Wydział Elektroniki i Technik Informacyjnych

Politechnika Warszawska

Instytut Automatyki i Informatyki Stosowanej


Praca dyplomowa inżynierska

# Komunikator preferencji

Jacob Jozef Tarasiewicz

Opiekun pracy

Dr inż. Mariusz Kamola

Warszawa, 2014

**Streszczenie:**

**Komunikator preferencji**

Celem pracy było wykonanie prototypu aplikacji mobilnej. Aplikacja korzysta z sieci WiFi w celu rozsyłania i odbierania haseł. Hasła te są ofertami przygotowanymi przez użytkowników. W każdej z nich zawarte są informacje dotyczące poszczególnego ogłoszenia. Wyróżniono dwa typy ofert: kupna i sprzedaży. Aplikacja odnajduje oferty w zasięgu sieci bezprzewodowej oraz następnie wyświetla oferty sprzedaży z innych urządzeń mobilnych odpowiadających ofertom kupna użytkownika programu.

Aplikację zaprojektowano na system Android. Z powodu ograniczeń systemowych do komunikacji została zastosowana technologia WiFi Direct. W celu przekazywania ofert, modyfikowana jest odpowiednio nazwa urządzenia. Użyto prostego kodowania, aby zamieścić w nazwie identyfikator aplikacji oraz najważniejsze parametry oferty. Po odnalezieniu odpowiedniej oferty użytkownik może nawiązać połączenie z innym urządzeniem w celu przesłania pełnej oferty.

W pracy pokazano kolejne etapy powstawania oraz pewne ciekawe szczegóły implementacji aplikacji.

**Abstract:**

**Communicator of preferences**

The aim of thesis was to execute a mobile application prototype. Application uses WiFi network in order to broadcast and receive selected slogans. Those slogans are offers prepared by users. Every each of them contains information about a particular announcement. There are two types of offers: buying and selling. The application finds offers in reach of wireless network and displays offers from other mobile devices corresponding to buy offers of the program user.

Application was designed for the Android system. Due to system restrictions to communicate WiFi Direct technology has been used. In order to transfer offers, the device name is appropriately modified. Simple encoding was used to post in the name of device the application ID and the most important parameters of the offer. After finding a suitable offer the user can establish a connection with another device in order to submit a full tender.

In the thesis is shown the successive stages of formation and some interesting implementation details of applications.

# Spis treści

# 1 Cel i motywacja pracy

Celem pracy było stworzenie prototypu aplikacji na urządzenia mobilne, wykorzystującej standardy sieci bezprzewodowych do odnajdywania osób o pasujących do siebie preferencjach, podanych przez użytkownika programu. Preferencje zostały odzwierciedlone jako oferty, co pozwala na ich sprawne i jednoznaczne określenie. Aplikacja umożliwia użytkownikowi wprowadzanie, modyfikowanie oraz usuwanie ofert typu kupna lub sprzedaży. Każda oferta posiada pewne ustalone parametry, takie jak kategoria, cena lub opis. Do każdej kategorii przypisane zostały również atrybuty charakterystyczne dla danej kategorii. Aplikacja wyświetla oferty sprzedaży z głównymi parametrami z innych urządzeń spełniające wymagania podane przez użytkownika. W celu pokazania pełnej oferty wraz z jej wszystkimi parametrami, aplikacja umożliwia nawiązanie połączenia pomiędzy urządzeniami a następnie pobranie pełnej oferty sprzedaży i jej wyświetlenie. Program umożliwia również zatrzymanie i wznowienie wyszukiwania urządzeń będących w zasięgu sieci bezprzewodowej. Dzięki tej aplikacji osoba korzystająca z tego programu może się dowiedzieć, że w najbliższej okolicy jest ktoś z odpowiadającymi jej preferencjami.

Zrealizowanie parowania odpowiednich użytkowników jest możliwe poprzez podanie przez użytkownika w programie odpowiednich ofert, określających zainteresowania, preferencje czy potrzeby użytkownika. Wyróżnione zostały dwa możliwe rodzaje ofert. Oferta sprzedaży informuje o tym, że dany użytkownik ma coś do zaoferowania innym użytkownikom. Staje się wtedy użytkownikiem rozgłaszającym. Drugim rodzajem oferty jest oferta kupna informująca o tym, że użytkownik czegoś poszukuje. Przez to zostaje użytkownikiem odbierającym. Rozróżnienie na te dwa rodzaje ofert pozwala dopasować odpowiednie oferty do siebie. Aplikacja na podstawie ofert kupna użytkownika może odnaleźć odpowiadające im oferty przeciwnego rodzaju. Odnajdowanie odpowiednich ofert odbywa się przez porównanie wybranych parametrów ofert. W ten sposób mogą zostać sparowani odpowiedni

użytkownicy i przesłane pełne oferty oraz umożliwione jest dokonanie transakcji przez dopasowane osoby.

W dokonaniu komunikacji pomiędzy urządzeniami wykorzystana została technologia Wi-Fi. Umożliwia ona odnajdowanie i komunikowanie urządzeń znajdujących się w pobliżu. Jednym z celów korzystania z technologii Wi-Fi było zapewnienie bezpośredniej komunikacji bez udziału jakichkolwiek zewnętrznych serwerów. Pamiętając przy tym i starając się maksymalnie możliwie ograniczyć zużycie energii. Należy pamiętać, że urządzenia mobilne działają na zasilaniu z baterii. Jest to energia, która potrafi się szybko skończyć. Podane przez użytkownika hasła są rozsyłane i odbierane stosując standardy sieci Wi-Fi. Dzięki takiemu sposobowi komunikacji korzystanie z aplikacji nie wymaga połączenia z Internetem, co owocuje całkowicie darmowym korzystaniem z programu. A także umożliwia jego działanie bez ograniczeń na miejsce przebywania, które mogą być pozbawione zasięgu sieci GSM. Takie rozwiązanie zapewnia sprawne działanie aplikacji w każdych warunkach.

W dzisiejszych czasach gdzie dane wpisywane w Internecie są zbieranie oraz przetwarzane przez wiele firm. Poczynając od dobrowolnego podawania danych na różnych portalach społecznościowych, do których użytkownik akceptując regulamin często zrzeka się praw do informacji oraz materiałów zamieszczanych przez siebie. Przez wyszukiwarki, które analizując poprzednio wyszukiwane hasła są w stanie przewidzieć, jakie reklamy i strony internetowe mogą być dla nas „najlepsze”. Kończąc na może trochę przesadzonej, nowej obawie kontroli rządów państw nad Internetem. Wydaje się, że bezpośrednia komunikacja poprzez Wi-Fi może zapewnić prywatność oraz anonimowość w sieci. Dzieje się tak, dlatego, że całkowicie znika potrzeba przechowywania danych użytkownika. Wystarczającym narzędziem do sprawnego działania zapewnia skanowanie sieci bezprzewodowych, co pozwala odnaleźć odpowiednie oferty od anonimowych użytkowników. Oferty użytkowników są

tylko dostępne, kiedy przebywają w zasięgu działania aplikacji. Gdy przestaną się w nim znajdować oferta zostaje skasowana.

Zapewnienie anonimowości użytkownikom i ich ofertom było główną motywacją powstania tej pracy. Nie są potrzebne żadne dane jak login czy e-mail użytkownika. Jedyne rozróżnianie użytkowników jest realizowane przez identyfikacje ofert, gdzie nie liczy się, kto daną ofertę przygotował, lecz sama jej treść. Na tej podstawie można stwierdzić, że użytkownik pozostaje anonimowy przez cały proces przetwarzania i dopasowywania ofert. Identyfikacja użytkownika jest potrzebna dopiero po dopasowaniu ofert w celu dokonania transakcji bądź rzeczywistego spotkania się osób w tym celu.

Reasumując, celem pracy było stworzenie aplikacji mobilnej umożliwiającej odnalezienie użytkowników w bliskiej odległości o odpowiadających sobie preferencjom, korzystającej ze standardów sieci bezprzewodowych. Zapewniając przy tym działanie w każdych warunkach. A także gwarantując zachowanie anonimowości przez korzystających z niej użytkowników poprzez obejście potrzeby przechowywania jakichkolwiek danych w scentralizowanej bazie danych, bądź na urządzeniach użytkowników.

# 2 Zastosowanie oraz istniejące rozwiązania

Stworzona aplikacja służy do odnajdywania osób, które mogą być zainteresowane podjęciem odpowiedniej, zależnej od oferty interakcji. Program może być użyty w wielu sektorach i branżach. Np.:

- Lokalizacja stoisk na giełdach, targach – często na giełdzie jest wiele stoisk, które wyglądają niemal identycznie. Czasami nawet nie mają sloganu z nazwą. W takim przypadku osoba obsługująca stoisko może uruchomić aplikację i dodać ofertę sprzedaży z nazwą stoiska. Wtedy osoba niemogąca odnaleźć danego punktu może również w swoim urządzeniu dodać ofertę kupna z nazwą stoiska. Kiedy osoba będzie w zasięgu sieci Wi-Fi dostanie powiadomienie o znalezieniu oferty i będzie mogła się rozejrzeć i już z łatwością odnaleźć szukany stragan.
- Lokalizacja odpowiednich sklepów w centrach handlowych – podobnie jak w poprzednim przypadku, z tym, że użytkownik szuka pewnego towaru i nie wie, który sklep może mu go zaoferować. Korzystając z aplikacji przechodząc pod odpowiednim sklepem, w którym jest aktywna aplikacja z dostępnym towarem, dostanie powiadomienie o tym, że w tym sklepie znajdzie interesujący towar.
- Pośrednictwo nieruchomości – użytkownik przyjeżdża do nowego miasta i chce znaleźć mieszkanie. Osoba wynajmująca mieszkanie może przygotować ofertę rodzaju sprzedaży mieszkania. Człowiek szukający ma odpowiednią ofertę na swoim urządzeniu. Przechodząc w pobliżu może dostać powiadomienie, że w tym miejscu jest mieszkanie do wynajęcia.
- Handel na ulicy – użytkownik ma do sprzedania książkę, a drugi użytkownik chce kupić książkę. Kiedy takie dwie osoby będą korzystały z aplikacji to przechodząc obok siebie na ulicy lub korzystając

z środków komunikacji miejskiej dostaną powiadomienie o znalezieniu oferty i mogą dobić targu.

Możliwości zastosowania tej aplikacji jest naprawdę bardzo wiele. To, że działa nawet bez zasięgu sieci, bez dostępu do Internetu i nie posiada scentralizowanej bazy danych może stworzyć jeszcze wiele innych możliwych zastosowań.

Do istniejących rozwiązań można porównać jedynie w pewnym stopniu:

- Serwisy aukcyjne:
  - Allegro
  - Gumtree
  - eBay (w skali globalnej)

  Serwisy te umożliwiają dodanie ogłoszenia i potem wyszukanie go przez innego użytkownika. Można wyszukać ogłoszenie z interesującego nas obszaru, ale nie ma możliwości powiadomienia w czasie rzeczywistym użytkownika o pojawiającym się ogłoszeniu w jego pobliżu. Również sprzedający nie ma sposobu odnalezienia zainteresowanych jego ofertą osób.

- Serwisy społecznościowe:
  - Facebook
  - Twitter

  Serwisy te umożliwiają dzielenie się informacjami pomiędzy posiadanymi na tych serwisach znajomymi. Często można się spotkać z różnymi typu ogłoszeniami lub ofertami napisanymi przez swoich znajomych. W tym jednak przypadku nie ma możliwości wyświetlenia tylko konkretnych ofert. Jest się również ograniczonym tylko do ogłoszeń opublikowanych przez swoich znajomych. Brak jest również zachowania anonimowości.

Nie było jeszcze takiej aplikacji, która mogłaby zaproponować automatyczne odnajdowanie określonych urządzeń mobilnych w konkretnym celu. Dlatego warto było stworzyć program, który może to zaoferować. Jak można również zaobserwować nie ma obecnie rozwiązania, które pozwalałoby zachować anonimowość przed odnalezieniem ofert. Od samego początku zawsze znana jest tożsamość osoby ogłaszającej ofertę. Z tych właśnie względów aplikacja może być pomocna ludziom w wielu przypadkach.

# 3 Wykorzystywana technologia

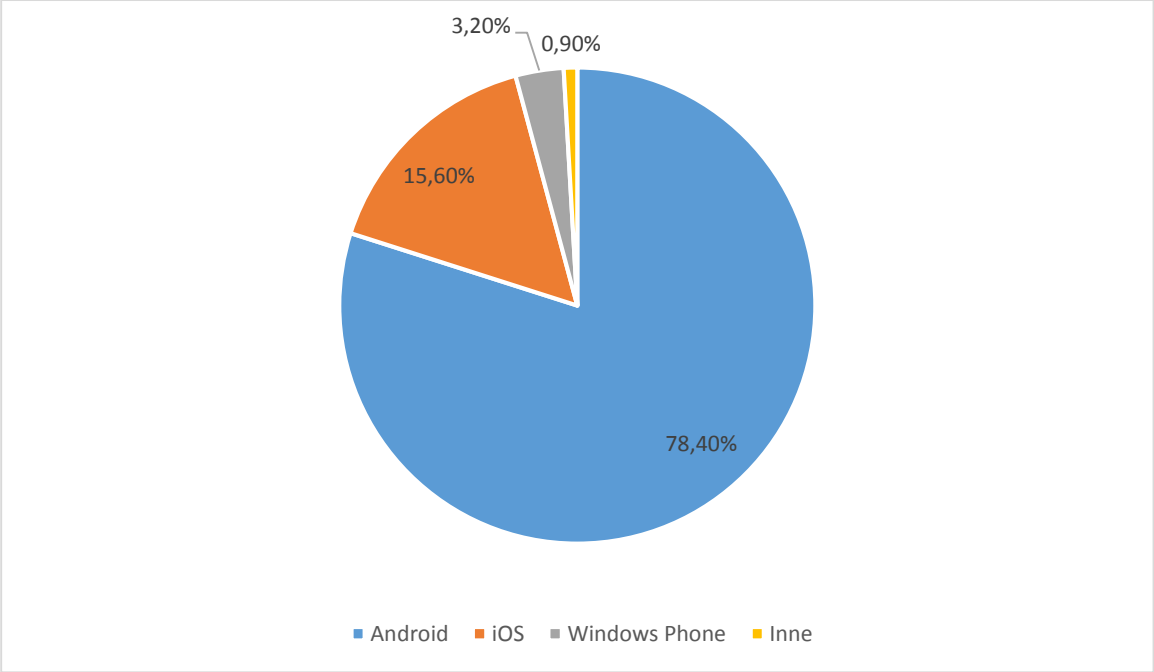
## 3.1 System operacyjny - Android

Każda aplikacja mobilna do działania musi być zainstalowana w systemie operacyjnym. Aby instalacja się powiodła potrzebne jest stworzenie odpowiedniej aplikacji, która wpiera daną platformę. Obecnie każdy system mobilny ma unikalną architekturę aplikacji jak i jej konfigurację, co sprawia, że dana aplikacja może działać tylko na jednej platformie. Żeby umożliwić jej działanie w innym systemie potrzeba jej kod źródłowy zmodyfikować lub przepisać w innym języku programowania tak, aby wspierał inny system i stworzyć odpowiednie pliki konfiguracyjne. Postanowiono, że aplikacja powstająca w jej wyniku powinna wspierać jedną wybraną platformę.

Obecnie na rynku są dostępne trzy liczące się systemy operacyjne dla urządzeń mobilnych. Są to:

- Windows Phone – tworzony przez Microsoft
- iOS – tworzony przez Apple
- Android – tworzony przez Google

*Rysunek 1 Procentowy podział rynku sprzedaży w roku 2013*




*Źródło: Gartner, www.Gartner.Com/newsroom/id/2665715, 18.07.2014*

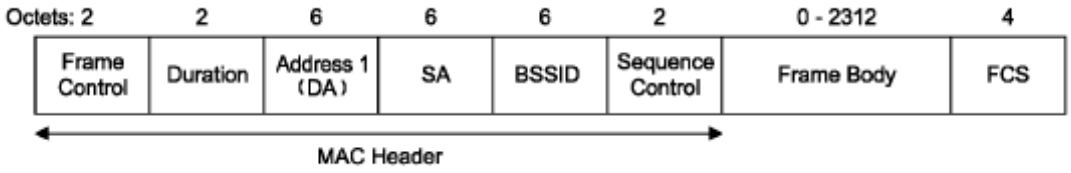
Potrzeba było wybrać system, na który zostanie zaimplementowana aplikacja. Jak widać na rysunku 1 największy udział w sprzedaży miały urządzenia mobilne działające z systemem Android. Ze względu na znikomy udział w rynku platformy Microsoftu, została ona od razu odrzucona. Pozostały systemy Android oraz iOS. Ten pierwszy ma dostępny szereg darmowych i oficjalnych narzędzi służących do tworzenia aplikacji oraz bardzo dobre wsparcie innych programistów w grupach dyskusyjnych oraz różnych forach internetowych. Z kolei Apple również posiada odpowiednie narzędzia, aczkolwiek, aby z nich korzystać należy posiadać komputer Apple oraz wykupić całoroczny abonament. Z tych powodów wybrany do stworzenia aplikacji został system Android.

## 3.2 Sieć bezprzewodowa Wi-Fi

Wi-Fi jest potocznym określeniem zestawu standardów stworzonych do budowy sieci bezprzewodowych. Technologia ta jest szeroko dostępna

w każdym smartfonie lub tablecie czy innym urządzeniu mobilnym. Do działania wykorzystuje fale radiowe propagowane przez wbudowany w urządzenie moduł Wi-Fi. Zakres jego działania wynosi do około 50 metrów w otwartej przestrzeni. Zasięg ten jest odpowiednio mniejszy, kiedy działa wewnątrz budynków, gdzie na drodze rozchodzenia się sygnału są różne przeszkody jak ściany.

*Rysunek 2 Ramka konfiguracyjna*




*Źródło: http://www.mathworks.com/help/comm/examples/ieee-802-11-wlan-beacon-frame.html, 16.07.2014*

Szukanie sieci bezprzewodowej, do której urządzenie chce się połączyć przebiega w następujący sposób. Każdy nadajnik rozsyła, co krótki, ustalony interwał czasowy ramki konfiguracyjne. Ramka konfiguracyjna jak przedstawiono na rysunku 2 zawiera w sobie nagłówek MAC (MAC Header), sumę kontrolną (FCS) oraz podstawowe dane o sieci zawarte w Frame Body wymagane, aby nawiązać połączenie np. SSID – identyfikator sieci. Zawiera ona również MAC adres urządzenia, z którego została nadana. Kiedy urządzenie odbierze ramkę z żądanym identyfikatorem odsyła żądanie połączenia. Następuje nawiązanie połączenia z punktem dostępowym i dane mogą być przesyłane.

# 4 Realizacja pracy i rozwiązywanie problemów

## 4.1 Przygotowanie do realizacji

Realizacja pracy inżynierskiej zaczęła się od sformułowania szczegółowych zadań na podstawie ogólnie przyjętych założeń. Głównym założeniem było powstanie aplikacji na system Android umożliwiającą bezpośrednią komunikację pomiędzy urządzeniami oraz znajdowanie odpowiednich ofert dla użytkownika. Na ich podstawie w porozumieniu z opiekunem pracy powstały następujące szczegółowe zadania:

1. Rozsyłanie i odbieranie komunikatów.
2. Sposób opisu treści rozgłaszanej – ofert.
3. Sposób rozgłaszania oferty.
4. Testowanie aplikacji.

Wykonanie powyższych zadań było możliwe po dokładnym dokonaniu analizy środowiska, w którym miało powstać oprogramowanie. Dopiero po zapoznaniu się z samym system operacyjnym można było zacząć rozwiązywanie zadań implementacyjnych aplikacji. Analiza systemu oraz znalezienie sposobów realizacji zadań nastąpiły w zeszłym semestrze podczas Pracowni Dyplomowej I.

## 4.2 Analiza środowiska

Decydując się na system operacyjny Android, dokonałem również wyboru technologii realizacji implementacji. Aplikacje tworzone na Android są pisane w języku Java. Można również dodawać moduły rodzime pisane w C lub C++, które mogą zwiększyć wydajność całej aplikacji. Java jest językiem programowania obiektowego, gdzie każdy element jest obiektem. Działanie polega na komunikowaniu się obiektów między sobą w celu wykonania zadań. Java jest kompilowana do kodu bajtowego, który jest uruchamiany przez wirtualną maszynę Javy – JVM. Dzięki temu ten sam kod źródłowy zadziała na wielu platformach, które posiadają JVM. Jednakże Android nie został

wyposażony w JVM, zamiast tego posiada technologię Dalvik. W porównaniu z JVM jest to nowsza oraz ulepszona wirtualna maszyna przeznaczona na system Android. Została stworzona z myślą o urządzeniach mobilnych, gdzie są małe zasoby pamięci, energii oraz wolne procesory w porównaniu do stacjonarnych komputerów.

Do zalet Javy można zaliczyć Garbage Collector, który zajmuję się zarządzaniem pamięcią. Samemu usuwając stworzone obiekty, do których nie ma już odwołań. Dzięki temu programista może się bardziej skupić na logice programu i nie martwić się aż tak bardzo wyciekami pamięci, które nadal może spowodować słabej jakości kod źródłowy. Kolejną zaletą jest ogromna ilość dostępnych bibliotek, które dzięki temu, że Java jest kompilowana na maszynę wirtualną, są wieloplatformowe.

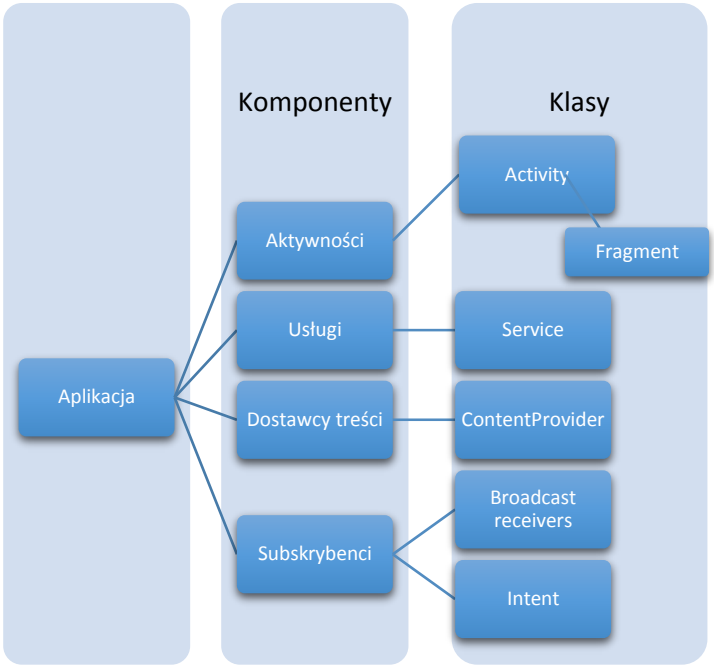
Android posiada specjalny serwis dla programistów na ich system. Zawiera on wiele przewodników, przykładów i dokumentację systemu. Chcąc stworzyć aplikację mobilną potrzebowałem się zaznajomić z tymi wszystkimi elementami i zaimplementować parę przykładowych, nieskomplikowanych programów. Android zapewnia również twórcom narzędzia deweloperskie. Istnieje specjalna wtyczka do zintegrowane środowiska programistycznego Eclipse o nazwie Android Developer Tools (ADT). Eclipse jest najpopularniejszym IDE do tworzenia programów w języku Java z tego względu posiada szerokie grono wsparcia technicznego w razie wypadków z konfiguracją otoczenia programistycznego. Plugin ADT można pobrać albo od razu z IDE Eclipse albo oddzielnie do już posiadanego IDE Eclipse. Trzeba jeszcze pobrać Android SDK (software development kit), który zawiera wszelkie biblioteki i narzędzia potrzebne do tworzenia aplikacji. Dzięki tak dobrze zorganizowanemu wsparciu od producenta platformy większość deweloperów korzysta z tych samych narzędzi i w razie jakichkolwiek problemów na forach dyskusyjnych można łatwo znaleźć rozwiązanie swojego problemu. Z powodu tego, że był to mój pierwszy projekt na urządzenia mobilne, często sam również korzystałem z tego

rodzaju wsparcia, zarówno z programistycznych for dyskusyjnych, jak i przewodników dostarczanych przez producenta. Dzięki temu można zaoszczędzić wiele godzin pracy.

## 4.3 Architektura aplikacji

Aplikacje na system Android składają się z komponentów. Każdy komponent jest miejscem, przez który system może porozumieć się z aplikacją.

*Rysunek 3 Komponenty aplikacji Android*




*Źródło: opracowanie własne na podstawie: Android, http://developer.android.com/guide/components/fundamentals.html#Components, 18.07.2014*

Na rysunku 3 pokazano komponenty, z których składa się aplikacja. Każdy z wymienionych komponentów spełnia inną rolę i ma inny cykl życia.

- Aktywności (Activities) – jest to podstawowa część każdej aplikacji posiadającej graficzny interfejs użytkownika GUI. Klasa ta reprezentuje każdy ekran wyświetlany użytkownikowi. Jedna aktywność może uruchomić następną np. przy przechodzeniu z jednego ekranu do drugiego. Aktywność może być złożona z fragmentów. Każdy fragment może być niezależny od innego fragmentu pomimo wyświetlania na tym samym ekranie. Każda aktywność dziedziczy po klasie Activity.

- Usługi (Services) – jest to komponent działający w tle aplikacji. Nie posiada GUI. Zazwyczaj służy do wykonywania skomplikowanych lub długotrwających działań aplikacji. Klasa Service jest dziedziczona przez każdą usługę.
- Dostawcy treści (Content providers) – służy do przechowywania danych oraz zarządzania nimi. Pozwala również na dostęp do danych przechowywanych przez inne aplikacje i system.
- Subskrybenci (Broadcast receivers) – służy do odbierania komunikatów wysyłanych przez system. Umożliwia on reagowanie aplikacji na zewnętrzne zdarzenia.

Aktywności, usługi oraz subskrybenci są uruchamiani poprzez przesłanie zamierzenia (Intent). Każde zamierzenie jest łączone z wybranym komponentem. Zamierzenie zawiera najczęściej działanie, jakie dany komponent ma podjąć. Może też zawierać dane przekazane z innego elementu. Dostawca treści jest wywoływany przy przekazywaniu danych z lub do aplikacji.

Każda aplikacja musi posiadać plik konfiguracyjny tak zwany manifest. Jest on zapisany w pliku *AndroidManifest.xml.* Każdy komponent aplikacji musi być w nim opisany. W przeciwnym wypadku nie zostanie on uruchomiony. Poza komponentami w pliku tym są jeszcze umieszczone m.in:

- Pozwolenia do zasobów systemu, które aplikacja powinna posiadać.
- Minimalna wersja API Androida, z którą aplikacja jest kompatybilna.
- Wymagania sprzętowe lub programowe, które wymaga aplikacja.

## 4.4 Ograniczenia systemowe

Dokonując analizy systemu i reguł działania aplikacji pod tą platformą doszedłem do wniosku, że Android posiada bardzo restrykcyjną politykę bezpieczeństwa. Każda aplikacja działa w tzw. piaskownicy (Sandbox). Oznacza

to całkowitą izolację aplikacji od innych programów jak i braku bezpośredniego dostępu do systemu i świata zewnętrznego. Potrzeba interakcji z systemem musi być uprzednio potwierdzona przez użytkownika, ale tutaj również nie ma pełnej swobody działań. Android umożliwia interakcję z system poprzez specjalnie przygotowane interfejsy programowania aplikacji (API). Można posiadać jedynie dostęp do tego, na co pozwalają przygotowane przez twórców Androida interfejsy. Chociaż jest ich naprawdę wiele, to do niektórych elementów systemu nie można mieć dostępu.

Dowiedziałem się również, że aby uzyskać pełen dostęp do systemu użytkownik korzystający z urządzenia mobilnego powinien mieć prawa administratora. Wtedy może korzystać bezpośrednio z zasobów systemowych pomijając pośredniczące w tym API lub do których nie było przygotowanych interfejsów. Aby nadać prawa administratora należy urządzenie „zrootować". Jest to termin określający przeprowadzenie operacji mających dać dostęp do urządzenia, jako superuser, czyli administrator. Niestety rootowanie telefonu nie jest wspierane przez Androida i w każdym wypadku oznacza ono utratę gwarancji urządzenia.

## 4.5 Komunikacja przez Wi-Fi

Główna idea, która przyświecała powstaniu opisanej pracy była, aby umożliwić bezpośrednią komunikację pomiędzy urządzeniami opierając się na sieciach bezprzewodowych. W takich sieciach do ich rozpoznawania używane są ramki konfiguracyjne. Budowa ramki została przedstawiona na rysunku 2. Zawierają one pola służące do rozpoznania urządzenia i sieci.

Pierwotnym pomysłem na rozwiązanie zadania komunikacji było wykorzystanie ramek konfiguracyjnych. Zabieg ten miał polegać na odpowiednim spreparowaniu ramki i wypełnieniu określonych pól w Frame Body tak, aby móc dokonać identyfikacji aplikacji oraz oferty. Tak przygotowane ramki z ofertami zostałyby rozgłaszane przez nadajnik Wi-Fi urządzenia a inne

urządzenia mogłyby je odbierać. Ramki konfiguracyjne są rozsyłane, co określony czas, zazwyczaj wynoszący kilkanaście milisekund. Każde urządzenie z włączonym modułem Wi-Fi odbiera cały czas rozsyłane przez inne sprzęty ramki. Technika ta nie powodowałaby również nadmiernego zużycia energii niż jest to nominalnie przy włączonym module sieci bezprzewodowej. Takie podejście pozwoliłoby na posiadanie wielu ofert w jednym czasie, ponieważ na każdą ramkę przypadałaby jedna oferta. Oferty mogłyby być rozsyłane po kolei i tak samo odbierane i następnie przetwarzane. Aplikacja wykonywałaby te czynności. Nadawałaby żądania wysłania określonych ramek do systemu. Również pobierałaby z systemu odebrane ramki i je analizowała w celu odnalezienia oferty spełniającej podane w programie wymagania.

Niestety twórcy Androida nie przewidzieli możliwości bezpośredniego odbierania i nadawania ramek konfiguracyjnych poprzez aplikację. Nie przygotowali żadnego interfejsu mogącego to wspierać. Jedyną możliwością, aby zrealizować w ten sposób komunikację pomiędzy urządzeniami było zrootowanie narzędzia, czyli nadanie użytkownikowi praw administratora. Nikt nie przeprowadzał badań, jaki procent użytkowników posiada aparat z prawami administratora, lecz analizując wypowiedzi na forach poświęconych rootowaniu urządzeń a także robiąc wywiad wśród znajomych. Doszedłem do wniosku, że urządzeń z prawami superuser nie może być więcej niż 5% spośród wszystkich. Robiąc również szybką analizę narzędzi systemowych Androida. Wysnułem wniosek, że aby móc preparować ramki na potrzeby aplikacji potrzebne jest dodanie stosownych funkcji do systemu, czyli modyfikacja całego systemu operacyjnego Android. Taki zabieg pozwoliłby na sprawne i łatwe działanie aplikacji. Aczkolwiek wątpliwym jest, aby ktokolwiek chciał instalować zmodyfikowany system specjalnie do korzystania z jednej dodatkowej aplikacji. Z powodu wystąpienie tych trudności zdecydowałem się na znalezienie innego rozwiązania.

Technologia Wi-Fi Direct, która stosunkowo od niedługiego okresu czasu jest dostępna w Androidzie albowiem dopiero od wersji Androida 4.0, która była wydana 19.10.2011 roku. Jest to technologia tak samo jak Wi-Fi działająca bezprzewodowo. Została stworzona również przez to samo stowarzyszenie – Wi-Fi Alliance. Największą zaletą w odróżnieniu od Wi-Fi jest bezpośrednia komunikacja pomiędzy urządzeniami, bez potrzeby łączenia się z punktem dostępowym i przesyłania wszystkich danych przez nie. Można porównać tę technikę do połączeń Bluetooth. Z tym, że korzystając z Wi-Fi Direct jest o wiele większy zasięg i prędkość przesyłania danych. Zdecydowałem się wykorzystać tę technologię do rozwiązania zadania komunikacji pomiędzy urządzeniami. Choć urządzeń z Wi-Fi Direct jest mniej niż z zwykłym Wi-Fi, to i tak jest ich zdecydowanie więcej niż urządzeń z prawami administratora. Z powodu, że jest to nowa technologia można wywnioskować, że liczba urządzeń posiadających możliwość korzystania z Wi-Fi Direct będzie stale się powiększała.

W tym wypadku Android do obsługi modułu Wi-Fi Direct przygotował specjalny framework o nazwie Wi-Fi Peer-to-Peer (P2P). Ten zestaw interfejsów został wprowadzony od Androida w wersji 4.0. Umożliwia on między innymi przeszukiwanie dostępnych urządzeń i nawiązywanie z nimi połączeń. A także powiadamia o nowych urządzeniach w zasięgu i o pomyślnej bądź negatywnej próbie nawiązania połączenia.

Z powodu oficjalnego wsparcia od Androida i przygotowanych API do korzystania z Wi-Fi Direct zdecydowałem się na wykorzystanie tego mechanizmu w komunikacji pomiędzy urządzeniami.

## 4.6 Sposób opisu treści rozgłaszanej

Od metody opisu haseł zależy działanie całego programu. W zależności od sposobu przetrzymywania treści mogą być użyte całkiem inne mechanizmy do analizy jak i rozsyłania ofert. Przyjąłem, że będą dwa rodzaje ofert. Oferta kupna wystawiana przez użytkownika odbierającego oferty od innych

użytkowników oraz oferta sprzedaży wystawiana przez użytkownika rozgłaszającego swoją ofertę innym użytkownikom. Mając tak zadane oferty należało jej treść odpowiednio zapisać.

Jeżeli oferta byłaby w formie ciągłego tekstu to za każdym razem należałoby rozpatrywać i przesyłać treść, jako całość, co by dość utrudniło sprawną komunikację. Również porównywanie ofert i szukanie odpowiadającej do oferty użytkownika byłoby dość skomplikowane. Wymagałoby to implementacji algorytmów wyszukujących ogłoszenia podobne, które mogą być napisane przy użyciu wyrazów synonimicznych jak i również zdania mogą mieć całkiem odmienną składnię i styl. W związku z tym treść oferty na pewno byłaby bardzo dobrze i dokładnie opisana oraz przez każdego użytkownika zrozumiana. Jednakże dla maszyny odnalezienie oferty interesującej użytkownika byłoby praktycznie niewykonalne ze względu na trudność i złożoność zadania.

Innym podejściem jest użycie słów kluczowych, zwanych potocznie tagami. Dzięki portalom społecznościowym robią się coraz bardziej popularne. W tym wypadku oferty składałyby się ze słów podanych przez użytkownika. Cała treść oferty mieściłaby się w obrębie kilkunastu słów, co nie stanowiłoby problemu w przesłaniu do innego użytkownika. Samo dopasowanie ofert również byłoby bardzo proste. Wystarczającym rozwiązaniem byłoby zastosowanie algorytmu porównującego słowa i ewentualnie ich synonimów z przygotowanej uprzednio bazy synonimów pobranych ze słownika. Dopasowanie oferty następowałoby, jeżeli odgórnie przyjęty procent zgodności słów zostałby przekroczony. Jednakże sam opis oferty pozostawiałby wiele niedomówień, ponieważ nie jest możliwym zawarciem całej oferty za pośrednictwem pojedynczych słów. Użytkownik miałby szybko i w łatwy sposób dostępne dopasowane oferty, ale musiałby podjąć dodatkowe kroki, aby poznać szczegóły oferty. Również dopasowanie ofert nie musiałoby zawsze zwracać wartościowych z punktu widzenia użytkownika. Wiele przedmiotów

różnych rodzajów, czy kategorii można opisać w nawet ten sam sposób. Nie oznacza to jednak, że jest to ten sam interesujący użytkownika przedmiot.

Inną możliwością było sięgnięcie po sprawdzone już sposoby, które większość ludzi zna i są dla nich odpowiednie. Serwisy aukcyjne lub różnego rodzaju sklepy stosują w pewien sposób mieszaninę opisanych powyżej metod. Występują w nich rodzaj, kategorie, parametry oraz opis słowny. Wszystko sprowadza się do podania rodzaju ogłoszenia(kupna lub sprzedaży). Wybrania odpowiedniej kategorii ogłoszenia, co od razu eliminuje problem znalezienia odpowiedniej oferty. Opis ogłoszenia następuje poprzez podanie jego z góry ustalonych parametrów. Parametry pozwalają dopasować do siebie odpowiednie znalezione z tej kategorii oferty. Na koniec podawany jest opis ogłoszenia, które jest jego dopełnieniem. W korzystaniu z takiego systemu największym problemem jest posiadanie bazy kategorii oraz ich parametrów, które są zależne od danej kategorii. Natomiast, kiedy ma się odpowiednią bazę kategorii samo dopasowanie ofert działa bardzo sprawnie i bez pomyłek. Dopasowanie ofert odbywa się na podstawie kategorii oraz spełnieniu warunków parametrów. Natomiast sam opis jest uzupełnieniem, gdzie użytkownik może się już dowiedzieć szczegółów oferty lub sposobu jej finalizacji.

Wybrałem taki sposób opisu treści z powodu możliwości największej kompletności opisu oferty w połączeniu z nieskomplikowaną metodą wyszukiwania odpowiadających ofert. Do implementacji kategorii początkowo miało posłużyć API Ceneo.pl, które ma bardzo dobrze rozwinięte kategorie oraz parametry. Pozostaje jeszcze problem implementacji takiej struktury w programie. Duże i skomplikowane struktury są przechowywane w plikach, najczęściej wykorzystując Schema *XML* – schemat uniwersalnego języka znaczników, który został stworzony, aby reprezentować tego typu dane.

*Rysunek 4 Przykładowa Schema XML*

```
<?xml version="1.0" encoding="UTF-8"?>
<ArrayOfCategory xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <Category>
    <Id>40</Id>
    <Name>Fotografia</Name>
    <Subcategories>
      <Category>
        <Id>46</Id>
        <Name>Aparaty fotograficzne</Name>
        <Subcategories/>
          <Category>
            <Id>3013</Id>
            <Name>Aparaty cyfrowe z wymienną optyką</Name>
            <Subcategories/>
          </Category>
                .
                .
                .
                .
                .
                .
      </Category>
      </Subcategories>
  </Category>
</ArrayOfCategory>
```

Na rysunku 4 przedstawiono przykładową strukturę zapisaną za pomocą XML. Zakładając, że pełny schemat kategorii oraz parametrów będzie wielkości, co najmniej kilkudziesięciu tysięcy linii kodu należało znaleźć sposób jego implementacji w programie. Dobrym rozwiązaniem jest przeprowadzenie analizy składniowej pliku z całą strukturą XML, pozwoli to na automatyczne przypisywanie kategorii oraz ich podkategorii wraz z nazwami parametrów. Po dokonaniu analizy składniowej takiego pliku przez aplikację, będzie ona wyposażona w pełną strukturę i hierarchię kategorii. Ostatnim problemem pozostaje ustawienie odpowiedniego typu zmiennych dla poszczególnych parametrów. Należy założyć, że atrybuty mogą być typu binarnego, liczbowego lub znakowego.

Najprostszym sposobem byłoby zastosowanie do wszystkich zmiennych typu znakowego, aczkolwiek takie podejście umożliwia podanie przez

użytkownika wielu błędnych danych i utrudnia porównywanie parametrów. W zasadzie ogranicza możliwości tylko do porównywania identyczności atrybutów. Jest to jedyna możliwość, jeżeli typ atrybutu nie jest podany w schemacie.

Natomiast zakładając, że typy atrybutów są podane wraz z parametrami można ten problem rozwiązać za pomocą instrukcji wyboru *switch*. Każdy parametr jest typu *String.* W zależności od wczytanego typu atrybutu zainicjowany parametr należy rzutować na dany typ. Należy pamiętać, że takie operacje trzeba wykonywać za każdym razem, kiedy chce się skorzystać z danego parametru. Rozwiązanie to nie jest skomplikowane, choć jest czasochłonne.

Często w serwisach takich jak Ceneo.pl, czy Allegro parametry typu znakowego mają ustalone możliwe wartości i mogą tylko je przyjąć na zasadzie wyboru z listy. W takiej sytuacji wszystkie parametry mogą być typu liczbowego, gdzie dla typu binarnego mogą przyjąć wartość 0 jako fałsz lub 1 jako prawdę, dla typu znakowego będzie to pozycja na liście dostępnych opcji, a dla typu liczbowego będzie to ich wartość. Takie rozwiązanie bardzo ułatwia i skraca czas implementacji całego programu. Natomiast dochodzą do analizy składniowej schematu kolejne elementy – listy wyboru.

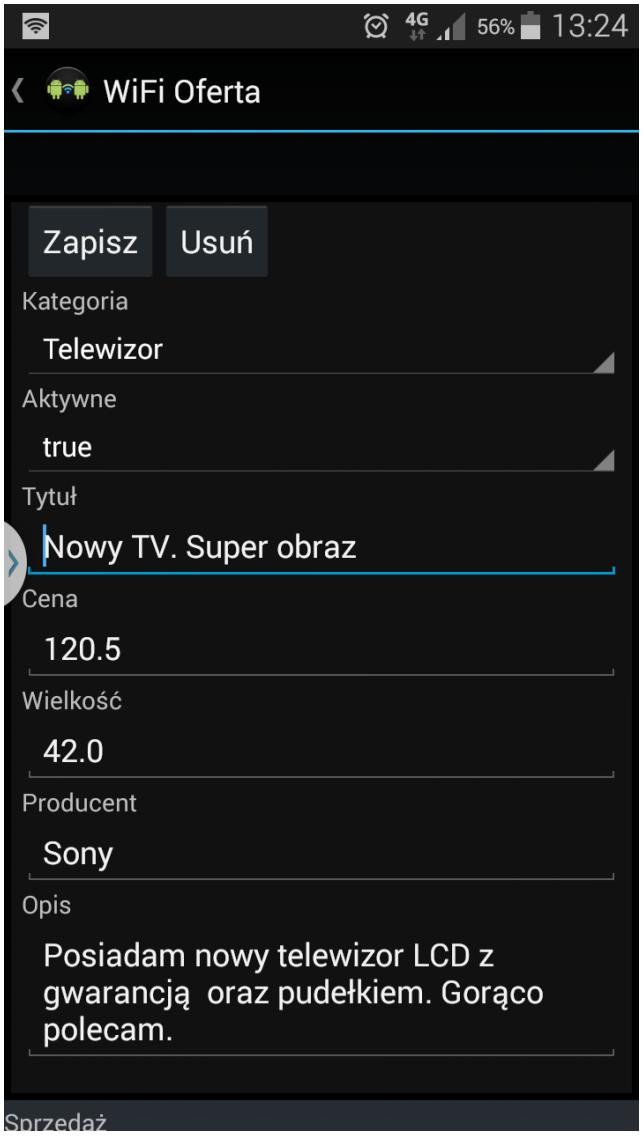
W zależności od posiadanego API, można implementować struktury kategorii na różne sposoby, jednak w tym wypadku Ceneo.pl odmówiło współpracy. W związku z tym, że aplikacja powstała w ramach tej pracy, miała z założenia być prototypem. Przyjęto, że wystarczającym będzie zaimplementowanie dwóch kategorii wraz z wybranymi parametrami do pełnego przetestowania aplikacji.

Oferta zawiera informację o:

- Rodzaj – kupno lub sprzedaż.
- Kategoria – przynależność do ustalonej grupy.

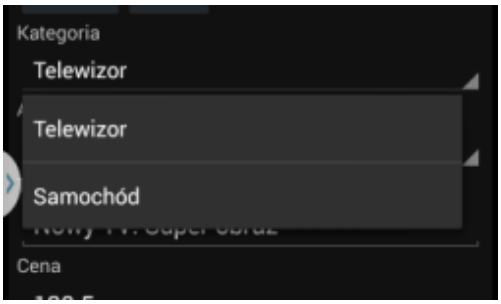
- Parametr1 – cecha charakterystyczna dla kategorii.
- Parametr2 – druga charakterystyczna cecha kategorii.
- Tytuł – krótki opis ogłoszenia.
- Cena – kwota wartości oferty.
- Aktywne – informacja o tym, czy ogłoszenie w danej chwili ma być analizowane przez aplikację. Tylko jedna oferta sprzedaży może być w danej chwili aktywna.
- Opis – szczegółowy opis oferty oraz zawarcia transakcji.

*Rysunek 5 Zarządzanie wybraną ofertą*




Na rysunku 5 przedstawiono okno zarządzania wybraną ofertą. Rodzaj oferty jest wybierany przy dodawaniu oferty. Można dodać ofertę kupna lub sprzedaży. Kategoria oferty jest wybierana z listy dostępnych ofert (przedstawionych na rysunku 6).

*Rysunek 6 Wybór kategorii*




Atrybut - Aktywne również jest wybierany z listy jako Tak lub Nie. Cena jest wpisywana jako liczba zmiennoprzecinkowa. Następne są dwa parametry zależne od kategorii. Pierwszy z nich jest podawany jako

liczba zmiennoprzecinkowa natomiast drugi parametr jest napisem. Na końcu podawany jest opis również jako napis. Ze względów na potrzebę przesyłania i porównywania parametrów oferty, długość ich pól została ograniczona. Wynoszą one odpowiednio:

- Tytuł – 25 znaków
- Cena – 6 znaków
- Parametr1 – 8 znaków
- Parametr2 – 8 znaków
- Opis – 250 znaków

Parametry rodzaj, kategoria oraz aktywne mogą przyjąć tylko wartości dostępne z listy, więc nie posiadają ograniczeń długości pól.

## 4.7 Sposób rozsyłania ofert

Po ustaleniu metod komunikacji i opisu oferty należało ustalić, w jaki sposób przesyłane będą oferty. Pierwszym zamysłem było nawiązywanie połączenia z każdym urządzeniem i przesyłaniem oferty. Było to rozwiązanie podyktowane udostępnionym API przez Androida umożliwiającym te czynności w prosty i szybki sposób. Jednak już podczas implementacji okazało się, że nie jest możliwe nawiązanie połączenia bez uprzedniej ingerencji użytkownika. Użytkownik musi potwierdzić każdą próbę nawiązania połączenia w ciągu 30 sekund, inaczej nie zostanie ono nawiązane. Jest to spowodowane restrykcyjnym podejściem do spraw bezpieczeństwa w Androidzie. Chociaż nawiązanie połączenia w obrębie dwóch aplikacji na różnych urządzeniach nie każdemu wydaje się być równie niebezpieczne jak producentom systemu. Przeszukując grupy dyskusyjne Androida natrafiłem na wypowiedzi deweloperów oraz producentów, gdzie mówiono o zniesieniu wymagania potwierdzania prób nawiązania połączeń. Aczkolwiek nie wiadomo, kiedy mogłoby to nastąpić. Natomiast wiadomo, że istniał kiedyś mechanizm to obsługi potwierdzeń z poziomu aplikacji bez ingerencji użytkownika, lecz

zrezygnowano z niego i został on z systemu usunięty ze względów bezpieczeństwa.

Rozsądnym rozwiązaniem na daną chwilę wydało się być modyfikowanie nazwy urządzenia używanej przez Wi-Fi Direct do celu identyfikacji kategorii oferty oraz jej parametrów. Następnie nawiązanie połączenia wymagającego potwierdzenia tylko w wypadku żądania pobrania szczegółowego opisu oferty. W zależności od wersji Androida nazwa urządzenia ma różną maksymalną długość. Od Androida wersji 4.0 do 4.3 włącznie jest to 21 znaków. Natomiast od Androida 4.4 jest to już 31 znaków. W związku z tym dane oferty przedstawione w nazwie należało w pewien sposób ograniczyć. Nazwę urządzenia zawierającą ofertę trzeba odpowiednio zabezpieczyć, aby osoby postronne nie mogły mieć dostępu do ofert, tylko sama aplikacja po odszyfrowaniu nazwy może ją podać użytkownikowi korzystającemu z niej. Do zabezpieczenia przed postronnymi osobami zostało użyte kodowanie *base64*. Pozwala ono zakodować nazwę, żeby była ona niezrozumiała dla osoby, która nie wie, jakiego kodowania użyto. Nazwę można prosto odszyfrować za pomocą tego samego algorytmu. Ze względu na brak poufnych danych w nazwie urządzenia, a jedynie ze względu na uniemożliwienie postronnej osobie przeglądania ofert, jest to wystarczające rozwiązanie.

Niestety Android nie przewidział w swoich interfejsach potrzeby modyfikacji danych urządzenia korzystających z Wi-Fi Direct. Po poszukiwaniach rozwiązania tego problemu zarówno na forach dyskusyjnych, jak i przeglądając kod źródłowy interfejsów, doszedłem do wniosku, że producenci Androida często stosują metody ukryte. Są one oznaczone w kodzie programu dodatkowym znacznikiem - *@hide.* Nie ma do nich bezpośredniego dostępu w momencie pisania kodu. Taki dostęp jest dopiero możliwy podczas wykonywania kodu. Według producentów systemu metody są ukrywane wtedy, kiedy niepewna jest ich przyszłość. W celu ułatwienia ich wycofania z API i zapewnienia pełnej kompatybilności są ukryte aż do momentu, kiedy na pewno

zostaną na stałe w danym interfejsie. Dostęp do takich metod możliwy jest dzięki zastosowaniu mechanizmu refleksji.

Mechanizm refleksji polega na operacjach na polach obiektu i obiektach, do których w momencie pisania kody nie posiadamy dostępu lub ich nie znamy. Dopiero w trakcie wykonywania programu możemy mieć dostęp do pola obiektu lub całego obiektu. Pozwala on na dynamiczne przystosowanie napisanego kodu do zachowania w trakcie wykonania.

Twórcy systemu przygotowali ukrytą metodę *setDeviceName* przyjmującą, jako argument zmienną typu String. Pozwala ona ustawiać nazwę urządzenia w Wi-Fi Direct. Metoda występuje od Androida 4.0 do 4.4 włącznie, niestety jej dalszy los jest nieznany. Wykorzystując tę metodę ustalam nazwę urządzenia zawierającą dane oferty. W związku z ograniczeniem długości nazwy urządzenia do 21 znaków zdecydowałem się na przekazanie tylko najważniejszych parametrów. Są to zaszyfrowane:

- id aplikacji – „+O/”, długość - 3 znaki
- kategoria oferty – liczba całkowita, dodatnia, długość - 1 znak
- cena oferty – liczba zmiennoprzecinkowa, poprzedzona „+”, maksymalna długość do 7 znaków
- parametr1 oferty – liczba zmiennoprzecinkowa, poprzedzona „+”, maksymalna długość do 9 znaków
- parametr2 oferty – napis, poprzedzony „/”, maksymalna długość do 9 znaków

Szyfrowanie przebiega stopniowo według wymienionych atrybutów albo do zakodowania wszystkich parametrów, albo aż długość zakodowanej nazwy nie przekroczy 21 znaków. W związku z narzuconymi ograniczeniami na długość poszczególnych parametrów w najgorszym wypadku nazwa urządzenia będzie złożona z id aplikacji, kategorii oraz ceny. Żeby móc zobaczyć resztę parametrów użytkownik musi się połączyć z innym urządzeniem. Wtedy zostanie

przedstawiona pełna oferta wraz ze wszystkim atrybutami. Główną wadą zastosowanego kodowania *base64* jest dopisywanie dodatkowego bajta do każdych 3 bajtów niezakodowanego słowa. Nie jest to rozwiązanie idealne, aczkolwiek wystarczające na sprawne działanie i odnajdowanie jak najbardziej interesujących użytkownika ofert oraz umożliwienie nawiązania połączenia w celu pobrania całej oferty wraz z szczegółowym opisem.

Przy wykorzystywaniu nazwy urządzenia, jako identyfikatora oferty należy zwrócić uwagę na wiążące się z tym problemy. Do głównych przeszkód są niewątpliwie zaliczane powtarzające się nazwy dla ofert różnych użytkowników oraz mylna interpretacja nazwy urządzenia niekorzystającego z tej aplikacji, jako oferty. Rozwiązanie obu tych problemów jest dosyć trywialne. Aby zapobiec wielokrotnym identycznym nazwom wystarczy dodać do niej liczbę porządkową. Aplikacja, na której jest wystawiana oferta sprzedaży skanuje dostępne sieci, jeżeli nazwa, którą chce nadać jest już zajęta to inkrementuje liczbę porządkową w nazwie oraz powtarza krok, aż do momentu posiadania unikalnej nazwy i ją wtedy nadaje. Po ustawieniu nazwy urządzenia również ciągle jest sprawdzana jej unikalność na wypadek, gdyby w zasięgu pojawiło się urządzenie z taka samą ofertą. W celu wykluczenia błędnej interpretacji nazwy urządzenia jako oferty można zastosować sumę kontrolną dodawaną na końcu nazwy w celu sprawdzenia czy jest ona zgodna z oczekiwanym wynikiem. Tylko dla zgodnych sum należy traktować nazwę urządzenia jako rzeczywistą ofertę. Podane rozwiązania wykluczają szanse na zajście wymienionych problemów

Metoda wykorzystania nazwy do określania oferty ograniczyła możliwą ilość rozgłaszanych ogłoszeń do pojedynczego ogłoszenia w danej chwili. Spowodowane jest to tym, że urządzenie ma tylko jedną nazwę. Obecnie, niektóre routery z odpowiednim oprogramowaniem pozwalają na konfigurację i jednoczesne działanie wielu sieci bezprzewodowych o różnych identyfikatorach – SSID używając pojedynczego tunera Wi-Fi. Oprogramowanie DD-WRT

dedykowane routerem na bazie Linuksa pozwala na działanie nawet do 16 sieci naraz. W takim wypadku na każdą konfigurację mogłaby przypadać jedna oferta sprzedaży, które byłyby jednocześnie aktywne. Niestety twórcy Androida nie widzą potrzeby, aby ustalać wiele sieci z jednego urządzenia.

W związku z powyższym aplikacji można posiadać maksymalnie jedną aktywną ofertę sprzedaży. Na jej podstawie jest ustawiana nazwa urządzenia. Aplikacja ma również podane przez użytkownika oferty kupna. Program skanuje dostępne urządzenia i sprawdza ich nazwę po zdekodowaniu. Jeśli natrafi na urządzenie mające nazwę pasującą do posiadanych ofert kupna wyświetli ofertę i jeżeli użytkownik uzna ją za interesującą może nawiązać połączenie z sprzedającym w celu pobrania pełnej oferty. Jeśli nadal będzie zainteresowany ofertą może odszukać w pobliżu drugiego użytkownika i dokonać transakcji. Drugi użytkownik może w szczegółowym opisie np. opisać jak go można rozpoznać lub znaleźć.

W ten sposób zostały nadane pewne ograniczenia. Posiadanie maksymalnie jednej oferty sprzedaży oraz konieczność potwierdzania nawiązania połączenia przez sprzedającego w celu przesłania pełnej oferty. Ograniczając w ten sposób możliwości aplikacji kierowałem się możliwościami implementacji zapewniającej poprawne działanie aplikacji, a także jak najmniejszego utrudnienia dla użytkowników. Użytkownik często może chcieć znaleźć parę rzeczy, które go akurat interesują. Jednocześnie użytkownik, który się przemieszcza być może nie ma przy sobie miejsca na więcej niż jedną rzecz do zaoferowania lub w przypadku zastosowania w pośrednictwie nieruchomościami osoba może przebywać w mieszkaniu, które ma do zaoferowania. Z tych powodów uważam nałożone ograniczenia za rozsądne, choć mogące utrudnić korzystanie z aplikacji, ale są one konieczne.

## 4.8 Testowanie aplikacji

Równie ważnym elementem powstawania aplikacji jest jej przetestowanie. Testowanie przeprowadza się w celu sprawdzenia poprawności działania programu i odnalezieniu ewentualnych defektów, które są manifestacją błędów. Najczęściej są to błędy we własnym kodzie źródłowym. Aczkolwiek zdarza się też, że błędy mogą również występować w wykorzystywanych interfejsach lub zaimportowanych bibliotekach. Bardzo ważnym elementem jest powtórzenie testów po naprawie błędów, aby sprawdzić, czy naprawiając jeden błąd nie spowodowaliśmy ujawnienia innych błędów. Z powodu potrzeby sprawdzenia poprawności zachowania się aplikacji oraz jej wydajności potrzebowałem przeprowadzić testy. Niestety, uczelnia nie posiada urządzeń mobilnych z Androidem przeznaczonych do użytku studentom. Sam jestem w posiadaniu jednego smartfonu mogącego posłużyć do testów. Jako że jest to aplikacja komunikująca się z innymi urządzeniami potrzebne były, co najmniej dwa urządzenia z Androidem, co najmniej w wersji 4.0.

Koszt zakupu dodatkowego urządzenia przerastał budżet przeznaczony na powstanie pracy. Wobec czego potrzebne było znalezienie innej metody testowania.

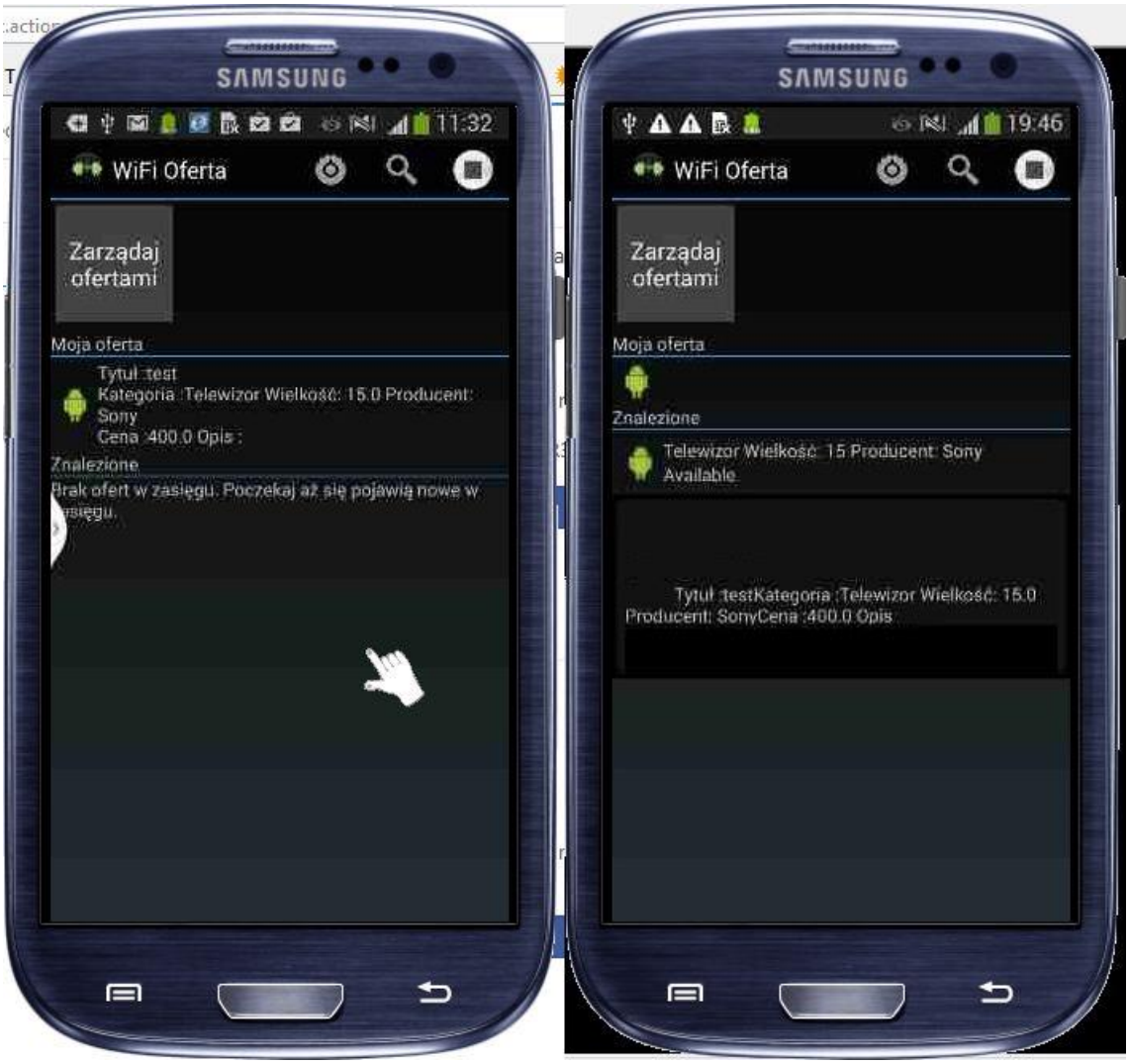
Jak wspominałem wcześniej Android zapewnia bardzo dobre wsparcie dla programistów. W tym również narzędzia do testowania. Jest to emulator Androida wbudowany w wtyczkę ADT, którą się instaluje do IDE Eclipse. Jest to odpowiednie rozwiązanie, ponieważ można emulować w ten sposób wiele urządzeń mobilnych z różnymi wersjami Androida i konfiguracjami urządzenia (m.in. różne rozdzielczości). Niestety emulator został pozbawiony opcji Wi-Fi. Twórcy emulatora z niewiadomych nikomu pobudek nie udostępnili opcji korzystania z sieci bezprzewodowych. Obecnie jest to problem szeroko

dyskutowany na forach i grupach dyskusyjnych Androida. Wszakże brak w nich jest jakiegokolwiek oficjalnego stanowiska ze strony Androida.

Na wspomnianych powyżej forach natknąłem się na informację, iż firma Samsung udostępnia dla zarejestrowanych na ich stronie internetowej programistów zdalne laboratorium. Działanie takiego laboratorium polega na tym, że deweloper ma możliwość połączenia się z rzeczywistym urządzeniem poprzez sieć. Po połączeniu się z urządzeniem można instalować na nim aplikacje z własnego komputera oraz w pełni je kontrolować. Wadami laboratorium były między innymi ograniczona liczba urządzeń, jako że są to rzeczywiste urządzenia ich liczba nie może być nieskończona. Platform, z których mogłem korzystać, czyli z Androidem w wersji, co najmniej 4.0 było około 20 sztuk. Kolejną wadą było ograniczenia czasowe korzystania z urządzeń. Maksymalnie dziennie można korzystać z nich 5 godzin w sumie, ponieważ do testowania potrzebowałem 2 urządzeń czas ten ulegał skróceniu do 2.5 godzin dziennie. Czas korzystania z urządzenia należało zadeklarować przed nawiązaniem połączenia. Z powodu, że były to rzeczywiste urządzenia, niektóre z nich nie działały poprawnie lub nie dało się z nimi połączyć, a zadeklarowany czas wtedy przepadał.

Z braku lepszych alternatyw zdecydowałem się na przeprowadzanie testów poprawnego działania w laboratorium firmy Samsung. Z opiekunem pracy ustaliliśmy, że testowanie wydajnościowe odbędzie się na bezpośrednio posiadanych lub pożyczonych urządzeniach, w celu możliwości określenia rzeczywistych czasów potrzebnych na wykonanie poszczególnych elementów aplikacji.

*Rysunek 7 Aktywna oferta oraz jej pobranie na drugim urządzeniu*



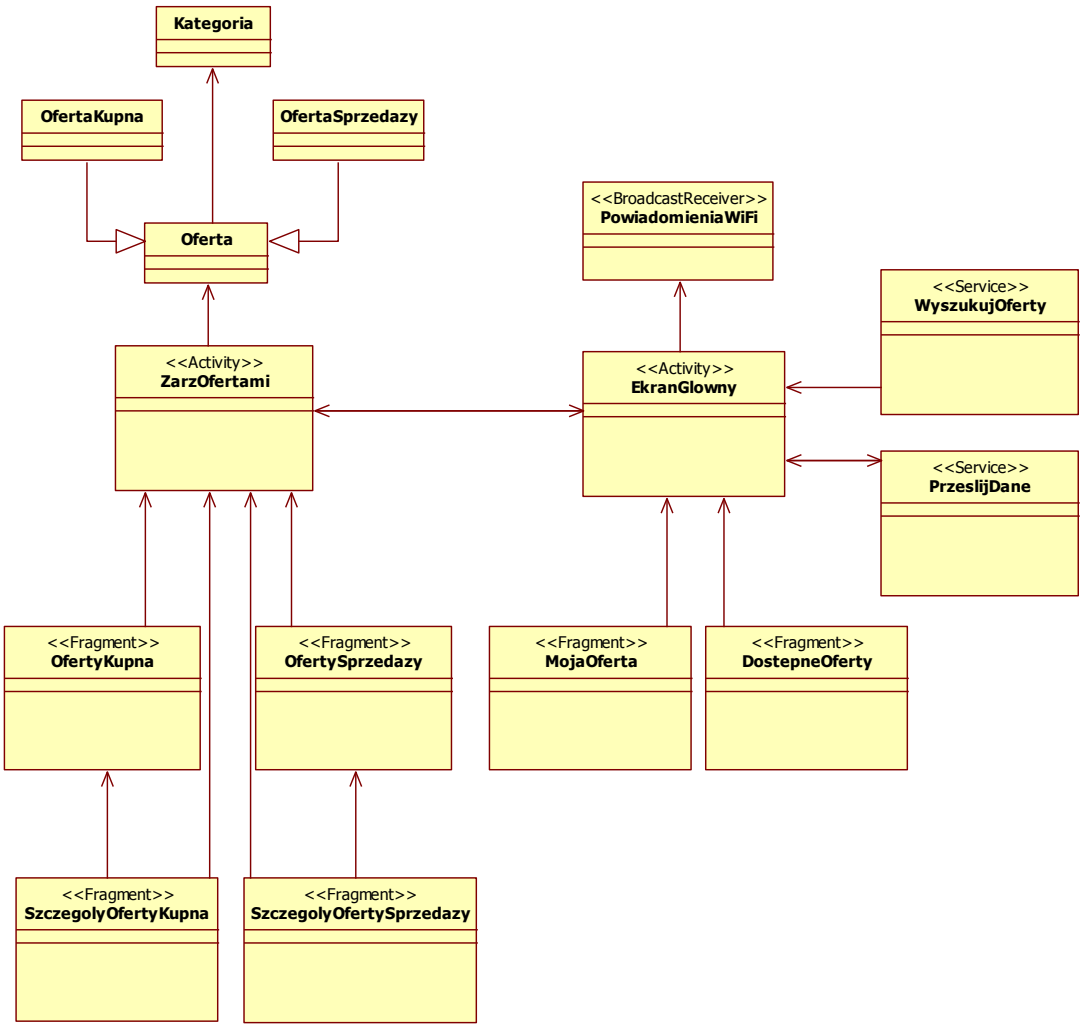
# 5 Wyniki i testy

## 5.1 Prototyp

W wyniku implementacji według powyższych rozwiązań powstał prototyp aplikacji o nazwie *WiFi Oferta.* Umożliwia on realizowanie w pełni postawionych tej aplikacji wymagań.

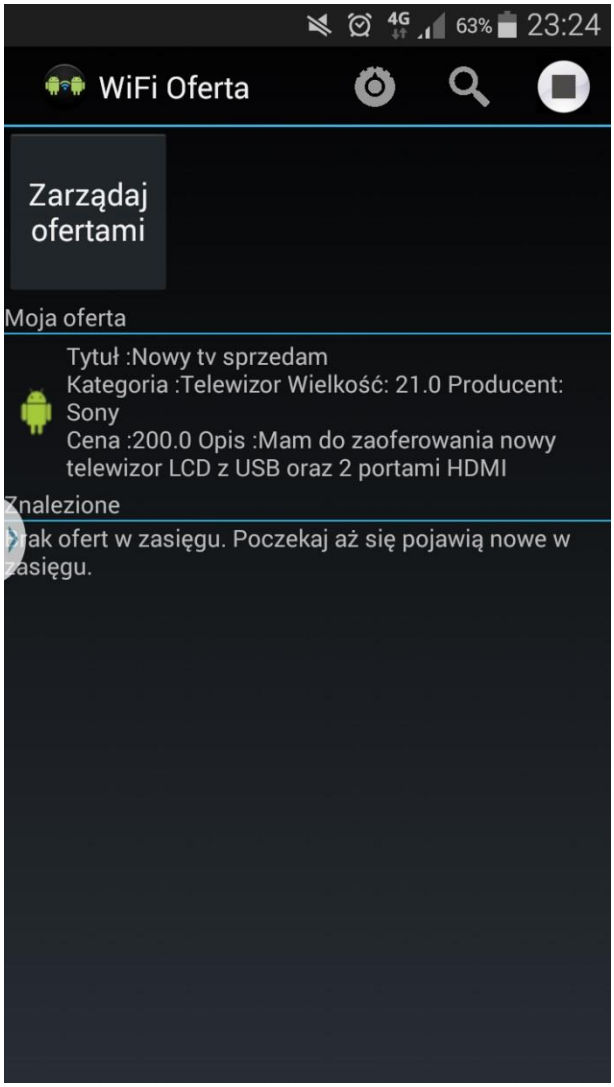
Aplikacja składa się z dwóch aktywności oraz dwóch usług. Aktywności składają się z fragmentów, aby móc intuicyjnie zarządzać aplikacją. Szczegółowy zarys znajduje się na rysunku 8.

*Rysunek 8 Schemat aplikacji*




Aplikacja została zrealizowana w sposób przejrzysty i intuicyjny dla użytkownika. Po uruchomieniu aplikacji pokazuje się użytkownikowi główny

ekran programu – *EkranGlowny*. Użytkownik od razu ma podgląd na aktualnie aktywną ofertę sprzedaży – *MojaOferta*. Ma również listę dostępnych dla niego ofert, wyselekcjonowanych przez program – *DostepneOferty*.




*Rysunek 9 Ekran główny*

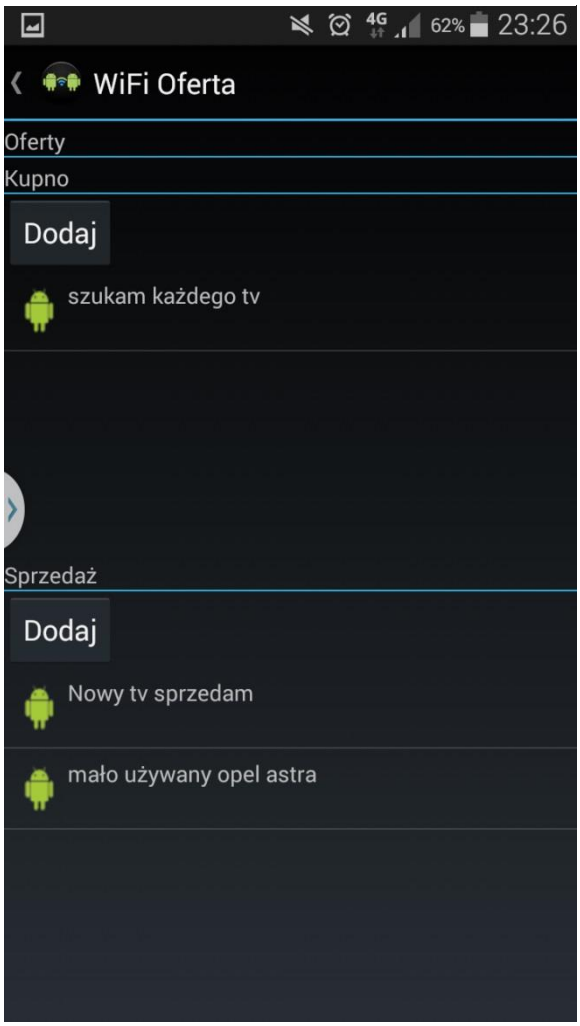
Dostępne są również przyciski pozwalające zatrzymać lub wznowić przeszukiwanie dostępnych urządzeń. Jest również możliwość bezpośredniego przejścia do ustawień Wi-Fi urządzenia w celu jego włączenia, aby aplikacja mogła działać. Wraz ze startem programu włączana jest usługa – *WyszukujOferty*, wyszukująca urządzenia korzystające z Wi-Fi Direct. Usługa działa w tle przez cały czas bycia aktywnym aplikacji. Nawet po wyjściu z niej, bez zamknięcia. W przypadku, gdy są dostępne oferty użytkownik może wybrać daną ofertę. Nawiązać połączenie z innym urządzeniem i zostaje automatycznie pobrana cała oferta ze szczegółami. Przy nawiązaniu połączenia uruchamiana jest usługa *PrzeslijDane* oczekująca na ofertę. Po przesłaniu jest ona wyświetlana na głównym ekranie przy zaznaczonej ofercie. Usługa jest zatrzymywana.

Nawiązanie połączenia odbywa się przez przesłanie żądania połączenia przez użytkownika poszukującego do użytkownika rozgłaszającego. Użytkownik wysyłający czeka na zatwierdzenie tej operacji. W przeciwnym wypadku

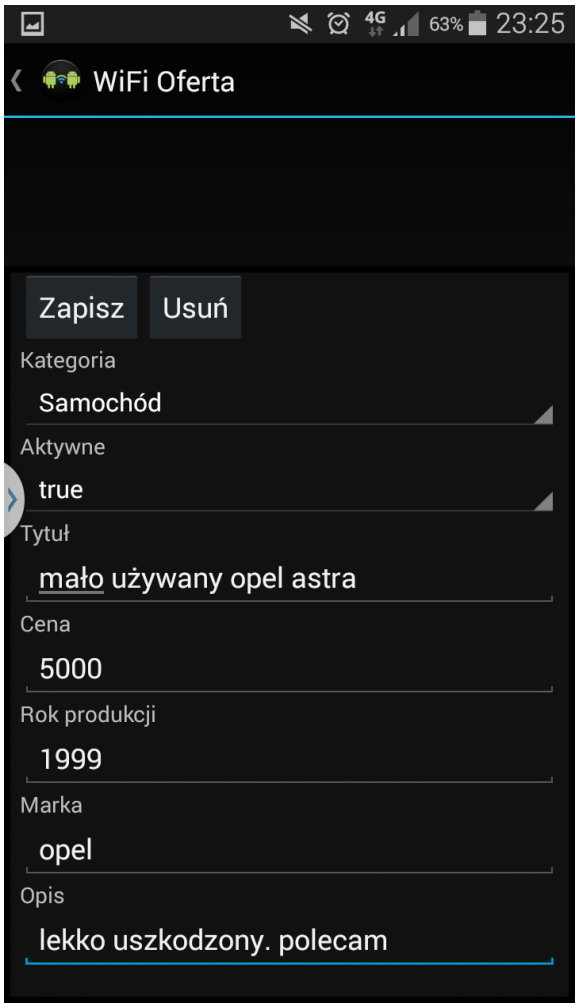
dostanie powiadomienie o nieudanej próbie nawiązania połączenia. W wypadku zatwierdzenia po chwili wyświetlona zostanie pełna oferta w oknie aplikacji. Użytkownik rozgłaszający dostaje powiadomienie o próbie nawiązania połączenia przez dane urządzenie. Może je zaakceptować i automatycznie zostanie przesłana oferta lub je odrzucić.

Użytkownik może przejść do zarządzania ofertami. Spowoduje to zatrzymanie aktualnej aktywności i uruchomienie aktywności związanej z zarządzaniem ofertami – *ZarzOfertami*. Użytkownik może przeglądać istniejące oferty oraz je edytować, dodawać lub usuwać. Wszystkie oferty użytkownika są zachowywane do pliku w celu ich odtworzenia przy następnym starcie aplikacji.

*Rysunek 10 Lista ofert*




*Rysunek 11 Edycja oferty*




Zostały zaimplementowane dwie kategorie z dodatkowymi parametrami.

1. Samochód
    a. Rok produkcji
    b. Marka
2. Telewizor
    a. Wielkość
    b. Producent

## 5.2 Szczegóły implementacji

Przedstawię pewne detale implementacji, które są nietypowe lub ciekawe dla programowania, a przy tym nieczęsto spotykane. Fragmenty kodu źródłowego aplikacji zostały przedstawione w rozdziale siódmym „Dodatek. Fragmenty kodu źródłowego".

W celu analizowania i przetwarzania ofert potrzebowałem, aby były one przekazywane pomiędzy fragmentami oraz aktywnościami. Rozwiązaniem tego w Androidzie są *zamierzenia (Intents).* Przy uruchamianiu, bądź zatrzymywaniu aktywności są przekazywane *zamierzenie*. Można do nich również zapisać dane bądź obiekty. Obiekty te muszą być serializowane (*serializable)* lub parcelowane (parcelable). Muszą zostać zamienione w szereg bajtów a potem odczytane i zamienione z powrotem na obiekty początkowe. Serializacja w języku Java jest bardzo prosta i polega w zasadzie na zaimplementowaniu interfejsu *Serializable*. Nie wymaga to już żadnego implementowania dodatkowych metod. Niestety sposób ten nie jest chętnie wspierany przez Androida, ze względu na wydajność. Twórcy zachęcają do korzystania z interfejsu *Parcelable.* Wymaga to jednak implementacji dodatkowych metod. Jednak ze względu na oficjalne wsparcie tego sposobu zdecydowałem się go używać. Zostało to pokazane na rysunku 12.

Nie jest to skomplikowana, lecz dosyć czasochłonna metoda. Każda zmiana atrybutów klasy wymaga zmian w paru miejscach. Często może prowadzić do błędów wykonania programu, kiedy jakiekolwiek zostanie pominięte bądź przez przypadek zmienione. Mając tak przygotowaną klasę można przesłać w *zamierzeniu* i odebrać.

Samo przekazanie *zamierzenia* odbywa się w prosty sposób – przez przesłanie go przez metodę uruchomiającą daną aktywność. W tym przypadku użyłem metody, która informuje, że aktywność zwróci wynik i będzie można uzyskać dane przesłane w zależności od otrzymanego rezultatu. Pokazano to na rysunku 13. Natomiast przy powrocie do aktywności sprawdzany jest wynik

i w zależności od niego jest odczytywane *zamierzenie* – rysunek 14. Odczytywane są oferty w nowej aktywności oraz jest uruchamiana usługa wyszukiwania nowych ofert. W zależności od ofert sprzedaży jest ustawiana nazwa urządzenia

Implementując aplikację korzystałem z techniki refleksji. Używanie tego mechanizmu było konieczne w celu korzystania z metody *setDeviceName*, która pozwala ustawić nazwę urządzenia. Jest ona zdefiniowana w klasie WiFiP2PManager i jest oznaczona jako ukryta, co zostało przedstawione na rysunku 15.

W technice tej najtrudniejsze jest dowiedzenie się, z jakiej metody, której klasy należy korzystać wraz z jej argumentami. W celu poprawnego wykonania należy pobrać klasę oraz jej metodę według nazw. W tym wypadku, ponieważ była to metoda, do której w normalnym wypadku nie ma się dostępu, należało dodatkowo ustawić ten dostęp i wywołać metodę z argumentami. Jest to zaprezentowane na rysunku 16.

## 5.3 Testy

Do testowania funkcjonalności aplikacji korzystałem z zdalnego laboratorium firmy Samsung. Natomiast mając już ukończoną aplikację należało wykonać szereg testów sprawdzających czas wykonania poszczególnych zadań oraz działanie w rzeczywistych warunkach.

Do tego celu posłużyły mi dwa smartfony. Na pierwszym urządzeniu posiadałem Androida w wersji 4.4. Drugie urządzenie posiadało go w wersji 4.1. Wykonałem 20 prób na każdy test, aby wyniki mogły być miarodajne. Każdy test uwzględniał również różne położenia urządzeń względem siebie. Z przeszkodami i bez nich. Wzięte pod uwagę również różnice w systemach operacyjnych i wykonywano po 10 testów w jednej konfiguracji i 10 testów w odmiennej konfiguracji.

### 5.3.1 Test szybkości odnajdywania urządzenia.

Test miał powiedzieć ile czasu urządzenia potrzebują, aby odnaleźć odpowiednią ofertę. Test polegał na posiadaniu w jednym urządzeniu aktywnej oferty kupna oraz na zmienianiu aktywności oferty sprzedaży dopasowanej na urządzeniu drugim. Czasy te wahały się od 1 do 7 sekund. Ze średnim wynikiem wynoszącym ok. 3,5 sekundy. Rozbieżności te nie były zależne od położenia urządzeń. Były one spowodowane zapętlonym wykonywaniem, co krótki interwał czasowy, funkcji odnajdującej urządzenia w usłudze uruchomionej w tle. Wyniki zależały, w którym momencie wykonywania była ta funkcja. Otrzymane wyniki mogę jak najbardziej uznać jako zadawalające.

### 5.3.2 Test czasu pobrania pełnej oferty.

W tym teście poznano rzeczywisty czas potrzebny na uzyskanie pełnej oferty sprzedaży z innego urządzenia. Na jednym urządzeniu była umieszczona aktywna oferta sprzedaży a na drugim odpowiednia oferta kupna. Po wykryciu ofert liczono czas od wybrania oferty i zażądania pobrania szczegółów oferty do momentu ich wyświetlenia. Początkowo test ten miałem zamiar zrealizować za pomocą dwóch mniejszych i niezależnych testach. Nawiązania połączenia między urządzeniami oraz przesłania danych. Jednak po paru próbach wykonanie drugiego testu okazało się zbyteczne, ponieważ oferta po zaakceptowaniu połączenia była wyświetlana w czasie poniżej 1 sekundy. Rzeczywiście w tym teście liczył się tylko czas nawiązania połączenia. Wyniósł on od 2 do 5 sekund, przy średnim wyniku ok. 2,8 sekundy na 20 dokonanych prób

Zostało udowodnione, że powstały prototyp aplikacji mobilnej działa poprawnie i wykonuje funkcje w czasie do zaakceptowania przez użytkowników. Cały proces odnalezienie i pobrania oferty wynosi ok. 10 sekund. Aczkolwiek czas pobrania oferty może ulec znacznemu wydłużeniu w przypadku zwlekania potwierdzenia połączenia przez osobę rozgłaszającą.

# 6 Wnioski i podsumowanie

Na zakończenie tej pracy chciałbym przedstawić pewien rodzaj podsumowania analiz i wniosków, które są wynikami moich działań oraz zebranych doświadczeń. Była to moja pierwsza stworzona tak rozbudowana aplikacja mobilna. Początkowo sądziłem, że umiejętności, które już posiadałem wystarczą, aby sprawnie mogła powstać ta aplikacja. Jednak tworzenie aplikacji mobilnych zgoła się różni od pisania programów na tradycyjne komputery.

Do największych różnic należał system operacyjny. Jest on oparty na systemie Linux. Jednakże restrykcyjne podejście producentów w tym wypadku do zasad bezpieczeństwa bardzo ogranicza możliwości deweloperów. Korzystanie z zasobów systemowych jest możliwe tylko przez specjalne interfejsy, które też na nie za wiele pozwalają. Tłumaczy to niewielką ilość aplikacji korzystających z Wi-Fi, ponieważ korzystanie z niego przez aplikację jest bardzo utrudnione oraz ograniczone.

Również sposób konfiguracji aplikacji oraz komunikacji wewnątrz niej, z którymi się spotkałem po raz pierwszy były dla mnie sporymi wyzwaniami na początku implementacji. Konieczność umieszczania deklaracji użycia komponentów w manifeście aplikacji, przysparzała czasem problemów. Parcelowanie obiektów w celu ich przekazywania pomiędzy komponentami również była dla mnie nowym doświadczeniem. Metoda, choć prosta, również powodowała błędy programu, aczkolwiek najczęściej były one powodowane moimi przeoczeniami.

Mogę potwierdzić, że Android ma świetne zaplecze techniczne. Zarówno od strony oficjalnej, przez udostępniane kompletne dokumentacje i przewodniki, które były szczególnie pomocne w początkowych etapach tworzenia programu, jak i wszelkie fora internetowe poświęcone tematyce Androida, z których wiele się dowiedziałem w późniejszych fazach powstawania

aplikacji. Dzięki temu korzystanie z API Wi-Fi P2P było bardzo proste i nie przysporzyło większych problemów.

Ważnym elementem powstania aplikacji jest dokładna analiza możliwości systemu względem wymagań funkcjonalnych aplikacji. Bez tego czynnika może się okazać, że wiele pomysłów podczas implementacji nie uda się zrealizować. Tak, jak w moim przypadku, dzięki analizie mogłem przed rozpoczęciem implementacji podjąć decyzję o zmianie metody komunikacji między urządzeniami.

Dzięki swoim obserwacjom mogę wyciągnąć wniosek, że rynek aplikacji mobilnych jest już dosyć pokaźny i ciągle ekspansywnie się rozwija. Mając do zaoferowania zarówno deweloperem jak i użytkownikom coraz więcej oraz coraz lepszej jakości usług.

Reasumując, tworzenie aplikacji mobilnych nie jest trudne, aczkolwiek wymaga pewnego doświadczenia w tej dziedzinie. Na pewno mogę stwierdzić, że stworzenie następnych aplikacji mobilnych będzie już dużo łatwiejsze, dzięki zdobytej wiedzy i umiejętnościom.

# 7 Dodatek. Fragmenty kodu źródłowego

*Rysunek 12 Listing parcelowania klasy Offer*

```
public class Offer implements Parcelable {

    private String title;
    private String description;
    private Double price;
    private Double param1;
    private String param2;
    private Category category;
    private String codedName;
    private boolean active;

    public Offer(Parcel in) {
        title = in.readString();
        description = in.readString();
        price = in.readDouble();
        param1 = in.readDouble();
        param2 = in.readString();
        category = in.readParcelable(Category.class.getClassLoader());
        codedName = in.readString();
        active = (Boolean) in.readValue(boolean.class.getClassLoader());
    }

    public static final Parcelable.Creator<Offer> CREATOR =
            new Parcelable.Creator<Offer>() {
        public Offer createFromParcel(Parcel in) {
            return new Offer(in);
        }

        public Offer[] newArray(int size) {
            return new Offer[size];
        }
    };

    @Override
    public int describeContents() {
        return 0;
    }

    @Override
    public void writeToParcel(Parcel dest, int flags) {
        dest.writeString(title);
        dest.writeString(description);
        dest.writeDouble(price);
        dest.writeDouble(param1);
        dest.writeString(param2);
        dest.writeParcelable(category, flags);
        dest.writeString(codedName);
        dest.writeValue(active);
    }

}
```

*Rysunek 13 Tworzenie aktywności i odebranie zamierzenia*

```
[
    Intent intent = new Intent(this, OffersManager.class);¶
    ¶
    Bundle bundle = new Bundle();¶
    bundle.putParcelable("offers", offers);¶
[
    intent.putExtras(bundle);¶
    startActivityForResult(intent, 0);¶
}¶
[
```

*Rysunek 14 Uruchomienie aktywności z zamierzeniem*

```
@Override¶
public void onActivityResult(int requestCode, int resultCode, Intent data) {¶
    super.onActivityResult(requestCode, resultCode, data);¶
    if (requestCode == 0) {¶
        if (resultCode == RESULT_OK) {¶
            Bundle bundle = data.getExtras();¶
            offers = (Offers) bundle.getParcelable("offers");¶
            intentService = new Intent(getBaseContext(), MyService.class);¶
            Bundle sendBundle = new Bundle();¶
            sendBundle.putParcelable("offers", offers);¶
¶
            intentService.putExtras(sendBundle);¶
            intentService.putExtra("MESSENGER", new Messenger(messageHandler));¶
            startService(intentService);¶
            if (!offers.getSales().isEmpty()) {¶
                for(Sale sale : offers.getSales()) {¶
                    if(sale.isActive()) {¶
                        String name = sale.getCodedName();¶
                        setName(name);¶
                        break;¶
                    }¶
                }¶
            }¶
        }¶
        if (resultCode == RESULT_CANCELED) {¶
            return;¶
        }¶
```

*Rysunek 15 Definicja metody setDeviceName*

```
/**¶
 * Set p2p device name.¶
 * @hide¶
 * @param c is the channel created at {@link #initialize}¶
 * @param listener for callback when group info is available. Can be null.¶
 */¶
public void setDeviceName(Channel c, String devName, ActionListener listener) {¶
```

*Rysunek 16 Listing kodu korzystającego z refleksji*

```
Class<?> cl = manager.getClass();
Method method = cl.getMethod("setDeviceName");
method.setAccessible(true);
String deviceName = name;
method.invoke(manager, channel, deviceName, null);
```