МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЧОРНОМОРСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ПЕТРА МОГИЛИ

Григор'єв Даниїл Олександрович

УДК 004.633.2

Інформаційна система для роботи з тегами ID3v2 для mp3 файлів. Інтерфейс користувача з використанням технології WPF та паттерну MVVM, робота з файловими ієрархіями.

Напрям підготовки 6.050101 – «Комп'ютерні науки»
ДР.ПЗ - 403.1510305

Автореферат
дипломної роботи на здобуття освітньої кваліфікації
«Бакалавр комп'ютерних наук»

Миколаїв – 2019

Дипломна робота є рукопис.

Робота виконана в Чорноморському національному університеті імені Петра Могили Міністерства освіти і науки України на кафедрі інтелектуальних інформаційних систем

Науковий керівник: кандидат технічних наук, доцент
Давиденко Євген Олександрович.

Рецензент: д.т.н., проф., завідувач кафедри
інтелектуальних інформаційних систем
Кондратенко Юрій Пантелійович.

Захист відбудеться «26» червня 2019 р. о $9^{\underline{30}}$ год. на засіданні екзаменаційної комісії (ауд. 2-403) у Чорноморському національному університеті імені Петра Могили за адресою: 54003, м. Миколаїв, вул. 68-ми Десантників, 10.

З дипломною роботою можна ознайомитися в бібліотеці Чорноморського національного університету імені Петра Могили за адресою: 54003, м. Миколаїв, вул. 68-ми Десантників, 10.

Автореферат представлений «20» червня 2019 р.

Секретар
екзаменаційної комісії,
викладач Кандиба І.О.

ЗАГАЛЬНА ХАРАКТЕРИСТИКА РОБОТИ

Актуальність теми.

У наш час важко знайти людину яка б не зберігала музичні файли на телефоні або персональному комп’ютері. Більшість таких користувачів зустрічаються з проблемою сортування та відображення музичної бібліотеки.

Цю проблему можна вирішити спираючись на метадані які містять в собі музичні файли. В них міститься багато цікавої та корисної інформації яку можна використати для впорядкування бібліотеки та для отримання цієї інформації. Іноді важливо мати можливість для редагування цих даних, синхронізування з хмарним сховищем або базою даних.

Як не дивно для таких цілей програм з повноцінним функціоналом на персональному комп’ютері виявилось небагато, що наштовшує на думку для створення нової програми, з більшою кількістю функціоналу та можливостями, білошую зручністю.

Архітектура програми має відповідати всім практикам розробки програмного забезпечення з використанням патерном MVVM. Який дозволить відділити логіку застосунку від дизайну. В купі з WPF розділення дизайну та логіки дозволяє виграти на декларативній мові розмітки, яка дозволяє використовувати можливості персонального комп’ютера з максимальною віддачею.

Метою дипломної роботи є

Розробка застосунку для редагування метаданих аудіофайлів та робота з файловими ієрархіями.

Практичне значення отриманих результатів.

Отримання навичок розробки застосунків робочого столу використовуючи паттерн MVVM.

Структура дипломної роботи. Пояснювальна записка до дипломної роботи складається із вступу, 3 розділів, висновків, додатків. Загальний обсяг роботи складає 65 сторінки, 5 рисунків, 0 таблиць та 6 посилань на літературні джерела.

ОСНОВНИЙ ЗМІСТ РОБОТИ

ПРОЕКТУВАННЯ ЗАСТОСУНКУ

Шаблон проектування MVVM

Патерн MVVM (Model-View-ViewModel) дозволяє відокремити логіку додатки від візуальної частини (подання). Даний патерн є архітектурним, тобто він задає загальну архітектуру програми.

Даний патерн був представлений Джоном Госсманом (John Gossman) в 2005 році як модифікація шаблону Presentation Model і був спочатку націлений на розробку застосунків в WPF. І хоча зараз цей патерн вийшов за межі WPF і застосовується в самих різних технологіях, в тому числі при розробці під Android, iOS, проте WPF є досить показовою технологією, яка розкриває можливості даного патерну.

MVVM складається з трьох компонентів: моделі (Model), моделі подання (ViewModel) і уявлення (View).

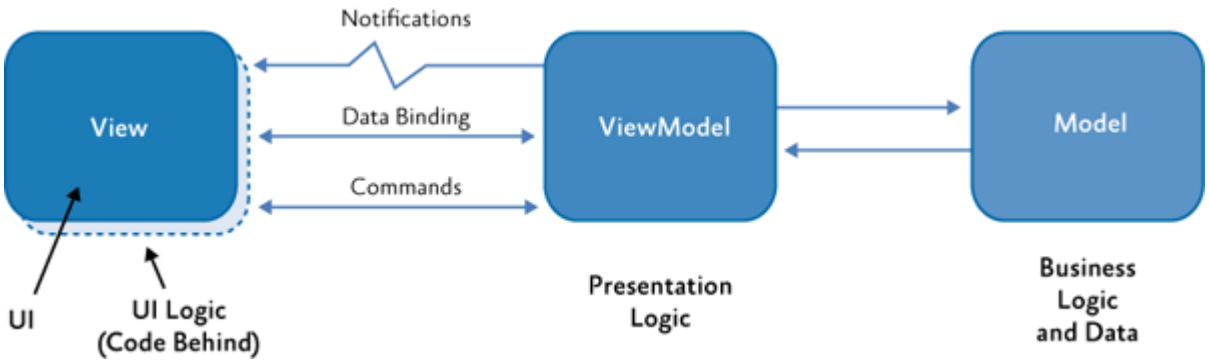

Рис 2. Патерн MVVM

Model

Модель описує використовувані в застосунку дані. Моделі можуть містити логіку, безпосередньо пов'язану цими даними, наприклад, логіку валідації

властивостей моделі. У той же час модель не повинна містити ніякої логіки, пов'язаної з відображенням даних і взаємодією з візуальними елементами управління.

Нерідко модель реалізує інтерфейси INotifyPropertyChanged або INotifyCollectionChanged, які дозволяють повідомляти систему про зміни властивостей моделі. Завдяки цьому полегшується прив'язка до подання, хоча знову ж пряму взаємодію між моделлю і представленням відсутня.

View

View або подання визначає візуальний інтерфейс, через який користувач взаємодіє з застосунком. Стосовно до WPF уявлення - це код в xaml, який визначає інтерфейс у вигляді кнопок, текстових полів та інших візуальних елементів.

Хоча вікно (клас Window) в WPF може містити як інтерфейс в xaml, так і прив'язаний до нього код C #, проте в ідеалі код C # не повинен містити якийсь логіки, крім хіба що конструктора, який викликає метод InitializeComponent і виконує початкову ініціалізацію вікна . Вся ж основна логіка додатки виноситься в компонент ViewModel.

Однак іноді в файлі пов'язаного коду все може знаходитися певна логіка, яку важко реалізувати в рамках паттерна MVVM у ViewModel.

Подання і не виконує жодних події за рідкісним винятком, а виконує дії в основному за допомогою команд.

ViewModel

ViewModel або модель уявлення пов'язує модель і уявлення через механізм прив'язки даних. Якщо в моделі змінюються значення властивостей, при реалізації моделлю інтерфейсу INotifyPropertyChanged автоматично йде зміна відображуваних даних в поданні, хоча безпосередньо модель і уявлення не пов'язані.

ViewModel також містить логіку по отриманню даних з моделі, які потім передаються в уявлення. І також VewModel визначає логіку по оновленню даних в моделі.

Оскільки елементи уявлення, тобто візуальні компоненти типу кнопок, не використовують події, то уявлення взаємодіє з ViewModel за допомогою команд.

Наприклад, користувач хоче зберегти введені в текстове поле дані. Він натискає на кнопку і тим самим відправляє команду під ViewModel. А ViewModel вже отримує передані дані і відповідно до них оновлює модель.

Підсумком застосування патерну MVVM є функціональний розподіл програми на три компонента, які простіше розробляти і тестувати, а також в подальшому модифікувати і підтримувати.

Застосування патерну у застосунку

Архітектура застосунку була побудована використовуючи патерн MVVM та бібліотеку MVVM-Light яка містить в собі набір базових класів для спрощення розробки застосунку. Нижче буде приведено діаграму класів:

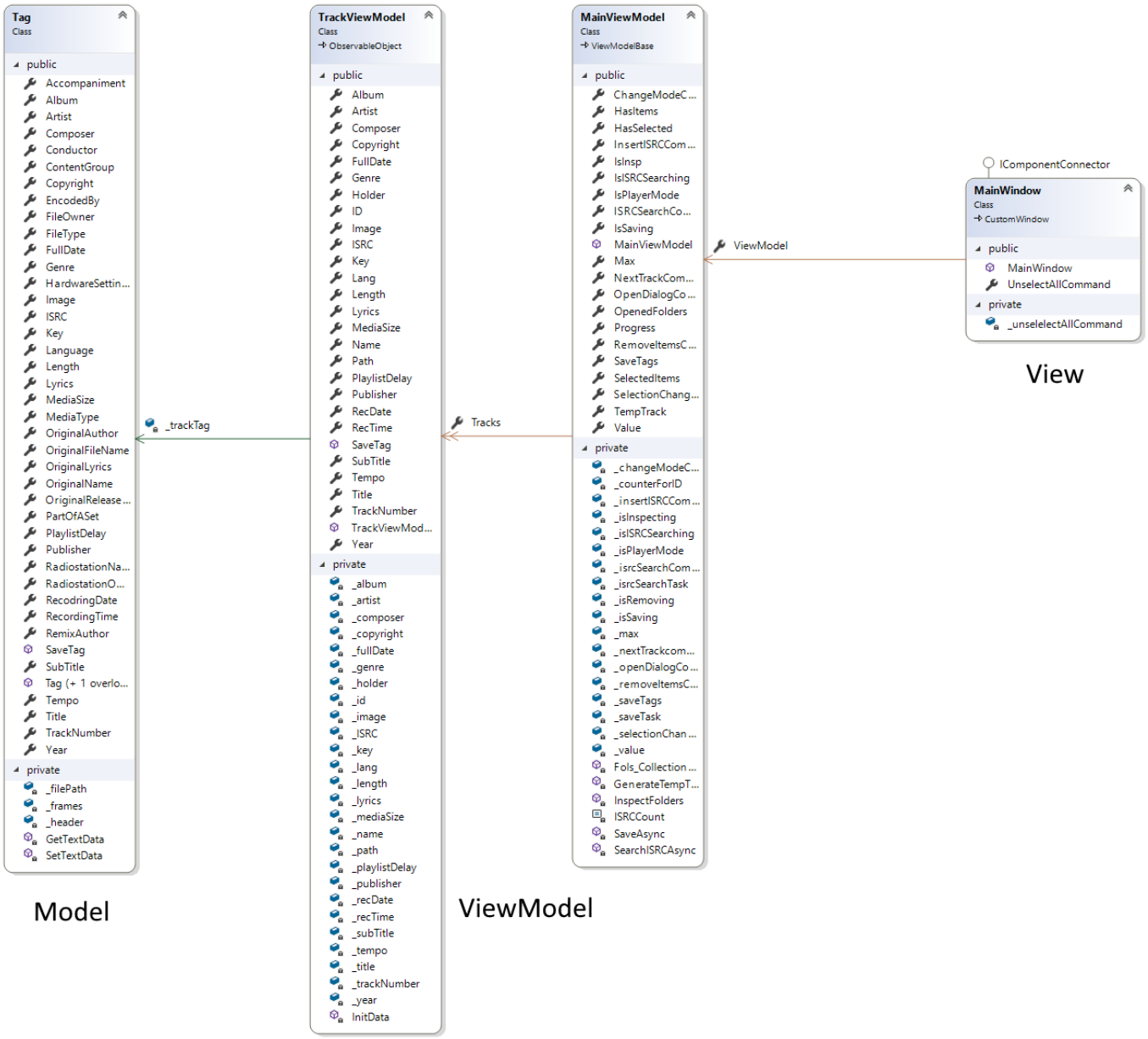

Рис 3. Структура застосунку.

На діаграмі представлені класи які приймають участь в становлені патерну для цього застосунку. Клас Tag відноситься до категорії Model, що представляє собою структуру даних, а саме даних взятих з аудіо-файлів.

Наступний компонент - ViewModel. Перший клас представляє собою розширену версію моделі Tag. Цей клас являє собою джерелом даних для списку аудіо-файлів у застосунку та відображення детальної інформації про них.

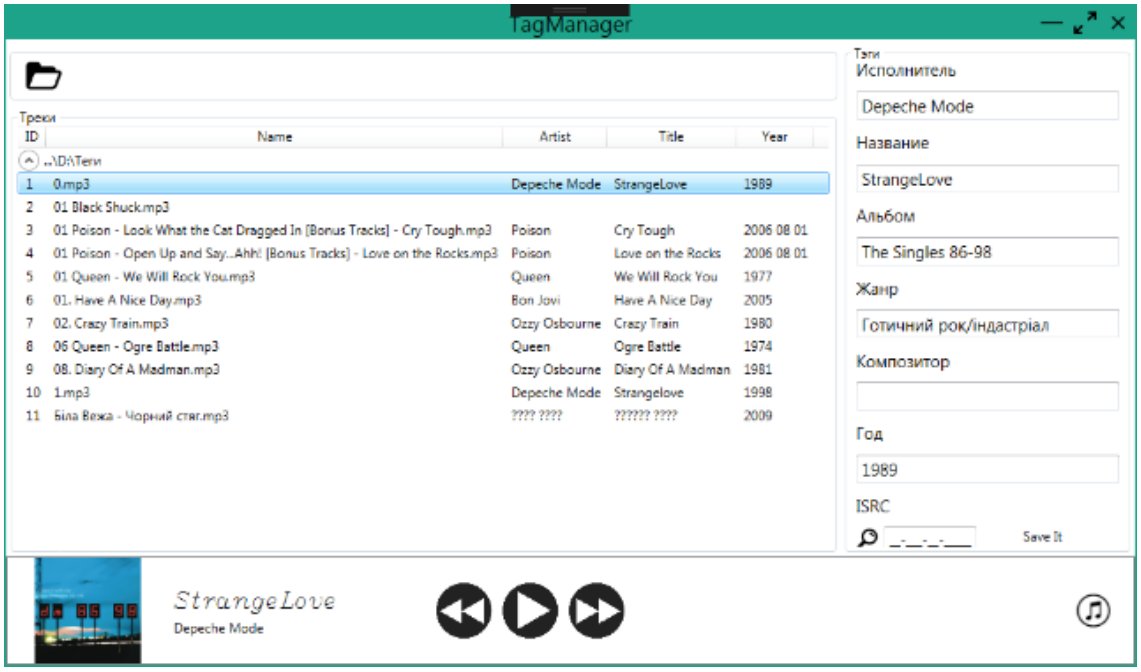

Рис 4. Застосунок.

На скріншоті видно колекцію типу TrackViewModel, обираючи будь-який елемент сбоку відображається детальна інформація.

Всі поля доступні для редагування та є властивостями класу TrackViewModel. Сама колекція знаходиться в іншій ViewModel, під назвою MainViewModel. MainViewModel містить в собі всі дані потрібні для функціонування головної View.

Останній компонент - View, розроблений за допомогою декларативної мови розмітки XAML. Містить в собі компоненти розмітки, поля, кнопки та так звані “біндінги” для зв’язування View та ViewModel.

Діалог вибору файлів

Для більш гнучкого функціонування застосунку було вирішено написати власне діалогове вікно для роботи з файловою системою.

Файлова система (англ. File system) - порядок, що визначає спосіб організації, зберігання та іменування даних на носіях інформації в комп'ютерах, а також в іншому електронному обладнанні: цифрових фотоапаратах, мобільних телефонах і т. П. Файлова система визначає формат вмісту і спосіб фізичного зберігання інформації, яку прийнято групувати у вигляді файлів. Конкретна файлова система визначає розмір імен файлів (і каталогів), максимальний можливий розмір файлу і розділу, набір атрибутів файлу. Деякі файлові системи надають сервісні можливості, наприклад, розмежування доступу або шифрування файлів.

Файлова система пов'язує носій інформації з одного боку і API для доступу до файлів - з іншого. Коли прикладна програма звертається до файлу, вона не має жодного уявлення про те, яким чином розташована інформація в конкретному файлі, так само як і про те, на якому фізичному типі носія (CD, жорсткому диску, магнітній стрічці, блоці флеш-пам'яті або іншому) він записаний. Все, що знає програма - це ім'я файлу, його розмір і атрибути. Ці дані вона отримує від драйвера файлової системи. Саме файлова система встановлює, де і як буде записаний файл на фізичному носії (наприклад, жорсткому диску).

З точки зору операційної системи (ОС), весь диск являє собою набір кластерів (як правило, розміром 512 байт і більше) [1]. Драйвери файлової системи організують кластери в файли і каталоги (реально є файлами, що містять список файлів в цьому каталозі). Ці ж драйвери відстежують, які з кластерів в даний час використовуються, які є вільними, марковані як несправні.

Однак файлова система не обов'язково безпосередньо пов'язана з фізичним носієм інформації. Існують віртуальні файлові системи, а також мережеві файлові системи, які є лише способом доступу до файлів, що знаходяться на віддаленому комп'ютері.

Для його функціонування використовується декілька ViewModel та View.

FolderViewModel являє собою представлення однієї папки, а саме файли передаються у застосунок у вигляді рядків та вже застосунок оперує самими файлами.

OpenFoldersDialogViewModel - модель для представлення діалогового вікна, містить інформацію необхідну для роботи з папками та файлами.

Рис 5. Структура файлового діалогу.

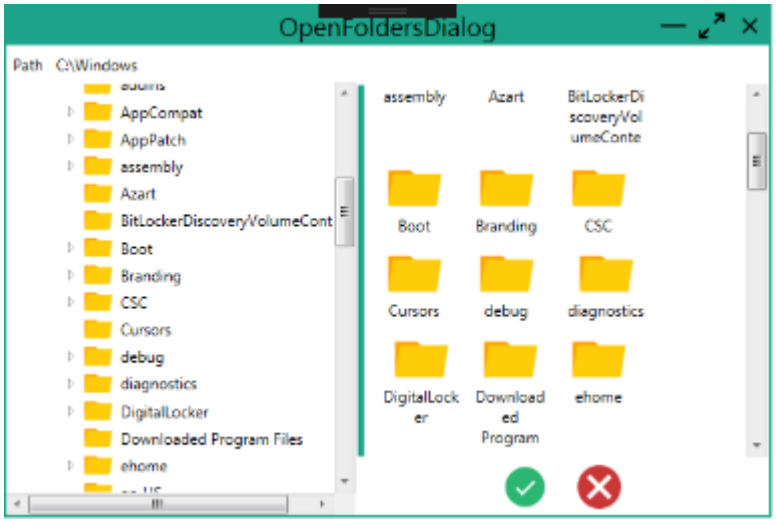

Рис 6. Файловий діаолог у застосунку.

ЗАГАЛЬНІ ВИСНОВКИ

Під час виконання дипломної роботи було спроектовано та розроблено застосунок для редагування метаданих аудіофайлів. Була використана популярна технологія WPF із пакету .NET, яка направлена на розробку застосунків робочого столу використовуючи апаратне прискорення та гнучку систему стилізації застосунків. Застосунок спроектований на базі популярного паттерну MVVM, який дозволяє розділити логіку та представлення застосунку, що робить застосунок більш гнучким. Було використано бібліотеку для роботи з метаданими аудіофайлів для їх декодування, отримання потрібної інформації та запису нових даних до аудіофайлу. Для вибору файлів було створено власне діалогове вікно з використанням технології WPF для розширення можливостей під час вибору файлів та біл естетичного виду.

Поглиблені знання в мові програмування C# та освоєнні навички використання паттернів в застосунках для покращення архітектурної складової. Адже використання популярних практик очищує код та робить його більш доступним для людей які захочуть його розширювати, незважаючи на те були вони залучені до розробки чи ні.

АНОТАЦІЯ

Інформаційна система для роботи з тегами ID3v2 для mp3 файлів. Інтерфейс користувача з використанням технології WPF та паттерну MVVM, робота з файловими ієрархіями.

Дипломна робота на здобуття освітньої кваліфікації «Бакалавр комп'ютерних наук». – Чорноморський національний університет імені Петра Могили, Миколаїв, 2019.

В роботі було розглянуто питання управління файлами які містять в собі метадані. В них міститься багато цікавої та корисної інформації яку можна використати для впорядкування бібліотеки та для отримання цієї інформації. Іноді важливо мати можливість для редагування цих даних, синхронізування з хмарним сховищем або базою даних.

Архітектура програми має відповідати всім практикам розробки програмного забезпечення з використанням патерном MVVM. Який дозволить відділити логіку застосунку від дизайну. В купі з WPF розділення дизайну та логіки дозволяє виграти на декларативній мові розмітки, яка дозволяє використовувати можливості персонального комп'ютера з максимальною віддачею.

Ключові слова: метдані, MVVM, декларативна мова розмітки, архітектура, WPF.

Information system for working with ID3v2 tags for mp3 files. User interface using WPF technology and MVVM pattern, work with file hierarchies.

Nowadays, it's hard to find a person who does not store music files on the phone or on a personal computer. Most of these users encounter the problem of sorting and displaying the music library.

This problem can be solved based on the metadata containing the music files. They contain a lot of interesting and useful information that can be used to organize the library and to obtain this information. It is sometimes important to have the ability to edit this data, synchronize with a cloud storage or database.

Oddly enough, for such purposes, programs with full functionality on the personal computer turned out to be a bit, suggesting the creation of a new program with more functionality and capabilities, white comfort.

The architecture of the program must be consistent with all software development practices using the MVVM pattern. Which will allow separating the logic of the application from the design. In a package with WPF, design and logic divisions allow you to win markup language in declarative language, which allows you to use the capabilities of a personal computer with maximum efficiency.

The purpose of the thesis is

Develop an application for editing audio metadata and work with file hierarchies.

The practical value of the results obtained.

Get skills for developing desktop applications using the MVVM pattern.

Keywords: MVVM, declarative markup language, architecture, WPF, metadata.