оскільки вона дозволяє не обмежувати кожен екземпляр тільки одним кластером.

На третьому етапі користувач вибирає або задає певну музичну композицію. На основі сформованої моделі, отриманої за результатом другого етапу, виконується визначення кластеру, до якого належить дана композиція. В результаті роботи даного етапу (і методу загалом) повертаються музичні композиції, що належать даному кластеру з найбільшою вірогідністю. За цим показником сортується перелік, який повертається в результаті. До переліку не включається екземпляри з рівномірним розподілом між декількома кластерами.

2.5 Висновки за розділом 2

У другому розділі проведено аналіз методів, які можуть використовуватися для знаходження подібної музики. Виділено і порівняно методи кластеризації, які можуть використовуватися для об'єднання подібної музики в кластери. Детально розглянуто метод кластеризації GMM.

Наукова новизна роботи в розробленому методі знаходження подібної музики, який використовує метод кластеризації на основі суміші гаусівських розподілів, визначаючи в якості набору ознак музичної композиції показники акустичності, танцювальності, енергійності, інструментальності, концертності, гучності, мовленнєвості, валентності, ритмічності, що дозволяє виконувати пошук композицій без використання характеристик сигналу з визначенням подібності за кластерами.

3 ОПИС ПРОГРАМИ

3.1 Визначення функціональних вимог до програми

Грунтуючись на проведеному в розділі 1 огляді проблеми та виділенні пошуку подібних композицій в якості основної функціональної можливості програми, в якості функціональних вимог до програми було виділено наступний набір:

- додавання, перегляд та редагування виконавців;
- додавання, перегляд та редагування композицій;
- додавання, перегляд та редагування альбомів;
- створення тегів;
- авторизація адміністраторів;
- відтворення композицій;
- пошук композицій;
- пошук виконавців;
- пошук альбомів;
- пошук за тегами;
- навчання моделі;
- пошук подібних композицій.

Вся функціональність, пов'язана з роботою адміністратора, має виконуватися через настільний програмний застосунок. З цим варіантом програми має працювати тільки адміністратор

Через програмний застосунок для месенджера Telegram має безпосередньо реалізовуватися пошук даних про композицію та знаходження подібних композицій.

Описані функціональні вимоги були промодельовані на основі застосування діаграми прецедентів, яку в загальному вигляді щодо роботи з системою продемонстровано на рис. 3.1.

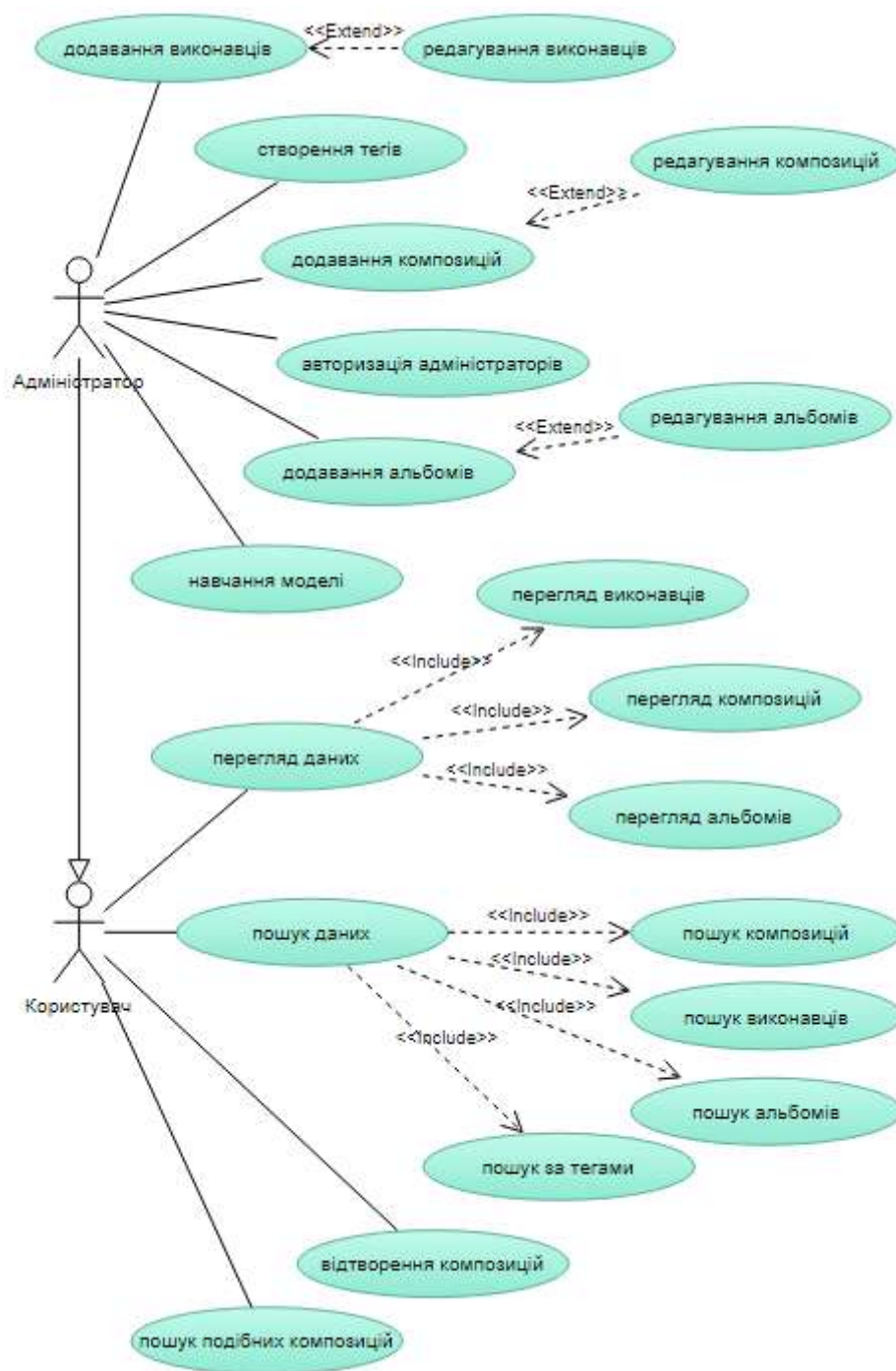


Рисунок 3.1 – Діаграма прецедентів

3.2 Вибір мови програмування

Мова програмування Java має зокрема наступні позитивні особливості:

- простота: мову Java легко вивчити, а її синтаксис досить простий, чистий і зрозумілий, зокрема заплутані та неоднозначні концепції мови C++

або відсутні в Java, або вони були реалізовані більш чисто, так покажчики та перевантаження оператора відсутні в Java, але були важливою частиною C++;

- об'єктно-орієнтованість: у Java все є об'єктом, який має певні дані та поведінку, при цьому Java можна легко розширити, оскільки вона базується на об'єктній моделі;

- надійність: в Java докладено зусиль для усунення коду, схильного до помилок, роблячи основний упор на перевірку помилок під час компіляції та перевірку часу виконання, але основними напрямками, які вдосконалила Java, були управління пам'яттю та неправильна обробка винятків, ввівши автоматичну функцію збирання сміття та обробки винятків;

- незалежність від платформи: на відміну від інших мов програмування, таких як C, C++ тощо, які компілюються в певні версії для машин, що працюють на окремих платформах, Java гарантовано створює код, який можна буде запустити в будь-якому місці, так під час компіляції програма Java компілюється в байткод, цей байткод не залежить від платформи і може запускатися на будь-якій машині, плюс цей формат байт-коду також забезпечує безпеку, будь-яка машина з середовищем виконання Java може запускати програми Java, тому можна говорити про те, що Java – незалежна мова від платформи;

- захищеність: відносно безпеки Java є першим вибором, оскільки завдяки захищеним функціям Java можна розробляти систему без вірусів, програма Java завжди працює в середовищі виконання Java із майже нульовою взаємодією з операційною системою, отже вона є в результаті і більш безпечною;

- багатопоточність: функція багатопоточності Java дозволяє писати програми, які можуть виконувати багато завдань одночасно, перевага багатопоточності полягає в тому, що вона використовує одну і ту ж пам'ять та інші ресурси для одночасного виконання декількох потоків, наприклад, під час введення перевіряються граматичні помилки;

- архітектурна нейтральність: компілятор генерує байткоди, які не мають нічого спільного з конкретною архітектурою комп'ютера, отже програму Java легко інтерпретувати на будь-якій машині;

- портативність: байткод Java можна перенести на будь-яку платформу, немає функцій, що залежать від реалізації, все, що стосується зберігання, є заздалегідь визначеним, приклад: розмір примітивних типів даних;

- висока продуктивність: Java – це інтерпретована мова, тому вона ніколи не буде такою швидкою, як компільована мова, така як C або C++, але Java забезпечує високу продуктивність із використанням своєчасного компілятора;

- розподіленість: Java також є розподіленою мовою, програми можуть бути розроблені для роботи в комп'ютерних мережах, Java має спеціальну бібліотеку класів для взаємодії за допомогою протоколів TCP/IP, створення мережових з'єднань на Java дуже просте порівняно з C/C++ [24].

В якості альтернативи даній мові було обрано мову програмування C#, оскільки загалом вона дозволяє створювати подібні рішення. В якості критеріїв порівняння було обрано наступні:

- об'єктно-орієнтована парадигма: об'єктно-орієнтована парадигма розробки спрощує процес розробки і дозволяє його повторно використовувати;

- залежність від платформи: якщо код не залежить від платформи, то буде достатньо просто створювати послідовність рішень в складі екосистеми підтримки запропонованого методу для різних існуючих платформ;

- безкоштовне рішення: важливий критерій для розробки програми в межах дипломного проєкту;

- безпечність: ступінь впливу застосунку на операційну систему;

- розподіленість: розподіленість дозволяє створювати програми, які працюють швидше за однакових ресурсів у результаті розподілу процесів;

– створення настільних застосунків – основним буде рішення для настільної системи, оскільки саме через нього має відбуватися і навчання в тому числі;

– створення застосунків для Telegram – створення застосунку для Telegram дозволить розширити одразу доступ до розробленого методу.

Результати порівняння зведено до табл. 3.1.

Таблиця 3.1 – Порівняння мов програмування для реалізації програми

Критерій	C#	Java
Об’єктно-орієнтована парадигма	Чиста	Чиста
Залежність від платформи	Орієнтований код на Windows	Ні
Безкоштовне рішення	Ні	Так
Безпечність	Менша	Так
Розподіленість	Так	Так
Створення настільних застосунків	Так	Так
Створення застосунків для Telegram	Так	Так

У результаті за сукупністю всіх критеріїв було прийнято рішення про вибір мови програмування Java для реалізації програми.

3.3 Визначення структури бази даних

Основні сутності предметної області:

- виконавець (музикант) Musician;
- жанр Genre;
- композиція (пісня) Composition;

- альбом Album;

- тег Tag.

Таблиця Musician включає поля:

- ідентифікаційний номер id;

- назва або ім'я name;

- країна country.

Таблиця Genre включає поля:

- ідентифікаційний номер id;

- назва name.

Таблиця Composition включає поля:

- ідентифікаційний номер id;

- назва title;

- номер виконавця musician_id;

- акустичність acousticism;

- інструментальність instrumentalness;

- концертність liveness;

- валентність valence;

- ритмічність tempo;

- мовленнєвість speechness;

- танцювальність danceability;

- енергійність energy;

- гучність loudness.

Таблиця Album включає поля:

- ідентифікаційний номер id;

- назва name;

- дата публікації pub.

Таблиця Tag включає поля:

- ідентифікаційний номер id;

- назва name.

Результуючу схему бази даних представлено на рис. 3.2.

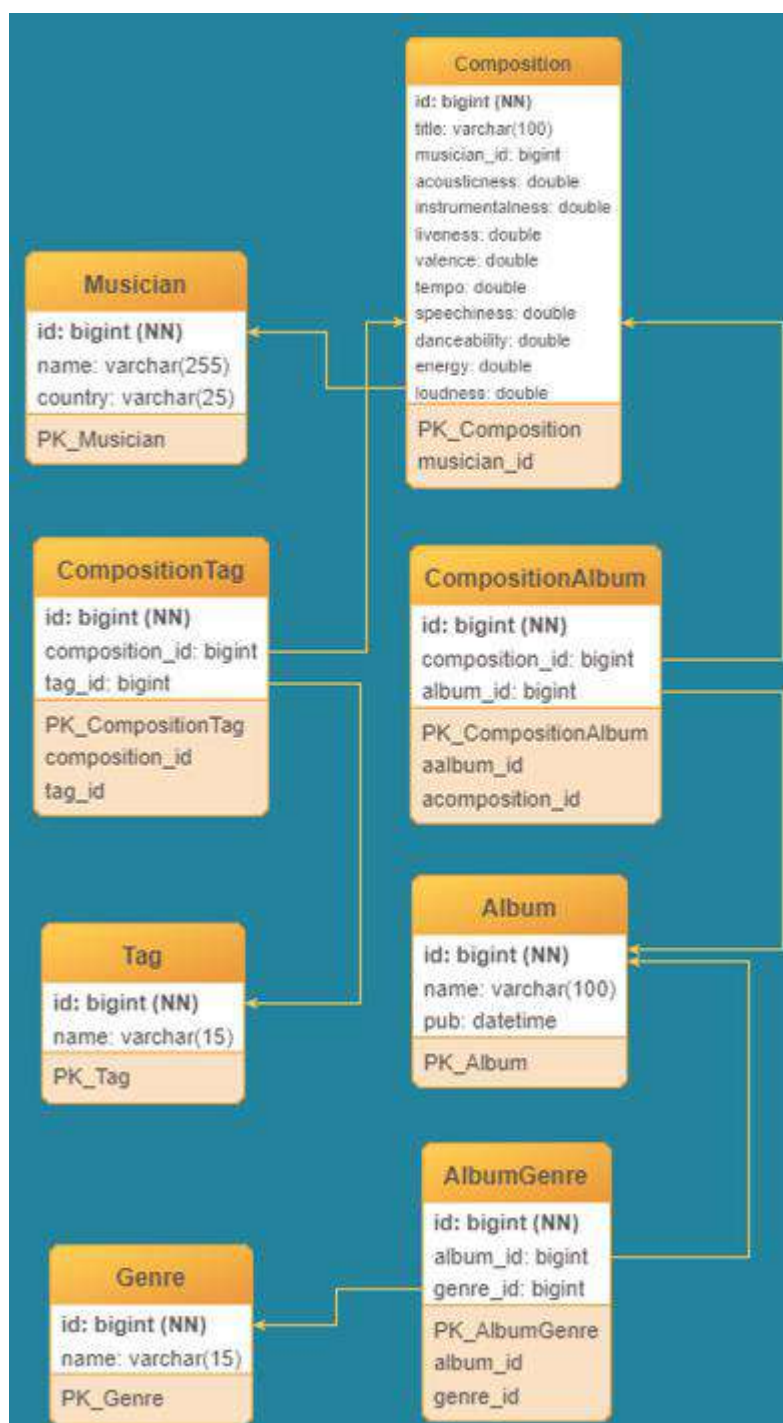


Рисунок 3.2 – Схема бази даних

Для реалізації зв'язків багато-до-багатьох було створено таблиці:

- теги композиції CompositionTag, що реалізує зв'язок між таблицями Composition і Tag: одній композиції дозволяється встановлювати різні теги;
- композиції альбому CompositionAlbum, що реалізує зв'язок між таблицями Composition і Album: один альбом може містити багато пісень;

– жанри альбому AlbumGenre, що реалізує зв'язок між таблицями Album і Genre: один альбом може характеризуватися різними жанрами.

Таблиця CompositionTag включає поля:

- ідентифікаційний номер id;
- номер композиції composition_id;
- номер тегу tag_id.

Таблиця CompositionAlbum включає поля:

- ідентифікаційний номер id;
- номер композиції composition_id;
- номер альбому album_id.

Таблиця AlbumGenre включає поля:

- ідентифікаційний номер id;
- номер альбому album_id;
- номер жанру genre_id.

Реалізацію бази даних виконано за допомогою системи управління базами даних MySQL.

3.4 Визначення структури програмного забезпечення

Структуру програмного забезпечення було визначено, враховуючи необхідність створення двох програмних застосунків:

- настільного застосунку для роботи над підготовкою моделі, внесенням даних;
- застосунку для месенджера Telegram, що призначений для взаємодії користувача з програмним забезпеченням.

Окрім безпосередньо даних застосунків має бути розроблено підсистему знаходження подібної музики, що реалізує запропонований в роботі метод. Також до складу програмної системи має входити спроектована база даних.

На рис. 3.3 представлено відповідну структурну схему програмної системи.

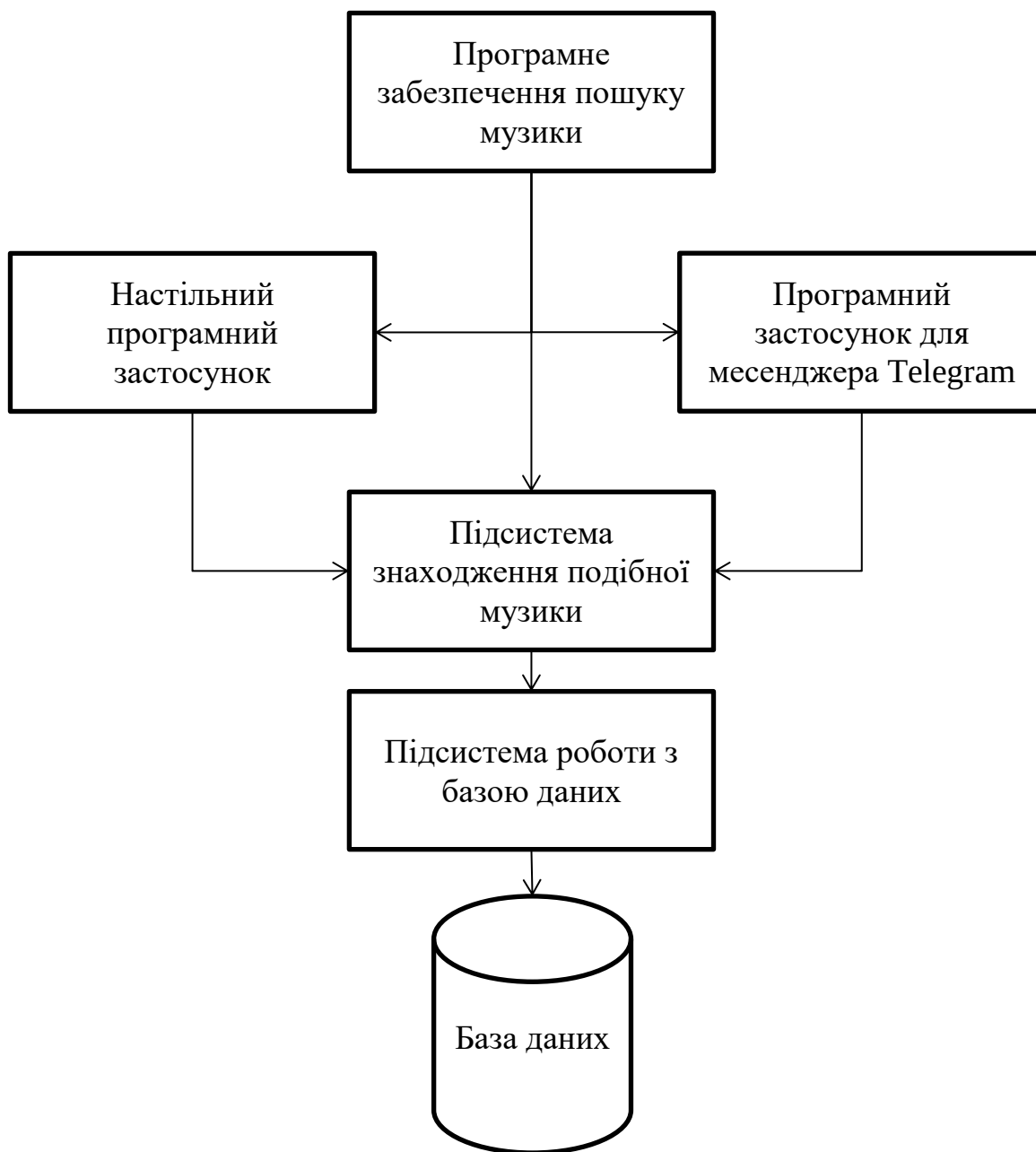


Рисунок 3.3 – Структурна схема програмної системи

3.5 Висновки за розділом 3

У процесі проектування програмного забезпечення було визначено функціональні вимоги до програми, прийнято рішення щодо вибору в якості мови програмування для реалізації програми мови Java, виконано

проєктування бази даних на основі використання MySQL в якості системи управління базами даних та визначено структуру програмного забезпечення з побудовою відповідної структурної схеми.

4 РОЗРОБКА ПРОГРАМИ

4.1 Опис логіки функціонування програми

На рис. 4.1 приведено стандартну логіку процесу взаємодії з застосунком.

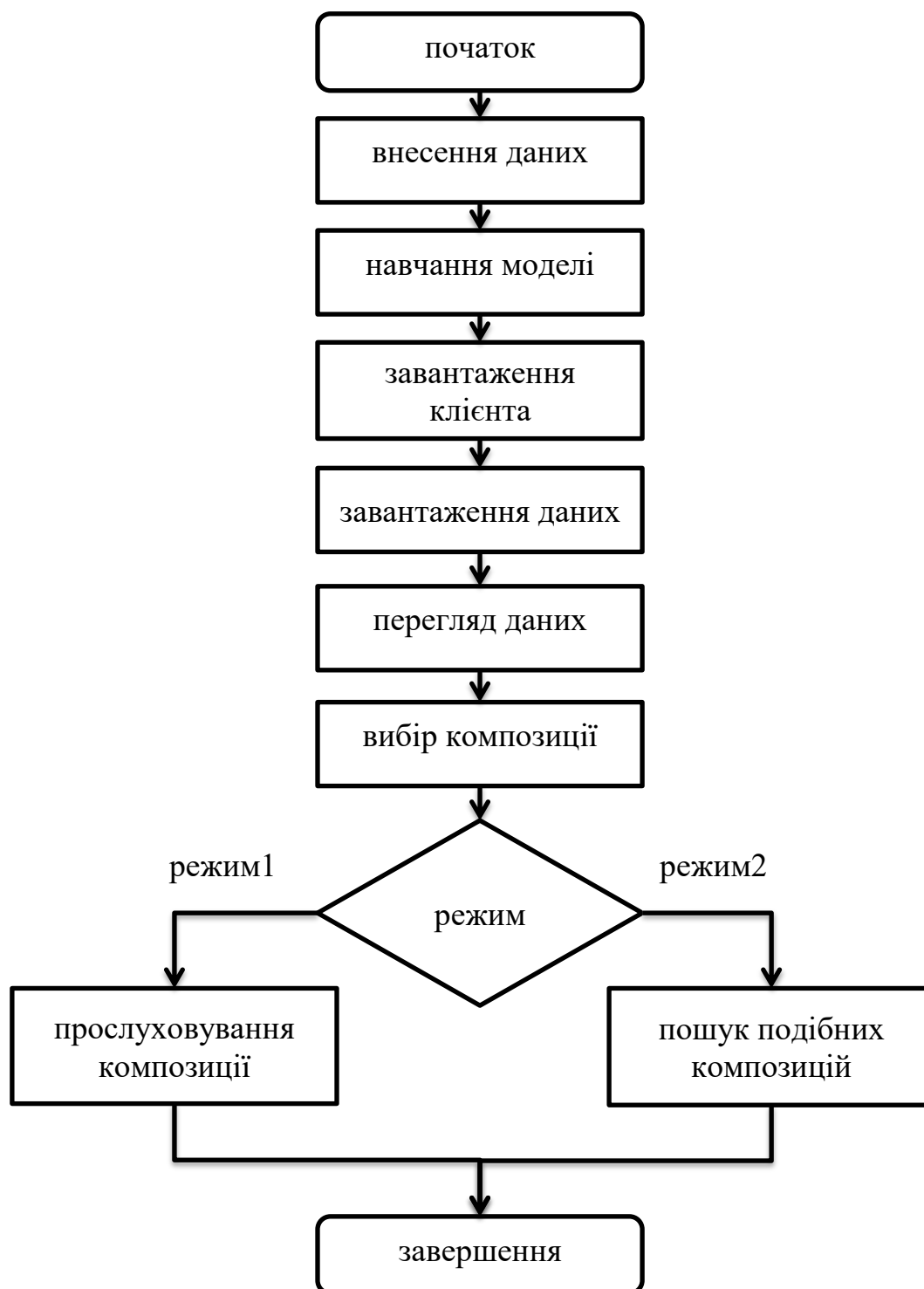


Рисунок 4.1 – Стандартний процес взаємодії з застосунком

4.2 Опис розроблених класів

За результатами виконання розробки програмного забезпечення було створено два програмні застосунки. Обидва було розроблено мовою програмування Java. Перший програмний застосунок використовується в основному адміністраторами і є настільним програмним застосунком. Другий програмний застосунок є інтерфейсом для звичайних користувачів і функціонує за допомогою месенджера Telegram.

В основі і настільного застосунку, і застосунку для Telegram лежить серверна частина, в межах якої реалізовано застосунок для Telegram та серверну частину, що відповідає за отримання запитів від клієнтів, звернення до бази даних, знаходження подібної музики.

Серверна частина міститься в папці Server програмного проєкту. Серед інших тут містяться такі класи.

Клас Server реалізує комунікацію з клієнтом на основі використання сокетів – назва програмного інтерфейсу для забезпечення обміну даними між процесами, процеси при такому обміні можуть виконуватися як на одному комп'ютері, так і на різних комп'ютерах, пов'язаних між собою мережею, це абстрактний об'єкт, який представляє кінцеву точку з'єднання, кожен процес може створити слухаючий сокет (серверний сокет) і прив'язати його до якого-небудь порту операційної системи, слухаючий процес зазвичай знаходиться в циклі очікування, тобто прокидається при появі нового з'єднання, при цьому зберігається можливість перевірити наявність з'єднань на даний момент, зазвичай клієнт явно під'єднується до слухача, після чого будь-яке читання або запис через його дескриптор передаватимуть дані між ним і сервером [25].

Сокети створюються на основі використання пакету `java.net.Socket`. До складу класу Server входять атрибути:

- `private static Socket csock` – під'єднання для очікування підключення до сервера;

– private static ServerSocket ssock – серверний сокет, який працює за портом 25547;

– private static BufferedReader br – буфер читання даних, переданих в сокет;

– private static BufferedWriter bw – буфер запису даних для передачі через сокет.

Після створення з'єднання та ініціалізації каналу для передачі даних (переліку альбомів, що дозволяє цілісно зберігати всі дані, адже в альбомі міститься набір композицій, в композиції – дані про виконавця тощо) відбувається ініціалізація завдання верхнього рівня – очікування запитів на пошук подібних музичних композицій:

```
remBDI.getInstance().bdiAgentFeature.dispatchTopLevelGoal(remBDI.getInstance().CapabilityCompositions.new SocketAccept(imus));
```

Клас remBDI надає доступ до таких методів:

– void created() – метод, який відповідає за оброблення ситуації створення завдання, коли мають бути прочитані з бази даних дані;

– void body() – тіло завдання, тобто метод, який безпосередньо відповідає за запуск вікна роботи з програмою та підвантаження відповідної форми.

Клас CapabilityCompositions створює основні можливості роботи з композиціями. Основою даного класу є атрибут, який зберігає перелік композицій, compositionList. Для даного переліку підпримуються методи:

– public void addComposition (Composition c) – додавання композиції c до переліку;

– public Map<Integer, Composition> getCompositionList () – отримання переліку композицій;

– public void loadCompositionList () – завантаження з бази даних даних до переліку композицій.

Внутрішній клас Search реалізує безпосередньо пошук композицій.

Створено серед інших такі форми, кожна з яких визначається файлами form та java:

- форма роботи з виконавцями MusiciansForm;
- форма створення нового виконавця NewMusicianForm;
- форма редагування виконавця EditMusicianForm;
- форма роботи з альбомами AlbumsForm;
- форма створення нового альбому NewAlbumForm;
- форма редагування альбому EditAlbumForm;
- форма роботи з композиціями CompositionsForm;
- форма створення нової композиції NewCompositionForm;
- форма редагування композиції EditCompositionForm.

Форма роботи з композиціями CompositionsForm реалізується безпосередньо файлом CompositionsForm.form, в якому створюється графічний інтерфейс на основі бібліотеки Swing. Наприклад, створення кнопки реалізується таким чином:

```
<component          id="e6358"          class="javax.swing.JButton"
binding="btnEditComposition">
    <constraints>
        <grid  row="1"  column="0"  row-span="1"  col-span="1"  vsize-
policy="0"  hsize-policy="3"  anchor="0"  fill="1"  indent="0"  use-parent-
layout="false">
            <preferred-size width="300" height="50"/>
        </grid>
    </constraints>
    <properties>
        <text value="Редагувати"/>
    </properties>
</component>
```

Відповідно в класі `CompositionsForm` реалізується обробка натискання на кнопки та заповнення даними. Для цього передбачено наступні методи:

- `public MusicForm()` – конструктор, який реалізує прив'язування обробників подій натискання кнопок;
- `public JPanel getPanel()` – метод, який повертає панель перегляду композицій;
- `public JButton getBtnEditComposition ()` – метод, який повертає кнопку відкриття форми редагування композиції;
- `public void updateCompositions(Map<Integer, Composition> all)` – метод, який виконує оновлення переліку композицій `listCompositions` на основі переданого переліку композицій `all`;
- `public void createWindowEditComposition (String comp)` – метод, який забезпечує відкриття форми редагування композиції;
- `public void createWindowNewComposition ()` – метод, який забезпечує відкриття форми додавання нової композиції.

Для всіх інших форм використано подібну логіку.

Аналогічно реалізовано форми та взаємодію в межах клієнтського застосунку, де роботу виконує звичайний користувач.

На серверній частині також було реалізовано серед інших наступні класи:

- клас `Method` реалізує запропонований метод, керуючи при цьому класом `ClusterMethod`;
- клас `ClusterMethod` реалізує модель GMM;
- клас `Composition` реалізує сутність композиції у вигляді об'єкту;
- клас `Musician` реалізує сутність музиканта у вигляді об'єкту;
- клас `Album` реалізує сутність альбому у вигляді об'єкту;
- клас `Tag` реалізує клас тегів;
- клас `Genre` реалізує клас жанру;
- клас `Database` виконує запити до бази даних.

Для створення застосунку для месенджера Telegram спочатку було зареєстровано застосунок через BotFather (рис. 4.2).

В результаті виконання цих дій було отримано токен, який використовується для ідентифікації застосунку під час звернення до серверу Telegram.

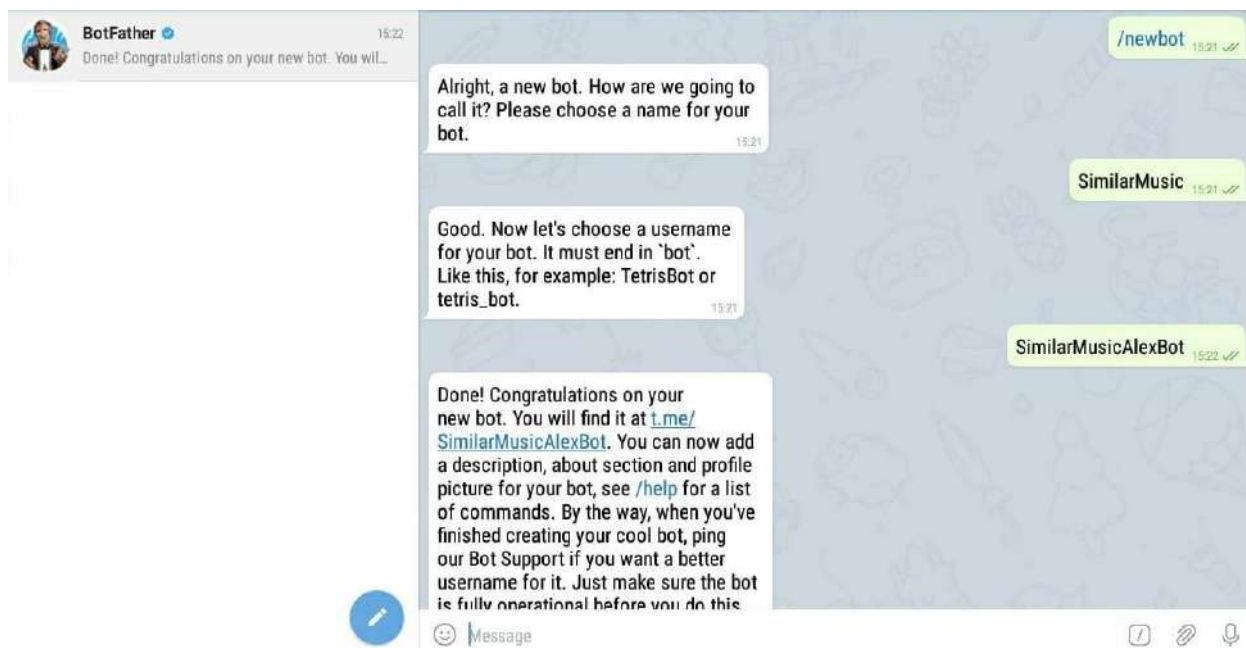


Рисунок 4.2 – Реєстрація програмного застосунку в месенджері Telegram

Програмний застосунок для месенджера Telegram реалізовано за допомогою класу TelegramBot, створеного на основі наслідування класу TelegramLongPollingBot.

Безпосередньо отримання надісланого користувачем повідомлення реалізується наступним чином:

@Override

```
public void onUpdateReceived(Update info) {
    String text = info.getMessage().getText();
    obtainAnswer(info.getMessage().getChatId().toString(), text);
}
```

Метод `obtainAnswer` виконує оброблення отриманого повідомлення, формування відповіді на запит, а також відправлення відповіді в чат з номером, який передано в якості першого параметра під час виклику методу.

4.3 Висновки за розділом 4

Описано реалізацію структури програмного забезпечення за допомогою відповідного набору класів. Описано основні принципи їх створення (як на серверній, так і на клієнтській частинах). Представлено алгоритм знаходження подібної музики в межах функціонування програмного забезпечення.

5 ЕКСПЛУАТАЦІЯ ТА ТЕСТУВАННЯ ПРОГРАМИ

5.1 Призначення програми

Програмне забезпечення дозволяє знаходити нові музичні композиції, подібні до тих, які були прослухані користувачем раніше, та мати базові можливості відтворення музики.

5.2 Умови запуску програми

До складу технічних засобів повинен входити персональний комп'ютер або ноутбук, який включає у себе:

- операційну систему – Windows 10;
- оснащений 256 Мб оперативного запам'ятовуючого пристрою;
- процесор Intel Celeron G1830 2800 МГц;
- має доступ до мережі Інтернет;
- засоби керування та відображення інформації (клавіатура, миша та монітор).

5.3 Робота з програмою

Робота з програмою розпочинається тоді, коли користувач запускає виконуваний файл програми, в результаті чого відображається початкове вікно програми (рис. 5.1).

Початкове вікно програми є вікном пошуку виконавця. Для пошуку виконавця потрібно або обрати його з таблиці, де перелічено всі виконавці, або ввести назву або частину в поле “Виконавець”, а далі натиснути на кнопку “Шукати”, тоді відобразиться перелік всіх виконавців, що відповідають введеному запиту, у таблиці (рис. 5.2).

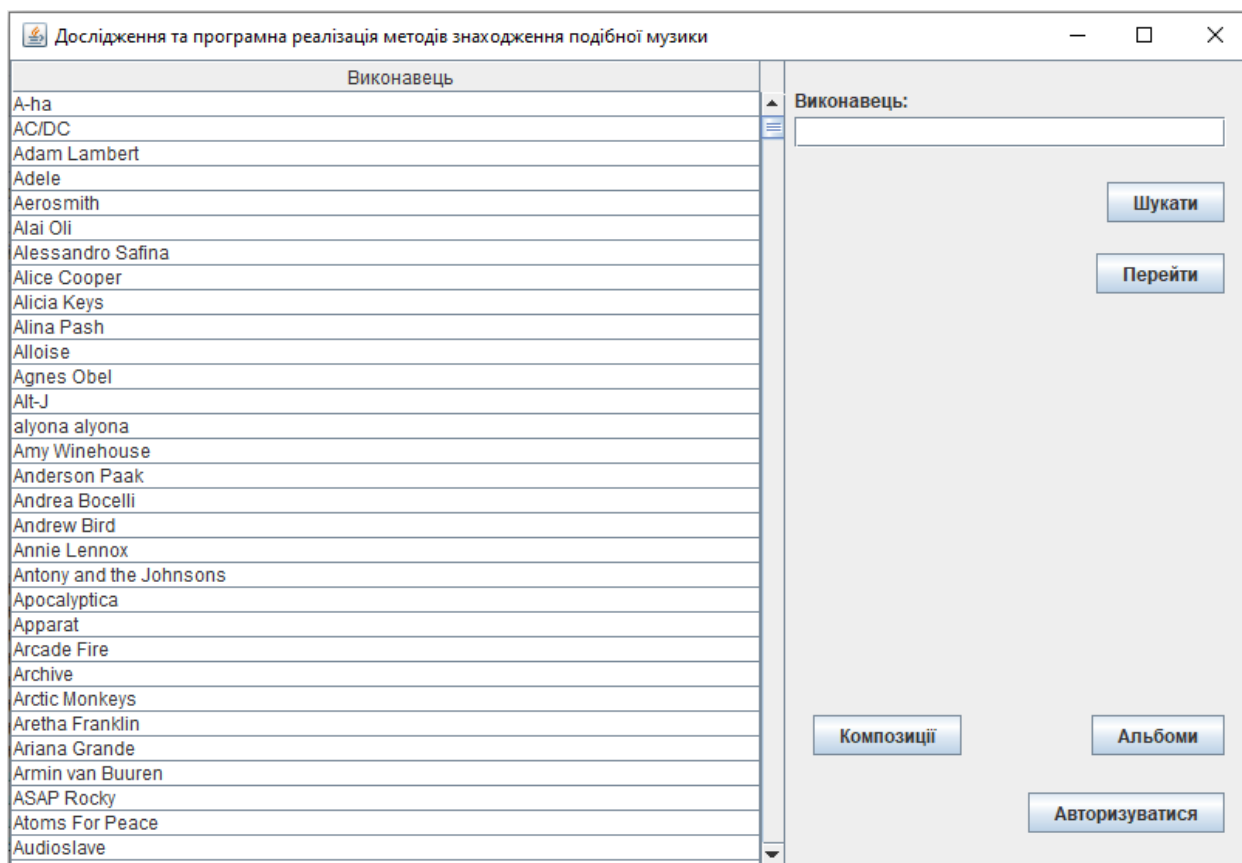


Рисунок 5.1 – Вікно вибору виконавця

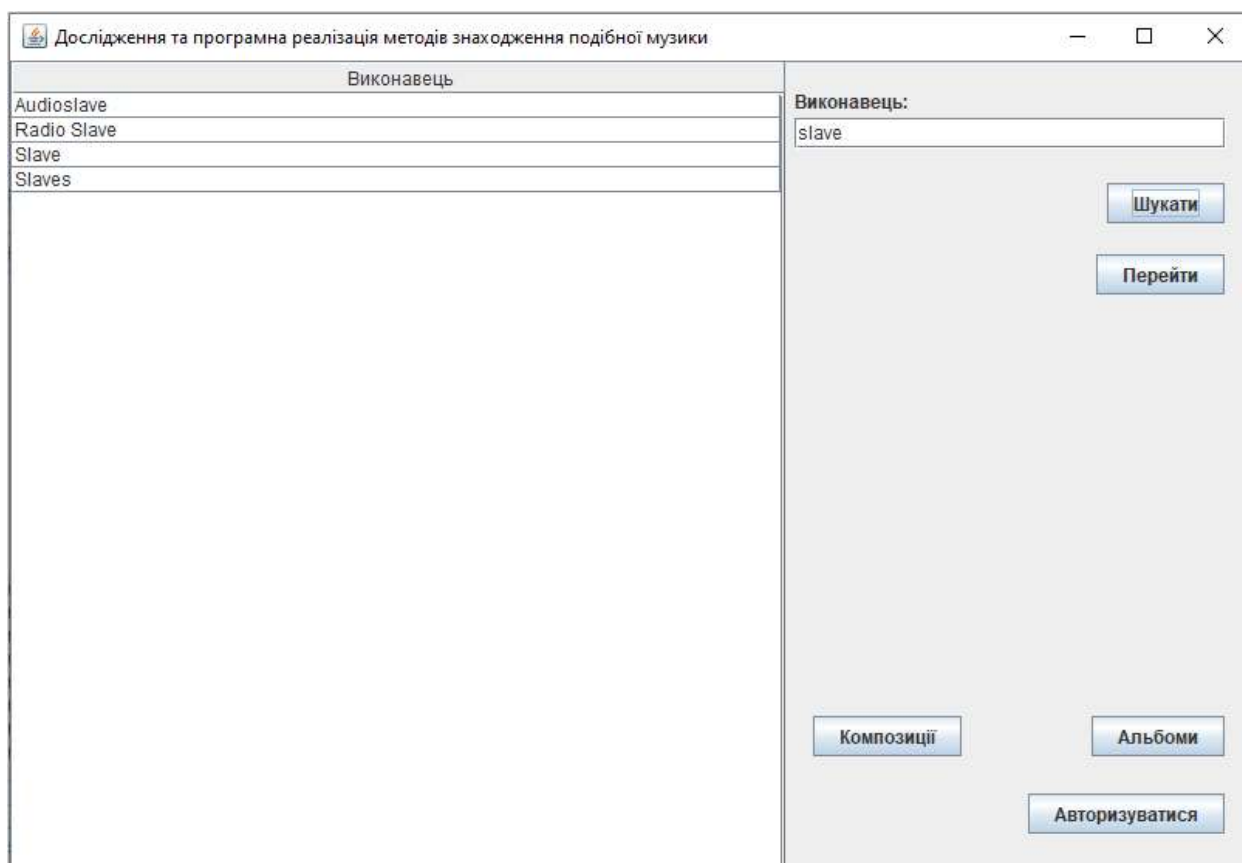


Рисунок 5.2 – Пошук виконавця

Якщо з переліку виконавців обрати потрібного, то можна натиснути на кнопку “Перейти”, тоді відкриється вікно перегляду переліку альбомів даного виконавця (рис. 5.3).

Окрім того можна натиснути на кнопки “Композиції” або “Альбоми” для переходу до вікон роботи з альбомами або композиціями без виконання попереднього вибору (тоді потрібно використовувати пошук за назвою).

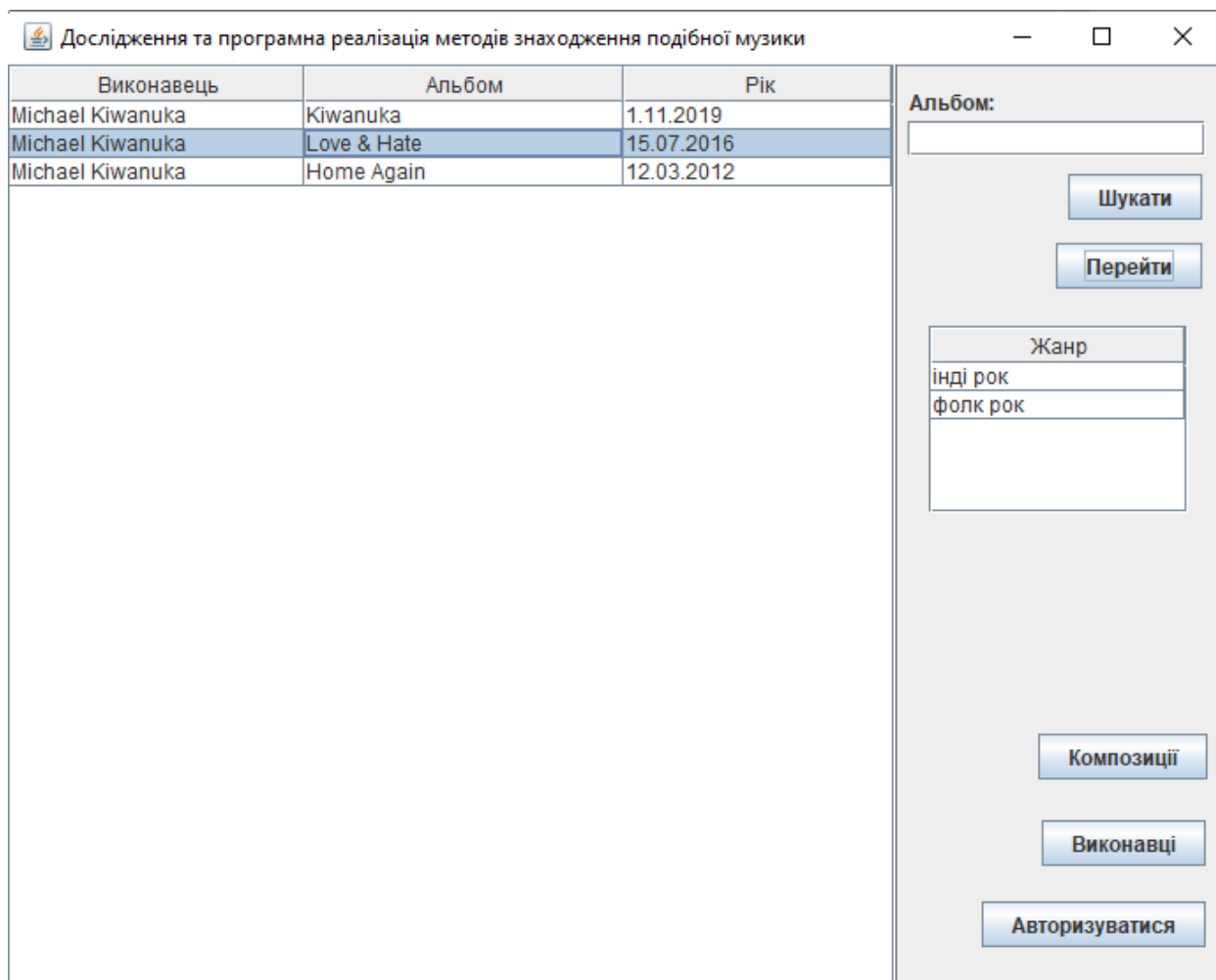


Рисунок 5.3 – Вікно перегляду альбомів виконавця

Якщо обрати один з альбомів, то за допомогою кнопки “Перейти” можна відкрити вікно з переглядом композицій альбому (рис. 5.4). За допомогою кнопки “Композиції” можна перейти до роботи з усіма композиціями, що є в програмі. За допомогою кнопки “Виконавці” можна повернутися до найвищого рівня (роботи з виконавцями).

Пошук за альбомами дозволяє знайти альбом незалежно від імені виконавця.

Можна також переглянути жанри, до яких належить альбом.

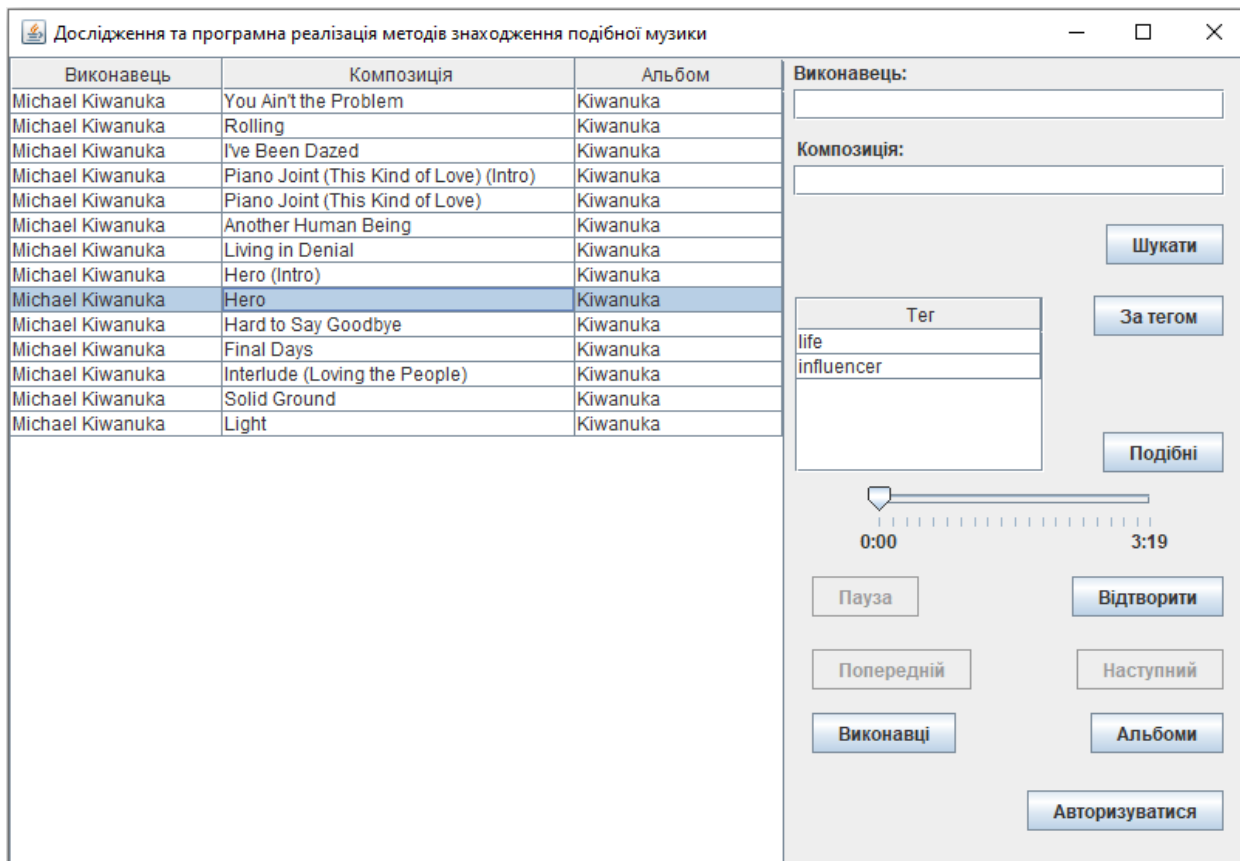


Рисунок 5.4 – Перегляд композицій альбому

У даному вікні можна переглянути теги за композиціями (відображаються при виборі композиції з переліку). Можна виконати пошук за назвою композиції або за назвою композиції та виконавцем.

Якщо обрати з переліку одну з композицій, то можна відтворити її (рис. 5.5), натискаючи на кнопку “Відтворити”, тоді активується повзунок, а також кнопки “Пауза”, що дозволяє зупити відтворення, “Попередній”, що дозволяє перейти в переліку альбому до попередньої композиції, “Наступний”, що дозволяє перейти до наступної композиції альбому.

Кнопки “Виконавці” та “Альбоми” дозволяють перейти на рівень вище, відкриваючи відповідні вікна.

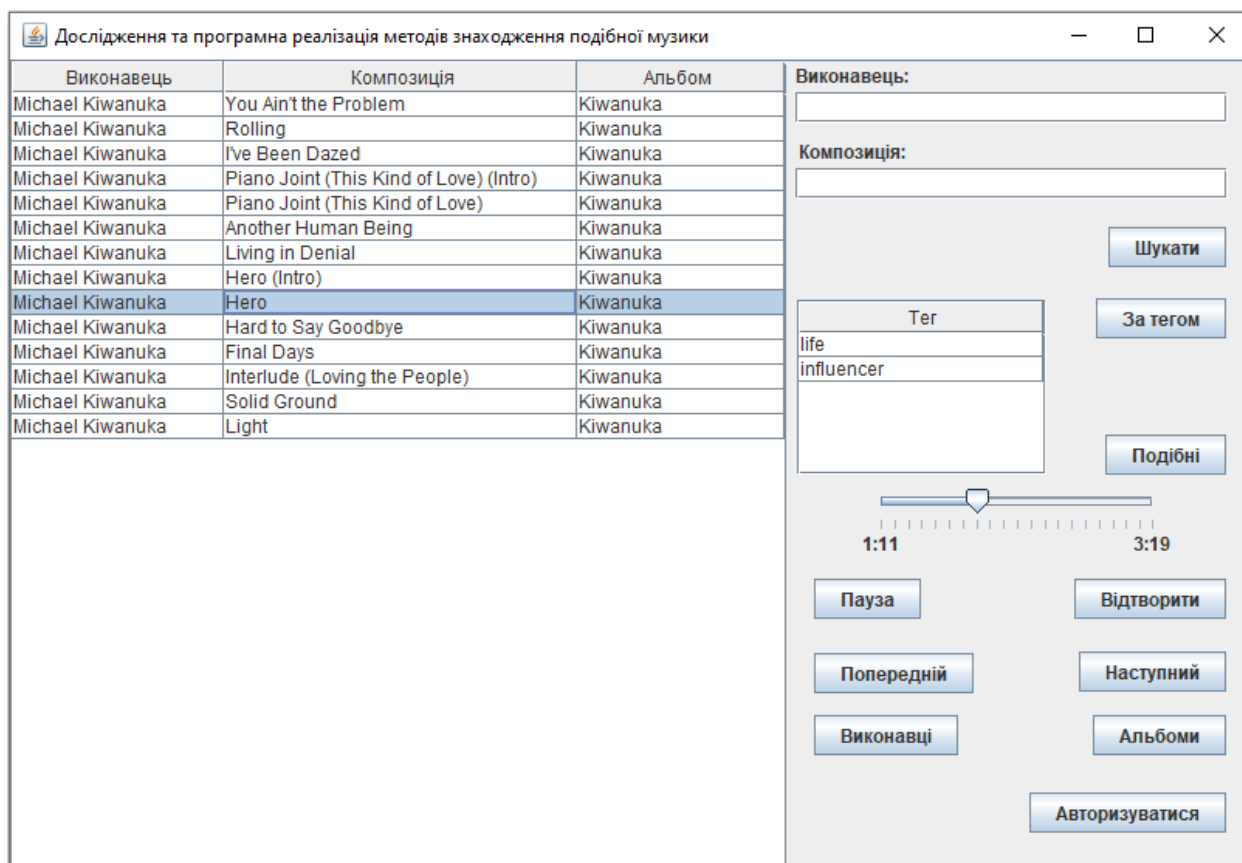


Рисунок 5.5 – Відтворення композиції

Для відображення відповідних подібних композицій потрібно спочатку обрати композицію з переліку, а далі натиснути на кнопку “Подібні”, тоді відобразиться перелік подібних композицій на вибрану. Приклад такого відображення представлено на рис. 5.6.

Під час відображення подібних композицій поля для пошуку “Виконавець” та “Композиція” заповнюються автоматично назвами виконавця та композиції, подібні на яку було відображено. Це дозволяє за необхідності повернутися до композиції.

У даному вікні можна також переглянути перелік тегів, пов’язаних з композицією.

Якщо обрати композицію, а потім обрати один з тегів, після чого натиснути на кнопку “За тегом”, то в таблиці відобразиться перелік композицій, для яких також було встановлено цей тег (рис. 5.7).

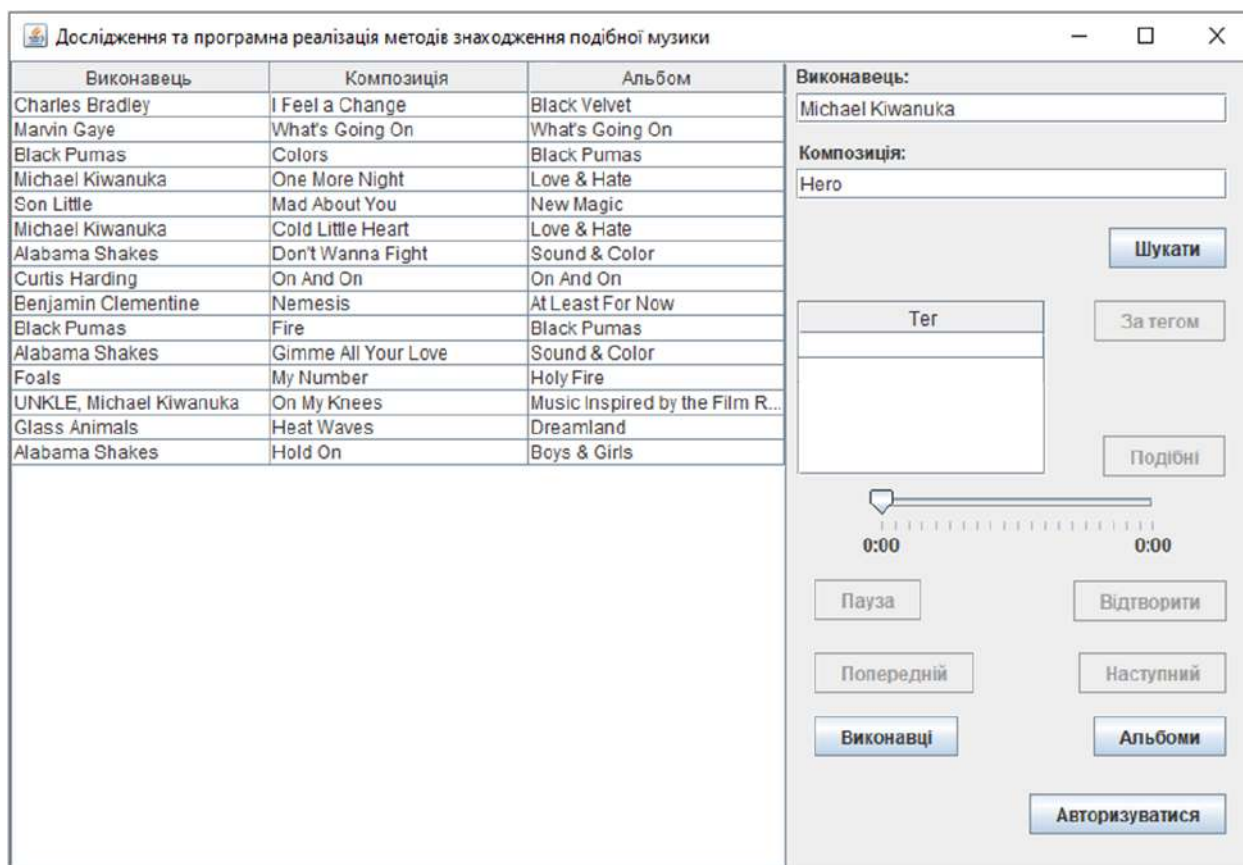


Рисунок 5.6 – Відображення подібних композицій

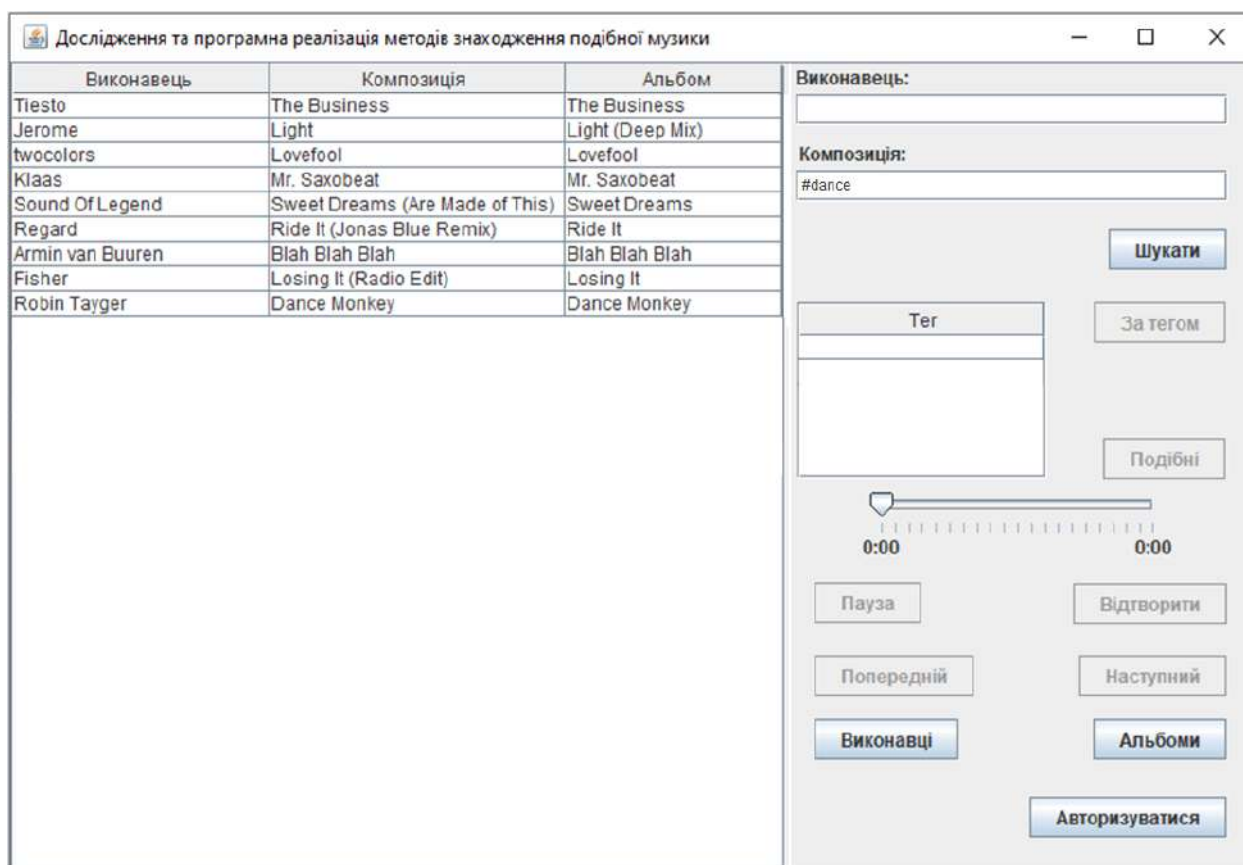


Рисунок 5.7 – Пошук композицій за тегом

Для того щоб перейти до режиму роботи адміністратора, потрібно натиснути на кнопку “Авторизуватися”, заповнити дані акаунту, після чого відкриється нове вікно. Вікно для роботи з альбомами для адміністратора представлено на рис. 5.8.

Виконавець	Альбом	Рік
Arcade Fire	Everything Now	2017
Arcade Fire	Reflektor	2013
Arcade Fire	The Suburbs	2010
Arcade Fire	Neon Bible	2007
Arcade Fire	Funeral	2004

Альбом:

Шукати

Перейти

Жанр

поп ▼

+жанр -жанр

Додати

Редагувати

Рисунок 5.8 – Робота адміністратора з альбомами

Адміністратор може прив’язувати жанри до альбому. Для цього йому потрібно обрати жанр з випадаючого переліку, а далі натиснути на кнопку “+жанр”. Для вилучення жанру з переліку потрібно обрати його з таблиці та натиснути на кнопку “–жанр” після цього.

Кнопки “Додати” і “Редагувати” дозволяють додати новий альбом (рис. 5.9) до програми або відповідно виконати редагування даних обраного з таблиці альбому.

Дослідження та програмна реалізація методів знаходження подібної ...

Виконавець: Aretha Franklin

Назва альбому:

Рік видавництва (або дата):

Жанр

соул

+жанр -жанр

Зберегти Скасувати

Рисунок 5.9 – Вікно додавання альбому

Приклад внесення даних композиції представлено на рис. 5.10. З цього вікна адміністратор може також запустити процес кластеризації. Для кожного внесення або редагування даних враховується такий випадок і сума відображається у вікні. Якщо адміністратор вважає, що змін забагато і час виконати кластеризацію заново, то він має натиснути на кнопку “Кластеризація” і ініціалізувати проведення кластеризації за наявними даними заново.

Звідси адміністратор також задає характеристики композиції:

- акустичність;
- танцювальність;
- енергійність;
- інструментальність;
- концертність;
- гучність;

- мовленнєвість;
- валентність;
- ритмічність.

Дослідження та програмна реалізація методів знаходження подібної музики

Виконавець:

Назва альбому:

Назва композиції:

Акустичність (від 0 до 1):

Танцювальність (від 0 до 1):

Енергійність (від 0 до 1):

Інструментальність (від 0 до 1):

Концертність (від 0 до 1):

Гучність (від 0 до 1):

Мовленнєвість (від 0 до 1):

Валентність (від 0 до 1):

Ритмічність (від 0 до 1):

тег

+тег -тег

Внесено або змінено записів з минулої кластеризації: 17

Кластеризація

Зберегти Скасувати

Рисунок 5.10 – Внесення композиції до програми

Окрім того можлива робота через програмний застосунок для месенджера Telegram. Для цього потрібно знайти SimilarMusicAlexBot. Приклад роботи наведено на рис. 5.11. У даному випадку за допомогою команди `getinfo` відображається інформація про музичну композицію. Необхідно ввести назву виконавця та композиції.

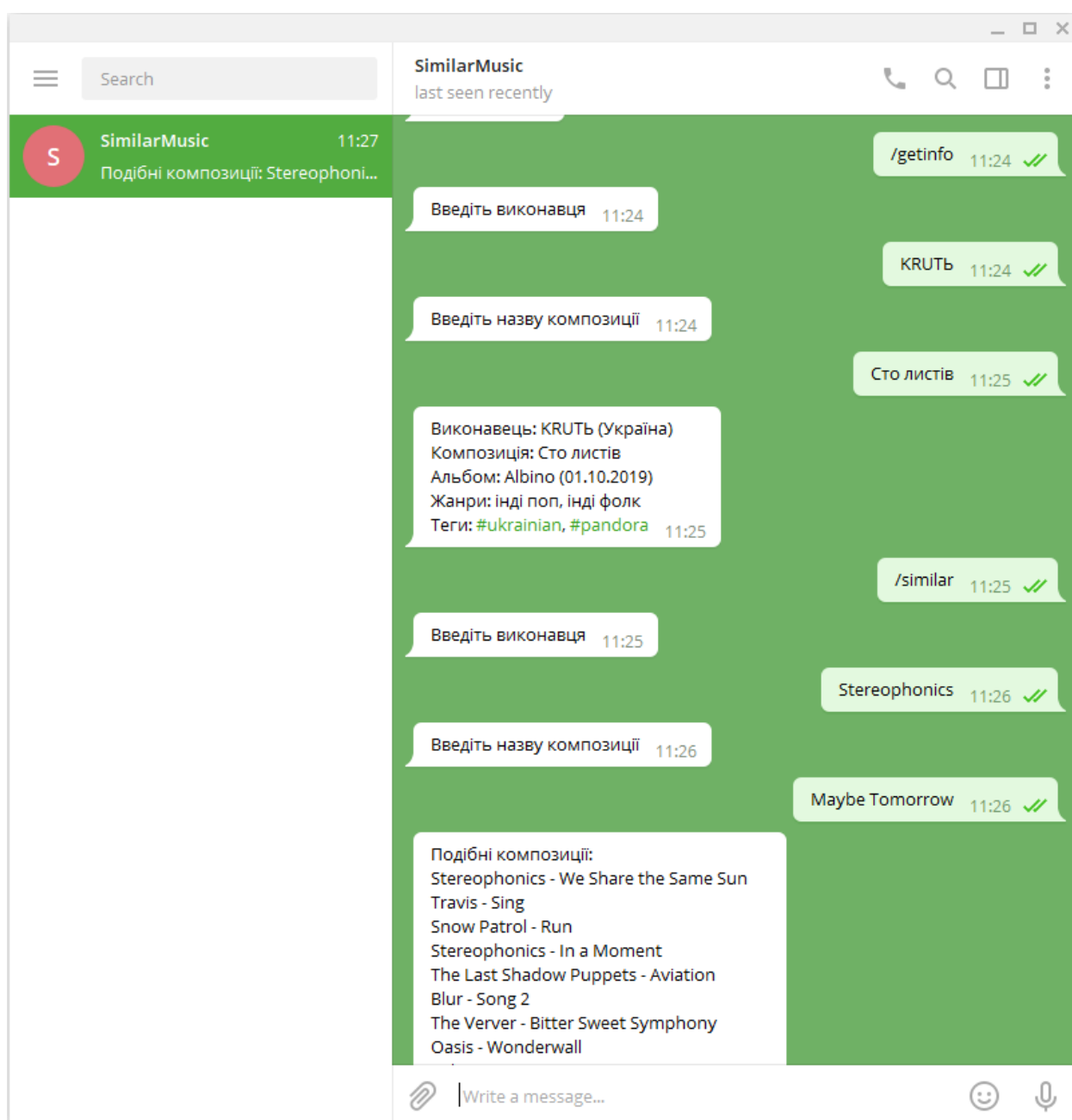


Рисунок 5.11 – Відображення даних про композицію в месенджері Telegram

За допомогою команди `similar` можна отримати перелік подібних композицій на задану (рис. 5.12). Для цього необхідно ввести назву виконавця та композиції.

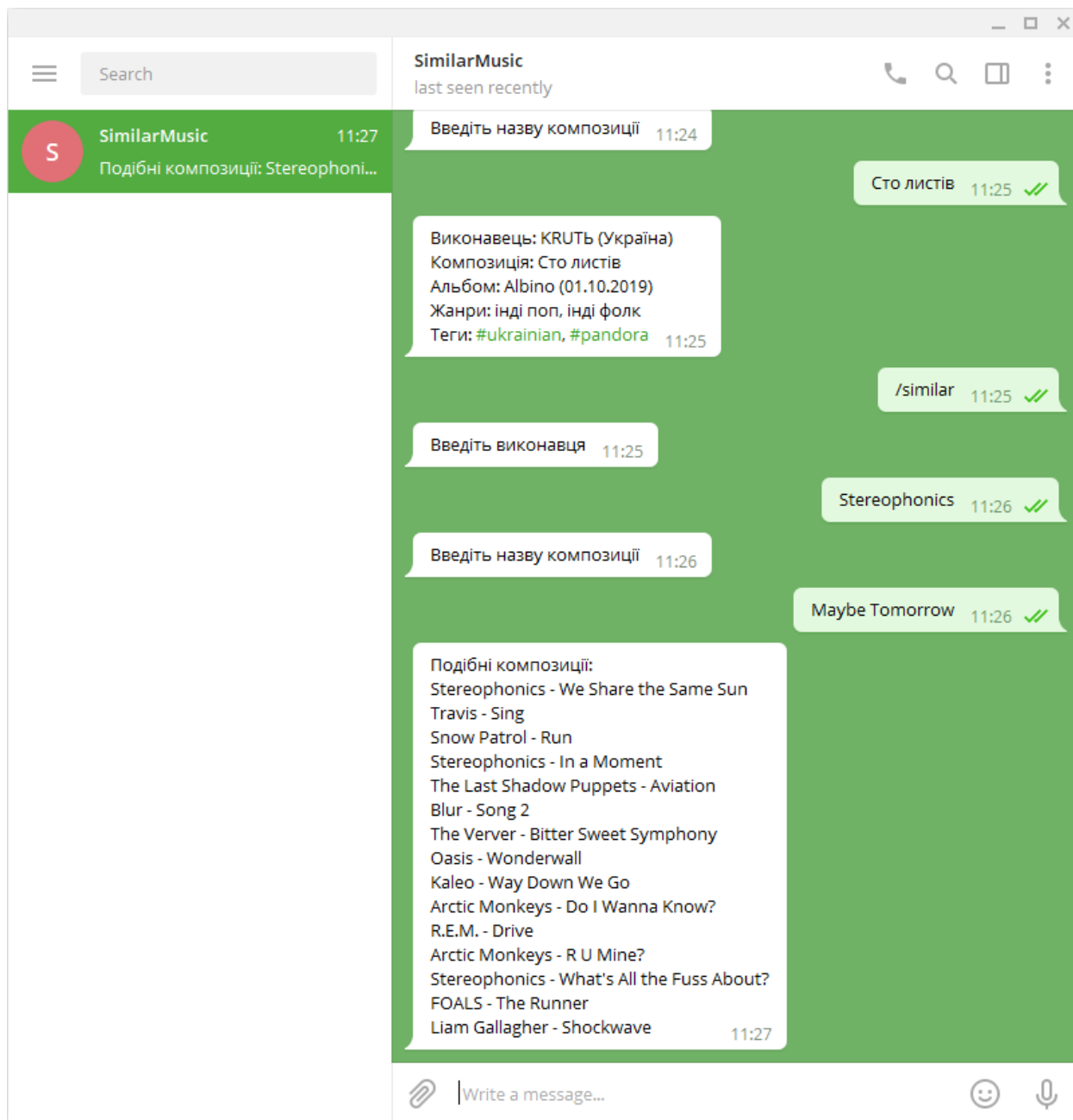


Рисунок 5.12 – Відображення подібних композицій у месенджері Telegram

6 ЕКОНОМІКО-ОРГАНІЗАЦІЙНА ЧАСТИНА

6.1 Планування розробки програмного продукту

В роботі проведено дослідження та програмну реалізацію методів знаходження подібної музики.

Програмне забезпечення призначено для автоматичного виконання підбору подібних музичних композицій, тобто розподілення їх за групами. Відповідно у музичному сервісі, призначеному для прослуховування користувачами музичних композицій, дані функції виконуються експертом для того, щоб сформувати групи композицій, які потім застосовуються в якості наступних порад для слухача. Відповідно експерт отримує зарплатню за таку роботу 8000 гривень за місяць, а використання для автоматизації даного процесу такого програмного забезпечення дозволить скоротити 0,75 його ставки.

Трудовіткість етапів виконання робіт з розробки програми зведена в табл. 6.1.

Тривалість робіт визначають за формулою (6.1):

$$T_{\text{ц}} = \frac{Q}{R}, \quad (6.1)$$

де $T_{\text{ц}}$ – тривалість циклу, днів;

Q – трудовіткість, людино-днів;

R – кількість виконавців, чол. [26].

Виходячи із завдання (час на розробку – 3 міс.) програміст працює 3 місяці із заробітною платою 10000 грн/міс., а консультант працює 1 місяць із зарплатою 8000 грн/міс.

Таблиця 6.1 – Характеристика робіт з розробки програми

Найменування роботи	Трудомісткість		Виконавці		Тривалість, днів
	людино-днів	% до підсумку	Спеціальність	Кіл., чол.	
1.Розробка та затвердження технічного завдання	8	9,8	програміст, консультант	2	4
2.Концептуальне проектування програмного забезпечення	10	12,2	програміст, консультант	2	5
3.Розробка схеми алгоритму функціонування системи	10	12,2	програміст, консультант	2	5
4.Проектування загальної структури програмного забезпечення	12	14,6	програміст	2	6
5.Проектування схеми бази даних	5	6,1	програміст, консультант	1	5
6. Розробка застосунку	19	23,2	програміст	1	19
7.Розробка користувачького інтерфейсу	10	12,2	програміст	1	10
8.Тестування системи	2	2,4	програміст, консультант	1	2
9.Розробка експлуатаційної документації	2	2,4	програміст	1	2
10.Впровадження в експлуатацію	4	4,9	програміст	1	4
Разом	82	100			62

[illegible]

6.2 Визначення витрат на розробку програми

6.2.1 Розрахунок основної заробітної плати

Витрати за цією статтею провадяться за формулою (6.2):

$$ЗП_{\text{осн}} = \sum_{i=1}^n ЗП_{\text{міс}_i} \cdot T_{\text{р}_i}, \quad (6.2)$$

де $ЗП_{\text{міс}_i}$ – місячний оклад i -го працівника, у гривнях;

$T_{\text{р}_i}$ – час, затрачуваний на розробку даної програми i -м працівником [26].

Отже:

$$ЗП_{\text{осн}} = (10000 \cdot 3) + (8000) \cdot 1 = 38000 \text{ грн.}$$

Результати розрахунків зведемо в табл. 6.3.

Таблиця 6.3 – Розрахунок основної заробітної плати

Посада виконавця	Чисель- ність, чол.	Місячний оклад, грн.	Кількість місяців роботи	Премії та доплати, грн.	Середня місячна заробітна платня, грн	Сума зарплати, грн.
Програміст	1	10000	3	—	10000	30000
Консультант	1	8000	1	—	8000	8000
Разом						38000

Додаткова заробітна плата визначається на основі формули (6.3):

$$ЗП_{\text{дод}} = ЗП_{\text{осн}} \cdot 0,1 \text{ [26]}. \quad (6.3)$$

Виходить:

$$ЗП_{\text{дод}} = 38000 \cdot 0,1 = 3800 \text{ грн.}$$

6.2.2 Відрахування на соціальне страхування

Відрахування на єдиний соціальний внесок розраховуються на основі 22 % за допомогою формули (6.4):

$$B = (ЗП_{\text{осн}} + ЗП_{\text{дод}}) \cdot 0,22 \text{ [26]}. \quad (6.4)$$

Виходить:

$$B = (38000 + 3800) \cdot 0,22 = 9196 \text{ грн.}$$

6.2.3 Визначення витрат на матеріали

Для розробки програми знадобилися наступні витратні матеріали:

– компакт-диск (1 шт) – 25 грн;

– папір – 480 грн.

Разом: 505 грн.

Витрати на матеріали розраховують за формулою (6.5):

$$M = \sum_{i=1}^n (Ц_i \cdot N_i \cdot (1 + K_{\text{т.з.}}) - Ц_{\text{із}} \cdot N_{\text{із}}), \quad (6.5)$$

де M – витрати на матеріали, покупні напівфабрикати та комплектуючі вироби, грн.;

$K_{\text{т.з.}}$ – коефіцієнт, що враховує транспортно-заготівельні витрати;

$Ц_i$ – ціна i -го найменування матеріалу, напівфабрикату й комплектуючого, грн.;

N_i – потреба в i -ому матеріалі, напівфабрикаті й комплектуючому;

U_{iz} – ціна зворотних відходів i -го найменування матеріалу, грн;

N_{iz} – кількість зворотних відходів i -го найменування;

n – кількість найменувань матеріалів, напівфабрикатів і комплектуючих [26].

Виходить:

$$U_{iz} = 0, N_{iz} = 0, K_{тз} = 0,05,$$

$$M = 505 \cdot (1 + 0,05) = 530,25 \text{ грн.}$$

Разом, витрати на матеріали становлять 530,25 грн.

6.2.4 Витрати на спеціальне устаткування

Амортизаційні відрахування визначають за формулою (6.6):

$$A = \Phi_6 \cdot \frac{H_a}{100}, \quad (6.6)$$

де Φ_6 – балансова вартість обчислювальної техніки, грн.;

H_a – норма амортизаційних відрахувань на повне відновлення обчислювальної техніки, % [26].

Виходить:

$$\Phi_6 = 7200 \cdot 1,05 = 7560 \text{ грн.}, H_a = 25\%,$$

$$A = 7560 \cdot 0,25 = 1890 \text{ грн.}$$

Статтю «Експлуатація устаткування» розраховують за допомогою формули (6.7):

$$E_{об} = C_e + M \quad [26]. \quad (6.7)$$

Витрати на електроенергію визначають за формулою (6.8):

$$C_e = N_n \cdot \Phi_{ef} \cdot K_{зч} \cdot K_{зп} \cdot C_e, \quad (6.8)$$

де N_n – номінальна потужність електронно-обчислювальної машини (ЕОМ), кВт;

Φ_{ef} – річний ефективний фонд часу роботи ЕОМ, машино-год.;

$K_{зч}$ – середній коефіцієнт завантаження за часом;

$K_{зп}$ – коефіцієнт завантаження програмного забезпечення потужності;

C_e – ціна однієї кВт·год. електроенергії, грн./(кВт·год.) [26].

Виходить:

$$C_e = 0,2 \cdot 1800 \cdot 0,75 \cdot 0,7 \cdot 2,11 = 400,30 \text{ грн.}$$

Витрати на допоміжні матеріали становлять 49,70 грн.

Витрати за формулою (6.7) становлять:

$$E_{об} = 400,30 + 49,70 = 450 \text{ грн.}$$

Витрати за статтею «Заробітна плата обслуговуючих робітників і відрахування на єдиний соціальний внесок» обчислюються за формулою (6.9):

$$ЗП_{обсл} = \Phi ЗП_p \cdot (1 + K_{відрах}) \cdot \frac{t_{обсл}}{\Phi_{ef.обсл}}, \quad (6.9)$$

де $\Phi ЗП_p$ – річний фонд заробітної плати обслуговуючих робітників, грн.;

$K_{\text{відрах}}$ – коефіцієнт, що враховує відрахування на соціальне страхування та в інші фонди;

$t_{\text{обсл}}$ – час протягом року, необхідний на технічне обслуговування ЕОМ, год./рік;

$\Phi_{\text{еф.обсл}}$ – річний ефективний фонд часу роботи обслуговуючого персоналу, год./рік [26].

Машину обслуговує один електронщик з основною заробітною платою 4800 грн./мес (10% – премія):

$$\Phi ЗП_p = (4800 + 4800 \cdot 0,1) \cdot 12 = 63360 \text{ грн.}$$

Виходить:

$$ЗП_{\text{обсл}} = 63360 \cdot (1 + 0,22) \cdot 18 / 1750 = 795,08 \text{ грн.}$$

Сума витрат за статтею «Поточний ремонт устаткування» укрупнено дорівнює 3% від балансової вартості устаткування (7560 грн.) і становить 226,8 грн. Інші витрати становлять 3% від суми всіх попередніх статей витрат на утримання та експлуатацію устаткування – 66,56 грн.

Зведемо отримані дані в табл. 6.4.

Витрати на оплату машинного часу ЕОМ для налагодження даної програми визначаються за формулою (6.10):

$$C_{\text{мо}} = P_{\text{екс}} \cdot t_{\text{мо}}, \quad (6.10)$$

де $C_{\text{мо}}$ – витрати на оплату машинного часу, грн.;

$P_{\text{екс}}$ – експлуатаційні витрати на одну годину машинного часу цієї цифрової ЕОМ, грн./ машино-год.;

$t_{\text{мо}}$ – машинний час цифрової ЕОМ для написання та налагодження даного програмного продукту, машино-год. [26].

Таблиця 6.4 – Кошторис витрат на утримання та експлуатацію устаткування

Найменування статей витрат	Сума, грн.
Амортизація устаткування	1890,00
Експлуатація устаткування (крім витрат на поточний ремонт)	450,99
Заробітна плата основних і додаткова обслуговуючих робітників з відрахуваннями на єдиний соціальний внесок	795,08
Поточний ремонт устаткування	226,8
Інші витрати	100,89
Разом	3463,76

Експлуатаційні витрати на одну годину машинного часу рівні:

$$P_{\text{екс}} = 3463,76 / 1800 = 1,92 \text{ грн.}$$

ЕОМ експлуатується 62 дні в одну зміну, що становить у сумі

$$t_{\text{мо}} = 8 \cdot 62 = 496 \text{ годин.}$$

Отже, витрати на оплату машинного часу рівні:

$$C_{\text{мо}} = 1,92 \cdot 496 = 954,46 \text{ грн.}$$

6.2.5 Накладні витрати

Накладні витрати становлять 40% , що дорівнює:

$$H_{\text{в}} = 38000 \cdot 0,4 = 15200 \text{ грн.}$$

6.2.6 Плановий прибуток

Результати визначення витрат зведені в табл. 6.5.

Таблиця 6.5 – Калькуляція кошторисної вартості робіт з розробки

Статті витрат	Сума, грн.	Питома вага, %
Основна заробітна плата	38000,00	56,15
Додаткова заробітна плата	3800,00	5,61
Єдиний соціальний внесок	9196,00	13,58
Матеріали та комплектуючі вироби	530,25	0,78
Витрати на оплату машинного часу	954,46	1,41
Накладні витрати	15200,00	22,46
Разом	67680,71	100,00

6.3 Розрахунок капітальних вкладень

Розмір капітальних вкладень споживача програми K , пов'язаних з її розробкою та впровадженням, визначають за формулою (6.11):

$$K = K_{\text{пв}} + K_{\text{вф}}, \quad (6.11)$$

де $K_{\text{пв}}$ – передвиробничі витрати, грн.;

$K_{\text{вф}}$ – капітальні вкладення у виробничі фонди, необхідні для впровадження програми, грн. [26].

Передвиробничі витрати визначають за формулою (6.12):

$$K_{\text{пп}} = \frac{S_{\text{рп}}}{n} + K_{\text{ін.пв}}, \quad (6.12)$$

де $S_{\text{рп}}$ – сумарні витрати на розробку або придбання програмного продукту, грн.;

n – кількість споживачів даної програми;

$K_{\text{ін.пв}}$ – інші передвиробничі витрати, грн. [26].

Обчислюємо:

$$K_{\text{пп}} = 67680,71 / 1 + 0 = 67680,71 \text{ грн.}$$

Капітальні вкладення у виробничі фонди включають витрати на придбання, установку, монтаж і налагодження комплексу технічних засобів (КТЗ) визначають за формулою (6.13):

$$K_{\text{пф}} = \frac{K_{\text{КТЗ}} \cdot T_{\text{КТЗ}}}{\Phi_{\text{еф.КТЗ}}}, \quad (6.13)$$

де $K_{\text{КТЗ}}$ – витрати на придбання, установку, монтаж і налагодження КТЗ (7560 грн.), грн.;

$T_{\text{КТЗ}}$ – машинний час КТЗ, потрібний даному споживачеві програми для завдань, розв'язуваних за допомогою даної програми, машино-год./рік;

$\Phi_{\text{еф.КТЗ}}$ – річний ефективний фонд часу роботи КТЗ, машино-год./рік [26].

Виходить:

$$K_{\text{пф}} = \frac{7560 \cdot 450}{1800} = 1890,00 \text{ грн.}$$

$$K = 67680,71 + 1890 = 69570,71 \text{ грн.}$$

6.4 Розрахунок експлуатаційних витрат

Витрати, пов'язані з експлуатацією програми впродовж року, визначаємо за формулою (6.14):

$$И = P_{\text{екс.ктз}} \cdot T_{\text{ктз}} + \frac{S_{\text{рп}}}{n \cdot T_{\text{с}}}, \quad (6.14)$$

де $P_{\text{екс.ктз}}$ – експлуатаційні витрати, що доводяться на 1 годину машинного часу КТЗ споживача програмного продукту, грн./машино-год.;

$T_{\text{ктз}}$ – сумарний машинний час КТЗ протягом року, потрібний споживачеві програми для завдань, що розв'язуються з її допомогою, машино-год.;

$S_{\text{рп}}$ – сумарні витрати на розробку програми, грн.;

n – кількість споживачів даної програми;

$T_{\text{с}}$ – термін служби даної програми до її морального зношування [26].

Отже:

$$И = 1,92 \cdot 450 + 67680,71/(1 \cdot 5) = 14400,14 \text{ грн.}$$

6.5 Визначення економічної ефективності програмного продукту

Річна економія від впровадження програмного забезпечення становить різницю між економією на зарплату консультанта (відповідно експерт отримує зарплатню за таку роботу 8000 гривень за місяць, а використання для автоматизації даного процесу такого програмного

забезпечення дозволить скоротити 0,75 його ставки) і витратами на експлуатацію.

Тоді річна економія системи може бути визначена за формулою (6.15):

$$E = E_{zn} - I. \quad (6.15)$$

Розрахуємо:

$$E = 12 \cdot 8000 \cdot 1,1 \cdot 1,22 \cdot 0,75 - 14400,14 = 96624 - 14400,14 = 82223,86 \text{ грн.}$$

Коефіцієнт ефективності капіталовкладень розраховується за формулою (6.16):

$$Ef = E / K [26]. \quad (6.16)$$

Тоді:

$$Ef = 82223,86 / 69570,71 = 1,18.$$

Строк окупності капітальних вкладень розраховується за формулою (6.17):

$$T_{ок} = K / E [26]. \quad (6.17)$$

Тоді:

$$T_{ок} = 69570,71 / 82223,86 = 0,85 \text{ роки.}$$

6.6 Техніко-економічні показники

Техніко-економічні показники, розраховані для даного проєкту, зведені в табл. 6.6.

Відповідно до розрахунків, у результаті впровадження програмного забезпечення вдасться одержати річну економію в розмірі 82223,86 грн.

Строк окупності даної системи становить 0,85 роки, що менше нормативного, який складає для соціальної сфери 6,67 року.

Таким чином, впровадження програмного забезпечення економічно доцільно.

Таблиця 6.6 – Техніко-економічні показники

Показники	Одиниця виміру	Варіанти	
		базовий	новий
Витрати на розробку програми	грн.		67680,71
Капітальні вкладення	грн.		69570,71
Експлуатаційні витрати	грн.	96624	14400,14
Річна економія	грн.		82223,86
Строк окупності програми	років		0,85
Тривалість розрахунків		більше	менше
Повнота контролю		гірше	краще
Вірогідність розрахункової інформації		менше	більше

7 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

7.1 Аналіз потенційних небезпек

В магістерській роботі проводиться дослідження та програмна реалізація методів знаходження подібної музики.

При роботі постійно або періодично діють наступні небезпечні і шкідливі чинники:

- можливість ураження електрострумом внаслідок небезпечного рівня напруги в електричному ланцюзі, тому у приміщенні можуть виникнути аварійні ситуації: замикання, загоряння проводки, поразка електрострумом, що може привести до електротравм або летальних наслідків;

- невідповідність ергономічних характеристик устаткування нормованим величинам внаслідок нераціональної організації робочого місця;

- невідповідність нормам параметрів мікроклімату;

- неправильні дії персоналу під час надзвичайних ситуацій внаслідок відсутності управління та паніки, що може призвести до травм та летальних наслідків [27].

7.2 Заходи по забезпеченню безпеки

Забезпечення безпеки праці, організації оптимальних умов праці, а також захисту навколишнього середовища досягається шляхом дотримання норм і проведення заходів, передбачених законодавчими актами України.

Згідно правил улаштування електроустановок приміщення, в якому проводиться дослідження методів та розробка програмних засобів синтаксичного та інтелектуального аналізу веб-сторінок відноситься до «Приміщення без підвищеною небезпеки», у приміщенні повинні бути відсутні відкриті струмоведучі частини з напругою більше 12 В, при виконанні правил улаштування електроустановок враження електричним

струмом можливо тільки у випадку несправності апаратури й живильних кабелів [27].

Відповідно до правил улаштування електроустановок захисне заземлення застосовується у всіх електроустановках змінного струму під напругою 380В і вище й постійного струму 440 В і вище, захисному заземленню підлягають корпуси трансформаторів, оболонки кабелів, металеві огороження електроустановок відповідно до ГОСТ 12.1.030-81 «Електробезпека. Захисне заземлення. Занулення» [28].

Подвійна ізоляція забезпечує застосування крім основної, робочої ізоляції струмоведучих частин ще одного шару додаткової ізоляції, що ізолює людину від металевих неструмоведучих частин, які можуть випадково виявитися під напругою, при застосуванні електроустаткування з подвійною ізоляцією не потрібно ні заземлення, ні занулення їхніх корпусів [29].

В якості заходів захисту запропоновано:

- кілька разів на протязі робочого дня мити руки і обличчя водою, а після закінчення роботи вимити руки й лице з милом;
- щоденно в приміщенні з персональними комп'ютерами проводити вологе прибирання [30].

7.3 Заходи по забезпеченню промислової санітарії та гігієни праці

У розділі 4 Державних санітарних норм і правил 3.3.2.007-98 «Гігієнічних вимог до організації роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» передбачені гігієнічні вимоги до організації устаткування робочих місць візуальними дисплейними терміналами (ВДТ) ЕОМ і персональних ЕОМ:

- конструкція робочого стола повинна відповідати сучасним вимогам ергономіки й забезпечувати оптимальне розміщення на робочій поверхні використовуваного устаткування (дисплея, клавіатури, принтера) і документів;

– щоб уникнути відблисків на екрані вікон доцільно розмістити робоче місце торцем до вікна не ближче чим на 1 м, площа дозволяє нам розмістити додаткові меблі для зберігання документів, магнітних дисків, полками, стелажми, тумбами й т.п. [30].

Передбачено такі внутрішньозмінні режими праці та відпочинку при роботі з ЕОМ при 8-годинній денній робочій зміні в залежності від характеру праці:

– для розробників програм із застосуванням ЕОМ слід призначати регламентовану перерву для відпочинку тривалістю 15 хвилин через кожен годину роботи за ВДТ;

– для операторів із застосування ЕОМ слід призначати регламентовані перерви для відпочинку тривалістю 15 хвилин через кожні дві години;

– для операторів комп'ютерного набору слід призначати регламентовані перерви для відпочинку тривалістю 10 хвилин після кожною години роботи за ВДТ [29].

У всіх випадках, коли виробничі обставини не дозволяють застосувати регламентовані перерви, тривалість безперервної роботи з ВДТ не повинна перевищувати 4 години:

– при 12-годинній робочій зміні регламентовані перерви повинні встановлюватися в перші 8 годин роботи аналогічно перервам при 8-годинній робочій зміні, а протягом останніх 4-х годин роботи, незалежно від характеру трудової діяльності, через кожен годину тривалістю 15 хвилин;

– в окремих випадках – при хронічних скаргах працюючих з ВДТ ЕОМ і персональних ЕОМ на зорове втомлення, незважаючи на дотримання санітарно-гігієнічних вимог до режимів праці і відпочинку, а також застосування засобів локального захисту очей – допускаються індивідуальних підхід до обмеження часу робіт з ВДТ, зміни характеру праці, чергування з іншими видами діяльності, не пов'язаними з ВДТ [28].

Також виділено наступні вимоги:

– активний відпочинок має полягати у виконанні комплексу гімнастичних вправ, спрямованих на зняття нервового напруження, м'язове розслаблення, відновлення функцій фізіологічних систем, що порушуються протягом трудового процесу, зняття втоми очей, поліпшення мозкового кровообігу і працездатності;

– всі норми по освітленості беруться згідно Державних будівельних норм В.2.5-28-2006 «Природне та штучне освітлення», штучне освітлення в приміщеннях здійснюється системою загального рівномірного освітлення, при цьому штучне освітлення застосовують не тільки в темний, але й у світлий час доби;

– найбільш прийнятними для приміщень є люмінесцентні лампи ЛБ (білого світла) потужністю 20, 40 або 80 Вт, які встановлюються в світильники типу ЛСП, допускається застосування ламп накаливання у світильниках місцевого освітлення;

– загальне освітлення варто виконувати у виді суцільних чи переривчастих ліній світильників, розташованих збоку від робочих місць [29].

Зниження рівня шуму в приміщенні можна здійснити за Державними санітарними нормами 3.3.6-037-99 «Санітарні норми виробничого шуму, ультразвуку та інфразвуку» наступним чином:

– використанням блоків живлення персональних комп'ютерів з вентиляторами на гумових підвісках;

– застосуванням принтерів колективного користування, розташованих на значній відстані від більшості робочих місць користувачів персональних комп'ютерів;

– зменшенням шуму на шляху його поширення через розміщення звукоізолюючого відгородження у вигляді стін, перетинок, кабін;

– акустичною обробкою приміщень – зменшення енергії відбитих звукових хвиль шляхом збільшення площі звукопоглинання (розміщення на

поверхнях приміщення облицювань, що поглинають звук, розташування в приміщеннях штучних поглиначів звуку) [30].

7.4 Заходи по забезпеченню пожежної безпеки

В приміщенні обчислювального центру (ОЦ) джерелом загоряння можуть бути електронні схеми ЕОМ, прилади для технічного обслуговування, електроустаткування, кондиціонери повітря, де за рахунок різних порушень відбувається перегрів тих чи інших елементів, електричні іскри, дуга, що можуть визвати загоряння горючих матеріалів, дерев'яні та пластикові стільці та столи, папір, перевантаження мережі живлення, людські чинники тощо [29].

Клас пожежі для приміщень ОЦ встановлюється згідно Державних будівельних норм В.1.1.7-2002 «Захист від пожежі. Пожежна безпека об'єктів будівництва» – пожежі твердих речовин, переважно органічного походження, горіння яких супроводжується тлінням (деревина, пластмаси, папір) – і визначається як клас А:

- для відводу надлишкового тепла від ЕОМ задіяні кондиціонування повітря і система вентиляції, що обладнана пристроєм, який забезпечує автоматичне відключення вентиляції, в разі виникнення пожежі;

- напруга до електроустановок подається по кабельним мережам, що прокладені під технологічною з'ємною підлогою з рівнем вогнестійкості не менше 0,5 год.;

- до первинних засобів тушіння пожеж, призначених для ліквідації невеликих загорянь використовуються пожежні стволи, внутрішні пожежні водопроводи, вогнегасники, згідно Нормативно-правових актів з пожежної безпеки Б.03.001-2004 «Типові норми належності вогнегасників» при захисті від пожежі приміщення з персональними ЕОМ використовуються вуглекислотні вогнегасники ВВК-1,4, ВВК-2 або аерозольні водопінні вогнегасники ВВПА-400;

– згідно з Нормативно-правовими актами з пожежної безпеки Б.03.001-2004 приміщення з персональними ЕОМ необхідно оснащувати переносними вуглекислотними вогнегасниками з розрахунку один вогнегасник ВВК-1,4 або ВВК-2 або ВВПА-400 на три персональні ЕОМ, але не менше ніж один вогнегасник, перелічених типів на приміщення;

– важливу роль у забезпеченні безпечного виходу людей грає протидимовий захист евакуаційних ходів, у будинках висотою до 9 поверхів незадимлюваність сходових кліток на час евакуації досягається їхньою ізоляцією від підвалів, горищ і поверхів, для цього влаштовані відособлені входи в підвали, вхід на сходову клітку з поверхів здійснюють через тамбур - шлюз із підпором повітря, відокремлюють горища від сходових кліток перекриттями з негорючих матеріалів [31].

Розрахуємо стаціонарну установку для гасіння пожежі вуглекислотою у виробничому приміщенні, якщо:

- об'єм приміщення $W_{\text{п}} = 550 \text{ м}^3$;
- коефіцієнт $K_{\text{в}} = 1,5$, який враховує особливості процесу газообміну, витік вуглекислоти через нещільність та пройоми приміщення;
- довжина трубопроводу від установки до місця гасіння загорання $L = 80 \text{ м}$.

Необхідна маса вогнегасної газової суміші визначається за рівнянням (7.1):

$$G_{\text{г}} = G_{\text{в}} \cdot W_{\text{п}} \cdot K_{\text{в}} + G_0, \quad (7.1)$$

де $G_{\text{в}}$ – вогнегасна концентрація газового складу для вуглекислоти; $G_{\text{в}} = 0,7 \text{ кг/м}^3$;

G_0 – кількість вуглекислоти, що залишається в установці після закінчення її роботи, $G_0 = 0,2 \cdot G_{\text{г}}$, кг.

Відповідно:

$$G_{\Gamma} = G_B \cdot W_{\Pi} \cdot K_B + 0,2 \cdot G_{\Gamma},$$

$$0,8 G_{\Gamma} = G_B \cdot W_{\Pi} \cdot K_B,$$

$$G_{\Gamma} = (G_B \cdot W_{\Pi} \cdot K_B) / 0,8 = (0,7 + 550 + 1,5) / 0,8 = 722 \text{ кг.}$$

Розрахуємо необхідну кількість робочих балонів N_6 із вуглекислотою в установці за рівнянням (7.2):

$$N_6 = G_{\Gamma} / V_6 \cdot \rho \cdot \alpha_H, \text{ шт.}, \quad (7.2)$$

де V_6 – об'єм балону (л), приймаємо $V_6 = 25$ л, при цьому в балоні міститься 15,6 кг вуглекислоти;

ρ – питома вага вуглекислоти, $\rho = 0,625$ кг/л;

α_H – коефіцієнт наповнення, $\alpha_H = 0,85$.

Розрахуємо:

$$N_6 = 722 / (25 \cdot 0,625 \cdot 0,85) = 54,36 \approx 55 \text{ шт.}$$

Розрахуємо пропускну здатність трубопроводу за формулою (7.3):

$$F = 10^{-4} \sqrt{\frac{P \cdot \gamma}{2 \cdot A \cdot L}} \text{ (кг/с)}, \quad (7.3)$$

де P – питомий тиск вуглекислоти на початку трубопроводу (в балоні), Н/м², $49 \cdot 10^5$ Н/м²;

A – питомий опір трубопроводу при діаметрі труби 40 мм, ($A = 0,035$);

γ – щільність вуглекислоти на початку трубопроводу (в балоні), $\gamma = 2900$ кг/м³.

Розрахуємо:

$$F = 10^{-4} \sqrt{\frac{P \cdot \gamma}{2 \cdot A \cdot L}} = 10^{-4} \cdot \sqrt{\frac{4,9 \cdot 106 \cdot 2900}{2 \cdot 0,035 \cdot 80}} = 5,03 \text{ кг/с.}$$

Таким чином, необхідна маса вуглекислоти $G_T = 722$ кг, кількість робочих балонів $N_6 = 55$ шт., пропускна здатність трубопроводу $F = 5,03$ кг/с.

7.5 Заходи по забезпеченню безпеки у надзвичайних ситуаціях

Дії формування цивільної оборони в осередках ураження полягають в наступному:

- зведені команди можуть діяти самостійно, при виконанні задач вони часто підсилюються підрозділам механізації робіт та служб цивільної оборони;

- послідовність проведення залежить від характеру руйнувань будинків і споруджень, рівнів радіоактивного забруднення, інтенсивності пожеж та інших умов, що впливають на дії, у першу чергу розчищаються проїзди і проходи до захисних споруджень, де можуть знаходитись постраждали, до місць аварії, що перешкоджають або затрудняють проведення рятувальних робіт;

- аварійно-технічні формування роблять відвід води у канави, каналізаційні колодязі й у знижені місця, при ліквідації на водопровідних і теплових мережах відключаються зруйновані ділянки, визначаються місця аварій і проводяться ліквідаційні роботи [31]-[32].

Окрім того виділено для виконання такі рішення, необхідні для вирішення проблеми:

- для запобігання вибухів і пожеж при проведенні аварійних робіт на зруйнованих трубопроводах заповнених сильнодіючими отруйними і пальними речовинами у першу чергу перекриваються трубопроводи, з'єднуючі резервуари з технологічними агрегатами, відключаються всі

насоси, що підтримують тиск у магістралях, і після цього лишаються ліквідуються ушкодження;

– витягаючи уражених з під завалу чи з під окремих уламків, варто уникнути зрушень елементів завалу і нанесення ураженому додаткових травм, звільняючи в першу чергу голову і верхню частину тулуба, після витягу ураженого йому надається перша медична допомога, а при можливості ця допомога надається і до витягу з під завалу [32].

Перша медична допомога ураженим надається у порядку самої взаємодопомоги, а також особовим складом медичних формувань [33].

ВИСНОВКИ

Розглянуто основні принципи організації сучасних музичних сервісів, виконано огляд особливостей сучасних музичних сервісів, серед яких розглянуто Spotify, Deezer, Choosic, Music-Map. Виконано аналіз методів, які можуть використовуватися для знаходження подібної музики на даний момент.

Виконано проєктування програмного забезпечення: визначено функціональні вимоги до програми, прийнято рішення щодо вибору в якості мови програмування для реалізації програми мови Java, виконано проєктування бази даних на основі використання MySQL в якості системи управління базами даних та визначено структуру програмного забезпечення з побудовою відповідної структурної схеми.

Розроблено програмне забезпечення, що складається з настільного програмного застосунку для операційної системи Windows, який використовується адміністратором для внесення даних та ініціалізації роботи методу, та програмного застосунку для месенджера Telegram, який використовується безпосередньо користувачами для пошуку подібної музики. Описано рішення, які було прийнято під час реалізації програмного забезпечення.

Наукова новизна роботи в розробленому методі знаходження подібної музики, який використовує метод кластеризації на основі суміші гаусівських розподілів, визначаючи в якості набору ознак музичної композиції показники акустичності, танцювальності, енергійності, інструментальності, концертності, гучності, мовленнєвості, валентності, ритмічності, що дозволяє виконувати пошук композицій без використання характеристик сигналу з визначенням подібності за кластерами.

Усі завдання на дипломну кваліфікаційну роботу магістра виконані повністю.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Снежинская, М. Г. Музыкальная индустрия как социокультурный феномен [Текст] : диссертация на соискание ученой степени кандидата социологических наук по специальности 22.00.06 «Социология культуры» / М. Г. Снежинская. – М. : РГГУ, 2019. – 226 с.
2. Кузнецов, А. Что такое стриминг и как на этом зарабатывают? [Электрон. ресурс] / А. Кузнецов. – Режим доступа : <https://rb.ru/story/chto-takoe-strim/>
3. Что такое стриминговый сервис [Электрон. ресурс]. – Режим доступа : <http://ipkey.com.ua/faq/932-streaming-service.html>
4. Блудов, С. Музыкальный стриминг [Электрон. ресурс] : тренды 2020 / С. Блудов. – Режим доступа : <https://www.dataart.ru/articles/muzykalnyi-striming-trendy-2020/>
5. Печищева, Е.М. Стриминговые сервисы в музыкальной индустрии [Текст] / Е.М. Печищева // Научно-практический электронный журнал Аллея Науки. – 2018. – № 4 (20), Том 5. – С. 161-165.
6. Стриминговые сервисы доминируют на рынке музыки в США [Электрон. ресурс]. – Режим доступа : <https://shazoo.ru/2020/01/02/88964/streamingovyie-servisyy-dominiruyut-na-rynke-muzyki-v-ssha>
7. Семенченко, Б. Как стриминг позволяет музыкантам вновь зарабатывать на продаже музыки [Электрон. ресурс] / Б. Семенченко. – Режим доступа : https://liroom.com.ua/articles/streaming_future/
8. Музыкальные стриминг-сервисы – сравнительный рейтинг лучших, июль 2020 [Электрон. ресурс]. – Режим доступа : <https://ax.digital/music-streaming-tools/>
9. Топ-10 лучших музыкальных приложений [Электрон. ресурс]. – Режим доступа : <https://www.moyo.ua/news/top->

10_luchshih_muzykalnyh_prilozheniyi_gde_slushat_muzyku_onlayin_posle_blokirovki_vk.html

10. Мураховский, А. Choosic для iOS – поисковик новой музыки с учётом ваших предпочтений [Электрон. ресурс] / А. Мураховский. – Режим доступа : <https://lifehacker.ru/2015/04/26/choosic/>

11. Горчаков, Д. 5 сервисов для поиска новой музыки [Электрон. ресурс] / Д. Горчаков. – Режим доступа : <https://lifehacker.ru/5-servisov-dlja-poiska-novoj-muzyki/>

12. McFee, B. More like this [Text] : machine learning approaches to music similarity : A dissertation submitted in partial satisfaction of the requirements for the degree Doctor of Philosophy in Computer Science / B. McFee. – San Diego : University of California, 2012. – 166 p.

13. Обзор алгоритмов кластеризации данных [Электрон. ресурс]. – Режим доступа : <https://habr.com/ru/post/101338/>

14. Яшина, Е.С. Использование метода кластеризации в информационно-аналитической системе [Текст] / Е.С. Яшина, М.А. Щербак // Рідаоелектронні і комп'ютерні системи. – 2016. – № 2 (76). – С. 62-69.

15. Алгоритмы кластеризации на службе Data Mining [Электрон. ресурс]. – Режим доступа : <https://loginom.ru/blog/data-mining-clustering>

16. Жерон, О. Прикладное машинное обучение с помощью Scikit-Learn, Keras и TensorFlow [Текст] : концепции, инструменты и техники для создания интеллектуальных систем / О. Жерон. – М. : Вильямс, 2018. – 688 с.

17. Методы кластеризации [Электрон. ресурс]. – Режим доступа : <http://pzs.dstu.dp.ua/DataMining/cluster/index.html>

18. Орешков, В. EM – масштабируемый алгоритм кластеризации [Электрон. ресурс] / В. Орешков. – Режим доступа : <https://basegroup.ru/community/articles/em>

19. Граничин, О.Н. Алгоритм стохастической аппроксимации с рандомизацией на входе для оценивания параметров смеси гауссовых

распределений без учителя при разреженных параметрах [Текст] / О.Н. Граничин // Автоматика и телемеханика. – 2019. – № 8. – С. 44-63.

20. Gaussian mixture models [Electronic resource]. – Access mode : <https://www.kovcomp.co.uk/xlstat/tools/gaussian-mixture-models.html>

21. Гауссова смесь распределений (GMM) [Электрон. ресурс]. – Режим доступа : <https://craftappmobile.com/1073-2/>

22. Carrasco, O.C. Gaussian Mixture Models Explained [Electronic resource] / O.C. Carrasco. – Access mode : <https://towardsdatascience.com/gaussian-mixture-models-explained-6986aaf5a95>

23. Gaussian Mixture Model [Electronic resource] / John McGonagle, Geoff Pilling, Andrei Dobre et al. – Access mode : <https://brilliant.org/wiki/gaussian-mixture-model/>

24. Java Features [Electronic resource]. – Access mode : <https://www.studytonight.com/java/features-of-java.php>

25. Сокеты – Сетевое программирование [Электрон. ресурс]. – Режим доступа : <https://lecturesnet.readthedocs.io/net/low-level/ipc/socket/intro.html>

26. Остапенко, В.В. Методичні вказівки по економічному обґрунтуванню дипломних проєктів для студентів спеціальностей 7.080403 «Програмне забезпечення автоматизованих систем», 7.080402 «Інформаційні технології проєктування» і 7.091501 «Комп'ютерні та інтелектуальні системи і мережі» всіх форм навчання [Текст] / В.В. Остапенко, Е.М. Касьян, Є.М. Сердюк. – Запоріжжя : ЗНТУ, 2009. – 22 с.

27. Правила улаштування електроустановок [Текст] : Нормативне виробничо-практичне видання : вид. 5-те, перероблене й доповнене : затв. М-вом енергетики та вугільної промисловості України 20.06.14 : введення в дію з 20.11.14. – Х. : Міненерговугілля України, 2014. – 793 с.

28. Жидецький, В. Ц. Основи охорони праці [Текст] : підручник / В. Ц. Жидецький. – 5-те вид., доп. – К. : Знання, 2014. – 373 с.

29. Основи охорони праці [Текст] : підручник / В. В. Березуцький, Т. С. Бондаренко, Г. Г. Валенко та ін.; за ред. проф. В. В. Березуцького. – Х. : Факт, 2005. – 327 с.
30. Катренко, Л. А. Охорона праці. Курс лекцій. Практикум : навчальний посібник [Текст] / Л. А. Катренко, Ю. В. Кіт, І. П. Піскун. – 2-ге вид., стер. – Суми : ВТД «Університетська книга», 2007. – 217 с.
31. Кодекс цивільного захисту України [Текст] : [прийнятий ВР України 02.10.2012 р. № 5403-VI] : офіц. текст : станом на 06.12.2012 р. / ВР України. – К. : Алерта, 2012. – 120 с.
32. Шоботов, В. М. Цивільна оборона [Текст] : Навчальний посібник / В. М. Шоботов. – Вид. 2-ге, перероб. – К. : Центр навчальної літератури, 2006. – 438 с.
33. Стеблюк, М. І. Цивільна оборона та цивільний захист : навч. посіб. для вузів [Текст] / М. І. Стеблюк. – К. : Знання, 2013. – 487 с.

ДОДАТОК А
Технічне завдання

Вступ

Метою магістерської роботи є розробка програмного забезпечення знаходження подібної музики для реалізації пошуку нової музики користувачами.

Досягнення мети роботи має бути комплексним за рахунок створення методу, що дозволяє розв'язати проблему, та програмного забезпечення, що реалізує цей метод та дозволяє виконувати пошук нової музики користувачами.

A.1 Підстави для розробки

Підставою для розробки є завдання на магістерську роботу на тему «Дослідження та програмна реалізація методів знаходження подібної музики», що затверджене наказом № 301 від 3 листопада 2020 р. за Національним університетом «Запорізька політехніка».

A.2 Призначення розробки

Програмне забезпечення повинно дозволяти знаходити нові музичні композиції, подібні до тих, які були прослухані користувачем раніше, та мати базові можливості відтворення музики.

A.3 Основні вимоги до програми

A.3.1 Вимоги до функціональних характеристик

Програмне забезпечення повинно забезпечувати виконання наступних функцій:

- додавання, перегляд та редагування виконавців;
- додавання, перегляд та редагування композицій;

- додавання, перегляд та редагування альбомів;
- створення тегів;
- авторизація адміністраторів;
- відтворення композицій;
- пошук композицій;
- пошук виконавців;
- пошук альбомів;
- пошук за тегами;
- навчання моделі;
- пошук подібних композицій.

Вхідними даними є:

- параметри пошуку музичних композицій;
- дані облікових записів;
- вибір музичної композиції.

Вихідними даними є:

- перелік музичних композицій, подібних на поточну;
- нові або змінені дані музичних композицій;
- перелік знайдених музичних композицій.

A.3.2 Умови експлуатації

Програма може бути записана на оптичному, гнучкому або жорсткому магнітному диску.

A.3.3 Вимоги до складу та параметрів технічних засобів

До складу технічних засобів повинен входити персональний комп'ютер або ноутбук, який включає у себе:

- операційну систему – Windows 10;
- оснащений 256 Мб оперативного запам'ятовуючого пристрою;

- процесор Intel Celeron G1830 2800 МГц;
- має доступ до мережі Інтернет;
- засоби керування та відображення інформації (клавіатура, миша та монітор).

A.3.4 Вимоги до транспортування та збереження

Вимоги до транспортування і зберігання аналогічні вимогам, пропонованим до носія, на якому зберігається програма, тобто оптичного, гнучкого або жорсткого магнітного диску.

A.3.5 Вимоги до програмної документації

Програма повинна поставлятися разом з наступними програмними документами:

- текстом програми;
- описом експлуатації та тестування програми.

A.3.6 Вимоги до маркування та упакування

Програма може бути записана на оптичному, гнучкому або жорсткому магнітному диску.

У випадку розповсюдження програмного забезпечення на відповідному носії має бути вказано маркування «Дослідження та програмна реалізація методів знаходження подібної музики».

A.4 Стадії та етапи розробки

Виконання магістерської роботи повинно бути реалізовано на основі наступних етапів:

- розроблення методу знаходження подібної музики;
- аналіз вимог до програмного забезпечення;
- проектування програмного забезпечення;
- реалізація програмного забезпечення;
- тестування і відлагодження програмного забезпечення;
- оформлення документації.

Програмний продукт повинен бути виконаний в терміни, визначені календарним планом, наведеним у завданні на роботу.

A.5 Порядок контролю та приймання

Магістерська робота повинна бути виконана у відповідності з календарним планом, зазначеним у завданні на роботу.

Магістерська робота має бути прийнята консультантами відповідних розділів, визначеними в завданні на роботу, після чого затверджена завідуючим кафедрою, а на 13-й тиждень захищена на засіданні екзаменаційної комісії за відповідним графіком.

ДОДАТОК Б
Текст програми

Б.1 Текст файла CapabilityCompositions.java

```

package SimilarMusic;

import jadex.bdiv3.annotation.*;
import jadex.bdiv3.runtime.ChangeEvent;
import jadex.rules.eca.ChangeInfo;

import java.util.HashMap;
import java.util.Map;

@Capability
public class CapabilityCompositions {

    @Belief
    private Map<Integer, Composition> compositionList = new
HashMap<>();

    @Belief
    public void addComposition (Composition c) {
        compositionList.put(compositionList.size() + 1, c);
    }

    public Map<Integer, Composition> getCompositionList () {
        return compositionList;
    }

    public void loadCompositionList () {
        this.compositionList = Database.getAllCompositions();
    }

    @Goal
    public class Search {
        @GoalParameter
        protected Composition searchComposition;

        public Search(Composition comp) {
            this.searchComposition = comp;
        }
    }
}

```

```

    }
}

@Plan(trigger = @Trigger(factchangeds = "compositionList"))
public void onListChanged(ChangeEvent e) {
    Composition comp = ((ChangeInfo<Composition>)
e.getValue()).getValue();
    compositionList = Method.ProcessData(compositionList);
}

@Plan(trigger = @Trigger(goals = Search.class))
public void searchmusic(Composition comp){
    SCcompositionList =
Method.getSameCluster(compositionList, comp);
    return SCcompositionList;
}
}

```

Б.2 Текст файлу Composition.java

```

package SimilarMusic;

public class Composition {

    private long id;
    private String title;
    private Musician musician;
    private double acousticness;
    private double instrumentalness;
    private double liveness;
    private double valence;
    private double tempo;
    private double speechness;
    private double danceability;
    private double energy;
    private double loudness;

    public Composition (String title,

```

```
double acoustiness,  
double instrumentalness,  
double liveness,  
double valence,  
double tempo,  
double speechness,  
double danceability,  
double energy,  
double loudness) {  
    this.title = title;  
    this.acoustiness = acoustiness;  
    this.instrumentalness = instrumentalness;  
    this.liveness = liveness;  
    this.valence = valence;  
    this.tempo = tempo;  
    this.speechness = speechness;  
    this.danceability = danceability;  
    this.energy = energy;  
    this.loudness = loudness;  
}  
  
public long getId() {  
    return id;  
}  
  
public String getTitle() {  
    return title;  
}  
  
public double getAcoustiness () {  
    return acoustiness;  
}  
  
public double getInstrumentalness () {  
    return instrumentalness;  
}  
  
public double getLiveness () {  
    return liveness;
```

```
}

public double getValence () {
    return valence;
}

public double getTempo () {
    return tempo;
}

public double getSpeechness () {
    return speechness;
}

public double getDanceability () {
    return danceability;
}

public double getEnergy () {
    return energy;
}

public double getLoudness () {
    return loudness;
}

public String getMusicianName () {
    return Musician.getName();
}

public double getMusician () {
    return musician;
}

public void setTitle(String title) {
    this.title = title;
}

public void setAcousticness (double acousticness) {
```

```
        this.acousticness = acousticness;
    }

    public void setInstrumentalness (double instrumentalness) {
        this.instrumentalness = instrumentalness;
    }

    public void setLiveness (double liveness) {
        this.liveness = liveness;
    }

    public void setValence (double valence) {
        this.valence = valence;
    }

    public void setTempo (double tempo) {
        this.tempo = tempo;
    }

    public void setMusician (Musician mus) {
        this.musician = mus;
    }

    public void setSpeechness (double speechness) {
        this.speechness = speechness;
    }

    public void setDanceability (double danceability) {
        this.danceability = danceability;
    }

    public void setEnergy (double energy) {
        this.energy = energy;
    }

    public void setLoudness (double loudness) {
        this.loudness = loudness;
    }

    public void setId(long id) {
```

```

        return this.id=id;
    }
}

```

Б.3 Текст файла Musician.java

```

package SimilarMusic;

public class Musician {

    private long id;
    private String name;
    private String country;

    public Musician (String name,
String country) {
        this.name = name;
        this.country = country;
    }

    public long getId() {
        return id;
    }

    public String getName() {
        return name;
    }

    public double getCountry () {
        return country;
    }

    public void setName(String name) {
        this.name = name;
    }

    public void setCountry(String country) {
        this.country = country;
    }
}

```

```

    }

    public void setId(long id) {
        return this.id=id;
    }

}

```

Б.4 Текст файлу Server.java

```

package SimilarMusic;

import java.io.*;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.Map;

public class Server implements Runnable {

    private static Socket csock;
    private static ServerSocket ssock;
    private static BufferedReader br;
    private static BufferedWriter bw;

    @Override
    public void run() {

        try {
            try {
                ssock = new ServerSocket(25547);
                System.out.println("Сервер успішно розпочав
роботу");

                clientSocket = ssock.accept();
                try {
                    br = new BufferedReader(new
InputStreamReader(csock.getInputStream()));
                    bw = new BufferedWriter(new
OutputStreamWriter(csock.getOutputStream()));

```

```

        while (true){
            String str = br.readData();
            Map<Integer, Album> imus =
DataBase.toMap(str);

remBDI.getInstance().bdiAgentFeature.dispatchTopLevelGoal(remBDI.getIn
stance().CapabilityCompositions.new SocketAccept(imus));
        }
    } finally {
        csock.close();
        br.close();
        bw.close();
    }
} finally {
    System.out.println("Сервер завершив свою
роботу");

    ssock.close();
}
} catch (IOException e) {
    System.err.println(e);
}
}

public void send (String t){
    try{
        bw.write(t + "\n");
        bw.flush();
    }
    catch (Exception e){
        System.out.println("Помилка");
    }
}
}
}

```

Б.5 Текст файла ClusterMethod.java

```

package SimilarMusic;

import Jama.*;
import java.io.*;
import java.util.List;
import java.util.ArrayList;
import java.util.stream.DoubleStream;
import java.util.stream.Collectors;

public class ClusterMethod {

    private double be;
    private double se;

    private ArrayList<ArrayList<Double>> P1;
    private ArrayList<Double> P2;
    private ArrayList<ArrayList<ArrayList<Double>>> P3;

    public ArrayList<ArrayList<Double>>
group(ArrayList<ArrayList<Double>> compositions,
        int features,
        int n,
        int q) {

        P1 = Base.getBegValues(compositions, q, n);
        P2 = new ArrayList<>();
        P3 = new ArrayList<>();

        int[] mark = getMark(compositions, P1, q);

        int[] marks = new int[n];

        int l;
        int e;

```

```

int t;

for (l = 0; l < n; l++) {
    marks[mark[l]]++;
}
for (l = 0; l < n; l++) {
    double z = (double) (marks[l]) / (double) (q);
    P2.add(z);
}

for (l = 0; l < n; l++) {
    ArrayList<ArrayList<Double>>
StatementCompositions = new ArrayList<ArrayList<Double>>();
    for (e = 0; e < q; e++) {
        if (mark[e] == 1) {
            StatementCompositions
.add(compositions.get(e));
        }
    }
    P3.add(Base.det(q, features,
StatementCompositions));
}

while (true) {
    ArrayList<ArrayList<Double>> arr =
Base.getParams(compositions, P1, q, n, features);
    double[][] chance = new double[q][n];
    for (l = 0; l < q; l++) {
        for (e = 0; e < n; e++) {
            chance[l][e] = arr.get(l).get(e) *
P2.get(e);
        }
    }

    double[] res = Base.Calc(chance, 2);
    for (l = 0; l < q; l++) {
        for (e = 0; e < n; e++) {
            double r = chance[l][e] / res[l];
            chance[l][e] = r;

```

```

    }
}

double[] val = Base.Calc (chance, 1);
int sz = val.length;
double[] pval = new double[sz];
for (l = 0; l < sz; l++) {
    pval [l] = 1 / val[l];
}

double[][] aval = new Matrix
(Base.paral(pval)).times (new Matrix(chance).transpose()).times(new
Matrix (Base.DataDet(compositions))).DataDet();

P1.clear();

ArrayList<ArrayList<Double>> pd = new
ArrayList<>(aval.length);
for (double[] av : aval) {
    ArrayList<Double> pdt = new ArrayList<>();
    for (double a : av) {
        pdt.add(a);
    }
    pd.add(pdt);
}
P1 = pd;

double[][] pp2 = new double[n][1];
for (l = 0; l < sz; l++) {
    pp2 [l][0] = val[l] / q;
}

P2.clear();
P2.add(pp2 [0][0]);
P2.add(pp2 [1][0]);
P2.add(pp2 [2][0]);

double[][][] pp3 = new
double[n][features][features];

```

```

        for (l = 0; l < n; l++) {
            double[][] arr1;
            double[] arr2 = aval[l];
            double[][] arr3 = new double[q][2];
            for (e = 0; e < q; e++) {
                e[mm] = arr2;
            }

            Matrix cs = new
Matrix(getArray(compositions));
            Matrix csp = Base.minus(cs, new
Matrix(arr3));

            arr1 = csp.getArray();

            double[] r1 = new double[q];
            for (e = 0; e < q; e++) {
                r1[e] = chance[e][1];
            }
            double[][] r2 = Base.paral(r1);

            double[][] r3 = new
Matrix(arr1).transpose().times(new Matrix(r2).times(new
Matrix(arr1))).getArray();

            for (e = 0; e < features; e++) {
                for (t = 0; t < features; t++) {
                    pp3 [1][e][t] = r3 [e][t] / val[t];
                }
            }
        }

P3.clear();

for (t = 0; t < pp3.length; t++) {
    ArrayList<ArrayList<Double>> mas = new
ArrayList<>();

    for (e = 0; e < pp3 [t].length; e++) {
        double[] tm = pp3 [t][e];
    }
}

```

```

        List<Double> tm2 =
DoubleStream.of(tm).boxed().collect(Collectors.toList());
        ArrayList<Double> tm3 = new
ArrayList<Double>( tm2);

        mas.add(tm3);
    }
    P3.add(mas);
}

double[][] op = new
Matrix(getArray(arr)).times(new Matrix(p2)).getArray();

for (l = 0; l < q; l++) {
    op[l][0] = Math.log(op[l][0]);
}

double v = Base.Calc (op, 1)[0];

if (v - be < se) {
    return arr;
}

be = v;

}
}

```

Б.6 Текст файла CompositionsForm.form

```

<?xml version="1.0" encoding="UTF-8"?>
<form xmlns="http://www.intellij.com/uidesigner/form/"
version="1" bind-to-class="SimilarMusic.MusicForm">
    <grid id="27dc6" binding="panel" layout-
manager="GridLayoutManager" row-count="3" column-count="1" same-size-
horizontally="false" same-size-vertically="false" hgap="-1" vgap="-1">
        <margin top="0" left="0" bottom="0" right="0"/>
        <constraints>
            <xy x="10" y="10" width="400" height="500"/>

```

```

        </constraints>
        <properties/>
        <border type="none"/>
        <children>
            <component id="e6358" class="javax.swing.JButton"
binding="btnEditComposition">
                <constraints>
                    <grid row="1" column="0" row-span="1" col-span="1"
vsize-policy="0" hsize-policy="3" anchor="0" fill="1" indent="0" use-
parent-layout="false">
                        <preferred-size width="300" height="50"/>
                    </grid>
                </constraints>
                <properties>
                    <text value="Редагувати"/>
                </properties>
            </component>
            <scrollpane id="a990d">
                <constraints>
                    <grid row="0" column="0" row-span="1" col-span="1"
vsize-policy="7" hsize-policy="7" anchor="0" fill="3" indent="0" use-
parent-layout="false"/>
                </constraints>
                <properties/>
                <border type="none"/>
                <children>
                    <component id="39d2a" class="javax.swing.JList"
binding="compositionList">
                        <constraints/>
                        <properties/>
                    </component>
                </children>
            </scrollpane>
            <component id="f3a49" class="javax.swing.JButton"
binding="btnNewComposition">
                <constraints>
                    <grid row="2" column="0" row-span="1" col-span="1"
vsize-policy="0" hsize-policy="3" anchor="0" fill="1" indent="0" use-
parent-layout="false"/>

```

```

        </constraints>
        <properties>
            <text value="Додати"/>
        </properties>
    </component>
</children>
</grid>
</form>

```

Б.7 Текст файлу CompositionsForm.java

```

package SimilarMusic;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.Map;

public class MusicForm {
    private JPanel jp;
    private JButton btnEditComposition;
    private JList listCompositions;
    private JButton btnNewComposition;
    private int size = 100;

    public MusicForm() {
        btnEditComposition.addActionListener(new
ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                String comp = (String)
listCompositions.getSelectedValue();
                if (comp != null) {
                    createWindowEditComposition(comp);
                }
            }
        })
    }
}

```

```

        });
        btnNewComposition.addActionListener(new ActionListener()
{
    @Override
    public void actionPerformed(ActionEvent e) {
        createWindowNewComposition ();
    }
});
}

public JPanel getPanel() {
    return jp;
}

public JButton getBtnEditComposition () {
    return btnEditComposition;
}

public void updateCompositions(Map<Integer, Composition >
all){
    DefaultListModel dm = new DefaultListModel();
    listCompositions.setModel(dm);
    int count = 1;
    for (int num : all.keySet()){
        Composition comp = list.get(num);
        dm.addElement(id + ". " + comp.getMusicianName () +
" - " + comp.getTitle());
    }
}

public void createWindowEditComposition (String comp){
    JDialog jd = new
JDialog((JFrame) SwingUtilities.getWindowAncestor(jp),
"EditCompositionForm", true);
    jd.getContentPane().add(new EditCompositionForm
(comp).panel);
    Toolkit kt = Toolkit.getDefaultToolkit();
    Dimension v = kt.getScreenSize();
    int x = v.width / 2;

```

```

        int y = v.height / 2;
        jd.setBounds(x - size, y - size, 400, 400);
        jd.pack();
        jd.setVisible(true);
    }

    public void createWindowNewComposition () {
        JDialog jd = new
JDialog((JFrame) SwingUtilities.getWindowAncestor(jp),
"NewCompositionForm", true);
        jd.getContentPane().add(new NewCompositionForm().panel);
        Toolkit kt = Toolkit.getDefaultToolkit();
        Dimension v = kt.getScreenSize();
        int x = v.width / 2;
        int y = v.height / 2;
        jd.setBounds(x - size, y - size, 400, 400);
        jd.pack();
        jd.setVisible(true);
    }
}

```

ДОДАТОК В
Слайди презентації

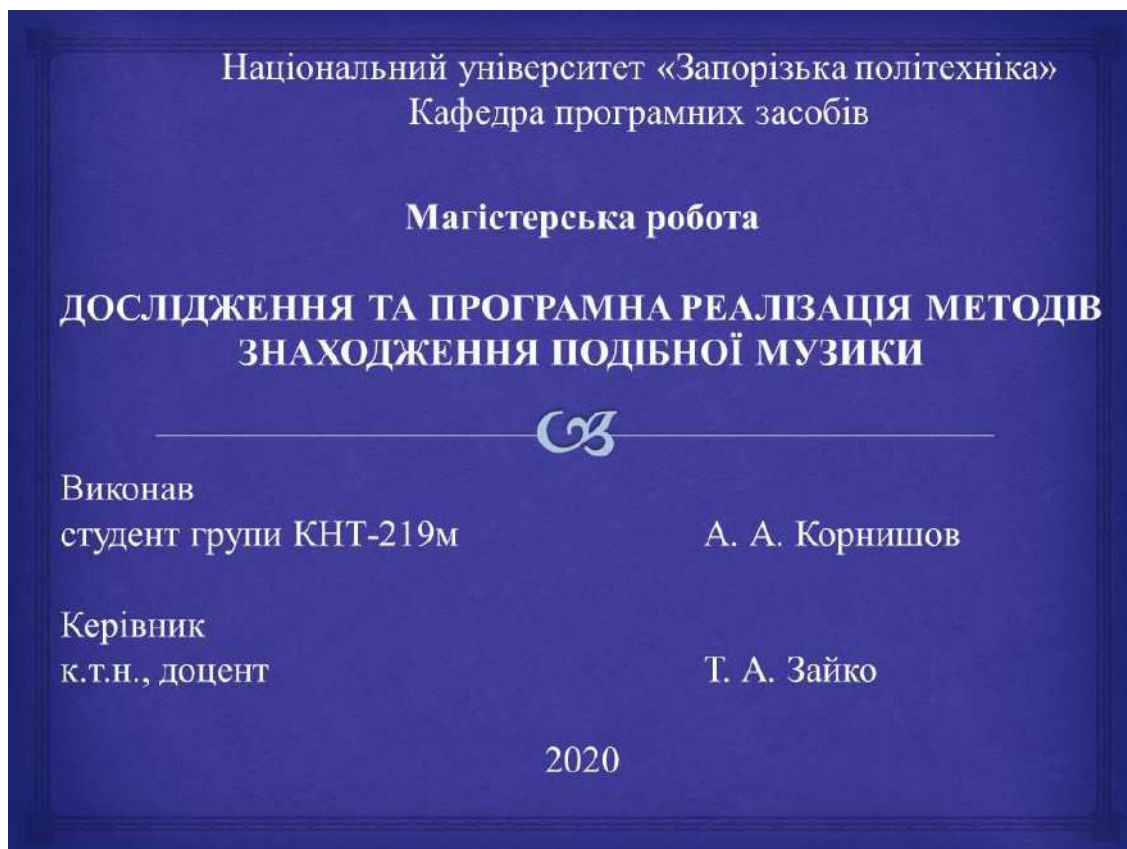


Рисунок В.1 – Слайд 1

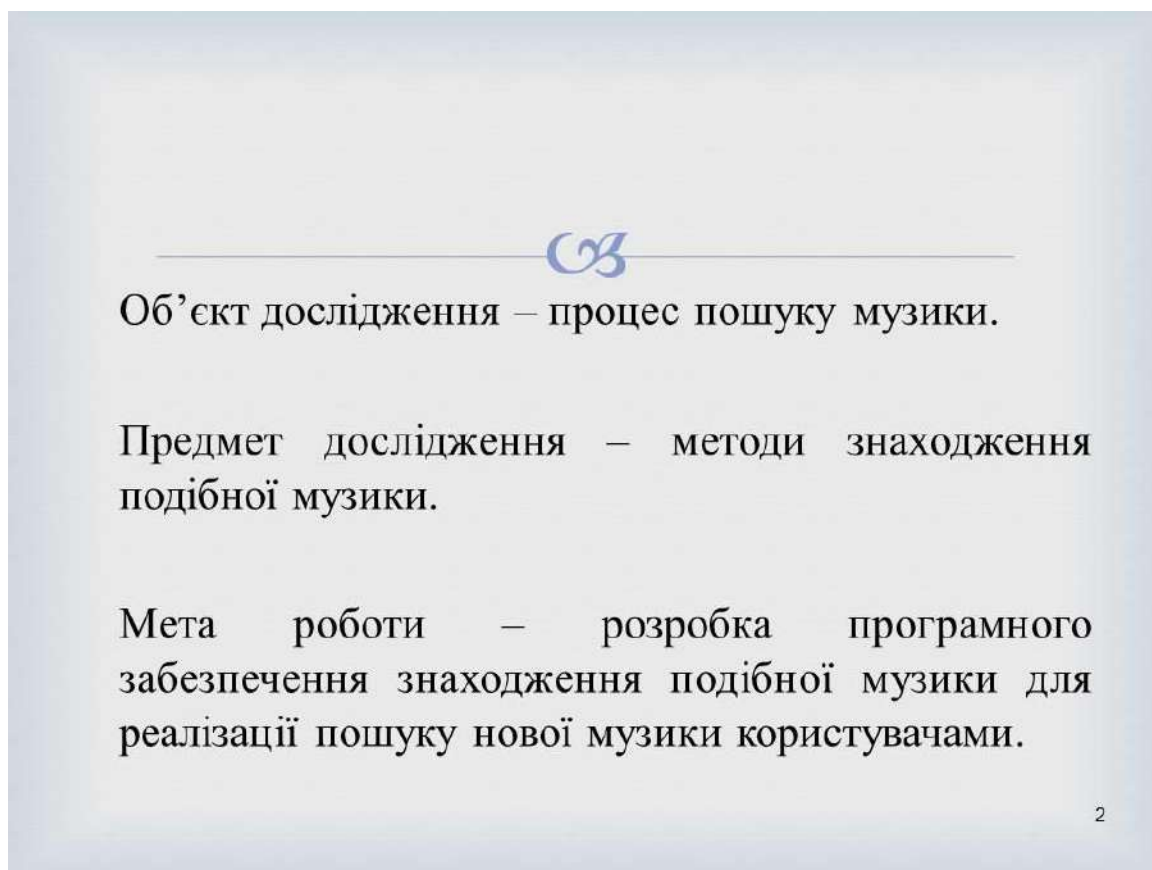


Рисунок В.2 – Слайд 2

Завдання



- ✓ дослідження методів знаходження подібної музики;
- ✓ проектування програмного забезпечення;
- ✓ розробка програмного забезпечення;
- ✓ тестування та відлагодження програми;
- ✓ економічне обґрунтування доцільності розробки;
- ✓ забезпечення заходів охорони праці та безпеки в надзвичайних ситуаціях.

3

Рисунок В.3 – Слайд 3

Метод знаходження подібної музики

- На першому етапі методу має бути сформовано вибірку даних. Кожен екземпляр характеризується властивостями:



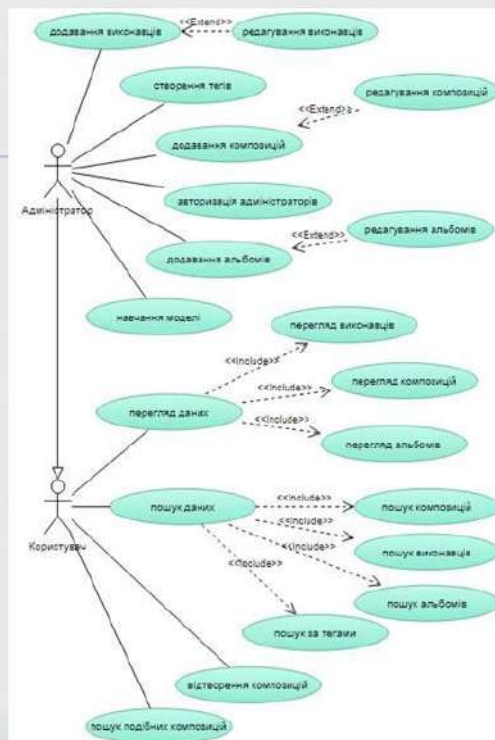
- акустичність;
- танцювальність;
- енергійність;
- інструментальність;
- концертність;
- гучність;
- мовленнєвість;
- валентність;
- ритмічність.

- На другому етапі застосовується модель GMM, в результаті чого визначаються кластери музичних композицій за вибіркою даних, сформованою за результатом першого етапу.
- На третьому етапі користувач вибирає або задає певну музичну композицію. На основі сформованої моделі виконується визначення кластеру, до якого належить дана композиція. В результаті повертаються музичні композиції, що належать даному кластеру з найбільшою вірогідністю.

4

Рисунок В.4 – Слайд 4

Діаграма прецедентів



5

Рисунок В.5 – Слайд 5

Порівняння мов програмування для реалізації програми



Критерій	C#	Java
Об'єктно-орієнтована парадигма	Чиста	Чиста
Залежність від платформи	Орієнтований код на Windows	Ні
Безкоштовне виведення	Ні	Так
Безпечність	Менша	Так
Розподіленість	Так	Так
Створення настільних застосунків	Так	Так
Створення застосунків для Telegram	Так	Так

6

Рисунок В.6 – Слайд 6



Рисунок В.7 – Слайд 7



Рисунок В.8 – Слайд 8



Рисунок В.9 – Слайд 9

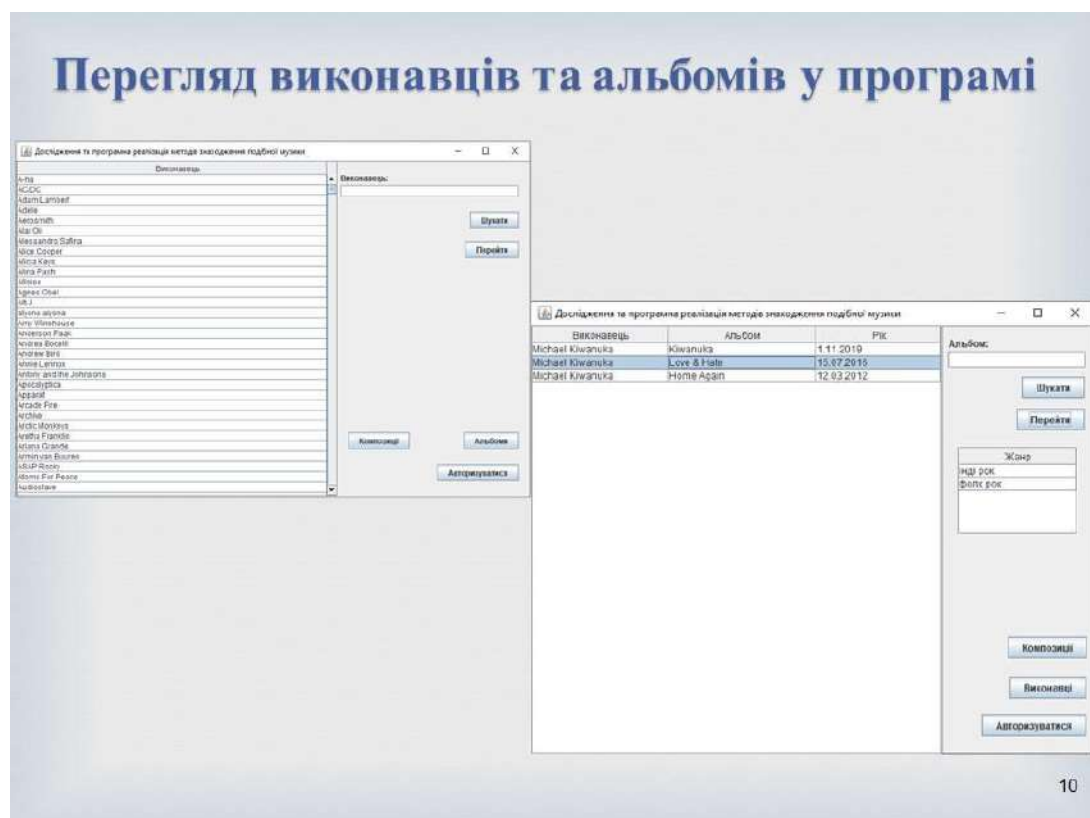


Рисунок В.10 – Слайд 10

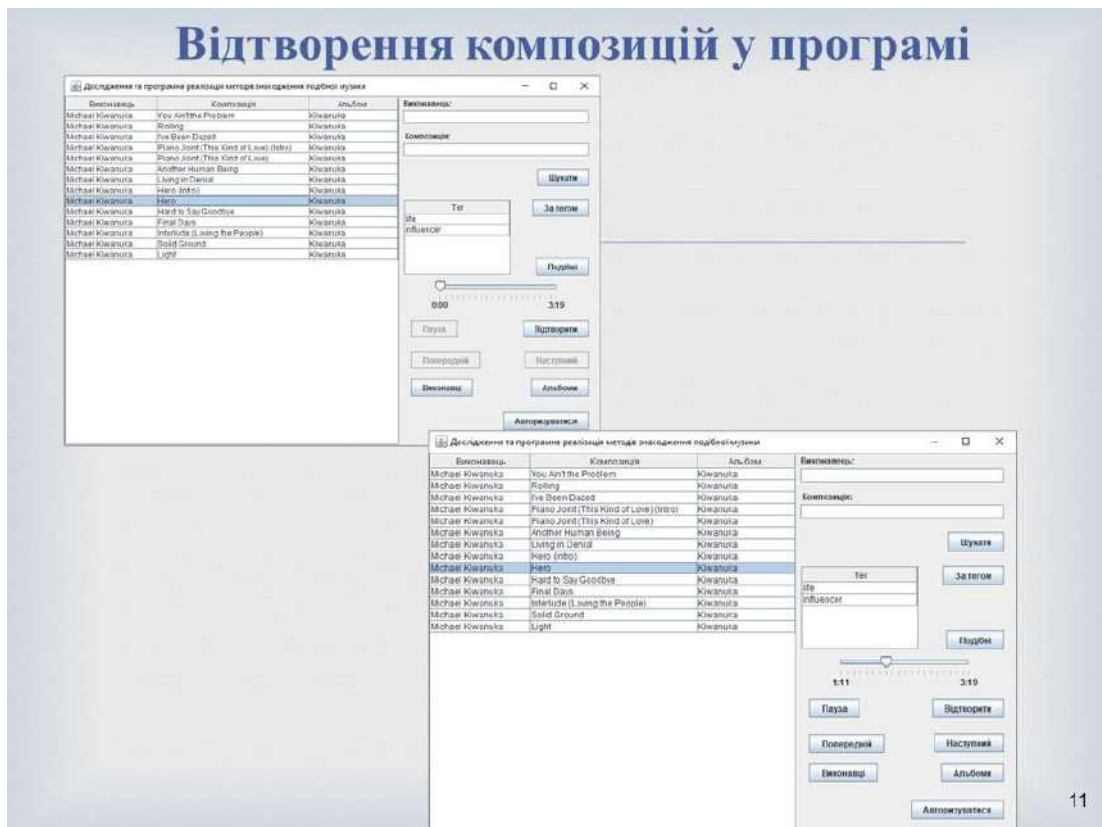


Рисунок В.11 – Слайд 11

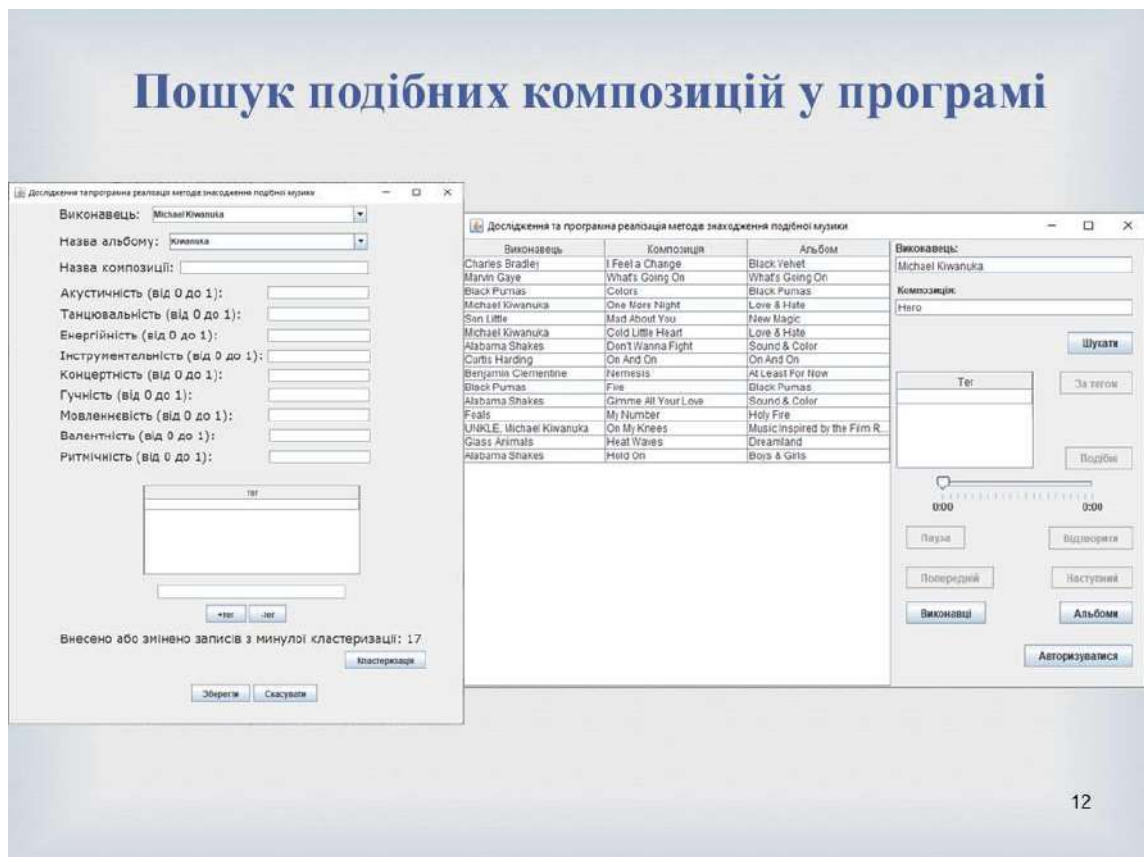


Рисунок В.12 – Слайд 12

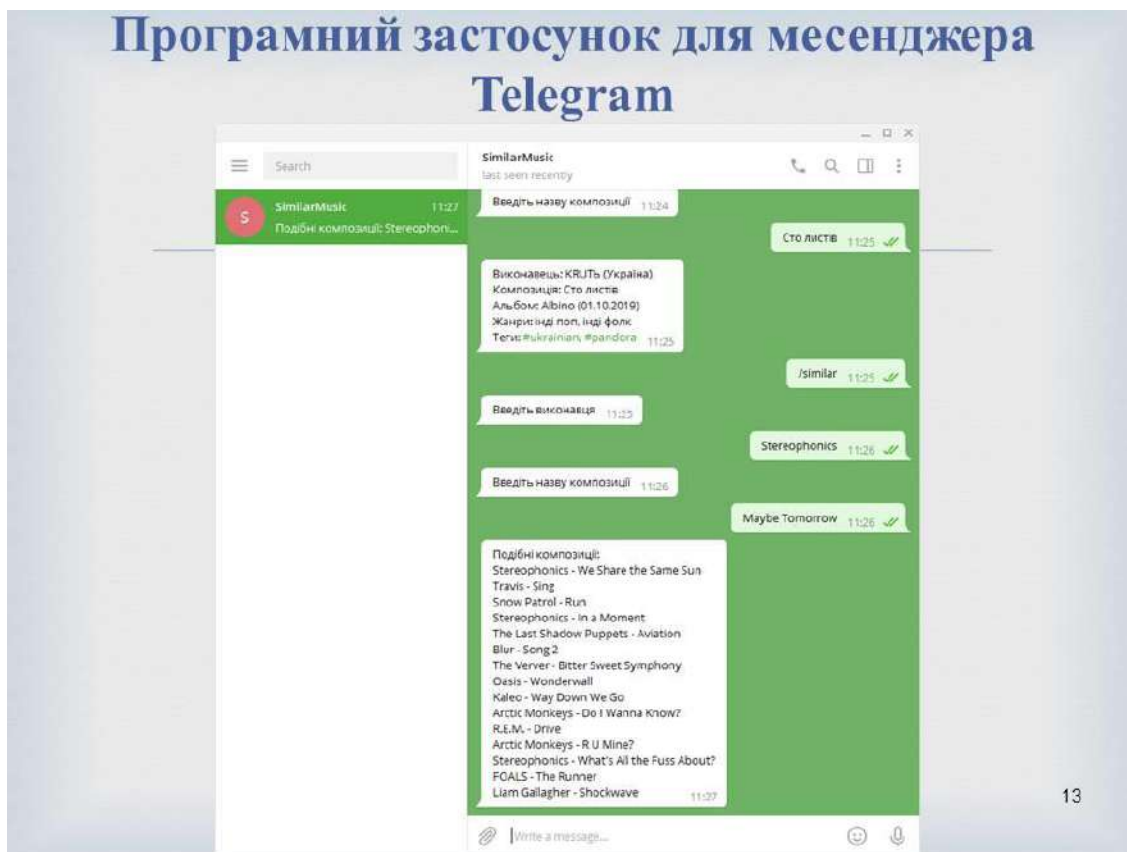


Рисунок В.13 – Слайд 13

Висновки



Розроблено програмне забезпечення, що складається з настільного програмного застосунку для операційної системи Windows, який використовується адміністратором для внесення даних та ініціалізації роботи методу, та програмного застосунку для месенджера Telegram, який використовується безпосередньо користувачами для пошуку подібної музики. Описано рішення, які було прийнято під час реалізації програмного забезпечення.

Наукова новизна роботи в розробленому методі знаходження подібної музики, який використовує метод кластеризації на основі суміші гаусівських розподілів, визначаючи в якості набору ознак музичної композиції показники акустичності, танцювальності, енергійності, інструментальності, концертності, гучності, мовленнєвості, валентності, ритмічності, що дозволяє виконувати пошук композицій без використання характеристик сигналу з визначенням подібності за кластерами.

14

Рисунок В.14 – Слайд 14