POZNAN UNIVERSITY OF TECHNOLOGY

Patryk Dąbrowski 100584
Aleksander Kędzierski 98875
Paweł Lampe 99277
Mateusz Sikora 99615

Platforma zarządzania zdarzeniami na urządzeniach mobilnych if{y}

Bachelor's Thesis

Supervisor: dr inż. Jerzy Błaszczyński

Poznań, 2014

Spis treści

Rozdział 1

Wstęp

1.1 Opis problemu i koncepcja jego rozwiązania (motywacja?)

Współczesne urządzenia mobilne dysponują ogromnym zbiorem możliwości. Nie są to już tylko telefony które dawniej służyły wyłącznie do komunikacji. Obecnie rynek pełen jest urządzeń z zakresu; od tabletu do smartfonu. Mnogość typów urządzeń oraz tendencja do upodabniania się między sobą uniemożliwia jednoznaczne stwierdzenie do czego właściwie służą.

Nie lepiej sytuacja ma się w przypadku programistów piszących aplikacje na urządzenia mobilne. Nie dość, że niemal każde urządzenie ma inny osprzęt, to co więcej, na rynku figuruje kilka wiodących mobilnych systemów operacyjnych. Wszystko to wpływa dość drastycznie na charakter rynku aplikacji mobilnych. W większej części są to proste aplikacje realizujące bardzo ściśle określone usługi. Prowadzi to do sytuacji, w której na przeciętnym smartfonie trzeba mieć 10-20 aplikacji które pozwolą osiągnąć stabilny poziom zadowolenia.

Gdyby istniała otwartoźródłowa aplikacja pozwalająca na stosunkowo prosty dostęp do możliwości telefonu oraz ułatwiająca proces programowania, wiele małych aplikacji mogło by stać się prostymi receptami instruującymi aplikację–matkę jak wykonać stosunkowo proste zadania. W tym przypadku 10-20 małych aplikacji można by zamienić na jedną zawierającą w środku 10-20 prostych kawałków kodu którymi można w pełni zarządzać.

Taka aplikacja oczywiście nie istnieje. Istnieją podobne, mniej lub bardziej udane rozwiązania. Wszystkie jednak są komercyjne co jest kluczową wadą. Zamkniętość kodu – bo o tym tutaj mowa, ogranicza zbiór osób zaangażowanych w tworzene i pielengnację kodu. Ma to ogromne implikacje na rozwój aplikacji, co z kolei uderza w jej użytkowników. W przypadku jakiegokolwiek złożonego problemu, użytkownik nie jest w stanie samemu sprawdzić czy problem leży po stronie jego kodu, czy po stronie kodu aplikacji.

Skoro wyżej wspomniana, otwartoźródłowa aplikacja nie istnieje, warto by ją stworzyć. Unicestwiło by to wszystkie wspomniane powyżej problemy. Taka też idea leży u podstaw tej pracy. Stworzyć wolną, otwartoźródłową aplikację służącom przeciętnym użytkownikom. Co jednak nawet ważniejsze, stworzyć kod, który będzie mógł zostać użyty przez zaawansowanych użytkowników-programistów.

1.2 Cele i zakres pracy

Postawowym celem niniejszej pracy, jest stworzenie otwartoźródłowej biblioteki uproszczającej dostęp do podzespołów urządzenia mobilnego. Jest to o tyle ważne, iż tworzy warstwę abstrakcji nad systemem operacyjnym. Dzięki temu, kod który przykładowo przetwarza dane z GPS, pozostaje identyczny dla systemów Android, iOS tudzież Windows Phone. Inna jest tylko implementacja biblioteki dla danej platformy. Niniejsza praca zakłada implementację biblioteki tylko dla systemu Android.

Drugim co do ważności celem, jest stworzenie przykładowej aplikacji prezentującej możliwości biblioteki. Z racji, iż implementacja biblioteki obejmuje tylko system Android, implementacja aplikacji również. Fakt, iż aplikacja jest tylko przykładem, nie oznacza, że większość wykonanej pracy stanowi biblioteka. Wręcz przeciwnie. Lwią część napisanego kodu stanowi aplikacja wraz z używanymi przez nią aplikacjami webowymi. Aplikacja bowiem, jako, że jest środowiskiem uruchomieniowym dla krótkich kawałków kodu – recept, potrzebuje zdalnego repozytorium bedącego niczym innym jak stroną internetową. Potrzebuje również serwera utrzymującego informacje dla recept które korzystają z komunikacji w obrębie grup użytkowników.

Reasumując, kod który musi powstać, to biblioteka, aplikacja, repozytorium recept oraz serwer dla recept grupowych.

1.3 Omówienie pracy

Nixx nett hier

Rozdział 2

Wymagania

2.1 Wymagania funkcjonalne

2.1.1 Przypadki użycia platformy

UC1 - Tworzenie Recepty 1. Użytkownik ma pomysł na Receptę. 2. Użytkownik loguje się do Targowiska. 3. Użytkownik tworzy nową Receptę. 4a. Użytkownik pisze kod w edytorze online. 4b. Użytkownik pisze kod lokalnie (np. w Eclipse) i wkleja kod do edytora. 5. Serwer kompiluje receptę. 6. Użytkownik pobiera receptę na telefon. 7. Recepta działa na telefonie.

2.1.2 Przypadki użycia Aplikacji - przykładowe Recepty

2.2 Wymagania pozafunkcjonalne

Rozdział 3

Zarządzanie zdarzeniami na urządzeniach mobilnych

3.1 Definicja pojęć

- Podfunkcjonalność (ang. Feature) – Część biblioteki zapewniająca Receptom dostęp do pozdbioru funkcjonalności Androida.
- Zdarzenie (ang. Event) – Zmiana stanu systemu, która powoduje uruchomienie kodu Recepty.
- Recepta (ang. Recipe) – Napisany przez użytkownika fragment kodu opisujący, co ma się zdarzyć po spełnieniu pewnych warunków.
- Targowisko (ang. Market) – Aplikacja internetowa pozwalająca tworzyć i pobierać Recepty.
- Aplikacja – Aplikacja androidowa wykorzystująca bibliotekę if{Y}.
- Serwer Grup – Komputer z działającą aplikacją, która zarządza grupami użytkowników i Zdarzeniami Grupowymi.
- Zdarzenie Grupowe – Zdarzenie związane z Grupą, wysyłane lub odbierane przez Aplikację z Serwera Grup.
- Grupa – Zbiór użytkowników identyfikowalny przez nazwę zdefiniowany na Serwerze Grup.

3.2 Istniejące rozwiązania

3.2.1 On X

Aplikacja Microsoftu umożliwiającą kontrolowanie telefonu z Androidem używając kodu w JavaScripcie. Umożliwia wysyłanie Zasad (Rules) na telefon poprzez stronę internetową. Dostęp do funckcjonalości Androida jest zapewniony przez api w postaci Wyzwalaczy (Triggers) i Akcji (Actions). Cały system jest niestety połączony z Facebookiem i wymaga posiadania tam konta. Na podstawie [1].

3.2.2 Tasker

Rozdział 4

Architektura platformy

System składa się z biblioteki, przykładowej aplikacji appIFY oraz aplikacji działających na serwerze - Serwera Grup oraz Targowiska. Aplikacja korzysta z biblioteki oraz komunikuje się z serwerem. Oprócz tego Serwer Grup oraz Targowisko udostępniają z poziomu przeglądarki takie funkcje jak rejestracja użytkowników czy tworzenie Recept. Kluczowym założeniem było maksymalne uproszczenie kodu recept.

Kod Aplikacji jest podzielony na dwie części:

- bibliotekę IFY
- aplikację appIFY

Celem takiego podziału jest ułatwienie tworzenia innych aplikacji opartych o bibliotekę.

4.1 Recepty

Miejscem, gdzie zawarta jest główna logika Aplikacji są Recepty – są w nich opisane wszystkie zdarzenia, które mają nastąpić po spełnieniu pewnych ściśle określonych warunków. Docelowo będą one tworzone przez użytkowników i pobierane z Targowiska, jednak istnieją także przykładowe Recepty wbudowane w Aplikację, mające na celu ułatwienie użytkownikom tworzenia nowych na ich podstawie oraz rozszerzenie początkowej funkcjonalności aplikacji. Na receptę składają się:

- opis używanych podfunkjonalności
- opis wymaganych parametrów
- opis jej właściwego działania

Deklarowanie używanych podfunkcjonalności ma dwa główne cele - po pierwsze, użytkownik widzi, czego używa recepta, co nieco poprawia jego bezpieczeństwo przy używaniu recept innych użytkowników, po drugie pozwala to inicjalizować nasłuchiwanie zdarzeń systemowych tylko wtedy, gdy istnieje aktywna recepta, która na nie reaguje - kod recepty nie musi inicjalizować większości podfunkcjonalności, wystarczy deklaracja ich używania. Wyjątkiem jest podfunkcjonalność grup, gdzie komunikację należy zainicjalizować.

Parametry pozwalają użytkownikowi na dostosowanie recepty do swoich wymagań, bez potrzeby pisania nowej. W naszych przykładowych receptach były to np. numer telefonu do wysłania SMS lub jego tekst czy też zasięg znajdowania znajomych na podstawie GPS.

Właściwa logika recepty jest zawarta w funkcji reakcji na zdarzenie. Uznaliśmy taki sposób za prostszy niż na przykład ciągłe działanie recepty w osobnych wątku i odpytywanie podfunkcjonalności o zdarzenia w kodzie recepty. Recepta staje się jednak obserwatorem automatycznie na podstawie zadeklarowanych podfunkcjonalności, a wszytkie zdarzenia wywołują tą samą metodę w Recepcie. Takie rozwiązanie minimalizuje ilość kodu w recepcie, co było kluczowym celem. Recepty mogą też zażądać pewnych danych od systemu, które są dostarczane asynchronicznie - na przykład przetłumaczenie danych z GPS na adres (Geocoder). Wyniki tego typu operacji również są przekazywane do recepty jako zunifikowane zdarzenia.

4.2 Biblioteka

Biblioteka zawiera głównie API dostępne z poziomu recept, czyli między innymi Podfunkcjonalności, które agregują i upraszczają dostęp do metod z API systemu Android. Oprócz tego znajduje się tam moduł odpowiedzialny za zarządzanie cyklem życia Recept i Podfunkcjonalności, który działa cały czas w tle. Podfunkcjonalności są inicjalizowane przez serwis przy uruchamianiu recepty, która deklaruje ich użycie. Zapewniają one dostęp do określonych funkcji, takich jak odczyt danych z sensorów, odbieranie i wysyłanie SMS'ów i wiele innych.

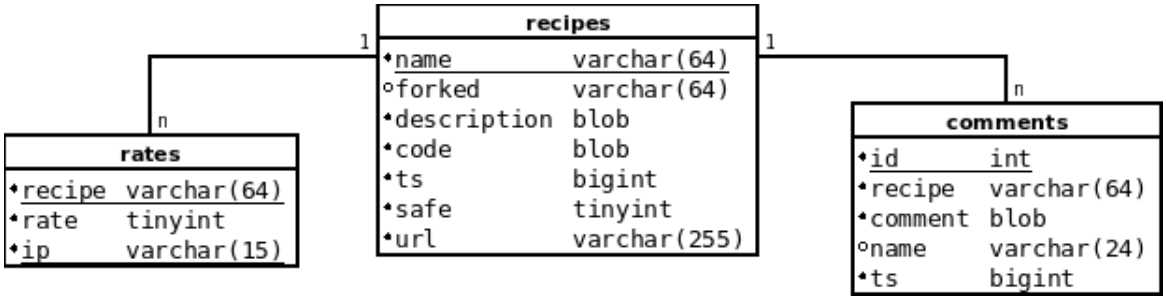
4.3 Aplikacja kliencka

Aplikacja to głownie interfejs użytkownika – ekrany takie jak wyświetlanie listy dostępnych lub aktywnych recept, ustalanie ich parametrów i ich włączanie i wyłączanie. Dodatkowo aplikacja jest zintegrowana z Targowiskiem umożliwiając pobieranie z niego recept. Umożliwia też logowanie się do Serwera Grup.

4.4 Targowisko

1. geneza 2. rozwiązanie problemu 3. forkowanie 4. schmat bazy + 5. edytor online (artykuł?)

4.5 Serwer



Rysunek 4.1: Schemat bazy danych targowiska

Rozdział 5

Opis implementacji

5.1 Recepty

Recepty dziedziczą po klasie abstrakcyjnej YRecipe i implementują jej abstrakcyjne metody. Obrazuje to poniższy przykład recepty, która odrzuca wszystkie nadchodzące połączenia.

```
public class YSampleCallsSMS extends YRecipe {

@Override
public long requestFeatures() {
return Y.Calls;
}

@Override
public void requestParams(YParamList params) {
//Message to send in SMS
params.add("MSG",YParamType.String, "Sorry, I'm busy.");
}

@Override
public void handleEvent(YEvent event) {
//event is incoming call
if(event.getId() == Y.Calls){
YCallsEvent e = (YCallsEvent) event;
//extract phone number
String phone = e.getIncomingNumber();
//discard call
mFeatures.getCalls().discardCurrentCall();
//send sms
mFeatures.getSMS().sendSMS(phone, mParams.getString("MSG"));
}
}
```

```
@Override
public String getName() {
return "YSampleCallsSMS";
}

@Override
public YRecipe newInstance() {
return new YSampleCallsSMS();
}
}
```

5.2 Biblioteka

5.3 Aplikacja kliencka

5.3.1 Moduł obsługi recept

5.3.2 Moduły dostępu do systemu

5.4 Targowisko

5.5 Serwer

5.5.1 Repozytorium recept

5.5.2 Serwer recept grupowych

5.6 Protokół komunikacji

Komunikacja aplikacji klienckich oparta jest o ciagłe odpytywanie (ang. polling). Wymiana danych odbywa się przy użyciu tekstowego formatu danych JSON.

5.7 Użyte technologie

W tej części zaprezentowano opis technologii użytych bezpośrednio w implementacji składowych platformy.

5.7.1 Android

System operacyjny z rodziny Linux przeznaczony dla urządzeń mobilnych. Aktualnie rozwijane przez sojusz biznesowy Open Handset Alliance.

5.7.2 Android SDK

Platforma programistyczna umożliwiająca tworzenie aplikacji dla systemu Android. Zawiera wtyczkę do środowiska Eclipse, narzędzia wspierające prace programisty, emulator i biblioteki potrzebne do zbudowania aplikacji. Programy dedykowne platformie pisane są w języku Java i uruchamiane na maszynie wirtualnej Dalvik.

5.7.3 Apache Commons

5.7.4 Apache HTTP Server

Otwartoźródłowy serwer HTTP. Najpopularniejsze narzędzie tego typu na świecie. Jego wielką zaletą jest mnogość informacji na jego temat dostępnych w internecie oraz dostępność na większość znaczących systemów operacyjnych.

5.7.5 Git

Rozproszony oraz wieloplatformowy system kontroli wersji będący wolnym oprogramowaniem. Preferowane narzędzie programistów związanych z otwartym oprogramowaniem.

5.7.6 HTML 5

Język programowania służący do tworzenia współczesnych stron internetowych. Jest rozwinięciem oraz uproszczeniem języka HTML 4.

5.7.7 Hibernate

Narzędzie odwzorowań obiektowo-relacyjnych (ang. object-relation mapping, ORM) rozwijany na zasadzie wolnego oprogramowania. Umożliwia odworowania obiektowo-relacyjne, pamięć podręczną, leniwe (ang. Lazy loading), chciwe pobieranie oraz rozproszoną pamięć podręczną.

5.7.8 JSON

Skrót od JavaScript Object Notation. Jest to lekki, tekstowy format wymiany danych niezależny od języka programowania. Został wybrany ze względu na swoją czytelność i wsparcie ze strony bibliotek programistyzcnych.

5.7.9 Java 6

Jezyk programowania cechujący się obiektowością (ang. Object-oriented programming, OOP) oraz silmnym typowaniem. Kod źródłowy Javy kompilowany jest do kodu bajtowego interpretowanego przez maszynę wirtualną zapewnia to większa niezależność od platformy niż w innych podobnych językach np. C++.

5.7.10 JavaScript

Skryptowy język oprogramowania stosowany na stronach internetowych.

5.7.11 Apache Maven

Narzędzie automatycznego budowania oprogramowania dla języka JAVA. Głównymi problemami jakie rozwiązuje Maven przy budowaniu aplikacji są: zarządzanie zależnościami, mozliwość wieloma modułami, wsparcie dla testów.

5.7.12 MySQL

System zarządzania relacyjnymi bazami danych. Jest to wolne oprogramowanie szczególnie upodobane przez twórców aplikacji internetowych. Bardzo dobrze współpracuje z językami takimi jak PHP czy Java

5.7.13 PHP

Obiektowy język programowania dedykowany generowaniu stron internetowych w czasie rzeczywistym. Szczególnie użyteczny w przypadku tworzenia prototypów tudzież niewielkich projektów wymagających stosunkowo niskiego poziomu abstrakcji.

5.7.14 RESTeasy

Framework oprogramowania służacy do tworzenia aplikacji rozproszonych, oparty na wzorcu architektury oprogramowania Representational State Transfer(REST).

5.7.15 SpringFramework

Framework(Szkielet) tworzenia aplikacji w języku Java a w szczególności JavaEE. Do najważniejszych fukcji Springa zalicza się wstrzykiwanie zależności (ang. dependency injection, DI) oraz programowanie aspektowe (ang. aspect-oriented programming, AOP).

5.7.16 Vaadin

Framework sieciowy służący do tworzenia aplikacji sieciowych w szczególnosci interfejsu użytkownika w oparciu o Google Web Toolkit (GWT) w języku JAVA.

5.7.17 JUnit

Biblioteka służaca do tworzenia testów jednstkowych w jezyku Java.

5.8 Użyte narzędzia

5.8.1 Apache Tomcat

Kontener aplikacji sieciowych.

5.8.2 Eclipse

Popularne zintegrowane środowisko programistyczne (IDE) wspierające głównie język Java (wtyczki pozwalają obsługiwać inne języki).

5.8.3 Android developer tools

Wtyczka do Eclipse pozwalająca tworzyć aplikacje androidowe. Dodaje takie funkcjonalności jak edycja plików XML odpowiadających za wygląd aplikacji (również w trybie graficznym) czy debugowanie na telefonach oraz emulatorze.

5.8.4 String Tool Suite

Zintegrowane środowisko programistyczne oparte o Eclipsa dostosowany do SpringFramework.

5.8.5 Emacs

Popularny, w pełni rozszerzalny edytor tekstowy spotykany głównie w systemach operacyjnych z rodziny Unix. Używany przez wysokiej klasy programistów oraz naukowców na całym świecie.

5.8.6 Git bash for windows

Narzędzie umożliwiające używanie Gita z linii poleceń w systemie Windows poprzez wbudowane środowisko MinGW.

5.8.7 Github

Serwis internetowy gromadzący społeczność programistów z całego świata. Służy jako hosting dla otwartoźródłowych projektów zarządzanych za pomocą systemu Git. Udostępnia szereg narzędzi wspierających - system śledzenia zadań, budowa statystyk.

5.8.8 Latex

5.8.9 Linux

Rodzina systemów operacyjnych będących wolnym oprogramowaniem oraz używajnących jądra Linux.

5.8.10 Notepad++

Prosty edytor tekstowy umożliwiający kolorowanie składni w wielu językach.

5.8.11 Przeglądarki internetowe

Programy takie jak Google Chrome, Mozilla Firefox czy Opera, używane w pracy do testowania rozwiązań mających postać strony internetowej.

5.8.12 Windows

System operacyjny firmy Microsoft.

5.9 Użyty sprzęt

5.9.1 Komputery klasy PC

Podstawowa platforma do wszystkich aspektów pracy, z wyjątkiem testowania, do którego użyliśmy także telefonów.

5.9.2 LG Swift GT540

Procesor: Qualcomm MSM7227 600 MHz Pamięć RAM: 256 MB System operacyjny: Android 4.0.1 (Cyanogen mod)

5.9.3 Media-Droid IMPERIUS EN3RGY MT7013

Procesor: dwurdzeniowy, 1GHz ARM7 MTK6577 Pamięć RAM: 256 MB System operacyjny: Android 4.1.2

5.9.4 Motorola Defy MB525

Procesor: TI OMAP3610 800 MHz Pamięć RAM: 512 MB System operacyjny: Android 4.3.1 (Cyanogen mod)

5.9.5 Sony LT18 Xperia Arc S

Procesor: Qualcomm MSM8255T 1,40 GHz Pamięć RAM: 512 MB System operacyjny: Android 4.0.4

5.9.6 Samsung Galaxy Mini GT-S5570

Procesor: Qualcomm MSM7227 600 MHz Pamięć RAM: 384 MB System operacyjny: Android 2.2

5.10 Opis pakietów

5.10.1 Pakiety Aplikacji

pl.poznan.put.cs.ify.app - główny pakiet Aplikacji. pl.poznan.put.cs.ify.jars - pakiet odpowiedzialny za zarządzanie plikami .jar zawierającymii recepty pobrane z Targowiska. pl.poznan.put.cs.ify.core - pakiet odpowiedzialny za zarządzanie dostępnymi i aktywowanymi Receptami. pl.poznan.put.cs.ify.appify.receipts - pakiet zawierający Recepty wbudowane w Aplikację. pl.poznan.put.cs.ify.app.ui - pakiet zawierający kontrolki interfejsu użytkownik. pl.poznan.put.cs.ify.app.ui.params - pakiet zawierający kontrolki interfejsu użytkownika wykorzystywane do wprowadzania parametrów przy inicjalizacji Recepty. pl.poznan.put.cs.ify.app.market - pakiet odpowiedzialny za pobieranie danych z Targowiska i wyświetlanie ich. pl.poznan.put.cs.ify.app.fragments - pakiet zawierający widoki ekranów aplikacji.

5.10.2 Pakiety Biblioteki

pl.poznan.put.cs.ify.api - pakiet główny Biblioteki. pl.poznan.put.cs.ify.api.exceptions - pakiet zawierający wyjątki, które mogą być rzucane przez metody z Biblioteki. pl.poznan.put.cs.ify.api.fea - pakiet zawietający Podfunkcjonalności i Zdarzenia. pl.poznan.put.cs.ify.api.group - pakiet odpowiedzialny za obsługę Recept Grupowych. pl.poznan.put.cs.ify.api.log - pakiet odpowiedzialny za obsługę logowania i domyślny widok logów. pl.poznan.put.cs.ify.api.params - pakiet zawierający typy parametrów wykorzystywanych przez Recepty. pl.poznan.put.cs.ify.api.security - pakiet odpowiedzialny za moduł uprawnień Biblioteki. pl.poznan.put.cs.ify.api.types - pakiet zawierający typy danych wykorzystywanych przez Biblioteke.

5.10.3 Pakiety Serwera

pl.poznan.put.cs.ify.webify - pakiet główny serwera. pl.poznan.put.cs.ify.webify.data.dao - pakiet zawierający warstwe dostępu do danych. pl.poznan.put.cs.ify.webify.data.entity - pakiet zawierający klasy odwzorowywane na bazę danych. pl.poznan.put.cs.ify.webify.data.enums - pakiet zawierajacy potrzebne w bazie danych typy wyliczeniowe(np. lista ról). pl.poznan.put.cs.ify.webify - pakiet główny graficznego interfejsu użytkownika. pl.poznan.put.cs.ify.webify.gui.windows - paiet zawierający wszytskie okna aplikacji sieciowej. pl.poznan.put.cs.ify.webify.gui.components - pakiet zawierający komponenty użyte w aplikacji. pl.poznan.put.cs.ify.webify.gui.session - pl.poznan.put.cs.ify.webify.service - pakiet zawierający logikę. pl.poznan.put.cs.ify.webify.rest - pakiet zawerajacy obsługę zapytań typu REST. pl.poznan.put.cs.ify.webify.utils - pakiet, w którym przechowywane są funkcje pomocnicze używane w całym projkcie.

Rozdział 6

Testy oraz wyniki?

Rozdział 7

Zakończenie

Dodatek A

Przewodnik użytkownika

A.1 Opis Podfunkcjonalności

A.1.1 Akcelerometr (YAccelerometerFeature.java)

Umożliwia reagowanie na odczyty akcelerometru wbudowanego w urządzenie.

A.1.2 Battery (YBatteryFeature.java)

Umożliwia reagowanie na zmiany poziomu baterii urządzenia.

A.1.3 SMS (YSMSFeature.java)

Umożliwia wysyłanie wiadomości SMS oraz reagowanie na wiadomości przychodzące.

A.1.4 Wifi (YWifiFeature.java)

Umożliwia włączanie i wyłączanie modułu WiFi urządzenia.

A.1.5 GPS (YGPSFeature.java)

Umożliwa śledzenie pozycji urządzenia za pomocą modułu GPS.

A.1.6 Sound (YSoundFeature.java)

Pozwala odtrzarzać pliki dźwiękowe.

A.1.7 RawPlayer (YRawPlayerFeature.java)

A.1.8 Group (YGroupFeature.java)

A.1.9 Geocoder (YGeocoderFeature.java)

Umożliwia pobranie adresu związanego z podaną długościa i szerokością geograficzną.

A.1.10 Time (YTimeFeature.java)

A.1.11 AudioManager (YAudioManager.java)

A.1.12 Text (YTextFeature.java)

A.1.13 Internet (YInternetFeature.java)

Umożliwia wysyłanie i pobieranie danych z podanego adresu.

A.1.14 Calls (YCallsFeature.java)

Umożliwia reagowanie na połączenia przychodzące i inicjowanie połączeń wychodzących.

A.1.15 Notification (YNotificationFeature.java)

Umożliwia wyświetlanie powiadomień w interfejsie graficznym urządzenia.

Bibliografia

[1] Projekt on{X} http://www.onx.ms/#!findOutMorePage. Ostatnio odwiedzone 6/02/13.

[2] C. Walls. *Spring in action, 3rd edition*. Manning Publication Co, 2011.

[3] Vaadin https://vaadin.com/book/vaadin6/-/page/preface.html

[4] E. Gamma. *Design Patterns, First edition*. Person Education, Inc, 1995.