

Uniwersytet Mikołaja Kopernika  
Wydział Matematyki i Informatyki

Paweł Marcin Chojnacki  
nr albumu: 260082  
informatyka

Praca inżynierska

# **Kopra – aplikacja mobilna wspomagająca inwestowanie w serwisie Kokos**

Opiekun pracy dyplomowej  
dr Jerzy Białkowski

Toruń 2016



# Spis treści

<b>Słownik pojęć</b>	<b>5</b>
<b>Wstęp</b>	<b>7</b>
Pomysł napisania aplikacji o tematyce finansowej . . . . .	7
Podobne rozwiązania w świecie aplikacji mobilnych . . . . .	8
Rozważane formy wdrożenia projektu . . . . .	8
Jaki jest cel aplikacji? . . . . .	8
Założenia projektowe . . . . .	8
Korzystanie z serwisów pożyczkowych na smartfonach . . . . .	9
<b>1. O pożyczaniu</b>	<b>11</b>
1.1. Krótka historia pieniądza . . . . .	11
1.2. Idea pożyczek społecznościowych . . . . .	11
1.3. Ryzyka związane z pożyczkami społecznościowymi . . . . .	12
1.4. Jakie zyski można osiągnąć? . . . . .	12
1.5. Porównanie najpopularniejszych instrumentów finansowych . . . . .	13
1.6. Porównanie polskich serwisów pożyczkowych . . . . .	13
1.7. Wybór systemu do implementacji . . . . .	14
1.8. WebAPI . . . . .	15
<b>2. Wykorzystane technologie i narzędzia</b>	<b>17</b>
2.1. Microsoft Visual Studio Community 2015 . . . . .	17
2.2. Język C# 5.0 . . . . .	17
2.3. .NET Framework . . . . .	18
2.4. Windows Phone 8.1 SDK . . . . .	18
2.5. Newtonsoft.Json . . . . .	19
2.6. Git i GitHub . . . . .	19
2.7. LinqPad . . . . .	20
2.8. ILSpy . . . . .	20
<b>3. Budowa projektu</b>	<b>21</b>
3.1. Wzorzec MVVM . . . . .	21
3.1.1. Opis wzorca Model View ViewModel . . . . .	21
3.1.2. Porównanie MVVM i MVC . . . . .	22
3.1.3. Implementacja MVVM w aplikacji . . . . .	22
3.2. Kokos WebAPI . . . . .	24
3.2.1. Opis interfejsu webowego . . . . .	24
3.2.2. Implementacja API w aplikacji . . . . .	25
3.3. Notyfikacje przez BackgroundTask . . . . .	25

<b>4. Prezentacja interfejsu aplikacji</b>	<b>27</b>
4.1. Etymologia nazwy . . . . .	27
4.2. Korzyści względem wersji przeglądarkowej . . . . .	27
4.3. Koszt użytkowania aplikacji . . . . .	27
4.4. Bezpieczeństwo danych przechowywanych w aplikacji . . . . .	27
4.5. Jak wygląda Kopra? . . . . .	28
4.5.1. Logo . . . . .	28
4.5.2. Splash screen . . . . .	28
4.5.3. Strona logowania . . . . .	29
4.5.4. Strona główna . . . . .	29
4.5.5. Dodawanie filtru . . . . .	29
4.5.6. Wyszukiwanie aukcji . . . . .	29
4.5.7. Wyniki wyszukiwania . . . . .	30
4.5.8. Lista filtrów . . . . .	30
4.5.9. Ustawienia . . . . .	30
4.5.10. Raport inwestora . . . . .	30
4.5.11. Szczegóły aukcji . . . . .	30
4.6. Instrukcja obsługi do aplikacji . . . . .	31
<b>5. Podsumowanie</b>	<b>33</b>
<b>Spis rysunków</b>	<b>33</b>
<b>Bibliografia</b>	<b>35</b>

# Słownik pojęć

## **Aktywa finansowe**

Środki finansowe przechowywane przez podmiot gospodarczy (osobę lub firmę), mające na celu przyniesienie w przyszłości korzyści ekonomicznych. Są to przechowywane środki pieniężne, umowy kredytowe, akcje innych firm.

## **Aukcja pożyczkowa**

Oferta umowy pożyczkowej wystawiona przez pożyczkobiorcę, zweryfikowana na określonej platformie zgodnie z jej kryteriami. Oferta taka skierowana jest do wielu podmiotów określonych jako *pożyczkodawców*.

## **Automat inwestycyjny**

Oprogramowanie przeznaczone do symulowania działań pożyczkodawcy przez wpłacanie środków na aukcje spełniające określone kryteria.

## **FriendlyScore**

Przyznanie punktacji użytkownikowi w oparciu o analizę danych z kont w serwisach społecznościowych, np. Facebooka czy Twittera oraz dodatkowe weryfikacje, np. weryfikację pracodawcy.

## **Giełda Papierów Wartościowych**

Jest to instytucja publiczna zapewniająca możliwość obrotu giełdowymi papierami wartościowymi. Pozwala ona na kupno oraz sprzedaż papierów wartościowych takich jak akcje i obligacje. W mojej pracy ograniczę się do polskiego rynku kapitałowego, dlatego pojęcie „Giełda Papierów Wartościowych” (w skrócie „Giełda”) będzie w niej oznaczać instytucję „Giełda Papierów Wartościowych w Warszawie SA”. [4]

## **Instrument finansowy (ang. *financial instrument*)**

Jest to kontrakt zawarty między dwiema stronami regulujący zależność finansową między tymi stronami. W tym kontrakcie zawarte są dwie strony. Strona długa (ang. *long party*), inaczej – posiadacz długiej pozycji, oraz strona krótka (*short party*), inaczej – posiadacz krótkiej pozycji. Przykładem instrumentu finansowego są papiery wartościowe. [3]

## **Instytucja finansowa**

Podmiot gospodarczy, który zajmuje się obrotem środków finansowych powierzonych przez klientów (udzielanie kredytów, inwestycje, gromadzenie środków pieniężnych).

## **Oferta inwestycyjna**

Przekazanie wkładu pieniężnego na wybraną pożyczkę przez jednego z pożyczkodawców. Wiąże się to ze wstępną akceptacją oferty pożyczkowej na warunkach wcześniej określonych przez osobę wystawiającą aukcję pożyczkową (*pożyczkobiorcę*).

## **Pożyczka**

Przekazanie określonemu podmiotowi podmiotu pieniędzy na określony czas z zastrzeżonym terminem zwrotu.

**Pożyczka społecznościowa (ang. *social lending*)**

Pożyczka wykonywana bezpośrednio pomiędzy osobami fizycznymi poprzez serwisy internetowe, bez pośrednictwa instytucji finansowych.

**Pożyczkobiorca**

Osoba lub instytucja biorąca pożyczkę.[5]

**Pożyczkodawca**

Osoba lub instytucja udzielająca pożyczki.[5]

**RRSO (Rzeczywista roczna stopa oprocentowania)**

Wszystkie koszty realnie ponoszone przez pożyczkobiorcę. Kwotę wyraża się w wartości procentowej całkowitej kwoty kredytu w stosunku rocznym. (Przykładowo, Tomek, pożyczając 1000 zł na 2 lata musi oddać 1200 zł. Roczna opłata za kredyt wynosi  $200/2 = 100$  zł, co jest wartością 10% z całkowitej pożyczzonej kwoty. RRSO wynosi wtedy 10%.)

**Saldo**

Stan konta w wybranej walucie lub jednostce aktywów (ilość akcji, kruszcu), po uwzględnieniu wszystkich transakcji rozliczonych i nierozliczonych w danej chwili.

**Toksyczne aktywa**

Aktywa finansowe, których ryzyko inwestycyjne zostało pierwotnie istotnie niedoszacowane przez posiadacza. Przeważnie są to papiery wartościowe oraz pożyczki obarczone wysokim ryzykiem kredytowym lub płynności, którego realizacja może skutkować poważnym pogorszeniem sytuacji finansowej posiadacza i w efekcie jego upadłością.[6]

**Windykacja**

Dochodzenie własności za pomocą dostępnych środków prawnych. Dążenie do odzyskania pieniędzy przekazanych podmiotowi przez osobę wszczynającą proces odzyskiwania należności.

**Zobowiązania**

Zmuszenie prawem do świadczenia pieniężnego na rzecz wierzyciela.

# Wstęp

## Pomysł napisania aplikacji o tematyce finansowej

Od 2010 roku interesuję się tematyką oszczędzania pieniędzy i pomnażania oszczędności. Początkowo starałem się szukać bezpiecznego miejsca dla oszczędności w lokatach, które do 2012 roku jeszcze przebiły inflację. Lokaty jednodniowe, potocznie znane jako „antybelki”, były popularnym sposobem na ominięcie podatku od dochodów kapitałowych. Po kilkukrotnej inwestycji w lokaty, zauważyłem, że większy zysk można osiągnąć odkładając kapitał na Giełdzie Papierów Wartościowych. Szybka lektura na temat podstaw inwestowania w papiery wartościowe okazała się być dla mnie lekcją pokory. Jeśli zacznę inwestować bez poświęcenia wielu godzin na naukę funkcjonowania giełdy, stracę wszystkie oszczędności. Inwestowanie w akcje jest obciążone wysokim ryzykiem. Wymaga praktyki, wiedzy i ciągłego obserwacji sytuacji na rynku. Nie byłem skory do zmiany planów życia, by wyciągnąć trochę więcej z moich oszczędności.

Szukając w internecie innych możliwości inwestycji natknąłem się na względnie nowy produkt na polskim rynku. Są nim pożyczki społecznościowe – prosta idea wykorzystująca zestaw technologii znany jako Web 2.0. Zawartość serwisu tworzą użytkownicy. Pożyczając sobie wzajemnie pieniądze, omijają banki i zarabiają pieniądze na klasycznych umowach pożyczkowych. Zalety tego rozwiązania, takie jak zysk z inwestycji wyższy niż na lokatach bankowych oraz bardzo niska krzywa nauki, zachęciły mnie do wkroczenia w świat kredytów. Już ponad 6 lat odkładam nadmiar gotówki z przyzwoitym wynikiem, który po uśrednieniu przekracza maksymalny możliwy zysk z lokaty bankowej.

Gdybym posiadał ciągły dostęp do serwisu, osiągnąłbym dużo wyższy zwrot z inwestycji, ponieważ wiedziałbym, kiedy pojawia się tzw. gorąca oferta (pożyczka na bardzo dobrych warunkach, zazwyczaj jest zamykana w ciągu godziny od założenia aukcji). Jednak nie byłbym w stanie poświęcić całego dnia na przeglądanie wszystkich ofert. Mogę natomiast napisać aplikację mobilną, która poinformuje mnie, gdy nadarzy się okazja godna odciążenia od innych obowiązków.

Zdaję sobie sprawę, że również inni inwestorzy często wspominają, jak sprzed nosa uciekały im prawdziwe żyły złota – tylko dlatego że nie mieli ciągłego dostępu do strumienia informacji. Biorąc pod uwagę zainteresowanie wokół problematyki, uważam, że temat mojej pracy inżynierskiej znajduje praktyczne zastosowanie i nie jest tylko sztuką dla sztuki. Uważam go za projekt z potencjałem do dalszego rozwoju. Wybrałem temat, który mnie interesuje, czyli instrumenty finansowe. Są one nie tylko bardzo ciekawym lecz także istotnym zagadnieniem. Jest to szczególnie widoczne, gdy spojrzysz na nie z szerszej perspektywy – np. przez pryzmat skutków społeczno-ekonomicznych, jakie można spowodować przez manipulacje rynkami finansowymi. Jedna osoba, operując sporymi ilościami aktywów, jest w stanie znacząco zmienić gospodarkę całego kraju. Przykładem na to są działania George’a Sorosa, który na spekulacjach walutowych wyprowadził ponad 1 miliard funtów brytyjskich.[7]

Niniejsza praca skupia się wokół rynku pożyczek finansowych, gdzie występują niewielkie obroty gotówkowe w przeliczeniu na osobę. Ze względu na swój mikro charakter pożyczki społecznościowe cieszą się moim szczególnym uznaniem. Uważam, że pomagają jednostkom wyjść z problemów ekonomicznych, a w wielu przypadkach urzeczywistniają wizję samodzielnego podnoszenia stopy życia. Zalet pożyczek opartych na społecznościach jest wiele. Przedstawiam je w dalszych rozdziałach, zestawiając razem z wadami i ryzykiem inwestycyjnym.

## Podobne rozwiązania w świecie aplikacji mobilnych

Przeanalizowałem rynek aplikacji mobilnych w sklepach: Google Play, App Store, Windows Store, Amazon AppStore oraz F-Droid. W sklepie Google Play znalazłem jedną aktywnie rozwijaną aplikację wspomagającą pożyczki społecznościowe – Kokoid. Jest to nowa aplikacja (z początku 2016 roku), przeznaczona na system Android. Swoją model biznesowy opiera na wyświetlaniu reklam w trakcie działania.

Oprogramowanie wspomagające inwestycje istnieje głównie w formie aplikacji internetowych. Popularnym narzędziem wspomagającym pożyczkodawców jest [www.zndpd.pl](http://www.zndpd.pl) czyli „Zestaw Narzędzi Pożyczkodawców”. Stanowi ono rozbudowane narzędzie umożliwiające przeglądanie informacji o aukcjach, pożyczkobiorcach i statystykach spłacalności z różnych serwisów. Wszystkie dane zebrane są jednym miejscem w formie strony z raportem. Funkcjonalność portalu zorientowana jest na wyświetlaniu statystyk i generowaniu raportów dotyczących własnych inwestycji.

Nie dostrzegłem żadnych aplikacji o podobnej tematyce (przynajmniej na polskim rynku), na których mógłbym się wzorować. Z tego powodu większość zaimplementowanych funkcjonalności to pomysły nowe, gotowe do przetestowania poza środowiskiem deweloperskim.

## Rozważane formy wdrożenia projektu

Najlepsze rozwiązanie pod względem satysfakcji użytkownika końcowego stanowi stworzenie zestawu narzędzi w formie aplikacji internetowej. Aplikacja ta udostępniałaby API dla aplikacji mobilnych lub aplikacji przeglądarkowej. Główną zaletą tego pomysłu jest dostęp do usługi niezależnie od posiadanej platformy mobilnej. Dodatkowo istnieje możliwość dostosowania danej aplikacji mobilnej do ergonomii wskazanego systemu operacyjnego, co zapewniłoby najwyższą wygodę użytkownika.

Niestety wdrożenie tej idei byłoby zbyt pracochłonne na jednoosobowy projekt. Aby go wdrożyć, należałoby zastosować podział na kilka niezależnych komponentów, wśród których każdy wykorzystywałby inne technologie. Zazwyczaj takie projekty realizowane są we współpracy wielu specjalistów z różnych dziedzin.

Z powyższych względów, najlepszym sposobem na zrealizowanie pomysłu jest napisanie prostej aplikacji mobilnej działającej pod kontrolą systemu Windows Phone 8.1.

Czynniki, które zdecydowały o implementacji projektu w formie aplikacji to: zajęcia z programowania Windows Phone, znajomość platformy .NET, doświadczenie w tworzeniu interfejsów użytkownika z wykorzystaniem XAML’a.

## Jaki jest cel aplikacji?

Zależy mi, by osoba korzystająca z aplikacji mogła w każdej chwili przeglądać najważniejsze informacje o wybranych przez siebie aukcjach, a także dowiadywać się w czasie rzeczywistym o nowych ofertach pożyczkowych. Umożliwi jej to w ciągu kilku godzin podjąć decyzję o nowej inwestycji. Celem zasadniczym jest **skrócenie czasu potrzebnego do podpisania umowy o pożyczkę**. By tego dokonać, zaprezentuję inwestorowi najistotniejsze dane. Przypomnę, gdy znajdzie potrzeba, że pojawiła się kolejna oferta oraz pozwolę wyszukać aukcje, które najbardziej interesują inwestora.

## Założenia projektowe

Cała praca skupia się na ułatwieniu dostępu do korzystania z najpopularniejszego serwisu pożyczek społecznościowych w Polsce. Aplikacja ma być bezpieczna, stabilna, niezawodna. Szybkość działania jest kwestią drugorzędą, gdyż aplikacja operuje na zbyt małych zbiorach danych, by ich przetwarzanie miało powodować znaczące spowolnienia w responsywności. Myślą przewodnią jest dostarczenie wyłuskanych informacji w odpowiednim momencie.



---

**Kopra** jest zgodna z wytycznymi Microsoftu dot. projektowania aplikacji mobilnych dla systemu Windows Phone: „*Aplikacja ma robić jedną rzecz i ma ją robić dobrze*”[8] oraz „*Do more*”[9]. Oba hasła odnoszą się do skupienia na produktywności i komforcie użytkownika.

## Korzystanie z serwisów pożyczkowych na smartfonach

Na polskim rynku nie istnieje obecnie żadne rozwiązanie łączące świat mobilny z rynkiem pożyczek społecznościowych. Brak natychmiastowego dostępu do własnych inwestycji wywołuje stres i zniechęca do poważnej aktywności, ponieważ decyzje powinny być podejmowane jak najkrótszym czasie.

Poszukiwanie nowych ofert na przeglądarce telefonu jest bardzo niewygodne. Ponadto korzystanie na małym ekranie ze starych, nieresponsywnych interfejsów serwisów internetowych wymaga od użytkownika wiele cierpliwości. Instytucje finansowe nie wprowadzają ułatwień dostępu dla urządzeń mobilnych.

Jako osoba inwestująca wziąłem sprawy we własne ręce i stworzyłem aplikację mobilną. Teraz, aby uzyskać dostęp do najnowszych okazji pożyczkowych, wystarczy posiadać urządzenie wyposażone w system operacyjny Windows Phone 8.1 lub nowszy oraz połączenie z internetem.

Uważam, że **Kopra** ułatwi użytkownikom systemu Windows Phone 8.1 szybki dostęp do informacji.



# Rozdział 1.

## O pożyczaniu

### 1.1. Krótka historia pieniądza

Pożyczki istnieją od tysięcy lat. Pierwsze informacje o pożyczaniu pieniędzy wywodzą się ze starożytnej Mezopotamii [10].

Zajęcie było to tak popularne, że grupy osób zaczęły się specjalizować w pośrednictwie między tymi, którzy mieli pieniądze oraz tymi, którzy te pieniądze chcieli pożyczyć. Dzisiaj te grupy stanowią banki.

Wypaczając ideę pożyczania pieniędzy od osób posiadających pieniądze (pożyczkodawców) do osób biorących te pieniądze na pewien czas (pożyczkobiorców), banki stały się przedsiębiorstwami szukającymi zysków na obrocie pieniędzmi – nie tylko na ich pożyczaniu. Doprowadziło to do bardzo wielu niebezpiecznych zachowań.

Najgorszym z nich jest **system rezerw częściowych**. Działa on poprawnie do czasu, kiedy następuje zalanie rynku **toksycznymi pożyczkami**. Jednakże gdy to nastąpi, bank nie jest w stanie utrzymać płynności finansowej. Okazuje się wówczas, że większość jego pieniędzy to zabezpieczenia bez pokrycia, których nie da się wyegzekwować.

Kolejnym niebezpieczeństwem, jakie niesie ze sobą system rezerw częściowych, jest stałe zagrożenie wahaniami nastrojów społecznych. W wyniku plotki rozpущzonej wśród klientów banku bądź też czasowej niewypłacalności depozytów może nastąpić **zjawisko paniki bankowej**. Wielu ekonomistów zgadza się z poglądem, iż dobrze zarządzany system rezerw częściowych prowadzi do szybszego rozwoju gospodarczego i uniezależnia bogactwo kraju od złóż naturalnych.[11]

### 1.2. Idea pożyczek społecznościowych

Pożyczki społecznościowe w języku angielskim znane są jako „*peer-to-peer lending*” (w skrócie: P2P). Jest to zbiór czynności polegających na korelowaniu osób potrzebujących kredytu gotówkowego z ludźmi chętnymi do udzielenia części kredytu przez platformę online.[12]

Dla osób z dodatkową gotówką jest to sposób na naukę oceny ryzyka inwestycji niewielkim kosztem (stawki wejściowe do inwestycji są bardzo niskie). Szczególną zachętą jest tutaj relatywnie duży zwrot z inwestycji biorąc pod uwagę współczynnik ryzyka.

W Polsce serwisy pożyczek społecznościowych upowszechniły się, kiedy oprocentowanie lokat spadło poniżej 5% (zaraz po zablokowaniu lokat jednodniowych).[13] Spowodowało to nagły wzrost zainteresowania rynkiem pożyczek – wliczając w to mniej bezpieczne umowy z wyższym oprocentowaniem. Doprowadziło to do sytuacji obniżenia oprocentowania pożyczek kwalifikowanych jako „wysokiego ryzyka”, czyli takich o wyższej niespłacalności przy stosunkowo niskim oprocentowaniu. Sytuacja po roku ustabilizowała się.

Ostatecznie wszystkie serwisy pożyczek społecznościowych w Polsce poprawiły wymogi weryfikacji finansowej w celu udzielenia pożyczki. Ponadto oprocentowanie maksymalne zostało obniżone

z 25% do 10% (stan na październik 2016). W efekcie instrumenty te stały się niewiele bardziej opłacalne od lokat sprzed czasów ustawy blokującej omijanie podatku od dochodów kapitałowych.

### 1.3. Ryzyka związane z pożyczkami społecznościowymi

Jako ryzyka związane z inwestowaniem w rynek pożyczek można wymienić:

- sprzeczne interpretacje prawne Naczelnego Sądu Administracyjnego przy czerpaniu zysków z pożyczek społecznościowych,
- problemy ze ściąganiem wierzytelności.

W akapitach poniżej znajduje się krótkie omówienie tych problemów oraz sposoby rozwiązania ich.

#### Urząd Skarbowy

Zyski z przychodów kapitałowych są opodatkowane 19-procentową stawką (na rok 2016). Jediną jasną kwestią jest to, iż od każdego wypracowanego zysku należy zapłacić podatek.

Problem pojawia się z interpretacją prawa – kiedy należy założyć działalność gospodarczą oraz w jaki sposób rozliczać się z Urzędem Skarbowym i fiskusem. Naczelny Sąd Administracyjny w sprawie o sygnaturze II FSK 1472/10 orzekł, że regularne inwestowanie w pożyczki nosi znamiona działalności gospodarczej.[14] Jednak w sprawie o sygnaturze II FSK 142/10 ten sam sąd wydał wyrok odwrotny, uznając wówczas, że pożyczanie pieniędzy nie nosi znamion prowadzenia działalności gospodarczej.[15]

Mając na uwadze powyższe wyroki, przed rozpoczęciem regularnego pożyczania, *należy zwrócić się do dyrektora właściwej sobie Izby Skarbowej z zapytaniem o sposób rozliczania*.[16]

#### Windykacja

Jak wygląda odzyskiwanie wierzytelności przy braku spłat? Pierwszą rzeczą, którą należy zrobić to próbować rozmowy z wierzycielem, wyjaśnić przyczyny opóźnień i znaleźć sposób na najszybszy powrót do regularnych spłat.

Jeśli kontaktu nie można nawiązać natychmiast, można skorzystać udostępnionych w serwisach monitów. Monit jest przypomnieniem o zaległych ratach, wysyłanym w formie SMS/telefonu/wiadomości email. Jednakże dla spłacającego monit stanowi dodatkowe obciążenie finansowe, dlatego nie jest to rozwiązanie zalecane jako pierwsza forma „kontaktu”.

Kolejny krok stanowi podjęcie działań prawnych. Na tym etapie zaczynają się problemy dla inwestora, gdyż musi podjąć decyzję, czy ścigać należności samodzielnie, wynająć firmę prawniczą lub też skorzystać z usług windykacyjnych dostępnych w serwisie.

Dobrym rozwiązaniem jest wspólna windykacja większości pożyczkodawców aukcji. Polega ona na tym, że inwestorzy wspólnie wynajmują firmę windykacyjną, gdzie za 10-15% odzyskanej kwoty mogą liczyć na większe prawdopodobieństwo odzyskania pieniędzy. Jeśli firma nie odzyska długu, nie dostaje wynagrodzenia.

### 1.4. Jakie zyski można osiągnąć?

**Średnie oprocentowanie** w serwisie **Kokos** wynosi 16,87% [17] (na dzień 28.10.2016).

**Stopa lombardowa** ustalona przez Narodowy Bank Polski wynosi 2,5% [18] (ustalona w dniu 05.03.2015).

**Maksymalne oprocentowanie** dla pożyczki w skali roku to stawka stopy lombardowej pomnożona razy cztery, czyli 10%.

Średnie oprocentowanie w serwisie jest wyższe niż 10%, ponieważ jest liczone od czasu powstania serwisu, kiedy to stawka lombardowa wynosiła 6,25%, co dawało maksymalny zysk do 25% rocznie.

Stawka oprocentowania rośnie również wówczas kiedy pożyczka zostaje spłacona przed czasem. Jest to możliwe szczególnie w przypadku gdy PB buduje swoją reputację i zobowiązuje się spłacić 3-letnie zobowiązanie w ciągu 1 roku.

Z moich osobistych obserwacji wyłania się obraz zwrotu z inwestycji na poziomie 12% rocznie, wliczając w to niespłacone zobowiązania. Warto zauważyć, iż w trakcie całej operacji inwestujemy także swój czas. Rozważając opłacalność pożyczek społecznościowych należy wliczyć również czas poświęcony na analizę i pilnowanie inwestycji oraz czas na kontaktowanie się z pożyczkobiorcą i innymi wpłacającymi. Zysk przekraczający zainwestowany czas można zaobserwować przy kwotach inwestycji powyżej 500 zł miesięcznie.

Utrzymywanie się z odsetek zarobionych na pożyczkach wymaga pełnoetatowego zaangażowania w zarządzanie narastającą ilością pożyczek. W takim przypadku można zająć się również inwestowaniem w akcje.

## 1.5. Porównanie najpopularniejszych instrumentów finansowych

Zysk rośnie wraz z ryzykiem. Przy inwestowaniu aktywów zawsze istnieje ryzyko straty części lub całości inwestycji. Można zaobserwować niemal wykładniczą zależność między ryzykiem utraty inwestycji, a wysokością osiągniętych zysków.

Poniżej prezentuję prostą tabelę z porównaniem przykładowych zysków i ryzyka dla poszczególnych instrumentów finansowych.

<i>Właściwość</i>	<b>Obligacje</b>	<b>Lokata</b>	<b>Fundusz inwestycyjny</b>	<b>Pożyczka społecznościowa</b>	<b>Giełda</b>
<b>Ryzyko</b>	Brak	Brak	Niskie	Średnie	Wysokie
<b>Rentowność</b>	Znikoma	Niska	Średnia	Średnia, Wysoka	Wysoka
<b>Czas inwestycji</b>	Min. 60 mies.	3–12 mies.	Min. 60 mies.	12–36 mies.	Min. 36 mies.
<b>Poziom skomplikowania</b>	Podstawowy	Podstawowy	Łatwy	Średni	Wysoki

## 1.6. Porównanie polskich serwisów pożyczkowych

Istniejące serwisy w Polsce to:

1. Finansowo.pl [19]
2. Zakra.pl [20]
3. Kokos.pl [21]
4. Lendico.pl [22]
5. Zakramini.pl [23]
6. AppleCredit.pl [24]
7. CapitalClub.pl [25]

<i>Serwis</i>	<b>Użytkownicy</b>	<b>Udzielone pożyczki</b>	<b>Splacone zobowiązania</b>	<b>Splacalność</b>	<b>Średnie oprocentowanie</b>	<b>Charakter</b>
<b>Finansowo</b>	b/d	b/d	85 813 tys. zł	91,9%	5,49%	Chwilówki
<b>Zakra</b>	39 tys.	b/d	b/d	b/d	10%	Konsument
<b>Kokos</b>	309 tys.	126 tys.	144 426 tys. zł	93%	16,86%	Konsument
<b>Zakramini</b>	17 tys.	44 tys.	13 706 tys. zł	b/d	b/d	Chwilówki
<b>AppleCredit</b>	8 tys.	2 tys.	1 135 tys. zł	b/d	9%	Chwilówki
<b>CapitalClub</b>	1 tys.	0,1 tys.	4 000 tys.	b/d	b/d	Inwestycje

W zestawieniu brakuje serwisu Lendico. Ten niemiecki serwis zawiesił działalność inwestycyjną – od kilku miesięcy nie ma ofert inwestycyjnych. Jest prawdopodobne, że ze względu na niskie zainteresowanie zostanie wkrótce wycofany z polskiego rynku.

## 1.7. Wybór systemu do implementacji

Pierwszym i obecnie najpopularniejszym serwisem jest **Kokos**. W ciągu 8 lat działalności pożyczono 143,4mln zł na 126 738 aukcjach. Dla porównania, CapitalClub informuje na stronie głównej o sumie 4 mln zł przez 102 udzielone pożyczki dla firm.

Kokos, którego właścicielem jest firma Bluemedia, posiada:

- największą bazę aktywnych inwestorów,
- wiele innowacyjnych zabezpieczeń, takich jak np. FriendlyScore,
- responsywny interfejs,
- przelewy ekspresowe,
- raporty inwestora,
- podstawowe API do zapytań o aukcję.

Ważnym aspektem jest również **dostęp do interfejsu programistycznego udostępnionego dla programistów**. Ze wszystkich platform, API jest udostępnione na trzech – Kokos, Zakra, Zakramini.

Zakra i Zakramini pozwalają korzystać programiście lub inwestorowi z zapytań do serwisu po osobistej prośbie o wygenerowanie klucza. Stanowi to wielką przeszkodę, gdyż każdy kto chce sprawdzić działanie funkcjonalności jest zobowiązany wystosować pisemną prośbę o wygenerowanie klucza i otrzyma dostęp do WebAPI dopiero po upływie kilku dni.

Kolejnym mankamentem jest bardzo okrojony zakres zwracanych informacji. Dostępne są następujące wywołania (wraz z odpowiedziami):

### getUserData(id użytkownika)

[id użytkownika, nazwa użytkownika, rok urodzenia, miasto zameldowania, id użytkownika w serwisie Zakramini, nazwa użytkownika w serwisie Zakramini, kwota zadłużenia w serwisie Zakramini, czas opóźnień spłaty na w serwisie Zakramini]

**getAuctionData(id aukcji)**

[id, id pożyczkobiorcy, oprocentowanie, kwota pożyczki, numer aukcji, bonus pożyczkodawcy, status aukcji, data utworzenia]

**getRunningLoansFromUser(id użytkownika)**

[tablica z aukcjami jak getAuctionData]

**getAuctionInstallmentsData(id aukcji)**

[id aukcji, dzień płatności, data pierwszej spłaty, ilość spłaconych rat, ilość zaległych rat]

Zwracają one tylko najbardziej podstawowe informacje [26], właściwie szcątkowe. Nie można z nich złożyć porządnej informacji kredytowej.

Pozostaje jeszcze Kokos ze swoim interfejsem. Pozwala na uzyskanie znacznie bardziej złożonych informacji o aukcji, użytkownikach, stanie finansowym. Wygenerowanie klucza jest niesamowicie proste – wystarczy kliknięcie w przycisk „Wygeneruj klucz” i od razu otrzymujemy klucz aktywny przez rok.

Mam nadzieję, że po przedstawieniu powyższych informacji jest jasne, że wybór, który serwis obsłużyć, tak naprawdę nie istnieje.

## 1.8. WebAPI

Interfejs programistyczny wchodzi w skład narzędzi użytkowników. Dokumentacja [27] jest prawie kompletna (parę brakujących elementów można szybko wywnioskować z kontekstu).

Aby rozpocząć korzystanie, należy wygenerować własny klucz i zapoznać się z regulaminem korzystania oraz dokumentacją użytkownika.

**Dokumentacja** posiada formę listy funkcji dostępnych dla programisty. Po wybraniu konkretnej metody otrzymuje się:

- stronę z opisem argumentów (parametrów) funkcji,
- listę zwracanych elementów,
- sposób użycia,
- przykładowe zapytanie i odpowiedź w formie XML.

Odpowiedź na zapytanie można otrzymać w formacie XML, JSON lub HTML. Zapytania wykonuje się w modelu REST – wszystkie parametry są przekazywane przy użyciu zapytania GET w postaci:

```
https://kokos.pl/webapi/search?key=<klucz>&<parametry>
```

Użytkownik może wykonywać maksymalnie **jedno zapytanie na sekundę**. Przy częstszych odpytaniach serwis zgłosi błąd 503 „*Service Temporarily Unavailable*”. Ta sama sytuacja pojawi się, kiedy stronę internetową odświeża się w przeglądarce zbyt często.

Do dokumentacji dołączony jest również **słownik oznaczeń** używanych w zapytaniach i odpowiedziach. Istnieje część słownika, która nie posiada możliwości dynamicznego pobierania celem aktualizowania na jego podstawie interfejsu użytkownika. Tę część należy wpisać na stałe w aplikacji. Dotyczy to jednak elementów niezmiennych, takich jak: nazwy województw, statusy spłat oraz statusy aktywności pożyczki.

Istotne jest także pamiętanie o wygenerowaniu klucza co roku. Aplikacje korzystające z klucza muszą go odnowić. Data ważności klucza jest wyświetlana na stronie dokumentacji w zakładce „*Klucz WebAPI*”. Niestety próba odświeżenia klucza przed jego wygaśnięciem nic nie zmienia – należy odczekać do jego wygaśnięcia, by móc aktywować go ponownie na rok.

Odpowiedzi z serwera bywają nieprzewidywalne. Podczas tworzenia aplikacji na zapytanie o listę inwestorów, otrzymywałem listę w formacie JSON. Jednak kiedy inwestorów nie było, zamiast pustej listy otrzymywałem jeden pusty element.

## 1.8. WEBAPI

---

W okolicy 10 dnia każdego miesiąca o wieczornej porze należy uzbroić się w cierpliwość, gdyż wówczas inwestorzy szturmują serwis w poszukiwaniu najlepszych aukcji. Wtedy z powodu opóźnień, częściej zamiast spodziewanej odpowiedzi otrzymamy błąd 503.



## Rozdział 2.

# Wykorzystane technologie i narzędzia

W tym rozdziale przedstawiam technologie i narzędzia, które znacząco pomogły mi w realizacji projektu.

### 2.1. Microsoft Visual Studio Community 2015

Miejscem, w którym spędziłem najwięcej czasu, napisałem i złożyłem projekt jest Visual Studio (VS) w wersji Community Edition 2015. Jest to obecnie najbardziej podstawowa ze wszystkich wersji IDE (*Integrated Developer Environment*) przeznaczonych do tworzenia aplikacji mobilnych na telefony z systemem Windows Phone 8.1. Narzędzia dostępne w tym oprogramowaniu w zupełności wystarczają, by tworzyć aplikacje takie jak **Kopra**.

Należy pamiętać, że VS jest środowiskiem wymagającym sporych zasobów sprzętowych. Na stronie producenta [28] można znaleźć informacje o minimalnych wymaganiach: procesor 1.6 GHz lub szybszy, 1 GB pamięci RAM, 4 GB miejsca na dysku twardym, karta graficzna wspierająca DirectX 9.0c i rozdzielczość wynosząca przynajmniej  $1024 \times 768$  pikseli. Jednak aby móc programować urządzenia Windows Phone, potrzebna jest dużo większa moc obliczeniowa – w szczególności jeśli programuje się przy użyciu emulatorów.

### 2.2. Język C# 5.0

Język programowania Microsoft Visual C# został zaprojektowany z myślą o tworzeniu aplikacji działających z zestawem narzędzi .NET Framework. Firma Microsoft projektowała go jako potężny, a zarazem prosty język orientowany obiektowo z bezpiecznym typowaniem.[29] Składnia wywodzi się z rodziny języków C. Dziedziczy wiele najlepszych cech języka C++ i Microsoft Visual Basic, odrzucając większość ich anachronizmów, a tworząc bardziej logiczną strukturę języka.

Pierwsza wersja oficjalnie zadebiutowała w 2001 roku. Wielu programistów uważało ją za próbę skopiowania Javy, jednak kolejne wersje przewyższają prostotą składni i klarownością instrukcji swój domniemany pierwowzór.

Premiera C# 2.0 ze środowiskiem Visual Studio 2005 wprowadzały kilka nowych elementów do języka, takich jak: typy generyczne, iteratory oraz metody anonimowe.

Kolejna wersja (C# 3.0) dzięki licznym innowacjom przyczyniła się do zwiększenia popularności języka wśród programistów. Wszystko to przez nowatorskie konstrukcje, które konkurencja wprowadzała przez kilka następnych lat. Moją ulubioną nowością tego wydania jest LINQ (*Language-Integrated Query*).

Również wersja C# 5.0 daje wrażenie dynamicznego rozwoju języka. Projektanci ułatwili tworzenie wywołań asynchronicznych przez dodanie modyfikatorów metod `async/await`.

## 2.3. .NET FRAMEWORK

---

Podsumowując, kolejne wersje języka wprowadzają udogodnienia pomagające pokonywanie największych wyzwań stojących przed programistą. Każda wersja jest ściśle powiązana ze środowiskiem Visual Studio C# oraz zestawem narzędzi .NET Framework, choć numeracja wersji nie zawsze się pokrywa.

## 2.3. .NET Framework

W odniesieniu do zmian wprowadzonych przez Microsoft w 2016 roku (ze względu na wprowadzenie nowego podziału architektury .NET), ograniczam się do Standardowej Biblioteki .NET (ang. *.NET Standard Library*) oraz do .NET Framework ograniczony do podzbioru WPF.[30]

.NET Framework jest platformą programistyczną wydaną w 2002 roku przez Microsoft. Do niedawna była to jedyna platforma dla programistów .NET.[30] W platformie znajduje się zestaw klas biblioteki oraz API systemu Windows, co znacząco podnosi produktywność programowania aplikacji typu desktop.

Kolejne wydania platformy .NET są ściśle powiązane z nowymi wydaniem języka C#. Wersja Frameworka używana przez Koprę została oznaczona numerem 4.5.1 i jest kompatybilna z Visual Studio 2013 na Windows 8.1 wraz z wersjami późniejszymi.

## 2.4. Windows Phone 8.1 SDK

Microsoft ukrył stronę z Windows Phone 8.1 SDK oraz zestawem emulatorów na rzecz promocji Windows 10 Mobile. Dlatego zestaw narzędzi jest dość trudny w odnalezieniu. By się dostać do zasobu, musiałem użyć zewnętrznej wyszukiwarki, ponieważ ta ze strony Microsoft.com nie zwraca wyników. Celem zdobycia zestawu narzędzi programistycznych na tę platformę, trzeba przeszukiwać archiwum Microsoftu. [31]

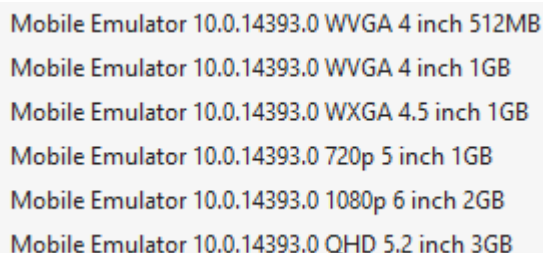
Do programowania na system Windows Phone potrzebne jest urządzenie fizyczne (preferowana Nokia Lumia) bądź emulator, do którego wymagany jest procesor ze wsparciem SLAT (*Second Level Address Translation*). Należy również pamiętać, że do komfortowej pracy z emulatorem potrzeba przynajmniej 8 GB pamięci RAM. Zestaw SDK zawiera 6 różnych konfiguracji emulatorów (rysunek 2.1):



Rysunek 2.1: Zestaw konfiguracji emulatorów dostępny w Windows Phone 8.1 SDK

Aplikację przetestowałem na każdym z wyżej wymienionych urządzeń wirtualnych, a także na fizycznych urządzeniach: Nokia Lumia 925, Microsoft Lumia 530, Nokia Lumia 625.

**Kopra** została również uruchomiona w emulatorze systemu Windows 10 Mobile i działa poprawnie w konfiguracjach zaprezentowanych poniżej (rysunek 2.2). Nie została przetestowana dokładnie na nich. Podobnie nie miałem dostępu do urządzeń fizycznych z systemem Windows 10 Mobile, stąd użytkownik powinien używać oprogramowania na tym systemie na własną odpowiedzialność.



Mobile Emulator 10.0.14393.0 WVGA 4 inch 512MB  
 Mobile Emulator 10.0.14393.0 WVGA 4 inch 1GB  
 Mobile Emulator 10.0.14393.0 WXGA 4.5 inch 1GB  
 Mobile Emulator 10.0.14393.0 720p 5 inch 1GB  
 Mobile Emulator 10.0.14393.0 1080p 6 inch 2GB  
 Mobile Emulator 10.0.14393.0 QHD 5.2 inch 3GB

Rysunek 2.2: Zestaw konfiguracji emulatorów dostępny w Windows 10 Mobile SDK

## 2.5. Newtonsoft.Json

Newtonsoft.Json stanowi najpopularniejsze narzędzie do pracy z danymi w formacie JSON. Oprogramowanie działa na jednej z najbardziej liberalnych licencji, MIT.[32] Wśród bibliotek dostępnych w menadżerze pakietów NuGet, zajmuje pierwsze miejsce mając ponad 15 milionów pobrań.[15]

Json.NET znacznie uprościł proces przetwarzania danych otrzymanych z serwisu. Zgodnie z opisem zamieszczonym na stronie projektu, biblioteka jest bardzo łatwa w użyciu, nawet w przypadku złożonych typach danych. Znalazłem się jednak kilkakrotnie w sytuacji, gdy musiałem rozbudować mechanizm serializujący o dodatkowe warunki. Działo się tak kiedy serwis zwracał mi różne rodzaje danych dla jednego zapytania.

Do pobierania elementów ze struktury HTML, których nie można pobrać przez API serwisu, istnieje mechanizm wspierający XPath. JSONPath [33] jest wbudowany w bibliotekę i świetnie wspiera pobieranie elementów wg ścieżki oraz natychmiastowe serializowanie ich do formatu JSON.

## 2.6. Git i GitHub

W utrzymaniu ładu podczas pracy z kolejnymi wersjami aplikacji pomógł mi Git. Jest to rozproszony system kontroli wersji, stworzony przez Linusa Torvaldsa jako narzędzie do zarządzania kodem źródłowym Linuxa. Motywacją autora do napisania własnego systemu kontroli wersji było pogorszenie relacji z firmą BitMover, gdyż w efekcie projekt Linux stracił możliwość darmowego korzystania z BitKeeper'a.

Git został zaprojektowany w oparciu o 5 cech, które do dziś stanowią jego podstawę: szybkość, prosta konstrukcja, wsparcie dla równoległego rozwoju wielu wersji jednocześnie, w pełni rozproszony, z możliwością zarządzania wielkimi projektami.[1]

Przekonującym argumentem do użycia właśnie tego systemu kontroli wersji jest dostęp do serwisów zintegrowanych z oprogramowaniem. Zmiany przechowywane są lokalnie w formie lokalnej bazy. Większość akcji wykonywanych na repozytorium Git to dodawanie danych do bazy. Sprawia to, że zniszczenie czegoś bez możliwości przywrócenia stanu jest prawie niemożliwe. Dzięki temu można śmiało dokonywać zmian w projekcie bez obaw o utratę danych.

Tworząc większe funkcjonalności w projekcie korzystałem z możliwości tworzenia własnych gałęzi (*branch*), a następnie po zaimplementowaniu łączyłem je do głównej linii produkcyjnej (*master*).

Zamiast korzystać z własnego serwera systemu kontroli wersji, użyłem darmowego i najbardziej popularnego serwisu oferującego usługi przechowania repozytorium Git – GitHub.com. Na chwilę obecną jest on największą platformą hostującą projekty open-source. Ponad 18 milionów użytkowników i 48 milionów założonych projektów pozwala twierdzić, że w trakcie tworzenia projektu serwis nie zostanie nagle wyłączony z użycia. Usługa oferuje możliwość przechowania kodu widocznego dla wszystkich (*Public*) oraz tylko dla wybranych użytkowników (*Private*). GitHub jest darmowy dla pierwszego typu hostingu.[34]

Również kod źródłowy Kopry jest widoczny dla każdego – jeśli ktoś zechce wykonać własną gałąź aplikacji i w jakikolwiek sposób ją poprawić, będzie to dla mnie miłym zaskoczeniem.

## 2.7. LinqPad

Narzędzie LinqPad wykorzystałem w celu szybkiego modelowania kodu metod. Jest to produkt powszechnie używany do modelowania fragmentów kodów w środowisku .NET.

Zaletę programu stanowi możliwość pisania interaktywnych zapytań SQL z wykorzystaniem LINQ oraz pisania kodu C# bez konieczności uruchamiania Visual Studio.[35]

Produkt jest tworzony na modelu Freemium, co oznacza, że program jest darmowy z wyłączeniem pewnych funkcjonalności, za które należy zapłacić.[36]

## 2.8. ILSpy

ILSpy to przeglądarka plików assembly i dekompilekator kodu .NET, oparty o licencję MIT. Projekt hostowany jest na GitHub, gdzie można pobrać jego źródła.[37]

Początkowo używałem tego narzędzia do dekompilacji biblioteki *HtmlAgilityPack*, która przez wzgląd na zbyt wiele błędów zmusiła mnie do przeglądania jej wewnętrznych metod. Na późniejszym etapie ILSpy pomógł mi dokładniej zrozumieć część biblioteki .NET zawartej w przestrzeni nazw `System.Net.Http`.

ILSpy stał się szczególnie popularny, kiedy narzędzie Microsoftu o nazwie IL DASM przestało stanowić część pakietu Visual Studio. Przed wersją Visual Studio 2012, IL DASM był dostępny z poziomu folderu *Microsoft SDK Tools* jako narzędzie udostępnione w zestawie razem z IL ASM. Ten zaś służy jako narzędzie do generowania kodu wykonywalnego z kodu pośredniego. Obecnie te narzędzia dostępne są w formie konsolowych programów.

ILSpy jest w pełni graficznym narzędziem. Uważam, że stanowi najlepsze darmowe rozwiązanie dla przeglądania plików assembly.

## Rozdział 3.

# Budowa projektu

Projekt został zbudowany w oparciu o bibliotekę Windows Phone SDK z wykorzystaniem wzorca architektonicznego MVVM. Z narzędzi zewnętrznych, zapewniłem integrację z Web API serwisu Kokos.

### 3.1. Wzorzec MVVM

W tym rozdziale przedstawię wzorzec architektoniczny MVVM, jego zalety i wady oraz sposób w jaki zaimplementowałem go w projekcie.

#### 3.1.1. Opis wzorca Model View ViewModel

Wprowadzony przez Microsoft w 2005 roku, wzorzec architektoniczny, opiera się na tych samych założeniach co zaprezentowany lata wcześniej model prezentacji (Presentation Model). PM dziedziczy po modelu Model Widok Prezenter MVP – (ang. *Model View Presenter*), wzorcu architektonicznym z początku lat 90 ubiegłego wieku. Z tym, że wzorzec PM jest szczególnie przydatny w programowaniu złożonych interfejsów użytkownika. Pod systemem Windows, Presentation Model bardzo dobrze współpracuje z interfejsem użytkownika zbudowanym z silnikiem graficznym Windows Presentation Foundation (WPF) oraz biblioteką Silverlight. Właśnie dlatego Microsoft opracował wersję wzorca PM specjalnie dla WPF i nazwał go Model-View-ViewModel (MVVM).[2] MVVM używa wielu specyficznych możliwości i narzędzi znanych z platformy Silverlight, takich jak na przykład wiązanie danych. Implementacja wzorca MVVM w aplikacjach opierających się na technologiach WPF, Silverlight jest zalecanym podejściem, ponieważ obie technologie są wyposażone w narzędzia, które ułatwiają poprawną implementację tego wzorca.

Do zalet Model View ViewModel względem tradycyjnego podejścia (przeciągania kontrolki z Toolboxa i pisanie dalej w formie logiki strony (ang. *Code-behind*) są:

- Umożliwia oddzielenie logiki od sposobu wyświetlania. W dłuższej perspektywie ułatwia to rozwój, testowanie i utrzymanie kodu.
- Jest wspierany przez platformy XAML.
- Kod interfejsu jest niezależny od platformy kodu wykonywalnego.
- Umożliwia podział zadań między projektanta interfejsu, który tworzy widoki i programistę, który tworzy logikę aplikacji. Dzięki takiemu podziałowi prac każdy z nich może tworzyć oddzielnie część aplikacji, za którą jest odpowiedzialny.

Wzorzec podzielony jest na trzy części, które wspierają testowanie aplikacji przez rozdzielenie elementów interfejsu użytkownika od logiki aplikacji.

**Widok (*View*)** odpowiada za strukturę, rozkład i wygląd tego co użytkownik widzi na ekranie.

W większości przypadków aplikacji na system Windows Phone, widok zdefiniowany jest jako „strona” zaimplementowana przy użyciu XAML’a. Widokiem można manipulować przez logikę strony lub przez widok-model.

**Model (*Model*)** jest odpowiedzialny za przechowywanie danych w obrębie swojej domeny oraz logiki biznesowej.

**Widok-model (*ViewModel*)** rozdziela logikę biznesową od widoku. Zadaniem tego komponentu jest pobranie danych z modelu i przedstawienie go w formie łatwo przyswajalnej dla widoku. Widok-model również może odpowiadać za stany widoku, jak na przykład przez przekazanie widokowi stanu ładowania.

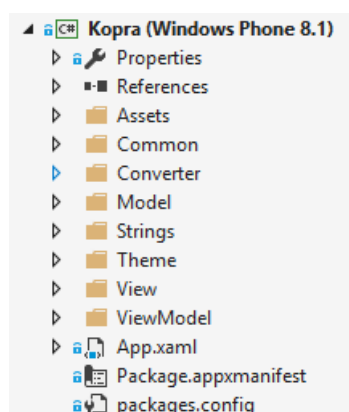
Taki rozdział pozwala na wymianę komponentów aplikacji, wewnętrzną zmianę implementację, wyizolowane testy jednostkowe. Komponenty na najwyższym poziomie abstrakcji nie wiedzą o sobie zbyt wiele. Widok (V) wie o istnieniu **widok-modelu** (ang. *ViewModel*) (VM), widok-model (VM) wie o istnieniu modelu, jednak model nie wie o widok-modelu (VM) i widok-model (VM) nie wie nic o Widoku (V).

#### 3.1.2. Porównanie MVVM i MVC

Praca nie byłaby kompletna, gdybym nie wspomniał nic o wzorcu, który wpłynął w znaczący sposób na rozwój MVVM. Wzorec bardzo podobny, który powstał jednak dużo wcześniej to Model-View-Controller (MVC). Jest to wzorec architektoniczny, używany do implementacji interfejsu użytkownika. Polega na podziale oprogramowania na trzy niezależne od siebie części. Wzorec ten został zaprezentowany po raz pierwszy przez Trygve Reenskaug jeszcze w latach 70-tych.[38] Wzorec dzieli modelowanie domeny, prezentacji i akcji na trzy oddzielne klasy. Model zarządza zachowaniem i danymi w domenie aplikacji, odpowiada na żądania informacji o swoim stanie oraz odpowiada na żądania zmiany stanu, które otrzyma od kontrolera. Widok (View) stanowi o graficznej części aplikacji, otrzymuje od kontrolera informacje w jaki sposób interpretować dane wejściowe otrzymane od użytkownika. Kontroler (Controller) służy do przetwarzania informacji otrzymywanych od użytkownika np. z urządzeń wejściowych. MVC stał się bardzo popularnym wzorcem przy projektowaniu aplikacji typu desktop, a później także Web.

#### 3.1.3. Implementacja MVVM w aplikacji

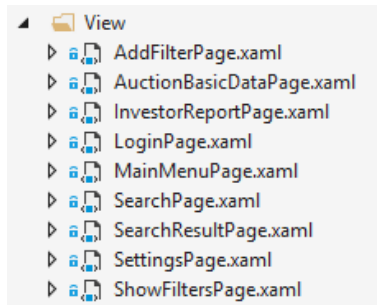
Aplikacje korzystające z wzorca MVVM mają z góry ustalony szkielet folderów.



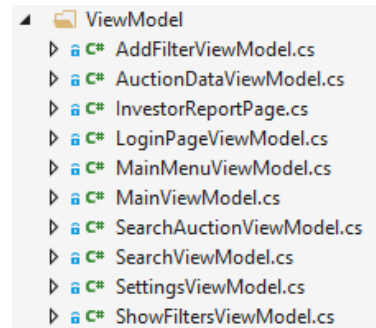
Rysunek 3.1: Struktura folderów w projekcie aplikacji Kopra

Mój projekt ściśle trzyma się tej struktury wraz z dodatkowymi elementami. Przyjąłem następującą konwencję nazewnictwa, dla każdego widoku dołączyłem sufiks „Page”, natomiast każdy

widok-model w nazwie ma dołączony sufix „ViewModel”. Każdy widok ma odpowiadający mu jeden widok-model.



Rysunek 3.2: Lista plików widoków w projekcie



Rysunek 3.3: Lista plików widok-modeli w projekcie

```
<Page.DataContext>
    <viewModel:NameViewModel></viewModel:NameViewModel>
</Page.DataContext>
```

Listing 3.1: Podpięcie widok-modelu do widoku w pliku XAML

```
private AddFilterViewModel _viewModel;

public AddFilterPage()
{
    _viewModel = DataContext as AddFilterViewModel;
}
```

Listing 3.2: Podpięcie widok-modelu w pliku logiki strony

W celu dodania mechanizmów zarządzających wiązaniem danych, utworzyłem klasę bazową `MainViewModel` implementującą interfejs `INotifyPropertyChanged`.

```
public class MainViewModel : INotifyPropertyChanged
{
    public event PropertyChangedEventHandler PropertyChanged;

    public void NotifyPropertyChanged(string propertyName)
    {
        PropertyChanged?.Invoke(this,
            new PropertyChangedEventArgs(propertyName));
    }
}
```

Listing 3.3: Implementacja interfejsu `INotifyPropertyChanged` w klasie `MainViewModel`

Po wspomnianej klasie dziedziczą pozostałe widok-modele. Używają one metody `NotifyPropertyChanged()` do odebrania powiadomień o zmianach wartości dowiązanych własności.

```
public string ApiKeyValue
{
    get { return _apiKeyValue; }
    set
    {
        _apiKeyValue = value;
        NotifyPropertyChanged(nameof(ApiKeyValue));
    }
}
```

Listing 3.4: Przykładowa właściwość wykorzystująca mechanizm NotifyPropertyChanged

## 3.2. Kokos WebAPI

Celem niniejszego rozdziału jest na prezentacja interfejsu programistycznego udostępnionego przez zespół BlueMedia. Przedstawiam w nim również niedogodności związane z wykorzystaniem API serwisu, sposób wykorzystania przez Kopkę oraz sposoby poprawienia komfortu użytkownika. Przed rozpoczęciem pracy z interfejsem programistycznym Kokosa jedynym wymogiem (oprócz aktywnego konta) jest wygenerowanie klucza, który służy pozwala odróżnić kto wykonuje zapytanie.

### 3.2.1. Opis interfejsu webowego

Zapytania do serwisu wykonuje się przez złożenie i wysłanie zapytania REST-owego. Czyli

`https://kokos.pl/webapi/FUNKCJA?key=KLUCZ&type=FORMAT_DANYCH&PARAMETR=WARTOŚĆ`

Aby uniknąć nadmiernego obciążenia i wykorzystania klucza przez więcej niż jednego użytkownika liczba zapytań jest ograniczona do maksymalnie jednego na sekundę. Przy próbie przekroczenia, zostanie zwrócony w odpowiedzi kod 503. Lista dostępnych funkcji:

#### **get-auction-data**

Pobranie szczegółowych danych jednej wybranej aukcji.

#### **get-auctions-by-status**

Pobranie listy pożyczek według statusu.

#### **search**

Wyszukiwanie aukcji według podanych parametrów.

#### **get-most-popular-auctions**

Pobranie najbardziej popularnych aukcji.

#### **get-recent-auctions**

Pobranie ostatnio założonych aukcji.

#### **get-recent-payments**

Pobranie ostatnich spłat za raty.

#### **get-recent-investments**

Pobranie najnowszych inwestycji.

#### **get-ended-auctions-amount**

Pobranie sumy wartości aukcji zakończonych sukcesem w ciągu ostatnich dni, których liczba jest parametrem przekazywanym w metodzie.



**get-service-stats**

Pobranie statystyk serwisu.

**get-user-id-by-nick**

Pobranie identyfikatora użytkownika na podstawie jego nick-u.

**get-payment-stats**

Pobranie statystyk spłacalności dla całego serwisu.

**get-vindication-stats**

Pobranie statystyk spłacalności pożyczek windykowanych.

Każda z funkcji ma określony spory zakres parametrów oraz tabelę zawierającą strukturę i typ zwracanych wartości.

### 3.2.2. Implementacja API w aplikacji

W celu uniknięcia powielania kodu podczas zapytań, stworzyłem klasę, która generuje zapytania do serwisu. Wykorzystuje ona mniej niż połowę z wymienionych powyżej funkcji, dlatego nie widziałem powodu by tworzyć osobny projekt na komunikację z serwisem. Elementy stałe przechowuję w formie prywatnych elementów klasy.

```
private const string BaseAddress = "https://kokos.pl/webapi/";
private const string DataType = "&type=json";
private const string Search = "search?";
private const string RecentAuctions = "get-recent-auctions?";
private const string AuctionData = "get-auction-data?";
private const string Comments = "comments=1";
private const string Records = "records=";
private const string Key = "key=";
private const string Id = "id=";
private const string Type = "type=";
```

Listing 3.5: Składowe budujące zapytanie do serwisu

Tworzenie parametrów zapytań API ukryte jest w klasie `RequestGenerator`. Przechowywany jest tam zbiór funkcji do tworzenia całościowych obiektów klasy `Uri`, gotowych do przekazania użycia przez serwis.

```
public Uri FilteredAuction(string filter)
public Uri ComposeSearchAuctionQuery(Dictionary<string, string> search)
public Uri MostRecentAuctions()
public Uri GetAuctionData(GetAuctionDataParameters parameters)
```

Listing 3.6: Deklaracje metod pozwalających składać zapytania do serwisu.

Komunikacja z serwisem odbywa się przy pomocy klasy `KokosConnectionManager`. Do komunikacji internetowej, wykorzystana jest klasa `HttpClient`.

## 3.3. Notyfikacje przez BackgroundTask

Windows Phone 8.1 pozwala wykonywać aplikacji zadania bez konieczności trzymania jej na pierwszym planie. Funkcjonalność ta nazywa się *Background Task* i użyłem jej do powiadamiania użytkownika o nowych aukcjach w serwisie.

Aby zaimplementować notyfikacje o nowych aukcjach, utworzyłem nowy projekt w Visual Studio typu *Windows Runtime Component*. W projekcie zawierającym aplikację, podpiąłem referencję

### 3.3. NOTYFIKACJE PRZEZ BACKGROUNDTASK

---

do nowo utworzonego projektu, dodałem do właściwości (ang. *Capabilities*) aplikacji deklarację wsparcia dla *Background Task*. Na koniec określiłem punkt wejściowy czyli klasę, która implementuje interfejs *IBackgroundTask*.

Kod między projektem aplikacji i projektem zadania w tle nie jest współdzielony przez problemy techniczne, które napotkałem przy próbie tworzenia biblioteki służącej do współdzielenia kodu (*Portable Class Library*). Do korzystania z funkcjonalności pobierania i przetwarzania aukcji, zdecydowałem się na utworzenie osobnych klas. Są one okrojona wersją klas pochodzących z projektu **Kopra**. Są to następujące klasy *Auction*, *Auctions*, *Paging*, *RequestGenerator*, *Response*, *SearchAuctionResult*, *SettingsManager*.

```
public sealed class NewAuctionsTask : IBackgroundTask
{
    ...
    public async void Run(IBackgroundTaskInstance taskInstance)
    {
        BackgroundTaskDeferral deferral =
            taskInstance.GetDeferral();
        var filterLink = await GetFilter();
        ...
        Uri link =
            _requestGenerator.FilteredAuction(filterLink);
        var auctions = await
            AuctionDownloader.GetAuctionsByParameters(link) as
            List<Auction>;
        _foundAuction = GetNewestAuctions(auctions);
        ...
        CreateToastNotification(_foundAuction);
        deferral.Complete();
    }
    ...
}
```

Listing 3.7: Implementacja wykonywanego zadania w tle.

## Rozdział 4.

# Prezentacja interfejsu aplikacji

### 4.1. Etymologia nazwy

Nazwa aplikacji powstała już w trakcie tworzenia, kiedy wiedziałem już z którym serwisem aplikacja będzie współpracować. Szukałem czegoś co jednym słowem opisze sposób myślenia inwestora. „*Jak wyciągnąć największe zyski z inwestycji?*”. W połączeniu z nazwą serwisu, który obsługuje, nie mogło być lepszej nazwy jak **Kopra**<sup>1</sup>. Co miało podkreślić, wyciągnięcie esencji z Kokosa.

### 4.2. Korzyści względem wersji przeglądarkowej

Stronę serwisu trudno jest obsługiwać na telefonach Windows Phone ze względu na brak dobrej przeglądarki internetowej na tych urządzeniach. Sam serwis nie jest dostosowany do małych ekranów urządzeń mobilnych (brak poprawnie zaimplementowanej responsywności). **Kopra** została zaprojektowana wyłącznie na takie urządzenia i jest to największa jej zaleta w stosunku do strony internetowej.

### 4.3. Koszt użytkowania aplikacji

Stworzona przeze mnie aplikacja jest darmowa. Również wersja, która zostanie wystawiona do sklepu będzie udostępniona bezpłatnie, bez reklam. To wyróżnia moją aplikację względem konkurencyjnego oprogramowania Kokoid, które na każdym ekranie wyświetla użytkownikowi baner reklamowy. Kod źródłowy aplikacji jest dostępny w oparciu o liberalną licencję MIT.

### 4.4. Bezpieczeństwo danych przechowywanych w aplikacji

Aplikacja po zalogowaniu ma dostęp do wszystkich danych osobowych umieszczonych w serwisie. Do tych danych należą między innymi: numer i seria dowodu osobistego, adres użytkownika, adresy IP z datą logowania do serwisu, historia kredytowa, aktualne inwestycje, saldo użytkownika, hasło oraz email. Z całego ogromu informacji, w aplikacji trzymane są tylko dane logowania i klucz dostępu do WebAPI użytkownika. Aby poprawić bezpieczeństwo, cały ruch między aplikacją a serwisem Kokos jest szyfrowany protokołem SSL. Również aby zapobiec ewentualnym wyciekom danych, aplikacja nie wysyła danych logowania na serwery zewnętrzne.

---

<sup>1</sup>Wysuszony miąższ orzechów palmy kokosowej, surowiec do wyrobu oleju.

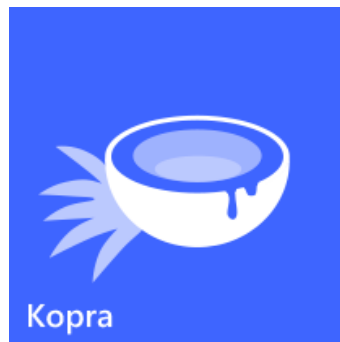
### 4.5. Jak wygląda Kopra?

#### 4.5.1. Logo

Logo aplikacji (rysunki 4.1, 4.2) ilustruje połowę owocu kokosowego z liśćmi drzewa kokosowego. Jak wspomniałem w pierwszych rozdziałach pracy, jest to nawiązanie nazwy do wykorzystywanego serwisu. Logo zostało wykonane w bieli, bez tła, dlatego jest przystosowane do każdego motywu kolorystycznego z nasycenymi barwami (różnymi od bieli).



Rysunek 4.1: Logo na szerokim kafelku



Rysunek 4.2: Logo na kwadratowym kafelku

#### 4.5.2. Splash screen

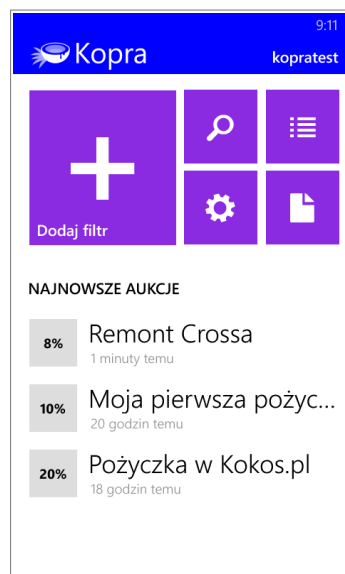
W trakcie uruchamiania aplikacji, użytkownik może zobaczyć duże logo aplikacji wraz z jej nazwą na niebieskim tle (rysunek 4.3).



Rysunek 4.3: Splash screen



Rysunek 4.4: Ekran logowania



Rysunek 4.5: Strona główna aplikacji

### 4.5.3. Strona logowania

Pierwszy ekran (rysunek 4.4), jaki widzi użytkownik, jest wykonany w motywie pośrednim. Pozwala on na łagodne przejście między domyślnym ciemnym motywem Windows Phone na motyw jasny, który będzie widoczny po zalogowaniu się do serwisu. Na tym ekranie użytkownik jest zmuszony do przekazania loginu (adresu email) oraz hasła bezpośrednio do strony Kokos.pl. Po wybraniu przycisku zaloguj, jest on zastąpiony przez napis „Logowanie...” oraz animację paska postępu. Dodatkowym elementem są komunikaty informujące użytkownika np. o braku połączenia z Internetem.

### 4.5.4. Strona główna

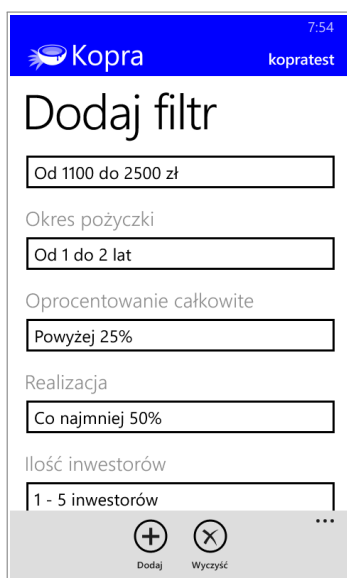
Ekran dostępny po zalogowaniu (rysunek 4.5) składa się z trzech części. Na górze ekranu po lewej stronie widnieje mała wersja logo oraz nazwy aplikacji. Po prawej stronie, znajduje się nazwa zalogowanego użytkownika. Drugą część stanowią przyciski, symulujące wyglądem kafelki znane z systemu Windows Phone 8. Ekran ten agreguje najważniejsze funkcje dostępne dla użytkownika.

### 4.5.5. Dodawanie filtru

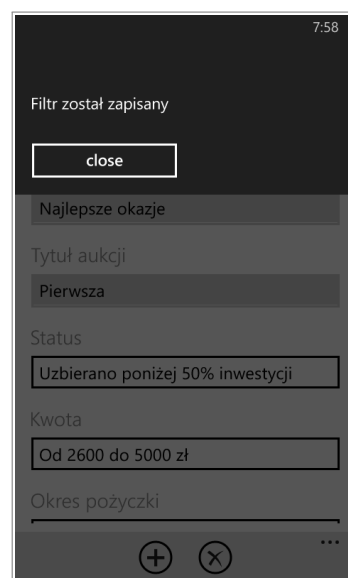
Ekran dodawania filtrów (rysunki 4.6, 4.7) znajduje się pod przyciskiem *Dodaj filtr* z ikoną znaku plus na stronie głównej aplikacji. Jest on przeznaczony do określania parametrów aukcji zgodnymi z wymaganiami użytkownika. Po wpisaniu nazwy filtru i określeniu kryteriów, filtr zostaje zapisywany w pamięci telefonu celem powtórnego użycia. Po wpisaniu kryteriów użytkownik może zapisać je na telefonie (rysunek 4.8). Filtry są świetnym sposobem na oszczędzenie czasu, ponieważ pozwalają zapamiętać preferencje użytkownika, są również niezbędne do działania powiadomień w tle.



Rysunek 4.6: Ekran dodawania filtrów



Rysunek 4.7: Ekran dodawania filtrów (cd.)



Rysunek 4.8: Potwierdzenie zapisania filtru

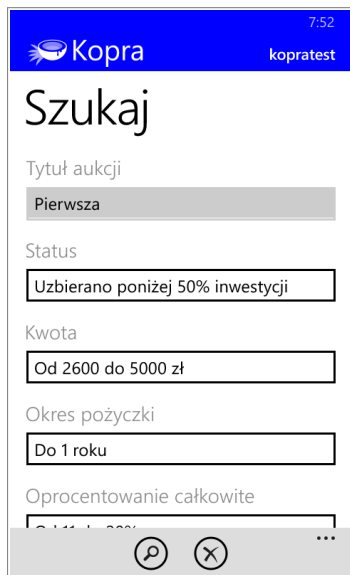
### 4.5.6. Wyszukiwanie aukcji

Ekran (rysunki 4.9, 4.10) oznaczony ikoną lupy pozwala na szybkie wyszukanie aukcji spełniających wymagania użytkownika. Po przytrzymaniu obszaru ekranu, wyświetli się menu kontekstowe pozwalające otworzyć aktualnie prezentowaną aukcję w przeglądarce internetowej.

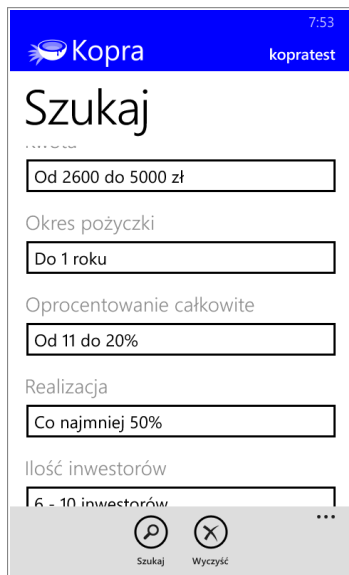
## 4.5. JAK WYGLĄDA KOPRA?

### 4.5.7. Wyniki wyszukiwania

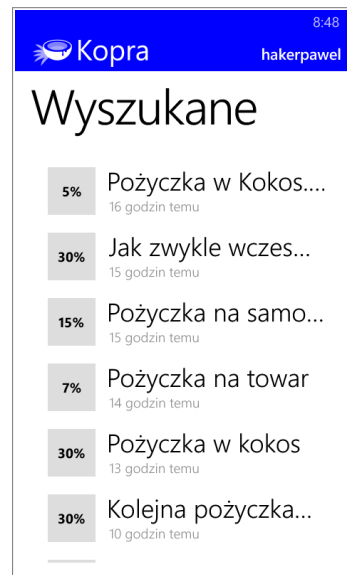
Ekran (rysunek 4.11) dostępny jest po wywołaniu funkcji szukaj (ikona lupy) lub wybraniu jednego filtru z listy. Wyświetla stronę ze znalezionymi aukcjami dla zadanych kryteriów wyszukiwania.



Rysunek 4.9: Ekran wyszukiwania aukcji



Rysunek 4.10: Ekran wyszukiwania aukcji (cd.)



Rysunek 4.11: Ekran z listą wyszukiwanych aukcji

### 4.5.8. Lista filtrów

Na tym ekranie (rysunek 4.12) dostępne są zdefiniowane przez użytkownika filtry. Pozwalają one na szybki dostęp do aukcji. Przytrzymując nazwę wybranego filtru na ekranie użytkownik otrzymuje możliwość usunięcia go listy.

### 4.5.9. Ustawienia

Ekran (rysunek 4.13) oznaczony na stronie głównej ikoną zębatego koła. Pozwala on na sprawdzenie poprawności pobranego klucza dostępu do serwisu, datę ważności klucza oraz zawiera funkcjonalność monitora aukcji.

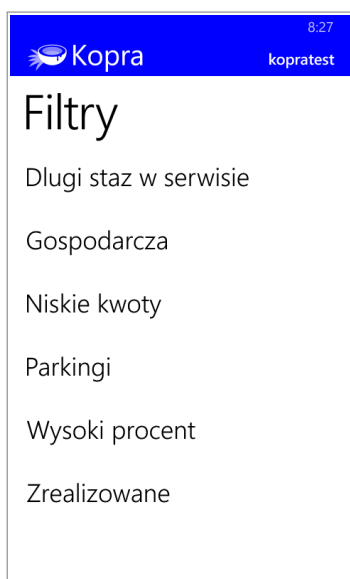
Monitor aukcji służy do przeszukiwania serwisu co 30 minut celem znalezienia nowych aukcji spełniających wybrane przez użytkownika kryteria. Kryteria określa się przez wybranie jednego ze zdefiniowanych filtrów z listy "Filtr do powiadomień" i wybranie przycisku "Monitor aukcji".

### 4.5.10. Raport inwestora

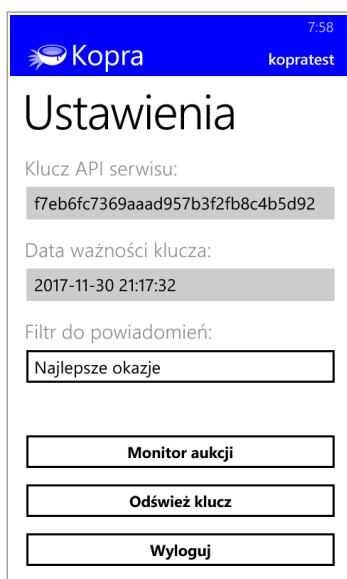
Na tym ekranie wyświetlane są statystyki spłacalności z poprzedniego miesiąca, prognoza przyszłych zysków, aktualne saldo oraz liczba pożyczek w trakcie spłaty (rysunek 4.14).

### 4.5.11. Szczegóły aukcji

Ekran zawierający wszystkie ważne informacje dotyczące pojedynczej aukcji. Jest to ekran typu *pivot*, co oznacza, że dane są podzielone sekcjami i należy przewinąć ekran w lewo lub w prawo by przejść do kolejnej sekcji (rysunki 4.15, 4.16, 4.17, 4.18, 4.19, 4.20).



Rysunek 4.12: Ekran z listą filtrów



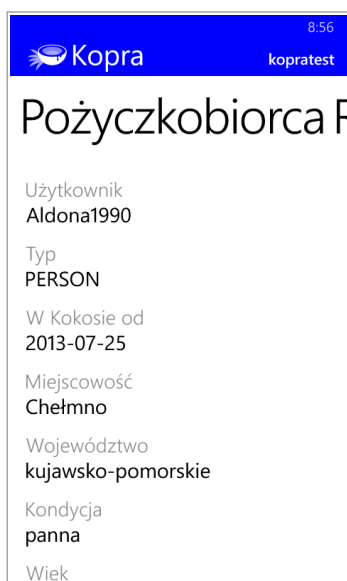
Rysunek 4.13: Ekran ustawień aplikacji



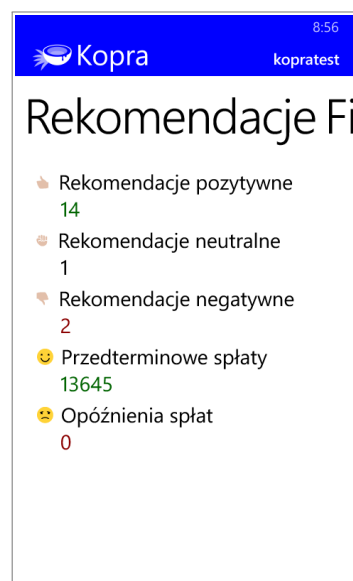
Rysunek 4.14: Ekran z raportem inwestora



Rysunek 4.15: Podsumowanie aukcji



Rysunek 4.16: Dane pożyczkobiorcy



Rysunek 4.17: Rekomendacje pożyczkobiorcy

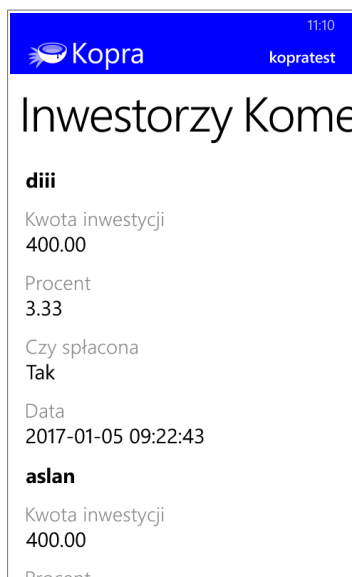
## 4.6. Instrukcja obsługi do aplikacji

Aby korzystać z aplikacji, należy mieć aktywne konto w serwisie kokos.pl oraz aktywować klucz API pierwszy raz przez przeglądarkę. Po uruchomieniu Kopry, należy podać adres email oraz hasło logowania do serwisu. Po poprawnej weryfikacji, użytkownik zostanie przeniesiony do ekranu głównego, skąd może się dostać do wszystkich najważniejszych funkcjonalności aplikacji. Użytkownik może przeskoczyć bezpośrednio do jednej z trzech najnowszych aukcji wyświetlanych na dole ekranu.

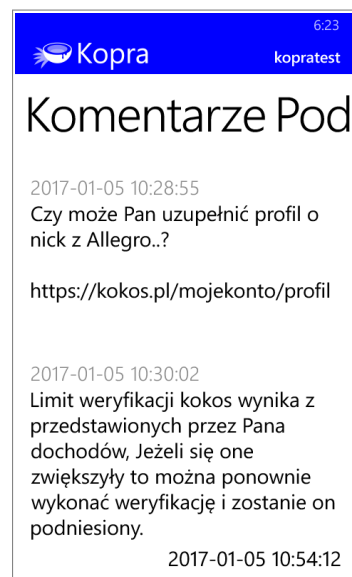
Aby uruchomić zadania w tle, należy mieć dodany przynajmniej jeden filtr w aplikacji, a na-



Rysunek 4.18: Finanse pożyczkobiorcy



Rysunek 4.19: Lista inwestorów



Rysunek 4.20: Komentarze do aukcji

stępnie z ekranu ustawień wybrać z listy, jakie aukcje mają być wyszukiwane w tle. Maksymalnie co 30 minut można otrzymać powiadomienie, z nową aukcją.



## Rozdział 5.

# Podsumowanie

Aplikacja ma w założeniu rozwiązać problem, jakim jest szybki dostęp do oferty pożyczkowej na urządzeniach mobilnych. Skupia się na obsłudze najpopularniejszego serwisu oferującego usługi typu „p2p lending”. **Kopra** ma wyciągnąć najważniejsze informacje potrzebne do dalszego procesu myślowego, czy warto zainteresować się ofertą pożyczkową.

Jasna kolorystyka aplikacji miała pomóc w zdobyciu zaufania użytkownika oraz nadać czystą strukturę interfejsowi. Aby lepiej poznać nawyki użytkowników systemu Windows Phone, zdecydowałem się na użytkowanie systemu przez rok, równoległe z procesem pisanie pracy. Jest to główny powód zmiany wyglądu interfejsu użytkownika po pierwszej iteracji, gdzie ekrany przypominały zmniejszoną wersję przeglądarkową serwisu. Ze względu na ogrom danych udostępnianych użytkownikowi w serwisie, zdecydowałem się na zaprezentowanie szczegółów aukcji przy pomocy ekranu typu *pivot*. Pozwala to na łatwe grupowanie powiązanych ze sobą tematycznie informacji.

Oprogramowanie stworzyłem z myślą o telefonach Windows Phone 8.1 wraz z kolejnymi wersjami, ponieważ jest to system stworzony z myślą o produktywności, szczególnie pod kątem ludzi biznesu.

Na chwilę obecną aplikacja ma możliwość sprawdzania w tle jednego filtru użytkownika. Funkcjonalność można rozszerzyć o wybór ilości powiadomień na godzinę, które użytkownik jest w stanie zaakceptować, oraz dodać listę filtrów, które miałyby być sprawdzane i ustawić dla nich priorytety. W kolejnych wersjach przewiduję również możliwość tworzenia kopii zapasowej filtrów na dysku OneDrive oraz dzielenie się nimi z innymi inwestorami. Kolejnym kamieniem milowym w rozwoju aplikacji może być automat inwestycyjny, który korzystając z funkcjonalności wyszukiwania aukcji w tle, inwestowałby automatycznie określoną kwotę w znalezione oferty inwestycyjne.

Stworzona przeze mnie aplikacja jest pierwsza w swojej kategorii. Liczę na odzew programistów po publikacji Kopry w Sklepie Microsoft. Zależy mi, by inwestowanie przez Koprę było łatwą formą pomnażania oszczędności bez konieczności płacenia wysokich prowizji pośrednikom, co pomoże poprawić domowy budżet wielu osób.



# Spis rysunków

2.1. Zestaw konfiguracji emulatorów dostępny w Windows Phone 8.1 SDK . . . . .	18
2.2. Zestaw konfiguracji emulatorów dostępny w Windows 10 Mobile SDK . . . . .	19
3.1. Struktura folderów w projekcie aplikacji Kopra . . . . .	22
3.2. Lista plików widoków w projekcie . . . . .	23
3.3. Lista plików widok-modeli w projekcie . . . . .	23
4.1. Logo na szerokim kafelku . . . . .	28
4.2. Logo na kwadratowym kafelku . . . . .	28
4.3. Splash screen . . . . .	28
4.4. Ekran logowania . . . . .	28
4.5. Strona główna aplikacji . . . . .	28
4.6. Ekran dodawania filtrów . . . . .	29
4.7. Ekran dodawania filtrów (cd.) . . . . .	29
4.8. Potwierdzenie zapisania filtra . . . . .	29
4.9. Ekran wyszukiwania aukcji . . . . .	30
4.10. Ekran wyszukiwania aukcji (cd.) . . . . .	30
4.11. Ekran z listą wyszukanych aukcji . . . . .	30
4.12. Ekran z listą filtrów . . . . .	31
4.13. Ekran ustawień aplikacji . . . . .	31
4.14. Ekran z raportem inwestora . . . . .	31
4.15. Podsumowanie aukcji . . . . .	31
4.16. Dane pożyczkobiorcy . . . . .	31
4.17. Rekomendacje pożyczkobiorcy . . . . .	31
4.18. Finanse pożyczkobiorcy . . . . .	32
4.19. Lista inwestorów . . . . .	32
4.20. Komentarze do aukcji . . . . .	32



# Bibliografia

- [1] CHACON S., STRAUB B.: *Pro Git*, Apress, 2014, Wydanie II, ISBN 978-14-842-0077-3.
- [2] DINO E.: *Programming Microsoft ASP.NET 4*, Microsoft Press, 2011, ISBN 978-0-7356-4338-3.
- [3] JAJUGA K., JAJUGA T.: *Inwestycje. Instrumenty finansowe, ryzyko finansowe, inżynieria finansowa*, Warszawa, Wydawnictwo Naukowe PWN SA, 2015, Wydanie III zm., ISBN 978-83-01-14957-4.
- [4] DĘBSKI W.: *Rynek finansowy i jego mechanizmy. Podstawy teorii i praktyki*, Warszawa, Wydawnictwo Naukowe PWN SA, 2014, Wydanie VI zm., ISBN 978-83-01-17996-0.
- [5] DRABIK L., SOBOL E.: *Słownik języka polskiego PWN*, Wydawnictwo Naukowe PWN, 2007, ISBN 978-8-3011-7377-7.
- [6] *Słownik serwisu NBP – Portal Edukacji Ekonomicznej*, <https://www.nbportal.pl/slownik/pozycje-slownika/toksyczne-aktywa>, dostęp: 10.12.2016.
- [7] *Billionaire who broke the Bank of England*, <http://www.telegraph.co.uk/finance/2773265/Billionaire-who-broke-the-Bank-of-England.html>, dostęp: 10.12.2016.
- [8] *Design Universal Windows Platform (UWP) app*, <https://dev.windows.com/en-us/design>, dostęp: 10.12.2016.
- [9] *DoMore Archives*, <http://blogs.microsoft.com/firehose/tag/DoMore/>, dostęp: 10.12.2016.
- [10] *Financing Civilization*, <http://vikingsom.yale.edu/will/finciv/chapter1.htm>, dostęp: 10.12.2016.
- [11] *Fractional Reserve Banking – An Economist’s Perspective (Transcript)*, <https://www.frbatlanta.org/education/classroom-economist/fractional-reserve-banking/economists-perspective-transcript>, dostęp: 11.12.2016.
- [12] *Taking a Peek at Peer-to-Peer Lending*, <http://business.time.com/2012/11/15/taking-a-peek-at-peer-to-peer-lending/>, dostęp: 10.12.2016.
- [13] *Koniec lokat antybelkowych. Co w zamian*, <http://www.money.pl/pieniadze/wiadomosci/artykul/koniec;lokat;antybelkowych;co;w;zamian,82,0,1057874.html>, dostęp: 08.12.2016.
- [14] *Wyrok NSA, sygn. II FSK 1472/10*, <http://orzeczenia.nsa.gov.pl/doc/1F82F6552C>, dostęp: 28.11.2016.
- [15] *Wyrok NSA, sygn. II FSK 142/10*, <http://orzeczenia.nsa.gov.pl/doc/862AF4722F>, dostęp: 28.11.2016.

- [16] *Oświadczenie Zespołu Kokos.pl w sprawie wyroku NSA o sygn. II FSK 1472/10*, [https://kokos.pl/aktualnosci/czytaj?id=2012\\_03\\_06](https://kokos.pl/aktualnosci/czytaj?id=2012_03_06), dostęp: 22.12.2016.
- [17] *Zostań Inwestorem – Kokos.pl*, <https://kokos.pl/info/chce-inwestowac>, dostęp: 28.10.2016.
- [18] *Podstawowe stopy procentowe NBP*, <http://www.nbp.pl/home.aspx?f=/dzienne/stopy.htm>, dostęp: 22.12.2016.
- [19] *Pożyczki społecznościowe finansowo.pl*, <https://www.finansowo.pl>, dostęp: 22.12.2016.
- [20] *Pożyczki społecznościowe zakra.pl*, <https://zakra.pl>, dostęp: 22.12.2016.
- [21] *Pożyczki społecznościowe kokos.pl*, <https://kokos.pl>, dostęp: 22.12.2016.
- [22] *Pożyczki społecznościowe lendico.pl*, <https://www.lendico.pl>, dostęp: 22.12.2016.
- [23] *Pożyczki społecznościowe zakramini.pl*, <https://zakramini.pl>, dostęp: 22.12.2016.
- [24] *Pożyczki społecznościowe applecredit.pl*, <https://applecredit.pl>, dostęp: 22.12.2016.
- [25] *Pożyczki społecznościowe capitalclub.pl*, <https://capitalclub.pl>, dostęp: 22.12.2016.
- [26] *Funkcje webAPI serwisu Zakra.pl*, <https://zakra.pl/api>, dostęp: 22.12.2016.
- [27] *Dokumentacja WebAPI serwisu Kokos.pl*, <https://kokos.pl/webapiinfo/dokumentacja>, dostęp: 08.11.2015.
- [28] *Microsoft Visual Studio Community 2015*, <https://www.microsoft.com/en-us/download/details.aspx?id=48146>, dostęp: 22.12.2016.
- [29] *Dokumentacja języka C# dla Visual Studio 2015*, <https://msdn.microsoft.com/en-us/library/kx37x362.aspx>, dostęp: 22.12.2016.
- [30] *Komponenty architektury zestawu narzędzi .NET*, <https://docs.microsoft.com/en-us/dotnet/articles/standard/components>, dostęp: 22.12.2016.
- [31] *Archiwum Windows SDK oraz emulatorów*, <https://developer.microsoft.com/en-us/windows/downloads/sdk-archive>, dostęp: 22.12.2016.
- [32] *Treść licencji MIT wykorzystywanej przez Newtonsoft.Json*, <https://github.com/JamesNK/Newtonsoft.Json/blob/master/LICENSE.md>, dostęp: 22.12.2016.
- [33] *Strona domowa projektu JsonPath*, <http://goessner.net/articles/JsonPath/>, dostęp: 22.12.2016.
- [34] *Plany abonamentowe serwisu GitHub.com*, <https://github.com/pricing>, dostęp: 22.12.2016.
- [35] *LINQPad as a Code Scratchpad*, <http://www.linqpad.net/CodeSnippetIDE.aspx>, dostęp: 22.12.2016.
- [36] *Opis LINQPad na Wikipedii*, <https://en.wikipedia.org/wiki/LINQPad>, dostęp: 22.12.2016.

- [37] *Repozytorium projektu ILSpy*, <https://github.com/icsharpcode/ILSpy>,  
dostęp: 22.12.2016.
- [38] *Oryginalna notka techniczna nt. wzorca MVC stworzona dla XEROX PARC w latach 1978-79*,  
<http://heim.ifi.uio.no/~trygver/themes/mvc/mvc-index.html>,  
dostęp: 22.2016.