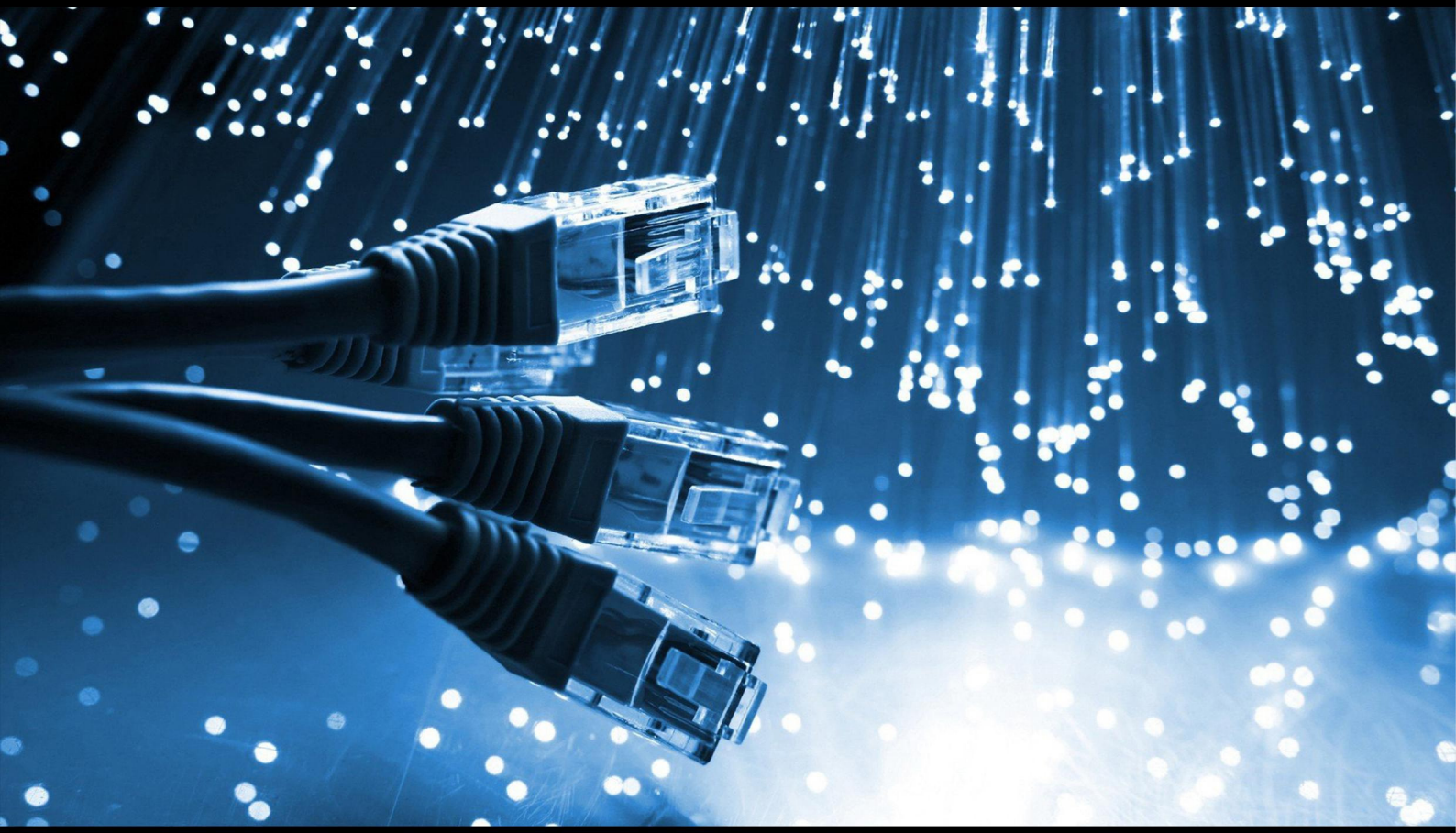
JCSI

Journal of Computer Sciences Institute

Volume 8/2018



Institute of Computer Science
Lublin University of Technology

jcsi.pollub.pl

ISSN: 2544-0764

Redakcja JCSI
e-mail: jcsi@pollub.pl
www: jcsi.pollub.pl
Instytut Informatyki
Wydział Elektrotechniki i Informatyki

Politechnika Lubelska
ul. Nadbystrzycka 36 b
20-618 Lublin

Redaktor naczelny:
Tomasz Zientarski
e-mail: t.zientarski@pollub.pl

Redaktor techniczny:
Beata Pańczyk,
e-mail: b.panczyk@pollub.pl

Recenzenci numeru:
dr inż. Grzegorz Kozieł
dr inż. Marek Miłosz
dr inż. Dariusz Gutek
dr inż. Jakub Smołka
dr inż. Małgorzata Plechawska-Wójcik
dr Edyta Łukasik
dr inż. Elżbieta Miłosz
dr inż. Piotr Kopniak
dr Mariusz Dzieńkowski
dr inż. Maria Skublewska-Paszkowska

Skład komputerowy:
Piotr Misztal
e-mail: p.misztal@pollub.pl

Projekt okładki:
Marta Zbańska

JCSI Editorial
e-mail: jcsi@pollub.pl
www: jcsi.pollub.pl
Institute of Computer Science
Faculty of Electrical Engineering and Computer Science
Lublin University of Technology
ul. Nadbystrzycka 36 b
20-618 Lublin, Poland

Editor in Chief:
Tomasz Zientarski
e-mail: t.zientarski@pollub.pl

Assistant editor:
Beata Pańczyk,
e-mail: b.panczyk@pollub.pl

Reviewers:
Grzegorz Kozieł
Marek Miłosz
Dariusz Gutek
Jakub Smołka
Małgorzata Plechawska-Wójcik
Edyta Łukasik
Elżbieta Miłosz
Piotr Kopniak
Mariusz Dzieńkowski
Maria Skublewska-Paszkowska

Computer typesetting:
Piotr Misztal
e-mail: p.misztal@pollub.pl

Cover design:
Marta Zbańska

ISSN 2544-0764

Spis treści

Contents

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 8 (2018) 215-219
Wysłane: 2017-12-14
Przyjęte: 2017-12-19

Rozpoznawanie obiektów w obrazie wideo z kamery do komputera

Oleksandr Cherednyk*, Elżbieta Miłosz

Politechnika Lubelska, Instytut Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie: Celem artykułu jest określenie skuteczności wykrywania obiektu w obrazie wideo za pomocą kamery internetowej. W trakcie pracy zostały zbadane i opisane podstawowe metody rozpoznawania obiektów na zdjęciu, a mianowicie wykorzystanie sztucznych sieci neuronowych i metod Viola-Jones. Do przeprowadzenia eksperymentów, na podstawie metody Viola-Jones, realizowano aplikację do rozpoznawania obiektów na wideo. Za pomocą tej aplikacji, przeprowadzono badanie w celu określenia skuteczności metody Viola-Jones do wykrywania obiektów na wideo.

Słowa kluczowe: Viola-Jones; gest; rozpoznawanie

*Autor do korespondencji.

Adres e-mail: oleksandr.cherendyk@gmail.com

Object recognition on video from camera to computer

Oleksandr Cherednyk*, Elżbieta Miłosz

Institute of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract. The goal is to determine the effectiveness of object detection in a video using the camera for the computer. In the course of work studied and described the main methods of recognition of objects in the image, namely the use of artificial neural networks and techniques of Viola-Jones. For the study, based on the method of Viola-Jones, implemented the application for object recognition in video, as this method is effective for solving this problem. With this application, a study was conducted to determine the effectiveness of the method of viola-Jones to detect objects in the video.

Keywords: Viola-Jones; gesture; recognition

*Corresponding author.

E-mail address: oleksandr.cherendyk@gmail.com

1. Wprowadzenie

Ludzie nieustannie borykają się z problemem rozpoznawania obrazów. Ludzki mózg radzi sobie z tym zadaniem bez problemu. Ale istnieją sytuacje, kiedy człowiek potrzebuje pomocy komputera, aby rozpoznawać obiekty. W dzisiejszych czasach komputer jest często używany do rozpoznawania obiektów. Technologia ta jest na przykład wykorzystywana w ruchu drogowym, w celu wyszukiwania numerów rejestracyjnych samochodów, na lotniskach dla wyszukiwania twarzy przestępców, a także do rozpoznawania gestów, pozwalających na realizację interakcji człowieka z komputerem. Pomimo tego, że technologia i narzędzia rozpoznawania obrazów osiągnęły dobry poziom, to nie są one doskonałe i wymagają dalszych badań.

2. Metody rozpoznawania obrazów

Wideo jest to sekwencja obrazów. Dlatego, aby rozpoznać obiekt na wideo, należy rozpoznać go w jednym obrazie. Jednymi z głównych sposobów rozpoznawania obiektu na obrazie są:

- zastosowanie metody Viola-Jones;
- wykorzystanie sztucznych sieci neuronowych.

W 2001 roku Paul Viola i Michael Jones zaproponowali algorytm pozwalający wykrywać obiekty na obrazie w czasie rzeczywistym (metoda Viola-Jones) [2][3]. Metoda pozwala wykrywać różne typy obiektów. Istnieją gotowe realizacje i ulepszenia tej metody, jedna z nich jest dostępna w zawartości biblioteki OpenCV [1]. Zalety metody Viola-Jonsa to przede wszystkim wysoka dokładność rozpoznawania obrazu i możliwość wyszukiwania obiektów na obrazie w czasie rzeczywistym. W warunkach pracy pod niewielkim kątem (do 30 stopni), metoda Viola-Jones ma wysoką szybkość rozpoznawania, a także skutecznie działa w różnych warunkach oświetlenia [4].

Praca algorytmu metody Viola-Jones opiera się na czterech koncepcjach [5]:

1. Reprezentacja obrazu w integralnej postaci, co pozwala szybko obliczyć niezbędne cechy obiektów.
2. Wyszukiwanie żądanego obiektu na podstawie prostokątnych kształtów zwanych falkami (cechami) Haara.
3. Zastosowanie algorytmu "boosting", co pozwala wybrać bardziej odpowiednie cechy obiektu w określonej części obrazu.
4. Korzystanie z kaskadowego filtrowania okien, gdzie nie znaleziono obiektu.

Zintegrowany obraz jest przedstawiony w postaci dwuwymiarowej macierzy, której rozmiar jest równy rozmiarowi przychodzącego obrazu. Obraz cyfrowy przechowuje w sobie wartość koloru piksela, dla obrazu czarno-białego, wartość od 0 do 255, a dla kolorowego wynosi od 0 do 〖255〗^3(wartości R, G, B). W integralnej postaci każdy element macierzy przechowuje w sobie sumę natężenia wszystkich pikseli znajdujących się na lewo

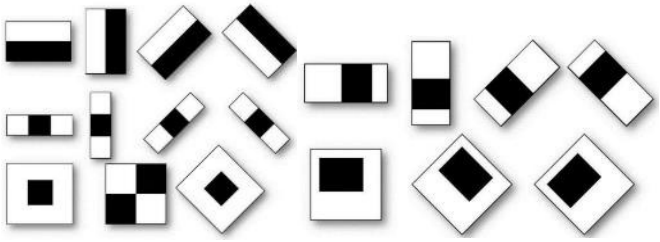
i powyżej tego elementu. Obliczenie elementów macierzy odbywa się za pomocą następującego wzoru [3]:

$$L(x,y) = \sum_{i=0,j=0}^{i\le x, j\le y} *I(i,j), \quad (1)$$

gdzie I(i, j) – jasność piksela obrazu źródłowego. Każdy element integralnego obrazu L(x, y) zawiera w sobie sumę wszystkich pikseli w prostokątnym odcinku od (0, 0) do (x, y) [4]. Obliczanie integralnego obrazu możliwe jest za pomocą następującego wzoru [10]:

$$L(x,y) = I(x,y) - L(x-1,y-1) + L(x,y-1) + L(x-1,y), \quad (2)$$

Cechy Haara, są używane do wyszukiwania obiektu. Obraz jest przetwarzany częściami, okno o określonej wielkości porusza się na obrazie i dla każdego odcinka przebytego obrazu, określa klasyfikatory Haara. Dostępność obiektu na obrazie zależy od różnicy między wartością cechy. W większości są używane prostokątne znaki (Rysunek 1).



Rys 1. Cechy wykorzystywane w klasyfikatorach Haara, po lewej główne oznaki, po prawej dodatkowe oznaki [9]

Wartość funkcji jest obliczana za pomocą wzoru:

$$F = X - Y, \quad (3)$$

gdzie: X – suma wartości jasności punktów na odcinku zamkniętym jasną częścią znaku, Y – suma wartości jasności punktów na odcinku zamkniętym ciemnym częścią znaku.

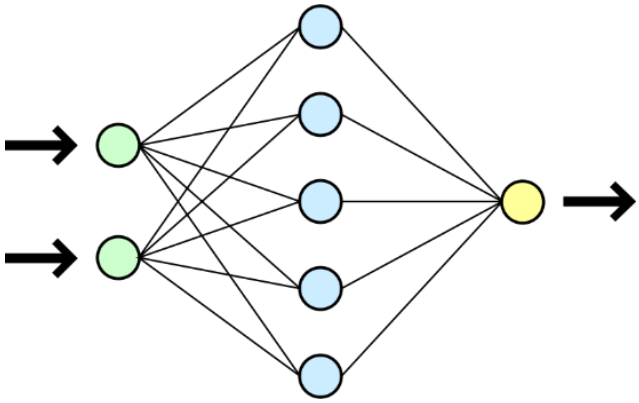
Do wyboru najbardziej odpowiednich cech szukanego obiektu na części obrazu wykorzystywany jest algorytm "boosting". "Boosting" oznacza poprawę. Metoda ta zwiększa skuteczność algorytmu uczenia, pozwala zbudować silny złożony klasyfikator łącząc słabe i proste klasyfikatory.

Bardziej ulepszony algorytm "boosting" – metoda AdaBoost, została zaproponowana w 1999 roku przez Yoavema Freundoem i Roberta Schapirem. Strumień wideo, uzyskany za pomocą kamery, jest sekwencją klatek. Dla każdego kadru jest obliczany jego zintegrowany obraz. Następnie kadr skanuje się oknem małych rozmiarów, zawierających znaki Haar. Dla każdego i-tego znaku odpowiedni kwalifikator określa wzór [4]:

$$h(z) = \begin{cases} 1, p_i f_i(z) < p_i Q_i \\ 0 \end{cases}, \quad (4)$$

gdzie: z – okno, Q_i – wartość progowa, p_i – kierunek znaku nierówności, f_i – cecha Haara.

Sztuczna sieć neuronowa to połączenie równoległo-szeregowe neuronów za pomocą synaps. Jej model matematyczny, realizowany za pomocą oprogramowania, jest mocno uproszczonym model mózgu (Rysunek 2).



Rys 2. Przykład sztucznej sieci neuronowej [6]

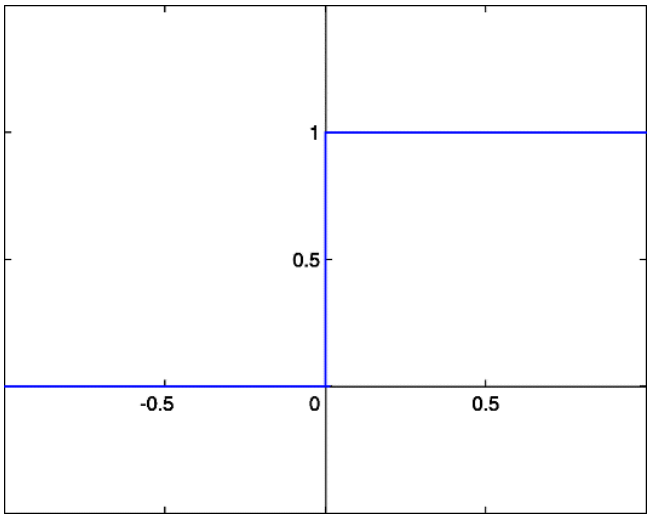
Sztuczne neuronowe sieci stosowane są do rozwiązywania trudnych zadań, wymagających analitycznego obliczenia, takich jak: klasyfikacja, przewidywanie i rozpoznawanie. Rozpoznawanie jet chyba najbardziej szerokim zastosowaniem sztucznych sieci neuronowych.

Przykładowa sieć neuronowa (Rysunek 2) składa się z neuronów i synaps. Posiadaona trzy warstwy neuronów: wejściową, ukrytą i wyjściową. W wejściowej warstwie podawane są dane wejściowe, a następnie w ukrytej warstwie odbywa się rozpoznawanie, po czym wyniki są przesyłane do warstwy wynikowej.

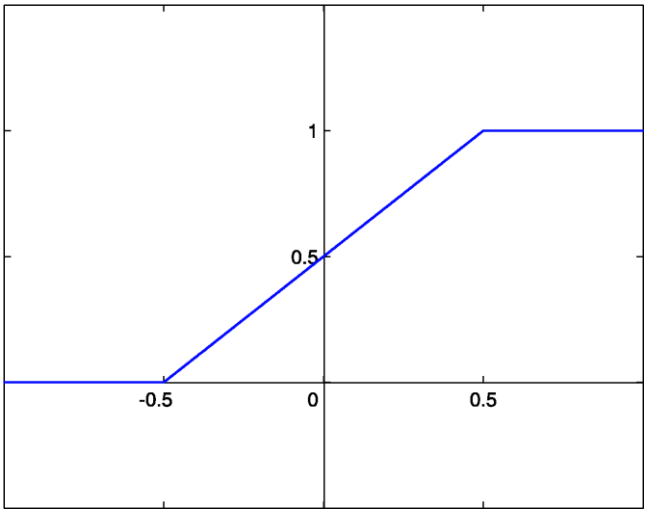
Neuron reprezentuje jednostkę obliczeniową, która wyznacza ważona sumę wartości wejściowych. Synapsa to łącznik neuronów, który ma swoją wagę. Sygnał wyjściowy zależy od sumy sygnałów wejściowych, pomnożonych przez ich wagi. W zależności od funkcji aktywacji, otrzymaną wartość aktywuje neuron i na wyjście będzie podawany sygnał 1, albo neuron nie zostanie aktywowany i w tym przypadku na wyjście będzie podawany sygnał 0.

Podstawowe funkcje aktywacji (Rysunki 3,4,5) to:

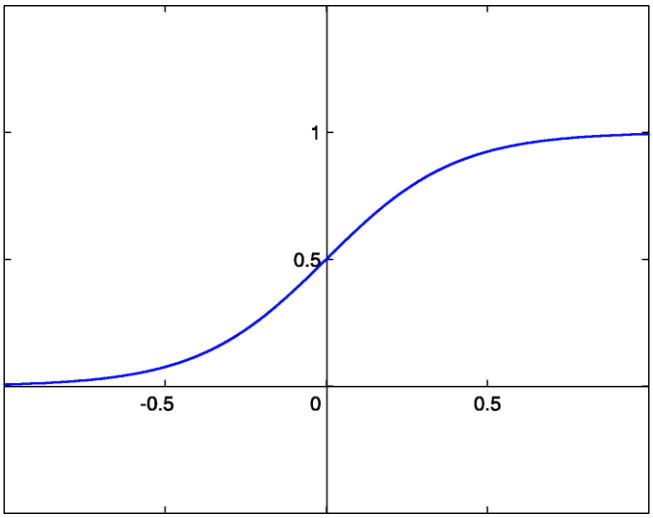
- graniczna funkcja,
- funkcja liniowa,
- "Sigmoidalna" funkcja.



Rys 3. Graniczna funkcja aktywacji [7]

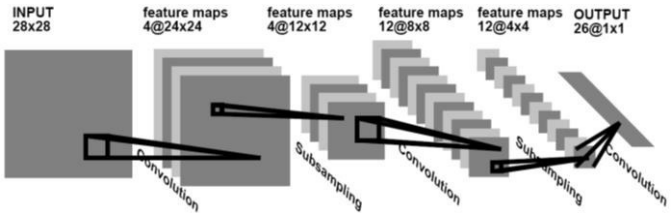


Rys 4. Liniowa funkcja aktywacji [7]



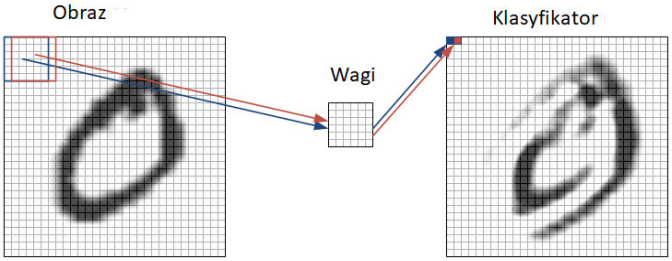
Rys 5. "Sigmoidalna" funkcja aktywacji [7]

Do pracy z obrazem można używać splotowe sieci neuronowe. Splotowe sieci neuronowe – to jeden z rodzajów sieci neuronowych często używany w komputerowym przetwarzaniu obrazów (Rysunek 6.) [11].



Rys 6. Praca splotowych sieci neuronowych [8]

Obraz uzyskany z kamery wchodzi w sieć neuronową nie w pełni, a rozbija się na obszarze o wymiarach n*n. Następnie te części po kolei przekazywane do pierwszej warstwy sieci neuronowej. Wszystkie przychodzące obszary zdjęcia są mnożone przez małą macierz wag. Rozmiar tej macierzy odpowiada przychodzącemu obszarowi obrazu, czyli n*n, taka macierz nazywa się jądrem. Wyniki mnożenia są sumowane i zapisywane w podobne stanowisko przychodzącego obrazu (Rysunek 7.).



Rys 7. Przykład zestawienia obrazu [8]

Każdy fragment obrazu element po elemencie jest mnożony przez niewielką macierz wag (rdzeń), wyniki są sumowane. Wynikiem mnożenia zostanie odebrana karta cech [11].

3. Cel i plan badań

Celem badań jest określenie skuteczności wykrywania obiektu na wideo za pomocą kamery internetowej podłączonej do komputera. Należy potwierdzić lub odrzucić następującą hipotezę:

Biblioteka OpenCV wykorzystującą metodę Viola-Jones pozwala na realizację komputerowego interfejsu sterowania gestami dłoni lub obrazem twarzy poprzez skuteczne rozpoznawanie obrazów dłoni lub twarzy na wideo za pomocą kamery podłączonej do komputera.

Do przeprowadzenia badań, została napisana aplikacja (Listing 1) z wykorzystaniem biblioteki OpenCV, która pozwala rozpoznawać obiekty za pomocą kamery podłączonej do komputera.

Listing 1. Kod programu

```
using System;
using System.Drawing;
using System.Windows.Forms;
using Emgu.CV;
using Emgu.CV.Structure;
using Emgu.CV.CvEnum;
namespace proga {
public partial class Form1 : Form {
private Capture capture;
private HaarCascade haarCascade;
int centr = 1;
public Form1() {
InitializeComponent();
}
private void timerTick(object sender, EventArgs e) {
using (Image<Bgr, byte> image = capture.QueryFrame()) {
if (image != null){
Image<Gray, byte> grayImage = image.Convert<Gray,
byte>();
var hands = grayImage.DetectHaarCascade(haarCascade,
1.4, 4, HAAR_DETECTION_TYPE.DO_CANNY_PRUNING,
new Size(image.Width / 8, image.Height / 8))[0];
foreach (var hand in hands){
image.Draw(hand.rect, new Bgr(0, double.MaxValue, 0), 3);
}
 pictureBox1.Image = image.ToBitmap();
}}}
private void btnStart_Click(object sender, EventArgs e){
capture = new Capture(0);
haarCascade = new HaarCascade("D:\\
haarcascade_frontalface_alt.xml.xml");
Application.Idle += timerTick;
}
private void label1_Click(object sender, EventArgs e){}
```

```
private void pictureBox2_Click(object sender, EventArgs e){}
}}
```

W celu określenia skuteczności rozpoznawania gestów przeprowadzono szereg eksperymentów. Zostały one przeprowadzone z udziałem 8 osób. Jako urządzenie został użyty komputer ze wbudowaną kamerą o rozdzielczości 1280x720 pikseli.Do pierwszego eksperymentu jest wykorzystywany klasyfikator do rąk, wyszkolony na przykładzie 500 pozytywnych obrazów rąk i 1000 negatywnych obrazów, na których nie ma rąk (tło i inny obiekty). Do drugiego i trzeciego eksperymentu użyto klasyfikator do rąk z biblioteki OpenCV. Do czwartego eksperymentu użyto klasyfikator do twarzy z biblioteki OpenCV. Badania przeprowadzono w pomieszczeniach z obecnością innych przedmiotów z uwzględnieniem różnego oświetlenia i różnej odległości od użytkownika do komputera. W trakcie badań każdy uczestnik pokazywał polecenie 10 razy. Wykonano 10 pomiarów, z których 7 pochodzi od 7 uczestników, i 3 pomiary przeprowadzone zostały przez jednego uczestnika. Wynikami badania są 3 wartości:

- a - rozpoznawanie;
- b - nie rozpoznawanie;
- c - mylne rozpoznawanie;

4. Wyniki badań

Do pierwszego eksperymentu użyto klasyfikator wyszkolonego za pomocą 500 pozytywnych zdjęć rąk i 1000 negatywnych zdjęć. Eksperyment jednak nie powiódł się, ponieważ klasyfikator był słabo nauczony.

Drugi eksperyment rozpoznawania ręki przeprowadzono w pomieszczeniu z naturalnym oświetleniem, podczas słonecznego dnia. Użyto klasyfikatora z biblioteki OpenCV. Jego wyniki przedstawiono w tabeli 1.

Tabela 1. Wyniki eksperymentu wykrywania ręki w słoneczną pogodę

Numer badania	Liczba wyników a/b/c				
	50cm	1m	1m 50cm	2m	2m 50cm
1	10/0/0	9/1/0	7/3/0	4/6/0	-
2	9/1/0	10/0/0	9/1/0	3/7/1	-
3	10/0/0	9/1/0	9/1/1	5/5/0	-
4	10/0/0	10/0/0	8/2/0	3/7/0	-
5	10/0/1	10/0/1	10/0/0	2/8/1	-
6	9/1/0	8/2/1	8/2/0	3/7/0	-
7	10/0/0	10/0/0	6/4/0	4/6/0	-
8	10/0/0	10/0/0	9/1/0	3/7/0	-
9	9/1/1	9/1/0	7/3/1	5/5/1	-
10	10/0/0	9/1/0	8/2/0	3/7/0	-

Z analizy wyników (Tabela 1) widać, że skuteczne rozpoznawanie ręki odbywa się w odległości do 1-go metra. Program działa normalnie w odległości do 1m 50cm. W odległości 2 metrów program działa źle, ponieważ współczynnik rozpoznawania jest niski. Praca z komputerem na odległość ponad 2 metrów nie jest możliwa, ponieważ rozpoznawanie ręki praktycznie nie zachodzi.

Trzeci eksperyment rozpoznawania ręki przeprowadzono w pomieszczeniu z naturalnym oświetleniem, w czasie wieczornej pory dnia. Użyto klasyfikatora z biblioteki OpenCV. Wyniki badań przedstawiono w tabeli 2.

Tabela 2. Wyniki eksperymentu wykrywania ręki w godzinach wieczornych

Numer badania	Wyniki a/b/c			
	50sm	1m	1m 50sm	2m
1	9/1/0	10/0/0	8/2/0	-
2	10/0/0	9/1/0	8/2/0	-
3	10/0/0	8/2/0	9/1/0	-
4	9/1/1	9/1/1	7/3/0	-
5	8/2/0	9/1/0	8/2/0	-
6	9/1/0	10/0/0	6/4/0	-
7	10/0/0	8/2/0	8/2/0	-
8	9/1/0	8/2/0	9/1/0	-
9	10/0/0	9/1/0	8/2/0	-
10	9/1/0	9/1/0	7/3/0	-

Prawidłowe rozpoznawanie ręki zachodzi w odległości do 1m 50cm. Ustalono, że słabsze oświetlenie zmniejszyło liczbę fałszywych rozpoznawań, ale jednocześnie zmniejszyła się skuteczność rozpoznawania ręki.

Następny eksperyment przeprowadzono przez rozpoznawanie twarzy. Był używany klasyfikator z biblioteki OpenCV. Rozmiar pliku klasyfikatora dla twarzy był kilka razy większy niż rozmiar pliku klasyfikatora dla rąk. Oznacza to, że dla nauczenia klasyfikatora twarzy użyto znacznie większą liczbę pozytywnych i negatywnych próbek obiektu. Wyniki badań przedstawiono w tabeli 3.

Tabela 3. Wyniki eksperymentu wykrywania twarzy

Numer badania	Wyniki a/b/c				
	50cm	1m	1m 50cm	2m	2m 50cm
1	10/0/0	10/0/0	10/0/0	10/0/0	-
2	10/0/0	10/0/0	10/0/0	10/0/0	-
3	10/0/0	10/0/0	10/0/0	10/0/0	-
4	10/0/0	10/0/0	10/0/0	9/1/0	-
5	10/0/0	10/0/0	10/0/0	10/0/0	-
6	10/0/0	10/0/0	10/0/0	10/0/0	-
7	10/0/0	10/0/0	10/0/0	10/0/0	-
8	10/0/0	10/0/0	10/0/0	9/1/0	-
9	10/0/0	10/0/0	10/0/0	10/0/0	-
10	10/0/0	10/0/0	10/0/0	10/0/0	-

Wyniki eksperymentu dotyczące wykrywania twarzy, okazały się najlepsze. Rozpoznawanie odbywa się na podobnej odległości, jak i rozpoznawania ręki. Ale przy tym rozpoznawanie twarzy ma wysoką skuteczność. Przy tym nie było żadnych fałszywych wykryć.

5. Wnioski

Do przeprowadzenia skuteczności rozpoznawania obiektów za pomocą kamery internetowej, została wybrana biblioteka OpenCV wykorzystująca metodę Viola-Jones, tak jak przypuszczano biblioteka była skuteczna dla rozwiązania tego zadania. W trakcie badań wykorzystano trzy karty cech i dwa rodzaje obiektów – ręka i twarz. W trakcie pracy z mniej

wyuczonym klasyfikatorem, wyniki wykazały niski poziom rozpoznawania. Podczas korzystania z lepiej wyuczonego klasyfikatora, wyniki wykazały wysoki poziom rozpoznawania. W trakcie badań z wykorzystaniem dobrze wyuczonego klasyfikatora do rozpoznawania twarzy, wyniki wykazały bardzo wysoką skuteczność. Wyniki badań potwierdziły założoną hipotezę: Biblioteka OpenCV wykorzystująca metodę Viola-Jones pozwala na realizację komputerowego interfejsu sterowania gestami dłoni lub obrazem twarzy poprzez skuteczne rozpoznawanie obrazów dłoni lub twarzy na wideo za pomocą kamery podłączonej do komputera. Jakość rozpoznawania zależy od jakości wyuczenia klasyfikatora:im lepiej wyuczony klasyfikator, tym wyższa skuteczność rozpoznawania.

Literatura

[1] Monali Chaudhari, Shanta Sondur, Gauresh Vanjare. «A review on Face Detection and study of Viola Jones method». International Journal of Computer Trends and Technology (IJCTT), 2015.

[2] Paul Viola , Michael Jones. “Rapid object detection using a boosted cascade of simple features”. Accepted conference on computer vision and pattern recognition. 2001.

[3] Paul Viola , Michael Jones. “Robust Real-time Object Detection”. International Journal of Computer Vision. 2004.

[4] Мурлин А.Г., Пиотровский Д.Л., Руденко Е.А., Янаева М.В. «Алгоритмы и методы обнаружения и распознавания жестов руки на видео в режиме реального времени». Научный журнал КубГАУ, 2014.

[5] Спицын В.Г., Буй Тхи Тху Чанг, Фан Нгок Хоанг. «Распознавание лиц на основе применения метода Виолы-Джонса, вейвлет-преобразования и метода главных компонент». Томский политехнический университет, 2012.

[6] https://commons.wikimedia.org/wiki/File:Neural_network.svg [13.12.2017]

[7] https://dic.academic.ru/dic.nsf/ruwiki/19743 [13.12.2017]

[8] https://geektimes.ru/post/74326/ [13.12.2017]

[9] https://habrahabr.ru/post/133826/ [13.12.2017]

[10] Метод Виолы Джонса как основа для распознавания лиц https://habrahabr.ru/post/133826/ [13.12.2017]

[11] Применение нейросетей в распознавании изображений. https://geektimes.ru/post/74326/ [13.12.2017]

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 8 (2018) 220-225
Wysłane: 2017-12-27
Przyjęte: 2018-01-12

Zabezpieczenia systemowe oraz sprzętowe dostępne dla użytkowników na urządzeniach pracujących pod kontrolą systemu Android

Tomasz Borysiewicz*

Politechnika Lubelska, Instytut Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie: Producent systemu Android już od pierwszej wersji platformy stara się oferować użytkownikom sposoby na ochronę przechowywanych na urządzeniach danych cyfrowych. Rozwój możliwości systemu oraz nowe rozwiązania technologiczne pozwalają na wprowadzanie coraz bardziej innowacyjnych zabezpieczeń, które cechują się nie tylko większym poziomem bezpieczeństwa ale również większym komfortem w codziennym użytkowaniu. Artykuł przedstawia porównanie dostępnych rozwiązań, zarówno względem zapewnienia najlepszej ochrony, jak i pod względem wygody użytkowania. Zagadnienie komfortu wymagało przeprowadzenia badań, w celu zebrania informacji o najbardziej popularnych konfiguracjach wśród użytkowników. Analiza bezpieczeństwa blokad oraz otrzymane wyniki badań, pozwoliły zweryfikować, czy właściciele urządzeń pracujących pod kontrolą systemu Android odpowiednio zabezpieczają dane przechowywane w pamięci wewnętrznej swoich urządzeń.

Słowa kluczowe: Android, zabezpieczenia; blokady; bezpieczeństwo

*Autor do korespondencji

Adres e-mail: borysiewicztomasz@gmail.com

System and hardware security options available for users on devices running Android operating system

Tomasz Borysiewicz*

Institute of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract. Producer of Android operating system have been trying to offer ways to protect digital data on devices since 1st version of this platform. Development of possibilities of operating system and new technological solutions allows to launch more innovate protections, which are not only more secure, but also very comfortable in everyday usage. This article presents comparison of available solutions, both in terms of the best protection and convenience of usage. The topic of convenience of usage required research to collect information about the most popular configurations among users. The analysis of security level of available locks and obtained results allowed to verify whether owners of devices running Android operating system are properly securing digital data stored in internal memory of their devices.

Keywords: Android, security; locks; safety

*Corresponding author

E-mail address: borysiewicztomasz@gmail.com

1. Wprowadzenie

Android jest systemem mobilnym, który po raz pierwszy na rynku zadebiutował 5 listopada 2007 roku. W zaledwie trzy lata zdobył ponad 80% rynku [1] i od tego czasu nieprzerwanie utrzymuje dominującą pozycję [2]. W pierwszym kwartale 2010 roku, a więc w momencie zdobycia przewagi nad konkurencją, pod kontrolą tej platformy pracowało około 50 milionów urządzeń, a liczba ta nieustająco wzrasta [2]. Producent systemu stara się zaoferować użytkownikom wydajny system o ogromnych możliwościach, który jednocześnie będzie bezpieczny. W tym celu już w pierwszej jego wersji pojawiły się 3 formy zabezpieczania urządzeń przed niepowołanym dostępem. Był to kod PIN, hasło alfanumeryczne oraz wzór blokady. Każde z tych rozwiązań cechowało się różnym poziomem bezpieczeństwa oraz komfortem użytkowania. Niestety to te najmniej bezpieczne blokady były najgodniejsze w użyciu, a więc Google nie ustawało w pracach mających na celu stworzenie lepszych rozwiązań. Fragmentacja Androida, która uznawana jest za jego największą wadę, okazuje się ogromną zaletą w kwestii rozwoju platformy. Każdy z producentów urządzeń, których pracą zarządza system Android, oferuje swoje autorskie rozwiązania zabezpieczania systemu, zarówno sprzętowe jak i programowe. Google ma zatem ogromną grupę testową, która weryfikuje nowatorskie metody blokady, a najlepsze z rozwiązań wprowadzane są do oficjalnej wersji systemu. Dzięki takim działaniom system wyposażony został w obsługę czytników linii papilarnych oraz różne rodzaje akcji, które sprawiają, że korzystanie z urządzenia ze skonfigurowanymi blokadami staje się wygodniejsze, co wielu użytkowników skłania do korzystania z dostępnych blokad. Dodatkowo, rozwój platformy zaowocował pojawieniem się możliwości szyfrowania pamięci wewnętrznej urządzenia, a także pojawieniem się usług *Google Find My Device*, która pozwala zdalnie zarządzać urządzeniem oraz lokalizować je.

Celem artykuł jest porównanie zaimplementowanych w systemie Android funkcji zabezpieczania systemu, których konfiguracja dostępna jest dla użytkowników. Porównanie ma na celu określenie zarówno poziomu bezpieczeństwa konkretnych rozwiązań, jak i wygodę korzystania z nich w życiu codziennym.

2. Potrzeba zabezpieczania urządzeń

Urządzenia pracujące pod kontrolą systemu Android towarzyszą użytkownikom w ich codziennym życiu. Telefony są narzędziem wykorzystywanym do komunikacji ze światem, a także zastępują wiele przedmiotów, z których korzystano przed rozwinięciem się technologii do obecnego stanu. W urządzeniu znajdują się aplikacje pozwalające zastąpić kalkulator, odtwarzacz muzyki, kalendarz, listę zadań, notatnik, budzik, mapę, nawigację, kartę bankową, a także programy pozwalające uzyskać dostęp do konta bankowego, wypożyczyć film, skorzystać z internetu, przesłać pliki, czy przeprowadzić rozmowy głosowe oraz video. Dodatkowo większość aplikacji pozwala synchronizację z serwerem, dzieci czemu wprowadzone treści na jednym z urządzeń dostępne są na pozostałych, powiązanych z tym samym kontem. Ogrom funkcjonalności oraz liczność aplikacji, które zapewniają do nich dostęp podnoszą także ryzyko wycieku, bądź przechwycenia wrażliwych danych. Potencjalny złodziej danych może uzyskać dostęp do bardzo wielu informacji o właścicielu urządzenia, takich jak historia lokalizacji, prywatne dokumenty czy notatki, zapamiętane hasła, dostępy do różnych kont (w tym bankowego), a także informacji, które nie wydają się być newralgiczne, między innymi upodobania muzyczne czy filmowe użytkownika. Wszystkie te dane w połączeniu tworzą skarbnicę wiedzy o właścicielu urządzenia, o jego danych personalnych, pracy, czy życiu codziennym, dlatego tak ważne jest jak najlepsze zabezpieczanie danych. Usługi, z których korzystają użytkownicy są na ogół bardzo dobrze zabezpieczone po stronie usługodawcy, jak również sam system Android stara się dostarczyć jak najlepsze mechanizmy zabezpieczania platformy. W przypadku ataków na serwery usługodawców użytkownik niestety nie ma żadnego wpływu na bezpieczeństwo swoich danych, jednak większość ataków na systemy mobilne skupia się na wykorzystaniu braku wiedzy użytkowników, bądź nieodpowiednim zarządzaniu blokadami w systemie. W tych kwestiach jednak użytkownik może zdobyć dodatkową wiedzę i wpłynąć na bezpieczeństwo swoich danych cyfrowych.

3. Dostepne zabezpieczenia

3.1. Rozwiązania wprowadzone przez producenta systemu Android

Najnowsze wersje platformy wciąż posiadają blokady, które obecne były w pierwszej kompilacji systemu, a więc kod PIN, hasło alfanumeryczne oraz wzór blokady. W kolejnych iteracjach pojawiła się obsługa czytnika linii papilarnych, szyfrowanie wewnętrznej pamięci urządzenia, kontrolowanie uprawnień aplikacji, a także usługę *Google Find My Device*. Dodatkowo wprowadzono opcje, które ułatwiają interakcję z systemem, na którym skonfigurowana zostały zabezpieczenia, a są to funkcje podwójnego stuknięcia, przesunięcia w celu odblokowania oraz podwójnego stuknięcia. Najnowszym tego typu rozwiązaniem jest zbiór inteligentnych blokad, w skład których wchodzą zaufane urządzenia, zaufane lokalizacje, zaufana twarz, zaufany głos oraz rozpoznawanie kontaktów z ciałem.

3.2. Rozwiązania wprowadzone przez producentów urządzeń pracujących pod kontrolą systemu Android

Producenci urządzeń mobilnych często modyfikują platformę aby dodać firmowy akcent estetyczny lub umieścić w nim własne aplikacje. Zmiany niejednokrotnie dotyczą także rozszerzenia funkcjonalności systemu, w tym obsługi dodatkowych urządzeń wykorzystywanych do podniesienia bezpieczeństwa systemu oraz rozszerzeń podnoszących komfort korzystania z urządzenia. Do tego typu funkcjonalności należą inteligentne gesty, *moto actions*, *knock code* oraz skaner tęczówki.

4. Porówanie zabezpieczeń

4.1. Porównanie zabezpieczeń pod względem poziomu bezpieczeństwa

Najlepsza formą zabezpieczania danych w pamięci wewnętrznej urządzenia jest szyfrowanie. Pozwala to uniknąć przechwycenia danych w przypadku utraty telefonu, pod warunkiem, że był on wyłączony w momencie trafienia w niepowołane ręce, bądź posiadał skonfigurowane blokady ekranu.

Najlepsze z dostępnych rozwiązań to aktualnie czytnik linii papilarnych oraz skner tęczówki. Poziom bezpieczeństwa jak obu blokad jest bardzo zbliżony, dzięki analogicznemu sposobowi działania. Z odczytanego obrazu tworzony jest graf, na podstawie charakterystycznych punktów, takich jak zakończenia i rozgałęzienia linii papilarnych, czy specyficzne rozdzielenia tkanki tęczówki. Stworzony graf porównywany jest z tym, który utworzony został podczas konfiguracji blokady, a warunkiem na przyznanie dostępu do obsługi systemu jest zawieranie się nowopowstałego grafu w pierwowzorze [3][4]. Dzięki dużej unikalności badanych punktów wśród społeczeństwa, obydwie blokady wykazują się wysokim poziomem bezpieczeństwa, a więc powinny być pierwszym wyborem użytkowników, którzy cenią sobie bezpieczeństwo. Ogromnym atutem tych rozwiązań jest również brak możliwości ominięcia ich przez osoby postronne. Nawet jeżeli potencjalny złodziej próbowałby zaobserwować sposób na odblokowanie urządzenia, po jego przejęciu niemożliwe byłoby uzyskanie dostępu do systemu bez obecności uwierzytelnionej osoby. Niestety blokady te uzależnione są od urządzenia, a więc nie wszyscy użytkownicy mają możliwość skorzystania z nich. Na szczęście platforma Android oferuje wiele innych rozwiązań, które wykorzystane mogą być na telefonach o słabszej specyfikacji sprzętowej, a producenci niektórych urządzeń oferują również swoje autorskie rozwiązania, podobnie jak Samsung, który umieścił w swoich flagowych modelach skaner tęczówki oka.

Spośród podstawowych metod blokady, dostępnych już od pierwszej wersji systemu Android najbezpieczniejszą opcją będzie blokada hasłem. Rozwiązanie to pozwala na stworzenie skomplikowanego ciągu alfanumerycznego, który nie tylko trudno odgadnąć bądź złamać, ale również ciężko podejrzeć z pozycji osoby trzeciej podczas wprowadzana go na klawiaturze telefonu przez osobę uwierzytelnioną. O wiele prostsze jest to w przypadku blokady kodem PIN. Długość kodu może wynosić maksymalnie 8 znaków oraz musi się ona składać wyłącznie z cyfr. Brak możliwości zastosowania znaków specjalnych oraz liter, bardzo uszczupla dostępną pulę możliwych kombinacji. Rozwiązanie tego typu jest zatem o wiele łatwiej złamać, a uproszczona klawiatura

wprowadzania kodu, sprawia że obserwacja właściwej kombinacji z pozycji osoby trzeciej jest również o wiele prostsza. Najmniej bezpiecznym rozwiązaniem jest natomiast wzór blokady, zwany popularnie wężykiem. Polega on na wprowadzeniu przez użytkownika na ekranie urządzenia linii przechodzącej przez odpowiednie punkty, konkretnej kolejności, bez możliwości wykorzystania punktów więcej niż jeden raz. Pomimo wygody tego rozwiązania, ilość możliwych kombinacji jest dość ograniczona, a obserwacja poprawnego wzoru jest o wiele łatwiejsza niż w przypadku wyżej opisywanych rozwiązań. Co więcej, po wprowadzeniu wzoru blokady, na urządzeniu pozostają charakterystyczne wzory pozostawione przez palec użytkownika, dzięki czemu odgadnięcie właściwej kombinacji niejednokrotnie okazuje się naprawdę proste. Rozwiązaniem dość podobnym do kodu PIN jest blokada wprowadzona przez firmę LG, nazwana *knock code*. Podczas korzystania z tej metody zabezpieczenia, użytkownik ma do dyspozycji cztery części ekranu, a odpowiednia kombinacja stuknięć w poszczególne ćwiartki powala uzyskać dostęp do systemu[5]. Ilość możliwych kombinacji jest bardzo niewielka, ponieważ tego typu blokadę można by rozważyć jako czteroznakowy kod PIN, których składać się może tylko z cyfr w zakresie 1-4. Złamanie takiego kodu jest stosunkowo proste, a sposób wprowadzania sekwencji stuknięć jest łatwy do zaobserwowania, a więc blokada ta powinna być ostatnim wyborem wśród użytkowników ceniących sobie bezpieczeństwo danych.

Dodatkowymi rozwiązaniami, które wprowadzone zostały przez producentów urządzeń pracujących pod kontrolą systemu Android, są inteligentne gesty, inteligentne blokady oraz *moto actions*. Wprawdzie głównym celem wprowadzenia tych rozwiązań było podniesienie komfortu korzystania z telefonu, jednak w wielu sytuacjach podnoszą one bezpieczeństwo danych w pamięci wewnętrznej urządzenia. Inteligentne gesty oraz *moto actions* pozwalają wywoływać przydatne funkcjonalności systemu (takie jak aparat czy latarka) bez potrzeby odblokowywania urządzenia, dzięki czemu użytkownik korzystający z łatwych do zaobserwowania metod blokady dostępu, może spokojnie skorzystać z niektórych funkcji systemu, bez ryzyka, że w tłocznym miejscu ktoś podejrzy wymagany kod lub wzór blokady. Rozwiązanie inteligentnych blokad oferuje możliwość całkowitego wyłączenia blokady w określonych sytuacjach, bądź pominięcie wprowadzania hasła przy spełnieniu określonych warunków. Zabezpieczenia mogą zostać wyłączone w określonych lokalizacjach, podczas korzystania z określonych urządzeń bluetooth w zasięgu telefonu, bądź w momencie, kiedy użytkownik przemieszcza się z urządzeniem, natomiast brak potrzeby wprowadzania hasła zaistnieje kiedy przednia kamera urządzenia rozpozna twarz właściciela, bądź mikrofon zarejestruje odpowiednią frazę, wypowiedzianą głosem uwierzytelnionej osoby. Korzyści używania tych rozwiązań są analogiczne jak w przypadku inteligentnych gestów oraz *moto actions*, jednak nie wszystkie charakteryzują się taką samą odpornością na próby złamania. Rozpoznawanie głosu, lokalizacji oraz urządzeń jest wystarczająco bezpieczne aby korzystać z tych rozwiązań na co dzień, pamiętając jednak by nie pozostawiać zaufanego urządzenia wraz z telefonem bez nadzoru oraz nie dodawać zbyt wielu zaufanych lokalizacji, szczególnie takich, które są miejscami publicznymi. Rozpoznawanie twarzy jest podatne na próby oszustwa zdjęciem uwierzytelnionej osoby, a wykrywanie kontaktu z ciałem może pozwolić uzyskać złodziejowi pełen dostęp do systemu, jeżeli napastnik wyciągnie go z torebki czy kieszeni właściciela, ponieważ blokada uruchomiła by się dopiero wtedy, kiedy urządzenie przestałoby być w ruchu, a więc korzystanie z tych rozwiązań nie jest bezpieczne.

Niewspomniane powyżej rozwiązania, takie jak podwójne stuknięcie czy przesunięcie ekranu w celu odblokowania, bez skonfigurowanych dodatkowych blokad, w żaden sposób nie zabezpieczają systemu przed dostępem niepowołanych osób. Korzystanie z nich nie wpływa zatem w żaden sposób na bezpieczeństwo danych na urządzeniu, zatem używanie tylko tych rozwiązań jest całkowicie niebezpieczne.

Poza metodami blokowania urządzenia, producent systemu Android wprowadził również funkcjonalność o nazwie *Google Find My Device*. Dostępne w ramach tej usługi opcje, pozwalają odnaleźć zagubione, bądź skradzione urządzenia poprzez wyświetlenie ich aktualnej pozycji geograficznej na mapie, zdalnie wywołać na urządzeniu dźwięk w celu zlokalizowania go w bliskim otoczeniu użytkownika, zablokować urządzenie, tak aby możliwe było skorzystanie z niego tylko po wprowadzeniu hasła do konta Google, a także zdalnie wyczyścić pamięć urządzenia. Usługa zdecydowanie podnosi bezpieczeństwo danych na urządzeniu, pozwalając zdalnie zablokować do nich dostęp lub usunąć je, a korzystanie z tego rozwiązania nie wymaga żadnej konfiguracji ze strony użytkownika.

W szóstej wersji platformy Android o nazwie Marshmallow, producent systemu wprowadził możliwość kontrolowania uprawnień aplikacji [6]. W celu wykonania danej operacji użytkownik musi zezwolić aplikacji na skorzystanie z danych zasobów systemu, bądź podzespołów urządzenia. Rozwiązanie to posiada pewne wady, jak na przykład zbyt szerokie grupowanie uprawnień, przez co użytkownik może nie wiedzieć, czy program chce jedynie odczytać informacje z danego zasobu czy też je nadpisać. Pomimo tego, jest to przyznanie właścicielowi urządzenia pewnego rodzaju kontroli nad aplikacjami, dzięki któremu może zablokować niektóre, niepożądane operacje, co niewątpliwie wpływa bardzo pozytywnie na bezpieczeństwo danych w pamięci wewnętrznej urządzenia.

Użytkownicy tabletów pracujących pod kontrolą systemu Android mają do wyboru te same metody blokowania urządzenia, co użytkownicy telefonów, za wyłączeniem rozwiązań wprowadzanych przez producentów urządzeń, a zatem inteligentnych gestów, skanera tęczówki oka, *knock code* oraz *moto actions*. Dostępne blokady działają w taki sam sposób jak na telefonach, a więc kolejność ich wyboru ze względu na poziom bezpieczeństwa, również będzie taka sama.

Właściciele inteligentnych zegarków mają możliwość skonfigurowania jedynie najprostszych metod blokady, a zatem hasła, kodu PIN oraz wzoru blokady. Stosowanie zabezpieczeń na inteligentnych zegarkach jest bardzo ważne, ponieważ kolejne aktualizacje systemu wprowadzają coraz więcej możliwości systemu, przez co narażone mogą być dane oraz finanse użytkownika [7]. Zasada działania dostępnych

blokad różni się nieco w porównaniu do telefonów oraz tabletów, ponieważ zabezpieczenie uruchamiane jest jedynie wtedy, kiedy urządzenie znajduje się w stanie spoczynku, a jeżeli użytkownik ma je cały czas przy sobie, blokada pozostaje wyłączona. Jednak sam proces odblokowywania urządzenia oraz możliwości konfiguracji kombinacji autoryzacyjnych są niezmienne, a więc hierarchia dostępnych rozwiązań, ze względu na poziom bezpieczeństwa, będzie analogiczna jak przy wyżej opisywanych urządzeniach.

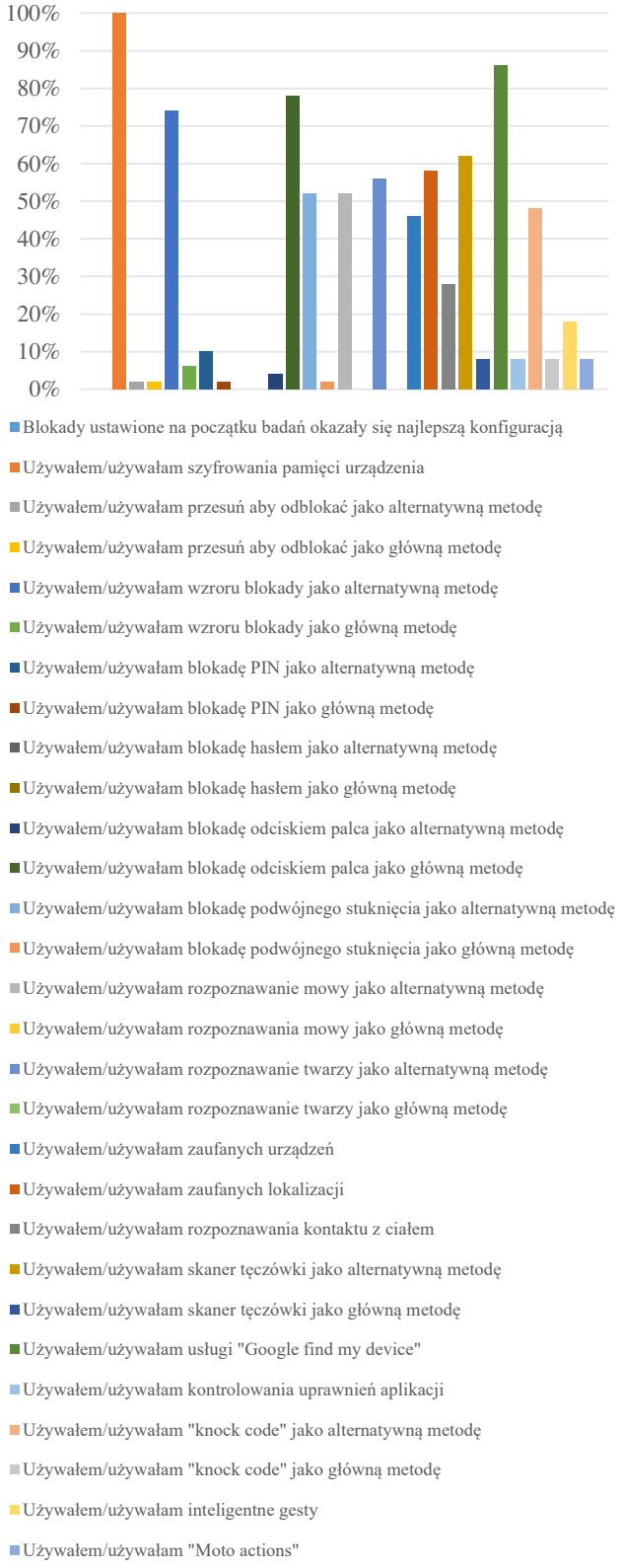
W przeciwieństwie do pozostałych urządzeń pracujących pod kontrolą systemu Android, właściciele telewizorów, których pracą również zarządza ta platforma, nie posiadają żadnej możliwości zabezpieczenia swoich danych cyfrowych. Co więcej Android pracujący na telewizorach jest jedyną wersją tego systemu, która nie wspiera usługi *Google Find My Device*, zatem użytkownicy powinni zwracać szczególną uwagę na to, aby nie przechowywać na tego typu urządzeniach wrażliwych informacji, prywatnych zdjęć, czy informacji wymaganych do płatności elektronicznych.

4.2. Prówanie zabezpieczeń pod względem komfortu użytkowania

W celu zweryfikowania jakie metody zabezpieczeń najchętniej wybierają użytkownicy przeprowadzone zostały trzytygodniowe badania na pięćdziesięcioosobowej grupie chętnych. Badani zostali dobrani tak, aby podczas testów mieli możliwość skorzystania urządzeń pochodzących od różnych wytwórców. Dzięki temu możliwe było wypróbowanie przez nich wszystkich metod zabezpieczania systemu. Uczestnicy zostali dobrani również pod względem zainteresowań i pochodzenia, aby badania przedstawiały globalne zachowanie użytkowników, którzy niekoniecznie interesują się tematem bezpieczeństwa oraz technologii. Po zakończonym okresie badań, uczestnicy poproszeni zostali o wypełnienie ankiety, której wyniki przedstawia wykres (Wykres 1). Poniższy test zawiera analizę otrzymanych wyników.

Jedynym rozwiązaniem, z którego korzystało 100% badanych jest szyfrowanie pamięci urządzenia. Rozwiązanie to jest dla użytkownika całkowicie niezauważalne, a więc w żaden sposób nie utrudnia codziennej pracy z urządzeniem, będąc jednocześnie bardzo dobrą ochroną danych przechowywanych w pamięci urządzenia.

Wśród blokad dostępu do systemu, zdecydowaną przewagę zyskał czytnik linii papilarnych, który aż 76% użytkowników skonfigurowało jako swoją główną metodę zabezpieczania urządzenia przed niepowołanym dostępem. Niewątpliwie jest to sukces producenta systemu Android, ponieważ jedna z najlepszych blokad oferowanych w platformie jest jednocześnie wystarczająco wygodna, aby użytkownicy chętnie korzystali z niej na co dzień. Druga pod względem bezpieczeństwa metoda blokady, zaproponowana przez firmę Samsung nie uzyskała już tak dobrego wyniku. Zaledwie 8% badanych zdecydowało się na skonfigurowanie skanera tęczówki oka jako główną metodę uwierzytelniania w systemie. Jednak jako alternatywna metoda, uzyskany wynik był o wiele lepszy, bo wynosił aż 62%. Prostsze metody blokady, takie jak *knock code*, kod PIN, hasło oraz wzór odblokowania nie zyskały zainteresowania użytkowników jako główna metoda blokady. Urządzenia badanych wyposażone były w wygodniejsze rozwiązania, a więc wybór użytkowników wydaje się być oczywisty.



Rys. 1. Procentowy wykaz wykorzystywania poszczególnych blokad wśród grupy badanych użytkowników

Najbezpieczniejsze z tych rozwiązań, a więc blokada hasłem okazała się niestety na tyle uciążliwa, że żaden z uczestników badania nie zdecydował się skonfigurować jej nawet jako

zabezpieczenie alternatywne. Jako dodatkową blokadę użytkownicy najchętniej wybierali wzór blokady, który wybrało aż 74% badanych. Mimo dużej podatności tego rozwiązania na obserwację wprowadzanej kombinacji z pozycji osoby trzeciej, jest to stosunkowo bezpieczna blokada jako zabezpieczenie dodatkowe. Użytkownik wprowadza wzór tylko w nielicznych sytuacjach, kiedy zawiodły inne możliwości uwierzytelnienia dostępu, a więc ciężko jest podejrzeć poprawną sekwencję. Potencjalny złodziej musiałby zatem spróbować złamać tę blokadę, co nie byłoby proste, ponieważ możliwych jest ponad 400 000 różnych kombinacji połączenia punktów [8].

Na korzystanie z rozwiązań, które w żaden sposób nie podnoszą bezpieczeństwa danych przechowywanych na urządzeniu, zdecydowało się łącznie jedynie 4% badanych, którzy skonfigurowali przesunięcie w celu odblokowania, bądź podwójne stuknięcie, jako główne metody blokady. Tak mały odsetek badanych spowodowany był najprawdopodobniej zbyt dużym ryzykiem przypadkowego odblokowania urządzenia w kieszeni czy torebce. Podwójne stuknięcie w ekran było jednak chętnie wybieraną alternatywną metodą blokady. Aż 52% badanych korzystało z tego rozwiązania w celu wybudzenia urządzenia ze stanu czuwania.

Z możliwości kontrolowania uprawnień aplikacji korzystało jedynie 8% badanych. Najprawdopodobniej użytkownicy nie posiadają odpowiedniej wiedzy, które uprawnienia są aplikacjom niezbędne do poprawnego działania i przyznają im wszystkie żądane dostępy. Z kolei wyłączenie raz przyznanych uprawnień jest „schowane" w systemie, a więc sposób obsługi dostępów nie jest wystarczająco przejrzysty dla użytkowników, aby zdecydowali się na rozważne kontrolowanie uprawnień.

Rozwiązania wprowadzone przed producentów urządzeń w celu podniesienia komfortu wykonywania prostych akcji na telefonie, bez potrzeby odblokowywania go, a więc *moto actions* oraz inteligentne gesty, nie były często wybieranymi rozwiązaniami wśród użytkowników. Łącznie, z obu blokad korzystało zaledwie 20% użytkowników. Opcje oferowane przez inteligentne gesty były wybierane przez użytkowników zdecydowanie częściej. Ponad 50% badanych korzystało z zaufanych lokalizacji, 46% skonfigurowało zaufane urządzenia, a 28% używało rozpoznawania kontaktu z ciałem w celu czasowej dezaktywacji zabezpieczenia systemu. Dodatkowo, jako alternatywne metody odblokowania urządzenia, użytkownicy wykorzystywali zaufany głos (52%) oraz rozpoznawanie twarzy (56%).

Usługa *Google Find My Device* okazała się chętnie wykorzystywanym rozwiązaniem wśród badanych. Aż 86% z nich zadeklarowało aktywne korzystanie z usługi podczas przeprowadzonych badań. Podobnie jak szyfrowanie pamięci urządzenia, korzystanie z tej usługi jest dla użytkownika całkowicie niezauważalne i w żaden sposób nie wpływa na komfort korzystania z urządzenia. Zaawansowane opcje zdalnej ochrony danych zachęciły jednak badanych do korzystania z tej usługi, w celu odszukania swoich urządzeń, bądź zabezpieczenia danych na telefonach.

Przeprowadzone badania dotyczyły korzystania z telefonów, jednak uzyskane wyniki można odnieść również do właścicieli tabletów, które pracują pod kontrolą systemu Android, ponieważ wszystkie dostępne zabezpieczenia działają w ten sam sposób, a więc wybór użytkowników również były taki sam. Na podstawie otrzymanych wyników można jednoznacznie stwierdzić, że użytkownicy nie są obojętni na bezpieczeństwo swoich danych, jednak przedkładają wygodę ponad ochronę danych. Z całą pewnością można zatem stwierdzić, że odsetek użytkowników korzystających z blokady ekranu na inteligentnych zegarkach byłby naprawdę niewielki, a właściciele telewizorów, których pracą zarządza platforma Android nie zdecydowaliby się na skorzystanie z jakichkolwiek blokad, nawet, jeżeli byłyby one dostępne.

5. Wnioski

Na podstawie wyników otrzymanych badań można zauważyć, że użytkownicy pomimo braku obojętności na bezpieczeństwo danych, wybierają jednak takie rozwiązania i blokady, które są jak najbardziej komfortowe w codziennym użytkowaniu i ułatwiają pracę z urządzeniem. Porównanie zabezpieczeń pod względem bezpieczeństwa blokad, w zestawieniu z odpowiedziami użytkowników na temat przeprowadzonych badań, ukazują że użytkownicy przedkładają wygodę ponad bezpieczeństwo danych przechowywanych w pamięci wewnętrznej urządzeń pracujących pod kontrolą systemu Android. Przyczyny takiego zachowania mogą być różne. Może być to brak wiedzy użytkowników o istniejących zagrożeniach, brak świadomości na temat istnienia niektórych blokad, jak również chęć wygodnej pracy z urządzeniem, pomimo świadomości narażania plików na ataki i wycieki. Niewątpliwie dałoby się ustalić konfigurację, która zapewniłaby odpowiednią ochronę, nie utrudniając jednocześnie pracy z urządzeniem. Problemem jednak byłoby zachęcenie właścicieli urządzeń do korzystania z takiego zestawu blokad, ponieważ musieliby zrezygnować z niektórych wygód, do których przyzwyczaili się przez lata korzystania z systemu Android. Co więcej, wielu z nich należałoby zainteresować tematem bezpieczeństwa oraz możliwości wykorzystania przechwyconych danych, aby mieć pewność, że w przypadku pojawienia się kolejnych metod blokady urządzenia, zweryfikują oni poziom jej bezpieczeństwa i rozważą korzystanie z niej. Producent systemu rozumie istniejące problemy i stara się dostarczać coraz lepsze rozwiązania, które są nie tylko coraz bardziej bezpieczne ale także coraz wygodniejsze w codziennej pracy z urządzeniem. Jak łatwo można zauważyć to na przykładzie czytnika linii papilarnych, z powodzeniem udaje się wprowadzać takie rozwiązania do codziennego użytku, jednak dzieje się to stosunkowo powoli. Ogromna grupa docelowa oraz mnogość wytwórców urządzeń sprawia, że niejednokrotnie te najlepsze rozwiązania trafiają do wielu użytkowników z dużym opóźnieniem, na przykład z powodu dużych kosztów wytworzenia urządzeń wspierających daną blokadę.

Google niewątpliwie stara się dostarczać jak najlepiej zabezpieczony system, nieustannie wprowadzając do systemu poprawki bezpieczeństwa. Rynek mobilny rozwija się jednak bardzo szybko i nawet tak ogromna firma nie jest w stanie zapewnić rozwiązań spełniających oczekiwania wszystkich użytkowników. Należy więc starać się uświadamiać

właścicieli urządzeń pracujących pod kontrolą systemu Android do korzystania z dostępnych zabezpieczeń oraz przekonywać ich do poświęcania części komfortu codziennej pracy z urządzeniem, w celu zapewnienia bezpieczeństwa swoim danym cyfrowym, a przez to i samym sobie.

Literatura

[1] 3 Android Data Showing "Why Android is the New King of Technology?", SOURCE DIGIT, http://sourcedigit.com/1913-smartphone-os-global-market-share-data-2014 [11.04.2016]

[2] Smartphone OS Market Share, 2017 Q1, IDC, http://www.idc.com/prodserv/smartphone-os-market-share.jsp, [02.01.2016]

[3] Poznajmy technologię: Jak działa czytnik linii papilarnych?, LenovoZone, https://lenovozone.pl/porady/poznajmy-technologie-jak-dziala-czytnik-linii-papilarnych [17.06.2017]

[4] T. Nick, Here is how the iris scanner on the Galaxy Note 7 works, phoneArena.pl, https://www.phonearena.com/news/Here-is-how-the-Galaxy-Note-7-iris-scanner-works_id82854 [15.08.2017]

[5] J. Stępień, Knock Code przełomowa metoda budzenia smartphona, www.shemag.pl/trend/knock-code-przelomowa-metoda-budzenia-smartphona [30.08.2017]

[6] Control your app permissions on Android 6.0 and up, Google Play Help, https://support.google.com/googleplay/answer/6270602 [15.08.2017]

[7] P. Lamkin, Android Wear 2.0: Ultimate guide to the major smartwatch update, WAREABLE, https://www.wareable.com/android-wear/android-wear-update-everything-you-need-to-know-2735 [16.09.2017]

[8] U. Malhotra, How many combinations of locking pattern are possible for a Samsung 3*3 locking grid?, Quora, https://www.quora.com/How-many-combinations-of-locking-pattern-are-possible-for-a-Samsung-3*3-locking-grid [03.10.2017]

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 8 (2018) 226-229
Wysłane: 2018-01-24
Przyjęte: 2018-01-30

Efektywność tworzenia warstwy prezentacji aplikacji we frameworkach AngularJS, Angular2, BackboneJS

Monika Tobiańska*, Jakub Smołka

Politechnika Lubelska, Instytut Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. Niniejszy artykuł poświęcony jest analizie porównawczej trzech frameworków służących do tworzenia warstwy prezentacji aplikacji. Przeprowadzone zostały trzy rodzaje badań na dwóch przeglądarkach, Google Chrome oraz Mozilla Firefox. Wzięto pod uwagę złożoność kodu, szybkość generowania widoku, płynność działania aplikacji przy obciążeniu danymi, ilość przesyłanych danych potrzebnych do uruchomienia aplikacji oraz zużycie pamięci zajmowanej przez program w zależności od liczby elementów na liście. Do pomiarów wykorzystano aplikacje TodoMVC napisane przy użyciu narzędzi: AngularJS, Angular2 i BackboneJS. Przeprowadzone eksperymenty wykazały, że framework Angular2 w przeglądarce Google Chrome uzyskał najlepszy wynik. BackboneJS natomiast był faworytem dla przeglądarki Mozilla Firefox.

Słowa kluczowe: AngularJS; Angular2; BackboneJS; Wydajność, Framework frontendowy

*Autor do korespondencji.

Adres e-mail: monika.tobianska@gmail.com

Efficiency of creating application's presentation layer with frameworks AngularJS, Angular2, BackboneJS

Monika Tobiańska*, Jakub Smołka

Institute of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract. This article is focused on comparative analysis of three frameworks to create presentation layer of application. Three analysis have been conducted on two browsers, Google Chrome and Mozilla Firefox. Code complexity, the speed of view rendering, the smoothness of working of overstretched application, the amount of sent data needed to start application and the amount of memory used by application depending on the number of elements on list was took into consideration. In research purposes TodoMVC applications made with AngularJS, Angular2 and BackboneJS were used. Conducted experiments shown that Angular2 in Google Chrome achieve the best result. BackboneJS was the winner in Mozilla Firefox.

Keywords: AngularJS; Angular2; BackboneJS; Performance, Frontend framework

*Corresponding author.

E-mail address: monika.tobianska@gmail.com

1. Wstęp

Obecnie Internet stał się najpopularniejszym źródłem informacji. Strony internetowe wypierane są przez bardziej funkcjonalne programy komputerowe dostępne przez przeglądarki internetowe.

Frameworki określają strukturę aplikacji, jej mechanizm działania oraz dostarczają biblioteki i komponenty przydatne podczas tworzenia programów. Od kilku lat na rynku dostępne są nowe szkielety do budowy aplikacji. Głównym celem twórców jest dostosowanie ich narzędzi do różnego rodzaju systemów. Co więcej chcą także sprostać wymaganiom programistów oraz wyeliminować niedogodności napotkane podczas korzystania z poprzednich wersji frameworków.

Wielu programistów korzysta z frameworków dlatego znalezienie literatury dotyczącej omawianych narzędzi nie było trudne.

W [1] porównany został AngularJS, EmberJS i BackboneJS. Autor stwierdził, że dwa pierwsze frameworki są podobne pod względem ilości dostarczanych funkcjonalności. Dwa ostatnie natomiast w zbliżony sposób obsługują szablony widoków.

W [2,3] zaprezentowano różnice między AngularJS a Angular2. Porównano sposób tworzenia i używania kontrolerów oraz konfigurację aplikacji. Te same technologie omówione zostały w [4]. Z artykułu wynika, że AngularJS nie został stworzony, aby być wydajnym rozwiązaniem, lecz aby łatwo się go używało. W pewnym momencie efektywność zaczęła odgrywać ważną rolę dlatego wydano Angular2, który zwiększa szybkości działania aplikacji 3-10 krotnie..

Jeden z pracowników Juniper Networks napisał artykuł zestawiający ze sobą frameworki AngularJS, Angular2, ReactJS i BackboneJS z MarionetteJS [5]. Przeprowadzona analiza pozwoliła mu na pokazanie wyższości Angular2 nad innymi narzędziami.

Ciekawe porównanie przedstawiono w [6]. Artykuł dotyczący EmberJS, AngularJS oraz BackboneJS przedstawia aspekty, na które należy zwrócić uwagę podczas wyboru narzędzia. Jednym z najważniejszych aspektów okazała się społeczność, im większa tym łatwiej znaleźć odpowiedź

na pojawiające się pytania. Równie ważny był czas ładowania strony ponieważ użytkownikom zależy na jak najszybszym znalezieniu interesujących ich treści.

Celem opisanej w niniejszym artykule pracy badawczej było stworzenie porównania wybranych frameworków do tworzenia warstwy prezentacji aplikacji webowej. Porównanie to ma ułatwić wybór narzędzia, z którego programiści będą korzystać przy pisaniu systemów. Ponadto zestawienie takie będzie uzupełnieniem informacji dostępnych w Internecie oraz książkach, ponieważ wiedza taka nie jest zgromadzona w jednym miejscu.

Zakres badań obejmował opisanie obiektów badań, czyli narzędzi AngularJS, Angular2 oraz BackboneJS. Następnie przeprowadzone zostały analizy złożoności kodu, szybkości generowania widoku, płynności działania aplikacji obciążonej danymi, ilość przesyłanych danych potrzebnych do uruchomienia aplikacji oraz zużycie pamięci zajmowanej przez program. Po zestawieniu wyników wytypowano najlepsze narzędzie.

2. Obiekt badań

Obiektem badań były trzy frameworki frontendowe. AngularJS, Angular2 oraz BackboneJS. Pierwszy został stworzony w celu zmniejszenia nakładów pracy potrzebnych do rozwoju i testowania aplikacji internetowych poprzez wdrożenie wzorca MVC [7]. Drugi jest nowszą wersją AngularJS, wprowadzone zostały ulepszenia i modyfikacje w celu wyeliminowania problemów napotkanych przez programistów. BackboneJS natomiast jest najmniejszym z frameworków. Jego zadaniem jest zwiększenie wydajności aplikacji.

3. Metoda badań

Badania przeprowadzone zostały przy użyciu projektu TodoMVC, który jest zbiorem identycznych aplikacji napisanych w różnych frameworkach do tworzenia warstwy prezentacji. Jako pierwsza przeprowadzona została analiza złożoności kodu. Następnie zmierzono szybkość generowania widoku, płynność działania aplikacji przy obciążeniu danymi, ilość przesyłanych danych potrzebnych do uruchomienia aplikacji oraz zużycie pamięci zajmowanej przez program w zależności od liczby elementów na liście. Zasymulowane zostało obciążenie aplikacji danymi. Lista Todo zawierała kolejno 100, 500 i 1000 elementów. Poniżej znajduje się konfiguracja sprzętowa maszyny, na której wykonano testy:

- Procesor: Intel® Core™ i7-4790K,
- Pamięć: 16GB,
- Dysk: 2000GB + 1000GB,
- Grafika: Intel® HD Graphics 4600, NVIDIA GeForce GTX 750 Ti,
- System operacyjny: Windows 10 Pro 64-bit.

Wszystkie przypadki pomiaru czasu i zużycia pamięci rozpatrzone zostały przy użyciu przeglądarek:

- Google Chrome, wersja 62.0.3202.94 (Oficjalna wersja) (64-bitowa),
- Mozilla Firefox, wersja 57.0.3 (64-bitowa).

Pomiary wykonano przy użyciu narzędzi programistycznych wymienionych w kolejnym punkcie.

3.1. Badane parametry

- **Złożoność kodu** – zestawienie liczby linii jakie zostały napisane do działania programu. Pominięto puste linie, pliki ze stylami, plik ze zdefiniowanymi bibliotekami, folder z narzędziami do budowania oraz folder zawierający testy aplikacji.

- **Szybkość generowania widoku** – czas ładowania obiektowego modelu dokumentu. Pomiary wykonano przy użyciu dodatków doinstalowanych do przeglądarek. W przypadku Google Chrome był to *Page load time*, Mozilla Firefox natomiast *app.telemetry Page Speed Monitor*.

- **Płynność działania przy obciążeniu danymi** – zmiana czasu ładowania obiektowego modelu dokumentu w zależności od liczby elementów na liście Todo.

- **Czas uruchomienia programu** - czas uruchomienia aplikacji od momentu przeładowania strony, do pojawienia się listy.

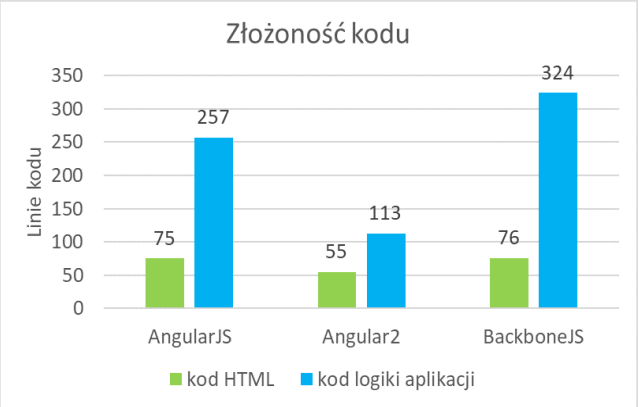
- **Ilość przesyłanych danych** – ilość danych jaka jest potrzebna do uruchomienia aplikacji. Wyniki uzyskano dzięki narzędziu *Network* przeglądarki internetowej.

- **Zużycie zasobów** – ile pamięci zajmowała aplikacja w zależności od obciążenia danymi. W tym przypadku wykorzystano narzędzie *Memory* przeglądarki.

4. Wyniki badań

4.1. Złożoność kodu

Na rysunku 1 zestawiono liczbę linii kodu dla aplikacji napisanej w trzech frameworkach.



Rys. 1. Zestawienie złożoności kodu wybranych frameworków

W tym przypadku faworytem został Angular2. Zawiera on bardzo dużo gotowych funkcjonalności, które ułatwiają i przyspieszają proces tworzenia systemów. Co więcej powoduje to redukcję liczby linii kodu.

4.2. Szybkość generowania widoku oraz płynność działania przy obciążeniu danymi

W tabeli 1 zestawiono średnie czasy generowania widoku dla dwóch przeglądarek przy zmiennym obciążeniu danymi.

Tabela 1. Średnie czasy generowania widoku dla wszystkich pomiarów

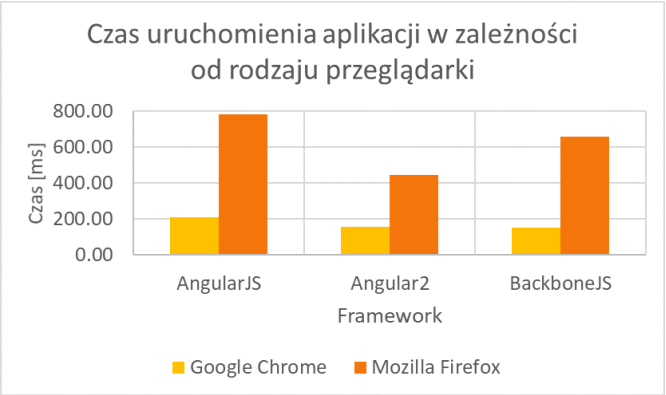
	Liczba elementów	Google Chrome	Mozilla Firefox
AngularJS	100	86,01ms	106,13ms
	500	87,77ms	111,65ms
	1000	92,04ms	119,71ms
Angular2	100	37,38ms	52,04ms
	500	44,73ms	63,02ms
	1000	51,7ms	81,26ms
BackboneJS	100	44,16ms	61,85ms
	500	50,25ms	69,08ms
	1000	62,75ms	99,9ms

Bez względu na rodzaj przeglądarki najdłuższy czas uzyskał AngularJS. Spowodowane jest to sposobem w jaki zaimplementowane zostało dwukierunkowe wiązanie danych w tym frameworku. Najszybszy okazał się Angular2, jednak BackboneJS osiągnął zbliżone rezultaty. Obydwa narzędzia nastawione zostały przez ich twórców na szybkość działania.

Zauważalna jest także duża rozbieżność pomiędzy Google Chrome a Mozilla Firefox. Pierwsza z przeglądarek o wiele szybciej generowała obiektowy model danych. Jednak obciążenie danymi nie miało dużego wpływu na działanie aplikacji.

4.3. Czas uruchomienia aplikacji

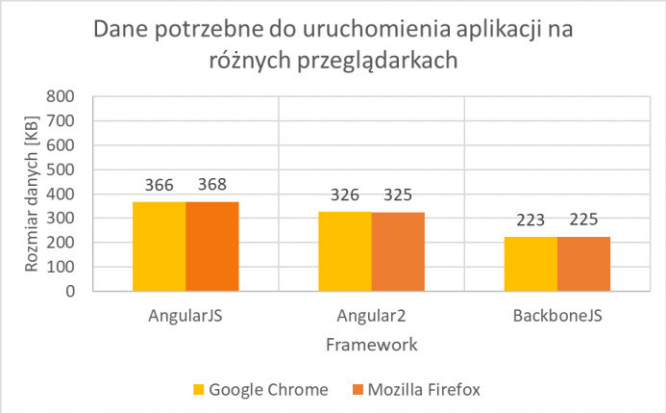
Poniżej zestawiono średnie czasy uruchomienia aplikacji w zależności od rodzaju przeglądarki (Rys. 2). Z łatwością da się także zauważyć różnice między programami napisanymi przy użyciu tych samych narzędzi. Największą odnotowano w przypadku aplikacji korzystających z AngularJS oraz BackboneJS. W przeglądarce Google Chrome programy uruchamiały się prawie cztery razy szybciej niż w Mozilla Firefox. Najmniej odbiegają od siebie wyniki jakie uzyskał Angular2. Framework ten okazał się trzykrotnie szybszy w Google Chrome.



Rys. 2. Czas uruchomienia aplikacji w zależności od rodzaju przeglądarki

4.4. Ilość przesyłanych danych

Rysunek 3 pokazuje jaka ilość danych musiała zostać przesłana w celu uruchomienia aplikacji.



Rys. 3. Dane potrzebne do uruchomienia aplikacji w przeglądarce

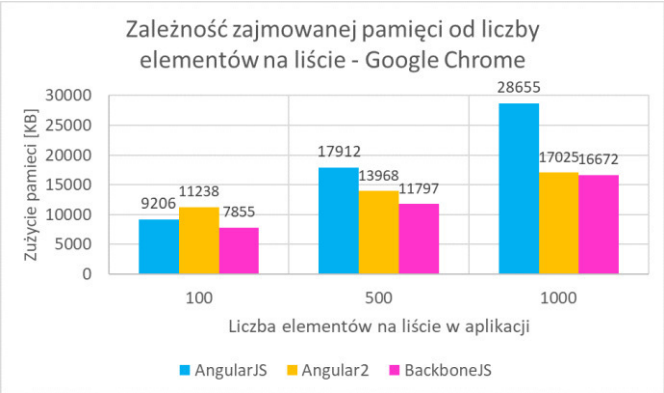
Niezależnie od przeglądarki wyniki były praktycznie identyczne. Nieznaczne różnice wynikać mogą z odmiennego sposobu interpretacji styli CSS, dodatków w aplikacji oraz obrazków przez Google Chrome i Mozilla Firefox.

Aplikacja napisana przy użyciu BackboneJS mimo dużej liczby dodatków jakie są potrzebne do działania systemu, wymagała najmniej danych. Spowodowane jest to niewielkim rozmiarem frameworka. Angular2 dzięki funkcjonalnościom dostarczanym z narzędziem nie wymaga tylu danych co AngularJS.

Twórcy narzędzia Angular2 długo pracowali nad zmniejszeniem liczby ładowanych plików, wersja *Candidate* dla projektu typu „Hello world" pobierała aż 594 kilobajty danych [8].

4.5. Zużycie zasobów

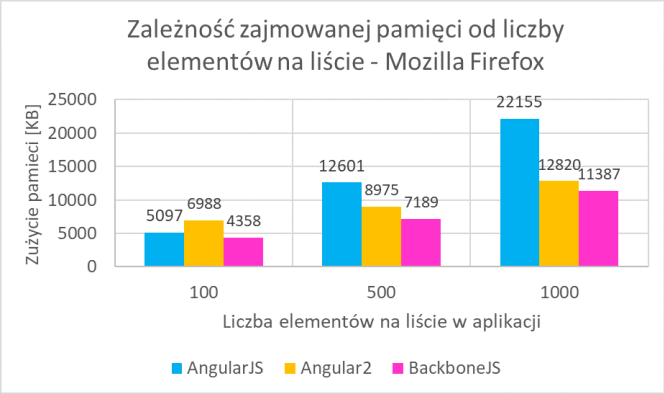
W tym podrozdziale zestawione zostało zużycie pamięci zajmowanej przez aplikacje w zależności od liczby elementów na liście. Wyniki zestawiono na poniższych wykresach (Rys. 4 -5).



Rys. 4. Zależność zajmowanej pamięci od liczby elementów na liście - Google Chrome

W przypadku Google Chrome AngularJS dla stu elementowej listy uzyskał drugi wynik. W kolejnych pomiarach wraz ze wzrostem liczby zadań na liście w aplikacji następował gwałtowny wzrost zajmowanej pamięci.
Dla Angular2 wraz ze wzrostem obciążenia danymi, parametr ten poprawia się. W pierwszym pomiarze framework ten miał najgorszy wynik, dla tysiąca elementów natomiast prawie zrównał się z faworytem.

BackboneJS okazał się najlepszy.



Rys. 5. Zależność zajmowanej pamięci od liczby elementów na liście - Mozilla Firefox

W przeglądarce Mozilla Firefox AngularJS również z drugiego miejsca spadł na ostatnią pozycję.
Angular2 po najgorszym wyniku dla stu elementowej listy uzyskał drugi wynik i do najlepszego narzędzia zabrakło mu jedynie 1,5MB, chociaż wartość ta w przypadku Google Chrome była znacznie mniejsza.
Framework BackboneJS bez względu na liczbę elementów na liście w aplikacji zużywał najmniej pamięci przeglądarki.
Mozilla Firefox uzyskała mniejsze wartości niż Google Chrome, ponieważ każda przeglądarka w inny sposób interpretuje aplikacje i bierze pod uwagę różne elementy.

5. Wnioski

Na podstawie przeprowadzonych analiz można stwierdzić, że Angular2 i BackboneJS uzyskały najlepsze rezultaty.

Pierwszy z nich dostarcza najwięcej wbudowanych funkcjonalności. Umożliwia to szybsze i łatwiejsze pisanie skomplikowanych programów przy użyciu mniejszej ilości kodu. Dzięki temu był w stanie osiągnąć najlepsze czasy generowania obiektowego modelu danych niezależnie od obciążenia aplikacji danymi.
Wielkość narzędzia spowodowała, że framework zajmował najwięcej pamięci w przeglądarce dla stu elementowej listy. Jednak posiada on modularną strukturę, co powoduje, że ładowane są tylko te partie aplikacji, które są potrzebne. Przy przejściu do innych części systemu następuje pobranie kolejnych elementów [9].

Drugi faworyt dzięki swoim niewielkim rozmiarom zajmował najmniej pamięci aplikacji oraz wymagał najmniejszej liczby danych do uruchomienia programu. Co więcej narzędzie osiągnęło bardzo dobry czas uruchomienia aplikacji, ponieważ wspiera mechanizm szablonów, który może być wcześniej ładowany na serwerze. Zwiększa to szybkość strony zwłaszcza na urządzeniach z niską mocą obliczeniową [10].

AngularJS okazał się najsłabszym frameworkiem. Przez sposób w jaki zaimplementowano dwukierunkowe wiązanie danych miał najgorsze czasy generowania widoku. Nie obsługuje on leniwego ładowania danych, co spowalnia proces uruchamiania. Ponadto narzędzie przechowuje dodatkową kopię każdego obiektu, dlatego wraz ze wzrostem obciążenia aplikacji danymi zwiększało się zużycie pamięci.

Aplikacje uruchamiane w przeglądarce Google Chrome działały dużo szybciej. Zaobserwowano mniejsze opóźnienia w generowaniu widoków i uruchamianiu aplikacji niż w przypadku Mozilla Firefox.

Literatura

[1] B. Dybowski, Angular, Backbone czy Ember? Wybieramy framework JavaScript, 20.07.2014, https://www.nafrontendzie.pl/ [30.09.2017].

[2] A. Moorthy, Angular vs AngularJS 2: A detailed comparison, 11.07.2016, https://www.cubettech.com/ [01.10.2017].

[3] M. Fakiolas, I have a new project, should I use AngularJS 1.x or Angular2?, 18.01.2016, https://angularjs-recipes.com/ [01.10.2017].

[4] D. Rathore, Angularjs vs Angular2 | what's the difference ?, 13.08.2016, https://www.dunebook.com [1.10.2017].

[5] F. He, Angular2, next generation UI solutions. 02.04.2017, https://www.linkedin.com/pulse [1.10.2017].

[6] S. Karn, Difference : AngularJS vs. Backbone.js vs. Ember.js, 08.06.2016, , https://www.linkedin.com/pulse [1.10.2017].

[7] A. Kalbarczyk, D. Kalbarczyk, AngularJS. Pierwsze krok, Helion, Gliwice 2015, 15.

[8] M. Miszczyszyn, Wstęp do Angular 2, 03.06.2016, https://typeofweb.com/ [2.11.2017].

[9] Praca zbiorowa: Rangle's Angular Training Book, GitBook, 2017, 297-300.

[10] https://versus.com/pl/angularjs-vs-backbone-js [2.12.2017].

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 8 (2018) 230-234
Wysłane: 2017-12-12
Przyjęte: 2017-12-19

System monitoringu użytkownika wykorzystujący sieci społecznościowe – budowa i analiza możliwości

Sofiia Lahoda *, Marek Miłosz

Politechnika Lubelska, Instytut Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. W artykule przedstawiono rezultaty badań metod, które pomogą tworzyć bazę danych o użytkownikach z wykorzystaniem serwisów internetowych jako podstawowego źródła informacji. Celem badania było pozyskanie danych o użytkownikach z sieci społecznościowej, sprawdzenie możliwości parsera oraz analiza efektywności wykorzystania utworzonej bazy danych. W badaniach zostały wykorzystane metody analizy tekstowej: parsing i scraping. Wyniki zostały przedstawione w postaci wykresów i poddane krytycznej analizie porównawczej.

Słowa kluczowe: parsowanie; scraping; baza danych; sieci społecznościowe; analiza tekstowa

*Autor do korespondencji.

Adres e-mail: lahoda.sof@gmail.com

User monitoring system using of social networks - structure and analysis of the opportunities

Sofiia Lahoda *, Marek Miłosz

Institute of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract. The article presents the results of a study that will help for user to create a end-user parameters database using other web-sites as the primary source. The main goal of the work is making analysis to obtain information about the user of social networks. Next goal is analysis of the gathering data and the possibility of their use. The experiment was done using two methods of text analysis: parsing and scraping. The results are presented graphically and critical compared to each other.

Keywords: parsing; scraping; data bases; social networking; text analysis

*Corresponding author.

E-mail address: lahoda.sof@gmail.com

1. Wstęp

W obecnej sytuacji, oprogramowanie, które jest niezbędne do wyszukiwania i analizowania informacji w obszarze sieci społecznościowych jest niezbędnym narzędziem dla działów analitycznych wielu firm w kraju - od małych przedsiębiorstw do dużych korporacji. Wyłączając bowiem ze swojej działalności użytkowników portali społecznościowych, takich jak "Facebook", "Instagram" czy "Twitter", firma traci wielu potencjalnych klientów, a co za tym idzie, traci potencjalne zyski. Walcząc o rynek firmy powinny walczyć także o członków sieci społecznościowych.

Informacje, które użytkownicy udostępniają w sieciach społecznościowych, najczęściej zawierają dane, w których zawarte są opinie na temat marki, produktu czy usługi. Używając danych z sieci społecznościowych, użytkownik otrzymuje możliwość wyszukiwania interesujących go informacji, a także zajmuje się aktualizacją danych, dokonuje selektywnego monitorowania interesującej informacji i jej pełnej analizy.

Często producent i jego menedżer nie wie, w którym kierunku ma iść, jak działać, aby osiągnąć swoje cele, czego potrzebuje klient, z którego miasta jest najwięcej kupujących itp.

Aby rozwiązać podobne problemy, konieczne jest systematyczne prowadzenie badań mających na celu analizę obiektywnej sytuacji społecznej i gospodarczej w danym obszarze tematycznym lub po prostu badań statystycznych dla np. miasta czy kraju.

Na wymagania takiej analizy całkowicie odpowiada monitoring społeczny jako metoda monitoringu sieci społecznościowych. Dużą zaletą monitoringu, w porównaniu z jednorazowymi badaniami, jest jego zdolność do systematycznego zwiększania i integrowania niezbędnych danych w szerokim zakresie standardowych wskaźników społecznych i tworzenia na tej podstawie stale aktualizowanych zbiorów danych. Pozyskiwanie danych jest permanentne i właśnie dlatego rośnie popularność tej metody w praktyce analizy socjologicznej.

Jednym z narzędzi, za pomocą których można przeprowadzić takie monitorowanie, są wspomniane wcześniej sieci społecznościowe. Za ich pomocą można uzyskać różnorodne informacje do analizy. Monitorowanie

obejmuje badanie faktów, zdarzeń i wyników związanych z przedmiotem obserwacji. Taką analizę sieci społecznościowej można wykonać za pomocą specjalnych aplikacji, bazujących na użyciu metod analizy składniowej danych.

2. Cel, teza i zakres artykułu

Celem artykułu jest zaprezentowanie rezultatów badań o metodach pozyskania informacji o użytkowniku z sieci społecznościowych do ich wykorzystania w różnych obszarach. Ocenione będą także korzyści z wykorzystania danych monitorowania z sieci społecznościowych i potencjalne obszary ich zastosowania.

Dla celów badań sformułowano następujące hipotezy:

- Hipoteza 1: Przy pomocy parsowania jest możliwe monitorowanie użytkowników sieci społecznościowej.
- Hipoteza 2: Monitorowanie użytkowników w sieci społecznościowej pomaga stworzyć bazę danych o użytkownikach sieci społecznościowej, którą można wykorzystać w różnych obszarach.

Tezą danej pracy jest: Monitoring użytkowników sieci społecznościowej pomaga uzyskać informacje, które nadają się do analizy biznesowej i statystycznej.

Zakres artykułu obejmuje:

- badanie literatury na temat analizy danych;
- analizowanie pracy parserów – analizatorów składniowych;
- analiza informacji na temat tworzenia parsera w języku programowania php;
- opracowanie i opis działania stworzonego programu do monitorowania użytkowników sieci społecznościowej;
- użycie stworzonego programu oraz pozyskanie i rejestrowanie danych użytkownika w bazie danych;
- przeprowadzenie eksperymentu na dwóch grupach w celu porównania skuteczności dostępności danych uzyskanych z sieci społecznościowych;
- wnioski.

3. Analiza składniowy danych

Analiza składniowa (ang. parsing) - jest sprawdzaniem poprawności składniowej programu lub modułu wykonywanym przez translator (kompilator) i polega na dokonaniu rozbioru gramatycznego analizowanej jednostki tekstu [1].

Zadanie analizy składniowej można rozpatrywać jako potwierdzenie lub zaprzeczenie zgodności tekstu na wejściu z daną gramatyką formalną, a w przypadku stwierdzenia zgodności — także wygenerowanie wszystkich drzew możliwych rozbiorów gramatycznych tekstu. Ogólnie takich rozbiorów może okazać się dużo, nie tylko z powodu znaczących różnic w strukturyzacji, ale także z powodu wielu możliwości grupowania struktur [2]. Z tej przyczyny, nawet jeśli zadanie analizy składniowej może być zrobione ręcznie, stopień złożoności języka naturalnego znajdujący odzwierciedlenie w złożoności opisującej jego gramatyki sprawia, że tylko analiza automatyczna ujawnia większą część potencjalnych niejednoznaczności rozbioru.

Analizator składniowy lub parser jest narzędziem, które służy do analizy przetwarzanych danych oraz ewentualnie wywołuje określone akcje. Analizator leksykalny lub skaner analizuje znaki i przekazuje te do parsera w postaci tokenów. Istnieje dużo narzędzi do tworzenia takich analizatorów składniowych oraz leksykalnych.

Drzewo rozbioru jest jedną z podstaw parsowania. Wierzchołki drzewa rozbioru oznacza [3]: albo symbolem terminalnym, albo symbolem ε, albo kategoriami syntaktycznymi. Tak zwane liście etykietuje się jedynie symbolami terminalnymi albo symbolem ε. Wierzchołki wewnętrzne są etykietowane tylko kategoriami syntaktycznymi. Każdy wewnętrzny wierzchołek reprezentuje zastosowanie otrzymanych danych.

Z tego wynika, że kategoria syntaktyczna, która etykietuje wierzchołek stanowi część nagłówka danych. Każde drzewo rozbioru proponuje ciąg symboli terminalnych s, który można nazwać wynikiem drzewa [3].

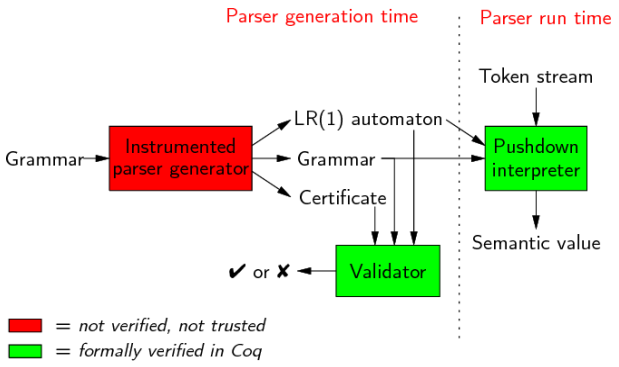
Ciąg ten składa się z tak zwanych etykiet liści drzewa ułożonych w specjalnej kolejności, od lewej do prawej strony. W przypadku, gdy drzewo posiada tylko jeden wierzchołek, on jest etykietowany symbolem terminalnym lub symbolem ε, ponieważ on jest liściem.

Gdy drzewo posiada więcej niż jeden wierzchołek, korzeń etykietuje się kategorią syntaksyczną, dlatego że korzeń drzewa, który posiada dwa lub więcej wierzchołków, zawsze będzie wierzchołkiem wewnętrznym. W tej kategorii syntaktycznej zawsze będzie wśród ciągów znaków również wynik drzewa.

Zazwyczaj wynikiem analizy składniowej będzie struktura składniowa zdania, zaproponowana albo w postaci drzewa komponentów, albo jako drzewo zależności, lub jako kombinacja pierwszej i drugiej przedstawianej metody [4].

Jakie są typy algorytmów do tworzenia parserów? Nie ma ich zbyt dużo. Pierwszy to rekurencyjny analizator syntaksyczny, który analizę zaczyna od znaku początkowego i kontynuuje do otrzymania wymaganej liczby tokenów [5]. On odpowiada za parser LL (Leftmost derivation). Drugim jest parser oddolny - zaczyna odzyskiwane z prawej części, zaczynając od tokenów i kończy na symbolu startowym. On dzieli się na parser LR (Rightmost derivation) (przykład na rysunku 1) oraz parser GLR (Generalized Left-to-right Rightmost derivation parser).

Parsowanie stron jest oczywiście analizą danych publikowanych na stronach internetowych. Tekst stron internetowych jest hierarchicznym zbiorem danych, który funkcjonuje razem z językiem, na przykład, angielskim i komputerowym. Język ludzki oferuje informacje, wiedzę. Języki komputerowe, na przykład - html, JavaScript, css - określają sposób wyświetlania informacji na monitorze [7].



Rys. 1. Proces parsera LR [6]

4. Eksperyment

W czasach współczesnych bardzo popularnym wśród programistów jest połączenie PHP i MySQL. PHP jest językiem programowania dość łatwym w użyciu, a jednocześnie bardzo elatycznym. Przy jego pomocy można także tworzyć połączenie z bazą danych MySQL. Pomaga on deweloperom tworzyć dynamiczne elementy na stronach.

W związku z powyższym, do napisania własnego parsera był wybrany język PHP. Spośród innych wariantów on idealnie odpowiadał wszystkim potrzebom.

Każda strona internetowa składa się z kodu HTML z dodatkowymi elementami technologicznymi (JavaScript, PHP, CSS i inne). Zadanie pozyskania informacji w tym przypadku polega na znalezieniu i wyodrębnieniu wzorców tekstowych, czyli sekwencji istniejących symboli, które spełniają zasady podane przez gramatykę [8].

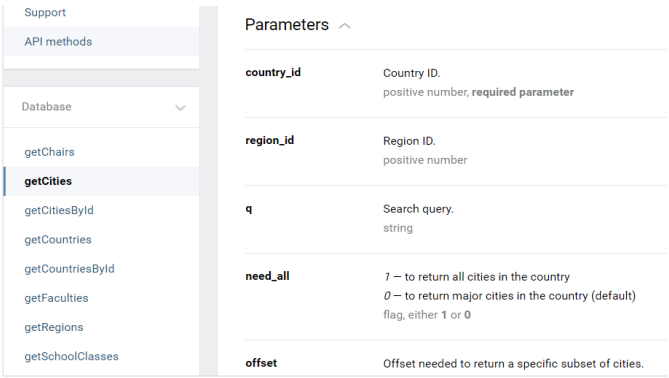
Parsowanie strony HTML to proces, który można podzielić na trzy etapy:

1) Uzyskanie kodu źródłowego z potrzebnej strony. W tym celu w różnych językach dostępne są odpowiednie metody. Na przykład PHP używa biblioteki cURL lub wbudowanej funkcji, takiej jak file_get_contents [9].

2) Otrzymanie niezbędnych danych z kodu HTML. Po otrzymaniu strony należy ją przetworzyć, w tym przypadku - oddzielić zwykły tekst od języka znaczników hipertekstowych, zbudować hierarchiczne drzewo struktury dokumentu, poprawnie zareagować na nieprawidłowy kod, wyodrębnić z tej strony dokładnie te informacje, dla pozyskania których wykonano całą aplikacje. Oczywiście można również użyć wyrażeń regularnych, ale łatwiejszym sposobem jest używanie biblioteki, która w tym specjalizuje się.

3) Zapisywanie wyniku. Po bezpiecznym przetworzeniu danych ze strony, trzeba zapisać je w określonym formacie. Otrzymane dane po parsowaniu są zwykle zapisywane do bazy danych, ale oprócz tego możliwe są i inne opcje. Czasami można zapisywać na przykład do pliku CSV (ang. Comma-Separated Values) lub w postaci hierarchicznej struktury JSON (ang. JavaScript Object Notation).

Dla napisania własnego parsera trzeba wykonać powyższe kroki. Parsowanie w artykule było wykonane dla sieci społecznościowej "Vkontakte". Stworzona aplikacja pobierała dane użytkowników sieci społecznościowej i zapisywała je do bazy danych. Do uzyskania danych było użyte API sieci "Vkontakte", które pomaga filtrować użytkowników odpowiednio wg miejsca zamieszkania, płci, wieku i innych kryteriów. Do pobierania danych była użyta sieciowa biblioteka cURL, także za jej pomocą dane były oddzielone od całego kodu HTML. Pozyskane dane są przypisywane do odpowiednich kategorii i automatycznie zapamiętywane w bazie danych. Aplikacja ma możliwość ściągać dane osób, które mieszkają wyłącznie w Warszawie. Dodatkowa funkcja to pobieranie danych z grup. Otrzymane dane można wykorzystać, na przykład, w reklamie [10]. Po użyciu metody GroupSearch API sieci „Vkontaktie" nie ma potrzeby pisać własnej funkcji, która będzie odpowiadać za ten proces (przykład na rysunku 2). Oprócz tego, aplikacja ma możliwość automatycznie publikować wiadomości do potrzebnych grup. Taką możliwość nadaje skrypt access_token.



Rys. 2. Metody wykorzystane z API sieci społecznościowej „Vkontaktie"

5. Rezultaty badań

Po napisaniu aplikacji, można użyć jej funkcjonalności i zweryfikować postawione hipotezy. Czy, jak było umówione wyżej, jest możliwość pozyskać dane użytkowników z sieci społecznościowej, i czy są z tego korzyści?

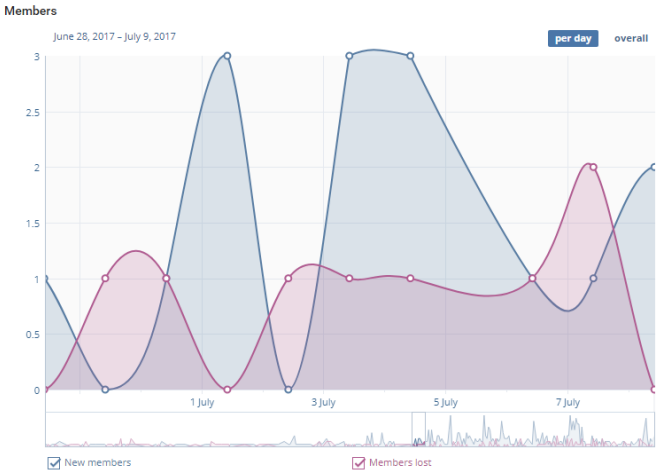
Na początku badania stworzone były dwie grupy użytkowników. Główny element różnicującym grupy są parametry członków zapraszanych do obydwu grup. Uczestnicy pierwszej grupy byli wybrani losowo. Dla określenia użytkowników zapraszanych do drugiej grupy był użyty parser. Aplikacja wyszukała użytkowników i ze wszystkich użytkowników, wybrała wyłącznie mieszkańców Warszawy, pobrała ich dane i zapisała do tabeli. Otrzymane dane były pobierane w postaci XML i zawierały różne pola jak pokazano na rysunku 3.

Analogicznie była stworzona baza danych użytkowników, posiadających wpisy ze wspólną tematyką. Wszystkie otrzymane dane użytkowników były zapisane do bazy danych. Każdy z nich był zaproszony do grupy społecznościowej. Dzięki danym statystycznym,

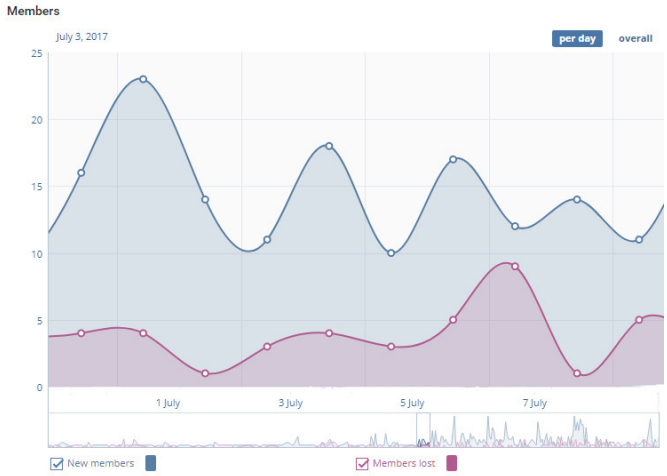
udostępnianym w wybranej sieci społecznościowej, można było porównać postęp rozwoju dwóch grup.

	A	B	C	D	E	F	G	H
1	Nazwisko	Imie	Link	Płeć	Kraj	Miasto	Data urod	Wiek
2	Zheka	Kocherov	http://vk.	m	Polska	Warszawa		
3	Irina	Porada	http://vk.	k	Polska	Warszawa	########	31
4	Sanya	Kitayev	http://vk.	m	Polska	Warszawa	08.янв	0
5	Maxim	Chornoba	http://vk.	m	Polska	Warszawa		
6	Mikhail	Baytsar	http://vk.	m	Polska	Warszawa	########	48
7	Vadim	Muzichuk	http://vk.	m	Polska	Warszawa	06.мар	0
8	Vitaly	Easy-Goin	http://vk.	m	Polska	Warszawa	19.дек	0

Rys. 3. Baza danych użytkowników cieci „Vkontakte" z Warszawy



Rys. 4. Wykres dodawania użytkowników do grupy A

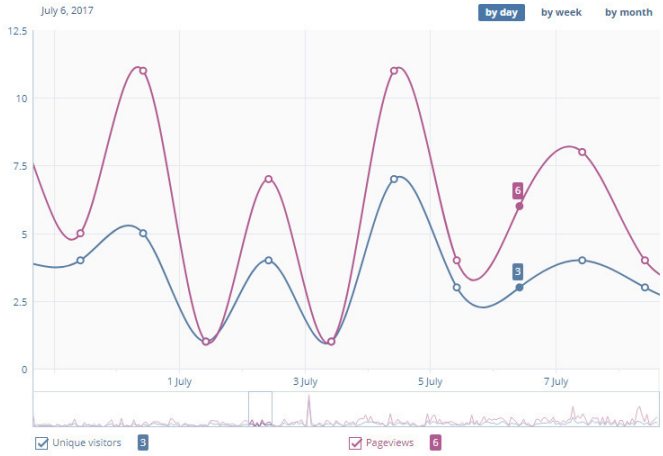


Rys. 5. Wykres dodawania użytkowników do grupy B

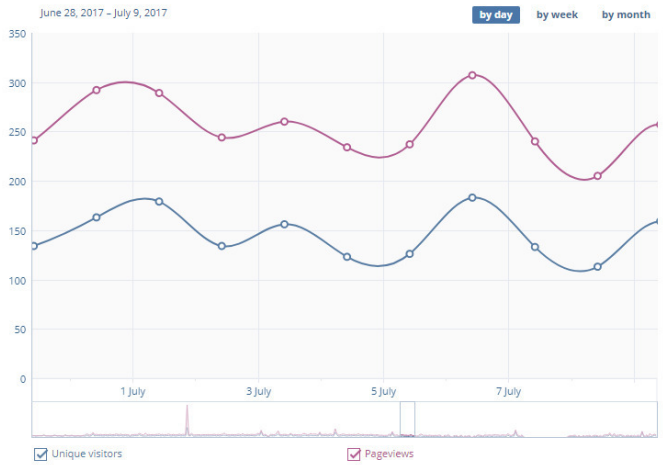
Na rysunkach 4 i 5 jest pokazana liczba nowych użytkowników obydwu grup. Dziennie do grupy, która nie była opracowana parserem tj. do grupy A, średnio dołączało się około 3 osób dziennie. Do grupy B, użytkownikami której były zainteresowane osoby, dołączało się koło 15 osób dziennie. Z takich wykresów można wywnioskować, że dołączenie do grupy osób wyselekcjonowanych na podstawie miasta zamieszkania i zainteresowań jest bardziej efektywne. Taki monitoring osób przez sieć społecznościową daje administratorowi możliwość odszukanie bardziej zainteresowanych danym tematem (w tym przypadku zainteresowań i miejsca) członków sieci społecznościowej.

Wiadomo, że duża liczba użytkowników to jeszcze nie sukces. Dlatego sprawdzono na ile użytkownicy są zainteresowani. Na rysunku 6 znajduje się wykres z pierwszej grupy A, która nie zainteresowała użytkowników. Oglądalność strony jest bardzo niska, użytkownicy nie są zainteresowani tym, co publikuje się w grupie. Wykres 6 pokazuje statystykę, otrzymaną po miesiącu utworzenia grupy przy 30 członkach grupy.

Przy porównaniu wykresów 6 i 7 można powiedzieć, że liczba wejść na stronę dwóch grup dużo różni się. W tym przypadku, liczba ta w grupie A równa się 5, a w grupie B, która była przygotowana za pomocą parsera – średnio 300 dziennie.



Rys. 6. Wykres wejść użytkowników do grupy A



Rys. 7. Oglądania wykres wejść użytkowników do grupy B

6. Wnioski

Celem danego artykuła było sprawdzanie hipotezy, czy monitorowanie użytkowników w sieci społecznościowej pomaga stworzyć bazę danych, która zawiera dane o wyselekcjonowanych użytkownikach sieci

społecznościowej do wykorzystania w różnych obszarach. Hipoteza 1 została zatem całkowicie udowodniona.

Hipoteza 2 również jest prawdziwa. Osoby wyselekcjonowane na podstawie danych pozyskanych z sieci społecznościowej były bardziej aktywne (i zainteresowane tematem) niźli te przypadkowe.

Wykorzystując informacje pozyskane ze stron członków grupy społecznościowej, można stworzyć bazę danych klientów ukierunkowanych na określony cel. Wynika to z faktu, że pozyskać można nie tylko dane o użytkowniku, ale też o jego zainteresowaniach.

Trzeba pamiętać, że dane mogą być zastrzeżone przez stronę internetową, dlatego trzeba sprawdzić reguły i wymagania strony przed jej parsowaniem.

Tworzenie parsera nie jest zbyt ciężkim zadaniem dla programisty, ponieważ istnieje dużo bibliotek oraz API, które ułatwiają ten proces.

Istnieje duża liczba parserów, których użycie i wykorzystanie rezultatów ich pracy umożliwia wyselekcjonowanie grupy docelowej w działaniach marketingowych.

Hipotezy zostały udowodnione i potwierdzone przy pomocy własnej aplikacji.

Literatura

[1] M. Collins, J. Ha, E. Brill, L. Ramshaw, C. Tillmann, A Statistical Parser for Czech. In Proceedings of ACL, University of Maryland, College Park, USA (1999), 505-512.

[2] M. Collins, Head-Driven Statistical Models for Natural Language Parsing. PhD thesis, University of Pennsylvania, 1999.

[3] https://www.parser.ru/ [28.04.2016]

[4] Hausser R.: Computation of Language, Springer-Verlag, 1989.

[5] http://myblaze.ru/chto-takoe-parser-grabber/ [4.01.2015]

[6] http://gallium.inria.fr/blog/verifying-a-parser-for-a-c-compiler/ [24.10.2012]

[7] Srinivas B., Doran C., Hockey B., Sarkar A.: Gramar & Parser Evaluation in the XTAG Project, Proceedings of the 1st International Conference on Language Resources and Evaluation. Granda, 1998.

[8] Daniel D. K., Temperley D., Parsing English with a Link Grammar, 1991.

[9] Stump G.T.: Morphological and syntactic paradigms: arguments for a theory of paradigm linkage, Yearbook of Morphology (2001), 147-80.

[10] http://excelvba.ru/programmes/Parser [14.11.2017]

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 8 (2018) 235-240
Wysłane: 2018-01-30
Przyjęte: 2018-02-11

Porównanie technologii mapowania obiektowo-relacyjnego danych w frameworku Symfony 3

Karol Sawłuk*, Marek Miłosz

Politechnika Lubelska, Instytut Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. W artykule przedstawiono rezultaty analizy porównawczej technologii mapowania obiektowo-relacyjnego w frameworku Symfony 3: Doctrine i Propel. Analizę przeprowadzono pod kątem szybkości wykonywania skryptu oraz zużycia pamięci podczas operacji na bazie danych. Analiza pozwoliła wskazać technologię o szybszych i wydajniejszych algorytmach. Technologia Doctrine jest nawet do trzech razy szybsza niż Propel.

Słowa kluczowe: ORM; php; symfony 3; SZRB; doctrine; propel

*Autor do korespondencji.

Adres e-mail: karol.sawluk@gmail.com

Comparison of object-relational data mapping technology in Symfony 3 framework

Karol Sawłuk*, Marek Miłosz

Institute of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract. The article presents the results of a comparative analysis of object-relation mapping technologies in the framework Symfony 3: Doctrine and Propel. The analysis was performed in terms of script execution speed and memory usage during database operations. The analysis allowed to identify the technology with faster and more efficient algorithms. Doctrine is up to three times faster than Propel.

Keywords: ORM; php; symfony 3; DBMS; doctrine; propel

*Corresponding author.

E-mail address: karol.sawluk@gmail.com

1. Wstęp

Gromadzenie informacji w dzisiejszych czasach jest czymś powszechnym. Prawie każdy system informatyczny, nie tylko webowy, korzysta z relacyjnych baz danych. Zdecydowana większość aplikacji posiada i korzysta z baz danych w sposób obiektowy - zaczynając od prostych stron firmowych i kończąc na zaawansowanych systemach zarządzania przedsiębiorstwami. Dzięki tej technice programowania, można konwertować dane między niezgodnymi systemami przy użyciu języków programowania obiektowego. Tworzy to w efekcie "wirtualną bazę danych", która może być używana w języku programowania, która jest najczęściej zwaną technologią ORM (ang. Object-relational mapping).

Mapowanie obiektowo-relacyjne (ORM) w informatyce jest techniką programowania służącą do konwersji danych pomiędzy niekompatybilnymi systemami używającymi języków programowania obiektowego i systemami zarządzania relacyjnymi bazami danych. Dostępne są darmowe i komercyjne pakiety wykonujące mapowanie obiektowo-relacyjne, które często są zawarte w podstawie wykorzystywanego frameworka. Symfony 3 posiada do wyboru dwie technologie ORM: Doctrine i Propel.

Symfony 3 jest frameworkiem, czyli abstrakcją, w której oprogramowanie dostarcza ogólne funkcjonalności i może być selektywnie zmieniane za pomocą dodatkowego kodu użytkownika, dostarczając w ten sposób specyficznego oprogramowania dla aplikacji [1]. Symfony 3 bazuje na wzorcu projektowym MVC (Model-View-Controller), który jest podzielony na trzy części [2]:

- Model - przechowuje dane, które są pobierane zgodnie z poleceniami kontrolera i wyświetlane w widoku.
- Widok - generuje nowe dane wyjściowe dla użytkownika w oparciu o zmiany w modelu.
- Kontroler - może wysyłać polecenia do modelu w celu aktualizacji stanu modelu (np. edytowania dokumentu).

Celem artykułu jest przedstawienie rezultatów testów na relacyjnej bazie danych z użyciem Doctrine i Propel, które mają określić, który ORM jest wydajniejszy.

Teza zawiera się w stwierdzeniu, że technologia Doctrine jest szybsza niż Propel.

2. Technologie mapowania obiektowo-relacyjnego

Wykorzystanie technologii mapowania obiektowo-relacyjnego w procesie tworzenia aplikacji internetowych jest teraz standardem. Kiedy system wymaga pobrania informacji z bazy danych, realizowana jest konkretna sekwencja: nawiązanie połączenia z bazą danych, wysyłanie zapytania SQL, otrzymanie wyniku zapytania oraz zamknięcie połączenia. Tradycyjne pisanie zapytań do bazy, może prowadzić do niejednego problemu np.: zmiany technologii baz danych (MySQL na PostgreSQL) czy zmiany w strukturze baz danych, które wymagają poprawy zapytań w wielu miejscach. Technologia ORM automatyzuje procedurę manipulowania danymi z bazy danych przy użyciu paradygmatu zorientowanego obiektowo [7].

Zautomatyzowanie procesu, pozwoli na rozwiązanie wyżej wymienionych problemów.

2.1. Doctrine

Doctrine to zestaw bibliotek PHP, skupiających się przede wszystkim na trwałości usług i relacyjnych funkcjonalnościach. Jest domyślnie używany w Symfony 3. Doctrine bazuje na wzorcu Data Mapper, który oddziela atrybuty obiektów od pól tabeli, które są w nich utrzymywane. Istotą wzorcu Data Mapper jest klasa, która odwzorowuje lub tłumaczy atrybuty obiektów domeny i/lub metody na pola tabeli bazy danych i odwrotnie. Zadaniem tego wzorca jest zarówno przedstawianie informacji, tworzenie nowych obiektów domeny w oparciu o informacje w bazie danych i aktualizowanie lub usuwanie informacji w bazie danych przy użyciu informacji z obiektów domeny [3].

Model biblioteki Doctrine oparty jest na mechanizmie encji. Encja jest lekkim, trwałym obiektem domeny. Encja opisuje powiązane ze sobą elementy (np. typy danych) wewnątrz modelu biznesowego aplikacji. Model Entity składa się z typów podmiotów i określa związki, które mogą istnieć między przypadkami tych typów podmiotów [9] [11].

W Doctrine relacje między klasami obiektów określa się poprzez asocjacje. Asocjacje umożliwiają wystąpienie jednej instancji obiektu, powodując kolejną czynność w jego imieniu. Ten związek jest strukturalny, ponieważ określa, że obiekty jednego rodzaju są połączone z obiektami innego i nie reprezentują tego samego zachowania [4]. Zamiast pracować z obcymi kluczami w kodzie, programista pracuje z odwołaniami do obiektów, a Doctrine przekonwertuje te odniesienia do kluczy obcych. Odniesienie do pojedynczego obiektu jest reprezentowane przez klucz obcy. Zbiór obiektów reprezentowany jest przez wiele obcych kluczy, wskazując na obiekt posiadający kolekcję.

Asocjacje dzielą się na cztery typy [5]:

- OneToMany - jedna instancja obecnego obiektu ma wiele instancji (odwołań) do odesłanej encji.
- ManyToOne - wiele wystąpień obecnej encji odnosi się do jednego wystąpienia odesłanej encji.
- OneToOne - jedna encja bieżącego podmiotu odnosi się do jednego wystąpienia odesłanej encji.
- ManyToMany - wiele wystąpień obecnej encji odnosi się do wielu wystąpień odesłanej encji.

2.2. Propel

Propel jest wolnym, otwartym źródłem na licencji MIT (ang. Massachusetts Institute of Technology) obiektowo-relacyjnym narzędziem mapującym napisanym w PHP. Jest to również integralna część systemu Symfony [12].

Propel bazuje na wzorcu Active Record, który jest najprostszym wzorcem projektowania bazy danych. Istotą Active Record jest model domeny (ang. Domain Model), w którym klasy ściśle pasują do struktury rekordu bazy danych. Każdy aktywny rekord jest odpowiedzialny za zapisywanie i ładowanie do bazy danych, a także za logikę domeny, która działa na danych. Może to być całość logiki aplikacji, lub część tej logiki, która jest przechowywana w skryptach transakcyjnych (ang. Transaction Scripts) ze wspólnym kodem zorientowanym na dane. Struktura danych Active Record powinna dokładnie odpowiadać strukturze bazy danych - jedno pole w klasie dla każdej kolumny w tabeli [7].

Propel generuje inteligentne klasy Active Record w oparciu o definicję schematów tabel. Obiekty Active Record oferują potężne API do intuicyjnego zarządzania danymi bazy danych. Dla każdej tabeli znajdującej się w schemacie XML, Propel generuje jedną klasę Active Record - nazywaną również klasą Model. Przykłady klas Active Record reprezentują pojedynczy wiersz z bazy danych, zgodnie z regułami wzoru projektowania [7]. To ułatwia tworzenie, edytowanie, wstawianie lub usuwanie pojedynczego wiersza.

3. Metoda badawcza

Celem pracy badawczej było sprawdzenie postawionej następującej tezy:

Technologia Doctrine jest szybsza niż Propel.

Eksperyment polegał na przeprowadzeniu badań na dwóch aplikacjach testowych napisanych w frameworku Symfony 3 - jedna dla Doctrine ORM, druga dla Propel ORM. Aplikacje testowe mają za zadanie zrealizować te same scenariusze testowe. Każdy scenariusz, realizowany w Doctrine i Propel, pozwala na odmierzenie czasu w milisekundach jaki potrzebuje aplikacja w celu wykonania zadania oraz zmierzenie ilości wykorzystywanej pamięci w megabajtach. Opisy scenariuszy zostały przedstawione w tabeli 1.

Tabela 1. Opis scenariuszy testowych

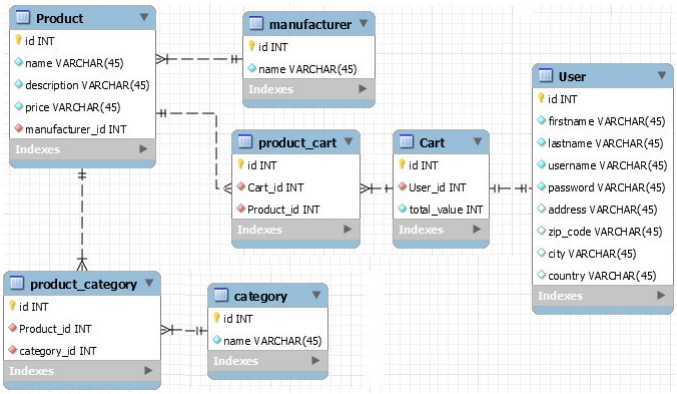
Lp	Opis
1.	Zapis 50, 100, 1000 i 10000 rekordów do relacyjnej bazy danych poprzez utworzenie obiektu reprezentującego poszczególny rekord.
2.	Zapis 50, 100, 1000 i 10000 rekordów powiązanych relacyjnie z trzema obiektami do relacyjnej bazy danych poprzez utworzenie obiektów reprezentujących poszczególny rekord.
3.	Odczyt 50, 100, 1000 i 10000 rekordów z relacyjnej bazy danych.
4.	Odczyt 50, 100, 1000 i 10000 rekordów powiązanych relacyjnie z trzema obiektami z relacyjnej bazy danych.

Do testów obu aplikacji została użyta ta sama baza danych, stworzona na potrzebę zrealizowania testów. Schemat bazy został przedstawiony na rysunku 1. Schemat bazy danych stworzony w programie MySQL WorkBench został przedstawiony na rysunku 1. Testowa baza danych obejmuje tabele:

- User - przechowuje informacje dotyczące użytkowników,
- Cart - pozwala na dodanie produktów do koszyka,
- Product - informacje na temat produktów,
- ProductCategory - tabela łącząca kategorie z produktem,
- Category - zawiera informacje na temat kategorii,
- Manufacturer - zawiera informacje o producencie.

Do pomiaru pamięci oraz czasu, w aplikacji używającej Doctrine, wykorzystano klasę do debugowania (DebugStack), która mierzy czas w mikrosekundach wykonanego skryptu.

W aplikacji z Propel, niestety nie ma tak rozbudowanego narzędzia do debugowania, więc użyto funkcji microtime(), która mierzy czas w mikrosekundach.



Rys. 1. Schemat bazy danych użytej do testów

Obie metody bazują na tej samej funkcji (microtime()), więc wyniki są mierzone w ten sam sposób. Do pomiaru zużytej pamięci poprzez pojedynczy skrypt użyto metody memory_get_usage(), która zwraca w bajtach wielkość pamięci zaalokowanej skryptu.

Opis środowiska testowego na którym zostały przeprowadzone testy, znajduje się w tabeli 2.

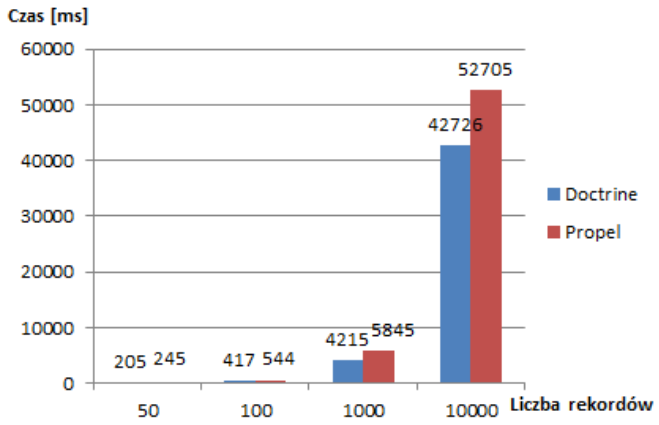
Tabela 2. Dane środowiska testowego

Nazwa	Wartość
System operacyjny	Windows 10
Procesor	Intel i5 4440 3.1GHz
Pamięć RAM	8GB DDR3 1600MHz
Serwer Apache	2.4.27
MySQL	5.7.19
PHP	5.6.30

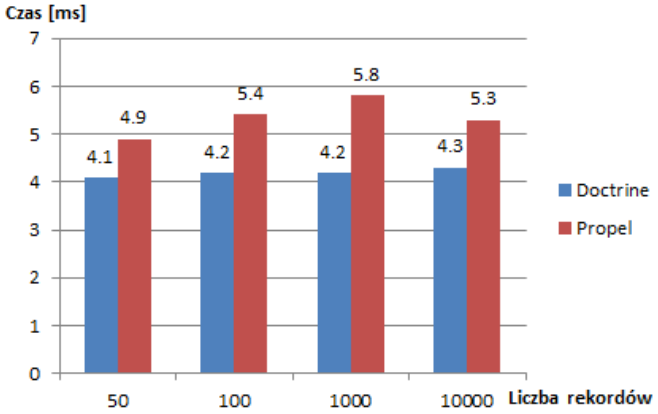
4. Rezultaty badań

- **Scenariusz 1**

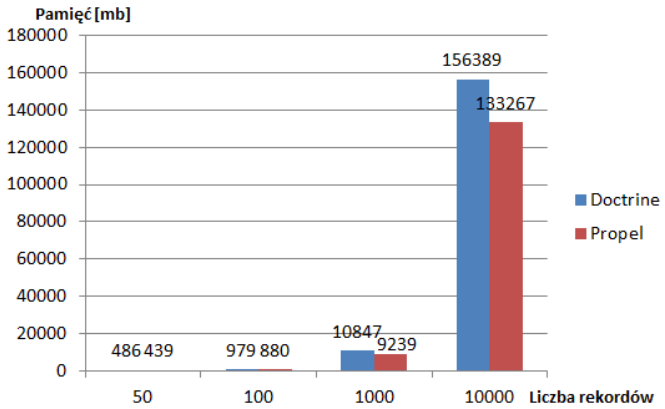
Pierwszy scenariusz polegał na zapisie 50, 100 oraz 1000 rekordów niepowiązanych do relacyjnej bazy danych.



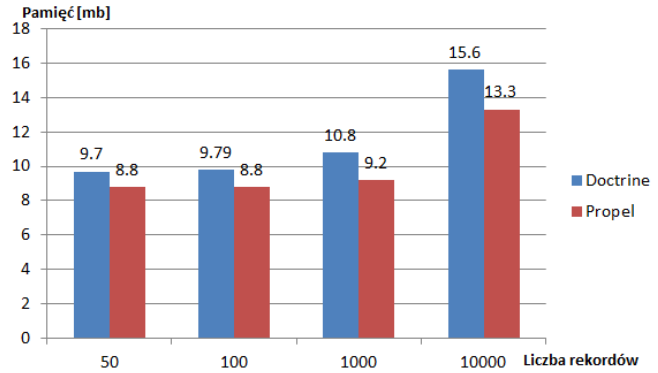
Rys. 2. Czas wykonywania skryptu w milisekundach



Rys. 3. Średni czas wykonywania skryptu w milisekundach



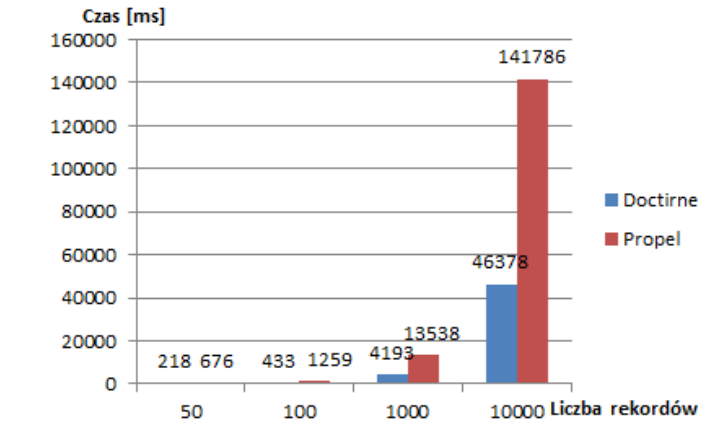
Rys. 4. Całkowite zużycie pamięci w megabajtach



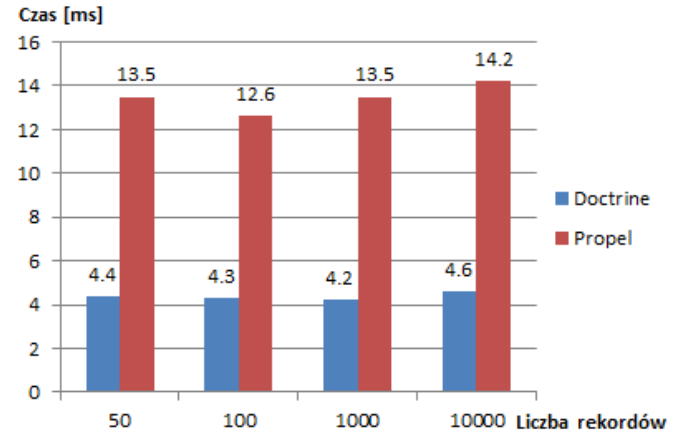
Rys. 5. Średnie zużycie pamięci w megabajtach

- **Scenariusz 2**

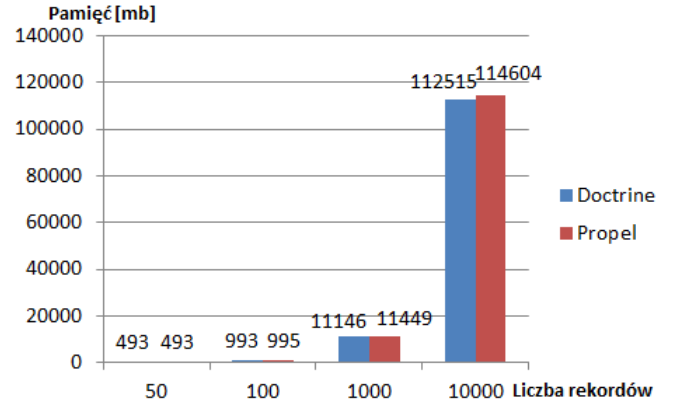
Drugi scenariusz polegał na zapisie 50, 100 oraz 1000 rekordów powiązanych do relacyjnej bazy danych.



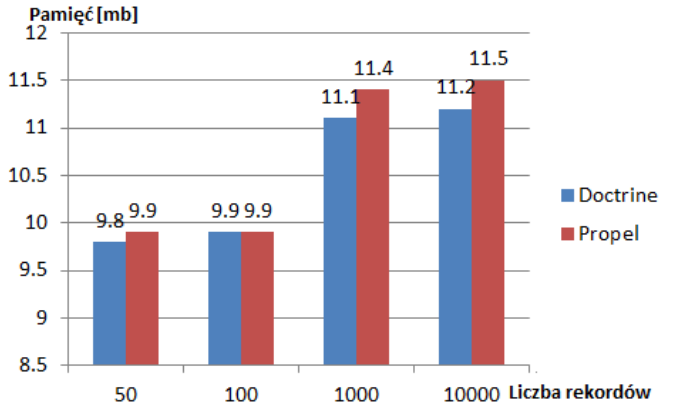
Rys. 6. Czas wykonywania skryptu w milisekundach



Rys. 7. Średni czas wykonywania skryptu w milisekundach



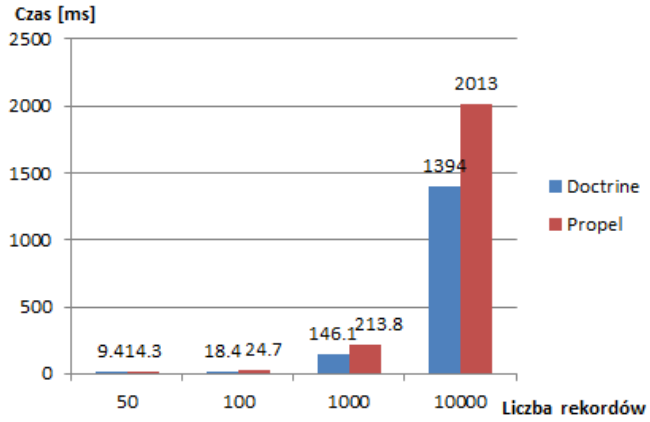
Rys. 8. Całkowite zużycie pamięci w megabajtach



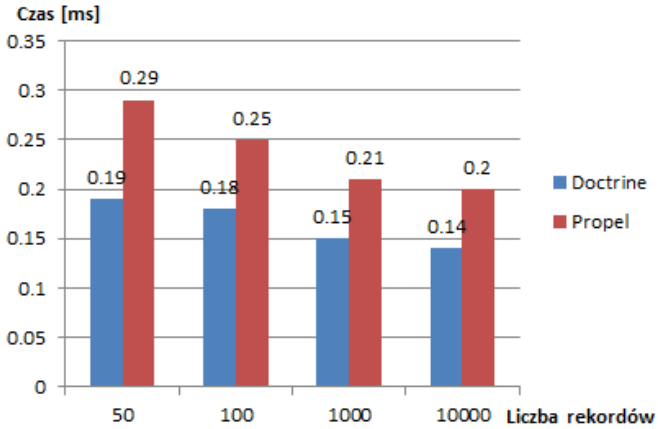
Rys. 9. Średnie zużycie pamięci w megabajtach

- **Scenariusz 3**

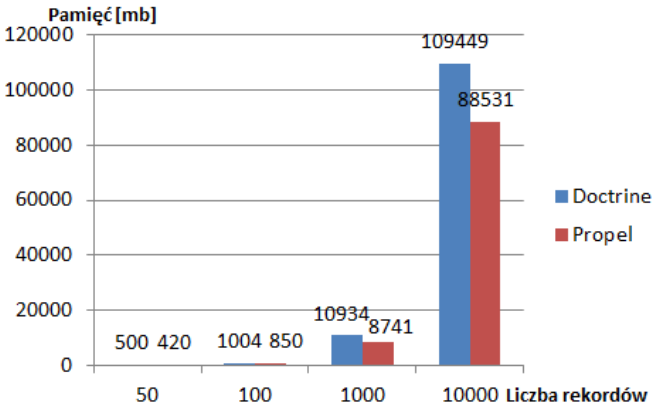
Trzeci scenariusz polegał na odczycie 50, 100 oraz 1000 rekordów niepowiązanych do relacyjnej bazy danych.



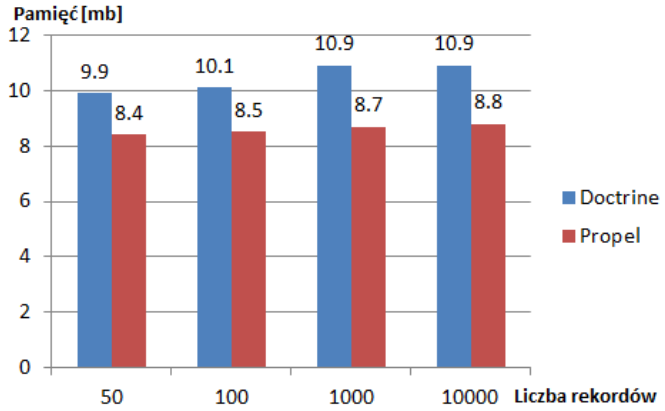
Rys. 10. Czas wykonywania skryptu w milisekundach



Rys. 11. Średni czas wykonywania skryptu w milisekundach



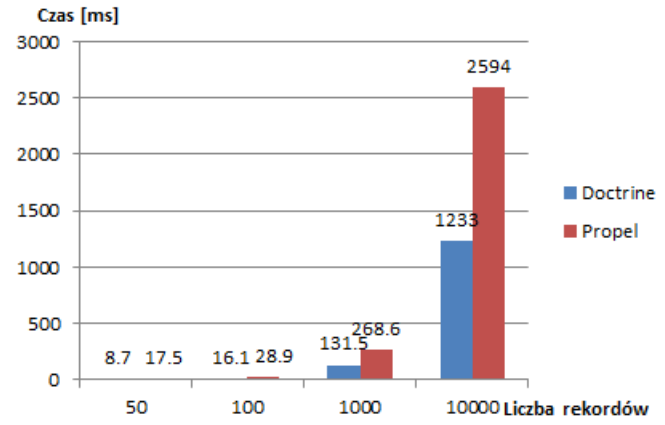
Rys. 12. Całkowite zużycie pamięci w megabajtach



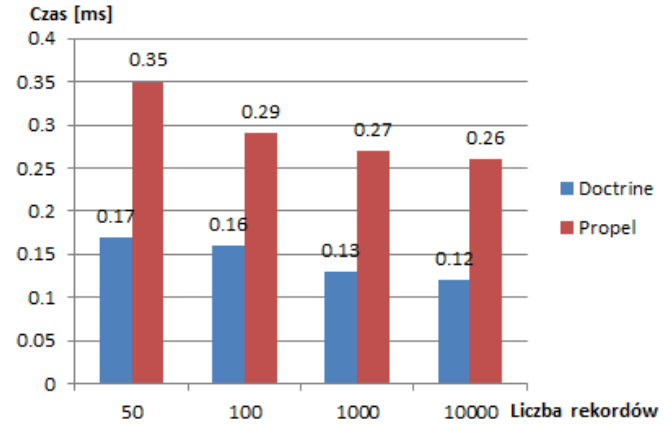
Rys. 13. Średnie zużycie pamięci w megabajtach

- **Scenariusz 4**

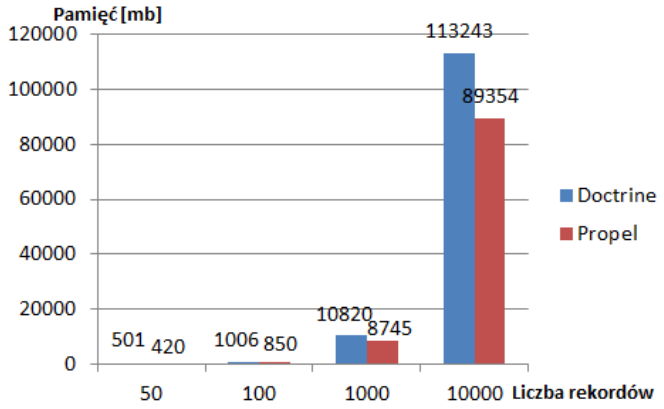
Czwarty scenariusz polegał na odczycie 50, 100 oraz 1000 rekordów powiązanych do relacyjnej bazy danych.



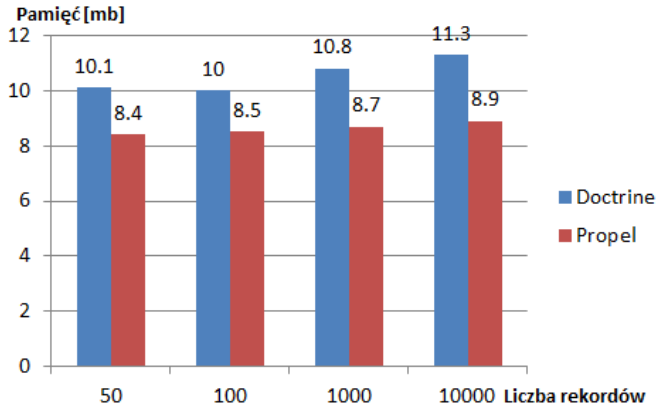
Rys. 14. Czas wykonywania skryptu w milisekundach



Rys. 15. Średni czas wykonywania skryptu w milisekundach



Rys. 16. Całkowite zużycie pamięci w megabajtach



Rys. 17.Średnie zużycie pamięci w megabajtach

5. Wnioski

Pierwszy scenariusz badał zapisanie pojedynczego rekordu bez asocjacji. W każdej sytuacji Doctrine przodował względem Propel w czasie wykonywania skryptu. W przypadku scenariusza pierwszego, Doctrine był szybszy o 17-28%, zależnie od ilości utrwalonych rekordów (Rysunek 2, Rysunek 3). Pomiar zużycia pamięci wykazał, że Propel zużywa od 10% do 17% mniej pamięci względem Doctrine (Rysunek 4, Rysunek 5).

Utrwalenie rekordów powiązanych wykazało, że Doctrine potrafi być trzy razy szybszy niż Propel. Wynik ten, można zaobserwować dla próby 50, 100 i 1000 rekordów (Rysunek 6, Rysunek 7). W przypadku zużycia pamięci, Propel potrafił zaoszczędzić do 6% pamięci (Rysunek 8, Rysunek 9).

Scenariusz trzeci badał odczyt rekordów bez relacji. Doctrine po raz kolejny był lepszy pod względem czasu wykonywania skryptu. Wspomniany zwycięzca był szybszy o 35-39%, zależnie od liczby utrwalonych rekordów (Rysunek 10, Rysunek 11). Propel tym razem także potrafił wykorzystać znaczne mniejszą ilość pamięci, bo aż do 25%.

W przypadku odczytu rekordów powiązanych, czas wykonywania skryptu potrafił być dwa razy większy w przypadku Propela w stosunku do Doctrine. Propel, jak i w poprzednich scenariuszach potrzebował zaalokować mniejszą ilość pamięci. Różnica w wykorzystania pamięci oscylowała w granicach 20-25%.

6. Podsumowanie

W ramach pracy przebadano i porównano obie technologie ORM pod względem wydajności, co pozwoliło sformułować wniosek, która technologia mapowania obiektowo-relacyjnego ma mniejszy czas wykonywania skryptu.

Z otrzymanych wyników jednoznacznie można stwierdzić, że Doctrine wykonuje operacje na bazie danych szybciej niż Propel, więc teza badawcza się potwierdziła. Badania wykazują, że obie technologie mają swoje mocne oraz słabe strony. W przypadku Doctrine, mocną stroną jest czas wykonywania skryptu, słabą zaś zużycie pamięci. Przy obserwacji wyników Propela, można zauważyć, że jest na odwrót – mniejsze zużycie pamięci i długi czas wykonywania skryptu.

Testy zapisu oraz odczytu rekordów dały jednoznaczne wyniki - Doctrine potrafi wykonać ten sam skrypt nawet trzy razy szybciej niż Propel. Zależnie od scenariusza, Doctrine był szybszy od 17% do nawet 300%. Badanie zużycia pamięci serwera wykazało, że Propel potrafi użyć mniej zasobów w przedziale 6%-25%.

W testach zostały przebadane dwa ważne czynniki, które jak się okazało potrafią być mocną oraz słabą stroną obu ORM. Szybszy czas wykonywania skryptu jest zawsze korzystniejszy dla użytkownika końcowego. Zużycie pamięci serwera jest kwestią podrzędną, gdyż pamięć RAM zawsze można zwiekszyć małym kosztem, a czas wykonywania skryptu nie da się żadnym prostym sposobem zmniejszyć, tym bardziej, kiedy różnica potrafi być trzy raz większa na korzyść Doctrine.

Podsumowując Doctrine jest wydajniejszą technologią mapowania obiektowo-relacyjnego w Symfony 3. W pracy zostały przeanalizowane jedynie dwa parametry. Ciekawym kierunkiem dalszych badań może być zbadanie innych parametrów takich jak zarządzanie transakcjami biznesowymi, konflikty zapytań, pojemność oraz konfiguracja [10][13].

Literatura

[1] Matt Zandstra: PHP Objects, Patterns and Practice, 5th Edition, 2016.

[2] Chris Pitt: Pro PHP MVC, 2012

[3] O'Reilly Media: Learning PHP Design Patterns, 2013

[4] Kevin Dunglas: Persistance in PHP with Doctrine ORM, 2013

[5] http://docs.doctrine-project.org/projects/doctrine-orm/en/latest/reference/association-mapping.html [18.08.2017]

[6] Jason E. Sweat: PHP Architect's Guide to PHP Design Patterns, 2005

[7] Martin Fowler: Patterns of Enterprise Application Architecture, 2002

[8] http://docs.doctrine-project.org/projects/doctrine-orm/en/latest/reference/working-with-objects.html [3.12.2017]

[9] http://www.vertabelo.com/blog/technical-articles/side-by-side-doctrine2-and-propel-2-comparison [20.09.2017]

[10] https://blog.appdynamics.com/engineering/top-6-database-performance-metrics-to-monitor-in-enterprise-applications/ [4.12.2017]

[11] http://docs.doctrine-project.org/projects/doctrine-orm/en/latest/reference/architecture.html [4.12.2017]

[12] https://en.wikipedia.org/wiki/Propel_(PHP) [4.12.2017]

[13] O'Reilly Media: High Performance MySQL, 3rd Edition, 2012

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 8 (2018) 241-246
Wysłane: 2018-03-15
Przyjęte: 2018-03-28

Analiza czasowa wydajności systemów Windows 10 oraz Windows 8.1 wykonana na podstawie aplikacji mobilnej

Jacek Chmiel*, Maria Skublewska-Paszkowska

Politechnika Lubelska, Instytut Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. W artykule przedstawiono analizę czasową wydajności systemów Windows 10 oraz Windows 8.1. Przeprowadzone badania skupiały się na porównaniu wydajności elementów systemu urządzenia mobilnego, które są ważne z punktu widzenia użytkownika a mianowicie na szybkości procesora, przepustowości RAM i pamięci masowej, szybkości generowania obrazu przez procesor graficzny oraz czasie dostępu do urządzeń. Na potrzeby analizy została stworzona aplikacja mobilna do pomiaru wyżej wymienionych elementów. Postawione w artykule hipotezy badawcze zostały zweryfikowane i częściowo udowodnione.

Słowa kluczowe: analiza czasowa; wydajność; systemy mobilne; Windows 10; Windows 8.1

*Autor do korespondencji.

E-mail: chmiel.jacek1990@gmail.com

Time analysis of the performance of Windows 10 and Windows 8.1 based on mobile application

Jacek Chmiel*, Maria Skublewska-Paszkowska

Institute of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract. This article presents time analysis of performance of Windows 10 and Windows 8.1. Presented research was mainly focused on performance comparison of mobile device system elements which are relevant from end user point of view like main processor, RAM and mass storage throughput, image generation speed by the graphics processor and access time to devices. In order to measure the performance of the above elements a mobile benchmark application was created. The hypotheses set in the article have been verified and partially confirmed.

Keywords: performance; time analysis; mobile systems; Windows 10; Windows 8.1

*Corresponding author.

E-mail address: chmiel.jacek1990@gmail.com

1. Wstęp

Urządzenia mobilne na przestrzeni ostatnich kilkunastu lat zyskały na popularności tak bardzo, że trudno sobie dzisiaj wyobrazić ich brak w codziennym życiu. Możliwości współczesnych urządzeń mobilnych są zbliżone do tych, jakie oferują komputery stacjonarne czy laptopy a przy tym ich możliwości przenoszenia są dużo większe. Wobec tej popularności i powszechnego występowania bardzo ważna jest wydajność całego systemu urządzenia mobilnego z punktu widzenia użytkownika w czasie codziennego pracy.

Celem pracy jest analiza czasowa wydajności systemów Windows 10 i Windows Phone 8.1. Aby możliwe było wykonanie analizy wydajności po tym kątem należało przeprowadzić eksperyment badawczy, który pozwoli zmierzyć czas wykonywania wybranych operacji za pomocą urządzenia mobilnego, a następnie porównać uzyskane wyniki pomiędzy systemami. Na potrzeby badań zostało stworzone autorskie, uniwersalne narzędzie badawcze, które za pomocą jednolitego zestawu testów jest w stanie przetestować obydwa systemy.

2. Badania literaturowe

Wydajność może być definiowana na wiele sposobów w zależności od kontekstu, w jakim chce się ją rozpatrywać [1]. Według Arnolda O. Allena wydajność jest miarą tego jak dobrze komputer wykonuje pracę, którą powinien wykonywać rozumianą jako wykonywanie poleceń [2]. W niniejszym artykule wydajność systemu operacyjnego jest rozumiana jako czas wykonywania operacji, które bezpośrednio wpływają na jego odbiór oraz komfort pracy z nim przez użytkownika końcowego. Istnieje szereg sposobów pomiaru wydajności systemów komputerowych jednak na początek należy proces analizy wydajności poprzedzić zebraniem wiedzy na temat badanego systemu jak i zdefiniować elementy, jakie należy poddać analizie oraz wyznaczyć kryteria ich oceny [3].

System operacyjny jest programem komputerowym, który pośredniczy między użytkownikiem komputera a jego podzespołami. Jego głównym celem jest stworzenie środowiska do uruchamiania programów, ich kontroli oraz dzielenia dostępnych zasobów. W tym celu system zajmuje się planowaniem i przydzielaniem czasu procesora, kontrolą

i przydziałem pamięci operacyjnej oraz obsługą sprzętu komputerowego [4].

Po przeprowadzeniu studiów literaturowych można zauważyć, że występuje niewiele prac w literaturze naukowej bezpośrednio porównujących jakiekolwiek dwa systemy operacyjne, czy to mobilne czy innego rodzaju. W przeważającej części są to analizy badające wydajność pojedynczych programów, algorytmów lub protokołów działających na urządzeniach mobilnych bądź możliwości jakie dają te urządzenia w postaci wsparcia w zbieraniu danych np.: do celów medycznych lub wspierania procesu nauki [5-8]. Prowadzone są także badania analizujące wydajność aplikacji mobilnych zbudowanych za pomocą rozwiązań międzyplatformowych [9].

Jednymi z nielicznych badań porównującymi mobilne systemy operacyjne pod kątem wydajności są badania przeprowadzone przez badaczy z uniwersytetu Maharashtra, opisujące porównanie mobilnego systemu operacyjnego Ubuntu Toch i Android [10] lub badania wydajności wbudowanych systemów mobilnych [11]. Nie zostały jak do tej pory przeprowadzone żadne badania wydajności systemów mobilnych firmy Microsoft. W związku z tym jest w pełni uzasadnione podjęcie tematu czasowej analizy wydajności systemu Windows 10 na urządzeniach mobilnych.

Obszarem objętym badaniem będzie wydajność systemów Windows 10 i Windows Phone 8.1 na podstawie aplikacji mobilnej. Do eksperymentu badawczego zostały wybrane dwa modele urządzeń mobilnych Nokia Lumia 830 i Nokia Lumia 930. Na obydwu urządzeniach były zainstalowane systemy Windows 10 i Windows Phone 8.1, tak więc do testów posłużyły 4 różne fizyczne urządzenia.

Zostały postawione następujące hipotezy badawcze. Wydajność systemu operacyjnego Windows Phone 8.1 będzie większa od wydajności systemu operacyjnego Windows 10 Mobile na urządzeniu mobilnym Nokia Lumia 830. Wydajność systemu operacyjnego Windows Phone 8.1 będzie większa od wydajności systemu operacyjnego Windows 10 Mobile na urządzeniu mobilnym Nokia Lumia 930.

3. Metoda badań

3.1. Obiekt badań

Obiektem badań są mobilne systemy operacyjne Windows 10 i Windows Phone 8.1. Na potrzeby analizy ich wydajności zostały wybrane elementy systemu urządzenia mobilnego, których szybkość i efektywność pracy ma wpływ na jego użytkowanie. Są to procesor, pamięć RAM, pamięć masowa, procesor graficzny oraz czas dostępu do interfejsu API, służący do obsługi funkcjonalności typowych dla urządzeń mobilnych.

Systemy Windows 10 i Windows Phone 8.1 zostały przetestowane na urządzeniach mobilnych o następującej specyfikacji:

- Nokia Lumia 830 posiada ekran o przekątnej 5 cali, pamięć operacyjną o rozmiarze 1 gigabajta i czterordzeniowy procesor Qualcomm Snapdragon 400 o taktowaniu pojedynczego rdzenia 1.2 gigaherca.
- Nokia Lumia 930 posiada 5 calowy ekran jednak ma 2 gigabajty pamięci operacyjnej i czterordzeniowy procesor Qualcomm Snapdragon 800 o taktowaniu 2.2 gigaherca.

3.2. Opis narzędzia badawczego

Na potrzeby przeprowadzenia badań zostało stworzone uniwersalne narzędzie badawcze, które jednym zestawem testów symulujących obciążenie systemu jest w stanie przetestować wydajność systemów Windows 10 oraz Windows Phone 8.1. Bada ono w następujący sposób elementy wybrane systemu urządzenia mobilnego.

W celu przetestowania procesora zostały utworzone algorytmy testujące instrukcje arytmetyczne operujące na liczbach stałoprzecinkowych i zmiennoprzecinkowych. Najprostszym sposobem na obciążenie procesora w stu procentach jest uruchomienie kolejnych operacji w pętli a następnie pomiar czasu ich wykonania. Na potrzeby tego badania została określona liczba obrotów pętli na sto milionów a w każdym z nich są wykonywane operacje dodawania, odejmowania, mnożenia i dzielenia.

W celu przetestowania przepustowości zapisu i odczytu pamięci RAM został stworzony algorytm, który zapisuje do pamięci i z niej wczytuje wcześniej przygotowany plik z danymi o rozmiarze 16 MB, który powstaje w czasie instalacji aplikacji testującej. Plik ten jest w teście zapisu iteracyjnie zapisywany do pamięci, a w teście odczytu zapisywany jeden raz, a następnie określoną ilość razy wczytywany z pamięci. Znając ilość iteracji oraz rozmiar pliku obliczana jest przepustowość pamięci podana w MB/s.

Do testów przepustowości pamięci masowej również jest wykorzystywany wcześniej przygotowany plik o rozmiarze 16 MB. W przypadku testu odczytu z pamięci masowej jest on odczytywany określoną ilość razy z pamięci masowej telefonu. Znając ilość iteracji obliczana jest przepustowość odczytu z pamięci masowej w MB/s. W przypadku testu zapisu plik ten jest wczytywany do pamięci RAM określoną liczbę razy, dynamicznie ustalaną w czasie wykonywania programu, a następnie tak przygotowana paczka danych jest zapisywana do pamięci masowej. Tutaj również przepustowość jest podawana w MB/s.

W celu przetestowania procesora graficznego została użyta biblioteka Win2D wspierająca rysowanie grafiki [12]. Wydajność jest tutaj badana jako średni czas generowania obrazu wyświetlanego na ekranie urządzenia mobilnego. Test trwa 30 sekund i co jedną sekundę jest dodawany do rysowanej sceny jedna losowa elipsa, co zwiększa obciążenie

procesora graficznego. Wynikiem testu jest średnia liczba klatek na sekundę.

Funkcjonalności specyficzne dla urządzeń mobilnych są testowane algorytmem mierzącym czas dostępu do metod API, służących do obsługi akcelerometru, notyfikacji użytkownika i wibracji. Test ten wskazuje na szybkość obsługi urządzeń na danym systemie.

Sposób wykonania narzędzia badawczego pozwala na uruchomienie wszystkich wyżej wymienione testów w kolejności sekwencyjnej za pomocą naciśnięcia jednego przycisku na głównym ekranie aplikacji. Po przeprowadzeniu wszystkich testów poszczególne wyniki osiągnięte w każdym z nich są prezentowane na interfejsie użytkownika.

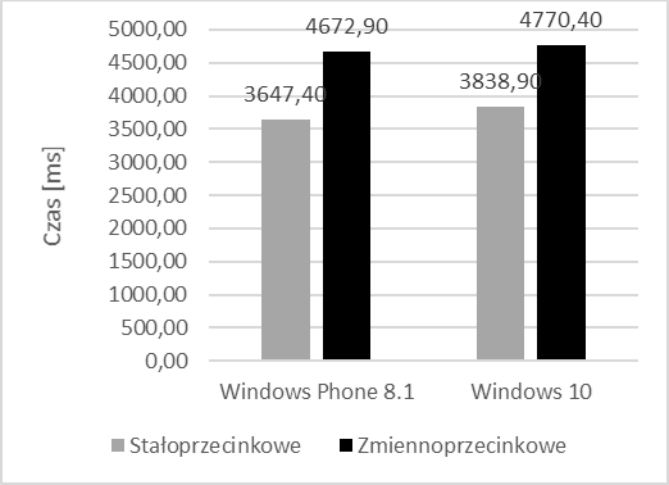
3.3. Procedura eksperymentów

Przeprowadzono po 10 testów wydajności na każdym urządzeniu co dało łącznie 40 testów. Sposób wykonania aplikacji powoduje, że aplikacja po naciśnięciu przycisku start na interfejsie użytkownika wykonuje wszystkie zaprojektowane testy sekwencyjnie.

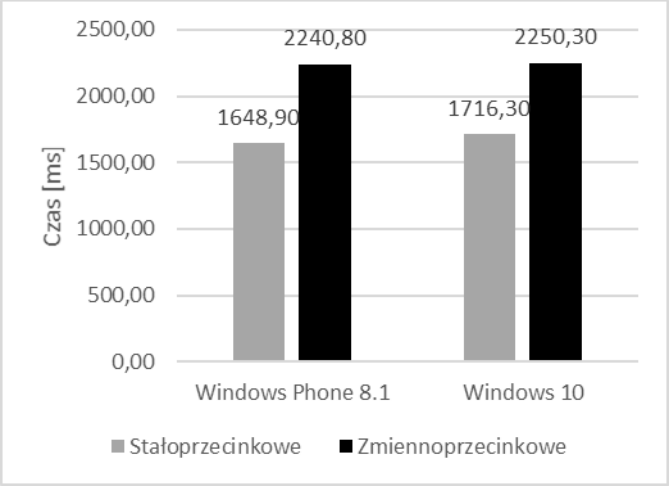
Każdy test był wykonywany po świeżym uruchomieniu aplikacji testującej, w celu wyeliminowania optymalizacji, jakie środowisko uruchomieniowe stosuje dla aplikacji na platformie .NET. Miało to na celu w jak największym stopniu przetestować różnicę pomiędzy dwoma systemami. Po każdym wykonanym teście spisywane były rezultaty uzyskane za jego pomocą.

4. Wyniki badań

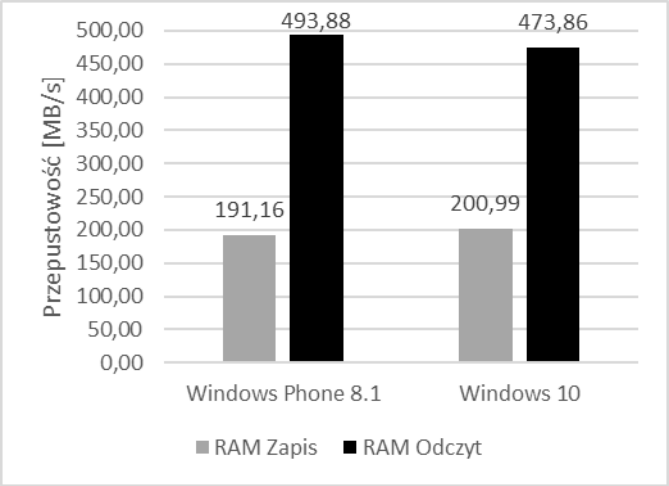
Na rysunkach 1-11 zostały przedstawione uśrednione wyniki ze wszystkich przeprowadzonych testów wydajności.



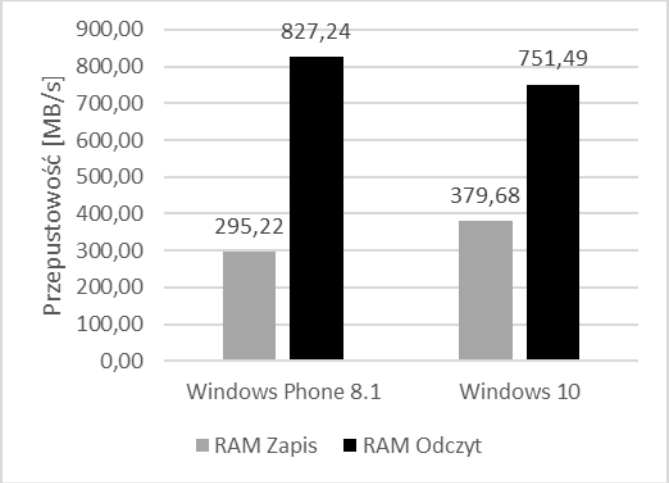
Rys. 1. Średnie czsy testów procesora na urządzeniu Nokia Lumia 830



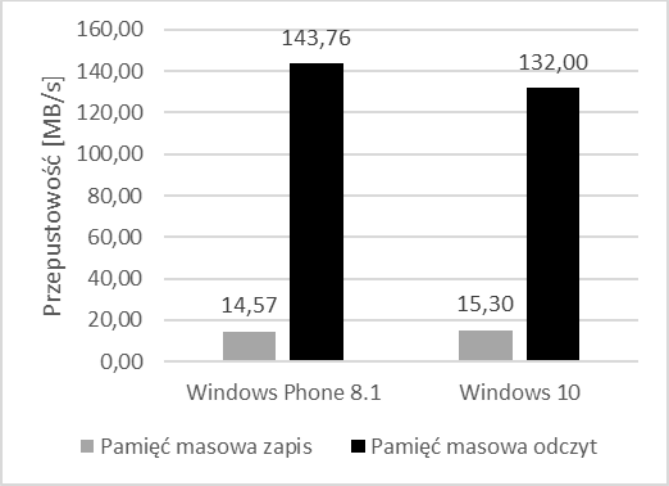
Rys. 2. Średnie czsy testów procesora na urządzeniu Nokia Lumia 930



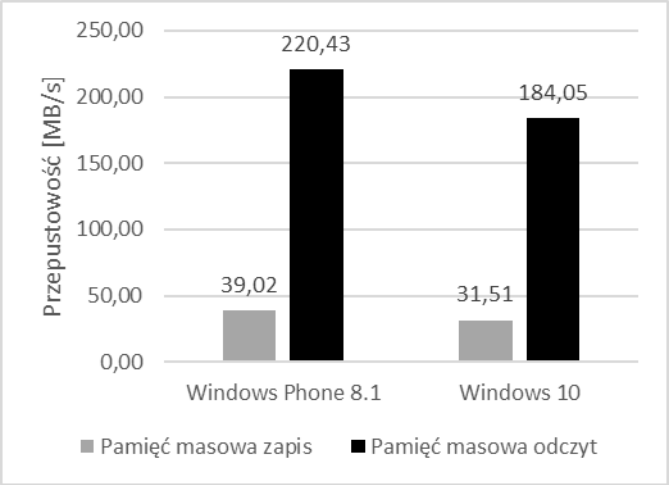
Rys. 3. Średnia przepustowość pamięci RAM na urządzeniu Nokia Lumia 830



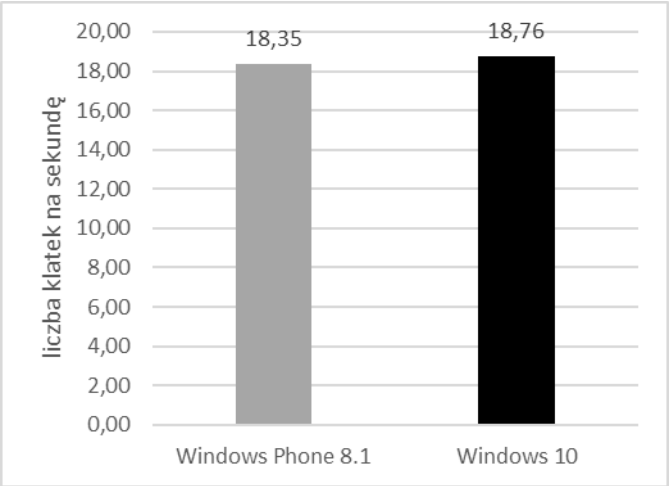
Rys. 4. Średnia przepustowość pamięci RAM na urządzeniu Nokia Lumia 930



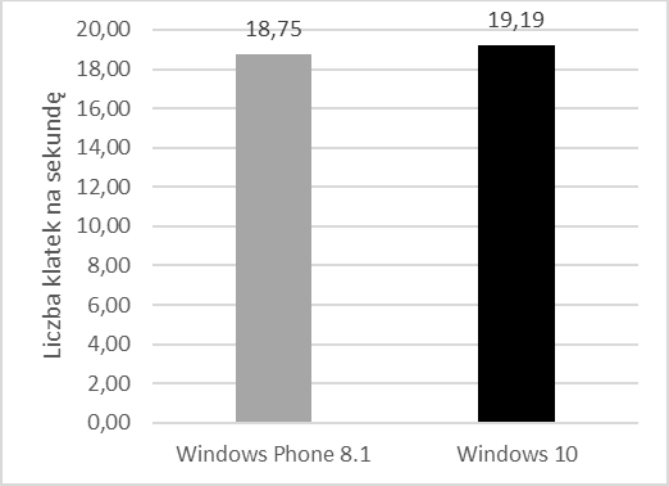
Rys. 5. Średnia przepustowość pamięci masowej na urządzeniu Nokia Lumia 830



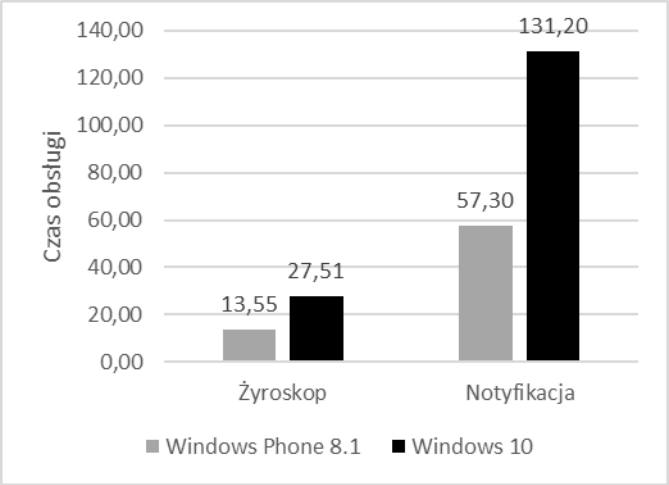
Rys. 6. Średnia przepustowość pamięci masoewj na urządzeniu Nokia Lumia 930



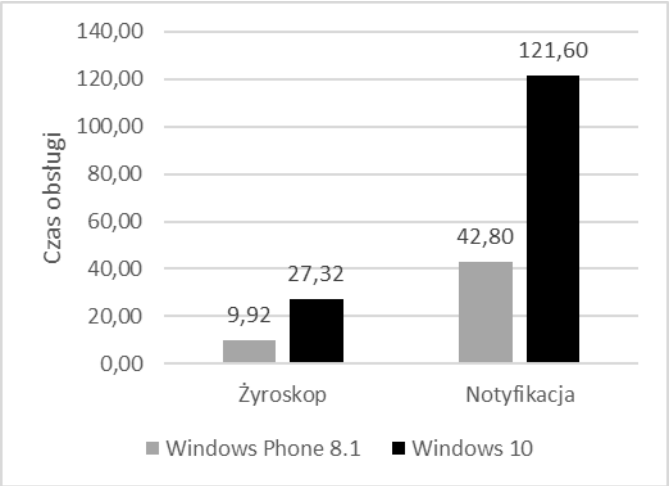
Rys. 7. Średnia liczba klatek na sekudę osiągnięta w teście procesora graficznego na urządzeniu Nokia Lumia 830



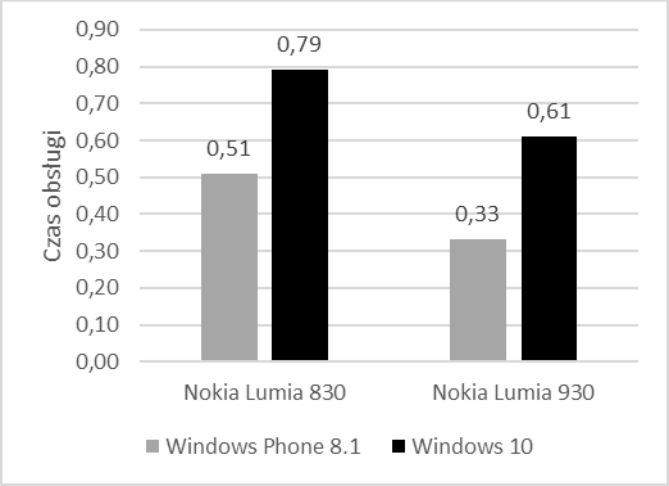
Rys. 8. Średnia liczba klatek na sekudę osiągnięta w teście procesora graficznego na urządzeniu Nokia Lumia 930



Rys. 9. Średnie czasy obsługi API żyroskopu i notyfikacji na urządzeniu Nokia Lumia 830



Rys. 10. Średnie czasy obsługi API żyroskopu i notyfikacji na urządzeniu Nokia Lumia 930



Rys. 11. Żestawione rednie czasy obsługi API wibracji obydwu systemach i urządzeniach

Na rysunkach 1 i 2 zostały uzyskane wyniki badania procesora jako czas wykonywania operacji arytmetycznych z użyciem liczb stałoprzecinkowych i zmiennoprzecinkowych. W teście tym na obydwu urządzeniach lepszy rezulaty od Windows 10 Mobile uzyskał system Windows Phone 8.1. Na urządzeniu Nokia Lumia 830 był o 4,99% szybszy w teście na liczbach stałoprzecinkowych i o 2,04% szybszy w teście na liczbach zmiennoprzecinkowych. Na urządzeniu Nokia Lumia 930 było szybszy o odpowiednio 3,93% oraz 0,42%.

Na rysunkach 3 i 4 pokazane są wyniki przepustowości pamięci RAM zarówno odczytu z pamięci jaki i zapisu do niej. Windows Phone 8.1 osiągnął lepszą przepustowość odczytu na obydwu urządzeniach o 4,49% w przypadku urządzenia Nokia Lumia 830 i o 9,26% w przypadku Nokia Lumia 930. Jednak Windows 10 osiągnął lepsze rezultaty w teście zapisu do pamięci o 5,14% na urządzeniu Nokia Lumia 830 i o 28,61% na urządzeniu Nokia Lumia 930.

Rysunki 5 i 6 pokazują wyniki testów przepustowości pamięci masowej. Na urządzeniu Nokia Lumia 830 Windows 10 w teście zapisu do pamięci osiągnął o 5,14% lepszy wynik od Windows Phone 8.1. W pozostałych przypadkach ten drugi osiągnął lepsze rezultaty. Na urzadzeniu Nokia Lumia 930 zapis do pamięci masowej było o 19,26% szybszy, natomiast odczyt o 9,16% szybszy. Na urządzeniu Nokia Lumia 830 osiągnął odczyt o 4,05% szybszy.

Na rysunkach 7 i 8 pokazane są wyniki testu procesora graficznego przedstawione jako średnia ilość klatek na sekundę. Na obydwu urządzeniach lepsze rezultaty osiągnął system Windows 10 Mobile. Dla Nokia Lumia 830 rezultat był o 2,24% a dla Nokia Lumia 930 2,36%.

Rysunki 9-11 pokazują wyniki testów czasów dostępu do API systemów urządzenia mobilnego żyroskopu, wibracji i notyfikacji. We wszystkich tych testach Windows 10 osiągnął dużo słabsze wyniki. Na urządzeniu Nokia Lumia 830 były one słabsze odpowiednio o 51,33%, 35,44% i 66,33%. Na urządzeniu Nokia Lumia 930 były one słabsze odpowiednio o 63,69%, 45,90% oraz o 64,8%.

5. Dyskusja wyników

Po przeprowadzonej analizie uzyskanych wyników postawione hipotezy zostały częściowo potwierdzone. System Windows 10 okazał się niemalże we wszystkich testach mniej wydajny od systemu Windows Phone 8.1. Największe różnice pomiędzy obydwoma systemami zostały ukazane w testach dostępu do urządzeń (Rys. 9-11). W tych testach Windows Phone 8.1 osiąga czasy około dwa niższe na urządzeniu Nokia Lumia 830 oraz około trzy razy niższe na urządzeniu Nokia Lumia 930. Windows 10 osiągnął lepsze rezultaty na obydwu urządzeniach jedynie w teście wydajności procesora graficznego (Rys. 7-8) oraz w teście zapisu do pamięci RAM (Rys. 3-4). Na urządzeniu Nokia Lumia 830 osiągnął dodatkowo lepszy wynik w teście zapisu do pamięci masowej (Rys. 5). Wyniki uzyskane podczas przeprowadzania analizy porównawczej potwierdzają przypuszczenia autora o niewielkiej wariancji pojedynczych rezultatów, która w przypadku niektórych testów wynosi około 5% średnich wartości uzyskanych w tych testach. Jest to podyktowane izolacją procesu, który wykonuje kod aplikacji na systemach mobilnych oraz przydzielania mu całych dostępnych zasobów sprzętowych urządzenia mobilnego przez system. Wyjątkiem tutaj jest limit przydzielanej pamięci operacyjnej dla pojedynczej aplikacji, różny dla każdego systemu oraz dostępnej pamięci na urządzeniu.

Otrzymana różnica w wydajności wynikać może z innego zestawu sterowników urządzeń, implementacji jądra obydwu systemów jak i samej praformy .NET oraz jej komponentów, takich jak zbieracz śmieci lub kompilator JIT (ang. Just in Time). Wszystkie te czynniki mogą powodować większy narzut podczas obsługi poszczególnych elementów systemu urządzenia mobilnego. Jednoznaczne stwierdzenie, który z powyższych elementów miał największy wpływ na ukazaną różnicę wydajności jest niemożliwe ze względu na zamknięcie ich źródeł przez firmę Microsoft.

6. Wnioski

Rezultaty przeprowadzonych badań dają obraz różnic pomiędzy obydwoma systemami i mogą posłużyć jako wskazówka dla osób chcących nabyć urządzenie mobilne z jednym z systemów mobilnych firmy Microsoft lub mogą zostać wykorzystane przez samych twórców systemu, używając przygotowane narzędzie do porównania wydajności. Do przedstawionej analizy porównawczej zostały wykorzystane dwa identyczne urządzenia mobilne, w związku z tym, ukazane różnice w wydajności mogą wynikać z odmiennej wewnętrznej budowy jądra systemów oraz platformy .NET jak również z innego zestawu sterowników.

W przyszłości pracując nad następną analizą można by było dokonać pomiaru całkowitego czasu wykonania wszystkich testów jako kolejny element badań. Kolejną sugestią jest również głębsze poznanie API

dostępnego do obsługi grafiki 2D. Aktualnie dostępny interfejs programisty nie daje możliwości próbkowania szybkości wyświetlania grafiki na ekranie urządzenia mobilnego co wymusza pomiar czasu wykonywania kodu z poziomu aplikacji, zamiast pobierania czasu generowania grafiki z poziomu API.

Literatura

[1] Lilja D.J.: Measurning Computer Performance: A Practitioner's guide, Cambridge University Press, 2005.

[2] Allen A.O.: Computer Performance Analysis with Mathematica. Academic Press, 1994.

[3] Jain R.K.: Art of Computer Systems Performance Analysis Techniques For Experimental Design Measurements Simulation And Modeling. Wiley, 1991.

[4] Hoare C.A.R.: Monitors: An Operating System Structuring Concept. Communication of the ACM, 10 (1974), 549-557.

[5] Sung Y., Chang K., Liu T.: The effects of integrating mobile device with teaching and learning on students' learning performance: a meta-analysis and research synthesis, Computer & Education, 3 (2016), 252–275.

[6] Shahabi S., Ghazvini M., Bakhtiarian M.: A modified algorithm to improve security and performance of AODV protocol against black hole attack. Springer Wireless Network, 5 (2015), 1505-1511.

[7] V. D. Blondel, A. Decuyper, G. Krings.: A survey of results on mobile phone datasets analysis, EPJ Data Science, 1 (2015), 1–55.

[8] Mazomenos E.B., Biswas D., Acharyya A., Chen T., Maharatna K., Rosengarten J., Morgan J., Curzen N.: A Low-Complexity ECG Feature Extraction Algorithm for Mobile Healthcare Applications, IEEE Journal of Biomedical and Health Informatics, 2 (2013), 459-469.

[9] Latif M., Lakhrissi Y., Nfaoui E.H., Es-Sbai, N.: Cross platform approach for mobile application development: A survey. IT4OD (2016), http://dx.doi.org/10.1109/IT4OD.2016.7479278.

[10] Shaikh A.S, Inamdar M.U.: Performance Analysis: Ubuntu Touch v/s Android OS. International Journal of Science, Engineering and Technology Research, 3 (2014), 1795-1800.

[11] Park S., Park Y., Choi J.: Performance Analysis of Embedded OS in ARM Platform, ICTC 2012

[12] https://blogs.windows.com/buildingapps/2014/09/05/introducing-win2d-gpu-accelerated-2d-graphics-programming-in-the-windows-runtime/#721RzUqREPKGhSRb.97, [27.02.2018]

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 8 (2018) 247-251
Wysłane: 2018-03-17
Przyjęte: 2018-03-28

Porównanie wybranych metod komunikacji sieciowych na platformie Android

Przemysław Żydek*, Jakub Smołka
Politechnika Lubelska, Instytut Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. Artykuł poświęcony jest porównaniu trzech metod komunikacji aplikacji mobilnej z serwerem. Analizie poddano wyniki sześciu badań przeprowadzonych przy użyciu protokołów HTTP, HTTPS oraz technologii gniazd serwera. Zbadano czas przesłania danych o różnych rozmiarach pomiędzy aplikacją klienta a serwerem oraz wpływ operacji na użycie procesora i zużycie baterii. Doświadczenia polegały na wysłaniu i odebraniu formularza, wysłaniu i odebraniu dużego zdjęcia oraz zapewnieniu komunikacji ciągłej w celu zbadania użycia urządzenia. Do przeprowadzenia analizy stworzono aplikację klienta na platformę Android obsługującą badane technologie. Stronę serwera stanowiły programy napisane w technologii Java uruchomione na serwerze Tomcat. Przeprowadzone badania umożliwiły wskazanie faworyta i jest nim technologia gniazd serwera.

Słowa kluczowe: http; HTTPS; gniazdo serwera; Android; komunikacja sieciowa

*Autor do korespondencji.

Adres e-mail: przemekkzydek@wp.pl

Comparison of selected network communication methods on the Android platform

Przemysław Żydek*, Jakub Smołka
Institute of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract. This paper is devoted to comparing three communication methods between mobile applications and servers. The analysis encompassed the results of six tests conducted using HTTP and HTTPS protocols, and server-socket technology. All sending times of data with various sizes between the client application and the server, and the impact of this operation on the processor load and battery use, were evaluated. The experiments consisted of sending and receiving a form, sending and receiving a large photo, and ensuring continuous communication to assess device use. To perform the analyses, an Android application was created to support the researched technology, while the server side was composed of programs written in Java, running on a Tomcat server. The conducted research made it possible to establish the best solution, which is server-socket technology.

Keywords: HTTP; HTTPS; socket; Andoid; network communication

*Corresponding author.

E-mail address: przemekkzydek@wp.pl

1. Wstęp

Duża liczba możliwości wykorzystania urządzeń mobilnych sprawiła, że znajdują one zastosowanie w prawie każdej dziedzinie. Dostarczane funkcjonalności realizowane są za pomocą aplikacji zainstalowanych na urządzeniu. Bardzo ważnym aspektem w działaniu tych rozwiązań jest rodzaj komunikacji sieciowej stosowany do wymiany danych z serwerem. Wykorzystywany w tych rozwiązaniach model klient-serwer zazwyczaj składa się z jednego centralnego serwera i wielu punktów klienckich [1]. Przed wyborem odpowiedniej technologii należy odpowiedzieć na pytanie, czy operacja przesłania danych ma być jak najbardziej wydajna, a może informacje na których operuje mają być bezpieczne i odporne na ataki.

Komunikacja sieciowa jest tematem wielu prac. w [2] Autorzy artykułu opisują tworzenie aplikacji sieciowych w środowisku Java przy pomocy gniazd serwera. Program uruchomiony po stronie klienta inicjuje połączenie poprzez wysłanie żądania na określony port, serwer interpretuje otrzymane dane i jeśli jest to możliwe zwraca zasób. Serwer centralny może działać w dwóch modelach TCP/IP. Pierwszy z nich to model iteracyjny, jest dość mocno ograniczony ponieważ jego architektura obsługuje tylko jedno żądanie, kolejne nadchodzące żądania są kolejkowane, co oznacza opóźnienia w obsłudze klientów. Drugi model współbieżny umożliwia obsługę wielu żądań jednocześnie. Gdy pojawia się połączenie z drugim klientem następuje rozwidlenie procesów. Zwiększa to znacznie wydajność obsługi większej ilości klientów w tym samym czasie. Kolejne opisane aspekty towarzyszące szyfrowaniu to konieczność nabycia certyfikatu, obciążenie sprzętu czy nakład pracy sieci.

W artykule [3] Autorzy poruszyli aspekt wydajności protokołu HTTP oraz jego szyfrowanej wersji HTTPS. Zwrócono uwagę na to, że każdej technologii zabezpieczającej dane towarzyszy koszt wykonania dodatkowych operacji. Wszystkie koszty są do zaakceptowania jeśli dane którymi operuje aplikacja są poufne. w artykule zaprezentowano ewolucję udziału przepływu komunikacji za pośrednictwem protokołu HTTPS, od kwietnia 2012 r. do września 2014 r. na przykładzie portali Facebook i YouTube zanotowano

dwukrotny wzrost popularności tego rozwiązania. Zjawisko to jest podyktowane ochroną prywatności użytkowników. Popularność technologii ma też wpływ na zmniejszenie kosztów infrastruktury. Do porównania technologii przeprowadzono badania, w których zmierzono czasy ładowania stron internetowych. Zbadano komunikację z wykorzystaniem protokołów HTTP oraz HTTPS. Otrzymane wyniki wykazują na to, że użycie HTTPS znacznie wydłuża ładowanie strony. Wydłużony czas jest spowodowany głównie nawiązywaniem połączenia szyfrowanego za pomocą protokołu TLS. Bardzo ważnym aspektem takiej komunikacji w urządzeniach mobilnych jest wpływ na żywotność baterii. Przeprowadzone badania nie wykazały znacznego wzrostu kosztów energii przy testowaniu protokołu HTTPS.

Artykuł [4] zawiera informacje o procesie badania komunikacji pomiędzy urządzeniem mobilnym, opartym na architekturze ARM a serwerem. Autor prezentuje jakich narzędzi należy użyć do przechwycenia ruchu sieciowego. Programy Tcpdump oraz Burp umożliwiają wygenerowanie pliku z całym ruchem sieciowym, który w dalszej części badań może posłużyć do analizy komunikacji.

Badanie w artykule [5] miało na celu zbadanie aplikacji mobilnych pod kątem zaimplementowanych rozwiązań sieciowych. Badana próbka to 639 283 aplikacji, z czego w 573 258 wykryto połączenia sieciowe. Analiza wykazała, że znaczna większość połączeń realizowana jest za pomocą protokołu HTTP, 414 194 aplikacji korzysta z tej technologii. Szyfrowana odmiana tego protokołu została znaleziona w 229 317 aplikacjach.

2. Przedmiot badań

Przedmiotem badań były trzy popularne metody komunikacji sieciowej. Zbadano protokoły HTTP, HTTPS oraz gniazdo serwera. Pierwszy z nich to natywny protokół znajdujący się w warstwie aplikacji modelu OSI, głównie wykorzystywany przez serwery WWW oraz przeglądarki internetowe. Dostęp do zasobów realizowany jest za pomocą żądań w postaci tekstu, wysyłanych do serwera [6, 7]. Drugi jest odmianą pierwszego z tą różnicą, że przesyłane dane są szyfrowane przez protokół TLS. HTTPS zapewnia w pełni poufność przesyłanych danych [8, 9]. Technologia gniazd opiera się na protokole TCP lub UDP. Transport danych odbywa się za pomocą strumienia, daje to możliwość odczytu i zapisu bajtów danych do wskazanego adresu IP i portu [10,11].

3. Przeprowadzone badania

Do przeprowadzenia analizy metod komunikacji zaimplementowano (1) aplikację mobilną na platformę Android pełniącą rolę klienta oraz (2) stronę serwera obsługującego żądania i zapisującego czasy doświadczeń w bazie danych. Programy obsługują wszystkie trzy opisywane sposoby komunikacji. Badania zostały przeprowadzone w środowisku lokalnym, w którego skład wchodzi komputer pełniący rolę serwera, ruter Wifi oraz urządzenie klienckie typu smartfon. Przyjęty format w jakim przesyłane są dane to JSON, składający się z trzech elementów. Pierwszy z nich zawiera informacje o przesyłanych danych, drugi właściwe dane, trzeci czas operacji wyrażony w ms. Wszystkie operacje poddane analizie wymieniono w kolejnym punkcie.

3.1. Badane operacje

Do analizy metod komunikacji klienta z serwerem przeprowadzono następujące operacje:

1) Wysłanie formularza składającego się z 20 pól opisowych, dane mogą odzwierciedlać zamówienie towaru lub przeprowadzenie badania poza biurem (jest to około 15 tys. znaków). Próbę wykonano 100 razy.
2) Pobranie formularza, aplikacja klienta wysyła żądanie o udostępnienie zasobu. Serwer w odpowiedzi zwraca formularz identyczny jak w poprzednim punkcie. Próbę wykonano 100 razy.
3) Wysłanie z pamięci lokalnej urządzenia zdjęcia o rozdzielczości 6000 x 4000 pikseli. Rozmiar pliku wynosi ok 7 MB o rozszerzeniu JPG. Próbę wykonano 100 razy.
4) Pobranie zdjęcia z serwera. Dane wykorzystane z poprzedniego punktu. Próbę wykonano 100 razy.
5) Pomiar zużycia baterii oraz użycia procesora podczas ciągłej komunikacji w interwale z pół sekundową przerwą. Użyto aplikacji *Monitor Systemu* oraz *CoolTool.*

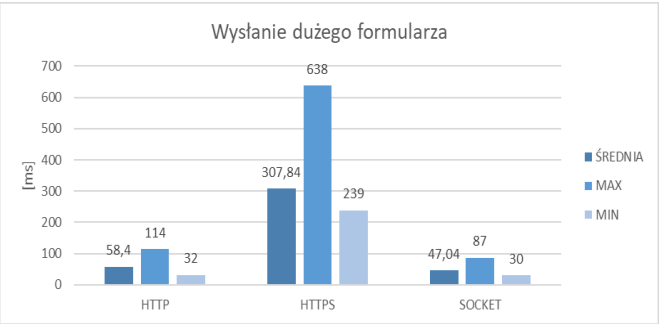
3.2. Platforma testowa

Wszystkie badania przeprowadzono na tych samych urządzeniach. Aplikacja kliencka została stworzona w środowisku programistycznym Android Studio w wersji 2.3.3, z wykorzystaniem narzędzi programistycznych SDK w wersji 26. Urządzenie klienta stanowi smartfon Samsung Galaxy J3 2016 z systemem Android w wersji 5.1.1 Lollipop. Stronę serwera zapewnia komputer typu laptop Acer Swift 3 z systemem operacyjnym Windows 10 Professional.

4. Wyniki przeprowadzonych badań

4.1. Wysłanie formularza

Rysunek 1 przedstawia wykres z czasem średnim, maksymalnym oraz minimalnym dla każdej technologii.

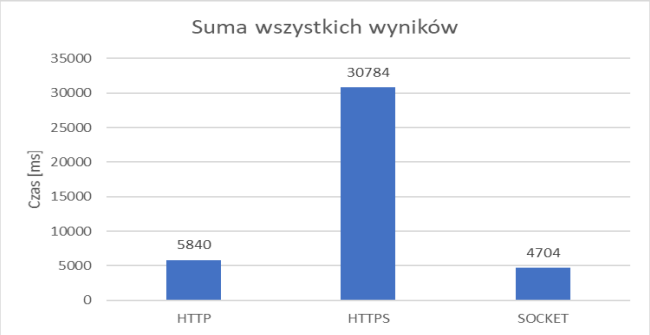


Rys. 1. Wykres z czasami wysłania formularza

Średni czas operacji wysłania formularza do serwera oraz odebrania odpowiedzi dla protokołu HTTP wyniósł 58 ms,

szyfrowane połączenie na wykonanie tej operacji potrzebowało 308 ms, a technologia gniazda osiągnęła wynik 47 ms.

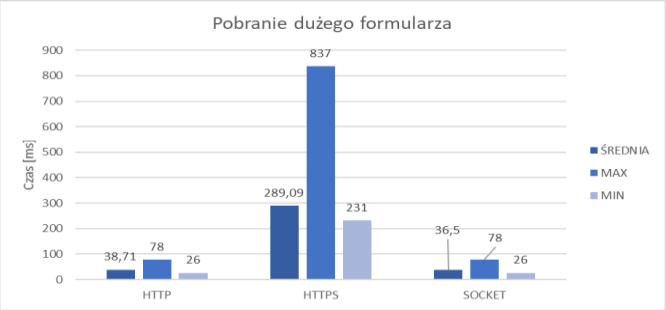
Analiza sumy wszystkich czasów osiągniętych przy 100 wykonanych operacjach, wskazuje bezkonkurencyjną technologię gniazd serwera z wynikiem 4707 ms. Najdłużej tą operację wykonuje protokół HTTPS z sumarycznym czasem 30 784 ms (rys 2).



Rys. 2. Suma wszystkich zgromadzonych wyników

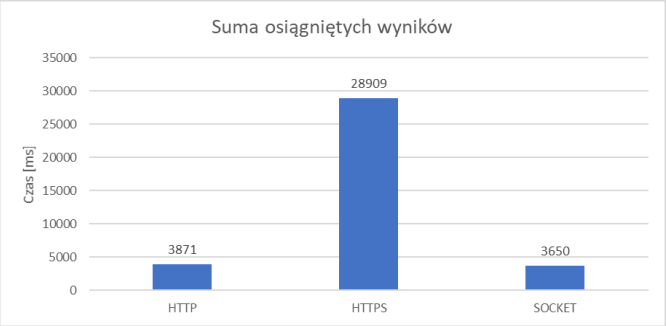
4.2. Pobranie formularza

Średni czas wykonania operacji dla badanych technologii HTTP, HTTPS i gniazda serwera wynoszą odpowiednio 39 ms, 289 ms i 37 ms. Zebrane dane prezentuje rysunek 3.



Rys. 3. Wykres z czasami pobrania formularza

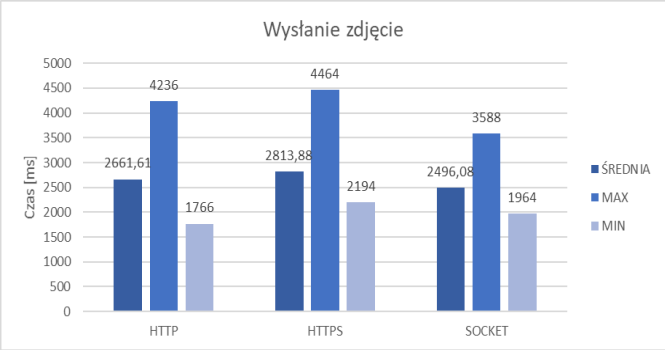
Łączny czas badanych operacji jest krótszy niż w poprzednim punkcie. Technologia gniazd bardzo zbliżyła się wydajnością do protokołu HTTP. Uśredniony czas obydwu technologii wynosi 3760 ms. Najdłuższy łączny czas o wartości 28909 ms zanotowano podczas badania protokołu HTTPS (rys. 4).



Rys. 4. Suma wszystkich zgromadzonych wyników

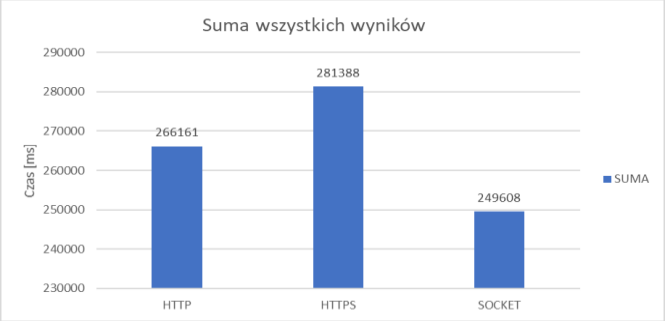
4.3. Wysłanie zdjęcia

Kolejnym badanym aspektem jest wysłanie zdjęcia o rozmiarze 7 MB do serwera. Najgorszy średni wynik uzyskał protokół HTTPS z czasem 4464 ms. Drugi w zestawieniu uplasował się protokół HTTP z rezultatem 2662 ms. Najkrótszy czas uzyskała technologia gniazd, osiągnęła wynik 2496 ms. Analizowane czasy umieszczono na rysunku 5.



Rys. 5. Wykres z czasami wysłania zdjęcia

Najlepszy łączny czas w analizowanym badaniu osiągnęła technologia gniazd z wynikiem 249 s. Zaraz za nią znajduję się rozwiązanie wykorzystujące protokół HTTP, 266 s. Ostatnie miejsce w zestawieniu z czasem 281 s zajęła metoda HTTPS. Wyniki przedstawia rysunek 6.

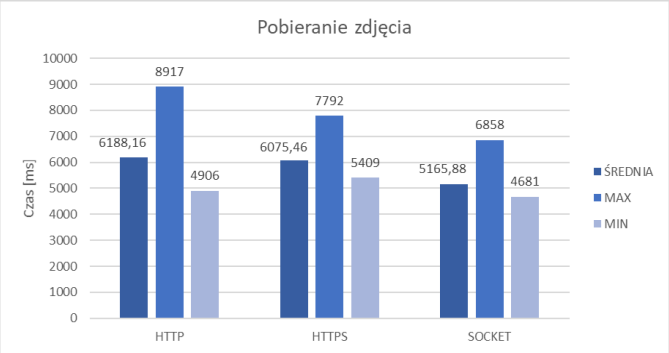


Rys. 6. Suma wszystkich zgromadzonych wyników

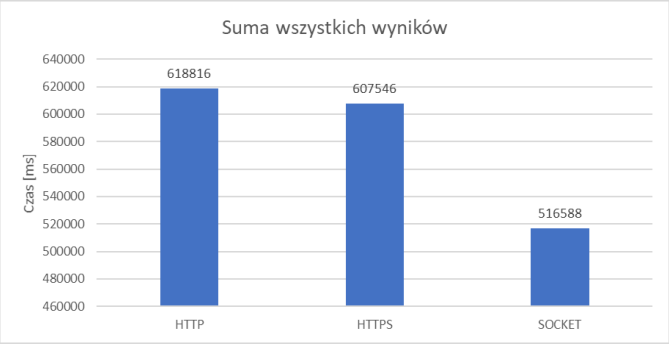
4.4. Pobranie zdjęcia

Analizowany eksperyment opiera się na operacji odebrania zasobu (zdjęcia) od serwera. Faworytem w tym zestawieniu okazała się technologia gniazd, osiągając najkrótszy czas 5165 ms. Drugim czasem wynoszącym 6075 ms może pochwalić się protokół HTTPS. Najgorzej w zestawieniu wypadło rozwiązanie HTTP z czasem 6188 ms. Omawiane wyniki zaprezentowano na rysunku 7.

Najkrótszy czas potrzebny do wysłania 100 dużych zdjęć do serwera to 517 s, osiągnęła go technologia gniazd. Drugi odnotowany czas to 608 s, przy analizie protokołu HTTPS. Najwięcej czasu potrzebował protokół HTTP z czasem 619 s. Przedstawione wyniki padań zaprezentowano na rysunku 8.



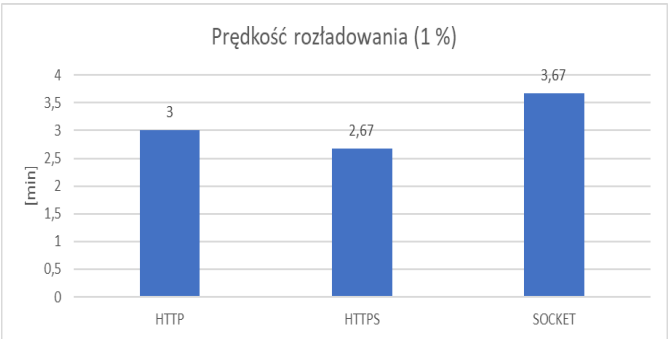
Rys. 7. Wykres z czasami pobrania zdjęcia



Rys. 8. Suma wszystkich zgromadzonych wyników

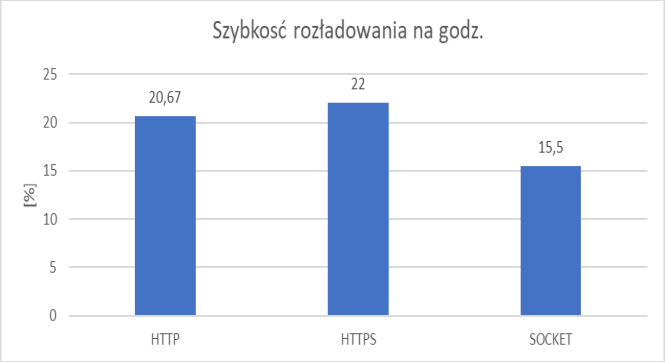
4.5. Pomiar użycia baterii

Pierwszym analizowanym przypadkiem jest czas w jakim bateria rozładuje się o 1%. Najlepszy wynik osiągnęła technologia gniazd, która potrzebowała 3,67 minuty do rozładowania analizowanej próbki. Pośrodku zestawienia znajduje się protokół HTTP z czasem 3 minut. Najszybciej rozładowuje się bateria przy komunikacji z wykorzystaniem protokołu HTTPS, w przeciągu 2,67 minuty rozładuje się 1% baterii na urządzeniu. Opisane wyniki przedstawiono na rysunku 9.



Rys. 9. Prędkość rozładowania 1% baterii

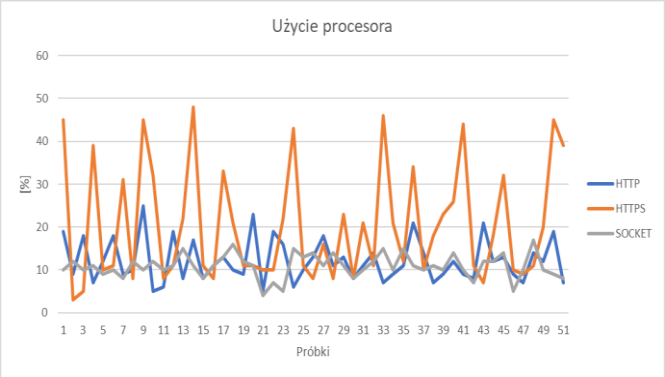
Drugim badanym aspektem jest ilość rozładowania wyrażona w procentach w przeciągu godziny. Najbardziej zostanie rozładowana bateria przy pracy z protokołem HTTPS, jest to około 22%. Zbliżoną wartość 21% osiągnięto przy komunikacji HTTP. Najmniejszą ilość zużycia baterii wynoszącą 15,5% odnotowano przy analizie technologii gniazd. Wyniki pokazano na rysunku 10.



Rys. 10. Szybkość rozładowania baterii

4.6. Pomiar użycia procesora

Zebrane wyniki oscylowały w okolicy 15% dla wszystkich badanych metod komunikacji. Najlepszy średni wynik użycia procesora uzyskała technologia gniazd z wartością 11%. Drugie miejsce zajął protokół HTTP z wynikiem 13%, rezultat protokołu HTTPS wyniósł 20% i stanowi największą wartość. Najbardziej stabilne wartości prezentuje rozwiązanie gniazd, różnica pomiędzy wartością maksymalną a minimalną wynosi 13% jest to o 246 % więcej od porównywanego protokołu HTTPS. Wykres prezentujący wyniki przestawia rysunek 11.



Rys. 11. Wykres użycia procesora

5. Wnioski

Analizując zebrane wyniki należy stwierdzić, że najbardziej wydajną technologią spośród testowanych są gniazda serwera. Osiągnęły we wszystkich eksperymentach najlepsze rezultaty. Różnica względem drugiego rozwiązania w zestawieniu (protokołu HTTP) jest niewielka i jej średnia ze wszystkich badań wynosi 27%. Porównanie technologii gniazda serwera do najwolniejszego rozwiązania wypada znacznie gorzej na rzecz protokołu HTTPS, tutaj różnica wynosi 510%.

Protokół HTTPS okazał się najwolniejszą metodą komunikacji. We wszystkich eksperymentach osiągnął najdłuższe czasy. Jedynym zadaniem w którym metoda była lepsza od protokołu HTTP, było pobranie dużej ilości danych w postaci zakodowanego zdjęcia. Wpływ na słabe wyniki ma ustanawianie przez protokół TLS połączenia szyfrowanego.

Najmniej czasu na rozładowanie 1% baterii potrzebuje aplikacja korzystająca z protokołu HTTPS (tj. 2,67 minut), faworytem w tym zestawieniu są gniazda z czasem 3,67 minuty. Najbardziej rozładuje się bateria w przeciągu godziny podczas pracy z technologią szyfrowaną (22%), drugi w zestawieniu z wartością 20,67% jest protokół HTTP, na pierwszym miejscu uplasowała się metoda gniazd serwera. Wyniki otrzymane podczas badania obciążenia procesora oscylowały w granicach 15%. Najmniej stabilne wyniki osiągnął protokół HTTPS, związane jest to z ustanawianiem połączenia szyfrowanego.

W nawiązaniu do powyższych wniosków można stwierdzić która z technologii jest najwydajniejsza, lecz to nie powinno mieć kluczowego wpływu na wybór metody komunikacji. Jeżeli projektowana aplikacja nie działa w sieci Intranet, a dane którymi operuje są wrażliwe powinno się przede wszystkim zadbać o bezpieczeństwo danych bez względu na koszty z tym związane.

Literatura

[1] http://zstzbaszynek.pl/blog/2021,era-systemow-klient-serwer/ [11.01.2018]

[2] A. Sawant, B. Meshram: Network programing in Java using Socket, Google Scholar, 2013.

[3] D. Naylor, A. Finamore, I. Leontiadis, Y. Grunenberger, M. Mellia, M. Munafo, K, Papagiannaki, P. Steenkiste: The Cost of the „s" in HTTPS, Google Schoolar, 2015.

[4] A. Serafinowicz: Android – historia prawdziwa, http://softonet.pl/publikacje/poradniki/Android-historia.prawdziwa, 1162, 2015.

[5] M. Oltrogge, Y. Acar, s. Dechand, M. Smith, s. Fahl: To Pin or Not to Pin – Helping App Developers Bullet Proof Their TLS Connections, Google Schoolar, 2015.

[6] J.Smołka: Programowanie aplikacji dla systemu Android, Politechnika Lubelska, 2014.

[7] B. Sosinsky: Networking Bible, Wiley Publishing, Inc.2009.

[8] T. Dierks, E. Rescorla: The Transport Layer Security (TLS) Protocol, RFC5246, 2008.

[9] http://edu.pjwstk.edu.pl/wyklady/mpr/scb/W8/W8.htm [20.01.2018]

[10] W. Frank Ableson, R. Sen, C. King, C. Enrique Ortiz: Android in action. Third editon, Manning Publications Co., 2011

[11] https://www.tutorialspoint.com/unix_sockets/what_is_socket.htm [01.01.2018]

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 8 (2018) 252-257
Wysłane: 2018-05-27
Przyjęte: 2018-06-12

Porównanie narzędzi automatyzacji testów z wykorzystaniem interfejsu użytkownika na przykładzie Sikuli i AutoIT

Tomasz Paczuski*, Beata Pańczyk
Politechnika Lubelska, Instytut Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. Celem artykułu jest porównanie narzędzi do automatyzacji testów interfejsu użytkownika na przykładzie Sikuli oraz AutoIT. W przeprowadzonych badaniach skoncentrowano się na czasie wykonywania skryptów testowych, ich złożoności, łatwości utrzymania oraz niezawodności. Na potrzeby badań stworzono aplikację w języku C# oraz napisano kilka reprezentatywnych testów automatycznych w każdym z narzędzi.

Słowa kluczowe: Sikuli; AutoIT; testy automatyczne

*Autor do korespondencji.

Adres e-mail: tpaczuski04@gmail.com

Comparison of tools for automated tests of the graphical user interface using the the Sikuli and AutoIT example

Tomasz Paczuski*, Beata Pańczyk
Institute of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract. The aim of the article is to compare the tools for automating user interface tests on the example of Sikula and AutoIT. The conducted research has focused on the time of testing scripts, their complexity, ease of maintenance and reliability. For the purposes of the study, an application in C # was created and several representative automatic tests were written in each tool.

Keywords: Sikuli; AutoIT; automated testing

*Corresponding author.

E-mail address: tpaczuski04@gmail.com

1. Wstęp

W dzisiejszym świecie oprogramowanie towarzyszy nam na każdym kroku. Można je spotkać praktycznie w większości urządzeń codziennego użytku. Niepoprawnie działające oprogramowanie może doprowadzić do utarty zaufania do producenta systemu, a co za tym idzie strat finansowych. Dlatego też firmy wytwarzające oprogramowanie zaczęły przywiązywać większą wagę do jakości wytwarzanego systemu. W sylabusie ISTQB [1] podane są intencje testów. Norma IEEE 829 [2] opisuje testowanie jako proces oparty o dokumentację oraz zawiera opis elementów planu testu. Według Myers [3] testowanie jest procesem uruchamiania oprogramowania w celu znajdywania błędów. Jednak nie do końca tak jest. Testowanie nie powinno tylko skupiać się na samym oprogramowaniu, ale również dodatkowo na innych elementach procesu wytwórczego [4].

Aby usprawnić proces testowy można wdrożyć narzędzia. Jak podaje [5] narzędzia do automatyzacji można wykorzystać np. do wykonania testów poprzez uruchomienie skryptów testowych, porównania oczekiwanych wyników testów z rzeczywistymi. Wdrożenie narzędzia do organizacji jest trudnym procesem, niesie ze sobą szereg korzyści, jak i ryzyk. Jak podaje [1], [6] przykładowymi ryzykami mogą być np. błędne oszacowanie kosztów oraz czasu wdrożenia narzędzia, błędne oczekiwanie w stosunku co do użycia narzędzia. Natomiast przykładowymi korzyściami wynikającymi ze wsparcia procesu testowego przez narzędzia to, np. wykonywanie testów zawsze w tej samej kolejności, redukcja monotonności pracy [6]. Wdrażając wybrane narzędzie - firma musi liczyć się również z kosztami. Przykładowe koszty, jak podaje [5], [7], to koszt opłat licencji i/lub wsparcia technicznego, koszt zakupu narzędzia, koszt związany z wytworzeniem własnego narzędzia.

Automatyzacja testów ma swoje metryki, dzięki którym można ocenić zyski lub przewidzieć kiedy testy zaczną je przynosić. Sylabus ISTQB dla inżyniera automatyzacji [8] przytacza m.in. takie metryki jak: czas trwania testów automatycznych, pracochłonność utrzymania testów, liczbę testów z wynikiem pozytywnym oraz negatywnym.

2. Przedmiot badań

Przedmiotem badań były dwa narzędzia do automatyzacji testów z wykorzystaniem interfejsu użytkownika. Wybrane narzędzia to SikuliX oraz AutoIT v3. Pierwsze z nich jest darmowe i działa pod systemami operacyjnymi Windows, Mac, Linux/Unix. Wykorzystuje rozpoznawanie obrazu oparte na bibliotece OpenCV do identyfikacji i sterowania

komponentami GUI. Narzędzie wspiera takie języki jak: Ruby, JavaScript, Python [9], [10].

AutoIT jest również darmowym narzędziem. Oparty jest o język podobny do BASIC. Przeznaczony jest jednak tylko na system Windows. Automatyzacja w tym narzędziu polega na symulacji użycia myszki oraz klawiatury. Napisane skrypty można kompilować do plików exe [11], [12].

3. Przeprowadzone badania

W celu porównania narzędzi została stworzona aplikacja desktopowa w języku C#. Przedmiotem badań są testy GUI. Aplikacja podzielona jest na moduły: użytkownika, pacjenta, lekarza. Aplikacja umożliwia:

- zalogowanie się,
- dodanie użytkownika,
- dodanie pacjenta,
- dodanie lekarza,
- rejestrację pacjenta.

Napisane skrypty testują poprawność działania interfejsu użytkownika w zakresie:

- test 1 - logowania na konta użytkownika admin,
- test 2 - niepoprawnego logowania,
- test 3 - wylogowania,
- test 4 - dodania użytkownika,
- test 5 - niepoprawnego dodania użytkownika,
- test 6 - dodania lekarza,
- test 7 - dodania pacjenta,
- test 8 - rejestracji pacjenta,
- test 9 - niepoprawnej rejestracji pacjenta.

Każdy z powyższych testów został uruchomiony 20 razy.

3.1. Kryteria porównania narzędzi

Jako kryteria, według których dokonano porównania narzędzi SikuliX oraz AutoIT v3 zostały przyjęte:

- czas wykonywania się skryptów przy uzyskaniu tego samego pokrycia,
- złożoność skryptu (liczba linii kodu),
- łatwość utrzymania,
- niezawodność skryptów przy wielokrotnym ich wykonaniu.

Powyższe kryteria wykorzystano do potwierdzenia tezy:
Nrzędzie AutoIT v3 jest bardziej efektywne w automatycznych testach GUI w porównaniu do SikuliX.

3.2. Platforma testowa

Tabela 1 przedstawia specyfikację maszyny, na której były uruchamiane testy.

Tabela 1. Specyfikacja maszyny testowej

Procesor	Intel Pentium T4400
Taktowanie procesora	2,20 GHz
Liczba rdzeni	2
RAM	8 GB
Rozdzielczość	1366 x 768
System operacyjny	Windows 7 Professional 64-bit SP1

4. Wyniki badań

4.1. Czas wykonywania się skryptów

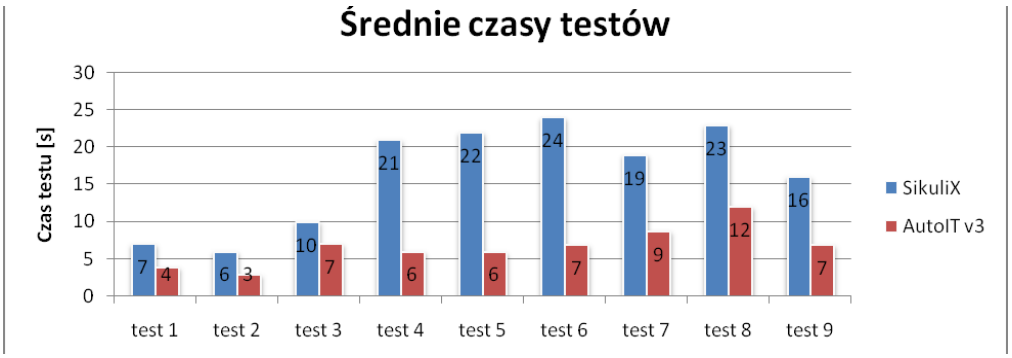
Na rysunku 1 zestawiono średnie czasy, jakie otrzymano dla poszczególnych testów w obu narzędziach. AutoIT okazało się najszybsze we wszystkich testach. Najszybszym testem okazał się test *Błędne logowanie*. Wykonywał się on w czasie 3 s., podczas gdy w SikuliX 6 s. Najdłużej wykonującym się skryptem w AutoIT był test *Rejestracja pacjenta*. Średni czas, jaki osiągnięto dla niego w tym narzędziu to 12 s. W SikuliX test ten okazał się drugim najwolniejszym testem, a trwał on 23 s. Natomiast najwolniejszym testem dla tego narzędzia okazał się test *Dodanie pacjenta*. Wykonywał się on sekundę dłużej niż skrypt wspomniany wcześniej, czyli 24 s.

Rysunek 2 prezentuje zestawienie czasów otrzymanych dla narzędzia SikuliX. Dla jednego z uruchomień testu *Dodanie pacjenta* otrzymano czas 33 s. Wynik ten okazał się największym czasem jaki osiągnięto spośród wszystkich uruchomień skryptów w SikuliX. Natomiast dla testu *Logowanie* oraz *Błędne logowanie* otrzymano w uruchomieniach najkrótszy czas, a wynosił on 4 s.

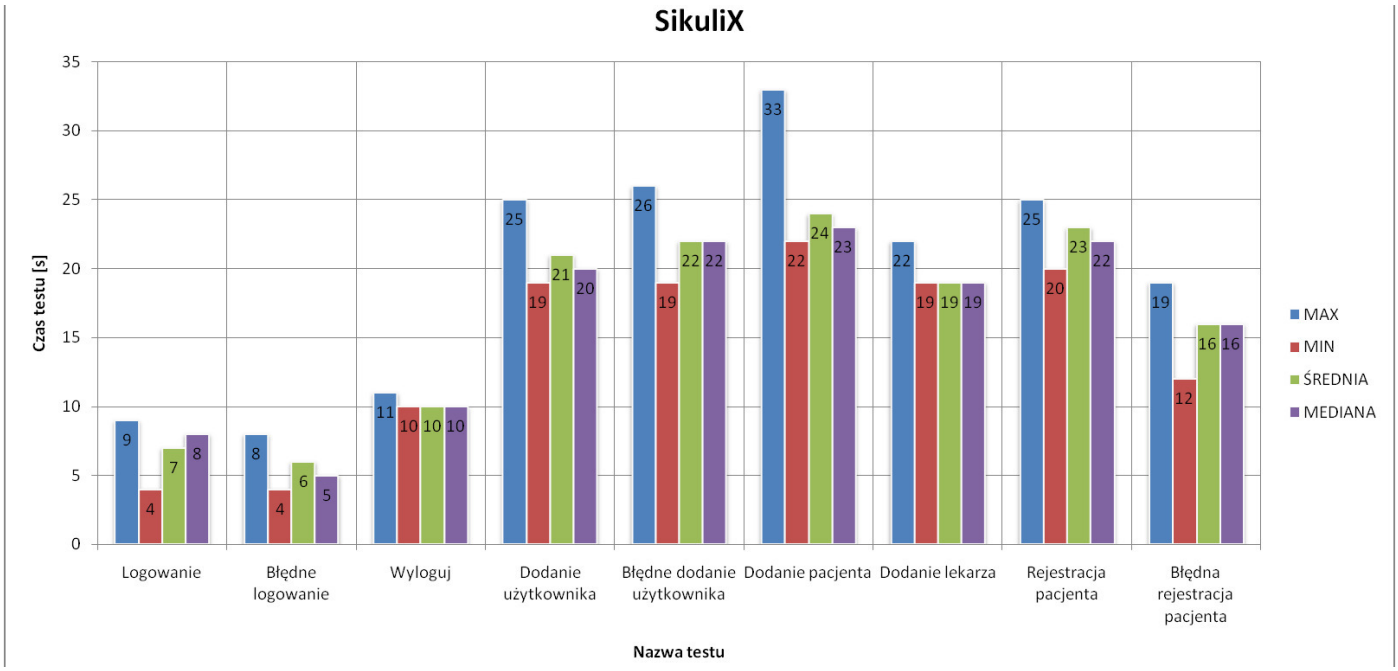
Na rysunku 3 przedstawiono zestawienie czasów otrzymanych w AutoIT. Dla testu *Rejestracja pacjenta* otrzymany czas podczas jednego z uruchomień skryptu okazał się największym wynikiem spośród wszystkich otrzymanych. Wynosił on 14 s. Najkrótszy czas uzyskano pośród wykonywania się testu *Błędne logowanie*. Wynik ten wynosi 2 s.

4.2. Złożoność skryptu

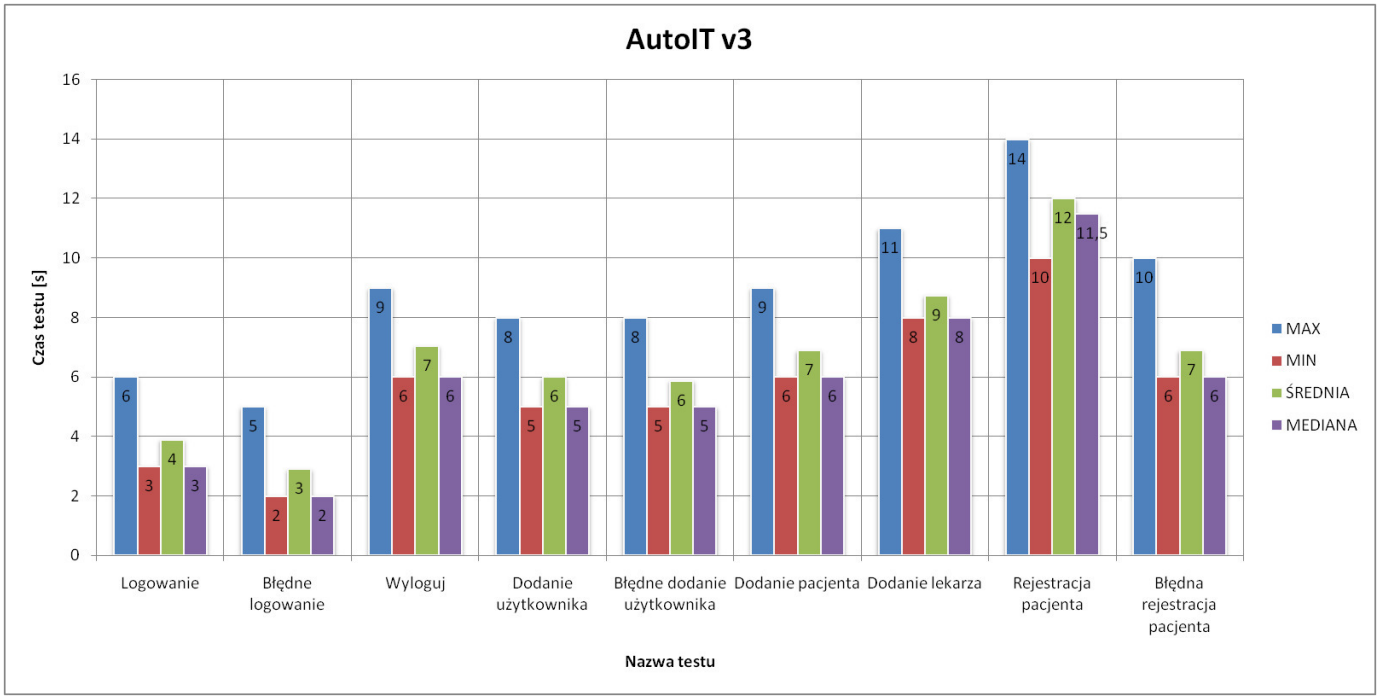
Na rysunku 4 przedstawiono zbiorcze zestawienie liczby linii kodu dla wszystkich testów napisanych w obu narzędziach. Testy napisane w SikuliX okazały się zajmować mniejszą liczbę linii kodu w porównaniu do AutoIT v3. Najmniejszą liczbę linii kodu w tym narzędziu otrzymano dla testu *Błędne logowanie*, a wynosiła ona 10 wierszy. Najdłuższym testem napisanym w SikuliX okazał się *Rejestracja pacjenta*. Liczył on 34 linie. Jednocześnie dla tego testu otrzymano największą różnicę w długości skryptu pomiędzy narzędziami. Test ten napisany w AutoIT zajmuje 59 linii, a więc różnica wynosi 25 wierszy. Najkrótsze testy w tym narzędziu to testy dotyczące logowania: *Logowanie* oraz *Błędne logowanie*, które zajmują odpowiednio 15 oraz 14 linii kodu. Są to jedyne testy, które pod względem złożoności są najbardziej zbliżone do testów napisanych w SikuliX.



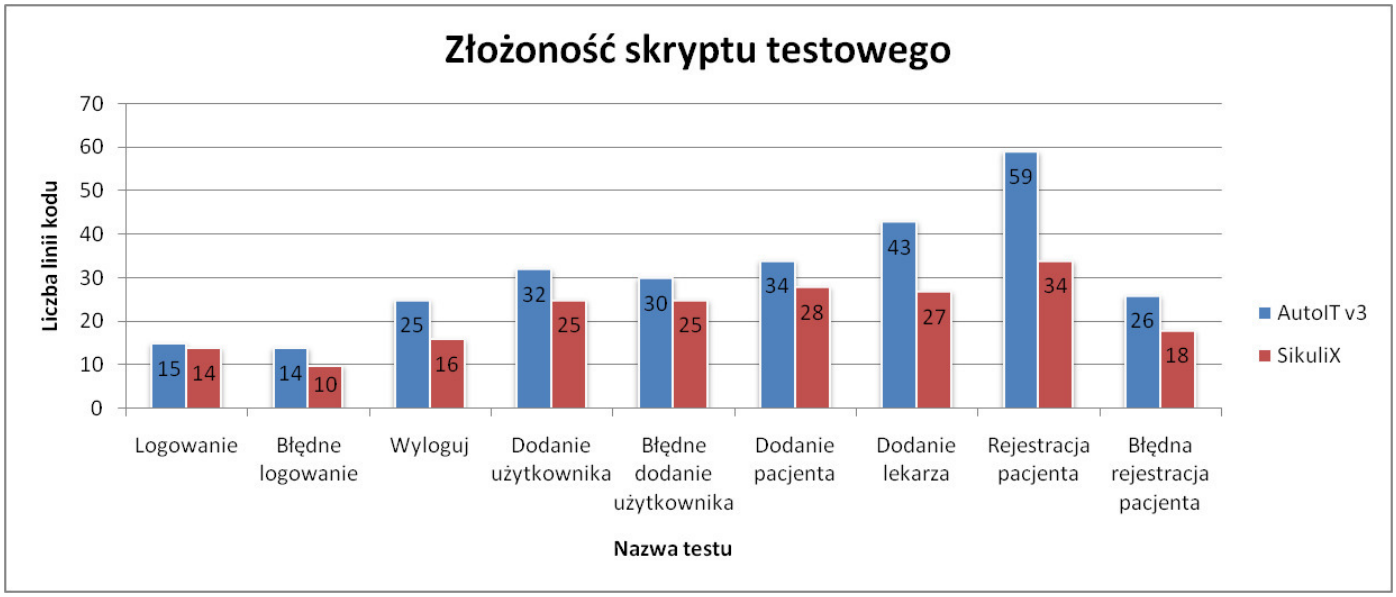
Rys.1. Średnie czasy testów w obu narzędziach



Rys. 2. Czasy otrzymane dla SikuliX



Rys. 3. Czasy otrzymane dla AutoIT



Rys. 4. Zestawieni liczby linii kodu skryptów testowych w obu narzędziach

4.3. Łatwość utrzymania

Następnym badanym aspektem była łatwość utrzymania skryptu testowego. Cecha ta jest niejako związana z samym zaprojektowaniem i sposobem napisania testu. Sam test musi być przemyślany, trzeba zaplanować, co dany test ma robić. Przy pisaniu testu warto koncentrować się na jak najprostszym sposobie, dzięki czemu wprowadzenie ewentualnych zmian w późniejszym życiu testu będzie łatwiejsze.

SikuliX opiera się na rozpoznawaniu obrazu, a w skryptach testowych wykorzystuje się zrzuty ekranu GUI. Testy w tym narzędziu są podatne na jakiekolwiek, nawet najmniejsze modyfikacje interfejsu użytkownika, zarówno w samym wyglądzie, jaki i rozmieszczeniu poszczególnych kontrolek. Wszystko to sprawia, że utrzymanie testów w tym narzędziu jest czasochłonne i mozolne. Dlatego warto zastanowić się tu, w jaki sposób można zaprojektować test. Jednym z rozwiązań jest wykonywanie tam gdzie to możliwe oddzielnych zrzutów ekranu dla poszczególnych elementów GUI, zamiast jednego, który uchwyci, np. całe okno, gdzie trzeba będzie wykonywać zmiany współrzędnych dla poszczególnych akcji. Wywoływanie innych skryptów z poziomu drugiego testu również wpłynie pozytywnie na łatwość utrzymania. Jeżeli skrypty wykonują jakąś wspólną część - warto ją zaimplementować jako oddzielny test, a wywoływać go w miejscach potrzebnych. Gdyby coś się zmieniło w tym skrypcie to potrzeba naniesienia zmian będzie tylko w jednym miejscu, a nie we wszystkich.
Testy napisane w AutoIT v3 wykorzystują symulację użycia myszy lub klawiatury komputerowej na elementach interfejsu użytkownika. Aby zdarzenie mogło zaistnieć narzędzie musi wiedzieć na jakim elemencie ma wykonać akcję. Identyfikacja elementów GUI odbywa się poprzez podanie jego identyfikatora, którym może być np.: nazwa elementu (np. nazwa okna), id kontrolki nadany na etapie kodowania, tekst widoczny na elemencie. Użycie w komendach bezpośrednio identyfikatora elementu, powoduje, że skrypt staje się nieczytelny oraz jest trudny w utrzymaniu. Niejednokrotnie w różnych skryptach identyfikatory będą się powielać, dlatego rozwiązaniem, które znacząco polepszy utrzymanie testów jest stworzenie oddzielnego pliku ze zmiennymi, do których zostaną przypisane identyfikatory elementów GUI użytych w skrypcie. Na rysunku 5 przedstawiono przykładowy plik ze zmiennymi.

```
1  $imie = "[NAME:imie_txtb]"
2  $nazwisko = "[NAME:nazwisko_txtb]"
3  $haslo2 = "[NAME:haslo_txtb]"
4  $potw_haslo = "[NAME:potwHaslo_txtb]"
5  $akceptuj = "[NAME:akceptuj_btn]"
6  $anuluj = "[NAME:anuluj_btn]"
7  $dodaj_uzyt = "[Title:Dodaj użytkownika]"
8  $potw_dod_uzyt = "[Title:Dodanie użytkownika]"
9  $ok = "[Text:OK]"
10 $msg_bledne_dane = "[Title:Niepoprawne dane]"
11 $err_prov = "[CLASS:WindowsForms10.Window.8.app.0.141b42a_r15_ad1; INSTANCE:1]"
```

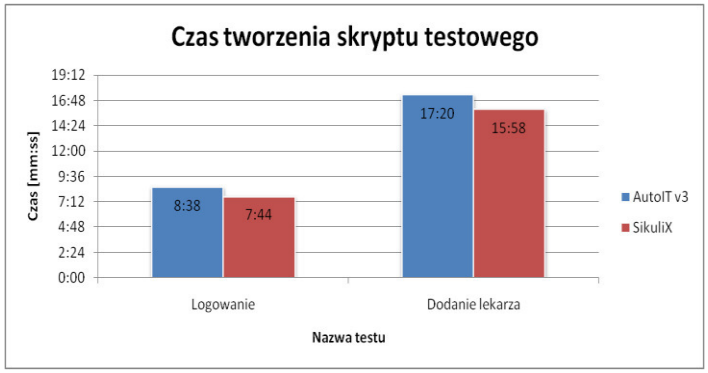
Rys. 5. Przykładowy plik ze zmiennymi

Na rysunku 6 przedstawiono czas, w jakim zostały napisane testy *Logowanie* oraz *Dodanie lekarza* celem zobrazowania czasochłonności. Czas poświęcony na tworzenie wybranych skryptów w obu narzędziach był zbliżony do siebie, jednak w narzędziu SikuliX testy powstawały szybciej. Napisanie skryptu testowego *Logowanie* w SikuliX zajęło autorowi 7 min i 44 s., a w AutoIT 8 min 38 s. Test w AutoIT powstawał 54 s. dłużej. Natomiast bardziej złożony test, jakim jest *Dodanie lekarza* wymagał ponad dwa razy tyle czasu. Napisanie go w SikuliX zajęło 15 min 58 s, podczas gdy w AutoIT 17 min 20 s.

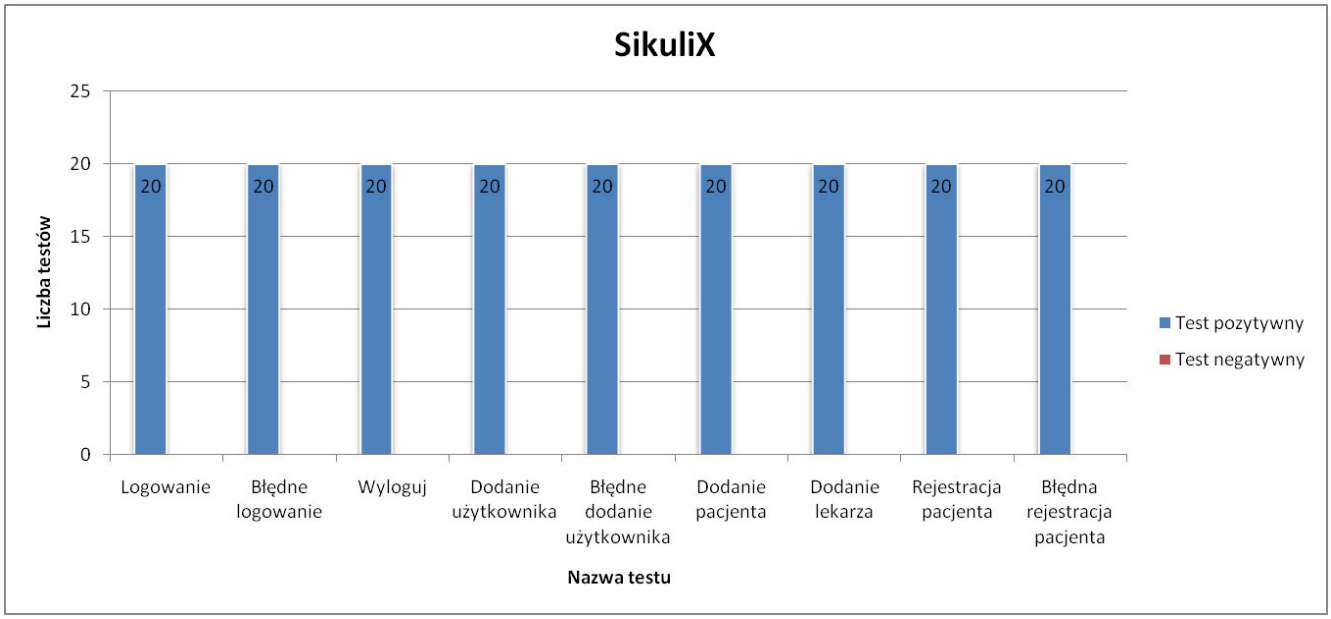
4.4. Niezawodność skryptów

W badanym aspekcie skoncentrowano się na liczbie testów, które zakończyły się powodzeniem. Każdy z testów został uruchomiony 20 razy, następnie zliczono liczbę uruchomień dających wynik pozytywny jak i negatywny. Na

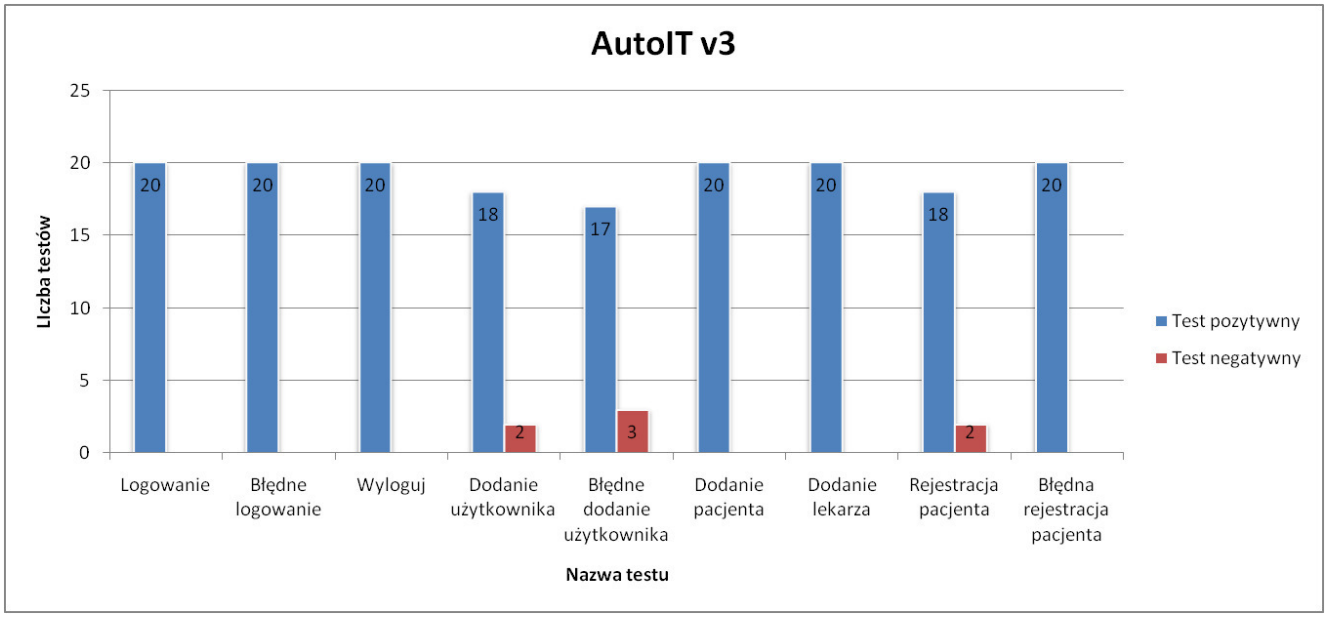
rysunku 7 oraz rysunku 8 przedstawiono wyniki dla poszczególnych narzędzi. Testy w SikuliX (rysunek 7) uzyskały 100% poprawności. Każde uruchomienie testu kończyło się wynikiem pozytywnym.
W przypadku testów napisanych w AutoIT v3 (rysunek 8) dla testów: *Logowanie*, *Błędne logowanie*, *Wyloguj*, *Dodanie pacjenta*, *Dodanie lekarza*, *Błędna rejestracja pacjenta* uzyskano przy każdym uruchomieniu wynik pozytywny. Natomiast dla testów *Dodanie użytkownika* oraz *Rejestracja pacjenta* otrzymano wynik pozytywny w 90%. *Błędne dodanie użytkownika* kończyło się największą liczbą niepowodzeń spośród wszystkich testów. Test ten trzy razy kończył się negatywnie, co daje 85% wyników pozytywnych.



Rys. 6. Czas pisania wybranych skryptów



Rys. 7. Niezawodność testów napisanych w SikuliX



Rys. 8. Niezawodność testów napisanych w AutoIT v3

5. Podsumowanie

Analizując otrzymane wyniki oraz odnosząc postawioną tezę do każdego z kryteriów oddzielnie można dostrzec, że teza znalazła potwierdzenie w kliku przyjętych kryteriach. Stąd oba narzędzia mają swoje słabe i mocne strony. AutoIT v3 okazało się narzędziem znacznie szybszym od SikuliX i to w niektórych testach kilkukrotnie. Jednakże testy w AutoIT były mniej niezawodne. W kilku przypadkach test kończył się niepowodzeniem. SikuliX pod względem niezawodności okazało się bezbłędne. Wszystkie uruchomienia poszczególnych testów dawały wynik pozytywny.

W aspekcie złożoności skryptów SikuliX również okazało się lepsze. Wszystkie skrypty w tym narzędziu były krótsze o klika, kilkanaście linii kodu. Różnica w głównej mierze wynika ze składni języka zastosowanego w narzędziach. Jednak analizując złożoność skryptów wraz z czasem ich wykonania można spostrzec, że liczba linii kodu nie ma znaczenia. Pomimo tego, że testy w AutoIT v3 były dłuższe to wykonywały się szybciej od SikuliX. Przykładem tu może być skrypt *Rejestracja pacjenta*. W AutoIT v3 zajmował 59 linii kodu, gdzie w SikuliX 34 linii. Jednak czas wykonywania był w przybliżeniu dwa razy krótszy na korzyść AutoIT.

Łatwość utrzymania wynika m.in. z samej konstrukcji narzędzi. Testy w AutoIT opierają się o identyfikatory elementów interfejsu użytkownika. Z racji, że zmieniają się one rzadko to testy w tym narzędziu są stabilniejsze od skryptów w SikuliX, które opierają się na rozpoznawaniu obrazu, a warstwa graficzna interfejsu użytkownika jest podatna na zmiany. Wprowadzanie ewentualnych zmian do skryptów tworzonych w tych narzędziach jest porównywalnie czasochłonne.

W ogólnym rozrachunku oraz przy badanej aplikacji lepszym rozwiązaniem jest oparcie testów GUI o narzędzie AutoIT. Aplikacja jest desktopowa na system Windows. Jest tu dostęp do identyfikatorów elementów GUI, a co za tym idzie skrypty będą mniej podatne na zmiany w warstwie graficznej. Testy będą szybsze i dodatkowo można je skompilować do plików exe.

Literatura

[1] Stowarzyszenie Jakości Systemów Informatycznych, Certyfikowany tester, Plan poziomu podstawowego, wersja 2011.1.1.

[2] IEEE 829 Standard for Software and System Test Documentation, IEEE Computer Society, 2008.

[3] Myers G. J., The Art of Software Testing, John Wiley & Sons, Inc., 2004.

[4] http://testerzy.pl/artykuly/definicja-testowania-oprogramowania-cz-1 [11.09.2017.].

[5] Roman A., Testowanie i jakość oprogramowania. Modele, techniki, narzędzia, Wydawnictwo Naukowe PWN, 2016.

[6] Black R., Advanced Software Testing Vol. 2, Guide to the ISTQB Advanced Certification as an Advanced Test Manager, Rock Nook Inc., 2011.

[7] Stowarzyszenie Jakości Systemów Informatycznych, Certyfikowany tester. Sylabus dla Poziomu Zaawansowanego. Kierownik Testów, werjsa 2012.

[8] International Software Testing Qualifications, Board Certified Tester Advanced Level Syllabus - Test Automation Engineer, 2016.

[9] http://sikulix.com/ [28.02.2018].

[10] https://media.readthedocs.org/pdf/sikulix-2014/latest/sikulix-2014.pdf.

[11] https://www.autoitscript.com/site/autoit/ [28.02.2018].

[12] https://opensourceforu.com/2017/01/autoit/ [dostęp 28.02.2018].

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 8 (2018) 258-262
Wysłane: 2018-05-29
Przyjęte: 2018-06-19

Porównanie modeli hostowania aplikacji na platformie ASP.NET Core

Kamil Zdanikowski*, Beata Pańczyk
Politechnika Lubelska, Instytut Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. W artykule przedstawione zostało porównanie modeli hostowania aplikacji na platformie ASP.NET Core. Dostępne modele hostowania zostały opisane i porównane, a następnie przeprowadzone zostały ich testy wydajnościowe. Dla każdego modelu zastosowano scenariusze testowe realizujące te same funkcje, a ich wydajność została określona za pomocą liczby przetwarzanych żądań na sekundę. Uzyskane rezultaty wskazują, że standardowa konfiguracja jest najmniej wydajna, a zastosowanie innej, np. IIS z serwerem Kestrel, czy same serwery Kestrel lub HTTP.sys mogą zapewnić kilkukrotny wzrost wydajności w porównaniu z modelem standardowym.

Słowa kluczowe: asp.net core; .net; model hostowania; iis; kestrel

*Autor do korespondencji.

Adres e-mail: kamil.zdanikowski@pollub.edu.pl

Hosting models comparison of ASP.NET Core application

Kamil Zdanikowski*, Beata Pańczyk
Institute of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract. The article presents hosting models comparison of ASP.NET Core application. Available hosting models were described and compared and then performance comparison was carried out. For each model the same test scenarios were executed and their performance was determined by number of requests per second which host was able to process. The results obtained show that standard model is the least efficient one and using one of the other configurations, for example, IIS with Kestrel (in-process), Kestrel or HTTP.sys might provide even several times better performance compared to standard model.

Keywords: asp.net core; .net; hosting model; iis; kestrel

*Corresponding author.

E-mail address: kamil.zdanikowski@pollub.edu.pl

1. Wstęp

ASP.NET Core to nowa platforma stworzona przez Microsoft, służąca do tworzenia aplikacji internetowych działających po stronie serwera. Pomimo tego, że nazwiązuje ona nazwą do poprzednika (ASP.NET 4.6 MVC) jest to jednak platforma napisana w całości od nowa.

Jej powstanie związane było bezpośrednio z powstaniem platformy .NET Core, czyli odpowiednika .NET Framework, który może działać nie tylko na systemie Windows, ale również na systemach Linux i macOS. Z tego powodu postanowiono napisać nową platformę ASP.NET Core – która również może działać na wielu systemach operacyjnych – i porzucić stare nazewnictwo (zgodnie z którym nowa platforma nazywałaby się ASP.NET 5) [1].

Z takimi zmianami musiały wiązać się również zmiany po stronie hostowania aplikacji. Aplikacje napisane przy użyciu poprzedniej wersji platformy hostowane mogły być tylko na serwerze IIS (Internet Information Services), który stworzony został przez Microsoft i uruchomiony mógł być jedynie na systemie operacyjnym Windows. W przypadku platformy ASP.NET Core postanowiono stworzyć nowy sposób hostowania, w którym aplikacja uruchamiana jest ze swoją wewnętrzną implementacją serwera HTTP (np. *Kestrel*). Dzięki temu możliwe jest uzyskanie jednakowego zachowania na każdej platformie (Windows, Linux, macOS) [2].

Z uwagi na prostotę oraz niedojrzałość niektórych serwerów wewnętrznych dostępnych w ASP.NET Core nie zawsze zalecane jest wystawianie ich bezpośrednio do sieci publicznej. W takich przypadkach stosuje się model, gdzie przed aplikacją z wewnętrznym serwerem znajduje się tzw. serwer reverse proxy (np. IIS), którego celem jest wstępne przetworzenie zapytania oraz przekazanie go dalej do serwera wewnętrznego.

W niniejszym artykule przedstawione zostaną najpopularniejsze modele hostowania aplikacji na platformie ASP.NET Core oraz przeprowadzone zostaną ich testy wydajnościowe. Dla każdego modelu uruchomiony zostanie ten sam zestaw testów, a wynikiem będzie liczba zapytań na sekundę, które możliwe były do przetworzenia przez aplikację w danym modelu hostowania.

W sieci można znaleźć testy wydajnościowe platformy ASP.NET Core, ale nie pokrywają one wszystkich dostępnych modeli hostingowych. Wyniki przedstawione w jednym z artykułów [3] dostępnych w internecie pokazują, że wykorzystanie serwera Kestrel zamiast serwera IIS może zapewnić nawet kilkukrotny wzrost wydajności.

2. Platforma oraz scenariusze testowe

Do przeprowadzenia testów wydajności wykorzystana została przygotowana do tego celu autorska aplikacja, która uruchamiana była w różnych modelach hostowania. Za każdym razem wykorzystywane były te same scenariusze testowe.

2.1. Konfiguracja środowiska i wykorzystane narzędzia

Konfiguracja sprzętowa środowiska testowego oraz wykorzystane narzędzia i biblioteki wraz z ich wersjami znajdują się w tabelach 1 i 2.

Tabela 1. Konfiguracja sprzętowa środowiska testowego

System operacyjny	Windows 10 x64
CPU	Intel Core i3-3110M
Pamięć RAM	12 GB DDR3

Tabela 2. Wykorzystane biblioteki i środowiska uruchomieniowe

Narzędzie	**Wersja**
.NET Core Runtime	2.1.300
ASP.NET Core	2.1.0-preview1
IIS	10.0.17134.1
Kestrel	2.1.0-preview1
k6	0.20.0

Do przetestowania wydajności użyte zostało narzędzie k6 wyprodukowane przez firmę Load Impact. Jest to darmowe oraz proste w obsłudze narzędzie umożliwiające generowanie ruchu http [4]. Umożliwia ono konfigurację kilku parametrów – wykorzystywane w testach parametry to:

- *vus* – wirtualni użytkownicy (w praktyce ilość pętli wysyłających zapytania)
- *duration* – czas trwania testu

Konfiguracja narzędzia oraz listing wykorzystywanego skryptu, który wysyła zapytania pod określony adres i sprawdza czy kod odpowiedzi jest równy 200 znajdują się odpowiednio w tabeli 3 i na listingu 1.

Tabela 3. Konfiguracja narzędzia k6

Parametr	**Wartość**
vus	8
duration	15s

Wartość parametru *vus* dobrano w taki sposób, aby wysycenie procesora podczas testów utrzymywało się na poziomie 100%. Czas testu ustalono na 15s, aby uniknąć wpływu krótkotrwałych niestabilności na wpływ całego testu. Przy czasie ustawionym na 15s testy pokazały, że uzyskiwane wartości dotyczące ilości zapytań na sekundę są stabilne i nie zmieniają się znacząco pomiędzy kolejnymi wywołaniami testu.

Przykład 1. Skrypt narzędzia k6 wykorzystywany podczas testów

```
import { check } from "k6";
import http from 'k6/http';
export default function() {
  const response = http.get(ADRES_APLIKACJI);
  check(response, {
    "is status 200": (r) => r.status === 200
  });                 };
```

2.2. Scenariusze testowe

W celu przetestowania wydajności przygotowane zostało kilka scenariuszy testowych. Obejmują one standardowe czynności wykonywane przez serwery aplikacji internetowych, czyli:

- generowanie dynamicznego widoku HTML,
- zwracanie danych w postaci czystego tekstu,
- zwracanie danych w formacie JSON,
- zwracanie danych w formacie XML.

Generowany widok HTML składa się ze 100 przycisków oraz 10 list wyboru (każda z 10 opcjami). Scenariusz testujący wydajność zwracania danych w postaci czystego tekstu zwraca wszystkie znaki alfabetu łacińskiego i cyfry. Przykładowe modele zwracane jako JSON i XML przedstawione zostały na listingach 2 i 3.

Przykład 2. Przykładowa struktura danych zwracanych jako JSON

```
{   "id": "5822a71f-d28e-449c-a220-d282239f9d09",
    "stringValue": "5822a71f",
    "intValue": 1580445059,
    "listOfInts": [
        1790594724,
        428614593,
        978875039,
        437814156,
        556857108,
        1579485844,
        904515399,
        1189173955,
        1344600976,
        159188863
    ]
}
```

Przykład 3. Przykładowa struktura danych zwracana jako XML

```
<SimpleClass
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <Id>7a9dcdfe-7c88-4a79-a8a9-53ea7e1437d6</Id>
    <StringValue>7a9dcdfe</StringValue>
    <IntValue>1933853648</IntValue>
    <ListOfInts>
        <int>546007864</int>
        <int>19000976</int>
        <int>1405144961</int>
        <int>921898622</int>
        <int>1267375503</int>
        <int>1710179990</int>
        <int>1652055825</int>
        <int>824934734</int>
        <int>1114826130</int>
        <int>136438785</int>
    </ListOfInts>
</SimpleClass>
```

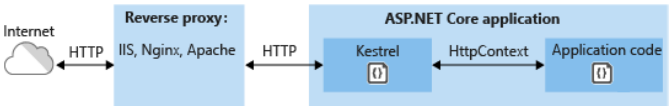
3. Modele hostowania aplikacji ASP.NET Core

Istnieje kilka możliwości hostowania aplikacji napisanych przy użyciu platformy ASP.NET Core. W kolejnych podrozdziałach przedstawione zostaną wykorzystane modele.

3.1. Kestrel + IIS (reverse proxy, out-of-process)

Ten model to standardowa i najpopularniejsza forma hostowania aplikacji ASP.NET Core. Kestrel wykorzystywany

jest jako wewnętrzny serwer HTTP, a dodatkowo przed nim wystawiony jest serwer IIS, który wstępnie przetwarza zapytanie i przekazuje je do Kestrela. Schemat konfiguracji przedstawiony został na rysunku 1 [5, 6].



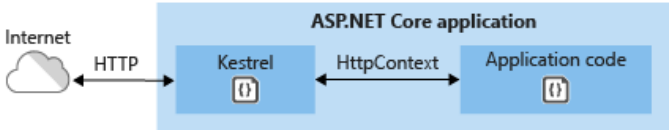
Rys. 1. Konfiguracja Kestrel + IIS (out-of-process) [6]

3.2. Kestrel + IIS (reverse proxy, in-process)

Ten model hostowania dostępny jest od wersji ASP.NET Core 2.1.0-preview1. W odróżnieniu od standardowej metody aplikacja z Kestrelem nie jest uruchamiana jako oddzielny proces, ale wewnątrz procesu IIS. Dzięki temu pominięty został narzut związany z przesłaniem zapytania HTTP z IIS do Kestrela i na odwrót pomiędzy procesami, co powinno pozytywnie wpłynąć na wydajność [7].

3.3. Kestrel

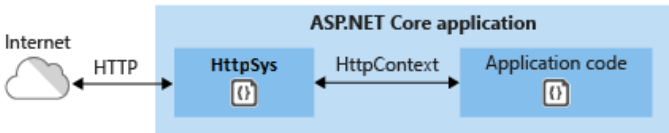
Model dostępny od początku istnienia platformy ASP.NET Core jednak przez długi czas niepolecany do stosowania w sieci zewnętrznej z uwagi na niedojrzałość serwera Kestrel (rozwijany jest dopiero od około 2 lat). Polega na hostowaniu aplikacji bez użycia pośredniczącego serwera reverse proxy. Umożliwia hostowanie aplikacji na dowolnym systemie operacyjnym. Schemat konfiguracji przedstawiony został na rysunku 2 [2].



Rys. 2. Konfiguracja Kestrel [2]

3.4. HTTP.sys

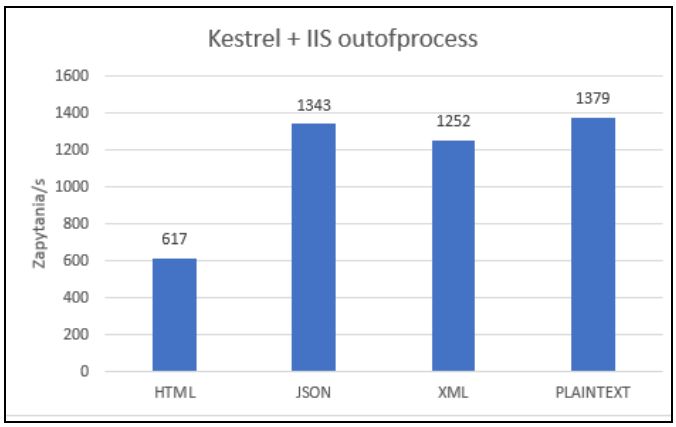
Podobnie jak Kestrel dostępny od początku istnienia ASP.NET Core (wcześniej używał nazwy WebListener). Możliwy do użycia tylko na systemie Windows z uwagi na wykorzystywanie jego wewnętrznych modułów obsługi HTTP. Użycie serwera HTTP.sys wyklucza użycie IIS jako reverse proxy, ponieważ nie są one ze sobą kompatybilne. Podobnie jak w przypadku IIS nie ma przeciwwskazań, aby wystawiać aplikację z tym serwerem do sieci zewnętrznej (HTTP.sys używany jest wewnętrznie przez IIS i rozwijany przez ponad 15 lat). Schemat konfiguracji przedstawiony został na rysunku 3 [2].



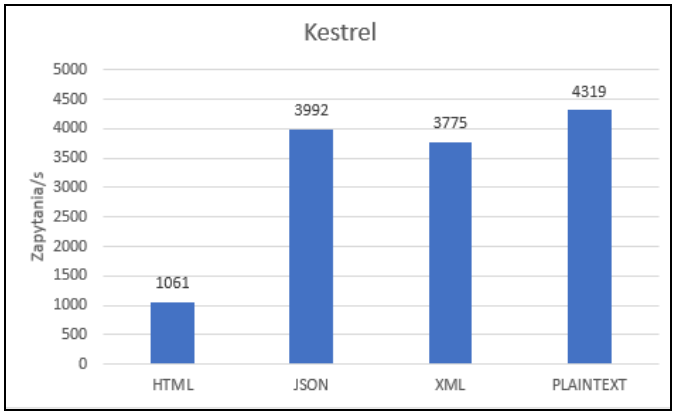
Rys. 3. Konfiguracja HTTP.sys [2]

4. Zestawienie wyników

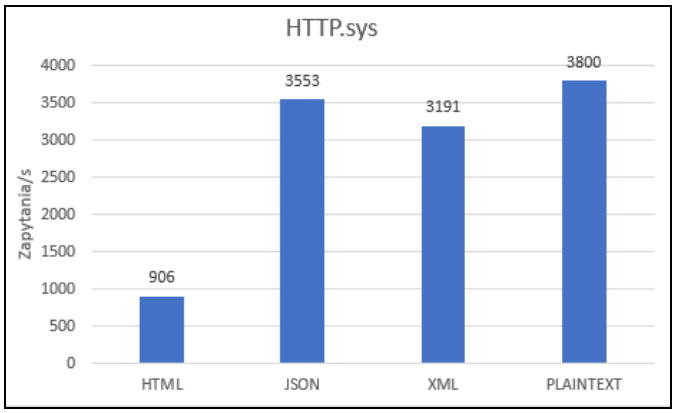
Wizualizacje wyników dla poszczególnych modeli hostowania i scenariuszy testowych przedstawione zostały w postaci wykresów na rysunkach 4, 5, 6 i 7.



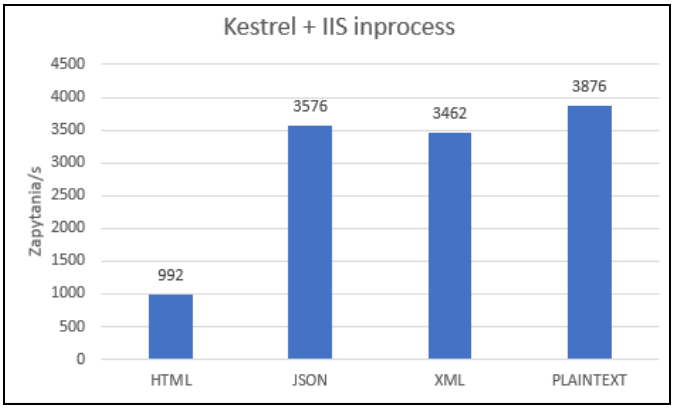
Rys. 4. Wykres dla modelu Kestrel + IIS (outofprocess)



Rys. 5. Wykres dla modelu Kestrel

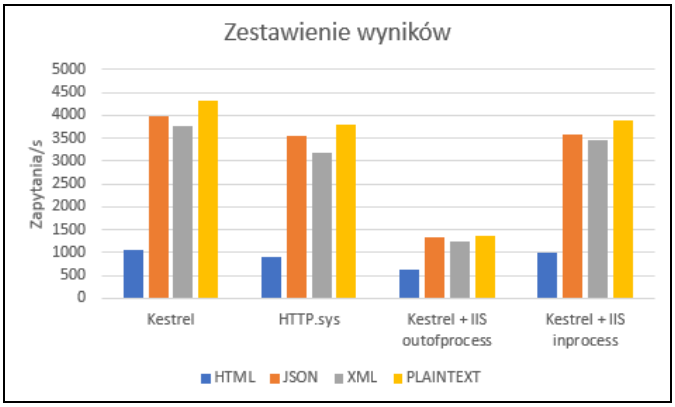


Rys. 6. Wykres dla modelu HTTP.sys



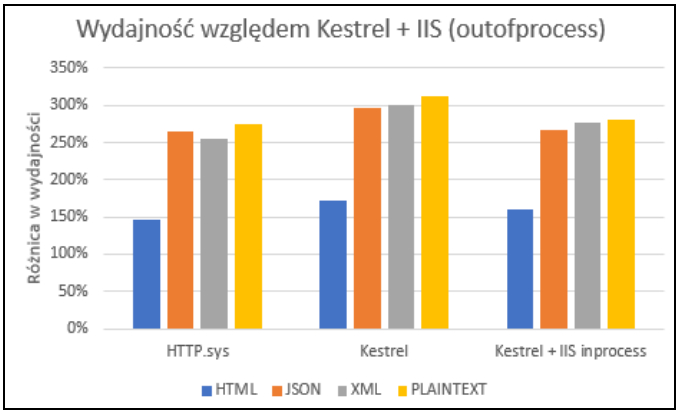
Rys. 7. Wykres dla modelu Kestrel + IIS (inprocess)

Na rysunku 8 znajduje się wykres ze zbiorczym zestawieniem wyników.



Rys. 8. Zbiorcze zestawienie wyników

Wydajność w porównaniu do standardowego modelu Kestrel + IIS (outofprocess) przedstawiona została na rysunku 9, gdzie 100% to wydajność modelu Kestrel + IIS (outofprocess).



Rys. 9. Wydajność względem modelu Kestrel + IIS (outofprocess)

5. Omówienie wyników

Standardowy oraz polecany model hostowania aplikacji to połączenie serwera wewnętrznego Kestrel z serwerem pośredniczącym (reverse proxy) IIS. Z testów wynika, że jest to najmniej wydajny model. Spowodowane jest to faktem, że zapytanie HTTP musi zostać przesłane dwukrotnie pomiędzy oddzielnymi procesami IIS oraz wewnętrznego serwera Kestrel. Pierwszy raz w momencie, gdy klient wysyła zapytanie do serwera, wtedy proces IIS wstępnie przetwarza zapytanie zgodnie z konfiguracją serwera IIS, a następnie przekazuje je do Kestrela. Drugi raz w momencie, gdy zapytanie zostanie przetworzone przez aplikację i odpowiedź jest gotowa do wysłania do klienta, wtedy to Kestrel przekazuje odpowiedź do IIS, a ten z kolei do klienta.

Kolejną mozliwością jest hostowanie aplikacji wyłącznie za pomocą serwera wewnętrznego Kestrel. Przez długi czas model ten polecany był do użycia tylko w przypadku aplikacji znajdujących się w sieci wewnętrznej ze względu na niedojrzałość serwera i możliwe braki w bezpieczeństwie oraz możliwościach konfiguracyjnych. Od wersji 2.0.0 serwera (czyli również od wersji 2.0.0 platformy ASP.NET Core) nadaje się on do wystawienia w sieci publicznej, jednak wciąż w porównaniu z IIS ma znikome możliwości konfiguracyjne. Dużą zaletą tego modelu jest możliwość hostowania aplikacji na dowolnym systemie operacyjnym. Z przeprowadzonych testów wynika, że jest to najszybsza ze wszystkich użytych konfiguracji. Na podstawie rysunku 9 można stwierdzić, że w porównaniu do standardowego modelu jest on wydajniejszy o około 70% w przypadku zapytań zwracających widok HTML i nawet 200% dla danych w postaci JSON, XML i zwykłego tekstu.

Następna konfiguracja to użycie wyłącznie serwera wewnętrznego HTTP.sys. Wadą takiego rozwiązania jest możliwość hostowania aplikacji wyłącznie na systemie operacyjnym Windows. Nie ma natomiast przeciwwskazań do wystawienia serwera HTTP.sys w sieci publicznej – serwer IIS wewnętrznie również korzysta z modułu HTTP.sys. Posiada on duże możliwości konfiguracyjne i jest rozwijany od ponad 15 lat. Po przeprowadzeniu testów można stwierdzić, że model ten umożliwia uzyskanie zdecydowanie wyższej wydajności od konfiguracji standardowej (50% w przypadku HTML i około 180% w przypadku JSON, XML i czystego tekstu). Jest on jednak mniej wydajny od serwera Kestrel o około 10-20%.

Kolejną ciekawą możliwością jest konfiguracja dostępna dopiero od wersji 2.1.0-preview1 platformy ASP.NET Core, która również polega na wykorzystaniu serwera wewnętrznego Kestrel i serwera pośredniczącego IIS. Jednak w odróżnieniu od modelu standardowego aplikacja z serwerem Kestrel nie jest uruchamiana wewnątrz własnego procesu, a wewnątrz procesu IIS. Dzięki takiemu rozwiązaniu zaobserwować można znaczny wzrost wydajności. Liczba przetwarzanych zapytań na sekundę jest mniejsza od konfiguracji z samym Kestrelem o około 10-20% jednak uzyskujemy dużo większe możliwości konfiguracyjne za sprawą wykorzystania IIS.

6. Wnioski

W artykule porównane zostały różne modele hostowania aplikacji napisanej przy użyciu platformy ASP.NET Core. Uniezależnienie się od serwera IIS oraz systemu operacyjnego

Windows to jeden z głównych elementów, który odróżnia platformę ASP.NET Core od jej poprzednika ASP.NET 4.6.

Najwydajniejszym z testowanych modeli okazał się model używający serwera Kestrel bez użycia serwera pośredniczącego reverse-proxy, jednak równocześnie jest to model posiadający dość małe możliwości konfiguracyjne. Najmniej wydajnym, ale z kolei zapewniającym największe bezpieczeństwo i możliwości konfiguracyjne jest model składający się z serwera wewnętrznego Kestrel oraz serwera reverse-proxy IIS.

Otrzymane wyniki zgadzają się z tymi, które przedstawione zostały w artykule znajdującym się w sieci [3]. W obu przypadkach zastosowanie samego serwera Kestrel, zamiast konfiguracji Kestrel + IIS (outofprocess) zapewniło kilkukrotny wzrost wydajności.

Należy zwrócić jednak uwagę na fakt, że testy przeprowadzone były na zwykłym laptopie z procesorem Intel Core i3-3110M, a liczba zapytań na sekundę możliwych do przetworzenia sięgała kilku tysięcy. Na wyspecjalizowanym sprzęcie prawdopodobnie bez problemu można uzyskać wartości rzędu kilkudziesięciu lub nawet kilkuset tysięcy zapytań na sekundę. Wartości takie prawdopodobnie nigdy nie będą konieczne do uzyskania w aplikacjach biznesowych, a nawet jeśli to zdecydowanie wcześniej natkniemy się inne ograniczenia wydajnościowe – np. związane z zapytaniami do bazy danych.

Wybór modelu hostingowego w większości przypadków należy zatem oprzeć o wymagane możliwości konfiguracyjne oraz łatwość zarządzania aplikacją. W przypadku aplikacji w sieci wewnętrznej najlepszym rozwiązaniem będzie użycie samego serwera wewnętrznego Kestrel z uwagi na jego dużą wydajność oraz łatwość uruchomienia. Dla aplikacji w sieci publicznej, które mogą wymagać bardziej skomplikowanej konfiguracji należy użyć konfiguracji Kestrel + IIS. Na ten moment najbezpieczniejszy jest model standardowy, czyli oddzielny proces IIS i oddzielny proces aplikacji z Kestrelem, jednak prace nad nowym modelem (Kestrel z aplikacją działający wewnątrz procesu IIS) ciągle trwają i niedługo powinien być on gotowy do użycia produkcyjnego [8]. Wtedy model ten będzie najodpowiedniejszym z uwagi na połączenie zalet wynikających z użycia samego Kestrela (duża wydajność) i serwera pośredniczącego IIS (duże możliwości konfiguracyjne i bezpieczeństwo).

Literatura

[1] A. Freeman, Pro ASP.NET Core MVC, Apress, 2016

[2] https://docs.microsoft.com/en-us/aspnet/core/fundamentals/servers/?view=aspnetcore-2.0&tabs=aspnetcore2x [20.05.2018]

[3] https://odetocode.com/blogs/scott/archive/2016/10/25/asp-net-core-and-the-enterprise-part-2-hosting.aspx [12.06.2018]

[4] https://docs.k6.io/docs [20.05.2018]

[5] A. Lock, ASP.NET Core in Action, Manning, 2017

[6] https://docs.microsoft.com/en-us/aspnet/core/fundamentals/servers/aspnet-core-module?view=aspnetcore-2.0 [20.05.2018]

[7] https://blogs.msdn.microsoft.com/webdev/2018/02/28/asp-net-core-2-1-0-preview1-improvements-to-iis-hosting/ [20.05.2018]

[8] https://github.com/aspnet/IISIntegration/issues/878 [12.06.2018]

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 8 (2018) 263-269
Wysłane: 2018-06-17
Przyjęte: 2018-06-22

Analiza wydajności relacyjnych baz danych Oracle oraz MSSQL na podstawie aplikacji desktopowej

Grzegorz Dziewit*, Jakub Korczyński*, Maria Skublewska-Paszkowska

[a] Politechnika Lubelska, Instytut Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. Porównywanie wydajności relacyjnych baz danych nie jest trywialnym zjawiskiem ze względu na różnice implementacji różnych systemów bazodanowych. Niniejszy artykuł przedstawia metodykę sposobu porównania relacyjnych systemów bazodanowych pod względem średniego czasu wykonania poszczególnych zapytań bazodanowych typu DML (ang. Data Manipulation Language) zawierających podzapytania oraz złączenia tabel. Metodyka może być dodatkowo dostosowana do badań wydajnościowych w zakresie samej bazy danych (badanie zapytań wywoływanych bezpośrednio w silniku bazodanowym). Zastosowana metodyka pozwala na stwierdzenie, który system bazodanowy jest lepszy w porównaniu do innych w zależności od funkcjonalności spełnianych przez aplikację zewnętrzną. W artykule przeprowadzono analizę średnich czasów wykonania poszczególnych zapytań bazodanowych typu DML. Zostały postawione dwie hipotezy badawcze: „System bazodanowy Microsoft SQL Server charakteryzuje się krótszym czasem wykonania zapytań typu INSERT oraz UPDATE, w porównaniu z bazą danych Oracle" oraz „Baza danych Oracle cechuje się szybszym wykonywaniem zapytań typu DML na danych binarnych, w porównaniu z Microsoft SQL Server'em".

Słowa kluczowe: wydajność baz relacyjnych; operacje DML; aplikacja desktopowa; Oracle; MSSQL

*Autor do korespondencji.

Adresy e-mail: grzegorz.dziewit@pollub.edu.pl, jakub.korczynski@pollub.edu.pl

Performance analysis of relational databases Oracle and MS SQL based on desktop application

Grzegorz Dziewit*, Jakub Korczyński*, Maria Skublewska-Paszkowska

[a] Institute of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract. Comparison of efficiency is not a trivial phenomenon because of disparities between different database systems. This paper presents a methodology of comparing relational database systems in respect of mean time of execution individual DML queries containing subqueries and conjunction of tables. The presented methodology can be additionally accommodated to studies of efficiency in a range of database system itself (study of queries executed directly in database engine). The described methodology allows to receive statement telling which database system is better in comparison to another in dependency of functionalities fulfilled by external application. In the article the analysis of mean time of execution individual DML queries was performed.Two research hypotheses have been put forward: "Microsoft SQL Server database system needs less time to execute INSERT and UPDATE queries than Oracle database" and "Oracle database system needs less time to execute DML queries with binary data than SQL Server"

Keywords: relational databases efficiency; DML operations; desktop application; Oracle; MSSQL

*Corresponding author.

E-mail addresses: grzegorz.dziewit@pollub.edu.pl, jakub.korczynski@pollub.edu.pl

1. Wstęp

W dzisiejszych czasach bazy danych są często stosowane w rozwiązaniach biznesowych. Na przestrzeni ostatnich lat powstało bardzo wiele różnorodnych systemów bazodanowych nierelacyjnych i relacyjnych (Oracle, MS SQL, My SQL, etc.). Ten ostatni typ systemów jest bardzo popularny pod względem użycia, jednakże w związku z narastającą liczbą zróżnicowanych bazodanowych systemów relacyjnych należy wprowadzić kryterium porównawcze, które pozwoli ocenić wydajność tych baz danych. Zanim zostanie wprowadzone to kryterium należy wspomnieć o tym, że niemalże każda aplikacja realizuje podobne funkcjonalności gromadzenia danych. Wprowadzone kryterium musi umożliwiać otrzymywanie wyników, które będą stanowić możliwość porównania wspomnianych relacyjnych baz danych, których działania koordynowane są z działaniem zewnętrznych aplikacji w tym przypadku desktopowych. Celem artykułu jest przeprowadzenie analizy średnich czasów wykonania poszczególnych zapytań bazodanowych typu DML. To kryterium zostało dobrane, ponieważ operacje DML są prawdopodobnie najczęstszym typem wykonywanych działań na systemie bazodanowym. Na potrzeby pracy badawczej została opracowana aplikacja desktopowa wspomagająca zarządzanie warsztatem samochodowym zawierająca moduł testowy odnoszący się do przeprowadzanych badań.

Mając na uwadze badania literaturowe przeprowadzone w celu pogłębienia wiedzy z zakresu wydajności systemów bazodanowych (rozdział 2), sformułowane zostały następujące hipotezy:

- System bazodanowy Microsoft SQL Server charakteryzuje się krótszym czasem wykonania zapytań typu INSERT oraz UPDATE, w porównaniu z bazą danych Oracle

- Baza danych Oracle cechuje się szybszym wykonywaniem zapytań typu DML na danych binarnych, w porównaniu z Microsoft SQL Server'em.

Autentyczność przedstawionych hipotez zostanie zweryfikowana w dalszej części artykułu na podstawie przeprowadzonych badań.

2. Przegląd literaturowy

Przegląd literaturowy został podzielony ze względu na kryteria, które zostały wyszczególnione w sposób następujący:

- rodzaj zapytań wykorzystanych w badaniach,
- wykorzystane narzędzia pomocnicze,
- sposób podziału danych ze względu na ich typ oraz rozmiar,
- sposób przechowywania danych (internal storage, external storage),
- sprzęt wykorzystany do badań.

Pierwszym z kryteriów był rodzaj zapytań wykorzystywanych w badaniach wydajnościowych. Pierwszą pozycją niosącą przekaz merytoryczny z zakresu badania wydajności relacyjnych baz danych jest praca P.Singh'a i innych[1]. W danej pracy zostały wykorzystane operacje DML tj. SELECT, UPDATE, DELETE. Dodatkowo zostały użyte podzapytania oraz funkcje agregujące. Inną pracą o podobnym kontekście jest artykuł M. Shapiro i innych [2], których metodyka badań również zawierała użycie operacji DML z tym że nie obejmowała ona zapytań DELETE. Zamiast tej operacji obiektem badań zostało zapytanie typu INSERT. Ostatnią z wymienionych podobnych prac jest pozycja autorstwa I. Khawara oraz innych [3]. Metodyka badań jest porównywalna z pracą P.Singh'a [1], ponieważ skupiona jest ona na zapytaniach bazodanowych tego samego rodzaju.

W niniejszym artykule zostaną poddane testom operacje DML tj. SELECT, INSERT, UPDATE ze względu na najczęstsze ich występowanie podczas przeprowadzania operacji zarządzania relacyjnym systemem bazodanowym. Przeanalizowanie wyników badań odnośnie tych operacji pozwoli na porównanie badanych systemów bazodanowych w kontekście czasu wykonania zapytań poprzez zewnętrzną aplikację desktopową.

Kolejnym kryterium sklasyfikowania pozycji przeglądu były wykorzystane narzędzia pomocnicze. To kryterium jest istotną perspektywą, ponieważ w żadnej z wymienionych prac nie zostały podane szczegółowe informacje odnośnie sposobu pozyskiwania danych wydajnościowych. Wyjątkiem jest jedynie praca Singh'a [1], w której wykorzystane zostało narzędzie systemu Windows – Task Manager, który umożliwia pozyskiwanie danych odnośnie zużycia procesora, jednostek pamięci twardej jak i pamięci RAM. Najważniejszym jednak punktem dla zakresu tematycznego pracy był czas wykonania zbioru operacji przez dany proces bazodanowy. W podanym sposobie pracy Singh'a i innych [1] możliwe jest wystąpienie nieścisłości ze względu na to iż na wspomniany zbiór operacji mogły składać się dodatkowe działania procesu bazodanowego wykraczające poza czas działania należącego do egzekucji konkretnych zapytań bazodanowych.

Opierając się na tej tezie, niniejszy artykuł zawierać będzie rozwiązanie niwelujące potencjalne wady opisanego rozwiązania w postaci wykorzystania widoków bazodanowych dla obydwu baz relacyjnych Oracle oraz MS SQL. Wykorzystanie to będzie polegać na pobieraniu danych z widoków, które dostarczają danych dotyczących czasu wykonania jedynie poszczególnych zapytań bazodanowych w odseparowaniu od zewnętrznych działań, które mogłyby pojawiać się podczas uruchomienia/działania danej operacji DML.

We wszystkich pracach zawarta została informacja o rozmiarze danych w postaci liczby przechowywanych rekordów, rozmiarze poszczególnych danych – zajętości pamięci dyskowej oraz samym typie danych. Jednakże żadna z prac nie zawierała wszystkich tych informacji. Aby rzetelność badań była spełniona w jak najwyższym stopniu, należałoby dokładnie opisać, sklasyfikować zróżnicowanie danych pod względem rozmiaru oraz typu, dlatego też w pracy zostanie zawarta klasyfikacja podmiotów badań w obrębie której zostanie zaprojektowana metodyka przedmiotu badań, którym jest czas wykonania poszczególnych zapytań bazodanowych.

Podobna zależność ukazuje się z perspektywy sposobu przechowywania danych. Sposób przechowywania dotyczy wewnętrznego przechowywania danych (surowych umieszczonych wewnątrz bazy) lub zewnętrznego (w bazie przechowywane są jedynie referencje do surowych danych) co może wpływać na jej wydajność. Jedynie praca M.Shapiro [2] zawiera informacje o sposobie przechowywania danych, ponieważ kryterium jego badań były obydwa wspomniane sposoby przechowywania danych. Praca ta zawiera informacje o podziale danych wielkości tj. 0.5 MB oraz 5MB.

Niniejsza praca będzie zawierać podobną charakteryzację jednakże o większych rzędach rozmiarów danych, z powodu funkcjonalności aplikacji desktopowej współpracującej z relacyjnymi bazami danych Oracle oraz MS SQL. Wspomnianą funkcjonalnością jest przechowywanie danych binarnych video o rozmiarach większych niż 100 MB.

W odniesieniu do sposobu składowania danych niniejsza praca ograniczy się do przetestowania sposobu wewnętrznego, ponieważ przy tym podejściu łatwiej jest zbadać wydajność bazy danych dlatego, iż referencje są o wiele mniejsze pod względem rozmiaru niż surowe dane.

Najważniejszym i ostatnim kryterium są wyniki zebrane podczas przeprowadzonych badań, a raczej wnioski wypływające z tych wyników. Z uwagi na brak możliwości krótkiego stwierdzenia, która z baz danych jest najbardziej wydajna wprowadzono podział wyników uzależniony od typu operacji DML. Poniżej zamieszczone zostały informacje wynikające z poszczególnych prac.

Według pracy Singh'a [1] wnioski istotne dla tej pracy niosą za sobą informację, iż baza Oracle okazała się mniej wydajna niż MS SQL w przeprowadzonych eksperymentach badawczych. Praca M.Shapiro i innych [2], która zajmowała się porównaniem samych operacji względem jednej bazy Oracle przedstawia następujące wnioski:

- dla operacji typu SELECT bardziej wydajne okazało się podejście z przechowywaniem plików binarnych bezpośrednio w bazie danych,
- dla operacji typu INSERT/UPDATE bardziej wydajne okazało się przechowywanie plików poza bazą danych, zaś samej bazie jedynie referencje do tych plików,
- wraz ze wzrostem wielkości bazy danych oba podejścia (internal storage, external storage) wykazywały spadek wydajności,
- wraz ze wzrostem wielkości plików obie bazy potrzebowały więcej czasu na ich przetworzenie.

Praca I. Khawary [3] przedstawia szereg wyników, które zostały przedstawione w sposób dzięki któremu można łatwo sformułować wnioski. Metodyka zakładała, że miało się odbyć siedem eksperymentów:

operacja typu select (324500 rekordów), z następującymi wynikami:
- MySQL,
- MS SQL,
- Oracle,

operacja typu select z klauzulą where (wyszukiwanie za pomocą ID, który jest primary key), z następującymi wynikami:
- MySQL,
- Oracle 3,
- Oracle,

operacja typu update na jednej kolumnie w obrębie całej tabeli (324500 rekordów), z następującymi wynikami:
- MS SQL,
- Oracle,
- MySQL,

przeniesienie danych z jednej tabeli do drugiej o takiej samej strukturze, z tą różnicą, że w tabeli docelowej nie ma primary key, z następującymi wynikami:
- Oracle,
- MS SQL,
- MySQL,

operacja typu select z klauzulą order by, z następującymi wynikami:
- MySQL,
- MS SQL,
- Oracle,

operacja typu select z klauzulą group by, z następującymi wynikami:
- MySQL,
- MS SQL,
- Oracle,

operacja typu select ze złączeniem dwóch tabel klauzulą join, z następującymi wynikami:
- MySQL,
- MS SQL,
- Oracle.

Porównanie wydajności sprowadziło się wówczas do zestawienia czasów w obrębie danego zapytania i stwierdzenia, które jest szybciej wykonywane w danym systemie bazodanowym. Wspomniane zestawienie zostało zastosowane w większości prac. Ten artykuł naukowy również będzie zawierać opis z podziałem na rodzaj zapytań ale też będzie zawierać wnioski odnośnie samego kontekstu kooperacji systemu bazodanowego z utworzoną aplikacją desktopową wspomagająca zarządzanie warsztatem samochodowym.

3. Aplikacja desktopowa

Przedstawione w artykule badania są przeprowadzone przy użyciu aplikacji desktopowej. Służy ona do wspomagania zarządzania warsztatem samochodowym w postaci prowadzenia ewidencji napraw, pracowników a także poradników napraw w postaci video. Ta ostatnia cecha odróżnia tę aplikację od innych o podobnym przeznaczeniu, ponieważ ten sam proces naprawy może różnić się w zależności od modelu samochodu. Wówczas pracownik jest zwolniony od pamiętania wszystkich sposobów napraw i wystarczy aby był zalogowany w aplikacji i miał dostęp do modułu zarządzania poradnikami w celu obejrzenia danego sposobu naprawy. Aplikacja została napisana w języku programistycznym C# przy wykorzystaniu technologii MVVM (ang. Model-View-ViewModel)[4]. Wykorzystuje ona dwa silniki bazodanowe Oracle[5] oraz MS SQL[6]. Aplikacja zawiera moduł testujący, za pomocą którego realizowane są badania.

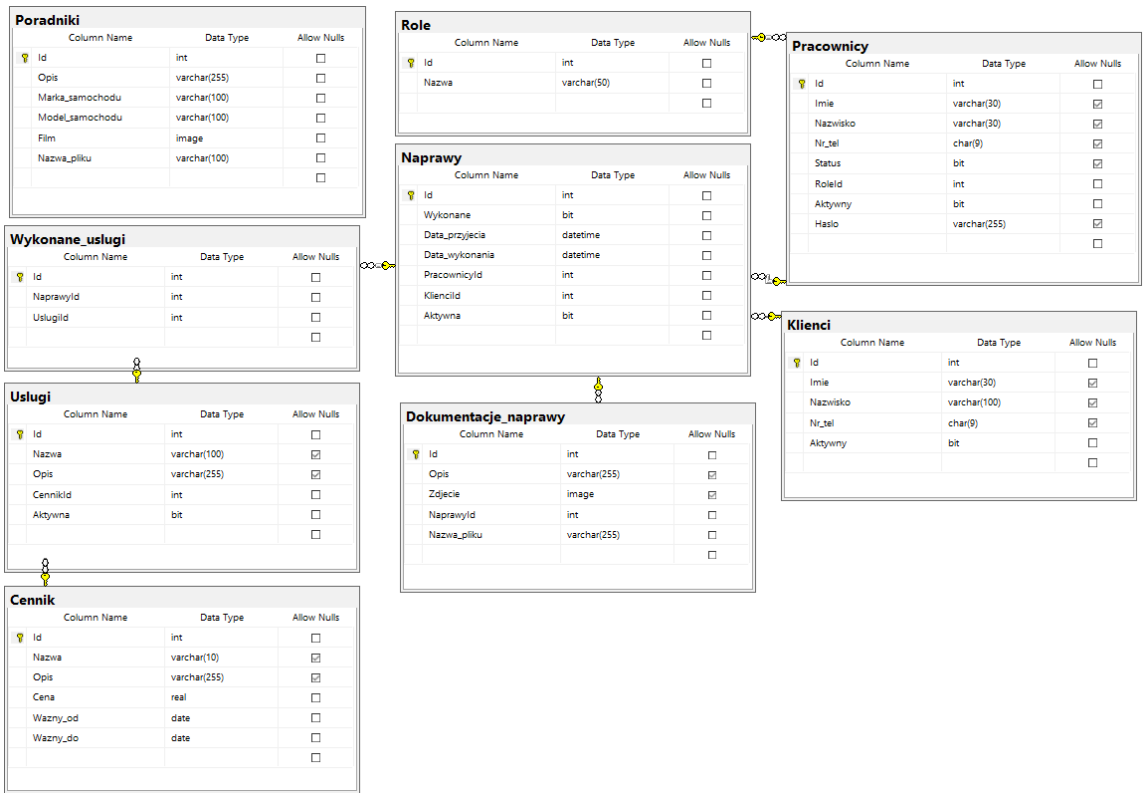
4. Systemy bazodanoweOracle

Oracle jest to system zarządzania bazą danych stworzony przez amerykańskie przedsiębiorstwo Oracle Corporation, posługujący się językiem zapytań SQL (ang. Structured Query Language). Technologia ta może być wykorzystywana zarówno lokalnie, jak również w globalnej sieci, ponadto twórcy udostępniają dedykowany komponent służący do komunikacji z bazą danych, co jest bardzo istotne dla programistów korzystających z tego rozwiązania.[5]

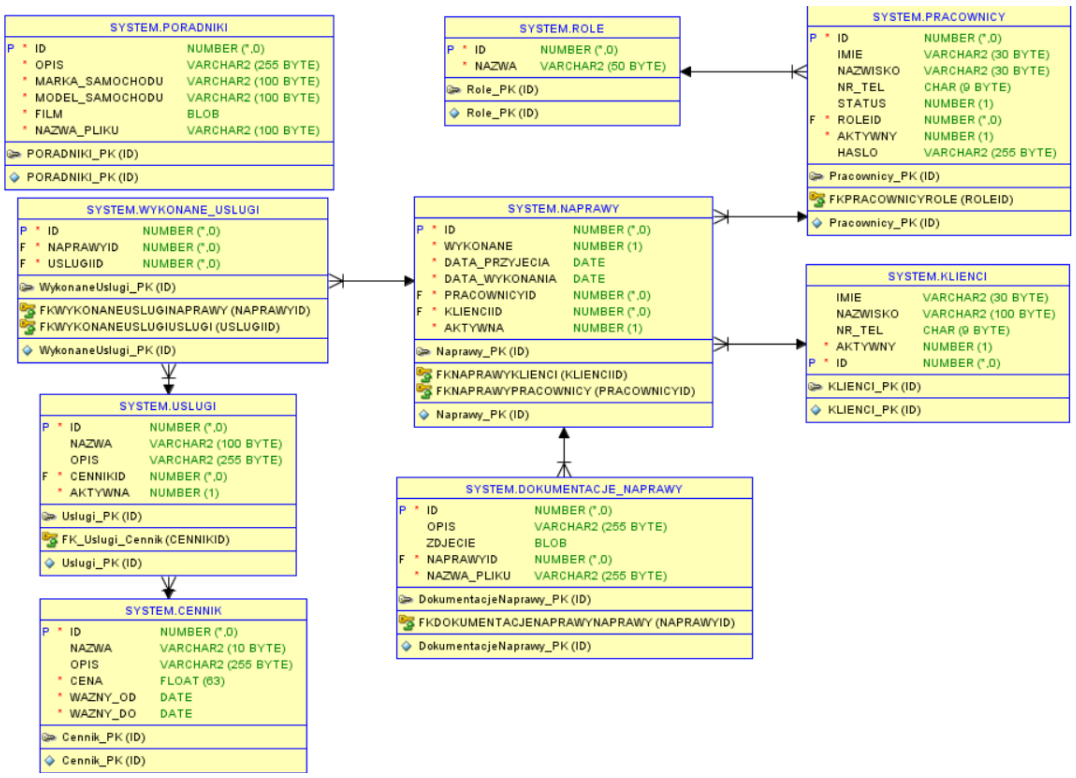
4.2 MS SQL

Microsoft SQL Server jest to system zarządzania bazą danych, którego twórcą jest dobrze znana firma Microsoft. Wykorzystuje on język zapytań T-SQL (ang. Transact-SQL), który jest implementacją języka SQL, dodającą zbiór nowych rozszerzeń. Wspomniana baza danych w wersji 2016 posiada wiele dodatkowych możliwości takich jak analiza operacji w czasie rzeczywistym lub wizualizacja danych [6].

Na rys.1 oraz rys.2 przedstawione zostały schematy baz danych wykorzystanych podczas przeprowadzania badań w formie diagramów ERD (ang. Entity Relationship Diagram) [7].



Rys. 1. Diagram ERD dla bazy danych Ms SQL



Rys. 2. Diagram ERD dla bazy danych Oracle

5. Metodyka

W niniejszym podrozdziale zostanie opisana metodyka, która dotyczy badania czasu odpowiedzi systemów bazodanowych MS SQL oraz Oracle w zależności od wykonywanych zapytań wywoływanych przez zewnętrzną (aplikacja pozasystemowa względem bazy danych) aplikację desktopową wspomagającą zarządzanie warsztatem samochodowym. Obiektem badań będzie czas realizacji zapytań bazodanowych. Zapytania budowane są przez zewnętrzną aplikację, która przekazuje treści zapytań do danego systemu bazodanowego, a następnie są one

wykonywane poprzez konkretny system. W momencie ukończenia wykonywania danego zapytania w systemie bazodanowym następuje zapisanie danych w widokach bazodanowych odnoszących się do statystyk wykonanego zapytania przekazanego przez aplikację. Wspomniane widoki to:

- **baza danych Oracle**: V$SQL – widok wyświetlający statystyki wykonanych zapytań, gdzie jeden wiersz odnosi się do pojedynczego zapytania. Dane statystyczne aktualizowane są po wykonaniu polecenia, bądź w trakcie jego trwania (co 5 sekund), jeżeli trwa on dłuższy czas. W kontekście badania czasu wykonania danego zapytania, najważniejszą kolumną wspomnianego widoku jest „ELAPSED_TIME", gdzie przechowywana jest informacja o czasie wykonania konkretnego polecenia w mikrosekundach [8],
- **baza danych MS SQL**: sys.dm_exec_query_stats – widok wyświetlający statystyki wykonanych zapytań. Jeden wiersz wynikowy odpowiada statystykom jednego wykonanego zapytania. Kolumną przechowującą dane odnośnie czasu wykonania zapytania jest „total_elapsed_time", a jednostką czas jest mikrosekunda [9].

Pobieranie danych statystycznych z widoków jest realizowane poprzez zapytania napisane w języku programistycznym SQL. Wspomniane zapytania z podziałem na bazę danych przedstawiają się następująco:

- Oracle: zapytanie pobierające statystyki wykonanych poleceń oraz czyszczące pamięć podręczną (rys. 3);
- SQL Server: zapytanie pobierające statystyki wykonanych poleceń oraz czyszczące pamięć podręczną (rys. 4).

```
-------------- zapytanie wydajnościowe
select
LAST_LOAD_TIME,
ELAPSED_TIME,
MODULE,
SQL_TEXT_elapsed,
EXECUTIONS,
from v$sql
where Module like '%Warsztat%'
order by LAST_LOAD_TIME desc;

-------- czyszczenie pamięci podręcznej
alter system flush shared_pool;
```

Rys. 3. Zapytanie pobierające statystyki wykonanych poleceń oraz czyszczące pamięć podręczną w bazie danych Oracle

Dane otrzymywane w wyniku przeprowadzanych badań to dane numeryczne odpowiadające krotności wykonania danego zapytania oraz czas odpowiedzi zapytania wyrażony w mikrosekundach. Dane wynikowe zostały podzielone ze względu na liczbę wykonań danego zapytania rzędu 500, 1000, 5000 dla danych małego rozmiaru, oraz 10, 50, 100 dla danych o większym rozmiarze. Przy każdorazowym wykonywaniu badania liczba rekrdów przechowywanych w tabelach odpowiadała liczbie rekordów, na której przeprowadzano poszczególne operacje DML. Rozmiar danych jest dodatkowym kryterium podziału danych ze względu na ich typ. Dane tekstowe odpowiadają rozmiarowi mniejszemu niż 1MB, zdjęcia w postaci danych binarnych odpowiadają rozmiarowi z zakresu 1-50 MB, natomiast największym rozmiarem danych charakteryzują się filmy również w postaci binarnej, których zajętość pamięciowa wynosi więcej niż 50 MB.

```
-------------------- zapytanie wydajnościowe
SELECT SUBSTRING(qt.TEXT, (qs.statement_start_offset/2)+1,
((CASE qs.statement_end_offset
WHEN -1 THEN DATALENGTH(qt.TEXT)
ELSE qs.statement_end_offset
END - qs.statement_start_offset)/2)+1) as command,
sum(qs.execution_count),
sum(qs.total_elapsed_time)
FROM sys.dm_exec_query_stats qs
CROSS APPLY sys.dm_exec_sql_text(qs.sql_handle) qt
CROSS APPLY sys.dm_exec_query_plan(qs.plan_handle) qp
group by SUBSTRING(qt.TEXT, (qs.statement_start_offset/2)+1,
((CASE qs.statement_end_offset
WHEN -1 THEN DATALENGTH(qt.TEXT)
ELSE qs.statement_end_offset
END - qs.statement_start_offset)/2)+1)

------------- czyszczenie pamięci podręcznej
DBCC FREEPROCCACHE
DBCC DROPCLEANBUFFERS
```

Rys. 4. Zapytanie pobierające statystyki wykonanych poleceń oraz czyszczące pamięć podręczną w bazie danych MS SQL

Poniżej przedstawiony został schemat postępowania podczas przeprowadzania badania:

1) Uruchomienie wybranego serwera bazy danych.
2) Uruchomienie aplikacji desktopowej i zalogowanie się jako konkretny użytkownik.
3) Wybranie opcji modułu testującego aplikacji.
4) Wykonanie zapytania przeznaczonego do badania wydajnościowego.
5) Uruchomienie zapytania statystycznego przy pomocy SQL Developer lub SQL Managment Studio.
6) Pobranie wyników oraz wyczyszczenie pamięci podręcznej.

Czyszczenie pamięci podręcznej wiąże się także z czyszczeniem planu zapytań w obydwu bazach danych i ta operacja wykonywana jest każdorazowo po przeprowadzonym badaniu. W bazie Oracle zapytaniem czyszczącym zarówno pamięć podręczną jak i plan zapytań było zapytanie przedstawione na Rys. 3 [10][11]. Natomiast zapytania wymienione kolejno na Rys. 4 odpowiadały za wyczyszczenie planu zapytania oraz wyczyszczenie pamięci podręcznej dla bazy MS SQL [12].Wszystkie badania wydajnościowe są przeprowadzone za pomocą zapytań DML, wyłączając z nich operacje DELETE. Dodatkowo zapytania zawierają złączenia (INNER JOIN), a także podzapytania.

6. Wyniki

Wszystkie wyniki stanowią średnią arytmetyczną czasów wykonań poszczególnych zapytań.

Tabela 1. Wyniki pomiarów dla operacji select dla danych binarnych

Liczba rekordów	MS SQL		Oracle	
	Średni czas [µs]	Std	Średni czas [µs]	Std
10	3300906	148136	5844	520
30	10487079	66586	6244	918
50	218454820	119136	5486	1242

Tabela 2. Wyniki pomiarów dla operacji update dla danych binarnych

Liczba rekordów	MS SQL		Oracle	
	Średni czas [µs]	Std	Średni czas [µs]	Std
1	5400762	2332554	8198944	1529275
3	19262326	6387159	26779215	5212174
5	20176755	1813704	61494904	15772001

Tabela 3. Wyniki pomiarów dla operacji select z pojedynczej tabeli na danych tekstowych/numerycznych

Liczba rekordów	MS SQL		Oracle	
	Średni czas [µs]	Std	Średni czas [µs]	Std
10000	16401	2441	6183	545
50000	101555	3828	15937	965
100000	199985	4749	27222	1376

Tabela 4. Wyniki pomiarów dla operacji insert na danych tekstowych/numerycznych

Liczba rekordów	MS SQL		Oracle	
	Średni czas [µs]	Std	Średni czas [µs]	Std
500	22974	511	95579	3370
1000	44404	723	189186	5002
5000	213822	4015	915339	15786

7. Analiza wyników

Celem niniejszej pracy było porównanie relacyjnych baz danych Oracle oraz SQL pod względem czasów wykonania poszczególnych zapytań bazodanowych. Badania zakończyły się sukcesem poprzez zastosowanie opisanej metodyki. Wyniki badań są dosyć zróżnicowane, baza Oracle zdecydowanie lepiej wykonuje operacje SELECT zarówno na danych binarnych (tabela 6.1) oraz tekstowych (tabela 6.3) w porównaniu do bazy MS SQL, natomiast ta druga o wiele lepiej realizuje operacje UPDATE (tabela 6.2) oraz INSERT (tabela 6.4) również w odniesieniu do obydwu typu danych. We wstępie odniesiono się do potrzeby wprowadzenia kryterium porównawczego, które pozwoli porównać relacyjne systemy bazodanowe współpracujące z zewnętrznymi w stosunku do nich aplikacjami. Kryterium to jest uzależnione od częstotliwości występowania danego zapytania bazodanowego przy wykorzystaniu zewnętrznej aplikacji. Zatem najlepszą bazą danych jest ta, która najszybciej wykonuje najczęściej wykonywane zapytania a wolniej te, które w rzadszym stopniu pojawiają się podczas zarządzania bazą danych. Mianowicie, jeśli wiadomo, iż działanie aplikacji najczęściej sprowadza się do wywoływania operacji SELECT wówczas lepszą bazą danych jest baza Oracle. W innym przypadku, gdy aplikacja zleca najczęściej wykonanie zapytań INSERT oraz UPDATE systemowi bazodanowemu, wówczas lepszą bazą danych okaże się baza danych MS SQL. Odnosząc się do głównego kontekstu czyli aplikacji wspomagającej zarządzanie warsztatem samochodowym, można stwierdzić iż lepszą bazą danych do tego typu aplikacji jest baza Oracle. Mimo iż dane statystyczne pokazują, że MS SQL jest lepszy od bazy Oracle w operacjach UPDATE (tabela 6.2) oraz INSERT (tabela 6.4), to operacje UPDATE są znacznie rzadziej wykonywane niż operacje SELECT. Dodatkowo można zauważyć, że rząd wielkości czasu wstawiania 5000 wierszy do tabeli zamyka się poniżej 1s (tabela 6.4). W związku z tym te niedoskonałości bazy Oracle są o wiele mniej znaczące w odniesieniu do różnicy w pobieraniu danych binarnych z tabeli. Ta różnica pomiędzy bazami jest ogromna, gdyż czas pobierania danych binarnych przez bazę Oracle zamyka się w czasie mniejszym niż 0.01s przy o wiele dłuższym czasie (10s) realizacji tego zapytania przez bazę MS SQL (tabela 6.1). Różnice te mogą wynikać ze sposobu przechowywania danych, ponieważ w bazie Oracle wykorzystanym typem danych binarnych był typ BLOB[13], natomiast w bazie danych MS SQL typem danych był image[14]. Ostatnim zagadnieniem, godnym uwagi jest fakt, że w badaniu średniego czasu wykonania operacji update na danych binarnych (Tabela 2) wykonywano operacje na maksymalnie 5 rekordach. Tak mała liczba wynika z czasu przetwarzania danych binarnych w tej operacji, ponieważ rozmiar tych rekordów przekraczał łącznie nawet 500 MB (duży rozmiar wiąże się z długim czasem przetwarzania) zajętości danych, które były przechowywane bezpośrednio w bazie danych. Mimo małej liczby rekordów przechowywanej w bazie danych wystarczyła ona na przeprowadzenie badań oraz analizę wyników, których dość długi czas wykonania został przedstawiony w Tabeli 2.

Podsumowując, wykonane badania wykazują, iż baza danych Oracle jest lepsza do tego typu aplikacji.

8. Podsumowanie

Celem niniejszego artykułu była analiza wyników przy użyciu zaprojektowanej metodyki, który udało się osiągnąć. Metodyka okazała się skuteczna oraz cechuje się możliwością dostosowania do innych typów badań jakimi mogą być np. badania wydajnościowe zapytań uruchamianych przez sam system bazodanowy. W związku z przeprowadzonymi badaniami postawiono hipotezy:

- System bazodanowy Microsoft SQL Server charakteryzuje się krótszym czasem wykonania zapytań typu INSERT oraz UPDATE, w porównaniu z bazą danych Oracle,
- Baza danych Oracle cechuje się szybszym wykonywaniem zapytań typu DML na danych binarnych, w porównaniu z Microsoft SQL Server.

Pierwsza z wymienionych hipotez została potwierdzona, gdyż badania wykazały, iż baza danych MS SQL lepiej realizuje operacje INSERT i UPDATE niż baza Oracle. Druga z hipotez została obalona, mimo że baza Oracle znacznie lepiej realizuje operacje SELECT dla danych binarnych to baza MS SQL szybciej wykonuje operacje UPDATE. W wyniku badań okazało się, że baza Oracle lepiej nadaje się do współpracy z aplikacjami zewnętrznymi.

Podsumowując całość artykułu, może on służyć jako źródło informacji odnośnie projektowanej metodyki w celu analizy wydajności innych relacyjnych systemów bazodanowych.

Literatura

[1] Singh P., Sharma S., Kaur S.: Performance Analysis of Different DBMS Systems. International Journal of Advance Foundation And Research In Science & Engineering, 2015,1. 1-9.

[2] Shapiro M., Miller E.: Managing databases with binary large objects. 16th IEEE Symposium on Mass Storage Systems in cooperation with the 7th NASA Goddard Conference on Mass Storage Systems and Technologies, San Diego 1999

[3] Khawar I., Kamran A., Syed B., Syed A.: Huge and Real-Time Database Systems: A Comparative Study and Review for SQL Server 2016, Oracle 12c & MySQL 5.7 for Personal Computer. Journal of Basic and Applied Sciences, 2017, 13.

[4] Mihailescu M., Sorensen E.: Model-View-ViewModel (MVVM) Design Pattern using Windows Presentation Foundation (WPF) Technology. Megabyte Journal, 2010

[5] Castel J.: Oracle 12c: SQL. Cengage Learning, 2015

[6] Sanka, D., Durkin, W., Radivojevic, M.: SQL Server 2016 Developer's Guide. Packt Publishing, 2017

[7] Specyfikacja diagramu ERD, https://msdn.microsoft.com/pl-pl/library/projektowanie-baz-danych--diagramy-erd-relacje-miedzy-tabelami-zwiazki-rekordy.aspx, [2018-06-20]

[8] Opis widoku V$SQL (Oracle), https://docs.oracle.com/cd/B19306_01/server.102/b14237/dynviews_2113.htm#REFRN30246, [2018-05-15]

[9] Opis widoku sys.dm_exec_query_stats (Ms SQL Server), https://docs.microsoft.com/en-us/sql/relational-databases/system-dynamic-management-views/sys-dm-exec-query-stats-transact-sql?view=sql-server-2017, [2018-05-15]

[10] Opis shared_pool dla bazy Oracle, http://www.dba-oracle.com/concepts/shared_pool_concepts.htm, [2018-06-20]

[11] Opis zapytania czyszczącego pamięć podręczną oraz plan zapytań dla bazy Oracle. http://www.dba-oracle.com/tp_Oracle_shared_pool_routine_flush.htm, [2018-06-20]

[12] Opis zapytań czyszczących pamięć oraz plan zapytań dla bazy MS SQL, http://www.sqlpedia.pl/aspekty-wydajnosciowe-zapytan-sql/, [2018-06-20]

[13] Opis bazodanowego typu BLOB w systemie Oracle, https://docs.oracle.com/javadb/10.8.3.0/ref/rrefblob.html, [2018-05-15]

[14] Opis bazodanowego typu Image w systemie SQL Server, https://docs.microsoft.com/en-us/sql/t-sql/data-types/ntext-text-and-image-transact-sql?view=sql-server-2017& viewFallbackFrom=sql-server-2017, [2018-05-15]

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 8 (2018) 270-276
Wysłane: 2018-06-23
Przyjęte: 2018-06-27

Metody weryfikujące poziom wiedzy i umiejętności programisty

Paweł Hajduk*, Norbert Wieruszewski*, Maria Skublewska-Paszkowska
Politechnika Lubelska, Instytut Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. Artykuł opisuje aktualnie stosowane metody weryfikacji poziomu wiedzy i umiejętności programistów. Do realizacji badań wykorzystano własne rozwiązanie w postaci aplikacji implementującej kilka wybranych metod, na której następnie przeprowadzono testy użytkowe przy udziale programistów o zróżnicowanym poziomie doświadczenia, wiedzy i umiejętności. Na podstawie analizy uzyskanych wyników wyciągnięto wnioski, które pozwoliły na ocenienie każdej z metod w następujących kategoriach: skuteczność sprawdzenia użytkownika, niezawodność działania metody, czas weryfikacji rozwiązania, atrakcyjność użytkowania oraz uniwersalność metody.

Słowa kluczowe: automatyczna ocena programistów; metody weryfikacji wiedzy; testy jednostkowe; analiza statyczna kodu

*Autor do korespondencji.

Adresy e-mail: pawel.hajduk@pollub.edu.pl, norbert.wieruszewski@pollub.edu.pl

Verification methods of a programmer's knowledge and skills

Paweł Hajduk*, Norbert Wieruszewski*, Maria Skublewska-Paszkowska
Institute of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract. The article describes currently utilized methods of a programmer's knowledge verification and skills. The research consisted of creating custom solution which was an application implementing chosen methods and carrying out test with the participation of programmers having various levels of experience, knowledge and skills. Effectiveness of assessment, reliability and verification time were evaluated based on an analysis of the results received from the research.

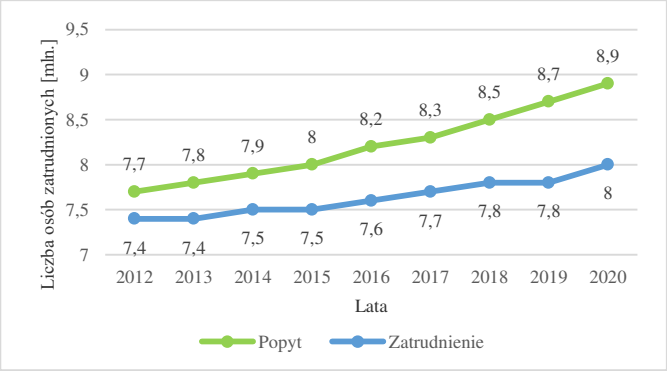
Keywords: automatic programmers assessment; knowledge verification methods; unit tests; static code analysis

*Corresponding author.

E-mail addresses: pawel.hajduk@pollub.edu.pl, norbert.wieruszewski@pollub.edu.pl

1. Wstęp

XXI wiek to dalszy ciąg okresu dynamicznego rozwoju branży informatycznej. Wynikiem tego jest fakt, że ogólnoświatowe zapotrzebowanie na specjalistów IT nie przestaje rosnąć. Jest to przede wszystkim efekt działań dotyczących pełnej informatyzacji kolejnych przedsiębiorstw w różnych gałęziach gospodarki. Rys. 1 przedstawia wykres wzrostu zatrudnienia w branży IT, z którego łatwo jest odczytać, że popyt znacznie przewyższa zatrudnienie.



Rys. 1. Wzrost zatrudnienia w branży ICT w krajach Unii Europejskiej (w mln) [1]

Firmy z branży IT stają przed problemem polegającym na zaspokojeniu wysokiego popytu na wykwalifikowanych i doświadczonych informatyków. Na rynku nie brakuje chętnych do pracy kandydatów, jednak znalezienie osoby o wymaganych kwalifikacjach, w założonym wcześniej czasie, jest bardzo trudne. Z uwagi na ogromną liczbę kandydatów, których tylko niewielki procent spełnia wymagania pracodawcy na dane stanowisko, a także na czas jaki musi poświęcić pojedyncza osoba techniczna na zweryfikowanie wiedzy i sprawdzenie testu praktycznego każdego z kandydatów, proces rekrutacji programistów wydaje się być dobrym przykładem procesu, który należałoby usprawnić i w jak najwyższym stopniu zautomatyzować [2].

Celem badania było przeprowadzenie analizy metod oceniających poziom wiedzy programistycznej, wykrycie ich wad oraz zaproponowanie alternatywnych rozwiązań. Na podstawie uzyskanych wyników metody zostały przeanalizowane w celu odnalezienia ich słabych punktów, które powinny zostać w przyszłości wyeliminowane.

Celem pośrednim była implementacja aplikacji pozwalającej na przeprowadzenie testów użytkowych z wykorzystaniem wybranych metod weryfikacji poziomu wiedzy programistycznej.

2. Przegląd literatury

W celu wprowadzenia automatyzacji do analizowanego procesu należy najpierw wydzielić z niego te fragmenty, które automatyzacji wymagają, a zarazem, dla których jest ona w ogóle możliwa. Do takich fragmentów z pewnością można

zaliczyć etap weryfikacji poprawności i jakości kodu programu. Poprawność składni, czyli zgodność kodu z gramatyką i zasadami danego języka, jest poddawana analizie między innymi na etapie kompilacji przeprowadzanej przez kompilatory, które posiadają algorytmy zaimplementowane specjalnie w tym właśnie celu. Poprawność logiczna natomiast, która ma największe znaczenie w kontekście oceny umiejętności badanej osoby, określa to, czy dany program działa prawidłowo z różnymi zadanymi wartościami wejściowymi. Tego typu weryfikacja może zostać przeprowadzona przy użyciu mechanizmu testów jednostkowych.

Prawidłowy kod programu nie zawsze charakteryzuje się wysoką jakością, co oznacza, że jej poziom powinien zostać poddany oddzielnej analizie. Przykładem takiego sprawdzenia jest analiza statyczna kodu badająca kod źródłowy programu bez jego uruchamiania w wyniku czego możliwe jest wyznaczenie parametrów takich jak złożoność obliczeniowa.

2.1. Kompilacja dowodem poprawności składni kodu programu

W poszukiwaniu rozwiązania mającego posłużyć za swego rodzaju weryfikator poprawności składni kodu rozważyć można wykorzystanie gotowych i w założeniu niezawodnych kompilatorów. Głównym zadaniem tych aplikacji jest przepisanie kodu programu wejściowego, napisanego najczęściej w języku wysokiego poziomu, na kod programu wyjściowego innego języka [3]. W czasie skomplikowanego procesu kompilacji przeprowadzane zostają między innymi trzy analizy: leksykalna, składniowa i semantyczna [4].

Analiza leksykalna dzieli kod wejściowy na pojedyncze jednostki leksykalne opisane w gramatyce danego języka programowania - tak zwane leksemy. Jeżeli kompilator znajdzie wyrażenie, które nie zostało opisane w gramatyce, to zostanie ono zgłoszone jako błąd leksykalny.

W fazie analizy składniowej sprawdzeniu podlegają wydzielone wcześniej tokeny, a dokładniej ich wzajemne ustawienia. Dobrym przykładem mogą być nawiasy klamrowe, które otaczają bloki kodu. Każdy otwarty nawias klamrowy powinien również zostać zamknięty. W przeciwnym wypadku taka sytuacja zostanie zgłoszona jako błąd składniowy [5].

Jako ostatnia z analiz w procesie kompilacji zostaje przeprowadzona analiza semantyczna. Jej zadaniem jest sprawdzenie kodu programu w celu wykrycia błędów wynikających z nieprawidłowego użycia elementów języka, które same w sobie można by uznać za poprawne.

2.2. Poprawność logiczna programu potwierdzona testami jednostkowymi

Testy jednostkowe to sposób na sprawdzanie działania najmniejszych części programu, które można logicznie odseparować od reszty. Przyjmuje się, że testowany fragment programu wykonuje pojedynczą jednostkę pracy, niekoniecznie jest to odrębna część kodu np. metoda lub klasa [6]. W procesie wytwarzania oprogramowania testy jednostkowe są jednym z ważniejszych wyznaczników poprawności działania systemów informatycznych, a dzięki popularności tej metody, istnieje wiele narzędzi pozwalających pisać i wykonywać testy jednostkowe. Odpowiednio dobrany zestaw testów jednostkowych może wykryć wiele przypadków, gdzie wykonywany kod nie realizuje swojego zadania [7].

2.3. Analiza statyczna kodu

Sprawdzenie poziomu skomplikowania programu to bardzo rozległe zagadnienie, ale na potrzeby stosowanych metod, wystarczającym okazało się zawężenie pojęcia złożoności kodu wyłącznie do następujących aspektów: łatwości zrozumienia kodu, testowalności kodu, łatwości utrzymania kodu [8]. Istnieje wiele różnych metod, którymi można wyznaczyć poziom złożoności kodu, ale na potrzeby niniejszego artykułu postanowiono opisać jedynie dwie z nich: obliczanie złożoności cyklicznej oraz złożoności poznawczej.

2.3.1. Złożoność cykliczna

Złożoność programu można ocenić na kilka sposobów. Jednym z nich jest obliczenie złożoności cyklicznej, która zlicza liczbę możliwych rozgałęzień scenariusza odtworzonego w programie przy pomocy grafu przepływu sterowania, czyli liczbę przypadków testowych koniecznych do wykonania w celu pełnego pokrycia funkcjonalności kodu. Niestety metoda ta posiada pewne niedoskonałości, np. jest nastawiona na sprawdzanie poziomu skomplikowania programu z punktu widzenia wykonującej go maszyny. Z tego względu można zauważyć pewne przypadki, gdy złożoność cykliczna nie jest wyznacznikiem łatwości zrozumienia, testowania i utrzymywania kodu [8].

2.3.2. Złożoność poznawcza

Kolejną metodą obliczania złożoności programu jest złożoność poznawcza [8]. Opiera się ona na ocenie, jak trudny jest do zrozumienia przepływ programu zgodnie z poniższymi regułami [8]:

- jeżeli występuje przerwa w ciągłości przepływu programu, zwiększ wskaźnik złożoności;
- jeżeli struktury powodujące zmianę przepływu programu są zagnieżdżone, zwiększ wskaźnik złożoności;
- jeżeli wiele linii kodu w czytelny sposób zostało zastąpionych pojedynczą linią kodu, nie zwiększaj wskaźnika złożoności.

3. Aktualnie stosowane metody weryfikacji wiedzy i umiejętności programisty

3.1. Weryfikacja poziomu wiedzy przy użyciu zadań otwartych

Zadania otwarte to typ zadań, który wymaga od osoby badanej udzielenia swobodnej i autorskiej odpowiedzi, zazwyczaj nieco dłuższej niż jedno zdanie. Pytania typu

otwartego wymagają najczęściej wyjaśnienia znaczenia danego zagadnienia, opisania działania metody/procesu lub przedstawienia jakiegoś konkretnego zjawiska. Sprawdzenie tego typu zadania bez udziału osoby sprawdzającej może okazać się trudne do osiągnięcia - niemożliwym jest przygotowanie pełnego klucza odpowiedzi.

3.2. Weryfikacja poziomu wiedzy przy użyciu zadań zamkniętych

Zadania zamknięte lub inaczej zadania wyboru wymagają udzielenia odpowiedzi poprzez wytypowanie jednej lub więcej z kilku możliwości przygotowanych wcześniej przez autora zadania. Dla pytań tego typu przygotowanie mechanizmu automatycznego sprawdzania nie stanowi większego problemu - oznaczenie prawidłowych odpowiedzi w definicji zadania i przyjęcie jednej z kilku możliwych zasad punktowania zadania wystarczą do zweryfikowania i ocenienia otrzymanych odpowiedzi.

3.3. Weryfikacja poziomu umiejętności przy użyciu zadań programistycznych

Powyższa kategoria zadań opiera się na modyfikacji kodu programu w celu osiągnięcia wskazanych w treści polecenia założeń. Wśród zadań programistycznych zostały wyróżnione następujące typy.

3.3.1. Zadania algorytmiczne

Zadania algorytmiczne polegają na zaimplementowaniu uniwersalnego i optymalnego rozwiązania problemu podanego w treści zadania. Oceniana jest poprawność działania algorytmu, czas wykonania programu, ale również czytelność i łatwość zrozumienia kodu. Rozwiązanie powinno zaliczyć wszystkie zdefiniowane przypadki testowe, składające się ze zwyczajnych, ale również wyjątkowych scenariuszy, pozwalających wykryć kompletność rozwiązania oraz zabezpieczenie go przed nietypowymi danymi wejściowymi [9].

3.3.2. Poprawienie błędów w kodzie

Umiejętność wprowadzania poprawek do istniejącego kodu jest również bardzo pożądana u dobrego programisty. Wspomniana czynność często stanowi większość czasu poświęconego na rozwijanie nowych funkcjonalności w oprogramowaniu. Zadanie z naprawianiem błędów polega na przeczytaniu i zrozumieniu kodu, zidentyfikowaniu jakie błędy w nim występują, a następnie wyeliminowaniu odnalezionych defektów. Poprawność działania programu jest weryfikowana uruchomieniem zestawu testów jednostkowych i analizą statyczną kodu.

3.3.3. Poprawne wykorzystanie istniejącego kodu

Obecnie oprogramowanie wytwarzane przez firmy informatyczne jest na tyle złożone, że prace nad nim prowadzone są przez całe zespoły programistów. Zdolność wykorzystania kodu napisanego przez innych jest cenna, gdyż pozwala zminimalizować czas poświęcony na rozwijanie i testowanie kodu. Poziom opanowania wspomnianej umiejętności jest weryfikowany przez przygotowanie metod, które następnie mają być wykorzystane w celu napisania programu realizującego podane w poleceniu cele. Poprawność działania programu jest sprawdzana testami jednostkowymi.

3.3.4. Przegląd kodu

Podobnie, jak w przypadku poprzednio opisanej metody, wykonywanie przeglądu kodu (ang. code review) jest następstwem konieczności współpracy przy wytwarzaniu oprogramowania. Wzajemne przeglądanie kodu jest dobrą praktyką pozwalającą na wczesne wykrycie wielu błędów zarówno projektowych lub architektonicznych, jak i logicznych. Sprawdzenie zdolności przeprowadzania przeglądu kodu będzie polegało na dostarczeniu działającego programu, ale w jego kodzie celowo zostaną pozostawione błędy, które będzie musiał wskazać uczestnik testu. Zadanie nie będzie obejmowało poprawy wskazanych błędów.

3.3.5. Refaktoryzacja kodu

Zakończony proces przeglądu kodu często prowadzi do potrzeby wprowadzenia poprawek. Zwykle zadania realizowane przez program nie ulegają zmianie, lecz sposób implementacji programu wymaga korekty. Na wspomnianym aspekcie wytwarzania oprogramowania koncentruje się refaktoryzacja kodu. Umiejętność przeprowadzania refaktoryzacji jest badana poprzez przedstawienie uczestnikowi testu kodu z celowo popełnionymi błędami, które będą wyraźnie zaznaczone, a zadaniem programisty będzie poprawienie ich. Końcowy sposób działania programu zostanie sprawdzony testami jednostkowymi [10].

3.3.6. Optymalizacja kodu

W zależności od zastosowania pisanych programów wydajność może być kluczowa lub mieć marginalne znaczenie, ale dobrą praktyką jest zwracanie uwagi na ograniczanie czasu wykonywania się kodu. Kandydat dostanie kod programu, który będzie napisany z pominięciem kwestii optymalizacji więc będzie wykonywał niepotrzebne lub nadmiernie złożone polecenia. Zadaniem uczestnika testu będzie wyeliminowanie defektów programu. Pod koniec nastąpi weryfikacja programu i zostanie zmierzony średni czas jego wykonania.

4. Aplikacja pomocnicza

Na potrzeby badania zaimplementowano aplikację umożliwiającą testowanie wybranych metod weryfikacji wiedzy i umiejętności programistów. Stworzona aplikacja składa się z dwóch głównych modułów:

Aplikacji Klienckiej służącej do tworzenia zadań (otwartych, zamkniętych i programistycznych), przeprowadzania testów i sprawdzania ich wyników;

Aplikacji Serwerowej odpowiedzialnej za kompilację zadań programistycznych, uruchamianie testów jednostkowych i przeprowadzanie analizy statycznej kodu.

5. Analiza wybranych metod

Metody opisane w poprzednich rozdziałach zostały poddane analizie. Do udziału w badaniu zostało wytypowanych dwanaście osób pracujących zawodowo, posiadających różne doświadczenie w zakresie programowania i zostały one podzielone na trzy równe pod względem liczebności grupy, a skład każdej z nich tworzyły: osoba po szkoleniu, osoba z rocznym stażem pracy, osoba z dwuletnim stażem pracy oraz osoba z ponad trzyletnim stażem pracy.

Przeanalizowane zostały metody polegające na rozwiązywaniu testu na kartce oraz za pomocą stworzonej aplikacji. Dodatkowo przebieg egzaminu przeprowadzanego na komputerze w jednym badaniu był nadzorowany (uczestnicy nie mogli korzystać z materiałów pomocniczych), a w drugim natomiast był przeprowadzany zdalnie, przy wykorzystaniu dowolnych pomocy, z wykluczeniem komunikacji pomiędzy uczestnikami.

Każdy test składał się z tych samych sześciu zadań: dwóch otwartych, dwóch zamkniętych i dwóch programistycznych. Na udzielenie odpowiedzi uczestnicy mieli godzinę. Pytania dotyczyły znajomości platformy Salesforce oraz podstaw algorytmiki.

5.1. Test pisemny w kontrolowanych warunkach

5.1.1. Założenia

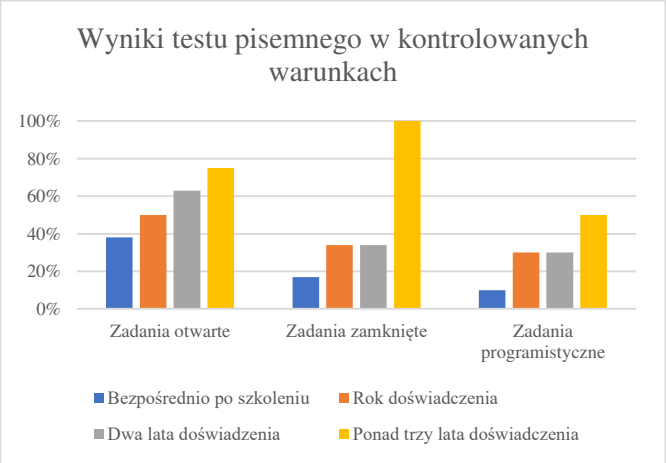
Podczas badań został zmierzony czas poświęcony na rozwiązanie poszczególnych zadań, czas spędzony na ich sprawdzenie oraz zweryfikowano, w jakim stopniu spełniają kryteria poprawności wymienione we wcześniejszej części rozdziału. W celu zredukowania wpływu obserwacji poczynań uczestników testu przez autorów pracy, zdecydowano się powierzyć obowiązek mierzenia czasu odpowiedzi na pytania samym programistom. Wiarygodność otrzymanych wyników nie była podważana przez wzgląd na profesjonalne podejście badanych osób, ale nie można wykluczyć, iż któraś z nich naruszyła ustalone zasady. Całość testu rozwiązywanego przez pojedynczego uczestnika nie powinna trwać dłużej niż godzinę.

5.1.2. Wyniki

Na Rys. 2 przedstawiono wyniki uzyskane z przeprowadzonego testu pisemnego w warunkach kontrolowanych.

5.1.3. Wnioski

Przeprowadzenie eksperymentu pozwoliło wyciągnąć liczne wnioski dotyczące skuteczności i użyteczności przeprowadzania testów pisemnych w celu weryfikacji wiedzy programistów.



Rys. 2. Wyniki testu pisemnego w kontrolowanych warunkach

Można stwierdzić, że poziom efektywności przeprowadzania testów na kartce papieru jest zadowalający dla zadań otwartych i zamkniętych. W przypadku bardziej złożonych zagadnień np. polegających na rozwiązywaniu problemów programistycznych ręczne rozwiązywanie zadań może okazać się czasochłonne i niemiarodajne. Wiele trywialnych problemów niejednokrotnie sprawiało, że programiści zamiast skupić się na rozwiązaniu problemu, zmagali się z kwestią techniczną proponowanej odpowiedzi. Takich sytuacji można było łatwo uniknąć wykorzystując powszechnie dostępne narzędzia.

Kolejnym wnioskiem, jaki może zostać wyciągnięty na podstawie wyników badania, jest potrzeba usprawnienia procesu sprawdzania zadań zawierających kod programów lub jego fragmenty zapisane odręcznie. W wymienionym przypadku utrudniona jest analiza logiki programu, ograniczona jest czytelność jego kodu, a przeprowadzanie testów dla różnych wartości wejściowych jest czasochłonne i narażone na błędy lub zaniedbanie ze względu na poziom skupienia, którego wymaga. Kwestie związane z badaniem wydajności programu dla przypadków z wieloma wartościami wejściowymi lub skomplikowanymi obliczeniami również jest niezwykle trudna do wykonania i czasochłonna.

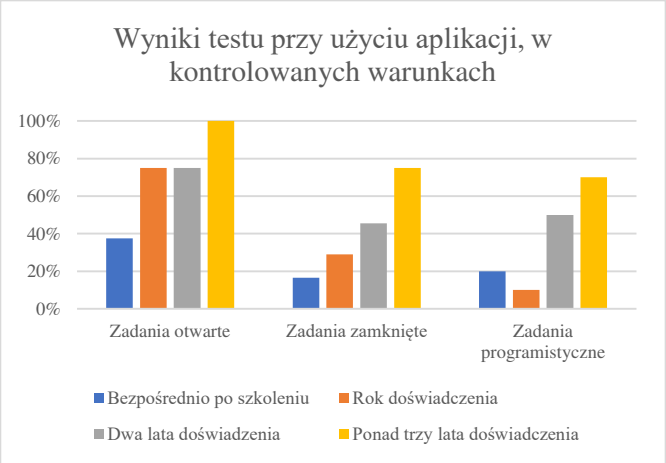
5.2. Test przeprowadzony przy użyciu stworzonej aplikacji w kontrolowanych warunkach

5.2.1. Założenia

Podczas trwania tego badania osoby rozwiązywały przygotowane zadania z wykorzystaniem aplikacji testowej. Każdy z ochotników był nadzorowany i nie miał dostępu do materiałów pomocniczych.

5.2.2. Wyniki

Na Rys. 3 przedstawiono wyniki uzyskane z przeprowadzonego testu przy użyciu stworzonej aplikacji w warunkach kontrolowanych.



Rys. 3. Wyniki testu przy użyciu aplikacji, w kontrolowanych warunkach

5.2.3. Wnioski

Na podstawie otrzymanych wyników można wysnuć kilka wniosków, które pozwalają stwierdzić, że zastosowana metoda okazała się być częściowo skuteczna. Jednym z nich jest fakt, iż znacznie lepiej widoczna jest zależność pomiędzy doświadczeniem zawodowym, a wynikami otrzymanymi w testach, co potwierdza dokładniejszą miarodajność tej metody.

Za kolejny czynnik wpływający na wiarygodność metody można uznać brak możliwości korzystania z materiałów pomocniczych, która pozwala wnioskować, że programiści posiadali niezbędną wiedzę do rozwiązania zadań. Choć brak dostępu do wiedzy na temat dziedziny rozwiązywanego problemu nie jest naturalną sytuacją w codziennej pracy programistów, to można uznać za niewątpliwą zaletę umiejętność radzenia sobie w takich okolicznościach.

Dane zgromadzone podczas wykonywanych testów pozwalają dokonać dość szczegółowej analizy prób podejmowanych przez programistów. Dzięki temu udało się stwierdzić, że możliwość kompilacji programu i zasugerowania się ewentualnymi błędami niejednokrotnie wpłynęła na zmianę koncepcji podczas wykonywania zadania oraz pomogła podjąć decyzje odnośnie technik stosowanych w celu osiągnięcia pożądanego rezultatu.

Pozytywnie należy również ocenić umożliwienie uczestnikom możliwości sprawdzania poprawności pisanych przez nich programów za pomocą zestawów testów jednostkowych, których wyniki mogły zasugerować zastosowanie alternatywnych rozwiązań oraz wskazać słabe punkty aktualnego kodu.

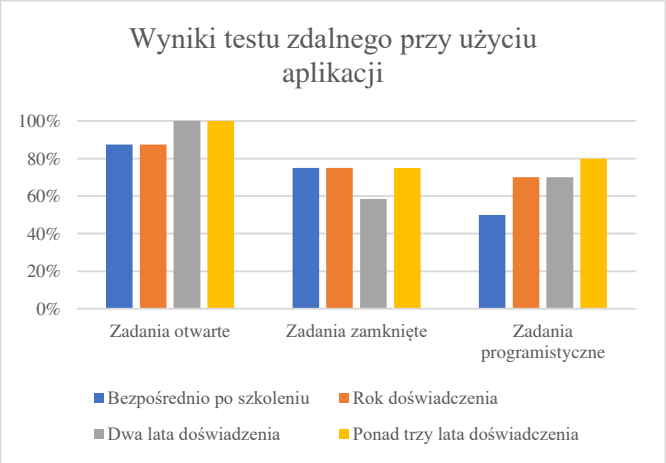
5.3. Zdalny test przeprowadzony przy użyciu stworzonej aplikacji

5.3.1. Założenia

Kolejną metodą, według której wykonano badania był zdalny test przeprowadzony przy użyciu przygotowanej aplikacji. W trakcie jego trwania uczestnicy mogli korzystać z dowolnych materiałów, ale zostali pouczeni o tym, że nie mogą się ze sobą komunikować. Modyfikacja wprowadzona w stosunku do poprzednio opisywanej metody miała na celu ukazanie rzeczywistego charakteru pracy programisty, który niejednokrotnie wspiera się różnymi źródłami, a także często ma dowolność, jeśli chodzi o czas i miejsce, w którym wykonuje swoją pracę.

5.3.2. Wyniki

Na Rys. 4 przedstawiono wyniki uzyskane ze zdalnie przeprowadzonego testu przy pomocy stworzonej aplikacji.



Rys. 4. Wyniki testu zdalnego przy użyciu aplikacji

5.3.3. Wnioski

Zadania teoretyczne w warunkach z dostępem do ogólno pojętych źródeł informacji zostały przez uczestników rozwiązywane bez większych problemów. Może to świadczyć o tym, że każdy z uczestników był w stanie zdobyć informacje niezbędne do udzielenia odpowiedzi na ww. pytania, co jest bardzo pożądaną cechą wśród programistów.

Dostrzeżono również fakt, iż uczestnicy badania, mając dostęp do dowolnych źródeł wiedzy, zdołali w krótszym czasie odpowiedzieć na pytania zamknięte i otwarte, dzięki czemu więcej czasu mogli poświęcić na dopracowanie rozwiązań zadań programistycznych.

Wyniki zadań programistycznych nie odbiegają znacznie od wyników przeprowadzonych z wykorzystaniem aplikacji w warunkach kontrolowanych. Można stwierdzić, że to przede wszystkim na ich podstawie należałoby oceniać badane osoby, ponieważ programowanie to umiejętność, którą nabywa się poprzez czasochłonną naukę oraz ćwiczenia. Nie jest możliwym znalezienie pełnego rozwiązania problemu programistycznego w ograniczonym czasie, gdy nie posiada się odpowiedniej wiedzy i umiejętności, a mając nawet nieograniczony dostęp do źródeł informacji.

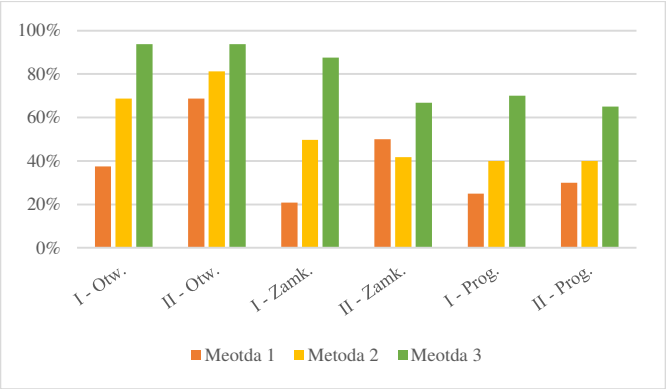
5.4. Porównanie wyników

Przeprowadzając i analizując badania wykonane różnymi metodami autorom pracy udało się zgromadzić dane, które mogą posłużyć do porównania poziomu skuteczności oceny

programistów, niezawodności działania, atrakcyjności użytkowania i uniwersalności stosowania.

5.4.1. Poziom skuteczności metod

Jako pierwszy czynnik, ocenie poddana została skuteczność weryfikacji wiedzy i umiejętności programistów. W tym celu uśrednione wyniki uzyskane przez uczestników testów w poszczególnych zadaniach zostały zestawione na wykresie przedstawionym na Rys. 2.



Rys. 5. Zestawienie średnich wyników w poszczególnych zadaniach w zależności od metody badawczej

Z Rys. 2 wynika, że istnieje dysproporcja pomiędzy wynikami zadań teoretycznych rozwiązywanych za pomocą różnych metod. Uczestnicy uzyskali najlepsze rezultaty w metodzie trzeciej - zdalnym teście przeprowadzonych w aplikacji. Przyczyn tego zjawiska można dopatrywać się w możliwości korzystania z materiałów pomocniczych w ww. metodzie. Dodatkowym czynnikiem może być również zredukowanie stresu poprzez umożliwienie zdawania testu w dogodnym miejscu i czasie. Nie można niestety również wykluczyć wpływu osób trzecich, które mogły pomagać rozwiązywać problemy badanym programistom.

Zauważalna jest także znaczna poprawa wyników zadań programistycznych przeprowadzonych przez aplikację względem testów pisemnych. Udogodnienia wynikające z możliwości kompilowania kodu, uruchamiania testów jednostkowych oraz podpowiadanie i kolorowanie składni na pewno pozytywnie wpłynęły na efektywność programistów. Podczas rozwiązywania testów pisemnych uczestnicy często się mylili, zmieniali koncepcje, próbowali różnych sposobów, ale zmagając się z aspektami technicznymi takimi jak nieznajomość interfejsów metod, niejednokrotnie zatrzymywali twórczy proces pisania kodu na rzecz prób rozwiązania trywialnych problemów ze składnią kodu.

Niepokojącą sytuacją natomiast może się wydawać fakt, iż wyniki uzyskane przez najmniej doświadczonych programistów w testach odbywających się za pośrednictwem aplikacji, nie odbiegały znacząco od rezultatów osiąganych przez osoby z najdłuższym stażem pracy. Sytuację tą można korygować odpowiednio dobierając poziom trudności zadań do poziomu zaawansowania kandydatów.

5.4.2. Niezawodność działania

Kolejnym kryterium oceny metod jest niezawodność ich działania. Niewątpliwie najlepiej w tej kategorii wypadają testy pisemne. Nie jest to metoda skomplikowana więc na ewentualne jej niepowodzenie wpływa najmniejsza liczba czynników. Odmienna sytuacja występuje w przypadku metod wykorzystujących aplikację przygotowaną na potrzeby pracy dyplomowej. Zachodzi w niej wiele procesów zależnych od siebie wzajemnie, zatem niepomyślne wykonanie jednego z nich może skutkować niepoprawnym działaniem aplikacji. Potencjalnych słabych punktów zaimplementowanego rozwiązania można dopatrywać się np. w komunikacji między modułami. Żądanie z aplikacji klienckiej musi dotrzeć do aplikacji serwerowej, ona zaś, musi się połączyć z bazą danych i pobrać kod z repozytorium Git udostępnionego na platformie GitLab, dysponując niezbędnymi danymi, aplikacja serwerowa wysyła odpowiedź do aplikacji klienckiej. Na każdym etapie ww. procesu może wystąpić zdarzenie, które spowoduje, że komunikacja zostanie zerwana. Pomimo tego, podczas testów nie odnotowano przypadków niepoprawnego działania aplikacji. Rozwiązaniem na potencjalne problemy wynikające z zakłóceń komunikacji pomiędzy modułami może być własna implementacja wszystkich modułów i tym samym uniezależnienie się od zagrożenia awaryjnością usług zewnętrznych dostawców.

5.4.3. Atrakcyjność użytkowania

Jako następny czynnik wpływający na ocenę metody, została oceniona atrakcyjność użytkowania aplikacji. Podczas badań od uczestników udało się zebrać cenne opinie na temat działania aplikacji, a także sugestie dotyczące jej usprawnień.

Na podstawie analizy zachowań osób biorących udział w eksperymencie oraz bezpośrednio otrzymywanych od nich uwag udało się ustalić, że nie były entuzjastycznie nastawione do testu pisemnego. Większość sądziła, że pisanie programów na kartce papieru jest niewygodne, bardziej czasochłonne i przyczynia się do popełniania błędów. Kolejnym argumentem był czas, który mieli poświęcić na rozwiązanie testu. Programiści niechętnie poświęcili godzinę swojego prywatnego czasu na udział w badaniach, ale mimo to przekonali się, że trudno było rozwiązać wszystkie zadania na czas.

Odmienne zdanie mieli natomiast uczestnicy badań z udziałem aplikacji, ponieważ nie kojarzyli testu z przykrym obowiązkiem, a traktowali go jak wyzwanie. Mogło to wynikać z ciekawości jak działa aplikacja, czy chęcią sprawdzenia nowego sposobu przeprowadzania testów. Po uczestnikach dało się zaobserwować większe rozluźnienie podczas rozwiązywania zadań, a poziom zaimplementowanych programów zdawał się odzwierciedlać nastrój programistów, gdyż były bardziej uporządkowane i przemyślane, niż w przypadku testów pisemnych.

Testy zdalne nie były obserwowane, ale z relacji uczestników wynika, że nie doświadczyli problemów

z działaniem aplikacji, a ich testy przebiegły bez zakłóceń. Od tych osób również otrzymano bardzo pozytywne opinie. Najbardziej chwalona była możliwość uruchamiania testów w trakcie rozwiązywania zadania, ponieważ wykazywały one pewne trudne do przewidzenia przypadki oraz niedociągnięcia programów. Na podstawie wyników testów programiści z łatwością identyfikowali i eliminowali wszelkie błędy. Uczestnicy lepiej kontrolowali czas rozwiązywania testu dzięki stale widocznemu zegarowi, a aplikacja uniemożliwiała jego przekroczenie, co miało miejsce w przypadku testów pisemnych.

6. Podsumowanie

Wybrany temat badań okazał się być niezwykle ciekawym zagadnieniem zarówno pod względem teoretycznym, jak i praktycznym. Autorom udało się przede wszystkim zgłębić wiedzę na temat dostępnych na rynku metod i zasadach ich działania, a także o ich docelowemu przeznaczeniu, wykorzystywanych przez nie technologiach oraz mocnych i słabych stronach każdej z nich.

Niezwykle interesującym wyzwaniem pod kątem technicznym była próba stworzenia prototypu aplikacji implementującej m.in. logikę automatycznej weryfikacji poprawności rozwiązania zadania programistycznego, którą można uznać za zakończoną wielkim sukcesem. Badania przeprowadzone z wykorzystaniem przygotowanego prototypu i przy udziale ochotników pracujących na co dzień w zawodzie programisty pozwoliły spojrzeć na proces weryfikacji wiedzy programistycznej z całkiem innej perspektywy. Dostrzeżono składowe całego procesu oraz analizowanych metod podatne na błędy ludzkie i niedoskonałości zaimplementowanych rozwiązań, które mogą skutkować znacznymi opóźnieniami, dodatkowymi kosztami, a przede wszystkim błędną oceną wiedzy i umiejętności badanej osoby.

Szacuje się również, że metody weryfikacji wiedzy i umiejętności programistów w najbliższym czasie wciąż będą rozwijane. Firmy oferujące tego typu rozwiązania będą dążyły do osiągnięcia jak najwyższej niezawodności, wszechstronności oraz zmniejszenia poziomu ingerencji osoby nadzorującej aplikację do niezbędnego minimum. Z tego typu systemów coraz częściej będą korzystały uczelnie szkolące przyszłych programistów oraz firmy poszukujące już wykwalifikowanych specjalistów.

Literatura

[1] Sedlak & Sedlak, „Prognozy wzrostu zatrudnienia i popytu w branży IT i telekomunikacji - Rynek Pracy” (2014), https://rynekpracy.pl/monitory/prognozy-wzrostu-zatrudnienia-i-popytu-w-branzy-it-i-telekomunikacji. [23.06.2018]

[2] S. Shahida, R. Rohaida i Z. Z. Kamal, „Improving Automated Programming Assessments: User Experience Evaluation Using FaSt-generator,” w The Third Information Systems International Conference (2015).

[3] K. Cooper i L. Torczon, Engineering a Compiler 2nd Edition, Elsevier, 2011

[4] A. V. Aho, R. Sethi i J. D. Ullman, Kompilatory. Reguły, metody i narzędzia, WNT, 2002

[5] R. Osherove, The art of Unit Testing, Manning Publications, 2013

[6] T. Kaczanowski, Złe testy, dobre testy, 2016.

[7] M. Janicki i K. Strzecha, Zastosowanie statycznej analizy do walidacji kodu języka Java, Wydawnictwa AGH, 2004.

[8] G. A. Campbell, "Cognitive Complexity. A new way of measuring undertandability," SonarSource, (2018), https://www.sonarsource.com/docs/CognitiveComplexity.pdf.

[9] Codility, „The Codility Task Library | Codility Help Center” (2018), http://support.codility.com/screening-candidates-with-codecheck/the-codility-task-library [23.06.2018]

[10] M. Jackson, S. Crouch i R. Baxter "Software Evaluation: Criteria-based Assessment" Software Sustainability Institute, 2011

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 8 (2018) 277-281
Wysłane: 2018-06-18
Przyjęte: 2018-06-27

Badanie możliwości platformy Arduino w kontekście monitorowania zagrożeń środowiskowych

Krzysztof Lenart*, Małgorzata Plechawska-Wójcik
Politechnika Lubelska, Instytut Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. Artykuł opisuje badania polegające na analizie możliwości jakie oferuje platforma Arduino w kontekście monitorowania środowiska i wykrywania zagrożeń. Do realizacji badan wykorzystano czujniki kompatybilne z Arduino umożliwiające monitorowanie środowiska. Badania polegały na monitorowaniu parametrów środowiska, monitorowano m. in. temperaturę i wilgotność powietrza, poziom natężenia dźwięku lub gazy szkodliwe dla zdrowia. Na podstawie uzyskanych rezultatów nastąpiła analiza możliwości platformy.

Słowa kluczowe: Arduino; zagrożenia środowiskowe; czujniki

*Autor do korespondencji.

Adres e-mail: krzysieklenart@gmail.com

Possibility analysis of environmental threat monitoring with the Arduino platform

Krzysztof Lenart*, Małgorzata Plechawska-Wójcik
Institute of Computer Science, Lublin University of Technology, Nadbystrzycka 36B,20-618 Lublin, Poland

Abstract. The paper describes results of the possibility analysis of environmental monitoring and detection threats with the Arduino platform. Sensors compatible with Arduino enabling environmental monitoring were used to conduct research. The research consisted in monitoring environmental parameters, monitoring among others air temperature and humidity, sound level or gases harmful to health., Capabilities of the platform have been analyzed based on the obtained results.

Keywords: Arduino; environmental threats; sensors

*Corresponding author.

E-mail address krzysieklenart@gmail.com

1. Wstęp

W dobie informatyzacji coraz więcej elementów otoczenia wypełniają urządzenia, które pozwalają kontrolować większość ludzkiego życia bez ingerencji innych ludzi. Obecnie powszechne są urządzenia monitorujące czynności życiowe, pozwalają one m. in. monitorować puls, ciśnienie krwi lub obserwować fazy snu [1]. W otoczeniu którym przebywa człowiek występują różne zagrożenia środowiskowe, które mogą niekorzystnie wpływać na stan ludzkiego zdrowia lub nawet mu zagrażać. Takimi zagrożeniami są miedzy innymi: wysoka temperatura oraz wilgotność powietrza, hałaś, gazy szkodliwe dla zdrowia lub pożar. Bez odpowiedniego monitoringu ludzie mogą być nieświadomi że ich życiu zagraża niebezpieczęstwo.

Rozwiązaniem problemu jest mikrokontroler Arduino oraz zestaw czujników umożliwiających monitorowanie środowiska. Arduino cieszy się wysoką popularnościa wsród kontrolerów. Dodatkowym autem jest jego dostępność na licencji wolnego oprogramowania, oznacza to że jedyne koszty jakie są ponoszone to zakup platformy oraz komponentów.

Celem artykułu jest przeprowadzenie analizy czy platforma Arduino umożliwia monitorowanie zagrożeń środowiskowych. Badania będą polegały na zgromadzeniu danych z czujników umożliwiających monitorowanie parametrów środowiska a następnie ich przenalizwanie.

Celem pośrednim było zaprojektowanie oraz zaimplementowanie aplikacji do przechowywania oraz prezentowania zebranych danych.

1.1. Przegląd literatury

Tematyka monitorowania środowiska [2],[3] przy użyciu platformy Arduino [4] staje się coraz bardziej popularna, stąd też liczba artykułów naukowych oraz książek o tej tematyce wzrasta.

Jednym z prekursorskich artykułów naukowych jest „A Cost-effective Wireless Sensor Network System for Indoor Air Quality Monitoring Aplications" [5]. Publikacja nastąpiła w 2014, autorami są Sherin Abraham oraz Xinrong Li. Autorzy zainspirowani artykułem „Indoor air quality tools for schools Communications guide" [6] mówiącym że zanieczyszczone powietrze w rankingu zagrożeń zdrowia publicznego znajduje się w pierwszej piątce postanowili zaprojektować i zbudować tani system monitorowania środowiska.

Kolejnym artykułem traktującym o monitorowaniu środowiska jest „Open Source data logger for low-cost environmental monitoring" [7], którego autorem jest Ed Baker. Założenia projektowe były podobne jak w przypadku wyżej opisanego projektu z tą różnicą że autor zaprezentował możliwość zgromadzenia danych bez wykorzystania komunikacji sieciowej.

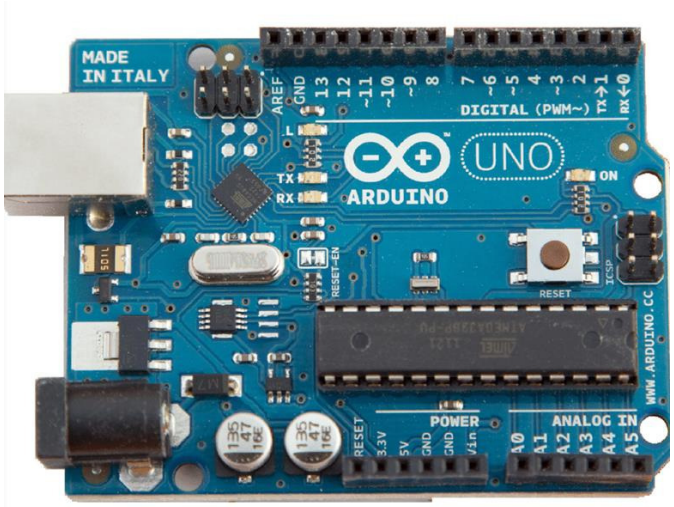
Publikacja „Environmental Monitoring with Arduino” [8] autorów Emilly Gertz i Patrick Di Justo wydana w 2012 roku przez O’Reilly Media dotyczy projektowania oraz budowy układów służących do monitorowania środowiska przy użyciu Arduino. Celem publikacji jest wprowadzenie do narzędzi oraz programowania mikrokontrolera. Przedstawia proste, budżetowe rozwiązanie umożliwiające monitorowanie różnych aspektów środowiska.

Artykuł „Development of a Low-Cost Arduino-Based Sonde for Coastal Applications” [9] ukazuję wykorzystanie Arduino jako sondy do badania obszarów przybrzeżnych. Autorzy udowodnili że system monitorujący oparty na Arduino może być świetną alternatywna dla profesjonalnych systemów, ponieważ cechuje się dużą wytrzymałością, niskim kosztem budowy, możliwością dostosowywania według zapotrzebowania oraz posiada darmowe wsparcie techniczne.

Powyższe publikacje ukazują że monitorowanie środowiska przy użyciu platformy Arduino staje się niezwykle popularne.

2. Platforma Arduino

Arduino jest popularną platformą zaprojektowaną we Włoszech w 2005 roku dla systemów wbudowanych. Arduino oparte jest na projekcie open hardware, oznacza to że platforma może być dowolnie modyfikowana przez użytkowników. Arduino może być wykorzystywane do tworzenia samodzielnych układów lub ich sterowania. Płytka Arduino standardowo zawiera mikrokontroler oraz wyprowadzone linie analogowe i cyfrowe, które mogą pracować jako wejście lub wyjście, oznacza to że można przeprowadzać określone interakcje sprzętowe. Zawiera również port USB, który służy do połączenia płytki z komputerem w celu sterowania układem jak również zaprogramowaniu mikrokontrolera poprzez wgranie programu. Umieszczone zostało również złącze umożliwiające doprowadzenie zasilania np. przy użyciu ładowarki sieciowej [10]. Platforma została zaprezentowana na rysunku 1.



Rys. 1. Platforma Arduino

2.1. Czujniki

Czujniki rozszerzają możliwości platformy Arduino. Podłączane są przy użyciu pinów analogowych lub pinów cyfrowych. Czujniki podłączone przez pin analogowy umożliwiają odczyt konkretnej wartości, natomiast podłączone pinem cyfrowym dostarczają informacje o wystąpieniu zdarzenia. W badaniach użyto następujące czujniki:

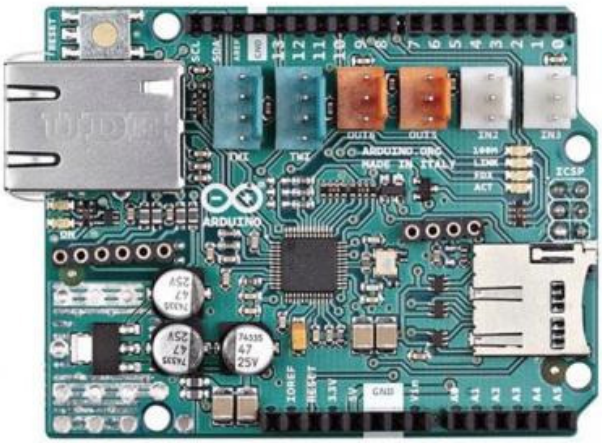
- moduł temperatury i wilgotności,
- czujnik wilgotności gleby,
- moduł z czujnikiem dźwięku,
- moduł opadów deszczu,
- czujnik pochylenia,
- czujnik płomienia,
- moduł MQ2 – czujnik gazów.

Przykładowy czujnik został przedstawiony na rysunku 2.

Rys. 2. Moduł MQ-2 – czujnik gazów

2.2. Moduły rozszerzające

W celu gromadzenia i przesyłania danych zostały zastosowane moduły rozszerzające funkcjonalność płytki Arduino. Rozszerzenia umożliwiają miedzy innymi: zapis zgromadzonych danych na kartę pamięci SD, przesłanie danych przez połączenie przewodowe przewodem RJ45 z routerem, przesył danych bezprzewodowo przy użyciu sieci WiFi. Przykładowy moduł rozszerzający został zaprezentowany na rys. 3.



Rys. 3. Moduł Ethernet Shield

3. Procedura badawcza

Celem przeprowadzonych badan było sprawdzenie czy platforma Arduino umożliwia monitorowanie zagrożeń

środowiskowych. Monitorowanie środowiska odbywało się na podstawie następujących scenariuszy:

- pierwszy scenariusz polegał na długoterminowej obserwacji środowiska, dane były zbierane cyklicznie, w stałym odstępie czasowym. Celem pierwszego scenariusza było przeanalizowanie parametrów środowiska w lesie oraz określenie wartości granicznych, mogących zwiększyć zagrożenie pożarem,
- drugi scenariusz polegał na symulacji zagrożenia jakim jest pożar w budynku,
- trzeci scenariusz polegał na symulacji trzęsienia ziemi,
- czwarty scenariusz polegał na symulacji wycieku gazu,
- piąty scenariusz polegał na symulacji poziomów wilgotności powietrza.

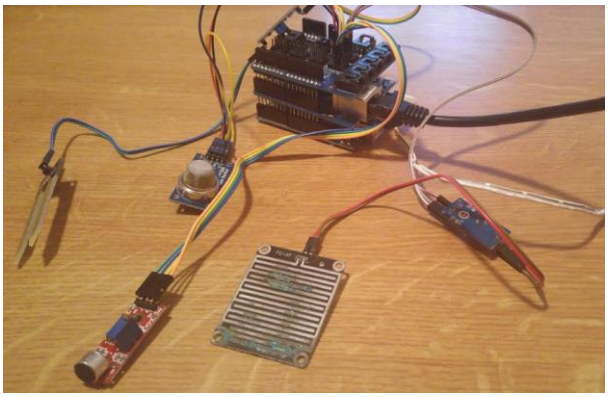
Odczytane dane z czujników były zapisywane w aplikacji, która powstała na potrzeby badan. Dane były gromadzone na trzy sposoby:

- komunikacja bezprzewodowa – odczytane wartości były natychmiastowo przesyłane do systemu przez sieć WiFi,
- komunikacja przewodowa – odczytane dane były natychmiast przysyłane przy użyciu przewodu RJ45 podłączonego do routera
- zapis na karcie SD – odczytane wartości były zapisywane na karcie pamięci podłączonej do Arduino. Dane były dostępne dopiero po ręcznym wprowadzeniu do systemu przez użytkownika.

Sposób gromadzenia danych ma wpływ na szybkość przekazywania informacji o zagrożeniu. W przypadku komunikacji przewodowej i bezprzewodowej odczyty były dostępne natychmiastowo, natomiast w przypadku zapisu na karcie pamięci dane były dostępne po ręcznym wprowadzeniu do systemu.

4. Budowa stacji monitorującej

Stacja monitorująca może być zbudowana na trzy sposoby w zależności od sposobu zapisu danych. Częścią wspólną rozwiązań są jest platforma Arduino Uno Rev 3, Sensor Shield oraz czujniki. W komunikacji przewodowej został użyty Ethernet Shield, do komunikacji bezprzewodowej posłużył moduł WiFi ESP-01, natomiast w przypadku braku możliwości komunikacji wyżej wymienionymi metodami został użyty moduł kart pamięci SD. Rozwiązanie przewodowe zostało zaprezentowane na rysunku 4.



Rys. 4. Stacja monitorująca – połączenie przewodowe

4.1. Aplikacja gromadząca dane

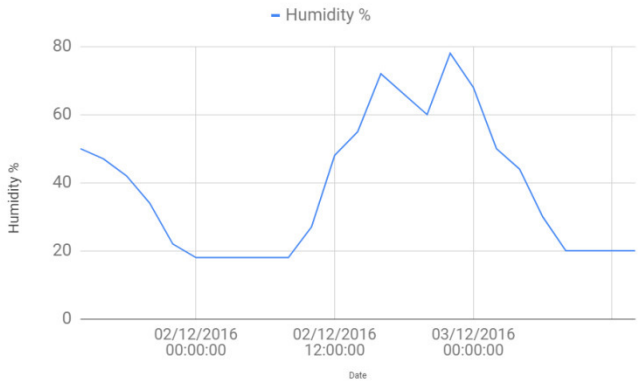
W ramach artykułu została stworzona aplikacja webowa, której przeznaczeniem jest gromadzenie danych otrzymanych z platformy oraz ich graficzna prezentacja.

5. Prezentacja zgromadzonych danych

Zgromadzone dane środowiskowe zostały przedstawione na wykresach graficznych. Przedstawione zostały dane z wybranych czujników.

5.1. Wilgotność powietrza

Dane zostały odczytane przy użyciu czujnika temperatury i wilgotności. Monitorowanie wilgotności powietrza odbyło się przy wykorzystaniu scenariusza piątego.

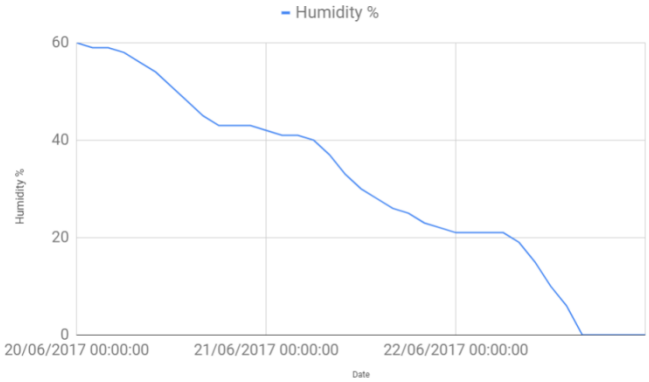


Rys. 5. Wilgotność powietrza

Na powyższym wykresie zostały zaprezentowane dane zebrane przez czujnik wilgotności i temperatury na przestrzeni trzech dni w okresie od 01.12.2016 do 03.12.2016. Dane odczytywane były w odstępie dwóch godzin.

5.2. Wilgotności gleby

Dane zostały odczytane przy użyciu modułu do pomiaru wilgotności gleby. Monitorowanie wilgotności gleby odbyło się przy wykorzystaniu scenariusza pierwszego.



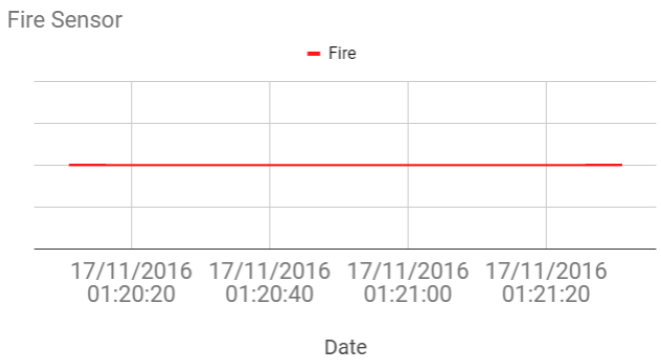
Rys. 6. Wilgotność gleby w procentach

Na powyższym wykresie zostały zaprezentowane dane zebrane przez czujnik wilgotności gleby na przestrzeni trzech dni w okresie od 20.06.2017 do 23.26.2017. Dane odczytywane były w odstępie dwóch godzin. Z wykresu możemy odczytać poziom wilgotności. W ciągu dnia pod wpływem słońca gleba zostawała stopniowo osuszana, co

widać na wykresie. W ciągu trzech dni gleby została całkowicie wysuszona.

5.3. Wykrywanie ognia

Detekcja ognia nastąpiła przy użyciu czujnika płomieni. Pomiar odbył przy użyciu scenariusza drugiego: symulacji zdarzenia.

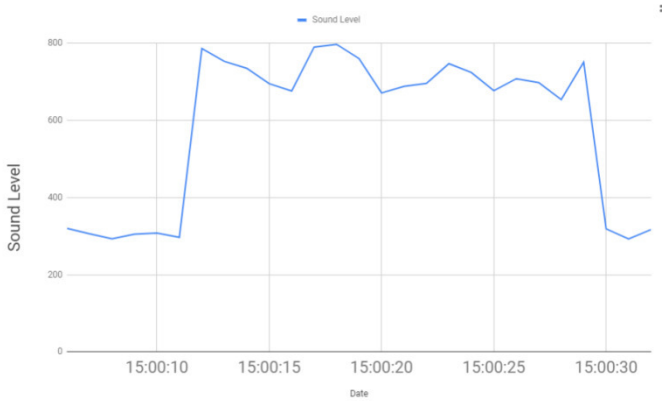


Rys. 7. Czas wykrycia ognia

Jak widać na rysunku 7 czujnik wykrył ogień o godzinie 1:20:15 dnia 17/11/2016. Ogień trwał minutę i piętnaście sekund. Całe zdarzenie było symulowane, źródło ognia znajdowało się w obrębie działania czujnika.

5.4. Monitorowanie poziomu natężenia dzwięku

Do monitorowania natężenia dźwięku został użyty moduł z czujnikiem dźwięku. Pomiar odbył przy użyciu scenariusza drugiego: symulacji zdarzenia.



Rys 8. Poziom natężenia dźwięku

Czujnik nie pozwala jednoznacznie określić poziomu natężenia dźwięku w decybelach. Niezbędne są tutaj punkty odniesienia. Odczytana wartość dla ciszy wynosi około 100, natomiast dla rozmowy wynosi około 300. Na podstawie punktów odniesienia można ustalić progi głośności.

6. Analiza zgromadzonych danych

Dla wilgotności jako wartości progowe przyjęto: wilgotność wyższą niż 60% oraz niższą niż 40%. Analizując wykres przedstawiony na rysunku 4 można stwierdzić że czujnik wykrywa zagrożenia środowiskowe. Czujnik zarejestrował wartości powyżej 60% lub poniżej 40%. Wartości te wpływają negatywnie na zdrowie i samopoczucie człowieka.

Dla czujnika wilgotności gleby jako wartości progowe oznaczające zagrożenie przyjęto wartości mniejsze niż 30%. Dane otrzymane z czujnika wilgotności gleby przedstawione na rysunku 5 również umożliwiają wykrywanie zagrożeń, wartości poniżej 30% oznaczają suchą glebę. Dane umożliwiają utrzymanie wilgotności gleby na wymaganym poziomie.

Kolejnym czujnikiem, który posłużył do badan był czujnik płomienia. Wartością oczekiwaną w przypadku pożaru jest otrzymanie sygnału z czujnika. Przeprowadzone symulacje pozwalają jednoznacznie stwierdzić, że czujnik bezbłędnie wykrywa zagrożenie środowiskowe jakim jest ogień. Wadą czujnika jest to że sam nie posiada ochrony przed ogniem.

Ostatnim czujnikiem opisanym w artykule jest czujnik dźwięku. .Czujnik nie umożliwia jednoznacznego określenia poziomu natężenia dźwięku w decybelach. Przeprowadzenie symulacji zdarzeń umożliwia określenie natężenia dźwięków dla charakterystycznych zdarzeń, na podstawie których można zdefiniować próg natężenia dźwięku, który będzie niebezpieczny dla człowieka. Wartości powyżej 600 oznaczają zagrożenie.

W artykule zostały przeanalizowane wybrane czujniki. Wartości progowe dla wszystkich czujników zostały przedstawione w tabeli 1.

Tabela 1. Wartości progowe czujników

Scenariusz	Nazwa czujnika	Wartość progowa
Scenariusz pierwszy	Czujnik wilgotności i temperatury	Temperatura powyżej 20 stopni Celsiusa
Scenariusz pierwszy	Czujnik wilgotności gleby	Wilgotność gleby mniejsza niż 30%
Scenariusz pierwszy	Czujnik opadów deszczu	Brak opadów na przestrzeni trzech dni
Scenariusz drugi	Czujnik natężenia dźwięku	Wartości powyżej 600
Scenariusz drugi	Czujnik płomienia	Wykrycie zdarzenia
Scenariusz trzeci	Czujnik pochylenia	Wykrycie zdarzenia
Scenariusz czwarty	Czujnik gazów MQ2	Wykrycie gazu o stężeniu większym niż 300ppm
Scenariusz piąty	Czujnik wilgotności i temperatury	Wilgotność wyższa niż 60% lub niższa niż 40%

7. Wnioski

Na podstawie zgromadzonych danych można wyciągnąć następujące wnioski:

- Arduino oparte jest na licencji open hardware, co umożliwia jego rozbudowę i dostosowywane,
- Arduino posiada wiele rozszerzeń umożliwiających komunikacje, nie ma ograniczenia co do miejsca zbierania danych,
- dostępnych jest wiele sensorów umożliwiających monitorowanie wielu aspektów środowiska,
- Arduino jak i sensory są powszechnie dostępne, za niewielką cenę można zbudować zaawansowaną stacje monitorującą środowisko.

8. Podsumowanie

Celem pracy było zbadanie możliwości platformy w kontekście monitorowania środowiska. Po przeprowadzeniu badań oraz ich przeanalizowaniu można stwierdzić że Arduino może być wykorzystane do monitorowania zagrożeń. Niewielki koszt budowy, licencja open hardware oraz dostępność różnorodnych czujników jest ciekawą alternatywą dla drogich profesjonalnych stacji monitorujących.

Literatura

[1] Dung Phan, Lee Yee Siong, Pubudu N. Pathirana, Aruna Senevirante: Smartwach: Performance evaluation for long –term heart rate monitoring. ISBB, pp. 144-147, 2015.

[2] Ashenafi Lambebo, Sasan Haghani: A Wireless Sensor Network for Environmental Monitoring of Greenhouse Gases. Zone I Conference, 2014, University of Bridgeport, Bridgpeort, CT, USA.

[3] Akram Syed Ali, Zachary Zanzinger, Deion Debose, Brent Stephens: Open Source Building Science Sensors (OSBSS): A low-cost Arduino-based platform for long-term indoor environmental data collection. Building and Environment Volume 100, 2016, pp. 114-126.

[4] Michael Margolis - Arduino Cookbook, 2nd Edition, 2011.

[5] Sherin Abraham, Xinrong Li: A Cost-effective Wireless Sensor Network System for Indoor Air Quality Monitoring Aplications. Procedia Computer Science Volume 34, 2014, pp. 165-171.

[6] United States Environmental Protection Agency's (EPA): Indoor air quality tools for schools Communications guide. 2009.

[7] Mr Ed Baker: Open Source data logger for low-cost environment al monitoring. Biodiversity Data Journal, 2014.

[8] Emily Gertz, Patrick Di Justo: Environmental Monitoring with Arduino. Wydanie I, O'Relly, 2012.

[9] Grant Lockridge, Brian Dzwonkowski, Reid Nelson, Sean Powers: Low-Cost Arduino-Based Sonde for Coastal Applications. Sensors, 2016.

[10] Alessandro D'Ausilio: Arduino: A low-cost multipurpose lab equipment. Behavior research methods, 2012.

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 8 (2018) 282-285
Wysłane: 2018-06-24
Przyjęte: 2018-06-27

Analiza porównawcza aplikacji mobilnych do zarządzania projektem informatycznym

Ewelina Wlaszczyk*, Elżbieta Miłosz
Politechnika Lubelska, Instytut Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. Artykuł prezentuje rezultaty analizy porównawczej wybranych aplikacji dla systemu mobilnego Android służących do zarządzania projektami informatycznymi. Badania przeprowadzono przy użyciu analizy wielokryterialnej na temat funkcjonalności, intuicyjności i wyglądu interfejsu oraz dostępnych integracji danych aplikacji.

Słowa kluczowe: analiza porównawcza; zarządzanie projektem; android

*Autor do korespondencji.

Adres e-mail: ewelina.a.wlaszczyk@gmail.com

Comparative analysis of mobile applications for IT project management

Ewelina Wlaszczyk*, Elżbieta Miłosz
Institute of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract. The article presents the results of a comparative analysis of selected applications for Android mobile system used to manage IT projects. The research was carried out using multicriteria analysis on the functionality, intuitiveness and appearance of the interface and the available data integration of the application.

Keywords: Comparative analysis; project management; android

*Corresponding author.

E-mail address: ewelina.a.wlaszczyk@gmail.com

1. Wstęp

Bardzo szybki rozwój branży komputerowej oraz mobilnej spowodował rosnące zapotrzebowanie na nowe oprogramowanie. Ludzie zauważyli jak bardzo może ono ułatwiać życie i wykonywanie wielu czynności. W związku z tym ciągle rozwija się rynek oprogramowania i nic nie wskazuje na zmianę tej sytuacji. Aktualnie każda organizacja bądź firma korzysta z oprogramowania komputerowego i coraz częściej również mobilnego.

Proces wytwarzania oprogramowania jest pierwszym etapem zarządzania projektem. Z reguły jest on złożony, wymaga on bardzo dobrej organizacji, ale również narzędzi które ułatwią nad nim pracę.

W tym celu powstały metodyki zarządzania wytwarzaniem oprogramowania które dzielą się na trzy grupy [1]: strukturalne (metody wodospadowa, równoległa i etapowa), szybkie metody (metoda etapowa, rapid, prototypowanie), oraz zwinne (agile, programowanie ekstremalne). Powstały również metodyki, którą pozwalają na efektywne zarządzanie projektem informatycznym są to na przykład: PRINCE2, PMBoK Guide, SCRUM oraz TenStep [1,2].

Innym narzędziem usprawniającym proces tworzenia oprogramowania, a następnie jego zarządzaniem są aplikacje stworzone w tym celu, niewątpliwie zainspirowane powyższymi metodykami lub przystosowane do wybranych z nich. Istnieje bardzo dużo aplikacji webowych, a wraz ze wzrostem popularności smartfonów powstaje coraz więcej aplikacji przeznaczonych na platformy mobilne.

Dzięki zastosowaniu wybranej metodyki wraz z wykorzystaniem aplikacji do usprawnienia wytwarzania i późniejszego zarządzania projektem, można bardzo dobrze zorganizować ten proces i ułatwić pracę całemu zespołowi.

Celem artykułu jest porównanie aplikacji mobilnych służących do zarządzania projektem informatycznym. Przeprowadzona zostanie analiza porównawcza wielokryterialna której szczegółowy plan zostanie omówiony w rozdziale 3.

2. Przegląd metodyk zarzadzania projektem

Model kaskadowy

Bardzo znanym modelem jest model kaskadowy, wprowadzony w 1970 roku przez Winstona W. Royca. Polega on na podziale projektu na sześć etapów, planowanie, analizę, projekt, implementację, testowanie oraz pielęgnację [3]. Aby przejść do kolejnego etapu wymagane jest wcześniejsze ukończenie poprzedniego.

Model kaskadowy jest statyczny, posiada dobrze zdefiniowane etapy pracy. Współcześnie rezygnuje się z korzystania z tego modelu, ponieważ aby projekt zakończył się sukcesem wymagania musiałyby się nie zmieniać w trakcie prac, co nie zdarza się często. Brak elastyczności w tym modelu jest dużą wadą. W fazie analizy ciężko przewidzieć błędy które nastąpią w trakcie prac nad projektem, a powrót do poprzednich faz powoduje dodatkowe koszty.

Model przyrostowy

Model przyrostowy powstał celem ulepszenia modelu kaskadowego, wyeliminowaniu jego wad. Model ten składa się z kilku etapów w którym każdy posiada następujące fazy: wymagań, analizy i projektu, testowania i oceny [4].

Model ten zdecydowanie jest bardziej elastyczny niż opisywany w poprzednio model kaskadowy, pozwala on na wprowadzenie zmian w wymaganiach w trakcie trwania projektu co czyni go bardziej praktycznym. Pozwala on też na wcześniejsze rozpoznanie problemów i ich eliminację bez ponoszenia dodatkowych kosztów.

Zwinne zarządzanie projektami

Metodyki zwinne są podejściem przyrostowym i alternatywą dla wcześniej omawianego modelu kaskadowego. Manifest Zwinnego Wytwarzania Oprogramowania, zwany również Manifestem Agile przedstawia zbiór wspólnych zasad stosowanych przy pracy nad projektami, które są następujące [5]:

„Odkrywamy nowe metody programowania dzięki praktyce w programowaniu i wspieraniu w nim innych.

W wyniku naszej pracy, zaczęliśmy bardziej cenić:

Ludzi i interakcje od procesów i narzędzi

Działające oprogramowanie od szczegółowej dokumentacji

Współpracę z klientem od negocjacji umów

Reagowanie na zmiany od realizacji założonego planu.

Oznacza to, że elementy wypisane po prawej są wartościowe, ale większą wartość mają dla nas te, które wypisano po lewej.”

Metodyki które powstały na podstawie Manifestu to między innymi Feature Driven Development, Extreme Programming i Scrum.

Scrum

Aktualnie najpopularniejszą metodą zarzadzania projektem jest SCRUM który, został stworzony we wczesnych latach dziewięćdziesiątych, przez Kena Schwabera i Jeffa Sutherlanda. Natomiast po jego przedstawieniu na konferencji OOPSLA w 1995 roku zaczął wypierać opisywany we wcześniejszym podrozdziale model kaskadowy, który dominował w tamtym czasie na rynku.

Scrum jest sposobem zarządzania i organizacji pracy zespołu, dzięki któremu można efektywnie wykonać projekt. Opiera się on na trzech filarach: przejrzystości, adaptacji oraz inspekcji [6,7].

Zespół pracuje w ustalonym okresie czasu zwanym sprintem, który powinien mieć stałą długość i trwać od jednego do czterech tygodni. Za każdym zakończonym sprintem powinna zostać dostarczona działająca wersja produktu.

Wyróżnione są 3 role w scrumie [9]:

- Zespół – są to osoby, które mają umiejętności i zasoby, aby wykonać projekt;
- Product Owner – jest to właściciel projektu, który reprezentuje klienta;
- Scrum Master – jego rolą jest pomoc zarówno zespołowi jak i Product Ownerowi, wspiera on też samoorganizacje i moderuje spotkania;

W metodyce tej występują następujące wydarzenia [9]:

- Planowanie Sprintu - polega na stworzeniu planu działania w najbliższym sprincie;
- Sprint - jest etapem realizacji zadań i trwa określony z góry czas;
- Codzienny Scrum - jest to codzienne krótkie spotkanie dla zespołu Deweloperskiego na którym omawiane są zadania wykonane, do wykonania oraz napotkane problemy przez każdego z członków zespołu;
- Retrospektywa Sprintu - występuje po zakończeniu jednego sprintu i przed zakończeniem kolejnego, prowadzi dyskusje na temat sukcesów oraz rzeczy, które należy poprawić w następnym sprincie;

3. Plan badań

Celem badań było przeprowadzenie analizy porównawczej aplikacji mobilnych do zarządzania projektem informatycznym. Ze względu na dominację systemu mobilnego Android na rynku, to aplikacje dla niego przeznaczone zostały porównane.

Na rynku dostępnych jest wiele aplikacji przeznaczonych do zarządzania, wybrane zostały:

1) Asana
2) Basecamp
3) Monday
4) Teamwork
5) Wrike

Kryteriami wyboru omawianych aplikacji była ich dostępność w wersji próbnej lub darmowej oraz popularność ich wersji webowych jak i mobilnych.

Analizując założenia twórców aplikacji oraz ich deklaracje na temat omawianych aplikacji, można zauważyć wiele zbieżności i wywnioskować, że narzędzia te nie będą się od siebie znacząco różniły.

W pracy postawiono następującą tezę badawczą: omawiane aplikacje są takie same pod względem funkcjonalności i przyjazności interfejsu użytkownika. Do weryfikacji słuszności tej tezy jako metodę badawczą przyjęto analizę wielokryterialną z następującymi kryteriami:

1) Funkcjonalności:
    a. dostępność rejestracji z poziomu aplikacji mobilnej
    b. tworzenie tablicy projektu
    c. śledzenie aktywności
    d. czat
    e. komentowanie
    f. śledzenie statusu zadania
    g. wykres Gantta
    h. definiowanie celów
    i. raporty i statystyki
    j. wyszukiwanie
    k. kalendarz zespołu
    l. tworzenie listy zadań
2) Intuicyjność interfejsu
3) Wygląd aplikacji
4) Integracje

Na podstawie omówionych wyżej kryteriów każdej z aplikacji przyznano punkty, za posiadanie funkcjonalności 1 punkt, za jej brak 0 punktów. Natomiast intuicyjność interfejsu, wygląd aplikacji oraz integracje oceniane były w skali 1 – 3, na podstawie subiektywnej oceny, z punktu widzenia nowego użytkownika. Aplikacje oceniane były przez jedną osobę.

4. Wyniki analizy

Analizę przeprowadzono korzystając z aplikacji na urządzeniu z systemem android wersji 6.0.1. W każdej z aplikacji wykonano czynności, które pozwoliły zweryfikować podane analizie funkcjonalności. Punkty przyznane za intuicyjność interfejsu oraz jego wygląd zostały ocenione subiektywnie na podstawie zgromadzonych danych oraz wrażenia odniesionego przy sprawdzaniu dostępnych funkcjonalności z punktu widzenia nowego użytkownika aplikacji. Integracje oceniono na podstawie ilości dostępnych narzędzi kompatybilnych z daną aplikacją.

Tabela 1. Wyniki analizy porównawczej

	Asana	**Basecamp**	**Monday**	**Teamwork**	**Wrike**
Rejestracja użytkownika	1	0	1	0	0
Tworzenie tablicy projektu	0	1	1	0	1
Śledzenie aktywności	1	1	1	1	1
Czat	1	1	1	0	0
Komentowanie	1	1	1	1	1
Śledzenie statusu zadania	1	0	1	0	1
Wykres Gantta	0	0	1	1	1
Definiowanie celów	1	0	0	1	1
Raporty i statystyki	1	1	1	1	1
Wyszukiwanie	1	1	1	0	0
Kalendarz zespołu	1	0	0	1	1
Tworzenie listy zadań	1	1	1	0	0
Intuicyjność interfejsu	2	2	3	2	2
Wygląd aplikacji	2	2	3	1	2
Integracje	3	2	1	3	3
Suma	**17**	**13**	**17**	**12**	**15**

Wyniki przeprowadzonej analizy przedstawiono w tabeli 1, każda z aplikacji została oceniona zgodnie z planem. Największą liczbę punktów – 17 uzyskały dwie aplikacje Asana i Monday, natomiast najmniejszy wynik 12 punktów został uzyskany dla aplikacji Teamwork. Maksymalna liczba punktów to 21.

Jak widać żadna z aplikacji nie uzyskała maksymalnej liczby punktów, nie posiadała wszystkich funkcjonalności ani takiego samego zestawu narzędzi.

Tabela 2 Zestawienie podstawowych danych o testowanych aplikacjach ze sklepu google play

	Asana	**Basecamp**	**Monday**	**Teamwork**	**Wrike**
Ocena ze sklepu	4,3	4,4	4,6	4,1	4,3
Liczba głosów	19 579	2 498	249	1 189	4 978
Rok powstania	2015	2013	2014	2013	2012
Wspiera-na wersja androida	4.0	4.4	4.0.3	4.1	4.0.3
Wersja darmowa	Tak	Okres próbny 30 dni	Okres próbny 14 dni	Tak	Tak
Liczba pobrań aplikacji	1 000 000+	100 000+	50 000+	100 000+	100 0 00+

Tabela 2 przedstawia zestawienie informacji na temat porównywanych aplikacji takich jak ocena użytkowników i liczba ich opinii ze sklepu google play, rok wydania aplikacji dla systemu android, wspierana wersja systemu oraz informacje na temat darmowej wersji. Wszystkie te dane pochodzą ze stron sklepu google play [10,11,12,13,14].

Aplikacje wspierają system od wersji 4.0, co daje gwarancję, że będą one mogły działać na ponad 90% urządzeń użytkowników. Podczas gdy aplikacje jak Basecamp i Teamwork wspierają wersje 4.4 i 4.1 zapewniają działanie na 100% urządzeń z androidem z których aktualnie korzystają użytkownicy tego systemu.

System android został zaprezentowany już w 2008 roku, a w 2013 był najpopularniejszym systemem mobilnym na świecie. Pierwsza z omawianych aplikacji, która powstała dla tego systemu to Wrike w 2012, a wzrost popularności systemu skutkował, że w 2013 roku powstały Basecamp i Teamwork. Jak można zauważyć w tabeli 2, rok powstania nie wpływa na liczbę pobrań aplikacji, gdyż to najmłodsza aplikacja ma największą liczbę pobrań i instalacji na systemie android.

5. Wnioski

Celem pracy było porównanie wybranych aplikacji mobilnych dla systemu android służących do zarządzaniem projektami informatycznymi. Cel ten ostał zrealizowany, a porównane aplikacje to: Asana, Basecamp, Monday, Teamwork, Wrike.

W ramach artykułu przedstawiono również najważniejsze metodyki zarządzania projektami, które niewątpliwie stanowiły inspirację do tworzenia aplikacji ułatwiających ten proces.

Zdecydowanie najważniejszym elementem przedstawionych narzędzi jest ich funkcjonalność, dlatego porównano dostępne podstawowe funkcje takie jak tablice projektu lub inny sposób ich reprezentacji, narzędzia komunikacyjne dla zespołu, oraz ogólną nawigację po menu aplikacji.

Kolejnym bardzo ważnym czynnikiem w przypadku aplikacji tego typu jest interfejs użytkownika. System android działa na znacznie mniejszych ekranach w porównaniu do wersji przeglądarkowych. Podczas gdy narzędzia do zarządzana projektami mają bardzo dużo różnych funkcjonalności oraz wiele informacji, które należy wyświetlić, bardzo ważne jest zachowanie przejrzystości i intuicyjności interfejsu.

Bardzo ważnym aspektem aplikacji do zarządzania są narzędzia z nimi kompatybilne, to one pozwalają na znaczne ułatwienie pracy. Każda z omawianych aplikacji posiada dużą liczbę takich narzędzi, co również zostało wzięte pod uwagę w badaniu.

Analizując wyniki przeprowadzonego porównania zamieszczone w tabeli 1 łatwo dostrzec, że każda aplikacja mimo tego samego celu i podobnych założeń jest inna. Różnią się one dostępnymi funkcjami, a żadna z omawianych aplikacji nie posiada wszystkich uwzględnionych w porównaniu funkcjonalności. Obala to postawioną tezę, która zakładała, że omawiane aplikacje będą posiadały takie same funkcje.

6. Literatura

[1] Kopczewski M., Alfabet zarządzania projektami, Helion 2012

[2] Koszlajda A., Zarządzanie projektami IT. Przewodnik po metodykach, Helion, 2012

[3] https://pl.wikipedia.org/wiki/Model_kaskadowy, Model kaskadowy [15.06.2018]

[4] https://pl.wikipedia.org/wiki/Model_przyrostowy , Model przyrostowy [15.06.2018]

[5] http://agilemanifesto.org/iso/pl/manifesto.html, Manifest Agille [20.06.2018]

[6] Sutherland J., Scrum. Czyli jak robić dwa razy więcej dwa razy szybciej, Wydawnictwo Naukowe PWN, 2015

[7] Zmitrowicz K., Stańczak R., Jakość w Agile. Zwinna droga do sukcesu, Wydawnictwo Naukowe PWN, 2018

[8] http://it-consulting.pl/autoinstalator/wordpress/2011/03/22/co-wybrac-czyli-cykl-zycia-projektu-tworzenia-oprogramowania/, Co wybrać, czyli cykl życia projektu tworzenia oprogramowania [20.06.2018]

[9] Kaczor K., Scrum i nie tylko. Teoria i praktyka w metodach Agile, Helion, 2015

[10] https://play.google.com/store/apps/details?id=com.asana.app, Asana [23.06.2018]

[11] https://play.google.com/store/apps/details?id=com.basecamp.bc3, Basecamp [23.06.2018]

[12] https://play.google.com/store/apps/details?id=com.monday.monday, Monday [23.06.2018]

[13] https://play.google.com/store/apps/details?id=net.teamworkpm.phone, Teamwork [23.06.2018]

[14] https://play.google.com/store/apps/details?id=com.wrike, Wrike [23.06.2018]

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 8 (2018) 286-291
Wysłane: 2018-06-17
Przyjęte: 2018-06-24

Porównanie wydajności platform integracyjnych

Bartłomiej Karol Flis*, Łukasz Kołyga, Maria Skublewska-Paszkowska
Politechnika Lubelska, Instytut Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. W artykule zostały porównane dwie nowoczesne platformy integracyjne stosowane w środowiskach produkcyjnych, jak i testowych. Jest to rozwiązanie darmowe -szyna danych WSO2 oraz szyna OSB, czyli Oracle Service Bus, która jest rozwiązaniem korporacyjnym. Zostały omówione testy, które były przeprowadzone na wyżej wymienionych szynach oraz uzyskane wyniki. Dodatkowo zostało dołączone porównanie wyników wraz z wnioskami, które zostały wyciągnięte po przeprowadzonych testach.

Słowa kluczowe: OSB; WSO2; Integracja; Enterprise

*Autor do korespondencji. Bartlomiej.Flis1@gmail.com

Comparing the performance of integration platforms

Bartłomiej Karol Flis*, Łukasz Kołyga, Maria Skublewska-Paszkowska
Institute of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract. The article compares two modern integration platforms used in production and test environments. These are, a free integrity solution WSO2 data bus and OSB bus, called Oracle Service Bus, which is a commercial software. The tests that were carried out on the above-mentioned buses and results were discussed. In addition, a comparison of results with conclusions that were drawn after the tests was attached.

Keywords: OSB; WSO2; Integrity; Enterprise

*Corresponding author. Bartlomiej.Flis1@gmail.com

1. Wstęp

Wraz z informatyzacją kolejnych dziedzin życia, powstała potrzeba wymiany informacji pomiędzy kolejnymi firmami, które zdecydowały się na takie kroki. Potrzeba ciągłej integracji skłoniła większość producentów do stworzenia produktów i standardów, które będą umożliwiały wydajną implementacje tych zagadnień.

Pojęcie integracji systemów informatycznych można dostrzec na wielu płaszczyznach życia. Dzięki użyciu platform integracyjnych klienci banków mogą wykonywać dowolną liczbę przelewów w bardzo krótkim czasie. Dzieje się tak dlatego, że instytucje finansowe komunikują się ze sobą i przekazują informacje o wykonanych operacjach. Wraz z rozbudową funkcjonalności, liczba wymienianych komunikatów pomiędzy systemami dziedzinowymi rośnie. Dlatego pojęcie wydajności w zagadnieniu integracji jest tak ważne. Natomiast, zagadnienie integracji może być postrzegane jako komunikacja pomiędzy aplikacjami na poziomie rozumienia przesyłanych bitów danych.

Przez długi czas firma Oracle posiadała jedno z bardziej konkurencyjnych rozwiązań integracyjnych, a jej produkt Oracle Service Bus był stosowany w wielu korporacjach z różnych branż. Rok 2005 był bardzo przełomowy, został wypuszczony produkt o nazwie WSO2[6]. W dość szybkim tempie objął znaczną część rynku prezentując rozwiązanie całkowicie darmowe.

Celem artykułu jest przeprowadzenie analizy wydajnościowej omawianych platform. Wydajność jest rozumiana jako szybkość przepływu informacji w stosunku do pracy, jaką musi wykonać platforma. Badania zostały przeprowadzone w środowisku przygotowanym w oparciu o maszyny wirtualne. Celem pośrednim było zaimplementowanie platformy do testowania wydajności tych dwóch platform.

2. Przygotowanie do przeprowadzenia testów wydajnościowych

W celu przeprowadzenia testów wydajnościowych mających za zadanie zbadać i porównać wydajność platform integracyjnych zostało przygotowane środowisko testowe w oparciu o maszyny wirtualne. W tym przypadku jako hipernadzorca był wykorzystany komputer klasy PC. Na maszynie zostało zainstalowane oprogramowanie VMware ESXi 5.5. Środowisko posiadało następujące zasoby:

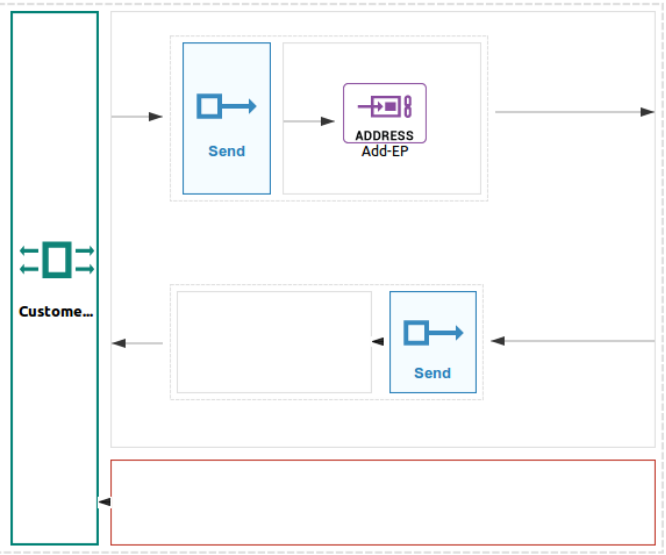
- CPU: Intel(R) Core(TM) i7-4790K CPU @ 4.00 GHz;
- RAM: 2x8GB DDR3 1600 MHz;
- Płyta główna: Asus Z97X Gaming 3;
- HDD: WD Green 1TB 7200 RPM;
- SDD: Samsung EVO 540 120 GB;
- Network: Gigabit Ethernet.

W środowisku testowym zostały utworzone dwie maszyny wirtualne o takich samych parametrach, dla każdej z platform. Trzecia maszyna natomiast została utworzona w celu przeprowadzenia testu obciążeniowego za pomocą oprogramowania SoapUI.

2.1. Scenriusze testowe

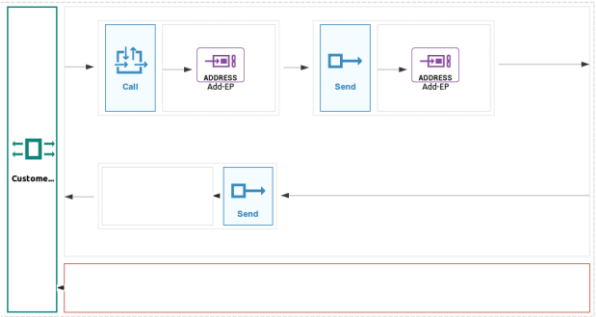
W celu przeprowadzenia miarodajnych testów na każdej z platform zostały zaimplementowane trzy usługi. Każda z usług charakteryzowała się innym przebiegiem wiadomości by sprawdzić zachowanie szyn w różnych warunkach[1]:

- scenariusz usługi prostej – usługa jest odpowiedzialna za komunikacje z systemem dziedzinowym, ciało komunikatu podczas wymiany nie ulega transformacji. Tego typu scenariusz pozwoli określić, jak wydajne są transporty obu rozwiązań. Implementacja pokazana jest na Rys. 1.



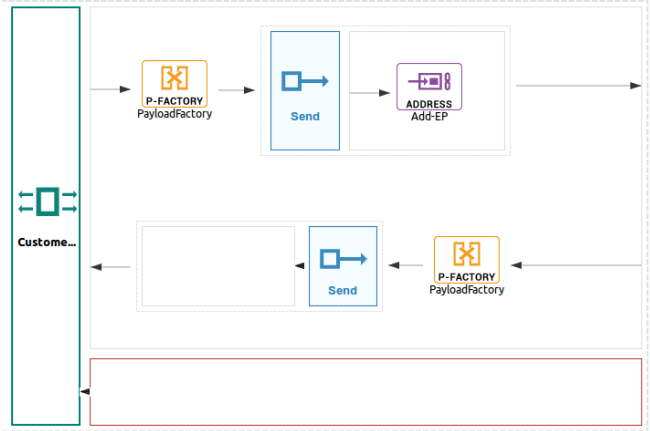
Rys. 1. Usługa prosta

- scenariusz usługi wzbogacania wiadomości – celem usługi jest wzbogacenie wiadomości, która ma trafić do systemu dziedzinowego[2]. Jako przykład przypadku użycia można podać dodawanie punktów lojalnościowych przy wykonywaniu transakcji przez klienta. Schemat usługi można zobaczyć na Rys. 2. Na scenariusz składa się:
  - odbiór komunikatu SOAP;
  - nawiązanie połączenia oraz wysłanie wiadomości do serwisu, który jest odpytywany o dodatkowe informacje;
  - wysłanie komunikatu do systemu dziedzinowego;
  - odbiór komunikatu od systemu dziedzinowego i dostarczenie go do klienta usługi.



Rys. 2. Usługa wzbogacania wiadomości

- scenariusz usługi transformacji (XQuery) – usługa za pomocą której można zmienić strukturę wiadomości XML i dostosować ją do wymagań usługi zewnętrznej. Usługa została przedstawiona na Rys. 3. Struktura scenariusza jest następująca:
  - odebranie komunikatu od klienta SOAP;
  - wywołanie transformacji XQuery komunikatu oraz nawiązanie połączenia z systemem zewnętrznym;
  - wysyłanie i odebranie komunikatów z systemu zewnętrznego;
  - wywołanie odpowiedzi oraz przekazanie wiadomości do kleinta SOAP.



Rys. 2. Usługa transformacji

2.2. Struktura wiadomości

Każda z usług posiada taką samą definicje wiadomości, czyli tak zwany znacznik message (ang. wiadomość), który definiuje z jakich składowych owe wiadomości się składają przy komunikacji SOAP. Na listingu 1 można zauważyć, że oprócz elementów wiadomości definiuje się również ich nazwy do późniejszego korzystania. Można zauważyć również, że definiuje się porttype, który nazywa zestaw komunikatów.

Przykład 1. Zastosowana wiadomość Message

```
<wsdl:message name="PackageRegistrationRequest">
<wsdl:part name="parameters" element="tns:PackageRegistration"/>
</wsdl:message><wsdl:message name="PackageRegistrationResponse">
<wsdl:part name="parameters"
element="tns:PackageRegistrationResponse"/>
</wsdl:message><wsdl:portType
name="CustomerServiceSimplePortType"><wsdl:operation
name="PackageRegistration"><wsdl:input
message="tns:PackageRegistrationRequest"
wsaw:Action="http://www.example.org/PackageService/PackageRegistrati
on"/><wsdl:output message="tns:PackageRegistrationResponse"
wsaw:Action="http://www.example.org/PackageService/PackageService/Pa
ckageRegistrationResponse"/></wsdl:operation></wsdl:portType><¶
```

3. Metodyka przeprowadzenia testów

Przeprowadzenie testów wydajnościowych obejmuje przetestowanie scenariuszy zawartych w punkcie 2.1. Testowanie przebiegło w kolejności sekwencyjnej, czyli jeden po drugim, używając tego samego środowiska. Podczas testów na maszynie nie były prowadzone inne akcje oraz nie były wysyłane wiadomości na inne usługi. Każda z wymienionych usług posiada unikatowy adres, pod którym jest osiągalna.

Testy oraz ich parametry zostały uwarunkowane w sposób następujący:

- czas trwania testu wynosił 5 minut;
- wielkość komunikatu wynosiła 15 KB, 500 KB;
- liczba wątków wysyłających kolejno 3, 15;
- kolejność wysyłania zachowana sekwencyjnie.

Testy wydajnościowe zostały wykonane w projekcie programu utworzonego w oprogramowaniu SoapUI. Wewnątrz projektu zostały umiezczone treści komunikatów, które były wysyłane do usług. Dodatkowo, oprogramowanie umożliwiało wygodne i szybkie sterowanie wartościami testu, takimi jak: czas trwania testu, liczba wątków, czy czas trwania wykonania testu. Do sterowania rozmiarem komunikatu zastosowano wartość pola „City" w dokumencie XML [3]. Zmniejszając i zwiększając długość tekstu, który był w wartości pola, regulowany był rozmiar wysyłanego zapytania.

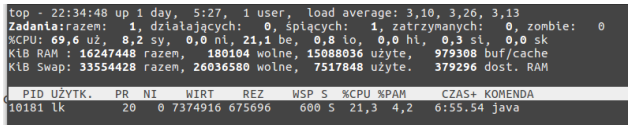
Oracle Service Bus jak i WSO2EI to oprogramowanie oparte o technologie Java, uruchamiane są więc na maszynie wirtualnej Java (JVM – Java Virtual Machine). W związku ze specyfiką maszyny wirtualnej Java[4], przed rozpoczęciem testu, usługi były obciążane przez kilku minut. Zapobiegało to występowaniu anomalii w wynikach testu. Szyny integracyjne to bardzo skomplikowane oprogramowanie i stosuje się w nich wiele technik optymalizacji. Jednym z takich zabiegów może być mechanizm "Codecache" [5] na maszynie JVM. W tym przypadku pewna czynność za pierwszym razem może trwać dłużej, lecz za każdym kolejnym wykonaniem czas wykonania tej czynności powinien być krótszy.

4. Wyniki przeprowadzonych testów wydajnościowych

Po zrealizowaniu testów uzyskane wyniki zostały przedstawione w postaci tabel dla poszczególnych pomiarów i danej szyny. Każda z usług miała cztery pomiary:

- pierwszy pomiar:
  - liczba wątków wysyłających: 3;
  - wielkość komunikatu: 5kb.
- drugi pomiar:
  - liczba wątków wysyłających: 15;
  - wielkość komunikatu: 5kb.
- trzeci pomiar:
  - liczba wątków wysyłających 3;
  - wielkość komunikatu: 500kb.
- czwarty pomiar:
  - liczba wątków wysyłających: 15;
  - wielkość komunikatu: 500kb.

Podczas wykonywania testów sprawdzany był stan pamięci RAM oraz utylizacji procesora. W przypadku WSO2 Enterprise Service Bus poziom zużycia procesora był na poziomie 8% - 14% oraz RAM był na poziomie od 500 mb do 950 mb, co można zobaczyć na Rys 5.



Rys. 5. Pomiar utylizacji CPU oraz RAM

Parametry systemu operacyjnego wahały się w podanych graniach. Dla Oracle Service Bus miały kolejno 10% - 16% utylizacji CPU, zaś RAM był na poziomie 1300 mb – 1500 mb.

Wyniki testów dla szyny WSO2 zaprezentowana w Tabeli 1.

Tabela 1. Wyniki testów wydajnościowych przeprowadzone na szynie WSO2

Usługa prosta				
Liczba wątków wysyłających	Rozmiar	Przepustowość [wiadomości/s]	Średni czas wiadomości [s]	Suma przesłanych wiadomości
3	5kb	654,61	3,98	196383
15	5kb	969,60	14,89	290880
3	500kb	49,68	59,69	14904
15	500kb	75,59	195,56	22677

Tabela 2. Wyniki testów wydajnościowych dla usług wzbogacania na szynie WSO2

Usługa wzbogacania wiadomości				
Liczba wątków wysyłających	Rozmiar	Przepustowość [wiadomości/s]	Średni czas wiadomości [s]	Suma przesłanych wiadomości
3	5kb	360,03	5,33s	108010
15	5kb	620,54	20,39s	186163
3	500kb	34,77	84,16s	10432
15	500kb	49,13	283,56s	14740

Tabela 3. Wyniki testów wydajnościowych dla usługi transformacji na szynie WSO2

Usługa transformacji				
Liczba wątków wysyłających	Rozmiar	Przepustowość [wiadomości/s]	Średni czas wiadomości [s]	Suma przesłanych wiadomości
3	5kb	510,59	5,13s	153178
15	5kb	688,41	18,31s	206524
3	500kb	38,75	68,64s	11625
15	500kb	60,47	228,80s	18141

Dla porównania zostały przedstawione tabele (4-6) z wynikami dla Oracle Service Bus, na której zostały wykonane takie same testy wydajnościowe.

Tabela 4. Wyniki testów wydajnościowych przeprowadzone na szynie Oracle Service Bus

Usługa prosta				
Liczba wątków wysyłających	Rozmiar	Przepustowość [wiadomości/s]	Średni czas wiadomości [s]	Suma przesłanych wiadomości
3	5kb	534,78	4,51s	160434
15	5kb	819,86	17,98s	245958
3	500kb	58,68	39,12s	17604
15	500kb	94,59	183,61s	28377

Tabela 5. Wyniki testów wydajnościowych dla usług wzbogacania na szynie Oracle service Bus

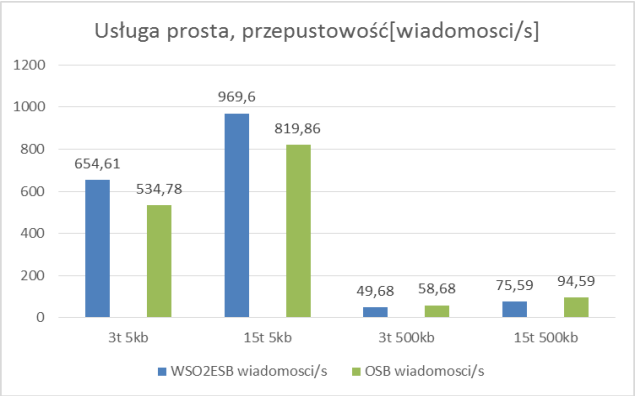
Usługa wzbogacania wiadomości				
Liczba wątków wysyłających	Rozmiar	Przepustowość [wiadomości/s]	Średni czas wiadomości [s]	Suma przesłanych wiadomości
3	5kb	374,34	6,44s	112303
15	5kb	508,31	24,45s	152493
3	500kb	39,31	52,02s	11794

Tabela 6. Wyniki testów wydajnościowych dla usługi transformacji na szynie Oracle Service Bus

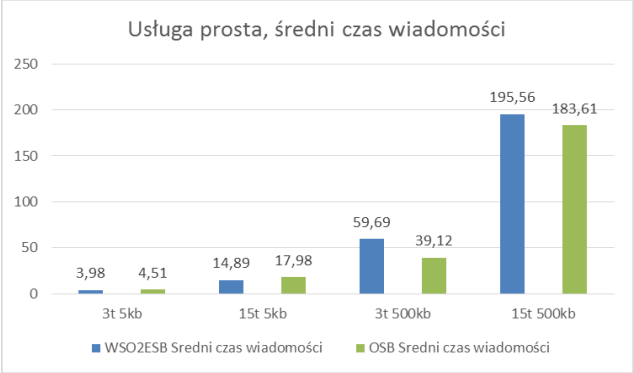
Usługa transformacji				
Liczba wątków wysyłających	Rozmiar	Przepustowość [wiadomości/s]	Średni czas wiadomości	Suma przesłanych wiadomości
3	5kb	390,38	5,41s	117116
15	5kb	590,29	21,93s	177089
3	500kb	48,70	44,98s	14611
15	500kb	69,99	216,65s	20998

5. Analiza otrzymanych wyników

Na podstawie wyników z testów wydajnościowych zostały stworzone wykresy. Na RysRys. przedstawiona jest liczba wysłanych wiadomości na sekundę, dla usługi prostej dla poszczególnej liczby wątków oraz rozmiaru komunikatu. Na tym wykresie można dostrzec, że przepustowość szyny WSO2 dla mniejszych komunikatów jest większa niż dla szyny OSB. Natomiast w przypadku komunikatów o większym rozmiarze, szyna WSO2 radziła sobie gorzej. Na kolejnym Rys. 6 został przedstawiony średni czas przetwarzania komunikatów. Na tym wykresie również widać, że szyna ESB znacząco lepiej radzi sobie z komunikatami o mniejszym rozmiarze, lecz przy obsłudze komunikatów, których rozmiar sięga 500kb czas obsługi komunikatu spada. Dla większych komunikatów oprogramowanie z firmy Oracle wypada lepiej.

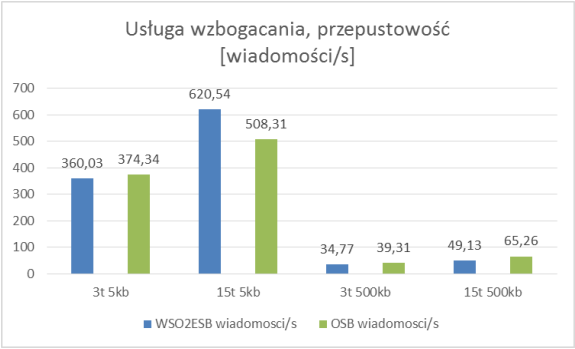


Rys. 6. Liczba przesłanych wiadomości na sekunde dla usługi prostej

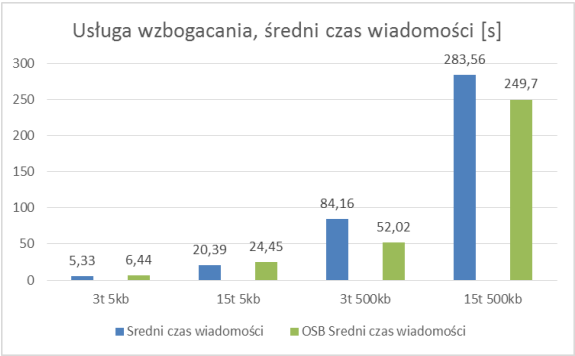


Rys. 7. Średni czas przetwarzania komunikatów w sekundach.

Kolejnym elementem, który był porównywany, była praca usługi wzbogacenia. Wyniki zostały przedstawione na Rys. 7 i Rys 8. W tym przypadku w porównaniu z prostą usługą, była znacznie wolniejsza. Spowodowane jest to specyfiką obsługi takiego wywołania. Na wywołanie takiej usługi składa się wykonanie dwóch zapytań do systemów dziedzinowych. Podczas pierwszego wywołania systemu dziedzinowego wątek, który obsługuje całą usługę jest zablokowany i oczekuje na jego zakończenie. W tym przypadku również szyna Oracle Service Bus lepiej radziła sobie z obsługą większych komunikatów zarówno w przypadku 3 i 15 wątków wysyłających.



Rys. 8. Liczba wiadomości na sekundę dla usługi wzbogacania



Rys. 9. Średni czas przetwarzania komunikatu dla usługi wzbogacania

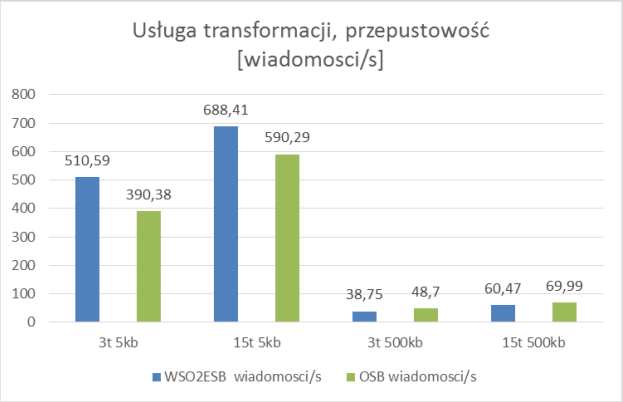
Ostatnią testowaną usługą była ta, której wyniki pokazano na Rys. 10 i Rys. 11. Usługa ta była szybsza od usługi wzbogacenia lecz wolniejsza od usługi prostej. Realizacja tej usługi polega na wywołaniu silnika XQuery wewnątrz szyny. Za wywołanie transformacji odpowiada

osobny wątek. W tym przypadku również widać różnice w prędkościach przetwarzanych komunikatów. W tym przypadku, jak w dwóch wcześniejszych, szyna WSO2 radzi sobie lepiej z komunikatami o mniejszym rozmiarze, natomiast szyna Oracle Service Bus przetwarza z większą wydajnością wiadomości o dłuższej treści.

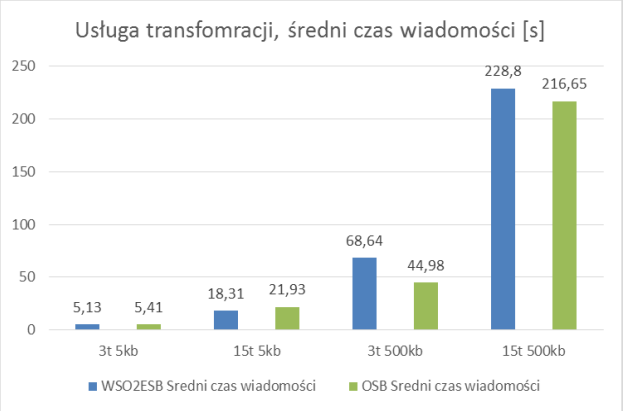
6. Wnioski

Na podstawie implementacji usług, przeprowadzonych testów wydajnościowych i analizy wyników można wyciągnąć następujące wnioski:

- podejście oraz budowa usług na szynie WSO2 Enterprise Integrator oraz Oracle Service Bus są bardzo podobne. Posiadając podstawową wiedzę na temat usług w intuicyjny sposób można poruszać się po obu środowiskach IDE;
- zapotrzebowanie Oracle Service Bus na pamięć RAM jest większa niż w przypadku szyny WSO2 Enterpsie Integrator;
- dla małych komunikatów rozwiązanie oparte o WSO2 Enterprise Integrator jest wydajniejsze niż Oracle Service Bus;
- zwiększenie liczby wątków wysyłających zwiększa liczbę przesłanych komunikatów w ciągu jednostki czasu, lecz wydłuża czas obsługi pojedynczego zapytania;
- w przypadku transformacji komunikatu, wydajność spada w porównaniu usługi prostej, lecz nie jest to znacząca strata wydajności w porównaniu z zyskiem funkcjonalności, którą daje transformacja komunikatu;
- najwolniejszym typem usługi jest usługa wzbogacająca, gdyż wiąże to się z dodatkową alokacją wątku na czas obsługi komunikatu;
- transformacje XQuery jest dostępna zarówno w przypadku WSO2 Enterprise Integrator jak i Oracle Service Bus.



Rys. 10. Liczba wiadomości na sekundę dla usługi transformacji



Rys. 11. Średni czas przetwarzania komunikatu dla usługi transformacji

7. Podsumowanie

Cel pośredni, czyli budowa platformy do testowania oraz cel bezpośredni porównanie wydajności tych dwóch platform zostały zrealizowane. Wyniki testów wydajnościowych dla wszystkich typów wykonywanych testów jednoznacznie wskazują na rozbieżności wydajności dla różnych wielkości komunikatów. Dla większych komunikatów wydajniejszym rozwiązaniem jest szyna Oracle Service Bus, w przypadku komunikatów o mniejszym rozmiarze większą przepustowością charakteryzowała się szyna WSO2 Enterprise Integrator.

Nie można jednoznacznie określić, która platforma integracyjna jest wydajniejsza, wybór rozwiązania zależy od kontekstu wykorzystania produktu oraz ekosystemu oprogramowania, jakie zastosowane jest w przedsiębiorstwie. Jeżeli w przedsiębiorstwie używane jest inne oprogramowanie Oracle np. Oracle Database lub Oracle Message Broker, w takim przypadku lepszym wyborem może okazać się OSB. Powodem tego jest bardzo dobra i wspierana integracja pomiędzy Oracle Service Bus, a brokerem lub bazą danych. Przykładem takiej zalety może być integracja z bazą danych, która umożliwi używanie kolejek bazodanowych na szynie. Ponadto posiadając wiele produktów Oracle można liczyć na obniżenie sumarycznej ceny za licencje. Szyna WSO2 Enterprise Integrator w porównaniu z Oracle Service Bus jest młodszym produktem. Jest dystrybuowany na licencji Open Source, dzięki czemu można jej używać bez ponoszenia dodatkowych kosztów licencyjnych. W przypadku, kiedy przedsiębiorstwo chciałoby korzystać ze wsparcia podczas implementacji lub utrzymania ma możliwość wykupienia oferowanego płatnego wsparcia firmy WSO2.

Oba rozwiązania świetnie sobie radzą z przetwarzaniem równoległym wielu wiadomości jednocześnie. Dzięki temu oba rozwiązania są w stanie obsługiwać jednocześnie bardzo wiele klientów korzystających z szyny. Ponadto dla zwiększenia wydajności jest możliwość zbudowania klastra, który zwiększy potencjał oraz możliwości platformy.

Dodatkowym plusem budowy klastra jest wysoka dostępność produkty. W przypadku, kiedy jedna z instancji zostanie uszkodzona, reszta klastra przejmuje jego role.

Literatura

[1] J. Duckett, Beginning HTML, XHTML, CSS and JavaScript, Wiley Publishing Inc., New York (2010), 507 - 510

[2] M.. Havey, SOA Cookbook, Packt Publishing, New York(2008) 89-90,

[3] XML, https://en.wikipedia.org/wiki/XML, [01.06.2018]

[4] J. Duckett, Beginning HTML, XHTML, CSS and JavaScript, Wiley Publishing Inc., New York (2010), 507 - 510

[5] Oracle Java SE Embedded: Developer's Guide, https://docs.oracle.com/javase/8/embedded/develop-apps-platforms/codecache.htm,: [01.06.2018r]

[6] D. Chappell, Enterprise Service Bus: Theory in Practise, O'Reilly Media, New York(2010) 112 - 114

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 8 (2018) 292-297
Wysłane: 2018-06-17
Przyjęte: 2018-06-27

Analiza właściwości metod steganografii odwracalnej

Piotr Zimnicki*, Grzegorz Kozieł
Politechnika Lubelska, Instytut Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. Steganografia, czyli nauka zajmująca się ukrywaniem informacji w informacji, zyskuje na popularności wraz ze wzrostem znaczenia internetu. Mwtody steganografii odwracalnej, będącej jej częścią, umożliwiają bezstratne odzyskanie zarówno ukrytej wiadomości jak i oryginału jej nośnika. Mają one jednak różne zastosowania i możliwości. Niniejszy artykuł poświęcony jest teoretycznej i praktycznej analizie różnych metod steganografii odwracalnej opracowanych przez badaczy.

Słowa kluczowe: steganografia odwracalna; grafika; dziedzina przestrzenna

*Autor do korespondencji.

Adres e-mail: zimnickipm@gmail.com

Analysis of properties of reversible steganography methods

Piotr Zimnicki*, Grzegorz Kozieł
Institute of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract. Steganography is a science that deals with hiding information in other information, and it's becoming increasingly popular with rapid growth of Internet utilization. Reversible steganography methods allow lossless recovery of not only the message, but the cover image as well. Different methods have different characteristics, and thus different applications. This paper aims to deepen the understanding of selected reversible methods, their strong and weak point, through both theoretical and practical analysis. The comparison of chosen methods is presented in the paper.

Keywords: reversible steganography; graphics; spatial domain

*Corresponding author.

E-mail address: zimnickipm@gmail.com

1. Wstęp

Wraz z postępem informatyzacji, zagadnienie ochrony danych staje się coraz istotniejsze. Choć korzenie steganografii sięgają znacznie głębiej, to jest ona jedną z odpowiedzi na ten problem. Z uwagi na ogromną popularność plików graficznych, metody steganograficzne wykorzystujące obrazy cyfrowe są zdecydowanie najpopularniejsze. Metody steganografii odwracalnej stanowią jedynie część wszystkich metod steganograficznych. Ich cechą charakterystyczną jest możliwość bezstratnego odzyskania nie tylko ukrytej wiadomości, ale również pliku, w którym była ona ukryta. Znajdują one zastosowanie w przypadku danych medycznych, dowodów sądowych, jako kanał ukrytej komunikacji lub jako jedna z metod uwierzytelnienia.

Mnogość potencjalnych zastosowań przyczyniła się, i przyczynia się nadal, do powstania znacznej ilości różnych metod steganografii odwracalnej. Niniejszy artykuł poświęcony jest teoretycznej i praktycznej analizie części tych metod. Ma on na celu umożliwienie ich lepszego zrozumienia, oraz poznania słabych i mocnych stron.

2. Dotychczasowe badania

Steganografia jest nauką stale rozwijaną przez badaczy z całego świata. Każdego roku publikowane są artykuły naukowe, opisujące nowe, bądź udoskonalone, metody ukrywania informacji. Trudno jednak odnaleźć publikację zestawiającąze sobą takie metody. Jedną z nielicznych jest praca Yun Q. Shi zatytuowana *Reversible Data Hiding* [1]. Autor dokonuje podziału metod steganograficznych ze względu na zastosowanie, przy czym wyróżnia trzy kategorie:

a) Uwierzytelnianie wrażliwe, gdzie nawet najmniejsza modyfikacja obrazu jest niedopuszczalna
b) Uwierzytelnianie mniej wrażliwe, enigmatyczne sformułowanie oznaczające uwierzytelnianie, gdzie pewne modyfikacje obrazu, jak na przykład kompresja, są akceptowalne
c) Wymagające wysokiej pojemności informacyjnej, to znaczy umożliwiające ukrycie dużych ilości informacji w obrazie

Tego typu podział jest oczywiście merytorycznie poprawny, jednak nie jest całkowicie wyczerpujący. Autor porusza tutaj temat metod steganograficznych działających na obrazach skompresowanych, co prowadzi nas do innego, bardziej technicznego podziału metod steganografii odwracalnej:

a) Operujące w dziedzinie przestrzennej (*ang. spatial domain*), ukrywające sekretne informacje bezpośrednio w obrazie. Jest to najprostsza, a zarazem najczęściej spotykana technika.

b) Operujące w dziedzinie transformacji (*ang. transform domain*), ukrywające treść wiadomości w częstotliwościach występujących w obrazie. W przypadku obrazów poddanych kompresji JPEG większość metod steganograficznych opiera się na manipulowaniu wartościami współczynników dyskretnej transformaty kosinusowej nośnika.

Interesującym przykładem metod operujących w dziedzinie przestrzennej, są metody oparte o modyfikację histogramu. Na podstawie obrazu-nośnika budowany jest histogram wartości pikseli, w którym to właśnie ukrywana jest wiadomość steganograficzna. Sposób ukrycia wiadomości zależny jest od metody – może opierać się o wartości ekstremalne [2] czy zmiany wartości sąsiadujących pikseli [3]. Jeśli natomiast chodzi o metody operujące w dziedzinie transformacji, to warta wspomnienia jest metoda opracowana przez badaczy z Tajwanu [4]. Używa ona transformaty falkowej do dekompozycji obrazu na poszczególne pasma, ukrywając sekretną wiadomość zarówno w wysokich jak i niskich częstotliwościach. Umożliwia ukrycie dużych ilości informacji i jest wysoce odporna na kompresję, jednak jej słabą stroną jest podatność na zmiany w dziedzinie przestrzennej - drobne manipulacje wartościami pikseli mogą uniemożliwić odczyt wiadomości, jak również odzyskanie oryginalnego medium. Studiując literaturę tematyczną można natknąć się również na inne, bardziej egzotyczne kryteria klasyfikacji metod steganograficznych [5][6]:

a) Techniki rozproszonego widma (*ang. spread spectrum*), stosujące techniki zapożyczone z szerokopasmowych systemów komunikacji. Sekretna wiadomość jest tu kodowana w szerokim spektrum częstotliwości, znacznie utrudniając jej wykrycie, lecz także odzyskanie oryginalnego obrazu.
b) Techniki statystyczne, kodujące informacje poprzez zmianę właściwości statystycznych obrazu, a wykorzystujące proces weryfikacji hipotez do ich odczytania.
c) Techniki oparte o zakłócenia, wstrzykujące w nośnik pewne zniekształcenia, których późniejsze wykrycie i pomiar umożliwia odczytanie wiadomości.

W pewnych sytuacjach bezstratne odzyskanie całego obrazu-nośnika może nie być wymagane. W takich przypadkach pomocne są metody oparte o tak zwany obszar zainteresowania (*ang. region of interest*). Za ich pomocą można odzyskać tylko ten obszar obrazu, który został odpowiednio oznaczony na etapie kodowania. Pozostała część może zostać nieodwracalnie zniekształcona, o czym należy pamiętać używając takich metod. Znajdują one zastosowanie m.in. w obrazach medycznych - z uwagi na charakter takich obrazów oraz wysoką pojemność informacyjną, oferowaną przez te metody [7].

Użycie metod steganograficznych nie oznacza jednak konieczności rezygnacji z innego sposobu zabezpieczania informacji – kryptografii. Metody steganograficzne w żaden sposób nie konfliktują z metodami kryptograficznymi. Część z nich stosuje wręcz szyfrowanie obrazu jako jeden z kroków działania, jeszcze mocniej zabezpieczając ukrytą informację.

3. Algorytmy metod steganografii odwracalnej

Badaniom porównawczym poddane zostały trzy metody, wybrane na podstawie analizy literatury tematycznej. Są one mocno zróżnicowane, dzięki czemu otrzymane wyniki powinny znacznie różnić się od siebie, uwypuklając mocne i słabe strony tych metod.

3.1. Rozszerzanie różnic

Metoda ta została opracowana z myślą o umożliwieniu ukrycia dużych ilości informacji w obrazie[8]. Opiera się ona o pary pikseli *(x, y)*, ich wartość średnią l oraz różnicę ich wartości h.

$$\begin{aligned} l &= \lfloor 0{,}5 \cdot (x+y) \rfloor \\ h &= x - y \\ x &= l + \lfloor 0{,}5 \cdot (h+1) \rfloor \\ y &= l - \lfloor 0{,}5 \cdot h \rfloor \end{aligned} \tag{1}$$

Wartość h przesuwana jest o jeden bit w lewo (mnożona przez dwa), a najmniej znaczący bit nowej wartości używany jest do ukrycia kolejnego bitu sekretnej wiadomości. Ostatecznie, nowe wartości pikseli obliczane są przy użyciu rozszerzonej różnicy oraz (niezmienionej) wartości średniej. Rozwiązaniem problemu przepełnienia jest w tym wypadku ukrywanie danych tylko w takich parach pikseli, których zmiana nie spowoduje przepełnienia wartości. Macierz wartości binarnych, kodujących informację o ukryciu bitu wiadomości w danej parze, poddawana jest kompresji bezstratnej i ukrywana wraz z właściwą treścią wiadomości. Przykładowy obraz poddany działaniu metody, oraz błędy odczytu po etapie kodowania (t.j. pary ukrywające wiadomość) przedstawia rysunek 1. Sposób doboru par jest właściwie dowolny, o ile jest powtarzalny. Podczas badań pary tworzone będą z następujących po sobie pikseli.

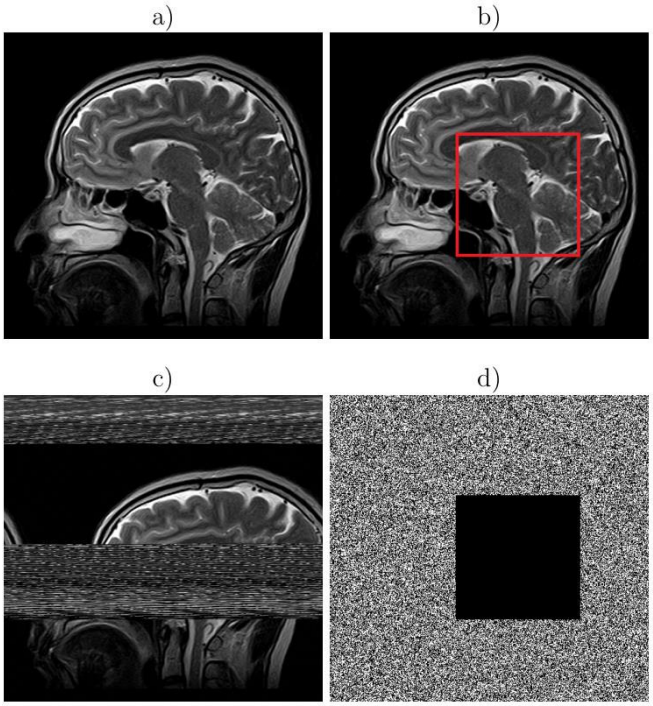
Rys 1 Obraz *clown* oraz błędy odczytu obrazu wynikowego względem oryginalnego po etapie ukrycia wiadomości

Odwracalność modyfikacji obrazu stosowanych w metodzie nie jest na pierwszy rzut oka oczywista – funkcja *podłoga* oznacza bowiem bezpowrotną utratę informacji o części dziesiętnej poddawanych niej liczb. Problem ten występuje jednak tylko pozornie. Zgodnie z falką Haara, działanie to jest odwracalne w obrębie takich par liczb całkowitych [8]. Odwracalność ta, eliminująca błąd zaokrąglenia, wraz z wspomnianą wcześniej macierzą kodującą informację o ukryciu wiadomości gwarantuje odwracalność metody.

3.2. Obszar zainteresowania

Metoda ta została opracowana z myślą o ukrywaniu danych pacjentów w obrazach stworzonych za pomocą rezonansu magnetycznego lub tomografii komputerowej [7]. Nie oznacza to jednak, że jest to jej jedyne możliwe zastosowanie. Fragment obrazu jest tutaj oznaczany jako ROI (*ang. region of interest, obszar zainteresowania*) i tylko tą część będzie można bezstratnie odzyskać. Pozostała część obrazu (*RONI, ang. region of non-interest, obszar braku zainteresowania*) posłuży do ukrycia zakodowanej wiadomości za pomocą algorytmu GLSB [9]. Warto wspomnieć o specjalnym wycinku obszaru RONI, oznaczonym jako krawędź (*ang. border*), znajdującym się na końcu pliku i posiadającym z góry określony rozmiar. Obszar ten zostanie użyty do ukrycia informacji pozwalających na odczytanie zakodowanej wiadomości. Wizualizacja działania algorytmu przedstawiona jest na rysunku 2.

Interesujący jest tu również fakt, że obraz może zostać zaszyfrowany dowolnym kluczem kryptograficznym w sposób niezależny i nie wpływający na możliwość ukrycia czy odczytu wiadomości steganograficznej. Klucz steganograficzny może zostać użyty do odczytania danych steganograficznych bez znajomości klucza kryptograficznego i *vice versa*.



Rys. 2. Wizualizacja algorytmu metody obszaru zainteresowania *a) Przykładowy obraz b) Obraz z wyznaczonym ROI c) Obraz zawierający ukrytą wiadomość d) Błędy odczytu*

Pierwszym krokiem algorytmu jest określenie obszarów ROI, RONI oraz krawędzi. Należy również wyznaczyć sumę kontrolną ROI, umożliwiającą późniejszą weryfikację poprawności odczytanych danych. Po podziale obrazu następuje jego reorganizacja. Dane dotyczące ROI są usuwane ze swojego oryginalnego miejsca w obrazie i umieszczane na samym początku pliku. Tak przygotowany obraz jest szyfrowany z użyciem szyfru strumieniowego, na przykład za pomocą operacji XOR.

Zaszyfrowany obraz może już posłużyć do ukrycia wiadomości steganograficznej za pomocą metody GLSB [9]. Jest to metoda stratna, ponieważ jednek dane ukryte zostaną jedynie w obszarze RONI - ROI nie zostanie poddane modyfikacji, a więc obszar ten można będzie w całości odzyskać. Wspomniany wcześniej obszar krawędzi nie jest używany do ukrycia treści wiadomości, a do przechowania informacji o położeniu i rozmiarze ROI oraz jego sumie kontrolnej.

3.3. Modyfikacja histogramu

Jest to jedna z pierwszych metod opierających się na histogramie obrazu [2]. Jest kompatybilna z wieloma formatami i typami plików oraz zapewnia dobrą jakość obrazu wynikowego. Umożliwia jednak ukrycie jedynie stosunkowo niewielkich ilości danych.

Warto zaznaczyć, że wszelkie operacje dotyczące modyfikacji wartości histogramu przeprowadzane są bezpośrednio na obrazie. Histogram jest jedynie cechą charakterystyczną obrazu, więc niemożliwa jest jego bezpośrednia modyfikacja. Przykładowo, operacja przesunięcia histogramu w prawo o jedną jednostkę, oznacza zwiększenie wartości każdego piksela obrazu o jeden – po wykonaniu tej operacji histogram obrazu wynikowego będzie przesunięty w prawo, zgodnie z oczekiwaniami.

Rys. 3 Wizualizacja działania metody opartej o modyfikację histogramu

Działanie algorytmu rozpoczyna stworzenie histogramu wartości pikseli obrazu wejściowego, oraz znalezienie występujących w nim ekstremów – minimum i maksimum. W przypadku obrazu *Lena*, widocznego powyżej, wartości te są równe odpowiednio $h(0) = 0$ oraz $h(154) = 2723$. Kolejnym krokiem jest przesunięcie fragmentu histogramu należącego do przedziału (0, 154) w lewo o jedną jednostkę. Spowoduje to powstanie pustej kolumny dla wartości 153, sąsiadującej z wartością maksymalną. Wiadomość steganograficzna może teraz zostać ukryta za pomocą tych

dwóch wartości – każdy kolejny piksel przyjmujący wartość 154 jest zestawiany z kolejnym bitem sekretnej wiadomości, i jeśli wartość bitu równa się jeden wartość piksela zmniejszana jest o jeden.

W przypadku, gdy wartość minimalna jest różna od zera należy zachować współrzędne pikseli osiągających wartość minimalną. Dane te należy później skompresować i dołączyć do ukrytej informacji jako narzut.

Odczyt ukrytej informacji odbywa się w sposób analogiczny, odwracając kierunek i kolejność kroków algorytmu. Aby jednak odczyt jakichkolwiek danych był możliwy adresat musi posiadać odpowiedni klucz steganograficzny. W przypadku tej metody składa się on z informacji o histogramie oryginalnego obrazu, a mianowicie wartości minimalnej i maksymalnej. Rysunek 3 przedstawia wizualizację działania tej metody.

4. Specyfika badań

Ponieważ treść wiadomości ukrywanej w obrazach jest bez znaczenia z punktu widzenia metody będącej obiektem analizy, wiadomość generowana będzie w sposób losowy. Weryfikacji poddana zostanie jedynie integralność danych odzyskanych bezpośrednio z obrazu. Kolejnym czynnikiem wymagającym uwagi jest sam obraz, służący za nośnik treści. Oczywiste jest, że większy obraz może umożliwić ukrycie większej ilości informacji. Aby wyniki były możliwie najbardziej miarodajne, wszystkie obrazy użyte do badań będą miały stałe wymiary, wynoszące 512x512 pikseli. Wszystkie będą również monochromatyczne, co oznacza, że na jeden piksel obrazu przypadać będzie zawsze dokładnie 8 bitów pamięci. Zawartość obrazu może również mieć znaczący wpływ na działanie algorytmów. Aby zniwelować potencjalne oddziaływanie tego czynnika próby prowadzone będą w dwojaki sposób: na pojedynczych, wybranych obrazach, oraz na zbiorze 109 obrazów, zgromadzonych specjalnie w tym celu.

Do badania wpływu metody na jakość obrazu wynikowego posłuży następujący zestaw miar:

- Błąd średniokwadratowy, wyrażony wzorem

$$MSE = \frac{1}{M \cdot N} \cdot \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} \left[x_{i,j} - x'_{i,j}\right]^2 \quad (2)$$

- Znormalizowany błąd średniokwadratowy, wyrażony wzorem

$$NMSE = \frac{\sum_{i=0}^{N-1} \sum_{j=0}^{M-1} \left[x_{i,j} - x'_{i,j}\right]^2}{\sum_{i=0}^{N-1} \sum_{j=0}^{M-1} \left[x_{i,j}\right]^2} \quad (3)$$

- Szczytowy stosunej sygnału do szumu, wyrażony wzorem

$$PSNR = 10 \cdot \log_{10} \frac{255^2}{MSE} \quad (4)$$

- Ilość bitów wiadomości na jeden piksel obrazu, wyrażona wzorem

$$\text{bpp} = \frac{\text{Pojemność-Narzut}}{\text{N·M}} \quad (5)$$

Gdzie N oraz M oznaczają wymiary obrazu, natomiast $x_{i,j}$ oraz $x'_{i,j}$ oznaczają odpowiednio kolejne piksele obrazu wejściowego oraz wynikowego. *Pojemnością* z kolei nazywamy całkowitą ilość bitów, które można zakodować w obrazie, a *narzutem* jest ilość bitów które musimy dodatkowo ukryć, aby możliwe było poprawne odzyskanie danych i/lub obrazu. W pewnych sytuacjach może wystąpić sytuacja, że *Narzut* ≥ *Pojemność* – przypadki takie będą pomijane, gdyż nie będzie możliwe zastosowanie danej metody. Współczynnik *bpp* wynosi tutaj zero. Dodatkowej ocenie poddana zostanie również wizualna jakość obrazu. Będzie ona całkowicie subiektywna i nie podlegająca żadnej standaryzacji, a jednak niezbędna.

5. Wyniki

Tabela 1. Średnie wyniki pomiarów po zastosowaniu metody rozszerzania różnic dla dużej próby

Mierzony parametr	Wartość
MSE	41,0115
NMSE	0,288%
PSNR	33,825
Pojemność	124939
Narzut	12000
Bpp	0,43083

Metoda oparta o rozszerzanie różnic umożliwia ukrycie znaczących ilości danych, osiągając średni poziom *bpp = 0,43083*, co jest wynikiem bardzo dobrym. Niestety, okazała się też najgorsza pod względem jakości obrazu. Jakość jest tu bowiem mocno powiązana z charakterystyką danego nośnika informacji. Rozszerzanie różnic może wprowadzić znaczne zakłócenia w przypadku par o dużym gradiencie wartości pikseli. Problem ten pojawia się zwłaszcza w obrazach, zawierających zmiany wysokiej częstotliwości. Rozwiązaniem pozwalającym na jego zniwelowanie może być inny sposób tworzenia par, oparty na przykład o generator liczb pseudolosowych zainicjowany znanym ziarnem,. Każdy przypadek należy jednak rozpatrywać indywidualnie. Wizualna jakość obrazu jest dobra, choć różnice między obrazem początkowym a steganograficznym są czasami zauważalne. Wyniki pomiarów dla tej metody przedstawia tabela 1.

Tabela 2. Średnie wyniki pomiarów po zastosowaniu metody modyfikacji histogramu dla dużej próby

Mierzony parametr	Wartość
MSE	0,5105
NMSE	0,006%
PSNR	52,258
Pojemność	16011
Narzut	223
Bpp	0,06023

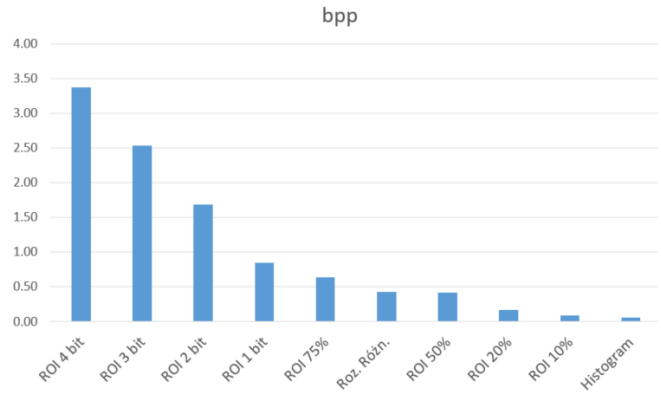
Metoda modyfikacji histogramu, której wyniki przedstawia tabela 2, osiągnęła najmniejszą pojemność spośród badanych metod, jednak jest to zgodne z oczekiwaniami. Powstała ona z myślą o ukrywaniu niewielkich ilości informacji, zachowując przy tym wysoką jakość obrazu wynikowego. Rzeczywiście, metoda ta zapewnia najlepszą spośród badanych metod jakość obrazu, w tym również wizualną. Jakość ta jest jednak zależna od

charakterystyki obrazu, t.j. od odległości dzielącej minimalną oraz maksymalną wartość histogramu. Minimalizacja tej odległości, o ile w danym przypadku jest to możliwe, zmniejszy ilość zakłóceń wprowadzanych przez algorytm, co zaowocuje poprawą jakości. Problem niskiej pojemności zapewnianej przez metodę można częściowo zniwelować poprzez wielokrotne kodowanie informacji w tym samym obrazie. Każda kolejna iteracja zwiększy długość ukrytej wiadomości, jednak za cenę jakości obrazu.

Tabela 3. Średnie wyniki pomiarów po zastosowaniu metody ROI dla dużej próby

Mierzony parametr	Wartość
MSE	0,4221
NMSE	0,004 %
PSNR	51,876
Pojemność	221120
Narzut	0
Bpp	0,84351

Metoda oparta o obszar zainteresowania osiąga niemal tak wysoką jakość, co metoda modyfikacji histogramu, zapewniając jednocześnie znacznie wyższą pojemność. Ceną za to jest brak całkowitej odwracalności – odzyskać można tu bowiem jedynie wskazany uprzednio fragment nośnika.

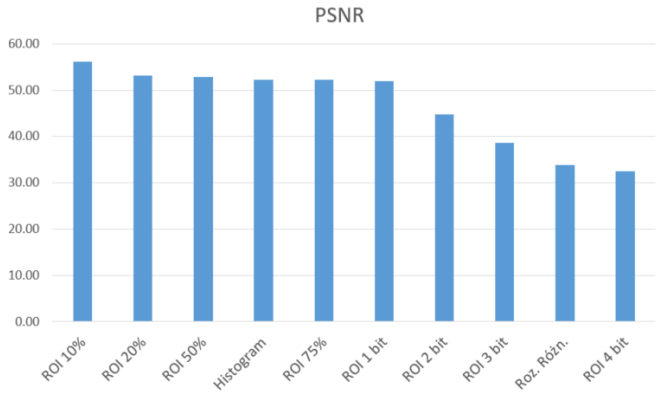


Rys. 4 Średnie wyniki *bpp* osiągane przez poszczególne metody i ich warianty

Metoda okazała się całkowicie niewrażliwa na obraz i jego cechy charakterystyczne. Dodatkowo, użycie krótszej niż jest to możliwe wiadomości (t.j. niewykorzystanie całego dostępnego miejsca) pozytywnie wpłynie na jakość. Dzięki zmianie liczby najmniej znaczących bitów odpowiedzialnych za zapis informacji przez algorytm GLSB, o który opiera się metoda obszaru zainteresowania, można umożliwić ukrycie jeszcze dłuższej wiadomości steganograficznej. Każdy kolejny bit użyty do zapisu informacji spowoduje liniowy wzrost pojemności. Dla badanych obrazów, zapisujących wartość pikseli za pomocą 8 bitów, użycie dwóch najmłodszych bitów nie wpłynie znacząco na pogorszenie jakości obrazu wynikowego. Użycie każdego kolejnego bitu znacząco wpłynie na jakość, przy czym zastosowanie więcej niż czterech okazuje się wysoce niepraktyczne, powodując zbyt duże zniekształcenia.

Zestawienie średnich wyników pomiarów dla poszczególnych metod i ich wariantów przedstawia tablea 4.

Jeśli chodzi o maksymalną długość wiadomości, przedstawioną na rysunku 4, metoda oparta o obszar zainteresowania okazała się bezkonkurencyjna. Co więcej, umożliwia ona dalsze zwiększenie pojemności poprzez zwiększenie liczby bitów, użytych do ukrycia wiadomości steganograficznej. Najgorsza okazała się metoda modyfikacji histogramu, przy czym jest to wynik zgodny z oczekiwaniami.



Rys. 5. Średnie wyniki *PSNR* osiągane przez poszczególne metody i ich warianty

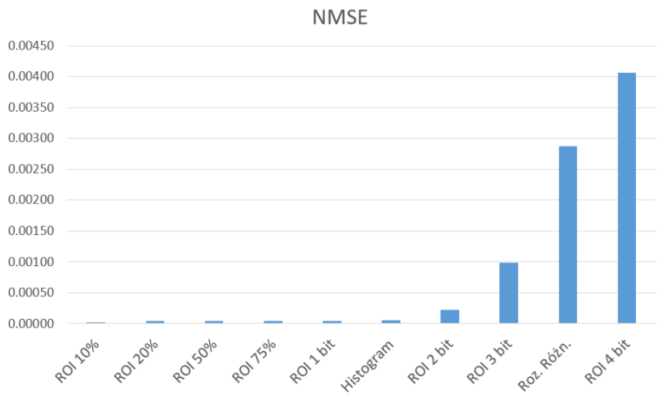
Tabela 4. Średnie wartości pomiarów dla poszczególnych metod i ich wariantów

Metoda	bpp	PSNR	NMSE
Roz. Różn.	0,43083	33,825	0,288%
Histogram	0,06023	52,258	0,006%
ROI 10%	0,08435	56,258	0,002%
ROI 20%	0,16870	53,107	0,004%
ROI 50%	0,42175	52,878	0,004%
ROI 75%	0,63263	52,247	0,004%
ROI 1 bit	0,84351	51,876	0,004%
ROI 2 bit	1,68701	44,831	0,023%
ROI 3 bit	2,53052	38,606	0,098%
ROI 4 bit	3,37402	32,508	0,406%

Szczytowy stosunek sygnału do szumu, przedstawiony na rysunku 5, okazał się najwyższy dla metody modyfikacji histogramu. Jednak w przypadku, gdy za pomocą metody obszaru zainteresowania ukryjemy wiadomość krótszą niż jest to możliwe, wykorzystując jedynie część dostępnego miejsca, możliwe jest osiągnięcie lepszych rezultatów. Przyjmując wykorzystanie dostępnego miejsca na poziomie 10%, co daje średni poziom *bpp* zbliżony do wyniku metody modyfikacji histogramu, metoda ROI osiąga lepszą jakość. Metoda oparta o rozszerzanie różnic osiągnęła tu najgorsze wyniki.

W przypadku znormalizowanego błędu średniokwadratowego, przedstawionego na rysunku 6, metoda modyfikacji histogramu oraz metoda obszaru zainteresowania osiągnęły bardzo zbliżone rezultaty. Wynika to bezpośrednio ze sposobu ich działania – wartość dowolnego piksela nośnika jest tu bowiem modyfikowana co najwyżej o 1. Użycie więcej niż jednego najmniej znaczącego bitu przez metodę ROI drastycznie zwiększa poziom błędu średniokwadratowego. Ponownie, najgorsza okazała się metoda rozszerzania różnic - może ona bowiem wprowadzać

duże zakłócenia w obrębie dobranych par, co znacznie zwiększa błąd.



Rys. 6. Średnie wyniki *NMSE* osiągane przez poszczególne metody i ich warianty

6. Podsumowanie

Wszystkie trzy wybrane metody udało się skutecznie zaimplementować. Zgodnie z oczekiwaniami każda z nich umożliwia ukrycie sekretnej wiadomości steganograficznej we wskazanym obrazie, oraz późniejszy bezbłędny odczyt zarówno wiadomości jak i jej nośnika – zgodnie z charakterystyką metody. Otrzymane wyniki różnią się od siebie nie tylko pomiędzy metodami, ale nawet w obrębie jednej metody. Okazuje się, że poszczególne metody są w różnym stopniu wrażliwe na cechy charakterystyczne samego obrazu, będącego nośnikiem informacji.

Na podstawie przeprowadzonych badań można wnioskować, że metoda oparta o obszar zainteresowania umożliwia ukrycie największej ilości informacji. Jeśli chodzi o jakość obrazu, metoda modyfikacji histogramu uzyskała najlepsze wyniki, przy czym metoda ROI nie pozostaje daleko w tyle. Wizualna jakość obrazów otrzymanych za pomocą wszystkich trzech metod jest bardzo dobra.

Metoda oparta o obszar zainteresowania okazała się najciekawsza pod względem badawczym, umożliwiając uzyskanie drastycznie różnych wyników poprzez modyfikację odpowiednich parametrów. Użycie więcej niż jednego bitu do zapisu treści wiadomości steganograficznej skutkuje liniowym wzrostem pojemności informacyjnej.

Metoda oparta o rozszerzanie różnic okazała się niezwykle wrażliwa na pewne cechy charakterystyczne obrazu, a mianowicie na zmiany wysokiej częstotliwości. Problem ten można jednak zniwelować stosując alternatywny sposób tworzenia par – na przykład w oparciu o algorytm generujący ciąg liczb pseudolosowych. Występuje on jednak stosunowo rzadko, a każdy taki przypadek wymaga indywidualnego podejścia.

Literatura

[1] Y. Shi. Reversible data hiding, *Department of Electrical and Computer Engineering, New Jersey Institute of Technology*, 2005.

[2] Zhicheng Ni, Yun-Qing Shi, Nirwan Ansari. Reversible data hiding, *IEEE Transaction on Circuits and Systems for Video Technology*, 16, 2006.

[3] Chia-Chen Lin, Wei-Liang Tai, Chin-Chen Chang. Multilevel reversible data hiding based on histogram modification of difference images, *Pattern Recognition*, 41, 2008.

[4] Ching-yu Yang, Chih-Hung Lin, Wu-Chih Hu. Reversible data hiding for high quality images based on integer wavelet transform, *Journal of Information Hiding and Multimedia Signal Processing*, 2, 2012.

[5] C.P. Sumathi, T. Santanam, G. Umamaheswari. A study of various steganographic techniques used for information hiding, *International Journal of Computer Science and Engineering Survey*, 4, 2013.

[6] Divya .A, S. Thenmozhi. Steganography: Various techniques in spatial and transform domain, *International Journal of Advanced Scientific Research and Management*, 1, 2016.

[7] Yuling Liu, Xinxin Qu, Goujiang Xin. A ROI-based reversible data hiding scheme in encrypted medical images, *Journal of Visual Communication and Image Representation*, 39, 2016

[8] J. Tian. Reversible data embedding using a difference expansion, *IEEE Transaction on Circuits and Systems for Video Technology*, 13, 2003

[9] Mehmet U. Celik, Gaurav Sharma, A. Murat Tekalp. Reversible data hiding, *International Conference on Image Processing*, 2, 2002

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 8 (2018) 298-301
Wysłane: 2018-06-20
Przyjęte: 2018-06-24

Porównanie wydajności testów automatycznych napisanych w technologii SeleniumWebDriver i HP UFT

Krzysztof Drążek*, Maria Skublewska-Paszkowska
Politechnika Lubelska, Instytut Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. Testy automatyczne stosowane są do weryfikowania funkcjonalności tworzonego oprogramowania. W artykule porównana została szybkość wykonywania automatycznych testów napisanych w popularnych technologiach: HP UFT i WebDriver Selenium. Dla każdej z nich powstały po 3 testy, wykonujące te same czynności. Na podstawie otrzymanych wyników zostały wyciągnięte wnioski dotyczące tego, która technologia jest lepszym wyborem, gdy liczy się wydajność wykonywanych testów.

Słowa kluczowe: testy automatyczne; testowanie; Selenium; UFT

*Autor do korespondencji.

Adres e-mail: krzydrazek@gmail.com

Performance comparison of automatic tests written in Selenium WebDriver and HP UFT

Krzysztof Drążek*, Maria Skublewska-Paszkowska
Institute of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract. Automation tests are used for verification of functionality of developing applications. The performance of automation tests written in HP UFT and Selenium WebDriver was compared. There were created 3 tests for both technologies that were executing the same tasks. According to the obtained results, it was concluded which technology is a better choice if the time efficiency of tests execution is important.

Keywords: automation tests; testing; Selenium; UFT

*Corresponding author.

E-mail address: krzydrazek@gmail.com

1. Wstęp

Testy automatyczne są jedną z gałęzi działu testów w procesie tworzenia oprogramowania. Ich głównym założeniem jest zmniejszenie nakładu pracy, którą trzeba wykonać by przetestować tworzoną aplikację [6]. Automatyzowane powinny być te testy, które są powtarzalne, a testy wykonywane są regularnie. Daje to pewność, że testy automatyczne spełnią swoją rolę.

Wiedząc, że automaty są potrzebne w projekcie, należy podjąć decyzję o tym, z jakiego narzędzia należy skorzystać. Może to zaważyć na tym, czy projekt automatyzacji odniesie sukces, czy też okaże się porażką [3]. Pewien wpływ na to ma wydajność testów, które są pisane. Jeśli się okaże, że są one niewiele szybsze od testów manualnych, to wysiłek i koszt poniesione do ich napisania okaże się zmarnowany. Dlatego w artykule został podjęty temat wydajności testów pisanych w dwóch popularnych technologiach dostępnych na rynku: Selenium i UFT. Przedstawione wyniki mają na celu odpowiedzi na pytanie, która z przedstawionych technologii jest wydajniejsza, czyli w rozpatrywanym przypadku szybsza.

2. Przegląd literatury

Porównanie Selenium i UFT było już poruszane w literaturze naukowej. Między innymi Prof. Dashrath Mane i Prachi Kunte zajmowali się tym zagadnieniem [2]. Porównywali obie technologie na tle wspieranych systemów i przeglądarek, wydajności sprzętowej i szybkości, z jaką testy działają.

Również Yogesh Kumar porównuje między innymi Selenium i UFT [1]. Na podstawie różnych parametrów ocenia, która technologia jest najlepsza. Parametry brane pod uwagę to: możliwość nagrywania, możliwość generowania skryptów, testowanie oparte na danych, wygląd raportów, ponowne użycie skryptów, szybkość wykonywania testów, łatwość w nauczeniu się danej technologii, koszt.

Dodatkowo Harsh Bajaj pisał na temat wyboru narzędzi do automatyzacji do projektu [3]. Szukając najlepszej technologii brał po uwagę następujące czynniki: łatwość zaadoptowania technologii do projektu, łatwość tworzenia skryptów i możliwości raportowania, techniczne możliwości technologii (wspierane systemy, przeglądarki itp.).

3. Materiały i metody

W celu sprawdzenia wydajności wybranych narzędzi należało stworzyć z użyciem tych technologii kilka testów. W artykule zostały utworzone 3 przypadki testowe dla testowanej aplikacji, którą była platforma e-learningowa Moodle, z której korzysta Politechnika Lubelska. Na podstawie tych przypadków testowych powstały testy automatyczne, które posłużyły do przetestowania wydajności Selenium i UFT.

3.1. Technologie

3.1.1. Selenium

Selenium to prawdopodobnie najpopularniejszy framework (szkielet aplikacji) do pisania testów automatycznych aplikacji internetowych. Jest całkowicie darmowy, dostępny na licencji open-source. Wspiera wiele różnych języków programowania (Java, C#, Python itp.) [1,7]. Wspiera też wiele narzędzi ciągłej integracji (ang. continuous integration) takich jak Bamboo czy Jenkins [5].

3.1.2. UFT

UFT to narzędzie do automatyzacji stworzone przez firmę HP (Hewlett-Packard), a obecnie rozwijane przez firmę MicroFocus [2]. Umożliwia tworzenie testów automatycznych GUI, API, aplikacji mobilnych, aplikacji desktopowych i innych. UFT jest dostępne dla systemu Windows i w swojej głównej funkcjonalności wykorzystuje język VBScript [4]. Największą zaletą tej technologii jest wbudowane repozytorium obiektów, gdzie przechowywane są definicje obiektów wykorzystywanych w testach [8]. Wystarczy kliknąć w obiekt na stronie, by ten pojawił się w repozytorium. Dzięki temu nie ma potrzeby pisać definicji obiektów w kodzie, tak jak to ma miejsce w Selenium.

3.2. Przypadki testowe

Przypadki testowe powstały, jako lista kroków, które należy wykonać by test zakończył się sukcesem. Każdy krok testu miał opisywał to, co należy wykonać oraz oczekiwany rezultatu, który świadczył o tym, że akcja została wykonana prawidłowo.

Pierwszy test polegał przede wszystkim na uzupełnianiu pól formularza. Poza wpisywaniem określonego tekstu do pól tekstowych, test musiał też rozwijać zakładki, w których te pola się znajdowały. Test został przedstawiony w Tabeli 1.

Uzupełnianie formularza to jedno z podstawowych zadań testów automatycznych, w czym sprawdzają się bardzo dobrze. Test automatyczny robi to o wiele szybciej od człowieka. Takie testy nadają się doskonale do automatyzacji.

Drugi test polegał na dodaniu do sprawozdania z laboratorium studenta pliku i dodania komentarza do tego pliku. Następnie Użytkownik wypisywał się z kursu i ponownie się do niego zapisywał. Test został zaprezentowany w Tabeli 2.

Tabela 1. Pierwszy przypadek testowy stworzony na potrzeby badań

Lp.	Opis	Rezultat
1.	Zaloguj się do aplikacji, jako student	Użytkownik zalogowany
2.	Wejdź do profilu użytkownika	Otwiera się ekran profilu użytkownika
3.	Uruchom modyfikację profilu użytkownika	Pojawił się formularz do modyfikacji profilu
4.	W zakładce ogólne zmień: imię, nazwisko email i miasto	Dane zostały zmienione
5.	W zakładce zdjęcie użytkownika dodaj zdjęcie	Zdjęcie zostało dodane
6.	W zakładce "Dodatkowe nazwy" uzupełnij pola: imię- fonetyczne, nazwisko-fonetyczne, drugie imię, alternatywna nazwa	Dane zostały zmienione
7.	W zakładce "Zainteresowana" uzupełnij pole: lista zainteresowań	Pole uzupełnione
8.	W zakładce "Opcjonalne" uzupełnij wszystkie pola	Pola uzupełnione
9.	W zakładce "Inne pola" zmień zawartość pól: Grupa dziekańska i rok rozpoczęcia studiów	Pola zmienione
10.	Zapisz zmiany	Zmiany zapisane. Powrót do ekranu profilu
11.	Wyloguj się	Użytkownik wylogowany

Test zawiera, poza standardowymi czynnościami jak klikanie i uzupełnianie pól tekstowych, załadowanie pliku z dysku komputera do sprawozdania.

Trzeci test miał na celu sprawdzenie, czy prawidłowo działa funkcja dodania do aplikacji nowego kursu dla studentów i dodanie do niego pliku, który będzie można pobrać na dysk. Test został przedstawiony w Tabeli 3.

Ponownie test polegał na załadowaniu pliku na serwer, jednak dodatkowo, ten plik jest następnie pobierany i test sprawdza, czy faktycznie plik się pobrał.

3.3. Testy automatyczne

Każdy z przypadków testowych został zaimplementowany w formie testu automatycznego dla obu badanych technologii. Zamierzeniem było, by testy były napisane w sposób zbliżony do siebie. Jeśli w jakimś kroku zostały kliknięte po sobie 3 przyciski, a następnie sprawdzone pojawienie się jakiegoś obiektu na stronie to w obu wersjach testu, dla Selenium i UFT powinny zostać wykonane te same czynności. W tym zakresie cel został osiągnięty, co widać na przykładach 1 i 2. Ten sam fragment testu pokazany w obu technologiach.

Tabela 2. Drugi przypadek testowy stworzony na potrzeby badań

Lp.	Opis	Rezultat
1.	Zaloguj się do aplikacji, jako student	Użytkownik zalogowany
2.	Przejdź do zakładki "Kursy"	Pojawia się lista działów
3.	Wyszukaj kurs "Zespołowy Projekt Programistyczny TWO IMNS_MP" i wejdź do niego	Pojawia się zawartość kursu
4.	Pobierz pierwszy plik na dysk	Plik zapisany na dysku
5.	Wejdź do sprawozdania i dodaj plik	Plik dodany do sprawozdania
6.	Dodaj komentarz do sprawozdania	Komentarz widoczny na karcie sprawozdania
7.	Wypisz użytkownika z kursu	Użytkownik wypisany z kursu. Zniknął z listy "Moje kursy"
8.	Wyszukaj ponownie kurs "Zespołowy Projekt Programistyczny TWO IMNS_MP" i wejdź do niego	Pojawia się ekran do wpisania klucza dostępu do kursu
9.	Podaj klucz dostępu by zapisać się na kurs	Pojawia się ekran kursu
10.	Wyloguj się z aplikacji	Użytkownik wylogowany

Tabela 3. Trzeci przypadek testowy stworzony na potrzeby badań

Lp.	Opis	Rezultat
1.	Zaloguj się do aplikacji	Użytkownik się zalogował
2.	Przejdź do zakładki 'Kursy'	Wyświetlona została lista działów
3.	Wejdź do działu 'Kursy testowe'	Wyświetlone zostały lista kursów w dziale
4.	Wciśnij przycisk 'Dodaj nowy kurs'.	Wyświetlony został formularz dla nowego kursu
5.	Uzupełnij pola obowiązkowe	Obowiązkowe pola zostały wypełnione
6.	Uzupełnij pole: 'Podsumowanie kursu'	Pole uzupełnione
7.	Wciśnij przycisk 'Zapisz i wyświetl'	Ekran kursu wyświetlony
8.	Włącz tryb edycji	Kurs przeszedł w tryb edycji
9.	Dodaj aktywność 'Plik' do 'Temat 1'	Wyświetlony ekran dodawania 'lekcji'
10.	Uzupełnij pola 'Nazwa' i 'Opis'	Pola uzupełnione
11.	Dodaj plik	Plik dodany
12.	Wciśnij przycisk 'Zapisz i wyświetl'	Plik jest widoczny na ekranie kursu
13.	Pobierz plik na dysk	Plik pobrany na dysk
14.	Wyloguj się z aplikacji	Użytkownik się wylogował

Przykład 1. Fragment kodu testu modyfikującego profil użytkownika, uzupełniający pola w sekcji „Inne nazwy" (Selenium)

```
.clickOptionalSection()
.setUrl("www.test.te")
.setContact(Contact.icq, "test")
.setContact(Contact.skype, "test")
.setContact(Contact.aim, "test")
.setContact(Contact.yahoo, "test")
.setContact(Contact.msn, "test")
.setIdNumber("111")
.setPhone1("134234223")
.setPhone2("123432344")
.setAddress("testowa 1/3")
```

Przykład 2. Fragment kodu testu modyfikującego profil użytkownika, uzupełniający pola w sekcji „Inne nazwy" (UFT)

```
'<-------------------------------- Step 7 ---------------------------->
Browser("Moodle").Page("Common").WebButton("Button role
button").SetTOProperty "name", Opcjonalne"
Browser("Moodle").Page("Common").WebButton("Button role button").Click

blnCheck = FnSetWebEdit("Strona WWW", "Opcjonalne", test.pl")
blnCheck = blnCheck AND FnSetWebEdit("Icq", "Opcjonalne", "test")
blnCheck = blnCheck AND FnSetWebEdit("Skype", "Opcjonalne", "test")
blnCheck = blnCheck AND FnSetWebEdit("AIM", "Opcjonalne", "test")
blnCheck = blnCheck AND FnSetWebEdit("Yahoo", "Opcjonalne", "test")
blnCheck = blnCheck AND FnSetWebEdit("MSN", "Opcjonalne", "test")
blnCheck = blnCheck AND FnSetWebEdit("Telefon", "Opcjonalne", "883388332")
blnCheck = blnCheck AND FnSetWebEdit("Tel. komórkowy", "Opcjonalne",
"111111222")
blnCheck = blnCheck AND FnSetWebEdit("Adres", "Opcjonalne", "Moodlowa 2/2")
```

3.4. Porównanie testów

Napisane testy zostały poddane pomiarowi. Każdy z nich został uruchomiony po 10 razy, a otrzymane wyniki zostały porównane ze sobą. Pomiary miały na celu zebranie wyników i wyciągnięcie z nich wartości średniej oraz odchylenia standardowego.

4. Wyniki

Dla wszystkich testów, średni czas trwania testu jest niższy niż dla testów pisanych z użyciem Selenium. Różnica, w przypadku testu nr 3 sięga aż pół minuty. To dużo, biorąc pod uwagę, że oba testy robią to samo. Wyniki zostały przedstawione w Tabeli 4.

Tabela 4. Pomiary czasu trwania egzekucji poszczególnych testów w obu technologiach

	Test 1	Test 2	Test 3
	Czas [minuty]		
HP UFT	01:45	02:48	02:10
	01:30	02:19	02:17
	01:37	02:23	02:26
	01:33	02:12	02:06
	01:25	02:40	02:11
	01:29	02:17	02:19
	01:25	02:14	02:25
	01:30	02:13	02:09
	01:25	02:33	02:11
	01:29	02:32	02:16
Średnia	01:30	02:25	02:15
Odchylenie standardowe	0,004353	0,008676	0,004721
Selenium	01:05	02:12	01:32
	01:06	02:02	01:53
	01:06	02:48	01:52
	01:13	02:35	01:43
	01:13	02:15	01:40
	00:59	02:22	01:46
	01:08	02:30	01:41
	01:03	02:09	01:41
	01:10	02:19	01:57
	01:10	02:29	01:50
Średnia	01:07	02:22	01:45
Odchylenie standardowe	0,003072	0,009502	0,00523

5. Wnioski

Z przeprowadzonych badań wynika, że jeśli ważnym argumentem przy wyborze technologii automatyzacji jest, by czas potrzebny na uruchomienie wszystkich testów był jak najkrótszy, to lepszym wyborem będzie zdecydowanie Selenium WebDriver. Jest ono szybsze pod każdym względem od UFT. Jednak wybór odpowiedniej technologii to o wiele bardziej złożona sprawa niż wybranie tej najszybszej. To czy dana technologia będzie najbardziej odpowiednia mają wpływ również takie aspekty jak:

- obsługiwane technologie;
- cena;
- łatwość znalezienia specjalistów od danej technologii, bądź wyszkolenia nowego;
- łatwość tworzenia skryptów.

Im większą posiada się wiedzę na temat tego, czego się oczekuje, tym łatwiej jest podjąć ostateczną decyzję. W przypadku, gdy nie zachodzi potrzeba automatyzowania zwykłej aplikacji komputerowej, to Selenium można ocenić, jako lepszy wybór niż UFT.

Literatura

[1] AUTOMATION TESTING Tutorial: Process, Planning &Tools, http://www.guru99.com/automationtesting.html [11.03.2018]

[2] H. P. Bajaj, Choosing the right automation tool and framework is critical to project success, https://pdfs.semanticscholar.org/526a/5b00714e1cad6809234020cd8f81fdfb905a.pdf [11.03.2018]

[3] P. Kunte, Prof. D. Mane, Automation Testing of Web based application with Selenium and HP UFT (QTP), International Research Journal of Engineering and Technology (IRJET), Volume: 04 Issue: 06 June -2017 https://www.irjet.net/archives/ /i6/IRJETV4I6500.pdf [11.03.2018]

[4] Y. Kumar, Journal of Emerging Technologies and Innovative Research (IRJET),, 09-2015, Volume 2, Issue 9 Comparative Study of Automated Testing Tools: Selenium, SoapUI, HP Unified Functional Testing and Test Complete, http://www.jetir.org/papers/JETIR1509007.pdf [11.03.2018]

[5] P. Wałachowski, Porównanie narzędzi do automatyzacji cross-przeglądarkowych testów funkcjonalnych aplikacji webowej, https://moodle.cs.pollub.pl/mod/resource/view.php?id=6709 [11.03.2018]

[6] Test Automation Tool Comparison – HP UFT/QTP vs. Selenium, http://www.aspiresys.com/WhitePapers/QTPvs Selenium.pdf [11.03.2018]

[7] Introduction to HP Unified Functional Testing (UFT) - Latest version of Quick Test Professional (QTP), http://www.softwaretestingclass.com/introduction-to-hp-unified-functional-testing-uft/ [11.03.2018]

[8] HP UFT/QTP vs. Selenium – Automated Test Tool Comparison, http://www.optimusinfo.com/blog/7392 [11.03.2018]