МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
Факультет комп'ютерних наук і технологій
Кафедра комп'ютерних інформаційних технологій

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри
Аліна САВЧЕНКО
"_____"_____2023 р.

КВАЛІФІКАЦІЙНА РОБОТА

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ «БАКАЛАВР»
ЗА СПЕЦІАЛЬНІСТЮ 122 «КОМП'ЮТЕРНІ НАУКИ»

Тема: «Музичний застосунок для мобільних пристроїв на платформі Android»

Виконавець: Харченко Станіслав Володимирович

Керівник: к.т.н., доцент Климова Асія Сабирівна

Нормоконтролер: Олександр ШЕВЧЕНКО
(підпис)

Київ 2023

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет комп’ютерних наук і технологій

Кафедра комп’ютерних інформаційних технологій

Освітній ступінь: «Бакалавр»

Галузь знань, спеціальність, освітньо-професійна програма:

12 «Інформаційні технології», 122 «Комп’ютерні науки», «Інформаційні управляючі системи та технології»

ЗАТВЕРДЖУЮ

Завідувач кафедри

Аліна САВЧЕНКО

“_____”_____2023 р.

ЗАВДАННЯ

на виконання кваліфікаційної роботи

Харченка Станіслава Володимировича

(прізвище, ім'я, по батькові випускника в родовому відмінку)

1. **Тема кваліфікаційної роботи:** «Музичний застосунок для мобільних пристроїв на платформі Android» затверджена наказом ректора №623/ст від 01.05.2023 р.
2. **Термін виконання роботи**: 15.05.2023 – 25.06.2023
3. **Вихідні дані до роботи**: дані про музичні застосунки для платформи Android, документація Android Studio, засоби та технології створення застосунків: Java, Kotlin, XML, API, Android Studio.
4. **Зміст пояснювальної записки**: аналіз предметної області, проектування застосунку, огляд використаних технологій в ході розробки застосунку, процес розробки музичного застосунку, пояснення розробки функціоналу застосунку.
5. **Перелік обов'язкового графічного (ілюстративного) матеріалу:** блок-схеми основних функцій застосунку, головні екрани музичного застосунку.

6. Календарний план-графік

№ пор.	Завдання	Термін виконання	Відмітка про виконання
1	Проаналізувати джерела за темою кваліфікаційної роботи та зробити аналіз предметної області.	15.05.2023-19.05.2023	Виконано
2	Написати перший розділ кваліфікаційної роботи.	22.05.2023-23.05.2023	Виконано
3	Провести аналіз вимог користувача та сформулювати опис екранів застосунку, обрати інструменти розробки.	24.05.2023-26.05.2023	Виконано
4	Написати другий розділ кваліфікаційної роботи.	29.05.2023-02.06.2023	Виконано
5	Розробити дизайн екранів музичного застосунку та створити музичний застосунок.	03.06.2023-05.06.2023	Виконано
6	Написати третій розділ кваліфакційної роботи та зробити висновки.	05.06.2023-08.06.2023	Виконано
7	Оформлення та друк пояснювальної записки кваліфікаційної роботи.	12.06.2023-13.06.2023	Виконано
8	Підготовка презентації та доповіді для виступу.	14.06.2023-16.06.2023	Виконано
9	Захист кваліфікаційної роботи.	19.06.2023-23.06.2023	Виконано

7. Дата видачі завдання: «15» травня 2023 р.

Керівник кваліфікаційної роботи _____ Асія КЛИМОВА
(підпис керівника) (П.І.Б.)

Завдання прийняв до виконання _____ Станіслав ХАРЧЕНКО
(підпис випускника) (П.І.Б.)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи «Музичний застосунок для мобільний пристроїв на платформі Android» містить 58 с., 24 рис., 19 літературних джерел.

Об’єкт дослідження – процес розробки музичного застосунку для мобільних пристроїв на платформі Android.

Предмет дослідження – функціональні можливості та технології реалізації музичного застосунку для мобільних пристроїв на платформі Android.

Мета кваліфікаційної роботи – розробити музичний застосунок для мобільних пристроїв на платформі Android з функціональністю відтворення музики, пошуку музики, налаштування звуку та іншими додатковими можливостями.

Методи дослідження – аналіз предметної області, вивчення технологій розробки мобільних застосунків на платформі Android, використання мов програмування Java та середовища розробки Android Studio, розробка застосунку з використанням архітектурного підходу Model-View-Controller (MVC).

Результатом кваліфікаційної роботи є створений музичний застосунок для мобільних пристроїв на платформі Android. Застосунок має широкий функціонал, включаючи відтворення музики, пошук музики, налаштування звуку та інші додаткові можливості. Практичне значення роботи полягає у створенні зручного і функціонального інструменту для любителів музики, що дозволяє насолоджуватися улюбленою музикою на мобільних пристроях.

МУЗИЧНИЙ ЗАСТОСУНОК, МОБІЛЬНІ ПРИСТРОЇ, ANDROID, JAVA, KOTLIN, XML, API, MVC, ВІДТВОРЕННЯ МУЗИКИ, ПОШУК МУЗИКИ, НАЛАШТУВАННЯ ЗВУКУ

ЗМІСТ

ВСТУП

У сучасному цифровому світі мобільні пристрої, такі як смартфони і планшети, стали невід'ємною частиною нашого повсякденного життя. Вони не тільки дозволяють нам залишатись зв'язаними зі світом і знаходити інформацію в будь-який час, але й надають безліч можливостей для розваг і розвитку. Одним із найпопулярніших способів використання мобільних пристроїв є музика.

Музика має велике значення в житті людей. Вона не тільки допомагає нам відпочивати і розважатись, але й впливає на наші почуття, настрій і навіть когнітивні функції. Музика може створювати атмосферу, передавати емоції і стати частиною нашої особистої ідентичності. І з розвитком технологій ми отримуємо все більше можливостей насолоджуватись музикою в будь-який час і в будь-якому місці.

Однак, розробка музичних застосунків для мобільних пристроїв на платформі Android є складним і актуальним завданням. Система Android, розроблена компанією Google, стала однією з найпопулярніших операційних систем для мобільних пристроїв. Її велика популярність, широка база користувачів і розвинена інфраструктура додатків створюють чудові можливості для розробників музичних застосунків.

Метою даної кваліфікаційної роботи є розробка музичного застосунку для мобільних пристроїв на платформі Android. Основною метою цього застосунку є надання користувачам нових функціональних можливостей для насолодження музикою, покращення їх досвіду та зручності використання мобільних пристроїв для прослуховування музики.

Для досягнення цієї мети, в роботі будуть вирішені наступні завдання:

1. Аналіз існуючих музичних застосунків для мобільних пристроїв на платформі Android, визначення їх переваг і недоліків;

2. Розробка архітектури музичного застосунку, яка включатиме такі основні компоненти, як відтворення музики, керування плейлистами, пошук інформації про виконавців і пісні, налаштування звуку тощо;
3. Реалізація програмного забезпечення музичного застосунку з використанням сучасних інструментів і технологій розробки під платформу Android.

Для досягнення поставлених цілей і вирішення завдань використовуватимуться такі методи дослідження, як аналіз літературних джерел, проектування архітектури та реалізація програмного забезпечення. Для розробки застосунку будуть використані сучасні інструменти та технології, що дозволять досягти бажаних результатів.

Очікується, що результати даної роботи будуть корисними для розробників мобільних застосунків, спрямованих на музичну сферу, а також для користувачів, які зацікавлені в якісному та зручному способі насолоджуватись музикою на своїх мобільних пристроях.

Отже, робота включатиме аналіз предметної області, проектування та реалізацію програмного забезпечення. Очікується, що розроблений застосунок стане корисним і популярним серед користувачів мобільних пристроїв, які цінують музику та хочуть мати зручний та функціональний інструмент для її насолодження.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Що таке мобільний застосунок та платформа Android?

Мобільний застосунок - це програмне забезпечення, розроблене для роботи на мобільних пристроях, таких як смартфони, планшети, ноутбуки та інші портативні пристрої. Вони стали невід'ємною частиною сучасного життя і забезпечують безліч можливостей для користувачів.

Мобільні застосунки можуть бути розроблені для різних операційних систем, таких як Android, iOS, Windows і т. д. Вони можуть бути завантажені з магазинів додатків, таких як Google Play або App Store, або встановлені за допомогою файлів APK. Зазвичай мобільні застосунки мають інтуїтивно зрозумілий і легкий інтерфейс, який дозволяє користувачам швидко та легко взаємодіяти з ними.

Застосунки можуть бути розроблені для різних цілей, включаючи комунікацію, розваги, роботу і навіть навчання. Існують мобільні застосунки для соціальних мереж, месенджерів, онлайн-ігор, здоров'я та фітнесу, фото та відео, музики та інших сфер життя.

Деякі мобільні застосунки можуть працювати в офлайні, що дозволяє користувачам використовувати їх без доступу до Інтернету. Інші можуть потребувати постійного підключення до Інтернету, щоб працювати повністю. Крім того, деякі мобільні застосунки можуть містити рекламу, тоді як інші можуть пропонувати платну версію з більшими функціональними можливостями.

Android є однією з найпопулярніших мобільних платформ у світі і знаходиться під керівництвом компанії Google. Платформа була запущена в 2008 році та стала конкурентом iOS від Apple. Android використовується на різноманітних мобільних пристроях, від смартфонів та планшетів до смарт-годинників та телевізорів.

Android базується на відкритому коді, що дозволяє розробникам вільно змінювати та розширювати функціональність платформи. Окрім того, розробни-ки можуть створювати додатки для Android на різних мовах програмування, та-ких як Java, Kotlin та C++.

Основною мовою програмування для розробки Android-застосунків є Java, що дозволяє розробникам створювати музичні застосунки з різноманітною функціональністю та інтерфейсом. У той же час, Kotlin також набуває популяр-ності серед розробників Android-застосунків завдяки своїм простоті, безпечності та швидкості.

Однією з особливостей Android є наявність різних магазинів додатків, та-ких як Google Play, Amazon Appstore, Samsung Galaxy Apps та інші, що дозво-ляє користувачам отримати доступ до безлічі різноманітних додатків та музич-них застосунків. Окрім того, Android дозволяє користувачам встановлювати до-датки з джерел поза магазином, що дозволяє їм встановлювати та використову-вати неофіційні музичні застосунки.

1.2. Технології, для створення застосунків на платформі Android

Для створення мобільних застосунків на платформі Android використо-вуються різні технології та інструменти, які дозволяють розробникам створюва-ти високоякісні та ефективні додатки. Однією з найбільш поширених мов про-грамування для розробки додатків на Android є Java, що має велику кількість фреймворків, бібліотек та інструментів розробки. Kotlin - новітня мова програ-мування для Android, яка пропонує більш простий та компактний синтаксис, зменшуючи кількість коду та збільшуючи продуктивність.

Для візуального оформлення інтерфейсів користувача використовується мова розмітки XML, що дозволяє розробникам створювати різноманітні компоненти інтерфейсів з високою точністю та швидкістю. Для роботи з даними та комунікації з серверами використовуються API - набір інтерфейсів програмування додатків, які надають доступ до функцій та можливостей платформи Android.

1.2.1. Java

Java є однією з найбільш популярних мов програмування, яка використовується для створення мобільних застосунків на платформі Android. Її використання дозволяє розробникам створювати застосунки, які працюють на будь-якому пристрої, що має підтримку Android.

Вона має декілька переваг, що роблять її популярною серед розробників.

По-перше, Java має велику кількість бібліотек, фреймворків та інструмен-тів, що значно спрощують процес розробки мобільних застосунків. Наприклад, інтегроване середовище розробки Android Studio підтримує мову Java та надає розробникам доступ до багатьох корисних інструментів та бібліотек.

По-друге, Java має велику спільноту розробників та підтримується вели-ким числом компаній. Це означає, що існує велика кількість матеріалів для ви-вчення мови програмування та що можна швидко отримати допомогу в разі по-треби.

Крім того, Java є мовою програмування, що використовується для розро-бки мобільних застосунків на платформі Android з перших днів її існування, то-му її можна вважати досить зрілою технологією, що має відносно високу стабі-льність та надійність.

1.2.2. Kotlin

Kotlin є мовою програмування, що стала досить популярною серед розробників мобільних застосунків на платформі Android в останні роки. Вона має декілька переваг, які дозволяють розробникам швидко та ефективно створювати якісні мобільні додатки.

По-перше, Kotlin є мовою програмування, що була створена спеціально для платформи Android, тому вона є найкращим вибором для розробки мобіль-них застосунків. Kotlin простіший та зрозуміліший для вивчення, порівняно з Java, що робить його більш привабливим для початківців.

По-друге, Kotlin є мовою програмування з високим рівнем безпеки та надійності. Вона має строгу типізацію та попередню перевірку типів, що дозволяє виявляти помилки на етапі компіляції, а не на етапі виконання програми. Це сприяє зменшенню кількості помилок та збільшенню надійності мобільних до-датків.

По-третє, Kotlin має велику кількість корисних функцій та інструментів, що дозволяють розробникам зосередитися на самій логіці програми, а не на ру-тинних завданнях. Наприклад, Kotlin підтримує функціональне програмування та забезпечує зручну роботу з низкою функцій, що дозволяє зменшити кількість коду та полегшити розуміння програми.

Крім того, Kotlin є мовою програмування з відкритим вихідним кодом, що дозволяє розробникам долучатися до розвитку мови та додавати нові функції. Відкритий код також дозволяє розробникам використовувати сторонні бібліоте-ки та фреймворки для розробки.

1.2.3. XML

XML (eXtensible Markup Language) є мовою розмітки, що широко використовується для створення інтерфейсів користувача в мобільних додатках на платформі Android. XML є мовою, яка дозволяє створювати власні теги та атрибу-ти,

що дозволяє розробникам створювати наступний рівень кастомізації та конт-ролю над відображенням інтерфейсу.

Однією з ключових переваг XML є те, що вона є легко читабельною та зрозумілою для розробників, що знижує кількість помилок та сприяє розвитку та збільшенню ефективності роботи розробника.

XML також дозволяє розробникам швидко та ефективно створювати інте-рфейс користувача, оскільки вона відокремлює відображення від логіки програ-ми. Це означає, що розробники можуть працювати з інтерфейсом незалежно від коду та логіки програми.

Ще одною перевагою XML є її гнучкість та розширюваність. Вона дозво-ляє розробникам створювати власні теги та атрибути, що дозволяє розширювати можливості інтерфейсу користувача та забезпечує високу кастомізацію.

XML також є основним форматом для збереження налаштувань та ресур-сів мобільного додатка, таких як зображення та текстові файли. Це дозволяє ро-зробникам швидко та легко змінювати ресурси додатка та забезпечує легку між-платформну сумісність.

1.2.4. API

API (англ. Application Programming Interface, інтерфейс програмування додатків) - це набір інструментів, функцій та процедур, які використовуються розробниками для взаємодії з програмним забезпеченням або конкретним серві-сом.

У випадку з Android, API забезпечують доступ до різноманітних функцій пристроїв, таких як камера, датчики руху, мікрофон, Bluetooth і т.д., а також до різних сервісів, наприклад, Google Maps, Firebase та інших.

Для взаємодії з API Android розробники можуть використовувати різні мови програмування, включаючи Java та Kotlin. Вони можуть використовувати API, щоб створювати свої власні додатки або розширювати функціональність існуючих.

Важливо зауважити, що API можуть змінюватись з релізу до релізу Android, тому розробники повинні слідкувати за оновленнями та змінами, щоб їх додатки

продовжували коректно працювати. Документація API зазвичай на-дається відповідними розробниками платформи, такими як Google.

1.2.5. Android Studio

Android Studio - це офіційне інтегроване середовище розробки (IDE) для платформи Android, яке надає розробникам потужні інструменти для створення мобільних додатків.

Android Studio побудовано на основі IntelliJ IDEA від Jetbrains і містить широкий спектр інструментів та функцій для розробки, налагодження та тесту-вання додатків на платформі Android. Для розробки Android додатків, розроб-ники можуть використовувати різні мови програмування, включаючи Java та Kotlin.

Android Studio надає зручний інтерфейс користувача, що дозволяє легко створювати і налагоджувати додатки, використовуючи різноманітні інструмен-ти, такі як візуальний редактор інтерфейсу користувача, графічний редактор XML, налагоджувач, систему збірки та інші. Крім того, Android Studio має вбу-довані засоби для розгортання додатків на різних пристроях, таких як емулято-ри пристроїв або фізичні пристрої.

Android Studio також надає доступ до багатьох різноманітних бібліотек та плагінів, які дозволяють розширювати функціональність IDE та додатків. Деякі з найпопулярніших бібліотек для розробки музичних додатків включають ExoPlayer, SoundPool та MediaPlayer.

Одним із найважливіших аспектів розробки додатків на платформі Android є оптимізація продуктивності та ефективності додатка. Android Studio має різні інструменти для аналізу продуктивності, такі як Android Profiler, який дозволяє відстежувати ресурсоємність додатка, включаючи використання про-цесора, пам'яті та мережі.

1.3. Що таке музичний застосунок на платформі Android?

Музичний застосунок для мобільних пристроїв на платформі Android - це програмне забезпечення, призначене для відтворення та організації музичних композицій на пристроях з операційною системою Android. Він забезпечує користувачам можливість організовувати свою музичну бібліотеку, відтворювати музику, створювати списки відтворення та ділитися музичним контентом з іншими користувачами.

Основна функціональність музичного застосунку для мобільних пристроїв на платформі Android включає наступні функції:

- Відтворення музики в різних форматах (наприклад, MP3, FLAC, AAC);
- Створення та редагування списків відтворення;
- Організація музичної бібліотеки за артистами, альбомами, жанрам та іншими критеріями;
- Пошук музичних композицій в Інтернеті та додавання їх до бібліотеки;
- Синхронізація музичної бібліотеки з хмарними сервісами, такими як Google Drive, Dropbox та іншими;
- Різноманітні налаштування звуку, такі як еквалайзер, підсилення басів, налаштування звуку в наушниках тощо;
- Підтримка різних мов та інтерфейсів користувача.

Найбільш очевидною перевагою музичного застосунку на платформі Android є можливість безкоштовно завантажувати різноманітний музичний контент з Інтернету. Крім того, багато з цих додатків мають велику базу даних пісень, яку можна легко організувати і зберігати на мобільному пристрої.

Додатки для прослуховування музики на Android зазвичай мають простий та зручний інтерфейс, що дозволяє користувачам швидко знайти та відтворити пісні. Більшість з них також мають можливість налаштувати якісність звуку та пристосувати її до власних потреб.

Окрім цього, багато музичних застосунків для Android мають функцію прямого відтворення, що дозволяє користувачам слухати музику безпосередньо з Інтернету, без необхідності завантажувати її на мобільний пристрій.

1.4. Типи музичних застосунків на платформі Android

На платформі Android доступні різноманітні музичні застосунки. До них належать музичні плеєри, які дозволяють відтворювати музику, створювати плейлисти та налаштовувати звукові ефекти. Потокові сервіси музики надають доступ до великого обсягу музичного контенту з Інтернету, дозволяючи слухати музику онлайн або офлайн. Також на платформі Android є музичні соціальні мережі, які забезпечують спілкування, обмін музикою та відкривають нові музичні відкриття. За допомогою цих музичних соціальних мереж користувачі можуть спілкуватися з іншими меломанами, ділитися улюбленими треками, отримувати рекомендації та відкривати нову музику.

1.4.1. Музичні плеєри

Музичні плеєри - це програми для відтворення музики на мобільних пристроях. Вони є одними з найбільш поширених та корисних музичних застосунків для Android-пристроїв. Музичні плеєри дозволяють користувачам відтворювати музику у форматах MP3, WAV, AAC та інших, які зберігаються на мобільних пристроях.

Основна функціональність музичних плеєрів полягає у відтворенні музики, але вони мають додаткові функції, які забезпечують більш зручне та ефективне використання. Наприклад, користувачі можуть створювати свої власні плейлисти, додавати альбоми до бібліотеки, переглядати метадані про треки, які відтворюються, тощо.

1.4.2. Музичні стрімінгові сервіси

Музичні стрімінгові сервіси - це застосунки, які дозволяють користувачам відтворювати музику через Інтернет без необхідності завантажувати або зберігати музичні файли на мобільному пристрої. Такі сервіси працюють на підписці, і користувачі мають доступ до великої кількості треків з різних жанрів та виконавців. Деякі з найпопулярніших музичних стрімінгових сервісів на Android-платформі: Spotify, Deezer, Google Play Music, YouTube Music, Apple Music та інші.

Основна перевага музичних стрімінгових сервісів - це доступ до великої кількості музики, яка може бути відтворена без необхідності завантажувати її на пристрій. Крім того, користувачі можуть створювати свої власні плейлисти, ділитися музикою з друзями та отримувати персоналізовані рекомендації в залежності від їхніх музичних уподобань.

1.4.3. Музичні соціальні мережі

Музичні соціальні мережі - це платформи, де користувачі можуть спілкуватися із друзями, обмінюватися музичними композиціями та слухати плейлисти, створені іншими користувачами. Такі застосунки можуть бути вбудовані в інші соціальні мережі, такі як Facebook або Instagram, або мати свій власний додаток.

Найбільш відомими музичними соціальними мережами на платформі Android є:

- SoundCloud - це популярна платформа для поширення музики, яка дозволяє користувачам завантажувати свої композиції та слухати музику, створену іншими користувачами. SoundCloud також має функції спілкування та обміну повідомленнями між користувачами;
- TikTok - це соціальна мережа, яка дозволяє користувачам створювати короткі відео з музикою, а також переглядати та коментувати відео інших користувачів;

- Last.fm - це музична соціальна мережа, яка дозволяє користувачам створювати свої власні радіостанції на основі своїх уподобань у музиці та слухати радіостанції, створені іншими користувачами. Last.fm також надає можливість відстежувати інформацію про те, яку музику ви слухаєте, та рекомендувати нові композиції на основі цієї інформації.

Переваги музичних соціальних мереж на платформі Android полягають у тому, що вони надають можливість користувачам обмінюватися музикою та спілкуватися з друзями на платформі, яка спеціалізується на музиці.

1.5. Основні функціональні можливості музичного застосунку

Музичні застосунки на платформі Android пропонують різноманітні функціональні можливості, які допомагають користувачам насолоджуватися музикою за своїми уподобаннями та відкривати нові музичні враження.

Ці застосунки зазвичай мають основні функції, такі як відтворення музики збережених на пристрої або потокових сервісів, створення та управління плейлистами. Вони надають користувачам можливість контролювати рівень гучності, перемотувати треки та використовувати еквалайзер для налаштування звукового спектру. Деякі музичні застосунки також пропонують функції автоматичного рекомендування треків, основані на історії прослуховування та музичних вподобаннях користувача.

Окрім базових функцій, деякі музичні застосунки на Android мають розширені можливості, такі як використання звукових ефектів, створення міксів, редагування треків та створення власних композицій. Деякі застосунки надають інтерактивність, де користувачі можуть брати участь у віртуальних спільнотах, обмінюватися музикою з іншими користувачами, або слідкувати за улюбленими виконавцями та отримувати оновлення про їхні нові випуски та концерти.

1.5.1. Відтворення музики

Однією з основних функцій музичного застосунку є відтворення музики на мобільному пристрої. Для цього застосунок повинен мати можливість відтворювати аудіофайли різних форматів, таких як MP3, WAV, AAC, OGG та інших. При відтворенні музики, музичний застосунок може показувати інформацію про відтворюваний трек, таку як назва, виконавець, назва альбому та обкладинка альбому. Для зручності користувача також може бути відображено поточний час треку та час відтворення вже пройденого часу.

Крім того, музичний застосунок повинен забезпечувати різноманітні можливості контролю за відтворенням музики. Користувач повинен мати можливість зупинити, пропустити або перемотати трек. Також може бути забезпечено можливість повторення або перемішування відтворення треків.

Крім того, музичний застосунок може мати можливість відтворювати музику в фоновому режимі, коли користувач використовує інші програми або знаходиться в режимі очікування. Це дозволяє користувачеві продовжувати слухати музику, не перериваючи роботу з іншими програмами.

1.5.2. Пошук музики

Пошук музики - це ще одна важлива функція музичного застосунку. Для того, щоб користувач міг знайти потрібну музику у застосунку, музичний застосунок повинен забезпечувати можливість пошуку за різними параметрами, такими як назва виконавця, назва треку, альбому, жанру та інші. Крім того, музичний застосунок може мати можливість пошуку музики за різними критеріями, наприклад, за роками, популярністю, рейтингом та іншими.

Окрім внутрішнього пошуку, музичний застосунок може використовувати різні онлайн сервіси для пошуку музики, такі як Spotify, Soundcloud, YouTube Music та інші. Користувач може виконувати пошук музики не тільки в межах вбудованої медіатеки, але і в онлайн сервісах, що значно розширює можливості користувача.

Музичний застосунок може забезпечувати також можливість прослуховування радіостанцій. Користувач може вибрати бажану радіостанцію зі списку доступних радіостанцій або ввести її назву в пошукову стрічку.

Крім того, музичний застосунок може забезпечувати можливість пошуку музики, що є важливим функціоналом для користувачів. Пошук музики може бути здійснений за допомогою різних параметрів, таких як назва артиста, назва пісні, альбом, жанр і т.д. Зазвичай, музичний застосунок має вбудовану функцію пошуку, що дозволяє користувачеві знайти потрібну пісню швидко та зручно.

Для забезпечення якісного пошуку, музичний застосунок може використовувати різні методи та алгоритми, такі як пошук за ключовими словами, пошук за атрибутами пісні, аналіз музичних характеристик тощо. Наприклад, використання технології машинного навчання може покращити якість пошуку, дозволяючи системі навчитися розпізнавати індивідуальні вподобання користувача та рекомендувати пісні на основі цих даних.

1.5.3. Пошук музики

Створення списку відтворення є важливою функцією музичного застосунку, оскільки дозволяє користувачеві організовувати та керувати своєю музикою. Ця функція дозволяє створювати персоналізовані плейлисти, які можуть містити пісні з різних жанрів та виконавців, а також відповідати різним настроям та ситуаціям.

Музичний застосунок на платформі Android може давати користувачеві можливість створювати списки відтворення різного типу, наприклад, за жанрами, альбомами, ріком виходу, частотою відтворення та іншими параметрами. Крім того, такі списки можуть бути збережені в пам'яті пристрою та легко відновлюватись при наступних відвідуваннях застосунку.

Для зручного створення списків відтворення музичного застосунку можуть бути додані функції, такі як автоматичне додавання пісень до списку на основі індивідуальних вподобань користувача, збірка плейлістів з рекомендаціями,

сортування пісень за різними параметрами (назвою, виконавцем, тривалістю тощо), можливість додавати та видаляти пісні зі списку відтворення.

Для покращення зручності користування, музичний застосунок може мати вбудовані функції плейлистів, такі як автозапуск плейлисту при запуску застосунку, переключення між плейлистами з одним дотиком, можливість додавання нових пісень до списку відтворення в режимі відтворення та інші.

1.5.4. Налаштування звуку

Музичний застосунок для мобільних пристроїв на платформі Android надає можливість користувачам налаштувати звукові параметри відтворення музики за допомогою різних функцій налаштування звуку.

Один з таких параметрів - це еквалайзер. Еквалайзер є інструментом для регулювання частотного спектру музики, що відтворюється. Він дозволяє користувачам налаштувати звук відповідно до їхніх власних вимог і відчуттів. У музичному застосунку на платформі Android доступні різні налаштування еквалайзера, наприклад, "Rock", "Jazz", "Pop" тощо. Користувачі також можуть налаштувати еквалайзер вручну, вибравши потрібні частоти та їхні значення.

Крім еквалайзера, музичний застосунок на платформі Android також має функції налаштування гучності, балансу та звуку з ефектами. Наприклад, можливо включити ефекти, такі як "реверберація" або "3D-звук", для покращення звукового досвіду користувача.

Крім того, музичний застосунок на платформі Android може містити функції, що дозволяють налаштувати звук на різних пристроях, таких як навушники, динаміки, Bluetooth-гарнітури тощо. Користувачі можуть мати різні вимоги до якості звуку, тому важливо мати можливість налаштувати його на свій смак. Типові параметри налаштування звуку включають еквалайзер, баланс лівого та правого каналів, затримку звуку і підсилення низьких частот.

1.6. Висновки до розділу

У даному розділі кваліфікаційної роботи було розглянуто основні аспекти, пов'язані з розробкою музичного застосунку для мобільних пристроїв на платформі Android. Було визначено, що мобільний застосунок є програмним забезпеченням, призначеним для використання на мобільних пристроях. Платформа Android була розглянута як одна з найпопулярніших платформ для розробки мобільних додатків, зокрема музичних застосунків.

Визначено основні технології, використовувані для створення застосунків на платформі Android, зокрема Java, Kotlin, XML, API та Android Studio. Вони надають розробникам потужні інструменти для реалізації різноманітних функціональних можливостей у музичних застосунках.

Досліджено сутність музичного застосунку на платформі Android та виділено його типи, такі як музичні плеєри, музичні стрімінгові сервіси та музичні соціальні мережі. Кожен з цих типів має свої особливості та функціональні можливості, спрямовані на забезпечення задоволення користувачів у сфері музичного відтворення та взаємодії з музикою.

Визначено основні функціональні можливості музичного застосунку. Ці функції дозволяють користувачам насолоджуватись музикою, знаходити та відтворювати їх улюблені треки, а також налаштовувати звукові параметри за своїми вподобаннями. Розуміння технологій, які використовуються для створення музичного застосунку, та функціональних можливостей допоможе створити більш ефективний та зручний музичний застосунок на платформі Android.

РОЗДІЛ 2
ПРОЄКТУВАННЯ МУЗИЧНОГО ЗАСТОСУНКУ

2.1. Вимоги до музичного застосунку

Визначення вимог до музичного застосунку дозволяє уточнити його функціональні та нефункціональні характеристики, що сприяє ефективному проектуванню та розробці застосунку. Зрозуміння потреб та очікувань користувачів є ключовим для створення музичного застосунку, який буде задовольняти їхні вимоги та надавати зручний та приємний досвід використання[3].

Аналіз вимог для музичного застосунку на платформі Android є вирішальним кроком у створенні високоякісного продукту, що забезпечує задоволення потреб користувачів та поліпшення їхнього користувацького досвіду. Вивчення унікальних потреб та переваг цільової аудиторії, проведення досліджень ринку та конкурентного середовища, а також збір зворотного зв'язку від потенційних користувачів дозволяє отримати повну інформацію про очікування користувачів щодо функціональності, дизайну, взаємодії та інших аспектів музичного застосунку. Такий аналіз створює основу для подальших етапів проектування та реалізації застосунку, гарантуючи відповідність між очікуваннями користувачів та функціональністю самого застосунку. Перш за все, вимоги можна поділити на функціональні та нефункціональні.

Функціональні вимоги визначають конкретні функції та можливості, які повинен мати музичний застосунок. Серед них можуть бути такі функції, як відтворення музики, пошук треків за різними критеріями, створення та управління списками відтворення, регулювання звуку, підтримка різних аудіоформатів тощо. Функціональні вимоги допомагають визначити, які можливості повинні бути доступні для користувачів музичного застосунку.

Музичний застосунок повинен мати наступні функціональні можливості, щоб задовольнити потреби користувачів:

- Відтворення музики: Застосунок повинен забезпечувати зручний та безперебійний процес відтворення музичних композицій у різних форматах, таких як MP3, WAV, AAC тощо. Користувачі повинні мати можливість вибирати та відтворювати треки, а також управляти програванням (пауза, перемотування, регулювання гучності).
- Пошук музики: Застосунок повинен мати функцію пошуку, яка дозволяє користувачам знаходити конкретні музичні композиції за назвою, виконавцем, альбомом або жанром. Результати пошуку повинні бути точними та швидкими, щоб забезпечити зручність використання.
- Створення списку відтворення: Застосунок повинен давати можливість користувачам створювати власні списки відтворення, додавати до них обрані треки та налаштовувати послідовність програвання. Це дозволяє користувачам насолоджуватися власними персоналізованими плейлистами та контролювати музичний досвід.
- Налаштування звуку: Застосунок повинен надавати можливість користувачам налаштовувати параметри звуку, такі як еквалайзер, баланс каналів, підсилення басів тощо. Це дозволяє користувачам налаштовувати звукові налаштування під свої вподобання та оточення.

Помимо функціональних вимог, музичний застосунок також повинен відповідати наступним нефункціональним вимогам, щоб забезпечити задоволення користувачів та оптимальну продуктивність:

- Відмінна продуктивність: Застосунок повинен працювати швидко та ефективно, забезпечуючи миттєву реакцію на команди користувача. Затримки відтворення, зависання та інші перебої мають бути мінімальними, щоб забезпечити безперервний музичний досвід.
- Зручний та привабливий інтерфейс: Застосунок повинен мати інтуїтивно зрозумілий та привабливий інтерфейс, який дозволяє користувачам легко зорієнтуватися та використовувати різні функції застосунку. Елементи

керування, кнопки та інші елементи інтерфейсу повинні бути розташовані зручно та естетично задля забезпечення комфортного користування.

- Сумісність та доступність: Застосунок повинен бути сумісний з різними версіями операційної системи Android та різними моделями мобільних пристроїв. Він повинен працювати стабільно та без помилок на різних пристроях. Крім того, застосунок повинен бути доступним для використання людьми з різними потребами, забезпечуючи підтримку адаптивного дизайну та можливість налаштування розміру шрифту та інших параметрів.
- Безпека та приватність: Застосунок повинен забезпечувати високий рівень безпеки та захисту приватності користувачів. Це включає захист від несанкціонованого доступу до особистої інформації користувачів, шифрування передачі даних та інші заходи безпеки для запобігання втраті чи зловживанню даними.

Відповідно до цих вимог, музичний застосунок буде забезпечувати широкий функціонал та задовольняти високі стандарти продуктивності, зручності використання, сумісності, безпеки та приватності.

2.2. Архітектура музичного застосунку

Архітектура музичного застосунку грає важливу роль у процесі розробки, оскільки вона визначає структуру та організацію компонентів, що взаємодіють між собою. Вибір відповідного архітектурного підходу допомагає забезпечити ефективну розробку, підтримку та розширення музичного застосунку.

2.2.1. Підхід Model-View-Controller (MVC)

Один з найпоширеніших підходів до архітектури музичних застосунків - Model-View-Controller (MVC). Цей підхід розбиває застосунок на три основні компоненти: модель (Model), представлення (View) та контролер (Controller).

Модель відповідає за обробку даних, таких як інформація про музичні треки, списки відтворення та налаштування звуку. Вона забезпечує доступ до даних та логіку їх обробки.

Представлення відповідає за візуальну частину застосунку, яку бачить користувач. Воно відображає дані з моделі та надає користувачеві можливість взаємодіяти з застосунком.

Контролер відповідає за обробку вхідних подій та керування взаємодією між моделлю та представленням. Він приймає вхідні дані від користувача, виконує необхідні операції та оновлює модель та представлення відповідно.

MVC підхід дозволяє розділити логіку, представлення та управління застосунком, що сприяє зрозумілості, перевикористанню та тестуванню коду. Діаграму взаємодіє можна переглянути на Рис.2.2.1.

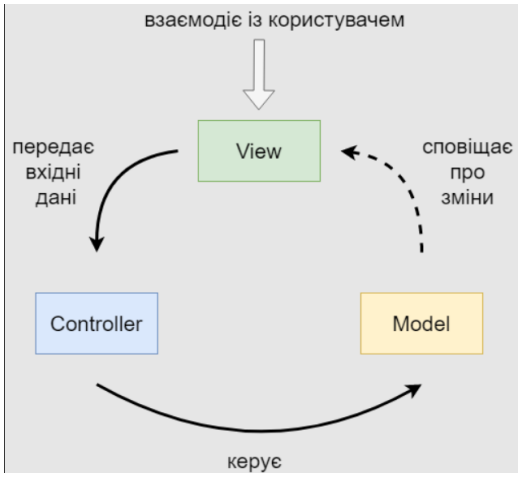

Рис.2.2.1. Діаграма взаємодії між компонентами шаблону MVC

2.2.2. Підхід Model-View-ViewModel (MVVM)

Інший популярний підхід до архітектури музичних застосунків - Model-View-ViewModel (MVVM). Цей підхід також розділяє застосунок на три компоненти: модель (Model), представлення (View) та модель представлення (ViewModel).

Модель в MVVM відповідає за управління даними, їх обробку та збереження (Рис.2.2.2.). Вона може включати логіку доступу до бази даних, мережеві запити та інші операції, пов'язані з обробкою даних музичного застосунку.

Представлення відображає інтерфейс користувача та забезпечує його взаємодію з застосунком. Воно відображає дані з моделі та передає введення користувача до моделі.

Модель представлення (ViewModel) використовується для забезпечення зв'язку між моделлю та представленням. Вона перетворює дані з моделі в формат, зручний для представлення, та надає команди та функції, які можуть викликати дії в моделі.

MVVM підхід дозволяє відокремити бізнес-логіку від представлення та спрощує тестування та розширення застосунку. Він також підтримує зручну реалізацію шаблонів даних та зв'язку між компонентами.

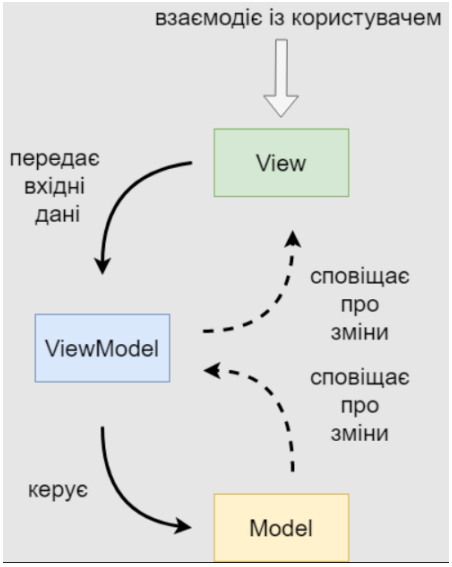

Рис.2.2.2. Діаграма взаємодії між компонентами шаблону MVVM

Було обрано підхід Model-View-Controller (MVC) для архітектури музичного застосунку з кількох причин.

По-перше, MVC є дуже популярним та широко використовуваним підходом в розробці програмного забезпечення. Це означає, що існує велика кількість ресурсів, документації та підтримки, що допоможе мені в розробці та розширенні мого застосунку.

По-друге, MVC дозволяє чітко розділити логіку застосунку на три основні компоненти: модель, представлення та контролер. Це сприяє зрозумілості коду та

полегшує підтримку та тестування. Модель відповідає за обробку даних музичних треків, списків відтворення та налаштувань звуку. Представлення забезпечує візуальну частину застосунку, де користувач може взаємодіяти з даними. Контролер обробляє вхідні події та керує взаємодією між моделлю та представленням.

Крім того, MVC сприяє перевикористанню коду та забезпечує легку модульність. Кожен компонент може бути розроблений та модифікований незалежно, що полегшує розвиток та розширення застосунку у майбутньому. Наприклад, якщо в майбутньому планується додати підтримку нових форматів музичних файлів, то тоді можна зосередитись на моделі, не змінюючи представлення та контролер.

Загалом, обравши підхід MVC, буде забезпечений структурований та організований підхід до розробки музичного застосунку. Це надасть гнучкість, легкість утримання та можливість легко масштабувати застосунок у майбутньому.

2.3. Вибір технологій для реалізації музичного застосунку

При виборі технологій для розробки веб-застосунків, беруться до уваги різні чинники, такі як цілі проєкту, вимоги до функціональності. У цьому процесі головний акцент іде на те, щоб обрати найбільш підходящі технології, які задовольнять потреби користувачів та сприятимуть успішному впровадженню проєкту.

2.3.1. Фреймворки та API

При розробці музичного застосунку для платформи Android було обрано фреймворк та декілька інструментів, які допоможуть у реалізації функціональності та покращенні користувацького досвіду:

- **Android SDK;**

Android SDK — середовище розробки додатків для операційної системи Android. Вона дозволяє створювати та тестувати Android-додатки, що

використовують камеру мобільного пристрою, акселерометр, компас, дані GPS, доступ по Bluetooth, Wi-Fi, EDGE і 3G[6].

Android SDK підтримує роботу з мультимедійним контентом (аудіо, відео, зображення у форматах MPEG4, H.264, MP3, AAC, AMR, JPG, PNG і GIF), базами даних SQLite, інтегрованим браузером на рушії WebKit, віртуальною машиною Dalvik, GSM телефонією і т.д. Користувачі Android SDK мають можливість тестувати створені ними програми за допомогою вбудованого емулятора[6].

Одним з основних компонентів, що буде використуватись, є Android SDK (Software Development Kit). Цей набір інструментів надає необхідні засоби, документацію та API для розробки Android-додатків. За допомогою Android SDK можна взаємодіяти зі звуковими пристроями, мережевими можливостями та базою даних, що є важливим для розробки музичного додатку. Android SDK включає в себе фреймворк Android, який надає основу для розробки додатків під платформу Android.

– **Cleveroad Visualizer API;**

Для розширення можливостей музичного застосунку було обрано Cleveroad Visualizer API. Це потужне інтерфейсне рішення, що дозволяє створювати унікальну візуалізацію звуку, включаючи анімовані спектрограми, хвильові форми та інші захоплюючі ефекти, які відповідають музичним трекам. Використання Cleveroad Visualizer API надасть нашому музичному застосунку ефектності та взаємодії з аудіовмістом, роблячи його особливо привабливим для користувачів.

Cleveroad Visualizer API дозволяє нам взаємодіяти з аудіофайлами, визначати їх характеристики, такі як частота, амплітуда та час, і трансформувати ці дані в візуальні елементи, які синхронізуються з музичними ритмами. Це дозволяє створювати захоплюючі візуальні ефекти, які доповнюють музичний досвід користувача.

Крім того, Cleveroad Visualizer API забезпечує гнучкість налаштування, що дозволяє налаштовувати параметри візуалізації, включаючи кольори, стилі, анімаційні ефекти та швидкість відтворення. Це дозволяє нам створювати унікальні

візуальні елементи, які відображають унікальність музичного контенту та створюють незабутні враження у користувачів.

З використанням Cleveroad Visualizer API музичний застосунок стає більш інтерактивним та захоплюючим. Візуалізація звуку, яку надає це API, додає ефектності та взаємодії з музикою, створюючи неповторний досвід для користувачів.

– **Android Equalizer API;**

Для забезпечення максимального контролю над звуковими налаштуваннями та індивідуалізації звучання до особистих переваг, було вирішено додати у музичний застосунок Android Equalizer. Цей потужний інструмент надає користувачам можливість налаштовувати еквалайзер та рівні звуку для досягнення оптимального звукового досвіду.

Android Equalizer дозволяє налаштовувати різні частотні діапазони звуку, такі як низькі, середні та високі частоти, для створення балансованого та точного звучання музики. Користувачі можуть змінювати рівні окремих частот, забезпечуючи підвищення або зниження інтенсивності звуку в конкретних діапазонах.

Однією з унікальних особливостей Android Equalizer є його можливість запам'ятовувати налаштування еквалайзера для різних аудіофайлів або жанрів музики. Це дозволяє користувачам налаштовувати звук спеціально під конкретні треки або типи музики, що створює персоналізований звуковий досвід.

Завдяки використанню Android Equalizer у музичному застосунку, користувачі зможуть налаштовувати звук відповідно до своїх власних уподобань та насолоджуватись звуком, який оптимально відповідає їхнім унікальним пристрастям. Це надає можливість відкрити нові рівні звукового виразності та зробити музичний досвід ще більш особистим та захоплюючим.

2.3.2. Інструменти розробки

Java

Для розробки музичного застосунку для мобільних пристроїв на платформі Android використовуватиму мову програмування Java.

Java (вимовляється Джава; у нас інколи Ява) — об'єктноорієнтована мова програмування, випущена компанією Sun Microsystems у 1995 році як основний компонент платформи Java. Синтаксис мови багато в чому походить від C та C++. Java програми виконуються у середовищі віртуальної машини Java. Java програми компілюються у байткод, який при виконанні інтерпретується віртуальною машиною для конкретної платформи. У 2009 році Sun Microsystems придбала компанія Oracle, яка продовжує розвивати «Джава»[6].

Java є однією з основних мов програмування, яка використовується для розробки мобільних застосунків на платформі Android. Вона пропонує широкий набір інструментів та функцій, що дозволяють ефективно створювати різноманітні музичні застосунки.

Одним з переваг використання Java є його висока переносимість. Застосунки, написані на Java для Android, можуть працювати на різних пристроях з операційною системою Android без необхідності внесення значних змін. Вона має велику спільноту розробників, багату екосистему та добре документовану API-бібліотеку, що робить її ідеальним вибором для реалізації музичного застосунку.

Android Studio

Для розробки музичного застосунку для платформи Android використовуватиметься Android Studio, інтегроване середовище розробки (IDE), розроблене компанією Google спеціально для створення Android-застосунків. Android Studio надає розробникам зручні та потужні інструменти для розробки, тестування та налагодження музичного застосунку.

Середовище розробки адаптоване для виконання типових завдань, що вирішуються в процесі розробки застосунків для платформи Android. У тому числі у

середовище включені засоби для спрощення тестування програм на сумісність з різними версіями платформи та інструменти для проектування застосунків, що працюють на пристроях з екранами різної роздільності (планшети, смартфони, ноутбуки, годинники, окуляри тощо). Крім можливостей, присутніх в IntelliJ IDEA, в Android Studio реалізовано кілька додаткових функцій, таких як нова уніфікована підсистема складання, тестування і розгортання застосунків, заснована на складальному інструментарії Gradle і підтримуюча використання засобів безперервної інтеграції[7].

Android Studio підтримує весь цикл розробки застосунку, починаючи зі створення проекту та написання коду на мові програмування Java, до компіляції, відлагодження та пакування готового застосунку для розповсюдження. Інтегровані інструменти Android Studio дозволяють візуально створювати та налаштовувати користувацький інтерфейс застосунку, додавати функціональність звуку та музики, а також використовувати інші розширені можливості, які надаються Android SDK та сторонніми бібліотеками.

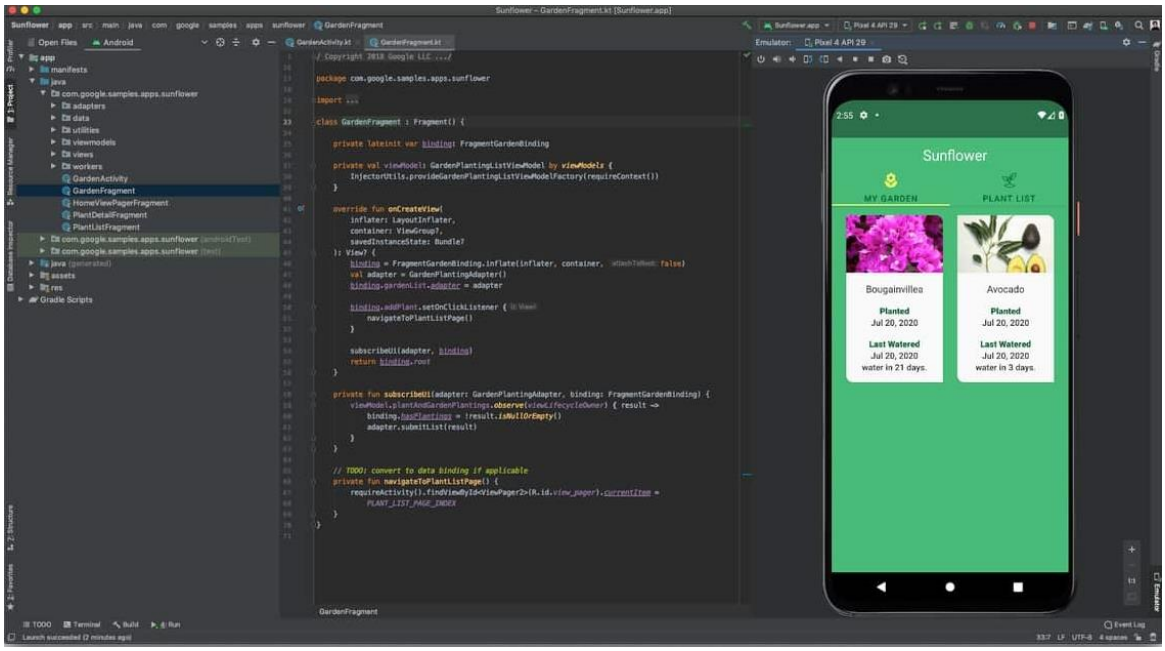

Рис. 2.3.1. Приклад інтерфейсу Android Studio

Figma

В процесі розробки музичного застосунку буде використаний Figma - потужний інструмент для дизайну та прототипування інтерфейсів користувача. Використання Figma дозволить створювати детальні макети та взаємодійні прототипи, що допоможуть візуалізувати та уточнити концепцію музичного застосунку перед його реалізацією.

Сервіс є безкоштовним для індивідуальних користувачів і платним для фахових команд. Даний редактор підходить як для створення простих прототипів і дизайн-систем, так і складних проєктів (мобільні додатки, портали)[8].

Модель розвитку Figma, спростила співробітництво в усьому процесі створення цифрових продуктів для дизайнерів, розробників, менеджерів і маркетологів дозволила на початку 2018 року залучити додаткові 25 мільйонів доларів інвестицій від партнерів[9].

Завдяки Figma буде можливо створювати естетично привабливий та інтуїтивно зрозумілий інтерфейс користувача для музичного застосунку. Буде використаний багатий набір інструментів та можливостей Figma для розміщення елементів усередині застосунку, створення стилів, кольорових палітр та іконок, а також для розташування контролів у зручний спосіб.

Крім того, Figma дозволить створювати взаємодійні прототипи, що дозволять відтворити потенційний користувацький досвід у музичному застосунку. Це дозволить перевірити функціональність та зручність використання застосунку, взаємодіяти з різними елементами і переконатися, що він відповідає очікуванням користувачів.

Застосування Figma в процесі розробки музичного застосунку значно зпростить та прискорить процес дизайну та прототипування, дозволяючи зосередитися на створенні привабливого та функціонального інтерфейсу, що задовольнятиме потреби користувачів.

2.4. Висновки до розділу

У цьому розділі були розглянуті технології, які використовуються при розробці музичного застосунку для мобільних пристроїв на платформі Android.

Була проведена аналітика вимог користувачів, а також було обрано оптимальну архітектуру для реалізації музичного застосунку.

Підсумовуючи, при проектуванні музичного застосунку використовуються різні технології, які взаємодіють між собою для досягнення успішного результату. Використання *Android SDK* дозволяє розробникам взаємодіяти зі звуковими пристроями, мережевими можливостями та базою даних, створюючи потужні музичні функції. Завдяки *Cleveroad Visualizer API* маємо можливість створювати вражаючі візуалізації звуку, додаючи ефектності та взаємодії зі звуковим вмістом у застосунку. *Android Equalizer* дозволяє користувачам налаштовувати звук під свої уподобання, що забезпечує персоналізацію та забезпечення оптимального звукового досвіду. Використання мови програмування *Java* та інтегрованого середовища розробки *Android Studio* дозволяє зручно та ефективно розробляти застосунок з використанням широкого спектру інструментів та підтримки спільноти розробників. Нарешті, *Figma* допомагає візуалізувати та уточнити концепцію музичного застосунку шляхом створення детальних макетів та взаємодійних прототипів, що допомагає створити естетичний та зручний інтерфейс для користувачів. Застосування цих технологій забезпечує потужність, функціональність та ефективність розробки музичного застосунку на платформі Android.

РОЗДІЛ 3
РОЗРОБКА МУЗИЧНОГО ЗАСТОСУНКУ

3.1. Розробка користувацького інтерфейсу

3.1.1. Визначення структури та навігації застосунку

Задача планування структури застосунку включає визначення основних екранів та їх взаємозв'язків. Нижче наведений план структури музичного застосунку для платформи Android:

1. Головний екран:
    - Відображення списку треків;
    - Можливість переходу до екрану відтворення музики;
    - Можливість переходу до екрану з улюбленою музикою;
    - Можливість переходу до екрану пошуку музики;
    - Можливість переходу до екрану налаштувань.
2. Екран відтворення музики:
    - Відображення інформації про трек;
    - Візуалізація звуку за допомогою *Cleveroad Visualizer*;
    - Керування відтворенням (відтворення/пауза, перемотування);
    - Можливість переходу до наступного або попереднього треку;
    - Можливість додавання або видалення треку з плейлисту «Улюблені»;
    - Можливість переходу до екрану зі словами треку.
3. Екран пошуку музики:
    - Введення критеріїв пошуку (назва треку, виконавець);
    - Відображення результатів пошуку (список треків, виконавців);
    - Можливість переходу до екрану відтворення вибраного треку.

4. Екран з улюбленою музикою:
    - Відображення списку треків;
    - Можливість переходу до екрану відтворення музики.
5. Екран зі словами треку
    - Відображення тексту пісні, якщо він є доступний;
    - Можливість переходу до головного меню.
6. Екран з еквалайзером
    - Відображення налаштувань еквалайзеру за допомогою *Android Equalizer*.
7. Екран налаштувань
    - Можливість увімкнути/вимкнути режим «Shake to change»;
    - Можливість переходу до головного екрану.
8. Екран «Про нас»
    - Відобреження версії застосунку та його логотипу;
    - Можливість переглянути ліцензії API *Cleveroad Visualizer* та *Android Equalizer*.

Ця послідовність переходів створює зручність використання та логічну навігацію для користувачів, допомагаючи їм швидко та ефективно переходити між різними функціями та взаємозв'язаними екранами.

3.1.2. Дизайн інтерфейсу

Для розробки привабливого та функціонального дизайну інтерфейсу музичного застосунку використовувався інструмент Figma. Цей інструмент дозволяє створювати естетично збалансовані макети з урахуванням принципів дизайну інтерфейсів.

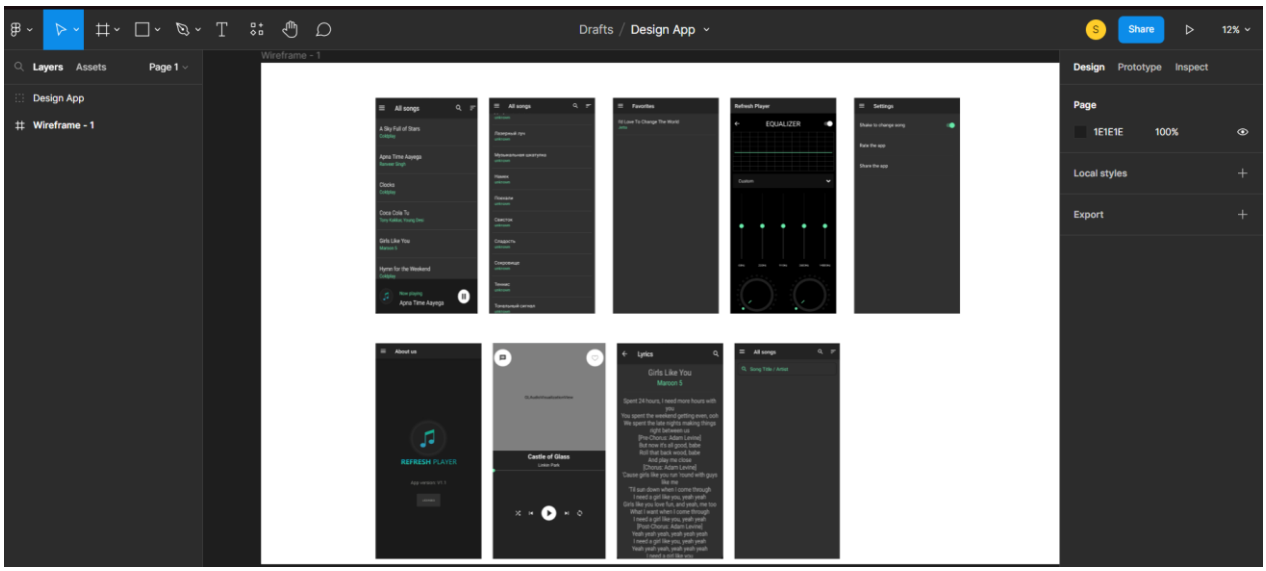

Рис. 3.1.1. Проектування музичного застосунку в Figma

Починаючи з вибору відповідної кольорової палітри та теми оформлення, була ціль створити затишний та сучасний інтерфейс для користувачів. Використовуючи Figma, було розроблено графічні елементи, такі як іконки, кнопки та в цілому загальний дизайн застосунку.

Згідно аналізу вимог користувачів та плану структури музичного застосунку для платформи Android було створено дизайн екранів:

- Головний екран (Рис. 3.1.2);

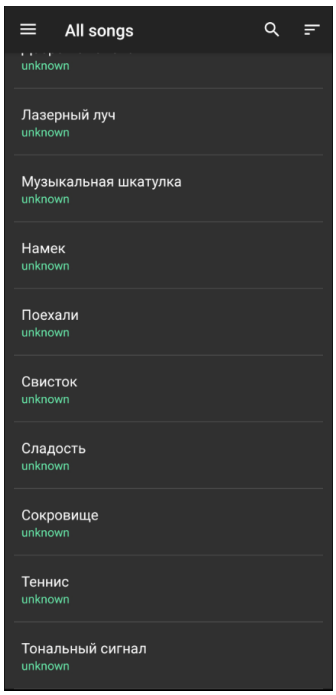

Рис. 3.1.2. Дизайн головного екрану

- Головний екран з увімкненою піснею (Рис. 3.1.3);

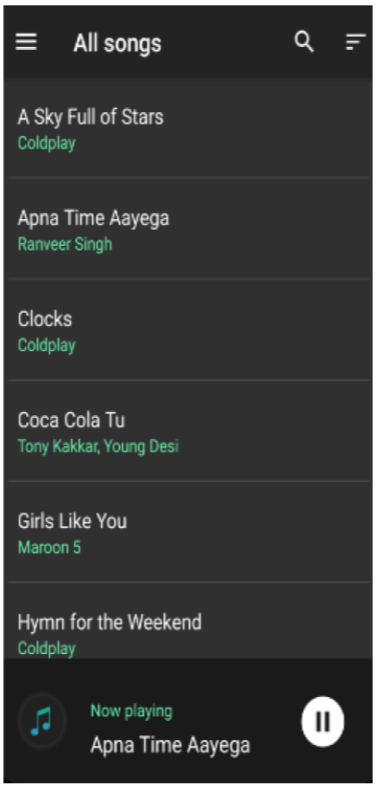

Рис. 3.1.3. Дизайн головного екрану з увімкненою піснею

- Екран відтворення музики (Рис. 3.1.4);

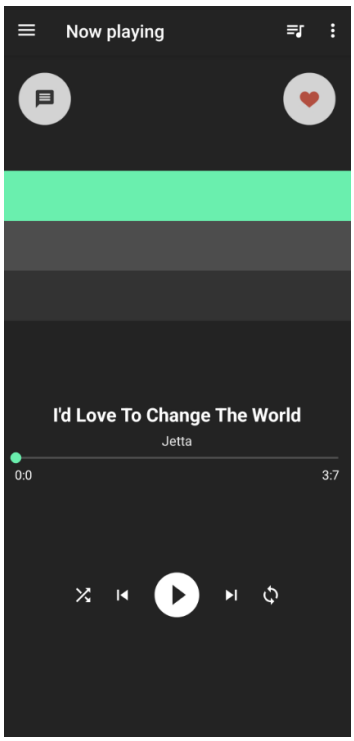

Рис. 3.1.4. Дизайн екрану відтворення музики

- Екран пошуку музики (Рис. 3.1.5);

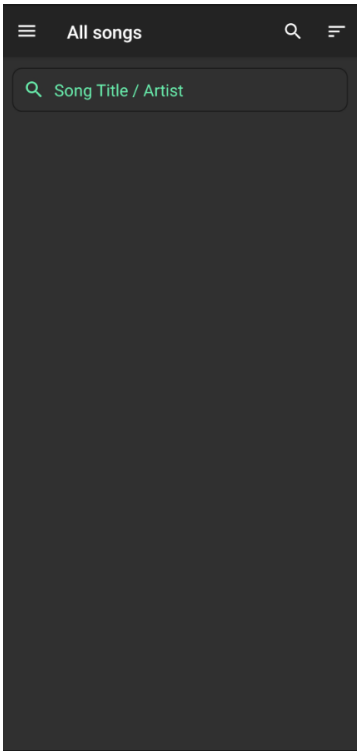

Рис. 3.1.5. Дизайн екрану пошуку музики

- Екран налаштувань (Рис. 3.6);

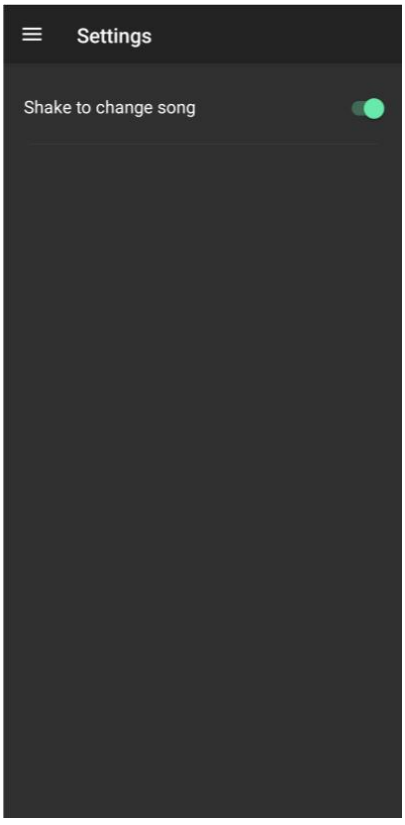

Рис. 3.1.6. Дизайн екрану налаштувань

- Екран з улюбленою музикою (Рис. 3.1.7);

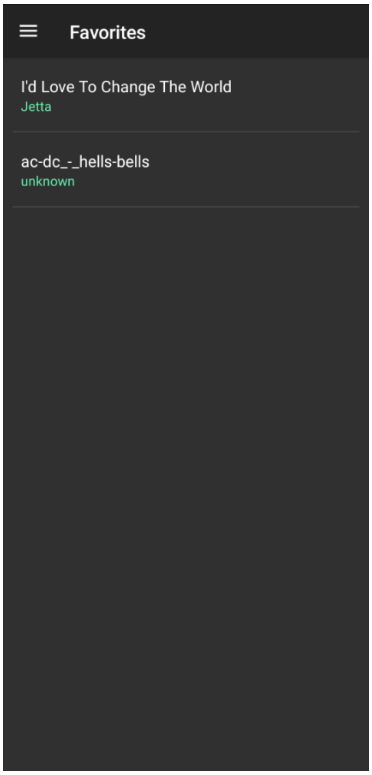

Рис. 3.1.7. Дизайн екрану з улюбленою музикою

- Екран зі словами треку (Рис. 3.1.8);

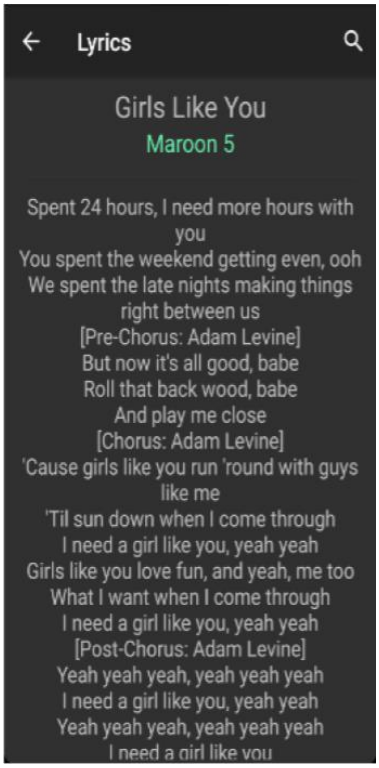

Рис. 3.1.8. Дизайн екрану зі словами треку

- Екран з еквалайзером (Рис. 3.1.9);

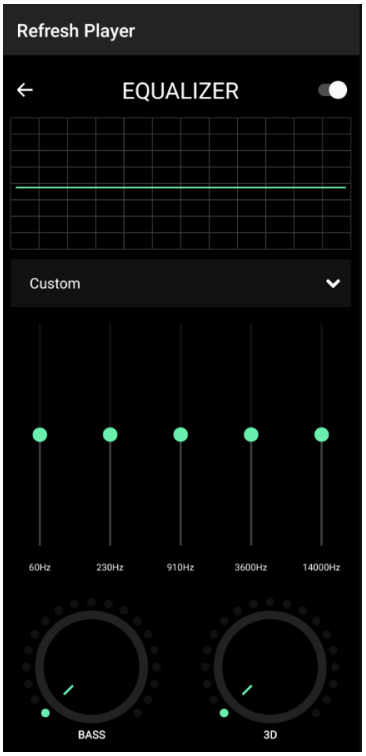

Рис. 3.1.9. Дизайн екрану з еквалайзером

- Екран «Про нас» (Рис. 3.1.10);

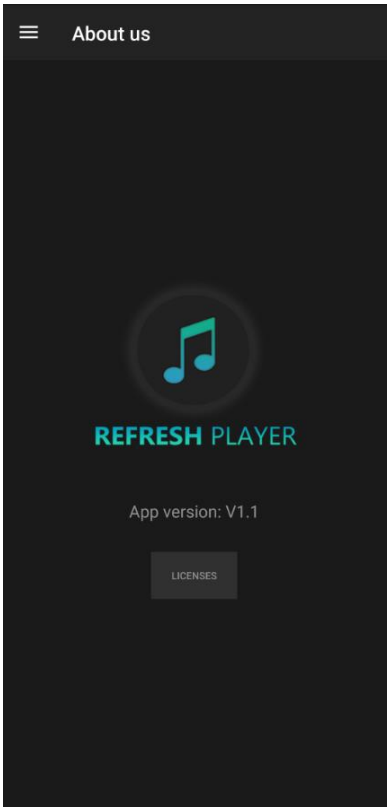

Рис. 3.1.10. Дизайн екрану «Про нас»

Застосування принципів Material Design в проектуванні користувацького інтерфейсу музичного застосунку дозволило досягти зрозумілості та зручності його використання, а також створити єдиний стиль та єдність між всіма елементами інтерфейсу. Material Design - це набір рекомендацій та стандартів дизайну, розроблений компанією Google, який сприяє розумінню та інтуїтивному користуванню програмами та додатками.

За допомогою Material Design було досягнуто чіткості та консистентності у використанні елементів інтерфейсу, таких як кнопки, іконки та макети. Один із ключових аспектів Material Design - це реалістичний підхід до візуального оформлення, який надає відчуття фізичності та простору. Елементи інтерфейсу виглядають так, ніби вони мають глибину та текстуру, що сприяє їх легкому розпізнаванню та взаємодії з користувачем.

Застосування Material Design також сприяє єдності та гармонії між різними частинами інтерфейсу. Колірна палітра, типографіка та просторові відношення елементів відповідають загальним принципам дизайну, що забезпечує єдиний стиль та структурованість. Це допомагає користувачеві легко орієнтуватись в застосунку та швидко освоїти його функціональність.

Таким чином, завдяки використанню принципів Material Design, музичний застосунок отримав зрозумілий, зручний та стильний інтерфейс, що сприяє позитивному користувацькому досвіду та задоволенню від використання додатку.

3.2. Реалізація функціональності

Для реалізації функціональності використовуватимуться потужне інтегроване середовище розробки Android Studio та Java. Android Studio та Java є основними інструментами розробки для платформи Android, що надають широкі можливості для розробників мобільних застосунків.

3.2.1. Знаходження аудіофайлів

Одним з найважливіших етапів реалізації функціоналу є знаходження аудіофайлів на пристрої користувача. Це необхідно для створення медіатеки та можливості та відповідно відтворювати музику.

Для реалізації знаходження аудіофайлів на пристрої користувача потрібно створити відповідну функцію. Приклад цієї функції зображений на Рис. 3.2.1.

```
private fun getSongsfromPhone(): ArrayList<Songs> {
    val arrayList = ArrayList<Songs>()
    val contentResolver = activity?.contentResolver
    val songUri = MediaStore.Audio.Media.EXTERNAL_CONTENT_URI
    val songCursor = contentResolver?.query(songUri, projection: null, selection: null, selectionArgs: null, sortOrder: null)
    try {
        if (songCursor != null && songCursor.moveToFirst()) {
            val songId = songCursor.getColumnIndex(MediaStore.Audio.Media._ID)
            val songTitle = songCursor.getColumnIndex(MediaStore.Audio.Media.TITLE)
            val songArtist = songCursor.getColumnIndex(MediaStore.Audio.Media.ARTIST)
            val songPathInt = songCursor.getColumnIndex(MediaStore.Audio.Media.DATA)
            val dateIndex = songCursor.getColumnIndex(MediaStore.Audio.Media.DATE_ADDED)
            while (songCursor.moveToNext()) {
                val currentId = songCursor.getLong(songId)
                val currentTitle = songCursor.getString(songTitle)
                val currentArtist = songCursor.getString(songArtist)
                val fullpath = songCursor.getString(songPathInt)
                val dateadded = songCursor.getLong(dateIndex)
                arrayList.add(Songs(currentId, currentTitle, currentArtist, fullpath, dateadded))
            }
        }
    }
    catch(e:Exception){}
    return arrayList
}
```

Рис. 3.2.1. Код функції getSongsfromPhone()

Опис роботи функці getSongsfromPhone():

1. Створюється пустий ArrayList з об'єктами типу **Songs**, який буде використовуватися для зберігання інформації про пісні;
2. Отримується **contentResolver** (резольвер вмісту), що дозволяє взаємодіяти з вмістом пристрою;
3. Визначається **songUri** як URI зовнішнього вмісту для аудіофайлів;
4. Виконується запит до **contentResolver** за допомогою **songUri**, що повертає курсор з результатами;

5. Якщо **songCursor** не є нульовим та вміщує щонайменше один рядок (пісню):
    - Отримуються індекси стовпців, що містять інформацію про пісню (ID, назву, виконавця, шлях до файлу та дату додавання);
    - Проходиться через кожен рядок курсора, щоб отримати дані про кожну пісню;
    - Створюється об'єкт **Songs** з відповідними даними про пісню (ID, назва, виконавець, шлях до файлу та дата додавання), і додається в ArrayList;
6. На завершення функція повертає заповнений ArrayList з інформацією про всі пісні, які були знайдені на пристрої.

3.2.2. Відтворення музики

Для реалізації відтворення музики потрібно звернутись до *MediaPlayer*.

MediaPlayer - це клас, який забезпечує базові функції відтворення аудіо та відео файлів в програмах на платформі Android. Для відтворення музики у MediaPlayer є декілька основних методів.

Основний метод для відтворення музики - це **start()**. Після створення об'єкта MediaPlayer і підготовки вхідного файлу за допомогою методу **setDataSource()**, можна викликати метод **start()**, який розпочне відтворення аудіофайлу з початку.

Після виклику **start()** MediaPlayer почне відтворювати музику у фоновому потоці, тобто це не буде блокувати виконання основного потоку програми. MediaPlayer автоматично виконує декодування аудіофайлу та аудіо-буферизацію для забезпечення плавного відтворення.

Крім методу **start()**, також є інші корисні методи MediaPlayer:

- **pause()**: призупиняє відтворення музики тимчасово. Виклик **start()** після **pause()** продовжить відтворення з місця призупинення;
- **stop()**: зупиняє відтворення музики і повертає MediaPlayer в початковий стан.

Для того, щоб використовувати ці методи потрібен обробник події натискання на кнопку відтворення/паузи. Приклад реалізації цього обробника події наведений на Рис. 3.2.2.

Рис. 3.2.2. Код обробника події натискання на кнопку відтворення/паузи

Цей код відповідає за обробку події натискання на кнопку відтворення/паузи (**playpauseImageButton**). Ось опис його функціональності:

1. При натисканні на кнопку перевіряється, чи відтворюється музика в даний момент (**mediaPlayer?.isPlaying**).
2. Якщо музика відтворюється, виконуються наступні дії:
    - Змінюється стан відтворення в currentSongHelper.isPlaying, встановлюючи його на true;
    - Встановлюється фон кнопки відтворення/паузи на значок "Play" за допомогою **playpauseImageButton?.setBackgroundResource(R.drawable.play)**;
    - Викликається метод **mediaPlayer?.pause()**, щоб призупинити відтворення музики.
3. Якщо музика не відтворюється, виконуються наступні дії:
    - Змінюється стан відтворення в **currentSongHelper.isPlaying**, встановлюючи його на **false**;
    - Встановлюється фон кнопки відтворення/паузи на значок "Pause" за допомгою **playpauseImageButton?.setBackgroundResource(R.drawable.pause)**;
    - Викликається метод **mediaPlayer?.seekTo(seekbar?.progress as Int)**, щоб перейти до поточної позиції відтворення;

- Викликається метод **mediaPlayer?.start()**, щоб продовжити відтворення музики.

3.2.3. Пошук музики

З точки зору користувача для пошуку пісні або виконавця він вводить назву пісні або ім’я автора. З точки зору розробника текст, що він ввів є ключовим словом **key**. Через це, реалізація пошуку композиції або ж артиста і його пісень буде ґрунтуватись на ключовому слові. Для цього потрібно написати функцію, код якої наведений нижче:

```
private fun final_list(key : String):ArrayList<Songs>{
    var result:ArrayList<Songs> = ArrayList()
    try{
        var old : ArrayList<Songs> = getSongsfromPhone()
        for (song in old){
            if((song.songTitle.contains(key,ignoreCase = true)) || song.artist.contains(key,ignoreCase = true)){
                result.add(song)
            }
        }
        songList =result
        return result
    }
    catch (e : java.lang.Exception){
        songList =result
        return result
    }
}
```

Рис.3.2.3. Код функції **final_list**, яка здійснює пошук музики

Цей код, що зображений на Рис.3.2.3 представляє функцію **final_list**, яка отримує ключове слово **key** і повертає список пісень, що відповідають цьому ключовому слову. Ось опис його функціональності:

1. У функції створюється порожній список **result**, в який будуть додаватись пісні, що відповідають ключовому слову;
2. Викликається функція **getSongsfromPhone()**, яка повертає список всіх доступних пісень;
3. У циклі проходиться крізь кожну пісню в отриманому списку **old**;

4. Для кожної пісні перевіряється, чи містить її назва або виконавець ключове слово, використовуючи метод **contains** з параметром **ignoreCase = true** для ігнорування регістру;
5. Якщо пісня відповідає ключовому слову, вона додається до списку **result**;
6. На кінці циклу, список **result** стає новим значенням змінної **songList**;
7. Незалежно від виконання циклу, функція повертає список **result**.

Цей код використовується саме для фільтрації списку пісень за ключовим словом, що вводиться користувачем. В результаті повертається список пісень, які містять у своїх назвах або виконавцях введене ключове слово.

3.2.4. Управління відтворенням музики

Управління відтворенням музики є важливою складовою багатьох програм, які працюють з аудіофайлами. Користувачі хочуть мати можливість контролювати відтворення музики відповідно до своїх вподобань і потреб. Тому необхідно додати такий функціонал як: перемішування пісень, зациклення відтворення, перехід до наступної або попердньої пісні. Для реалізації цього будуть написані відповідні функції. Приклади цих функцій будуть наведені нижче.

```
shuffleImageButton?.setOnClickListener({ it: View!

    val editorShuffle = Statified.activity?.getSharedPreferences(MY_PREFS_SHUFFLE, MODE_PRIVATE)?.edit()
    val editorLoop = Statified.activity?.getSharedPreferences(MY_PREFS_LOOP, MODE_PRIVATE)?.edit()
    if (currentSongHelper.isShuffle) {
        shuffleImageButton?.setBackgroundResource(R.drawable.shuffle_off)
        currentSongHelper.isShuffle = false
        editorShuffle?.putBoolean("feature", false)
        editorShuffle?.apply()
    } else {
        currentSongHelper.isShuffle = true
        currentSongHelper.isLoop = false
        shuffleImageButton?.setBackgroundResource(R.drawable.shuffle_on)
        loopImageButton?.setBackgroundResource(R.drawable.loop_off)
        editorShuffle?.putBoolean("feature", true)
        editorShuffle?.apply()
        editorLoop?.putBoolean("feature", false)
        editorLoop?.apply()
    }

})
```

Рис.3.2.4. Код функції, що здійснює перемішування пісень

```
loopImageButton?.setOnClickListener({ it: View!

    val editorLoop = Statified.activity?.getSharedPreferences(MY_PREFS_LOOP, MODE_PRIVATE)?.edit()
    val editorShuffle = Statified.activity?.getSharedPreferences(MY_PREFS_SHUFFLE, MODE_PRIVATE)?.edit()
    if (currentSongHelper.isLoop) {
        currentSongHelper.isLoop = false
        loopImageButton?.setBackgroundResource(R.drawable.loop_off)
        editorLoop?.putBoolean("feature", false)
        editorLoop?.apply()
    } else {
        currentSongHelper.isLoop = true
        currentSongHelper.isShuffle = false
        loopImageButton?.setBackgroundResource(R.drawable.loop_on)
        shuffleImageButton?.setBackgroundResource(R.drawable.shuffle_off)
        editorLoop?.putBoolean("feature", true)
        editorLoop?.apply()
        editorShuffle?.putBoolean("feature", false)
        editorShuffle?.apply()
    }

})
```

Рис.3.2.5. Код функції, що здійснює зациклення відтворення

```
fastforwardImageButton?.setOnClickListener({ it: View!
    currentSongHelper.isPlaying = true
    if (currentSongHelper.isShuffle) {
        SongPlayingFragment.Staticated.playNext( check: "playNextLikeNormalShuffle")
    } else {
        SongPlayingFragment.Staticated.playNext( check: "PlayNextNormal")
    }
})
rewindImageButton?.setOnClickListener({ it: View!
    currentSongHelper.isPlaying = true
    if (currentSongHelper.isLoop) {
        loopImageButton?.setBackgroundResource(R.drawable.loop_off)
    }
    currentSongHelper.isLoop = false
    playPrevious()
})
```

Рис.3.2.6. Код функцій, що здійснюють перехід до наступної/попередньої пісні

На перший погляд може здаватись, що код функцій схожий, але різниця полягає в тому, що є різні умови для виконання цих функцій. Наприклад, для функції **fastForwardImageButton** потрібно відразу перевірити чи увімкнене перемішування пісень, відповідно, якщо воно увімкнене, то наступна пісня буде випадкова, якщо ні, тоді програватиметься наступна пісня з поточного плейлисту.

Щодо функції **rewindImageButton**, що відповідає за реалізацію увімкнення попердньої пісні, то метод роботи відносно схожий, але він взаємодіє з функцією, що здійснює зациклення відтворення. Щоб увімкнути попередню композицію, потрібно перевірити чи увімкнена функція **loop**, якщо так, то спочатку вимикається ця функція, а потім вмикається попердня пісня. Це зроблено для того, щоб користувач не натискав на кнопку **rewindButton** декілька разів.

3.2.5. Налаштування звуку

Для налаштування звуку використовується Android Equalizer API для надання користувачам можливості налаштування звуку в музичному застосунку. Це API дозволяє змінювати параметри звукової обробки, такі як еквалайзер, баланс, гучність тощо, що дозволяє користувачам настроювати звучання музики під свої особисті вподобання.

```
var equalizerFragment : EqualizerFragment?=null

override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_equalize)
    try{
    val sessionId = mediaPlayer?.getAudioSessionId();
    equalizerFragment=EqualizerFragment.newBuilder()
            .setAccentColor(Color.parseColor( colorString: "#6aefae"))
            .setAudioSessionId(sessionId as Int)
            .build();
    getSupportFragmentManager().beginTransaction()
            .replace(R.id.eqFrame, equalizerFragment)
            .commit();
    mediaPlayer?.setLooping(true);}
    catch (e:Exception)
    {
        var main = Intent( packageContext: this@EqualizeActivity,MainActivity::class.java)
        startActivity(main)
        Toast.makeText( context: this, text: "You must play a song first!",Toast.LENGTH_SHORT).show()
    }
}
```

Рис. 3.2.7. Код EqualizeActivity - Активність еквалайзера

Код встановлює вміст макету активності на основі activity_equalize.xml, створює екземпляр EqualizerFragment з використанням ідентифікатора аудіо-сесії mediaPlayer, замінює контейнер для фрагменту і налаштовує mediaPlayer для

відтворення в режимі повторення. Це дозволяє відображати і налаштовувати еквалайзер для поточного відтворюваного аудіо. У разі виникнення помилки або відсутності активного відтворення відбувається перехід до активності MainActivity та виводиться сповіщення Toast про необхідність спочатку відтворити пісню.

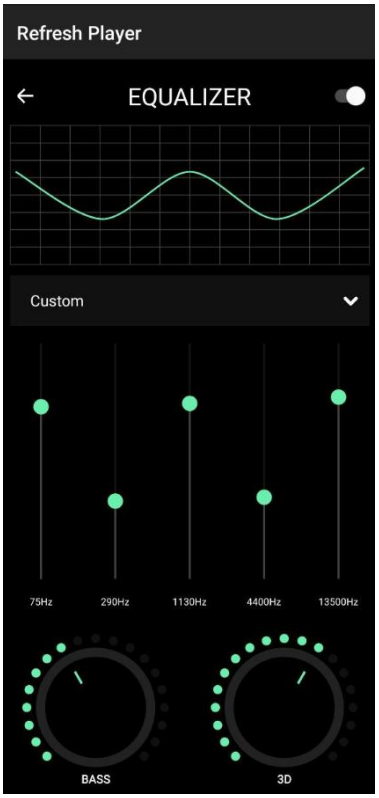

Рис.3.2.8. Вигляд еквалайзеру з деяким налаштуванням звуку

3.2.5. Додатковий функціонал застосунку

Для розширення функціоналу було вирішено додати функцію під назвою «Shake to change», суть якої полягає в тому, щоб переключити пісню необхідно потрясти пристроєм користувача. Для цього потрібно написати функцію щоб відслідковувати рух пристрою за допомогою акселерометра і спрацьовувати певні дії, якщо виявлено сильний рух (прискорення більше 12). На Рис. 3.2.9 наведений код для реалізації цього функціоналу.

```
fun bindShakeListener() {
    try {
        Statified.mSensorListener = object : SensorEventListener {
            override fun onAccuracyChanged(p0: Sensor?, p1: Int) {
            }
            override fun onSensorChanged(p0: SensorEvent) {
                val x = p0.values[0]
                val y = p0.values[1]
                val z = p0.values[2]
                mAccelerationLast = mAccelerationCurrent
                mAccelerationCurrent = Math.sqrt(((x * x + y * y + z * z).toDouble())).toFloat()
                val delta = mAccelerationCurrent - mAccelerationLast
                mAcceleration = mAcceleration * 0.9f + delta
                if (mAcceleration > 12) {
                    val prefs = Statified.activity?.getSharedPreferences(Statified.MY_PREFS_NAME, Context.MODE_PRIVATE)
                    val isAllowed = prefs?.getBoolean("feature", false)
                    Statified.currentSongHelper?.isPlaying = true
                    Statified.playpauseImageButton?.setBackgroundResource(R.drawable.pause)
                    var editorLoop = Statified.activity?.getSharedPreferences(Statified.MY_PREFS_LOOP, Context.MODE_PRIVATE)?.edit()
                    if (Statified.currentSongHelper?.isLoop as Boolean) {
                        Statified.currentSongHelper?.isLoop = false
                        editorLoop?.putBoolean("feature", false)
                        editorLoop?.apply()
                        Statified.loopImageButton?.setBackgroundResource(R.drawable.loop_off)
                        Toast.makeText(Statified.activity,  text: "Loop Disabled", Toast.LENGTH_SHORT).show()
                    }
                    if (isAllowed as Boolean) {
                        if (Statified.currentSongHelper?.isShuffle as Boolean) {
                            Staticated.playNext( check: "PlayNextLikeNormalShuffle")
                        } else {
                            Staticated.playNext( check: "PlayNextNormal")
                        }
```

Рис.3.2.9. Код для реалізації «Shake to change»

Дії, що спрацьовують при сильному русі пристроєм:

1. Перевіряє, чи дозволено виконання дій під час руху (зчитує значення з Shared Preferences);
2. Встановлює прапорець, що пісня відтворюється (currentSongHelper.isPlaying) на значення true;
3. Змінює зображення кнопки відтворення/паузи (playpauseImageButton);
4. Перевіряє, чи включений режим повторення (loop). Якщо так, вимикає його і відображає відповідну іконку;
5. Виконує відповідну дію залежно від дозволених дій:
    - Якщо включений режим перемішування, викликає функцію playNext() з параметром "PlayNextLikeNormalShuffle" для відтворення наступної випадкової пісні;
    - Якщо режим перемішування вимкнено, викликає функцію playNext() з параметром "PlayNextNormal" для відтворення наступної пісні звичайним порядком.

Цей код дозволяє реалізувати функціональність відтворення наступної пісні при виявленні руху пристрою.

Також для розширення функціоналу було додано функцію, що дозволяє користувачеві переглянути текст(слова) композиції, якщо вони наявні.

```
protected String doInBackground(String... params) {
    String song = params[0];
    String artist = params[1];
    String url = "https://www.google.com/search?&q="+artist+"+"+song+"+lyrics+metrolyrics";
    try {
        Document googlePage = Jsoup.connect(url).get();
        elem1 = googlePage.getElementsByClass( className: "g").first();
        data = elem1.getElementsByTag( tagName: "a").first();
        int http=data.toString().indexOf("http");
        int html=data.toString().indexOf("html");
        metroLyrics = data.toString().substring(http,html+4);
        Document document = Jsoup.connect(metroLyrics).get();
        Elements elem = document.getElementsByClass( className: "verse");
        for (Element item:elem) {
            for (Element j : item.getAllElements()) {
                ans += j.wholeText();
            }
            ans += "\n";
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
    return ans;
}
```

Рис. 3.2.10. Код функції для знаходження слів пісні

Основні кроки, які він виконує:

1. Отримує параметри пісні та виконавця з переданих аргументів;
2. Формує URL-адресу для пошуку тексту пісні на веб-сайті MetroLyrics, використовуючи пошуковий запит з назвою пісні та виконавцем;
3. Виконує HTTP-запит до Google, щоб отримати результати пошуку;
4. Витягує посилання на MetroLyrics з результатів пошуку Google;
5. Виконує HTTP-запит до MetroLyrics, щоб отримати сторінку з текстом пісні;
6. Скраплює (витягує) тексти пісень з відповідного елементу HTML сторінки;
7. Збирає всі скраплені тексти пісень разом в одну рядок (змінна ans);
8. Повертає отриманий рядок з текстом пісні.

Отриманий текст пісні буде використаний для подальшої обробки і відображення користувачу.

Для створення більш інтерактивного музичного застосунку було також додано візуалізацію звуку за допомогою API Cleveroad Visualizer. Це дозволить створити захоплюючі візуальні ефекти, які синхронізуються зі звуковим потоком і нададуть відмінну візуальну динаміку музичному застосунку. Вигляд візуальних ефектів зображений на Рис. 3.2.11.

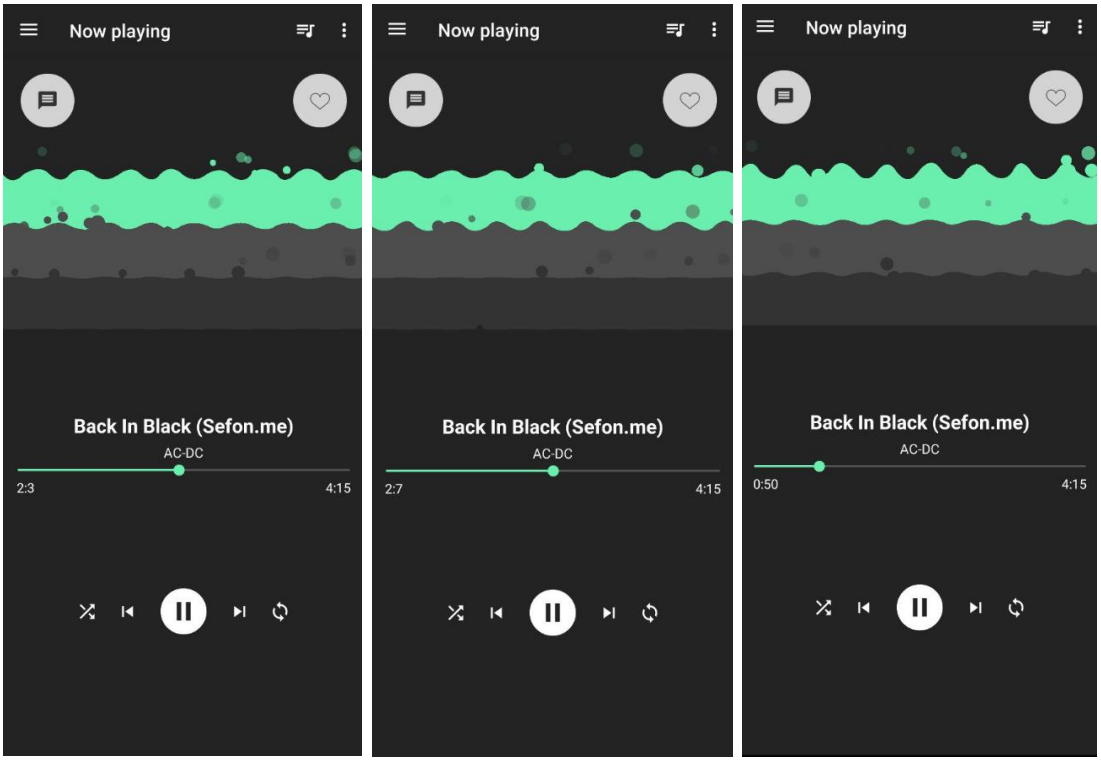

Рис. 3.2.11. Приклад візуалізації звуку за допомогою API Cleveroad Visualizer

3.3. Висновки до розділу

У даному розділі була проведена робота над розробкою користувацького інтерфейсу та реалізацією функціональності музичного застосунку. Під час розробки було визначено структуру та навігацію застосунку, що дозволяє забезпечити зручне та логічне взаємодію з користувачем. Також був розроблений дизайн інтерфейсу, використовуючи *Figma*, що сприяє єдності та зрозумілості елементів інтерфейсу. Застосування принципів *Material Design* сприяло створенню

єдиного стилю та єдності між елементами інтерфейсу, що позитивно впливає на користувацький досвід. Застосунок володіє функціональністю, яка задовольняє потреби користувачів, та забезпечує високий рівень користувацької задоволеності.

Під час реалізації функціональності музичного застосунку були використані різні технології та інструменти, зокрема *Android Studio, API Cleveroad Visualizer* та *Android Equalizer*. Застосування цих інструментів дало змогу створити потужний та функціональний застосунок, здатний до відтворення музики, створення та керування плейлистами, пошуку та фільтрації музики, а також налаштування звуку.

Процес знаходження аудіофайлів на пристрої був успішно реалізований завдяки використанню функціоналу *Android Studio* та відповідних API. Це забезпечило можливість створення медіатеки та доступу до різноманітних музичних файлів на пристрої користувача.

В процесі реалізації функціональності було успішно реалізовано відтворення музики, пошук музики, управління відтворенням музики, налаштування звуку, а також додатковий функціонал застосунку. Це дозволило забезпечити багатофункціональний та зручний застосунок, що задовольняє потреби користувачів у відтворенні та керуванні музикою.

У результаті розробки було досягнуто поставленої мети - створення музичного застосунку для мобільних пристроїв на платформі Android.

ВИСНОВКИ

У ході розробки даної дипломної роботи було проведено детальний аналіз предметної області, а саме розробки музичних застосунків на платформі Android. Було досліджено ключові аспекти створення мобільних застосунків, зокрема платформу Android, мови програмування Java та Kotlin, мову розмітки XML, а також API та інструменти розробки, зокрема Android Studio. Цей аналіз дав змогу зрозуміти основні принципи та технології, що використовуються при розробці музичних застосунків на даній платформі.

У результаті проведеного дослідження було встановлено, що розробка музичних застосунків на платформі Android вимагає глибокого розуміння архітектурних підходів та вміння ефективно використовувати функціональні можливості платформи. Під час проектування архітектури музичного застосунку було обрано підхід Model-View-Controller (MVC), який дозволяє чітко визначити роль кожного компонента та забезпечити їх взаємодію. Також розглянуто альтернативний підхід Model-View-ViewModel (MVVM), який може бути використаний для подальших розширень та покращень музичного застосунку.

Одним із важливих аспектів реалізації музичного застосунку є функціональність, яка дозволяє користувачам насолоджуватися відтворенням музики, знаходити потрібні аудіофайли, керувати відтворенням та налаштовувати звукові параметри. Було успішно реалізовано функціональність знаходження аудіофайлів на пристрої, їх відтворення, пошуку та фільтрації музики, а також управлінням відтворенням та налаштуванням звуку.

Застосування пошуку дозволяє користувачам швидко знаходити потрібні треки за назвою, виконавцем. Це дозволяє користувачам з легкістю організовувати та знаходити свою улюблену музику у великій колекції треків.

Налаштування звуку є ще одним важливим аспектом музичного застосунку. Користувачі повинні мати можливість налаштовувати звукові параметри, такі як еквалайзер, гучність, ефекти звуку та інші. У розробленому застосунку було

реалізовано можливість налаштовування звуку, дозволяючи користувачам змінювати налаштування згідно зі своїми вподобаннями та акустичними потребами.

Під час розробки було використано популярні фреймворки та API, зокрема Android Media Player для відтворення музики та Android Equalizer для налаштування звуку. Такі інструменти розробки, як Android Studio, забезпечили зручну роботу над проектом та високу продуктивність.

В цілому, розроблений музичний застосунок на платформі Android демонструє успішну реалізацію задуманої функціональності та архітектури. Він відповідає вимогам, поставленим до музичних застосунків, та забезпечує зручну навігацію, ефективне відтворення музики та можливість налаштування звуку.

У подальшому розвитку даної роботи можна розглядати можливість розширення функціональності застосунку, вдосконалення його дизайну та розгортання на різних пристроях та версіях платформи Android. Також, варто розглянути можливість інтеграції з іншими сервісами та соціальними мережами для покращення користувацького досвіду та розширення функціональних можливостей застосунку.

Таким чином, дана дипломна робота успішно виконана, поставлені завдання досягнуті, а розроблений музичний застосунок на платформі Android відповідає вимогам та виконує потрібну функціональність. Результати роботи мають практичне значення та можуть бути використані як основа для подальших досліджень та розробок в галузі мобільних музичних застосунків.

СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Бойченко С. В., Положення про дипломні роботи (проєкти) випускників Національного авіаційного університету. / С. В. Бойченко, О. В. Іванченко. – Київ: НАУ, 2017. – 63 с.
2. Типи мобільних додатків [Електронний ресурс] – Режим доступу: https://smile-ukraine.com/ua/mobile-apps/mobile-apps-types. (Дата звернення: 23.05.2023) – Назва з екрана.
3. Васильєв О. Програмування мовою Java. / О. Васильєв. – Київ: Видавництво "Навчальна книга Богдан", 2020. – 496 с.
4. Android Studio: An IDE built for Android [Електронний ресурс] – Режим доступу: *https://android-developers.googleblog.com/2013/05/android-studio-ide-builtforandroid.html.* (Дата звернення: 23.05.2023) – Назва з екрана.
5. Pricing & Plans For Professional Teams. Figma is free for individuals. [Електронний ресурс] – Режим доступу: *https://www.figma.com/pricing*. (Дата звернення: 25.05.2023) – Назва з екрана.
6. Хорстманн Дж. Big Java: Early Objects. / Дж. Хорстманн. – Посібник. – Київ: Видавництво "Фабула", 2023. – 200 с.
7. Oracle. "Java SE Documentation" [Електронний ресурс]. Режим доступу: https://docs.oracle.com/en/java/javase/. (Дата звернення: 25.05.2023) – Назва з екрана.
8. Vogella. "Android Development Tutorial" [Електронний ресурс]. Режим доступу: https://www.vogella.com/tutorials/android.html. (Дата звернення: 26.05.2023) – Назва з екрана.
9. Google Developers. "Android Studio Documentation" [Електронний ресурс]. Режим доступу: https://developer.android.com/studio/documentation. (Дата звернення: 26.05.2023) – Назва з екрана.

10. GitHub. "Official Android Repository" [Електронний ресурс]. Режим доступу: https://github.com/android. (Дата звернення: 25.05.2023) – Назва з екрана.
11. Figma. "Official Figma Documentation" [Електронний ресурс]. Режим доступу: https://www.figma.com/resources/docs/. (Дата звернення: 26.05.2023) – Назва з екрана.
12. Stack Overflow. "Java Questions and Answers" [Електронний ресурс]. Режим доступу: https://stackoverflow.com/questions/tagged/java. (Дата звернення: 26.05.2023) – Назва з екрана.
13. Android Developers Blog. "Official Android Developers Blog" [Електронний ресурс]. Режим доступу: https://android-developers.googleblog.com/. (Дата звернення: 26.05.2023) – Назва з екрана.
14. CodePath. "Android Guides and Tutorials" [Електронний ресурс]. Режим доступу: https://guides.codepath.com/android. (Дата звернення: 30.05.2023) – Назва з екрана.
15. Medium. "Android Development Stories and Articles" [Електронний ресурс]. Режим доступу: https://medium.com/tag/android. (Дата звернення: 31.05.2023) – Назва з екрана.
16. Android Weekly. "Android Development Newsletter" [Електронний ресурс]. Режим доступу: https://androidweekly.net/. (Дата звернення: 31.05.2023) – Назва з екрана.
17. Javatpoint. "Java Tutorial" [Електронний ресурс]. Режим доступу: https://www.javatpoint.com/java-tutorial. (Дата звернення: 31.05.2023) – Назва з екрана.
18. Udacity. "Android Development Courses" [Електронний ресурс]. Режим доступу: https://www.udacity.com/courses/android. (Дата звернення: 31.05.2023) – Назва з екрана.