



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

| | | |
|----------|---|-----------|
| 1 | Wstęp | 5 |
| 1.1 | Opis problemu i koncepcja jego rozwiązania (motywacja?) | 5 |
| 1.2 | Cele i zakres pracy | 6 |
| 1.3 | Omówienie pracy | 6 |
| 2 | Wymagania | 7 |
| 2.1 | Wymagania funkcjonalne | 7 |
| 2.1.1 | Przypadki użycia platformy | 7 |
| 2.1.2 | Przypadki użycia Aplikacji - przykładowe Recepty | 7 |
| 2.2 | Wymagania pozafunkcjonalne | 7 |
| 3 | Zarządzanie zdarzeniami na urządzeniach mobilnych | 9 |
| 3.1 | Definicja pojęć | 9 |
| 3.2 | Istniejące rozwiązania. | 9 |
| 3.2.1 | On X | 9 |
| 3.2.2 | Tasker | 10 |
| 4 | Architektura platformy | 11 |
| 4.1 | Recepty | 11 |
| 4.2 | Biblioteka | 11 |
| 4.3 | Aplikacja kliencka | 11 |
| 4.4 | Targowisko | 11 |
| 4.5 | Serwer. | 11 |
| 4.6 | Moduły systemu - TODElete | 11 |
| 5 | Opis implementacji | 13 |
| 5.1 | Użyte technologie | 13 |
| 5.1.1 | Android | 13 |
| 5.1.2 | Android SDK | 13 |
| 5.1.3 | Apache Commons | 13 |
| 5.1.4 | Apache HTTP Server | 13 |
| 5.1.5 | Git | 13 |
| 5.1.6 | HTML 5 | 13 |
| 5.1.7 | Hibernate | 14 |

| | | |
|--------|---|----|
| 5.1.8 | JSON | 14 |
| 5.1.9 | Java 6 | 14 |
| 5.1.10 | JavaScript | 14 |
| 5.1.11 | Apache Maven | 14 |
| 5.1.12 | MySQL | 14 |
| 5.1.13 | PHP | 14 |
| 5.1.14 | RESTeasy | 14 |
| 5.1.15 | SpringFramework | 15 |
| 5.1.16 | Vaadin. | 15 |
| 5.1.17 | JUnit | 15 |
| 5.2 | Użyte narzędzia | 15 |
| 5.2.1 | Apache Tomcat | 15 |
| 5.2.2 | Eclipse. | 15 |
| 5.2.3 | Android developer tools | 15 |
| 5.2.4 | String Tool Suite. | 15 |
| 5.2.5 | Emacs | 15 |
| 5.2.6 | Git bash for windows | 15 |
| 5.2.7 | Github. | 16 |
| 5.2.8 | Latex | 16 |
| 5.2.9 | Linux | 16 |
| 5.2.10 | Notepad++. | 16 |
| 5.2.11 | Przeglądarki internetowe | 16 |
| 5.2.12 | Windows. | 16 |
| 5.3 | Użyty sprzęt | 16 |
| 5.3.1 | Komputery klasy PC | 16 |
| 5.3.2 | LG Swift GT540. | 16 |
| 5.3.3 | Media-Droid IMPERIUS EN3RGY MT7013. | 16 |
| 5.3.4 | Motorola Defy MB525 | 16 |
| 5.3.5 | Sony LT18 Xperia Arc S | 17 |
| 5.3.6 | Samsung Galaxy Mini GT-S5570 | 17 |
| 5.4 | Recepty | 17 |
| 5.5 | Biblioteka | 17 |
| 5.6 | Aplikacja kliencka | 17 |
| 5.6.1 | Moduł obsługi recept | 17 |
| 5.6.2 | Moduły dostępu do systemu | 17 |
| 5.7 | Targowisko | 17 |
| 5.8 | Serwer. | 17 |
| 5.8.1 | Repozytorium recept | 17 |
| 5.8.2 | Serwer recept grupowych | 17 |
| 5.9 | Protokół komunikacji. | 17 |
| 5.10 | Opis pakietów. | 17 |
| 5.10.1 | Pakiety Aplikacji | 17 |
| 5.10.2 | Pakiety Biblioteki | 18 |
| 5.10.3 | Pakiety Serwera | 18 |

| | | |
|----------|---|-----------|
| 6 | Testy oraz wyniki? | 19 |
| 7 | Zakończenie | 21 |
| A | Przewodnik użytkownika | 23 |
| A.1 | Opis Podfunkcjonalności | 23 |
| A.1.1 | Akcelerometr (YAccelerometerFeature.java) | 23 |
| A.1.2 | Battery (YBatteryFeature.java) | 23 |
| A.1.3 | SMS (YSMSFeature.java) | 23 |
| A.1.4 | Wifi (YWifiFeature.java) | 23 |
| A.1.5 | GPS (YGPSFeature.java) | 23 |
| A.1.6 | Sound (YSoundFeature.java) | 23 |
| A.1.7 | RawPlayer (YRawPlayerFeature.java) | 23 |
| A.1.8 | Group (YGroupFeature.java) | 23 |
| A.1.9 | Geocoder (YGeocoderFeature.java) | 23 |
| A.1.10 | Time (YTimeFeature.java) | 24 |
| A.1.11 | AudioManager (YAudioManager.java) | 24 |
| A.1.12 | Text (YTextFeature.java) | 24 |
| A.1.13 | Internet (YInternetFeature.java) | 24 |
| A.1.14 | Calls (YCallsFeature.java) | 24 |
| A.1.15 | Notification (YNotificationFeature.java) | 24 |
| | Bibliografia | 25 |

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 tworzenie i pielęgnowanie 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. Musi w związku z tym zgłosić się do działu obsługi klienta i mieć nadzieję, że w skończonym czasie jakiś programista należący do zbioru wtajemniczonych przyjrzy się tej sprawie bliżej.

Skoro wyżej wspomniana, otwartoźródłowa aplikacja nie istnieje, warto by ją stworzyć. Unieściłoby to wszystkie wspomniane powyżej problemy. Taka też idea leży u podstaw tej pracy. Stworzyć wolną, otwartoźródłową aplikację służącą 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 pokazującej 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 będącego 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

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

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 JavaScriptcie. Umożliwia wysyłanie Zasad (Rules) na telefon poprzez stronę internetową. Dostęp do funkcjonalności 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

«««< HEAD

Architektura platformy

4.1 Recepty

4.2 Biblioteka

4.3 Aplikacja kliencka

Kod Aplikacji jest podzielony na dwie części:

- bibliotekę IFY
- aplikację appIFY

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

Kluczowym założeniem było maksymalne uproszczenie kodu recept.

4.4 Targowisko

4.5 Serwer

4.6 Moduły systemu - TODElete

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. 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.

Opis implementacji

5.1 Użyte technologie

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

5.1.1 Android

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

5.1.2 Android SDK

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

5.1.3 Apache Commons

5.1.4 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.

5.1.5 Git

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

5.1.6 HTML 5

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

5.1.7 Hibernate

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

5.1.8 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 programistycznych.

5.1.9 Java 6

Język programowania cechujący się obiektowością (ang. Object-oriented programming, OOP) oraz silnym 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++.

5.1.10 JavaScript

Skryptowy język oprogramowania stosowany na stronach internetowych.

5.1.11 Apache Maven

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

5.1.12 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

5.1.13 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.

5.1.14 RESTeasy

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

5.1.15 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).

5.1.16 Vaadin

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

5.1.17 JUnit

Biblioteka służąca do tworzenia testów jednostkowych w języku Java.

5.2 Użyte narzędzia

5.2.1 Apache Tomcat

Kontener aplikacji sieciowych.

5.2.2 Eclipse

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

5.2.3 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.

5.2.4 String Tool Suite

Zintegrowane środowisko programistyczne oparte o Eclipse dostosowany do SpringFramework.

5.2.5 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.

5.2.6 Git bash for windows

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

5.2.7 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.

5.2.8 Latex

5.2.9 Linux

Rodzina systemów operacyjnych będących wolnym oprogramowaniem oraz używających jądra Linux.

5.2.10 Notepad++

Prosty edytor tekstowy umożliwiający kolorowanie składni w wielu językach.

5.2.11 Przeglądarki internetowe

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

5.2.12 Windows

System operacyjny firmy Microsoft.

5.3 Użyty sprzęt

5.3.1 Komputery klasy PC

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

5.3.2 LG Swift GT540

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

5.3.3 Media-Droid IMPERIUS EN3RGY MT7013

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

5.3.4 Motorola Defy MB525

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

5.3.5 Sony LT18 Xperia Arc S

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

5.3.6 Samsung Galaxy Mini GT-S5570

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

5.4 Recepty

5.5 Biblioteka

5.6 Aplikacja kliencka

5.6.1 Moduł obsługi recept

5.6.2 Moduły dostępu do systemu

5.7 Targowisko

5.8 Serwer

5.8.1 Repozytorium recept

5.8.2 Serwer recept grupowych

5.9 Protokół komunikacji

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

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 zawierającymi 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 użytkownika. 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.features` - pakiet zawierający 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 Bibliotekę.

5.10.3 Pakiety Serwera

`pl.poznan.put.cs.ify.webify` - pakiet główny serwera. `pl.poznan.put.cs.ify.webify.data.dao` - pakiet zawierający warstwy 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 zawierający potrzebne w bazie danych typy wyliczeniowe (np. lista ról). `pl.poznan.put.cs.ify.webify.gui` - pakiet główny graficznego interfejsu użytkownika. `pl.poznan.put.cs.ify.webify.gui.windows` - pakiet zawierający wszystkie 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 zawierający obsługę zapytań typu REST. `pl.poznan.put.cs.ify.webify.utils` - pakiet, w którym przechowywane są funkcje pomocnicze używane w całym projekcie.

Testy oraz wyniki?

Zakończenie

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)

Umożliwia śledzenie pozycji urządzenia za pomocą modułu GPS.

A.1.6 Sound (YSoundFeature.java)

Pozwala odtwarzać 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 długością 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.