МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

Факультет комп'ютерних наук,
управління та адміністрування
Кафедра інформаційних технологій

Бакалаврська кваліфікаційна робота

на тему: Розробка програмної реалізації аудіо плеєра з сенсорним управлінням

Виконав студент 4 курсу групи К-25
Спеціальність 122 комп'ютерні науки
Тіщенко Микита Сергійович

Керівник асистент
Штефан Наталія Зінов'ївна

Консультант к.т.н., доцент
Великодний Станіслав Сергійович

Рецензент к.ф.-м.н., доц.
Ткач Тетяна Борисівна

Одеса 2020

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

AAC − Advanced Audio Coding

FLAC − Free Lossless Audio Codec

UML − Unified Modeling Language

WMA – Windows Media Audio

WPF – Windows Presentation Foundation

XAML − Extensible Application Markup Language

Xml – eXtensible Markup Language

Баги – помилки у програмному коді

Кодекі – виконувати перетворення даних або сигналу

Плагін – незалежно компільований програмний модуль, що динамічно підключається до основної програми і призначений для розширення і / або використання її можливостей

Плейлист – список відтворення

Скробблінг – збір інформації про прослуховування музичних треків

Юзабіліті – ступінь зручності використання

Actor – актор на діаграмі варіантів використання

AC3 – формат аудіо файлу

Class Diagram – діаграма класів

CodeLens – індикатор посилань на код

Common Language Runtime – виконуюча середовище для байт-коду C/C#

DirectX – набір API, розроблених для вирішення завдань, пов'язаних з програмуванням під Microsoft Windows

JavaScript – мова програмування

JetAudio – аудіо плеєр

iTunes – аудіо плеєр

Managed API – керований API-інтерфейс

MP3 – формат аудіо файлу

Microsoft Expression Blend – програма від корпорації Microsoft, призначена для розробки дизайну веб-інтерфейсів і графічних настільних додатків

Microsoft Silverlight – програмна платформа для написання і запуску багато-функціональних інтернет-додатків

Microsoft .NET – програмна платформа

Milcore.dll – забезпечує інтеграцію компонентів WPF з DirectX

MusicBee – аудіо плеєр

Plexamp – аудіо плеєр

PresentationCore.dll – файл компоновки .NET Framework

PresentationFramework.dll – файл компоновки .NET Framework

Unmanaged – API використовується для інтеграції вищого рівня з DirectX

Use Case – варіант використання

Use Case Diagram – діаграма варіантів використання

VB.NET – об’єктно-орієнтована мова програмування

Visual Studio 2017 – середовище візуальної розробки

WAV – формат файлу-контейнера для зберігання запису цифрованого аудіо потоку

Winamp – аудіо плеєр

WindowsBase.dll – файл компоновки .NET Framework

WindowsCodecs.dll – файл компоновки .NET Framework

Windows CE – операційна система для смартфонів

Windows Forms – технологія Visual Studio 2017

Windows Mobile – мобільна операційна система

Windows Phone – мобільна операційна система

XAML – ContextMenu допомагає в створенні навігаційних меню програми

XAML – PageViewer елемент управління, який працює зі змістом сторінки

.ape – формат аудіофайлу

.flac – формат аудіофайлу

.mp3 – формат аудіофайлу

.m4a – формат аудіофайлу

.ogg – формат аудіофайлу

.wav – формат аудіофайлу

.NET Compact Framework – версія .NET Framework, яка розроблена для запуску додатків на пристроях, заснованих на платформі Windows CE

.NET Framework – програмна платформа

ВСТУП

Серед засобів мультимедіа, звук – явище особливе, аналог фотографії, точна цифрова копія введених ззовні звуків. Це може бути зроблений з мікрофона запис вашого голосу, копія звукових доріжок з компакт-диска та інших джерел.

Для відтворення аудіо-файлів необхідні спеціальні програми-програвачі – плеєри, тобто комп'ютерна програма для відтворення цифрової аудіо- чи відео-інформації. Для даного проекту є важливим програвати аудіо різноманітних форматів, найпоширенішим з яких це FLAC, MP3, WAV, AAC, WMA, AC3 та багато інших.

MP3 є одним з найбільш популярним форматом цифрового кодування звукової інформації з втратами. Він широко використовується в файлообмінних мережах для оціночної передачі музичних творів. Формат може програватися практично у всіх популярних операційних системах, на більшості портативних аудіоплеєрів, а також підтримується всіма сучасними моделями музичних центрів і DVD-плеєрів.

У форматі MP3 використовується алгоритм стиснення з втратами, розроблений для істотного зменшення розміру даних, необхідних для відтворення запису і забезпечення якості відтворення звуку дуже близького до оригінального, хоча спеціалісти кажуть про відчутну різницю.

Нині існує велика різноманітність музичних програвачів з великим спектром функцій. Найрозповсюдженіший аудіо-формат mp3 має в собі систему тегів, в яких описується:

- назва (title);
- виконавець (artist);
- рік (date);
- жанр (genre);
- альбом (album);

– трек (track number).

Темою дипломного проекту є розробка програмної реалізації аудіо плеєра з сенсорним управлінням.

Програмний додаток повинен забезпечити вибір аудіо файлу, регулювання гучності звуку, перемотування файлу, зміни треку та мати зручний інтерфейс. Крім цього, необхідно реалізувати сенсорне управління.

Загальні характеристики кваліфікаційної роботи:

- повний обсяг сторінок пояснювальної записки – 48;
- кількість рисунків – 14;
- кількість таблиць – 0;
- кількість посилань – 15.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис предметної області

Мультимедіа – це збірне поняття для різних комп'ютерних технологій, при яких використовується кілька інформаційних середовищ, таких, як графіка, текст, відео, фотографія, рухомі образи (анімація), звукові ефекти, високоякісний звуковий супровід.

Багато людей люблять слухати музику і дивитися фільми на своєму персональному комп'ютері або ноутбуці. Для цього можливостей стандартного програвача може не вистачити, тому що найчастіше немає необхідних кодеків, що дозволяють працювати з тим чи іншим форматом мультимедійних файлів [1][1)].

Для користувача на ринку представлено велика кількість програвачів, але знайти оптимальний для більшості потреб не так просто. Ось чому створення повноцінного плеєру є досить актуальним питанням в сучасному житті.

1.2 Характеристика об’єкту розробки

Об’єктом розробки є програмна реалізація аудіо плеєра з сенсорним управлянням. Плеєр повинен реалізовувати стандартні функції, такі як завантаження та створення плейлисту, управління рівнем звуку, відображення інформації на інтерфейсі щодо активного треку.

Крім цього, до задач плеєру додано сенсорне управління (корисно використовувати на ноутбуках), відображення графічної інформації для кожного треку. Також до вимог до програмного продукту поставлено реалізація оптимального інтерфейсу для користувача з урахуванням кольорової гами.

[1)] [1] 11 лучших музыкальных плееров для Windows. URL: https://geeker.ru/music/muzykalnyj-pleer-dlya-windows/. (дата звернення 18.03.2020).

1.3 Аналіз існуючих аналогів

Для чіткої постановки вимог і задач до об’єкту розробки на першому етапі його проектування слід обов’язково розглянути декілька аналогів. Сьогодні існує величезна кількість платних і безкоштовних додатків, які дозволяють відтворювати абсолютно будь-які відео і аудіо.

Першим аналогом розглянемо плеєр Plexamp (рис. 1):

Рисунок 1 – Скріншот робочого вікна плеєру Plexamp

Маючи в своїй назві очевидну відсилання до Winamp, Plexamp фокусується на тому, щоб максимально спростити доступ до треків, візуалізувати плейлисти і альбоми через привабливі обкладинки і безліч приємних фішок.

Поки що програма має невеликі баги, і тому користувачу знадобиться Plex Media Server для доступу до пісень, але якщо ви перш за все ви цінуєте простоту, то варто подивитися, як додаток використовує пошукові запити і інструменти автоматичного виявлення замість стандартної сітки треків [2][1)].

Оптимальний інтерфейс для користувача допомагає плеєру бути у топі серед програмних додатків.

[1)] [2] Plex обновила iOS-версии плеера Plexamp и программы Plex Dash. URL: https://stereo.ru/news/plex-plexamp-plex-dash-update. (дата звернення 18.03.2020).

Наступний аналог – програмний додаток «MusicBee» (рис.2):

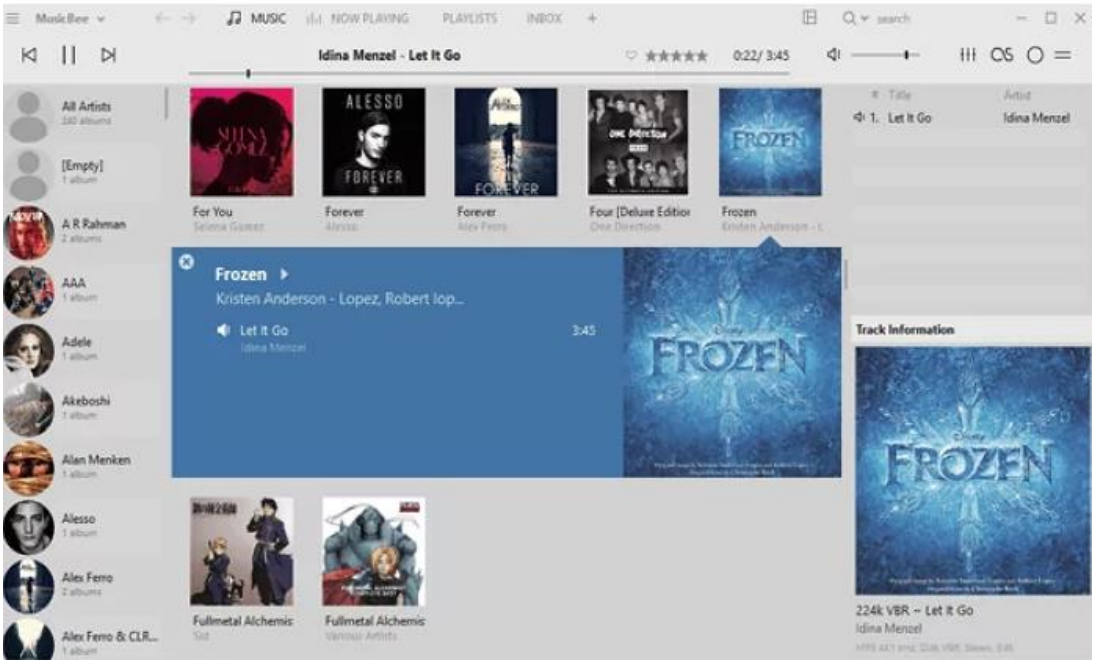

Рисунок 2 – Скріншот робочого вікна плеєру MusicBee

«MusicBee» включає в себе чи не будь-яку функцію, яка вам потрібна: від нормалізації гучності і копіювання компакт-дисків до скробблінга Last.fm і вбудованого еквалайзера, щоб можливо було налаштувати звук на свій смак. Програма швидко виконає роботу по скануванню файлів, що зберігаються на комп'ютері, при необхідності може імпортувати треки з iTunes, а також надасть безліч способів обробки файлів.

Якщо користувач виявить, що інтерфейс за замовчуванням трохи перевантажений, є можливість використовувати розумний режим театру – корисну альтернативну розкладку. Завдяки підтримці інтелектуальних плейлистів, сумісності з декількома плагінами Winamp, синхронізації з рядом пристроїв, вибору скінів і спільноті користувачів «MusicBee» ідеально підходить для тих, кому набрид iTunes [1][1)].

[1)] [1] 11 лучших музыкальных плееров для Windows. URL: https://geeker.ru/music/muzykalnyj-pleer-dlya-windows/. (дата звернення 18.03.2020).

«JetAudio» (рис. 3) є багатофункціональним програвачем аудіо- і відеофайлів. Відрізняється, в першу чергу, від багатьох інших плеєрів – точкової налаштуванням звуку, що дозволяє домогтися відмінного звучання практично на будь-яких звукових картах [3][1)].

Рисунок 3 – Скріншот робочого вікна плеєру JetAudio

Основні переваги:

- підтримка всіх популярних форматів аудіо: .wav, .mp3, .ogg, .flac, .m4a, .mpc, .tta, .wv, .ape, .mod, .spx і ін .;
- дуже якісний звук: є "улучшалкі" у вигляді: BBE, BBE ViVA, Wide, Reverb, X-Bass;
- так само програвач (на відміну від багатьох інших) відтворює 32 бітний звук;
- 10-смуговий еквалайзер (+32 попередні налаштування для нього);
- можливість кодування музики з одного формату в інший (копіювання CD-дисків);

[1)] [3] Рейтинг лучших музыкальных плееров для Windows. URL: https://vseprost.ru/rejting-luchshix-muzykalnyx-pleerov-dlya-windows.html. (дата звернення 18.03.2020).

- можливість пошуку і прослуховування радіостанцій в інтернеті і ін.

Загалом, програвач можна рекомендувати, в першу чергу, тим, хто не задоволений якістю звуку. Бо початківців користувачів може дещо збентежити відсутність великої різноманітності обкладинок, відсутність російської мови, і мала кількість візуальних ефектів.

Після огляду аналогів до об’єкту розробки поставлені наступні вимоги:

- оптимальний інтерфейс для користувача;
- створення і завантаження плейлистів;
- робота з аудіо файлами різного формату;
- реалізація функції сенсорного управління.

1.4 Інструментальні засоби розробки

1.4.1 Мова програмування C#

Будь-яка мова програмування є тільки формою вираження людських думок – для того щоб щось порахувати ми використовуємо числа, передавати словесну інформацію ми пишемо, візуальну інформацію – малюємо тощо. Те ж саме стосується світу програмування – універсального набору інструментів просто-напросто не існує. Усі мови програмування створені для досягнення максимальної ефективності конкретних задач. Для кожного проекту є свій набір інструментів, котрий найкраще відпрацює саме для нього.

C # є об'єктно-орієнтованою мовою, але підтримує також і компонентно-орієнтоване програмування. Важлива особливість таких компонентів – це модель програмування на основі властивостей, методів і подій. Кожен компонент має атрибути, які надають декларативні відомості про компоненті, а також вбудовані елементи документації [4][1)].

[1)] [4] Язык C#. Алексей Федоров. URL: https://compress.ru/article.aspx?id=11719. (дата звернення 20.03.2020).

Ще одна з причин розробки компанією Microsoft нової мови програмування – це створення альтернативи мови Java. Як відомо, реалізація Java у Microsoft була ліцензійно чистою – Microsoft у властивій їй манері внесла в свою реалізацію багато чого від себе.

Компанія Sun, власниця Java, подала на Microsoft в суд, і Microsoft цей суд програла. Тоді Microsoft вирішила взагалі відмовитися від Java, і створити свій Java-подібна мова, який і отримав назву C #.

Головною особливістю мови C# є його орієнтованість на платформу Microsoft .NET – творці C# ставили собі за мету надання розробникам природних засобів доступу до всіх можливостей платформи .NET.

C# надає мовні конструкції, безпосередньо підтримують таку концепцію роботи. Завдяки цьому C # відмінно підходить для створення і застосування програмних компонентів.

Ось лише кілька функцій мови C#, що забезпечують надійність і стійкість додатків:

- прибирання сміття автоматично звільняє пам'ять, зайняту знищеними і невикористовуваними об'єктами;
- обробка виключень дає структурований і розширюваний спосіб виявляти і обробляти помилки;
- сувора типізація мови не дозволяє звертатися до не ініціалізованих змінним, виходити за межі масиву або виконувати неконтрольоване приведення типів.

Щоб забезпечити сумісність програм і бібліотек C# при подальшому розвитку, при розробці C# багато уваги було приділено управлінню версіями. Таким чином, C# є нову мову програмування, орієнтований на розробку для платформи .NET і придатний як для швидкого прототипування додатків, так і для розробки великомасштабних додатків [5][1)].

[1)] [5] C#. Объектно-ориентированный язык программирования. URL: https://habr.com/ru/hub/csharp/. (дата звернення 25.03.2020).

Багато мови програмування обходять увагою це питання, і в результаті програми на цих мовах ламаються частіше, ніж хотілося б, при виході нових версій залежних бібліотек.

1.4.2 Вибір середи розробки

Microsoft Visual Studio – серія продуктів фірми Майкрософт, які включають інтегроване середовище розробки програмного забезпечення та ряд інших інструментальних засобів.

При розробці і реалізації програмного продукту було використано середовище візуальної розробки Visual Studio 2017 (рис. 4):

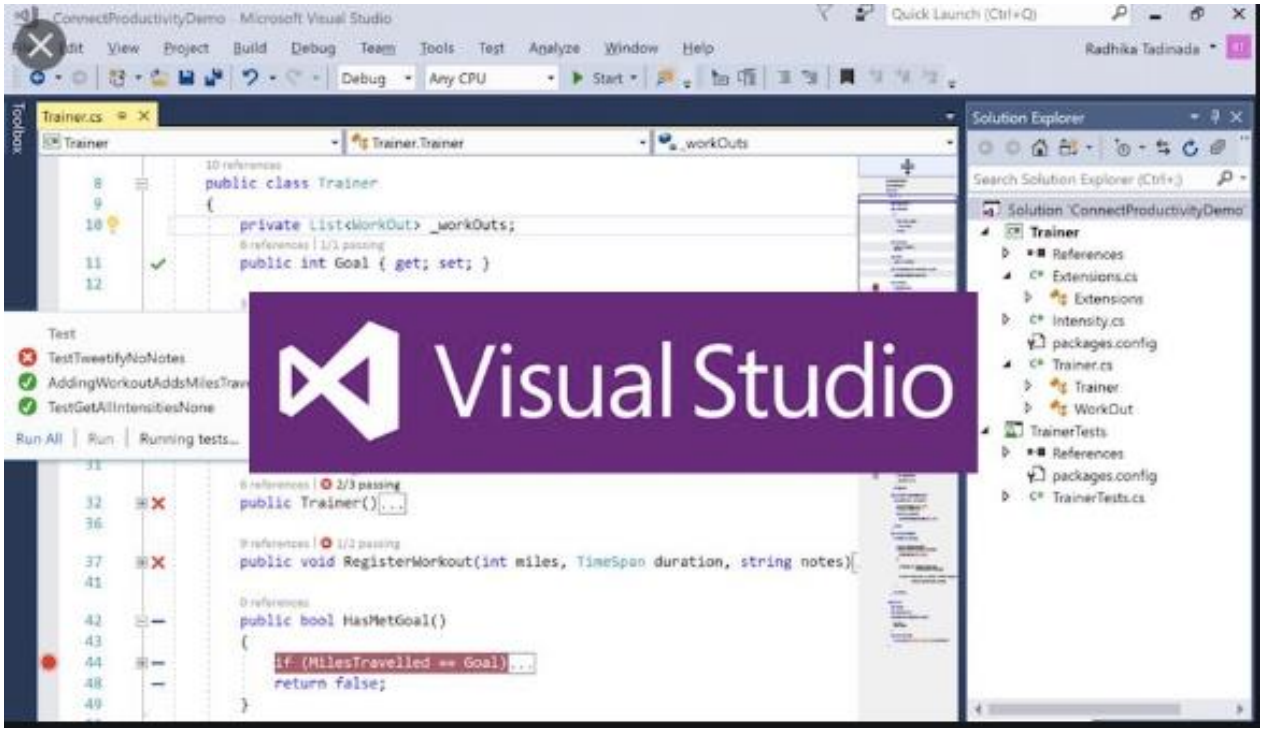

Рисунок 4 – Скріншот робочого вікна Visual Studio 2017

Visual Studio 2017 Professional – універсальне багато платформне інтегроване середовище всебічного тестування модулів і розробки додатків.

Ці продукти дозволяють розробляти як консольні програми, так і програми з графічним інтерфейсом, в тому числі з підтримкою технології Windows Forms, а також веб-сайти, веб-застосунки, веб-служби як в рідному, так і в керованому кодах для всіх платформ, що підтримуються Microsoft Windows, Windows Mobile, Windows Phone, Windows CE, .NET Framework, .NET Compact Framework та Microsoft Silverlight.

Visual Studio дотримує всі основні критерії для забезпечення максимальної якості коду. З інструментом CodeLens створення, налагодження та розгортання додатків в Visual Studio 2017 значно спрощуються на всіх етапах проектування [6][1)].

Visual Studio 2017 Professional являє собою універсальний засіб для фахівців з розробки і проектування, що підтримує більшість платформ розробки, в тому числі Windows і Windows Server, хмарну і веб-середовище, гарантує 100% коректність кінцевого коду.

Visual Studio 2017 Professional є платною версією. Для розробки програмного продукту було вибрано програмне середовище Visual Studio 2017 Community з наступних причин:

- безкоштовне розповсюдження;
- Visual Studio 2017 є інтегрованим середовищем швидкої розробки програмного забезпечення для роботи під Microsoft Windows, на даний момент найпоширенішою операційною системою;
- середовище програмування Visual Studio 2017 року було обрано завдяки можливості безпроблемного зв'язку з офісним пакетом Microsoft Office.

Досить проста установка зв'язку з цим пояснюється тим, як уже зазначалося вище, що навколишнє середовище Visual Studio 2017 – це один з найвідоміших засобів створення Windows-додатків.

[1)] [6] Microsoft Visual Studio. URL: https://open-file.ru/programs/microsoft-visual-studio. (дата звернення 25.03.2020).

Дуже легка до налаштування ы користується все більшою популярністю серед розробників програмних додатків та відеоігор.

- Visual Studio 2017 року було обрано через доступності цієї програми будь-якому користувачеві;
- відзначається своєю стабільністю, швидкістю і низькими вимогами до апаратного забезпечення комп'ютера.

Функціональна структура Visual Studio 2017 середовища включає в себе:

- редактор вихідного коду, який включає безліч додаткових функцій, як автодоповнення IntelliSense, рефракторінг коду і т. д .;
- відладчик коду;
- редактор форм, призначений для спрощеного конструювання графічних інтерфейсів;
- веб-редактор;
- дизайнер класів;
- дізайнерсхем баз даних.

Visual Studio також дозволяє створювати і підключати сторонні додатки (плагіни) для розширення функціональності практично на кожному рівні, включаючи додавання підтримки систем контролю версій вихідного коду (Subversion і VisualSourceSafe), додавання нових наборів інструментів (для редагування і візуального проектування коду на предметно-орієнтованих мовах програмування або інструментів для інших аспектів процесу розробки програмного забезпечення) [7][1)].

[1)] [7] Знакомство с Visual Studio. URL: https://habr.com/ru/sandbox/79099/. (дата звернення 26.03.2020).

2 ПРОЄКТУВАННЯ ОБ'ЄКТУ РОЗРОБКИ

2.1 Використання технології WPF і мови розмітки XAML

Windows Presentation Foundation (WPF, кодова назва – Avalon) – графічна (презентаційна) підсистема в складі .NET Framework 3.0, що має пряме відношення до XAML. WPF є високорівневим об'єктно-орієнтованим функціональним шаром (англ. framework), що дозволяє створювати двовимірні та тривимірні інтерфейси.

Багато років .NET розробники створювали настільні прикладні програми, використовуючи технологію Windows Forms. Але робота технології жорстко пов'язана на архітектурі операційної системи і деякі завдання, які повинні виконуватися швидко і без зайвого програмування, вимагають великих витрат часу і зусиль [8][1)].

Windows Presentation Foundation (WPF) змінила світ програмування настільних прикладних програм. Поклавши в основу технологію DirectX, Microsoft надає можливість розробникам швидко створювати складні елементи управління і повністю керувати процесом візуалізації.

Тепер створити красиву кнопку з анімаційними ефектами можна не написавши жодного рядка коду на C#.

Працювати з мультимедійним вмістом стало набагато простіше, розширилася модель зв'язування даних, друку і роботи з документами. WPF комбінує кращі аспекти традиційної розробки для Windows, з безліччю нововведень, дозволяючи будувати насичені графікою інтерфейси для користувача.

Окрім цього, така технологія має всього дві прості вимоги: знання мови C# та вміння користуватися бібліотеками с [9][2)].

Схематично архітектура WPF представлена на рис. 5:

[1)] [8] Введение в WPF. Особенности платформы WPF. URL: https://metanit.com/sharp/wpf/1.php. (дата звернення 26.03.2020).

[2)] [9] Windows Presentation Foundation (WPF). URL: http://sibakademsoft.ru/windows-presentation-foundation-wpf. (дата звернення 29.03.2020).

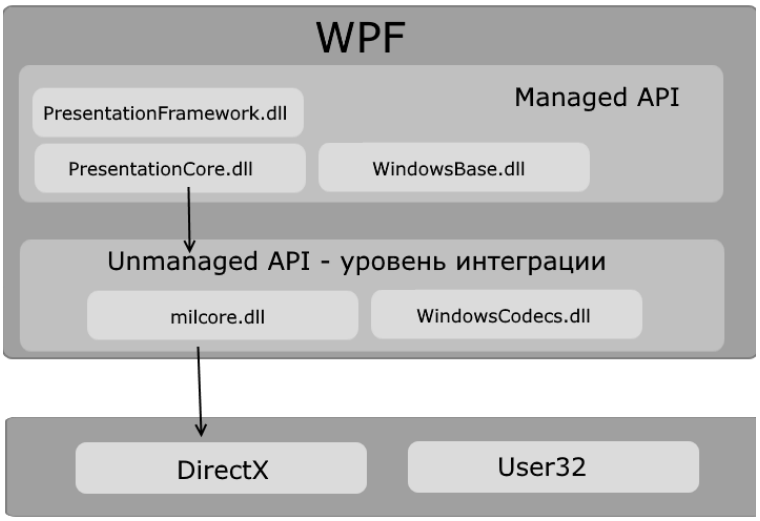

Рисунок 5 – Схема архітектури WPF

WPF розбивається на два рівня: managed API і unmanaged API (рівень інтеграції з DirectX). Managed API (керований API-інтерфейс) містить код, що виконується під управлінням загальномовного середовища виконання .NET – Common Language Runtime. Цей API описує основний функціонал платформи WPF і складається з наступних компонентів:

- PresentationFramework.dll: містить всі основні реалізації компонентів і елементів управління, які можна використовуються при побудові графічного інтерфейсу;
- PresentationCore.dll: містить всі базові типи для більшості класів з PresentationFramework.dll;
- WindowsBase.dll: містить ряд допоміжних класів, які застосовуються в WPF, але можуть також використовуватися і поза даної платформи;
- Unmanaged API використовується для інтеграції вищого рівня з DirectX;
- Milcore.dll: власне забезпечує інтеграцію компонентів WPF з DirectX. Даний компонент написаний на некерованому коді (C/ C ++) для взаємодії з DirectX;

- WindowsCodecs.dll: бібліотека, яка надає низькорівневу підтримку для зображень в WPF [8][1)].

Ще нижче власне знаходяться компоненти операційної системи і DirectX, які виконують візуалізацію компонентів програми, або виконують іншу низькорівневу обробку. Зокрема, за допомогою низькорівневого інтерфейсу Direct3D, який входить до складу DirectX, відбувається трансляція.

Переваги, які надає WPF розробнику:

- використання традиційних мов .NET-платформи – C# і VB.NET для створення логіки додатка;
- можливість декларативного визначення графічного інтерфейсу за допомогою спеціальної мови розмітки XAML, заснованому на xml і представляє альтернативу програмному створення графіки та елементів управління, а також можливість комбінувати XAML і C# / VB.NET;
- незалежність від роздільної здатності екрану: оскільки в WPF всі елементи вимірюються в незалежних від пристрою одиницях, додатки на WPF легко масштабуються під різні екрани з різною роздільною здатністю.
- нові можливості, створення тривимірних моделей, прив'язка даних, використання таких елементів, як стилі, шаблони, теми та інші;
- хорошу взаємодію з WinForms, завдяки чому, наприклад, в додатках WPF можна використовувати традиційні елементи управління з WinForms [9][2)];
- багаті можливості по створенню різних додатків: це і мультимедіа, і двомірна і тривимірна графіка, і багатий набір вбудованих елементів управління, а також можливість самим створювати нові елементи,

---

[1)] [8] Введение в WPF. Особенности платформы WPF. URL: https://metanit.com/sharp/wpf/1.php. (дата звернення 26.03.2020).

[2)] [9] Windows Presentation Foundation (WPF). URL: http://sibakademsoft.ru/windows-presentation-foundation-wpf. (дата звернення 29.03.2020).

створення анімацій, прив'язка даних, стилі, шаблони, теми і багато іншого;

- апаратне прискорення графіки – незалежно від того, чи працюєте ви з 2D або 3D, графікою або текстом, всі компоненти програми транслюються в об'єкти, зрозумілі Direct3D, і потім візуалізуються за допомогою процесора на відеокарті, що підвищує продуктивність, робить графіку більш плавною.

XAML (скорочення від Extensible Application Markup Language – розширювана мова розмітки застосунків) – є мовою розмітки, яку використовують для створення екземплярів об'єктів .NET. Хоча мова XAML – це технологія, що може бути застосовна до багатьох різних предметних областей, її головне призначення – конструювання інтерфейсів користувачів WPF.

Документи XAML визначають розташування панелей, кнопок та інших елементів керування, що становлять вікна в застосунку WPF. Малоймовірно, що вам доведеться писати код XAML вручну. Замість цього ви використовуєте інструмент, що генерує необхідний код XAML [10][1)].

За допомогою XAML насамперед описується інтерфейс. Вся логіка керується процедурним кодом (VB, JavaScript, C# і т.д.) XAML може бути використаний як і для браузерних додатків, так і для звичайних додатків.

XAML включає основні чотири категорії елементів:

– панелі;

– елементи управління;

– елементи, пов'язані з документом і графічні фігури.

Додатки, оголошені в XAML, можуть включати безліч сторінок. PageViewer – дозволяє розбивати зміст на сторінки і забезпечує навігацію по ним. ContextMenu – допомагає в створенні навігаційних меню програми.

XAML широко використовується в Windows Presentation Foundation

---

[1)] [10] Технології WPF-застосувань. URL: https://knute.edu.ua/file/MzU4Mw==/de88c81c5530e45a7d3f457361804cc4.html. (дата звернення 29.03.2020).

(WPF), Windows Workflow Foundation (WWF), .NETFramework3.0. В WPF XAML використовується як мова розмітки для користувача інтерфейсу, для визначення елементів призначеного для користувача інтерфейсу, прив'язки даних, підтримки подій і ін. властивостей.

XAML файли можна створювати і редагувати за допомогою інструментів візуального конструювання, таких як: Microsoft Expression Blend, Microsoft Visual Studio, WPF visual designer. Також, їх можна створювати за допомогою стандартного текстового редактора, редактора коду такого як: XAMLPad, або графічного редактора.

Все створене або реалізоване в XAML може бути виражено за допомогою більш традиційних .NET мов, таких як C# та ін. Однак, ключовим аспектом технології є зменшення складності використовуваних для обробки XAML інструментів, так як XAML заснований на XML. В результаті цього з'являється безліч продуктів, що створюють засновані на XAML додатки. Оскільки XAML базується на XML, у розробників і дизайнерів існує можливість одночасно працювати над вмістом без необхідності компіляції [11][1)].

Мова XAML включає деякі оптимізації, які роблять розмітку зручніше для людського сприйняття. Одна з оптимізацій полягає в тому, що якщо певна властивість приймає тип колекції, елементи, оголошені в розмітці як дочірні в межах значення цієї властивості, стають частиною колекції. В цьому випадку колекція дочірніх об'єктних елементів є значенням, яке задається для властивості колекції.

XAML відображає поточну функцію мови, за рахунок чого клас може призначити тільки одне зі своїх властивостей як властивість вмісту XAML. Дочірні елементи даного об'єктного елемента використовуються для завдання значення цієї властивості вмісту. Іншими словами, для властивості вмісту (і

[1)] [11] Відновлення файлів формату XAML. Керівництво для Windows, MacOS, Android і IOS систем в 2020. URL: https://datarecovery.in.ua/xaml-file-recovery. (дата звернення 05.04.2020).

тільки для нього) можна опустити елемент властивості, вказавши цю властивість в XAML-розмітці, і тим самим створити більш наочну метафору батьківського-дочірнього елементів в розмітці [11][1)].

2.2 Проектування системи UML-засобами

Моделювання системи проводиться як спуск по рівнях: від концептуальної моделі до логічної, а потім до фізичної моделі програмної системи. Концептуальна модель виражається у вигляді діаграм варіантів використання (use case diagram) [12][2)].

Цей тип діаграм служить для проведення ітераційного циклу загальної постановки завдання разом із замовником.

Логічна модель дозволяє визначити два різних погляди на систему: статичний і динамічний. Ці погляди перетворюються у різні підходи до моделювання системи. Статичний підхід виражається діаграмами класів (class diagram). Саме діаграми класів є основою для генерації програмного коду цільовою мовою програмування.

Можливе дуже гнучке настроювання генерації коду з врахуванням конкретних угод (наприклад, за префіксами імен ідентифікаторів), прийнятих у команді розробників проекту [13][3)].

У найпростішому випадку варіант використання створюється в процесі обговорення з користувачами тих речей, які вони хотіли б одержати від системи. При цьому кожній окремій функції привласнюється ім'я й записується її короткий текстовий опис.

[1)] [11] Відновлення файлів формату XAML. Керівництво для Windows, MacOS, Android і IOS систем в 2020. URL: https://datarecovery.in.ua/xaml-file-recovery. (дата звернення 05.04.2020).

[2)] [12] Общая характеристика языка UML. URL: http://www.informicus.ru/default.aspx?SECTION=6&id=73&subdivisionid=2. (дата звернення 23.03.2020).

[3)] [13] Что такое Юзкейс (Use Case) или "Сценарий Использования" в Тестировании ПО? URL: https://software-testing.org/testing/chto-takoe-yuzkeys-use-case-ili-scenariy-ispolzovaniya-v-testirovanii-po.html. (дата звернення 06.04.2020).

Це все, що необхідно у фазі аналізу. Знання деяких деталей може знадобитися, якщо передбачається, що даний варіант використання містить важливі архітектурні відгалуження. Більшість варіантів використання може бути деталізована під час конкретної ітерації в процесі проектування [13][1)].

Розробка діаграми варіантів використання переслідує цілі:

- визначити загальні межі і контекст модельованої предметної області на початкових етапах проектування системи;
- сформулювати загальні вимоги до функціонального поведінки проектованої системи;
- розробити вихідну концептуальну модель системи для її подальшої деталізації у формі логічних і фізичних моделей;
- підготувати вихідну документацію для взаємодії розробників системи з її замовниками і користувачами.

Суть даної діаграми складається в наступному: проектована система представляється у вигляді безлічі сутностей або акторів, що взаємодіють з системою за допомогою так званих варіантів використання.

При цьому актором (actor) або дійовою особою називається будь-яка сутність, що взаємодіє з системою ззовні. Це може бути людина, технічний пристрій, програма або будь-яка інша система, яка може служити джерелом впливу на моделируемую систему так, як визначить сам розробник.

У свою чергу, варіант використання (use case) служить для опису сервісів, які система надає актору. Іншими словами, кожен варіант використання визначає деякий набір дій, який чинять системою при діалозі з актором. При цьому нічого не говориться про те, яким чином буде реалізовано взаємодію акторів з системою [14][1)].

Use-Case діаграму для об’єкту розробки представлено на рисунку 6:

[1)] [13] Что такое Юзкейс (Use Case) или "Сценарий Использования" в Тестировании ПО? URL: https://software-testing.org/testing/chto-takoe-yuzkeys-use-case-ili-scenariy-ispolzovaniya-v-testirovanii-po.html. (дата звернення 06.04.2020).

[1)] [14] Диаграмма вариантов использования (use case diagram) URL: http://khpi-iip.mipk.kharkiv.edu/library/case/leon/gl4/gl4.html. (дата звернення 06.04.2020).

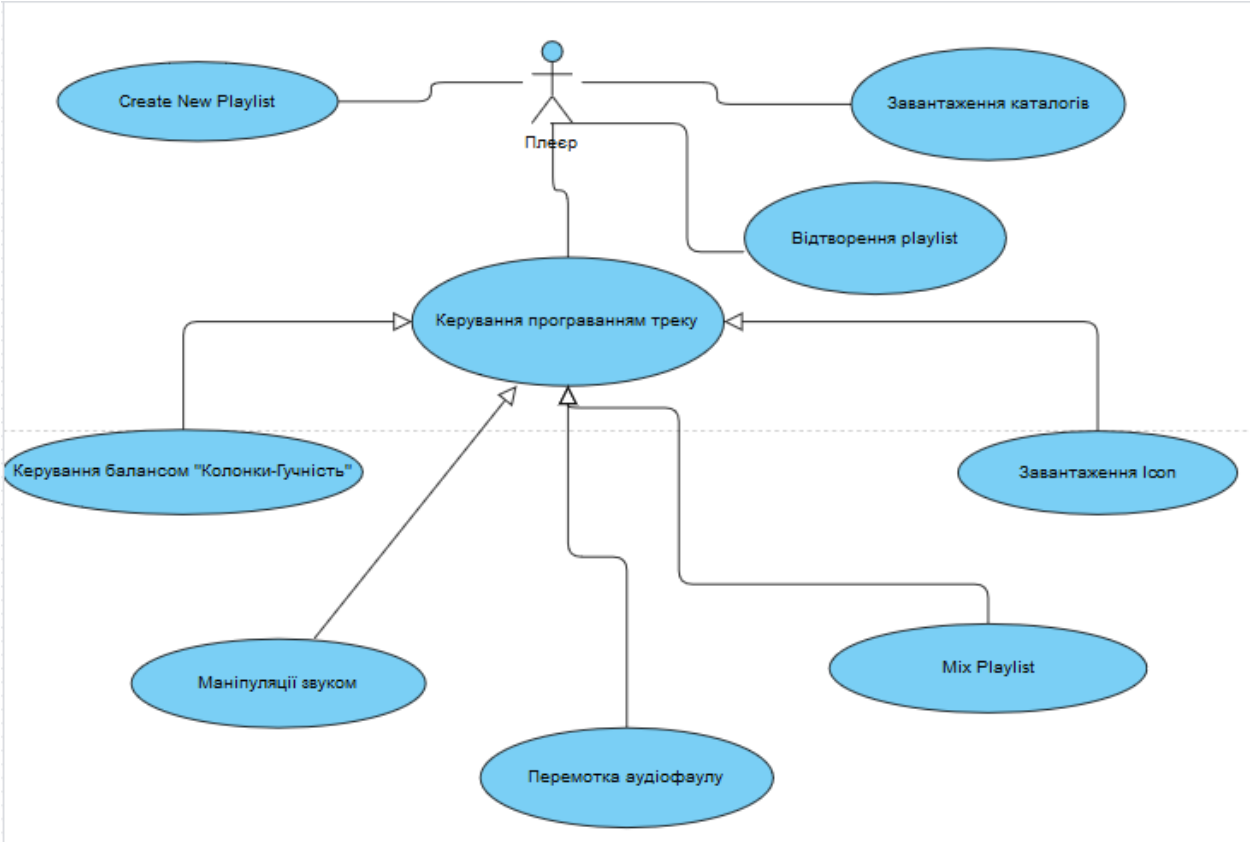

Рисунок 6 – Діаграма Use-Case

2.3 Проектування макету інтерфейсу програмного додатку

Одним із етапів проектування є створення макетів для інтерфейсу користувача. У 1990 році Якоб Нільсен сформулював 10 евристик юзабіліті для оцінки користувальницьких інтерфейсів.

Ці правила пройшли перевірку часом, і тепер у дизайнерів є простий спосіб оцінити, наскільки зручний інтерфейс програмного забезпечення, з опорою на універсальні принципи дизайну.

Десять евристик Нільсена:

- поінформованість про стан системи: користувачі завжди повинні знати, що на даний момент відбувається з системою. Для цього необхідна повноцінна і швидка зворотний зв'язок;

- схожість системи з реальним світом: система повинна спілкуватися з користувачем на зрозумілій йому мові. Краще використовувати слова, фрази і поняття, з якими користувач знаком, а не вузькі терміни.

Все повинно відбуватися, як в реальному світі, де інформація з'являється природно і логічно. Крім цього потрібно враховувати наступне:

- свобода дій і контроль: користувачі часто вибирають деякі функції системи помилково, і їм необхідна кнопка «аварійного виходу», щоб повернути попередній стан системи, не вступаючи в тривалий діалог.
- единбурзі і стандарти: користувач не повинен сидіти і ворожити, чи означають різні слова, ситуації і дії різний або одне і те ж. Слід дотримуватися загальноприйнятих для платформи правил [15][1)].
- запобігання помилок: краще хороших повідомлень про помилки може бути тільки продуманий дизайн, який допоможе запобігти проблему взагалі;
- на увазі, а не по пам'яті: слід не завантажувати пам'ять користувача, нехай всі об'єкти, дії і опції будуть на увазі. Переміщаючись по системі, користувач не повинен запам'ятовувати інформацію − все інструкції до системи повинні бути видні або легко витягатися при необхідності;
- гнучкість і ефективність: функції, що прискорюють взаємодія, яких не помітить новачок, можуть відчутно допомогти досвідченому користувачеві, і потрібно, щоб система була зручною для користувачів будь-якого рівня;
- естетика і мінімалізм в дизайні: у діалогах не повинно бути зайвої інформації. Все зайве в діалозі буде конкурувати з потрібною інформацією, роблячи її менш помітною для користувача;

[1)] [15] Usability Heuristics for Bots. URL: https://chatbotsmagazine.com/usability-heuristics-for-bots-7075132d2c92. (дата звернення 14.04.2020).

- розуміння проблем та їх вирішення. Повідомлення про помилки повинні бути написані простою мовою (ніяких кодів), чітко вказувати на проблему і пропонувати конструктивне рішення;
- довідкові матеріали і документи: якби система працювала без документації, але іноді в ній з'являється необхідність [15][1)].

У користувача не повинно виникати проблем при пошуку будь-корисної інформації. Вона повинна відповідати завданням користувача, вказувати, що конкретно потрібно зробити, і не бути дуже об'ємною. Макет інтерфейсу програмного додатку представлено на рисунку 7:

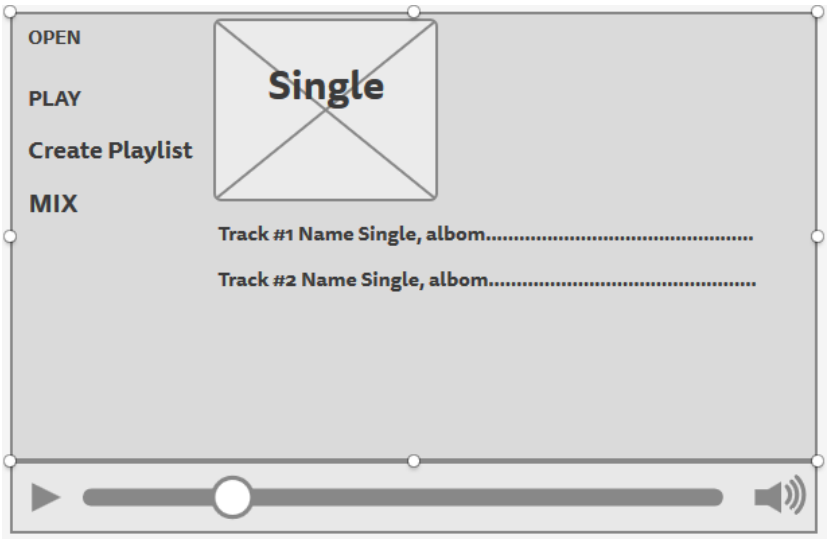

Рисунок 7 – Макет інтерфейсу програмного додатку

[1)] [15] Usability Heuristics for Bots. URL: https://chatbotsmagazine.com/usability-heuristics-for-bots-7075132d2c92. (дата звернення 14.04.2020).

3 РЕАЛІЗАЦІЯ ОБ’ЄКТУ РОЗРОБКИ

3.1 Опис інтерфейсу користувача

Після запуску програми на екрані з’являється вікно (рис. 8), у якому пропонується додати каталоги з файлової системи Windows у яких розміщені аудіофайли. Меню знаходиться в лівій частині вікна. Натиснувши «Додати розміщення», відчиняється провідник Windows (рис. 9), у якому можна обрати каталог.

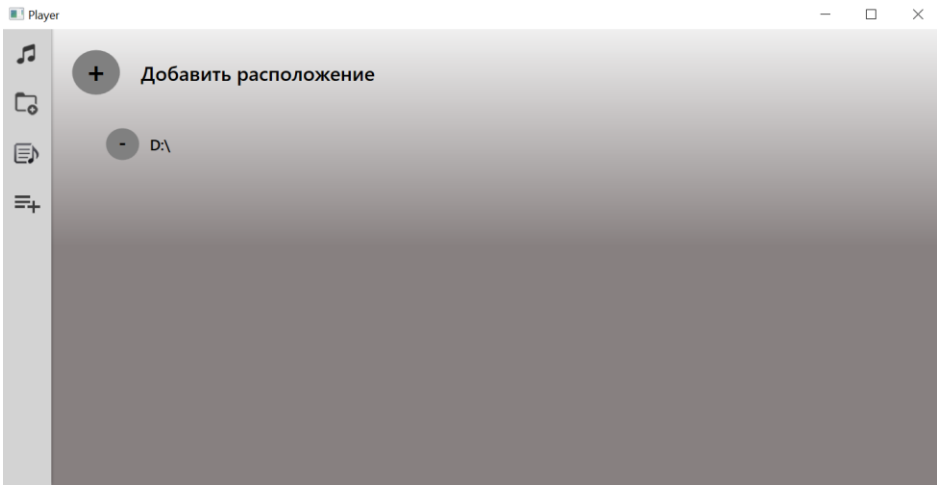

Рисунок 8 – Вікно розміщень аудіофайлів

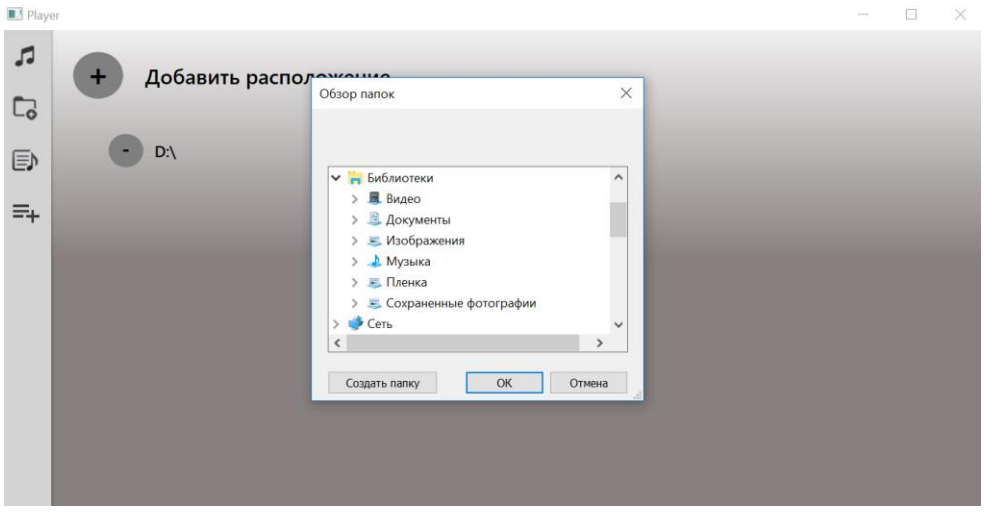

Рисунок 9 – Вікно додавання нового розміщення

Натиснувши на верхній пункт у меню «Відтворити» екрані буде показано список аудіофайлів знайдених у обраних каталогах (рис. 10). Після натискання на будь який аудіофайл з’являються панелі керування програванням (рис. 11).

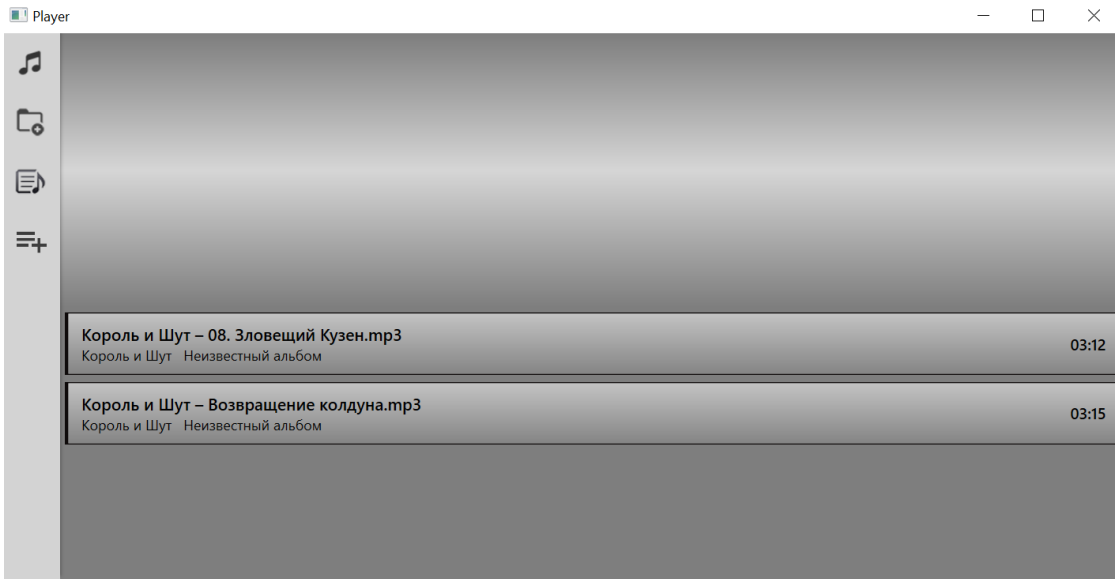

Рисунок 10 – Список аудіофайлів

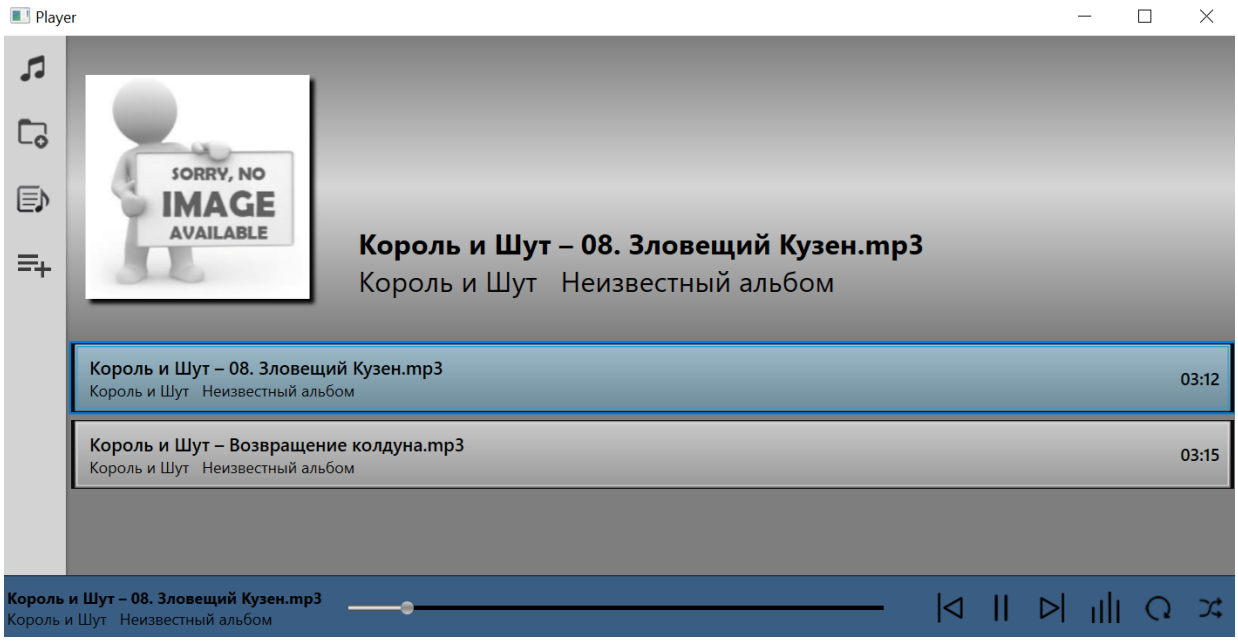

Рисунок 11 – Панель керування програванням

На панелі керування програванням (рис. 11) відображається назва файлу, назва композиції, зображення альбому, виконавець та назва альбому. Можна керувати перемоткою, переключенням треків, паузою, типом програвання: циклічне та випадковий вибір треку. Також можна керувати балансом між колонками і гучністю (рис 12).

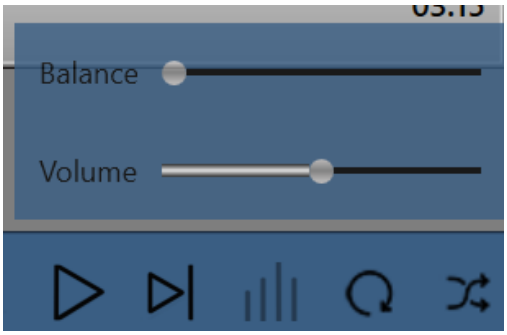

Рисунок 12 – Керування гучністю

Третій пункт плану відкриває вікно керування списками відтворення (рис 13), а четвертий пункт дозволяє створити новий плейлист.

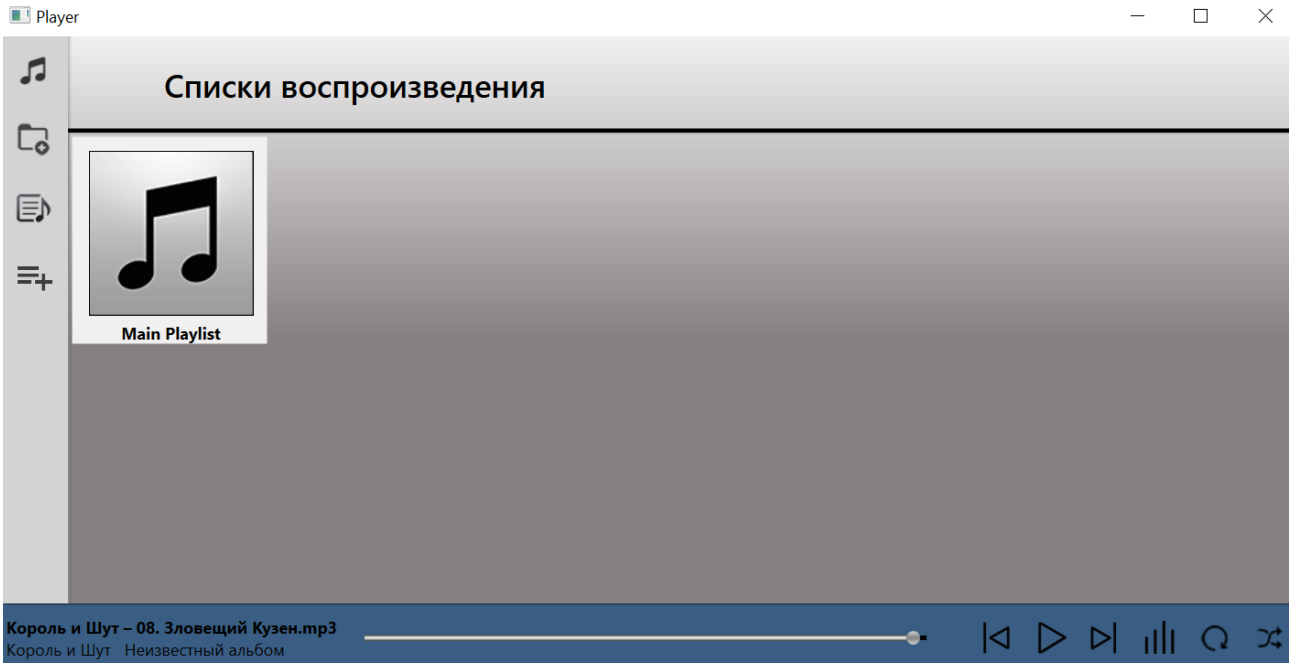

Рисунок 13 – Списки відтворення

Натиснувши на назву плейлиста (рис. 14), відкривається вікно редагування листа, у якому можна перейменувати лист, видалити лист, додати чи видалити композиції.

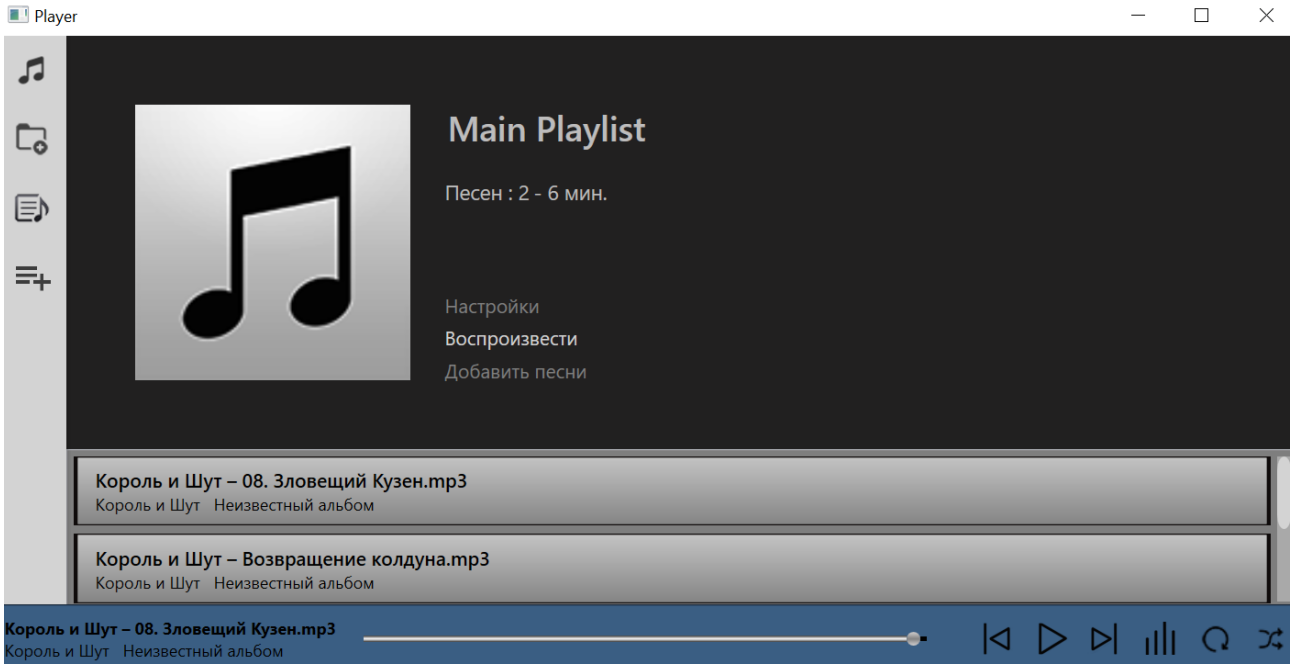

Рисунок 14 – Редагування плейлисту

3.2 Опис програмного коду основних класів

При розробці додатку не використовувалися сторонні бібліотеки для відтворення аудіо. Стандартний набір бібліотек Visual Studio має наступні класи для відтворення аудіо:

- SoundPlayer;
- SoundPlayerAction;
- SystemSounds.

Класи SoundPlayer, SoundPlayerAction і SystemSounds легко використовувати, але всі вони відносно малопотужні. У сучасному світі, замість справжнього формату WAV набагато більш поширений стислий формат звуку MP3 для всіх цілей, за винятком найпростіших звуків.

Для відтворення MP3-аудіо або MPEG-відео призначені класи MediaPlayer і MediaElement. Обидва вони залежать від ключових елементів технології, наданої програвачем Windows Media.

Клас MediaPlayer (що знаходиться в специфічному для WPF просторі імен System.Windows.Media) – це WPF-еквівалент класу SoundPlayer. Хоча ясно, що він не настільки легковагий, все ж він працює приблизно так само. Об'єкт MediaPlayer створюється, за допомогою методу Open () і завантажується аудіофайл, а викликом методу Play() запускається асинхронне відтворення. Опція синхронного відтворення не передбачена.

MediaPlayer створюється поза обробника подій, тому він існує на протязі життєвого циклу вікна. Причина в тому, що метод MediaPlayer.Close() викликається тоді, коли об'єкт MediaPlayer видаляється з пам'яті. Якщо створити об'єкт MediaPlayer в обробнику подій, то він буде видалений з пам'яті майже негайно і, ймовірно, незабаром після цього буде видалений складальником сміття, і тоді буде викликаний метод Close() і відтворення перерветься.

Обов'язково повинен бути створений обробник події Window.Unloaded, в якому викликається метод Close() для зупинки будь-якого відтворюється в даний момент звуку при закритті вікна.

Місцезнаходження файлу вказується у вигляді URI. На жаль, це не синтаксис упакованих URI, так що вбудувати аудіофайл і відтворити його з використанням класу MediaPlayer не вийде. Це обмеження пояснюється тим, що клас MediaPlayer побудований на функціональності, яка не є вбудованою в WPF, а надана окремим, некерованим компонентом програвача Windows Media.

Код обробки винятків відсутня. Це обурливо, але методи Open() і Play() не генерують виключення (в деякій мірі цьому винна процеси асинхронної завантаження і відтворення). Натомість пропонується самостійно обробляти події MediaOpened і MediaFailed, щоб визначити, чи було запущено відтворення аудіо.

MediaPlayer досить простий, хоча володіє великими можливостями, ніж SoundPlayer.

MediaElement – це елемент WPF, службовець оболонкою функціональності класу MediaPlayer. Подібно до всіх елементів, MediaElement поміщається безпосередньо в призначений для користувача інтерфейс.

У разі застосування MediaElement для відтворення аудіо його розташування не має значення, але якщо відтворюється відео, він повинен бути розміщений там, де планується відображатися відео-вікно.

3.2.1 Програмне відтворення аудіо

Зазвичай в тонкому управлінні відтворенням необхідності немає. Наприклад, може знадобитися, щоб в певний момент воно було запущено, постійно відтворювалося повторно і т.д. Один спосіб досягти цього полягає в застосуванні методів класу MediaElement в належний момент.

Поведінка MediaElement при запуску визначається його властивістю LoadedBehavior, яке є одним з декількох властивостей, які додані в класі MediaElement і відсутні в класі MediaPlayer.

Властивість LoadedBehavior приймає будь-яке значення з перерахування MediaState. За замовчуванням воно дорівнює Play, але можна також використовувати Manual – в цьому випадку файл завантажується, а за запуск відтворення в потрібний момент відповідає код. Інше значення цієї властивості – Pause, установка якого призупиняє відтворення, але не дозволяє використовувати методи відтворення. Замість цього доведеться запускати відтворення за допомогою тригерів і розкадровки.

Клас MediaElement також надає властивість UnloadedBehavior, що визначає, що відбудеться під час вивантаження елемента. В даному випадку єдиним осмисленим вибором може бути Close, оскільки це закриє файл і звільнить все системні ресурси.

У типовому мультимедійних програвачів можна виконувати типові команди на зразок відтворення, паузи і зупинки більш ніж одним способом. Очевидно, що це відмінне місце для того, щоб застосувати модель команд WPF. Фактично, для цього існує клас команд, який вже включає деяку зручну інфраструктуру, і цей клас – System.Windows.Input.MediaCommands. Однак MediaElement не має ніяких прив'язок команд за замовчуванням, які підтримують клас MediaCommands.

На розробника покладається завдання написання логіки обробки подій, яка реалізує кожну команду і викличе відповідний метод MediaElement. Можна організувати код так, щоб кілька елементів призначеного для користувача інтерфейсу були прив'язані до однієї і тієї ж команди для скорочення дублювання коду.

MediaElement не генерує виняток, якщо не може знайти або завантажити файл. Замість цього пропонується обробити подію MediaFailed.

Використання класів MediaPlayer та MediaElement дозволяє реалізувати відтворення аудіо наступних форматів:

- aif;
- m3u;
- m4a;
- mid;
- mp3;
- mpa;
- wav;
- wma.

У класів MediaElement та MediaPlayer є один недолік, неможливість читати теги аудіо файлу. Для читання тегів використано бібліотеку taglib. TagLib – бібліотека для читання і редагування метаданих, вбудованих в аудіофайли. Вона здатна читати і редагувати всі відповідні формати метаданих для аудіофайлів, включаючи APEv2, ID3 і Vorbis коментарі. Він може знайти теги

в ряді різних форматів (.mp3, .ogg, .spx, .mpc, .ape, .FLAC, .wv, .tta, .wma, .m4a, M4B, m4p, MP4, 3G2, .WAV, .aif (e), .opus. Також підтримується Unicode.

TagLib розроблений в C++ і не має ніяких залежностей під час виконання від іншого програмного забезпечення. Бібліотека поширюється як вільне програмне забезпечення, відповідно до положень або GNU Lesser General Public License (LGPL) або Mozilla Public License (MPL). Це не залежить від платформи і офіційно підтримує кілька Unix-like операційних систем (таких як Linux і OS X), і Windows.

Відповідно до визначення в специфікації JSP бібліотека тегів – це колекція дій, інкапсулюючих деяку функціональність, які можуть використовуватися з сторінки JSP. Тобто, бібліотека тегів являє собою набір java-класів, що реалізують певну бізнес-логіку відповідно до Tag Libraries Interface – інтерфейс бібліотеки тегів.

Структура тегів нагадує структуру сервлетів, які можуть бути багаторазово використані протягом циклу життя тегів. Бібліотека тегів включає один або декілька тегів-класів і XML-дескриптор, що містить опис тегів і параметрів, використовуваних тегами.

Застосування бібліотеки тегів нагадує використання компонентів JavaBean, оскільки в обох випадках частина дій і логіка виноситься за межі сторінки JSP, вирішуючи завдання модульного програмування інформаційних систем.

Істотною перевагою використання бібліотеки тегів перед компонентами JavaBean полягає в тому, що тег (java-клас) має доступ до сторінки JSP і може оперувати вмістом самої сторінки JSP.

Бібліотека тегів може визначати складні розрахунки в більш компактній формі. Однак установка бібліотек тегів вимагає більшої роботи в порівнянні з компонентами JavaBean. Компоненти JavaBean особливо ефективні для організації взаємодії між іншими компонентами, оскільки компонент JavaBean

може бути створений в одному Сервлет, а використовуватися іншими Сервлетами і JSP. Бібліотеки тегів, як правило, представляють більш замкнуту і самодостатню систему.

Код java-класу, оформлений у вигляді тега для виконання певних дій, приховує від користувача набір операцій, що визначають його функціональність. При використанні бібліотеки тегів розробка java-коду на сторінці JSP полягає в описі користувальницьких дескрипторів і визначенням необхідних атрибутів тега (java-класу).

Для використання / створення бібліотеки тегів необхідно розробити:

- клас (Tag handler class), що обробляє тег і визначає його функціональні властивості;
- XML-файл – дескриптор бібліотеки тегів Tag Library Descriptor (файл TLD) – зв'язує найменування елементів опису тегів з їх реалізацією.

Клас Audio створює сутність з інформацією про аудіо файл, а саме: назва треку, заголовок (ім’я файлу), шлях до каталогу, виконавець, назва альбому, довжина треку.

```
public class Audio
    {
        public string Name { get; set; }
        public string Title { get; set; }
        public string DirectoryName { get; set; }
        public string Singer { get; set; }
        public string Album { get; set; }
        public TimeSpan Duration { get; set; }
```

Конструктор класу Audio:

```
        public Audio(string Name, string Title, string Directo-
ryName, string Singer, string Album, TimeSpan Duration)
        {
            this.Name = Name;
            this.Title = Title;
            this.DirectoryName = DirectoryName;
            this.Singer = Singer;
            this.Album = Album;
            this.Duration = Duration;
```

```
            if(String.IsNullOrEmpty(this.Album))
            {
                this.Album = "Неизвестный альбом";
            }

            if(String.IsNullOrEmpty(this.Singer))
            {
                this.Singer = "Неизвестный исполнитель";
            }
```

Клас LoadAudio дозволяє робити пошук аудіо файлів у каталогах та додавати їх у список.

```
class LoadAudio
    {
```

Перелік форматів аудіо з якими працює розроблений плеєр.

```
string[] Formats = new string[] { ".aif", ".m3u", ".m4a", ".mid",
".mp3", ".mpa", ".wav", ".wma" };
```

Метод пошуку аудіо файлів у каталозі, що в якості результату повертає список з аудіо файлами.

```
public List<Audio> GetMusic(string Path)
    {
        List<Audio> AudioPathList = new List<Audio>();
        try
        {
            DirectoryInfo dir = new DirectoryInfo(Path);
```

Отримуємо список файлів вказаного каталогу у якості масиву, та у циклі перевіряємо усі файли:

```
FileInfo[] files = dir.GetFiles();
    foreach (FileInfo f in files)
      {
```

Якщо розширення файлу співпадає з дозволеним:

```
            if (Formats.Contains(f.Extension))
                {
```

За допомогою бібліотеки TagLib отримуємо теги аудіо файлу, створюємо об’єкт класу Audio та додаємо у список:

```
            TagLib.File FD = TagLib.File.Create(f.FullName);
             AudioPathList.Add(new Audio(f.Name, FD.Tag.Title, f.Di-
rectoryName, FD.Tag.FirstArtist, FD.Tag.Album, FD.Properties.Du-
ration));
                    }
                }
                return AudioPathList;
            }
```

Обов’язкова обробка виключення:

```
            catch (Exception e)
            {
                MessageBox.Show(e.Message);
                return AudioPathList;
            }
        }
    }
```

Клас PlayList розроблен для роботи з плейлистами, що дозволяє створити сутність плейлисту з списком треків, ім’ям та загальним часом:

```
public class PlayList
    {
        public List<Audio> AudioList { get; set; } = new List<Au-
dio>();
        public string Name { get; set; }
        public int AudioTime { get; set; }
```

Метод розрахунку загального часу треків плейлиста:

```
    public void GetTime()
        {
            AudioTime = 0;
            foreach(Audio val in AudioList)
```

```
            {
                AudioTime += val.Duration.Minutes;
            }
        }
    }
```

Основний код керування плеєром включає до себе наступні єтапи. Спочатку зберігаємо шлях до аудіо файлу у форматі Uri та передаємо об'єкту класу MediaElement:

```
ms.Source = new Uri(audioContext.DirectoryName + "\\" + audioCon-
text.Name);
            BottomPanel.Visibility = Visibility.Visible;
            TopPlayGrid.DataContext = audioContext; // обновляем
верхнюю панель
```

Запускаємо плеєр та перевіряємо наявність файлу передпрогаванням:

```
      Playing = true;
            ms.Play();

            CurrentIndex = CurrentList.IndexOf(audioContext);

            (PlayButton.Child      as      Image).Source      =
LoadImage(@"pack://application:,,,/Resources/Pause.png", false);

            if(!System.IO.File.Exists(audioContext.DirectoryName
+ "\\" + audioContext.Name))              {
```

Якщо такий файл не знайдено, то створюється діалогове вікно:

```
        MyDialogWindow dialog = new MyDialogWindow();
                dialog.TX.Text = "Этот файл был удален или пере-
мещен... Удалить данные о нем?";
                dialog.Owner = this;
                dialog.ShowDialog();
                if  (dialog.DialogResult.HasValue  &&  dialog.Di-
alogResult.Value)
                {
                    playLists  =  playLists.Select(x  =>  {  x.Audi-
oList = x.AudioList.Where
                        (s => s.Name != audioContext.Name || s.Di-
rectoryName != audioContext.DirectoryName).ToList(); x.GetTime();
return x; }).ToList();
```

з списку плейлисту створюється новий список, в якому не буде такого файлу:

```
                mainPL.AudioList = mainPL.AudioList.Where(x => 
x.Name != audioContext.Name || x.DirectoryName != audioContext.Di-
rectoryName).ToList();
```

Тіж самі дії виконуються і для головного плейлисту:

```
CurrentList = CurrentList.Where(x => x.Name != audioContext.Name 
|| x.DirectoryName != audioContext.DirectoryName).ToList(); // то 
же и для текущего списка воспроизведения
                          Play.ItemsSource = CurrentList;
```

Наступним етапом є оновлення головного списку аудіофайлів:

```
 if (CurrentList.Count > 0)
                    {
                        ChangeAudio(CurrentList[--
CurrentIndex]); // воспроизводим следующую песню
                    }
                    return;
                }
                else
                {
                    ChangeAudio(CurrentList[++CurrentIndex]);  // 
воспроизводим следующую песню
                    return;
                }
            }
```

За допомогою бібліотеки TagLib завантажуємо зображення з аудіо файлу, чи встановлюємо стандартне у разі відсутності.

```
      TagLib.File Ds = 
TagLib.File.Create(audioContext.DirectoryName + "\\" + 
audioContext.Name);
            if (Ds.Tag.Pictures.Length > 0)
            {
                TagLib.IPicture pic = Ds.Tag.Pictures[0];
                MemoryStream s = new 
MemoryStream(pic.Data.Data);
                s.Seek(0, SeekOrigin.Begin);
                BitmapImage bitmap = new BitmapImage();
```

```
                bitmap.BeginInit();
                bitmap.StreamSource = s;
                bitmap.CacheOption = BitmapCacheOption.OnLoad;
                bitmap.DecodePixelWidth = 200;
                bitmap.DecodePixelHeight = 200;
                bitmap.EndInit();
                s.Close();
                Img_Audio.Source = bitmap;
                GC.Collect();
            }
            else
            {
                Img_Audio.Source =
LoadImage(@"pack://application:,,,/Resources/NoImg.jpg", true);
            }
            BottomInfo.DataContext = audioContext;
        }
```

У зв’язку з тим, що велика кількість новітніх пристроїв має сенсорний екран потрібно включати підтримку цього. Застарілі бібліотеки Windows Forms не можуть розлічити короткого або довгого натиску на елемент, свайп по екрану. Тому для можливості роботи з сенсорним екраном використано WPF.

3.2.2 Сенсорний багатопозиційний ввод

Багатопозиційне введення – засіб взаємодії з додатком за допомогою дотику екрану. Він відрізняється від більш традиційного пір'яного введення тим, що дозволяє користувачеві працювати відразу декількома пальцями. У найбільш складному варіанті багатопозиційний введення розпізнає жести – особливі способи руху декількох пальців користувача як команди для виконання однієї з поширених операцій.

Наприклад, якщо доторкнутися до сенсорного екрану двома пальцями, а потім звести їх, то зазвичай це означає "зменшити масштаб", а рух одного пальця навколо іншого означає "повернути".

А оскільки користувач виконує ці жести безпосередньо на вікні, то кожен жест природно зв'язується з конкретним об'єктом. Наприклад, просте додаток з підтримкою такого введення може вивести на віртуальний робочий стіл кілька зображень і дозволити користувачеві впорядковувати їх за своїм бажанням: перетягувати, змінювати розмір і повертати кожне зображення. Так додаток демонструє лише невелику частину можливостей, але WPF дозволяє створити його практично без зусиль.

Багато розробників вважають, що багатопозиційний введення поступово стане звичайною справою при взаємодії з додатками на таких пристроях, як настільні комп'ютери і ноутбуки. Але поки підтримка подібного введення обмежена невеликим набором сенсорних планшетів, настільних моноблоків і РК-моніторів.

3.2.3 Рівні підтримки багатопозиційного введення

Як вище було описано, WPF дозволяє працювати з введенням з клавіатури і миші як на високому рівні (наприклад, клацання і зміни тексту), так і на низькому (руху миші і натиснення клавіш). Це важливо, тому що деяким додаткам необхідно набагато точніше управління, ніж іншим.

Все це відноситься і до багатопозиційними введення: WPF надає три окремих рівня його підтримки. А саме:

- просте дотик – це найнижчий рівень підтримки; він дає доступ до всіх торкань користувача. Його недолік в тому, що ваш додаток має саме зіставляти окремі повідомлення торкань і інтерпретувати їх. Обробка простих дотиків має сенс, якщо ви збираєтесь не розпізнавати стандартні жести, а реагувати на одночасні торкання особливим чином. Прикладом може служити програма малювання на зразок Windows 7 Paint, яка дозволяє користувачам "малювати" на сенсорному екрані відразу декількома пальцями;

- маніпуляції – цей рівень абстракції зручний для інтерпретації простих дотиків у вигляді осмислених жестів - приблизно так само, як елементи WPF інтерпретують послідовність подій MouseDown і MouseUp як подія більш високого рівня MouseDoubleClick. Елементи WPF зазвичай розпізнають прокручування, зміна масштабу, обертання і постукування;
- підтримка, вбудована в елементи – які елементи вже реагують на багатопозиційні події, без необхідності написання спеціального коду. Наприклад, елементи управління з прокруткою - такі як ListBox, ListView, DataGrid, TextBox і ScrollViewer - підтримують пальцеву прокрутку.

У преєрі користувач може за домогою сенсорного екрану натискати кнопки, легко відтворювати свайпи для прокрутки списків, чи керувати полосою відтворення треку або гучності.

ВИСНОВКИ

Що потрібно від хорошого програмного програвача? В першу чергу – всеїдність. Софт повинен розуміти всі актуальні для меломанів формати аудіофайлів. Не менш важливо зручність використання.

Крім того, плеєр повинен коректно виводити аудіопотоки на апаратний конвертор і делікатно проводити обробку звуку при необхідності.Об’єктом розробки є програмна реалізація аудіо плеєра з сенсорним управлянням. За вимогами до проекту плеєр повинен реалізовувати стандартні функції, такі як завантаження та створення плейлисту, управління рівнем звуку, відображення інформації на інтерфейсі щодо активного треку.

Крім цього, до задач плеєру додано сенсорне управління і відображення графічної інформації для кожного треку, реалізація оптимального інтерфейсу для користувача з урахуванням кольорової гами.

Під час аналізу предметної області було:

- розглянуті популярні аналоги аудіо плеєрів та поставлені вимоги до об’єкту проектування;
- обрана середа розробки і мова програмування для програмної реалізації плеєру.

Також, під час роботи над проектом отримані навики у роботі з технологією WPF і мовою розмітки XAML через середу розробки Visual Studio 2017.

В ході виконання дипломної роботи було отримано повнофункціоний програмний продукт «Аудіоплеєр для операційної системи Windows», повністю готовий до застосування.

Розроблена програма має можливість вибору аудіо файлу, регулювання гучності звуку, перемотування файлу, зміни треку, додавання треків до плейлисту, перегляд усіх можливих для програвання файлів в окремо обраних папках.

Графічний інтерфейс виконано з використанням технології WPF. Інтерфейс створеної програми зручний, простий, наочно відображає її можливості.

Після розробки, проект можна удосконалювати увесь час, робити його більш функціональним та більш зручним для користування.

Результат дипломного проектування повністю задовольняє всім вимогам, поставленим на етапі постановки завдання.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. 11 лучших музыкальных плееров для Windows. URL: https://geeker.ru/music/muzykalnyj-pleer-dlya-windows/. (дата звернення 18.03.2020).
2. Plex обновила iOS-версии плеера Plexamp и программы Plex Dash. URL: https://stereo.ru/news/plex-plexamp-plex-dash-update. (дата звернення 18.03.2020).
3. Рейтинг лучших музыкальных плееров для Windows. URL: https://vseprost.ru/rejting-luchshix-muzykalnyx-pleerov-dlya-windows.html. (дата звернення 18.03.2020).
4. Язык C#. Алексей Федоров. URL: https://compress.ru/article.aspx?id=11719. (дата звернення 20.03.2020).
5. C#. Объектно-ориентированный язык программирования. URL: https://habr.com/ru/hub/csharp/. (дата звернення 25.03.2020).
6. Microsoft Visual Studio. URL: https://open-file.ru/programs/microsoft-visual-studio. (дата звернення 25.03.2020).
7. Знакомство с Visual Studio. URL: https://habr.com/ru/sandbox/79099/. (дата звернення 26.03.2020).
8. Введение в WPF. Особенности платформы WPF. URL: https://metanit.com/sharp/wpf/1.php. (дата звернення 26.03.2020).
9. Windows Presentation Foundation (WPF). URL: http://sibakademsoft.ru/windows-presentation-foundation-wpf. (дата звернення 29.03.2020).
10. Технології WPF-застосувань. URL: https://knute.edu.ua/file/MzU4Mw==/de88c81c5530e45a7d3f457361804cc4.html. (дата звернення 29.03.2020).
11. Відновлення файлів формату XAML. Керівництво для Windows, MacOS, Android і IOS систем в 2020. URL:

https://datarecovery.in.ua/xaml-file-recovery. (дата звернення 05.04.2020).

12. Общая характеристика языка UML. URL: http://www.informicus.ru/default.aspx?SECTION=6&id=73&subdivisionid=2. (дата звернення 23.03.2020).
13. Что такое Юзкейс (Use Case) или "Сценарий Использования" в Тестировании ПО? URL: https://software-testing.org/testing/chto-takoe-yuzkeys-use-case-ili-scenariy-ispolzovaniya-v-testirovanii-po.html. (дата звернення 06.04.2020).
14. Диаграмма вариантов использования (use case diagram) URL: http://khpi-iip.mipk.kharkiv.edu/library/case/leon/gl4/gl4.html. (дата звернення 06.04.2020).
15. Usability Heuristics for Bots. URL: https://chatbotsmagazine.com/usability-heuristics-for-bots-7075132d2c92. (дата звернення 14.04.2020).