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 додатку для прослуховування аудіо файлів під систему Android**»

Виконав: студент 4 курсу, групи ПД-41

спеціальності

121 Інженерія програмного забезпечення

Лісовий Д.О.

Керівник Дібрівний О.А

Рецензент

Київ – 2021

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ

ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Факультет 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

Лісовий Денис Олександрович

1. Тема роботи: «Розробка додатку для прослуховування аудіо файлів під систему Android»
Керівник роботи Дібрівний Олесь Андрійович, старший викладач кафедри інженерії програмного забезпечення
затверджені наказом вищого навчального закладу від “12” березня 2021 року №65.

2. Строк подання студентом роботи 01.06.2021

3. Вихідні дані до роботи:
Методи розробки програмного забезпечення;
Microsoft Visual Studio;

Xamarin
Офіційна документація мови програмування C#.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
4.1 Характеристика та опис програмного продукту;
4.2 Опис програмних засобів;
4.3 Опис реалізації;
4.4 Тестування;

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;
7. Розроблений додаток;

6. Дата видачі завдання 19.04.2021

КАЛЕНДАРНИЙ ПЛАН

№ з /п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Вибір теми бакалаврської роботи	01.10.2020	Виконано
2	Дослідження актуальності теми	15.11.2020	Виконано
3	Розгляд аналогів	15.01.2021	Виконано
4	Дослідження програмних засобів	10.03.2021	Виконано
5	Проектування	25.03.2021	Виконано
6	Розробка додатку з використанням Xamarin	30.04.2021	Виконано
7	Тестування стабільності роботи додатку	10.05.2021	Виконано
8	Проходження нормоконтролю та антиплагіату	25.05.2021	Виконано
9	Захист дипломної роботи	01.06.2021	

Студент _____ Лісовий Д.О.

Керівник роботи _____ Дібрівний О.А.

РЕФЕРАТ

Текстова частина бакалаврської роботи с. 51, рис. 38, джерел 8.

Об`єкт сфери дослідження – додатки для відтворення аудіо файлів.

Предмет дослідження – аудіо програвач.

Мета дослідження – розробка додатку для програвання аудіо файлів з зручним інтерфейсом, без реклами та порівняння Xamarin як фреймворку для розробки додатків під систему Android.

У дипломній роботі було розроблено та протестовано додаток, проаналізовано його аналоги, поріняно Xamarin як фреймворк з альтернативними рішеннями розробки додатків під систему Android, висвітлено можливі шляхи розвитку даного додатку.

Ключові слова: Xamarin, C#, Android, Visual Studio, Об’єктно орієнтоване програмування.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

API	Application Programming Interface
BSD	Berkeley Software Distribution
XAML	eXtensible Application Markup Language
UWP	Universal Windows Platform
WPF	Windows Presintation Foundation
LIFO	Last In First Out
ART	Android Runtime

ВСТУП

Сьогодні мало хто може уявити своє життя без музики, люди слухають її вдома, по дорозі на роботу, школу, університет та інших місцях. Звісно існують спеціальні пристрої для прослуховування музики, але найпоширенішим є телефон. Хоча нажаль не всі вони мають добре зроблений програвач.

Об`єкт сфери дослідження – додатки для відтворення аудіо файлів.

Предмет дослідження – аудіо програвач.

Мета дослідження – розробка додатку для програвання аудіо файлів з зручним інтерфейсом, без реклами та порівняння Xamarin як фреймворку для розробки додатків під систему Android.

У дипломній роботі було розроблено та протестовано додаток, проаналізовано його аналоги, поріняно Xamarin як фреймворк з альтернативними рішеннями розробки додатків під систему Android, висвітлено можливі шляхи розвитку даного додатку.

РОЗДІЛ 1. ТЕОРЕТИЧНА ЧАСТИНА

1.1. Музичний плеєр

1.1.1. Визначення та актуальність

Плеєр – це додаток, який дозволяє відтворювати медіа файли.

Існують різного роду медіа файли: аудіо, відео, фото та ін. Що стосується додатку, презентованого в даній дипломній роботі – нас найбільше цікавлять аудіо файли.

Ні для кого не секрет, що сьогодні більшість людей полюбляє слухати музику, це стає невід'ємною частиною нашого життя ще з малого віку, особливо в шкільну пору, адже вид музики яку ти слухаєш є чудовим способом самовираження, тому це є важливою складовою кожної особистості.

Хоча не всі використовують музику таким чином, існують різні способи її використання, це може бути просто відпочинок. Прослуховуючи аудіо можна знайомитись з різними мовами, адже розібравши пісню її досить легко вчити, таким чином збільшуючи свій словниковий запас. Дехто використовує музику для мотивації займаючись спортом, або іншими справами зв'язаними з фізичною працею. Можливо також використовувати аудіо для приглушення інших шумів, як наприклад коли ваш сусід розпочав ремонт, адже слухати музику набагато приємніше ніж перфоратор.

Говорячи про все це з'являється думка про засоби, які допомагають людям насолоджуватись цього роду медіафайлами. На думку одразу ж спадають спеціальні кишенькові плеєри, смартфони та комп'ютери. Із цих трьох варіантів був обраний другий оскільки він має деякий список суттєвих переваг над іншими:

1. Смартфони – це найпоширеніший гаджет сьогодення.
2. Не потрібно купувати та носити з собою ще один девайс.

3. Досить компактний пристрій, який можна використовувати будь-де.

Наступний аспект, який був розглянутий – це система для якої буде розроблюватись програмне забезпечення. Станом на дві тисячі двадцять перший рік найпопулярнішою системою для даного роду пристроїв є Android. Також альтернативою може бути IOS система, але у неї є декілька недоліків в порівнянні з попередньою:

1. Все ж Android система поширеніша ніж IOS.
2. Для поширення додатку потрібні кошти в обох випадках, проте на Android системах він складає 25 доларів США і це разовий платіж, а на IOS системі дана можливість доступна лише при наявності щомісячного платежу, який складає 25 доларів США.

1.1.4. Аналіз аналогів

В сфері мобільних аудіо плеєрів цей додаток не першопроходець, існує велика кількість аналогів.

Audify Music Player

Це додаток з топу при пошуку «Music Player» в Google Play рис.1.1.

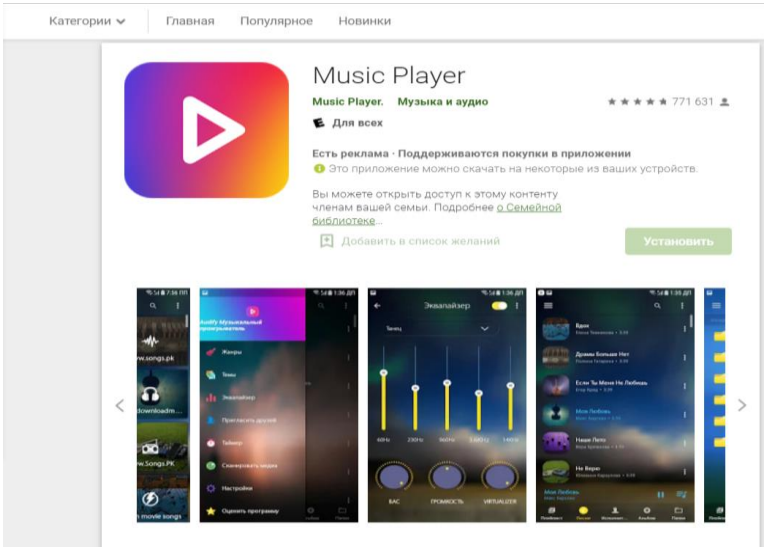

Рисунок 1.1 - Audify Music Player

Він має кілька переваг у порівнянні з аналогами, а саме:

1. Кращий еквалайзер.
2. Можливість створення плейлистів.

Проте не обійшлося без недоліків:

1. Велика кількість платного контенту.
2. Баг з програванням музики у плейлисті по колу.
3. Велика кількість реклами.

Music Player

Mobile_V5 пропонує ринку свою версію додатку для аудіо файлів рис. 1.2.

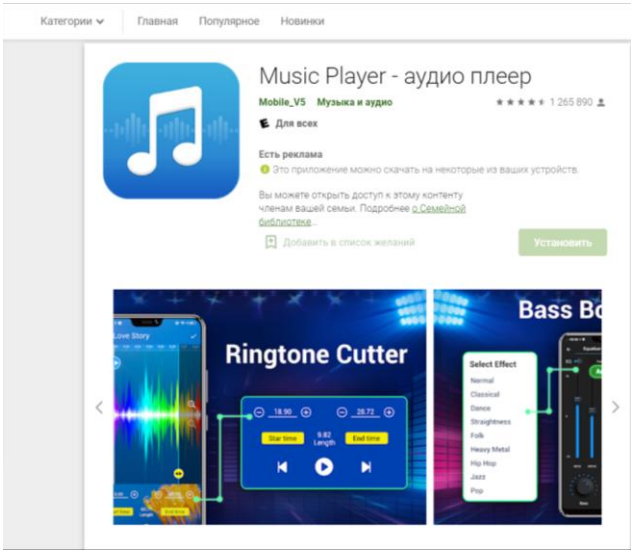

Рисунок 1.2 – Music Player Mobile_V5

До переваг даного додатку можна віднести:

1. Можливість часткового редагування аудіофайлів.
2. Наявність еквалайзера.

Його недоліками є:

1. Зміна екрану блокування смартфону при використанні.
2. Велика кількість реклами.
3. Часті вильоти додатку.

Музичний плеєр – MP3 – плеєр

Також гарна альтернатива, він як і вище перечислені додатки має гарний функціонал, доволі приємний та зрозумілий інтерфейс, тож до його переваг можна віднести:

1. Простота інтерфейсу.
2. Можливість часткового редагування аудіофайлів.
3. Еквалайзер.

Рисунок 1.3 – Інтерфейс екрану блокування

Його недоліками є:

1. Реклама.
2. Зміна екрану блокування при використанні рис. 1.3.
3. Баг з самовільним програванням аудіо.

РОЗДІЛ 2. ОПИС ПРОГРАМНИХ ЗАСОБІВ

Для розробки даного програмного продукту під систему «Android» було обрано мову програмування C# та Xamarin фреймворк. Звісно існують інші інструменти для розробки мобільних додатків такі як Android Studio у поєднанні з Java, але було обрано попередній варіант задля порівняння ефективності розробки на даній платформі.

2.1. Використані засоби

При розробці даного програмного забезпечення були застосовані такі засоби:

1. Комп’ютер з ОС Windows 10.
2. Visual Studio Community 2019.
3. Xamarin .Net Framework.
4. Мова програмування C#.
5. Мова розмітки XAML.
6. GIT.
7. SourceTree.
8. Android Device Manager.
9. П’ять реальних Android пристроїв.

2.2. Visual Studio Community 2019

Visual Studio – це інтегроване середовище розробки, яке використовується в цілях написання, редагування та відладки коду. Також даний засіб включає в себе інструменти автозавершення коду, графічні редактори, компілятори та містить

багато інших можливостей, які полегшують процес розробки програмного забезпечення.

Центральне вікно – це редактор коду, він відображає вміст файлів та дозволяє його доповнювати, копіювати та видаляти.

Також з правого боку є велика кількість допоміжних панелей керування, що допомагають у виконанні різного роду задач, також є панелі без яких з’явилося б багато дискомфорту при розробці програмного забезпечення, як наприклад:

1. Провідник рішення – панель, шо показує дерево файлів у файловій системі, дозволяє відкривати або видаляти існуючі, також створювати нові файли та керувати їх ієрархією у своєму рішенні рис. 2.1.

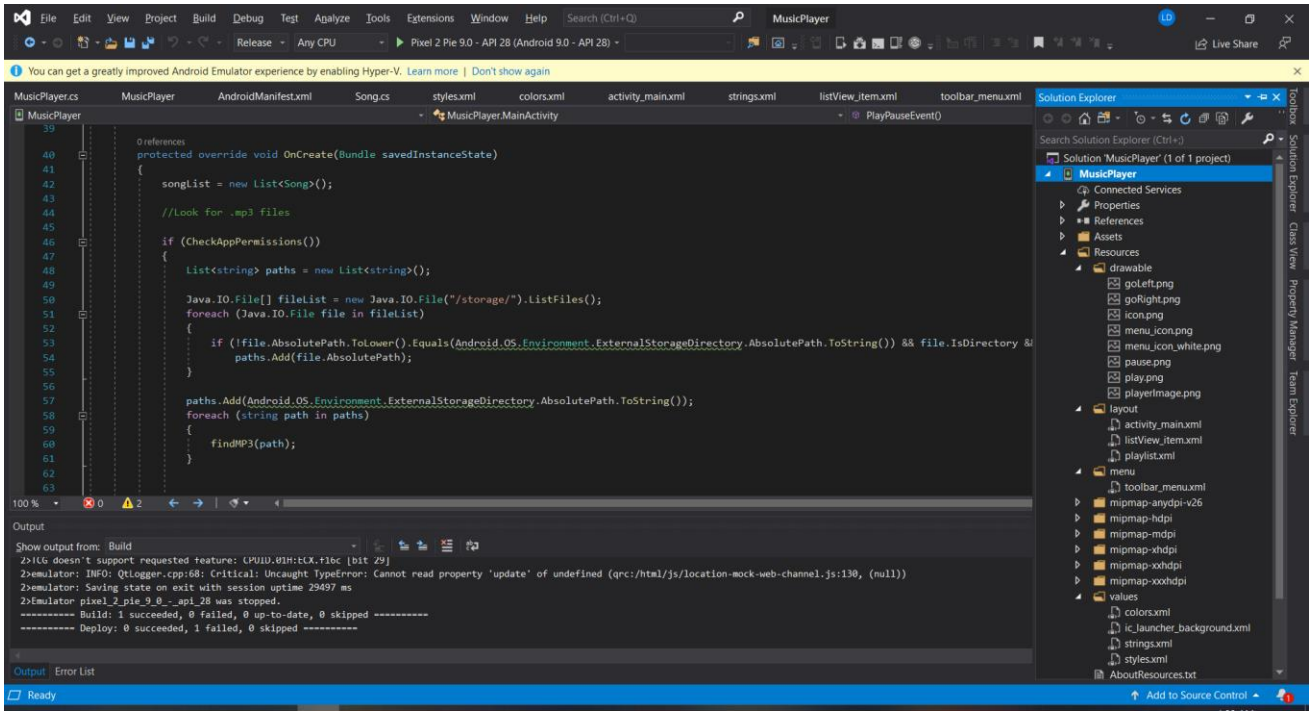

Рисунок 2.1 – Інтерфейс Visual Studio Community 2019

2. Також існує вікно властивостей створеного об’єкту, воно стає в нагоді при формуванні інтерфейсу додатку, оскільки весь інтерфейс складається з об’єктів.

3. Третє вікно, без якого неможливо уявити створення додатку – це вікно інструментів, яке містить велику кількість різних об'єктів, що можна застосовувати при розробці інтерфейсу програмного забезпечення.

При написанні коду – неможливо не згадати про автозавершення коду, в даному середовищі розробки його назва – IntelliSense. Цей інструмент пропонує варіанти подальшої побудови розпочатої команди, або доповнення складного звернення до об’єктів, чи їх властивостей рис. 2.2.

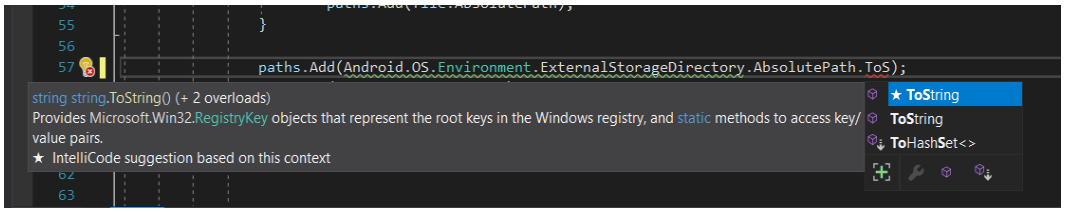

Рисунок 2.2 – Visual Studio IntelliSense

Створення проекту в середі розробки Visual Studio проводиться таким чином:

1. При старті Visual Studio у стартовому вікні натискаємо кнопку «Create a new project» рис. 2.3.

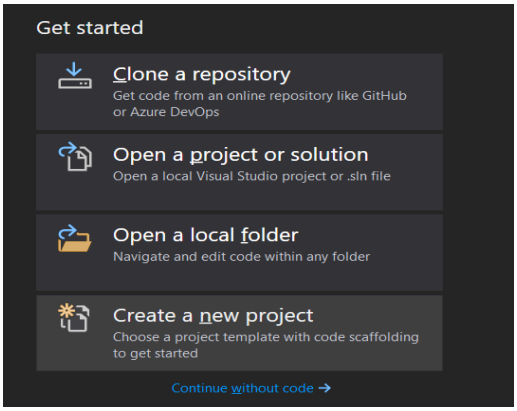

Рисунок 2.3 – Створення нового проекту

2. Далі за потреби обираються фільтри такі як: мова програмування, система під яку ведеться розробка та тип проекту, в нашому випадку – це C#, Android та Mobile відповідно. Із списку варіантів обираємо потрібну нам заготовку під проект – Android App (Xamarin) повністю задовольняє нашим потребам рис. 2.4.

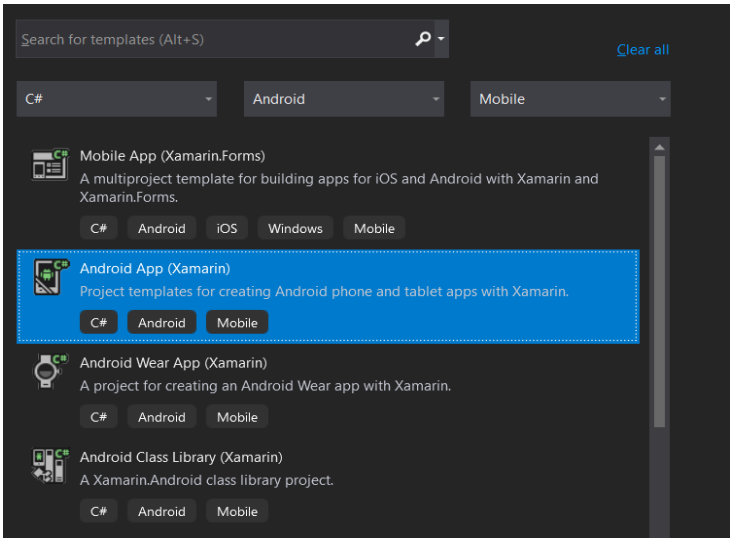

Рисунок 2.4 – Android App (Xamarin)

3. Останніми кроками для створення проекту будуть: назва проекту, місце його розташування на електронному носії та назва рішення рис. 2.5.

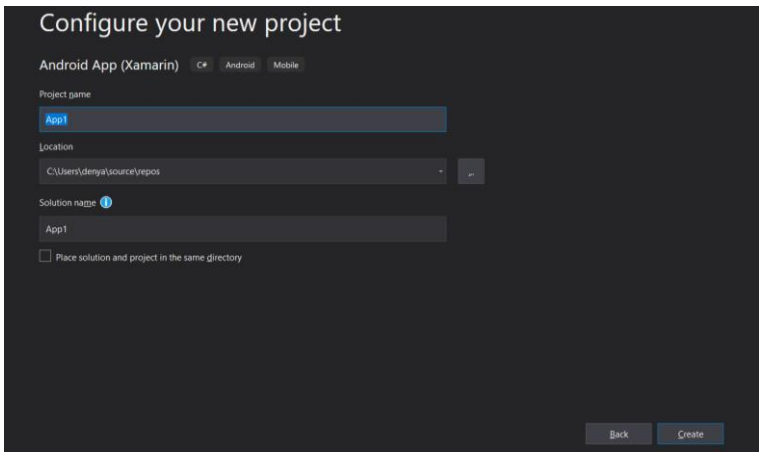

Рисунок 2.5 – Специфікації проекту Visual Studio

2.3. Xamarin

Xamarin – це набір інструментів з відкритим кодом, що дозволяє розробляти різного роду додатки під такі системи як Android, iOS та Windows за допомогою .Net.

Xamarin дає можливість шаблонної розробки додатків під різні системи одночасно, наприклад під iOS та Android. Це дозволяє зменшити кількість проектів, час на розробку, сприяє меншій повторюваності файлів що ідентичні для обох платформ та систематизувати всю інформацію в одному проекті. Також це дозволяє швидше інтегрувати програмний засіб під іншу операційну систему. Хоча дана перевага веде за собою і недолік, оскільки в такому випадку розробник повинен володіти знаннями з розробки на .Net під системи Android та iOS одночасно через різницю в деяких модулях.

Одною з головних функцій даної платформи є компіляція у власний пакет, наприклад .apk файл для Android, чи .ipa для iOS.

Дана платформа працює на основі .Net, що дає можливість користуватись такими речами як автовиділення пам’яті та збір сміття.

Xamarin дозволяє використовувати різного роду файли стилів, вони дозволяють швидко змінювати наприклад колір всього додатку змінивши цей колір лише у файлі стилю. Ці файли мають розширення .xml, та їх вміст написаний за допомогою XML розмітки.

Xamarin дає можливість створення різного інтерфейсу для різних платформ. Також всі об’єкти додатку є спільними для обох платформ, це дозволяє не створювати їх для кожної платформи окремо що дозволяє економити час. Інтерфейс у свою чергу створюється окремо під кожну платформу через особливості розробки під кожну систему. Для кращого розуміння можна

розглянути діаграму, що показує загальну архітектуру кросплатформених додатків Xamarin рис. 2.6.

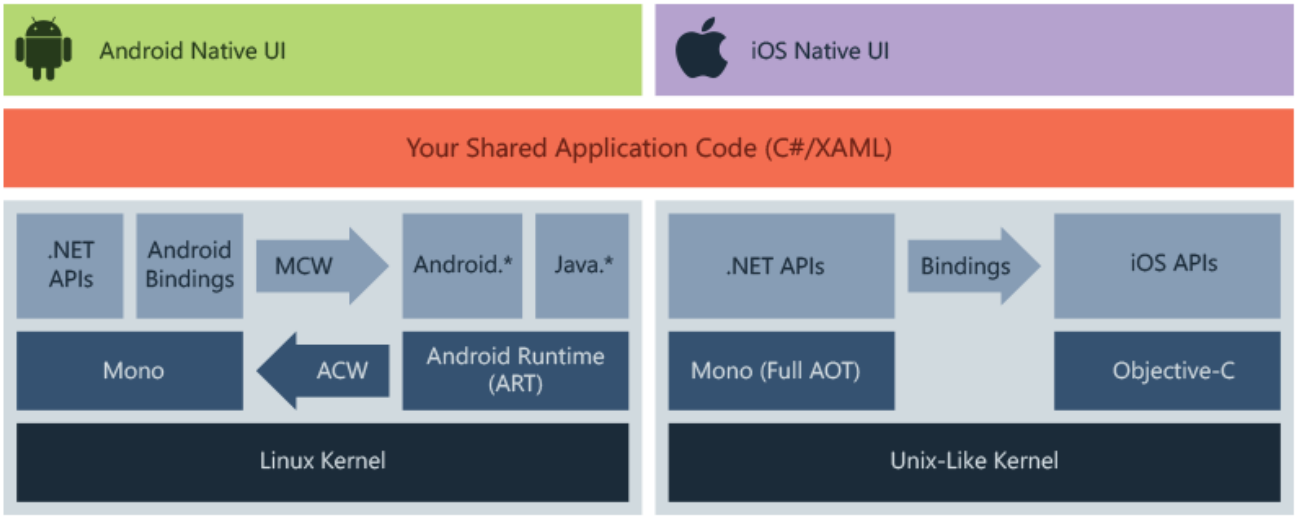

Рисунок 2.6 – Діаграма архітектури кросплатформених додатків Xamarin

Весь інтерфейс програмного засобу також пишеться за допомогою XML розмітки, хоча можна також користуватись конструктором.

Одними із найважливіших бібліотек даної платформи є:

1. Xamarin.Android.
2. Xamarin.iOS.
3. Xamarin.Essentials.
4. Xamarin.Forms.

Xamarin.Essentials

Дана бібліотека дає можливість взаємодії з Android пристроєм. Наприклад:

1. Отримання інформації про пристрій.
2. Отримання доступу до файлової системи.
3. Телефонний набір.

4. Екран блокування
5. Мультимедійне керування.

За великим рахунком – Xamarin.Essentials – це бібліотека, яка дозволяє вам маніпулювати поведінкою пристрою та отримувати від нього різного роду дані.

Xamarin.Android

Додатки Xawmarin.Android компілюються з C# на проміжну мову, яка потім компілюється у самомтійний пакет при запуску програми. Додатки Xamarin.Android працюють у середовищі Mono разом із віртуальною машиною Android Runtime (ART).

Xamarin має інтерфейс для змін у специфікаціях Android проекту, що вказані в Android Manifest, це спрощує роботу з дозволами та дає змогу швидко змінювати специфікації за нагальності, також можна змінювати основну версію операційної системи тим самим оптимізувавши роботу своєї програми під неї рис 2.7.

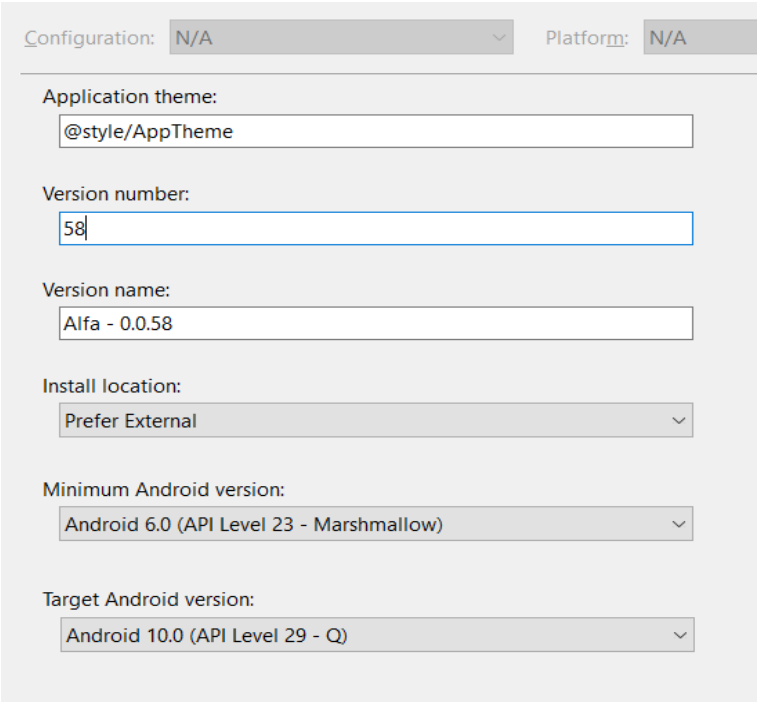

Рисунок 2.7 – Android Manifest

2.4. Мова програмування C#

C# - це обєктно-орієнтована та строго типізована мова програмування, що довзоляє розробникам створювати додатки які працюють у середовищі .Net. Вона має багато схожостей з такими мовами як C, C++, Java та JavaScript оскільки належить до їх сімейства, саме тому для людей котрі вже знайомі з такими мовами – з C# не виникне проблем.

Ця мова програмування дуже популярна, сьогодні її використовують для вирішення широкого спектру завдань, оскільки вона дозволяє створювати додатки під різні системи, такі як Windows, Linux, Android та навіть iOS. Підтримується ця мова компанією Microsoft, тож мова постійно вдосконалюється, під неї розроблюються безліч інструментів при чому не тільки самою компанією Microsoft, а й іншими. Це спрощує користування даною мовою.

У мови програмування C# є доволі велика кількість додаткових функцій, що спрощують життя програмісту. Наприклад:

1. Прибиральник сміття – дана функція допомагає збирати та видаляти об’єкти що уже не використовуються або є недоступними.
2. Допустимі типи значень захищають від випадкової зміни типу даних в змінних.
3. Виключення – функція, що дозволяє краще взаємодіяти з різного роду помилками не завершуючи при цьому роботу програми.
4. Лямбда-вирази – вирази, що дозволяють спростити написання деяких структур у коді та сприяють кращому його розумінню.
5. Асинхронні операції – функціонал, що дозволяє виконувати задані операції у фоновому режимі.

6. Уніфікована система типів даних – у разі необхідності є можливість використання унікальних змінних, які динамічно присвоюють собі тип даних згідно присвоєному об’єкту.

Робота з локальними змінними проходить так само як більшості існуючих мов програмування – за принципом LIFO що розшифровується як Last In First Out – останній вийшов, перший вийшов рис. 2.8. Це дозволяє швидко знищувати змінні, що вже не використовуються.

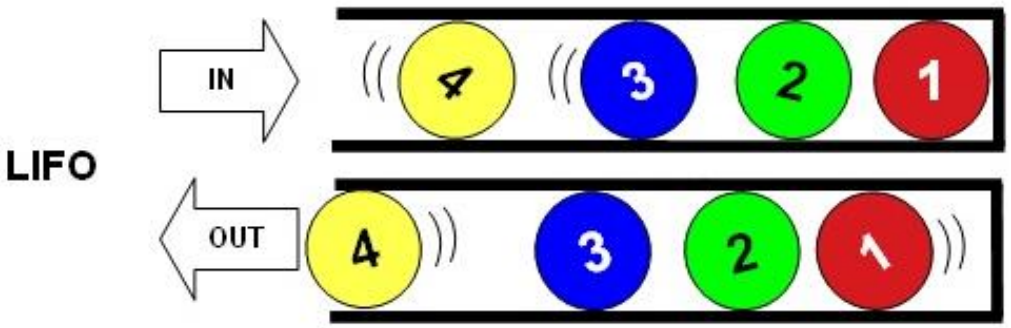

Рисунок 2.8 – принцип роботи LIFO

Важкі конструкції такі як класи, динамічні масиви, словники зберігаються у хмарі задля уникнення ситуацій з переповненням стеку.

2.5. Мова розмітки XAML

Для найпростішої взаємодії з додатком – йому потрібен інтерфейс. Він спрощує користування, систематизує інформацію на екрані та дозволяє краще її сприймати. Для створення інтерфейсу і створена мова розмітки XAML.

eXtensible Application Markup Language (XAML) була розроблена компанією Microsoft на основі XML для створення об’єктів і компонування цих

об’єктів в ієрархії. Вона входила до пакету Windows Presintation Foundation (WPF) але сьогодні він використовується не лише там, а й в Silverlight, Windows Phone та Windows Runtime, Universal Windows Platform (UWP) та у технологіях .Net.

WPF – набір бібліотек, що містять функціонал для створення та запуску додатків, написаних під системи Windows 7 та вище. Даний набір бібліотек допомагає створювати, відображати та керувати інтерфейсами.

XAML дає розробникам можливість визначати користувальницькі інтерфейси в проектах Xamarin.Forms за допомогою розмітки рис. 2.9, тож код використовується рідко. XAML не є обов’язковим у додатках Xamarin.Forms, однак він часто є більш стислим ніж код, також його простіше розуміти. Хоча використання коду дає більше впевненість у розумінні ієрархічної структури інтерфейсу та надає можливість кращого рефакторінгу задля більшої чистоти у коді.

```
<android.support.design.circularreveal.CircularRevealLinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:p4="http://xamarin.com/mono/android/designer"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:minWidth="25px"
    android:minHeight="25px"
    android:weightSum="6" >

    <android.support.v7.widget.Toolbar
        android:id="@+id/toolBar"
        android:layout_width="match_parent"
        android:layout_height="?android:attr/actionBarSize"
        android:minHeight="?android:attr/actionBarSize"
        android:background="■?android:attr/colorPrimary"
        android:elevation="4dp"
        android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar">
        <LinearLayout
            android:layout_width="40dp"
            android:layout_height="40dp">
            <ImageButton
                android:layout_width="40dp"
                android:layout_height="40dp"
```

Рисунок 2.9 – Приклад XAML розмітки

Розробники мають можливість створювати XAML файли у Xamarin.Forms, вони дають можливість визначати користувальницькі інтерфейси з використанням макетів, сторінок та власних класів Xamarin.Forms. Існує можливість скомпілювати або вбудувати XAML виконавчий файл. В любому випадку інформація що знаходиться у XAML файлі аналізується під час побудови проекту для того щоб знайти іменовані об'єкти, повторно дана операція проводиться під час виконання для створення екземплярів, ініціалізації об'єктів та встановлення зв'язку між програмним кодом та даними об’єктами.

Переваги XAML:

1. XAML краще сприймається, ніж код.
2. Ієрархія дочірніх елементів, основа якої була взята з XML, допомагає XAML з більшою чіткістю імітувати таку ж ієрархію об’єктів інтерфейсу.
3. XAML може бути як написаний в ручну програмістами, так і переведений з іншої мови, а також є можливість візуального конструктора рис 2.10.

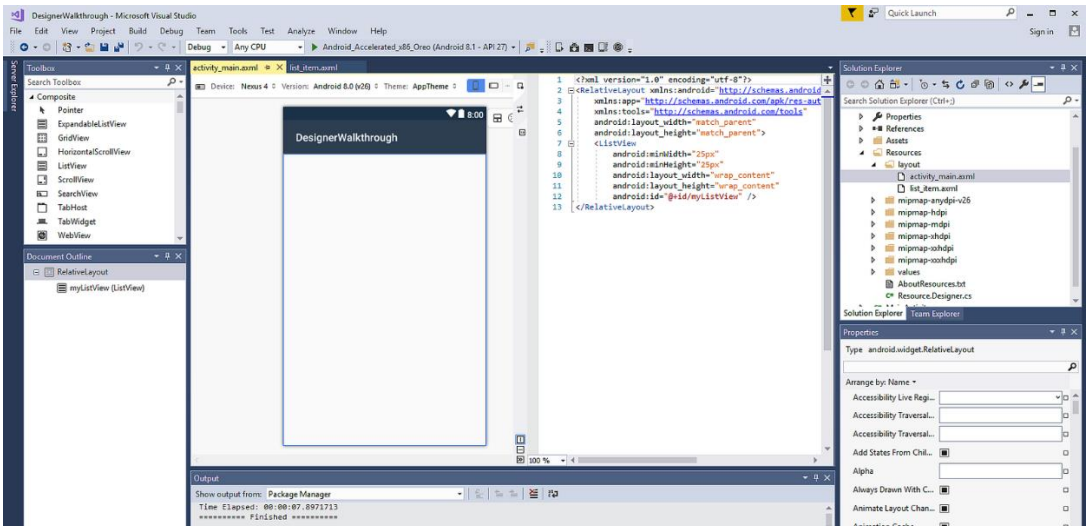

Рисунок 2.10 – XAML designer

Недоліки XAML:

1. Усі обробники подій повинні міститись у .cs файлі, оскільки XAML як мова розмітки не може їх обробляти.
2. Також є неможливим використання циклічних конструкцій.
3. Дана мова не може створювати класів з конструкторами, що приймають параметри.
4. XAML не може викликати на виконання методи класу.

2.6. Android

Платформа Android використовується для мобільних технологій, що в свою чергу забезпечує смартфони та інші мобільні пристрої потужною і мобільною операційною системою Linux, надійністю і гнучкістю стандартної мови високого рівня і інтерфейсу прикладного програмування, а також достатнім списком корисних додатків. Додатки під платформу Android написані переважно на мові Java (використовуючи такі інструменти, як Eclipse і Android Studio), скомпільовані Android API і переведені в байт-код для віртуальної машини Android.

Таким чином, платформа Android має зв'язок з операційними системами, призначеними для інших проектів в основі яких Linux. Мова програмування об'єднує платформу Android з платформою Java ME для телефонів BlackBerry, а також з більшою сферою додатків Java а також платформи Java Enterprise.

Вважається, що сьогодні платформа Android займає величезну частину світового ринку смартфонів, хоча вона не зайняла першості як iPad від фірми Apple на ринку. Кількість продажів змінюються постійно, але зрозуміло, що Android надовго залишиться однією з домінуючих платформ серед мобільного просторіу.

Система Android доступна у вигляді декількох відмінних між собою платформ. Платформа Android Wear реалізує програмування Android в смарт годинниках та подібних аксесуарах для таких додатків, як браслети для контролю спортивної активності. Платформа Android Auto використовується для управління різними пристроями в автомобілях. Android TV як платформа працює в смарт телевізорах і контролерах менш технологічних телевізорів. Також, платформа Android Things спроектована для вбудованих систем, ринок яких тепер відомий як Інтернет речей (Internet of Things - IoT). Усі ці платформи дуже цікава, але, ми використовуватимемо в основному звичайну платформу Android, призначену для мобільних і планшетних додатків.

Архітектура Android ділиться на чотири рівні рис. 2.11:

1. Рівень ядра.
2. Рівень бібліотек і середовища виконання.
3. Рівень каркасу додатків.
4. Рівень додатків.

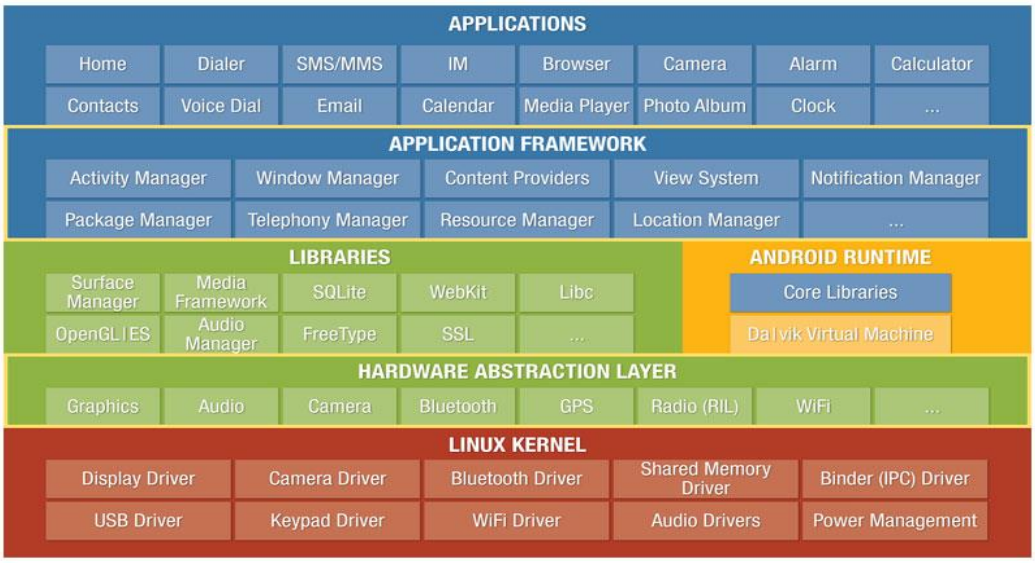

Рисунок 2.11 – Рівні архітектури Android

Система Android - це програмний стек який використовується в мобільних пристроях, включає в себе операційну систему, програмне забезпечення

проміжного шару (middleware), а також основні користувальницькі додатки (календар, карти, контакти та інші).

Рівень ядра.

Ядро є шаром абстракції між обладнанням та частиною програмного стека. Саме в цьому рівні розташовуються основні служби управління процесами, управління файловою системою а також розподілу пам'яті.

Ядро Android базується на ядрі Linux версії 2.6, але сама система Android не являється Linux системою у звичайному вигляді, має відмінності і містить в собі унікальні для Android розширення ядра, - власні механізми взаємодії між процесами, розподілу пам'яті та інші.

Основні компоненти рівня ядра:

1. Драйвер взаємодії між процесами (IPC Driver).
2. Драйвер управління живленням (Android Power Management).
3. Набір драйверів для різного обладнання, що розташовані в мобільному пристрої.

Рівень бібліотек.

Рівень який слідує за ядром Linux являє собою набір бібліотек C / C ++ вигляду WebKit, OpenGL, SSL, FreeType, бібліотеки підтримки libc, бази даних SQLite а також Media Framework. Системна бібліотека базується на Berkeley Software Distribution (BSD) і була розроблена для використання мобільними пристроями які використовують Linux.

Також цей рівень включає набір бібліотек C / C ++, які використовуються різними компонентами ОС. Доступ до різних функцій бібліотек для розробників реалізований через використання Application Framework - каркасу додатків. За своїм функціональним призначенням бібліотеки цього рівня поділяються на наступні групи:

1. Менеджер поверхонь.
2. Системну бібліотеку C.
3. Функціональні бібліотеки C / C ++.

Рівень каркасу додатків.

Рівень каркасу додатків знаходиться на вершині функціональних а також системних бібліотек. На цьому рівні знаходяться основні служби Android для управління пакетами, ресурсами, життєвим циклом додатків та іншим.

Програміст володіє повний доступом до усіх API, які використовуються основними додатками. Саме метою такої архітектури додатків було спрощення багаторазового використання компонентів. Довільна розроблювана програма може використовувати можливості базових додатків і, будь-який сторонній додаток має змогу використовувати можливості вашого додатку (з урахуванням встановлених дозволів). Такий механізм дозволяє багаторазово використовувати вже готові компоненти.

Рівень додатків.

Мобільний пристрій Android одразу має набір основних додатків, включаючи в себе програму для роботи з SMS, календар, навігаційні карти, браузер, контакти та ще багато інших.

Важливо підмітити, що платформа Android не робить різниці між основними додатками телефону і стороннім програмним забезпеченням – саме тому основні програми, що входять в стандартний набір програмного забезпечення, можна замінити на будь-які альтернативні програми. Додатки для Android пишуться в основному на мові Java.

Перевагою при розробці додатків на Android програмісти мають повний доступ до всієї функціональності операційної системи. Архітектура додатків побудована так, що дає легкість використання основних компонентів, які

надаються системою. Корисною також є можливість створювання своїх компонентів і надання їх у відкрите використання.

2.7. GIT

Багато людей використовують системи контролю версій, оскільки копіювати проект у сусідню папку кожен раз коли хочеш внести зміни – доволі незручно. Тим паче що не виключена ймовірність просто заплутатися у створених копіях, навіть цілком ймовірно, що згодом розробник просто заплутається. Для того щоб цього уникнути і були створені системи контролю версій.

GIT – це система контролю версій, що допомагає розробникам відстежувати усі зміни, що відбуваються у проекті, зберігати версії проекту та легко ними маніпулювати.

У ній використовується система так званих гілок. Кожна гілка – це окрема копія проекту, що працює незалежно від своїх аналогів у інших гілках, це дає можливість вносити зміни та тестувати програмний засіб без страху його повністю зламати, а у разі вдалого внесення змін – прийняття їх до проекту.

Тож зазвичай існує декілька гілок, найчастіше вони звуться: master та develop. Де master – гілка з працюючим та відлагодженим програмним засобом, а develop – гілка для впровадження та тестування нових функцій.

У разі вдалого впровадження змін – є можливість злиття гілок. При злитті декількох гілок – всі файли, що не є спільними – просто зберігаються, а файли зі змінами замінюють собою старі.

Також існує можливість відкату, що допомагає не вратити проект у разі помилки. І звісно ж є можливість у любий момент використовувати усі попередні версії проекту.

Для того щоб додати новий функціонал додатку не вплинувши на новий – можна створювати розгалуження гілок, при цьому процесі створюється нова гілка з ідентичною копією проекту.

Взагалі – дана система контролю версій використовується для синхронізації проекту, коли над ним працюють декілька чоловік, це дає змогу синхронізації проекту в цілому, дає можливість роботи над різними її модулями одночасно та завжди підтримувати його актуальну версію. Саме тому її часто використовують у компаніях. Також не малу роль відіграє те, що даний інструмент – безкоштовний.

У даного програмного засобу є можливість також зберігати версії локально, це дуже корисно для людей що працюють над проектом самі, це спрощує взаємодію з версіями та прибирає плутаницю з голови.

На превеликий жаль – GIT працює у режимі консолі та сприймає тільки команди, але існують програмні засоби, що створюють приємний інтерфейс та спрощують взаємодію з системою контролю версій такі як: SourceTree, SVN, Bazaar та інші.

2.8. SourceTree

SourceTree – це Git клієнт, що доступний для платформ Windows та iOS. Дане програмне забезпечення є абсолютно безкоштовним. Він доволі зручний для новачків, оскільки має інтуїтивно зрозумілий інтерфейс рис. 2.12, що поєднує користувача з Git репозиторієм.

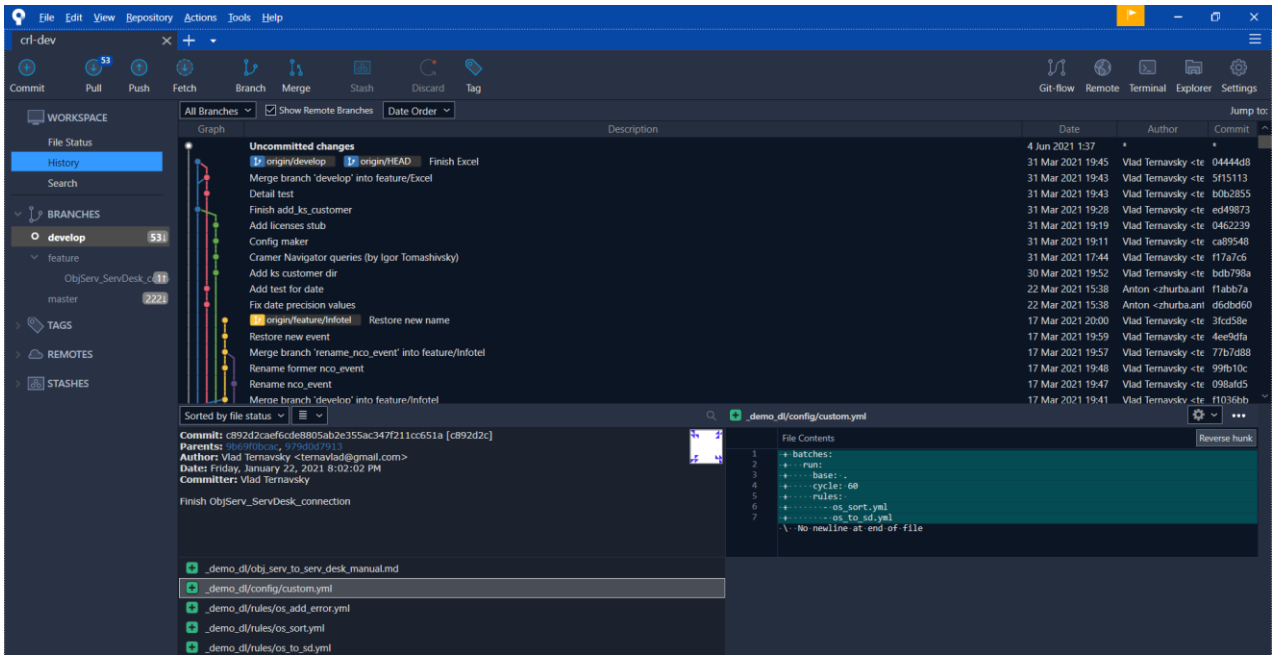

Рисунок 2.12 – Інтерфейс програми SourceTree

Використовуючи даний програмний засіб не обов’язково знати команди GIT, оскільки тут є набагато зручніші контролери, що зроблять все швидше.

Головною перевагою цього додатку є інтерфейс, оскільки ми сприймаємо інформацію набагато краще коли бачимо з чим маємо справу.

Хоча даний програмний засіб – це не лише інтерфейс для системи контролю версій, він містить також декілька цікавих функцій:

1. Можна отримувати завдання для роботи над ним.
2. Візуально зображені гілки, вони допомагають краще розуміти ситуацію в проекті.
3. Також є можливість взаємодії з великими файлами, оскільки SourceTree підтримує технологію Git Large File.
4. Для взаємодії з гілками не потрібно виходити з додатку, усі можливі взаємодії можна проводити у ньому ж.
5. Даний програмний засіб містить функцію перебазування даних, він робить коміти чистішими та більш зрозумілими.

6. Можливо здійснювати керування проектами за допомогою підмодулів. Є можливість групувати їх, встановлювати залежності тощо.

Тож програма SourceTree дозволяє взаємодію з Git без написання команд. Здійснювати це їй допомагають такі кнопки, як: Commit, Pull, Push, Branch, Merge, Git-flow.

Commit – кнопка, що дозволяє зберегти зміни для обраних файлів в активну гілку, обираються файли для коміту, є можливість написання коментаря рис. 2.13.

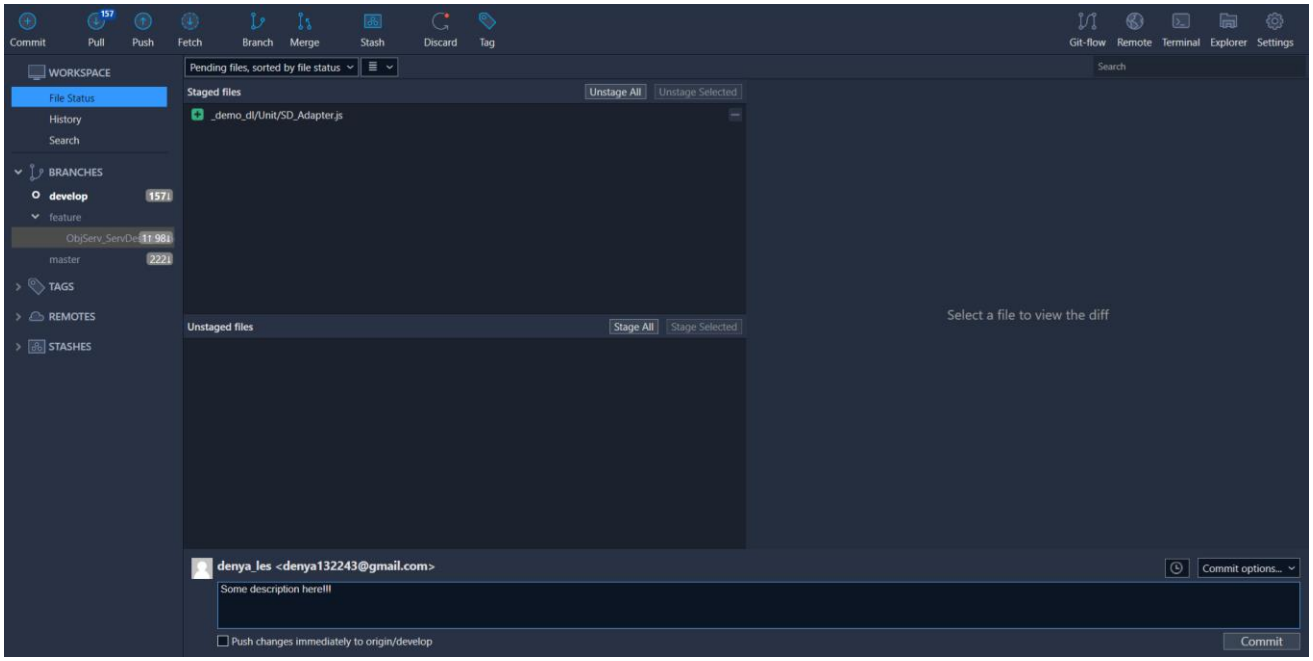

Рисунок 2.13 – SourceTree Commit

Pull – дозволяє оновити копію проекту з гілки до пристрою рис. 2.14.

При цьому є можливість обрати гілку з якої проводити копіювання та додано перелік додаткових опцій, що можуть стати в нагоді.

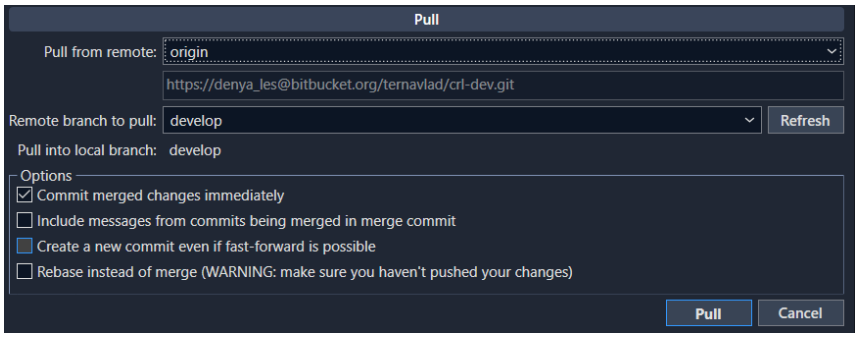

Рисунок 2.14 – SourceTree Pull

Push – відправляє зміни на сервер та вносить зміни до обраної гілки.

При виконанні даної команди – потрібно обрати галочкою локальну гілку та гілку реозиторія у випадаючому списку до якої будуть вноситись зміни. Є можливість проводити дану операцію для декількох гілок одразу рис. 2.15.

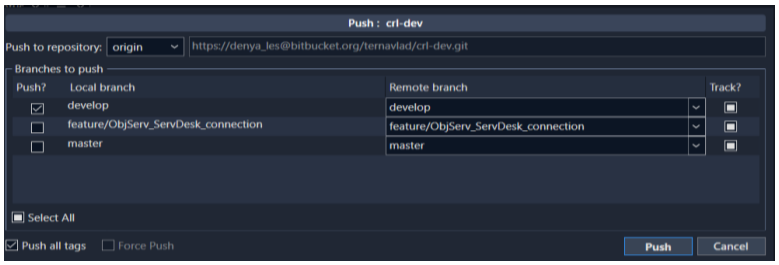

Рисунок 2.15 – SourceTree Push

Branch – створює копію обраної версії проекту у новій гілці.

При виборі даної функції з’являється вікно, що запитує назву нової гілки, також дозволяє одразу створити специфічний коміт якщо це необхідно рис. 2.16.

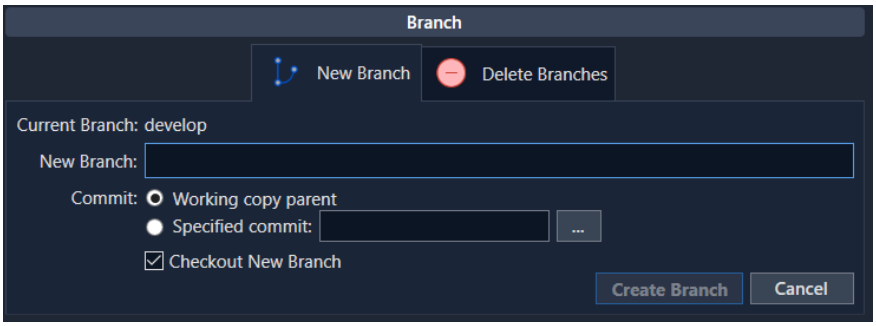

Рисунок 2.16 – Створення нової гілки

Merge – проводить злиття двух гілок.

Усі файли, що не були спільними для обох гілок просто зберігаються точніше зберігаються посилання на них із попередніх версій проекту, а спільні – замінюються файлами із гілки яку зливають, до гілки у яку зливають. У даній ситуації обирається гілка та конкретна версія проекту рис. 2.17.

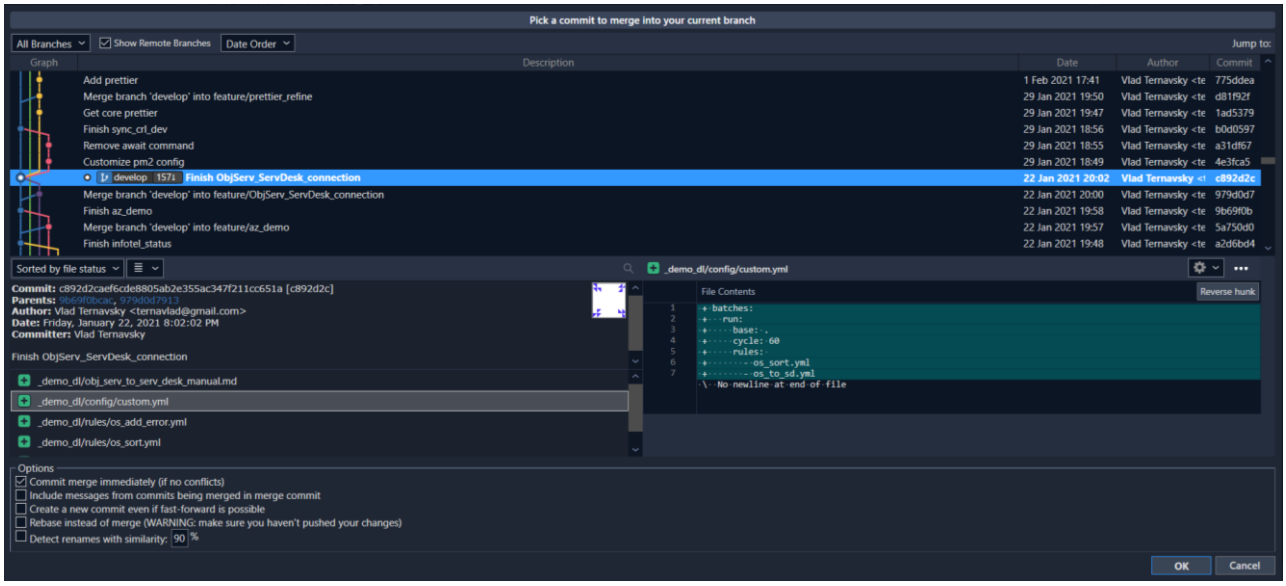

Рисунок 2.17 – Злиття гілок

РОЗДІЛ 3. РОЗРОБКА ПРОГРАМНОГО ЗАСОБУ

3.1. Проектування

При розробці додатку важливо уявляти кінцевий результат. Це допомагає випадково не почати розробку непотрібного функціоналу тим самим забираючи час на розробку. Для кращого уявлення взаємодії та архітектури майбутнього додатку використовують різного роду діаграми. Наприклад діаграма взаємодії рис. 3.1.

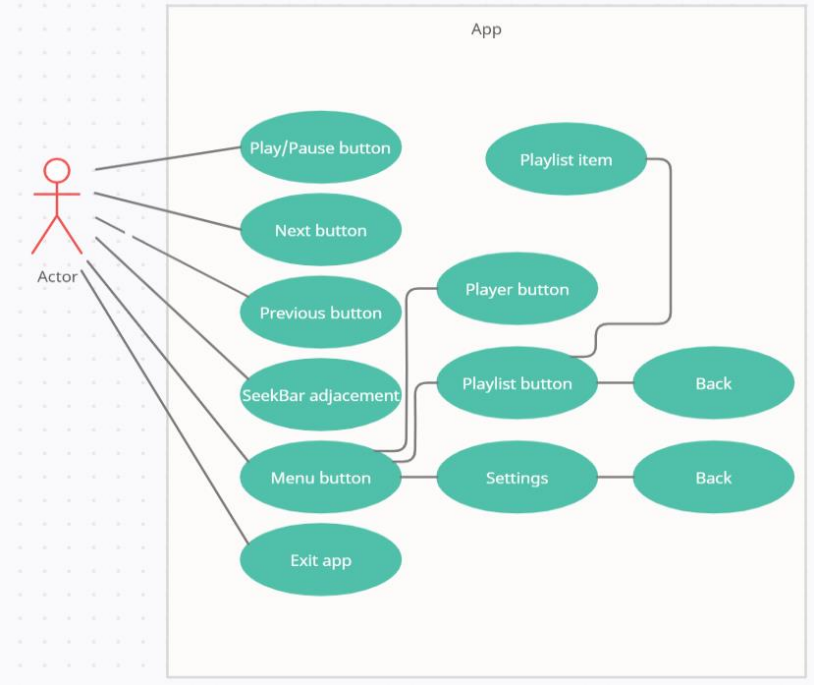

Рисунок 3.1 – UML діаграма варіантів використання

Також задля кращого розуміння можна визначитися з класами, що будуть використані у ході розробки. Для плеєра накрищим варіантом буде виділити аудіофайли як окремий клас, це дозволить краще деталізувати пісні, оскільки клас

дає можливість зберігати та швидко застосовувати данні без їх вилучення кожного разу. Наприклад можна створити змінну жанру, де буде зберігатися назва для кожної пісні окремо, а в разі необхідності, наприклад при сортуванні – швидко застосовуватись.

3.2. Розробка додатку

Для розробки додатку завантажуємо та встановлюємо Visual Studio Community 2019 та інші інструменти для розробки мобільних додатків. Для цього перейдемо на офіційний сайт Microsoft рис. 3.2.

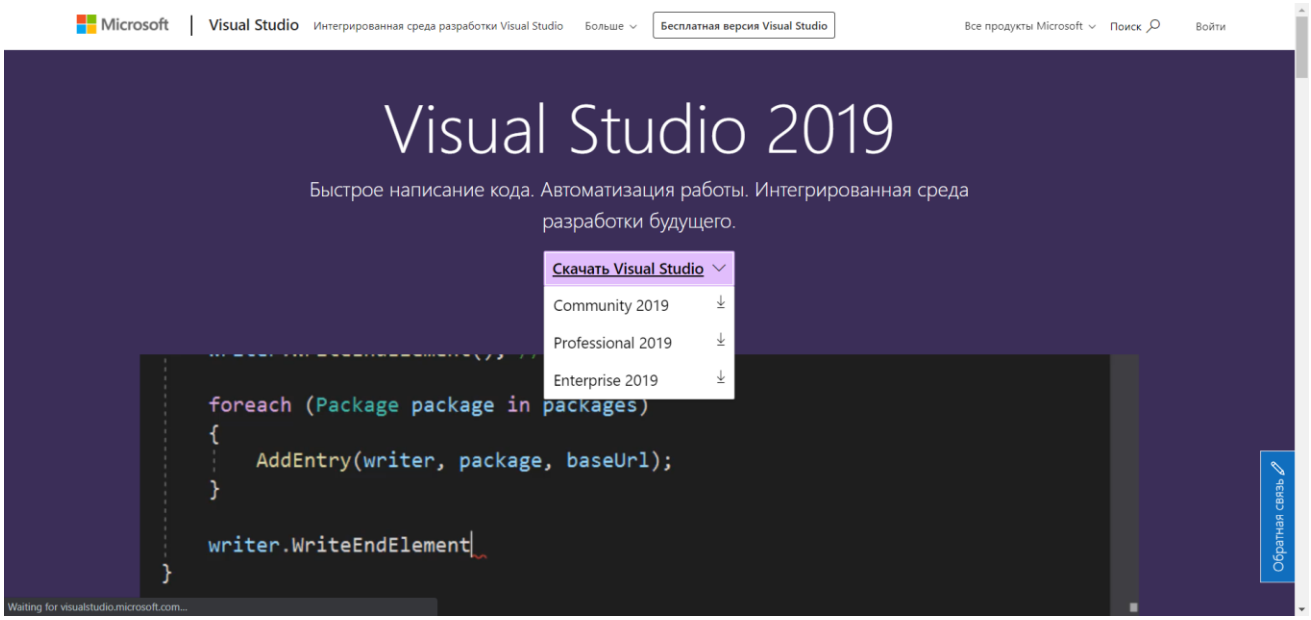

Рисунок 3.2 – Завантаження Visual Studio Community 2019

При встановленні даного програмного засобу – одразу обираємо галочки біля Mibile development with .NET, даний пакет включає в себе підтримку мови програмування C# та інструмент розробки Xamarin рис. 3.3. Натискаємо Install та очікуємо встановлення усіх програмних засобів.

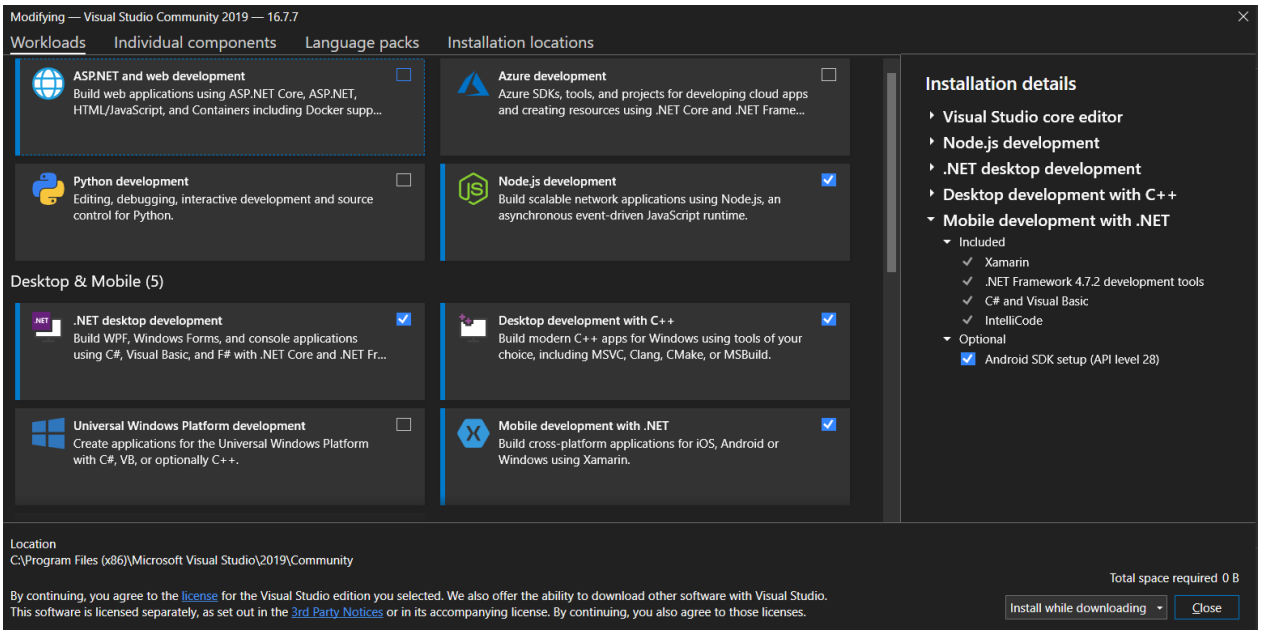

Рисунок 3.3 – Додатковий набір інструментів Visual Studio Community 2019

Далі потрібно створити проект як це було описано у підрозділі 2.3. Натискаємо Create a new project -> Android App (Xamarin) -> даємо назву проекту та визначаємо його місце зберігання -> натискаємо Create -> обираємо один із запропонованих шаблонів та мінімальну підтримувану версію Android рис. 3.4.

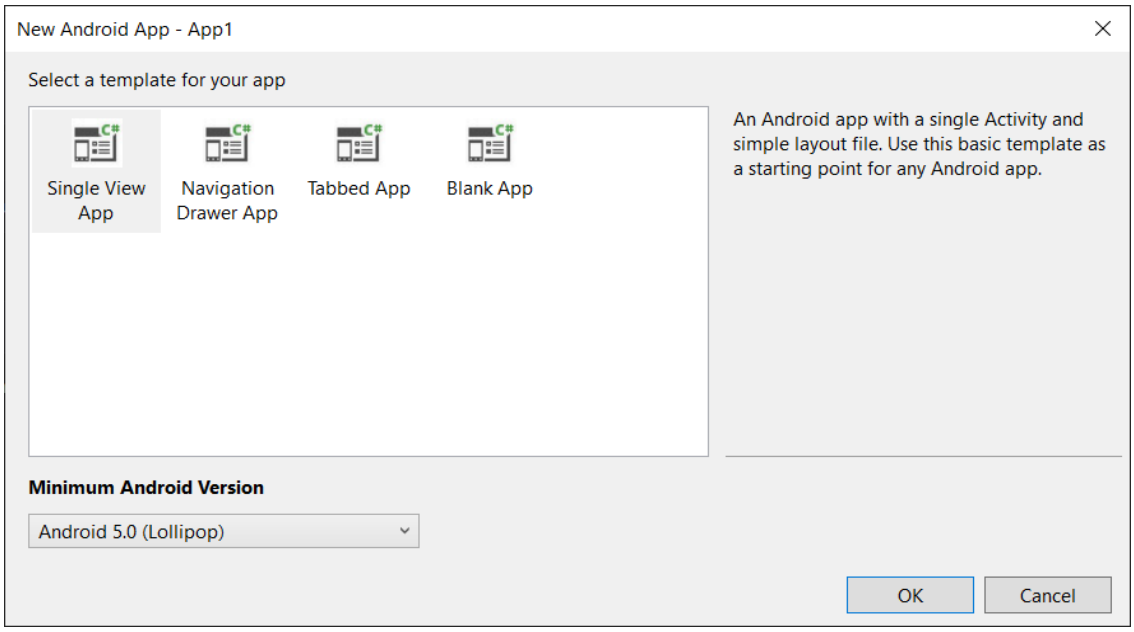

Рисунок 3.4 – Стандартні шаблони додатку Xamarin

Найбільше нам у нашому випадку підходить Single View App. Було створено початковий додаток Xamarin, що буде використаний як основа для написання подальшого функціоналу нашої програми.

Основними файлами для нас є activity_main.xml, де ми будемо писати розмітку інтерфейсу нашого додатку рис.3.5, також основний файл для написання коду – MainActivity.cs, де будуть обробники подій та реалізація інших функціональних можливостей додатку рис 3.6.

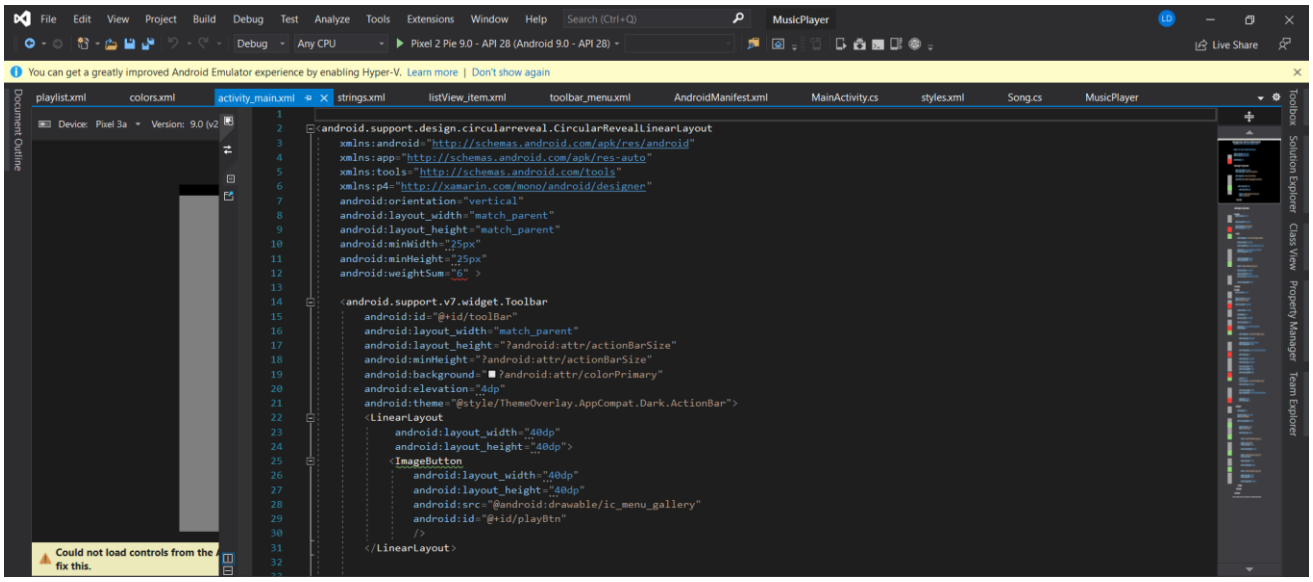

Рисунок 3.5 – Файл activity_main.xml

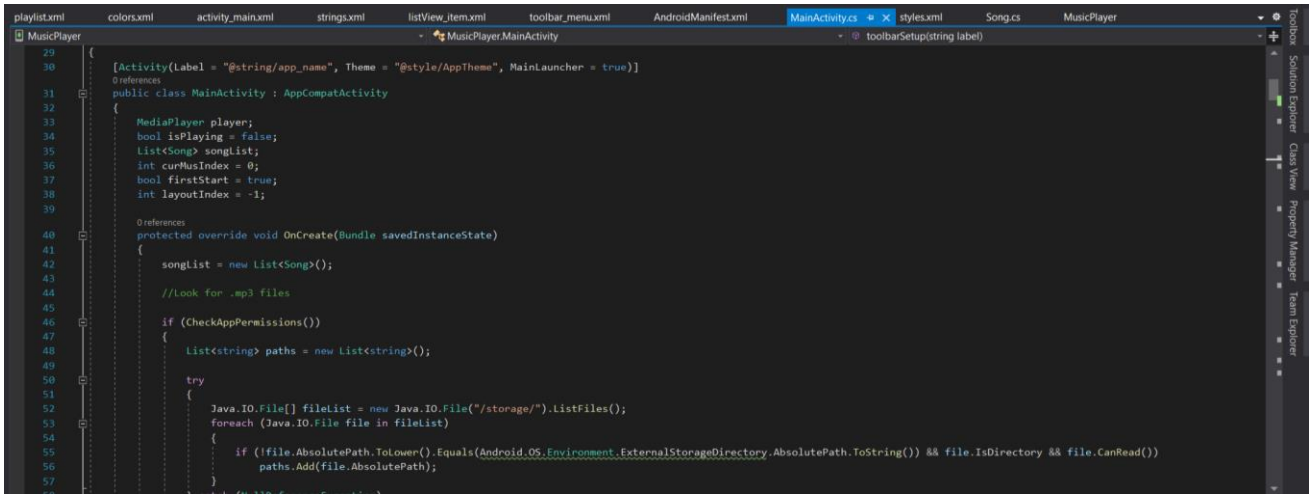

Рисунок 3.6 – Файл MainActivity.cs

Для того, щоб зберегти та використати у проекті різного роду додаткові бібліотеки – можна скористатися References у проекті. Там міститься вся інформація щодо бібліотек проекту. Нас цікавить наявність таких додаткових бібліотек як:

1. Xamarin.Android.Support.Design.
2. Xamarin.Android.Support.v7.AppCompat.
3. Xamarin.Essentials.
4. Xamarin.Forms.

Тепер за допомогою конструктору, або вручну можна розпочинати розробку інтерфейсу додатку у файлі activity_main.xml.

Для написання інтерфейсу даного додатку було обрано варіант з написанням мови розмітки вруну, оскільки це дає можливість поглибити свої знання у даній сфері та зрозуміти принцип створення ієрархічного інтерфейсу за допомогою мови розмітки.

Для того, щоб користувач мав можливість взаємодії з плеєром – до інтерфейсу були додані такі елементи керування:

1. Кнопка Play/Pause (Грати\Пауза).
2. Кнопка Next (Наступний).
3. Кнопка Previous (Попередній).
4. Шапка додатку з його назвою та меню - Toolbar.
5. Повзунок Seekbar.
6. Текстові поля для відображення поточного та повного часу треку.
7. Текстове поле для відображення назви файлу, що програється.
8. Картинка треку.

В результаті при створенні ієрархічної розмітки маємо отримати інтерфейс головної сторінки плеєра, що буде містити досить засобів керування, що

дозволить користувачеві взаємодіяти з аудіофайлами та буде приємним у користуванні рис. 3.7.

Рисунок 3.7 – Інтерфейс додатку MusicPlayer

Тож тепер у нас є головна сторінка нашого плеєру, також йому потрібен інтерфейс з плейлістом, тому створимо ще один .xml файл та назвемо його playlist.xml.

Доки що у ньому нічого не буде, оскільки ще не реалізований пошук аудіо файлів, але для відображення плейлісту нам потрібен ListView. Даний елемент відображає елементи масиву об’єктів у вигляді списку та задаємо йому базову розмірність рис. 3.8.

```
31    <ListView
32        android:minWidth="25px"
33        android:minHeight="25px"
34        android:layout_width="match_parent"
35        android:layout_height="match_parent"
36        android:id="@+id/musicList"/>
37
```

Рисунок 3.8 – Елемент для відображення плейлісту

Тепер за наявності необхідного інтерфейсу можна розпочинати написання функціоналу.

Розпочати потрібно із створення класу Song рис. 3.9, до дозволить краще систематизувати дані про треки, даний клас буде містити таку інформацію:

1. Назву треку без розширення.
2. Дорогу до файлу у сховищі.
3. Посилання наступний трек.
4. Посилання на попередній трек.
5. Конструктор, що дозволить створювати екземпляр даного класу та вносити дані до вище вказаних полів.

```
15  class Song
16  {
17      public string name;
18      public string path;
19      public Song next;
20      public Song prev;
21
        1 reference
22      public Song(string name, string path, Song next = null, Song prev = null)
23      {
24          this.name = name;
25          this.path = path;
26          this.next = next;
27          this.prev = prev;
28      }
```

Рисунок 3.9 – Клас Song

Далі потрібно налаштувати файл Android Manifest вказавши у ньому такі дозволи: READ_EXTERNAL_STORAGE та WRITE_EXTERNAL_STORAGE. Тепер потрібно описати функцію, що буде запитувати дозвіл на використання запам’ятовуючого пристрою. Спочатку перевіряється версія Android, якщо вона нижча за 5 – то запитувати дозвіл не потрібно, оскільки від автоматично надається користувачем при завантаженні додатку з Play Market. Далі вже перевіряються надані дозволи, якщо їх не надали – можна запитати їх за допомогою функції RequestPermissions до якої передаються список потрібних дозволів та створюється код успішності операції. Це робиться за допомогою коду на рис. 3.10.

```
113         private bool CheckAppPermissions()
114         {
115             if ((int)Build.VERSION.SdkInt < 23)
116             {
117                 return true;
118             }
119             else
120             {
121                 if (PackageManager.CheckPermission(Manifest.Permission.ReadExternalStorage, PackageName) != Permission.Granted
122                     && PackageManager.CheckPermission(Manifest.Permission.WriteExternalStorage, PackageName) != Permission.Granted)
123                 {
124                     var permissions = new string[] { Manifest.Permission.ReadExternalStorage, Manifest.Permission.WriteExternalStorage};
125                     RequestPermissions(permissions, 77);
126                 }
127                 if (PackageManager.CheckPermission(Manifest.Permission.ReadExternalStorage, PackageName) != Permission.Granted
128                     && PackageManager.CheckPermission(Manifest.Permission.WriteExternalStorage, PackageName) != Permission.Granted)
129                 {
130                     return false;
131                 }
132                 else return true;
133             }
134         }
```

Рисунок 3.10 – Функція CheckAppPermissions

```
if (CheckAppPermissions())
{
    List<string> paths = new List<string>();

    Java.IO.File[] fileList = new Java.IO.File("/storage/").ListFiles();
    try
    {
        foreach (Java.IO.File file in fileList)
        {
            if (!file.AbsolutePath.ToLower().Equals(Android.OS.Environment.ExternalStorageDirectory.AbsolutePath.ToString()) && file.IsDirectory && file.CanRead())
                paths.Add(file.AbsolutePath);
        }
    }
    catch (NullReferenceException) { }

    paths.Add(Android.OS.Environment.ExternalStorageDirectory.AbsolutePath.ToString());
    foreach (string path in paths)
    {
        findMP3(path);
    }
```

Рисунок 3.11 - Пошук директорій з файлами

Після виконання роботи даною функцією – можна шукати аудіо файли, що містяться на девайсі. Для цього ще раз проведемо перевірку на надання дозволів та скористаємося бібліотеками Java, оскільки дана мова краще взаємодіє з носіями Android пристроїв. Також при її використанні у даному модулі – підвищується швидкодія додатку. Для взаємодії з файловим сховищем Java має багато функцій. Для початку – беруться усі файли та папки зі сховища рис. 3.11, після чого вони перевіряються на можливість читання, у разі успіху – їх розміщення додається до масиву. Далі вже запускається функція пошуку саме аудіо файлів що рекурсивно

перевіряє наявність .mp3 файлів у кожній піддерикторії, створює для них об’єкт класу Song та додає їх до глобальної змінної songList рис. 3.12.

```
2 references
public void findMP3(string path)
{
    //Looking for music in directory
    try
    {
        string[] songs = Directory.GetFiles(path, "*.mp3");
        foreach (string song in songs)
        {
            songList.Add(new Song(Path.GetFileName(song).Replace(".mp3", ""), song));
        }


        // Loocking for music in subdirectories

        string[] subDirectories = Directory.GetDirectories(path.ToString());
        foreach (string file in subDirectories)
        {
            findMP3(file);
        }

    } catch (Exception ex){
        Log.Error("MusicPlayer", ex.Message);
    }

}
```

Рисунок 3.12 – Пошук файлів з розширенням .mp3

Далі відображаємо головну сторінку плеєру, та присвоюємо елементам інтерфейсу відповідні іконки, що були заздалегідь створені у Adobe Photoshop.

Тепер маючи список аудіо файлів на пристрої ми маємо змогу працювати над обробниками подій, що викликаються при взаємодії з елементами інтерфейсу. Створимо обробники подій на такі елементи як: Play/Pause, Next, Previous, SeekBar.

Play/Pause.

Для глобальної взаємодії з любого роду медіафайлами на Android системі існує бібліотека Android.Media, що містить потрібний нам клас MediaPlayer. Даний клас має можливості до програвання та відтворення аудіо та відео файлів, містить функціонал, що дає можливість регулювати звук на пристрої,

перемотувати та зупиняти медіафайл. Для програвання медіа – він може отримувати шлях до медіафайлу, також це може здійснюватися через Uri чи навыть з папки res/raw проекту. Також у ньому присутній функціонал для зациклення відтворюваного файлу.

Тож для програвання музики – ми присвоюємо об’єкту класу MediaPlayer шлях до .mp3 файлу та викликаємо метод класу під назвою Start(). Для зупинки існує метод Pause(), або Stop(). Ці два методи відрізняються тим, що Stop() окрім того, що зупиняє, ще повертає трек на початок, а Pause() лише зупиняє його.

Для розуміння що кнопці потрібно робити застосовується змінна класу MediaPlayer – isPlaying. Вона містить бульове значення правдиве, або хибне. Виходячи із цього можна застосовувати метод, що програє або зупиняє трек рис 3.13.

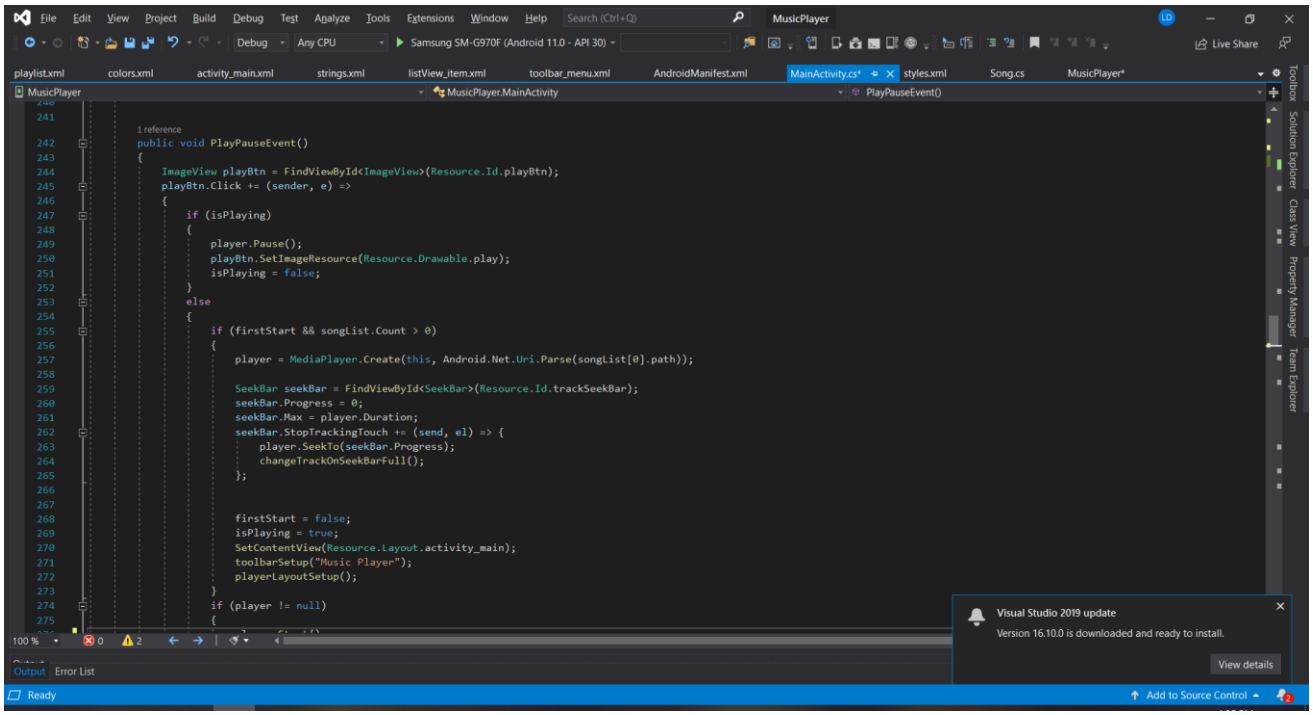

Рисунок 3.13 – Обробник події Play/Pause

Звісно ж при написані враховуються різні можливості виникнення виключень та можливі збої при роботі наприклад спроба запустити трек без його наявності. Усі ці події також враховуються та створені додаткові перевірки, що запобігають цьому.

Next

Дана клавіша допомагає користувачеві переміщуватись далі по плейлісту.

Є два способи пересування по плейлісту:

1. За допомогою індексатора.
2. За посиланнями у класі Song.

За звичайних умов переміщення між треками здійснюється за допомогою індексатора, це дозволяє не запам’ятовувати послідовність медіафайлів у плейлисті. Оскільки наш плейліст – це масив аудіофайлів, що мають свої індекси, перехід до наступного треку відбувається зміною індексатора на одиницю більше, проводиться нове присвоєння шляху об’єкту MediaPlayer та викликається метод Start() у функції PlayMusic() рис. 3.14. Також при зміні треку відбувається оновлення SeekBar та присвоєння текстовим полям що вказують на довжину треку нових значень.

```
public void goRightEvent()
{
    ImageView goRightBtn = FindViewById<ImageView>(Resource.Id.goRightBtn);
    goRightBtn.Click += (sender, e) =>
    {
        if (curMusIndex < songList.Count - 1)
        {
            curMusIndex++;
            PlayMusic();
            FindViewById<TextView>(Resource.Id.trackName).Text = songList[curMusIndex].name.ToString();
            FindViewById<TextView>(Resource.Id.TrackTime).Text = GetTrackDuration(player.Duration);
            FindViewById<SeekBar>(Resource.Id.trackSeekBar).Progress = 0;
            FindViewById<SeekBar>(Resource.Id.trackSeekBar).Max = player.Duration;
        }
        else if (curMusIndex >= songList.Count - 1)
        {
            curMusIndex = 0;
            PlayMusic();
            FindViewById<TextView>(Resource.Id.trackName).Text = songList[curMusIndex].name.ToString();
            FindViewById<TextView>(Resource.Id.TrackTime).Text = GetTrackDuration(player.Duration);
            FindViewById<SeekBar>(Resource.Id.trackSeekBar).Progress = 0;
            FindViewById<SeekBar>(Resource.Id.trackSeekBar).Max = player.Duration;
```

Рисунок 3.14 – Обробник події Next

Єдиним недоліком даного методу є постійна перевірка на останній елемент у масиві, щоб не вийти за його рамки.

Previous.

Обробник даної події реалізований аналогічно попередньому, за вийнятком того, що індексатор змінюється на одиницю меньше.

SeekBar.

Елемент інтерфейсу під назвою SeekBar – це об’єкт що допомагає користувачеві перемотувати аудіо доріжку до моменту що йому потрібно.

Реалізований даний обробник події не через подію натискання як усі інші інтерактивні елементи інтерфейсу, а через подію зміни стану користувачем. При перемотці треку SeekBar приймає цілочисельне значення що відповідає мілісекундам. Далі отримані мілісекунди присвоюються об’єкту класу MediaPlayer за допомогою методу SeekTo(). Цей метод приймає цілочисельні значення, що відповідають мілісекундам, тож немає проблеми присвоїти треку значення SeekBar рис. 3.15.

```
seekBar.StopTrackingTouch += (sender, e) => {
    player.SeekTo(seekBar.Progress);
    changeTrackOnSeekBarFull();
};
```

Рисунок 3.15 – Обробник події перемотки треку

Також при програванні аудіо – для елементу SeekBar створюється додатковий потік, який у фоні змінює його значення кожну секунду щоб відповідати позиції треку. При зміні треку йому цьому елементу присвоюєть нове максимельне значення що відповідає довжині треку.

Далі можна працювати над меню та додавання функціональності сторінки плейлістів. Меню додається у шапку додатку. Для цього потрібно створити файл toolbar_menu.xml до якого додамо усі елементи що потрібні нам у меню рис. 3.16.

```xml
<?xml version="1.0" encoding="utf-8" ?>
<menu
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto" >

<item
     android:id="@+id/menu_Player"
     app:showAsAction="never"
     android:title="Player"/>

<item
    android:id="@+id/menu_Playlist"
    app:showAsAction="never"
    android:title="Playlist"/>

<item
    android:id="@+id/menu_Settings"
    app:showAsAction="never"
    android:title="Settings"/>
</menu>
```

Рисунок 3.16 – Елементи меню у шапці додатку

Перезаписуємо функцію OnCreateOptionMenu для того, щоб воно відображалось на тулбарі.

Також нам потрібен обробник подій при виборі одного із елементів меню.

Тому перезаписуємо подію OnOptionsItemSelected у якій визначаємо зміну шаблонів інтерфейсу відповідно до пунктів у меню.

Для відображення списку знайдений аудіофайлів було створено ще одну функцію addSongsToView() рис. 3.17.

```
public void addSongsToView()
{
    List<string> tmp = new List<string>();
    if (songList.Count > 0)
    {
        foreach (Song song in songList)
            tmp.Add(song.name.ToString());
        ListView trackList = FindViewById<ListView>(Resource.Id.musicList);
        ArrayAdapter<string> adapter = new ArrayAdapter<string>(this, Resource.Layout.listView_item, tmp);
        trackList.Adapter = adapter;
        trackList.ItemClick += musicSelected;
    }
}
```

Рисунок 3.17 – Відображення списку файлів у плейлисті

Вона також містить обробник подій при обиранні певного треку зі списку. Це зроблено також за допомогої ідексаторів, оскільки індекс файлу у списку

відповідає індексу файлу у масиві та одразу викликається функція PlayMusic() рис 3.18.

```
public void musicSelected(object sender, AdapterView.ItemClickEventArgs e)
{
    curMusIndex = e.Position;
    isPlaying = true;
    if (firstStart)
    {
        firstStart = false;
        player = MediaPlayer.Create(this, Android.Net.Uri.Parse(songList[curMusIndex].path.ToString()));
    }
    PlayMusic();
}
```

Рисунок 3.18 – Обробник події при обиранні треку з плейлиста

РОЗДІЛ 4. ТЕСТУВАННЯ

4.1. Тестування

Для забезпечення стабільної роботи додатку без збоїв та відхилень від очікуваного результату було розроблено тест-кейси, що допомагали б виявляти невідповідності у роботі додатку.

Тест №1.

Перевірка на відсутність збоїв при відхиленні користувачем дозволу на використання сховища пристрою.

Кроки для перевірки:

1. Відхилення дозволу до використання сховища пристрою.
2. Відкриття сторінки плейлисту.

Очікування: збою та вильоту з додатку не відбулося.

Результат: збою та вильоту з додатку не відбулося.

Тест №2.

Перевірка на відсутність збоїв при натисканні інтерактивних клавіш інтерфейсу при відсутності та з наявністю аудіо файлів.

Кроки для превірки:

1. Відхилення дозволу на використання сховища пристрою.
2. Натискання клавіш Play/Pause, Next, Previous та прокрутка SeekBar.

Очікування: за відсутності аудіофайлів при натисканні клавіші нічого не відбувається, за наявності – починає грати музика.

Результат: за відсутності аудіофайлів при натисканні клавіші нічого не відбувається, за наявності – починає грати музика.

Тест №3.

Перевірка на зміну треку при заповненні SeekBar на 100%.

Кроки для перевірки:

1. Надання дозволу на використання сховища пристрою.
2. Натискання клавіши Play/Pause.
3. Перемотка SeekBar до кінця треку.

Очікування: після скінчення граючого треку грає наступний.

Результат: після скінчення граючого треку грає наступний.

Тест №4.

Перевірка інтерфейсу на гнучкість за допомогою запуску додатку на різних пристроях.

Кроки для перевірки:

1. Встановлення додатку на пристрій.
2. Відкриття додатку.

Очікування: інтерфейс підлаштовується під розширення екрану пристрою.

Результат: інтерфейс підлаштовується під розширення екрану пристрою.

Тест №5.

Перевірка на відсутність втрати контролю над треком при згортанні та розгортанні додатку.

Кроки для перевірки:

1. Запуск треку.
2. Згортання додатку.
3. Спроба регулювання звуку.
4. Розгортання додатку.
5. Зміна треку.

Очікування: при згорнутому додатку гучність змінюється, після зміни треку він не дублюється.

Результат: при згорнутому додатку гучність змінюється, після зміни треку він не дублюється.

ВИСНОВКИ

У дипломній роботі було розроблено та протестовано додаток для прослуховування аудіо файлів.

Був проведений аналіз його аналогів, виявлені їх недоліки та переваги у порівняні з розробленим додатком.

Використано Xamarin як фреймворк для розробки додатків під систему Android з перспективами на масштабізацію додатку. Також було розроблено інтерфейс за допомогою мови XAML. Функціонал був розроблений за допомогою мови програмування C#.

У майбутній розвитод даного програмного засобу входить:

1. Поєднання з YouTube API для програвання музики через інтернет.
2. Xamarin – фреймворк для розробки кросплатформенних додатків, тож переведення додатку на іншу систему не створить великих труднощів.
3. Розробка бази даних уже створеними плейлистами для користувачів.
4. Можливість зміни кольорової палітри додатку під смаки користувача.
5. Покращення роботи додатку, збільшення швидкодії, підтримка нових форматів аудіо та навіть відео файлів

ПЕРЕЛІК ПОСИЛАНЬ

1. 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: 20.05.2021.

2. What is Xamarin? [Електронний ресурс]: [Веб-сайт]. –Електронні дані. – Режим доступу: https://docs.microsoft.com/en-us/xamarin/get-started/what-is-xamarin. Дата звернення: 20.05.2021.

3. A tour of the C# language. [Електронний ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу: https://docs.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/#:~:text=C%23%20(pronounced%20%22See%20Sharp%22,NET%20ecosystem.&text=C%23%20is%20an%20object%2Doriented%2C%20component%2Doriented%20programming%20language. Дата звернення: 20.05.2021.

4. Create a UI by using XAML Designer. [Електронний ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу: https://docs.microsoft.com/en-us/visualstudio/xaml-tools/creating-a-ui-by-using-xaml-designer-in-visual-studio?view=vs-2019 Дата звернення: 20.05.2021.

5. Get started with Soutcetree. [Електронний ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу: https://confluence.atlassian.com/get-started-with-sourcetree. Дата звернення: 20.05.2021.

6. Welcome to the Visual Studio IDE. [Електронний ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу: https://docs.microsoft.com/en-us/visualstudio/get-started/visual-studio-ide?view=vs-2019. Дата звернення: 20.05.2021.

7. Joseph Albahari, Ben Albahari, C# 7.0 in a Nutshell: The Definitive Reference /- "O'Reilly Media, Inc.", 2017 – с. 146 - 152.

8. Charles Petzold, Creating Mobile Apps with Xamarin.Forms /-"Xamarin Inc.", 2016 – с. 27.