



**A G H**

**AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE**

**WYDZIAŁ ELEKTROTECHNIKI, AUTOMATYKI,  
INFORMATYKI I INŻYNIERII BIOMEDYCZNEJ**

KATEDRA INFORMATYKI STOSOWANEJ

Praca dyplomowa magisterska

*Metody mediacji wiedzy w systemach affective computing  
Methods for knowledge mediation in affective computing systems*

Autor: *inż. Filip Biernat*  
Kierunek studiów: *Informatyka*  
Opiekun pracy: *dr inż. Szymon Bobek*

Kraków, 2019

*Uprzedzony o odpowiedzialności karnej na podstawie art. 115 ust. 1 i 2 ustawy z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (t.j. Dz.U. z 2006 r. Nr 90, poz. 631 z późn. zm.): „Kto przywłaszcza sobie autorstwo albo wprowadza w błąd co do autorstwa całości lub części cudzego utworu albo artystycznego wykonania, podlega grzywnie, karze ograniczenia wolności albo pozbawienia wolności do lat 3. Tej samej karze podlega, kto rozpozna bez podania nazwiska lub pseudonimu twórcy cudzy utwór w wersji oryginalnej albo w postaci opracowania, artystycznego wykonania albo publicznie zniekształca taki utwór, artystyczne wykonanie, fonogram, videogram lub nadanie.”, a także uprzedzony o odpowiedzialności dyscyplinarnej na podstawie art. 211 ust. 1 ustawy z dnia 27 lipca 2005 r. Prawo o szkolnictwie wyższym (t.j. Dz. U. z 2012 r. poz. 572, z późn. zm.): „Za naruszenie przepisów obowiązujących w uczelni oraz za czyny uchybiające godności studenta student ponosi odpowiedzialność dyscyplinarną przed komisją dyscyplinarną albo przed sądem koleżeńskim samorządu studenckiego, zwanym dalej «sądem koleżeńskim».”, oświadczam, że niniejszą pracę dyplomową wykonałem(-am) osobiście i samodzielnie i że nie korzystałem(-am) ze źródeł innych niż wymienione w pracy.*





# **Spis treści**

<b>1. Wprowadzenie .....</b>	7
1.1. Cel pracy .....	7
1.2. Zastosowania wykrywania emocji .....	7
1.3. Gromadzenie danych .....	7
<b>2. Mechanizmy mediacji wiedzy .....</b>	9
2.1. Obszary zastosowań .....	9
2.2. Jawne metody mediacji wiedzy .....	9
2.3. Niewjawne metody mediacji wiedzy .....	10
2.4. Pozyskiwanie wiedzy o stanie emocjonalnym w systemach <i>affective computing</i> .....	10
<b>3. Architektura rozwiązania.....</b>	13
3.1. Wprowadzenie do architektury .....	13
3.2. Silnik wnioskujący .....	14
3.3. Model HMR .....	15
3.4. Framework AWARE .....	15
3.5. Pakiet <i>HeaRT-AWARE</i> .....	15
3.6. Pakiet <i>HowAreYou</i> .....	16
3.6.1. Plugin AWARE .....	16
3.6.2. Dialog z użytkownikiem .....	17
3.6.3. Pakiet <i>photo</i> .....	18
3.6.4. Menu ustawień .....	19
<b>4. Implementacja .....</b>	23
4.1. Wprowadzenie do implementacji .....	23
4.2. Model HMR .....	23
4.2.1. Konstruowanie modelu wnioskującego .....	23
4.2.2. Zaawansowany model wnioskujący .....	24
4.2.3. Uproszczony model wnioskujący .....	29
4.3. Pakiet <i>HeaRT-AWARE</i> .....	30

4.3.1. Pakiet <i>agh.heart.actions</i> .....	30
4.3.2. Pakiet <i>agh.heart.observers</i> .....	31
4.3.3. Pakiet <i>agh.heart.callbacks</i> .....	32
4.4. Pakiet <i>HowAreYou</i> .....	33
4.4.1. Plugin <i>AWARE</i> .....	33
4.4.2. Dialog z użytkownikiem .....	35
4.4.3. Pakiet <i>photo</i> .....	37
5. Ewaluacja.....	39
6. Podsumowanie .....	41
6.1. Podsumowanie i wnioski .....	41
6.2. Możliwości dalszego rozwoju .....	42
A. Instalacja i pierwsze uruchomienie – instrukcja krok po kroku .....	45
A.1. Etap 1: Zezwolenie na instalację dodatkowych pakietów .....	45
A.2. Etap 2: Pobranie i instalacja aplikacji .....	47
A.3. Etap 3: Konfiguracja aplikacji .....	50
B. Kwestionariusze użytkownika .....	55

# **1. Wprowadzenie**

## **1.1. Cel pracy**

Celem pracy jest zaprojektowanie, implementacja i ewaluacja mechanizmu pozyskiwania wiedzy o stanie emocjonalnym użytkownika w systemach *affective computing*. W szczególności w ramach pracy przeprowadzona zostanie analiza istniejących rozwiązań, na bazie którego opracowane zostanie rozwiązanie pozwalające na odpytywanie użytkownika o stan emocjonalny w sposób nieintruzywny.

## **1.2. Zastosowania wykrywania emocji**

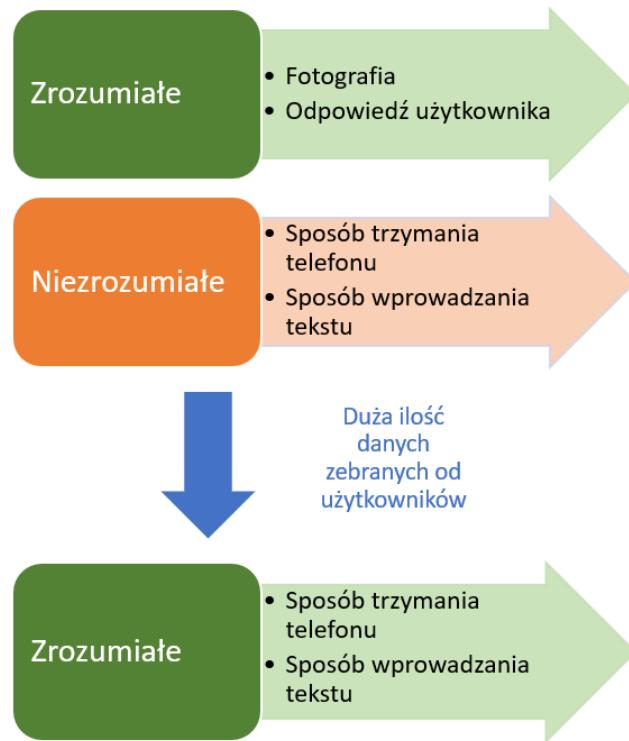
Wyobraźmy sobie świat, w którym maszyna potrafi rozpoznawać emocje. Po ciężkim dniu wchodzimy do domu, a z zestawu stereo wybrzmiewa już muzyka adekwatna do naszego nastroju. Uruchamiamy grę wideo, a jej algorytm potrafi rozpoznać, jak reagujemy na kolejne wyzwania. Chcemy poszczycić się horyzonty — inteligentny system e-learningowy prowadzi nas w dokładnie takim tempie, jakiego potrzebujemy. Kiedy jesteśmy przybici jesienną aurą, strona, która wyświetla nam reklamę, nie proponuje zakupu trampoliny, ale ciepłe bambosze USB. A gdy, nie daj Boże, przytrafia nam się nieszczęście, system awaryjny wzywa pomóc, której udaje się zdążyć na czas.

Ale w jaki sposób mamy przekazywać swoje emocje do aplikacji? Komputer ma mikrofon, klawiaturę i myszkę, które można wykorzystywać do przekazywania mu informacji. Problem w tym, że nie zawsze jesteśmy przy komputerze i że nie zawsze wystarczy mikrofon, klawiatura i myszka. Na szczęście jest urządzenie, które współczesny człowiek ma niemal zawsze przy sobie — telefon komórkowy.

## **1.3. Gromadzenie danych**

Niestety jego możliwości poznawcze są ograniczone. Telefon potrafi odczytać położenie za pomocą żyroskopu, czy tekst za pomocą dotykowej klawiatury. Idea *HowAreYou* polega na tym, żeby te ograniczone sygnały połączyć i za ich pomocą odczytać nastrój użytkownika. Smartfon może też zrobić zdjęcie kamerą – za pomocą fotografii już teraz można całkiem dokładnie określić nastrój użytkownika. Można też wprost zapytać użytkownika telefonu — jeżeli nie nadużyjemy tego rozwiązania, stwarza ono szeroki wachlarz możliwości.

Celem HowAreYou jest zbieranie dużej ilości danych – zarówno zrozumiałych (czyli takich, które system potrafi odczytać), jak i niezrozumiałych. Dzięki bazie, która może powstać w przyszłości, przyszłe aplikacje będą mogły bezinwazyjnie odczytywać nastrój użytkownika.



**Rys. 1.1.** Schemat przedstawia jak *HowAreYou* lub podobna aplikacja może wpływać na uczynienie niezrozumiałych obecnie czynników zrozumiałymi. W niniejszym badaniu nie został uwzględniony sposób wprowadzania tekstu. Podano go tutaj jako jedną z możliwości rozwinięcia pluginu *HowAreYou* w przyszłości.

Projektowanie tej aplikacji jest zadaniem niezwykle delikatnym. Człowiek, co naturalne, lubi strzec swojej prywatności. Zrezygnuje, jeżeli będzie nieustannie niepokojony, albo poczuje, że traci kontrolę nad danymi, które trafiają do bazy. Konieczne jest zaprojektowanie takiego systemu, który da użytkownikowi możliwość wyboru i pewność, że treści, które udostępnia, są przez niego kontrolowane i całkowicie bezpieczne.

## **2. Mechanizmy mediacji wiedzy**

### **2.1. Obszary zastosowań**

Kluczowymi zastosowaniami mechanizmów mediacji wiedzy są te powiązane z rozpoznawaniem emocji. Przede wszystkim chodzi tu o systemy rekomendacji, które najbardziej mogą skorzystać na dodatkowym rodzaju danych wejściowych, jakimi są emocje użytkownika.

Wyobraźmy sobie system rekomendacji muzyki dużej aplikacji takiej jak Spotify. Dzięki dodatkowej wiedzy, mógłby zaproponować nam muzykę adekwatną do naszego nastroju. Podobnie sprawa ma się z serwisami filmowymi takimi jak Netflix czy YouTube. Takie serwisy poza wyszukiwarką dysponują też stroną główną, na której przedstawiają użytkownikowi jak najlepsze propozycje. Wykorzystując informacje o naszych emocjach, te propozycje mogłyby być jeszcze lepsze. Kolejne możliwości pojawiają się w obszarze zakupów. Platformy sprzedawcze takie jak Amazon czy Allegro mogłyby skorzystać z efektów mediacji wiedzy proponując bardziej adekwatne produkty. Idąc dalej, przeglądarki internetowe takie jak Google czy DuckDuckGo mogłyby wyświetlać coraz lepsze propozycje.

Innym obszarem, gdzie mogłyby znaleźć zastosowanie wiedza zdobyta dzięki mechanizmom mediacji, mogłyby być gry i materiały edukacyjne. Emocje są niezwykle ważnym elementem przeżywania gier komputerowych. Znając reakcję gracza, system mógłby lepiej dostosować otaczający go świat.

### **2.2. Jawne metody mediacji wiedzy**

Pierwszą z kategorii metod mediacji wiedzy są metody jawne. Oznacza to, że w proces mediacji zaangażowany jest sam użytkownik. To rozwiązanie ma swoje wady i zalety. Z pewnością największą zaletą jest skuteczność rozwiązania – nikt nie wie więcej o stanie emocji niż sam użytkownik. Z kolei główną wadą jest fakt, że takie badanie może okazać się uciążliwe i irytujące, a na dłuższą metę zbytnio ingerujące w życie człowieka, czy wręcz niemożliwe.

Przykładem jawnej metody może być praca Emilii Pieczonki realizowana na Katedrze Informatyki Stosowanej. W badaniach, jakie przeprowadziła, „grupa użytkowników korzystających z telefonu Android i aplikacji mobilnej, używała podczas codziennych czynności urządzenia sensorycznego na swoim nadgarstku i odpowiadała na pytania dotyczące ich samopoczucia” [1].

Warto również zwrócić uwagę na pracę Arkadiusza Lisa, również na Katedrze Informatyki Stosowanej. W tej pracy zostało wybrane w oparciu o literaturę naukową kilka najbardziej obiecujących sposobów mediacji wiedzy. Następnie zaimplementowano zestaw widgetów, z pomocą których użytkownik "powinien być w stanie subiektywnie określić stan emocjonalny (...), a następnie przekazać tą informację do systemu" [2].

W pracy opisanej w artykule [3] naukowcy we współpracy z psychiatrą stworzyli system, z którego pomocą badany z wykorzystaniem kolorowego suwaka może określić poziom swojego niepokoju czy gniewu. Badacze skupili się przede wszystkim nie na sensorach, ale na wzorcach korzystania z telefonu udostępnianych przez system operacyjnych takich jak wykonywanie połączeń, pisanie SMSów czy lokalizacja.

Mediacja nie musi odbywać się z wykorzystaniem ekranu. Na rynku pojawiają się rozwiązania, w których system głosowy taki jak Alexa, Asystent Google, Cortana czy Siri podczas rozmowy sam wypytuje użytkownika o nastrój.

## 2.3. Niejawne metody mediacji wiedzy

Drugą kategorią metod mediacji wiedzy są metody niejawne. De facto sprowadza się to do braku bezpośredniego uczestnictwa człowieka w przekazywaniu wiedzy. Oczywiście człowiek jest zaangażowany w proces na przykład poprzez wykonywanie różnych czynności, ale nie zostanie nigdy zapytany wprost. Podobnie jak pierwsza kategoria, ta druga również ma swoje wady i zalety. Do zalet można zaliczyć wygodę – korzystanie z tych metod jest nieinwazyjne, nie ingeruje w codzienną egzystencję. Minusem jest oczywiście mniejsza skuteczność.

Przykładem takiej mediacji może być praca [4], w której naukowcy podejmują próby rozpoznania aktywności człowieka wykorzystując czujniki multimodalne. Do przetworzenia danych wykorzystywane są algorytmy wyboru cech.

Podobną metodą wykorzystali naukowcy w pracy [5] wykorzystując stworzony przez siebie model matematyczny w celu określenia czy kierowca samochodu jest trzeźwy czy pijany.

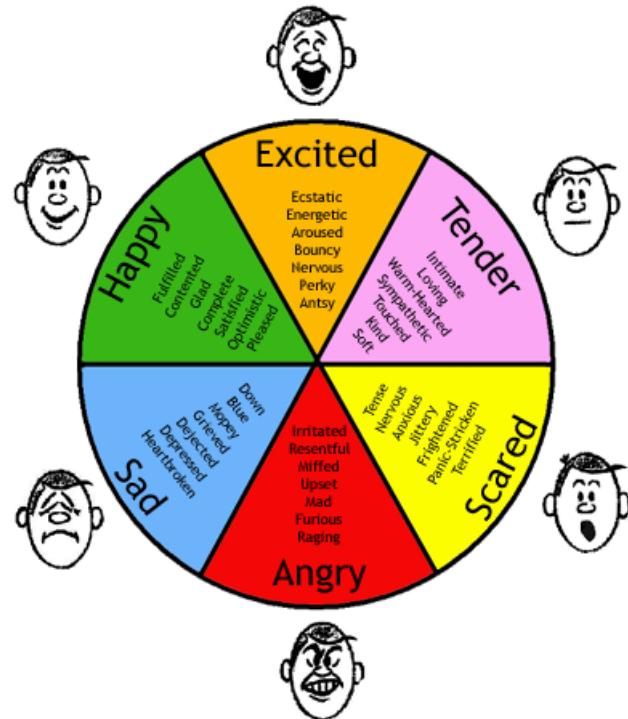
## 2.4. Pozyskiwanie wiedzy o stanie emocjonalnym w systemach *affective computing*

*Affective computing* jest paradygmatem zaproponowanym przez Rosalind Picard z Laboratorium Mediów MIT w 1997 roku[6].

Wykorzystuje on rezultaty inżynierii biomedycznej, psychologii i sztucznej inteligencji. Celem jest pozwolenie systemom komputerowym wykrywać, wykorzystywać, a nawet wyrażać emocje[7].

Przykładem pozyskiwania wiedzy może być gra wideo *Bridge Scroll-runner* stworzona w ramach pracy [7]. Naukowcy rozszerzyli możliwości gry wykorzystując czujniki tętna, temperatury skóry oraz reakcji galwanicznej skóry.

Rosalind Picard w swojej pionierskiej pracy [6] jako przykład podaje komputerowy system uczący gry na pianinie. Taki system mógłby być znacznie skuteczniejszy od klasycznego systemu dysponując możliwością wyczuwania zainteresowania, przyjemności i smutku.



**Rys. 2.1.** Koło podstawowych emocji (szczęście, podekscytowanie, rozczerlenie, strach, gniew i smutek) i ich mimicznych ekspresji.

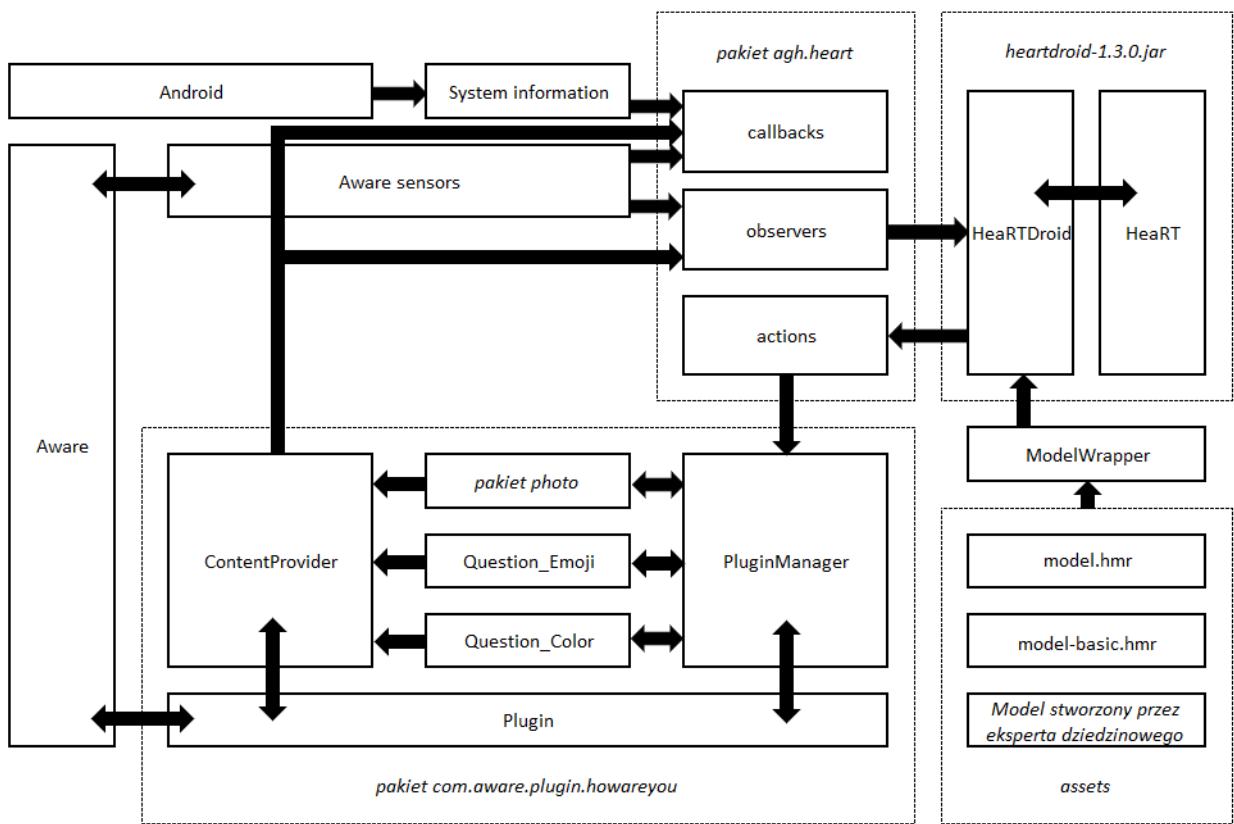
W niniejszej pracy zastosowano zarówno metody jawne jak i niejawne. W celu pozyskania wiedzy o stanie emocjonalnym zdecydowano się na syntezę odpowiedzi użytkownika i działania automatycznego systemu. Odpowiedzi użytkownika mogą być realizowane wprost poprzez aktywność z listą emocji (jak na powyższym diagramie) i emotikon do wyboru lub (bardziej psychologicznie) z wykorzystaniem palety kolorów. Automatyczny system wykorzystuje zewnętrzne API, które zwraca wartości wykrytych emocji na podstawie fotografii, która została wcześniej wykonana. W przyszłości możliwe jest rozszerzenie aplikacji o system uczący się rozpoznawać emocje z sensorów AWARE porównując duże ilości danych niezrozumiałych z sensorów z danymi otrzymanymi z wykorzystaniem jawnych i niejawnych metod mediacji wiedzy.



## 3. Architektura rozwiązania

### 3.1. Wprowadzenie do architektury

Trzon rozwiązania stanowi plugin *HowAreYou*. Plugin działa w ramach frameworka *AWARE*. Do wykorzystywania pluginu konieczna jest więc aplikacja kliencka *AWARE*. Ponadto wewnętrz pluginu wkompilowana jest również biblioteka rdzeniowa *AWARE-core*. Sam plugin działa więc jako niezależna aplikacja i jako osobna instancja *AWARE*. Aplikacja kliencka służy jedynie do konfiguracji rozszerzenia.



Rys. 3.1. Schemat przedstawiający ogólną architekturę rozwiązania.

Wewnątrz pluginu wkompilowana jest również biblioteka silnika wnioskującego *HeaRTDroid*. Dołączony pakiet *agh.heart* (*HeART-AWARE*) stanowi proxy pomiędzy zasadniczą częścią aplikacji a samym

silnikiem. Plugin zawiera też kod *XTT2* modeli wnioskujących dla silnika oraz implementuje rozwiązania, dzięki którym może komunikować się z użytkownikiem, gromadzić dane i przesyłać te dane do wspólnej bazy.

Powyższy schemat prezentuje najważniejsze składowe rozwiązania i zależności między nimi. Poza opisanymi tutaj elementami użytkownik ma do dyspozycji jeszcze aktywność ustawień, gdzie można konfigurować działanie oraz monitorować stan aplikacji.

## 3.2. Silnik wnioskujący

*HeaRTDroid* jest opartym na regułach silnikiem wnioskującym stworzonym z myślą zarówno o urządzeniach mobilnych jak i rozwiązaniach desktopowych. Bazuje on na silniku wnioskującym *HeaRT* i jest rozpowszechniany na licencji GNU GPL. Najważniejszymi cechami tego rozwiązania są[8]:

- wsparcie dla reprezentacji reguł *XTT2* i dla języka HMR+, który je tekstowo reprezentuje.
- implementacja w czystej Javie, co pozwala na integrację z dowolnym innym kodem Javy, włączając to aplikacje mobilne dla systemu Android.
- integracja z frameworkm *AWARE* zrealizowana w formie rozszerzenia *HeaRTDroid Plug-In*.
- mechanizm tzw. *callbacks* oparty na refleksji w języku Java, który pozwala na łatwą integrację z innymi systemami.
- mechanizm zarządzania niepewnością oparty na algebrze współczynników pewności i podejściu probabilistycznym.
- różnorodne tryby wnioskowania i możliwość uruchomienia wnioskowania w czasie rzeczywistym.
- oraz interaktywna powłoka linii komend *HaQuNa*[8].

Na potrzeby pluginu *HowAreYou* należało znaleźć rozwiązanie, które pozwoli na:

- swobodne zmiany modelu wnioskowania,
- możliwość uruchamiania silnika bezpośrednio na urządzeniu mobilnym z systemem operacyjnym Android,
- łatwą współpracę z frameworkiem *AWARE*, służącym jako tzw. *middleware* do akwizycji danych,
- możliwość uruchamiania wnioskowania bezpośrednio, gdy zmieni się jeden z zewnętrznych czynników.

Stąd silnik *HeaRTDroid* stał się oczywistym rozwiązaniem, jako że spełnia wszystkie powyższe założenia.

### 3.3. Model HMR

Plugin *HowAreYou* został stworzony w ten sposób, że wszystkie reguły wnioskujące silnika *HeaRTDroid* mieszą się poza kodem źródłowym aplikacji. W strukturze projektu *Android Studio* pliki dodatkowe umieszcza się zazwyczaj w katalogu *assets* - tak jest również z plikiem *model.hmr*, który zawiera szczegółowy opis reguł w języku zrozumiałym dla silnika wnioskującego.

### 3.4. Framework AWARE

Framework *AWARE* to zaawansowane narzędzie pozwalające na gromadzenie, współdzielenie i wykorzystywanie szeroko pojętego mobilnego kontekstu. Umożliwia odczyt danych z różnego rodzaju sensorów, filtrację i transfer do zdalnego serwera. Informacje kontekstowe przechowywane są w przejrzystej strukturze bazy danych SQLite.

*AWARE* stanowi skuteczne oprogramowanie pośredniczące (ang. *middleware*) umieszczone architektonicznie pomiędzy fizycznym sensorem a aplikacją użytkownika. Jednak jego głównym atutem jest możliwość realizacji pluginów – czyli rozszerzeń, dodatków – tworzonych przez niezależnych twórców bez ingerencji w kod źródłowy aplikacji klienta czy rdzenia frameworku[9].

### 3.5. Pakiet *HeaRT-AWARE*

Jedną z pierwszych kwestii, jakie trzeba było poruszyć na etapie projektowania rozwiązania, była kwestia zintegrowania ze sobą framework'a *AWARE* i silnika wnioskującego *HeaRTDroid*. Zdecydowano się tutaj skorzystać z istniejącego rozwiązania, jakim jest plugin *HeaRT-AWARE*. Aplikacja była realizowana przez Marka Wawrzosa na Katedrze Informatyki Stosowanej, a kod źródłowy został udostępniony studentom w ramach platformy AI WIKI[10].

Struktura projektu opiera się tutaj na dwóch pakietach: *agh.heart* – realizującym interakcje pluginu z silnikiem wnioskującym i *com.aware.plugin.template* – realizującym standardową strukturę i standarde zadania rozszerzenia *AWARE*. W *HowAreYou* zdecydowano się zrezygnować z tej drugiej części, jako, że dysponowano już innym pakietem *com.aware.plugin.howareyou*. Za to kluczową rolę w realizacji aplikacji odegrał pakiet pierwszy, stanowiąc trzon proxy pomiędzy *HeaRTDroidem* a *AWARE*.

Na pakiet *agh.heart* składają się następujące elementy:

- *actions* – zestaw akcji, które mogą być podjęte (uruchomione) przez silnik regułowy (czyli *de facto* mogą być wynikiem wnioskowania).
- *callbacks* – zestaw wywołań, które pozwalają mechanizmom wnioskującym na odwołanie się do parametrów zewnętrznych na przykład poprzez dostęp do baz danych pluginu, baz danych sensorów, czy informacji systemowych udostępnianych przez pakiety systemu Android.

- *model* – zestaw narzędzi pozwalających na wczytanie i obsługę modelu HMR. W pierwotnym rozwiązaniu model wczytywany był ze statycznego pola typu String. Na etapie projektowania zdecydowano się na zastąpienie statycznego wykorzystania modelu wyborem dynamicznym.
- *observers* – zestaw tzw. *BroadcastReceivers*, które odpowiadają za uruchomienie wnioskowania w chwili, gdy zostanie spełniony konkretny warunek.
- oraz klasa *HeaRTService.java* odpowiedzialna za planowanie wnioskowania. Dzięki temu rozwiązaniu poszczególne zadania wnioskujące nie są uruchamiane równocześnie[10].

Powyższe rozwiązanie zostało w ramach niniejszej pracy znacznie rozbudowane przede wszystkim poprzez zaimplementowanie dodatkowych *actions*, *callbacks* i *observers*.

## 3.6. Pakiet *HowAreYou*

### 3.6.1. Plugin AWARE

Plugin AWARE działa na takiej zasadzie jak sensor AWARE – gromadzi dane. Może to jednak robić w sposób bardziej inteligentny niż domyślnie zaimplementowane sensory – wykorzystując dodatkowe informacje z innych sensorów, filtrację informacji, czy akwizycję danych tylko w konkretnych momentach. Może też łączyć w sobie działanie wielu różnych źródeł informacji. Aby uniknąć nieoczekiwanych sytuacji, przy projektowaniu i implementacji rozszerzenia należy trzymać się struktury projektu wyznaczonej przez deweloperów AWARE:

- *Syncadapter* – usługa odpowiedzialna za synchronizację lokalnej bazy SQLite ze zdalnym serwrem.
- *ContextCard* – widok pluginu w aplikacji klienta AWARE. Pozwala użytkownikowi obserwować pracę pluginu.
- *Plugin* – klasa zarządzająca pozwoleniami systemu Android oraz wykorzystywanymi sensorami framework'a AWARE.
- *ContentProvider* – klasa zarządzająca bazami danych. Umożliwiainicjalizację tablic i wypełnianie ich danymi.
- *Settings* – klasa ustawień charakterystycznych dla danego pluginu. Z jej pomocą użytkownik może spersonalizować działanie aplikacji.

Twórcy rdzenia framework'a proponują, by komunikacja pomiędzy poszczególnymi elementami aplikacji odbywała się w formie broadcastu. Realizacja pozostałych pakietów i klas pozostawiona jest w gestii deweloperów[9].

### 3.6.2. Dialog z użytkownikiem

Jednym z najrzetelniejszych źródeł wiedzy o użytkowniku urządzenia mobilnego jest on sam. Dlaczego więc nie wykozystać tej wiedzy i nie zapytać go „Jak się czujesz?”. Należy tu jednak uważyć, aby nie przekroczyć granicy, która jest bardzo cienka. Statystyczny Kowalski niemal natychmiast zrezygnuje z naszego oprogramowania, jeżeli zacznie ono mu się naprzykrzać. Stąd konieczność pytania jedynie w uzasadnionych przypadkach – na przykład kiedy inne źródła danych kontekstowych wskażą, że nastrój właściciela telefonu uległ w ostatnim czasie zmianie. W projekcie wykorzystany jest zestaw reguł do decydowania, kiedy należy pytać o emocje.

W pluginie zaimplementowano dwa sposoby odpytywania użytkownika o nastrój: oparty o *emoji* i oparty o kolory. W obecnej realizacji właściciel telefonu komórkowego w ustawieniach aplikacji ma możliwość wyboru: żadnego, jednego lub obu sposobów.

W przypadku pytania o emocje zadaniem użytkownika po wyświetleniu ekranu jest kliknięcie odpowiedniej emotikony z napisem. Jeżeli żadna ikona nie zostanie wybrana, ekran zniknie po kilku sekundach.

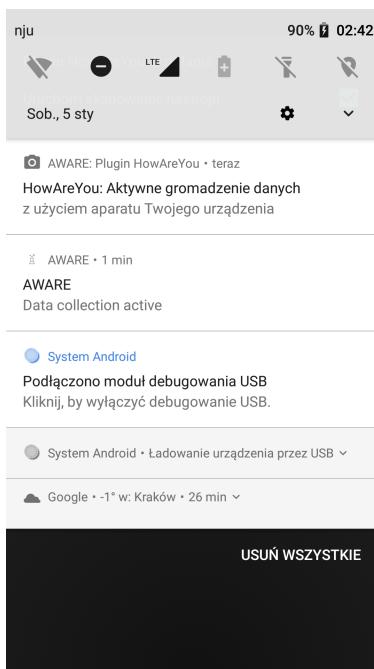
Dodatkowo wprowadzono bardziej psychologiczne pytanie – pytanie o kolory. W tym drugim przypadku zadaniem użytkownika po wyświetleniu ekranu jest wybór odpowiedniego koloru. W tym celu zaimplementowano mechanizm wyboru koloru. Jeżeli żaden kolor nie zostanie wybrany, ekran zniknie po kilku sekundach. Jeżeli zostanie wybrany, użytkownik może zatwierdzić wybór przyciskiem, zmienić wybór lub zaczekać - wówczas ekran zniknie i zostanie zapisana ostatnio wybrana odpowiedź.



**Rys. 3.2.** Dialog z użytkownikiem w formie pytania o emocje (a) oraz pytania o kolory (b).

### 3.6.3. Pakiet photo

W dwudziestym pierwszym wydaniu systemu Android, Google udostępniło deweloperom narzędzie, które pozwala kontrolować dostępne w smartfonie aparaty – *android.hardware.camera2 API*. Zaimplementowany w pluginie *HowAreYou* pakiet *photo* jest odpowiedzialny za wszystkie czynności potrzebne do wykonania zdjęcia „w tle”, to znaczy automatycznie, bez konieczności podglądu użytkownika, odpowiednim aparatem. Zdjęcia wykonywane są tylko i wyłącznie wtedy, gdy jest na to zgoda użytkownika.



Rys. 3.3. Użytkownik może zdecydować, aby powiadomienie przypominało mu o możliwości rejestracji wizerunku.

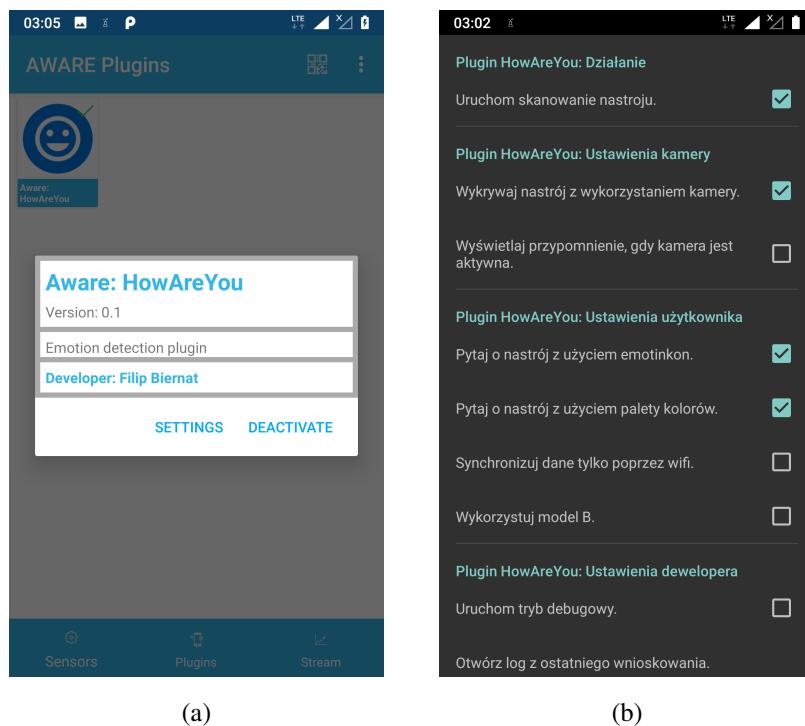
Podobnie jak dialog z użytkownikiem, rozpoznawanie emocji z wykorzystaniem kamery telefonu zostało zrealizowane w pluginie od zera – wykorzystując doświadczenie zdobyte w trakcie tworzenia aplikacji *HowAreYou* i łącząc je z możliwościami framework'a *AWARE*. Poprawiono wykorzystanie *camera2 API*. Zdjęcia są teraz wyraźne i odpowiednio naświetlone. Fotografie robione są w tle, bez ingerencji w działanie użytkownika. W przypadku niepowodzenia wykrycia emocji proces może być powtórzony zadaną ilość razy w zadanym interwale.

Samo wykrywanie emocji składa się z dwóch etapów. W pierwszym etapie wykorzystywany jest *FaceDetector* Google. Z jego pomocą plugin sprawdza obecność twarzy w kadrze zdjęciowym i dopiera najlepszą rotację zdjęcia dla dalszego procesowania. W dalszej części plik jest wysyłany do serwera *MS Face API* gdzie następuje rozpoznanie emocji. Informacja zwrotna zostaje zapisana w bazie *SQLite*.

### 3.6.4. Menu ustawień

Aby uruchomić aktywność ustawień, w aplikacji klienckiej AWARE w menu *Plugins* należy wybrać *HowAreYou*, a następnie *Settings*. Po naciśnięciu przycisku oczom użytkownika ukaże się menu, za pomocą którego można konfigurować plugin. Z poziomu menu możliwe jest:

- całkowite wyłączenie i ponowne włączenie pluginu *HowAreYou*.
- włączenie/wyłączenie wykorzystania kamery. Ta opcja może być ważna dla osób, które cenią swoją prywatność. Dzięki niej mogą kontrolować, kiedy dokładnie wykonywane są fotografie.
- włączenie/wyłączenie wyświetlania przypomnienia, gdy kamera jest aktywna. Jeżeli użytkownik bierze udział w krótkim badaniu, może nie być przyzwyczajony do tego, że jego kamera jest aktywna. Ikona aparatu na pasku powiadomień i powiadomienie po rozwinięciu górnego paska mogą mu o tym przypomnieć.
- włączenie/wyłączenie odpytywania użytkownika z użyciem emotikon oraz kolorów. Odpytywanie można wyłączyć całkowicie. Można też zdecydować się na jedną z dwóch dostępnych opcji.
- włączenie/wyłączenie trybu synchronizacji danych tylko przez wifi. Ilość danych do zsynchronizowania może być naprawdę niemała, dlatego została zaprojektowana taka opcja z myślą o osobach, które nie dysponują dużym pakietem zbędnych gigabajtów.

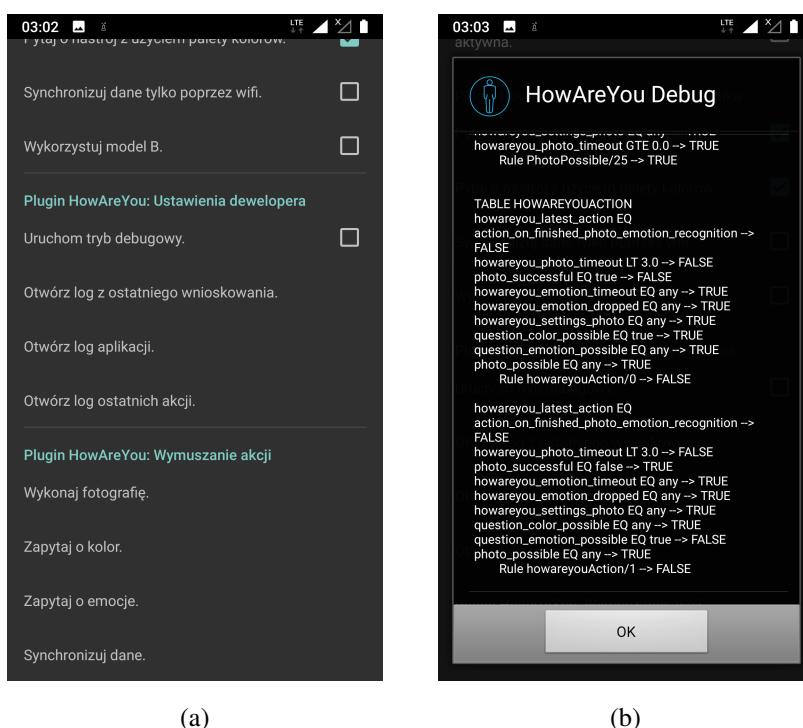


**Rys. 3.4.** Przejście do menu ustawień z aplikacji klienckiej (a) oraz widok aktywności ustawień (b).

- wybranie między modelem A (zaawansowany model HMR, patrz: rozdział 4), a modelem B (wersja uproszczona modelu). Przy instalacji wersji A (patrz: dodatek A), domyślnie wybrany jest model A, przy wersji B - model B.
- uruchomienie trybu debugowego. W trybie debugowym na ekranie urządzenia wyświetlają się w formie wiadomości *Toast* lub okna dialogowego dodatkowe informacje dotyczące działania algorytmów w tle. Na dłuższą metę użytkowanie telefonu z *HowAreYou* w trybie debugowym może być bardzo irytujące i dlatego jest niezalecane.

Z poziomu menu ustawień możemy też obserwować działanie aplikacji. Możliwe jest:

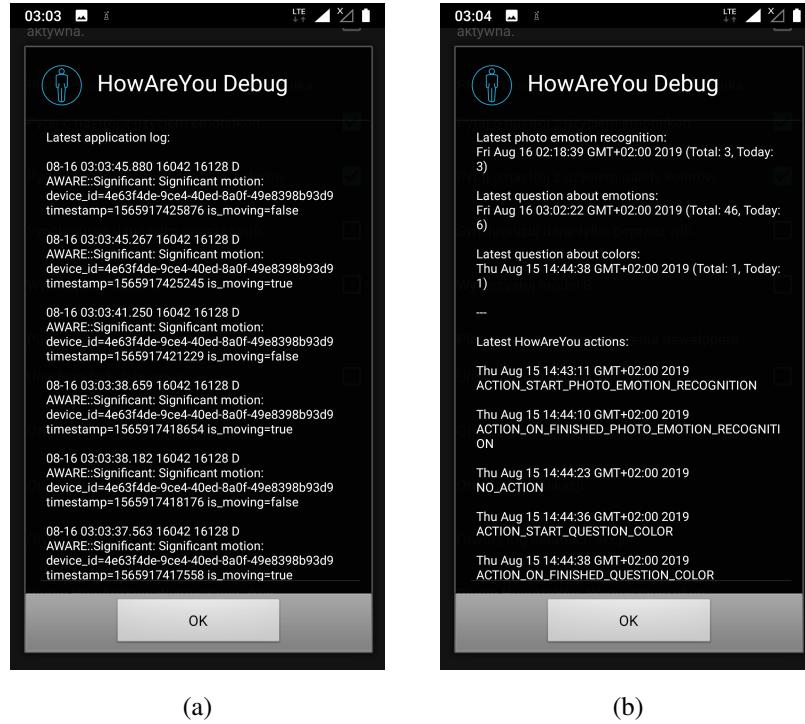
- uruchomienie logu z ostatniego wnioskowania. Taki log zawiera informacje, jakie zwraca silnik wnioskujący *HeaRTDroid*. Dane są przetworzone z użyciem specjalnego parsera, aby można je było analizować w nieco bardziej przejrzystej formie.
- uruchomienie logu aplikacji. Użytkownik może śledzić kolejne akcje tak, jakby to robił patrząc w konsoli *Logcat*. Widok umożliwia śledzenie aplikacji bez *ADB*. Log samodzielnie się nie odświeża.



**Rys. 3.5.** Widok dalszej części aktywności ustawień (a) oraz log z ostatniego wnioskowania (b).

- uruchomienie logu ostatnich akcji. Poza samą listą kolejnych akcji użytkownik ma do dyspozycji informację o znaku czasu ostatniej akcji każdego typu, o łącznej ilości akcji danego typu jaka

została wykonana na urządzeniu, oraz o ilości akcji danego typu, która wykonała się w ciągu ostatnich 24 godzin.



Rys. 3.6. Log aplikacji *HowAreYou* (a) oraz log ostatnich akcji (b).

Ostatnią z możliwości menu Ustawień jest opcja wymuszenia. Ręcznie można wymusić zapytanie o kolor, zapytanie o emocje, czy skanowanie emocji z wykorzystaniem kamery telefonu. Użytkownik może też wymusić synchronizację z zewnętrzną bazą danych. Mimo, że sama synchronizacja uruchamiana jest przez plugin regularnie, wzięto pod uwagę, że może pojawić się sytuacja, kiedy synchronizacja jest potrzebna natychmiast, na przykład podczas krótkiej chwili połączenia z siecią WiFi lub na zakończenie badania, aby nie stracić ostatnich wyników.



## 4. Implementacja

### 4.1. Wprowadzenie do implementacji

Telefon komórkowy jest urządzeniem, które na co dzień mamy przy sobie. Z tego względu już na najwcześniejszym etapie projektowania rozwiązania wybór padł właśnie na to urządzenie. Jeżeli chodzi o systemy operacyjne, zdecydowano się na system Android. Jego główną zaletą jest fakt, że pozwala deweloperom aplikacji na szeroki dostęp do opcji systemowych, sensorów, itp. Jest to również najpopularniejszy system operacyjny dla urządzeń mobilnych.

Wraz z potrzebą akwizycji dużej ilości różnego rodzaju danych z sensorów urządzenia mobilnego pojawiła się konieczność realizacji tej akwizycji. Zdecydowano się pominąć własną implementację tego mechanizmu wykorzystując szeroki wachlarz możliwości jaki daje jeden z otwartoźródłowych frameworków. Ostatecznie wybór padł na framework *AWARE*. Rozwiązanie realizujące różne metody mediacji wiedzy zostało więc stworzone jako plugin *HowAreYou* - nieoficjalne rozszerzenie *AWARE*.

Implementacja pluginu została zrealizowana w języku Java, rekomendowanym przez twórców frameworka *AWARE*[9]. Najstarszą wersją systemu operacyjnego Android, jaką wspiera *HowAreYuu* jest Lollipop (minimalne SKD zostało ustawione na 22). Środowiskiem wykorzystanym w celu realizacji zadania było Android Studio. Do kontroli wersji wykorzystano narzędzie git, do przechowywania kodu źródłowego – platformę github: [https://github.com/filipbiernat/HowAreYou\\_plugin](https://github.com/filipbiernat/HowAreYou_plugin).

### 4.2. Model HMR

#### 4.2.1. Konstruowanie modelu wnioskującego

Do tworzenia pliku *model.hmr* wykorzystana została aplikacja webowa *HeaKatE Web Editor* (w skrócie *HWE*). *HWE* jest edytorem online (stworzonym z wykorzystaniem środowiska *node.js*) służącym do tworzenia i edytowania modeli, które później wykorzystywane są przez silnik wnioskujący *HeaRTDroid* [8]. Zawarty w nim parser przetwarza reguły *XTT2* na czytelne dla użytkownika zestawy połączonych ze sobą tabel, a następnie po wybraniu opcji *Export* przekształca tablice w plik HMR z regułami *XTT2* [8].

Samo *XTT2* jest formalizmem reprezentacji wiedzy stworzonym z myślą o regułach. Służy algebraicznej i logicznej specyfikacji reguł pozwalając na zwięzły, przejrzysty i efektywny sposób wizualnej

reprezentacji wiedzy. W odróżnieniu od tradycyjnych systemów pozwala wykorzystanie „płaskiego” (jednoopoziomowego) zestawu reguł. Wprowadza tablice, wykorzystywane do reprezentowania zestawów reguł mających podobną strukturę, oraz połączenia między tablicami. Struktura zestawów tabel przypomina drzewa decyzyjne [11].

Podczas opracowywania modelu wykorzystane zostało również narzędzie *HeARTDroid Query Notation* (w skrócie *HaQuNa*). *HaQuNa* jest prostym językiem, który może być wykorzystywany w interaktywnej powłoce. Zestaw poleceń linii komend umożliwia wczytywanie, modyfikację oraz uruchamianie modeli HMR[8]. Przy implementacji *HowAreYou HaQuNa* była wykorzystywana do weryfikacji poprawności tworzonego modelu.

Należy jeszcze podkreślić, że model HMR został architektonicznie oddzielony od implementacji aplikacji. Dzięki zastosowaniu tego rozwiązania, plugin *HowAreYou* nie musi być powiązany z obecnie zaimplementowanym domyślnym modelem. Ekspert domenowy, który nie musi być nawet osobą techniczną, może bez trudu z wykorzystaniem edytora online stworzyć nowy zestaw reguł *XTT2* i zastąpić w strukturze projektu plik *model.hmr*.

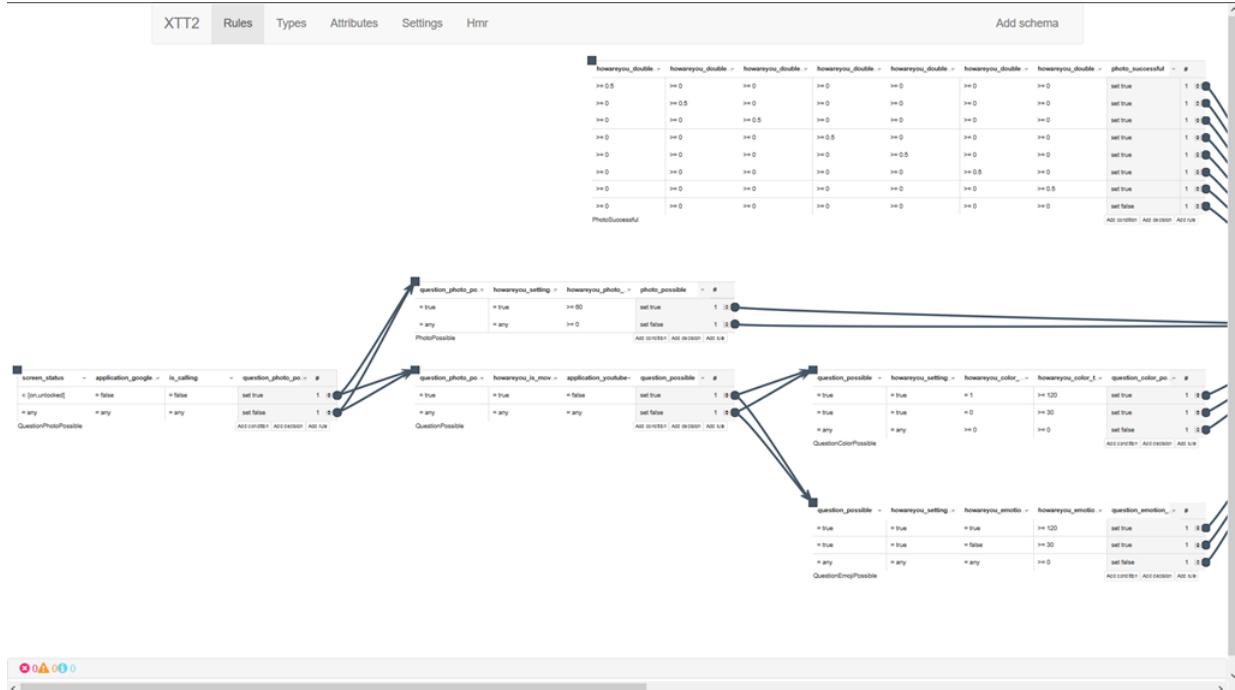
#### 4.2.2. Zaawansowany model wnioskujący

##### 4.2.2.1. Realizacja zaawansowanego modelu wnioskującego

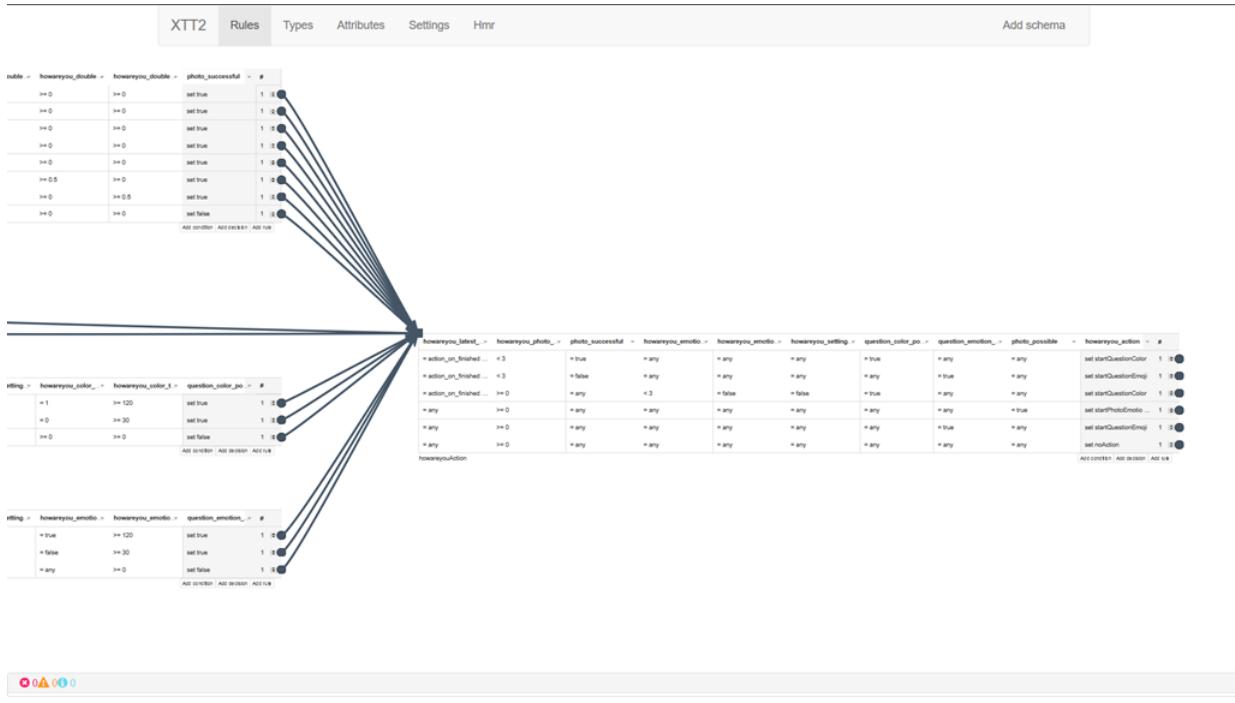
Celem realizacji zaawansowanego modelu wnioskującego było opracowanie modelu, z pomocą którego zostaną wybrane jak najlepszy moment i sposób na zapytanie użytkownika o jego samopoczucie. Możliwe jest zapytanie niejawne poprzez wykonanie i przetworzenie fotografii lub zapytanie jawne - o kolor lub bezpośrednio o emocje.

W procesie wnioskowania brane są pod uwagę czynniki takie jak czas, który upłynął od ostatniego zapytania, korzystanie z nawigacji samochodowej, wykonywanie połączenia telefonicznego, oglądanie filmów, czy wykonywanie ruchu. Uwzględniane są też wyniki analizy fotografii oraz wiedza, czy w danej chwili użytkownik korzysta z ekranu telefonu.

Poniższe obrazy przedstawiają widok modelu HMR w edytorze *HWE*. Model został skonstruowany jako zestaw tabel, które zależąc od informacji z aplikacji oraz od siebie nawzajem realizują reguły decyzyjne. Decyzja z jednej tabeli ma bezpośredni wpływ na działanie kolejnej. Ostatecznie zgodnie z wynikiem z ostatniej tabeli *howareyouAction* plugin *HowAreYou* podejmuje określoną akcję. W kolejnej części rozdziału omawiane zostają krok po kroku zasady działania poszczególnych tabel poniższego modelu.



Rys. 4.1. Zaawansowany model wnioskujący: część 1.



Rys. 4.2. Zaawansowany model wnioskujący: część 2.

#### 4.2.2. Tabela QuestionPhotoPossible

Tabela zwraca prawdę, jeżeli spełnione są wspólne warunki konieczne do zapytania użytkownika lub wykonania zdjęcia.

Żeby uruchomić pytanie lub zrobić zdjęcie ekran telefonu musi być aktywny, telefon nie może prowadzić nawigacji, ani wykonywać połączenia.

screen_status	application_google_maps	is_calling	question_photo_possible
in [on,unlocked]	eq false	eq false	set true
eq any	eq any	eq any	set false

Rys. 4.3. Zaawansowany model wnioskujący: tabela QuestionPhotoPossible.

#### 4.2.2.3. Tabela PhotoPossible

Tabela zwraca prawdę, jeżeli spełnione są warunki konieczne do wykonania zdjęcia.

Żeby zrobić zdjęcie telefon musi spełniać warunki konieczne wspólne dla zdjęcia i pytania. Dodatkowo opcja wykonywania zdjęć musi być uruchomiona i czas, jaki upłynął od ostatniego zdjęcia, musi być większy lub równy 60 minut.

question_photo_possible	howareyou_settings_photo	howareyou_photo_timeout	photo_possible
eq true	eq false	gte 60	set true
eq any	eq any	gte 0	set false

Rys. 4.4. Zaawansowany model wnioskujący: tabela PhotoPossible.

#### 4.2.2.4. Tabela QuestionPossible

Tabela zwraca prawdę, jeżeli spełnione są warunki konieczne do zapytania użytkownika.

Żeby uruchomić pytanie telefon musi spełniać warunki konieczne wspólne dla zdjęcia i pytania. Dodatkowo musi znajdować się w ruchu (jeżeli jest w bezruchu, najprawdopodobniej jest odłożony) i nie może być uruchomiona aplikacja do oglądania filmów.

question_photo_possible	howareyou_is_moving	application_youtube	question_possible
eq true	eq true	eq false	set true
eq any	eq any	eq any	set false

Rys. 4.5. Zaawansowany model wnioskujący: tabela QuestionPossible.

#### 4.2.2.5. Tabela QuestionColorPossible

Tabela zwraca prawdę, jeżeli spełnione są warunki konieczne do zapytania użytkownika o kolor.

Żeby uruchomić pytanie o kolor telefon musi spełniać warunki konieczne do zadania pytania. Dodatkowo ustawienia pluginu muszą pozwalać zapytać o kolor. Jeżeli ostatnie pytanie o kolor było odrzucone, kolejne pytanie może być zadane po 120 minutach. Jeżeli nie było odrzucone – po 30 minutach.

question_possible	howareyou_settings_question_color	howareyou_color_dropped	howareyou_color_timeout	question_color_possible
eq true	eq true	eq 1	gte 120	set true
eq true	eq true	eq 0	gte 30	set true
eq any	eq any	gte 0	gte 0	set false

Rys. 4.6. Zaawansowany model wnioskujący: tabela QuestionColorPossible.

#### 4.2.2.6. Tabela QuestionEmojiPossible

Tabela zwraca prawdę, jeżeli spełnione są warunki konieczne do zapytania użytkownika o emocje.

Żeby uruchomić pytanie o emocje telefon musi spełniać warunki konieczne do zadania pytania. Dodatkowo ustawienia pluginu muszą pozwalać zapytać o emocje. Jeżeli ostatnie pytanie o emocje było odrzucone, kolejne pytanie może być zadane po 120 minutach. Jeżeli nie było odrzucone – po 30 minutach.

question_possible	howareyou_settings_question_emoji	howareyou_emotion_dropped	howareyou_emotion_timeout	question_emoji_possible
eq true	eq true	eq true	gte 120	set true
eq true	eq true	eq false	gte 30	set true
eq any	eq any	eq any	gte 0	set false

Rys. 4.7. Zaawansowany model wnioskujący: tabela QuestionEmojiPossible.

#### 4.2.2.7. Tabela PhotoSuccessful

Tabela zwraca prawdę, jeżeli najnowsze zdjęcie w bazie danych uznaje się za udane.

Żeby uznać zdjęcie za udane wystarczy, że wartość prawdopodobieństwa jednej z wykrytych emocji (za wyjątkiem emocji *neutral*) jest większa niż próg (który obecnie wynosi 50 procent pewności).

howareyou_double_anger	howareyou_double_contempt	howareyou_double_disgust	howareyou_double_fear	howareyou_double_happiness	howareyou_double_sadness	howareyou_double_surprise	photo_successful
gte 0.5	gte 0	gte 0	gte 0	gte 0	gte 0	gte 0	set true
gte 0	gte 0.5	gte 0	gte 0	gte 0	gte 0	gte 0	set true
gte 0	gte 0	gte 0.5	gte 0	gte 0	gte 0	gte 0	set true
gte 0	gte 0	gte 0	gte 0.5	gte 0	gte 0	gte 0	set true
gte 0	gte 0	gte 0	gte 0	gte 0.5	gte 0	gte 0	set true
gte 0	gte 0	gte 0	gte 0	gte 0	gte 0.5	gte 0	set true
gte 0	gte 0	gte 0	gte 0	gte 0	gte 0	gte 0.5	set true
gte 0	gte 0	gte 0	gte 0	gte 0	gte 0	gte 0	set false

Rys. 4.8. Zaawansowany model wnioskujący: tabela PhotoSuccessful.

#### 4.2.2.8. Tabela howareyouAction

Tabela zwraca prawdę, jeżeli najnowsze zdjęcie w bazie danych uznaje się za udane.

- Jeżeli ostatnią akcją jaką plugin wykonał było zdjęcie i to zdjęcie było niedawno (mniej niż 3 minuty temu) i to zdjęcie było udane, to telefon zapyta o kolor, jeżeli spełnione są warunki do zapytania o kolor.
- Jeżeli ostatnią akcją jaką plugin wykonał było zdjęcie i to zdjęcie było niedawno (mniej niż 3 minuty temu), ale nie było udane, to telefon zapyta o emocje, jeżeli spełnione są warunki do zapytania o emocje.
- Jeżeli ostatnią akcją jaką plugin wykonał było pytanie o emocje i to pytanie o emocje było niedawno (mniej niż 3 minuty temu) i użytkownik odpowiedział na to pytanie, to telefon zapyta o kolor, jeżeli spełnione są warunki do zapytania o kolor. Ta reguła (de facto dwa pytania jedno po drugim) wykona się tylko wtedy, gdy użytkownik w ustawieniach wyłączył wykonywanie zdjęć.

4. Jeżeli jest możliwe zrobienie zdjęcia, to telefon wykona zdjęcie.
5. Jeżeli nie jest możliwe zrobienie zdjęcia, to telefon zapyta o emocje, jeżeli spełnione są warunki do zapytania o emocje.
6. Jeżeli żadna z akcji nie jest możliwa, telefon nie wykona akcji.

howareyou_latest_action	howareyou_photo_timeout	photo_successful	howareyou_emotion_timeout	howareyou_emotion_dropped	howareyou_settings_photo	question_color_possible	question_emoji_possible	photo_possible	howareyou_action
[1]	lt 3	eq true	eq any	eq any	eq any	eq true	eq any	eq any	set startQuestionColor
[1]	lt 3	eq false	eq any	eq any	eq any	eq any	eq true	eq any	set startQuestionEmoji
[2]	gte 0	eq any	lt 3	eq false	eq false	eq true	eq any	eq any	set startQuestionColor
eq any	gte 0	eq any	eq any	eq any	eq any	eq any	eq any	eq true	[3]
eq any	gte 0	eq any	eq any	eq any	eq any	eq any	eq true	eq any	set startQuestionEmoji
eq any	gte 0	eq any	eq any	eq any	eq any	eq any	eq any	eq any	set noAction

[1]: eq action\_on\_finished\_photo\_emotion\_recognition

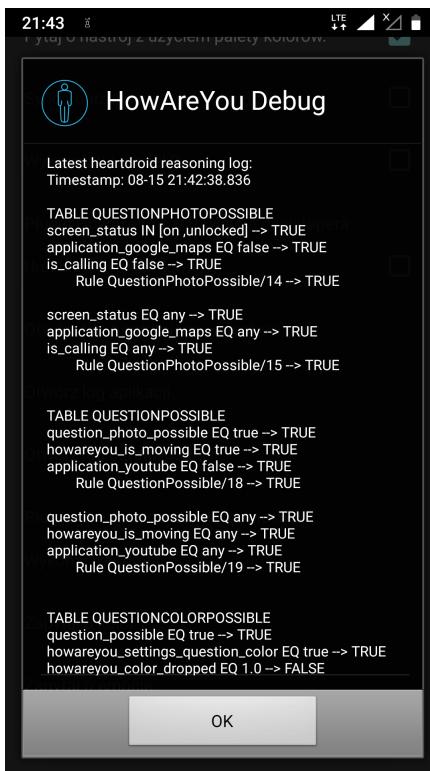
[2]: eq action\_on\_finished\_question\_emoji

[3]: set startPhotoEmotionRecognition

Rys. 4.9. Zaawansowany model wnioskujący: tabela howareyouAction.

#### 4.2.2.9. Obserwowanie reguł wnioskujących

Działanie systemu wnioskującego wykorzystującego zaawansowany model wnioskujący można obserwować na bieżąco z wykorzystaniem trybu debugowego. Po wybraniu opcji *Otwórz log z ostatniego wnioskowania* w ustawieniach pluginu *HowAreYou*, oczom użytkownika ukaże się okno ze szczegółowym opisem elementów wnioskowania: tabel, reguł i decyzji, jakie zostały podjęte.



Rys. 4.10. Tryb debugowy: widok z fragmentem loga z wnioskowania HMR.

### 4.2.3. Uproszczony model wnioskujący

#### 4.2.3.1. Realizacja uproszczonego modelu wnioskującego

W celach badawczych został stworzony również model uproszczony. Dzięki niemu osoba przeprowadzająca badanie jest w stanie porównać, jaki wpływ na odbieranie przez uczestnika uciążliwości badania ma złożoność samego modelu. Model prosty został stworzony w kontraste do pierwszego modelu. Sam model wykorzystuje tylko 2 czynniki: wiedzę czy ekran jest urządzenia włączony oraz czas od ostatniego zapytania. Model symuluje prosty i toporny system odpytywania użytkownika.

Zdecydowaną zaletą *HowAreYou* jest fakt, iż w celach porównawczych nie było niezbędnego tworzenie dodatkowej aplikacji, która odpytywałaby użytkownika w sposób uproszczony, ani nawet modyfikacja kodu źródłowego aplikacji. Jedyną potrzebną czynnością była zmiana samego modelu HMR. Model uproszczony można aktywować w ustawieniach aplikacji. Można również zainstalować plugin *HowAreYou* bezpośrednio z modelem uproszczonym. Podczas pobierania pliku instalatora należy wybrać wersję B (patrz: *dodatek A*).

#### 4.2.3.2. Tabela howareyouAction

Tabela zwraca prawdę, jeżeli najnowsze zdjęcie w bazie danych uznaje się za udane.

1. Jeżeli ekran urządzenia jest aktywny i od ostatniego zapytania minęło co najmniej 60 minut, aplikacja zapyta użytkownika o emocje.
2. Jeżeli któryś z powyższych warunków nie jest spełniony, telefon nie wykona żanej akcji.

screen_status	howareyou_emotion	howareyou_action	#
∈ [on,unlocked]	>= 60	set startQuestionEmoji	1
= any	= any	set noAction	1

howareyouAction

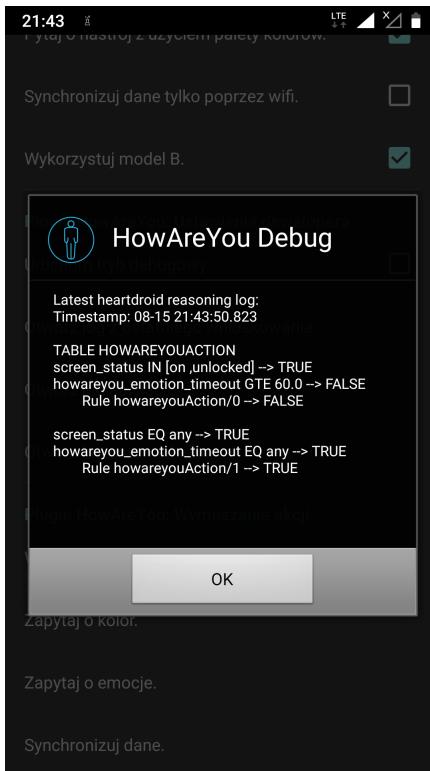
Add condition Add decision Add rule

Rys. 4.11. Uproszczony model wnioskujący: tabela howareyouAction.

#### 4.2.3.3. Obserwowanie reguł wnioskujących

Podobnie jak w przypadku zaawansowanego modelu, działanie systemu wnioskującego wykorzystującego uproszczony model można obserwować na bieżąco z wykorzystaniem trybu debugowego. Tutaj

jednak log z wnioskowania jest znacznie krótszy - do tego stopnia, że obejmuje tylko jedną tabelę i tylko dwie reguły.



**Rys. 4.12.** Tryb debugowy: widok z loga z wnioskowania HMR z modelem uproszczonym.

### 4.3. Pakiet HeaRT-AWARE

Jak już wspomniano w poprzednim rozdziale, zadaniem pakietu *agh.heart* było zintegrowanie ze sobą framework AWARE i silnika wnioskującego *HeaRTDroid*. Zostało ono zaprojektowane przez autora pakietu w formie trzech pomniejszych specjalistycznych pakietów: *actions*, *observers* i *callbacks*[10].

#### 4.3.1. Pakiet *agh.heart.actions*

W pluginie *HowAreYou* zrezygnowano z dostępnych początkowo w pakiecie akcji *DisableBlueTooth* (ang. wyłącz Bluetooth) i *EnableBlueTooth* (ang. włacz Bluetooth). Przy realizacji aplikacji monitorującej nastrój użytkownika nie było potrzeby wykorzystywania tego typu akcji. W zamian zaimplementowano akcje związane ze specyfiką zadania:

- *HowAreYou\_StartPhotoEmotionRecognition* – uruchom usługę odpowiedzialną za rozpoznawanie emocji użytkownika na podstawie fotografii.

- *HowAreYou\_StartQuestionEmoji* – uruchom aktywność odpowiedzialną za zapytanie użytkownika o nastrój z wykorzystaniem widoku emotikon.
- *HowAreYou\_StartQuestionColor* – uruchom aktywność odpowiedzialną za zapytanie użytkownika o nastrój z wykorzystaniem widoku kolorów.

Każda z powyższych akcji przesyła wiadomość broadcastową do PluginManagera w pakiecie *com.aware.plugin.howareyou*. PluginManager uruchamia następnie zaplanowaną aktywność lub usługę.

Aby uprościć rozwiązanie, zaimplementowano również klasę bazową *HowAreYou\_Action*, z której korzystają powyższe klasy. Dzięki takiemu rozwiązaniu, jeżeli w przyszłości pojawiłaby się potrzeba uwzględnienia w modelu nowej akcji dołączonej do PluginManagera, dołączenie jej do zbioru *actions* powinno być bardzo proste.

#### 4.3.2. Pakiet *agh.heart.observers*

W swojej początkowej formie Pakiet *HeART-AWARE* dostarczał użytkownikowi trzy różne klasy obserwatora: *Accelerometer*, *Location* i *Screen*.

Na potrzeby niniejszej pracy zaimplementowano jeszcze jedną dodatkową klasę *HowAreYou*. Pozwala ona silnikowi wnioskującemu reagować na sytuacje, kiedy użytkownik udzieli odpowiedzi na pytanie (lub odrzuci pytanie wybierając opcję *Nie teraz*), czy też na sytuację, kiedy zostanie wykonana fotografia i zewnętrzne API zwróci wyniki rozpoznawania emocji. W tym przypadku silnik wnioskujący może na bieżąco reagować na te wyniki podejmując decyzję, czy na przykład zapytać dodatkowo o kolor (jeżeli wiemy, co obecnie czuje użytkownik) lub o emocje (jeżeli nie udało ich się rozpoznać automatycznie).

Nowa klasa ma postać *BroadcastReceivera*, który reaguje na akcje:

- *ACTION\_ON\_FINISHED\_QUESTION\_COLOR* – zakończono zapytanie użytkownika o kolor,
- *ACTION\_ON\_FINISHED\_QUESTION\_EMOJI* – zakończono zapytanie użytkownika bezpośrednio o emocje,
- *ACTION\_ON\_FINISHED\_PHOTO\_EMOTION\_RECOGNITION* – zakończono (z powodzeniem lub nie) proces rozpoznawania emocji użytkownika z wykorzystaniem kamery telefonu.

W modelu HMR dostępnym w ramach *HowAreYou* zrezygnowano z wykorzystywania *observers*: *Accelerometer* oraz *Location*. Pozostałe dwa wystarczająco dobrze radzą sobie z uruchamianiem wnioskowania. Kluczowy jest tutaj obserwator ekranu, który daje znać, czy użytkownik korzysta z telefonu. Wspomaga go nowy obserwator z akcjami specyficznymi dla bieżącego rozszerzenia.

Jeżeli chodzi o dane z akcelerometru, kłopotliwe może okazać się ich przetwarzanie i filtrowanie – mamy tutaj do czynienia z ogromną ilością „surowych” danych. W przypadku danych z lokalizacji, ciężko wyobrazić sobie dla nich zastosowanie jako dla czynnika uruchamiającego wnioskowanie w tym konkretnym przypadku. W innym przypadku wiążałoby się to pewnie ze stworzeniem złożonych map

obszarów, gdzie po przekroczeniu pewnej granicy, wnioskowanie może zostać uruchomione. Nic nie stoi jednak na przeszkodzie, żeby w dalszych pracach nad *HowAreYou* powrócić do obecnie pominiętych *observers*.

#### 4.3.3. Pakiet *agh.heart.callbacks*

Ostatnim z kluczowych elementów przy wnioskowaniu są tzw. *callbacks*, czyli wywołania, które pozwalają mechanizmom wnioskującym na dostęp do parametrów zewnętrznych. Początkowo dysponowaliśmy trzema takimi wywołaniami – dla żyroskopu (*Gyroscope*), lokalizacji (*Location*) i ekranu urządzenia (*Screen*). Konieczne było jednak opracowanie kolejnych *callbacks*. Im większa ich liczba – tym dokładniejszy model HMR można skonstruować.

W pierwszej kolejności zaimplementowano nowe klasy bazowe upraszczające strukturę pakietu. Dzięki zastosowaniu takich ogólnych *callbacks*, uzyskano znacznie łatwiejszą możliwość rozbudowy o kolejne czynniki zewnętrzne.

- *GenericDbCallback* – klasa opakowuje wszystkie funkcjonalności, jakie potrzebne są do pozyskania odpowiednich informacji z bazy danych. Klasa dziedzicząca po *GenericDbCallback* musi jedynie wyspecyfikować parametry dostępu do bazy danych SQLite takie jak URI i lista kolumn.
- *Application\_Generic* – wykorzystuje sensor AWARE *Applications\_Foreground*. Po pobraniu informacji z bazy danych sensora porównuje nazwę zdefiniowanego pakietu z zapisanym w podklasie pakietem konkretnej aplikacji. Zwraca 1, jeżeli aktualnie wykorzystywana aplikacją jest ta zdefiniowana w podklasie, w przeciwnym przypadku zwraca 0.
- *HowAreYou\_Generic* – początkowo klasa została stworzona z myślą o wywołaniach charakterystycznych dla konkretnego pluginu. Z czasem znalazła zastosowanie również dla wykorzystywania klasycznych tablic framework AWARE. Cechą charakterystyczną jest dodatkowa kolumna "*\*\_timeout*". Dzięki niej silnik wnioskujący otrzymuje informację jak dawno miało miejsce wydarzenie, do którego się odwołuje. *Timeout* mierzony jest jako różnica pomiędzy bieżącym czasem systemowym oraz czasem zdarzenia zapisanym w bazie danych (w kolumnie *timestamp*) i jest wyrażony w minutach.

Lista nowo-dodanych wywołań ma się następująco:

- *Application\_Facebook* – informuje silnik czy aktualnie uruchomiona jest aplikacja *Facebook*.
- *Application\_GoogleMaps* – informuje silnik czy aktualnie uruchomiona jest nawigacja.
- *Application\_YouTube* – informuje silnik czy użytkownik ogląda filmy w aplikacji *YouTube*.
- *Communication* – informuje silnik czy obecnie trwa połączenie telefoniczne. Wywołanie nie wykorzystuje bazy danych, w zamian na obiekcie *CommunicationObserver* uruchamiana jest metoda *isCalling*. W ten sposób wykorzystany jest sensor *Communication* framework AWARE.

- *HowAreYou\_Color* – przekazuje do silnika składowe RGB ostatnio wybranego przez użytkownika koloru. Przesyła także informację, czy użytkownik nie odrzucił ostatniego zapytania.
- *HowAreYou\_Emotion* – przekazuje do silnika informacje: o ostatnio wybranej przez użytkownika ikonie emocji oraz czy użytkownik nie odrzucił ostatniego zapytania.
- *HowAreYou\_IsMoving* – informuje silnik czy urządzenie aktualnie jest w ruchu. Wykorzystywany jest tutaj sensor *Significant* framework'a AWARE.
- *HowAreYou\_LatestAction* – informuje silnik o ostatniej akcji, jaka została wywołana w PluginManagerze. Dzięki temu model HMR jest w stanie zareagować odpowiednio na konkretną akcję, na przykład poprzez sprawdzenie wyników rozpoznawania zdjęć, czy zapytania użytkownika.
- *HowAreYou\_Photo* – przekazuje do silnika 8 współczynników emocji rozpoznanych przez *MS Face API* (*ANGER*, *CONTEMPT*, *DISGUST*, *FEAR*, *HAPPINESS*, *NEUTRAL*, *SADNESS*, *SURPRISE*).
- *HowAreYou\_Settings* – informuje silnik o trybie w jakim działa obecnie aplikacja. Model HMR musi mieć możliwość rozróżnienia trybu pracy, aby podejmować odpowiednie decyzje, przykładowo nie zlecać wykonania fotografii, jeżeli użytkownik nie wyraża zgody na wykorzystywanie kamery. Ustawienia, do jakich dostęp ma model to:
  - *SETTINGS\_PLUGIN\_HOWAREYOU* – informacja czy skanowanie nastroju jest aktywne,
  - *SETTINGS\_PHOTO* – informacja czy użytkownik wyraził zgodę na wykorzystywanie kamery,
  - *SETTINGS\_QUESTION\_EMOJI* – informacja czy użytkownik wyraził zgodę na odpytywanie o nastroj z wykorzystaniem aktywności z emotikonami emocji,
  - *SETTINGS\_QUESTION\_COLOR* – informacja czy użytkownik wyraził zgodę na odpytywanie o nastroj z wykorzystaniem aktywności z paletą kolorów,
  - *SETTINGS\_PHOTO\_NOTIFICATION* – informacja czy użytkownik życzy sobie pokazywać przypomnienie o fakcie, że uruchomione jest skanowanie nastroju z wykorzystaniem kamery telefonu.

## 4.4. Pakiet *HowAreYou*

### 4.4.1. Plugin AWARE

Twórcy framework'a AWARE zalecają, aby jednym z głównych zadań klasy Plugin w każdym z rozszerzeń, była poprawna inicjalizacja wszystkich zależności[9]. W *HowAreYou* większość odpowiedzialności tej klasy to właśnie poprawne uruchomienie i skonfigurowanie poszczególnych elementów.

1. W celu zapewnienia bezpieczeństwa system Android wymaga od użytkownika przynajmniej jednorazowego wyrażenia zgody na wykorzystywanie wrażliwych składowych systemu takich jak dostęp do: pamięci wewnętrznej, kontaktów, czy Internetu. Na twórcach aplikacji spoczywa odpowiedzialność za upewnienie się, że użytkownik wyraził odpowiednie zgody. Właśnie w klasie Plugin realizowane jest takie zapytanie o zgody. Zazwyczaj ma ono miejsce tylko przy pierwszym uruchomieniu. Przy uruchomieniu aplikacja sprawdza w systemie czy zgody zostały już wcześniej udzielone, jeżeli nie – formułuje zapytanie.
2. W przypadku jeżeli zostały wyrażone zgody wymagane przez funkcjonalności pluginu i poszczególnych sensorów, egzekucja może przejść o krok dalej. W pierwszej kolejności sprawdzane są bazy danych. Jeżeli ich brak - odpowiednie tablice są inicjalizowane.
3. Następnie następuje stworzenie i rejestracja tzw. *observers* z pakietu *agh.heart*. Przypomnijmy, że dzięki tym klasom stale nasłuchującym wiadomości broadcastowych, silnik wnioskujący jest w stanie szybko reagować na zmieniające się otoczenie.
4. W dalszej części inicjalizowane są ustawienia: część zawsze (na przykład ustawienia ogólne dotyczące choćby parametrów synchronizacji danych, częstotliwości czyszczenia danych, czy możliwości wykorzystywania transferu danych), a część tylko przy pierwszym uruchomieniu.
5. Kolejną czynnością jest sprawdzenie czy użytkownik jest już uczestnikiem *Study*. Jest to ważne zwłaszcza przy pierwszym uruchomieniu lub przy jednym z kolejnych, jeżeli przy wcześniejszych nie udało się nawiązać połączenia internetowego. *Study* (ang. badanie) jest dostarczany przez framework *AWARE* zestawem narzędzi umożliwiającym przeprowadzenie badania. Zespół badawczy może korzystać z bezpłatnych przestrzeni baz danych, z możliwości wizualizacji danych oraz z możliwości komunikowania się z uczestnikami badania. W przypadku niniejszej pracy, dołączenie do study jest realizowane automatycznie przez plugin. Adres badania to: <https://api.awareframework.com/index.php/webservice/index/2415/4a13qF3BHs8y>. Nie są gromadzone dane wrażliwe jak na przykład fotografie, a jedynie „suche” liczby w tabelach dotyczących emocji, koloru, czy fotografii.
6. Klasa Plugin podczas uruchamiania aplikacji inicjalizuje też niezależną instancję frameworka *Aware*, która działa wewnątrz pluginu, i niektóre jej sensory.
7. Ostatnim etapem jest uruchomienie ikony powiadomienia o wykorzystaniu kamery telefonu, jeżeli taka opcja została wybrana przez użytkownika w menu ustawień.

Kolejną ważną klasą w pakiecie *com.aware.plugin.howareyou* jest *PluginManager*. Jest BroadcastReceiverem, gdzie spotykają się wszystkie najważniejsze wiadomości. Reaguje na decyzje silnika wnioskującego *HeART-AWARE* po wywołaniu akcji, a także na wymuszenie wywołania akcji (rozpoczęcie procedury skanowania nastroju z użyciem fotografii, pytanie o kolory lub pytanie o emocje) z menu ustawień. *PluginManager* odpowiada również za stworzenie logu ostatnich akcji, który może zostać wyświetlony

użytkownikowi po wybraniu odpowiedniej opcji w aktywności ustawień. Ponadto klasa zapisuje ostatnio wykonaną akcję i przekazuje ją dalej do mechanizmu wnioskującego.

Ostatnim z elementów, na które warto zwrócić uwagę jest ContentProvider. Jego odpowiedzialnością jest komunikacja z bazą danych SQLite. ContentProvider inicjalizuje w pamięci wewnętrznej telefonu plik „plugin\_howareyou.db”, a następnie wypełnia go trzeba tabelami: „photo\_data”, „color\_data” i „emotion\_data”.

1. Pierwsza z tablic została stworzona z myślą o przechowywaniu wyników działania usługi rozpoznawania emocji z wykorzystaniem telefonu komórkowego. Poza standardowymi kolumnami jakimi są ID oraz znak czasu (ang. *timestamp*) mamy tutaj do czynienia z: „double\_anger”, „double\_contempt”, „double\_disgust”, „double\_fear”, „double\_happiness”, „double\_neutral”, „double\_sadness”, „double\_surprise”. Nazwy i charakter tych tablic są ściśle powiązane z właściwościami *MS Face API*, które zwraca rezultat rozpoznawania twarzy właśnie jako sumę siedmiu emocji: złości, pogardy, niesmaku, strachu, szczęścia, obojętności, smutku i zaskoczenia. Poszczególne współczynniki zawsze sumują się do jedności, stąd zdecydowano się na zmienoprzecinkowy typ reprezentacji danych.
2. Zadaniem drugiej tablicy jest przechowywanie informacji o wynikach zapytań użytkownika z wykorzystaniem mapy kolorów. Po wybraniu koloru, jego wartość zostaje rozbita na trzy składowe i wpisana do kolumn „color\_red”, „color\_green” i „color\_blue” reprezentujących składowe RGB. Dodatkowa kolumna „color\_dropped” zapisuje informację o tym, czy udało się zrealizować pytanie o kolor. Po wyświetleniu się pytania użytkownik ma zazwyczaj kilka sekund na odpowiedź. Jeżeli pytanie pozostaje bez odpowiedzi, zostaje to również odnotowanie. Ponadto użytkownik ma do dyspozycji opcję „Nie teraz”. Przesuwając szybko palcem w bok może odrzucić pytanie, jeżeli jest na przykład irytujące. To również zostaje odnotowane. Kolumny tabeli zawierają tylko liczby całkowite.
3. Ostatnia z tablic przechowuje informacje o wynikach zapytań użytkownika z wykorzystaniem emotikon. Wybierając emotikony zdecydowano się na 6 podstawowych emocji: „emotion\_happy”, „emotion\_excited”, „emotion\_tender”, „emotion\_scared”, „emotion\_angry”, „emotion\_sad” opisujących: szczęście, podekscytowanie, rozczulenie, strach, gniew i smutek. Podobnie jak powyżej, wprowadzono również kolumnę dropped. Kolumny tabeli zawierają tylko liczby całkowite.

#### 4.4.2. Dialog z użytkownikiem

Najważniejszą metodą mediacji wiedzy w niniejszej pracy jest pytanie w prost o emocje. Takie pytanie daje nam największą pewność, co do tego co czuje użytkownik aplikacji. Niestety nadużywanie może okazać się irytujące, stąd aby je udoskonalić dokonano stosowanie silników wnioskujących, aby pytać możliwie w jak najlepszych momentach.

Samo odpytywanie realizowane jest poprzez uruchomienie aktywności Question\_Emoji. Na widok aktywności składa się 6 przycisków – każdy z nich zrealizowany poprzez połączenie emotikony (grafiki dostarczanej na wolnej licencji przez Google) oraz podpisu. Użytkownik może przekazać, że jest szczęśliwy, podekscytowany, rozczułony, przestraszony, zdenerwowany lub smutny.

Klasa Question\_Emoji dziedziczy po SlidableActivity – funkcjonalności rozpowszechnianej na wolnej licencji Apache i udostępnionej społeczności przez stronę <https://github.com/r0adkll/Slidr>. Szybkie przesunięcie palcem w bok po ekranie pozwala odrzucić pytanie o emocje. W ten sposób można przekazać sygnał, że w danej chwili zapytanie jest niepożądane – jest to tak zwana opcja „Nie teraz”. Może się okazać szczególnie przydatna w przyszłości, jeżeli w grę wejdzie samodzielne odkrywanie reguł.

Równocześnie z aktywnością uruchamia się zadanie asynchroniczne. W tle realizowane jest odliczanie 10 sekund. Jeżeli w tym czasie nie zostanie udzielona odpowiedź, aktywność znika po prostu z ekranu. Próba dialogu zostaje oczywiście odnotowana.

Wybór opcji, czy to emocji, czy opcji „Nie Teraz”, czy też wygaśnięcia timera jest przekazywany do ContentProvidera, a następnie zapisywany w bazie danych. Do Plugin Managera zostaje przesłana informacja, że zapytanie zostało zakończone.

Dodatkową jawną metodą mediacji wiedzy jest pytanie o kolor. Bez zebrania dużej ilości danych ciężko przewidzieć jak odczytywać takie odpowiedzi. Odpowiedź z wykorzystaniem mapy kolorów można odnosić na przykład do niejawnej metody mediacji, jaką jest rozpoznanie emocji z wykorzystaniem kamery. Dopiero łącząc te wyniki, można wysnuć wnioski. Ta forma badania ma raczej format psychologiczny a nie inżynierski.

Pytanie o kolor jest zrealizowane w postaci aktywności Question\_Color. Czołowe miejsce w układzie tej aktywności zajmuje dwuwymiarowa mapa kolorów Color Picker – funkcjonalność oparta na wolnej licencji Apache i udostępniona społeczności przez stronę <https://github.com/QuadFlask/colorpicker>. Poza mapą Color Picker dostarcza również pasek jasności koloru, z którego można skorzystać.

Dodatkowo na ekranie pojawiają się dwa przyciski. Pierwszym jest przycisk zatwierdzenia koloru. Jeżeli popatrzylibyśmy na wcześniej omówioną aktywność, nie zastosowano tam takiego rozwiązania. Przy wyborze koloru jest po prostu wiele różnych możliwości i na przykład przesunięcie po mapie może być wygodniejsze dla użytkownika, niż pojedyncze kliknięcie. Przycisk Anuluj działa tak jak szybkie przesunięcie palcem w bok.

Sama opcja przesunięcia („Nie teraz”) omówiona wcześniej w tej sekcji jest również dostępna w tej aktywności. Podobnie jak funkcjonalność, która po 10 sekundach powoduje zniknięcie aktywności i powrót do wcześniejszego stanu. Dodatkowo wprowadzono też zadanie asynchroniczne, które uruchamia licznik po wybraniu palcem jakiegoś koloru. Po trzech sekundach bezruchu aplikacja uznaje ten kolor za wybrany i zatwierdza w ten sam sposób, jakby kliknięto przycisk Wybierz.

Po zamknięciu aktywności trzy składowe RGB przekazywane są do ContentProvidera, a następnie trafiają do bazy danych SQLite. Podobnie jak wcześniej, do bazy trafia również informacja, czy pytanie zostało zignorowane lub odrzucone.

#### 4.4.3. Pakiet photo

Kiedy silnik wnioskujący podejmie decyzję, że przyszedł czas na wykonanie fotografii (lub czynność ta zostanie wymuszona z poziomu ustawień) PluginManager uruchamia usługę EmotionRecognitionService odpowiedzialną za rozpoznawanie emocji. Usługa zostaje uruchomiona w tle, w osobnym wątku tak, aby nie wpływać w żaden sposób na działanie aplikacji. Celem usługi jest wykonanie zdjęcia, a następnie nawiązanie łączności z zewnętrznym API, które zwróci wynik w postaci szeregu współczynników. Jeżeli na którymkolwiek etapie rozpoznawania się nie powiedzie, cała procedura zostaje uruchomiona ponownie po określonej liczbie sekund. I tak czynność powtarzana jest określoną ilość razy.

Najważniejszym zadaniem EmotionRecognitionService jest konfiguracja *camera2 API* i wykonanie zdjęcia „w tle”, czyli bez ingerencji w to, co robi użytkownik na ekranie telefonu. Wykonanie takiego zdjęcia znacząco różni się od tego, co znamy z aplikacji *Aparat*. Samo wywołanie czynności „zrób zdjęcie” nie przyniesie nam pożdanego efektu. Konieczne jest przeprowadzenie całego procesu wyboru, uruchomienia i konfiguracji urządzenia.

Po skonfigurowaniu kamery uruchamiany jest w tle obiekt słuchacza EmotionRecognitionPhotoProcessor, który implementuje funkcjonalność *ImageReader.OnImageAvailableListener* dostarczaną przez jeden z pakietów systemu Android. Zasada działania słuchacza sprowadza się do tego, że w momencie, gdy kamera aparatu wykona zdjęcie, zostanie wywołana metoda *onImageAvailable()*. Implementacja samej metody sprowadza się do przetwarzania wykonanego zdjęcia do odpowiednich formatów tak, aby mogło być wykorzystane do dalszych operacji.

W pierwszej kolejności wywoływana jest metoda *detect()* na obiekcie *FaceDetector* z pakietu *com.google.android.gms.vision.face* dostarczanego przez system Android. Takie wykrycie twarzy dokonuje się lokalnie na naszym urządzeniu. Na tym etapie nie jest potrzebne wysłanie grafiki i przetworzenie jej zdalnie na zewnętrznym serwerze. Dopiero kiedy mamy pewność, że obraz, jaki wykonaliśmy, zawiera całą twarz pojedynczej osoby, egzekucja przechodzi do wykrywania emocji z użyciem zewnętrznego API.

Firma Microsoft w ramach Usług Kognitywnych platformy Azure dostarcza usługę Rozpoznawanie twarzy. Pozwala ona zewnętrznym aplikacjom na weryfikację twarzy, wykrywanie twarzy i rozpoznawanie emocji. Bezpłatny pakiet limituje funkcjonalność do 20 zapytań na minutę i 30 tysięcy zapytań miesięcznie. Po przekroczeniu tego limitu dalsze korzystanie z API staje się niemożliwe do momentu wygaśnięcia ograniczeń lub wykupienia płatnej funkcjonalności. Jak na potrzeby niniejszej pracy pakiet bezpłatny jest wystarczającym rozwiązaniem. Firma Microsoft definiuje funkcjonalność rozpoznawania emocji w następujący sposób: „Interfejs API rozpoznawania twarzy zawiera teraz funkcje rozpoznawania emocji, które zwracają poziom pewności dla zestawu emocji, takich jak gniew, pogarda, odraza, strach, zadowolenie, smutek i zaskoczenie, oraz braku emocji w przypadku każdej twarzy widocznej na obrazie. Te emocje są rozumiane jako międzykulturowe i komunikowane przez określoną uniwersalną mimikę twarzy”[12].

W celu komunikacji z zewnętrznym serwerem zaimplementowano specjalne zadanie asynchroniczne *EmotionRecognitionTask*. Dostarczany przez firmę Microsoft obiekt *FaceServiceClient* komunikuje się

z chmurą z wykorzystaniem REST API. Aby ograniczyć transfer danych, skonfigurowany jest jedynie do przesyłania informacji o samych emocjach użytkownika. Po odebraniu rezultatów, wyniki przesyłane są z powrotem do EmotionRecognitionPhotoProcessor i wpisywane do bazy danych. Każda z emocji ma w bazie danych odpowiadającą sobie kolumnę w typie zmiennoprzecinkowym. Z jednego obrazu może być wykryte kilka emocji, ale współczynniki zwracane przez *MS Face API* sumują się zawsze do jedności.

## **5. Ewaluacja**



## **6. Podsumowanie**

### **6.1. Podsumowanie i wnioski**

- Odczytywanie emocji przez urządzenia może przyczynić się do polepszenia jakości życia w kategoriach: rozrywki, nauki, pracy, marketingu, bezpieczeństwa i wielu innych.
- Smartfon, jako urządzenie, które zawsze jest pod ręką, nadaje się bardzo dobrze do monitorowania nastroju użytkownika.
- Na dłuższą metę nagrywanie dźwięku, robienie zdjęć, czy odpytywanie użytkownika nie ma racji bytu z uwagi na naturalną dla człowieka potrzebę chronienia swojej prywatności. Konieczne jest wprowadzenie nowych metod rozpoznawania nastroju, najlepiej przez wykorzystanie istniejących sposobów akwizycji danych.
- Dane z dotychczas zrozumiałych i niezrozumiałych dla maszyny źródeł są częściowo skorelowane. Przez czasową akwizycję informacji z jednej i drugiej kategorii u wielu ludzi można stworzyć bazę, dzięki której w przyszłości rozpoznawanie nastroju nie będzie dla użytkownika w żaden sposób uciążliwe.
- Konieczna jest aplikacja do gromadzenia danych. Musi być bezpieczna, lekka, nieuciążliwa, prosta i intuicyjna. Musi dawać użytkownikowi kontrolę nad treściami i przesyłać zgromadzone informacje do bazy dostępnej na serwerze.
- Framework *AWARE* pozwala deweloperom na skutecną implementację: inteligentnego mechanizmu odpytywania użytkownika, mechanizmu automatycznej obsługi kamery bez ingerencji użytkownika, transferu danych w kierunku serwera czy odpowiedniego reagowania na zdarzenia takie jak odblokowanie ekranu, wykonanie zdjęcia, wybór przycisku powiadomienia, zmiana opcji.

## 6.2. Możliwości dalszego rozwoju

- Możliwości aplikacji możemy w łatwy sposób zwiększyć rozbudowując model HMR. W tym celu należałoby zwiększyć liczbę czynników, które można wykorzystać w tabelach w modelu wnioskującym (tzw. *callbacks*). Implementacja dodatkowych wywołań opartych na sensorach, informacjach udostępnianych przez system Android czy nawet dodatkowych elementach oprogramowania (takich jak np. specjalistyczna klawiatura software'owa pozwalająca gromadzić kolejne dane) może pozwolić na tworzenie bardziej zaawansowanych i bardziej skutecznych modeli.
- Implementacja dodatkowych obserwatorów (ang. *observers*) może pozwolić na częstsze wnioskowanie, a co za tym idzie na bardziej dynamiczne, szybsze, skuteczniejsze reagowanie na zmieniające się zewnętrzne warunki.
- Inną z możliwości jest wprowadzenie kolejnej dozy sztucznej inteligencji. Dysponując danymi zebranymi przez aplikację oraz informacją czy użytkownik jest zadowolony ze skuteczności odpytywania (jaką daje na przykład przesunięcie w bok, czyli opcja *Nie teraz* w widoku emocji czy widoku koloru – taki wybór będzie sygnałem dla pluginu, że w tej sytuacji nie należy przeszkadzać użytkownikowi), można stworzyć model samodzielnie odkrywający zależności i ostatecznie w ten sposób dynamicznie wykształtać reguły, które obecnie muszą być zapisane w formacie *XTT2* w modelu HMR.
- Ostatnią z zaproponowanych w niniejszej pracy możliwości dalszego rozwoju pluginu *HowAreYou* jest rozbudowanie systemu odpytywania użytkownika o emocje. Jedną z opcji jest tutaj integracja już istniejącej aplikacji, która realizuje taki dialog z użytkownikiem.

Przykładem może być tutaj rozwiązywanie zrealizowane w ramach pracy dyplomowej inżynierskiej na Katedrze Informatyki Stosowanej przez Arkadiusza Lisa. W ramach tej pracy dokonana została analiza możliwości przeniesienia i uruchomienia na platformie Android istniejących środowisk do przeprowadzania eksperymentów, a następnie opracowany został zbiór *widgetów* służących do zbierania subiektywnych afektywnych ocen użytkowników w eksperimentach z użyciem urządzeń mobilnych [2].

Poniższe grafiki przedstawiają przykładowe *widgety* z powyższej pracy, które zostały dołączone do pluginu *HowAreYou* z wykorzystaniem widoku *WebView*. Są to:

- *Geneva Emotion Wheel* – „Metoda opracowana przez profesora Klausu Scherera składa się z dyskretnych grup emocji ułożonych w formie koła” [2].
- Koło Plutchika – *widget* oparty o postać koła w formie „róży wiatrów” [2].



Rys. 6.1. Zaawansowane widgety zintegrowane z pluginem *HowAreYou: Geneva Emotion Wheel* (a) oraz Koło Plutchika (b).



## A. Instalacja i pierwsze uruchomienie – instrukcja krok po kroku

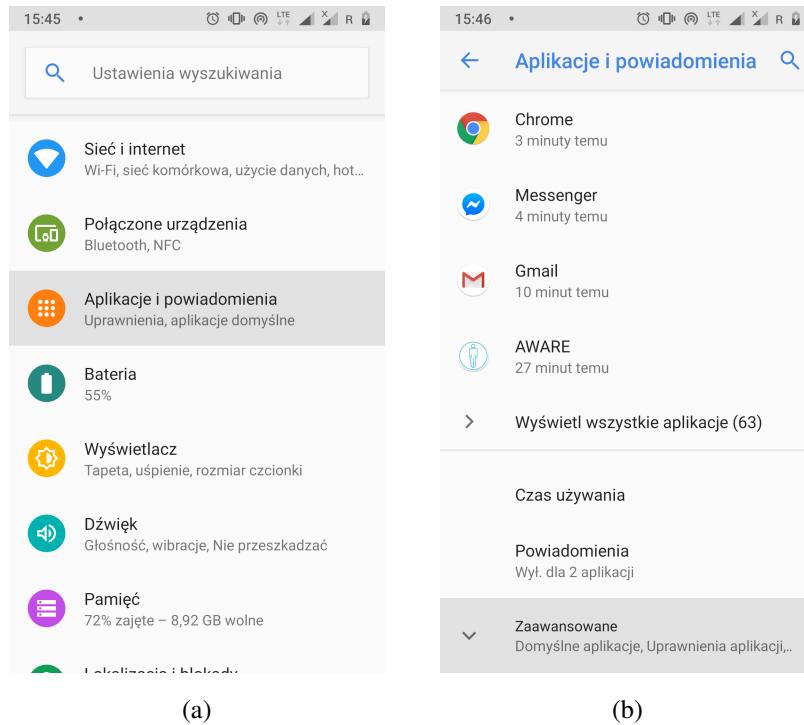
Klient *AWARE* jest aplikacją w fazie intensywnego rozwoju. W ostatnim czasie twórcom niemal udało się dostosować go do najnowszych wymagań firmy Google dla najnowszej wersji systemu Android Pie i umieścić wersję beta w Sklepie Play. Jednym z warunków była rezygnacja ze wsparcia pluginów tworzonych przez niezależnych deweloperów i integracja wewnętrz aplikacji klienckiej oficjalnie dostępnych pluginów[9].

W związku z powyższym, aplikacja *HowAreYou* nie zostanie zainstalowana w klasyczny sposób. Zamiast sklepu Play do instalacji zostanie wykorzystana przeglądarka. Użytkownik musi przejść poprzez 3 etapy, które pozwolą na poprawną instalację, konfigurację i uruchomienie aplikacji klienta oraz pluginu *HowAreYou*. Niniejsza instrukcja wraz z graficznymi ilustracjami krok po kroku prowadzi przez ten proces, który na pierwszy rzut oka może wydawać się skomplikowany.

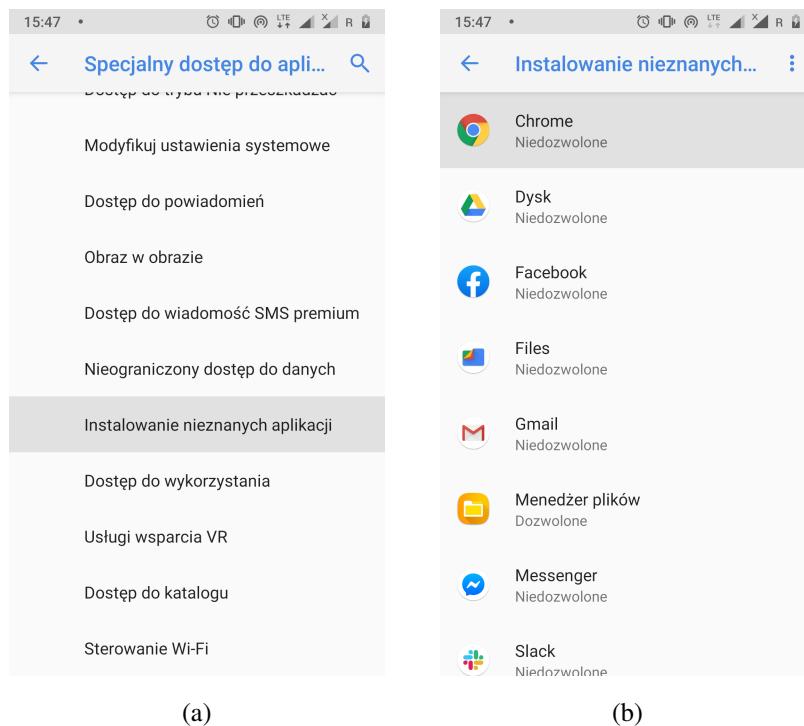
### A.1. Etap 1: Zezwolenie na instalację dodatkowych pakietów

W pierwszej kolejności pozwolimy przeglądarce na instalację dodatkowych pakietów.

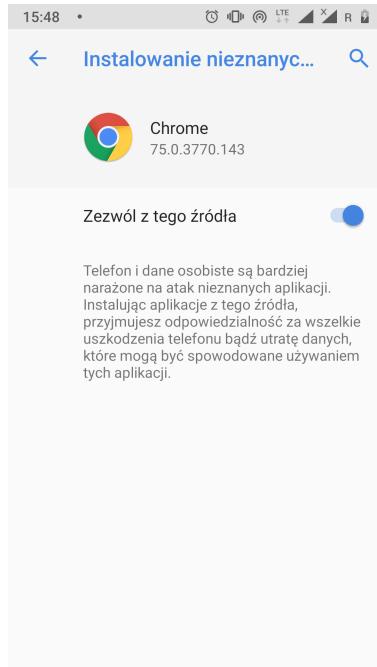
1. Przejdź do *Ustawień* systemu Android.
2. Wybierz *Aplikacje i powiadomienia*, a następnie *Zaawansowane*.
3. Wybierz *Specjalny dostęp do aplikacji*, a następnie *Instalowanie nieznanych aplikacji*.
4. Znajdź przeglądarkę, którą na co dzień używasz.
5. Kliknij *Zezwól z tego źródła*.



Rys. A.1. Kroki 1 i 2: Przejście do zaawansowanych ustawień aplikacji.



Rys. A.2. Kroki 3 i 4: Aktywność *Specjalny dostęp do aplikacji* (a) oraz aktywność *Instalowanie nieznanych aplikacji* (b).

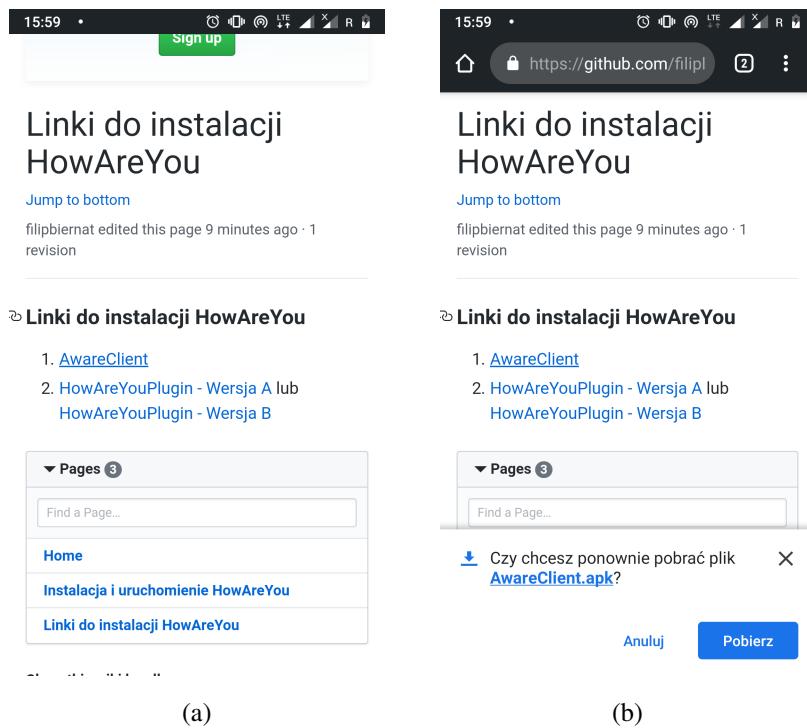


Rys. A.3. Krok 5: Aktywność ustawień aplikacji.

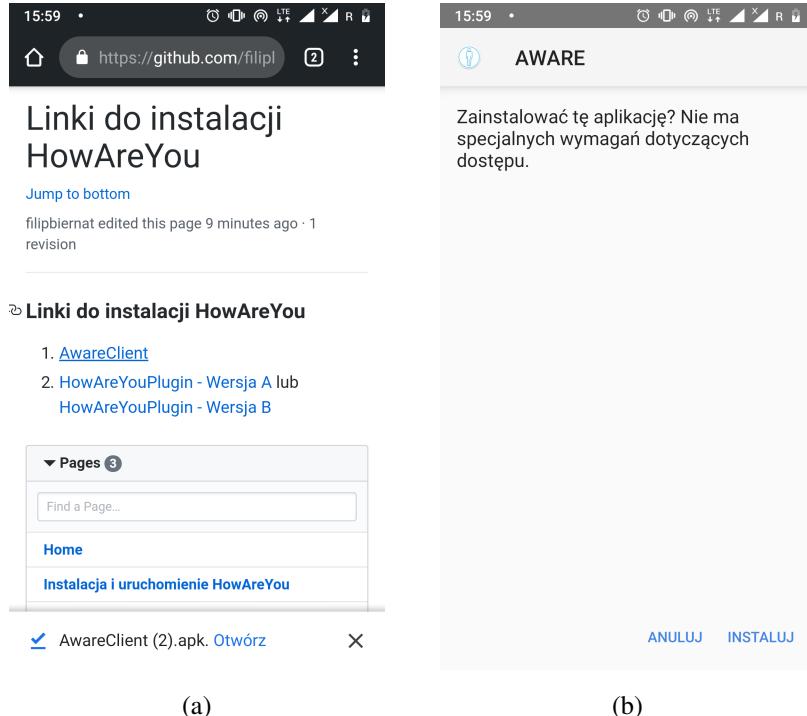
## A.2. Etap 2: Pobranie i instalacja aplikacji

Następnie pobierzemy i zainstalujemy aplikacje potrzebne do przeprowadzenia badania.

1. W przeglądarce telefonu, którą wybrałeś w poprzednim kroku, odwiedź stronę [tiny.cc/97tz9y](http://tiny.cc/97tz9y) lub [https://github.com/filipbiernat/HowAreYou\\_plugin/wiki/Linki-do-instalacji-HowAreYou](https://github.com/filipbiernat/HowAreYou_plugin/wiki/Linki-do-instalacji-HowAreYou).
2. Pobierz plik instalatora klikając na *AwareClient*.
3. Przeglądarka powiadomi Cię o pobraniu pakietu. Kliknij *Otwórz*.
4. Rozpocznie się instalacja aplikacji. Kliknij *Instaluj*.

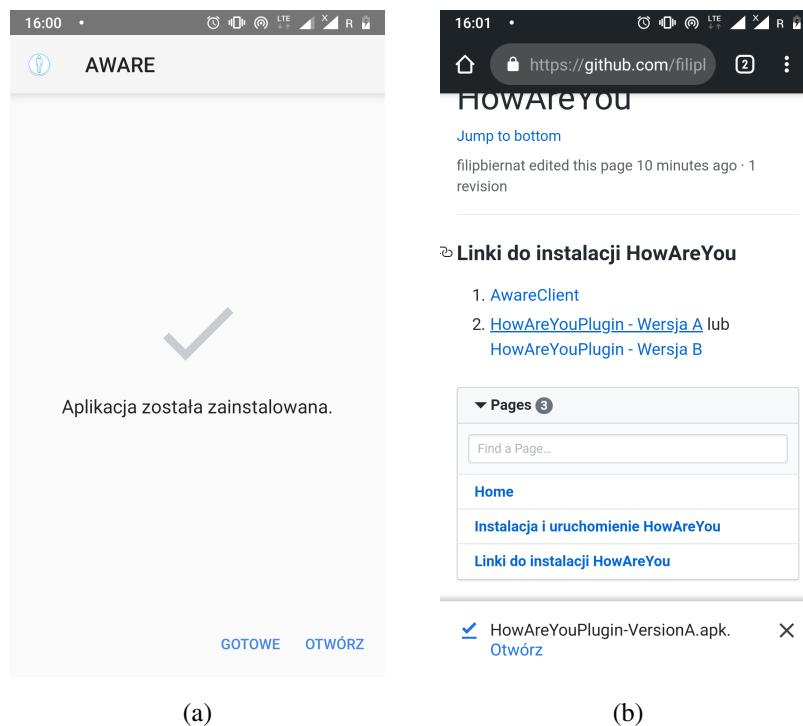


**Rys. A.4.** Kroki 1 i 2: Strona www z linkami umożliwiającymi pobranie aplikacji (a) oraz rozpoczęty proces pobierania aplikacji (b).

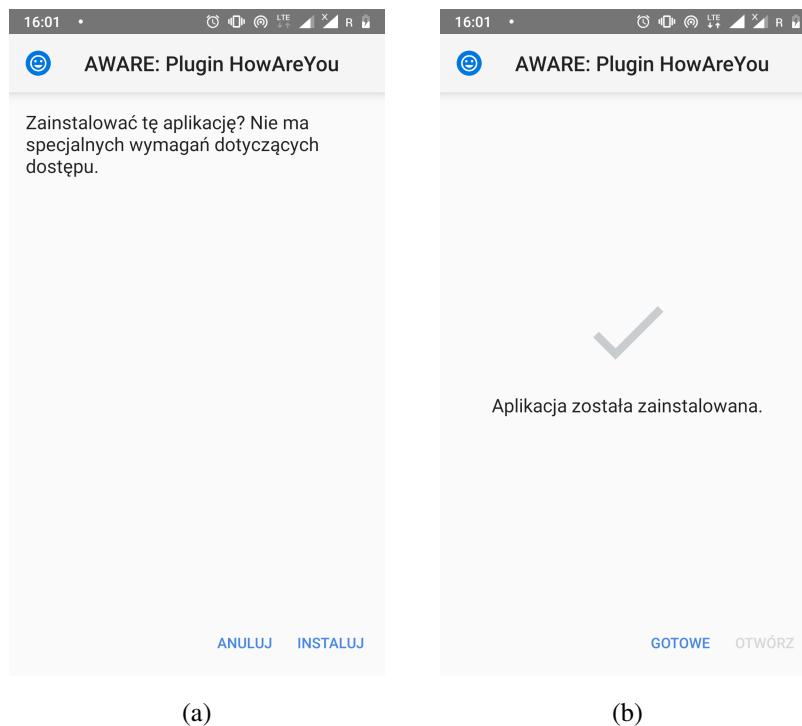


**Rys. A.5.** Kroki 3 i 4: Wybór przycisków *Otwórz* (a) oraz *Instaluj* (b).

5. Po zakończeniu instalacji kliknij *Gotowe* i powróć do przeglądarki.
6. Przed rozpoczęciem badania zostałeś poproszony o pobranie wersji A lub wersji B aplikacji. Upewnij się którą wersję powinieneś wykorzystać.
7. Pobierz plik instalatora klikając na *HowAreYouPlugin - Wersja A* lub *HowAreYouPlugin - Wersja B*.
8. Podobnie jak wcześniej, kliknij *Otwórz*. Rozpocznie się instalacja aplikacji. Kliknij *Instaluj*.
9. Ponownie kliknij przycisk *Gotowe*. Zamknij przeglądarkę i powróć do ekranu głównego.



Rys. A.6. Kroki 5 i 7: Wybór przycisku *Gotowe* (a) oraz wybór wersji pluginu (b).

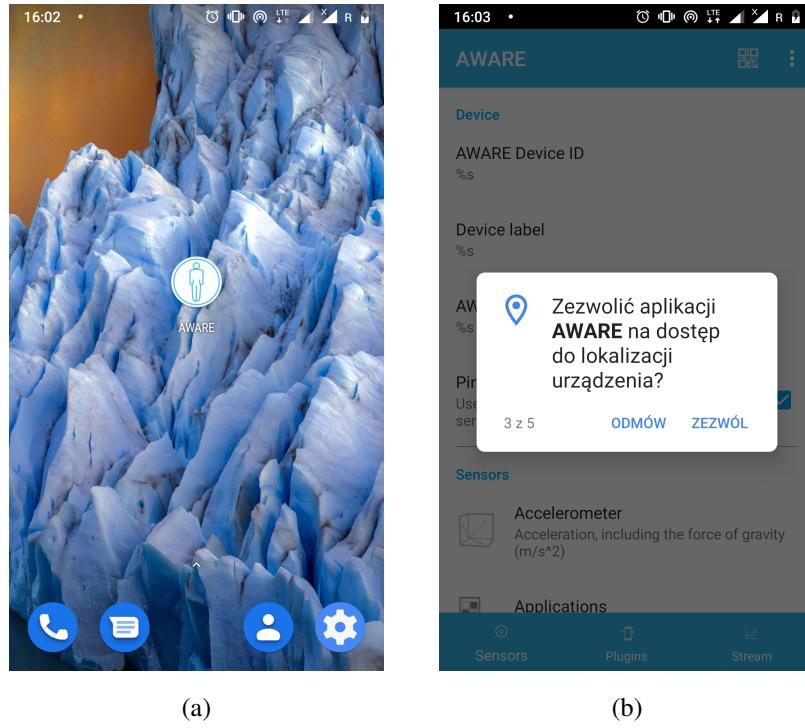


Rys. A.7. Kroki 8 i 9: Wybór przycisków *Instaluj* (a) oraz *Gotowe* (b).

### A.3. Etap 3: Konfiguracja aplikacji

Ostatnim krokiem będzie konfiguracja aplikacji klienckiej oraz pluginu *HowAreYou*.

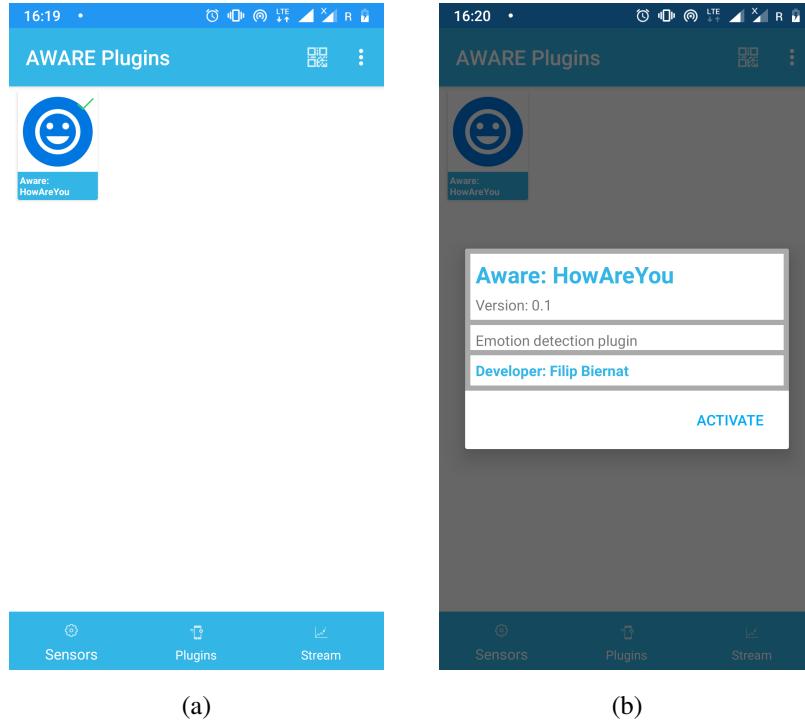
1. Uruchom aplikację AWARE.
2. Zostaniesz zapytany o zgody na wykorzystanie zasobów telefonu. Zatwierdź odpowiednie zgody.
3. W aplikacji przejdź do zakładki *Plugins*.
4. Wybierz plugin *HowAreYou*. Jeżeli plugin nie jest jeszcze aktywny, kliknij *Activate*.



(a)

(b)

**Rys. A.8.** Kroki 1 i 2: Uruchomienie aplikacji AWARE (a) oraz widok zapytania użytkownika o zgodę (b).

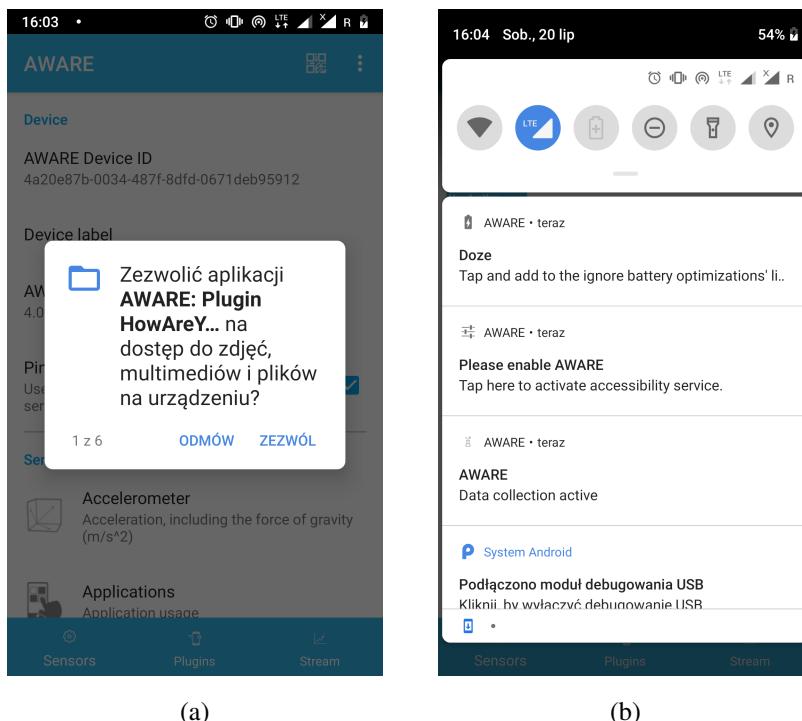


(a)

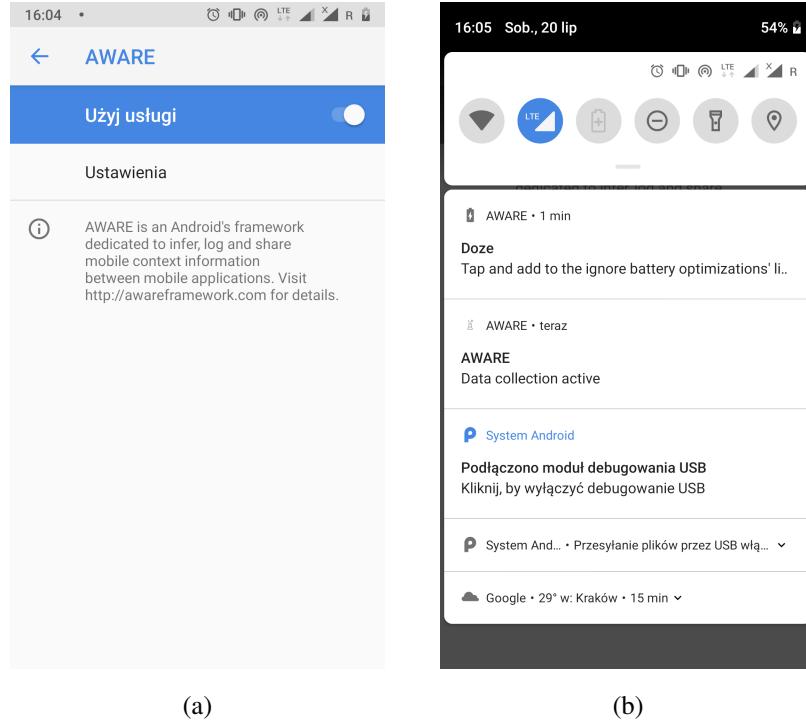
(b)

**Rys. A.9.** Kroki 3 i 4: Widok listy dostępnych pluginów (a) oraz menu aktywacji pluginu (b).

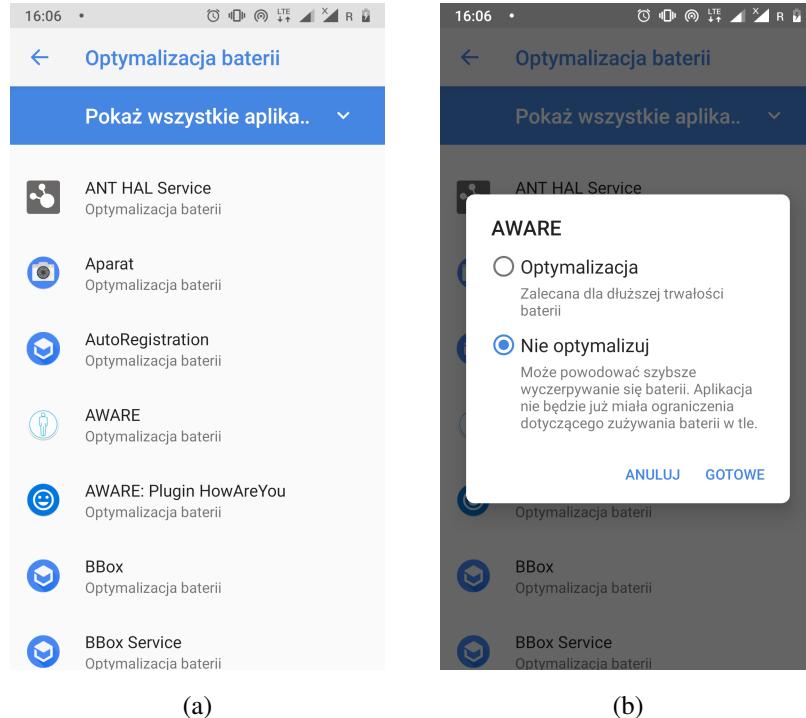
5. Zostaniesz zapytany o zgody na wykorzystanie zasobów telefonu. Zatwierdź odpowiednie zgody.
6. W systemie pojawi się powiadomienie *Please enable AWARE*. Kliknij na obszar powiadomienia. Otworzy się okno *Accessibility Service*.
7. Kliknij na aplikację *AWARE* i przyznaj uprawnienia klikając *Użyj usługi*. Następnie kliknij na plugin *HowAreYou* i przyznaj uprawnienia klikając *Użyj usługi*.
8. Jeżeli pojawi się powiadomienie *Tap and add to the ignore battery optimizations' list*, kliknij na obszar powiadomienia.
9. Wybierz *Pokaż wszystkie aplikacje*.
10. Zarówno na aplikacji *AWARE* jak i na pluginie *HowAreYou* wybierz *Nie Optymalizuj*.



**Rys. A.10.** Kroki 5 i 6: Widok zapytania użytkownika o zgodę (a) oraz powiadomienie *Please enable AWARE* (b).



**Rys. A.11.** Kroki 7 i 8: Widok przyznawania uprawnień *Accessibility Service* (a) oraz powiadomienie *Tap and add to the ignore battery optimizations' list* (b).



**Rys. A.12.** Kroki 9 i 10: Menu ustawień *Optymalizacja Baterii* (a) oraz menu wyboru decyzji (b).



## **B. Kwestionariusze użytkownika**



## Bibliografia

- [1] Pieczonka E. *Emotion detection with mobile device sensors*. 2019.
- [2] Lis A. *Metody interakcji z użytkownikiem przy użyciu urządzeń mobilnych w eksperymentach afetywnych*. 2019.
- [3] Galen Chin-Lun Hung i in. „Predicting negative emotions based on mobile phone usage patterns: an exploratory study”. W: *JMIR research protocols* 5.3 (2016), e160.
- [4] Mi Zhang i Alexander A Sawchuk. „A feature selection-based framework for human activity recognition using wearable multimodal sensors”. W: *Proceedings of the 6th International Conference on Body Area Networks*. ICST (Institute for Computer Sciences, Social-Informatics and ... 2011, s. 92–98.
- [5] Jiangpeng Dai i in. „Mobile phone based drunk driving detection”. W: *2010 4th International Conference on Pervasive Computing Technologies for Healthcare*. IEEE. 2010, s. 1–8.
- [6] Rosalind W Picard. „Affective computing MIT press”. W: *Cambridge, Massachusetts* (1997).
- [7] Grzegorz J Nalepa i in. „Affective Design Patterns in Computer Games. Scrollrunner Case Study.” W: *FedCSIS Communication Papers*. 2017, s. 345–352.
- [8] Strona [www HeaRTDroid](http://heartdroid.re).  
[https://heartdroid.re](http://heartdroid.re).
- [9] Strona [www AWARE: Open-source Context Instrumentation Framework For Everyone](http://awareframework.com).  
[https://awareframework.com](http://awareframework.com).
- [10] Strona [www AI Wiki: Plugin HRTD do AWARE](https://ai.ia.agh.edu.pl/wiki/pl:dydaktyka:wshop:prv:2016:hrtd_plugin:start).  
[https://ai.ia.agh.edu.pl/wiki/pl:dydaktyka:wshop:prv:2016:hrtd\\_plugin:start](https://ai.ia.agh.edu.pl/wiki/pl:dydaktyka:wshop:prv:2016:hrtd_plugin:start).
- [11] Strona [www AI Wiki: HeKatE](https://ai.ia.agh.edu.pl/hekate:start).  
<https://ai.ia.agh.edu.pl/hekate:start>.
- [12] Microsoft Azure: Rozpoznawanie twarzy.  
<https://azure.microsoft.com/pl-pl/services/cognitive-services/face/>.

# Spis rysunków

1.1 Schemat przedstawia jak <i>HowAreYou</i> lub podobna aplikacja może wpływać na uczynienie niezrozumiałych obecnie czynników zrozumiałymi. W niniejszym badaniu nie został uwzględniony sposób wprowadzania tekstu. Podano go tutaj jako jedną z możliwości rozwinięcia pluginu <i>HowAreYou</i> w przyszłości. . . . .	8
2.1 Koło podstawowych emocji (szczęście, podekscytowanie, rozczulenie, strach, gniew i smutek) i ich mimicznych ekspresji. . . . .	11
3.1 Schemat przedstawiający ogólną architekturę rozwiązania. . . . .	13
3.2 Dialog z użytkownikiem w formie pytania o emocje (a) oraz pytania o kolory (b). . . . .	17
3.3 Użytkownik może zdecydować, aby powiadomienie przypominało mu o możliwości rejestracji wizerunku. . . . .	18
3.4 Przejście do menu ustawień z aplikacji klienckiej (a) oraz widok aktywności ustawień (b). . . . .	19
3.5 Widok dalszej części aktywności ustawień (a) oraz log z ostatniego wnioskowania (b). . . . .	20
3.6 Log aplikacji <i>HowAreYou</i> (a) oraz log ostatnich akcji (b). . . . .	21
4.1 Zaawansowany model wnioskujący: część 1. . . . .	25
4.2 Zaawansowany model wnioskujący: część 2. . . . .	25
4.3 Zaawansowany model wnioskujący: tabela QuestionPhotoPossible. . . . .	26
4.4 Zaawansowany model wnioskujący: tabela PhotoPossible. . . . .	26
4.5 Zaawansowany model wnioskujący: tabela QuestionPossible. . . . .	26
4.6 Zaawansowany model wnioskujący: tabela QuestionColorPossible. . . . .	26
4.7 Zaawansowany model wnioskujący: tabela QuestionEmojiPossible. . . . .	27
4.8 Zaawansowany model wnioskujący: tabela PhotoSuccessful. . . . .	27
4.9 Zaawansowany model wnioskujący: tabela howareyouAction. . . . .	28
4.10 Tryb debugowy: widok z fragmentem loga z wnioskowania HMR. . . . .	28
4.11 Uproszczony model wnioskujący: tabela howareyouAction. . . . .	29
4.12 Tryb debugowy: widok z loga z wnioskowania HMR z modelem uproszczonym. . . . .	30

6.1 Zaawansowane <i>widgety</i> zintegrowane z pluginem <i>HowAreYou: Geneva Emotion Wheel</i> (a) oraz Koło Plutchika (b). . . . .	43
A.1 Kroki 1 i 2: Przejście do zaawansowanych ustawień aplikacji. . . . .	46
A.2 Kroki 3 i 4: Aktywność <i>Specjalny dostęp do aplikacji</i> (a) oraz aktywność <i>Instalowanie nieznanych aplikacji</i> (b). . . . .	46
A.3 Krok 5: Aktywność ustawień aplikacji. . . . .	47
A.4 Kroki 1 i 2: Strona www z linkami umożliwiającymi pobranie aplikacji (a) oraz rozporządzły proces pobierania aplikacji (b). . . . .	48
A.5 Kroki 3 i 4: Wybór przycisków <i>Otwórz</i> (a) oraz <i>Instaluj</i> (b). . . . .	48
A.6 Kroki 5 i 7: Wybór przycisku <i>Gotowe</i> (a) oraz wybór wersji pluginu (b). . . . .	49
A.7 Kroki 8 i 9: Wybór przycisków <i>Instaluj</i> (a) oraz <i>Gotowe</i> (b). . . . .	50
A.8 Kroki 1 i 2: Uruchomienie aplikacji <i>AWARE</i> (a) oraz widok zapytania użytkownika o zgodę (b). . . . .	51
A.9 Kroki 3 i 4: Widok listy dostępnych pluginów (a) oraz menu aktywacji pluginu (b). . . .	51
A.10 Kroki 5 i 6: Widok zapytania użytkownika o zgodę (a) oraz powiadomienie <i>Please enable AWARE</i> (b). . . . .	52
A.11 Kroki 7 i 8: Widok przyznawania uprawnień <i>Accessibility Service</i> (a) oraz powiadomienie <i>Tap and add to the ignore battery optimizations' list</i> (b). . . . .	53
A.12 Kroki 9 i 10: Menu ustawień <i>Optymalizacja Baterii</i> (a) oraz menu wyboru decyzji (b). .	53