ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра інженерії програмного забезпечення

Пояснювальна записка

до бакалаврської роботи
на ступінь вищої освіти бакалавр
на тему: **«Розробка десктопного додатку для перегляду відеофайлів мовою C#»**

Виконав: студент 4 курсу, групи ПД-41
спеціальності
121 Інженерія програмного забезпечення
(шифр і назва спеціальності/спеціалізації)

Кисельов В.О.
(прізвище та ініціали)

Керівник Трінтіна Н.А.
(прізвище та ініціали)

Рецензент _____
(прізвище та ініціали)

Київ –2021

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра Інженерії програмного забезпечення
Ступінь вищої освіти - «Бакалавр»
Спеціальність – 121 Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри
інженерії програмного забезпечення

Негоденко О.В.

“____”_____2021 року

ЗАВДАННЯ
НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

Кисельов В’ячеслав Олександрович
(прізвище, ім’я, по батькові)

1. Тема роботи: «Розробка десктопного додатку для перегляду відеофайлів мовою C#.»

Керівник роботи Дібрівний Олесь Андрійович, доктор філософії
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від “12”березня 2021року №65.

2. Строк подання студентом роботи 1.06.2021

3. Вихідні дані до роботи:
Microsoft visual studio;
WPF;
GitHub;
SourceTree;
Офіційна документація мови програмування C#.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
4.1 АНАЛІЗ ТА ХАРАКТЕРИСТИКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ
4.2 ОПИС ПРОГАМНИХ ЗАСОБІВ РЕАЛІЗАЦІЇ
4.3 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ
4.4 ПРОЕКТУВАННЯ
4.5 Тестування та експлуатація

5. Перелік графічного матеріалу
1. Титульний слайд
2. Мета, об’єкт дослідження, предмет дослідження
3. Аналіз існуючих аналогів

4. Технічне завдання
5. Засоби програмної реалізації
6. Методи та класи
7. Висновки

6. Дата видачі завдання 19.04.2021

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи до	Примітка
1	Вибір теми бакалаврської роботи	01.10.2021	Виконано
2	Дослідження актуальності теми	21.04.2021	Виконано
3	Розгляд аналогів	11.04.2021	Виконано
4	Проектування логіки проекту	17.04.2021	Виконано
5	Реалізація функцій проекту	20.04.2021	Виконано
6	Здача роботи в деканат	1.06.2021	

Студент _____ Кисельов В.О.
(підпис) (прізвище та ініціали)

Керівник роботи_____ Дібрівний О.А.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Текстова частина бакалаврської роботи с.69, рис. 47, джерел 14.

Об'єкт дослідження – програмне забезпечення для роботи з файлами мультімедії.

Предмет дослідження – додаток для перегляду відеофайлів.

Мета дослідження – розробити десктопний додаток для перегляду відеофайлів.

У роботі проведено аналіз існуючих рішень програмного забезпечення, які дозволяють роботу з файлами мультімедії. Розглянуто використання програмних засобів, аналіз їх переваг та недоліків. Було вибрано найкращі програмні інструменти для створення додатку.

Проведено описання використаних програмних засобів та середи розробки.

Реалізована система на мові програмування C#, на основі графічного інтерфейсу .NET Framework, з використанням технології WPF (Windows Presentation Foundation).

Ключові слова: C#, .Net, WPF, XAML, SourceTree, Об’єктно орієнтоване програмування.

ЗМІСТ

ВСТУП

Об'єкт дослідження – програмне забезпечення для роботи з файлами мультімедії.

Предмет дослідження – додаток для перегляду відеофайлів.

Мета дослідження – розробити десктопний додаток для перегляду відеофайлів.

Новизна проекту – удосконалення функціоналу існуючих додатків, розробка додатку без платного функціоналу та інтерфейсу, який зрозумілий та очевидний для кінцевого користувача.

У дипломному проекті було проведено аналіз між існуючими аналогами відеплеєрів та проведено аналіз недоліків.

Додаток був розроблений на мові програмування C#, на основі графічного інтерфейсу .NET Framework з використанням техології WPF та мови XAML. Всі об’єкти інтерфейсу були створені в графічному редакторі Adobe Photoshop. Додаток має внутрішню ієрархічну структуру, кожен компонент логічно і програмно пов’язаний із суміжним до нього об’єктом. Основними об’єктами меню є кнопки з різним призначенням і візуальним оформленням.

Досліджено та створено додаток, що дозволяє переглядати відеофайли.

1.АНАЛІЗ ТА ХАРАКТЕРИСТИКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Актуальність

Невід’ємна частина нашого життя – це відпочинок.

Оскільки багато з нас працюють довгими годинами, щоб не відставати від постійно зростаючого навантаження, і ризикуючи вигоряти лише для того, щоб не відставати, про важливість відпочинку легко забути. Ми уникаємо відпусток і замість цього боремося зі стресом, хворобами та постійним тиском, щоб знайти час для всіх своїх зобов’язань поза роботою. Але відпочинок - це не те, що можна сприймати легковажно, це важлива частина вашої найкращої роботи, підвищення продуктивності на роботі, і це те, для чого більша частина з нас, повинна розставити пріоритети.

У наш час важко вирватись і поїхати кудись, сходити в кіно на улюблений фільм або навіть прогулятися – це ризик, а домашні розваги – досить обмежені. Але у наші дні є способи, якими ви можете насолоджуватися фільмами, не обов’язково відвідуючи кінотеатри. Тепер ви можете відпочити та насолодитися своїм фільмом, не виходячи з власного будинку чи двору.

Перегляд фільмів може дати чудову можливість зняти стрес із нашого життя. Медично доведено, що стрес викликаний постійним накопиченням напруги в людині, а коли немає способу його зняти, то стрес неминучий. Одним з найкращих способів зняти накопичення напруги є перегляд улюблених фільмів або серіалів. Перегляд фільмів насправді може викликати поштовх до вашого життя. Спостереження за тим, як звичайні люди перетворюються на героїв під час історії, може надихнути або спонукати вас робити те саме у своєму повсякденному житті. Ви розумієте, що такі звичайні люди, як ти, і я також здатні на великі речі в житті.

1.2 Аналіз аналогів

На сьогоднішній день можна знайти багато десктопних відеоплеєрів.

Один з перших таких додатків це – “**BS.player**”. Доволі популярний медіаплеєр, який має широкий функціонал.

Рисунок 1.1 – Головне меню, підпункт «Видео»

Даний медіаплеєр набув широкої популярності через свою низьку вимогливість до обладнання користувача. Кількість завантажень приблизно 1.8 міліонів. Також у нього доволі широкий спектр функціоналу рис.1.1:

1. Відео.
2. Аудіо.
3. Радіо.
4. ТВ.

Основною проблемою даного додатку є те, що весь додатковий функціонал є платним та інтерфейс доволі складний, для звичайного користувача.

Також доволі популярним є “**MP4 Player**”.

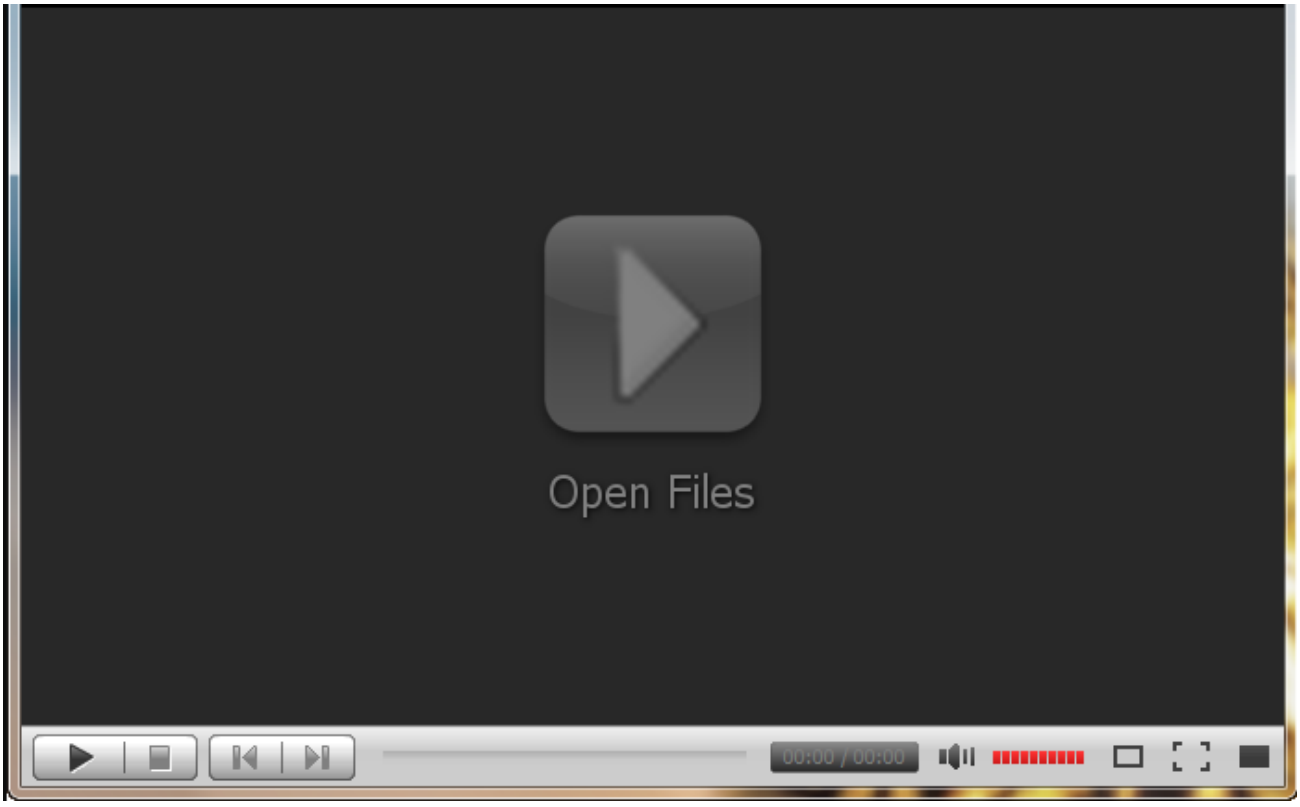


Рисунок 1.2 – Головне меню “MP4 Player”

Даний додаток набув широкої популярності через свій простий інтерфейс рис.1.2 та невимогливість до обладнання користувача. Також в ньому доступна функція створення плейлістів. Головними його недоліками є те, що він платний та не підтримує ніяких мов, крім англійської.

“MKV Player” – доволі маловідомий плеєр.

Даний додаток не набув популярності через свій скудний функціонал та дуже відштовхуючий інтерфейс рис.1.3, головна його фішка - це можливість відтворення файлів формату MKV, що є важливим для багатьох користувачів.

Основними його недоліками є те, що він не має можливості відтворення відеофайлів інших форматів, крім MKV.

MKV Player – не найкращий вибір для відтворення відео чи аудіо. Однак він робить ту роботу, яку може зробити надзвичайно добре. Якщо все, що потрібно, - це відтворювати музичні композиції або дивитися фільми, то ця програма ідеально підходить. Для більш досконалого використання знадобиться краща альтернатива.

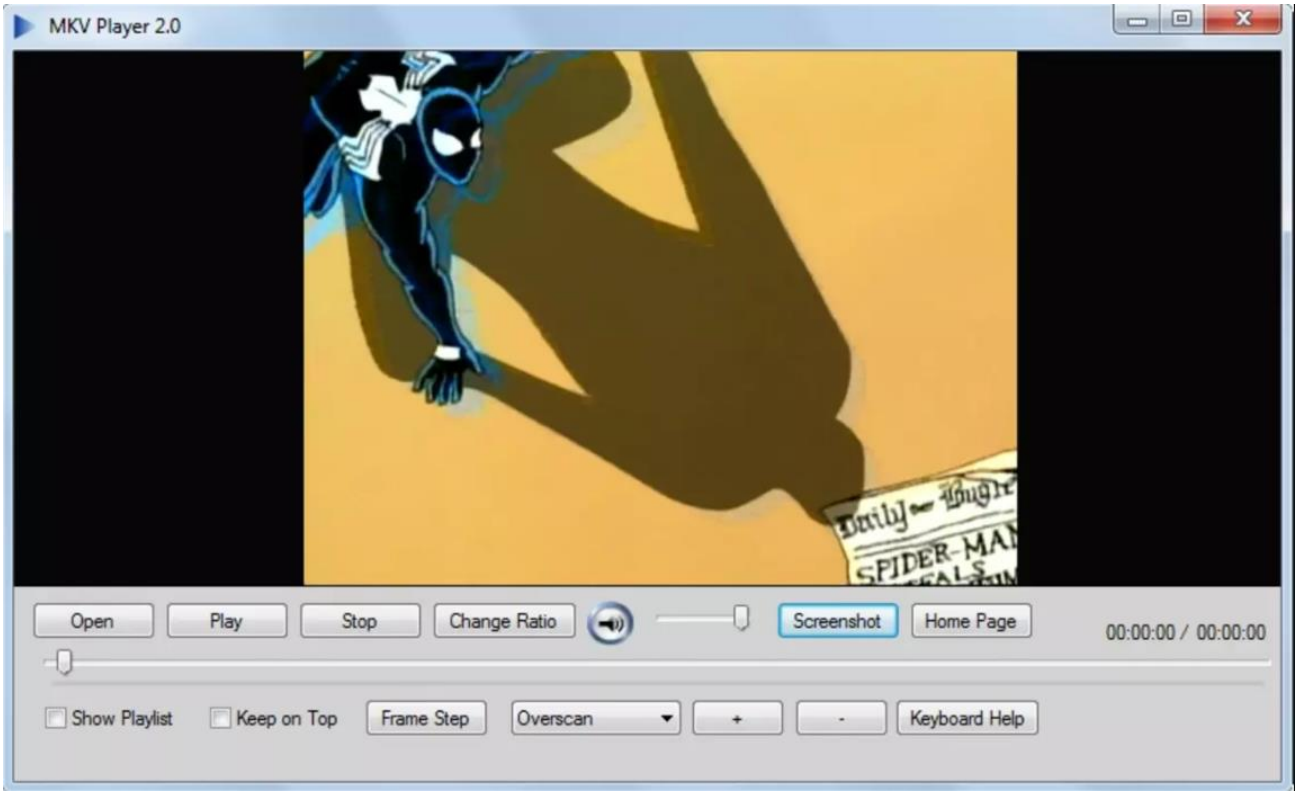


Рисунок 1.3 – Головне меню “MKV Player”

“Winamp” – один з найпопулярніших медіплеєрів.

Рисунок 1.4 – Головне меню “Winamp”

Даний додаток був флагманом медіаплеєрів ще з 2007 років. Розроблений компанією **“Nullsoft”** ще в 1997 році, він не набув широкої популярності, але

Winamp 5 стала новим подихом для додатку і його використовували більшість користувачів. Головними особливостями додатку:

1. Підтримка аудіо та відео.
2. Підтримка найпопулярніших аудіо та відео форматів.
3. Підтримка функції “Потокового мовлення”.
4. Стильний графічний інтерфейс рис.1.4.
5. Підтримка медіабібліотек.
6. Автоматична сортировка плейлістів.

“Winamp” – став гарним прикладом розробки медіаплеєрів. Основним недоліком даного додатку став занепад розробки та перехід до платного функціоналу, який почав відштовхувати багатьох користувачів.

1.3 Постановка технічного завдання

Провівши аналіз аналогів та визначивши їх недоліки, було вирішено створювати додаток на мові c#. Це одна з найкращих мов програмування для створення десктопних рішень.

Також важливо обрати середовище розробки серед:

1. Eclipse.
2. IntelliJ IDEA.
3. Microsoft Visual Studio 2019.

Eclipse - це безкоштовне середовище розробки від не комерційної організації Eclipse Foundation. Узагальнюючи, програма – це основа для підключення модулів.

Eclipse продуктивне середовище, що може запускатися на слабких комп’ютерах, з офіційною російськомовною версією та з великою кількістю доповнень (таких, як робота з БД, робота з сервером та інше). Дане середовище надає велику кілкість функціоналу, та забезпечує роботу з базами даних.

IntelliJ IDEA – одне з популярних середовищ розробки. Доступене в двух версіях, безкоштовній та платній. Порівнюючи з Eclipse, IntelliJ IDEA має кращі механізми для відлагодження програми та більш зручний інтерфейс. Але серед

недолікив можна виділити те, що використовувати мову C# на цьому середовищі розробки буде не зручно, тому що підтримка .net у ній налаштовується плагінами.

Microsoft Visual Studio 2019 – найкращий вибір для розробки десктопного додатку на платформі .NET Framework використовуючи мову c#, оскільки і то, і то є продуктами Microsoft і вони відмінно підходять для роботи з один одним.

Є безплатна версія Community, яка підтримує всі інструменти розробки для мови c#, яка ідеально підійде для роботи. Також дане середовище підтримує IntelliSense для оптимізації роботи коду.

Microsoft Visual Studio підтримує UI фреймворк WPF, який є ідеальним рішенням для розробки десктопних додатків. Він дозволить реалізувати всі наступні вимоги до додатку:

1. Можливісь відтворення та перегляду відеофайлів.
2. Можливість маніпуляції з відеофайлом.
3. Можливість створення плейлистів.
4. Гнучкість та адаптивність до змін.
5. Гнучкий та простий інтерфейс.
6. Наявність активних кнопок, зрозумілих користувачу.
7. Наявність простої і зрозумілої робочої області з можливостю програвання відеофайлів.
8. Можливість створення плейлістів.

WPF дозволить реалізувати гнучкий та адаптивний інтерфейс, за допомогою мови XAML.

Супроводжуватись додаток в ході розробки та тестування буде за допомогою системи контроля версії та управління версіями git, використовуючи безкоштовний візуальний клієнт SourceTree.

2. ОПИС ПРОГАМНИХ ЗАСОБІВ РЕАЛІЗАЦІЇ

2.1 Мова програмування C#

C# – сучасна об'єктно-орієнтована мова програмування і належить сімейству мов C. Вона може здатися добре знайомою кожному, хто працював з C, C ++, Java або JavaScript.

C# належить до об’єктно-орієнтованої мови, але підтримує також і компонентно-орієнтоване програмування. Розробка сучасних додатків з кожним днем набуває все більшого розташування до створення програмних компонентів у вигляді автономних пакетів, що допомагають реалізувати окремі функціональні можливості. Головна особливість таких компонентів в тому, що вони являють собою модель програмування звластивостями, методами і подіями. У них є атрибути, що надають декларативні відомості про компоненті. Вони включають в себе власну документацію. C# надає мовні конструкції, безпосередньо підтримують таку концепцію роботи. Завдяки цьому C# підходить для того, щоб створювати та застосовувати програмні компоненти.

C# в значній мірі підтримується Microsoft. При багах синтаксичні поліпшення та нові функції оновлюються швидше, ніж в інших мовах програмування. Ця мова є однією з найпопулярніших мов програмування. Це важливо для розробників, оскільки популярність мови прямо пропорційна тому, наскільки для нього будуть доступні онлайн-матеріали. Найчастіше, всі звертаються до Google або Stack Overflow для вирішення завдань в розробці і найчастіше можна знайти велику кількість відповідей по C#. Це заощадить величезну кількість часу новачкам при вирішенні різних завдань в розробці.

Також свою популярність C# набув за рахунок гнучкості, якщо порівнюватиз деякими іншими мовами програмування. Є велика кількість різних додатків, які можуть бути написані за допомогою C#, .Net і Visual Studio:

1. Додатки для Windows.
2. Мобільні додатки.
3. Веб-додатки.

4. Ігри.
5. Додатки для Android і ios, які розробляються за допомогою додатковихфреймворків, таких як Xamarin або Mono.

Звичайно, всі ці речі можливо виконувати і за допомогою інших мов програмування, але зазвичай, в таких випадках, використовуються сторонні інструменти інших розробників. Розробники, які працюють з C# достатньо довго мають вже згрупований набір інструментів, які підтримуються Microsoft для розробки різних типів програми.

2.2 WPF

WPF - розшифровується як Windows Presentation Foundation, - це підхід Microsoft до фреймворку GUI, що використовується з фреймворком .NET рис. 2.1.

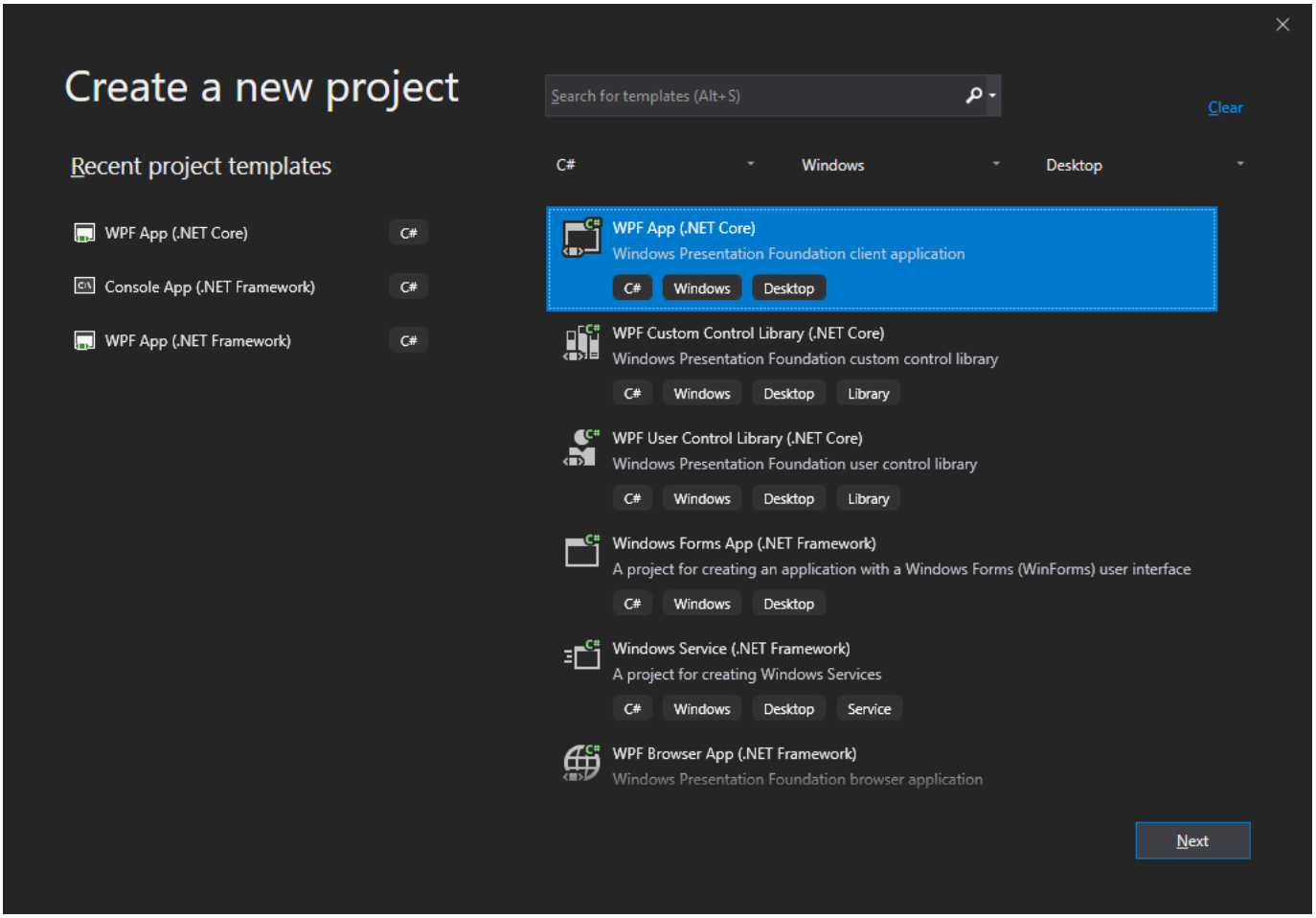


Рисунок 2.1 – Створення проекту використовуючи шаблон WPF

Але що таке GUI? GUI розшифровується як Графічний користувальницький інтерфейс. Windows має графічний інтерфейс для роботи з вашим комп'ютером, а браузер.

Структура графічного інтерфейсу дозволяє створювати додаток із широким набором елементів графічного інтерфейсу, таких як мітки, текстові поля та інші добре відомі елементи. Без графічного інтерфейсу користувача вам довелося б малювати ці елементи вручну та обробляти всі сценарії взаємодії з користувачами, такі як введення тексту та миші. Це БАГАТО роботи, тому натомість більшість розробників використовуватимуть графічний інтерфейс, який виконуватиме всю основну роботу і дозволить розробникам зосередитися на створенні чудових додатків.

Існує багато графічних інтерфейсів, але для розробників .NET найбільш цікавими на даний момент є WinForms та WPF. WPF є найновішим, але Microsoft все ще розвиває та підтримує WinForms. Між двома фреймворками є досить багато відмінностей, але їх призначення однакове - полегшити створення додатків із чудовим графічним інтерфейсом.

WPF - це механізм, який відповідає за створення, відображення та маніпулювання користувальницькими інтерфейсами, документами, зображеннями, фільмами та медіа в операційних системах Windows 7 та пізніших версіях. WPF - це набір бібліотек, які мають усі функції, необхідні для створення, запуску, запуску та керування клієнтськими програмами Windows.

XAML - це нова описова мова програмування, розроблена корпорацією Майкрософт для написання користувацьких інтерфейсів для керованих додатків наступного покоління.

XAML використовується для побудови користувальницьких інтерфейсів для Windows та мобільних додатків, які використовують Windows Presentation Foundation (WPF), UWP та Xamarin Forms. Призначення XAML просте - створити користувальницькі інтерфейси за допомогою мови розмітки, схожої на XML. Здебільшого ви будете використовувати дизайнер для створення XAML, але ви можете вільно маніпулювати XAML вручну.

XAML використовує формат XML для елементів та атрибутів. Кожен елемент у XAML представляє об'єкт, який є екземпляром типу. Сфера застосування типу (клас, перерахування тощо) визначається простором імен, який фізично знаходиться у збірці (DLL) бібліотеки .NET Framework. Подібно до XML, синтаксис елемента XAML завжди починається з відкритої кутової дужки (<) і закінчується закритою кутовою дужкою (>). Кожен тег елемента також має початковий та кінцевий теги рис. 2.2.

Рисунок 2.2 – приклад використання XAML

Хоча XAML використовується для побудови користувальницьких інтерфейсів у WPF, C # використовується як кодова мова в WPF. Хоча Windows та їх елементи керування створюються в XAML під час проектування, вони також можуть бути створені під час виконання за допомогою мови C #.

C # також використовується для написання програмування всіх подій та бізнес-логіки. Усі дії, події та візуалізація виконуються в коді, що стоїть за кодом C #.

WPF підтримує два типи ресурсів, статичний та динамічний.

Статичні ресурси: Статичний ресурс буде виконано під час завантаження XAML, до того, як програма дійсно запущена. Він буде виконаний лише один раз, а будь-які зміни до словника ресурсів ігноруються.

Динамічні ресурси: Динамічний ресурс призначає значення властивості під час завантаження, але фактично не шукає ресурс до часу виконання. Це відкладає пошук ресурсу, поки він не буде необхідний під час виконання.

Елемент “Style” рис.2.3 у XAML представляє стиль

“Style” - це спосіб групувати подібні властивості в одному об’єкті “Style” та

застосовувати до кількох об’єктів. Зазвичай стиль додається до ресурсів FrameworkElement. Даний вираз “x:” є унікальним ідентифікатором ключа стилю.

```
<Window.Resources>
        <Style x:Key="MyButtonStyle">
            <Setter Property="Control.FontFamily" Value="Calibri"></Setter>
            <Setter Property="Control.FontSize" Value="18"></Setter>
            <Setter Property="Control.FontWeight" Value="Bold"></Setter>
            <Setter Property="Control.Padding" Value="5"></Setter>
            <Setter Property="Control.Margin" Value="5"></Setter>
        </Style>
</Window.Resources>
```

Рисунок 2.3 – Використання стилю у WPF

Шаблони є невід’ємною частиною дизайну інтерфейсу користувача в WPF. WPF має наступні три типи шаблонів: **Control Template, Items Panel Template та Data Template**.

Control Template або **Шаблон керування** визначає зовнішній вигляд елемента керування. Ми можемо змінити або визначити новий вигляд та зовнішній вигляд елемента керування, просто змінивши Control Template елемента управління. Також, даний шаблон ще більш корисний, коли ви пишете власні елементи керування. Ви можете визначити зовнішній вигляд елементів керування. Наприклад, елемент керування кнопкою WPF має прямокутний макет, але за допомогою **ControlTemplates** ви можете створити власну кнопку, яка має круговий макет і змінює свій колір, коли ви наводите курсор миші на неї або натискаєте.

Items Panel Template – визначає стиль макету кастомного або готового елементу.

Data Template – шаблон, який визначає стиль відображення данних.

Також, у WPF доступна прив’язка данних.

Прив’язка дозволяє зв’язати вихідний об’єкт з деяким елементом управління. Існують такі типи прив’язок:

1. Одностороння прив'язка: прив'язує джерело до інтерфейсу.
2. Двостороння прив'язка: прив'язує джерело до інтерфейсу і назад до джерела.

Інтерфейс **INotifyPropertyChanged** дозволяє джерелам взаємодіяти з

інтерфейсом та оновлювати його при зміні значень. Для прив'язки об'єкта або списку до елемента потрібно встановити властивість **DataContext**. Можна прив'язати об'єкт або список об'єктів, а можна прив'язати один елемент до іншого. Щоб налаштувати спосіб відображення пов'язаних даних, можна встановити **DataTemplate** з елемента керування.

У WPF розвинена система тригерів, як в C#.

Тригери використовуються для виконання певних дій при виконанні заданої умови. Тригери використовуються для створення візуальних ефектів на елементах управління та елементах фреймворку. Тригери є частинами стилів і завжди визначаються всередині стилю.

В основному існує три типи тригерів, це:

1.Тригер власності.

2.Тригер даних.

3.Тригер подій.

WPF підтримує медіаелементи, що стане основою нашого додатку рис.2.4.

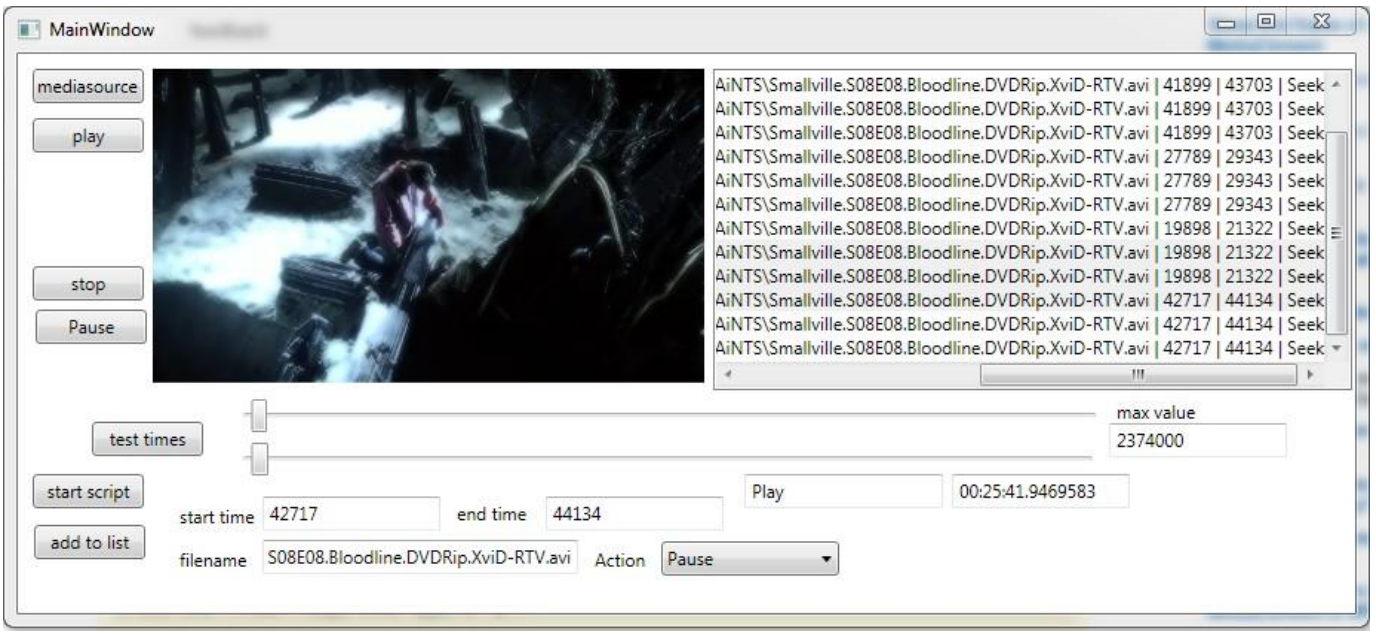


Рисунок 2.4 – Приклад використання MediaElement

WPF має два класи роботи з аудіо, відео та відео зі звуком - MediaElement та MediaPlayer. MediaElement є частиною XAML UIElement і підтримується як XAML,

так і кодом WPF, але MediaPlayer доступний лише в коді WPF.

WPF має набір розширених елементів керування інтерфейсом. Ці елементи керування підтримують такі візуальні дії, як перетягування, встановлення властивостей та подій, прив'язка даних та налаштування ресурсів та шаблонів.

Список основних елементів керування, які підтримує WPF:

1. DatePicker Control.
2. DockPanel Control.
3. ListBox Control.
4. ComboBox Control.
5. MessageBox Control.
6. DataGrid.
7. ProgressBar.
8. TreeView.

Елемент керування **DatePicker** рис.2.5 використовується для створення візуального DatePicker, який дозволяє користувачеві вибрати дату та запускати подію при виборі дати. У цій статті показано, як створити та використовувати елемент керування DatePicker у WPF за допомогою XAML та C #.

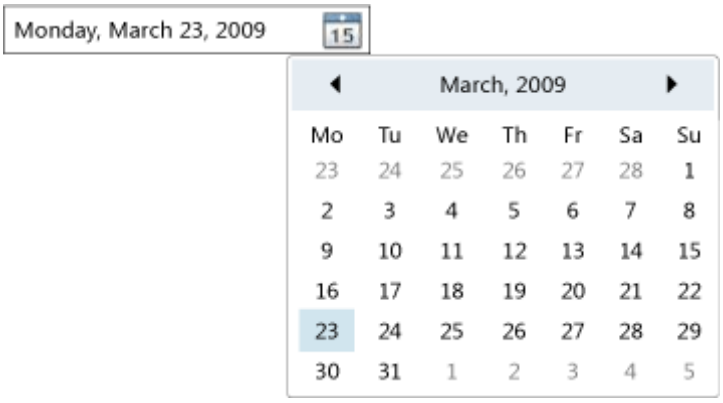


Рисунок 2.5 – Приклад використання DatePicker Control

DockPanel використовується для стикування дочірніх елементів у лівому, правому, верхньому та нижньому положеннях відповідних елементів рис.2.6.

Положення дочірніх елементів визначається властивістю Dock відповідних дочірніх елементів та відносним порядком цих дочірніх елементів. Залишається

значення за замовчуванням властивості Dock.

Властивість Dock має тип enumDock (перерахування), що має значення Left, Right, Top і Bottom.

```
<DockPanel Name="dcPanel">
    <Button Name="TopRect" DockPanel.Dock="Top" Background="LightGreen" Height="50" Content="Top
    <Button Name="LeftRect" DockPanel.Dock="Left" Background="LightBlue" Width="50" Content="Lef
    <Button Name="RightRect" DockPanel.Dock="Right" Background="LightSalmon" Width="50" Content=
    <Button Name="BottomRect" DockPanel.Dock="Bottom" Background="LightCyan" Height="50" />
    <Button Name="Fill" Background="LightGray" />
</DockPanel>
```

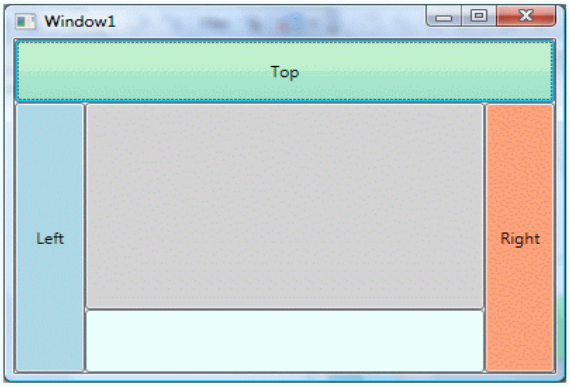


Рисунок 2.6 – Приклад використання DockPanel

WPF ListBox - Елемент XAML ListBox рис.2.7 предствляє можливість створення листів елементів. Властивості Width і Height представляють ширину та висоту ListBox. Властивість Name представляє ім'я елемента керування, який є унікальним ідентифікатором елемента управління. Властивість Margin вказує розташування ListBox на батьківському елементі управління.

Даний елемент керування підтримує динамічне додавання даних, тобто, під час роботи програми, що відкриває більше можливостей для динамічної роботи з даними.

Також, даний елемент підтримує можливість завантаження файли зображення. Файли зображення можна редагувати та підстроювати їх під потрібні розміри.

ListBox має можливість завантаження даних з бази даних, та динамічної їх обробки. При прив’язці даного елементу контроля до класу, який взаємодіє з базою даних, дані в класі будуть динамічно оновлюватись, при редагувані їх.

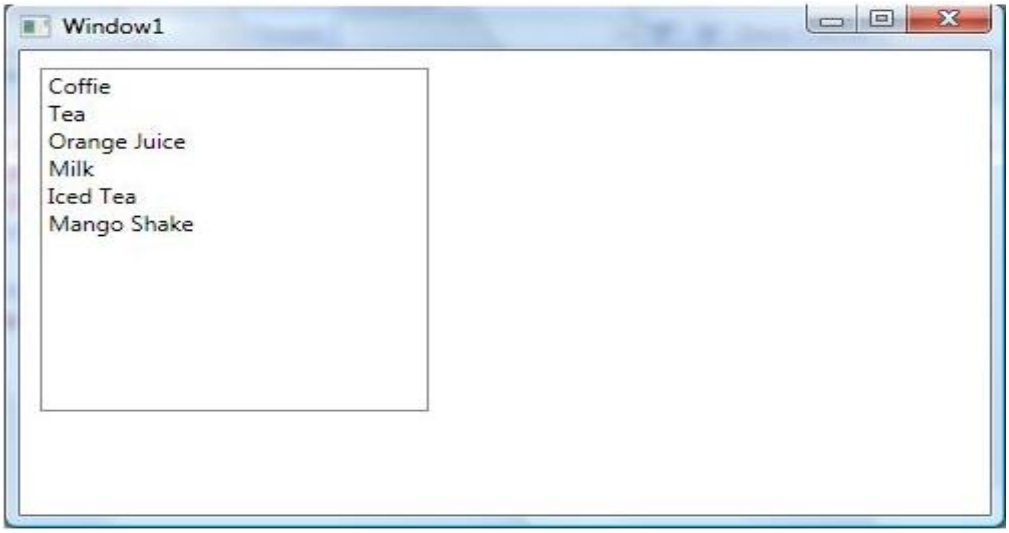


Рисунок 2.7 – Приклад використання ListBox

ComboBox - це елемент керування, який працює як елемент керування ListBox, але одночасно видно лише один елемент із колекції, і натискання на ComboBox робить колекцію видимою і дозволяє користувачам вибрати елемент із колекції. На відміну від елемента керування ListBox, ComboBox не має декількох елементів вибору. Елемент керування

ComboBox - це комбінація трьох елементів керування - кнопки, спливаючого вікна та текстового ящика рис.2.8. Елемент керування кнопкою використовується для показу або приховування доступних елементів, а елемент керування спливаючого вікна відображає елементи та дозволяє користувачеві вибрати один елемент зі списку. Потім елемент керування TextBox відображає вибраний елемент.

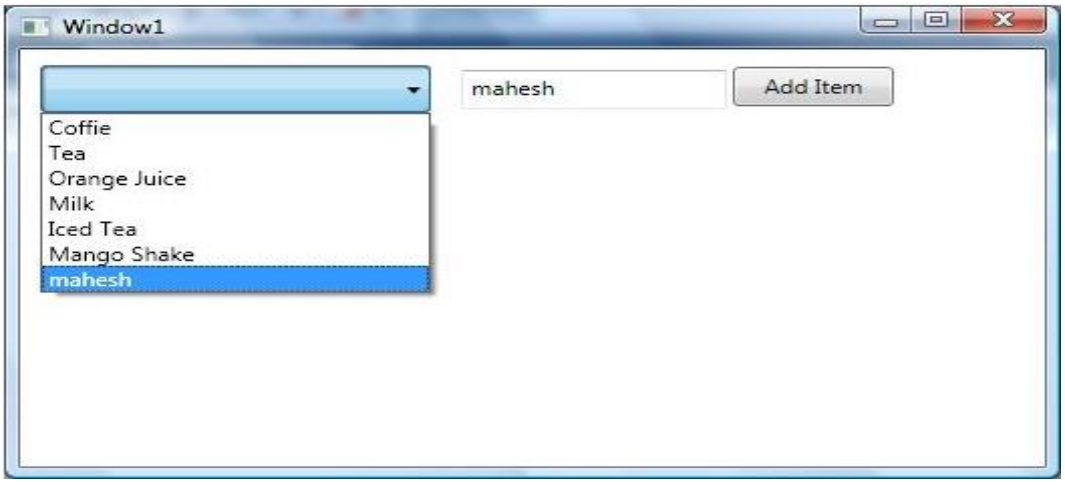


Рисунок 2.8 – Приклад використання ComboBox

Клас **MessageBox** у WPF представляє діалогове вікно модального вікна повідомлення, яке визначено у просторі імен System.Windows рис.2.9.

Show.MessageBox - єдиний метод, який використовується для відображення вікна повідомлення. Метод Show повертає перерахування MessageBoxResult, яке має значення None, OK, Cancel, Yes і No. Ми можемо використовувати MessageBoxResult, щоб визначити, на яку кнопку було натиснуто MessageBox, і вжити відповідних дій.

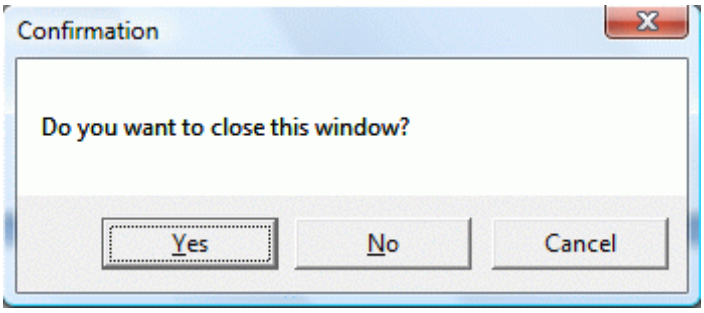


Рисунок 2.9 – Приклад використання MessageBox

Тег ProgressBar у XAML представляє елемент керування WPF ProgressBar.

<ProgressBar> </ProgressBar>

Властивості Width і Height представляють ширину і висоту панелі ProgressBar. Властивість Name представляє ім'я елемента керування, який є унікальним ідентифікатором елемента керування. Властивість Margin вказує розташування ProgressBar на батьківському елементі управління рис 2.10. Властивості HorizontalAlignment та VerticalAlignment використовуються для встановлення горизонтального та вертикального вирівнювання.

ProgressBar підтримує динамічне відстежування даних, що ідеально підійде для створення відеотаймлайну, та динамічного його відестежування.

Також, даний елемент керування підтримує методи контролю від користувача, що дозволить створити зручний та контролюємий відеотаймлайн.

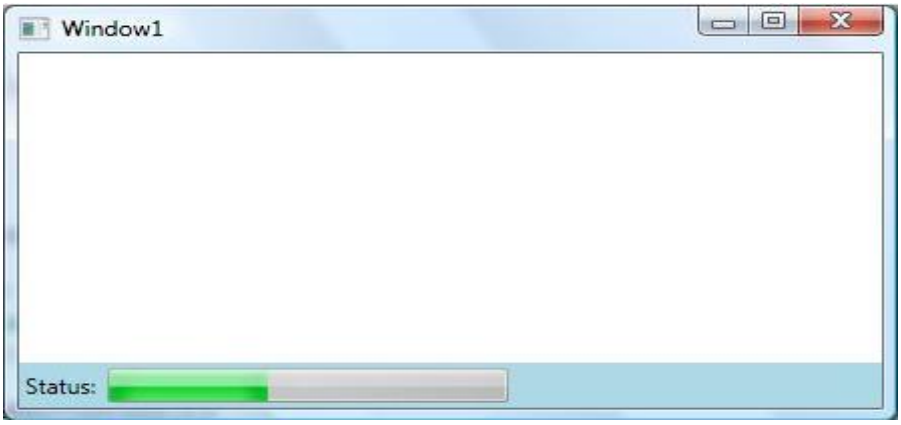


Рисунок 2.10 – Приклад використання ProgressBar

WPF – підтримує TreeView.

TreeView представляє дані в ієрархічному виді, поданні у батьківських дочірніх відносинах, де батьківський вузол може бути розширений або згорнутий. Ліва бічна панель Провідника Windows є прикладом TreeView.

WPF – підтримує MediaTransportClass, який представляє елементи керування для відтворенням елемента медіаплеєра.

MediaTransportClass дозволяють користувачам взаємодіяти зі своїми носіями, забезпечуючи стандартний досвід відтворення, що складається з різних кнопок, включаючи відтворення, паузу, субтитри та інші.

Він має безліч властивостей, що дозволяють легко налаштовувати інтерфейс та конфігурацію, які кнопки видно або ввімкнено. Його можна використовувати, щоб полегшити користувачам управління своїм аудіо- та відеовмістом.

Клас MediaTransportControls призначений для використання лише у поєднанні з елементом управління MediaElement або MediaPlayerElement рис 2.11. Він не функціонує як самостійний елемент керування.

Ви отримуєте доступ до екземпляра MediaTransportControls та через властивість MediaPlayerElement.TransportControls. Зазвичай, в ній описано макет, та 2 рядка елемнтів керування, для створення зручного контролю додатку. По-факту, даний клас, це складний елемент, який складається з декількох інших елементів керування XAML, які містяться в корні елемента Grid.

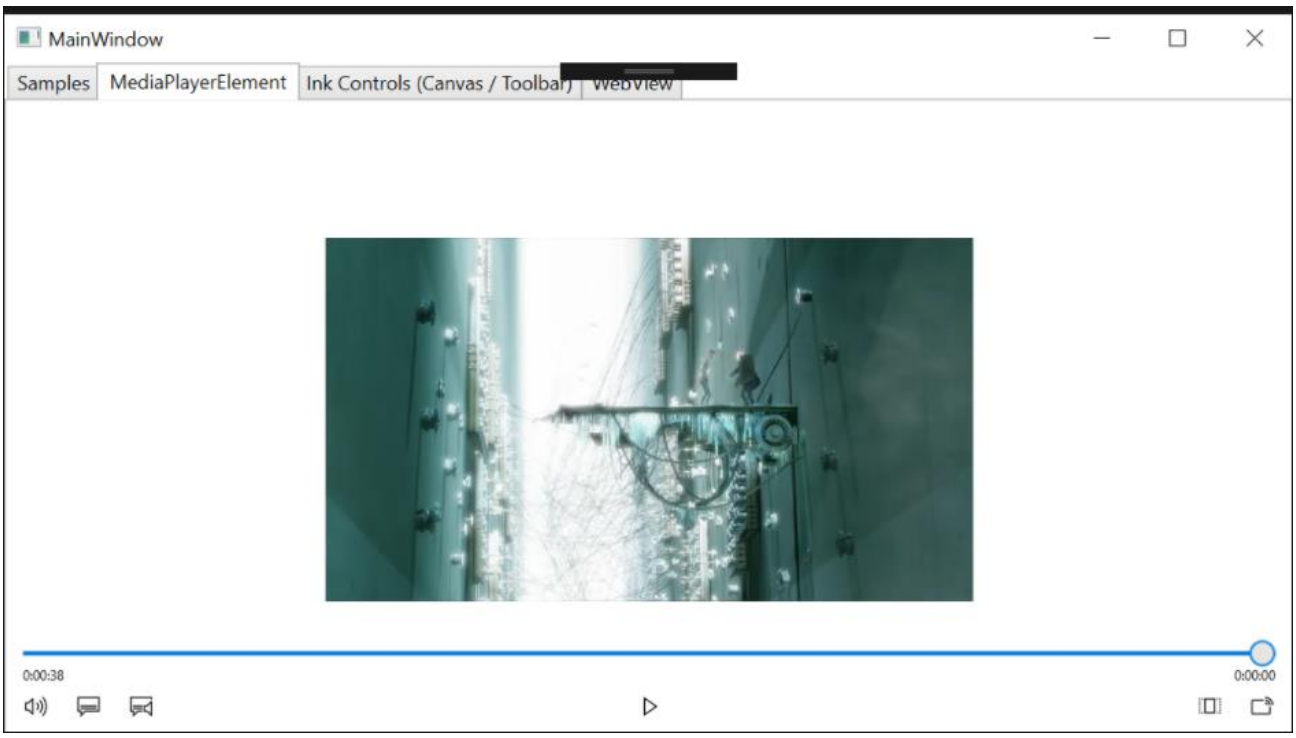


Рисунок 2.11 – Приклад використання MediaTransportClass

Наступні властивості обертають відповідні властивості загорнутого об'єкта UWP Windows.UI.Xaml.Controls.MediaPlayerElement та покращують їх:

1. AreTransportControlsEnabled типу bool.
2. AreTransportControlsEnabledProperty типу DependencyProperty.
3. AutoPlay типу bool.
4. AutoPlayProperty типу DependencyProperty.
5. IsFullWindow типу bool.
6. MediaPlayer типу MediaElement.
7. Source типу string.

Також, даний клас робить набагато легшим створення основних елементів керування медіаелементом, таких як:

1. Seek bar (повзунок керування).
2. Play/pause (програвати, пауза).
3. Fast forward (вперед на 5 секунд).
4. Fast rewind (запустити відео з початку).
5. Stop (зупинити відео).

6. Volume (керування гучності).
7. Full screen (на весь екран).
8. Auto Track Selection (автоматичне перемикання меідафайлів)

Більшість з цього буде використано у створенні додатку, але, деякі потрібно буде реалізовувати самостійно, оскільки, функціональність доволі обмежена.

2.3 GIT

Що таке "контроль версій"? Контроль версій - це система, яка фіксує зміни у файлі чи наборі файлів з часом, щоб була можливість згадувати певні версії пізніше. Якщо ви розробник хочете зберегти кожну версію вашої програми, то система котролю версій - дуже розумна річ для використання. Це дозволяє повернути вибрані файли назад у попередній стан, повернути весь проект назад у попередній стан, порівняти зміни з часом, побачити, хто востаннє змінив щось, що може спричинити проблему, хто вніс проблему та коли та багато іншого.

СКВ дає змогу працювати над одним проектом одразу декільком розробникам без шкоди один одному. Існують три типи СКВ: локальна, централізована та розподілена. Далі розглянемо більш детально розподілену систему контролю версій.

Розподілена система контролю версій – у даній системі, користувачі повністю копіюють репозиторій. Тобто, у кожного розробника наявна копія всього початкового коду та змін, котрі були в нього внесені під час розробки проекту.

Це означає, якщо один із серверів вийде з ладу, будь-який інший клієнтський репозиторій може бути скопійований на інший сервер для продовження роботи, без втрати даних. Схема розподільної системи контролю версій зображено на рисунку 2.12

Надалі, робота буде проходити з віддаленим репозиторієм, користувач, коли захоче, зможе скопіювати собі робочу версію додтаку, та, якщо буде проблема з сервером, він нічого не втратить.

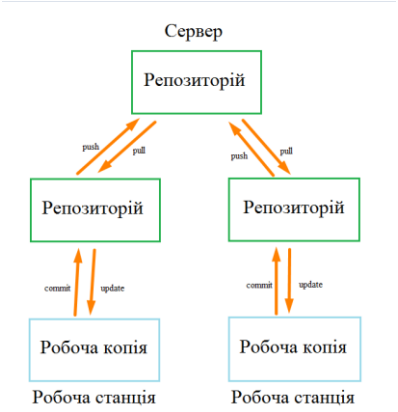


Рисунок 2.12 – Схема розподільної системи контролю версій

Git – це розподілена система контролю версій, система дає можливість розробникам відстежувати зміни в файлах та сумісно працювати з іншими розробниками . Найбільш відомі особливості Git – це простий дизайн, швидкодія, підтримка нелінійної розробки, повна децентралізація та, за необхідності, можливість роботи з великими проектами.

GitHub – сервіс онлайн-хостинга репозиторіїв, котрий вміщує в собі всі ті функції, що підтримує Git - функції розподільного контролю версій і функціональність управління початковим кодом. Використання GitHub разом з Git дає розробникам можливість збереження коду онлайн та зручно взаємодіяти з іншими розробниками. Важлива та зручна в розробці функція – це історія комітів, завдяки чому просто відстежувати які саме зміни було внесено. В історії комітів виділено файли та частини коду, в котрі було внесено зміни (рис. 2.13). Можливість порівняти частини коду спрощує пошук можливих помилок та відладку коду.

Все в Git має контрольну суму перед тим, як зберігатись, а потім посилається на цю контрольну суму. Це означає, що неможливо змінити вміст будь-якого файлу чи каталогу, щоб git не взнав про це. Ця функціональність вбудована на найнижчих рівнях і є невід'ємною частиною. Ви не можете втратити інформацію або отримати пошкодження файлів. Механізм, який Git використовує для цього контрольного підсумовування, називається хешем SHA-1. Це 40-символьний рядок, що складається з шістнадцяткових символів (0–9 та a – f) і обчислюється на основі вмісту файлу або структури каталогів у Git.

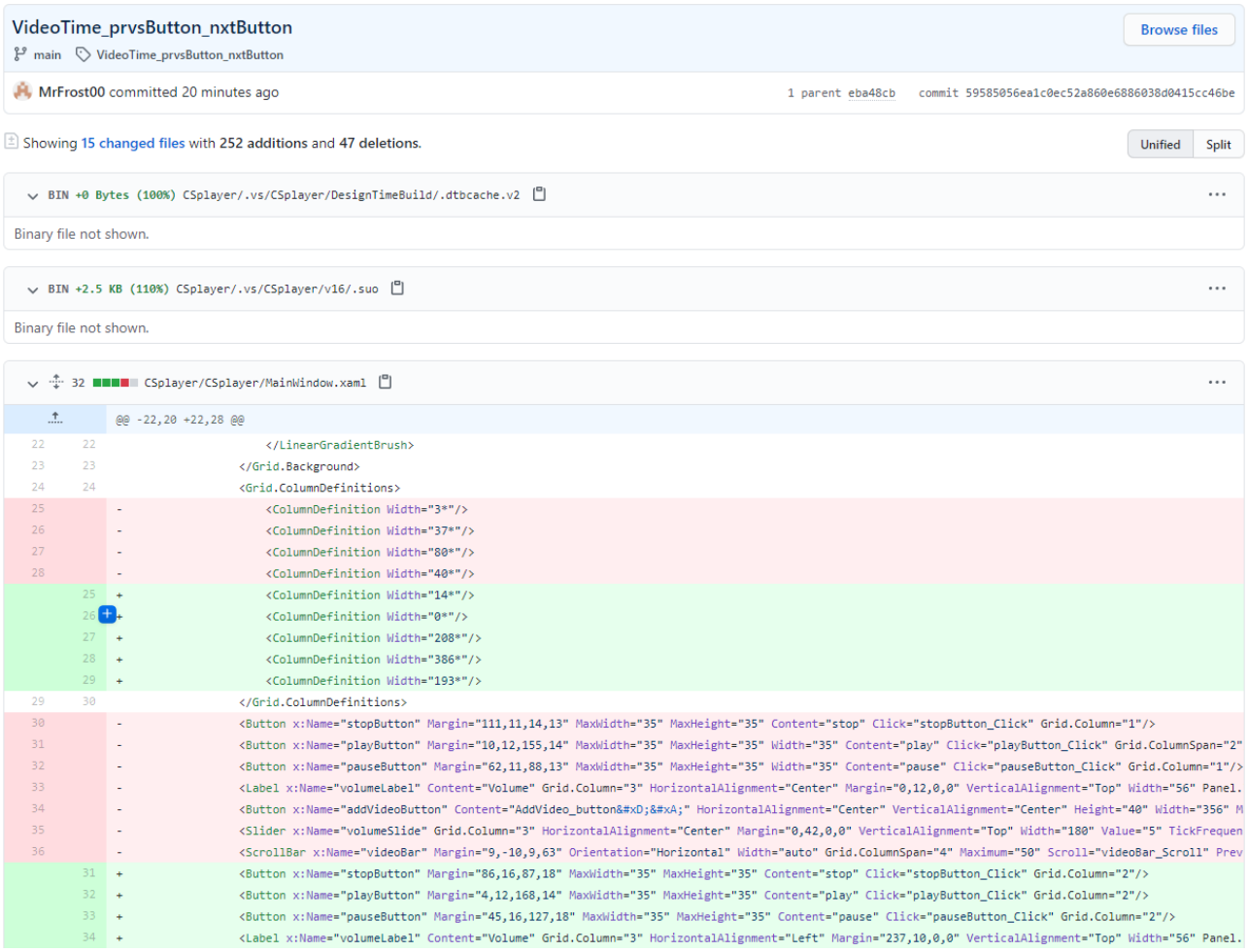


Рисунок 2.13 – Вигляд коміту на сервісі GitHub

Отже, що таке Git у двох словах? Основною відмінністю між Git та будь-якою іншою системою контролю версій є те, як Git думає про свої дані. Концептуально більшість інших систем зберігають інформацію як перелік файлових змін. Ці інші системи (CVS, Subversion, Perforce, Bazaar тощо) сприймають інформацію, яку вони зберігають, як набір файлів та зміни, внесені до кожного файлу з часом (це зазвичай називають контролем версій на основі дельти). Git не думає і не зберігає свої дані таким чином. Натомість Git думає про свої дані більше як про серію знімків мініатюрної файлової системи. Кожного разу, коли ви фіксуєте або зберігаєте стан свого проекту, Git в основному робить фотографію того, як виглядають всі ваші файли на той момент, і зберігає посилання на цей знімок. Щоб ефективніше, якщо файли не змінилися, Git не зберігає файл знову, лише посилання на попередній ідентичний файл, який він уже зберігав. Git думає про свої дані

більше як потік знімків.

Це важлива відмінність між Git та майже всіма іншими системами контролю версій. Це робить Git більш схожим на міні-файлову систему з неймовірно потужними інструментами, побудованими поверх неї, а не просто на системою контролю версій.

Це надає додатковий захист користувачів гіт і робить його унікальним серед інших систем контролю версій, оскільки, при роботі декількох людей з одним розробником, дуже важко втратити дані, для кожного, оскільки у кожного є свій локальний репозіторій, тобто, версія програми, а контрольна сума відстежує, щоб дані файли не були втрачені або пошкоджені.

Все це робить гіт найкращим вибором для супроводження даного додатку.

2.4 SourceTree

Хоч гіт і дуже зручний в плані супроводження проекту, у нього є свої недоліки, один з них – постійні десятки консольних команд, які забирають багато часу та не завжди коректно працюють, оскільки, якщо була допущена помилка – взнати де вона і чому вона виникла – це доволі складна задача.

На допомогу у вирішені цих проблем приходить SourceTree.

SourceTree було створено для того, щоб зробити Git доступним для кожного розробника - особливо тих, хто новачок у Git. Кожна команда Git - це лише один клік за допомогою інтерфейсу SourceTree. Даний додаток спрощує наступний функціонал:

1. Створення та клонування репозиторіїв з будь-якого місця.
2. Commit, push, pull та merge в декілька кліків.
3. Розвинена система пошуку помилок та конфліктів.
4. Відстежування всього життєвогу циклу проекту.

Як вже зазначалось, супроводження реалізації додатку буде проводитись за допомогою GitHub. Використання SourceTree спрощує роботу з Гіт до декількох кліків рис.2.14.

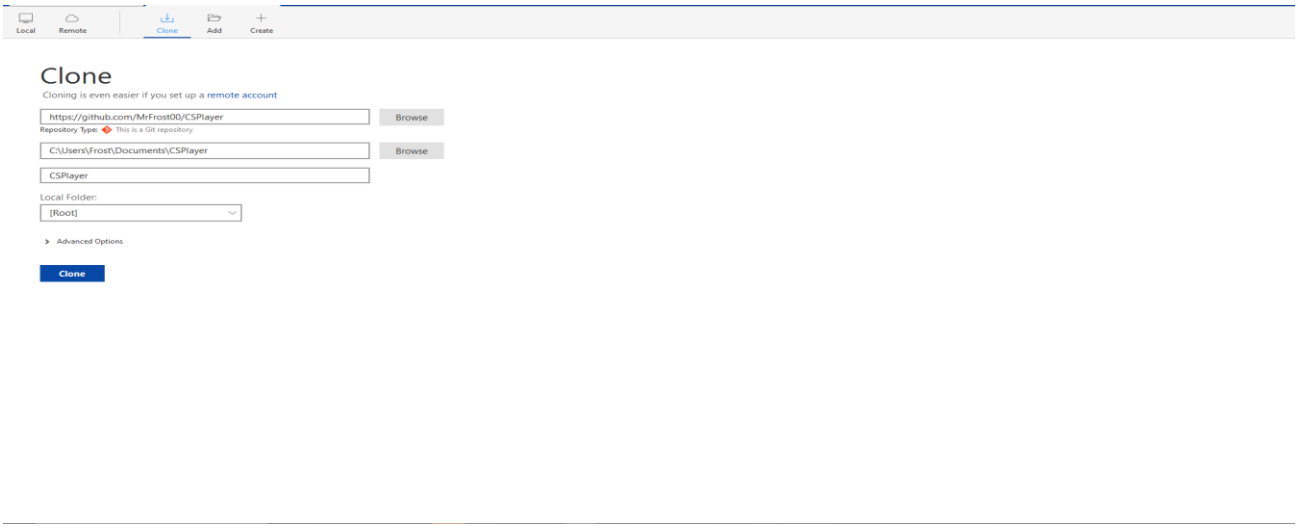


Рисунок 2.14 – Клонування\створення репозиторію

Одним з основого функціоналу гіта є – комміт внесених у проект змін, при звичайному використанні прийшлось би писати декілька консольних команд, у SourceTree все зводиться до 2 кліків рис.2.15.

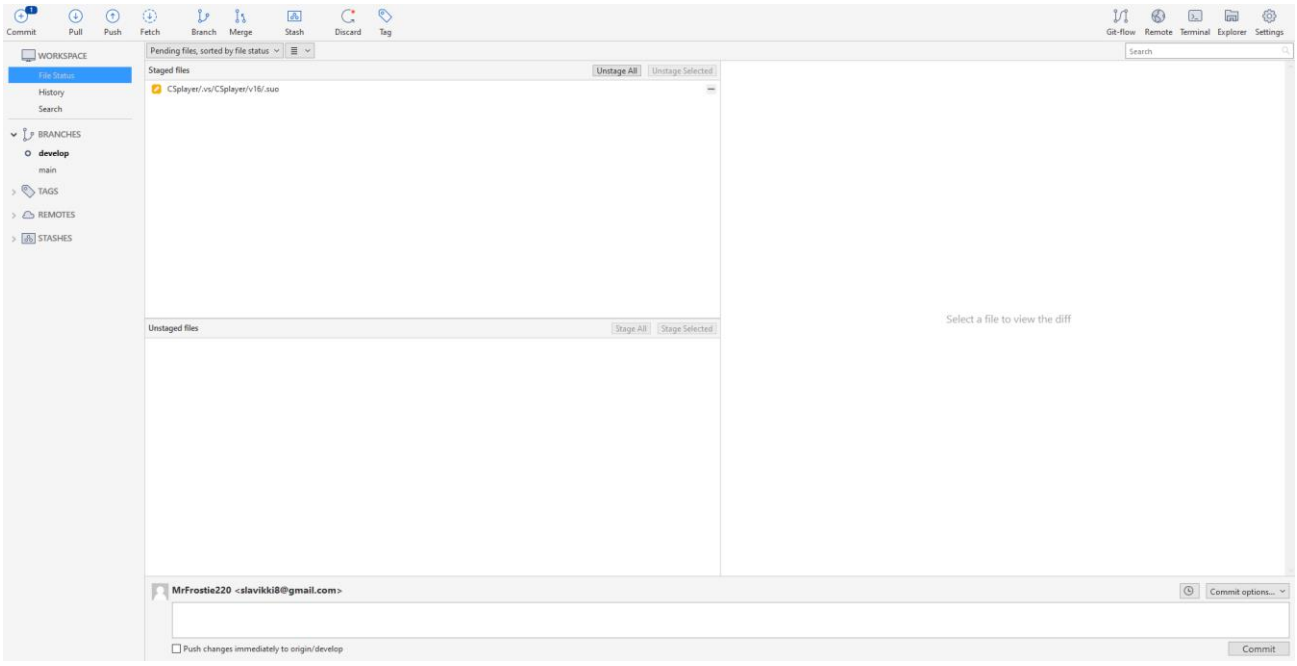


Рисунок 2.15 – Комміт внесених змін

Також даний додаток спрощує систему гілок гіта, оскільки, зазвичай, використовують 2 гілки – master та develop, виникає необхідність їх зливання одна з одною. При опису цього процесу консольними команда – зазвичай виникають

труднощі. SourceTree – автоматизував цей процес рис.2.16.

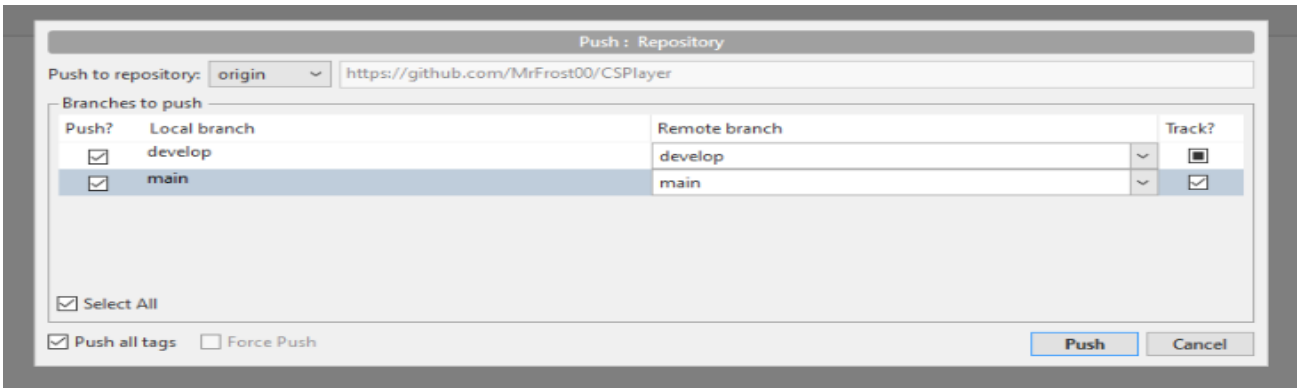


Рисунок 2.16 – Пуш та автоматичне зливання гілок

Також реалізована зручна візуалізація всього життєвого циклу процесу, яку можна відстежувати крок за кроком. В один клік можна побачити всі зміни внесені на потрібному пуші, та в один клік можна відкатити проект до потрібної версії рис.2.17.

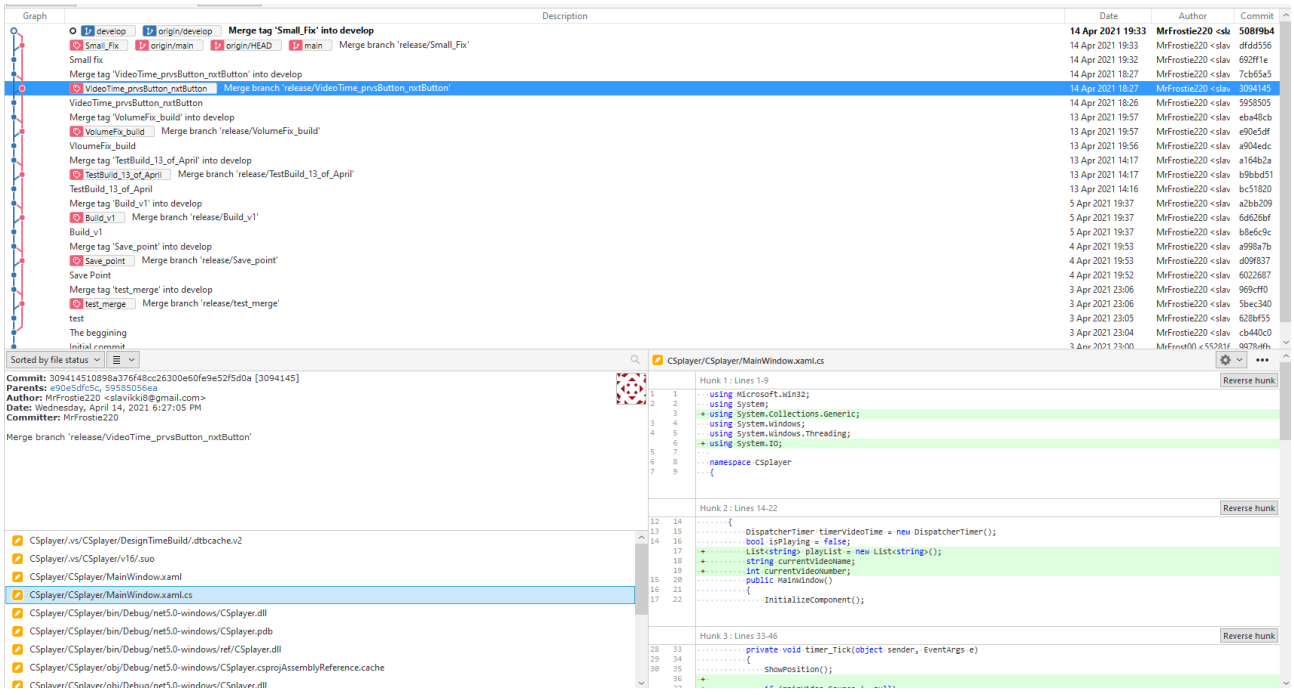


Рисунок 2.17 – Візуалізація життєвого циклу проекту

2.5 Adobe Photoshop

Adobe Photoshop - це програма для редагування зображень та ретушування фотографій для використання на комп'ютерах з ОС Windows або MacOS.

Photoshop пропонує користувачам можливість створювати, вдосконалювати чи іншим чином редагувати зображення та ілюстрації. Зміна фонів, імітація картини з реального життя або створення альтернативного уявлення про Всесвіт - все це можливо в Adobe Photoshop. Це найбільш широко використовуваний програмний інструмент для редагування фотографій, маніпуляцій із зображеннями та ретуші для численних форматів файлів зображень та відео.

Інструменти в Photoshop дають можливість редагувати як окремі зображення, так і великі партії фотографій. Існує кілька версій Photoshop, включаючи Photoshop CC, Photoshop Elements, Photoshop Lightroom і Photoshop Express, версію Photoshop для iOS зі зменшеними функціями.

Це найкращий вибір для створення сучасного дизайну окремих елементів додатку.

Adobe Photoshop CC - це версія Creative Cloud Photoshop, доступна за передплатою. Вважається версією продуктів професійного рівня сімейства Photoshop. Photoshop CC доступний разом із Photoshop Lightroom або як частина більшої передплати на Creative Cloud.

Photoshop CC - це вдосконалене програмне забезпечення для обробки зображень, яке використовується дизайнерами, веб-професіоналами, відеоредакторами та фотографами для зміни чи маніпулювання цифровими зображеннями. Photoshop в основному використовується для редагування 2D-зображень, хоча він пропонує деякі функції редагування 3D-зображень. Photoshop включає функцію аналізу зображень і може бути використаний для підготовки зображень до використання в Інтернеті або в друкованому вигляді.

Adobe Photoshop Elements - версія споживчого рівня сімейства продуктів

Photoshop. Photoshop Elements містить багато професійних можливостей, які можна знайти в Adobe Photoshop CC, однак вони мають більш спрощені опції, розроблені з урахуванням користувача початкового рівня. Більш конкретно, він призначений для фотографів-любителів та любителів цифрової фотографії. Photoshop Elements побудований з використанням тієї самої основної технології цифрових зображень, що і Photoshop CC. До загальновживаних можливостей Photoshop Elements належать:

1. Маніпулювання кольором зображення.
2. Обрізання зображень.
3. Виправлення недоліків, таких як пил на об'єктиві або червоні очі.
4. Малювання на зображенні пером або олівцем.
5. Додавання тексту до зображень.
6. Видалення людей або предметів із зображення.
7. Впорядкування фотографій для швидкого доступу.
8. Публікація зображень в Інтернеті або надсилання електронною поштою.

Щоб використовувати Photoshop на комп’ютері з ОС Windows, комп’ютер повинен відповідати цим вимогам:

1. Процесор Intel® Core 2 або AMD Athlon® 64.
2. Процесор 2 ГГц або більше.
3. Microsoft Windows 7 із пакетом оновлень 1, Windows 8.1 або Windows 10.
4. 2 ГБ або більше оперативної пам'яті. 2,6 ГБ або більше вільного місця на жорсткому диску для 32-розрядної установки.
5. 3,1 ГБ або більше вільного місця на жорсткому диску для 64-розрядної установки; а також додатковий вільний простір, необхідний для установки.
6. Дисплей 1024 x 768 із 16-бітовим кольором та виділеною VR-пам'яттю 512 МБ або більше.
7. Система з підтримкою OpenGL 2.0. Підключення до Інтернету потрібно для активації програмного забезпечення, перевірки підписки та доступу до різних Інтернет-служб.

3.ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

3.1 Розробка додатку

Завдання, котре повинен вирішувати застосунок – це надати можливість користувачеві, переглядати відеофайли самих поширених форматів.

Оскільки, даний додаток націлений на аудиторію з базовими знаннями користування комп’ютером - важливо розробити простий та інтуїтивно зрозумілий інтерфейс для зручного користування.

Основні можливості даного програмного забезпечення: можливість перегляду відеофайлів, можливість створення плейлістів, можливість маніпулювати відеофайлом.

Перш за все потрібно завантажити Microsoft visual studio. Завантажити її можна з офіційного сайта Microsoft рис.3.1, але важливо обрати версію Community, оскільки вона безкоштовна та націлена, здебільш, на студентів.

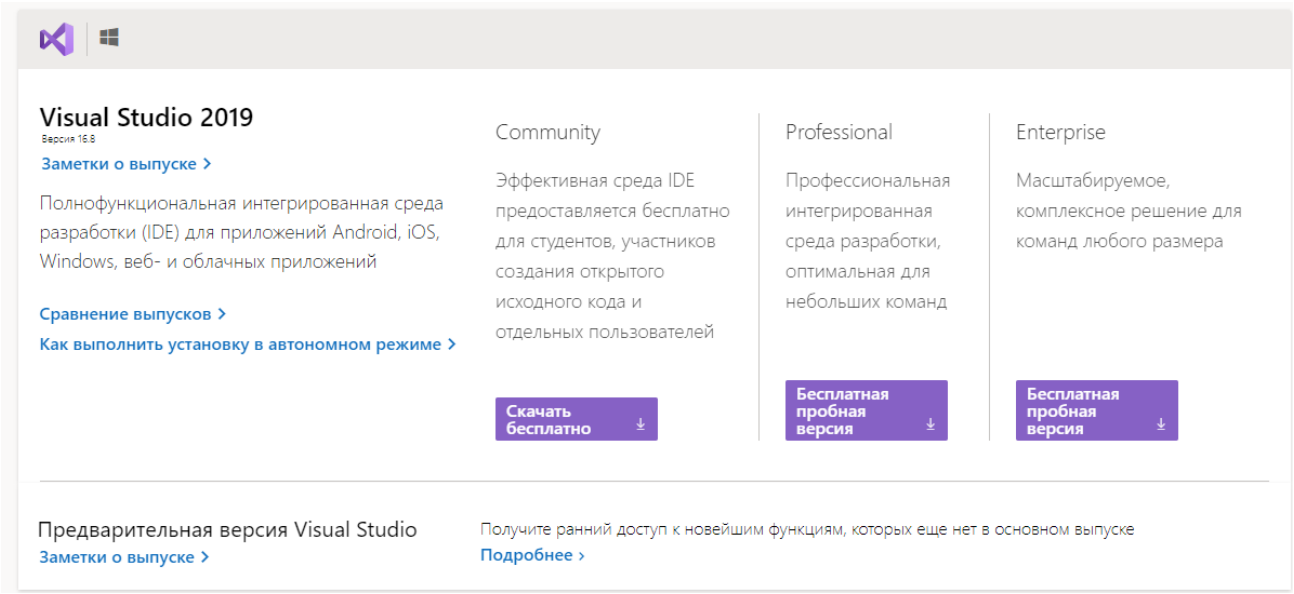


Рисунок 3.1 – Завантаження Visual Studio 2019

Надалі я буду використовувати Visual Studio 2019 Community.

Надалі потрібно створити наш проект, використовуючи шаблон WPF, який було описано раніше. В меню File (Файл) обираємо New (Створити) > Project (Проект). Перед нами відкриється діалогове вікно рис.3.2 створення проекту, в

якому ми обираємо шаблон WPF Application.

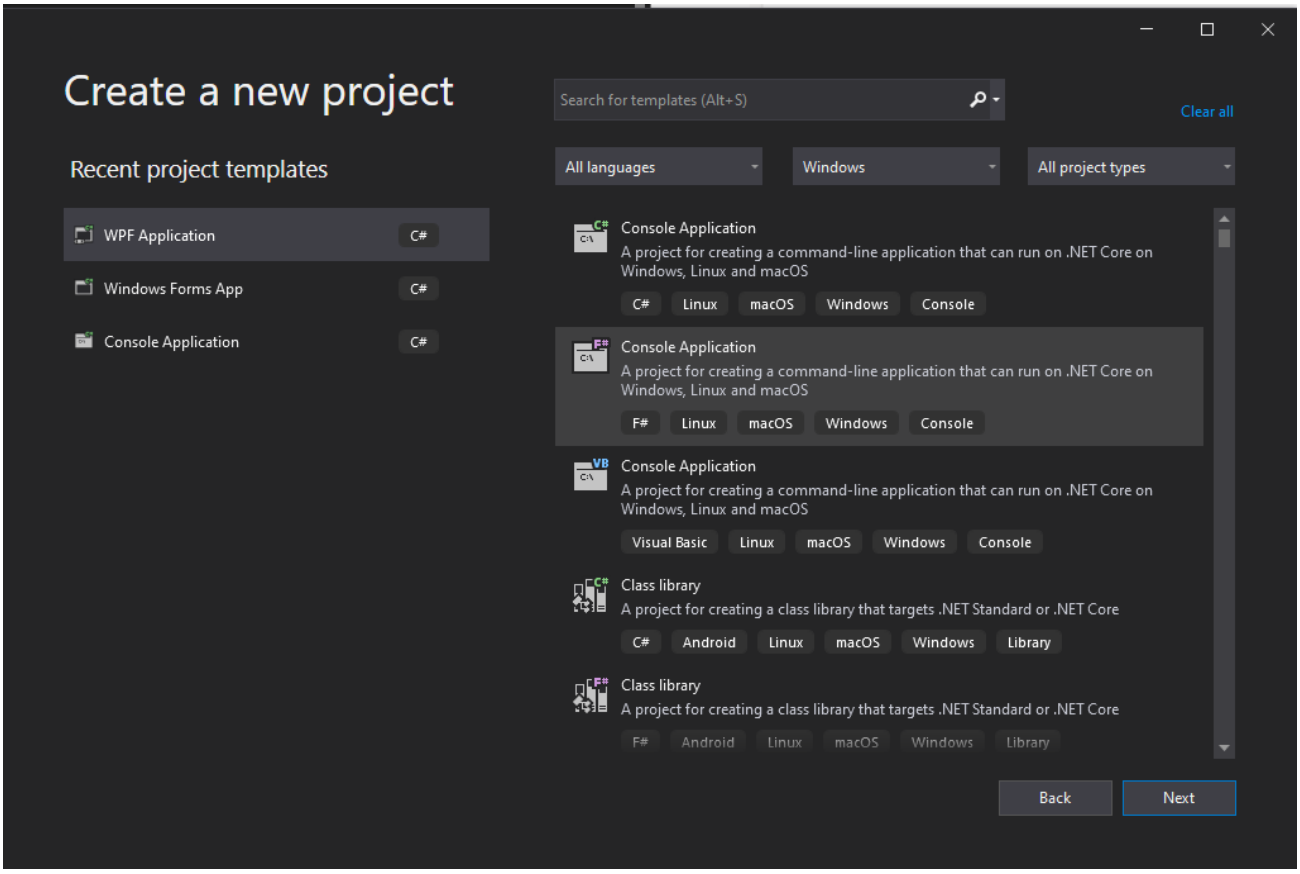


Рисунок 3.2 – Створення проекту

Після встановлення назви, студія автоматично створить на новий проект рис.3.3. В структурі проекту WPF слід виділити, що MainWindow.xaml та MainWindow.xaml.cs 2 основні файли розробки нашого проекту, коли буде запускатися зборка всього додатку – він буде створюватися з реалізованого інтерфейсу у файлі MainWindow.xaml та підкорятись функціоналу реалізованому у MainWindow.xaml.cs

Також, важливою частиною проекту є References. Там, ми маємо можливість завантажити додатокві бібліотеки, для реалзіації нашого додатку.

Наприклад, раніше описаний класс MediaTransportClass додається з зовнішніх бібліотек, в яких описано весь функціонал, для реалізації WindowsMediaPlayer, частину якого, можна буде використати, для створення додатку.

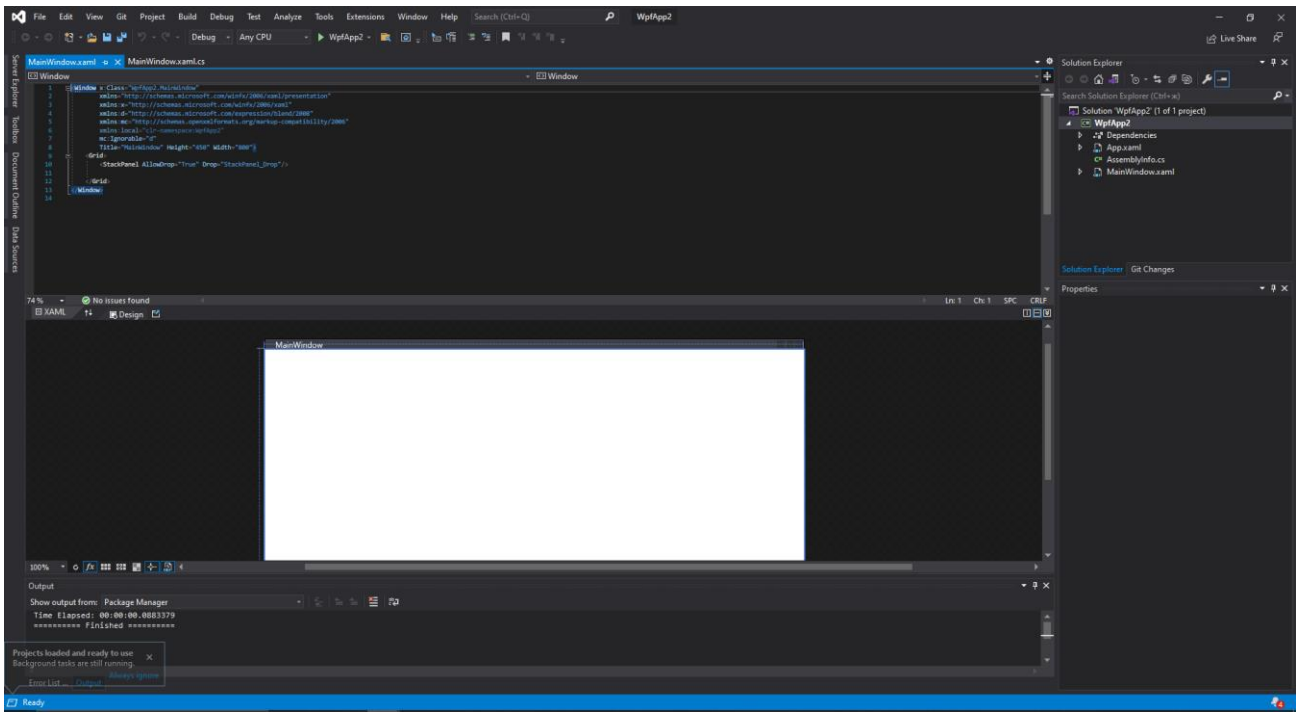


Рисунок 3.3 – Проект на основі шаблону WPF

З цього етапу і починається реальна розробка.

Перш за все, необхідно створити початковий інтерфейс, за допомогою конструктора або напряму використовуючи XAML рис.3.4.

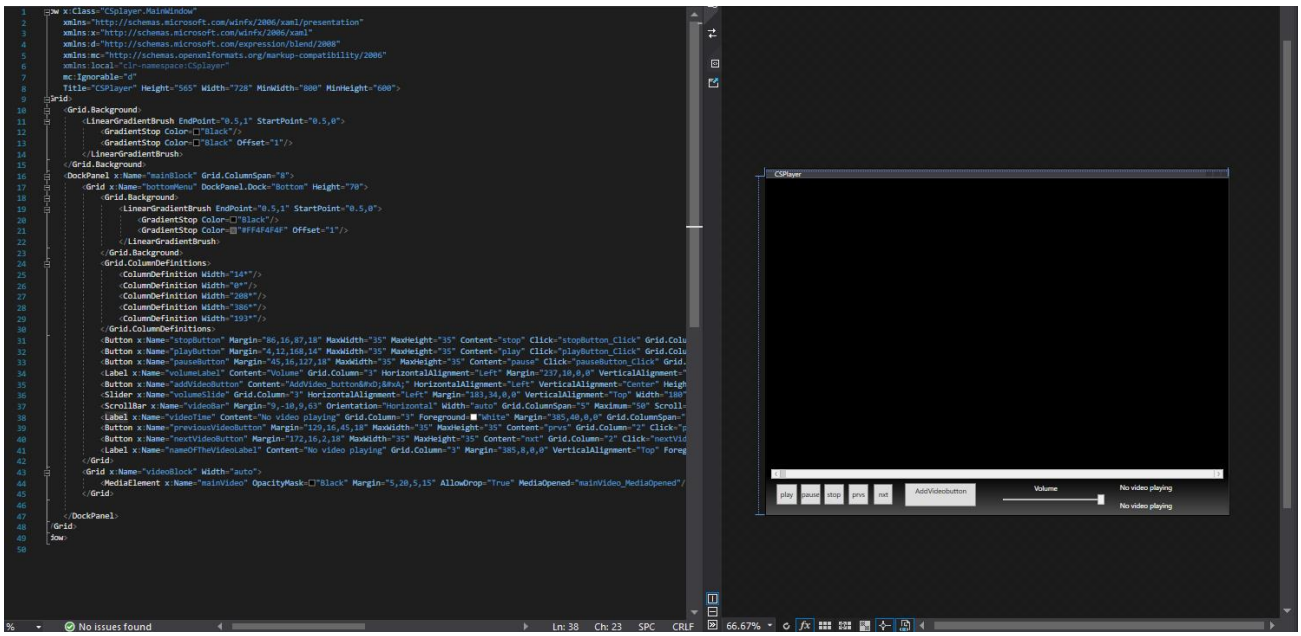


Рисунок 3.4 – Початковий інтерфейс додатку

Основний інтерефейс додатку буде включати в себе наступні об’єкти:

1. Основне вікно для завантаження\перегляду відеофайлів.
2. Повзунок для регулювання гучності відеофайлу.
3. Повзунок для контролю відеофайлу.
4. Текстове поле для відображення назви відеофайлу.
5. Текстове поле для відображення загального та поточного часу відеофайлу.
6. Кнопку Play (Програвати).
7. Кнопку Pause (Пауза).
8. Кнопку Stop (Зупинити).
9. Кнопку Next (Наступне відео з плейліста).
10. Кнопку Previous (Минуле відео з плейліста).
11. Кнопку AddVideo (Завантаження одного відео чи плейліста).

Для створення основного вікна для перегляду відеофайлів – будемо використовувати елемент WPF MediaElement, який дозволяє відтворювати аудіо\відеофайли усіх популярних форматів. Також MediElement надає простий функціонал для маніпуляції з відео, доступні наступні функції без необхідності їх реалізації рис.3.5:

```
private void playButton_Click(object sender, RoutedEventArgs e)
{
    CheckVideoState();
}

1 reference
private void pauseButton_Click(object sender, RoutedEventArgs e)
{
    CheckVideoState();
}

1 reference
private void stopButton_Click(object sender, RoutedEventArgs e)
{
    if (isPlaying == true)
    {
        mainVideo.Stop();
        isPlaying = false;
        mainVideo.Position = default;
    }
```

Рисунок 3.5 – Реалізація кнопок Play, Pause, Stop

1. mediaElement.play – (програє файл з поточної секунди)
2. mediaElement.pause - (зупиняє програвання файлу)
3. mediaElemnt.stop – (зупиняє програвання файлу – наступний його запуск

буде з смаого початку)

Наступним кроком, щоб полегшити собі роботу в майбутньому була реалізована можливість завантаження відеофайлу в додаток.

На кнопку AddVideo був підписаний метод, який при натисканні викликав діалогове вікно вибору з сортування по аудіоформатам. При виборі потрібного відеофайла – діалогове вікно закривалось, та файл автоматично починав програватись рис.3.6.

```
private void addVideoButton_Click(object sender, RoutedEventArgs e)
{
    OpenFileDialog ofg = new OpenFileDialog()
    {
        Filter = "(mp3,wav,mp4,mov,wmv,mpg)|*.mp3;*.wav;*.mp4;*.mov;*.wmv;*.mpg;*.avi|all files|*.*",
        InitialDirectory = Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments)
    };
    ofg.Multiselect = true;
    if (ofg.ShowDialog() == true)
    {
        foreach (string item in ofg.FileNames)
            {
                playList.Add(item);
            }
        VideoLoad(playList[0], 0);
    }
```

Рисунок 3.6 – Реалізація методу завантаження віедофайлу

Також, одразу був реалізований, так званний, MultiSelect, який дозволить користувачу створювати свої плейлісти.

Реалізувати повзунок маніпуляції віедофайлом буде більш важкою задачою, оскілки ніякого функціоналу на цей рахунок MediaElement не підтримує. Для того щоб створити “живий” повзунок, який буде відстежувати час відеофайлу в реальному часі, я створив таймер DispatcherTimer та підписав на його івент Tick метод timer_Tick рис. 3.7. DispatcherTimer - це таймер, інтегрований у чергу диспетчера, який обробляється через певний проміжок часу та з певним пріоритетом. Основна ідея в тому, що при завантажені будь-якого відеофайлу, автоматично буде запускатись таймер, який, на кожен свій Tick (тік) буде змінювати позицію повзунка, залежно від пройденого часу відео. Максимальне число повзунка буде рівне довжині завантаженого файлу.

```
public void ShowPosition()
{
    videoBar.Value = mainVideo.Position.TotalSeconds;
    //txtPosition.Text =
    //    minionPlayer.Position.TotalSeconds.ToString("0.0");
}
```

Рисунок 3.7 – Реалізація контролю відеофайлу

Важлива частина будь-якого медіаплеєра – можливість регулювання гучності медіфайлу.

Для реалізація даного функціоналу у своєму додатку я використав Slider.

Slider представляють елементи, засновані на діапазонах значень. Тобто вони зберігають і відображають числові дані на певному діапазоні. Він є наслідником класа RangeBase, того наслідує від нього такі параметри, як **Value** та **Minimum\Maximum**.

У класі Slider описані наступні властивості:

1. Orientation - вказує орієнтацію повзунка - горизонтальну (Horizontal) або вертикальну (Vertical)
2. Delay - вказує час в мілісекундах, після закінчення якого повзунок переміститься на одну одиницю після клацання.
3. Interval: вказує час в мілісекундах, після закінчення якого повзунок може переміщатися
4. TickPlacement: задає візуалізацію шкали повзунка. За замовчуванням має значення None (відсутність шкали). Значення BottomRight створюють шкалу в нижній частині повзунка, TopLeft - у верхній, Both - по обом сторонам.
5. TickFrequency: вказує частоту появи відміток на шкалі повзунк.
6. IsSelectionRangeEnabled: задає затінення ділянки повзунка. Якщо воно встановлено в True, то початкова і кінцева відмітка затінення задаються за допомогою властивостей SelectionStart і SelectionEnd.

Все що потрібно для реалізації регулятора гучності – це при виклику івенту

ValueChanged (зміна значення – якщо користувач посуне повзунок) прирівнювати значення гучності відеофайлу до значення повзунка рис.3.8.

```
private void volumeSlide_ValueChanged(object sender, RoutedPropertyChangedEventArgs<double> e)
{
    if (isPlaying)
        mainVideo.Volume = (double)volumeSlide.Value;
}
```

Рисунок 3.8 – Реалізація контролю гучності

Наступним етапом, для більш user-friendly інтерфейсу – потрібно реалізувати можливість відстежувати довжину та назву завантаженого файлу.

Перш за все реалізуємо назву, для цього буде використано текстовий елемент Label (мітка). Головною особливістю міток є підтримка мнемонічних команд-клавіш швидкого доступу, які передають фокус пов'язаного елементу.

При завантажені файлу, у глобальну змінну буде записуватись весь шлях до файлу. Далі, потрібно зчитати назву файлу зі всього шляху до нього. У цьому допоможе бібліотек System.IO.

System.IO - містить типи, що дозволяють читати та записувати у файли та потоки даних, а також типи, що забезпечують базову підтримку файлів та каталогів. Використовуючи метод GetFileName – ми зможемо отримати назву файла з шляху до нього рис.3.9, рис.3.10.

```
public void SetVideoName()
{
    nameOfTheVideoLabel.Content = Path.GetFileName(currentVideoName);
}
```

Рисунок 3.9 – Отримання назви завантаженого файлу

Одразу протестуємо даний код.

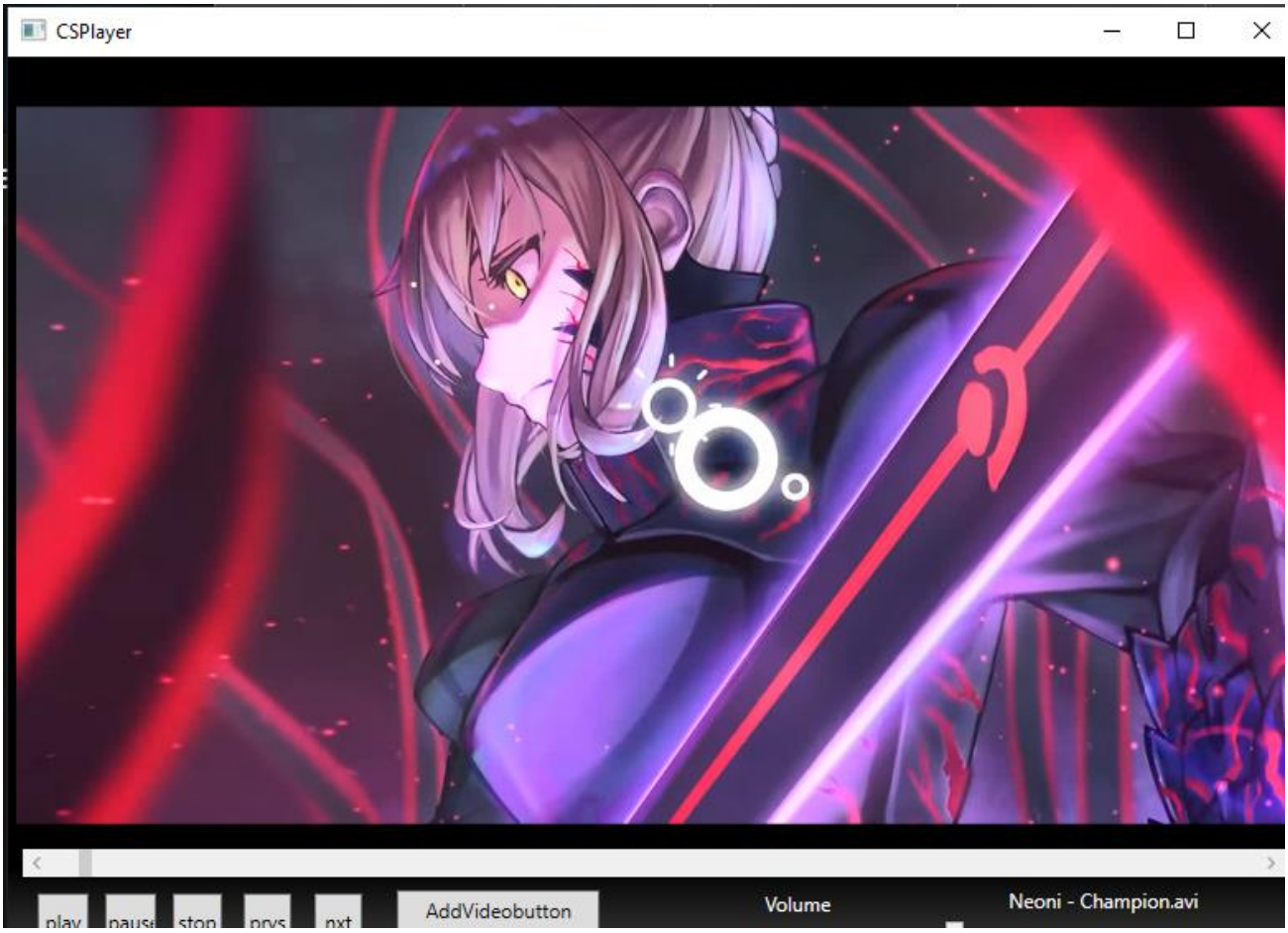


Рисунок 3.10 – Перевірка роботи

Наступним етапом – потрібно виводити загальну довжину файлу та поточний час програвання. Для цього також буде використаний текстовий елемент Label.

Для реалізації “живого” таймеру відстежування часу програвання відеофайлу, я буду використовувати вже раніше реалізований таймер, який був використаний для контролю позвунка відеофайлу. В метод timer_tick я додам ще один метод, який буде прив’язувати загальний час та поточний час до створеної раніше Label у стрінговому форматі “00:00,хв:сек” рис3.11, рис 3.12.

```
if (mainVideo.Source != null)
{
    if (mainVideo.NaturalDuration.HasTimeSpan)
        videoTime.Content = String.Format("{0} / {1}", mainVideo.Position.ToString(@"mm\:ss"), mainVideo.NaturalDuration.TimeSpan.ToString(@"mm\:ss"));
}
else
    videoTime.Content = "No file selected...";
```

Рисунок 3.11 – Реалізація “живого” таймеру.

Одразу протестуємо даний код.

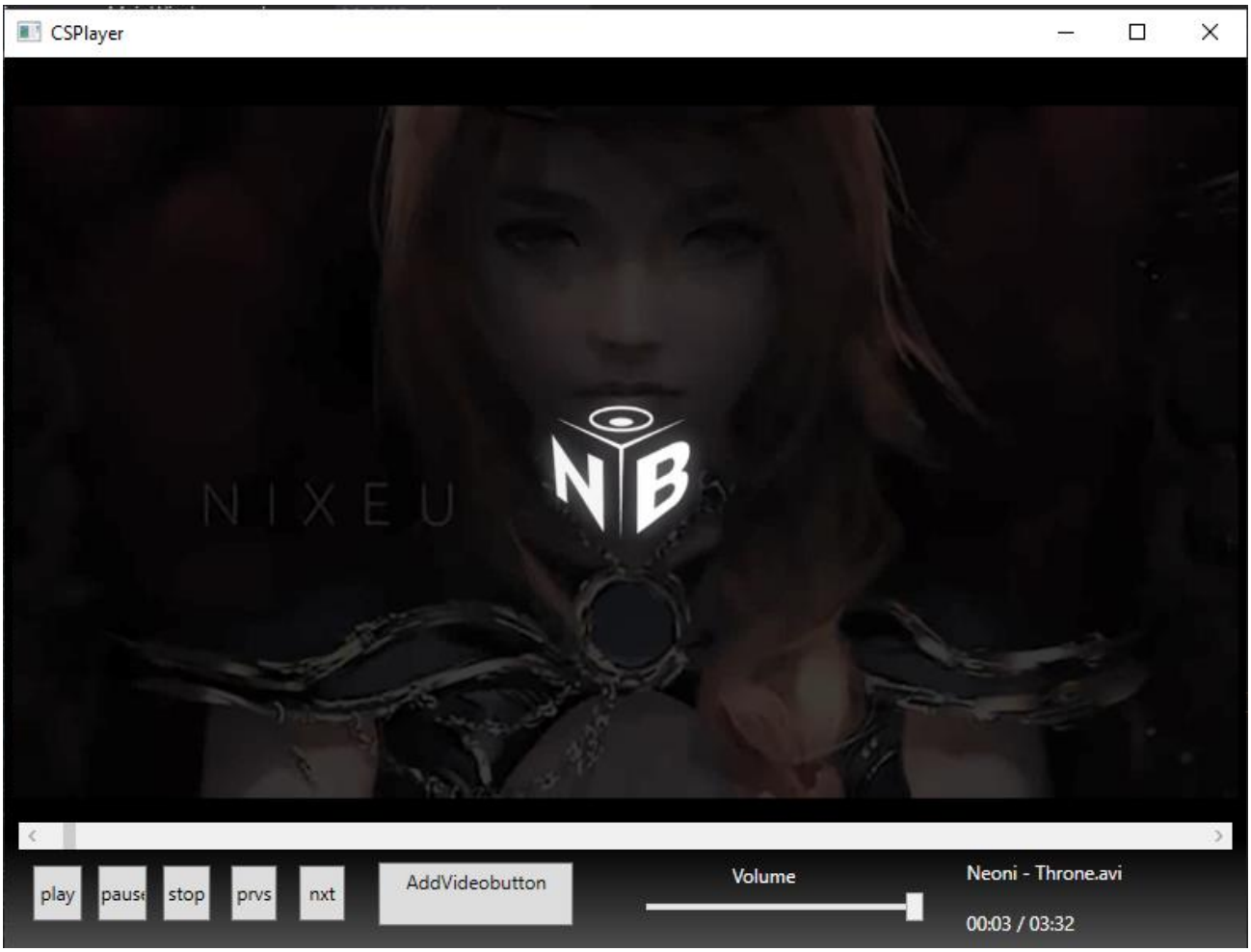


Рисунок 3.12 – Перевірка роботи.

Останнім етапом реалізації буде створення кнопок Previous (минуле відео), Next (наступне відео).

Перш за все, потрібно створити лист, який, при завантажені 2 або більше файлів – буде заповнюватись. Надалі, я буду використовувати цей лист, як плейлист, щоб була можливість переключати відео. Надалі треба створити методи для котнролю самих кнопок.

Для кнопки Next суть метода буде полягати в, перш за все, перевірці, чи відео грає, для цього створена глобальна зміна isPlaying типу bool, яка відстежує статус відео. Надалі, потрібно перевіряти, яке відео з плейлісту грає, та, якщо воно не останнє – переходити на наступний елемент листу, якщо відео останнє – запускати перший елемент плейлисту рис.3.13.

```
private void nextVideoButton_Click(object sender, RoutedEventArgs e)
{
    if (isPlaying)
    {
        if (currentVideoNumber + 1 != playList.Count)
        {
            ChangeVideo(playList[currentVideoNumber + 1], currentVideoNumber + 1);
        }
        else
        {
            ChangeVideo(playList[0], 0);
        }
        mainVideo.Play();
    }
}
```

Рисунок 3.13 – Реалізація функціоналу кнопки Next

Для кнопки Previous функціонал буде майже аналогічний, але буде перевірятись, якщо грає не перший елемент, то має завантажитись минуле відео з плейліста, якщо грає перший елемент плейліста, то має завантажитись останній елемент плейлісту рис.3.14.

Головною проблемою реалізації цих кнопок є те, що вони обидві мають перевіряти кількість елементів в плейлісті, якщо більше ніж 1, то вони повинні переключати відео, якщо 1, то відео має просто запускатися з самого початку.

```
private void previousVideoButton_Click(object sender, RoutedEventArgs e)
{
    if (isPlaying)
    {
        if (currentVideoNumber == 0)
        {
            ChangeVideo(playList[playList.Count-1], playList.Count-1);
        }
        else
        {
            ChangeVideo(playList[currentVideoNumber - 1], currentVideoNumber - 1);
        }
        mainVideo.Play();
    }
```

Рисунок 3.14 – Реалізація функціоналу кнопки Previous

3.2 Дизайн

Основний дизайн був створений за допомогою влаштованих можливостей WPF, використовуючи мову XAML.

Перш за все, був перероблений дизайн регулятора гучності, за допомогою XAML рис.3.15.

Рисунок 3.15 – Новий дизайн регулятора гучності

Надалі треба покращити дизайн регулятора відео рис 3.16. Також за допомогою XAML.

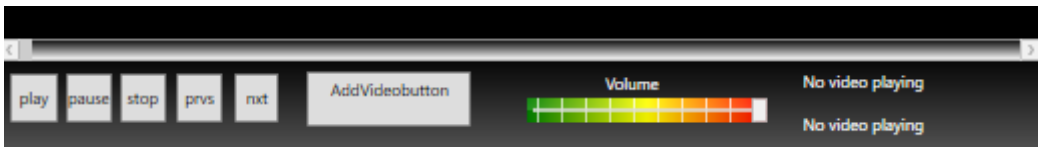


Рисунок 3.16 – Новий дизайн регулятора відео

Надалі потрібно змінити дизайн кнопок, на більш звичний для будь-якого користувача медіаплеєрів.

Дизайн кнопок краще зробити у фотошопі, оскільки, використовуючи XAML буде виконаний зайвий об’єм роботи, який можна спростити, використовуючи фотошоп рис.3.17.

Створення дизайну, за допомогою XAML – це повний опис всього дизайну в ручну, що не має сенсу, оскільки, фотошоп має весь встроєний функціонал, та велику кількість інстурментів для швидкої реалізації стильних та гарних елементів контролю.

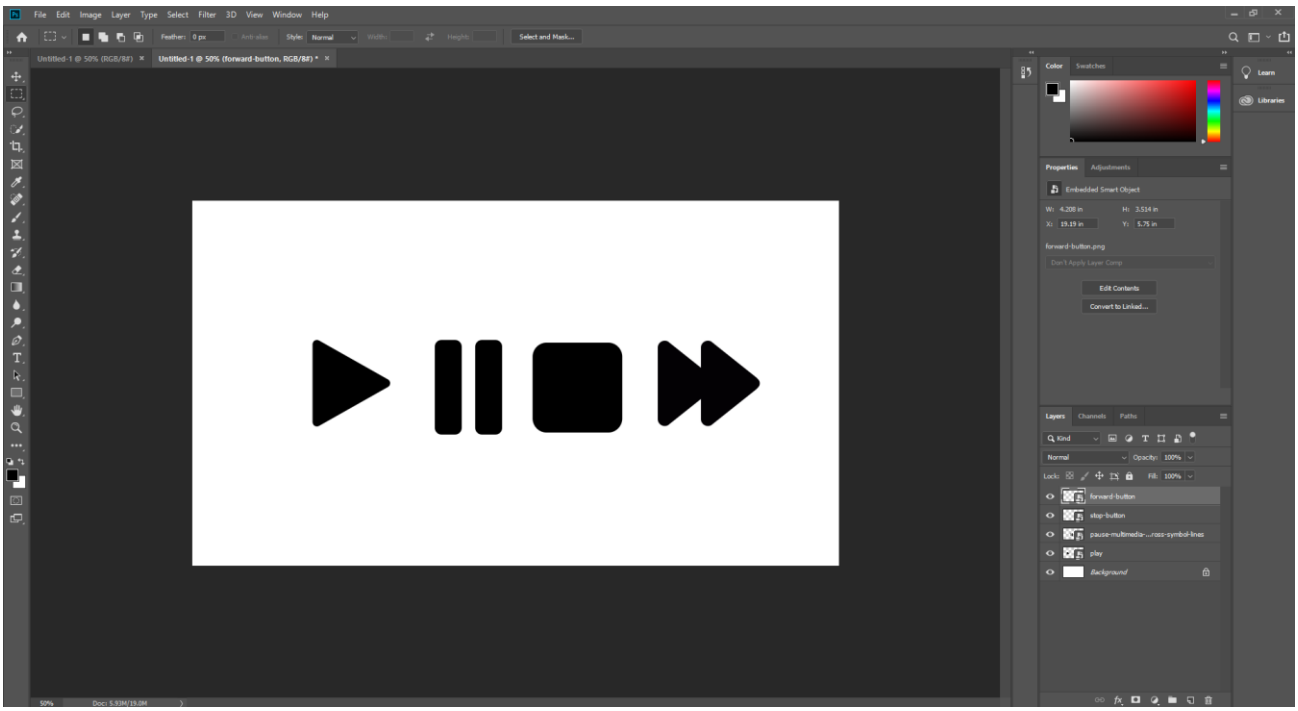


Рисунок 3.17 – Дизайн основних кнопок

Дані кнопки одразу можна включити в дизайн додатку, та перевірити рис.3.18.

Рисунок 3.18 Використання створених кнопок

Також, створюємо кнопку додавання відеофайлів рис.3.20.

Рисунок 3.19 – Редизайн кнопки додавання відеофайлів

Одразу використаємо дану кнопку у нашому додатку рис.3.19.

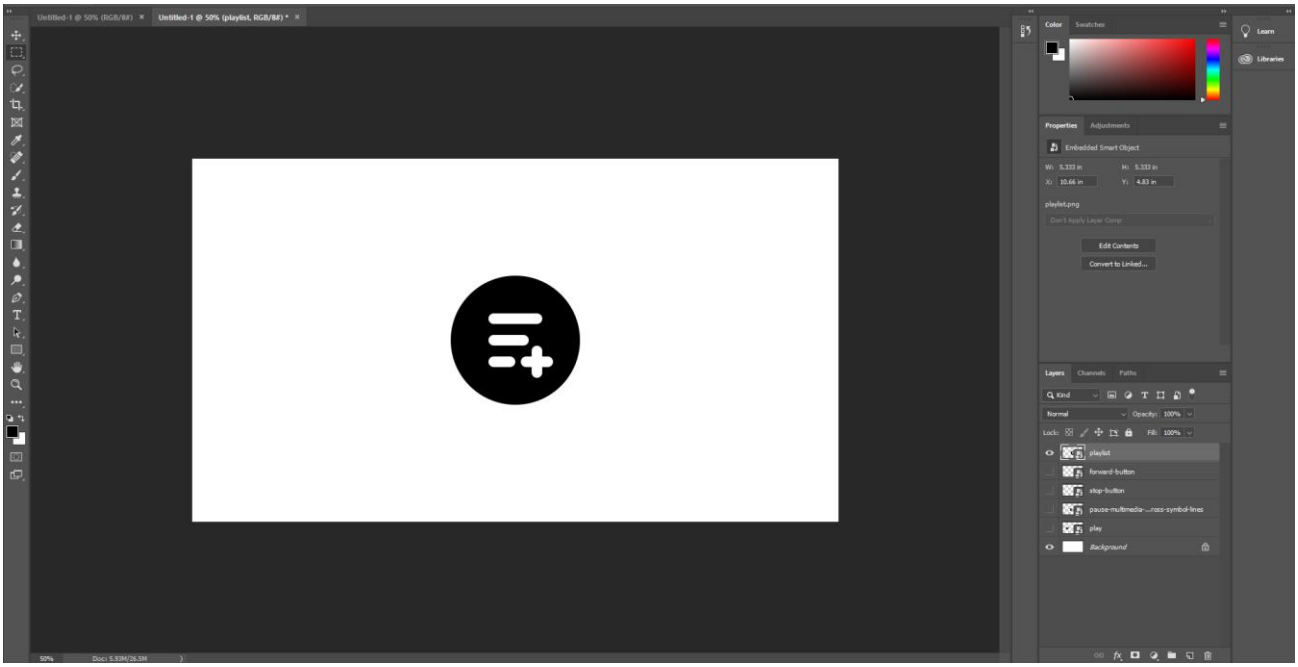


Рисунок 3.20 – Створення кнопки додавання відеофайлів

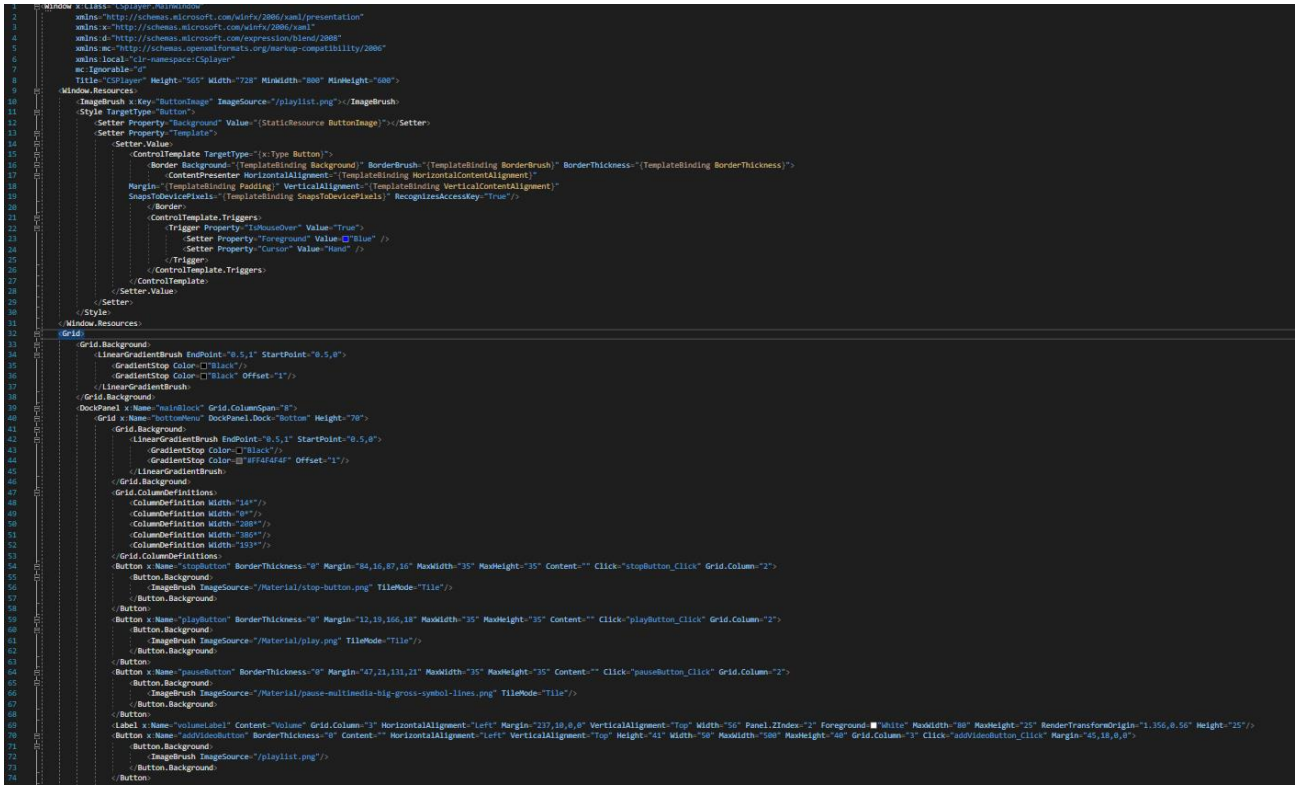
Також, треба пам’ятати, що кнопки, у WPF, за замовчуванням підтримують Hover ефект, тобто, коли користувач наведе мишкою на кнопку, картинку для якої, ми реалізували у фотошопі, вона зникне. Щоб уникнути цього, треба переопреділити параметри кнопки за замовчуванням рис.3.21.

```
<Window.Resources>
    <ImageBrush x:Key="ButtonImage" ImageSource="/playlist.png"></ImageBrush>
    <Style TargetType="Button">
        <Setter Property="Background" Value="{StaticResource ButtonImage}"></Setter>
        <Setter Property="Template">
            <Setter.Value>
                <ControlTemplate TargetType="{x:Type Button}">
                    <Border Background="{TemplateBinding Background}" BorderBrush="{TemplateBinding BorderBrush}" BorderThickness="{TemplateBinding BorderThickness}">
                        <ContentPresenter HorizontalAlignment="{TemplateBinding HorizontalContentAlignment}"
                    Margin="{TemplateBinding Padding}" VerticalAlignment="{TemplateBinding VerticalContentAlignment}"
                    SnapsToDevicePixels="{TemplateBinding SnapsToDevicePixels}" RecognizesAccessKey="True"/>
                    </Border>
                    <ControlTemplate.Triggers>
                        <Trigger Property="IsMouseOver" Value="True">
                            <Setter Property="Foreground" Value="Blue" />
                            <Setter Property="Cursor" Value="Hand" />
                        </Trigger>
                    </ControlTemplate.Triggers>
                </ControlTemplate>
            </Setter.Value>
        </Setter>
    </Style>
</Window.Resources>
```

Рисунок 3.21 – Перезапис параметрів кнопки за замовчуванням

Даний перезапис дозволить створювати більш адаптивний контроль над функціоналом, який реалізований у WPF за замовчуванням, що дозволить створювати додатковий функціонал у даному додатку, в майбутньому.

Весь інтерфейс додатку реалізований на мові XAML рис 3.22.



```
<Window x:Class="CSplayer.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:CSplayer"
        mc:Ignorable="d"
        Title="CSPlayer" Height="565" Width="728" MinWidth="800" MinHeight="600">
    <Window.Resources>
        <ImageBrush x:Key="ButtonImage" ImageSource="/playlist.png"></ImageBrush>
        <Style TargetType="Button">
            <Setter Property="Background" Value="{StaticResource ButtonImage}"></Setter>
            <Setter Property="Template">
                <Setter.Value>
                    <ControlTemplate TargetType="{x:Type Button}">
                        <Border Background="{TemplateBinding Background}" BorderBrush="{TemplateBinding BorderBrush}" BorderThickness="{TemplateBinding BorderThickness}">
                            <ContentPresenter HorizontalAlignment="{TemplateBinding HorizontalContentAlignment}"
                        Margin="{TemplateBinding Padding}" VerticalAlignment="{TemplateBinding VerticalContentAlignment}"
                        SnapsToDevicePixels="{TemplateBinding SnapsToDevicePixels}" RecognizesAccessKey="True"/>
                        </Border>
                        <ControlTemplate.Triggers>
                            <Trigger Property="IsMouseOver" Value="True">
                                <Setter Property="Foreground" Value="Blue" />
                                <Setter Property="Cursor" Value="Hand" />
                            </Trigger>
                        </ControlTemplate.Triggers>
                    </ControlTemplate>
                </Setter.Value>
            </Setter>
        </Style>
    </Window.Resources>
    <Grid>
        <Grid.Background>
            <LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">
                <GradientStop Color="Black"/>
                <GradientStop Color="Black" Offset="1"/>
            </LinearGradientBrush>
        </Grid.Background>
        <DockPanel x:Name="mainBlock" Grid.ColumnSpan="8">
            <Grid x:Name="bottomMenu" DockPanel.Dock="Bottom" Height="70">
                <Grid.Background>
                    <LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">
                        <GradientStop Color="Black"/>
                        <GradientStop Color="#FF4F4F4F" Offset="1"/>
                    </LinearGradientBrush>
                </Grid.Background>
                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="14*"/>
                    <ColumnDefinition Width="0*"/>
                    <ColumnDefinition Width="208*"/>
                    <ColumnDefinition Width="386*"/>
                    <ColumnDefinition Width="193*"/>
                </Grid.ColumnDefinitions>
                <Button x:Name="stopButton" BorderThickness="0" Margin="84,16,87,16" MaxWidth="35" MaxHeight="35" Content="" Click="stopButton_Click" Grid.Column="2">
                    <Button.Background>
                        <ImageBrush ImageSource="/Material/stop-button.png" TileMode="Tile"/>
                    </Button.Background>
                </Button>
                <Button x:Name="playButton" BorderThickness="0" Margin="12,19,166,18" MaxWidth="35" MaxHeight="35" Content="" Click="playButton_Click" Grid.Column="2">
                    <Button.Background>
                        <ImageBrush ImageSource="/Material/play.png" TileMode="Tile"/>
                    </Button.Background>
                </Button>
                <Button x:Name="pauseButton" BorderThickness="0" Margin="47,21,131,21" MaxWidth="35" MaxHeight="35" Content="" Click="pauseButton_Click" Grid.Column="2">
                    <Button.Background>
                        <ImageBrush ImageSource="/Material/pause-multimedia-big-gross-symbol-lines.png" TileMode="Tile"/>
                    </Button.Background>
                </Button>
                <Label x:Name="volumeLabel" Content="Volume" Grid.Column="3" HorizontalAlignment="Left" Margin="237,10,0,0" VerticalAlignment="Top" Width="56" Panel.ZIndex="2" Foreground="White" MaxWidth="80" MaxHeight="25" RenderTransformOrigin="1.356,0.56" Height="25"/>
                <Button x:Name="addVideoButton" BorderThickness="0" Content="" HorizontalAlignment="Left" VerticalAlignment="Top" Height="41" Width="50" MaxWidth="500" MaxHeight="40" Grid.Column="3" Click="addVideoButton_Click" Margin="45,18,0,0">
                    <Button.Background>
                        <ImageBrush ImageSource="/playlist.png"/>
                    </Button.Background>
                </Button>
```

Рисунок 3.22 – Реалізація інтерфейсу додатку

XAML дозволить змінювати інтерфейс додатку в декілька кліків, залишаючи його адаптивним і зрозумілим для будь-якого користувача, що дозволить, в майбутньому, збільшувати функціонал доволі простим шляхом.

4. ПРОЕКТУВАННЯ

Всі внутрішні компоненти програми діляться на блоки, відповідно до призначення і типу рис.4.1

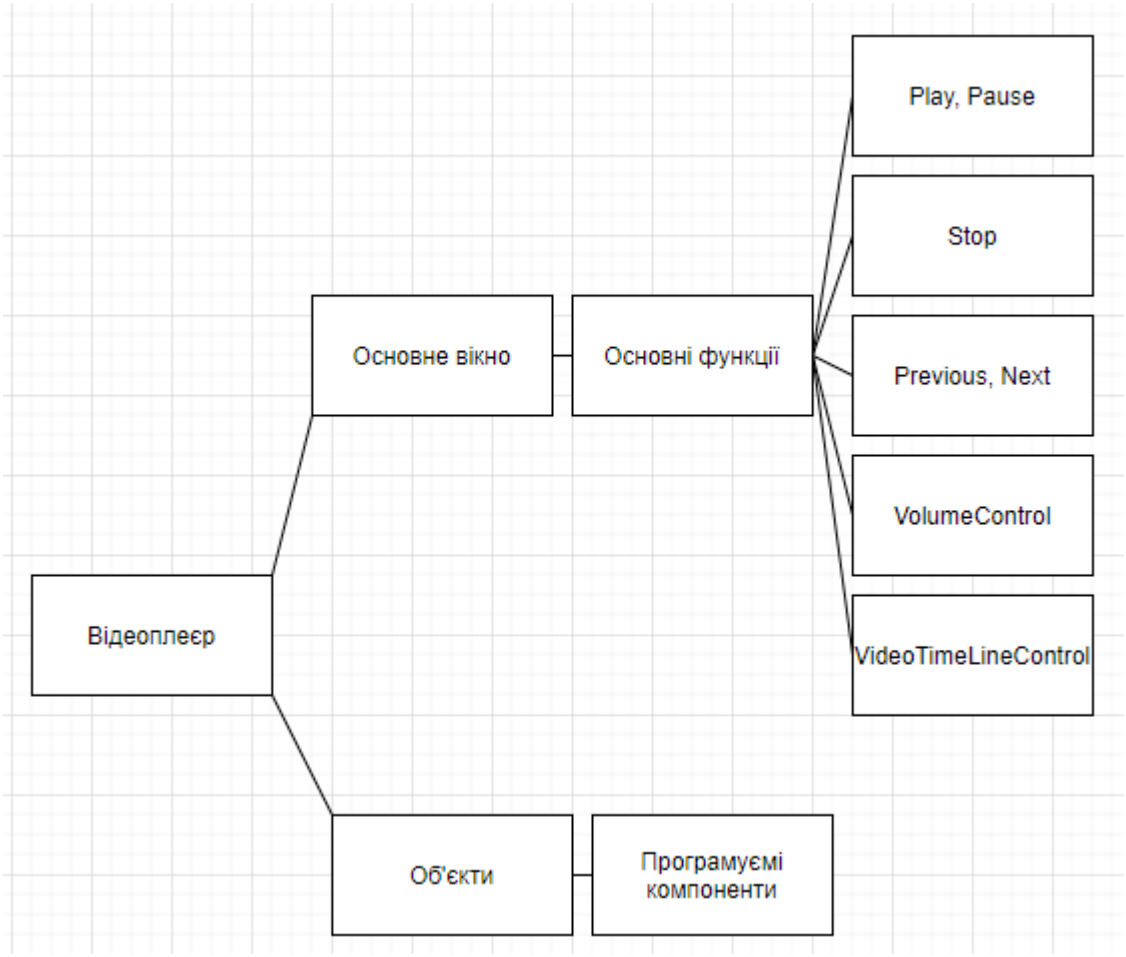


Рисунок 4.1 – Схема внутрішніх компонентів додатку

В даному додатку реалізовані наступні патерни проектування, такі як:

1. Single Responsibility Principle – кожен метод відповідає за 1 функціонал.
2. Template Method – шаблонний метод, який дозволяє наслідникам перевизначати кроки алгоритму, не змінюючи його.
3. Mediator – посередник, який дозволяє об’єктам взаємодіяти через певний об’єкт – посередник.

Інтерфейс додатку розроблений за принципом KISS (keep it short and simple), тобто, для кінцевого користувача він зрозумілий та очевидний.

Принцип KISS можна описати наступними пунктами:

1. Не використовуйте складну логіку, якщо це можна зробити просто.
2. Не бійтеся викидати код. Під час рефакторингу коду завжди видаляйте непотрібний код.
3. Вирішіть проблему, а потім закодуйте її. Перш за все витратьте час, щоб зрозуміти проблему та способи її вирішення. Тільки після цього стрибка вирішити це. Розбийте свої завдання на підзадачі та вирішуйте одне за іншим, роблячи їх простими.
4. Не використовуйте техгології без необхідності.

Даний додаток можна представити у вигляді UML-діаграми рис.4.2.

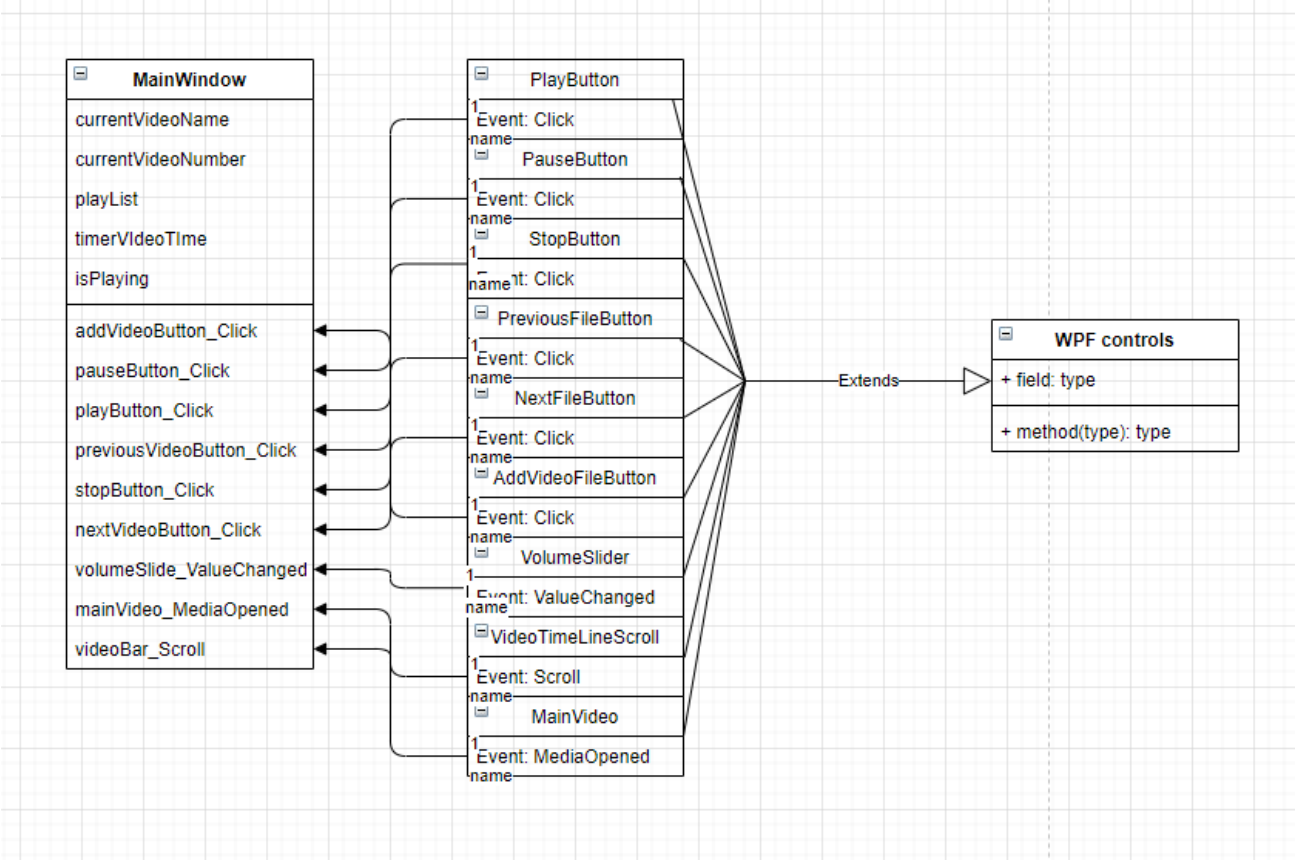


Рисунок 4.2 – UML-діаграма

Діаграма випадків використання UML - це основна форма системних / програмних вимог до нової слаборозвиненої програми.

У випадках використання вказується очікувана поведінка (що), а не точний

спосіб її здійснення (як). Описані випадки використання можна позначити як текстове, так і візуальне подання (тобто діаграму використання).

Ключова концепція моделювання випадків використання полягає в тому, що це допомагає нам розробити систему з точки зору кінцевого користувача. Це ефективний прийом для комунікації поведінки системи з точки зору користувача шляхом вказівки всієї видимої зовні системи поведінки. Діаграма випадків використання зазвичай проста. Вона не відображає деталей випадків використання:

1. Вона лише узагальнює деякі взаємозв'язки між випадками використання, суб'єктами та системами.

2. Вона не відображає порядок, у якому виконуються кроки для досягнення цілей кожного випадку використання.

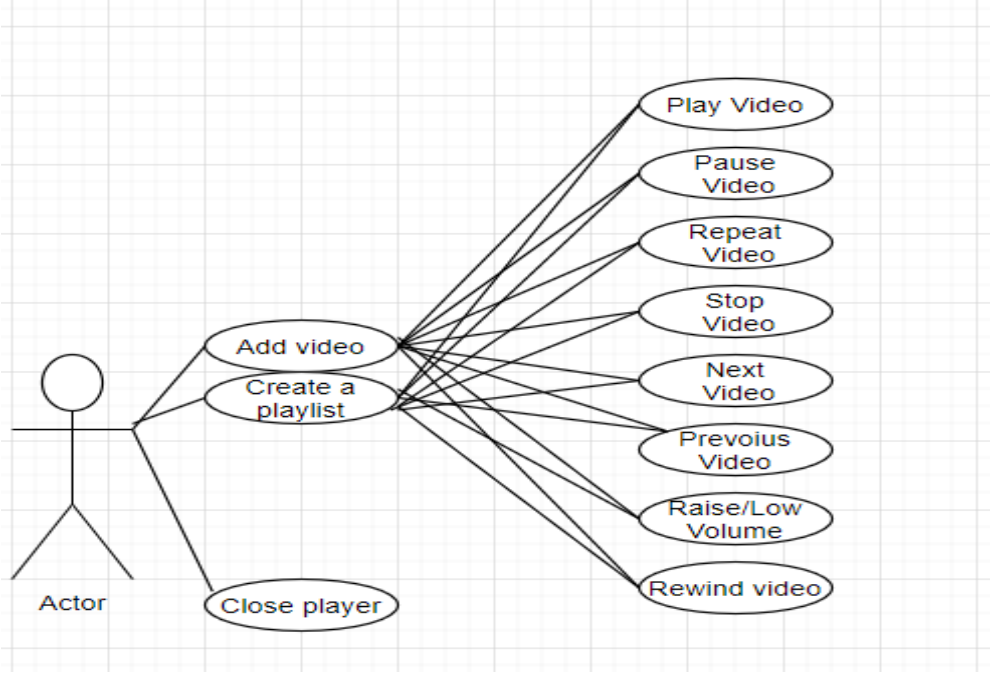


Рисунок 4.3 – UML-діграма випадків використання

Основні цілі діаграма випадків використання:

1. Вказати контекст системи.
2. Охопити вимоги системи
3. Перевірка архітектури системи.
4. Стимулювати реалзіацію та опис тест-кейсів

5. ТЕСТУВАННЯ ТА ЕКСПЛУАТАЦІЯ

5.1 Тестування

Створений застосунок має працювати чітко та без збоїв, з метою забезпечення належної якості, для тестування застосунку було створено набір фундаментальних тестів.

Тест-кейс – це артефакт, який описує сукупність кроків, умов та параметрів, необхідних для перевірки реалізації функції, що тестується або її частини.

Тест №1.

Додавання 1 відеофайлу.

Вхідні дані: користувач, за допомогою діалогового вікна додає відеофайл у відеоплеєр

Очікуваний результат: відеофайл додався і почав програватись.

Отриманий результат: відеофайл додався і почав програватись.

Тест пройдено успішно.

Тест №2.

Додавання декількох відеофайлів.

Вхідні дані: користувач, за допомогою діалогового вікна додає відеофайли у відеоплеєр

Очікуваний результат: відеофайли додались і почали програватись з першого.

Отриманий результат: відеофайли додались і почали програватись з першого..

Тест пройдено успішно.

Тест №3.

Тестування функціоналу кнопок керування відеофайлом.

Вхідні дані: користувач, після додавання відеофайлу, натискає наступні кнопки: Play, Pause, Stop, Next, Previous

Очікуваний результат: відеофайл почав програватись, відеофайл зупинився, відеофайл зупинився і відмотався на самий початок, завантажено наступний

відеофайл з плейліста, завантажено минулий відеофайл з плейліста.

Отриманий результат: відеофайл почав програватись, відеофайл зупинився, відеофайл зупинився і відмотався на самий початок, завантажено наступний відеофайл з плейліста, завантажено минулий відеофайл з плейліста.

Тест пройдено успішно.

Тест №4.

Перевірка слайдеру звуку.

Вхідні дані: користувач, за допомогою слайдеру звуку змінює гучність програвання відеофайлу.

Очікуваний результат: гучність змінилась.

Отриманий результат: гучність змінилась.

Тест пройдено успішно.

Тест №5.

Тестування кнопок керування за відсутності відеофайлу.

Вхідні дані: користувач, не додавши відеофайл, натискає кнопку Next\Prevoious.

Очікуваний результат: нічого не відбулось.

Отриманий результат: нічого не відбулось.

Тест пройдено успішно.

Тест №6.

Тестування кнопок керування за відсутності відеофайлу.

Вхідні дані: користувач, не додавши відеофайл, натискає кнопку Play\Pause.

Очікуваний результат: нічого не відбулось.

Отриманий результат: нічого не відбулось.

Тест пройдено успішно.

Тест №7.

Тестування кнопок керування за відсутності відеофайлу.

Вхідні дані: користувач, не додавши відеофайл, використвоує слайдер звуку.

Очікуваний результат: нічого не відбулось.

Отриманий результат: нічого не відбулось.

Тест пройдено успішно.

5.2 Інструкція користувача

Це програмне забезпечення надає користувачеві спектр можливостей по роботі з «Аудіо плеєр" 3pmPlayer "» - відтворення аудіо файлів, підстроювання частот відтвореного треку, красива візуалізація.

Для функціонування програми необхідно наступне технічне забезпечення з наступними мінімальними характеристиками:

1. Процесор - Intel Pentium IV / 1.8 Ghz або вище.
2. Обсяг оперативної пам'яті - 512 Mb RAM.
3. Місце на жорсткому диску - 10 Mb HDD.
4. Вбудована / зовнішня звукова карта.
5. Вбудована / зовнішня відеокарта.
6. Монітор.

Після запуску додатку, для початку перегляду відеофайла, потрібно натиснути кнопку, для створення плейліста або відкрити натинути правою кнопкою миші на відеофайл, який потрібно відкрити, та обрати пункт “Відкрити за допомогою” і обрати “CSPlayer” рис.5.1.

Для коректної роботи додатку, рекомендується виокристовувати Windows версії 7 або вище. Перш за все, у налаштуваннях Windows (актуально для Windows 10,8,7) додати даний плеєр у список стандартних відеоплеєрів, на рівні Windows Media Player.

Також, необхідні наступні умови для запуску Windows 7,8,10 на комп'ютері:

1. 32-розрядний (x86) або 64-розрядний (x64) процесор з тактовою частотою 1 GHz (ГГц) або вище.
2. 1 гігабайт (ГБ) (для 32-розрядної системи) або 2 ГБ (для 64-розрядної системи) оперативної пам'яті (ОЗУ).
3. 16 гігабайт (ГБ) (для 32-розрядної системи) або 20 ГБ (для 64-розрядної системи) простору на жорсткому диску.

4. Графічний пристрій DirectX 10 з драйвером WDDM версії 1.0 або вище.

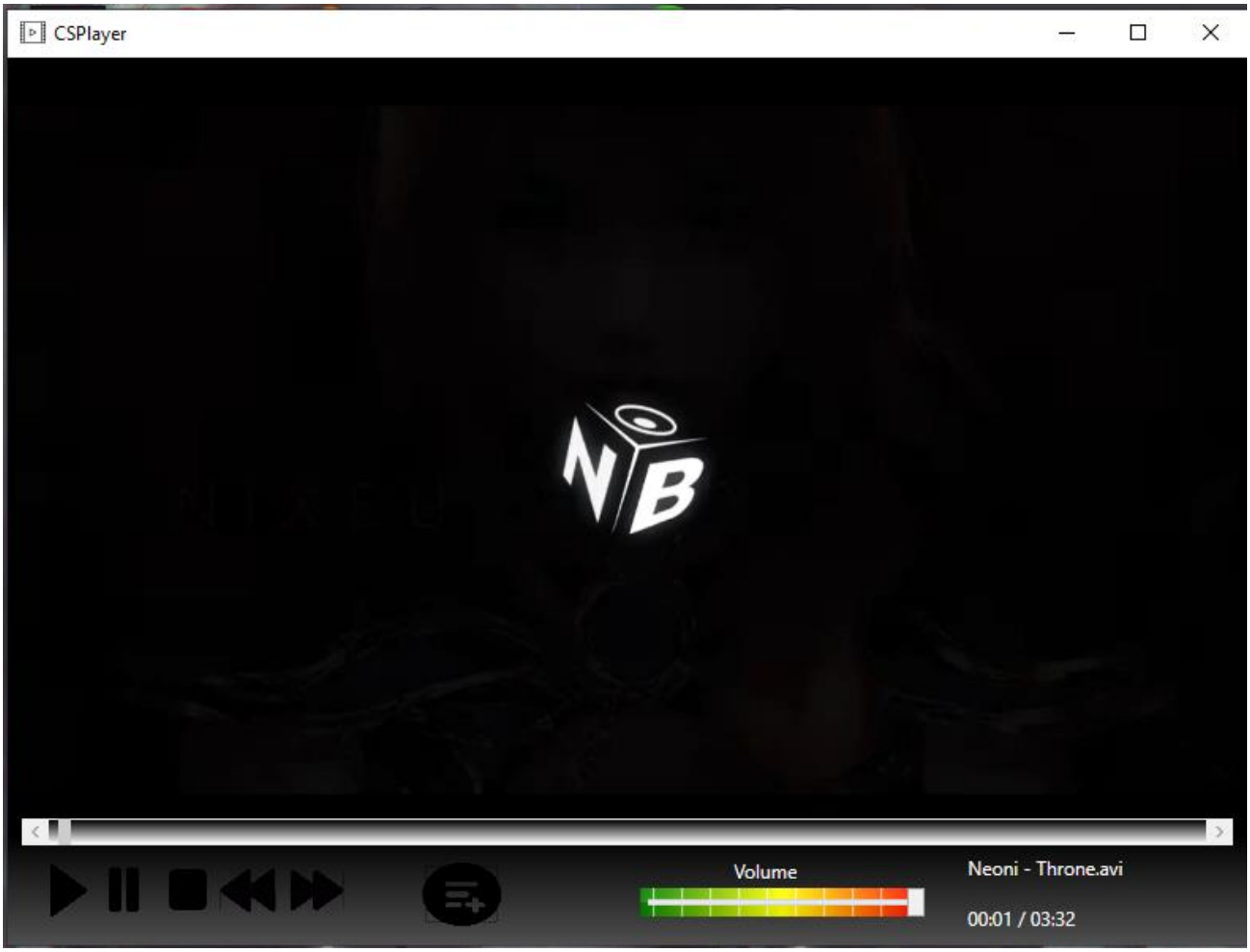


Рисунок 5.1 – Запуск додатку та відтворення відеофайлу

Надалі, при перегляді, у користувача є можливість маніпулювати відеофайлом, використовуючи доступні кнопки та слайдери.

Користувач може завантажувати файл форматів: .mp4, .mov, .asf, .avi, .wmv.

Надалі, кількість доступних форматів буде збільшуватись та будуть додаватися аудіо формати.

ВИСНОВКИ

У результаті виконання даної дипломної роботи було розроблено десктопний додаток, який дозволяє переглядати відеофайли. У роботі міститься основні принципи розробки додатку.

1. Під час роботи над дипломним проектом було проаналізовано існуючі аналоги, та програми, схожі на розроблений продукт, що знаходяться у вільному доступі. Було розглянуто проблеми, які виникають під час розробки додатків, які підтримують режим віртуальної реальності.

2. Створено концепцію проекту.

3. Розроблений основний функціонал відеоплеєра.

4. Дизайн додатку був розроблений з урахуванням всіх нюансів, а саме: основу дизайну складає XAML, але деякі його частини були розроблені за допомогою сторонніх інструментів, таких як Photoshop.

Розроблена система дасть змогу відтворювати відеофайли, для подальшого перегляду. Також додаток дає моєливість повністю контролювати даний відеофайл і створювати свої плейлісти.

У подальшому планується розробти:

1. Кросплатформеність додатку.
2. Покращений дизайн.
3. Доповнення функціоналу (можливість програвати .webm файли, аудіо файли та радіо)

ПЕРЕЛІК ПОСИЛАНЬ

1. Mark J. Price, C# 7 and .NET Core: Modern Cross-Platform Development - Second Edition/- Packt Publishing– с. 96 – 120.
2. Ben Albahari, Joseph Albahari C♯ 6.0: Pocket Reference /- O'Reilly Media; 1st edition - с. 117 – 123.
3. ANDREW TROELSEN, Philip Japikse, C# 6.0 and the .NET 4.6 Framework /- Apress, 2015 – с. 1083.
4. Joseph Albahari, Ben Albahari, C# 7.0 in a Nutshell: The Definitive Reference /- "O'Reilly Media, Inc.", 2017 – с. 120.
5. Griffiths I. Programming C# 8.0: Build Cloud, Web, and Desktop Applications / – "O'Reilly Media Inc.", 2019 – с. 140.
6. Коноваленко І.В. Програмування мовою C# 6.0 / – Тернопіль, ТНТУ, 2016. с. 227.
7. Desktop Guide (WPF .NET). [Електронний ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу: https://docs.microsoft.com/en-us/dotnet/desktop/wpf/overview/?view=netdesktop-5.0 Дата звернення: 06.04.2021.
8. Dialog boxes overview (WPF .NET). [Електронний ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу: https://docs.microsoft.com/en-us/dotnet/desktop/wpf/windows/dialog-boxes-overview?view=netdesktop-5.0 Дата звернення: 07.04.2021.
9. Git Branching. [Електронний ресурс]: [Веб-сайт]. –Електронні дані. – Режим доступу: https://git-scm.com/book/en/v2/Git-Branching-Branches-in-a-Nutshell Дата звернення: 07.04.2021.
10. Photoshop. About painting tools, presets, and options. [Електронний ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу: https://helpx.adobe.com/photoshop/using/painting-tools.html Дата звернення: 15.04.2021.

11. Photoshop. Customize color settings. [Електронний ресурс]: [Веб-сайт]. – Електронні дані. – Режимдоступу: https://helpx.adobe.com/photoshop/using/color-settings.html Дата звернення: 15.04.2021.

12. Scott Chacon and Ben Straub, Pro GIT/- APRESS– с. 65 – 118.

13. Andrew Faulkner, Conrad Chavez, Adobe Photoshop CC Classroom in a book/- Adobe Press 2017– с. 15 – 48.

14. Jeffrey Richter, CLR via C#/- Pearson Education 2012– с. 150.

ДОДАТОК

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«Розробка десктопного додатку для перегляду відеофайлів мовою C#»

Виконав студент 4 курсу
Групи ПД-41
Кисельов В'ячеслав Олександрович
Керівник роботи
Дібрівний Олесь Андрійович

Київ – 2021

МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

У наш час важко вирватись і поїхати кудись, сходити в кіно на улюблений фільм або навіть прогулятися – це ризик, а домашні розваги – досить обмежені. Але у наші дні є способи, якими ви можете насолоджуватися фільмами, не обов'язково відвідуючи кінотеатри. Тепер ви можете відпочити та насолодитися своїм фільмом, не виходячи з власного будинку чи двору.

Мета роботи: розробити десктопний додаток для перегляду відеофайлів.

Об'єкт дослідження: програмне забезпечення для роботи з файлами мультімедії.

Предмет дослідження: додаток для перегляду відеофайлів.

АНАЛОГИ
“BS.player”

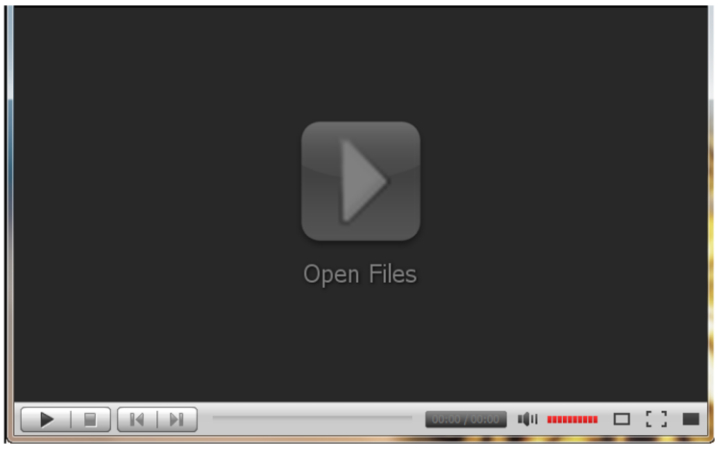
Переваги

1. Широкий спектр функціоналу.
2. Низька вимогливість до обладнання користувача.

Недоліки

1. Весь додатковий функціонал є платним.
2. Інтерфейс доволі складний, для звичайного користувача.

АНАЛОГИ
“MP4 Player”



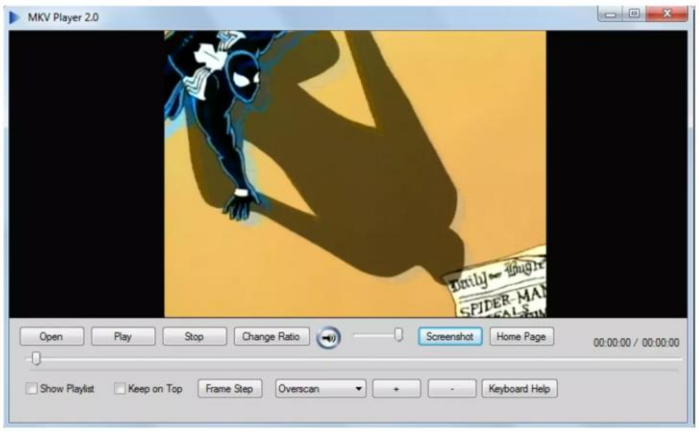
Переваги

1. Простий інтерфейс.
2. Низька вимогливість. до обладнання користувача
3. Доступна функція створення плейлістів.

Недоліки

1. Сам медіаплеєр платний.
2. Не підтримує інших мов, крім англійскої.

АНАЛОГИ
“MKV Player”



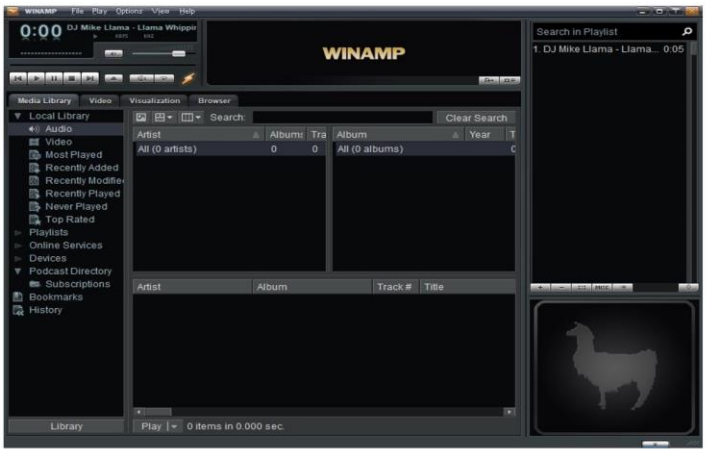
Переваги

1. Підтримує формат MKV.

Недоліки

1. Інтерфейс доволі складний, для звичайного користувача.
2. Слабий функціонал.
3. Не підтримує інших форматів, крім MKV.

АНАЛОГИ
“WINAMP”



Переваги

1. Підтримка аудіо та відео.
2. Підтримка найпопулярніших аудіо та відео форматів.
3. Стильний графічний інтерфейс.
4. Підтримка медіабібліотек.

Недоліки

1. Занепад розробки.
2. Перехід до платного функціоналу.

ТЕХНІЧНІ ЗАВДАННЯ

Має бути реалізований наступний функціонал:

1. Можливісь відтворення та перегляду відеофайлів.
2. Можливість маніпуляції з відеофайлом.
3. Можливість створення плейлистів.
4. Гнучкість та адаптивність до змін.
5. Гнучкий та простий інтерфейс.
6. Наявність активних кнопок, зрозумілих користувачу.
7. Наявність простої і зрозумілої робочої області з можливостю програвання відеофайлів.
8. Можливість створення плейлістів.

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

Мова програмування C#

C# – сучасна об'єктно-орієнтована мова програмування і належить сімейству мов C. C# набув за рахунок гнучкості, якщо порівнювати з деякими іншими мовами програмування.

Є велика кількість різних додатків, які можуть бути написані за допомогою C#.Net і Visual Studio:

1. Додатки для Windows.
2. Мобільні додатки.
3. Веб-додатки.
4. Ігри.
5. Додатки для Android і ios, які розробляються за допомогою додаткових фреймворків, таких як Xamarin або Mono.

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

WPF

WPF - розшифровується як Windows Presentation Foundation, - це підхід Microsoft до фреймворку GUI, що використовується з фреймворком .NET.

Існує багато графічних інтерфейсів, але для розробників .NET найбільш цікавими на даний момент є WinForms та WPF. WPF є найновішим, але Microsoft все ще розвиває та підтримує WinForms. Між двома фреймворками є досить багато відмінностей, але їх призначення однакове - полегшити створення додатків із чудовим графічним інтерфейсом.

WPF - це механізм, який відповідає за створення, відображення та маніпулювання користувальницькими інтерфейсами, документами, зображеннями, фільмами та медіа в операційних системах Windows 7 та пізніших версіях. WPF - це набір бібліотек, які мають усі функції, необхідні для створення, запуску, запуску та керування клієнтськими програмами Windows.

XAML - це нова описова мова програмування, розроблена корпорацією Майкрософт для написання користувацьких інтерфейсів для керованих додатків наступного покоління. Вона є основою для інтерфейсу WPF.

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

Adobe Photoshop

Adobe Photoshop - це програма для редагування зображень та ретушування фотографій для використання на комп'ютерах з ОС Windows або MacOS.

Photoshop пропонує користувачам можливість створювати, вдосконалювати чи іншим чином редагувати зображення та ілюстрації. Зміна фонів, імітація картини з реального життя або створення альтернативного уявлення про Всесвіт - все це можливо в Adobe Photoshop. Це найбільш широко використовуваний програмний інструмент для редагування фотографій, маніпуляцій із зображеннями та ретуші для численних форматів файлів зображень та відео.

Інструменти в Photoshop дають можливість редагувати як окремі зображення, так і великі партії фотографій. Існує кілька версій Photoshop, включаючи Photoshop CC, Photoshop Elements, Photoshop Lightroom і Photoshop Express, версію Photoshop для iOS зі зменшеними функціями.

Це найкращий вибір для створення сучасного дизайну окремих елементів додатку.

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ
GIT

Git – це розподілена система контролю версій, система дає можливість розробникам відстежувати зміни в файлах та сумісно працювати з іншими розробниками . Найбільш відомі особливості Git – це простий дизайн, швидкодія, підтримка нелінійної розробки, повна децентралізація та, за необхідності, можливість роботи з великими проектами.

GitHub – сервіс онлайн-хостинга репозиторіїв, котрий вміщує в собі всі ті функції, що підтримує Git - функції розподільного контролю версій і функціональність управління початковим кодом. Використання GitHub разом з Git дає розробникам можливість збереження коду онлайн та зручно взаємодіяти з іншими розробниками. Важлива та зручна в розробці функція – це історія комітів, завдяки чому просто відстежувати які саме зміни було внесено. В історії комітів виділено файли та частини коду, в котрі було внесено зміни (рис. 2.13). Можливість порівняти частини коду спрощує пошук можливих помилок та відладку коду.

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ
SoureTree

Хоч гіт і дуже зручний в плані супроводження проекту, у нього є свої недоліки, один з них – постійні десятки консольних команд, які забирають багато часу та не завжди коректно працюють, оскільки, якщо була допущена помилка – взнати де вона і чому вона виникла – це доволі складна задача.

На допомогу у вирішені цих проблем приходить SourceTree. SourceTree було створено для того, щоб зробити Git доступним для кожного розробника - особливо тих, хто новачок у Git. Кожна команда

Git - це лише один клік за допомогою інтерфейсу SourceTree. Даний додаток спрощує наступний функціонал:

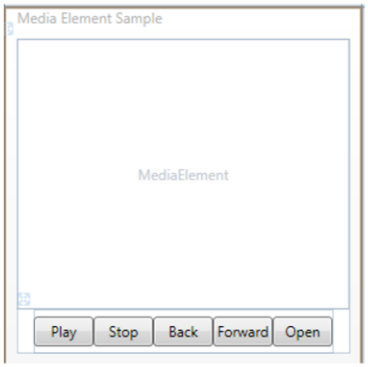
1. Створення та клонування репозиторіїв з будь-якого місця.
2. Commit, push, pull та merge в декілька кліків.
3. Розвинена система пошуку помилок та конфліктів.
4. Відстежування всього життєвогу циклу проекту.

Як вже зазначалось, супроводження реалізації додатку буде проводитись за допомогою GitHub. Використання SourceTree спрощує роботу з Гіт до декількох кліків.

МЕТОДИ ТА КЛАСИ ПРОГРАМИ

Для створення основного вікна для перегляду відеофайлів – будемо використовувати елемент WPF MediaElement, який дозволяє відвторювати аудіо\відеофайли усіх популярних форматів.

Також MediElement надає простий функціонал для маніпуляції з відео.



Використовуючи даний елемент ми отримуємо доступ до наступних методів, без потреби їх реалізовувати:

1. **mediaElement.play** – (програє файл з поточної секунди)
2. **mediaElement.pause** - (зупиняє програвання файлу)
3. **mediaElemnt.stop** – (зупиняє програвання файлу – наступний його запуск буде з смаого початку)

МЕТОДИ ТА КЛАСИ ПРОГРАМИ

Наступним кроком, щоб полегшити собі роботу в майбутньому була реалізована можливість завантаження відеофайлу в додаток.

На кнопку AddVideo був підписаний метод, який при натисканні викликав діалогове вікно вибору з сортування по аудіоформатам. При виборі потрібного відеофайла – діалогове вікно закривалось, та файл автоматично починав програватись

```
private void addVideoButton_Click(object sender, RoutedEventArgs e)
{
    OpenFileDialog ofg = new OpenFileDialog()
    {
        Filter = "(mp3,wav,mp4,mov,wmv,mpg)|*.mp3;*.wav;*.mp4;*.mov;*.wmv;*.mpg;*.avi|all files|*.*",
        InitialDirectory = Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments)
    };
    ofg.Multiselect = true;
    if (ofg.ShowDialog() == true)
    {
        foreach (string item in ofg.FileNames)
            {
                playList.Add(item);
            }
        VideoLoad(playList[0], 0);
    }
```

МЕТОДИ ТА КЛАСИ ПРОГРАМИ

Важлива частина будь-якого медіаплеєра – можливість регулювання гучності медіфайлу.

Для реалізація даного функціоналу у своєму додатку я використав Slider.

Slider представляють елементи, засновані на діапазонах значень. Тобто вони зберігають і відображають числові дані на певному діапазоні. Він є наслідником класа RangeBase, того наслідує від нього такі параметри, як **Value** та **Minimum\Maximum**.

Все що потрібно для реалізації регулятора гучності – це при виклику івенту ValueChanged (зміна значення – якщо користувач посуне повзунок) прирівнювати значення гучності відеофайлу до значення повзунка

```
private void volumeSlide_ValueChanged(object sender, RoutedPropertyChangedEventArgs<double> e)
{
    if (isPlaying)
        mainVideo.Volume = (double)volumeSlide.Value;
}
```

МЕТОДИ ТА КЛАСИ ПРОГРАМИ

Для кнопки Next суть метода буде полягати в, перш за все, перевірці, чи відео грає, для цього створена глобальна зміна isPlaying типу bool, яка відстежує статус відео. Надалі, потрібно перевіряти, яке відео з плейлісту грає, та, якщо воно не останнє – переходити на наступний елемент листу, якщо відео останнє – запускати перший елемент плейлисту

```
private void nextVideoButton_Click(object sender, RoutedEventArgs e)
{
    if (isPlaying)
    {
        if (currentVideoNumber + 1 != playList.Count)
        {
            ChangeVideo(playList[currentVideoNumber + 1], currentVideoNumber + 1);
        }
        else
        {
            ChangeVideo(playList[0], 0);
        }
        mainVideo.Play();
    }
}
```

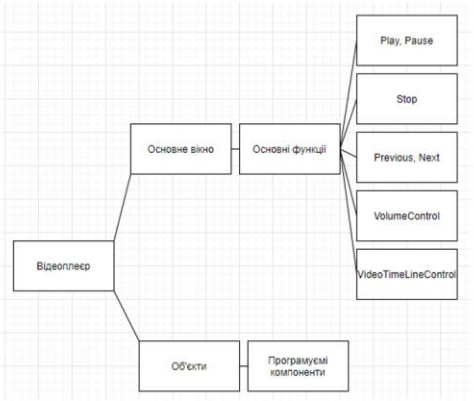
МЕТОДИ ТА КЛАСИ ПРОГРАМИ

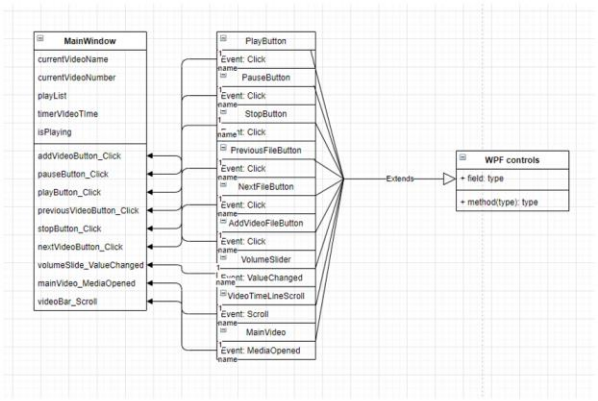
Для кнопки Previous функціонал буде майже аналогічний, але буде перевірятись, якщо грає не перший елемент, то має завантажитись минуле відео з плейліста, якщо грає перший елемент плейліста, то має завантажитись останній елемент плейлісту

```
private void previousVideoButton_Click(object sender, RoutedEventArgs e)
{
    if (isPlaying)
    {
        if (currentVideoNumber == 0)
        {
            ChangeVideo(playList[playList.Count-1], playList.Count-1);
        }
        else
        {
            ChangeVideo(playList[currentVideoNumber - 1], currentVideoNumber - 1);
        }
        mainVideo.Play();
    }
```

МЕТОДИ ТА КЛАСИ ПРОГРАМИ

Всі внутрішні компоненти програми діляться на блоки, відповідно до призначення і типу



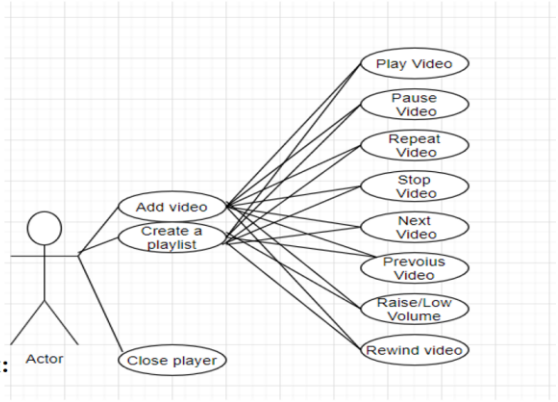


Даний додаток можна представити у вигляді UML-діаграми

МЕТОДИ ТА КЛАСИ ПРОГРАМИ

Діаграма випадків використання UML - це основна форма системних / програмних вимог до нової слаборозвиненої програми.

У випадках використання вказується очікувана поведінка (що), а не точний спосіб її здійснення (як). Описані випадки використання можна позначити як текстове, так і візуальне подання (тобто діаграму використання).



Основні цілі діаграма випадків використання:

1. Вказати контекст системи.
2. Охопити вимоги системи
3. Перевірка архітектури системи.
4. Стимулювати реалзіацію та опис тест-кейсів

ВИСНОВКИ

1. Під час роботи над дипломним проектом було проаналізовано існуючі аналоги, та програми, схожі на розроблений продукт, що знаходяться у вільному доступі. Було розглянуто проблеми, які виникають під час розробки додатків, які підтримують режим віртуальної реальності.

2. Створено концепцію проекту.

3. Дизайн додатку був розроблений з урахуванням всіх нюансів, а саме: основу дизайну складає XAML, але деякі його частини були розроблені за допомогою сторонніх інструментів, таких як Photoshop.

4. Методи розробки, що були використані у додатку дозволяють використовувати додаток, людям без спеціальних навичок і знань у роботі з комп'ютером.

У подальшому планується розробти:

1. Кросплатформеність додатку.
2. Покращений дизайн.
3. Доповнення функціоналу (можливість програвати .webm файли, аудіо файли та радіо)

ДЯКУЮ ЗА УВАГУ!