Politechnika Łódzka
Instytut Informatyki

PRACA DYPLOMOWA INŻYNIERSKA

Aplikacja mobilna do zarządzania wydatkami grupowymi

Wydział Fizyki Technicznej, Informatyki i Matematyki Stosowanej
Promotor: dr. inż. Michał Karbowańczyk
Dyplomant: Bartosz Banachowski
Nr albumu: 203849
Kierunek: Informatyka
Specjalność: Inżynieria Oprogramowania i Analiza Danych

Łódź, 19.02.2020

Instytut Informatyki
90-924 Łódź, ul. Wólczańska 215, budynek B9
tel. 042 631 27 97, 042 632 97 57, fax 042 630 34 14 email: office@ics.p.lodz.pl

Spis treści

1. Wstęp

1.1 Wprowadzenie w tematykę

Dzielenie wydatków nie musi ograniczać się tylko do zapisywania dłużników w notatniku, elektronicznym notesie czy tworzeniu skomplikowanych arkuszy kalkulacyjnych. Tematykę problemu z podziałem rachunków możemy zaobserwować w różnych dziedzinach naszego życia. Począwszy od wydatków skupiających się wokół mieszkania ze współlokatorami, a skończywszy na rachunku za kolacje ze znajomymi. W każdych z tych sytuacji trudzimy się z pytaniem – jak szybko i sprawnie rozłożyć wydatek wśród naszej grupy.

1.2 Problematyka i zakres pracy

Ta oto praca porusza tematykę dzielenia grupowych wydatków wraz z systemem notyfikacji dowolnego użytkownika o jego zadłużeniu. Podział rachunków, może okazać się dużym wyzwaniem w momencie, w którym chcemy, aby każdy podzielony wydatek, był odpowiednio zapisany wraz z podziałem kwotowym, ilością dłużników czy też datą dokonania transakcji. Z pomocą mogą przyjść nam tradycyjne rozwiązania w postaci elektronicznych notatników czy arkuszy kalkulacyjnych, jednakże są one czasochłonne oraz mało efektywne. Na rynku aplikacji mobilnych istnieją częściowe rozwiązania, niestety nie pokrywają całościowo poruszonej problematyki. Aplikacje są w większości płatne, bez możliwości zaproszenia drugiego użytkownika do wspólnej grupy czy bez systemu do powiadamiania użytkowników o swoich zadłużeniach.

W skład pracy wchodzi przegląd istniejących na rynku rozwiązań odnoszących się do dzielenia wydatków grupowych wraz z analizą pod kątem funkcjonalności oraz zaprojektowanie i implementacja aplikacji mobilnej, która wspomaga proces dzielenia wydatków grupowych wraz z systemem notyfikacji dowolnego dłużnika.

1.3 Cele pracy

Główne cele pracy:

- Przegląd istniejących rozwiązań na rynku odnoszących się do dzielenia wydatkami grupowymi oraz analiza pod kątem ich funkcjonalności
- Zaprojektowanie a następnie stworzenie aplikacji mobilnej, która będzie miała możliwość rejestracji wydatków grupowych wraz z możliwością powiadamiania użytkowników o ich zadłużeniu
- Ocena przydatności stworzonej aplikacji w kontekście istniejących już rozwiązań

1.4 Opis układu pracy

W rozdziale pierwszym została opisana problematyka i zakres pracy wraz z głównymi celami. Rozdział drugi przedstawia przegląd istniejących rozwiązań dotyczących dzielenia wydatkami grupowymi. Projekt aplikacji mobilnej do dzielenia wydatków, w którym zawarte zostały wymagania funkcjonalne i niefunkcjonalne, architektura aplikacji, użyte technologie w projekcie oraz moduły do logowania, rejestracji, dodawania znajomych, rejestracji rachunków czy powiadamiania użytkownika o zadłużeniu został zaprezentowany w rozdziale trzecim. Rozdział czwarty zawiera szczegółowy opis implementacji aplikacji wraz z podziałem na warstwę modelu danych, warstwę logiki biznesowej oraz warstwę prezentacji. W rozdziale piątym znajduję się podsumowanie wraz z oceną przydatności aplikacji w ramach dostępnych rozwiązań. Na końcu jest zawarta bibliografia, spis rysunków i listingów.

2. Przegląd istniejący rozwiązań do dzielenia wydatków grupowych

2.1 Podstawowe definicje pojęć

Wydatek – zmniejszenie funduszu środków pieniężnych, który to fundusz jest sumą algebraiczną należności (bieżących), środków pieniężnych (odpowiednio zdefiniowanych) oraz zobowiązań (bieżących) [1].

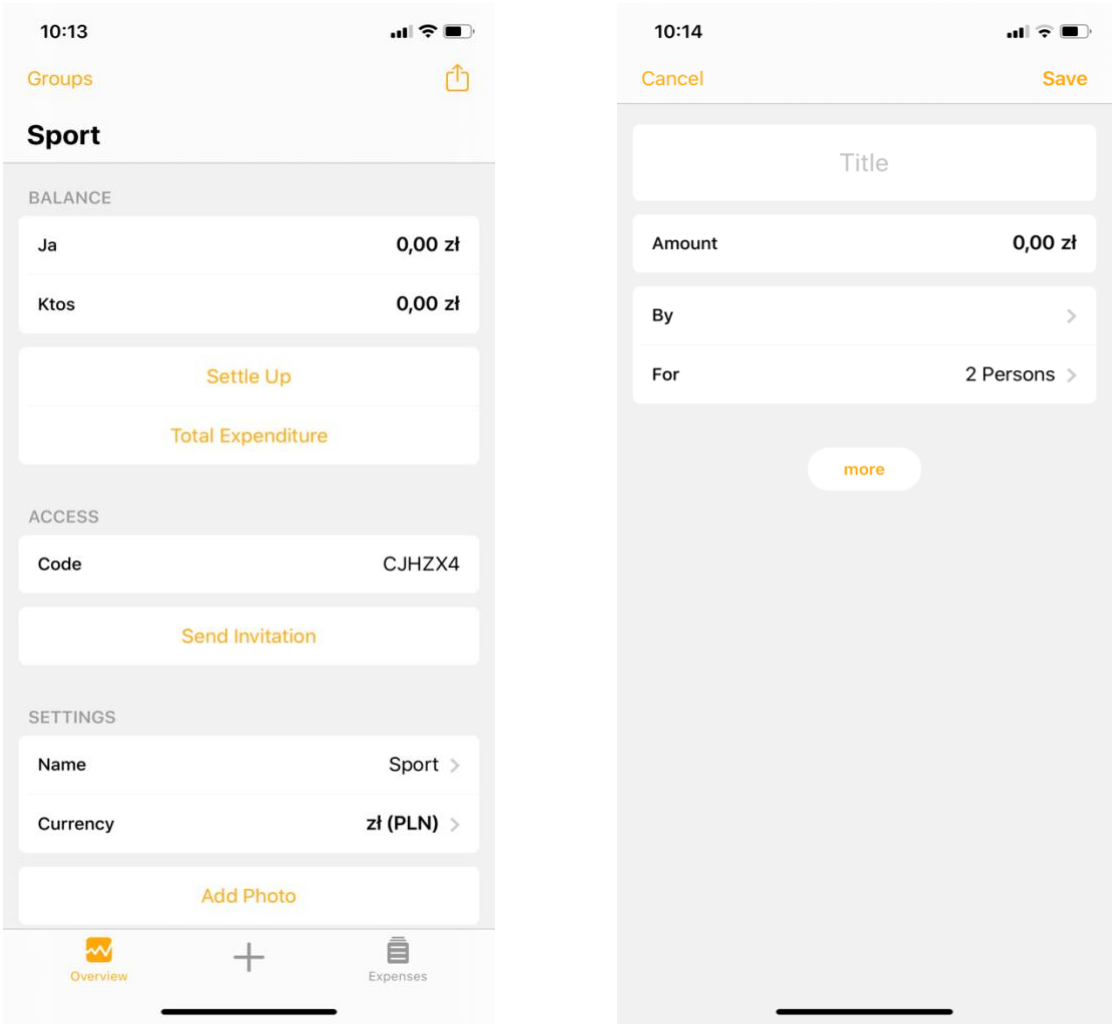
Użytkownik - osoba, grupa osób, instytucja lub organizacja, która ma prawo do użytkowania dobra (usługi, technologii, narzędzia) oraz do pobierania z niego pożytku. Użytkownik może korzystać z rzeczy w sposób podobny, lecz jednak nie identyczny jak czyni to jej właściciel [2].

2.2 Przedstawienie aplikacji mobilnych do dzielenia wydatków grupowych

2.2.1 Aplikacja „Splid”

Pierwszą aplikacją, poddaną analizie jest „Splid” (Rys. 2.1), dostępna w dwóch wersjach płatnej i bezładnej na platformy Android i iOS. W wersji bezpłatnej aplikacja pozwalana na stworzenie jednej grupy składającej się z dowolnej liczby osób. Użytkownik będąc w grupie może dodać swój wydatek, który jest podzielony na wybraną liczbę osób. Oprócz podstawowych informacji takich jak tytuł i rozmiar rachunku, możemy również dodać datę dokonania transakcji. Użytkownik dysponuje możliwością uregulowania swoich długów, a także dodania nowych członków do swojej grupy.

Do plusów aplikacji zdecydowanie można zaliczyć prostolinijny i przejrzysty wygląd aplikacji. Oprócz podstawowej funkcjonalności, aplikacja oferuje również możliwość zaproszenia dodatkowych osób, poprzez wysłanie kodu dostępu do grupy po przez wiadomość SMS oraz wiadomość e-mail. Ze względu na fakt, że aplikacja nie posiada możliwości stworzenia konta użytkownika, nie jest możliwe odtworzenie stanu stworzonych grup na innym urządzeniu bądź po odinstalowaniu i ponownym zainstalowaniu aplikacji co jest zdecydowanym minusem. W wersji płatnej mamy możliwość stworzenia więcej niż jedną grupę. Dostajemy również możliwość wyeksportowania zawartych informacji do formatu arkusza kalkulacyjnego oraz dzielenia wydatków według kategorii.



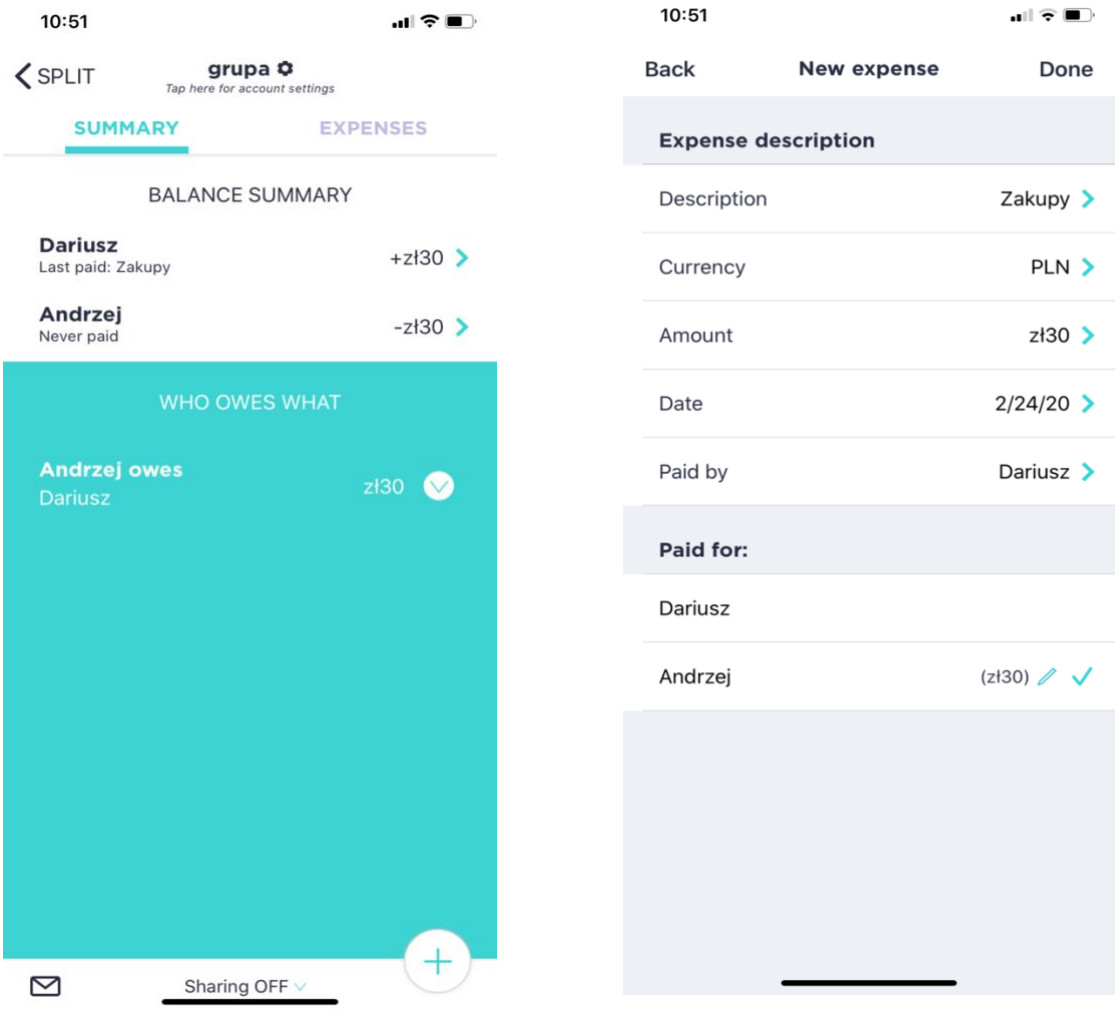
(a) Ekran grupy. *(b) Ekran dodawania rachunku.*

Rys. 2.1: Wygląd interfejsu użytkownika aplikacji „Splid”

2.2.2 Aplikacja „Split: group expenses manager”

Kolejną aplikacją służącą do zarządzania grupowymi wydatkami jest „Split: group expenses manager” (Rys. 2.2). Jest dostępna tylko na systemy iOS, zarówno w wersji bezpłatnej ze zmniejszoną ilością możliwości oraz w wersji płatnej, gdzie użytkownik ma sposobność wykupienia dodatkowych funkcjonalności. Aplikacja wspiera dodawanie rachunków w różnych walutach, a także wysyłanie podsumowań wydatków poprzez wiadomość e-mail.

W wersji płatnej aplikacji dostajemy dostęp do nielimitowanej ilości grup, synchronizacji wydatków grupowych pomiędzy członków grupy w czasie rzeczywistym oraz mamy możliwość podziału rachunku w sposób nieregularny.



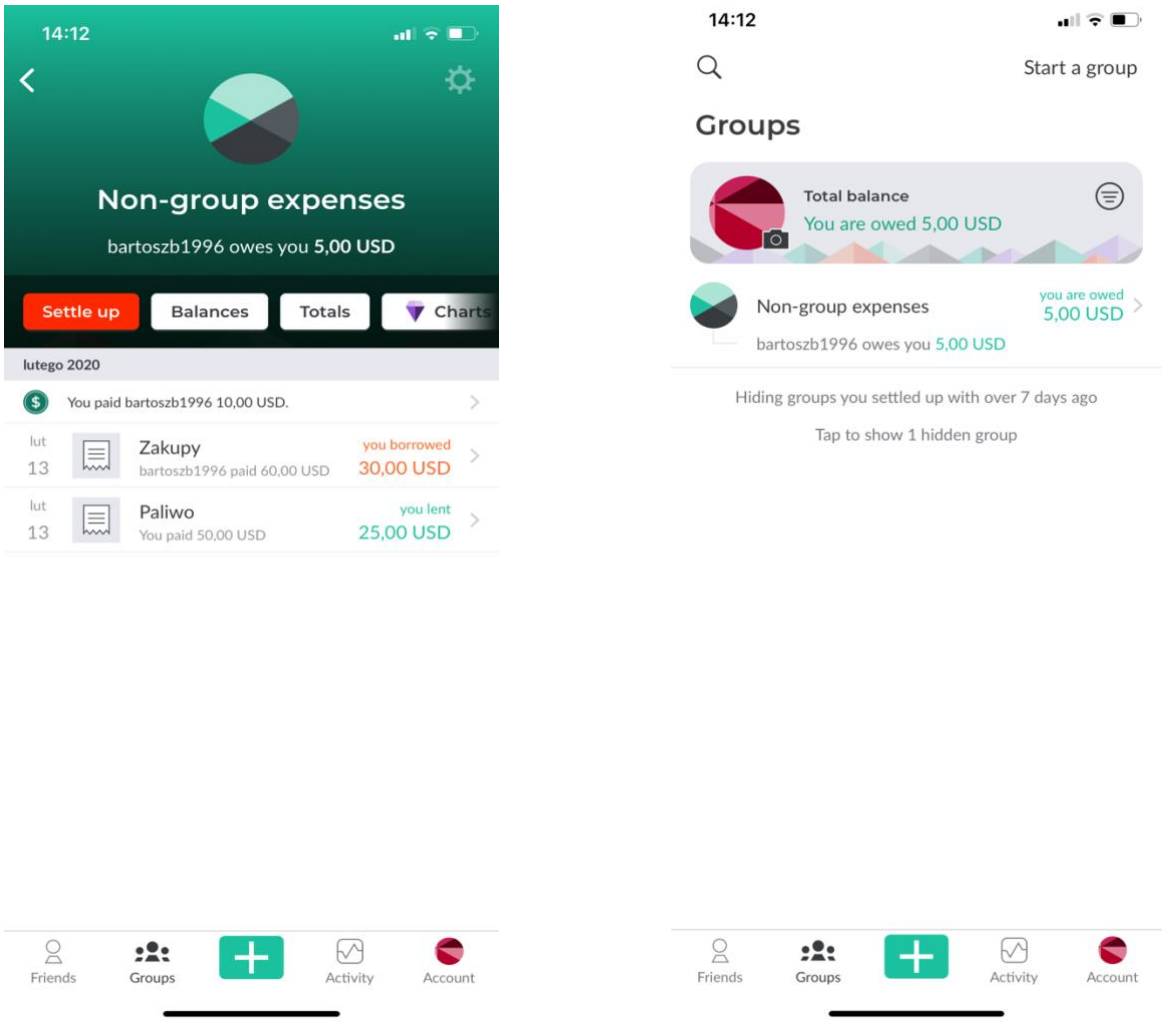
(a) Ekran grupy *(b) Ekran dodawania rachunku*

Rys. 2.2: Wygląd interfejsu użytkownika aplikacji „Split: group expenses manager”

2.2.3 Aplikacja „Splitwise”

„Splitwise” (Rys. 2.3) jest aplikacją dostępną poprzez systemy iOS i Android oraz przeglądarkę internetową. W wersji darmowej, podobnie do powyżej wymienionych aplikacji jest możliwość dzielenia wydatków ze znajomymi poprzez tworzenie grup i dodawanie rachunków. Aplikacja obsługuje eksportowanie podsumowań do arkusza kalkulacyjnego, kategoryzację wydatków, a także podział rachunku na równy, nieregularny lub procentowy. W wersji płatnej możemy przechowywać w wysokiej rozdzielczości zdjęcie rachunków, tworzyć kopie zapasowe naszych danych, które mogą być zapisane na dysku twardym, konwertować stworzone wcześniej wydatki do różnych walut czy przeszukiwać pełną historie wprowadzonych zmian w grupach.

Do plusów należy dodać możliwość stworzenia konta użytkownika i synchronizacji swoich danych na dowolnym urządzeniu posiadającym zainstalowaną aplikację. „Splitwise”, choć posiada szereg udogodnień, nie zawiera funkcjonalności do powiadamiania użytkownika o swoich długach co jest zdecydowanym minusem.



(a) Ekran grupy *(b) Ekran listy grup*

Rys. 2.3: Wygląd interfejsu użytkownika aplikacji „Splitwise”

3. Projekt aplikacji ExpensesDivider

3.1 Wymagania funkcjonalne

Wymagania funkcjonalne aplikacji, zostały w sposób szczegółowy opisane poniżej:

- **rejestracja użytkownika** – każdy niezalogowany użytkownik aplikacji ma możliwość stworzenia własnego konta użytkownika przy podaniu swoich danych osobowych tj. nazwa użytkownika, adres e-mail oraz hasło,
- **logowanie użytkownika** – każdy użytkownik, który posiada własne konto użytkownika ma możliwość zalogowania się do aplikacji przy pomocy adresu e-mail oraz hasła,
- **wyświetlanie listy znajomych** – użytkownik zalogowany ma możliwość wglądu do swojej listy znajomych,
- **dodawanie znajomego** – zalogowany użytkownik może dodać nowego znajomego, w zależności czy ów znajomy istnieje w bazie danych, zaproszenie będzie w stanie oczekującym bądź zaakceptowanym,
- **zaakceptowanie/odrzucenie zaproszenia od znajomego** – użytkownik zalogowany ma możliwość zaakceptowania bądź odrzucenia zaproszenia od znajomego
- **usunięcie znajomego** – zalogowany użytkownik ma możliwość usunięcia znajomego z listy,
- **wyświetlanie listy grup** – użytkownik zalogowany ma możliwość wglądu do wszystkich grup, które utworzył lub został zaproszony,
- **dodawanie grupy** – użytkownik zalogowany ma możliwość dodania nowej grupy, która będzie zawierać nazwę oraz listę użytkowników zaproszonych do grupy,
- **usuwanie grupy** – administrator grupy ma możliwość całkowitego usunięcia grupy,
- **edycja grupy** – administrator grupy ma możliwość edycji nazwy grupy, a także dodania bądź usunięcia poszczególnych użytkowników z grupy,
- **wyświetlanie listy rachunków –** użytkownik zalogowany ma możliwość wyświetlania listy rachunków dodanych przez siebie oraz każdego uczestnika grupy,
- **dodawanie nowego rachunku –** użytkownik zalogowany ma możliwość dodawania nowego rachunku, który zawiera nazwę i wartość rachunku, datę transakcji, użytkownika, który płacił oraz listę osób z którymi rachunek ma być podzielony,
- **usuwanie rachunku –** użytkownik zalogowany, który jest uczestnikiem grupy, ma możliwość usuwania rachunków z grupy,
- **uregulowanie zadłużenia –** użytkownik, który jest zadłużony ma możliwość uregulowania płatności, osobie, która zapłaciła za niego,

- **wyświetlenie bilansu –** użytkownik zalogowany, który należy do grupy ma wgląd w całościowy bilans pieniężny, uwzględniający listę wszystkich rachunków,
- **notyfikacja użytkownika o zadłużeniu –** uczestnik grupy może powiadomić drugiego uczestnika o jego zadłużeniu

3.2 Wymagania niefunkcjonalne

Lista wymagań niefunkcjonalnych aplikacji jest szczegółowo opisana poniżej:

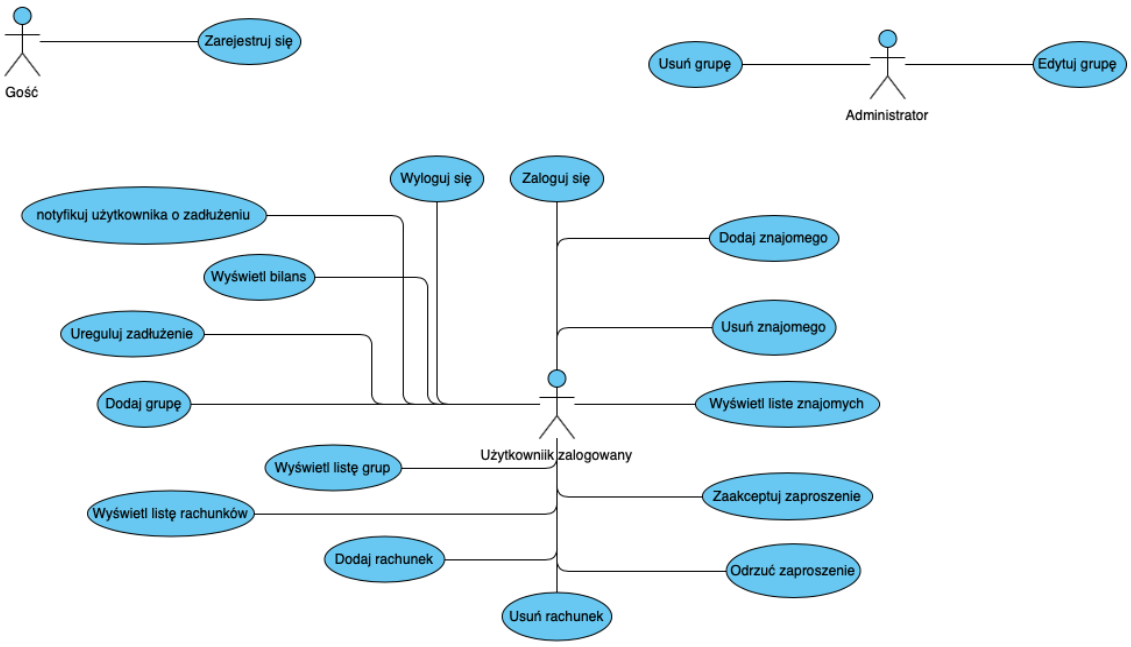
- **powiadamianie użytkownika o wystąpieniu błędu –** w momencie, gdy czynność wykonywana przez użytkownika ulegnie niepowodzeniu, na ekranie zostanie wyświetlona informacja o błędzie,
- **poziom skomplikowania hasła –** osoba próbująca stworzyć własne konto użytkownika musi posiadać hasło zawierające co najmniej 8 znaków w tym wielką i małą literę, znak specjalny oraz cyfrę,
- **ograniczony dostęp do aplikacji –** do aplikacji będą miały dostęp tylko osoby, które stworzą i następnie zalogują się do własnego konta użytkownika,
- **ograniczony dostęp do edycji grupy –** tylko użytkownik, który jest zalogowany oraz jest administratorem grupy może ją edytować oraz usuwać,
- **internacjonalizacja –** aplikacja obsługiwana w języku polskim i angielskim

3.3 Aktorzy oraz diagram przypadków użycia

Stworzona aplikacja zawiera trzy poziomy aktorów:

- gość – ma on możliwość stworzenia konta użytkownika, a następnie zalogowania do aplikacji
- użytkownik zalogowany – ma możliwość dodawania znajomych, akceptacji bądź odrzucenia zaproszenia, tworzenia nowych grup, dodawania i usuwania rachunków, a także uregulowania długów oraz notyfikacji pozostałych uczestników grupy o zadłużeniu
- administrator – posiada możliwości edycji i usuwania grupy

Podstawowym poziomem aktora, po zakończonej rejestracji jest użytkownik zalogowany. Użytkownik aplikacji może posiadać uprawnienia więcej niż jednego aktora. Aktorzy i ich uprawnieniami są pokazane na diagramie przypadków użycia (Rys. 3.1).



Rys. 3.1 Diagram przypadków użycia (opracowane przez autora)

3.4 Model warstwy danych

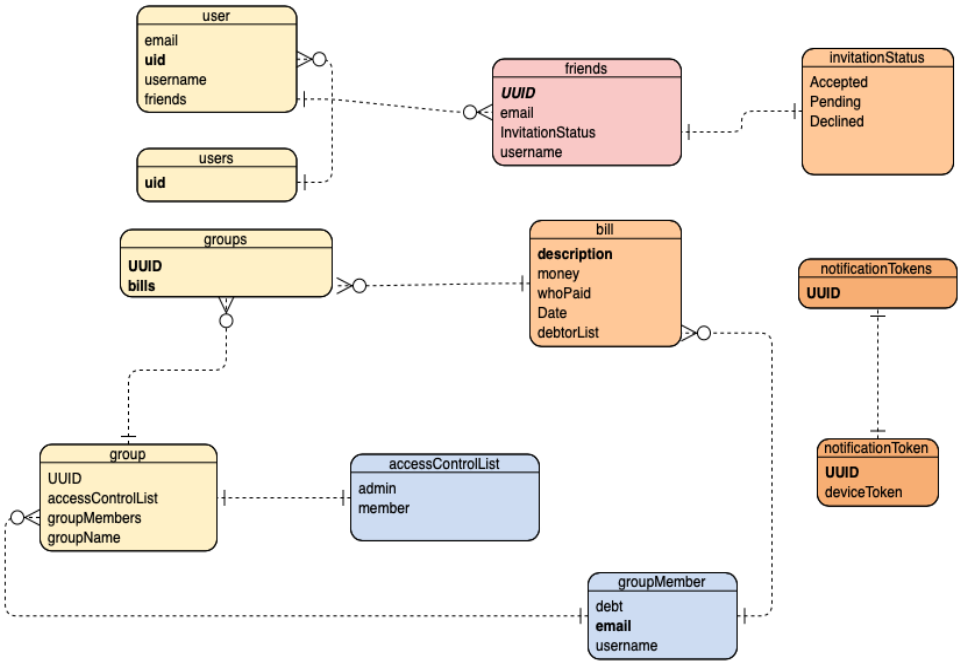
Model warstwy danych będzie składać się z:

- Kolekcji users
- Kolekcji groups
- Kolekcji notificationTokens

Kolekcja users zawiera listę wszystkich zarejestrowanych użytkowników. Poszczególny użytkownik posiada podstawowe informacji podane przy rejestracji tj. nazwa użytkownika oraz e-mail. W kolekcji zawarte są również dane znajomych dodanych przez użytkownika. Każda znajomy ma swój unikalny identyfikator wraz z nazwą użytkownika oraz statusem zaproszenia, który może być zaakceptowany, w trakcie bądź odrzucony.

Kolekcja groups zawiera listę wszystkich stworzonych grup. Poszczególna grupa zawiera swój unikalny identyfikator wraz z listą praw dostępu do grupy, nazwą oraz listą użytkowników dodanych do grupy. Lista kontrolna dostępu zawiera dwa poziomy – admin oraz member. Każda grupa posiada informacje o dodanych wydatkach, gdzie każdy rachunek dodany do aplikacji posiada swój opis, wielkość rachunku, kto zapłacił za rachunek, datę wykonanej transakcji oraz listę dłużników.

Kolekcja notificationTokens zawiera listę tokenów identyfikujących urządzenie, do którego ma zostać wysłane powiadomienie o oczekującym zadłużeniu.



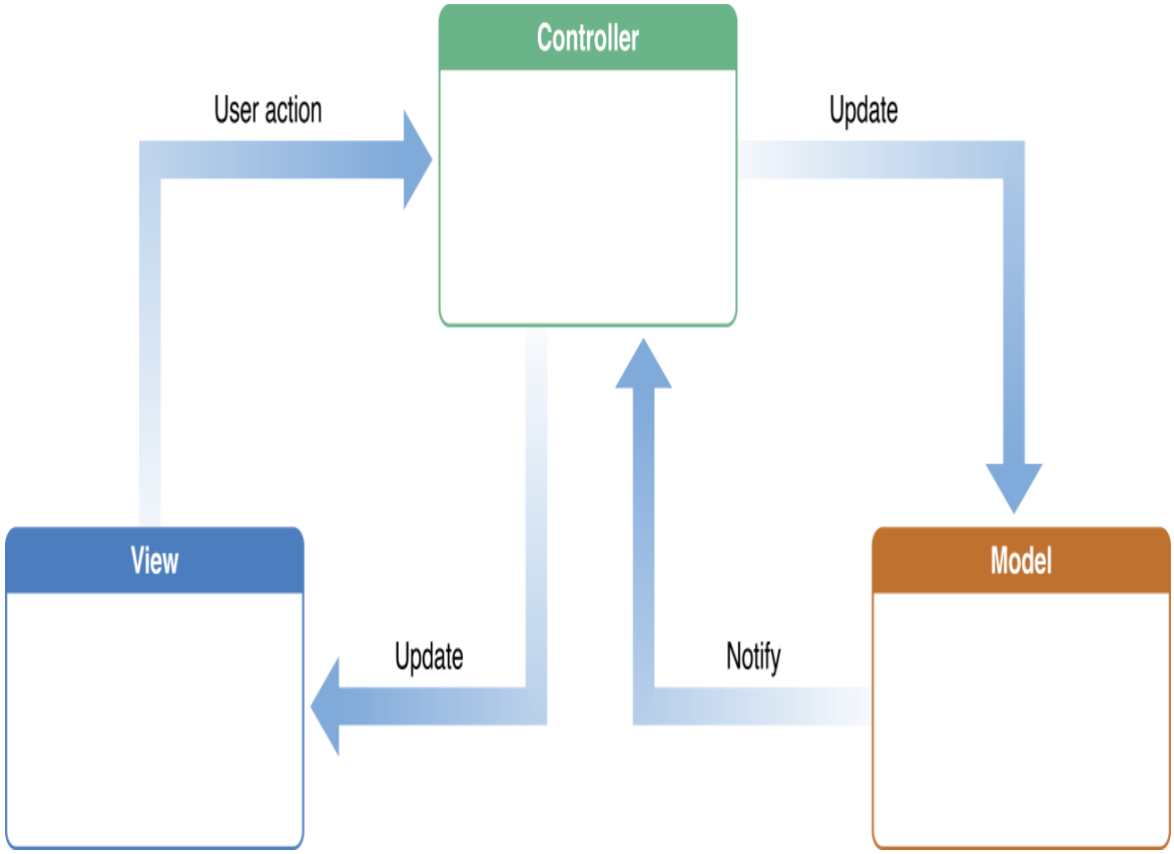
Rys. 3.2 Diagram bazy danych (opracowane przez autora)

3.5 Opis wykorzystanych technologii przy procesie tworzenia aplikacji

3.5.1 Wzorzec projektowy MVC

Model-View-Controller (MVC) jest podstawowym wzorcem projektowym, który służy do oddzielenia logiki biznesowej od logiki interfejsu użytkownika [3]. Wzorzec zawiera w sobie trzy warstwy [4]:

- warstwa danych służy do pobierania, wstawiania lub aktualizacji danych do bazy danych powiązanej z naszą aplikacją.
- warstwa interfejsu użytkownika służy do przygotowania interfejsu naszej aplikacji. Korzystając z interfejsu, użytkownik wchodzi w interakcje z naszą aplikacją.
- warstwa logiki biznesowej służy do odpowiadania na żądania użytkownika. Klasy warstwy logiki biznesowej wybierają odpowiedni widok tak aby wyświetlony ekran był zgodny z żądaniami użytkownika



Rys. 3.3 Wzorzec projektowy Model-View-Controller [5]

3.5.2 Język programowania Swift

Język programowania Swift został w całości stworzony przez firmę Apple Inc. Ze względu na wielką popularność do systemu iOS, jego poprzednik Objective-C stał się jednym z najczęściej wybieranych języków, jednakże ze względu na głęboko zakorzenioną składnie wywodzącą się z rodziny języków C, nie było możliwości modernizacji niektórych aspektów składni języka. Swift to całkowicie nowy język programowania, zaprojektowany specjalnie w celu przyspieszenia wytwarzania oprogramowania i zmniejszenia podatności na błędy programisty [6].

Przedstawienie podstawowej składni jest pokazane na listingu 3.1.

```
print("Hello, world!")
// Prints "Hello, world!"
```

Listing 3.1 Wyświetlanie podstawowego napisu „Hello, world!” [7].

3.5.3 Baza danych Firebase

Warstwa serwerowa aplikacji jest oparta o platformę dostępną w ramach usługi dostarczanej przez Google Firebase. Głównym zadaniem programisty jest zintegrowanie dostarczanych zasobów w ramach własnej aplikacji. Google Firebase posiada wszystkie niezbędne funkcjonalności potrzebne do zbudowania aplikacji biznesowej.

W ramach budowy aplikacji ExpensesDivider zostały wykorzystane niżej opisane usługi:

- **Cloud Firestore –** łatwo skalowalna baza danych do tworzenia aplikacji mobilnych, serwerowych lub internetowych. Baza danych utrzymuję synchronizację danych między aplikacjami klienckimi za pośrednictwem nasłuchiwania w czasie rzeczywistym oraz oferuje wsparcie offline dla urządzeń mobilnych [8]. Cloud Firestore jest nierelacyjną bazą danych NoSQL, która rozwiązuje problemy związane z zarządzaniem danymi. Baza danych jest zaprojektowana tak, aby wykorzystywać wiele serwerów jednocześnie. Dla bazy danych NoSQL istnieją cztery strategie zarządzania danymi: baza klucz-wartość, baza dokumentów, baza rodziny kolumn oraz bazy grafowe [9].

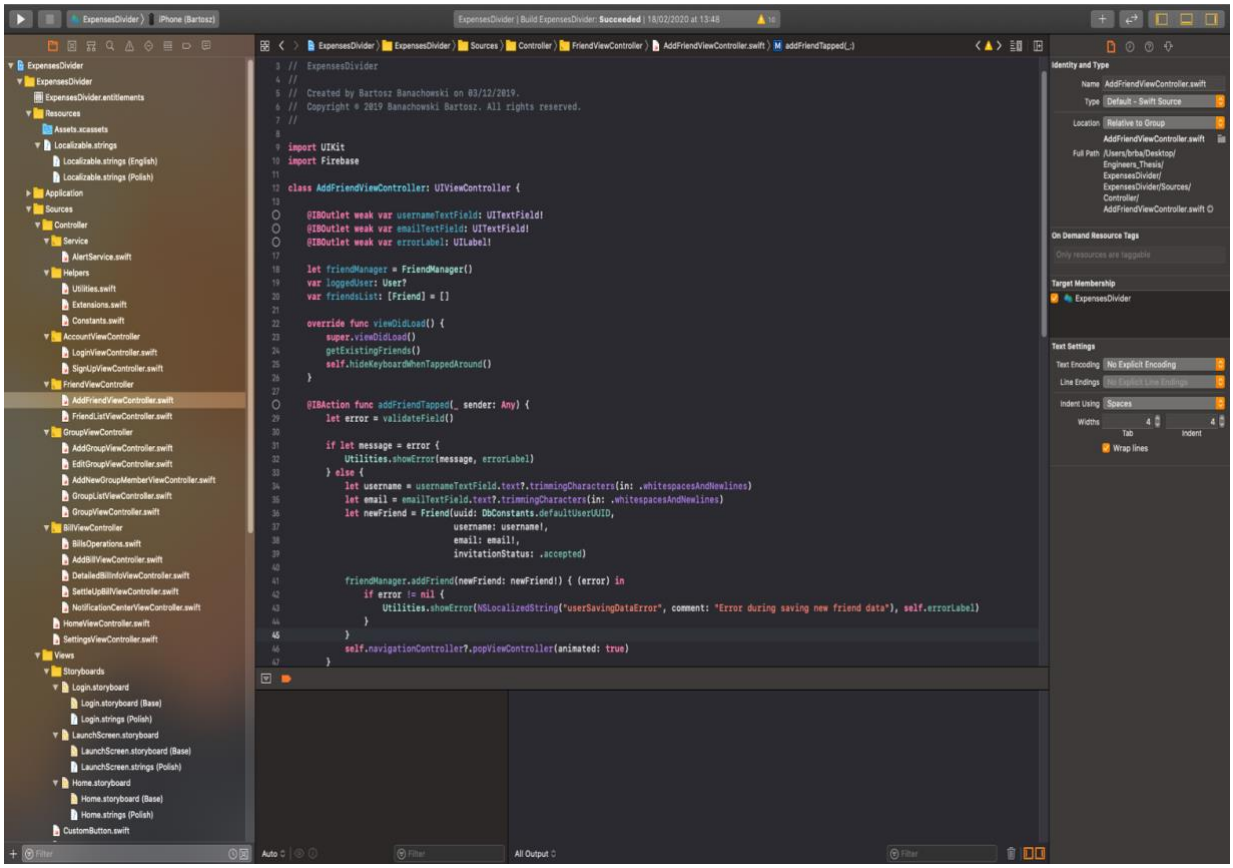
- **Firebase Authentication –** jest to usługa umożliwiająca w prosty sposób autentykację użytkowników aplikacji. Firebase authentication wspiera autentykację za pomocą numeru telefonu, adresu e-mail i hasła oraz jest możliwość przeprowadzenia procesu autentykacji za pomocą serwisów Google, Facebook. Twitter [10].

- **Cloud Messaging -** wieloplatformowa usługa zapewniająca możliwość notyfikacji aplikacji klienckiej o nowych danych dostępnych do synchronizacji. Z pomocą Cloud Messaging, użytkownik jest w stanie wysyłać określone powiadomienia do innych użytkowników aplikacji [11].

3.5.4 Wykorzystane narzędzia programistyczne

W ramach pracy nad aplikacją mobilną, zostały wykorzystane następujące narzędzia programistyczne:

Xcode IDE to bezpłatne, profesjonalne środowisko programistyczne stworzone przez firmę Apple Inc, które służy do tworzenia aplikacji mobilnych na platformę iOS oraz aplikacji desktopowych na platformę macOS. Środowisko zapewnia również możliwość uruchamiania aplikacji w symulatorze oraz narzędzia do mierzenia wydajności [12].



Rys. 3.4 Środowisko programistyczne Xcode

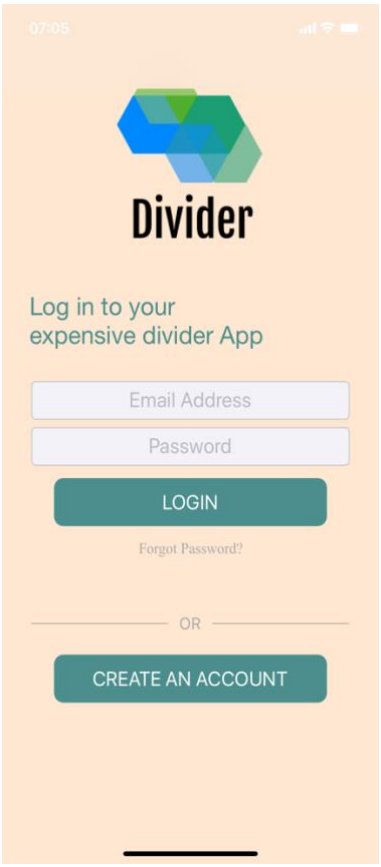
CocoaPods to narzędzie napisane w języku Ruby, służące do zarządzania zależnościami poziomu aplikacji. Wykorzystywany dla języków Swift oraz Objective-C, posiada dostępną publicznie listę bibliotek. Podczas dodawania nowych bibliotek do projektu, menedżer zależnościami może zmodyfikować stworzony wcześniej projekt, tworząc nowy plik, w którym podlinkowane są wszystkie wybrane biblioteki [13].

3.6 Segment logowania i rejestracji

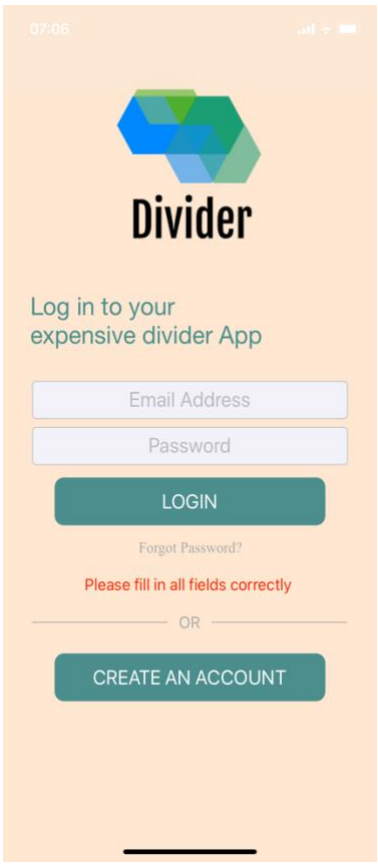
Moduł logowania i rejestracji odpowiada za następujące dwa przypadki użycia:

Rejestracja nowego użytkownika (Rys. 3.7) – po otworzeniu aplikacji ExpensesDivider, mamy możliwość stworzenia nowego konta użytkownika, klikając na przycisk „Create an account”. Następnie otwiera nam się formularz, w którym musimy wypełnić nazwę użytkownika, adres e-mail, oraz wpisać hasło dostępu wraz z jego potwierdzeniem. Hasło musi zawierać co najmniej osiem znaków włączając w to znak specjalny, cyfrę oraz wielką i małą literę. Po kliknięciu na przycisk „Sign up”, zostajemy automatycznie przeniesieni do ekranu głównego.

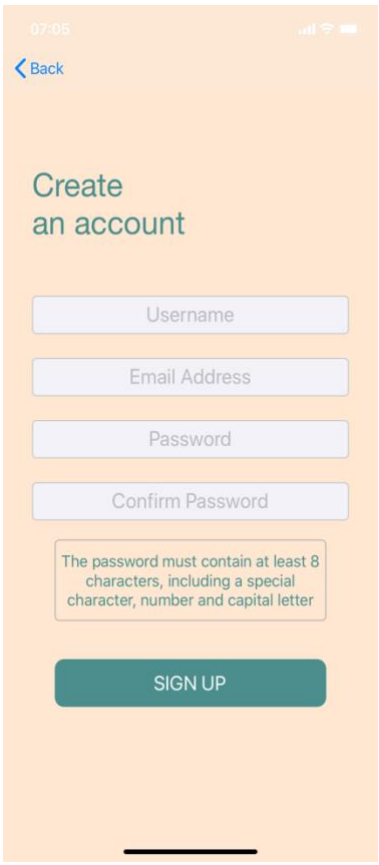
Logowania do aplikacji (Rys. 3.5) – to proces uwierzytelniania za pomocą podanego podczas rejestracji adresu e-mail oraz hasła. Przy braku wypełnieniu pól formularza bądź wprowadzeniu niewłaściwych danych logowania pojawia się komunikat (Rys. 3.6), w przeciwnym wypadku pojawia się ekran główny aplikacji.



Rys. 3.5 Widok logowania



Rys. 3.6 Informacja o błędzie



Rys. 3.7 Widok rejestracji

3.7 Segment zarządzania listą znajomych

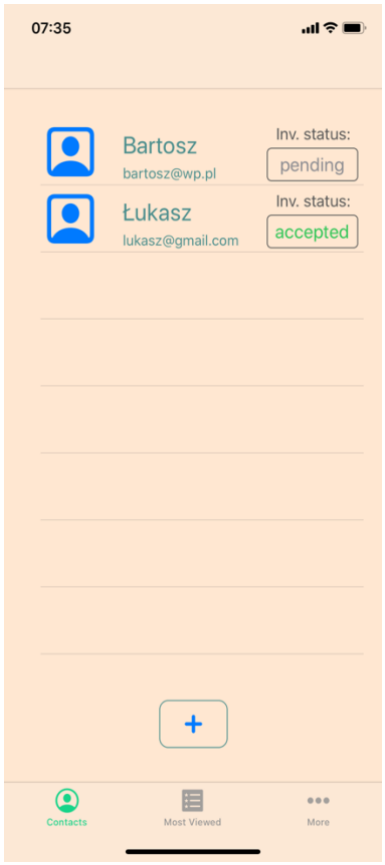
Moduł pokrywa następujące przypadki użycia:

Wyświetlanie listy znajomych (Rys. 3.8) **–** gdy użytkownik przejdzie pozytywnie proces logowania, zostanie pobrana lista jego znajomych i wyświetlona jako pierwszy ekran po zalogowaniu.

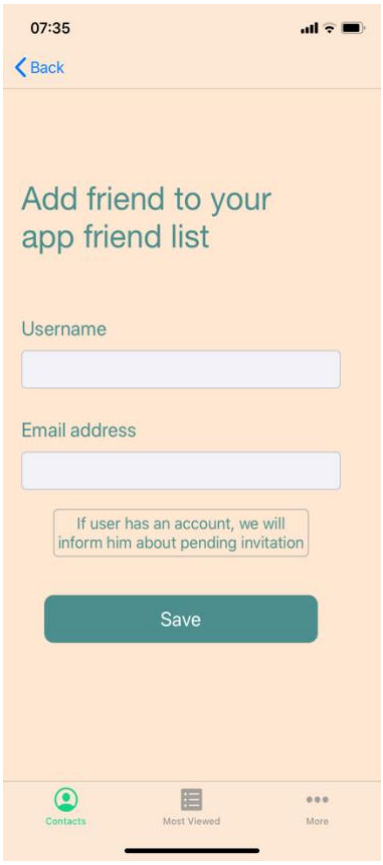
Dodawanie znajomego (Rys. 3.9) **–** po naciśnięciu na przycisk plusa na dole ekranu, aplikacja przeniesie nas do formularza. Obligatoryjnymi danymi są nazwa użytkownika oraz adres e-mail. Przy próbie ponownego wprowadzenia istniejącego użytkownika pojawi się komunikat informujący użytkownika o błędzie.

Usuwanie znajomego (Rys. 3.10) – użytkownik ma możliwość usunięcia znajomego z listy co skutkuje usunięciem znajomego z bazy danych. Aby poprawnie usunąć znajomego należy wybrać odpowiadającą komórkę tabeli i przesunąć palcem w lewo.

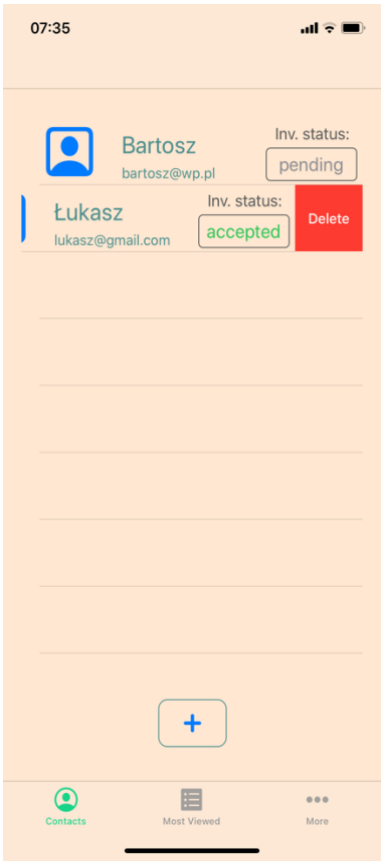
Zarządzanie zaproszeniem do listy znajomych (Rys. 3.11) – proces dodawania znajomego generuje zaproszenie, które jest jemu wysyłane. Po zalogowaniu znajomego na konto, wyświetla mu się komunikat, dzięki któremu może zaakceptować bądź odrzucić zaproszenie.



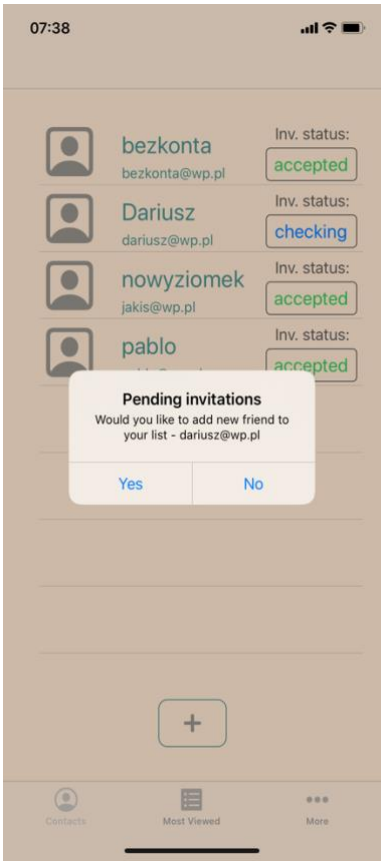
Rys. 3.8 Widok listy znajomych



Rys. 3.9 Widok dodawania znajomego



Rys. 3.10 Widok usuwania znajomego



Rys. 3.11 Menu zarządzania zaproszeniami

3.8 Segment zarządzania grupami oraz rejestracji wydatków

Wyświetlanie listy grup (Rys. 3.12) – użytkownik klikając na środkowy przycisk znajdujący się na dolnym pasku, przejdzie do sekcji wyświetlającej listę grup. Każda komórka w tabeli zawiera nazwę grupy oraz poziom dostępu jaki zalogowany użytkownik ma w danej grupie.

Dodawanie nowej grupy (Rys. 3.13) – po naciśnięciu na przycisk plusa użytkownik musi wypełnić pole odpowiadające nazwy użytkownika oraz dodać znajomych ze swojej listy. Tylko znajomi, którzy mają status „accepted" mogą być dodani do grupy

Edycja grupy – użytkownik o uprawnieniach administratora może edytować grupę. W celu edycji należy przesunąć odpowiednią komórkę tabeli w lewo, a następnie kliknąć przycisk „Edit"

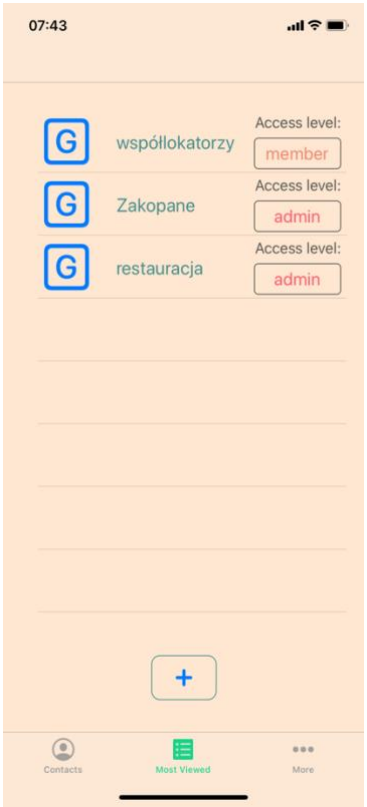
Usuwanie grupy (Rys. 3.14) – użytkownik, który jest administratorem grupy ma prawo do usuwania. Gdy użytkownik o uprawnieniach „member" będzie próbował usunąć grupę, pojawi się komunikat o niewystarczających uprawnieniach.

Wyświetlanie listy rachunków – po poprawnym zalogowaniu do aplikacji lista wszystkich rachunków jest pobierana z bazy danych. Aby sprawdzić listę rachunków dla danej grupy należy kliknąć w odpowiadającą komórkę tabeli.

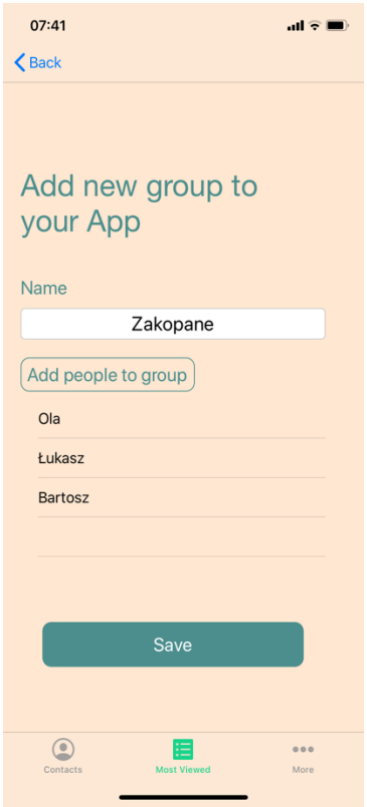
Dodawanie rachunku (Rys. 3.15) **–** przycisk plusa powoduję przejście do formularza, w którym trzeba wypełnić wszystkie pola. Pole „description" jest unikalne i nie może się powtarzać wśród listy rachunków. W razie niewypełnienia pól formularza bądź wypełnienia go w sposób nieprawidłowy, zostanie wyświetlony komunikat o błędzie.

Usuwanie rachunku – aby usunąć rachunek należy przesunąć w lewą stronę odpowiadającą komórkę tabeli a następnie kliknąć na przycisk „Delete". Rachunek jest usuwany dla wszystkich użytkowników należących do danej grupy.

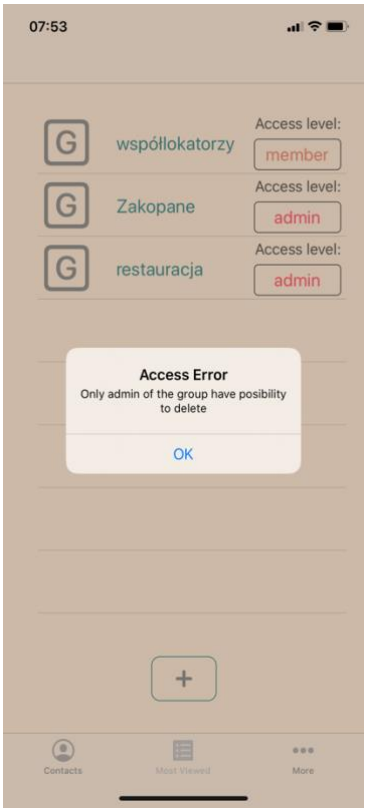
Rozliczanie rachunku (Rys. 3.16) **–** uregulowanie rachunku polega na naciśnięciu przycisku „Settle up", a następnie wpisaniu kwoty oraz wybraniu osoby, dla której chcemy uregulować dług.



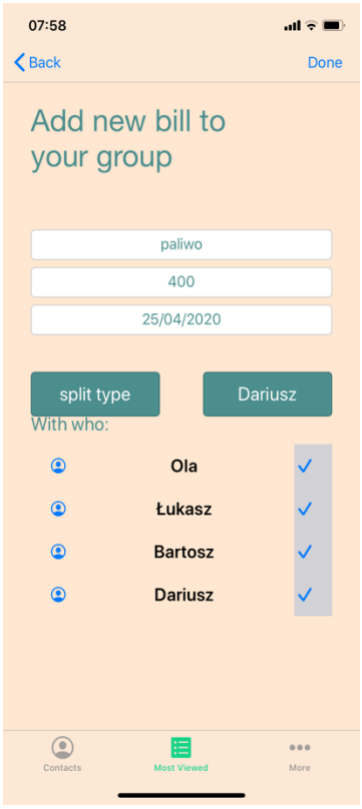
Rys. 3.12 Widok listy grup



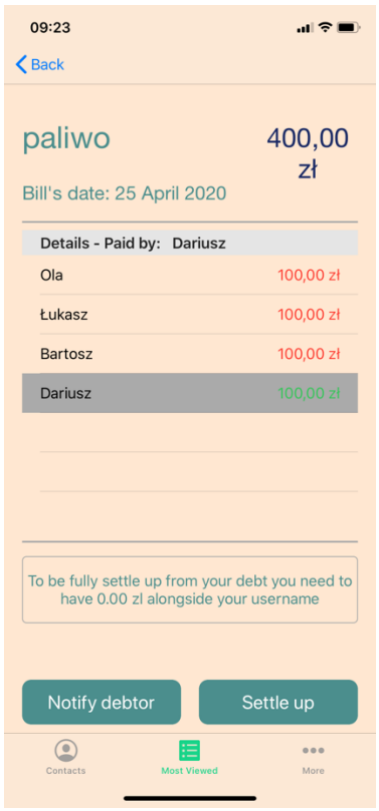
Rys. 3.13 Widok dodawania grupy



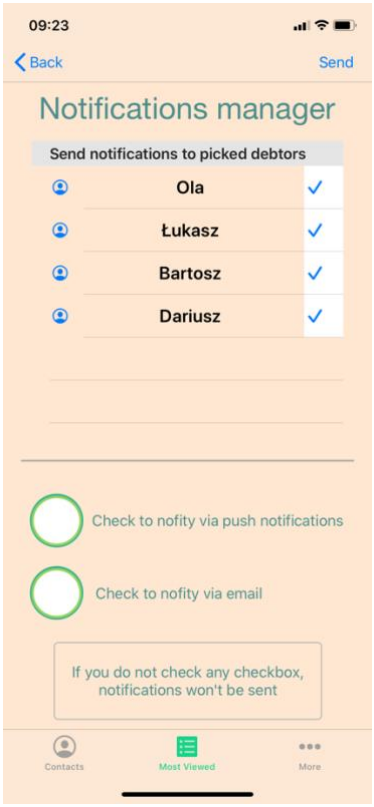
Rys. 3.14 Informacja o braku odpowiednich uprawnień



Rys. 3.15 Widok dodawania rachunku



Rys. 3.16 Widok szczegółowy rachunku



Rys. 3.17 Widok menadżera notyfikacji

3.9 Segment powiadamiania użytkownika o zadłużeniu

Notyfikacja użytkownika (Rys. 3.17) – daję możliwość powiadamiania wybranego użytkownika o jego zadłużeniu. Po naciśnięciu przycisku „Notify debtor”, aplikacja wyświetla widok menedżera notyfikacji. Z listy dłużników należy wybrać osoby, które zostaną powiadomione oraz sposób notyfikacji: za pomocą wiadomości e-mail lub powiadomienia typu Push.

4. Implementacja aplikacji ExpensesDivider

4.1 Warstwa prezentacji danych

Warstwa prezentacji jest podzielona między trzy pliki typu storyboard:

- Login.storyboard
- LaunchScreen.storyboard
- Home.storyboard (Rys. 4.1)

Rys. 4.1 Zawartość pliku Home.storyboard w programie Xcode

Plik Login.storyboard zawiera zaprojektowane widoki do logowania i rejestracji użytkownika. Launch.storyboard zawiera widok, który jest pokazywany wraz z uruchomieniem aplikacji. W ramach pliku Home.storyboard znalazły się zaprojektowane widoki do obsługi modułu znajomych, grup oraz rachunków, a także menadżera notyfikacji.

Tabele wyświetlające dane dotyczące znajomych, grup oraz rachunków zostały zmodyfikowane na potrzeby wyświetlania niestandardowych zestawów informacji.

```
func getColorForInvitationStatus(InvitationStatus) -> UIColor{
        switch status {
        case .accepted:
            return .systemGreen
        case .checking:
            return .systemBlue
        case .pending:
            return .systemGray
```

Listing 4.1 Funkcja zmieniająca kolor statusu zaproszenia

4.2 Warstwa logiki biznesowej

Warstwa logiki biznesowej jest zgrupowana w obrębie czterech modułów oraz zestawu klas wspomagających. Każdy z modułów zawiera kontrolery widoków odpowiadające bezpośrednio za przetwarzanie danych, które są wyświetlane na ekranie. Każdy kontroler, który dokonuje zmiany danych, ma dostęp do menadżerów, które wykonują działanie na bazie danych.

Moduł zarządzania kontem zawiera logowanie i rejestracje użytkownika. Przed wysłaniem danych użytkownika do bazy danych, aplikacja sprawdza czy wypełnione dane są poprawne. Podczas logowania jest tworzony token uwierzytelniający, który jednoznacznie identyfikuje użytkownika, ale również przechowuje podstawowe informacje o użytkowniku takie ja identyfikator czy adres e-mail. Na listingu 4.2 jest pokazana implementacja funkcji logującej użytkownika.

```
@IBAction func loginTapped(_ sender: UIButton) {
        let error = validateField()

        if let message = error {
            Utilities.showError(message, errorLabel)
        } else {
            let login =
loginTextField.text?.trimmingCharacters(in:
.whitespacesAndNewlines)
            let password =
passwordTextField.text?.trimmingCharacters(in:
.whitespacesAndNewlines)

            Auth.auth().signIn(withEmail: login!, password:
password!) { (_, error) in
                if let error = error {

Utilities.showError(NSLocalizedString("userLoginError",
comment: "Error during log in user") +
error.localizedDescription,
                                                self.errorLabel)
                    NSLog("User login error:
\(error.localizedDescription)")
                } else {
                    NSLog("Login succesful")
                    self.goToHome()
                }
            }
        }
}
```

Listing 4.2 Implementacja funkcji odpowiedzialnej za logowanie do aplikacji

Moduł zarządzania listą znajomych w momencie logowania do aplikacji zaczyna pobierać dostępną listę znajomych oraz zaczyna nasłuchiwać w czasie rzeczywistym na ewentualnie zmiany, przykładowo zmiana statusu zaproszenia po zaakceptowaniu przez znajomego. Na listingu 4.3 przedstawiona jest funkcja, która aktualizuje listę znajomych. Podczas wyświetlania listy znajomych, sprawdzana jest także, lista ewentualnych zaproszeń.

W momencie dodawania nowego znajomego, sprawdzany jest adres e-mail, który nie może istnieć w obecnej liście znajomych oraz czy wszystkie pola w formularzu zostały wypełnione. Gdy użytkownik postanowi usunąć znajomego, wpis z listy jest usuwany zarówno u użytkownika jak i znajomego.

```
func startListeningForFriends() {
       friendManager.getFriendsListener { friendList, error in
            if error != nil {
                let errorAlert =
AlertService.getErrorPopup(title:
NSLocalizedString("ErrorTitle", comment: "error"),
body: NSLocalizedString("ErrorBody", comment: "Error"))
    self.present(errorAlert, animated: true, completion: nil)
            } else {
             self.checkInvitations(list: friendList) { friends in
                       self.friendList = friends
                       self.myFriendListTableView.reloadData()
                   }
            }
     }
}
```

Listing 4.3 Implementacja funkcji nasłuchującej ewentualne zmiany w liście znajomych

Moduł zarządzania grupami pobiera listę wszystkich grup, do których należy użytkownik oraz zaczyna nasłuchiwać na ewentualne zmiany. Podczas operacji tworzenia, aplikacja sprawdza, czy istnieję już grupa z taką samą nazwą oraz czy dodany jest co najmniej jeden użytkownik. Lista potencjalnych osób do dodania jest wypełniana tylko o użytkowników, którzy wcześniej zaakceptowali zaproszenie. Podczas usuwania bądź edycji grupy, aplikacja sprawdza najpierw stopień dostępu danego użytkownika. Tylko uczestnicy z poziomem „admin” mają prawo dokonać tych dwóch operacji.

W momencie ładowania widoku szczegółowego danej grupy, aplikacja wylicza bilans użytkownika biorąc pod uwagę zakres wszystkich rachunków, do których należy. W zależności czy użytkownik jest na bilansie ujemny czy dodatnim, będzie następować zmiana koloru bilansu z czerwonego na zielony. Na listingu 4.4 mamy przedstawioną funkcję do obliczania bilansu użytkownika oraz w zależności od wyliczeń, ustawienie koloru etykiety.

```
static func getMyBalance(bills:[Bill], user: String)-> UILabel
{
        let textLabel = UILabel()
        var balance: Decimal = 0
        var moneyBorrowed: Decimal = 0
        var moneyOwed: Decimal = 0
        for bill in bills {
            if bill.whoPaid == user {
                for debtor in bill.debtorList where
debtor.email != user {
                    moneyBorrowed += debtor.debt
                }
            } else {
                for debtor in bill.debtorList where
debtor.email == user {
                    moneyOwed += debtor.debt
                }
            }
        }

        balance = moneyBorrowed - moneyOwed
        if balance < 0 {
            textLabel.textColor = .red
        } else if balance > 0 {
            textLabel.textColor = .systemGreen
        }
        textLabel.text = Utilities.currencyFormatter(currency:
balance)
        return textLabel
         }
```

Listing 4.4 Implementacja funkcji wyliczającej bilans pieniężny użytkownika

Moduł zarządzania rachunkami pozwala na wyświetlanie listy rachunków oraz nasłuchiwanie w czasie rzeczywistym na dokonane zmiany. Podczas dodawania nowego rachunku, aplikacja sprawdza czy każde pole formularza jest wypełnione poprawnymi danymi. Po dodaniu bądź uregulowaniu rachunku, moduł uaktualnia bilans pieniężny użytkownika.

Menadżer notyfikacji pozwala na powiadomienie dowolnego użytkownika z danej grupy o niezapłaconym długu. Wysyłanie powiadomienia za pomocą wiadomości e-mail jest możliwe dzięki klasie `MFMailComposeViewController`, która umożliwia otworzenie standardowego klienta do wysyłania wiadomości e-mail. Wysyłanie notyfikacji typu push jest dostępne dzięki usłudze „Cloud Messaging”. Podczas logowania do aplikacji, użytkownik generuję swój token do powiadomień typu push, a następnie zapisuje w bazie danych wraz z unikalnym identyfikatorem użytkownika. W momencie wysyłania powiadomienia, aplikacja szuka identyfikatora użytkownika, którego zaznaczyliśmy w formularzu i pobiera token urządzenia, na które ma wysłać powiadomienie.

4.3 Warstwa modelu danych

W ramach warstwy modelu danych jest zawarte:

- Baza danych Cloud Firestore
- Struktury mapowania obiektów
- Klasy menadżerów

Zapytanie kierowane do bazy danych są obsługiwane przez klasy menadżerów, dzięki którym możemy wykonywać operacje odczytu, zapisu, edycji oraz usunięcia. Dane otrzymane z bazy danych, które są w postaci struktury klucz/wartość są mapowane na obiekty. Model obiektu to struktura, która zawiera typy proste oraz konstruktor, który zainicjalizuje wartości zmiennych. Na listingu 4.5 jest przedstawiona struktura mapowania obiektu „NotificationToken", która zawiera dwa pola typu String.

```
struct NotificationToken: Codable {
    var UUID: String
    var deviceToken: String

    init?(data: [String: Any]) {
       guard let UUID = data["UUID"] as? String,
       let deviceToken = data["deviceToken"] as? String else{
            return nil
        }

        self.UUID = UUID
        self.deviceToken = deviceToken
    }

    init?(UUID: String, deviceToken: String) {
        self.UUID = UUID
        self.deviceToken = deviceToken
    }
}
```

Listing 4.5 Implementacja struktury do mapowania danych

5. Podsumowanie

Podział wydatków grupowych staję się problemem, o którym wiele osób chciałoby jak najszybciej się uporać. Do tej pory, każdy kto chciał się pokonać przeszkodę, musiał zapisywać wydatki na kartce bądź w elektronicznym notesie. Takie rozwiązania pochłaniają dużą ilość czasu i energii. Na rynku istnieją aplikację pozwalające na podział wydatków, jednakże w wersji podstawowej oferują niewielki wachlarz funkcjonalności, przykładowo ograniczając możliwość liczby stworzonych grup do jednej.

Stworzona aplikacja ExpensesDivider jest systemem dostarczającym usługę podziału wydatków w grupie znajomych. Każdy użytkownik może stworzyć własne konto, pod którym będą zapisywane wszystkie informacje odnoszące się do listy znajomych, grup czy rachunków. Uruchomiona aplikacja oferuję unikatową funkcjonalność w postaci wysyłania notyfikacji o jego zadłużeniu do dowolnie wybranego użytkownika za pomocą wiadomości e-mail lub powiadomienia typu push.

Przeprowadzona analiza wśród wymienionych aplikacji pokazała brak pełnego rozwiązania postawionego problemu. System ExpensesDivider dzięki oferowaniu kompletnej usługi dzielenia wydatków oraz dodatkowo wyróżniając się unikalną funkcjonalnością jest korzystnym rozwiązaniem wśród oferowanych aplikacji.

Bibliografia

[1] Anna M. Kostur: TERMINOLOGIA RACHUNKOWOŚCI - PROBLEMY JEDNOZNACZNOŚCI W OPISIE STRUMIENI WYNIKU FINANSOWEGO ORAZ PŁYNNOŚCI BIEŻĄCEJ. http://docplayer.pl/docview/55/37690512/#file=/storage/55/37690512/37690512.pdf, dostęp: 01.2020

[2] Arkadiusz Wiktor, Dominika Przebięda: Użytkownik. https://mfiles.pl/pl/index.php/Użytkownik, dostęp: 01.2020

[3] Gupta P, Govil MC: MVC Design Pattern for the multi framework distributed applications using XML, spring, and struts framework. http://www.enggjournals.com/ijcse/doc/IJCSE10-02-04-48.pdf, dostęp: 01.2020

[4] A Majeed: MVC Architecture: A Detailed Insight to the Modern Web Applications Development. https://crimsonpublishers.com/prsp/pdf/PRSP.000505.pdf , dostęp: 01.2020

[5] Pavel Gnatyuk: Modern MVC. https://miro.medium.com/max/1304/1*la8KCs0AKSzVGShoLQo2oQ.png, dostęp: 01.2020

[6] Neil Smyth: iOS 10 App Development Essentials: Learn to Develop iOS 10 Apps with Xcode 8 and Swift 3. CreateSpace, 2016, ISBN-10: 1539675033

[7] Apple Developers: The Swift Programming Language. https://docs.swift.org/swift-book/GuidedTour/GuidedTour.html, dostęp: 01.2020

[8] Google Developers: Cloud Firestore. https://firebase.google.com/docs/firestore, dostęp: 01.2020

[9] Dan Sullivan: NoSQL for Mere Mortals. Addison-Wesley Professional, 2015, ISBN-10: 0134023218

[10] Google Developers: Firebase Authentication. https://firebase.google.com/docs/auth, dostęp: 01:2020

[11] Google Developers: Firebase Cloud Messaging. https://firebase.google.com/docs/cloud-messaging, dostęp: 01.2020

[12] Brandon Alexander, J. Bradford Dillon, Kevin Y. Kim: Pro iOS 5 Tools: Xcode Instruments and Build Tools. Apress, 2012, ISBN-10: 1430236086

[13] Eric Downey: Practical Swift. Apress, 2016, ISBN: 978-1-4842-2280-5

Spis rysunków

Spis listingów