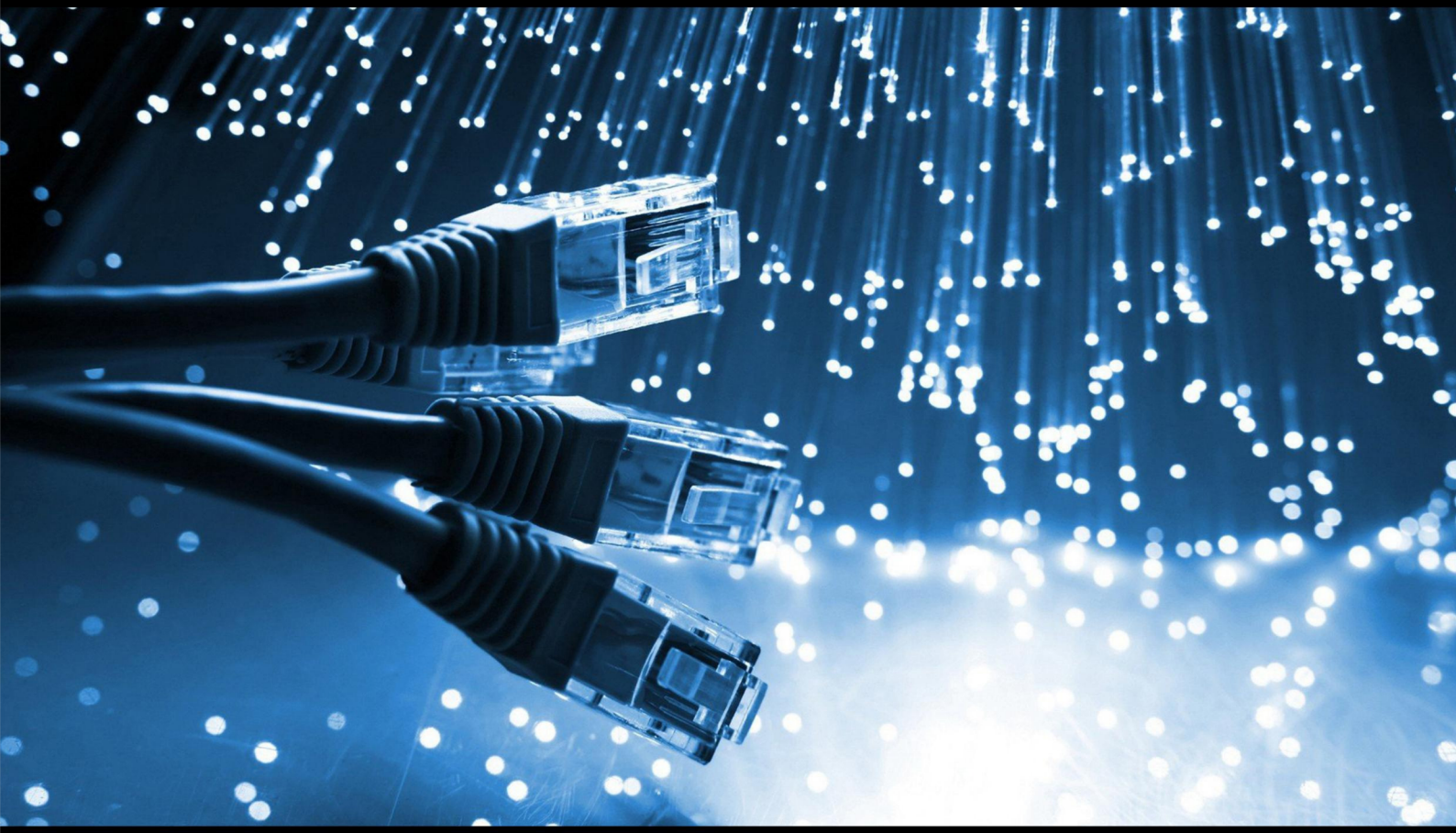
JCSI

Journal of Computer Sciences Institute

Volume 12/2019



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. Sławomir Przyłucki
dr inż. Maciej Pańczyk
dr Edyta Łukasik
dr inż. Maria Skublewska-Paszkowska
dr Paweł Powroźnik
dr inż. Piotr Kopniak
dr Beata Pańczyk
dr inż. Jacek Kęsik
dr inż. Jakub Smołka
dr inż. Marek Miłosz
dr inż. Małgorzata Plechawska-Wójcik

Skład komputerowy:
Monika Kaczorowska
e-mail: m.kaczorowska@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:
Sławomir Przyłucki
Maciej Pańczyk
Edyta Łukasik
Maria Skublewska-Paszkowska
Paweł Powroźnik
Piotr Kopniak
Beata Pańczyk
Jacek Kęsik
Jakub Smołka
Marek Miłosz
Małgorzata Plechawska-Wójcik

Computer typesetting:
Monika Kaczorowska
e-mail: m.kaczorowska@pollub.pl

Cover design:
Marta Zbańska

ISSN 2544-0764

Spis treści

Contents

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 12 (2019) 167-171
Wysłane: 2019-04-30
Przyjęte: 2019-05-06

Hybrydowe metody pracy z bazami danych w aplikacjach JEE

Katarzyna Jóźwicka*, Mariusz Mitrus*
Politechnika Lubelska, Instytut Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. W artykule przedstawiono możliwości hybrydyzacji metod pracy z bazami danych w aplikacjach JEE. Do przeprowadzenia badań wykorzystano aplikacje wykonane w oparciu o interfejs JDBC, Hibernate oraz Spring. Analiza wydajności aplikacji dotyczyła czasu wykonywania oraz zużycia pamięci RAM dla podstawowych operacji CRUD w bazie danych.

Słowa kluczowe: JEE; Hibernate; Spring; JDBC; Bazy danych; wydajność

*Autor do korespondencji.

Adresy e-mail: katrzyna.jozwicka@pollub.edu.pl, mariusz.mitrus@pollub.edu.pl

Hybrid methods of working with databases in JEE applications

Katarzyna Jóźwicka*, Mariusz Mitrus*
Institute of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract. The article presents the research on hybridization of methods of working with databases in JEE applications. The test applications were made based on the JDBC interface, Hibernate and Spring framework. Application performance analysis covered the execution time and RAM usage for basic CRUD operations in the database.

Keywords: JEE; Hibernate; Spring; JDBC; Databases; performance

*Corresponding author.

E-mail addresses: katrzyna.jozwicka@pollub.edu.pl, mariusz.mitrus@pollub.edu.pl

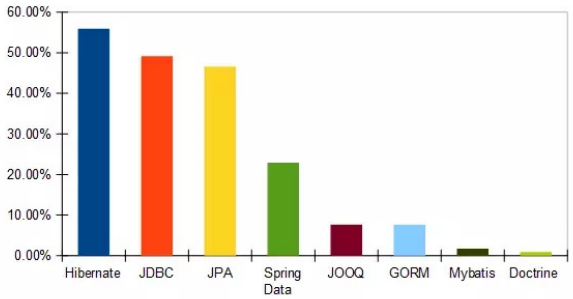
1. Wstęp

Rozwój i powszechność programowania obiektowego sprawiły, że programiści zmuszeni zostali do opracowania rozwiązania, które sprawnie konwertuje bazy danych na obiekty i odwrotnie. Do tej pory jednak nie udało się wykreować „powszechnie akceptowanej koncepcji obiektowej bazy danych" [1, 2]. Oczywiście podejmowano próby stworzenia takich baz danych np. z wykorzystaniem repozytorium XML lub dedykowanych rozszerzeń dla języka Java, czy C# [2]. Jednak pomimo upływu lat model relacyjny jest nadal podstawowym modelem organizacji danych [3].

Mapowanie obiektowo-relacyjne (ang. Object-Relational Mapping ORM) konwertuje dane między relacyjną bazą danych, a obiektami w aplikacji. Rozwiązanie oparte na ORM staje się bardziej użyteczne wraz ze wzrostem rozmiaru i złożoności projektu [4].

14 sierpnia 2010 roku została zaprezentowana biblioteka jOOQ (ang. Java Object Oriented Querying), która nie realizuje technologii mapowania. Tworząc bibliotekę jOOQ założono, że to SQL powinien być najważniejszy, jeśli chodzi o integrację z bazą danych. Dlatego też biblioteka ta nie wprowadza nowego języka zapytań (jak np. HQL w Hibernate), ale pozwala konstruować zwykły SQL z obiektów jOOQ i kodu wygenerowanego ze schematu baz danych [6].

Pomimo zalet JOOQ, najbardziej popularnym frameworkiem do pracy z bazami danych jest ciągle Hibernate. Jego największą zaletą, oprócz szybkości i możliwości pracy z wieloma bazami danych, jest fakt, że można go wykorzystać zarówno w małych, jak i dużych projektach [7]. Kolejnym według popularności interfejsów jeżyka JAVA króluje JDBC, zgodnie z rysunkiem numer 1.



Rys. 1. Popularność interfejsów bazodanowych [8]

2. Cel i teza badań

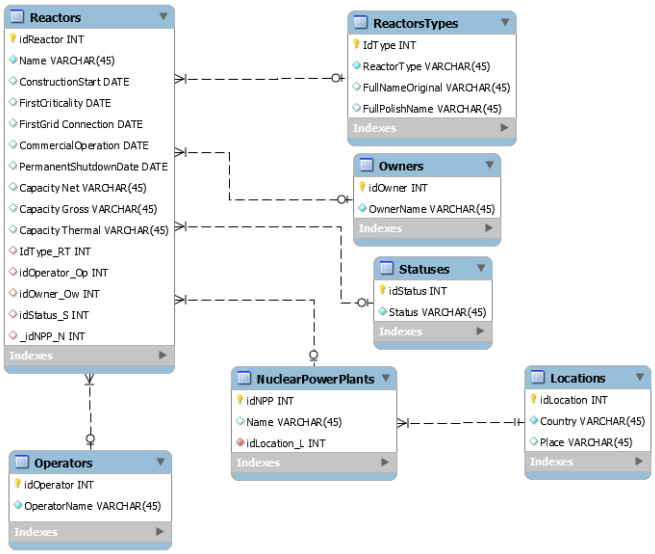
Celem badań była analiza możliwości hybrydyzacji metod pracy z bazami danych w celu zwiększenia wydajności pracy z danymi w aplikacjach JEE.

Za tezę badawczą przyjęto:

Hybrydyzacja metod pracy z bazami danych w aplikacjach JEE pozwoli przyspieszyć pracę z danymi w stosunku do rozwiązań standardowych.

3. Scenariusze i metoda badań

Na potrzeby badań utworzono bazę reaktorów uzupełnioną rzeczywistymi informacjami na temat wszystkich światowych 786 reaktorów (Rys. 2). Wszystkie dane pobrano ze strony http://www.world-nuclear.org/ [9]. Następnie dodano ponad 1000000 nowych, wygenerowanych losowo rekordów.



Rys.2. Diagram związków encj (ERD) bazy danych

Platforma oraz serwer zostały utworzone na dwóch stanowiskach pomiarowych, których parametry prezentuje tabela 1. Na serwerze została zainstalowana relacyjna baza danych MySql.

Tabela 1. Parametry stanowisk pomiarowych

Nazwa stanowiska	CPU	Pamięć RAM	Karta Graficzna	System
Stanowisko numer 1	Intel i7-3612QM, 4x2.1GHz	8.00GB	Intel Graphics 4000 + Nividia Geforce GT630M	Windows 7, 64 bitowy
Stanowisko numer 2	Intel i5 3337U	8.00GB	Intel Graphics 4000	Windows 7, 64 bitowy

Do badań wykorzystane zostały aplikacje testowe, które wykonywały klasyczne operacje CRUD zaimplementowane z wykorzystaniem JDBC, Hibernate i Spring. Badania odbyły się z użyciem zapytań: INSERT INTO, UPDATE, DELETE oraz SELECT, których użycie w scenariuszach prezentuje tabela 2. Po każdym teście przeprowadzonym zgodnie z danym scenariuszem, baza była przywrócona do stanu wcześniejszego. Dzięki temu dostępna była identyczna baza dla każdego scenariusza, a wyniki były w większym stopniu miarodajne. Każdy ze scenariuszy powtórzono 4 razy.

Tabela 2. Przykładowe scenariusze wykorzystane do badań

Numer scenariusza	Rodzaj zapytania	Opis
S1.1	SELECT	Odczyt nazw 5000 najstarszych reaktorów
S1.2	SELECT	Odczyt nazw 50000 najstarszych reaktorów
S1.3	SELECT	Odczyt nazw 100000 najstarszych reaktorów
S1.4	SELECT	Odczyt nazw 300000 najstarszych reaktorów
S1.5	INSERT INTO	Zapis 5000 reaktorów o do tabeli REACTORS z nazwą 'TEST'
S1.6	INSERT INTO	Zapis 50000 reaktorów do tabeli REACTORS z nazwą 'TEST'
S1.7	INSERT INTO	Zapis 100000 reaktorów do tabeli REACTORS
S1.8	INSERT INTO	Zapis 300000 reaktorów do tabeli REACTORS z nazwą 'TEST'

Metoda pomiaru wydajności opierała się na pomiarze zużycia pamięci RAM oraz czasu realizacji danego scenariusza (Przykład 1).

Przykład 1. Pomiar wydajności

```
long beforeUsedMem=Runtime.getRuntime().totalMemory()-
Runtime.getRuntime().freeMemory();//pomiar zużycia RAM
long elapsedTime=0;
long afterUsedMem=Runtime.getRuntime().totalMemory()-
Runtime.getRuntime().freeMemory();//Pomiar RAM
long actualMemUsed=afterUsedMem-beforeUsedMem;
out.println("Zużycie RAM[MB]: "+actualMemUsed/1000000 );
//konwersja na milisekundy:
float elapsedTimeMili
   =TimeUnit.NANOSECONDS.toMillis(elapsedTime);
out.println("<p>Czas w milisekundach: "+elapsedTimeMili);
```

W celu utworzenia aplikacji hybrydowych, wykonano najpierw scenariusze z wykorzystaniem standardowych metod. Na podstawie otrzymanych wyników okazało się, że dla rozważanej bazy danych, najlepszym rozwiązaniem pod względem wydajności dla scenariuszy od 1.1 do 1.4 oraz od 1.16 do 1.20 jest interfejs JDBC, a dla innych okazał się Hibernate. Jeśli chodzi o złożone zapytania, sytuacja przedstawiła się podobnie (tabela 3 i 4).

Tabela 3. Zestawienie najlepszych rozwiązań standardowych dla scenariuszy od 1.1 do 1.32

Liczba rekordów\ Rodzaj zapytania	5000	50000	100000	300000
SELECT	JDBC	JDBC	JDBC	JDBC
INSERT INTO	Hibernate	Hibernate	Hibernate	Hibernate
UPDATE	Hibernate	Hibernate	Hibernate	Hibernate
DELETE	Hibernate	Hibernate	Hibernate	Hibernate

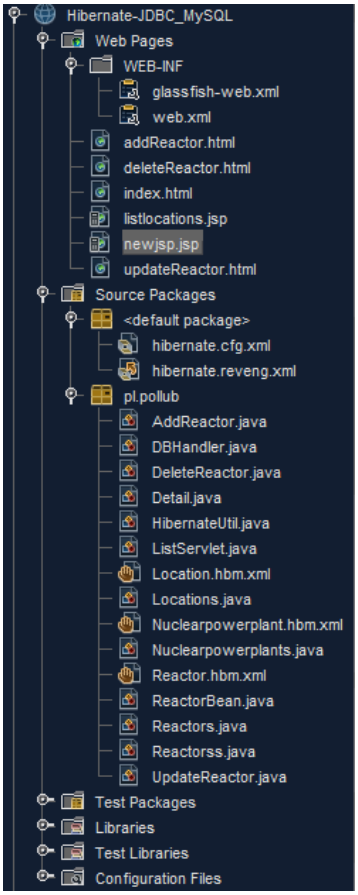
Tabela 4. Zestawienie najlepszych rozwiązań standardowych dla scenariuszy od 2.1 do 2.32

Liczba rekordów\ Rodzaj zapytania	5000	50000	100000	300000
SELECT	Spring+JDBC/JDBC	Spring+JDBC/JDBC	Spring+JDBC/JDBC	Spring+JDBC/JDBC
INSERT INTO	Hibernate	Hibernate	Hibernate	Hibernate
UPDATE	Hibernate	Hibernate	Hibernate	Hibernate
DELETE	Hibernate	Hibernate	Hibernate	Hibernate

Na podstawie wyników przedstawionych w tabeli 3 i 4 zaproponowano rozwiązania hybrydowe.

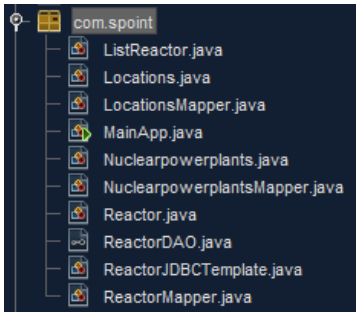
4. Programy hybrydowe

Hybrydyzacja JDBC i Hibernate (Rys. 3) została zaimplementowana w postaci aplikacji internetowej z możliwością wyboru żądanego typu zapytania na stronie *index.html*. Dodatkowe akcje na bazie danych obsługują odpowiednio strony *updateReactor.html*, *deleteReactor.htlm* oraz *addReactor.html*. *SelectReactor.html* jest stroną z formularzem wykorzystującym klasę *ListServlet*, która wywołuje metody *loadReactors* klasy *DBHandler,* dokładnie jak w przypadku programu korzystającego tylko z połączenia JDBC. Modyfikacja zapytań typu SELECT polegała na zmianie kodu w metodzie *loadReactors*.



Rys 3. Struktura programu hybrydowego Hibernate+JBDC

Hybrydyzacja JDBC, Spring + JBDC i Hibernate polegała na dodaniu do kopii hybrydy JDBC i Hibernate metod z aplikacji Spring + JDBC. Wszystkie klasy i metody JDBC i Hibernate są identyczne jak w przypadku wcześniejszej hybrydy. Służą one do zapytań typu SELCECT w scenariuszach od 1.1 do 1.32, oraz typu INSERT INTO, DELETE i UPDATE dla scenariuszy od 2.1 do 2.32. Pliki programu Spring + JDBC zostały dodane w oddzielnym pakiecie *com.spoint* (Rys. 4) oraz do pliku konfiguracyjnego *Beans.xml*. W celu wykonania powiązanych zapytań typu SELECT i wyświetlenia ich wyników, konieczne było utworzenie dodatkowej klasy *ListReactor*. Zmiana zapytania została wykonana w metodzie *loadReactor()* klasy *ReactorJDBCTemplate.*



Rys. 4. Pakiet *com.spoint* dla Spring+JDBC

5. Wyniki testów

Średnie wartości pomiarów dla programów hybrydowych przedstawiają tabele 5, 6 oraz rysunki 5, 6 dla scenariuszy z zapytaniem typu SELECT. Program hybrydowy JDBC i Hibernate okazał się najbardziej efektywny.

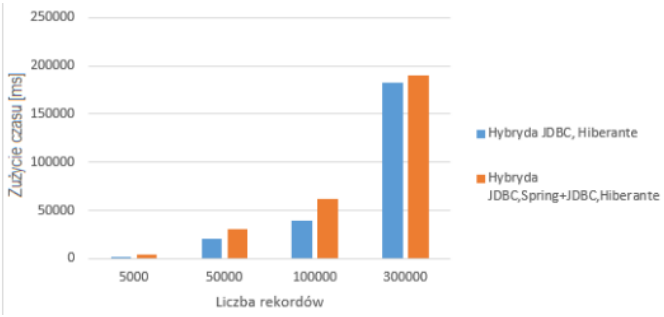
Tabela 5. Średnie wyniki testów programów hybrydowych dla scenariuszy od 1.1 do 1.32

Rodzaj zapytania	Liczba rekordów	Hybryda JDBC, Hiberante		Hybryda JDBC,Spring+JDBC,Hiberante	
		RAM[MB]	Czas[ms]	RAM[MB]	Czas[ms]
Select	5000	19	2002	46	3546
	50000	31	20363	150	31012
	100000	56	38770	222	61513
	300000	113	182536	386	190562
Insert into	5000	27	2674	27	2867
	50000	95	24943	96	24820
	100000	77	44855	76	45159
	300000	103	158848	107	156538
Update	5000	71	8014	72	7713
	50000	93	42391	98	42106
	100000	105	54207	110	54797
	300000	115	124693	128	126575
Delete	5000	44	4731	43	5655
	50000	34	34855	35	34624
	100000	85	75506	85	80413
	300000	92	281957	91	289255

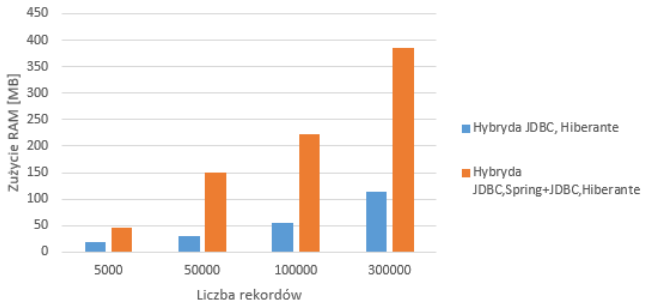
Tabela 6. Średnie wyniki testów dla programów hybrydowych dla scenariuszy od 2.1 do 2.32

Rodzaj zapytania	Liczba rekordów	Hybryda JDBC, Hiberante		Hybryda JDBC,Spring+JDBC,Hiberante	
		RAM[MB]	Czas[ms]	RAM[MB]	Czas[ms]
Select	5000	23	6302	264	6664
	50000	40	56299	729	57077
	100000	65	161474	1170	168300
	300000	149	716576	2194	730326
Insert into	5000	29	3172	27	3207
	50000	93	27406	91	27477
	100000	82	49294	77	49098
	300000	104	174231	104	175689
Update	5000	78	8184	77	7636
	50000	95	45361	93	46025
	100000	103	59102	97	60617
	300000	113	137886	118	137697
Delete	5000	51	27617	51	33076
	50000	43	204162	43	190210
	100000	102	412749	113	424652
	300000	120	1572076	119	1582120

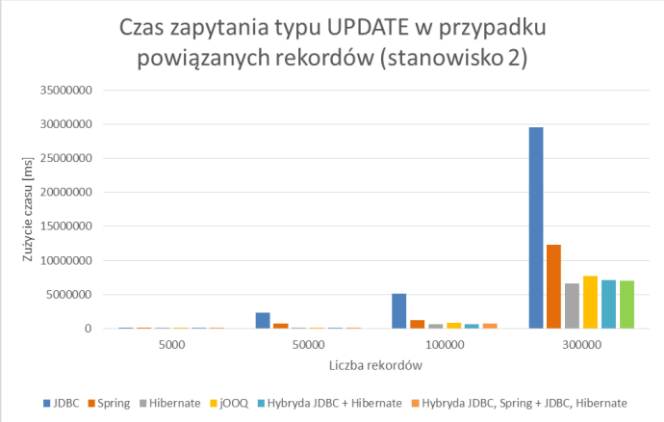
Rozwiązania hybrydowe są znacząco lepsze od standardowych metod pracy z bazami danymi, co przedstawiają rysunki 7 i 8. Średnie wyniki programu hybrydowego JDBC+Hibernate w porównaniu do średniej standardowych metod prezentuje tabela 7. Dla hybrydy Spring+JDBC i Hibernate wyniki przedstawia tabela 8.



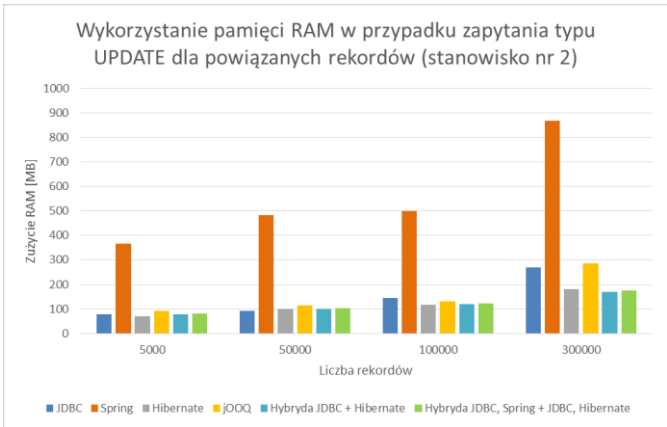
Rys. 5. Średni czas zapytań typu SELECT dla scenariuszy 1.1-1.32



Rys. 6. Średnie wykorzystanie pamięci RAM w przypadku scenariuszy 1.1-1.32



Rys. 7. Średni czas zapytania typu UPDATE w przypadku scenarisuzy 2.1-2.32 (Stanowisko numer 2)



Rys. 8. Średnie wykorzystanie pamięci RAM w przypadku zapytania typu UPDATE dla scenarisuzy 2.1-2.32(Stanowisko numer 2)

Tabela 7. Poprawa (w %) średnich wyników dla JDBC+Hibernate w porównaniu ze średnimi wynikami standardowych metod

Typ zapytania	Zużycie RAM [MB]	Zużycie czasu [ms]
SELECT	34%	6%
INSERT INTO	2%	44%
UPDATE	13%	39%
DELETE	-18%	37%

Tabela 8. Poprawa (w %) średnich wyników dla Spring+JDBC+Hibernate w porównaniu ze średnimi standardowych metod

Typ zapytania	Zużycie RAM [MB]	Zużycie czasu [ms]
SELECT	55%	6%
INSERT INTO	3%	43%
UPDATE	11%	39%
DELETE	88%	33%

6. Wnioski

Przeprowadzone badania wykazały, że w pracy z bazami danych dużą rolę w optymalizacji, zwiększeniu szybkości dostępu do rekordów może odgrywać hybrydyzacja.

Na podstawie uzyskanych wyników wybrano najlepsze rozwiązania i stworzono dwie aplikacje hybrydowe: JDBC i Hibernate, a także JDBC, Spring+JDBC i Hibernate. Poprzez analizę uzyskanych rezultatów stwierdzono, że najlepszym rozwiązaniem jest aplikacja hybrydowa łącząca JDBC i Hibernate. Zastosowanie JDBC dla zapytań typu SELECT, Hibernate dla INSERT, UPDATE i DELETE pozwoliło ograniczyć zużycie pamięci RAM, a także zmniejszyć czas potrzebny do zwrócenia wyników zapytań.

Wykorzystując najlepsze rozwiązania dla poszczególnych typów zapytań, można więc znacząco poprawić wydajność aplikacji, umożliwić użytkownikowi sprawniejszy dostęp do bazy, niezależnie od tego, jakiego rodzaju danych potrzebuje. Oczywiście takie rozwiązanie wymaga od programisty dogłębnego zapoznania się z dostępnymi na rynku rozwiązaniami oraz samą bazą danych, wykonania odpowiednich badań i ich późniejszej analizy. Jednak włożony na początku wysiłek, może później znaczącą skrócić pracę nad późniejszą optymalizacją gotowej aplikacji.

W prowadzeniu badań bardzo pomocne są wyniki uzyskiwane przez innych badaczy. Niniejszy artykuł stanowi rozwinięcie badań prowadzonych w publikacji [10, 11], ale wyniki nie mogą być bezpośrednio porównane. W niniejszej pracy zwiększono liczbę scenariuszy oraz liczbę rekordów. W obu pracach, w przypadku stosowania rozwiązań standardowych, Hibernate okazał się najszybszą metodą dla zapytań INSERT a JDBC dla SELECT. Proponowanym (ale nie testowanym) rozwiązaniem w pracy [10] jest zastosowanie aplikacji hybrydowej z Hibernate dla większości zapytań, lecz z użyciem jOOQ, zamiast JDBC, jak w pracy niniejszej.

Istotnym elementem jest struktura samej bazy danych, serwerów baz danych oraz parametrów środowiska. Na przykład w pracy [10] testy prowadzono dla JBOSS oraz MSSQL. W niniejszej pracy wykorzystano WampServer

z MySQL, i najbardziej optymalne wyniki uzyskano dla JDBC i Hibernate.

Wyniki przedstawione w pracy pozwoliły potwierdzić, postawiona na wstępie tezę.

Hybrydyzacja metod pracy z bazami danych w aplikacjach JEE pozwoli przyspieszyć pracę z danymi w stosunku do rozwiązań standardowych.

Literatura

[1] Oracle, Java™ EE at a Glance, https://www.oracle.com/ technetwork/ java/javaee/overview/index.html, [maj 2018].

[2] Rosiek Z.: Mapowanie obiektowo-relacyjne (ORM) – czy to dobra idea? Zeszyty Naukowe Rok 4 Zeszyt 4, Warszawa 2010, s. 99-112.

[3] Teamquest, Relacyjne bazy danych – dlaczego warto je znać?, https://teamquest.pl/blog/461_relacyjne-bazy-danych-dlaczego-warto-je-znac, [czerwiec 2018].

[4] Object Oriented Application Cooperation Methods with Relational Database (ORM) based on J2EE Technology, P. Ziemniak , B. Sakowicz , A. Napieralski, 2007 9th International Conference - The Experience of Designing and Applications of CAD Systems in Microelectronics, 2007.

[5] NoSQL2: SQL to NoSQL Databases, J. Adriana, M. Holanda, World Conference on Information Systems and Technologies, 2018.

[6] Working with JOOQ, K. Siva Prasad Reddy, 2017.

[7] Javapipe, 10 Best Java Web Frameworks to Use in 2019, https://javapipe.com/blog/best-java-web-frameworks/, [maj 2018].

[8] Bill William, Are Spring and Hibernate still popular?, https://www.quora.com/Are-Spring-and-Hibernate-still-popular, [czerwiec 2018].

[9] World Nuclear Association and IAEA Power Reactor Information System, Information Library - World Nuclear Association , http://www.world-nuclear.org/information-library/facts-and-figures/reactordatabase-search.aspx, [maj 2018].

[10] Grzesińska M., Waszczyńska M.: Analiza porównawcza technologii ORM stosowanych w internetowych aplikacjach JEE. Praca magisterska, Politechnika Lubelska, 2016.

[11] Grzesińska M., Waszczyńska M., Pańczyk B. Wydajność pracy z bazami danych w aplikacjach JEE. IAPGOŚ- 2016, nr 4, vol. 6, s. 73-76.

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 12 (2019) 172-178
Wysłane: 2019-05-10
Przyjęte: 2019-05-14

Narzędzia wspomagające projektowanie interfejsu użytkownika aplikacji webowych – analiza porównawcza

Paweł Serafin*, Marek Miłosz

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

Streszczenie. Analiza porównawcza narzędzi do projektowania graficznych interfejsów użytkownika została przeprowadzona za pomocą analizy wielokryterialnej. Przeprowadzone badania wybranych narzędzi pozwoliły określić wartości kryteriów diagnostycznych, które posłużyły do wyboru najlepszego narzędzia.

Słowa kluczowe: interfejs użytkownika; projektowanie; narzędzia wspomagające; graficzny interfejs użytkownika

*Autor do korespondencji.

Adresy e-mail: pawel.serafin93@o2.pl, m.milosz@pollub.pl

Tools to support user interface design web applications - comparative analysis

Paweł Serafin*, Marek Miłosz

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

Abstract. Comparative analysis tools for the design of graphical user interfaces was carried out using a multi-criteria analysis. The research helped to determine the selected tools of diagnostic criteria, which were used to choose the best tool.

Keywords: user interface; designing; supporting tools; graphical user interface

*Corresponding author.

E-mail addresses: pawel.serafin93@o2.pl, m.milosz@pollub.pl

1. Wstęp

W dobie Internetu aplikacje webowe w systemach informatycznych używane są przez większość społeczeństwa każdego dnia, zarówno prywatnie oraz zawodowo. Klienci coraz częściej oczekują aplikacji o nowoczesnym, ergonomicznym, kolorowym oraz przejrzystym wyglądzie.

Interfejsem użytkownika nazywa się program, który umożliwia współpracę człowieka z oprogramowaniem komputera czy urządzenia mobilnego.

Liczba dostępnych narzędzi wraz z ich dodatkowymi rozszerzeniami, za pomocą, których można tworzyć interfejs aplikacji jest dość duża. Również rola projektantów interfejsu zaczyna wykraczać poza samo projektowanie. Dziś od projektantów wymaga się wszechstronności oraz znajomości zagadnień związanych z prototypowaniem i ergonomią.

2. Projektowanie w cyklu wytwarzania oprogramowania

Wytwarzanie oprogramowania składa się z kilku kluczowych faz, na które składają się następujące elementy: określenie wymagań, projektowanie, implementacja, testowanie, konserwacja.

W celu ułatwienia wytwarzania oprogramowania stworzono modele, które w bardzo dużym stopniu pomagają uporządkować cały ten proces. Jednym z nich jest model kaskadowy (wodospadowy), który polega na wykonywaniu każdego z elementów przedsięwzięcia jako odrębnej fazy, następują one jedna po drugiej. W przypadku gdy, pierwsza z faz zwróci niepożądany efekt, następuje powrót, wykonując iterację aż do momentu uzyskania satysfakcjonującego rozwiązania. Tego modelu najczęściej używa się gdy wymagania systemu są zrozumiałe oraz przejrzyste, ponieważ każdy powrót (dodatkowa iteracja) jest czasochłonna i kosztowna [1, 2].

2.1. Graficzny interfejs użytkownika

Graficzny interfejs użytkownika jest to sposób prezentacji informacji przez komputer oraz interakcji z użytkownikiem. Głównym elementem jest okno programu (lub kilka okien). Warstwa interfejsu graficznego jest warstwą najbliżej użytkownika, dlatego prócz wizualnej strony warto zadbać o jej funkcjonalność [3].

Zaprojektowanie odpowiedniego graficznego interfejsu użytkownika jest skomplikowanym i trudnym procesem. Zazwyczaj wykorzystywane jest podejście, które zawiera wzory okien czy formularzy. Zawierają one poszczególne elementy, kontrolki graficznego interfejsu zaprezentowane jako figury geometryczne czy przyciski [4, 5].

2.2. Metody projektowania interfejsów użytkownika

Powstało wiele różnych sposobów tworzenia interfejsów. Jedną z nich jest metoda doświadczalno-badawcza, która polega na stworzeniu, testowaniu prototypu interfejsu w trybie cyklicznym. Początkowy etap zawiera prymitywne

zarysy interfejsów, które z czasem stają się zaawansowanymi projektami. Każdy powrót do fazy początkowej skutkuje dążeniem do uzyskania lepszej charakterystyki funkcjonalnej.

Kolejną metodą jest Lean UX. Ten sposób projektowania polega na wzajemnej współpracy oraz komunikacji w zespole realizującym dany projekt. Głównym założeniem tego sposobu projektowania jest wzajemne współdziałanie, dzięki czemu deweloperzy biorący udział w przedsięwzięciu programistycznym, rozumieją go już od samego początku. Dzięki ograniczonej dokumentacji w tym modelu, możliwe jest szybsze przystąpienie do prac programistycznych [3].

3. Wybór narzędzi do projektowania interfejsów

Podczas projektowania graficznego interfejsu użytkownika należy dostosować odpowiednie narzędzia zarówno, dla danego projektanta, jak i do realizowanego projektu. Wybór identycznego oprogramowania dla wszystkich programistów może okazać się kosztowny, nieefektowny, powodujący znaczny wzrost kosztów, które zostały przeznaczone na realizację przedsięwzięcia programistycznego [6, 7, 8].

Rynek narzędzi wspomagających projektowanie interfejsu użytkownika jest bardzo szeroki, więc wybór może okazać się dość trudny i skomplikowany. Gdy projektant staje przed wyborem, musi dokładnie określić specyfikę projektu, dzięki czemu uzyska kryteria, które muszą zostać spełnione przez program do tworzenia interfejsów. Wybrane narzędzie jest kluczowe podczas dalszego projektowania, ponieważ w kolejnych etapach prac może ograniczać projektanta lub pozwolić mu na zrealizowanie innowacyjnych pomysłów [6, 7, 8].

Kryteria, którymi powinno posługiwać się podczas wybierania odpowiedniego narzędzia do projektowania interfejsów są następujące [6, 7, 8]:

- rodzaj urządzenia;
- system operacyjny;
- współpraca zespołu projektowego;
- koszt narzędzia;
- prezentacja stworzonego interfejsu;
- efektywność.

Do analizy porównawczej, ze względu na swą wzajemną konkurencyjność, zostały wybrane narzędzia Adobe XD oraz Figma.

3.1. Narzędzie Adobe XD

Narzędzie Adobe XD jest jednym ze stworzonych programów firmy Adobe Systems. Program jest dostępny na platformy Windows i Mac OS z opcją pracowania w trybie offline. Umożliwia tworzenie interaktywnych projektów, prototypów stron internetowych czy aplikacji mobilnych. Przeznaczony jest dla projektantów interfejsów graficznych. Aplikację można pobrać ze strony producenta po wcześniejszej rejestracji. W tej wersji bezpłatnej projektant otrzymuje podstawowe biblioteki, niewielką przestrzeń dysku chmurowego do wykorzystania wraz z możliwością udostępnienia tylko jednego projektu w czasie rzeczywistym. Wersja płatna kosztuje 12,29 € miesięcznie, przy czym poszerza funkcjonalności wymienionych ograniczeń. Adobe XD posiada niezwykle prosty interfejs, który został odziedziczony po pozostałych aplikacjach z pakietu firmy Adobe. Umożliwia projektowanie interfejsów dla różnych platform włączając w to również strony internetowe oraz urządzenia mobilne. Po włączeniu, program udostępnia wiele standardowych szablonów oraz komponentów. Obszar roboczy podzielony jest na dwa widoki: projekt oraz prototyp. Poszczególne elementy można dostosować do tworzonego projektu [9].

3.2. Narzędzie Figma

Narzędzie Figma coraz szybciej staje się alternatywą dla popularnych aplikacji do projektowania graficznych interfejsów, takich jak Sketch czy Adobe XD. Program ten nie wymaga pobrania oraz instalacji na komputerze. Do pracy z tą aplikacją potrzebne są tylko: dostęp do Internetu oraz przeglądarka internetowa, którą należy wybrać według własnych preferencji. Bezpłatna wersja oferuje trzy aktywne projekty oraz nieograniczoną liczbę użytkowników projektu wraz z trzydziestodniową historią zmian w projekcie. Płatna wersja o wartości 12 $ miesięcznie oferuje dodatkowe integracje z bibliotekami wraz z nielimitowaną historią zmian. Do rozpoczęcia pracy z programem wymagana jest wcześniejsza rejestracja na stronie. Figma posiada uproszczony interfejs zbliżony do konkurencyjnych programów. Na początku użytkownik otrzymuje kilka gotowych szablonów. Głównymi zaletami tej aplikacji są: prostota, zachowanie elementów opartych o kaskadowe arkusze stylów, obsługa oraz edycja obiektów wektorowych, możliwość eksportowania projektów o wysokiej jakości, bogaty zestaw narzędzi potrzebnych do projektowania aplikacji webowych oraz mobilnych [10].

4. Metodyka analizy porównawczej

Podczas wybierania narzędzia należy poddać analizie czynniki, które wpłyną na wybór najlepszego rozwiązania. Przed dokonaniem analizy porównawczej zostały dobrane odpowiednie zmienne, które posłużyły do celów diagnostycznych. Ich dobór ma bezpośredni wpływ na wynik badań, a ich nieprawidłowe wybranie może mieć negatywny skutek w postaci przekłamanych wyników. Wybranie zmiennych diagnostycznych dokonywane jest za pomocą kryteriów poza statystycznych oraz formalnych. Do analizy zostały wybrane cechy, które odgrywają znaczną rolę dla badanych narzędzi.

Zmienne traktować należy w sposób współmierny lub przypisać im odpowiednie wagi różnicowe, przy czym nie mogą być one ujemne a ich suma musi wynosić jeden. Przeprowadzana wielowymiarowa analiza porównawcza wiąże się z dokonaniem transformacji wybranych zmiennych w celu ujednolicenia parametrów. Etap ujednolicenia parametrów jest nazywany transformacją normalizacyjną. Jedną z metod normalizacji jest unitaryzacja, polegająca na przedstawieniu różnicy pomiędzy wartością maksymalną

a minimalną, której wynikiem jest zmienna z przedziału [0;1]. Powyższą formułę przedstawia wzór [11]:

$$Z_{ij} = \frac{X_{ij} - \min\{X_{ij}\}}{\max\{X_{ij}\} - \min\{X_{ij}\}} \quad (1)$$

gdzie:

X_{ij} – wartość j-tej zmiennej w i-tym obiekcie,

$\min\{X_{ij}\}$ – wartość minimalna j-tej zmiennej w i-tym obiekcie,

$\max\{X_{ij}\}$ – wartość maksymalna j-tej zmiennej w i-tym obiekcie.

Kolejnym krokiem jest przystąpienie do uśrednienia wcześniej unormowanych wartości. W tym celu należy przypisać do nich wagi [11, 12]. Wyznaczenie syntetycznej zmiennej realizowane jest przy pomocy wzoru [11]:

$$S_i = \sum_{j=1}^{m} Z_{ij} W_j \quad (2)$$

gdzie:

Z_{ij} – znormalizowana wartość j-tej zmiennej w i-tym obiekcie,

W_j – wartość wagi kryterium j-tej zmiennej.

5. Skala użyteczności systemu (System Usability Scale)

Skala użyteczności systemu (SUS) – stworzona przez Johna Brooka w 1986 roku [13, 14]. Jest szybkim pomiarem użyteczności sprzętu, systemów informatycznych, stron internetowych i aplikacji dokonywanym za pomocą ankiety.

Korzyści płynące z używania SUS są następujące [13, 17]:

- badanie jest łatwe dla uczestników biorących udział w badaniu,
- możliwe jest użycie małolicznych prób badawczych wraz z wiarygodnymi wynikami,
- pozwala skutecznie rozróżnić systemy użyteczne i mało użyteczne.

Wady płynące z używania SUS są następujące [13, 17]:

- skomplikowany system punktacji,
- istnieje możliwość omyłkowego zinterpretowania wyników jako procenty, ponieważ są w skali od 0 do 100,
- uzyskanie rankingu procentowego wiąże się z przeprowadzeniem normalizacji wyników,
- nie służy do celów diagnostycznych, tylko do sklasyfikowania użyteczności stron, aplikacji czy testowanych środowisk.

„ (…) Ankieta oparta jest o skalę Likerta, składa się z 10 pytań wraz z 5 odpowiedziami:

Pytania:

1. Będę często korzystał z systemu.
2. System jest niepotrzebnie skomplikowany.
3. System jest łatwy w użyciu.
4. Będę potrzebował wparcia technicznego, aby korzystać z systemu.
5. Różne funkcje systemu są łatwo dostępne.
6. W systemie jest zbyt wiele niespójności.
7. Większość osób będzie w stanie opanować system bardzo szybko.
8. System jest kłopotliwy w użyciu.
9. Czuję się bardzo pewnie korzystając z systemu.
10. Musiałem opanować wiele rzeczy przed rozpoczęciem pracy z systemem.

Odpowiedzi:

- Zdecydowanie nie zgadzam się.
- Raczej nie zgadam się.
- Nie mam zdania.
- Raczej zgadzam się.
- Zdecydowanie zgadzam się. (…)” [13, 14, 15, 16]

Wyznaczenie wskaźnika ankiety SUS odbywa się w trzech krokach. Wynik jest prezentowany w punktach w skali od 0 do 100.

Krok 1 – przyporządkowanie wartości.

Należy przyporządkować do każdej z odpowiedzi wartości od 0 do 4 punktów, zgodnie z tabelą 1.

Tabela 1. Przyporządkowanie wartości punktowej do odpowiedzi [13, 14, 15]

Odpowiedzi	Numery pytań	
	1, 3, 5, 7, 9	2, 4, 6, 8, 10
Zdecydowanie nie zgadzam się	0	4
Raczej nie zgadam się	1	3
Nie mam zdania	2	2
Raczej zgadzam się	3	1
Zdecydowanie zgadzam się	4	0

Krok 2 – obliczanie wartości SUS.

Kolejnym krokiem jest zsumowanie otrzymanych wartości punktowych, pomnożenie ich przez 2,5 oraz wyliczenie średniej. Otrzymany wynik powinien znajdować się w przedziale od 0 do 100 [13, 14, 15].

Krok 3 - interpretacja wyników.

Wynik SUS, który znajduje się powyżej 68 punktów, uznaje się za wartość powyżej średniej. Poszczególne przedziały ocen zostały przedstawione w tabeli 2.

Tabela 2. Przedział wyników SUS [15]

Wynik SUS	Stopień
> 80,3	A
68 – 80,3	B
68	C
51 – 68	D
< 51	F

Otrzymane wyniki należy postrzegać jako szybkie uzupełnienie testów. Skala użyteczności systemu nie może zastąpić pracy badawczej przeprowadzanej przez użytkowników.

6. Problem badawczy i plan badań

Zagadnieniem poruszonym w pracy jest problem projektowania i narzędzia wspomagające budowę graficznego interfejsu użytkownika aplikacji webowych. Każde z dostępnych rozwiązań ofertuje różny zestaw możliwości, który może wpłynąć na efekt końcowy projektu. Przeprowadzając analizę porównawczą można dokonać weryfikacji przydatności zestawu cech oferowanych przez wybrane narzędzia i dokonać najlepszego wyboru.

W pracy została postawiona następująca teza badawcza:

Narzędzie Adobe XD wspomagające projektowanie interfejsu użytkownika aplikacji webowych jest lepsze od narzędzia Figma w takich samych warunkach.

6.1. Przebieg badań

Na podstawie przestawionej tezy badawczej zostały przeprowadzone badania literaturowe, które obejmowały:

- źródła naukowe i internetowe obejmujące problem projektowania graficznego interfejsu użytkownika za pomocą dedykowanych rozwiązań oraz programów dostępnych na rynku,
- dobór kryteriów selekcji dla narzędzi do projektowania interfejsów.

Doświadczenie polegało na zaprojektowaniu interfejsu przez wybraną grupę osób. Celem zadania było uchwycenie efektywności narzędzi podczas tworzenia wizualnej strony systemu. Pierwszym kryterium, które zostało wzięte pod uwagę jest całkowity czas budowy projektu graficznego interfejsu użytkownika systemu. Czas zapoznania się z przydzielonym narzędziem nie był brany pod uwagę. Przed rozpoczęciem badań została stworzona dokumentacja, która była podstawą do stworzenia projektu systemu. Wytyczne zostały przekazane wszystkim osobom biorącym udział w doświadczeniu. Każde z narzędzi było testowane przez osiem osób. Po skończeniu pracy z narzędziem, wszyscy uczestnicy otrzymali do wypełnienia dwie ankiety. Pierwszą zawierającą pytania na temat interfejsu, komponentów i stabilności danego narzędzia oraz drugą na temat oceny użyteczności systemu. W kolejnych rozdziałach zostały przedstawione wyniki badań oraz ankiet wraz z pełną specyfikacją projektu.

Scenariusz realizacji badań był następujący:

- określenie kryteriów analizy porównawczej dla wybranych narzędzi do projektowania interfejsów,
- dobór kryteriów dotyczących parametrów technicznych,
- przypisanie procentowych wag do wybranych kryteriów,
- prezentacja uzyskanych czasów tworzenia interfejsu przez grupę badawczą,
- przedstawienie wyników ankiet,
- normalizacja danych,
- przypisanie wartości punktowych wybranym cechom narzędzi,
- uśrednienie uzyskanych wyników,
- obliczenie wskaźnika oceny użyteczności systemu – SUS,
- ocena wyników.

6.2. Charakterystyka grupy badawczej

Grupa badawcza została złożona z szesnastu osób w wieku od 25 do 28 lat oraz podzielona na dwie podgrupy. Aby można było uzyskać prawdziwe oraz wiarygodne wyniki wybrano grupę złożoną ze studentów ostatniego roku uczelni wyższej na kierunku Informatyka. Ze względu na zbyt małą liczbę kobiet, do badania została wybrana grupa składającą się tylko z mężczyzn. Do każdej z podgrup zostało przypisane jedno z narzędzi. Zadaniem każdego z uczestników badania było stworzenie projektu graficznego interfejsu użytkownika wraz ze zmierzeniem całkowitego czasu budowy, wykluczając czas potrzebny do zapoznania się z przydzielonym narzędziem. Uczestnicy nie znali przydzielonego im zadania przed rozpoczęciem badań.

6.3. Specyfikacja projektu

Do weryfikacji efektywności narzędzi została stworzona dokumentacja wraz z wymaganiami. Zadanie obejmowało zaprojektowanie prostej aplikacji webowej, dzięki której zalogowany użytkownik będzie miał możliwość stworzenia własnej książki telefonicznej z możliwością dodawania oraz usuwania numerów. Zadanie zostało zaprojektowane pod kątem użycia różnych komponentów dostępnych w narzędziach.

Wymagania dotyczące aplikacji:

- każdy nowy użytkownik, chcący skorzystać z aplikacji musi przejść proces rejestracji, podając login, hasło oraz adres e-mail;
- użytkownik loguje się do systemu za pomocą danych podanych podczas rejestracji za pośrednictwem panelu logowania;
- użytkownik posiada możliwość odzyskania swojego loginu albo hasła, którego zapomniał;
- po zalogowaniu użytkownik może dodawać, usuwać lub edytować już wprowadzone dane do książki telefonicznej;
- po zakończonej pracy użytkownik ma możliwość wylogowania z systemu.

Wymagania projektu:

- projekt interfejsu powinien być przejrzysty, intuicyjny oraz funkcjonalny;
- projektant powinien wykorzystać jak najwięcej dostępnych w narzędziu komponentów.

6.4. Kryteria doboru oprogramowania

Ważnym elementem doboru parametrów porównawczych są czynniki wpływające na odpowiedni wybór narzędzia do projektowania graficznych interfejsów użytkownika aplikacji

webowych. Jednym z takich czynników jest sam projekt, drugim osoba go tworząca. Zdefiniowanie odpowiednich wartości punktowych tzw. wag dla poszczególnych parametrów pozwoli na uniknięcie błędnych wyników badań. Również podzielenie parametrów na dwie grupy: techniczne oraz odnoszące się do procesu projektowania pozwoli uzyskać dokładniejsze wyniki. Parametry techniczne wraz z wartościami punktowymi, zostały przedstawione w tabeli 3.

Tabela 3. Techniczne kryteria oprogramowania

Kryterium (waga w %)	Wskaźnik	Wartość punktowa
Kompatybilność z systemami operacyjnymi (30%)	Wszystkie	1 pkt.
	Windows	0,5 pkt.
	Mac OS	0,5 pkt.
Projektowanie na różne platformy (20%)	Wszystkie	1 pkt.
	Android	0,25 pkt.
	iOS	0,25 pkt.
	Web	0,25 pkt.
Dostępność podstawowych narzędzi (20%)	Duża	0,5 pkt.
	Średnia	0,3 pkt.
	Mała	0,2 pkt.
Eksport projektu (15%)	Wszystkie	1 pkt.
	HTML	0,33 pkt.
	Aplikacja mobilna	0,33 pkt.
	PDF, PNG, JPG	0,34 pkt.
Koszt zakupu (15%)	Wersja płatna	0,4 pkt.
	Wersja bezpłatna	0,6 pkt.

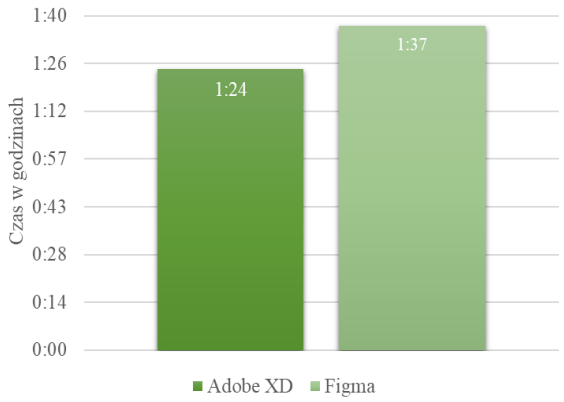
Tabela 4 przedstawia kryteria odnoszące się do procesu projektowania. Czynnikiem o największej wartości punktowej jest czas stworzenia projektu, ponieważ to od niego zależy, czy narzędzie było łatwe w poznaniu i późniejszej obsłudze. Im wartość czasu budowy projektu jest niższa tym wyższa wartość punktowa tego kryterium. Pozostałe czynniki również wpływają na wydajność podczas procesu projektowania, ale w mniejszej skali.

Tabela 4. Kryteria dotyczące tworzenia projektu interfejsu

Kryterium (waga w %)	Wskaźnik	Wartość punktowa
Czas tworzenia projektu (50%)	Najniższy	0,7 pkt.
	Średni	0,2 pkt.
	Najwyższy	0,1 pkt.
Dostępność dokumentacji, poradników (10%)	Duża	0,7 pkt.
	Średnia	0,2 pkt.
	Mała	0,1 pkt.
Poziom trudności tworzenia projektu (dane z ankiety) (10%)	Mały (1-2)	0,7 pkt.
	Średni (2-4)	0,2 pkt.
	Duży (4-5)	0,1 pkt.
Liczba komponentów (dane z ankiety) (10%)	Duża (4-5)	0,7 pkt.
	Średnia (2-4)	0,2 pkt.
	Mała (1-2)	0,1 pkt.
Łatwość poruszania się po interfejsie narzędzia (dane z ankiety) (10%)	Duża (4-5)	0,7 pkt
	Średnia (2-4)	0,2 pkt
	Mała (1-2)	0,1 pkt
Ocena użyteczności (dane z ankiety) (10%)	> 80,3	0,4 pkt.
	68 – 80,3	0,3 pkt.
	68	0,2 pkt.
	51-68	0,1 pkt.
	< 51	0 pkt.

7. Rezultaty eksperymentu

Wynikiem, który został potraktowany jako główny wyznacznik przeprowadzonych badań, w ramach których uczestnicy mieli za zadanie stworzyć projekt graficznego interfejsu użytkownika był czas jego tworzenia (Rys.1).



Rys. 1. Średni czas tworzenia projektu interfejsu

Tabela 5. Uśrednione wyniki ankiet

Treść pytania (skala)	Narzędzie do prototypowania	
	Adobe XD	Figma
Trudność projektowania graficznego interfejsu (1-5)	1,50	2,25
Liczba komponentów (1-5)	4,25	3,88
Łatwość poruszania się po interfejsie narzędzia (1-5)	4,5	4,38
Ankieta oceny użyteczności (SUS)	90,9	87,19

Do każdego z kryterium technicznego przyznane zostały wartości punktowe zgodnie z danymi zawartymi w tabeli 3. Punkty zostały przydzielone na podstawie źródeł internetowych tj.: dokumentacje, poradniki producenta oprogramowania oraz własnych testów narzędzi przeprowadzonych na potrzeby badań. Tabela 6 zawiera ocenę parametrów technicznych wraz z ich średnimi wartościami.

Tabela 6. Analiza narzędzi według wybranych parametrów technicznych

	Narzędzia	
Kryterium	**Adobe XD**	**Figma**
Kompatybilność z systemami operacyjnymi	1	1
Projektowanie na różne platformy	1	1
Dostępność podstawowych narzędzi	0,5	0,3
Eksport projektu	0,34	0,34
Koszt zakupu	0,6	0,6
Wartość średnia	**0,74**	**0,70**

Wyznaczenie wyników porównania kryteriów dotyczących efektywności wiązało się z przypisaniem wartości punktowych do wybranych parametrów, które zostało oparte na ich szczegółowej analizie. Część wyników uzyskano z wypełnionych ankiet po dokonaniu normalizacji. Tabela 7 zawiera ocenę parametrów efektywności wraz z ich średnimi wartościami.

Tabela 7. Analiza narzędzi według wybranych parametrów efektywności

	Narzędzia	
Kryterium	**Adobe XD**	**Figma**
Czas tworzenia projektu	0,6	0,4
Dostępność dokumentacji, poradników	0,7	0,7
Poziom trudności tworzenia projektu	0,7	0,2
Liczba komponentów	0,7	0,2
Łatwość poruszania się po interfejsie narzędzia	0,7	0,7
Ocena użyteczności	0,4	0,4
Wartość średnia	**0,62**	**0,42**

Głównym parametrom, które zostały poddane analizie (tj. technicznym i dotyczącym tworzenia projektu interfejsu) została przypisana taka sama waga. Średnia uzyskanych wyników przedstawi ostateczną wartość punktów uzyskanych przez narzędzia objęte analizą – tabela 8.

Tabela 8. Końcowe zestawienie wyników

	Narzędzia	
Kryterium	**Adobe XD**	**Figma**
Parametry techniczne	0,74	0,70
Parametry efektywności	0,62	0,42
Wartość średnia	**0,68**	**0,56**

Głównym wskaźnikiem, który został wzięty pod uwagę był czas tworzenia projektu graficznego interfejsu użytkownika przez osoby należące do grupy badawczej. Przeprowadzając analizę wykresu, łatwo zauważyć, że różnica pomiędzy średnim czasem tworzenia projektu (Rys. 1) wynosi 13 minut na korzyść narzędzia Adobe XD.

Analiza poszczególnych kryteriów wpływających na efektywność projektowania graficznego interfejsu użytkownika, umożliwiła porównanie wybranych narzędzi na poziomie wydajności. Na podstawie wyników w tabeli 7, można dostrzec, że narzędzie Adobe XD uzyskało wyższą liczbę punktów. Głównym czynnikiem, który wpłynął na średni wynik był czas tworzenia projektu. Różnice w cząstkowych wynikach są również widoczne w przypadku dwóch kryteriów: poziom trudności tworzenia projektu oraz liczba komponentów, w których narzędzie Figma uzyskało mniejszą liczbę punktów.

Analiza danych z tabeli 6, w której zostały przedstawione cząstkowe wyniki parametrów technicznych dla wybranych narzędzi, pozwala zaobserwować wyrównane wyniki. Narzędzie Adobe XD uzyskało minimalnie większą liczbę punktów. Parametrem, który zaważył o niższym wyniku dla narzędzia Figma była dostępność podstawowych narzędzi.

8. Wnioski

Przeprowadzone badania oraz uzyskane wyniki pozwoliły na zrealizowanie analizy porównawczej dwóch narzędzi do projektowania graficznego interfejsu użytkownika. Do procesu porównania została użyta metoda przypisania wag do wybranych kryteriów.

Końcowy rezultat badań zawarty w tabeli 8 zawiera wyniki dla parametrów technicznych oraz parametrów efektywności wybranych narzędzi. Dzięki uśrednieniu wyników, można stwierdzić, że program Adobe XD jest lepszym narzędziem do projektowania graficznego interfejsu użytkownika od programu Figma. Z uzyskanej analizy i wyników można stwierdzić, że postawiona teza jest prawdziwa.

Przeprowadzona analiza nie może być fundamentem do wybrania odpowiedniego narzędzia do projektowania, ponieważ każdy projekt ma inne wymagania. Dobór odpowiedniego narzędzia wspomagającego projektowanie graficznego interfejsu użytkownika korzystnie wpływa na etap projektowania oraz koszt jego stworzenia.

Literatura

[1] Jaszkiewicz A.: Inżynieria oprogramowania. Helion, Gliwice, 1997.

[2] Sacha K.: Inżynieria oprogramowania. Wydawnictwo naukowe PWN, 2010.

[3] Miłosz M.: Ergonomia systemów informatycznych. Politechnika Lubelska, 2014.

[4] Spolsky J.: Projektowanie interfejsu użytkownika: poradnik dla programistów. Mikom 2001.

[5] https://docplayer.pl/3268459-Projektowanie-interfejsu-uzytkownika-1-jaroslaw-kuchta-projektowanie-aplikacji-internetowych.html [03.2019].

[6] Lipski S., Miłosz M.: Analiza porównawcza narzędzi do budowy prototypów interfejsów. Journal of Computer Sciences Institute (JCSI), vol. 1 (2016) p. 38-43.

[7] Łasocha A., Miłosz M.: Analiza porównawcza systemów wspomagających prototypowanie interfejsów. Journal of Computer Sciences Institute (JCSI), vol. 4 (2017) p. 122-127.

[8] Kultys B., Miłosz M.: Wykorzystanie User Experience w procesie poprawy GUI aplikacji. Journal of Computer Sciences Institute (JCSI), vol. 1 (2016) p. 25-29.

[9] Adobe XD — materiały szkoleniowe i pomoc techniczna https://helpx.adobe.com/pl/xd/user-guide.html [03.2019].

[10] Figma Features https://www.figma.com/features/ [03.2019].

[11] Panek, T.: Statystyczne metody wielowymiarowej analizy porównawczej. Szkoła Główna Handlowa – Oficyna Wydawnicza, 2009.

[12] Cid-López A., Hornos M., Carrasco R., Herrera-Viedma E., Applying a linguistic multi-criteria decision-making model to the analysis of ICT suppliers' offers. Expert Systems with Applications, 57, 2016, 127-138.

[13] Brooke J.: SUS: A 'quick and dirty' usability scale. In P. Jordan, B. Thomas, & B. Weerdmeester (Eds.), Usability Evaluation in Industry. London, UK: Taylor & Francis.

[14] Brooke J.: SUS: A retrospective. Journal of Usability Studies.

[15] Lewis J. R., Sauro J.: Item Benchmarks for the System Usability Scale. Journal of Usability Studies. May 2018, vol. 13 Issue 3.

[16] Wierusz J.: Skala użyteczności systemu (SUS) https://www.mysurveylab.com/pl/blog/skala-uzytecznosci-systemu-sus/ [03.2019].

[17] System Usability Scale (SUS) https://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html [03.2019].

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 12 (2019) 179-186
Wysłane: 2019-05-07
Przyjęte: 2019-05-19

Analiza możliwości wykorzystania wirtualnej rzeczywistości do badania reakcji na bodźce

Marcin Łukasiak, Kamil Machul, Mateusz Maj*
Politechnika Lubelska, Instytut Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. Badanie miało na celu sprawdzenie możliwości wykorzystania technologii VR w celu wywołania konkretnych emocjiu badanego. Przeprowadzone badania literaturowe wykazały wysokie prawdopodobieństwo uzyskania pomyślnych wyników, jednak dotychczas badania opierane były na pomiarach uzyskanych przy użyciu wyspecjalizowanego, drogiego sprzętu. Niniejsza praca miała wykazać, że uzyskanie podobnych rezultatów jest możliwe również z wykorzystaniem budżetowych rozwiązań. Uzyskane wyniki pozwalają stwierdzić, że nawet korzystając z mało wyszukanych technologii można uzyskać zadowalające rezultaty, które mogą określić wpływ obrazu wyświetlanego w goglach na emocje badanego.

Słowa kluczowe: emocje; VR; EEG; puls

*Autor do korespondencji.

Adres e-mail: mateusz.maj@pollub.edu.pl

Analysis of possibility of virtual reality usage for investigating reaction on given conditions

Marcin Łukasiak, Kamil Machul, Mateusz Maj*
Institute of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract. The study was tend to checking the possibility of using VR technology to invoke specific emotions on the subject. Literature studies have shown a high probability of obtaining successful results, but so far studies have been based on measurements obtained using specialized, expensive equipment. This study was about to show that obtaining similar results is also possible with usage of low budget solutions. Received results allow to conclude even using the less sophisticated technology, satisfactory results can be gained, which can determine the effect of the image displayed in the goggles on the emotions of the subject.

Keywords: emotions; VR; EEG; puls

*Corresponding author.

E-mail address: mateusz.maj@pollub.edu.pl

1. Wprowadzenie

W ostatnich latach wirtualna rzeczywistość rozwinęła się w dużym stopniu. Dzięki szybkiemu rozwojowi technologii komponenty VR stały się szerzej dostępne i coraz częściej trafiają do gospodarstw domowych. Wirtualna rzeczywistość używana jest przede wszystkim do rozrywki, dominującym nurtem jest tu rynek gier. Na ten moment wiele gier posiada wersje dedykowane na zestawy VR, przez co użytkownicy mogą przeżywać gra na całkowicie innej płaszczyźnie. Mogą głębiej zanurzyć się w wirtualny świat.

Jednak VR nie służy tylko do rozrywki, może też mieć zastosowanie naukowe. Możemy badać różne aspekty rzeczywistości używając modeli prezentowanych za pomocą gogli. Niniejszy artykuł stara się odpowiedzieć na pytanie czy istnieje możliwość wykorzystania VR do badań emocji występujących u pacjenta w celu diagnozowania fobii lub profilowania emocji odczuwanych przez badanego. W przypadku upowszechnienia technologii VR możliwym stałoby się wytworzenie uniwersalnych programów-gier diagnostycznych, które można wykorzystać w domu. Jeśli emocje wywoływane przez wirtualną rzeczywistość pokrywają się z tymi towarzyszącymi sytuacji rzeczywistej, gogle VR wraz z odpowiednimi trójwymiarowymi modelami mogłyby posłużyć również do wstępnych badań predyspozycji do wykonywania zawodu. Dzięki temu można określić, czy potencjalny pracownik byłby zdolny do wykonywania robót na wysokości lub czy jego zdolność odczuwania empatii do zwierząt pozwala mu na pracę w schronisku. Pozwoliłoby to na uniknięcie prób podjęcia zawodu w przypadku oczywistych przeciwwskazań psychologicznych.

2. Materiały i metody

W celu zgłębienia tematu zostały wykonane badania literaturowe, który pozwoliły wyłonić metody, za pomocą których przeprowadzaono badania.

2.1. Obecność i strach w VR

W pracy wykonano eksperyment na 22 kobietach ze stwierdzonym lękiem przed pająkami (arachnofobia). Modele prezentowane były za pomocą technologii Powerwall (Rys. 1). Poziomy strachu oraz obecności były określane na podstawie badania EEG. Praca wykazała korelację pomiędzy poziomem

poczucia obecności (immersji) w wirtualnej rzeczywistości a poziomem odczuwanego strachu. Wszelkie przerwy w poczuciu obecności skutkowały zmniejszeniem poziomu lęku odczuwanego przez badane [1].

Rys. 1. Zdjęcie aparatury wykorzystanej przez badaczy

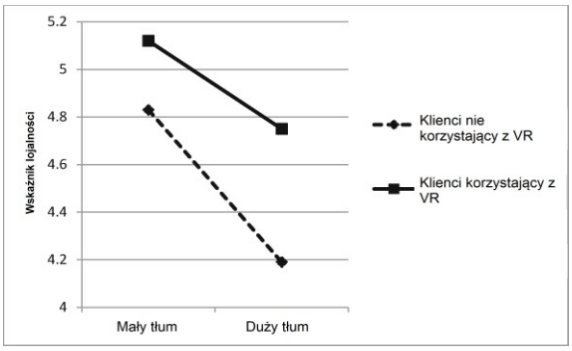
2.2. Kondycjonowanie sygnału strachu

Praca traktująca o kondycjonowaniu strachu jednej z najczęściej wykorzystywanej procedur leczenia zaburzeń lękowych. Badanie zostało przeprowadzone na grupie 50 kobiet - 25 o podwyższonym stopniu strachu przed pająkami i 25 nie wykazujących lęku ponad normę. Pająki były prezentowane w środowisku VR na scenie symulującej pokój biurowy (Rys. 2). Reakcja była oceniana na podstawie przewodnictwa skóry oraz stopnia poczucia zaskoczenia u badanego. Badania wykazały wyższe poczucie strachu u osób, które deklarowały silniejszy lęk przed pająkami. Wnioskiem postawionym przez badaczy było stwierdzenie, że VR stwarza duże możliwości do sprawdzania i klasyfikowania procedur warunkowania strachu [2].

Rys. 2. Zrzut ekranu z aplikacji użytej przez badaczy

2.3. Odpoczynek od tłumu badanie w centrumhandlowym

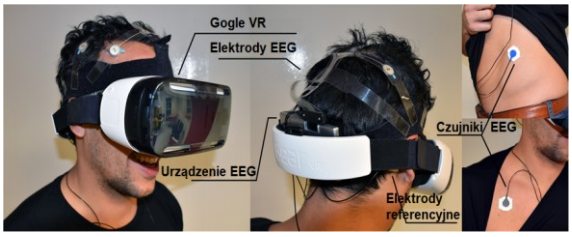
Tematem badania był wpływ tłumu w czasie zakupów w centrum handlowym na samopoczucie badanego. Badacze wyszli z założenia, że tłok w galerii wpływa na odczuwaną satysfakcję z dokonywanych zakupów. Celem badania było sprawdzenie, czy umożliwienie klientom skorzystania ze stanowiska VR, na którym prezentowana była scena przejażdzki na saniach zaprzężonych w renifery, wpłynie na satysfakcję klientów. Badanie zostało przeprowadzone w centrum handlowym „Les Bastions" w belgijskim Tournai. Wyniki wykazały, że skorzystanie ze stanowisko miało bardzo wysoki wpływ na odczuwaną satysfakcję, zadowolenie ze skorzystania z centrum oraz zwiększało szansę na kolejną wizytę klienta (Rys. 3) [3].



Rys. 3. Wykres porównujący deklaracje lojalności względem centrum handlowego

2.4. Rozpoznanie emocji na podstawie eeg i tętna

Artykuł opisuje eksperyment który miał na celu obserwację zmiany stanu emocjonalnego u osób umieszczonych w wirtualnym świecie. Badaniu została poddana grupa 60 osób, dla której przygotowano 4 pomieszczenia zaprezentowane w VR. Reakcja badanych na prezentowane otoczenie została zmierzony przy użyciu EEG i EKG (Rys. 4). Badania potwierdziły możliwość stymulacji scen wywołujących emocje przy użyciu wirtualnej rzeczywistości [4].



Rys. 4. Urządzenia zastosowane przez badaczy

2.5. Zrozumienie VR

Artykuł opisuje różne definicje wirtualnej rzeczywistości przedstawione przez poszczególnych autorów. Każdy z nich na przestrzeni lat miał inne podejście do przedstawionego tematu. Wraz z upływem czasu poszczególne definicje zostały skonfrontowane z obecnymi warunkami panującymi na rynku. Wraz z łatwiejszym dostępem do technologii oraz jej gwałtownym rozwojem skierowano swoje starania na zwiększenie poziomu immersji, czy to przez rozwinięcie płaszczyzny dźwiękowej, czy udoskonalenie kontrolerów do interakcji w symulacji. Definicja VR jest ciągle otwartą kwestią, która wciąż jest badana i dokładane są wszelkie starania w celu jej zrozumienia [5].

2.6. Rytmserca w rozpozaniu normalnej i patologicznejadaptacji do wysiłku mentalnego

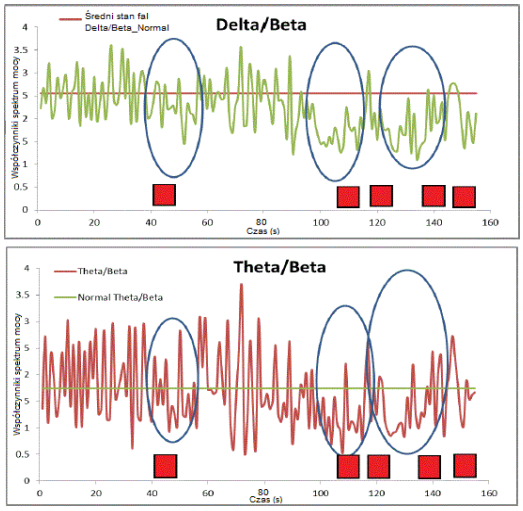
Artykuł objaśniający mechanizm zmian tętna w przypadkach typowych oraz u osób z zaburzeniami układu nerwowego. Praca wskazuje na minimalne odchylenia odczytów pulsu u dzieci z drobnymi zaburzeniami rozwojowymi. Wnioskami, które mogą być istotne dla niniejszej pracy są obserwacje jedynie minimalnych różnic w odczytach pulsu w przypadku sytuacji stresowych [6].

2.7. Rozpoznowanie emocji za pomocą eeg podczas wyświetlania filmu

W powyższej publikacji zostaje poruszona kwestia rozpoznawania emocji przy pomocy EEG. Badanie zostało przeprowadzone na grupie 110 studentów (57 mężczyzn i 53 kobiety). Badacze podjęli próbę zakwalifikowania trzech emocji - złości, strachu i zaskoczenia odczuwanego przez badanych. Eksperymentatorzy doszli do wniosku, że badanie emocji za pomocą EEG daje możliwość odczytu emocji; za pomocą zastosowania trzech elementów badawczych - opasek częstotliowściowych,asymetriimózgowej oraz funkcjonalności spójności EEG - można uzyskać precyzję na poziomie 68.1% dla gniewu, 61.5% dla strachu oraz 69.2% dla zaskoczenia [7].

2.8. Algorytm klasyfikacji emocji zapomocą przenośnego EEG

Praca opisująca utworzenie algorytmu, który przetwarzając strumień pomiarów EEG byłby w stanie przewidywać wystąpienie określonych emocji. Badacze postulują, że wykrycie uczucia strachu u osób pracujących jako na przykład zawodowi kierowcy, pozwoliłoby uniknąć sytuacji stwarzających zagrożenie w ruchu drogowym. Badacze oceniali skuteczność algorytmu na podstawie pomiarów w czasie wyświetlania filmu grozy obiektom badawczym. Na podstawie pomiaru fal Beta, Delta oraz Theta (Rys. 5) udało im się uzyskać ostrzeżenia o wystąpieniu lęku w momencie, gdy tylko badany zaczynał odczuwać strach w reakcji na wyświetlany obraz [8].



Rys. 5. Wykresy prezentujące zmiany w falach mózgowych badanych w czasie prezentacji filmu

2.9. Wzory fal mózgowych podczas stresu

W niniejszej pracy badacze obrali sobie za cel uzyskanie wzorów przebiegu spektrum mocy fal mózgowych alfa i beta w reakcji na stres. Badanie zostało przeprowadzone na 86 ochotnikach, którzy najpierw odpowiadali w ankietach na pytania dotyczące czynników powodujących u nich lęk. Autorzy pracy osobno traktowali odczyty z lewej i prawej półkuli mózgu (Rys. 6). Wyniki wskazały na to, że przebiegi aktywności fal mózgowych alfa i beta są uniwersalne w sytuacji stresowej niezależnie od badanego człowieka [9].

Spearman's Rho	Alpha_R	Alpha_L	Beta_R	Beta_L
Index 1	-1.77	0.286	-0.207	0.195
Index 2	0.50	-0.40	0.417	0.020

Rys. 6. Fragment tabeli prezentującej wyniki pomiarów

2.10. Fizjologia emocji

Książka wydana w roku 1986 stanowiąca kompletne studium na temat odczuwania emocji i psychologii emocji. Publikacja była przydatna do interpretacji uzyskanych wyników [10].

3. Zestaw badawczy

W celu przeprowadzenie badań zostało wykonane stanowisko badawcze, które składa się z kilku elementów (Rys. 7) - każdy odpowiedzialny za inny zakres badanych wskaźników.

Rys. 7. Zdjęcie stanowiska badawczego

Skład zestawu:

- EEG - MindWave HeadSet,
- Mi Band 2,
- Google Cardboard i Google VR Box 3D,
- Kinect,
- PC.

3.1. EEG - MindWave HeadSet

Urządzenie umożliwia bezpieczne mierzenie oraz bezprzewodowe przesyłanie widma fal m.in. alpha czy beta poprzez Bluetooth do wybranego komputera PC. Urządzenie monitoruje poziom aktywności mózgu, a dostarczone przez niego dane pozwalają stwierdzić jak mózg reaguje na zadane mu bodźce. W budowie zestawu należy rozróżnić dwa istotne elementy: spinacz mocowany do ucha oraz ramię z sensorem przykładanym do czoła zaraz nad linią oczu. Dzięki udostępnionemu kodu źródłowemu możliwe jest przystosowanie urządzania oraz postaci zwróconych pomiarów do własnych potrzeb.

3.2. Pomiar pulsu - Mi Band 2

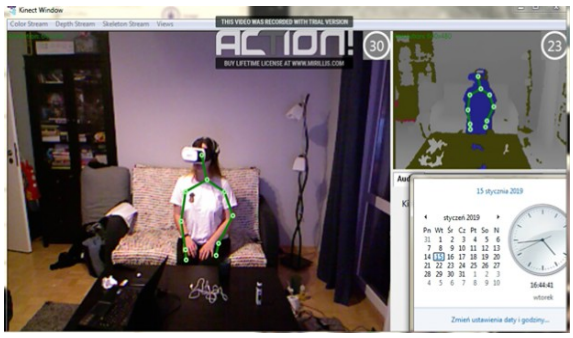
Pomiar pulsu przeprowadzany jest za pomocą opaski Xiaomi Mi Band 2. Opaska łączy się przez Bluetooth z urządzeniem mobilnym, gdzie przesyłane są zebrane przez nią dane zebran, a następnie interpretowane na różnych płaszczyznach, np. analiza jakości snu. Liczba zastosowań jest szeroka a udostępniony kod źródłowy pozwala tę liczbę zwiększyć. Opaska komunikuje się z aplikacją zbiorczą umieszczoną na komputerze klasy PC poprzez autorską aplikację na urządzeniu mobilnym. W maju 2018 roku zadebiutowała kolejna wersja opaski w wersji 3, jednak kod źródłowy nie został udostępniony w momencie pisania artykułu.

3.3. Zestaw VR Google VR Box 3D

W badaniach został użyty zestaw z goglami do smartfona rdzeń zestawu stanowi smartfon umieszczony w obudowie, wydajność całego zestawu determinują parametry telefonu, który zapewnia ekran i moc obliczeniową. Zestaw ten nie dorównuje wydajnością dedykowanym headsetom, ale dzięki prostej konstrukcji i użyciu smarfona stanowi tańszą alternatywę i niejednokrotnie jest pierwszym kontaktem z technologią VR przed przejściem na bardziej zaawansowane urządzenia.

3.4. Urządzenie do badania reakcji ruchowej Kinect

Jest to ruchowy kontroler wyprodukowany przez firmę Microsoft, pozwalający na interakcję z konsolą za pomocą ruchów i gestów. Jego kampanii reklamowej przewodziło hasło „Ty jesteś kontrolerem". Początkowo urządzanie było popularne pośród posiadaczy konsol jednakże z powodu spadku zainteresowania urządzeniem oraz małej ilości aplikacji wykorzystujących kontrolę ruchową Microsoft zaprzestał produkcji urządzenia w październiku 2018 roku. Mimo braku wsparcia ciągle jest wykorzystywany m.in. w projektach badawczych obejmujących zagadnienia związane z ruchem. Na potrzeby badania Kinect został skonfigurowany do pracy z komputerem klasy PC (Rys. 8).



Rys. 8. Zrzut ekranu z nagrania wykonanego kinectem w czasie badania

3.5. Komputer klasy PC

Aplikacja zbiorcza gromadząca dane z badań przesłanych z MindWave BLE 4.0 EEG, Mi Band 2 oraz Kinect umożliwiająca ich dalszą analizę.

4. Scenariusze badawcze

W celu zweryfikowania poprawności postawionej tezy grupa kontrolna przy użyciu zestawu badawczego została poddana wcześniej przygotowanym badaniom w skład, których wchodziły trzy scenariusze.Podczas tworzenia scenariuszy zwrócono szczególną uwagę na właściwy wybór motywów przewodnich. Zakres emocji jakie może odczuwać badany podczas oglądania sceny powinien być możliwie szeroki - od czegoś przyjemnego, do wzbudzającego strach i w pewnym stopniu różnić się między sobą.

4.1. Pająk

Powyższy temat został wybrany po przeprowadzeniu badania ankietowegow którym badani mieli za zadanie wybrać jak duży dyskomfort odczuwają na zadane bodźce. Spośród wielu opcji m.in. pająki, węże, owady, zamknięte pomieszczenia największy dyskomfort sprawiały właśnie pająki. Arachnofobia może potęgować uczucie lęku przed pająkami wśród badanych, dlatego zachowano szczególną ostrożność w trosce o zdrowie badanych podczas badania przy użyciu prawdziwego stworzenia. Celem tego scenariusza jest wywołanie u badanego lęku, zaskoczenia i niepokoju. Na potrzeby pracy została stworzona aplikacja, w której prezentowana jest scena składająca się ze stołu, przy którym siedzi badany, a na którym znajdują się pająki. Pająki wykonują ruchy zbliżając się ku badanemu mając wywołać u niego reakcje. Kolejnym etapem badania jest wyświetlenie materiału z serwisu YouTube, w którym znajdują się nagrania pająków w różnych sytuacjach np. pająk poruszający się po ręce lub znajdujący się na ścianie pomieszczenia.

4.2. Publiczne wystąpienie

Aby odtworzyć sytuację w wirtualnej rzeczywistości wykorzystano jedną z aplikacji ze sklepu Google Play, w której badany znajduje się w sali pełnej słuchaczy i w zależności od wykonanych działań audiencja reaguje pozytywnymi lub negatywnymi szeptami. Generuje to u badanego uczucie niepokoju, zagrożenia oraz stres, który

niejednokrotnie towarzyszy publicznym wystąpieniom. Film, który był prezentowany badanym pochodził z serwisu YouTube i traktował o fobii społecznej, zaprezentowane są tam różne sceny, które pogłębiają uczucie niepokoju i odosobnienia. W rzeczywistości badany miał za zadanie wygłosić referat przed grupą ludzi w zamkniętym pomieszczeniu. Widownia miała za zadanie pogłębić stres u badanego wymieniając uwagi między sobą oraz ciągle obserwać badanego.

4.3. Piękny krajobraz

W ostatnim z użytych scenariuszy wykorzystany został motyw krajobrazów, które odprężają, uspokajają i powodują radość. Ponownie została przeprowadzona ankieta, z której został wyłoniony najbardziej przyjemny widok spośród m.in. krajobrazów górskich, dzieł malarskich oraz widoków miejskich. Pośród ankietowanych opcją powodującą najbardziej przyjemne emocje okazały się sceny podwodne.

Badanemu został przedstawiony film z serwisu YouTube gdzie pokazywane są ujęcia z życia pod wodą, zarówno rośliny jak i podwodne stworzenia. Ze względów logistycznych niemożliwe było odtworzenie danej scenerii w warunkach rzeczywistych. Następnie film został wyświetlony za pomocą gogli VR, w celu porównania reakcji badanego.

5. Metoda badawcza

W badaniach wzięło udział 5 osób w wieku 22-26 lat: 1 kobieta i 4 mężczyzn (Rys. 9). W badaniach ankietowych zostały zebrane informacje na temat czego badani boją się najbardziej oraz co sprawia, że się relaksują.

Rys. 9. Zdjęcie osoby badanej w oprzyrządowaniu

5.1. Badania porównawcze

Badania porównawcze polegały na przygotowaniu scenariuszy nie wykorzystując technologii VR, przedstawianie je badanemu oraz zebranie wyników badań. Ten cel osiągnięto na dwa sposoby. Pierwszym jest odtworzenie badanemu sytuacji w realnym świecie co stanowi najbardziej wartościowy wynik badań w kwestii zestawiania wyników. Drugim sposobem jest przedstawienie danego scenariusza na ekranie (bez użycia VR) razem z przeprowadzeniem krótkiej ankiety na dany temat (Tabela 1).

Tabela 1. Tabela prezentująca wyniki ankiety dotyczącej lęków

Typ sceny	Ocena
Pająk	7
Owady	4,6
Psy	3,2
Tłum ludzi	5,4
Wilki	3,8
Wysokość	4,8
Wystąpienie publiczne	5,4
Zamknięte pomieszczenia	5,2
Węże	3,2

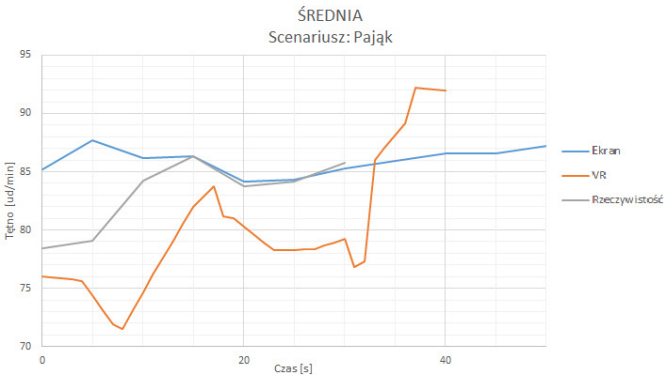
6. Wyniki badań

6.1. Pająk

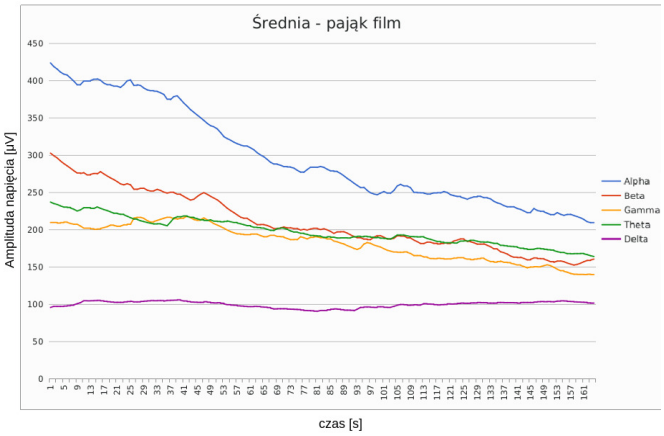
Po zestawieniu średnich wykresów każdego z typów scenariusza można zauważyć, że początkowa wartość pulsu była najniższa w badaniu w wirtualnej rzeczywistości a najwyższa w badaniu z użyciem ekranu (Rys. 10). Po chwili wykres badania rzeczywistego i wirtualnej rzeczywistości wzrasta. w tym momencie badani mieli styczność z obiektami pająków kolejno prawdziwym i wirtualnym. Można więc wysnuć wniosek, że w tym przypadku wirtualna rzeczywistość może zastąpić świat realny - wykresy posiadają taką samą tendencję zachowując niemal takie same wartości referencyjne.

Zgodnie z oczekiwaniami wszystkie scenariusze wywołały spadek fal alfa (Rys. 11, 12, 13) fal odpowiedzialnych za odprężenie. Fale beta, które mogą świadczyć o chęci podjęcia akcji, wzrosły zwłaszcza w przypadku prawdziwego pająka, jednak odczyt dla VR nie różnił się znacznie od wartości w rzeczywistości. Fale delta, gamma oraz beta dla ogółu badanych nie wykazywały jednoznacznej tendencji wzrostu lub spadku we wszystkich scenariuszach.

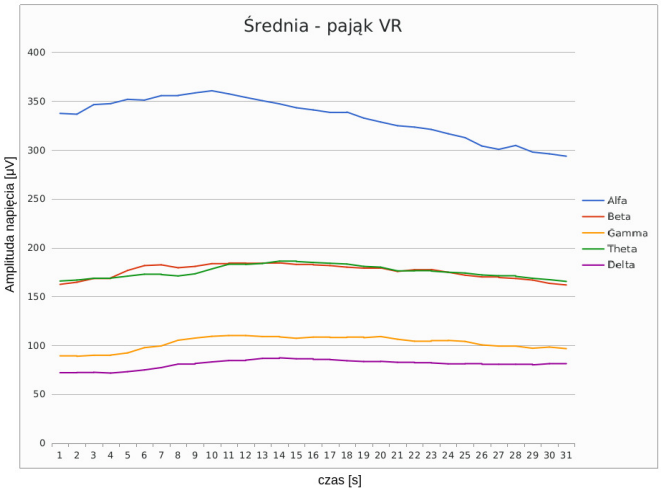
Badanie z wykorzystaniem Kinect oraz obserwacji przeprowadzonej przez badaczy nie wykazywały wspólnej tendencji do wykonywania ruchów. Jedynie osoba cierpiąca na arachnofobię reagowała w żywiołowy sposób, pozostali badani wykonywali nieznaczne ruchy.



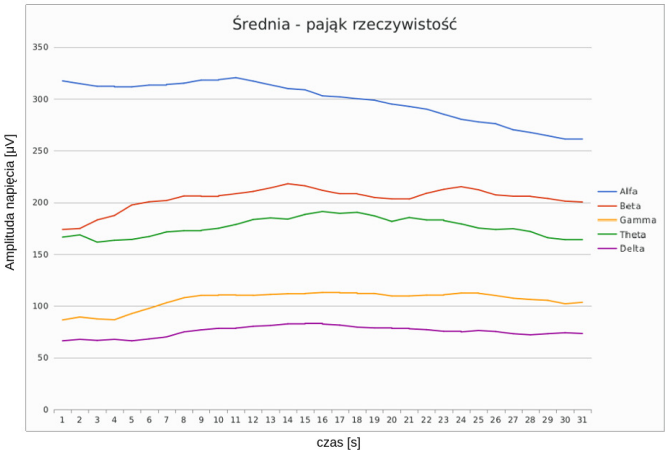
Rys. 10. Wykres średniej wartości pulsu u wszystkich badanych w scenariuszach tego samego typu - pająk



Rys. 11. Wykres średniej wartości fal mózgowych u wszystkich badanych w scenariuszu pająk - film wyświetlony na ekranie



Rys. 12. Wykres średniej wartości fal mózgowych u wszystkich badanych w scenariuszu pająk - animacja 3D w wirtualnej rzeczywistości



Rys. 13. Wykres średniej wartości fal mózgowych u wszystkich badanych w scenariuszu pająk - żywy okaz

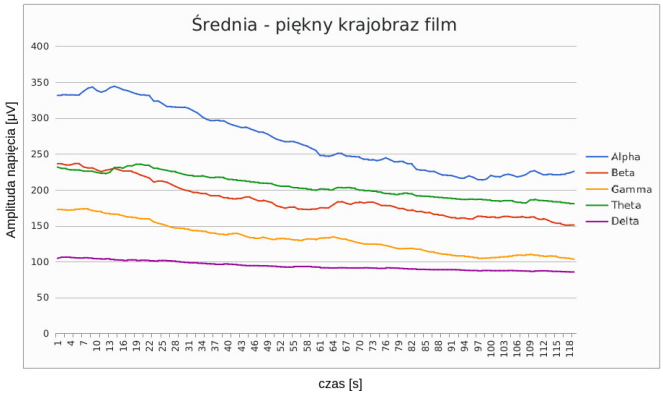
6.2. Publiczne wystąpienie

Podczas badania wirtualną rzeczywistością do połowy badania można zauważyć że puls utrzymuje się na stałym poziomie. Natomiast przy scenariuszu z ekranem spada a następnie gwałtownie wzrasta (Rys. 14), co można wyjaśnić większym zaangażowaniem badanego w przedstawiony mu materiał. Wraz z postępem badania można stwierdzić, że badanie VR lepiej oddziaływało na badanego. Puls badanego zwiększa się w przeciwieństwie do badania materiałem filmowym. Badanie VR wykreowało mocniejsze bodźce, które poskutkowało u badanych zwiększoną wartością tętna.

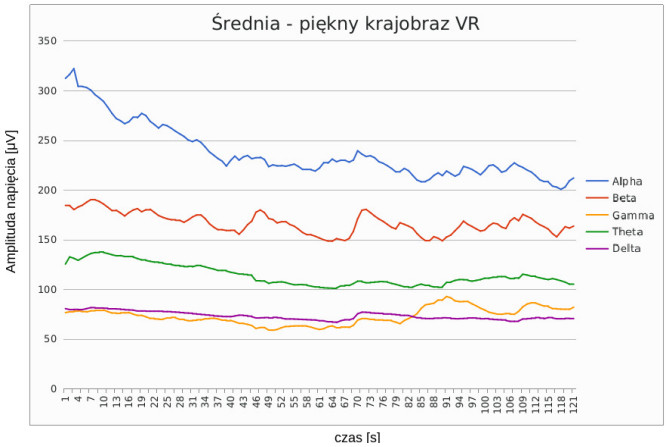
Pomiar EEG wskazuje, że prezentowany film był znacznie bardziej skuteczny przy wyświetlaniu z wykorzystaniem VR, niż w konwencjonalny sposób(Rys. 15, 16) Fale alfa oraz beta wzrastały w podobnych momentach filmu w przypadku wirtualnej rzeczywistości, co może świadczyć o wzroście odprężenia w sytuacjach ciekawych dla badanego - nie można powiedzieć tego samego o pomiarach przy wyświetlaniu na ekranie. Podobnie fale gamma - uzyskano wzrost przy prezentacji w goglach, co nie miało miejsca przy standardowym sposobie wyświetlania.

Kinect zarejestrował podążanie wzrokiem za pojawiającymi się zwierzętami przy prezentacji w VR. Wyświetlaniu na ekranie nie towarzyszył żaden ruch u badanych.

Rys. 14. Wykres średniej wartości pulsu u wszystkich badanych w scenariuszach tego samego typu - piękny krajobraz



Rys. 15. Wykres średniej wartości fal mózgowych u wszystkich badanych w scenariuszu piękny krajobraz - film wyświetlony na ekranie



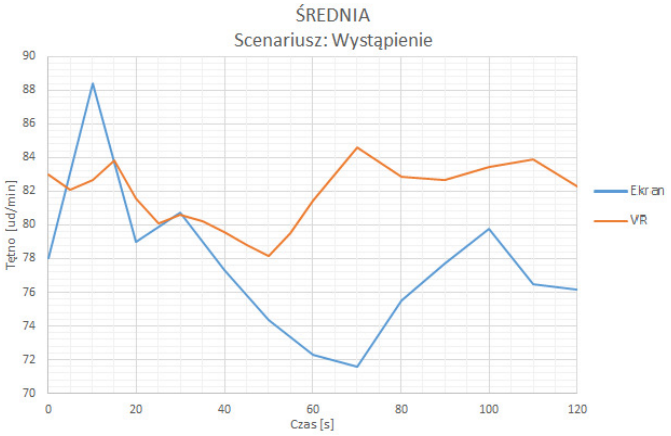
Rys. 16. Wykres średniej wartości fal mózgowych u wszystkich badanych w scenariuszu piękny krajobraz - film wyświetlony w wirtualnej rzeczywistości

6.3. Wystąpienie publiczne

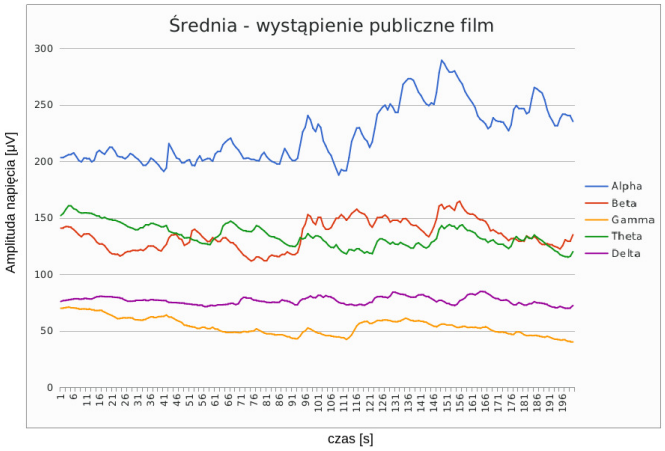
Wartości pulsu w pierwszych sekundach badania są wyższe w konfiguracji z filmem niż w wirtualnej rzeczywistości. Początkowo film ustanowił wysoki poziom immersji, za co odpowiadała jego ciekawa formuła. Jednak wraz ze upływem czasu ta formuła się wyczerpuje co wskazuje spadek wartości na wykresie (Rys. 17) w przypadku wirtualnej rzeczywistości odczyty utrzymują się, a w momencie zwiększenia grupy publiczności w scenariuszu - zwiększają się. w końcowej fazie puls nieznacznie spada co może być związane z znudzeniem badanego. Badanie w wirtualnej rzeczywistości przyniosło lepsze wyniki i jest trafniejszym wyborem jako substytut rzeczywistości w kontekście wykorzystania jej w dalszych badaniach.

W badaniu EEG nie udało się uzyskać zamierzonego efektu - wzrost fal alfa, nikły wzrost fal beta i theta (Rys. 18, 19) w późniejszej fazie badania świadczy raczej o znudzeniu filmem, niż wywołaniem jakiegokolwiek dyskomfortu. Znacznie lepsze efekty dało badanie przy użyciu wirtualnej rzeczywistości - w tym scenariuszu amplituda fal alfa zmalała, zaś fale beta jednoznacznie wzrosły - można na tej podstawie postulować, że badani odczuli faktyczny dyskomfort związany z badaniem.

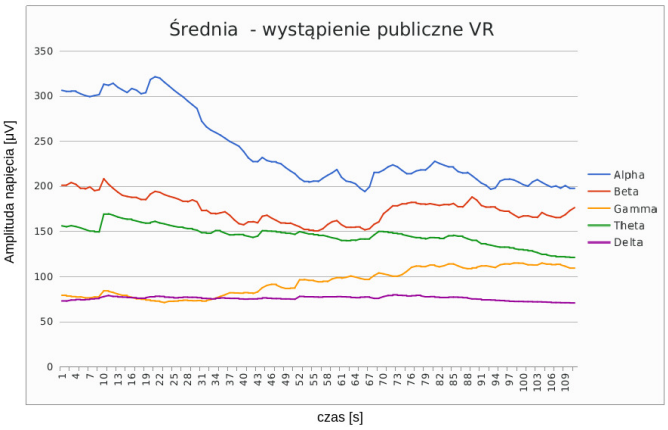
Obraz rejestrowany przez Kinect nie przedstawiał żadnego schematu ruchu w przypadku filmu wyświetlanego na ekranie. Aplikacja wyświetlana w goglach zmusiła badanych do podążania za źródłem dźwięku, co może świadczyć o wyższej imersji.



Rys. 17. Wykres średniej wartości pulsu u wszystkich badanych w scenariuszach tego samego typu - wystąpienie publiczne



Rys. 18. Wykres średniej wartości fal mózgowych u wszystkich badanych w scenariuszu wystąpienie publiczne - film wyświetlony na ekranie



Rys. 19. Wykres średniej wartości fal mózgowych u wszystkich badanych w scenariuszu wystąpienie publiczne - animacja 3D w wirtualnej rzeczywistości

7. Dyskusja wyników i wnioski

Po przeprowadzonych badaniach nastąpiła ich analiza i interpretacja. Po skonfrontowaniu z postawioną tezą stwierdzono, że możliwe jest użycie wirtualnej rzeczywistości jako alternatywy do sytuacji przedstawionej w rzeczywistości. Badania dostarczyły danych, które pozwoliły potwierdzić, że

badany podobnie reaguje na sceny w wirtualnej rzeczywistości jak i poza nią. Wpływ na badanie miało wiele czynników. m. inczy dana osoba cierpi na fobię, która powoduje silniejszą reakcję na dany bodziec, czy jest świadoma tematu badania lub czy w momencie badania jest w pełni skupiona na jego przebiegu. Aby to zniwelować badanie przeprowadzono na wielu badanych i zapewniono im dogodne warunki oraz dano wystarczającą ilość czasu aby w pełni skupili się na badaniu. Wykresy EEG i pulsu w zależności od czasu dostarczają informacji jak reagował badany na daną sytuację, czy spowodowało to u niego jakąś reakcję obronną a w połączeniu z analizą funkcji motorycznych daje pełny obraz badania. Z pewnością zwiększenie grupy badawczej oraz użycie bardziej zaawansowanego zestawu badawczego poprawiłoby jakość otrzymanych wyników, jednak użyte komponenty, mimo ich budżetowych konfiguracji, spełniły swoje zadanie i w dostateczny sposób umożliwiły sprawdzenie postawionej tezy. Wirtualna rzeczywistość ma przed sobą wielką przyszłość, rozwój technologii i zwiększanie mocy obliczeniowej urządzeń a co za tym idzie - zwiększanie się jakości oprawy graficznej coraz trudniej będzie rozróżnić rzeczywistość od wirtualnej rzeczywistości.

Literatura

[1] Andreas Mühlberger , Henrik M. Peperkorn, Temporal dynamics in the relation between presence and fear in virtual reality. Computers in Human Behavior, 48:542–547, 7 2015.

[2] Iris M . Engelhard, GaëtanMertens, Patrick Wagensveld, Cue conditioning using a virtual spider discriminates between high and low spider fearful individuals. Computers in Human Behavior, 91:192–200, 2 2019.

[3] Kim Willems, Helena Van Kerrebroeck, MalaikaBrengman, Escaping the crowd: An experimental study on the impact of a virtual reality experience in a shopping mall. Computers in Human Behavior, 77:437–450, 12 2017.

[4] Alberto Greco -Jaime Guixeres , Carmen Llinares , Enzo Pasquale Scilingo , Mariano Alcañiz , Gaetano Valenza, Javier Marín-Morales, Juan Luis Higuera-Trujillo, Affective computing in virtual reality: emotion recognition from brain and heartbeat dynamics using wearable sensors. Springer Nature, 9 2018.

[5] M. Kaufmann, Understanding virtual reality. Elsevier, 6 2003.

[6] D. Napalkov, M. Sosenko, and I. Shestova, Cardiac rhythm as an indicator of normal and pathological adaptation to mental effort. Kybernetes, 24:10–19, 1995.

[7]] M. Park, H. Oh, H. Jeong, J. Sohn, Eeg-based emotion recogntion during emotionally evocative films. IEEE, 2 2013.

[8]] S. Cheemalapati, M. Gubanov, M. Del Vale, A. Pyayt, A real-time classification algorithm for emotion detection using portable eeg. IEEE, 8 2013.

[9] N. Hamid, N. Sulaiman, Z. Murat, M. Taib, Brainwaves stress pattern based on perceived stress scale test. IEEE, 8 2015.

[10] P. Simonov, The Physiology of Emotions. Springer, Boston, MA, 1986.

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 12 (2019) 187-192
Wysłane: 2019-05-23
Przyjęte: 2019-05-29

Analiza wpływu projektu interfejsu graficznego na liczbę i czas odwiedzin strony

Izabela Wasyluk*, Grzegorz Kozieł

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

Streszczenie. Artykuł prezentuje, jak praca nad projektem strony internetowej może zwiększyć biznesowe korzyści płynące z jej posiadania oraz zmniejszyć koszty inwestowania w reklamy. Do realizacji badań wykorzystano stronę internetową realnie funkcjonującej szkoły języka angielskiego posiadającej rzeczywiste potrzeby zwiększenia konwersji użytkowników. Podczas testów gromadzono szczegółowe informacje o odwiedzinach, między innymi długość pobytu na stronie i liczbę odwiedzin na poszczególnych podstronach. W badaniach szczególną uwagę przywiązano do zmian takich cech elementów graficznych jak wielkość, kolor oraz położenie na stronie.

Słowa kluczowe: graficzny interfejs użytkownika; projekt strony internetowej; konwersja użytkowników

*Autor do korespondencji.

Adres e-mail: izabela.wasyluk@pollub.edu.pl

The analysis of the influence of a graphical user interface's design on the number and time of website visits

Izabela Wasyluk*, Grzegorz Kozieł

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

Abstract. Article presents how working on a website's design can significantly influence business benefits deriving from running the site, and if it can reduce the expenses connected with investing in advertisements. The research was based on the website of an operating English language school with real needs to increase user conversion. During the tests, specific information concerning visits was collected, including the length of a visit and the number of visits paid to particular subpages. Special attention in this research was paid to changes on graphic elements such as size, colour or location on the website.

Keywords: graphical user interface; website's design; user conversion

*Corresponding author.

E-mail address: izabela.wasyluk@pollub.edu.pl

1. Wstęp

W dzisiejszych czasach 64% małych przedsiębiorstw na świecie posiada własną stronę internetową [1], zaś w Polsce odsetek ten sięga 62,6% [2]. Strony internetowe wykorzystywane są nie tylko do kontaktu z klientem, ale także do prezentacji firmy oraz oferowanych przez nią usług. Wraz z postępującą popularyzacją Internetu, witryna internetowa stała się istotnym elementem wizerunku marki, sposobem na budowanie jej świadomości oraz lojalności wśród klientów [3]. Wzrost znaczenia stron internetowych w działaniach marketingowych przedsiębiorstw zauważony zostało nie tylko w krajach zachodnich, ale także i w Polsce [4]. Jako narzędzie dostępne na szeroką skalę, prezentowane nieustannie i nieustannie komunikujące się z nowymi odbiorcami, własna strona internetowa daje przedsiębiorstwom wiele nowych możliwości przy stosunkowo niewielkim nakładzie finansowym [5].

Dodatkowo warto nadmienić, iż według statystyk z 2016 roku, w Stanach Zjednoczonych koszt pozyskania nowego odbiorcy na stronie internetowej poprzez reklamy sięga około 92$, podczas gdy koszt konwersji użytkownika na klienta to około 1$ [6]. Z danych tych można wywnioskować, iż w dobie zwiększających się kosztów prowadzenia działań marketingowych, głównym zadaniem twórcy strony internetowej jest osiągnięcie jak najwyższego współczynnika konwersji odwiedzających.

Coraz istotniejsza rola strony internetowej w procesie pozyskiwania klientów każe zastanowić się nad tym, jak proces ten może zostać usprawniony i zoptymalizowany pod kątem swojej efektywności, by usprawnić proces konwersji. Duży wpływ na konwersję ma z pewnością treść prezentowana na stronie. Umieszczenie dokładnych opisów poszczególnych usług wraz z ich cenami, czy adresu siedziby firmy wraz z mapą lub szczegółowym opisem dojazdu z pewnością mogą przyczynić się do zwiększenia współczynnika konwersji.

Innym elementem strony internetowej, którego wpływ może znacząco decydować o tym, jak marka zostanie odebrana przez poszczególne grupy konsumentów, jest wygląd strony internetowej. Analizując podstawowe badania z zakresu psychologii barw oraz bazując na własnych doświadczeniach związanych z odwiedzaniem witryn internetowych, postawić można hipotezę, iż układ elementów w szacie graficznej oraz dobór barw, wielkości tekstu, a także dodatkowych elementów graficznych, może znacząco wpływać na zainteresowanie użytkowników treścią prezentowaną na stronie.

Celem prezentowanych badań jest wykazanie stopnia wpływu projektu graficznego interfejsu użytkownika na poziom zainteresowania treścią oraz czy odpowiednio przygotowany projekt może zwiększyć wysokość współczynnika konwersji. Wyniki badań mają na celu wskazać, czy praca nad projektem szaty graficznej może obniżyć koszty inwestowania w reklamy strony internetowej lub znacząco zwiększyć płynące z niej korzyści.

2. Analiza Literatury

Jednym z problemów, z jakimi mierzą się przedsiębiorcy w sieci internetowej, jest duża liczba odrzuceń witryny [7]. Wskaźnik odrzuceń witryny jest to procent użytkowników, którzy pojawili się na witrynie i podczas odwiedzin nie kliknęli w żaden odnośnik [8]. Obecnie w literaturze naukowej znaleźć można liczne przykłady badań na temat wpływu układu szaty graficznej na aktywność użytkowników na stronie. Przykładem takich badań mogą być testy prowadzone na jednej ze stron internetowych w 2010 roku. Głównym problemem, na którym skupione były badania, była niska aktywność użytkowników na stronie głównej witryny. Na przestrzeni sześciu miesięcy zespół badawczy przeprowadził testy A/B układu szaty graficznej na próbie około 500 tysięcy odwiedzających, nie zmieniając przy tym wyglądu wyświetlanych elementów. Po wykonaniu 54 testów A/B, autorzy badania zdołali sześciokrotnie zwiększyć aktywność użytkowników na stronie. Badanie to potwierdziło, iż układ witryny internetowej, często zmieniony w sposób nieznaczny, lecz dokładnie przemyślany może wywrzeć bardzo silny wpływ na liczbę podejmowanych aktywności przez użytkowników [9].

Podobne badania zostały przeprowadzone przez twórców serwisu Netflix na przełomie lat 2009-2010. Głównym założeniem badania było dokonanie zmian pomiędzy układem zakładek wyświetlanych w menu podręcznym na stronie głównej witryny. Badania te skupiły się przede wszystkim na ułożeniu dwóch zakładek umożliwiających korzystanie z serwisu w dwojaki sposób. Wyniki badań pokazały, iż zakładka wyświetlana jako pierwsza w menu była odwiedzana częściej niż poprzednio, jednocześnie nie zmniejszyła się natomiast liczba odwiedzin na stronie, do której prowadziła zakładka druga [10].

Innym problemem związanym z wyglądem interfejsu użytkownika oraz jego wpływem na reakcje odwiedzających, jest psychologia barw. Istotnym jest, w jaki sposób poszczególne grupy osób reagują na barwy elementów wyświetlanych na stronie. Nad psychologią barw przeprowadzono już szereg badań, skupiając się między innymi na tym, jakie emocje wywołują w poszczególnych grupach społecznych lub jakie wywołują skojarzenia [11]. W badaniu przeprowadzonym w 2005 roku udział wzięło kilka grup osób w wieku 20-30 lat. Badanym przedstawiano kolorowe prostokąty oraz postawiono pytanie o emocje, które wywołują ich barwy. Wszystkie barwy proste u większości badanych osób wywoływały emocje pozytywne, brak było istotnej różnicy w odpowiedziach mężczyzn i kobiet. W przypadku pozostałych barw łatwo było zauważyć odwrotne tendencje w odczuciach negatywnych i pozytywnych pomiędzy grupami badanych mężczyzn i kobiet [12].

Ponadto oddziaływanie barwy na zachowania ludzkie badane są pod kątem marketingowym. Przykładem mogą być badania prowadzone na grupach osób w każdym wieku. Badanie oparte było o ankietę, w której uczestnicy pytani byli między innymi o wpływ koloru na ich decyzje przy zakupie produktów codziennego użytku. Aż 65% ankietowanych odpowiedziało, iż przy zakupach kierują się kolorem produktów [13].

Innym sposobem badania wpływu obserwowanych barw na człowieka jest obserwacja reakcji organizmu ludzkiego. Badania przeprowadzone na grupie kanadyjskich studentów wykazały, iż obserwowanie koloru czerwonego przez dłuższy czas prowadzi do podwyższenia ciśnienia, przyspieszenia pulsu i oddechu [14]. Z kolei obserwowanie koloru niebieskiego prowadziło do wyciszenia systemu nerwowego. Zaobserwowane zjawiska pozwoliły udowodnić, iż obserwacja poszczególnych barw ma wpływ na reakcje układu nerwowego, a więc może istotnie wpłynąć na to, czy dane barwy kojarzone są przez ludzki organizm z pozytywnymi lub negatywnymi odczuciami [15].

Wśród publikacji naukowych oraz książek związanych z projektowaniem graficznego interfejsu użytkownika, obecnie dużą popularnością cieszy się temat użyteczności aplikacji. Temat ten nie dotyczy jednak wartości i korzyści biznesowych płynących z optymalizacji interfejsu, a skupia się raczej nad tym, czego użytkownik doświadcza podczas korzystania z aplikacji. W obrębie tego zagadnienia wymieniane są między innymi takie pojęcia jak funkcjonalność i efektywność aplikacji [16]. Choć efektywność i użyteczność aplikacji są pojęciami, które nie dotyczą ściśle wartości biznesowych, analiza doświadczeń sprzedawców pracujących dla wielkich korporacji pokazuje, iż niespełnione oczekiwania klientów względem użyteczności mają silny wpływ na odbiór nowych produktów przez konsumentów [17]. Bez względu na funkcjonalność produktu, może on okazać się porażką producenta, jeśli zawiedzie filar użyteczności lub efektywności.

Badania prowadzone nad wyglądem stron internetowych to temat stosunkowo nowy względem badań nad psychologicznym oddziaływaniem barw oraz różnicami w ich odbiorze pomiędzy różnymi grupami osób badanych. Również temat użyteczności i efektywności aplikacji został dotychczas zgłębiony, a wyniki badań pozwalają na właściwe ukierunkowanie procesu projektowania aplikacji w celach biznesowych. Większość obserwacji związanych z projektami szaty graficznej pod kątem sprzedaży, a także ich wpływ na zachowanie użytkowników, to w gruncie rzeczy temat przebadany przez praktyków przy pomocy eksperymentów prowadzonych w celach biznesowych.

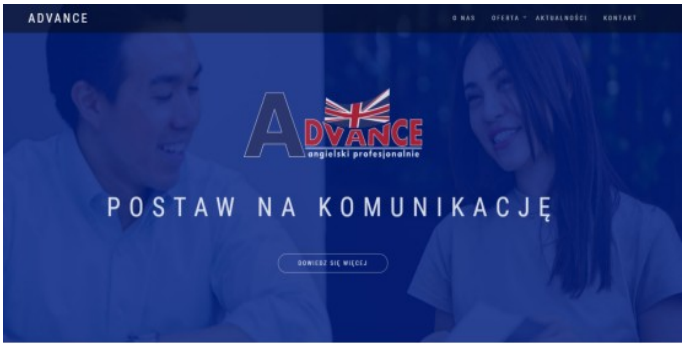
3. Metodyka Badawcza

Badania przeprowadzono na dedykowanej stronie internetowej. Witryna prezentuje informacje na temat działającej realnie szkoły języka angielskiego oferującej zajęcia stacjonarnie oraz przez komunikator z użyciem kamerki internetowej. Druga z form powoduje, iż grono odbiorców strony internetowej poszerza się z mieszkańców określonego miasta do wszystkich osób posługujących się językiem polskim.

3.1. Projekt strony internetowej

Ze względu na charakter badań, projekt strony internetowej był jednym z najistotniejszych elementów w procesie przygotowawczym. W przypadku badań wydajnościowych, projekt graficzny jest zazwyczaj pomijany, w tym przypadku jednak odpowiednie przygotowanie szaty graficznej stanowiło klucz do uzyskania rzetelnych wyników pomiarowych.

Graficzny interfejs użytkownika musiał być przygotowany w taki sposób, aby już pierwsza, niezoptymalizowana jego wersja była łatwa w obsłudze i przyjazna użytkownikowi. W sytuacji gdy projekt strony sam w sobie byłby nieczytelny lub innymi słowy nieużyteczny, drobne zmiany wprowadzane pomiędzy poszczególnymi wersjami mogłyby być zbyt małe, aby przynieść jakiekolwiek efekty.



Rys. 1. Projekt początkowy strony internetowej.

W podstawowej wersji strony internetowej w tle umieszczone zostały zdjęcia osób pogodnych, radosnych i rozmawiających ze sobą, co miało na celu wywołanie u odwiedzających skojarzeń ze swobodną i przyjazną atmosferą podczas zajęć prowadzonych przez szkołę językową. Celowo uniknięto na zdjęciu skojarzeń takich jak książki czy szkolne ławki, aby zakomunikować potencjalnym klientom charakter prowadzonych przez szkołę zajęć. Użyte zdjęcia udostępnione zostały na licencji CC0.

3.2. Gromadzenie i analiza danych

Badania opisane w niniejszej pracy zostały przeprowadzone metodą testowania A/B na platformie Google Analytics przy użyciu narzędzia Google Optimize.

Testy A/B to metoda testowania oparta na zasadzie porównania efektów [18]. Główne zasady testów A/B to :

- równolegle testowane są tylko dwie wersje projektu,
- poszczególne wersje przedstawiane są użytkownikom w sposób losowy,
- pomiędzy dwiema wersjami projektu istnieje tylko jedna różnica [19].

Wyżej wymienione zasady pomagają w sposób klarowny wyodrębnić wersję, która osiąga wyższy wskaźnik konwersji. Wskaźnik konwersji ustalany jest przed rozpoczęciem testów, może to być na przykład kliknięcie w przycisk “Kontakt” lub odwiedziny dłuższe niż 1 minuta.

Podczas testów A/B gromadzi się określone dane o użytkownikach, na przykład wiek lub płeć, dzięki czemu osoba testująca może dogłębniej przyjrzeć się wynikom i przeanalizować wskaźniki konwersji dla poszczególnych wersji z wyodrębnieniem określonych grup odbiorców.

Przeprowadzenie serii testów A/B na witrynie internetowej nazywane jest optymalizacją strony internetowej i ma na celu zwiększenie ogólnego wskaźnika konwersji, czyli np. zwiększenie sprzedaży produktów o określony procent [20].

Przy pomocy testów A/B możliwe jest testowanie treści prezentowanej na stronie (na przykład tekstu umieszczonego na przycisku lub nagłówka powitalnego), testowanie wyglądu poszczególnych elementów, a także organizacji (ułożenia) elementów na stronie.

Podczas obróbki danych wykluczono wszystkie wyniki pochodzące z nieokreślonych źródeł (bez informacji o płci, wieku lub narodowości), a także wyniki pochodzące spoza grupy badanej.

3.3. Opis eksperymentu badawczego

Eksperyment badawczy rozpoczął się od umieszczenia strony internetowej w sieci oraz udostępnienia jej wszystkim użytkownikom bez ograniczeń. Do tego celu wykorzystano istniejącą już domenę, na której dotychczas znajdowała się witryna o tej samej treści, jednak z mało nowoczesnym projektem szaty graficznej. Dzięki wykorzystaniu istniejącej strony internetowej, możliwe było przeprowadzenie badań na szerszej grupie użytkowników ze względu na ruch wygenerowany przy pomocy reklam. Następnie przy pomocy narzędzia Google Optimize skonfigurowano warianty pierwszego testu, wykorzystując przy tym pierwszy scenariusz badawczy. Dodatkowo zgodnie ze scenariuszem ustawiono wskaźnik konwersji i uruchomiono test.

Czas trwania badania dla jednego scenariusza testowego wynosił dwa tygodnie. Po tym czasie uruchomiono kolejny test, wykorzystując do tego wskaźnik konwersji i warianty z kolejnego scenariusza badawczego. Czynność ta powtarzana była dla wszystkich scenariuszy badawczych.

Osoby badane nie zostały dobrane w żaden określony sposób, ani też informowane o udziale w badaniu. Dzięki temu zachowania osób badanych są naturalne i niewymuszone.

Podczas eksperymentu badawczego każdy scenariusz posiadający wyższy wskaźnik konwersji przyjmowany jest jako wersja wyjściowa dla kolejnego scenariusza. Dzięki temu możliwe jest pełne wykorzystanie możliwości optymalizacji strony internetowej oraz zaobserwowanie różnic w zachowaniach użytkowników pomiędzy wersją pierwszą a wersją zoptymalizowaną.

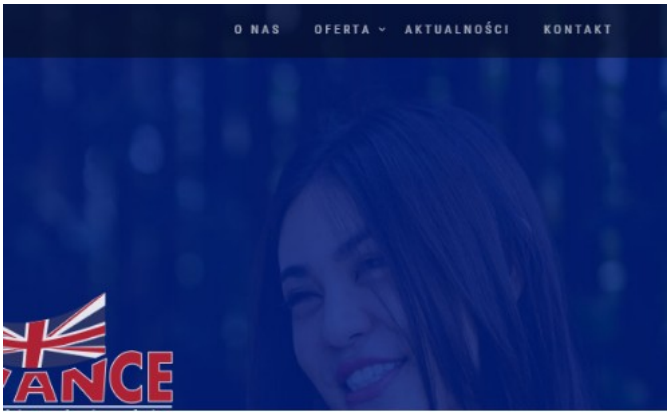
Oprócz wskaźnika konwersji, podczas przeprowadzania każdego testu gromadzone są dodatkowe dane związane z zachowaniami użytkowników. Między innymi są to:

- liczba odwiedzin na określonych podstronach,
- długość czasu pobytu na stronie,
- liczba użytkowników podejmujących kontakt ze szkołą.

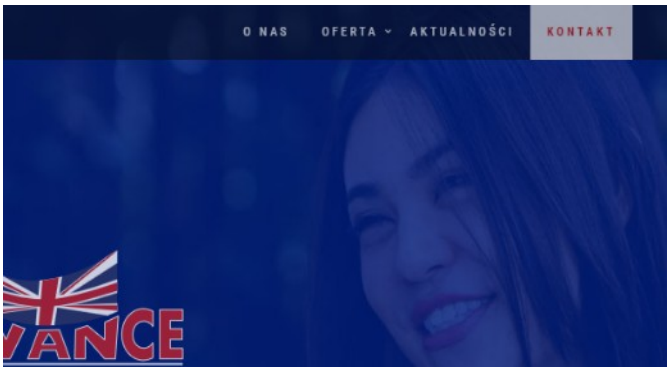
3.4. Scenariusze badawcze

Badanie składało się z pięciu scenariuszy badawczych, z czego każdy zawierał dwa warianty testowe (wersje) oraz osobny wskaźnik konwersji wraz z dodatkowym

wskaźnikiem konwersji. Przykładową różnicę pomiędzy poszczególnymi wersjami scenariusza przedstawia Rys. 2. oraz Rys. 3. Podstawowy wskaźnik konwersji służył do wyznaczenia zwycięskiej wersji w danym scenariuszu, natomiast dodatkowy wskaźnik konwersji pozwalał na wykluczenie przypadkowości wyników.



Rys. 2. Wersja A przykładowego scenariusza badawczego



Rys. 3. Wersja B przykładowego scenariusza badawczego.

Scenariusze badawcze zawierały porównanie efektów następujących zmian:

- Scenariusz 1:
  - Wersja A: przycisk Kontakt bez tła,
  - Wersja B: dodanie białego, półprzezroczystego tła dla przycisku Kontakt,
  - Wskaźnik konwersji: Liczba wyświetleń strony Kontakt,
- Scenariusz 2:
  - Wersja A: przycisk Kontakt umieszczony jako pierwszy element w Menu,
  - Wersja B: przycisk Kontakt umieszczony jako ostatni element w Menu,
  - Wskaźnik konwersji: Liczba wyświetleń strony Kontakt,
- Scenariusz 3:
  - Wersja A: tekst na przycisku “Dowiedz się więcej” w rozmiarze 11px,
  - Wersja B: tekst na przycisku “Dowiedz się więcej” w rozmiarze 18px,
  - Wskaźnik konwersji: Liczba wyświetleń strony z ofertą,
- Scenariusz 4:
  - Wersja A: zdjęcie uczącego się mężczyzny jako tło strony głównej,
  - Wersja B: zdjęcie uśmiechniętej, rozmawiającej pary jako tło strony głównej,
  - Wskaźnik konwersji: liczba odrzuceń strony,
- Scenariusz 5:
  - Wersja A: jednolity kolor częściowo przysłaniający zdjęcie w tle (niebieski),
  - Wersja B: gradient częściowo przysłaniający zdjęcie w tle (niebiesko-różowy),
  - Wskaźnik konwersji: liczba odrzuceń strony.

Pełną listę rysunków przedstawiających poszczególne wersje scenariuszy zawarto w pracy Wasyluk I.: “Analiza wpływu projektu interfejsu graficznego na liczbę i czas odwiedzin strony”.

4. Wyniki badań

Badania prowadzone były na losowej grupie użytkowników pojawiających się na stronie internetowej, dlatego też wyniki musiały zostać poddane obróbce polegającej na odfiltrowaniu wszystkich wyników generowanych przez osoby spoza grupy docelowej. Spośród otrzymanych wyników wymazano zatem wszystkie rekordy, w których narzędzie do testowania nie zdołało określić narodowości oraz wszystkie rekordy gdzie narodowość została określona inaczej niż wskazano w opisie grupy docelowej. Po przefiltrowaniu wyników, rekordy zostały zsumowane oraz podzielone na podstawie scenariusza badawczego. Sumy otrzymane dla poszczególnych wskaźników przedstawiono w tabelach 1. - 5.

Tabela 1. Wyniki badań dla scenariusza 1

Scenariusz 1	**Wersja A**	**Wersja B**
Liczba wyświetleń strony głównej	950	951
Liczba wyświetleń strony Kontakt	26	14
Liczba wyświetleń strony z ofertą	42	38
Liczba odrzuceń strony	84	102
Średni czas pobytu na stronie	93s	92s
Liczba wyświetleń strony "O Nas"	15	13
Liczba użytkowników, którzy podjęli kontakt	2	0

Tabela 2. Wyniki badań dla scenariusza 2

Scenariusz 2	**Wersja A**	**Wersja B**
Liczba wyświetleń strony głównej	1068	1068
Liczba wyświetleń strony Kontakt	31	11
Liczba wyświetleń strony z ofertą	50	32
Liczba odrzuceń strony	114	86
Średni czas pobytu na stronie	102s	99s
Liczba wyświetleń strony "O Nas"	17	14
Liczba użytkowników, którzy podjęli kontakt	5	1

Tabela 3. Wyniki badań dla scenariusza 3

Scenariusz 3	**Wersja A**	**Wersja B**
Liczba wyświetleń strony głównej	996	995
Liczba wyświetleń strony Kontakt	34	30
Liczba wyświetleń strony z ofertą	46	52
Liczba odrzuceń strony	96	90
Średni czas pobytu na stronie	104s	114s
Liczba wyświetleń strony "O Nas"	8	14
Liczba użytkowników, którzy podjęli kontakt	4	0

W tabelach wyszczególniono takie wskaźniki jak liczba wyświetleń strony głównej, strony “Kontakt”, strony “O Nas”, strony z ofertą, a także średni czas pobytu na stronie, liczbę odrzuceń strony oraz liczbę osób, które nawiązały kontakt z firmą poprzez formularz kontaktowy.

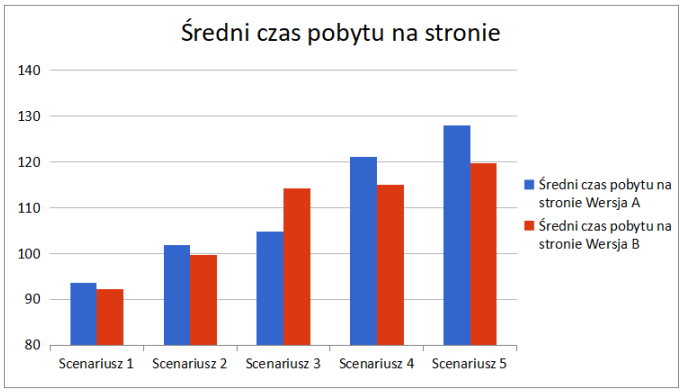
Tabela 4. Wyniki badań dla scenariusza 4

Scenariusz 4	**Wersja A**	**Wersja B**
Liczba wyświetleń strony głównej	1120	1119
Liczba wyświetleń strony Kontakt	35	39
Liczba wyświetleń strony z ofertą	50	62
Liczba odrzuceń strony	78	96
Średni czas pobytu na stronie	121s	115s
Liczba wyświetleń strony "O Nas"	10	11
Liczba użytkowników, którzy podjęli kontakt	7	3

Tabela 5. Wyniki badań dla scenariusza 5

Scenariusz 5	**Wersja A**	**Wersja B**
Liczba wyświetleń strony głównej	796	798
Liczba wyświetleń strony Kontakt	24	22
Liczba wyświetleń strony z ofertą	37	32
Liczba odrzuceń strony	58	69
Średni czas pobytu na stronie	128s	119s
Liczba wyświetleń strony "O Nas"	12	9
Liczba użytkowników, którzy podjęli kontakt	5	4

W wyniku wykonanych badań udało się wydłużyć średni czas pobytu na stronie z 93 sekund (wynik uzyskany dla projektu początkowego) do 128 sekund (Rys. 4).



Rys. 4. Wykres średniego czasu pobytu na stronie dla poszczególnych wersji scenariuszy.

W trakcie ostatnich dwóch scenariuszy udało się również zwiększyć szczególnie istotny współczynnik konwersji, jakim jest przesłanie formularza kontaktowego przez użytkownika. Wartość tego współczynnika została zwiększona z 0 do 7.

5. Wnioski

Proces optymalizacji strony internetowej przeprowadzony został etapowo. Każdy z etapów, tj. scenariuszy testowych, posiadał osobny wskaźnik konwersji. Wyraźne różnice pomiędzy pomiarami wskaźników konwersji były zauważalne tam, gdzie dokonywano zmian powiązanych z tym wskaźnikiem. Za przykład mogą posłużyć tutaj zmiany na przycisku Kontakt, które skutkowały wyraźnymi zmianami w liczbie wyświetleń strony “Kontakt” przez użytkowników. Istotnym w tej kwestii jest również fakt, iż różnica ta nie była tak wyraźna w scenariuszach, które dokonywały zmian na innych elementach strony.

Dwa pierwsze scenariusze badawcze posiadające ten sam wskaźnik konwersji (liczba kliknięć w przycisk Kontakt na stronie), w których edytowano wygląd oraz położenie przycisku na stronie, spowodowały wyraźną zmianę liczby kliknięć w przycisk pomiędzy wersjami scenariusza. Zmiany te były dużo mniejsze w przypadku pozostałych scenariuszy.

Dzięki obserwacji długości czasu pobytu na stronie, podczas prowadzenia badań można było zaobserwować, iż wraz z procesem optymalizacji strony, użytkownicy chętniej pozostają na niej dłużej. Biorąc pod uwagę fakt, iż od scenariusza 3. do scenariusza 5. wzrost ten był znacznie większy niż w przypadku dwóch pierwszych scenariuszy, można wywnioskować, iż zmiany dokonane w tych scenariuszach pomogły zwiększyć zainteresowanie użytkowników treścią. Wiążąc długość czasu odwiedzin ze wzrastającą liczbą wyświetleń na stronie z ofertą, można wyciągnąć wniosek, iż wraz z postępującą optymalizacją, coraz więcej użytkowników interesowało się i zapoznawało się z ofertą firmy.

Dodatkowo szczególną uwagę należy zwrócić na liczbę osób, które nawiązały kontakt z firmą w trakcie poszczególnych scenariuszy. Na przestrzeni kolejnych badań, liczba osób, które podejmowały kontakt z firmą stale rosła. Liczba osób które podjęły kontakt wzrosła od 0 w przypadku scenariusza 1. do 7 w przypadku scenariusza 4.

Istotnym elementem prowadzonych badań był pomiar liczby wyświetleń strony “O Nas”, który nie był wskaźnikiem w żadnym ze scenariuszy badawczych. Obserwacja wyników pomiarowych dla tego wskaźnika pozwoliła na weryfikację braku losowości w przeprowadzonych badaniach. Na wykresie liczby wyświetleń strony “O Nas” trudno jest dostrzec jakiekolwiek zależności pomiędzy wynikami w poszczególnych scenariuszach. Oznacza to, iż bez zmian związanych z elementami nawigacyjnymi prowadzącymi do strony “O Nas” liczba odwiedzin na tej stronie nie zmieniała się w żaden określony sposób, a drobne zmiany były jedynie efektem losowości.

Literatura

[1] Delgado M., Small Business Websites in 2018. “Clutch”. [03.02.2019].

[2] Orłowska J., Łapiński J., Raport o stanie sektora małych i średnich przedsiębiorstw w Polsce. Polska Agencja Rozwoju przedsiębiorczości, 2018.

[3] Barr C., Weiss A., Million Dollar Web Presence: Leverage the Web to Build Your Brand and Transform Your Business. Entrepreneur Press, 2012.

[4] Talar S., Kos-Łabędowicz J., Internet w działalności polskich przedsiębiorstw. Wydawnictwo Uniwersytetu Ekonomicznego w Katowicach, 2014.

[5] Michalak P.R., Daszkiewicz D., Musz A., Marketing wirusowy w internecie. Helion, 2012.

[6] Hubspot, Search Engine Optimization Statistics. [03.02.2019].

[7] Bonek T., Smaga M., Biznes w internecie: Praktyczny poradnik o marketingu, sprzedaży, public relations on-line i promocji w mediach społecznościowych. Wolters Kluwer, 2012.

[8] Siroker D., Koomen P., Testy A/B. Od kliknięcia do klienta". Helion, 2013.

[9] King R., Churchill E. F., Tan C., Designing with Data. O'Reilly Media, 2017.

[10] Elliot AJ., Color and psychological functioning: a review of theoretical and empirical work. Frontiers in psychology, 04/2015.

[11] Kolek Z., Psychofizyka barwy. Katedra Metrologii i Analizy Instrumentalnej Wydział Towaroznawstwa UEK, 2010.

[12] Causse J.G., Niesamowita moc kolorów". Wydawnictwo Sonia Draga, 2015.

[13] Tatarska J., Rola koloru w reklamie prasowej. Wyższa Szkoła Promocji, 2013.

[14] McNamara N., Kirakowski J., Functionality, Usability and User Experience: Three Areas of Concern. Interactions, 11/2006.

[15] Goodman E., Kuniavsky M., Moed A.: Observing the User Experience, A Practitioner's Guide to User Research. Elsevier, 2012.

[16] Strzębska A., Jak działają testy A/B? [10.04.2019].

[17] Goward C., You Should Test That: Conversion Optimization for More Leads, Sales and Profit or The Art and Science of Optimized Marketing. Sybex, 2013.

[18] Eisenberg B., von Tivadar J. Q.., Always Be Testing: The Complete Guide to Google Website Optimizer. Sybex, 2008.

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 12 (2019) 193-198
Wysłane: 2019-06-14
Przyjęte: 2019-06-22

Porównanie efektywności składowania modeli UML w wybranych technologiach bazodanowych

Andrii Filatov, Paweł Flis*, Beata Pańczyk
Politechnika Lubelska, Instytut Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. Artykuł odpowiada na pytanie, która baza danych jest najlepszym wyborem do efektywnego składowania danych opisujących modele UML. Wzięto pod uwagę trzy produkty: MongoDB, PostgreSQL i Neo4J. Na badanie efektywności składa się pomiar czasu odpowiedzi zapytań zapisujących oraz pobierających dane. Uwzględnia również stopień wzrostu pamięci podczas wstawiania danych oraz ocenę poziomu skomplikowania implementacji mapującej dane testowe do wykorzystania w zapytaniach do baz danych.

Słowa kluczowe:UML; MongoDB; Neo4J; PostreSQL

*Autor do korespondencji.

Adresy e-mail: andriymasteridze@gmail.com, stercage@gmail.com

Storage efficiency comparison of UML models in selected database technologies

Andrii Filatov, Paweł Flis*, Beata Pańczyk
Institute of Computer Science, Lublin University of Technology, Nadbystrzycka 36B,20-618 Lublin, Poland

Abstract. The study answers the question which database is the best choice for efficient data storage of UML models. Three products were considered: MongoDB, PostgreSQL and Neo4J. The effectiveness test consists of measurement the response time of queries that save and load data. This study also take into account the memory increase ratio during data insertion and the level of complexity of the implementation of the test data mappers used in database queries.

Keywords: UML; MongoDB; Neo4J; PostgreSQL

*Corresponding author.

E-mail addresses: andriymasteridze@gmail.com, stercage@gmail.com

1. Wstęp

W obecnych czasach, modelowanie staje się coraz ważniejszym elementem wytwarzania oprogramowania, od którego w dużym stopniu zależy sukces całego przedsięwzięcia. Pozwala ono graficznie wyrazić najważniejsze cechy projektu oraz jego otoczenia. Jest mniej abstrakcyjne niż kod programu. W efekcie pozwala to na łatwe zaprojektowanie oprogramowania i częściowo automatyczną implementację jego kodu. Dzięki modelowaniu uzyskuje się ogólny obraz systemu. Można więc przedstawić abstrakcyjny model systemu klientowi przed jego powstaniem. W przypadku, gdy nastąpi potrzeba zmiany lub dodania nowej funkcjonalności, wystarczy zmienić to w modelu. Redukuje to koszt i czas realizacji całego projektu.

Na dzień dzisiejszy, 30-letnie języki i metody programowania są nadal używane. Zaistniała jednak pilna potrzeba nowoczesnego podejścia, aby uniknąć błędów programowania, które prowadzą do konieczności długotrwałego rozwiązywania problemów. Nowoczesne programowanie zorientowane na modelowanie (ang. Model Oriented Programming) nie wymaga kodowania ręcznego, kod tworzy komputer, więc wykluczone są wszelkie błędy występujące podczas ręcznej transkrypcji abstrakcyjnego modelu w kodzie oprogramowania[1].

Ponieważ stosowanie modeli staje się coraz bardziej pożądane, należałoby zadbać o właściwy sposób ich przechowywania, tak, aby dostęp do danych możliwy był w jak najkrótszym czasie i jak najniższym kosztem.

Ze względu na rozmiar modeli wykorzystanie pewnego typu bazy danych staje się wymogiem. Można wybierać spośród baz relacyjnych oraz nierelacyjnych (NoSQL). Porównując, bazy relacyjne z różnego typu bazami nierelacyjnymi, pod względem szybkości zapisu i odczytu danych, trudno wyłonić zwycięzcę [2, 3, 4]. Żadna z porównywanych baz nie wygrywa dla wszystkich przypadków testowych. Z tego powodu jednoznaczny wybór bazy danych odpowiedniej do przechowywania modeli nie jest łatwy.

Z wymienionych wcześniej powodów przeprowadzono badanie mające rozstrzygnąć opisany problem. Polegało ono na analizie porównawczej efektywności składowania oraz dostępu do modeli UML na przykładzie wybranych technologii bazodanowych. Efektywność została określona w kilku kategoriach, m. in. czasu dostępu do danych, czasu wstawiania danych, poziomu skomplikowania całego procesu oraz ilości zajmowanej pamięci. Dodatkowo cel badań obejmował utworzenie aplikacji, pozwalającej na przeprowadzenie badań, zebranie wyników oraz ich przetworzenie. Każda z wybranych technologii bazodanowych była reprezentowana przez odpowiednio:

PostgreSQL – bazy relacyjne, MongoDB – bazy dokumentowe, Noe4j – bazy grafowe.

Została postawiona następująca hipoteza badawcza: ***Baza dokumentowa reprezentowana przez MongoDB jest najlepszym wyborem w kwestii efektywnej obsługi struktur danych modeli UML (ang. Unified Modeling Language).***

2. Przegląd litaratury

Nie znaleziono badań zajmujących się dokładnie opisywanym problemem, dlatego przeanalizowano kilka o podobnej tematyce.

Analizę parametrów baz danych z punktu widzenia praktycznego wykorzystania z dużymi zbiorami danych przedstawiono w artykule napisanym przez grupę Marokańskich badaczy[5]. Porównanie zostało dokonane między bazami SQL oraz NoSQL. Zbadano ich bezpieczeństwo, skalowalność, wydajność itp. Wstęp nadmienia, że NoSQL wprowadzono i udoskonalono w celu obsługi Big Data, natomiast język SQL służy do obsługi strukturalnych baz danych i wykonywania transakcji. Bazy danych SQL są skalowalne tylko pionowo, więc aby poradzić sobie z rosnącym obciążeniem, występuje potrzeba zwiększenia pojemności i wydajności na serwerze. Aby to osiągnąć, można na przykład zwiększyć szybkość procesora, pojemność dysku SSD serwera bazy danych lub pamięć RAM. Jednak odkładanie wielu tabel w dużych klastrach lub siatkach jest kosztowne i złożone. Natomiast baza NoSQL jest skalowalna w poziomie. Zatem, obsługa dużych zbiorów danych polega na dodaniu serwera do struktury bazy danych, w wyniku czego, używanie NoSQL daje łatwą i tanią skalowalność systemu. Z drugiej strony relacyjne bazy danych wymagają predefiniowanego modelu danych i danych strukturalnych. Oferują one zaawansowaną funkcjonalność do zarządzania, aktualizacji i wyszukiwania danych za pomocą SQL.

Badanie na temat porównania wydajności baz danych dokonali badacze z Korei[2]. Opracowali program, który umożliwiał wstawianie, aktualizowanie, wybieranie i usuwanie w celu porównania baz relacyjnych (PostgreSQL) i nie relacyjnych (MongoDB). Te operacje zostały wykonane dla różnych rozmiarówdanych. A mianowicie dla każdego z 30000, 90000, 150000, 210000 i 300000. Eksperyment obejmował operację wstawiania, która została wykonana najpierw w PostgreSQL, następnie w MongoDB zaprojektowaną podobnie do modelu relacyjnego, a potem z projektem podobnym do modelu niestrukturalnego. Dla każdej operacji wstawiania porównano czas, jaki upłynął od wyboru, aż do dostępności danych. Badania wykazały, że dla każdej operacji zarówno z projektowaniem niestrukturalnym jak i projektowaniem z relacyjnym modelem danych, baza MongoDB była szybsza. Dodatkowo wykazano, że projektowanie z niestrukturalnym modelem danych wydaje się lepsze niż projektowanie z relacyjnym modelem danych. Używanie MongoDB z niestrukturalnym modelem danych zapewnia ogólną poprawę wydajności. Jednakże, gdy środowisko wymaga precyzyjnego i uporządkowanego modelu danych, używanie RDBMS (ang. Relational Database Management System), takich jak PostgreSQL, będzie wykazywać wyższą jakość wykonania.

Holenderscy autorzy artykułu[4] poszukiwali bazy danych, która by była wydajna podczas wykonywania pojedynczych zapisów i wielu odczytów. Porównano 3 bazy danych pod kątem wydajności. Są to MongoDB, Cassandra, PostgreSQL. Żadna z nich nie sprawdziła się w obu przypadkach. PostgreSQL wygrała przy wielu odczytach, natomiast MongoDB wygrała w kategorii pojedynczych zapisów.

Odpowiedź na pytanie, która baza danych SQL i NoSQL jest najlepsza dla dużych zbiorów danych zawiera artykuł [6]. Ponieważ wiele danych jest niestrukturalnych lub pół-strukturalnych, wynika potrzeba bazy danych, która byłaby w stanie sprawnie je przechowywać. W relacyjnych bazach danych niemożliwe jest szybkie dodanie nowych typów danych, które by nie pasowały do strukturalnych danych, ponieważ schemat jest sztywno zdefiniowany. Natomiast bazy NoSQL zapewniają taki model danych, który lepiej odwzorowuje te potrzeby.

Praca Douglas Kunda i HazaelPhiri z uniwersytetu Mulungushi[7] miała na celu odpowiedzieć na pytanie, czy bazy NoSQL zastąpią bazy relacyjne. Bazy NoSQL stają się coraz popularniejszym wyborem, ponieważ bardziej odpowiadają współczesnym wymaganiom. Jednym z wielu jest wysoki poziom skalowalności. Relacyjne bazy danych mogą być skalowalne, zaś poziom tej skalowalności będzie ograniczony przez sprzęt. Bazy NoSQL skalowane są w poziomie, co oznacza, że sprzęt nie jest ograniczeniem, ponieważ maszyny serwerowe można połączyć, co pozwala zamienić posiadanie jednego kosztownego serwera kilkoma mniejszymi i tańszymi. Więc wydajniejsze jest użycie baz NoSQL. Użycie pamięci ulotnej w przeciwieństwie do relacyjnych baz danych, zwiększa wydajność baz NoSQL: wykonanie podstawowych zapytań, odczytu i aktualizacji. Jednakże nie zastąpią całkowicie relacyjnych baz danych. Relacyjne bazy danych są łatwe do wdrożenia, niezawodne, spójne i bezpieczne, ale nie radzą sobie z niestrukturalnymi danymi.

Trójka badaczy z Indii dokonała badania na uniwersytecie Patiala. W swoim artykule [8] przedstawili kilka prac badawczych mających na celu analizę technologii NoSQL. Zestawiono ze sobą cztery kategorie baz NoSQL: rodzina kolumn, klucz-wartość, grafowe oraz dokumentowe. Z badania wynika, że dla aplikacji wymagającej połączenia z mediami społecznościowymi, najlepszą bazą NoSQL będzie ta bazująca na grafie. Jeśli czas odpowiedzi na zapytanie powinien być jak najkrótszy, najkorzystniejszą bazą będzie typ klucz-wartość. Skalowalność jest również ważnym parametrem w przetwarzaniu dużej ilości danych. Najlepszą pod tym względem jest rodzaj: rodzina kolumn oraz klucz-wartość.

Doktor Małgorzata Plechawska-Wójcik wraz z Damianem Rykowskim z Politechniki Lubelskiej[3] przedstawili porównanie baz relacyjnych (PostgreSQL), dokumentowych (MongoDB) oraz grafowych (Neo4J). Badania przeprowadzono pod kątem wydajnościowym. Aplikacja

testowa to sieć społecznościowa. Testy wydajności przedstawiają wyniki uzyskane dla pięciu różnych zadań. A mianowicie: wybór obiektów powiązanych z przyjaciółmi, wyszukiwanie pełno-tekstowe, znalezienie liczby przyjaciół, ładowanie tematu oraz ocena tematu. Te zadania zostały wykonane dla różnych przypadków danych. A mianowicie dla każdego z 1000, 5000, 10000 przykładowych danych. Analiza wykazała, że baza grafowa najbardziej odpowiada wymaganiom danej aplikacji. Najprostsze w użyciu są zapytania do bazy dokumentowej. Pod względem wydajności PostgreSQL okazał się najlepszy.

Czwórka badaczy z uniwersytetu Jadavpur w Indiach[9] skupiła się na czterech różnych systemach baz danych: strukturalnych (PostgreSQL) i niestrukturalnych (MongoDB, OrientDB, Neo4J). Używając małego zestawu danych, na podstawie prostych zapytań wykonują różne eksperymenty, analizują i porównują wydajność wybranych baz. W eksperymencie wykorzystano zalety każdej z baz. Aplikacja testująca została opracowana wykorzystując zalety baz. Dla PostgreSQL przykładowe dane zawarte są w tabeli „Student" z kolumnami: nazwą, szkołą, klasą, wiekiem. Podobnie w przypadku MongoDB kolekcja „Student" jest reprezentowana jako pary pole-wartość zgodnie ze strukturą dokumentu JSON. Tutaj „nazwa", „szkoła", „klasa", „wiek" są reprezentowane jako klucz wraz z pewnymi wartościami. W przypadku OrientDB dane są przedstawiane jako dokument JSON. W Neo4j „Student" odpowiada nazwie etykiety, która łączy wszystkie węzły lub dane, które są tworzone. Tutaj „nazwa", „szkoła", „klasa", „wiek" to wszystkie właściwości węzła. Wydajność czterech rozważanych baz danych została oceniona na podstawie siedmiu różnych kryteriów: czas tworzenia tabeli lub zbioru, wstawienie jednego elementu danych, wstawianie N rekordów jednocześnie, zapytanie z jednym warunkiem filtrowania, zapytanie o wszystkie rekordy (całkowita liczba = 21), usunięcie jednego rekordu, usunięcie wszystkich rekordów. Z analizy wynika, że początkowy koszt instalacji Neo4j jest większy, ale koszt wyboru, wstawienia, usunięcia jest najbardziej kosztowny dla PostgreSQL. We wszystkich kategoriach zwycięża MongoDB.

3. Opis procedury badawczej

3.1. Platfora testowa

Do przeprowadzenia badań wykorzystany został komputer przenośny o następujących parametrach:

- procesor: Intel Core i5 6300HQ,
- pamięć operacyjna: 2x4 GB DDR4,
- dysk twardy: magnetyczny Toshiba MQ02ABD100H,
- system: Windows 10.

Testy zostały przeprowadzone przy minimalnym obciążeniu maszyny, gdzie w momencie każdego z testów uruchomione było tylko środowisko jednej bazy danych.

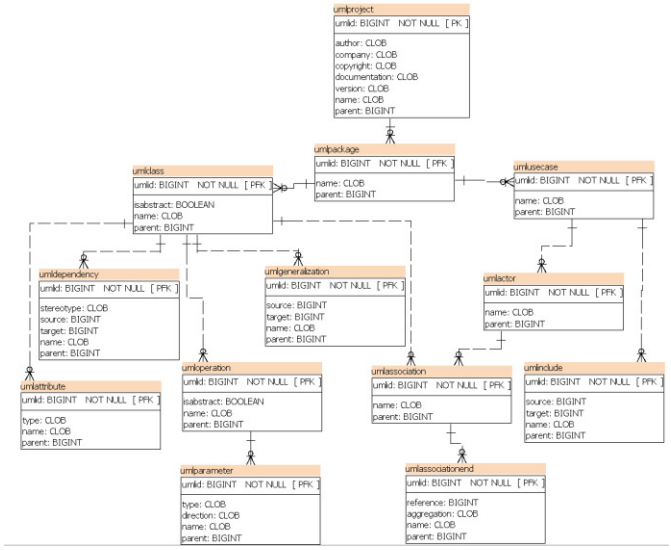
3.2. Dane badawcze

Zbiorem testowym stały się dane reprezentujące wewnętrzną strukturę projektu aplikacji StartUML, czyli dane wykorzystywane przez StartUML do zapisu projektu (diagramów UML) użytkownika na dysku [10]. Projekt taki składa się z definicji projektu, definicji widoku diagramów oraz definicji modeli z grupowaniem i podziałem na typy. Zapisany jest w pliku o rozszerzeniu *.mdj* w formacie JSON. Zawartość tego pliku, zwana dalej projektem UML, będzie wykorzystywana do zapisu i odczytu z poszczególnych baz danych.

W przypadku relacyjnej bazy danych (PostgreSQL) było wymagane stworzenie struktury tabel do przechowywania danych (Rys. 1.).

Przedstawiony diagram ukazuje jedynie główne zależności pomiędzy obiektami wykorzystanymi w badaniach. Struktura aplikacji StartUML pozwala jednak na dowolne zawieranie się elementów w elementach grupujących takich jak *UMLClass*. Z tego powodu schemat ERD nie zwiera powiązań kluczy obcych. Łączenie obiektów odbywa się poprzez identyfikatory *parent* oraz *umlid* zawarte we wszystkich obiektach. Widoczne na rysunku 1 powiązania prezentują jedynie przykładowe zależności pomiędzy tabelami, które pojawiły się w danych testowych.

UMLProject to projekt UML. Element grupujący najwyższego poziomu zawiera wszystkie informacje wymagane do wczytania diagramów stworzonych przez użytkownika w aplikacji StartUML.



Rys. 1. Poglądowy diagram ERD bazy PostgreSQL

UMLPackage to elementy grupujące odpowiadające za budowę diagramów np.: klas, przypadków użycia. Reprezentują pojedyncze diagramy i są wymagane do budowy poszczególne ekranów wyświetlających graficzny schemat modelu. W danych testowych nie zawarto elementów *UMLDiagram*, które pierwotnie znajdowały się w pakietach (*UMLPackage*) i opisywały rozmieszczenie poszczególnych obiektów graficznych, ich kolorystykę, wielkość oraz styl. Zdecydowano się na taki ruch, ponieważ wspomniane elementy nie są wymagane do poprawnej reprezentacji modelu oraz nie wpływają na jego znaczenie – nie wnoszą do

niego nic wartościowego, natomiast powodują znaczący rozrost danych reprezentujących diagramy *UMLPackage* oraz zwiększają poziom skomplikowania kodu aplikacji.

Pozostałe, nie omówione powyżej obiekty są elementami diagramów – reprezentują fragmenty ich struktury, między innymi: klasy *UMLClass*, przypadki użycia *UMLUseCase*, aktorów *UMLActor*.

3.3. Aplikacja testowa

Przeprowadzenie tak dużej liczby testów wymagało odpowiedniego skryptu lub aplikacji, więc zaimplementowana została aplikacja w języku Java. Jej zadaniem było wczytanie projektu UML i stworzenie jego reprezentacji w postaci obiektów języka Java. Tak przygotowane obiekty znacznie uprościły dalsze przetwarzanie projektu i jego wstawianie do bazy relacyjnej i grafowej, ponieważ bazy te wymagają zdefiniowanej struktury (przynajmniej na najwyższym poziomie w przypadku baz grafowych). Aplikacja umożliwiała również zapis zebranych wyników w plikach w formacie csv.

W przypadku testów wymagających więcej niż jednej instancji projektu UML, aplikacja wygenerowała pozostałe, kopiując bazowy projekt, wstawiając nowe identyfikatory oraz aktualizując powiązania.

3.4. Scenariusze badawcze

Aplikacja wykonała pomiary czasu odpowiedzi w 5 kategoriach:

- T1 – wstawianie pojedynczego projektu UML.
- T2 – wstawianie 100 unikalnych projektów.
- T3 – pobranie całego projektu UML wykorzystując identyfikator liczbowy wygenerowany poza bazą danych.
- T4 – pobranie całego projektu UML zawierającego obiekt aktor (UMLActor) wykorzystując identyfikator liczbowy aktora.
- T5– pobranie obiektów UMLClass zawierających w swojej nazwie frazę Builder.

Testy zostały przeprowadzone w kolejności wymienionej powyżej. Dodatkowo pod uwagę wzięto poziom skomplikowania kodu odpowiedzialnego za odwzorowanie projektu UML w języku Java oraz łatwość dostępu do bardziej zagnieżdżonych danych. Przypisano im jedną wartość z następujących trzech: łatwy, przeciętny, trudny.

Zbadano również poziom zajmowanej pamięci w trzech różnych momentach w czasie:

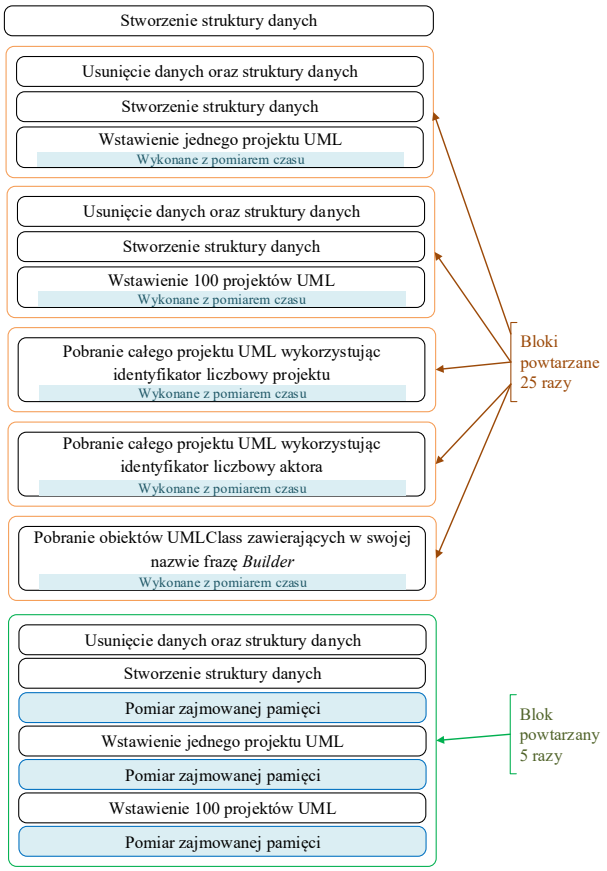
- T6 – przed wstawieniem danych,
- T7 – po wstawieniu jednego projektu,
- T8 – po wstawieniu 100 projektów.

3.5. Przebieg badania

Testy wykonywane były dla poszczególnych baz po kolei, tzn. wszystkie testy dla pierwszej bazy, następnie wszystkie dla drugiej, itd. W momencie testowania konkretnej bazy, pozostałe dwie były wyłączone aby nie wprowadzać zakłóceń w pomiarach. Rys. 2. przedstawia przebieg testów dla pojedynczej bazy.

Dla każdej bazy wykonano ten sam zestaw testów. Pierwsze 5 testówpowtarzane było po 25 razy. Powtórzenia wykonywane były dla poszczególnych testów niezależnie, tzn. wszystkie powtórzenia dla pierwszego, następnie wszystkie dla drugiego, itd. Dla pomiarów zajmowanej pamięci zdecydowano na wykonanie po 5 prób, ponieważ każda próba wymagała wyłączenia i ponownego włączenia bazy. Było to jedyne możliwe rozwiązanie problemu całkowitego wyczyszczenia danych dla bazy Neo4J z powodu alokacji pamięci przez tą bazę bez względu na ilość danych.

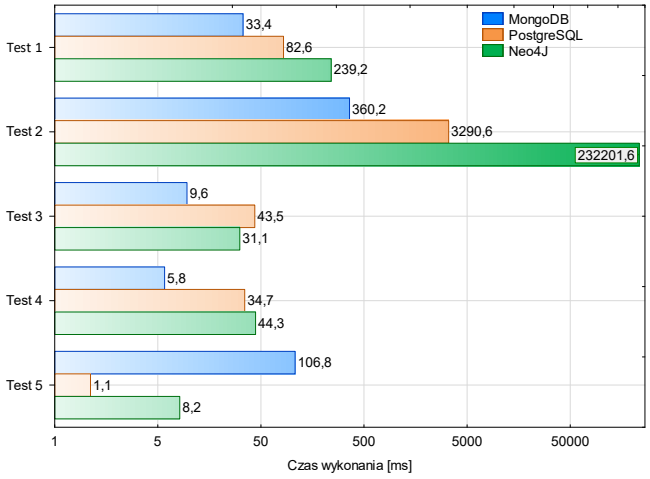
Ze względu na specyfikę baz dokumentowych, test dotyczący pobierania klas *UMLClass* dla MongoDB przebiegał nieco inaczej niż dla pozostałych baz. Ograniczeniem bazy dokumentowej była możliwość pobierania tylko dokumentów nadrzędnych bez wydzielania zagnieżdżonych, tzn. możliwe było tylko pobieranie całych projektów, a nie klas *UMLClass*. Dlatego podczas tego testu wybrano wszystkie projekty zawierające wybrane klasy, a następnie w kodzie Java wyszukano wskazane obiekty *UMLClass* i zwrócono.



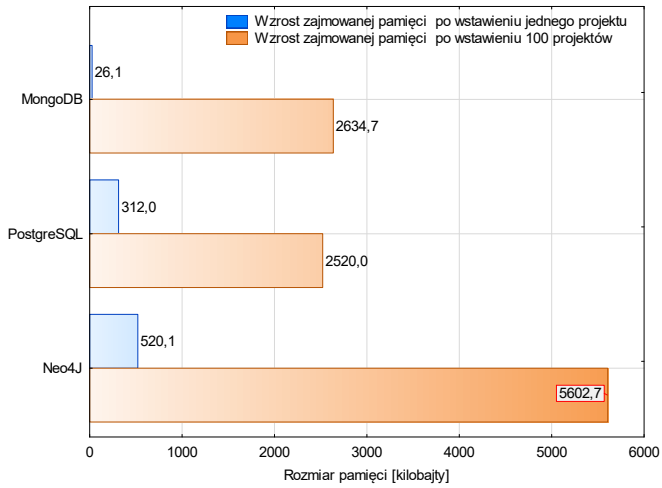
Rys. 2. Przebieg testu

4. Wyniki badania

Pomiary czasu odpowiedzi przedstawiono na rysunku 3. Rysunek 4 obrazujeróżnice w zajmowanej pamięci pomiędzy kolejnymi seriami badań.



Rys. 3. Podsumowanie wyników pomiaru czasu odpowiedzi zapytań



Rys. 4. Wzrost zajmowanej pamięci po wstawieniu 1 oraz 100 projektów

W tabeli 1 przedstawiono zaobserwowane wyniki oceny trudności w kategorii odwzorowania struktury danych w kodzie oraz dostępu do zagnieżdżonych danych. Wykorzystano trójstopniową skalę: łatwy, przeciętny, trudny.

Tabela 1. Wyniki badań trudności obsługi przetwarzania danych

	PostgreSQL	**MongoDB**	**Neo4J**
Trudność odwzorowania w kodzie	trudny	łatwy	trudny
Trudność dostępu do zagnieżdżonych danych	trudny	łatwy	łatwy

5. Dyskusja otrzymancyh wyników

W kategorii wstawiania danych, bazą o najkrótszym czasie odpowiedzi jest MongoDB. Zwycięża prawie dwukrotnie z Neo4J w teście 2. W przypadku pobierania danych, czas odpowiedzi zależy od sposobu przechowywania danych. Bazy PostgreSQL oraz Neo4J zapisują poszczególne elementy modelu (np. *UMLPackage*, *UMLCLass*) w oddzielnych strukturach danych. Dla tych baz widoczne jest zmniejszenie czasu odpowiedzi wraz z zawężeniem zakresu pobieranych danych – pobieranie samej klasy w teście nr 5 jest szybsze porównując z testami nr 3 i 4. Pomiary dla MongoDB potwierdzają wspomnianą powyżej zależność. MongoDB nie posiada oddzielnej struktury danych dla poszczególnych elementów, ponieważ jest to baza dokumentowa i przechowuje cały projekt UML w pojedynczym dokumencie. Wykazuje więc wzrost czasu dostępu do zagnieżdżonych danych ze względu na wymagane przetwarzanie danych po ich pobraniu w celu uzyskania żądanych danych. W teście nr 5 wspomniane przetwarzanie to przekształcenie danych oraz wyszukanie klas *UMLClass* o zadanej nazwie (Rys. 3.).

W przypadku pomiarów pamięci najważniejsze są nie tyle wartości bezwzględne, co różnice pomiędzy kolejnymi seriami badań, ponieważ istotna jest wielkość wzrostu pamięci wraz ze wstawianiem kolejnych danych. Im większe wartości, tym zapotrzebowanie na pamięć bardziej wzrasta (Rys. 4.).

Tempo przyrostu zajmowanej pamięci jest najwyższe dla bazy MongoDB i wynosi ok. 100-krotność (dla pozostałych ok. 10-krotność), tzn. rozmiar pamięci po wstawieniu danych jest ok. 100 razy większy niż przed ich wstawieniem. Oznacza to również, że dla każdego wstawionego projektu UML zajmowana pamięć zwiększa się o stałą wartość równą wielkości pamięci dla pierwszego projektu. Bazy PostgreSQL oraz Neo4J lepiej zarządzają pamięcią, ponieważ wraz z wstawianiem kolejnych danych przyrost pamięci zmniejsza się – wstawienie kolejnych 100 projektów UML zwiększyło zajmowaną pamięć przez te bazy tylko około 10-krotnie.

Wysoka trudność odwzorowania w kodzie dla bazy PostgreSQL oraz Neo4J wynika z wymogu stworzenia w kodzie programu pełnej struktury obiektów, które definiują poszczególne fragmenty modelu, np. przypadki użycia, klasy, aktywności(Tabela 1). Wszystkie te obiekty wymagają zdefiniowania pól dostępnych dla każdego z nich, za pomocą których program będzie tworzył żądania do bazy danych. Każdy nowy fragment modelu o unikatowych właściwościach wymaga utworzenia unikatowego dla tego obiektu odwzorowania. Natomiast dla bazy MongoDB utworzenia odwzorowania jest opcjonalne, ponieważ akceptuje ona projekt w postaci mapy tj. kolekcji klucz – wartość. Odwzorowanie może być również częściowe. W MongoDB przekształcanie danych oraz ich przekazanie do bazy jest mniej skomplikowane, a cały projekt może być zapisany lub odczytany przy pomocy jednego żądania.

Dostęp do zagnieżdżonych danych w bazie PostgreSQL jest dużo trudniejszy niż w dwóch pozostałych, ponieważ

wymaga tworzenia rozbudowanych żądań zawierających wiele zależności. Kolejne żądania wymagają identyfikatorów elementów nadrzędnych, aby możliwe było ich wykonanie. Powoduje to dłuższy czas oczekiwania na dane oraz bardziej złożoną implementację. W MongoDB dostęp do danych zawieranych przez obiekt jest prostszy – odbywa się za pomocą znaku kropki („.”), a w Neo4J na podobnej zasadzie – przy użyciu symbolu relacji („-->” lub „-[nazwa]->”).

6. Podsumowanie i wnioski

Celem niniejszej pracy była analiza porównawcza efektywności składowania oraz dostępu do modeli UML na przykładzie wybranych technologii bazodanowych. Przeprowadzone analizy potwierdzają prawdziwość hipotezy, iż baza dokumentowa reprezentowana przez MongoDB jest najlepszym wyborem w kwestii efektywnej obsługi struktur danych modeli UML. Technologia MongoDB zwycięża w 4 z 5 testów wydajności. Jedynie pobieranie danych zagnieżdżonych powoduje pewne problemy i wymaga mocy obliczeniowej aplikacji testowej. Jednak problem ten w większości sytuacji może być zminimalizowany poprzez przeniesienie procesu przetwarzania na stronę serwerową posiadającą znacznie większe zasoby sprzętowe.

Dodatkowo baza MongoDB jest najlepszym wyborem ze względu na trudność odwzorowania struktury danych modeli UML w kodzie aplikacji – pozwala na stworzenie aplikacji nie zwierającej takiego odwzorowania, a sama baza go nie wymaga. Poza tym dostęp do zagnieżdżonych danych, podczas tworzenia zapytań, jest wśród wszystkich testowanych technologii najłatwiejszy.

W kategorii wielkości zajmowanej pamięci baza MongoDB odznacza się najgorszym wynikiem na tle dwóch pozostałych. Nie wykazała ona widocznych optymalizacji rozrostu pamięci i pomimo małego rozmiaru początkowego już dla 100 wstawionych projektów zużycie pamięci zrównuje się z bazą PostgreSQL przechowującą tę samą liczbę projektów. Należy zaznaczyć, że otrzymany współczynnik przyrostu pamięci jest liniowy, dlatego nie można powiedzieć, że baza MongoDB jest nieodpowiednia do przechowywania danych struktur modeli UML, jednak należy zwrócić uwagę na zapewnienie jej odpowiednio dużej przestrzeni dyskowej, co ułatwia jej wysoki poziom skalowalności w poziomie.

Test nr 5 pokazuje, że technologia baz dokumentowych reprezentowanych przez MongoDB, nie radzi sobie dobrze z pobieraniem obiektów zagnieżdżonych – wymaga pobrania całego dokumentu oraz późniejszego jego filtrowania. Z tego powodu wraz ze wzrostem poziomu zagnieżdżeń obiektów danych (na poziom o wiele wyższy niż wykorzystany w testowym projekcie UML), wartą rozważenia będzie baza Neo4J ze względu na łatwy sposób budowania zapytań oraz uzyskane korzystne pomiary.

Literatura

[1] The Coming Software Apocalypse. https://www.theatlantic.com/technology/archive/2017/09/saving-the-world-from-code/540393/ [05.05.2019].

[2] Jung M., Youn S., Bae J. Choi Y.: A Study on Data Input and Output Performance Comparison of MongoDB and PostgreSQL in the Big Data Environment. 8th International Conference on Database Theory and Application (DTA). IEEE, 2015.

[3] Plechawska-Wójcik M., Rykowski, D.: Comparison of relational, document and graph databases in the context of the web application development. 36th ISAT Conference, Part II, s. 3-13. Springer, Cham, 2015.

[4] Van der Veen J. S., Van der Waaij B., Meijer R. J.: Sensor Data Storage Performance: SQL or NoSQL, Physical or Virtual. Fifth International Conference on Cloud Computing, Honolulu, s. 431-438. IEEE, 2012.

[5] Oussous A., Benjelloun F. Z., Lahcen A. A., Belfkih S.: Comparison and classification of nosql databases for big data. International Journal of Database Theory and Application, 6, April, 2013.

[6] Shetty Deepika V., Chidimar S. J.: Comparative Study of SQL and NoSQL Databases to evaluate their suitability for Big Data Application. International Journal of Computer Science and Information Technology Research, June, s. 314-318. 2016.

[7] Kunda, D., Phiri, H.: A Comparative Study of NoSQL and Relational Database. Zambia ICT Journal. Tom 1, s. 1-4. 2017

[8] Bathla, G., Rani, R., Aggarwal, H.: Comparative study of NoSQL databases for big data storage. International Journal of Engineering & Technology, 2018.

[9] Mondal, A. S., Sanyal, M., Chattopadhyay, S., Mondal, K. C.: Comparative Analysis of Structured and Un-Structured Databases. International Conference on Computational Intelligence, Communications, and Business Analytics, s. 226-241. Springer, Singapore, March, 2017.

[10] Introduction - StarUML documentation. https://docs.staruml.io/[05.05.2019].

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 12 (2019) 199-205
Wysłane: 2019-06-04
Przyjęte: 2019-06-11

Metody rozpoznawania gatunków grzybów na podstawie zdjęcia

Kamil Chodoła*, Grzegorz Czyż, Maria Skublewska-Paszkowska

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

Streszczenie. Celem artykułu jest porównanie dwóch metod rozpoznawania gatunków grzybów. W artykule zostały opisane dwie metody oparte na jednych z najpopularniejszych rozwiązań w dziedzinie image recognition, czyli Tensorflow oraz OpenCV. Do przeprowadzenia badań stworzono aplikację mobilną, w której obie metody zostały zaimplementowane oraz przetestowane. Dodatkowo aplikację wyposażono w mechanizmy ułatwiające zbieranie danych o aplikacji oraz algorytmach. Rezultaty badań wykazały, iż metoda oparta o Tensorflow o 9% skuteczniej rozpoznaje gatunki grzybów.

Słowa kluczowe: OpenCV;Tensorflow; image recognition

*Autor do korespondencji.

Adres e-mail: kamil.chodola@gmail.com

Methods for recognizing mushroom species on the basis of the photo

Kamil Chodoła*, Grzegorz Czyż, Maria Skublewska-Paszkowska

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

Abstract. The aim of the article is to compare two methods for identifying mushroom species. In article, two methods based on one of the most popular solutions in the field of image recognition, Tenosorflow and OpenCV, have been described. A research application was created to carry out the research, in which both algorithms were implemented and tested. In addition, the application was equipped with mechanisms facilitating the collection of application data and algorithms. The results of the research have show that the method based on Tensorflow by 9% more effectively recognizes mushroom species.

Keywords: OpenCV;Tensorflow; image recognition

*Corresponding author.

E-mail address: kamil.chodola@gmail.com

1. Wstęp

Szacuje się, iż obecnie w Polsce istnieje około 5 tysięcy gatunków grzybów[6]. Wśród nich istnieje wiele gatunków, które w mniejszym bądź większym stopniu zagrażają zdrowiu lub życiu człowieka po zjedzeniu lub nawet kontakcie z nimi. Co gorsza wiele popularnych, jadalnych grzybów jest podobnych do grzybów, które nie są zdatne do spożycia dla człowieka. Sprawia to, iż nawet osoby z dużym doświadczeniem w grzybiarstwie popełniają katastrofalne w skutkach błędy. Według statystyk, z roku na rok spada liczba zatruć spowodowanych spożyciem trujących grzybów, jednak ciągle poszukiwane są rozwiązania, które mogą spowodować zmniejszenie występowania tych przypadków do minimum.

W niniejszymartykule uwaga skupiać będzie się na metodach opartych o biblioteki OpenCV oraz Tensorflow. Do celów badawczych została utworzona aplikacja mobilna, w której zostały zaimplementowane wyżej wymienione metody. Po przeprowadzeniu badań została przeanalizowana poprawność zwracanych wyników, czas w jakim wyniki zostały zwrócone oraz wpływ danej metody na zasoby urządzenia. Na podstawie wyników tej analizy przedstawione zostały wnioski, dzięki którym wskazana została lepsza metoda oraz argumenty przemawiające za tym wyborem.

2. Metody rozpoznawania obiektów

Wykonane badania dotyczą zastosowania dwóch szkieletów aplikacji (ang. frameworków) w celu rozpoznawania obiektów na fotografiach. Są to: Tensorflow oraz OpenCV. W celu ich zaimplementowania w aplikacji mobilnej użyto dwóch nakładek (ang. wrapperów): OpenCVSharp oraz TensorflowSharp.

2.1. Tensorflow

Tensorflow jest jedną z najpopularniejszych bibliotek rozwiązujących problemy związane z uczeniem maszynowym (ang. machine learning) oraz głębokim uczeniem (ang. deep learning)[7]. Została ona zaprojektowana oraz wydana przez Google 9 listopada 2015 rokujako open-source. Tensorflow składa się z trzech modeli [4]:

- data model,
- execution model,
- programming model.

Data model składa się z tensorów, czyli podstawowych jednostek danych tworzonych, manipulowanych i zapisywanych w programie Tensorflow. *Execution model* polega na odpaleniu węzłów wykresu obliczeniowego w sekwencji zależności. Wykonanie rozpoczyna się od uruchomienia węzłów, które są bezpośrednio połączone z wejściami i zależą tylko od obecnych wejść.*Programming model* obejmuje wykresy przepływu danych lub wykresy obliczeniowe.

Jedną z największych zalet tego frameworku jest TensorBoard [3]. To narzędzie służy do wizualizacji danych. Dzięki niemu można w czytelny i prosty sposób prezentować m. in. dane wejściowe oraz wyjściowe, grafy obliczeniowe,

a nawet postęp treningu. Wizualizacja ułatwia wykrywanie błędów w strukturze sieci neuronowych oraz jej optymalizację.

Drugą bardzo użyteczną zaletą jest możliwość w łatwy sposób wykorzystywania tego samego modelu w wielu projektach bez względu na język programowania. W przypadku innych frameworków chcąc załadować parametry modelu wymuszane jest podanie całego kodu. Wiąże się to często z koniecznością przepisania kodu i ponowną kompilacją. Tensorflow natomiast potrzebuje jedynie pliku z wagami oraz nazwami warstw używanych do inferencji (najważniejsza jest warstwa wejściowa, ponieważ nie można bez niej wykonać grafu obliczeniowego).

Kolejną zaletą jest wsparcie wielu języków programowania takich jak:Python,Java,JavaScript,C++,Swift.

Dostęp do wszystkich funkcjonalności posiada jedynie Python, ponieważ jest najczęściej używany. Na potrzeby korzystania ze szkieletu aplikacji w niewspieranych językach zostały utworzone wiązania, dzięki czemu może być on używany w takich językach jak Ruby, czy C#. Biblioteka posiada również wady. Największą z nich jest nieuporządkowany kod. Nazwy wbudowanych metod nie pozwalają jednoznacznie stwierdzić, której z nich należy użyć, a wiele z nich nazywa się bardzo podobnie. Dokumentacja nie zawsze zawiera wszystkie informacje o instrukcjach jakie należyzawrzeć w kodzie, aby pewne rzeczy działały prawidłowo. Zdarzają się przypadki braku kompatybilności wstecznej niektórych funkcjonalności w wydawanych nowych wersjach, które pojawiają się dość często, bo nawet co 1-2 miesiące.

2.2. OpenCV

OpenCV (Open Source ComputerVisionlibrary) jest biblioteką open-source przeznaczoną na wiele platform, która dostarcza elementy dla aplikacji dot. wizji komputerowej (ang. computervision) [2]. Zapewnia wysokopoziomowe interfejsy do przechwytywania, przetwarzania i prezentacji danych obrazu. Na przykład wyodrębnia szczegóły dotyczące sprzętu kamery i alokacji macierzy. OpenCV jest szeroko stosowana zarówno w środowisku akademickim, jak i w rozwiązaniach deweloperskich [1].

Biblioteka ta została stworzona w języku C oraz C++, przez co jest mocno zoptymalizowana oraz wykorzystuje w pełni możliwości procesorów wielordzeniowych, co dla aplikacji czasu rzeczywistego jest niezwykle istotne. Obecnie rozwijana jest przy użyciu wrapperów w różnych językach programowania wśród których są:C#,Python,Ruby,Java,Matlab.

W dzisiejszych czasach wizja komputerowa może docierać do konsumentów w wielu kontekstach poprzez kamery internetowe, telefony z aparatami fotograficznymi i czujniki gamingowe, takie jak Kinect. OpenCV może pomóc w poszukiwaniu rozwiązań dla tych wymagań w języku wysokiego poziomu i w znormalizowanym formacie danych, który jest interoperacyjny z różnymi bibliotekami naukowymi. Rozbudowany zestaw narzędzi wspierających wykrywanie krawędzi oraz segmentację obrazu w bardzo skuteczny sposób potrafi wyabstrahować ze zdjęcia pożądane obiekty.

2.3. TensorflowSharp

Jest to wrapper umożliwiający wykorzystanie Tensorflow (które domyślnie zostało stworzone w Pythonie) na platformie .NET [8]. W tym przypadku jednak wrapper ten korzysta z API utworzonego w języku C. Jego źródła są dostępne publicznie, zatem aby rozpocząć korzystanie z niego wystarczy te źródła skompilować. Istnieje również możliwość zainstalowania dodatku do projektu w Visual Studio poprzez manager pakietów Nuget.

2.4. OpenCVSharp

Wrapper, dzięki któremu programiści C# mają możliwość tworzenia aplikacji zarówno mobilnych, jak i webowych lub desktopowych z użyciem OpenCV [5]. Implementuje on natywne metody OpenCV w sposób umożliwiający ich wykorzystanie przy rozpoznawaniu obiektów. Wiele klas zaimplementowanych w tym frameworku implementuje interfejs IDisposable, przez co programista nie musi martwić się o zarządzanie niebezpiecznymi zasobami pamięci. Dodatkowo OpenCVSharp nie wymusza programowania obiektowego. Wszystkie funkcje natywne można wywołać jedną prostą komendą. Do większości z nich należy przekazać obiekt typu Mat lub IplImage, które są reprezentacją obrazów przekonwertowanych w odpowiedni sposób. Problematyczne zatem jest zwrócenie z takiego bytu obrazu, który można wykorzystać do wyświetlenia rezultatu. OpenCVSharp jednak udostępnia metody BitmapFactory, dzięki którym istnieje możliwość przekonwertowania wielu obiektów na prosta bitmapę.

3. Cel i plan badań

Celem badań jest przeprowadzenie porównania metod rozpoznawania gatunków grzybów na podstawie zdjęcia. Należy potwierdzić lub odrzucić poniższą tezę:

Metoda oparta o bibliotekęTensorflow jest skuteczniejsza w rozpoznawaniu gatunków grzybów od metody opartej o OpenCV.

Do celów badawczych została utworzona aplikacja mobilna, w której zaimplementowano metody oparte o frameworkiTensorflow oraz OpenCV. Klasyfikatory/sieci szkolone były na komputerze stacjonarnym o następujących parametrach:

- procesor: AMD FX-6300 3,5 GHz,
- RAM: 8GB DDR3 2133MHz,
- karta graficzna: AMD Radeon R7 200 Series 1GB DDR5,
- dysk: SSD 60GB.

Zostały przygotowane zbiory danych dla następujących gatunków grzybów: Borowik szlachetny, Czubajka kania, Pieczarka, Muchomor czerwony, Pieprznik jadalny. Przy wyborze gatunków grzybów czynnikiem wiodącym była różnorodność budowy, rozmiarów oraz ich kolorystyki. Dla każdego z nich zostały pobrane zdjęcia w takiej liczbie, aby sumarycznie na nich oznaczyć 1000 obiektów wzorcowych.

OpenCV do rozpoznawania obiektów ze zdjęć wykorzystuje wcześniej wygenerowane pliki wzorcowe, w których zawarte są szczegółowe informację na temat danego obiektu. Aby przystąpić do procesu trenowania sieci neuronowej w przypadku Tensorflow, bądź klasyfikatora

w OpenCV, oprócz uprzednio przygotowanego zbioru zdjęć grzybów, nazywanego dalej zbiorem pozytywnym, należy przygotować również odpowiedni zbiór zdjęć negatywnych. Zdjęcia negatywne to takie na których obiekt nie występuje, ale zawierają tło na którym może on występować. Kolejnym krokiem jest utworzenie pliku .vec, który będzie zawierał obrobione zdjęcia pozytywne. Aby go stworzyć należy wykorzystać metodę opencv_createsamples przekazując następujące argumenty:

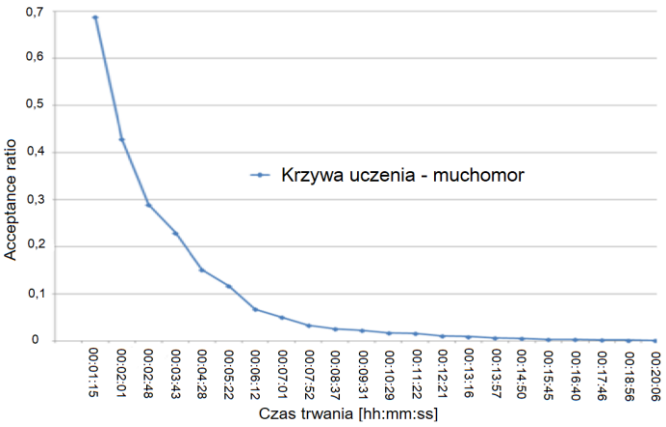
- info - ścieżka do pliku, w którym umieszczone są lokalizację zdjęć pozytywnych. Dodatkowo do każdego zdjęcia należy dodać informację o liczbie wystąpień obiektu na zdjęciu, lokalizację obiektu oraz jego rozmiar,
- num - liczba zdjęć,
- vec - ścieżka do zapisu pliku wynikowego,
- w - szerokość zdjęć pozytywnych,
- h - wysokość zdjęć pozytywnych,
- bg - ścieżka do pliku ze spisem zdjęć negatywnych.

Posiadając plik z przykładami zdjęć pozytywnych można rozpocząć pracę nad tworzeniem nowego wzorca na potrzeby identyfikacji. Należy skorzystać wówczas z metody opencv_traincascade do której należy przekazać odpowiednie argumenty:

- data - ścieżka do zapisu wyniku,
- vec - uprzednio wygenerowany plik zdjęć pozytywnych,
- bg - ścieżka do pliku ze spisem zdjęć negatywnych,
- numPos - liczba zdjęć pozytywnych,
- numNeg - liczba zdjęć negatywnych,
- numStages - liczba etapów szkolenia,
- w - szerokość zdjęć pozytywnych,
- h - wysokość zdjęć pozytywnych,
- featureType - typ cech na podstawie których tworzony będzie wzorzec,
- minHitRate - minimalne trafienie klasyfikatora podczas weryfikacji efektu szkolenia.

Najistotniejszym parametrem jest featureType, czyli rodzaj cech na podstawie których rozpoznawany jest obiekt. W tym przypadku został zastosowany typ LBP (ang. LocalBinaryPatterns)[2]. Polega on na konwersji obrazu na skalę szarości. Dla każdego piksela w obrazie w skali szarości wybierane jest sąsiedztwo o rozmiarze r, otaczające piksel środkowy. Wartość LBP jest następnie obliczana dla tego środkowego piksela i zapisywana w wyjściowej tablicy 2D z taką samą szerokością i wysokością jak obraz wejściowy. Proces progowania, gromadzenia łańcuchów binarnych i przechowywania wyjściowej wartości w tablicy jest następnie powtarzany dla każdego piksela. Wynikiem wykonania powyższych kroków jest plik wzorcowy, który jest wykorzystywany do rozpoznawania obiektów.

Szkolenie klasyfikatora opartego na OpenCV rozbijane jest na wiele etapów. Po każdym etapie w konsoli pojawia się informacja o aktualnej wartości parametru acceptanceRatio. Z przeprowadzonych testów wynika, iż najlepszym momentem na zakończenie szkolenia jest osiągnięcie wartości wcześniej wymienionego parametru w przedziale od 0,00003 do 0,00001. W celu wizualizacji wyników została utworzona krzywa uczenia dla muchomora widoczna na rysunku 1.



Rys. 1. Krzywa uczenia muchomora przy wykorzystaniu OpenCV

Jak widać w obu przypadkach wartość parametru acceptanceRatio dąży do zera. Wartość mieszcząca się w przedziale, o którym wspomniano wcześniej została osiągnięta w różnych przedziałach czasowych. Jest to spowodowane różną jakością przygotowanego zbioru fotografii dla obu gatunków.

Do identyfikacji stworzona została metoda w oparciu o wrapperOpenCVSharp [5], dzięki któremu istnieje możliwość tworzenia aplikacji oraz bibliotek w języku C# przy zachowaniu wszystkich funkcjonalności bibliotekiOpenCV. Dla każdego pliku wzorcowego wywoływana jest metoda FindMushroom, która pobrane zdjęcie przedstawia w skali szarości, następnie nakłada na zdjęcie maskę Otsu, która wyodrębnia ze zdjęcia najjaśniejsze fragmenty oraz krawędzie i tak obrobione zdjęcie przekazuje do metody DetectMultiScale (przykład1) przekazując następujące informacje:

- image - obiekt przechowujący zdjęcie,
- scaleFactor - współczynnik skalowania obrazu,
- minNeighbors - współczynnikokreślajacy minimalna liczbę nakładających się na siebie obszarów. Ustawiona np. wartość 4 określa, iż dla danego miejsca muszą zostać zwrócone minimalnie cztery nakładające się na siebie kwadraty, aby dany obszar został określony jako obiekt rozpoznany,
- flags - dodatkowy ustawienie wspierające określenie krawędzi na fotografii,
- minSize - minimalny rozmiar obiektu, który ma zostać zwracany.

Przykład 1. Metoda służąca do rozpoznawania grzybów na zdjęciu za pomocą OpenCVSharp

```
privatestaticvoidFindMushroom(File cascadeFile, Mat image,
List<Tuple<string, Rect, float>>mushrooms,
RectFmushroomRect, intoffsetY)
{
CascadeClassifiermushroom =
newCascadeClassifier(cascadeFile.AbsolutePath);
Mat ugray = new Mat();
Mat threshold = new Mat();
Cv2.CvtColor(image, ugray,
ColorConversionCodes.BGR2GRAY);
Cv2.Threshold(ugray, threshold, 0, 255,
ThresholdTypes.Otsu);

Rect[] mushroomsDetected =
mushroom.DetectMultiScale(threshold, 1.01, 4,
```

```
HaarDetectionType.FindBiggestObject,
newOpenCvSharp.Size(68, 80));

floatbestResult = 0;
string filename = cascadeFile.Name.Replace(".xml", "");
if (mushroomsDetected.Any())
mushrooms.Add(Tuple.Create(filename,
BestRect(mushroomsDetected, mushroomRect, out
bestResult, offsetY), bestResult));
}
```

W przeciwieństwie do metody opartej na OpenCV, Tensorflow do pełnego działania wymaga wygenerowania dwóch plików:

1) plik modelu (z rozszerzeniem .pb),
2) plik z nazwami obiektów (z rozszerzeniem .pbtxt).

Uprzednio przygotowany zbiór fotografii konkretnego gatunku grzyba należy odpowiednio spreparować w celu rozpoczęcia szkolenia modelu.

Pierwszym krokiem jest utworzenie pliku zawierającego wszystkie zdjęcia wzorcowe danego obiektu. Efekt ten można osiągnąć poprzez wykorzystanie metody tf.train.Example w Pythonie, która utworzy plik o rozszerzeniu .record. Po wykonaniu powyższego kroku dla wszystkich przygotowanych zbiorów fotografii należy również utworzyć w dokładnie ten sam sposób plik .record dla zbioru testowego tj. porcji fotografii wzorcowych, dzięki którym podczas szkolenia sieć neuronowa będzie mógł sprawdzać aktualną skuteczność.

Posiadając pliki z przykładami należy również utworzyć plik z nazwami oraz identyfikatorami obiektów (.pbtxt), które będą zawierały się w modelu. Zawarte w nim informacje to identyfikator obiektu oraz jego nazwa. Nazwa musi odpowiadać nazwie wykorzystanej w poprzednim kroku do oznakowania obiektów na zdjęciach. Plik ten będzie wykorzystany do trenowania sieci oraz oznaczania i opisywania znalezionych obiektów na fotografii.

Przed rozpoczęciem trenowania należy mieć przygotowany plik konfiguracyjny oraz plik “checkpoint”, czyli wstępnie wytrenowany model, który można doszkolić na swoje potrzeby. Tensorflow udostępnia przykładowe modele wraz z ich checkpointami oraz plikami konfiguracyjnymi, dzięki czemu rozpoczęcie tworzenia własnego modelu jest prostsze. Po pobraniu plików należy przystąpić do rekonfiguracji pliku .config. Należy w nim podać informacje między innymi o skalowaniu obrazu, lokalizacji pliku checkpoint, lokalizacji pliku z nazwami i identyfikatorami obiektów (.pbtxt) itp.

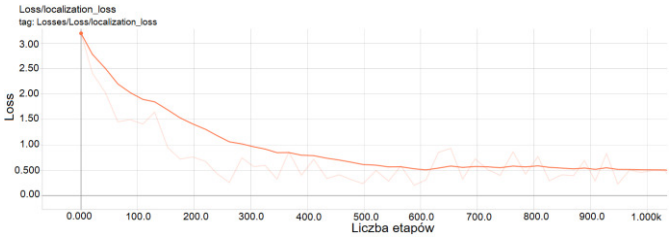
Po skompletowaniu wszystkich potrzebnych plików można rozpocząć trenowanie modelu. Tensorflow w swoich źródłach posiada plik train.py. Poniżej znajduje się przykładowe wywołanie tego pliku z poziomu linii poleceń:

train.py --logtostdderr --train_dir=path --pipeline_config_path=path

W przypadku wcześniej wspomnianej metody train.py użyta została jednokierunkowa sieć neuronowa. Jej główną cechą charakterystyczną są neurony ułożone w jednej warstwie, które są zasilane jedynie z węzłów wejściowych. Nie występują tutaj żadne sprzężenia zwrotne, tzn. każdy sygnał przekazany do sieci przechodzi przez każdy neuron tylko raz w całym swoim cyklu. Węzły te nie tworzą warstwy neuronowej, ponieważ nie zachodzą w nich żadne procesy obliczeniowe. Sieci jednokierunkowe można podzielić na: jednowarstwowe, dwuwarstwowe, wielowarstwowe. Wybór rodzaju sieci zależy od skomplikowania problemu, który ma zostać rozwiązany.

Sam proces trenowania należy kontynuować do momentu osiągnięcia parametru loss na poziomie od 0 do 1.Następnie po wytrenowaniu sieci należy skorzystać z pliku export_inference_graph.py udostępnionego przez Tensorflow. Korzysta on z utworzonego, dzięki poprzedniej metodzie, pliku checkpoint i generuje z niego model w rozszerzeniu .pb.

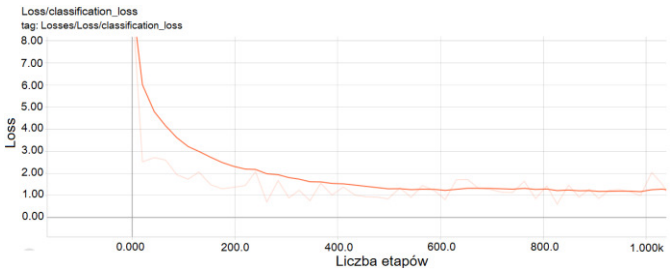
W przypadku sieci szkolnej w Tensorflow najważniejszym parametrem, który ma wpływ na jej jakość jest loss, dzięki któremu można stwierdzić jak dobrze wyszkolony jest model. Loss jest to wynik niezgodności pomiędzy przewidywaną wartością,a rzeczywistą etykietą [7]. Krzywe uczenia generowane są przy użyciu komendy tensorboard --logdirModelPath. Warto zwrócić uwagę na dwa wykresy, a mianowicie wykres przedstawiający niezgodność predykcji klasy obiektu oraz niezgodność lokalizacji obiektu.Jak można zauważyć na krzywej uczenia z rysunku 2 sieć oparta o Tensorflow najgorzej lokalizuje obiekt w początkowych etapach szkolenia. Następnie stopniowo spada, aż osiąga poziom około 0,5 loss, co jest bardzo dobrym wynikiem.



Rys. 2. Krzywa uczenia oparta o lokalizację obiektu

Świadczy to o tym, iż sieć lokalizuje obiekty bardzo trafnie. Wyniki te Tensorflow wnioskuje z porównania sieci w bieżącym stanie z przekazanym w pliku konfiguracyjnym zbiorem testowym w którym przekazane są informacje o lokalizacji obiektów znajdujących się na fotografiach.

Na rysunku 2 widać kolejne etapy szkolenia sieci i jej wyników dotyczących klasyfikowania obiektów. Są one nieco gorsze niż wyniki z krzywej uczenia z rysunku 3, co oznacza nieznacznie słabszą zdolność do rozpoznawania konkretnego gatunku.



Rys. 3. Krzywa uczenia oparta o klasyfikację obiektu

Wynik ten mógłby ulec poprawie w przypadku przekazania do sieci większej liczby zdjęć, jednak podczas jej

testowania uznano, iż jest ona satysfakcjonująca pod względem aktualnej skuteczności. Na rysunkach 2 oraz 3 widoczna jest druga linia, która również reprezentuje krzywą uczenia, jednak jest to wynik przed wygładzeniem wykresu. Opcja wygładzania jest wbudowana w TensorBoard.

Do identyfikacji stworzona została metoda w oparciu o wrapperTensorflowSharp [8] (przykład 2). Metoda ta jest dużo prostsza w implementacji, ponieważ składa się z trzech kluczowych fragmentów:

1) Utworzenie tensora na podstawie fotografii.
2) Utworzenie oraz uruchomienie runnera odpowiadającego za rozpoznanie grzybów na podstawie pliku modelu oraz pliku z opisami obiektów.
3) Skorzystanie z rezultatu w celu narysowania prostokątów na obrazie oznaczających rozpoznane obiekty.

Przykład 2. Metoda rozpoznająca gatunki grzybów przy użyciu TensorflowSharp

```
public static Bitmap Recognize(Activity activity, string
picturePath, RectFmushroomRect, out List<Tuple<string,
Rect, float>>mushrooms)
{
Activity = activity;

vargraph = newTFGraph();
varbytes =
System.IO.File.ReadAllBytes(_modelTuple.Item2.Path);
graph.Import(newTFBuffer(bytes));

varsession = newTFSession(graph);
var tensor =
ImageUtil.CreateTensorFromImageFile(picturePath,
TFDataType.UInt8);
varrunner = session.GetRunner();

runner
.AddInput(graph["image_tensor"][0], tensor)
.Fetch(
graph["detection_boxes"][0],
graph["detection_scores"][0],
graph["detection_classes"][0],
graph["num_detections"][0]);
varoutput = runner.Run();

varboxes = (float[,,])output[0].GetValue(jagged: false);
varscores = (float[,])output[1].GetValue(jagged: false);
varclasses = (float[,])output[2].GetValue(jagged: false);
varnum = (float[])output[3].GetValue(jagged: false);

return DrawBoxes(boxes, scores, classes, picturePath,
mushroomRect, out mushrooms,
MIN_SCORE_FOR_OBJECT_HIGHLIGHTING);
}
```

Jak można zauważyć metoda ta oprócz zwracania obszarów z potencjalnymi obiektami oraz ich nazwami zwraca również wynik. Wynik ten jest wyliczony bezpośrednio przez framework. W autorskiej aplikacji wynik ten jest dodatkowo sprawdzany poprzez porównanie obszarów wyznaczonych przez algorytm do obszaru narysowanego przez użytkownika podobnie jak to miało miejsce w przypadku metody opartej na OpenCV.

Po wyszkoleniu sieci/klasyfikatorów dla obu bibliotek przystąpiono do ich testowania oraz porównywania. W tym celu wykorzystano wcześniej napisaną aplikację oraz dwa telefony o różnych specyfikacjach.

Telefon 1:

- model: Huawei P20 Lite,
- RAM: 4GB,
- system: Android 8.0,
- procesor: HiSilicon Kirin 659,
- taktowanie procesora: 2360 MHz,
- aparat: 16 Mpx,
- rozdzielczość aparatu: 4608x3456 px.

Telefon 2:

- model: Xiaomi Pocophone F1,
- RAM: 6GB,
- system: Android 9.0,
- procesor: QualcommSnapdragon 845,
- taktowanie procesora: 2800 MHz,
- aparat: 12 Mpx,
- rozdzielczość aparatu: 4032x3024 px.

Badania zostały przeprowadzone przy użyciu aparatów telefonów w czterech różnych konfiguracjach. Niestety nie było możliwości skonfigurowania urządzeń do takich samych parametrów z powodu różnych wersji systemu. Zostały użyte zbliżone ustawienia, aby osiągnąć jak najbardziej porównywalne wyniki. W tym celu wybrano kadry 4:3 oraz podobne do siebie kadry 16:9 i 18:9. Dodatkowo w przypadku Huawei'a wybrane zostały rozdzielczości aparatu 5,8,10,16 MPx. W Xiaomi były do wyboru jedynie trzy opcję: słaba jakość, normalna jakość, bardzo dobra jakość. Do badania zostały wybrane dwa skrajne ustawienia oraz zmieniany był kadr. Same zdjęcia były wykonywane w takich samych warunkach otoczenia przy zachowaniu tego samego kąta ustawienia aparatu.

4. Wyniki badań

Metody zostały porównane, zwracając szczególna uwagę na następujące aspekty:skuteczność rozpoznawania,czas pracy,wpływ na zużycie zasobów.

4.1. Skuteczność rozpoznawania

Dla każdego gatunku grzyba został przygotowany zestaw 50 zdjęć zawierających jeden egzemplarz danego gatunku. Żadne z tych zdjęć nie znajdowało się wśród zdjęć pozytywnych wykorzystywanych przy szkoleniu sieci/klasyfikatorów. Są to zdjęcia w naturalnym środowisku, które zostały przed samym badaniem odpowiednio przygotowane. Poniżej przedstawiono wyniki pozytywnej weryfikacji grzyba, czyli gdy gatunek grzyba miał najlepszy wynik procentowy spośród wszystkich dostępnych.Z tabeli 1 można odczytać, iż nieznacznie lepsze wyniki osiągnęła metoda oparta na Tensorflow.

Tabela 1. Wyniki rozpoznawania gatunków grzybów przy użyciu poprawnych zdjęć

	OpenCV	**Tensorflow**
Muchomor czerwony	66%	78%
Prawdziwek	58%	74%
Kurka	48%	58%
Pieczarka	42%	44%
Kania	50%	58%

Dodatkowo w tabeli widać, iż tylko w trzech przypadkach osiągnięta została skuteczność poniżej 25 poprawnie

rozpoznanych gatunków, czyli poniżej 50%. Najlepszy wynik został osiągnięty w przypadku muchomora. Może to być wynikiem lepszego zbioru zdjęć oraz bardziej charakterystycznym wyglądem. Z drugiej strony najsłabiej wypadła pieczarka, która przez swój jednolity kolor oraz budowę zbliżoną do nie w pełni rozwiniętego muchomora lub prawdziwka często była z nimi mylona.

4.2. Czas pracy

Następnym czynnikiem, który został wzięty pod uwagę był czas jaki kluczowe metody służące do detekcji potrzebowały na obliczenie wyniku. Pomiar czasu odbywał się poprzez otoczenie kluczowej metody kodem mierzącym czas wykonania danego bloku operacji. Pomiary dla każdych ustawień urządzeń zostały wykonane dziesięciokrotnie, a następnie z otrzymanych wyników zostały wyciągnięte średnie czasy. W ten sposób osiągnięto wyniki z dokładnością do sześciu miejsc po przecinku, które następnie zostały zaokrąglone do czterech miejsc po przecinku. W tabelach 2 oraz 3 zostały przedstawione średnie czasy działania metody opartej na OpenCV w wybranych konfiguracjach aparatu.

Tabela 2. Czas działania metody opartej na OpenCV na telefonie marki Huawei

	3264 x 1616 18:9 (Aparat)	3264 x 2448 4:3 (Aparat)	4608 x 2272 18:9 (Aparat)	4608 x 3456 4:3 (Aparat)
Czas działania[s]	10,1720	9,9728	9,6862	10,2009

Tabela 3. Czas działania metody opartej na OpenCV na telefonie marki Xiaomi

	Słaba jakość 4:3 (Aparat)	Dobra jakość 4:3 (Aparat)	Słaba jakość 16:9 (Aparat)	Dobra jakość 16:9 (Aparat)
Czas działania[s]	4,4345	4,5751	4,5028	4,8938

W zestawieniu dużo słabiej wypadło urządzenie Huawei, najprawdopodobniej ze względu na słabsze parametry sprzętowe. Nie miało to jednak wpływu na wyniki zwracane przez metody, które zostaną omówione w kolejnej części badania. Dodatkowo widać zależność rosnącej jakości oraz rozmiaru zdjęcia do czasu działania metody. Wraz z coraz mocniejszymi ustawieniami aparatu wzrastał również czas analizy fotografii. Wyniki mogły być delikatnie zakłócone poprzez aplikacje systemowe działające w tle.

Następnie przy użyciu tych samych parametrów i przy użyciu tego samego zdjęcia wzorcowego zbadano metodę opartą na Tensorflow. Zmierzone czasy zostały przedstawione w tabelach 4 i 5.

Tabela 4. Czas działania metody opartej na Tensorflow na telefonie marki Huawei

	3264 x 1616 18:9 (Aparat)	3264 x 2448 4:3 (Aparat)	4608 x 2272 18:9 (Aparat)	4608 x 3456 4:3 (Aparat)
Czas działania[s]	11,3520	11,7354	11,6489	12,5733

Tabela 5. Czas działania metody opartej na Tensorflow na telefonie marki Xiaomi

	Słaba jakość 4:3 (Aparat)	Dobra jakość 4:3 (Aparat)	Słaba jakość 16:9 (Aparat)	Dobra jakość 16:9 (Aparat)
Czas działania[s]	4,9456	5,4237	5,3477	6,0123

Rezultaty potwierdzają wcześniej stwierdzony fakt korelacji pomiędzy wielkością zdjęcia, a czasem działania. Czasy są większe niż w przypadku rozwiązania opartego o OpenCV. Dodatkowym aspektem, który warto zaznaczyć jest spełnienie w przypadku Xiaomi początkowego założenia dotyczącego maksymalnego czasu działania metody, który wynosił 10 sekund. W przypadku Huaweia granica ta została delikatnie przekroczona, jednak uśredniając wszystkie wyniki średnia mieści się w normie. Ponadto Huawei dysponował lepszym aparatem tylnym niż Xiaomi co zwiększało szczegółowość zdjęcia oraz jego rozmiar co mogło wpłynąć na czas działania.

4.3. Wpływ na zużycie zasobów

Zużycie zasobów podczas tworzenia i testowania metod można podzielić na dwie kategorie:

1) Zużycie zasobów komputera podczas szkolenia algorytmu.
2) Zużycie zasobów telefonu podczas rozpoznawania obiektów na zdjęciach.

Podczas nauki sieci/klasyfikatorów opartych na OpenCV w zależności od momentu szkolenia w danym etapie następował wzrost obciążenia poszczególnych podzespołów. Podczas ładowania zdjęć pozytywnych i negatywnych elementami, które zostały najbardziej obciążone był procesor, dysk oraz nieznaczny wzrost pamięci RAM. Zużycie CPU wzrastało momentami do 100%, podobnie było w przypadku dysku, jednak wzrosty były na tyle krótkotrwałe, że nie wpływały w znaczącym stopniu na pracę komputera. Po załadowaniu zdjęć następowało właściwe szkolenie podczas którego wzrastało zużycie pamięci RAM i w tym przypadku trwało aż do zakończenia etapu. Maksymalne zużycie jest definiowane parametrycznie i jego domyślna wartość to 1GB. Metoda oparta na Tensorflow po uruchomieniu trenowania i wewnętrznej konfiguracji od razu przystępuje do szkolenia. Przy użyciu domyślnych ustawień framework ten pobiera tyle zasobów komputera ile tylko jest w stanie zaalokować. Efektem tego jest bardzo mocna praca komputera oraz duże trudności w innych działaniach.

Do celów pomiarowych została wykorzystana aplikacja “Monitor zasobów Mini”, która w czytelny sposób wyświetla na ekranie telefonu aktualnie dostępne zasoby sprzętowe. Pomiary zostały wykonane dziesięciokrotnie dla każdych ustawień urządzeń, z których została wyciągnięta średnia. Przy rozpoznawaniu obiektów za pomocą OpenCV odnotowano wzrosty zużycia zasobów na telefonach widoczne w tabelach 6 oraz 7.

Tabela 6. Wyniki zużycia zasobów na telefonie marki Huawei przy użyciu metody opartej na OpenCV

	3264 x 1616 18:9 (Aparat)	3264 x 2448 4:3 (Aparat)	4608 x 2272 18:9 (Aparat)	4608 x 3456 4:3 (Aparat)
RAM[MB]	1300	1250	1310	1290
CPU	30%	60%	64%	65%

Tabela 7. Wyniki zużycia zasobów na telefonie marki Xiaomi przy użyciu metody opartej na OpenCV

	Słaba jakość 4:3 (Aparat)	Dobra jakość 4:3 (Aparat)	Słaba jakość 16:9 (Aparat)	Dobra jakość 16:9 (Aparat)
RAM[MB]	1200	1240	1400	1370
CPU	70%	65%	73%	75%

Przy rozpoznawaniu obiektów za pomocą Tensorflow odnotowano wzrosty zużycia zasobów na telefonach widoczne w tabelach 8 oraz 9.

Tabela 8. Wyniki zużycia zasobów na telefonie marki Huawei przy użyciu metody opartej na Tensorflow

	3264 x 1616 18:9 (Aparat)	3264 x 2448 4:3 (Aparat)	4608 x 2272 18:9 (Aparat)	4608 x 3456 4:3 (Aparat)
RAM[MB]	1320	1330	1310	1330
CPU	55%	62%	64%	71%

Tabela 9. Wyniki zużycia zasobów na telefonie marki Xiaomi przy użyciu metody opartej na Tensorflow

	Słaba jakość 4:3 (Aparat)	Dobra jakość 4:3 (Aparat)	Słaba jakość 16:9 (Aparat)	Dobra jakość 16:9 (Aparat)
RAM	1600MB	1650MB	1710MB	1750MB
CPU	72%	73%	77%	80%

Z otrzymanych wyników zauważyć można, iż oprogramowanie zabiera większość dostępnych zasobów w celu analizy przekazanej fotografii. Wraz ze wzrostem jakości zdjęcia zauważalny jest wzrost wykorzystania zasobów sprzętowych. Różnica pomiędzy urządzeniami wynika głównie z różnicy ich specyfikacji. W przypadku Xiaomi istniała możliwość pobrania większej ilości pamięci RAM oraz większego wykorzystania procesora. Aplikacja w momencie rozpoznania była nieużywalna, jednak samo działanie urządzenia nie zostało zakłócone w znaczącym stopniu. Metoda stworzona w Tensorflow potrzebowała nieznacznie więcej zasobów.

5. Wnioski

Postawioną tezę:

Metoda oparta o bibliotekęTensorflow jest skuteczniejsza w rozpoznawaniu gatunków grzybów od metody opartej o OpenCV,

udało się udowodnić. W przypadku najważniejszego aspektu badawczego, czyli skuteczności rozpoznawania gatunków grzybów Tensorflow okazał się lepszy o 9%. Jednym z powodów tego stanu rzeczy może być fakt, iż Tensorflow korzysta ze zbioru testowego podczas trenowania sieci neuronowej, który zostaje ułożony przez programistę przed trenowaniem. Sprawia to, iż aktualny stan rozwoju sieci jest porównywany do dobrze dobranego zbioru wzorcowego. Dodatkowo Tensorflow jest zdecydowanie świeższym rozwiązaniem, ale jednocześnie wystarczająco dojrzałym, przez co posiada szeroką gamę funkcji, które usprawniają pracę oraz zapewniają lepsze rezultaty.

Warto jednak zwrócić uwagę nie tylko skuteczność metody, ale również na inne cechy, które zostały zbadane. Proces rozpoznawania gatunku grzyba trwał 1 do 2 sekund dłużej w przypadku Tensorflow. Czas trwania spowodował, że na urządzeniu o słabszej specyfikacji sprzętowej została przekroczona granica dziesięciu sekund wyznaczona w wymaganiach niefunkcjonalnych. Dodatkowym atutem przemawiającym za OpenCV jest mniejszy pobór zasobów urządzenia, na którym dokonuje się detekcji gatunku grzyba. Jest to bardzo ważny argument przemawiający na korzyść tego frameworka, ponieważ istnieje możliwość uruchomienia aplikacji z tą metodą na telefonach bądź komputerach o słabszej specyfikacji bez obawy o stabilność działania programu. Powodem tego stanu rzeczy może być fakt, iż pliki z których korzysta OpenCV podczas analizy obrazu są zdecydowanie mniejsze niż pliki modelu Tensorflow. Pliki te podczas detekcji są ładowane do pamięci, przez co jej zużycie znacząco wzrasta.

Literatura

[1] Baggio D. L.: Mastering OpenCV with Practical Computer Vision Projects, Packt Publishing Ltd, 2012.

[2] Bradski G., Kaehler A.: Learning OpenCV, O'Reilly 2008.

[3] He K., Zhang X., Ren S., Sun J.: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, s. 770-778.

[4] Krzyśko M., Wołyński W., Górecki T., Skorzybut M.: Systemy uczące się, Wydawnictwo Naukowo-Techniczne Sp. z o.o., 2008.

[5] OpenCVSharp, https://github.com/shimat/OpenCVsharp [07.05.2019].

[6] Staniszewski P.: Użytkowanie grzybów leśnych – praktyka i problemy badawcze, Stud. Mater. CEPL w Rogowie, 2014, s. 143 – 152.

[7] Tang J.: Intelligent Mobile Projects with Tensorflow, Packt Publishing Ltd. 2018.

[8] TensorflowSharp, https://github.com/migueldeicaza/TensorflowSharp [11.05.2019].

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 12 (2019) 206-213
Wysłane: 2019-06-03
Przyjęte: 2019-06-11

Metody zwiększające wydajność i bezpieczeństwo aplikacji internetowych

Tomasz Machulski*, Grzegorz Nowakowski*, Maria Skublewska-Paszkowska
Politechnika Lubelska, Instytut Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. Artykuł prezentuje metody zwiększające wydajność i bezpieczeństwo aplikacji internetowych oraz ocenia ich wartość i zasadność użycia. Każda z przedstawionych metod została zaimplementowana w aplikacji testowej. W pracy zostały przedstawione wyniki badań porównujących stan aplikacji przed i po implementacji danej metody. Na podstawie rezultatów badań sformułowano wnioski dotyczące wpływu metod wydajności i bezpieczeństwa na zachowanie aplikacji testowej.

Słowa kluczowe: wydajność aplikacji internetowych; bezpieczeństwo; aplikacje internetowe

*Autor do korespondencji.

Adresy e-mail: tomaszekem2@gmail.com, grz55@vp.pl

Methods of enhancing the performance and security of web applications

Tomasz Machulski*, Grzegorz Nowakowski*, Maria Skublewska-Paszkowska
Institute of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract. This article presents the methods of enhancing the performance and security of web applications. It also evaluates them and describes how to use them. The methods have been implemented in a test application. The article presents results of the research comparing state of the application before and after implementation of every listed method. Conclusions about impact of the methods of enhancing performance and security of web applications are based on the results of the research.

Keywords: performance of web applications; security; web-applications

*Corresponding author.

E-mail addresses: tomaszekem2@gmail.com, grz55@vp.pl

1. Wstęp

Rozwój Internetu oraz rosnąca prędkość łącz spowodował ogromny wzrost zainteresowania siecią Web jako obszarem funkcjonowania i utrzymania aplikacji. Jeszcze niedawno programy komputerowe instalowane były w większości na dyskach lokalnych, całość pracy programu wykonywana była przez komputer użytkownika, a aplikacje rzadko kiedy łączyły się z siecią. Wprowadzanie zmian i poprawek w takich programach, było utrudnione. Producentowi łatwiej zarządzać, modyfikować i uaktualniać program znajdujący się na jego serwerze, niż wprowadzać zmiany w nierzadko milionach komputerów.

Przeniesienie funkcjonowania aplikacji na serwer jest zaletą nie tylko dla właściciela oprogramowania, ale także użytkownika. Centralne utrzymania aplikacji przez twórcę oraz możliwość zdalnej pracy przez klienta to tylko jedne z przykładów, które zaważyły o sukcesie, które odniosły aplikacje internetowe.

Internet zawiera bardzo wiele ważnych danych. Użytkownik nierzadko zostawia tam swoją tożsamość lub wykonuje operacje finansowe na porządku dziennym. Ogromna liczba urządzeń użytkowników podłączonych do Internetu implikuje sytuację, że bazy danych przechowują coraz większe zbiory danych. Biorąc pod uwagę oba fakty, naturalnym jest stwierdzenie, że wydajność i bezpieczeństwo aplikacji internetowych jest bardzo znaczące. Rozwój technologii powoduje, że powstają coraz nowsze mechanizmy tworzenia aplikacji internetowych, jak i zwiększania ich wydajności. Mnogość informacji zawartych w bazach danych wymaga ścisłego kontrolowania wydajności systemu, mimo znacząco rosnącej prędkości dysków, procesorów i łączy sieciowych. Ważność danych znajdujących się na serwerach przyciąga hakerów, przez co na utrzymujących oprogramowanie internetowe ciąży duża odpowiedzialność zapewnienia bezpieczeństwa danych.

Ukazane przypadki wskazują, jak ważnym zagadnieniem jest utrzymanie wysokiej wydajności i bezpieczeństwa aplikacji internetowych. Istnieje wiele metod wspomagających oba zagadnienia. Dynamiczny rozwój Internetu sprawia jednak, że wymagają one ciągłych badań i rozwoju. Tematem pracy jest przeprowadzenie badań i zweryfikowanie postawionych tez: „Metody wydajnościowe frameworków Hibernate i Spring Boot w bardzo znaczący sposób poprawiają szybkość działania aplikacji internetowych", oraz „Metody zwiększające bezpieczeństwo są skutecznym sposobem ochrony aplikacji internetowych". Do zawartych w pracy metod zwiększających bezpieczeństwo należą: haszowanie haseł w bazie danych, użycie JSON Web Token, ustanowienie uprawnień użytkowników, konfiguracja nagłówków http oraz ustanowienie uprawnień użytkowników w bazie danych. Rosnący zbiór danych będących w sieci powoduje, że

zagadnienia ujęte w tezach mają obecnie bardzo duże znaczenie.

2. Przegląd literatury

Informatyka rozwija się bardzo szybko, co jest skutkiem prowadzenia badań tej dziedziny na szeroką skalę. Nie inaczej jest z obszarem aplikacji internetowych. Istnieje wiele sieciowych katalogów prac naukowych, takich jak IEEE Xplore, Scopus lub Web of Science, które publikują tematy w zbliżonej dziedzinie poprawy wydajności i bezpieczeństwa aplikacji internetowych.

Wydajność i optymalizacje aplikacji internetowych opartych o platformę Java EE były analizowane przez autorów publikacji [1]. Badano czasy odpowiedzi serwera przez zasymulowanie jego obciążenia za pomocą aplikacji JMeter. Badaną aplikacją była platforma dla studentów i wykładowców umożliwiająca udostępnianie ocen i obecności. Analiza wyników wskazuje, że baza danych jest głównym czynnikiem mającym wpływ na wydłużenie czasu odpowiedzi serwera przy wzroście obciążenia.

Analizę narzędzi używanych w wykrywaniu podatności na ataki na bazy danych aplikacji internetowych prezentuje praca [2]. Wysiłek autorów skupiony został na testowaniu zachowania aplikacji podczas ataku. Nie zostały tu poruszone zagadnienia metod zabezpieczających.

Przedstawienie sposobu projektowania i budowy systemów zabezpieczających aplikacje internetowe zawiera artykuł [3]. Praca przedstawia ogólny, teoretyczny proces modelowania zabezpieczeń w aplikacjach internetowych. Autorzy wskazują podstawowe metody zabezpieczające, lecz nie przeprowadzają ich głębszej analizy.

Metoda puli połączeń jako wysoce usprawniająca wydajność aplikacji internetowej wskazana zostaje w pracy [4]. Prawdziwość tezy o skuteczności tej metody jest udowodniona badaniami z użyciem narzędzia JMeter. Jest to wartościowe badanie, zawierające przykłady implementacji metody oraz jasno sformułowane wyniki, jednak nie wyczerpuje całkowicie tematu usprawniania wydajności aplikacji webowych.

Ukazane badania oraz wiele innych znajdujących się w licencjonowanych bazach danych przedstawiają głównie pojedyncze metody usprawniające i nie przedstawiają skali różnic przed oraz po ich zastosowaniu. Brakuje zgrupowania najlepszych metod i jasnego określenia stopnia w jaki zwiększają wydajność oraz bezpieczeństwo aplikacji webowych, dlatego uzasadnionym jest przeprowadzanie dalszych badań i analiz w tym kierunku.

W pracy zostały przedstawione i zanalizowane metody zwiększające wydajność należące do frameworków Hibernate oraz Spring. Metody zapewniające bezpieczeństwo dotyczą back-endu aplikacji oraz bazy danych.

3. Metody zwiększające wydajność

Do budowy i rozwijania aplikacji internetowych stosuje się frameworki, które stanowią szkielet aplikacji. Frameworki Hibernate i Spring Boot posiadają wiele możliwości usprawnienia wydajności. Użytkownicy wybierają odpowiednie metody w zależności od obszaru aplikacji, w której wydajność powinna być zwiększona.

3.1. Metody bazodanowe

W bazach danych globalnych systemów komercyjnych znajduje się ogromna liczba rekordów. Wraz ze wzrostem ilości danych utrudnione staje się przeszukiwanie danych w realnym czasie. Odpowiedzią na ten problem jest użycie indeksów bazodanowych.

Indeksowanie polega na utworzeniu odrębnej struktury danych, która przechowuje wartość danej kolumny oraz wskaźnik do rekordu z którym jest powiązany [5]. Struktura jest automatycznie sortowana binarnie po każdej manipulacji na danych tabeli. Indeksowanie należy wprowadzać rozważnie, gdyż indeksy zwiększają rozmiar danych. Są one automatycznie tworzone przez system bazodanowy na kolumnach zawierających wartości klucza głównego lub klucza unikalnego.

Wykorzystane polecenia języka SQL dodające indeksy w tabelach testowej bazy danych przedstawia przykład 1.

Przykład 1. Ustawienie indeksów na kolumnach w tabelach bazy danych

```
CREATE INDEX idx_contr_branch_id ON contract(branch_id);
CREATE INDEX idx_branch_name ON branch(name);
```

3.2. Metody we frameworku Hibernate

3.2.1. Pamięć podręczna II poziomu

Częste odwołania do bazy danych przez ich wysoki koszt, są niepożądane. Należy przez to minimalizować liczbę odniesień do bazy danych. Pamięć podręczna (ang. cache) jest usługą aplikacji pozwalającą na tymczasowe składowanie danych pochodzących ze źródła o wolniejszym dostępie [6]. Odczyt z danych zapisanych w cache jest dużo szybszy od dostępu bazodanowego.

Framework Hibernate w podstawowej konfiguracji wykorzystuje cache I poziomu. Podczas każdej wymiany informacji między aplikacją, a bazą danych tworzona jest sesja. W trakcie jej działania wszystkie wyniki zapytań do bazy danych są zapisywane do cache i poziomu.

W programie może istnieć wiele sesji równocześnie, a każda z nich ma wydzielony indywidualnie cache dla własnego użytku. Ponowne zapytanie o te same dane w ciągu tej samej sesji nie wymaga komunikacji z bazą danych.Po zakończeniu sesji dane w cache I poziomu zostają tracone.

W przypadku użycia pamięci podręcznej II poziomu, w aplikacji powstaje cache globalny, współdzielony przez wszystkie sesje. Sposób działania wszystkich poziomów cache jest identyczny, różnią się one tylko zasięgiem [7].

Objęcie danych pochodzących z tabeli bazodanowej mechanizmem pamięci podręcznej drugiego poziomu uzyskano dzięki dodaniu adnotacji *@Cache* nad nagłówkiem klasy mapującej tą tabelę. Przedstawiono to na przykładzie 2.

Przykład 2. Objęcie klasy reprezentującej tabelę bazodanową mechanizmem pamięci cache II poziomu

```
@Entity
@Table(name = "branch")
@Cache
public class Branch {
    @Id
    private Long id;
}
```

3.2.2. Cachowanie danych słownikowych

W bazie danych istnieją rekordy niezmienne. Pomimo częstego dostępu do nich, ich wartości nigdy nie są nadpisywane, a tylko odczytywane. Duży koszt dostępu do bazy danych sprawia, że zasadnym staje się przeniesienie tych danych do pamięci podręcznej na cały czas działania aplikacji [8].

Do zarejestrowania encji bazodanowej na stałe w cache przez Hibernate, użyto adnotacji *@Cache* z opcją *READ_ONLY* nad klasą reprezentującą tabelę. Zostało to przedstawione na przykładzie 3.

Przykład 3. Objęcie klasy reprezentującej tabelę bazodanową mechanizmem cache'owania danych słownikowych

```
@Entity
@Table(name = "authority")
@Cache(usage=CacheConcurrencyStrategy.READ_ONLY)
public class Authority implements Serializable {
    @Id
    private String name;
}
```

3.2.3. Query cache

W bazie danych istnieją tabele, których rekordy rzadko ulegają zmianom. Częsty dostęp do nich może być kosztowny. Rzadka zmiana rekordów implikuje małą częstotliwość zmiany wyników zapytań, dlatego framework Hibernate umożliwia przechowywanie tych wyników przez określony czas. Podczas wykonywania zapytania zarejestrowanego przez Query cache, jego wynik ładowany jest z pamięci podręcznej, co ogranicza liczbę odniesień do bazy danych [9]. Istnieje przez to ryzyko uzyskania nieaktualnych danych, dlatego zapytania objęte Query cache nie powinny dotyczyć danych krytycznych do funkcjonowania systemu.

Na przykładzie 4 za pomocą adnotacji *@Cacheable* objęto metodę wykonującą zapytanie do bazy danych. Hibernate rejestruje wyniki tego zapytania w pamięci podręcznej.

Przykład 4. Objęcie metody pobierającej dane z bazy danych mechanizmem Query cache

```
@Repository
interface NewsArticleRepository extends
JpaRepository<NewsArticle, Long> {
    String ARTICLES_BY_BRANCH_ID = "articlesByBranchId";
    @Cacheable(cacheNames=ARTICLES_BY_BRANCH_ID)
     Page<NewsArticle> findAllByBranchId(Long branchId,
    Pageable pageable);
}
```

3.2.4. Ładowanie danych

Podczas korzystania z frameworka Hibernate ładowanie danych ze źródeł może odbywać się według dwóch wzorców. Wyróżnia się zachłanny (ang. eager) oraz leniwy (ang. lazy) tryb dostępu do danych [10]. Pierwszy z nich inicjalizuje oraz przygotowuje wszystkie dane natychmiastowo i zwraca gotowy do wykorzystania obiekt reprezentujący daną encję. Zaleca się stosować go w przypadku pewności, że dane zostaną wykorzystane natychmiastowo.

Z kolei leniwy dostęp do danych przygotowuje obiekt z podstawowymi danymi, lecz odracza jego inicjalizację. Odraczane dane są asocjacjami pomiędzy encjami (relacje jeden-do-wielu, wiele-do-wielu). Ich inicjalizacja następuje dopiero podczas bezpośredniego do nich odniesienia.

Każdy z trybów dostępu posiada zalety i wady, dlatego należy stosować je zgodnie z przeznaczeniem. Ładowanie zachłanne może skrócić czas dostępu do danych, lecz odbywa się to nierzadko kosztem ładowania zbyt dużej ich ilości, co może powodować spadki wydajności. Ładowanie leniwe konsumuje mniej pamięci, gdyż większość ładowanych danych jest niezainicjalizowana. Dostęp do nich wymaga dodatkowego czasu, co może potencjalnie wpływać na spadek wydajności odczytu.

Na przykładzie 5 znajduje się klasa *Client* reprezentująca rekord tabeli przechowującej dane o użytkownikach. Pola tej klasy które mapują relacje wiele-do-wielu lub wiele-do-jednego reprezentowane są jako kolekcje w języku Java. Oznaczono tryb odczytu tych kolekcji jako leniwy (*LAZY*), tak aby ich odczyt został odroczony do momentu pierwszego odniesienia do nich.

Przykład 5. Objęcie klasy reprezentującej tabelę bazodanową mechanizmem cache'owania danych słownikowych

```
@Entity
public class Client extends User {
    @ManyToMany(
        cascade = CascadeType.ALL,
        fetch = FetchType.LAZY)
    private Set<GymClass> classes = Sets.newHashSet();
```

3.3. Metody we frameworku Spring Boot

3.3.1. Mechanizm Spring Cache

Framework Spring podobnie jak Hibernate wspiera mechanizm wykorzystania pamięci podręcznej. Spring Cache pozwala przechowywać wyniki metod klas wykonujących

kosztowne obliczenia [11,12]. Rzadsze wykonywanie obliczeń redukuje wykorzystanie mocy obliczeniowej serwera. Czas przechowywania wyniku w pamięci podręcznej ustala się w konfiguracji mechanizmu Spring Cache.

Na przykładzie 6 przedstawiono funkcję umieszczającą do pamięci podręcznej instancję obiektu będącego rezultatem obliczeń. Obiekt *cacheManager* przechowuje wartości obliczeń jako para klucz-wartość.

Przykład 6. Ładowanie obiektu do pamięci podręcznej za pomocą mechanizmu Spring Cache

```
public void setAsCurrent(Branch branch) {
    var cache= cacheManager.getCache(BRANCH_CACHE);
    cache.clear();
    cache.put(CURRENT_BRANCH_KEY, branch);
}
```

3.3.2. Paginacja danych

Paginacja jest mechanizmem polegającym na ładowaniu ograniczonej ilości danych pochodzących z bazy danych. Użytkownik aplikacji internetowej nie potrzebuje zwykle wizualizacji wszystkich danych jednocześnie. Dzięki temu aplikacja może pobierać dane z bazy danych porcjami i eliminuje problem wczytywania nadmiaru danych [13]. Wpływa to również pozytywnie na szybkość wyświetlania interfejsu użytkownika oraz redukuje obciążenie bazy danych.

Mechanizm paginacji został zaimplementowany we frameworku Spring. Polega na przekazaniu argumentu typu *Pageable* poczynając od metody klasy kontrolera aż po metodę klasy wykonującej zapytanie do bazy danych. Instancja tego argumentu jest tworzona automatycznie przez framework Spring na podstawie parametrów żądania http. Metodę klasy kontrolera obsługującą paginację zaprezentowano na przykładzie 7.

Przykład 7. Metoda klasy kontrolera obsługująca paginację danych za pomocą parametru klasy *Pageable*

```
@GetMapping("/users")
List<UserDTO>getUsers(Pageable pageable) {
    var page = userService.getAllManagedUsers(pageable);
    return page.getContent();
}
```

4. Metody zwiększające bezpieczeństwo

4.1. Haszowanie haseł w bazie danych

Dostęp do większości aplikacji internetowych zabezpieczany jest mechanizmem logowania. Sposobem na utrudnienie odczytania hasła, nawet po jego wykradzeniu jest haszowanie. Polega ono na użyciu tzw. funkcji skrótu, która generuje unikalny ciąg znaków dla dowolnego ciągu wejściowego [14]. Funkcja ta jest nieodwracalna. Na podstawie uzyskanego wyjściowego ciągu znaków nie można określić wejściowego ciągu znaków. Wszelkie próby operacji odwrotnych wymagają ogromnej mocy obliczeniowej i przez to stają się niedostępne dla zwykłych użytkowników.

W aplikacji testowej wykorzystano mechanizm haszowania haseł zaimplementowany we frameworku Spring. Na przykładzie 8 znajduje się fragment kodu odpowiadający za hashowanie podanego hasła podczas rejestracji użytkownika. Metoda *encode* obiektu *passwordEncoder* wykorzystuje funkcję hashującą *bcrypt*.

Przykład 8. Kod odpowiadający za haszowanie hasła użytkownika

```
var passwordHash = passwordEncoder.encode(password);
```

4.2. JSON Web Token

Każda aplikacja internetowa powinna chronić swoje dane przed niepowołanym dostępem. Stosuje się do tego autentykację polegającą na potwierdzaniu tożsamości użytkownika oraz autoryzację, która nadaje dostęp do systemu.

JSON Web Token (JWT) jest standardem definiującym bezpieczną wymianę informacji pomiędzy klientem, a serwerem [15]. Nośnikiem tej informacji jest obiekt JSON. Po poprawnym zalogowaniu się, klient otrzymuje token będący zaszyfrowanym obiektem JSON. Token przechowuje dane zaszyfrowane kluczem szyfrującym, który znajduje się na serwerze. Tylko on potrafi zweryfikować poprawność tokenu przychodzącego wraz z żądaniami http.

JWT dzięki użyciu klucza szyfrującego nie jest możliwy do spreparowania. Odszyfrowanie danych przez serwer stanowi wystarczające zweryfikowanie tożsamości użytkownika bez konieczności ponownych zapytań do bazy [16]. Oprócz zapewnienia bezpieczeństwa, daje to pozytywny wpływ na wydajność zapytań http.

Obiekt JSON składa się z nagłówka, danych oraz podpisu cyfrowego serwera, co przedstawiono na rysunku 2.

```
{
  "alg": "HS256",
  "typ": "JWT"
}
{
  "sub": "1234567890",
  "name": "John Doe",
  "iat": 1516239022
}
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  your-256-bit-secret
)
```

Rys. 1. Przykładowy obiekt JSON

Na przykładzie 9 przedstawiono fragment kodu generującego token po pomyślnym weryfikacji poświadczeń użytkownika.

Przykład 9. Generowanie tokena po weryfikacji tożsamości użytkownika

```
return Jwts.builder()
    .setSubject(authentication.getName())
    .claim(AUTHORITIES_KEY, authorities)
    .signWith(key, SignatureAlgorithm.HS512)
    .setExpiration(validity)
    .compact();
```

4.3. Uprawnienia użytkowników

Aplikacje internetowe udostępniają funkcje wielu grupom użytkowników. Użytkownicy nie powinni posiadać dostępu do funkcji oraz zasobów im nieprzeznaczonych. Aby uporządkować ten dostęp definiuje się role, które przydziela się użytkownikom. Najczęściej tworzony jest podział na administratorów oraz użytkowników. Administrator ma kontrolę nad funkcjami zarządzającymi systemem oraz nadaje lub ogranicza dostęp do zasobów systemowych innym użytkownikom [17]. Użytkownicy najczęściej posiadają uprawnienia niezbędne do korzystania z funkcji oferowanych przez aplikację.

Istnieje wiele narzędzi zawierających gotowe implementacje mechanizmu uprawnień. Jedną z nich jest ta oferowana przez framework Spring. Opiera się ona na wykorzystaniu adnotacji języka Java, które wskazują rolę użytkownika niezbędną do uruchomienia funkcji aplikacji.

Na przykładzie 10 przedstawiono użycie adnotacji *@PreAuthorize* nad metodą klasy kontrolera, która wykonuje funkcję przeznaczoną tylko dla administratora.

Przykład 10. Objęcie funkcji przeznaczonej tylko dla administratora adnotacją *@PreAuthorize*

```
@PostMapping("/branches")
@PreAuthorize(IS_ADMIN)
public BranchDTO create(BranchDTO branchDTO) {
    return super.create(branchDTO);
}
```

4.4. Konfiguracja nagłówków http

Klient aplikacji internetowej komunikuje się z serwerem poprzez protokół http. Działa on na podstawie architektury klient-serwer. Komunikat http składa się z nagłówków oraz ciała wiadomości. Odpowiedzi żądań http zawierają wiele informacji niezbędnych do prawidłowego zachowania przeglądarki internetowej [18]. Znajdują się one w nagłówkach. Domyślne zachowanie przeglądarki podatne jest na wiele ataków m.in. Cross Site Scripting (XSS), który polega dołączeniu do wyświetlanej treści strony internetowej złośliwego skryptu [19]. Na przykładzie 11 przedstawiono fragment kodu konfiguracji serwera, który ustawia nagłówek *X-XSS-Protection* dla każdej odpowiedzi.

Przykład 11. Konfiguracja serwera dla generacji nagłówków zabezpieczających

```
void configure(HttpSecurity http){
   http
     .headers()
          .xssProtection()

     .block(true)
     .apply();}
```

Odpowiedź serwera zawierająca nagłówek zabezpieczający przed atakiem XSS znajduje się na rys. 3.

```
HTTP/1.1 200 OK
Server: keycdn-engine
Date: Wed, 09 Mar 2016 20:06:01 GMT
Content-Type: text/html; charset=UTF-8
X-XSS-Protection: 1; mode=block
```

Rys. 2. Odpowiedź serwera wraz z nagłówkami

4.5. Uprawnienia użytkowników w bazie danych

Architektury aplikacji internetowych składają się z wielu warstw. Zabezpieczenie każdej z nich zmniejsza ryzyko niepowołanego dostępu do systemu z zewnątrz. Baza danych stanowi część systemu, do której dostęp powinien być najściślej zabezpieczony. Wyciek danych wiąże się poważną utratą zaufania użytkowników i zaburzeniem działania całego systemu.

Systemy bazodanowe oferują mechanizmy nadawania uprawnień użytkownikom [20]. Uprawnienia określają dostęp do zbioru działań możliwych do zrealizowania na schemacie bazy danych. Użytkownicy bazy danych przynależą do ról [21]. Każda z nich ma zdefiniowany zbiór przywilejów, które administrator może swobodnie przydzielać lub odbierać.

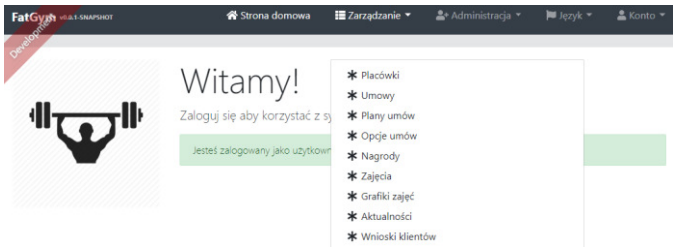
Przydział przywilejów do zdefiniowanej roli przedstawiony został na rysunku 4.

```
GRANT create table, create view
TO manager;
Grant succeeded.
```

Rys. 3. Przydział przywilejów do zdefiniowanej roli

5. Metoda badań

Aplikacja o tytule *FatGym*, wykorzystana do testowania oraz implementacji metod usprawniających jest systemem zarządzania międzynarodową siecią siłowni. Do jej implementacji wykorzystano język Java oraz frameworki Spring oraz Hibernate. Główne menu programu zostało przedstawione na rysunku 5.



Rys. 4. Menu główne aplikacji FatGym

Do przeprowadzania badań metod zwiększających wydajność postępowano według schematu, który obejmował:

- jednorazowe wypełnienie biorących udział w badaniu tabel bazy danych generowanymi danymi symulującymi globalny system obsługi klientów
- pomiar wydajności przed implementacją metody usprawniającej
- implementacja wybranej metody
- pomiar wydajności po implementacji metody usprawniającej
- powrót do stanu aplikacji sprzed implementacji metody usprawniającej.

Pomiar czasów odbywał się za pośrednictwem programu JMeter. Program generował i rozpoczynał wykonywanie 100 wątków równocześnie. Powtarza tą akcję co sekundę przez 10 sekund. W przypadku niedokończenia wykonywania wątków przed generacją nowych - ich liczba kumuluje się.

Badanie metod zwiększania bezpieczeństwa aplikacji obejmowało implementację każdej z metod oraz sprawdzanie poprawności jej działania.

Otrzymane wyniki analizowano w kontekście przydatności metody i zgodności z jej przeznaczeniem, a później zostały wyciągnięte wnioski. Efekt badań pozwolił zweryfikować postawione na początku prac tezy.

6. Wyniki

6.1. Metody zwiększające wydajność aplikacji

Wyniki badań związanych z metodami zwiększającymi wydajność aplikacji koncentrują się głównie wokół czasów wykonania zapytań bazodanowych oraz czasów odpowiedzi na żądania HTTP.

Podsumowanie czasów przed oraz po implementacji dla każdej metody zwiększającej wydajność przedstawione zostało w tabeli 1.

Różnice pomiędzy średnimi czasami odpowiedzi serwera przed i po implementacji metod zwiększających wydajność obrazuje rysunek 6.

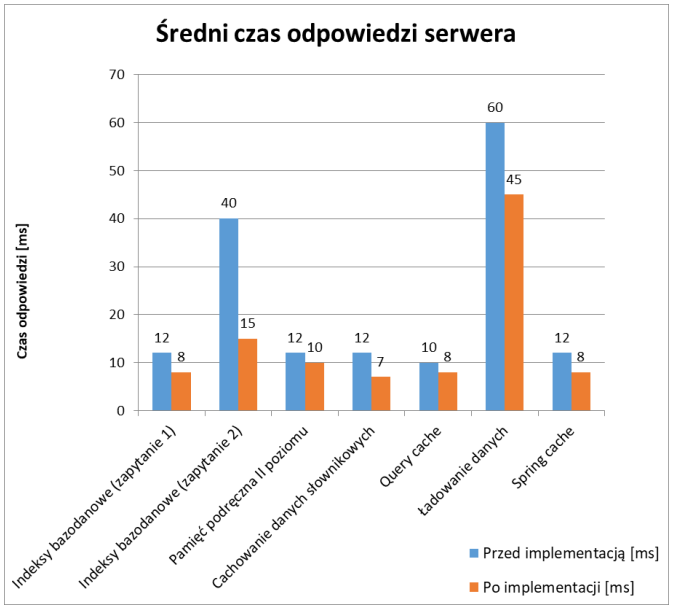
6.2. Metody zwiększające bezpieczeństwo aplikacji

Wyniki badań związanych z metodami zapewniającymi bezpieczeństwo aplikacji koncentrują się wokół sprawdzania poprawności ich działania.

Przeprowadzone testy rejestracji użytkownika wykazały poprawność działania funkcji hashującej BCrypt na podawanych przez użytkowników hasłach. Przetestowano również mechanizm logowania, w którym hash podanego hasła zostaje porównany ze wzorcem znajdującym się w bazie danych. Działanie funkcji przebiega zgodnie z oczekiwaniem.

Tabela 1. Czasy metod zwiększających wydajność aplikacji internetowej

Metoda zwiększająca wydajność	Najwyższy czas odpowiedzi przed implementacją metody [ms]	Średni czas odpowiedzi przed implementacją metody [ms]	Najwyższy czas odpowiedzi po implementacji metody [ms]	Średni czas odpowiedzi po implementacji metody [ms]	Różnica pomiędzy średnimi czasami po i przed implementacją [%]
Indeksy bazodanowe (indeks na kolumnie tekstowej)	55	12	15	8	33.33%
Indeksy bazodanowe (indeks na kluczu obcym)	48	40	22	15	62,50%
Pamięć podręczna II poziomu	19	12	14	10	16,67%
Cachowanie danych słownikowych	28	12	10	7	41,67%
Query cache	160	10	110	8	20.00%
Ładowanie danych	800	60	450	45	25,00%
Spring cache	270	12	110	8	33,33%
Paginacja danych	-	17000	-	9000	47,06%



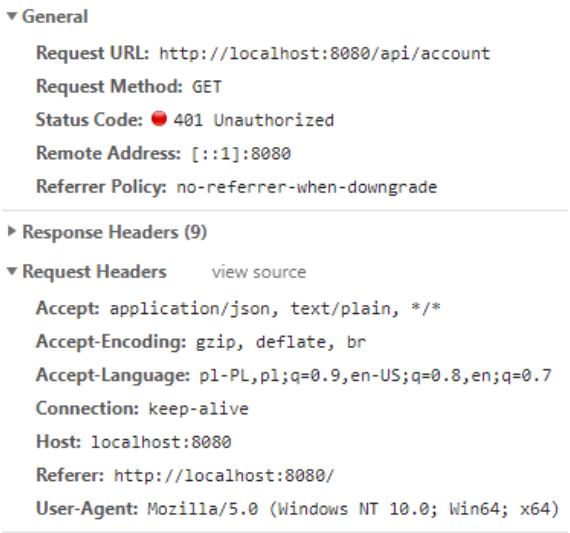
Rys. 5. Porównanie czasów odpowiedzi serwera przed i po implementacji danej metody wydajnościowej

Na rysunku 7 znajduje się fragment tabeli *Users,* która w poprawny sposób przechowuje hashe haseł.

id	login	password_hash
2	anonymoususer	$2a$10$j8S5d7Sr7.8VTOYNviDPOeWX8KcYILUVJBsYV83Y…
4	user	$2a$10$VEjxo0jq2YG9Rbk2HmX9S.k1uZBGYUHdUcid3g/v…
1101	jkowalski	$2a$10$P2ZqM8cad35yfYkWpmfjVu4bXO.ggBduPTKUVU3w…
3	admin	$2a$10$gSAhZrxMllrbgj/kkK9UceBPpChGWJA7SYIb1Mqo…
1	system	$2a$10$mE.qmcV0mFU5NcKh73TZx.z4ueI/.bDWbj0T1BYy…
1401	mpudzianowski	$2a$10$CoNk3VBeump2GUpGs7l0ue7uwsD4XX9DcP8oWzWx…

Rys. 6. Fragment tabeli Users, w której przechowywane są hashe haseł

Testy działania mechanizmu JSON Web Token wykazały odporność aplikacji na nieautoryzowany do niej dostęp. Wysłanie żądania bez tokena kończy się odmową dostępu do systemu, , co prezentuje rysunek 8.



Rys. 7. Odpowiedź na żądanie niezawierające tokena

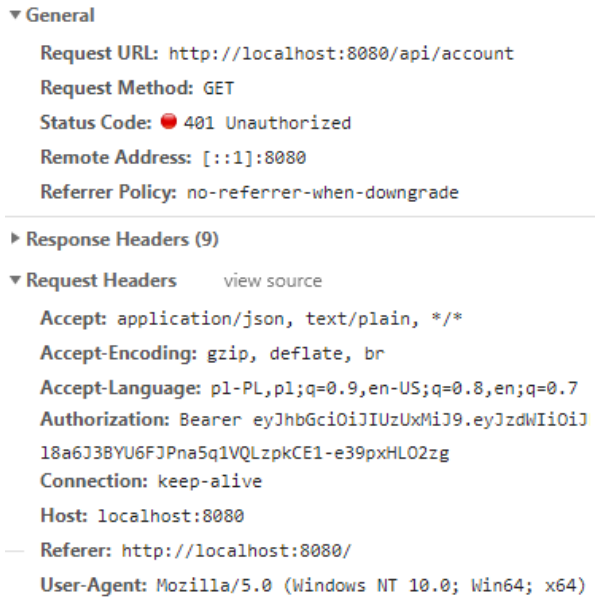
Testy aplikacji po implementacji uprawnień wykazały poprawne działanie blokady dostępu użytkowników do funkcji przeznaczonych tylko dla administratorów. Próba otwarcia przez użytkownika strony dostępnej tylko dla administratora kończy się odmową dostępu. Odpowiedź na żądanie prezentuje rysunek 9.

Serwer zwraca odpowiedzi z nagłówkami X-XSS-Protection oraz Strict-Transport-Security. To potwierdza zabezpieczenie aplikacji przed wstrzykiwaniem złośliwego kodu oraz wymusza komunikację z użyciem protokołu https.

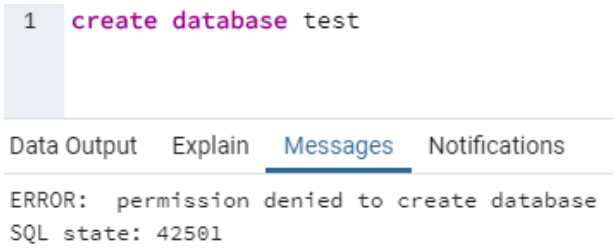
Baza danych skutecznie blokuje dostęp aplikacji do funkcji ingerujących w schemat bazy danych. Próba utworzenia nowej bazy danych kończy się odmową dostępu, co prezentuje rysunek 10.

7. Wnioski

Aplikacja *FatGym* okazała się odpowiednią do implementacji metod zwiększających wydajność i bezpieczeństwo. Implementacja wszystkich metod przebiegła pomyślnie.



Rys. 8. Odpowiedź na próbę otwarcia strony dostępnej tylko dla administratora przez użytkownika



Rys. 9. Odmowa dostępu do akcji niedostępnych dla zwykłego użytkownika

Na podstawie badań przeprowadzonych można sformułować następujące wnioski:

- wraz ze wzrostem liczby użytkowników systemu rośnie potrzeba zapewnienia wysokiej wydajności systemu
- metody zwiększające wydajność, będące integralną częścią frameworków Spring i Hibernate oferują zauważalną poprawę wydajności aplikacji (uzyskano poprawę wydajności średnio o 25%)
- stosowanie metod poprawy wydajności na wielu poziomach (back-end i baza danych), potęguje wzrost szybkości działania aplikacji (uzyskano poprawę wydajności średnio o 30%)
- metody poprawy wydajności powinny być implementowane zgodnie z ich przeznaczeniem;
- użycie metody poprawy wydajności niezgodnie z jej przypadkiem użycia może nawet spowolnić działanie aplikacji
- przedstawione w pracy metody zapewniania bezpieczeństwa aplikacji internetowej powinny stanowić podstawę jej działania
- prezentowane metody dotyczące bezpieczeństwa skutecznie zabezpieczają aplikację przed większością zagrożeń

- aplikacje internetowe powinny być zabezpieczane na wszystkich poziomach: od front-endu, przez back-end do bazy danych.

Podsumowując wszystkie badania i wnioski, potwierdzone zostają następujące tezy: „Metody wydajnościowe frameworków Hibernate i Spring Boot w bardzo znaczący sposób poprawiają szybkość działania aplikacji internetowych" oraz „Metody zwiększające bezpieczeństwo są skutecznym sposobem ochrony aplikacji internetowych". Do metod zwiększających bezpieczeństwo należą: haszowanie haseł w bazie danych, użycie JSON Web Token, ustanowienie uprawnień użytkowników, konfiguracja nagłówków http oraz ustanowienie uprawnień użytkowników w bazie danych. W zależności od użytego mechanizmu udało się skrócić czasy wykonywania żądań od 15% do 60%.

Literatura

[1] Qinglin Wu, Yan Wang; Performance Testing and Optimization of J2EE-based Web Applications; 2010 Second International Workshop on Education Technology and Computer Science; 2010.

[2] Abdulrahman Alzahrani, Ali Alqazzaz, Ye Zhu, Huirong Fu, Nabil Almashfi; Web Application Security Tools Analysis; 2017 ieee 3rd international conference on big data security on cloud (bigdatasecurity), ieee international conference on high performance and smart computing (hpsc), and ieee international conference on intelligent data and security (ids); 2017.

[3] Mahesh Bang, Himanshu Saraswat; Building an effective and efficient continuous web application security program; 2016 International Conference On Cyber Situational Awareness, Data Analytics And Assessment (CyberSA), 2016.

[4] Kirti Gupta, Manish Mathuria; Improving Performance of Web Application approaches using Connection Pooling; 2017 International conference of Electronics, Communication and Aerospace Technology (ICECA); 2017.

[5] Smith G., Wysoko wydajny PostgreSQL, Helion, 2011.

[6] Bauer C., Java Persistence: programowanie aplikacji bazodanowych w Hibernate, Helion, 2017.

[7] Hibernate Second-Level Cache | Baeldung, https://www.baeldung.com/hibernate-second-level-cache, [15.05.2019].

[8] Hibernate Cache Strategy Work, https://vladmihalcea.com/how-does-hibernate-read_only-cacheconcurrencystrategy-work/, [12.05.2019].

[9] Hibernate. Caching, https://docs.jboss.org/hibernate/orm/4.0/devguide/en-US/html/ch06.html, [13.05.2019].

[10] Eager/Lazy Loading in Hibernate | Baeldung, https://www.baeldung.com/hibernate-lazy-eager-loading, [14.05.2019].

[11] Mak G., Rubio D., Long J., Spring. Receptury, Helion, 2014

[12] Spring. Caching, https://docs.spring.io/spring-boot/docs/current/reference/html/boot-features-caching.html, [14.05.2019].

[13] Spring. Paging and Sorting, https://docs.spring.io/spring-data/rest/docs/2.0.0.M1/reference/html/paging-chapter.html, [15.05.2019].

[14] Lis M., Tworzenie bezpiecznych aplikacji internetowych, Helion, 2014.

[15] JSON Web Token Introduction, https://jwt.io/introduction/, [15.05.2019].

[16] Barnett R., Web Application Defender's Cookbook: Battling Hackers and Protecting Users, Wiley, 2012.

[17] Introducing to Spring Method Security | Baeldung, https://www.baeldung.com/spring-security-method-security, [15.05.2019].

[18] The 8 HTTP Security Headers Best Practices, https://www.globaldots.com/8-http-security-headers-best-practices/, [15.05.2019]

[19] Hope P., Walther B., Testowanie bezpieczeństwa aplikacji internetowych. Receptury, Helion, 2017

[20] Elsmari R., Navathe S., Wprowadzenie do systemów baz danych, Helion, 2019.

[21] Understanding Users, Privileges, and Roles, https://www.vertica.com/blog/understanding-users-privileges-roles/, [14.05.2019].

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 12 (2019) 214-219
Wysłane: 2019-05-25
Przyjęte: 2019-06-18

Porównanie wydajności rozwiązań wirtualizacyjnych na przykładzie Proxmox, OpenVZ, OpenNebula, Vmware ESX i Xen Server

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

Streszczenie. Celem artykułu jest porównanie wydajności pięciu popularnych rozwiązań wirtualizacyjnych stosowanych w modelu chmury prywatnej. Analiza została przeprowadzona dla pięciu wirtualizatorów: Proxmox, OpenVZ, OpenNebula, Vmware ESX, Xen Server. Do badań wykorzystano narzędzia OpenSSL, GeekBench i Phoronix-test-suite w celu wykonania testów wydajnościowych. w porównaniu uwzględniono wybrane metody szyfrowania, wielkość bloku danych oraz prędkość kompilacji.

Słowa kluczowe: Wirtualizacja; Hipernadzorca; Chmura prywatna

Autor do korespondencji.

Adres e-mail: Grzegorz.Rycaj@pollub.edu.pl

Comparison of virtualization performance of Proxmox, OpenVZ, OpenNebula, Vmware ESX and Xen Server

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

Abstract. The aim of the article is to compare the performance of five popular virtualization solutions used in the private cloud model. The analysis was conducted for five virtualizers: Proxmox, OpenVZ, OpenNebula, Vmware ESX, Xen Server. OpenSSL, GeekBench and Phoronix-test-suite tools were used to perform performance tests. The comparison included selected encryption methods, data block size, and compilation speed.

Keywords: Virtualization; Hypervisor; Private Cloud

Corresponding author.

E-mail address: Grzegorz.Rycaj@pollub.edu.pl

1. Wstęp

Artykuł ma na celu zebranie i szczegółowe omówienie wybranych rozwiązań wirtualizacyjnych. W obecnych czasach wirtualizacja jest powszechnie stosowana, przy budowie systemów informatycznych, umożliwiając większą niezależność, od zastosowanej architektury sprzętowej, co przekłada się na wyższą dostępność środowiska systemu informatycznego.

Za cel badań autor postawił sobie przeprowadzenie analizy porównawczej rozwiązań wirtualizacyjnych pod kątem wydajnościowym, środowisk wykorzystywanych w modelu chmury prywatnej. Analiza zawierała następujące rozwiązania wirtualizacyjne: Proxmox, OpenVZ, OpenNebula, Vmware ESX, Xen Server.

Testy wydajnościowe przygotowanych platform, przeprowadzono za pomocą narzędzi (OpenSSL, Geekbench4, Phoronix-test-suite). Otrzymane wyniki, pozwoliły na szczegółowe omówienie zależności rozwiązań wirtualizacyjnych, wraz z porównaniem ich wydajności w danym środowisku wirtualizacyjnym. Wyniki zaprezentowano w sposób tabelaryczny i graficzny.

Chmura obliczeniowa (ang. cloud computing) jest to przetwarzanie w chmurze, które polega na dostarczeniu klientowi usług obliczeniowych takich jak (serwery, magazyn danych, oprogramowanie bazodanowe, sieć, narzędzia programistyczne) za pośrednictwem Internetu [1].

Chmurę obliczeniową można podzielić na:

- Chmura Publiczna (ang. public cloud computing) polega na udostępnieniu zasobów za pośrednictwem Internetu, całą infrastrukturą sprzętową, oprogramowaniem zarządza dostawca usługi.
- Chmura Prywatna (ang. private cloud computing) polega na udostępnieniu zasobów za pośrednictwem Internetu w obrębie jednej organizacji, wszystkie dane i usługi są udostępniane w ramach jednej organizacji (fizycznie ze względów zapewnienia bezpieczeństwa zazwyczaj maszyny serwerowe są rozlokowane w kilku miejscach) [4].
- Chmura Hybrydowa (ang. hybrid cloud computing) polega na połączeniu infrastruktury chmury prywatnej z chmurą publiczną, z wykorzystaniem udostępnienia danych i aplikacji. Zapewnia to większą elastyczność i optymalizację aktualnie posiadanej infrastruktury [5].

2. Przegląd wybranych rozwiązań

- PVE (ang. proxmox virtual environment) – środowisko wirtualizacyjne zaimplementowane w języku Perl, należy do darmowych rozwiązań wirtualizacyjnych o otwartym kodzie źródłowym dostępnym dla społeczności, oferuje szeroki zakres funkcjonalności, który nie ustępuje pod żadnym względem rozwiązaniom komercyjnym. Proxmox pracuje na bazie dystrybucji systemu operacyjnego Linux opartego na dystrybucji Debian, która umożliwia wszechstronną wirtualizację przedsiębiorstwa, integrację KVM, kontenerów LCX (OpenVZ), systemów pamięci masowej (ceph) czy tworzeniem dodatkowych wirtualnych sieci lan. Proxmox w łatwy sposób umożliwia zarzadzanie klastrami, jak również narzędziami do szybkiego przywrócenia infrastruktury po awarii. Oprogramowanie zawiera bardzo przejrzysty i intuicyjny interfejs graficzny dostępny w przeglądarce internetowej, istnieje także możliwość konfiguracji za pomocą konsoli, oraz udostępnia API dla integracji z innymi narzędziami. Oprogramowanie w przypadku pamięci masowych obsługuje zaawansowane rozwiązania, między innymi: Pamięci z ISCSI, Fibre Channel, NFS, CEPH, GlusteFS. Dużą zaletą platformy jest możliwość migracji maszyn wirtualnych online bez przestojów w pracy środowiska, przykładowo systemu produkcyjnego (w momencie przełączenia między maszynami wirtualnymi może zostać utraconych kilka pakietów danych) [6, [7].
- OpenVZ – platforma wirtualizacyjna systemu operacyjnego Linux, która umożliwia uruchomienie kilku maszyn wirtualnych (ang. virtual private serwer), wszystkie systemy wirtualne korzystają z tego samego jądra systemowego, co hipernadzorca. Każda z maszyn wirtualnych zarządzana jest osobno, posiadają w pełni niezależne zasoby fizyczne lub wirtualne między innymi pamięć operacyjna, system plików, zarządzanie siecią, konta użytkowników. Migracja maszyn wirtualnych między węzłami fizycznymi pracującymi pod kontrolą OpenVZ, jest możliwa w sposób online z zachowaniem wysokiego SLA dostępności infrastruktury, podobnie jest w przypadku rozbudowy zasobów na danej maszynie wirtualnej [8].
- OpenNebula – jest to platforma wirtualizacyjna zaimplementowana w językach Java, C++, C, Yacc, Ruby. Platforma OpenNebula należy do darmowych rozwiązań wirtualizacyjnych o otwartym kodzie źródłowym, oferuje bardzo szeroki zakres funkcjonalności, do zarządzania heterogenicznymi infrastrukturami rozproszonych centrów danych. Oprogramowanie w pełni zarządza całą infrastrukturą i umożliwia tworzenie prywatnych, publicznych czy także hybrydowych rozwiązań wirtualizacyjnych jako usługi np. model IaaS. OpenNebula pracuje na bazie dystrybucji systemu operacyjnego Linux oraz umożliwia scentralizowanie zarządzania w jednym miejscu, za sprawą wielu technologii do przechowywania danych, wirtualizacji sieci czy monitorowania, jak i bezpieczeństwa systemów. OpenNebula zawiera zestaw narzędzi, które pozwalają na integrację z takmi produktami jak: Amazon EC2, vClocud, OpenCloud oraz hipernadzorcami (KVM, XEN, Vmware, OpenStack, Openshift). Oprogramowanie zawiera bardzo przejrzysty i intuicyjny interfejs graficzny realizowany za pomocą przeglądarki internetowej, wiersza poleceń oraz interfejsami API np: AWS EC2, OGF, OCCI [9, 10].
- Vmware ESX – jest to zaawansowana platforma wirtualizacyjna stosowana do budowy wysoce dostępnych środowisk wirtualizacyjnych, oparta jest o hipernadzorcę typu pierwszego. Vmware ESX jest liderem na rynku wirtualizacji serwerów, istnieje wiele narzędzi do integracji produktów z usługami Vmware ESX, między innymi oprogramowanie do zarządzania i tworzenia kopii zapasowych np. Netapp, Veeam. Kolejną ogromną zaletą przemawiającą za wyborem produktów Vmware jest zaawansowana łączność sieciowa, pomiędzy maszynami wirtualnymi, możliwość integracji wirtualnych przełączników sieciowych (ang. vSwitch), które są w pełni kompatybilne z infrastrukturą sieciową. Umożliwiają zarządzanie, tworzenie profili czy provisioning. Odbywa się to za pomocą wirtualnego switcha Nexus 1000v, który można podzielić na dwa moduły sieciowe, nadzorujący (VSM) i modułu ethernetowego (VEM) na każdym hoście ESX. Maksymalna liczba hostów nadzorujących ESX wynosi 64 [11, 12].
- Xen Center – jest to platforma wirtualizacyjna, która korzysta z rozwiązań wirtualizacji sprzętowej oraz parawirtualizacji, pracuje na poziomie hipernadzorcy typu pierwszego. Umożliwia uruchomienie wielu maszyn wirtualnych wykorzystujących różnego typu systemy operacyjne między innymi, dystrybucje Linuksa, jak i systemy operacyjne Windows. Platforma Xen umożliwia pracę na dwóch poziomach, między innymi przy pomocy parawirtualizacji, gdzie wymagana jest modyfikacja jądra systemowego gościa, jak i sterowników. Wówczas to systemy gości odwołują się bezpośrednio do hipernadzorcy i nie jest już wymagana emulacja sprzętu. Kolejnym typem pracy platformy wirtualizacyjnej Xen jest wykorzystanie wirtualizacji sprzętowej, w tym przypadku procesor musi posiadać sprzętowe wsparcie wirtualizacji (AMD - V lub Intel – VT), tego typu rozwiązanie nie wymaga modyfikacji jądra systemów w porównaniu do parawirtualizacji [13].

3. Opis badań

Rozwiązania wirtualizacyjne zostały wybrane na podstawie ogólnie dostepnych funkcjonalności danego oprogramowania, które są wykorzystywane w modelu chmury prywatnej. Dobór konkretnych rozwiązań miał umożliwić szybkie uruchomienie produkcyjne wraz z pracującą maszyną wirtualną, bez wiekszego nakładu pracy i wykonanie testów wydajnościowych na każdym z użytych wirtualizatorów.

3.1. Wykorzystane narzędzia

OpenSSL jest to narzędzie, które zawiera implementację protokołów SSL i TLS o otwartym kodzie źródłowym, najnowsze wydanie OpenSSL to 1.1.1. wspierane do 2023 roku. Narzędzie OpenSSL obsługuje wiele różnych algorytmów kryptograficznych, między innymi:

- Alforytmy szyfrowania: AES, Blowfish, Camellia, DES, IDEA, RC5, Triple DES, SEED,

- Funkcje kryptograficzne: MD5, MD4, MD2, SHA-1, SHA-2, SHA-3, SM3, BLAKE2,
- Kryptografia klucza publicznego: RSA, DSA, X25519, X448, Ed448, GOST R, SM2 [15].

Główne założenia przyjęte do testów narzędziem OpenSSL:

- Do przeprowadzenia testów wykorzystano najnowszą wersję oprogramowania (1.1.1).
- Polecenie openssl speed | tee test.txt (program tee jest odpowiedzialny za zapisanie otrzymanych wyników w dokumencie formatu txt).
- Testy powtórzono 3 krotnie.

GeekBench4 jest to uniwersalne i zawierające kompletny zestaw testów wydajnościowych narzędzie. Zawiera testy wydajnościowe między innymi (procesora, pamięci, I/O czy testy wydajnościowe bazy danych).

Phoronix Test Suite jest to kompleksowe narzędzie, które zawiera wiele narzędzi testujących system Linux ze skryptami PHP.

Uwzględnione testy dla narzędzia Phoronix Test Suite:

- John The Ripper (narzędzie służące do łamania haseł) polecenie: phoronix-test-suite run john-the-ripper
- Kompilacja aplikacji polecenie: phoronix-test-suite run pts/build-mplayer-1.3.0
- Test kompresji, polecenie: phoronix-test-suite run pts/compress-gzip-1.1.0

3.2. Opis środowiska testowego

Do przeprowadzonych badań wykorzystano dedykowany serwer z pełną wirtualizacją KVM, która umożliwiła przeprowadzenie zamierzonych badań wydajnościowych poszczególnych środowisk wirtualizacyjnych. Wszystkie testy wydajnościowe zostały przeprowadzone na bazie systemu operacyjnego Linux. Przestrzeń dysku została podzielona na sześć oddzielnych partycji przestrzeni dyskowej ze względu na konieczność odseparowania poszczególnych technologii wirtualizacji. Na każdym z badanych środowisk zostały utworzone dwie maszyny wirtualne o identycznych parametrach sprzętowych (1 vCpu, 1 GB RAM, 20 GB przestrzeni dyskowej). Szczegółowe parametry maszyn wirtualnych.

Tabela 1. Parametry maszyny wirtualnej.

PARAMETRY MASZYN WIRTUALNYCH	
Liczba vCPU	1
RAM	2 GB
Dysk twardy	20 GB
Przestrzeń wymiany	1 GB
Sieć	w trybie (Bridged)
Grafika	bez akceleracji 3D
Wirtualny OS	Ubuntu 18.04 LTS

3.3. Kryteria wykonanych testów

- Analiza wydajności algorytmów szyfrowania - z otrzymanych wyników wyszczególniono najgorszy wynik dla danego algorytmu. Wynik ten później posłużył jako wartość liczbowa do porównania z innymi rozwiązaniami wirtualizacyjnymi.
- Analiza wyników wydajności szyfrowania z użyciem różnych bloków danych - z otrzymanych wyników wyszczególniono najgorszy wynik w danym teście blokowym, który później posłużył jako wartość liczbowa do porównania z innymi rozwiązaniami wirtualizacyjnymi.
- Analiza z testów przeprowadzonych narzędziem Geekbench4 i Phoronix-test-suite takich jak wydajność procesora z uwzględnieniem testów jednordzeniowych, jak i również wielordzeniowych, kompresji, kompilacji programu, łamania haseł - z otrzymanych wyników wyszczególniono najgorszy wynik w danej sekcji, który później posłużył jako wartość liczbowa do porównania z innymi rozwiązaniami wirtualizacyjnymi.
- Analiza pod kątem sumy uzyskanych wyników w testach wydajnościowych poszczególnych technologii wirtualizacyjnych (maszyn wirtualnych). Utworzono również ranking najbardziej wydajnych rozwiązań wirtualizacyjnych stosowanych w modelu chmury prywatnej.

4. Wyniki Badań

4.1. Wyniki badań za pomocą narzędzia OpenSSL

Tabela 2 Analiza porównawcza wydajności szyfrowania dla poszczególnych bloków danych wykonana narzędziem OpenSSL

Wirtualizator	16 bajtów	64 bajtów	256 bajtów	1024 bajtów	8192 bajtów	16384 bajtów
Proxmox	195978	298775	413305	526780	568602	572459
OpenNebula	221836	353300	466396	562891	627958	642970
OpenVZ	196937	298869	414187	528063	569286	573677
XEN	193192	293982	412065	523523	566639	570331
Vmware ESX	224846	357033	373401	579054	518981	626233

Tabela 2 przedstawia wyniki testów wydajnościowych szyfrowania różnych bloków danych. Można stwierdzić, że rozwiązania stosowane do wirtualizacji w modelu chmury prywatnej przedstawiały się następująco:

- Wydajność szyfrowania dla bloku danych 16 bajtów pierwsze miejsce Vmware ESX z wynikiem lepszym o 16,39 % w porównaniu do maszyny wirtualnej o najgorszej wydajności. Na drugim miejscu uplasowała się maszyna wirtualna uruchomiona pod kontrolą hipernadzorcy OpenNebula z wynikiem lepszym o 14,83 %, na trzecim uplasowało się rozwiązanie OpenVZ z wynikiem lepszym o 1,94%. Czwarte miejsce zajmuje wirtualizator Proxmox z wynikiem lepszym o 1,44 %, na ostatnim miejscu znalazło się rozwiązanie firmy Xen z wynikiem 193192 Kb/s (wartość bazowa).
- Wydajność szyfrowania dla bloku danych 64 bajtów: pierwsze miejsce Vmware ESX z wynikiem lepszym o 21,45 % w porównaniu do maszyny wirtualnej o najgorszej wydajności. Na drugim miejscu uplasowała się maszyna wirtualna uruchomiona pod kontrolą hipernadzorcy OpenNebula z wynikiem lepszym o 20,18 %, na trzecim uplasowało się rozwiązanie OpenVZ z wynikiem lepszym o 1,66 %. Czwarte miejsce zajmuje wirtualizator Proxmox z wynikiem lepszym o 1,63 %, na

ostatnim miejscu znalazło się rozwiązanie firmy Xen z wynikiem 293982 Kb/s (wartość bazowa).

- Wydajność szyfrowania dla bloku danych 256 bajtów: pierwsze miejsce OpenNebula z wynikiem lepszym o 24,90 % w porównaniu do maszyny wirtualnej o najgorszej wydajności. Na drugim miejscu uplasowała się maszyna wirtualna uruchomiona pod kontrolą hipernadzorcy OpenVZ z wynikiem lepszym o 10,92 %, na trzecim uplasowało się rozwiązanie Proxmox z wynikiem lepszym 10,69%. Czwarte miejsce zajmuje wirtualizator Xen z wynikiem lepszym o 10,35 %, na ostatnim miejscu znalazło się rozwiązanie firmy Vmware ESX z wynikiem 373401 Kb/s (wartość bazowa).
- Wydajność szyfrowania dla bloku danych 1024 bajtów: pierwsze miejsce Vmware ESX z wynikiem lepszym o 10,61 % w porównaniu do maszyny wirtualnej o najgorszej wydajności. Na drugim miejscu uplasowała się maszyna wirtualna uruchomiona pod kontrolą hipernadzorcy OpenNebula z wynikiem lepszym o 7,52 %, na trzecim uplasowało się rozwiązanie OpenVZ z wynikiem lepszym o 0,87%. Czwarte miejsce zajmuje wirtualizator Proxmox z wynikiem lepszym o 0,62 %, na ostatnim miejscu znalazło się rozwiązanie firmy Xen z wynikiem 523523 Kb/s (wartość bazowa).
- Wydajność szyfrowania dla bloku danych 8192 bajtów: pierwsze miejsce OpenNebula z wynikiem lepszym o 21 % w porównaniu do maszyny wirtualnej o najgorszej wydajności. Na drugim miejscu uplasowała się maszyna wirtualna uruchomiona pod kontrolą hipernadzorcy OpenVZ z wynikiem lepszym o 9,69 %, na trzecim uplasowało się rozwiązanie Proxmox z wynikiem lepszym o 9,56 %. Czwarte miejsce zajmuje wirtualizator Xen z wynikiem lepszym o 9,18 %, na ostatnim miejscu znalazło się rozwiązanie firmy Vmware ESX z wynikiem 518981 Kb/s (wartość bazowa).
- Wydajność szyfrowania dla bloku danych 16384 bajtów: pierwsze miejsce OpenNebula z wynikiem lepszym o 12,74 % w porównaniu do maszyny wirtualnej o najgorszej wydajności. Na drugim miejscu uplasowała się maszyna wirtualna uruchomiona pod kontrolą hipernadzorcy Vmware ESX z wynikiem lepszym o 9,80 %, na trzecim uplasowało się rozwiązanie OpenVZ z wynikiem lepszym o 0,59 %. Czwarte miejsce zajmuje wirtualizator Proxmox z wynikiem lepszym o 0,37 %, na ostatnim miejscu znalazło się rozwiązanie firmy Xen z wynikiem 570331 Kb/s (wartość bazowa).

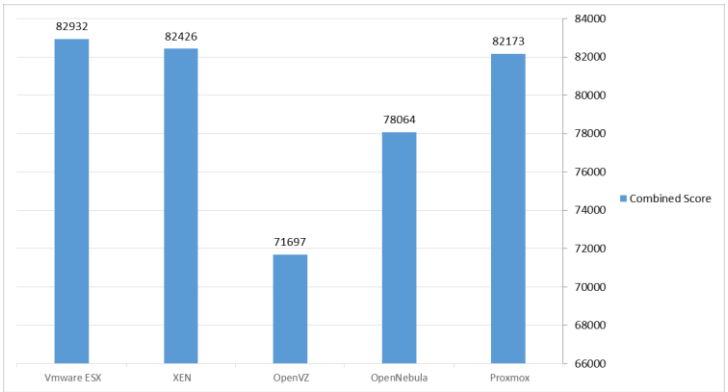
4.2. Wyniki badań otrzymane za pomocą narzędzia Geekbench4

Combined Score w przypadku testów narzędziem GeekBench zawiera testy takie jak: aes, lzma, jpeg, canny, lua, Dijkstra, sqlite, html5 parse, html5 dom, pdf rendering, llvm, memory copy, memory latency, memory Bandwidth wyrażonych w Kb/s (Rys. 1).

Analizując wyniki testów wydajnościowych uzyskanych za pomocą narzędzia GeekBench (tabela 3 i rys. 1, pierwsze miejsce zajmuje rozwiązanie Vmware ESX z wynikiem lepszym o 15,67 % w porównaniu do maszyny wirtualnej o najgorszej wydajności. Na drugim miejscu uplasowała się maszyna wirtualna uruchomiona pod kontrolą hipernadzorcy Xen z wynikiem lepszym o 14,96 %, na trzecim uplasowało się rozwiązanie Proxmox z wynikiem lepszym o 14,61 %. Czwarte miejsce zajmuje wirtualizator OpenNebula z wynikiem lepszym o 8,88 %, na ostatnim miejscu znalazło się rozwiązanie firmy OpenVZ (wartość bazowa).

Tabela 3. Analiza poszczególnych benchmarków wykonana narzędziem GeekBench

Rodzaj	Proxmox	OpenNebula	OpenVZ	Xen	VmwareESX
AES (MB/s)	285	333	284	286	290
LZMA (MB/s)	5013	4877	4982	5035	5034
JPEG (Mpx/s)	5664	5655	5684	5724	5768
CANNY (Mpx/s)	5516	5648	4676	5513	5565
LUA (KB/s)	4157	4305	3187	4039	4031
Dijkstra (MTE/s)	6360	6465	4640	6453	6530
SQLite (Kr/s)	4989	4947	3705	4951	5025
HTML5 Parse (MB/s)	5300	5236	3969	5247	5265
HTML5 DOM (MB/s)	5299	5409	3908	5452	5221
Histogram Equalization (Mpx/s)	4909	4808	3833	4917	4806
PDF Rendering (Mpx/s)	5923	5923	5639	5949	6006
LLVM (func/s)	9142	8835	8228	9187	9231
Memory Copy (GB/s)	6533	6054	6105	6561	6655
Memory Latency (ns)	8195	4957	8040	8223	8332
Memory Bandwidth (GB/s)	4888	4612	4817	4889	5173



Rys. 1. Sumaryczne wyniki wartości testów wykonana narzędziem GeekBench w porównaniu do wirtualizatora o najgorszym wyniku.

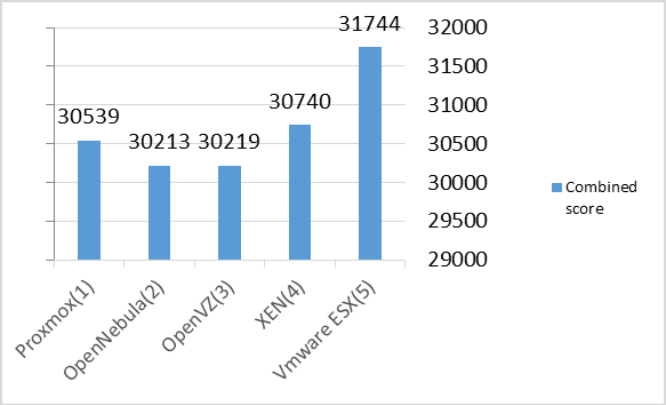
4.3. Wyniki badań za pomocą narzędzia Phoronix Test Suite

Combined Score w przypadku testów narzędziem Phoronix test suite zawiera sumaryczne wartości z przeprowadzonych testów takich jak blowfish, traditional des oraz md5 wyrażonych w Kb/s.

Analizując wyniki testów wydajnościowych za pomocą narzędzia Phoronix-test-sutie (tabela 4 i rys. 3) pod względem przeprowadzonych testów można stwierdzić, że w przypadku rozwiązań stosowanych do wirtualizacji w modelu chmury prywatnej. Pierwsze miejsce zajmuje rozwiązanie Vmware ESX z wynikiem lepszym o 5.07 % w porównaniu do maszyny wirtualnej o najgorszej wydajności. Na drugim miejscu uplasowała się maszyna wirtualna uruchomiona pod kontrolą hipernadzorcy Xen z wynikiem 1.75 %, na trzecim uplasowało się rozwiązanie Proxmox z wynikiem 1.08 %. Czwarte miejsce zajmuje wirtualizator OpenVZ z wynikiem 0.02 %, na ostatnim miejscu znalazło się rozwiązanie firmy OpenNebula z wynikiem 30213 Kb/s (wartość bazowa).

Tabela 4. Analiza dla przeprowadzonych testów wydajnościowych za pomocą narzędzia Phoronix-test-suite.

Rodzaj testu		Proxmox	OpenNebula	OpenVZ	XEN	Vmware ESX
John-the-ripper	Blowfish (Kb/s)	822	794	804	809	931
	Traditional DES (Kb/s)	30512131	30184531	30191475	30712262	31712397
	MD5 (Kb/s)	26417	27412	26731	27049	30259
Compress (s)		22,32	25.86	24,86	20,27	19,12
Compile (s)		9,76	11,12	9,83	9,64	9,52



Rys 2. Sumaryczne wyniki testów wykonanych narzędziem Phoronix w porównaniu do wirtualizatora o najgorszym wyniku

Tabela 5. Wynik końcowy przedstawiony jako ranking wydajnościowy testowanych rozwiązań wirtualizacyjnych.

Maszyna wirtualna	Wynik końcowy dla testu OpenSSL	Wynik końcowy dla testu GeekBench	Wynik końcowy dla testu Phoronix	Wynik sumaryczny	Ranking
OpenNebula	28753518280	78064	30212748	28783809092	1
Vmware ESX	26795470530	82932	31743616	26827297078	2
OpenVZ	25810185110	71697	30219045	25840475852	3
Proxmox	25758989640	82173	3078484	25762150297	4
XEN	25597314560	82426	30740150	25628137136	5

Podsumowując, wyniki sumaryczne Tabeli 5 z przeprowadzonych testów, można stwierdzić iż w przypadku rozwiązań w modelu chmury prywatnej najbardziej wydajnym środowiskiem wirtualizacyjnym okazało się rozwiązanie OpenNebula z wynikiem lepszym aż o 12,31 % w porównaniu do maszyny wirtualnej o najgorszej wydajności (Xen). Na drugim miejscu uplasowała się maszyna pod kontrolą Vmware ESX z wynikiem lepszym o 4,68 %, na trzecim miejscu uplasowało się rozwiązanie OpenVZ z wynikiem lepszym o 0,83 %, na czwartym miejscu rozwiązanie Proxmox z wynikiem lepszym o 0,52 %. Najmniej wydajnym środowiskiem okazało się rozwiązanie firmy Xen z wynikiem bazowym 25628137136 Kb/s.

Porównując uzyskane wyniki z przeprowadzonych testów z badaniami, które zostały wykonane w 2017 roku przez zespół badawczy Politechniki Rzeszowskiej [14]. W przypadku rozwiązań Vmware Esx i Xen potwierdza się teza, że rozwiązanie Vmware klasyfikuję się zdecydowanie lepszymi wynikami niż rozwiązanie Xen.

5. Wnioski

Wykonano testy i przeanalizowano wyniki łączenie 5 rozwiązań wirtualizacyjnych obecnie stosowanych do budowania wysoce dostępnych i wydajnych chmur obliczeniowych z wykorzystaniem nowych technologii. Z przeprowadzonych testów wydajnościowych przy użyciu narzędzi (OpenSSL, GeekBench oraz Phoronix-test-suite) wykazano, że najwydajniejszym rozwiązaniem wirtualizacyjnym w przypadku testowanych rozwiązań, opartych o model chmury prywatnej w testach zwyciężyło rozwiązanie OpenNebula, lecz tylko i wyłącznie ze względu na metodę testową OpenSSL, gdyż tutaj znacząco prowadził w testach, co miało duży wpływ na wynik. Rozwiązanie Vmware ESX znacząco przewyższa w testach inne rozwiązania wirtualizacyjne, jak również rozwiązanie OpenNebula, zapewne ze względu na wieloletnie doświadczenie komercyjne, szeroki zakres zastosowania, jak i również nieustanny rozwój.

Literatura

[1] Dziennik internautów, Chmura publiczna, prywatna i hybrydowa, http://di.com.pl/chmura-publiczna-prywatna-i-hybrydowa-czym-sie-roznia-46699 [21.04.2019].

[2] Gerald J.Popek, Robert P. Goldberg : Formal Requirements for Virtualizable Third Generation Architectures, http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.141.4815&rep=rep1&type=pdf [21.04.2019].

[3] N. Naik, Defence School of Communications and Information Systems Ministry of Defence, Building A Virtual System of Systems Using Docker Swarm in Multiple Clouds, [21.04.2019].

[4] Searchservervirtualization: Type 1 vs Type 2 virtualization hypervisior, https://searchservervirtualization.techtarget.com/feature/Whats-the-difference-between-Type-1-and-Type-2-hypervisors [22.04.2019].

[5] Wikipedia: Protection Ring (computer security), https://en.wikipedia.org/wiki/Protection_ring [21.04.2019].

[6] Proxmox: Open-Source Virtualization Platform, https://www.proxmox.com/en/proxmox-ve [23.04.2019].

[7] Wikipedia: OpenVZ, https://pl.wikipedia.org/wiki/OpenVZ [23.04.2019].

[8] OpenVZ: Open source container-based virtualziation for Linux, https://openvz.org/ [22.04.2019].

[9] Opennebula: The Opennebula Project, https://opennebula.org/

[23.04.2019].

[10] Admin-Magazine: Setting u an OpenNebula Cloud, http://www.admin-magazine.com/CloudAge/Articles/Setting-up-an-OpenNebula-Cloud [24.04.2019].

[11] VMware: What ESXi Delivers, https://www.vmware.com/products/esxi-and-esx.html [22.04.2019].

[12] Wikipedia: Vmware ESXi, https://en.wikipedia.org/wiki/VMware_ESXi [21.04.2019].

[13] X. Miao, J. Han, The Design of a Private Cloud Infrastructure Based on XEN 2016.

[14] P. Kałucki, P. Dymora, M. Mazurek, Badanie wydajności wybranych systemów wirtualizacji, https://yadda.icm.edu.pl/baztech/element/bwmeta1.element.baztech-ff27331f-5aa8-43eb-a453-0fd21a6b4a43/c/kalucki_badanie_wydajnosci_2_2017.pdf [22.04.2019].

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 12 (2019) 220-225
Wysłane: 2019-06-10
Przyjęte: 2019-06-21

Analiza wpływu technologii oraz metod przesyłania hologramu na parametry i efekty transmisji

Krzysztof Mazur*, Damian Mazur
Politechnika Lubelska, Instytut Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. Celem artykułu jest porównanie i analiza wpływu technologii i technik przesyłania danych, pod kątem wyświetlania obrazu przy pomocy ostrosłupa holograficznego. Oceniając użyteczność rozwiązania pod uwagę będą brane parametry: czas przesyłania klatek obrazu, użycie parametrów fizycznych maszyny i parametry pracy Wirtualnej Maszyny Javy.

Słowa kluczowe: Hologram; OpenCv; Java; JDK

*Autor do korespondencji.

Adres e-mail: krzysztof.mazur94@gmail.com

Analysis of the impact of technologies and methods of hologram transmission on the parameters and effects of transmission

Krzysztof Mazur*, Damian Mazur
Institute of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract. The aim of the article is to compare and analyze the impact of technologies and data transfer techniques in term of displaying the image using a holographic pyramid. When assessing the usability of the solution, the following parameters will be taken into account: time of image transfer, use of physical parameters of the machine and parameters of the Java Virtual Machine.

Keywords: Hologram; OpenCv; Java; JDK

*Corresponding author.

E-mail address: krzysztof.mazur94@gmail.com

1. Wstęp

Człowiek od zarania dziejów starał się prezentować otaczający go świat przy pomocy obrazów. Pierwsze rysunki datowane są na okres kultury magdaleńskiej (17000 – 15000 lat p.n.e.) i znajdują się one w jaskini Lascaux. Przedstawiają one w większości zwierzęta, namalowane przy pomocy palców, pędzli czy prymitywnych aerografów zrobionych z kości.

Od powstania malowideł w Lascaux, minęły już tysiąclecia, a technika znacząco się rozwinęła. W 1920r. polski fizyk, profesor Mieczysław Wolfke, opracowując teoretyczne podstawy i rozbijając proces tworzenia obrazu na oddzielne fazy, stworzył podwaliny dzisiejszej holografii.

Myśląc o zastosowaniu holografii w pierwszym momencie pojawia się wizja trójwymiarowych projekcji, czyli holowizji znana z telewizji. Należy jednak pamiętać, że takie hologramy stoją jeszcze poza zasięgiem ludzkości. Jednak hologramy stosowane są już w medycynie, przy wykrywaniu komórek rakowych, w informatyce jako pamięci holograficzne, czy zabezpieczenia przed fałszowaniem dokumentów.

Przed osiągnięciem hologramów znanych z telewizji czy gier komputerowych, musimy pokonać jeszcze kilka barier technologicznych. Możemy jednak już z powodzeniem symulować podobny efekt przy pomocy urządzenia nazywanego piramidą holograficzną.

Przesyłanie obrazu holograficznego wymaga przesłania sekwencji obrazów holograficznych. Aby osiągnąć płynność wyświetlania tego typu projekcji niezbędne jest osiągnięcie odpowiedniej liczny klatek na sekundę. Dlatego też autorzy podjęli się określenia czynników wpływających na szybkość przesyłania tego typu danych. Zbadano wpływ użycia wybranych technologii oraz technik transmisji, a uzyskane wyniki przeanalizowano.

2. Piramida holograficzna

Piramida holograficzna przedstawiona na rysunku 1, jest to ostrosłup, ze ściętym szczytem, wykonany z przezroczystych ścian. Wstawiany jest on na ekranie urządzenia. Zapewnienie odpowiedniej jakości obrazu jest kluczowym elementem, by hologram sprawiał wrażenie rzeczywistego. Poszarpane krawędzie wyświetlanego obiektu sprawią, że obraz pokazany na ściankach piramidy, będzie wyglądał sztucznie.

Tworzenie hologramu w przypadku przeprowadzanych badań, polegało na pobraniu obrazu z kamery oraz wycięciu z klatki tła, co odróżnia proces od standardowego przesyłania wideo z kamer.

Rys. 1. Przykład piramidy holograficznej

3. Opis badań oraz aplikacji

Aplikacja przygotowana na potrzeby wykonania badań, umożliwiała dowolne wymienianie technologii użytych w procesie tworzenia hologramu. Napisana została w języku Java. Umożliwia ona pobranie klatek z podłączonych kamer, przeprocesowaniu ich przez algorytm wycinający tło oraz wysłaniu klatki do aplikacji klienckiej. Ważnym elementem aplikacji, jest jej konfigurowalność, umożliwiająca wyłączenie poszczególnych elementów procesu, dzięki czemu na badany jest rzeczywisty wpływ danej technologii np. dla testów WebSocket oraz Http wyłączono procesowanie klatki przez algorytmy wycinające tło, ponieważ badany był jedynie czas przesyłania danych od momentu wysłania klatki z serwera do odebrania jej po stronie klienta.

Analiza podzielona została na 4 etapy. Pierwszym z nich jest badanie metody przesyłania danych, w której to porównane zostały technologie Http i WebSocket. W tym celu te same obrazki, przesyłane były z użyciem obu technologii, na tym etapie analizy, brany pod uwagę był jedynie czas w jakim przesłano określoną liczbę klatek.

Kolejnym etapem analizy, było zbadanie wydajności działania aplikacji skompilowanej i uruchomionej przy pomocy różnych JDK (Java Development Kit). Wybrane zostały cztery najpopularniejsze wydania JDK: OpenJDK, Oracle JDK, GraalVM JDK i Zulu JDK. Badanie polegało na skompilowaniu i uruchomieniu aplikacji na każdym z podanych JDK. Jako kryteria oceny brane pod uwagę były: czas uruchomienia aplikacji, czas zajętości procesora, użycie pamięci RAM.

Przedostatnim badanym elementem, był serwer aplikacyjny. Aplikacja z poprzednich badań uruchomiona została na Jetty i Tomcat. Dla każdego z nich analizowano: czasy przesyłania klatek przy pomocy WebSocket, czas zajętości procesora i użycie pamięci RAM.

Ostatnie badanie polegało na przeprocesowaniu tego samego obrazka, przy pomocy dwóch algorytmów, służących do wycinania tła. Analizowane były algorytmy, z biblioteki OpenCV, KNN i MOG. Oba algorytmy wycinały tło z tego samego obrazka, a badane były: czas procesu oraz jakość efektu algorytmu.

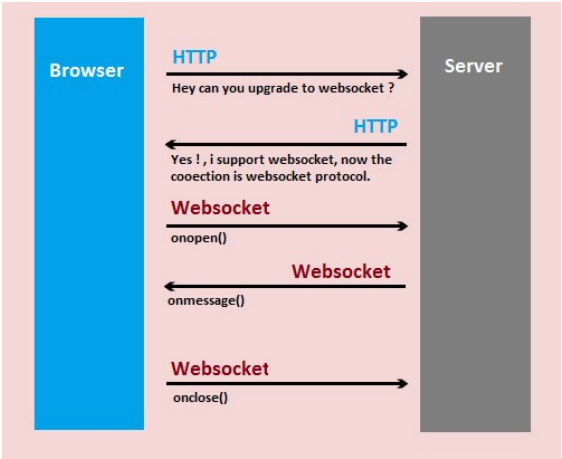
Przeprowadzenie wszystkich badań, pozwoliło na wybranie najlepszych zestawów technologii, służących do wyświetlania hologramu przy pomocy piramidy holograficznej, dla aplikacji napisanych w języku Java.

4. Http i WebSocket

Przesyłanie informacji jest głównym zagadnieniem do którego wykorzystywany jest Internet. Na przestrzeni lat, opracowano kilka protokołów, którymi można komunikować ze sobą oddzielne jednostki.

Jednym z nich jest protokół HTTP. Komunikacja odbywa się przy pomocy żądań klienta do serwera oraz odpowiedzi serwera do klienta na te żądania. Oba elementy mają określoną strukturę, którą muszą spełniać. HTTP zaliczany jest do protokołów bezstanowych, czyli serwer w momencie wysłania odpowiedzi do klienta, nie przechowuje informacji o obsłużonym żądaniu. W przypadku chęci zapamiętania stanu, należy wykorzystać mechanizm ciasteczek lub utrwalać sesję po stronie serwera. Architekturą komunikacji bazującą na protokole HTTP, jest REST

WebSocket jest protokołem komunikacyjnym zapewniającym dwustronną komunikację. Odbywa się ona na zasadzie zestawienia połączenia, między dwiema jednostkami i wymianie wiadomości. Różni się od HTTP ze względu na fakt, że wiadomości mogą być wysyłane, przez zestawione połączenie, bez schematu żądań/odpowiedzi.

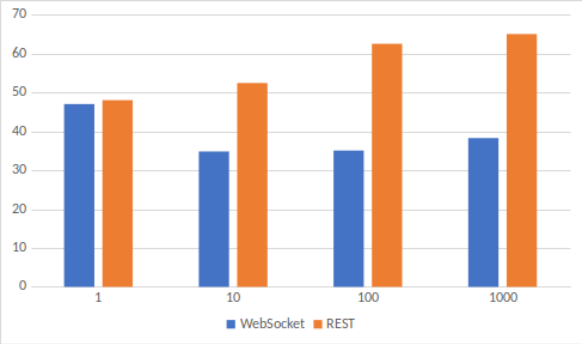


Rys. 2. Schemat zestawiania połączenia WebSocket, między przeglądarką a serwerem [https://www.pushtechnology.com/support/kb/understanding-the-java-virtual-machine-heap-for-high-performance-applications/]

Jak można zauważyć na rysunku 2 nawiązywanie połączenia WebSocket podzielone jest na kilka etapów. Pierwszym z nich jest wysłanie żądania HTTP rozpoczynając fazę "uścisku dłoni". W momencie, gdy serwer odpowie, że umożliwia komunikację przy pomocy WebSocket, połączenie przechodzi aktualizację i z HTTP zamieniane jest na połączenie WebSocket. Od tego momentu i klient i serwer mogą przesyłać wiadomości przy pomocy zestawionego połączenia [1].

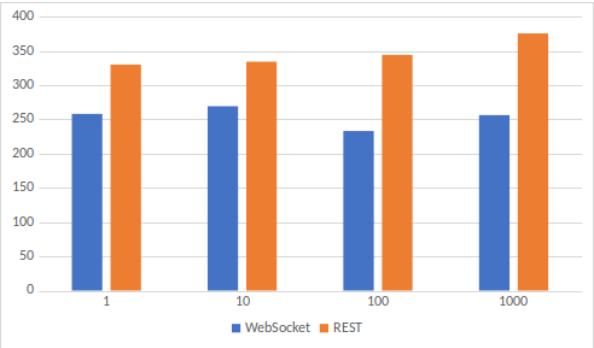
Analizę wpływu protokołu na czas przesyłania danych podzielono na trzy części. W każdej z nich użyto pliku o innej

wielkości, są to odpowiednio: 250KB, 500KB i 1MB. Każdy z obrazków przesyłano przy pomocy WebSocket i HTTP wykorzystując architekturę REST.



Rys. 3. Średnia czasu obsługi żądania dla obrazku 250KB

Jak widać na wykresie z rysunku 3, przesyłając jedną paczkę przy pomocy obu technik nie ma dużej różnicy. Można powiedzieć, że czasy są takie same. Różnica zaczyna być istotna dla dziesięciu paczek. Średnia czasów dla WebSocket jest mniejsza niż wynik dla jednej paczki. Jest to spowodowane tym, że przed zestawieniem połączenia WebSocket, musi zostać ono zaktualizowane z połączenia HTTP. Każda kolejna wiadomość nie jest już obarczona dodatkowym czasem. Średnia czasów REST jest większa niż dla jednego żądania. Największa różnica jest widoczna dla stu próbek. Średnia WebSocket jest dwukrotnie mniejsza niż dla REST i wynosi 35ms (milisekund), gdzie użycie protokołu HTTP wymaga średnio 60ms. Dla tysiąca próbek widać tą samą tendencję co dla poprzedniego badania.



Rys. 4. Średnia czasu obsługi żądania dla obrazku 250KB

Wyniki dla badania przeprowadzonego dla obrazu o wielkości 1MB, rysunek 4, potwierdzają wcześniej wyciągnięte wnioski.

5. Java

Java jest językiem programowania, którego główną zaletą jest możliwość uruchomienia na dowolnych systemie, jest to możliwe, dzięki użyciu Wirtualnej Maszyny Javy. Oczywiście wraz z korzyściami jakie niesie za sobą takie rozwiązanie, trzeba liczyć się z tym, że uruchamiany program wykona się wolniej, niż kod natywny pod konkretną platformę.

Wydajność aplikacji Javowych, zależy m.in. od użytego JDK (Java Development Kit - z ang. Zestaw Narzędzi Deweloperskich), użytej technologii przesyłania danych czy od rodzaje serwera, na którym aplikacja jest uruchomiona. Każde JDK, mimo że spełniają jeden standard, to różnią się działaniem i szczegółami implementacyjnymi. Podobnie jest w przypadku serwerów, choć każdy z nich jest w stanie uruchomić te same aplikacje, sposób działania może się różnić, a co za tym idzie, mogą mieć lepsze i gorsze zastosowania. Wymiana jednego z elementów, może skutkować poprawą działania programu lub spadkiem wydajności.

Aplikacja została skompilowana i uruchomiona na czterech różnych JDK:

- Oracle JDK,
- Zulu JDK,
- OpenJDK,
- GraalVM.

Każdy z programów działał przez 2 minuty i 35 sekund. Po dwudziestu sekundach działania programu, rozpoczyna się publikowanie klatek obrazu z odstępem stu milisekund.

5.1. Czas uruchomienia programu

Program przygotowany wykorzystuje framework Spring Boot w wersji 2.1.3. Jest to dodatkowy moduł do frameworka Spring, który upraszcza przygotowanie konfiguracji. Czas uruchomienia aplikacji różnił się znacząco między JDK:

- GraalVM - 2.6 sekund,
- OpenJDK - 3.6 sekund,
- Zulu JDK - 4.1 sekund,
- Oracle JDK - 2.2 sekund.

Każdy z JDK załadował inną liczbę klas do JVM.

Tabela 1. Liczba załadowanych klas

Nazwa JDK	Liczba klas
GraalVM	6741
OpenJDK	6468
Oracle JDK	6544
Zulu JDK	6558

Podobnie, jak w przypadku badania przeprowadzonego przez autora artykułu [2], GraalVM załadował więcej klas niż pozostałe JDK, lecz jest to o wiele mniejsza różnica. W artykule było to 25% w porównaniu do OpenJDK, w aktualnym badaniu różnica wynosiła tylko 5%. Jest prawdopodobne, że wraz z kolejnymi wersjami, GraalVM zoptymalizuje mechanizmy ClassLoadera. Należy jednak pamiętać, że sam fakt ładowania większej ilości klas do pamięci nie jest problematyczny, ale może mieć to wpływa na czas działania mechanizmów JVM np. Garbage Collectora.

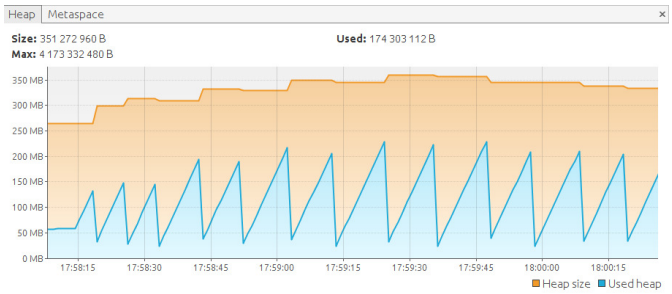
5.2. Użycie procesora

Wykresy pracy programu dla wszystkich JDK pokazały, że skoki użycia procesora pokrywają się z uruchomieniem Garbage Collectora. W standardzie Javy 8, domyślnym algorytmem Garbage Collectora jest Parallel GC [3].

Charakteryzuje się on częstymi uruchomieniami, wykorzystuje dwie strategie Mark-Copy dla nowej generacji oraz Mark-Sweep-Compact dla starej generacji. Faza stop-the-world blokuje wykonywanie programu przez cały czas trwania usuwania obiektów. Dodatkowo, mimo obaw, aplikacja uruchomiona na GraalVM nie wykorzystuje procesora w większym stopniu niż pozostałe JDK.

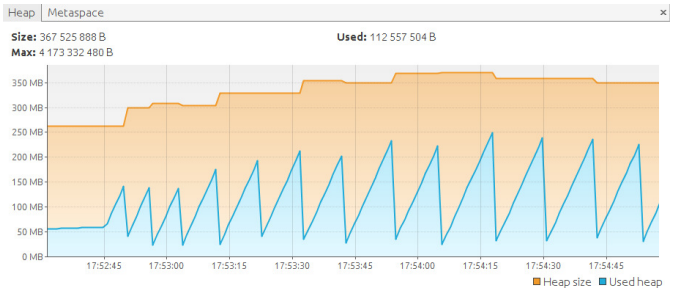
5.3. Użycie pamięci

Wszystkie aplikacje uruchamiane były bez dodatkowych parametrów określających wielkość sterty. Garbage Collector uruchamiał się automatycznie, bez wymuszania jego pracy.



Rys. 5. Wykres użycia pamięci dla Oracle JDK

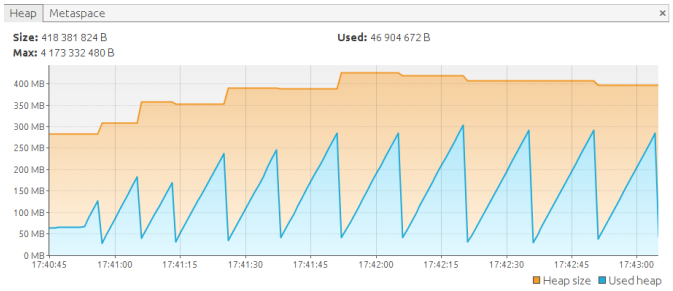
Analizując wykres wielkości sterty dla Oracle JDK z rysunku 5, można zauważyć, że aplikacja wykorzystywała w szczytowym momencie około 240MB pamięci. Każde uruchomienie Garbage Collectora powodowało usunięcie zbędnych obiektów i zwolnienie zasobów do poziomu około 50MB. Garbage Collector uruchamiał się w podobnych momentach. Przez pierwsze 30 sekund działania granicą startu algorytmu było około 150MB, następnie nastąpiła ewaluacja algorytmu i granica została przesunięta do poziomu około 200MB, by od minuty po uruchomieniu programu zatrzymać się na około 240MB.



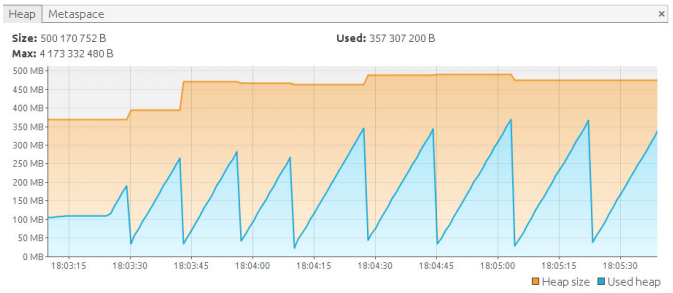
Rys. 6. Wykres użycia pamięci dla Open JDK

Podobne zachowanie, można zauważyć w przypadku wykresu dla Open JDK (rysunek 6). Profil uruchamiania się Garbage Collectora jest zbliżony zachowaniem do Oracle JDK. Jedyną różnicą jest maksymalne użycie pamięci, sięga ono równo 250MB, gdzie dla Oracle JDK wynosiło około 240MB. Jest to jednak nieznaczna różnica i może wynikać z obciążenia środowiska w trakcie działania aplikacji.

GraalVM wykorzystał nieznacznie więcej pamięci, największe użycie wynosiło 300MB (rysunek 7). Zdecydowanie więcej zasobów zajęło Zulu JDK, osiągając 350MB RAMu.



Rys. 7. Wykres użycia pamięci dla GraalVM JDK



Rys. 8. Wykres użycia pamięci dla Zulu JDK

Wszystkie badania przeprowadzane były z użyciem domyślnych ustawień dla JDK 8. Java oferuje dużą ilość parametrów pozwalającą skonfigurować środowisko w zależności od potrzeb. Jednym z takich ustawień jest *Xms* i *Xmx*. Oba parametry dotyczą ustawień sterty, pierwszy odpowiada za jej minimalną wielkość, a drugi za maksymalną. Pozwala to na ograniczenie pamięci jaką JVM będzie zajmować w systemie oraz bezpośrednio wpływa na uruchamianie algorytmu Garbage Collectora. Sam Garbage Collector posiada kilka wersji. Domyślnie dla JDK 8 używany jest Parallel, charakteryzuje się on tym, że nawet w przypadku znaczącego zmniejszenia wielkości sterty, zasoby nie są oddawane do systemu. Stosując dodatkowy parametr *XX:+UseG1GC* [4], mamy możliwość włączenia algorytmu G1, który znacząco zmienia sposób usuwania zbędnych obiektów z pamięci oraz w przeciwieństwie do Parallel zwalnia on zasoby systemu.

6. Serwer aplikacyjny

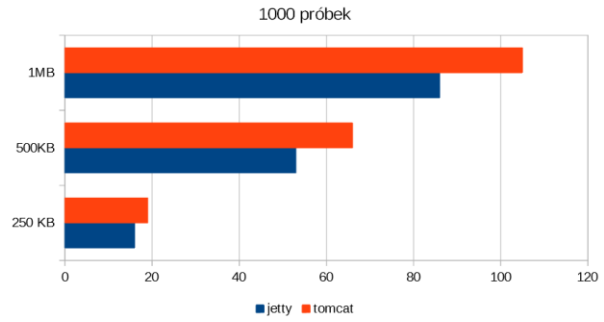
Aplikacje udostępniające dane do Internetu, napisane w języku Java, wymagają uruchomienia na serwerze lub kontenerze aplikacyjnym. Wynika to z faktu, że wykorzystując one technologię Servlet. Każdy z Servlet'ów posiada własny adres, którego to kontener będzie szukał w celu przekazania żądania.

Cykl życia servletu [5]:

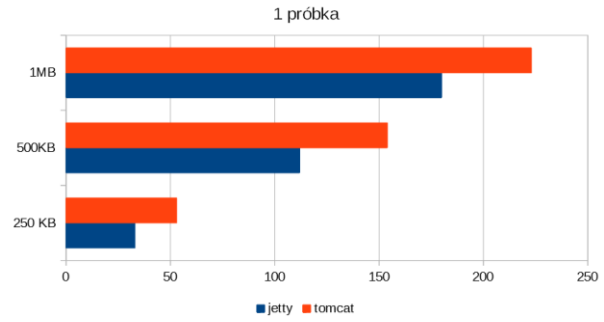
- kontener otrzymuje żądanie przez jedno ze swoich połączeń,
- kontener biorąc pod uwagę adres szuka servletu, który obsługuje żądanie,
- kontener sprawdza czy plik klasy jest załadowany, jeżeli nie, ładuje go, po czym tworzy instancję servletu,
- wywoływana jest metoda init(),
- wywoływana jest metoda service(),

- na zakończenie cyklu życia wywoływana jest metoda destroy().

Aplikacja została uruchomiona na Tomcacie oraz Jettym. Porównane zostały użycie pamięci oraz procent zajętości procesora w czasie działania aplikacji. Główną badaną statystyką jest czas w jakim przesłana zostanie klatka obrazu. Aplikacja wysyłała obrazy, przy pomocy technologii WebSocket, o rozmiarach odpowiednio 250KB, 500KB i 1MB. Próbki podzielone zostały na paczki po 1, 10, 100 i 1000 próbek.

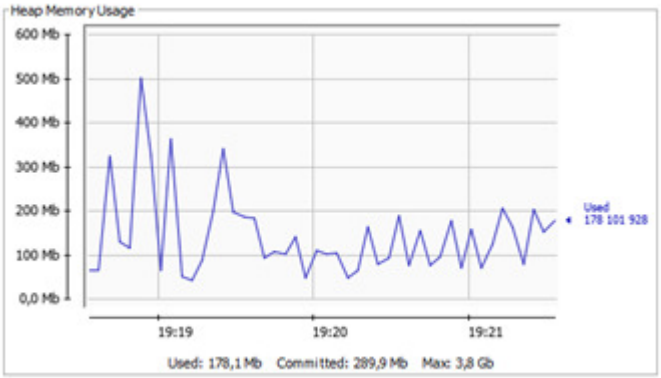


Rys. 9. Czasy dla przesyłu dla Tomcat i Jetty dla 1000 próbek

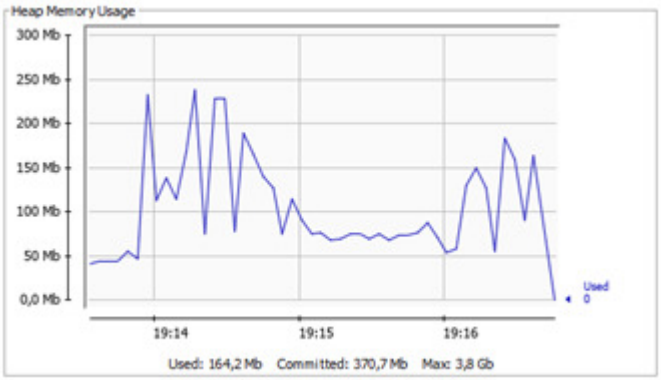


Rys. 10. Czasy dla przesyłu dla Tomcat i Jetty dla 1 próbki

Dane na wykresach z rysunków 9 i 10. faworyzują wybór Jetty nad Tomcatem. W obu przypadkach przesyłanie obrazu trwało krócej dla Jettiego. Podobnie było w przypadku paczek po 10 i 100 klatek. Analizując wykresy użycia pamięci (rysunki 11 i 12), zarezerwowanej na stertę, można dojść do wniosku, że Tomcat jest lepszym wyborem, w momencie gdy serwer posiada mniej pamięci RAM. Tomcat maksymalnie osiągnął zajętość około 240MB, co jest dwukrotnie mniejszą wielkością niż w przypadku Jettiego.



Rys. 11. Wykres użycia pamięci dla Jetty



Rys. 12. Wykres użycia pamięci dla Tomcat

Należy jednak zwrócić uwagę na fakt, że oba wykresy prezentują nierównomierne uruchamianie się Garbage Collectora.

7. OpenCV

Open Source Computer Vision jest to biblioteka programistyczna do przetwarzania obrazu w czasie rzeczywistym. Prace nad biblioteką rozpoczęły się oficjalnie w 1999 roku. Dotychczas zostało wydane 5 wersji aplikacji, natomiast pierwsza z nich została zaprezentowana w 2000 roku na konferencji poświęconej technikom rozpoznawania wzorców. Została ona napisana w jeżyku C, udostępniony jest jednak zbiór nakładek umożliwiający wykorzystanie biblioteki w językach takich jak:

- Java
- C
- C++
- Python.

Biblioteka OpenCv posiada wiele metod i technik przetwarzania obrazu w czasie rzeczywistym, dziedziny w których najczęściej wykorzystywana jest biblioteka to:

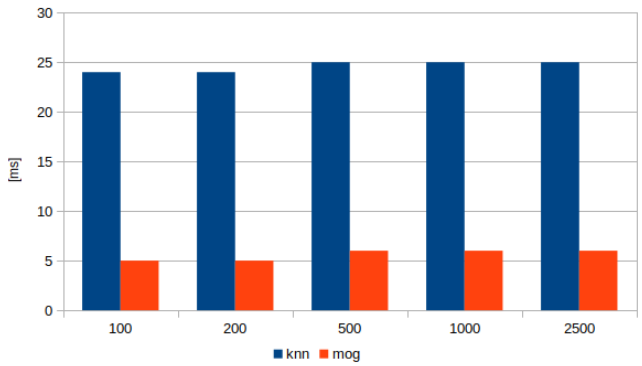
- rozpoznawanie twarzy,
- rozpoznawanie gestów,
- segmentacja obrazu,
- rozszerzona rzeczywistość,
- stereowizja,
- interakcja człowieka komputer.

Istotnym elementem pracy była implementacja mechanizmu do przetwarzania obrazu.

Przetwarzanie obrazu odbywało się z wykorzystaniem biblioteki OpenCv i polegało na wyekstrahowaniu pierwszego planu i wyświetlenie go na jednolitym tle którym zastąpiono plan drugi.

W bibliotece OpenCV do wyboru, są dwa algorytmy MOG i KNN [6]. Oba bazują na algorytmie GMM - Gaussian Mixture Model (z ang. Model Mieszaniny Gausa), w którym każdy piksel dostaje wagę w zależności od tego, jak długo piksel pozostaje w tym samym kolorze. Piksele, których kolor nie zmienia się długi czas, zostają sklasyfikowane jako tło [7].

Przeprowadzone badania, polegały na wyliczeniu czasu potrzebnego na ukończenie dwóch etapów, różnicowania i maskowania, dla pojedynczej klatki obrazu. Dodatkową zmienną był poziom progowania ustawiony odpowiednio na 100, 200, 500, 1000, 2500 jednostek.



Rys. 13. Czasy różnicowania dla algorytmów KNN i MOG

Na rysunku 13 widać znaczącą różnice w czasach. Algortym KNN potrzebował średnio pięciokrotnie więcej czasu, na różnicowanie tej samej klatki, dla tych samych wartości progowania.

Drugi etap algorytmu – maskowanie – trwał w obu przypadkach około 1ms.

Analizując wykresy, można stwierdzić, że algorytmy z grupy MOG, powinny być wybierane najczęściej, ponieważ są znacznie szybsze niż algorytm KNN. Dodatkowo dla jednej klatki, różnica w jakości uzyskanego, wyciętego obrazu jest znikoma. Należy jednak pamiętać, że w przypadku algorytmu MOG, każdorazowe poruszenie obiektu wycinanego z tła, powoduje zniekształcenie i konieczność ponownego ustalenia klatki tła. Dodatkową wadą jest wpływ oświetlenia, które musi być stałe, ponieważ w innym przypadku konieczne będzie ponowne dostrajanie. Na uzyskany efekt, duży wpływa mają parametry kamery. W przypadku słabego kontrastu i optyki, algorytm MOG może w zły sposób rozróżniać obiektu.

8. Wnioski

Celem przeprowadzonych badań było porównanie i analiza wpływu technologi i techniki przesyłania danych, pod kątem wyświetlania obrazu przy pomocy ostrosłupa holograficznego.

Początkowo badania przeprowadzane były przy pomocy dwóch komputerów. Na jednym pobierany był obraz, a drugi wyświetlał go przy pomocy ostrosłupa holograficznego. Dużą przeszkodą dla tej wersji badania okazywało się połączenie internetowe, które w znacznym stopniu wpływało na wahania czasów. Badania musiały zostać powtórzone i wykonane tylko na jednej jednostce roboczej, tak by odizolować środowisko badawcze od wpływów czynników zewnętrznych.

Oczywiste jest też to, że na polepszenie jakości, z zachowaniem dużej prędkości działania rozwiązania, mają wpływ parametry komputera. Badania przeprowadzone były przy użyciu dwóch komputerów: stacjonarnym i laptopie. Jednostka stacjonarna posiadała lepszy procesor, co skutkowało krótszymi czasami przetwarzania.

Podsumowując badania, można łatwo dojść do wniosku, że najlepszym zestawem technologii i metod przesyłania jest:

- WebSocket - jako metoda przesyłu, jest zdecydowanie szybsza i używa mniej transferu niż porównywany z nią REST,
- Zulu JDK - ponieważ cechował się szybkością działania, odpowiednim zarządzaniem pamięcią oraz jest posiada on darmową licencję,
- Jetty - mimo używania większej ilości zasobów, jest to szybki kontener i w przypadku, gdy posiadamy lepszą infrastrukturę, powinien być pierwszym wyborem. Oczywiście może zostać z powodzeniem zastąpiony Tomcatem, w momencie, gdy zależy nam na wydajnym używaniem maszyn,
- KNN dla poruszających się obiektów / MOG dla statycznych obiektów - wybór algorytmu musi być podyktowany zastosowanym źródłem hologramu. Nie ma generycznego algorytmu pozwalającego na uzyskiwanie idealnego odróżnienia tła od obiektu.

Literatura

[1] M. West, An Introduction to WebSocket, https://blog.teamtreehouse.com/an-introduction-to-websockets [05.06.2019].

[2] K. Kumar, What are the various components of JDK (Java Development Kit) environment?, http://cs-fundamentals.com/tech-interview/java/components-of-jdk-environment.php [12.05.2019].

[3] Standard Edition HotSpot Virtual Machine Garbage Collection Tuning Guide, https://docs.oracle.com/javase/8/docs/technotes/guides/vm/gctuning/ergonomics.html#ergonomics [05.06.2019].

[4] Tuning JVM Garbage Collection for Production Deployments, https://docs.oracle.com/cd/E40972_01/doc.70/e40973/cnf_jvmgc.htm [05.06.2019].

[5] Servlet Lifecycle, https://docs.oracle.com/javaee/6/tutorial/doc/bnafi.html [05.06.2019].

[6] A. Kaehler, G. Bradski. Learning OpenCV 3: Computer Vision in C++ with the OpenCV Library, https://books.google.pl/books?id=SKy3DQAAQBAJ&pg=PT26&redir_esc=y#v=onepage&q&f=fals [11.04.2019].

[7] Reynolds, Douglas A., Quatieri, Thomas F., Dunn, Robert B, Speaker verification using adapted Gaussian mixture models. Digital Signal Process, https://www.sciencedirect.com/science/article/pii/S1051200499903615?via%3Dihub [11.04.2019].

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 12 (2019) 226-231
Wysłane: 2019-06-11
Przyjęte: 2019-06-24

Analiza możliwości skrócenia czasu tworzenia aplikacji mobilnej na systemy Android oraz iOS przy użyciu technologii Xamarin

Daniel Molenda*, Maria Skublewska-Paszkowska

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

Streszczenie. Artykuł przedstawia porównanie czasu tworzenia aplikacji mobilnej na system Android i iOS wykonanej przy pomocy wieloplatformowej technologii Xamarin oraz wykonanej natywnie dla obu systemów osobno. Do badań wykorzystano autorską aplikację zaimplementowana w trzech środowiskach programistycznych: VisualStudio (przy użyciu technologii Xamarin), Android Studio oraz XCode. Wyniki przeprowadzonych badań wykazały, że utworzenie aplikacji w technologii Xamarin trwa o około połowę mniej czasu.

Słowa kluczowe: aplikacja mobilna; aplikacja wieloplatformowa; Android; iOS; Xamarin

*Autor do korespondencji.

Adres e-mail: daniel.m.92@wp.pl

Analysis of the possibility of shortening the time of creating a mobile application for Android and iOS systems using Xamarin technology

Daniel Molenda*, Maria Skublewska-Paszkowska

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

Abstract. The article presents a comparison of times needed for creating a mobile application for Android and iOS system using multi-platform Xamarin technology and respective native technologies for both systems. The authors' application was implemented in three programming environments: VisualStudio (using Xamarin technology), Android Studio and XCode. The results of the conducted research have shown that the creation of an application in Xamarin technology takes approximately half the time.

Keywords: mobile application; cross-platform application; Android; iOS; Xamarin

*Corresponding author.

E-mail address: daniel.m.92@wp.pl

1. Wstęp

Optymalizacja czasu tworzenia oprogramowania jest niezwykle istotna nie tylko z punktu widzenia inwestorów, którzy jak najmniejszym kosztem chcą otrzymać gotowy produkt, ale także dla twórców tego oprogramowania. Bardzo dobrze jest to widoczne na przykładzie aplikacji mobilnych, gdzie występuje różnorodność systemów operacyjnych, na które są one przeznaczone. W tym przypadku producenci chcąc dotrzeć do jak największej grupy odbiorców, muszą liczyć się ze wzrostem kosztów jakie muszą ponieść, aby to osiągnąć. Natomiast twórcy tych aplikacji często stają przed zadaniem napisania tej samej funkcjonalności kilkukrotnie w różnych językach programowania. Z tego powodu powstają różnego rodzaju technologie, które pozwalają w mniejszym lub większym stopniu na „ujednolicenie" tego procesu.

Jedną z takich technologii jest Xamarin, który posiada wspólny interfejs do tworzenia aplikacji na najpopularniejsze systemy mobilne: Android oraz iOS [1]. Technologia ta została wykorzystana do zbadania w jakim stopniu przy jej użyciu można skrócić czas wytworzenia aplikacji mobilnej na te dwa systemy.

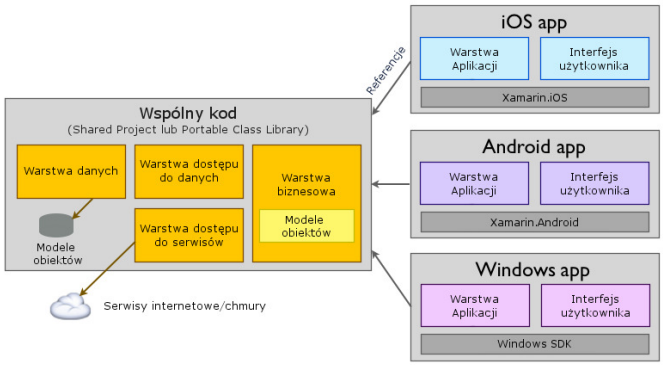
Istniej wiele publikacji dotyczących porównań technologii Xamarin z tradycyjnym podejściem do budowania aplikacji mobilnych. Większość z nich porusza tematykę porównania wydajności aplikacji, jak np. pozycja [2], w której autor porównuje kilka w różnym stopni skomplikowanych aplikacji – od „Hello World", aż po zaawansowane funkcjonalnie aplikacje pobierające dane z baz SQL oraz łączące się z serwisami internetowymi. Często publikowane są artykuły przedstawiające wady i zalety użycia technologii wieloplatformowych [3]. Nigdzie natomiast nie odnaleziono publikacji dotyczących porównania czasu tworzenia aplikacji mobilnej przy użyciu natywnych narzędzi (Android Studio, XCode) oraz aplikacji utworzonej w wieloplatformowej technologii Xamarin (Visual Studio). Być może jest to spowodowane trudnością oszacowania czasochłonności zbudowania takiej aplikacji, chociażby ze względu na wykorzystanie różnych języków programowania (Java, Swift, C#) oraz na znaczne różnice w dostępnych narzędziach ułatwiających ten proces.

2. Charakterystyka technologii Xamarin

Xamarin jest ujednoliconą oraz uproszczoną platformą do tworzenia aplikacji mobilnych na systemy Android oraz iOS, która wykorzystuje język programowania C# oraz platformę

.NET do dostarczania uniwersalnych rozwiązań na oba te systemy operacyjne. Platforma ta stworzona została w 2011 roku przez twórców Mono (zestawu narzędzi umożliwiających uruchomienie aplikacji stworzonych dla platformy Microsoft.NET w wielu środowiskach – m.in. w systemie Linux oraz macOS [4]), w której skład wchodzi Xamarin.Android (formalnie Mono for Android) oraz Xamarin.iOS (formalnie MonoTouch). Początkowo była to platforma posiadająca swoje własne środowisko programistyczne – Xamarin Studio, jednak po wykupieniu jej przez firmę Microsoft, stała się integralną częścią rozbudowanego środowiska programistycznego Visual Studio, które pozwalało na utworzenie osobnych projektów dla każdej z obsługiwanych platform (Android, iOS). Każde z utworzonych rozwiązań posiadało dedykowane narzędzia do tworzenia m.in. interfejsu użytkownika. Poza tym programista miał możliwość wyboru metody udostępniania wspólnego kodu (w postaci dodatkowego projektu) dla tych platform (Rys. 1):

1) Shared Project – którego kompilacja odbywała się przy wykorzystaniu bibliotek aktualnie wybranej platformy (Android lub iOS);
2) Portable Class Libraries – projekt kompilowany niezależnie, zawierający wspólne biblioteki dla obu platform.



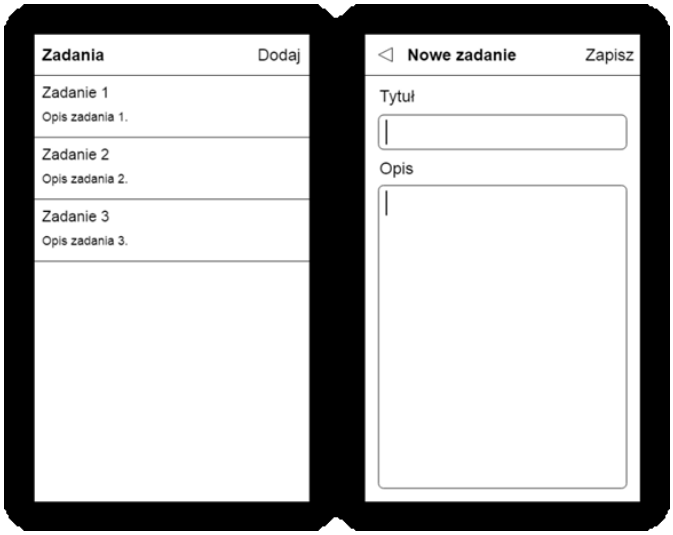
Rys. 1. Architektura udostępniania wspólnego kodu w platformie Xamarin

Dążąc do jeszcze większego ujednolicenia procesu tworzenia aplikacji mobilnych na wiele systemów operacyjnych, 28 maja 2014 roku twórcy platformy Xamarin opublikowali nowe narzędzie o nazwie Xamarin.Forms – dzięki któremu umożliwiono tworzenie jednego wspólnego interfejsu użytkownika dla wszystkich obsługiwanych platform. Jest to biblioteka składająca się z około 40 komponentów (układów oraz kontrolek), z których każdy ma swój natywny odpowiednik [5]. Początkowo możliwości tego narzędzia były ograniczone, jednak w miarę jego rozwoju twórcy zdecydowali się udostępnić możliwość dowolnego konfigurowania wyglądu oraz zachowania kontrolek na docelowych platformach za pomocą tzw. rendererów [6].

3. Aplikacja mobilna

W celu przeprowadzenia badań stworzona została aplikacja mobilna „Zadania". Umożliwia ona tworzenie listy zadań, które określane są poprzez tytuł oraz opis. Aplikacja posiada opcje wyświetlania oraz dodawania zadań na liście. Składa się ona z dwóch okien (Rys. 2):

1) Lista zadań – okno składające się z przewijanej listy zadań
2) Utworzenie nowego zadania – okno zawierające kontrolki edycyjne poszczególnych informacji o zadaniu.



Rys. 2. Prototyp okien aplikacji „Zadania". Po lewej – lista zadań, po prawej – tworzenie nowego zadania

Ten z pozoru prosty projekt realizuje wiele zagadnień związanych z funkcjonowaniem aplikacji mobilnych, takich jak: zarządzanie nawigacją pomiędzy oknami aplikacji oraz przesyłanie danych między nimi, oprogramowanie listy przewijanej wykorzystującej niestandardowy wygląd wierszy (spotykanej niemal w każdej aplikacji), czy wykorzystanie podstawowych kontrolek do wyświetlania oraz edycji teksu.

Aplikacja zaimplementowana została na dwa systemy mobilne: Android oraz iOS w trzech różnych środowiskach programistycznych:

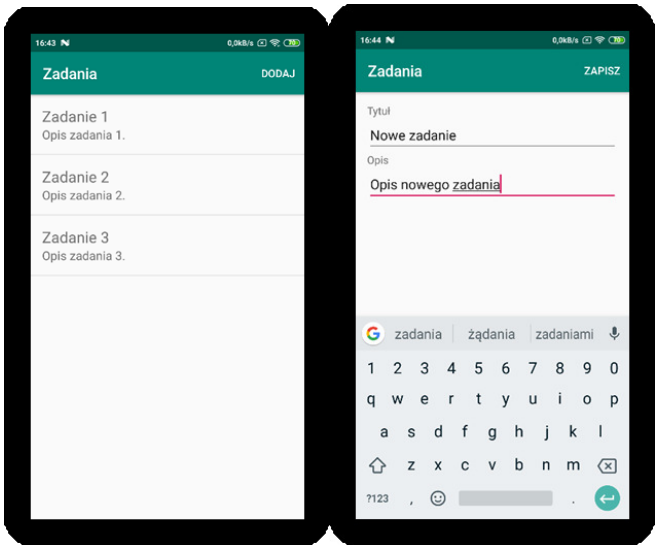
1) Android Studio przy użyciu języka programowania Java – system Android (Rys. 3);
2) XCode przy użyciu języka Swift 4 – system iOS (Rys. 4);
3) Visual Studio przy użyciu języka C# oraz technologii Xamarin. – wieloplatformowa aplikacja na systemy Android (Rys. 5) oraz iOS (Rys. 6).

Podczas badań porównywane będą aplikacje napisane natywnie – w środowiskach przeznaczonych wyłącznie do tworzenia aplikacji na jeden system (Android Studio, XCode) – oraz aplikacja napisana na obie platformy jednocześnie (Xamarin).

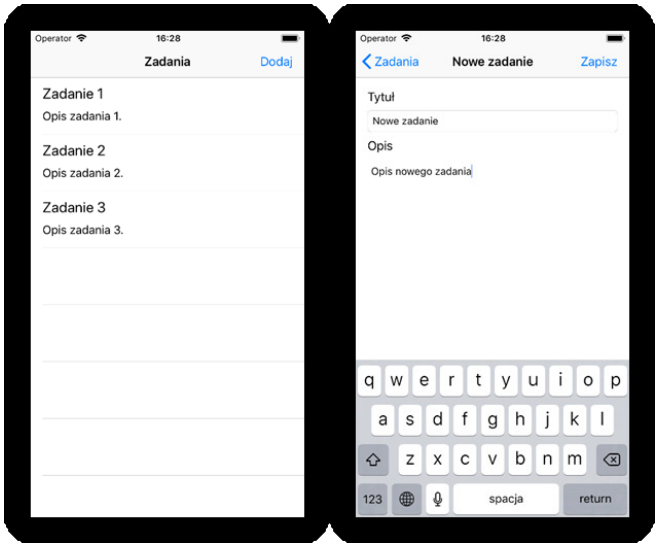
4. Przeprowadzone badania

Badania zostały podzielone na dwa etapy. Pierwszym z nich było porównanie czasów kompilacji aplikacji dla trzech wariantów:

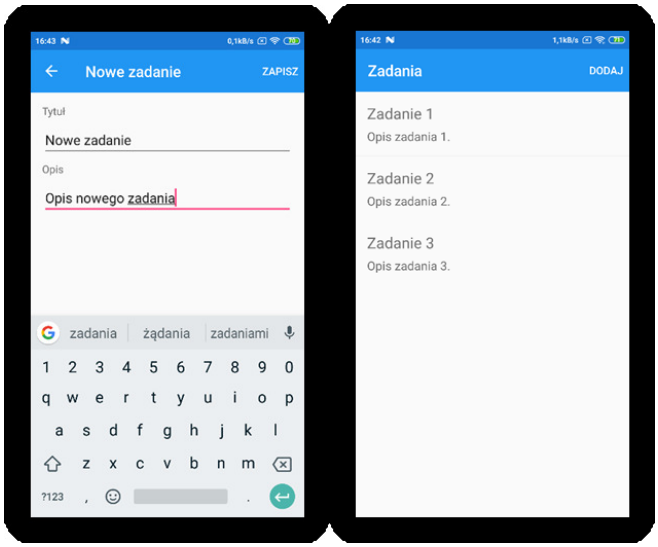
1) Pierwsza kompilacja – zmierzenie czasu kompilacji nowo utworzonego projektu;
2) Kompilacja po wprowadzeniu zmiany w implementacji aplikacji;
3) Kompilacja wynikowej aplikacji przeznaczonej do dystrybucji (np. w sklepie Google Play lub AppStore).



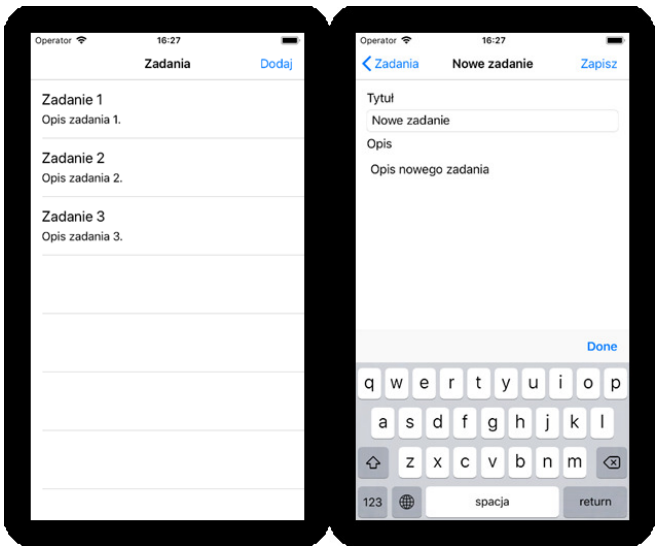
Rys. 3. Wygląd aplikacji mobilnej zadania na systemie Android utworzonej w środowisku Android Studio



Rys. 4. Wygląd aplikacji mobilnej zadania na systemie iOS utworzonej w środowisku XCode



Rys. 5. Wygląd aplikacji mobilnej „Zadania" na systemie Adnroid utworzonej w środowisku Visual Studio przy użyciu technologii Xamarin



Rys. 6. Wygląd aplikacji mobilnej „Zadania" w systemie iOS utworzonej w środowisku Visual Studio przy użyciu technologii Xamarin

Kolejny etap to porównanie czasu utworzenia samej aplikacji. Spotkano się tutaj z problemem oszacowania czasu ze względu na dość istotne różnice w wykorzystanych środowiskach programistycznych. Niektóre z nich to:

- wykorzystanie różnych języków programowania i związane z nimi możliwości,
- znaczne różnice w budowaniu interfejsu graficznego aplikacji,
- różnorodność narzędzi wspomagających tworzenie aplikacji, takich jak generowanie szablonów oraz automatyczne uzupełnianie kodu.

Z tego powodu postanowiono dokładnie opisać proces tworzenia przykładowej aplikacji „Zadania" na każdym z omawianych środowisk, a następnie odtworzenie go w rzeczywistości i zmierzenie czasu jego trwania poprzez nagranie ekranu w trakcie jego realizacji. Proces ten podzielono na trzy części:

1) Utworzenie i uruchomienie projektu;
2) Utworzenie oraz oprogramowanie okna listy zadań;
3) Dodanie okna oraz funkcji tworzenia nowych zadań.

Dzięki takiemu podziałowi, wraz z wynikami z poprzedniego etapu badań, utworzono dwa uproszczone procesy utworzenia oraz dystrybucji aplikacji „Zadania" na dwa systemy operacyjne Android oraz iOS: jeden zakładający wykonanie aplikacji tradycyjnie w dwóch osobnych środowiskach (Android Studio, XCode) oraz drugi – przy użyciu wieloplatformowej technologii Xamarin w środowisku Visual Studio (Tabela 1).

Mając na uwadze przytoczone wcześniej różnice w wykorzystanych do utworzenia aplikacji „Zadania" środowiskach, podjęto następujące kroki, aby zaprezentowany sposób porównania czasu tworzenia tej aplikacji był w jak największym stopniu wiarygodny:

1) Dokładnie sprecyzowano krok po kroku każdą wykonaną czynność podczas tworzenia aplikacji;

2) Proces tworzenia został w maksymalnym stopniu zoptymalizowany dla każdego z wykorzystanych środowisk;

3) Aplikacja wykonana została przez programistę z 4 letnim doświadczeniem w tworzeniu aplikacji mobilnych na systemy Android ora iOS, w każdym z odmawianych środowisk;
4) Starano się zachować jak największą płynność w trakcie tworzenia aplikacji.

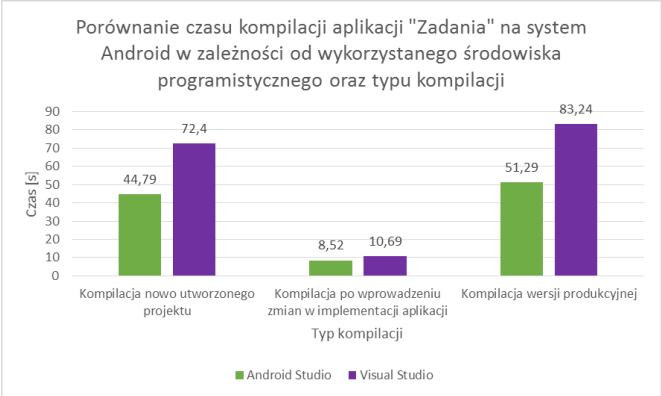
Tabela 1. Procesy tworzenia aplikacji mobilnej „Zadania" z podziałem na wykorzystane środowisko programistyczne

Lp.	Etap	
	Android Studio, XCode	**Visual Studio**
1	Utworzenie i uruchomienie projektu w środowisku Android Studio	Utworzenie i uruchomienie projektu w środowisku Visual Studio
2	Utworzenie oraz oprogramowanie okna listy zadań	Utworzenie oraz oprogramowanie okna listy zadań
3	Kompilacja nowo utworzonego projektu	Kompilacja nowo utworzonego projektu na system Android
4	Dodanie okna oraz funkcji tworzenia nowych zadań	Kompilacja nowo utworzonego projektu na system iOS
5	Kompilacja i uruchomienie aplikacji po wprowadzeniu zmian w implementacji	Dodanie okna oraz funkcji tworzenia nowych zadań
6	Kompilacja wynikowej aplikacji na system Android przeznaczonej do dystrybucji	Kompilacja i uruchomienie aplikacji po wprowadzeniu zmian w implementacji na systemie Android
7	Utworzenie i uruchomienie projektu w środowisku XCode	Kompilacja i uruchomienie aplikacji po wprowadzeniu zmian w implementacji na systemie iOS
8	Utworzenie oraz oprogramowanie okna listy zadań	Kompilacja wynikowej aplikacji na system Android przeznaczonej do dystrybucji
9	Kompilacja nowo utworzonego projektu	Kompilacja wynikowej aplikacji na system iOS przeznaczonej do dystrybucji
10	Dodanie okna oraz funkcji tworzenia nowych zadań	
11	Kompilacja i uruchomienie aplikacji po wprowadzeniu zmian w implementacji	
12	Kompilacja wynikowej aplikacji na system iOS przeznaczonej do dystrybucji	

5. Wyniki badań

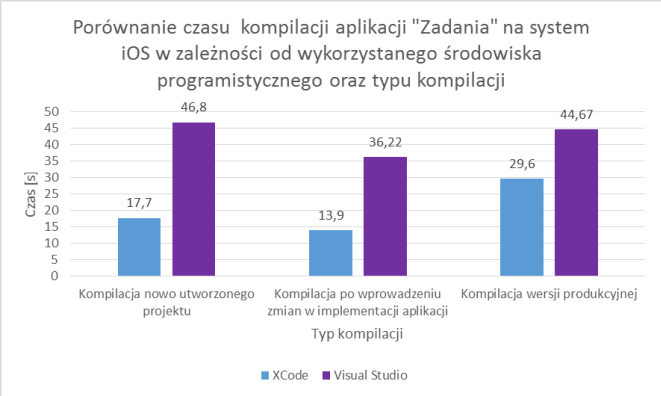
Na Rys. 7 oraz Rys. 8 przedstawiono wyniki badań polegających na zmierzeniu czasu trwania kompilacji aplikacji mobilnej na systemy Android (Rys. 7) oraz iOS (Rys. 8) ze względu na użyte środowisko programistyczne oraz warunki w jakich kompilacja została przeprowadzona. Zbadano kolejno: czas kompilacji nowo utworzonego projektu, czas kompilacji po wprowadzeniu zmiany w implementacji aplikacji oraz czas kompilacji aplikacji w wersji produkcyjnej przeznaczonej do dalszej dystrybucji. Na podstawie tych danych na pierwszy rzut oka można stwierdzić, że niezależnie od systemu operacyjnego oraz warunków przeprowadzenia kompilacji w każdym przypadku kompilacja aplikacji wykonanej przy użyciu technologii Xamarin trwa znacznie dłużej.

W przypadku systemu Android (Rys. 7) kompilacja nowo utworzonego projektu w środowisku Visual Studio trwała o około 61% dłużej czasu niż w środowisku Android Studio. Podobnie więcej czasu zajęła także kompilacja wersji produkcyjnej aplikacji (około 62%). Warto natomiast przyjrzeć się czasom kompilacji aplikacji po wprowadzeniu zmian w implementacji, która jest wykonywana stosunkowo częściej niż dwa przeanalizowane powyżej typy, ponieważ czynność tę powtarza się przy każdej okazji poprawiania błędów w implementacji, czy dodawania nowych funkcji aplikacji. Różnice w tym przypadku są niewielkie, ponieważ jest to tylko o około 25% więcej czasu w przypadku środowiska Visual Studio.



Rys. 7. Porównanie czasu kompilacji aplikacji "Zadania" na system Android w zależności od wykorzystanego środowiska programistycznego oraz typu kompilacji

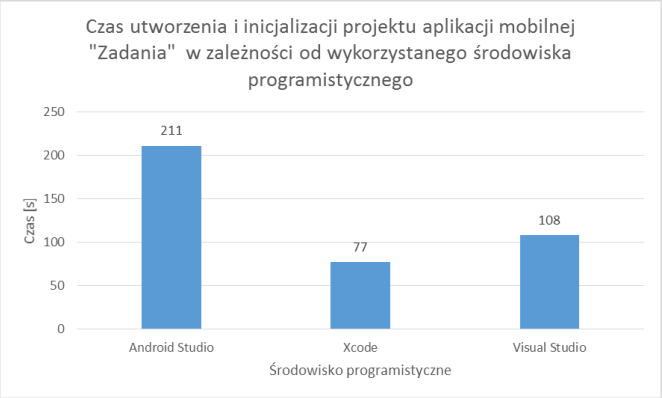
W przypadku systemu iOS (Rys. 8) różnice dla tych samych typów kompilacji są znacznie wyraźniejsze niż dla systemu Android. Największą różnicę stanowi tutaj kompilacja nowo utworzonego projektu, ponieważ o około 164% więcej czasu zajęła ona w środowisku Visual Studio. Około 160% więcej czasu zajęła także kompilacja po wprowadzeniu zmiany w implementacji aplikacji w środowisku Visual Studio. Najmniejszą różnice w czasach można zaobserwować podczas kompilacji wersji produkcyjnej – zajęło to o około 50% więcej czasu w przypadku środowiska Visual Studio.



Rys. 8. Porównanie czasu kompilacji aplikacji "Zadania" na system iOS w zależności od wykorzystanego środowiska programistycznego oraz typu kompilacji

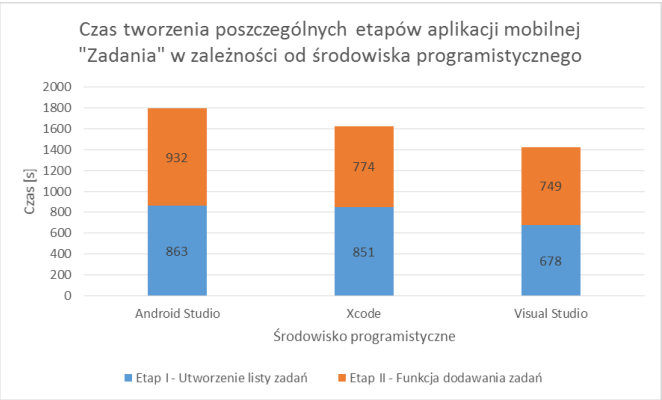
Na Rys. 9 przedstawiono czas utworzenia i inicjalizacji projektu aplikacji „Zadania", w każdym z trzech

wykorzystanych środowisk programistycznych: Android Studio, XCode oraz Visual Studio. Warto zwrócić uwagę na środowisko Android Studio, które potrzebowało znacznie więcej czasu na inicjalizację i przygotowanie projektu do pracy. W stosunku do środowiska XCode było to o około 174% więcej czasu, natomiast w porównaniu do Visual Studio – około 95% więcej czasu.



Rys. 9. Czas utworzenia i inicjalizacji projektu aplikacji mobilnej w zależności od wykorzystanego środowiska programistycznego

Rys. 10 przedstawia porównanie czasów trwania poszczególnych etapów implementacji aplikacji mobilnej „Zadania" w zależności od wykorzystanych środowisk programistycznych. Na podstawie przedstawionych wyników można stwierdzić, że najmniej czasu zajęło zaimplementowanie aplikacji w środowisku Visual Studio. Należy tutaj także wspomnieć, że implementacja ta jednocześnie kierowana jest na dwa systemy operacyjne (Android oraz iOS), ponieważ wykonana została przy użyciu wieloplatformowej technologii Xamarin. W stosunku do implementacji w środowisku Android Studio czas ten był krótszy o około 20% oraz o około 12% – w porównaniu ze środowiskiem XCode.



Rys. 10. Czas tworzenia poszczególnych etapów aplikacji mobilnej "Zadania" w zależności od środowiska programistycznego

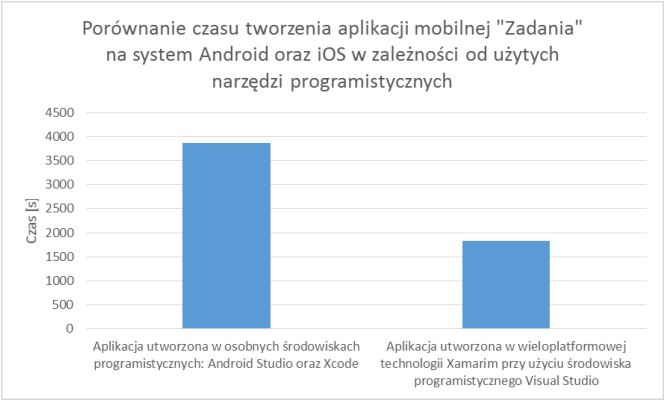
W Tabela 2 oraz Tabela 3 przedstawiono czasy trwania poszczególnych Etapów tworzenia aplikacji „Zadania" w zależności od wybranej metody tworzenia: w dwóch osobnych środowiskach (Android Studio, XCode) (Tabela 2) oraz w wieloplatformowej technologii Xamarin przy użyciu środowiska programistycznego Visual Studio (Tabela 3). Podsumowanie tych czasów jako sumaryczny czas tworzenia aplikacji mobilnej „Zadania" na system Android oraz iOS w zależności od użytych narzędzi przedstawiono na Rys. 11, z którego wynika, że czas potrzebny do utworzenia aplikacji mobilnej „Zadania", przy użyciu wieloplatformowej technologii Xamarin oraz środowiska programistycznego Visual Studio, jest krótszy o około 53% w porównaniu z utworzeniem tej samej aplikacji używając dwóch oddzielnych środowisk: Android Studio oraz XCode.

Tabela 2. Czas trwania poszczególnych etapów tworzenia aplikacji w dwóch osobnych środowiskach programistycznych Android Studio oraz XCode

Lp.	Etap	Czas trwania [s]
1	Utworzenie i uruchomienie projektu w środowisku Android Studio	211
2	Utworzenie oraz oprogramowanie okna listy zadań	863
3	Kompilacja nowo utworzonego projektu	44,79
4	Dodanie okna oraz funkcji tworzenia nowych zadań	932
5	Kompilacja i uruchomienie aplikacji po wprowadzeniu zmian w implementacji	8,52
6	Kompilacja wynikowej aplikacji na system Android przeznaczonej do dystrybucji	51,29
7	Utworzenie i uruchomienie projektu w środowisku XCode	77
8	Utworzenie oraz oprogramowanie okna listy zadań	851
9	Kompilacja nowo utworzonego projektu	17,7
10	Dodanie okna oraz funkcji tworzenia nowych zadań	774
11	Kompilacja i uruchomienie aplikacji po wprowadzeniu zmian w implementacji	13,9
12	Kompilacja wynikowej aplikacji na system iOS przeznaczonej do dystrybucji	29,6
	Suma	3873,8

Tabela 3. Czas trwania poszczególnych etapów tworzenia aplikacji w wieloplatformowej technologii Xamarin przy użyciu środowiska programistycznego Visual Studio

Lp.	Etap	Czas trwania [s]
1	Utworzenie i uruchomienie projektu w środowisku Visual Studio	108
2	Utworzenie oraz oprogramowanie okna listy zadań	678
3	Kompilacja nowo utworzonego projektu na system Android	72,4
4	Kompilacja nowo utworzonego projektu na system iOS	46,8
5	Dodanie okna oraz funkcji tworzenia nowych zadań	749
6	Kompilacja i uruchomienie aplikacji po wprowadzeniu zmian w implementacji na systemie Android	10,69
7	Kompilacja i uruchomienie aplikacji po wprowadzeniu zmian w implementacji na systemie iOS	36,22
8	Kompilacja wynikowej aplikacji na system Android przeznaczonej do dystrybucji	83,24
9	Kompilacja wynikowej aplikacji na system iOS przeznaczonej do dystrybucji	44,67
	Suma	1829,02



Rys. 11. Porównanie czasu tworzenia aplikacji mobilnej "Zadania" na system Android oraz iOS w zależności od użytych narzędzi programistycznych

6. Wnioski

Podsumowując zebrane wyniki badań można stwierdzić, że użycie technologii Xamarin w kontekście czasu tworzenia aplikacji na dwa systemy mobilne (Android oraz iOS), jest jak najbardziej uzasadnione, ponieważ pozwala to na zaoszczędzenie ponad połowy czasu w stosunku do wytworzenia tej samej aplikacji osobno na każdy z systemów. Warto jednak mieć na uwadze wyniki z pierwszego etapu badań (dotyczącego czasów kompilacji aplikacji), gdzie wskazano znaczne różnice przemawiające na niekorzyść technologii Xamarin. Różnice te wynikają z dodatkowych zadań jakie są wykonywane podczas kompilacji aplikacji w środowisku Visual Studio przy użyciu technologii Xamarin. Z uwagi na to, że kod pisany jest w języku C# (innym niż natywny dla danej platformy), to musi on być najpierw odpowiednio przetworzony (w zależności platformy) do postaci natywnej [7]. Dodatkowo, w trakcie tego przetwarzania, wykonywany jest proces linkowania – usuwania z wynikowego kodu nieużywanych klas, w celu zmniejszenia objętości aplikacji [8] [9]. Niemniej jednak pomimo, iż kompilacje aplikacji w procesie jej tworzenia są dość często wykonywaną czynnością (chociażby przy wprowadzaniu drobnych poprawek, czy podczas poszukiwania i poprawiania błędów w implementacji), to w stosunku do sumarycznego czasu potrzebnego do jej wytworzenia jest to niewielki ułamek.

Literatura

[1] http://gs.statcounter.com/os-market-share/mobile/worldwide/#monthly-201903-201904-bar [15.05.2019].

[2] https://www.altexsoft.com/blog/engineering/performance-comparison-xamarin-forms-xamarin-ios-xamarin-android-vs-android-and-ios-native-applications [15.05.2019].

[3] Corral L., Janes A., Remencius T.: Potential advantages and disadvantages of multiplatform development frameworks – A vision on mobile environments. Procedia Computer ScienceVolume 10, 2012, s. 1202-1207.

[4] https://pl.wikipedia.org/wiki/Mono_(oprogramowanie) [15.05.2019].

[5] https://docs.microsoft.com/en-us/xamarin/xamarin-forms/app-fundamentals/custom-renderer/renderers [15.05.2019].

[6] Ch. Petzold, Cross-platform C# programming for iOS, Android, and Windows, Microsoft Press, Redmond, 2016.

[7] https://docs.microsoft.com/en-us/xamarin/cross-platform/app-fundamentals/building-cross-platform-applications/understanding-the-xamarin-mobile-platform [15.05.2019].

[8] https://docs.microsoft.com/en-us/xamarin/ios/deploy-test/linker [15.05.2019].

[9] https://docs.microsoft.com/en-us/xamarin/android/deploy-test/linker. [15.05.2019].

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 12 (2019) 232-239
Wysłane: 2019-06-14
Przyjęte: 2019-06-22

Autorski system inteligentnego budynku w porównaniu z rozwiązaniem otwarto - źródłowym

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

Streszczenie. Artykuł ten jest próbą odpowiedzi na pytanie: czy i w jakich warunkach, opłacalne jest opracowanie autorskiego systemu inteligentnego budynku, kiedy dostępnych jest wiele bezpłatnych systemów o otwartym kodzie źródłowym. Publikacja przedstawia charakterystykę autorskiego systemu automatyki domowej – sHome, a także systemu open-source – Domoticz, w konfiguracji realizującej możliwie najbardziej zbliżoną do funkcjonalności systemu autorskiego. Pracę kończy analiza porównawcza systemów oraz wnioski z przeprowadzonej analizy.

Słowa kluczowe: inteligentny budynek; automatyka domowa; system autorski; system otwarto-źródłowy.

*Autor do korespondencji.

Adres e-mail: cezary@kryczka.eu

Own system of intelligent building in comparison with an open - source solution

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

Abstract. This article is an attempt to answer the question whether and under what conditions it is beneficial to develop an own intelligent building system, when many free open source systems are available. The publication presents the characteristics of author's own home automation system - sHome, as well as the open-source system - Domoticz, in a configuration that is as close to the functionality of the author's system as possible. The work ends with a comparative analysis of the systems and conclusions from the analysis.

Keywords: intelligent building; smart home; home automation; own system; open-source system.

*Corresponding author.

E-mail address: cezary@kryczka.eu

1. Wstęp

Artykuł ten jest próbą odpowiedzi na pytanie: czy i w jakich warunkach, opłacalne jest opracowanie autorskiego systemu inteligentnego budynku, kiedy dostępnych jest wiele bezpłatnych systemów o otwartym kodzie źródłowym, które można wykorzystać do realizacji nawet bardzo rozbudowanych instalacji.

Ponieważ autor artykułu nie odnalazł żadnego rzetelnego opracowania na temat popularności systemów open-source w inteligentnych budynkach, przeprowadził samodzielny przegląd dostępnych w Internecie artykułów dotyczących takich systemów i na tej podstawie określił jakie rozwiązania są popularne oraz który z systemów jest najczęściej opisywany. Metody wybrane przez autora do zbadania powyższych zagadnień to przegląd artykułów oraz eksperyment, polegający na budowie autorskiego systemu i późniejszym porównaniu go z najpopularniejszym rozwiązaniem open-source.

Autor zakłada, że poprzez budowę i analizę nawet prostego systemu automatyzującego sterowanie urządzeniami w domu oraz porównanie go z istniejącymi rozwiązaniami można przeprowadzić szereg analiz i wyciągnąć wnioski z uruchomienia takich systemów.

2. Definicja domu inteligentnego

Określeniem „inteligentny dom” można określić zarówno prosty system monitorujący podstawowe parametry jak temperatura, czy stan czujników oraz pozwalający na zdalną kontrolę odbiorników prądu, ale również rozbudowany system oparty o sztuczną inteligencję i sieci neuronowe, który będzie umożliwiał nie tylko sterowanie za pomocą dedykowanego interfejsu ale również będzie w stanie prowadzić swobodną konwersację w języku naturalnym [1]. Cechą wspólną dla wszystkich systemów automatyki jest to, że każdy z nich do prawidłowego działania potrzebuje czujników i elementów wykonawczych, którymi będzie mógł sterować.

3. Przegląd rozwiązań open-source

Na podstawie dokonanego przeglądu artykułów na stronach internetowych, które określają ogólną popularność systemów automatyki domowej oraz prognoz rynkowych, powołując się na badania firm z tej branży lub branż pokrewnych, autor określił najbardziej popularne systemy inteligentnych budynków oferowane na licencji open-source, przedstawione w tabeli 1 [5-14]:

Tabela 1. Ranking popularności systemów open-source

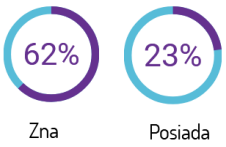
Lp.	Nazwa systemu	Ilość wystąpień
1	Domoticz	10
2	OpenHAB	10
3	Home Assistant	9
4	Calaos	7
5	OpenMotics	6
6	MisterHouse	3
7	ioBroker	2
8	Jeedom	2
9	LinuxMCE	2
10	OpenNetHome	2
11	PiDome	2
12	EventGhost	1
13	Fhem	1
14	FreeDomotic	1
15	Guardian Project	1
16	MajorDoMo	1
17	Mycontroller	1
18	Mycroft	1
19	MyPi	1
20	Node-RED	1
21	OpenRemote	1
22	Pimatic	1
23	Pytomation	1
24	Smarthomatic	1

W analizowanych artykułach wymienione były 24 systemy, z czego dwa, Domoticz i OpenHAB, występowały w każdej publikacji. Home Assistant to nieznacznie mniej popularny system, wystąpił on w 9 na 10 artykułów.

Według danych Komisji Europejskiej z 2018 roku, obecnie w Unii Europejskiej jest około 8,5 mln inteligentnych domów i apartamentów, na rok 2021 szacuje się liczbę 80 mln (Rys. 1.), co oznacza niemal dziesięciokrotny wzrost popularności takich rozwiązań w okresie 3 lat [2]. Trudno przewidzieć jaką część z nich będą stanowiły rozwiązania open-source, szczególnie, że znakomita większość tych rozwiązań dostarcza tylko oprogramowanie, a cześć sprzętową użytkownicy muszą uruchomić samodzielnie korzystając np. z popularnych mikrokomputerów jak Raspberry Pi.

Rys. 1. Prognozy rynkowe [2]

Według badań serwisu Oferteo.pl przeprowadzonych wśród osób, które w 2017 roku budowały dom, rozwiązania inteligentnych budynków są znane 62% ankietowanych. 23% pytanych osób zdecydowało się na wdrożenie takich rozwiązań (Rys. 2) [6].



Rys. 2. Znajomość systemów inteligentnych domów [6]

Badania przeprowadzone dla firmy Somfy pokazują, że 31% respondentów planuje posiadanie urządzeń będących elementem inteligentnych budynków. Eksperci uważają, że wraz z rosnącą satysfakcją z użytkowania takich systemów będzie przybywało jej zwolenników i posiadaczy. Wśród badanych osób tylko 15% użytkowników nie jest zadowolonych z posiadania takich rozwiązań, natomiast 62% jest z nich zadowolonych [8].

W serwisie github.com, w którym przechowywany może być kod źródłowy aplikacji open-source, zapytanie „smart home" zwraca ponad 7000 wyników, a zapytanie „home automation system" ponad 1300 wyników. Świadczy to o tym, że istnieje wiele otwarto źródłowych systemów inteligentnych budynków. Oczywiście duża część wyników dotyczy dodatkowych modułów do istniejących rozwiązań, jednak zakładając, że 10% to unikatowe systemy daje to liczbę około 800 systemów open-source.

4. Kryteria oceny badanych systemów

Aby rzetelnie ocenić, czy w obecnych warunkach opracowywanie autorskiego systemu inteligentnego budynku jest uzasadnione, należy wyznaczyć kryteria oceny, które pozwolą wyciągnąć wnioski, na temat okoliczności w jakich opracowanie takiego systemu jest racjonalne.

4.1. Kryterium czasu opracowania

W tym kryterium analizowany był czas potrzeby do zbudowania i uruchomienia autorskiego projektu, przy założeniu, że autor ma podstawową wiedzę w zakresie posługiwania się przynajmniej jednym językiem programowania. Czas stworzenia systemu autorskiego został porównany z czasem uruchomienia wybranego systemu open-source o podobnej funkcjonalności.

4.2. Kryterium bezpieczeństwa aplikacji i komunikacji

Analiza przeprowadzona na potrzeby niniejszej publikacji ma charakter bardzo uproszczonego badania na wąski zakres podatności. Na potrzeby niniejszego artykułu aplikacje zostały przebadane pod kątem podatności na najpopularniejsze od wielu lat zagrożenie dla aplikacji internetowych tj. SQL Injection [21, 22].

W kontekście bezpieczeństwa komunikacji systemy zostały przebadane pod kątem wykorzystanych mediów i protokołów zabezpieczających transmisję danych wewnątrz

systemu oraz komunikację z użytkownikiem za pośrednictwem dedykowanego interfejsu.

4.3. Kryterium skalowalności

Badanie aplikacji pod względem skalowalności polegało na porównaniu jakie możliwości rozbudowy i dostosowania systemu do innego wdrożenia oferuje rozwiązanie autorskie, a jakie rozwiązanie open-source.

4.4. Kryterium kosztów

Pod względem biznesowego wdrożenia aplikacji, duże znaczenie mają koszty i pracochłonność rozwiązania. W kryterium kosztów analizowana jest opłacalność budowy systemu autorskiego oraz oszacowane zostały koszty wdrożenia wybranego systemu open-source.

4.5. Kryterium wartości dodanych

W tym kryterium analizie zostały poddane umiejętności i kompetencje, które można rozwinąć podczas wdrożenia gotowego rozwiązania open-source, w porównaniu z tymi, jakie można zdobyć lub rozwinąć projektując i wdrażając system autorski.

4.6. Kryterium trwałości w czasie

W tym kryterium aplikacje zostaną poddane analizie pod kątem starzenia się kodu, określone zostaną także nakłady niezbędne do utrzymania dobrej jakości systemu.

5. Systemu autorski – sHome

System opracowany na potrzeby niniejszej publikacji jest prostym systemem inteligentnego domu, którego funkcjonalność sprowadza się do odczytu danych z czujników i sterowania elementami wykonawczymi.

5.1. Wymagania funkcjonalne

Wymagania funkcjonalne określają realizowane przez system funkcje i ich zakres. W przedstawionym systemie autorskim określono następujące wymagania funkcjonalne:

1) Sterowanie oświetleniem: włącz/wyłącz.
2) Sterowanie zasilaniem urządzeń elektrycznych (sterowanie gniazdami elektrycznymi).
3) Monitorowanie stanu sterowanych urządzeń.
4) Automatyczna reakcja na zagrożenia, np. odcięcie dopływu wody za pomocą elektrozaworu w przypadku wykrycia zalania.
5) Zarządzanie użytkownikami systemu.

5.2. Wymagania niefunkcjonalne

Wymagania niefunkcjonalne określają cechy jakimi powinien charakteryzować się system aby był użyteczny i wygodny w obsłudze. System autorski zakłada następujące wymagania w tym zakresie:

1) Dostępność systemu 24 godziny na dobę, przez 7 dni w tygodniu.
2) Prosty, czytelny, atrakcyjny wizualnie i ergonomiczny interfejs.
3) Dostępność również dla osób bez wykształcenia technicznego.
4) Czas ładowania interfejsu nie powinien przekraczać 1,5s.
5) Jak najmniejsza ingerencja w istniejące instalacje.
6) Interfejs www działający na większości popularnych przeglądarek.
7) Elastyczność implementacji – możliwość wdrożenia systemu na etapie projektowania instalacji, a także uruchomienia w budynku z istniejącą instalacją.

5.3. Architektura systemu

Architektura systemu zakłada działanie centralnego urządzenia (serwera) zarządzającego całym system. W systemie sHome rolę tę pełni mikrokomputer Raspberry Pi 3+, pracujący pod kontrolą systemu operacyjnego Raspbian. Do sterowania urządzeniami wykonawczymi oraz do odczytu danych z czujników wykorzystane są wbudowane w Raspberry Pi piny GPIO (General Purpose Input/Output, ang. wejście/wyjście ogólnego przeznaczenia), które stanowią interfejs wejścia i wyjścia. Interfejs użytkownika stanowi strona internetowa napisana w języku PHP, za pomocą której wywoływane są skrypty napisane w języku Python, które odpowiadają za komunikację z elementami peryferyjnymi (czujnikami, przekaźnikami i diodami). Dane są przechowywane w bazie danych MySQL, a komunikacja sieciowa oparta jest na sieci WiFi, z zastosowaniem szyfrowania WPA.

5.4. Platforma sprzętowa i jej parametry techniczne

Istotną częścią systemu automatyki domowej są czujniki. W systemie sHome zastosowane zostały dwa rodzaje czujników. Czujniki temperatury DS18B20, pracujący w zakresie temperatur: -55°C to +125°C, o dokładności ±0.5°C w zakresie -10°C – +85°C, poza tym zakresem, dokładność wynosi ±2°C. Komunikacja odbywa się za pomocą protokołu 1-Wire [20]. Czujnik zalania Yl-83 Rain Detector, charakteryzuje się czułością minimalną na poziomie 0.05cm² (istnieje możliwość regulacji), opóźnienie wyłączenia to około 5min (zależne od czasu schnięcia wody). Posiada on zarówno wyjście cyfrowe (DO, detekcja – stan niski), jak i analogowe (AO) [21]. Badany system korzysta z wyjścia cyfrowego. Elementami wykonawczymi są podwójne moduły przekaźników, które odpowiadają za sterowanie gniazdami elektrycznymi oraz dioda LED, symbolizująca oświetlenie.

5.5. Czas przygotowania systemu

System sHome został napisany przez jedną osobę – autora niniejszej publikacji, który ma niewielkie doświadczenie programistyczne. Czas powstania systemu to około pięć tygodni, czas pracy w tygodniu wyniósł średnio 40 godzin. Zatem łączny czas realizacji projektu zajął około 200 godzin.

5.6. Bezpieczeństwo systemu i komunikacji

Dzięki zastosowaniu sanityzacji i walidacji ciągu znaków wpisywanych do formularzy przez użytkowników, za pomocą funkcji htmlentities() oraz mysqli mysql_real_escape_string(), system sHome jest odporny na ataki typu SQL Injection. Dzięki wykorzystaniu tych funkcji, wprowadzana do formularza treść jest filtrowana i wykonane jest jedynie właściwe zapytanie. Drugą istotną kwestią dotyczącą bezpieczeństwa systemu jest wyłonienie grup użytkowników, którzy mają dostęp tylko do niezbędnych funkcji.

Za zapewnienie bezpieczeństwa komunikacji w systemie odpowiada kilka mechanizmów, w zależności od tego, z jakim elementem systemu zachodzi komunikacja. Bezpieczeństwo dostępu do interfejsu webowego jest zapewnione poprzez protokół SSL, który zapewnia szyfrowanie transmisji pomiędzy serwerem, a przeglądarką internetową. Bezpieczny dostęp do terminala systemu Raspbian odpowiada protokół SSH, który jest jedynym dopuszczonym do komunikacji z konsolą systemową. Bezpieczeństwo transmisji sieciowej jest zapewnione przez protokół WPA, który odpowiada za szyfrowanie danych pomiędzy bezprzewodowym punktem dostępowym i mikrokomputerem Raspberry Pi.

5.7. Skalowalność

Skalowalność systemu sHome jest zapewniana przez konstrukcję interfejsu i bazy danych, które pozwalają na proste dodawanie nowych urządzeń do systemu. Dodanie nowego urządzenia wykonawczego sprowadza się do dodania nowego wpisu do bazy za pomocą interfejsu graficznego. Wybrany model obsługi czujników temperatury pozwala na dodawanie kolejnych termometrów poprzez dołączenie ich do istniejącej szyny 1-Wire oraz odczytanie i konwersję wartości temperatury za pomocą skryptu. Dzięki wbudowanej w Raspberry Pi natywnej obsłudze sieci, można również rozbudować system o bezprzewodowe moduły (np. ESP8266), za pomocą których można zbierać dane z odległych czujników oraz sterować urządzeniami elektrycznymi po połączeniu z czujnikami lub modułami przekaźników. Powyższe czynności mogą być wykonane podczas pracy systemu i nie wymagają jego rekompilacji, dzięki wykorzystaniu skryptowych języków programowania. Możliwe jest również uruchomienie systemu sHome na innej platformie sprzętowej, jednak wymagałoby to opracowania nowych skryptów do obsługi interfejsu wejścia/wyjścia.

5.8. Koszty

Aby określić wysokość kosztów poniesionych na zbudowanie systemu sHome, należy określić co będzie uwzględniało to kryterium. Z pewnością najłatwiej jest obliczyć koszt użytego sprzętu:

- Raspberry Pi 3 B+
- Karta 32GB
- Zasilacz
- Płytka prototypowa
- Moduł Przekaźnika (podwójny)
- Czujnik zalania
- Dioda LED

Trudniej jest określić koszt pracy. Tutaj należałoby zwrócić uwagę na koszt alternatywny, rozumiany przez wartość najlepszej utraconej korzyści, w wyniku poświęcenia czasu na budowę systemu. Zatem określenie kosztów pracy zostało zawężone jedynie do ilości godzin poświęconych na stworzenie systemu. W przypadku systemu autorskiego czas poświęcony na budowę systemu wyniósł około 200 godzin, nie uwzględniono tu jednak czasu przewidywanego na utrzymanie, aktualizację, czy rozbudowę systemu.

5.9. Wartości dodane

Podczas budowania systemu autor poszerzył swoją wiedzę z obszaru programowania w językach PHP, HTML, CSS oraz Python, którą może wykorzystać w innych projektach. Jest to wiedza o rozwiązaniach programistycznych, która jest uniwersalna i może stanowić bazę do jej pogłębiania i tworzenia nowych projektów, jak również podnosi wartość autora na rynku pracy. Zdobyte doświadczenie pozwala również na wybranie specjalizacji, która najbardziej odpowiada preferencjom autora.

5.10. Trwałość w czasie

Specyfika systemów informatycznych jest taka, że zmiany zachodzące w tym środowisku są bardzo szybkie i napisanie systemu raz, jako skończonej całości jest praktycznie niemożliwe. System sHome zbudowany przez jedną osobę, nie jest odporny na porzucenie go przez autora, zatem nie jest on również odporny na zmiany zachodzące w czasie. Dopóki zastosowane rozwiązania będą obowiązującymi, system będzie aktualny.

6. System open-source – Domoticz

System Domoticz pojawiał się we wszystkich analizowanych artykułach dotyczących inteligentnych domów opartych na otwartym kodzie źródłowym [5-14]. Jest on również pierwszy w przedstawionym w tabeli 1 zestawieniu. W związku z tym, system ten został wybrany do porównania go z systemem autorskim sHome. Przeprowadzenie analizy kolejnych systemów może stanowić wartościową kontynuację pracy badawczej autora.

6.1. Opis autorskiego systemu open-source – Domoticz

System Domoticz wspiera wiele rodzajów czujników i urządzeń wykonawczych, można go uruchomić na wielu platformach sprzętowych (procesory ARM32, ARM64, x86) i systemach operacyjnych takich jak: Windows, Mac/OSX, Linux, czy Synology. System ten wspiera także platformę Raspberry Pi oraz współpracuje z różnymi, również komercyjnymi rozwiązaniami automatyki domowej, takimi jak: MySensors, RFXCOM, Z-Wave, RF-Link, P1 Smart Meter, YouLess, Smart Meter, Pulse Counters, EnOcean, iTead Sonoff, Xiaomi. Interfejs webowy Domoticz pozwala na korzystanie z różnych tematów, które ułatwiają

personalizację jego wyglądu. Dostępna jest również aplikacja mobilna, która pozwala na sterowanie systemem oraz na odczyt stanu czujników i systemu.

6.2. Architektura systemu

Kod źródłowy systemu Domoticz napisany jest w języku C/C++. Głównym elementem tego systemu jest centralne urządzenie (serwer) zarządzające całym systemem. W badanej konfiguracji rolę tę pełni mikrokomputer Raspberry Pi 3+. Czujniki i urządzenia wykonawcze komunikują się z mikrokomputerem poprzez piny GPIO. System umożliwia także użycie bezprzewodowych modułów ESP8266, które dzięki wbudowanym własnym pinom GPIO pozwalają na bezprzewodowy odczyt danych z czujników, jak również sterowanie zasilaniem urządzeń elektrycznych za pośrednictwem np. przekaźników. Dla zachowania podobieństwa funkcjonalności do systemu autorskiego, ta funkcjonalność w badanej konfiguracji użyto wyłącznie pinów GPIO. Głównym interfejsem służącym do sterowania, zarządzania i monitorowania systemu jest strona www. Dane są przechowywane w bazie danych SQLite. Istnieje także aplikacja mobilna w bezpłatnej wersji, która umożliwia sterowanie automatyką domową oraz podgląd stanu systemu.

6.3. Czas przygotowania systemu

Na czas potrzebny na uruchomienie systemu Domoticz, składają się: przygotowanie i połączenie elementów sprzętowych, uruchomienie systemu operacyjnego, zapoznanie z dokumentacją, instalacja, uruchomienie i konfiguracja systemu Domoticz. Proces instalacji przebiega automatycznie po uruchomienia skryptu automatycznej instalacji. Łączny czas potrzebny do uruchomienia systemu w badanej konfiguracji wyniósł około 16 godzin.

6.4. Bezpieczeństwo systemu i komunikacji

Bezpieczeństwo systemu Domoticz jest zapewniane przez wiele mechanizmów. Jednym z nich są konta użytkowników i ich podział na grupy, pod względem uprawnień do sterownia i modyfikacji systemu. System Domoticz posiada trzy predefiniowane grupy użytkowników. Oprócz tego, system umożliwienie otwarcia interfejsu bez autoryzacji systemu użytkownikom łączącym się z określonych adresów IP, a także możliwe jest wymuszenie autoryzacji dla czujników zewnętrznych podłączanych poprzez protokoły sieciowe oraz istnieje możliwość zabezpieczenia hasłem dostępu sterowania przełącznikami.

System Domoticz został także zbadany pod kątem podatności na wstrzykiwanie kodu SQL (SQL Injection) [15]. Aby zbadać tę podatność zostały wykonane testy, polegające na wprowadzeniu w pole nazwy użytkownika i hasła, ciągu znaków pozwalających na modyfikację zapytania SQL w taki sposób by uzyskać dostęp do interfejsu systemu bez znajomości danych autoryzacyjnych. System Domoticz przeszedł te testy pomyślnie. Według informacji ze strony vuldb.com [16], system Domoticz, w wersji poniżej 4.10578 może być podatny na atak SQL Injection, poprzez funkcję CWebServer::GetFloorplanImage w pliku WebServer.cpp. Podatność polega na możliwości manipulacji parametrem idx tak by wykonać wstrzyknięcie kodu SQL. Opisana podatność ma wpływ na poufność, spójność i dostępność systemu. Informacja o podatności została opublikowana w dniu 2019-03-31. Identyfikatorem tej podatności jest CVE-2019-10664 [17]. Na dzień pisania artykułu szczegóły techniczne ataku są już znane, jednak pomimo, że możliwe jest zdalne przeprowadzenie ataku, według informacji zawartych na stronie vuldb.com, wiadomo że obecnie nie ma dostępnego exploita wykorzystującego tę podatność. Aktualizacja do wersji 4.10578 eliminuje tę podatność [16]. Wersja 4.10717, która została zainstalowana na potrzeby niniejszej publikacji, jest odporna na to zagrożenie.

6.5. Bezpieczeństwo komunikacji

Za bezpieczeństwo transmisji danych pomiędzy przeglądarką a serwerem systemu odpowiada protokół HTTPS, wykorzystujący protokół SSL do szyfrowania danych. Bezpieczny dostęp do terminala systemu operacyjnego zapewnia protokół SSH. Bezpieczeństwo komunikacji elementów systemu jest zapewniane przez przewodowe połączenie z czujnikami i urządzeniami wykonawczymi, co znacząco utrudnia np. atak "Man in the middle", polegający na modyfikacji danych pomiędzy serwerem, a czujnikami i urządzeniami wykonawczymi. Również atak typu DoS (brak dostępu do usługi ang. Denial of Service) jest znacznie utrudniony. Wymienione wyżej rodzaje ataku były możliwe jedynie poprzez fizyczny dostęp do infrastruktury, a atak typu DoS mógłby zostać wykonany także poprzez wywołanie silnego impulsu elektromagnetycznego. W przypadku, gdyby użytkownik chciał wykorzystać bezprzewodowe czujniki, czy też bezprzewodową komunikację z elementami wykonawczymi za pomocą sieci WIFI, za bezpieczną transmisję danych odpowiada protokół WPA, wspierany na poziomie systemu operacyjnego Raspbian.

6.6. Skalowalność

System Domoticz, poprzez swoją modułową budowę, jak również wykorzystanie możliwości definiowania wirtualnych urządzeń, daje bardzo duże możliwości rozbudowy automatyki domowej. Społeczność, programistów pracujących nad tym projektem, stara się nadążyć za współczesnymi wymaganiami takich systemów i dostarcza nowe rozwiązania. Bardzo duży wpływ na skalowalność tego rozwiązania ma fakt, że użytkownik może samodzielnie pisać własne skrypty, które stworzą bardzo szerokie pole do automatyzacji procesów w inteligentnym domu.

6.7. Koszty

Koszty budowy systemu open-source ograniczają się jedynie do warstwy sprzętowej, która jest tożsama z wykorzystaną w systemie autorskim. W warstwie programowej, wszystkie niezbędne elementy można uzyskać bezpłatnie, dzięki temu jest to projekt o otwartym kodzie. Jedynym kosztem dodatkowym może być płatna wersja aplikacji mobilnej, jednak udostępniana jest również bezpłatna wersja, która dostarcza najważniejsze

funkcjonalności. Ważnym czynnikiem wpływającym na koszty wdrożenia systemu jest czas potrzebny do jego uruchomienia, który, w przypadku badanej konfiguracji wyniósł około 16 godzin roboczych, tj. 2 dni.

6.8. Wartości dodane

W przypadku systemu open-source, bardzo duże znaczenie ma wsparcie społeczności tworzącej system Domoticz. Społeczność szybko reaguje na znalezione luki bezpieczeństwa oraz nieustannie rozbudowuje system. Oprócz tego istnieje możliwość przyłączenia zdalnych bezprzewodowych czujników m.in. za pomocą modułów ESP8266, co zwiększa jeszcze bardziej możliwości rozbudowy systemu. Istotny jest również krótki czas potrzebny do uruchomienia systemu.

6.9. Trwałość w czasie

System Domoticz jest tworzony przez grupę entuzjastów, którzy często są również użytkownikami tego systemu oraz pasjonatami nowoczesnych rozwiązań. Projekt jest na bieżąco aktualizowany i rozbudowywany. Jego kod podlega publicznej weryfikacji i dyskusji, a błędy są na bieżąco naprawiane. Nie zmienia to faktu, że jak każdy rozbudowany system jest podatny na błędy popełniane przez deweloperów. System będzie odporny na starzenie się tak długo, jak będzie istniała aktywnie rozwijająca go społeczność.

7. Analiza porównawcza

Aby porównanie systemów było jak najbardziej rzetelne, obydwa systemy pracowały na identycznym sprzęcie, tak, by różnice wynikały wyłącznie z obszaru oprogramowania i środowiska obydwu aplikacji (tabela 2).

Tabela 2. Porównanie systemów

Cecha systemu	sHome	Domoticz
Język programowania	PHP, Python	C/C++
Czas przygotowania	200 godz.	16 godz.
Środowisko sprzętowe	Raspberry Pi 3+	Raspberry Pi 3+
System operacyjny	Raspbian	Raspbian
Aplikacja mobilna	Nie	Tak

Zarówno system autorski, jak również system open-source Domoticz, w badanej konfiguracji spełniają podstawowe założenia dotyczące systemów inteligentnych budynków, określone na wstępie, w szczególności dzięki samodzielnemu reagowaniu na sygnały z czujników, czego przykładem jest automatyczne zamykanie zaworów wody w przypadku wykrycia zalania. Pod kątem złożoności kodu i środowiska programistycznego systemy różnią się zasadniczo. System Domoticz napisany został w języku C/C++ i oferuje znacznie więcej funkcjonalności niż system sHome. System autorski jest stworzony głównie w języku PHP z elementami języka Python i jest bardzo uproszczony względem systemu Domoticz, ponieważ zawiera tylko niezbędne funkcje określone w pierwotnych wymaganiach.

7.1. Kryterium czasu opracowania

W analizowanym przypadku, czas wdrożenia systemu open-source wyniósł 16 godzin, natomiast zakres prac niezbędnych do stworzenia autorskiego projektu jest zdecydowanie większy niż przy wdrożeniu gotowego projektu, co miało również znaczący wpływ na czas opracowania systemu. System autorski sHome powstawał przez około 200 godzin.

7.2. Kryterium bezpieczeństwa aplikacji i komunikacji

Ze względu na pierwotne ograniczenia nałożone podczas formułowania kryteriów analizy systemów pod kątem bezpieczeństwa oraz ograniczonej funkcjonalności systemów, zostały one przeanalizowane pod kątem podziału uprawnień użytkowników oraz podatności na najbardziej popularne zagrożenie dla aplikacji internetowych jakim według raportów OWASP (Open Web Application Security Project) jest SQL Injection [15], a także pod kątem wykorzystanych protokołów i mediów transmisyjnych odpowiadających za komunikację.

Oba analizowane systemy posiadają predefiniowane grupy użytkowników, które podnoszą poziom bezpieczeństwa systemów poprzez ograniczenie dostępu tylko do odpowiednich dla danej grupy użytkowników funkcji. Przeprowadzone testy pokazały również, że obydwa systemy są odporne na ataki SQL Injection. Starsze wersje systemu Domoticz były podatne na ten rodzaj ataku, przez błąd w zaimplementowanej funkcji, jednak dzięki szybkiej reakcji społeczności deweloperów zagrożenie to zostało wyeliminowane w kolejnych wersjach systemu. W badanej konfiguracji, w obydwu systemach, za bezpieczeństwo dostępu do konsoli systemu operacyjnego odpowiada protokół SSH, który jest jedynym protokołem komunikacyjnym uruchomionym na potrzeby zdalnego zarządzania systemem operacyjnym. Za bezpieczeństwo komunikacji z interfejsem webowym odpowiada protokół HTTPS, którzy szyfruje przesyłane dane za pomocą protokołu SSL. Czujniki i elementy wykonawcze są połączone za pomocą przewodowych połączeń, dzięki czemu możliwość ingerencji w przesyłane informacje jest znacznie ograniczona, a atak musiałby zakładać fizyczny dostęp do systemu lub wywołanie potężnego impulsu elektromagnetycznego. W przypadku zastosowania bezprzewodowych elementów wejścia/wyjścia, o które mogą być rozbudowane obydwa systemy, za szyfrowaną komunikację bezprzewodową odpowiada protokół WPA.

7.3. Kryterium skalowalności

System sHome, w podstawowym stopniu umożliwia jego rozbudowę. Jednak uproszczony sposób jest dostępny tylko dla elementów wykonawczych i czujników juz zdefiniowanych w systemie. Rozbudowa systemu o obsługę bezprzewodowych modułów ESP8266 wymagała by jednak opracowania kolejnych elementów systemu w postaci nowych skryptów i instalacji dodatkowych pakietów oprogramowania z systemie Raspbian.

System Domoticz zaraz po instalacji udostępnia szeroki wachlarz możliwości przyłączania nowego sprzętu, bez konieczności edycji kodu. Cała konfiguracja jest dostępna z poziomu strony internetowej. Przyłączenie dodatkowych elementów komunikujących się bezprzewodowo wymaga jedynie ich konfiguracji w systemie. Domoticz współpracuje między innymi z bardzo popularnym modułem bezprzewodowym ESP8266. Możliwość wgrania alternatywnego oprogramowania do modułu ESP (np. ESPEasy) [18], umożliwia bardzo proste dodawanie kolejnych elementów systemu.

7.4. Kryterium kosztów

Koszt elementów sprzętowych jest taki sam dla obydwu badanych systemów, ponieważ działają one na tym samym sprzęcie. Niewątpliwie największym kosztem systemu autorskiego jest czas jego projektowania i uruchomienia, który jest 12,5 krotnie większy od czasu wdrożenia systemu open-source.

7.5. Kryterium wartości dodanych

System open-source oferuje możliwość bardzo szybkiego uruchomienia i prostej rozbudowy systemu o kolejne elementy i funkcjonalności. Zbudowanie systemu autorskiego posiada znaczne walory edukacyjne. Napisanie własnego systemu wymaga rozważenia wielu problematycznych kwestii oraz przeglądu technologii, tak by wybrać tę najbardziej adekwatną do zadania. Daje również znacznie większą elastyczność w dopasowaniu rozwiązania do konkretnego wdrożenia.

7.6. Kryterium trwałości w czasie

System Domoticz, który powstał w 2014 roku, jest na bieżąco rozwijany i ulepszany przez społeczność programistów. Dzięki tej aktywności projekt stale ewoluuje i ma zdecydowanie większe prawdopodobieństwo przetrwania niż projekt sHome, wykonany przez jednego autora. Według danych z serwisu github.com (Rys. 3), system Domoticz tworzy grupa ponad 80 współpracowników, którzy tylko w okresie 30.04.2019 – 31.05.2019 dokonali zmian w 923 plikach, na które składa się 44117 zmian i 26847 usunięć w kodzie [19].

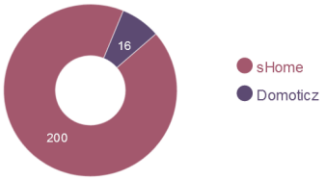
Rys. 3. Wkład społeczności w rozwój projektu Domoticz z wyłączeniem zatwierdzeń scalania (ang. commits) [19]

System sHome od momentu powstania jego finalnej wersji nie zmienił się ani razu.

8. Wnioski

Systemy inteligentnych budynków są coraz bardziej powszechne, budowa i wdrożenie takiego systemu ma wpływ na poprawę komfortu użytkowania domu, czy mieszkania, a także, pozwala na automatyczną reakcję systemu w przypadku wykrycia zagrożenia, bez konieczności wykonywania akcji przez użytkownika. Celem artykułu była próba odpowiedzi na pytanie: czy i w jakich warunkach uzasadnione jest budowanie takiego systemu, mając na względzie mnogość dostępnych bezpłatnie, gotowych rozwiązań o otwartym kodzie źródłowym. Zapewne porównanie kilku lub wszystkich wymienionych w niniejszym artykule systemów (tabela 1) z systemem autorskim znacznie podniosło by wartość merytoryczną publikacji, jednak ograniczenia czasowe i objętościowe dotyczące artykułu pozwoliły autorowi na przeprowadzenie analizy tylko jednego wybranego systemu open-source. Analiza porównawcza pozostałych systemów może stanowić wartościową kontynuację przeprowadzonych badań.

Analiza bezpieczeństwa badanych systemów wykazała, że system Domoticz, poprzez upublicznienie kodu i swoją popularność, jest bardziej niż system sHome narażony na ataki hackerskie, jednak aktywność społeczności podnosi poziom bezpieczeństwa, poprzez szybkie usuwanie odnalezionych luk w kodzie systemu. System autorski, może być również podatny na wiele zagrożeń, jednak tajność kodu i niewielka jego popularność działa na jego korzyść, poprzez zmniejszenie prawdopodobieństwa ataku. W warstwie komunikacyjnej odporność systemów wynika wprost z wykorzystanych, powszechnie znanych i stosowanych mechanizmów bezpieczeństwa.



Rys. 4. Czas uruchomienia systemów w godzinach

Czas potrzebny na opracowanie systemu autorskiego był 12,5-krotnie większy niż czas uruchomienia systemu open-source (Rys. 4). Poddaje to w wątpliwość opłacalność pisania takiego systemu jedynie na potrzeby jednego wdrożenia, szczególnie w przypadku tak prostej i popularnej funkcjonalności jaka jest założona w niniejszym opracowaniu. Aby system autorski, z punktu widzenia opłacalności biznesowej, miał szanse konkurować z systemem open-source, musi oferować wartość, której system open-source nie posiada. Taką wartością może być ekskluzywny charakter takiego systemu, możliwość pełnej kontroli nad jego strukturą i dostosowanie interfejsu dokładnie do wymagań klienta. W przypadku, kiedy wdrażane byłyby specyficzne, niszowe rozwiązania, czas wdrożenia systemu open-source mógłby znacząco się wydłużyć, ponieważ wymagałoby to napisania własnych modułów do obsługi nietypowego sprzętu czy protokołu oraz dogłębnego poznania struktury systemu Domoticz lub przynajmniej jego API (Application Programming Interface).

Można zatem stwierdzić, że w przypadku typowego zastosowania, system open-source będzie rozwiązaniem zdecydowanie szybszym od systemu autorskiego, jednak przy niszowym specyficznym zastosowaniu można rozważyć opcję budowy systemu autorskiego.

System Domoticz oferuje zdecydowanie większe możliwości skalowania systemu niż system sHome. Jednak jeśli system autorski zostałby opracowany pod kątem konkretnego wdrożenia daje o wiele większą elastyczność i prostotę obsługi, poprzez implementację wyłącznie tych funkcji jakie rzeczywiście w danym wdrożeniu są wykorzystywane. Dojrzały system już na etapie projektowania powinien zakładać trwałość i utrzymanie kodu w czasie. Popularny system open-source jest w tym wypadku znacznie bardziej odporny od systemu autorskiego na starzenie się kodu oraz pojawiające się zagrożenia bezpieczeństwa, szczególnie kiedy system autorski jest projektem tworzonym przez niewielki zespół lub jednego autora. System autorski jest na tyle odporny na starzenie się kodu, na ile jego autor będzie zdecydowany kontynuować projekt. Istnieją co najmniej dwie możliwości uniknięcia sytuacji zamknięcia projektu: pierwsza z nich to upublicznienie kodu i próba stworzenia wokół niego społeczności programistów gotowych dalej go rozwijać, drugą możliwością jest komercjalizacja rozwiązania, poprzez zawarcie stałej umowy pomiędzy potencjalnym klientem, a autorem i zapewnienie ten sposób finansowania dalszych prac nad projektem oraz zatrudnienie do jego rozwoju zespołu deweloperów.

Konkluzją wynikającą z analizy badanych systemów jest stwierdzenie, że budowa systemu autorskiego ma uzasadnienie głównie w projektach służących edukacji i nabywaniu praktycznych umiejętności programowania. Drugą przesłanką przemawiającą za budową autorskiego projektu, jest chęć pełnej kontroli nad kodem aplikacji i dostosowania systemu do bardzo specyficznych indywidualnych warunków wdrożenia oraz wymagań użytkownika. Istnieją jednak dość poważne ograniczenia i zagrożenia wypływające z budowy takiego systemu, wynikające przede wszystkim z konieczności stałej kontroli i aktualizacji oprogramowania, a także dużego ryzyka porzucenia projektu. Zdaniem autora, stając przez wyborem uruchomienia systemu dla typowego rozwiązania, autor byłby skłonny wykorzystać dostępny system open-source, oferujący także wsparcie społeczności, pomocne w przypadku problemów z funkcjonowaniem czy wykryciem podatności na ataki hackerskie. System autorski zaprojektowany z poszanowaniem elementarnych zasad bezpieczeństwa budowy oprogramowania, powinien również zapewniać bezpieczeństwo, ze względu na to, że system niszowy obniża prawdopodobieństwo wykrycia podatności. Będzie on również mniej atrakcyjnym celem dla cyberprzestępców, którzy zazwyczaj wybierają rozwiązania popularne, po to by zmaksymalizować zasięg ataku. System autorski zapewnia również większe poczucie prywatności, szczególnie w porównaniu z systemami chmurowymi, które przesyłają uzyskane dane do chmury obliczeniowej, gdzie są analizowane przez algorytmy sztucznej inteligencji, a użytkownik traci kontrolę nad tym, gdzie i w jaki sposób analizowane są dane z systemu zarządzania automatyką w jego domu. Zatem budowa autorskiego systemu inteligentnego budynku jest uzasadniona szczególnie w przypadku zapewnienia prywatności, dostosowania systemu do specyficznych nietypowych zastosowań oraz do zdobycia wiedzy i doświadczenia w kwestii budowy aplikacji.

Literatura

Pomimo usilnych poszukiwań artykułów naukowych w języku polskim i angielskim, dotyczących problematyki systemów inteligentnych budynków udostępnianych na licencji open-source, autor artykułu nie dotarł do żadnych opracowań tym temacie. Dlatego na podstawie przedstawionych w literaturze artykułów przeprowadził samodzielne badania na temat takich rozwiązań oraz ich popularności. Na potrzeby badań autor artykułu korzystał z następujących pozycji internetowych:

[1] https://pl.wikipedia.org/wiki/Inteligentny_budynek [07.04.2019].

[2] https://cyfrowa.rp.pl/technologie/13104-inteligentny-dom-powoli-zyskuje-popularnosci [02.05.2019].

[3] https://analizarynku.eu/125 [02.05.2019]

[4] https://inwestor.newseria.pl/newsy/rynek_inteligentnych,p1432688098 [02.05.2019].

[5] https://smarthomegearguide.com/5-open-source-home-automation-tools/ [03.05.2019].

[6] https://randomnerdtutorials.com/9-home-automation-open-source-platforms-for-your-projects/ [03.05.2019].

[7] http://24-7-home-security.com/open-source-home-automation-software/ [03.05.2019].

[8] https://zwavezone.com/open-source-home-automation-systems/ [03.05.2019].

[9] https://www.smarthomeblog.net/openhab-home-assistant-domoticz/ [03.05.2019].

[10] https://www.quora.com/What-is-the-best-open-source-home-automation-platform-I-e-Plays-nice-with-other-popular-smart-device-ecosystems-can-be-deployed-on-Linux-etc [03.05.2019].

[11] https://opensource.com/tools/home-automation [03.05.2019].

[12] https://medium.com/@ronnymenestrel/seven-open-source-home-automation-tools-d25c27816f8 [03.05.2019].

[13] https://www.electromaker.io/blog/article/9-best-raspberry-pi-smart-home-software-options [03.05.2019].

[14] https://www.techupyourhome.com/home-security/best-open-source-home-automation-tools/ [03.05.2019].

[15] https://www.owasp.org/images/7/72/OWASP_Top_10-2017_%28en%29.pdf.pdf

[16] https://vuldb.com/pl/?id.132644 [28.05.2019].

[17] https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2019-10664 [28.05.2019].

[18] https://www.letscontrolit.com/wiki/index.php/ESPEasy

[19] https://github.com/domoticz/domoticz/graphs/contributors [28.05.2019].

[20] https://datasheets.maximintegrated.com/en/ds/DS18B20.pdf [07.04.2019].

[21] https://www.openhacks.com/uploadsproductos/rain_sensor_module.pdf [07.04.2019].

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 12 (2019) 240-245
Wysłane: 2019-06-19
Przyjęte: 2019-06-24

Analiza porównawcza rozwiązań wysokiej dostępności

Michał Sylwester Borsewicz*, Daniel Bieniek
Politechnika Lubelska, Instytut Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. Celem niniejszej pracy jest przedstawienie analizy porównawczej wybranych rozwiązań wysokiej dostępności. Problematyka wyboru odpowiednich rozwiązań HA jest nieodzowną częścią każdej firmy korzystającej z infrastruktury IT. Jako, że komercyjnie najczęściej stosowane są rozwiązania wirtualizacyjne firm Microsoft: Hyper-V oraz VMWare: vSphere, to zostały one poddane dogłębniejszej analizie. W tym celu wykonano testy wydajnościowe systemów gości na powyższych hypervisorach oraz przetestowano działanie klastra w sytuacji awarii jednego z hostów.

Słowa kluczowe: klaster; wysoka dostępność; Hyper-V; vSphere

*Autor do korespondencji.

Adres e-mail: borsewicz.michal@gmail.com

Comparative Analysis of High Availability Solutions

Michał Sylwester Borsewicz*, Daniel Bieniek
Institute of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract. The aim of the thesis was to perform a comparative analysis of selected high availability solutions. The issue of choosing the right HA solutions is an indispensable part of any company that uses IT infrastructure. As the most commonly used solutions are the virtualization solutions of Microsoft: Hyper-V and VMWare: vSphere, they have been subjected to in-depth analysis. For this purpose, we have performed performance tests of guest systems on the above hypervisors and we tested the behavior of the cluster in the event of failure of one of the hosts.

Keywords: cluster; high availability; Hyper-V; vSphere

*Corresponding author.

E-mail address: borsewicz.michal@gmail.com

1. Wstęp

Obecnie jednym z najważniejszych problemów, z którym muszą się zmierzać wszystkie firmy wykorzystujące rozwiązania IT, bez względu na ich wielkość, czy typy wykorzystywanych aplikacji jest zapewnienie nieprzerwalnej dostępności do infrastruktury IT. Każdy przestój w działaniu aplikacji ukierunkowanych na obsługę klientów stanowi poważne straty finansowe, a zarazem może być przyczyną utraty zaufania przez klientów. Aby zapobiegać takim sytuacjom, gdy np. awaria serwera może spowodować przestój, stosuje się rozwiązania wysokiej dostępności tzw. High Availability (HA).

Dostępność to stopień, w jakim aplikacje, usługi lub funkcje są dostępne dla użytkowników. Stopień dostępności mierzony jest poprzez odczucia użytkownika aplikacji. Użytkownicy odczuwają frustrację, gdy ich dane są niedostępne lub system komputerowy nie działa zgodnie z oczekiwaniami. Zwykły, szary użytkownik komputera nie rozumie zależności pomiędzy komponentami składającymi się na działanie aplikacji i zazwyczaj wcale go to nie interesuje. Najważniejsze dla niego jest to, aby aplikacja na której musi pracować po prostu działała. W sytuacji, gdy użytkownik nie może uzyskać dostępu do systemu, mówi się, że system jest niedostępny.

Celem niniejszego artykułu jest przedstawienie analizy porównawczej wybranych rozwiązań wysokiej dostępności czyli produktów Hyper-V oraz vSphere.

2. Analiza obecnego stanu wiedzy

Od początku lat 90 tematy wysokiej dostępności były często poruszane na łamach czasopism naukowych. J. Gray i D. Siewiorek w artykule [1], przedstawiają ówczesne techniki stosowane do budowy wysoce dostępnych systemów komputerowych począwszy od początku lat 70. Wtedy to elementy sprzętowe były głównym źródłem usterek i przestojów. Obecnie usterki sprzętu stanowią niewielki procent wszystkich awarii systemu. Teraz najczęstszymi przyczynami są błędy w oprogramowania i błędy środowiska. Dzisiaj na tę pozycję można patrzeć jedynie pod kątem historycznym.

Inną pozycją, która dobrze opisuje działanie i wykorzystanie klastrów, jest publikacja [2] autorstwa K. Kopper. Pomimo, że obecne rozwiązania są inne, to zasady działania pozostały podobne. Publikacja opisuje w głównej mierze działanie klastrów. Ukazuje darmowe rozwiązania

dotyczące wysokiej dostępności. Książka zawiera informacje o tym, jak połączyć parę serwerów aby uzyskać wysoką dostępność za pomocą pakietu Heartbeat, jak korzystać z oprogramowania do równoważenia obciążenia Linux Virtual Server, jak skonfigurować niezawodny system drukowania w środowisku klastra Linux, jak zbudować harmonogram zadań w systemie Linux.

W artykule autorstwa Prashanta K. D. [3], porównane są dwa opensourcowe rozwiązania wirtualizacyjne: hyperwisory Xen, KVM oraz komercyjne VMware vSphere (ESXi), Microsoft Hyper-V. Autor opisuje ich architekturę, sposoby implementacji.

W następnym artykule R. Alroobaea, A. Ghiduk, M. Ikbal [4] przeprowadzono analizę porównawczą wymienionych w tytule hyperwisorów. Porównywano w głównej mierze wpływ hyperwizorów na wydajność. Z przeprowadzonych eksperymentów wynika, że najlepsze wyniki wydajnościowe osiągnął KVM. Jedynie w teście wydajności systemu plików lepszy okazał się XEN. Autorzy stwierdzili, że KVM najlepiej nadaje się do aplikacji wymagających dużej mocy obliczeniowej i pamięci np. aplikacje internetowe. XEN natomiast lepiej nadaje się do aplikacji zależnych od systemu plików np. aplikacje bazodanowe będą lepiej pracować jeśli zostaną wdrożone na hyperwisorze XEN.

Graniszewski i Arciszewski w artykule [5] porównali Hyper-V, ESXi, OVM, VirtualBox i Xen. W swoim artykule sprawdzili wpływ powyższych hyperwizorów na CPU, NIC oraz pamięć RAM. Ich wyniki ukazały, że najlepszym hyperwizorem pod względem wydajności jest ESXi.

W kolejnym ważnym artykule [6] autorstwa A. Bhatia, G. Bhattal przeprowadzono testy wydajnościowe dla hyperwizorów: Vmware ESXI, Xen, Hyper-V i KVM. Autorzy wykorzystali w tym celu narzędzie SIGA oraz Passmark Sofware. W swoim artykule autorzy wykazali, że najlepszym rozwiązaniem pod względem wydajności jest ESXI.

W artykule [7] autorzy H. Fayyad-Kazan, L. Perneel, M. Timmerman przetestowali wydajność wymienionych w tytule hyperwisorów. Z przeprowadzonych badań autorom wyszło, że najlepszym z hyperwizorów jest XEN, gdy ostatnie miejsce zajął ESXi od VMWare.

Jak widać na powyższych przykładach, różni autorzy doszli do różnych konkluzji, jeżeli chodzi o testy wydajnościowe najpopularniejszych hyperwizorów.

Problematyka wyboru odpowiednich rozwiązań HA jest nieodzowną częścią każdej firmy korzystającej z infrastruktury IT. Analiza wybranych rozwiązań może okazać się pomocna dla administratora systemów czy baz danych. Przeprowadzenie analizy porównawczej dwóch najczęściej stosowanych rozwiązań wirtualizacyjnych, czyli VMware ESX 6.0.0 U3 oraz Microsoft Hyper-V 2019 Version 1809 pozwoli na pewno na lepsze zrozumienie ich działania. Poza tym będzie można odpowiedzieć na pytanie, które rozwiązanie wypadło lepiej w testach wydajności oraz czasu reakcji potrzebnego na ponowny dostęp do danych w momencie awarii jednego z hostów.

3. Wstęp

Hyper-V to oprogramowanie stosowane do wirtualizacji fizycznych maszyn, którego autorem jest firma Microsoft. Występuje w dwóch wariantach: jako samodzielny produkt o nazwie Hyper-V Server lub jako doinstalowywany zewnętrzny komponent do systemu Windows Server. MS Hyper-V wymaga procesora ze sprzętową obsługą wirtualizacji. Wymagane jest, aby procesor był zgodny z technologią Intel VT lub AMD-V. Powinien również wspierać technologię SLAT (Second Level Address Translation) oraz DEP (Data Execution Prevention).

Hyper-V obsługuje izolację maszyn wirtualnych wykorzystując partycje. Partycja to logiczna jednostka izolacji obsługiwana przez hyperwisor, w którym uruchamiany jest każdy system operacyjny gościa. Jest to zasadniczo po prostu maszyna wirtualna. Oprogramowanie do wirtualizacji Hyper-V działa w partycji nadrzędnej i jest to jedyna maszyna wirtualna która posiada bezpośredni dostęp do zasobów sprzętowych. Partycja nadrzędna tworzy partycje podrzędne, które obsługują systemy gości. Wszystkie partycje gości, aby uzyskać dostęp do urządzenia muszą przejść przez partycję nadrzędną. Partycja podrzędna nie ma dostępu do procesora. Hyperwisor obsługuje przerwania do procesora i przekierowuje je do odpowiedniej partycji za pomocą SynIC. Hyper-V może przyspieszyć sprzętową translację adresów przestrzeni adresów wirtualnych gości za pomocą translacji adresów drugiego poziomu, nazywanej EPT dla Intel i RVI (dawniej NPT) dla AMD [8]. Partycje podrzędne nie mają dostępu do zasobów sprzętowych, ale mają dostęp do zasobów urządzeń wirtualnych. Każde żądanie wysłane do urządzeń wirtualnych jest przekierowywane przez VMBus do urządzeń w partycji nadrzędnej, która zarządza żądaniami. VMBus umożliwia komunikację między partycjami. Odpowiedź jest również zarządzana przez VMBus. Jeśli urządzenia w partycji nadrzędnej są również urządzeniami wirtualnymi, zostaną przekierowane dalej, aż dotrą do partycji nadrzędnej, gdzie uzyskają dostęp do urządzeń fizycznych. Partycje nadrzędne uruchamiają VSP, który łączy się z VMBus i obsługuje żądania dostępu do urządzeń z partycji podrzędnych [9].

vSphere (ESXi) to hyperwisor typu 1 (tzw. bare metal), czyli jest instalowany bezpośrednio na sprzęcie systemowym. Został opracowany przez VMWare. Nazwa ESX powstała jako skrót od Elastic Sky X. ESXI zawiera i integruje ważne komponenty systemu operacyjnego, takie jak jądro VMKernel. Należy zaznaczyć, że Jądro Linuksa jest uruchamiane jako pierwsze, a następnie jest używane do ładowania innych składników wirtualizacji w tym m.in. VMKernel [10].

Inne najważniejsze produkty VMWare działające z ESXI to:

- vCenter Server - umożliwia monitorowanie i zarządzanie wieloma hostami ESX, ESXi i GSX. Ogólnie rzecz ujmując, Center Server umożliwia zarządzać platformą vSphere.

- vSphere Client - umożliwia monitorowanie i zarządzanie pojedynczą instancją serwera ESX lub ESXi.

4. Skalowalność Microsoft Hyper-V i VMWare vSphere

W tabeli 1 zostały porównane najważniejsze z zasobów dotyczących obydwu rozwiązań: Hyper-V oraz vSphere. Jak widać w pewnych aspektach lepiej wypada produkt VMWare – np. w maksymalnej liczbie procesorów logicznych oraz w liczbie wirtualnych CPU przypadających na hosta, jak i odwrotnie, w innych lepiej wypada produkt Microsoftu – np. w obsłudze maksymalnej pamięci fizycznej hosta, czy w liczbie wirtualnych CPU przypadających na maszynę wirtualną.

Oczywiście podczas wyboru konkretnego hyperwisora nie należy polegać wyłącznie na skalowalności. Ważne są również kwestie dostępnych funkcjonalności, kompatybilności z systemem operacyjnym, czy chociażby dostęp i profesjonalizm pomocy technicznej.

Tabela 1. Skalowalność zasobów Hyper-V 2019 i vSphere 6.7

System	Resources	Microsoft Hyper-V 2019	VMware vSphere 6.7		
			Free Hypervisor	Essential Plus	Enterprise Plus
Host	Logical Processors	512	768	768	768
	Physical Memory	24 TB	4 TB	4 TB	16TB
	Virtual CPUs per Host	2048	4096	4096	4096
	VM per Host	1024	1024	1024	1024
	Nested Hypervisor	Yes	Yes	Yes	Yes
VM	Virtual CPUs per VM	240 for Generation2 64 for Generation1	8	128	128
	Memory per VM	12 TB for Generation2 1 TB for Generation1	6128 GB	6128 GB	6128 GB
	Maximum Virtual Disk	64 TB for VHDX format 2040 GB for VHD format	62 TB	62 TB	62 TB
	Number of disks	256 (SCSI)	256 (SCSI)	256 (SCSI)	256 (SCSI)
Cluster	Maximum Nodes	64	64	64	64
	Maximum VMs	8000	8000	8000	8000

5. Metodyka badań

W celu wykonania dokładnego porównania obu hyperwisorów użyto dwóch serwerów:

HPE MicroServer Gen8 o następujących parametrach:

- Procesor: Intel® Xeon® Processor E3-1265L v2 – 4 rdzenie i 8 wątków o taktowaniu 2,50 GHz i 3,50 GHz w trybie turbo, 8MB pamięci cache L3;
- Pamięć RAM: 12 GB DDR3 1Rx8 PC3L 12800E-11-13-D1; Unbuffered ECC (8GB + 4GB);
- Karty sieciowe: 2 x Ethernet 1 Gbit + 1x port iLO;
- Dysk: 240GB SSD MTL;

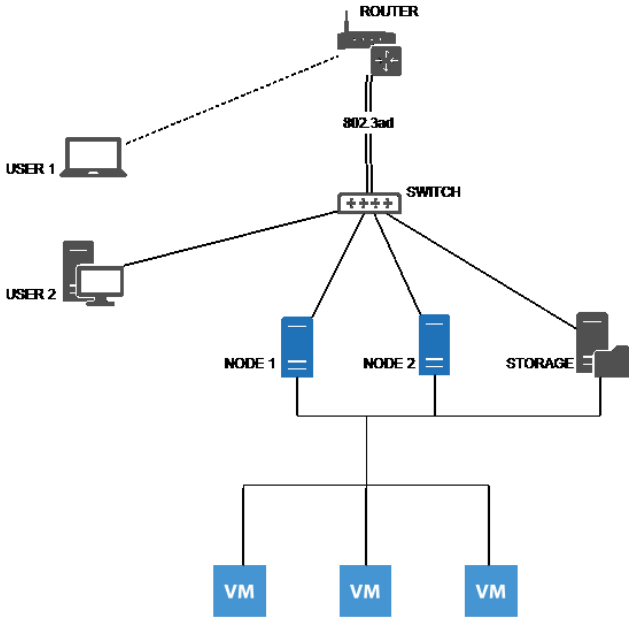
Dysk został wykorzystany jako storage do instalacji poszczególnych nadzorców.

Drugim serwerem został Fujitsu PRIMERGY TX120 S3P, który służył jako iSCSI Target Server:

- Procesor: Intel Celeron G1610T 2.3 GHz- 2 rdzenie i 2 wątki o taktowaniu 2,30 GHz, 2MB pamięci cache L3
- Pamięć RAM: 8 GB DDR3 1Rx8 PC3L 12800E-11-13-D1; Unbuffered ECC (2 x 4GB);
- Karty sieciowe: 2 x Ethernet 1 Gbit + 1x port iRMC;
- Karta RAID:LSI SAS 1064E.

Na dysku SSD została zainstalowana dystrybucja Linuxa odpowiedzialna za udostępnianie dysków po protokole ISCSI. Dyski SAS 2 x 300GB zostały skonfigurowane z RAID0 i przeznaczone dla maszyn wirtualnych, natomiast dysk HDD 500GB pełnił rolę magazynu na obrazy ISO czy pliki instalacyjne programów.

Wszystkie serwery zostały podłączone do przełącznika Ubiquiti ES-24-LITE za pomocą skrętki komputerowej Cat.6A S/FTP 2m. Każdy z serwerów został podłączony 3 takimi przewodami. Jeden służy do zarządzania serwerem a dwie kolejne do wykorzystania przez system zainstalowany na serwerach. Wszystkie elementy platformy testowej działają z przepustowością 1Gbps. Na potrzeby testów została stworzona sieć 10.200.200.0/24 z dostępem do Internetu. Na rysunku 1 przedstawiono schemat środowiska testowego.



Rys. 1. Schemat środowiska testowego

Testowano VMware ESX 6.0.0 U3 oraz Microsoft Hyper-V 2019 Version 1809. Jako goszczonych systemów operacyjnych, pracujących na hypervisorach w czasie testów, używano kombinacji instancji systemów operacyjnych Windows 10 Version 1809 oraz dystrybucji systemu Linux CentOS 7.6.1810. Zastosowano edycje 64-bitowe dla Windows 10 i CentOS 7 oraz VMware i Hyper-V. Windows i Linux instalowano, używając tej samej pamięci i ustawień vCPU dla obu hyperwisorów.

Dla systemu Windows było to:

- Procesor: 2 rdzenie i 2 wątki
- Pamięć RAM: 2 GB
- Karty sieciowe: 1 x 1 Gbit
- Dysk: 50 GB

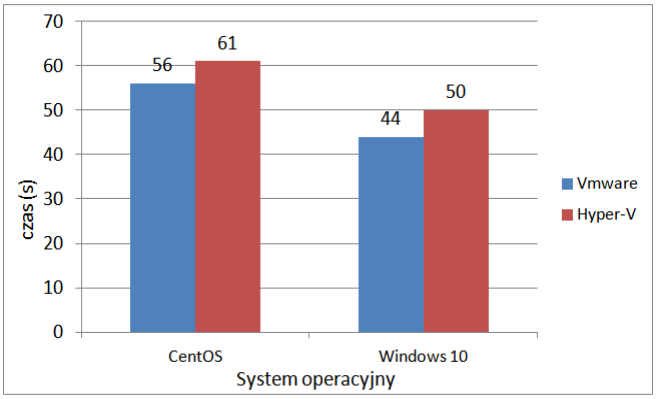
Dla systemu Linux było to:

- Procesor: 1 rdzeń i 1 wątek
- Pamięć RAM: 1 GB
- Karty sieciowe: 1 x 1 Gbit
- Dysk: 16 GB

6. Wyniki badań

6.1. Awaria hosta

Pierwszym etapem badań było zachowanie klastra w sytuacji awarii jednego z węzłów. Symulacja awarii polegała na wyłączeniu zasilania hosta, na którym znajdowały się maszyny wirtualne. W czasie awarii sprawdzany był czas jaki jest potrzeby aby wszystkie maszyny wirtualne były ponownie dostępne. Za pomocą polecenia PING sprawdzano utratę pakietów. Wykonano trzy próby a wyniki uśredniono. Średnia liczba utraconych pakietów dla VMware wyniosła 14 pakietów, a dla Hyper-V 16 pakietów.

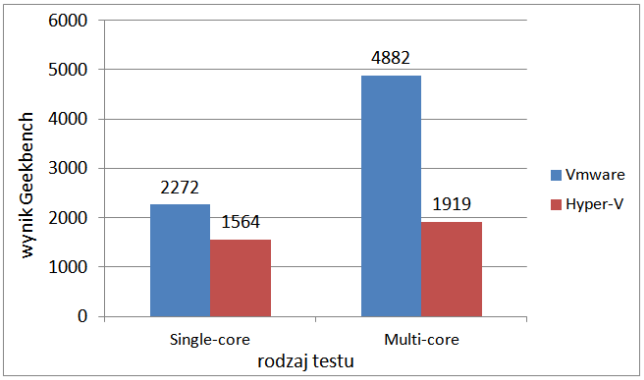


Rys. 2. Średnie czasy po jakich systemy były gotowe do działania w zależności od systemu operacyjnego

Jak obrazuje wykres (Rys. 2), dla obydwu systemów operacyjnych, zarówno na Windows 10, jak i na CentOS 7, lepsze czasy uzyskał hyperwisor VMware. Na systemie Windows hosty gotowe były do ponownej pracy po 44 sekundach, a na systemie CentOS po 56 sekundach. Czasy wymagane do ponownej pracy przy użyciu rozwiązania wirtualizacyjnego firmy Microsoft to 50 sekund dla systemu Windows oraz 61 sekund dla systemu CentOS. Wyraźnie widać, że po awarii hosta na ponowną pracę szybciej pozwala hyperwisor VMware. Różnica względem Hyper-V wynosiła 5 oraz 6 sekund.

6.2. Wydajność

W testach wydajności przebadano prędkość procesora oraz średnie czasy operacji dyskowych oferowanych przez badane rozwiązania. Do testowania pomiarów prędkości procesora na systemie operacyjnym Windows 10 wykorzystano aplikację Geekbench. Aplikacja ta testuje m.in. szybkość zapisu i odczytu danych oraz obsługę strumieni. Sprawdza jak PC radzi sobie podczas szyfrowania, kompresji tekstu i obrazów. Wbudowane w nim benchmarki pozwalają na testowanie procesorów jedno- lub wielordzeniowych w aspektach wielowątkowości i wielordzeniowości. Testy procesora Single-Core oraz Multi-Core szacują, jak szybko procesor jest w stanie wykonać wiele różnych obliczeń. Im szybciej procesor jest w stanie wykonać obliczenia, tym wyższy wynik testu porównawczego. Wyniki testów przedstawiono na rysunku 3.

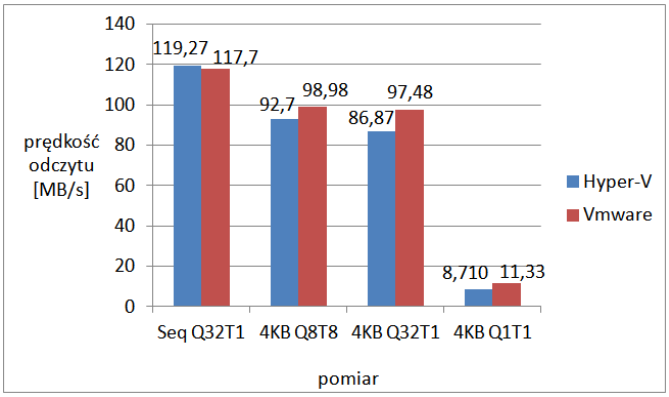


Rys. 3. Wyniki pomiaru prędkości CPU w programie Geekbench dla systemu Windows

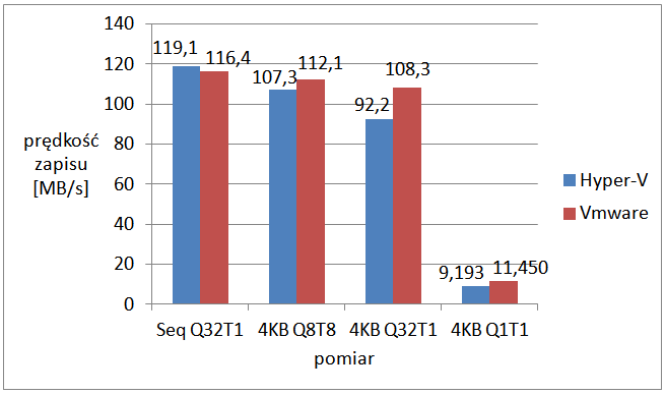
W teście wydajności operacji wejścia/wyjścia dysku wykorzystano aplikację CrystalDiskMark. Jest to niewielki benchmark służący do testowania wydajności dysków twardych, który na bazie wykonanych testów losowych oraz sekwencyjnych operacji zapisu/odczytu danych, potrafi dokonać pomiaru prędkości dysku. Prędkość wyznaczana jest dla 4 pomiarów:

- Seq Q32T1: testuje w trybie wielowątkowości. Plik jest zapisany w segmentach o wielkości 128 KB.
- 4K Q8T8: wielkość segmentów to 4KB, 8 kolejek, wątków 8.
- 4K Q32T1: wielkość segmentów to 4KB, 32 kolejki, wątków 1.
- 4K Q1T1: gdzie, wielkość segmentów to 4KB, 1 kolejka, wątków 1.

Uzyskane wyniki zostały przedstawione na rysunku 4 oraz 5.



Rys. 4. Średnie prędkości odczytu danych z dysku dla obydwu hyperwisorów dla systemu Windows

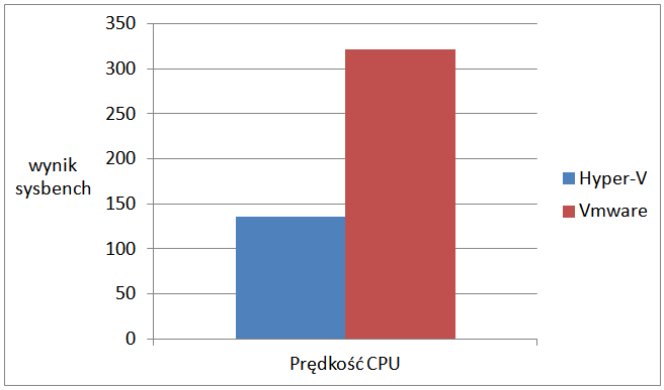


Rys. 5. Średnie prędkości zapisu danych z dysku dla obydwu hyperwisorów dla systemu Windows

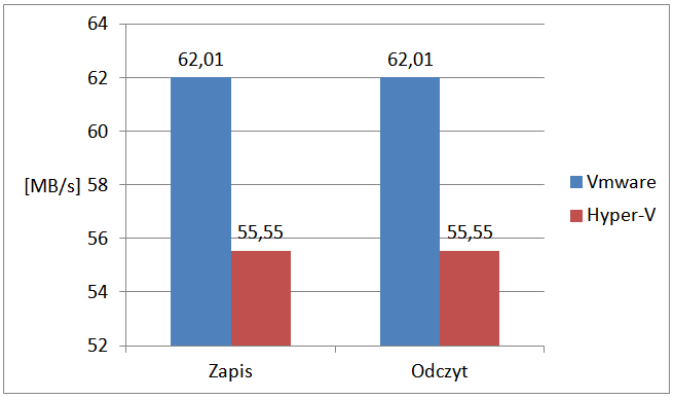
Dla systemu CentOS oba benchmarki zostały wykonane za pomocą programu SysBench. Jest to prosta w obsłudze aplikacja, za pomocą której szybko przetestujemy podstawowe parametry komputera/serwera:

- Procesor (CPU)
- Pamięć RAM
- System plików (transfer I/O)
- Bazę danych (OLTP)

Uzyskane wyniki zostały przedstawione na rysunku 5 i 6.



Rys. 6. Wynik pomiaru prędkości CPU w programie sysbench dla systemu CentOS



Rys. 7. Średnie prędkości odczytu i zapisu danych z dysku dla systemu CentOS wykonane przy użyciu programu sysbench

7. Wnioski

Przeprowadzone badania wykazały, że produkt VMWare – vSphere był lepszy w teście dotyczącym awarii jednego z hostów. Maszyny wirtualne przy użyciu rozwiązania vSphere były szybciej dostępne do ponownej pracy. Różnica względem Hyper-V wynosiła nieco ponad 5 sekund na korzyść vSphere. Różnica niby niewielka, ale w środowisku produkcyjnym, gdzie liczy się każda sekunda niedostępności danego sprzętu, systemu, czy usługi lepszym wyborem będzie VMware.

W testach wydajności goszczonych systemów również lepszym okazał się produkt VMWare.

W przeprowadzonych testach dotyczących goszczonego systemu operacyjnego Windows 10 w teście CPU zdecydowanie lepiej wypadł produkt Vmware gdzie różnica w pomiarze pojedynczego rdzenia wynosiła ponad 30% a w przypadku wielu rdzeni już prawie 40%. Natomiast w teście prędkości zapisu/odczytu plików, Hyper-V poradził sobie jedynie lepiej w pracy z większymi plikami. System Windows goszczony na VMware dużo szybciej pracował z mniejszymi plikami. Z testów wynika, że hyperwisora VMware lepiej używać np. do utrzymania baz danych, gdy Hyper-V powinien lepiej się sprawować w aplikacjach internetowych.

W testach dotyczących goszczonego systemu CentOS 7, w teście prędkości CPU, prawie dwukrotnie lepszy wynik, wynoszący ponad 40%, otrzymał hyperwisor vSphere. Natomiast w teście prędkości zapisu/odczytu, zarówno zapis jak i odczyt odbywa się szybciej przy wykorzystaniu rozwiązania VMware. Średnio te operacje wykonują się o 10% szybciej przy użyciu hyperwisora VMWare, niż w przypadku Hyper-V.

Literatura

[1] Gray J., Siewiorek D.: High-Availability Computer Systems. Carnegie Mellon University, 1991.

[2] Kopper K.: The Linux Enterprise Cluster - Build a Highly Available Cluster with Commodity Hardware and Free Software. No Starch Press, 2005.

[3] Prashanta K. D.: Comparative Study on XEN, KVM, VSphere, and Hyper-V. [W]: Emerging Research Surrounding Power Consumption and Performance Issues in Utility Computing, 2016.

[4] Alroobaea R., Ghiduk A., Ikbal M.: Performance Evaluation of Xen, KVM, and Proxmox Hypervisors. [W]: International Journal of Open Source Software and Processes. 9(2), 2018.

[5] Graniszewski W, Arciszewski A.: Performance analysis of selected hypervisors (Virtual Machine Monitors – VMMs). [W]: International Journal of Electronics and Telecommunications, 62(3), 2016.

[6] Bhatia A., Bhattal G.: A comparative study of Various Hypervisors Performance. [W]: International Journal of Scientific & Engineering Research. Tom 7, zeszyt 12, 2016.

[7] Fayyad-Kazan H., Perneel L., Timmerman M.: Banchmark the Performance of Microsoft Hyper-V server, VMware ESXi and Xen Hypervisors. [W]: Journal of Emerging Trends in Computing and Information Sciences, 2013.

[8] https://www.microsoft.com/en-us/download/details.aspx?id=56495 [01.06.2019].

[9] https://docs.microsoft.com/en-us/virtualization/hyper-v-on-windows/reference/hyper-v-architecture [01.06.2019].

[10] https://blogs.vmware.com/vsphere/2018/04/introducing-vmware-vsphere-6-7.html [01.06.2019].

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 12 (2019) 246-251
Wysłane: 2019-06-18
Przyjęte: 2019-06-26

Analiza rozwoju środowiska uruchomieniowego systemu Android

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

Streszczenie. W artykule przedstawiono rozwój środowiska uruchomieniowego systemu Android. Zaprezentowano własną aplikację dla systemu Android, w której zaimplementowano testy wydajności wykorzystane do przeprowadzenia badań w różnych wersji środowiska. Test jest metodą, która implementuje różne operacji w systemie Android oraz mierzy czas ich wykonania.

Słowa kluczowe: Android; środowisko uruchomieniowe; Dalvik; ART

*Autor do korespondencji.

Adres e-mail: kostiantyn.honcharenko@pollub.edu.pl

Analysis of the development Android's runtime

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

Abstract. The article presents the development of the Android runtime. Own application for Android is presented, which implements performance benchmarks used to test different versions of the Android runtime. The methods measure the benchmark execution times in different versions of the Android runtime environment.

Keywords: Android; run time; Dalvik; ART

*Corresponding author.

E-mail address: kostiantyn.honcharenko@pollub.edu.pl

1. Wstęp

Historia wersji systemu operacyjnego Android rozpoczęła się wraz z wydaniem Androida beta w listopadzie 2007. Pierwsza komercyjna wersja, czyli Android 1.0, została wydana we wrześniu 2008 roku [1]. System ten doczekał się wielu aktualizacji, które krok po kroku modyfikują jego pierwotną wersję. Aktualizacje te są wydawane najczęściej w celu poprawiania błędów, zmian wizualnych, dodawania nowych funkcji i zwiększania ogólnej wydajności systemu.

Zaczynając od wersji 1.0 do wersji 4.3 Android wykorzystywał maszynę wirtualną Dalvik, jako domyślnie środowisko, w którym działały wszystkie aplikacje Android. W wersji 4.4 po raz pierwszy pojawiła się alternatywa – ART (Android Runtime). Użytkownik mógł własnoręcznie, w ustawieniach telefonu, zmienić środowisko wykonawcze z Dalvik na ART i odwrotnie. Twórcy Androida skupili się na wymianie kompilatora JIT (just-in-time) na AOT (ahead-of-time) i zapewniali, że zmiana kompilatora przyspieszy pracę urządzenia. Już od wersji 5.0, całkowicie zrezygnowano z maszyny Dalvik i na wszystkich urządzeniach środowisko ART było stosowane domyślnie.

2. System Android

System Android oparty jest na jądrze Linuksa. Jądro zarządza zasobami smartfona, w tym dostępem do sprzętu, pamięcią stałą, uruchamianiem, zatrzymywaniem i migracją procesów między rdzeniami procesora jak i wielozadaniowością. Struktura systemu jest widoczna na rysunku 1. Jak w przypadku każdego innego systemu, jądro — to serce systemu Android [3].

W najniższej warstwie systemu Android znajduje się jądro systemu operacyjnego. Zapewnia ono funkcjonowanie systemu i jest odpowiedzialne za bezpieczeństwo, zarządzanie pamięcią, zużyciem energii, procesami, a także stosem sieciowym i sterownikami. Jądro jest najważniejszą częścią systemu operacyjnego Linux, w przeciwieństwie do innych jego części, zostało ono przeniesione do systemu Android prawie bez zmian [4].

Warstwa abstrakcji sprzętowej (Hardware Abstarction Layer) udostępnia standardowe interfejsy, które mapują możliwości urządzeń sprzętowych na wyższy poziom interfejsu Java API. HAL składa się z wielu modułów biblioteki, z których każdy implementuje interfejs dla określonego typu składnika sprzętowego.

Powyżej w kolejnej warstwie systemu operacyjnego, znajduje się zestaw bibliotek (Native C/C++ Libraries), przeznaczony do rozwiązywania typowych zadań wymagających wysokiej wydajności. Ta warstwa jest odpowiedzialna za:

- Zapewnienie realizowanych algorytmów dla kolejnych warstw;
- Wsparcie formatów plików;
- Realizację kodowania i dekodowania informacji (na przykład, multimedialne kodeki);

- Renderowanie grafiki i wiele więcej.

Biblioteki są zaimplementowane w języku C/C++ i skompilowane dla konkretnej architektury urządzenia, wraz z którym są dostarczane przez producenta [5].

Środowisko uruchomieniowe Android (ART) pozwala uruchomić wiele wirtualnych maszyn na urządzeniach o małej pamięci, wykonując pliki DEX, format bajtowy zaprojektowany specjalnie dla Androida, zoptymalizowany pod kątem wykorzystania pamięci.

Rys. 1. Struktura systemu Android [3]

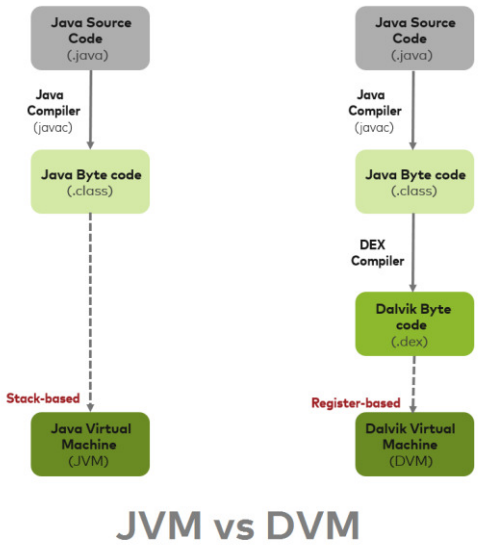
Cały zestaw funkcji systemu Android jest dostępny dla użytkowników za pośrednictwem interfejsów API napisanych w języku Java. Te interfejsy API tworzą bloki potrzebne do tworzenia aplikacji na Androida, upraszczając ponowne wykorzystanie podstawowych, modułowych komponentów systemu i usług.

Na szczycie w architekturze systemu Android znajduje się warstwa aplikacji (Applications). W skład tej warstwy wchodzą: przeglądarka www, klient poczty elektronicznej, program do wysyłania SMSów, mapy, kalendarz, menedżer kontaktów i wiele innych. Oprócz tego podstawowego zestawu do poziomu aplikacji należą wszystkie aplikacje dla platformy Android, w tym te zainstalowane przez użytkownika [3].

2.1. Dalvik

Dalvik Virtual Machine jest częścią platformy mobilnej Android. Jest to maszyna wirtualna, której autorem jest Dan Bronstein. Kod źródłowy jest rozpowszechniany jako wolne oprogramowanie na warunkach zgodnej z BSD licencji Apache 2.0. W dużej mierze fakt ten odegrał swoją rolę w decyzji Google aby zrezygnować z JME (Java Micro Edition) [6], na którą należało uzyskać licencję od firmy Sun.

Dalvik wykorzystuje własny kod bajtowy. Aplikacje dla Androida są tłumaczone przez kompilator do postaci specjalnego niezależnego od maszyny niskopoziomowego kodu. Podczas uruchamiania kodu na platformie Dalvik, ona interpretuje i wykonuje program [7]. Ponadto, Dalvik jest w stanie przetłumaczyć kod bajtowy Java do kodu we własnym formacie, a także wykonywać go w swoim środowisku wirtualnym. Przykłady przetwarzania kodu maszyn wirtualnych Java oraz Dalvik przedstawiono na rysunku 2. Kod aplikacji tworzy się w języku Java, a po kompilacji pliki .class mogą być konwertowane do formatu .dex (nadające się do interpretacji w maszynie Dalvik) za pomocą compilatora DEX, wchodzącego w skład SDK systemu Android.



Rys. 2. Przetwarzania kodu JVM a Dalvik VM [8]

Dalvik został zaprojektowany specjalnie dla platformy Android. Uwzględniono fakt, że platforma uruchamia wszystkie procesy w izolacji, każdy w swojej przestrzeni adresowej. Maszyna wirtualna jest zoptymalizowana pod kątem niskiego zużycia pamięci i pracy na mobilnym sprzęcie. Począwszy od wersji systemu Android 2.2, Dalvik, używa kompilacji JIT (Just-in-Time) [7].

2.2. ART

ART to środowisko uruchomieniowe aplikacji Android, które zostało opracowane przez Google, jako zamiennik maszyny Dalvik. Główną różnicą jest zmiana kompilatora JIT na kompilator AOT [9].

Podczas instalacji oprogramowania aplikacji, kod bajtowy dex kompiluje się do kodu maszynowego (kompilacja AOT), podczas gdy Dalvik kompiluje kod bajtowy do dex (Dalvik executable) a po uruchomieniu programu do kodu maszynowego w czasie rzeczywistym (kompilacja JIT). Z tego względu ART przyczynia się do oszczędzania energii. Co prawda, odbywa się to kosztem zwiększenia ilości używanego miejsca i spowolnienia instalacji aplikacji. Google twierdzi, że spowolnienie nie jest krytyczne. Innowacje w rodzaju ART stały się możliwe dzięki zwiększeniu ilości pamięci w nowoczesnych smartfonach [1].

Kompilator AOT również przegrywa z kompilatorem JIT pod względem możliwości optymalizacji kodu maszynowego. Po prostu nie ma wystarczająco dużo informacji na temat zachowania aplikacji i specyfiki jej pracy. Można było ją uzyskać, tylko uruchamiając aplikację. Dodatkowo kompilator AOT znacznie zwalniał instalację aplikacji i pierwsze uruchomienie systemu operacyjnego [10].

W czerwcu 2014 roku firma Google poinformowała o obsłudze 64-bitowej architektury w wersji Android L. Do zalet architektury 64-bitowej wymienionych w informacji firmy Google zaliczają się: zwiększona liczba rejestrów, nowe zestawy instrukcji i rozszerzone adresowane miejsca w pamięci, co pozwala do adresowania większej puli adresów [11].

Jeśli aplikacja jest napisana w języku Java, kod automatycznie skorzysta z zalet nowej 64-bitowej architektury. Firma Google zaktualizowała NDK do wersji 10b i dodała obraz emulatora, który można wykorzystać do przygotowania aplikacji do pracy na urządzeniach z 64-bitowymi procesorami Intel. Jest to bardzo wygodne do tworzenia aplikacji dla użądzeń z 64-bitową architekturą [11].

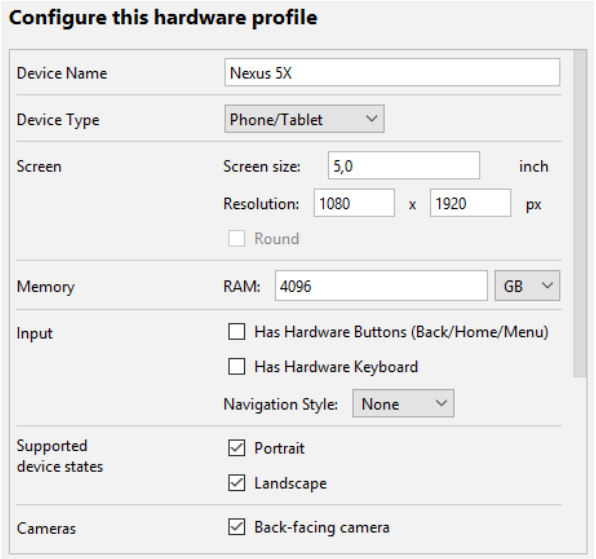
W systemie Android N dodano dynamiczny kompilator JIT z profilowaniem kodu dla środowiska ART, który pozwala stale podnosić wydajność aplikacji Android w trakcie ich pracy. Kompilator JIT uzupełnia bieżący kompilator AOT i pomaga zwiększyć wydajność, zmniejszyć zużycie pamięci, a także przyspieszyć aktualizacje systemu i aplikacji [12].

Jedną z najbardziej wymiernych korzyści kompilatora JIT w ART jest szybkość instalacji aplikacji i aktualizacji systemu. Nawet duże aplikacje, które wymagają kilku minut w celu ich optymalizacji i instalacji na Androidzie 6.0, w aktualnych wersjach systemu Android mogą być zainstalowane w ciągu kilku sekund [12]. Aktualizacje systemu również są wykonywane szybciej, ponieważ etap optymalizacji teraz jest nieobecny.

W Android Pie została wprowadzona optymalizacja profili ART w Play Cloud - nowa funkcja optymalizacji, która znacznie poprawia czas uruchamiania aplikacji po nowej instalacji lub aktualizacji. Średnio aplikacje uruchamiają się o 15% szybciej [13] (zimny start) na różnych urządzeniach.

3. Aplikacja badawcza

Dla uruchomienia aplikacji badawczej wydajności środowiska Androida od wersji 4.1 do 9.0, zdecydowano się wykorzystać emulator Android Studio. Dla emulatora wybrano profil urządzenia Nexus 5X, który przedstawiony na rysunku 3. Emulator ma ekran 5 cali z rozdzielczością 1080x1920 pikseli i RAM o ilości 4096 MB.



Rys. 3. Główny widok aplikacji

Testy wykonano z pomocą aplikacji stworzonej na Androida z wykorzystaniem Android SDK. Aplikacja jest kompatybilna ze wszystkimi wersjami Androida, licząc od Androida 4.1. Do testowania środowiska uruchomieniowego od wersji Android 4.1 do wersji Android 9.0 zostały przygotowane następujące scenariusze testowe:

- Szyfrowanie tekstu algorytmem AES;
- Sortowanie tablicy typu long;
- Operacji z tablicą.

Strukturę części kodu, który bada wydajność środowiska, pokazano na listingu 1. Po kliknięciu przyciska rozpoczęcia testu, aplikacja włącza stoper i zaczyna wykonywać scenariusz testowy. Po poprawnym zakończeniu testu stoper wyłącza się, rezultat wyświetla się na ekranie emulatora i się resetuje.

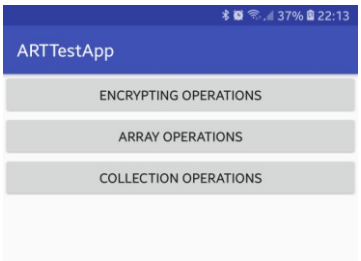
Przykład 1. Metoda badająca wydajności

```
start.setOnClickListener(view -> {
    try {
        stopwatch.start();
        // scenariusz testowy
        stopwatch.stop();
        resultText.setText(stopwatch.toString());
        stopwatch.reset();
    } catch (Exception e) {
        e.printStackTrace();
    }
});
```

Aplikacja składa się z 3 tematycznych klas dziedziczących po Activity, z których każda uwzględnia przypadek testowy. Rysunek 4 przedstawia główny widok programu, w którym można wybrać rodzaj testów.

Do zapisu dużej ilości danych, na przykład dla zachowania tablicy o wielkości 10 000 elementów, wykorzystano package assets, który jest używany dla

przechowywania plików z informacją, które będą przetwarzane i używane w przypadku potrzeby. Ponieważ maszyna wirtualna Java ogranicza długość nazw pól i metod, opisów pól i metod, a także innych stałych wartości ciągów do 65535 znaków [14].



Rys. 4. Główny widok aplikacji

4. Wyniki testów

W ramach badań wykonano kilka rodzajów testów reprezentujących elementarne operacje wykonywane w aplikacjach. Każdy test był powtarzany 25 razy. Opisano je poniżej.

4.1 Szyfrowanie tekstu

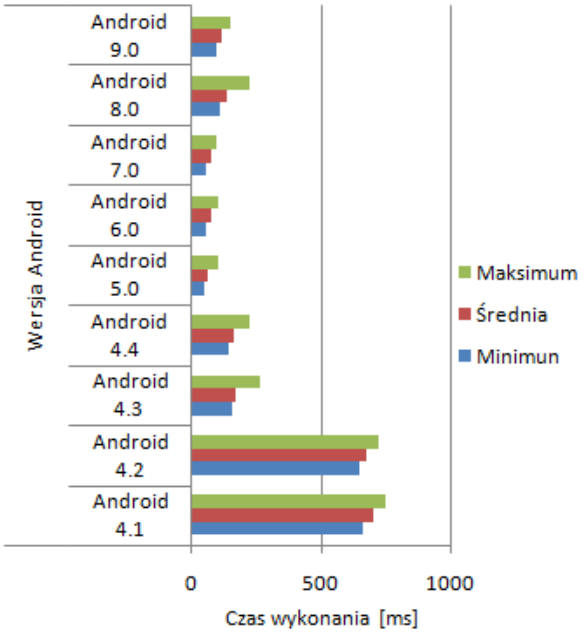
W trakcie testu najpierw tworzona jest zmienna String o długości 100 000 znaków, po czym, z pomocą symetrycznego szyfru AES bajty są szyfrowane kluczem o rozmiarze 192 bit. Następnie uzyskana tablica jest deszyfrowana i otrzymywane są dane początkowe.

Tabela 1. Wyniki testu szyfrowania tekstu

		Minimun	Średnia	Maksimum
Czas wykonania [ms]	Android 4.1	663	707,2	752
	Android 4.2	651	676,52	728
	Android 4.3	159	173,44	271
	Android 4.4	150	170,92	229
	Android 5.0	56	66,44	109
	Android 6.0	63	78,72	107
	Android 7.0	61	80,32	99
	Android 8.0	112	142,4	228
	Android 9.0	104	120,52	155

Przejście z Dalvik na nowe środowisko uruchomieniowe dało znaczny wzrost wydajności (Rysunek 4, tabela 1). Najszybciej z zadaniem poradził sobie emulator z systemem Android w wersji 5.0. Jednak zaczynając od wersji 5.0 do wersji 9.0, czas wykonywania testu wzrósł.

Najwolniejszym systemem Android w wersji ze środowiskiem ART okazała się wersja 8.0, która ma maksymalny czas wykonywania testu zbliżony do wskaźników wersji 4.3 i 4.4. Rezultaty testu zamieszczono w tabeli 1.



Rys. 4. Wykres wartości minimalnej, medialny oraz maksymalnej dla testu szyfrowanie tekstu

4.2 Operacje na kolekcjach

Test polega na tworzeniu nowej kolekcji ArrayList z tablicy typu long z liczbą elementów wynoszącą 10 000 w zakresie od 0 do 9223372036854775807. Tablica została wstępnie załadowana z zasobów assets. Po tym kolekcja jest sortowana i wykonywane jest ponowne zapisanie elementów. Dalej do utworzonej, nowej kolekcji skopiowano zawartość poprzedniej za pomocą klasy Collections. W realizacji tego testu użyto klasy Collections, która zawiera zestaw przydatnych metod pomagających wykonywać różne czynności z kolekcjami. Kod źródłowy opisanego testu przedstawiono na listingu 2.

Przykład 2. Operacji z kolekcją

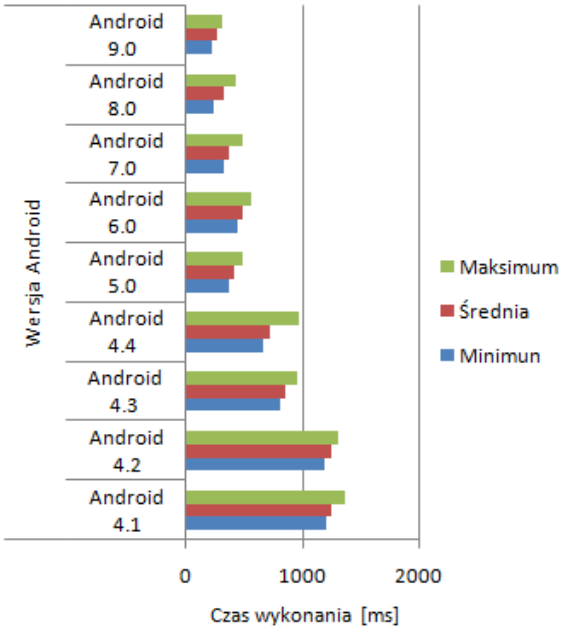
```
private void collectionOperation (long[] longArrTmp) {
    ArrayList<Long> longColl = new ArrayList<>();
    for (Long l : longArrTmp) {
        longColl.add(l);
    }
    Collections.sort(longColl);
    Collections.reverse(longColl);
    ArrayList<Long> newLongColl = new ArrayList<>();
    Collections.copy(longColl, newLongColl);
}
```

Rezultaty wykonania zaprezentowano w tabeli 2 i na rysunku 6. Otrzymano następujące wyniki - od wersji 4.1 do wersji 9.0 następowała stała poprawa wydajności.

Porównując początkową i najnowszą wersję wydajność środowiska wzrosła więcej niż 4 razy. Skok wydajności widoczny jest po przejściu na wersję 5.0. Średni czas wykonania zmniejszył się z 723 ms do 407 ms. Od wersji 6.0 do wersji 9.0 czas realizacji zadania stopniowo się zmniejszał.

Tabela 2. Wyniki testu operacje na kolekcjach

		Minimun	Średnia	Maksimum
Czas wykonania [ms]	Android 4.1	1206	1251	1366
	Android 4.2	1194	1246,64	1312
	Android 4.3	802	853,96	951
	Android 4.4	660	723,16	965
	Android 5.0	372	407,08	488
	Android 6.0	440	491,88	555
	Android 7.0	320	369,76	491
	Android 8.0	240	330,04	423
	Android 9.0	215	260,88	308



Rys. 6. Wykres wartości minimalnej, medialny oraz maksymalnej dla testu operacje na kolekcjach

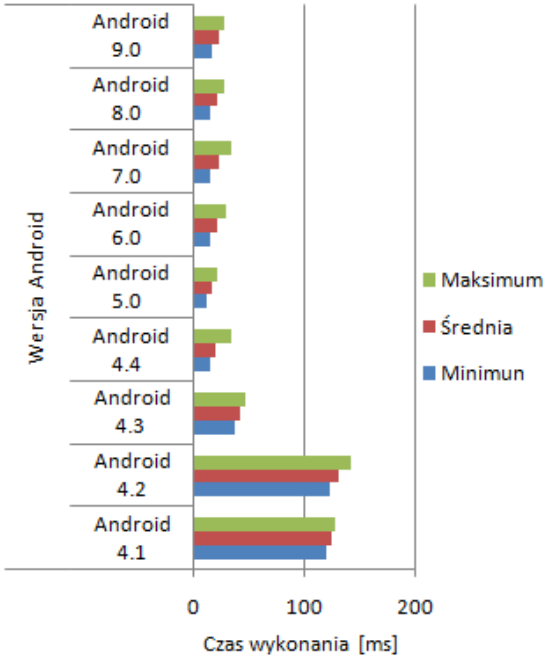
4.3 Sortowanie tablicy liczb całkowitych

Metoda badawcza sortuje tablicę elementów typu long, która ma w sobie 10 000 elementów. Typ danych long przechowuje liczbę, którą można umieścić w 64 bitach. Zakres liczby jest od -9223372036854775808 do 9223372036854775807. Wszystkie elementy dla tablicy zostały wygenerowane losowo w zakresie od 0 do 9223372036854775807.

Najbardziej zauważalna różnica jest między wynikami wykonania testu w wersji 4.2 i 4.3. Jednak w przeciwieństwie do poprzednich wyników już nie ma stopniowego wzrostu wydajności w każdej kolejnej wersji. Z wyników należy, że najszybszy emulator jest działający na wersji na Androida 5.0. Przyczyną tego wyniku może być dodanie w systemie Android wsparcia dla 64-bitowej architektury w wersji 5.0. Szczegółowe wyniki testu przedstawiono w tabeli 3 i na rysunku 7.

Tabela 3. Wyniki testu operacji na tablicy elementów long

		Minimun	Średnia	Maksimum
Czas wykonania [ms]	Android 4.1	120	123,8	128
	Android 4.2	123	130,04	142
	Android 4.3	38	42,04	48
	Android 4.4	16	20,6	35
	Android 5.0	13	17,2	22
	Android 6.0	16	22,12	30
	Android 7.0	16	24,12	34
	Android 8.0	16	21,32	28
	Android 9.0	18	23,56	28

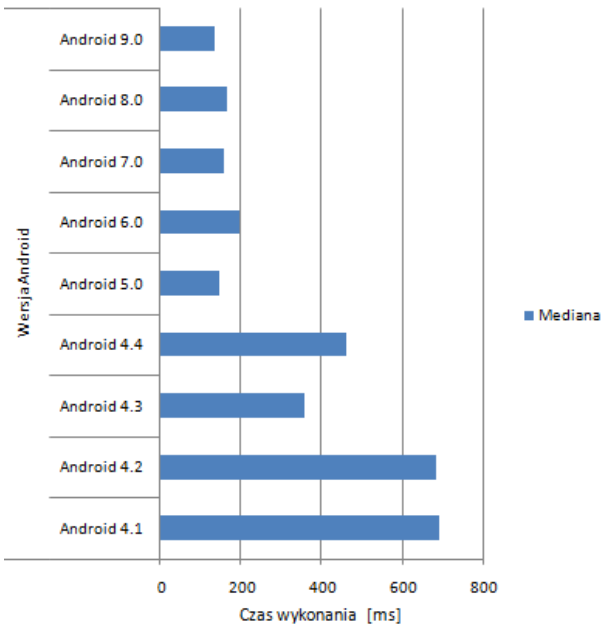


Rys. 7. Wykres wartości minimalnej, medialny oraz maksymalnej dla testu sortowanie tablicy liczb całkowitych

5. Wnioski

Na podstawie uzyskanych wyników można zauważyć, że w trakcie testowania różnych wersji Android była zauważalna ogólna tendencja, która widoczna na rysunku 8. Mediana pokazuje średni czasu wykonania trzech przypadków testowych w każdej wersji Androida. Z tego wynika, że najszybszymi wersjami były 5.0 i 9.0.

Warto zauważyć, że w piątej wersji system Android zaczął używać środowiska ART jako środowiska domyślnego. Najwolniejsze wersje są 4.1, 4.2 i 4.4. Główną przyczyną może być to, że oni działają w starym środowisku Dalvik.



Rys. 8. Wykres wartości medialny dla wszystkich operacji

Największa różnica w wydajności istnieje pomiędzy wersjami 4.4 i 5.0. Jest to związane z następującymi czynnikami: przejście z wersji testowej ART na produkcyjną, pełnej rezygnacji z maszyny Dalvik, wsparcia dla 64-bitowej architektury. Kolejne wersje nie przyniosły znaczącej zmiany.

Badania pokazało, że wydajność środowiska uruchomieniowego Android nie rosła w każdej nowej wersji w przygotowanych scenariuszach testowych. To może być związane z konfiguracją emulatora i go stałymi charakterystykami. Ponieważ w trakcie rozwinięcia systemu Android wydajność urządzeń mobilnych rośnie w każdym roku. Na przykład telefon Nexus 4 z wersją Androida 4.1 pokazuje wynik o ilości 27 300 punktów w teście wydajności Antutu, podczas gdy Pixel 3XL z wersją 9.0 dostaje 275 000 punktów [15]. Dlatego system Android nie musi mieć więcej wydajności w każdej nowej wersji, jednak jak widać z wyników ona znacznie wzrosła od wersji systemu 4.1 do 9.0 .

Literatura

[1] A. Frumusanu: A Closer Look at Android RunTime (ART) in Android L: AnandTech, 2014.

[2] C. Stewart , B. Phillips , K. Marsicano: Programowanie aplikacji dla Androida The Big Nerd Ranch Guide: Helion, 2017.

[3] https://tech-geek.ru/how-android-works, How does the operating system Android, Linux core and Android Runtime [04.2019].

[4] D. A. Heger: Mobile Devices – An Introduction to the Android Operating Environment, Design, Architecture, and Performance Implications: DHTechnologies (DHT), 2012.

[5] R. Meier: Professional Android 4 Application Development: John Wiley & Sons, 2012.

[6] https://studopedia.su/12_142429_Java-mashina-Dalvik.html, Dalvik virtual machine structure, core features, standart libraries [02.2019].

[7] B. Cheng; B.Buzbee:. A JIT Compiler for Androids Dalvik VM: Google, 2010.

[8] https://android.jlelse.eu/closer-look-at-android-runtime-dvm-vs-art-1dc5240c3924, Closer look at Android Runtime, comparing DVM vs ART, execution Java code in Android [03.2019].

[9] https://wikipedia.org/wiki/Android_Runtime, Android Runtime structure, advantages and disadvantages, compilators [05.2019].

[10] https://xakep.ru/2018/01/10/android-5-core-techs, The five pillars of Android, virtual machine, Google services, Linux core and runtime [03.2019].

[11] https://software.intel.com/ru-ru/android/articles/64-bit-android-and -android-run-time, 64-bit versions of Android and Android runtime, development and support 64-bit processors for Android [03.2019].

[12] https://developer.android.com/about/versions/nougat/android-7.0?hl=ru, Android N for developers, new features, changing JIT compilator for perfomance [03.2019].

[13] https://android-developers.googleblog.com/2019/04/improving-app-performance-with-art.html, Improving app performance with ART optimizing profiles in the cloud, instruction for implementing [04.2019].

[14] https://docs.oracle.com/javase/specs/jvms/se7/html/jvms-4.html#jvms-4.1, Chapter describes the Java Virtual Machine class file format, limits for name pool and values [03.2019].

[15] https://www.kimovil.ru, Performance tests for smartphones from old versions to current.

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 12 (2019) 252-257
Wysłane: 2019-06-04
Przyjęte: 2019-06-26

Analiza i ocena narzędzi wspomagających pracę grupową w chmurze

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

Streszczenie. Celem niniejszej pracy, była analiza i ocena narzędzi wykorzystywanych do pracy grupowej w chmurze oraz na podstawie przyjętych kryteriów, wybranie najlepszego narzędzia. Wybór programów ukierunkowany był na użytkowników jakimi są studenci oraz pracownicy małych firm. W pracy wykorzystano analizę literatury oraz metody badawcze. Praca złożona jest z rozdziałów, w których uwagę skupiono się na zastosowaniu chmury obliczeniowej, objaśnieniu modeli i rodzajów chmur, ogólnym przedstawieniu wybranych narzędzi, przedstawieniu analizy problemu i metod badawczych, oraz prezentacji dokonywanych badań i wniosków z nich wynikających.

Słowa kluczowe: praca grupowa w chmurze; chmura; narzędzia

*Autor do korespondencji.

Adres e-mail: pawel.j.gustaw@gmail.com

Analysis and evaluation of tools supporting group work in the cloud

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

Abstract. The aim of this work was to analyze and evaluate the tools used for group work in the cloud and based on the adopted criteria, choose the best tool. The choice of programs was targeted at users such as students and small business employees. The work uses literature analysis and research methods. The work consists of chapters in which attention is focused on the use of cloud computing, explanation of models and types of clouds, general presentation of selected tools, presentation of the analysis of the problem and research methods, and presentation of research and conclusions resulting from them.

Keywords: cloud collaboration; cloud; tools

*Corresponding author.

E-mail address/addresses: pawel.j.gustaw@gmail.com

1. Wstęp

W dzisiejszych czasach zdolność do pracy grupowej często jest wartością, którą cenią sobie w pracowniku przedsiębiorstwa poszukujące nowych osób. Opłaca się więc, rozwijać w sobie tę umiejętność, która w przyszłości może procentować. Okazji do sprawdzania się pod tym względem w codziennym funkcjonowaniu jest wiele, chociażby podczas nauki we wcześniejszych jej etapach czy podczas studiów, gdzie zadań w grupie nad różnego rodzaju projektami jest o wiele więcej. Praca grupowa charakteryzuje się koniecznością komunikacji pomiędzy uczestnikami grupy. Komunikacja w pracy grupowej to podstawa, bez której efekt końcowy podjętego projektu bądź zadania może skończyć się niepowodzeniem. Płynna wymiana informacji oraz sprawne porozumiewanie się ma oddziaływanie na panującą atmosferę w zespole oraz efektywne działanie.

W dobie Internetu, dostęp do sieci jest obecnie czymś naturalnym i praktycznie osiągalnym w cywilizowanych miejscach na całym świecie. To właśnie przy pomocy Internetu można nawiązać komunikację z drugą osobą wykorzystując narzędzia informatyczne, które także mogą posłużyć do pracy grupowej. Aktualnie osiągalnych jest szereg narzędzi i oprogramowania do wspierania komunikacji w grupie, takich jak chociażby sprzęt komputerowy i mobilny, wiadomości email, komunikatory, portale społecznościowe, fora, wideokonferencje czy specjalnie dedykowane platformy. Praca zespołowa online to cały czas rozwijający się sposób przetwarzania w chmurze obliczeniowej, służący do komunikacji, i współpracy pomiędzy członkami zespołu. Postęp technologii sprawił, że korzystając z dowolnego narzędzia do współpracy oraz pracując w grupie, członkowie mają możliwość wymiany pomiędzy sobą różnego rodzaju plików, komentowania, czatowania czy edycji jednego dokumentu przez kilka osób, gdzie wprowadzane zmiany na dokumencie są widoczne w czasie rzeczywistym.

W obliczu dużej ilości ogólnodostępnych narzędzi do pracy grupowej w chmurze, rodzi się pytanie, jakie narzędzie będzie spełniało większość wymagań i będzie najlepszym wyborem do pracy zespołowej. Zapewne nie ma narzędzi idealnych, tak i te, które zostały wybrane w celu przeprowadzenia analizy a następnie oceny posiadają plusy i minusy w swych funkcjonalnościach. Wybrane narzędzia do badań są skierowane do studentów oraz pracowników małych firm.

2. Zastosowanie chmury obliczeniowej w pracy grupowej

Praca grupowa jest bardzo wszechstronnym określeniem, które zawiera współpracę w różnych formach. W dzisiejszych czasach, zaawansowanych technologicznie dla wielu osób, studentów czy pracowników firm, których praca opiera się na korzystaniu ze sprzętu komputerowego, mobilnego i Internetu oznacza komunikację online, dzięki której możliwa jest praca nad danym zagadnieniem z różnych miejsc świata, miasta czy chociażby budynku firmy. Praca grupowa w chmurze jest to

rodzaj współpracy, która ułatwia użytkownikom zespołową pracę nad różnego rodzaju typami plików, które magazynowane są kontach i dyskach platform chmurowych. Członkowie grup używają oprogramowania opartego na chmurze do tworzenia, edytowania czy udostępniania plików, i zespołowej pracy nad projektami. Intencja współpracy w chmurze ma swój początek, kiedy użytkownik stworzy lub załączy dokument czy plik, a w dalszym ciągu udostępnia go z możliwością nadania mu określonych praw dla wybranych członków zespołu. Użytkownicy mogą edytować treść w dokumencie w jakimkolwiek czasie, również wtedy, gdy członkowie pracują w tej samej chwili [1] [2].

Modele chmur

IaaS

Infrastruktura jako usługa (IaaS - *Infrastructure as a Service*). IaaS można określić jako najbardziej wszechstronny rodzaj usług w chmurze. Praktycznie oferują kompletnie zwirtualizowaną infrastrukturę obliczeniową, która przy pomocy Internetu jest administrowana i dostarczana. Dostawcy tego modelu chmury zarządzają fizycznym aspektem infrastruktury takim jak serwer, dostępne miejsce czy magazynowanie danych, natomiast klienci mają możliwość dopasować wymagania do własnych potrzeb [3].

PaaS

Platforma jako usługa (PaaS - *Platform as a Service*). Model IaaS zapewnia sprzęt i oprogramowanie w chmurze do zbudowania infrastruktury przez użytkownika według postawionych wymagań. Model PaaS różni się od modelu IaaS tym, że jest bardziej ukierunkowany. W miejsce wolnej infrastruktury, model ten oferuje podstawy potrzebne do wytwarzania, testów, wdrożeń, administracji i aktualizacji aplikacji. PaaS stosuje taką samą główną infrastrukturę co model IaaS, jednak zawiera jeszcze systemy operacyjne, oprogramowanie, narzędzia programistyczne czy systemy baz danych konieczne do wytwarzania oprogramowania [3].

SaaS

Oprogramowanie jako usługa (SaaS - *Software as a Service*). Dla osób korzystających z platform chmurowych, model SaaS może być najbardziej kojarzoną formą przetwarzania online. SaaS można określić jako całkowicie rozwinięte oprogramowanie, z którego przy pomocy Internetu można używać w sposób darmowy lub po zakupie subskrypcji. Dostawca modelu SaaS administruje całą infrastrukturą oraz oprogramowaniem niezbędnym do funkcjonowania aplikacji, przy czym zapewnia użytkowników o możliwości korzystania z aplikacji w dowolnym miejscu i czasie [3].

Rodzaje chmur

Chmura prywatna

Chmurę prywatną można określić jako usługi obliczeniowe oferowane przez Internet lub prywatną sieć wewnętrzną. Chmura prywatna jest wykorzystywana przez przedsiębiorstwa do tworzenia osobistych centrali danych i administrowania nimi w celu zaspokojenia pewnych potrzeb informatycznych czy interesów biznesowych. Chmura prywatna gwarantuje większy nadzór nad skalowaniem, konfiguracją i elastycznością, tym samym pomaga uefektywniać bezpieczeństwo kapitału i ruchów biznesowych [4].

Chmura publiczna

Chmura publiczna jest modelem przetwarzania w chmurze, za pomocą którego użytkownicy dostają dostęp do dużych zasobów mocy obliczeniowej przy pomocy Internetu. Jedną z ważnych korzyści jest dostępność szybkiego skalowania. Dostawcy chmury obliczeniowej posiadają potężną moc obliczeniową, którą rozdysponowują pomiędzy rzeszę swoich klientów. Rodzaj chmury publicznej jest najczęściej wybieraną usługą w technologii przetwarzania w chmurze [4].

Chmura hybrydowa

Rodzaj chmury hybrydowej jest połączeniem chmury prywatnej z publiczną. Struktura chmury hybrydowej daje możliwość przedsiębiorstwom na korzystanie z chmury publicznej w razie konieczności ze względu na prostą migrację obciążenia. Firmy mogą użytkować chmurę publiczną do otwierania programów o dużej pojemności, jak np. wiadomości email i używać chmury prywatnej do ważnych zasobów, jak kopie i odzyskiwanie danych, budżet firmy, czy w czasie konserwacji [4].

3. Cel i metodyka badań

Głównym celem badań jest przeprowadzenie analizy narzędzi wspomagających pracę grupową w chmurze – Google Drive, Dropbox, OneDrive, Slack, Asana i Trello, oraz ocenienie, które z badanych narzędzi najlepiej spełni przyjęte założenia oraz kryteria według których będą testowane, czyli które z narzędzi będzie najlepszym rozwiązaniem dla studentów oraz pracowników małych firm.

Wybrane narzędzi do pracy grupowej w chmurze

Narzędzia analizowane w niniejszej pracy zostały wyszukane przy pomocy Internetu, na podstawie opisów i zestawień narzędzi służących do pracy grupowej w chmurze. Wybrane narzędzia zostały wytypowane na podstawie własnych doświadczeń z tego typu programami oraz ocen i doświadczeń innych użytkowników dzielących się opinią ale także przy pomocy artykułów i recenzji.

1) Google Drive [5].
2) Dropbox [6].
3) OneDrive [7].
4) Slack [8].
5) Asana [9].
6) Trello [10].

Pytania badawcze:

W niniejszej pracy przygotowano następujące pytania badawcze.

- Które z narzędzi w podstawowej wersji bezpłatnej zapewnia najwięcej funkcjonalności?
- Które z narzędzi oferuje najlepszy sposób komunikacji z innymi użytkownikami lub członkami zespołu?
- Które z narzędzi zapewnia współpracę w czasie rzeczywistym?

- Które z narzędzi najlepiej funkcjonuje na urządzeniu mobilnym?

Hipoteza:

W uzyskaniu dokładniejszej odpowiedzi na postawione pytania badawcze, została przedstawiona następująca hipoteza: *Narzędzie Google Drive stanowi najlepsze rozwiązanie do pracy grupowej dla studentów oraz pracowników małych firm.*

4. Metody przeprowadzenia badań

W niniejszej pracy dyplomowej zostały wykorzystane dwie metody badawcze, czyli analiza porównawcza oraz eksperyment naukowy. W badaniach zastosowano praktyczną analizę opartą na kryteriach:

1) Sposób w jaki odbywa się komunikacja pomiędzy użytkownikami
2) Sposób udostępniania plików
3) Współpraca w czasie rzeczywistym
4) Działanie narzędzia na urządzeniu mobilnym
5) Przyjazność interfejsu
6) Ocena podstawowego pakietu i dostępu offline oraz opis pozostałych pakietów

Eksperyment naukowy jako metodę badawczą, wykorzystano do praktycznego przeanalizowania narzędzi i ich funkcjonalności, wykorzystywanych do pracy grupowej w chmurze. Wyniki tej metody, na podstawie przyjętych kryteriów poddano subiektywnej ocenie. Eksperyment naukowy opierał się na wykorzystaniu dwóch różnych kont dla każdego z badanych narzędzi, w celu symulacji pracy grupowej. Badanie głównie skupiało się na kryteriach, które według subiektywnej oceny autora niniejszej pracy, stanowią podstawowe elementy narzędzia do pracy grupowej w chmurze. Zadania jakie zostały zaplanowane i wykonane podczas eksperymentu naukowego, opierały się na przyjętych kryteriach. Dla każdego z narzędzi, dla których zostały utworzone dwa konta użytkownika wykonano następujące zadania:

- Nawiązanie komunikacji pomiędzy kontami oraz sprawdzenie możliwości konwersacji,
- Udostępnianie plików pomiędzy kontami oraz ich odbiór wraz z różnymi nadanymi prawami dostępu,
- Wysłanie zaproszenia lub udostępnienie dokumentu do współpracy oraz obserwacja i testowanie możliwości jednoczesnej pracy na tym samym dokumencie,
- Instalacja wersji mobilnych narzędzi na urządzeniu mobilnym wraz z wykonaniem tych samych zadań,
- Ogólne poruszanie się po narzędziu w celu oceny interfejsu,
- Testowanie dostępu offline.

Wszystkie kryteria były oceniane w zakresie 0 – 3 punktów. (Punkty: 0 – źle, 1 – przeciętnie, 2 – dobrze, 3 – bardzo dobrze). Kryterium oceniające wykorzystywane podstawowe bezpłatne pakiety każdego z narzędzi, było oceniane w sposób subiektywny oraz obiektywny. Pozostałe kryteria oceniane były w sposób subiektywny.

5. Rezultaty badań badań

Opis przebiegu badań

Na samym początku badania niezbędnym było utworzenie konta dla wszystkich analizowanych narzędzi przeznaczonych dla studentów i pracowników małych firm, które zostały wybrane i opisane w niniejszej pracy, oraz pobranie i instalacja narzędzi w przypadku wersji instalacyjnej na urządzenia desktopowe oraz urządzenia mobilne. W przypadku kryteriów głównie odnoszących się do współpracy, konieczne było utworzenie jeszcze jednego konta, a następnie połączyć je w celu przeprowadzenia praktycznego badania, dzięki któremu możliwym byłoby uzyskanie lepszej odpowiedzi na postawione kryteria i hipotezy.

Komunikację w narzędziu Dysk Google oceniono na 2 punkty. W narzędziu jest dostępny czat, aktywny przy danym dokumencie, a nie osobny komunikator dostępny także dla osób spoza wybranego dokumentu.

Udostępnianie oceniono na 3 punkty. Google Drive posiada dużo możliwości dotyczących udostępniania danych, co umożliwia użytkownikowi większą kontrolę nad danymi do których daje dostęp innym użytkownikom.

Współpracę w czasie rzeczywistym w narzędziu Google Drive oceniono na 3 punkty. Użytkownicy mają możliwość współpracy w tym samym czasie na różnego rodzaju edytorach, dostępnych do utworzenia przez Dysk Google.

Działanie narzędzia na urządzeniu mobilnym oceniono na 3 punkty. Narzędzie dostępne jest poprzez aplikację mobilną lub przeglądarkę. Obie te opcje oferują te same funkcjonalności co na wersji desktopowej.

Narzędzie w całości dostępne jest w języku polskim. Wygląd narzędzia jest bardzo czytelny, przyjazny oraz przejrzysty i oceniono go na 3 punkty.

Korzyści z bezpłatnego planu, oraz pozostałe pakiety oceniono na 3 punkty. Użytkownik w podstawowym planie otrzymuje dużą ilość przestrzeni dyskowej oraz niezbędne funkcjonalności do pracy grupowej w chmurze. Narzędzie posiada aplikację mobilną.

Wyniki uzyskane przez narzędzie Dysk Google przedstawia tabela (Tabela 1).

Tabela 1. Wyniki oceny narzędzia Dysku Google

Google Drive	
Kryterium:	**Ocena [pkt]:**
1. Sposób w jaki odbywa się komunikacja pomiędzy użytkownikami	2
2.Sposób udostępniania plików	3
3.Współpraca w czasie rzeczywistym	3
4.Działanie narzędzia na urządzeniu mobilnym	3
5. Przyjazność interfejsu	3
6. Ocena podstawowego pakietu i dostępu offline oraz opis pozostałych pakietów	3
Suma punktów	**17**

Komunikację w narzędziu Dropbox oceniono na 3 punkty. Zintegrowana platforma umożliwia bardzo dobrą komunikację tekstową, w której możliwe jest także umieszczanie grafik, mediów, kalendarzy czy zadań.

Udostępnianie oceniono na 3 punkty. Dostępne opcje dają możliwość kontroli nad danymi oraz to, komu są one udostępniane.

Współpracę w czasie rzeczywistym oceniono na 3 punkty. Platforma umożliwia dodawanie członków oraz płynną wymianę tekstową lub innych danych.

Dostęp do najważniejszych funkcji osiągalny jest poprzez widoczne i intuicyjne przyciski. Wygląd aplikacji jest przyjazny i czytelny. Działanie Dropbox na urządzeniu mobilnym oceniono na 3 punkty.

Narzędzie nie posiada skomplikowanego interfejsu co wpływa na łatwiejsze korzystanie. Narzędzie dostępne jest w polskiej wersji językowej. Na 2 punkty oceniono przyjazność narzędzia, które jest bardzo czytelne.

Korzystanie z darmowego pakietu w narzędziu Dropbox oceniono na 2 punkty. W podstawowym pakiecie użytkownik otrzymuje mało miejsca na dane, a różnica pomiędzy oferowaną przestrzenią dyskową następnego pakietu jest bardzo duża. Jednak poza małą ilością miejscem w chmurze, Dropbox umożliwia zastosowanie tego narzędzia do pracy grupowej w chmurze. Dropbox posiada aplikację mobilną na urządzenia z systemem Android i iOS. Dostęp offline możliwy jest tylko z aplikacji mobilnej.

Wyniki uzyskane przez narzędzie Dropbox przedstawia tabela (Tabela 2).

Tabela 2. Wyniki oceny narzędzia Dropbox

Dropbox	
Kryterium:	**Ocena [pkt]:**
1. Sposób w jaki odbywa się komunikacja pomiędzy użytkownikami	3
2.Sposób udostępniania plików	3
3.Współpraca w czasie rzeczywistym	3
4.Działanie narzędzia na urządzeniu mobilnym	3
5. Przyjazność interfejsu	2
6. Ocena podstawowego pakietu i dostępu offline oraz opis pozostałych pakietów	2
Suma punktów	**16**

Komunikację w narzędziu OneDrive oceniono na 3 punkty. Dzięki zapisanym kontaktom, po rozwinięciu listy użytkownik widzi, które osoby są w danym czasie dostępne w razie nawiązania kontaktu.

Udostępnianie oceniono na 3 punkty. Możliwość nadawania praw oraz możliwość wyboru, czy osoba musi posiadać konto Microsoft dają pewność, że osoba bez takiego konta nie będzie posiadać dostępu do danych.

Dostępne są różne edytory do współpracy lecz ich liczba jest mniejsza niż w przypadku Dysku Google. Współpracę w czasie rzeczywistym oceniono na 2 punkty.

Działanie na urządzeniu mobilnym oceniono na 3 punkty. Użytkownik dzięki aplikacji może tworzyć nowe dokumenty oraz w bardzo prosty sposób je udostępniać, podając adres email i prawa jakie ta osoba będzie posiadać to tego pliku.

Główna cześć interfejsu przeznaczona jest na wyświetlenie danych znajdujących się na dysku w chmurze. Przyjazność interfejsu oceniono na 2 punkty.

Korzyści z darmowej wersji OneDrive oceniono na 3 punkty. Narzędzie OneDrive umożliwia korzystanie z aplikacji mobilnej na systemach Android i iOS. Narzędzie posiada dostęp offline, który możliwy jest tylko z aplikacji mobilnej.

Wyniki uzyskane przez narzędzie OneDrive przedstawia tabela (Tabela 3).

Tabela 3. Wyniki oceny narzędzia OneDrive

OneDrive	
Kryterium:	**Ocena [pkt]:**
1. Sposób w jaki odbywa się komunikacja pomiędzy użytkownikami	3
2.Sposób udostępniania plików	3
3.Współpraca w czasie rzeczywistym	2
4.Działanie narzędzia na urządzeniu mobilnym	3
5. Przyjazność interfejsu	2
6. Ocena podstawowego pakietu i dostępu offline oraz opis pozostałych pakietów	3
Suma punktów	**16**

Komunikację w narzędziu Slack oceniono na 3 punkty. Jako narzędzie przeznaczone do grupowej komunikacji spełnia oczekiwania stawiane przed tego typu programami.

Udostępnianie plików w narzędziu Slack oceniono na 1 punkt. Przy zainstalowanej wtyczce Google, która umożliwia udostępnianie plików z własnego dysku Google opcja dzielenia danych jest szersza. Jednak bez takiej wtyczki udostępnianie możliwe jest z urządzenia na którym użytkownik pracuje, do osób lub kanałów zawartych w narzędziu.

Dzięki integracji z Google współpracę oceniono na 2 punkty.

Korzystanie z aplikacji Slack przeznaczonej do komunikacji grupowej jest bardzo przyjemne. Przestrzeń konwersacji wypełnia prawie w całości ekran, dzięki czemu reszta opcji i paneli nie rozprasza i nie powoduje przypadkowych uruchomień i przejść do innych funkcji. Slack na urządzenia mobilne otrzymuje 3 punkty.

Przyjazność interfejsu oceniono na 3 punkty dzięki nieskomplikowanej strukturze narzędzia, oraz łatwości korzystania z dostępnych funkcjonalności, mimo braku polskiej wersji językowej.

Korzystanie z bezpłatnego planu oceniono na 2 punkty. W bardzo prosty sposób narzędzie umożliwia tworzenie kanałów oraz dodawanie do nich wybranych osób, co pozwala na dobrą organizację pracy. Narzędzie posiada aplikację mobilną na systemy Android i iOS. Dostęp offline możliwy jest z aplikacji mobilnej.

Wyniki uzyskane przez narzędzie Slack przedstawia tabela (Tabela 4).

Tabela 4. Wyniki oceny narzędzia Slack

Slack	
Kryterium:	**Ocena [pkt]:**
1. Sposób w jaki odbywa się komunikacja pomiędzy użytkownikami	3
2.Sposób udostępniania plików	1
3.Współpraca w czasie rzeczywistym	2
4.Działanie narzędzia na urządzeniu mobilnym	3
5. Przyjazność interfejsu	3
6. Ocena podstawowego pakietu i dostępu offline oraz opis pozostałych pakietów	2
Suma punktów	**14**

Komunikację w narzędziu Asana oceniono na 2 punkty. Użytkownicy mają możliwość konwersacji lecz tylko za pomocą projektu.

Udostępnianie oceniono na 1 punkt. Narzędzie posiada wybór dysków chmurowych jako źródła danych, lecz udostępnianie odbywa się jedynie wewnątrz projektu.

Współpracę oceniono na 1 punkt. Narzędzie nie umożliwia jednoczesnej pracy kilku użytkowników na jednym dokumencie.

Próba uruchomienia narzędzia Asana przez przeglądarkę internetową na urządzeniu mobilnym powoduje przekierowanie na aplikację mobilną i jej uruchomienie. Działanie na urządzeniu mobilnym oceniono na 2 punkty.

Ogólne korzystanie z narzędzia nie powinno sprawiać problemów podczas pracy. Narzędzie nie posiada polskiej wersji językowej. Przyjazność interfejsu narzędzia oceniono na 2 punkty.

Asana otrzymuje 2 punkty za możliwość darmowego pakietu. Darmowa wersja w porównaniu do płatnych pakietów, dostarcza tylko podstawowe funkcje dla początkujących. Asana posiada aplikację mobilną na systemy Android i iOS. Dostęp offline możliwy jest tylko z aplikacji na urządzenia mobilne.

Wyniki uzyskane przez narzędzie Asana przedstawia tabela (Tabela 5).

Tabela 5. Wyniki oceny narzędzia Asana

Asana	
Kryterium:	**Ocena [pkt]:**
1. Sposób w jaki odbywa się komunikacja pomiędzy użytkownikami	2
2.Sposób udostępniania plików	1
3.Współpraca w czasie rzeczywistym	1
4.Działanie narzędzia na urządzeniu mobilnym	2
5. Przyjazność interfejsu	2
6. Ocena podstawowego pakietu i dostępu offline oraz opis pozostałych pakietów	2
Suma punktów	**10**

Komunikację w narzędziu Trello oceniono na 1 punkt. Użytkownicy mają możliwość porozumiewania się w komentarzach lecz nie mają możliwości osobnej konwersacji z wybranym członkiem zespołu.

Udostępnianie oceniono na 1 punkt. Możliwość przesyłania plików dostępna jest jedynie w kartach.

Trello nie oferuje w swoich funkcjonalnościach narzędzia, który byłby dostępny do jednoczesnej pracy grupie użytkowników. Za to kryterium narzędzie otrzymuje 1 punkt.

Aplikacja Trello udostępnia te same funkcje co wersja przeglądarkowa, oraz posiada bardzo przydatną funkcję offline, która może być istotna dla osób nie mających możliwości stałego dostępu do Internetu. Działanie na urządzeniu mobilnym oceniono na 2 punkty.

Przyjazność interfejsu oceniono na 2 punkty. Jest on bardzo prosty oraz czytelny. Dodatkowo narzędzie dostępne jest w polskiej wersji językowej.

Korzyści z darmowego pakietu oceniono na 1 punkt. Trello posiada aplikację mobilną na Androida i iOS. Dostęp offline umożliwia jedynie aplikacja mobilna.

Wyniki uzyskane przez narzędzie Trello przedstawia tabela (Tabela 6).

Tabela 6. Wyniki oceny narzędzia Trello

Trello	
Kryterium:	**Ocena [pkt]:**
1. Sposób w jaki odbywa się komunikacja pomiędzy użytkownikami	1
2.Sposób udostępniania plików	1
3.Współpraca w czasie rzeczywistym	1
4.Działanie narzędzia na urządzeniu mobilnym	2
5. Przyjazność interfejsu	2
6. Ocena podstawowego pakietu i dostępu offline oraz opis pozostałych pakietów	1
Suma punktów	**8**

6. Wnioski

Rezultatem zastosowanych metod badawczych czyli przeprowadzonej analizy porównawczej badanych narzędzi do pracy grupowej w chmurze, oraz eksperymentu naukowego polegającego na praktycznym przeanalizowaniu narzędzi kierując się przedstawionymi kryteriami, zostały uzyskane rezultaty przedstawione w tabeli (Tabela 7).

Tabela 7. Wyniki oceny wszystkich narzędzi

Narzędzie / Punkty	**Dysk Google**	**Dropbox**	**OneDrive**	**Slack**	**Asana**	**Trello**
Suma punktów [pkt]/Maksymalna liczba punktów	**17/18**	**16/18**	**16/18**	**14/18**	**10/18**	**8/18**

Celem niniejszej pracy była analiza i ocena narzędzi wspomagających pracę grupową w chmurze, przeznaczonych dla studentów oraz pracowników małych firm, oraz w efekcie wybranie najlepszego narzędzia spełniającego przyjęte kryteria. Analiza badanych narzędzi udowodniła, że wszystkie aplikacje spełniają przyjęte kryteria. Analiza badanych narzędzi udowodniła, że wszystkie aplikacje spełniają w większym lub mniejszym stopniu oczekiwania i wymagania dla tego typu programów. Postawiona w pracy hipoteza: *Narzędzie Dysk Google stanowi najlepsze rozwiązanie do pracy grupowej dla studentów oraz pracowników małych firm* okazała się prawdziwa. Na podstawie eksperymentu naukowego, analizy porównawczej oraz przyjętych kryteriów oceny narzędzi, narzędzie Dysk Google według obiektywnej i subiektywnej oceny, posiada w testowanym bezpłatnym pakiecie funkcjonalności, które umożliwiają docelowym grupom osób, czyli studentom oraz pracownikom małych firm wykorzystywanie ich do prac grupowych. Każde z kryterium posiadało taką samą wagę. Według subiektywnej opinii autora niniejszej pracy, przyjęte kryteria opisują podstawowe funkcjonalności dla programów przeznaczonych do pracy grupowej w chmurze. Dostępność narzędzi do pracy grupowej w chmurze jest duża. Wybór odpowiedniego narzędzia powinien być uzasadniony zakresem prac, jakie w danym zespole będą wykonywane.

Literatura:

[1] Jason R. Rich, Working in the Cloud: Using Web-Based Applications and Tools to Collaborate Online, Que, 2017.
[2] Erik van Ommeren, Sander Duivestein , John deVadoss, Clemens Reijnen, Erik Gunvaldson, Collaboration in the Cloud - How Cross-Boundary Collaboration Is Transforming Business, Sogeti, 2009.
[3] Michael J. Kavis, Architecting the Cloud: Design Decisions for Cloud Computing Service Models (SaaS, PaaS, and IaaS), Wiley, 2014.
[4] Ric Messier, Collaboration with Cloud Computing, Social Media, and Unified Communications, Syngress, 2014.

Źródła internetowe:

[5] https://www.google.com/drive/
[6] https://www.dropbox.com/pl/
[7] https://onedrive.live.com/about/pl-pl/
[8] https://slack.com/intl/en-pl/?eu_nc=1
[9] https://asana.com/
[10] https://trello.com/

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 12 (2019) 258-266
Wysłane: 2019-06-20
Przyjęte: 2019-06-28

Analiza możliwości wykorzystania technologii IoT w systemach inteligentnych domów

Arkadiusz Bęben*, Piotr Kopniak
Politechnika Lubelska, Instytut Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. Wiele urządzeń wyposażenia domowego dostępnych na rynku, jest wyposażona w podzespoły umożliwiające odbieranie i generowanie danych oraz moduły komunikacji pozwalające na połączenie z Internetem. Dzięki temu urządzenia mogą być łączone w systemy inteligentnych domów. Analiza przedstawia poszczególne możliwości urządzeń, które mogą pracować w takich systemach. Przedstawione są urządzenia dostępne na rynku oraz wykonany system inteligentnego domu pracujący w sieci Wi-Fi oraz Z-Wave.

Słowa kluczowe: Internet of Things; inteligentny dom; inteligentne rzeczy

*Autor do korespondencji.

Adres e-mail: arekbeben@gmail.com

Analysis of the application possibilities of IoT technology in smart home systems

Arkadiusz Bęben*, Piotr Kopniak
[a] Institute of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract. Many home appliances available on the market are equipped with components to receive and generate data and communication modules allow connection to the Internet. As a result, the devices can be combined to smart homes systems. The analysis presents the individual capabilities of devices that can work in such systems. Presented are devices available on the market and made a smart home system working in Wi-Fi and Z-Wave networks.

Keywords: Internet of Things; smart home; smart things

*Corresponding author.

E-mail address:arekbeben@gmail.com

1. Wstęp

W ostatnich latach postęp technologiczny umożliwił wprowadzenie na rynek urządzeń wyposażonych w różnego rodzaju czujniki oraz moduły sieciowe pozwalające na komunikację z innymi urządzeniami, a w następstwie podłączenie ich do Internetu. Konsekwencją takiej sytuacji jest rosnąca liczba inteligentnych urządzeń w pomieszczeniach mieszkalnych. Obecnie coraz więcej wykonywanych czynności domowych, jest wspierane poprzez zastosowanie w nich urządzeń posiadających połączenie internetowe.

1.1. Internet of Things

Internet rzeczy (ang. IoT – Internet of Things) jest koncepcją informatyczną, powstałą w wyniku potrzeby określenia urządzeń codziennego użytku, które potrafią generować, przesyłać lub przechowywać dane, bez konieczności ingerencji użytkownika. Cechą charakterystyczną przedmiotów przynależnych do tej idei, jest podłączenie do sieci komputerowej. Oczywiście nie wyklucza to wykorzystania innych sposobów komunikacji, zarówno z użytkownikiem końcowym, jak i z innymi urządzeniami wchodzącymi w skład poszczególnych instalacji. Główną zaletą takiego rozwiązania jest możliwość śledzenia przez użytkownika bieżącego stanu oraz pomiarów czynionych przez takie urządzenia, a także wydawania poleceń mających wpływ na obiekty, które są pod kontrolą tych urządzeń.

Celem wyposażenia pomieszczeń w przedmioty korzystające z rozwiązań zawartych w omawianej koncepcji, jest usprawnienie codziennych czynności, wykonywanych przez osoby przebywające w zasięgu ich działania. Ustalenie takiego priorytetu IoT powoduje, że urządzenia wykorzystujące daną ideę obejmują bardzo szerokie spektrum obszarów życia. Wpływ technologii wchodzących w skład pojęcia Internetu rzeczy jest coraz częściej zauważalny i rozpoczyna swoje działanie już od systemów wspierających wykonywanie codziennych obowiązków w mieszkaniach i gospodarstwach domowych. Kolejne miasta wprowadzają urządzenia IoT do systemów miejskiego monitoringu, czy wczesnego wykrywania zagrożeń [5]. Wielkie przedsiębiorstwa odpowiedzialne za produkcję energii elektrycznej, wspierają działania odpowiedniego rozprowadzania energii, poprzez zastosowanie urządzeń IoT monitorujących bieżące zużycie energii w danych regionach. Czynności handlowe, związane z logistyką, czy magazynowaniem towarów, również są obsługiwane przez szereg urządzeń podłączonych do sieci [6].

1.2. Inteligentny domy

Rozwój technologii w ostatnich latach przyczynił się znacząco do modyfikacji podejścia w kwestii projektowaniu budynków. Duży wpływ na to miał spadek cen urządzeń elektronicznych i rozwój kolejnych sposobów komunikacji i przesyłania danych. Dzięki temu, iż kolejne urządzenia gospodarstwa domowego, zaczęły być wyposażane w obsługę nowych rozwiązań technologicznych, takich jak wszelkiego rodzaju czujniki, przełączniki, czy moduły umożliwiające podłączenie do sieci telekomunikacyjnej; zaczęto projektować specjalne systemy, w obrębie których wszelkie podmioty, mogą być obsługiwane z poziomu jednego urządzenia sterującego. w przypadku urządzeń IoT takim punktem sterującym najczęściej jest smartfon lub tablet lokatora budynku. Możliwości jakie dają nowoczesne urządzenia zaczęto uwzględniać już na poziomie projektowania budynków i traktować jako integralną część nowoczesnego budownictwa. Takie podejście do projektowania budynków przybrało określenie inteligentnego domu (ang. smart home) [1].

Dzięki implementacji jednego z dostępnych na rynku systemów inteligentnych domów, użytkownik dostaje kontrolę nad wieloma urządzeniami, z których każde może być odpowiedzialne za obsługę odrębnych czynności. Wyposażenie kolejnych przedmiotów gospodarstwa domowego w obsługę nowych technologii sprawiło, że dobrze znane nam urządzenia otrzymały nowe funkcjonalności. To z kolei przełożyło się na większe spersonalizowanie obecnego stanu pomieszczenia (poprzez np. ustawienie temperatury lub poziomu oświetlenia), a w dalszej kolejności na wzrost komfortu jego lokatora. Jednak zastosowanie urządzeń IoT ma również inne pozytywne skutki. Wyposażenie budynku w urządzenia monitorujące jego stan, takie jak przykładowo czujniki dymu, wody, otwarcia okien i drzwi, wpływają bezpośrednio na wzrost bezpieczeństwa mieszkańców, a w połączeniu z modułem pozwalającym zaalarmować odpowiednie służby (policja, straż pożarna) skracają czas reakcji w sytuacji zagrożenia zdrowia i życia lokatorów.

Nie bez znaczenia jest również kwestia wykorzystania energii elektrycznej przez użytkowników danego pomieszczenia. Dzięki zautomatyzowaniu pracy urządzeń i instalacji elektrycznych, prąd zużywany jest tylko wtedy gdy rzeczywiście jest niezbędny do wykonywania domowych czynności. Szczególnie ważnym aspektem jest zarządzanie oświetleniem. System dzięki czujnikom ruchu oraz natężenia światła, potrafi utrzymać prawidłowe oświetlenie pomieszczenia, a w przypadku gdy nie wykryje żadnych osób lub naturalne światło okaże się wystarczające – wyłączy oświetlenie zasilane energią elektryczną. Pomocne również są harmonogramy, w których użytkownik może określić, kiedy w domu nikogo nie ma (np. podczas czasu pracy poza domem), dzięki czemu system wyłącza niepotrzebne urządzenia elektryczne. Takie działania powodują zmniejszenie zużycia energii, co finalnie prowadzi do oszczędności przy płaceniu rachunków.

2. Sposoby komunikacji urządzeń IoT

Urządzenia IoT wykorzystują różne sposoby komunikacji bezprzewodowej. Obecnie dostępne są protokoły przygotowane specjalnie z myślą o urządzeniach przesyłających małe ilości danych w krótkich odstępach czasu. Systemy inteligentnych domów opierają swoje działanie najczęściej o konkretny protokół przesyłania danych pomiędzy połączonymi podmiotami. Cechą charakterystyczną koncepcji IoT jest to, że systemy wykorzystujące to rozwiązanie składają się głównie z wielu bezprzewodowych sieci czujnikowych WSN (ang. Wireless sensor network) i urządzeń korzystających z połączeń radiowych RFID (ang. Radio-frequency identification). Paradygmat WSN definiuje sieć złożoną z wielu czujników potrafiących komunikować się ze sobą poprzez łączność radiową. Urządzenia RFID składają się przeważnie z układu scalonego z pewnymi możliwościami obliczeniowymi i anteny służącej do komunikacji [3].

2.1. Technologie połączeń

Podstawowym sposobem komunikacji urządzeń IoT z Internetem jest przeważnie łączność Wi-Fi. Jest to spowodowane przede wszystkim popularnością technologii i faktem, że w przypadku jej wykorzystania, urządzeniem dostępowym do Internetu może być domowy router. Jednak nie jest to najbardziej optymalny sposób komunikacji dla urządzeń IoT, gdyż charakteryzuje się dużą szybkością przesyłania danych, która nie jest konieczna w urządzeniach generujących je w małych ilościach. Ponadto moduł Wi-Fi posiada największe zapotrzebowanie na energię w porównaniu do przedstawionych dalej technologii, co jest ważną kwestią w przypadku urządzeń posiadających zasilanie z baterii.

Kolejnym sposobem komunikacji między urządzeniami może być wykorzystanie technologii Bluetooth. Charakterystyka przesyłania danych opierająca się na krótkim zasięgu, małej konsumpcji energii elektrycznej i małej szybkości sprawia, że jej wykorzystanie idealnie wpasowuje się w sieć urządzeń obejmujących jedno pomieszczenie. Minusem jest konieczność posiadania centrali służącej jako punkt dostępu do Internetu, a plusem możliwość połączenia z urządzeniem mobilnym.

Następnym protokołem komunikacji bezprzewodowej, który został stworzony głównie na potrzeby urządzeń IoT jest Zigbee. Wykorzystuje on pasma sieci 2,4 GHz, 915 MHz lub 868 MHz, przesyłając dane na odległość do 20 metrów i tworząc sieci w topologii siatki, co sprawia, że każde z urządzeń odbiera i przekazuje dane z innych znajdujących się w sieci. Dzięki takiemu rozwiązaniu z łatwością możliwe jest pokrycie większych powierzchni, czy też wielu pięter mieszkania. Charakterystyczną cechą technologii jest niskie zużycie energii [2].

Bezpośrednim konkurentem dla protokołu Zigbee jest pracująca na pasmach 868.42 MHz i 908.42 MHz, Z-Wave. Charakteryzuje się ona również działaniem w topologii siatki, jednak oferuje mniejsze możliwości jeżeli chodzi

o maksymalną liczbę urządzeń podłączonych w jednej sieci. Dla Z-Wave wynosi ona 232, a Zigbee potrafi utrzymać połączenie pomiędzy aż 65 000 urządzeń. Z-Wave jest natomiast lepszy pod względem maksymalnej odległości pomiędzy urządzeniami, która może wynosić nawet do 100 metrów [4].

2.2. Oprogramowanie sterujące

Aspektem, od którego zależy wydajność, ale przede wszystkim funkcjonalność sieci zbudowanej z urządzeń IoT jest oprogramowanie. Urządzenia dostępne na rynku, coraz częściej otrzymują możliwość obsługi wybranych systemów inteligentnych domów, lub są wręcz projektowane i produkowane z myślą o konkretnym systemie. Nie oznacza to, że nie występują również przyrządy posiadające autorskie oprogramowanie z przygotowanymi specjalnie aplikacjami, jednak wadą takiego rozwiązania jest brak możliwości reakcji na dane generowane przez inne urządzenia. Najpopularniejszymi systemami są te wykorzystujące systemy urządzeń mobilnych, takie jak Google Home od Google, czy HomeKit od Apple, które to firmy posiadają duopol na rynku systemów smartfonów (Android i IOS). Kolejnym ze zjawisk, które występują na rynku systemów inteligentnych domów, jest obsługiwanie urządzeń, które najczęściej zostały wyprodukowane przez producenta oprogramowania – przeważnie aplikacji mobilnej. Do takich systemów zalicza się przykładowo Mi Home od Xiaomi, czy FIBARO Home Center od Fibaro. Oddzielną kategorią systemów inteligentnych domów, jest oprogramowanie dostępne na licencji open source, takie jak OpenHAB, Domoticz, czy Home Assistant. z racji udziału społeczności w tworzeniu takich systemów, obsługują one coraz więcej urządzeń i funkcjonalności. Ciekawym rozwiązaniem, pozwalającym na łączenie urządzeń różnych producentów jest również IFTTT. Jest to serwis internetowy, który pozwala na obsługę i tworzenie własnych apletów, czyli prostych poleceń wykonujących się w koncepcji „IfThis Then That", co można przetłumaczyć na „jeśli to, to to". Pozwala ona dodać zależności pomiędzy aplikacjami obsługiwanymi przez serwis i wykonać polecenia na podstawie ich powiadomień, czyli wybrać zdarzenie wywołane przez jedno z urządzeń i gdy ono wystąpi wykonać ustaloną akcję.

3. Cel i plan badań

Podstawą technologii IoT jest generowanie, przesyłanie i przetwarzanie odebranych danych przez urządzenia, dlatego wykorzystanie możliwości zostało przedstawione na przykładzie logów z funkcjonalnego systemu inteligentnego domu. Logi pobrane z systemu ukazują przepływ danych, przetwarzanych w systemie oraz w konsekwencji, zachowanie poszczególnych urządzeń, w zależności od odebranych danych i przygotowanego wcześniej oprogramowania. Celem badań było wykazanie możliwości technologii Internet of Things poprzez zachowanie poszczególnych urządzeń, które obecnie są dostępne na rynku i sprawdzenie, czy system inteligentnego domu, może zostać wykonany w prosty sposób, przy wykorzystaniu kilku urządzeń i oprogramowania typu open source. Ponadto został przedstawiony i zweryfikowany proces przygotowania i pracy takiego systemu z wykorzystaniem protokołu sieciowego Z-Wave. Tezą postawioną w niniejszej pracy jest stwierdzenie:

Urządzenia IoT dostępne na rynku nadają się do wykorzystania w systemach inteligentnych domów.

Badania doprowadziły do zweryfikowania postawionych hipotez:

Hipoteza 1: Dostępne urządzenia IoT można włączyć do systemu inteligentnego domu dzięki ogólnie dostępnemu oprogramowaniu.

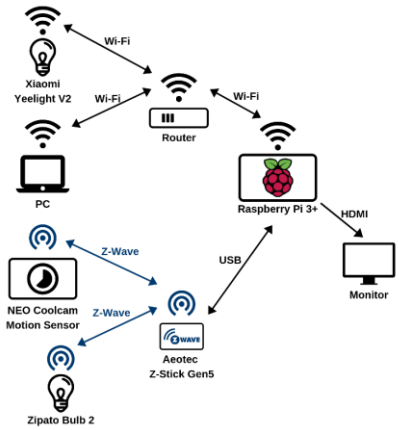
Hipoteza 2: Możliwe jest oprogramowanie działania poszczególnych urządzeń IoT działających w systemie inteligentnego domu.

Hipoteza 3: Dzięki stosowanym standardom IoT urządzenia w systemie inteligentnego domu mogą ze sobą współpracować.

W celu weryfikacji postawionych założeń, został przygotowany system inteligentnego domu, który łączy się z urządzeniami poprzez sieć Wi-Fi oraz sieć wykorzystującą protokół Z-Wave.

3.1. System inteligentnego domu

Model systemu inteligentnego domu składa się z kilku urządzeń działających w dwóch sieciach bezprzewodowych. Podstawowym urządzeniem jest centrala Raspberry Pi 3+ z zainstalowanym systemem kontrolującym pracę urządzeń. Centrala posiada moduł sieci Wi-Fi, jak i moduł sieciowy Aeotec Z-Stick Gen5, umożliwiający stworzenie sieci bezprzewodowej w technologii Z-Wave. w systemie jako urządzenia przeznaczone do sterowania, zastosowano dwie żarówki połączone w dwóch różnych sieciach bezprzewodowych. Zipato Bulb 2 obsługuje połączenie bezprzewodowe korzystające z protokołu Z-Wave, a Xiaomi Yeelight V2 w technologii Wi-Fi. Urządzeniem posiadającym czujnik trzech właściwości (temperatura, natężenie światła, ruch) jest NEO Coolcam Motion Sensor i to na podstawie odczytów tego czujnika wykonywane były akcje pozostałych urządzeń.



Rys. 1. Schemat ideowy połączeń pomiędzy urządzeniami

Do instalacji na wcześniej przedstawionej konfiguracji sprzętowej został wybrany system inteligentnego budynku Domoticz, udostępniane na zasadzie licencji open source. Instalacja systemu, wymagała instalacji na urządzeniu Raspberry Pi systemu Raspbian, a następnie z konsoli systemowej, wpisania komendy pobierającej i instalującej Domoticz.Głównym celem systemu inteligentnego domu jest zautomatyzowanie czynności. Domoticz opiera proces automatyzacji o wydarzenia (ang. events) i skrypty (ang. scripts). Każde wydarzenie ma własną nazwę, metodę skryptu i typ, który określa kiedy wydarzenie ma być uruchamiane. Typ wydarzenia może być określony jako zależny od urządzenia (uruchamiany podczas zmiany danych przesyłanych z urządzenia), zabezpieczenia (zmiana uzbrojenia systemu) lub czasu (wydarzenie uruchamiane cyklicznie, z określonym odstępem czasowym). Każde wydarzenie dodatkowo ma określoną wartość określającą, jego stan aktywności (aktywne/ nie aktywne). Skrypty przeznaczone do wykorzystania w wydarzeniach mogą być przygotowane za pomocą edytora Blockly lub języków skryptowych takich jak LUA, dzVents, Perl czy PHP.

4. Konfiguracja systemu Domoticz

Aby zainstalować Domoticz należało posłużyć się instalacją systemu Raspbian i z poziomu konsoli systemowej należało wpisać komendę:

sudo curl -L install.domoticz.com | bash

Proces instalacji przebiegł bezproblemowo, a po jej zakończeniu system został zrestartowany. Wtedy możliwe było sprawdzenie jaki adres sieciowy został przypisany do urządzenia. Następnym krokiem było wpisanie tego adresu w przeglądarce Google Chrome, na systemie Windows 10, który był zainstalowany na komputerze podłączonym do tej samej sieci Wi-Fi co Raspberry Pi, w celu połączenia z witryną internetową do zarządzania udostępnioną pod tym adresem poprzez zainstalowany serwis Domoticz. Dzięki temu było możliwe zarządzanie systemem Domoticz z poziomu systemu internetowego.

W celu dodania urządzeń sieci Z-Wave, należało w sekcji „Sprzęt" dodać moduł Aeotec Z-Stick Gen5 jako urządzenie typu Open Z-Wave USB, a następnie włączyć skanowanie urządzeń. w trakcie skanowania należało nacisnąć przycisk na czujniku NEO Coolcam Motion Sensor, w celu włączenia do sieci Z-Wave. Żarówka Zipato Bulb 2 została wykryta automatycznie, bez konieczności wykonywania jakichkolwiek akcji. Żarówkę Xiaomi Yeelight V2, pracującą w sieci Wi-Fi podłączono do systemu poprzez dodanie w sekcji „Sprzęt" wirtualnego kontrolera Yellight LED. Następnie należało w aplikacji Yeelight służącej do kontrolowania pracy urządzenia, a zainstalowanej na urządzeniu mobilnym z systemem Android, włączyć możliwość sterowania żarówką poprzez sieć LAN. Po wykonaniu tej czynności, włączono skanowanie na kontrolerze w systemie Domoticz, który odnalazł urządzenie YeeLight RBGW.

W sekcji „Urządzenia" należało włączyć podmioty, które zostały dodane do systemu. Obydwie żarówki zostały sklasyfikowane jako urządzenia tego samego typu, z funkcjonalnością, pozwalającą zarządzać natężeniem oraz kolorem emitowanego światła. Podstawową różnicą w przypadku obu urządzeń była sekcja określona jako „Sprzęt", gdzie w przypadku połączenia poprzez sieć Wi-Fi została wskazana nazwa urządzenia, a dla Zipato Bulb 2 widniała nazwa modułu sieciowego, dzięki któremu możliwe było połączenie. Różnicę da się również zobaczyć w sekcji „Podtyp" gdzie dla urządzenia Yellight wskazany jest RGBWW, czyli kontrola emitowanego światła w palecie barw RGB oraz światła chłodnego białego i ciepłego białego. w przypadku drugiej żarówki, podtyp określono jako RGBWWZ, co pozwala na kontrolę w paletach jak w poprzednim wypadku oraz mieszania palety odpowiedzialnej za kolory RGB i palety zakresów światła białego. Czujnik był pokazany w systemie jako trzy oddzielne moduły służące do pomiaru temperatury, ruchu oraz natężenia światła.

Rys. 2. Urządzenia systemu Domoticz

W celu przeprowadzenia przypadków testowych, należało włączyć urządzenia:

- YeeLight RBGW – kontrola żarówki Yellight
- Zipato RGBW – kontrola żarówki Zipato
- Lux Sensor– odczyt czujnika wartości natężenia światła
- Motion Sensor – odczyt czujnika ruchu
- Temperature Sensor – odczyt czujnika temperatury

5. Przypadki testowe

Przypadki testowe obejmowały instalację i konfigurację systemu z wykorzystaniem opisanych urządzeń, oraz próbę uzależnienia inteligentnych żarówek od czujnika natężenia światła, czujnika ruchu oraz wartości mierzonej temperatury. Pierwszym etapem konfiguracji było sprawdzenie, czy żarówki połączone poprzez sieć bezprzewodową Wi-Fi i Z-Wavesą w stanie regulować natężenie oświetlenia w zależności od czujnika ruchu. w drugim przypadku została przeprowadzona konfiguracja wykorzystująca czujnik natężenia oświetlenia, który po zmianie wartości, wymusza na systemie zmianę mocy świecenia żarówek. Trzeci przypadek obejmie regulację koloru światła żarówki, w zależności od zmierzonej wartości temperatury.

Podstawową funkcjonalnością NEO Coolcam Motion Sensor, jest wykrywanie ruchu. Aby umożliwić prawidłową pracę czujnika ruchu, należało ustawić parametry konfiguracyjne urządzenia:

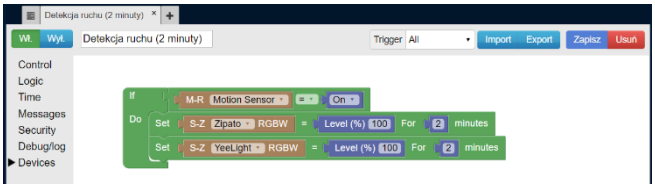
Tabela 1. Ustawione parametry konfiguracyjne urządzenia NEO Coolcam Motion Sensor

Numer parametru	Nazwa	Wartość
1	Motion detection sensitivity	High Sensivity
2	Motion detection ON time	30
3	Basic Set Level	255
4	Motion detection function	Enable
6	Motion detection blind time	5
10	Motion detection LED indicator	Enable

Aby uzyskać pewność, że nawet najmniejszy ruch użytkownika zostanie wykryty, ustawiono parametr numer 1 jako wysoką czułość. Parametrem numer 2 określono, że urządzenia włączone poprzez pomiar czujnika ruchu mają pozostać włączone przez 30 sekund. w parametrze numer 6 ustawiono 5 sekund jako czas po jakim czujnik zacznie reagować na kolejny ruch w jego obszarze, od momentu wykrycia poprzedniego ruchu. Dodatkowo w parametrze numer 10 włączono sygnalizację diodą LED po wykryciu ruchu przez czujnik.

W przypadku testowym „Detekcja ruchu" uzależniono włączenie żarówek od wykrycia ruchu przez sensor. Pierwszy przypadek powoduje włączenie oświetlenia po wykryciu ruchu, a następnie jego wyłączenie po zakończeniu aktywności czujnika. Drugi przypadek „Detekcja ruchu (2 minuty)" powoduje włączenie oświetlenia po wykryciu ruchu na czas 2 minut. Obydwa przypadki zostały obsłużone przez stworzenie wydarzeń ze skryptami edytora Blockly.

Rys. 3. Skrypt wydarzenia „Detekcja ruchu"



Rys. 4. Skrypt wydarzenia „Detekcja ruchu (2 minuty)"

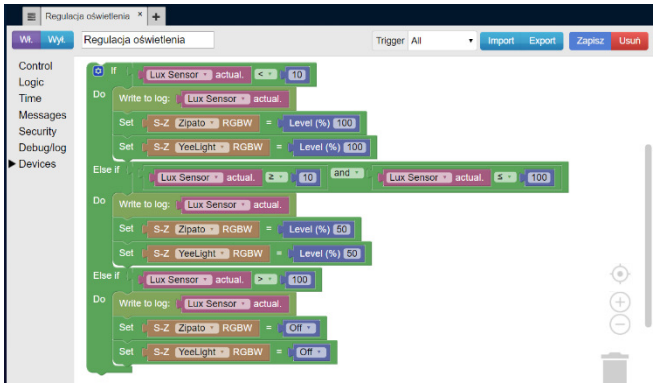
Drugą z funkcjonalności urządzenia NEO Coolcam Motion Sensor jest pomiar natężenia światła. w celu zapewnienia poprawnego działania sensora ustawiono następujące parametry:

Tabela 2. Ustawione parametry konfiguracyjne urządzenia NEO Coolcam Motion Sensor

Numer parametru	Nazwa	Wartość
7	Illumination reporting interval	180
8	Illuminationfunction	Enable
9	Illuminationreport threshold	5

Parametr numer 7, ustawiono 180 sekund, jako czas jaki musi upłynąć aby urządzenie przesłało zmierzoną wartość natężenia światła, w przypadku gdy nie odnotowano zmiany większej, niż wartość parametru numer 9. w parametrze 9 ustawiono wartość 5 lux, jako różnicę od poprzedniego pomiaru, która powoduje przesłanie nowego pomiaru do kontrolera.

W pierwszym przypadku testowym „Regulacja oświetlenia" regulowano intensywność emitowanego światła żarówek, w zależności od wartości zmierzonej przez czujnik natężenia oświetlenia. Gdy wartość zmierzona wynosiła poniżej 10 lux, to żarówki świeciły z maksymalną mocą. Dla zmierzonych wartości pomiędzy 10 lux a 100 lux, moc żarówek była ustawiana na 50%. w przypadku zmierzenia ponad 100 lux, system wyłączał oświetlenie. Wraz z każdą zmianą wartości odczytanej z urządzenia Lux Sensor, system zapisywał odebraną wartość.



Rys. 5. Skrypt wydarzenia „Regulacja oświetlenia"

Drugi przypadek testowy „Automatyczna regulacja oświetlenia" pokazuje w jaki sposób możliwe jest wykonanie automatycznej regulacji oświetlenia, do ustalonej wartości natężenia światła. Tym razem skrypt do wydarzenia został przygotowany w języku skryptowym dzVents. Zadaniem skryptu jest regulowanie mocy świecenia żarówek w ten sposób, aby uzyskać wartość oświetlenia w pomieszczeniu zadeklarowaną w zmiennej LuxValue.

Wł. Wył. Automatyczna regulacja oś

```
1  return {
2      on = {
3          devices = { 'Lux Sensor' }
4      },
5      execute = function(domoticz, item, triggerInfo)
6          local LuxValue = 50
7          local GranicaBledu = 5
8          local LuxDevice = domoticz.devices('Lux Sensor')
9          local LuxLevel = LuxDevice.lux
10         local ZipatoDevice = domoticz.devices('Zipato RGBW')
11         local ZipatoLevel = ZipatoDevice.level
12         local YeeLightDevice = domoticz.devices('YeeLight RGBW')
13         local YeeLightLevel = YeeLightDevice.level
14
15         if LuxLevel + GranicaBledu < LuxValue then
16             ZipatoDevice.dimTo(ZipatoLevel + 10)
17             YeeLightDevice.dimTo(ZipatoLevel + 10)
18         end
19         if LuxLevel - GranicaBledu > LuxValue then
20             ZipatoDevice.dimTo(ZipatoLevel - 10)
21             YeeLightDevice.dimTo(ZipatoLevel - 10)
22         end
23     end
24 }
```

Rys. 6. Skrypt wydarzenia „Automatyczna regulacja oświetlenia”

Skrypt w linii 3 deklaruje urządzenie, którego zmiana wartości ma powodować wykonanie kodu. Tym urządzeniem jest Lux Sensor, czyli czujnik natężenia światła. Następnie deklarowane są zmienne:

- LuxValue – wartość oświetlenia jaką chcemy uzyskać w pomieszczeniu
- GranicaBledu – różnica pomiędzy pomiarem czujnika oświetlenia a docelową wartością (LuxValue), jaka nie spowoduje wywołania skryptu
- LuxDevice – zmienna przechowująca odwołanie do urządzenia Lux Sensor
- LuxLevel – zmienna przechowująca wartość zmierzoną przez urządzenie Lux Sensor
- ZipatoDevice – zmienna przechowująca odwołanie do urządzenia Zipato Bulb 2
- ZipatoLevel – zmienna przechowująca wartość mocy oświetlenia emitowanej przez żarówkę Zipato Bulb 2
- YelLightDevice – zmienna przechowująca odwołanie do urządzenia Xiaomi Yeelight V2
- YeeLightLevel – zmienna przechowująca wartość mocy oświetlenia emitowanej przez żarówkę Xiaomi Yeelight V2

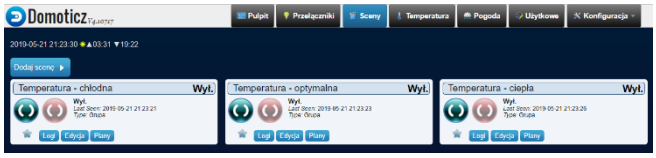
W ostatniej części skryptu wywoływane są instrukcje warunkowe „if”, które sprawdzają czy obecny pomiar czujnika natężenia światła, jest mniejszy bądź większy od docelowej wartości i odpowiednio zwiększają bądź redukują moc świecenia żarówek.

Pomiary temperatury, w przeciwieństwie do poprzednich wartości mierzonych przez urządzenie, nie posiadają możliwości konfiguracji.

W przypadku testowym posłużono się żarówkami jako urządzeniem informującym o obecnej temperaturze w pomieszczeniu. Temperatury skategoryzowano i podzielono na 3 zakresy:

- mniej niż 20°C – temperatura chłodna
- od 20°C do 25°C – temperatura optymalna
- więcej niż 25°C – temperatura ciepła

Aby możliwe było ustawienie kolorów emitowanych przez żarówki, utworzono trzy sceny. w każdej scenie należało dodać dwie żarówki i ustawić ich wartości w kodzie heksadecymalnym, przedstawione w Tabeli 3.



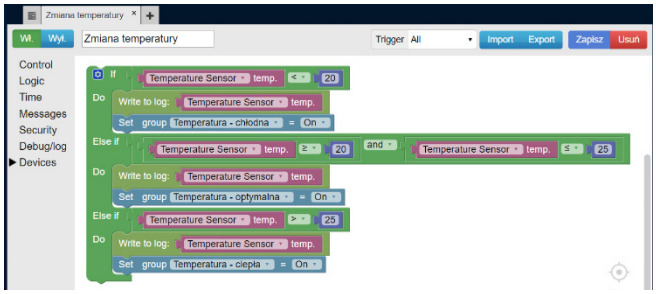
Rys. 7. Sceny w systemie Domoticz

Aktywacja każdej ze scen powoduje włączenie i ustawienie określonych kolorów na żarówkach:

Tabela 3. Kolory żarówek ustawione dla danej sceny

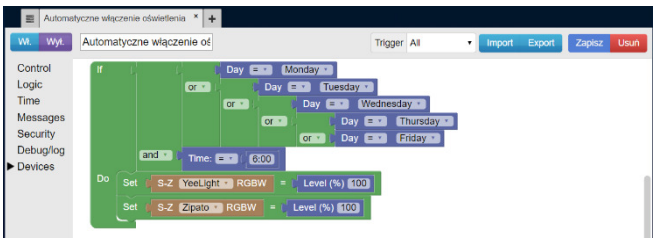
Nazwa sceny	Kod Hex	Wartość RGB	Kolor
Temperatura – chłodna	0100FF	1, 0, 255	
Temperatura – optymalna	2CFF00	44, 255, 0	
Temperatura – ciepła	FF000C	255, 0, 12	

Za aktywację scen odpowiada wydarzenie „Zmiana temperatury” ze skryptem w edytorze Blockly. Skrypt zmienia stan konkretnej sceny w zależności od zmierzonej temperatury.



Rys. 8. Skrypt wydarzenia „Zmiana temperatury”

Edytor Blockly dostępny z poziomu systemu Domoticz, pozwala również na ustawienie zależności innych urządzeń od obecnego czasu. w utworzonym wydarzeniu „Automatyczne włączanie oświetlenia” posłużono się funkcjonalnością, która włącza oświetlenie na maksymalną wartość, o godzinie 6:00, w dniach od poniedziałku do piątku.



Rys. 9. Skrypt wydarzenia „Automatyczne włączanie oświetlenia”

6. Wyniki

Instalacja systemu Domoticz na urządzeniu Raspberry Pi 3+, nie sprawiła problemów i nie stworzyła nieprzewidzianych sytuacji. Wszystkie kroki zostały wykonane według instrukcji instalacji, dostępnej na stronie internetowej oprogramowania. Dodanie urządzeń i utworzenie połączeń zarówno w sieci Wi-Fi, jak i Z-Wave było możliwe dzięki intuicyjnemu interfejsowi systemu inteligentnego budynku i przebiegło bezproblemowo.

W pierwszym przypadku „Detekcja ruchu” system włączał oświetlenie, w momencie wykrycia ruchu przez czujnik. System po wykryciu zmiany stanu czujnika ruchu Motion Sensor informuje o uaktywnieniu wydarzenia i zmianie stanu żarówek. System zmienia poziom oświetlenia komendą ustawiając wartość Level: 99. Po około 30 sekundach, czyli wartości ustawionej w konfiguracji czujnika ruchu, system aktywuje ponownie wydarzenie wyłączając oświetlenie (Level: 0).

```
20:36:10.869 (Z-Stick) Light/Switch (Motion Sensor)
20:36:10.370 EventSystem: Event triggered: Detekcja ruchu_1
20:36:10.373 OpenZWave: Domoticz has send a Switch command!, Level: 99, NodeID: 2
(0x02)
20:36:10.388 (Z-Stick) Color Switch (Zipato RGBW)
20:36:10.941 (Z-Stick) Light/Switch (Level)
20:36:11.008 (YeeLight) Color Switch (YeeLight RGBW)
20:36:41.999 (Z-Stick) Light/Switch (Motion Sensor)
20:36:42.001 EventSystem: Event triggered: Detekcja ruchu_2
20:36:42.026 OpenZWave: Domoticz has send a Switch command!, Level: 0, NodeID: 2
(0x02)
20:36:42.040 (Z-Stick) Color Switch (Zipato RGBW)
20:36:42.113 (YeeLight) Color Switch (YeeLight RGBW)
```

Rys. 10. Logi systemu Domoticz - aktywacja wydarzenia "Detekcja ruchu"

Kolejny przypadek „Detekcja ruchu (2 minuty)” posiadała podobną sekwencję wydarzeń, jednak tym razem komenda aktywująca skrypt wyłączający oświetlenie wysyłana jest dopiero po około 2 minutach od momentu wykrycia ruchu, czyli zgodnie z przygotowanym skryptem wydarzenia.

```
20:39:47.706 Light/Switch (Motion Sensor)
20:39:47.707 (Z-Stick) EventSystem: Event triggered: Detekcja ruchu (2 minuty)
20:39:47.737 OpenZWave: Domoticz has send a Switch command!, Level: 49, NodeID: 2
(0x02)
20:39:47.758 (Z-Stick) Color Switch (Zipato RGBW)
20:39:47.804 (Z-Stick) Light/Switch (Level)
20:39:47.885 (YeeLight) Color Switch (YeeLight RGBW)
20:40:18.809 (Z-Stick) Light/Switch (Motion Sensor)
20:41:47.717 OpenZWave: Domoticz has send a Switch command!, Level: 0, NodeID: 2
(0x02)
20:41:47.731 (Z-Stick) Color Switch (Zipato RGBW)
20:41:47.801 (YeeLight) Color Switch (YeeLight RGBW)
```

Rys. 11. Logi systemu Domoticz - aktywacja wydarzenia "Detekcja ruchu (2 minuty)"

Przypadek „Regulacja oświetlenia” pozwolił na zmianę poziomu mocy świecenia żarówek. System po odebraniu nowej wartości z czujnika Lux, informuje, że wydarzenie zostało uruchomione. Wysyła komendę „Switch command” z wartością „Level” określającą zakres mocy, w skali 0-100. Następnie informuje, że urządzenia Zipato RGBW i YeeLight RGBW zmieniły swój stan. Sekwencja powtarza się trzykrotnie, gdzie w zależności od poziomu zmierzonego przez czujnik Lux, zostaje uruchomiona inna instrukcja warunkowa. Za każdym razem po zmianie wartości Lux Sensor, system zapisuje odczytaną wartość komendą „Status”.

```
21:03:29.084 (Z-Stick) Lux (Lux Sensor)
21:03:29.086 EventSystem: Event triggered: Regulacja oświetlenia_1
21:03:29.118 OpenzWave: Domoticz has send a Switch command!, Level: 99, NodeID: 2
(0x02)
21:03:29.140 (Z-Stick) Color Switch (Zipato RGBW)
21:03:29.224 (Z-Stick) Light/Switch (Level)
21:03:29.268 (YeeLight) Color Switch (YeeLight RGBW)
21:03:29.086 Status: 2
21:03:34.756 (Z-Stick) Lux (Lux Sensor)
21:03:34.757 EventSystem: Event triggered: Regulacja oświetlenia_2
21:03:34.759 OpenzWave: Domoticz has send a Switch command!, Level: 50, NodeID: 2
(0x02)
21:03:34.772 (Z-Stick) Color Switch (Zipato RGBW)
21:03:34.931 (YeeLight) Color Switch (YeeLight RGBW)
21:03:34.758 Status: 14
21:03:40.495 (Z-Stick) Lux (Lux Sensor)
21:03:40.497 EventSystem: Event triggered: Regulacja oświetlenia_3
21:03:40.502 OpenzWave: Domoticz has send a Switch command!, Level: 0, NodeID: 2
(0x02)
21:03:40.521 (Z-Stick) Color Switch (Zipato RGBW)
21:03:40.594 (YeeLight) Color Switch (YeeLight RGBW)
21:03:40.497 Status: 132
```

Rys. 12. Logi systemu Domoticz - aktywacja wydarzenia "Regulacja oświetlenia"

Przypadek „Automatyczna regulacja oświetlenia” reguluje poziom świecenia żarówek za każdym razem o 10%. System w omawianym przypadkusekwencyjnie odczytywał nowe wartości z czujnika natężenia oświetlenia, oraz wykonywał skrypt, zwiększając moc światła emitowanego przez żarówki. Finalnie po zmierzeniu wartości, która mieściła się w granicy błędu wartości docelowej, system wykonał skrypt, nie zmieniając pracy żarówek, co pozwoliło na ustabilizowaniu oświetlenia pomieszczenia na wymaganym poziomie.

Po odebraniu nowej wartości z czujnika temperatury, uruchomiony został skrypt wydarzenia „Zmiana temperatury”. w logach system zapisał aktywowanie sceny „Temperatura – optymalna”, oraz obu żarówek. Do żarówek zostało wysłane polecenie ustawienia mocy emitowania światła przez urządzenia (Level) oraz ustalenia koloru, podane w pięciu wartościach (Red, Green, Blue, White, cWhite). System zapisał również temperaturę zmierzoną przez urządzenie Temperature Sensor.

W funkcjonalnościach systemu niezależnych od wartości odczytanych przez czujniki została przedstawiona sytuacja z włączeniem oświetlenia o konkretnej godzinie ustawionej w skrypcie „Automatyczne włączanie oświetlenia”. Skrypt został zaprogramowany tak, aby aktywować urządzenia tylko w dni robocze tj. od poniedziałku do piątku.

```
21:15:33.939 (Z-Stick) Lux (Lux Sensor)
21:15:40.073 OpenZWave: Domoticz has send a Switch command!, Level: 10, NodeID: 2
(0x02)
21:15:40.081 (Z-Stick) Color Switch (Zipato RGBW)
21:15:40.204 (YeeLight) Color Switch (YeeLight RGBW)
21:15:40.055 Status: dzVents: Info: Handling events for: "Lux Sensor", value: "0"
21:15:40.055 Status: dzVents: Info: ------ Start internal script: Automatyczna
regulacja oświetlenia: Device: "Lux Sensor (Z-Stick)", Index: 19
21:15:40.057 Status: dzVenta: Info: ------ Finished Automatyczna regulacja
oświetlenia
21:15:40.058 Status: EventSystem: Script event triggered:
/home/pi/domoticz/dzVenta/ runtime/dzVenta.lua
21:15:43.769 (Z-Stick) Lux (Lux Sensor)
21:15:49.904 OpenZWave: Domoticz has send a Switch command!, Level: 20, NodeID: 2
(0x02)
21:15:49.950 (Z-Stick) Color Switch (Zipato RGBW)
21:15:49.884 Status: dzVents: Info: Handling events for: "Lux Sensor", value: "21"
21:15:49.884 Status: dzVents: Info: ------ Start internal script: Automatyczna
regulacja oświetlenia: Device: "Lux Sensor (Z-Stick)", Index: 19
21:15:49.886 Status: dzVenta: Info: ------ Finished Automatyczna regulacja
oświetlenia
21:15:49.887 Status: EventSystem: Script event triggered:
/home/pi/domoticz/dzVenta/ runtime/dzVenta.lua
21:15:50.021 (YeeLight) Color Switch (YeeLight RGBW)
21:16:02.446 (Z-Stick) Lux (Lux Sensor)
21:16:02.507 OpenZWave: Domoticz has send a Switch command!, Level: 30, NodeID: 2
(0x02)
21:16:02.612 (Z-Stick) Color Switch (Zipato RGBW)
21:16:02.722 (YeeLight) Color Switch (YeeLight RGBW)
21:16:02.591 Status: dzVents: Info: Handling events for: "Lux Sensor", value: "27"
21:16:02.591 Status: dzVents: Info: ------ Start internal script: Automatyczna
regulacja oświetlenia: Device: "Lux Sensor (Z-Stick)", Index: 19
21:16:02.593 Status: dzVenta: Info: ------ Finished Automatyczna regulacja
oświetlenia
21:16:02.594 Status: EventSystem: Script event triggered:
/home/pi/domoticz/dzVenta/ runtime/dzVenta . lua
21:16:03.139 (Z-Stick) Lux (Lux Sensor)
21:16:05.336 OpenZWave: Domoticz has send a Switch command!, Level: 40, NodeID: 2
(0x02)
21:16:05.340 (Z-Stick; Color Switch (Zipato RGBW)
21:16:03.451 (YeeLight) Color Switch (YeeLight RGBW)
21:16:09.300 Status: dzVents: Info: Handling events for: "Lux Sensor", value: "35"
21:16:09.300 Status: dzVents: Info: ------ Start internal script: Automatyczna
regulacja oświetlenia: Device: "Lux Sensor (Z-Stick)", Index: 19
21:16:09.302 Status: dzVenta: Info: ------ Finished Automatyczna regulacja
oświetlenia
21:16:09.303 Status: EventSystem: Script event triggered:
/home/pi/domoticz/dzVenta/ runtime/dzVenta . lua
21:16:13.743 (Z-Stick) Lux (Lux Sensor)
21:16:18.860 Status: dzVents: Info: Handling events for: "Lux Sensor", value: "47"
21:16:18.860 Status: dzVents: Info: ------ Start internal script: Automatyczna
regulacja oświetlenia: Device: "Lux Sensor (Z-Stick)", Index: 19
21:16:18.861 Status: dzVenta: Info: ------ Finished Automatyczna regulacja
oświetlenia
```

Rys. 13. Logi systemu Domoticz - aktywacja wydarzenia "Automatyczna regulacja oświetlenia"

```
20:54:56.281 (Z-Stick) Temp (Temperature Sensor)
20:54:56.284 EventSystem: Event triggered: Zmiana temperatury 2
20:54:56.313 Activating Scene/Group: [Temperatura – optymalna]
20:54:56.314 Activating Scene/Group Device: Zipato RGBW (On)
20:54:56.314 OpenzWave: Domoticz has send a Switch command!, Level: 99, NodelD: 2
(0x02)
20:54:56.315 Red: 044, Green:255, Blue:000, wwhite:000, cwhite:000
20:54:56.330 (Z-Stick) Color Switch (Zipato RGBW
20:54:56.379 (Z-Stick) Light/Switch (Level)
20:54:56.381 Activating Scene/Grcup Device: YeeLight RGBW (On)
20:54:56.551 (YeeLight) Color Switch (YeeLight RGBW)
20:54:56.284 Status: 24.5
```

Rys. 14. Logi systemu Domoticz - aktywacja wydarzenia "Zmiana temperatury"

```
06:00:00.004 EventSystem: Event triggered: Automatyczne włączenie oświetlenia_1
06:00:00.152 (YeeLight) Color Switch (YeeLight RGBW)
06:00:00.152 OpenzWave: Domoticz has send a Switch command!, Level: 99, NodelD: 2
(0x02)
06:00:00.170 (Z-Stick) Color Switch (Zipato RGBW)
06:00:00.239 (Z-Stick) Light/Switch (Level)
```

Rys. 15. Logi systemu Domoticz - aktywacja wydarzenia "Automatyczne włączanie oświetlenia"

7. Wnioski

Uzależnienie świecenia żarówek od czujnika ruchu umożliwia pomoc lokatorowi budynku w sytuacji, gdy nie ma możliwości fizycznego naciśnięcia przełącznika, np. gdy lokator niesie coś w rękach. Przypadki obsługiwały sytuacje, w których światło włączało się po wykryciu ruchu i zostało utrzymywane przez 30 sekund, czyli wartość ustawioną w parametryzacji czujnika ruchu, bądź włączane na precyzyjnie określony czas 2 minut. Obie sytuacje pozwalają na zmniejszenie zużycia energii elektrycznej, gdyż światło jest gaszone w krótkim czasie po wykryciu ostatniej aktywności w zasięgu sensora.

Funkcjonalności związane z czujnikiem natężenia światła pozwoliły na odpowiednie oświetlenie pomieszczenia, w czasie, gdy naturalne światło słoneczne nie jest wystarczające. w pierwszym przypadku „Regulacja oświetlenia" proste przełączanie w zależności od zmierzonej wartości może być wykorzystane w sytuacji, gdy na czujnik natężenia światła nie będą miały wpływu podłączone urządzenia oświetlające. Żarówki zmieniające poziom mocy oświetlenia, w pomieszczeniu, w którym znajduje się czujnik, mogłyby zakłócić jego pomiary i powodować włączanie niechcianych skryptów, co finalnie prowadziłoby do pracy urządzeń niezgodnie z przeznaczeniem. Drugi przypadek „Automatyczna regulacja oświetlenia" pokazuje możliwość automatycznej regulacji oświetlenia do zadeklarowanego poziomu w sytuacji, gdy czujnik natężenia światła znajduje się w tym samym pomieszczeniu co sterowane żarówki. Tym razem jednak należało użyć bardziej zaawansowanego języka skryptowego dzVents.

Zmiana koloru emitowanego światła w zależności od temperatury, ma charakter informacyjny i może być przydatna w sytuacjach, gdzie przykładowo, temperatura w pomieszczeniu musi być utrzymana w pewnym zakresie, a nie ma możliwości ustawienia jej konkretnej wartości w systemie ogrzewania.

Przypadki testowe pozwoliły na ukazanie kluczowego aspektu możliwego do zrealizowania dzięki technologii IoT w systemach inteligentnych domów, czyli automatyzacji czynności wykonywanych przez urządzenia. Automatyzacja możliwa była dzięki przechowywaniu utworzonych przez użytkownika skryptów i przetwarzaniu ich w zależności od wartości odebranych danych, pochodzących z urządzeń znajdujących się w systemie.

Wykonany system inteligentnego domu pozwolił potwierdzić wszystkie sformułowane początkowo hipotezy. Dzięki bezproblemowemu podłączeniu wszystkich urządzeń, do systemu Domoticz udostępnionemu na licencji open source, udało się potwierdzić tezę stwierdzającą, że „Dostępne urządzenia IoT można włączyć do systemu inteligentnego domu dzięki ogólnie dostępnemu oprogramowaniu". Stworzenie oprogramowania w edytorze Blockly oraz przy wykorzystaniu języka skryptowego dzVents potwierdziło, że „Możliwe jest oprogramowanie działania poszczególnych urządzeń IoT działających w systemie inteligentnego domu". Wyniki przeprowadzonych przypadków testowych, w których żarówki były uzależnione od odczytów czujników ruchu, temperatury i natężenia światła, pozwalają potwierdzić hipotezę, która twierdziła, że „Dzięki stosowanym standardom IoT urządzenia w systemie inteligentnego domu mogą ze sobą współpracować."

Udowodnienie postawionych hipotezoraz stworzenie działającego systemu inteligentnego domu pozwalają potwierdzić sformułowaną początkowo tezę, która twierdzi, że „**Urządzenia IoT dostępne na rynku nadają się do wykorzystania w systemach inteligentnych domów**".

Praca analityczna nie objęła wykorzystania połączeń innych niż poprzez Wi-Fi i protokół Z-Wave. Następne badania

mogłyby objąć pozostałe sposoby komunikacji pomiędzy urządzeniami wykorzystującymi technologię Internet of Things. Ponadto można by zbadać możliwość sterowania systemem spoza sieci wewnętrznej, tak aby możliwa była kontrola oraz odczyt wartości z urządzeń, podczas nieobecności domowników. Szczególną uwagą należałoby objąć wtedy obszar powiadomień alarmowych w systemach inteligentnych domów.

Literatura

[1] Augusto J., Nugent C., Designing Smart Homes: The Role of Artificial Intelligence, 2006.

[2] Deshpande R., Deshmukh T., Puranik M., Sagar P., a novel approach to design Smart Home Architecture in Energy Efficient way based on IOT-A Survey, 2018.

[3] Fouladi B., Ghanoun S., Security Evaluation of the Z-Wave Wireless Protocol, 2013.

[4] Stojkoska B., Trivodaliev T., a review of Internet of Things for smart home: Challenges and solutions, 2016.

[5] Talari S., Shafie-khah M., Siano P., Loia V, Tommasetti A., Catalão J., a Review of Smart Cities Based on the Internet of Things Concept, 2017.

[6] Wan j., Humar I., Zhang D., Industrial IoT Technologies and Applications: International Conference, Industrial IoT 2016, GuangZhou, China, March 25-26, 2016, Revised Selected Papers, 2016.