Projektowanie, zastosowania
i rozwój aplikacji mobilnych

Wyższa Szkoła Biznesu w Dąbrowie Górniczej

# PROJEKTOWANIE, ZASTOSOWANIA I ROZWÓJ APLIKACJI MOBILNYCH

Maciej Rostański
Wojciech Borczyk
Paweł Buchwald
Jakub Duda
Krystian Mączka
Paweł Świtała

Dąbrowa Górnicza 2015

Prace naukowe Wyższej Szkoły Biznesu w Dąbrowie Górniczej

Recenzent: dr hab. inż. Bartłomiej Zieliński

Korekta tekstów:
Laura Maria Ryndak

Projekt okładki:
Digitalpress Lidia Jaworska

ISBN 978-83-64927-74-4

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Projekt jest współfinansowany ze środków Unii Europejskiej w ramach Europejskiego Funduszu Społecznego z Programu Operacyjnego Kapitał Ludzki, Działania 4.3 – Wzmocnienie potencjału dydaktycznego uczelni w obszarach kluczowych w kontekście celów strategii Europa 2020. „Nowoczesna wiedza = nowoczesna gospodarka – program rozwoju potencjału Wyższej Szkoły Biznesu w Dąbrowie Górniczej" Publikacja dystrybuowana bezpłatnie.

Wydawca:
Wyższa Szkoła Biznesu w Dąbrowie Górniczej,
ul. Cieplaka 1c, 41-300 Dąbrowa Górnicza
tel. (32) 262 28 05
e-mail: info@wsb.edu.pl, www.wsb.edu.pl



Skład i druk:
Digitalpress Lidia Jaworska, ul. Kasztanowa 3/9, 42-500 Będzin

# SPIS TREŚCI

# Wstęp

Publikacja niniejsza jest efektem doświadczeń i pracy dydaktyczno-naukowej osób związanych z prowadzeniem specjalności „Aplikacje Mobilne" na kierunku Informatyka w Wyższej Szkole Biznesu w Dąbrowie Górniczej. Jak każą dobre praktyki, zajęcia i prace badawcze prowadzone są z udziałem instytucji gospodarczych – firm i środowiska przemysłowego, więc także zespół osób prowadzących, jak i autorów niniejszej książki stanowią zarówno naukowcy i wykładowcy akademiccy, jak i praktycy, realizujący się w działalności informatycznej związanej z aplikacjami mobilnymi.

Celem usystematyzowania, w niniejszej książce rozważania nad powstawaniem oprogramowania mobilnego rozpoczęto od wprowadzenia do języka i platformy programistycznej języka Java. Jako podstawową platformę wybrano Android, i nieprzypadkowo większość przykładów odnosi się do tego systemu, jego implementacji i aplikacji uruchamianych w tym właśnie środowisku. Kolejne poruszane zagadnienia, od architektury aplikacji mobilnych, przez możliwe do wykorzystania interfejsy, po problematykę bezpieczeństwa środowisk mobilnych, rozwijają coraz bardziej aspekty możliwości oraz wyzwań stojących przed projektującymi aplikacje dla urządzeń mobilnych. Celowo występuje więc tutaj ewolucja treści od rozdziałów o charakterze dydaktycznym, wprowadzającym do przeglądowego, czy naukowego wręcz spojrzenia na aspekty poruszane w pracy.

Mamy nadzieję, że niniejsza praca będzie interesująca i przydatna dla osób, które zajmują się – bądź zamierzają się zająć – wdrażaniem i rozwojem aplikacji mobilnych.

*Autorzy*

# 1

# PROGRAMOWANIE APLIKACJI MOBILNYCH PRZY UŻYCIU JĘZYKA JAVA

## 1.1. Wprowadzenie do języka Java

Pierwsza część niniejszego rozdziału zawiera wprowadzenie do języka i platformy Java. Należy je traktować jako niezbędną podstawę do zrozumienia najważniejszych elementów programowania urządzeń mobilnych w tym języku. Wprowadzenie do systemu Android i tworzenia nań aplikacji w języku Java zawarte jest w drugiej części opracowania.

### 1.1.1. Czym jest Java

Java jest obiektowym językiem programowania ogólnego zastosowania, zaprojektowanym by działać niezależnie od platformy sprzętowej (wieloplatformowość) – w idealnej sytuacji oznacza to, że programy napisane w tym języku można uruchamiać na różnych platformach i w różnych systemach operacyjnych bez konieczności rekompilacji kodu. Javą nazywa się również całą platformę służącą do uruchamiania napisanych w tym języku programów – pełni ona rolę pośrednika między programem a sprzętem i systemem operacyjnym.

Prace nad językiem Java rozpoczęto w roku 1991 w firmie Sun Microsystems. Za „ojca" języka powszechnie uznaje się Jamesa Goslinga, ale w pierwotnym zespole pracowali również Mike Sheridan i Patrick Naughton (pierwotnie zespół twórców podstaw języka *java* utworzył język programowania zwany OAK, który jest uznawany za pierwszą wersję Javy). James Gosling pracował nad Javą aż do przejęcia przez Oracle, co skłoniło go do odejścia w 2010 roku.

Początkowo głównym zastosowaniem języka Java miały być urządzenia gospodarstwa domowego, magnetowidy, telewizory czy telefony zbudowane w oparciu o różnorakie architektury sprzętowe. Okazało się, że Java znakomicie sprawdza się w zastosowaniach związanych z inną nowo powstałą wówczas technologią – Internetem. Już w chwili wydania wersji 1.0 wirtualna maszyna Javy była wbudowana w przeglądarkę Netscape Navigator (od której swoje korzenie wywodzi Mozilla Firefox). Właśnie zastosowaniom sieciowym (początkowo po stronie klienta – w formie osadzanych na stronie internetowej appletów) a następnie po stronie serwera (do tych zastosowań powstała edycja JavaEE, o której więcej napisano w dalszej części niniejszego opracowania) Java zawdzięcza swoją popularność.

### 1.1.2. Założenia

Projektanci założyli, że technologia Java musi pozwalać na tworzenie *bezpiecznych, efektywnych i wydajnych* aplikacji pracujących na *wielu platformach* w *niejednorodnych, rozproszonych* sieciach[1]. To prowadziło do sformułowania najważniejszych celów projektu:

#### „prosty, zorientowany obiektowo i znajomy"

W odróżnieniu od uznawanych wówczas za standard języków Java ma być prosta w użyciu, bez konieczności długotrwałej nauki, zgodny z aktualnymi praktykami. Podstawy powinny być szybko przyswajalne, aby zapewnić jak największą produktywność programisty.

Java od podstaw jest zorientowana obiektowo. Coś, co dziś wydaje się oczywiste, wówczas uznawano za istotną nowość, w szczególności w kontraście do będącego standardem języka C.

Pomysł wykorzystania języka C++ jako języka implementacji Javy zarzucono, jednak składnia jest do niego bardzo zbliżona, co powoduje, że wiele elementów jest znajomych dla programistów pracujących z językiem C++.

#### „wydajny i bezpieczny"

Java z założenia ma być niezawodnym językiem, promującym praktyki wspierające tworzenie niezawodnego oprogramowania. Model zarządzania pamięcią jest bardzo prosty (operator new), nie ma dostępu ani możliwości manipulacji wskaźnikami, a za zwalnianie pamięci odpowiada automatyczny proces *garbage collection*.

Ponieważ Java ma pracować w środowiskach rozproszonych, bardzo istotnym aspektem jest bezpieczeństwo – służą temu mechanizmy wbudowane w sam język, jak i w środowisko uruchomieniowe (ang. *runtime system*)[2].

#### „niezależny od architektury i przenośny"

[1] http://www.oracle.com/technetwork/java/intro-141325.html

[2] Z perspektywy czasu właśnie bezpieczeństwo środowiska uruchomieniowego Javy okazało się jego piętą achillesową.

Java wspiera możliwość tworzenia programów pracujących i komunikujących się w niejednorodnej sieci, a co za tym idzie uruchamianych na różnym sprzęcie i różnych systemach operacyjnych (z dzisiejszej perspektywy może to być na przykład serwer pracujący pod kontrolą systemu Linux i telefon z systemem Android[3]). Aby to osiągnąć, kompilator języka Java produkuje nie tradycyjny, zależny od sprzętu i systemu operacyjnego kod binarny (będący formatem zapisu języka maszynowego – assemblera) ale tzw. **bytecode** – niezależny od sprzętu format pośredni, zaprojektowany by przenosić ten sam kod na różne platformy. Za jego wykonanie na poszczególnych urządzeniach i systemach odpowiada platforma **maszyna wirtualna** języka Java (Java VM). Z perspektywy czasu ten aspekt platformy Java okazał się szczególnie istotny.

Tę niezależność od architektury firma Sun określiła skrótem WORA (*Write Once Run Anywhere*).

### „wysoko wydajny"

*Garbage collector* pracuje w tle zapewniając pewny dostęp do pamięci, a wymagające tego, intensywne obliczeniowo fragmenty kodu (tzw. hot spots) mogą zostać przetłumaczone na język maszynowy danej platformy, by zapewnić ich szybsze wykonanie. W tym przypadku autorom specyfikacji nie do końca udało się zrealizować powzięte założenie i choć dziś język Java w wielu zastosowaniach i w testach syntetycznych jest uznawany za rozwiązanie najlepsze, lub tylko nieznacznie ustępujące językowi C++, to właśnie wydajność była przez długi czas słabą stroną języka Java.

### „interpretowany, wielowątkowy, dynamiczny"

Interpreter może wykonywać kod bajtowy Javy bezpośrednio na każdej maszynie, na którą przeniesiono sam interpreter i środowisko uruchomieniowe, co prowadzi do znacznej oszczędności czasu w procesie rozwoju oprogramowania w porównaniu z tradycyjną kompilacją dla poszczególnych platform. Wielowątkowość jest wspierana zarówno na poziomie języka, jak i środowiska uruchomieniowego.

Nowe klasy i moduły mogą być linkowane dynamicznie (również ze źródeł zewnętrznych – sieć) w miarę potrzeb.

Komentując ten punkt, warto zaznaczyć, że idea interpretacji języka nie sprawdziła się w przypadku aplikacji wymagających dużej wydajności i skalowalności – do samej Javy w późniejszych edycjach wprowadzono technologię HotSpot umożliwiającą tłumaczenie intensywnych obliczeniowo fragmentów kodu do języka natywnego, na konkurencyjny dla Javy system .Net co umożliwia kompilację całych programów do natywnego środowiska. Z kolei wielowątkowość jest dziś, w epoce procesorów wielordzeniowych „oczywistą oczywistością". Podobnie dynamiczne linkowanie klas i bibliotek stało się powszechne, z wyjątkiem systemów, które z przyczyn bezpieczeństwa lub biznesowych tego nie umożliwiają (np. system iOS uniemożliwia dynamiczne ładowanie bibliotek).

[3] Należy zauważyć, iż twórcy systemu Android nie zastosowali zgodnej ze specyfikacją Javy maszyny JVM, ale wprowadzili unikatową maszynę uruchomieniową Dalvik, co powoduje pewne ograniczenia języka.

### 1.1.3. Najważniejsze pojęcia związane z językiem i platformą

#### Java Maszyna wirtualna

Maszyna wirtualna Javy (Java VM) jak sama nazwa wskazuje jest abstrakcyjnym komputerem[4] odpowiadającym za wykonanie każdego programu skompilowanego do kodu bajtowego (bytecode) – zapewnia warstwę abstrakcji między programem Java a komputerem. Innymi słowy – program w języku Java nie wykonuje się bezpośrednio na komputerze, ale we wspomnianej maszynie wirtualnej, która izoluje go od właściwego sprzętu i systemu operacyjnego. Takie rozwiązanie ma kilka istotnych zalet, z których najważniejsze to bezpieczeństwo i przenośność oprogramowania.

#### Bezpieczeństwo

Maszyna wirtualna w teorii zapewnia całkowitą izolację programu od właściwego sprzętu i oprogramowania, zapewniając, że w przypadku nieprawidłowego lub złośliwego (*malware*, koń trojański) działania program nie będzie w stanie zaszkodzić reszcie systemu.

#### Przenośność

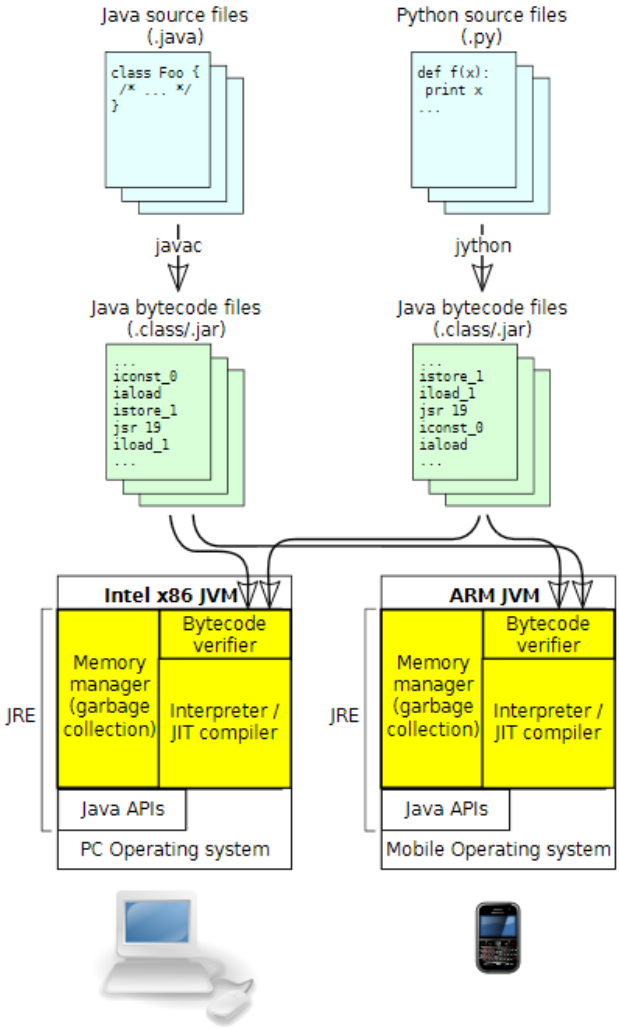
Raz napisany program można uruchomić na każdym sprzęcie i systemie operacyjnym, na który przeniesiono maszynę wirtualną Javy i jej środowisko uruchomieniowe (runtime). Dzięki temu ten sam program można uruchomić np. pod systemem Windows pracującym na urządzeniu w architekturze Intel jak i pod Linuxem w architekturze ARM.

**Uwaga!** W przypadku systemu Android wspomniana przenośność jest mocno ograniczona – ze względu na inny zestaw klas, implementacji maszyny wirtualnej i zmieniony proces kompilacji programów powstałych dla Java Runtime Environment (standardowa edycja desktop) nie można uruchamiać pod systemem Android i odwrotnie.

Pierwotnie do kodu bajtowego można było kompilować tylko programy napisane w języku Java, jednak od wersji 7 jest możliwe również kompilowanie innych języków, np. Pythona (rys. 1)

[4] Zdefiniowanym według specyfikacji: http://docs.oracle.com/javase/specs/jvms/se7/html/

Rysunek 1. Kompilowanie programów napisanych w różnych językach.




Źródło: Wikimedia commons, licencja Creative Commons CC0 1.0 Universal Public Domain Dedication

### Applet

Appletami nazywa się programy w języku Java osadzane na stronach internetowych i uruchamiane za pomocą technologii wtyczek (plugin).

### API

Ang. Application Programming Interface – interfejs programowania aplikacji. Najogólniej rzecz biorąc jest to zbiór reguł i opisów definiujących

komunikację między środowiskami i programami. Mianem API często określa się klasy i udostępniane przez nie metody pozwalające realizować określone zadania w danym środowisku programistycznym, tego znaczenia używa się mówiąc o API języka Java.

### Javadoc

Jest to technologia umożliwiająca tworzenie dokumentacji oprogramowania bezpośrednio w kodzie źródłowym programu – pozwalająca wygenerować kompletną dokumentację programu na podstawie jego komentarzy przygotowanych w określonym formacie. Cała dokumentacja systemowej biblioteki klas Java powstała właśnie w tej technologii.

### Garbage collection

Podobnie jak dla wielu innych pojęć z dziedziny IT dla tego zagadnienia również brakuje dobrej nazwy w języku polskim, choć czasem używa się sformułowania „odśmiecanie pamięci."

Jest to metoda automatycznego zarządzania pamięcią, w której za zwalnianie zaalokowanej pamięci odpowiada nie programista lecz specjalny program zarządzający, tzw. garbage collector. Tak więc o ile np. w języku C++ każda instrukcja alokująca new powinna mieć odpowiadającą jej instrukcję delete (w języku C są to odpowiednio malloc i free) w języku Java zwalnianie pamięci odbywa się automatycznie.

Historycznie operacja „odśmiecania" była zawsze relatywnie droga obliczeniowo – wczesne implementacje mogły zatrzymać wykonywanie właściwego programu lub jego wątków na całe sekundy. I choć od tego czasu technologia *garbage collection* poczyniła olbrzymie postępy, wciąż należy pamiętać o unikaniu zbędnej alokacji pamięci (prowadzącej do częstego uruchamiania *garbage collectora*) – w szczególności na dysponujących ograniczonymi zasobami urządzeniach mobilnych i w przypadku wymagających płynnej wydajnej i niezakłóconej pracy aplikacji (np. gry).

### Bytecode

Bytecode (albo też kod bajtowy) to zestaw instrukcji używany do reprezentacji programów przeznaczonych do wykonania w maszynie wirtualnej (interpreterze).

Przykładowo dla kodu źródłowego w Javie:

```
for (int i = 2; i < 1000; i++) {
    for (int j = 2; j < i; j++) {
        if (i % j == 0)
            continue outer;
    }
    System.out.println (i);
}
```

Kompilator generuje następujący kod bajtowy:

```
     0:   iconst_2
      1:   istore_1
      2:   iload_1
      3:   sipush  1000
      6:   if_icmpge       44
      9:   iconst_2
      10:  istore_2
      11:  iload_2
      12:  iload_1
      13:  if_icmpge       31
      16:  iload_1
      17:  iload_2
      18:  irem
      19:  ifne    25
      22:  goto    38
      25:  iinc    2, 1
      28:  goto    11
      31:  getstatic       #84; //Field
Java/lang/System.out:LJava/io/PrintStream;
      34:  iload_1
      35:  invokevirtual   #85; //Method
Java/io/PrintStream.println:(I)V
      38:  iinc    1, 1
      41:  goto    2
      44:  return
```

W pewnym uproszczeniu można go porównać do assemblera dla maszyny wirtualnej.

Pliki źródłowe języka Java są przechowywane w plikach z rozszerzeniem *.Java, pliki kodu bajtowego (zapisane w specjalnym formacie[5]) w plikach z rozszerzeniem .*.class. Ponownie upraszczając i porównując do języka C++ pliki Java odpowiadają plikom *.cpp, a pliki class skompilowanym plikom exe.

**Uwaga!** W przypadku języka Java inaczej niż w językach C i C++ nie stosuje się plików nagłówkowych (*.h), wszystkie deklaracje zawarte są bezpośrednio w plikach *.Java.

Jeden plik class odpowiada jednej klasie języka Java.

## JAR

Po angielsku oznacza po prostu słoik (pojemnik) – i to właśnie naczynie widzimy z reguły na przedstawiających go ikonach, oficjalnie jest to skrót od Java Archive. Jest archiwum zip (zawartość można podejrzeć zmieniając rozszerzenie pliku), służący do przechowywania składających się na program pli-

[5] Opisanym w JSR 202, https://www.jcp.org/en/jsr/detail?id=202

ków klas oraz towarzyszących im zasobów (np. ikon i grafik) oraz metadanych (plik manifest, opisany szerzej w dalszej części niniejszego opracowania.

### JIT

Just-in-time to stosowana między innymi w środowisku Javy technologia dynamicznej kompilacji kodu w trakcie jego wykonywania. Pierwotnie kod Javy miał być w całości interpretowany, jednak wobec istotnych ograniczeń w wydajności wprowadzono technikę hybrydową – najczęściej wykorzystywane i najbardziej wymagające obliczeniowo fragmenty programu (w kodzie bajtowym) są w trakcie wykonywania (interpretacji) programu kompilowane do kodu natywnego. Implementacja kompilatora JIT stosowana w Java Standard Edition nosi nazwę **Hotspot**.

### IDE

Z ang. *Integrated Development Environment* – czyli środowisko programistyczne wykorzystywane do tworzenia oprogramowania. Przykładem IDE jest np. Visual Studio firmy Microsoft. Dla języka Java istnieje kilka bardzo dobrych środowisk, np. netbeans (http://www.netbeans.org) Eclipse (http://www.eclipse.org) albo IntelliJ (https://www.jetbrains.com/idea/). Umieszczone na końcu książki ćwiczenia opierają się na wykorzystaniu będącego standardem dla aplikacji Android środowiska Eclipse. Jednak ukazała się już pierwsza wersja stabilna opracowanego przez Google na bazie IntelliJ Android Studio, należy się więc spodziewać, że w przyszłości to środowisko stanie się standardem dla rozwoju aplikacji systemu Android.

### JRE

Ten skrót oznacza Java Runtime Environment – czyli środowisko uruchomieniowe Java. Pakiet JRE można pobrać ze stron firmy Oracle za darmo, po zaakceptowaniu licencji pod adresem *http://www.oracle.com/technetwork/Java/index.html*

JRE zawiera maszynę wirtualną, bibliotekę klas systemowych, wtyczkę (plugin) dla przeglądarek internetowych oraz inne narzędzia niezbędne do uruchamiania programów Java. JRE dostępne jest w wersji 32 i 64 bitowej dla systemów Windows, Linux, Solaris oraz OSX[6]. Jeśli system operacyjny na to pozwala, zalecane jest stosowanie wersji 64 bitowej.

### JDK

JDK to po prostu Java Development Kit – zestaw narzędzi niezbędnych do tworzenia i uruchamiania programów w Javie – JDK zawiera w sobie JRE. W innych językach programowania można spotkać się ze skrótem **SDK** – Software Development Kit, stąd czasem można trafić na określenie Java SDK w odniesieniu do JDK. Podobnie jak w przypadku JRE, niezbędny pakiet można pobrać ze stron Oracle[7].

[6] W listopadzie 2015, http://www.oracle.com/technetwork/Java/Javase/downloads/jre8-downloads-2133155.html
[7] W listopadzie 2015, http://www.oracle.com/technetwork/Java/Javase/downloads/jdk8-downloads-2133151.html

### 1.1.4. Edycje Platformy Java

W czasie, gdy rozwojem platformy Java zajmowała się firma Sun, zdefiniowano cztery podstawowe edycje języka Java:

#### JavaCard

Najmniejsza i przeznaczona dla najbardziej ograniczonego środowiska edycja języka, przeznaczona do pracy na kartach SIM oraz kartach bankomatowych. Jej celem jest dostarczenie przenośnego i bezpiecznego środowiska obliczeniowego dla tzw. „smart cards".

Ostatnia wersja specyfikacji pochodzi z 2011 roku, co pozwala przypuszczać, że obecnie ta wersja nie jest aktywnie rozwijana.

#### Java Micro Edition (Java ME)

Edycja Javy przeznaczona do pracy na urządzeniach mobilnych oraz systemach wbudowanych (ang. *embedded*). W praktyce oznacza to zastosowania przemysłowe, urządzenia telewizyjne (set-top boxes) oraz (najpopularniejsze) telefony komórkowe (tzw. *feature phones*). Jeszcze parę lat temu Java ME była standardem tworzenia oprogramowania dla telefonów komórkowych, obsługiwanym przez wszystkich największych (wówczas) dostawców urządzeń. Sytuacja diametralnie zmieniła się wraz ze wzrostem popularności smartfonów pracujących pod kontrolą systemów iOS i Android – obecnie Java ME coraz bardziej odchodzi w zapomnienie, choć wciąż jest standardem dla popularnych w krajach rozwijających się (np. kraje Afryki, Indie) tzw. „feature phones".

**Uwaga!** Java Micro Edition, choć z pozoru podobnego zastosowania, nie ma NIC wspólnego z implementacją Javy dla systemu Android.

#### Java Enterprise Edition (Java EE)

Java Enterprise Edition powstała z myślą o zastosowaniach biznesowych – do pewnego momentu była dominującym rozwiązaniem w aplikacjach serwerowych, biznesowych i bankowych. I choć od pewnego czasu istnieje silna konkurencja w postaci platformy .Net i popularnych w zastosowaniach chmurowych technologii NoSQL czy też node.js Java EE wciąż jest i jeszcze długo pozostanie silną technologią na rynku rozwiązań biznesowych.

#### Java Standard Edition (JavaSE)

Jak sama nazwa wskazuje jest to standardowa edycja języka, przeznaczona do wykonywania aplikacji tzw. „desktopowych" – a więc na komputerach i laptopach. Jest też podstawą większości narzędzi korzystających z języka Java, w tym również środowiska developerskiego dla systemu Android. Swego czasu jednym z najpopularniejszych zastosowań Java SE – były tzw. aplety – napisane w języku Java programy osadzane w stronach internetowych, które mogły wykonywać się w przeglądarce. Jednak wobec wzrostu popularności technologii zbiorczo nazywanych HTML5 oraz serii krytycznych błędów bezpieczeństwa związanych z apletami Javy ta technologia jest w całkowitym

odwrocie, a wiele przeglądarek (np. Firefox) całkowicie blokuje wykonywanie appletów.

#### JavaFX

Technologia JavaFX powstała z myślą o tworzeniu tzw. „rich internet application" (RIA)[8]. Pierwotnie JavaFX miała zastąpić starzejące się applety i stać się prawdziwą konkurencją dla Adobe Flash. Miała też wreszcie stworzyć nowoczesne i wygodne API tworzenia interfejsów użytkownika i obsług multimediów dla środowiska Java (istniejące technologie nie wytrzymywały konkurencji i próby czasu). Jednak wobec odwrotu od stosowania wtyczek na stronach internetowych JavaFX skupiono się w całości na tym ostatnim zadaniu. JavaFX **nie jest** oddzielną edycją Javy, od wersji Java 7 stanowi integralną część Java Standard Edition, a od wersji 8 posiada taką samą numerację (stąd przeskok z JavaFX 2.2 do JavaFX 8.0).

#### Wersje

Z przyczyn historycznych w obiegu jest co najmniej kilka równoległych metod nazywania wersji Javy. I choć oficjalnie od pewnego czasu używa się nazewnictwa Java 7, Java 8 itd., programista prędzej czy później zetknie się z nomenklaturą obowiązującą wcześniej.

Pierwsza edycja Javy nosiła oznaczenie 1.0. Kolejną nazwano 1.1. Następną wersję 1.2 ze względu na bardzo duże zmiany nazwano Javą 2, stąd przez pewien czas obowiązywało nazewnictwo „Java 2 Standard Edition 1.3". Po pewnym czasie nazwa Java 2 została zarzucona, a cała Java przejęła nazwę zgodną wcześniej z częścią dziesiętną wersji. I tak, Java 1.7.0 oznacza oficjalnie Javę 7, a „1.7.0_71" (czasem oznaczany jako Java 7u71) oznacza Javę 7, uaktualnienie nr 71[9] (dlaczego nie 1.7.71?)

Najnowszą dostępną na rynku w chwili powstawania niniejszego opracowania jest Java 8 (Java 1.8.0). Wersja minimalna wymagana do przygotowywania oprogramowania dla systemu Android to wersja 7.0 (co nie znaczy, że Java w tym systemie jest zgodna z Javą 7!)[10].

Ze względu na wspomniane już luki w bezpieczeństwie zalecane jest regularne aktualizowanie zainstalowanych wersji Javy.

### 1.1.5. Java, .Net a Javascript

Najważniejszą konkurencją dla języka i platformy Java jest opracowany przez Microsoft język C# działający na platformie .Net. Środowisko .Net przedstawione przez Microsoft w roku 2000, jest w wielu przypadkach oparte na założeniach bardzo podobnych do tych, które przyjęła firma Sun, jednak

[8] To określenie powstało z myślą o stronach internetowych tworzonych w całości w technologii Flash (a więc za pomocą technologii wtyczek osadzających w stronach internetowych kod natywny), obecnie obejmuje większość aplikacji internetowych tworzonych za pomocą technologii określanych jako HTML 5: http://pl.wikipedia.org/wiki/Rich_Internet_Application

[9] Można polecić dokument opisujący zasady numerowania uaktualnień: http://www.oracle.com/technetwork/java/javase/overview/jdk-version-number-scheme-1918258.html

[10] Jeszcze niedawno było to 6.0.

będąc dużo młodszym unika wielu problemów, które trapiły szczególnie wczesne wersje platformy Java.

Pierwotnie istotną różnicą było przywiązanie .Net do systemu operacyjnego Windows (wieloplatformowość ograniczała się do różnych wersji tego systemu) podczas gdy Java od początku pracowała pod kontrolą Windows oraz środowisk Unixowych (w tym Solarisa i Linuxa). Co więcej Java uznawana była za system otwarty w zestawieniu z zastrzeżonymi (*proprietary*) technologiami Microsoftu. Z kolei język C# wprowadził wiele wygodnych dla programistów rozwiązań (np. *autoboxing*), które wówczas wyprzedzały możliwości Javy, a sama platforma .Net była od początku przygotowana na obsługę wielu języków programowania (w tym Visual Basic, C++ a nawet dialektu języka Java zgodnego z wersją 1.1). Obecnie sytuacja całkowicie się zmieniła – od kilku lat na rynku dostępna jest otwarta (open-source), wieloplatformowa implementacja .Net o nazwie Mono, a w listopadzie 2014 roku sama platforma .Net stała się projektem open-source. Natomiast platforma Java wprowadziła możliwość kompilacji innych niż Java języków, a do składni języka trafiło wiele rozwiązań zapoczątkowanych przez C#.

Pomimo pewnych podobieństw składni (w obu przypadkach wywodzi się ona z języka C), Java i Javascript nie są ze sobą związane technicznie – podobieństwo nazwy to chwyt marketingowy firmy Netscape, która chciała wypromować własną technologię korzystając z popularności języka Java.

Java jest statycznym językiem opartym o klasy, podczas gdy Javascript to dynamiczny język skryptowy oparty o prototypy. Javascript jest najczęściej wykorzystywany w przeglądarkach internetowych po stronie klienta (wchodzi w skład grupy technologii sieciowych określanych jako HTML5). Co ciekawe, dzięki bardzo wydajnej technologii V8 wykorzystywanej do obsługi JavaScript w przeglądarce Chrome ten język stał się bardzo popularny w rozwiązaniach serwerowych (wykorzystuje go między innymi technologia node.js[11]).

### 1.1.6. Bezpieczeństwo

Pierwotnie jedną z największych zalet języka Java były wbudowane w sam język oraz w platformę mechanizmy bezpieczeństwa. Piętą achillesową systemu okazała się jednak napisana w językach natywnych (C++) maszyna wirtualna Javy, która okazała się całkowicie nieprzygotowana na różnego rodzaju zagrożenia. W latach 2012 i 2013 wykryto całą serię luk bezpieczeństwa, które w wielu przypadkach pozwoliły atakującym na przejęcie kontroli nad komputerem ofiary poprzez uruchamiane na stronie internetowej aplety języka Java. W rezultacie tego, nastąpił spotęgowany przez często spóźnione reakcje firmy Oracle istotny spadek zaufania do całej platformy Java. W połączeniu z ogólną tendencją do zastępowania tzw. wtyczek na stronach www (np. Adobe Flash) przez mechanizmy HTML5, applety Javy stały się technologią wymierającą.

[11] Jest to platforma oparta na środowisku uruchomieniowym JavaScript przeglądarki Chrome, służąca budowaniu szybkich i skalowalnych aplikacji sieciowych: http://nodejs.org/

Należy tutaj wyraźnie zaznaczyć, że większość problemów dotyczyła „Javy w przeglądarce" a firma Oracle opóźniła premierę wersji 8, by poprawić bezpieczeństwo platformy, więc nie należy ich traktować jako dyskwalifikujących język w zastosowaniach serwerowych czy desktopowych.

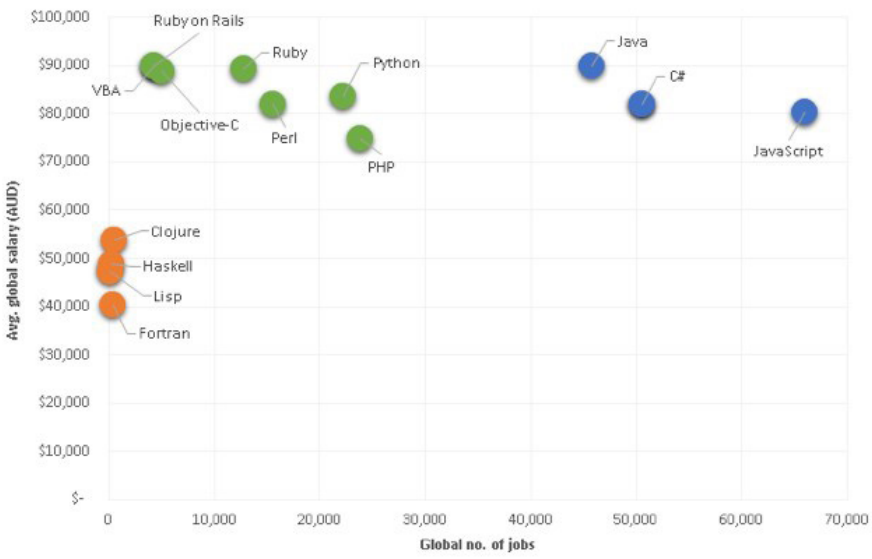
### 1.1.7. Wydajność

Historycznie Java rzeczywiście była postrzegana jako język mało wydajny, a sama platforma jako wymagająca dużej ilości zasobów. Obecnie Java w wielu przypadkach osiąga wydajność porównywalną z językiem C++ (w szczególności w testach numerycznych). Wciąż jednak zużycie pamięci i czas startu aplikacją są zauważalnie większe niż w przypadku programów natywnych. Ogólnie można przyjąć, że programy napisane w języku Java będą w testach:

- porównywalne lub nieco wolniejsze od języków kompilowanych (C lub C++),
- porównywalne do innych języków używających kompilacji Just-In-Time (C#),
- wolniejsze od języków skryptowych, nie posiadających w standardzie takiej możliwości kompilacji (np. Perl, czy Python).

### 1.1.8. Uwarunkowania biznesowe

Rysunek 2. Popularność języków programowania na rynku pracy specjalistów.




Źródło: https://gooroo.io/GoorooTHINK/Article/16191/Which-language-wins-in-terms-of-salaryde-mand-July-2014/14105

Na rys. 2 przedstawiono zestawienie - ilość dostępnych miejsc pracy i średnie roczne wynagrodzenie dla programistów poszczególnych języków. Są to wyniki przeprowadzonego w 2014 roku (okres styczeń – lipiec) badania 1.5 miliona ogłoszeń o pracy dla programistów, na terenie Stanów Zjednoczonych, Wielkiej Brytanii i Australii. Kolorem niebieskim oznaczono języki uznane za „liderów" – Javę, C# oraz JavaScript. Jak widać na umiejętność programowania w tych językach istnieje duże zapotrzebowanie (ilość miejsc pracy), poparty wysoki wynagrodzeniami (średnia wysokość). Planując przyszła karierę zawodową warto zwrócić uwagę na wysoką (utrzymującą się pomimo wyraźnego spowolnienia rozwoju języka w ostatnich latach) pozycję języka Java.

## 1.2. Wprowadzenie do systemu Android

Android jest mobilnym systemem operacyjnym opartym na jądrze Linuxa (wyższe warstwy są już specyficzne dla tego systemu) dystrybuowanym na zasadach open-source. Początkowo rozwijany przez niezależną firmę Android Inc. Jednak jeszcze przed wydaniem pierwszej wersji została ona przejęta przez Google. Został zaprojektowany z myślą o urządzeniach sterowanych przez ekrany dotykowe. Pierwotnie były to smartfony, od wersji 3.0 również tablety. Obecnie istnieją specjalne edycje:

- Android TV – jak sama nazwa wskazuje dla telewizorów,
- Android Car – dla systemów samochodowych,
- Android Wear – dla tzw. smartwatches i podobnych.

Z myślą o krajach rozwijających się opracowano standard Android One (który można nazwać referencyjną implementacją oprogramowania i sprzętu). Istnieje też (zapomniana nieco) inicjatywa Android@home przeznaczona dla rynku automatyzacji urządzeń domowych (ang. *home automation*, a ogólniej Internet of Things).

We wrześniu 2013 roku Google poinformował, że na świecie aktywowano miliard urządzeń pod kontrolą Androida (dla iOS ta wartość wynosi 470 mln)

### 1.2.1. Historia

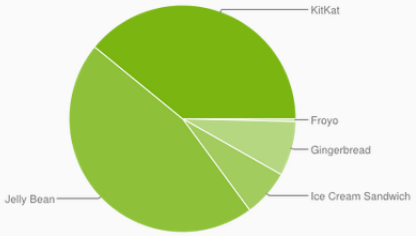
Pierwsze wersje systemu pojawiły się na rynku w roku 2008, od datowanej na rok następny wersji 1.5 noszą nazwy kodowe związane z „łakociami" – Mrożony Jogurt (Froyo), Donut (Pączek) itd.

Wersja 3.0 („Plaster miodu") była pierwszą przeznaczoną dla tabletów (i tylko dla tabletów), dopiero od wersji 4.0 zarówno smartfony jak i tablety są obsługiwane przez ten sam system. Najnowsza, wydana pod koniec 2014 roku wersja 5.0 („Lizak") przynosi znaczące zmiany, jednak ze względu na trudną dostępność nie zdobyła jeszcze znaczącej pozycji na rynku.

### 1.2.2. Popularność poszczególnych wersji

Rysunek 3. Popularność wersji systemu Android.

| Version | Codename | API | Distribution |
|---|---|---|---|
| 2.2 | Froyo | 8 | 0.4% |
| 2.3.3 - 2.3.7 | Gingerbread | 10 | 7.8% |
| 4.0.3 - 4.0.4 | Ice Cream Sandwich | 15 | 6.7% |
| 4.1.x | Jelly Bean | 16 | 19.2% |
| 4.2.x | | 17 | 20.3% |
| 4.3 | | 18 | 6.5% |
| 4.4 | KitKat | 19 | 39.1% |




Źródło: Google.

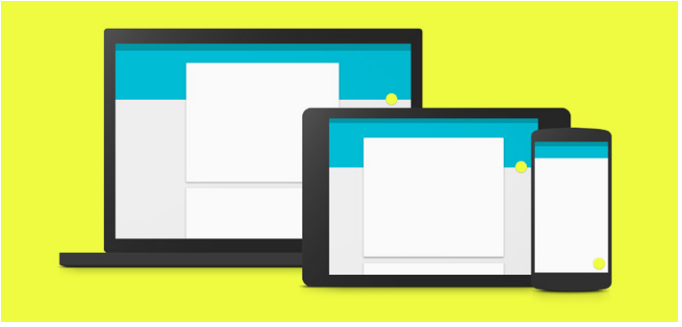
Według danych opublikowanych przez firmę Google w styczniu, obecnie najpopularniejsze na rynku edycje Androida to 4.1 (Jelly Bean) i 4.4 (KitKat). Jednak wciąż wyraźnie obecne są wersje 2.2 oraz 2.3, stąd większość programów obecnych na rynku musi wspierać również te wersje (warto o tym pamiętać realizując komercyjne projekty). Co ciekawe, najnowsza wersja systemu (Lollipop) nie jest nawet obecna na wykresie, ponieważ jej popularność nie przekroczyła 0.1%.

### 1.2.3. Co nowego w Android 5.0 (Lollipop)

Wydana pod koniec 2014 roku wersja 5.0 przynosi wiele istotnych zmian, w pełni usprawiedliwiając tak dużą zmianę w numeracji:

**Material design** – nowa filozofia projektowania (*design language*) interfejsów użytkownika, której celem jest zapewnienie nowoczesnego estetycznego, i opartego na zasadach uniwersalnych dla różnych urządzeń interfejsu użytkownika[12].

Rysunek 4. Koncepcja responsive design – interfejsu użytkownika dopasowanego do urządzenia



[12] Szczegółowe informacje można znaleźć pod adresem: http://www.google.com/design/spec/material-design/introduction.html

**Lepszy wygląd** – bazując na material design poprawiono wygląd całego interfejsu użytkownika.

**Współpraca z Chrome OS** – w wersji 5.0 firma Google poświęciła sporo uwagi poprawie współpracy między systemami Android i Chrome OS.

**Poprawiona wydajność baterii** – wersja 5.0 ma przynieść użytkownikom lepsze wykorzystanie baterii.

**Nowa środowisko uruchomieniowe** – od wersji 5.0 do historii przechodzi wcześniejsza maszyna wirtualna Androida (Dalvik VM), zastąpiona przez nowe środowisko uruchomieniowe ART. (Android RunTime)[13].

Oczywiście zmian jest o wiele więcej, szczegółowe informacje można znaleźć w Internecie, np. na stronie http://www.android.com/versions/lollipop-5-0/

### 1.2.4. Inne systemy operacyjne dla urządzeń mobilnych

Rynek urządzeń mobilnych rozwija się bardzo szybko, co owocuje dużą ilością rozwiązań w dziedzinie systemów operacyjnych. Poniżej krótko omówiono najważniejszych konkurentów systemu Android.

**iOS** – największy konkurent Androida, który (według firmy Apple) skopiował wiele zawartych w iOS rozwiązań. Opracowany przez Apple, jest systemem zamkniętym pracującym tylko na urządzeniach tej firmy. Choć pozostaje zdecydowani w tyle pod względem liczby urządzeń (ok. 470 mln), jest największym konkurentem Androida i zdecydowanie wyprzedza go pod kątem **dochodu generowanego przez aplikacje** – większość gier i aplikacji zarabia więcej na iOS niż na Androidzie. Językiem programowania natywnym dla iOS jest Objective C, który ma zostać zastąpiony przez nowy język programowania Swing.

**Windows Phone** – wciąż relatywnie mało popularny i niedoceniany system, który jednak dzięki dobrym rozwiązaniom technologicznym, gigantycznym inwestycjom firmy Microsoft i pracy na bardzo dobrych urządzeniach Nokii (wykupionej przez Microsoft) ma realną szansę zdobyć duży udział w rynku. Aplikacje dla tego systemu tworzy się w środowisku .Net, dla którego domyślnym językiem programowania jest C#.

**Blackberry OS** – system opracowany przez producenta urządzeń o tej samej nazwie, wciąż popularny wśród klientów biznesowych, jednak od pewnego czasu znajdujący się na równi pochyłej. Trwają spekulacje co do zakupu firmy Blackberry np. przez firmę Lenovo, która mogłaby kontynuować rozwój systemu.

**Tizen** – również oparty na Linuxie system, mający długą historię, obecni rozwijany przez firmę Samsung, która planuje jego wykorzystanie również w produkowanych przez siebie telewizorach.

Jeśli Samsung zdecyduje się w pełni wspierać ten system na urządzeniach mobilnych (gdzie obecnie dominuje Android), na pewno zyska on istotną pozycję na rynku.

**Sailfish** – kolejny system oparty na Linuxie, opracowany w Finlandii między innymi przez weteranów wywodzących się z Nokii (firmę Jolla). Obecny na rynku od niedawna, jednak dość popularny wśród developerów.

[13] Jest to odpowiednik Java Runtime Environment dla systemu Android.

**Firefox OS** – system operacyjny opracowany przez fundację Mozilla, również oparty na Linuxie.

**Ubuntu** – próba przeniesienia popularności system Ubuntu na urządzenia mobilne, podjęta przez firmę Canonical.

Ta lista z pewnością nie jest wyczerpująca, wymienia jednak najważniejszych obecnych i potencjalnych konkurentów dla systemu Android.

### 1.2.5. Wieloplatformowe (cross-platform) środowiska programowania aplikacji mobilnych

Obecnie projektując aplikacje mobilne należy uwzględniać przynajmniej trzy liczące się na rynku systemy operacyjne: Android, iOS i Windows Phone, nie licząc mniejszych (wymienionych w poprzednim punkcie graczy). Każdy z tych systemów jest oparty na innych założeniach i technologii, co więcej wykorzystuje jako natywny inny język programowania – są to odpowiednio Java, Objective C i C# (.Net)

W tej sytuacji coraz popularniejsze stają się rozwiązania umożliwiające tworzenie aplikacji wieloplatformowych, czyli bez zmiany kodu pracujących na różnych platformach sprzętowych, pod kontrolą różnych systemów operacyjnych. Takich narzędzi jest na rynku bardzo dużo, poniżej krótko omówiono najważniejsze.

**Unity3D**[14] – najpopularniejszy na rynku wieloplatformowy silnik wspierający urządzenia mobilne, dedykowany dla gier i wizualizacji 3D.

Bez zmiany kodu umożliwia uruchamianie raz napisanych programów na wszystkich najważniejszych systemach mobilnych (iOS, Android, Windows Phone, Tizen). Co więcej wspierane są również konsole do gier oraz systemy Windows i OSX.

Unity3D oparty jest o Mono (wieloplatformową, otwartą implementację platformy .Net), programy pisane są w językach C# lub Unity Script (pochodna JavaScript)

**Xamarin**[15] – najpopularniejsze rozwiązanie dla aplikacji użytkowych i biznesowych. Podobnie jak Unity 3D, wykorzystuje platformę Mono.

**PhoneGap** (Apache Cordova) – reprezentuje inną filozofię tworzenia aplikacji. Zapewnia natywne „opakowanie" umożliwiające uruchamianie na systemach mobilnych aplikacji stworzonych w technologiach sieciowych, określanych zbiorczą nazwą HTML5. Zaletą takiego podejścia jest możliwość wykorzystania wiedzy związanej z projektowaniem aplikacji internetowych oraz szybkie portowanie różnego rodzaju serwisów i portali na urządzenia mobilne. PhoneGap[16] jest komercyjnym rozwiązaniem opartym na silniku Cordova.

Warto pamiętać, że dedykowane aplikacje natywne wciąż odgrywają istotną rolę w wielu przypadkach, w szczególności, gdy istotne są:

- natywny interfejs użytkownika,
- szybkość pracy,

[14] http://unity3d.com/
[15] http://xamarin.com/
[16] http://phonegap.com/, http://cordova.apache.org/

– zużycie pamięci,
– funkcje dedykowane tylko dla jednej platformy.

Warto zauważyć, że np. popularne serwisy społecznościowe Facebook i LinkedIn odstąpiły od aplikacji mobilnych będących tylko „opakowaniami" dla kodu HTML5 na rzecz w pełni natywnych aplikacji, tłumacząc to względami wydajności i zarządzania zasobami.

### 1.2.6. Różnice między Java SE a Android API

Jednym z częściej popełnianych błędów jest utożsamianie programów dla systemu Android z programami dla Java Standard Edition. Główna różnica polega na tym, że biblioteki (API) Androida są podobne, ale nie takie same. Większość standardowych podstawowych pakietów (java.util, java.text, Java.net) jest taka niemal identyczna, jednak już funkcje zaawansowane – jak interfejs użytkownika, obsługa mediów są zrealizowane w inny sposób. Do tego dochodzi szereg funkcji specyficznych dla Androida jako mobilnego systemu operacyjnego. Istnieją też drobniejsze różnice w zachowaniu API – przykładowo program HelloWorld mógłby nie zadziałać na Androidzie, ponieważ według specyfikacji strumień *System.out* nie jest obsługiwany, zamiast tego sugeruje się używanie funkcji *Log*[17].

Podstawowe biblioteki Androida są oparte na bibliotekach Java 5 – wtedy drogi Androida i Java SE się rozeszły. Upraszczając, można powiedzieć, że Android w większości obsługuje podstawowe pakiety java.* (core Java) i nie obsługuje pakietów zaawansowanych javax.* (Advanced Java)

Wspierane pakiety:

– java.io – File and stream I/O;
– java.lang (except java.lang.management) – Language and exception;
– support;
– java.math – Big numbers, rounding, precision;
– java.net – Network I/O, URLs, sockets;
– java.nio – File and channel I/O;
– java.security – Authorization, certificates, public keys;
– java.sql – Database interfaces;
– java.text – Formatting, natural language, collation;
– java.util (including java.util.concurrent) – Lists, maps, sets, arrays, collections;
– javax.crypto – Ciphers, public keys;
– javax.net – Socket factories, SS;
– javax.security (except javax.security.auth.kerberos, javax.security.auth.spi, and javax.security.sasl);
– javax.sound – Music and sound effects;
– javax.sql (except javax.sql.rowset) – More database interfaces
– javax.xml.parsers – XML parsing;
– org.w3c.dom (but not sub-packages) – DOM nodes and elements;
– org.xml.sax – Simple API for XML;

[17] Na emulatorze i większości urządzeń następuje przekierowanie funkcji System.out.println do Log.

**Niewspierane pakiety[18]:**

Java.applet,
Java.awt,
Java.beans,
Java.lang.management,
Java.rmi,
Javax.accessibility,
Javax.activity,
Javax.imageio,
Javax.management,
Javax.naming,
Javax.print,
Javax.rmi,
Javax.security.auth.kerberos,
Javax.security.auth.spi,
Javax.security.sasl,
Javax.swing,
Javax.transaction,
Javax.xml (except Javax.xml.parsers),
org.ietf.*,
org.omg.*,
org.w3c.dom.* (sub-packages).

**Biblioteki zewnętrzne** – standardowo Android obsługuje też kilka bibliotek zewnętrznych:

- org.apache.commons.codec – Utilities for encoding and decoding;
- org.apache.commons.httpclient – HTTP authentication, cookies, methods, and protocol;
- org.bluez – Bluetooth support;
- org.json – JavaScript Object Notation.

Co więcej, maszyna wirtualna Androida, Dalvik VM[19] nie wykonuje programów w kodzie bajtowym (*bytecode*). Skompilowane pliki class muszą być dodatkowo przekodowane do formatu dex (Dalvik EXecutable), którego format został zoptymalizowany na potrzeby urządzeń mobilnych. W odróżnieniu od wielu plików class, cały program jest zawarty w jednym pliku dex[20], dlatego można powiedzieć, że pełni on rolę podobną do archiwów Jar w Java SE.

**Uwaga!** Javy dla Androida nie należy też mylić z Java Micro Edition.

### 1.2.7. Android Studio

Firma Google ogłosiła niedawno, że oficjalnym IDE dla Androida jest An-

[18] Lista według http://www.zdnet.com/article/java-vs-android-apis/
[19] Od wersji 5.0 domyślną maszyną jest ART – Android RunTime.
[20] Według projektantów systemu Android, format dex pozwala zmniejszyć rozmiar programów o ponad połowę: http://davidehringer.com/software/android/The_Dalvik_Virtual_Machine.pdf

droid Studio, którego wersja 1.0 ukazała się w grudniu 2014 roku – stąd większość materiałów na stronie developer.android.com została zmieniona, by odnosić się właśnie do tego środowiska.

W odróżnieniu od uniwersalnego Eclipse Android Studio ma być dedykowane dla programowania w tym systemie, dlatego należy spodziewać się większej prostoty obsługi i dopracowania dostępnych narzędzi. Więcej informacji na ten temat znaleźć można pod adresem https://developer.android.com/tools/studio/index.html

## 1.3. Najważniejsze elementy systemu Android

Jak już wspomniano, Android jest systemem opartym na założeniach Linuxa. Dla bezpieczeństwa (przynajmniej w teorii) każda aplikacja w systemie ma ściśle wydzieloną przestrzeń:

- własnego użytkownika systemu
- wydzielony proces
- oddzielną maszynę wirtualną

W określonych sytuacjach aplikacje mogą współdzielić wymienione wyżej zasoby w celu oszczędności energii. Obowiązuje zasada najmniejszego uprzywilejowania – każda aplikacja ma dostęp tylko do tych zasobów, które są niezbędne do jej działania. Dodatkowo aplikacje mogą prosić o przyznanie określonych przywilejów, np.:

- Dostęp kontaktów,
- Wysyłanie SMS,
- Dostęp do zdjęć,
- Wykorzystanie aparatu,
- Użycie Bluetooth.

### 1.3.1. Komponenty aplikacji

Programy Androida nie udostępniają jednego punktu wejścia – funkcji *main*. Zamiast tego składają się z autonomicznych komponentów, z których każdy (przynajmniej w teorii) może być uruchomiony niezależnie. Istnieją cztery główne rodzaje komponentów:

- **Activities** – Aktywność – pojedynczy ekran z interfejsem użytkownika (scena);
- **Services** – Komponent pracujący w tle (np. lokalizacja GPS);
- **Content providers** – Zarządza danymi aplikacji (baza, Web) – np. kontakty;
- **Broadcast receivers** – Odpowiada na systemowe „rozgłoszenia" – np. zakończono pobieranie.

*Activities* (podobnie jak *broadcast receivers* i *content providers*) uruchamiane są przez asynchroniczne komunikaty (*message*) noszące nazwę intencji (*Intent*). Dzięki temu aplikacje mogą wzajemnie wykorzystywać swoje aktywności – np. program kamery może wysłać zdjęcie pocztą, uruchamiając odpowiednią aktywność programu pocztowego, bez konieczności jej samodzielnego implementowania. Przykładowo – uruchomienie aktywności przeglądania strony internetowej wygląda następująco:

```
    Intent i = new Intent(Intent.ACTION_VIEW, Uri.parse
("http://www.wsb.edu.pl/"));
    startActivity(i);
```

Dane typy aktywności (np. SEND_MAIL) mogą być implementowane przez więcej niż jedną aplikację.

Wyczerpujące informacje na temat komponentów aplikacji, podane w przystępnej formie znaleźć można pod adresem http://developer.android.com/guide/components/index.html.

### 1.3.2. Plik manifest

Opis najważniejszych elementów aplikacji dla systemu Android znajduje się w pliku AndroidManifest.xml (można znaleźć go w głównym katalogu projektu).

Zawiera on następujące informacje:

– nazwę pakietu Java aplikacji

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.wsb"
    android:versionCode="1"
    android:versionName="1.0" >
```

– deklaracje komponentów (activity, service, provider, receiver), w tym nazwy klas oraz informacje jaką funkcjonalność realizują – jakie intencje obsługują i w jakich warunkach można je aktywować, np.:

```
    <activity
android:name="com.example.project.ComposeEmailActivity">
            <intent-filter>
                <action android:name="android.intent.action.SEND" />
                <data android:type="*/*" />
                <category
android:name="android.intent.category.DEFAULT" />
            </intent-filter>
        </activity>
```

– określenie procesu aplikacji,

– deklaracje, których pozwoleń wymaga aplikacja aby korzystać z chronionych elementów API i komunikować się z innymi (np. dostęp do Bluetooth, dostęp do Internetu)

```
<uses-permission android:name="android.permission.INTERNET"/>
```

– deklarację pozwoleń, których potrzebują inni by komunikować się z aplikacją,

– wymagania sprzętowe (np. ekrany multitouch albo odbiornik GPS),

– dane o instrumentacji kodu (związane z testowaniem i uruchamianiem aplikacji),

– minimalny wymagany poziom Android API (czyli wersję Androida potrzebną do uruchomienia aplikacji),

```
    <manifest ... >
    <uses-sdk android:minSdkVersion="14"
android:targetSdkVersion="19" />
    ...
</manifest>
```

– listę bibliotek, które wykorzystuje aplikacja (np. Google Maps)

```
    <uses-library android:name="com.google.android.maps"/>
```

Pełne omówienie formatu i danych pliku manifest znajduje się pod adresem http://developer.android.com/guide/topics/manifest/manifest-intro.html. W większości przypadków korzystając z nowoczesnego IDE (Eclipse, Android Studio) wielu informacji nie trzeba edytować ręcznie, IDE uaktualnia je w trakcie tworzenia programu.

Warto zwrócić uwagę na znaczenie dobrze sformułowanych wymagań programu – Google Play (cyfrowy sklep z aplikacjami dla Androida) nie pozwoli na zainstalowanie aplikacji na urządzeniach niespełniających minimalnych wymagań, co jest bardzo istotne dla satysfakcji użytkowników – wyobraźmy sobie bardzo dobry program do robienia zdjęć otrzymujący dużą ilość negatywnych ocen dlatego, że użytkujący go użytkownicy nie posiadają aparatów w swoich urządzeniach.

Alternatywnie – dostępność określonych funkcji można sprawdzać z poziomu aplikacji, blokując tylko fragmenty programu, zamiast jego całości:

```
    PackageManager pm = getPackageManager();
if (!pm.hasSystemFeature(PackageManager.FEATURE_SENSOR_COMPASS)) {
    // This device does not have a compass, turn off the compass feature
    disableCompassFeature();
}
```

### 1.3.3. Prawa aplikacji

Domyślnie żadna aplikacja nie ma praw pozwalających zagrozić:

– innym aplikacjom,
– systemowi operacyjnemu,
– użytkownikowi.

Jest to tzw. *process sandbox* – wyseparowana logicznie przestrzeń, w której pracują aplikacje.

Pozwolenia przyznawane są statycznie (w chwili instalacji), nie dynamicznie (w trakcie wykonywania). Wszystkie aplikacje muszą być podpisane kluczem developera – przy generowaniu nie jest wymagane *certificate authority*[21].

[21] Więcej informacji na temat podpisów cyfrowych i kryptografii asymetrycznej można znaleźć w opracowaniu A. Grzywak, J. Klamka, P. Buchwald, P. Pikiewicz, M. Rostanski i M. Sobota, *Podpis elektroniczny i identyfikacja użytkowników w sieci Internet*, Wyższa Szkoła Biznesu w Dąbrowie Górniczej, Dąbrowa Górnicza 2013.

### 1.3.4. Podsumowanie

Programowanie w systemie Android jest bardzo rozległym tematem, któremu poświęcono niejedną publikację czy „tutorial". Celem niniejszego rozdziału jest stworzenie solidnych podstaw, które w połączeniu z załączonym na końcu książki samodzielnym projektem (Dodatek A) pozwolą zainteresowanym rozwinąć wiedzę praktyczną oraz zapewnić zrozumienie kluczowych zagadnień.

# 2

# ARCHITEKTURA NOWOCZESNYCH APLIKACJI MOBILNYCH OPARTYCH O MODEL RESTFUL

Niniejszy rozdział dotyczy budowy nowoczesnych aplikacji sieciowych, gdzie klientem może być zarówno przeglądarka internetowa jak i natywna aplikacja mobilna pisana na platformy np. IOS czy Android. Aplikacje sieciowe wykorzystują wiele dodatkowych technologii jak np. Ajax i korzystają zazwyczaj z protokołu HTTP jako kanału komunikacji klient – serwer. W dużym uproszczeniu architektura wielowarstwowa polega na rozdzieleniu interfejsu użytkownika od przetwarzania danych oraz ich składowania na kilka warstw, co powoduje łatwiejszy ich rozwój oraz utrzymanie. Taki podział nie powoduje zakłóceń, a poszukiwanie ewentualnych błędów jest łatwiejsze. Dodatkowo upraszcza konstrukcję, co prowadzi do elastycznej i rozszerzalnej budowy.

## 2.1. Co to jest REST?

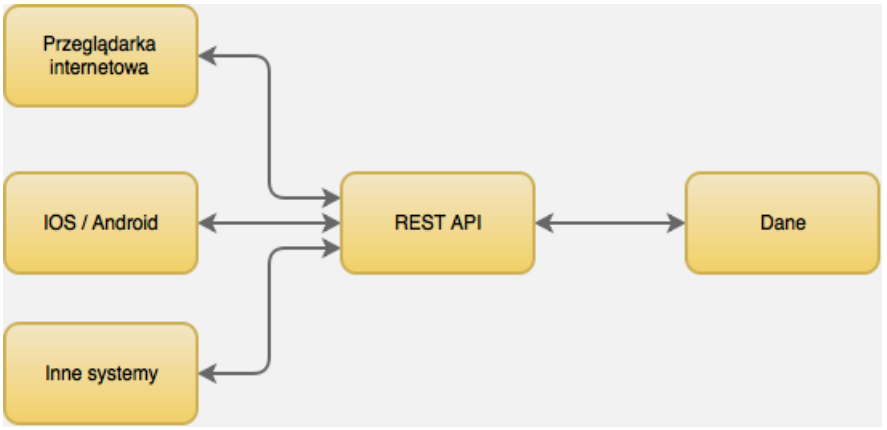
REST to skrót od REpresentation State Transfer – zmiana stanu poprzez reprezentację, czyli architektura oprogramowania mająca swoje źródła w protokole HTTP. REST korzysta z jednorodnego interfejsu, bezstanowej komunikacji, zasobów i reprezentacji.

Jedną z najważniejszych cech modelu REST dla aplikacji sieciowych jest bezstanowość komunikacji pomiędzy klientem (np. aplikacją mobilną, przeglądarką internetową, aplikacją na platformy Windows / MAC OSX) i serwerem. Każde żądanie od klienta musi zawierać komplet informacji, aby zostało

zrozumiane przez serwer, dodatkowo klient nie jest powiadamiany o zmianie stanu serwera pomiędzy żądaniami (np. restart usługi). Pozwala to na obsługę przez dowolny serwer który posiada logikę aplikacji np. w chmurze prywatnej. W celu poprawy wydajności korzysta się również z technologii buforowania danych.

Funkcjonalność oraz stan aplikacji w modelu REST podzielony jest na zasoby. Zasób jest przekazywany klientom i jest najważniejszym elementem koncepcyjnym modelu. Pod pojęciem zasobu rozumiemy obiekty, rekordy z bazy danych, algorytmy czy media udostępniane w ramach aplikacji. Każdy z zasobów powinien mieć unikalny adres URI (ang. *Universal Resource Identifier* – Ujednolicony Identyfikator Zasobów – jest standardem internetowym umożliwiającym łatwą identyfikację zasobów w sieci).

Rysunek 5. Schemat ogólny modelu REST




Wszystkie zasoby dzielą jednolity interfejs dla zapisu stanu pomiędzy serwerem a klientem i wykorzystują następujące metody protokołu HTTP:

- GET – reprezentuje sposób przekazywania danych pomiędzy kolejnymi żądaniami od klienta i polega na umieszczeniu par *parametr=wartość* w adresie żądania np.:
  *www.przyklad.pl/login.php?kategoria=kwiatki&nazwa=orchidea*
- POST – metoda przesyłania danych w sieci Internet wykorzystywana zazwyczaj do wysyłania danych z formularzy znajdujących się na stronach internetowych, przykład wywołania poniżej:

```
Host: www.przyklad.pl

User-Agent: Mozilla/42.0

Content-Length: 29

Content-Type: application/x-www-form-urlencoded
```

```
login=uzytkownik&haslo=ZAQ12wsx
```

– PUT – przyjęcie danych wysyłanych przez klienta na serwer, najczęściej w celu aktualizacji zasobu
– DELETE – żądanie usunięcia zasobu, metoda włączona zazwyczaj tylko dla uprawnionych użytkowników.

Kolejnym istotnym elementem modelu REST jest budowa warstwowa ograniczająca dostęp komponentów systemu tylko do warstw z nim powiązanych logicznie i z którym interakcja jest wymagana. Tego rodzaju ograniczenia mają wpływ na złożoność systemu oraz niezależność poszczególnych warstw. Dzięki takiemu podziałowi skalowalność aplikacji jest bardzo duża i pozwala na obsługę wielu klientów jednocześnie, ale nie tylko, zmniejsza również opóźnienia pomiędzy klientem i serwerem, ułatwia zrozumienie architektury dla nowych członków zespołu programistycznego w szczególności jeżeli różne osoby są odpowiedzialne za różne moduły oraz osób zewnętrznych, które korzystają z funkcji udostępnianych jako API (z **ang.** *Application Programming Interface* – Interfejs programistyczny aplikacji).

## 2.2. Usługa w modelu REST i RPC

Do tej pory najczęściej wykorzystywanym modelem udostępniania klientom usług sieciowych był model SOAP (ang. Simple Object Access **Protocol** – protokół komunikacyjny wykorzystujący język XML) który czerpał inspiracje z RPC (z ang. Remote Procedure Call – Zdalne wywołanie procedury) i był najpopularniejszym przedstawicielem implementacji architektury opartej na usługach (SOA – ang. Service – **Oriented** Architecture). Klienci usługi przesyłali dokument XML (ang. envelope – koperta) wraz z nagłówkami, danymi oraz argumentami za pomocą protokołu HTTP, przykład poniżej:

```
POST /InStock HTTP/1.1
Host: www.example.org
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
<soap:Body xmlns:m="http://www.example.org/stock">
  <m:GetStockPrice>
    <m:StockName>IBM</m:StockName>
  </m:GetStockPrice>
</soap:Body>
</soap:Envelope>
```

Serwer odczytuje informacje z koperty i wykonuje metody wraz z podanymi argumentami, następnie wyniki uzyskane z działania algorytmu na serwerze

pakowane są w kopertę zwrotną i przesyłane do klienta usługi jako odpowiedź. Klient otrzymuje kopertę z wynikami i odczytuje jej zawartość. Każdy z obiektów ma swój unikalny zbiór metod i na zewnątrz udostępniany jest jedynie URI, który jest jedynym punktem dostępu, zarazem ignorowana jest większość dostępnych **metod protokołu** HTTP, pozostaje do użycia jedynie metoda POST.

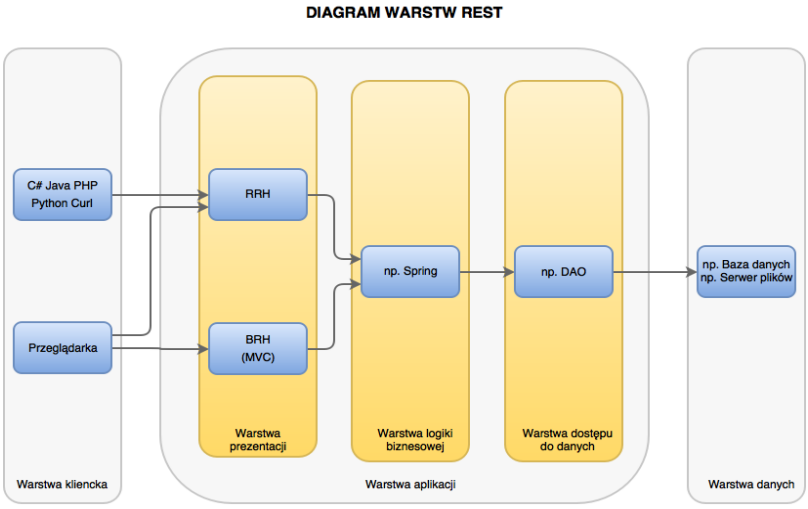
Popularność modelu REST zwiększa się głównie z powodu lekkości użycia i możliwości wykorzystania przesyłania danych bezpośrednio poprzez protokół HTTP. Zakres języków obsługujących **implementację jest** ogromny począwszy od języka Java, C#, Perl, **Ruby**, Python, PHP i **Javascript** włącznie z wykorzystaniem technologii Ajax. Popularność zawdzięcza również z powodu prostoty użycia, gdzie użytkownik w zasadzie może bezpośrednio generować odpowiednie zapytania GET wprost w przeglądarce internetowej i odczytywać zwracane wyniki w postaci nie obrobionej, ale zazwyczaj zrozumiałej. Większość interakcji wykonywanych jest poprzez zautomatyzowane procedury pobierające i przesyłające żądania w imieniu użytkownika.

W przypadku usług sieciowych udostępnianych w modelu REST każdy zasób ma adres, który wykorzystywany jest do wywoływania metod, która jest taka sama dla wszystkich zasobów w postaci standardowych wbudowanych w protokół HTTP w tym wcześniej wspomniane GET, POST, PUT, DELETE. W przypadku architektury w stylu RPC główny nacisk jest na metody, natomiast REST największą wagę przykłada do reprezentacji w postaci zasobów na których wykonywany jest zbiór zdefiniowanych metod wcześniej wymienionych, połączonych ze sobą hiperłączami. Jest jeszcze pewna kombinacja REST i RPC w postaci systemów hybrydowych, gdzie usługa przesyła dane przez protokół HTTP, ale nie korzysta ze standardowych metod HTTP do operacji na zasobach. Informacje o metodach zapisywane są w adresie URI zapytania. Tego typu architekturę hybrydową wykorzystuje np. Yahoo Flickr API.

## 2.3. Wielowarstwowa architektura usług sieciowych

Wykorzystanie wielowarstwowej architektury daje wiele korzyści, najważniejsza z nich to możliwość wykorzystania logiki biznesowej aplikacji oraz dostępu do danych w taki sam sposób zarówno z poziomu aplikacji z interfejsem oraz zarówno przez zautomatyzowane systemy naszych dostawców, kontrahentów jak również przez użytkowników końcowych chcących zwiększyć komfort współpracy. Główną różnicą jest warstwa prezentacji i obsługi żądań. Dodatkowo dzięki rozdzieleniu warstw danych i dostępu do danych uniezależniamy nasz system od technologii bazodanowej pozwalając na swobodny jej wybór w zależności od potrzeb.

Rysunek 6. Diagram warstw w architekturze wielowarstwowej




Na diagramie w warstwie klienckiej znajdują się zarówno klienci zautomatyzowani w postaci aplikacji napisanych w językach programowania C#, C++ czy Java, ale również aplikacje języków skryptowych Python, Perl, Ruby, PHP, a także narzędzia uruchamiane z linii poleceń jak Curl. W warstwie klienta należy jeszcze wymienić technologie działające wewnątrz przeglądarki w postaci Ajax czy Flash, które wykonują operacje w imieniu użytkownika. Następną warstwą po warstwie klienckiej jest warstwa aplikacji, na którą składa się kilka mniejszych warstw, jak prezentacji, logiki biznesowej i dostępu do danych.

W warstwie prezentacji obsługę zapewniają obiekty RRH (ang. *Resource Request Handler*) i BRH (ang. *Browse Request Handler*), które odpowiadają na żądania warstwy klienckiej. Są one punktem dostępowym dla warstwy klienckiej i potrzebują informacji o ścieżce, kontekście czy id żądanego zasobu. Odpowiedzialne są za obsługę żądań i konwersję do formy zrozumiałej dla klienta korzystającego z protokołu HTTP. Odpowiedzialne są również za proces kompozycji, gdy zwracany jest więcej niż zasób lub występuje ich zagnieżdżenie. Zapytania z warstwy klienckiej są bezstanowe i zawierają informacje o metodzie (GET, POST, PUT lub DELETE) w nagłówku, które następnie są zamieniane na odpowiednie operacje w RRH. Każde z wywołań zawiera komplet informacji (wraz z uwierzytelnianiem).

Obiekty RRH komunikują się z logiką biznesową wywołując odpowiednie operacje, które wykonuje aplikacja. Dla zasobów przydzielane są odpowiadające im unikalne URI. Warstw logiki biznesowej komunikuje się z abstrakcyjną warstwą dostępu do danych ukrywającą technologie przechowywania danych. Następnie warstwa dostępu danych wykonuje operacje bezpośrednio na danych

Należy tutaj również wspomnieć o naturalnej koegzystencji technologii Ajax i RESTful. Obydwie korzystają z powszechnie znanych i używanych technologii jak HTML, Javascript, XML, JSON i HTTP. Nie ma potrzeby instalacji żadnych dodatkowych bibliotek, czy zakupu dodatkowych licencji w celu efektywnej współpracy pomiędzy nimi. Ajax za pomocą żądań uzyskuje dostęp do zasobów na serwerze i metod operacji na nich poprzez REST.

Przeglądarka internetowa jak, Firefox, Chrome, Safari czy Internet Explorer występuje jako interfejs pomiędzy użytkownikiem i serwerem pełniąc funkcję wyświetlania HTML generowanego przez warstwę prezentacji. Warstwa prezentacji powinna być zaimplementowana w modelu MVC (ang. *Model-View-Controller* – model-widok-kontroler). Do najczęściej wykorzystywanych należą np. Struts czy ASP.Net MVC. MVC prościej jest rozpatrywać jako złożony model wykorzystujący prostsze wzorce projektowe, w tym:

- kompozyt – praca z zagnieżdżonymi widokami
- obserwator – umożliwia powiadamianie widoku o zmianach w modelu
- strategia – widok pozostawia obsługę zachowania w gestii implementacji kontrolera
- metoda wytwórcza – domyślny kontroler dla całej aplikacji
- dekorator – pozwala łatwo rozszerzać widoki o nowe funkcje.

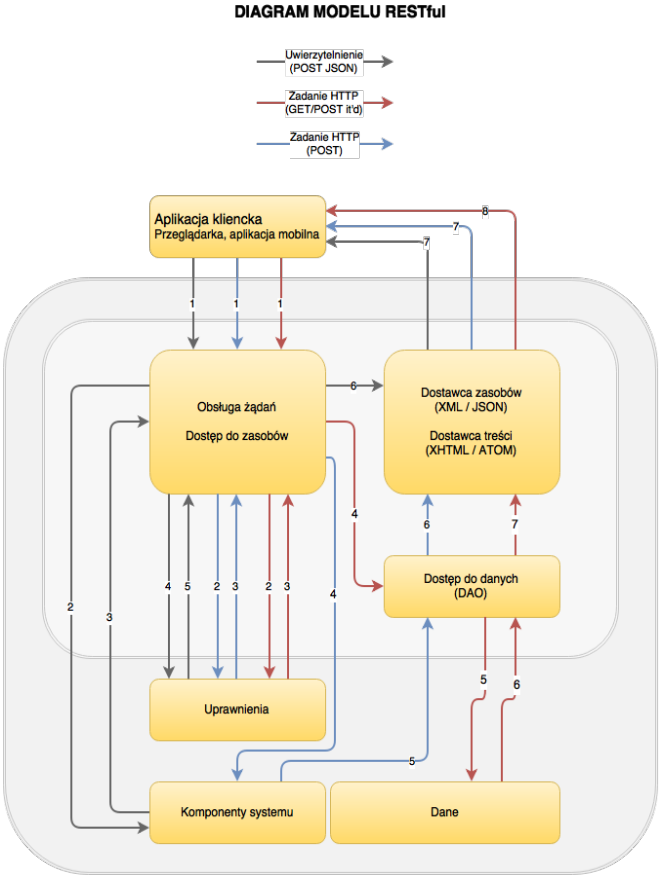
Podsumowując MVC obsługuje żądania przeglądarki, wykonuje odpowiednie funkcje logiki aplikacji, generuje stronę i zwraca kod HTML, Javascript i CSS do przeglądarki.

Logika aplikacji znajduje się w **warstwie logiki biznesowej**, która wymienia informacje z warstwą dostępu do danych i warstwą prezentacji. Dane przekazywane do warstwy prezentacji w postaci obiektów i wartości. Oddzielenie obiektów RRH i BRH od warstwy logiki biznesowej pozwala na łatwiejsze ponowne użycie kodu, zwiększa elastyczność oraz rozszerzalność (rys. 7).

**Warstwa dostępu do danych** często implementowana jest przy użyciu wzorca projektowego DAO (data Access Object – obiekt dostępu do danych). Wzorzec ten dostarcza jednolity interfejs komunikacji między aplikacją i zbiorem danych. Implementuje się w nim trwałość, transakcje i alokację zasobów. Największą zaletą jest separacja kodu dostępu do danych od logiki biznesowej aplikacji. Warstwa dostępu do danych może stanowić punkt integracji z innymi systemami w celu udostępniania danych z różnych systemów w ten sam sposób.

Ostatnią warstwą jest **warstwa przechowywania danych,** których najczęstszym przedstawicielem jest system bazodanowy. Wykorzystanie takiego podziału na warstwy ma same zalety. Może udostępniać dane firmom współpracującym, klientom z różnych platform czy po prostu poprzez przeglądarkę internetową. Zagadnienia związane z warstwą przechowywania danych są na tyle istotne, że stanowią treść kolejnego rozdziału.

Rysunek 7. Diagram przepływu i wywołań pomiędzy komponentami systemu wielowarstwowego




Uwierzytelnienie
- Żądanie uwierzytelnienia użytkownika,
- Wywołanie żądania do komponentu systemu zarządzania użytkownikami,
- Zwracany jest uwierzytelniona tożsamość,
- Tworzenie tokenu zabezpieczeń (np. odczyt uprawnień),
- Zwrot tokenu zabezpieczeń,
- Wygenerowanie odpowiedzi,
- Zwrot odpowiedzi do klienta.

Logika aplikacji – żądanie HTTP
- Żądanie z odpowiednimi metodami i tokenem zabezpieczeń,
- Sprawdzenie tokenu zabezpieczeń,
- Weryfikacja uprawnień,
- Wywołanie odpowiedniego komponentu systemu,

– Komunikacja z warstwą dostępu danych,
– Zwrot danych do warstwy prezentacji,
– Odpowiedź do klienta.

Wykonanie komendy – np. generowanie raportu,
– Żądanie z odpowiednimi metodami i tokenem zabezpieczeń,
– Sprawdzenie tokenu zabezpieczeń,
– Weryfikacja uprawnień,
– Zapytanie o dane do warstwy dostępu danych,
– Żądanie udostępnienia danych przez np. system bazodanowy,
– Odpowiedź z warstwy danych,
– Wygenerowanie odpowiedzi np. w formacie JSON.

## 2.4. Bezpieczeństwo w usłudze REST

Bardzo ważnym aspektem każdego systemu informatycznego są kwestie bezpieczeństwa i poprawnego rozpoznawania użytkowników naszego systemu. Na bezpieczeństwo składa się uwierzytelnienie, autoryzacja, uprawnienia dostępu, kopie zapasowe czy bezpieczeństwo fizyczne. Nas interesuje z punktu widzenia aplikacji wielowarstwowej głównie uwierzytelnianie oraz prawa dostępu.

Uwierzytelnienie to proces sprawdzenia i potwierdzenia tożsamości stron komunikacji. Ważny jest również aspekt pewności, że dany podmiot jest tym za kogo się podaje i w zależności od wymagań możemy wprowadzać wiele poziomów. Do najbardziej zaawansowanych służą metody biometryczne, a w systemach zdalnych certyfikaty imienne wystawiana dla użytkowników i uwierzytelnianie dwuskładnikowe np. hasło + sms. Kolejnym elementem jest autoryzacja, która służy do potwierdzenia czy użytkownik (lub inny system) jest uprawniony do żądanego zasobu. Proces uwierzytelniania został zakończony przyznaniem tokenu zabezpieczeń, z którym sprawdzany jest poziom uprawnień.

W procesie uwierzytelnienia w modelu REST dane służące do uwierzytelnienia przesyłane są w nagłówku HTTP. Jeżeli dane nie są poprawne zwracany jest kod 401 – nieautoryzowany dostęp.

```
HTTP/1.0 401 Unauthorized
Date: Mon, 1§ Jan 2015 11:17:51 GMT
Server: Apache/2.0.55 (Unix) mod_ssl/2.0.55 OpenSSL/0.9.7g
PHP/5.4.1\
WWW-Authenticate: Basic realm=„Hasło”
Accept-Ranges: bytes
Content-Length: 3172
Content-Type: text/html
X-Cache: MISS from www.przyklad.pl
```

W przypadku uwierzytelnienia bazowego dane są kodowane algorytmem *base64* i przesyłane do serwera w postaci jawnego tekstu. Dopełnieniem procesu autoryzacji powinno być szyfrowanie połączenia pomiędzy klientem i serwerem uwierzytelniającym protokołem SSL. Kwestie bezpiecznego kanału komunikacji mamy zamkniętą, ale jest jeszcze jeden problem bezpieczeństwa ,który trzeba rozważyć. Dlaczego użytkownik ma ufać oprogramowaniu z którego korzysta. Spróbujmy skonkretyzować problem. Czy ufasz swojej przeglądarce gdy korzystasz z bankowości internetowej? Pewnie tak, ale czy masz 100% pewności że Twoje dane logowania nie są wysyłane jeszcze gdzieś indziej? Czy system z którego API korzystasz uwierzytelniając się jest na pewno bezpieczny? Możemy korzystać z aplikacji o otwartym kodzie, którego źródła są ogólnie dostępne i wielu użytkowników na świecie ten kod przegląda. Niestety ale bardzo często nie mamy możliwości przeglądania kodu usługi sieciowej z której korzystamy i nie mamy pełnej gwarancji, w jaki sposób informacje, które wymieniamy pomiędzy klientem i usługą, przebywają drogę.

## 2.5. REST – prosty przykład

Zobaczmy przykład prostej usług sieciowej, która pobiera szczegółowe informacje o właścicielu samochodu – wszystko co wiemy to numer tablicy rejestracyjnej.

W przypadku usług sieciowych i protokołu SOAP żądanie będzie wyglądać mniej więcej tak:

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
<soap:body pb="http://www.przyklad.pl/listasamochodow">
     <pb:PobierzDaneSamochodu>
          <pb:idSamochodu>ST4281G</pb:idSamochodu>
     </pb:PobierzDaneSamochodu>
</soap:Body>
</soap:Envelope>
```

Taka „koperta” zostanie teraz przesłana do serwera metodą POST protokołu HTTP. Rezultatem może być np. plik XML, który musi zostać „zapakowany” do SOAP i odesłany jako odpowiedź.

W przypadku protokołu REST zapytanie wygląda dokładnie tak:

```
http://www.przyklad.pl/listasamochodow/daneSamochodu/samochod/ST4281
```

http://www.przyklad.pl/listasamochodow/daneSamochodu/samochod/ST4281

Zauważmy, że to nie jest treść zapytania, a po prostu adres internetowy. Takie zapytanie jest wysyłane bezpośrednio do serwera metodą GET, zwra-

cane dane nie są niczym opakowane, to dane których można od razu użyć zazwyczaj w formacie JSON lub XML. Jedyne, czego potrzebujemy to dostęp do Internetu, możemy użyć do testów nawet przeglądarki internetowej. Często w przypadku REST wykorzystuje się rzeczowniki zamiast czasowników dla oznaczenia zasobów.

Korzystając z SOAP w zasadzie dosłownie opakowujemy zapytania i odpowiedzi, które należy przygotować przed wysłaniem, a później zdekodować co powoduje dodatkowy narzut i wzrost zużycia nie tylko zasobów (czas procesora, pamięć) ale również przepustowości. Bezpieczeństwo protokołu powinno zostać zapewnione poprzez szyfrowanie protokołu HTTP mechanizmami SSL. Bez szyfrowania zarówno SOAP jak i REST przesyłane są jawnym tekstem.

### 2.5.1. Złożone żądania REST

Sprawdziliśmy, że wykorzystanie modelu REST jest bardzo proste i zarazem niesamowicie skuteczne. Poprzedni przykład przekazywał jeden parametr, czyli numer tablicy rejestracyjnej. Spróbujemy przygotować bardziej zaawansowane żądanie przesyłające kilka parametrów przy wykorzystaniu metody GET.

```
http://www.przyklad.pl/listasamochodow/daneSamochodu/samochod?nrTabl
icyRej=ST4281&rokRejestracji=2015
```

Jeżeli mamy potrzebę przesłania danych binarnych, lub których długość jest zbyt duża dla metody GET, możemy przesłać je za pomocą metody POST protokołu HTTP. Wykorzystując metodę GET powinno unikać się modyfikowania stanu serwera oraz danych – nie jest to przymusem, ale dobrą praktyką. Jeżeli będziemy modyfikowali dane korzystajmy z metody POST, która może być wykorzystywana do odczytu danych. W usługach REST w przypadku żądań XML jest rzadziej stosowany gdyż często wystarczy adres URL, w przypadku odpowiedzi zazwyczaj jest ona zwracana w formacie XML lub JSON.

Odpowiedź serwera w REST zwracająca informacje o użytkownikach samochodów:

W formacie XML:

```
<listaSamochodow>
 <samochod nrTablicyRej="ST4281A">
  <imieWlasciciela>Jan</imieWlasciciela>
  <nazwiskoWlasciciela>Nowak</nazwiskoWlasciciela>
  <marka>Audi A8</marka>
  <rokRejestracji>2015</rokRejestracji>
  <opis>
   Samochód firmowy
  </opis>
```

```
 </samochod>
  <samochod nrTablicyRej="SK12811">
  <imieWlasciciela>Maria</imieWlasciciela>
  <nazwiskoWlasciciela>Szpak</nazwiskoWlasciciela>
  <marka>Fiat Panda</marka>
  <rokRejestracji>2014</rokRejestracji>
  <opis>
   Samochód prywatny
  </opis>
 </samochod>
</listaSamochodow>
```

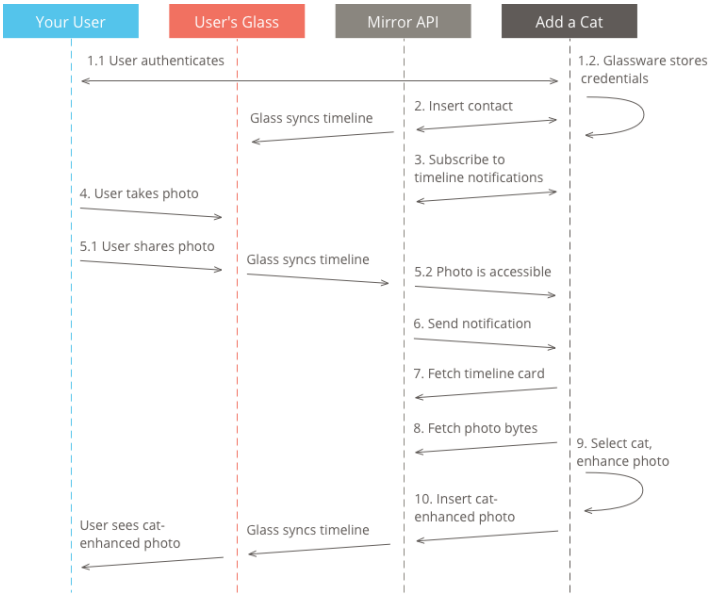
W formacie JSON:

```
{
  "listaSamochodow": {
    "samochod": [
      {
        "nrTablicyRej": "ST4281A",
        "imieWlasciciela": "Jan",
        "nazwiskoWlasciciela": "Nowak",
        "marka": "Audi A8",
        "rokRejestracji": "2015",
        "opis": "Samochód firmowy"
      },
      {
        "nrTablicyRej": "SK12811",
        "imieWlasciciela": "Maria",
        "nazwiskoWlasciciela": "Szpak",
        "marka": "Fiat Panda",
        "rokRejestracji": "2014",
        "opis": "Samochód prywatny"
      }
    ]
  }
}
```

Dużą zaletą wykorzystania REST jest brak przymusu wykorzystywania określonej technologii zwracanych danych. Może to być zarówno XML, podobnie jak w przypadku SOAP, ale również często używany JSON lub nawet tekst rozdzielany średnikami czy innymi znakami. Natomiast każdy z formatów ma swoje zalety, do których należą np. w przypadku XML łatwa rozszerzalność, a JSON – duża ilość oprogramowania klienckiego czytającego strukturę. Trzeba zaznaczyć, że format zwracanych danych z modelu REST jest z jednej strony zrozumiały do pewnego stopnia dla człowieka, natomiast nie powinien zawierać informacji nie powiązanych z danymi np. stylów wyświetlania jak to jest np. w języku HTML.

Często poważne firmy udostępniają zarówno tradycyjne API w modelu SOAP jak również w czystym formacie REST, na pewno warto wspomnieć Google Glass Mirror API dostępny dla trzech języków programowania w tym Java, PHP i Python. Na rysunku 8 pokazano przykładowy diagram przepływu dla aplikacji stworzonej z wykorzystaniem tego API dla usługi „Add a Cat to That".

Rysunek 8. Diagram przepływu w aplikacji wykorzystującej REST API udostępnione przez Google Glass Mirror API.




Źródło: Google Glass

– Twój użytkownik uwierzytelnia się z wykorzystaniem OAuth 2.0. Twoja usługa zapisuje dane użytkownika.
– Po uwierzytelnieniu usługa dodaje kontakt o nazwie „Add a Cat to That".
– Następnie Twoja usługa zapisuje się na aktualizacje w linii czasu użytkownika.

– Za jakiś czas użytkownik wrzucił zrobił zdjęcia.
– Użytkownik udostępnia zdjęcie w „Add a Cat to That". To tworzy linię czasu powiązaną z Twoją usługą.
– Ponieważ Twoja usługa obserwuje aktualizację linii czasu, jest powiadamiana o aktualizacji. Łącze do aktualizacji zawiera udostępnione zdjęcie.
– Usługa sprawdza powiadomienie zawierające id zdjęcia.
– Pobiera zdjęcie.
– Twoja usługa wybiera losowe zdjęcie kota i tworzy udostępnioną zawartość.
– Ostatecznie usługa tworzy nową linię czasu i wstawia w linię czasu użytkownika z załączonym zdjęciem.

Kolejnym przykładem API udostępnionym w modelu REST jest Twitter Public API, które udostępnia programistyczny dostęp w trybie do odczytu i zapisu danych na Twitterze. API Twittera również wykorzystuje protokół OAuth do identyfikacji użytkowników, z tą różnicą, że odpowiedzi są wysyłane w formacie JSON.

Również Amazon.com udostępnia wiele usług REST np. dla usługi *Simple Storage Service*, która w zamyśle jest internetową platformą składowania danych. Z usługi można korzystać poprzez prosty i intuicyjny interfejs strony internetowej *AWS Management Console* lub skorzystać z udostępnianego API, dzięki czemu możemy zautomatyzować wiele operacji.

## 2.6. AJAX i REST

AJAX to technologia bazująca na Javascript, pozwalająca na tworzenie interaktywnych witryn bez potrzeby odświeżania całej strony. Żądanie wysyłane jest za pomocą obiektów XMLHttpRequest do serwera, natomiast odpowiedź jest przetwarzana przez Javascript, który aktualizuje (lub nie) wybrane elementy strony (aplikacji internetowej). Każde żądanie XMLHttpRequest możemy podejrzeć jako usługę REST wysyłającą zapytanie metodą GET. Wynik bardzo często zwracany jest w formacie JSON, szczególnie popularnym w przypadku usług w modelu REST. Aby aplikacja była w pełni zgodna z REST, powinno przestrzegać się zasad projektowania REST, co ma pozytywny wpływ na jej jakość, ponieważ zawiera wiele dobrych wzorców oraz zasad projektowania i implementowania aplikacji.

### 2.6.1. Wzorce i szkielet do budowy aplikacji w modelu REST

Wzrost popularności i wykorzystania REST spowodował wysyp szkieletów wspomagających tworzenie aplikacji i wpływających na czas ich pisania. W dalszej części przedstawione zostaną popularne szkielety aplikacji w Restlet (Java), Django (Python), Ruby on Rails (Ruby) i Slim (mikro szkielet PHP). REST sam w sobie nie jest architekturą ale dobrą praktyką prowadzącą do uproszczenia budowania i udostępniania usług sieciowych, z których później korzystają aplikacje mobilne.

ROA (ang. *Resource Oriented Architecture* – architektura zorientowana na zasoby) wymaga podziału domeny aplikacji na zasoby, które istnieją w ujęciu

koncepcyjnym (abstrakcyjnym). Jednym ze sposobów na zbudowanie aplikacji wraz z API jest budowa od podstaw, jednak często najlepszym sposobem jest wykorzystanie gotowego szkieletu, który ułatwi i przyspieszy udostępnienie planowanego rozwiązania.

**Restlet**

Szkielet Restlet jest jednym z częściej używanych rozwiązań open source dla programistów języka Java, którzy planują pisać i używać API. Wykorzystanie szkieletu Restlet pozwala na budowanie dobrych jakościowo interfejsów programistycznych wykorzystujących architekturę REST. Na pewno ważnym aspektem jest duże środowisko programistów wykorzystujących ten szkielet i duża dostępność różnego rodzaju materiałów w Internecie. Nie da się ukryć że udostępnienie jako *open source* ogranicza koszty wdrożenia – jest za darmo do ściągnięcia i można go używać w oparciu o licencje ASL (Apache Software License).

Zaawansowane możliwości, zunifikowany klient i serwer Java API pozwala na budowanie łatwo rozszerzalnych rozwiązań z wykorzystaniem modelu REST. Dostępny jest dla wielu najważniejszych platform jak (Java, Google AppEngine, GWT czy Android) oraz dostępnych jest bardzo wiele rozszerzeń szkieletu.

**Ruby on Rails**

Uproszczenie jest głównym powodem sukcesu szkieletu Ruby on Rails. Podejście w Rails jest, aby nie udostępniać dużej liczby narzędzi do osiągnięcia określonych celów ale w jeden sposób ukończyć wykonywanie wielu codziennych zadań. Dzięki Rails szybko udostępnimy dane przechowywane w relacyjnej bazie danych, pod warunkiem, że dane są uporządkowane i mają określone nazwy, podobnie jeżeli planuje się wykorzystać model MVC (model-widok-kontroler). Ponieważ bardzo często przy budowie aplikacji internetowych spotyka się te same wyzwania, to skorzystanie z Ruby on Rails ułatwi wiele codziennych prac programistycznych. Poprzednie wersje Rails wykorzystywały architekturę hybrydową, ale celem jest jak największa zgodność z modelem REST.

**Django**

Django to kolejny szkielet pozwalający na łatwy rozwój aplikacji sieciowych i usług, ale tym razem dla języka Python. Projekt szkieletu zbliżony jest do Rails, czyli w jeden sposób pozwala na obsługę wielu typowych zadań czekających na programistów. Można skorzystać z podstawowych procedur ROA zamiany zbioru danych w zasób REST i korzystać z niego bezpośrednio w Django. Atutem szkieletu jest nacisk na kontekst zabezpieczeń, co pozwala na uniknięcie wielu błędów robionych przez początkujących (i nie tylko) programistów.

**Slim PHP – mikro szkielet**

Slim jest w pełni funkcjonalnym, udostępnianym jako oprogramowanie open source mikro szkieletem dla języka PHP. Wyrafinowana architektura kierowania i architektury pośredniczącej powoduje że jest idealny dla aplikacji internetowych i protypowania API.

Slim pozwala na szybkie tworzenie i udostępnianie API w modelu REST wraz z uwierzytelnianiem i złożonymi żądaniami / odpowiedziami w najczęściej wykorzystywanych formatach.

## 2.7. Podsumowanie

Głównym elementem architektury REST jest zasób. Jest on identyfikowany poprzez URI. Zarówno stan jak i funkcjonalność reprezentowane są poprzez zasoby inaczej niż się to dzieje w przypadku metod czy usług wykorzystywanych w RPC czy SOAP. Nie wydaje się poleceń w stylu *pobierzNazweSamochodu*, a następnie *pobierzCeneSamochodu*. Dane o samochodzie udostępniane są jako zasób, który powinien zawierać wszystkie wymagane informacje (i / lub linki do nich). W związku z tym zasoby nie powinny zawierać również zbędnych informacji, raczej te, które są najczęściej wykorzystywane, a do dodatkowych powinny być użyte łącza.

Nie używa się również stanu połączenia, czyli interakcje są bezstanowe. Każde wywołanie powinno zawierać komplet informacji w celu zwrócenia określonych wyników i nie musi być zależny od poprzednich żądań. Zasoby powinny dać się zapisywać w pamięci podręcznej wraz z informacją o terminie wygaśnięcia. Klient usługi powinien przestrzegać informacji o okresie przechowywania dla każdego zasobu, z którego korzysta. Warto wspomnieć że np. aplikacja mobilna nie musi mieć nic wspólnego z modelem REST i bez problemu korzystać z usługi w tym stylu (REST) zaprojektowanej. Zastosowanie modelu w całości wraz z ograniczeniami jest: proste, skalowalne, wydajne i bezpieczne.

Usługi w modelu REST wyrosły jako alternatywa dla usług opartych na SOAP i dzięki prostocie, lekkości i możliwości transmisji danych bezpośrednio poprzez protokół HTTP szybko zyskały przychylność wielu programistów i są wykorzystywane szeroko przez największych dostawców aplikacji sieciowych i mobilnych. Wielowarstwowa architektura aplikacji sieciowych pozwala na łatwe budowanie zarówno aplikacji z GUI dla człowieka np. na platformy IOS czy Android, jak również zautomatyzowane systemy które wymieniają informacje pomiędzy sobą bez udziału człowieka. Separacja interfejsu od logiki i warstwy danych zwiększa bezpieczeństwo i ułatwia wyszukiwanie potencjalnych błędów w aplikacjach. Dodatkowo łatwość użycia w tandemie z technologią AJAX rozszerza jej zastosowanie do obsługi dynamicznych serwisów internetowych których klientem nie musi być tylko przeglądarka internetowa.

# 3

# METODY SKŁADOWANIA DANYCH WYKORZYSTYWANE W APLIKACJACH MOBILNYCH

## 3.1. Architektura rozwiązań składowania danych w aplikacjach mobilnych

Dane są integralną częścią dzisiejszych aplikacji przeznaczonych na urządzenia mobilne. W związku z tym, natywne mechanizmy zaimplementowane w dzisiejsze systemy operacyjne przeznaczone na telefony komórkowe, tablety i inne urządzenia przenośne pozwalają na wykorzystanie mechanizmów zarządzania danymi. W wielu przypadkach, gdy zachodzi konieczność zastosowania danych w formie strukturalnej stosowany jest relacyjny model danych. Relacyjne bazy danych są wykorzystywane powszechnie jako warstwa składowania danych w wielu rozwiązaniach aplikacyjnych już od lat 70. ubiegłego stulecia.

Ostatnie lata przyniosły jednak urozmaicenie w dziedzinie modeli przechowywania i przetwarzania danych. Pojawiający się w ostatnich latach nurt „NoRel" nakazuje postrzeganie różnych form i metod składowania danych w sposób komplementarny i wykorzystanie takiego modelu danych, który jak najbardziej pasuje do opisu specyfiki implementowanego problemu. Nurt „NoSQL" (ang. *Not Only SQL*)[22], który doprowadził do rozpowszechnienia innych modeli danych i języków zapytań do baz danych, oraz przyczynił się do powstania narzędzi pozwalających na współpracę warstwy aplikacji z danymi przechowywanymi w innych postaciach[23] niż postać relacyjna. Podejście pozwala często na osiągnięcie lepszej efektywności czasowej aplikacji[24].

Modele danych oparte o reprezentacje wchodzące w skład nurtu NoRel są stosowane w informatycznych systemach rozproszonych i z powodzeniem

---

[22] Shashank Tiwari, *Professional NoSQL*; Wrox 2012.

[23] P.J. Sadalage, M.Fowler; *NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence*. Addison-Wesley Professional 2012.

[24] Por. P. Buchwald, M. Rostański, A. Jurasz, *Comparative Analysis of Database Solutions in Applications with the Android Operating System*, [w:] P. Pikiewicz, M. Rostanski (Eds.), *Internet in the information Society. Computer Systems Architecture and Security*, Academy of Business in Dabrowa Gornicza 2013, s. 79-92 *lub Relative an"d non-relative databases performance with an Android platform application"* s. 224-238 (Theoretical and Applied Informatics 2/2013) tychże autorów.

mogą być zastosowane w projektowaniu warstwy składowania danych dla aplikacji mobilnych. Jednym z przykładów baz danych NoRel jest baza danych Neo4j, która wykorzystuje graf jako postać reprezentacji danych[25]. Zastosowanie w implementacji aplikacji przeznaczonych na urządzenia mobilne innych niż relacyjny model danych ułatwia obecność w nowoczesnych językach programowania szeregu narzędzi i bibliotek[26], które pozwalają na budowę warstwy pośredniczącej pomiędzy elementami aplikacji przetwarzającymi dane, a tym samym i mechanizmami odpowiedzialnymi za przechowywanie danych[27]. Podejście wykorzystujące warstwę pośredniczącą, która ułatwia aplikacji dostęp do danych przedstawiono na rysunku 9.

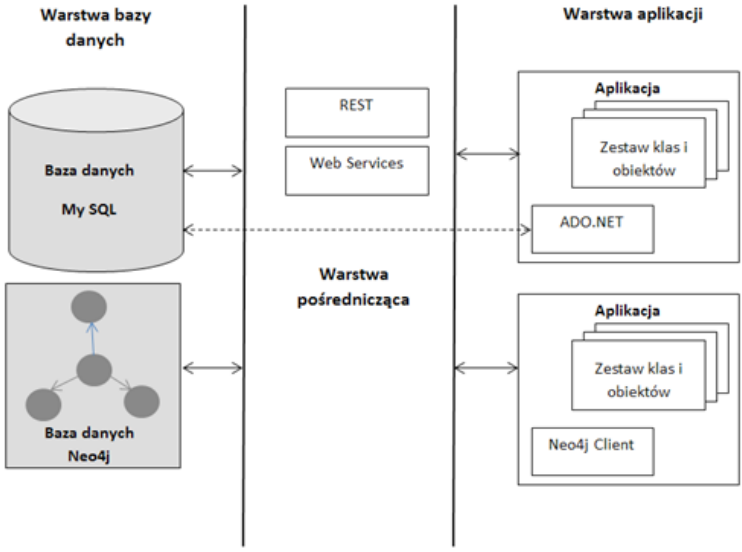
Podczas projektowania warstwy składowania danych dla aplikacji przeznaczonych na urządzenia mobilne oprócz modelu danych ważny jest również dobór prawidłowej architektury związanej z lokalizacją danych[28]. W tym przypadku możliwe są trzy rozwiązania:

- Lokalna baza danych – baza danych jest zaimplementowana w oparciu o usługi zarządzania danymi, które są obecne w systemie operacyjnym urządzenia mobilnego, lub działają jako usługi dodatkowych aplikacji zainstalowanych na urządzeniu mobilnym. To rozwiązanie pozwala na korzystanie z usług zarządzania danymi przez wiele aplikacji, które działają na urządzeniu mobilnym.
- Lokalna baza danych wbudowana w aplikację – takie rozwiązanie pozwala na wykorzystanie bibliotek API odpowiedzialnych za funkcjonalność zarządzania danymi, oraz ich składowania. Daje to dużą elastyczność w wykorzystaniu modelu danych, oraz doboru bibliotek do zarządzania danymi. W takim rozwiązaniu dostęp do danych odbywa się poprzez wyspecjalizowane obiekty bibliotek. Z tego względu rozwiązanie umożliwia zarządzanie danymi w obrębie jednej aplikacji. Aplikacja chcąc udostępniać dane innym komponentom zainstalowanym na urządzeniu mobilnym musi posiadać implementację swoich obiektów zarządzania treścią.
- Zdalna baza danych – poprzednie rozwiązania opierały się o wykorzystanie jedynie zasobów lokalnych urządzenia mobilnego. W wielu przypadkach ograniczenia sprzętowe urządzenia – dostępna pamięć i moc obliczeniowa nie pozwalają na przetwarzanie dużej liczby danych. Jeśli zachodzi taka potrzeba konieczne jest zastosowanie zewnętrznego mechanizmu składowania danych uruchomionego na serwerze, który pozwoli na obsługę żądań kierowanych wprost z aplikacji. W takim scenariuszu wykorzystuje się również warstwy pośredniczące, które ułatwiają uniezależnienie warstwy aplikacji od użytego modelu danych

[25] Bardzo dobrze opisane w: I. Robinson, J. Webber, E. Eifrem, *Graph Databases*; O'Reily 2013.
[26] P. Buchwald, *Obiektowe mechanizmy przechowywania informacji dostarczanych przez systemy nadrzędnej kontroli dystrybucji danych jako alternatywa dla rozwiązań opartych o relacyjne bazy danych, XII Konferencja Systemów Czasu Rzeczywistego* (SCR 2005), s. 131-142.
[27] Zgodnie z dokumentacją techniczną bazy danych Neo4j: http://neo4j.com/docs
[28] Zgodnie z dokumentacją techniczną bazy danych IBM DB2 Everywhere.

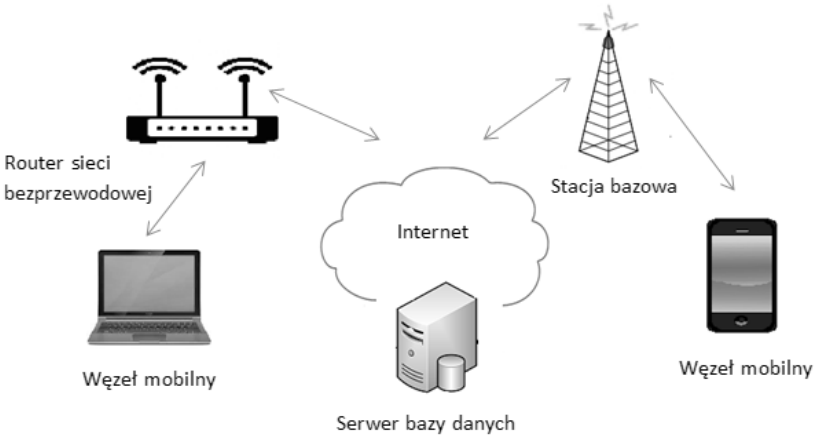
i pozwalają na późniejszą zamianę warstwy składowania danych. Dostęp do usług przetwarzania danych jest możliwy z zastosowaniem sieci komputerowych.

Rysunek 9. Sposób dostępu aplikacji do danych z użyciem warstwy pośredniczącej




– Model hybrydowy wykorzystujący bazę danych zdalną i kopie lokalne jest z kolei modelem typowym dla rozproszonych systemów przetwarzania danych wykorzystujących urządzenia mobilne. W takim modelu architektonicznym możemy wyróżnić trzy typy urządzeń: węzły stacjonarne, węzły mobilne, infrastrukturalne węzły sieciowe. Węzły mobilne to przenośne komputery i urządzenia mobilne typu smartphone, czy tablet. Infrastrukturalne węzły sieciowe to urządzenia zapewniające dostęp do sieci bezprzewodowej i pozwalające na komunikację pomiędzy węzłami stacjonarnymi a urządzeniami mobilnymi. Są to najczęściej stacje bazowe sieci komórkowych, lub routery wykorzystywanych sieci bezprzewodowych. W tym modelu architektonicznym węzły mobilne posiadają usługi baz danych, które pozwalają na synchronizację z lokalną bazą danych, jak i efektywne zarządzanie transakcjami. Koncepcja takiego modelu architektury przetwarzania danych została przedstawiona na rysunku 10.

Rysunek 10. Model połączenia aplikacji mobilnych z bazą danych




Ze względu na swój mobilny charakter rozwiązania dotyczące składowania danych dla aplikacji przeznaczonych na urządzenia mobilne muszą dopuszczać istnienie pewnych dodatkowych problemów związanych z dostępem do danych, jak i z ich przetwarzaniem. Do najważniejszych problemów jakie występują podczas projektowania rozwiązań składowania danych dla urządzeń mobilnych należą:

- Ograniczone zasoby urządzenia mobilnego – Pomimo znacznego wzrostu mocy obliczeniowej i pamięci urządzeń mobilnych są one o wiele uboższe od dedykowanych urządzeń klasy stacje robocze i serwery podłączone do dzisiejszej sieci Internet. Ze względu na rozmiar wielu używanych w praktycznych zastosowaniach baz danych zastosowanie urządzeń mobilnych do przetwarzania danych pochodzących z rzeczywistych baz danych wiąże się niejednokrotnie z koniecznością zastosowania kopii części danych do pamięci podręcznej urządzenia z jednoczesnym uproszczeniem wykonywanych na urządzeniu operacji.
- Konieczność oszczędzania energii – Jednym z największych ograniczeń urządzeń mobilnych w stosunku do stacjonarnych jednostek komputerowych przetwarzania danych jest ograniczony czas ich działania, oraz konieczność ładowania baterii. W celu wydłużenia czasu pracy urządzenia mobilnego na jednym cyklu ładowania stosuje się algorytmy pozwalające na mniejszą liczbę dostępu do zasobów zdalnych poprzez sieć, oraz optymalizację metod przetwarzania danych. Optymalizacja pod kątem wydajności zużycia energii jest ważnym aspektem projektowania każdej aplikacji mobilnej, także rozwiązań dotyczących przetwarzania danych.
- Problemy z połączeniem sieciowym – moduły sieci GSM umieszczone w dzisiejszych urządzeniach mobilnych pozwalają na permanentne połączenie z siecią Internet. Urządzenia mobilne są coraz częściej wyposażone w komponenty sprzętowe umożliwiające współpracę z sieciami

3G i LTE. Rozwiązania te oferują przepustowości pozwalające pracować z urządzeniem przenośnym w sposób podobny jak ze stacjonarną stacją roboczą. Rozwiązania bazujące na urządzeniach przenośnych są jednak częściej narażone na problemy z brakiem połączenia sieciowego. Warunki pogodowe, ukształtowanie terenu czy zakłócenia sieci bezprzewodowych mogą utrudnić korzystanie z sieci bezprzewodowej. Podczas poruszania się urządzenia mobilnego moc odbieranego sygnału sieci bezprzewodowych w miejscach takich jak tunele, niektóre budynki czy tereny o mniejszym zagęszczeniu stacji bazowych może być znacznie niższa. Interakcja pomiędzy bazą danych a urządzeniem mobilnym zależy w dużym stopniu od możliwości wykorzystania połączenia sieciowego, oraz parametrów połączenia. W wielu przypadkach może okazać się istotna nie tylko przepustowość łącza, ale również opóźnienie, czy fluktuacje parametrów w czasie. Problemy związane z połączeniem sieciowym w metodach przetwarzania danych dedykowanych dla urządzeń mobilnych są rozwiązywane poprzez zastosowanie lokalnych kopii fragmentów baz danych, oraz algorytmy zarządzania transakcjami rozwiązujące konflikty podczas modyfikacji danych.

## 3.2. Metody dostępu do danych dla urządzeń mobilnych

Podczas projektowania metod zarządzania danymi dla potrzeb aplikacji mobilnych często wykorzystuje się pośredniczące obiekty zwane dostawcami treści. Obiekty te są zgodne z koncepcją działania systemów ORM, jednak ich zastosowanie może być wykorzystane we współpracy z innymi modelami danych niż postać relacyjna. Obiekty pełniące rolę dostawców treści są powszechnie używane podczas konstruowania metod dostępu do danych w aplikacjach przeznaczonych dla systemu operacyjnego Android – najpopularniejszego systemu operacyjnego dla urządzeń mobilnych. Do najważniejszych obiektów dostawców treści należą:

- Dostawca treści kontaktów zapisanych w urządzeniu mobilnym;
- Dostawca informacji o historii wykonywanych przez użytkownika połączeń;
- Dostawca danych multimedialnych (pliki dźwiękowe, pliki graficzne, pliki video);
- Dostawca informacji o ustawieniach urządzenia mobilnego.

Niezależnie od użytego modelu przechowywania danych obiekty dostawców treści pozwalają na ujednolicony mechanizm dostępu do danych z jednoczesnym uwzględnieniem aspektów bezpieczeństwa i ochrony informacji. Takie podejście pozwala na oddzielenie warstwy aplikacji i warstwy dostępu do danych. Obiekty dostawców treści dostarczają interfejsów dostępu identyfikowanych poprzez adresy URI rozpoczynające się od prefiksu „content://". Mechanizm ten pozwala na wymianę danych pomiędzy aplikacjami bez konieczności zarządzania fizycznym dostępem do warstwy składowania danych. Wykorzystanie dostawców treści jest jedną z powszechnie stosowanych metod zarządzania danymi zapisanymi lokalnie na urządzeniu mobilnym.

### 3.2.1. Wbudowane mechanizmy systemu Android do przechowywania danych zgodnych z modelem relacyjnym

Aplikacje dla urządzeń mobilnych, które wymagają zastosowania solidnego mechanizmu zarządzania danymi zapisanymi w sposób ustrukturyzowany mogą używać metod dostępu do relacyjnej bazy danych SQLite. Jest to relacyjna baza danych dystrybuowana na licencji Open Source. Ze względu na ich kompaktowy charakter i zastosowanie plików jako sposobu składowania danych konsumują one stosunkowo niedużo zasobów sprzętowych urządzenia mobilnego i mogą być wykorzystywane jako solidny sposób składowania i przetwarzania danych na urządzeniu lokalnym. Podstawowa wersja bazy danych SQLite wymaga od 250kB pamięci, co pozwala na jej zastosowanie jako usługę wbudowaną w system operacyjny Android. Pewność dysponowania mechanizmem przetwarzania danych w postaci relacyjnej bez konieczności instalowania dodatkowych komponentów czyni bazę danych SQLite podstawowym mechanizmem składowania i przetwarzania danych strukturalnych. Mechanizm dostępu do danych używający baz danych SQLite pozwala również na rozdzielenie danych prywatnych, które są dostępne w obrębie jednej aplikacji, oraz danych publicznych dystrybuowanych poprzez obiekty dostawców treści. Przetwarzanie danych zawartych w relacyjnym modelu z zastosowaniem bazy SQLite może być zrealizowane poprzez zastosowanie bibliotek SDK, które zwalniają programistów z implementowania zapytań SQL na rzecz konstrukcji składniowych tworzonych w językach programowania wysokiego poziomu takich jak opisywana w pierwszym rozdziale *java*.

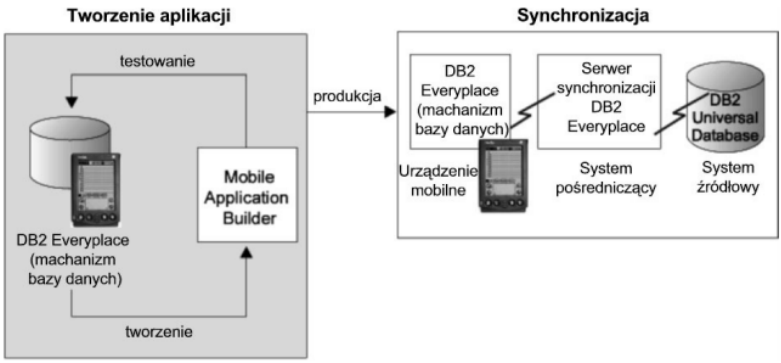
## 3.3. Przykłady relacyjnych baz danych wykorzystywanych w rozwiązaniach mobilnych oferowanych przez innych producentów

Do najpopularniejszych systemów baz danych, które są oparte o model relacyjny i oferują swoje wersje przeznaczone dla urządzeń mobilnych należą: IBM DB2 Everyplace, Oracle Lite, Microsoft SQL Server CE, Sybase SQL.

Mobilna baza danych IBM DB2 Everyplace to baza oparta o model relacyjny i przeznaczona do instalacji na urządzenia mobilne. Oferuje ona możliwość dostępu do danych za pomocą uniwersalnych interfejsów programistycznych, takich jak JDBC (ang. Java Database Connectivity), ODBC (Open database Connectivity), metod związanych z biblioteką ADO.NET, oraz obsługiwanego zestawu funkcji CLI (Call Level Interface). Zestaw wspieranych interfejsów czyni to narzędzie łatwym do integracji z warstwą aplikacyjną dla każdej systemowej platformy przeznaczonej dla urządzeń mobilnych. Dzięki rozwiązaniu IBM DB2 Everyplace możliwa jest synchronizacja danych korporacyjnych pomiędzy urządzeniem mobilnym a centralną bazą danych. Rozwiązanie to oferuje kompleksowe podejście do zarządzania danymi w rozproszonym środowisku wykorzystującym aplikacje mobilne. Komponenty oferowane w ramach narzędzia IBM DB2 Everywhere zapewniają wsparcie podczas całego cyklu życia systemu przetwarzania danych od projektowania aplikacji przetwarzania danych po jej instalację i prawidłowe działanie uwzględniające synchronizację

i bezpieczną aktualizację danych. Sposób współpracy pomiędzy poszczególnymi komponentami rozwiązania IBM DB2 Everywhere przedstawia rysunek 11.

Rysunek 11. Sposób współpracy komponentów w ramach systemu IBM DB2 Everywhere




Do najważniejszych komponentów systemu IBM DB2 Everywhere należą:

- Mobilna baza danych DB2 Everywhere – jest to oprogramowanie uruchomione na urządzeniu mobilnym, które pozwala na wykorzystanie lokalnych kopii fragmentów bazy danych, oraz współpracę z danymi nawet w trybie rozłączenia. Zapewnia możliwość bezpiecznej modyfikacji danych przez warstwę aplikacyjną
- Serwer synchronizacji – Jest narzędziem zbudowanym w architekturze klient – Server. Pozwala na zarządzanie dwukierunkową synchronizacją danych pomiędzy lokalnymi kopiami danych a centralnym repozytorium. Serwer synchronizacji pośredniczy między oprogramowaniem klienta synchronizacji na urządzeniu mobilnym a bazą danych. Współpracuje też z oprogramowaniem centrum administrowania urządzeniami mobilnymi. Dzięki jego funkcjonalności możliwe jest zdefiniowanie które z podzbiorów danych zgromadzonych w centralnej bazie danych będą mogły być udostępniane poszczególnym grupom użytkowników za pośrednictwem aplikacji mobilnych.
- Klient synchronizacji – Działa bezpośrednio na urządzeniu mobilnym. Stanowi zestaw interfejsów API wbudowanych w aplikację. Za pomocą tych interfejsów aplikacje przeznaczone na urządzenia mobilne mogą w sposób bezpieczny obsługiwać synchronizację z serwerem centralnym i modyfikację danych.
- Mobile Application Builder – Narzędzie programistyczne umożliwiające tworzenie i testowanie aplikacji przetwarzania danych przeznaczonych dla urządzeń mobilnych. Na uwagę zasługuje fakt, iż narzędzie jest rozpowszechniane bezpłatnie przez firmę IBM[29].

Wsparcie dla rozwiązań dedykowanych dla urządzeń mobilnych oferuje również baza danych SQL Server. Narzędzie SQL Server CE daje możliwość

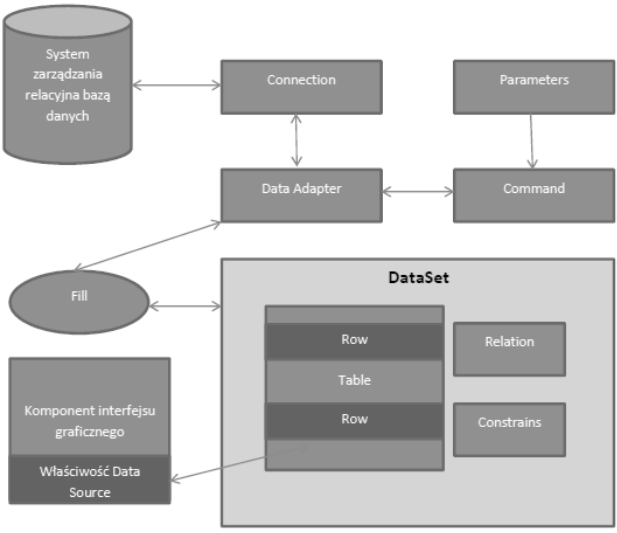
[29] Zgodnie z dokumentacją techniczną bazy danych IBM DB2 Everywhere, http://www-01.ibm.com/support/docview.wss?uid=swg27009474

zarządzania danymi na urządzeniu mobilnym, oraz oferuje wspólny model programowania dla budowy aplikacji rozproszonych opartych o urządzenia stacjonarne, oraz przeznaczone na urządzenia przenośne. Używając bazy danych SQL Server CE można wykorzystać większość programistycznych technologii dostępnych również w stacjonarnej wersji bazy danych SQL Server.

W niektórych przypadkach zniwelowanie niedogodności związanych z brakiem permanentnego połączenia ze stacjonarną bazą danych może być zrealizowane z użyciem tzw. trybu bezpołączeniowego wspieranego przez bibliotekę ADO.NET. Tryb ten zakłada wykorzystanie kopii lokalnej części bazy danych, która jest umieszczona w pamięci dostępnej dla warstwy aplikacji. Lokalna kopia bazy danych jest obsługiwana dzięki wprowadzeniu przez bibliotekę ADO.NET zestawu klas pozwalających na odzwierciedlenie danych z relacyjnej bazie danych w postaci obiektowej. Bogaty zestaw przestrzeni nazw pozwala na pracę nie tylko z systemami bazodanowymi firmy Microsoft, ale również ze źródłami danych w postaci plików płaskich, arkuszy kalkulacyjnych, relacyjnych baz danych innych producentów jak Oracle, IBM DB2. Tryb bezpołączeniowy, w odróżnieniu od trybu połączeniowego nie wymaga ciągłego połączenia ze zdalnym serwerem. Zastosowanie przy obsłudze operacji dostępu do danych trybu połączeniowego charakteryzuje się tym, iż dane są pobierane lub aktualizowane wiersz po wierszu, a ewentualne przerwy w dostępie do baz danych powodują niemożliwość wykonywania zadań przetwarzania danych po stronie aplikacji. Podczas pracy z danymi w trybie bezpołączeniowym jest realizowany odmienny scenariusz postępowania. Początkowo podobnie jak w trybie połączeniowym należy utworzyć obiekt *connection*. Następnie za pomocą obiektu *command* należy utworzyć komendy pozwalające na aktualizację, kasowanie, wstawianie i pobieranie rekordu z tabeli. Każda tabela w bazie danych jest powiązana z odpowiadającym jej obiektem *Table* w lokalnej kopii obsługiwanej za pomocą klasy *DataSet* poprzez komponent *DataAdapter*. Pobranie z bazy danych do tabeli będącej w obiekcie *DataSet* odbywa się poprzez wywołanie metody *Fill* na rzecz tego obiektu. Z wykorzystaniem metody *update* można dokonać synchronizacji danych w *DataSet* z bazą danych do której zostało przypisane połączenie. W takim scenariuszu pracy z danymi raz pobrane dane mogą być wykorzystywane wiele razy przez komponenty wizualizacyjne warstwy aplikacji. Komponenty te mają za zadanie z jednej strony powiązanie z danymi pobranymi z bazy w celu ich wizualizacji, a z drugiej strony obsługę zdarzeń wynikających z interakcji budowanego rozwiązania aplikacyjnego z użytkownikiem. Każdy z komponentów wizualizacyjnych ma właściwość DataSource. Za jej pomocą można powiązać kontrolkę interfejsu graficznego użytkownika z danymi umieszczonymi w obiekcie *Table* lub *DataSet.* Taką samą metodę powiązania komponentów graficznych z danymi można wykorzystać w przypadku innych typów danych wspieranych przez standardowe przestrzenie nazw platformy .Net, takich jak listy, tablice czy zbiory. Zunifikowany scenariusz postępowania w celu realizacji wizualizacji danych z pomocą standardowych kontrolek, na podstawie których budowany jest graficzny interfejs użytkownika upraszcza

budowanie aplikacji opartych o relacyjne bazy danych. Scenariusz współpracy komponentów obiektowych wykorzystywanych w trybie bezpołączeniowym biblioteki ADO.NET przedstawiono na rysunku 12. Dostępność trybu bezpołączeniowego zwalnia jednocześnie z konieczności posiadania ciągłego połączenia z bazą danych. Możliwość przechowywania lokalnej kopii części danych w warstwie aplikacji pozwala sprostać wymaganiom mobilnych baz danych w zakresie rozwiązania problemu z chwilowym brakiem dostępu do sieci. Niestety mechanizm ten nie posiada wbudowanych rozwiązań pozwalających na obsługę kolizji transakcji, oraz priorytetyzacji operacji na danych w bazie centralnej oraz kopiach umieszczonych na węzłach mobilnych. W wielu przypadkach istnieje więc konieczność zastosowania wersji bazy danych SQL Server CE dla urządzeń mobilnych.

Rysunek 12. Współpraca ze źródłem danych z wykorzystaniem trybu bezpołączeniowego ADO.NET.




Jest to kompaktowa baza danych oparta o model relacyjny, która pozwala na łatwą integrację z modelem obiektowym i platformą .NET. Dzięki wykorzystaniu tego narzędzia możliwe jest ujednolicenie scenariusza projektowania aplikacji przetwarzania danych dla urządzeń mobilnych oraz stacjonarnych. Środowisko SQL Server CE jest złożone z następujących komponentów:

- Silnik bazy danych SQL Server CE – komponent odpowiedzialny za zarządzanie danymi na urządzeniu mobilnym. Pozwala na replikację danych z serwera głównego, oraz posiada funkcjonalność śledzenia zmian lokalnych.
- SQL Server Client Agent – komponent programistyczny, który udostępnia zestaw obiektów pozwalających na współpracę z bazą danych, takich jak: Replication, RemoteDataAccess i Engine. Poprzez użycie tych obiektów aplikacje używające bazy daych SQL Server CE mogą mieć kontrolę nad połączeniem z serwerem i wymianą danych.

– Server Agent – komponent usługowy, który pośredniczy w wymianie danych pomiędzy serwerem centralnym a kopią lokalną danych. Jest odpowiedzialny za przyjmowanie żądań od agentów klienckich, oraz zwracanie do nich rekordów wynikowych.

Microsoft SQL Server dla urządzeń mobilnych pozwala na replikację danych na urządzenie mobilne z centralnej bazy danych poprzez dwa mechanizmy:

– Mechanizm RDA (ang. *Remote Data Access*) – mechanizm ten może być użyty w celu modyfikacji struktury danych przez aplikację mobilną na zdalnym serwerze. W tym celu aplikacja wysyła żądanie wykonania polecenia języka Data Manipulation Language. Warstwa aplikacyjna może także wysłać żądanie realizacji zapytania SQL. Wynik tego zapytania w postaci zbioru rekordów może być użyty do stworzenia lokalnej kopii danych.
– Replikacje scalające (ang. *Merge replication*) – ta metoda pozwala na jednoczesne aktualizacje danych na urządzeniu mobilnym, oraz serwerze centralnym. Dane mogą być zsynchronizowane podczas nawiązania połączenia z serwerem. Możliwe jest zdefiniowanie replikacji danych tylko do odczytu dla takich podzbiorów danych, które nie powinny być modyfikowane przez aplikację mobilną. Dodatkowo można zdefiniować lokalne podzbiory danych dla różnych grup mobilnych użytkowników z zastosowaniem filtracji danych. Metoda replikacji scalających jest zgodna z wprowadzonym do serwerowych wersji narzędzia modelem replikacji danych.

### Architektura replikacji danych SQL Server

W celu realizacji replikacji danych pomiędzy centralnym serwerem a bazą danych urządzenia mobilnego konieczne jest zdefiniowanie topologii przesyłania danych. Metody replikacji danych w oparciu o SQL Server są zrealizowane poprzez implementację scenariusza publikacji i subskrypcji. Serwer wysyłający informację jest publikatorem danych, natomiast serwer odbierający dane jest subskrybentem. Przygotowanie replikacji i synchronizacji danych pomiędzy serwerem a urządzeniem mobilnym wymaga zdefiniowania następujących węzłów topologii replikacji:

– Nadawca (ang. *Publisher*) – jest to wskazanie na instancje bazy danych, która będzie odpowiedzialna za dystrybuowanie części przechowywanych danych i obiektów bazodanowych do mobilnego urządzenia.
– Węzeł dystrybucyjny (ang. *Distributor*)– jest to instancja, która stanowi miejsce składowania danych pochodzących od jednego lub wielu nadawców. Dodatkowo węzeł dystrybucyjny jest odpowiedzialny za utrzymywanie metadanych opisujących przesyłane przez nadawców publikacje. Czasami węzeł ten pełni również rolę kolejki podczas kopiowania danych pomiędzy nadawcą o węzłem docelowym. W wielu scenariuszach implementacyjnych węzeł dystrybucyjny i nadawca są obsługiwane przez jedną instancję bazy danych.
– Subskrybent (ang. *Subscriber*) – lokalna instancja bazy danych SQL

Server na urządzeniu mobilnym przeznaczona do odbioru informacji z serwera. Subskrybent może pobierać dane z wielu subskrypcji przekazywanych przez wielu subskrybentów. Możliwe jest więc zaimplementowanie w pełni rozproszonego systemu wymiany danych w oparciu o urządzenia mobilne i wiele źródeł danych.

- Artykuł (ang. *Article*) – identyfikuje część bazy danych – obiekt, który podlega przekazywaniu poprzez sybskrypcję. Subskrybowanym obiektem może być w ogólnym scenariuszu wymiany tabela, procedura wbudowana, czy widok. Możliwe jest użycie filtrów w celu wyselekcjonowania interesujących danych z poszczególnych obiektów.
- Publikacja – zestaw artykułów – obiektów przygotowanych do wysłania. Może być użyty do przygotowania kopii danych na wiele instancji docelowych.
- Subskrypcja – pozwala na zdefiniowanie żądania otrzymywania konkretnych publikacji, czasu ich otrzymywania i sposobu akwizycji danych.

Zastosowanie urządzeń mobilnych do budowy rozproszonego systemu przetwarzania danych wymaga zastosowania sposobów replikacji danych pomiędzy węzłem klienckim a głównym serwerem. Choć w scenariuszu akwizycji danych pomiędzy publikatorem a subskrybentem istnieje kilka typów replikacji wersja bazy danych dla urządzeń mobilnych wspiera jedynie replikacje scalające. Pozwalają one na rozwiązanie następujących problemów:

- Wiele urządzeń mobilnych w tym samym czasie wysyła żądania aktualizacji danych
- Konieczność obsługi scenariusza, w którym baza danych na urządzeniu mobilnym otrzymuje kopie danych, modyfikuje je lokalnie a następnie musi zaktualizaowac dane na serwerze centralnym.
- Mogą istnieć konflikty pomiędzy transakcjami aktualizacji danych zainicjowanymi przez aplikacje mobilne. Informacja o takiej sytuacji może być zasygnalizowana przez mechanizm publikacji, a warstwa aplikacyjna może uruchomić procedury rozwiązujące konflikty.
- Warstwa aplikacyjna może być informowana w sposób asynchroniczny o zmianie danych na serwerze, co zwalnia aplikację mobilną od periodycznego sprawdzania statusu danych na serwerze centralnym.

Przedstawiony model replikacji danych oferowany przez SQL Server umożliwia jego zastosowanie do realizacji rozproszonego systemu przetwarzania danych z użyciem urządzeń mobilnych uwzględniając typowe problemy występujące w środowiskach mobilnych.

## 3.4. Nierelacyjne metody składowania danych dla potrzeb aplikacji mobilnych

Podczas projektowania warstwy składowania danych coraz częściej używane są alternatywne do relacyjnego modelu sposoby przechowywania danych. Jednym z mankamentów modelu relacyjnego jest konieczność mapowania danych, które są przechowywane w warstwie aplikacji do modelu relacyjnego. Wymaga to wiele nakładu pracy i komplikuje rozwiązanie aplikacyjne. W przypadku zastosowania podejścia NoRel można wykorzystać taki model

danych, który lepiej pasuje do opisu rozwiązywanego problemu, a tym samym do potrzeb warstwy aplikacyjnej. Pozwala to w znacznym stopniu ograniczyć konieczność mapowania danych i uprościć warstwę aplikacyjną, co jest niewątpliwą korzyścią zwłaszcza przy projektowaniu rozwiązań dla urządzeń mobilnych dysponujących ograniczonymi zasobami.

Problemem trudnym do rozwiązania przy pomocy relacyjnego modelu danych jest również szybkość przetwarzania bardzo dużych zbiorów danych. Podstawy modelu relacyjnego sięgają lat 70. ubiegłego stulecia kiedy dominowała architektura z jednym centralnym źródłem danych. Dzisiejsze systemy mobilne są systemami rozproszonymi, które wymagają dostępu do danych zagregowanych w wielu różnych węzłach podłączonych do sieci Internet. Systemy bazodanowe należące do rodziny rozwiązań NoRel są zaprojektowane do pracy w architekturze klastrowej, a przez to wspierają budowę rozproszonego przetwarzania danych[30]. Do najważniejszych modeli danych wspieranych przez rozwiązania NoRel należą:

- Klucz – Wartość: Model ten zakłada dostęp do poszczególnych danych podlegających przetwarzaniu z wykorzystaniem mapy odzwierciedlającej związki pomiędzy danymi a ich kluczami. Mapa takich związków jest realizowana w postaci zestawu jednokierunkowej funkcji skrótu. W bazach tego typu wartością jest pole BLOB (ang. *Binary Large Object*). Pozwala to na zapisywanie danych o różnym charakterze bez wnikania w ich typ i schemat bazy danych. W bazach danych klucz – wartość może więc być przechowywany każdy typ danych. Ze względu na wykorzystanie kluczy bazy te cechują się wysoką wydajnością i podlegają dużej skalowalności. Bazy te pozwalają na utrzymanie spójności danych dla operacji w obrębie jednego klucza.
- Model dokumentowy – Dane w tym modelu są przechowywane w dokumentach. Dokument to struktura drzewiasta, która charakteryzuje się hierarchicznością . Dokumenty mogą być również tworzone w postaci uroszczonych struktur klucz – wartość. Każda struktura dokumentu może składać się z map, kolekcji lub wartości skalarnych. Dokumentowe bazy danych nie wymagają ścisłej kontroli schematu. Dwa dokumenty mogą się różnić pomiędzy sobą schematem (np. Posiadać różne nazwy węzłów XML) i mogą być zapisane w tej samej kolekcji. Jedna instancja dokumentowej bazy danych posiada wiele baz danych, natomiast jedna baza danych posiada wiele kolekcji dokumentów. Dokumentowe bazy danych pozwalają również na osadzanie jednego dokumentu w drugim. Dzięki słabej kontroli schematu możliwe jest dokładanie atrybutów jednego dokumentu bez konieczności modyfikowania innych dokumentów w kolekcji, co polepsza parametry wydajności. Dzięki zastosowaniu klastrów dokumentowe bazy danych cechują się lepszą dostępnością. Te same dane znajdują się na kilku serwerach, a aplikacja kliencka zainsta-

[30] Podając za: C. Rodrigues, J. Afonso, P. Tomé; *Mobile Application Webservice Performance Analysis: Restful Services with JSON and XML,* [in:] Communications in Computer and Information Science, Vol. 220, 2011, pp. 162-169.

lowana na urządzeniu mobilnym może otrzymać wyniki zapytań nawet w przypadku awarii serwera głównego.

– Rodzina kolumn – W tych bazach danych dane zamiast w wierszach jak to ma miejsce w modelu relacyjnym, są przechowywane w kolumnach. Jest to rozwiązanie przeznaczone dla implementacji dużych baz danych i może być użyte do projektowania hurtowni danych. Podstawowa jednostka przechowywania danych to kolumna, która składa się z pary klucz – wartość i dodatkowo posiada stempel czasowy wykorzystywany do sprawdzania ważności danych. Wiersz to kolekcja kolumn przypisanych do klucza, natomiast rodzina kolumn jest złożona z podobnych wierszy. Wykorzystanie reprezentacji kolumnowej pozwala na przetrzymywanie blisko siebie spokrewnionych danych. Bazy danych kolumnowe charakteryzują się wysoką dostępnością, która jest uzyskiwana dzięki replikacji danych na wiele węzłów klastra. Niektóre bazy danych z grupy rodziny kolumn (np. baza danych Cassandra) nie posiadają serwera głównego, a dane są replikowane na węzły klastra. W takich bazach danych skalowalność odbywa się poprzez dodanie następnych węzłów do klastra. Z tego powodu bazy te cechują się znakomitymi właściwościami skalowalności.
– Model grafowy – Podstawą działania tych baz danych jest teoria grafów. Grafowe bazy danych agregują dane w węzłach, które zawierają atrybuty. Węzły są połączone pomiędzy sobą krawędziami, które reprezentują związki pomiędzy danymi. W praktycznych rozwiązaniach spotyka się rozszerzenia tego modelu (bazy danych hipergrafowe). Bazy danych oparte na grafach są doskonałym narzędziem, które przeznaczone jest do przechowywania danych z dużą liczbą skomplikowanych powiązań. W grafowych bazach danych wyszukiwanie poszczególnych krawędzi przy pomocy związków (trawersowanie po grafie) jest operacją znacznie mniej kosztowną niż w przypadku realizacji złączeń przy pomocy kluczy głównych i obcych w relacyjnych bazach danych. Ponadto bazy grafowe pozwalają na podobny zakres drążenia danych (ang. *data drilling*) jak bazy oparte o model relacyjny. Bazy danych oparte o graf są w większym stopniu przeznaczone do pracy na jednym serwerze, lub do pracy w architekturze z serwerem centralnym.

Niewątpliwą zaletą baz danych opartych o model relacyjny jest istnienie ujednoliconego dostępu do danych w postaci języka SQL. Systemy baz danych reprezentujące nurt NoRel ze względu na swoją różnorodność obsługują bogatszy zestaw języków dostępu do danych, oraz manipulowania danymi. W większości bazy danych z nurtu NoRel są pozbawione schematu. Ma to wiele zalet w związku z możliwością składowania danych w niestandardowych postaciach. Brak schematu po stronie bazy danych utrudnia jednak sprawdzanie spójności danych i wymusza zaimplementowanie mechanizmów walidacji tej spójności po stronie aplikacji klienckiej. Aplikacja jest w tym modelu odpowiedzialna za kontrolę poprawności danych a projektanci aplikacji muszą wprowadzić schemat dla potrzeb kontroli poprawności

wprowadzanych danych. Jeśli z danych umieszczonych w bazie ma korzystać wiele aplikacji może to skomplikować ich współpracę z bazą danych[31].

Różnorodność języków dostępu do danych dla rozwiązań nierelacyjnych może być utrudnieniem przy integracji poszczególnych elementów rozproszonego systemu, lub podczas konieczności podmiany warstwy składowania danych[32].

Rozwiązanie problemu z braku jednolitego sposobu dostępu do danych poprzez język zapytań jest wprowadzenie takich sposobów komunikacji z bazą danych, które bazują na wykorzystaniu ujednoliconych interfejsów aplikacyjnych. Enkapsulacja integracji z bazą za pomocą warstwy pośredniczącej jest również sposobem na prowadzenie swoistego schematu pozwalającego na kontrolowanie danych wprowadzanych przez warstwę aplikacji. Wiele z praktycznych implementacji baz danych zgodnych z podejściem NoRel pozwala na obsługę komunikacji z bazą poprzez wykorzystanie sieciowej komunikacji wykorzystującej protokół http. Do najpopularniejszych interfejsów dostępu do nierelacyjnych baz danych należą:

– Interfejsy realizowane na bazie usług WebServices
– Interfejsy zgodne z architekturą full REST.

Usługi Web Services są jednym ze sposobów komunikacji międzyprocesowej, co oznacza że umożliwiają różnym aplikacjom wymianę danych, oraz przetwarzanie danych na rzecz innych aplikacji. Ta metoda komunikacji pomiędzy aplikacjami jest niezależna od zastosowanej technologii, systemu operacyjnego czy architektury sprzętowej. Niezależność tę osiągnięto poprzez zastosowanie ujednoliconych standardów opartych na składni XML. Do wywoływania usług Web Services stosuje się zestandaryzowany zestaw protokołów, takich jak SOAP (ang. *Simple Object Access Protocol*) oraz UDDI (ang. *Universal Description Discovery and Integration*). Nie są one własnością żadnych konsorcjów i nie objęte są żadnymi prawami patentowymi. Dzięki temu jest możliwe ich zastosowanie na szeroką skalę w rozmaitych rozwiązaniach. Nad utrzymaniem tych standardów czuwa światowa organizacja World Wide Web Consortium (W3C). Mimo że usługi Web Services działają niezależnie od siebie, możliwe jest ich wzajemne łączenie celem rozwiązania bardziej złożonych problemów.

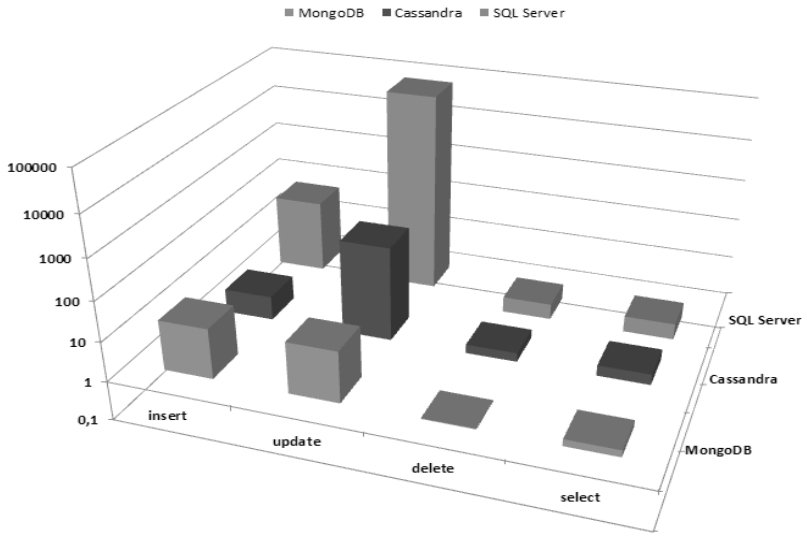
Model REST jako wzorzec architektury oprogramowania wywiedziony z doświadczeń przy pisaniu specyfikacji protokołu http, wprowadza dobre praktyki tworzenia architektury aplikacji rozproszonych. Charakterystycznym elementem (opisywanego w poprzednim rozdziale) wzorca architektury REST jest interfejs usług webowych, w którym parametry wywołania danej usługi są umieszczane w ścieżce adresu URL, a nie w części przeznaczonej na parametry, jak w klasycznych wywołaniach GET lub POST.

W niektórych wypadkach możliwa jest rezygnacja z bardzo restrykcyjnych mechanizmów utrzymania spójności danych obecnych w systemach zarządzania relacyjną bazą danych i wybranie takiego modelu, który gwarantu-

[31] Szerzej na ten temat w: J.Mena, *Data Mining Mobile Device*, CRC Press, New York 2013.
[32] Zgodnie z: P.J. Sadalage, M. Fowler; *NoSQL Kompedium Wiedzy*, Helion, Gliwice 2014.

je lepszą efektywność czasową. Badania czasów wykonania podstawowych operacji na danych w przypadku zastosowania różnych modeli i systemów baz danych zostały przedstawione na rysunku 13. Z przedstawionych badań wynika, iż analizowane bazy danych nurtu NoRel, które dopuszczają możliwość zastosowania mniej restrykcyjnych mechanizmów zachowania spójności danych, wykazują lepszą efektywność czasową podstawowych operacji na danych. W wielu systemach przechowywania danych dla potrzeb aplikacji mobilnych zachowanie jedynie podstawowych metod zachowania spójności danych jest wystarczające, a tym samym poprzez wykorzystanie jednej z baz danych NoRel możliwe jest polepszenie czasów odpowiedzi całego systemu przetwarzania danych wykorzystującego urządzenia mobilne.

Rysunek 13. Czasy wykonania podstawowych operacji na wybranych bazach danych.




Źródło: Ian Robinson, Jim Webber, Emil Eifrem: Graph Databases, O'Reily 2013.

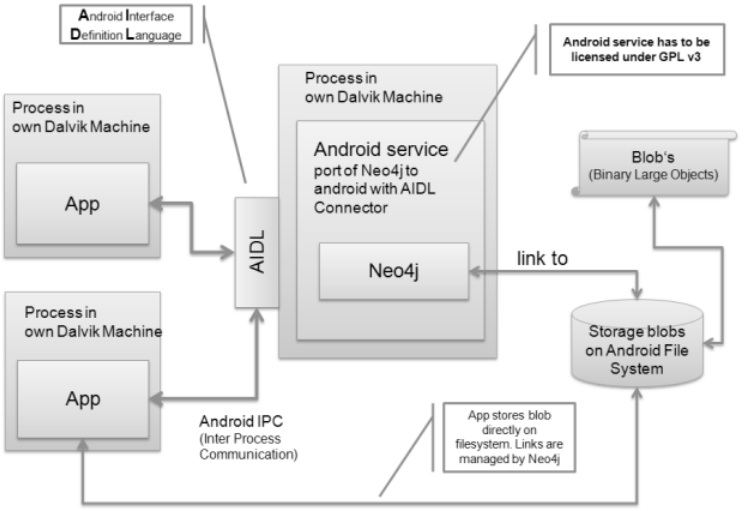
Obecne rozwiązania przechowywania danych dla aplikacji mobilnych charakteryzują się heterogenicznością. Nawet w obrębie jednej aplikacji można spotkać różne sposoby i metody składowania danych od danych niestrukturalnych, poprzez dane zgodne z modelem obiektowym aż po dane przechowywane za pomocą rozwiązań NoRel. Szeroki wybór różnych narzędzi i modeli przechowywania danych pozwala na elastyczny dobór poziomu niezawodności, transakcyjności, bezpieczeństwa i efektywności czasowej metod odpowiedzialnych za przechowywanie danych w systemach komputerowych wykorzystujących urządzenia mobilne.

## 3.5. Neo4j jako przykład bazy danych NoRel dla urządzeń mobilnych

Przykładem bazy danych NoRel przeznaczonej do uruchomienia na urzą-

dzeniach mobilnych działających pod kontrolą systemu operacyjnego Android jest *Neo4j Mobile*. Narzędzie jest dystrybuowane w formie projektu na platformie GitHub. Projekt jest w fazie rozwojowej, jednak jest w pełni funkcjonalny i może być zastosowany jako lokalne narzędzie składowania danych w modelu zgodnym z reprezentacją grafową.

Rysunek 14. Poglądowy schemat współpracy bazy danych Neo4j Mobile i aplikacji w systemie operacyjnym Android.




Źródło: K. Harężlak, Data replication on mobile devices, w: Studia Informatica 2009 Nr 2A(83), Wydawnictwo Politechniki Śląskiej, Gliwice 2009.

Baza danych Neo4j Mobile wykorzystuje infrastrukturę usług systemu operacyjnego Android do współdziałania z warstwą aplikacji. Dzięki temu usługa składowania danych może być użyta przez wiele aplikacji działających w systemie operacyjnym, które mogą posiadać dostęp do tych samych danych. Uruchomienie bazy danych Neo4j Mobile jako usługi pozwala na działanie warstwy składowania danych w tle. Funkcjonalność związana ze składowaniem danych jest dostępna z wykorzystaniem interfejsu AIPC (ang. *Android Inter Process Communication*) przy pomocy języka AIDL (ang. *Android Interface Definition Language*). Poglądowy schemat przedstawiający współdziałanie komponentów bazy danych Neo4j Mobile i aplikacji mobilnych przedstawia rysunek 14.

Baza danych Neo4j Mobile bazuje na wersji narzędzia przeznaczonej do uruchomienia na serwerze i również posiada modułową architekturę. Elementy charakterystyczne dla infrastruktury serwerowej, takie jak zaawansowane mechanizmy komunikacji pomiędzy węzłami w architekturze rozproszonej i moduły pozwalające na konfigurację narzędzia z wiersza poleceń zostały jednak usunięte z wersji przeznaczonej na urządzenia mobilne. Najważniejsze komponenty, które tworzą bazę danych Neo4j dla urządzeń mobilnych z systemem operacyjnym Android wraz z ich opisem przedstawiono w tabeli 1.

Tabela 1. Komponenty bazy danych Neo4j Mobile

| Komponent | Przeznaczenie |
|---|---|
| Neo4j-kernet | Jądro bazy danych, które jest odpowiedzialne za przechowywanie danych zgodnie z modelem grafowym. Pozwala na realizację w sposób transakcyjny operacji manipulujących danymi. Wprowadza również interfejs wejścia – wyjścia do komunikacji pomiędzy modułami architektury rozproszonej. |
| Neo4j-graph-algo | Moduł odpowiedzialny za realizację tworzenia struktur opartych o reprezentację grafową w bazie danych. Jest odpowiedzialny za efektywne wstawianie danych i ich modyfikację. |
| Neo4j-graph-marching | Moduł odpowiedzialny za wyszukiwanie w grafie węzłów zgodnych z kryteriami zadanymi podczas formułowania zapytań do bazy danych. Pozwala na efektywne przeszukiwanie grafu (trawersowanie) w celu zwrócenia odpowiednich danych. |
| Neo4j-lucene-indexing | Moduł ten wprowadza możliwości indeksowania węzłów. Możliwość indeksowania węzłów nie jest naturalną funkcjonalnością rozwiązań bazodanowych opartych o reprezentacje danych w postaci grafu. Dodanie takiej możliwości zwiększa jednak efektywność czasową pobierania danych z grafu. Przed rozpoczęciem przeszukiwania (trawersowania) musi być wybrany węzeł startowy, który jest wyszukiwany na podstawie zadanych kryteriów. Wyszukiwanie węzła startowego przy pomocy indeksów jest o wiele szybsze. Z tego względu baza danych daje możliwość zaindeksowania węzłów według wartości atrybutów, które są najczęściej wykorzystywane do określenia kryteriów wyszukiwania węzła startowego w grafie. |

Niezależnie od wersji bazy danych Neo4j Mobile istnieje możliwość użycia bibliotek Neo4j jako komponentów wbudowanych w aplikację. Dzięki temu można implementować warstwę przetwarzania danych zgodnie z modelem grafowym bez instalowania dodatkowych komponentów w systemie operacyjnym. Takie podejście będzie na ogół wykorzystywane w przypadkach, gdy przechowywane w postaci grafu dane będą dostępne tylko dla jednej aplikacji, lub aplikacja samodzielnie implementuje metody dostarczania treści innym komponentom obecnym w systemie Android.

# 4

# NOWOCZESNE INTERFEJSY CZŁOWIEK-KOMPUTER W STEROWANIU URZĄDZEŃ MOBILNYCH

## 4.1. Wprowadzenie

Ostatni czas (rok 2015) to okres wzmożonego zainteresowania interfejsami zewnętrznymi, współpracującymi z urządzeniami mobilnymi, bądź o nie opartymi. Staje się tak szczególnie w przypadku urządzeń kreowania rzeczywistości wirtualnej i/lub rozszerzonej (odpowiednio *virtual reality* oraz *augmented reality*), a także wspomagających sterowanie obiektami w nich zrealizowanych. Środowiska owe posiadają charakter „immersyjny", tzn. następuje niejako zanurzenie użytkownika w kreowanym środowisku. Jak pisze K. Prajsner, „Z uwagi na nasze pożądanie doświadczenia immersji, koncentrujemy się na pojawiającym się świecie i używamy inteligencji raczej po to, by wspomagać niż kwestionować realność tego doświadczenia"[33].

### 4.1.1. Rzeczywistość rozszerzona

Rzeczywistość rozszerzoną (ang. *augmented reality*, AR) można określić jako zestawienie rzeczywistego otoczenia użytkownika z elementami generowanymi przez komputer w jedno środowisko, zsyntetyzowane z punktu widzenia użytkownika w jedną całość. Wiele opracowań zgodnych jest co do faktu, że technika AR pozwala na osiągnięcie znaczących korzyści w efektywności, ale też intuicyjności wykonywanych przez człowieka działań, płynących z ich stosowania. Dotyczy to zwłaszcza zadań złożonych, a więc projektowania i konstruowania.

### 4.1.2. Rzeczywistość wirtualna

Pojęcie rzeczywistości wirtualnej (ang. *virtual reality*, VR) według S. Byrsona i J. Lanier'a należy rozumieć jako efekt interaktywny uzyskany za pomocą

[33] K. Prajzner, *Tekst jako świat i gra. Modele narracyjności w kulturze współczesnej*, Łódź 2009, s. 27.

technologii komputerowej poprzez wykreowanie interaktywnego, trójwymiarowego modelu, w którym obiekty dają wrażenie przestrzennej fizycznej obecności[34]. Prekursorem rzeczywistości wirtualnej jest Myron W. Krueger, który jest twórcą wielu instalacji będących prototypami systemów rzeczywistości wirtualnej mających zastosowanie w edukacji i psychoterapii.

Rozwój dzisiejszych aplikacji do modelowania grafiki 3D, nowoczesnych technik komputerowych pozwalających na implementowanie dokładnych rozwiązań wizualizacyjnych oraz obecność na rynku wyrafinowanych urządzeń pozwalających na interakcję użytkownika z komputerem[35] sprawiły, iż stało się możliwe zaimplementowanie środowisk symulacyjnych i wizualizacyjnych do obrazowania przestrzennego. Intencją takich środowisk jest lepsze odwzorowanie projektowanych modeli w celu dokładniejszego zobrazowania ich właściwości i szczegółów.

### 4.1.3. Rosnąca popularność VR i AR

W raporcie Business Insider z maja 2015, dotyczącym tzw. „gogli wirtualnej rzeczywistości" (ang. *VR headset*) zauważono bardzo istotny szczegół – że o ile technologie VR i AR kojarzone były głównie z rynkiem gier i rozrywki multimedialnej, zastosowania ich nawet w przypadku typowego użytkownika końcowego niosą obecnie dużo większe możliwości, chociażby w pracy z wideokonferencją, wykorzystaniem nowych usług z zakresu e-commerce czy też sterowaniem urządzeniami domowymi.

Przytaczając kilka kluczowych punktów ze wspomnianego raportu, obecny czas, właśnie ze względu na dużo większe możliwości oraz rozpowszechnienie urządzeń mobilnych, uważa się za nowe otwarcie rynku wirtualnej i rozszerzonej rzeczywistości, a re-debiut na rynku konsumenckim może przynieść rozwój zaawansowanej i bardzo immersywnej platformy dla szerokiej gamy zawartości. Business Insider ocenia, że sam amerykański rynek sprzętu VR osiągnie do 2020 roku niecałe trzy miliardy dolarów wartości (obecnie około 40 milionów dolarów). Co najważniejsze, większość czujników/sensorów/gogli/urządzeń VR są i będą stosunkowo sprzętem elektronicznym stosunkowo nisko-kosztowej kategorii, i podobnie jak *smartwatch* będą parowane z urządzeniami użytkownika typu smartfon czy inne urządzenie mobilne. Z tego powodu rynek VR będzie napędzany zarówno przez rozwój technik i sprzętu VR/AR, jak również możliwościami urządzeń mobilnych oraz sieci Internetu Rzeczy, w których prawdopodobnie będą funkcjonować[36].

VR/AR ma szansę stać się ważną platformą do przekazywania nowej zawartości multimedialnej czy nawet marketingu. Eksperymenty firmy Oculus pokazały, że VR może służyć do poprawy doświadczeń klientów online, a na-

[34] G. Burdea, P. Coiffet, V*irtual Reality Technology*, Published by John Wiley & Sons, New Jersey 2003.
[35] Temat rozwinięto w: P. Buchwald, *Nowoczesne interfejsy HMI*, [w:] P. Kostka (red.), *Wybrane zagadnienia bioinformatyki*, WSB Dąbrowa Górnicza 2012.
[36] *Kontekst Internetu Rzeczy i roli urządzeń mobilnych* poruszono w: P. Buchwald, M. Rostański M, *Globalizacja a rozwój Internetu. Internet wszystkich rzeczy (Internet of Things) i nowoczesne sposoby komunikacji maszyn*, [w:] W. Kojs, E. Rostańska, K. Wójcik (red.), *Edukacja i gospodarka w kontekście procesów globalizacji*, Oficyna Wydawnicza „Impuls", Kraków 2014, s. 285-298.

wet skłonić ich bardziej do zakupu rzeczy, którą „obejrzeli" w wirtualnej rzeczywistości.

Wraz z wprowadzaniem innowacyjnych sposobów wyświetlania obrazu użytkownikowi systemu, koniecznym staje się umożliwienie intuicyjnego sterowania. W sytuacji, gdy użytkownik siedzi za kierownicą pojazdu, gdy przed odbiorcą systemu wyświetlana jest rzeczywistość wirtualna, lub gdy człowiek porusza się po rozległym terenie, korzystając z technik AR, sprzężonych z goglami, trudno spodziewać się, że komunikacja z systemem nadal będzie przebiegać przy użyciu klawiatury.

Wycofywany właściwie już z obiegu Glass, wynalazek firmy Google, wykorzystywał delikatne ruchy głową użytkownika, a także przesuwanie palcem po ramce okularów. Innowacyjne pomysły sięgają coraz dalej – do obsługi rzeczywistości wirtualnej można posłużyć się chociażby specjalnymi rękawicami, jak IGS Glove, czy też opaskami, jak Myo Armband. Ogromna rzesza rozwiązań natomiast dotyczy wykorzystania komend głosowych w pracy z urządzeniami mobilnymi.

## 4.2. Typologia nowoczesnych interfejsów dla urządzeń mobilnych

Starając się dokonać jakiejkolwiek klasyfikacji, trzeba zdawać sobie sprawę z ogromnego potencjału innowacyjności rynku urządzeń mobilnych. Ogólnie jednak interfejsy obecnie wykorzystywane lub rozwijane można spróbować podzielić na cztery podstawowe grupy na podstawie danych wejściowych, jakie są przetwarzane przez urządzenie:

– **kategoria informacji wideo** – urządzenia funkcjonują na podstawie rozpoznawania obrazu, jego interpretacji i realizacji stosownych działań zgodnie z zaprogramowanym postępowaniem.

Mowa tutaj o interfejsach wyjściowych (przedstawiających dane użytkownikowi), od prostych technik wyświetlania obrazu na różnych powierzchniach, do skomplikowanych układów wirtualnej i rozszerzonej rzeczywistości, realizujących różne techniki 3D i holowizji. Dynamiczny rozwój urządzeń typu smartphone, które posiadają dobrej jakości wyświetlacze o dużych rozdzielczościach, pozwolił na ich wykorzystanie w zakresie wizualizacji 3D. W przypadku wynalazków typu Dourovis Dive czy wręcz Google Cardboard, stosunkowo prosty pomysł polegający na umieszczeniu urządzenia w specjalnych goglach umożliwia uzyskanie zadziwiająco dobrego efektu.

– **kategoria** głosowa – urządzenia funkcjonujące na podstawie rozpoznawania głosu (mowy)

Komunikacja werbalna jako najbardziej naturalny sposób świadomej komunikacji między ludźmi bardzo mocno nasuwa wykorzystanie tego sposobu do interakcji człowiek-maszyna. Interfejs głosowy odniósł sukcesy w telefonach mobilnych – S-Voice, Siri, czy Cortana są świetnymi przykładami. Ze względu na swój charakter, stosowanie interfejsu głosowego jest najbardziej wydajne podczas wykonywania zadań, w których ilość informacji na wejściu i wyjściu jest niewielka[37].

[37] R. Hinman, *The Mobile Frontier: A Guide for Designing Mobile Experience*s, 2012.

– **kategoria czujników ruchu**, gdzie typowym jest wykorzystanie danych IMU (ang. *Inertial Measurement Unit*) – urządzenia funkcjonują na podstawie czujników inercyjnych.

Zestaw danych IMU jest znany zwłaszcza z dziedziny sterowania samolotami bezzałogowymi, związanych z zapisem ruchu. Mierzona jest prędkość, orientacja, przyspieszenie, przy użyciu kombinacji akcelerometrów, żyroskopów, a czasem magnetometrów[38]. Bardzo często te interfejsy są używane, czy też łączone z innymi – jak chociażby „okulary", w których trzeba śledzić położenie i ruch głowy.

– **kategoria czujników fizycznych** – kategoria oznacza urządzenia funkcjonujące na podstawie czujników śledzących fizyczne dane o użytkowniku.

Zaawansowanym przykładem czujnika tej kategorii jest EMG. Elektromiograf jest w stanie, wykrywając potencjał elektryczny generowany przez mięśnie zaktywowane neurologicznie, analizować biomechanikę ruchu człowieka. Innym przykładem mogłoby być mierzenie tętna użytkownika bądź temperatury (choć oczywiście ciężko użytkownikowi sterować owymi danymi naumyślnie, nadal odpowiedni stan może wywołać określoną reakcję systemu). W tej kategorii rozwija się również nie najnowszy, ale bardzo obecnie unowocześniany pomysł wykorzystania EEG (elektroencefalografii) – czyli pomiarów zmiany potencjału elektrycznego na powierzchni głowy, pochodzące od aktywności neuronów kory mózgowej, które po odpowiednim ich wzmocnieniu tworzą z nich zapis – elektroencefalogram.

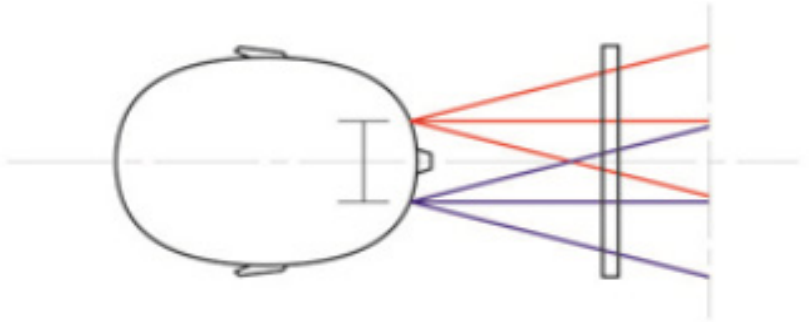
## 4.3. Przykłady realizacji poszczególnych typów interfejsów sterowania urządzeniami mobilnymi

### 4.3.1. Urządzenia oparte o informacje wideo

Oczywiście, podstawową grupą urządzeń funkcjonujących w oparciu o informacje wideo są wyświetlacze. Znacząca większość interfejsów wizyjnych wspiera już dzisiaj wyświetlanie trójwymiarowe. Zasada działania popularnych urządzeń do wizualizacji 3D opiera się na właściwościach stereograficznego widzenia. Poprzez akwizycję różnych obrazów do prawego i lewego oka, uzyskuje się złudzenie przestrzennej obecności wizualizowanych obiektów 3D. Obrazy dla lewego i prawego oka są generowane za pomocą kamer zwróconych na ten sam przedmiot i przesuniętych w przestrzeni trójwymiarowej względem siebie o odległość równą odległości pomiędzy jednym a drugim okiem odbiorcy. Tak wygenerowane obrazy są przesyłane do oddzielnych wyświetlaczy, lub jednego wspólnego wyświetlacza (sposób wyświetlania obrazów 3D *side by side*). Jeśli każdy z obrazów jest widziany tylko przez jedno oko, to użytkownik ma wrażenie pełnego trójwymiarowego efektu. Poglądowy sposób działania urządzeń wizualizacyjnych 3D w oparciu o taką zasadę przedstawia rysunek 15.

[38] Inertial Navigation: *40 Years of Evolution* – http://www.imar-navigation.de

Rysunek 15. Schemat działania urządzenia do wizualizacji trójwymiarowych



Źródło: Dokumentacja techniczna urządzenia Oculus Rift

W przypadku nowoczesnych interfejsów wyświetlania obrazu część ekspertów wyciąga daleko posunięte wnioski – cytując ComputerWorld, „wszystko wskazuje na to, że po wielu dekadach «komputeryzacji» niemal każdej dziedziny ludzkiego życia, w końcu nastąpi prawdziwy przełom – całe nasze otoczenie przeistoczy się w jeden wielki interfejs. Ściany, podłogi, sufity, blaty stołów, okna i pozostałe powierzchnie w naszych domach i biurach zamienią się w interaktywny wyświetlacz”[39]. Wizja ta jest bardzo pociągająca, jednak należy wziąć pod uwagę krzywą oczekiwań, na której nowoczesny produkt czy technologia zawsze znajdują się w pewnym momencie na tzw. „hype peak”, czyli krótko mówiąc, są przeceniane i za dużo się od nich oczekuje, zdaniem autorów tak też jest w tym przypadku.

Niemniej jednak, najprostszą koncepcją jest tutaj wyświetlanie obrazu bezpośrednio na dowolnych powierzchniach – przykładem może być tutaj projekt Lumo, który zakłada wyświetlanie animowanych scen na płaskiej powierzchni, a także umożliwienie reakcji na „dotyk” dłoni lub stóp z wyświetlanym obrazem. Jest to przykład bardzo podstawowego wdrożenia pomysłu, który ma szansę (zdaniem redaktorów ComputerWorld) wkrótce stać się dość powszechny: wykorzystania istniejących powierzchni (ścian, sufitów, podłóg, stołów, okien itd.) jako interaktywnych ekranów komputerowych.

Innym przykładem, zakładającym użycie bezpośrednio przystosowanego urządzenia, może być bardziej zaawansowany „konkurent” wspomnianego wcześniej Dourovis Dive, w odróżnieniu od tamtego projektu funkcjonujący w rzeczywistości rozszerzonej, nie wirtualnej. Mowa o Microsoft HoloLens.

[39] Za: www.computerworld.pl/news/401726/Interfejsy.ktore.wywolaja.rewolucje.html

Rysunek 16. Microsoft HoloLens w akcji.

Źródło: materiały reklamowe firmy Microsoft.

Microsoft HoloLens jest w istocie specyficznym urządzeniem, ponieważ samo w sobie jest autonomicznym urządzeniem mobilnym (nie musi być połączone z komputerem lub smartfonem) wyposażonym w zestaw sensorów, wyświetlacze oraz specjalny procesor do przetwarzania obrazów w rozszerzonej rzeczywistości. Ważną różnicą HoloLens w porównaniu do typowych aplikacji AR jest wykorzystanie renderowanych obiektów w specyficznych lokalizacjach rzeczywistych – kilku użytkowników HoloLens jest w stanie „widzieć" ten sam obiekt w tej samej lokalizacji. Konstrukcja aplikacji dla HoloLens jest przy tym identyczna jak budowa uniwersalnej aplikacji dla Windows, z wykorzystaniem pośredniczącego oprogramowania dla wizualizacji i komunikacji ze środowiskiem mobilnym, jak Unity. Daje to duże możliwości kolaboracji w wielu polach, włącznie z medycyną, inżynierią czy rozrywką.

Z punktu widzenia zagadnienia sterowania przez użytkownika, kamery wideo mogą spełniać rolę informatora o ruchach, gestach i pozycji osoby obserwowanej przez kamerę, jak w przypadku LeapMotion czy Microsoft Kinect.

Rysunek 17. Działanie układu wykrywającego gesty w praktyce.

Źródło: Materiały reklamowe LeapMotion.

Wykorzystanie kontrolera ruchu daje bardzo duże możliwości sterowania aplikacją poprzez śledzenie ruchu całej sylwetki użytkownika, gestów ręki czy mimiki twarzy. Technologia ta posiada obecnie szerokie zastosowanie przede wszystkim w grach i aplikacjach reklamowych (zwłaszcza na wszelkiego rodzaju wydarzeniach, gdyż doskonale przyciąga uwagę obecnych tam osób), ale coraz szersza literatura naukowa wskazuje kolejne kierunki wykorzystania kontrolera ruchu, tym bardziej że SDK firmy Microsoft pozwala na dość zaawansowane wykorzystanie wszystkich funkcji Kinect i tym samym realizację aplikacji, które rozpoznają całe ciało użytkownika i odzwierciedlają ruch poszczególnych kończyn.

Układ rozpoznający gesty za pomocą wizji pozwala na tworzenie aplikacji dających użytkownikowi możliwość wchodzenia w bardzo szczegółowe interakcje z elementami wirtualnymi bez konieczności korzystania z jakiegokolwiek kontrolera (klawiatury, myszy, ekranu dotykowego etc.). Interfejs wykrywający gesty jest jednak trudny w realizacji i we wdrożeniu do codziennego działania użytkownika. Przytaczając fragment (dość krytycznej) recenzji LeapMotion: „Leap Motion dziedziczy wszystkie wady interfejsu dotykowego i ani jednej jego zalety. Interfejs dotykowy jest bowiem intuicyjny, ale przy korzystaniu z hybrydy jak z laptopa wymusza on odrywanie ręki od biurka. Leap Motion również wymusza ów ruch, ale za to nie oferuje owej intuicyjności”[40].

### 4.3.2. Urządzenia oparte o informację audio

Jak trafnie pisze Mielczarek[41], urządzenia pracujące z interfejsem głosowym (VUI, ang. *Voice User Interface*), gdzie do interakcji człowiek-maszyna wykorzystuje się komunikację werbalną, stanowiły źródło wielu bardzo po-

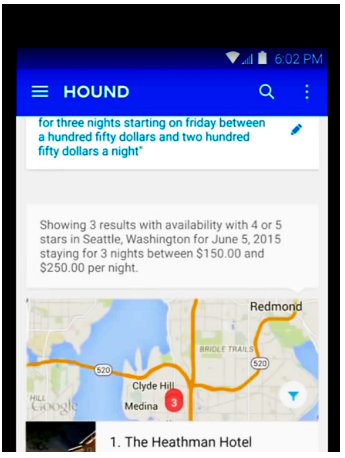
[40] M. Gajewski, *Leap Motion – miał być Kinect-killerem, a jest rynkową klęską*, http://www.spidersweb.pl/2014/03/leap-motion.html

[41] J. Mielczarek, *Głosowe interfejsy użytkownika*, blog webusability.pl, 2012.

chopnych dywagacji co do rozwoju komputerów pod koniec wieku dwudziestego. Szacowano wówczas, że największe nadzieje pokłada się właśnie w rozwój interfejsów głosowych. Wielkiego przełomu w sterowaniu komputerem jednak nie było, a interfejs głosowy stanowi źródło wielu problemów i nie jest wolny od ograniczeń technologicznych. Mowa tu o rozpoznawalności mowy, ale także stosunkowo niewielkiej ilości informacji, zarówno na wyjściu, jak i wejściu, którą można obsłużyć za pośrednictwem systemów głosowych, co uwidacznia się chociażby podczas komunikacji z automatycznym telefonicznym biurem obsługi.

Powiew świeżości na płaszczyźnie interfejsów głosowych przyniósł dopiero rozwój algorytmów rozpoznawania mowy, zarówno pod względem technicznym, jak i syntaktycznym, a nawet, czy może zwłaszcza, semantycznym. Rozwijane przez głównych producentów oprogramowania na smartfony oprogramowanie „asystenta", czyli Siri czy Cortana, jest kwintesencją koncepcji – wraz z dodaną „osobowością" oprogramowania, użytkownik ma wrażenie, że „rozmawia" z własnym urządzeniem, mogąc poprosić o wykonanie prostej usługi w sieci, znalezienie informacji czy zapisanie notatki.

Rysunek 18. Interfejs programu Hound, przykładowego „asystenta" na smartfon z systemem Android




Źródło: www.youtube.com

Bardziej wyspecjalizowanymi przykładami mogą być tutaj Dragon Dictation, aplikacja mobilna, której można podyktować cały tekst celem zapisu bezpośrednio do dokumentu tekstowego, czy też aplikacje SoundHound lub Shazam, które służą do rozpoznawania muzyki, słyszanej właśnie przez mikrofon telefonu komórkowego. Problemem interfejsu głosowego i wysoką barierą technologiczną dla produktu jest konieczność dostosowywania, a właściwie konstrukcji od nowa, systemu dialogowego dla każdego języka, dla którego odbiorców kierowany jest produkt. Systemy dialogowe, komunikujące się z użytkownikiem za pomocą mowy składają się bowiem z 3 podstawowych

modułów: rozpoznawania, syntezy mowy i nadzorcy dialogu, a dla każdego z obsługiwanych języków trzeba wypracować rozwiązania każdego z trzech modułów. Rozpoznawanie mowy polega bowiem w dużym uproszczeniu na słownikach, synteza mowy potrzebuje do działania bazy danych utworzonej z przetworzonych i posegmentowanych nagrań mowy naturalnej, nadzorca dialogu z kolei do ekstrakcji informacji z wypowiedzi użytkownika wykorzystuje zaawansowane techniki przetwarzania języka naturalnego.

Dzięki gotowym technologiom do wykorzystania (i kupienia) autor aplikacji nie musi zmagać się z realizacją systemu VUI samodzielnie. Powstające od niedawna coraz częściej konkurencyjne pakiety głosowe (np. Vac czy Tazti) pozwalają na szybkie implementacje do realizowanej aplikacji. Ciekawym (i sztandarowym w roku 2014) przykładem na polu rozrywki elektronicznej jest chociażby głosowe sterowanie statkiem w grze Elite: Dangerous.

### 4.3.3. Urządzenia oparte o czujniki ruchu

Sensory inercyjne używane w urządzeniach opartych o czujniki ruchu mają za zadanie przekazanie do jednostki sterującej informacje pozwalające na odwzorowanie położenia, przyspieszenia i trajektorii ruchu w środowisku użytkownika. Dobrym przykładem jest tutaj Inertial – rękawica pokazana na rys. 19. Inertial używa wysoce dokładnych sensorów inercyjnych do uchwycenia danych o poruszeniu ręki czy palców. Użycie rękawicy ma bardzo ważną zaletę – pozwala na śledzenie i pobieranie danych o ruchu nawet, gdy ręka jest całkowicie zasłonięta przed ewentualnymi kamerami (np. podczas naprawy urządzenia). Stanowi to dużą różnicę w stosunku do stosowanych do podobnych celów systemów mocap (ang. motion capture – śledzenia ruchu). Rękawice dostępne na rynku najczęściej w pełni współpracują z systemami modelowania 3D typu Unity, czy też przemysłowymi, np. Siemens JACK.

Rysunek 19. Wykonana z lycry rękawica typu Inertial. Charakterystyczny brak widocznych elementów sterowania

Zupełnie innym koncepcyjnie podejściem jest rezygnacja z sensorów inercyjnych i poleganie na pobraniu informacji o ruchu użytkownika za pomocą zewnętrznych urządzeń pomiarowych. Całkowicie nowatorskie próby uchwycenia informacji o gestach, ruchu i ogólnej aktywności fizycznej użytkownika skupiają się na wykorzystaniu informacji o zakłóceniach fal elektromagnetycznych czy wręcz radaru.

Project Soli firmy Google jest przykładem podejścia, w którym za pomocą fal elektromagnetycznych o wysokiej częstotliwości śledzi się gesty użytkownika. Wysyłając ciągły sygnał, który jest odbijany i zniekształcany, można przy użyciu ciągłego przetwarzania owego sygnału i zaawansowanych metod Machine Learning, wykrywać i rozpoznawać gesty. Zdaniem twórców Project Soli, wykorzystanie fal elektromagnetycznych i ich wysoka częstotliwość jest kluczową przewagą w stosunku do systemów opartych na wideo – sensory Soli potrafią wykonywać śledzenie ruchów z dokładnością o 10000 ramek na sekundę, a radar może odbierać fale również poprzez pewne typy obiektów, pozwalając na adaptację w środowiskach, w których nie da się efektywnie użyć kamer wideo.

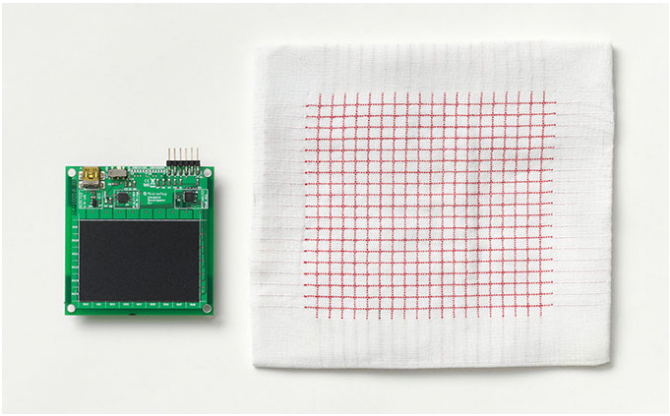
Rysunek 20. Wizualizacja działania Project Soli.


Źródło: materiały reklamowe firmy Google

### 4.3.4. Interfejsy oparte o czujniki fizyczne

Wykorzystanie czujników fizycznych może być bardzo proste w swojej naturze, jednak powiązane z urządzeniem mobilnym posiadającym wiedzę o danych z czujników daje bardzo duży potencjał kreatywności. Dość dobrze ilustruje to projekt Jacquard (https://www.google.com/atap/project-jacquard), w którym najważniejszym składnikiem jest przewodzące włókno, mogące służyć jako podstawowy składnik tkanin, ubioru czy innych materiałów tekstylnych. Wplecenie takiego włókna w materiał pozwala na projektowanie codziennych obiektów użytkowych rozpoznających gesty użytkownika, bądź dające możliwości interakcji.

Rysunek 21. Powierzchnia materiału pozwalająca na interakcję.

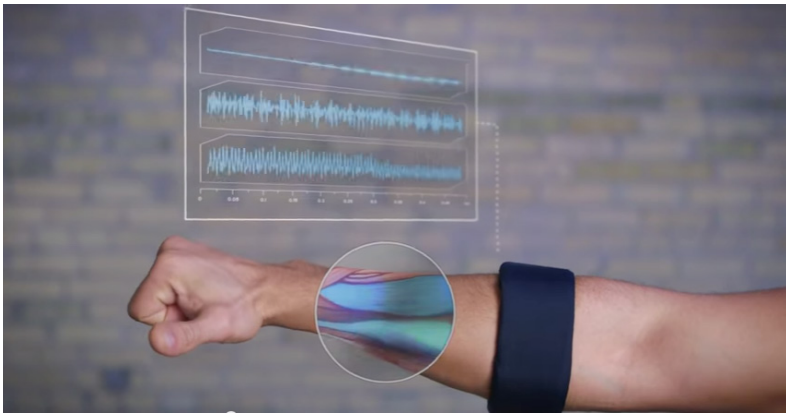


Źródło: strona projektu Jacquard

Czujniki fizyczne mogą jednak być dużo bardziej skomplikowane niż przewodzące włókna – dwa poniżej opisane przykłady posługują się skomplikowanymi sensorami wykrywającymi zmiany elektromagnetyczne – elektromiograf i encelofalograf.

W przypadku elektromiografii, chodzi o wykrycie zmian napięcia mięśni przedramienia podczas wykonywania gestów przez użytkownika. Dane EMG uzyskiwane są dzięki technice oceniania i nagrywania elektrycznej aktywności wytworzonej przez mięśnie szkieletowe[42]. Dane potrzebne do interpretacji sterowania potrafią być przetworzone przez kontroler nałożony na ramię jak zwykła opaska (jak pokazano na rys. 22).

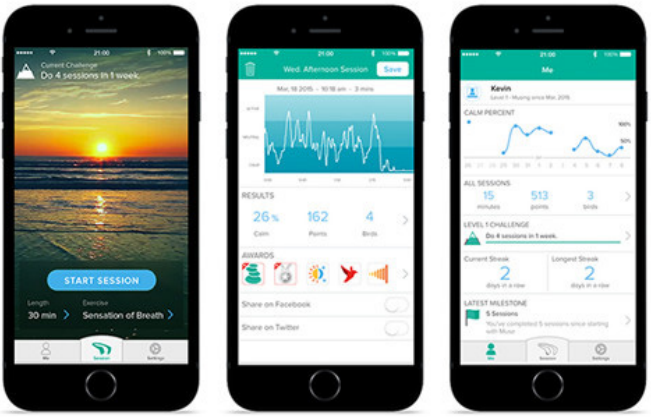
Rysunek 22. Opaska Myo – przykład wykorzystania elektromiografii do sterowania



[42] G. Kamen, *Electromyographic Kinesiology. In Robertson*, DGE et al. *Research Methods in Biomechanics. Champaign*, IL: Human Kinetics Publ., 2004.

Najlepszym przykładem wykorzystania z kolei nowoczesnego interfejsu EEG dla urządzeń mobilnych może być działanie słuchawek *Muse* oraz towarzyszącej im aplikacji Calm. Tzw. *Muse headset* to zestaw do mierzenia fal mózgowych EEG, podobny do *Emotiv*[43] czy *NeuroSky*[44]. Aplikacja Calm, oferowana przez producenta Muse, prowadzi użytkownika poprzez ćwiczenia skupienia uwagi, informując o postępach.

Rysunek 23. Ćwiczenia skupienia uwagi – aplikacja Calm firmy Muse




Źródło: www.choosemuse.com

Rysunek 24. „Słuchawki" EEG Muse


Źródło: http://www.idgconnect.com/abstract/9611/muse-a-stylish-brain-tech-challenger

[43] http://emotiv.com/
[44] http://neurosky.com/

Możliwości terapeutyczne są zresztą częstym tematem dyskusji nad wykorzystaniem nowoczesnych interfejsów – chociażby, w 2012 opublikowano artykuł w British Medical Journal[45], w którym wykazano pozytywny wpływ na remisję stanów depresyjnych przy użyciu terapii zawierającej elementy związane z grami komputerowymi, zwłaszcza wykorzystującymi urządzenia mobilne.

Oczywiście, istnieje wiele innych zastosowań tego interfejsu, włącznie z rozrywkowymi. Firma *MyndPlay* oferuje gry, w których również istotne jest sprzężenie użytkownika z którymś z monitorów fal mózgowych, wspomnianych powyżej. Najbardziej obrazowym przykładem jest tutaj *Nevermind*, gra z gatunku horrorów, w której podczas grania mierzony jest poziom stresu gracza i na tej podstawie modyfikowana jest rozgrywka.

Stosunkowo niska cena zestawów EEG daje szansę na rozpowszechnienie wśród użytkowników końcowych, jeśli tylko zostanie wykreowany odpowiedni produkt – co daje z kolei szansę odpowiedniemu pomysłowi na startup.

## 4.4. Technologiczne zasady realizacji projektu wykorzystującego nowoczesne interfejsy dla urządzeń mobilnych

W kontekście poprzedniego punktu, interfejsy sterowania dla urządzeń mobilnych spełniać mogą jeszcze bardziej istotną rolę „przedłużenia" interfejsów sterowania dla rzeczywistości wirtualnej i rozszerzonej, ale także w charakterze serca całego systemu. Urządzenia mobilne posiadają bowiem niezaprzeczalną zaletę – są odpowiednio zminiaturyzowane oraz zdolne do sterowania całym układem systemów. Doskonałym przykładem są gogle Samsung VR czy Dourovis Dive. Istotą działania takich systemów jest próba wykorzystania gogli 3D, których głównym składnikiem jest telefon komórkowy z systemem operacyjnym Android, oraz komputer PC pozwalający na przekazywanie strumienia wideo bezpośrednio do telefonu komórkowego, aby otrzymać lepsze efekty wizualizacji 3D[46].

Rysunek 25. Okulary 3D firmy Dourovis Dive


[45] The effectiveness of SPARX, a computerised self help intervention for adolescents seeking help for depression: randomised controlled non-inferiority trial, doi: http://dx.doi.org/10.1136/bmj.e2598

[46] P. Buchwald, M. Rostański, K. Mączka, *Virtual reality and mobile devices in 3d objects designing and prototyping*, [w:] R. Knosala (red.), *Innowacje w zarządzaniu i inżynierii produkcji*. Tom II, Oficyna Wydawnicza Polskiego Towarzystwa Zarządzania Produkcją, Opole 2015, s. 645-654.

Niewątpliwą zaletą urządzenia Dourovis Dive jest możliwość współpracy z oferowanymi przez producenta bibliotekami programistycznymi, które umożliwiają obsługę detekcji kierunku patrzenia przez użytkownika oraz pozwalają na wyświetlanie obrazu 3D niewielkim nakładem pracy. Producenci udostępnili SDK dla systemu operacyjnego Android i iOS, oraz gotowy komponent pozwalający na współpracę aplikacji utworzonych w środowisku Unity z urządzeniem. Biblioteki programistyczne udostępniane przez producentów dzisiejszych rozwiązań do wizualizacji 3D pozwalają na utworzenie stosunkowo małym nakładem pracy wizualizacji trójwymiarowych, które cechują się dużym stopniem realizmu i pozwalają na interakcję z użytkownikiem. Jest to wykonalne, również dzięki obsłudze formatów plików używanych w środowiskach deweloperskich przez popularne środowiska CAD, pozwalających na projektowanie i implementację aplikacji 3D przeznaczonych na urządzenia mobilne i komputery stacjonarne.

Twórcy najpopularniejszych bibliotek do tworzenia wizualizacji trójwymiarowych, takich jak Unreal lub Unity 3D dbają w swoich produktach o kompatybilność z urządzeniami i interfejsami zewnętrznymi. Kolejnym przykładem takiej realizacji może być Oculus Rift firmy Oculus. Jednym z ważnych elementów wyróżniającym urządzenie jest możliwość śledzenia ruchów głowy za pomocą wbudowanego żyroskopu. W ostatniej wersji urządzenia producenci osiągnęli dość dobre parametry wizualizacji. Zastosowanie wyświetlaczy LCD o niskiej bezwładności zaowocowało wyeliminowaniem efektów rozmycia i zmniejszeniem opóźnień. W przypadku Oculus Rift, przełamano również barierę monopolistyczną – pierwsza wersja tego urządzenia została we wrześniu 2014 r. udostępniona publicznie na licencji Creative Commons 4.0 Attribution (co pozwala na adaptowanie również komercyjne), włączając w to plany układów cyfrowych, kod firmware oraz projekty mechaniczne[47]. Oznacza to, że każdy może wytworzyć podobny produkt, zmodyfikować projekt do własnych potrzeb, a także rozwijać go komercyjnie[48].

## 4.5. Podsumowanie

W niniejszym rozdziale podjęto próbę scharakteryzowania całej płaszczyzny możliwości urządzeń mobilnych i ich interfejsów z użytkownikiem, a także przybliżono pewne kategorie rozwiązań. Celem rozdziału nie jest jednakże próba skatalogowania wszystkich możliwości – pojawiają się one bowiem na rynku w sposób ciągły i na pewno pojawiają się również w czasie czytania owej książki – część z owych projektów pozostanie w formie badawczej, część osiągnie sukces rynkowy. Istotnym celem niniejszego rozdziału jest natomiast uświadomienie ogromu możliwości wykorzystania urządzeń mobilnych do sprzęgnięcia z wieloma urządzeniami zewnętrznymi, służącymi do

[47] P. Buchwald, M. Rostański, K. Mączka, *Rzeczywistość rozszerzona oraz wirtualna we wspomaganiu prototypowania przy pomocy nowoczesnych urządzeń mobilnych*.
[48] B. Lang, *Oculus Open-sources Rift DK1: Mechanical Designs, Firmware, and More All Freely Available*, http://www.roadtovr.com/oculus-rift-dk1-open-source-manufacturing-the-dk1-nirav-patel/)

sterowania (bądź jego wizualizacji), prowadzące do uruchomienia inspiracji Czytelnika, jego pokładów kreatywności, a uogólniając – pobudzenia wyobraźni w tej dziedzinie.

# 5

# BEZPIECZEŃSTWO TECHNOLOGII MOBILNYCH

W niniejszym rozdziale poruszone zostaną tematy szeroko pojętego bezpieczeństwa technologii mobilnych, zarówno w aspekcie urządzeń jak i ich oprogramowania. Tematyka podejmowana w rozdziale dotyczyć będzie zarówno wykorzystania technologii mobilnych do zwiększania bezpieczeństwa użytkowników w sieci jak i potencjalnych zagrożeń wymierzonych w użytkowników rozwiązań mobilnych. Doświadczenia związane z szybkim rozwojem nowoczesnych technologii ukazują, iż producenci często w pierwszej kolejności koncentrują się na jak najszybszym wprowadzeniu nowego produktu na rynek, a dopiero potem weryfikują w toku realnej eksploatacji zagrożenia, które mogą być wymierzone w użytkowników ich produktów. To niekorzystne zjawisko szczególnie widoczne jest na rynku bardzo szybko rozwijających się technologii mobilnych i dotyczy zarówno urządzeń mobilnych jak i oprogramowania na nim instalowanego. Jednak rozwiązania mobilne nie tylko wiążą się z zagrożeniami, mają także szerokie zastosowanie w zwiększaniu bezpieczeństwa użytkowników.

## 5.1. Zastosowanie technologii mobilnych do zwiększania bezpieczeństwa użytkowników

Zapewnienie bezpieczeństwa danych przechowywanych w systemach informatycznych opiera się na realizacji procesów kontroli dostępu do zasobów. Zasobami, do których dostęp jest chroniony poprzez proces autoryzacji użytkownika mogą być informacje przechowywane w postaci elektronicznej (wpisy w bazie danych, pliki, fotografie itd.) lub wybrane funkcje danego systemu – na przykład: możliwość aktualizacji pól bazy danych lub przykładowo możliwość uruchomienia procesu wykonywania kopii bezpieczeństwa systemu. Proces

autoryzacji użytkownika opierać się musi na co najmniej jednej z trzech metod przekazywania atrybutu uwierzytelniania przez użytkownika do systemu. Użytkownik, aby uzyskać dostęp do zasobów systemu legitymować się może w procesie uwierzytelniania:

- wiedzą – udowadniając, że zna pewne złożenie słów/ciąg znaków przydzielony mu w celu potwierdzania uwierzytelnienia (tzw. metoda *z ang. something you know*, np. hasło, pin, odpowiedź na pytanie pomocnicze).
- fizycznym przedmiotem, którego posiadanie potwierdza, iż uwierzytelniany podmiot jest tym, za kogo się podaje (tzw. metoda *z ang. something you have*, np. certyfikat cyfrowy, token).
- cechą osobistą (fizyczną lub behawioralną) udowadniając swoją tożsamość na podstawie pomiaru biometrycznego (metoda *z ang. something you are*, do najpopularniejszych metod należy tutaj skan odcisku palca czy na przykład skan tęczówki oka).

Powyższe techniki uwierzytelniania użytkowników w systemach informatycznych stosowane są od dawna i nie są zagadnieniem nowym. Natomiast szeroki rozwój technologii mobilnych pozwolił na znaczne rozbudowanie wskazanych wyżej technik uwierzytelniania w każdym z trzech wymienionych scenariuszy.

Jako pierwszy przykład wskazać można technikę uwierzytelniania bazującą na wiedzy użytkownika. Do najpopularniejszych należy tutaj wskazanie hasła lub pinu, który potwierdzić ma tożsamość użytkownika. Metoda ta należy do najbardziej popularnych, jednak posiadanie dużej liczby haseł prowadzi do problemów z ich zapamiętywaniem. Doskonale sprawdzają się w takim przypadku urządzenia mobilne, z wykorzystaniem których użytkownik może przechowywać w bezpieczny sposób na urządzeniu swoje hasła i posiadać je zawsze przy sobie. Na rynku istnieje wiele rozwiązań umożliwiających przechowywanie dla wygody użytkownika wielu haseł do różnych kont zabezpieczonych jednym hasłem do aplikacji przechowującej hasła (wskazać tu można na przykład usługi: LastPass, mSecure, czy aWallet). Dzięki takim rozwiązaniom użytkownik może posługiwać się różnymi hasłami dla każdego z używanych portali (poczta, bank, itd.) nie obawiając się, iż duża liczba złożonych i długich haseł nie zostanie przez niego zapamiętana. Używanie różnych haseł dla każdego z kont, jakie posiada użytkownik, jest kluczowe z punktu widzenia bezpieczeństwa, ponieważ w przypadku, gdy użytkownik korzysta z jednego hasła do wielu kont, dojść może do sytuacji, w której kompromitacja jednego hasła daje atakującemu dostęp nie tylko do skompromitowanego konta, ale także do innych kont, gdzie użytkownik posiadał takie same hasło (próby uzyskania dostępu do różnych kont internetowych użytkownika na podstawie wycieku danych użytkownika to częsty scenariusz działań oszustów). Rozwiązanie polegające na wykorzystaniu mobilnego kontenera do przechowywania haseł nie obniża bezpieczeństwa procesu uwierzytelniania do kont użytkownika pod warunkiem, iż użytkownik zachowuje bezpieczeństwo podczas wprowadzania hasła do aplikacji przechowującej hasła, a sama aplikacja przechowuje powierzone hasła z wykorzystaniem metod kryptografii.

Kolejnym przykładem rozwinięcia klasycznych metod uwierzytelniania użytkownika z użyciem mobilnych technologii jest uwierzytelnianie oparte na posiadaniu fizycznego unikalnego przedmiotu, który potwierdza fakt, iż uwierzytelniany podmiot jest tym za kogo się podaje. Dzięki coraz większej powszechności mobilnych urządzeń, dostawcy usług takich, jak na przykład bankowość elektroniczna (gdzie autentyczność klienta usługi jest kluczowa), zyskali potężne narzędzie pozwalające na zwiększenie bezpieczeństwa procesu uwierzytelniania w oparciu o zastosowanie zasady 2FA (*ang. two-factor authentication)*, czyli w oparciu o kombinację dwóch różnych technik dystrybucji atrybutu uwierzytelniania użytkownika. Pierwszą techniką jest zazwyczaj znajomość loginu i hasła do konta, natomiast drugą może być fakt posiadania dostępu do urządzenia mobilnego. Użytkownik chcąc na przykład uzyskać dostęp do zasobów bankowości internetowej loguje się do portalu banku w klasyczny sposób – podając swój login i hasło. Wykonanie dyspozycji bankowych wymaga jednak już dodatkowej weryfikacji użytkownika na podstawie informacji przesłanych przez podmiot autoryzujący do urządzenia mobilnego podmiotu uwierzytelnianego. Najbardziej popularną metodą realizacji takiego scenariusza uwierzytelniania jest wysyłanie do użytkownika jednorazowego kodu dla danej transakcji w postaci wiadomości SMS (*ang. Short Message Service*), którym to kodem użytkownik legitymuje się w systemie w celu zatwierdzenia dyspozycji bankowej. Dzięki takiemu zabiegowi, dostawca usługi (w tym przykładzie bank) upewnia się, że dyspozycja nie jest składana przez nieuprawnioną osobę, która weszła w posiadanie loginu i hasła uprawnionego użytkownika na przykład metodą podsłuchu lub podglądu wizualnego. Unikalnym przedmiotem, jakim legitymuje się w takim przypadku użytkownik, jest karta SIM w telefonie. Dedykowane aplikacje mobilne do bankowości internetowej dodatkowo wiążą konto swojego użytkowania z danymi identyfikacyjnymi konkretnego urządzenia takimi jak na przykład numer IMEI telefonu, który jest unikalnym numerem przypisanym do urządzenia. Dzięki temu usługodawca może wykryć próbę logowania do własnych zasobów z innego urządzenia niż do tej pory, co może zaistnieć w przypadku, gdy osoba nieuprawniona wykonała klon karty SIM użytkownika uprawnionego.

Innym przykładem realizacji procesu uwierzytelniania użytkownika mogą być zastosowane w urządzeniach niektórych producentów (telefony, tablety) podzespoły umożliwiające przeprowadzenie uwierzytelniania na podstawie wybranych cech biometrycznych. Ciekawym przykładem może być tutaj zastosowany w najnowszych telefonach firmy Apple czytnik linii papilarnych, który oprócz procesu autoryzacji do zasobów telefonu ma dodatkowo zastosowanie w procesie autoryzacji do zasobów internetowych środowiska iTunes/Appstore, w którym użytkownik może dokonywać zakupów treści multimedialnych, gier czy programów dedykowanych do urządzeń firmy Apple. W tym przypadku, użytkownik legitymując się swoim odciskiem palca (tzw. *Touch ID*) potwierdza swoją tożsamość w systemie komputerowym.

Kolejną wartą przytoczenia nowoczesną techniką uwierzytelniania biometrycznego zastosowanego w urządzeniach mobilnych, jaką oferują producenci

wybranych modeli smartfonów czy tabletów, jest uwierzytelnianie na podstawie rozpoznawania twarzy użytkownika. W proces uwierzytelniania zaangażowana jest kamera urządzenia umieszczona nad wyświetlaczem, która jest uruchamiana przy próbie uzyskania dostępu. Użytkownik zbliżając urządzenie do swojej twarzy ustawia je tak, aby twarz rejestrowana przez kamerę znajdowała się w polu wskazanym na ekranie. W następnej kolejności urządzenie dokonuje akwizycji obrazu i porównuje go z zapisanym wcześniej obrazem twarzy właściciela. Implementację tego typu techniki uwierzytelniania znaleźć można między innymi w urządzeniach z systemem Android posiadających przednią kamerę. Zaznaczyć należy, iż implementacja tej metody w dzisiejszych urządzeniach mobilnych daleka jest od doskonałości, przede wszystkim dlatego, iż nie jest odporna na próbę uwierzytelniania na podstawie zdjęcia osoby uprawnionej, dlatego też nie powinna być stosowana jako jedyna technika zabezpieczenia zasobów. Na rysunku 26 zaprezentowane zostało ostrzeżenie dla użytkowników systemu Android dotyczące bezpieczeństwa metody uwierzytelniania na podstawie rozpoznawania twarzy.

Rysunek 26. Ostrzeżenie w systemie Android dotyczące stosowania autoryzacji na podstawie rozpoznawania twarzy użytkownika


Uwierzytelnianie na podstawie cech biometrycznych gwarantuje wysoki poziom bezpieczeństwa pod warunkiem, że system i urządzenia realizujące taki proces cechują się wysoką precyzją pomiaru, a programiści aplikacji obsługującej proces zadbali o bezpieczeństwo przechowywanych w systemie zapisów wzorcowych cechy biometrycznej. Techniki uwierzytelniania biometrycznego wzbudzają najwięcej kontrowersji spośród wszystkich opisanych technik, w szczególności jeśli technika ma służyć do uwierzytelniania w sieci Internet (jak w przypadku *Touch ID* firmy Apple), głównie ze względu na obawy dotyczące rzetelności w zakresie bezpieczeństwa przechowywania i zacho-

wania poufności danych biometrycznych użytkownika, które przechowywane są u operatora, który takiego uwierzytelniania dokonuje.

Opisane powyżej przykłady odnoszą się do zwiększania bezpieczeństwa uwierzytelniania użytkowników w systemach informatycznych przez wykorzystanie technologii mobilnych. Jednak to nie jedyny obszar, gdzie nowoczesne urządzenia i ich aplikacje przyczyniają się do zwiększania bezpieczeństwa użytkowników. Wskazać bowiem można szereg zastosowań w obszarze zdrowia i życia, gdzie wykorzystywane są technologie mobilne. Jednym z takich zastosowań jest monitoring lokalizacji (tzw. geolokalizacja). Zdecydowana większość dzisiejszych urządzeń mobilnych posiada wbudowane urządzenie do odbioru sygnału GPS (*ang. Global Positioning System*), które oprócz klasycznego zastosowania (jak na przykład nawigacja samochodowa, mapy) może być wykorzystywane do zapewnienia bezpieczeństwa osób i mienia. W tym celu nie wystarczy jednak sam bierny odczyt sygnału GPS, koniecznym jest także regularne i automatyczne przesyłanie odczytanego położenia urządzenia do zewnętrznego serwera łącząc się przez wbudowany w urządzenie modem lub na przykład za pomocą wiadomości SMS. Taki scenariusz wykorzystany został w wielu specjalistycznych aplikacjach i w mobilnych urządzeniach dedykowanych funkcjonalności śledzenia położenia (tzw. *geotracking*). Przykładowe urządzenie dedykowane zaprezentowane zostało na rys. 27. Urządzenie po odczytaniu swojego położenia zapisuje je w postaci punktu na mapie, wykorzystując do tego silnik Google Maps.

Rysunek 27. Lokalizator GPS. Urządzenie odczytuje swoje położenie na podstawie sygnału GPS i na bieżąco wysyła je do serwera

Rolę narzędzia do geolokalizacji użytkownika może spełniać także aplikacja zainstalowana w smartfonie, tablecie, czy smartwatchu. Aplikacja w analogiczny sposób jak urządzenia dedykowane odczytuje i raportuje swoje położenie. Nietrudno wyobrazić sobie ogromną liczbę zastosowań takiego rozwiązania. W pierwszej kolejności wskazać tu można aspekty bezpieczeństwa osób: monitorowanie miejsca przebywania dzieci lub osób starszych, monitoring pracowników mobilnych. Kolejnym zastosowaniem w zakresie bezpieczeństwa może być monitoring mienia ruchomego (samochody, motocykle, rowery, łodzie itd.) na wypadek kradzieży. W innym przypadku, kierujący akcją

ratunkową znając aktualne położenie jednostek ratowniczych (pojazdów, czy poszczególnych osób – np. patrolu policji), może zaangażować do akcji zespoły, które znajdują się najbliżej miejsca zdarzenia. Ciągły rozwój urządzeń mobilnych, za którym idzie miniaturyzacja i wzrost dostępności (niższe ceny) urządzeń i tego typu aplikacji powoduje, iż zastosowanie narzędzi do *geotrackingu* staje się coraz bardziej powszechne.

Ciekawym rozwiązaniem mobilnym mającym bezpośrednie odniesienie do zwiększania bezpieczeństwa użytkownika przemieszczającego się pojazdem jest zastosowanie wbudowanego w urządzenie żyroskopu w połączeniu z odczytem położenia GPS do wykrywania ewentualnego wypadku (kolizji pojazdu) celem automatycznego (w pełni lub po uproszczonym zatwierdzeniu przez użytkownika) wezwania służb ratunkowych. Urządzenie reaguje na wstrząs (żyroskop) połączony z nagłym zatrzymaniem przemieszczania się użytkownika (GPS). Osoba uczestnicząca w wypadku samochodowym przez kilkanaście minut może przebywać w szoku i skuteczne wezwanie pomocy może być prawdziwym wyzwaniem w szczególności, gdy ofiara wypadku znajduje się na terenie innego kraju niż ojczysty. Aplikacje tego typu posiadają bazę numerów alarmowych większości krajów świata, a fakt iż aplikacja dodatkowo pobiera informacje o położeniu użytkownika na podstawie sygnału GPS sprawia, że nie jest konieczne każdorazowe konfigurowanie aplikacji w zależności od kraju przez który aktualnie przejeżdża użytkownik. Kwestią przyszłości jest funkcjonalność automatycznego wzywania pomocy w języku natywnym dla kraju, w którym znajduje się użytkownik.

Zastosowanie technologii mobilnych do zapewnienia i zwiększania bezpieczeństwa użytkowników jest zagadnieniem szerokim i szybko rozwijającym się. Oprócz zastosowań omówionych w niniejszym podrozdziale spodziewać się można w niedalekiej przyszłości wielu innych obszarów, w których technologie mobilne pomagać będą użytkownikom w zakresie ich bezpieczeństwa. Trwają intensywne prace wielu ośrodków naukowych nad wykorzystaniem mobilnych urządzeń – w szczególności zestawienia smartfonów z urządzeniami pomiarowymi wykorzystującymi technologię bezprzewodowej komunikacji Bluetooth do ciągłego monitorowania funkcji życiowych osób, które takiego nadzoru wymagają. Rozwiązania takie na pewną znajdą szerokie zastosowanie, w szczególności dedykowane będą osobom przewlekle chorym, czy starszym. Już w tej chwili na rynku znaleźć można wiele mobilnych urządzeń (np. smartwatche), które posiadają na przykład wbudowany pulsometr. Jednakże na tym etapie służą użytkownikowi raczej do samokontroli aktywności serca na przykład podczas treningów sportowych, a scentralizowany pełny monitoring czynności życiowych analizowany półautomatycznie przez zespół nadzorujących specjalistów to raczej kwestia przyszłości.

## 5.2. Metody zabezpieczania urządzeń mobilnych przed nieuprawnionym do nich dostępem

W poprzednim podrozdziale opisane zostały technologie mobilne, których zadaniem jest wspieranie bezpieczeństwa użytkownika na polu uwierzytelnia-

nia do systemów informatycznych jak i jego bezpieczeństwa fizycznego (zdrowia i życia). Kolejnym ważnym zagadnieniem jest bezpieczeństwo samych urządzeń mobilnych, które jako nośnik danych wrażliwych użytkownika także mogą być celem ataku.

Podejmując temat bezpieczeństwa danych przechowywanych w urządzeniach mobilnych wskazać należy, jak bardzo ważne stały się urządzenia mobilne dla swoich użytkowników. Na przestrzeni kilkunastu lat urządzenia mobilne znacząco wyewoluowały zarówno w zakresie miniaturyzacji, zastosowanych podzespołów jak i ich funkcjonalności. Postęp ten pociągnął za sobą wzrost ilości danych jakie przechowywane są przez użytkowników w swoich urządzeniach – także danych uważanych za „czułe", które dla dobra i bezpieczeństwa użytkownika powinny mieć charakter prywatny. Do takich danych z pewnością należy lista kontaktów użytkownika, która w dzisiejszych urządzeniach jest bardzo rozbudowanym zbiorem danych. Dodatkowo oprócz numeru telefonu wpis dotyczący kontaktu może zawierać między innymi: adres e-mail kontaktu (prywatny i służbowy), nazwę firmy dla której kontakt pracuje, adres profilu w portalu społecznościowym, prywatną stronę WWW, notatki dotyczące kontaktu, a także zdjęcie kontaktu. Z punktu widzenia osoby chcącej przeprowadzić atak na ofiarę (niezależnie, czy ma to być atak cyfrowy, czy przestępstwo w klasycznej formie) informacje takie są bardzo istotne. Co więcej, atakujący niekoniecznie musi zdobyć informacje o obiekcie swojego ataku poprzez uzyskanie dostępu do jego urządzenia mobilnego, ale na przykład wchodząc w posiadanie urządzenia (smartfonu, czy tabletu) osoby będącej znajomą celu. Z przykładu tego płynie wniosek, że nasze bezpieczeństwo nie jest uzależnione jedynie od naszej dbałości o dane przechowywane w urządzeniach mobilnych, ale zależy także od dbałości i świadomości osób, którym te dane powierzamy. Innym przykładem danych mogących znacznie wpłynąć na obniżenie bezpieczeństwa użytkownika w przypadku, gdy zapozna się z nimi potencjalny atakujący, są wiadomości e-mail przechowywane przez aplikację kliencką w urządzeniu mobilnym. Treść wiadomości może dotyczyć aspektów zawodowych i prywatnych użytkownika, ale pośród wiadomości e-mail często znajdują się wiadomości aktywacyjne do portali internetowych, potwierdzenia przelewów bankowych, czy inne dane mogące obniżać bezpieczeństwo jego właściciela. Tego typu zagrożenia odnoszą się do biernego dostępu osoby nieuprawnionej do skrzynki wiadomości e-mail. Jeśli osoba nieuprawniona weszła w posiadanie urządzenia mobilnego, które nadal jest w stanie odbierać wiadomości e-mail użytkownika, zagrożenie jest dużo większe, ponieważ potencjalny agresor może na przykład dokonać procedury odzyskania dostępu do wybranego portalu internetowego, w którym konto posiada właściciel urządzenia, a która to procedura najczęściej wiąże się z wysłaniem przez portal wiadomości e-mail z adresem URL umożliwiającym zmianę hasła do portalu.

Jako przykłady innego typu danych, które przechowywane są w urządzeniach mobilnych wskazać należy fotografie, które mogą obniżać bezpieczeństwo zabezpieczeń fizycznych użytkownika. Agresor dokonując analizy fotografii użytkownika wyciągać może wnioski dotyczące rozkładu pomiesz-

czeń (domu, biura), zastosowanych środków ochrony fizycznej, ale także składu osobowego pracowników firmy, czy rodziny ofiary. Dodatkowym zagrożeniem, związanym z dostępem do urządzenia mobilnego przez osobę nieuprawnioną, jest możliwość pozyskania informacji z zainstalowanych przez użytkownika aplikacji. Największym zagrożeniem w tym zakresie są aplikacje utrzymujące dostęp do zasobów swojego serwisu. W takim schemacie działa większość aplikacji mobilnych – po zainstalowaniu użytkownik przechodzi proces uwierzytelniania jednorazowo, a aplikacja utrzymuje dostęp do zasobów portalu. Osoba nieuprawniona, która uzyskuje dostęp do urządzenia mobilnego automatycznie może uzyskać dostęp także do zasobów na kontach użytkowania w różnych portalach internetowych (na przykład aplikacje: Facebook, Google Plus, Endomondo), a w konsekwencji także do informacji (tekst, zdjęcia, geolokalizacja) publikowanych przez znajomych użytkownika, które najczęściej nie są widoczne dla osób postronnych. Podobny przykład stanowić mogą dedykowane aplikacje usług bankowych. Aplikacje te wymagają, w przeciwieństwie do wymienionych wyżej przykładów, przeprowadzenia każdorazowo procesu uwierzytelniania użytkownika, jednak pamiętać należy, iż sama już informacja o nazwie banku, w którym użytkownik posiada konto prowadzić może do pewnych zagrożeń, a atakujący może przeprowadzić inną formę ataku w oparciu o tę informację (*phishing*, atak socjotechniczny, itp.).

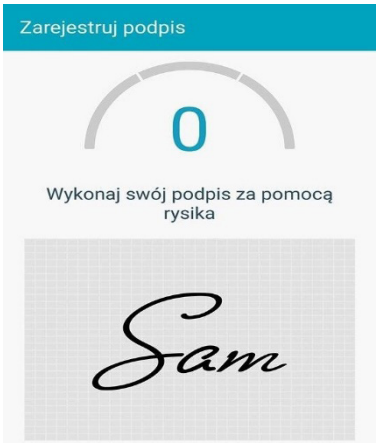
Wyżej przytoczone przykłady podkreślają konieczność stosowania technik ograniczania i kontroli dostępu do urządzenia mobilnego, ze względu na dużą liczbę przechowywanych w urządzeniach danych, które należ chronić. Pomijając aspekt bezpieczeństwa fizycznego urządzenia mobilnego, które także jest ważnym aspektem, jako najbardziej popularne techniki ochrony zasobów urządzenia wskazać można uwierzytelnianie użytkownika na podstawie pewnej unikalnej wiedzy mającej poświadczyć, iż jest on uprawnionym użytkownikiem (metoda „coś, co wiesz"). Wskazać tutaj można uwierzytelnianie na podstawie wprowadzanego do urządzenia pinu, hasła lub wzorca. Z technicznego punktu widzenia, pin i hasło to takie same formy zabezpieczenia, z tą różnicą, że pin obejmuje użycie jedynie znaków numerycznych (0-9), a hasło umożliwia zastosowanie liter, cyfr i znaków specjalnych. Dla przykładu: w systemie Android (wersja 5.0) możliwe jest użycie kodu pin i hasła o długości nie krótszej niż 4 znaki oraz nie dłuższej niż 16 znaków, co w przypadku kodu pin daje od $10^4$ do $10^{16}$ kombinacji dla potencjalnego włamywacza, a w przypadku hasła (system umożliwia zastosowanie 93 znaków do budowy hasła) daje od $7.48 \cdot 10^7$ do $3.13 \cdot 10^{31}$ kombinacji. Metoda zabezpieczenia na podstawie wzorca dla przywołanego systemu (rys. 28) umożliwia użytkownikowi zdefiniowanie wzoru odblokowującego urządzenie. Wzór może minimalnie posiadać 4, a maksymalnie 9 sekwencji. Liczba kombinacji zastosowanych sekwencji wynosi odpowiednio (w zależności od długości wybranej przez użytkownika sekwencji wzorca): 1624 (dla 4 kroków), 7152 (dla 5 kroków), 26016 (dla 6 kroków), 72912 (dla 7 kroków), 140704 (dla 8 kroków), 140704 (dla 9 kroków). Zauważyć więc można, że dopiero 6 krokowa sekwencja zapewnia większe bezpieczeństwo (biorąc pod uwagę liczbę możliwych kombinacji) od 4 znako-

wego kodu pin, a najdłuższa sekwencja (9 krokowa) nie zapewnia takiej liczby kombinacji jak najkrótsze możliwe, czyli 4 znakowe hasło.

Rysunek 28. Ekran definicji wzorca zabezpieczającego dostęp do urządzenia w systemie Android


Ostatnią grupę zabezpieczeń dostępu do urządzeń mobilnych stanowi uwierzytelnianie biometryczne. Do grupy tej należy opisane w poprzedniej części rozpoznawanie twarzy (rys. 26), uwierzytelnianie na podstawie odcisku palca, czy na przykład należące do grupy metod biometrycznych behawioralnych rozpoznawanie podpisu odręcznego (rys. 29, dla urządzeń posiadających rysik).

Rysunek 29. Ekran definicji wzorca podpisu jako metoda uwierzytelniania do urządzenia w systemie Android



Zaznaczyć jednak należy, iż urządzenia mobilne dostępne na rynku nie posiadają wbudowanych modułów do uwierzytelniania biometrycznego, które gwarantowałyby wysoką jakość akwizycji cech biometrycznych. Dlatego nawet autorzy systemu wskazują, iż metody biometryczne zaimplementowane w dzisiejszych urządzeniach mobilnych gwarantują najniższy poziom bezpieczeństwa spośród przedstawionych w tym rozdziale metod uwierzytelniania użytkownika. Firma Apple w swoim najnowszym urządzeniu – iPhone 6, zastosowała najnowocześniejszy do tej pory na rynku czytnik linii papilarnych, jednak już po 48 godzinach od premiery urządzenia zaprezentowane zostały w sieci Internet metody ominięcia tego zabezpieczenia poprzez wydruk w wysokiej rozdzielczości na specjalnej przeźroczystej folii odcisku palca właściciela urządzenia, który to odcisk zebrany został z przedmiotu, którego dotykał właściciel (z wykorzystaniem metod klasycznej kryminalistyki).

Podsumowując stwierdzić należy, iż ze względu na coraz większą liczbę „czułych" danych przechowywanych w urządzeniach mobilnych konieczne jest stosowanie kontroli dostępu do urządzeń celem ich ochrony przed dostępem fizycznym osób nieuprawnionych. Najwyższy poziom gwarantują hasła, jednak w przypadku urządzeń mobilnych są one najmniej wygodne z użytkowego punktu widzenia (konieczność każdorazowego wprowadzania hasła). Rozwój technologiczny sprawia, iż kolejne urządzenia oferowane na rynku posiadają coraz więcej pamięci, przez co potencjalna liczba danych skradzionych wraz z urządzeniem może być większa. Największa zaleta urządzeń mobilnych jest zarazem ich największą wadą (pod kątem bezpieczeństwa danych) – urządzenia posiadają niewielkie gabaryty, co ułatwia ich kradzież lub przypadkowe zgubienie. Całkowite wyeliminowanie ryzyka pozyskania danych z urządzenia przez osobę nieuprawnioną wymagałoby przeniesienia wszystkich wrażliwych danych użytkownika poza urządzenie mobilne do jednego z wielu serwisów oferujących przechowywanie plików (tzw. *storage clouds*), oczywiście przy założeniu, iż aplikacja służąca do połączenia z serwisem przechowywania plików nie utrzymuje do niego stałego dostępu i użytkownik musi każdorazowo uwierzytelniać się w serwisie. Jednak w tym przypadku wątpliwości budzić może bezpieczeństwo powierzonych danych u operatora takiej usługi.

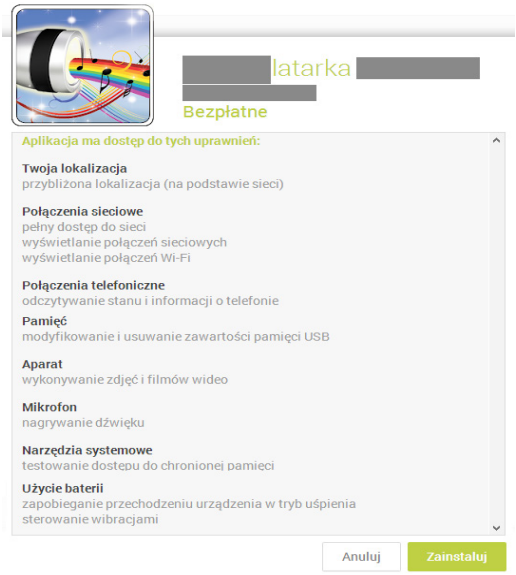
## 5.3. Techniki szpiegowania urządzeń mobilnych

W poprzednim podrozdziale poruszona została tematyka zabezpieczeń urządzeń mobilnych przed nieuprawnionym do nich dostępem fizycznym, którego konsekwencją może być uzyskanie dostępu przez osobę nieuprawnioną do danych przechowywanych w urządzeniu. Jednak dane użytkowników i dane o użytkownikach mogą być pozyskiwane przez osoby nieuprawnione także za pomocą innych technik, nie związanych z fizycznym uzyskaniem dostępu do urządzenia. Do takich zagrożeń należą szeroko pojęte techniki szpiegowania użytkowników rozwiązań mobilnych. Wyróżnić można aplikacyjne metody pozyskiwania informacji o użytkowniku (aplikacje szpiegujące) oraz techniki infrastrukturalne (na przykład: klonowanie kart SIM, klonowanie BTS itp.). Dlaczego użytkownicy technologii mobilnych są narażeni na takie

zagrożenia? Odpowiedź jest prosta: urządzenia mobilne stały się powszechne i łączą w sobie szeroką funkcjonalność. Służą do komunikacji, pozyskiwania i przechowywania informacji, wykonywania zdjęć, filmów, nagrań dźwięku, a także przechowują informacje o geolokalizacji (aktualnej lub historycznej) użytkownika. Są skarbnicą informacji, która służyć może nie tylko przestępcom, czy na przykład służbom specjalnym i wywiadowczym, ale także przedsiębiorcą, którzy chcą pozyskiwać jak najwięcej informacji o potencjalnych klientach.

Użytkownik instalując aplikację w swoim urządzeniu, wyraża zgodę na dostęp aplikacji do wybranych funkcji systemu mobilnego. Aplikacja powinna żądać dostępu do takich funkcji urządzenia mobilnego, jakie są jej niezbędnie potrzebne do funkcjonowania. Jednak wiele aplikacji żąda szerszych uprawnień niż wydaje się to konieczne. Na rys. 30 przedstawiono przykładową aplikację „latarka", której jedyną funkcją jest uruchomienie lampy błyskowej aparatu fotograficznego smartfonu, przez co telefon pełnić może funkcję latarki.

Rysunek 30. Przykładowa aplikacja żądająca szerszych praw dostępu niż to wynika z realnych potrzeb jej funkcjonowania (nazwa aplikacji i producenta celowo zanonimizowana)




Jak można zauważyć, aplikacja pomimo ograniczonej funkcjonalności żąda szeregu uprawnień. O ile zrozumiałe jest żądanie dostępu do aparatu fotograficznego telefonu (uruchomienie lampy błyskowej aparatu), do odczytywania stanu telefonu (w trakcie połączenia przychodzącego aplikacja zostanie zamknięta), czy użycia baterii (aby urządzenie nie przechodziło w tryb uśpienia), to całkowicie niezrozumiałym jest żądanie dostępu do lokalizacji użytkownika, połączenia sieciowego, pamięci USB (do plików użytkownika) i do funkcji nagrywania dźwięku. Przykład ten nie jest odosobniony. Podob-

nych aplikacji można znaleźć wiele w sklepach producentów systemów operacyjnych urządzeń mobilnych (portale dystrybucji oprogramowania mobilnego). Pośród funkcji systemu, które są najczęściej żądane przez aplikacje, pomimo że funkcjonalność aplikacji nie wymaga ich bezpośrednio, jest informacja o geolokalizacji użytkownika. Określanie fizycznej lokalizacji użytkowników na podstawie samego adresu IP, bazując na bazach danych przedsiębiorców telekomunikacyjnych, nie jest doskonałe, ponieważ adresy fizyczne wskazywane w tych bazach są adresami rejestracji działalności gospodarczej przedsiębiorcy telekomunikacyjnego, a nie klienta używającego w danym czasie tego adresu IP. Dlatego też zbieranie informacji o geolokalizacji użytkowników urządzeń mobilnych wraz z ich adresami IP stało się intratnym biznesem. Producent tworząc aplikację, która wymaga dostępu do geolokalizacji staje się posiadaczem informacji o geolokalizacji danego adresu IP. Informację taką może wykorzystywać komercyjnie, jeśli użytkownik zezwolił na to, akceptując odpowiednie zapisy licencji instalowanego oprogramowania.

Żądanie dostępu przez aplikację do danej funkcjonalności nie oznacza jeszcze, iż aplikacja ta szpieguje użytkownika, jednak to, że użytkownik wyraził zgodę i zainstalował taką aplikację, umożliwia producentowi aplikacji takie działania z technicznego punktu widzenia. Natomiast z punktu widzenia prawa, fakt, iż użytkownik wyraził zgodę na dostęp aplikacji do danej funkcjonalności urządzenia mobilnego, nie upoważnia jeszcze producenta aplikacji do pozyskiwania informacji o użytkowniku z wykorzystaniem tej funkcjonalności, chyba że informacja o możliwości zbierania takich danych znajdowała się w zapisach licencji programu, którą użytkownik zaakceptował. Kontrowersyjnym jest też brak reakcji producentów systemów operacyjnych urządzeń mobilnych, którzy są też głównymi właścicielami platform dystrybucji aplikacji (Google Play, Apple Itunes, Windows Phone Marketplace) na aplikacje żądające zbyt dużych uprawnień. Domniemywać można, iż głównym problemem jest tutaj skala (jak podaje producent – firma Google, średnio co miesiąc do Android Marketu trafia średnio 30 tysięcy nowych aplikacji). Przy tak dużej liczbie publikowanych programów, rzetelne sprawdzenie, czy każda aplikacja potrzebuje uprawnień, których żąda, wydaje się być niemożliwe, gdyż proces wymaga oceny człowieka i nie jest możliwy do automatyzacji.

Oprócz aplikacji szpiegujących zagrożeniem dla użytkowników urządzeń mobilnych mogą być także bardziej wyrafinowane metody podsłuchu wykorzystujące techniki infrastrukturalne. Przykładem może być zabieg uruchomienia fałszywej stacji BTS (*ang. Base Transceiver Station*), która będzie pośredniczyć w komunikacji, pomiędzy telefonem użytkownika, a właściwą stacją BTS. Telefon użytkownika korzysta ze stacji BTS, której sygnał w danej lokalizacji jest najsilniejszy. Jeśli atakujący dysponuje odpowiednim sprzętem może wprowadzić w błąd urządzenie, które rozpocznie komunikację poprzez fałszywą stację BTS uruchomioną przez atakującego. Atakujący łączy się następnie z właściwą stacją, tak aby mogło dojść do połączenia pomiędzy podsłuchiwanym użytkownikiem, a jego rozmówcą. Sprzęt do tego typu zabiegów należy do drogich rozwiązań, ale niestety jego sprzedaż nie jest reglamentowana.

Urządzenia mobilne do połączeń z siecią Internet mogą używać połączeń WiFi. W stosunku do tej technologii także istnieją ataki, które na celu mają podsłuch komunikacji użytkownika. Fałszywe punkty dostępu o nazwach SID sugerujących zaufaną instytucję, czy techniki klonowania innych zapamiętanych przez urządzenie mobilne sieci WiFi, prowadzić mogą do uzyskania dostępu do atakowanej sieci, a następnie do podsłuchu transmisji. Ataki tego typu oczywiście mogą być wymierzone nie tylko w urządzenia mobilne, jednak to urządzenia tego typu pozostając w stanie włączonym przemieszczają się wraz ze swoim użytkownikiem i w tym czasie mogą nieustannie próbować wyszukiwać zapamiętane sieci WiFi i logować się do nich. Stanowi to dobrą okazję dla agresora do przeprowadzenia ataku wykorzystującego fałszywy punkt dostępowy w lokalizacji bardziej dogodnej dla atakującego. Przeprowadzanie takiego ataku na przykład z samochodu na ulicy będzie łatwiejsze, niż w budynku firmy, gdzie intruz może zostać zidentyfikowany. Dlatego właśnie użytkownicy rozwiązań mobilnych powinni w bardziej świadomy sposób zarządzać połączeniami WiFi poprzez dezaktywację automatycznego łączenia się do znanych sieci lub poprzez wyłączanie karty WiFi, gdy nie jest używana.

Polskie prawodawstwo odnosi się do przestępstwa podsłuchu w artykule 267 § 3 Kodeksu karnego, który wskazuje, iż karą grzywny, karą ograniczenia albo pozbawienia wolności do lat 2 zagrożony jest każdy, kto „*w celu uzyskania informacji, do której nie jest uprawniony, zakłada lub posługuje się urządzeniem podsłuchowym, wizualnym albo innym urządzeniem lub oprogramowaniem*". Przepis w jasny sposób definiuje, iż oprogramowanie (w tym także aplikacje mobilne) wykorzystywane do uzyskania informacji w sposób nieuprawniony także podlegają pod wskazane regulacje. Pamiętać jednak należy, iż według zapisów §5 przytoczonego artykułu: „*ściganie przestępstwa określonego w § 1-4 następuje na wniosek pokrzywdzonego*", zatem poszkodowany sam musi mieć świadomość, iż do takiego przestępstwa doszło i w celu ścigania osoby odpowiedzialnej musi złożyć zawiadomienie o zaistniałym procederze.

Podsumowując, stwierdzić należy, iż użytkownicy aplikacji mobilnych powinni dokładnie weryfikować uprawnienia jakie żądane są przez instalowane aplikacje. Jeśli żądane uprawnienia są zbyt szerokie, instalacja takiej aplikacji niesie za sobą ryzyko inwigilacji. Kluczowe jest tutaj zaufanie użytkownika do producenta danego oprogramowania oraz świadomość użytkownika o zagrożeniach wynikających z eskalacji praw dostępu aplikacji jak i z możliwości infrastrukturalnego podsłuchu.

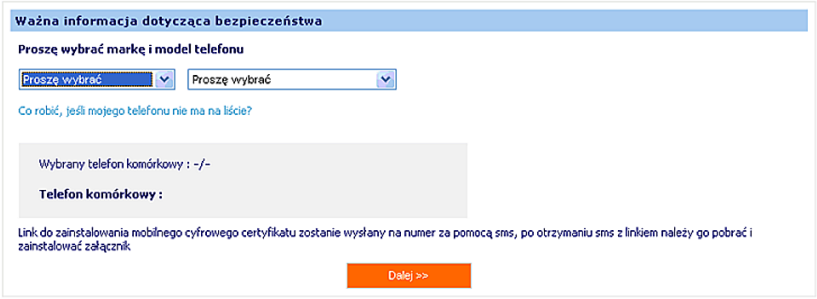
## 5.4. Ataki na użytkowników technologii mobilnych

Przedstawione w poprzednim podrozdziale problemy związane ze szpiegowaniem użytkowników urządzeń mobilnych to poważne, ale nie jedyne zagrożenie wymierzone w mobilne technologie. Wyróżnić można także inne, złożone ataki, które angażują urządzenia mobilne, choć nie zawsze są one celem głównym ataku.

Jednym z najgłośniejszych ataków, którego elementem było wykorzystanie telefonu użytkownika, był wirus Zeus. Wirus ten był wymierzony w banko-

wość elektroniczną, a głównym celem atakujących było pozyskanie środków finansowych ofiary. W pierwszej kolejności ofiara infekowana była złośliwym oprogramowaniem na swoim komputerze (wirus dystrybuowany był klasycznymi metodami, na przykład jako załącznik w wiadomościach e-mail, albo wraz z pirackim oprogramowaniem). Wirus w momencie logowania się użytkownika do internetowej witryny banku podmieniał tę stronę na własną – wyglądającą identycznie jak właściwa witryna banku, a następnie po podaniu przez użytkownika swojego loginu i hasła, wysyłał dane potrzebne do zalogowania się atakującemu. W następnej kolejności, użytkownik po zalogowaniu się do fałszywej strony banku widział ostrzeżenie (które także przygotowane było przez atakującego) dotyczące konieczności instalacji *„mobilnego cyfrowego certyfikatu”* (rys. 31). Rzekomym powodem takiego zabiegu miała być chęć zwiększenia bezpieczeństwa usługi bankowej poprzez wysyłanie do użytkownika jednorazowych kodów autoryzujących transakcję drogą szyfrowaną.

Rysunek 31. Wirus Zeus: podmieniona strona internetowa bankowości elektronicznej z prośbą o wskazanie marki i modelu telefonu użytkownika




Zauważyć można, iż wirus żądał od użytkownika wskazania marki i modelu telefonu (a we wcześniejszym kroku także numeru telefonu). Zabieg ten był konieczny, ponieważ w następnym kroku ataku użytkownik otrzymywał wiadomość SMS z adresem URL do rzekomego certyfikatu cyfrowego, który tak naprawdę był kolejnym złośliwym programem, tym razem wymierzonym w system operacyjny urządzenia mobilnego. Wirus posiadał implementacje dla wszystkich najpopularniejszych systemów urządzeń mobilnych, dlatego właśnie w pierwszym kroku atakujący oprócz loginu i hasła użytkownika, chcieli pozyskać dane dotyczące modelu, marki i numeru telefonu ofiary. Wirus na urządzeniu mobilnym przekierowywał jednorazowe kody autoryzujące transakcję do atakującego w taki sposób, że ofiara nie zauważała, że taki SMS z kodem został przesłany do jej telefonu (wirus po przesłaniu SMS z kodem do atakującego usuwał go z telefonu ofiary, a telefon nie wydawał dźwięku dla przychodzącej z banku wiadomości).

Mniej złożonym technicznie przykładem zagrożeń wymierzonych w użytkowników technologii mobilnych, mogą być próby zachęcenia użytkownika do przejścia na konkretną stronę internetową z wykorzystaniem kodów QR. Kody QR to dwuwymiarowe (kwadratowe) kody graficzne, które doskonale spraw-

dziły się na potrzeby technologii mobilnych. W kodzie QR zapisana może zostać dowolna wiadomość, wizytówka kontaktu, adres e-mail, czy adres URL. Użytkownik do odczytania kodu wykorzystuje dedykowaną aplikację, która za pomocą wbudowanego aparatu fotograficznego urządzenia dokonuje skanu kodu. Dzięki kodom QR, użytkownik może w łatwy sposób zapisać w telefonie dane kontaktowe lub przejść do strony internetowej za pomocą prostej czynności polegającej na wykonaniu zdjęcia kodu. Przykładowy kod z zakodowanym adresem URL zaprezentowany został na rys. 32.

Rysunek 32. Przykładowy kod QR zawierający zakodowany adres do strony WWW

Metoda pozwala użytkownikowi w łatwy sposób przejść na stronę internetową, co jest szczególnie wygodne, gdy adres strony wskazany jest na plakacie, reklamie, ulotce itp. Jednak technika ta może nieść za sobą także pewne ryzyko. Wyobrazić można sobie sytuację, w której atakujący umieszcza na kodzie QR dostępnym w miejscu publicznym (reklama, plakat, ogłoszenie) przygotowany przez siebie kod QR, który prowadzi do przygotowanej przez atakującego strony internetowej ukierunkowanej na wyłudzenie danych użytkownika (strony łudząco przypominającej tę, której spodziewał się użytkownik). Atakujący w celu uśpienia czujności swojej ofiary może dodatkowo posłużyć się zbliżonym adresem domenowym lub umieścić dla niepoznaki w adresie URL adres IP serwera, na którym umieszczona została fałszywa strona.

Opisane powyżej ataki są atakami, które dedykowane są nowoczesnym urządzeniom mobilnym. Bez zaawansowanych funkcji smartfonu powyższe ataki nie byłyby możliwe do realizacji. Jednak nadal wskazać można klasyczne ataki, których historia sięga pierwszych telefonów komórkowych, a są one nadal wykorzystywane przez atakujących. Do tego typu zagrożeń należy atak o nazwie SMiShing (*SMS phishing*), polegający na wysyłaniu do ofiary wiadomości SMS mającej na celu przekonanie jej do podjęcia jakiejś akcji, która będzie korzystna dla osoby atakującej. Atak ukierunkowany jest głównie na przekonaniu ofiary, że znalazła się w jakimś niekorzystnym dla siebie położeniu i w celu odwrócenia tej sytuacji ofiara musi podjąć jakąś konkretną czynność. Przykładem ataku może być plaga wysyłanych w bieżącym roku wiadomości SMS o treści: „*Usługa ABC została aktywowana. Koszt usługi: 6,15 zł/dzień. Aby wyłączyć, wyślij SMS o treści AP na numer XYZ*". Wiadomość wysyłana jest do losowych osób, a celem atakującego jest przekonanie odbiorcy o nieprawdziwym fakcie włączenia dla niego kosztownej usługi. Atakujący chce przekonać swoją ofiarę, że musi niechcianą usługę wyłączyć wysyłając

wiadomość SMS o konkretnej treści na wskazany numer. Jednak to właśnie ta czynność jest związana z wysoką opłatą, która trafia do agresora. Atakujący zarejestrował wcześniej u operatora usługę płatności wiadomością SMS (najczęściej posługując się fałszywymi danymi i kontem bankowym założonym w nielegalny sposób na nieprawdziwe dane). Atak ten, choć ma już prawie 20 lat nadal jest z powodzeniem wykorzystywany.

Ataki prowadzone z wykorzystaniem nowoczesnych urządzeń mobilnych są trudne do wykrycia, ponieważ w większości przypadków są zjawiskiem nowym, a użytkownicy nie są przygotowani na dane incydent bezpieczeństwa. Przeciwdziałać tego typu atakom można jedynie poprzez podnoszenie świadomości użytkowników urządzeń mobilnych o tego typu zjawiskach.

## 5.5. Przyszłość technologii mobilnych w zakresie bezpieczeństwa danych

Przedstawione w niniejszym rozdziale aspekty bezpieczeństwa ukazują, że urządzenia mobilne zarówno mogą być skutecznie wykorzystywane do zwiększania bezpieczeństwa użytkowników sieci, ale także mogą być celem ataków wymierzonych w dane użytkownika (szpiegostwo) lub mające na celu doprowadzenie użytkownika do straty finansowej. Informacja w dzisiejszym świecie jest cennym towarem, a urządzenia mobilne stały się ogromnym agregatem informacji o użytkowniku, dlatego domniemywać można, iż wraz z dalszym rozwojem technologii mobilnych rozwijane będą także techniki ataków na urządzenia mobilne. Analiza przedstawionych w rozdziale przykładów zagrożeń nie pozwala na wskazanie jedynej słusznej technologii, która pozwoliłaby skutecznie wyeliminować wszelkie zagrożenia bezpieczeństwa. Dostęp do danych wrażliwych przechowywanych w urządzeniu mobilnym zabezpieczony może być za pomocą szyfrowania systemu plików urządzenia, jednak technologia ta musiałaby być transparentna dla użytkownika. Zatem problem znowu sprowadzi się do uwierzytelniania użytkownika w systemie mobilnym. Dobrym rozwiązaniem są techniki uwierzytelniania biometrycznego, ale te wymagają zastosowania lepszej klasy sprzętu do akwizycji cech biometrycznych, ponieważ aktualnie dostępne moduły urządzeń mobilnych nie gwarantują dostatecznego bezpieczeństwa tych metod. Innym pomysłem jest przechowywanie danych w zewnętrznych serwisach przechowywania plików (tzw. *storage clouds*), jednak tutaj najwięcej wątpliwości budzi bezpieczeństwo danych u dostawcy usługi, bowiem użytkownik nie ma pewności, kto będzie mógł uzyskać dostęp do tych danych oprócz niego.

Ataki wymierzone w użytkowników technologii mobilnych, które oparte są o zabiegi socjotechniczne są szczególnie groźne, ponieważ działanie atakującego najczęściej doprowadza do sytuacji, w której to sam użytkownik podejmuje niekorzystne dla siebie działania (w przytoczonym przykładzie wirusa „Zeus" użytkownik sam ściągał i uruchamiał złośliwą aplikację na swoim smartfonie wierząc, iż pobrany plik jest certyfikatem cyfrowym banku). Stwierdzić zatem można, iż tylko poprzez zwiększanie wiedzy i świadomości użytkowników o istniejących zagrożeniach oraz o metodach im przeciwdziałania możliwe jest skuteczne zwiększanie bezpieczeństwa użytkowników mobilnych rozwiązań.

# 6

# DYSTRYBUCJA I MARKETING APLIKACJI MOBILNYCH

## 6.1. Wstęp

Aplikacje mobilne zdobyły swoją popularność na przestrzeni ostatnich 20 lat, przy czym największą dynamikę wzrostu rynku obserwujemy od ok. 2008 roku. W najbliższych latach ich wzrost utrzyma rosnącą dynamikę, należy także spodziewać się dalszej penetracji rynku, w szczególności w zakresie aplikacji biznesowych. Obecnie uważa się, że na rynku występuje duży niedostatek deweloperów biznesowych aplikacji mobilnych, a popyt przekracza podaż nawet kilkukrotnie.

W niniejszym rozdziale omówione zostaną podstawowe aspekty dystrybucji i marketingu aplikacji mobilnych. Przedstawiona zostanie koncepcja lejka aplikacji mobilnej i omówione zostaną jego poszczególne etapy. Następnie użytkownik zapozna się z formami dystrybucji aplikacji mobilnych i rynkiem dystrybucji aplikacji mobilnych. Przedstawiona zostanie wielkość rynku i prognozy wzrostu. W dalszym toku omówione zostaną kanały dystrybucji aplikacji mobilnych. W końcowej części rozdziału przedstawione zostanie nieco szersze omówienie najważniejszych aspektów kanałów dystrybucji aplikacji mobilnych.

## 6.2. Lejek aplikacji mobilnych

Aby dobrze zrozumieć zagadnienia związane z dystrybucję i marketingiem aplikacji mobilnych, należy zapoznać się z koncepcją lejka aplikacji mobilnych (rys. 33).

Zgodnie z tą koncepcją, cykl wzrostu aplikacji mobilnej podzielony jest na szereg etapów. Odpowiadają one przejściu użytkownika od etapu świadomości o istnieniu aplikacji, przez jej pobranie i instalację, używanie, aż po moment, w którym jest gotowy zaprosić do niej innych użytkowników.

Pierwszym z nich jest etap świadomości – to etap, na którym odbiorca nabiera świadomości o istnieniu aplikacji. Przyczyniają się do tego przede wszystkim działania marketingowe. Będą to działania marketingowe w mediach on-line, mediach tradycyjnych i kampanie nabywania użytkowników. Ale do działań takich zaliczamy także utrzymywanie relacji z właścicielem sklepu dystrybucji cyfrowej, mogące prowadzić do uzyskania tzw. featuringu, czyli promocji aplikacji na witrynie sklepu dystrybucji cyfrowej. W ramach etapu świadomości istotne są też optymalizacje typu SEO (ang. *Search Engine Optimization* – optymalizacje pod kątem wyszukiwania).

Drugim etapem jest etap pozyskiwania, na tym etapie wpływ mają m.in. oceny i recenzje aplikacji, ale także jej opis, ikona, prezentacja na tle innych aplikacji. Kluczową częścią tego etapu jest też optymalizacja tzw. „*landing page*" aplikacji, czyli strony w sklepie dystrybucji cyfrowej, na którą trafia użytkownik skierowany do aplikacji przez zewnętrzne źródła lub też w wyniku przeprowadzenia procesu wyszukiwania.

Rysunek 33. Lejek aplikacji mobilnej


Trzeci etap to aktywacja. Polega on na wkroczeniu przez użytkownika w fazę regularnego korzystania z aplikacji. Następuje to w momencie, w którym aplikacja spełnia potrzeby użytkownika, robiąc to w sposób efektywny i przekonywujący. Etap aktywacji może zostać pogłębiony przez zbudowanie dobrego doświadczenia użytkownika i zaprojektowanie aplikacji w sposób ukierunkowany na dobry odbiór aplikacji.

Zaktywowany użytkownik przechodzi następnie do etapu czwartego, retencji. Etap retencji polega na podtrzymaniu zaangażowania użytkownika. Retencję określa się przez tzw. parametr retencji dnia n-tego. Definiuje się go w ten sposób, że „retencja dnia n-tego" to procentowo wyrażony stosunek liczby użytkowników powracających do aplikacji w n-tym dniu po instalacji aplikacji do liczby użytkowników, którzy zainstalowali aplikację tego samego dnia:

$$r(n) = \frac{u_n}{u_0}$$

gdzie:
$r$ – retencja
$n$ – dzień, dla którego mierzymy retencję
$u_n$ – liczba użytkowników, którzy zainstalowali aplikację w dniu 0 i byli aktywni n-tego dnia po instalacji
$u_0$ – liczba użytkowników, którzy zainstalowali aplikację w dniu 0

Retencję typowo mierzy się dla 1, 7 i 28-go dnia po instalacji. Retencja będzie tym wyższa, im większe będzie zaangażowanie użytkownika i im bardziej zbudowana zostanie jego lojalność.

Kolejnym etapem jest przychód (jeżeli aplikacja posiada dodatkowe kanały przychodu, takie jak *in-app purchases*). Wpływ na przychód na tym etapie ma przede wszystkim to, jaką wartość postrzega dla siebie użytkownik i ile jest gotów zapłacić za otrzymanie tej wartości. Istnieje wiele sposobów mierzenia przychodu w aplikacji mobilnej – jednym z typowych parametrów jest ARPDAU (Average Revenue Per Daily Active User) – średni przychód na dziennie aktywnego użytkownika. Możemy obliczyć go ze wzoru:

$$ARPDAU = \frac{\sum_{i=0}^{n} \frac{p(i)}{a(i)}}{n}$$

gdzie:
$ARPDAU$ – średni przychód na dziennie aktywnego użytkownika
$p(i)$ – przychód dnia $i$-tego
$a(i)$ – liczba aktywnych użytkowników dnia $i$-tego
$n$ – liczba dni, dla których mierzymy ARPDAU

Ostatnim etapem jest etap referencji, na którym zadowolenie użytkownika przekuwa się na polecanie aplikacji nowym użytkownikom. W najbardziej korzystnym przypadku liczba użytkowników instalujących aplikację dzięki temu, że pozyskali informacje o aplikacji od innych użytkowników, może być na tyle duża, że aplikacja będzie miała tzw. wzrost wirusowy – będziemy obserwować wzrost nawet bez zaangażowania marketingu.

Decyduje o tym współczynnik wiralności, który można zdefiniować następująco:

$$w = \frac{n}{p}$$

gdzie:
$w$ – współczynnik wiralności
$n$ – liczba nowych instalacji aplikacji z polecenia przez dotychczasowych użytkowników
$p$ – liczba użytkowników polecających aplikację

Wzrost wirusowy obserwujemy w momencie, gdy $w > 1$. Wzrost taki można stymulować przez integrację platform społecznościowych, czy też proszenie użytkowników o ocenę aplikacji w obrębie sklepu dystrybucji cyfrowej. Ten etap wpływa na zwiększenie liczby użytkowników, którzy przystępują do 1 i 2 etapu.

W dalszej części tego rozdziału poruszone zostaną zagadnienia związane z pierwszymi dwoma etapami, a więc świadomością i pozyskiwaniem. Stanowią one punkt wejściowy lejka i to od nich zależy jak dużo użytkowników trafi do ostatnich etapów lejka, a więc przekuje się na wzrost aplikacji i wzrost przychodów z monetyzacji.

## 6.3. Formy dystrybucji aplikacji mobilnych

Na długo przed tym, zanim na rynku zagościły smartfony i tablety, a wcześniej urządzenia typu PDA (Personal Digital Assistant), rynek mobilny istniał w postaci przenośnych konsol gier wideo (tzw. *Handhelds*). Początkowo urządzenia te miały wbudowane oprogramowanie (gry) na stałe, lecz z czasem powstały pierwsze urządzenia z wymiennymi grami, sprzedawanymi w postaci modułów elektronicznych zwanych kartridżami. Sprzedaż ta miała charakter fizyczny, odbywała się za pomocą tradycyjnych kanałów sprzedaży (przede wszystkim sklepy). Wiązało się to z kilkoma istotnymi ograniczeniami i problemami:

- Konieczność dostępu do fizycznych sieci dystrybucji,
- Trudności z dostępem do rynków zagranicznych,
- Brak możliwości aktualizowania aplikacji,
- Kanibalizacja sprzedaży przez dostępność używanych egzemplarzy,
- Wysoka cena dla odbiorcy końcowego.

Wraz z pojawieniem się urządzeń PDA, smartfonów i internetu, zaczęto wprowadzać cyfrowe formy dystrybucji. Istniały co prawda modele telefonów, które część oprogramowania dystrybuowały w formie fizycznych kartridży (np. Nokia N-Gage, rys. 34), ale jednak zdecydowana większość oprogramowania dystrybuowana zaczęła być w formie cyfrowej.

Początkowo dystrybucja cyfrowa miała formę rozproszoną. Dostępne były liczne strony internetowe i inne kanały oferujące dostęp do aplikacji mobilnych. Płatności za nie realizowane były często przez wysyłanie płatnych SMSów na numery premium.

Rysunek 34. Telefon Nokia N-Gage z widocznym kartridżem z oprogramowaniem


Instalowanie nowych aplikacji i płatności nie były komfortowe dla użytkownika. Instalacja była często bardzo skomplikowanym procesem. Sama procedura zakupu często była dość złożonym procesem. Użytkownik musiał sam znaleźć wersję kompatybilną ze swoim urządzeniem, następnie zanotować kod do pobrania aplikacji i numer telefonu, wysłać SMS z kodem na ten numer, a następnie otrzymywał dopiero link do pobrania aplikacji. Im więcej etapów w zakupie, tym większe prawdopodobieństwo rozmyślenia się użytkownika i powstrzymania przed finalizacją zakupu.

Aplikacje z kolei miały problemy z kompatybilnością, ze względu na wiele platform sprzętowych obsługujących dany format aplikacji i niekompatybilności na poziomie API i systemów operacyjnych różnych urządzeń. To wpływało na dalsze etapy lejka, a w efekcie zmniejszało liczbę użytkowników skłonnych do pobrania aplikacji.

Przełom w dystrybucji cyfrowej nastąpił w momencie wejścia na rynek firmy Apple ze sklepem AppStore dla telefonów iPhone. Apple uprościło proces instalacji i płatności – żadna inna firma nie mogła pochwalić się tak dużą liczbą klientów z kartą płatniczą lub kredytową powiązaną bezpośrednio z kontem w sklepie dystrybucji cyfrowej. Firma Apple zapewniła także bardzo wysoką kompatybilność aplikacji z urządzeniami opartymi o własny system operacyjny. Było to o tyle łatwe, że była ona twórcą i właścicielem zarówno platformy oprogramowania, jak i platformy sprzętowej swoich rozwiązań. Apple zapoczątkowało dynamiczny wzrost mobilnych rynków aplikacji. Wkrótce po Apple do rynku tego dołączyły firmy takie, jak Google, Microsoft, Samsung i wiele innych.

## 6.4. Rynek dystrybucji aplikacji mobilnych

Rys. 35 przedstawia dane historyczne i prognozy wzrostu wartości rynku dystrybucji cyfrowej aplikacji mobilnych – jak widać na rok 2017 przewiduje się 2-krotny wzrost rynku w stosunku do roku 2015, a w dodatku obserwowana i przewidywana jest dodatnia dynamika wzrostu.

Rysunek 35. Wzrost rynku aplikacji mobilnych w milionach USD (źródło: Gartner 2014)

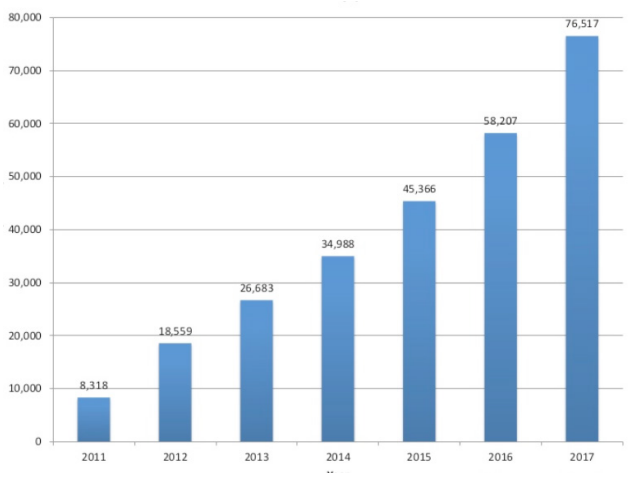
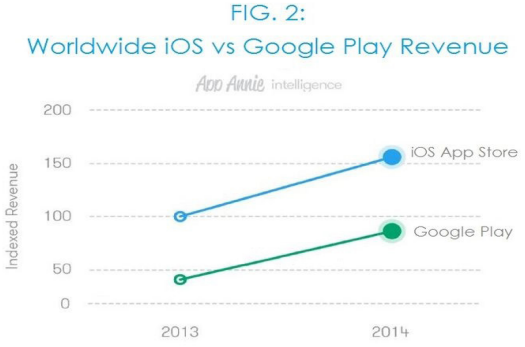



Tabela 2 przedstawia udział poszczególnych platform w rynku urządzeń mobilnych. Udział w rynku nie przekłada się proporcjonalnie do udziału w sprzedaży aplikacji mobilnych. Rys. 36 pokazuje, że platforma iOS jest dominująca w zakresie sprzedaży, mimo znacznie mniejszego udziału w rynku.

Tabela 2. Udział platform w rynku urządzeń mobilnych (źródło: IDC 2015)

| Okres | Android | iOS | Windows Phone | BlackBerry OS | Pozostali |
|---|---|---|---|---|---|
| 2015Q2 | 82.8% | 13.9% | 2.6% | 0.3% | 0.4% |
| 2014Q2 | 84.8% | 11.6% | 2.5% | 0.5% | 0.7% |
| 2013Q2 | 79.8% | 12.9% | 3.4% | 2.8% | 1.2% |
| 2012Q2 | 69.3% | 16.6% | 3.1% | 4.9% | 6.1% |

Rysunek 36. Udział względny sprzedaży na rynku aplikacji mobilnych (indeksowany względem iOS 2013) (Źródło AppAnnie 2015)



Jest wiele przyczyn takiego stanu rzeczy. Jak już wspomniano wcześniej, na platformie iOS znacznie większy procent użytkowników posiada kartę płatniczą powiązaną ze swoim kontem. To zmniejsza barierę zakupu. Należy również podkreślić, że wiele urządzeń opartych o platformę Android, mimo że oferowanych jako smartfony, nie jest nigdy podłączanych do Internetu, a więc nie ma dostępu do nowych aplikacji. Wynika to z faktu oferowania ich na tzw. rynkach wschodzących oraz w najtańszych taryfach, nie oferujących pakietów danych lub oferujących minimalne pakiety danych. Brak dostępu do nowych aplikacji oznacza oczywiście brak możliwości generowania sprzedaży z aplikacji mobilnych, a więc de facto nie stanowią one rynku w rozumieniu twórcy aplikacji mobilnych.

## 6.5. Kanały dystrybucji aplikacji mobilnych

Dystrybucja aplikacji mobilnych wykracza poza samo umieszczenie aplikacji na platformie dystrybucji. Poniżej omówione zostaną najważniejsze kanały dystrybucji aplikacji mobilnych. Producent czy też wydawca aplikacji powinien zadbać o to, by objąć swoimi działaniami możliwie dużo z tych kanałów, tak, by zmaksymalizować zasięg swoich produktów.

### 6.5.1. Sklepy z aplikacjami

Podstawowym kanałem dystrybucji jest oczywiście umieszczenie aplikacji w cyfrowym sklepie. Dla platformy iOS jest to AppStore, dla platformy Android jest to przede wszystkim Google Play Store. Firma Apple nie pozwala na istnienie legalnych, niezależnych sklepów dystrybuujących aplikacje na swoje platformy mobilne, jednak dla Androida istnieją także inne, niezależne i legalnie działające cyfrowe sklepy. Do najpopularniejszych należą m.in. Amazon Appstore czy też Samsung Galaxy Apps. Umieszczenie aplikacji w sklepie polega m.in. na przygotowaniu:

- Obrazu aplikacji do dystrybucji;
- Materiałów graficznych, takich jak ikona, zrzuty ekranu, filmy;
- Materiałów tekstowych w postaci opisu aplikacji;
- Listy słów kluczowych;
- Kategorii wiekowej aplikacji;
- Polityki prywatności;
- Innych materiałów, wedle wymogów właściciela platformy.

### 6.5.2. Preinstalacje

Preinstalacjami określamy kanały dystrybucji, w których na bazie umowy z producentem lub dystrybutorem urządzenia mobilnego, aplikacja jest instalowana na urządzeniu jeszcze przed jego sprzedażą, a więc użytkownik ma do niej dostęp bez żadnych dodatkowych działań. Preinstalacje są bardzo ciekawym sposobem dystrybucji aplikacji, lecz w ostatnim czasie tracą one na znaczeniu.

### 6.5.3. Optymalizacja wyszukiwania i strony aplikacji

Optymalizacja wyszukiwania polega na wykorzystaniu rezultatów wyszukiwania jako kanału dystrybucji. Dokonujemy jej przez optymalizację słów kluczowych, czy też umieszczanie odnośników do aplikacji na dostępnych producentowi lub wydawcy stronach internetowych. Optymalizacja wyszukiwania ma na celu zwiększenie liczby użytkowników, którzy pobierają aplikację w efekcie przeszukiwania zasobów internetowych. Optymalizacja strony aplikacji polega na takim dobraniu elementów widocznych dla użytkownika na stronie aplikacji (ikona, opis, zrzuty ekranu itd.), aby w jak największym stopniu zachęcić do pobrania aplikacji.

### 6.5.4. Prasa i portale internetowe

Kolejnym kanałem dystrybucji jest prasa i portale internetowe, w formie stron, blogów, stron typu fan-page, kanałów filmowych i innych. W większości przypadków dystrybucja w tej formie odbywa się przy wykorzystaniu agencji PR. Umożliwia to poszerzenie liczby użytkowników, którzy mają świadomość istnienia aplikacji. Użycie kanałów internetowych zamiast klasycznych jest tu o tyle istotne, że umożliwia bezpośrednie przekierowanie użytkownika do sklepu.

### 6.5.5. Nabywanie użytkowników

Nabywanie użytkowników jest formą reklamy, w której płaci się za efekt w postaci pozyskania nowego użytkownika aplikacji. Są one oparte przede wszystkim o wyświetlanie reklam aplikacji w obrębie innych aplikacji mobilnych, tak aby maksymalnie skrócić tzw. „*call to action*", a więc liczbę kroków, która dzieli użytkownika między obejrzeniem reklamy, a instalacją aplikacji.

## 6.6. Sklepy z aplikacjami

Największe sklepy z aplikacjami mobilnymi należą do właścicieli poszczególnych platform mobilnych. W przypadku systemu iOS i firmy Apple jest to *AppStore*, a dla platformy Android i firmy Google jest to *Google Play Store*. W praktycznie wszystkich przypadkach zasady dystrybucji są bardzo podobne.

Po pierwsze, należy zostać certyfikowanym deweloperem danej platformy i wyrazić akceptację zasad oraz warunków działania. W większości sklepów stroną może zostać osoba fizyczna lub firma. Ze względu na bezpieczeństwo prawne, związane np. z odpowiedzialnością za naruszenie dóbr stron trzecich, warto w kontekście dystrybucji cyfrowej zastanowić się nad przyjęciem formy prawnej co najmniej w postaci spółki z ograniczoną odpowiedzialnością. Należy albowiem pamiętać, że dystrybuując aplikację na całym świecie, trafimy na rynki takie, jak np. rynek amerykański, na których pozwy o naruszenie dóbr intelektualnych zdarzają się na porządku dziennym.

Utrzymywanie licencji deweloperskiej może wiązać się z cyklicznymi opłatami. W przypadku np. programu deweloperskiego firmy Apple, w darmowym programie nie ma możliwości testowania aplikacji na urządzeniach ani dystry-

bucji aplikacji. Dopiero przystąpienie do programu płatnego umożliwia testowanie i dystrybucję.

Samo umieszczenie aplikacji w sklepie nie podlega opłatom. Zamiast tego, każda z platform pobiera opłatę prowizyjną od sprzedaży – typowo jest to poziom ok. 30% od przychodu ze sprzedaży. O ile marża ta może wydawać się wysoka, należy pamiętać że w przypadku dystrybucji w fizycznych kanałach dystrybucji suma marż poszczególnych pośredników oraz innych opłat, często przekraczała 50%. W przypadku kanałów SMSowych było to również zazwyczaj ok. 50%.

W obecnych czasach w sklepach dystrybucji cyfrowej każdego dnia pojawiają się tysiące(!) nowych aplikacji. W 2015 w Apple AppStore rejestrowano ponad 1000 nowych aplikacji każdego dnia. Oznacza to, że każda z aplikacji musi mierzyć się z ogromną konkurencją. Aby znacząco zwiększyć szanse danej aplikacji, należy postarać się o otrzymanie promocji ze strony właściciela danej platformy. Promocje te nie są płatne – nie da się ich po prostu kupić. Ich uzyskanie wymaga posiadania odpowiednio dobrego produktu – wówczas budując relacje bezpośrednie z właścicielem platformy można przesłać aplikację do wglądu osobom decydującym za promocje na danym sklepie i w efekcie uzyskać promowanie produktu. W niektórych przypadkach, przy osiągnięciu odpowiedniej dynamiki organicznego wzrostu użytkowników aplikacji, retencji oraz poziomu przychodów, można uzyskać promocję aplikacji bez uprzedniego budowania relacji i kontaktu z właścicielami platformy.

## 6.7. Preinstalacje

Preinstalacje, jak wspomniano wcześniej, polegają na zainstalowaniu aplikacji na urządzeniu docelowym przed jego trafieniem do sprzedaży. Był to do niedawna bardzo popularny sposób dystrybucji aplikacji. Niestety, wskutek jego nadużywania przez instalowanie dużej liczby aplikacji, o niskiej wartości dla użytkownika końcowego, w ostatnich latach ten kanał dystrybucji stracił na znaczeniu. Pojawiło się nawet specjalne określenie „*bloatware*" oznaczające oprogramowanie, które jest preinstalowane na urządzeniu i w efekcie powoduje „spuchnięcie" zawartości urządzenia, bez korzyści dla użytkownika.

Wielu użytkowników stara się usunąć wszystkie preinstalowane aplikacje tak, by uzyskać jak najwięcej wolnego miejsca na własne dane i aplikacje. W przypadku platformy Android, wielu użytkowników pożąda zainstalowania na swoim systemie tzw. „stock OS", czyli systemu operacyjnego w wersji „gołej", pozbawionej dodatkowych aplikacji, ale także dodatków producenta. Zapewnia to maksymalną wydajność oprogramowania i zwiększa dostępną dla użytkownika przestrzeń danych.

Mimo to, preinstalacje stanowią ciągle istotny kanał dystrybucji. Ich popularność związana jest obecnie głównie z platformą Android (iOS nie umożliwia preinstalacji). Preinstalację można uzyskać przez nawiązanie odpowiedniej relacji biznesowej z twórcą danej platformy sprzętowej lub firmą telekomunikacyjną. W typowym przypadku w zamian za preinstalację trzeba będzie zrezygnować z opłaty wstępnej za aplikację, a często dodatkowo dać użyt-

kownikowi darmową zawartość wewnątrz aplikacji. Okazją do monetyzacji staną się w tej sytuacji dodatkowe zakupy typu *in-app purchase* lub też wyświetlane w aplikacji reklamy.

Nieco nowszą formą preinstalacji są wbudowane w system kanały marketingowe, np. takie jak wyświetlane na stałe bannery z ofertami aplikacji. Stanowią one alternatywę, w której aplikacja nie jest de facto preinstalowana, lecz użytkownikowi sugerowana jest jej instalacja. Metodę tę stosuje i promuje m.in. Samsung w ramach swojego programu „Samsung Galaxy Gifts". W programie tym użytkownicy nagradzani są dodatkową zawartością w zamian za pobranie i zainstalowanie aplikacji.

## 6.8. Optymalizacja wyszukiwania

Optymalizację wyszukiwania możemy podzielić na dwa obszary: optymalizacja wyszukiwania wewnątrz sklepu z aplikacjami oraz optymalizacja wyszukiwania poza sklepem z aplikacjami.

### 6.8.1. Optymalizacja wyszukiwania w sklepie z aplikacjami

Polega ona na takim doborze parametrów aplikacji, by zmaksymalizować częstotliwość pojawiania się aplikacji w wynikach wyszukiwania oraz zmaksymalizować liczbę pobrań per liczba wyświetleń.

W większości sklepów z aplikacjami użytkownik w pierwszym kontakcie z aplikacją widzi następujące elementy: ikonę, nazwę, cenę i kategorię aplikacji. Każdy z tych elementów powinien być przedmiotem optymalizacji.

Rysunek 37. Typowy widok aplikacji w sklepie dystrybucji cyfrowej: widoczna ikona, nazwa, cena i kategoria aplikacji. (Źródło: Apple AppStore)


W przypadku ikony, należy dążyć do wykonania jej w sposób prosty, czytelny, oddający ducha aplikacji. W przypadku gry często będzie to głowa głównego bohatera, w przypadku aplikacji symboliczne oddanie głównej funkcjonalności lub elementu logotypu aplikacji. Poniżej przedstawiono kilka zalecanych zasad jeżeli chodzi o tworzenie ikony:

– Zastosowanie unikalnego kształtu widocznego na ikonie. Pozwala na łatwe i szybkie rozpoznanie ikony.

– Ograniczenie palety barw. Zapewni harmonię odbioru ikony.
– Unikanie zdjęć i fotorealizmu. Wprowadzają one zbyt wiele detalu, który nie będzie dobrze prezentował się na niewielkiej przestrzeni.
– Unikanie dużej ilości tekstu – nie będzie on czytelny
– Sprawdzenie jak prezentuje się ikona na tle różnych tapet oraz w sąsiedztwie ikon innych aplikacji.

Nazwa aplikacji powinna opisywać jej funkcjonalność, ale także być chwytliwa i łatwa w zapamiętaniu. Cena jest kluczowym elementem decyzyjnym, więc również musi zostać dostosowana do oferowanej wartości, typu aplikacji i cen konkurencyjnych. Poniżej przedstawiono kilka zasad doboru nazwy:

– Krótka (tak, by zmieściła się w całości w maksymalnie wielu kontekstach).
– Przejrzystość nazwy – powinna mówić o czym jest aplikacja.
– Łatwość do wymówienia i zapamiętania (szczególnie na rynku docelowym, np. amerykańskim)
– Połączenie 2 krótkich słów w 1 może być dobrym pomysłem (np. Evernote, Wunderlist).

Dalszym elementem optymalizacji jest dobór słów kluczowych, opisu aplikacji, zrzutów ekranu i filmu z aplikacji. Większość sklepów pozwala na umieszczenie wszystkich wyżej wymienionych elementów, ale należy pamiętać o ich ograniczeniach. Przykładowo Apple AppStore nie indeksuje treści opisu, a w słowach kluczowych nie wolno posługiwać się nazwami zarejestrowanymi firm trzecich.

Bardzo istotnym elementem optymalizacji jest lokalizacja zasobów tekstowych i graficznych na dane rynki. W szczególności obejmuje to również lokalizację słów kluczowych i napisów na zrzutach ekranu.

### 6.8.2. Optymalizacja wyszukiwania poza sklepami z aplikacjami

W ramach optymalizacji wyszukiwania poza sklepami należy oczywiście zastosować wszystkie techniki optymalizacji wyszukiwania stron internetowych. Warto również zaangażować wszystkie dostępne kanały, tak, by zmaksymalizować liczbę odnośników do aplikacji.

Wiele sklepów dystrybucji cyfrowej lub narzędzi analitycznych pozwala śledzić źródła instalacji oraz efektywność kliknięć, co pozwala na mierzenie efektywności działań w zakresie optymalizacji wyszukiwania.

## 6.9. Prasa i portale internetowe

Budowanie swojej pozycji w prasie i na portalach internetowych rzadko przekuwa się bezpośrednio i natychmiast na pobrania aplikacji, ale służy budowaniu świadomości marki, a w efekcie ma wpływ na pojawienie się produktu w kanałach socjalnych i ma wpływ na wirusowy wzrost zainteresowania aplikacją.

Jednym ze sposobów realizacji jest korzystanie z pośrednictwa agencji PR. Agencja taka jest w stanie zapewnić kontakt z szeroką liczbą mediów, ale

zazwyczaj nie jest w stanie zagwarantować rezultatów w postaci określonej liczby publikacji. W niektórych przypadkach można oczywiście opłacić sponsorowane artykuły, ale znacząco podnosi to koszt nakładów marketingowych.

Interesującą formą istnienia w prasie i portalach stanowi w ostatnich latach formuła wideorecenzji. Jest to obecnie oczywiście płatna forma marketingu, ale cechująca się dużą dynamiką wzrostu zasięgu oraz stosunkowo wysoką efektywnością.

## 6.10. Nabywanie użytkowników

Nabywanie użytkowników jest formą marketingu i dystrybucji, która narodziła się wraz z pojawieniem się aplikacji typu „freemium", tj. dostępnych do pobrania za darmo i oferujących zakupy wewnątrz aplikacji.

Przykładami platform oferujących nabywanie użytkowników są Facebook, iAd, AdMob, Chartboost, TapJoy, AdColony, Applovin i wiele innych. Oferują one możliwość nabywania instalacji aplikacji za określoną cenę (zależną od popytu i podaży na rynku w danej chwili, jest to zazwyczaj cena oscylująca w granicy jednego do kilku dolarów za użytkownika). Platformy te wyświetlają reklamy naszej aplikacji w innych aplikacjach, monitorują następnie kliknięcia i instalacje i w efekcie pobierają opłatę zależną od liczby wyświetleń, albo opartą o skuteczne zainstalowanie i uruchomienie przez użytkownika aplikacji. W pierwszym przypadku mówimy o kampaniach typu CPM (Cost Per Mille – koszt liczony od każdego tysiąca wyświetleń), w drugim przypadku o kampaniach typu CPI (Cost Per Install).

O ile kampanie te są najskuteczniejszym sposobem pozyskania użytkownika aplikacji, o tyle są one opłacalne jedynie w takiej sytuacji, w której przychód z każdego pozyskanego użytkownika jest wyższy od kosztu nabycia użytkownika:

$$\overline{CPI} > ARPU$$

Gdzie:

$\overline{CPI}$ – średni koszt instalacji

$ARPU$ – (Average Revenue Per User) średni przychód wygenerowany z każdego użytkownika aplikacji.

## 6.11. Podsumowanie

Na podstawie aktualnych trendów rynkowych należy spodziewać się, iż w najbliższych latach aplikacje mobilne będą w dalszym ciągle zyskiwać na znaczeniu. Dystrybucja i marketing aplikacji mobilnych to złożone zagadnienia. Ich zakres obejmuje pierwsze dwa etapy lejka aplikacji mobilnej, a więc stanowi bezpośrednie przełożenie na wzrost i przychód generowany przez aplikację.

Kluczowe jest zrozumienie przez deweloperów i wydawców znaczenia dobrej dystrybucji i marketingu aplikacji oraz dołożenie niezbędnych starań mających na celu zmaksymalizowanie szans produktu na dobre zaistnienie na rynku. W obecnych czasach nie można zakładać, że korzystając jedynie

z jednego kanału dystrybucji oraz nie czyniąc kroków w zakresie marketingu aplikacji można liczyć na jakikolwiek sukces. Wiele firm i osób poniosło ogromne nakłady celem wytworzenia oprogramowania mobilnego tylko po to, by uzyskać sprzedaż na poziomie kilkunastu lub kilkudziesięciu sztuk w ciągu miesiąca, co nie pozwoliło na pokrycie nawet niewielkiej części kosztów wytworzenia.

Należy jednocześnie pamiętać, że mamy do czynienia z rynkiem dynamicznego rozwoju i nowych technologii. Dlatego należy na bieżąco śledzić trendy i zmiany, gdyż jest to jedyny sposób na zachowanie należytej konkurencyjności oraz stosowanie metod adekwatnych do aktualnej sytuacji rynkowej.

# 7
# LITERATURA


Buchwald P., *Obiektowe mechanizmy przechowywania informacji dostarczanych przez systemy nadrzędnej kontroli dystrybucji danych jako alternatywa dla rozwiązań opartych o relacyjne bazy danych*, XII Konferencja Systemów Czasu Rzeczywistego (SCR 2005), Ustroń 2005.

Buchwald P., Rostański M., Jurasz A., *Comparative Analysis of Database Solutions in Applications with the Android Operating System*, [in:] Pikiewicz P., Rostanski M. (eds.), *Internet in the information Society. Computer Systems Architecture and Security*, Academy of Business in Dabrowa Gornicza 2013.

Buchwald P., Rostański M., Jurasz A., *Relative and non-relative databases performance with an Android platform application*, [in:] Theoretical and Applied Informatics 2/2013.

Buchwald P., *Mechanizmy integracji danych i logiki biznesowej w kooperacyjnych systemach sterowania procesami przedsiębiorstwa,* [w:] Knosala R. (red.), *Komputerowo Zintegrowane Zarządzanie*, Wydawnictwo PTZP, Opole 2011.

Buchwald P., Rostański M., Mączka K., *Virtual reality and mobile devices in 3d objects designing and prototyping*, [w:] Knosala R. (red.), *Innowacje w zarządzaniu i inżynierii produkcji*. Tom II, Oficyna Wydawnicza Polskiego Towarzystwa Zarządzania Produkcją, Opole 2015.

Buchwald P., *Zastosowanie wybranych mechanizmów transformacji danych na potrzeby symulacyjne na przykładzie SQL Server,* [w:] Studia Informatica Vol. 31, 2010.

Burdea G., Coiffet P., *Virtual Reality Technology* Published by John Wiley Hinman R., *The Mobile Frontier: A Guide for Designing Mobile Experiences*, 2012.

Cave K., '*Computer game therapy': A treatment for depression*, IDG Connect [dostęp: 24.05.2015].

Chuck M., *The Third Screen: Marketing to Your Customers in a World Gone Mobile.* Nicholas Brealey Publishing, 2011.

*Dokumentacja techniczna bazy danych Neo4j,* http://neo4j.com/docs [dostęp: 27.10.2015].

*Dokumentacja techniczna bazy danych IBM DB2 Everywhere,* http://www-01.ibm.com/support/docview.wss?uid=swg27009474 [dostęp: 01.09.2015].

Gajewski M., *Leap Motion – miał być Kinect-killerem, a jest rynkową klęską*" http://www.spidersweb.pl/2014/03/leap-motion.html

Genadinik A., *Mobile App Marketing And Monetization.* CreateSpace Independent Publishing Platform, 2015.

Harężlak K., *Data replication on mobile devices,* [in:] Studia Informatica 2009 Nr 2A(83), Wydawnictwo Politechniki Śląskiej, Gliwice 2009.

Holiday R., *Growth Hacker Marketing.* Profile Books, 2014.

*Inertial Navigation: 40 Years of Evolution – Overview* at http://www.imar-navigation.de

Kawasaki G., *The Art of Social Media: Power Tips for Power Users*. Portfolio Penguin, 2014.

Lang B., *Oculus Open-sources Rift DK1: Mechanical Designs, Firmware, and More All Freely Available*, http://www.roadtovr.com/oculus-rift-dk1-open-source-manufacturing-the-dk1-nirav-patel/

Lin W., Veeravalli B., *Object Management in Distributed Database Systems for Stationary and Mobile Computing Environments. A Competitive Aproach*; Springer Science + Business Media, Springer US, New York 2003.

Mielczarek J., *Głosowe interfejsy użytkownika*, blog webusability.pl, 2012.

Prajzner K., *Tekst jako świat i gra. Modele narracyjności w kulturze współczesnej*, Łódź 2009.

Robinson I., Webber J., Eifrem E., *Graph Databases*, O'Reily 2013.

Rodrigues C., Afonso J., Tomé P., *Mobile Application Webservice Performance Analysis: Restful Services with JSON and XML,* [in:] *Communications in Computer and Information Science,* Vol. 220, 2011.

Sadalage P.J., Fowler M., *NoSQL Kompendium Wiedzy*, Helion, Gliwice 2014.

Sadalage Pramod J., Fowler M., *NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence*, Addison-Wesley Professional 2012.

Scott D.M., *The New Rules of Marketing and PR: How to Use Social Media, Online Video, Mobile Applications, Blogs, News Releases, and Viral Marketing to Reach Buyers Directly.* Wiley 2015.

Shashank T., *Professional NoSQL*, Wrox 2012. Mena J., *Data Mining Mobile Device*, CRC Press, New York 2013.

*Strona projektu Neo4j Mobile for Android* http://neo4j.com/blog/new-incubator-project-neo4j-mobile-for-android-v0-1/ [dostęp: 01.09.2015].

Ufituwe M., *How to Sell Products and Services with Mobile Apps.* Ecommerce Maurice Victor, 2015.

*The effectiveness of SPARX, a computerised self help intervention for adolescents seeking help for depression: randomised controlled non-inferiority trial, doi*: http://dx.doi.org/10.1136/bmj.e2598)

Yeung A., Stephenson P., Pang N., *Oracle 9i Build Wireless, Offline, Push and Voice Application*, Oracle Press, Redwood City 2002.

# 8

# DODATEK A

## 8.1. Przygotowanie do pracy i pierwszy program

### 8.1.1. Instalacja środowiska Java

Jak wspomniano wcześniej – narzędzia niezbędne do tworzenia i uruchamiania programów w języku Java można pobrać ze stron firmy Oracle. Z podanej strony należy pobrać i zainstalować pakiet Java Development Kit (JDK) w wersji dla Java 8:

http://www.oracle.com/technetwork/Java/Javase/downloads/jdk8-downloads-2133151.html

Po zainstalowaniu środowiska należy ustawić odpowiednio zmienne systemowe[49]:

JAVA_HOME

oraz

JDK_HOME

na katalog, w którym zainstalowano JDK, np.:

c:\Program Files\Java\jdk1.7.0_71\

oraz dodać do zmiennej

PATH

Ścieżkę do katalogu, w którym znajdują się pliki wykonywalne JDK, np.:

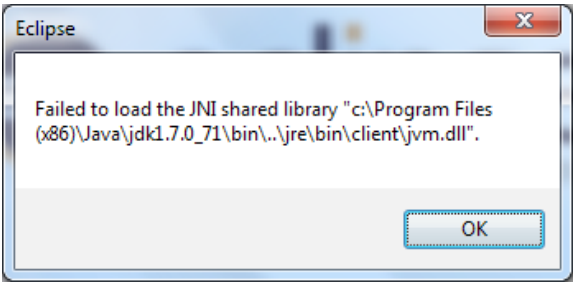
c:\Program Files\Java\jdk1.7.0_71\bin\

[49] Szczegółowy opis jak ustawiać zmienne systemowe dla Windows można znaleźć tutaj: http://www.computerhope.com/issues/ch000549.htm. W Internecie są też dostępne instrukcje dla Linuxa i OSX.

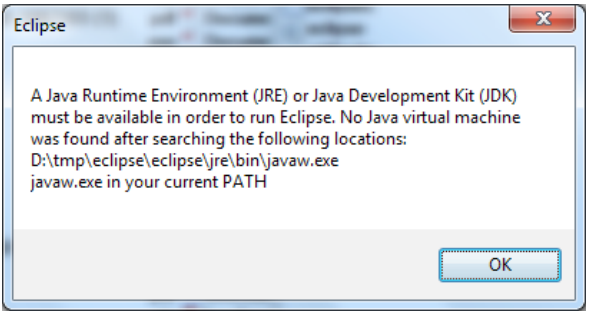
### 8.1.2. Instalacja Eclipse

Kolejnym krokiem będzie pobranie oraz instalacja środowiska projektowego (IDE) Eclipse ze strony http://www.eclipse.org

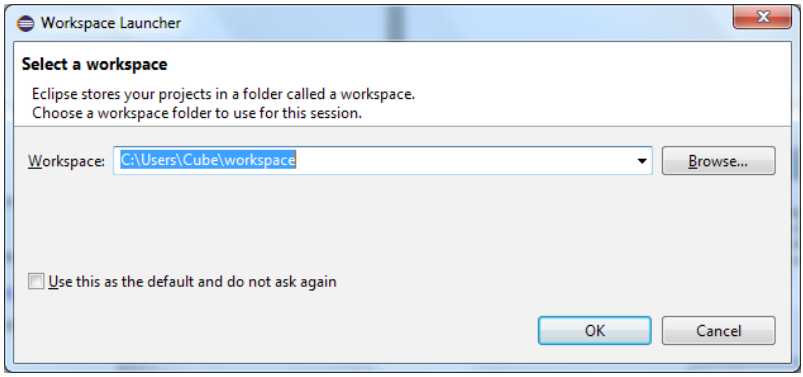
**Uwaga!** Należy zwrócić uwagę na zgodność wersji JDK i Eclipse – sugerowana jest instalacja 64-bitowej wersji obu pakietów (x64). Niezgodność wersji może spowodować następujący błąd:




Również nieprawidłowe ustawienie zmiennych systemowych opisanych wcześniej skutkować będzie komunikatem błędu, np.:




Przy pierwszym uruchomieniu Eclipse zapyta użytkownika o umiejscowienie na dysku katalogu workspace, w którym będą przechowywane tworzone projekty. Warto zapamiętać:

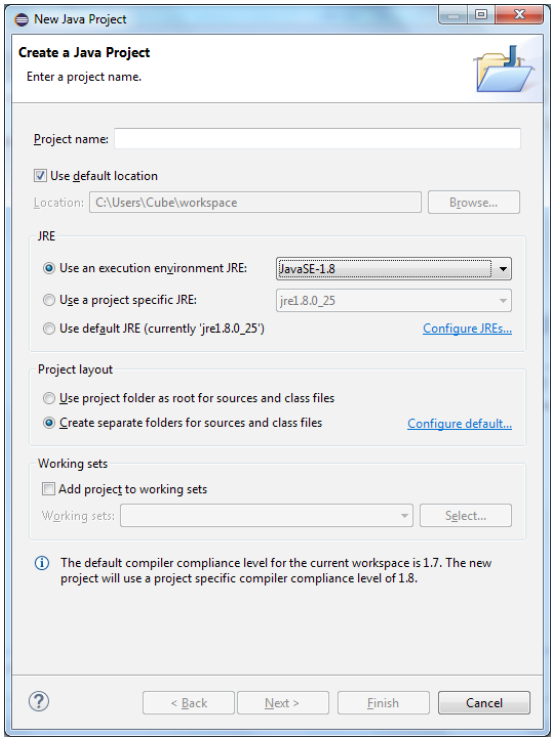



### 8.1.3. Javadoc

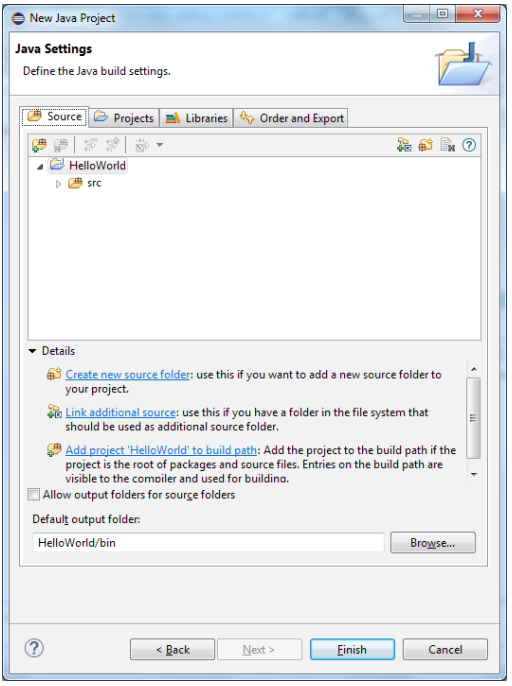
Pracując w języku Java zawsze warto mieć „pod ręką" dokumentację API biblioteki standardowej języka, dostępną w Internecie w formie Javadoc: http://docs.oracle.com/Javase/8/docs/api/

### 8.1.4. Pierwszy program

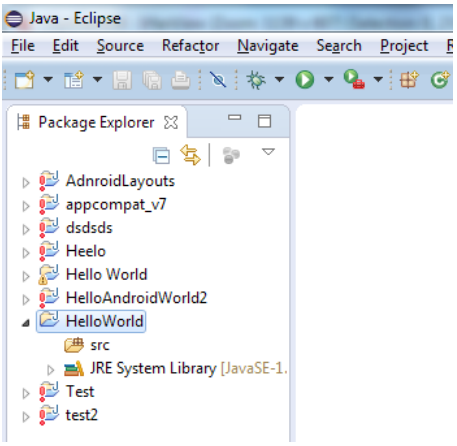
Po uruchomieniu środowiska można utworzyć pierwszy projekt. Należy w tym celu wybrać z menu opcję File -> New –> Java Project



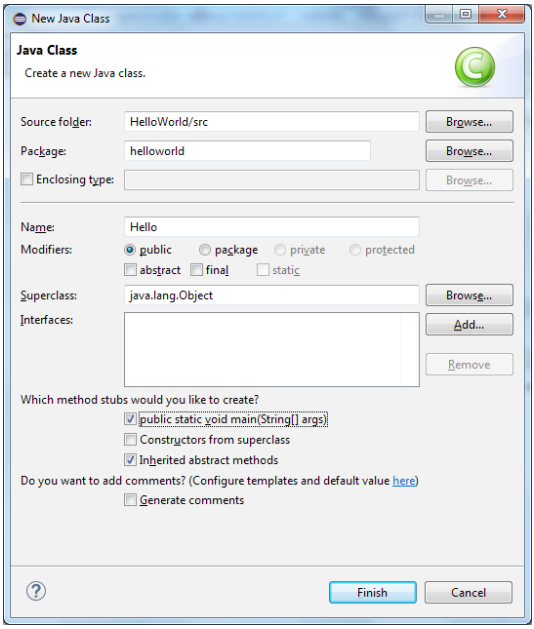
Przy prawidłowym przygotowaniu środowiska pracy powinno pojawić się okno odpowiadające przedstawionemu powyżej. Poza wpisaniem nazwy projektu HelloWorld i kilknięciem Next, nie należy zmieniać ustawionych parametrów. Również w kolejnym oknie należy pozostawić wartości domyślne, warto jednak zapoznać się z dostępnymi ustawieniami przed kliknięciem Finish.




Efektem tego powinno być utworzenie pustego projektu, widocznego w oknie Package Explorer:



Kolejnym krokiem powinno być dodanie do projektu głównej klasy – poprzez kliknięcie prawym przyciskiem na nazwie projektu i wybranie New-> Class



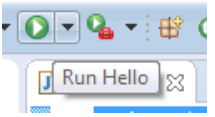

W oknie dialogowym należy podać nazwę pakietu (powyżej: helloworld), nazwę klasy (powyżej: Hello) i zaznaczyć dodanie metody main – podobnie, jak w przypadku języków C/C++ jest to główna, wejściowa metoda programu. W efekcie tego powinna zostać wygenerowana pusta klasa zawierająca wspomnianą metodę.

Zadaniem pierwszego programu będzie wypisanie na ekranie tekstu „Hello World!"[50]

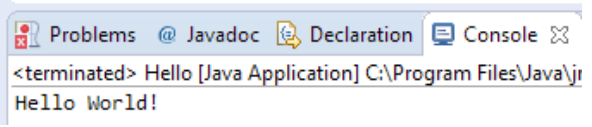
```
package helloworld;

public class Hello {

 public static void main(String[] args) {
       // TODO Auto-generated method stub
       System.out.println("Hello World!");
 }

}
```

[50] Osobom zastanawiającym się, dlaczego większość kursów programowania zaczyna się właśnie od tego programu i tego tekstu należy polecić klasyczną już książkę *Programowanie w ANSI C* autorstwa Briana W. Kernighana i Dennisa M. Ritchiego, w Polsce wydaną przez Helion: http://helion.pl/ksiazki/jezyk-ansi-c-programowanie-wydanie-ii-brian-w-kernighan-dennis-m-ritchie,jansic.htm (W polskiej wersji program wypisuje „*Ahoj przygodo!*")

Program można przetestować, klikając przycisk Run dostępny na pasku narzędzi:




Efekt działania powinien być widoczny w oknie Console w dolnej części przestrzeni roboczej Eclipse:




### 8.1.5. Instalacja SDK Android

Środowisko rozwojowe Androida (Android SDK) wymaga do pracy co najmniej Javy 7[51], obecnie nic nie stoi na przeszkodzie by korzystać z najnowszej wersji w kompilacji 64-bitowej. Tak więc by kontynuować, należy wpierw prawidłowo zainstalować i skonfigurować JDK oraz Eclipse, co opisano powyżej. Zakładając, że te komponenty są już zainstalowane, kolejnym krokiem powinno być pobranie samodzielnej (standalone) wersji SDK dla Androida:

https://developer.android.com/sdk/index.html#Other

Pod tym adresem dostępny jest Android SDK Manager, za pomocą którego można pobrać właściwe elementy SDK, zależne od wersji Androida, którą chcemy się zajmować.

Po jego uruchomieniu należy zaznaczyć elementy:

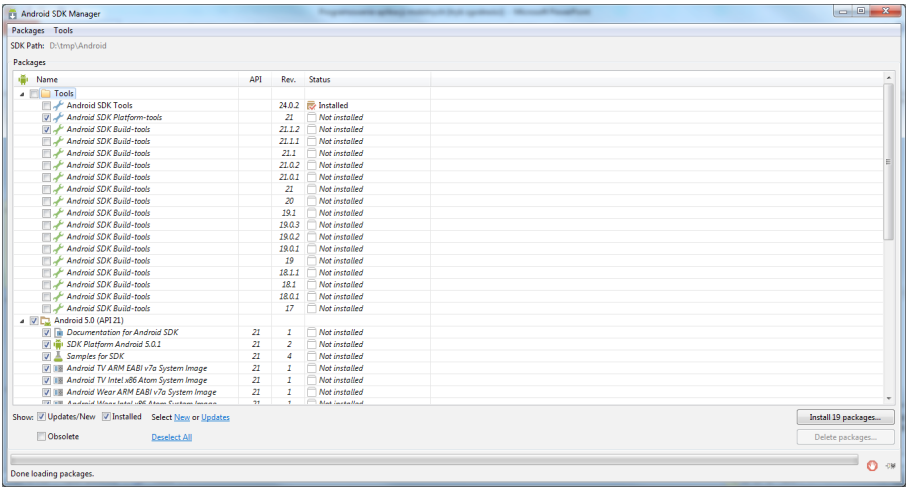
Android SDK Tools

Android SDK Platform-tools

Android SDK Build-tools (najwyższa wersja)

Po uruchomieniu Android SDK Manager zaproponuje instalację odpowiednich narzędzi w najnowszej wersji. Dodatkowo, by zachować zgodność w wcześniejszymi wersjami można zaznaczyć wsparcie dla Androida 4.4.2 (API 19)

[51] Jeszcze do nie dawna jako wymóg podawano Javę 6




### 8.1.6. Wtyczka ADT dla Eclipse

Kolejnym krokiem powinna być instalacja wtyczki ADT dla Eclipse[52] – rozszerzającą możliwości Eclipse o narzędzia związane z tworzeniem aplikacji dla Androida.

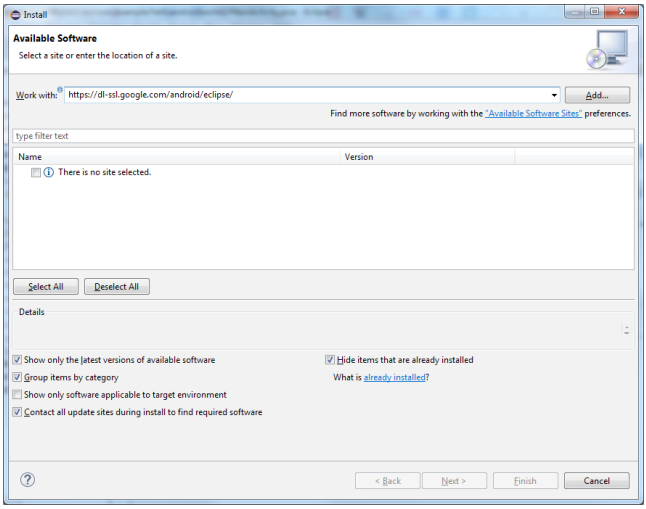
Aby to zrobić należy w samym Eclipse wybrać opcję

Help->Install New Software

A następnie w polu adresu wpisać

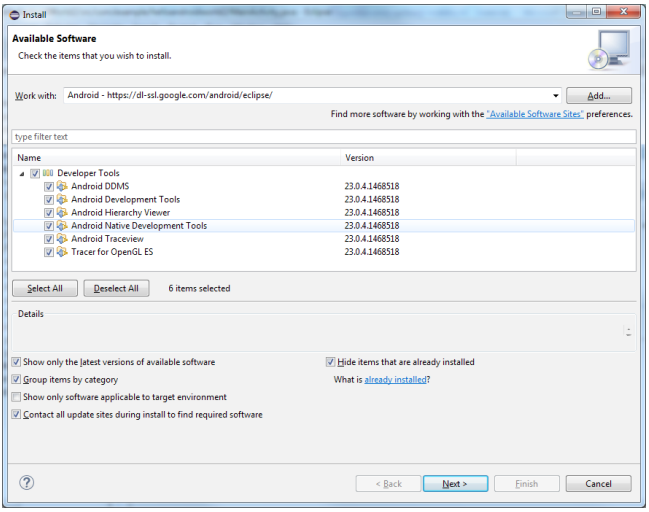
```
https://dl-ssl.google.com/android/eclipse/
```

i kliknąć Add




[52] Szczegółowe instrukcje dostępne są pod adresem: https://developer.android.com/sdk/installing/installing-adt.html

Po tej operacji w oknie poniżej pojawi się pozycja Developer Tools, którą należy zaznaczyć:



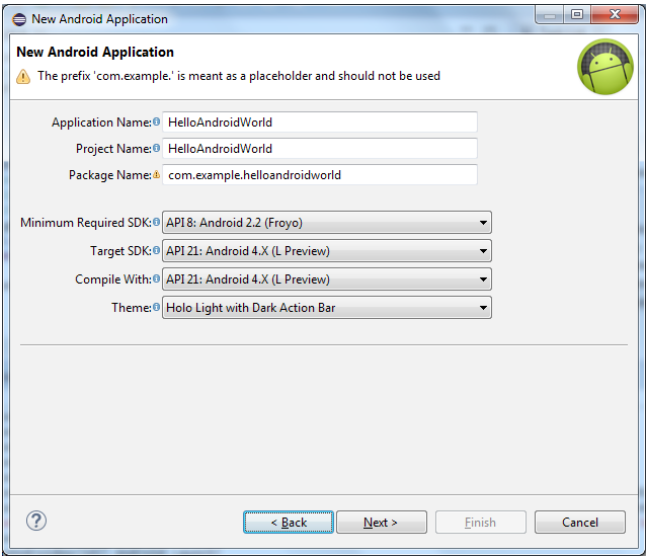

Dalej kolejno klikając Next i akceptując licencję użytkownik uruchomi instalację narzędzi. Po jej zakończeniu konieczny jest restart Eclipse.

## 8.2. Pierwsza aplikacja

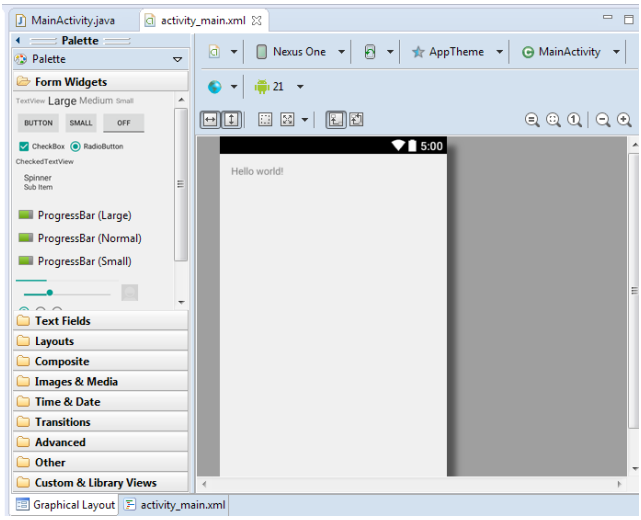
Po prawidłowym zainstalowaniu wszystkich komponentów system powinien być gotowy do stworzenia pierwszej aplikacji dla Androida.

Aby to zrobić, należy wybrać z menu opcję

File -> New ->Other -> Android Application Project

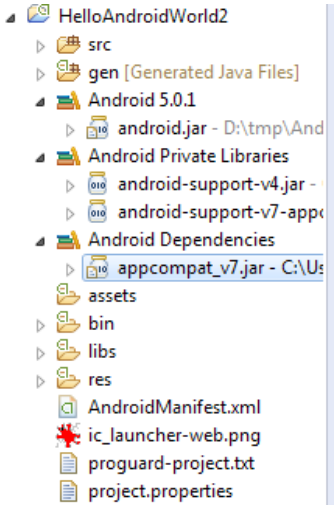


Za wyjątkiem podania nazwy projektu nie należy zmieniać domyślnych ustawień, przeklikując kolejne ekrany przyciskiem Next. Oczywiście warto zapoznać się z dostępnymi opcjami.




### 8.2.1. Struktura katalogów projektu

Rozwijając nowo utworzony projekt w oknie Package Explorer można zaobserwować, że struktura katalogów jest bardziej rozbudowana, niż w przypadku projektu dla Java SE:




/src – pliki źródłowe
/gen – pliki wygenerowane przez SDK

/res – zasoby aplikacji (np. grafiki png)
/assets – zasoby dostępne jako „surowe" dane (my musimy obsłużyć otwarcie)
/Android x.x.x – biblioteka systemowa (odpowiednik JRE system library dla projektu Java SE)
AndroidManifest.xml – metadane dotyczące aplikacji

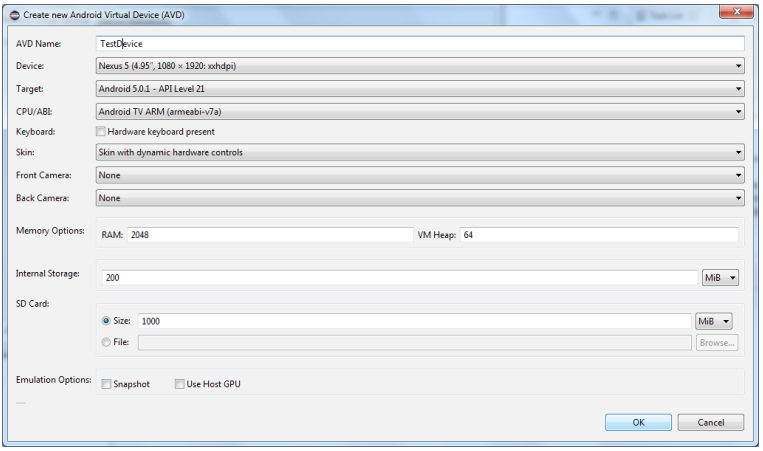
Do projektu zostały też dołączone niezbędne biblioteki systemowe (android.jar, android-support-v4.jar, itd.)[53]

### 8.2.2. Tworzenie wirtualnego urządzenia

Aby uruchomić projekt, konieczne jest skonfigurowanie emulatora (wirtualnego urządzenia), lub podłączenie rzeczywistego, skonfigurowanego w trybie developerskim.

Aby utworzyć nowe urządzenie wirtualne, należy wybrać opcję

Window->Android Virtual Device Manager -> Create




Na ekranie powyżej podano przykładową konfigurację urządzenia.

**Uwaga!** W nazwie urządzenia (AVD Name) nie można używać spacji.

Po prawidłowym utworzeniu nowego urządzenia, opcja Run umożliwi uruchomienie programu na wirtualnym urządzeniu. Robiąc to, należy uzbroić się w cierpliwość.

## 8.3. Uruchamianie na rzeczywistym urządzeniu

Alternatywnie programy uruchamiać można bezpośrednio na rzeczywistym urządzeniu pracującym pod kontrolą systemu Android – należy je jednak wpierw prawidłowo skonfigurować.

[53] Lista dołączonych bibliotek może być inna – zależy od wersji SDK i konfiguracji projektu.

### Uruchamianie trybu developerskiego

W zależności od typu urządzenia i wersji Androida odblokowanie trybu developers. Informacje te można bez trudu znaleźć w sieci, zadając wyszukiwarce pytanie w rodzaju „how to unlock developer mode Samsung galaxy s2”[54]

Zazwyczaj wymaga to odnalezienia nr wersji (Build version lub build number) w ustawieniach telefonu i tapnięciu nań 7 razy (!). Zwykle opcję tę można znaleźć w menu Settings->General->About

W polskiej wersji może to być np.:

Ustawienia->Informacje o telefonie->Numer kompilacji.

Po prawidłowym wykonaniu tej operacji w menu Ustawień powinna pojawić się pozycja Opcje programistyczne. Używając jej należy włączyć opcję „Debugowanie przez USB” (w zależności od wersji może ona nosić inną nazwę, np.: po prostu „Debugowanie”).

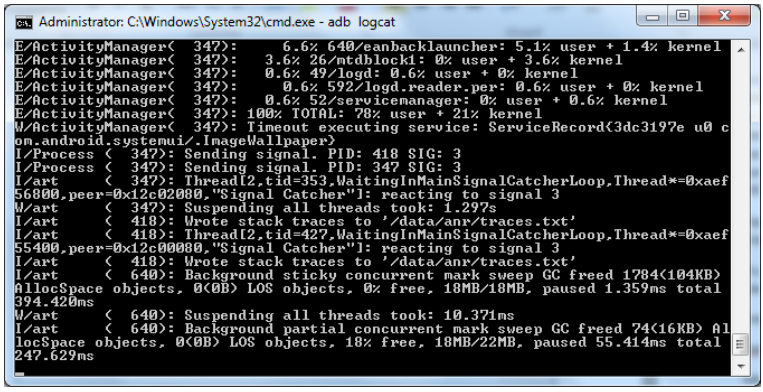
### Instalacja sterowników

Kolejnym krokiem powinna być instalacja dedykowanych dla danego urządzenia sterowników umożliwiających komunikację przez USB. Trzeba przy tym pamiętać, że nie muszą być to te same sterowniki, co te służące np. do kopiowania muzyki. Procedura wyszukiwania wygląda różnie dla różnych urządzeń (tu znów pomocna okaże się wyszukiwarka). W przypadku urządzeń firmy Samsung, zazwyczaj niezbędne pakiety instalowane są automatycznie razem z oprogramowaniem KIES dedykowanym do zarządzania telefonem/tabletem.

Poprawność powyższych operacji można zweryfikować sprawdzając dostępność logów pracy urządzenia poleceniem ADB logcat z Android SDK.

Android\platform-tools\adb logcat

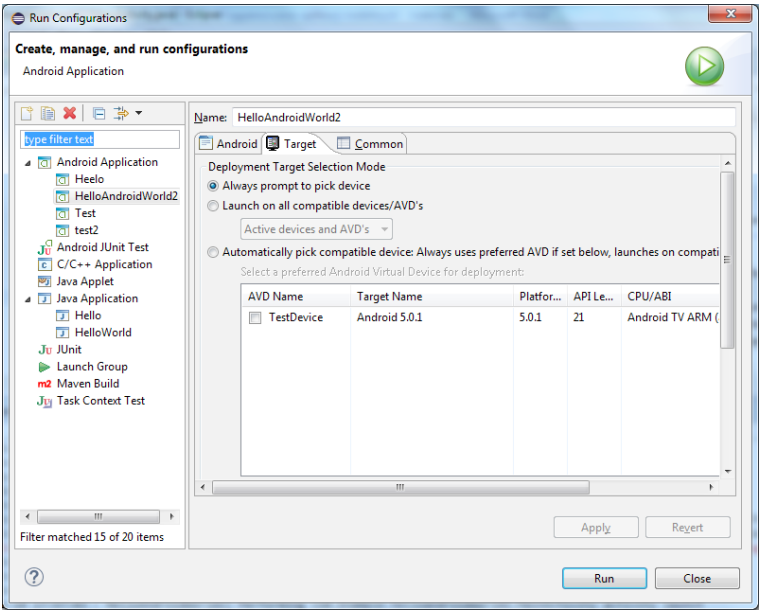
Po podłączeniu urządzenia i wydaniu powyższego polecenia użytkownik powinien zobaczyć ciąg tekstów opisujących aktualnie wykonywane przez urządzenie zadania, np.:




[54] Opis dla kilku popularnych urządzeń i wersji Androida można znaleźć pod tym adresem: http://www.droidviews.com/how-to-enable-developer-optionsusb-debugging-mode-on-devices-with-android-4-2-jelly-bean/

Natomiast komunikat “Waiting for device” oznacza brak połączenia z urządzeniem.

W przypadku obecności więcej niż jednego urządzenia (np.: wirtualnego i rzeczywistego) warto dodatkowo zaznaczyć opcję „Always prompt to pick device” w Run->RunConfigurations->Target




Po prawidłowym wykonaniu opisanych powyżej czynności użytkownik zostanie nagrodzony widokiem pierwszego własnego programu dla systemu Android: