ПОЛТАВСЬКИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТОРГІВЛІ
Навчально-науковий інститут денної освіти
Форма навчання денна
Кафедра комп’ютерних наук та інформаційних технологій

Допускається до захисту
Завідувач кафедри
_____ Олена ОЛЬХОВСЬКА
(підпис)

«___» _____ 2024 р.

КВАЛІФІКАЦІЙНА РОБОТА

на тему
РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ЗАСТОСУНКУ ДЛЯ КЕРУВАННЯ МЕДІА КОНТЕНТОМ

зі спеціальності 122 Комп’ютерні науки
освітня програма «Комп’ютерні науки»
ступеня магістра

Виконавець роботи Лукашенко Дмитро Олександрович
_____ «____»_____ 2024 р.
(підпис)

Науковий керівник к.пед.н., доцент, Кошова Оксана Петрівна
_____ «____»_____ 2024 р.
(підпис)

Рецензент

ПОЛТАВА 2024 р.

ЗАТВЕРДЖУЮ
Завідувач кафедри _____Олена ОЛЬХОВСЬКА
(підпис)
«___» _____ 2023 р.

ЗАВДАННЯ І КАЛЕНДАРНИЙ ГРАФІК
ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ

на тему «Розробка програмного забезпечення застосунку для керування медіа контентом**»**
зі спеціальності 122 Комп'ютерні науки
освітня програма «Комп'ютерні науки»
ступеня магістр
Прізвище, ім'я, по батькові Лукашенко Дмитро Олександрович
Затверджена наказом ректора № 8-Н від «16» січня 2023 р.
Термін подання студентом роботи «___»_____ 2024 р.
Вихідні дані до кваліфікаційної роботи: статті та документації з теми розробки застосунків для керування медіаконтентом.

Зміст пояснювальної записки (перелік питань, які потрібно розробити)

ВСТУП
РОЗДІЛ 1. ПОСТАНОВКА ЗАДАЧІ
РОЗДІЛ 2. ОГЛЯД СИСТЕМ АНАЛОГІЧНОГО ПРИЗНАЧЕННЯ
2.1. Онлайн застосунки аналогічного призначення
2.2. Мобільні застосунки для керування відеоконтентом
2.3. Переваги та недоліки розробки ПЗ та використання застосунків для керування аудіо та відео контентом
РОЗДІЛ 3. ТЕОРЕТИЧНА ЧАСТИНА
3.1. Опис проектних рішень, інструментів та підходів до розробки ПЗ
3.2. Опис методології створення програмного забезпечення
РОЗДІЛ 4. ПРАКТИЧНА ЧАСТИНА
4.1. Опис та побудова діаграми класів розробленого застосунку
4.2. Інструкція з використання застосунка
ВИСНОВКИ
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ
ДОДАТОК А

Перелік графічного матеріалу: 1 аркуш блок-схем, 17 ілюстрацій.

Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали, посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Постановка задачі	Кошова О.П.		
Інформаційний огляд	Кошова О.П.		
Теоретична частина	Кошова О.П.		
Практична частина	Кошова О.П.		

Календарний графік виконання кваліфікаційної роботи

Зміст роботи	Термін виконання	Фактичне виконання
Вступ		
Вивчення методичних рекомендацій та стандартів та звіт керівнику		
Постановка задачі		
Інформаційний огляд джерел бібліотек та інтернету		
Теоретична частина		
Практична частина		
Закінчення оформлення		
Доповідь студента на кафедрі		
Доробка (за необхідністю), рецензування		

Дата видачі завдання «__» _____ 2023 р.

Здобувач вищої освіти _____ Лукашенко Дмитро Олександрович
(підпис)

Науковий керівник _____ к.пед.н., доц. Кошова О.П.
(підпис) (науковий ступінь, вчене звання, ініціали та прізвище)

Результати захисту кваліфікаційної роботи

Кваліфікаційна робота оцінена на _____
(балів, оцінка за національною шкалою, оцінка за ECTS)

Протокол засідання ЕК № ____ від «____» _____ 2024 р.

Секретар ЕК_____ _____
(підпис) (ініціали та прізвище)

Затверджую
Зав. кафедрою _____
к.ф.-м.н. Олена ОЛЬХОВСЬКА
«____» _____ 2023 р.

Погоджено
Науковий керівник _____
к.пед.н., Оксана КОШОВА
«____» _____ 2023 р.

План

дипломного проекту з фаху
спеціальності 122 Комп’ютерні науки
освітня програма 122 Комп’ютерні науки
на тему «Розробка програмного забезпечення застосунку для керування медіаконтентом»
Прізвище, ім'я, по батькові Лукашенко Дмитро Олександрович

ВСТУП
РОЗДІЛ 1. ПОСТАНОВКА ЗАДАЧІ
РОЗДІЛ 2. ОГЛЯД СИСТЕМ АНАЛОГІЧНОГО ПРИЗНАЧЕННЯ
2.1. Онлайн застосунки аналогічного призначення
2.2. Мобільні застосунки для керування відеоконтентом
2.3. Переваги та недоліки розробки ПЗ та використання застосунків для керування аудіо та відео контентом
РОЗДІЛ 3. ТЕОРЕТИЧНА ЧАСТИНА
3.1. Опис проектних рішень, інструментів та підходів до розробки ПЗ
3.2. Опис методології створення програмного забезпечення
РОЗДІЛ 4. ПРАКТИЧНА ЧАСТИНА
4.1. Опис та побудова діаграми класів розробленого застосунку
4.2. Інструкція з використання застосунка
ВИСНОВКИ
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ
ДОДАТОК А

Здобувач вищої освіти _____ Лукашенко Д. О.

(підпис)

«____» _____ 2023 р.

РЕФЕРАТ

Записка: 61 сторінка, 17 рисунків, 1 додаток, 12 літературних джерел.

Мета дипломної роботи розробка програмного забезпечення застосунку для керування медіаконтентом.

Об'єкт розробки - затосунок для керування медіаконтентом.

Методи дослідження – використання сучасних підходів для написання якісних застосунків для керування медіаконтентом; мови програмування C#, мови розмітки XML із використанням технології WinForms; середовища розробки Microsoft Visual Studio.

Здійснено аналіз застосунків схожого призначення, опрацьовані особливості ПЗ застосунків для керування медіаконтентом. Встановлено конкретну методологію для розробки програмного забезпечення. Складено діаграму класів ПЗ розробленого застосунку. Розроблено інтерфейс та програмне забезпечення застосунку та підготовлено користувацьку інструкцію.

ЗМІСТ

ВСТУП 7
РОЗДІЛ 1. ПОСТАНОВКА ЗАДАЧІ 10
РОЗДІЛ 2. ОГЛЯД СИСТЕМ АНАЛОГІЧНОГО ПРИЗНАЧЕННЯ 13
2.1. Онлайн застосунки аналогічного призначення 13
2.2. Мобільні застосунки для керування відеоконтентом 18
2.3. Переваги та недоліки розробки ПЗ та використання застосунків для керування аудіо та відеоконтентом 21
РОЗДІЛ 3. ТЕОРЕТИЧНА ЧАСТИНА 30
3.1. Опис проектних рішень, інструментів та підходів до розробки ПЗ 30
3.2. Опис методології створення програмного забезпечення 37
РОЗДІЛ 4. ПРАКТИЧНА ЧАСТИНА 41
4.1. Опис та побудова діаграми класів розробленого застосунку 41
4.2. Інструкція з використання застосунка 42
ВИСНОВКИ 45
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 47
ДОДАТОК А 49

ВСТУП

Актуальність. Актуальність розробки програмного забезпечення для керування медіаконтентом для студентів спеціальності «Комп’ютерні науки» є надзвичайно високою в сучасних умовах розвитку інформаційних технологій. Сьогодні студенти щодня стикаються з великою кількістю медіафайлів, зокрема аудіо, відео, зображеннями та презентаціями, обсяги яких постійно зростають. Це викликає необхідність у створенні ефективних інструментів для організації, зберігання та обробки таких даних.

З огляду на навчальний процес у галузі комп'ютерних наук, розробка застосунку для керування медіаконтентом є не лише корисною, а й необхідною для розвитку практичних навичок студентів. Такий проєкт дозволяє застосувати знання, набуті під час вивчення різноманітних дисциплін, зокрема програмування, роботи з базами даних, створення користувацьких інтерфейсів та алгоритмів обробки інформації. Це сприяє кращому розумінню та інтеграції різних технологій у реальних умовах.

У сучасному технологічному середовищі також важливою є мультиплатформенність програмного забезпечення. Студенти часто використовують різні пристрої, тому застосунок повинен підтримувати роботу на різних операційних системах та легко інтегруватися з іншими сервісами, наприклад, хмарними сховищами. Це підвищує гнучкість та зручність у використанні, роблячи розробку такого програмного забезпечення ще більш актуальною.

Зростання обсягів відео та аудіоконтенту, такого як аудіолекції та відеолекції, подкасти, інтерв'ю та навчальні записи, створює необхідність у використанні ефективних інструментів для організації, обробки та використання цих ресурсів. Медіаконтент стає все більш популярним у навчанні завдяки його мобільності та можливості сприйняття інформації під час виконання інших видів діяльності.

Розробка програмного забезпечення для керування медіаконтентом має наукову цінність, оскільки передбачає вирішення низки технічних та інтелектуальних задач. Вона включає в себе застосування сучасних методів роботи з аудіо та відеоданими, таких як алгоритми розпізнавання мови, індексації та пошуку за аудіофайлами, а також інтеграції з іншими системами управління медіаконтентом. Це сприяє розвитку практичних навичок студентів у роботі з обробкою аудіо та відео, створенні інтерфейсів користувача та розробці ефективних алгоритмів управління даними. Таким чином, робота над подібним програмним забезпеченням дозволяє майбутнім фахівцям у галузі комп’ютерних наук набути актуальних знань та вмінь, необхідних для вирішення реальних інженерних проблем.

Використання медіаконтенту у навчанні також позитивно впливає на оптимізацію освітнього процесу. Завдяки спеціалізованим застосункам студенти отримують можливість швидко знаходити потрібні медіафайли, прослуховувати чи дивитися їх у зручний для себе час, робити нотатки та закладки, що підвищує ефективність засвоєння матеріалу. Універсальність програмного забезпечення, яке може бути адаптоване під різні потреби студентів та викладачів, дозволяє розширити його функціональні можливості. Наприклад, можна додати модулі для автоматичного транскрибування аудіо, сортування файлів за тематикою чи рівнем складності, або інтегрувати систему з іншими освітніми платформами.

Наукова цінність такої роботи полягає також у можливості створення інноваційних рішень для аналізу та обробки медіаконтенту. Це включає дослідження ефективних методів кодування, стиснення медіафайлів, а також застосування штучного інтелекту для автоматичного розпізнавання та класифікації медіаінформації. Крім того, розробка таких систем сприяє прогресу у вивченні взаємодії людини з комп'ютером, оскільки інтеграція голосового управління та систем розпізнавання мови дозволяє створити більш інтуїтивно зрозумілі інтерфейси для користувачів.

Крім того, можливість адаптації та розширення функціоналу програмного забезпечення робить його універсальним для використання в різних навчальних контекстах. Наприклад, можна додати модулі для обробки аудіо та відео, створення нотаток або планування навчальних завдань, що збільшує практичну цінність застосунку.

Таким чином, розробка програмного забезпечення для керування медіаконтентом є актуальною не тільки для підвищення ефективності навчання студентів спеціальності «Комп’ютерні науки», але й для розв’язання актуальних завдань, пов'язаних з організацією та використанням навчального матеріалу в сучасному інформаційному середовищі.

Мета даної дипломної роботи полягає у розробці програмного забезпечення застосунку для керування медіаконтентом.

Об’єктом розробки є затосунок для керування медіаконтентом.

Предметом розробки є програмне забезпечення затосунку для керування відеоконтентом за допомогою мови програмування C# та Winforms.

Розроблений застосунок для керуваннння відеоконтентом реалізований за допомогою мови програмування C#, мови розмітки XML із використанням технології WinForms та середовища розробки Microsoft Visual Studio. WinForms (Windows Forms) є однією з найпоширеніших бібліотек для створення графічного інтерфейсу користувача (GUI) у середовищі Windows.

Пояснювальна записка включає чотири основних розділи: формулювання завдань, аналіз існуючих аналогів, теоретичні та практична частини. Структура роботи розрахована на логічне викладення матеріалу та повне розкриття теми дослідження.

РОЗДІЛ 1. ПОСТАНОВКА ЗАДАЧІ

Мета додатку

Метою розробки додатку для керування відеоконтентом є створення ефективного інструменту, який забезпечить зручне зберігання, організацію та використання відеоматеріалів у навчальному процесі студентів спеціальності «Комп'ютерні науки». Додаток повинен надавати користувачам можливість швидкого доступу до відеофайлів, їх сортування за тематикою, тривалістю або датою створення, а також здійснювати пошук за вмістом чи ключовими словами. Важливим елементом цієї розробки є підтримка різноманітних відеоформатів та інтеграція з іншими медіасистемами, що дозволить суттєво розширити функціональні можливості застосунку.

Крім базових функцій управління відеоконтентом, додаток має забезпечувати інструменти для обробки відео, зокрема виділення ключових фрагментів, додавання позначок і створення нотаток під час перегляду. Це дозволить користувачам не лише впорядковувати відеоматеріали, але й аналізувати їх зміст для глибшого розуміння та ефективного засвоєння навчального матеріалу. Іншим важливим завданням є створення інтуїтивно зрозумілого інтерфейсу, який спростить взаємодію користувача з програмою та забезпечить максимальну продуктивність і зручність у роботі.

Таким чином, головна мета полягає у розробці комплексного, адаптивного та функціонального інструменту, який не лише оптимізує роботу з відеоконтентом, але й сприятиме підвищенню ефективності навчального процесу та розвитку навичок студентів у галузі обробки відеоданих.

Основні функції застосунку

Основні функції та складові елементи, які повинні бути реалізовані у додатку для керування медіаконтентом, спрямовані на забезпечення повноцінної організації, обробки та використання відеоматеріалів. Однією з

ключових функцій такого застосунку є можливість ефективного зберігання відеофайлів з підтримкою різноманітних форматів. Для цього необхідно впровадити модуль для завантаження та конвертації файлів, що дозволить користувачам зберігати записи у зручному для них форматі, а також гарантувати сумісність з іншими пристроями та платформами.

Ще однією важливою складовою є система каталогізації та сортування відеоконтенту. Для оптимальної організації матеріалів застосунок має містити можливості створення та редагування категорій, підкатегорій, а також надавати інструменти для додавання тегів та метаданих до кожного файлу. Це забезпечить можливість швидкого доступу до потрібних матеріалів та полегшить їх пошук за різними критеріями, такими як тематика, дата створення або тривалість. Також необхідно впровадити розширену функцію пошуку, яка дозволить користувачам знаходити аудіозаписи за ключовими словами, з урахуванням інформації, що міститься у метаданих та транскрипціях.

Для більш глибокого використання відеоматеріалів застосунок повинен містити інструменти обробки аудіо та відео. Однією з таких функцій є автоматичне транскрибування, що дозволить перетворювати аудіозаписи на текст, спрощуючи процес аналізу та опрацювання інформації. Крім того, слід передбачити можливість створення нотаток та закладок під час прослуховування, що сприятиме більш детальному вивченню матеріалу та забезпечить можливість повернутися до важливих моментів запису. Функціонал додатку може бути розширений за рахунок інструментів для редагування аудіо та відео, зокрема виділення окремих фрагментів, регулювання гучності або швидкості відтворення, що підвищить його універсальність та практичність.

Інтерфейс користувача також є важливою складовою частиною такого додатку. Він повинен бути інтуїтивно зрозумілим, забезпечуючи зручну навігацію та доступ до основних функцій. Інтеграція з іншими сервісами та хмарними сховищами дозволить користувачам легко завантажувати та синхронізувати аудіоконтент між різними пристроями. Для підвищення

продуктивності та зручності роботи слід впровадити систему сповіщень та нагадувань про наявні матеріали або заплановані медіаактивності.

Таким чином, додаток для керування відеоконтентом буде містити в собі комплексний набір функцій, спрямованих на ефективне зберігання, організацію, обробку та використання відеоматеріалів. Це дозволить створити універсальний інструмент, що відповідатиме сучасним вимогам навчального процесу та сприятиме підвищенню якості освіти для студентів спеціальності «Комп'ютерні науки».

РОЗДІЛ 2. ОГЛЯД СИСТЕМ АНАЛОГІЧНОГО ПРИЗНАЧЕННЯ

2.1. Онлайн застосунки аналогічного призначення

Сучасний ринок інформаційних технологій пропонує ряд онлайн застосунків для керування медіаконтентом, які набули значної популярності та широкого використання серед користувачів різних категорій, включаючи студентів, викладачів і професіоналів у галузі цифрових медіа. Одним з найвідоміших рішень у цій сфері є застосунок VLC Media Player, який забезпечує підтримку великої кількості аудіо- та відеоформатів і пропонує розширений набір функцій для відтворення, сортування та управління медіаконтентом. Завдяки своїй гнучкості та відкритому коду, VLC Media Player є універсальним інструментом, який надає можливість налаштування під потреби користувачів та інтеграції з іншими системами.

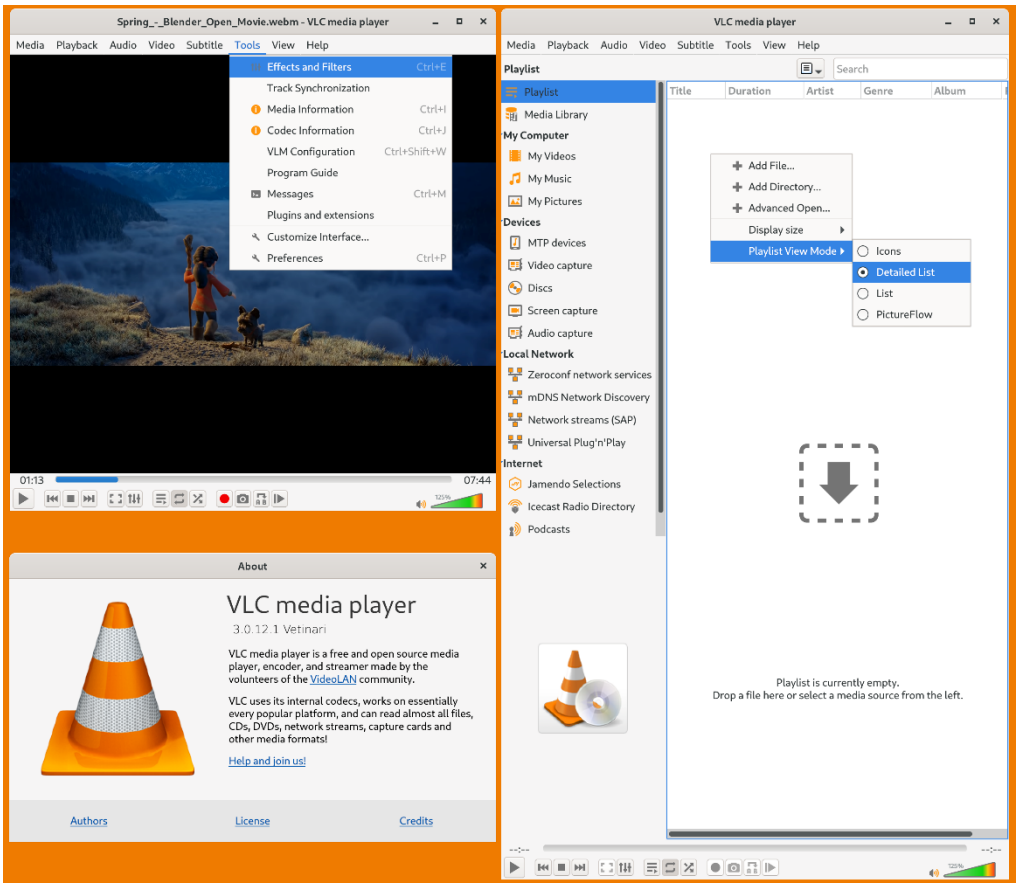

Рисунок 2.1. VLC media player

Іншим відомим застосунком є Plex, який дозволяє створювати власні медіасервери для організації, зберігання та потокової передачі медіаконтенту. Plex підтримує широкий спектр форматів, пропонує функції автоматичної каталогізації медіафайлів, а також забезпечує можливість спільного доступу до контенту з різних пристроїв. Його популярність пояснюється інтеграцією з хмарними сховищами, а також підтримкою різноманітних платформ, включаючи мобільні додатки та Smart TV. Це дозволяє використовувати застосунок як універсальне рішення для організації та відтворення медіаконтенту в домашніх та навчальних умовах.

Риснок 2.2. Застосунок Plex

Варто також відзначити застосунок iTunes, розроблений компанією Apple, який є одним із провідних інструментів для керування аудіоконтентом. iTunes дозволяє організовувати аудіобібліотеки, створювати списки відтворення та здійснювати покупку контенту через онлайн-магазини. Окрім функцій для керування аудіоконтентом, він пропонує можливості синхронізації з іншими пристроями Apple, що робить його популярним серед користувачів екосистеми Apple. Завдяки зручному та інтуїтивному інтерфейсу, iTunes забезпечує користувачів широкими можливостями для організації та використання музичного та подкаст-контенту.

Рисунок 2.3. Лого iTunes

Розглядаючи спеціалізовані інструменти для роботи з аудіоматеріалами, слід звернути увагу на застосунок Audacity. Це безкоштовний, відкритий аудіоредактор, який забезпечує багатофункціональне середовище для обробки аудіоконтенту. Audacity дозволяє користувачам записувати, редагувати та зберігати аудіофайли в різних форматах, використовуючи широкий набір інструментів для редагування звуку. Його можливості включають спектральний аналіз, багатодоріжковий монтаж та широкий спектр аудіоефектів, що робить його цінним інструментом для обробки та аналізу аудіоматеріалів у навчальних та наукових цілях.

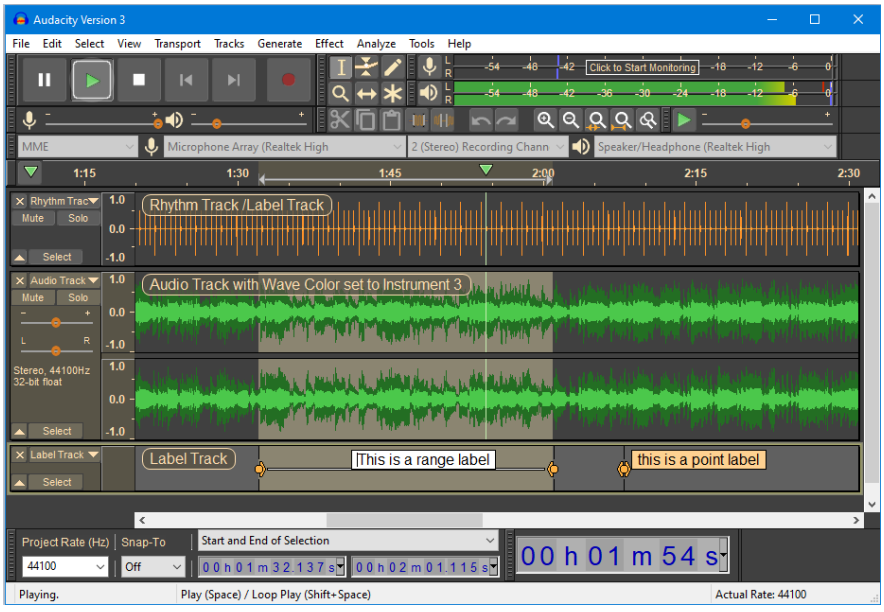

Рисунок 2.4. Інтерфейс Audacity

Водночас застосунки для керування аудіоконтентом є важливими інструментами в сучасному цифровому середовищі, особливо для студентів, викладачів, музикантів та всіх, хто працює з аудіоматеріалами. Існує ряд

спеціалізованих програм, призначених для ефективного управління, редагування та організації аудіофайлів. Кожен з таких застосунків пропонує унікальні можливості та функції, що сприяють оптимізації роботи з аудіоконтентом.

Одним з найпопулярніших застосунків для керування аудіофайлами є iTunes. Розроблений компанією Apple, iTunes дозволяє користувачам створювати та організовувати власні аудіобібліотеки, включаючи музику, подкасти, аудіокниги та лекції. Однією з ключових особливостей iTunes є можливість створення списків відтворення, що спрощує сортування та відтворення аудіоконтенту. Крім того, інтеграція з екосистемою Apple забезпечує просту синхронізацію аудіоматеріалів між пристроями, що є особливо зручним для користувачів, які використовують різні гаджети компанії. Завдяки вбудованому магазину контенту, iTunes також дозволяє купувати музику, подкасти та інший аудіоконтент, забезпечуючи доступ до широкого спектру матеріалів.

Іншим потужним інструментом є Audacity, безкоштовний та відкритий аудіоредактор, який пропонує розширені можливості для запису, редагування та обробки аудіофайлів. Audacity є універсальним застосунком, що дозволяє користувачам виконувати спектральний аналіз аудіо, застосовувати різноманітні ефекти, обрізати, з’єднувати або змінювати звукові доріжки. Особливо корисною є його можливість працювати з багатодоріжковими записами, що робить Audacity незамінним для створення аудіопроєктів, подкастів або редагування лекцій. Цей застосунок підтримує широкий спектр аудіоформатів, дозволяючи зберігати результати обробки у різних форматах, включаючи WAV, MP3 та OGG. Audacity також підтримує роботу з плагінами, що розширює його функціональні можливості для професійної обробки звуку.

Для більш просунутих користувачів, які працюють з великими бібліотеками аудіоконтенту, MusicBee пропонує високоефективні інструменти організації та відтворення. MusicBee дозволяє імпортувати музичні бібліотеки з різних джерел, автоматично впорядковувати файли за тегами та метаданими, а

також надає розширені можливості для налаштування відтворення аудіо. Вбудовані засоби конвертації файлів та підтримка різних форматів аудіо забезпечують гнучкість та зручність у роботі з контентом. Крім того, цей застосунок інтегрується з інтернет-радіо та подкастами, що дозволяє користувачам легко керувати поточним аудіоконтентом у реальному часі.

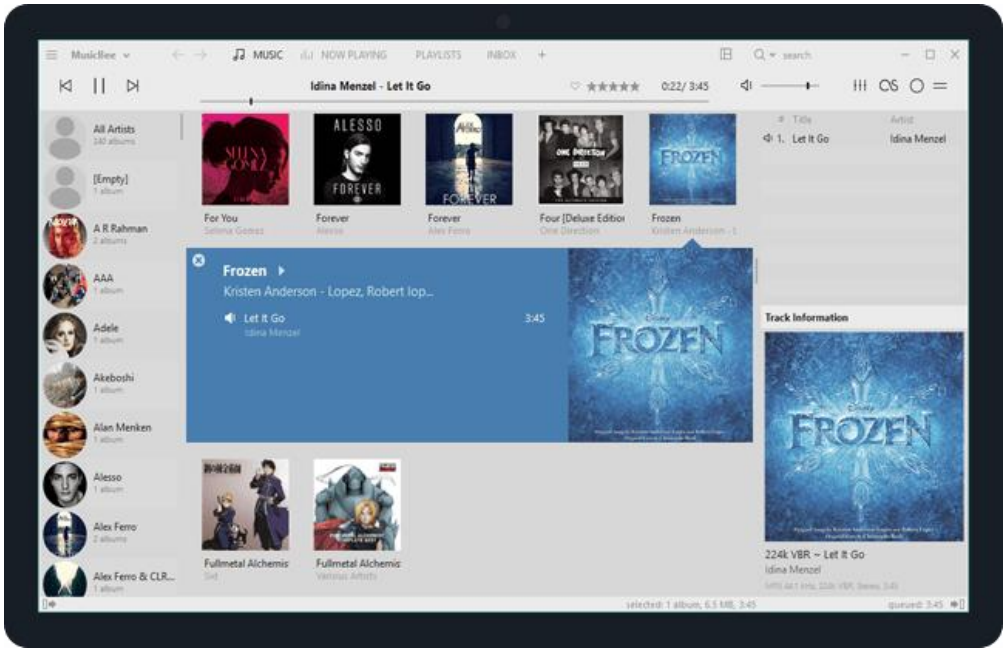

Рисунок 2.5. Інтерфейс MusicBee

Foobar2000 є ще одним відомим застосунком для управління аудіоконтентом, відомим своїм мінімалістичним дизайном та високим рівнем налаштування. Його функціональність включає підтримку великої кількості аудіоформатів, можливість редагування тегів, налаштування інтерфейсу користувача, а також конвертацію аудіофайлів. Foobar2000 також має розширену систему плагінів, яка дозволяє користувачам додавати додаткові функції, такі як підтримка мережевого відтворення та потокової передачі аудіо.

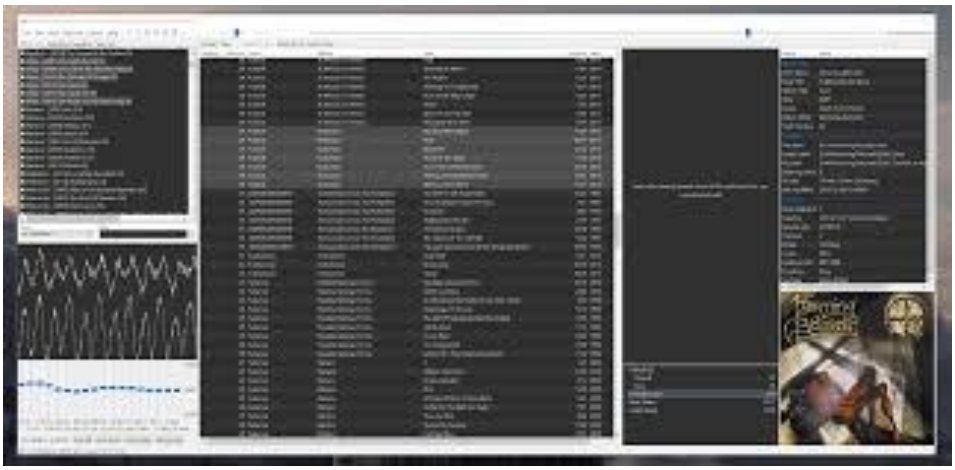

Рисунок 2.6. Інтерфейс Foobar2000

Таким чином, застосунки для керування медіаконтентом забезпечують різноманітні можливості для організації, обробки та використання відуо та аудіофайлів, сприяючи оптимізації навчального процесу та професійної діяльності. Кожен з розглянутих інструментів має свої унікальні особливості, які дозволяють користувачам обирати саме той застосунок, що найкраще відповідає їхнім потребам та специфіці роботи з аудіоматеріалами.

2.2. Мобільні застосунки для керування відеоконтентом

Застосунки для керування медіаконтентом на платформі Android відіграють важливу роль у повсякденному житті користувачів, забезпечуючи ефективну організацію, відтворення та управління різноманітними медіафайлами, включаючи музику, відео, подкасти та аудіокниги. Завдяки широкому спектру доступних застосунків користувачі мають змогу обирати інструменти, що відповідають їхнім індивідуальним потребам у роботі з аудіо- та відеоконтентом.

Одним з найпопулярніших медіаплеєрів на Android є VLC for Android, який є мобільною версією відомого медіаплеєра VLC. Цей застосунок підтримує велику кількість аудіо- та відеоформатів, включаючи MP3, MP4,

AVI, MKV, FLAC та інші. Окрім базових можливостей відтворення, VLC for Android забезпечує користувачам функції сортування медіафайлів за категоріями, створення списків відтворення та автоматичну організацію контенту, що спрощує його використання. Додатковою перевагою цього застосунку є можливість потокового відтворення медіа через мережу, що робить його універсальним інструментом для роботи з медіаконтентом будь-якого типу.

Рисунок 2.7. VLC for Android

Інший відомий застосунок - Poweramp, орієнтований переважно на аудіоконтент. Poweramp вирізняється розширеними функціями для відтворення музики, пропонуючи користувачам можливості налаштування еквалайзера, обробки звуку та редагування тегів. Цей медіаплеєр підтримує широкий спектр аудіоформатів і дозволяє організовувати бібліотеки за жанрами, альбомами, виконавцями та плейлистами. Додаток також забезпечує функції для відтворення без пауз, кросфейду та оптимізації звуку, що робить його одним з найкращих рішень для музичних ентузіастів. Інтуїтивний інтерфейс та можливість налаштування теми також сприяють популярності Poweramp серед користувачів Android.

Рисунок 2.8. Poweramp

Plex є ще одним багатофункціональним застосунком для керування медіаконтентом на Android, що забезпечує створення особистого медіасервера для потокової передачі аудіо та відеофайлів на різних пристроях. Завдяки підтримці синхронізації з хмарними сховищами, Plex дозволяє зберігати та організовувати великі бібліотеки контенту, забезпечуючи до них доступ з будь-якого місця. Користувачі можуть сортувати файли за жанрами, роками випуску або створювати власні колекції, що полегшує управління великими обсягами медіаконтенту. Інтеграція з онлайн-сервісами дозволяє Plex відображати метадані, обкладинки альбомів та додаткову інформацію, роблячи використання медіабібліотеки більш інформативним та зручним.

Рисунок 2.9. Plex

Для користувачів, які надають перевагу роботі з подкастами та аудіокнигами, застосунок Podcast Addict є одним із найкращих рішень на Android. Він надає можливість не лише прослуховувати та завантажувати подкасти, а й організовувати їх за тематикою, епізодами та відтворюваністю. Podcast Addict підтримує створення плейлистів, сортування за датою виходу, а також можливість налаштування швидкості відтворення, що є зручним для прослуховування навчального контенту. Інтеграція з іншими платформами дозволяє користувачам імпортувати RSS-стрічки та синхронізувати підписки між пристроями, забезпечуючи повний контроль над аудіоконтентом.

Рисунок 2.10. Podcast Addict

Таким чином, Android застосунки для керування медіаконтентом пропонують широкий спектр функцій для організації, відтворення та управління аудіо- та відеоматеріалами. Вони дозволяють користувачам ефективно працювати з медіафайлами, забезпечуючи зручність у використанні та оптимізацію процесу управління контентом. Кожен з розглянутих застосунків має унікальні можливості, що дозволяють задовольнити потреби різних категорій користувачів та сприяти ефективній роботі з медіаконтентом у повсякденному житті.

2.3. Переваги та недоліки розробки ПЗ та використання застосунків для керування аудіо та відеоконтентом

Розробка програмного забезпечення та використання застосунків для

керування аудіоконтентом має як переваги, так і недоліки, які потрібно враховувати під час створення і впровадження таких інструментів. Ці аспекти впливають на ефективність роботи з аудіоматеріалами та визначають можливості оптимізації навчальних або професійних процесів.

Однією з основних переваг розробки програмного забезпечення для керування аудіоконтентом є можливість адаптації функціоналу під конкретні потреби користувачів. Це дозволяє створювати універсальні рішення для студентів, викладачів, подкастерів чи музикантів, включаючи інструменти для організації, редагування, зберігання та пошуку аудіофайлів. Наприклад, такі функції, як автоматичне транскрибування аудіо, створення плейлистів, каталогізація за метаданими та пошук за ключовими словами, значно полегшують управління великими обсягами контенту та підвищують продуктивність. Додатково, сучасні застосунки надають можливості потокової передачі аудіо, інтеграції з хмарними сховищами та синхронізації між різними пристроями, що дозволяє користувачам працювати з контентом у будь-який час і з будь-якого місця.

Крім того, спеціалізовані аудіозастосунки сприяють кращій організації навчального процесу. Вони забезпечують швидкий доступ до навчальних матеріалів, аудіолекцій чи подкастів, дозволяють робити нотатки та закладки під час прослуховування. Це підвищує ефективність засвоєння інформації та допомагає студентам більш гнучко управляти своїм навчальним часом. Для професіоналів у галузі медіа такі застосунки є незамінними для монтажу аудіофайлів, роботи з багатодоріжковими записами та застосування різноманітних аудіоефектів.

Незважаючи на численні переваги, розробка та використання програмного забезпечення для керування аудіоконтентом має і свої недоліки. Основною проблемою є складність розробки та налаштування таких застосунків, яка вимагає значних технічних знань та ресурсів. Для забезпечення підтримки різноманітних аудіоформатів, багатофункціонального інтерфейсу та інтеграції з іншими платформами розробникам необхідно працювати з різними

технологіями та алгоритмами обробки аудіо. Це може призвести до збільшення витрат часу та фінансових ресурсів на реалізацію проєкту.

Ще одним недоліком є обмеженість деяких застосунків у підтримці різних форматів аудіо та можливостей інтеграції з іншими системами. Наприклад, деякі програми можуть не забезпечувати повноцінну роботу з нестандартними форматами аудіофайлів або не мати функцій для редагування та обробки звуку. Крім того, у випадку комерційних продуктів існує проблема вартості. Деякі професійні аудіозастосунки можуть бути дорогими, що може стати перешкодою для студентів або користувачів з обмеженим бюджетом.

Також, одним із викликів є питання безпеки та конфіденційності. Застосунки, що працюють з хмарними сервісами або синхронізуються через мережу, можуть піддаватися ризику несанкціонованого доступу або витоку особистих даних, особливо якщо йдеться про конфіденційні аудіозаписи. Це вимагає від розробників додаткових зусиль для забезпечення захисту інформації та впровадження надійних засобів шифрування та аутентифікації.

Таким чином, розробка та використання застосунків для керування аудіоконтентом мають суттєві переваги, пов'язані з ефективною організацією, обробкою та використанням аудіоматеріалів, що є важливим у навчальному та професійному середовищі. Однак, існують і певні недоліки, зокрема складність реалізації, обмежені можливості деяких програм, вартість та питання безпеки. Врахування цих факторів є необхідним для досягнення оптимального балансу між функціональністю та практичністю застосунків.

Розробка програм для керування аудіоконтентом вимагає використання різноманітних інструментів, технологій та мов програмування, які забезпечують обробку аудіофайлів, їхню організацію, зберігання та інтеграцію з іншими системами. Ці інструменти дозволяють створювати багатофункціональні застосунки, здатні працювати з різними форматами аудіо, надавати засоби редагування, транскрипції, відтворення та управління контентом.

Серед мов програмування, які найчастіше використовуються для

розробки програм для керування аудіоконтентом, особливе місце займає C++ . Ця мова відома своїми високими показниками продуктивності та гнучкості, що є важливими для розробки медіазастосунків, де обробка аудіоданих повинна бути швидкою та ефективною. Багато популярних аудіоплеєрів та редакторів, таких як Audacity та VLC, розроблені з використанням C++. Мова дозволяє працювати на низькому рівні з пам'яттю та процесорами, що важливо для оптимізації складних операцій, таких як обробка звуку в реальному часі, спектральний аналіз та застосування аудіоефектів.

Також C# можна використовувати для розробки застосунків для керування аудіоконтентом. Завдяки своєму універсальному характеру та широкому набору інструментів, C# є відмінним вибором для створення десктопних, мобільних та навіть веб-застосунків, які працюють з аудіофайлами. Найбільш поширеними середовищами для розробки таких програм є .NET та .NET Core (для кросплатформенних застосунків), а також Xamarin для мобільної розробки на iOS та Android.

Для роботи з аудіоконтентом на платформі .NET існують різноманітні бібліотеки та фреймворки, які спрощують обробку звуку. Серед них:

- NAudio : це одна з найпопулярніших бібліотек для роботи з аудіо у середовищі .NET. Вона надає широкий спектр можливостей, включаючи відтворення, запис, конвертацію аудіофайлів, а також застосування ефектів та фільтрів. NAudio підтримує різні формати файлів, такі як WAV, MP3, AAC, FLAC та інші. Завдяки своїй гнучкості та широкому набору функцій, ця бібліотека підходить для створення аудіопрогравачів, редакторів аудіо та інших спеціалізованих програм.

- CSCore : це ще одна бібліотека для обробки звуку в C#. Вона надає більш низькорівневий доступ до обробки аудіосигналів і дозволяє працювати з різними форматами та джерелами звуку. CSCore можна використовувати для розробки застосунків, які потребують високого рівня контролю над аудіопотоками, наприклад, в аудіоредакторах або програмах для живого мікшування звуку.

- Xamarin : для розробки мобільних застосунків на iOS та Android за допомогою C# можна використовувати фреймворк Xamarin. Він дозволяє створювати кросплатформні програми з єдиною кодовою базою, використовуючи спеціальні бібліотеки, такі як Xamarin.AudioManager або Plugin.MediaManager , для роботи з аудіофайлами та управління відтворенням аудіо.

Хоча більшість аудіоплеєрів та професійних аудіоредакторів традиційно розробляються на C++ через його високу продуктивність у низькорівневій обробці звуку, існують і деякі програми, створені з використанням C#. Одним з таких прикладів є MusicBee .

- MusicBee : це потужний музичний менеджер та плеєр, розроблений на C# із застосуванням фреймворку .NET. MusicBee пропонує багатий набір функцій для організації музичної бібліотеки, відтворення аудіо різних форматів, створення плейлистів, редагування тегів та багато іншого. Завдяки своїй гнучкості та налаштовуваному інтерфейсу, MusicBee став одним із найпопулярніших медіаплеєрів для Windows.

Крім MusicBee, багато внутрішніх та спеціалізованих програм для роботи з аудіоконтентом також розробляються на C#. Наприклад, корпоративні інструменти для обробки аудіо в кол-центрах, програми для медіавиробництва або освітні програми, які вимагають інтеграції з іншими системами, часто створюються з використанням технологій .NET через їхню стабільність та можливість інтеграції з іншими сервісами Microsoft.

Таким чином, C# є потужним інструментом для розробки застосунків для керування аудіоконтентом, завдяки своїм можливостям у роботі з аудіо, гнучкій обробці сигналів та інтеграції з різноманітними платформами. Використання бібліотек, таких як NAudio та CSCore, а також кросплатформних інструментів, таких як Xamarin, значно розширює можливості розробників, дозволяючи їм створювати багатофункціональні та зручні у використанні аудіопрограми.

Python також широко застосовується для розробки аудіозастосунків завдяки своїй простоті та багатству бібліотек. Серед найбільш популярних

бібліотек для обробки аудіо є PyDub, яка надає інструменти для обробки та аналізу аудіофайлів різних форматів. Крім того, бібліотека LibROSA використовується для аналізу та обробки музичних аудіосигналів, включаючи спектральні та часові характеристики звуку. Python є зручним для розробки прототипів, проведення аудіоаналізу та реалізації інструментів машинного навчання, таких як автоматичне розпізнавання мовлення.

Для мобільних застосунків, призначених для платформи Android, зазвичай використовують Java або Kotlin у поєднанні з Android SDK. Для роботи з аудіоконтентом на Android існує кілька бібліотек, таких як ExoPlayer для відтворення медіафайлів та OpenSL ES для низькорівневої обробки звуку. Використання цих інструментів дозволяє створювати потужні мобільні застосунки, що підтримують різноманітні аудіоформати, забезпечують налаштування відтворення та обробку аудіосигналів.

На платформі iOS для розробки застосунків для керування аудіоконтентом використовують Swift або Objective-C . Серед інструментів розробки особливе місце займає AVFoundation – фреймворк, який надає засоби для обробки, відтворення та запису аудіо. Крім цього, фреймворк Core Audio дозволяє працювати на низькому рівні з аудіоданими, включаючи обробку сигналів та використання аудіоефектів. Це дозволяє створювати високопродуктивні мобільні застосунки для iOS, що можуть конкурувати з професійними аудіоінструментами.

Для розробки кросплатформних застосунків часто використовують фреймворки Flutter (з мовою програмування Dart) та React Native (з мовою JavaScript). Вони дозволяють створювати застосунки для Android та iOS одночасно, спрощуючи процес розробки. Для роботи з аудіоконтентом у таких застосунках використовуються плагіни та бібліотеки, зокрема audioplayers для Flutter та react-native-sound для React Native, які забезпечують функціональність відтворення, запису та управління аудіо.

У контексті десктопних застосунків та професійних аудіоредакторів також широко використовуються JUCE та Qt. JUCE – це C++ фреймворк,

орієнтований на розробку аудіопрограм та плагінів. Він забезпечує інструменти для роботи з MIDI, аудіоефектами, обробкою звуку та створенням графічного інтерфейсу. Qt, у свою чергу, є потужним кросплатформним фреймворком, що надає інструменти для створення складних інтерфейсів та роботи з медіаконтентом на різних платформах, включаючи Windows, macOS та Linux.

Отже, розробка застосунків для керування аудіоконтентом вимагає застосування різноманітних мов програмування, бібліотек та фреймворків, які забезпечують ефективну обробку, відтворення та організацію аудіофайлів. Вибір інструментів залежить від платформи, цільової аудиторії та специфіки функціональних вимог до застосунку, що дозволяє розробникам створювати адаптивні та багатофункціональні програми для роботи з аудіоконтентом.

Розробка програм для керування відеоконтентом є складним процесом, що вимагає використання різноманітних інструментів, технологій та мов програмування. Створення таких програм передбачає обробку, зберігання, передачу та відтворення відеофайлів різних форматів, а також забезпечення зручного та функціонального інтерфейсу користувача. У зв’язку з цим, розробники використовують спеціалізовані бібліотеки, фреймворки та програмні засоби, які дозволяють ефективно працювати з відеоданими.

Однією з ключових технологій у цій галузі є FFmpeg — набір програм з відкритим кодом для обробки мультимедіа файлів. FFmpeg дозволяє конвертувати відеофайли з одного формату в інший, обрізати, з'єднувати, стискати відео, а також працювати зі звуковими доріжками. Його використовують у багатьох застосунках для керування відеоконтентом, оскільки він підтримує широкий спектр відео- та аудіоформатів. Цей інструмент написаний на мові програмування C , що забезпечує високу швидкість обробки даних. Бібліотеки FFmpeg інтегруються з іншими мовами програмування, такими як Python, JavaScript та C++, що дозволяє розробникам легко включати їх у власні проєкти.

Ще одним важливим інструментом для роботи з відео є OpenCV (Open Source Computer Vision Library). Це бібліотека з відкритим кодом, яка надає

широкий набір функцій для обробки відеозображень та комп'ютерного зору. OpenCV дозволяє розробникам виконувати аналіз відеопотоку в режимі реального часу, застосовувати фільтри, виявляти об'єкти та виконувати інші операції з відеоданими. Бібліотека написана на C++ та має обгортки для Python, Java та інших мов програмування, що робить її універсальним інструментом у проєктах з обробки відеоконтенту.

Мова програмування Python також широко використовується для розробки програм для керування відеоконтентом, оскільки має велику кількість бібліотек для обробки медіафайлів. Наприклад, бібліотека MoviePy дозволяє редагувати відео: вирізати сцени, додавати ефекти, працювати зі звуком та текстовими доріжками. Python також має бібліотеки для взаємодії з FFmpeg, що спрощує процес конвертації та обробки відеофайлів у різних форматах. Використання Python забезпечує гнучкість та швидкість у розробці програм, а наявність великої спільноти розробників дозволяє легко знайти підтримку та необхідні ресурси.

Для створення користувацьких інтерфейсів у застосунках для керування відеоконтентом часто використовуються фреймворки на базі JavaScript , такі як Electron та React . Electron дозволяє створювати кросплатформні настільні застосунки, використовуючи вебтехнології (HTML, CSS, JavaScript). Це є особливо зручним для розробників, які хочуть розробити застосунок для керування відеоконтентом, сумісний з різними операційними системами. React у свою чергу забезпечує можливість створення динамічних і гнучких користувацьких інтерфейсів, які можуть легко взаємодіяти з відеовмістом та надавати користувачам інструменти для редагування та організації контенту.

У мобільній розробці застосунків для керування відеоконтентом використовують фреймворки React Native та Flutter. React Native, що базується на JavaScript, дозволяє створювати кросплатформні застосунки для Android та iOS з використанням спільної кодової бази. Flutter, розроблений Google, використовує мову програмування Dart і дозволяє створювати високопродуктивні додатки з привабливими інтерфейсами, які можуть

ефективно відтворювати та обробляти відеоконтент.

На серверній стороні для зберігання та передавання відеоконтенту застосовуються різні платформи та сервіси, такі як AWS (Amazon Web Services), Google Cloud, та Microsoft Azure . Вони надають інструменти для потокової передачі відео, зберігання великих обсягів даних та забезпечення швидкого доступу до контенту з різних точок світу. Також для організації роботи з базами даних відеофайлів використовуються реляційні (MySQL, PostgreSQL) та нереляційні (MongoDB) бази даних, що дозволяє ефективно керувати метаданими та каталогами відеоконтенту.

Всі ці технології забезпечують можливість ефективної обробки, організації та відтворення відеоматеріалів, створення зручних користувацьких інтерфейсів та забезпечення надійної роботи програмного забезпечення як на настільних, так і на мобільних платформах.

РОЗДІЛ 3. ТЕОРЕТИЧНА ЧАСТИНА

3.1. Опис проектних рішень, інструментів та підходів до розробки ПЗ

Для розробки програмного забезпечення застосунку для керування відеоконтентом було використано наступні технології:

- мову програмування C#;
- мову розмітки XML;
- технологію створення графічного інтерфейсу користувача WinForms.
- середовище розробки Microsoft Visual Studio.

Мова програмування C#

Мова програмування C# є однією з найпопулярніших та найпотужніших мов у сучасній розробці програмного забезпечення, особливо у середовищі Windows [11, 12]. Розроблена компанією Microsoft, C# є об'єктно-орієнтованою мовою, що поєднує в собі елементи простоти та високої продуктивності, властиві мовам C-подібного синтаксису, з потужними інструментами для розробки комплексних додатків. Завдяки інтеграції з .NET Framework та .NET Core, C# забезпечує широкий спектр можливостей для створення різноманітних типів програм, включаючи настільні, веб- та мобільні застосунки, а також програми для керування мультимедійним контентом, зокрема відео.

Особливістю використання C# у розробці додатків для керування відеоконтентом є його висока сумісність з бібліотеками та фреймворками, орієнтованими на обробку мультимедіа. Одним із ключових інструментів є бібліотека FFmpeg, яка може бути інтегрована з C# через обгортки, такі як FFmpeg.AutoGen. Ця бібліотека дозволяє здійснювати різноманітні операції з відеофайлами, включаючи конвертацію, стиснення, обрізання та з'єднання відео, а також обробку звукових доріжок. Використання C# в цьому контексті спрощує взаємодію з такими складними інструментами, надаючи доступ до

їхніх функціональних можливостей за допомогою зрозумілого та інтуїтивного синтаксису.

Для створення графічного інтерфейсу користувача в застосунках для керування відеоконтентом C# пропонує кілька технологій, включаючи Windows Forms (WinForms), Windows Presentation Foundation (WPF) та Universal Windows Platform (UWP). Кожна з цих технологій надає засоби для побудови інтуїтивно зрозумілих інтерфейсів, у тому числі інструменти для відтворення відео, створення плейлистів та організації відеофайлів. Наприклад, WPF дозволяє використовувати елементи управління MediaElement для відтворення відео, підтримуючи різноманітні формати та забезпечуючи функції керування відтворенням, такі як пауза, перемотування та регулювання гучності. Крім того, WPF підтримує створення складних анімацій та візуальних ефектів, що робить його придатним для побудови сучасних та привабливих інтерфейсів.

C# також забезпечує глибоку інтеграцію з системними ресурсами операційної системи Windows, що дозволяє використовувати медіаінфраструктуру Windows Media Foundation (WMF). Це надає розробникам додаткові можливості для роботи з відеоконтентом, включаючи захоплення відео з камери, кодування та декодування відеопотоків, а також створення програм для обробки відеоданих у режимі реального часу. Використання WMF у поєднанні з мовою C# дозволяє розробляти продуктивні програми для керування відеоконтентом, які відповідають сучасним вимогам щодо ефективності та якості.

Крім того, C# підтримує асинхронне програмування за допомогою ключових слів `async` та `await`, що є особливо корисним при роботі з великими відеофайлами або під час обробки потокового відео. Це дозволяє розробникам створювати програми, які можуть одночасно виконувати обробку відеоданих та реагувати на взаємодію з користувачем без затримок, підвищуючи загальну продуктивність та зручність використання додатку.

Інтеграція з іншими інструментами та сервісами, такими як хмарні платформи Azure або Amazon Web Services (AWS), дозволяє розробникам

використовувати C# для створення застосунків, здатних керувати відеоконтентом у хмарному середовищі. Це відкриває можливості для масштабованого зберігання відеофайлів, потокової передачі та аналітики, а також для реалізації функцій спільного доступу та обробки відео в мережі.

Таким чином, C# є потужною та гнучкою мовою програмування, яка надає багатий набір інструментів та технологій для розробки застосунків для керування відеоконтентом. Її інтеграція з різними бібліотеками, підтримка графічного інтерфейсу користувача та можливості роботи з мультимедіа роблять C# одним з найкращих виборів для створення високоякісних програм у цій сфері.

Технологія створення графічного інтерфейсу WinForms

WinForms (Windows Forms) є однією з найстаріших та найпоширеніших бібліотек для створення графічного інтерфейсу користувача (GUI) у середовищі Windows [2, 10]. Ця технологія, що є частиною .NET Framework від компанії Microsoft, дозволяє розробникам створювати інтерактивні настільні застосунки з використанням мови програмування C#. WinForms базується на подієво-орієнтованій моделі, де кожен елемент інтерфейсу реагує на певні дії користувача, такі як натискання кнопок, введення тексту або взаємодія з меню. Завдяки цьому підходу розробники мають можливість створювати гнучкі та інтуїтивно зрозумілі програми, що спрощують роботу користувача з різноманітними функціями.

Однією з ключових особливостей WinForms є її простота у використанні та висока інтеграція з операційною системою Windows. Ця технологія надає широкий набір стандартних компонентів, включаючи текстові поля, кнопки, списки, меню та інші елементи управління, що можуть бути легко розміщені на формі та налаштовані за допомогою властивостей. Завдяки цьому розробники мають можливість створювати функціональні інтерфейси без потреби глибоких знань у низькорівневому програмуванні або роботі з графічними бібліотеками. Візуальний дизайнер WinForms у середовищі розробки Visual Studio спрощує

процес розробки інтерфейсу, дозволяючи розробникам додавати, редагувати та налаштовувати елементи управління за допомогою технології "drag-and-drop".

WinForms також забезпечує потужну підтримку роботи з подіями та обробки користувацьких дій. Це дозволяє розробникам легко реалізовувати реакції на різноманітні дії користувача, включаючи натискання кнопок, зміну значень у полях введення або вибір елементів зі списків. Така модель програмування спрощує створення інтерактивних та динамічних застосунків, що реагують на дії користувача в режимі реального часу. Крім того, можливості розширення та налаштування компонентів WinForms дозволяють розробникам створювати власні елементи управління, що відповідають специфічним вимогам певного проєкту.

WinForms тісно інтегрована з можливостями .NET Framework, що забезпечує доступ до різноманітних бібліотек та функціональностей, включаючи роботу з базами даних, файлами, мережевими протоколами та іншими системними ресурсами. Це дозволяє створювати комплексні настільні застосунки, що можуть взаємодіяти з іншими програмами та сервісами. Завдяки об’єктно-орієнтованому підходу до розробки, WinForms надає розробникам гнучкість у створенні застосунків із багатим функціоналом та можливістю легкого розширення.

Попри численні переваги, WinForms має певні обмеження. Через свою тісну прив’язку до Windows ця технологія не є кросплатформною, що унеможливлює її використання для створення застосунків для інших операційних систем. Крім того, у порівнянні з новітніми технологіями, такими як WPF (Windows Presentation Foundation) або сучасними вебфреймворками, WinForms пропонує обмежені можливості для створення складних графічних інтерфейсів із підтримкою анімації та векторної графіки.

Проте WinForms залишається актуальним інструментом для створення бізнес-застосунків та внутрішніх корпоративних систем, де важлива швидка розробка, стабільність та проста підтримка. Його надійність, зрілість та гнучкість у поєднанні з можливостями .NET Framework роблять WinForms

потужним інструментом для розробки настільних застосунків під Windows [2, 10].

Мова розмітки XML

Формат .xml (Extensible Markup Language) є універсальною мовою розмітки, розробленою для зберігання, передачі та організації даних у текстовому вигляді. XML є розширюваною мовою, що означає можливість створення власних тегів і структур даних відповідно до вимог конкретної сфери застосування [3, 8]. Ця гнучкість формату дозволяє використовувати його в різних галузях, таких як обмін даними між вебсервісами, зберігання конфігурацій програмного забезпечення, визначення структури документів та забезпечення сумісності між різними системами.

Основою формату XML є структура, що складається з ієрархічно вкладених елементів, кожен з яких представлений у вигляді тегів. Теги, що мають вигляд парних відкривних і закривних елементів, визначають логічну структуру та вміст даних у документі. Кожен елемент може містити текст, атрибути, а також інші вкладені елементи, що дозволяє створювати складні та багаторівневі структури даних. Завдяки такій структурованості XML забезпечує просте зчитування, аналіз та обробку даних як людиною, так і програмним забезпеченням.

Однією з важливих властивостей XML є його незалежність від платформи та мови програмування. Це означає, що дані, представлені у форматі .xml, можуть бути оброблені різноманітними програмними засобами на різних операційних системах, що забезпечує високу ступінь інтероперабельності та спільного використання інформації. З цієї причини XML широко використовується для обміну даними між вебсервісами та додатками, оскільки він дозволяє стандартизувати формат повідомлень і спростити процес передачі даних у розподілених системах.

Додатковою перевагою формату XML є можливість визначення схеми (XML Schema) або DTD (Document Type Definition), які визначають структуру

та типи даних, що містяться в документі. Це дозволяє забезпечити валідацію даних, гарантувати їхню цілісність та відповідність визначеним правилам. Завдяки цьому XML стає надійним засобом для зберігання та передачі даних у критично важливих системах, де важлива коректність і точність інформації.

Отже, формат .xml є потужним інструментом для представлення, обміну та зберігання структурованих даних у різних галузях, таких як веброзробка, бази даних, конфігураційні файли та системи обміну інформацією. Його гнучкість, розширюваність та сумісність з різноманітними технологіями роблять XML одним з найпоширеніших стандартів для роботи з даними у сучасних інформаційних системах [3, 8] .

Середовище розробки Visual Studio

Visual Studio - це інтегроване середовище розробки (IDE), створене компанією Microsoft, яке забезпечує потужні інструменти та сервіси для розробки різноманітного програмного забезпечення, включаючи веб-, мобільні та настільні додатки, а також серверні системи [4, 5]. Основною метою Visual Studio є спрощення та оптимізація процесу розробки програм за допомогою набору інструментів, які дозволяють ефективно створювати, налагоджувати, тестувати та розгортати програмні продукти (Рисунок 3.1).

Рисунок 3.1. Логотип Microsoft Visual Studio

Однією з ключових особливостей Visual Studio є підтримка багатьох мов програмування, серед яких найпоширенішими є C#, C++, Visual Basic, JavaScript та Python. Завдяки цьому розробники мають можливість працювати

над проєктами різних типів, зокрема настільними застосунками, вебсайтами, мобільними додатками, ігровими проєктами, системами управління базами даних та сервісами, що працюють у хмарному середовищі.

Visual Studio забезпечує розширений редактор коду з функціями підсвічування синтаксису, автозавершення коду, рефакторингу та інтелектуальних підказок на основі контексту, що суттєво підвищує продуктивність розробників. Крім того, вбудований відладчик дозволяє відстежувати та виправляти помилки у програмному коді, здійснюючи покрокове виконання програми, аналізуючи значення змінних та контролюючи потік виконання додатку. Завдяки можливостям налагодження розробники можуть ідентифікувати помилки та оптимізувати код ще на етапі його написання, забезпечуючи стабільність і продуктивність програмного продукту.

Середовище Visual Studio також включає потужні засоби для роботи з графічним інтерфейсом користувача. Зокрема, підтримка технологій Windows Forms, Windows Presentation Foundation (WPF) та Universal Windows Platform (UWP) надає розробникам інструменти для створення інтерактивних інтерфейсів користувача. Завдяки візуальним конструкторам розробники можуть швидко створювати та налаштовувати компоненти інтерфейсу, використовуючи технології перетягування (drag-and-drop), що значно спрощує процес проєктування застосунків.

Visual Studio також інтегрується з системами контролю версій, такими як Git, SVN та Team Foundation Server (TFS), що полегшує спільну розробку програмного забезпечення. Ця інтеграція дозволяє розробникам відслідковувати зміни в коді, керувати гілками, здійснювати злиття змін та вирішувати конфлікти, забезпечуючи ефективний робочий процес навіть у великих командах.

Іншим важливим аспектом Visual Studio є підтримка інструментів для тестування програмного забезпечення. Середовище включає засоби для написання та виконання модульних тестів, що допомагає розробникам перевіряти окремі компоненти системи на відповідність вимогам та виявляти

потенційні проблеми ще на ранніх етапах розробки. Автоматизація тестування спрощує процес перевірки якості програмного забезпечення та забезпечує його стабільність під час впровадження нових функцій або модифікації існуючого коду.

Visual Studio має потужну підтримку хмарних сервісів і дозволяє розробникам легко інтегрувати свої проєкти з платформою Microsoft Azure. Це відкриває можливості для створення хмарних додатків, розгортання вебсервісів та управління базами даних у хмарному середовищі. Завдяки інструментам, вбудованим у Visual Studio, розробники можуть швидко публікувати свої додатки в хмарі, автоматизувати процеси CI/CD (безперервної інтеграції та розгортання), а також моніторити продуктивність і використання ресурсів.

Таким чином, Visual Studio виступає як комплексне середовище розробки, що надає широкий спектр інструментів для написання, налагодження, тестування та розгортання програмного забезпечення різного рівня складності. Завдяки своїй гнучкості, підтримці різних мов програмування та інтеграції з сучасними сервісами, Visual Studio залишається одним з провідних інструментів для розробників програмного забезпечення, забезпечуючи їм ефективний та зручний робочий процес [4, 5].

3.2. Опис методології створення програмного забезпечення

Для розробки застосунків для керування відеоконтентом на основі C# та WinForms найдоцільніше застосувати гнучкі методології розробки програмного забезпечення, зокрема методологію Agile та її різновид - Scrum. Ці методології є підходящими для таких проєктів завдяки їхній адаптивності, ітеративному підходу та можливості швидко реагувати на зміни вимог і вдосконалення продукту протягом усього циклу розробки.

Використання Agile дозволяє розбити весь процес розробки на невеликі ітерації, що зазвичай тривають від одного до чотирьох тижнів. Під час кожної

ітерації (спринту) команда розробників може фокусуватися на реалізації конкретного функціоналу застосунку для керування відеоконтентом, наприклад, створення інтерфейсу для перегляду та сортування відео, впровадження функцій обробки відеофайлів чи інтеграції з іншими медіасистемами. Це забезпечує можливість поступового та поетапного розвитку проєкту, а також сприяє впровадженню і тестуванню окремих компонентів програмного забезпечення.

Scrum як частина Agile підходить для невеликих команд, які працюють над конкретним функціоналом застосунку. Використання цього підходу дозволяє команді швидко виявляти недоліки та покращувати окремі модулі програми, забезпечуючи тим самим її стабільність та надійність. Постійний зворотний зв'язок між розробниками, тестувальниками та кінцевими користувачами, що є невід'ємною частиною Scrum, сприяє уточненню вимог і своєчасному внесенню змін до проєкту. Це особливо корисно при роботі з C# та WinForms, де різноманітні функції, такі як обробка відеофайлів чи створення інтуїтивно зрозумілого інтерфейсу, можуть потребувати частих удосконалень на основі отриманих відгуків.

Для розробки застосунку з використанням WinForms методологія Agile є оптимальною також тому, що ця технологія дозволяє швидко створювати прототипи графічного інтерфейсу користувача та взаємодіяти з ним. Візуальний дизайнер WinForms у середовищі розробки Visual Studio дає можливість розробникам швидко створювати та змінювати елементи інтерфейсу, реагуючи на нові вимоги або ідеї, що виникають протягом розробки. Такий підхід сприяє реалізації інкрементальної розробки, коли застосунок поступово вдосконалюється на кожному етапі, додаючи нові функції чи покращуючи існуючі.

Крім того, Agile підходить для проєктів із розробки застосунків для керування відеоконтентом завдяки своїй гнучкості в плануванні ресурсів та зміні пріоритетів завдань. Наприклад, якщо під час розробки з'ясується, що певна функція обробки відеофайлів або інтеграція з бібліотекою потребує

додаткових досліджень чи змін, команда може швидко скоригувати план робіт і розподілити ресурси для вирішення нових викликів без значного впливу на загальний прогрес проєкту.

Отже, використання гнучких методологій розробки програмного забезпечення, зокрема Agile та Scrum, є найбільш доцільним підходом для створення застосунку для керування відеоконтентом на основі C# та WinForms. Цей підхід забезпечує адаптивність процесу розробки, сприяє швидкому впровадженню змін, підвищує якість продукту та забезпечує постійний зворотний зв’язок із користувачами, що є критично важливим у розробці складних і функціонально багатих застосунків.

Для розробки мобільного застосунку для керування відеононтентом була використана Agile модель. Agile - це сучасна методологія управління проєктами, що використовується в розробці програмного забезпечення та інших галузях, де важлива гнучкість і швидкість реагування на зміни [9].

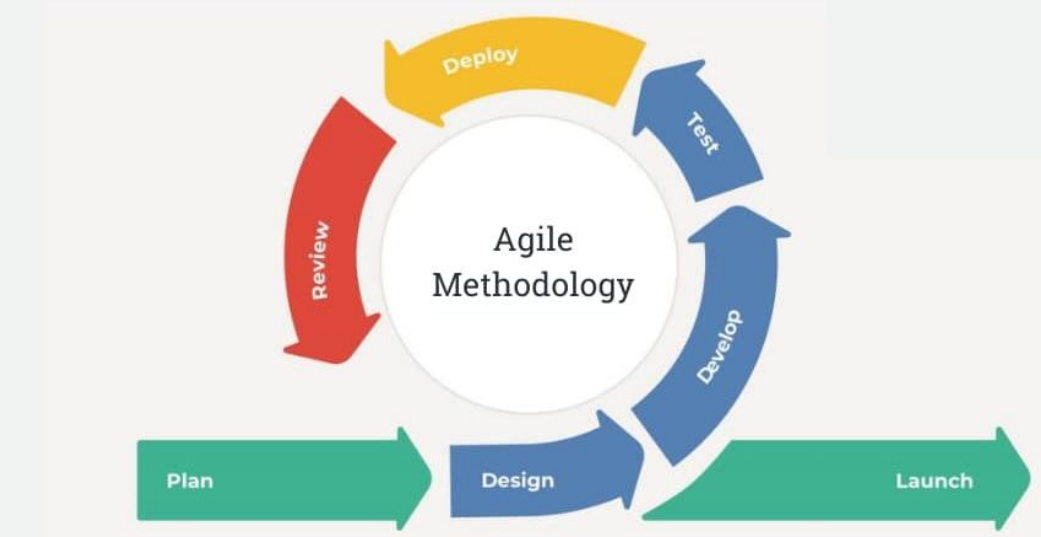

Рисунок. 3.2. Модель розробки застосунку

Основні фази Agile моделі включають:

1. Концепція (Concept) - розробка ідей та визначення основних вимог до продукту.

2. Запуск (Inception) - планування та приготування необхідних ресурсів.

3. Ітерація (Iteration) - циклічна розробка, де продукт постійно

вдосконалюється через регулярні оновлення.

4. Випуск (Release) - завершення розробки та випуск продукту для користувачів.

5. Підтримка (Maintenance) - оновлення та виправлення помилок продукту після його випуску.

6. Відставка (Retirement) - заміна продукту або його функцій новими.

Agile зосереджений на адаптивності та відповідності до змінюваних потреб користувачів та ринку, що дозволяє розробникам швидко реагувати на зміни та постійно покращувати продукт, тому ми і обрали її для розробки програмного забезпечення нашого застосунку [9].

РОЗДІЛ 4. ПРАКТИЧНА ЧАСТИНА

4.1. Опис та побудова діаграми класів розробленого застосунку

У контексті нашого застосунку, діаграма класів (Рисунок 4.1), демонструє ключові можливості та функціональність розробленого програмного забезпечення. Така діаграма допомагає краще зрозуміти особливості програмного забезпечення розробленого застосунку.

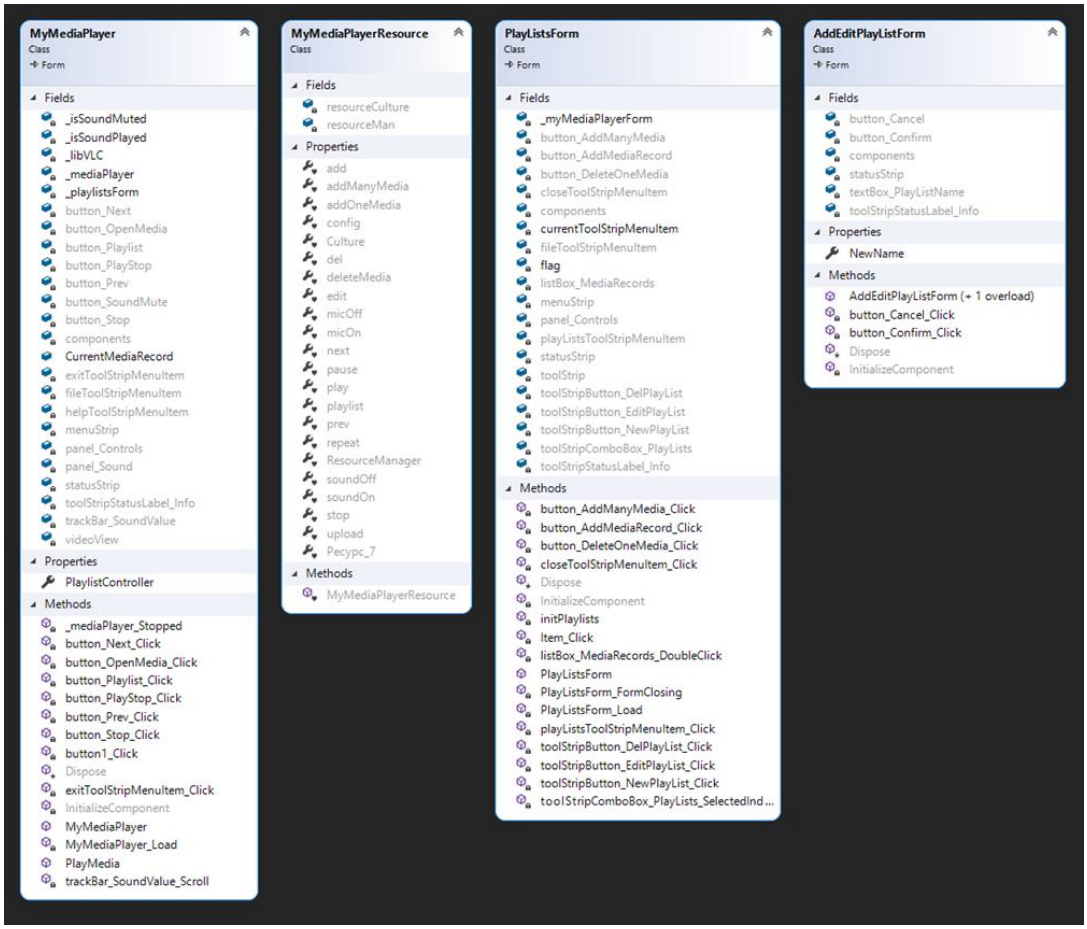

Рисунок 4.1. Діаграма класів розробленого застосунку

Ця діаграма допомагає зрозуміти основні взаємодії у розробленому програмному забезпеченні, а також основні задачі в рамках процесу розробки та використання.

Використовуючи інтерфейс користувача користувач може вибрати і

відтворити медіа контент. Для зручності користувача розроблена система управління користувальницькими плейлистами.

Система зберігання та маніпулювання квестами представлена на наступній діаграмі класів.

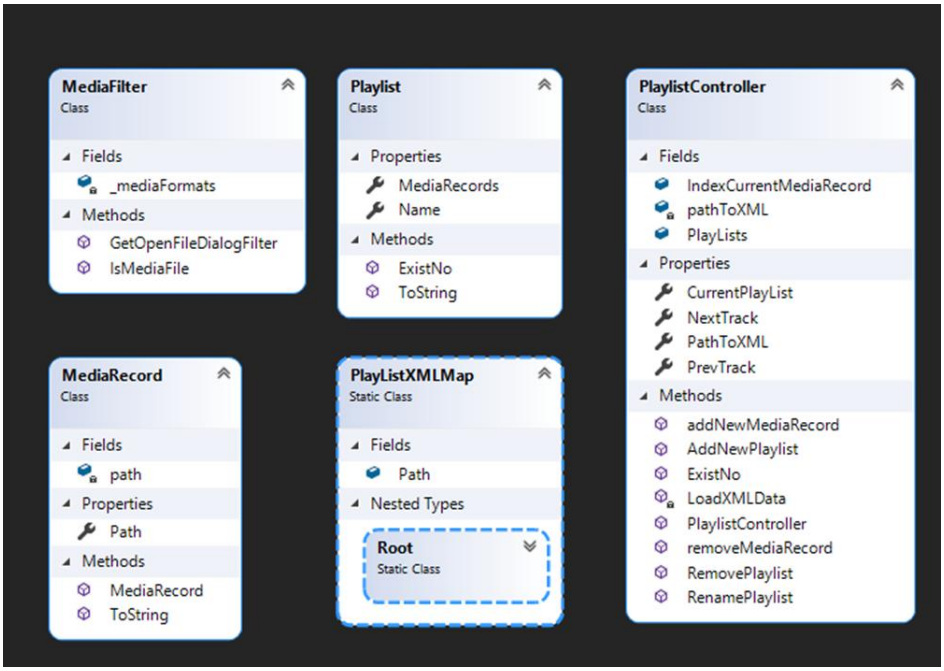

Рисунок 4.2. Діаграма класів для управління квестами

Інформація зберігається у форматі .xml.

Користувачеві доступний функціонал зі створення, редагування та управління плейлистами. У кожний плейлист користувач може додати сутність типу медіа запис.

4.2. Інструкція з використання застосунка

На скріншоті ми бачимо інтерфейс користувача застосунку для керування відеоконтентном (Рисунок 4.3.). У плеєрі містяться всі основні кнопки для керування відеоконтентом «Додати відео», «Відтворити запис», «Зупинити запис», «Відтворити попереднє відео», «Створити плейлист», «Рівень звуку» тощо.

Рис. 4.3. Основний екран застосунку

Для управління плейлистами та медіазаписами розроблена форма (Рисунок 4.4).

Рисунок 4.4. Управління плейлистами

Застосунок працює лише із файлами завантаженими на комп’ютер.

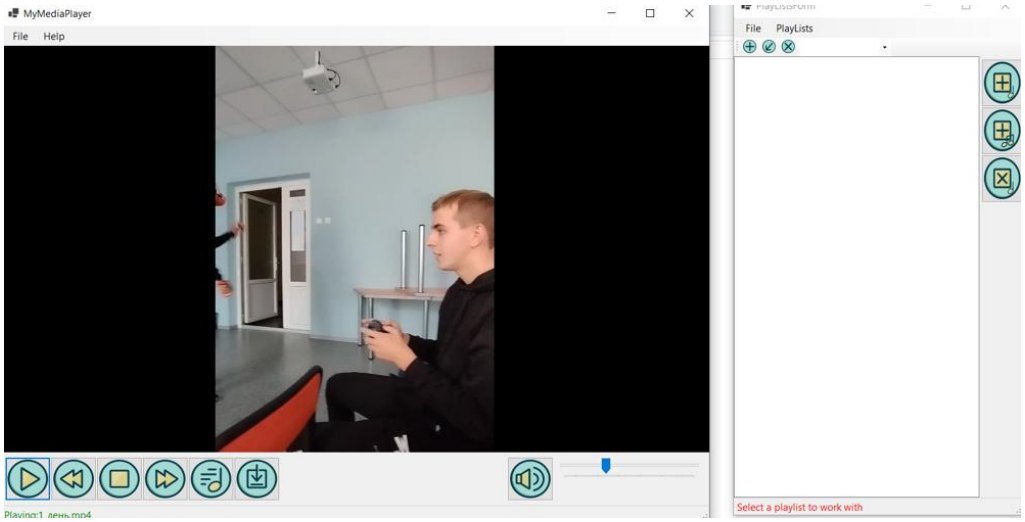

Рисунок 4.5. Процес відтворення відеоконтенту

Для більш зручного керування плейлистами користувачеві надаються підказки внизу екрану.

ВИСНОВКИ

У процесі роботи було проаналізовано основні вимоги до програмного забезпечення такого типу, обрано інструменти та технології, оптимальні для створення ефективного та зручного у використанні застосунку.

Розроблений застосунок успішно реалізує функції організації, сортування, відтворення та обробки відеоконтенту, що дозволяє користувачам швидко і зручно працювати з відеоматеріалами. Завдяки використанню мови програмування C# та бібліотеки WinForms, вдалося створити інтуїтивно зрозумілий графічний інтерфейс, який забезпечує легкий доступ до основних функцій програми.

У роботі застосовано гнучку методологію розробки Agile, яка дозволила ефективно планувати і виконувати процес створення застосунку, здійснювати поетапне впровадження функціональності та вносити корективи на основі зворотного зв’язку. Завдяки цьому підходу вдалося досягти високої якості програмного забезпечення, яке відповідає вимогам користувачів і сучасним стандартам розробки.

У результаті проведеного дослідження та розробки було підтверджено, що мова C# у поєднанні з WinForms є ефективним інструментом для створення застосунків із керування відеоконтентом. Вибраний технологічний стек забезпечив необхідний рівень функціональності, стабільності та продуктивності програми, що дозволяє використовувати її в різних навчальних і професійних середовищах.

Подальший розвиток проєкту може передбачати розширення функціональних можливостей застосунку, зокрема впровадження більш складних інструментів для редагування відео, інтеграцію з хмарними сервісами для зберігання даних та реалізацію додаткових модулів для обробки аудіо- та відеоданих. Таким чином, створений застосунок має значний потенціал для подальшого вдосконалення та адаптації під потреби користувачів.

Отже, у результати виконання дипломного проєкту нами було повністю

виконано поставлені завдання дипломної роботи, а саме:

1) Визначено ключові вимоги до розробки застосунку для керування медіаконтентом;

2) Проведено аналіз переваг та обмежень аналогічних застосунків для керування відео та аудіо;

3) Проаналізовано та описано підходи та інструменти, використані при розробці ПЗ застосунку;

4) Розроблено мобільний інтерфейс застосунку;

5) Представлено діаграму класів розробленого застосунку та діаграму класів управління квестами у застосунку;

6) Розроблено та протестовано застосунок для керування відеоконтентом.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Черненко О. О. Методичні рекомендації щодо виконання кваліфікаційної роботи студентів спеціальності 122 Комп'ютерні науки освітня програма «Комп'ютерні науки» ступеня магістра / С. В. Гаркуша, О. В. Ольховська, О. О. Черненко. – Полтава: ПУЕТ, 2023. – 68 с. Режим доступу: URL: http://elib.puet.edu.ua/action.php?kt_path_info=ktcore.SecViewPlugin.actions.document&fDocumentId=824868 - Назва з екрану.

2. Edin Kapic. Windows Forms Binding Improvements in .NET 7 for MVVM Support. Режим доступу: URL: https://www.infoq.com/news/2023/02/winforms-binding-mvvm-net-7/ - Назва з екрану.

3. Eileen Roche. Explaining XML. Режим доступу: URL: https://hbr.org/2000/07/explaining-xml - Назва з екрану.

4. Introduction to Visual Studio. Режим доступу: URL: https://www.geeksforgeeks.org/introduction-to-visual-studio/ - Назва з екрану.

5. Madhu Murali. What is Visual Studio? All You Need to Know. Режим доступу: URL: https://blog.hubspot.com/website/what-is-visual-studio - Назва з екрану.

6. Meet Android Studio. Режим доступу: URL: https://developer.android.com/studio/intro - Назва з екрану.

7. 1. The Benefits of Using Version Control Systems Like Git in Software Development. Режим доступу до ресурсу: URL: https://hojaleaks.com/the-benefits-of-using-version-control-systems-like-git-in-software-development – Назва з екрану.

8. Vijay Kanade. What Is XML (Extensible Markup Language)? Meaning, Elements, and Benefits. Режим доступу: URL: https://www.spiceworks.com/tech/tech-general/articles/what-is-xml/ - Назва з екрану

9. What Is Agile? And When to Use It. Режим доступу: URL: https://www.coursera.org/articles/what-is-agile-a-beginners-guide - Назва з екрану.

10. WPF vs WinForms – Making the Right Decision in 2024. Режим доступу: URL: https://blog.ndepend.com/wpf-vs-winforms-choosing-the-proper-framework-for-your-project/ - Назва з екрану.

11. Євгенія Стенцель. Що таке C#? Кому підходить програмування на Сі Шарп? Режим доступу: URL: https://beetroot.academy/blog/shcho-take-c-chi-pidhodit-meni-cya-mova-programuvannya-chomu-vona-kruta - Назва з екрану.

12. Як вивчити мову програмування C# та стати .NET розробником. Режим доступу: URL: https://edu.cbsystematics.com/ua/blog/learn-csharp-become-dnet - Назва з екрану.

ДОДАТОК А
Код програми

```
<Project Sdk="Microsoft.NET.Sdk.WindowsDesktop">

  <PropertyGroup>
    <OutputType>WinExe</OutputType>
    <TargetFramework>netcoreapp3.1</TargetFramework>
    <UseWindowsForms>true</UseWindowsForms>
  </PropertyGroup>

  <ItemGroup>
    <PackageReference Include="LibVLCSharp.Forms" Version="3.7.0" />
    <PackageReference Include="LibVLCSharp.WinForms" Version="3.7.0" />
    <PackageReference Include="VideoLAN.LibVLC.Windows" Version="3.0.18" />
  </ItemGroup>

  <ItemGroup>
    <Compile Update="MyMediaPlayerResource.Designer.cs">
      <DesignTime>True</DesignTime>
      <AutoGen>True</AutoGen>
      <DependentUpon>MyMediaPlayerResource.resx</DependentUpon>
    </Compile>
  </ItemGroup>

  <ItemGroup>
    <EmbeddedResource Update="MyMediaPlayerResource.resx">
      <Generator>ResXFileCodeGenerator</Generator>
      <LastGenOutput>MyMediaPlayerResource.Designer.cs</LastGenOutput>
    </EmbeddedResource>
  </ItemGroup>

</Project>

//------------------------------------------------------------------------------
// <auto-generated>
//     This code was generated by a tool.
//     Runtime Version:4.0.30319.42000
//
//     Changes to this file may cause incorrect behavior and will be lost if
//     the code is regenerated.
// </auto-generated>
//------------------------------------------------------------------------------

namespace MyMediaPlayer {
    using System;


    /// <summary>
    ///   A strongly-typed resource class, for looking up localized strings, etc.
    /// </summary>
```

```
    // This class was auto-generated by the StronglyTypedResourceBuilder
    // class via a tool like ResGen or Visual Studio.
    // To add or remove a member, edit your .ResX file then rerun ResGen
    // with the /str option, or rebuild your VS project.

[global::System.CodeDom.Compiler.GeneratedCodeAttribute("System.Resources.Tools.StronglyT
ypedResourceBuilder", "16.0.0.0")]
    [global::System.Diagnostics.DebuggerNonUserCodeAttribute()]
    [global::System.Runtime.CompilerServices.CompilerGeneratedAttribute()]
    internal class MyMediaPlayerResource {

        private static global::System.Resources.ResourceManager resourceMan;

        private static global::System.Globalization.CultureInfo resourceCulture;


[global::System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
"CA1811:AvoidUncalledPrivateCode")]
        internal MyMediaPlayerResource() {
        }

        /// <summary>
        ///   Returns the cached ResourceManager instance used by this class.
        /// </summary>

[global::System.ComponentModel.EditorBrowsableAttribute(global::System.ComponentModel.Edi
torBrowsableState.Advanced)]
        internal static global::System.Resources.ResourceManager ResourceManager {
            get {
                if (object.ReferenceEquals(resourceMan, null)) {
                    global::System.Resources.ResourceManager temp = new
global::System.Resources.ResourceManager("MyMediaPlayer.MyMediaPlayerResource",
typeof(MyMediaPlayerResource).Assembly);
                    resourceMan = temp;
                }
                return resourceMan;
            }
        }

        /// <summary>
        ///   Overrides the current thread's CurrentUICulture property for all
        ///   resource lookups using this strongly typed resource class.
        /// </summary>

[global::System.ComponentModel.EditorBrowsableAttribute(global::System.ComponentModel.Edi
torBrowsableState.Advanced)]
        internal static global::System.Globalization.CultureInfo Culture {
            get {
                return resourceCulture;
            }
            set {
                resourceCulture = value;
            }
```

```
}

/// <summary>
///   Looks up a localized resource of type System.Drawing.Bitmap.
/// </summary>
internal static System.Drawing.Bitmap add {
   get {
      object obj = ResourceManager.GetObject("add", resourceCulture);
      return ((System.Drawing.Bitmap)(obj));
   }
}

/// <summary>
///   Looks up a localized resource of type System.Drawing.Bitmap.
/// </summary>
internal static System.Drawing.Bitmap addManyMedia {
   get {
      object obj = ResourceManager.GetObject("addManyMedia", resourceCulture);
      return ((System.Drawing.Bitmap)(obj));
   }
}

/// <summary>
///   Looks up a localized resource of type System.Drawing.Bitmap.
/// </summary>
internal static System.Drawing.Bitmap addOneMedia {
   get {
      object obj = ResourceManager.GetObject("addOneMedia", resourceCulture);
      return ((System.Drawing.Bitmap)(obj));
   }
}

/// <summary>
///   Looks up a localized resource of type System.Drawing.Bitmap.
/// </summary>
internal static System.Drawing.Bitmap config {
   get {
      object obj = ResourceManager.GetObject("config", resourceCulture);
      return ((System.Drawing.Bitmap)(obj));
   }
}

/// <summary>
///   Looks up a localized resource of type System.Drawing.Bitmap.
/// </summary>
internal static System.Drawing.Bitmap del {
   get {
      object obj = ResourceManager.GetObject("del", resourceCulture);
      return ((System.Drawing.Bitmap)(obj));
   }
}

/// <summary>
```

```
///   Looks up a localized resource of type System.Drawing.Bitmap.
/// </summary>
internal static System.Drawing.Bitmap deleteMedia {
   get {
      object obj = ResourceManager.GetObject("deleteMedia", resourceCulture);
      return ((System.Drawing.Bitmap)(obj));
   }
}

/// <summary>
///   Looks up a localized resource of type System.Drawing.Bitmap.
/// </summary>
internal static System.Drawing.Bitmap edit {
   get {
      object obj = ResourceManager.GetObject("edit", resourceCulture);
      return ((System.Drawing.Bitmap)(obj));
   }
}

/// <summary>
///   Looks up a localized resource of type System.Drawing.Bitmap.
/// </summary>
internal static System.Drawing.Bitmap micOff {
   get {
      object obj = ResourceManager.GetObject("micOff", resourceCulture);
      return ((System.Drawing.Bitmap)(obj));
   }
}

/// <summary>
///   Looks up a localized resource of type System.Drawing.Bitmap.
/// </summary>
internal static System.Drawing.Bitmap micOn {
   get {
      object obj = ResourceManager.GetObject("micOn", resourceCulture);
      return ((System.Drawing.Bitmap)(obj));
   }
}

/// <summary>
///   Looks up a localized resource of type System.Drawing.Bitmap.
/// </summary>
internal static System.Drawing.Bitmap next {
   get {
      object obj = ResourceManager.GetObject("next", resourceCulture);
      return ((System.Drawing.Bitmap)(obj));
   }
}

/// <summary>
///   Looks up a localized resource of type System.Drawing.Bitmap.
/// </summary>
internal static System.Drawing.Bitmap pause {
```

```
    get {
        object obj = ResourceManager.GetObject("pause", resourceCulture);
        return ((System.Drawing.Bitmap)(obj));
    }
}

/// <summary>
///   Looks up a localized resource of type System.Drawing.Bitmap.
/// </summary>
internal static System.Drawing.Bitmap play {
    get {
        object obj = ResourceManager.GetObject("play", resourceCulture);
        return ((System.Drawing.Bitmap)(obj));
    }
}

/// <summary>
///   Looks up a localized resource of type System.Drawing.Bitmap.
/// </summary>
internal static System.Drawing.Bitmap playlist {
    get {
        object obj = ResourceManager.GetObject("playlist", resourceCulture);
        return ((System.Drawing.Bitmap)(obj));
    }
}

/// <summary>
///   Looks up a localized resource of type System.Drawing.Bitmap.
/// </summary>
internal static System.Drawing.Bitmap prev {
    get {
        object obj = ResourceManager.GetObject("prev", resourceCulture);
        return ((System.Drawing.Bitmap)(obj));
    }
}

/// <summary>
///   Looks up a localized resource of type System.Drawing.Bitmap.
/// </summary>
internal static System.Drawing.Bitmap repeat {
    get {
        object obj = ResourceManager.GetObject("repeat", resourceCulture);
        return ((System.Drawing.Bitmap)(obj));
    }
}

/// <summary>
///   Looks up a localized resource of type System.Drawing.Bitmap.
/// </summary>
internal static System.Drawing.Bitmap soundOff {
    get {
        object obj = ResourceManager.GetObject("soundOff", resourceCulture);
        return ((System.Drawing.Bitmap)(obj));
```

```
        }
    }

    /// <summary>
    ///   Looks up a localized resource of type System.Drawing.Bitmap.
    /// </summary>
    internal static System.Drawing.Bitmap soundOn {
        get {
            object obj = ResourceManager.GetObject("soundOn", resourceCulture);
            return ((System.Drawing.Bitmap)(obj));
        }
    }

    /// <summary>
    ///   Looks up a localized resource of type System.Drawing.Bitmap.
    /// </summary>
    internal static System.Drawing.Bitmap stop {
        get {
            object obj = ResourceManager.GetObject("stop", resourceCulture);
            return ((System.Drawing.Bitmap)(obj));
        }
    }

    /// <summary>
    ///   Looks up a localized resource of type System.Drawing.Bitmap.
    /// </summary>
    internal static System.Drawing.Bitmap upload {
        get {
            object obj = ResourceManager.GetObject("upload", resourceCulture);
            return ((System.Drawing.Bitmap)(obj));
        }
    }

    /// <summary>
    ///   Looks up a localized resource of type System.Drawing.Bitmap.
    /// </summary>
    internal static System.Drawing.Bitmap Pecypc_7 {
        get {
            object obj = ResourceManager.GetObject("Pecypc_7", resourceCulture);
            return ((System.Drawing.Bitmap)(obj));
        }
    }
  }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace MyMediaPlayer
{
```

```
    static class Program
    {
        /// <summary>
        ///  The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.SetHighDpiMode(HighDpiMode.SystemAware);
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new MyMediaPlayer());
        }
    }
}

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;

namespace MyMediaPlayer
{
    public partial class AddEditPlayListForm : Form
    {
        public AddEditPlayListForm()
        {
            InitializeComponent();
            this.Text = "Create new PlayList";
        }
        public AddEditPlayListForm(string oldPlayListName)
        {
            InitializeComponent();
            textBox_PlayListName.Text = oldPlayListName;
            this.Text = "Rename PlayList";
        }
        public string NewName { get; internal set; }
        private void button_Confirm_Click(object sender, EventArgs e)
        {
            if (textBox_PlayListName.Text.Length >= 3)
            {
                this.NewName = textBox_PlayListName.Text;
                this.DialogResult = DialogResult.OK;
            }
            else
            {
                toolStripStatusLabel_Info.ForeColor = Color.Red;
                toolStripStatusLabel_Info.Text = "Set a name for the new playlist";
            }
        }
```

```
        private void button_Cancel_Click(object sender, EventArgs e)
        {
            this.Close();
        }
    }
}

namespace MyMediaPlayer
{
    partial class AddEditPlayListForm
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Windows Form Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            this.textBox_PlayListName = new System.Windows.Forms.TextBox();
            this.button_Confirm = new System.Windows.Forms.Button();
            this.button_Cancel = new System.Windows.Forms.Button();
            this.statusStrip = new System.Windows.Forms.StatusStrip();
            this.toolStripStatusLabel_Info = new System.Windows.Forms.ToolStripStatusLabel();
            this.statusStrip.SuspendLayout();
            this.SuspendLayout();
            //
            // textBox_PlayListName
            //
            this.textBox_PlayListName.Font = new System.Drawing.Font("Segoe UI", 14F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point);
            this.textBox_PlayListName.Location = new System.Drawing.Point(13, 13);
            this.textBox_PlayListName.Name = "textBox_PlayListName";
```

```
            this.textBox_PlayListName.Size = new System.Drawing.Size(259, 32);
            this.textBox_PlayListName.TabIndex = 0;
            //
            // button_Confirm
            //
            this.button_Confirm.BackgroundImage                                    =
global::MyMediaPlayer.MyMediaPlayerResource.add;
            this.button_Confirm.BackgroundImageLayout                              =
System.Windows.Forms.ImageLayout.Zoom;
            this.button_Confirm.Font = new System.Drawing.Font("Segoe UI", 14F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point);
            this.button_Confirm.Location = new System.Drawing.Point(232, 52);
            this.button_Confirm.Name = "button_Confirm";
            this.button_Confirm.Size = new System.Drawing.Size(40, 40);
            this.button_Confirm.TabIndex = 1;
            this.button_Confirm.UseVisualStyleBackColor = true;
            this.button_Confirm.Click += new System.EventHandler(this.button_Confirm_Click);
            //
            // button_Cancel
            //
            this.button_Cancel.BackgroundImage                                     =
global::MyMediaPlayer.MyMediaPlayerResource.del;
            this.button_Cancel.BackgroundImageLayout                               =
System.Windows.Forms.ImageLayout.Zoom;
            this.button_Cancel.Font = new System.Drawing.Font("Segoe UI", 14F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point);
            this.button_Cancel.Location = new System.Drawing.Point(186, 52);
            this.button_Cancel.Name = "button_Cancel";
            this.button_Cancel.Size = new System.Drawing.Size(40, 40);
            this.button_Cancel.TabIndex = 1;
            this.button_Cancel.UseVisualStyleBackColor = true;
            this.button_Cancel.Click += new System.EventHandler(this.button_Cancel_Click);
            //
            // statusStrip
            //
            this.statusStrip.Items.AddRange(new System.Windows.Forms.ToolStripItem[] {
            this.toolStripStatusLabel_Info});
            this.statusStrip.Location = new System.Drawing.Point(0, 99);
            this.statusStrip.Name = "statusStrip";
            this.statusStrip.Size = new System.Drawing.Size(284, 22);
            this.statusStrip.TabIndex = 2;
            this.statusStrip.Text = "statusStrip1";
            //
            // toolStripStatusLabel_Info
            //
            this.toolStripStatusLabel_Info.Name = "toolStripStatusLabel_Info";
            this.toolStripStatusLabel_Info.Size = new System.Drawing.Size(48, 17);
            this.toolStripStatusLabel_Info.Text = "Ready...";
            //
            // AddEditPlayListForm
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(7F, 15F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
```

```
            this.ClientSize = new System.Drawing.Size(284, 121);
            this.Controls.Add(this.statusStrip);
            this.Controls.Add(this.button_Cancel);
            this.Controls.Add(this.button_Confirm);
            this.Controls.Add(this.textBox_PlayListName);
            this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.Fixed3D;
            this.MaximizeBox = false;
            this.MinimizeBox = false;
            this.Name = "AddEditPlayListForm";
            this.StartPosition = System.Windows.Forms.FormStartPosition.CenterParent;
            this.statusStrip.ResumeLayout(false);
            this.statusStrip.PerformLayout();
            this.ResumeLayout(false);
            this.PerformLayout();

        }

        #endregion

        private System.Windows.Forms.TextBox textBox_PlayListName;
        private System.Windows.Forms.Button button_Confirm;
        private System.Windows.Forms.Button button_Cancel;
        private System.Windows.Forms.StatusStrip statusStrip;
        private System.Windows.Forms.ToolStripStatusLabel toolStripStatusLabel_Info;
    }
}

using LibVLCSharp.Shared;
using MyMediaPlayer.PlaylistCommons;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace MyMediaPlayer
{
    public partial class MyMediaPlayer : Form
    {
        private LibVLC _libVLC;
        private MediaPlayer _mediaPlayer;
        private PlayListsForm _playlistsForm;
        public MediaRecord CurrentMediaRecord;
        private bool _isSoundMuted = false;
        private bool _isSoundPlayed = false;
        public PlaylistController PlaylistController { get; private set; }

        public MyMediaPlayer()
```

```
        {
            InitializeComponent();
            _libVLC = new LibVLC();
            _mediaPlayer = new MediaPlayer(_libVLC);
            videoView.MediaPlayer = _mediaPlayer;
            _mediaPlayer.Volume = trackBar_SoundValue.Value;
            _mediaPlayer.Stopped += _mediaPlayer_Stopped;
        }

        private void _mediaPlayer_Stopped(object sender, EventArgs e)
        {
            if (_mediaPlayer.Media == null) return;
            if (_mediaPlayer.Media.State == VLCState.Ended)
            {
                PlayMedia(PlaylistController.NextTrack);
            }
        }

        private void button_OpenMedia_Click(object sender, EventArgs e)
        {
            OpenFileDialog openFileDialog = new OpenFileDialog();
            openFileDialog.Filter = MediaFilter.GetOpenFileDialogFilter();
            openFileDialog.InitialDirectory =
Environment.GetFolderPath(Environment.SpecialFolder.MyVideos);
            if (openFileDialog.ShowDialog() == DialogResult.OK)
            {
                if (MediaFilter.IsMediaFile(openFileDialog.FileName)) {

                    CurrentMediaRecord = new MediaRecord(openFileDialog.FileName);
                    var media = new Media(_libVLC, new Uri(CurrentMediaRecord.Path));
                    _mediaPlayer.Play(media);
                    _isSoundPlayed = true;
                    toolStripStatusLabel_Info.ForeColor = Color.Green;
                    toolStripStatusLabel_Info.Text = "Playing:" +
Path.GetFileName(CurrentMediaRecord.Path);
                }
                else
                {
                    toolStripStatusLabel_Info.ForeColor = Color.Red;
                    toolStripStatusLabel_Info.Text = "The file you selected is not in a valid format";
                    return;
                }
            }
        }

        private void trackBar_SoundValue_Scroll(object sender, EventArgs e)
        {
            _mediaPlayer.Volume = trackBar_SoundValue.Value;
            button_SoundMute.BackgroundImage = MyMediaPlayerResource.soundOn;
            _isSoundMuted = false;
        }

        private void button1_Click(object sender, EventArgs e)
```

```
        {
            if (_isSoundMuted == false)          //звук не выключен
            {
                _isSoundMuted = true;
                _mediaPlayer.Volume = 0;
                trackBar_SoundValue.Value = 0;
                button_SoundMute.BackgroundImage = MyMediaPlayerResource.soundOff;
            } else
            {
                _isSoundMuted = false;
                _mediaPlayer.Volume = 50;
                trackBar_SoundValue.Value = 50;
                button_SoundMute.BackgroundImage = MyMediaPlayerResource.soundOn;
            }
        }

        private void MyMediaPlayer_Load(object sender, EventArgs e)
        {
            PlayListXMLMap.Path = "PlayList.xml";
            PlaylistController = new PlaylistController(PlayListXMLMap.Path);
        }

        private void button_Playlist_Click(object sender, EventArgs e)
        {
            if (_playlistsForm == null)
            {
                _playlistsForm = new PlayListsForm(this);
            }
            _playlistsForm.Location = new Point(this.Location.X + this.Width - 12, this.Location.Y);
            _playlistsForm.Show();
        }

        public void PlayMedia(MediaRecord record)
        {
            if (record == null) return;

            CurrentMediaRecord = record;
            _mediaPlayer.Play(new Media(_libVLC, new Uri(CurrentMediaRecord.Path)));

            _isSoundPlayed = true;
            button_PlayStop.BackgroundImage = MyMediaPlayerResource.play;

            toolStripStatusLabel_Info.ForeColor = Color.Green;
            toolStripStatusLabel_Info.Text = "Playing:" +
Path.GetFileName(CurrentMediaRecord.Path);
        }

        private void button_Stop_Click(object sender, EventArgs e)
        {
            if (_mediaPlayer != null)
            {
                _mediaPlayer.Stop();
                _isSoundPlayed = false;
```

```
        }
    }

    private void button_PlayStop_Click(object sender, EventArgs e)
    {
        if(_isSoundPlayed)
        {
            _mediaPlayer.Pause();
            _isSoundPlayed = false;
            button_PlayStop.BackgroundImage = MyMediaPlayerResource.pause;
        } else
        {
            _mediaPlayer.Play();
            _isSoundPlayed = true;
            button_PlayStop.BackgroundImage = MyMediaPlayerResource.play;
        }

    }

    private void button_Next_Click(object sender, EventArgs e)
    {
        PlayMedia(PlaylistController.NextTrack);
    }

    private void button_Prev_Click(object sender, EventArgs e)
    {
        PlayMedia(PlaylistController.PrevTrack);
    }

    private void exitToolStripMenuItem_Click(object sender, EventArgs e)
    {
        this.Close();
    }
  }
}
```