POLITECHNIKA ŁÓDZKA
Wydział Elektrotechniki, Elektroniki,
Informatyki i Automatyki

Praca dyplomowa inżynierska

Mobilny system jako informator o wydarzeniach dla dzieci

Mobile system as an informer of events for children

Krzysztof Pijanowski

Nr albumu: 222540

Promotor:
Dr inż. Paweł Marciniak

Promotor pomocniczy:
Dr inż. Michał Szermer

Łódź, 2022

Streszczenie

Celem niniejszej pracy dyplomowej było opracowanie oraz zaimplementowanie systemu umożliwiającego dodawanie i przeglądanie wydarzeń dla dzieci na internetowej mapie Google. Aplikacja złożona jest z *serwera*, który przechowuje informacje, a następnie udostępnia je użytkownikom. Wśród przechowywanych danych znajdują się informacje o wydarzeniach, użytkownikach, postach oraz polubieniach. Serwer przechowuje też zdjęcia wydarzeń oraz awatary użytkowników.

Drugi element systemu stanowi *klient*, którego głównym zadaniem jest umożliwienie użytkownikom dodawania i przeglądania wydarzeń oraz korzystania z szeregu innych funkcji, takich jak wyszukiwarka czy własny profil.

Opracowany system mobilny został przystosowany wyłącznie do użytkowania w systemie operacyjnym Android. Aplikacja posiada nawigację, która umożliwia użytkownikowi szybki dostęp do kluczowych miejsc aplikacji po jednym kliknięciu w ekran smartfona. Dodatkowym udogodnieniem jest system powiadomień o nadchodzących wydarzeniach.

Słowa kluczowe: Flutter, Firebase, Android, Google Maps, wydarzenia

Abstract

The purpose of this thesis was to develop and implement a system that allows adding and viewing events for children on the Google Map. The application consists of *the server* that stores information and then makes it available to users. The stored data includes information about events, users, posts and likes. The server also stores photos of events and user avatars.

The second element of the system is *the client*, whose main task is to enable users to add and view events to the application. Moreover they can use many of other functions, such as event searching or modify their own profile.

The developed mobile system has been adapted only for Android operating system. The application has navigation that allows the user to quickly get to main places of the application with one click. An additional convenience is the system of notifications about upcoming events.

Keywords: Flutter, Firebase, Android, Google Maps, events

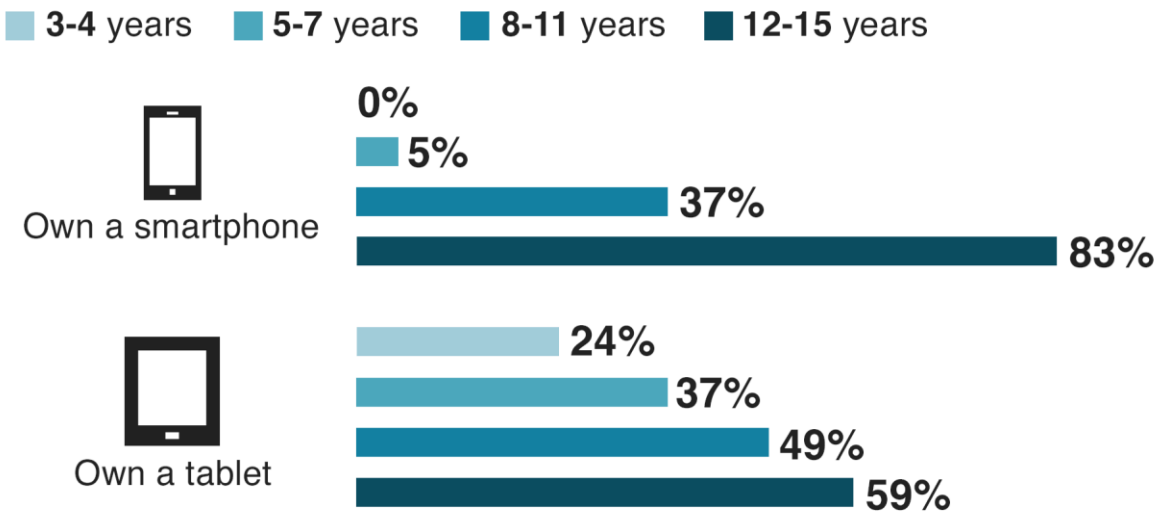
Spis treści

1. Wstęp

W obecnych czasach, gdy technologie informatyczne zawładnęły praktycznie każdą dziedziną naszego życia, większość ludzi korzysta z urządzeń mobilnych, takich jak np. telefony komórkowe i aplikacji na nich zainstalowanych. Telefon, który w XX wieku umożliwiał wyłącznie prowadzenie rozmów telefonicznych, w późniejszym okresie także przesyłanie wiadomości, współcześnie stał się dla wielu osób odpowiednikiem przenośnego komputera. Mimo niewielkich rozmiarów z powodzeniem zastępuje im ciężki sprzęt komputerowy i ułatwia dostęp do przydatnych funkcji niezależnie od miejsca, w którym się znajdują. Coraz wygodniejszy staje się swobodny, natychmiastowy wgląd w interesujące użytkownika treści. Często bez wychodzenia z domu (do czego zmuszają m.in. czasy szalejącej pandemii), siedząc na kanapie, można posługiwać się np. aplikacją bankową, przeglądać ogłoszenia, robić zakupy przez internet przy użyciu własnego smartfona.

W skomputeryzowanym świecie nie wolno zapominać o najmłodszych, którzy spędzają samotnie wiele godzin przed ekranami monitorów, zamiast bawić się z rówieśnikami i rozwijać swoje zainteresowania. Rodzice zajęci swoją karierą zawodową, a niejednokrotnie walczący o zdobywanie środków finansowych na utrzymanie rodziny, znajdują coraz mniej czasu dla swoich pociech, szczególnie by szukać dla nich rozrywki. Dawniej szkoły organizowały zajęcia pozalekcyjne, a świetlice szkolne zapewniały dzieciom ciekawe spędzanie czasu. W placówkach oświatowych i w zakładach pracy rozprowadzane były bilety na ciekawe imprezy kulturalne, informacje o takich wydarzeniach zamieszczane były w prasie, rozklejane na słupach ogłoszeniowych, murach budynków, latarniach czy ogrodzeniach. Ta forma promocji wydarzeń kulturalnych odchodzi w zapomnienie, a popularność komputerów i telefonów komórkowych może rozwiązywać wszystkie nasze problemy.

Warto przywołać dane statyczne z 2019 roku wygenerowane przez agencję rządową *Ofcom* (rysunek 1.1), które ujawniają, że w Wielkiej Brytanii około 50% dzieci w wieku 8-11 lat posiadało już do własnej dyspozycji tablet lub smartfon.



Rysunek 1.1. Odsetek użytkowników telefonów i tabletów w poszczególnych grupach wiekowych [1]

W starszej grupie wiekowej (12-15 lat) odsetek posiadaczy smartfonów jest znacznie wyższy i przekracza 80% [1]. Wraz z upływem czasu trend rosnący będzie się utrzymywał, w związku z czym warto pomyśleć o innych ciekawszych dla dzieci zajęciach niż ciągłe poruszanie się w środowisku gier i mediów społecznościowych.

Dynamiczny rozwój technologii informatycznych, wychodząc naprzeciw zapotrzebowaniom rynku, sprawia, że coraz częściej wszystkie aplikacje poza wersją przeglądarkową, tworzone są na potrzeby telefonów oraz tabletów.

1.1. Motywacja

Jednym z głównych motywów podjęcia tematu pracy stał się zamysł poszerzenia wiedzy na temat technologii związanych z tworzeniem aplikacji mobilnych oraz zdobycia umiejętności praktycznego ich zastosowania.

Wzrost zainteresowania wykorzystaniem nowoczesnych rozwiązań w codziennym życiu wśród coraz większej części populacji, skutkuje dynamicznym rozwojem rynku i bez wątpienia przyszłość będzie należała do mobilnych systemów informatycznych.

O wyborze aplikacji zadecydowała także chęć stworzenia programu korzystającego z Google Maps, obecnie jednego z najbardziej popularnych i rozpoznawalnych systemów na świecie. Nie istniałaby możliwość przygotowania aplikacji korzystającej z map bez oprogramowania firmy Google.

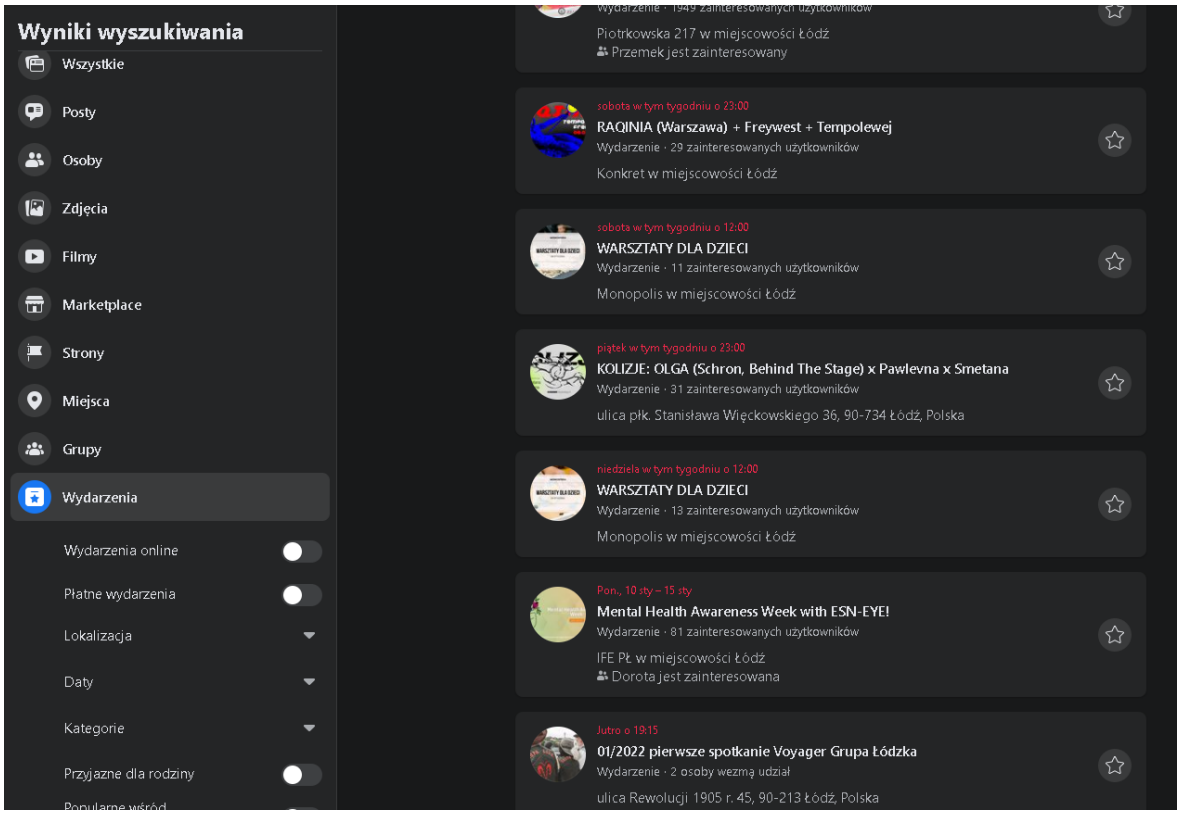
Za podjęciem takiego tematu przemawiała również perspektywa wygodnej pracy z platformą Firebase, która wykonuje większość zadań za programistę, a część serwerowa ogranicza się do wyboru odpowiednich ustawień i niewielkiego nadzoru nad systemem. Firebase zapewnia programiście autoryzację, bazę danych, przechowywanie plików, system powiadomień oraz wiele innych przydatnych funkcji.

Ostatnim argumentem, który miał wpływ na wybór tematyki stał się fakt niemal całkowitego braku odpowiedników tego typu aplikacji mobilnej (więcej na ten temat napisano w rozdziale 1.2). W przeciwieństwie do serwisów handlowych, takich jak Allegro, OLX czy zagraniczny eBay, gdzie panuje znaczna konkurencja, w przypadku aplikacji pozwalającej na przeglądanie wybranych elementów na mapie, pula propozycji wyraźnie się zawęża. Zrodziła się idea zaprojektowania nowego rozwiązania, które mogłoby wzbudzić powszechne zainteresowanie, a nawet stać się „towarem" rozchwytywanym.

1.2. Przykłady podobnych aplikacji

Na istniejącym obecnie rynku oprogramowania wyróżniają się dwa modele rozwoju. Pierwszy z nich, reprezentowany przez największą liczbę aplikacji, opiera się na wytyczonych już wcześniej standardach, wprowadzając w nich nowe funkcjonalności. Na uwagę zasługują także pojawiające się nowe trendy, które w obecnej perspektywie mają niewielki zasięg i stanowią nieznaczny odsetek udziału na rynku oprogramowania. W przyszłości mogą stanowić one rodzaj wskazówki w tworzeniu jakościowo nowych wzorców, wyznaczając nowe kierunki rozwoju aplikacji. Historia rozwoju technologii informatycznych, w tym również tworzenia projektów z tej dziedziny, potwierdza, iż niejednokrotnie projekty innowacyjne zrywające z obowiązującymi normami i schematami, stanowiły swoistą rewolucję, a w przyszłości wyznaczały nawet nowe trendy i standardy jakościowe (np. Windows).

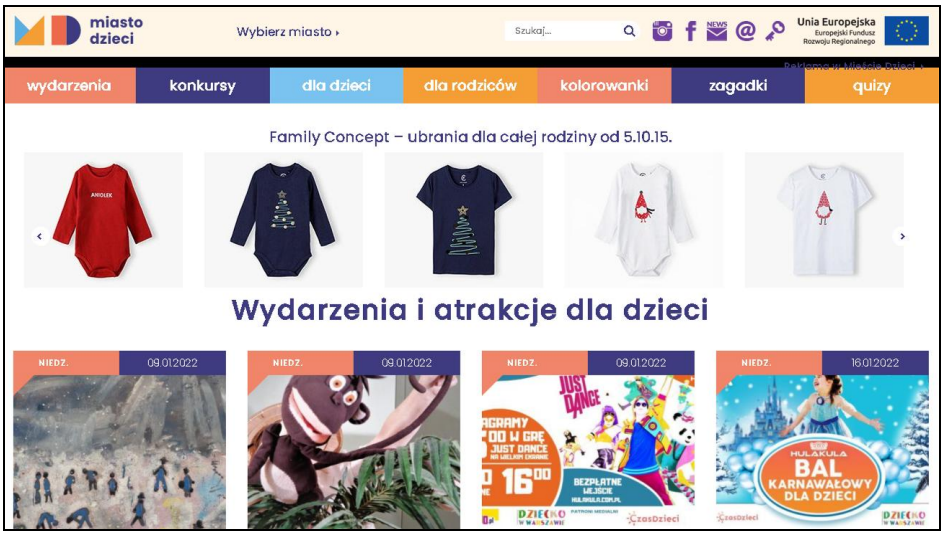
Przeciętny użytkownik, zainteresowany przeglądaniem wydarzeń z dowolnej dziedziny, pierwsze swe kroki w poszukiwaniu źródeł informacji kieruje najczęściej bez wątpienia w stronę systemu zaprojektowanego przez firmę Meta Platforms Inc. – popularnego Facebooka, który od wielu lat niezmiennie pozostaje przy tej samej formule wydarzeń (rysunek 1.2).



Rysunek 1.2. Facebook – widok wydarzeń [2]

Jest to serwis społecznościowy, który wyróżnia duża wyszukiwarka, ogromna baza użytkowników oraz szereg dodatkowych funkcji, jak np. możliwość dodawania zdjęć i filmów, wszystko połączone i zamknięte w monstrualnym internetowym gigancie. Wielkość i popularność Facebooka nie w każdym przypadku musi oznaczać zaspokojenie potrzeb użytkownika. Właśnie z uwagi na rozmiary i szeroki zakres udostępnianej tematyki, odnalezienie zawężonego zagadnienia staje się czasochłonne i motywuje do poszukiwania innych serwisów.

Znaną aplikacją umożliwiającą wyszukiwanie i przeglądanie wydarzeń dla dzieci i rodziców jest strona internetowa *MiastoDzieci.pl*. W przypadku tej domeny, obok bardzo ładnej szaty graficznej, przyciąga szereg rzadko spotykanych dodatków (rysunek 1.3). Są to między innymi piosenki, kolorowanki, bajki, zagadki, quizy itp. Filtrowanie wydarzeń oraz ich przeglądanie są intuicyjne. Szeroki wachlarz kategorii pomaga w wyborze tej konkretnej, najbardziej interesującej dla użytkownika oraz jego dziecka. Jedną z wad tego serwisu jest fakt, że nie przewidziano możliwości założenia na niej konta użytkownika, umożliwiającego zapisanie ustawień, preferencji wyszukiwania, kategorii wiekowej czy ulubionego rodzaju wydarzeń.

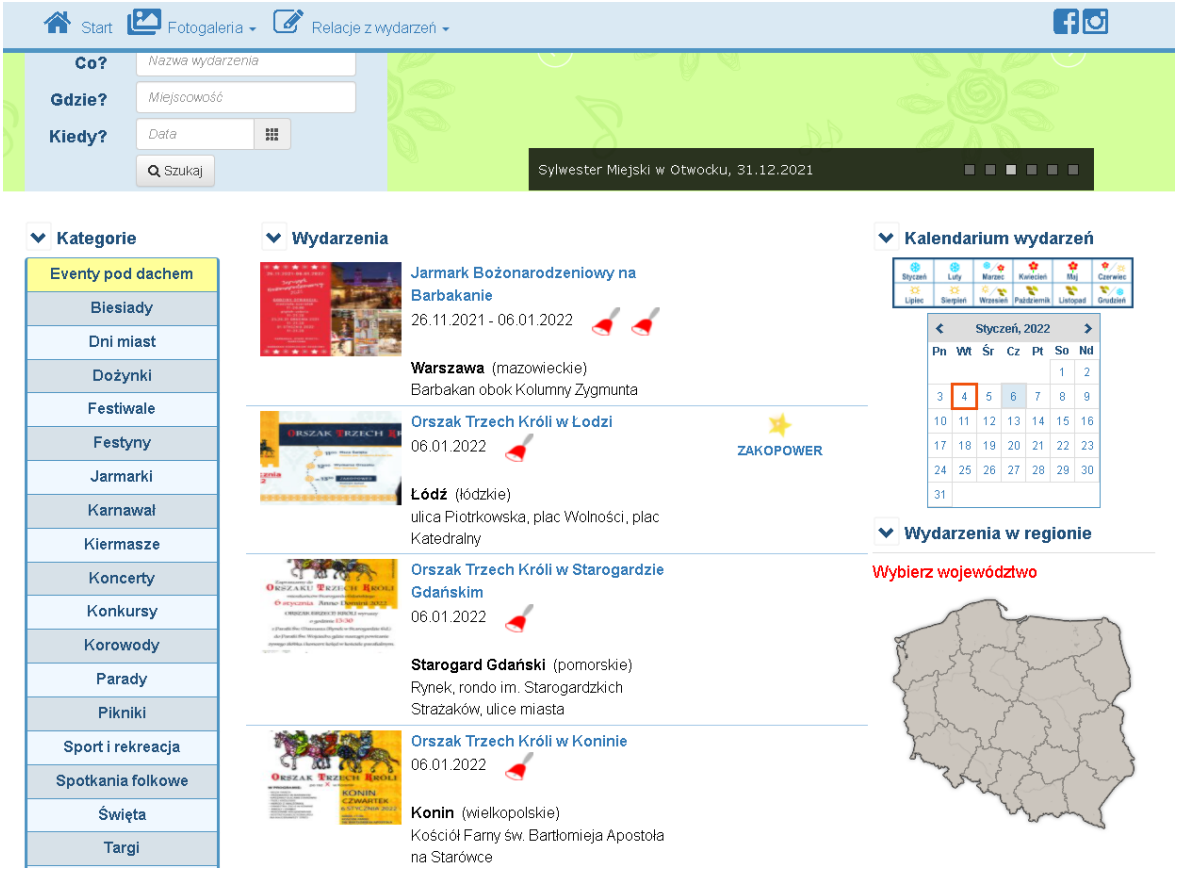


Rysunek 1.3. Widok strony głównej serwisu MiastoDzieci.pl [3]

Kolejnym przykładem tematycznej domeny dającej możliwość dokonania przeglądu wydarzeń dla dzieci jest strona *czasdzieci.pl*. Posiada ona dobrą wyszukiwarkę, oferującą wiele możliwości (rysunek 1.4). Można w niej wyszukać organizowane wycieczki, odnaleźć propozycje interesujących książek, czy zaczerpnąć pomysły na zapewnienie dziecku ciekawego zajęcia. Pomimo obiecującego pierwszego wrażenia, strona nie jest zbyt popularna, wydarzeń jest stosunkowo mało, a domena wydaje się być „martwą".

Rysunek 1.4. Widok strony głównej serwisu czasdzieci.pl [4]

Ostatnim godnym uwagi przykładem aplikacji dedykowanej dzieciom jest serwis *imprezowoplenerowo.pl*.



Rysunek 1.5. Widok strony głównej serwisu imprezowoplenerowo.pl [5]

Dość przestarzały wygląd strony (rysunek 1.5) jest rekompensowany przejrzystą, wygodną w użyciu wyszukiwarką. Stronę wzbogaca duża liczba kategorii, a poszczególne elementy są filtrowane bezpośrednio po kliknięciu. Strona jest chętniej odwiedzana przez użytkowników, ale nadal nie jest to zainteresowanie na skalę Facebooka.

Istniejąca, ograniczona liczba propozycji, jakich dostarcza internet, utwierdza w przekonaniu o konieczności podjęcia próby stworzenia nowej aplikacji z myślą o najmłodszych.

1.3. Założenia oraz cel projektu

Przed rozpoczęciem jakiegokolwiek projektu ważnym elementem staje się zdefiniowanie początkowych założeń, a następnie podjęcie próby ich pełnej realizacji. Celem niniejszej pracy jest implementacja mobilnego systemu, przeznaczonego dla telefonów z systemem operacyjnym Android, który powinien w łatwy i przejrzysty sposób umożliwiać użytkownikowi przeglądanie wydarzeń na mapie oraz śledzenie tych, które go interesują. Dodatkowo aplikacja powinna być dostosowana funkcjonalnością i treściami dla osób z dziećmi.

System powinien opierać się na dwustronnej wymianie danych pomiędzy bazą danych (Firebase) a telefonem. Zmiany powinny być widoczne w realnym czasie, tzn.: kiedy zostanie coś dodane do aplikacji przez użytkownika „A” - zmiana powinna być natychmiast widoczna dla użytkownika „B”.

Aplikacja powinna zawierać:

- mechanizm logowania i rejestracji,
- podgląd mapy z dodanymi wcześniej wydarzeniami oraz podstawowym ich opisem,
- opis wydarzenia,
- system dodawania postów z dostępną opcją „polubień”,
- system dodawania nowych wydarzeń z formularzem informacji na ich temat,
- profil użytkownika z możliwością edycji danych, usuwania konta, dodania zdjęcia profilowego oraz przeglądu wydarzeń, na które jest on zapisany,
- wyszukiwarkę,
- zaimplementowaną rolę administratora, który zarządza użytkownikami (nadaje im prawa) oraz danymi w bazie danych,
- system powiadomień o zbliżającym się wydarzeniu.

2. Technologie

Przed przystąpieniem do tworzenia każdego projektu niezbędny jest dobór odpowiednich technologii informatycznych, dzięki którym realizacja wszystkich jego założeń i celów ma szansę powodzenia. Spośród licznych, najbardziej znanych i popularnych technologii wybrano i krótko omówiono te, które uznano za przydatne w procesie przygotowywania aplikacji.

2.1. Flutter

Flutter jest technologią wydaną przez firmę Google w maju 2017 roku. Opiera się ona głównie na języku programowania *Dart* i wspiera aplikacje stworzone na trzy podstawowe systemy operacyjne przeznaczone dla komputerów (Windows, Linux oraz macOS), a także systemów mobilnych (Android oraz iOS). Dzięki temu istnieje możliwość tworzenia serwisów internetowych oraz aplikacji mobilnych, dając dużą uniwersalność omawianej technologii. Warto dodać, że w 2020 roku Flutter był najchętniej stosowanym narzędziem do budowania aplikacji mobilnych. Szereg dostępnych materiałów pomocniczych i poradników poprawia jakość i daje przyjemność programowania [6][7].

Sam *Dart* jest obiektowym językiem programowania, który posiada mechanizm „odśmiecania pamięci” (*garbage-collector*) - jednej z metod zarządzania dynamicznie przydzielaną dla naszego programu pamięcią. W przypadku języka *Dart* programista nie jest odpowiedzialny za jej zwalnianie, co ułatwia pisanie programu [8]. Możliwość kompilacji do natywnego kodu, otwiera przed nami kolejne możliwości [9].

Ważnym do wyjaśnienia zagadnieniem jest natywność i hybrydowość aplikacji. Natywność daje bardziej indywidualne podejście do tworzonego programu, a hybrydowość zapewnia wieloplatformowość. Oba podejścia do programowania posiadają zalety, ale również nie są pozbawione wad. Flutter w pewnym sensie łączy w sobie obie funkcjonalności. Raz napisany kod będzie działał na wielu platformach, pozostając przy tym bardzo wydajnym oraz optymalnym [10][11][12].

2.2. Android OS

Z powodu ograniczonej dostępności mobilnego sprzętu umożliwiającego testowanie programu na systeme iOS, zakres niniejszej pracy obejmuje jedynie stworzenie aplikacji poświęconej systemowi Android.

Zgodnie z danymi dostępnymi w serwisie statcounter zajmującym się analityką internetową [13], w grudniu 2021 roku około 70% smartfonów korzystało z Androida, 29% z iOS, pozostały 1% wypełniały inne mobilne systemy operacyjne. Daje to wyraźną wskazówkę, że aplikacje napisane na system operacyjny Android mogą pokrywać zapotrzebowanie rynku niemal w 3/4 przypadków.

Sam system operacyjny został stworzony przez firmę Android Inc., później prawa do niej odkupiła firma Google. Pozwala to na korzystanie z wielu różnorodnych usług (firmy Google) dedykowanych specjalnie dla tego systemu. Poniżej wymieniono jedne z najbardziej popularnych:

- *Asystent Google* – inteligentny asystent, potrafiący prowadzić dwustronną konwersację w wielu językach,
- *Dysk Google* oraz *Dokumenty Google* – usługa umożliwiająca przechowywanie, synchronizację i udostępnianie plików,
- *Gmail* – serwis mailowy,
- *Mapy Google* – serwis umożliwiający przeglądanie i wyszukiwanie różnych obiektów na mapie (więcej w rozdziale 2.4),
- *Google Play* – serwis internetowy, udostępniający aplikacje, gry, muzykę, książki itd.
- *Youtube* – serwis internetowy pozwalający na umieszczanie na swojej stronie filmów, ich komentowanie oraz nadawanie na żywo.

Najbardziej kluczową funkcją jest to, że umożliwia korzystanie z map Google, które stanowią niezbędną podstawę dla tego projektu [14]. Dzięki udostępnionej usłudze map Google, możliwe jest dodawanie własnych znaczników wydarzeń, które każdy użytkownik aplikacji będzie mógł zobaczyć.

2.3. Firebase

Technologia Firebase sięga swoją historią do 2011 roku. Pierwotnie Firebase stanowiła niezależną firmę, jednak w 2014 roku prawa do tej platformy nabyła firma Google. Obecnie jest ona jedną z najbardziej popularnych technologii używanych do tworzenia aplikacji internetowych oraz mobilnych. Ważnym udogodnieniem Firebase jest możliwość synchronizacji danych na różnych urządzeniach. Same informacje przechowywane są w „chmurze", która stanowi rozbudowaną bazę danych. W dzisiejszych czasach platforma pomaga programistom w tworzeniu aplikacji mających współpracować ze sobą i wymieniać dane w czasie rzeczywistym [15][16].

Obecnie rynek pełen jest technologii serwerowych. Zapewniają one szybki i łatwy sposób rozpoczęcia pracy nad aplikacją. Programista nie musi konfigurować części serwerowej, co znacząco przyspiesza tworzenie kodu, komfort pracy i testowanie [17]. Firebase oferuje szereg usług, które były kluczowe dla przygotowania niniejszej pracy. Są to m.in.:

- *Firebase Auth*, który dostarcza wiele opcji uwierzytelniania: przez Gmaila, Facebooka, Twittera czy GitHuba. W przypadku przygotowywanej aplikacji będzie to opcja autoryzacji za pomocą e-maila oraz hasła;
- *Firebase Storage*, który zapewnia magazyn do przechowywania danych (tj. obrazów, audio, wideo czy plików tekstowych);
- *Firebase Cloud Messaging*, który daje możliwość wysyłania wcześniej określonych powiadomień do telefonu użytkownika;
- *Baza danych czasu rzeczywistego Firebase*, która zapewnia podstawową i najważniejszą usługę, jaką jest przechowywanie danych (w formacie JSON). Daje ona możliwość wielokierunkowej komunikacji między smartfonami użytkowników a bazą danych. Wszelkie aktualizacje informacji są odzwierciedlane w czasie rzeczywistym dla wszystkich połączonych klientów.

Oprócz wyżej wymienionych istnieje szereg innych usług przydatnych dla programisty, takich, jak np.:

- *Firebase Hosting*, która daje możliwość bezpiecznego, globalnego hostingu CDN,
- *Pakiet ML Kit*, który dostarcza możliwości uczenia maszynowego,
- *Google Analytics for Firebase* – pakiet analityczny, pokazujący statystyki użytkowników (w jaki sposób korzystają z aplikacji),

- *Firebase Predictions*, które są prognozami opartymi na danych zebranych przez Google Analytics,
- *Firebase In App Messaging* – pakiet podobny do pakietu Firebase Cloud Messaging. W przypadku FCM nadawcą powiadomienia jest serwer, natomiast powiadomienia FIAM pochodzą od aplikacji,
- *Firebase Remote Config*, który daje programiście możliwość wprowadzenia zmian w mechanizmie działania aplikacji lub w jej wyglądzie, bez konieczności publikowania aktualizacji,
- *Cloud Functions* – zdefiniowane wcześniej funkcje, automatycznie uruchamiane w przypadku konkretnego zdarzenia.

Firebase jest dobrym rozwiązaniem dla prawie każdego projektu. Jedynym ograniczeniem może być platforma, dla której jest tworzony. Bez wątpienia jedną z wyjątkowych zalet, którą ten system się wyróżnia jest baza danych, zapewniająca aktualizację danych „w czasie rzeczywistym" (podczas zmian w bazie danych).

2.4. Mapy Google

Mapy Google to jedna z najbardziej rozpoznawalnych usług firmy Google. Jest to serwis internetowy, dzięki któremu można m.in. wyszukiwać obiekty, oglądać mapy i zdjęcia lotnicze powierzchni Ziemi, planować trasy itd. Elementem kluczowym dla przygotowywanego projektu jest Google Maps API, który pozwala na wygenerowanie specjalnego klucza, dającego możliwość wstawienia i modyfikowania własnej mapy w dowolnej aplikacji czy serwisie internetowym [18].

2.5. Emulator (Android)

Emulatorem określa się program komputerowy, który uruchomiony w danym systemie operacyjnym udostępnia analogiczne funkcje charakterystyczne dla innego systemu [19]. W przypadku emulowania systemu Android niezbędna jest instalacja specjalistycznego programu, Android Studio. Jedną z funkcji tej aplikacji, poza środowiskiem do pisania kodu, jest łatwość utworzenia dowolnego emulatora Android. Program ten daje możliwość wyboru urządzeń spośród smartfonów, telewizorów czy tabletów, w widoku dostosowanym do wielu rozmiarów ekranu.

Ważnym etapem każdego projektu jest jego testowanie. W przypadku poniższej pracy można to było wykonać na dwa sposoby, tj. poprzez:

- testowanie z wykorzystaniem smartfona – daje to możliwość bardziej realnych doznań w odniesieniu do napisanej aplikacji,
- testowanie z wykorzystaniem emulatora – zapewnia dużą elastyczność, szybkość i wygodę weryfikacji działania.

2.6. GitHub

Ważnym elementem w tworzeniu każdej aplikacji jest zabezpieczenie swoich danych przed ich utraceniem. GitHub stanowi jeden z najbardziej rozpoznawalnych systemów kontroli wersji Git, który od 2008 roku jest dynamicznie wspierany i rozwijany. W przypadku wykrycia błędów można zawsze dokonać cofnięcia zmian i rozwiązać problem. Zaletą korzystania z Gita jest jego prostota. Aplikacja desktopowa GitHuba sprawia, że przy użyciu kilku „kliknięć" można dodawać lub usuwać aktualizacje projektu [20].

3. Usługi Firebase

Wprowadzające informacje dotyczące usługi Firebase zostały przedstawione w rozdziale 2.3. W tym miejscu poruszone będą zagadnienia nawiązujące do struktury systemu, projektowania i przechowywania informacji w bazie danych.

3.1. Struktura systemu

Głównymi filarami systemu jest telefon użytkownika oraz technologia Firebase. Cała aplikacja opiera się na dwustronnej wymianie informacji pomiędzy tymi dwoma podmiotami [15]. W obecnym stanie, autoryzacja i zaprojektowane role stanowią limiter. W przypadku, kiedy użytkownik nie posiada autoryzacji (nie jest zalogowany), jedyne wymieniane informacje to dane logowania i dane użytkownika. W przyszłości, przy wprowadzeniu roli „gość”, wymiana informacji mogłaby być jednostronnie możliwa. Przegląd wydarzeń i innych elementów mógłby być dostępny, ale dodawanie nowych - już nie. Poniższy rysunek 3.1 prezentuje aktualną architekturę systemu.

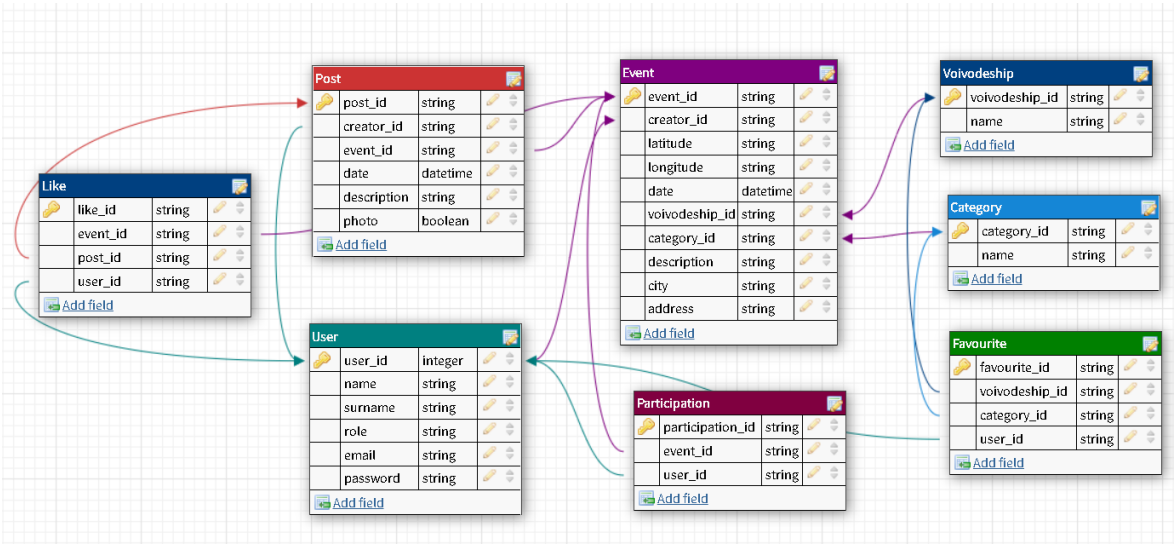
Rysunek 3.1. Schemat architektury systemu aplikacji [opracowanie własne]

Wyróżnia się trzy statusy komunikacji:

- *Niezalogowany użytkownik* – jedyna wymiana, jaka jest możliwa, to rejestracja oraz sprawdzenie danych logowania, nie ma komunikacji i wymiany informacji pomiędzy innymi kolekcjami (szczegóły opisane zostały w rozdziale 3.2)
- *Zalogowany użytkownik* – występuje dwustronna wymiana danych w przypadku wszystkich tabel bazy danych,
- *Administrator* – komunikacja występuje w przypadku wszystkich elementów jak u zalogowanego użytkownika. Dodatkowo administrator ma większe możliwości kontrolowania bazy danych. Może wysyłać do niej informacje, tj. dodawać nowe kategorie lub usuwać użytkowników.

3.2. Schemat bazy danych

Nieodłacznym elementem tworzonego projektu było przygotowanie bazy danych, która przechowywałaby wszystkie niezbędne dane [15]. Drugim ważnym zadaniem było odpowiednie przemyślenie powiązania pomiędzy tzw. kolekcjami (tabele w bazie danych). Ostateczny schemat bazy danych został przedstawiony na rysunku 3.2.



Rysunek 3.2. Ostateczny schemat bazy danych [opracowanie własne]

Schemat ten zawiera osiem kolekcji. Każdy pojedynczy obiekt tworzony w tej bazie danych otrzymał unikalny numer ID. Zadaniem kolekcji jest przechowywanie kluczowych powiązań pomiędzy obiektami oraz informacji na ich temat (w formie referencji).

Baza danych jest nierelacyjną bazą danych (kolekcje pozbawione są relacji między sobą), ale dzięki referencjom można utrzymywać pewne powiązania pomiędzy różnymi kolekcjami. Wyróżniono następujące kolekcje:

- *Użytkownik (user)* – podstawowa kolekcja, która ma przechowywać informacje o zarejestrowanych w systemie użytkownikach. Pola, które posiada to: imię, nazwisko, e-mail, rola oraz ID. Osobnym elementem jest hasło, które jest „haszowane" i przechowywane w innym miejscu (dzięki Firebase Auth);
- *Województwo (voivodeship)* – kolekcja przechowująca tylko nazwę konkretnego województwa z unikalnym ID;
- *Kategoria (category)* – kolekcja przechowująca tylko nazwę kategorii z unikalnym ID;
- *Wydarzenie (event)* – kolekcja opisująca wydarzenie dodane do serwisu. Zawiera wszystkie kluczowe informacje dotyczące miejsca oraz czasu. Dwa pola rzutują na tabelę kategorii oraz województwa;
- *Post* – kolekcja przechowująca informacje o konkretnym poście. System pozwala wydobyć informację o dacie dodania, autorze oraz zamieszczeniu zdjęć.

Pozostałe trzy kolekcje spajają informacje z innych kolekcji, określając pewnego rodzaju status:

- *Udział w wydarzeniu (participation)* – kolekcja łącząca kolekcję użytkowników oraz wydarzeń,
- *Polubienia (like)* – kolekcja łącząca kolekcję użytkowników, wydarzeń oraz postów,
- *Ulubione wyszukiwania (favourites)* – kolekcja łącząca kolekcję kategorii i województw z użytkownikiem.

3.3. Firebase Storage – przechowywanie zdjęć

Zdjęcia wydarzeń i użytkowników wymagają przechowywania w serwerze. Niemożliwym jest, aby każdy użytkownik mógł je gromadzić w pamięci swojego telefonu, ponieważ wówczas dane nie byłyby widoczne dla innych użytkowników (awatary i zdjęcia w poście). W tym przypadku należało skorzystać z usługi Firebase Storage.

Osobnym zadaniem było zaprojektowanie oryginalnej mechaniki przechowywania zdjęć (tak, by wszystko funkcjonowało). Obecnie każdy awatar posiada nazwę pliku zgodną z numerem ID użytkownika. W przypadku, kiedy użytkownik nie posiada awatara lub pierwszy raz loguje się do aplikacji, wyświetlane jest domyślne zdjęcie. Trudniejsze do przechowywania są zdjęcia dodawane do postów. W tym przypadku należy tworzyć ścieżkę do plików graficznych, ponieważ jedno wydarzenie może mieć wiele postów ze zdjęciami. Zdjęcia (jeśli istnieją) będą znajdować się pod ścieżką */event_id/post_id.*

Dostępność zdjęć dla konkretnego posta jest przechowywana w tabeli jako zmienna typu *bool* (prawda lub fałsz). W przypadku usunięcia konta lub wydarzenia, zaimplementowany mechanizm powinien sam usunąć wszystkie nieaktualne zdjęcia.

4. Aplikacja kliencka

Głównym zadaniem pracy było zaprojektowanie i zaprogramowanie aplikacji klienckiej. Powinna ona zawierać kompletne rozwiązania zgodne z założeniami projektu. Aplikacja musi komunikować się z serwerem Firebase i pobierać z niego informacje.

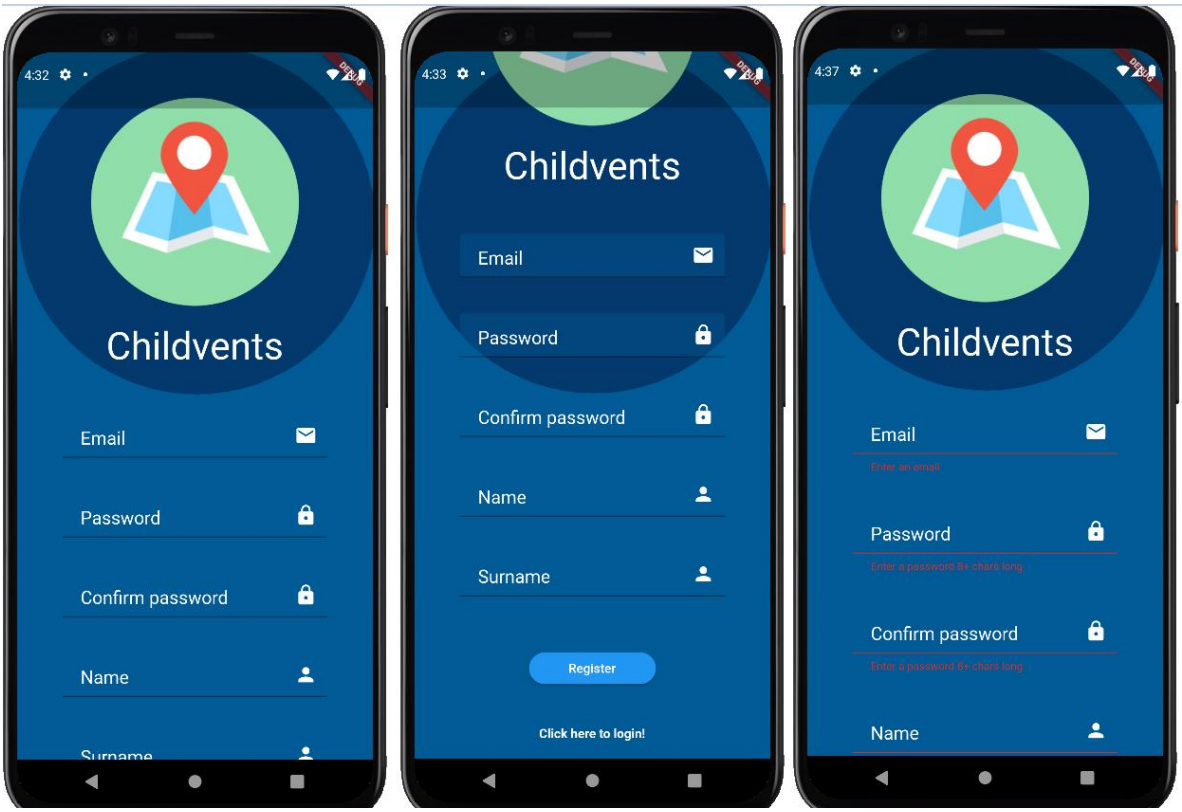
4.1. Ekran logowania

Podstawową funkcją aplikacji jest możliwość logowania się do niej. Na pierwszym planie widnieje logo z nazwą aplikacji oraz pola do wprowadzania danych. Po wciśnięciu przycisku *„Login”* sprawdzana jest poprawność pól. Odpowiednio jest to użycie wyrażenia regularnego do sprawdzenia e-maila i przeliczenie długości hasła. W przypadku wprowadzania hasła, wpisywane dane są ukryte. Po sprawdzeniu walidacji pól, aplikacja łączy się z Firebase i sprawdza czy podane dane istnieją już w systemie [15]. W przypadku, kiedy autoryzacja Firebase będzie odrzucona, pojawi się wyraźny komunikat (nad przyciskiem *„Login”*). Ostatnim elementem użytkowym na ekranie (poniżej przycisku *„Login”*) jest przycisk przejścia do rejestracji konta (rysunek 4.1).

Rysunek 4.1. Ekrany logowania przed i po walidacji [opracowanie własne]

4.2. Ekran rejestracji

Ekran rejestracji jest wykonany w bardzo podobnym stylu jak ekran logowania (rysunek 4.2). W tym przypadku przewidziano dużo więcej pól do wprowadzenia danych. Oprócz podstawowych pól (tj. hasło oraz e-mail), ekran ten posiada dodatkowo: potwierdzenie hasła, imię oraz nazwisko. Walidacja w przypadku dwóch pierwszych pól jest taka sama jak wcześniej. Drugie hasło musi być identyczne jak pierwsze. W przypadku imienia oraz nazwiska ciężko jest dokonać walidacji, dlatego w pola te muszą zostać wprowadzone jakiekolwiek dane. Po wciśnięciu przycisku *„Register”* następuje sprawdzenie wprowadzonych informacji. W momencie, kiedy dane są zgodne, aplikacja łączy się z Firebase (podobnie jak podczas logowania) i przeszukuje czy w bazie danych użyto wcześniej wpisanego adresu e-mail. Jeśli istnieje użytkownik posługujący się takim samym e-mailem, aplikacja natychmiast o tym informuje. W innym przypadku „haszuje” hasło i zapisuje dane do tabeli.

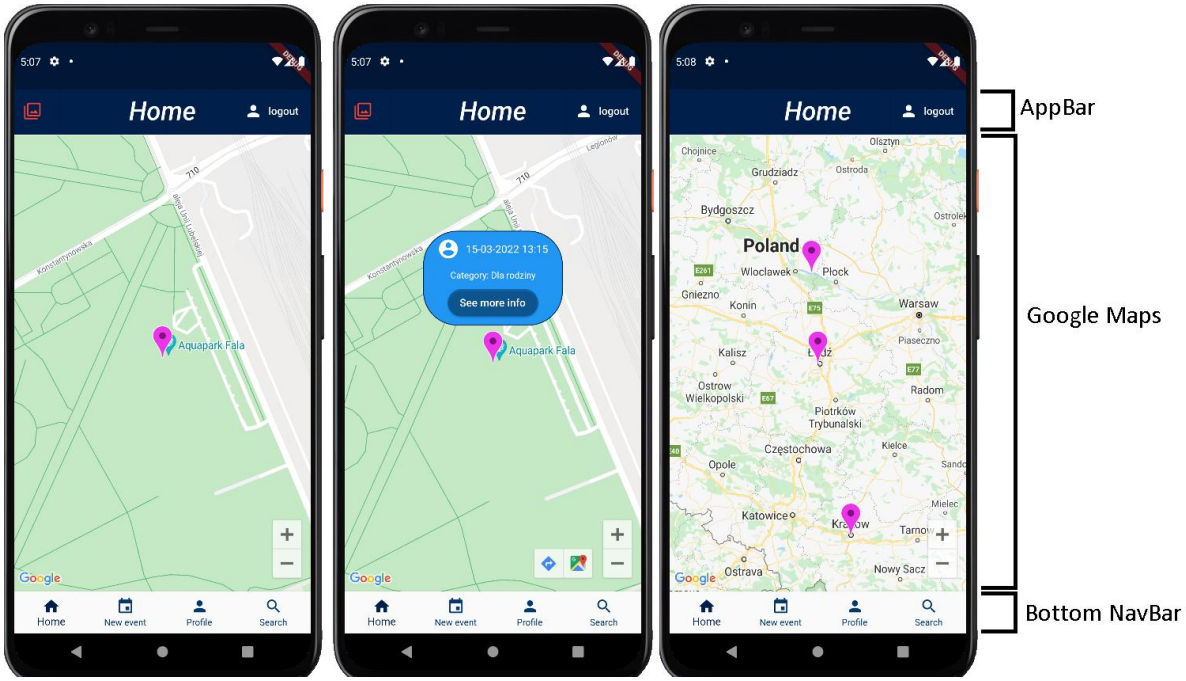


Rysunek 4.2. Ekrany rejestracji przed i po walidacji [opracowanie własne]

Warto wspomnieć w tym miejscu o metodzie odzyskiwania hasła. Firebase nie przechowuje go w formie „surowej", jest ono zahaszowane, więc jedyną opcją jest jego resetowanie. Zapewnia to bezpieczeństwo danych użytkowników, ponieważ w przypadku wykradnięcia danych logowania, złodziej nie może ich wykorzystać („haszem" się nie zaloguje). Ostatnim elementem jest przycisk powrotu do ekranu logowania.

4.3. Ekran główny (Home)

Przed omówieniem ekranu głównego, warto wspomnieć o dwóch elementach aplikacji. Są nimi: *„Provider"* oraz *„Wrapper"*. Pierwszy odpowiada za odpowiedni status aplikacji. Sprawdzany jest stan zalogowania. W ten sposób, po zamknięciu aplikacji i ponownym jej uruchomieniu, nie będzie potrzeby ponownego wprowadzania danych logowania. Drugi element jest klasą, odpowiedzialną za nawigację w obrębie dwóch ekranów (logowania i Home'a). W zależności od statusu przenosi użytkownika do odpowiedniego ekranu.



Rysunek 4.3. Ekrany Home'a przedstawiające znaczniki na Mapie Google'a [opracowanie własne]

Ekran główny aplikacji podzielony jest na trzy sekcje (rysunek 4.3):

- Pasek tytułowy *„AppBar”*,
- Mapę Google z niestandardowymi znacznikami,
- Pasek nawigacji *„BottomNavBar”* lub *„NavBar”*.

Pasek tytułowy posiada zawsze co najmniej jeden widoczny przycisk - *„logout”*, służący do wylogowywania się. Drugi, czerwony, pojawiający się po lewej stronie, jest widoczny tylko w sytuacji, kiedy są aktywowane filtry nakładane na mapę. Po kliknięciu w czerwony przycisk filtry znikają i widoczne są wszystkie znaczniki odpowiadające tematowi wyszukiwania.

Mapa Google posiada niestandardowe fioletowe znaczniki. Po kliknięciu na jeden z nich pojawia się niebieskie okno z podstawowymi informacjami (datą oraz kategorią wydarzenia). Okienko posiada także przycisk służący do przejścia w detale danego wydarzenia. Mapę można dowolnie oddalać i przybliżać.

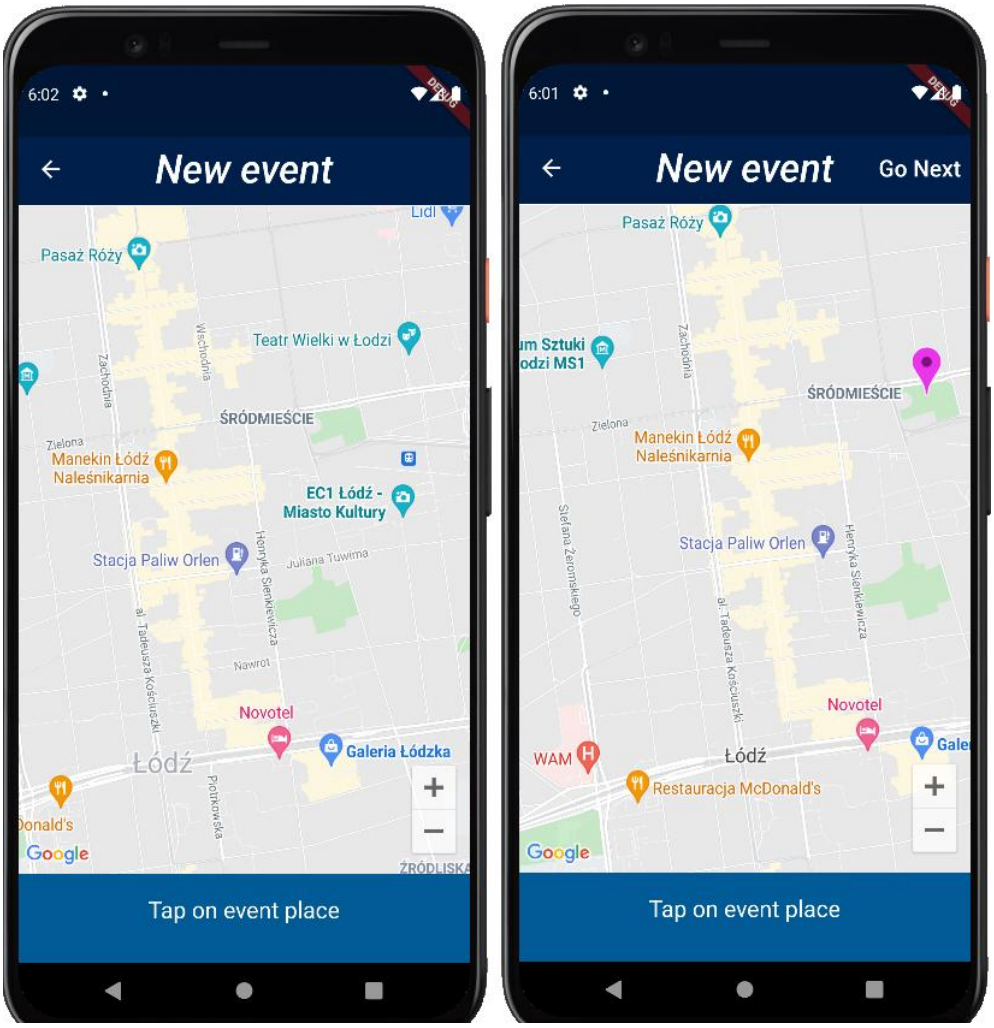
Ostatnim elementem ekranu głównego aplikacji jest pasek nawigacji. Za jego pomocą można przemieszczać się w wygodny sposób. Odpowiednio (od lewej strony):

- do ekranu głównego,
- do tworzenia nowego wydarzenia,
- do profilu użytkownika,
- do menu wyszukiwania.

Warto wspomnieć, że „*Bottom NavBar*” oraz „*AppBar*” będzie się pojawiał jeszcze w kilku miejscach aplikacji.

4.4. Tworzenie nowego wydarzenia

W aplikacji można wyróżnić dwa etapy tworzenia nowego wydarzenia: dodanie znacznika na mapie oraz wypełnienie formularza (rysunek 4.4).

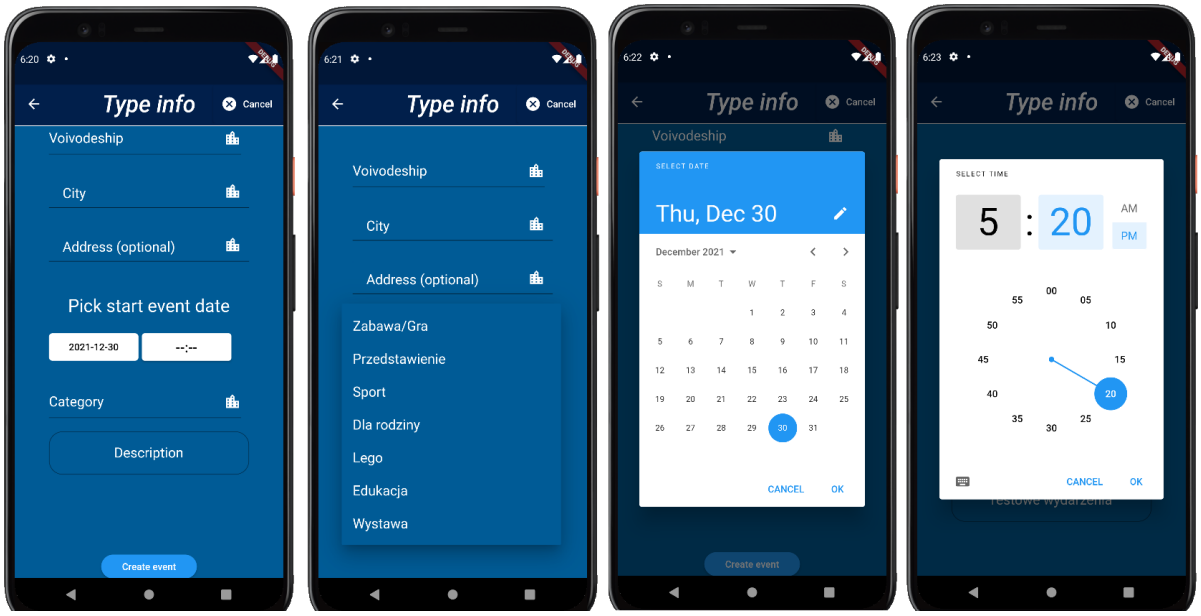


Rysunek 4.4. Dodanie znacznika na mapę [opracowanie własne]

Ekran w bardzo prosty sposób ma umożliwić użytkownikowi dodanie dokładnego położenia na mapie. Pasek tytułowy w tym przypadku posiada dwa przyciski. Przycisk powrotu (po lewej stronie) oraz przycisk *„Go next”*, wyświetlany dopiero po umieszczeniu znacznika na mapie, nawigujący do formularza wydarzenia.

4.5. Formularz nowego wydarzenia

W przypadku formularza nowego wydarzenia, pasek tytułowy pozwala na cofnięcie się do wyboru miejsca lub anulowanie całości.



Rysunek 4.5. Przykładowe widoki formularza nowego wydarzenia [opracowanie własne]

Sam ekran posiada funkcję przewijania, co pozwala na dostosowanie go do różnej wielkości ekranów telefonów. Poszczególne pola mają osobną walidację i specjalne cechy (rysunek 4.5).

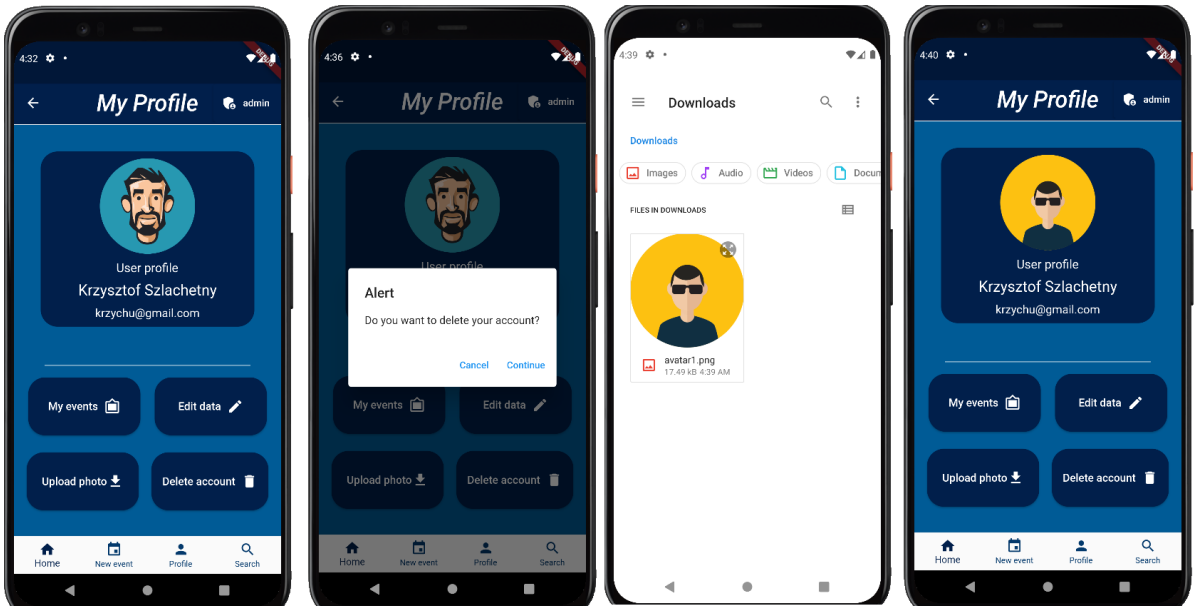
Tabela 4.1. Opis poszczególnych pól formularza [opracowanie własne]

Pole	Funkcja pola	Walidacja
Voivodeship	Wybór województwa z listy województw (pobieranej z Firebase)	Czy dokonano wyboru?
City	Wprowadzenie nazwy miasta	Czy wpisano?
Address	Wprowadzenie adresu wydarzenia (opcjonalnie)	Czy wpisano?
Data	Wybranie z kalendarza daty	Czy wybrano datę, która jest dzisiejsza lub przyszła?
Czas	Wybranie z zegara dokładnej godziny wydarzenia	Czy wprowadzono?
Category	Wysuwana lista kategorii (pobierana z Firebase)	Czy dokonano wyboru?
Description	Wprowadzenie opisu wydarzenia	Czy wpisano?

Po wciśnięciu przycisku, następuje walidacja danych i wyświetlenie odpowiedniego komunikatu (w przypadku błędu). Przy pomyślnej weryfikacji, dodawane jest wydarzenie do Firebase, skąd od tego momentu inne urządzenia będą pobierać je i nakładać na swoją mapę wydarzeń.

4.6. Profil użytkownika

Profil użytkownika jest jednym z bardziej zaawansowanych elementów aplikacji. Posiada wiele przycisków do nawigacji pomiędzy różnymi ekranami, ogromną ilość funkcji oraz danych pobieranych z bazy danych (rysunek 4.6).

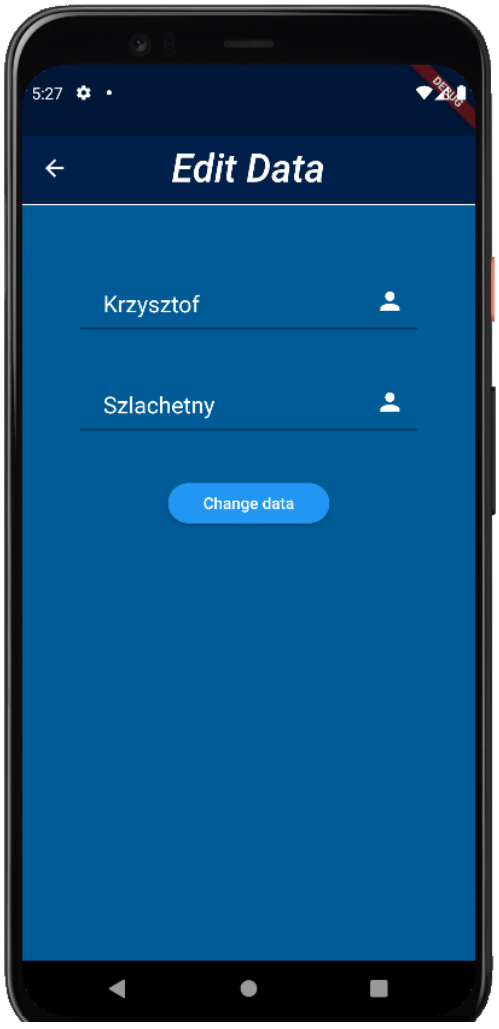


Rysunek 4.6. Widok profilu, dodawania nowego awatara i usuwania konta [opracowanie własne]

Pasek tytułowy wyposażony jest w standardowy element do cofania się w aplikacji oraz ukryty przycisk dostępny tylko dla osób z rolą administratora, nawigujący do panelu. Widoczne w największym oknie dane pobierane są z Firebase'a, natomiast awatar użytkownika z Storage'a.

Poniżej okna z danymi użytkownika umieszczono 4 przyciski. Dają one możliwość dokonania przeglądu śledzonych wydarzeń, zmiany danych lub awatara oraz usunięcia bezpowrotnie konta. Usuwanie konta odbywa się w prosty sposób poprzez potwierdzenie wyświetlającego się komunikatu.

W przypadku zmiany awatara, konieczne jest dokonanie wyboru zdjęcia z galerii smartfona. W kolejnym kroku następuje zmiana nazwy zdjęcia (nowa nazwa to ID użytkownika w bazie danych oraz rozszerzenie pliku graficznego), a następnie usunięcie starego awatara z Firebase Storage'a (jeśli już taki istnieje). Od tej pory przy każdorazowym wejściu do profilu wyświetlany będzie nowy, zaktualizowany awatar.

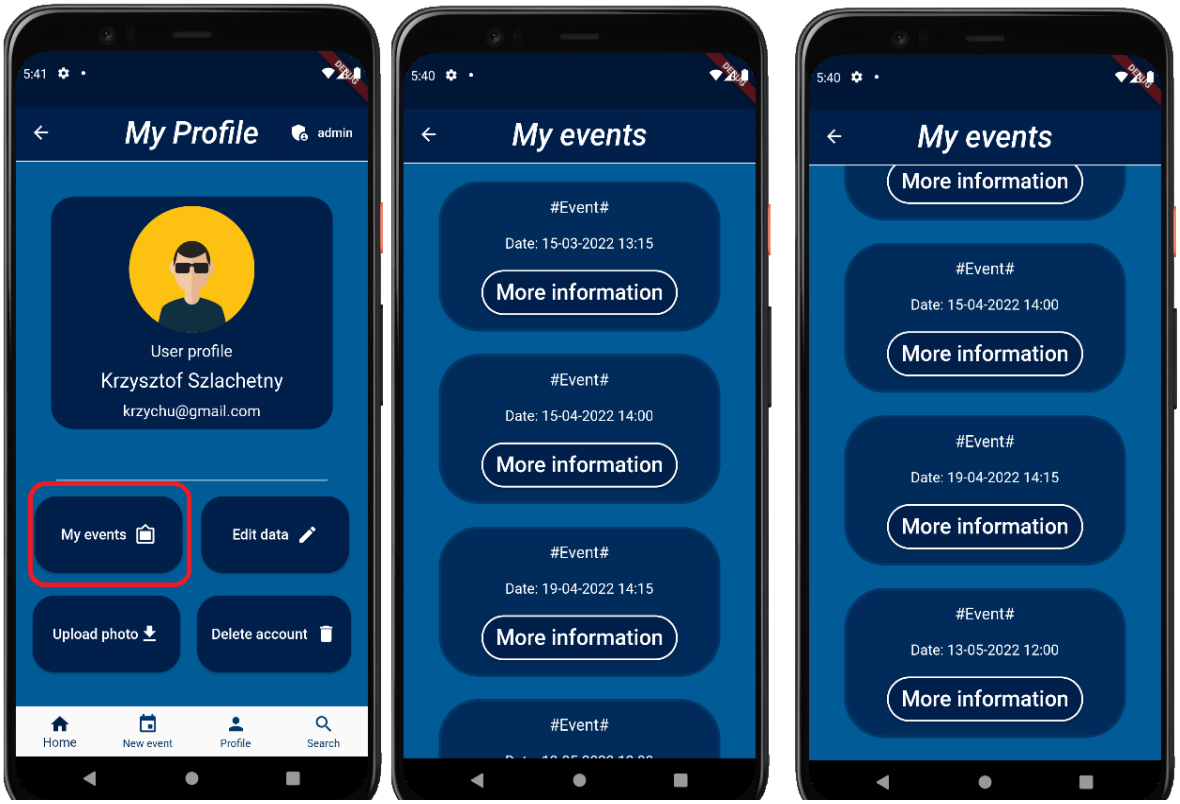


Rysunek 4.7. Ekran edycji imienia oraz nazwiska w aplikacji [opracowanie własne]

Bardzo prostym ekranem w aplikacji z dużą możliwością rozwoju jest zmiana danych. W obecnym stanie aplikacja pozwala na edycję imienia oraz nazwiska (rysunek 4.7). W przyszłości, w zależności od potrzeb, istnieje możliwość rozszerzenia opracowania i wprowadzenie m.in. resetowania hasła, zmiany e-maila i innych danych.

4.7. Moja lista wydarzeń

Do ekranu wydarzeń można dostać się z profilu użytkownika. Można na nim zapoznać się z wszystkimi swoimi wydarzeniami, które będą miały miejsce w przyszłości (starsze, niż obecna data, nie są wyświetlane). Lista jest przewijana, co pozwala na przeglądanie dużej liczby wydarzeń. Każda ramka zawiera datę oraz przycisk przechodzący do detali konkretnego wydarzenia (rysunek 4.8).

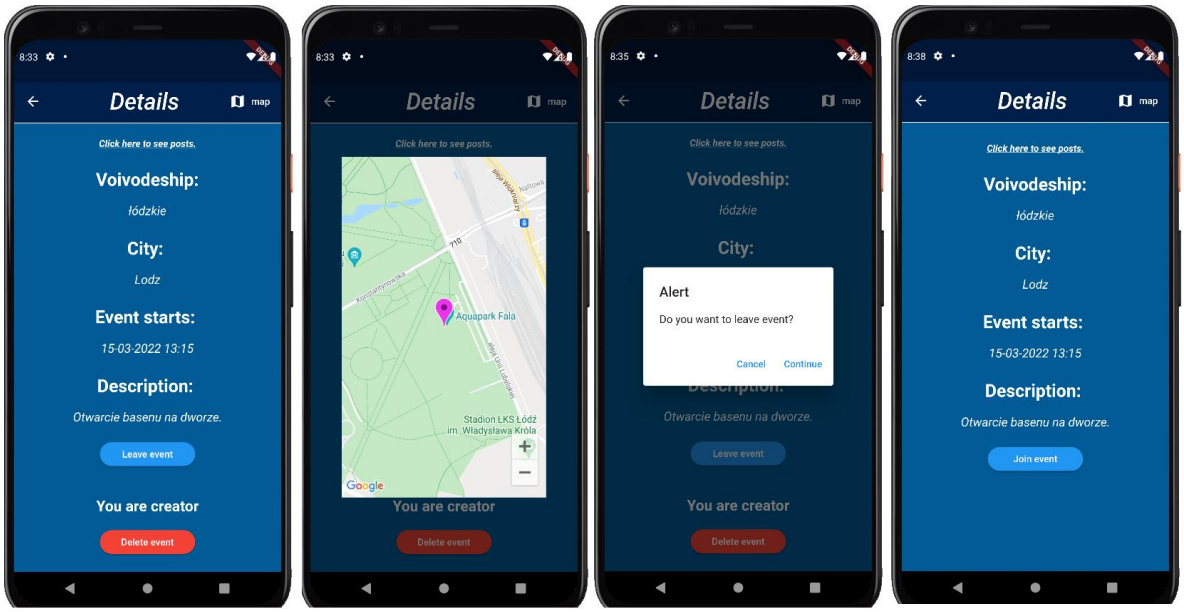


Rysunek 4.8. Ekrany związane z „Moimi wydarzeniami” [opracowanie własne]

4.8. Detale wydarzenia

W zależności od statusu nadanego użytkownikowi wyświetlany jest inny zestaw dostępnych dla niego detali wydarzenia. Są to statusy:

- użytkownik – założyciel,
- użytkownik – standardowy,
- użytkownik, który zgłosił już swój udział w danym wydarzeniu,
- użytkownik, który jeszcze nie dołączył do danego wydarzenia.



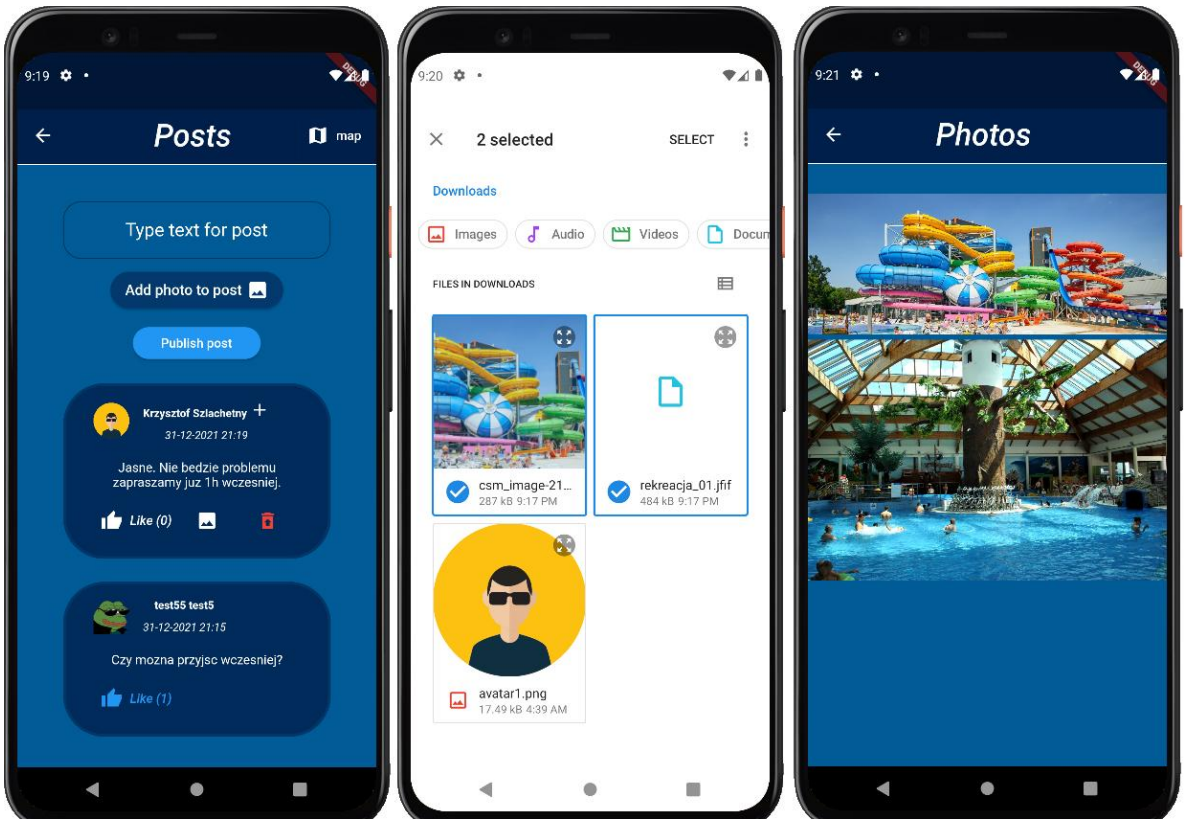
Rysunek 4.9. Ekrany związane z detalami wydarzenia [opracowanie własne]

Użytkownikowi–założycielowi przysługuje prawo usunięcia danego wydarzenia, w konsekwencji czego z bazy danych będą wyczyszczone wszystkie związane z nim posty, zdjęcia oraz uczestnicy. Użytkownik standardowy nie ma takiej możliwości (dla niego ten przycisk jest ukryty). W zależności od przysługującego statusu użytkownik może dołączyć do wydarzenia lub je opuścić. Jeżeli osoba dołączy do grona uczestników, dane wydarzenie będzie się jej wyświetlało po kliknięciu okna „Moje wydarzenia” w profilu użytkownika. Wszystkie elementy wyświetlane na ekranie, pobierane są z Firebase. Bardzo użyteczny jest podgląd wydarzenia na mapie (długość i szerokość geograficzna nakładane są na mapę) w momencie kliknięcia przycisku. W każdej chwili użytkownik ma możliwość dokonania przeglądu postów związanych z danym wydarzeniem (rysunek 4.9).

4.9. Posty

Postowanie jest metodą komunikowania się użytkowników w obrębie konkretnego wydarzenia. Każda osoba (niezależnie od statusu) może dodać swój własny post oraz „polubić” post innego użytkownika aplikacji. W przypadku dodawania postu ze zdjęciami, w pierwszej kolejności konieczne jest kliknięcie przycisku i wybranie zdjęcia z galerii swojego smartfona. Każdy post zawiera informacje o osobie, która go dodała, datę i godzinę oraz obecny awatar użytkownika. Ekran daje możliwość przewijania

i przeglądania starszych postów, gdyż są one wyświetlane w kolejności od najnowszego do najstarszego. Założyciel wydarzenia oznaczony jest charakterystycznym symbolem plusa, pojawiającym się przy jego nazwisku. Twórca posta może go w dowolnym momencie usunąć, klikając w czerwoną ikonkę „kosza", nie ma jednak możliwości usunięcia postów zamieszczonych przez innego użytkownika.

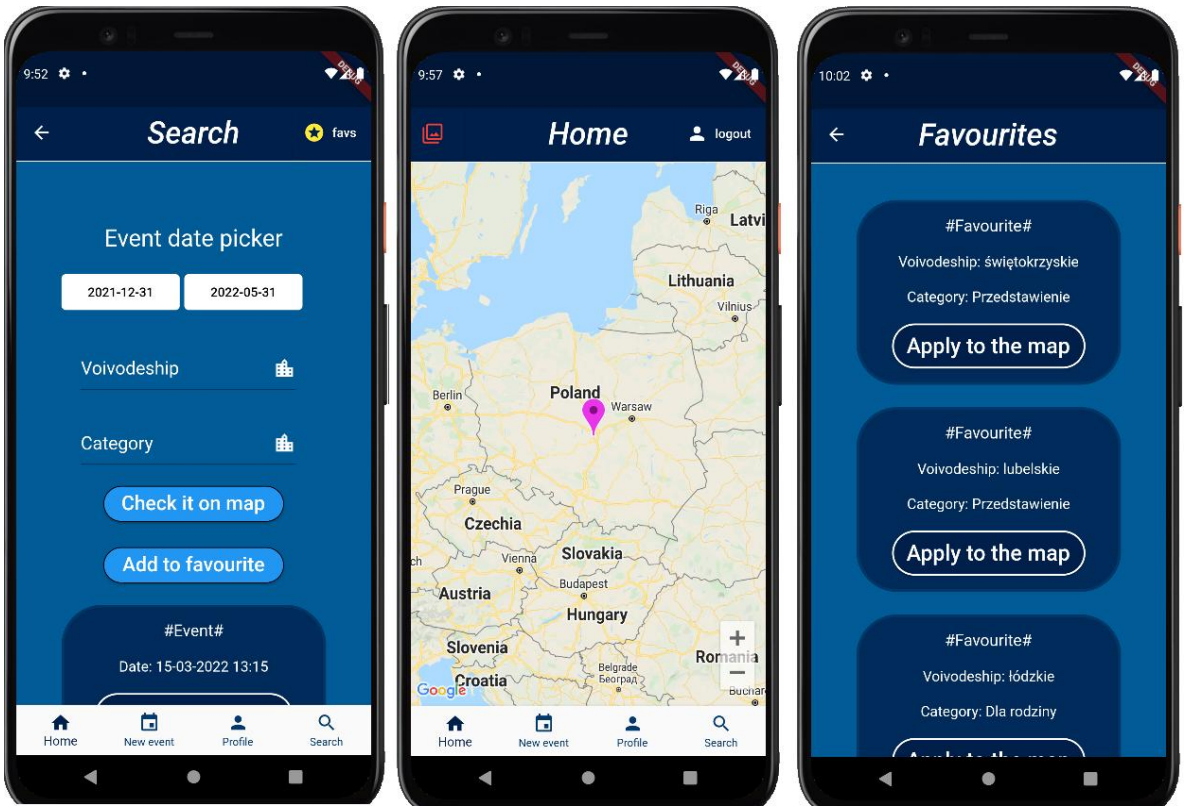


Rysunek 4.10. Postowanie oraz galeria [opracowanie własne]

O załączeniu do posta zdjęć informuje ikonka obrazka, widoczna tylko w przypadku, kiedy są one faktycznie dodane (rysunek 4.10). Po wciśnięciu ikony aplikacja przenosi użytkownika do kolejnego ekranu, w którym zdjęcia są wyświetlane na przewijanej liście. Każdy post zawierający zdjęcia, otrzymuje również adresowaną indywidualnie ścieżkę w Firebase Storage.

4.10. Wyszukiwarka

Użytkownik ma nieograniczoną możliwość filtrowania wydarzeń. Filtr można ustawić na konkretną datę lub zakres dat, wybraną kategorię albo województwo (rysunek 4.11).

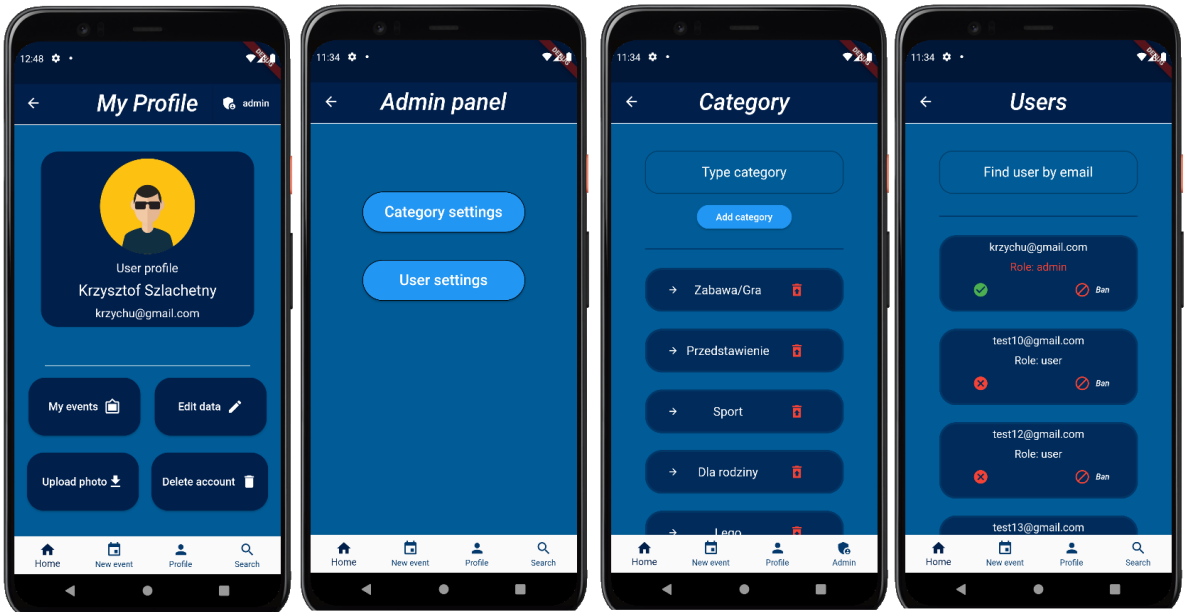


Rysunek 4.11. Wyszukiwarka, filtr na mapie oraz ulubione wyszukiwania [opracowanie własne]

Lista wydarzeń jest automatycznie odświeżana po nałożeniu filtra. Użytkownik, w zależności od preferencji, może zobaczyć wyniki wyszukiwania sortowane chronologicznie wg daty lub prześledzić wydarzenia bezpośrednio na mapie. Jak wspomniano w rozdziale 4.3, czerwony przycisk na pasku tytułowym analogicznie służy usunięciu filtrów z mapy. W każdej chwili użytkownik może dodać konkretne filtry do zakładki „ulubionych". Wszystkie ulubione wyszukiwania są dostępne po kliknięciu przycisku „*favs*", skąd można je nanieść na mapę.

4.11. Administrator i jego funkcje

Jak wspomniano przy opisie profilu użytkownika, przyznana rola administratora umożliwia przejście do panelu administratora poprzez odpowiedni przycisk dający wybór dowolnej opcji. W obecnym stanie pracy istnieje możliwość dodawania lub usuwania kategorii oraz zarządzania wszystkimi dodanymi użytkownikami (rysunek 4.12). W przypadku dodawania, administrator otrzymuje pole do wpisywania nazwy kategorii oraz przycisk dodający ją do bazy danych. Usunięcie następuje po kliknięciu czerwonego przycisku „kosza”.



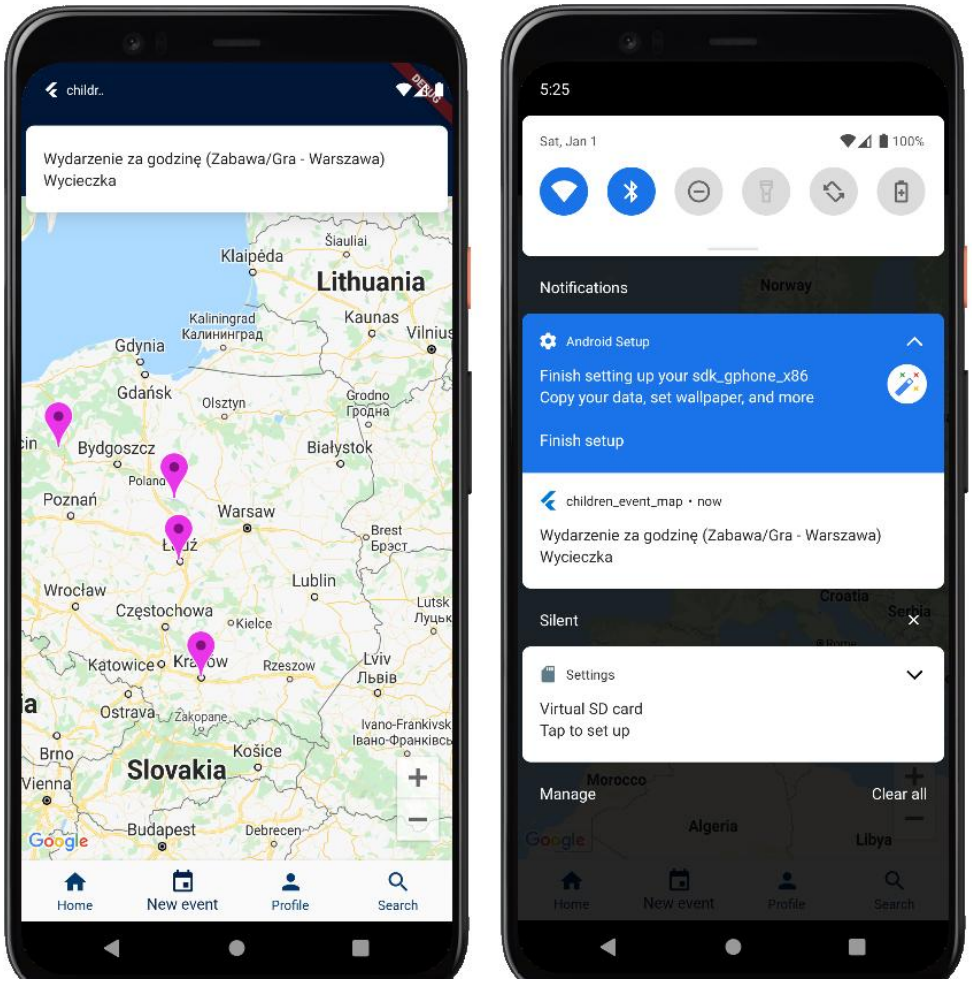
Rysunek 4.12. Panel administratora i jego zawartość [opracowanie własne]

Nieco bardziej złożone jest zarządzanie użytkownikami. Administrator w każdej chwili może dowolnej osobie zmienić rolę ze zwykłego użytkownika na administratora i odwrotnie poprzez zastosowanie odpowiedniego znacznika. Czerwony znacznik „X” to odpowiednik zwykłego użytkownika, natomiast zielony znacznik „V” - przydzielona rola administratora. Odnalezienie konkretnej osoby ułatwia wyszukiwarka, która wraz z dodawanymi kolejno literami, filtruje wszystko w czasie rzeczywistym.

Administrator posiada pełne uprawnienia, m.in. w dowolnym momencie ma możliwość bezpowrotnego usunięcia konta każdego z użytkowników.

4.12. Powiadomienia o nadchodzącym wydarzeniu

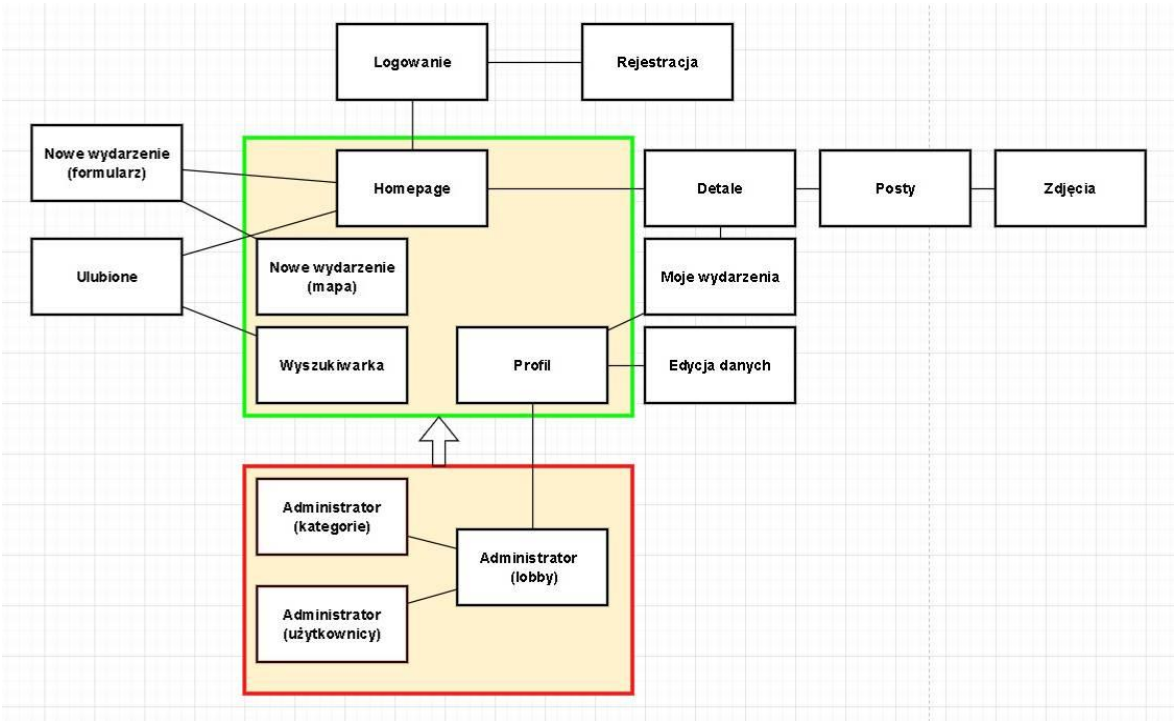
Użytkownik otrzymuje powiadomienia jedną godzinę przed planowanym terminem rozpoczęcia wydarzenia. W przypomnieniu wyświetla się miasto, kategoria wydarzenia oraz jego cały opis (rysunek 4.13).



Rysunek 4.13. Przychodzące powiadomienie [opracowanie własne]

4.13. Nawigacja w aplikacji

Nieodłącznym elementem każdej aplikacji mobilnej musi być odpowiednia nawigacja. Powinna ona być możliwie jak najbardziej intuicyjna, pozwalająca użytkownikowi na mało skomplikowane i szybkie dotarcie do interesujących go elementów aplikacji. W przypadku omawianej aplikacji dobrym rozwiązaniem jest zastosowanie dolnego paska nawigacyjnego, który umożliwia szybkie dotarcie do podstawowych ekranów po jednym kliknięciu w odpowiednią ikonkę. W skład tych ekranów wchodzi: ekran główny, tworzenie nowego wydarzenia, profil oraz wyszukiwarka. Dodatkowo w aplikacji po każdym wybranym ekranie zawsze istnieje funkcja cofania (nawigacja działa dwustronnie). Na schemacie (rysunek 4.14) przedstawiono wszystkie ekrany oraz możliwe przejścia pomiędzy nimi.



Rysunek 4.14. Schemat nawigacji w aplikacji [opracowanie własne]

Sektor wyróżniony na schemacie zieloną ramką symbolizuje „*BottomNavBar*”. W jego obrębie istnieje możliwość obustronnego przechodzenia pomiędzy wyszczególnionymi ekranami. Dla ekranów zaznaczonych czerwoną obramówką działa wyłącznie jednostronna nawigacja, czyli można z nich dostać się do dowolnego innego elementu znajdującego się w sektorze „*BottomNavBar*”.

5. Wybrane funkcjonalności systemu

Każdy ekran aplikacji jest inny, jednak wszystkie tworzone są z pewnych klas obiektów, które zapewniają konkretny wygląd. Programowanie ich jest rzeczą dość schematyczną. W poniższym rozdziale zostaną opisane wybrane funkcjonalności systemu, które są jednymi z ciekawszych i nie pojawiają się w kodzie programu zbyt często.

5.1. Opis techniczny StreamBuilder

Podstawowym elementem we Flutterze i nieodłącznym elementem projektu było korzystanie z tzw. StreamBuildera. Trzeba skorzystać z niego w każdym przypadku, w którym istnieje potrzeba wykorzystania informacji zawartych w bazie danych Firebase. Pozwala on na przesłanie do bazy danych konkretnego zapytania. Zaletą StreamBuildera jest odbudowa ekranu przy zmianie danych w Firebase. Oznacza to, że w przypadku dodania przez użytkownika A nowego elementu, użytkownik B będzie widział go od razu. Poniżej omówiono przykład zastosowania.

```
55                StreamBuilder(
56                  stream: DatabaseService(uid: '')
57                      .favouriteCollection
58                      .where('user_id',
59                          isEqualTo: FirebaseAuth.instance.currentUser!.uid)
60                      .snapshots(),
61                  builder: (BuildContext context,
62                      AsyncSnapshot<QuerySnapshot> snapshot) {
63                    if (!snapshot.hasData) {
64                      return Center(
65                        child: CircularProgressIndicator(),
66                      ); // Center
67                    }
68                    print(snapshot.data!.docs.length);
69
70                    return ListView(
```

Rysunek 5.1. Fragment kodu dotyczący StreamBuildera [opracowanie własne]

Przedstawiony na rysunku 5.1 fragment kodu dotyczy wyświetlania ulubionych wyszukiwań. Można go podzielić na dwa elementy składowe:

- *stream* – element zawarty w liniach 56-60, który dotyczy budowania strumienia, czyli zapytania do bazy danych. W tym przypadku zastosowano odwołanie do Firebase, następnie do kolekcji *„favourites”*, by wreszcie wyszukać rekordy pokrywające się polem *„user_id”* z ID obecnie zalogowanego użytkownika;
- *builder* – budulec (linie 61-70 i kolejne), czyli element odpowiedzialny za przyjmowanie danych oraz budowanie z nich wyglądu ekranu. W przypadku, kiedy nie ma jeszcze danych, wyświetlane jest kręcące się kółko (linia 65). W każdym innym tworzona jest przewijana lista, która pokazuje na ekranie zdefiniowany wygląd.

5.2. Ukrywanie zawartości

Czasem łatwiej jest zaprojektować ukrycie danego elementu niż zaprogramować wyjątek, w którym dany element nie występuje. Motyw ukrywania zawartości pojawia się dość często w kodzie programu. Na rysunku 5.2 przedstawiono przykład zastosowania.

```
499                                   Visibility(
500                                     visible: document['photo'] == true,
501                                     child: Align(
502                                       alignment: Alignment.centerRight,
503                                       child: TextButton.icon(
504                                         onPressed: () async {
505                                           final destination =
506                                               "${widget.event_id}/${document.id}/";
507                                           Navigator.push(
508                                             context,
509                                             MaterialPageRoute(
510                                               builder: (context) =>
511                                                   EventPhotos(
512                                                       event_id:
513                                                           widget.event_id,
514                                                       post_id: document.id,
515                                                       destination:
516                                                           destination), // EventPhotos
517                                             ), // MaterialPageRoute
518                                           );
519                                         },
```

Rysunek 5.2. Ukrywanie ikonki zdjęć w ekranie postów [opracowanie własne]

Element „*Visibility*” jest przydatny w sytuacji wyświetlania przycisku przejścia do zdjęć w ekranie postów. Jeśli dany post nie posiada zdjęć, przycisk ten jest ukrywany (linia 500). W innym przypadku jest on wyświetlony i po kliknięciu przechodzi do ekranu zdjęć, zabierając ze sobą kilka zmiennych, które będą tam przydatne (linie 507-517). Motyw ukrywania będzie pojawiał się w przypadku usuwania postów, wyświetlania przycisku przejścia do panelu administratora (profil) oraz przy tworzeniu nowego wydarzenia (postawienie znacznika na mapie sprawia, że przycisk „*Go next*” się pojawia).

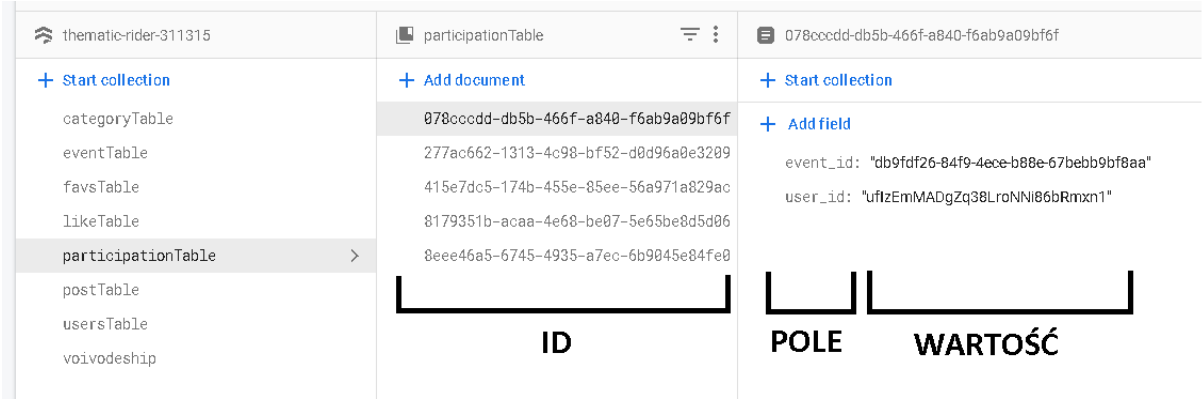
5.3. Zapytania do bazy danych

Osobnym elementem aplikacji są zapytania do bazy danych. W rozdziale 5.1 wspomniano, że StreamBuildery budują wygląd aplikacji (działają w jednym kierunku Firebase –> aplikacja). Konieczne staje się zaprogramowanie komunikacji w drugim kierunku, tak aby baza danych mogła być uzupełniana o nowe treści. W obecnym stanie aplikacja zawiera 22 zapytania do podstawowej bazy danych, wzbogacone o funkcje autoryzacji użytkownika oraz 7 zapytań odnoszących się do Firebase Storage.

Zapytania do podstawowej bazy danych, można podzielić na 4 grupy:

- ustawiające („*set*”) – zapytanie ustawia dane pola z podaną informacją,
- usuwające („*delete*”) – zapytanie usuwa dany dokument,
- modyfikujące („*modify*”) – zapytanie działa na podobnej zasadzie, co „*set*”, jednak można wyłącznie zmodyfikować dane pola, a nie ustawiać je od nowa,
- pobierające („*get*”) – zapytanie zwraca konkretne dane.

Na rysunku 5.3 zaprezentowano przykładowy element znajdujący się w bazie danych.



Rysunek 5.3. Przykładowy element bazy danych [opracowanie własne]

Każdy element w Firebase posiada trzy wartości:

- *ID* – identyfikator dokumentu,
- *Pole* – nazwa pola,
- *Wartość* – wartość przypisana do konkretnego pola.

```
145  Future deletePost(String post_id) async {
146    var result =
147        await likesCollection.where('post_id', isEqualTo: post_id).get();
148    for (var res in result.docs) {
149      print(res.id);
150      await likesCollection.doc(res.id).delete();
151    }
152    return await postsCollection.doc(post_id).delete();
153  }
154
```

Rysunek 5.4. Przykładowy fragment kodu [opracowanie własne]

Przedstawiony fragment kodu (rysunek 5.4) usuwa dany post oraz związane z nim „polubienia" z bazy danych. „Post_id" zostaje przekazany do funkcji, ponieważ łączy kolekcję „polubień" i postów (linia 145). Wyszukane zostają wszystkie „polubienia" związane z danym postem i zapisane do zmiennej (linie 146-147). Następuje przejście po każdym „polubieniu" oraz jego późniejsze usunięcie z bazy danych (linie 148-151). Ostatnia, 152. linia kasuje dany post.

Większość zapytań przyjmuje określoną formułę:

- *kolekcja* – wskazanie kolekcji, z którą będzie podjęte dane zapytanie,
- „*doc*" lub „*where*" – określenie czy odwołanie dotyczy identyfikatora dokumentu („*doc*") czy wartości konkretnego pola („*where*"),
- *zapytanie* – określenie, jakiego typu zadanie będzie wykonane (ustawiające, usuwające, modyfikujące czy pobierające).

6. Wykorzystane biblioteki (packages)

W wielu projektach nie da się zrezygnować z załączenia do nich tzw. bibliotek. Dodają one często funkcjonalności, wspierają różne wersje i implementacje oraz pozwalają na skomunikowanie się z innym oprogramowaniem. Niniejsza praca nie stanowi wyjątku od reguły – obok standardowych zastosowano tutaj wiele dodatkowych bibliotek.

Przy tworzeniu pracy wykorzystano 13 dodatkowych bibliotek. Są to:

- *firebase_auth* (wersja 3.1.3) – służąca do logowania, autoryzacji Firebase'a. Dzięki tej bibliotece można w projekcie użyć Firebase Authentication API. Dodatek ten daje możliwość komunikacji telefon-Firebase o charakterze autoryzacyjnym;
- *cloud_firestore* (wersja 2.5.3) – konieczna do korzystania z bazy danych. Podstawowa biblioteka, która hostuje w chmurze bazę danych oraz daje bezpośredni dostęp do niej urządzeniom z zainstalowaną aplikacją. Bez niej nie można byłoby tworzyć zapytań do bazy danych;
- *provider* (wersja 6.0.1) – dołączająca provider, o którym wspomniano w rozdziale 4.3;
- *firebase_core* (wersja 1.10.0) – łącząca aplikację Fluttera z projektem Firebase. Zapewnia w projekcie wszystkie akcje związane z zapytaniami oraz pobieraniem danych z bazy danych;
- *google_maps_flutter* (wersja 2.1.1) – umożliwiająca korzystanie z map Google. Pozwala na wyświetlanie mapy m.in.: w Home oraz przy detalach wydarzenia;
- *geolocator* (wersja 7.7.1) – ułatwiająca korzystanie z usług geolokalizacji. Daje możliwość pobierania oraz przetwarzania danych lokalizacyjnych telefonu. Używana jest szczególnie przy dodawaniu znacznika na mapę Google;
- *uuid* (wersja 3.0.5) – zapewniająca tworzenie unikatowych ciągów znaków, przydatnych przy tworzeniu nowego wydarzenia itp. Przed dodaniem do bazy danych system generuje unikalne ID. Prawie wszystkie identyfikatory w kolekcjach bazy danych były tworzone przy udziale tej biblioteki;

- *custom_info_window* (wersja 1.0.1) – umożliwiająca stworzenie własnej ramki (po kliknięciu znacznika na mapie). Dodanie przycisku, konkretnych informacji lub innych komponentów w ramce wydarzenia, nie byłoby możliwe bez użycia tej biblioteki;
- *firebase_storage* (wersja 10.2.0) – dająca możliwość korzystania z Firebase Storage (gdzie przechowywane są awatary oraz zdjęcia wydarzenia). Mechanizm przechowywania oraz pobierania zdjęć został opisany w rozdziale 3.3;
- *file_picker* (wersja 4.2.7) – umożliwiająca wybieranie pliku z pamięci smartfona,
- *image_picker* (wersja 0.8.4+4) – umożliwiająca wybieranie plików graficznych z pamięci smartfona. Biblioteka używana przy wyborze nowego awatara użytkownika lub zdjęć wydarzenia;
- *flutter_local_notification* (wersja 9.1.5) oraz firebase_messaging (wersja 11.2.4) – dające możliwość wysyłania powiadomień. Obie biblioteki odpowiedzialne są za wysyłanie powiadomień. Pierwsza dotyczy lokalnych powiadomień. Do systemu Android dodawany jest „alarm”, określony na konkretną datę i godzinę. Druga biblioteka odpowiada za powiadomienia wysyłane przez Firebase.

7. Możliwości rozwoju

Specyfiką projektów z dziedziny informatyki jest nieskończoność procesu ich tworzenia oraz ich elastyczność i otwartość. Projekty takie nie stanowią nigdy systemów zamkniętych, a ich modyfikacje są naturalnie wymuszone wdrażaniem nowych technologii na rynku sprzętu komputerowego, jak również rosnącymi wymaganiami użytkowników produktów informatycznych w aspekcie zwiększenia możliwości użytkowych i jakościowych istniejących aplikacji, wywierając presję na producentów oprogramowania użytkowego.

Istotnym elementem w cyklu życia projektu informatycznego są również względy ekonomiczne – wymuszone koniecznością sprostania wymogom rosnącej konkurencji na rynku informatycznym, jak również zapotrzebowaniem podmiotów i tendencji występujących na rynku gospodarczym.

Aplikacja, by funkcjonować poprawnie, wymaga przeprowadzania częstych aktualizacji, idących w ślad za kolejnymi, nowymi wersjami systemu operacyjnego oraz Fluttera i powinna nadążać za panującymi trendami.

W obecnych czasach jednym z obserwowanych trendów jest znaczny ruch migracyjny ludności świata. Czynniki społeczno-polityczne (takie, jak: konflikty zbrojne, represje, niebezpieczeństwo zagrożenia zdrowia i życia), czynniki ekonomiczne (np. chęć poprawy warunków bytowych rodziny), a niejednokrotnie nawet czynniki środowiskowe (m.in. ucieczka przed skutkami klęsk żywiołowych), zmuszają znaczny odsetek mieszkańców naszego globu do czasowych lub trwałych migracji w poszukiwaniu lepszego życia poza ojczyzną. Bardziej krótkotrwałe są podróże w celach turystycznych czy leczniczych. Wszystko to sprawia, że w konsekwencji jeden obszar geograficzny może zamieszkiwać wiele narodowości odmiennych językowo i kulturowo. Bez wątpienia każda projektowana aplikacja powinna uwzględniać wielojęzyczność jej użytkowników. Funkcja ta jest stosunkowo prosta do zaimplementowania, a znacząco zwiększa liczbę odbiorców danej aplikacji i poprawia komfort korzystania z niej.

Skoro aplikacja powinna docierać do szerszego grona użytkowników, celowym byłoby wsparcie systemu operacyjnego iOS, a tym samym telefonów firmy Apple. W ten sposób z aplikacji mogłoby korzystać aż 99% użytkowanych smartfonów.

Niewątpliwie bardzo ważna w każdej aplikacji jest rola administratora. Powinien on mieć nieograniczoną możliwość prowadzenia monitoringu wprowadzanych elementów, moderowania oraz usuwania treści zbędnych lub niezgodnych z regulaminem. Warto byłoby dodać tej roli więcej funkcjonalności.

Rozszerzeniem funkcjonalności aplikacji bez wątpienia mogłaby stać się opcja dodawania filmów do postów oraz ich przechowywanie w bazie. Sam ekran dedykowany zdjęciom wymagałby zmian graficznych poprawiających jego estetykę, z dodatkowym wprowadzeniem możliwości powiększania zdjęć oraz opcji przesuwania w lewo lub w prawo.

Utrudnieniem dla nowych użytkowników zainteresowanych aplikacją jest konieczność zakładania kont oraz potwiedzania ich mailowo. Korzystne byłoby uzupełnienie aplikacji o nową rolę, jaką jest „gość", dzięki której użytkownik mógłby przeglądać interesujące go wydarzenia i posty bez możliwości dodawania elementów i ich śledzenia. Wskazane byłoby poprawienie interaktywności aplikacji z użytkownikiem. Aplikacja powinna być przejrzysta i czytelna dla nowej osoby. Można to uzyskać np. poprzez zwiększenie liczby pojawiających się komunikatów. Warto pamiętać również o tym, by nawigacja była stosunkowo „płytka". Nie powinno projektować się ekranów bardzo oddalonych od ekranów głównych.

Kolejną propozycją ulepszenia aplikacji może być zaimplementowanie mechanizmu blokującego niepotrzebny spam. W obecnym stanie aplikacji jeden użytkownik może zamieścić nieograniczoną liczbę nowych wydarzeń oraz dodać mnóstwo postów. Gdyby użytkownik stał się komputerowo zaprogramowanym botem, mógłby spowodować obciążenie całego systemu, a jednocześnie promować swój produkt w postach dotyczących różnych wydarzeń. Takie praktyki nie są pożądane, gdyż sam administrator nie nadążałby z usuwaniem zbędnych treści.

Mechanizmem godnym zaimplementowania jest dodanie funkcji wysyłania prywatnych wiadomości oraz dodawania innych użytkowników do grona znajomych. Dzięki temu zaistniałaby m.in. możliwość zacieśniania więzi międzyludzkich i tworzenia

bardziej zżytej społeczności. Ostatnim elementem przydatnym dla użytkownika, a jednocześnie wzbogacającym aplikację może być funkcja śledzenia danej kategorii, osoby lub miejsca. Użytkownik poprzez powiadomienie mógłby otrzymywać aktualne informacje zawierające treści, które go interesują, bez straty czasu na samodzielne wyszukiwanie nowości.

Należy zdawać sobie sprawę z tego, że zamieszczone powyżej propozycje rozszerzenia i ulepszenia aplikacji, stanowią nadal otwarty katalog, w którym – w zależności od oczekiwań użytkowników – będą mogły pojawiać się kolejne całkiem nowe funkcje.

8. Podsumowanie

Opracowany w ramach pracy inżynierskiej system umożliwiający przeglądanie wydarzeń dla dzieci na mapie odpowiada początkowym założeniom projektowym.

Najważniejszym zamysłem pracy było zaprojektowanie aplikacji w taki sposób, aby wszystkie jej elementy były przejrzyste oraz czytelne dla użytkowników oraz by możliwie jak najbardziej naśladowała ona mechanikę przeglądania obiektów w Google Maps. Przejrzystości dodała „płytka" nawigacja, która sprawia, że użytkownik ma bardzo szybki dostęp do wszystkich elementów aplikacji i nie czuje się zagubiony podczas przeglądania wydarzeń.

Dodatkowym założeniem było zaprojektowanie elementów technicznych dla aplikacji. Wymagało to przemyślenia sposobu przechowywania danych w bazie danych oraz mechanizmu wymiany informacji. Dynamika aplikacji sprawiła, że należało pamiętać o usuwaniu zbędnych treści. W przypadku usunięcia konkretnego wydarzenia, w bazie danych nie powinny zalegać niepotrzebne już tabele odnoszące się do postów w tym wydarzeniu oraz „polubienia" i lista użytkowników biorących w nim udział. Nie można było zapominać również o zdjęciach w Firebase Storage. Wskazana była jak najmniejsza i możliwie jak najbardziej efektywna wymiana informacji, a transfer danych użytkowników z założenia nie powinien być zużywany na informacje, które nie będą wyświetlone w aplikacji. W związku tym, by uniknąć problemu, istniała konieczność kierowania przemyślanych zapytań do bazy danych.

Realizowany projekt był dużym wyzwaniem dla autora, wiele rozwiązań musiało zostać zaimplementowanych samodzielnie, włącznie z szatą graficzną.

Aplikacja ma uniwersalny charakter, została zaprojektowana w taki sposób, aby niewielkim nakładem pracy z powodzeniem mogła wspierać także inne wydarzenia z wielu dziedzin naszego życia. Otwartą opcją jest dalsza rozbudowa aplikacji i dodanie jej kolejnych funkcjonalności i mechanik.

Stworzenie projektu wymagało rozszerzenia wiedzy, poznania i zastosowania w praktyce szeregu technologii, wcześniej znanych tylko w teoretycznym zarysie. Dostarczyło ciekawego doświadczenia w zakresie tworzenia aplikacji mobilnych oraz ich testowania i komunikacji. Bezcenne było zrozumienie jak złożonym procesem jest tworzenie oprogramowania, bez którego coraz większy odsetek populacji nie wyobraża sobie codzienności.

Spis rysunków

Spis tabel

Bibliografia

[1] *https://www.bbc.com/news/technology-51358192 (04.02.2020)*

[2] *https://www.facebook.com/search/top?q=wydarzenia (04.01.2022)*

[3] *https://miastodzieci.pl (04.01.2022)*

[4] *https://czasdzieci.pl/lodz/ (04.01.2022)*

[5] *http://www.imprezowoplenerowo.pl (04.01.2022)*

[6] *Flutter – tutorialspoint, simply easy learning (e-book), 2019, Tutorials Point (I) Pvt. Ltd.- https://www.tutorialspoint.com/flutter/flutter_tutorial.pdf*

[7] *https://pl.wikipedia.org/wiki/Flutter (04.09.2021)*

[8] *https://pl.wikipedia.org/wiki/Od%C5%9Bmiecanie_pami%C4%99ci (03.10.2020)*

[9] *https://en.wikipedia.org/wiki/Dart_(programming_language) (28.12.2021)*

[10] Zammetti F., 2019, Practical Flutter: Improve your Mobile Development with Google's Latest Open-Source SDK. Wyd. Apress

[11] *https://pub.dev (20.12.2021)*

[12] *https://www.freecodecamp.org/news/why-mobile-apps-makers-are-in-love-with-flutter/ (22.04.2020)*

[13] *https://gs.statcounter.com/os-market-share/mobile/worldwide (20.12.2021)*

[14] *https://pl.wikipedia.org/wiki/Android_(system_operacyjny) (23.01.2022)*

[15] Moroney L., 2017, *The Definitive Guide to Firebase: Build Android Apps on Google's Mobile Platform*. Wyd. Apress

[16] *https://en.wikipedia.org/wiki/Firebase (11.11.2021)*

[17] *https://www.c-sharpcorner.com/article/introduction-to-google-firebase/ (26.09.2019)*

[18] *https://pl.wikipedia.org/wiki/Mapy_Google (13.01.2022)*

[19] *https://pl.wikipedia.org/wiki/Emulator (17.04.2020)*

[20] *https://github.com (18.01.2022)*