Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

МОДЕЛЮВАННЯ ТА РОЗРОБЛЕННЯ ВЕБЗАСТОСУНКУ МРЗ-ПЛЕЄРА ДЛЯ ВІДТВОРЕННЯ ТА ОРГАНІЗАЦІЇ МУЗИЧНИХ ФАЙЛІВ ОФЛАЙН
(тема)

Виконав:
студент 4 курсу, групи ІТІНФ-19-2

Шуліка Д.О.
(прізвище, ініціали)

Спеціальності 122 Комп’ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник доц. Творошенко І.С.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Кобилін О.А.
(підпис) (прізвище, ініціали)

2023 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)

Кафедра Інформатики
(повна назва)

Рівень вищої освіти перший (бакалаврський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Шуліці Дмитру Олександровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Моделювання та розроблення вебзастосунку MP3-плеєра для відтворення та організації музичних файлів офлайн

затверджена наказом університету від 15 травня 2023 року № 474 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 22 травня 2023 р.

3. Вихідні дані до роботи науково-методична та науково-технічна література, матеріали конференцій, дані інтернет-мережі, Аудіо та MIDI бібліотека NAudio, бібліотека для роботи з метаданими файлів TagLib-sharp, мова програмування C#, середовище розробки Microsoft Visual Studio, бібліотека для роботи з базами даних SQLite, система управління базами даних SQLite Studio.

4. Перелік питань, що потрібно опрацювати в роботі
1. Огляд існуючих аналогів розроблюваного застосунка та визначення переваг та недоліків
2. Моделювання бази даних для застосунка MP3-плеєра.
3. Моделювання наповнення застосунка MP3-плеєра.
4. Програмна реалізація застосунка MP3-плеєра.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) Актуальність проблеми організації музичної бібліотеки, постановка задачі, опис обраного варіанта організації музичних файлів.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Консультант з дотримання діючих стандартів та норм	Доцент Творошенко І.С.		

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	10.04.2023	
2	Аналіз завдання, підбір літератури	11.04.23-16.04.23	
3	Аналіз літератури з досліджуваної проблеми	17.04.23-21.04.23	
4	Аналіз технічних засобів	22.04.23-30.04.23	
5	Моделювання структури та наповнення застосунка	01.05.23-09.05.23	
6	Програмна реалізація	10.05.23-23.05.23	
7	Оформлення пояснювальної записки	24.05.23-26.05.23	
8	Перевірка на плагіат	27.05.23	
9	Рецензування	28.05.23	
10	Підготовка презентації та доповіді	29.05.23-30.05.23	
11	Занесення роботи в електронний архів	31.05.23	
12	Попередній захист кваліфікаційної роботи	31.05.23	

Дата видачі завдання 10 квітня 2023 р.

Студент _____
(підпис)

Керівник роботи _____ доц. Творошенко І.С.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 68 с., 3 табл., 28 рис., 33 джерела.

ВІДТВОРЕННЯ АУДІОФАЙЛІВ, ОРГАНІЗАЦІЯ МУЗИЧНИХ КОМПОЗИЦІЙ, СПИСКИ ВІДТВОРЕННЯ, ТЕГИ, МЕТАДАНІ.

Об’єктом роботи є набір папок з файлами на персональному комп’ютері та аудіофайли у форматі .mp3, що містяться в них.

Метою роботи є розробка застосунку MP3-плеєра для відтворення та організації музичних файлів офлайн, який дозволятиме користувачу організовувати свою власну аудіотеку та комфортно відтворювати аудіофайли.

Використано методи системного аналізу при моделюванні структури застосунку та структури бази даних. Розглянуто наявні інструменти для виконання поставленої задачі.

У результаті роботи спроєктовано та реалізовано програмний застосунок для відтворення та організації аудіофайлів. Виконано функції відтворення та відображення даних музичних файлів.

PLAYBACK OF AUDIO FILES, ORGANIZE MUSIC TRACKS, PLAYLISTS, TAGS, METADATA.

The object of the research is a set of folders with files on a personal computer and audio files in .mp3 format contained in them.

The aim of the research is to develop an MP3 player application for playing and organizing music files offline, which will allow the user to organize their own audio library and comfortably play audio files.

The methods of system analysis were used in modeling the application structure and database structure. The available tools for performing the task are considered.

As a result, a software application for playing and organizing audio files was designed and implemented. Music file playback and data display functions are performed.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ПК – персональний комп’ютер

Трекліст – список композицій в альбомі або списку відтворення

Лейбл – компанії, яка випускає та продюсує музику. Лейбл може мати власні звукозаписні студії, промоутерів та інші ресурси для того, щоб допомогти музикантам записати та продати свою музику

БД – база даних

Плейлист – список відтворення композицій, здебільшого створюється користувачем

ВСТУП

Музика в житті сучасної людини займає важливу нішу, як і століття тому. В нинішні часи існує багато способів зберігання та відтворення музики. Пластинки та диски давно зайняли своє місце в історії та відображають віддання шани минулому, є об’єктами зацікавлення колекціонерів та поціновувачів виконавців тих чи інших часів. Також вони є предметом ексклюзивності. Можна побачити як відомі музиканти та групи сьогодення видають альбоми на дисках, особливо, якщо такі альбоми мають позначку Deluxe edition, тобто мають в собі унікальні, або перероблені композиції, недоступні простим слухачам.

Більшості меломанів та звичайних поціновувачів музики є доступним широке коло стрімінгових платформ (сервісів, де можна викладати і прослуховувати музику) або вебсайтів, на котрих можна знайти і завантажити бажаний аудіофайл. Перший варіант може стати комфортним лише у разі отримання платної підписки на сервіс. Також проблемою може стати бажання артиста працювати з тою чи іншою платформою. Тому сайти, присвячені музиці лишаються досить популярними.

Але далеко не завжди користувач має доступ до мережі Інтернет. Попередньо встановлені програвачі не задовольняють певні побажання слухача.

Так, виникає потреба в засобі зберігання та, що важливіше, в організації аудіофайлів. Це стосується не лише мобільних пристроїв або ПК окремо. Принципи роботи комфортного відтворення, відображення та налаштування музикальної колекції користувача однакові для будь-якого пристрою. Простота, інтуїтивність, комфорт, порядок – ось що саме потрібно користувачу.

1 АНАЛІЗ ІСНУЮЧИХ ЗАСТОСУНКІВ MP3-ПЛЕЄРА ДЛЯ ВІДТВОРЕННЯ ТА ОРГАНІЗАЦІЇ МУЗИЧНИХ ФАЙЛІВ ОФЛАЙН В УКРАЇНІ ТА ЗА КОРДОНОМ

1.1 Сучасний стан розвитку застосунків MP3-плеєра для відтворення та організації музичних файлів офлайн в Україні та за кордоном

Людина завжди прагне порядку. Фраза «тому, хто приручив безлад, хаос благоволить» в жодному разі не може бути застосована до предметної області «Музика». Не зважаючи на довільність форми представлення композиції слухачам, є певні правила, котрі тою чи іншою мірою мають бути дотримані.

Для порівняння відомих та не надто відомих екземплярів застосунків програвача спочатку потрібно зрозуміти, що містить в собі музична композиція у вигляді аудіофайлу. В даний момент не беруться до уваги способи створення, конвертації та відтворення музичних файлів. Мова піде про інформацію, призначену для користувача або повністю, або частково. Маються на увазі ознаки композиції, тобто її назва, автор(и), назва альбому, жанр та ін. Така інформація міститься у так званих тегах. Це, як атрибути об’єкта класу у програмуванні, але можуть приймати здебільшого будь-які текстові значення. Завдяки тегам відбувається сортування композицій, виведення коректних даних пісні чи мелодії, котра відтворюється програмою в даний момент, а також формування альбомів (списків відтворення, створених автором, а не власне користувачем).

Саме так, коректне відображення порядку пісень на сторінці «Альбоми» відбувається завдяки тегам та їх правильному заповненні.

Не зважаючи на відмінність методів розробки для різних платформ, застосунки плеєра всі повинні мати спільні риси. Тому зазвичай, програвачі для мобільних пристроїв мають повне право виступати в якості прикладів для аналогів на ПК.

Нижче буде наведено декілька знімків екрана, котрі зображують правильний порядок пісень в автоматично створеному альбомі в мобільному плеєра PI Music player від розробників Musicophilia (рис. 1.1, 1.2 [1]).

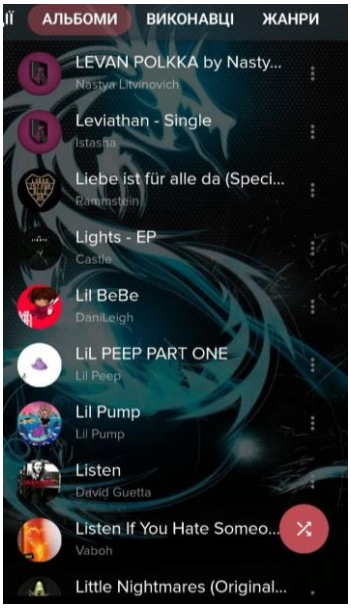

Рисунок 1.1 – Вкладка «Альбоми» у мобільному плеєрі PI Music player

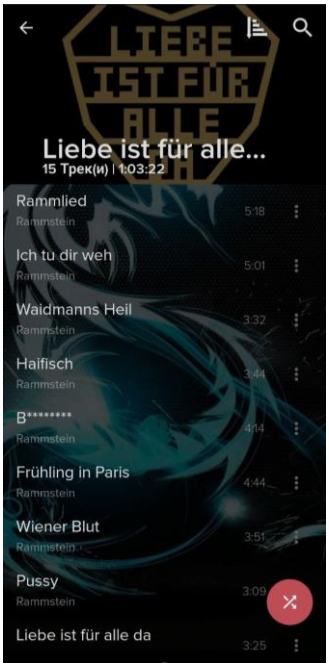

Рисунок 1.2 – Вміст альбому «Liebe Ist Für Alle Da»
у мобільному плеєрі PI Music player

На рисунках 1.3 та 1.4 [2] зображено знімки екрана одного з найпопулярніших редакторів тегів для мобільних пристроїв від розробника filobotto – AutomaTag. Цей застосунок наведено для того, щоб показати всі теги аудіофайла. Функціонал даного програмного забезпечення буде взятий, як приклад правильної роботи з тегами, оскільки дає можливість користувачу, автоматично або власноруч, повністю редагувати теги аудіофайла аби забезпечити систематичність музичної бібліотеки.

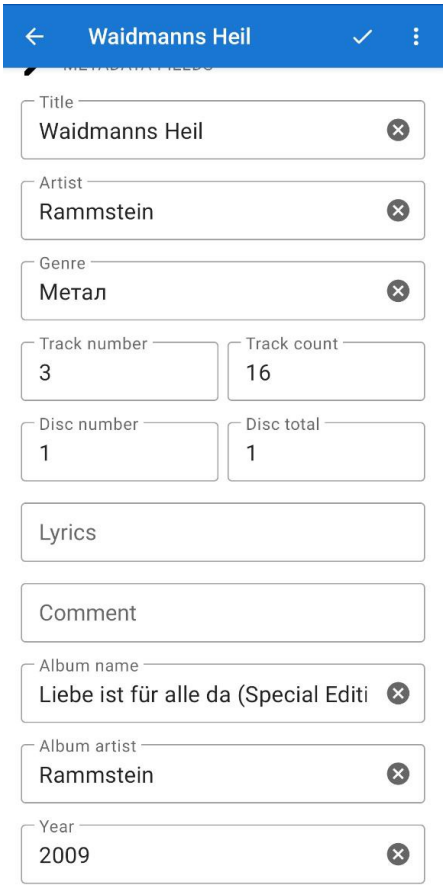

Рисунок 1.3 – Вкладка редагування пісні Weidmans Heil редактора тегів AutomaTag

Оскільки основну вимогу для застосунку MP3-плеєра розглянуто, можна переходити до розгляду існуючих програм для ПК.

Найважливішими критеріями будуть:

– наявність простого та інтуїтивно зрозумілого користувацького інтерфейсу;

– функціональне наповнення застосунку.

Windows Media Player. Вбудований медіапрогравач від Microsoft, певне, широко відомий серед поціновувачів музики. Даний застосунок дозволяє відтворювати величезну кількість мультимедіа файлів. Це чудовий варіант, щоб організовувати свою бібліотеку фільмів та зображень. Щодо теми «Музика», то застосунок частково задовольняє потреби користувача, проте лише частково. Нижче наведено скріншот вкладки музика (рис. 1.4).

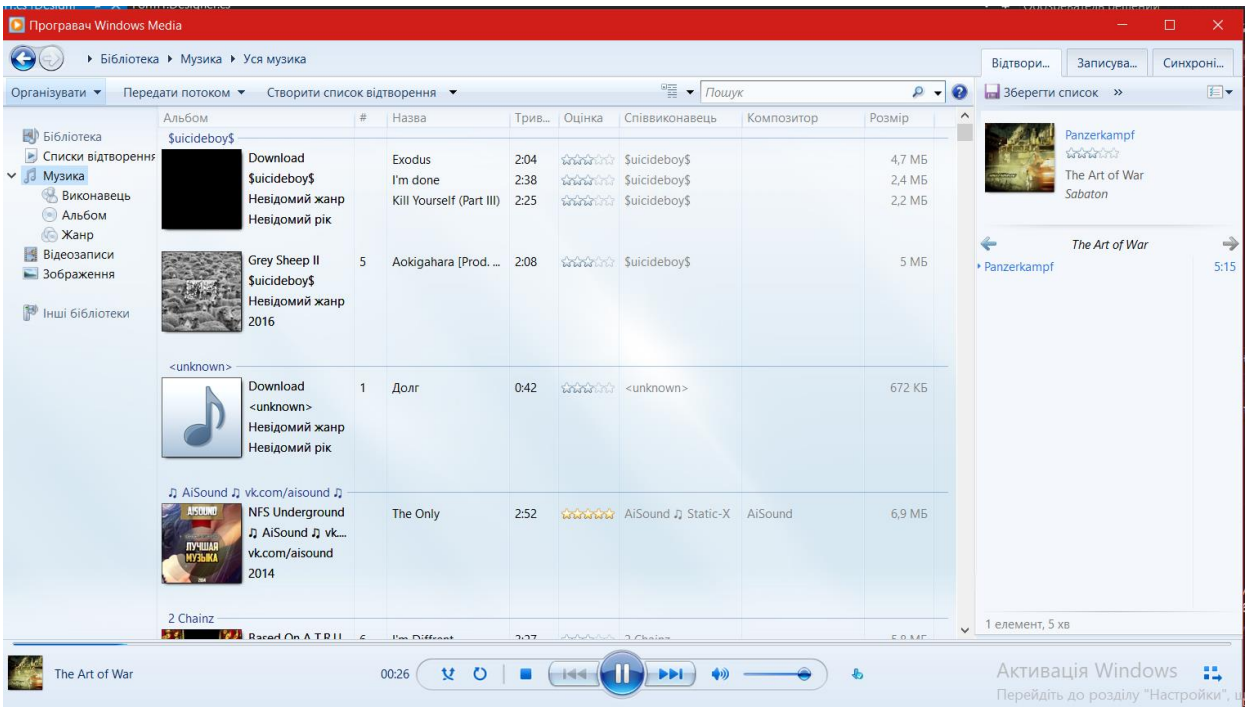

Рисунок 1.4 – Windows Media Player, вкладка музики

Порівнюючи з аналогами для різних платформ, можна заявити про незручність інтерфейсу (інтерфейс – річ дуже ситуативна, тому кожному користувачу програма може бути комфортна по-різному). Що саме мається на увазі?

Вся музика на пристрої об’єднана в одну бібліотеку, проте не об’єднана в одну чергу за замовчуванням (пісні програються в порядку черги лише при переході в режим відтворення і відсортовані по виконавцях). Для рядового користувача це може викликати незручності у користуванні. Це також впливає на організацію списків відтворення певним чином.

Незручність полягає в тому, що інформація, хоч і представлена в широкому обсязі, проте не надто вдалим способом.

Як можна побачити на знімку екрана, музика надана у списку, що сортується за виконавцем і об’єднана за альбомами. Щоб перейти до прослуховування, потрібно провести кілька зайвих маніпуляцій.

На прикладі з рисунку бачимо багато невідформатованих належним чином композицій: невідомий альбом, виконавець, рік тощо. Це не було б недоліком, якби була присутня можливість одразу прослухати композицію, а вже потім її власноруч відредагувати.

Стосовно формування користувацьких списків відтворення, застосунок має велику перевагу: можна як перетягнути всі пісні виконавця, так і конкретну обрану композицію. Також при формуванні надана можливість змінювати порядок пісень просто перетягнувши обрану за допомогою миші в потрібне місце в черзі.

Програвання списків відтворення має певні проблеми, котрі стосуються інформативності. На рисунку 1.5 зображене вікно списку відтворення.

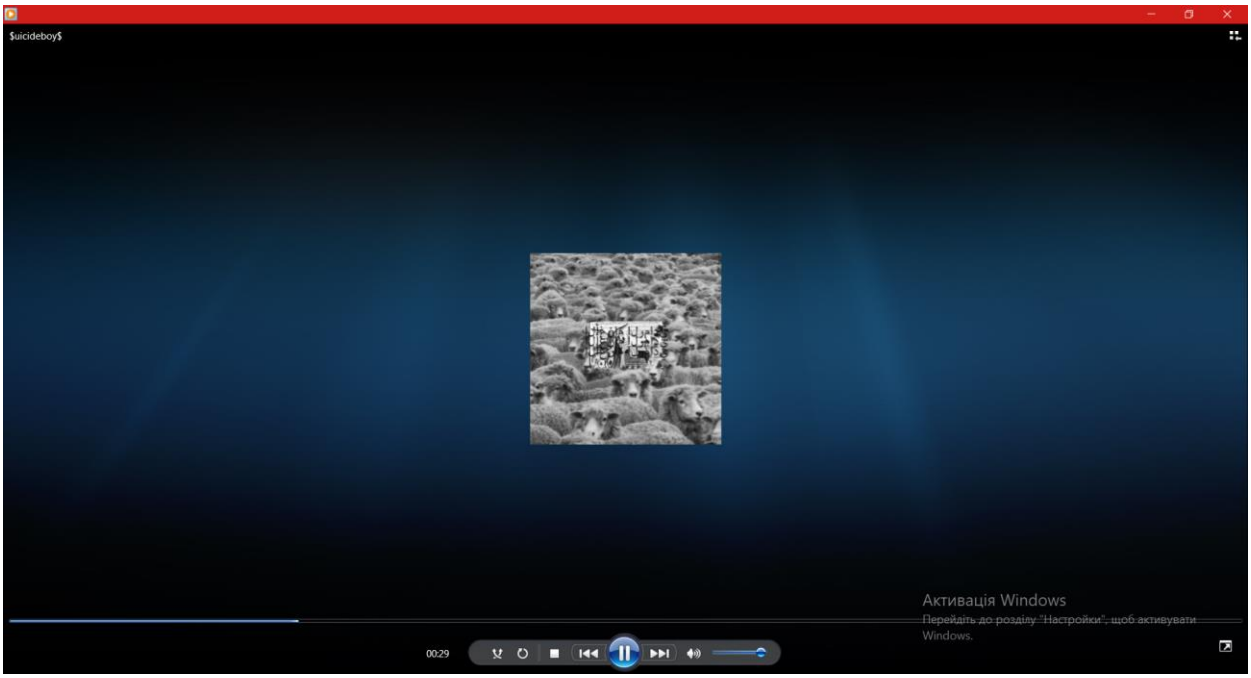

Рисунок 1.5 – Список відтворення Windows Media Player

Інформацію про композицію, що програється в даний момент, подано досить дивно. У верхньому лівому кутку надана інформація змінюється, показуючи назву альбому, виконавця та саму назву пісні почергово, що є не надто зручним рішенням для відображення вмісту тегів.

Редагування інформації про пісню є неповним і відбувається на відповідній сторінці (рис. 1.4). Представлені можливості, хоч і дають більш розширений вибір тегів для змін, але не повні.

Тому можна зробити висновки про незручність даного застосунку для постійного використання, навіть не зважаючи на деякі очевидні переваги.

Музика Groove. Розроблений Microsoft, з урахуванням проблем попередника, застосунок Groove music став повноцінною заміною для Windows Media Player. Цей програвач цілком задовольняє користувачів своєю інтуїтивністю та простотою використання, зручними представленням та організацією бібліотеки. Як і в програвачі Windows можна додавати різні папки до бібліотеки, проте виконана дана функція набагато комфортнішою. Виведення всіх композицій на екран за замовчуванням відбувається за алфавітним порядком (спочатку символи, потім латиниця, а далі кирилиця). Сортування можна просто налаштувати за альбомом, виконавцями, назвою, або датою додання. Останнє дозволяє швидко знаходити нещодавно додані пісні і оперативно редагувати інформацію про них (рис. 1.6).

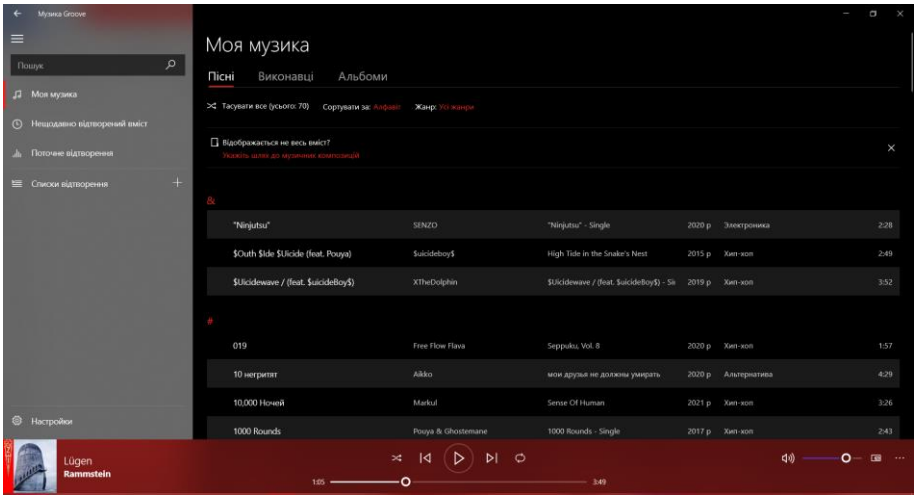

Рисунок 1.6 – Скріншот головного меню Groove Music

Зовнішній вигляд програми чітко дає зрозуміти, яка композиція зараз відтворюється, уся інформація, як в черзі так і в меню програвача подана доступно і чітко.

Рухаючись по меню вгорі можна побачити виконавців та альбоми. У вікні збоку можна створити список відтворення або перейти до вже існуючих.

Організація альбомів вдала, чітко показує позицію пісні в треклісті та виводить основну інформацію (рис. 1.7). Такий підхід до відображення вмісту альбому дає правильно зрозуміти користувачу порядок та наявність композицій і виявити помилки в тегах файлів, просто звірившись з інформацією з Інтернету стосовно альбому. Єдиним незначним недоліком можна вважати відсутність загальної тривалості альбому, проте це розробники перекривають наявністю багатьох інших функцій.

Існує навіть можливість перейти до виконавця (якщо точніше до власника альбому), а також відтворити всі наявні пісні виконавця, перейти до іншого альбому цього автора.

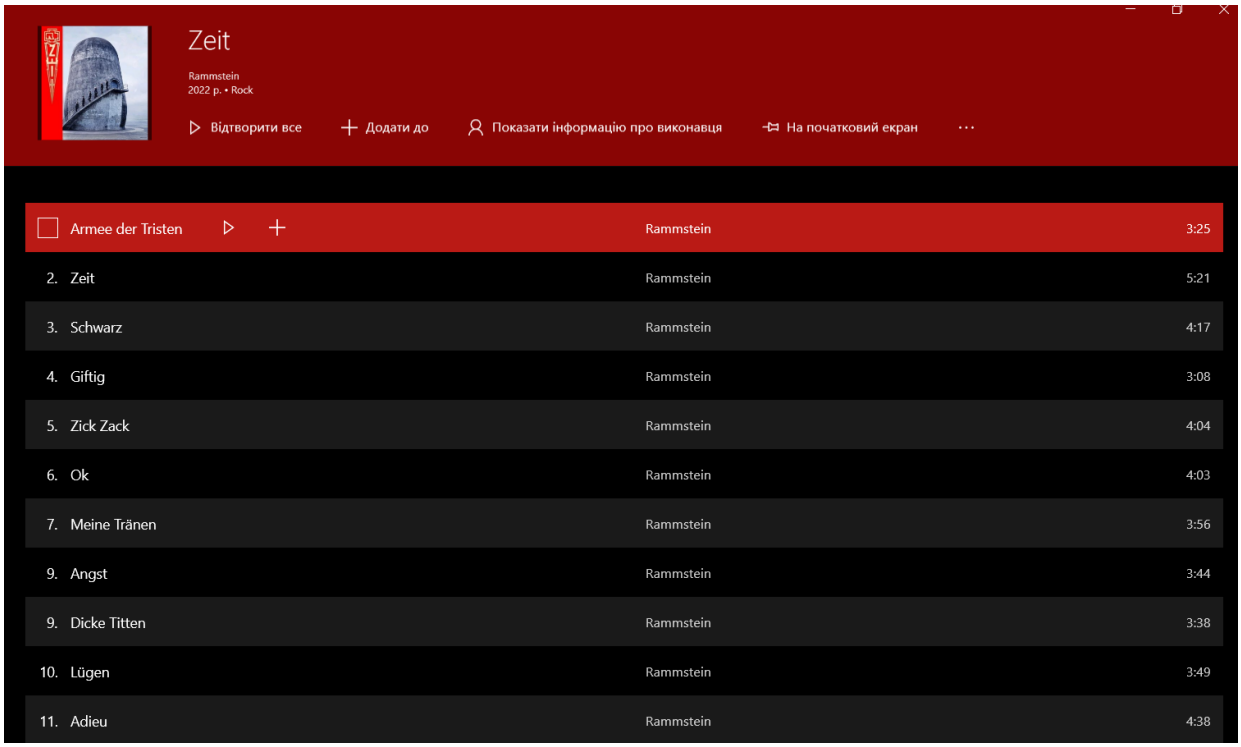

Рисунок 1.7 – Вкладка альбому Rammsteien (Zeit) у Groove Music

Редагування тегів майже повне, проте воно є необхідним та достатнім для швидкої організації списків відтворення та бібліотеки в цілому, тому нарікати на неповність недоречно. Інформація щодо тегів майже повністю відповідає даним про музику в AutomaTag, за деякими винятками. Ці винятки не надто важливі для рядового користувача.

Текст пісні, коментарі, кількість пісень в альбомі зазвичай просто пропускається, за непотрібністю. Враховуючи наведене вище, можна зробити висновок про корисність Groove Music та його конкурентоспроможність.

Після того, як було розглянуто предвстановлені застосунки, описано їх наповнення та виділено переваги а недоліки, варто розглянути сторонні аудіопрогравачі.

AIMP. Одним з найвідоміших варіантів заміни програвачів від Microsoft є AIMP (рис. 1.8 [3]). AIMP – це аудіоплеєр для ПК, розроблений українським програмістом.

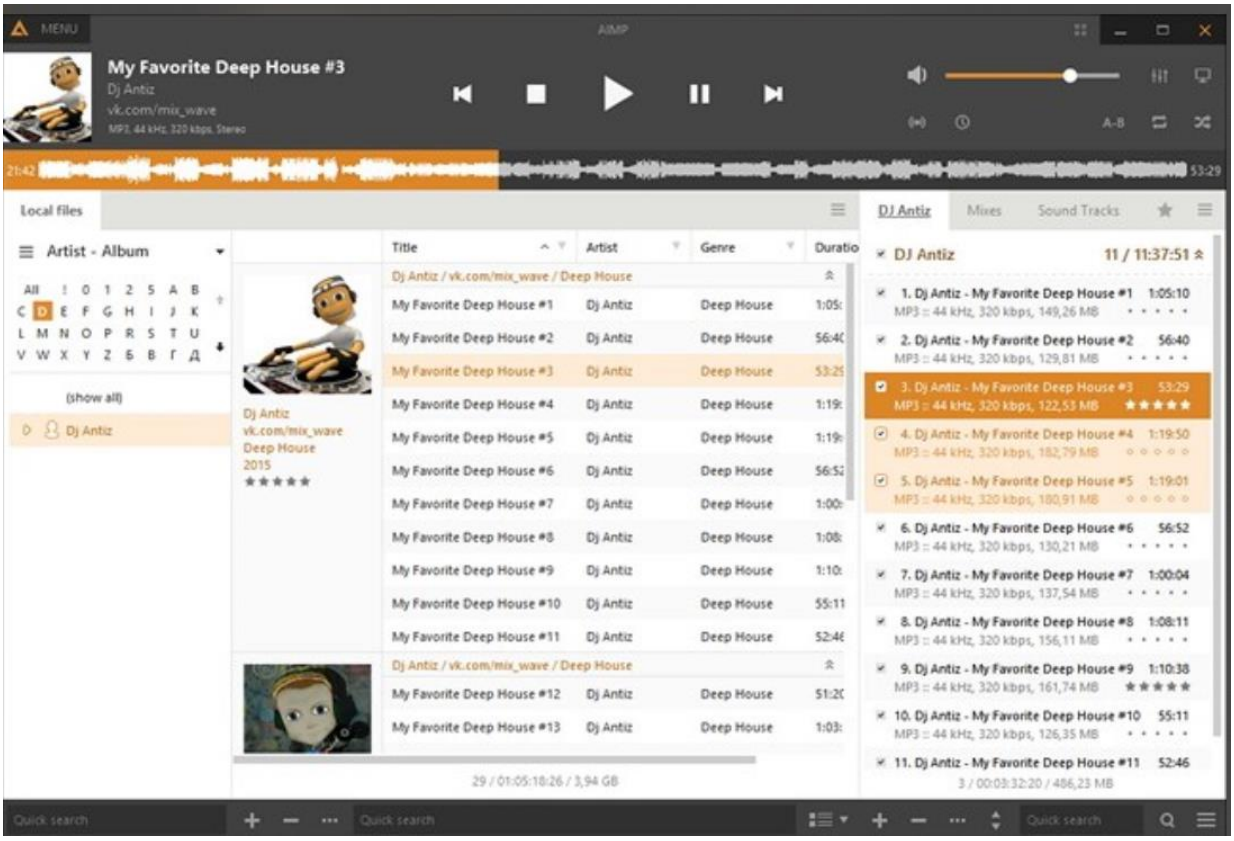

Рисунок 1.8 – Програвач AIMP

Цей плеєр має безкоштовну та платну версії, проте обидві версії мають багато корисних функцій, таких як підтримка багатьох форматів аудіофайлів, підтримка різних режимів відтворення, еквалайзер, можливість створення плейлистів, та багато іншого.

Переваги AIMP:

– підтримка багатьох форматів: AIMP підтримує багато форматів аудіофайлів, що дозволяє відтворювати майже будь-який файл;

– підтримка великої кількості музичних форматів: AIMP підтримує велику кількість аудіоформатів, таких як MP3, FLAC, OGG, WAV, WMA, AAC та багато інших;

– корисні функції: AIMP має багато корисних функцій, таких як еквалайзер, можливість створення плейлистів та інше, що дозволяє налаштувати плеєр під свої потреби;

– легкий інтерфейс: інтерфейс AIMP простий і легкий у використанні, що робить його зручним для користування навіть для початківців.

Недоліки AIMP:

– важкість: AIMP може виявитись дещо важким плеєром для комп’ютерів з меншою кількістю ресурсів, що може призвести до повільної роботи комп’ютера;

– брак підтримки деяких форматів: AIMP підтримує багато форматів аудіофайлів, але не всі. Наприклад, він не підтримує формат Apple Lossless (ALAC);

– відсутність оновлень: у деяких випадках може виникати проблема з роботою застосунка;

– створення плейлистів може викликати труднощі в початківців.

Billy. Billy (рис. 1.9 [4]) – це музичний плеєр для ПК з відкритим вихідним кодом, розроблений командою з Бразилії.

Ця програма дозволяє користувачеві насолоджуватися відтворенням музики з його комп’ютера, а також управляти музичними файлами і створювати власні плейлисти.

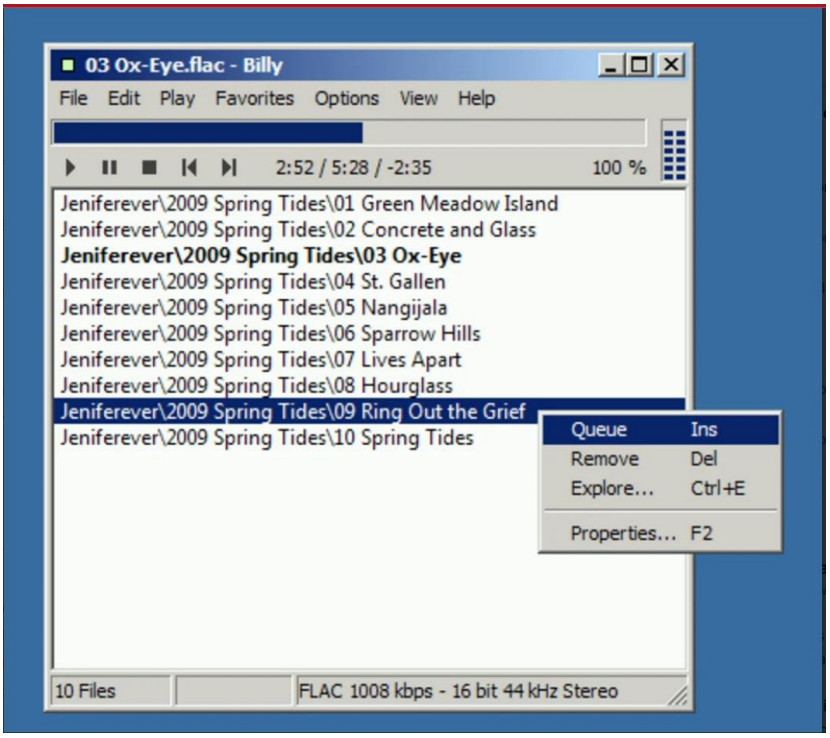

Рисунок 1.9 – Вигляд програвача Billy

Основні переваги Billy:

– безкоштовний та з відкритим вихідним кодом: Billy є програмою з відкритим вихідним кодом, що дозволяє користувачам змінювати та розповсюджувати її безкоштовно;

– простий інтерфейс: інтерфейс Billy дуже простий та легкий у використанні, що робить його зручним для користування навіть для початківців;

– наявність потужного еквалайзера: Billy має потужний 10-смуговий еквалайзер, що дозволяє користувачам налаштовувати звук на свій смак;

– підтримка багатьох форматів: Billy підтримує багато форматів аудіофайлів, що дозволяє відтворювати майже будь-який файл;

– наявність різноманітних налаштувань: Billy має багато налаштувань, що дозволяє користувачам налаштувати плеєр під свої потреби;

– мінімальний вплив на ресурси комп’ютера: Billy має низький рівень споживання системних ресурсів, що дозволяє користувачам працювати з іншими програмами без перебоїв.

Недоліки Billy: відсутність деяких додаткових функцій, Billy не має деяких додаткових функцій, таких як вбудована магазин музики або можливість редагування тегів.

1by1. 1by1 (рис. 1.10 [5]) – це мінімалістичний аудіоплеєр для ПК, розроблений для простого відтворення музичних файлів без складних функцій. Програма має дуже простий інтерфейс і займає дуже мало місця на жорсткому диску, що робить її ідеальним вибором для тих, хто шукає легкий та швидкий спосіб відтворення музики на своєму комп'ютері.

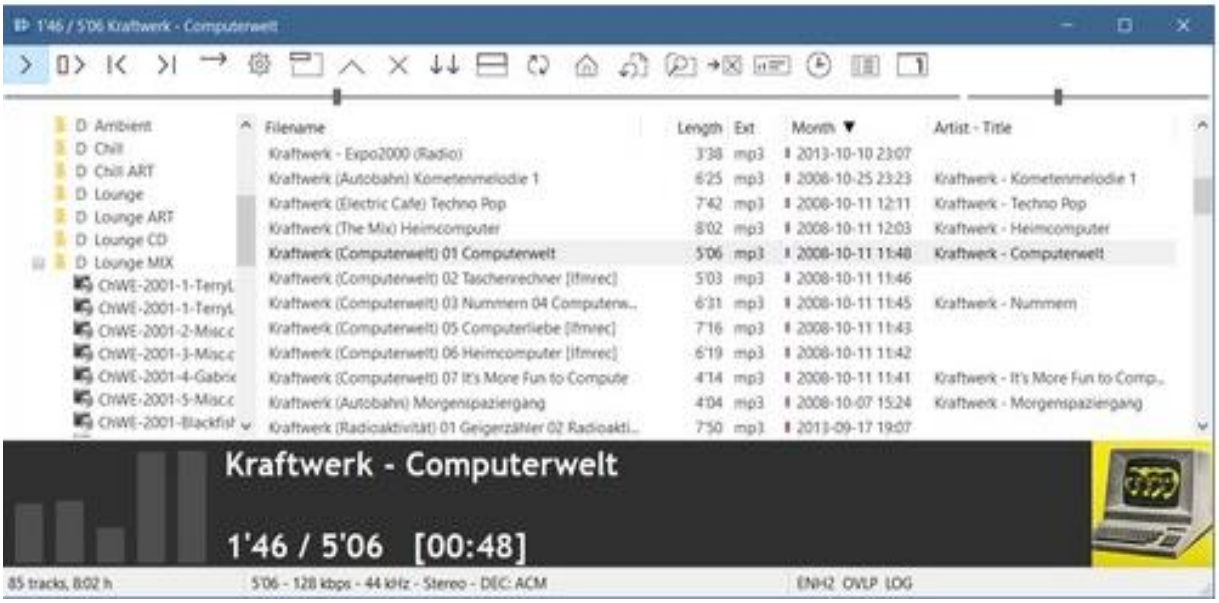

Рисунок 1.10 – Вигляд програвача 1by1

Основні переваги 1by1:

– простий інтерфейс: інтерфейс 1by1 дуже простий і легкий у використанні, що робить його зручним для користування для початківців;

– підтримка різних форматів: 1by1 підтримує багато різних форматів аудіофайлів, включаючи MP3, WAV, AAC та інші;

– не потребує інсталяції: 1by1 можна запустити без інсталяції на комп'ютері, просто розпакувавши архів з програмою;

– різноманітні налаштування: 1by1 має багато налаштувань, що дозволяє користувачам налаштувати плеєр під свої потреби;

– мінімальний вплив на ресурси комп’ютера: 1by1 має дуже маленький розмір та низький рівень споживання системних ресурсів, що дозволяє користувачам працювати з іншими програмами без перебоїв;

– відтворення папок і плейлистів: 1by1 може відтворювати не тільки окремі файли, але й цілі папки та плейлисти.

Недоліки 1by1:

– відсутність підтримки радіо та потокової музики: 1by1 не підтримує відтворення радіостанцій або потокової музики;

– можуть виникати проблеми при встановленні.

Виходячи з наведених прикладів застосунків аудіопрогравача, їх переваг та недоліків зроблено висновок щодо стану розвитку аудіопрогравачів для ПК у світі.

Рівень наповнення, як функціонального, так і візуального, відрізняється від застосунку до застосунку, проте більшість основних можливостей, таких як відтворення музики або формування плейлистів, надаються користувачу кожним з перерахованих вище варіантів програмного інструменту аудіопрогравача. Деякі із застосунків можуть повною мірою задовольнити навіть великих поціновувачів музики, пропонуючи широкий набір аудіоефектів та додаткових функцій. Інші ж призначенні для повсякденного використання звичайними пересічними користувачами, задовольняючи базові вимоги для програмного забезпечення такого типу.

1.2 Аналіз літературних джерел щодо існуючих підходів моделювання та реалізації мобільних застосунків MP3-плеєра

Задля правильної розробки програмного застосунку аудіопрогравача було проведено роботу з дослідження та вивчення великої кількості актуальної наукової літератури. В цьому підрозділі наведені та описані результати аналізу літературних джерел відповідно до теми роботи.

Літературне джерело [6] описує відмінності між великими публічними та персональними музикальними бібліотеками, а також проблеми, пов’язані з веденням і підтримкою власної музикальної колекції. Були проаналізовані методи доступу та організації, що використовуються для керування особистою аудіотекою. Проаналізовано та наведено правила поведінки користувача при веденні персональної цифрової музичної бібліотеки.

Наступна робота [7] частково доповнює дослідження попереднього літературного джерела. Проте більшу увагу приділено поведінці користувачів стрімінгових платформ та сервісів організації даних. Тут наголошується про те, що, не зважаючи на цифровий формат пісень, вони не втрачають своєї колекційності. Мається на увазі особистий підхід до організації аудіотеки, приклади якого можуть бути впроваджені в програмні інструменти аудіопрогравача. Та все ж увагу приділено більше важливості курування музичною бібліотекою з точки зору емоційності користувача.

Літературне джерело [8] описує розробку та програмну реалізацію застосунку для пошуку та підбору музики в замкненому каталозі. Програма дозволяє відтворювати музичні файли та надає базовий функціонал аудіокатологу з програванням музики. Програмний інструмент дозволяє отримувати статистичні дані про прослуховування.

Була досліджена стаття [9] в якій досліджувався пошук пісень за акустичним вводом. Розроблена система здійснює підбор музики за відповідністю тексту пісні, проспіваної користувачем. Експериментальними дослідженнями було перевірено, наскільки точно люди співають відомі пісні.

Також досліджено статтю [10]. Метою цієї статті є повідомлення про останні досягнення у спільному проєкті, спрямованому на розробку контент-орієнтованих методів пошуку музичної інформації (MIR) як альтернативи стандартним текстовим режимам доступу до цифрових музичних бібліотек. У документі описуються поточні практики та поточні дослідження, а також обговорюються потенційні застосування для майбутнього використання.

Аналогічно попереднім джерелам досліджено статтю [11] Через невдоволення наявним на даний момент програмним забезпеченням на той час автори створили програму управління музикою для великих цифрових музичних бібліотек. Вони пропонують систему, яка вирішує деякі труднощі організації цих бібліотек. Система вводить новий стиль візуальної взаємодії з музичною колекцією користувача. Фізична система в поєднанні з жанровою інформацією альбому дозволяє користувачеві просторово впорядковувати свою музику на екрані та робити вибір.

Джерело [12] описує складну систему, котра поєднує різні типи користувачів, будуючи граф знань для музичних композицій, репетиційних записів, партитур диригентів та ін. Виконавці та слухачі можуть бачити контекстну інформацію з анотацій вчених, а вчені, навпаки, отримують інформацію з поведінки інших користувачів та соціальних дискусій. Всі дані представлені у вільному доступі з подальшими використанням поза межами проєкту, інтерпретацією та подальшим взаємозв’язком.

Досліджена стаття [13] присвячена алгоритмам класифікації на основі вмісту, або іншими словами, алгоритми, які обробляють результати аналізу музичних особливостей музики, такі як темп або середня гучність. Основна мета цього дослідження полягає в тому, щоб побачити, наскільки ефективними стали алгоритми з точки зору класифікації музики. Ця проблема вирішується за допомогою декількох алгоритмів, які аналізують гармоніки, закономірності, стилістику музики (темп, біти тощо) та багато іншого, щоб класифікувати їх на різні жанри.

Стаття [14] присвячена проблемі автоматичного тегування аудіофайлів. Рішення, запропоновані авторами дозволяють нівелювати завади, з яким стикаються користувачі, такі як: неправильне тегування необізнаними в музичній сфері людьми, або розпізнання невідомої музики. Надані два варіанти вирішення вище згаданих проблем.

Джерело [15] висвітлює процес розробки застосунку музичного програвача та актуальність даної ніші у розробці застосунків цього типу.

Важливість розробки підкреслюється великою популярністю мулики в повсякденному житті. Автор висуває основні вимого до програмного застосунку плеєра такі як: функції відтворення аудіофайла, перехід до попереднього або наступного файлу в черзі, функції паузи/продовження програвання, можливість створювати та програвати плейлисти. Виділено можливості та функції існуючих мобільних застосунків і відсортовано потрібні від непрактичних або дорогих. Автор описує етапи розробки та підкреслює їх гнучкість, що корисно впливає на роботу у випадку зміни побажань клієнта.

Після ретельного дослідження різноманітного літературного матеріалу, зроблено висновки, щодо актуальності даної роботи. Деякі з розглянутих методів частково, або повністю застосовуються у створенні програмних застосунків програвача музики. Також певні дослідження несуть інформацію про фактори, котрі хоч і не обов’язково, можуть бути взяті до уваги.

1.3 Постановка задачі

Таким чином організація музичної бібліотеки та комфортне відтворення її вмісту є актуальним завданням на теперішній час. Тому ставиться задача проєктування та програмної реалізації застосунка MP3-плеєра для відтворення та організації музичних файлів офлайн.

Об’єктом роботи є набір папок з файлами на персональному комп’ютері та аудіофайли у форматі .mp3, що містяться в них.

Метою роботи є розробка застосунку MP3-плеєра для відтворення та організації музичних файлів офлайн, який дозволятиме користувачу організовувати свою власну аудіотеку та комфортно відтворювати аудіофайли.

Для досягнення мети необхідно вирішити такі питання:

– провести аналіз існуючих аналогів;

– спроєктувати структуру та наповнення застосунка;

– реалізувати застосунок MP3-плеєра для відтворення та організації музичних файлів офлайн.

2 МОДЕЛЮВАННЯ СТРУКТУРИ МОБІЛЬНОГО ЗАСТОСУНКУ МР3-ПЛЕЄРА ДЛЯ ВІДТВОРЕННЯ ТА ОРГАНІЗАЦІЇ МУЗИЧНИХ ФАЙЛІВ ОФЛАЙН

2.1 Застосування методів системного аналізу до предметної області «Музика»

Методи системного аналізу можуть бути застосовані до предметної області «Музика» для розуміння та аналізу музичних систем, їх взаємодії та впливу на суспільство. Музична система може являти собою не лише якийсь застосунок або сервіс для відтворення музики, а також може бути сферою ринку, навчання, витвором мистецтва. Будь-яка група людей, що виконує певну композицію теж є система, кожен елемент якої має свої відповідні функції. Далі буде розглянуто деякі приклади застосування системного аналізу для музичних систем:

Аналіз музичного ринку: застосування системного аналізу дозволить вивчити музичний ринок як систему, що складається з різних елементів, таких як музичні виконавці, лейбли, продюсери, рекламні кампанії та інше. Для подібної задачі чудово підходить метод дерева цілей і подібні йому.

Метод дерева цілей був розроблений спеціально для потреб системного аналізу і наразі є домінуючим методом [16-20]. З математичної точки зору, таке дерево має вигляд зв’язного графа, вершини якого позначаються, як цілі, а дуги – зв’язки між цими цілями. Таке «дерево» створюється шляхом поділу загальної цілі на підцілі, які стають більш детальними компонентами, які працюють на нижчих рівнях і з деякого моменту є функціями. В наслідок таких дій формується повна і відносно стабільна структура цілей та проблем. З часом вона майже не піддається неминучим змінам, котрі мають місце у випадках динамічних систем [21-26].

Систему «Ринок музики» можна поділити різним способами на відгалуження, що пов’язані між собою і мають свої гілки. Така система надто

складна для описання та дослідження у повному обсязі, тому варто робити розбиття на підсистеми за рівнями і займатися вже їх дослідженням.

У даному випадку потрібні експертні опитування, дискусії та дибати з приводу гіпотетичного покращення такої масштабної системи. Це дозволить визначити основні фактори, що впливають на ринок, тенденції розвитку та прогнозувати зміни. Проте велику роль відіграє саме людський фактор і неконтрольовані зміни в одному аспекті можуть потягнути за собою каскад реформувань системи. Таким чином, система повинна буде знову проаналізована і очікуватиме на впровадження покращень.

Невід'ємною частиною дослідження є методи економічного аналізу. Вони вирішують деяку кількість завдань на яку перетворюється система.

Аналіз музичної індустрії: Системний аналіз допоможе вивчити взаємодію музичних компаній, співробітництво виконавців та продюсерів, розподіл прибутків між учасниками ринку та інші фактори, що впливають на музичну індустрію. Як і у попередньому прикладі можна музичну індустрію можна поділити на багато систем, які в свою чергу розділені на все менші підсистеми. Аналогічним чином, будуються зв'язки між системами. Постійні незначні зміни також відбуваються, проте мають не значний вплив на загальний вигляд системи. Великі неконтрольовані зміни здебільшого є прогнозованими.

При прогнозуванні майбутніх умов розвитку можуть використовуватися різновиди статистичного аналізу, який найбільшою мірою підходить для передбачення трендів, що займає ледь не ключове місце на ринку музики та індустрії в цілому.

Аналіз музичного творчості: Системний аналіз дозволить вивчити музичний твір як систему, що складається з різних елементів, таких як мелодія, гармонія, ритм, текст та інше. Це дозволить визначити основні стилі та жанри музики, виявити тенденції розвитку та вплив творів на суспільство.

Виділяються частоти, в яких звучить голос або той чи інший інструмент. Важливо, щоб кожен звук підпорядковувався певним законам та правилам з точки зору ритму, гучності і музики, як науки взагалі.

Аналіз музичної освіти: Системний аналіз може бути застосований для аналізу музичної освіти як системи, що складається з різних елементів, таких як викладачі, студенти, методи навчання, програми та інше. Це дозволить визначити проблеми та переваги системи музичної освіти та запропонувати шляхи її покращення або реформації. Кожен клас в музичній чи то звичайній школі може бути поділений на окремі компоненти, пов’язані ієрархічними зв’язками. Кожна програма навчання являє собою певний сценарій, пункти якого протягом вивчення відповідного матеріалу приєднуються шар за шаром до загального обсягу знань учня.

Програмне забезпечення це теж система, основна ціль якої – створення або відтворення музики, аналогічно поділяється на підцілі та функції. Кожен інструмент управління повинен чітко виконувати свою функцію, при цьому не шкодячи іншим компонентам. Тут можна використовувати, як методи дерев та і методи записної книжки, тобто виділення всіх існуючих фактів, які стосуються досліджуваної проблеми, побічних завад, що мають безпосередній вплив на виконання задачі або виникають під час її виконання та подальше обмірковування можливих шляхів вирішення проблем, нівеляція шкідливих факторів або захист від них [27-32].

Даний метод можна використовувати як для частини, так і для всієї системи в цілому. Щоправда у другому випадку описання всіх можливих чинників, побічних проблем та вирішень може бути занадто громіздким та складним. Саме тому бажано уникати поділу на підзадачі в рамках описання головної проблеми і виділяти такі підзадачі в окремі «записні книжки» і поступово займатися їх рішенням.

У подібному випадку, як і у деяких інших описаних вище, може стати у нагоді метод сценаріїв. Цей метод є одним з експертних методів прогнозування, його суть полягає письмовому описанні покрокової логічної

послідовності розвитку досліджуваної системи. Даний можливий розвиток заснований на імовірнісних припущеннях, щодо стану різних факторів, котрі допомагають спрогнозувати можливі стан та поведінку системи.

Для такого можна використовувати, наприклад діаграми послідовності, котрі відображатимуть кроки виконання певних задач, а також взаємодію між компонентами системи або іншими системами.

Музичне обладнання: Описання всіх елементів також потребує чіткої структурованості і порядку. Кожна частина приладу представлена для розуміння спеціалістів у вигляді схем, діаграм та креслень, котрі дають чітке розуміння роботи до найменших подробиць.

Зважаючи на всі вищезгадані приклади з предметної області «Музика», зроблено висновок, що різні методи системного аналізу можуть бути застосовані до певних сфер даної області. В основному будуть корисні методи дерев цілей, сценаріїв, статистичні та методи економічного аналізу. Залежно від конкретних випадків методи з даних груп виявляться надзвичайно корисними. Також методи можна поєднувати.

2.2 Особливості бази даних мобільного застосунку MP3-плеєра для відтворення та організації музичних файлів офлайн

Через очевидне спрощення представлення даних про композицію у вигляді аудіофайлу, навіть приблизне уявлення про бази даних застосунків програвача для пересічного користувача у більшості випадків може виявитися хибним.

Із точки зору розробника, програміста або більш-менш користувача застосунку програвача все набагато простіше. Щоб правильно систематизувати музичну бібліотеку, достатньо розуміти теги аудіофайлу, про не раз згадувалося у підрозділі 1.1.

Саме завдяки тегам користувачі бачать гарні колекції музики у аудіаплеєрах на ПК та мобільних пристроях.

Основними тегами музичного файлу є Назва (Title), Автор (Artist), Назва альбому (Album name), Жанр (Genre). Ці теги найчастіше можна побачити в доступних для редагування. Особливо це твердження стосується мобільних застосунків музичного програвача. Комп'ютерні аналоги можуть похизуватися більшими можливостями для редагування, а як наслідок для організації. Названі теги власне дозволяють бачити назву, автора, альбому та жанр композицій. Також ці теги дозволяють навігацію по застосунку, тобто знайти композицію по ключовому слову, отримати всі пісні виконавця тощо. Далі буде розглянуто всі теги аудіофайлу, котрі є необхідними та достатніми для програмної організації аудіотеки.

Теги:

– Title (Назва) – зберігає назву композиції;

– Artist/Performer (Автор) – зберігає автора(ів) ДАНОЇ композиції;

– Album (Альбом) – назва альбому, до якого належить композиція;

– Genre (Жанр) – містить жанр композиції;

– Track number (Номер доріжки) – порядковий номер композиції в альбомі;

– Track Count (Кількість доріжок) – кількість пісень в альбомі;

– Disc number (Номер диску) – номер диску у багатодисковому альбомі;

– Disc Count (Кількість дисків) – кількість дисків у багатодисковому альбомі;

– Album Artist (Виконавець альбому) – один з найважливіших тегів, зазвичай прихованих від користувача. Саме він відповідає за створення альбомів на ряду з назвою та кількістю пісень;

– Lyrics (Текст) – тут міститься текст пісні, якщо автор додає його при створенні файлу;

– Comment (коментарі).

Останні два теги зі списку не часто використовуються навіть попри свою інформативність. Текст часто навіть не додають самі виконавці, автоматичні редактори тегів теж не завжди знаходять цю інформацію.

Щодо коментарів… зазвичай це поле лишається пустим. За рідким винятком там може розміщатися реклама чи якась інша сміттєва інформація, що могла бути випадково занесена в тег під час створення або редагування.

Інші теги можуть бути використані для створення бази даних для застосунку, або навіть дати можливість програмно реалізувати організацію й інші функції в обхід використання БД. Щоб показати повні можливості тегів, буде представлено приклад можливої реалізації бази даних.

Залежно від цілей застосунку і побажань замовника кількість таблиць може відрізнятися. Наприклад можна часто зустріти функцію виводу результатів пошуку по жанру. Дана операція має сенс, якщо всі музичні файли в бібліотеці попередньо відформатовані (хоча б для цього поля). Проте, зазвичай, дане поле має або пусті значення, або сміттєві дані. Також може бути надзвичайно велика кількість жанрів, поєднань тощо. Навіть для предметної області «Музика» не існує повної класифікації за жанрами. Тому дана функція може бути опущена, оскільки має сенс лише у випадку повного форматування поля жанр власноруч самим користувачем відповідно до правил БД.

Загальний вигляд бази даних для застосунку музичного програвача може мати вигляд, зображений на рисунку 2.1.

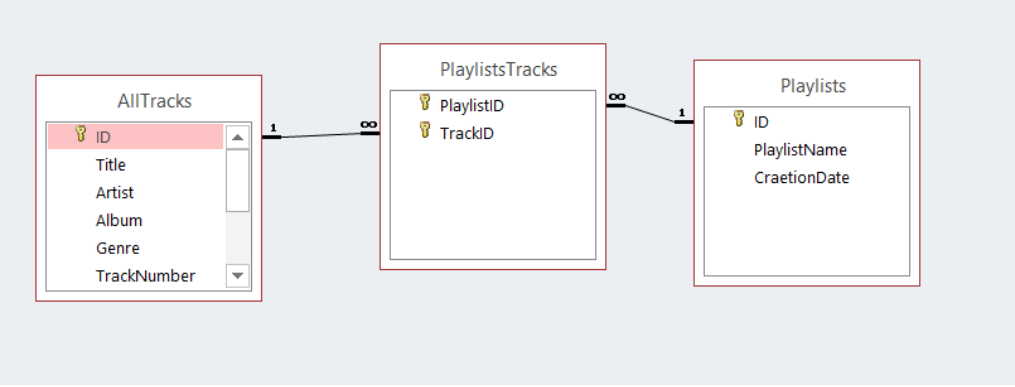

Рисунок 2.1 – Представлення можливого вигляду БД для застосунку аудіопрогравача

Як можна дізнатися з рисунку, за відображення, збереження інформації про пісні відповідає таблиця AllTracks (всі пісні).

Ця таблиця є головною в наведеній базі даних. Саме вона буде використовуватися постійно, для відображення, подальшого відтворення, або інших маніпуляцій з бібліотекою. Інші таблиці, хоч і теж зберігають важливу інформацію, проте використовуватимуться для інших функцій, наприклад пошуку або створення плейлистів.

Всі таблиці пов’язані зв’язком «один до багатьох». Цей зв’язок передбачає, що одному рядку з батьківської таблиці може відповідати багато рядків з таблиці-нащадку. Тобто один виконавець може мати безліч пісень.

У процесі розробки було створено три таблиці. Така маленька кількість таблиць і порушення деяких правил реляційних БД зумовлена даними, які використовуються у застосунку. Сама інформативність музичного файлу виключає потребу у створенні складної схеми даних. Також виникає проблема використання складної схеми даних при додаванні або видаленні пісні з бібліотеки, адже, наприклад, якщо виділити окрему таблицю виконавців і брати інформацію з неї в таблицю пісень, то при додаванні нової пісні в бібліотеку потрібно спочатку додавати виконавця, а потім і саму пісню. Аналогічно виникають складнощі при видаленні. Подібна схема БД може бути застосована для програвачів аудіо, де всі виконавці, чиї пісні доступні для завантаження/додавання у власну бібліотеку, спочатку реєструються власноруч, власне, як виконавці й інформація про них підтягується в БД.

У даному ж випадку, база даних потрібна лише для організації плейлистів. Оскільки програмне створення і подальша підтримка функціонування списків відтворення займає багато часу, ресурсів пристрою, та часу виконання, простіше просто зберігати списки відтворення у вигляді простої БД, яка зберігатиме інформацію про присутні треки в бібліотеці, власне, списки відтворення, та об’єднуючу таблицю, котра міститиме інформацію, про вміст кожного з плейлистів.

Далі буде розглянуто кожну таблицю більш детально.

Перша таблиця – AllTracks (табл. 2.1).

Таблиця 2.1 – Таблиця AllTracks БД застосунку МП3 програвача на ПК

Data	Type	Example
ID	Integer (counter)	15
Title	String	FUKUSHIMA
Artist	String	ZOMBIEZ
Album	String	GOTT IST TOT
Genre	String	Hip-hop
TrackNumber	Integer	7
TrackCount	Integer	16
DiscNumber	Integer	1
DiscCount	Integer	1
AlbumArtist	String	ZOMBIEZ
PathToTrack	String	/storage/emulated/0/Download/ ZOMBIEZ – FUKUSHIMA.mp3

Ця таблиця містить основну інформацію про композицію. Вона потрібна для відображення вмісту бібліотеки, сортування, відображення альбомів, виконавців, вмісту альбомів, всіх пісень певного виконавця чи жанру. Майже вся реалізація згаданих вище функцій полягає у маніпуляції з умовами для виведення інформації. Вона не потребує складної схеми даних для роботи, адже таке представлення може значно ускладнити і сповільнити, як реалізацію, так і подальшу роботу застосунка.

Більшість полів в таблиці AllTracks просто містять інформацію тегів музичного файлу. Поля виконавців, назви, альбомів та жанрів використовуються для відображення, сортування, та формування груп за виконавцями, альбомами, жанрами. Інші поля мають цінність у формуванні

альбомів при розділенні композиції за альбомами, виконавцями альбомів і кількістю пісень, а також сортуванні цих пісень за їх номером.

Наступна таблиця – Playlists (табл. 2.2).

Таблиця 2.2 – Таблиця Playlists БД застосунку MP3 програвача на ПК

Data	**Type**	**Example**
ID	Integer	2
PlaylistName	String	Chill Vibes
CreationDate	Date	26.12.2022

Ця таблиця зберігає назви всіх плейлистів та дату їх створення.

Вміст списків відтворення міститься у наступній таблиці, поєднуючи дані з таблиць Playlists та AllTracks.

Таблиця PlaylistsTracks (табл. 2.3).

Таблиця 2.3 – Таблиця PlaylistsTracks БД застосунку MP3 програвача на ПК

Data	Type	Example
PlaylistID	Integer	7
TrackID	Integer	378

Саме з використанням цієї таблиці відбувається об’єднання даних з двох попередніх таблиць. Обидва поля є первинними ключами. Вони містять ідентифікатори пісень і плейлистів. Таким чином не можна додати одну і ту ж саму композицію в список відтворення. Крім того, при фізичному видаленні файлу з пристрою, дані про нього видаляються з головної таблиці а також з таблиці PlaylistsTracks. Це запобігає виникненню помилок з відкриттям неіснуючого файлу.

Використання такої схеми даних є як перевагою, так і недоліком: з одного боку можна додавати нові композиції до бібліотеки і потім редагувати

інформацію про них, а з іншого отримується багато сміттєвої інформації, котра потребує редагування.

Для впровадження БД в застосунок використовуватиметься база даних SQLite, адже для її використання не потрібно мати окремий сервер, звіряти версії серверів розробника та користувача та ін.

2.3 Моделювання структури та наповнення мобільного застосунку MP3-плеєра для відтворення та організації музичних файлів офлайн

У процесі моделювання будь-якої системи необхідним є прорахунок всіх сценаріїв, котрі можуть бути виконані користувачем. Потрібно взяти до уваги можливу необізнаність користувача у програмних інструментах даної сфери. Важливо передбачити різні розвитку подій. Цими сценаріями може керуватися розробник при проектуванні системи.

Рисунок 2.2 демонструє діаграму прецедентів системи MP3 програвача офлайн. На рисунку зображені загальні задачі, які повинен виконувати користувач.

Як видно з рисунку система має лише одну роль – користувача, котрий виконує різні завдання. Послідовність цих завдань може бути різною, а деякі, як «відтворення музичного файлу» можуть бути виконанні, як і при завантаженні програми, так і після маніпуляцій з бібліотекою (наприклад після пошуку всіх пісень виконавця Killstation, користувач зможе відтворити будь-яку з наявних пісень цього автора).

Користувач має можливість здійснювати пошук, переходити між вкладками «композиції», «виконавці», «альбоми», вибирати послідовність відображення пісень, відтворювати обрану пісню, перемикатися між піснями вгору та вниз по списку, зупиняти та продовжувати відтворення.

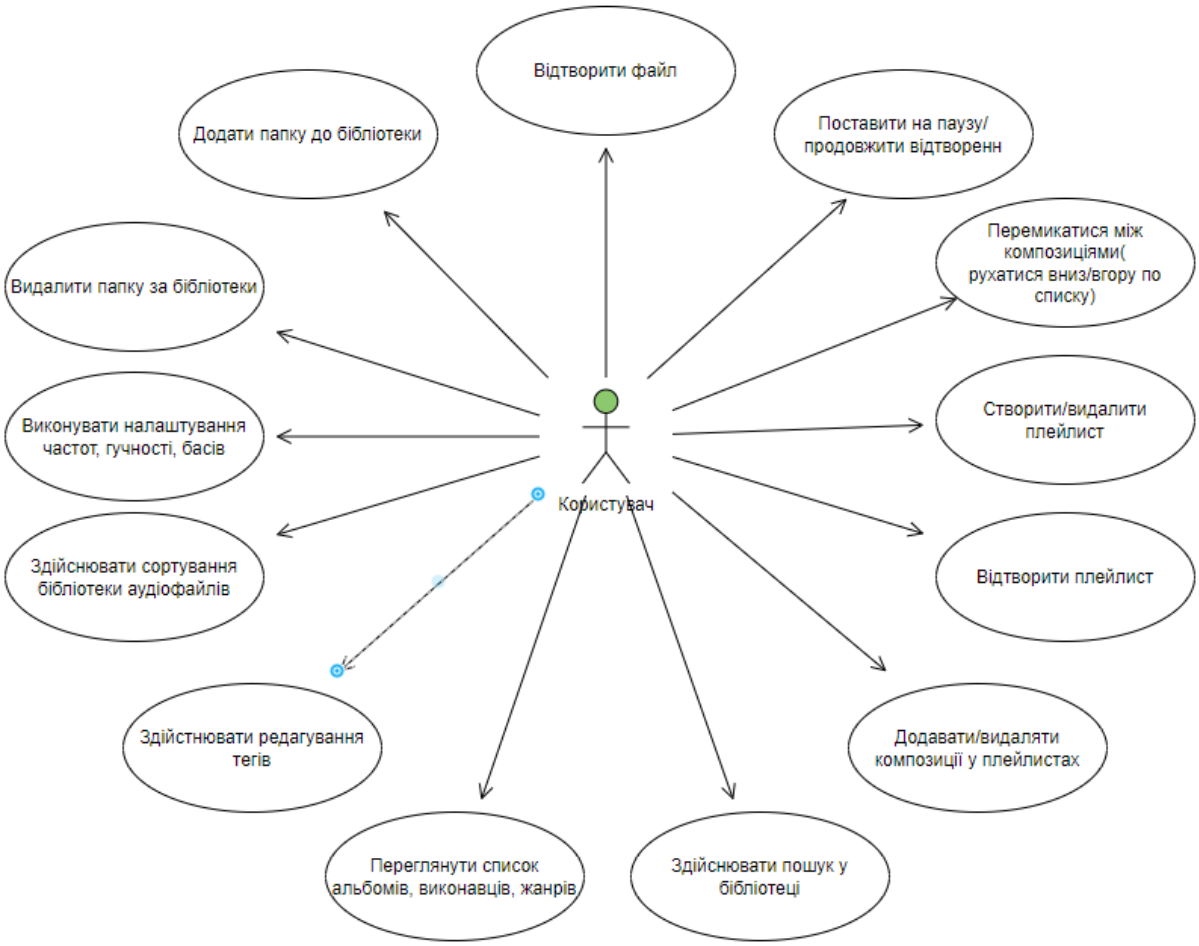

Рисунок 2.2 – Діаграма прецедентів системи MP3 програвача

Також користувач в змозі створювати або видаляти списки відтворення, додавати, видаляти пісні з них та міняти порядок відтворення композицій у плейлисті.

Для користувача доступні функції редагування тегів файлу та еквалайзер (вирівнювач), що дозволяє налаштувати гучність, підсилити або зменшити баси й відрегулювати частоти.

Оскільки це офлайн застосунок, то система не має адміністратора, а база даних є закритою, що користувач не мав змоги вносити якісь зміни, що може призвести до некоректної роботи застосунка.

Не менш важливим етапом у проектуванні є створення діаграми послідовності, котра демонструватиме взаємодію користувача та компонентів системи.

На цій діаграмі (рис. 2.3) показана можлива послідовність дій, які виконує користувач під час використання застосунка.

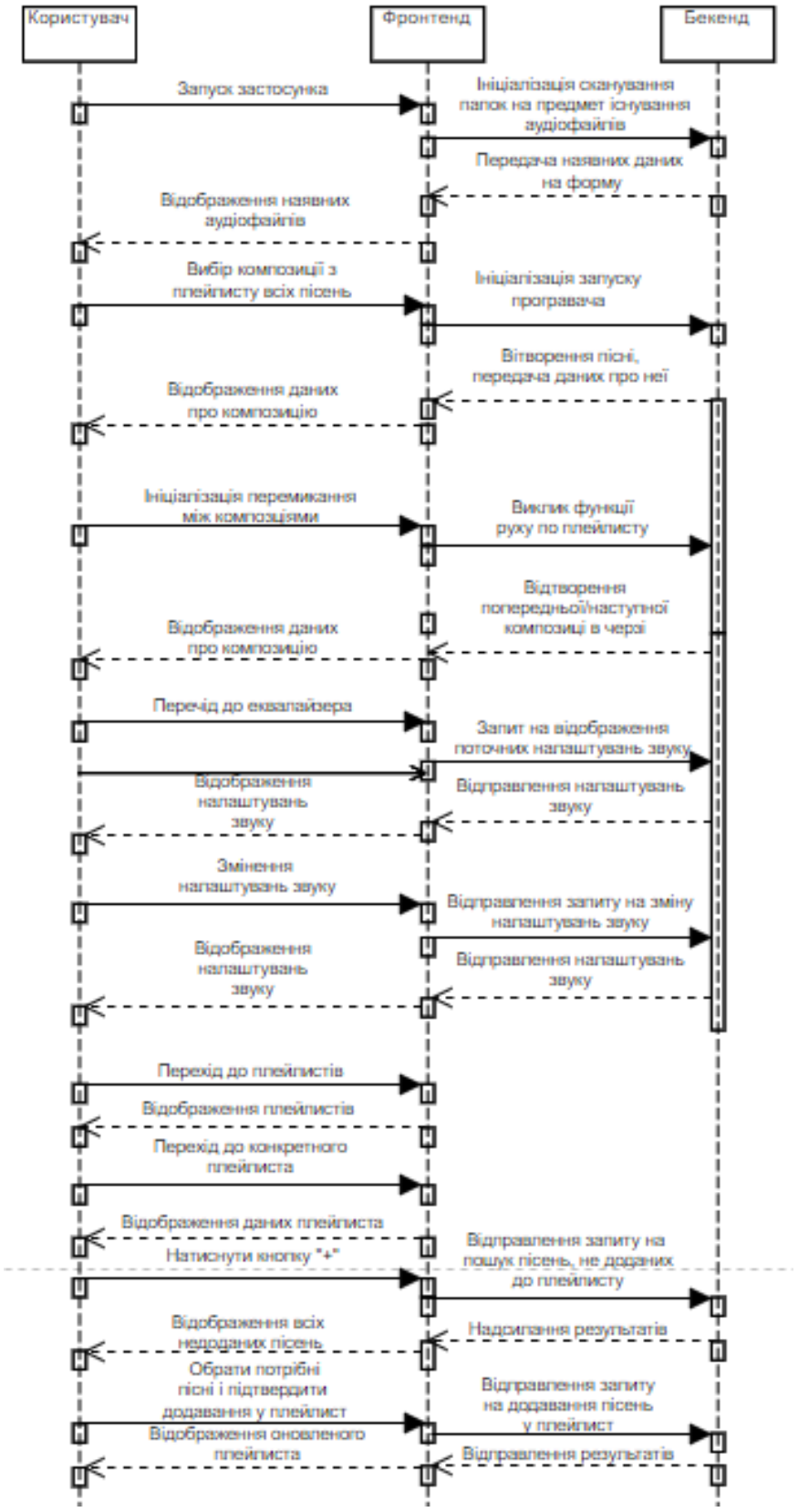

Рисунок 2.3 – Діаграма послідовностей

В процесі розробки діаграми було прийнято рішення про відображення послідовностей взаємодії користувача з графічним інтерфейсом застосунка (Frontend) та контролером (Backend) – частиною проєкту, що реалізує і виконує всі функції, приймає, зберігає та повертає дані.

Дії, продемонстровані на діаграмі (рис. 2.3), являють собою візуалізацію кроків користувача, а саме: ініціалізацію застосунка, ініціалізацію деяких функцій інтерфейсом, запити до бази даних, маніпуляції з композиціями такі, як відтворення обраної пісні, руху списку до попередньої/наступної пісні. Також відображено виклик користувачем еквалайзера та встановлення налаштувань звуку.

Останніми кроками є перехід до плейлистів, вибір конкретного списку відтворення та додавання пісень до обраного плейлиста шляхом пошуку ще не доданих пісень.

Варто зауважити, що деякі процеси можуть відбуватися паралельно. Наприклад, користувач може створювати списки відтворення, здійснювати пошук, або, що більш можливо встановлювати свої налаштування звуку під час відтворення обраної раніше черги пісень.

Виходячи з вище описаних схем, наведених в процесі проектування, можна зробити висновки стосовно задач, які має виконувати розроблюваний застосунок.

3 РОЗРОБЛЕННЯ ВЕБЗАСТОСУНКУ ЗАСТОСУНКУ МР3-ПЛЕЄРА ДЛЯ ВІДТВОРЕННЯ ТА ОРГАНІЗАЦІЇ МУЗИЧНИХ ФАЙЛІВ ОФЛАЙН

3.1 Вибір інструментальних засобів для реалізації поставленої задачі

У процесі дослідження засобів створення застосунків в обраній предметній області було виявлено велике різноманіття інструментів розробки. Розвиток програмного забезпечення аудіопрогравачів стимулює розвиток як середовищ розробки, так і додаткових інструментів, створених спеціалістами-ентузіастами і великими компаніями.

Для створення застосунків відтворення аудіо існує досить широкий вибір мов програмування. Серед них: Java, C++, C#, Python, Kotlin та інші Наведені мови програмування часто дозволяють створювати повноцінні застосунки для ПК, кросплатформенні програми, які підтримуються на операційних системах Windows, Android, IOS і програми для браузерів. Переважна більшість мов та методів програмування дозволяють поєднувати елементи, написані різними мовами, наприклад: при розробці ПЗ на C# можна використовувати Visual Basic та F#.

Для реалізації поставленої задачі було обрано середовище розробки Microsoft Visual Studio 2019.

Microsoft Visual Studio – серія продуктів компанії Майкрософт, які містять інтегроване середовище розробки програмного забезпечення та низку інших інструментальних засобів. Ці продукти дають змогу розробляти як консольні програми, так і програми з графічним інтерфейсом, включно з підтримкою технології Windows Forms, а також вебсайти, вебзастосунки, вебслужби як у рідному, так і в керованому кодах для всіх платформ.

Перевагою версії 2019 року є більша стабільність деяких компонентів, котрі ще повністю або частково не адаптовані для версії 2022 року.

Для розробки застосунка офлайн MP3 програвача було обрано пакет «Розробка класичних застосунків .NET». Даний інструментарій дозволяє створювати програмне забезпечення на мові C# та VB. Він надає можливість розробнику створити консольні проєкти, проєкти Windows Forms, WPF та інше.

Для розробки першої версії застосунка плеєра було обрано проєкт Windows Forms. Не зважаючи на обмежений функціонал, цей варіант є достатньо непоганим. Дизайн можна перероблювати під цілі застосунка, а відображення даних дозволено реалізувати різноманітними способами.

Також зручність використання автоматично створюваних структур проєкту в Visual Studio, таких, як Windows Forms Application, є те, що можна реалізувати окремо дизайн, а окремо основний функціонал застосунка. Таким чином розробник може створити окрему реалізацію функцій, яку потім можна перенести на новий дизайн, наприклад у проєкт WPF.

Великою перевагою описуваної середи розробки є можливість створювати посилання на інші проєкти, або підключати сторонні бібліотеки. Такими бібліотеками в даному випадку є NAudio і TagLib-sharp.

NAudio[] – бібліотека, яка дозволяє працювати з аудіофайлами різних розширень. Інструментарій цієї бібліотеки надає можливість створювати аудіоефекти, працювати з аудіопотоками, має клас Player, методи якого допомагають керувати відтворенням.

TagLib-sharp[] – бібліотека, створена для мови C#, уможливлює роботу з тегами аудіофайлів через код програми. Вона дозволяє отримати будь-яку інформацію з файлу: назву пісні, автора, назву альбому, автора альбому тощо. Здебільшого за допомогою цього інструмента вдається налаштувати організацію музичної колекції.

Для виведення інформації й організації списків відтворення використовуватиметься БД SQLite. SQLite – полегшена реляційна система управління базами даних. Вона втілена у вигляді бібліотеки. Перевагами цієї БД є відсутність потреби сервера на пристрої. Вона надає бібліотеку, з якою

програма компілюється, що спрощує програму, часові затрати і не потребує оновлення або навпаки встановлення старіших версій серверу. Другим же не менш важливим пунктом в перевагах є її безкоштовність. Ця база даних зазначена, як публічне надбання, що свідчить про можливість її повного, необмеженого та безоплатного використання.

Для швидкої роботи з БД обрано сервіс SQLiteStudio. Він дозволяє швидко створити базу даних, налаштувати таблиці, значення, типи і вид полів. Єдиним недоліком у випадку використання даного застосунка при розробці програми аудіоплеєра є відсутність можливості створити поле типу DateTime або подібного, наявного наприклад в SQL Server. Проте можна обійти цей недолік, створивши поле типу Text і записавши в нього правильно відформатований у коді розроблюваної програми тип Timestamp, конвертований у тип String. В теорії, сортування даних за цим полем, наприклад довжиною пісні, повинно працювати стабільно.

3.2 Етапи реалізації застосунку MP3-плеєра для відтворення та організації музичних файлів офлайн

Розробка програмного застосунку – достатньо комплексна задача. Враховуючи широкий вибір програм аудіопрогравачів як на ПК, так і на мобільні пристрої, спершу було проведено дослідження існуючого та необхідного і достатнього функціоналів для подібного програмного забезпечення. У даному випадку було виявлено велику перевагу у функціональному наповненні та зручності мобільних версій програвачів. По цій причині постала задача розроблення програмного забезпечення плеєра на ПК з впровадженням зручності мобільного застосунку.

Для реалізації спочатку створено тестовий проєкт «Застосунок Windows Forms» та імплементовано додавання різних засобів для відтворення та демонстрації інформації музичних файлів. Створено просту форму для

відображення списку файлів з обраної папки, налаштовано відтворення файлів та відображення інформації і поточного стану відтворення. Вигляд тестового застосунку зображено на рисунку 3.1.

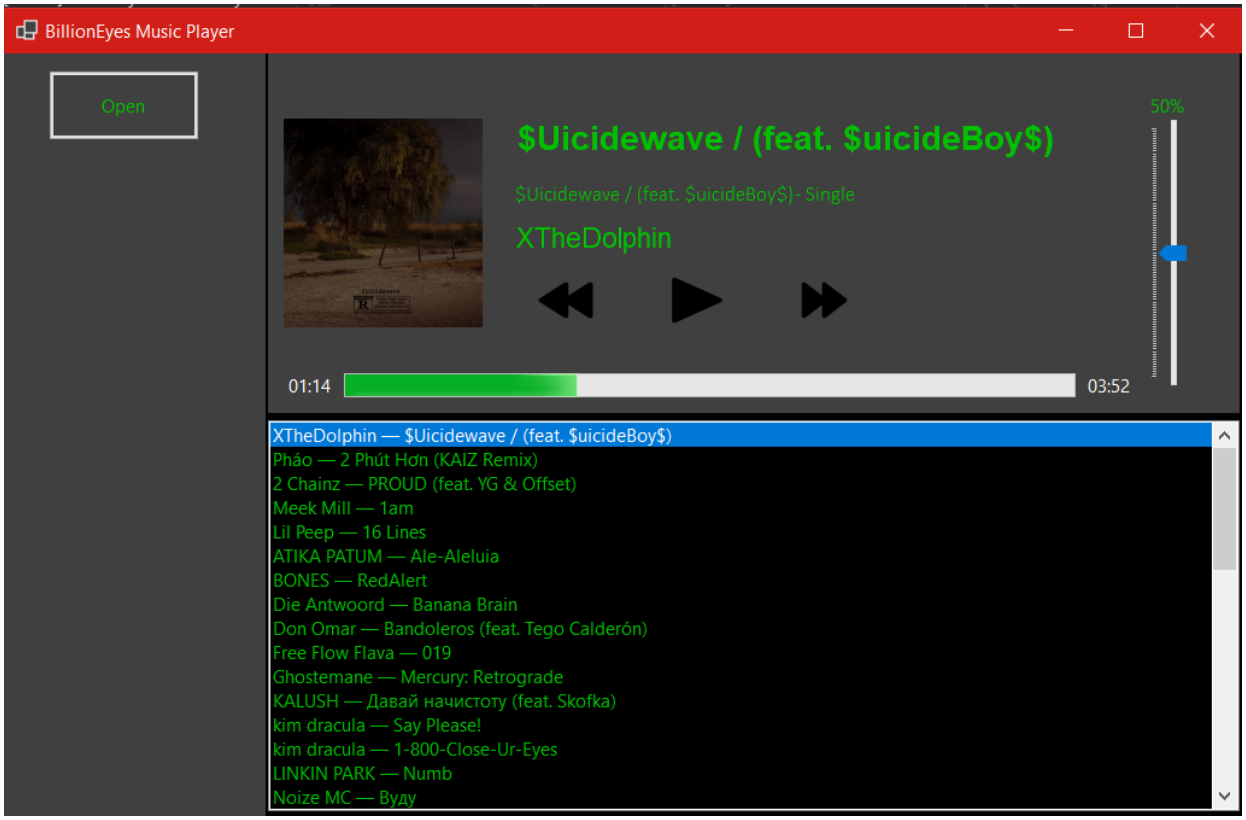

Рисунок 3.1 – Тестовий застосунок аудіопрогравача

На рисунку продемонстровано скріншот тестового застосунка MP3-програвача офлайн. Цей застосунок потрібен для перевірки сторонніх бібліотек та можливостей подальшої реалізації. У даному застосунку реалізовано виведення інформації про музичний файл, відтворення пісень у порядку черги, примітивні налаштування.

Форма містить такі елементи, що виконують певні функції:

– кнопку «Open», яка відкриває провідник операційної системи та за допомогою множинного вибору дозволяє обрати файли для відтворення;

– ListBox (список) – виводить невідсортований список всіх обраних пісень у форматі «виконавець – назва пісні»;

– PictureBox (контейнер зображення) – ці елементи відповідають за кнопки паузи та перемикання пісень вперед і назад, а також за відображення обкладинки композиції;

– Labet (надпис) відповідають за відображення інформації про пісню: назва, альбом, виконавець, тривалість і поточну позицію відтворення пісні. Також один з цих елементів відповідає за позначення рівня гучності у процентному співвідношенні;

– ProgressBar (лінія прогресу) показує прогрес відтворення композиції. При натисканні переходить на той момент пісні, значення якого вибране коричтувачем;

– TrackBar (контролер) – відповідає за налаштування звуку.

Для реалізації виводу інформації про композицію використано бібліотеку TagLib-sharp. Вона показала себе, як чудовий інструмент для роботи з аудіофайлами, а точніше з їх інформацією.

Для відтворення музики спочатку мав використовуватися елемент Windows Media Player, проте, він мав багато складнощів з підключенням, некоректну роботу та проблеми сумісності версій. Аналогічно виникли проблеми в інструменту namespace MediaPlayer. Функціонал цього простору імен дозволяє відтворювати аудіо, але має надто обмежений функціонал, стосовно обробки та ефектів. Також, значення звуку приймали значення від 0 до -∞ з умовою зникнення звуку на значенні -4001. Це ускладнювало конвертацію значень в звичні для людини.

Вибір впав на бібліотеку NAudio. Вона має величезний функціонал, дозволяє працювати з файлами різних типів, налаштовувати звукові ефекти.

Описаний вище тестовий застосунок використано для перевірки інструментарію не сумісність. Він не має бази даних, функцій плейлистів, сортування, пошуку, редагування тегів. Все це реалізовано в основному проєкті.

Першим чином, враховуючи досвід розробки тестового застосунку та вимог до аудіопрогравача, було вирішено розпочати зі створення бази даних

Для цього встановлено інструмент SQLiteStudio та створено базу даних. На рисунку 3.2 зображено момент створення таблиці Playlists.

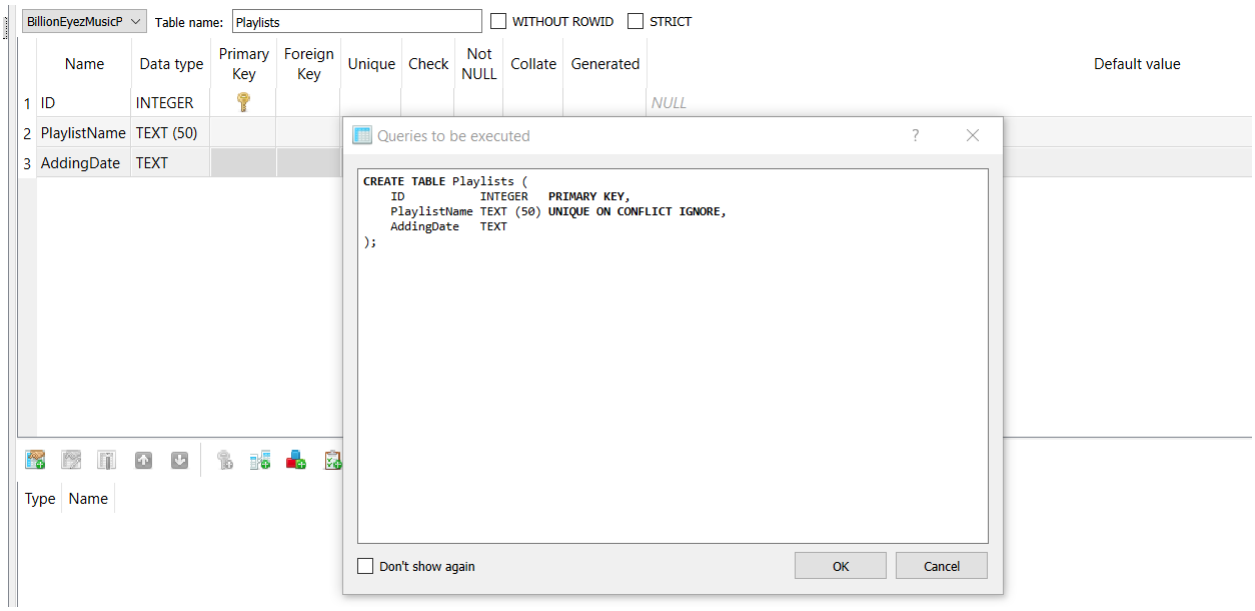

Рисунок 3.2 – Створення таблиці Playlists в SQLiteStudio

Всі стовпці створюються основі маніпуляцій з елементами інтерфейсу програми: створюються назва стовпця, тип даних, обмеження і т.д. Проте складні обмеження такі, як складний первинний ключ, потрібно додавати в редакторі SQL це можна зробити як у окремому вікні, так і при підтвердженні створення таблиці. Так була створена таблиця PlaylistsTracks (рис. 3.3).

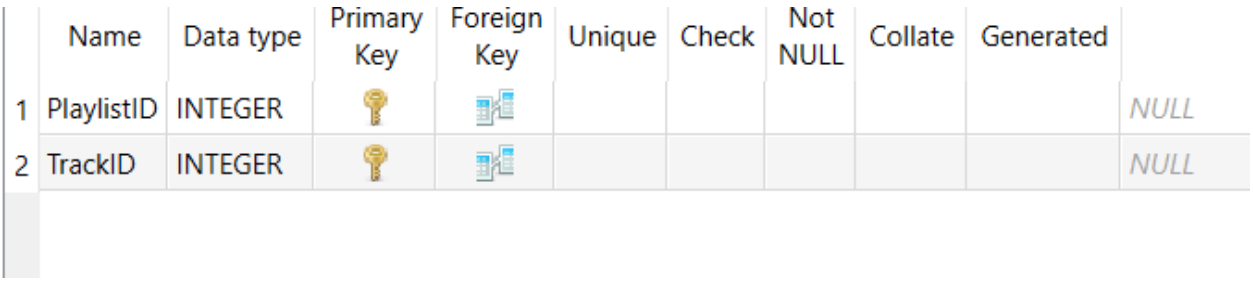

Рисунок 3.3 – Таблиця PlaylistsTracks бази даних застосунку аудіопрогравача

Як можна спостерігати на рисунку, таблиця має лише два поля, що передаються з таблиць AllTracks та Playlists. Вони (поля) містять ідентифікатори і обидва є первинними ключами.

Для розширення функціоналу програми також додано таблицю Folders, що містить поля «FolderName» і «PathToFolder». Перше поле залучено більш для естетичного характеру і виводить назву папки (обрізано рядок шляху до папки до першого символа «\» з кінця) в той час, як друге поле містить шлях до папки. В майбутньому це дасть змогу реалізувати додавання папок з музикою. Такий підхід потрібен для того, щоб користувач не обтяжував себе задачею переміщення музичних файлів в конкретну папку.

Після всіх маніпуляцій з БД: створення таблиць, залежностей між ними та обмежень було розпочато повноцінне розроблення застосунка аудіопрогравача. За основу був взятий тестовий застосунок.

Головною проблемою версії для перевірки роботи бібліотек є проблема коректності відображення даних. Елемент управління ListBox не пристосований для серйозної взаємодії з базами даних, тому його вирішено замінити на елементи DataGridView, Функції взаємодії з БД та файлами розділено між двома такими елементами: dbTrackList та dbListsWiew.

Перший відображає всі композиції з таблиці AllTracks згідно з умовами SELECT-запиту до цієї таблиці, а другий виконує відображення виконавців, альбомів, папок, списків відтворення.

Згідно з планом щодо реалізації застосунку розроблено дизайн форми та додано всі елементи контролю. Результат виконаної роботи продемонстровано на рисунку 3.4.

На рисунку 3.4 можна побачити 6 кнопок та список всіх пісень. Далі всі елементи та їх функціонал буде описано детальніше. Спочатку варто розглянути список всіх пісень. Композиції представлені у форматі «Назва, Виконавець, Альбом, Тривалість». Елемент dbTrackList також отримує з БД і унікальний ідентифікатор пісні, проте довжина колонки з ним дорівнює 0 і, на додачу, ця колонка є невидимою.

Рисунок 3.4 – Головна сторінка застосунка аудіопрогравача офлайн

Користувач не повинен бачити певні аспекти роботи програми, тому значення ідентифікаторів як пісень, так і плейлистів є прихованими. Всі пісні, незалежно від умов виведення на екран відображені у вищезазначеному форматі.

При натисканні на кнопки «Folders», «Performers», «Albums», «Playlists» з’являється елемент dbListsVeiw. Він виводить списки всіх доданих папок (формат «Назва папки, Шлях до папки»), виконавців (формат «Назва виконавця, кількість пісень в БД»), альбомів (формат «Назва альбому, Виконавець альбому, загальна кількість пісень в альбомі (саме в альбомі, а не в БД в цьому альбомі)»), плейлисти (формат «Назва плейлиста, дата створення»). Після подвійного натискання на пункт зі списку dbListsVeiw відбувається заповнення значень змінних id, album, albumArtist, trackcount, artist залежно від того, що відображав dbListsVeiw та його заміна на dbTrackList з відображенням пісень з урахуванням певної умови. Елемент dbListsVeiw стає знову невидимим та не допускає взаємодії користувача з собою. Варто зазначити, що так як використовуються лише два елементи DataGridView для відображення вмісту БД потрібно додати деякі умови та перевірку їх виконання. Також описання всіх варіантів взаємодії з базою даних достатньо громіздке і потребує частого використання в коді для різних елементів.

Для комфортної роботи створено новий проєкт типу «Бібліотека класів», в якій описано всі методи, виконуючі запити до БД. Такий підхід дозволяє швидко додавати методи обробки значень, або виведення відповідної інформації на екран. Для впровадження та перевірки правильності виконання методів достатньо зібрати проєкт бібліотеки та під'єднати його до основного проєкту. Після додавання нового метода достатньо перезібрати бібліотеку та викликати метод в коді головного проєкту.

Аналогічно класам та їх методам додано об'єкти типу enum та створені стани відображення з бази даних у застосунок (рис. 3.5).

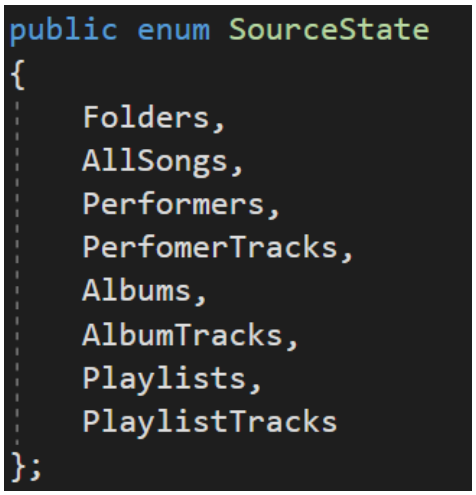

Рисунок 3.5 – Опис об'єкта enum, що відповідає за стани відображення вмісту з БД

З рисунку можна зрозуміти, що існує 8 станів відображення: для папок, всіх пісень, виконавців, пісень виконавця, альбомів, пісень з альбому, плейлистів, пісень з плейлиста. Це допомагає нівелювати засмічення коду програми повторюваним кодом та часові затрати на його обробку, замінюючи все звичайним викликом методу з бібліотеки. Всі методи, що мають у назві частину «Out» виконують заміну стану на відповідний. Даний спосіб надає можливість чітко розуміти поточний стан та робити перевірки його відповідності визначеним значенням в коді основної програми.

Наприклад: метод класу DBPlayer (саме цій клас відповідає за взаємодію з БД) OutFolders встановлює стан «Folders», що дає можливість змінити налаштування відображення для елементу dbListsView. Опис такої заміни налаштувань зображено на рисунках 3.6 та 3.7.

```
public static DataTable OutFolders()
{
    SQLiteDataReader dreader;
    using (SQLiteCommand CMD = connectionToDateBase.CreateCommand())
    {
        CMD.CommandText = "SELECT * FROM Folders";
        dreader = CMD.ExecuteReader();
    }
    DataTable Table = new DataTable();
    Table.Load(dreader);
    State = SourceState.Folders;
    return Table;
}
```

Рисунок 3.6 – Код методу DBPlayer.OutFolders() з бібліотеки класів

```
if (DBPlayer.GetState() == SourceState.Folders)
{
    dbListsView.Columns[0].Width = 400;
    dbListsView.Columns[1].Width = 762;
}
```

Рисунок 3.7 – Фрагмент коду метода timer1_Tick основного проєкта

На рисунку 3.6 підкреслено помаранчевою лінією рядок коду, в якому полю State класу DBPlayer присвоюється значення «Folders». А ось на рисунку 3.7 відображено частину перевірки стану й у разі відповідності стану «Folders» виконується форматування довжини колонок елемента dbListsView.

Аналогічним чином виконуються форматування всіх інших таблиць, що передаються в елементи типу DataGridView. Наприклад, в залежності від викликаного метода, виведення інформації кількість та довжина колонок dbListsView змінюються.

Наступним пунктом є налаштування сортування композицій за значеннями стовпців. Створено елемент управління типу panel, на якому розміщено надпис «Sort by:», випадаючий список та елемент підтвердження. Налаштовано захист від введення некоректних даних: вимкнено пошук елементів з випадаючого списку та заборонено введення тексту. Впроваджено умову відображення для панелі сортування – вона видима і доступна лише тоді, коли sbListView невидимий. Проте сортування працює лише для елемента, що відображує пісні (рис. 3.8).

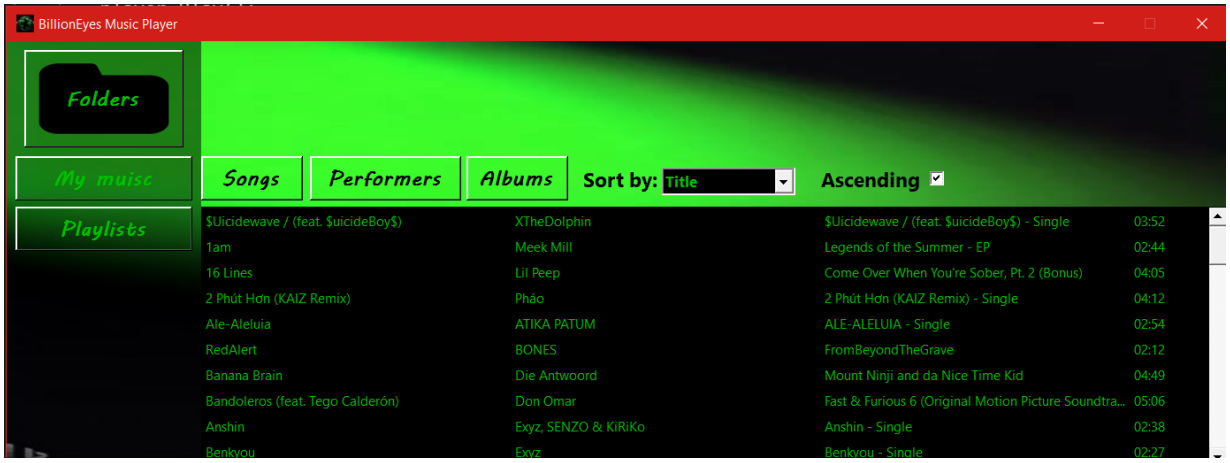

Рисунок 3.8 – Відображення головної сторінки після додавання панелі управління сортуванням

Пісні за замовчуванням сортуються за назвою у алфавітному порядку. Це, як і переважна більшість налаштувань за замовчуванням, описано у методі Form1_Load. Основні налаштування зовнішнього вигляду програми, стартові стани елементів описані саме тут. Їх зміна, як і налаштування вигляду елементів, що вимагають постійного форматування, а також деякі функції, містяться у методі timer1_tick.

Якщо попередній метод відповідав за налаштування за замовчуванням та відображення головної сторінки, то цей описує велику кількість умов для відображення елементів управління протягом всього періоду роботи застосунка.

Метод описує умови та значення для варіантів форматування представлень даних, залежності від дозволу зміни вибору пункту з dbTracklist (у даному випадку мається на увазі відображення панелі програвача), автоматичне перемикання на наступну пісню, відображення поточного прогресу відтворення композиції.

Перед розглядом панелі програвача варто ознайомитися з принципом відтворення музичних файлів у розроблюваному застосунку. У більшості застосунків подібного типу використовується черга відтворення. Цей принцип дозволяє пересуватися сторінками плеєра не перериваючи прослуховування, дозволяє зберігати останню чергу, що відтворювалася в останнє. Розроблюваний застосунок використовує іншу схему відтворення. Ідея полягає у тому, щоб відтворювати пісню безпосередньо з елемента dbTrackList й виконувати рух по черзі, використовуючи індексацію рядків елементів DataGridView.

Такий метод потребує постійного нагляду за обраним індексом, щоб програма сама не обирала випадковий індекс. Наприклад: при першому завантаженні вкладки з піснями (незважаючи на умови відображення) програма автоматично обирає нульовий індекс, але не додає до черги інші пісні, тобто довжина черги дорівнює одному. При таких умовах ламається перемикання між піснями і виникають виключення «Index Out Of Range». Навіть постійне використання методу очищення вибору не допомагає подолати цю проблему.

Тому, якщо не вдається виправити автоматичне створення нової черги, то розумним варіантом просто уникати ситуацій, що спричиняють появу таких виключень. Під час розробки програмного інструменту відтворення музичних файлів було прийняте рішення про заборону події SelectionChanged.

Для цього використано звичайну змінну типу bool під назвою disableSelectionChanged. Вона за замовчуванням має значення true, що дозволяє заборонити автоматичний вибір елемента зі списку пісень.

Проте ця змінна при значенні true також ще робить панель програвача невидимою та недоступною. Як тільки елемент обрано, вона приймає значення false і на формі з’являється елемент, що відображає інформацію про поточну відтворювану пісню, дозволяє перемотати пісню, поставити на паузу/продовжити відтворення, перемикатися між піснями і налаштувати гучність відтворення.

На рисунку 3.9 зображено процес відтворення композиції зі списку. Панель програвача містить три кнопки: попередньої пісні, наступної та паузи/продовження відтворення, 6 елементів label, котрі відповідають за відображення назви, виконавця та альбому пісні, її довжини, поточного часу відтворення, рівня гучності, полосу прогресу, яка повідомляє користувача про те, яка частина пісні вже прогнана, елемент повзунка, що налаштовує гучність.

Також є контейнер із зображенням. Він повинен містити обкладинку альбома, проте, якщо її немає в музичному файлі то встановлюється зображення з папки icons.

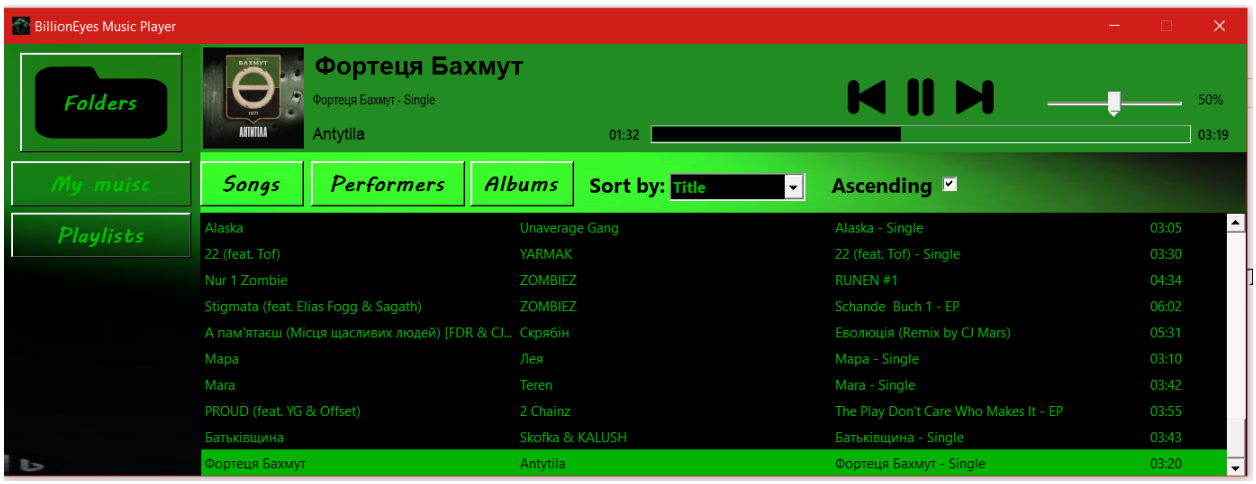

Рисунок 3.9 – Процес відтворення ораної композиції зі списку

Вся інформація про пісні отримується завдяки використанню бібліотеки TagLib-sharp. Вона дозволяє, як отримувати метадані аудіафайлів, так і записувати інформацію в ці метадані.

За відтворення музики відповідають методи з бібліотеки NAudio. Це непоганий варіант для початкової версії застосунка. Бібліотека має достатній інструментарій для запису, відтворення та базової обробки звуку. Для більш серйозних налаштувань потрібно встановлювати різні додаткові пакети. Прикладом є розроблюваний плеєр, який використовує пакет NAudio.Vawe. Ця частина бібліотеки дозволяє відтворити музичний файл маючи шлях до нього.

Для цього було створено об’єкти двох класів: *WaveOutEvent* (програвач) та *MediaFoundationReader* (зчитувач)

Перший клас є інструментом відтворення в той час як другий приймає шлях до файлу, зчитує файл, та налаштовує деякі параметри відтворення залежно від файлу. *MediaFoundationReader* – засіб для відтворення будь-якого медіаформату. Він сам визначає тип файлу та обробляє отримані дані. Ці класи об’явлено поза методами подій елементів форми.

У методі dbTracklist_SelectionChanged зчитувач отримує шлях до файлу з прихованої колонки PathToTrack вибраного рядка dbTrackList. Програвач ініціалізується з використанням зчитувач і починається відтворення. Також у цьому методі відбувається часткове заповнення даних для панелі плеєра за допомогою бібліотеки TagLib-sharp. Метод виконує відтворення пісні та наповнення панелі плеєра за умови, що елемент dbTracklist містить додатну ненульову кількість рядків і заборона на зміну індексу знято. Умова потрібна для запобігання автоматичного вибору елемента зі списку пісень.

Далі більш детально було описано функціонал кнопок перемикання між піснями та кнопки паузи/відтворення.

Кнопка паузи/відтворення перевіряє значення зчитувача і якщо воно не пусте залежно від стану відтворення ставить пісню на паузу або продовжує програвання, а також змінює зображення кнопки аналогічно в залежності від стану.

Кнопка перемикання назад виконує перевірку індексу поточного обраного рядка виконує його заміну на попередній, якщо він більший за 0.

У разі нульового індексу, тобто першого рядка зі списку) впроваджено перемикання на останню пісню в списку.

Кнопка перемикання вперед теж перевіряє індекс, проте замінює його на наступний. Якщо ж вибрано останній рядок, тоді обирається перший елемент зі списку. Код метод, що описує дії при події натискання на цю кнопку викликається в методі timer1_Tick за умови, що значення поточного часу відтворення дорівнює довжини пісні. Таким чином виконано ручне і автоматичне перемикання між піснями.

Обидві кнопки перемикання дозволяють зміну обраного індексу.

Наступним постало завдання реалізувати події натискання кнопок переходу до папок, виконавців, альбомів та плейлистів. Першим виконано перехід до вкладки папок. Ця вкладка відображає всі папки, додані до БД. Також створено функцію додання натисканням кнопки «Add Folder» нових папок в базу даних шляхом відкриття файлового діалогу провідника Windows.

Аналогічно виконано реалізацію функцій переходу до виконавців та альбомів. Проте додавати нових виконавців чи альбомів не передбачається логікою програми. Єдине, що доступно на цих сторінках, це повернення до головної або перехід до всіх пісень виконавців чи альбому.

Вкладка плейлистів працює таким же чином, але має кнопку створення нового плейлиста. При переході до пісень в плейлисті ця кнопка замінюється на іншу, котра дозволяє додати пісні до списку відтворення.

Після натискання вищеописаної кнопки повинна з’явитися нова панель поверх основної форми. Вона містить третій елемент управління DataGridView, в колонки якого було додано кнопку прапорця, і кнопку «Додати», яка має додавати всі обрані пісні. Для вмісту елемента DataGridView виконано перевірку,чи пісня вже є в даному плейлисті і якщо ні, то вона не буде додана до списку. При натисканні на кнопку додавання реалізовано вставку всіх обраних пісень у список відтворення.

Після перевірки виконання операції додавання пісень до плейлисту виявлено проблему оновлення відображуваних даних.

Для вирішення завдання виведення коректних даних після виконання зазначених дій було створено метод Refresh(). Його задача полягає в зчитуванні стану відображення з БД і перезавантаження джерела даних для елементів DataGridView залежно від стану.

Лістинг 3.1 Реалізація оновлення відображуваних композицій:

```
        private void Refresh()
        {
            bool isVisible = getliststate();
            if (isVisible)
            {
                if (DBPlayer.GetState() ==
PlayerLib.SourceState.Performers)
                {
                    dbListsView.DataSource = null;
                    dbListsView.DataSource =
DBPlayer.OutPreformers();
                    dbListsView.Refresh();
                    dbListsView.ClearSelection();
                }
                else if (DBPlayer.GetState() ==
PlayerLib.SourceState.Albums)
                {
                    dbListsView.DataSource = null;
                    dbListsView.DataSource =
DBPlayer.OutAlbums();
                    dbListsView.Refresh();
                    dbListsView.ClearSelection();
                }

                else if (DBPlayer.GetState() ==
PlayerLib.SourceState.Playlists)
                {
                    dbListsView.DataSource = null;
                    dbListsView.DataSource =
DBPlayer.OutPlaylists();
                    dbListsView.Refresh();
                    dbListsView.ClearSelection();
                }
                else if (DBPlayer.GetState() ==
PlayerLib.SourceState.Folders)
```

```
                {
                    dbListsView.DataSource = null;
                    dbListsView.DataSource =
DBPlayer.OutFolders();
                    dbListsView.Refresh();
                    dbListsView.ClearSelection();
                }
            }
            else
            {
                if (DBPlayer.GetState() ==
PlayerLib.SourceState.AllSongs)
                {
                    dbTracklist.DataSource = null;
                    dbTracklist.DataSource =
DBPlayer.OutTracks();
                    dbTracklist.Refresh();
                    dbTracklist.ClearSelection();
                    disableSelectionChanged = true;
                }
                else if (DBPlayer.GetState() ==
PlayerLib.SourceState.AlbumTracks)
                {
                    dbTracklist.DataSource = null;
                    dbTracklist.DataSource =
DBPlayer.OutAlbumTracks(album, albumartist, trackcount);
                    dbTracklist.Refresh();
                    dbTracklist.ClearSelection();
                    disableSelectionChanged = true;
                }

                else if (DBPlayer.GetState() ==
PlayerLib.SourceState.PlaylistTracks)
                {
                    dbTracklist.DataSource = null;
                    dbTracklist.DataSource =
DBPlayer.OutPlaylistTracks(id);
                    dbTracklist.Refresh();
                    dbTracklist.ClearSelection();
                    disableSelectionChanged = true;
                }
                else if (DBPlayer.GetState() ==
PlayerLib.SourceState.AllSongs)
                {
```

```
                    dbTracklist.DataSource = null;
                    dbTracklist.DataSource =
DBPlayer.OutTracks();
                    dbTracklist.Refresh();
                    dbTracklist.ClearSelection();
                    disableSelectionChanged = true;
                }
            }
        }
```

Як можна зрозуміти з коду, спочатку виконується перевірка на видимість елемента dbListsView. Якщо він видимий, то тоді виконується перевірка станів виводу, які встановлюються при відображенні вмісту елемента. У випадку невидимості dbListsView перевіряються стани для dbTrackList. Якщо виконується умова перевірки значення стану, то джерело даних поточного елемента DataGridView спочатку стирається, потім знову визначається, але тепер вже з оновленими даними з БД. Далі відбувається виклик методу DataGridView.Refresh() для оновлення вмісту елемента. Фінальна дія – виклик методу DataGridView.ClearSelection() для очищення вибраного індексу.

Метод Refresh заплановано викликати при всіх змінах у відображення вмісту елементів контролю типу DataGridView будь то додавання або видалення пісень, зміна порядку відображення, взагалі після кожної зміни станів.

Наступними були визначені процеси, які виконувалися при виклику події натискання для кнопок Performers, Albums, Songs та My music.

Для кожної з них були створені методи, що звертаються до бази даних, аналогічні тим, які використовувалися для виводу всіх пісень, списків відтворення та вмісту списків відтворення. Після натискання кнопки із перерахованих стан виводу, джерело даних змінювалися згідно запланованій задачі кожного з елементів. Метод Refresh виявився надзвичайно корисним при реалізації функціоналу кнопок.

Оскільки основні налаштування відображення елементів управління та функції кнопок прописані постала задача формування змісту dbTrackList при умові виведення всіх пісень обраного виконавця чи альбому. Ініціалізацію цього процесу вирішено залишити в події подвійного натискання на рядок зі списка dbListsView, як це виконано для демонстрації пісень зі списку відтворення.

До існуючого коду методу обробки події подвійного натискання додано умови перевірки станів виводу та реалізовано методи отримання пісень з бази даних але з виконанням певних обмежень на список пісень.

Для виконавців задача виявилася достатньо простою. Потрібно всього лише отримати всі пісні в котрих значення Artist дорівнювало значенню однойменної колонки з dbListsView.

Для альбомів умова відображення береться аналогічно умові виведення всіх альбомів. Альбоми створюються за трьома полями: назвою, виконавцем альбому, та кількістю композицій в альбомів. Така процедура формування альбомів є найкращою. Вона надає можливість виявлення неправильно визначених метаданих пісень. Іноді існує перевірка, чи однакові обкладинки альбому. Так у формування додається ще один параметр. Проте він є достатньо неефективним, оскільки більшість програвачів не дозволяють користувачу встановити обкладинку.

Таким чином умови, використані для формування списку альбомів, також імплементуються в отримання всіх пісень з конкретного альбому.

Оскільки основні налаштування відображення елементів управління та функції кнопок прописані, наступним завданням реалізації функціоналу було створити і налаштувати випадаюче меню при натисканні правою кнопкою миші на рядок з елементів управління типу DataGridView. Для цього було додано обробники події натискання мишею для елементів dbTrackList та dbListsView. У залежності від стану виводу даних створюване випадаюче меню повинно мати різний список дій. Реалізовано загальний список

можливих виконуваних дій для кожного представлення даних з вищезазначених.

Так при натисканні правою кнопкою миші на рядок dbListsView створено список з двома пунктами: «Редагувати» і «Видалити».

При натисканні пункту редагування налаштована поява панелі редагування (рис. 3.10).

Рисунок 3.10 – Панель редагування виконавця застосунка MP3-плеєра офлайн

Ця панель виводить поточну назву виконавця, альбому або списку відтворення, має текстове поле для введення нової назви. Текстове поле захищене від введення пустих рядків. Також є кнопка, котра виконує заміну даних в БД, та кнопка, яка відміняє редагування, ховаючи панель. Варто зауважити, що ця функція не змінює метадані аудіофайлів.

При виборі пункту видалення з’являється вікно, котре запитує користувача, чи дійсно він хоче видалити альбом, список відтворення або виконавця. У разі натискання кнопки підтвердження залежно від вкладки, на якій перебуває користувач, відбувається отримання всіх файлів, що відповідають параметрам видалення, в процесі якого видаляються всі файли.

Після цього відбувається видалення всієї інформації з БД, що відповідає обраним критеріям.

Останньою, але не за значенням, виявилася задача реалізації редагування метаданих пісень. Виклик цієї функції відбувається аналогічно такому для списків альбомів чи виконавців. При натисканні правою кнопкою миші з’являється випадаюче меню з двома пунктами: редагувати інформацію

та видалити пісню. Якщо вибрати перший пункт, то на екран виведеться панель з інформацією аудіофайла (рис. 3.11).

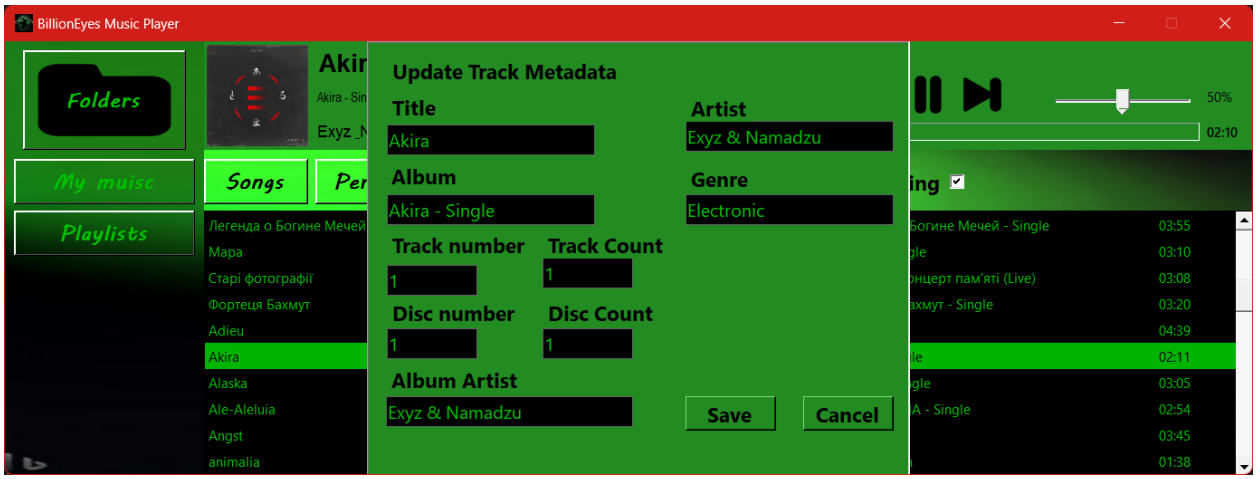

Рисунок 3.11 – Вікно редагування метаданих застосунка аудіопогравача

З рисунку 3.11 видно, що програма виводить основну потрібну інформацію. Такою є назва пісні, виконавець, назва альбому, жанр, кількість пісень в альбомі, номер пісні, номер диска, кількість дисків та виконавець альбому. Ці поля не можуть бути пустими і, як тільки поле стає пустим, колір фону змінюється на червоний, сигналізуючи користувачу про пустий рядок.

Подібний алгоритм використовується і для підтвердження, що всі рядки не пусті при натисканні кнопки «зберегти». Якщо всі поля не пусті, то дані зберігаються у\для файлу, а потім вже для бази даних.

Єдиною проблемою є правильне отримання метаданих, щоб заповнити поля перед їх зміненням. Файли можуть бути неправильно відформатовані, тому навіть після перезапису, можуть видавати помилки.

Поки не вирішено до кінця причину такої поведінки метаданих, що спричиняє критичні помилки при спробі редагувати метадані, діє обмеження, котре спочатку перевіряє, чи можна замінити інформацію тегів значеннями з форми і у разі невдачі дані номерів пісні, диску, кількості пісень та виконавець альбому лишаються незмінними.

Варто зазначити, що попереднє форматування метаданих за допомогою сервісу AutomaTag у більшості випадків позбавляє розроблюваний застосунок проблем виникнення помилки при редагуванні даних з тегів. Також були спроби відредагувати метадані спочатку у вбудованому програвачі Windows 11, проте результативність такого редагування, а точніше нівеляція помилок досить спірна.

На останок впроваджено функцію видалення пісні при виборі пункту «видалити» у випадаючому меню. Після натискання, з'являється вікно повідомлення, аналогічне тому, що й при видалення плейлистів, але підтвердивши операцію, відбувається видалення пісні та оновлення вмісту бази даних та елементів представлення даних.

3.3 Тестування розробленого застосунку та аналіз результатів

Після перевірки всіх встановлених залежностей налаштовано іконку застосунка і виконано збірку проєкту. Всі файли не приховані, оскільки на даний момент програма не буде підлягати розповсюдженню.

Сформовано .zip архів та переміщено в інше розташування. Після розпакування архіву було очищено базу даних.

Першим чином було виконано запуск застосунка через ярлик, що був створений попередньо у папці. Програма завантажується і стабільно працює, як і при розробці.

Оскільки базу даних було очищено, аудіотека порожня, Після додавання папки з'явилися всі файли в з паки і стали доступні для відтворення. Також було перевірено функціонал видалення папки і автоматичного видалення пісень з бази даних, що містяться у видаленій папці. Тестування коректної роботи з паками завершене успішно.

Додані композиції відображаються на головній стартовій сторінці при повторному запуску програми. Як і планувалося, відбувається автоматичне

сортування пісень за назвою. Результати запуску аудіопрогравача можна спостерігати на рисунку 3.12.

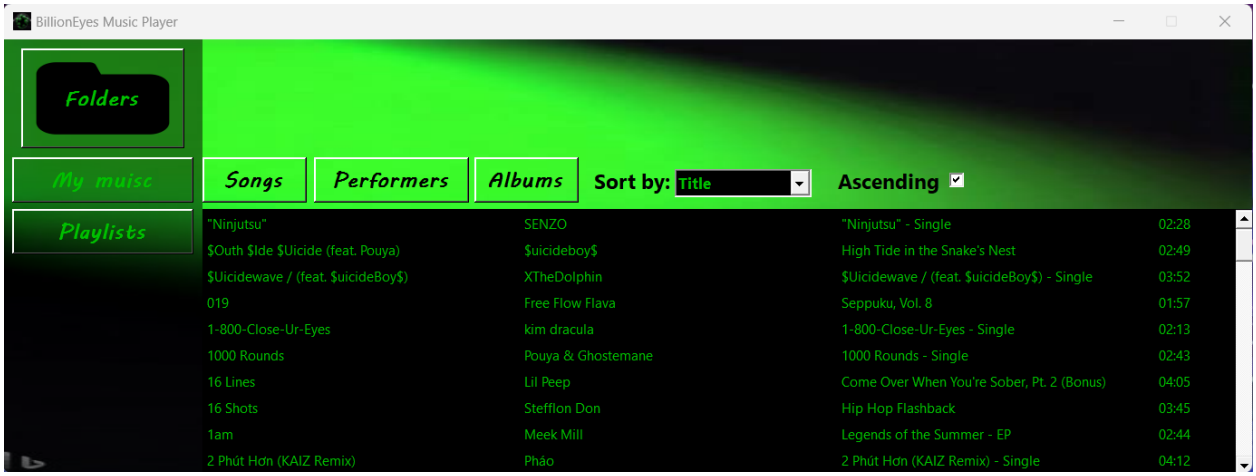

Рисунок 3.12 – Стартова сторінка застосунка аудіопрогравача

Усі пісні вдало відсортовано. При натисканні клавіші «My music» всі пісні виводяться на екран без сортування. Ця кнопка задумувалася для того, щоб оновлювати бібліотеку та спостерігати за порядком додавання пісень.

При завантаженні програми відбувається сканування, котре перевіряє вміст наявних папок і видаляє пісні з БД, якщо їх немає у папках, або навпаки – додає в базу даних, у разі їх наявності в доданих папках та відсутності записів про них в БД.

Всі пісні відтворюються нормально, проте після переназначення джерела виведення іноді відбуваються затримки при першому намаганні відтворити музику.

Можливо при формуванні черги можна було б позбутися цієї проблеми.

Власне весь функціонал, що стосується відтворення працює задовільно. Перемикання між піснями в обох напрямках по черзі відбувається вдало. Автоматичний перехід до наступної пісні також виконується. Що стосується прогресу виконання пісні, функціонал аналогічно добре працює. Налаштування гучності виконує свою задачу: можна підвищити гучність, або взагалі прибрати звук в програвачі.

Проте варто зробити зауваження, щодо якості звуку. Він достатньо плаский, оскільки не реалізовано еквалайзера, тобто налаштування гучності частот, підсилення басів, віртуалізатор. Таких налаштувань явно не вистачає для того, щоб задовольнити потреби вибагливих користувачів. Для пересічного користувача такого звуку буде досить. Проте зробити предвстановлені налаштування звуку та, що більш важливо, функціонуючий еквалайзер хоча б на 6 доріжок для налаштування частот, підсилення басів та віртуалізатор варто впровадити в наступних версіях. Як показує досвід використання програвачів, таких налаштувань досить для охоплення великої кількості аудиторії. Ефекти в стилі ехо або хору не надто актуальні.

Також слід впровадити налаштування для використання багатьох колонок для об'ємного звуку. Проте для таких цілей слід обрати інші бібліотеки для відтворення та обробки аудіо, наприклад CSCore.

Функціонал, що стосується відображення плейлистів працює нормально.

Плейлисти навіть після видалення пісень, плейлисти залишаються в БД, проте їх вміст затирається. Додавання нових списків відтворення, їх видалення задовольняє поставлені задачі. Аналогічно можна додавати пісні до плейлисту. Демонстрація роботи цієї функції варта окремої уваги.

Вигляд роботи додавання пісень у список відтворення відображено на рисунку 3.13.

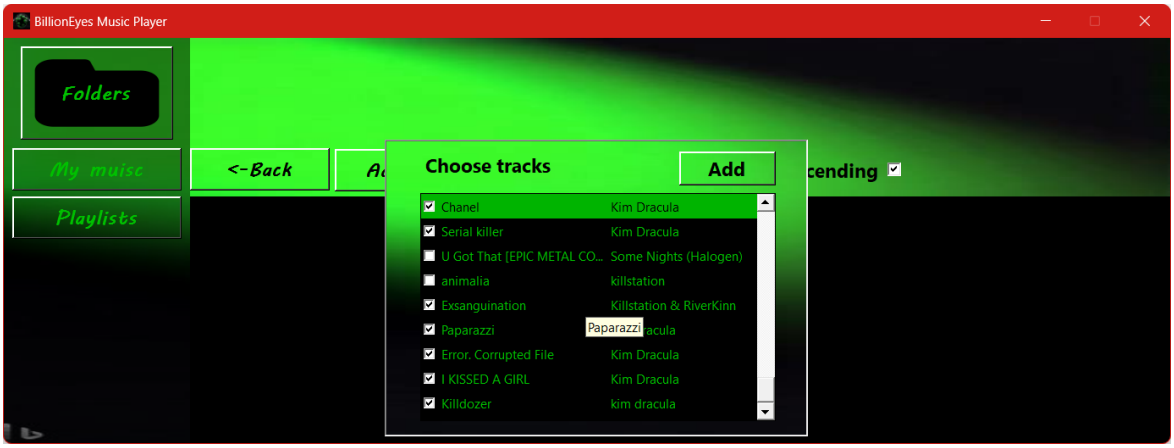

Рисунок 3.13 – Процес додавання пісень у список відтворення

Як і планувалося, при виявленні дій поза формою застосунка Billion Eyez Music Player панелі редагування тегів, додавання пісень, перейменувань та створення плейлистів зникають, не вносячи зміни в базу даних.

На рисунку 3.14 можна спостерігати додані пісні до списку відтворення та одну з пісень, що вже є у процесі програвання.

Можна переконатися, що правильно відформатовані файли з коректними метаданими відображаються у точності, як потребують того вимоги.

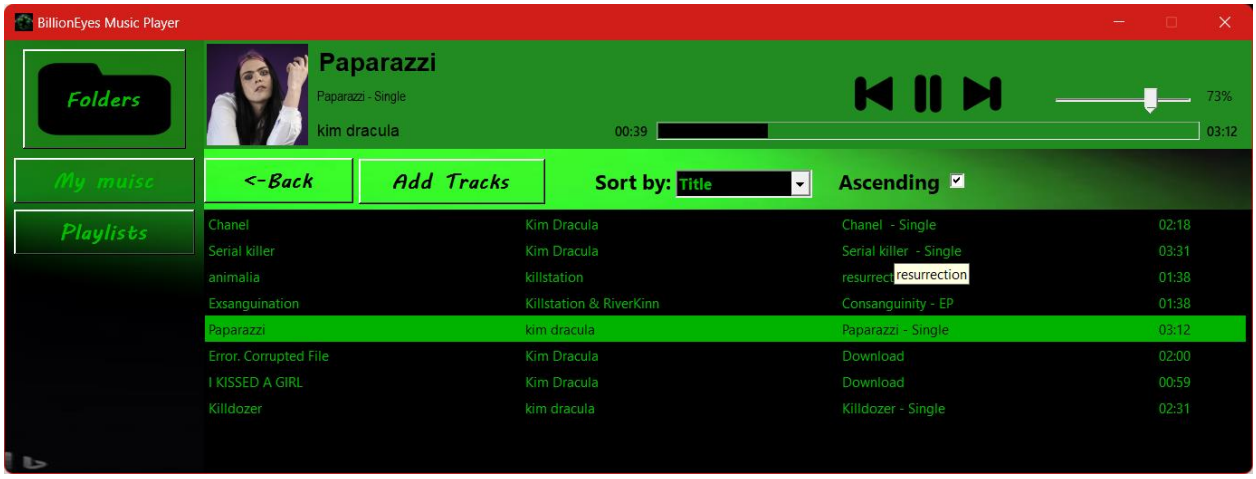

Рисунок 3.14 – Результати додавання пісень в плейлист New playlist та відтворення однієї з них

Далі виконано перевірку виведення виконавців та альбомів, видалення всіх пісень виконавця або альбому, а також редагування інформації.

Всі функції працюють добре.

Проте редагування даних відбувається лише в базі даних. Метадані пісень при цьому не чіпаються. Якщо не видаляти папки з піснями, то суттєвою проблемою це не стане. Багато програвачів не редагують теги файлів таким чином. Що більш важливо, часто застосунки плеєрів навіть при редагуванні метаданих часто автоматично виводять інформацію про пісні обрізаною. Мається на увазі виведення інформації з відкиданням зайвих пробілів в кінці. Саме тому при переносі своєї аудіотеки користувачі часто

стикаються з проблемою, що треба знову нормально організовувати бібліотеку.

Наскільки відомо, таким прикладом може послужити застосунок MI Music від компанії Xiaomi у ранніх його версіях. Він мав вбудовану базу даних і користувач вручну організовував бібліотеку. Можливо і виконувалося пряме редагування тегів, але виконання таких дій може бути дуже й дуже складною задачею. Тому не варто нарікати на програвачі, що не редагують теги напряму.

Часто користувачам навіть не потрібне таке прискіпливе редагування метаданих. Тому створення бази даних є чудовим виходом не ламати функціонал програми. Що це означає? При заміні даних в тегах файлу, його потрібно зберегти. Проте це є майже неможливим завданням, якщо файл, у якому проводяться зміни відтворюється в даний момент. Тому при використанні програвачів при зміні інформації про пісню відтворення не переривається, бо змінюються лише значення в БД.

Таким чином розроблена версія застосунка аудіопрогравача не просто так відображає в списку пісень інформацію з бази даних, а в меню програвача метадані. При використанні відредагованих файлів звісно різниці не буде, але при «сирих» файлах, різниці буде очевидна.

Це пояснення потрібне для того, що дати розуміння проблеми редагування тегів аудіофайлів. У розроблюваному застосунку втілювалася спроба реалізувати редагування метаданих. Якщо файл не відтворюється, наприклад, коли є затримка програвача при першому виборі пісні для відтворення. Якщо ж пісня вже відтворюється, було виконано обхід збереження файлу – програма запам’ятовує час відтворення, перемикає пісню на наступну, зберігає файл, повертається назад та встановлює поточний час відтворення.

Дана функція не виконується у разі лише однієї пісні в черзі. Тому варто зазначити, що такі впровадження, як редагування метаданих повинні бути переглянуті та перероблені.

Також зміна метаданих може викликати багато помилок. Все залежить від правильного формату файлу, його можливих помилок або пошкодженості. Запис нової інформації, навіть підходящої за умовою, здатен викликати виключення, котрі досить важко правильно обробити. Демонстрацію редагування метаданих можна спостерігати на рисунку 3.15.

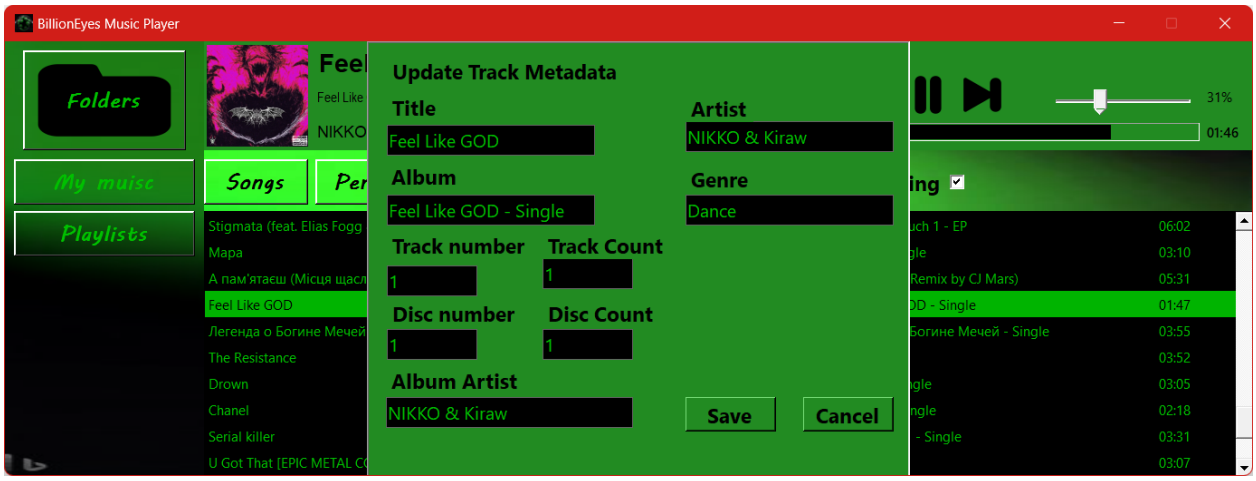

Рисунок 3.15 – Процес редагування метаданих у застосунку аудіопрогравача

Функціонал же стосовно зміни інформації у базі даних цілком задовольняю вимогам від застосунку програвача аудіо. Організація списків відтворення та особливо альбомів виконана належним чином.

3.4 Перспективи подальшої роботи

Після перевірки результатів виконаного розроблення можна з упевненістю заявити про доцільність продовження роботи над застосунком аудіопрогравача. У процесі проектування, програмної реалізації та тестування виявлено переваги та недоліки обраних інструментів та методів розробки. Ці фактори потрібно урахувати в наступних версіях програмного забезпечення. Варто змінити аудіобібліотеку для відтворення музики та впровадження ефектів, спробувати інші варіанти редагування метаданих. Також все ж слід

створювати чергу відтворення, що не залежить від виведеного на екран списку пісень.

Не зайвим буде розширити функціонал застосунку. Додавання в список відтворення прямо зі списку всіх пісень, вдосконалення та ускладнення бази даних. Щодо типу БД, SQLite показала себе відмінно при роботі застосунку, тому вирішено працювати поки що саме з нею.

Звісно, хоч і чорно-зелене оформлення залишиться класичним для інструмента Billion Eyez Music Player, планується редизайн, перенесення розробки на WPF (Windows Presentation Foundation), можливість змінювати теми та налаштовувати користувацькі кольори.

Також одним з пунктів подальшого розроблення програми аудіоплеєра є підтримка перекладу різними мовами: основними з яких повинні бути українська та англійська. Інші мови будуть впроваджені поступово в залежності від побажань користувачів.

Що стосується редагування метаданих, можна створити окремий застосунок, або вкладку в основній програмі, який би автоматично намагався відредагувати дані з тегів, а після редагування оновлював би базу даних.

Ще потрібно виконати дуже багато роботи, але, враховуючи вже наявний досвід розробки та приклади, представлені на ринку програмного забезпечення, створюваний застосунок має шанси на конкурентоспроможність у найближчому майбутньому.

ВИСНОВКИ

У рамка кваліфікаційної роботи був розроблений програмний застосунок MP3-плеєр офлайн.

У процесі виконання роботи було опрацьовано наступні питання:

– розглянуто та проаналізовано існуючі екземпляри застосунків для відтворення аудіо та організації аудіотеки, які надали можливість виділити основні проблеми в програмних інструментах, аналогічних до розроблюваного;

– розглянуто стан розвитку застосунків аудіопрогравачів, створених в Україні та за кордоном, на сучасний період. Виявлено особливості структур цих програмних інструментів та методології, що використовуються в них для відтворення музики та організації аудіотеки;

– виконано аналіз стану предметної області «Музика», та наведено можливі методи системного аналізу, що могли б використовуватися для розробки аналогів програм аудіопрогравача;

– проаналізовано численні літературні джерела на предмет існуючих методологій розробки подібних застосунків та підходів до організації аудіотек;

– розглянуто особливості бази даних для застосунку MP3-плеєра офлайн та виявлено можливі варіанти її вдосконалення;

– виконано моделювання структури та наповнення застосунку MP3-плеєра офлайн з використанням Use-Case діаграм:

– здійснено вибір інструментальних засобів для розробки застосунку MP3-плеєра офлайн та подальшої його підтримки та вдосконалення. Також виділено деякі інші можливі інструменти;

– опрацьовано етапи програмної реалізації застосунку MP3-плеєра офлайн;

– виконано тестування розробленого застосунку MP3-плеєра офлайн, проведено аналіз результатів та усунення несправностей системи;

– розглянуто перспективи подальшої роботи над покращенням розроблюваного застосунку.

Програмний інструмент працює стабільно і задовільно. Створений функціонал має можливість конкурувати з існуючими аналогами. Виявлено проблеми обраних методів відтворення музики та зроблено висновки з приводу підходів до відтворення аудіо та аудіотеки в цілому.

Розроблюваний програмний інструмент призначений для використання в межах України та за кордоном.

Результати роботи апробовано у вигляді тез доповіді під час XVIII Міжнародної науково-практичної конференції «Теоретичні та прикладні аспекти розвитку науки», Більбао, Іспанія [33].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. PI Music Player. URL: https://play.google.com/store/apps/details?id=com.Project100Pi.themusicplayer (дата звернення 10.04.2023).

2. Automatic Tag Editor. URL: https://play.google.com/store/apps/details?id=com.fillobotto.mp3tagger (дата звернення 10.04.2023).

3. AIPM. URL: https://www.microsoft.com/store/productId/9PCLLLH15SMT (дата звернення 10.04.2023).

4. Billy. URL: https://billy.en.softonic.com/?ex=DINS-635.1 (дата звернення 10.04.2023).

5. 1by1 Directory Player. URL: https://mpesch3.de/1by1.html (дата звернення 10.04.2023).

6. Cunningham, S. J., Jones, M., & Jones, S. (2004, October). Organizing digital music for use: an examination of personal music collections. ISMIR.

7. Bromberg, L. (2019). The art of music collection behaviours in the digital age. *The iJournal: Student Journal of the Faculty of Information*, *4*(2), 61-69.

8. Хабло, Д. О. (2022). *Дослідження та програмна реалізація методів підбору музики* (Master's thesis, Національний університет" Запорізька політехніка").

9. McNab, R.J., Smith, L.A., Witten, I.H., Henderson, C.L., & Cunningham, S.J. (1996, April). Towards the digital music library: Tune retrieval from acoustic input. In *Proceedings of the first ACM international conference on Digital libraries* (pp. 11-18).

10. Pablo Bello, J., & Underwood, K. (2012). Improving access to digital music through content-based analysis. *OCLC Systems & Services: International digital library perspectives*, *28*(1), 17-31.

11. Dalhuijsen, L., & van Velthoven, L. (2010, July). MusicalNodes: the visual music library. In *Proceedings of the 2010 international conference on Electronic Visualisation and the Arts* (pp. 232-236).

12. M. Weigl, D., Goebl, W., Crawford, T., Gkiokas, A., F. Gutierrez, N., Porter, A., ... & Van Tilburg, M. (2019, November). Interweaving and enriching digital music collections for scholarship, performance, and enjoyment. In *6th International Conference on Digital Libraries for Musicology* (pp. 84-88).

13. Grinstead, J. Analysis and Genre Classification of Music in Digital Music Libraries.

14. Lo, H.Y., Wang, J.C., Wang, H.M., & Lin, S.D. (2011). Cost-sensitive multi-label learning for audio tag annotation and retrieval. *IEEE Transactions on Multimedia*, *13*(3), 518-529.

15. Tan, J.Y. (2022). *MP3-music player application development* (Doctoral dissertation, UTAR).

16. Tvoroshenko, I., & Kharchenko, A. (2021). Some aspects of modern development for sign language recognition systems.

17. Tvoroshenko, I., & Kuznetsov, M. (2021). Research results of functional, white box and smoke testing methods for mobile applications.

18. Творошенко, І.С. (2018). Особливості застосування сучасних принципів штучного інтелекту до розробки ефективних механізмів моделювання складних систем. *Science and Technology of the Present Time: Priority Development Directions of Ukraine and Poland*, 118-121.

19. Творошенко, І.С. (2018). Дослідження особливостей побудови нечітких відношень під час відображення динамічних взаємодіючих нечітких процесів складних систем.

20. Гороховатський, В. О., & Творошенко, І. С. (2021). Методи інтелектуального аналізу та оброблення даних: навч. посібник.

21. Tvoroshenko, I., & Almakaieva, A. (2020). Application of procedural generation of game content using software algorithms.

22. Tvoroshenko, I. (2020). Information technologies for decision-making on the conditions of spatially distributed objects. In *I International Scientific and Practical Conference. Problems and perspectives of modern science and practice, Austria* (pp. 45-50).

23. Гороховатський, В.О., Творошенко, І.С., & Сидоренко, Д. (2021). Класифікація зображень із використанням кластерного подання.

24. Гороховатський В.О., Творошенко І.С., Чмутов Ю.В. (2022). Застосування систем ортогональних функцій для формування простору ознак у методах класифікації зображень. *Сучасні інформаційні системи*, 6 (3), С. 5–12.

25. Гороховатський В., Передрій О., Творошенко І., Марков Т. (2023). Матриця відстаней для множини компонентів структурного опису як інструмент для створення класифікатора зображень. *Сучасні інформаційні системи*, 7(1), С. 5-13.

26. Tvoroshenko, I.S., & Kuznetsov, M. (2021). About the role of testing in process of mobile application development.

27. Daradkeh Y.I., Gorokhovatskyi V., Tvoroshenko I., and Zeghid M. (2022). Tools for Fast Metric Data Search in Structural Methods for Image Classification. *IEEE Access*, 10, pp. 124738-124746.

28. Tvoroshenko, I., & Babochkin, O. (2021). Object identification method based on image keypoint descriptors.

29. Tvoroshenko, I., & Temchur, K. (2021). Features of software application development for food recognition using deep machine learning methods.

30. Tvoroshenko, I., & Koriakin, I. (2021). Analysis of methods for detecting and classifying the likeness of human features.

31. Tvoroshenko, I., & Kukharchuk, V. (2021). Current state of development of applications for recognition of faces in the image and frames of video captures.

32. Gorokhovatskyi V., Tvoroshenko I., Kobylin O., and Vlasenko N. (2023) Search for visual objects by request in the form of a cluster representation for the structural image description, *Advances in Electrical and Electronic Engineering*, 21(1), pp. 19-27.

33. Шуліка Д. (2023) Моделювання структури мобільного застосунку MP3-плеєра для відтворення та організації музичних файлів офлайн, Abstracts of XVIII International Scientific and Practical Conference «Theoretical and applied aspects of the development of science» (May 09 – 12, 2023). Bilbao, Spain, pp. 505-511.