НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп'ютерних систем

До захисту допущено

Завідувач кафедри
_____ Віталій РОМАНКЕВИЧ
(підпис)
"___" _____ 2021 р.

Дипломний проєкт

на здобуття ступеня бакалавра

по спеціальності **123 «Комп'ютерна інженерія»**

на тему: Андроїд-додаток «Слухай» на мові програмування Kotlin

Виконав :

студент IV курсу, групи КВ-72

Цимбалюк Андрій Олександрович _ _____
(підпис)

Керівник асистент каф. СПіСКС Радченко К.О. _ _____
(підпис)

Консультант з нормоконтролю, доц.каф.СПіСКС, к.т.н. Клятченко Я.М. _____
(підпис)

Рецензент _____
(підпис)

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2021 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп'ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Спеціальність 123 «Комп'ютерна інженерія»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Віталій РОМАНКЕВИЧ
(підпис) (ініціали, прізвище)
«___» _____2021 р.

ЗАВДАННЯ
на дипломний проєкт студента
Цимбалюка Андрія Олекснадровича

1. Тема проєкту Андроїд-додаток «Слухай» на мові програмування Kotlin, керівник проєкту Радченко К.О., асистент кафедри системного програмування і спеціалізованих комп'ютерних систем, затверджені наказом по університету від «___» _____ 2021 р. №_____

2. Термін подання студентом проєкту 25 травня 2021 р

3. Вихідні дані до проєкту: див. Технічне завдання

4. Зміст пояснювальної записки
   - Аналіз технологій розроблення Android-додатків.
   - Аналіз і вибір засобів реалізації.
   - Розробка додатку.

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо)
   - Структурна схема Android-додатка
   - Алгоритм взаємодії з користувачем
   - Діаграма класів проєкту.
   - Діаграма бази даних проєкту.
   - Презентація за темою роботи.

6. Консультанти розділів проєкту*

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Клятченко Я.М., к.т.н., доцент каф. СПіСКС		

7. Дата видачі завдання:

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Вивчення літератури за тематикою проєкту	16.11.2020	
2	Розроблення та узгодження технічного завдання	30.11.2020	
3	Аналіз існуючих рішень	10.02.2021	
4	Підготовка матеріалів першого розділу дипломного проєкту	23.04.2021	
5	Підготовка матеріалів другого розділу дипломного проєкту	28.04.2021	
6	Підготовка матеріалів третього розділу дипломного проєкту	3.05.2021	
7	Підготовка матеріалів четвертого розділу дипломного проєкту	8.05.2021	
8	Підготовка графічної частини дипломного проєкту	14.05.2021	
9	Оформлення документації дипломного проєкту	17.05.2021	
10	Попередній огляд матеріалів диплому на кафедрі	25.05.2021	

Студент _____ Андрій Цимбалюк
(підпис)

Керівник проєкту _____ Костянтин Радченко
(підпис)

* Консультантом не може бути зазначено керівника дипломного проєкту.

АНОТАЦІЯ

Кваліфікаційна робота включає пояснювальну записку (61 с., 33 рис., 4 додатки).

Об’єкт розробки – Android-додаток «Слухай» для прослуховування онлайн аудіо-творів. Можливість створення профілю користувача для збереження вподобаних творів та оцінювання, можливість сортування, пошук в списку та вибір по категоріях. Зручний інтерфейс та зрозумілий в користуванні.

Розроблений додаток надає:

- онлайн прослуховування аудіо;
- можливість зберегти аудіо які сподобались;
- створення профілю користувача;
- оцінювання творів;
- пошук в списку творів;
- відображення вибраних категорій.

Для розробки використана мова програмування Kotlin з використанням фреймворку Android та середовище розробки Google Android Studio. У якості бази даних використано Firebase.

В ході виконання дипломного проєкту:

- проведено аналіз існуючих рішень;
- визначено архітектуру системи
- розроблений зручний Android-додаток для прослуховування аудіо.

Впровадження цього додатку дозволить комфортніше прослуховувати аудіо-твори для користувачів смартфонів на ОС Android.

Ключові слова: АНДРОЇД ДОДАТОК, KOTLIN, JETBRAINS, ANDROID STUDIO, ANDROID, FIREBASE, БАЗА ДАНИХ, ANDROID STUDIO, AUDIO, AUDIO BOOK.

ABSTRACT

Qualification work includes an explanatory note (61 c., 33 fig, 4 appendices).

The object of development is the Android application "Listen" for listening to online audio works in Ukrainian. Ability to create a user profile to save your favorite works and evaluation, the ability to sort the list and display by category. The user-friendly interface is easy to use.

The developed application provides:

- online audio listening;
- the ability to save the audio you like;
- creating a user profile;
- evaluation of works;
- search in the list of works;
- display selected categories.

Kotlin programming language using Android framework and Google Android Studio development environment was used for development. Firebase is used as the database.

During the implementation of the diploma project:

- analysis of existing solutions;
- the system architecture is defined
- developed a convenient Android application for listening to audio.

The introduction of this application will make it more comfortable to listen to audio works for smartphone users on OC Android.

Keywords: ANDROID APPENDIX, KOTLIN, JETBRAINS, ANDROID STUDIO, ANDROID, FIREBASE, AUDIO, DATABASE, ANDROID STUDIO, AUDIO BOOK.

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
			Документація загальна			
			Новорозроблена			
	А4	ІАЛЦ. 467100.002 ТЗ	Android-додаток «Слухай»			
			мовою програмування			
			Kotlin			
			Технічне завдання			
	А4	ІАЛЦ. 467100.003 ТП	Android-додаток «Слухай»			
			мовою програмування			
			Kotlin			
			Відомість технічного			
			проекту			
	А4	ІАЛЦ. 467100.004 ПЗ	Android-додаток «Слухай»			
			мовою програмування			
			Kotlin			
			Пояснювальна записка			
	А4	ІАЛЦ. 467100.005 Д1	Android-додаток «Слухай»			
			мовою програмування			
			Kotlin			
			Схема структурна			

Змін	Арк.	№ докум.	Підпис	Дата	ІАЛЦ. 467100.001 ОА		
Розробив		Цимбалюк А.О.			Android-додаток «Слухай» мовою програмування Kotlin Опис альбому	Літ.	Аркуш 1 / Аркушів 2
Перевірив		Радченко К.О.					
Н. контроль		Клятченко Я.М.				НТУУ "КПІ" ФПМ КВ-72	
Затвердив		Романкевич В.О.					

ЗМІСТ

ДОДАТКИ

Додаток 1. Копії графічного інтерфейсу

- ІАЛЦ.467100.005 Д1. Android-додаток «Слухай» мовою програмування Kotlin Схема Activities. Схема структурна.
- ІАЛЦ.467100.006 Д2. Android-додаток «Слухай» мовою програмування Kotlin. Діаграма взаємовикликів класів. Схема структурна.
- ІАЛЦ.467100.007 Д3 Android-додаток «Слухай» мовою програмування Kotlin. База даних. Схема структурна.
- ІАЛЦ.467100.008 Д4. Android-додаток «Слухай» мовою програмування Kotlin. Діаграма класів. Схема структурна.

Додаток 2. Фрагменти програмного коду

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ ТА ТЕРМІНІВ

ANDROID – Операційна система смартфону;

Android Studio- IDE- Середовище для розробки Android-додатків;

API – Application Programming Interface;

FIREBASE – Серверний сервер та база даних для роботи додатку;

Glide- Модуль для завантаження зображень в додатку;

JVM – Java Virtual Machine;

Kotlin – Мова програмування, яка використовується в даному проєкті для розробки на базі Android;

MediaPlayer- Плеєр для завантаження та відтворення аудіо в додатках;

MVC- Патерн проектування програм (Mode-View-Contoller);

ОС – Операційна система;

SDK – Software Development Kit;

Realtime Database- Онлайн база даних, побудована на архітектурі NoSQL;

UI – User Interface – Інтерфейс користувача;

ПЗ – Програмне забезпечення;

ВСТУП

На сьогоднішній день смартфони складають 80% інтернет трафіку. Люди ведуть через них бізнес, блоги, продажі, спілкуються та інші справи, тому розвиток контенту на мобільні пристрої являється пріоритетним для розробників. Виходом для цього є розробка додатків, через які можливе все що було наведено вище. Android-додаток являє собою різні екрани між якими переключається користувач та взаємодіє з ними. Порівняно з сайтами, додатки мають більш активне управління чим краще утримують користувачів, також вони мають інструмент лояльності чим і виглядають більш перспективнішими в розробці. Провідними компаніями було досліджено, що користувачів приваблюють додатки які містять зрозумілий в користуванні UI, що включає в себе яскраві кольори та привабливі анімації, адаптивний інтерфейс, підтримка великої кількості пристроїв та синхронізація між різними ними.

У даному випадку, дуже важливим є виконання наведених критеріїв, та розробка додатку для великої аудиторії.

Більшість додатків мають незрозумілий в користуванні UI, або не підтримуються на старших версіях ОС Android, тому було розроблено додаток який не має наведених проблем, і дає можливість прослуховувати твори українських авторів.

1. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ МЕДІА-ДОДАТКІВ

1.1 Основні визначення

Android - це операційна система для смартфонів та багатьох інших пристроїв, що базується на ядрі Linux.

Основним елементом цієї операційної системи є реалізація Dalvik віртуальної машини Java, і все програмне забезпечення базується на цій реалізації Java.

Android Inc. була заснована в 2003 році і куплена компанією Google у 2005 році. Перша версія Android була випущена в 2008 році і отримала назву 1.0 Astroboy. На сьогоднішній день актуальна версія 12 S, яка була представлена в 2021 році. Телефон для Android розробляється і підтримується Open Handset Alliance (OHA).

Першим пристроєм з операційною системою Android став смартфон HTC Dream, який був розроблений HTC в 2003 році.

У 2007 році OHA представила набір для розробки програмного забезпечення для Android, який включав засоби розробки та налагодження, документацію, приклади, емулятор та навчальний посібник. Розробка вимагала завантаження Android SDK для платформ x86.

У 2014 році ОС Android було встановлено на 84% проданих смартфонів. У 2017 році вона стала найпопулярнішою операційною системою для доступу до Інтернету. Дана операційна система має великий потенціал і розвивається так, що розробка додатків для неї завжди буде актуальною.

З 2008 року Android пройшов численні оновлення, які поступово вдосконалювали операційну систему, додаючи нові функції та виправляючи помилки в попередніх версіях. І тепер кодова назва кожної основної версії Android, починаючи з версії 1.5, - це назва десерту. Перші літери імен у порядку версій відповідають буквам латинського алфавіту.

Рис. 1 – Дата виходу версій Android

Android використовує бази даних SQLite для структурованих даних.

Програми, написані на Java, можуть бути скомпільовані в біт-код Dalvik і запускатися на віртуальній машині Dalvik, віртуальній машині, спеціально розробленій для використання на мобільних телефонах, навіть якщо це не стандартна JVM.

1.2 Аналіз існуючих реалізацій

На сьогоднішній день написано безліч додатків для ОС Android, є безліч веб-ресурсів та статей про їх написання. Найскладнішим являється поставлення ТЗ для розробника та розробка індивідуального зовнішнього вигляду, що буде асоціюватись з розробленим продуктом. Якщо дивитись на вже відомі додатки стрімінгу аудіо можна назвати такі як Apple Music, YouTube Music, Spotify, SoundCloud та інші... Якщо ж взяти подкаст-додатки то можна назвати лише Apple подкасти але це трохи інший вид додатків. Саме медіа додатків аудіо-книг немає, таких що мають велику популярність, це означає що даний контент є направленим лише для певних груп користувачів. Причинами, що такі додатки не популярні може бути незрозумілий інтерфейс, неоптимізований код під модель телефону користувача, не наявність потрібних функцій або ж великий розмір додатку, що відштовхує від завантаження на смартфон через брак пам'яті.

1.3 Висновки

З огляду на все вище наведене, можна сказати, що на сьогоднішній день немає явних лідерів серед медіа-додатків аудіо-книг контенту. А наявні додатки являються не популярними в користувачів. Причиною цього можна назвати недостатню кількість розробників готових цим займатись або ж недостатню кількість контенту в даний момент. І саме тому, метою створення

даного дипломного проєкту – є розробка додатку, який би зацікавлював користувачів контентом, інтерфейсом, був зрозумілий та красивим, мав велику кількість функцій що спрощував би користування, мав можливість синхронізацію між різними пристроями, підтримку великої кількісті пристроїв та мав невеликий розмір додатку.

2. АНАЛІЗ ЗАСОБІВ РОЗРОБКИ ANDROID-ДОДАТКІВ

2.1 Засоби реалізації

Для написання програм з операційною системою Android використовуються такі мови програмування: Java, Kotlin, C ++. За допомогою програмного забезпечення Android (SDK) також можна розробляти на мовах програмування, які не належать до JVM, деякі з них: Go, JavaScript, C, C ++, Assembler. Розробка цих мов програмування вимагає допомоги мовного коду JVM, який може надаватися інструментами з обмеженою підтримкою API. Додатки Android мають розширення .apk або нещодавно додані формати AAB. AAB - це оптимізована версія APK, з меншим розміром. В останні роки Google рекомендує використовувати Kotlin як основну мову для розробки програм для Android, а з 2021 року оголошено, що за допомогою Kotlin ви можете розробляти кроссплатформенні програми в ОС Android та iOS. Для програмістів мова програмування Kotlin є пріоритетною, оскільки вона підтримує функції та файли, загалом написані мовою програмування Java. Причина в тому, що Kotlin та Java компілюються в байт-код, але Java не підтримує Kotlin. Ось чому дипломний проект написаний у Котліні з доповненнями. На момент написання статті поточна версія Kotlin становить 1.5.0.

Kotlin - це статично набрана мова програмування, яка працює на додаток до JVM і розроблена JetBrains. Також складено в JavaScript. Автори поставили за мету створити мову, яка була більш стислою та набраною, ніж Java, і простішою, ніж Scala. Спрощення також призвело до швидшої компіляції та кращої підтримки IDE, ніж Scala. Мова розробляється з 2010 року і була опублікована в 2011 році. Milestone 1 був випущений для користувачів у лютому, що включало доповнення для IDEA. Milestone 2 з підтримкою Android вийшов у червні. Milestone 4 вийшов у грудні 2012 року і забезпечив підтримку Java 7. 15 лютого 2016 року вийшла версія 1.0. З 2017 року вона включена в список офіційно підтримуваних мов для розробки програм для платформи Android. Станом на

2019 рік це рекомендована мова програмування для розробки програм для Android.

Переваги для вибору Kotlin для розробки Android-додатків.

1. Сумісність з Java.

Платформа Java - це в першу чергу екосистема: крім «офіційних» продуктів Oracle, вона включає багато проектів з відкритим кодом: бібліотеки та фреймворки різних профілів, на основі яких будується безліч програм. Тому сумісність із існуючим кодом, написаним на Java, дуже важлива для мови, складеної для цієї платформи. Необхідно, щоб існуючі проекти поступово перетворювались на нову мову, тому не тільки код у Kotlin просто називає код на Java, але і навпаки.

Гарантії статичної правильності. Під час компіляції коду на статично набрану мову існує безліч перевірок, щоб переконатися, що під час виконання не виникають помилки. Наприклад, компілятор Java гарантує, що об'єкти, на яких названий метод, можуть його виконати, тобто ці методи реалізовані у відповідних класах1. Окрім цієї дуже важливої функції, Java, на жаль, не гарантує майже нічого. Це означає, що успішно скомпільовані програми закінчуються помилками виконання (викликають винятки). Яскравим прикладом є встановлення нульового посилання на Null, яке називається NullPointerException під час виконання. Важливою передумовою нової мови є посилення статичних гарантій. Це дозволяє виявити більше помилок під час фази складання і, таким чином, зменшити зусилля при тестуванні.

2. Швидкість складання.

Статичні елементи управління полегшують програмування, але збірка відбувається повільно, і вам потрібно знайти баланс. Досвід створення мов із потужною стандартною системою (найяскравіший приклад - Scala) показує, що такий баланс знайти непросто: компіляція часто є неприпустимо тривалою.

Загалом, така особливість мови, як редагування, може здаватися другорядною, але в промисловому розвитку, коли кількість скомпільованого коду дуже велика, цей фактор дуже важливий. Швидка компіляція, зокрема, є однією з головних переваг Java перед C ++, і Kotlin повинен підтримувати цю перевагу..

3. Лаконічність.

Відомо, що програмісти часто витрачають більше часу на читання коду, ніж на його написання. Ось чому важливо, щоб наявні структури в мові програмування дозволяли писати програми стисло і чітко. Ява вважається церемоніальною мовою (церемоніальна мова - «церемоніальна мова»), а завдання Котліна - поліпшити ситуацію в цьому сенсі. На жаль, суворі методи оцінки стислості мов розвинені слабо, але існують непрямі критерії; Однією з них є можливість створення бібліотек, які тісно співпрацюють з доменними мовами (DSL). Створення таких бібліотек вимагає певної гнучкості синтаксису завдяки кращим конструкціям; найпоширеніші функції вищого порядку, тобто функції, які приймають інші функції як параметри.

4. Доступність для навчання.

Складні статичні тести, гнучкий синтаксис та спрощена конструкція ускладнюють і ускладнюють вивчення мови, тому вам потрібно певною мірою обмежити кількість підтримуваних браузерів, щоб зробити мову доступною для вивчення. Розробники Kotlin, які слухали уроки Scala та інших сучасних мов, а також надто складні концепції, не були включені в цю мову.

5. Інструментальна підтримка.

Сучасні розробники активно використовують різноманітні автоматизовані засоби, центральне положення яких займають інтегровані середовища розробки (IDE). Десятирічний досвід роботи в JetBrains показує, що деякі мовні функції можуть ускладнити інструментальну підтримку. Розробляючи Kotlin, ми

враховуємо цей факт і одночасно створюємо IDE із компілятором.Основні елементи мови

1. Функції.

Kotlin є об'єктно-орієнтованою мовою, але на відміну від Java, ви можете оголошувати функції поза класами. У Java для цього використовуються статичні методи, що фактично призводить до класів наступним чином: їх випадки ніколи не створюються, викликаються лише статичні методи.

Оголошення в Kotlin групуються у просторі імен, і функція може бути оголошена безпосередньо у просторі імен.

Оператору функції передує ключове слово fun, а типи параметрів вказуються після двокрапки після імені параметра. Тип значення функції має такий самий ефект. Цей синтаксис відповідає традиціям мов "функціонального світу", таких як ML та Scala. Це полегшує пропуск стандартних інструкцій, якщо тип можна автоматично вивести з контексту компілятора.

2. Змінні.

У Kotlin, як і в Scala, змінні оголошуються за допомогою ключових слів val (незмінні) і var (змінні).

Тип змінної може бути опущений, оскільки компілятор може автоматично його генерувати із значення правої частини визначення змінної. Змінна змінної sinSum вказана явно лише для відображення відповідного синтаксису; і в цьому випадку компілятор має достатньо інформації для виведення типу.

3. Класи.

Основним інструментом декомпозиції в Kotlin, як і в інших об'єктно-орієнтованих мовах, є класи. При оголошенні класу список параметрів конструктора вказується безпосередньо в заголовку:

```
class IntPair (x: Int, y: Int) {...}
```

Екземпляри класу створюються прямим викликом конструктора; ключового слова new в Kotlin немає:

val xy = IntPair (x, y)

У заголовку класу також вказується список типових параметрів (generic) в кутових дужках і список супертіпа, тобто типів, від яких даний клас успадковується, відокремлюваний двокрапкою:

class MyList (length: Int): List, Serializable {...}

4. Трейти.

Клас може успадковуватися від одного класу або від декількох трейтов (trait - дослівно «характерна риса», особливість). Трейти схожі на класи тим, що вони теж визначають типи, теж можуть мати функції-члени і успадковуватися від інших трейтов. Основна відмінність полягає в тому, що трейти не мають конструкторів і, як наслідок, не можуть мати стану (полів). Можна сказати, що трейти - це знайомі всім інтерфейси з мови Java, тільки функції в них можуть мати реалізацію. Дізнатися про обмеження трейти, дозволяють уникнути труднощів, пов'язаних з множинним спадкуванням класів.

5. Зовнішні функції.

Ще один механізм «розширення» типів в Kotlin - це зовнішні функції (extension function - «функція-розширення»). Такі функції можуть бути оголошені поза будь-якого класу і при цьому викликатися так, як ніби вони були оголошені всередині.

Тип, розширюваний цією функцією, вказується перед її ім'ям і відокремлюється крапкою. Це відповідає оголошенню «неявного» параметра, який всередині функції позначається ключовим словом this. Наприклад, в нашому прикладі функція abs () розширює тип Int, і неявний параметр this є цілим числом. Таку

функцію можна викликати за допомогою операції «точка», як і функції - члени класу:

```
val x = (-1) .abs ()
```

Такий синтаксис дозволяє реалізовувати в класах лише необхідний мінімум функціональності без шкоди для читання програми.

Зовнішні функції зв'язуються статично, тобто не є віртуальними (virtual).

6. Керуючі конструкції. When

Kotlin підтримує традиційні для імперативних мов керуючі конструкції if, for і while, на яких ми не будемо зупинятися докладно, а також конструкцію when - операцію розгалуження, яку можна розглядати як розширену версію традиційного оператора switch.

Зліва від знака «=>» вказується вираз або список виразів, розділених комами, званий умовою. Якщо аргумент конструкції when (в нашому прикладі - змінна x) дорівнює хоча б одного з цих виразів, виконується тіло даної умови, тобто вираження або блок, зазначений праворуч від знака «=>». Умови перевіряються послідовно, зверху вниз. Якщо жодна з умов не виконана, виконується код, вказаний після слова else. На відміну від switch, при використанні when умови не є мітками, тому не потрібно закінчувати тіло умови словом break.

Крім простого порівняння на рівність, when дозволяє перевіряти аргумент на приналежність колекції, за допомогою операції in.

Після ключового слова in вказується вираз, що має будь-який тип, що підтримує метод contains (). Зокрема, може бути вказана колекція (як в першому умови в даному прикладі) або проміжок (як у другому). Проміжки можуть бути утворені як цілими, так і дробовими числами, причому крайні значення (в даному прикладі 1 і 10) включаються.

Ще один вид умов відзначається ключовим словом is і дозволяє перевіряти тип аргумента2. Наприклад, перевірити до якого з кількох типів відноситься значення x, можна наступним чином.

Система типів

1. Нульові посилання.

Система типів мови Kotlin дозволяє гарантувати відсутність в програмах деяких видів помилок, наприклад разадресації нульовий посилання. Типи в Kotlin діляться на що містять null і не містять null. Типи, що містять null, анотує знаком питання:

```
fun isEmpty (s: String?): Boolean {...}
```

Знак питання після імені типу (String) означає, що посилання s вказує на об'єкт класу String або має значення null. Результат функції isEmpty, в свою чергу, повинен бути булевим значенням і не може мати значення null, оскільки відповідний тип не проаннотірован знаком питання.

Компілятор не дозволяє разадресовивать посилання типу String? без попередньої перевірки:

```
return s.length () == 0 // Помилка: розіменування нульовий посилання
```

Необхідно явно перевірити, посилається чи s на існуючий об'єкт:

```
if (s! = null) {
  return s.length () == 0 // s точно посилається на існуючий об'єкт
} Else {
  return true
}
```

або

return (s == null) || s.length () == 0 // Оператор || забезпечує перевірку

Часто зустрічаються довгі ланцюжки викликів, кожен з яких може повернути null. В результаті ми отримуємо кілька вкладених умов, які перевіряють, що повернув кожен з викликів в ланцюжку. Щоб уникнути захаращення коду, в Kotlin підтримується оператор безпечного виклику, позначається, «?.»:

a? .getB () ?. getC () ?. getD ()

Якщо a не дорівнює null, вираз a? .GetB () повертає a.getB (), а в іншому випадку - null.

2. Автоматичне приведення типу.

Ми наводили приклад того, як компілятор враховує інформацію, що міститься в умовах, і дозволяє разименовивать вже перевірені посилання. Аналогічний механізм автоматично вставляє операцію приведення типу, якщо раніше в програмі перевірялося відповідне умова. Оператор перевірки типу (аналог instanceof в Java) в Kotlin називається is.

У цьому прикладі посилання x перевіряється на приналежність до типу String, і, якщо перевірка пройшла успішно, на екран виводиться довжина рядка. При виконанні функції length () компілятор автоматично вставляє приведення x до типу String, оскільки воно безпечне в цьому місці програми.

Автоматичне приведення типу працює для всіх умовних конструкцій: if, when, while, ||, && і т. д.

Функції вищих порядків

Користувачі функціональних мов програмування добре знайомі з функціями вищих порядків: механізмом, що дозволяє передавати функції в якості аргументів інших функцій, записувати функції в змінні і т. Д. Як вже говорилося, цей механізм значно полегшує створення бібліотек. В об'єктно-орієнтованих мовах функції вищих порядків зазвичай емулюються за допомогою

паттерна Strategy . Наприклад, для того щоб реалізувати фільтрацію для колекцій довільного типу, необхідно параметризрвані функцію filter () об'єктом, який «уміє» відповідати на питання, чи потрібно включити даний елемент в результуючу колекцію. Цей об'єкт є стратегією фільтрації. У функціональному мовою створювати стратегію не потрібно - можна просто передати функцію.

Будучи об'єктно-орієнтованою мовою, Kotlin проте підтримує функції вищих порядків. Це означає в першу чергу, що функція в Kotlin може бути значенням, які мають відповідний тип:

val predicate: fun (x: Int): Boolean = ...

В даному прикладі змінна predicate має функціональний тип «fun (x: Int): Boolean», тобто збережене нею значення є функцією, що приймає цілочисельний параметр і повертає булевское значення. Зокрема, ми можемо викликати цю функцію:

val yesOrNo = predicate (1)

Така функція може використовуватися, наприклад, при фільтрації набору цілих чисел:

fun Collection.intFilter (predicate: fun (x: Int): Boolean): Collection {...}

А для довільних колекцій можна її узагальнити за допомогою типових параметрів (generic).

(Тут приведена лише наївна реалізація функції filter (). Реалістичніша реалізація вимагає ледачих обчислень, але ця тема виходить за рамки даної статті.)

Найцікавіше - як ставити значення аргументу, що має функціональний тип. Для цього широко використовуються вже згадувані функціональні літерали:

ints.filter ({x => x% 2 == 0})

В даному прикладі аргумент зовнішньої функції filter () являє собою функціональний літерал, тобто коротке оголошення функції. Він обов'язково полягає в фігурні дужки, до символу «=>» слідують оголошення параметрів, а після - тіло функції, причому оператор return не потрібно, оскільки результатом вважається останній вираз в тілі литерала. Таким чином, з колекції ints будуть обрані тільки парні числа, оскільки наш буквальний повертає true, тільки якщо залишок від ділення x на 2 дорівнює нулю.

У наведеному прикладі тип параметра функціонального литерала не вказано, оскільки компілятор автоматично виведе його з контексту. При необхідності типи можна вказати, але в більшості випадків така коротка запис можлива і дозволяє зробити код значно більш читабельним.

Для зручності використання функціональних литералов в Kotlin прийняті наступні синтаксичні конвенції. По-перше, якщо функціональний літерал має рівно один параметр, цей параметр можна не оголошувати, і він автоматично отримує ім'я it (а його тип виводиться з контексту):

ints.filter ({it% 2 == 0}) // Виклик аналогічний попередньому прикладу

По-друге, якщо останнім аргументом при виконанні функції є функціональний літерал, його можна передати поза круглих дужок, а якщо інших аргументів немає, то і самі круглі дужки можна опустити:

ints.filter {it% 2 == 0} // Виклик аналогічний двом попереднім прикладам

Такий синтаксис дозволяє записувати перетворення колекцій в стилі, що нагадує LINQ :

ints.select {it * it}. where {it% 2 == 0} // Серед квадратів елементів колекції // вибрати парні

(Тут функція where () робить те ж, що і функція filter ().)

Крім того, дана конвенція робить виклики функцій більше схожими на звичні керуючі конструкції. Наведемо ще один приклад. Функція synchronized () приймає два параметри: об'єкт синхронізації (монітор) і функцію. Під час виконання спочатку захоплюється монітор, далі в блоці try..finally виконується функція, а потім монітор звільняється.

Для того щоб викликати фрагмент коду з синхронізацією, використовуючи цю функцію, достатньо написати.

Даний приклад показує, як в Kotlin можна засобами мови висловити конструкцію, яка в Java є вбудованої.

Предметно-орієнтовані мови

У вступі згадувалося про бібліотеки, робота з якими нагадує використання предметно-орієнтованих мов, тобто «маленьких» мов, як би вбудованих Kotlin. Прикладом може служити бібліотека для опису модулів, тобто одиниць компіляції, яка використовується в нашій мові. Модуль описується складальним сценарієм - програмою на Kotlin, що викликає функції стандартної бібліотеки; цей сценарій виконується під час компіляції. Розглянемо приклад складального сценарію.

Незважаючи на те що перед нами звичайна програма на Kotlin, вона виглядає як програма на спеціалізованій мові, призначеному для декларативного опису модулів. Такий підхід до опису структур даних вельми популярний в Groovy і інших динамічних мовами, оскільки дозволяє уникнути громіздких і важко читаються дескрипторів, написаних на XML. У Groovy такий підхід відомий під назвою Builders . Яскравим прикладом його використання є бібліотека Gradle .

У порівнянні з Groovy і іншими динамічними мовами, важливою відмінністю внутрішніх DSL в Kotlin є те, що при тій же стислості синтаксису, система типів статично гарантує коректність програми.

Розглянемо принцип реалізації статично типізованих Builders в Kotlin. Для цих цілей досить невеликого фрагмента мови складальних сценаріїв: опису залежностей між модулями в найпростішому випадку.

Ця функція є функцією вищого порядку, оскільки параметр init сам є функцією, причому зовнішньої функцією: про це говорить тип KotlinModule, вказаний перед (порожнім) списком параметрів у функціональному типі. Це означає, що функцію module () можна викликати наступним чином.

Це схоже на приклад сценарію, який ми вже бачили. Зауважимо, що всередині функціонального литерала доступний неявний параметр this типу KotlinModule (оскільки цей літерал має тип «зовнішня функція»), і ми можемо його використовувати.

Залишилося помітити, що this, як зазвичай, можна опустити, і ми отримаємо в точності такий же синтаксис.

Аналогічним чином можна реалізувати багато декларативні мови, включаючи мови розмітки, такі як HTML. При цьому замість тегів будуть використовуватися виклики функцій вищих порядків, і коректність використання таких «тегів», як і їх атрибутів, буде гарантуватися системою типів:

Для написання зовнішнього вигляду при розробці Activities використовується мова розмітки XML – eXtensible Markup Language, розшифровується як розширена мова розмітки. Аналогом являється HTML, яка використовується при розробці веб-сторінок. Якщо коротко то XML це мова для опису і передачі даних в зрозумілому для людини і машини форматі. Використання XML в Android Studio значно полегшує роботу з додатком, а саме з його інтерфейсом, та допомагає запрограмувати всі елементи, які містяться в додатку.

XML теги визначають структуру та значення ваших даних: чим є дані.

Опис структури та значення даних дозволяє вам використовувати їх повторно різними способами. Наприклад, якщо у вас є блок даних про продажі з чітко

визначеними елементами, ви можете завантажити лише необхідні дані до звіту про продажі та завантажити інші елементи до бази даних бухгалтерського обліку. Іншими словами, ви можете створювати дані в одній системі та позначати їх тегами XML, а потім обробляти ці дані в інших системах з різними платформами пристроїв та операційними системами. Завдяки цій універсальності XML є однією з найпопулярніших технологій обміну даними.

XML також не залежить від платформи. Це означає, що будь-яка програма, призначена для використання XML, може читати та обробляти дані XML незалежно від обладнання та операційної системи. Наприклад, за допомогою правильних тегів XML ви можете використовувати класичну програму для відкриття та роботи з основним комп'ютером. Незалежно від того, хто створює текст даних XML, ви можете працювати з одними і тими ж даними в багатьох програмах Office. Завдяки своїй мобільності XML став однією з найпопулярніших технологій аналізу даних між базами даних та робочими столами користувачів.

Документ XML має ієрархічну, логічну структуру і може відображатися у вигляді дерева. Вузлами цього дерева можуть бути:

- Елементи, фізична структура яких складається з:

  - правильна пара відкриваючих та закриваючих міток;
  - порожні теги елементів.

- атрибути, які приймають форму пар ключ-значення і знаходяться або у початковому тегу, або в порожньому тегу;

- Інструкція з редагування документа;

- Коментарі;

- Текст.

Документ XML може мати лише один кореневий елемент. Інші елементи є субелементами цього кореневого елемента.В інтернеті існує багато середовищ розробки Android-додатків, і в кожного є свої прихильники, найбільш популярні: IntelliJ IDEA, Android Studio, NetBeans IDE, Visual Studio Code, Eclipse, Xamarin.Android.

Для написання дипломного проєкту було обрано IDE Android Studio, воно зручніше у використанні, рекомендоване Google для розробки Android-додатків та має влаштовий в IDE AVD(Android Virtual Device).

AVD- це емулятор Android який імітує Android пристрої на комп'ютері, щоб можна було протестувати розроблену програму на різних пристроях та різних ОС Android. Емулятор забезпечує майже всі можливості справжнього пристрою Android

Приклад встановлення IDE Android Studio:

1. Перейти в будь-якому веб-браузері за посиланням https://developer.android.com/studio. На екрані буде видно веб-сторінку зображену на рисунку 2.

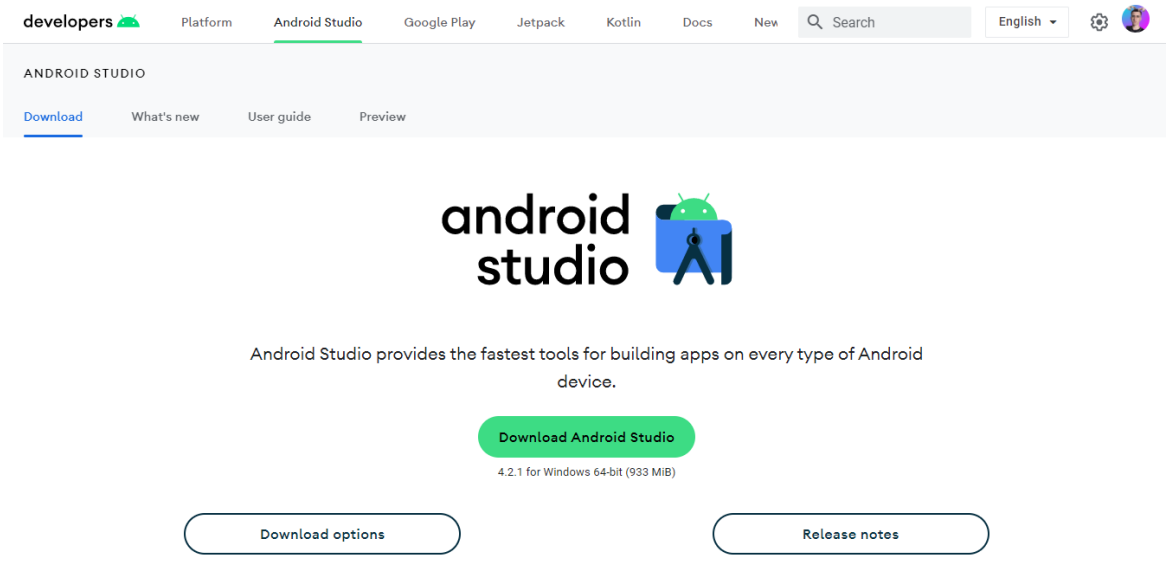


Рис 2.

2. Натиснути кнопку «Download Android Studio» та завантажити файл на компьютер.
3. Запустити завантажений файл «android-studio-ide-.exe».
4. Вибрати місце інсталяції та інші опції.
5. Підтвердити встановлення.
6. Після встановлення запуститься вікно IDE зображене на рисунку 3.

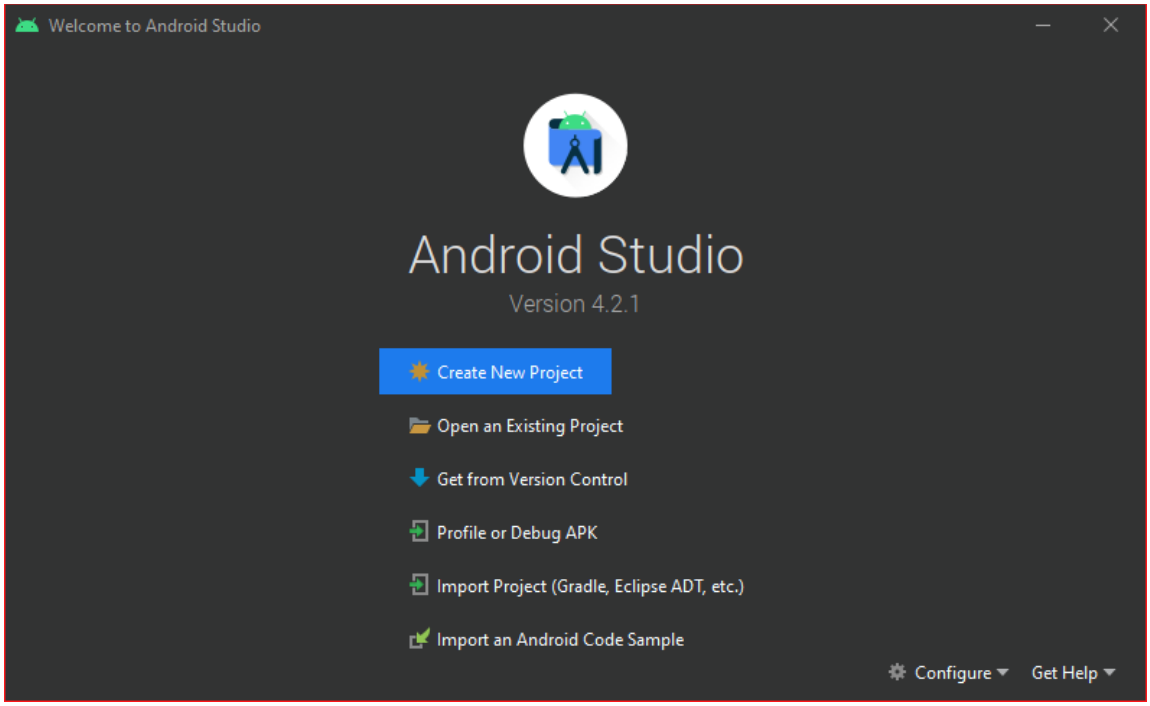
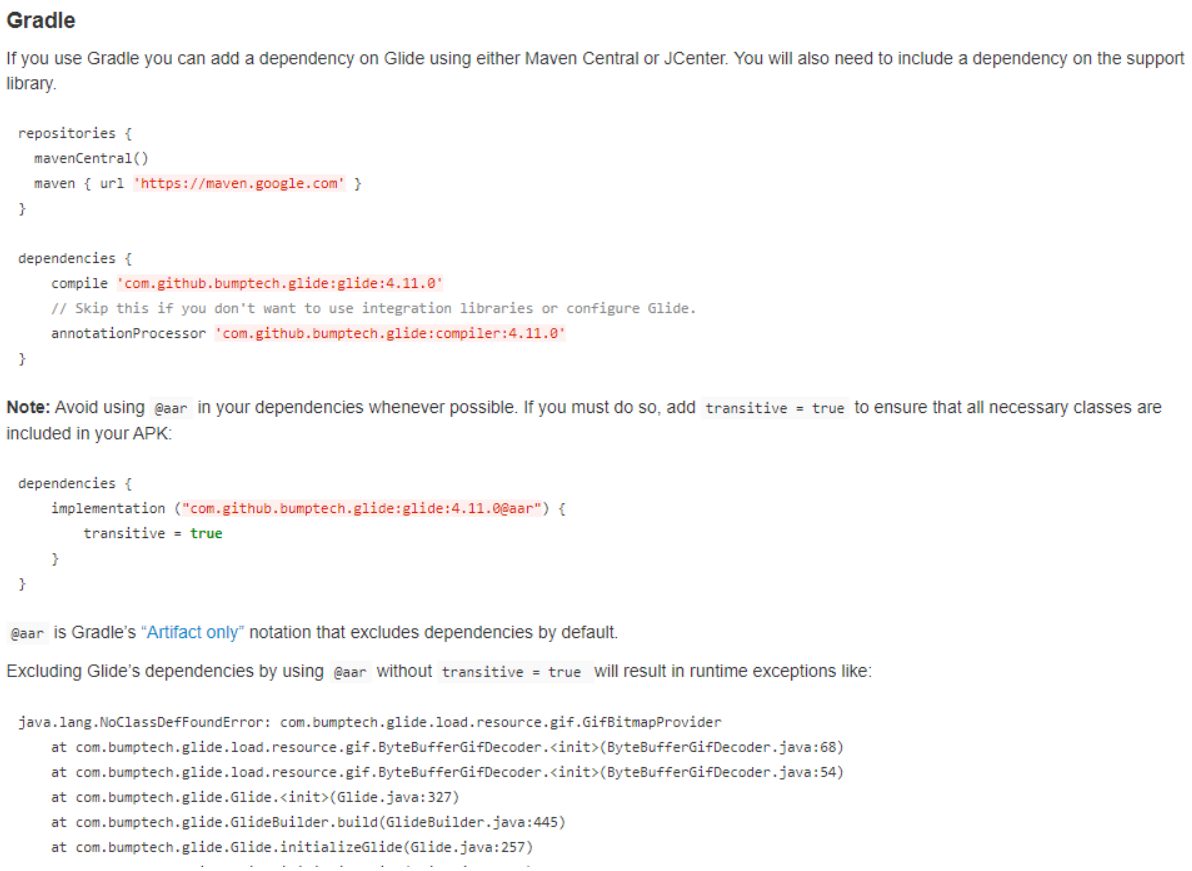


Рис. 3

Для завантаження зображень через інтернет є багато способів. найпопулярніші це стороні бібліотеки відтритого коду: Glide, Picasso та інші… Завантаження зображень в додатку виконується за допомогою Glide. Він має зрозумілий, зручний для використання API, легко імпортується в проєкт, підтримує велику кількість версій ОС Android(Ice Cream, API level 14 or higher) та рекомендований Google для використання.

Приклад імпорту модуля Glide в проект:

1. Перейти в будь-якому веб-браузері за посиланням https://bumptech.github.io/glide/doc/download-setup.html#gradle. На екрані буде видно веб-сторінку зображену на рисунку 4.



Gradle

If you use Gradle you can add a dependency on Glide using either Maven Central or JCenter. You will also need to include a dependency on the support library.

```
repositories {
  mavenCentral()
  maven { url 'https://maven.google.com' }
}

dependencies {
    compile 'com.github.bumptech.glide:glide:4.11.0'
    // Skip this if you don't want to use integration libraries or configure Glide.
    annotationProcessor 'com.github.bumptech.glide:compiler:4.11.0'
}
```

Note: Avoid using `@aar` in your dependencies whenever possible. If you must do so, add `transitive = true` to ensure that all necessary classes are included in your APK:

```
dependencies {
    implementation ("com.github.bumptech.glide:glide:4.11.0@aar") {
        transitive = true
    }
}
```

`@aar` is Gradle's "Artifact only" notation that excludes dependencies by default.

Excluding Glide's dependencies by using `@aar` without `transitive = true` will result in runtime exceptions like:

```
java.lang.NoClassDefFoundError: com.bumptech.glide.load.resource.gif.GifBitmapProvider
    at com.bumptech.glide.load.resource.gif.ByteBufferGifDecoder.<init>(ByteBufferGifDecoder.java:68)
    at com.bumptech.glide.load.resource.gif.ByteBufferGifDecoder.<init>(ByteBufferGifDecoder.java:54)
    at com.bumptech.glide.Glide.<init>(Glide.java:327)
    at com.bumptech.glide.GlideBuilder.build(GlideBuilder.java:445)
    at com.bumptech.glide.Glide.initializeGlide(Glide.java:257)
```

Рис. 4

2. Слідуючи інструкціям додаєм в файл build.gradle(Project Name) рядки:

```
repositories {
 mavenCentral()
 maven { url 'https://maven.google.com' }
}
```

3. Також додаєм в файл build.gradle(app) рядки:

```
dependencies {
   implementation 'com.github.bumptech.glide:glide:4.11.0'
   kap 'com.github.bumptech.glide:compiler:4.11.0'
}
```

4. Синхронізуємо проект.

Завантаження аудіо в додатку виконується за допомогою Media Player, який входить в пакет платформи Android, і не потребує імпорту.

Для зберігання контенту додатка та користувачів, в додатку реалізовано використання бази даних. В інтернеті доступна велика кількість БД: Amazon Web Services, Microsoft Azure, Firebase та інші… Більшість з наведених сервісів являються платними, а безкоштовний план який надається замалий, або недоступна одна з двох потрібних умов. Для додатка потрібно по-перше, база даних для профілів користувачів та медіа контенту а по-друге, місце зберігання файлів. З цих всіх було обрано Firebase від Google, він включає в себе Realtime Database та Cloud Firestore.

Realtime Database- База даних в режимі реальному часі- це база даних, яка використовує обробку в реальному часі для обробки навантажень, стан яких постійно змінюється. Цим вона відрізняється від інших БД. Firebase Realtime Database являється NoSQL базою даних, дані зберігаються як JSON і синхронізуються в режимі реального часу.

NoSQL – це база даних яка забезпечує механізм зберігання та видобування даних відміний від підходу таблиць-відношень в реляційних базах-даних. NoSql розшифровується як Not only SQL – для підкреслення того, що вони можуть підтримувати SQL-подібну структуру та мову запитів. Даний механізм забезпечує: простоту дизайну схеми БД, значно спрощене горизонтальне масштабування та кластери машин, і тонкий контроль над доступністю.

Firebase пропонує Cloud Firestore або Realtime Database. Firestore найновіша база даних Firebase для розробки мобільних додатків. Він базується на успіхах бази даних у реальному часі за допомогою нової, більш інтуїтивної моделі даних. Realtime Database це оригінальна база даних Firebase. Це ефективне рішення з низькою затримкою для мобільних додатків, які потребують синхронізації станів між клієнтами в режимі реального часу.

Для імпорту БД Realtime Database в проєкт потрібно перейти за посиланням https://firebase.google.com/docs/database/android/start?authuser=0 зображено на рисунку 5 і скопіювати в build.gradle «implementation 'com.google.firebase:firebase-database-ktx:20.0.0'».

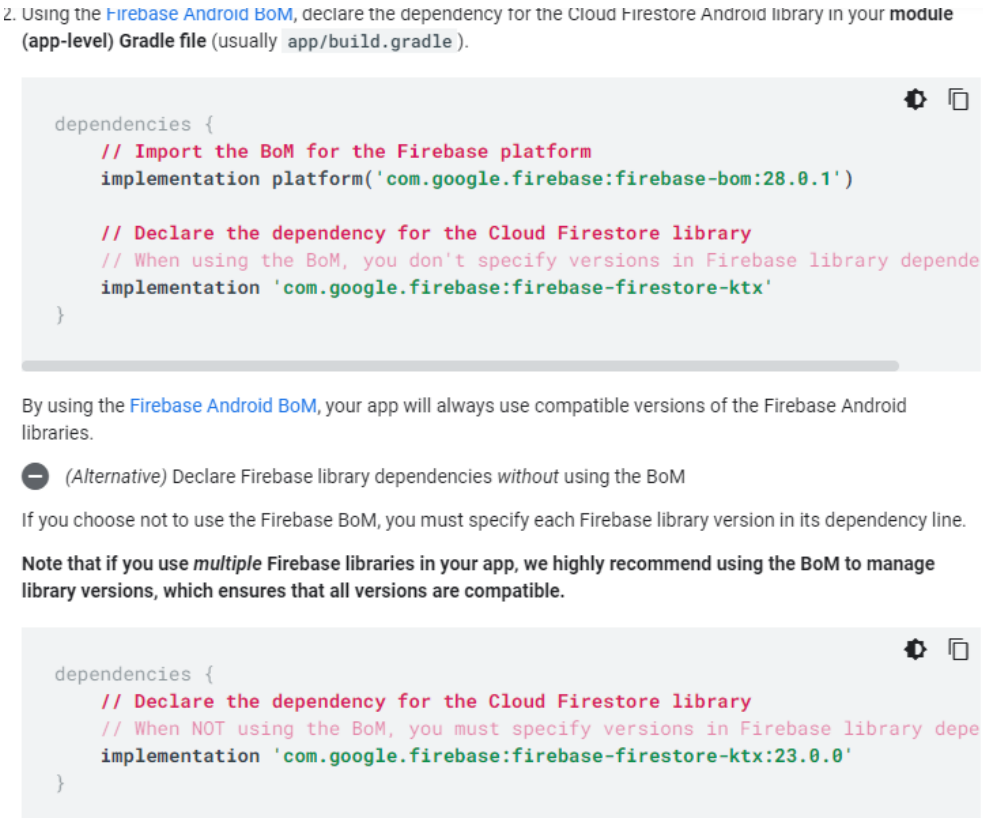


Рис. 5

Для аутентифікації користувачів в проєкт потрібно додати модуль Firebase Authentication, потрібно перейти за посиланням https://firebase.google.com/docs/auth/android/start?authuser=0 зображено на рисунку 6 і скопіювати в build.gradle «implementation 'com.google.firebase:firebase-auth-ktx:21.0.1'».

Рис. 6

Для імпорту модуля Cloud Firestore в проєкт потрібно перейти за посиланнямhttps://firebase.google.com/docs/firestore/quickstart?authuser=0#kotlin+ktx зображено на рисунку 7, скопіювати в проєкт implementation 'com.google.firebase:firebase-firestore-ktx:23.0.0' та сихронізувати його.

Рис. 7

Тестування проекту має відбуватися на декількох віртуальних та декількох фізичних пристроях. Адже велика кількість смартфонів, планшетів та інших гаджетів використовують різні розміри і екрану, та різні настроювання над ОС Android, які потрібно брати до уваги.

2.2 Системні вимоги розробки Android-додатків

Перед початком розробки додатку для Android потрібно інсталювати середу розробки та всі технічні засоби. В моєму випадку була інстальована Android Studio, JDK, використовувалась операційна система Windows 10, та екран 22’ з розширенням 1920x1080px.

На офіційній сторінці в документації Android Studio вказані такі системні вимоги:

- Microsoft Windows 10/8 (32 або 64-біт)
- Рекомендовано 8 GB RAM;
- 30 GB простору для Android Studio;
- Java Development Kit (JDK) 8;
- Мінімальна роздільність дисплею 1280x800.

2.3 Висновки

В даному розділі дипломного проєкту було розглянути засоби для розробки та реалізації додатку. Серед мови програмування було обрано Kotlin через масовий перехід розробників та зручність написання. Серед мови розмітки обрано XML. Серед великої кількості IDE було обрано Android Studio також через її зручність у використанні та встроєному AVD. Також було розказано які стороні модулі використовуються в проєкті і як їх імпортувати. Та приведені рекомендовані та системні вимоги для розробки.

3 РОЗРОБКА ANDROID-ДОДАТКА «СЛУХАЙ»

3.1 Основні визначення

Розроблюваний додаток «Слухай» складається з 4 так званих «пакетів»: models, views, activities, utils.Зображено на рисунку 8.

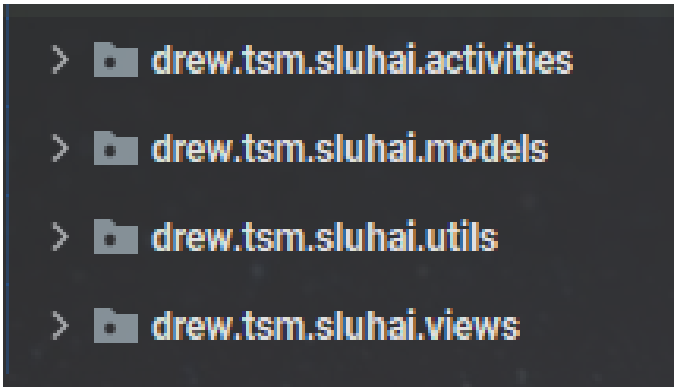


Рис. 8

Пакет- це простір імен, що організовує набір пов’язаних класів та інтерфейсів. Концептуально можна вважати пакети подібні до різних папок.

Життєвий цикл Activity контролюється системою і залежить від потребностей користувача і можливостей системи. Основні методи життєвоного циклу зображено на рисунку 9.

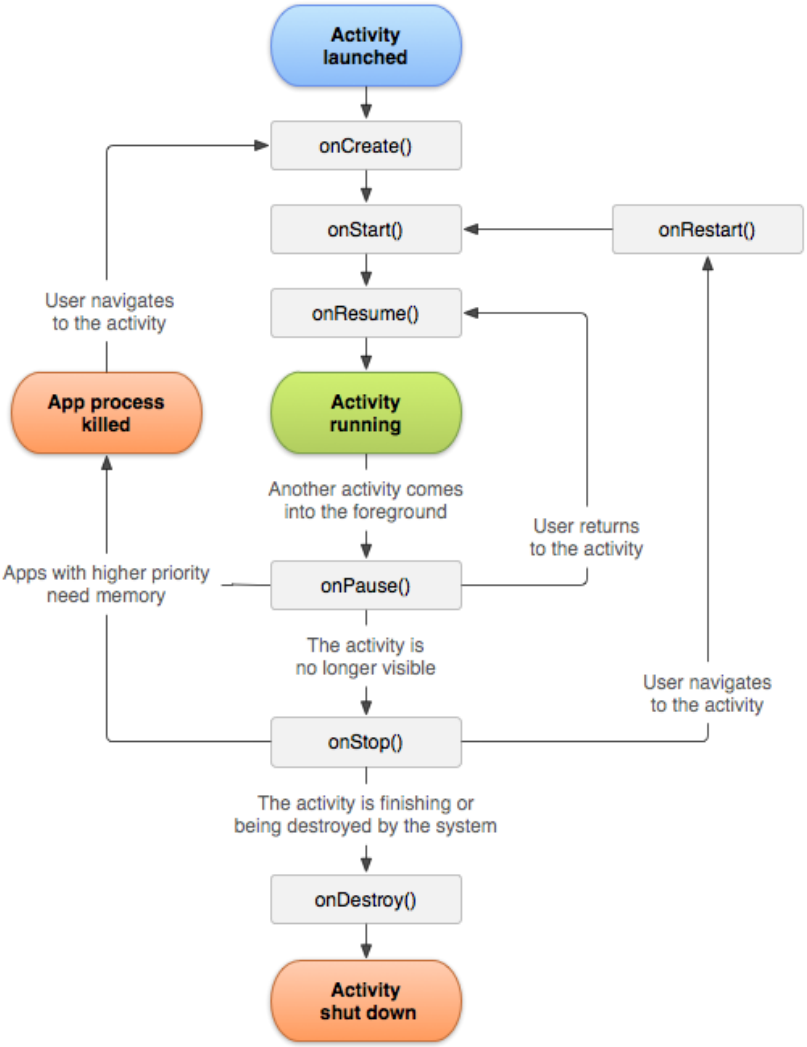


Рис. 9

Концепції життєвого циклу Activity

Клас Activity забезпечує базовий набір із шести зворотних викликів для навігації між фазами життєвого циклу діяльності: onCreate (), onStart (), onResume (), onPause (), onStop () і onDestroy (). Система викликає кожен із цих зворотних викликів, коли діяльність переходить у новий стан.

Коли користувач починає залишати діяльність, система викликає методи звільнення активності. У деяких випадках цей демонтаж є лише частковим; дія все ще залишається в пам'яті (наприклад, коли користувач переходить на іншу програму) і все ще може повернутися на передній план. Коли користувач повернеться до цієї дії, дія продовжиться там, де він зупинився. За деякими

винятками, програмам заборонено виконувати будь-які дії у фоновому режимі.

Ймовірність того, що система скасує цей процес та дії, які він містить, залежить від поточного стану активності. Стан активності та витіснення з пам'яті надають більше інформації про взаємозв'язок між станом та сприйнятливістю до витіснення.

Залежно від складності вашої діяльності, можливо, вам не доведеться застосовувати всі методи життєвого циклу. Однак важливо розуміти кожного з них і впроваджувати ті, які забезпечать, щоб ваш додаток поводився так, як очікують користувачі.

Зворотні виклики життєвого циклу

Цей розділ містить концептуальну інформацію та інформацію про реалізацію методів зворотного виклику, що використовуються в життєвому циклі діяльності.

Деякі дії, такі як виклик setContentView (), самі по собі є методами життєвого циклу діяльності. Однак код, який реалізує дії залежного компонента, повинен знаходитись у самому компоненті. Для цього вам потрібно усвідомити залежний компонент про життєвий цикл. Прочитайте розділ "Робота з життєвими циклами за допомогою компонентів, що усвідомлюють життєвий цикл", щоб дізнатися, як усвідомити життєвий цикл залежних компонентів.

onCreate ()

Вам потрібно реалізувати цей зворотний виклик, який запускається системою, коли дія створюється вперше. Коли створюється дія, створюється дія. У методі onCreate () ви виконуєте основну логіку запуску програми, яка повинна виконуватися лише один раз за час дії. Наприклад, ваша реалізація onCreate () може прив'язувати дані до списків, прив'язувати дію до ViewModel та створювати екземпляри деяких змінних діапазону класів. Цей метод отримує параметр saveInstanceState, який являє собою набір, що містить попередньо

збережений статус діяльності. Якщо діяльність ніколи не існувала, пакет встановлюється на нуль.

Наступний приклад методу onCreate () показує базову настройку дії, наприклад Наприклад, оголошення користувальницького інтерфейсу (визначеного у файлі XML-макета), визначення змінних-членів та налаштування деяких елементів інтерфейсу користувача. Цей приклад встановлює XML-файл макета, передаючи ідентифікатор ресурсу файлу R.layout.main_activity в setContentView ().

onStart()

Коли діяльність переходить у статус Розпочато, система викликає цей зворотний дзвінок. Виклик onStart () робить діяльність видимою для користувача, поки програма готується до того, що діяльність вийде на перший план і стане інтерактивною. Наприклад, за допомогою цього методу програма ініціаліує код, який керує користувальницьким інтерфейсом.

Коли дія переходить у стан "Розпочате", кожен компонент, що усвідомлює життєвий цикл, пов'язаний із життєвим циклом діяльності, отримує подію ON_START.

Метод onStart () завершується дуже швидко, і, як і у випадку зі створеним статусом, дія не залишається постійною у статусі Розпочато. Як тільки цей зворотний виклик буде завершено, дія переходить у стан відновлення, і система викликає метод onResume ().

onResume ()

Коли дія переходить у відновлений стан, вона виходить на перший план, а потім система викликає зворотний виклик onResume (). Це стан, в якому додаток взаємодіє з користувачем. Додаток залишатиметься в такому стані, поки не станеться щось, що відверне увагу від програми. Така подія може включати отримання телефонного дзвінка, перехід користувача до іншої діяльності або

вимкнення екрана пристрою.

Коли дія переходить у відновлений стан, кожен компонент, який усвідомлює життєвий цикл, пов'язаний із життєвим циклом діяльності, отримує подію ON_RESUME. Тут компоненти життєвого циклу можуть активувати всі функції, які повинні виконуватися, поки компонент видно та на передньому плані, наприклад Б. Початок попереднього перегляду камери.

Коли відбувається подія переривання, дія змінюється на призупинений стан, і система викликає зворотний виклик onPause ().

Коли дія повертається із призупиненого стану у відновлений стан, система знову викликає метод onResume (). З цієї причини вам слід реалізувати onResume (), щоб ініціалізувати компоненти, які ви випускаєте під час onPause (), і виконати будь-які інші ініціалізації, які потрібно виконати кожного разу, коли діяльність переходить у відновлений стан.

Ось приклад компонента, що знає життєвий цикл, який отримує доступ до камери, коли компонент отримує подію ON_RESUME.

onPause ()

Система називає цей метод першим свідченням того, що користувач залишає вашу активність (хоча це не завжди означає, що діяльність руйнується); це вказує на те, що діяльність більше не на передньому плані (хоча вона все ще може бути видимою, якщо користувач перебуває в режимі декількох вікон). Використовуйте метод onPause (), щоб призупинити або відрегулювати операції, які ви не хочете відновлювати (або повинні поновлюватися помірно), поки діяльність перебуває в призупиненому стані і, як очікується, відновиться незабаром. Є кілька причин, через які діяльність може перейти в цей стан.

В ОС Android 7.0 (рівень API 24) або вище кілька програм працюють у режимі багато вікон. Оскільки фокус має лише одна з програм (вікно), система призупиняє всі інші програми.

Відкривається нова напівпрозора діяльність (наприклад, діалогове вікно). Поки діяльність все ще частково видно, але не у фокусі, вона залишається призупиненою.

Коли дія переходить у призупинений стан, будь-який компонент, який усвідомлює життєвий цикл, пов'язаний із життєвим циклом діяльності, отримує подію ON_PAUSE. Тут компоненти життєвого циклу можуть зупинити будь-які функції, які не потрібно виконувати, поки компонент не на передньому плані, наприклад Б. зупинка попереднього перегляду камери.

Ви також можете використовувати метод onPause (), щоб звільнити системні ресурси, ручки для датчиків (наприклад, GPS) або інші ресурси, які можуть вплинути на час автономної роботи, коли ваша діяльність призупинена, і користувач у цьому не потребує. Однак, як згадувалося в розділі onResume () вище, призупинені дії все ще можуть бути повністю видимими в режимі багато вікон. Тому вам слід розглянути можливість використання onStop () замість onPause (), щоб повністю звільнити або налаштувати ресурси та операції, пов'язані з користувацьким інтерфейсом, для кращої підтримки багатовіконного режиму.

Наступний приклад LifecycleObserver, який реагує на подію ON_PAUSE, протилежний наведеному вище прикладу для події ON_RESUME і запускає камеру, ініціалізовану після отримання події ON_RESUME:

onStop ()

Якщо ваша активність більше не видна користувачеві, вона досягла статусу Зупинено, і система викликає зворотний виклик onStop (). Це може статися, наприклад, коли перезапущена діяльність охоплює весь екран. Система також може викликати onStop (), коли діяльність закінчилася і повинна бути закінчена.

Коли активність переходить у зупинений стан, будь-який компонент, який знає життєвий цикл, прив'язаний до життєвого циклу діяльності, отримує подію

ON_STOP. Тут компоненти життєвого циклу можуть зупинити всі функції, які не потрібно виконувати, поки компонент не видно на екрані.

За допомогою методу onStop () програма повинна звільнити або відкоригувати непотрібні ресурси, тоді як програма не відображається користувачеві. Наприклад, ваш додаток може призупиняти анімацію або переходити від дрібнозернистого до грубозернистого оновлення місцезнаходжень. Використання onStop () замість onPause () гарантує, що робота, пов'язана з користувальницьким інтерфейсом, продовжується, навіть якщо користувач переглядає вашу активність у режимі декількох вікон.

onDestroy ()

onDestroy () викликається до того, як діяльність буде знищена. Система викликає цей зворотний виклик або з таких причин:

дія закінчується (тому що користувач повністю відхиляє активність або тому, що для дії викликається finish ()), або

система тимчасово знищує активність через зміну конфігурації (наприклад, обертання пристрою або багатовіконний режим)

Коли діяльність переходить у зруйнований стан, кожен компонент, що усвідомлює життєвий цикл, пов'язаний із життєвим циклом діяльності, отримує подію ON_DESTROY. Тут компоненти життєвого циклу можуть очистити все, що потрібно зробити, перш ніж діяльність буде знищена.

Замість того, щоб додавати логіку до своєї діяльності, щоб визначити, чому вона руйнується, розгляньте можливість використання об'єкта ViewModel для зберігання відповідних даних перегляду для вашої діяльності. Якщо діяльність відтворюється заново через зміну конфігурації, ViewModel не повинен нічого робити, оскільки вона зберігається та передається наступному екземпляру діяльності. Якщо дія не створюється заново, на ViewModel викликається метод onCleared (), який можна використовувати для очищення будь-яких необхідних

даних до їх знищення.

Ви можете використовувати метод isFinishing (), щоб розрізнити ці два сценарії.

Коли дія закінчується, onDestroy () є останнім зворотним викликом життєвого циклу, який отримує дія. Коли onDestroy () викликається в результаті зміни конфігурації, система негайно створює новий екземпляр діяльності, а потім викликає onCreate () для цього нового екземпляра в новій конфігурації.

Зворотний виклик onDestroy () повинен звільнити всі ресурси, які ще не були звільнені попередніми зворотними викликами, такими як onStop ().

3.2 Модуль activities

Модуль activities містить лише Activity додатку, екрани які відображаються на пристрої для користувача. В програмі реалізовані наступні 11 Activity:

MainActivity- екран який відображається після SplashScreean та являється головним екраном додатку, який містить список творів, поле для пошуку, кнопку для сортування списку, кнопку для вибіру категорій, та кнопку для переходу на ProfileActivity та при натисканні на елемент списку відкриває PlayerActivity з ним. Зображено на рисунку 10.

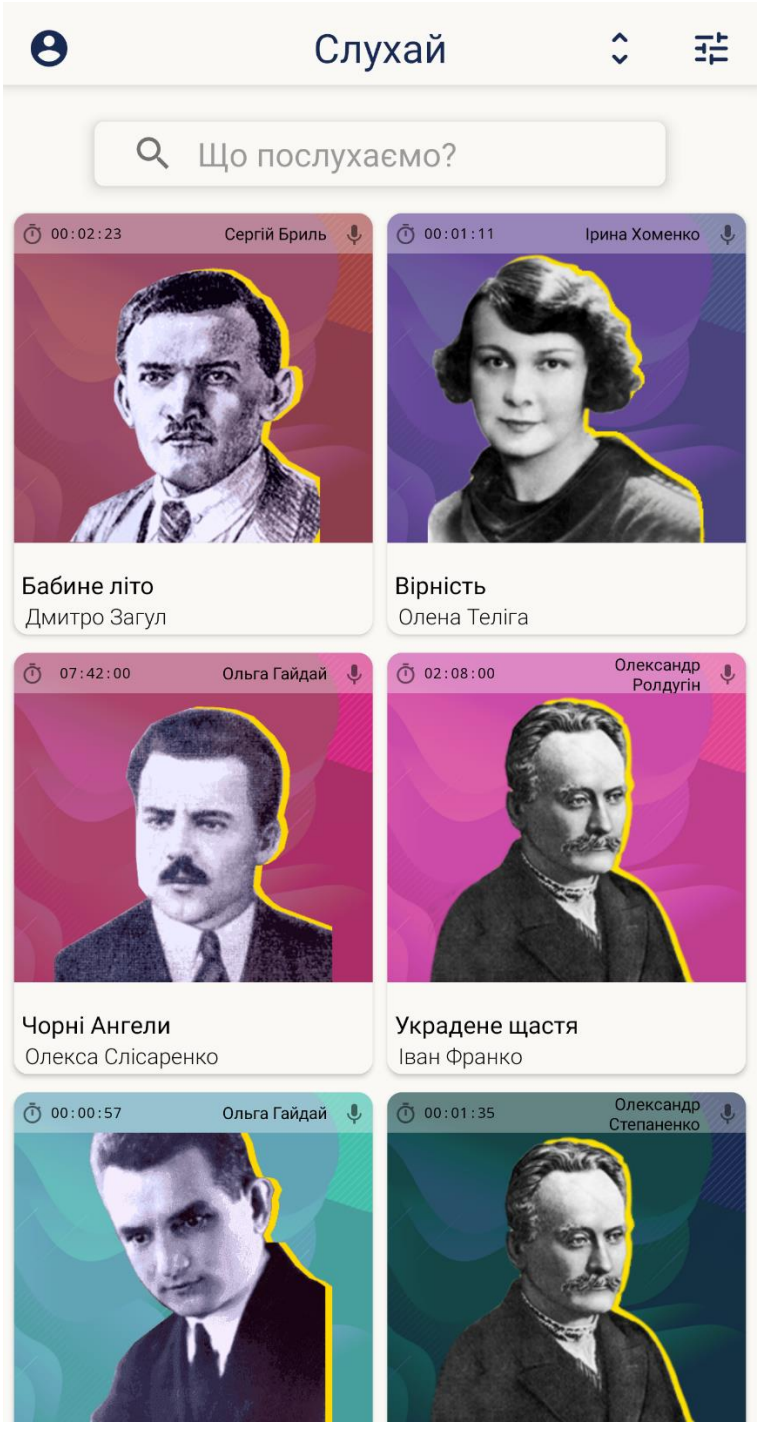

Слухай
Що послухаємо?
00:02:23
Сергій Бриль
Бабине літо
Дмитро Загул
00:01:11
Ірина Хоменко
Вірність
Олена Теліга
07:42:00
Ольга Гайдай
Чорні Ангели
Олекса Слісаренко
02:08:00
Олександр Ролдугін
Украдене щастя
Іван Франко
00:00:57
Ольга Гайдай
00:01:35
Олександр Степаненко

Рис. 10

На рисунку 11 зображено можливість пошуку за назвою твору. При натискані на поле вводу відкривається клавіатура.

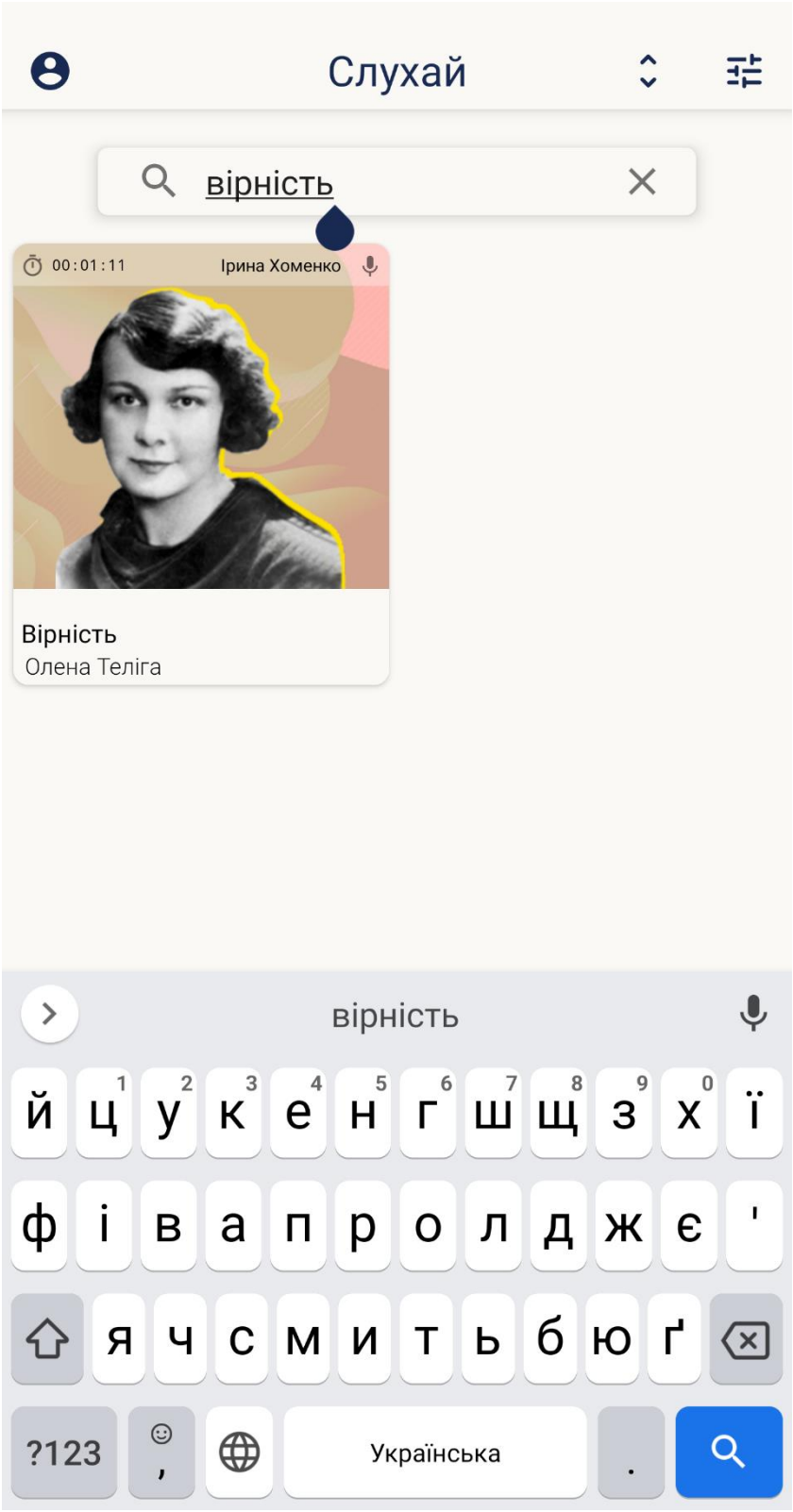


Рис. 11

В додатку реалізовано можливість сортувати список з творами за 3 полями з можливістю кожного від найменшого до більшого і навпаки(Зображено на рисунку 12), наприклад: рейтинг, назва твору, та новизною додавання в додаток. Приклад зображено на рисунку 13.

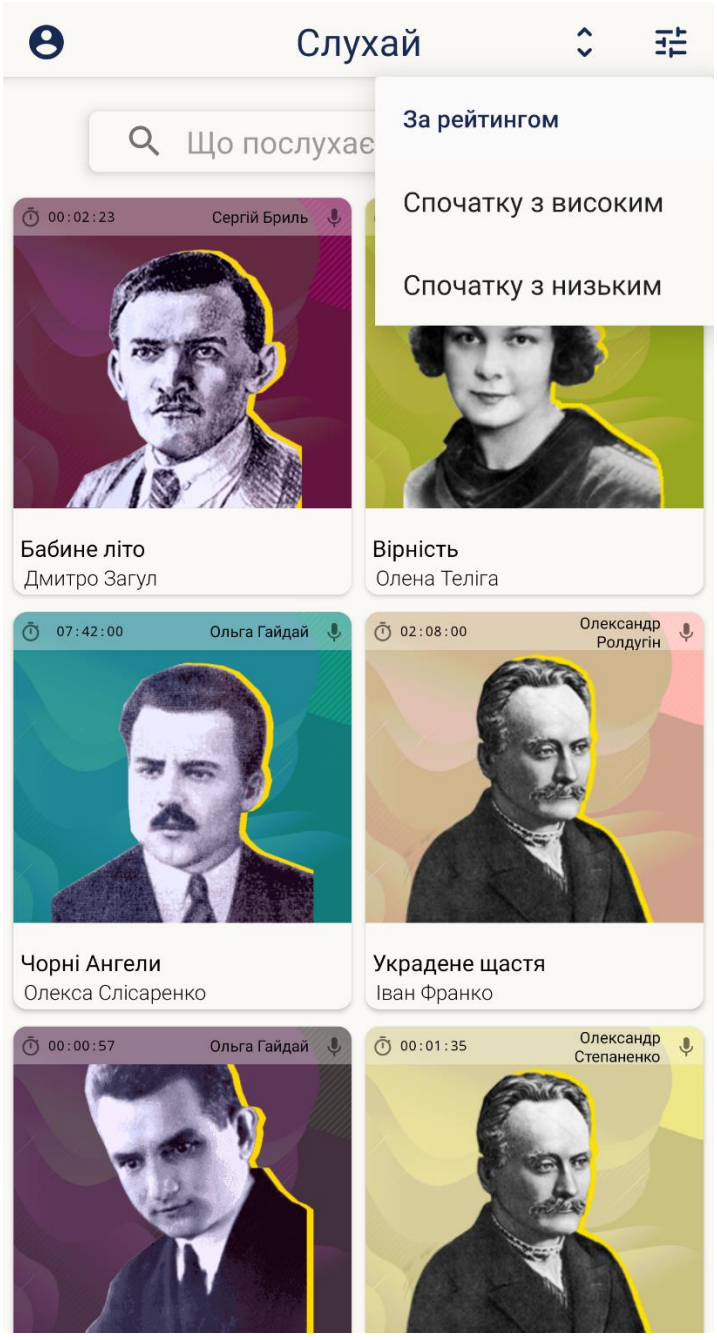


Рис. 12

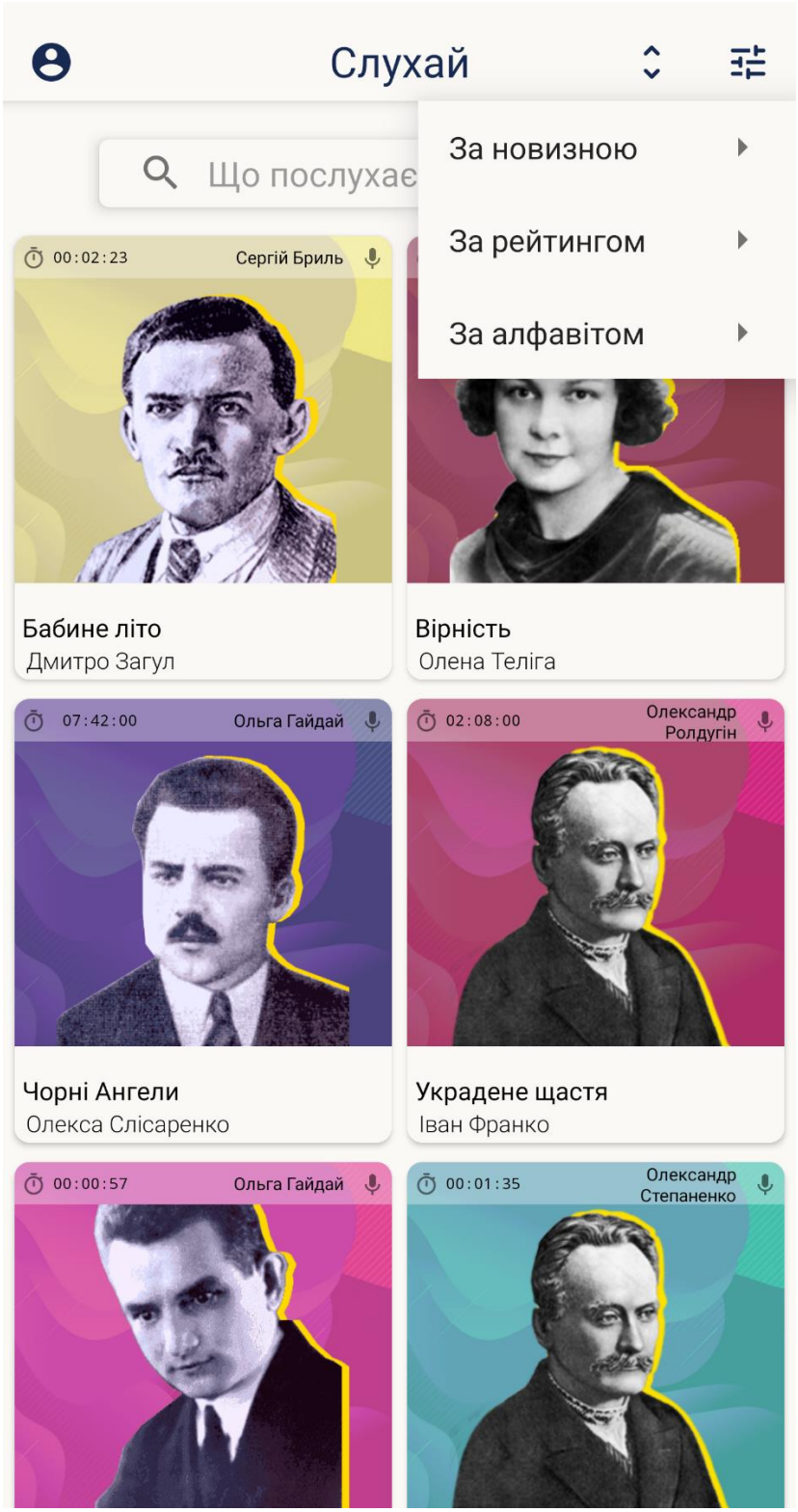

Слухай
За новизною
За рейтингом
За алфавітом
Що послухає
00:02:23
Сергій Бриль
Бабине літо
Дмитро Загул
Вірність
Олена Теліга
07:42:00
Ольга Гайдай
Чорні Ангели
Олекса Слісаренко
02:08:00
Олександр Ролдугін
Украдене щастя
Іван Франко
00:00:57
Ольга Гайдай
00:01:35
Олександр Степаненко

Рис. 13

Також реалізована можливість вибору зі списку творів за категоріями, такі як: Автор, Жанр, Виконавець, та скидання вибраного.(Зображено на рисунках 14,15).

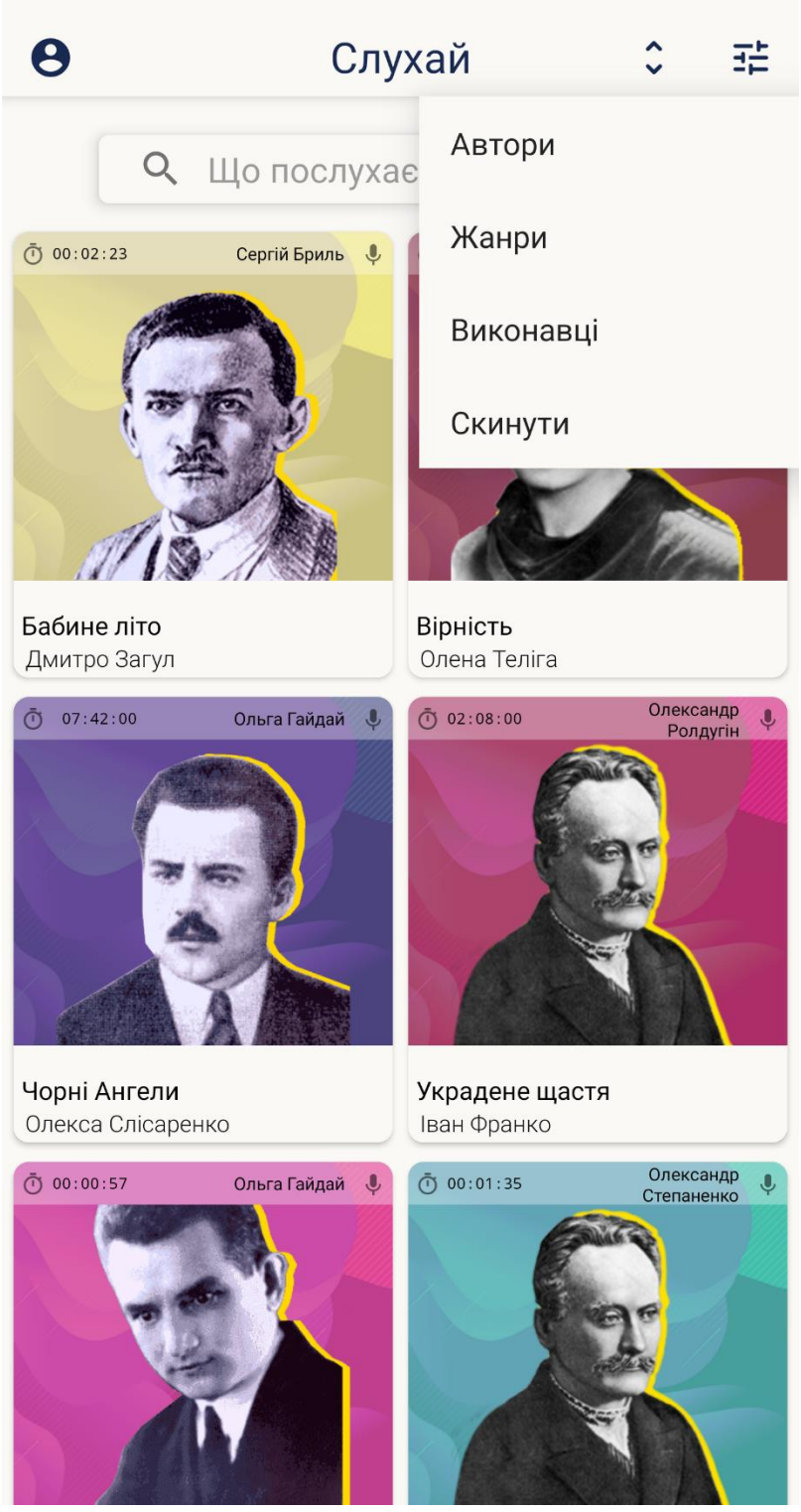


Рис. 14

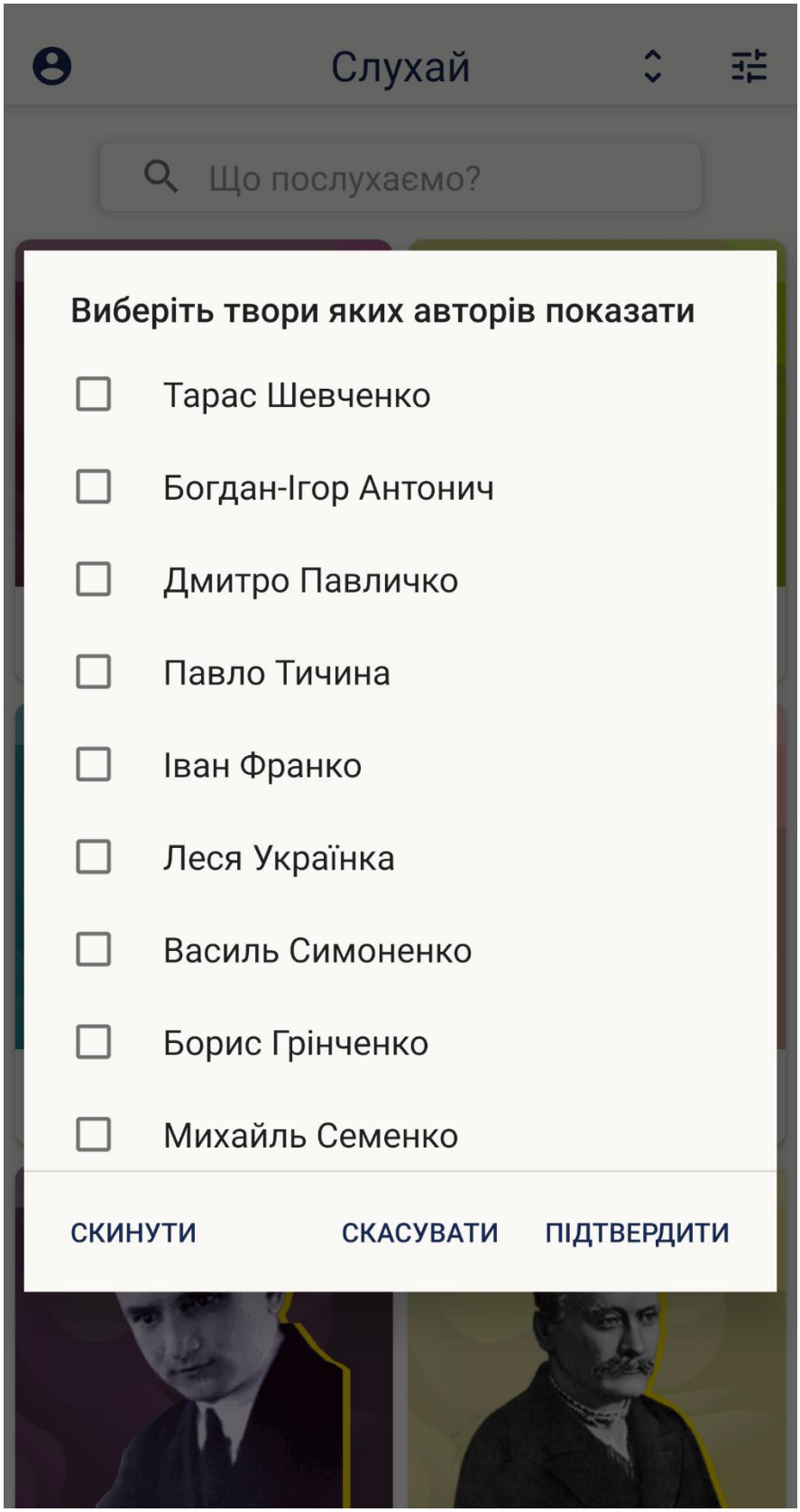

Слухай
Що послухаємо?
Виберіть твори яких авторів показати
Тарас Шевченко
Богдан-Ігор Антонич
Дмитро Павличко
Павло Тичина
Іван Франко
Леся Українка
Василь Симоненко
Борис Грінченко
Михайль Семенко
СКИНУТИ
СКАСУВАТИ
ПІДТВЕРДИТИ

Рис. 15

ProfileActivity- екран користувача з якого можна перейти на EditProfileActivity, HelpActivity, LoginActivity, SavedActivity при натискані на відповідну кнопку та повернення назад на головний екран при натисканні стрілки. Зображено на рисунку 16.

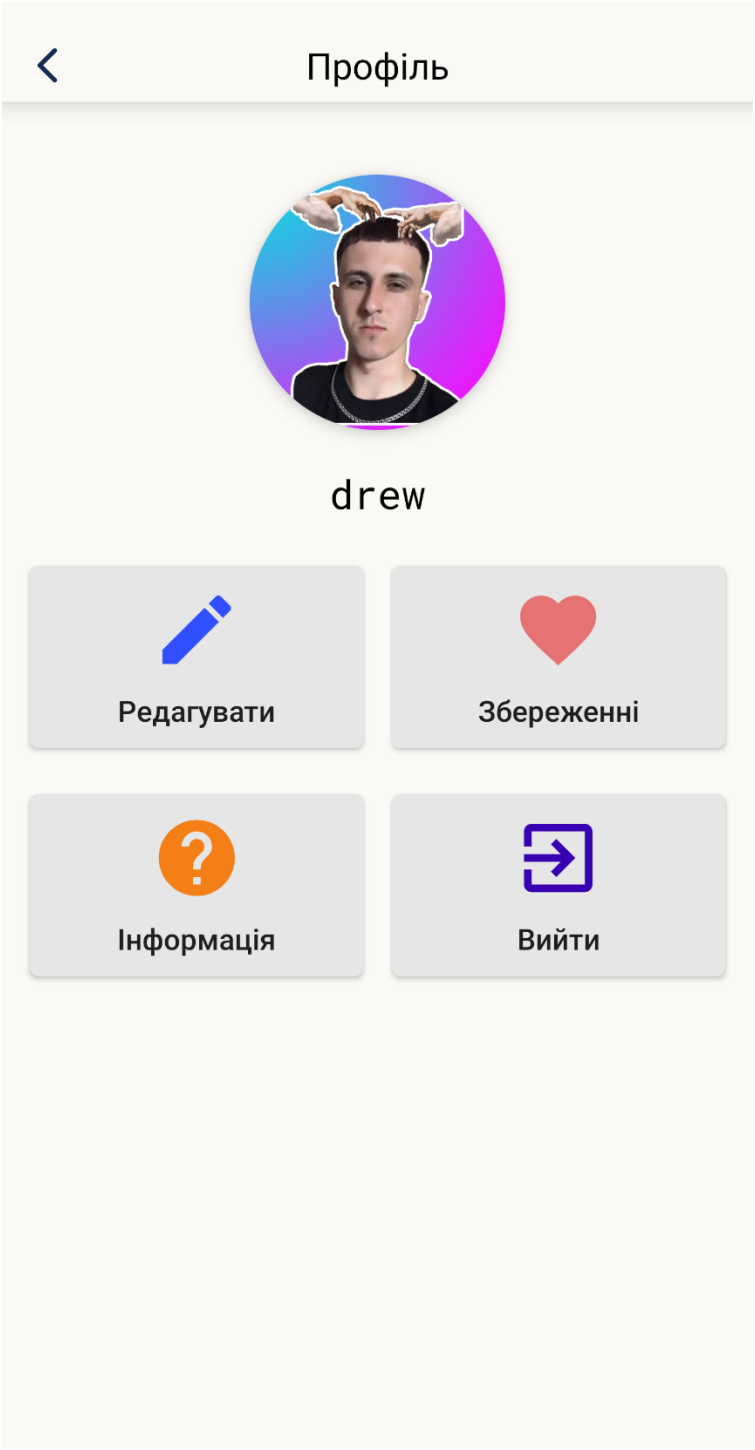


Рис. 16

RegisterActivity- екран для реєстрації нового користувача, він розроблений як оболонка для 2 фрагментів: Email та PasswordName Зображено на рисунках 17, 18.

Рис. 17

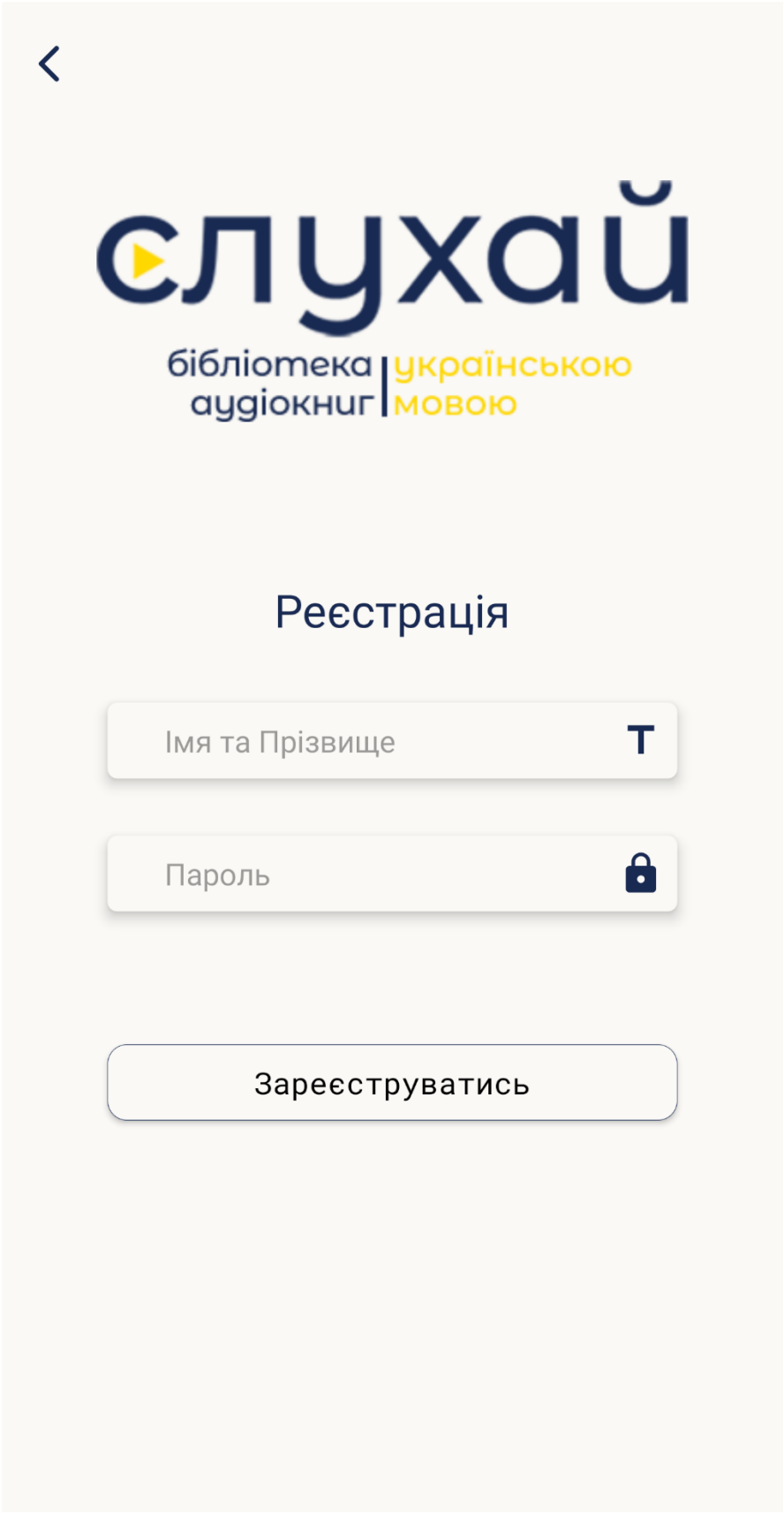

слухай
бібліотека аудіокниг
українською мовою
Реєстрація
Ім'я та Прізвище
Пароль
Зареєструватись

Рис. 18

LoginActivity- екран для входу в профіль користувача. Для входу потрібнго вести email та пароль, з цього Activity можна перейти на екран реєстрації нового користувача, екран відновлення забутого пароля або повернутись на головний екран. Зображено на рисунку 19.

Рис. 19

EditProfileActivity- екран для редагування даних профілю користувача та завантаження фото користувача. Для зміни email появляється вікно для вводу пароля щоб підтвердити особу. Зображено на рисунках 20, 21.

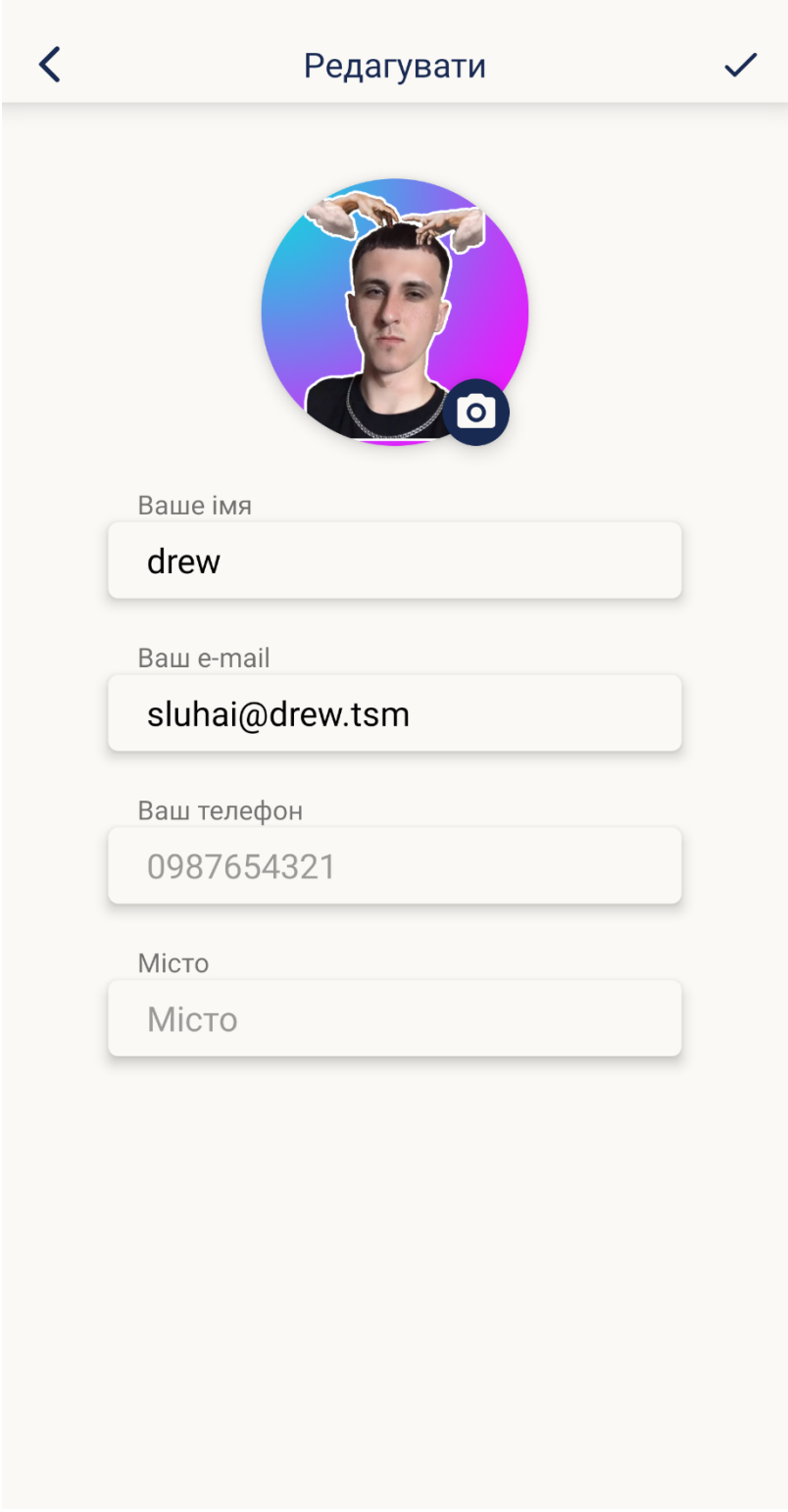


Рис. 20

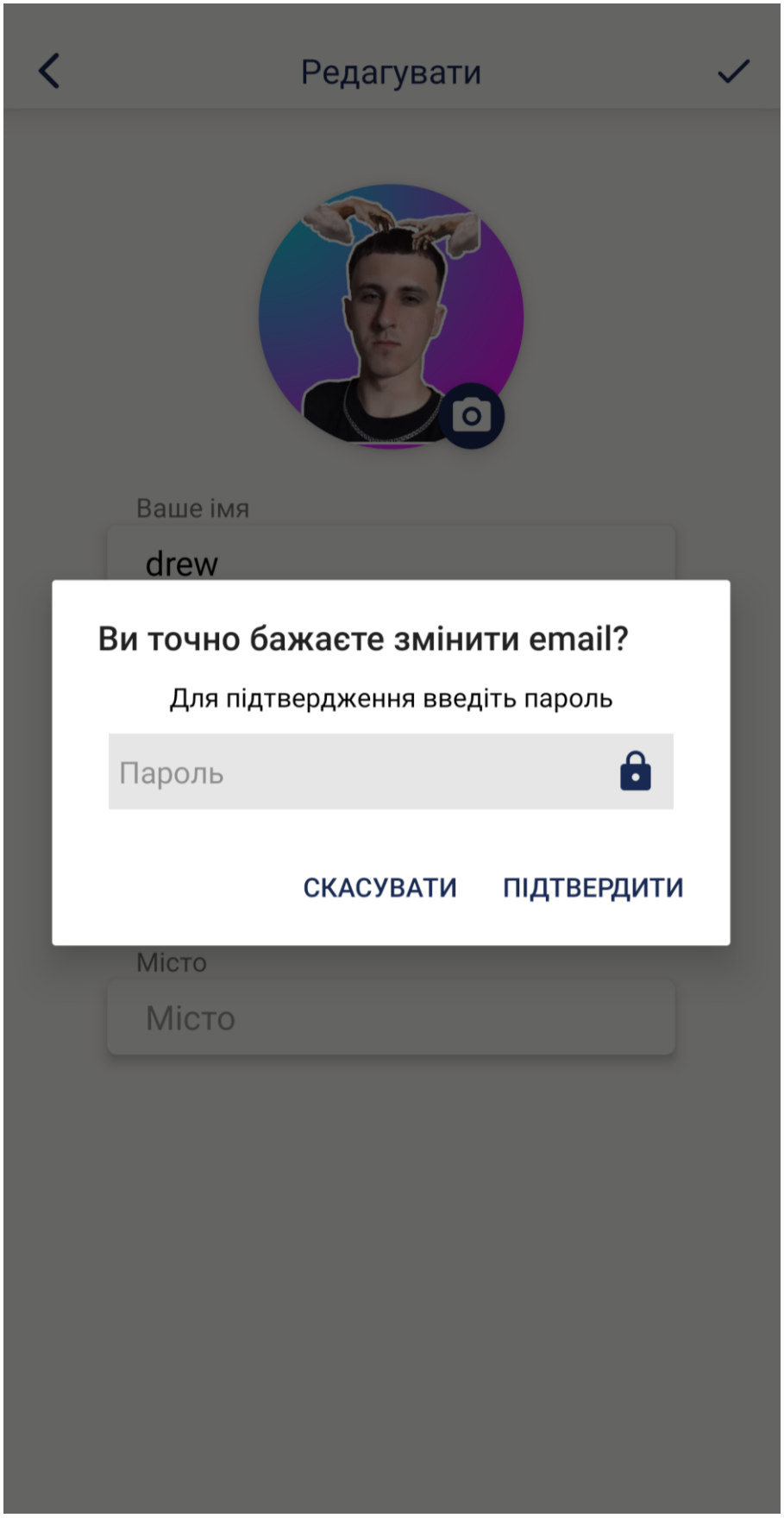

Редагувати
Ваше імя
drew
Ви точно бажаєте змінити email?
Для підтвердження введіть пароль
Пароль
СКАСУВАТИ
ПІДТВЕРДИТИ
Місто
Місто

Рис. 21

HelpActivity- екран для ознайомлення з додатком та надсилання зауважень. Зображено на рисунках 22, 23.

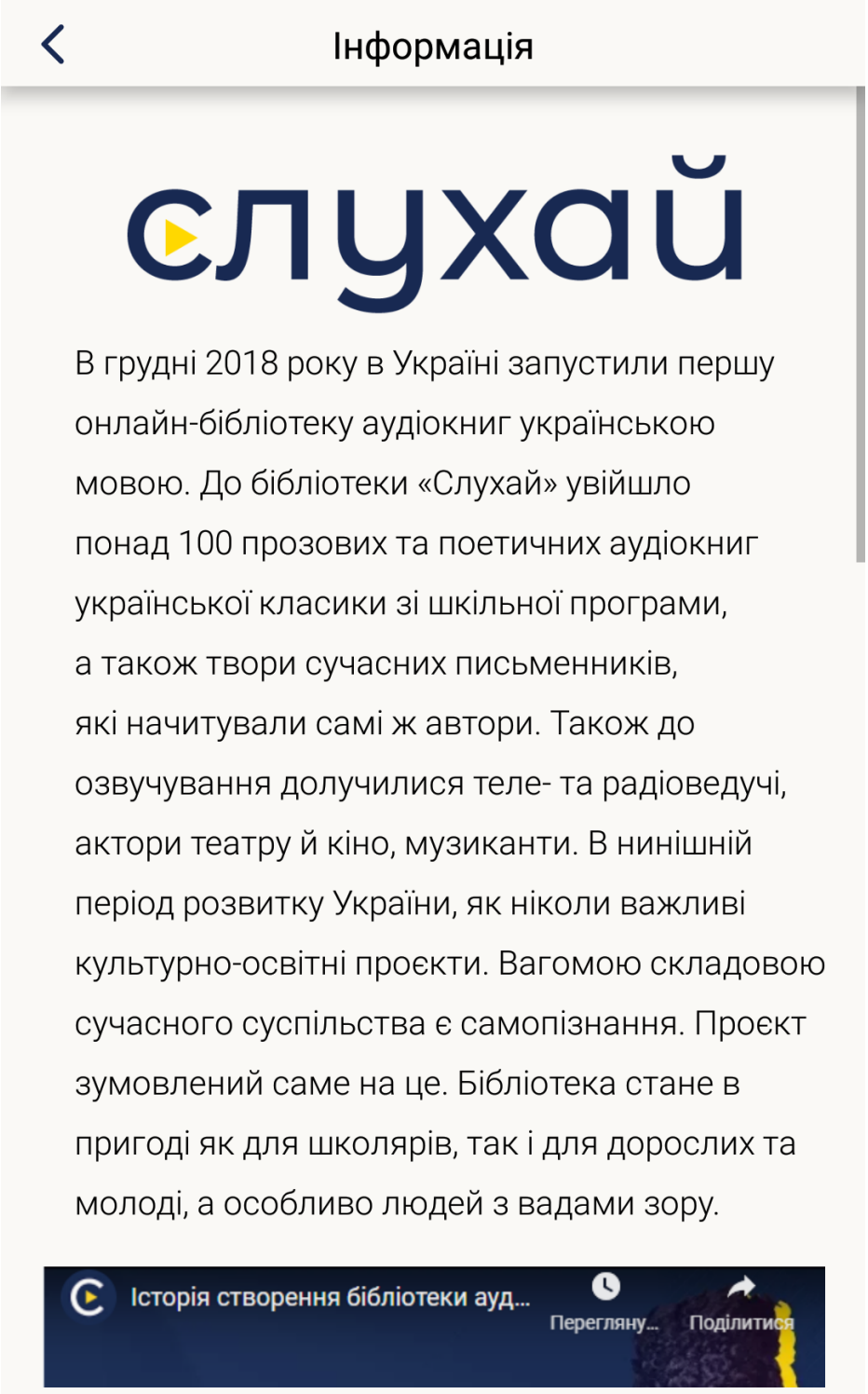


Рис. 22

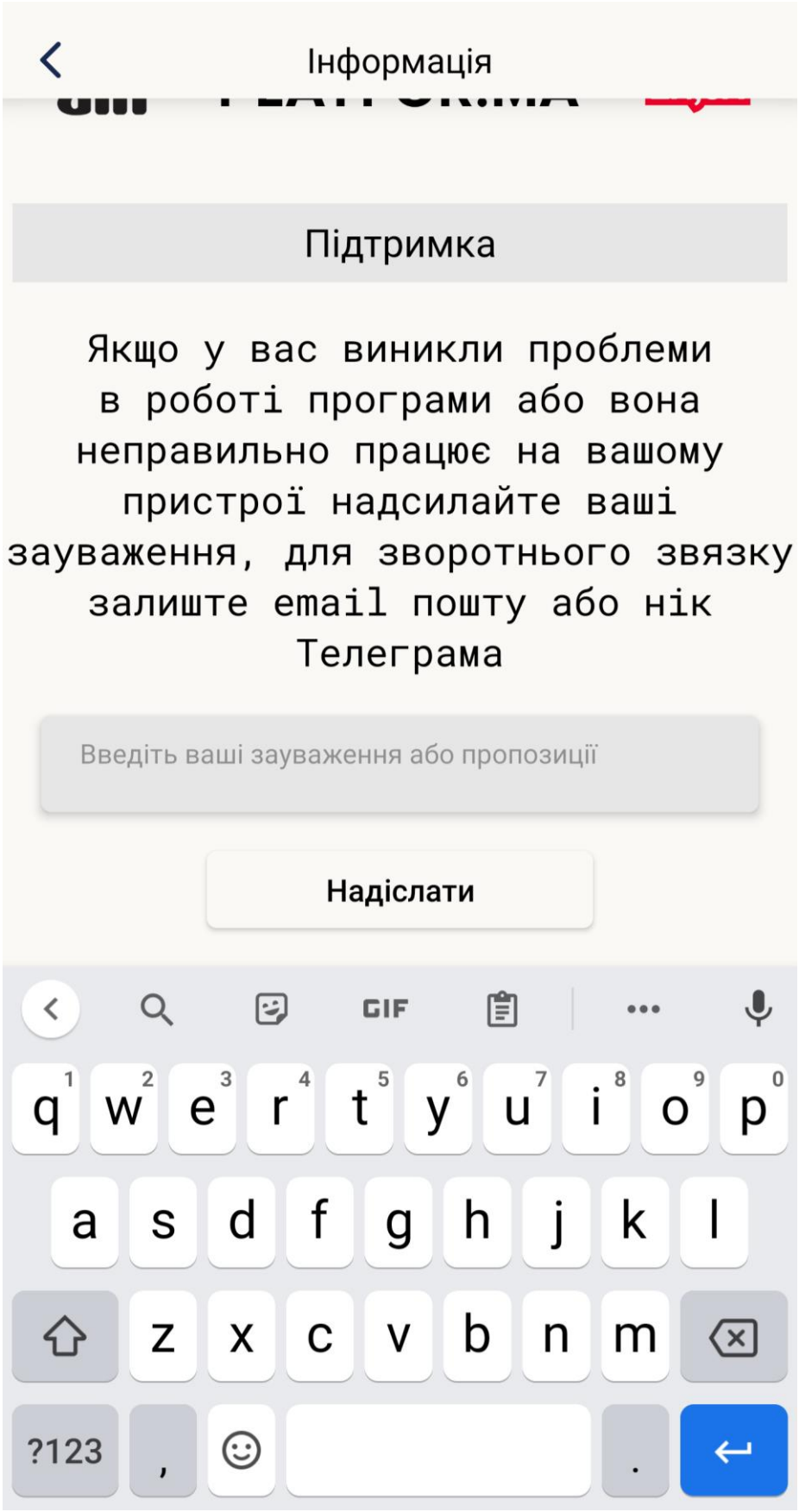

Інформація
Підтримка
Якщо у вас виникли проблеми в роботі програми або вона неправильно працює на вашому пристрої надсилайте ваші зауваження, для зворотнього звязку залиште email пошту або нік Телеграма
Введіть ваші зауваження або пропозиції
Надіслати

Рис. 23

ResetPasswordActivity- екран для скидання пароля для входу, якщо користувач забувся. На пошту буде надіслано лист з інструкцією по скиданю. Зображено на рисунку 24.

Рис. 24

SplashScreen- стартовий екран додатку, відображаться після запуску поки підгружаються дані, після завершення переходить на MainActivity. Зображено на рисунку 25.

Рис. 25

SavedActivity- екран зі збереженими творами користувача. Зображено на рисунку 26.

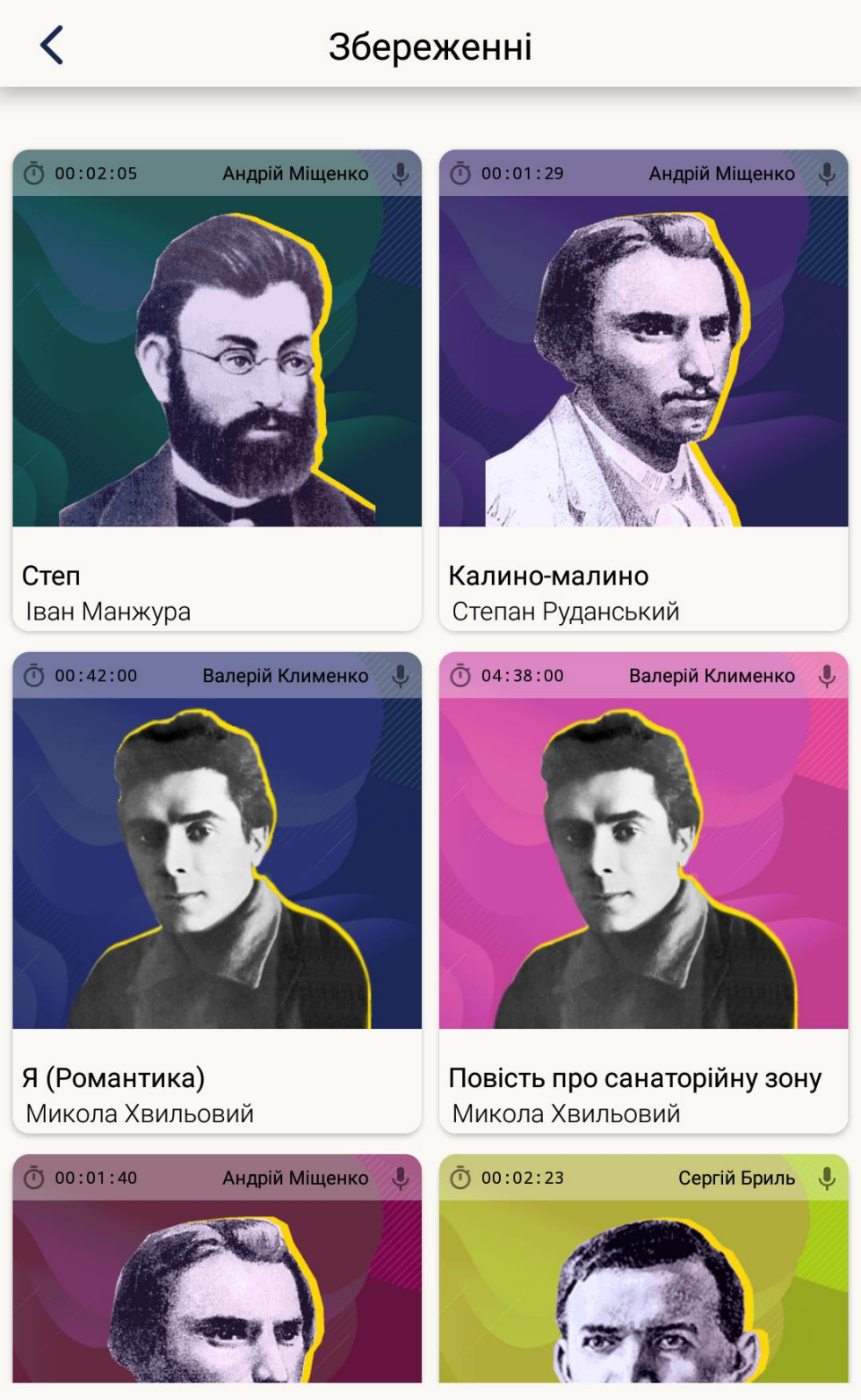


Рис. 26

PlayerActivity- екран для відтворення творів, та перегляду іншої інформації. Зображено на рисунках 27,.28, 29, 30.

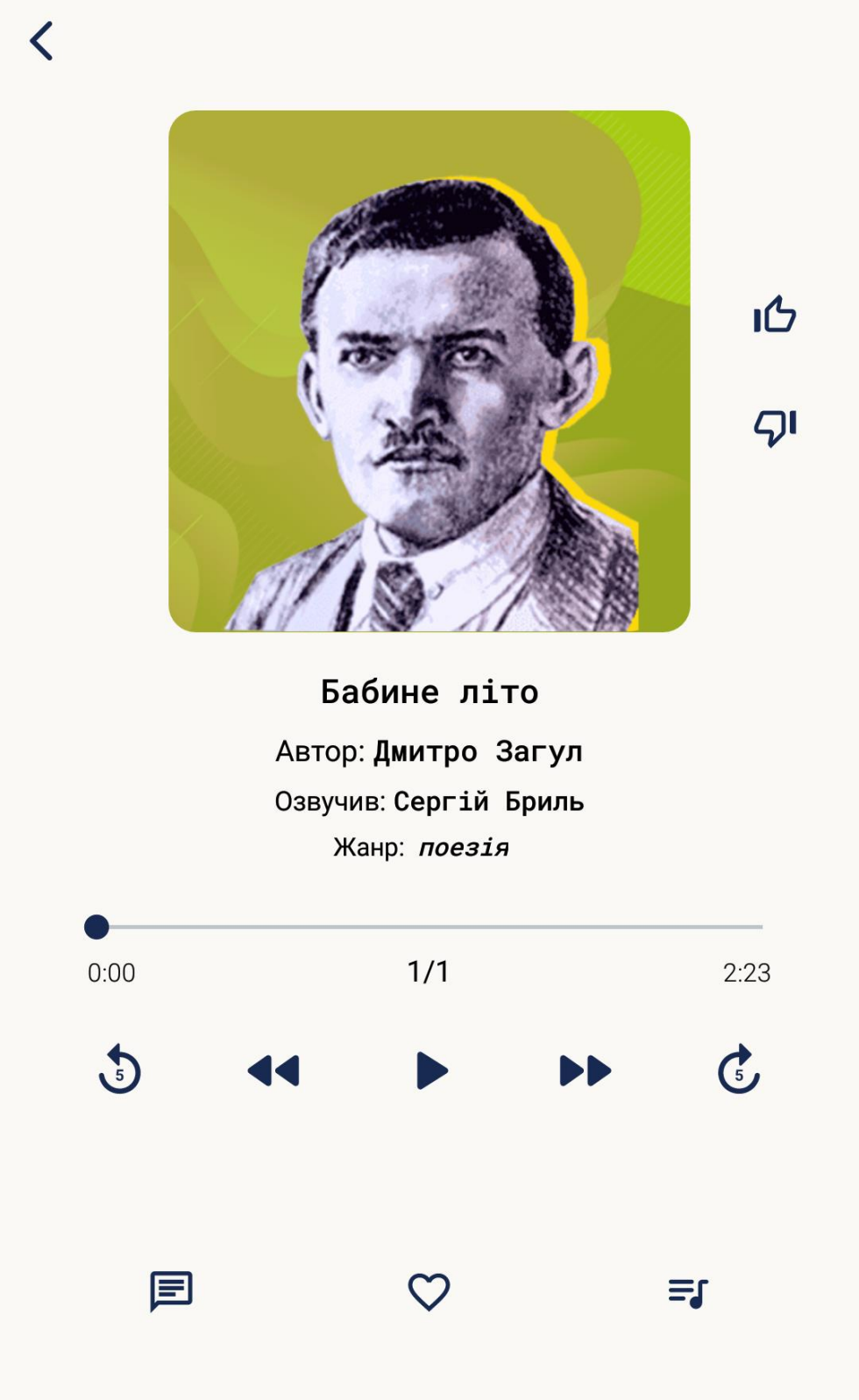


Рис. 27

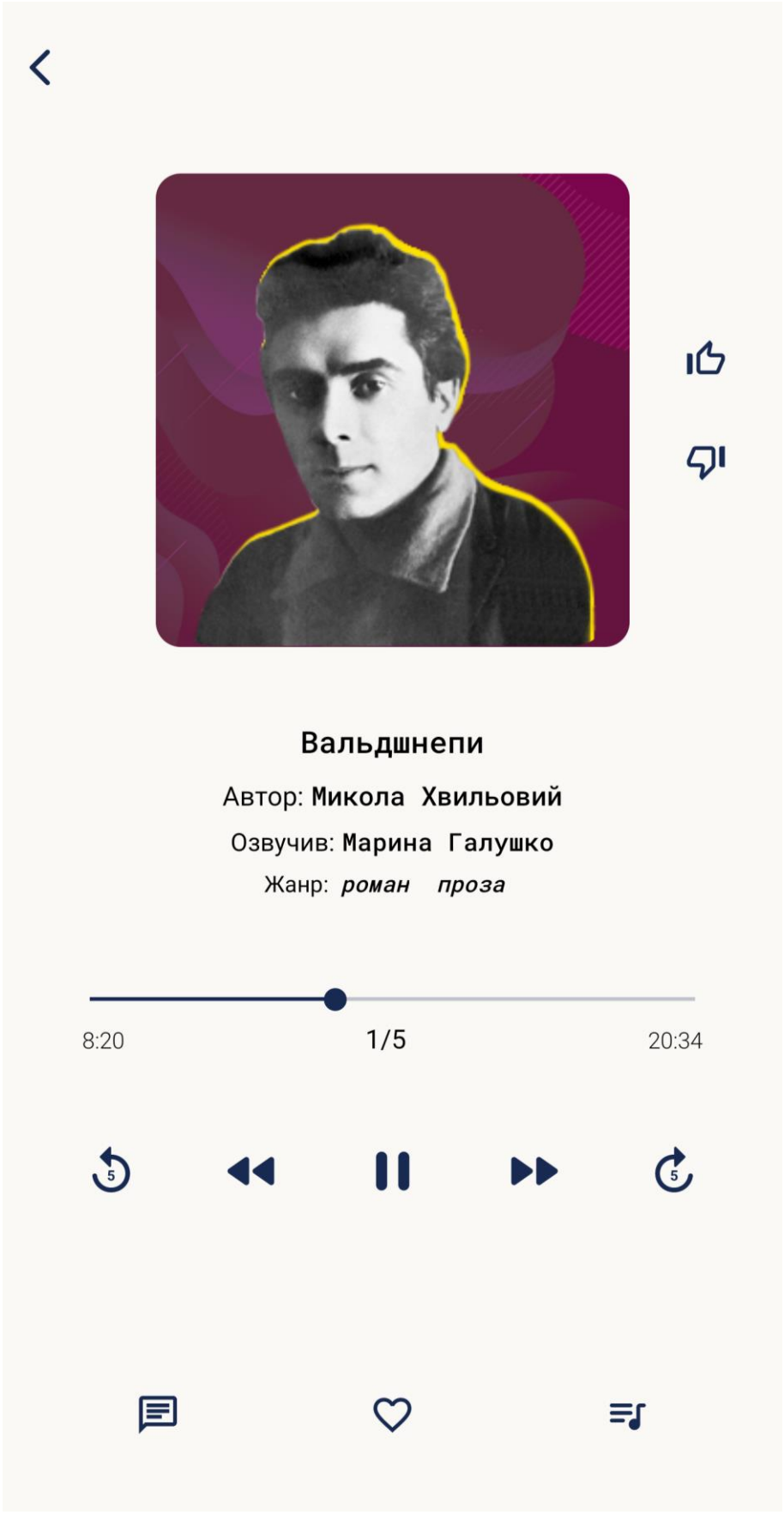

Вальдшнепи
Автор: Микола Хвильовий
Озвучив: Марина Галушко
Жанр: роман проза
8:20
1/5
20:34

Рис. 28


Плейлист
Вальдшнепи 1/5
Вальдшнепи 2/5
Вальдшнепи 3/5
Вальдшнепи 4/5
Вальдшнепи 5/5
0:00
1/5
20:34

Рис. 29

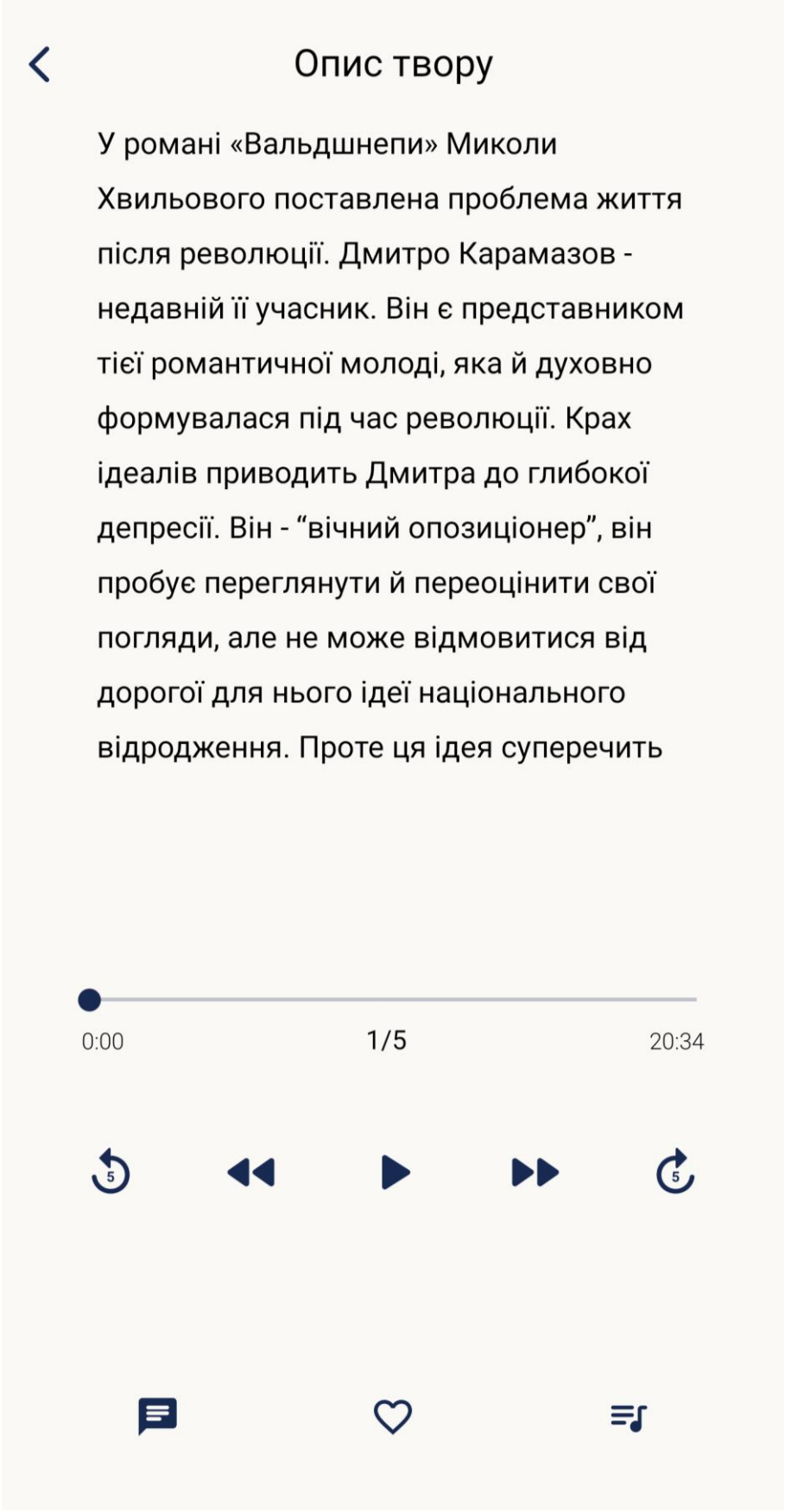

Опис твору
У романі «Вальдшнепи» Миколи Хвильового поставлена проблема життя після революції. Дмитро Карамазов - недавній її учасник. Він є представником тієї романтичної молоді, яка й духовно формувалася під час революції. Крах ідеалів приводить Дмитра до глибокої депресії. Він - “вічний опозиціонер”, він пробує переглянути й переоцінити свої погляди, але не може відмовитися від дорогої для нього ідеї національного відродження. Проте ця ідея суперечить
0:00
1/5
20:34

Рис. 30

3.3 Модуль utils

В модулі utils містяться 3 класи, зображені на рисунку 31. Вони використовуються всіма Activity. В FirebaseHelper винесені функції для роботи з БД. CustomGlideModule це перевизначення модуля Gilde для завантаження зображень, згідно документації. ValueEventListenerAdapter клас для зручного отримання інформації з БД RealtimeDatabase.

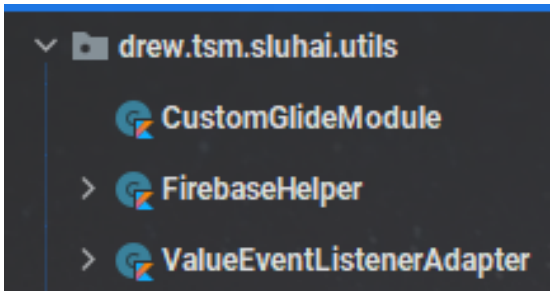


Рис. 31

3.3 Модуль views

В модулі views знаходяться 4 класи, зображені на рисунку 32. CardAdapter та ListAdapter використовуються для контролю, відображення та обробки RecyclerView. KeyboardAwareScrollView використовується для підлаштовування екрана при відкриті клавіатури для введення, щоб всі потрібні елементи для користувача були видимі. PasswordDialog діалогове вікно при зміні email користувача при редагуванні профілю, використоється для обробки веденого пароля та передачі в EditProfileActivity

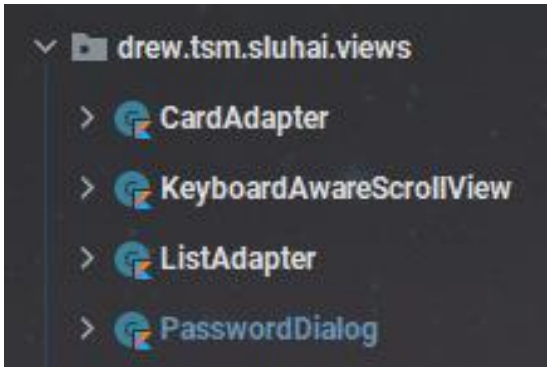


Рис. 32

3.4 Модуль models

В модулі models містяться 5 класів, зображені на рисунку 33. Вони потрібні для зберіжання в них інформації, яка отримується з Firebase, та від користувача.

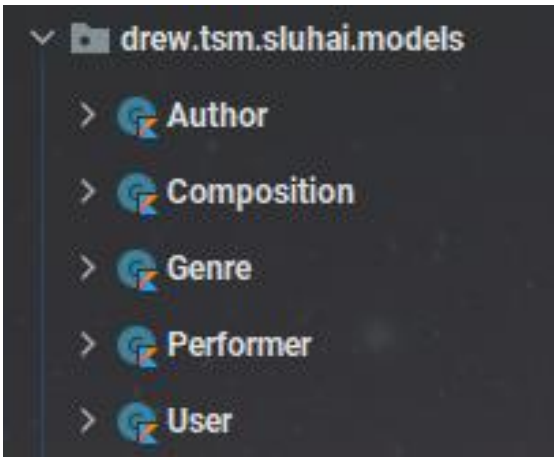


Рис.33

3.5 Висновки

В даному розділі було наведено та продемонстровано розроблений інтерфейс додатка та розказано з яких модулів складається програма та для чого вони призначені. Всі приклади супроводжувались рисунками на яких зображений інтерфейс додатку, та всі екрани між якими може переключатись користувач.

ВИСНОВКИ

Метою даного дипломного проєкту є розробка Android-додатку «Слухай» з зручним користувацьким інтерфейсом. Розробка є актуальною у зв'язку зі збільшенням потреб користувачів, а також збільшенням кількості пристроїв на ОС Android. Розроблений додаток надає можливість користувачу використовувати його для прослуховування аудіо-творів українських авторів, доповненого зручним та зрозумілим інтерфейсом та іншими функціями.

Аналіз існуючих рішень показав що у вже наявних додатків є деякі недоліки, які наявні в розробленому додатку.

Головними перевагами розробленого додатку є:

- швидкий користувацький інтерфейс;
- можливість створення профілю користувача;
- можливість відновлення паролю;
- реалізація повного функціоналу медіа-додатку;
- зберігання творів які сподобались;
- адаптивний дизайн;
- пошук у списку творів
- вибір творів по категоріях
- сортування списку творів

В майбутньому планується додати можливість коментування творів, відтворення в фоні, зберігання в кеш телефону, темна тема та інші покращення.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Руководство по языку Kotlin. URL: https://kotlinlang.ru/ (дата звернення 28.03.2021).
2. Documentation for app developers. URL: https://developer.android.com/docs (дата звернення 6.04.2021).
3. Start Android - учебник по Android для начинающих и продвинутых. URL: https://startandroid.ru/ru/ (дата звернення 11.04.2021).
4. Kotlin в действии от Жемеров Д.|Исакова С. URL: https://ru.pdfdrive.com/kotlin-%D0%B2-%D0%B4%D0%B5%D0%B9%D1%81%D1%82%D0%B2%D0%B8%D0%B8-e187850272.html (дата звернення 16.04.2021).
5. Developer Guides. URL: https://developer.android.com/guide (дата звернення 19.04.2021).
6. Патерн проектування MVC URL: https://www.tutorialspoint.com/design_pattern/mvc_pattern.htm (дата звернення 24.04.2021).
7. Firebase. URL: https://firebase.google.com/docs (дата звернення 1.05.2021).
8. Glide. URL: https://bumptech.github.io/glide/ (дата звернення 4.05.2021).
9. Stackoverflow. URL: https://stackoverflow.com/ (дата звернення 8.05.2021).
10. Google I/O. URL: https://events.google.com/io/?lng=en (дата звернення 10.05.2021).
11. Android Architecture Components. URL: https://developer.android.com/topic/libraries/architecture (дата звернення 12.05.2021).
12. Android API reference URL: https://developer.android.com/reference (дата звернення 15.05.2021).

13. Head First Android Development: A Brain-Friendly Guide URL: https://www.amazon.com/Head-First-Android-Development-Brain-Friendly/dp/1449362184 (дата звернення 18.02.2021).
14. Android Programming: The Big Nerd Ranch Guide URL: https://www.amazon.com/Android-Programming-Ranch-Guide-Guides/dp/0321804333 (дата звернення 27.02.2021).
15. Android Cookbook: Problems and Solutions for Android Developers URL: https://www.amazon.com/Android-Cookbook-Problems-Solutions-Developers/dp/1449374433 (дата звернення 4.03.2021).
16. Head First Kotlin: A Brain-Friendly Guide URL: https://www.amazon.com/Head-First-Kotlin-Brain-Friendly-Guide/dp/1491996692 (дата звернення 8.03.2021).
17. Android Studio URL: https://developer.android.com/studio (дата звернення 12.02.2021).