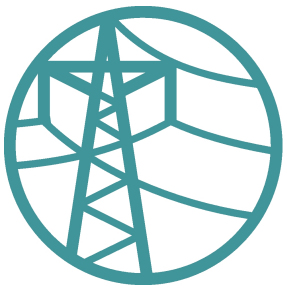


WYDZIAŁ
ELEKTROTECHNIKI
I INFORMATYKI
POLITECHNIKI RZESZOWSKIEJ

Rafał Stępień

Książkowy dziennik w chmurze

Praca dyplomowa inżynierska

Opiekun pracy:
dr inż. Mariusz Mączka

Rzeszów, 2025

Spis treści

1. Wstęp

W dobie rosnącej cyfryzacji i dynamicznego rozwoju technologii mobilnych, coraz więcej osób poszukuje nowoczesnych narzędzi wspierających codzienne czynności, w tym także zarządzanie swoimi pasjami i zainteresowaniami. Jedną z takich pasji, cieszącą się niezmienną popularnością, jest czytanie książek. Wraz z rosnącą liczbą dostępnych tytułów zarządzanie osobistą biblioteką może stać się problematyczne.

Niniejsza praca dotyczy zaprojektowania i implementacji aplikacji mobilnej o nazwie „BookTracker", stworzonej z wykorzystaniem technologii Jetpack Compose oraz zintegrowanej z internetową bazą danych Supabase. Aplikacja oferuje użytkownikom możliwość oznaczania książek jako posiadanych lub przeczytanych, śledzenia nadchodzących premier literackich oraz zarządzania osobistą biblioteką w sposób przejrzysty i dostępny z każdego miejsca, dzięki wykorzystaniu technologii chmurowych. Tematyka ta została wybrana ze względu na rosnącą popularność aplikacji wspierających organizację życia codziennego oraz potrzebę zapewnienia użytkownikom narzędzi umożliwiających wygodne i efektywne zarządzanie ich kolekcjami literackimi.

Zakres pracy obejmuje zaprojektowanie kluczowych funkcji aplikacji, implementację interfejsu użytkownika oraz integrację z bazą danych w chmurze. Głównym celem pracy jest stworzenie aplikacji mobilnej, która w przejrzysty sposób umożliwi użytkownikom zarządzanie osobistą biblioteką książek z wykorzystaniem technologii chmurowych.

2. Podobne aplikacje

W dzisiejszych czasach niemal każda aplikacja ma już swoje odpowiedniki na rynku. Podczas tworzenia aplikacji postawiono więc na skupienie się na grupie docelowej użytkowników, którzy oczekują konkretnych rozwiązań.

2.1. Bookmory

Bookmory to przykład wielu istniejących aplikacji dzięki którym można zarządzać swoją biblioteką, do jej odpowiedników można zaliczyć jeszcze kilka aplikacji takich jak: StoryGraph, Bookshelf, czy Bookly. Każda z wymienionych aplikacji spełnia podobne zadania, a dla większości użytkowników wybór pomiędzy nimi nie ma większego znaczenia, ponieważ różnice między nimi są minimalne.

Istnienie tych aplikacji zainspirowało stworzenie alternatywy, która będzie różniła się od swoich poprzedników unikalną funkcjonalnością i skupieniem się na konkretnej grupie odbiorców.

2.2. TV Time

TV Time to popularna aplikacja służąca do śledzenia postępu oglądania seriali i filmów. Pomimo tego, że aplikacja ta nie ma możliwości zarządzania książkami i skupia się wyłącznie na produkcjach filmowych, to funkcjonalność śledzenia seriali stała się inspiracją do stworzenia alternatywnej aplikacji do śledzenia postępu czytania książek. Główną różnicą do istniejących już rozwiązań jest skupienie się na seriach książek wydawanych w tomach i ułatwieniu użytkownikom obserwowania i zarządzania książkami, które są ze sobą powiązane.

3. Środowiska i technologie

Wybór odpowiednich środowisk i technologii jest kluczowym elementem każdego projektu informatycznego, ponieważ wpływa zarówno na efektywność pracy, jak i na jakość końcowego rozwiązania. Ważnym jest przedstawienie wybranych rozwiązań i uzasadnienie dlaczego były one wybrane zamiast alternatywnych opcji.

3.1. Android Studio

Android Studio to oficjalne zintegrowane środowisko programistyczne (IDE) do tworzenia aplikacji na system Android bazujące na IntelliJ IDEA firmy JetBrains, stworzone i rozwijane przez firmę Google. Jest to kompleksowe narzędzie, które oferuje zestaw funkcji, mających na celu ułatwienie tworzenia, debugowania i publikowania aplikacji.

Środowisko to jest nieustannie aktualizowane przez Google, aby wspierać najnowsze wersje systemu Android oraz dodawać nowe funkcje. Android Studio obsługuje programowanie w językach takich jak Java, Kotlin (rekomendowany przez Google), a także w mniejszym stopniu w C++.

Kluczowym elementem jest edytor kodu, który obsługuje autouzupełnianie, refaktoryzację oraz podpowiedzi kontekstowe, co znacząco ułatwia pisanie czystego i efektywnego kodu.

Android Studio zostało wybrane ze względu na to, że jest najpełniejszym narzędziem do tworzenia aplikacji na system android i w przeciwieństwie do np. Visual Studio Code, można praktycznie od razu rozpocząć pracę zamiast zajmowania się instalowaniem rozszerzeń i potrzebnych komponentów. Android Studio oferuje wszystkie niezbędne funkcje w jednym pakiecie.

3.2. Kotlin

Kotlin to nowoczesny, statycznie typowany język programowania, który został stworzony przez firmę JetBrains i jest oficjalnie wspierany przez Google do tworzenia natywnych aplikacji na system Android. Kotlin jest zaprojektowany z myślą o prostocie, bezpieczeństwie i interoperacyjności z Javą - język ten jest w pełni kompatybilny z istniejącym ekosystemem Javy, co umożliwia łatwą integrację z istniejącym kodem i bibliotekami.

Kotlin oferuje wiele nowoczesnych funkcji, które czynią go atrakcyjnym wyborem. Jego kluczowe cechy to m.in. zwięzła składnia, która pozwala na pisanie czytelnego i mniej podatnego na błędy kodu, oraz zaawansowane mechanizmy, takie jak obsługa funkcji wyższych rzędów, rozszerzenia klas czy programowanie funkcyjne. Wbudowane mechanizmy bezpieczeństwa, takie jak system typów zapobiegający błędom typu null pointer exception (tzw. "null safety"), znacząco zwiększają niezawodność aplikacji.

Kotlin wspiera także współbieżność dzięki korutynom, które są lekkim mechanizmem współbieżności umożliwiającym pisanie asynchronicznego kodu w czytelnym i intuicyjnym stylu. Korutyny działają w ramach istniejących wątków, wykorzystując mechanizmy wstrzymywania i wznawiania, co pozwala na efektywne zarządzanie zasobami systemowymi bez potrzeby blokowania wątków. To podejście znacząco upraszcza tworzenie wydajnych aplikacji, zwłaszcza tych, które intensywnie korzystają z asynchronicznej komunikacji sieciowej, przetwarzania dużych ilości danych czy operacji wejścia/wyjścia.

3.3. Jetpack Compose

Jetpack Compose to nowoczesny framework interfejsu użytkownika stworzony przez Google, który pozwala na tworzenie aplikacji na Androida w sposób deklaratywny. Zamiast używać tradycyjnych plików XML do definiowania widoków, Jetpack Compose umożliwia definiowanie interfejsu w kodzie Kotlin, co prowadzi do tworzenia prostszych, bardziej zwięzłych i łatwiejszych w utrzymaniu aplikacji.

Jedną z największych zalet Jetpack Compose jest możliwość dynamicznego reagowania na zmiany stanu aplikacji. Dzięki podejściu opartemu na deklaratywnej reaktywności, widoki automatycznie aktualizują się w odpowiedzi na zmiany danych, co eliminuje konieczność ręcznego zarządzania aktualizacjami interfejsu użytkownika.

W Jetpack Compose funkcje komponowalne (ang. composable functions) stanowią podstawę tworzenia interfejsu użytkownika. Są to specjalne funkcje oznaczone adnotacją @Composable, które pozwalają na deklaratywne definiowanie i łączenie elementów UI. Funkcje komponowalne w Jetpack Compose działają na zasadzie deklaratywnego określania struktury i zawartości interfejsu użytkownika, zamiast szczegółowego opisywania sposobu jego wyświetlania. Co umożliwia koncentrację na logice aplikacji, a nie na szczegółach implementacji interfejsu. Każda funkcja komponowalna może zawierać inne funkcje komponowalne, tworząc w ten sposób złożone i hierarchiczne struktury

UI w sposób naturalny i łatwy do zrozumienia. Takie podejście umożliwia tworzeniee intuicyjnych, skalowalnych i łatwych w utrzymaniu aplikacji.

Jetpack Compose został wybrany, ponieważ jest rozwiązaniem w pełni zintegrowanym z Androidem, co zapewnia najlepszą optymalizację i wydajność w tworzeniu aplikacji natywnych na tę platformę.

3.4. Supabase

Supabase to nowoczesna platforma typu Backend-as-a-Service (backend jako usługa), która umożliwia tworzenie aplikacji z wykorzystaniem bazy danych PostgreSQL. Jest to narzędzie do budowy aplikacji, bez potrzeby zarządzania infrastrukturą serwerową.

Kluczowym elementem Supabase jest integracja z PostgreSQL, która umożliwia dostęp do bazy danych, oferującej funkcje takie jak zaawansowane zapytania SQL, funkcje typu trigger oraz bezpieczeństwo na poziomie wiersza (RLS). Supabase automatycznie tworzy RESTful API na podstawie tabel w bazie danych, co pozwala na szybkie wdrażanie aplikacji.

Supabase oferuje również intuicyjne narzędzia do zarządzania bazą danych, które umożliwiają łatwe projektowanie i modyfikowanie struktury tabel oraz relacji między nimi. Dzięki wbudowanemu panelowi administracyjnemu, użytkownicy mogą przeglądać dane, zarządzać użytkownikami oraz monitorować aktywność w bazie danych, co znacząco upraszcza proces rozwoju aplikacji.

3.5. GitHub

GitHub to powszechnie używana platforma do zarządzania kodem źródłowym i współpracy w zespołach programistycznych, oparta na systemie kontroli wersji Git. Umożliwia śledzenie zmian w kodzie, zarządzanie historią projektu oraz łatwą współpracę wielu programistów nad jednym projektem.

3.6. Cloudinary

Cloudinary to platforma do zarządzania multimediami w chmurze, która umożliwia przechowywanie, optymalizację i dostarczanie obrazów, wideo oraz innych plików multimedialnych. Dzięki zaawansowanym funkcjom, takim jak automatyczna optyma-

lizacja, zmiana rozmiaru, kadrowanie i konwersja formatów, Cloudinary pozwala na łatwe dostosowanie zasobów do różnych urządzeń i platform.

4. Architektura bazy danych Supabase

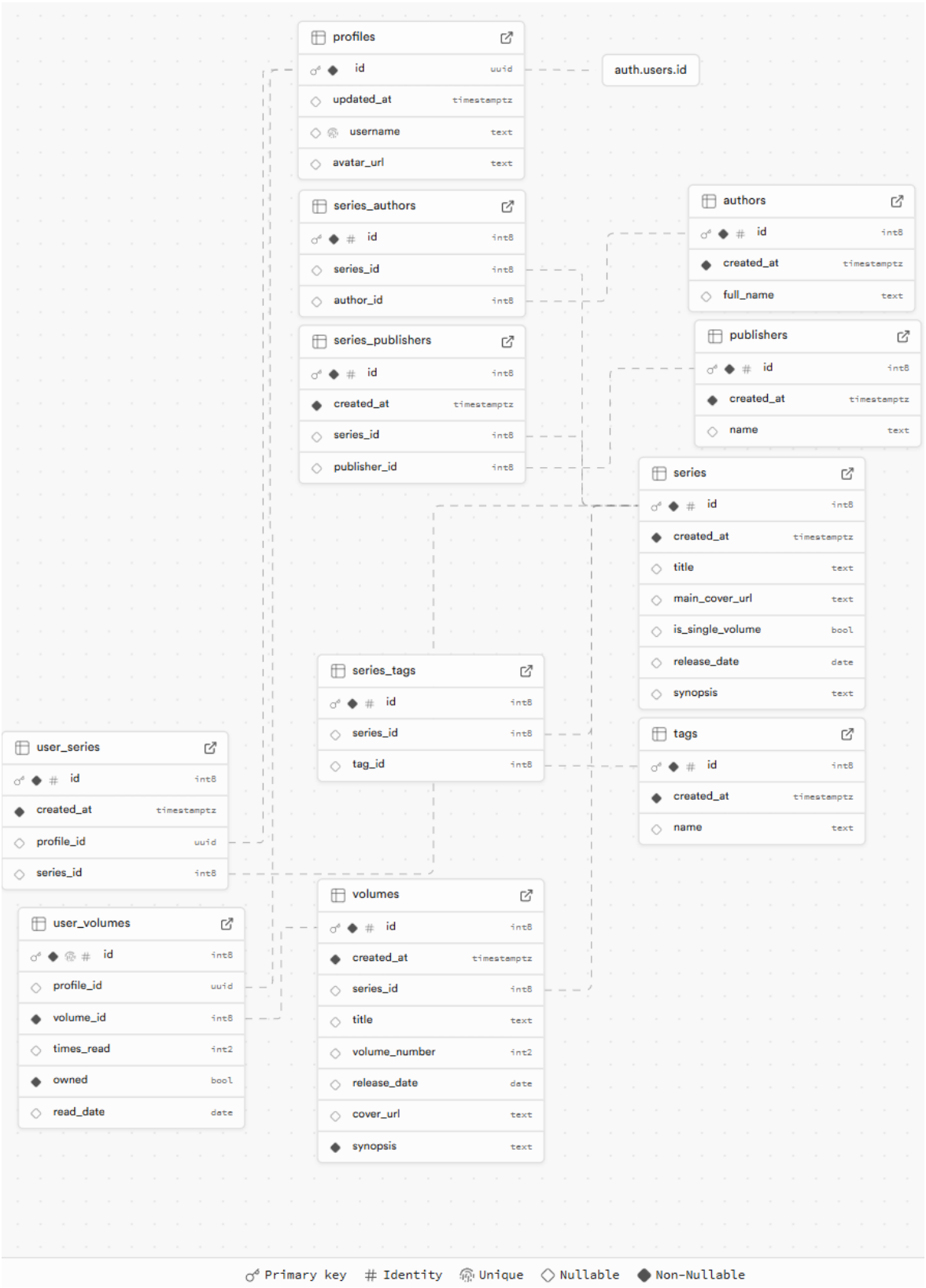
W tym rozdziale przedstawiona zostanie struktura bazy danych PostgreSQL i przykładowe funkcje wykorzystywane do pobierania danych z bazy.

4.1. Schemat bazy danych

Schemat Public bazy danych utworzony przez autora składa się z 11 tabel. Tabela profiles zawiera klucz główny public.profiles.id, który odpowiada kluczowi głównemu auth.users.id w tabeli users w schemacie auth utworzonym automatycznie przez Supabase odpowiedzialnym za autentykację użytkownika.

Za główną tabelę bazy danych można określić tabelę series, w której znajdują się informacje o seriach książek. Seria książek w kontekście tego rozwiazania to zbiór powiązanych ze sobą tematycznie lub fabularnie tomów, które stanowią część większego cyklu. Tabela ta zawiera atrybuty opisujące serię, takie jak tytuł, URL okładki czy streszczenie, a także pola wykorzystywane w logice, takie jak is_single_volume, które pozwalają określić, czy dana pozycja stanowi pełnoprawną serię, czy jest to pojedyncza książka, niezwiązana z żadnym cyklem.

W przypadku sytuacji „wiele do wielu” na przykładzie tabeli user_series, rozwiązanie polega na przechowywaniu relacji między użytkownikami a ich obserwowanymi seriami książek. Tabela ta zawiera pola takie jak profile_id i series_id, które umożliwiają przypisanie użytkownika do jednej lub wielu serii książek. Dzięki temu jeden użytkownik może obserwować wiele serii, a jedna seria może być obserwowana przez wielu użytkowników. Więcej bardziej szczegółowych informacji na temat bazy danych jest ukazane na diagramie ERD wygenerowanym w Supabase na rysunku 4.1.



Rysunek 4.1: Schemat bazy danych

4.1.1. Normalizacja

Normalizacja bazy danych to proces organizowania danych w taki sposób, aby zminimalizować redundancję i zapewnić integralność danych. Celem jest poprawienie struktury bazy w sposób, który ułatwia jej zarządzanie i utrzymanie, eliminując potencjalne problemy związane z nieefektywnym przechowywaniem informacji.

a) pierwsza postać normalna (1NF) na podstawie kodu z listingu 1

```
    CREATE TABLE public.series (
      id bigint GENERATED BY DEFAULT AS IDENTITY NOT NULL,
      title text NULL,
      ...
      CONSTRAINT Series_pkey PRIMARY KEY (id)
      );

```

Listing 1: kod tworzenia tabeli series

Tabela series spełnia pierwszą postać normalną, ponieważ kolumny takie jak title zawierają atomowe wartości, a klucz główny (id) zapewnia unikalność każdego rekordu,

b) druga postać normalna (2NF) na podstawie kodu z listingu 2

```
    CREATE TABLE public.volumes (
      id bigint GENERATED BY DEFAULT AS IDENTITY NOT NULL,
      series_id bigint NULL,
      title text NULL,
      ...
      CONSTRAINT volumes_pkey PRIMARY KEY (id),
      CONSTRAINT volumes_series_id_fkey FOREIGN KEY (
  series_id) REFERENCES series(id)
    );

```

Listing 2: kod tworzenia tabeli volumes

Tabela volumes spełnia drugą postać normalną, ponieważ jest w pierwszej postaci normalnej oraz wszystkie kolumny, które nie są częścią klucza głównego, są w pełni zależne od klucza głównego (id). Kolumna series_id jest kluczem obcym, a kolumny jak title zależą od id, co eliminuje zależności częściowe,

c) trzecia postać normalna (3NF) na podstawie kodu z listingu 3

```
    CREATE TABLE public.user_series (
      id bigint GENERATED BY DEFAULT AS IDENTITY NOT NULL,
      profile_id uuid NULL,
      series_id bigint NULL,
```

```
    ...
    CONSTRAINT user_series_pkey PRIMARY KEY (id),
    CONSTRAINT user_series_profile_id_fkey FOREIGN KEY (profile_id) REFERENCES profiles(id),
    CONSTRAINT user_series_series_id_fkey FOREIGN KEY (series_id) REFERENCES series(id)
 );

```

Listing 3: kod tworzenia tabeli user_series

Tabela user_series spełnia trzecią postać normalną, ponieważ jest w drugiej postaci normalnej, a dodatkowo nie występują zależności przejściowe. Kolumny, które nie są częścią klucza głównego, zależą tylko od klucza głównego (id), a nie od innych kolumn, co zapewnia brak zbędnych zależności między danymi. Tabela ta spełnia również BCNF, ponieważ każda kolumna w niej zależy bezpośrednio od głównego identyfikatora (id),

d) czwarta postać normalna (4NF) na podstawie kodu z listingu 4

```
 CREATE TABLE public.series_authors (
 id bigint GENERATED BY DEFAULT AS IDENTITY NOT NULL,
 series_id bigint NULL,
 author_id bigint NULL,
 CONSTRAINT series_author_pkey PRIMARY KEY (id),
 CONSTRAINT series_author_author_id_fkey FOREIGN KEY (author_id) REFERENCES authors(id) ON UPDATE CASCADE ON DELETE CASCADE,
 CONSTRAINT series_author_series_id_fkey FOREIGN KEY (series_id) REFERENCES series(id) ON UPDATE CASCADE ON DELETE CASCADE
);

```

Listing 4: kod tworzenia tabeli series_authors

Tabela series_authors spełnia czwartą postać normalną, ponieważ nie zawiera zależności wielowartościowych. Relacja między series_id a author_id jest wyrażona w sposób atomowy, gdzie każda kombinacja serii i autora jest reprezentowana przez pojedynczy wiersz. Tabela nie wprowadza redundancji związanej z wieloma wartościami przypisanymi do jednej kolumny

4.2. Funkcje bazy danych

Funkcje bazy danych stanowią ważny element w zarządzaniu danymi, umożliwiając efektywne wykonywanie różnych operacji w obrębie samej bazy danych. Funkcje

PostgreSQL pozwalają na grupowanie złożonych zapytań, usprawnienie przetwarzania danych oraz zwiększenie wydajności operacji. Dzięki funkcjom bazy danych możliwe jest wykonywanie operacji specyficznych dla aplikacji bezpośrednio na poziomie bazy danych, co pozwala na zmniejszenie obciążenia aplikacji i serwera.

Kod przykładowej funkcji bazy danych znajduje się w listingu 5

```
1   CREATE OR REPLACE FUNCTION public.get_volume_by_id(
    p_volume_id bigint)
2   RETURNS TABLE(id bigint, title text, cover_url text,
    volume_number smallint, user_volume_id bigint, release_date
    timestamp with time zone, times_read smallint, owned boolean,
    synopsis text, read_date timestamp with time zone)
3   LANGUAGE plpgsql
4    AS $function$
5    BEGIN
6      RETURN QUERY
7      SELECT
8        v.id,
9        v.title,
10       v.cover_url,
11       v.volume_number,
12       uv.id AS user_volume_id,
13       v.release_date,
14       COALESCE(uv.times_read, 0) AS times_read,
15       COALESCE(uv.owned, false) AS owned,
16       v.synopsis,
17       uv.read_date
18     FROM
19       volumes v
20     LEFT JOIN
21       user_volumes uv ON v.id = uv.volume_id
22     WHERE
23       v.id = p_volume_id;
24   END;
25   $function$;
26
```

Listing 5: kod funkcji get_volume_by_id

funkcja ta pozwala na podstawie identyfikatora pobrać z bazy danych informacje o wybranym tomie książki (volume), z bazy danych. Funkcja zwraca wynik w postaci teabeli zawierającej dane o tomie oraz dodatkowe informacje związane z użytkownikiem jeżeli są dostępne, takie jak pobrane z tabeli user_volumes, times_read informujące o tym ile razy użytkownik przeczytał dany tom. Może wystąpić sytuacja, w której pobierany tom nie ma odpowiadającego rekordu w tabeli user_volumes, ponieważ użytkownik nie oznaczył go jako przeczytanego ani posiadanego. W takim przypadku dane użytkownika zostaną uzupełnione domyślnymi wartościami, takimi jak 0 dla liczby przeczytań czy false dla statusu posiadania, co jest zrealizowane z

wykorzystaniem funkcji COALESCE, której użycie można zaobserować w liniach 14 i 15.

5. Architektura i struktura aplikacji

Architektura aplikacji została oparta na wzorcu MVVM, co pozwala na efektywne zarządzanie stanem i separację logiki biznesowej od interfejsu użytkownika. Właściwa organizacja plików w projekcie jest kluczowa dla utrzymania porządku i skalowalności aplikacji, umożliwiając łatwe zarządzanie komponentami i ich zależnościami.

5.1. Struktura projektu

5.2. MVVM (Model-View-ViewModel)

Model-View-ViewModel to popularny wzorzec architektoniczny stosowany w tworzeniu aplikacji, który pomaga w organizacji kodu i rozdzieleniu odpowiedzialności pomiędzy różne warstwy aplikacji. Jego głównym celem jest zwiększenie modularności, łatwości testowania oraz oddzielenie logiki biznesowej od interfejsu użytkownika.

a) Model reprezentuje warstwwę danych i logiki biznesowej. Jest odpowiedzialny za zarządzanie danymi, które mogą pochodzić z różnych źródeł, takich jak bazy danych, API czy pliki lokalne. Model nie zawiera żadnej logiki związanej z interfejsem użytkownika ani sposobem prezentacji danych,

b) View (Widok) odpoawiada za warstwę prezentacji. Widok to interfejs użytkownika (UI), który jest odpowiedzialny za wyświetlenie danych i odbieranie interakcji od użytkownika. Widok powinien jedynie reagować na dane dostarczane przez ViewModel. W android widokiem są komponenty Jetpack Compose,

c) ViewModel to warstwa pomiędzy modelem a widokiem. pobiera dane z modelu i przekształca je w taki sposób, aby były gotowe do wyświetlenia w widoku. Ponadto zarządza stanem widoku (np. przechowywaniem stanu aplikacji w przypadku zmiany orientacji ekranu).

5.2.1. Model

W poniższym listingu

Literatura

POLITECHNIKA RZESZOWSKA im. I. Łukasiewicza Rzeszów, 2025
Wydział Elektrotechniki i Informatyki

STRESZCZENIE PRACY DYPLOMOWEJ INŻYNIERSKIEJ

KSIĄŻKOWY DZIENNIK W CHMURZE

Autor: Rafał Stępień, nr albumu: EF-169625
Opiekun: dr inż. Mariusz Mączka
Słowa kluczowe: (max. 5 słów kluczowych w 2 wierszach, oddzielanych przecinkami)

Treść streszczenia po polsku

RZESZOW UNIVERSITY OF TECHNOLOGY Rzeszow, 2025
Faculty of Electrical and Computer Engineering

BSC THESIS ABSTRACT

CLOUD-BASED BOOK JOURNAL

Author: Rafał Stępień, nr albumu: EF-169625
Supervisor: Dr. Eng. Mariusz Mączka
Key words: (max. 5 słów kluczowych w 2 wierszach, oddzielanych przecinkami)

Treść streszczenia po angielsku