

<p style="text-align: center;"><b>Politechnika Świętokrzyska</b>  <b>Wydział Elektrotechniki, Automatyki i Informatyki</b></p>	
<p><b>Laboratorium:</b> Wprowadzenie do komunikacji człowiek-komputer  <b>Interfejs:</b> Dokumentacja techniczna</p>	
<p><b>Wykonawca:</b>  Przemysław Kałuziński  Michał Kaczor</p>	<p><b>Grupa:</b>  3ID13A</p>
<p><b>Data wykonania:</b> 18.05.2023</p>	

## Opis wybranego tematu

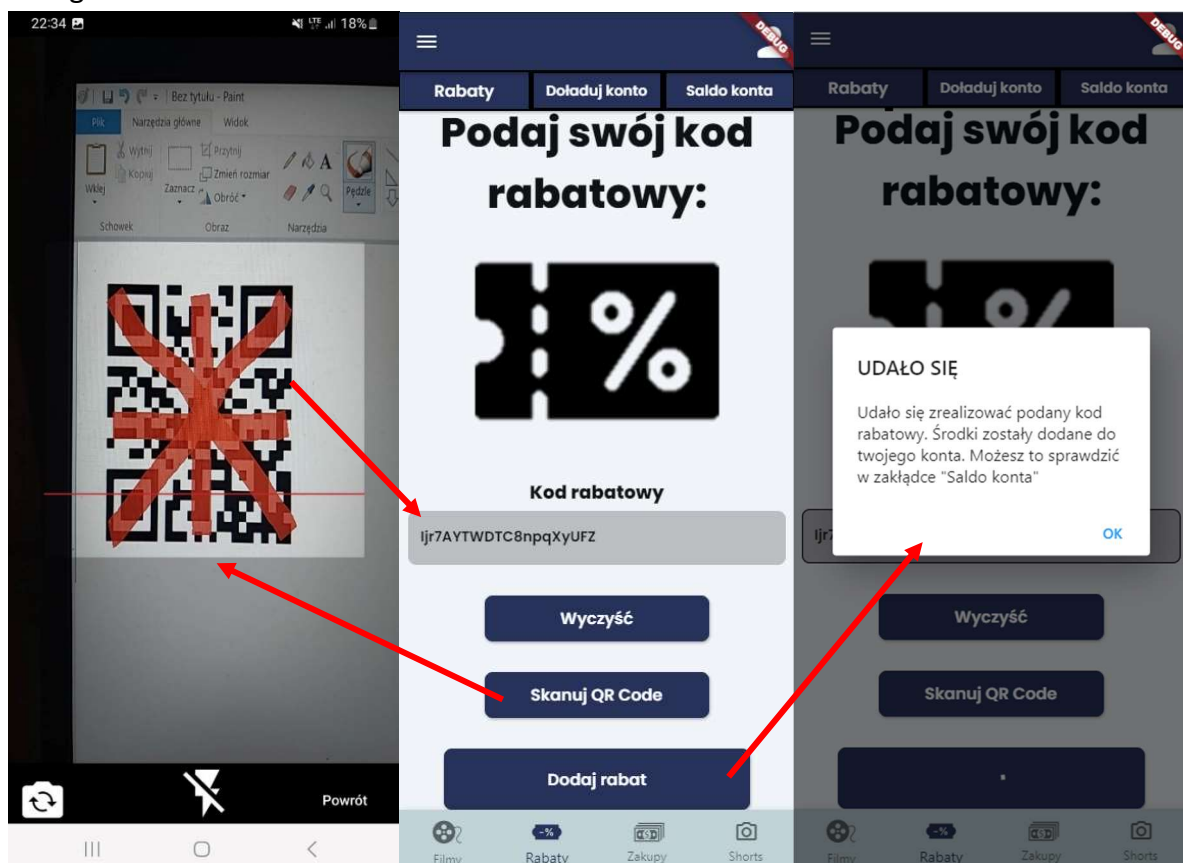
Wybrany przez nas tematem jest aplikacja użytkowa, umożliwiająca klientowi korzystanie z usług kinowych oraz streamingowych. Oferuje ona funkcjonalności zawarte w założeniach. Filmy i trailery wyświetlane w naszej aplikacji będą hostowane na serwisie rozrywkowo multimedialnym Youtube. Filmiki będą wyświetlane przy pomocy specjalnego odtwarzacza bezpośrednio w naszej aplikacji – bez przechodzenia do aplikacji Youtube. Część serwerową aplikacji postanowiliśmy rozwiązać przy pomocy zestawu usług hostingowych Firebase, który pozwoli nam na przechowywanie kluczowych informacji o kontach użytkowników oraz odnośników do filmów i trailerów.

## Urządzenia

Aplikacja korzysta z poniższych urządzeń telefonu:

- Czytnik kodów QR, który jest wykorzystywany do wczytywania kodów rabatowych
- GPS, który jest wykorzystywany do zlokalizowania użytkownika i przypisania mu najbliższej placówki kinowej

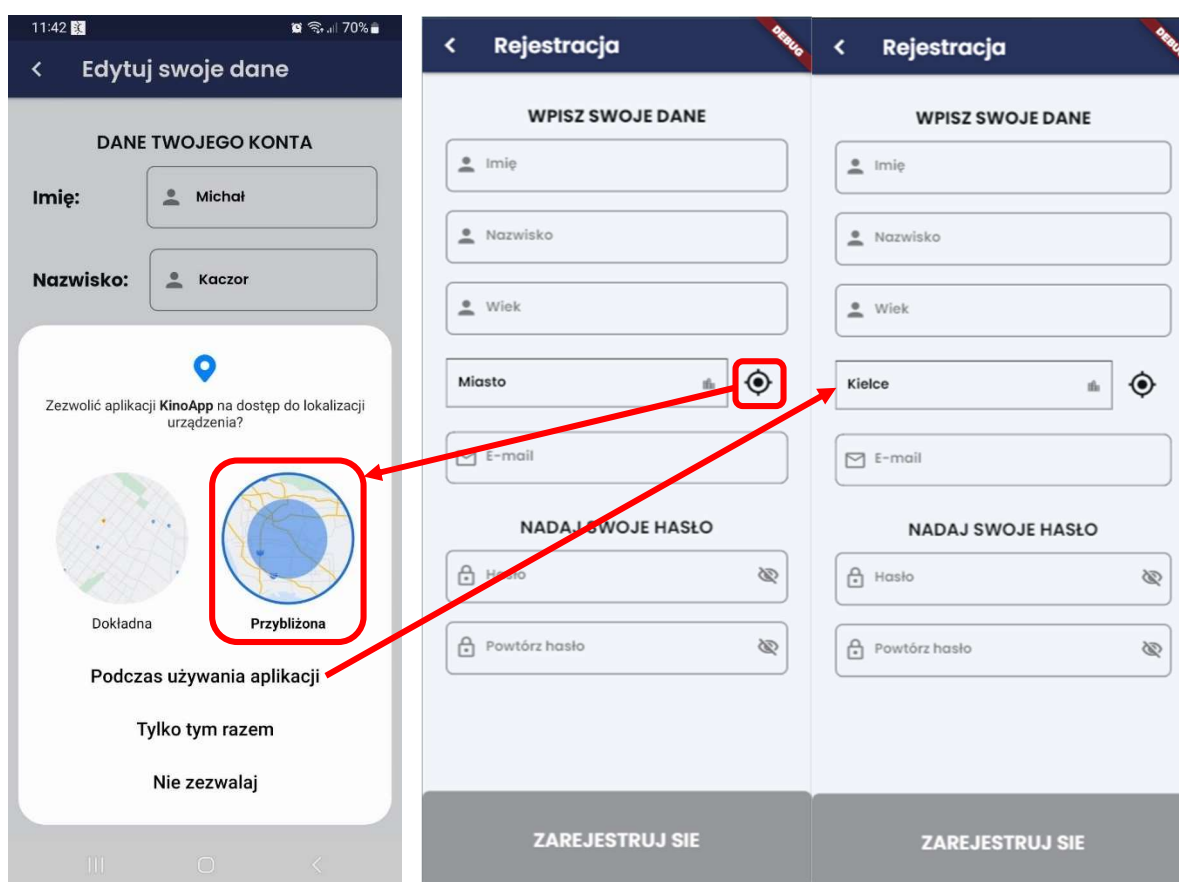
Aby wejść w tryb skanowania należy kliknąć przycisk „Skanuj kod QR” na ekranie kodów rabatowych. Otwarty zostanie aparat wraz ze skanerem. Należy pamiętać, że przy pierwszym uruchomieniu aplikacji, poprosi nas ona o odpowiednie uprawnienia. Po zeskanowaniu kodu, wrócimy do ekranu kodów rabatowych, a pole do wpisania kodu będzie już wypełnione. Następnie klikamy przycisk akceptacji i jeśli wszystko jest w porządku, to zostanie wyświetlony komunikat o powodzeniu operacji, a środki zostaną dodane do naszego konta.



### 3

Następnym urządzeniem jest lokalizacja telefonu (GPS). Wykorzystaliśmy je do ułatwienia użytkownikom zlokalizowania najbliższej placówki kinowej. Zamiast wybierać konkretne miasto w sposób manualny, poprzez wybór pozycji z DropDownList, użytkownik może nacisnąć ikonkę GPS obok, a aplikacja automatycznie zlokalizuje najbliższe kino w okolicy. Narzędzie to bazuje na aktualnej lokalizacji urządzenia i przy pomocy funkcji matematycznej liczy odległość między położeniem telefonu, a kolejnymi miastami. Następnie najbliższą znajdującą się lokalizacją jest automatycznie wybierana z listy rozwijanej. Jej zastosowanie można zauważyć na ekranach rejestracji i edycji danych osobowych. Podobnie jak w poprzednim przypadku należy pamiętać, że przy pierwszym uruchomieniu aplikacji na telefonie, zostaniemy poproszeni o udzielenie uprawnień na korzystanie z lokalizacji urządzenia.

Dodatkowym ważnym aspektem podczas udzielania uprawnień jest wybranie lokalizacji przybliżonej, a nie dokładnej. Dzięki temu, najbliższe kino będzie znajdowane o wiele szybciej, ze względu na fakt, że GPS nie marnuje czasu na ustalenie dokładnej lokalizacji urządzenia.

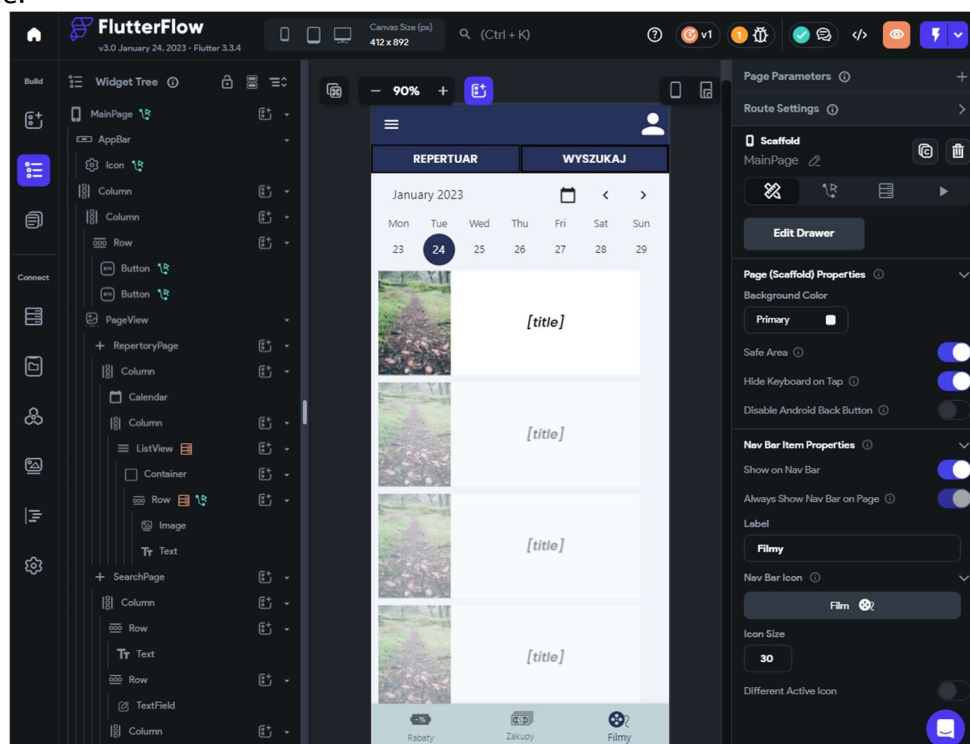


## 4

# Opis narzędzi i środowisk

## Flutterflow

Do prac nad projektem postanowiliśmy wykorzystać zaawansowane narzędzie programistyczne – Flutterflow. Jest to platforma do tworzenia aplikacji mobilnych i webowych za pomocą graficznego interfejsu użytkownika. Narzędzie to wykorzystuje popularny framework Flutter, który pozwala na szybkie tworzenie responsywnych aplikacji z użyciem natywnych elementów interfejsu użytkownika. W naszym projekcie wykorzystujemy Flutterflow jako główne narzędzie do projektowania i budowania aplikacji mobilnej. Dzięki temu możemy w łatwy sposób tworzyć interaktywne elementy interfejsu użytkownika, przeciągać i upuszczać elementy, łączyć je ze sobą i konfigurować zachowanie aplikacji za pomocą wbudowanych w narzędzie funkcji. Korzystając z jego prostego interfejsu typu „przeciągnij i upuść”, można zbudować w pełni funkcjonalną aplikację o wiele prościej niż w normalny sposób. Pozwala ono także testować aplikację w emulatorze telefonu w przeglądarce internetowej. Dzięki temu, nie trzeba za każdym razem debugować projektu przy użyciu kabla lub innego emulatora. Środowisko to pozwala także na stosowanie własnego kodu w postaci funkcji, które realizują rzeczy niewspierane przez narzędzie.



W celu płynniejszego testowania działania aplikacji, można wygenerować plik z rozszerzeniem .apk, a następnie zainstalować go na swoim urządzeniu mobilnym. Na szczęście nie trzeba robić tego manualnie, gdyż narzędzie Flutterflow od razu udostępnia nam opcję pobrania takiego pliku. W ten sposób łatwiej będzie wychwycić błędy. Co więcej, jest to jedyny sposób na sprawdzenie działania wykorzystanych narzędzi (skaner kodów QR oraz GPS), gdyż internetowy emulator nie oferuje takiej możliwości.

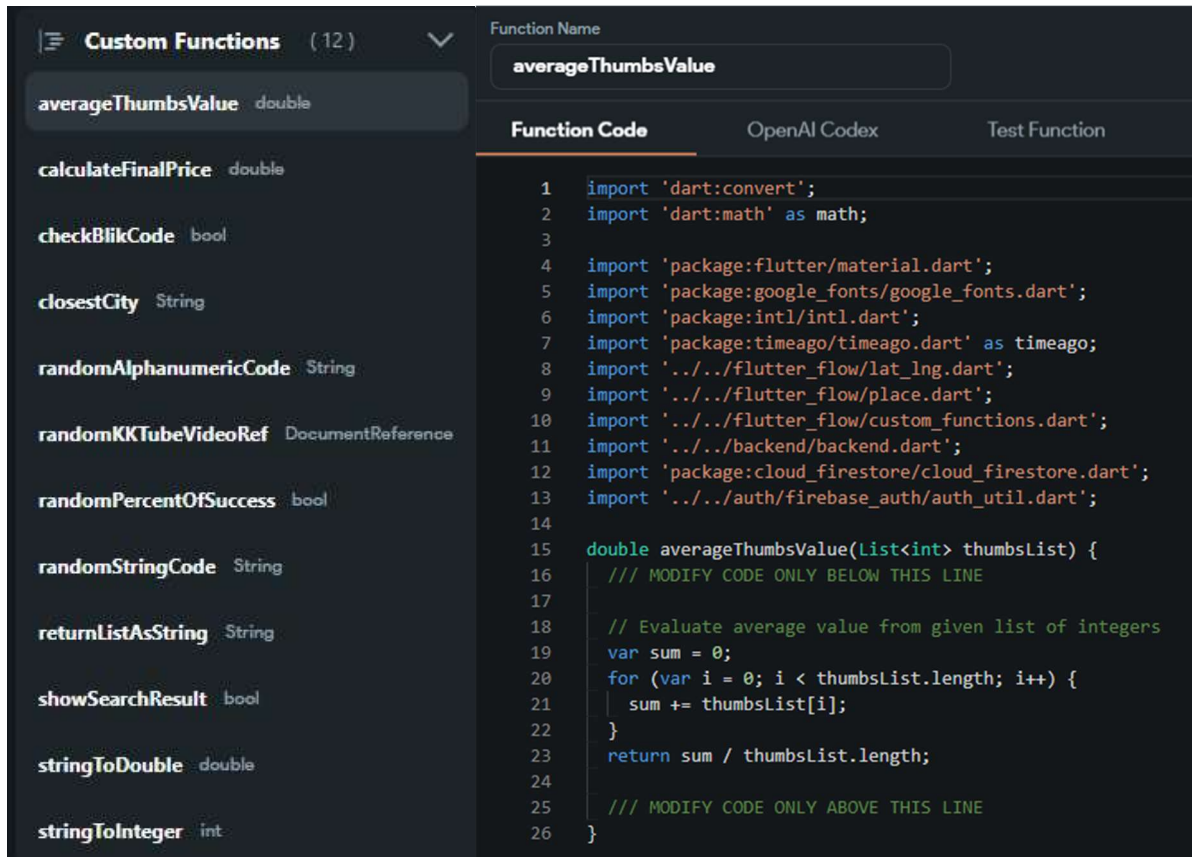
## 5

W przypadku, gdy środowisko Flutterflow nie posiada interesującej użytkownika funkcjonalności, to może on zaimplementować ją sam w postaci własnych funkcji. Najczęściej są to operacje matematyczne i tym podobne, dlatego w projekcie znalazły one zastosowanie głównie podczas operacji na ciągach znaków i liczbach. Dzięki nim, byliśmy w stanie przeprowadzać konwersje typów danych, generować losowe ciągi znaków lub sprawdzać ich poprawność. Wyszukiwarki filtrujące listy dokumentów w aplikacji również opierają swoje działanie na własnych funkcjach. Aplikacja wykorzystuje łącznie 12 własnych funkcji:

- **averageThumbsValue** – oblicza średnią wartość łapek w dół lub w górę dla wszystkich filmików danego użytkownika
- **calculateFinalPrice** – oblicza cenę końcową biletu na seans, bazując na zniżkach, wybranych miejscach oraz na tym czy użytkownik dokupił dodatkowo plakat
- **checkBlikCode** – sprawdza poprawność wpisywanego kodu Blik. Wymaganiem są same cyfry oraz długość 6 znaków
- **closestCity** – zwraca nazwę najbliższego miasta z placówką kinową bazując na obecnej lokalizacji urządzenia. Wykorzystywana wraz z urządzeniem GPS opisanym wyżej
- **randomAlphanumericCode** – generuje losowy ciąg alfanumeryczny o wskazanej długości. Wykorzystana do generowania kodów QR dla zakupionych biletów
- **randomKKTubeVideoRef** – miesza kolejność w otrzymanej liście referencji do filmików, dzięki czemu filmiki w zakładce shorts są zawsze wyświetlane w losowej kolejności
- **randomPercentOfSuccess** – zwraca true lub false w zależności od wylosowanej wartości typu Int. Służy do symulowania niepowodzenia operacji doładowywania konta Blikiem
- **randomStringCode** – generuje losowy ciąg o określonej długości złożony z samych liter
- **returnListAsString** – zwraca przekazaną listę jako ciąg znaków String. Dodatkowo opakuje każdy z elementów listy w znaki nawiasów. Wykorzystywana do wypisywania wybranych przez użytkownika miejsc siedzących w kinie.
- **showSearchResult** – filtruje dane na podstawie otrzymanych argumentów. Przyjmuje 2 parametry – czego ma szukać i gdzie ma szukać. Zwraca tylko te dane, które zawierają przekazaną w parametrze frazę.
- **stringToDouble** – konwertuje ciąg znaków String na zmienną typu Double
- **stringToInteger** – konwertuje ciąg znaków String na zmienną typu Integer

## 6

Poniżej znajduje się przykładowa implementacja funkcji obliczającej średnią wartość łapek w dół/górę, aby zobrazować format pisania własnych funkcji. Środowisko Flutterflow generuje nam podstawowy szablon, na podstawie wybranych parametrów funkcji, a użytkownik dopisuje tylko wewnętrzną logikę. Takie podejście zapewnia integralność własnych funkcji z narzędziem.



## 7

Poza własnymi funkcjami, w aplikacji wykorzystano jedno odwołanie API Call typu GET. Jest ono wykorzystane do sprawdzania poprawności wpisanego przez użytkownika kodu rabatowego. Z racji na fakt, że Flutterflow nie pozwala na wykonywanie zapytań BackendQuery w innym momencie niż podczas ładowania ekranu, musieliśmy skorzystać z innego rozwiązania. Zasada działania tej operacji opiera się na kodzie HTML zwracanym przez odwołanie. Po wpisaniu kodu rabatowego i kliknięciu przycisku, API Call próbuje pobrać z bazy danych dokument o odpowiednim ID. Jeśli taki dokument istnieje, odwołanie zwróci kod 200 oraz pobrany dokument w postaci pliku Json, w przeciwnym wypadku – kod 404. Na podstawie tych kodów oraz kilku innych zabezpieczeń już wewnątrz aplikacji rozróżniane jest czy kod rabatowy jest poprawny i spełnia wszystkie warunki.

**Define API Call**  
Provide the name and configuration for this API call.

**Call Definition** Response & Test

API Call Name  
GetPromoCode

Method Type  
**GET**

API URL  
https://firestore.googleapis.com/v1/projects/flutterkino/databases/(default)/documents/promoCodes/[id]

Headers Query Parameters **Variables** Advanced Settings

**Variables**

Name	Type	Is List	Default Value
id	String	False	

+ Add Variable

Delete Save















## 8

Do przechowywania danych lokalnych podczas wielu operacji wykorzystano zmienne lokalne nazywane w środowisku Flutterflow zmiennymi AppState. Dzięki nim, dane mogły być przechowywane i modyfikowane na bieżąco w trakcie wykonywania różnych czynności przez użytkownika bez użycia kosztownych odwołań do bazy danych. Dopiero na koniec danej operacji (np. kupna biletu) dane ze zmiennych lokalnych są aktualizowane w bazie danych. Kolejną zaletą takiego podejścia jest łatwy powrót do poprzednich wartości w przypadku, gdy użytkownik przerwie operację w trakcie. Najliczniejszą grupę zmiennych lokalnych w aplikacji stanowią miejsca siedzące na sali kinowej, ponieważ każde z nich posiada własną AppState. Zaznaczane miejsca podczas kupna biletu wpływają jedynie na ich stan w zmiennych lokalnych, a dopiero po zatwierdzeniu operacji będą widoczne dla innych użytkowników jako zajęte. Zmienne lokalne mogą również być zapisywane w urządzeniu, aby przywrócić ich stan po ponownym uruchomieniu aplikacji. W naszym przypadku nie było to konieczne i wszystkie zmienne wracają do swoich wartości domyślnych podczas zamknięcia aplikacji.

## App State

Define variables you want to store on the device across different pages.  
You can enable storage on device by checking the "Persisted" toggle on any variable.

+ Add State Variable

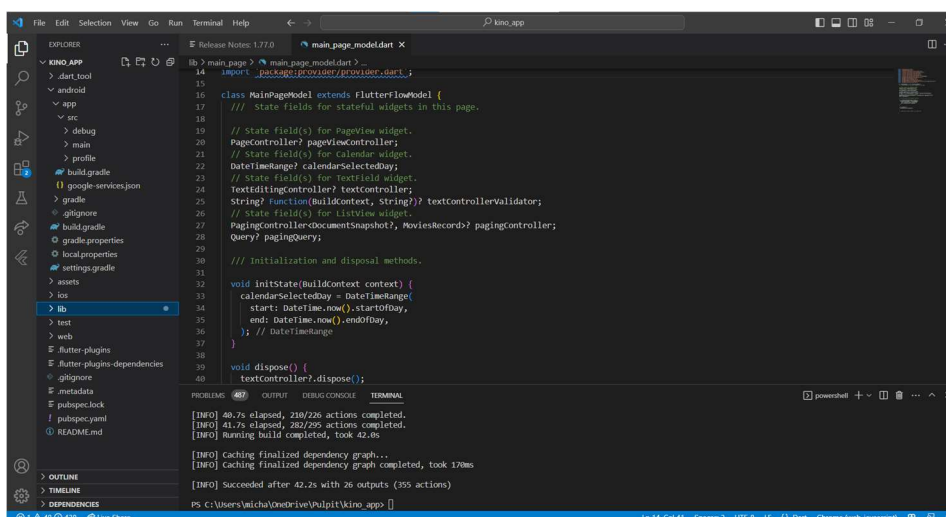
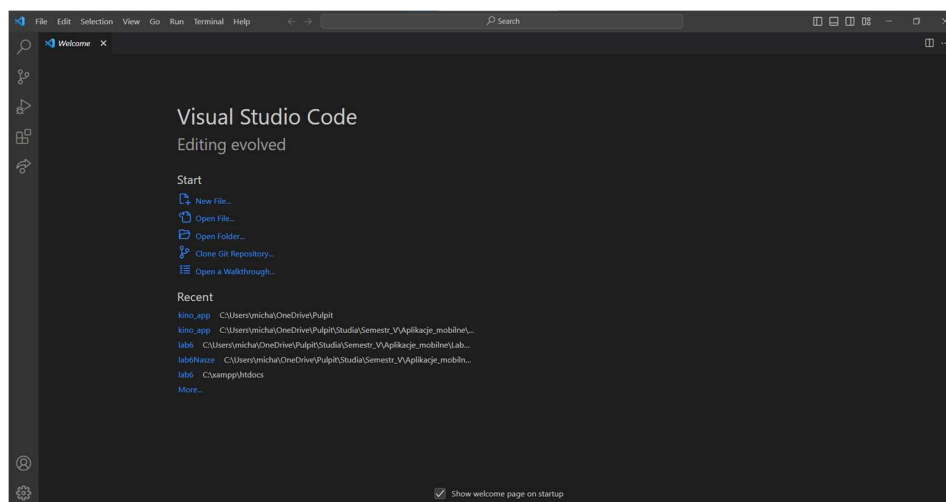
Field Name	Data Type	Persisted	Default Value	
closestCity	String	false	 closestCity	
seat00	Integer	false	 0	
selectedSeats	List < String >	false	 List < String >	
seat01	Integer	false	 0	
seat02	Integer	false	 0	
seat03	Integer	false	 0	
seat04	Integer	false	 0	



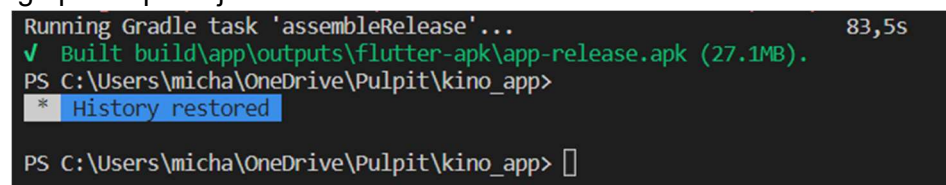
## 9 Visual studio

Flutterflow umożliwia również integrację z innymi narzędziami programistycznymi, co pozwala na większą elastyczność w pracy nad projektem. Podczas prac wykorzystaliśmy także ogólnodostępne środowisko programistyczne Visual Studio Code, które pozwoliło nam na wprowadzanie własnych modyfikacji w kodzie projektu, które byłyby ciężkie do zrealizowania przy użyciu powyższego narzędzia.

Visual Studio Code to popularne, otwarte i darmowe środowisko programistyczne, które umożliwia tworzenie, edycję i debugowanie kodu w wielu językach programowania. W przypadku naszego projektu, zdecydowaliśmy się wykorzystać Visual Studio Code jako dodatkowe narzędzie programistyczne do modyfikacji kodu aplikacji. Dzięki wykorzystaniu Visual Studio Code, można wprowadzać bardziej złożone modyfikacje w kodzie projektu, które byłyby trudne do zrealizowania bezpośrednio przy użyciu Flutterflow.



Po wprowadzeniu jakichkolwiek poprawek bezpośrednio w kodzie, należało wygenerować plik .apk manualnie z poziomu środowiska Visual Studio. Do przeprowadzenia tej operacji należy uruchomić wewnętrzny terminal środowiska i wywołać polecenie „flutter build apk”. Po krótkiej chwili zostaniemy poinformowani o zakończeniu operacji i będziemy mogli użyć utworzonego pliku aplikacji.



## 10

Najważniejszą ze zmian wprowadzanych poza środowiskiem Flutterflow było zablokowanie przycisku „Powrotu” wbudowanego w dolne menu nawigacji Androida. Powodowało to wiele problemów, ponieważ mimo zablokowania możliwości powrotu do danego ekranu z poziomu buildera Flutterflow, użytkownik mógł wykonać tą operację poprzez menu nawigacji urządzenia. Problem pojawiał się tylko dla wybranych ekranów, dlatego ta poprawka została wprowadzona tylko w niektórych miejscach, np. użytkownik był w stanie pominąć ekran powitalny bez klikania przycisku „Dalej”, co powodowało sytuację, że ekran ten był mu pokazywany za każdym razem, gdy się zalogował. Powodem tego był brak odznaczenia flagi w jego dokumencie w bazie danych, co realizował przycisk „Dalej”.

Poniżej znajduje się przykład implementacji powyższego zabezpieczenia dla ekranu powitalnego.

```
2  @override
3  void dispose() {
4    _unfocusNode.dispose();
5    super.dispose();
6  }
7
8  //WillPopScope sprawia, że przycisk back nawigacji Androida nie działa.
9  @override
10 Widget build(BuildContext context) {
11   return WillPopScope(
12     onWillPop: () async {
13       return false;
14     },
15     child: Scaffold(
16       key: scaffoldKey,
17       backgroundColor: FlutterFlowTheme.of(context).primaryBackground,
18       appBar: AppBar(
19         backgroundColor: Color(0xFF27325A),
20         automaticallyImplyLeading: false,
21         title: Text(
22           'HELLO THERE',
23           textAlign: TextAlign.center,
24           style: FlutterFlowTheme.of(context).title2.override(
25             fontFamily: 'Poppins',
26             color: Colors.white,
27             fontSize: 22,
28           ),
29       ),
30     ),
31   );
```

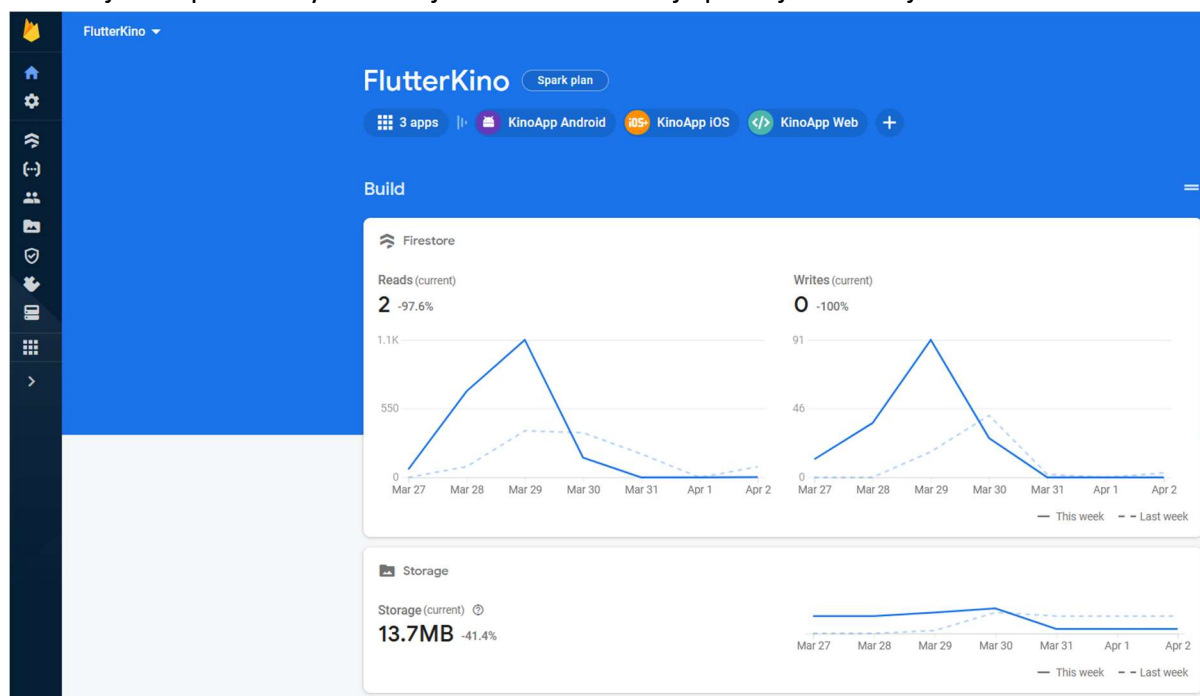
Aby zablokować nawigację wsteczną oferowaną bezpośrednio przez urządzenie, należało zastąpić domyślny WidgetBuild zaznaczonym kodem. Następnie obok Scaffold zamieniamy słowo kluczowe return: na child:. Od teraz przycisk „Powrót” znajdujący się na dolnym menu nawigacji urządzenia będzie nieaktywny dla danego ekranu. Jeśli chcielibyśmy zastosować tą operację dla wielu ekranów, to należy powtórzyć tą czynność w każdym z plików reprezentujących osobny ekran.

## 11 Firebase

Jako magazyn danych dla naszej aplikacji zdecydowaliśmy się wykorzystać bazę danych Firebase. Posłużyła nam ona do przechowywania kluczowych danych, obrazów oraz filmików. Firebase to platforma stworzona przez Google, która umożliwia łatwe przechowywanie, synchronizowanie oraz przetwarzanie danych w czasie rzeczywistym. Oferuje ona szeroki zakres narzędzi do tworzenia aplikacji mobilnych oraz webowych. W jej skład wchodzi m.in. bazy danych, funkcje backendowe, narzędzia do analizy i uwierzytelnianie użytkowników. Korzystanie z Firebase pozwala na łatwe i szybkie przechowywanie oraz synchronizację danych między aplikacją, a bazą danych.

W naszej aplikacji mobilnej wykorzystaliśmy funkcjonalności Firebase, takie jak autoryzacja, której zadaniem jest umożliwienie użytkownikom utworzenia nowego konta oraz późniejszego zalogowania się do aplikacji w celu dostępu do konkretnych funkcjonalności. Firebase Storage posłużył nam do przechowywania obrazów i filmików, które stanowią nieodłączny element aplikacji kinowo-streamingowej z dodatkiem funkcji Shorts. Firebase Database natomiast pozwolił na przechowywanie i zarządzanie informacjami dotyczącymi użytkowników oraz informacjami o aktywnościach w aplikacji.

Korzystając z Firebase, mogliśmy w łatwy sposób zarządzać bazą danych, a także zyskać dostęp do wielu przydatnych narzędzi i funkcjonalności, które umożliwią nam tworzenie bardziej skomplikowanych funkcjonalności w naszej aplikacji mobilnej.



## 12

Dane są przechowywane w kilku zależnych od siebie tabelach. Pojedynczy wiersz danych jednej z takich tabel nazywany jest dokumentem i można go określić jako samodzielny obiekt reprezentujący jakiś zestaw danych, np. informacje o filmie. Nasza aplikacja wykorzystuje łącznie 12 tabel, z których pobiera i zapisuje dane wyświetlane na ekranach.

flutterkino	cities	01uheY4j47czhfOrx8ko
+ Start collection	+ Add document	+ Start collection
+ Add field		
cities >	01uheY4j47czhfOrx8ko >	location: [51.4027235999999° N, 21.1471333° E] name: "Radom"
kkTubeAccount	5sCHpJY4sX0Mt4E0VcdG	
kkTubeVideoComment	A045JtMmoTP3F9TXZ0a	
kkTubeVideos	kkeLx3A6cVZTdUCCNKcX	
movies	1nb6MFAoyryxU3N0yENi	
promoCodes		
repertory		
repertoryDetails		
transactions		
userStreamings		
userTickets		
users		

Pliki multimedialne wykorzystywane w aplikacji przechowywane są w chmurze bazy danych. Aplikacja uzyskuje do nich dostęp poprzez skorzystanie z odnośnika do konkretnego pliku. Dlatego też, muszą być one umieszczane w zwykłych dokumentach w tabelach bazy danych, ponieważ firebase nie oferuje funkcji przeglądania plików z poziomu aplikacji.

FlutterKino

Storage

Files Rules Usage Extensions

Protect your Storage resources from abuse, such as billing fraud or phishing

gs://flutterkino.appspot.com

Upload file

Name	Size	Type	Last modified
MoviePosters/	—	Folder	—
cms_uploads/	—	Folder	—
users/	—	Folder	—
camera_upload.png	3.83 KB	image/png	Mar 23, 2023
defaultThumbnail.png	16.06 KB	image/png	Mar 23, 2023
defaultVideo.mp4	2.87 MB	video/mp4	Mar 23, 2023
user_avatar.png	53.22 KB	image/png	Mar 22, 2023
xqjOWQTZrFHQErN59hsX9K1kYCKiYT.jpg	86.69 KB	image/jpeg	Apr 3, 2023

gs://flutterkino.appspot.com > MoviePosters

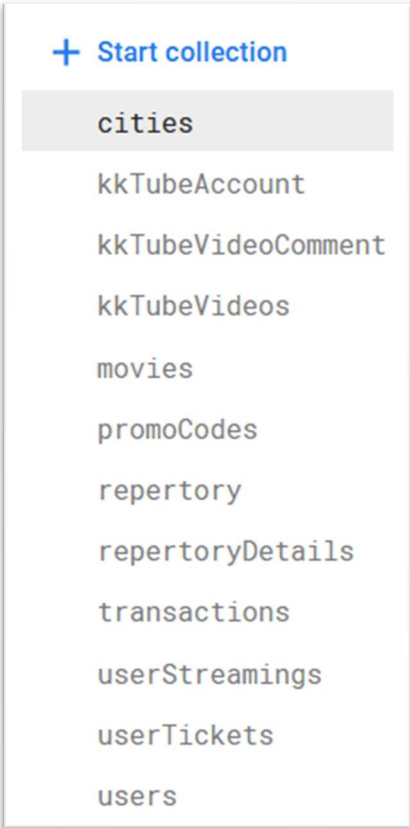
Upload file

Name	Size	Type	Last modified
007_Quantum_of_Solace.jpg	219.35 KB	image/jpeg	Jan 11, 2023
50_twarzy_Greya.jpg	134.8 KB	image/jpeg	Jan 11, 2023
CasinoRoyale.jpg	175.33 KB	image/jpeg	Jan 11, 2023
Epoka_lodowcowa.jpg	223.11 KB	image/jpeg	Jan 11, 2023
Epoka_lodowcowa_2.jpg	236.74 KB	image/jpeg	Jan 11, 2023

007\_Quantum\_of\_S...

007\_Quantum\_of\_Solace.jpg

Size: 224,615 bytes

	<p><b>cities</b> – przechowuje dane na temat miast, w których znajdują się placówki naszych kin</p>
	<p><b>kkTubeAccount</b> – przechowuje dane kont użytkowników na platformie KKTube. Najważniejszymi z nich są nick oraz lista subskrybentów. Jest to rozszerzenie do zwykłego konta użytkownika, które pozwala na korzystanie z zakładki Shorts, ale nie jest wymagane do korzystania z pozostałej części aplikacji</p>
	<p><b>kkTubeVideoComment</b> – przechowuje informacje na temat pojedynczego komentarza dodanego do któregoś z filmików innych użytkowników. Zawiera min. referencję do autora komentarza oraz do filmiku, pod którym został napisany.</p>
	<p><b>kkTubeVideos</b> – reprezentuje pojedynczy obiekt filmiku dodanego przez użytkownika KKTube. Zawiera wszystkie niezbędne informacje (tytuł, data utworzenia, itp.) oraz odnośniki (miniaturka, wideo). Dodatkowo każdy dokument filmików posiada 2 pola reprezentujące ilość tapek w górę lub dół.</p>
	<p><b>movies</b> – przechowuje informacje na temat wszystkich filmów oferowanych w naszych kinach w postaci seansów kinowych lub streamingów</p>
	<p><b>promoCodes</b> – przechowuje informacje na temat kodów rabatowych oraz wysokości doładowań jakie oferują</p>
	<p><b>repertory</b> – przechowuje informacje na temat filmów wyświetlanych w konkretnych dniach. W skrócie jest to tabela zawierająca referencję do obiektu tabeli movies oraz pole daty seansu</p>
	<p><b>repertoryDetails</b> – przechowuje informacje na temat godziny konkretnego seansu. W skrócie jest to tabela zawierająca referencję do obiektu tabeli repertory oraz pola charakterystyczne dla konkretnego seansu (godzina, siedzenia, itp.)</p>
	<p><b>transactions</b> – przechowuje informacje na temat wszystkich transakcji przeprowadzanych przez użytkowników w aplikacji. Transakcje konkretnych użytkowników są rozróżniane po polu identyfikatora użytkownika.</p>
	<p><b>userStreamings</b> - przechowuje informacje na temat wszystkich streamingów wykupionych przez wszystkich użytkowników aplikacji. Streamingi konkretnych użytkowników są rozróżniane po polu identyfikatora użytkownika.</p>
	<p><b>userTickets</b> - przechowuje informacje na temat wszystkich biletów kinowych wykupionych przez wszystkich użytkowników aplikacji. Bilety konkretnych użytkowników są rozróżniane po polu identyfikatora użytkownika.</p>
	<p><b>users</b> – przechowuje dodatkowe informacje na temat zarejestrowanych użytkowników, np. dane osobowe, miasto, itp. Nie przechowuje haseł.</p>

## 14

Hasła oraz inne dane autoryzacyjne są przechowywane w osobnym miejscu. Firebase oferuje osobną zakładkę o nazwie „Authentication”, w której przechowuje te dane. Wszelkie odwołania do bazy weryfikujące poprawność danych uwierzytelniających (np. podczas logowania) odbywają się w tym miejscu. To właśnie obiekt tej zakładki reprezentuje zarejestrowanego użytkownika, a nie obiekt tabeli users.

### Authentication

Users	Sign-in method	Templates	Usage	Settings
<div>Search by email address, phone number, or user UID</div> <div>Add user</div>				
Identifier	Providers	Created ↓	Signed In	User UID
pluczy10@gmail.com		Dec 12, 2022	Dec 14, 2022	wHcC9MPxhtfALnuMIColcnUX2h2

## Zabezpieczenia i walidacja danych

Aby umożliwić bezproblemowe korzystanie z aplikacji, zaimplementowaliśmy wiele zabezpieczeń mających na celu uniemożliwienie użytkownikom przeprowadzenie konkretnych działań. Są to min.

- Uniemożliwienie zakupu streamingu po raz drugi. Jeśli użytkownik podejmie próbę zakupu filmu, który już został przez niego wcześniej zakupiony, to aplikacja wyświetli stosowny komunikat i nie pozwoli mu przejść dalej.
- Uniemożliwienie wykorzystania tego samego kodu rabatowego drugi raz. Każdy z kodów rabatowych oferowanych przez nasze kino jest jednorazowy, a próba jego ponownego wykorzystania nie powiedzie się.
- Zabezpieczenie przeprowadzania operacji płatności w taki sposób, że aplikacja uniemożliwia zakup biletu lub streamingu jeśli użytkownik nie posiada wystarczającej ilości pieniędzy na koncie. Zostanie wtedy wyświetlony odpowiedni komunikat oraz nastąpi przekierowanie do ekranu doładowania salda konta.
- Zabezpieczenie formularzy wypełnianych przez użytkownika (np. rejestracja, edycja danych osobowych) w taki sposób, aby uniemożliwić mu pozostawienie pustych pól. Dodatkowo większość pól posiada nałożone odpowiednie wymagania wpisywanych danych, aby wymusić jak największą ich poprawność (np. imię i nazwisko może posiadać tylko wielkie i małe litery lub email, który musi spełniać określony regex).