

Wydział Transportu i Informatyki

Specjalność: Inżynieria Oprogramowania i Technologie

Mobilne

Projekt inżynierski

**Projekt i implementacja aplikacji mobilnej “LitHaven”**

Wykonał: Oleksandr Fedyk 39503

Lublin 2025

# **Spis treści**

[**Spis treści** 1](#_Toc188187622)

[**Wprowadzenie** 2](#_Toc188187623)

[Temat 2](#_Toc188187624)

[Cel projektu 2](#_Toc188187625)

[Zakres projektu 2](#_Toc188187626)

[**1.Założenie projektowe** 3](#_Toc188187627)

[Funkcjonalności aplikacji: 3](#_Toc188187628)

[**2.Metody, Techniki i Narzędzia Realizacji Projektu** 4](#_Toc188187629)

[Zastosowane technologie i narzędzia programistyczne 4](#_Toc188187630)

[Metody projektowe i techniki pracy 5](#_Toc188187631)

[**3.Etapy Projektowania i Implementacji** 6](#_Toc188187632)

[Analiza wymagan 6](#_Toc188187633)

[Planowanie projektu 6](#_Toc188187634)

[Projektowanie UX/UI 6](#_Toc188187635)

[Przygotowanie środowiska do implementacji projektu 9](#_Toc188187636)

[Realizacja funkcjonalności i UI 11](#_Toc188187637)

[**Prezentację opracowanego projektu** 20](#_Toc188187638)

[**Wnioski i zakończenia** 21](#_Toc188187639)

[**Literatura** 22](#_Toc188187640)

[**Spis rysunków** 23](#_Toc188187641)

[**Streszczenie** 24](#_Toc188187642)

# 

# **Wprowadzenie**

## Temat

Projekt i implementacja aplikacji mobilnej “LitHeaven”

## Cel projektu

Zaprojektowanie oraz implementowanie aplikacji mobilnej przeznaczonej do pisania oraz czytania książek innych użytkowników

## Zakres projektu

Obejmuje trzy rozdziały. W rozdziale pierwszym przedstawione są opisy funkcjonalności. W rozdziale drugim przedstawione są metody, techniki i narzędzia prowadzących do realizacji celu projektu wraz z uzasadnieniem ich wyboru. W rozdziale trzecim przedstawione są opisy etapów procesu implementacji projektu. W rozdziale czwartym przedstawione są zarzuty ekranów ukazujących na działanie aplikacji. Dokumentacja zawiera również wnioski i zakończenie

# 

# **1.Założenie projektowe**

W tym rozdziale zostaną przedstawione funkcjonalności aplikacji, które będą zaimplementowane w projekcie. Aplikacja będzie służyła do pisania książek przez użytkowników, jak i do czytania książek napisanych przez innych użytkowników

## Funkcjonalności aplikacji:

1. **Logowanie do aplikacji**
   * Aplikacja umożliwia użytkownikowi zalogowanie się za pomocą adresu e-mail
2. **Zalozenia konta**
   * Użytkownik ma możliwość założenia konta w aplikacji, podając adres e-mail oraz wymagane dane rejestracyjne
3. **Stworzenia książki**
   * Aplikacja pozwala użytkownikowi na stworzenie nowej książki. Użytkownik podaje tytuł książki, opis książki oraz tematy dla stworzonej książki. Podawania okładki książki jest możliwe z urządzenia użytkownika
4. **Zarządzanie książką**
   * Aplikacja umożliwia użytkownikowi edytowanie książki i zakończenia książki, oraz nadaje możliwość dodawać nowe rozdziały
5. **Czytanie rozdziału**
   * Użytkownik może wybrać dowolny rozdział książki i go przeczytać
6. **Wyszukiwania książki**
   * Aplikacja oferuje funkcję wyszukiwania książek według tytułu lub za pomocą przypisanych do nich tematów
7. **Historia przeglądu**
   * Aplikacja zapisuje oraz nadaje użytkownikowi jego historię przeglądu
8. **Subskrypcja na autora**
   * Użytkownik może zasubskrybować autora, co pozwala na łatwe odnalezienie go w specjalnej liście autorów
9. **Dodanie książki do biblioteki**
   * Użytkownik może dodawać wybrane książki do swojej biblioteki, co ułatwia ich późniejsze odnalezienie

# 

# **2.Metody, Techniki i Narzędzia Realizacji Projektu**

W tym rozdziale przedstawiono metody, techniki i narzędzia zastosowane do realizacji projektu. Kryteriami wyboru były m.in. wydajność, dostępność technologii oraz możliwość szybkiej implementacji funkcjonalności.

## Zastosowane technologie i narzędzia programistyczne

**Język programowania:**

Do realizacji projektu wybrano język **Kotlin**. Dla tego że ma on znacznie nowocześniejszy kod, wymaga mniej kodu dla tych samych operacji, co zmniejsza ryzyko błędów, ma lepsze zarządza nullami i ma lepsze wsparcie dla nowych projektów na Androida. W 2017 roku Google ogłosiło wsparcie dla Kotlina jako oficjalnego języka programowania dla Androida. Od 2019 roku Kotlin jest preferowanym językiem dla nowych projektów na Androida. Na osnowie tego co jest wyżej napisane językiem programowania jest wybrany Kotlin

**Framework**

W projekcie zastosowano framework **Android Jetpack**. Android Jetpack jest to zestaw bibliotek od Google, które ułatwiają tworzenie nowoczesnych aplikacji na Androida. W

szczególności wykorzystano bibliotekę Navigation, która umożliwia łatwe zarządzanie przejściami między fragmentami aplikacji.

**Baza danych**

Do przechowywania danych wybrano bazę **Firestore Cloud**. Firestore Cloud jest to skalowalna, dokumentowa baza danych NoSQL zaprojektowana do przechowywania i synchronizowania danych dla aplikacji mobilnych, webowych oraz serwerowych. Jest zintegrowana z Firebase SDK, co ułatwia jej wdrożenie w projekcie. Dodatkowo oferuje szybki odczyt zapytań i współpracę z Firebase Authentication oraz Firebase Storage, które również zostały wykorzystane w projekcie.

**Narzędzia pomocnicze**

1. **Firebase Authentication**
   * Firebase Authentication to usługa umożliwiająca łatwe zarządzanie użytkowników w aplikacjach. Pozwala na logowanie i rejestrację użytkowników przy użyciu różnych metod, takich jak adres e-mail, hasło, konta społecznościowe czy anonimowe logowanie. Wykorzysta jest ze względu na łatwą integrację z Firestore Cloud
2. **Firebase Storage**
   * Firebase Storage to usługa przechowywania danych w chmurze zaprojektowana specjalnie do obsługi aplikacji mobilnych i webowych. Umożliwia przechowywanie dużych plików, takich jak obrazy, filmy, dokumenty czy audio, z łatwym dostępem dla użytkowników aplikacji.Wykorzysta jest ze względu na łatwą integrację z Firestore Cloud

**Środowiska programistyczne**

Do tworzenia aplikacji wybrano **Android Studio**. Android Studio to oficjalne zintegrowane środowisko programistyczne (IDE) do tworzenia aplikacji na platformę Android. Przyczyną wyboru jest to że Android Studio oferuje narzędzia i funkcje zaprojektowane specjalnie do tworzenia aplikacji mobilnych na urządzenia z systemem Android.

## Metody projektowe i techniki pracy

W projekcie zastosowano **modelowanie UML.** Modelowanie UMLto standardowy język graficzny stosowany do wizualizowania, specyfikowania, projektowania i dokumentowania systemów informatycznych. Jest wybrany ze względu na to że jest wygodny w planowaniu struktury i funkcjonowania systemu przed jego implementacją

# 

# **3.Etapy Projektowania i Implementacji**

W tym rozdziale przedstawiono etapy projektowania i implementacji aplikacji “LitHaven”. Proces tworzenia aplikacji podzielono na następujące etapy:

1. Analiza wymagan
2. Planowanie projektu
3. Projektowanie UX/UI
4. Przygotowanie środowiska do implementacji projektu
5. Realizacja funkcjonalność i UI

## Analiza wymagan

W tym etapie jest definiowane są cele projektu, zakres projektu, ograniczenia projektowe. Cele projektu, zakres projektu oraz ograniczenia projektowe jest napisane we wprowadzeniu.

## Planowanie projektu

Na tym etapie definiowane są kluczowe funkcjonalności aplikacji. Ustalono, jakie technologie, narzędzia programistyczne oraz metodyki projektowe i techniki pracy zostaną zastosowane.

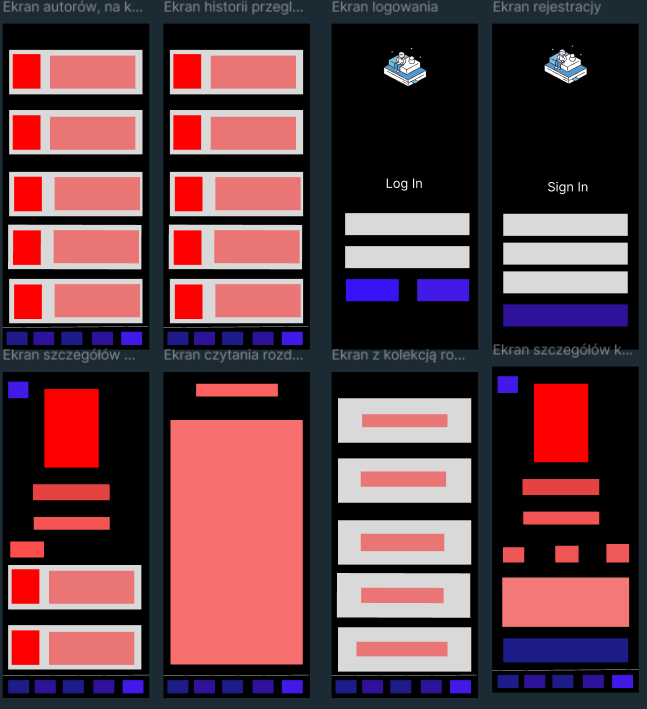
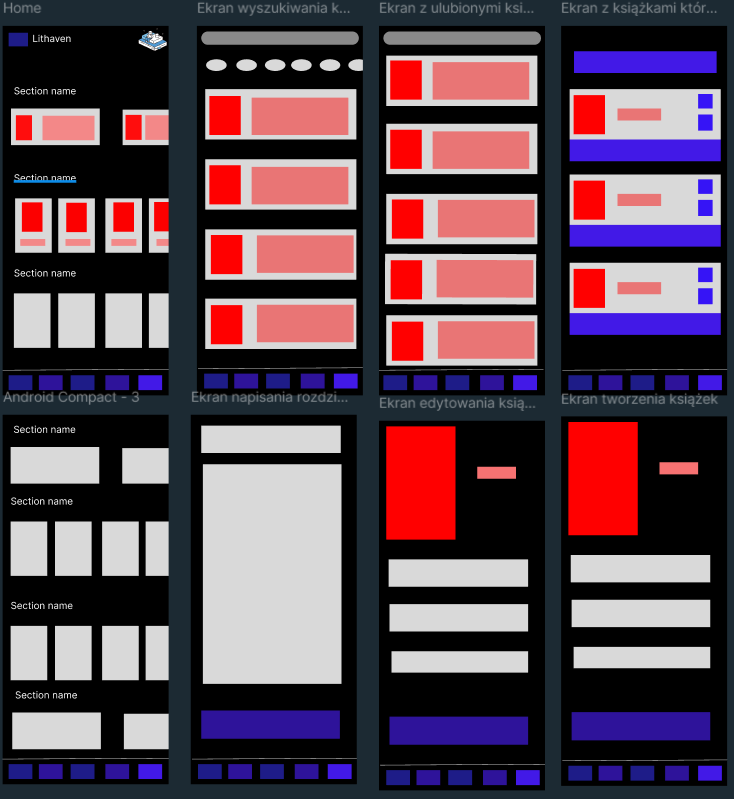
Kluczowe funkcjonalności zostały opisane w rozdziale **Założenia projektowe**, natomiast zastosowane technologie, narzędzia programistyczne oraz metodyki i techniki pracy – w rozdziale **Metody, Techniki i Narzędzia Programistyczne**.

## Projektowanie UX/UI

Na tym etapie zaprojektowano interfejs użytkownika, diagram ruchu użytkownika w aplikacji oraz interakcje użytkownika z aplikacją. Szkic układu ekranów został stworzony przy pomocy narzędzia Figma. Figma jest wybrana ze względu tego że jest wątła w używaniu oraz projekty są przechowywane w chmurze i dostępne z dowolnego urządzenia

Aplikacja będę składała się z:

1. **Ekran logowania**
2. **Ekran rejestracji**
3. **Ekran główny (Home)**
4. **Ekran wyszukiwania książek**
5. **Ekran z ulubionymi książkami**
6. **Ekran z książkami które stworzył użytkownik**
7. **Ekran tworzenia książek**
8. **Ekran edytowania książki**
9. **Ekran napisania rozdziału**
10. **Ekran szczegółów książki**
11. **Ekran z kolekcją rozdziałów**
12. **Ekran czytania rozdziału**
13. **Ekran historii przeglądania**
14. **Ekran autorów, na których użytkownik zasubskrybował**
15. **Ekran szczegółów autora**

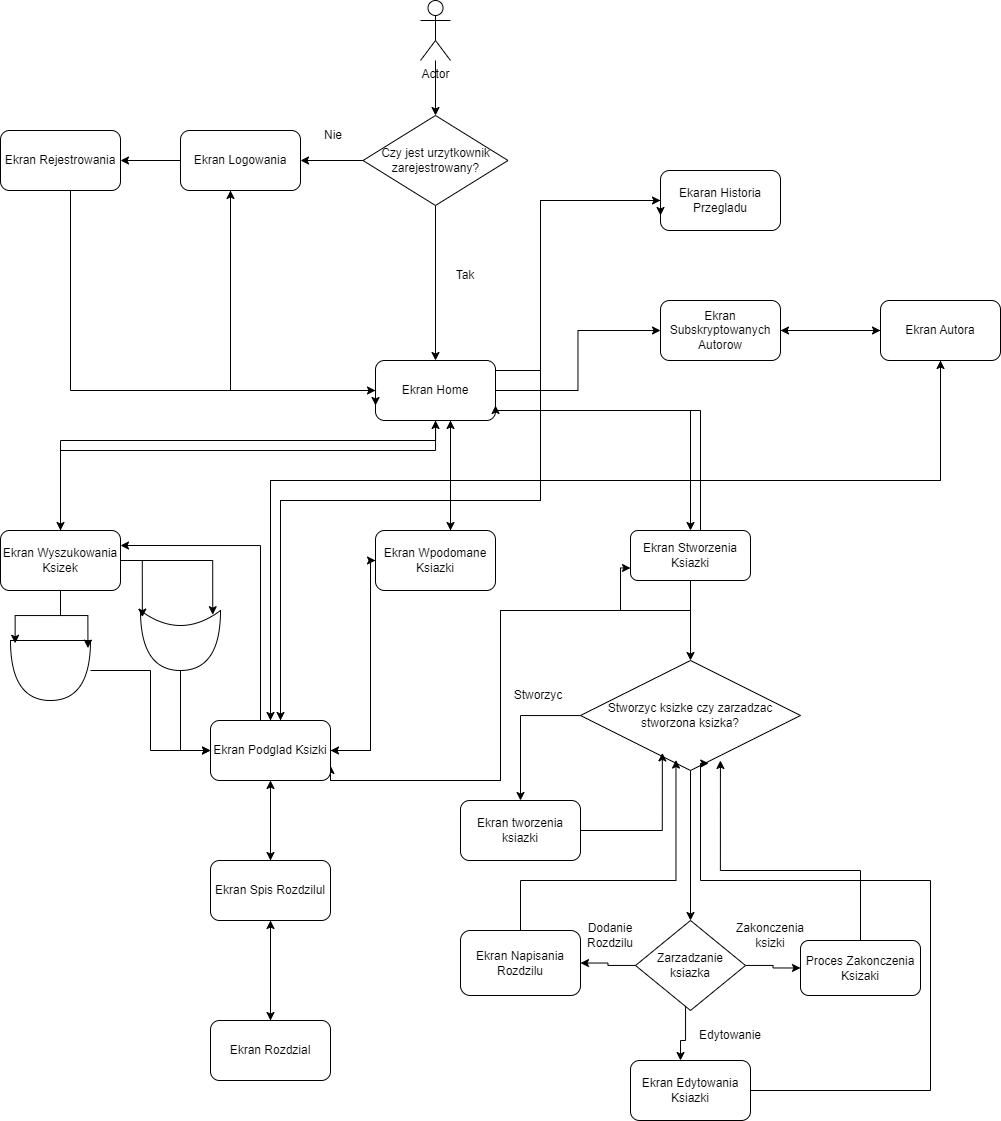


Rys. . Szkic ekranów cz 2

Rys. . Szkic ekranów cz 1

Diagram ruchu użytkownika w aplikacji oraz interakcje użytkownika z aplikacją został zaprojektowany w narzędziu **Draw.io** przy użyciu języka graficznego UML.

Na rysunku 3 przedstawiono diagram ruchu użytkownika w aplikacji oraz jego interakcje z aplikacją.



Rys. 3. Diagram UML

## Przygotowanie środowiska do implementacji projektu

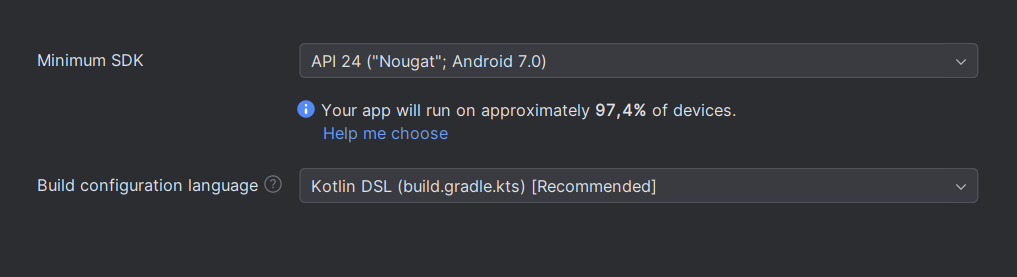
Na tym etapie zainstalowano środowisko programistyczne, utworzono projekt, podłączono bazę danych oraz skonfigurowano dodatkowe narzędzia programistyczne. Dodano również grafiki do projektu i zainicjowano kolory, które będą wykorzystywane w aplikacji.

**Przygotowanie środowiska**

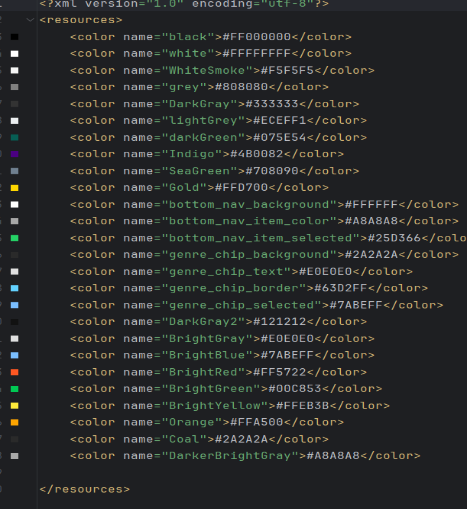
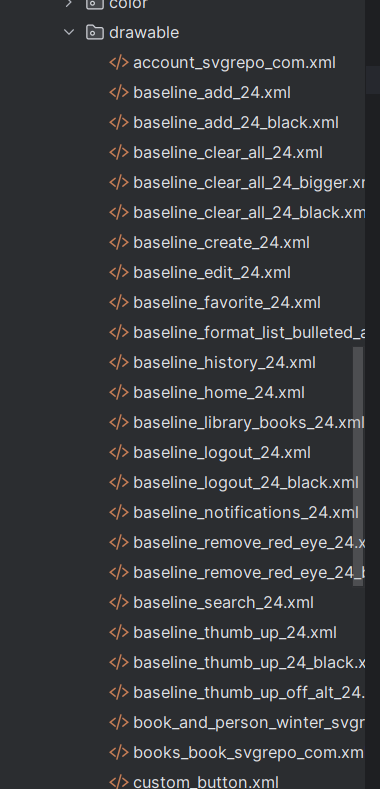
Jako środowisko programistyczne wybrano **Android Studio**, ponieważ zostało stworzone specjalnie do budowy aplikacji mobilnych na platformę Android. Android Studio zostało pobrane z oficjalnej strony [Android Developers](https://developer.android.com/studio?hl=pl)

Minimalną wersją SDK ustawiono **API 24** („Nougat”; Android 7.0), co pozwala na działanie aplikacji na około **97% urządzeń z systemem Android**

Językiem programowania wybranym do projektu jest **Kotlin**, który jest rekomendowany przez Android Studio i szeroko wspierany przez społeczność deweloperów Androida



Rys. 4. Konfiguracja projekta

****Do projektu dodano wszystkie wymagane grafiki (rysunek 5) oraz zdefiniowano paletę kolorów (rysunek 6), która będzie konsekwentnie stosowana w aplikacji

Rys. . Dodanie kolorów

Rys. . Dodanie grafiki

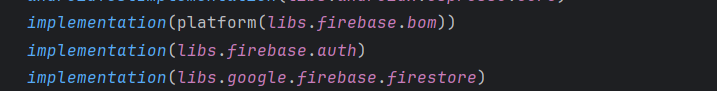
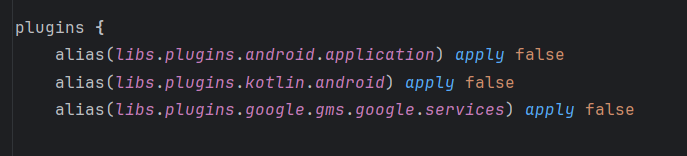
**Podłączenia bazy danych oraz dodatkowych narzędzi programistycznych**

W projekcie wykorzystano następujące narzędzia i usługi:

1. **Firestore Cloud**
   * **Firestore Cloud** (Google Firestore) to baza danych **NoSQL** w chmurze, oferowana przez Google w ramach platformy Firebase.
2. **Firebase Authentication**
   * Używane do zarządzania procesem rejestracji i logowania użytkowników.
   * Obsługuje różne metody logowania, takie jak e-mail i hasło, Google, Facebook itp.
3. **Firebase Storage**
   * Wykorzystywane do przechowywania i pobierania plików, takich jak obrazy książek, w sposób skalowalny i bezpieczny.

**Firestore Cloud** jest podłączony przy pomocy **Firebase SDK**. W przypadku, gdy pojawią się problemy na etapie konfiguracji, konieczne może być ręczne dodanie następujących zależności do pliku build.gradle:

* **'com.google.gms:google-services:4.4.2'**
* **apply plugin: 'com.google.gms.google-services'**
* **'com.google.firebase:firebase-firestore:25.1.1'**

Na rysunku 7 oraz rysunku 8 pokazano komendy które zostały dodane samodzielnie w ramach tego projektu.

Rys. . Dodanie bibliotek

Rys. 8. Dodane plaginy

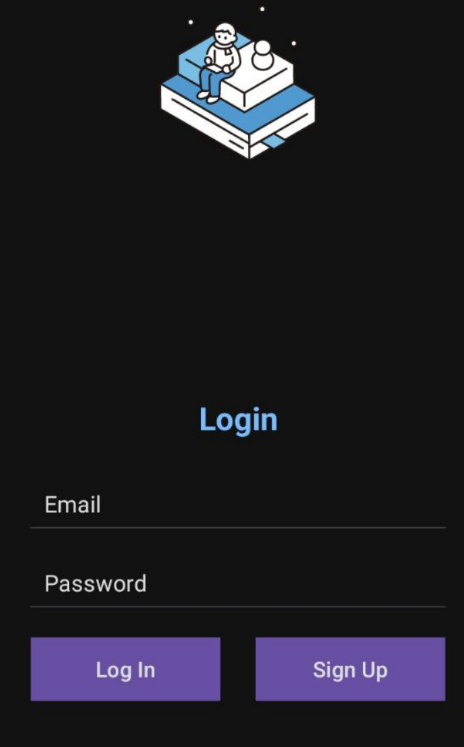
Instalacja **Firebase Storage** oraz **Firebase Authentication** jest możliwa przez **Firebase Console**. Komendy dla **Firebase Storage** oraz **Firebase Authentication** zostały pokazane na rysunku 7 oraz rysunku 8

## Realizacja funkcjonalności i UI

Na tym etapie realizowane są funkcjonalności oraz interfejs użytkownika (UI) dla każdego z ekranów aplikacji. Proces ten obejmuje następujące kroki:

1. **Projektowanie interfejsu użytkownika**W tym kroku tworzony jest wygląd aplikacji, który odpowiada za interakcję użytkownika z systemem. Dla każdego ekranu:
   * Projektowane są elementy interfejsu, takie jak przyciski, pola tekstowe, formularze, listy książek, itp.
   * Interakcje użytkownika są definiowane poprzez odpowiednie połączenia między elementami UI
2. **Tworzenie i implementacja funkcji**Dla każdego ekranu aplikacji definiowane są kluczowe funkcje, które umożliwiają realizację założonych celów. Do funkcjonalności tych należą:
   * Logowanie i rejestracja użytkowników
   * Przeszukiwanie bazy danych książek
   * Tworzenie i edytowanie książek
   * Dodanie rozdziału
   * Dodanie książki do biblioteki użytkownika
   * Wyświetlanie szczegółowych informacji o książkach
   * Historia przeglądania książek
   * Subskrypcje autorów
3. **Integracja funkcji z interfejsem**Na tym etapie funkcjonalności są łączone z interfejsem użytkownika:
   * Każdy przycisk, formularz czy lista jest powiązany z odpowiednią logiką
   * Testowane są interakcje, aby upewnić się, że aplikacja działa zgodnie z założeniami, a użytkownik otrzymuje odpowiednie komunikaty zwrotne

**Ekran logowania**

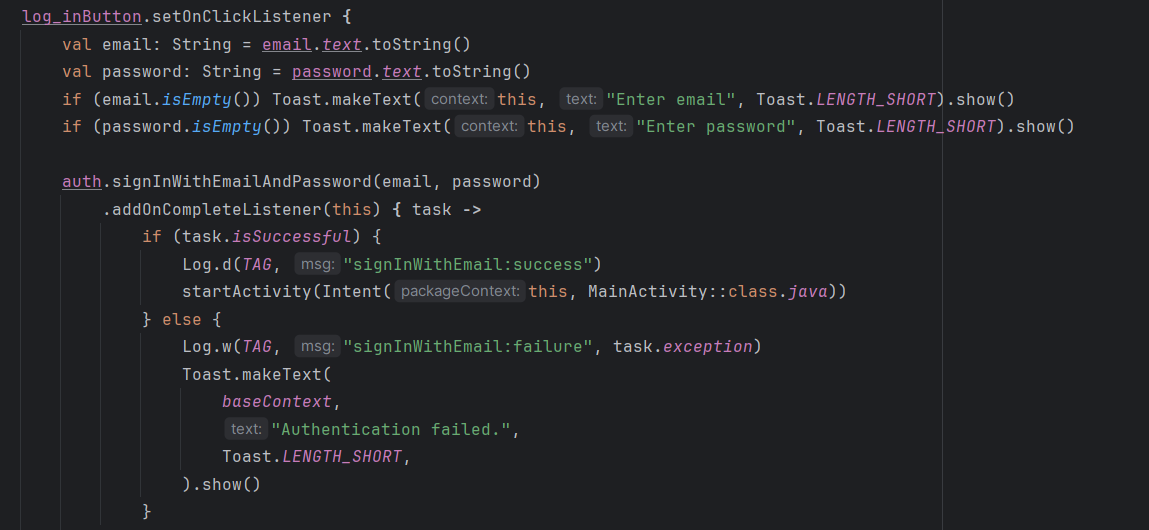


Rys. . Ekran logowania

Na rysunku 9 jest pokazany ekran logowania. Składa się z dwóch elementów **EditText**: jedno pole do wprowadzenia “Email” oraz drugie do wprowadzenia “Password”, dwóch przycisków **Button** z tekstami “Log In” oraz “Sign Up”, jednego elementu **TextView** z tekstem “Login” oraz elementu **ImageView**.

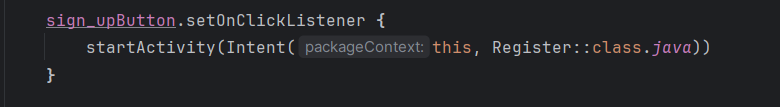
Przycisk **“Log In”**, po kliknięciu, pobiera dane (e-mail i hasło) wprowadzone przez użytkownika i sprawdza ich poprawność.

Na rysunku 10 pokazano logikę logowania. Jeśli którykolwiek z pól (e-mail lub hasło) jest pusty, wyświetlany jest odpowiedni komunikat **Toast**. Jeśli dane są poprawne, funkcja **signInWithEmailAndPassword** z Firebase Authentication jest wywoływana.W przypadku sukcesu użytkownik zostaje przekierowany do głównej aktywności aplikacji **MainActivity**. W przypadku błędu, użytkownik zostaje poinformowany o niepowodzeniu **Toast**.



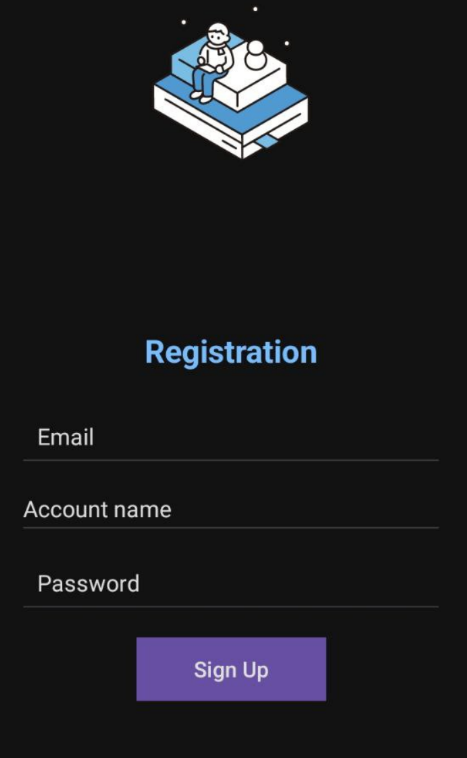
Rys. 10. Logika logowania

Przycisk **“Sign Up”**, po kliknięciu użytkownik zostaje przeniesiony do ekranu rejestracji **Register**

****

Rys. 11. Przeniesienie do ekranu refestracji

**Ekran rejestrowania**



Rys. . Ekran rejestrowania

Na rysunku 12 jest pokazany ekran rejestrowania. Składa się z trzech elementów **EditText**: jedno pole do wprowadzenia “Email”, drugie do wprowadzenia “Account name”, oraz trzecie do wprowadzenia “Password”, jednego przycisku **Button** z tekstem “Sign Up”, jednego elementu **TextView** z tekstem “Registration” oraz elementu **ImageView**.

Po naciśnięciu przycisku **“Sign Up”**, aplikacja pobiera dane z pól **“Email”**, **“Account Name”** oraz **“Password”**, które użytkownik wprowadził. Następnie te dane są walidowane w celu zapewnienia, że wszystkie wymagane pola zostały wypełnione. Jeśli pola są puste, to użytkownik otrzymuje odpowiedni komunikat

Jeśli wszystkie dane zostały wprowadzone prawidłowo, aplikacja przechodzi do próby rejestracji użytkownika w Firebase Authentication przy użyciu funkcji **createUserWithEmailAndPassword(email, password).**

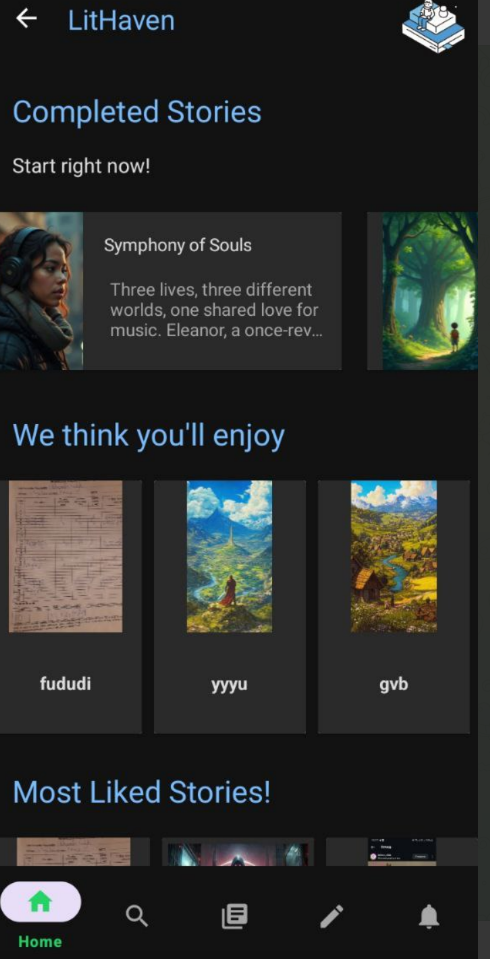
Jeśli proces rejestracji w Firebase Authentication zakończy się sukcesem, użytkownik zostaje zapisany w systemie i utworzony jest nowy dokument w bazie danych Firestore. W tym dokumencie znajdują się informacje o użytkowniku, takie jak “userId”, “email” oraz “accountName”.

Po pomyślnej rejestracji użytkownik zostaje przekierowany do głównej aktywności aplikacji **MainActivity**



Rys. . Logika rejestrowania

**Ekran Home**

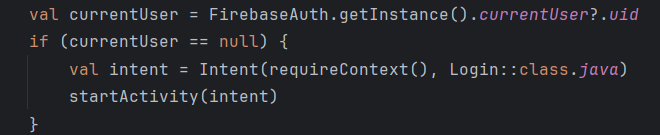
Na rysunku 14 jest pokazany ekran Home. Jest to fragment aplikacji odpowiedzialny za wyświetlanie różnych kategorii książek w postaci poziomych list RecyclerView. Głównym celem tego fragmentu jest zaprezentowanie użytkownikowi spersonalizowanych rekomendacji i popularnych książek.

Rys. . Ekran Home

Fragment zawiera kilka RecyclerView, które prezentują różne zestawy książek, takie jak:

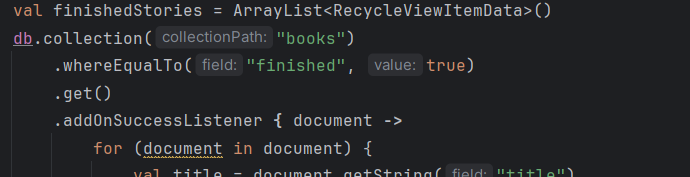
1. Ukończone historie (Finished Stories)
2. Książki, które mogą się spodobać (Books You Will Like)
3. Najczęściej lubiane książki (Most Liked Books)
4. Topowe wybory (Top Picks)
5. Obszerne historie (Huge Stories)
6. Najpopularniejsze książki (Most Popular Books).

Fragment sprawdza, czy użytkownik jest zalogowany. Jeśli nie, następuje automatyczne przekierowanie na ekran logowania **Login Activity**.



Rys. 15. Sprawdzenia czy użytkownik jest zalogowany

Fragment zawiera szereg funkcji do pobierania danych z Firestore i aktualizowania RecyclerView:

**loadBooksForFinishedStories()**

Rys. . loadBooksForFinishedStories()w

Pobiera listę ukończonych książek i aktualizuje adapter RecyclerView.

**loadBooksWeThinkYouWillEnjoy()**

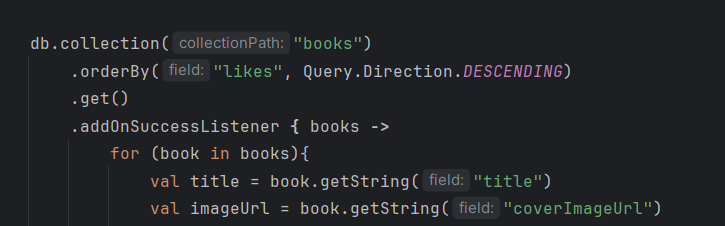
Pobiera książki napisane przez autorów, których użytkownik polubił. W tym celu wykorzystuje listę **likedAuthors** zapisaną w dokumencie użytkownika.



Rys. 17. loadBooksWeThinkYouWillEnjoy()

**loadBookTheMostLikes()**

Pobiera książki posortowane według liczby polubień **likes** w kolejności malejącej.

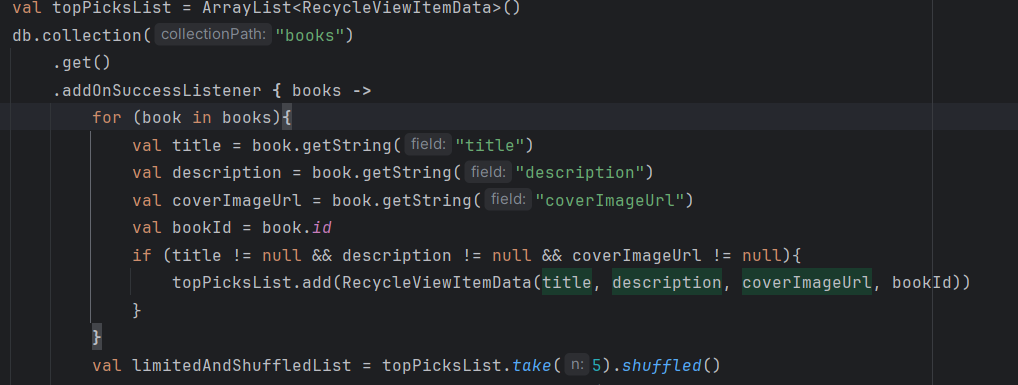
****

Rys. 18. loadBooksTheMostLikes()

**loadBooksForTopPicks()**

Pobiera losowe książki z bazy i ogranicza wynik do 5 pozycji, które są następnie wyś

wietlane w RecyclerView.



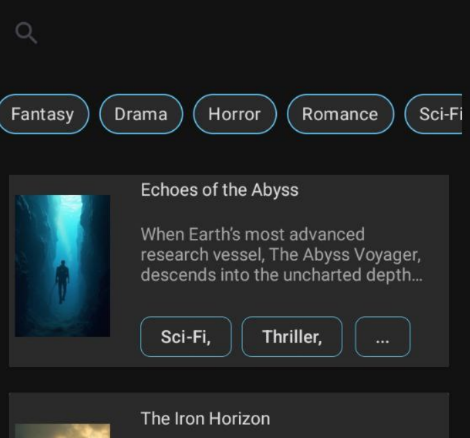
Rys. 19. loadBooksForTopPicks()

**loadBooksHugeStories()**

Pobiera książki z największą liczbą rozdziałów **chapterCount** w kolejności malejącej.

**loadBooksForMostPopular()**

Pobiera najczęściej czytane książki **amountOfReads** w kolejności malejącej.

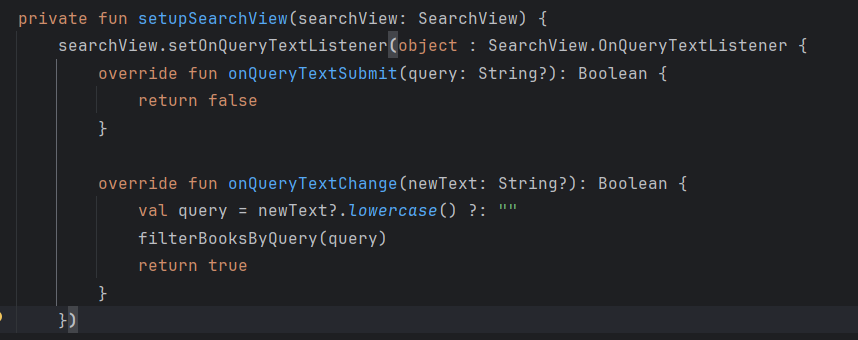
**Ekran wyszukiwania książek**

Rys. . Ekran wyszukiwania

Ekran Search (rys.20) to fragment aplikacji odpowiedzialny za wyświetlanie książek w postaci listy RecyclerView. Głównym celem tego fragmentu jest znalezienie książki zależnie od podanej nazwy oraz wybranego tematu.

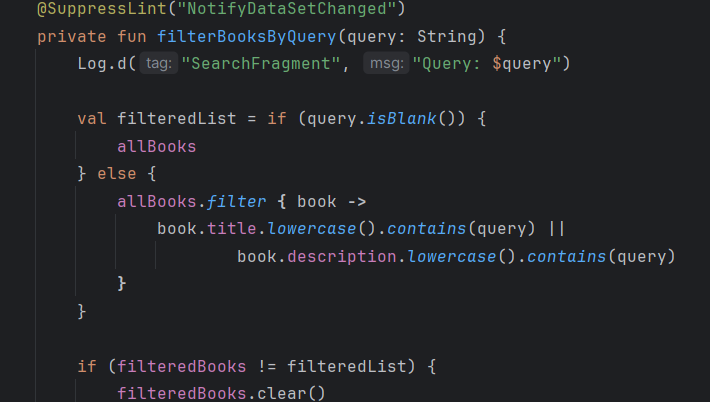
Mechanizm wyszukiwania w aplikacji opiera się na wykorzystaniu komponentu **SearchView** oraz **ChipGroup**

**SearchView**

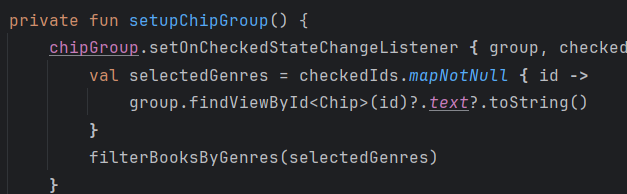
Wprowadzony tekst jest monitorowany nasłuchiwaczowi zdarzeń **OnQueryTextListener**, który reaguje na każdą zmianę tekstu w polu wyszukiwania. Kiedy użytkownik wpisuje lub modyfikuje zapytanie, funkcja **onQueryTextChange(newText: String?)** automatycznie przekazuje aktualny tekst do metody **filterBooksByQuery(query: String)**. 

Rys. . Konfiguracja SearchView

Rys. . Filtrowanie książek za nazwą

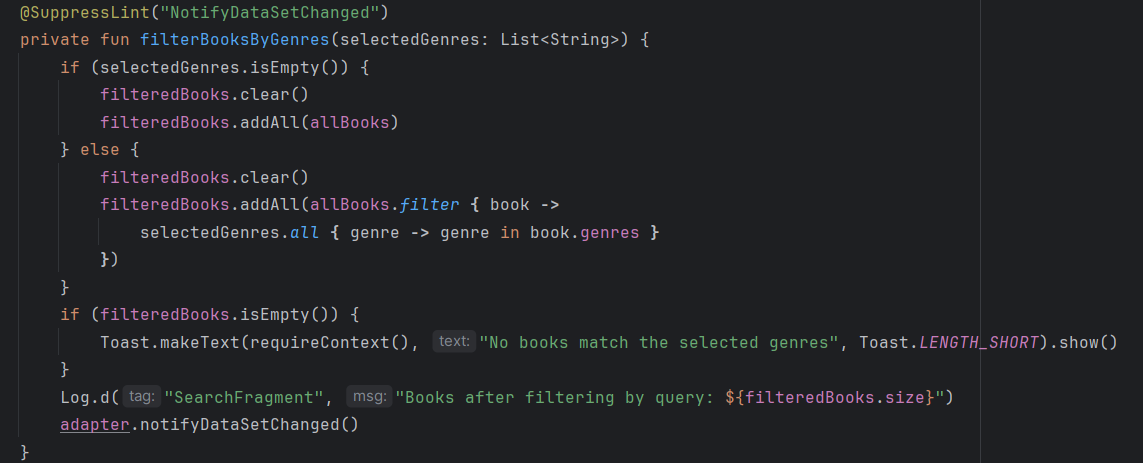
W metodzie **filterBooksByQuery** zaimplementowano logikę, która analizuje pełną listę książek **allBooks** i wyodrębnia te pozycje, które spełniają kryteria wyszukiwania.

**ChipGroup**

Kiedy użytkownik wybiera lub odznacza gatunek, zdarzenie to jest przechwytywane przez nasłuchiwacz **setOnCheckedStateChangeListener** przypisany do elementu **ChipGroup**. Dla każdego wybranego elementu identyfikator zaznaczonego chipu jest przekazywany do funkcji **filterBooksByGenres(selectedGenres: List<String>)**, która przetwarza dane w celu aktualizacji widoku listy książek.

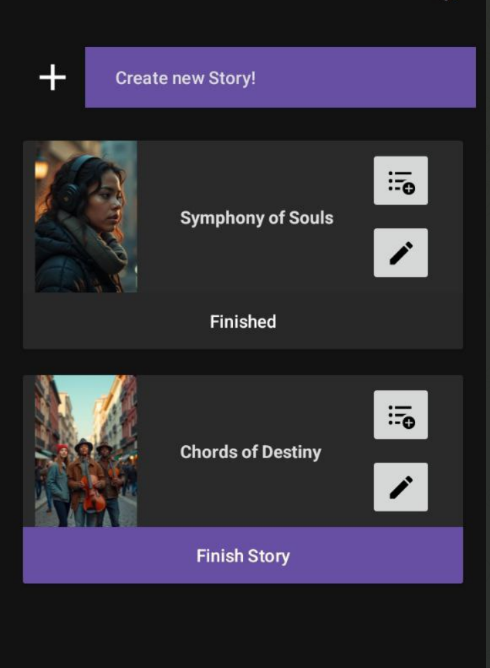
Rys. . Konfiguracja ChipGroup

Lista wszystkich książek **(allBooks)** jest przeszukiwana w celu znalezienia pozycji, których gatunki spewnia warunki

Jeżeli użytkownik nie wybierze żadnego gatunku, zostaje wyświetlona pełna lista książek. Lista wyników filtrowania jest przechowywana w zmiennej **filteredBooks**, a adapter **RecyclerView** jest powiadamiany o zmianach za pomocą metody **adapter.notifyDataSetChanged()**

Rys. . Filtrowanie ksiązek za gatunkem

**Ekran do stworzenia książek**



Rys. . Ekran do stworzenia książek

**CreateFragment** to fragment odpowiedzialny za zarządzanie książkami użytkownika w aplikacji

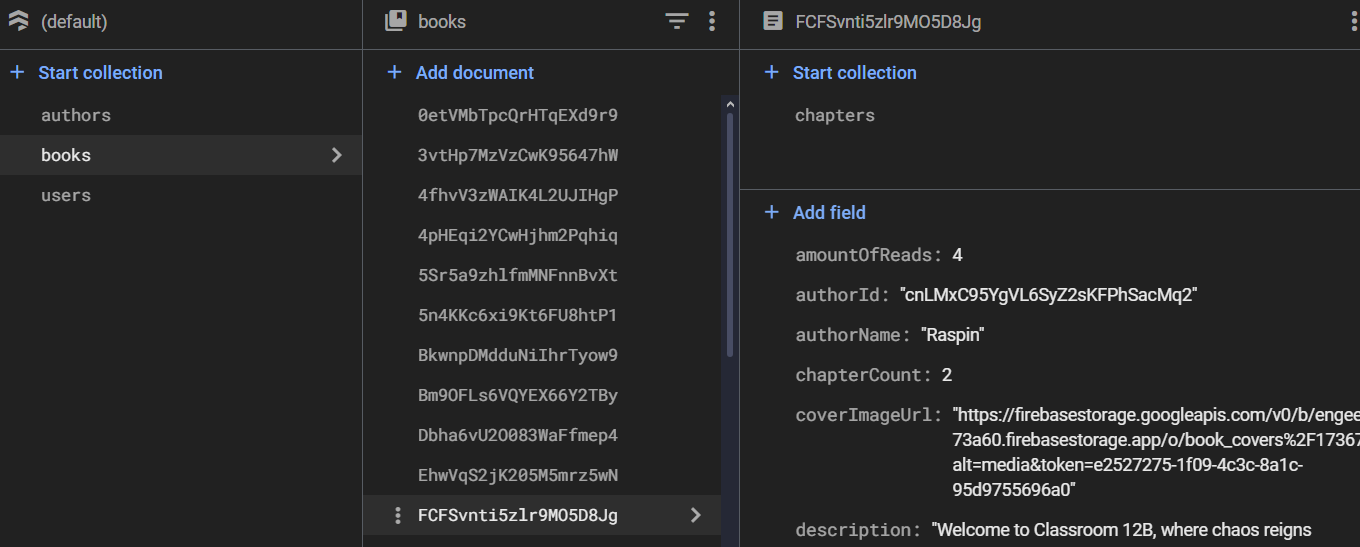
Na ekranie tego fragmentu dostępne są następujące funkcjonalności:

1. **Wyświetlanie listy książek użytkownika**
2. **Dodanie rozdziału do książki**
3. **Edycja książki**
4. **Oznaczenie książki jako ukończonej**
5. **Tworzenie nowej książk**

Po naciśnięciu odpowiednich przycisków użytkownik zostanie przekierowany do odpowiednich ekranów aplikacji

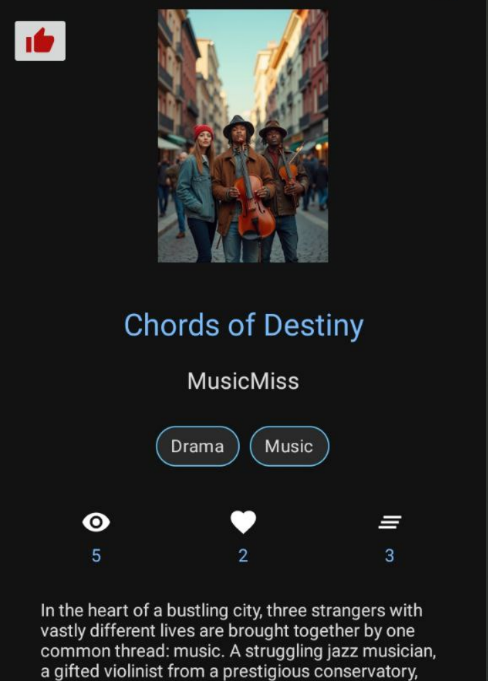
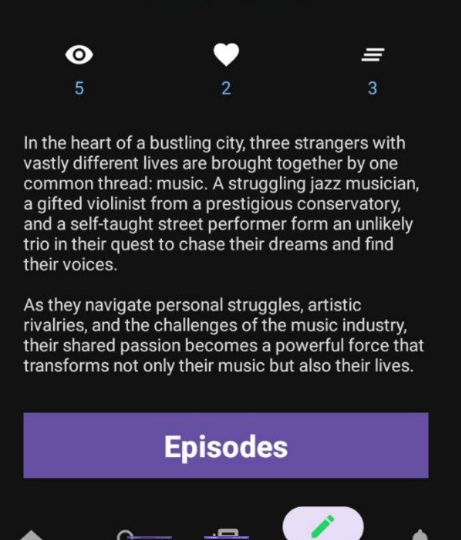
Kliknięcie na książkę w **RecyclerView** automatycznie przenosi użytkownika do ekranu szczegółów wybranej książki. Funkcja ta jest zaimplementowana we wszystkich **RecyclerView**, które wyświetlają książki.

Wszystkie dane są pobierane z bazy danych **Firestore Cloud**



Rys. 26. Baza danych Firestore Cloud

**Ekran szczegóły książki**

****

Rys. 27. Ekran szczegół książki cz 2

Rys. 28. Ekran szczegół książki cz 1

**Ekran szczegóły książki** to fragment odpowiedzialny za wyświetlanie szczegółów wybranej książki. Umożliwia użytkownikowi zapoznanie się z informacjami o książce oraz wykonanie określonych akcji, takich jak polubienie książki czy przejście do listy rozdziałów

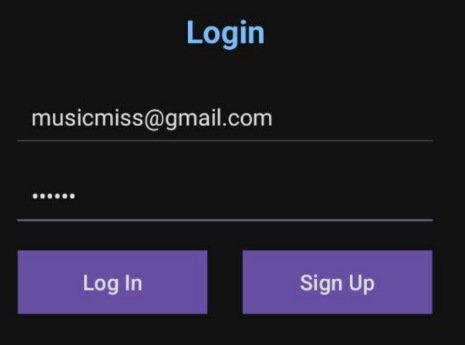
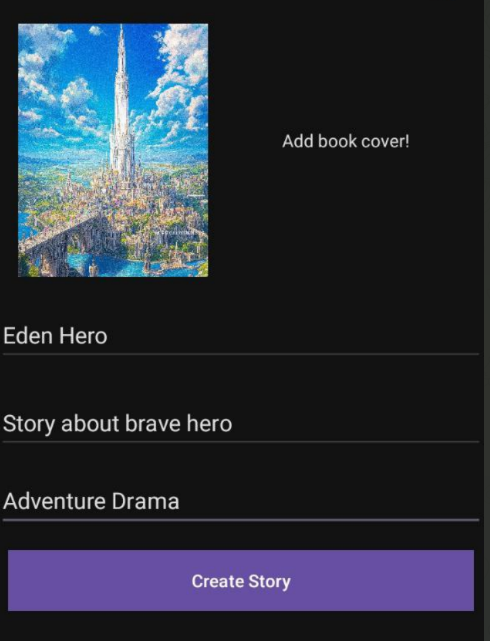
Po naciśnięciu na nazwę autora użytkownik zostanie przeniesiony na stronę profilu autora.

Przycisk **"Episodes"** pozwala przejść do listy rozdziałów wybranej książki.

Użytkownik może polubić lub cofnąć polubienie książki. Polubienia są synchronizowane z bazą danych Firestore, a ich liczba jest automatycznie aktualizowana.

Każda odwiedzona książka jest zapisywana w kolekcji historii użytkownika w Firestore. Każdorazowe wyświetlenie szczegółów książki zwiększa licznik jej wyświetleń.

# **Prezentację opracowanego projektu**



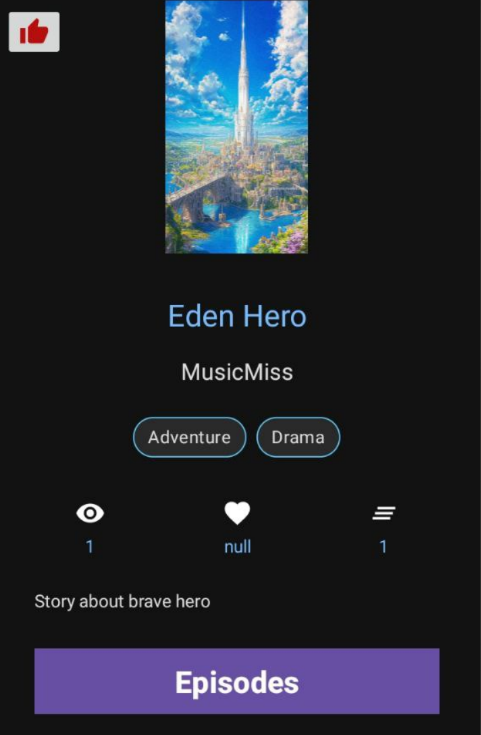
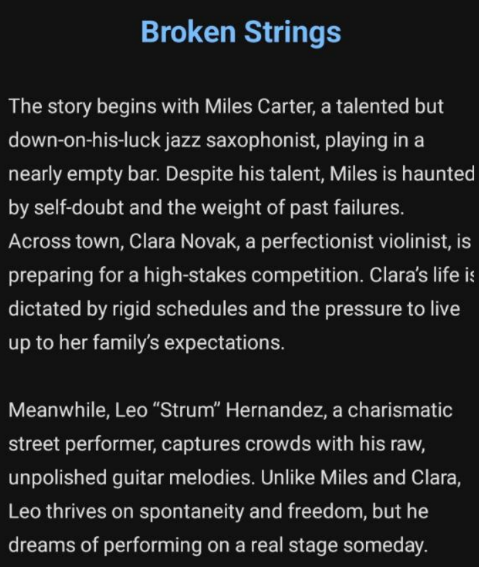
Rys. . Tworzenia ksiązki

Rys. . Logowanie jako użytkownik

rys …

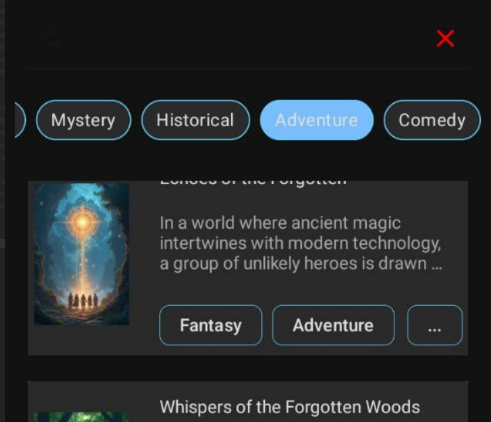
rys …

rys …

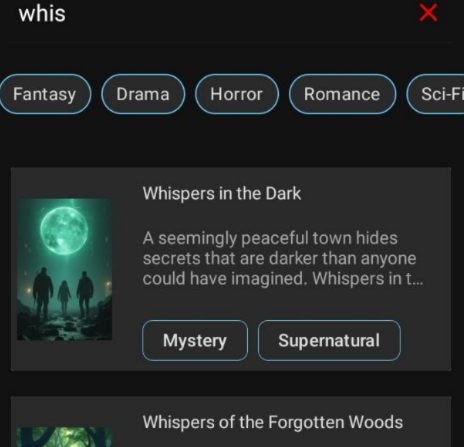


Rys. . Tekst rozdilu

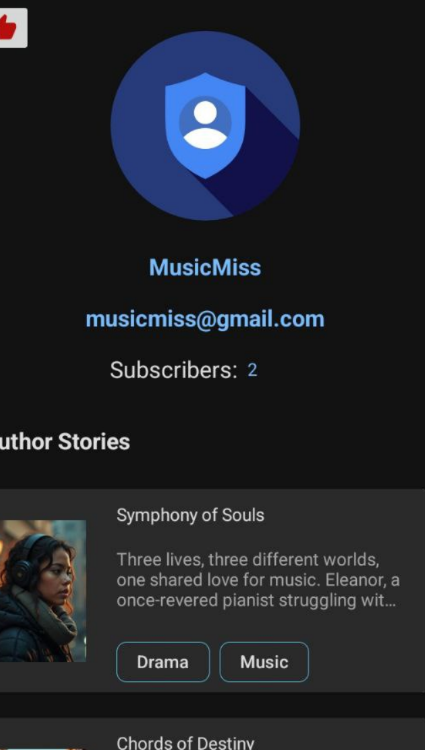
Rys. . Szczegóły książki



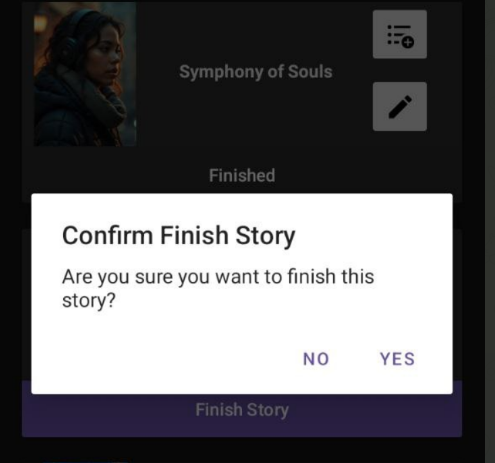
Rys. . Wyszukiwanie za pomocą ChipGroup



Rys. . Wyszukiwanie książki za nazwą



Rys. . Ekran autora



Rys. . Zakończenia książki

rys.36

# **Wnioski i zakończenia**

Analiza literatury oraz praca nad projektem stanowią podstawę do sformułowania następujących wniosków:

1. **Bezpieczeństwo i łatwość obsługi**Firebase Authentication zapewnia szybkie i bezpieczne logowanie oraz rejestrację użytkowników w aplikacji
2. **Zaawansowane wyszukiwanie książek**Wykorzystanie ChipGroup znacznie usprawnia proces wyszukiwania książek według gatunku. Jednak implementacja automatycznego generowania chipów mogłaby dodatkowo zwiększyć elastyczność i użyteczność tej funkcji.
3. **Personalizacja treści**Funkcja subskrypcji autorów oraz możliwość dodawania książek do osobistej biblioteki umożliwiają użytkownikom szybki dostęp do ulubionych treści i personalizację aplikacji zgodnie z ich preferencjami.
4. **Elastyczność w tworzeniu treści**Funkcja tworzenia i edytowania książek, w tym dodawania rozdziałów, w pełni odpowiada na potrzeby użytkowników, którzy chcą tworzyć i publikować swoje książki.
5. **Inspiracja istniejącymi rozwiązaniami**Aplikacje takie jak Wattpad oferują użytkownikom zarówno możliwość pisania, jak i czytania książek w jednym miejscu. Brak podobnych rozwiązań na rynku zainspirował projekt do stworzenia aplikacji o podobnej funkcjonalności

**Zakończenia**

Projekt aplikacji mobilnej do tworzenia i zarządzania książkami osiągnął swoje główne cele, oferując funkcjonalności umożliwiające użytkownikom tworzenie, edytowanie i publikowanie treści. Aplikacja wspiera interakcje między twórcami a czytelnikami dzięki funkcjom subskrypcji i personalizacji.

**Ograniczenia projektu:**

* Aplikacja wymaga stałego połączenia z Internetem

**Mocne strony:**

* Intuicyjny interfejs użytkownika oparty na ciemnej kolorystyce.
* Wygodny funkcjonal do wyszukiwania książek oraz ich za ich śledzeniem

**Słabe strony:**

* Brak trybu offline, co ogranicza użytkowanie w niektórych warunkach.
* Brak możliwości otrzymać wiadomość o nowych rozdziałach oraz książkach

# **Literatura**

**Książki:**

1. Phillips J., Stewart B.: *Kotlin Programming: The Big Nerd Ranch Guide*. Big Nerd Ranch, Atlanta 2019.
2. Suvorov V., Ponomarenko A.: *Kotlin In-Depth: A Guide to a Multipurpose Programming Language*. Independently Published, 2020.
3. Mertens D., Vermeulen B.: *Java to Kotlin: A Refactoring Guidebook*. O'Reilly Media, Sebastopol 2020.
4. Nalini S., Kumar A.: *Android Development with Kotlin*. Packt Publishing, Birmingham 2018.
5. Ustheide E.: *Kotlin for Android Developers: Learn Kotlin the Easy Way While Developing Android Apps*. Leanpub, 2017.
6. González J., Martín E.: *Building Android Apps with Kotlin: Learn How to Develop Android Applications Using Kotlin*. Apress, New York 2020.
7. Wójcik P.: *Android. Programowanie aplikacji mobilnych*. Helion, Gliwice 2017.
8. Górski M.: *Kotlin. Szybki start*. PWN, Warszawa 2020.
9. Ciołek K., Kaczmarek P.: *Android 10. Programowanie aplikacji mobilnych*. PWN, Warszawa 2020.
10. Ławniczak M.: *Kotlin. Programowanie na Androida*. Helion, Gliwice 2018.

**Źródła internetowe**

1. GeeksforGeeks:

[*https://www.geeksforgeeks.org/kotlin-programming-language/*](https://www.geeksforgeeks.org/kotlin-programming-language/)

1. Stack Overflow: [*https://stackoverflow.com/questions/tagged/kotlin*](https://stackoverflow.com/questions/tagged/kotlin)
2. Google Developers: [*https://developer.android.com/*](https://developer.android.com/)
3. YouTube, All UML Diagrams in 10 minutes. Dostep: <https://www.youtube.com/watch?v=hF4yg1yFrdU>
4. YouTube, Android Kurs Tworzenia Aplikacji [2022/2023] #0

<https://www.youtube.com/watch?v=8u7UP4C7OcQ&list=PL7i9G1Htb_PvQhkckHBhaOH3ft5lTNGun>

# **Spis rysunków**

[Rys. 1. Szkic ekranów cz 2 7](#_Toc188187069)

[Rys. 2. Szkic ekranów cz 1 7](#_Toc188187070)

[Rys. 3. Diagram UML 8](#_Toc188187071)

[Rys. 4. Konfiguracja projekta 9](#_Toc188187072)

[Rys. 5. Dodanie kolorów 9](#_Toc188187073)

[Rys. 6. Dodanie grafiki 9](#_Toc188187074)

[Rys. 7. Dodanie bibliotek 10](#_Toc188187075)

[Rys. 8. Dodane plaginy 10](#_Toc188187076)

[Rys. 9. Ekran logowania 12](#_Toc188187077)

[Rys. 10. Logika logowania 12](#_Toc188187078)

[Rys. 11. Przeniesienie do ekranu refestracji 12](#_Toc188187079)

[Rys. 12. Ekran rejestrowania 13](#_Toc188187080)

[Rys. 13. Logika rejestrowania 13](#_Toc188187081)

[Rys. 14. Ekran Home 14](#_Toc188187082)

[Rys. 15. Sprawdzenia czy użytkownik jest zalogowany 14](#_Toc188187083)

[Rys. 16. loadBooksForFinishedStories()w 14](#_Toc188187084)

[Rys. 17. loadBooksWeThinkYouWillEnjoy() 15](#_Toc188187085)

[Rys. 18. loadBooksTheMostLikes() 15](#_Toc188187086)

[Rys. 19. loadBooksForTopPicks() 15](#_Toc188187087)

[Rys. 20. Ekran wyszukiwania 16](#_Toc188187088)

[Rys. 21. Konfiguracja SearchView 16](#_Toc188187089)

[Rys. 22. Filtrowanie książek za nazwą 17](#_Toc188187090)

[Rys. 23. Konfiguracja ChipGroup 17](#_Toc188187091)

[Rys. 24. Filtrowanie ksiązek za gatunkem 17](#_Toc188187092)

[Rys. 25. Ekran do stworzenia książek 18](#_Toc188187093)

[Rys. 26. Baza danych Firestore Cloud 18](#_Toc188187094)

[Rys. 27. Ekran szczegół książki cz 2 19](#_Toc188187095)

[Rys. 28. Ekran szczegół książki cz 1 19](#_Toc188187096)

[Rys. 29. Tworzenia ksiązki 20](#_Toc188187097)

[Rys. 30. Logowanie jako użytkownik 20](#_Toc188187098)

[Rys. 31. Tekst rozdilu 20](#_Toc188187099)

[Rys. 32. Szczegóły książki 20](#_Toc188187100)

[Rys. 33. Wyszukiwanie za pomocą ChipGroup 20](#_Toc188187101)

[Rys. 34. Wyszukiwanie książki za nazwą 20](#_Toc188187102)

[Rys. 35. Ekran autora 20](#_Toc188187103)

[Rys. 36. Zakończenia książki 20](#_Toc188187104)

# **Streszczenie**

**Streszczenie**

Celem projektu jest zaprojektowanie i implementacja aplikacji mobilnej przeznaczonej do pisania i czytania książek innych użytkowników. Aplikacja oferuje funkcjonalności takie jak logowanie, rejestracja użytkowników, tworzenie i edytowanie książek, dodawanie rozdziałów, czytanie książek, wyszukiwanie oraz subskrypcję autorów. Dodatkowo umożliwia użytkownikom dodawanie książek do własnej biblioteki oraz śledzenie historii przeglądanych książek.

Do realizacji projektu wybrano język Kotlin, framework Android Jetpack oraz bazę danych Firestore Cloud. Narzędzia takie jak Firebase Authentication i Firebase Storage zostały wykorzystane do zarządzania użytkownikami i przechowywania danych. Projekt opiera się na modelowaniu UML, co ułatwia planowanie struktury aplikacji przed jej implementacją.

Aplikacja charakteryzuje się intuicyjnym interfejsem użytkownika w ciemnej kolorystyce, zapewniając wygodę korzystania. Projekt osiągnął pełną funkcjonalność, umożliwiając łatwe tworzenie i zarządzanie książkami. Ograniczeniem projektu jest brak trybu offline oraz brak powiadomień o nowych rozdziałach i książkach. Mimo tych niedogodności projekt stanowi wartościowe rozwiązanie, pozwalające użytkownikom na tworzenie, publikowanie i zarządzanie treściami w mobilnej aplikacji.

**English Summary**

The goal of the project is to design and implement a mobile application for writing and reading books by other users. The application offers features such as login, user registration, book creation and editing, adding chapters, reading books, searching, and subscribing to authors. Additionally, it allows users to add books to their personal library and track their reading history.

For the implementation of the project, the Kotlin programming language, the Android Jetpack framework, and the Firestore Cloud database were selected. Tools such as Firebase Authentication and Firebase Storage were used for user management and data storage. The project is based on UML modeling, which facilitates the planning of the application's structure before implementation.

The application features an intuitive user interface in a dark color scheme, ensuring a comfortable user experience. The project achieves full functionality, allowing users to easily create and manage books. The limitations of the project include the lack of offline mode and the absence of notifications for new chapters and books. Despite these limitations, the project offers a valuable solution, enabling users to create, publish, and manage content within the mobile application.