Dokumentacja projektu BGM-Firebase Mobilne interfejsy multimedialne

Dawid Bitner

27 marca 2020

Politechnika Śląska
Wydział Matematyki Stosowanej

Rok akademicki 2019/2020

Spis treści

1 Narzędzia, skrypty, źródła zewnętrzne

Do realizacji projektu użyłem jako środowiska programistycznego *Android Studio* w wersji *3.5.1.*
Postanowiłem, że użyję frameworka do języka *Dart - Flutter'a* w wersji *v1.12.13+hotfix.8.*
Emulator symulował urządzenie *Nexus 5X*, na którym był zainstalowany system *Android* w wersji *8.1 API 27.*

Podczas realizacji projektu użyłem kilku zależności dodanych do projektu we Flutterze - opiszę je krótko na podstawie pliku *pubspec.yaml*

```
dependencies:
  flutter:
    sdk: flutter
  fluttertoast: ^4.0.0
  uuid: 2.0.4
  progress_dialog: ^1.2.2
  # The following adds the Cupertino Icons font to your application.
  # Use with the CupertinoIcons class for iOS style icons.
  cupertino_icons: ^0.1.2
  flutter_simple_dependency_injection: ^1.0.2
  bloc_pattern: ^2.5.1
  provider: ^2.0.1
  rxdart: ^0.23.1
  json_serializable: ^3.2.5
  retrofit: ^1.3.1
  firebase_auth: ^0.16.0
  firebase_database: ^3.1.5
  logger: ^0.8.3


dev_dependencies:
  flutter_test:
    sdk: flutter
  build_runner: ^1.7.4
  retrofit_generator: ^1.3.1+9
```

cupertino-icons - jest to repozytorium zasobów zawierające domyślny zestaw zasobów ikon używanych przez Flutter - wykorzystałem ikonę lupy do wyszukiwania miasta.

retrofit - wykorzystany w celu łatwego dostępu do API, bez pisania dużej ilości kodu, doskonale współgra z serializatorem i deserializatorem *JSON*.

flutter-simple-dependency-injection - dzięki temu mogłem wykorzystać wstrzykiwanie zależności w swoich programie.

json-serializable - wykorzystany w celu serializacji danych, ułatwia pisanie programu poprzez tworzenie plików, przy użyciu komendy na podstawie prostych zależności zostały stworzone pliki *g.dart* dzięki który można było pobierać dane z API, zarówno dla *Firebase* jak i dla zewnętrznego API do wyszukiwania filmów - w moim przypadku *OMDBApi* które to pobiera dane z bazy filmów *IMDB.com*. Przykład:

plik game.dart:

```
import 'package:firebase_database/firebase_database.dart';
import 'package:json_annotation/json_annotation.dart';

part 'game.g.dart';

@JsonSerializable(explicitToJson: true)
class Game{
  String title;
  String year;
  String genre;
  String studio;
  bool played;
  String id;

  Game({this.title, this.year, this.genre, this.studio, this.played,  this.id});

  void setUuid(String uuid){
    this.id = uuid;
  }

  factory Game.fromJson(Map<String, dynamic> json) => _$GameFromJson(json);
  Map<String, dynamic> toJson() => _$GameToJson(this);

}
```

Oraz stworzony na jego podstawie plik game.g.dart:

```
// GENERATED CODE - DO NOT MODIFY BY HAND

part of 'game.dart';

// **************************************************************************
// JsonSerializableGenerator
// **************************************************************************

Game _$GameFromJson(Map<String, dynamic> json) {
  return Game(
    title: json['title'] as String,
    year: json['year'] as String,
    genre: json['genre'] as String,
    studio: json['studio'] as String,
    played: json['played'] as bool,
    id: json['id'] as String,
  );
}

Map<String, dynamic> _$GameToJson(Game instance) => <String, dynamic>{
      'title': instance.title,
      'year': instance.year,
      'genre': instance.genre,
      'studio': instance.studio,
      'played': instance.played,
      'id': instance.id,
    };
```

bloc-pattern - w celu prostszego zastosowania logiki biznesowej w projekcie

rxdart - w celu wykorzystania programowania reaktywnego **fluttertoast** - dołączone standardowo - w celu używania wyskakujących powiadomień - *Toastów.*

intl - ten pakiet zapewnia funkcje internacjonalizacji i lokalizacji, w tym tłumaczenie wiadomości, liczbę mnogą i płcie, formatowanie i analizę dat / liczb oraz tekst dwukierunkowy. Dołączany standardowo.

uses-material-design - w celu używania *Google Design.*

firebase-auth i **firebase-database** - w celu połaczenia się z bazą *Firebase* i autoryzacji użytkownika *Firebase*

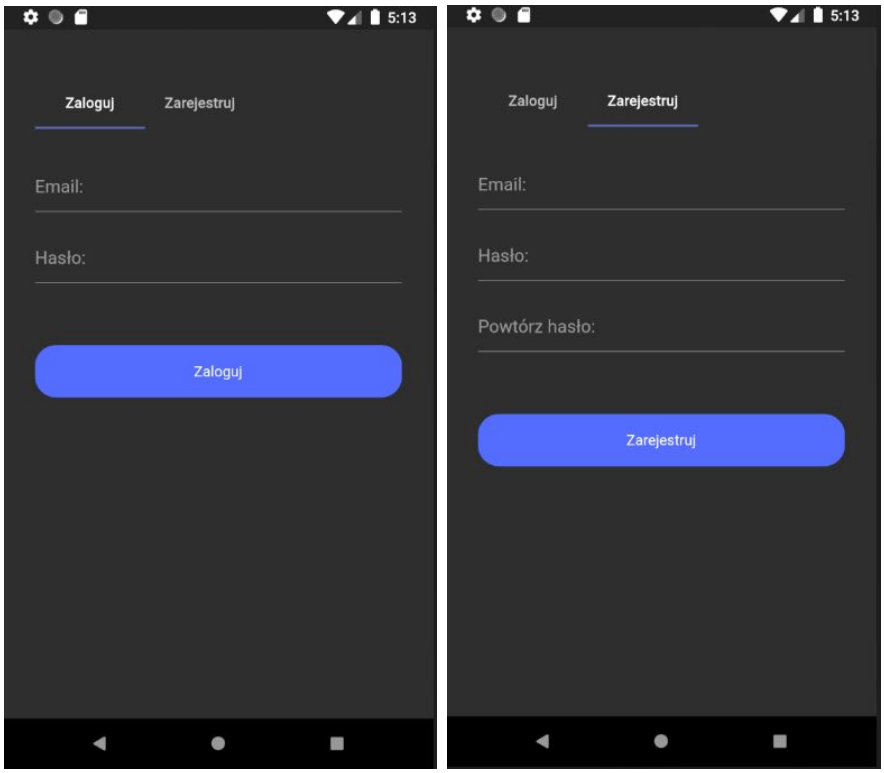
logger - w celu łatwiejszego i przyjemniejszego dostępu do logów w logcacie.

2 Problemy podczas realizacji

Nie miałem większych problemów podczas realizacji tego projektu we Flutterze. Dużą część warstwy widokowej aplikacji posiadałem już zaimplementowaną, ponieważ w poprzednim semestrze realizowałem z innymi osobami projekt na zaliczenie innego przedmiotu bardzo podobny do tego. Do głównych różnic należy zastosowanie *Firebase* jako bazy zdalnej, a nie jak w poprzednim projekcie *SQLite* jako bazy lokalnej, oraz zaimplementowanie modułu zdalnej autoryzacji. Największym problemem okazało się dla mnie dostosowanie wyglądu aplikacji - moim zdaniem stylizacja elementów we Flutterze wymaga dużego nakładu czasowego i jest mało intuicyjne. Posiłkowalem się gotowymi fragmentami kodu znalezionymi w serwisie *medium.com*, dokumentacji, oraz na *stackoverflow*.

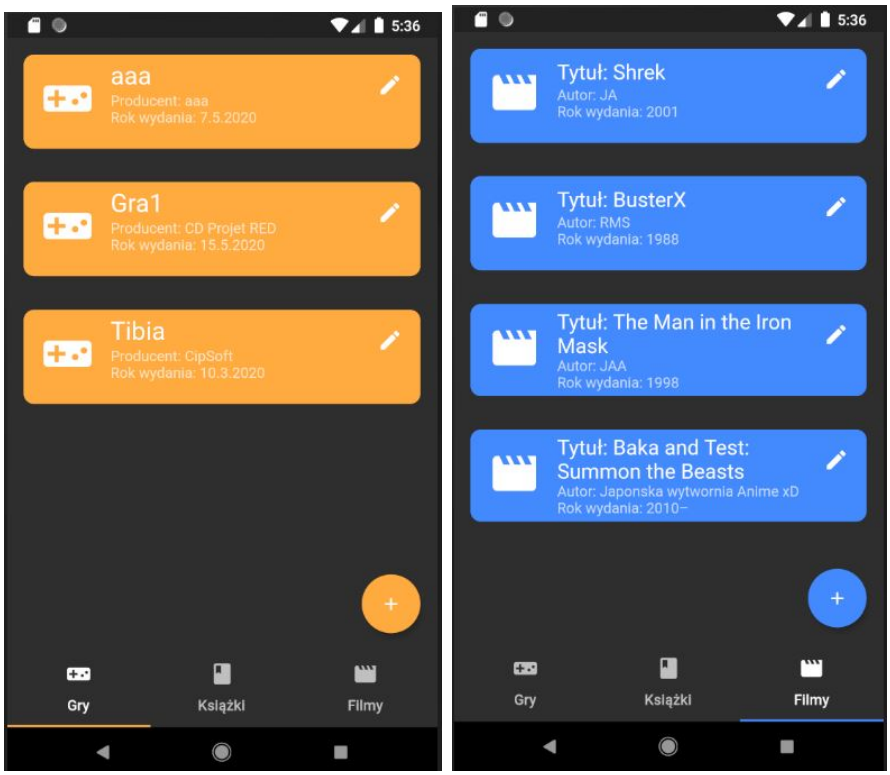
3 Główna część projektu

Jako stronę powitalną mamy do użytku element z dynamiczną zakładką, dzięki któremu możemy się zarejestrować w naszej bazie, bądź też zalogować na już istniejące konto.:



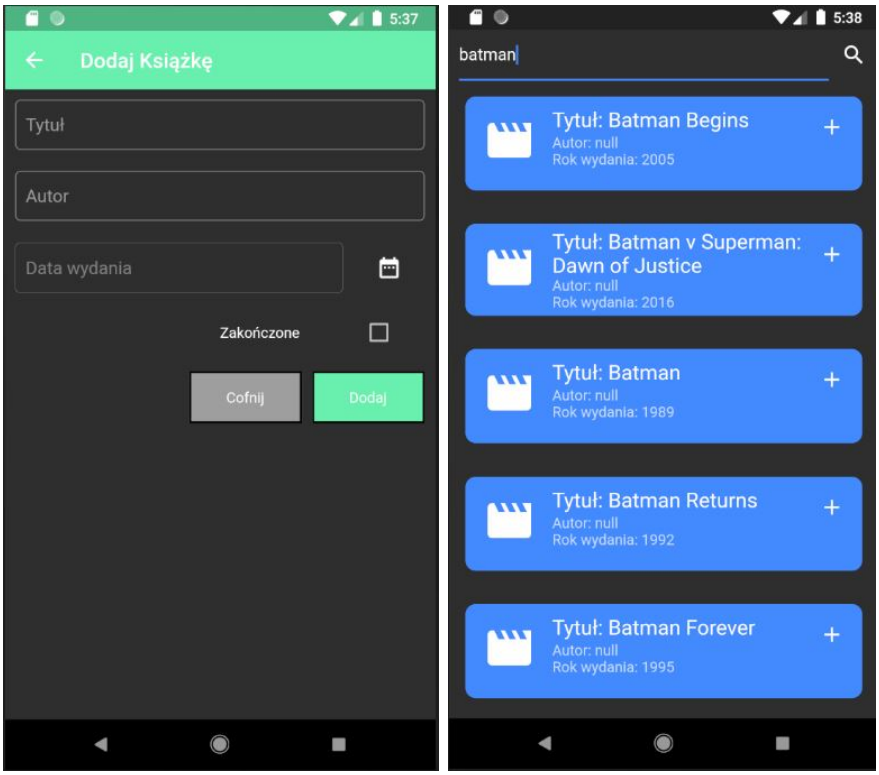
Zakładki logowania i rejestracji

Po zarejestrowaniu, gdzie sprawdzana jest poprawność adresu e-mail, długość hasła, oraz poprawność powtórzenia, zostaniemy przeniesieni do naszego pustego obszaru, analogicznie po zalogowaniu się już na istniejące konto zostaniemy przeniesieni do obszaru, gdzie znajdują się nasze gry, filmy oraz książki.



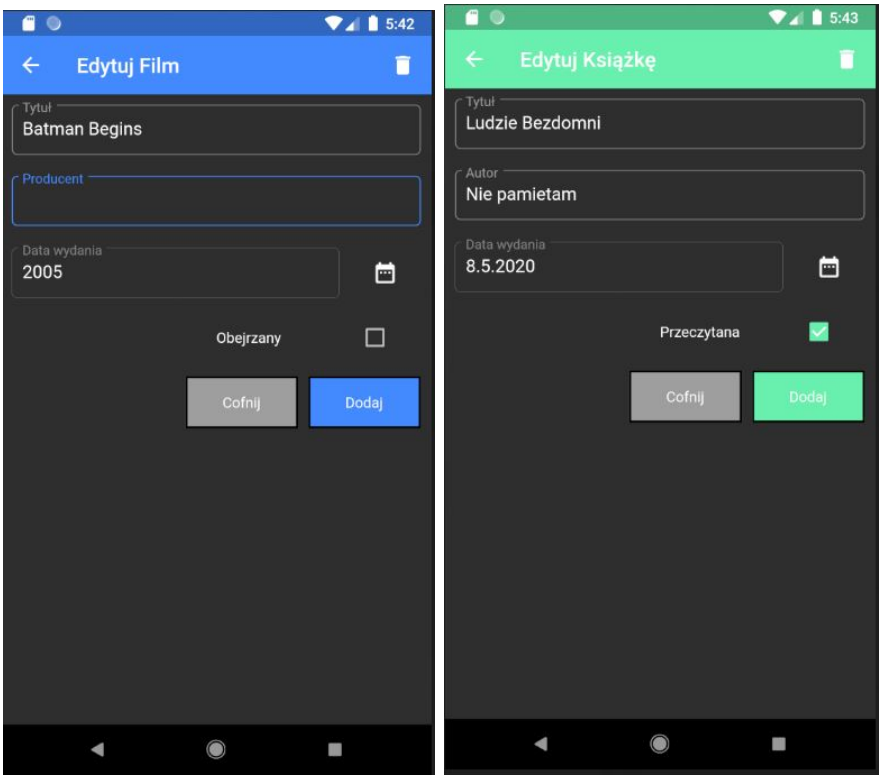
Zakładka gier i filmów

Klikając plus, na zakładce książek i gier, możemy dodać interesujące nas tytuły ręcznie, natomiast klikając plus w zakładce filmów przechodzimy do widgetu z wyszukiwarką filmów w *OMDB Api*



Zakładka gier i filmów

Dodane już elementy do naszej bazy możemy usuwać i edytować. Możemy usunąć przesuwając taki element na liście w prawo, bądź klikając w edycję, a następnie usunąć poprzez dotknięcie ikony śmietnika, gdzie naturalnie możemy także dokonywać zmian w bazie. Poprzez *OMDB Api* nie jest pobierany reżyser filmu, więc w takim przypadku musimy go dodać ręcznie.



Zakładka gier i filmów

4 Opinia

Próbowałem stworzyć aplikację która będzie przyjazna dla użytkownika - przede wszystkim w działaniu i wizualnie, zastosowałem proste widgety, komponenty. Aplikacja oferuje pełen CRUD na bazie Firebase. W mojej opinii jest dobrą podstawą pod rozbudowę i stworzenie bardzo zaawansowanego systemu do zarządzania rekordami w bazie za pomocą Fluttera i Firebase.

W repozytorium znajduje się kalendarium prac, dokumentacja, pliki projektu, oraz film prezentujący. https://github.com/dawbit/flutter-firebase