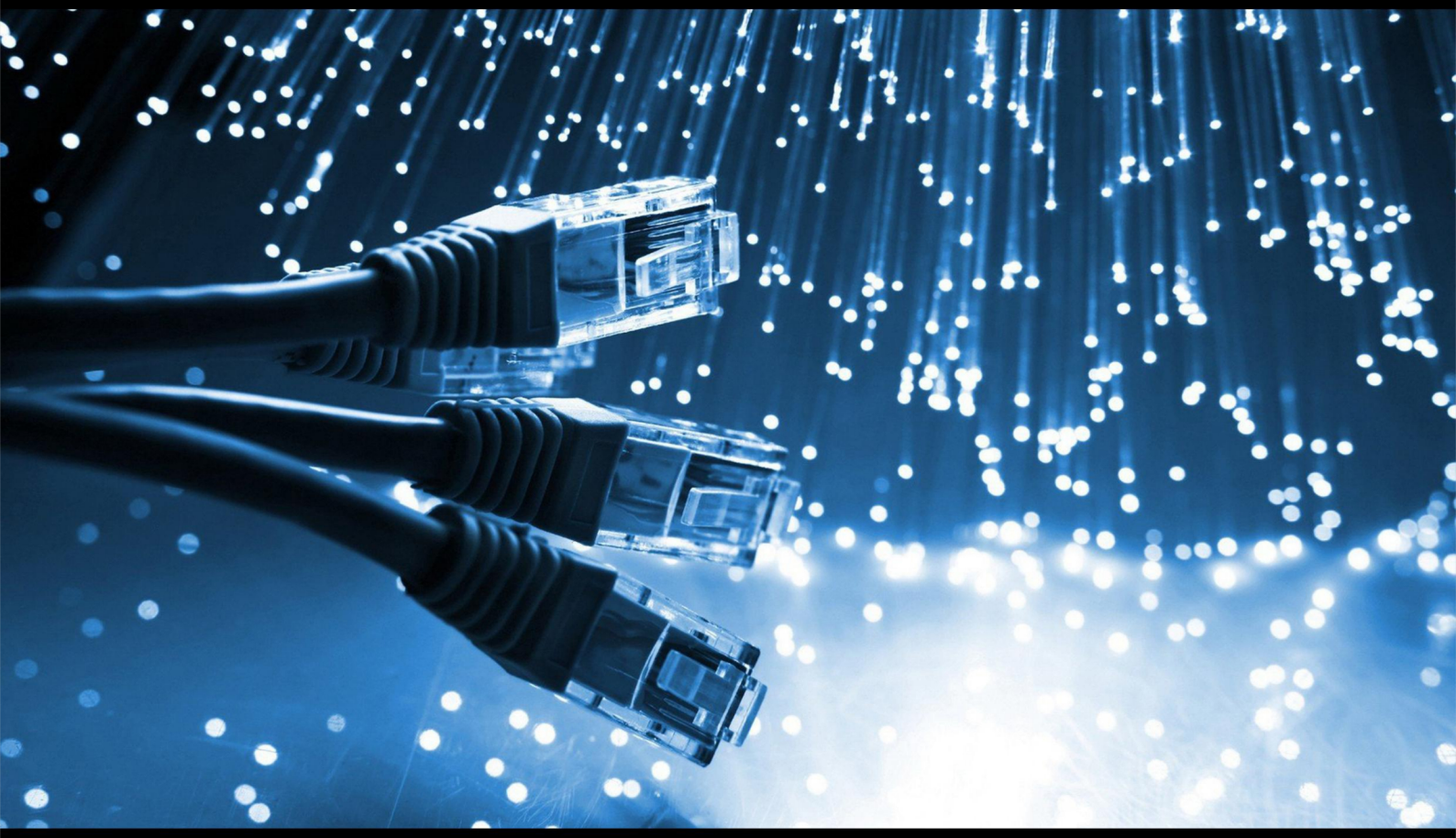
JCSI

Journal of Computer Sciences Institute

Volume 14/2020



Department of Computer Science
Lublin University of Technology

jcsi.pollub.pl

ISSN: 2544-0764

Redakcja JCSI
e-mail: jcsi@pollub.pl
www: jcsi.pollub.pl
Katedra 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. Piotr Kopniak
dr inż. Maciej Pańczyk
dr inż. Marek Miłosz
dr Mariusz Dzieńkowski
dr inż. Maria Skublewska-Paszkowska
dr Paweł Powroźnik
dr inż. Jakub Smołka
dr Beata Pańczyk
dr inż. Piotr Muryjas
dr inż. Dariusz Gutek
dr Edyta Łukasik
dr inż. Grzegorz Kozieł
dr inż. Tomasz Nowicki
dr inż. Kamil Żyła
dr inż. Jacek Kęsik

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
Department 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:
Piotr Kopniak
Maciej Pańczyk
Marek Miłosz
Mariusz Dzieńkowski
Maria Skublewska-Paszkowska
Paweł Powroźnik
Jakub Smołka
Beata Pańczyk
Piotr Muryjas
Dariusz Gutek
Edyta Łukasik
Grzegorz Kozieł
Tomasz Nowicki
Kamil Żyła
Jacek Kęsik

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

Cover design:
Marta Zbańska

ISSN 2544-0764

Spis treści

1. **Analiza porównawcza wydajności relacyjnych baz danych MySQL, PostgreSQL, MariaDB oraz H2**
   Katarzyna Krocz, Oleksandra Kizun, Maria Skublewska-Paszkowska.....1-7
2. **Porównanie aplikacji wspierających zastosowanie metodyk zwinnych w wytwarzaniu oprogramowania**
   Tomasz Bławucki, Siarhei Ramanovich, Maria Skublewska-Paszkowska.....8-13
3. **Ocena portalu nauczania języków obcych**
   Marek Szmit, Paweł Wojtaszko, Grzegorz Kozieł.....14-18
4. **Analiza funkcjonalna i wydajnościowa wybranych brokerów komunikatów w aplikacji rozproszonej**
   Tobiasz Kaciuczyk, Tomasz Korga, Jakub Smołka.....19-25
5. **Analiza porównawcza technologii Windows Presentation Foundation i Windows Forms**
   Michał Pasztaleniec, Maria Skublewska-Paszkowska.....26-30
6. **Wpływ języka programowania aplikacji chmurowej na wydajność jej implementacji w wybranych środowiskach serverless**
   Krzysztof Bezrąk, Sławomir Przyłucki.....31-36
7. **Zastosowanie maszyny wektorów nośnych w sterowaniu sygnalizacją świetlną**
   Artur Całuch, Adam Cieślikowski, Małgorzata Plechawska-Wójcik.....37-42
8. **Porównanie systemów mapowania obiektowo relacyjnego greenDao i Room**
   Maciej Lewiński.....43-47
9. **Ocena świadomości studentów informatyki w zakresie bezpieczeństwa komunikatorów internetowych**
   Paweł Stręciwilk, Grzegorz Kozieł.....48-54
10. **Metody wyznaczania wskaźników podobieństwa ruchu trójwymiarowego**
    Piotr Flisiak, Marcin Kuszyk.....55-58
11. **Analiza porównawcza wydajności frameworków Angular oraz Vue.js**
    Roman Baida, Maksym Andriienko, Małgorzata Plechawska-Wójcik.....59-64
12. **Analiza możliwości zastosowania gry typu serious game do nauki postępowania podczas udzielania pierwszej pomocy**
    Klaudia Zaborek, Małgorzata Plechawska-Wójcik.....65-72
13. **Porównanie wydajności narzędzi do tworzenia interfejsu aplikacji typu SPA na przykładzie React i Vue.js**
    Krzysztof Boczkowski, Beata Pańczyk.....73-77
14. **Analiza porównawcza technologii widoków dla aplikacji Spring**
    Vadym Borys, Roman Slezenko, Beata Pańczyk.....78-81
15. **Analiza wydajnościowa wybranych narzędzi do budowy aplikacji Single Page Application**
    Yehor Timanovskyi, Małgorzata Plechawska-Wójcik.....82-87
16. **Analiza wybranych metod oceny użyteczności w procesie tworzenia aplikacji internetowych**
    Krzysztof Nowak, Daniel Samolej.....88-93
17. **Analiza wydajności systemów bazodanowych: MySQL, MS SQL, PostgreSQL w kontekście aplikacji internetowych**
    Katarzyna Lachewicz.....94-100
18. **Metody wytwarzania realistycznych pomieszczeń – skanowanie 3D oraz modelowanie 3D**
    Aleksandra Salwierz, Tomasz Szymczyk.....101-108

Contents

1. **Perfomance analysis of relational databases MySQL, PostgreSQL, MariaDB and H2**
   Katarzyna Krocz, Oleksandra Kizun, Maria Skublewska-Paszkowska.....1-7
2. **Applications supporting utilization of agile methods in software development process**
   Tomasz Bławucki, Siarhei Ramanovich, Maria Skublewska-Paszkowska.....8-13
3. **An assessment of portal to learn foreign languages**
   Marek Szmit, Paweł Wojtaszko, Grzegorz Kozieł.....14-18
4. **Functional and performance analysis of selected message brokers in a distributed application**
   Tobiasz Kaciuczyk, Tomasz Korga, Jakub Smołka.....19-25
5. **Comparative analysis of Windows Presentation Foundation and Windows Forms**
   Michał Pasztaleniec, Maria Skublewska-Paszkowska.....26-30
6. **Impact of the cloud application programming language on the performance of its implementation in selected serverless environments**
   Krzysztof Bezrąk, Sławomir Przyłucki.....31-36
7. **Application of support vector machine in a traffic lights control**
   Artur Całuch, Adam Cieślikowski, Małgorzata Plechawska-Wójcik.....37-42
8. **Comparison of GreenDao and Room ORM Systems**
   Maciej Lewiński.....43-47
9. **An Assessment of IT Students' Awareness in the Field of Instant Messengers Security**
   Paweł Stręciwilk, Grzegorz Kozieł.....48-54
10. **Methods of determining indicators of similarity of 3D motion**
    Piotr Flisiak, Marcin Kuszyk.....55-58
11. **Perfomance analysis of frameworks Angular and Vue.js**
    Roman Baida, Maksym Andriienko, Małgorzata Plechawska-Wójcik.....59-64
12. **Possibility analysis of applying serious game to learn the first aid procedures**
    Klaudia Zaborek, Małgorzata Plechawska-Wójcik.....65-72
13. **Comparison of the performance of tools for creating a SPA application interface - React and Vue.js**
    Krzysztof Boczkowski, Beata Pańczyk.....73-77
14. **Comparative analysis of view technologies for the Spring application**
    Vadym Borys, Roman Slezenko, Beata Pańczyk.....78-81
15. **Performance analysis of selected tools for building a Single Page Application**
    Yehor Timanovskyi, Małgorzata Plechawska-Wójcik.....82-87
16. **Analysis of selected usability assessment methods in the process of creating web applications**
    Krzysztof Nowak, Daniel Samolej.....88-93
17. **Performance analysis of selected database systems: MySQL, MS SQL, PostgerSQL in the context of web applications**
    Katarzyna Lachewicz.....94-100
18. **Methods of creating realistic spaces – 3D scanning and 3D modelling**
    Aleksandra Salwierz, Tomasz Szymczyk.....101-108

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 14 (2020) 1-7
Wysłane: 2019-07-04
Przyjęte: 2019-07-08

Analiza porównawcza wydajności relacyjnych baz danych MySQL, PostgreSQL, MariaDB oraz H2

Katarzyna Krocz*, Oleksandra Kizun*, Maria Skublewska-Paszkowska
Politechnika Lubelska, Katedra Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. Celem artykułu jest porównanie wydajności baz danych MySQL, PostgreSQL, MariaDB i H2 oraz wybór najlepszej z nich. Zbadana zostanie prędkość wykonywania prostych oraz bardziej złożonych zapytań SQL. Odpowiedź na pytanie, która z nich jest bardziej wydajna, zostanie udzielona na podstawie wyników z przeprowadzonych badań.

Słowa kluczowe: SZBD; relacyjny model; analiza porównawcza

*Autorzy do korespondencji.

Adresy e-mail: oleksandra.kizun@pollub.edu.pl, katarzyna.krocz@pollub.edu.pl

Perfomance analysis of relational databases MySQL, PostgreSQL, MariaDB and H2

Katarzyna Krocz*, Oleksandra Kizun*, Maria Skublewska-Paszkowska
Department of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract. This article will compare the performance of relational databases MySQL, PostgreSQL, MariaDB and H2. The speed of executing will be tested on simple and more complex SQL queries. The tests results will show which database is more efficient.

Keywords: DBMS; relational model; performance analysis

*Corresponding authors.

E-mail addresses: oleksandra.kizun@pollub.edu.pl, katarzyna.krocz@pollub.edu.pl

1. Wstęp

Obecnie istnieje wiele dużych firm i przedsiębiorstw przemysłowych działających w różnych obszarach biznesu, nauki i świadczenia usług, które korzystają z relacyjnych baz danych. W pewnym momencie rozwoju przedsiębiorstwa bazy danych stają się jednym z kluczowych elementów jego infrastruktury. W nich przechowywane są kluczowe informacje, których dostępność wpływa na wydajność pracy większości działów w firmie. Im szybciej baza danych będzie w stanie dostarczyć niezbędnych informacji, tym cenniejsze będą te informacje.

Istnieje wiele różnych baz danych przeznaczonych do wykorzystania w poszczególnych aplikacjach, ale zazwyczaj nie jest łatwo określić, która z nich okaże się lepsza w tych czy innych warunkach wykorzystania. Szeroka gama baz danych stwarza trudności dla programistów w wyborze odpowiedniego systemu, co doprowadziło do napisania tego artykułu. Porównanie często wykorzystywanych przez programistów baz danych MySQL, PostgreSQL, oraz mniej popularnych MariaDB i H2 pozwoli zidentyfikować ich mocne i słabe strony, a także ich przydatność w rozwiązywaniu określonego zadania. Wśród innych baz H2 różni się tym, że jest to baza pamięciowa, wykorzystywana głównie w małych projektach. Dane przechowywane w niej nie są trwałe. Bazy danych odgrywają ważną rolę w aplikacjach, a zły wybór może mieć negatywne konsekwencje w przyszłości, ponieważ trudno jest zmienić bazę danych w procesie działania systemu.

2. Przegląd literatury

Aby lepiej poznać możliwości wybranych systemów bazodanowych, została przeprowadzona analiza artykułów o podobnym temacie. Badanie porównawcze między bazami danych MySQL, PostgreSQL, MariaDB i SQLite dokonał w swoim artykule Yusuf Abubakar [1]. Przeprowadzone badania polegały na mierzeniu czasu wykonania zapytań do operacji odczytu, zapisu i aktualizacji danych. Z wyniku badań, okazało się, że PostgreSQL ma większą wydajność od pozostałych baz danych. W operacji zapisu i aktualizacji danych najlepszą wydajność uzyskał PostgreSQL, a dla odczytu najlepszą okazała się baza MariaDB. W artykule [2] Sasalak Tongkaw i Aumnat Tongkaw przeprowadzili eksperymenty, z których wynika, że MySQL ma znacznie lepszą wydajność niż MariaDB. W kolejnym artykule [3] Meekyung Min porównała wydajność baz danych o otwartym kodzie źródłowym MySQL oraz MariaDB do komercyjnej bazy danych MS-SQL Server. Przeprowadzone badania polegały na mierzeniu czasu wykonania różnych typów zapytań dla liczby rekordów w przedziale od 200 do 500000. Operacje zostały wykonane z wykorzystaniem indeksów i bez nich. Autorzy zauważyli, że przy wykorzystaniu indeksów, wydajność baz danych o otwartym kodzie źródłowym często przewyższa wydajność baz danych komercyjnych w zależności od typu zapytania. Duża liczba rekordów

również wydłuża czas wykonania zapytań w przypadku baz danych MySQL i MariaDB. W artykule [4] Poljak, Pošcić i Jakšić porównali trzy popularne systemy bazodanowe: MySQL, PostgreSQL oraz Oracle 11g. Podobnie jak w poprzednim artykule [3] dla wybranych baz danych został zmierzony czas wykonania zapytań. Zgodnie z wynikami badań szybkość wykonywania jest najlepsza w systemie Oracle, bezpośrednio za nim znajduje się MySQL i trochę wolniejszy jest PostgreSQL.

Z analizy wybranych artykułów wynika, że bazy danych o otwartym kodzie źródłowym mogą zastąpić komercyjne bazy danych, gdy używane są indeksy i liczba rekordów nie jest zbyt duża. Bez wykorzystania indeksów dopuszczalna liczba rekordów ok. 1000000, a z wykorzystaniem indeksów w kilka razy większa.

W literaturze nie znaleziono porównań baz danych przedstawionych w niniejszym artykule.

3. Obiekty badań

3.1. MySQL

MySQL jest to jeden z najpopularniejszych relacyjnych systemów zarządzania bazami danych [5] o otwartym kodzie źródłowym wspierany przez firmę Oracle od 2010 roku [6]. MySQL to darmowy pakiet oprogramowania rozpowszechniany na licencji GNU General Public License [6], ale razem z tym istnieją specjalne płatne wersje przeznaczone do użytku komercyjnego.

MySQL jest uważany za dobre rozwiązanie dla małych i średnich aplikacji. Wchodzi do zestawu pakietu XAMPP i WAMP. Często MySQL jest używany jako serwer, do którego uzyskują dostęp lokalni lub zdalni klienci, ale również może być osadzony w aplikacjach. Jest dostępny na wielu platformach: Windows, Linux, MacOS, Solaris itp. [6] W MySQL obsługiwane są takie podstawowe silniki jak MyISAM, InnoDB, MEMORY oraz Berkeley DB [6], co pozwala na zmianę funkcjonalności narzędzia i wykonywanie przetwarzania danych, przechowywanych w różnych typach tabel. Elastyczność MySQL dostarczana ze wsparciem dużej liczby typów tabel. Na przykład, można wybrać tabele MyISAM, które obsługują wyszukiwanie pełnotekstowe lub InnoDB obsługujące transakcje na poziomie poszczególnych rekordów. Ten system zapewnia łatwość obsługi, elastyczność, skalowalność, a także wysoką wydajność [7]. MySQL został opracowany dla aplikacji webowych w celu zapewnienia szybkiego indeksowania danych w repozytoriach i optymalizacji sekwencyjnego dostępu do danych.

3.2. MariaDB

MariaDB to system zarządzania relacyjnymi bazami danych, opracowany przez twórcę MySQL Michaela „Monty" Wideniusa na początku 2009 roku [8]. Jest rozpowszechniany na bezpłatnej otwartej licencji GNU GPL. MariaDB jest rozwijana i wspierana przez firmę MariaDB Corporation AB i fundację MariaDB Foundation [8]. Według [5] jest to szybko rozwijający się system zarządzania bazami danych, który z każdym rokiem nabywa większej popularności. MariaDB jest dostępna na platformach: Windows, Solaris, Linux, CentOS 5/RedHat 5 [8].

Głównym celem twórców MariaDB jest stworzenie produktu, który jest w pełni kompatybilny z MySQL, ale znacznie ulepszony. Jednym z ulepszeń wprowadzonych w MariaDB jest silnik Aria, który zastąpił MyISAM i w rzeczywistości jest znacznie bardziej wydajny, szybszy i bardziej odporny na awarie. Jeśli oryginalny MyISAM był szybki kosztem rezygnacji z transakcji, co mogło spowodować możliwą utratę danych, to Aria jest zarówno produktywny, jak i bezpieczny. Również MariaDB używa nowego silnika pamięci masowej MyRocks [9], który pozwala na kompresję danych bez utraty prędkości. Został on opracowany we współpracy z Facebookiem, aby umożliwić przetwarzanie większej ilości danych przy mniejszych zasobach.

3.3. PostgreSQL

PostgreSQL jest to system zarządzania relacyjnymi bazami danych rozwijany przez PostgreSQL Global Development Group od 1977 roku [10]. Jest to darmowe oprogramowanie o otwartym kodzie źródłowym. Może obsługiwać obciążenia dla aplikacji na pojedynczą maszynę, jak również usługi internetowe oraz hurtownie danych z wieloma użytkownikami. PostgreSQL działa na wszystkich głównych systemach operacyjnych: macOS Server, Linux, FreeBSD, OpenBSD i Windows [11]. PostgreSQL oferuje wsparcie dla funkcji systemu zarządzania relacyjną bazą danych takich jak możliwość aktualizacji widoków, wyzwalaczy, kluczy obcych oraz funkcji i procedur.

PostgreSQL jest zgodny ze standardem SQL. Wiele funkcji wymaganych przez ten standard jest obsługiwana, czasami z trochę inną składnią lub funkcją. Baza spełnia 160 z 179 obowiązkowych funkcji występujących w standardzie SQL:2011 [11]. Zyskała dobrą reputację dzięki swej architekturze, niezawodności, integracji danych, zestawie funkcji, rozszerzalności oraz zaangażowaniu społeczności. PostgreSQL posiada wiele funkcji, które pomagają programistom w tworzeniu aplikacji oraz administratorom do ochrony integralności danych, oraz odporność na awarie. Dzięki tym cechom PostgreSQL stał się popularnym wyborem dla wielu ludzi i organizacji.

3.4. H2

H2 jest to relacyjny system do zarządzania bazą danych napisany w języku Java, który może być osadzony w aplikacjach Java lub uruchomiony w trybie klient-serwer. Silnik bazy danych napisał Thomas Mueller [12]. Rozwój silnika H2 rozpoczął się w maju 2004 roku, a po raz pierwszy opublikowano go w grudniu 2005 roku [13].

Główną cechą H2 jest obsługiwanie standardu SQL. Interfejs programistyczny aplikacji wykorzystywany do programowania to SQL, JDBC oraz obsługiwany jest sterownik ODCB PostgreSQL działając jako serwer PostgreSQL [13]. Baza danych H2 głównie jest skonfigurowana do przetwarzania danych w pamięci, co oznacza, że dane nie będą przechowywane na dysku. Ze względu na wbudowaną bazę danych nie jest ona wykorzystywana do rozwoju produkcji, ale głównie do wykorzystania w procesie tworzenia i testowania aplikacji.

4. Metoda badań

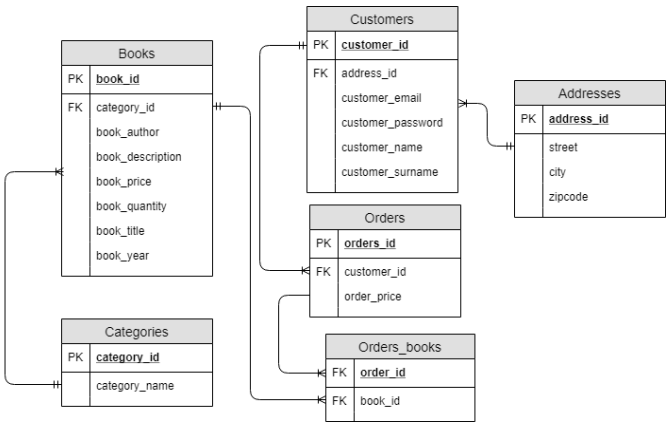
Porównanie wydajności relacyjnych baz danych będzie polegać na przeprowadzeniu operacji dodania, aktualizacji, usuwania i wybierania danych. Każde badanie zostało wykonane 10 razy dla 1, 100, 2500, 5000, 7500 i 10000 rekordów, a następnie obliczony został czas średni. W tabeli 1 przedstawione są zapytania dla poszczególnych badań.

Testy zostały przeprowadzone na komputerze Lenovo Z51-70 z procesorem Intel Core i5-5200U z zainstalowaną pamiecią RAM 8 GB na systemie operacyjnym Windows 10 Home. Do obsługi serwerów baz danych użyto programu WampServer (3.1.7), w którego skład wchodzi serwer baz danych MySQL (5.7.24) oraz MariaDB (10.3.12). Do zarządzania bazą PostgreSQL został użyty program pgAdmin w wersji 3.4. Aplikacja została zaimplementowana w języku Java z wykorzystaniem Spring Framework i biblioteki Hibernate.

Tabela 1. Scenariusze badawcze

Nazwa operacji	Kod SQL
Wstawienie danych	INSERT INTO books (book_author, category_id, book_description, book_price, book_quantity, book_title, book_year, book_id) values (?, ?, ?, ?, ?, ?, ?, ?)
Aktualizacja danych	UPDATE books SET book_author = ”Autor” WHERE book_author = ”Inny Autor”
Usunięcie wszystkich rekordów	DELETE FROM books
Usunięcie wszystkich rekordów z klauzulą WHERE	DELETE FROM books WHERE ‘book_title‘ = ”Test”
Wybieranie danych	SELECT * FROM books
Wybieranie danych z klauzulą WHERE	SELECT * FROM books WHERE book_title = ”Test”
Łączenie danych	SELECT * FROM books b JOIN categories c ON c.category_id = b.category_id
Sortowanie danych	SELECT * FROM books ORDER BY book_author
Grupowanie danych	SELECT COUNT(book_id), book_author, book_title FROM books GROUP BY book_title,book_author

Wszystkie badania zostały przeprowadzone na bazie danych, która została opracowana na potrzeby aplikacji webowej do zarządzania biblioteką osobistą. Schemat tej bazy danych został przedstawiony na rysunku 1.



Rys. 1. Schemat bazy danych

W celu realizacji bazy danych dla biblioteki osobistej zostały zaprojektowane następujące encję:

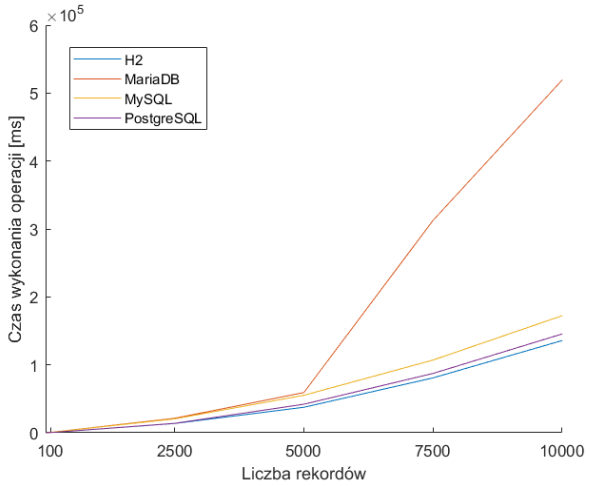
- *Books* – przechowuje pełną informację dotyczącą wybranych książek;
- *Categories* – przechowuje informację dotyczącą kategorii książek;
- *Orders* – przechowuje informację dotyczącą cen książek;
- *Addresses* – adresy zamówień klientów;
- *Customers* – dane o kliencie i jego zamówieniach.

5. Wyniki badań

5.1. Operacja dodania danych

Dla większości systemów bazodanowych operacja wstawiania danych jest jedną z najczęściej wykonywanych, dlatego informacja o prędkości jej wykonywania jest dość istotna. Badanie przedstawione w tym podrozdziale polega na obliczeniu czasów wykonania operacji dodania danych. Aby wykonać operację wstawiania, została wykorzystana klauzula INSERT INTO (tabela 1).

Rysunek 2 przedstawia wykres czasów wykonania operacji dodania rekordów do bazy danych. Na wykresie wyraźnie widać, że czasy wykonania zapytania dla MySQL, PostgreSQL i H2 są bardzo zbliżone. Wśród tych trzech baz danych najlepsze wyniki wykazała H2. Warto zauważyć, że ich czas wykonania zapytań rośnie liniowo wraz ze wzrostem liczby rekordów, czego nie można powiedzieć o MariaDB. Czas wykonania zapytań tej bazy rośnie liniowo do połowy wykresu i znacznie zwiększa się w drugiej połowie. MariaDB wykazała podobne do pozostałych baz danych wyniki dla liczby rekordów poniżej 5000. Natomiast dla liczby rekordów powyżej 5000 MariaBD wykazała bardzo długi czas wykonania zapytań w porównaniu do MySQL, H2 i PostgreSQL. W przedziale od 7500 do 10000 rekordów czas wykonania zapytań dla MariaDB w 2,8 dłuższy niż dla H2. Taka niska wydajność MariaDB może być związana z wykorzystaniem silnika InnoDB. Przy domyślnych ustawieniach bufor dziennika jest zapisywany w pliku dziennika przy każdym zatwierdzeniu transakcji. Te działania mogą znacznie spowolnić dodanie lub aktualizację danych.

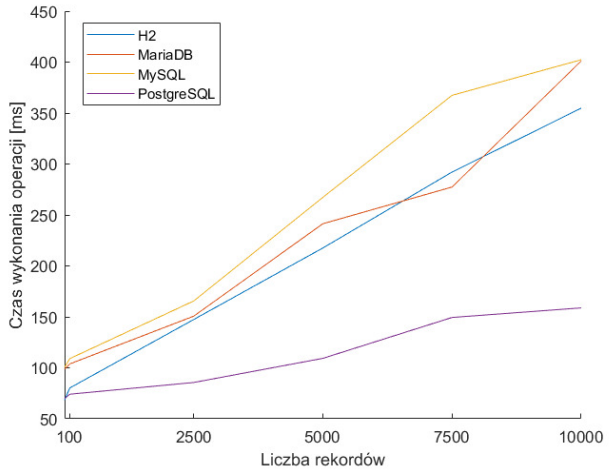


Rys. 2. Średnie czasy wykonania operacji dodania danych

5.2. Operacja aktualizacji danych

Kolejne badanie polega na obliczeniu czasów wykonania operacji aktualizacji danych. Aby dokonać aktualizacji danych wykorzystywana jest klauzula UPDATE (tabela 1).

Rysunek 3 przedstawia wykres średnich czasów dla operacji aktualizacji danych. Na wykresie można zaobserwować, że najlepsze czasy wykonania zapytań uzyskała baza PostgreSQL. Warto zauważyć, że czas wykonania zapytania w PostgreSQL dla 1 rekordu różni się od 10000 tylko 2,3 razy. Bazy MariaDB i H2 uzyskały podobne wyniki, jednakże dla 10000 rekordów H2 okazała się szybsza o 11,5%. MySQL wykazała najgorsze wyniki, ale dla 10000 rekordów jej czas jest bardzo zbliżony do MariaDB. PostgreSQL uzyskała znacznie lepsze czasy wykonania zapytań w porównaniu do MariaDB, MySQL i H2. Dla 10000 rekordów różnica między MySQL a PostgreSQL wynosi 243,5 milisekund.

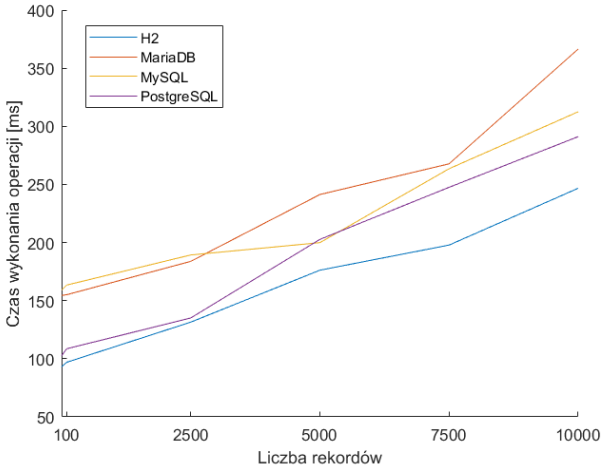


Rys. 3. Średnie czasy wykonania operacji aktualizacji danych

5.3. Operacje usunięcia danych

W tym podrozdziale przedstawione jest badanie, które polega na obliczeniu czasów wykonania operacji usunięcia danych. Do wykonania operacji usunięcia została wykorzystana klauzula DELETE FROM (tabela 1). W wyniku zapytania została usunięta tabela Books.

Rysunek 4 przedstawia wykres średnich czasów dla operacji usunięcia danych. Na otrzymanym wykresie można zaobserwować, że czas usunięcia wszystkich rekordów z tabeli dla poszczególnych baz danych wzrasta liniowo odpowiednio do zwiększenia liczby rekordów w tabeli. Najlepszą w tym przypadku okazała się baza danych H2, a najgorszą MariaDB. Średni czas wykonania zapytań w bazie danych H2 dla 10000 rekordów różni się od MariaDB o 119 ms, co stanowi dość dużą różnicę. MySQL i PostgreSQL wykazali zbliżone wyniki dla liczby rekordów powyżej 5000, natomiast ich czasy znacznie różnią się dla liczby rekordów poniżej 5000.

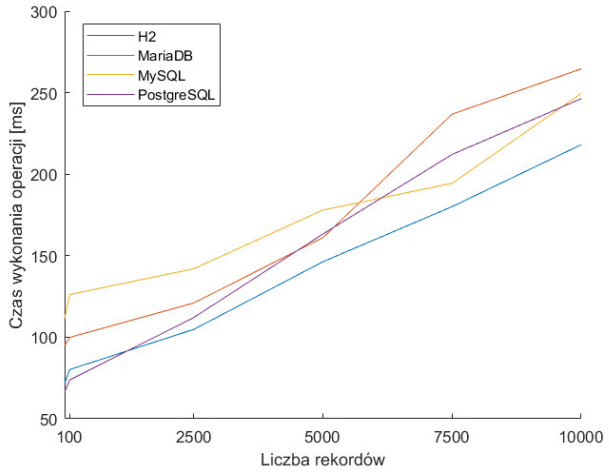


Rys. 4. Średnie czasy wykonania operacji usunięcia danych

Kolejne badanie polega na obliczeniu czasów wykonania operacji usunięcia danych z klauzulą WHERE (tabela 1).

Ten test został przeprowadzony w celu sprawdzenia szybkości wykonania zapytania z podaniem dodatkowego warunku. Wyniki tego badania nie pokrywają się z oczekiwaniami. Pod kątem wydajności baz danych są one podobne do wyników poprzedniego badania. Natomiast czasy wykonania zapytań są krótsze w przypadku wykorzystania klauzuli WHERE. Na przykład, dla MariaDB czas wykonania zapytań został skrócony na ok. 100 milisekund. Szybsze wykonania zapytania może być spowodowane tym, że w procesie wykonania zapytania najpierw jest sprawdzony warunek wyboru i zatem wyświetlone są dane spełniające określony warunek.

Rysunek 5 przedstawia wykres średnich czasów dla operacji usunięcia danych z klauzulą WHERE. Na wykresie widać, że najlepsze wyniki uzyskała baza H2. Baza PostgreSQL i MariaDB uzyskały podobne wyniki, jednakże PostgreSQL jest trochę szybsza. MySQL uzyskała lepsze wyniki od MariaDB i PostgreSQL w przedziale od 6000 do 10000 tysięcy rekordów, ale dla mniejszej liczby rekordów jest ona wolniejsza od pozostałych baz danych. Różnica czasów wykonania zapytania pomiędzy bazami H2 a PostgreSQL dla 1 rekordu wynosi 45,1 ms.

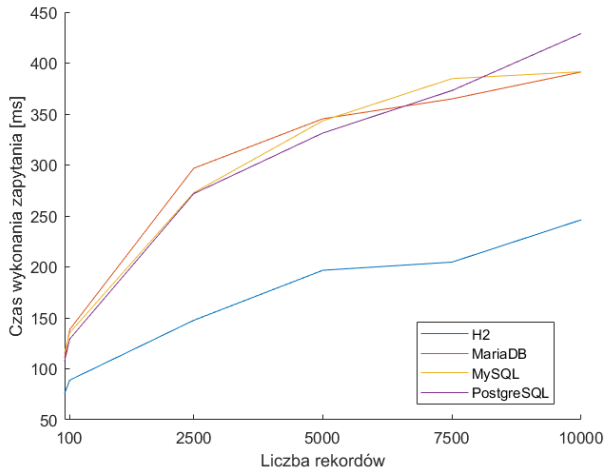


Rys. 5. Średnie czasy wykonania operacji usunięcia danych z klauzulą WHERE

5.4. Operacje wybierania danych

Kolejne badanie jest to operacja wybrania wszystkich danych z jednej tabeli Books (tabela 1).

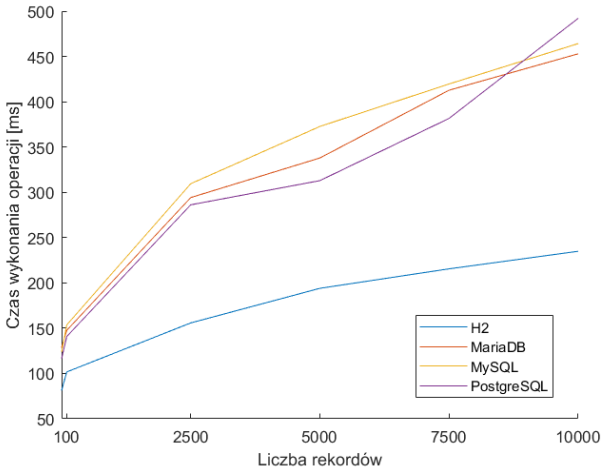
Rysunek 6 przedstawia wykres średnich czasów dla operacji wybrania danych. Z wykresu można zauważyc, że najlepszy czas uzyskała baza H2. Bazy MariaDB, MySQL oraz PostgreSQL uzyskały podobne czasy dla danej operacji, jednakże PostgreSQL dla 10000 rekordów zanotował zwiększony czas wykonania operacji o ok. 38 milisekund w porównaniu do MariaDB i MySQL. Baza H2 ma znaczącą przewagę w czasie wykonania operacji. Różnica pomiędzy H2 a baza MariaDB dla 10000 rekordów wynosi 145,1 milisekund.



Rys. 6. Średnie czasy wykonania zapytania dla operacji wybierania danych

Następne badanie polega na obliczeniu czasu wykonania operacji wybrania danych z klauzulą WHERE, która w podanej operacji do analizy czasu, będzie ograniczać wynik zapytania do książek o wybranym tytule.

Z rysunku 7 można zaobserwować, że tak jak w poprzednim badaniu najlepszy czas uzyskała baza H2, a następnie bazy MariaDB, MySQL oraz PostgreSQL uzyskały podobne czasy. Tak jak na poprzednim wykresie jest widoczna różnica pomiędzy bazą H2 a pozostałymi bazami pod względem czasu wykonania zapytania. Różnica pomiędzy bazą H2 a MariaDB dla 10000 rekordów wynosi 218 milisekund. Najwyższy wzrost wykonania operacji wybrania danych z klauzulą i bez klauzuli uzyskał MySQL. Zapytania wykonywały się średnio o 11,6% dłużej. Najmniejszy wzrost jest w przypadku bazy H2 gdzie zapytanie wykonywało się średnio o 4,4% dłużej.

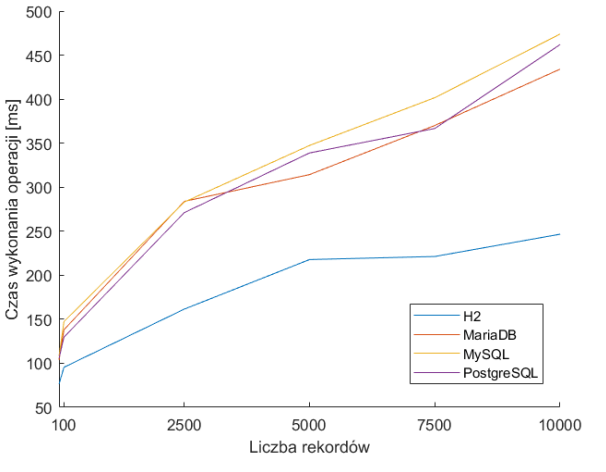


Rys. 7. Średnie czasy wykonania zapytania dla operacji wybierania danych z klauzulą WHERE

5.5. Operacja łączenia danych

Następnym zapytaniem, wykonanym w ramach badania nad wydajnością baz danych, jest operacja wybierania danych z łączeniem tabel. Tabele zostały złączone poprzez identyfikator kategorii (tabela 1).

Rysunek 8 przedstawia wykres czasu wykonania operacji wybrania danych ze złączeniem tabel. Z wykresu można zauważyć, że najlepsze czasy uzyskała baza H2. Dla MySQL, MariaDB i PostgreSQL czasy są bardzo zbliżone jednak MariaDB i PostgreSQL uzyskują nieznacznie lepszy czas od MySQL. Ponownie jest widoczna różnica w czasie wykonania operacji pomiędzy H2 a pozostałymi bazami. Różnica czasów dla bazy H2 a drugą co do wydajności bazą MariaDB dla 10000 wynosi 187,5 milisekundy.

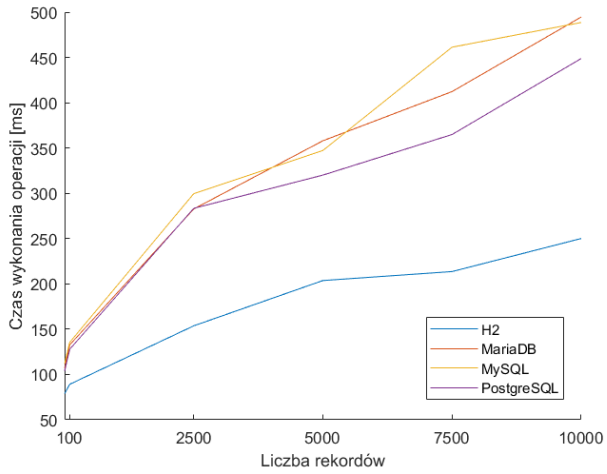


Rys. 8. Średnie czasy wykonania zapytania dla operacji złączenia tabel

5.6. Operacja sortowania danych

Kolejnym badaniem, którego będzie dotyczyła analiza długości czasu wykonywania zapytania, jest operacja sortowania danych. Do sortowania jest używana klauzula ORDER BY (tabela 1).

Rysunek 9 przedstawia wykres czasu wykonania zapytania dla operacji sortowania danych. Można tam zaobserwować, że najkrótszy czas wykonania operacji uzyskała baza H2, a pozostałe bazy osiągają o wiele wyższe wyniki. Różnica pomiędzy H2 a PostgreSQL dla 10000 rekordów wynosi 198,7 ms. Na drugim miejscu pod względem szybkości wykonania zapytania jest baza PostgreSQL. Bazy MariaDB i MySQL uzyskują bardzo zbliżone czasy wykonania zapytania.

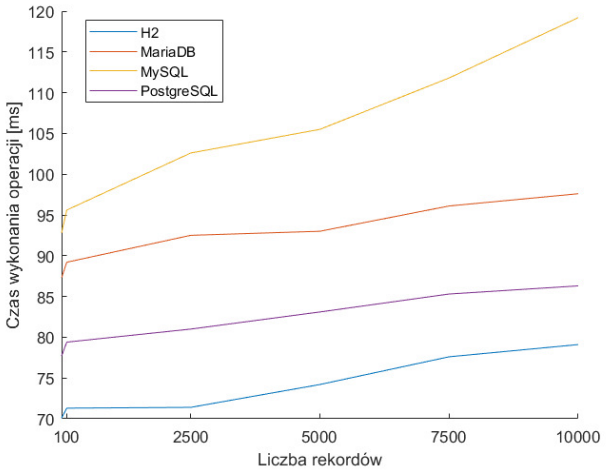


Rys. 9. Średnie czasy wykonania zapytania dla operacji sortowania danych

5.7. Operacja grupowania danych

Ostatnią operacją w analizie wydajności systemów zarządzania baz danych będzie grupowanie danych. Za pomocą klauzuli GROUP BY można utworzyć grupy rekordów o tej samej wartości. Klauzula ta jest często stosowana z funkcjami agregującymi. Dla zbadania wydajności zostało utworzone zapytanie z funkcją COUNT(), która zwraca liczbę wierszy o określonym kryterium. Następnie w zapytaniu zostały wybrane kolumny book_author, book_title z tabeli Books i grupowanie zostało utworzone na podstawie tych kolumn (tabela 1).

Na rysunku 10 zaprezentowany jest wykres czasu wykonania zapytania dla operacji grupowania danych. Z wykresu wynika, że najlepszy czas uzyskała baza H2, a następnie po niej jest PostgreSQL i MariaDB. Najgorzej (z najdłuższym czasem) poradziła sobie baza MySQL. Im większa jest liczba rekordów, tym bardziej wydłuża się czas wykonania operacji. Różnica czasów pomiędzy bazami H2 i MySQL dla 1 rekordu wynosi 15,1 ms a dla 10000 rekordów 32,9 ms.



Rys. 10. Średnie czasy wykonania zapytania dla operacji grupowania danych

6. Wnioski

W tym artykule zostały porównane cztery systemy bazodanowe: MySQL, MariaDB, PostgreSQL i H2. Uzyskano następujące wyniki:

1. MySQL wykazała słabą wydajność podczas pracy z dużą ilością danych;
2. Najlepsze średnie czasy wykonania wszystkich operacji (za wyjątkiem aktualizacji) uzyskała baza H2. Dla operacji sortowania, złączenia, wybierania różnica w czasie dość istotna ok. 100 ms;
3. Wyniki otrzymane dla MySQL, MariaDB oraz PostgreSQL są zbliżone dla operacji wybierania, złączenia i usuwania danych, jednakże PostgreSQL okazał się trochę szybszy od pozostałych dwóch;
4. PostgreSQL wykazał najlepsze wyniki dla operacji aktualizacji.

Warto zauważyć, że szybkość wykonywania operacji zależy od typu zapytania oraz liczby rekordów.

Uzyskane wyniki są dość oczekiwane w przypadku MySQL, MariaDB i PostgreSQL. Odpowiadają one wynikom badań przedstawionych w pracach innych autorów [1][2][3][4]. Natomiast możliwości bazy H2 nie były rozważane i omówione w innych pracach. Mimo tego, że baza H2 jest mniej popularna, ponieważ można z niej korzystać tylko na etapie tworzenia aplikacji, jej wydajność okazała się znacznie lepsza od pozostałych baz. Jest to spowodowane tym, że tryb wbudowany bazy danych jest szybszy od bazy klient-serwer [13].

Literatura

[1] Y. Abubakar, Benchmarking popular open source RDBMS: a performance evaluation for it professionals, International Journal of Advanced Computer Technology (IJACT), 2014.

[2] S. Tongkaw, A. Tongkaw, A Comparison of Database Performance of MariaDB and MySQL with OLTP Workload, IEEE Conference on Open Systems (ICOS), 2016.

[3] M. Meekyung, Experiments of Search Query Performance for SQL-Based Open Source Databases, International Journal of Internet, 2018.

[4] R. Poljak, P. Pošcić, D. Jakšić, Comparative Analysis of the Selected Relational Database Management Systems, MIPRO, 2017.

[5] Ranking SZBD według ich popularności, https://db-engines.com/en/ranking, [20.03.2019].

[6] Opis bazy MySQL, https://pl.wikipedia.org/wiki/MySQL, [20.03.2019].

[7] Dokuntacja MySQL, https://dev.mysql.com/doc/, [20.03.2019].

[8] Opis bazy MariaDB, https://en.wikipedia.org/wiki/MariaDB, [20.03.2019].

[9] Silnik MyRocks, https://mariadb.com/kb/en/library/about-myrocks-for-mariadb/, [20.03.2019].

[10] J. D. Drake , J. C. Worsley, Practical PostgreSQL, O'Reilly Media, Inc., 2002.

[11] Dokumentacja PostgreSQL, https://www.postgresql.org/docs/11/index.html, [20.03.2019].

[12] Opis bazy H2, https://en.wikipedia.org/wiki/H2_(DBMS), [20.03.2019].

[13] Strona bazy H2, http://www.h2database.com/html/main.html, [20.03.2019].

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 14 (2020) 8-13
Wysłane: 2019-09-20
Przyjęte: 2019-09-30

Porównanie aplikacji wspierających zastosowanie metodyk zwinnych w wytwarzaniu oprogramowania

Tomasz Bławucki*, Siarhei Ramanovich, Maria Skublewska-Paszkowska
Politechnika Lubelska, Katedra Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. Artykuł przedstawia porównanie pod względem wymagań sprzętowych aplikacji wspierających wprowadzanie metodyk zwinnych do procesu wytwarzania oprogramowania. Przedmiotem badań były popularne aplikacje mobilne i internetowe wspomagające procesy Agile w przedsiębiorstwach. W celu określenia znaczenia poszczególnych wymagań technicznych dla użytkowników, przeprowadzono serię eksperymentów badawczych opartych na scenariuszach typowego i brzegowego użytkowania badanych systemów. Na potrzeby przeprowadzonej analizy została dodatkowo opracowana aplikacja wspierająca proces zwinnego wytwarzania oprogramowania. Wyniki pomiarów były rejestrowane za pomocą specjalistycznych narzędzi monitorujących pracę systemu i profilujących działanie przeglądarki internetowej. Rezultaty prac badawczych przedstawiono w formie tabel.

Słowa kluczowe: agile; aplikacje mobilne; aplikacje internetowe; wymagania sprzętowe

*Autor do korespondencji.

Adres e-mail: tomasz.blawucki@pollub.edu.pl

Applications supporting utilization of agile methods in software development process

Tomasz Bławucki*, Siarhei Ramanovich, Maria Skublewska-Paszkowska
Department of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract. The article presents a comparison in terms of hardware requirements of applications that supports the agile software development processes. For research purposes, popular mobile and internet applications supporting agile software development were chosen. In order to determine the significance of individual technical requirements for end-users, a series of research experiments, based on scenarios of typical and boundary use was conducted. In addition to research, the application supporting agile software development process was implemented. The results of research were recorded by specialized monitoring and profiling tools. The results of performed work are presented in tabular form.

Keywords: agile; mobile applications; web applications; hardware requirements

*Corresponding author.

E-mail address: tomasz.blawucki@pollub.edu.pl

1. Wstęp

Na przestrzeni lat charakterystyka procesu wytwarzania oprogramowania podlegała ewolucji, czy to ze względu na postęp technologiczny, uzależnienie globalnego rynku od narzędzi informatycznych, czy też zmian zachodzących w samych zespołach projektowych. Niewątpliwie na dzień dzisiejszy ciężko jest sobie wyobrazić szeroko pojęty biznes bez systemów i produktów informatycznych. Tempo globalnej cyfryzacji rynków bezpośrednio wymusza na przedsiębiorstwach informatycznych odpowiednie przyspieszenie wytwarzania oprogramowania, przy jednoczesnym zachowaniu wymaganej przez inwestorów jakości.

Ze względu na często wysoki stopień skomplikowania komercyjnych rozwiązań IT, wymaganego przez klientów, konieczne jest zatrudnienie odpowiednio dużego zespołu projektowego dla ich realizacji. Liczne kadry specjalistów z różnych dziedzin IT realizujących wspólny projekt za sobą obciążenie w postaci utrudnionej komunikacji w grupie oraz konieczności odpowiedniej organizacji pracy, pozwalającej na efektywne wykorzystanie potencjału produkcyjnego. W celu ułatwienia procesu organizacji pracy grup projektowych, przedsiębiorstwa informatyczne często korzystają z narzędzi w formie specjalistycznego oprogramowania przeznaczonego do wspomagania zwinnego zarządzania procesami wytwarzania oprogramowania.

Wybór systemu wspierającego proces zwinnego wytwarzania oprogramowania, odpowiedniego dla kontekstu projektu i specyfiki pracy zespołu, nie jest trywialny. Decydent powinien wziąć pod uwagę wiele czynników związanych z różnorodnymi aspektami pracy firmy i klienta. Zadanie jest dodatkowo utrudnione przez brak jasno sprecyzowanych zasad i wytycznych doboru takiego oprogramowania. Jedynie kilka publikacji bezpośrednio opisuje proces doboru narzędzi mających wspierać procesy *agile.* Ogólnodostępne blogi eksperckie, czy strony o tematyce *agile* w definicji procedury postępowania przy wyborze dedykowanego oprogramowania skupiają się głównie na dostarczanych funkcjonalnościach [14, 15, 16, 18]. Warto również dodać, że niektóre badania w zakresie dopasowania narzędzi do wprowadzania metodyk zwinnych zostały przeprowadzone na zlecenie firm lub korporacji, co potwierdza istnienie potrzeby eksploracji tego obszaru [1, 2].

Przegląd literatury dotyczącej wymagań technicznych systemów wspierających organizację pracy zespołów projektowych, jako kryterium doboru narzędzi *agile*, wykazał

brak publikacji o podobnej tematyce. Niniejsze opracowanie próbuje wypełnić tą niszę.

W przeciągu ostatnich trzech dekad można zauważyć płynne przejście w kontekście metodyk wytwarzania oprogramowania z metody kaskadowej (*ang.waterfall)* do metod zwinnych [4, 5], przy czym podejścia te są diametralnie różne. W przeszłości znakomita większość systemów IT powstawała w podejściu kaskadowym [6], dzisiaj uznawanym jako tradycyjne. Do końca lat 90-tych kaskadowe wytwarzanie oprogramowania było wystarczająco wydajne [13]. Metoda ta jednak około 20 lat temu przestała spełniać rosnące wymagania środowiska IT [12]. Obecnie, według ankiety przeprowadzonej przez J. Jeremiah [3], metody zwinne są podstawowym podejściem przy procesie wytwórczym systemów informatycznych. Przełom nastąpił w latach 2008-2010, w których wiele znaczących przedsiębiorstw zauważyło zalety *agile* i wdrożyło nowatorskie metodyki we własnych zespołach projektowych, pomimo wad i trudności takich jak: ograniczone planowanie projektów, ograniczenia finansowe, czy wymuszone częste spotkania z docelowym odbiorcą produktu [4].

Zazwyczaj metodyki z rodziny metodyk *agile* nie stanowią kompletnej alternatywy dla tradycyjnych metod zarządzania procesem wytwarzania oprogramowania. Ich umiejętne wykorzystanie pozwala jednak na zastąpienie klasycznego podejścia w tym kontekście [7, 8].

U. Kelter i inni [9] już w 2003 roku przedstawili kluczowe różnice pomiędzy metodykami z rodziny *agile,* a tradycyjnymi podejściami do zarządzania procesami wytwarzania oprogramowania. W badaniach wykorzystano komercyjne doświadczenia praktyków oraz rezultaty z innych, podobnych opracowań. Istotne ze względu na aplikacje wspierające wprowadzanie metodyk zwinnych było zestawienie najważniejszych cech i funkcjonalności wynikających ze specyfikacji poszczególnych metodyk. Obszary, w których zastosowanie podejścia *agile* może przynieść korzyść według autorów to [9]:

- komunikacja, koordynacja i kooperacja współpracowników w obrębie zespołów projektowych;
- bieżące planowanie prac nad realizowanym projektem informatycznym;
- umożliwienie oceny jakości produktu oraz rezultatów pośrednich przez zewnętrznych referentów;
- wytwarzanie i dokumentacja - skupienie się na utrzymaniu jakości produktu, dokumentacja jako integralnej części oprogramowania;
- zarządzanie konfiguracjami projektu - umożliwienie szybszego reagowania na zachodzące zmiany wymagań;
- automatyzacja prostych procesów - generowanie raportów oraz sprawdzanie integralności modułów aplikacji w dużych systemach informatycznych (ciągła integracja).

Wprowadzenie dedykowanych narzędzi *agile*, według autorów, ma niebagatelne znaczenie dla poprawy wydajności zespołów projektowych oraz wzrostu prawdopodobieństwa pomyślnego zakończenia projektów informatycznych. Jako przeciwwagę dla zalet wykorzystania rozwiązań do zwinnego zarządzania projektami IT, przytacza się trudność konfiguracji oprogramowania do specyfiki projektów oraz adaptacji samych metodyk do organizacji procesu wytwórczego oprogramowania [9].

W 2012 roku G. Azizyan i inni [2] przeprowadzili szczegółową analizę różnorodnych aspektów pracy i struktury przedsiębiorstwa, w celu doboru odpowiedniego narzędzia pozwalającego na skuteczne zwinne zarządzanie wymaganiami oraz optymalizację procesu realizacji projektów. Studium podzielono na dwie główne części: przegląd najważniejszych funkcjonalności i cech oprogramowania takich jak elastyczność i możliwość rozbudowy oraz obserwacja procesów wewnętrznych firmy w celu specyfikacji faktycznych potrzeb odnośnie docelowego narzędzia. Według autorów podział procesu wyboru narzędzia był konieczny do podjęcia właściwej decyzji. Etap pierwszy miał na celu usystematyzowanie faktycznych potrzeb wynikających ze specyfiki procesów oraz środowisk, w których zostały zaadaptowane metodyki zwinne. Jako rezultat analizy otrzymano listę 21 cech systemu, kluczowych z punktu widzenia charakterystyki przedsiębiorstwa, dalej wykorzystanych przy ocenie rozwiązań. Kryteriami wyboru rozwiązań podlegających dalszemu, szczegółowemu badaniu były [2]:

- zainteresowanie przedstawicieli przedsiębiorstwa danym rozwiązaniem;
- popularność rozwiązania;
- wsparcie dla metodyk zwinnych wykorzystywanych w przedsiębiorstwie;
- możliwości integracji z innymi rozwiązaniami;
- licencja oprogramowania;
- wspierane platformy systemowe.

Interesujące pod względem możliwości oceny również współcześnie dostępnych aplikacji mogą okazać się cechy wyszczególnione przez J. Cabot'a i innych [10]. Najistotniejsze z nich to:

- uniwersalność ze względu na dobór metodyki zarządzania;
- docelowa grupa odbiorców;
- rozmiar zespołów;
- rodzaj licencji oprogramowania oraz jej koszt;
- lokalizacja oprogramowania - czy serwer aplikacji znajduje się na osprzęcie producenta, czy w środowisku klienckim;
- integracja oprogramowania z zewnętrznymi narzędziami;
- możliwość adaptacji aplikacji do zachodzących zmian w obszarze wytwarzania oprogramowania.

M. Taheri i S. MosoudSadjadi [11] w swoim opracowaniu określili następujące cechy oprogramowania do wspierania procesów zwinnych, mające kluczowe znaczenie dla przedsiębiorstw IT. Są to [11]:

- elastyczność systemów pod kątem wymagań przedsiębiorstw;
- łatwość użycia narzędzi przez członków zespołów projektowych;
- rodzaj/rozmiar docelowego przedsiębiorstwa;
- cena oraz zasady licencjonowania oprogramowania;
- wsparcie ze strony producenta;

- funkcjonalność systemów.

Dodatkowo istotnym aspektem są zasady licencjonowania rozwiązań. Autorzy dokonali podziału rozwiązań na oprogramowanie płatne i otwarto źródłowe.

Pomimo dogłębnego przeglądu literatury, nie znaleziono artykułów naukowych, w których jako kryterium wyboru aplikacji do zwinnego zarządzania zespołami projektowymi, odniesiono się do ich wymagań sprzętowych.

2. Aplikacja do zwinnego zarządzania informatycznym zespołem projektowym

Celem dopełnienia niniejszego opracowania autorzy zdecydowali się na projekt i implementację dedykowanego narzędzia *agile*. Zrealizowana aplikacja docelowo ma usprawniać pracę zespołów projektowych w branży IT w kontekście zwinnego wytwarzania oprogramowania, poprzez ułatwienie wprowadzenia metodyk zwinnych do projektu oraz kontroli poprawności ich wykorzystywania. Dodatkowo system ma stanowić wsparcie kierownika zespołu programistów między innymi poprzez możliwość zdalnego przydzielania zadań poszczególnym pracownikom oraz dokumentację postępu prac.

Rozwiązanie dodatkowo ma zespolić wybrane funkcjonalności obecne w niezależnych, istniejących już na rynku rozwiązaniach. Pozwoli to na ograniczenie liczby narzędzi koniecznych do skutecznego zarządzania pracą zespołów. Rozwiązanie można podzielić na następujące moduły funkcjonalne:

- moduł ewidencji i autoryzacji użytkowników;
- moduł wiadomości i notyfikacji;
- moduł ewidencji zadań;
- moduł ewidencji notatek użytkownika;
- moduł ewidencji i zarządzania grupami projektowymi;
- moduł ewidencji i zarządzania zdarzeniami grupowymi;
- moduł administracyjny.

System informatyczny przyjął formę aplikacji mobilnej działającej na urządzeniach wyposażonych w system operacyjny Android w wersji powyżej 6.0 (Marshmallow) włącznie. Architektura systemu została oparta na modelu klient-serwer, kanał komunikacji między interfejsem klienckim a serwerem został zrealizowany w postaci wymiany dokumentów JSON poprzez REST API. Technologie wybrane do implementacji części serwerowej stanowią: język Python w wersji 3.7 przy wykorzystaniu frameworku web Django w wersji 2.1.7 oraz modułu Django REST Framework. Rolę serwera WSGI pełni instancja Gunicorn w wersji 19.9.0 przeznaczona dla środowisk Unix. Aplikacja kliencka została zrealizowana głównie jako rozwiązanie mobilne, jednak ze względu na pracę administratorów systemu wprowadzono również panel administratora, dostępny poprzez przeglądarkę www.

Na system do zarządzania relacyjnymi bazami danych wybrano PostgreSQL. Operacje na bazie danych (BD) są realizowane poprzez warstwę ORM dostarczoną wraz z frameworkiemDjango. Serwer doskonale nadaje się do konteneryzacji i skalowania, odpowiednio do obciążenia zapytaniami.

System w założeniu stanowi bazę do dalszej rozbudowy. Ze względu na ograniczone zasoby, rozwiązanie nie stanowi konkurencji dla obecnych na rynku podobnych narzędzi, a jedynie przedstawia jedną z możliwych ścieżek realizacji takiego systemu.

3. Metodyka badawcza

Hipotezą badawczą postawioną w niniejszym artykule było określenie czy wymagania sprzętowe części klienckich systemów informatycznych wspierających zwinne wytwarzanie oprogramowania mogą stanowić jedno z kryteriów wyboru takiego rozwiązania przez przedsiębiorstwa IT. Hipotezę rozpatrzono w dwóch dziedzinach: wydajności aplikacji internetowych oraz dedykowanych aplikacji mobilnych.

Badania pozwoliły na określenie narzutu sprzętowego generowanego przez poszczególne rozwiązania. Miały one formę serii eksperymentów polegających na wielokrotnym powtórzeniu scenariuszy badawczych,przy wykorzystaniu kolejnych aplikacje do zwinnego zarządzania procesem wytwarzania oprogramowania i równoczesne monitorowanie zużycia zasobów urządzenia gospodarza.Każdy scenariusz powtórzono co najmniej pięciokrotnie, celem minimalizacji obciążenia wyników przez wartości odstające i uzyskania bardziej reprezentatywnych wyników. Wskaźnikami technicznymi pracy aplikacji były:

1) dla aplikacji mobilnych:
- zużycie pamięci RAM;
- transfer sieciowy;
- czas uruchomienia i reakcji;
- wykorzystanie procesora;
- zajętość przestrzeni trwałej urządzenia.

2) dla aplikacji internetowych:
- zużycie pamięci RAM;
- zużycie pamięci CACHE;
- czas ładowania strony głównej (bez użycia pamięci CACHE);
- czas ładowania strony głównej (z użyciem pamięci CACHE);
- czas budowania widoku strony (bez użycia pamięci CACHE);
- czas budowania widoku strony (z użyciem pamięci CACHE);
- objętość załadowanej strony (bez użycia pamięci CACHE);
- objętość załadowanej strony (z użyciem pamięci CACHE);
- zużycie procesora (%);
- liczba pobranych plików (bez użycia pamięci CACHE);
- liczba pobranych plików (z użyciem pamięci CACHE).

Do badania poszczególnych wskaźników wymagań sprzętowych aplikacji posłużyły scenariusze 1 – 4 opisane w tabeli nr 1.

Tabela. 1. Scenariusze badawcze wykorzystane do badania poszczególnych wskaźników wymagań sprzętowych aplikacji

Scenariusz nr 1		
Wymagania	**Przebieg eksperymentu**	**Cele wykonania scenariusza**
Użytkownik uzyskał dostęp do systemu, posiada aktywne konto, jest członkiem odpowiedniej grupy projektowej, obecne uprawnienia pozwalają na aktywną pracę w obrębie wybranego projektu.	Po zalogowaniu użytkownik wybiera projekt do którego został przypisany; dodaje do projektu nowe zadania, przegląda tablicę Scrum/Kanban, edytuje jej wybrane elementy, zmienia status zadań, dokonuje innych działań dla pracownika o podstawowych uprawnieniach.	Dzięki wykonaniu scenariusza, narzędzia profilujące i monitorujące parametry pracy systemu dostarczają informacji na temat średniego wykorzystania zasobów urządzenia przez aplikację takich jak: zużycie procesora, zajętość pamięci RAM czy wykorzystanie sieci.
Scenariusz nr 2		
Wymagania	**Przebieg eksperymentu**	**Cele wykonania scenariusza**
Użytkownik posiada aktywne konto w systemie oraz uprawnienia kierownika zespołu lub projektu, może zarządzać danymi istniejących projektów i tworzyć nowe projekty.	Użytkownik po zalogowaniu do aplikacji tworzy nowy projekt oraz wprowadza podstawowe, informacje o nim (nazwa, opis, członkowie i terminy realizacji). Następnie zgodnie z dokumentacją aplikacji tworzy nową tablicę Scrum/Kanban. Użytkownik dodaje nowe zadania do tablicy i przypisuje do wybranych podwładnych.	Dzięki wykonaniu scenariusza, rozszerzono zbiór pomiarów odnoszących się do wymagań technicznych pracy aplikacji takich jak: zużycie procesora, zajętość pamięci RAM czy wykorzystanie sieci. Przetestowano także stabilność pracy aplikacji oraz określono ogólne odczucia związane z użytkowaniem aplikacji.
Scenariusz nr 3		
Wymagania	**Przebieg eksperymentu**	**Cele wykonania scenariusza**
Użytkownik posiada dostęp do aplikacji, nie musi posiadać aktywnego konta.	Użytkownik wielokrotnie uruchamia i wyłącza aplikację.	Przy każdorazowym powtórzeniu sekwencji włączenie-> wyłączenie aplikacji dokonywany jest pomiar czasu potrzebnego na załadowanie aplikacji, pozwalający na zdefiniowanie komfortu korzystania z niej. Dodatkowo eksperyment pozwala na sprawdzenie stabilności pracy aplikacji oraz pomiar zajętości pamięci RAM oraz skrajnego wykorzystania czasu procesora.
Scenariusz nr 4		
Wymagania	**Przebieg eksperymentu**	**Cele wykonania scenariusza**
Użytkownik posiada dostęp do aplikacji oraz podstawowe uprawnienia pozwalające na swobodną nawigację w systemie i wykonywanie rutynowych działań w obrębie projektu/grupy projektowej.	Użytkownik loguje się do systemu i wybiera dowolny projekt, w którym uczestniczy. W obrębie projektu swobodnie nawiguje, przegląda dostępne informacje, zmienia status przypisanych do siebie zadań oraz wykonuje inne, dostępne akcje. Działania mają symulować rutynową pracę z aplikacją w ciągu około pięciu minut na pojedyncze powtórzenie.	W danym badaniu istotne jest zachowanie określonego okresu czasowego korzystania z aplikacji pozwalającego ocenić ją pod względem niezawodności i stabilności pracy. Dodatkowo dokonano pomiaru średniego wykorzystania czasu procesora, pamięci RAM oraz transferu sieciowego w czasie.

W tabeli 2 zaprezentowano specyfikację techniczną urządzenia mobilnego wykorzystanego do badań parametrów pracy aplikacji mobilnych. W tabeli 2 przedstawiono specyfikację techniczną jednostki komputerowej wykorzystanej do badań aplikacji internetowych.

Tabela. 2. Specyfikacja techniczna urządzenia mobilnego wykorzystanego do przeprowadzenia badań

Urządzenie	LG Nexus 5
CPU	QualcommSnapdragon 800@ 2.26GHz
RAM	1.5 GB LPDDR3
Dysk	16GB FLASH
System operacyjny	Android 6.0.1 Marshmallow
Sieć	Download 25 Mb/s, Upload 22 Mb/s

Tabela 3. Specyfikacja techniczna jednostki komputerowej wykorzystanej do przeprowadzenia badań

Urządzenie	Laptop ASUS N55JM
CPU	Intel(R) Core(TM) i7-4710HQ CPU @ 2.50GHz
RAM	8GB DDR3 1600MHz
Karta graficzna	NVIDIA GeForce GTX 860M
Dysk	SSD 120GB
System operacyjny	Manjaro 18.0.4 IllyriaArchlinux - 64-bitowy
Sieć	Download 25 Mb/s, Upload 22 Mb/s

4. Rezultaty badań

Wyniki badań jasno wskazywały na różnice między badanymi aplikacjami. Zauważalne różnice mogły mieć źródło w stopniu skomplikowania rozwiązań - aplikacje bogatsze funkcjonalnie miały większe wymagania techniczne - lub w stopniu optymalizacji narzędzi oraz technologiach wykorzystanych przez producentów. W tabeli 3 przedstawiono zbiorcze porównanie badanych aplikacji mobilnych, w tabeli 4 przedstawiono rezultaty badań dla aplikacji internetowych. Kolorem zielonym i jasnozielonym oznaczono najlepsze rezultaty, natomiast kolor czerwony oznacza najgorszy wynik wskaźnika w zestawieniu. W tabeli 4 oprócz aplikacji komercyjnych dodano do porównania aplikację autorską *Agile_app*. Aplikacje pod względem narzutu sprzętowego zauważalnie się od siebie różnią, co oznacza że istnieje możliwość dyskryminacji niewłaściwych dla wymagań użytkownika narzędzi na podstawie narzutu sprzętowego.

Obecnie kryteria wyboru aplikacji wspierających proces zwinnego wytwarzania oprogramowania głównie skupiają się na funkcjonalnościach dostarczanych przez dane rozwiązanie [2, 11, 14]. Przeprowadzone badania wykazały, że niektóre wymagania sprzętowe aplikacji klienckich mogą również mieć znaczenie dla przedsiębiorstw IT oraz użytkowników końcowych. Średnie wykorzystanie pamięci RAM znacznie różniące się dla poszczególnych aplikacji internetowych i mobilnych może mieć znaczenie dla zachowania płynności pracy. Wymagania odnośnie szybkości procesora lub optymalizacja wykorzystania pamięci CACHE widocznie wpływają na komfort użytkowania aplikacji mobilnych i internetowych.

Tabela 4. Wyniki pomiarów parametrów pracy aplikacji mobilnych- wartości średnie, mediany i odchylenia standardowe badanych wskaźników

Parametry	Agile_app	MeisterTask	Yodiz	nTask	yougile	Jira	Trello
Zużycie pamięci RAM % – średnia	2,12	7,16	2,40	2,26	1,94	4,22	2,84
Zużycie pamięci RAM % – mediana	2,10	7,20	2,40	2,30	1,90	4,20	2,80
Zużycie pamięci RAM % – Och. Std.(odchylenie standardowe)	0,26	0,42	0,25	0,40	0,21	0,19	0,27
Zużycie pamięci RAM [MB] – średnia	32,56	109,98	36,86	34,71	29,80	64,82	43,62
Zużycie pamięci RAM [MB] – mediana	32,26	110,59	36,86	35,33	29,18	64,51	43,01
Zużycie pamięci RAM [MB] – Och. Std.	3,98	6,39	3,92	6,20	3,19	2,95	4,15
Transfer sieciowy Upload [MB] – średnia	0,194	0,110	0,056	0,094	0,051	0,669	0,458
Transfer sieciowy Upload [MB] – mediana	0,202	0,111	0,056	0,091	0,051	0,667	0,467
Transfer sieciowy Upload [MB] – Och. Std.	0,063	0,023	0,019	0,047	0,020	0,084	0,066
Transfer sieciowy Download [MB] – średnia	0,527	0,160	0,284	0,320	0,238	0,667	0,614
Transfer sieciowy Download [MB] – mediana	0,532	0,165	0,283	0,272	0,162	0,605	0,533
Transfer sieciowy Download [MB] – Och. Std.	0,128	0,035	0,112	0,127	0,167	0,176	0,180
Czas uruchomienia i reakcji [s] – średnia	1,75	1,74	3,22	6,82	3,30	2,40	
Czas uruchomienia i reakcji [s] – mediana	1,77	1,74	3,22	6,82	3,32	2,26	3,55
Czas uruchomienia i reakcji [s] – Och. Std.	0,11	0,03	0,02	0,03	0,07	0,32	0,04
Wykorzystanie procesora [%] – średnia	53,8	72,6	44,0	78,6	83,4	81,0	79,4
Wykorzystanie procesora [%] – mediana	54,0	72,0	43,0	78,0	85,0	81,0	79,0
Wykorzystanie procesora [%] – Och. Std.	13,26	7,96	5,57	4,62	7,02	6,28	5,18
Zajętość przestrzeni trwałej urządzenia [MB]	26,26	22,75	14,60	24,14	38,24	54,83	31,50

Podczas badań niektóre aplikacje znacznie przekraczały akceptowalny czas oczekiwania użytkownika na reakcję lub uruchomienie 2~3 sekund, negatywnie wpływając na komfort użytkowania [19]. Jedynie aplikacje internetowe *Yodiz* i *nTask* oraz aplikacje mobilne *MeisterTask, JiraCloud* oraz opracowana aplikacja Agile_app spełniły to kryterium.

Tabela 5. Wyniki pomiarów parametrów pracy aplikacji internetowych- wartości średnie, mediany i odchylenia standardowe badanych wskaźników

Parametry	MeisterTask	Yodiz	nTask	yougile	Jira	Trello
Zużycie pamięci RAM [MB] – średnia	245,40	190,60	216,60	124,00	328,60	141,00
Zużycie pamięci RAM [MB] – mediana	262,00	188,00	217,00	122,00	348,00	157,00
Zużycie pamięci RAM [MB] – Odch. Std. (odchylenie standardowe)	50,98	19,82	22,23	13,21	51,74	30,56
Zużycie pamięci CACHE [MB] – średnia	2,09	2,20	2,99	2,06	5,36	1,69
Zużycie pamięci CACHE [MB] – mediana	2,09	2,20	2,99	2,06	5,36	1,69
Zużycie pamięci CACHE [MB] –Odch. Std.	0,00	0,00	0,00	0,00	0,00	0,00
Czas ładowania strony głównej (bez użycia pamięci CACHE) [s] – średnia	3,21	3,69	2,71	1,24	8,05	3,85
Czas ładowania strony głównej (bez użycia pamięci CACHE) [s] – mediana	3,17	3,66	2,83	1,26	8,13	3,90
Czas ładowania strony głównej (bez użycia pamięci CACHE) [s] – Odch. Std.	0,11	0,62	0,34	0,07	0,27	0,22
Czas ładowania strony głównej (z użyciem pamięci CACHE) [s] – średnia	2,35	2,82	1,77	0,86	5,09	3,46
Czas ładowania strony głównej (z użyciem pamięci CACHE) [s] – mediana	2,31	2,80	1,82	0,87	5,12	3,44
Czas ładowania strony głównej (z użyciem pamięci CACHE) [s] – Odch. Std.	0,04	0,20	0,19	0,10	0,22	0,09
Czas budowania widoku strony (bez użycia pamięci CACHE) [s] – średnia	4,18	4,60	2,79	2,27	8,23	3,37
Czas budowania widoku strony (bez użycia pamięci CACHE) [s] – mediana	4,17	4,72	2,82	2,29	8,29	3,39
Czas budowania widoku strony (bez użycia pamięci CACHE) [s] – Odch. Std.	0,09	0,62	0,20	0,07	0,29	0,20
Czas budowania widoku strony (z użyciem pamięci CACHE) [s] – średnia	3,28	3,78	1,81	1,66	5,10	1,97
Czas budowania widoku strony (z użyciem pamięci CACHE) [s] – mediana	3,22	3,81	1,85	1,57	5,06	1,99
Czas budowania widoku strony (z użyciem pamięci CACHE) [s] – Odch. Std.	0,083	0,076	0,189	0,322	0,236	0,041
Objętość załadowanej strony (bez użycia pamięci CACHE) [MB] – średnia	2,10	2,30	3,46	2,18	5,60	1,82
Objętość załadowanej strony (bez użycia pamięci CACHE) [MB] – mediana	2,10	2,30	3,50	2,10	5,60	1,80
Objętość załadowanej strony (bez użycia pamięci CACHE) [MB] – Odch. Std.	0,00	0,00	0,09	0,11	0,00	0,04
Objętość załadowanej strony (z użyciem pamięci CACHE) [MB] – średnia	0,01	0,10	0,47	0,12	0,24	0,13
Objętość załadowanej strony (z użyciem pamięci CACHE) [MB] – mediana	0,01	0,10	0,47	0,12	0,24	0,13
Objętość załadowanej strony (z użyciem pamięci CACHE) [KB] – Odch. Std.	0,00	0,00	0,00	0,00	0,00	0,00
Zużycie procesora [%] – średnia	11,00	13,00	13,80	10,60	16,20	10,20
Zużycie procesora [%] – mediana	11,00	12,00	15,00	10,00	16,00	11,00
Zużycie procesora [%] – Odch. Std.	2,16	2,35	2,39	1,82	2,86	1,92
Liczba pobranych plików (bez użycia pamięci CACHE) – średnia	31,00	105,00	119,60	113,20	94,60	81,60
Liczba pobranych plików (bez użycia pamięci CACHE) – mediana	31,00	105,00	121,00	111,00	94,00	82,00
Liczba pobranych plików (bez użycia pamięci CACHE) – Odch. Std.	0,00	0,00	2,88	3,96	1,34	1,67
Liczba pobranych plików (z użyciem pamięci CACHE) – średnia	11,00	20,00	38,20	18,60	43,20	39,80
Liczba pobranych plików (z użyciem pamięci CACHE) – mediana	11,00	20,00	40,00	18,00	43,00	40,00
Liczba pobranych plików (z użyciem pamięci CACHE) – Odch. Std.	2,16	1,87	4,97	1,52	1,48	0,84

Pomimo braku celowej optymalizacji aplikacji, wyniki badania wymagań sprzętowych w porównaniu z pozostałymi narzędziami nie różniła się znacząco lub, w niektórych przypadkach osiągała lepsze wyniki. Dodatkowo, podczas eksperymentów kilkukrotnie doszło do awarii aplikacji mobilnych *JiraCloud* oraz *nTask*. Ciekawą obserwacją jest fakt ogromnej popularności rozwiązania z oferty firmy *Atlassian*, [1, 10] pomimo najwyższych zmierzonych wymagań sprzętowych. Niektóre aplikacje mobilne znaczną część danych i plików przechowywała lokalnie na urządzeniu, co pozwalało na dostęp do części informacji bez dostępu do sieci Internet lub przy wykorzystaniu łącz o niskiej przepustowości. Te aplikacje odpowiadały wymaganiom dostępności i funkcjonalności [10, 11] stawianym przez niektóre przedsiębiorstwa.

Uzyskane wyniki pozwoliły na stwierdzenie małego znaczenia wymagań sprzętowych części klienckich systemów informatycznych wspierających zwinne wytwarzanie oprogramowania w kontekście wyboru tych rozwiązań przez przedsiębiorstwa IT. Nie wykluczono jednak okoliczności w których może wystąpić konieczność ich uwzględnienia, np. w przypadku skrajnych ograniczeń technicznych przedsiębiorstwa.

5. Podsumowanie i wnioski

Celem badań było sprawdzenie czy aspekty techniczne działania aplikacji tak mobilnych jak i aplikacji internetowych w zakresie zwinnego zarządzania projektami informatycznymi mogą mieć znaczenie w procesie wyboru odpowiedniego narzędzia przez przedsiębiorstwa IT. Aktualnie proces ten jest stosunkowo chaotyczny i obciążony miarą popularności danych rozwiązań.

Przedstawione badania wykazują mierną optymalizację narzędzi, będących aktualnie liderami na rynku oprogramowania *agile*. Istotnie gorsze wyniki wynikają ze stopnia rozbudowania i skomplikowania tych rozwiązań.

Wskaźniki techniczne mimo tego że dziś nie są już tak istotne jak kiedyś, nadal mogą stanowić wartościowe kryterium dla małych lub średnich przedsiębiorstw IT, poszukujących aplikacji wspierających ich procesy *agile*. Pomimo obniżenia kosztów utrzymania sieci dla przedsiębiorstw, wybór łącz o niższej przepustowości, nadal pozwalających na swobodną pracę firmy może doprowadzić do kolejnych oszczędności. Również duże firmy, wybierając aplikacje *agile* o niższych wymaganiach odnośnie przepustowości sieci ograniczają prawdopodobieństwo nadmiernego przeciążenia infrastruktury.

Aplikacje mobilne o niższym narzucie sprzętowym będą płynnie działały na starszych lub tańszych urządzeniach, dzięki czemu przedsiębiorstwa planując zakup urządzeń służbowych dla pracowników mają możliwość kolejnych oszczędności.

Optymalizacja wykorzystania pamięci CACHE przez aplikacje internetowe oraz użycia lokalnej pamięci urządzenia mobilnego do przechowywania danych aplikacji w przypadku aplikacji mobilnych ma duże znaczenie dla komfortu pracy użytkownika. Dzięki tym działaniom znacząco spada czas oczekiwania na załadowanie treści stron i widoków, aplikacje są bardziej responsywne a łącze internetowe nie jest niepotrzebnie obciążane.

Przeprowadzone badania odnosiły się wyłącznie do wymagań sprzętowych wybranych aplikacji przeznaczonych do zwinnego zarządzania projektami informatycznymi. Przegląd literatury wykazał brak podobnych opracowań, co może wskazywać na potencjalny kierunek dalszych badań.

Pomimo braku celowej optymalizacji , autorska aplikacja zaprezentowana w artykule nie odbiegała znacząco od pozostałych analizowanych rozwiązań pod względem wymagań sprzętowych. W niektórych przypadkach osiągała lepsze wyniki, od innych badanych narzędzi.

Zasadność przeprowadzonych eksperymentów aktualnie jest podważalna ze względu na ciągle rosnącą moc obliczeniową, zmniejszający się koszt pamięci RAM oraz przestrzeni dyskowej czy też powszechny dostęp do szerokopasmowej sieci Internet. Większe zasoby sprzętowe nie oznaczają jednak braku konieczności optymalizacji aplikacji i respektowania ograniczeń docelowych urządzeń klienckich na które obecnie trafiają.

Literatura

[1] VersionOne.com. The 1-13th Annual “State of Agile” - Survey

[2] G. Azizyan, M. Magarian, M. Kajko-Mattsson. The Dilemma of Tool Selection for Agile Project Management. ICSEA 2012, The Seventh International Conference on Software Engineering Advances. 2012

[3] Survey:Is agile the new norm?,https://techbeacon.com/app-dev-testing/survey-agile-new-norm, [10.08.2019]

[4] M. S. Raunak, D. Binkley. Agile and other trends in software engineering, 2017

[5] A. M. Dima, M. A. Maassen. From Waterfall to Agile software: Development models in the IT sector, 2006 to 2018. Impacts on company management, 2018

[6] S. Balaji, M Sundararajan Murugaiyan. Waterfall vs v-model vs agile: A comparative study on SDLC, 2012

[7] K. Vilkki. When Agile is not enough. 2010

[8] D. Turk, R. France, and B. Rumpe. Limitations of agile software processes. 2014

[9] U. Kelter, M. Monecke, M. Schild. Do we need Agile Software Development tools? 2003

[10] J. Cabot, G. Wilson. Tools for Teams: A Survey of Web-Based Software Project Portals. 2009

[11] M. Taheri, S. M. Sadjadi. A Feature-Based Tool-Selection Classification for Agile Software Development. 2015

[12] N. Abbas, A. M. Gravell, G. B. Wills. Historical Roots Of Agile Methods: Where Did “Agile Thinking” Come From?. 2008

[13] Gilb, T. Evolutionary Delivery versus the "waterfall model". ACM SIGSOFT Software Engineering Notes. 1985

[14] 7 Criteria for Selecting the Best Project Management Software and Collaboration Tool, https://www.planview.com/resources/articles/selecting-best-project-management-software/, [21.09.2019]

[15] Buyers’ Guide for Beginners Selecting PM Software, https://www.projectmanager.com/blog/buyers-guide-selecting-pm-software, [21.09.2019]

[16] How to pick right agile tool, https://www.smartsheet.com/how-pick-right-agile-tool, [21.09.2019]

[17] Agile tool selection, https://www.slideshare.net/choldorf/agile-tools, [21.09.2019]

[18] Choosing Project Management Software: A Buyer's Guide, https://www.businessnewsdaily.com/9976-project-management-software-buyers-guide.html, [21.09.2019]

[19] F. Fui-Hoon Nah, A study on tolerable waiting time: how long are Web users willing to wait?, 2007

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 14 (2020) 14-18
Wysłane: 2019-09-17
Przyjęte: 2019-10-06

Ocena portalu nauczania języków obcych

Marek Szmit*, Paweł Wojtaszko, Grzegorz Kozieł
Politechnika Lubelska, Katedra Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. Nauka języków obcych staje się coraz bardziej popularna. Wiele osób decyduje się na naukę, ale w wielu przypadkach istnieją pewne bariery. W małych miasteczkach i wioskach nie ma wykwalifikowanych nauczycieli. Ten problem można rozwiązać za pomocą zdalnego nauczania. Ostatnio jest to popularna metoda tworzenia lekcji języka za pośrednictwem komunikatorów głosowych. Taka metoda powinna być wspierana przez inne narzędzie do wysyłania zadań do wykonania studentom lub do ich zbadania. Portal wspierający naukę języków obcych i zapewniający komunikację między nauczycielem a studentami został stworzony przez autorów. Celem artykułu jest przedstawienie jego możliwości i sprawdzenie jego przydatności.

Słowa kluczowe: portal; edukacja; język obcy

*Autor do korespondencji.

Adresy e-mail: marek.szmit@pollub.edu.pl, pawel.wojtaszko@pollub.edu.pl, g.koziel@pollub.pl

An assessment of portal to learn foreign languages

Marek Szmit*, Paweł Wojtaszko, Grzegorz Koziel
Department of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract. Foreign languages learning is becoming more and more popular. A lot of people decides to start learning but in many cases some barriers exist. In small towns or villages there are no qualified teachers. This problem can be solved by remote teaching. Last time it is a popular method to make language lessons via voice communicators. Such a method should be supported by another tool to send tasks to do to students or to examine them. A portal to support learning foreign languages and to ensure communication between teacher and students was created by authors. The goal of the paper is to present its capabilities and to verify its usefulness.

Keywords: portal; education; foreign language

*Corresponding author.

E-mail addresses: marek.szmit@pollub.edu.pl, pawel.wojtaszko@pollub.edu.pl, g.koziel@pollub.pl

1. Wstęp

We współczesnych czasach coraz bardziej popularna staje się nauka języków obcych. Powodów tego zjawiska może być wiele, na pewno przyczynia się do tego wszechobecna globalizacja, oraz popularyzacja Internetu nawet w mniejszych miejscowościach. Aby wyjść naprzeciw oczekiwaniom ludzi na rynku zaczęło pojawiać się coraz więcej serwisów internetowych wspierających naukę języków obcych. Jednym z przykładów jest portal Dummy, stworzony przez autorów, a niniejszy artykuł ma na celu ocenić jego przydatność i porównać efektywność nauki za pomocą wymienionego portalu ze standardowymi metodami nauki. Dla celów niniejszego badania postawiono hipotezę: portal Dummy ułatwia naukę języka angielskiego oraz pozwala uzyskać w stosunku do tradycyjnej metody trwałe wyniki.

Aby zweryfikować przydatność zrealizowanego rozwiązania oraz udowodnić hipotezę główną, postawiono trzy hipotezy robocze:

H1. Użycie portalu Dummy pozwala skrócić czas potrzebny na naukę.

H2. Nauka języka przy wykorzystaniu proponowanej aplikacji jest bardziej skuteczna niż tradycyjna metoda polegająca na uczeniu się słów.

H3. Osoby, które uczyły się przy wykorzystaniu aplikacji pamiętają wyuczone słowa dłużej niż osoby uczące się w sposób tradycyjny.

W celu zweryfikowania tych hipotez przeprowadzono badanie na grupie dziewięciu osób. Każdy z uczestników badania miał za zadanie nauczyć się dwóch grup słów języka angielskiego. Pierwszej grupy słów uczestnicy uczyli się za pomocą portalu Dummy. Druga grupa była grupa kontrolną. Uczono się jej przy użyciu metod tradycyjnych. Każdy uczestnik badania miał za zadanie prowadzić notatki dokumentujące przebieg nauki. Następnie z każdym z uczestników badania przeprowadzono wywiad na temat przebiegu nauki. Dodatkowo efekty nauki wszystkich uczestników zostały zweryfikowane poprzez dwukrotne przeprowadzenie testu wiedzy. Pierwszy test przeprowadzony został bezpośrednio po zakończeniu nauki i miał na celu określenie stopnia opanowania materiału. Drugi test przeprowadzony został po upływie jednego dnia od zakończenia nauki co pozwoliło ustalić czy uczestnicy badania nadal pamiętają wyuczone słówka pomimo upływu czasu.

1.1. Przegląd literatury

Według „Effects of e-learning on Language Learning” [1] główną zaletą e-learningu jest to, że zwiększa zaangażowanie i motywację studentów, które są niezbędne do nauki. Pojawienie się Internetu usprawniło naukę języka angielskiego, ponieważ Internet to miejsce, w którym wszyscy porozumiewamy się podobnym językiem, dzięki czemu znikają bariery kulturowe.

Nauka języków to nie tylko uczenie się słówek na pamięć czy rozwiązywanie quizów, ale także można to robić w znacznie przyjemniejszy sposób. Internetowe gry komputerowe pokazują potencjał nie tylko w zakresie angażowania i rozrywki użytkowników, ale także w promowaniu nauki. Szczególnie dzieci w Turcji spędzają długie godziny grając w gry komputerowe w kafejkach internetowych, głównie w języku angielskim. Badania pokazują, że ma to pozytywny wpływ na naukę języka [2].

W „Computer-supported collaborative learning” [3] przedstawiono ideę połączenia obu metod w celu optymalizacji procesu nauki. Podczas wspólnej nauki w małych grupach dużo szybciej osiągane są kolejne poziomy zaawansowania lingwistycznego. Jest to rozwiązanie ciekawe i warte zastanowienia, ponieważ zgodnie z tą pozycją literaturową kooperacyjne uczenie się z technologią okazuje się dość skomplikowane. Połączenie współpracy, mediacji komputerowej i kształcenia na odległość zakwestionowało samo pojęcie uczenia się i dominujące założenia, jak je zgłębiać.

Ciekawe badania zostały przedstawione w „Improving Learning and Reducing Costs for Online Learning.” [4] pokazują, że zamiast poprawiać jakość, większość kursów opartych na technologii przynosi efekty uczenia się, które są „tak dobre” jak ich tradycyjne odpowiedniki - co często określa się mianem zjawiska „bez znaczącej różnicy”.

Przygotowując się do napisania niniejszej pracy autorzy zapoznali się z publikacjami innych autorów dotyczącymi nauki języków obcych (w szczególności języka angielskiego), dzięki czemu możliwe było opracowanie metodyki planowanego badania pod kątem nauki języków obcych [5-8].

2. Badanie

2.1. Serwis Dummy

Do sprawdzenia skuteczności nauki online wykorzystano projekt inżynierski, którym był serwis nauki języków obcych – Dummy – pozwala on każdemu chętnemu założyć konto „nauczyciela” i dodawać własne kursy. Ankietowani musieli zalogować się na konta uczniowskie i poznawać nowe słowa w języku angielskim.

2.2. Dobór grupy badawczej

Została wyselekcjonowana nieduża grupa dziewięciu osób pochodzących z różnych środowisk oraz będących w zróżnicowanym wieku. Ankietowani dostali plik z instrukcją obsługi, gdzie mieli opisane wszystkie kroki niezbędne to przeprowadzenia badania. Badani poznali cel badania oraz dostali linki do serwisu Dummy jak i ankiety Google Forms, na której zbierane były wyniki. W pliku znajdowała się instrukcja jak założyć konto w serwisie oraz który kurs został przygotowany specjalnie do celów przeprowadzenia niniejszego badania. Jego przebieg przedstawiony został w rozdziale 2.

2.3. Przebieg badań

Na potrzeby badań została utworzona lista wybranych 16 trudnych i niecodziennych słów w języku angielskim. Lista ta została przedstawiona na przykładzie 1.

Przykład 1. Wybrane słowa wraz z tłumaczeniem

```
Commision - prowizja
Subjectrification - uprzedmiotowienie
to pestek - gnębić
carnivore - mięsożerne
predator - drapieżnik
irrefutable - niezaprzeczalny
to daunt - zrażać
vast - olbrzymi
immersive - wciągający
venture - przedsięwzięcie
disambiguation - ujednoznacznienie
abdomen - odwłok
placate - przebłagać
abase - upokorzyć
staphylococcus - gronkowiec
conflagration - pożar
```

W serwisie Dummy został utworzony specjalny kurs zawierający losowo wytypowane 8 z wybranych 16 słów (na przykładzie 1 zostały one zaznaczone na zielono). Pozostała ósemka była wykorzystywana podczas uczenia się tradycyjnymi metodami (na przykładzie 1 zostały one zaznaczone kolorem niebieskim).

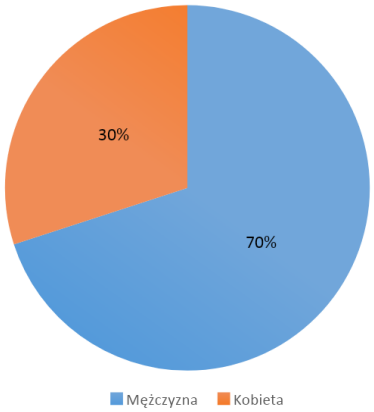
Ankietowani musieli zalogować się w serwisie Dummy, a następnie dołączyć do kursu o nazwie „Badania”. Następnym zadaniem było przejście całego kursu kończąc go wewnętrznym egzaminem. Po zakończeniu części online, ankietowani mieli zapisać czas poświęcony na naukę, a następnie nauczyć się pozostałych 8 słówek w dowolny „manualny” sposób, np. przy pomocy słownika.

Po zakończeniu nauki badani mieli do wypełnienia formularz wygenerowany w Google Forms. Znajdowały się tam pytania klasyfikujące uczniów wg. wieku, doświadczenia z językiem, oraz ponownie test ze znajomości słówek poznanych w aplikacji, oraz test z efektów edukacji manualnej.

Ostatnim etapem badania był powrót do ankiety następnego dnia i wypełnienie jej ponownie celem weryfikacji poziomu utrwalenia informacji.

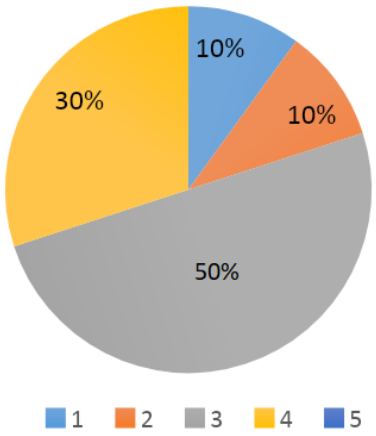
3. Dyskusja wyników

Do badania przystąpiło 10 osób, z czego jedna nie przystąpiła do weryfikacji utrwalenia wiedzy. W badanej grupie znalazło się 7 mężczyzn i 3 kobiety.



Rys. 1. Podział ankietowanych ze względu na płeć

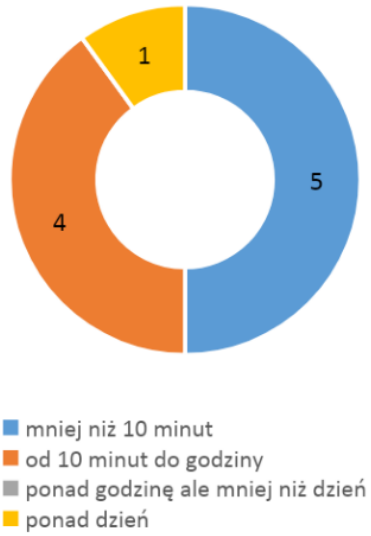
Według ankietowanych ich średnia znajomość angielskiego wynosiła 3 na 5, gdzie 1 to bardzo słaba znajomość języka, a 5 oznacza biegłą znajomość, umożliwiającą swobodną komunikację. Dokładne dane znajdują się na wykresie przedstawionym na rysunku 2.



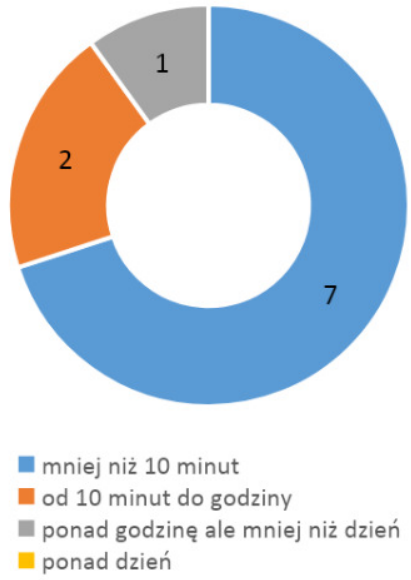
Rys. 2. Subiektywna ocena znajomości języka angielskiego wśród ankietowanych w skali 1-5

Na podstawie tych danych można zauważyć, że badana grupa była całkiem zróżnicowana pod względem ich subiektywnej oceny znajomości języka.

Zagłębiając się w bardziej szczegółowe dane, można zauważyć, że czas poświęcony przez uczestników na naukę słówek metodami tradycyjnymi wygląda tak jak przedstawiono na rysunkach 3 i 4. Wartości na wykresach należy czytać jako liczbę osób spełniających dane kryterium. Czyli na przykład: 5 badanych poświęciło na naukę tradycyjną mniej niż 10 minut, podczas gdy ponad jeden dzień został poświęcony przez jedną osobę.



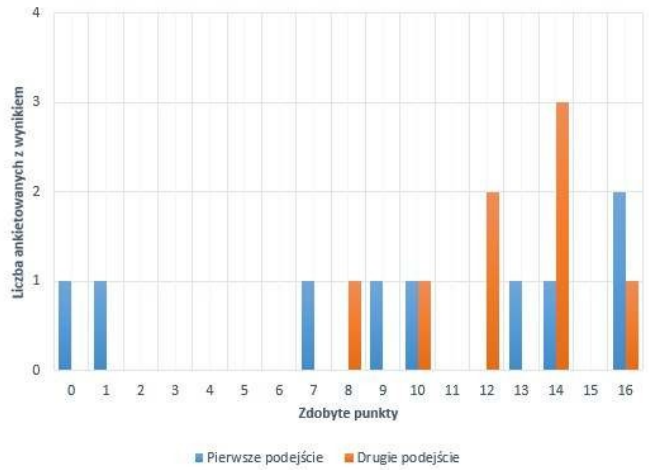
Rys. 3. Czas poświęcony na naukę tradycyjną wśród ankietowanych



Rys. 4. Czas poświęcony na naukę online wśród ankietowanych

Można zauważyć, że ankietowani poświęcili znacznie mniej czasu na naukę przy pomocy serwisu Dummy w porównaniu do metod tradycyjnych. Potwierdza to hipotezę H1.

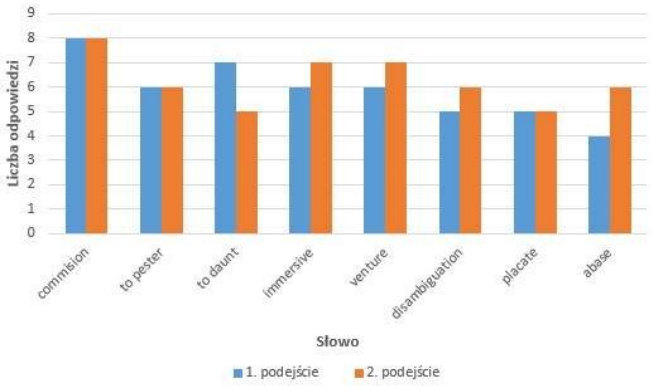
Po pierwszym wykonaniu testu średnia punktów wynosiła 10 na możliwych do zdobycia 16. Następnego dnia wartość ta wzrosła do 13. Rozkład wyników uzyskanych przez poszczególnych badanych przedstawia rysunek 5.



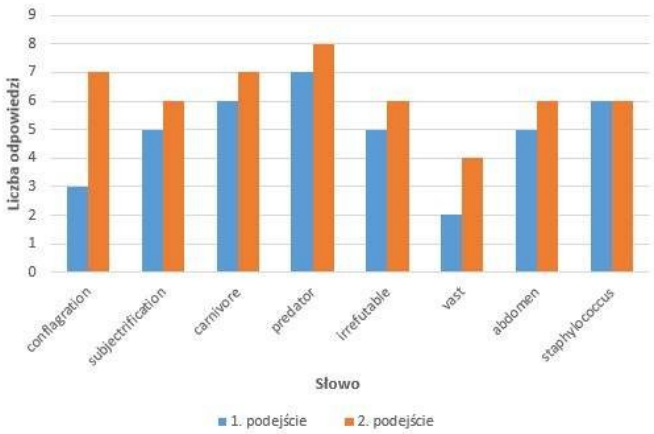
Rys. 5. Liczba osób z poszczególnym wynikiem testu przy pierwszym i kolejnym podejściu.

Skuteczność metod nauczania dobrze widać na wykresach widocznych na rysunkach 6 i 7. W obu przypadkach można zaobserwować poprawę znajomości słów w języku obcym. Oba wykresy przedstawiają liczbę poprawnych odpowiedzi przy pierwszym podejściu zestawiając tą wartość z liczbą poprawnych odpowiedzi dnia następnego. Wyniki wyraźnie wskazują na fakt, że grupa uczestników badania uzyskała znacząco lepsze wyniki testu w przypadku słów, do których nauczenia się wykorzystany został portal Dummy. Średnia ocen uzyskanych przez uczestników bezpośrednio po nauce dla słów uczonych metodami tradycyjnymi wyniosła 4,25 podczas gdy dla słów uczonych za pomocą ocenianego portalu było to już 5,87. Pozwala to na zweryfikowanie hipotezy H2.

Analiza wyników uzyskanych w drugim podejściu do testu wykazuje zbliżone wyniki uzyskane przez ankietowanych dla obydwy grup słów. Podczas nauki tradycyjnej przy drugim podejściu ankietowani mieli zdecydowanie lepsze wyniki niż w podejściu pierwszym. Nauka przy pomocy serwisu Dummy dała bardziej wyrównane wyniki. Wszystkie z zadanych słów zostały zapamiętane, a część badanych dodatkowo na podstawie poprzednich porażek przy kolejnym podejściu zmieniła odpowiedź na poprawną. Zapamiętywalność słów w obu przypadkach była bardzo dobra. Nie da się jednak jednoznacznie określić wpływu pierwszego testu na wyniki zapamiętania poszczególnych słów. Z tego względu nie jest możliwe jednoznaczne zweryfikowanie hipotezy H3.



Rys. 6. Liczba poprawnych odpowiedzi dla poszczególnych słów nauczanych metodami tradycyjnymi



Rys. 7. Liczba poprawnych odpowiedzi dla poszczególnych słów nauczanych przy wykorzystaniu serwisu Dummy

4. Wnioski

Celem niniejszego artykułu było uzyskanie odpowiedzi na pytanie: czy nauka online przy pomocy serwisu Dummy jest efektywniejsza od nauki tradycyjnej. Aby odpowiedzieć na to pytanie zebrano grupę ankietowanych, którzy to podjęli próbę nauczenia się szesnastu względnie trudnych i nieznanych słów w języku angielskim. W celu porównania obu metod, zestaw słów został podzielony na dwie części, każda przypisana do innego sposobu nauki. Jako przykład szkolenia online został wykorzystany serwis Dummy, w którym to ankietowani mieli możliwość poznania nowych słów. Metody tradycyjne nie zostały sprecyzowane w badaniu, uwzględniały one szukanie tłumaczenia na własną rękę przy pomocy dostępnych słowników oraz indywidualne sposoby zapamiętywania nowych informacji.

Analizując uzyskane dane, najbardziej zauważalną różnicą między obiema metodami jest czas jaki zajęła ankietowanym nauka. Jest on dużo krótszy w przypadku nauki online, gdzie aż 70% badanych ukończyło naukę w mniej niż 10 minut. Takim czasem w edukacji tradycyjnej mogło pochwalić się 50% ankietowanych. Co ciekawe, jednemu z ankietowanych nauka z innych źródeł zajęła ponad jeden dzień.

Następnie warto przyjrzeć się trafności odpowiedzi. W przypadku nauki online była ona o wiele lepsza oraz wiedza ta praktycznie od razu się utrwalała. Inaczej się sytuacja miała w przypadku metod tradycyjnych, gdzie pierwsze podejście do testu miało nie najlepszą trafność odpowiedzi. Podczas drugiego podejścia następnego dnia odpowiedzi były o wiele lepsze. Być może miało na to wpływ wykonanie testu online za pierwszym razem, co również przemawia na korzyść nauki przy pomocy Internetu.

Przeprowadzone badania pozwoliły na jednoznaczne zweryfikowanie i potwierdzenie dwóch spośród trzech postawionych hipotez roboczych potwierdzających efektywność i szybkość procesu nauki prowadzonej za pomocą portalu Dummy. Weryfikacja hipotezy trzeciej nie dała jednoznacznych wyników, jednak pozwoliła ustalić, że trwałość efektów nauki jest porównywalna w przypadku

metod tradycyjnych oraz ocenianego serwisu. Pozwala to na potwierdzenie hipotezy głównej: portal Dummy ułatwia naukę języka angielskiego oraz pozwala uzyskać trwałe wyniki.

Literatura

[1] Mohammadi, Neda, Vahid Ghorbani, and Farideh Hamidi. "Effects of e-learning on language learning." Procedia Computer Science 3 (2011): 464-468.

[2] Turgut, Yıldız, and Pelin İrgin. "Young learners' language learning via computer games." Procedia-Social and Behavioral Sciences 1.1 (2009): 760-764.

[3] Stahl, Gerry, Timothy D. Koschmann, and Daniel D. Suthers. Computer-supported collaborative learning. na, 2006.

[4] Twigg, Carol A. "Improving Learning and Reducing Costs for Online Learning." Encyclopedia of Distance Learning. IGI Global, 2005. 1054-1060.

[5] Rosenberg, Marc J., and Rob Foshay. "E-learning: Strategies for delivering knowledge in the digital age." Performance Improvement 41.5 (2002): 50-51.

[6] Celce-Murcia, Marianne, and Lois McIntosh. "Teaching English as a second or foreign language." (1991).

[7] Gu, Yongqi, and Robert Keith Johnson. "Vocabulary learning strategies and language learning outcomes." Language learning 46.4 (1996): 643-679.

[8] Alqahtani, Mofareh. "The importance of vocabulary in language learning and how to be taught." International journal of teaching and education 3.3 (2015): 21-34.

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 14 (2020) 19-25
Wysłane: 2019-10-04
Przyjęte: 2019-10-15

Analiza funkcjonalna i wydajnościowa wybranych brokerów komunikatów w aplikacji rozproszonej

Tobiasz Kaciuczyk, Tomasz Korga*, Jakub Smołka
Politechnika Lubelska, Katedra Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. W artykule przedstawiono wyniki badań wydajności wybranych brokerów komunikatów: Apache ActiveMQ, RabbitMQ oraz Apache Kafka. Analizie został poddany czas przesłania wiadomości wyznaczony na podstawie czasu wysłania i odebrania komunikatu. Testy zostały przeprowadzone za pomocą autorskich aplikacji klienckich napisanych w języku Java. Badania uzupełniono opisem teoretycznym architektury każdego z narzędzi, w tym specyfikacji JMS i AMQP, oraz podstawowym opisem funkcjonalności brokerów.

Słowa kluczowe: broker komunikatów; mikrousługi; komunikacja asynchroniczna

*Autor do korespondencji.

Adres/adresy e-mail: jakub.smolka@pollub.pl, tobiasz.kaciuczyk@pollub.edu.pl, tomasz.korga@pollub.edu.pl

Functional and performance analysis of selected message brokers in a distributed application

Tobiasz Kaciuczyk, Tomasz Korga*, Jakub Smołka
Department of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract. Article presents results of performance analysis of selected message brokers: Apache ActiveMQ, RabbitMQ and Apache Kafka. To analyze has been subjected time of messaging determined based by time of sending and receiving message. Tests were carried out by authorial client application, written in Java language. The research was supplemented with a theoretical description of each tools architecture, including JMS and AMQP specifications and a basic description of brokers functionality.

Keywords: message broker; microservices; asynchronous communication

*Corresponding author.

E-mail address/addresses: jakub.smolka@pollub.pl, tobiasz.kaciuczyk@pollub.edu.pl, tomasz.korga@pollub.edu.pl

1. Wstęp

Brokery komunikatów są odpowiedzią na wiele problemów, przed którymi stają architekci oprogramowania. Ich głównym zadaniem jest wyeliminowanie synchronicznej komunikacji pomiędzy odrębnymi modułami programu oraz zmniejszenie sprzężeń występujących między nimi. Funkcjonalności oferowane przez brokery komunikatów znajdują zastosowanie w systemach przetwarzających dużo informacji, które nie mogą być przetworzone w sposób synchroniczny. Dlatego są szczególnie przydatne w systemach rozproszonych, ale z powodzeniem znajdują zastosowanie również w systemach związanych z Internetem rzeczy, w systemach opartych na architekturze sterowanej zdarzeniami, czy w systemach analizujących duże ilości danych, na przykład w celu zasilania hurtowni danych.

2. Cel i przedmiot badań

Głównym celem przeprowadzonych badań było scharakteryzowanie popularnych brokerów komunikatów pod względem wydajności, mierzonej w czasie przetwarzania i przesyłania komunikatów. Badania zostały uzupełnione teoretycznymi opisami architektury brokerów oraz opisem podstawowych funkcjonalności, użytecznych z punktu widzenia użytkowników.

Do analizy zostały wybrane trzy narzędzia realizujące funkcję brokera komunikatów: Apache ActiveMQ, RabbitMQ oraz Apache Kafka.

3. Apache ActiveMQ

Apache ActiveMQ [1], [2] jest brokerem komunikatów opartym na specyfikacji JMS (Java Message Service) w wersji 1.1. Został napisany w języku Java, ale posiada wiele bibliotek klienckich dla innych języków. Jest rozwijany przez Apache Software Foundation jako otwarte oprogramowanie i został wydany na licencji Apache 2.0, która umożliwia każdemu dowolne używanie i modyfikowanie kodu źródłowego.

ActiveMQ jest narzędziem zaprojektowanym jako warstwa pośrednia zorientowana na wymianę komunikatów (ang. message-oriented-middleware). Jego podstawowym celem jest przekazywanie wiadomości i zdarzeń w aplikacji rozproszonej, gwarantując ich odebranie przez odpowiednie

komponenty. Żeby odpowiednio realizować tę funkcję broker powinien cechować się wysoką dostępnością, wydajnością i łatwą skalowalnością. Celem inżynierów pracujących nad ActiveMQ było dostarczenie narzędzia opartego na standardzie z funkcjonalnością integracji pomiędzy różnymi technologiami. ActiveMQ jest elastyczny w kwestii wykorzystywanego protokołu do realizacji komunikacji. Wspierane są między innymi rozwiązania takie jak AMQP, STOMP, MQTT, a także TCP, UDP, czy XMPP.

3.1. Architektura

ActiveMQ jest implementacją specyfikacji JMS, dlatego opis architektury tego narzędzia warto rozpocząć od opisania tego standardu.

Java Message Service został wydany w celu stworzenia standardowego API do wysyłania i odbierania komunikatów w języku Java. Dzięki tej specyfikacji programista może zaimplementować w swojej aplikacji obsługę wiadomości z brokera komunikatów bez specjalistycznej wiedzy o konkretnej implementacji brokera.

JMS definiuje w swojej specyfikacji następujące artefakty związane z architekturą komunikatów [1]:

- *JMS client* - klient kolejki, aplikacja do wysyłania i odbierania wiadomości napisana w języku Java, może odbierać wiadomości w sposób blokujący wątek programu lub nie;
- *Non-JMS client* - klient kolejki stworzony za pomocą dedykowanych rozwiązań dla języków programowania innych niż Java;
- *JMS producer* - producent, klient brokera, który tworzy i wysyła wiadomości;
- *JMS consumer* - konsument, klient brokera, który odbiera i przetwarza wiadomości;
- *JMS provider* - dostawca JMS, broker, kolejka, implementacja interfejsów JMS napisana w języku Java
- *JMS message* - wiadomość, komunikat, podstawowy artefakt specyfikacji JMS, wysyłany i odbierany przez klientów JMS;
- *JMS domains* - dziedziny, dwa modele przekazywania wiadomości: bezpośrednio, lub w architekturze publikacji/subskrypcji;
- *Administered objects* - obiekty konfiguracyjne, zawierają dane konfiguracyjne potrzebne klientom, dostępne za pomocą interfejsu JNDI (Java Naming and Directory Interface) języka Java;
- *Connection factory* - fabryka połączeń, używana przez klientów do tworzenia połączeń z brokerem (JMS provider);
- *Destination* - cel, obiekt do którego wiadomości są adresowane, lub od którego są odbierane.

Architektura JMS umożliwia skorzystanie z dwóch modeli dystrybucji wiadomości: tematów (ang. topic) topików i kolejek (ang. queue). Temat jest to model przekazywania komunikatów, w którym może być wiele subskrybentów danego kanału. Wiadomość przekazana do tematu, zostanie przesłana do każdego zarejestrowanego subskrybenta. W modelu kolejki wiadomość może zostać odebrana tylko raz przez jednego subskrybenta.

Wiadomość w JMS składa się z trzech elementów [2]:

- ciała wiadomości (ang. body) - może to być tekst, mapa, tablica bajtów, obiekt, lub strumień obiektów o typach prymitywnych;
- nagłówka - za jego pomocą programista może sterować sposobem, w jaki wiadomość jest przetwarzana przez broker;
- właściwości (ang. properties) - zawierają dodatkowe zmienne sterujące aplikacją, ale nie brokerem.

Specyfikacja JMS dostarcza wiele predefiniowanych nagłówków, pozwalających na sterowanie brokerem, niektóre z nich to [1]:

- *JMSDestination* - reprezentuje cel wiadomości: kolejkę lub temat
- *JMSDeliveryMode* - zawiera informację, czy wiadomość powinna zostać trwale zapisana;
- *JMSExpiration* - czas długości życia wiadomości wyrażony w milisekundach;
- *JMSPriority* - priorytet, wartość całkowita od 0 do 9;
- *JMSMessageID* - unikalny identyfikator wiadomości;
- *JMSTimestamp* - czas, w którym wiadomość została wysłana;
- *JMSReplyTo* - opcja umożliwiająca wyjątkowe sterowanie logiką aplikacji za pomocą komunikatu, reprezentuje ten sam obiekt co *JMSDestination*, programista może zdecydować do jakiej kolejki trafi odpowiedź na wysłany przez niego komunikat.

4. RabbitMQ

RabbitMQ [3-5] jest otwartym oprogramowaniem, został napisanym w języku Erlang, jego rozwijaniem zajmuje się Pivotal Software Inc. RabbitMQ został oparty na protokole AMQP (Advanced Message Queuing Protocol), który definiuje zarówno protokół komunikacji w sieci, ale również podstawowy model brokera.

4.1. Architektura AMQP

Specyfikacja AMQP 0-9-1 definiuje zarówno protokół komunikacji sieciowej jak również niektóre usługi i zachowania brokera. Model AMQ (Advanced Message Queuing) definiuje trzy główne komponenty brokera [4]:

- *exchange* (punkt wymiany) - odbiera komunikaty od wydawcy i kieruje je do kolejek;
- *queue* (kolejka) - odbiera komunikaty z punktu wymiany i wysyła je do odbiorców. Stanowi strukturę danych na dysku lub w pamięci podręcznej, miejsce przechowywania komunikatów;
- *binding* (wiązanie) - zasady wykorzystywane przez punkt wymiany w celu skierowania komunikatu do odpowiedniej kolejki, definiowane przez klientów.

Protokół AMQP 0-9-1 definiuje 4 rodzaje punktu wymiany [4]:

- *direct exchange* - dostarcza wiadomość do kolejki na podstawie klucza trasowania (ang. routing key) zawartego w wiadomości;
- *fanout exchange* - dostarcza wiadomość do wszystkich kolejek powiązanych z danym punktem wymiany, klucz trasowania jest ignorowany;
- *topic exchange* - dostarcza wiadomość do jednej lub wielu kolejek, na podstawie klucza trasowania;
- *headers exchange* - klucz trasowania jest ignorowany, zamiast niego o przekierowaniu wiadomości decydują opcje zawarte w jej nagłówku.

Kolejnym podstawowym elementem modelu AMQP jest kolejka. Każda kolejka musi posiadać unikalną nazwę. Może być trwała lub nietrwała: trwała zostanie przywrócona po restarcie aplikacji, nietrwała zostanie usunięta. Warto zaznaczyć, że w przypadku trwałej kolejki, przywrócone zostaną tylko te wiadomości, które zostały oznaczone flagą trwałości (persisted). Istnieje możliwość stworzenia kolejki wyłącznie dla konkretnego połączenia. Taka kolejka zostanie usunięta po przerwaniu danego połączenia. Kolejki mogą być też automatycznie usuwane gdy stracą wszystkich konsumentów wiadomości. Specyfikacja AMQP udostępnia również mechanizm kontrolowania dostarczenia wiadomości ACK (ang. acknowledgements). Wiadomość zostanie usunięta z kolejki dopiero po otrzymaniu informacji o jej poprawnym odebraniu przez konsumenta [4].

Wiadomość w AMQP posiada cechy nazywane atrybutami, na przykład [5]:

- *content type* - typ treści wiadomości;
- *content encoding* - sposób kodowania treści;
- *routing key* - klucz trasowania;
- *delivery mode* - sposób dostarczenia (trwały lub nie);
- *message priority* - priorytet wiadomości;
- *message publishing timestamp* - czas utworzenia wiadomości;
- *expiration period* - czas ważności wiadomości;
- *publisher application id* - identyfikator aplikacji publikującej wiadomość.

Treść wiadomości jest traktowana jako tablica bajtów. Broker nie modyfikuje ani nie czyta treści wiadomości. Atrybuty związane z treścią (*content-type*) są potrzebne dla zachowania konwencji przy używaniu formatów takich jak JSON. Wiadomości, które nie mogą zostać przetworzone, mogą być zwrócone do producentów, usunięte lub przekazane na inną kolejkę [3].

Duża elastyczność RabbitMQ wynika ze sposobu w jaki komunikaty mogą być przekazywane z punktu wymian do kolejek, oraz możliwości definiowania zasad przekazywania wiadomości podczas jej tworzenia. Wiązania pomiędzy punktem wymiany i kolejkami to podstawa architektury bazującej na komunikatach. RabbitMQ umożliwia łatwe przystosowanie aplikacji do zmieniających się wymagań [3].

5. Apache Kafka

Apache Kafka [6-8] powstała w odpowiedzi na potrzeby firmy LinkedIn w zakresie komunikacji wielu komponentów w systemie rozproszonym. Została udostępniona jako otwarte oprogramowanie.

Realizuje wzorzec wydawcy i subskrybenta (ang. publish/subscribe). Celem inżynierów pracujących nad Kafką było stworzenie narzędzia o następujących cechach: bardzo dużej wydajności, możliwości przetwarzania dużej liczby komunikatów jednocześnie, ich trwałym przechowywaniu, a także zachowaniem kolejności, w której komunikaty trafiły do brokera. Dodatkowo Kafka zapewnia mechanizmy zabezpieczeń przed awariami i wysoką skalowalność [6].

5.1. Architektura

Pojedyncza jednostka danych w Apache Kafka jest nazywana wiadomością, komunikatem, wierszem lub rekordem, w praktyce jest to tablica bajtów bez określonego typu. Komunikat może posiadać również identyfikator nazywany kluczem. Wiadomości mogą być grupowane w kolekcje nazwane partiami (ang. batch). Grupowanie wiadomości w partie wiąże się z dużo szybszym przetworzeniem całej partii, niż gdyby wiadomości byłyby wysyłane osobno. Ale czas przetworzenia pojedynczej wiadomości będzie oczywiście dłuższy (musi zostać przetworzona cała partia). Partie są zazwyczaj kompresowane co podnosi wydajność procesu [6].

Wiadomości są umieszczane w tematach. Temat w Apache Kafka jest analogicznym bytem do tabeli w bazie danych. Każdy temat musi zawierać przynajmniej jedną partycję. Wysłanie wiadomości polega na dopisaniu na koniec partycji. Odczytanie jest pobraniem wiadomości z zachowaniem kolejności, od pierwszej do ostatniej. Nie ma możliwości modyfikacji wiadomości, która znajduje się już w kolejce. Zachowanie kolejności jest możliwe tylko w obrębie pojedynczej partycji. Każda partycja może się znajdować fizycznie na innej maszynie, co znacząco ułatwia skalowanie już na poziomie tematów.

Pojedyncza instancja Apache Kafka jest nazywana brokerem. Wielu brokerów można łączyć w klastry. Broker jest właścicielem partycji. W przypadku gdy partycja należy do wielu brokerów, jeden broker staje się liderem partycji, a pozostałe służą jako mechanizm powielania danych na wypadek awarii.

Oprócz mechanizmów powielania Kafka posiada również mechanizmy retencji, czyli składowania danych, z wieloma możliwościami konfiguracyjnymi. W przypadku zatrzymania działania serwera dane nie zostaną utracone, ponieważ Apache Kafka domyślnie zapisuje wszystkie dane na dysku twardym.

6. Zestawienie funkcjonalności

W celu porównania funkcjonalności opisywanych brokerów, zostały one zestawione ze sobą w tabeli 1. Porównanie zostało opracowane na podstawie materiałów źródłowych [1-8], a także na podstawie praktycznych doświadczeń, zdobytych podczas pracy nad aplikacją badawczą wykorzystującą każdą z kolejek.

Tabela 1. Wydajność brokerów dla różnych rozmiarów wiadomości

Apache ActiveMQ	**RabbitMQ**	**Apache Kafka**
Protokół przesyłania wiadomości		
Wiele wspieranych protokołów, m. in.: Stomp, AMQP 1.0, MQTT, XMPP.	Wspierane protokoły: AMQP 0-9-1 z rozszerzeniami, Stomp, AMQP 1.0, MQTT.	Własna implementacja protokołu sieciowego opartego na protokole TCP.
Format wiadomości		
Według specyfikacji JMS: mapa, tablica bajtów, obiekt, strumień obiektów o typach prymitywnych.	Obiekt serializowany do tablicy bajtów. Możliwość własnej implementacji mechanizmu serializacji i deserializacji.	Obiekt serializowany do tablicy bajtów. Możliwość własnej implementacji mechanizmu serializacji i deserializacji.
Trwałość wiadomości		
Możliwość wyboru różnych sposobów persystencji: interfejs JDBC lub replikowane, plikowe bazy danych: KahaDB lub Replicated LevelDB Store.	Mechanizm persystencji wiadomości za pomocą rozproszonej bazy danych Mnesia (dane mogą być przechowywane w pamięci operacyjnej lub na dysku).	Każda wiadomość jest domyślnie zapisywana na dysku twardym.
Potwierdzenie dostarczenia wiadomości		
Tak, posiada konfigurowalny mechanizm.	Tak, posiada konfigurowalny mechanizm.	Brak.
Bezpieczeństwo		
Wspiera dołączane mechanizmy bezpieczeństwa. Domyślnie JAAS (Java SE Security) do uwierzytelniania oraz plik konfiguracyjny do autoryzacji.	Wspiera dołączane mechanizmy autoryzacji (np. LDAP) i uwierzytelniania (hasło lub certyfikaty), wspiera protokół TLS do szyfrowania komunikacji.	Wspiera dołączane mechanizmy autoryzacji i uwierzytelniania, wspiera protokół SSL/TLS do szyfrowania komunikacji.
Model subskrypcji brokera		
Kolejka (*JMSQueue*) lub temat (*JMSTopic*).	Oparty o mechanizm punktu wymiany, który ma wiele możliwości. Opisany w rozdziale 4.1.	Temat (ang. *topic*).
Środowisko rozproszone		
Wspiera tworzenie sieci brokerów (logicznie rozłącznych), klastra brokerów, i topologię *Master/Slave*.	Instancje mogą być łączone w klastry - tworząc jeden logiczny broker - lub federacje - brokerzy są logicznie oddzielni, mogą komunikować się ze sobą.	Został zaprojektowany dla systemów rozproszonych, używa narzędzia Zookeeper do zarządzania instancjami, łatwo skalowalna.
Zarządzanie brokerem		
Graficzny interfejs użytkownika dostępny przez przeglądarkę lub interfejs linii poleceń.	Graficzny interfejs użytkownika dostępny przez przeglądarkę lub interfejs linii poleceń.	Interfejs linii poleceń.
Interfejs programistyczny		
Biblioteka w języku Java, interfejs HTTP REST, interfejs WebSocket. Dodatkowe biblioteki klienckie dla większość popularnych języków programowania.	Oficjalne biblioteki klienckie w językach: Java, Erlang, .NET. Wiele dodatkowych bibliotek i rozszerzeń.	Oficjalne biblioteki klienckie w językach: Java, .NET. Wiele dodatkowych bibliotek i rozszerzeń.
Monitoring		
Monitoring możliwy przez JMX, interfejs graficzny, aplikację linii poleceń, dodatkowe narzędzia.	Metryki udostępnianie przez interfejs API, aplikację linii poleceń, interfejs graficzny.	Metryki domyślnie udostępniane przez JMX.
Inne		
Wsparcie standardu SOAP, integracja z Apache Camel, Spring, integracja z serwerami aplikacyjnymi poprzez wsparcie mechanizmu JNDI.	Integracja z narzędziem Spring.	Integracja z narzędziem Spring.

7. Badania wydajności

7.1. Środowisko badawcze

Badania zostały przeprowadzone za pomocą dwóch programów napisanych w języku Java, które pełniły rolę wydawcy oraz odbiorcy wiadomości dla każdego z brokerów.

Każda z kolejek została uruchomiona w osobnym kontenerze utworzonym za pomocą narzędzia Docker.

Testy wydajności zostały przeprowadzone na komputerze z parametrami technicznymi:

- procesor Intel Core i5-7200U CPU@2.5Ghz (4 rdzenie),
- 16GB pamięci RAM, dysk twardy typu SSD o pojemności 256GB,
- system operacyjny Ubuntu 18.04.

Programy zostały napisane w języku Java w wersji 1.11.0, dodatkowo zostały wykorzystane następujące narzędzia: serwer aplikacyjny Apache Tomcat 9.0.21, Spring Boot 2.1.6, Docker 18.06, Zookeeper 3.4.13.

Do badania wybrano stabilne wersje każdej z kolejek: Apache ActiveMQ 5.15.6, RabbitMQ 3.7.17, oraz Apache Kafka 2.12.

7.2. Metodyka badawcza

Do wykonania pomiarów czasu przesyłania wiadomości wykorzystano dwa programy. Pierwszy z nich przesyłał wiadomości do kolejek oraz odpowiadał za zapisanie czasu wysłania wiadomości. Drugi program miał za zadanie odebranie wiadomości oraz zapisanie czasu odebrania.

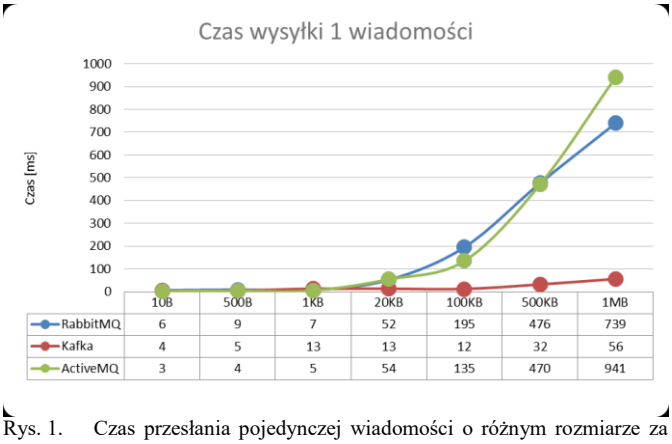
Podczas badań pozostawiono domyślną konfigurację każdego z brokerów. Wszystkie próby zostały przeprowadzone według wzorca: jeden wydawca - jeden subskrybent.

W ramach przypadków testowych dla każdego z testowanych brokerów stworzono siedem wiadomości o wielkościach: 10 B, 500 B, 1 KB, 20 KB, 100 KB, 500 KB oraz 1 MB. Każda z tych wiadomości w ramach pojedynczego przypadku testowego została przesłana: 1 raz, 100 razy, 1000 razy, 3000 razy, 5000 razy.

Różnica czasu pomiędzy odebraniem ostatniej, a wysłaniem pierwszej wiadomości pozwoliła na określenie czasu przesyłania wiadomości przez broker wiadomości, a dalsza obróbka danych umożliwiła określenie wydajności każdej z kolejek. Wszystkie testy odbyły się na jednym komputerze w jak najbardziej zbliżonych do siebie warunkach. Rezultaty badań pozwoliły na wygenerowanie wykresów widocznych w następnym punkcie.

7.3. Wyniki

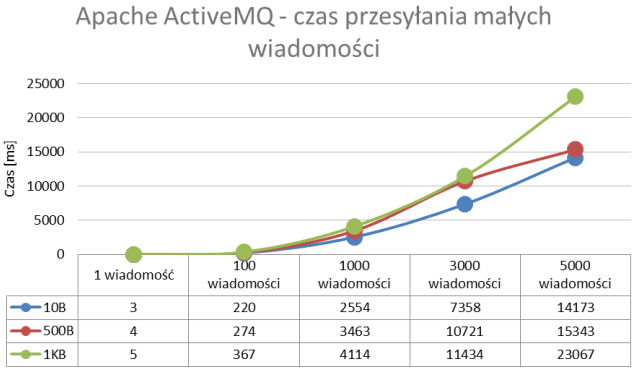
Pierwszym ze stworzonych testów badawczych było porównanie czasów wysyłki jednej wiadomości dla różnej jej długości w analizowanych kolejkach. Wyniki badania przedstawiono na Rys. 1.



	10B	500B	1KB	20KB	100KB	500KB	1MB
RabbitMQ	6	9	7	52	195	476	739
Kafka	4	5	13	13	12	32	56
ActiveMQ	3	4	5	54	135	470	941

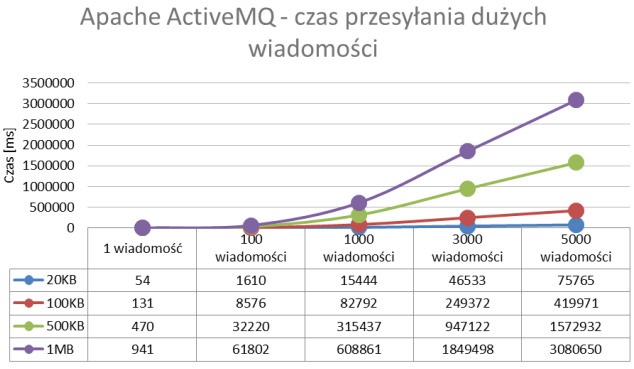
Rys. 1. Czas przesłania pojedynczej wiadomości o różnym rozmiarze za pomocą każdej z kolejek.

Kolejnymi testami były pomiary czasu przesłania wiadomości. Tutaj zostały one podzielone na dwie grupy: wiadomość mała (rozmiar poniżej 1 KB) oraz wiadomość duża (rozmiar powyżej 1 KB). Podział ten został wykonany z racji różnicy wielkości jednostki czasu i poprawy czytelności wyników.



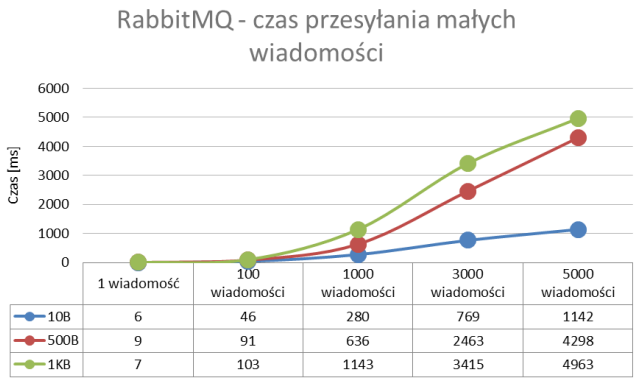
	1 wiadomość	100 wiadomości	1000 wiadomości	3000 wiadomości	5000 wiadomości
10B	3	220	2554	7358	14173
500B	4	274	3463	10721	15343
1KB	5	367	4114	11434	23067

Rys. 2. Czas przesłania różnych ilości wiadomości o rozmiarach 10 B, 500 B i 1 KB za pomocą Apache ActiveMQ.



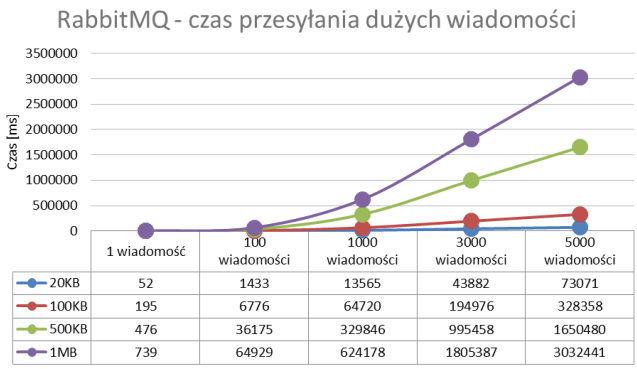
	1 wiadomość	100 wiadomości	1000 wiadomości	3000 wiadomości	5000 wiadomości
20KB	54	1610	15444	46533	75765
100KB	131	8576	82792	249372	419971
500KB	470	32220	315437	947122	1572932
1MB	941	61802	608861	1849498	3080650

Rys. 3. Czas przesłania różnych ilości wiadomości o rozmiarach 20 KB, 100 KB, 500 KB i 1 MB za pomocą Apache ActiveMQ.



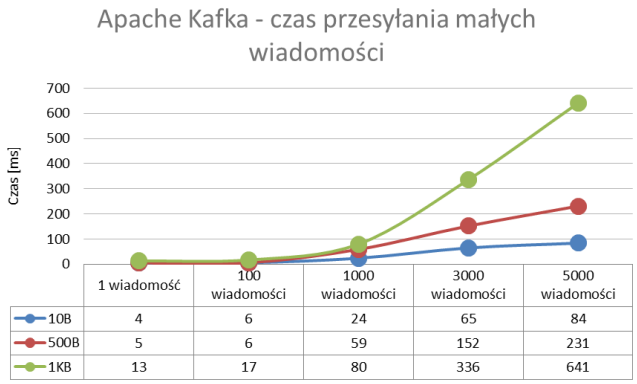
	1 wiadomość	100 wiadomości	1000 wiadomości	3000 wiadomości	5000 wiadomości
10B	6	46	280	769	1142
500B	9	91	636	2463	4298
1KB	7	103	1143	3415	4963

Rys. 4. Czas przesłania różnych ilości wiadomości o rozmiarach 10 B, 500 B i 1 KB za pomocą RabbitMQ.



	1 wiadomość	100 wiadomości	1000 wiadomości	3000 wiadomości	5000 wiadomości
20KB	52	1433	13565	43882	73071
100KB	195	6776	64720	194976	328358
500KB	476	36175	329846	995458	1650480
1MB	739	64929	624178	1805387	3032441

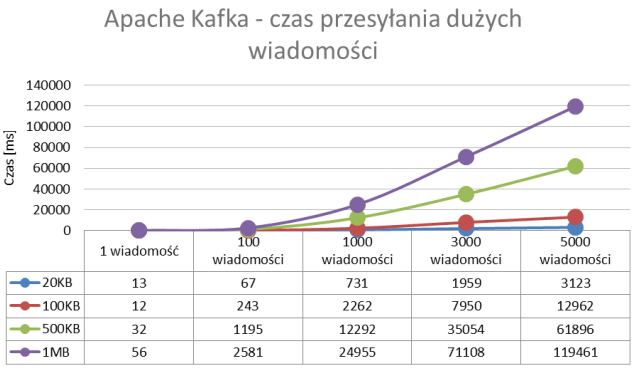
Rys. 5. Czas przesłania różnych ilości wiadomości o rozmiarach 20 KB, 100 KB, 500 KB i 1 MB za pomocą RabbitMQ.



	1 wiadomość	100 wiadomości	1000 wiadomości	3000 wiadomości	5000 wiadomości
10B	4	6	24	65	84
500B	5	6	59	152	231
1KB	13	17	80	336	641

Rys. 6. Czas przesłania różnych ilości wiadomości o rozmiarach 10 B, 500 B i 1 KB za pomocą Apache Kafka.

Dzięki przeprowadzonym testom możliwe było wyznaczenie wydajności kolejek dla różnych rozmiarów przesyłanych danych. W Tabeli 1. przedstawiono wydajność dla próbki 5000 wiadomości.



	1 wiadomość	100 wiadomości	1000 wiadomości	3000 wiadomości	5000 wiadomości
20KB	13	67	731	1959	3123
100KB	12	243	2262	7950	12962
500KB	32	1195	12292	35054	61896
1MB	56	2581	24955	71108	119461

Rys. 7. Czas przesłania różnych ilości wiadomości o rozmiarach 20 KB, 100 KB, 500 KB i 1 MB za pomocą Apache Kafka.

Tabela 2. Wydajność brokerów dla różnych rozmiarów wiadomości

Wydajność na sekundę			
	RabbitMQ	Kafka	ActiveMQ
10B	4 378,28	59 523,81	352,78
500B	1 163,33	21 645,02	325,88
1KB	1 007,46	7 800,31	216,76
20KB	68,43	1 601,02	65,99
100KB	15,23	385,74	11,91
500KB	3,03	80,78	3,18
1MB	1,65	41,85	1,62

8. Wnioski

Z przesyłaniem pojedynczej wiadomości (Rys. 1.) najlepiej poradził sobie Apache Kafka. Czas wysyłki pojedynczej wiadomości za pomocą tego brokera jest prawie niezmienny dla każdego z badanych rozmiarów. Dla pozostałych kolejek czas zaczyna znacząco rosnąć przy wiadomościach o rozmiarze powyżej 20 KB. Z tego badania wynika, że zarówno Apache ActiveMQ oraz RabbitMQ nie zostały zoptymalizowane pod względem przetwarzania wiadomości o dużych rozmiarach.

Wydajność poszczególnych brokerów najlepiej obrazuje Tabela 2. w której przedstawiono liczbę wysłanych wiadomości na sekundę. Zdecydowanie najbardziej wydajną kolejką jest Apache Kafka - dla każdego z badanych rozmiarów wiadomości Kafka osiągała wynik wielokrotnie lepszy od pozostałych brokerów. W porównaniu Apache ActiveMQ z RabbitMQ lepiej wypadł ten drugi, choć znaczące różnice wystąpiły tylko dla trzech najmniejszych badanych rozmiarów.

ActiveMQ w badaniu przesyłania małych wiadomości (Rys. 2.), wykazał podobne czasy dla każdego z badanych rozmiarów, wzrost czasu przesłania był spowodowany głównie zwiększeniem ilości przesyłanych komunikatów. W badaniu dużych wiadomości (Rys. 3), różnice w czasie przesłania pomiędzy poszczególnymi rozmiarami były już dużo większe.

RabbitMQ okazał się kolejką dużo szybszą od ActiveMQ w przypadku przesyłania małych wiadomości (Rys. 4.), co potwierdza wynik badania wydajności. W przypadku tego brokera jest jednak widoczna duża różnica pomiędzy najmniejszym rozmiarem komunikatu i pozostałymi, przy próbie wysłania 3000 i więcej wiadomości. W badaniu czasu przesyłania dużych wiadomości (Rys. 5.), wynik RabbitMQ okazał się niemal identyczny z wynikiem ActiveMQ, co również potwierdza wynik badania wydajności.

Wyniki przesyłania wiadomości o małych i dużych rozmiarach w przypadku Apache Kafka okazały się przewidywalne, czasy przesłania rosną proporcjonalnie z liczbą wysyłanych wiadomości oraz z ich rozmiarem (Rys. 6, Rys. 7.).

Z analizy funkcjonalnej wynika, że Apache Kafka oferuje najmniej dodatkowych funkcjonalności. Udostępnia tylko jeden model subskrypcji wiadomości (temat), brakuje mechanizmu kontroli dostarczania komunikatów. W przeciwieństwie do pozostałych badanych narzędzi, Kafka nie ma wbudowanego graficznego narzędzia administracyjnego. Mimo to posiada wszystkie podstawowe funkcjonalności, takie jak szyfrowanie komunikacji, interfejs linii poleceń, biblioteki klienckie, czy udostępnianie metryk.

Warto zwrócić uwagę, że Apache Kafka jest wyjątkowym narzędziem na tle pozostałych. W komunikacji nie zdecydowano się na wykorzystanie jednego z gotowych protokołów komunikacji, czy specyfikacji, ale zastosowano autorskie rozwiązania. Szczególny jest również sposób przechowywania wiadomości w kolejce, gdyż każdy komunikat jest automatycznie zapisywany na dysku twardym. Nie jest to dodatkowa opcja bezpieczeństwa, ale naturalny sposób na zarządzanie komunikatami w brokerze.

Powyższe cechy wskazują, że Apache Kafka jest narzędziem o bardzo dużej wydajności, która została osiągnięta częściowo przez podjęcie kompromisów i rezygnację z niektórych funkcjonalności

ActiveMQ oraz RabbitMQ pod względem funkcjonalności są również bardzo podobne do siebie. ActveMQ jest implementacją specyfikacji JMS, co sprawia, że jest bardzo proste w użyciu w systemach opartych o język Java. RabbitMQ wydaje się oferować większą uniwersalność. Za sprawą mechanizmu punktów wymiany, dostarcza wyjątkową elastyczność w konfiguracji narzędzia do własnych potrzeb.

Literatura

[1] Snyder, B., Bosanac, D., & Davies, R., Introduction to Apache Activemq. Active MQ in Action, 2017.

[2] Dokumentacja techniczna do Apache ActiveMQ https://activemq.apache.org/components/classic/documentation [22.09.2019]

[3] Roy G. M., RabbitMQ in Depth, Manning, 2017.

[4] Wprowadzenie do AMQP w RabbitMQ https://www.rabbitmq.com/tutorials/amqp-concepts.html [22.09.2019]

[5] Dokumentacja techniczna do RabbitMQ https://www.rabbitmq.com/documentation.html [22.09.2019]

[6] Narkhede N., Shapira G., Palino T., Kafka: The Definitive Guide. Real-Time Data and Stream Processing at Scale, O'Reilly, 2017.

[7] John V., Liu, X., A survey of distributed message broker queues. arXiv preprint arXiv:1704.00411, 2017.

[8] Dokumentacja techniczna do Apache Kafka https://kafka.apache.org/documentation/ [22.09.2019]

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 14 (2020) 26-30
Wysłane: 2019-10-09
Przyjęte: 2019-10-18

Analiza porównawcza technologii Windows Presentation Foundation i Windows Forms

Michał Pasztaleniec*, Maria Skublewska-Paszkowska
Politechnika Lubelska, Katedra Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. Artykuł przedstawia porównanie dwóch technologii do tworzenia aplikacji desktopowych opartych o język C#. Analiza dotycząca wykorzystania procesora, pamięci RAM oraz czasu wykonywania operacji została przeprowadzona na technologiach Windows Presentation Foundation i Windows Forms. Do badań wykorzystano aplikacje o tej samej funkcjonalności w obu technologiach. W analizie porównawczej uwzględniono czasy danych operacji oraz wydajność aplikacji.

Słowakluczowe: .NET; WPF; Windows Presentation Foundation; Windows Forms; WinForms; analiza wydajności

*Autor do korespondencji.

Adres/adresy e-mail: mmichal210@gmail.com, maria.paszkowska@pollub.pl

Comparative analysis of Windows Presentation Foundation and Windows Forms

Michał Pasztaleniec*, Maria Skublewska-Paszkowska
Department of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract. The article presents a comparison of two technologies for creating desktop applications based on C#. The analysis conserns the use of the processor, RAM and time of the operation was carried out on the technologies of Windows Presentation Foundation and Windows Forms. The research use applications with the same functionality in both technologies. The comparative analysis took into account the times of operations and application performance.

Keywords: .NET; WPF; Windows Presentation Foundation; Windows Forms; WinForms; performance analysis

*Corresponding author.

E-mail address/addresses: mmichal210@gmail.com, maria.paszkowska@pollub.pl

1. Wstęp

W artykule porównano dwie technologie, które pomimo swoich lat, nadal są używane do tworzenia aplikacji desktopowych. Wybór został uwarunkowany tym, że na rynek aplikacji desktopowych pisanych w .NET programiści najczęściej wybierają Windows Presentation Foundation i Windows Forms [1]. Drugim powodem jest to, że większość artykułów skupia się na porównaniu teoretycznych różnic tych technologii.

Z corocznych badań serwisu stackoverflow wynika, że aplikacje desktopowe dla systemu Windows zajmują 41% rynku. Także język, w którym pisane są aplikację w Windows Presentation Foundation i Windows Forms, cieszy się dużą popularnością. Język C# według badań serwisu stackoverflow, w 2017 zajmował czwarte miejsce z 34,1% oraz w 2018 na ósmym miejscu z 34,4% udziału [2, 3].

Artykuł skupia się głównie na porównaniu Windows Presentation Foundation i Windows Forms pod względem zużycia pamięci RAM, procesora, a także szybkości wykonywanych operacji. Na potrzeby przeprowadzonej analizy zostało stworzonych osiem aplikacji. Mają one za zadanie wygenerować 1000 kontrolek (typu przycisk lub pola tekstowego) przy uruchomieniu programu, jak i odpowiednio 1000, 2000, 5000 podczas jej działania. Dodatkowo będzie badana szybkość generowania tych kontrolek.

Inne artykuły o podobnym temacie opisanym w tym artykule zwykle skupiają się na różnicach teoretycznych Windows Presentation Foundation i Windows Forms lub tylko zahaczają o tematykę wydajności [4, 5, 6].

2. Windows Presentation Foundation

Windows Presentation Foundation (WPF) jest częścią systemu operacyjnego Windows, która zapewnia graficzny interfejs użytkownika oraz środowisko dla aplikacji i usług. Aplikacje utworzone w tej technologii są uruchamiane tylko w systemie operacyjnym Windows. WPF jest częścią .NET Frameworks 3.0 dzięki czemu posiada narzędzia do tworzenia i obsługiwania grafiki oraz multimediów[7].

Windows Presentation Foundation (WPF) opiera się na języku xaml. Jest to język bardzo podobny do HTML, przez co jest łatwy do zrozumienia oraz bardzo łatwo jest opisać interfejs aplikacji [8]. Dzięki niemu łatwo jest rozdzielić logikę biznesową od interfejsu aplikacji dzięki zastosowaniu

wzorców projektowych takich jak Model-View-Controller (MVC) lub Model–View–ViewModel (MVVM) [9].

3. Windows Forms

Windows Forms jest nazwą graficznego interfejsu programowania aplikacji (API) i częścią pakietu Microsoft.NET Framework [10]. Zapewnia dostęp do macierzystych elementów interfejsu systemu operacyjnego Microsoft Windows. WinForms jest częścią środowiska .NET Framework, dzięki czemu umożliwia natywny dostęp do elementów interfejsu graficznego Microsoft Windows. Aplikacje napisane przy pomocy tej technologii bazują na zdarzeniach, co oznacza, że aplikacja oczekuje na działanie użytkownika takie jak np. kliknięcie na przycisk, wpisanie tekstu do pola tekstowego itp. Tworzenie interfejsu graficznego aplikacji w Windows Forms może przebiegać na dwa sposoby. Pierwszy i najłatwiejszy to przeciąganie kontrolek na ekran aplikacji oraz samodzielne ich ustawianie, drugi to pisanie całej kontrolki w warstwie kodu programu poprzez ustawienie rozmiaru, pozycji na ekranie, czy też jej nazwy [11].

4. Aplikacje testowe

Aplikację badające wydajność napisane są w Windows Presentation Foundation i Windows Forms.Są to programy do generowania kontrolek pól tekstowych oraz przycisków. Następna aplikacja służy do wyświetlania listy w kontrolce widoku listy(ListView). Dodatkowo została stworzona aplikacja wspomagająca pracę kierownika zespołu.

Pierwsze aplikacje testowe mają za zadanie generowanie 1000 razy danej kontrolki (pola tekstowego lub przycisku) podczas uruchomienia aplikacji oraz generowanie ich podczas uruchomionego już programu po naciśnięciu odpowiedniego przycisku.

Następna aplikacja polega na wyświetleniu przy uruchomieniu 1000 elementów w widoku listy (ListView). Implementacja tych list jest identyczna w obu technologiach.

Ostatnia aplikacja ma za zadanie wspomagać pracę kierownika. Posiada trzech użytkowników w systemie. Administrator może dodawać pracowników i ich edytować oraz nadawać uprawnienia. Kierownik może dodawać projekty oraz nowe zadania, do zadań można dodawać pracownika bezpośrednio przy tworzeniu ich lub już do istniejącego. Aplikacja korzysta z lokalnej bazy danych MS SQL oraz z Entity Framework 6.

Powyższe aplikacje zostały stworzone tak, aby jak najbardziej były podobne pod względem wizualnym, jak i pod względem implementacyjnym.

5. Metoda i platforma testowa

Aplikacje zostały sprawdzone pod względem szybkości wykonania operacji, zużycia pamięci RAM oraz procesora. Jedynie aplikacja wspomagająca pracę kierownika została zbadana tylko pod względem wykorzystania pamięci RAM i procesora.

Aplikacje służące do generowania zarówno pól tekstowych jak i przycisków mają za zadanie wygenerować podczas uruchomienia 1000 kontrolek. Podczas generowania zostanie mierzony czas tej operacji. Kolejnym zadaniem jest generowanie 1000, 2000 i 5000 kontrolek po naciśnięciu odpowiedniego przycisku. Czas generowania tych operacji również jest mierzony.

Następna aplikacja ma za zadanie wyświetlić listę 1000-elementową. Lista jest generowana w momencie uruchomienia programu, a następnie przekazywana do kontrolki widoku listy (ListView) w celu wyświetlenia ich.W tym przypadku sam czas wyświetlania jest liczony.

W ostatniej aplikacji wspomagającą pracę kierownika jest badane zużycie procesora oraz pamięci RAM. Wykonywane są jej podstawowe funkcjonalności, czyli logowanie się na konta użytkownika, dodawanie/edytowanie pracownika, dodawanie nowych zadań lub projektów.

Do testowania aplikacji wykorzystano sprzęt o parametrach podanych w tabeli 1. Testowanie aplikacji odbywało się przy minimalnym obciążeniu sprzętu przez inne zbędne aplikacje na czas testowania, między innymi program antywirusowy został wyłączony.

Tabela 1. Parametry sprzętu badawczego

Nazwa	Dell Inspirion 3542
System operacyjny	Windows 8.1
Procesor	Intel(R) Core(TM) i5-4210U CPU @ 1.70GHz
Pamięć RAM	4 GB
Dysk	SSD

Do lepszego zbadania wykorzystania podzespołów przez aplikację posłużono się programem Process Explorer [12]. Jest to program podobny do zwykłego menadżera zadań z większym zakresem możliwości. Pozwala lepiej monitorować zachowanie danego procesu.

6. Wyniki badań

Badania zostały przeprowadzone głównie pod kątem zużycia pamięci RAM, procesora oraz szybkości wykonania danej operacji. Pierwsze wyniki badań dotyczą aplikacji testowej wspomagającej pracę kierownika, gdzie badane było tylko zużycie procesora i pamięci RAM. Zostały one zaprezentowane w tabeli 2.

Tabela 2. Obciążenie systemu przy działającej aplikacji

Nazwa technologii	**Obciążenie procesora przez aplikację (%)**	**Obciążenie pamięci RAM przez aplikację (MB)**
WPF	12,30	68
WinForms	9,19	28,5

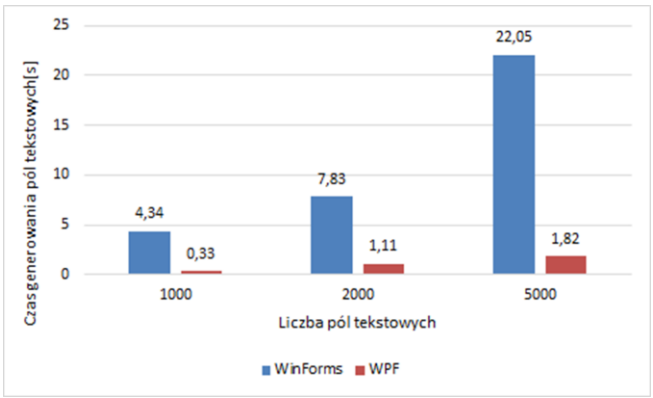
Wyniki uzyskane podczas generowania kontrolek pola tekstowego i przycisku zostały zebrane w Tabeli 3 i 4. Badane było obciążenie procesora, pamięci RAM i szybkość wykonanej operacji przy generowaniu 1000 kontrolek przy uruchomieniu aplikacji oraz wygenerowaniu 1000, 2000 i 5000 kontrolek już przy działającej aplikacji. Na rysunkach od 1 do 6 przedstawiono wykresy porównawcze uśrednionych wyników dla generowania kontrolek przy uruchomionym już programie.

Tabela 3. Średnie wyniki przy generowaniu 1000 Pól tekstowych przy uruchomieniu aplikacji WPF i WinForms

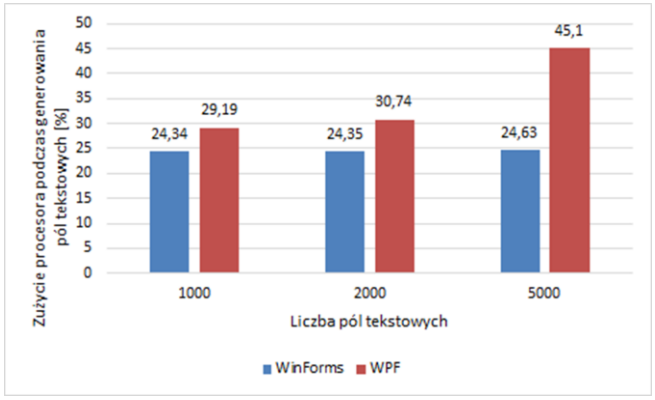
Nazwa technologii	**Obciążenie procesora przez aplikację (%)**	**Obciążenie pamięci RAM przez aplikację (MB)**	**Czas wygenerowania Pól tekstowych (s)**
Średnia dla WPF	29,12	65,96	0,33
Odchylenie standardowe dla WPF	±2,31	±0,11	±0,02
Średnia dla WinForms	23,24	17,48	0,55
Odchylenie standardowe dla WinForms	±0,99	±0,33	±0,06

Tabela 4. Średnie wyniki przy generowaniu 1000 przycisków przy uruchomieniu aplikacji WPF i WinForms

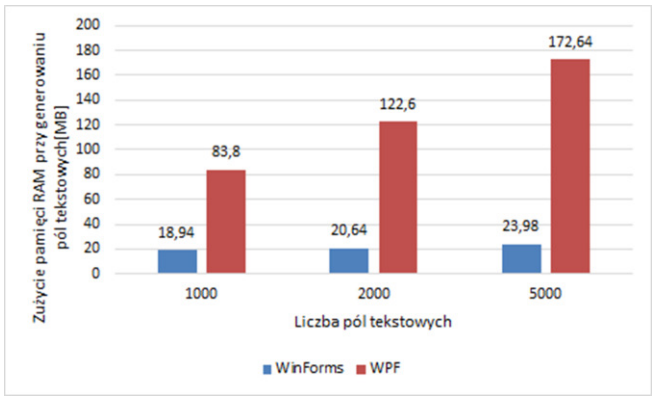
Nazwa technologii	**Obciążenie procesora przez aplikację (%)**	**Obciążenie pamięci RAM przez aplikację (MB)**	**Czas wygenerowania przycisków (s)**
Średnia dla WPF	18,38	49,82	0,09
Odchylenie standardowe dla WPF	±4,32	±0,22	±0,0
Średnia dla WinForms	14,17	14,14	0,31
Odchylenie standardowe dla WinForms	±2,81	±0,11	±0,01



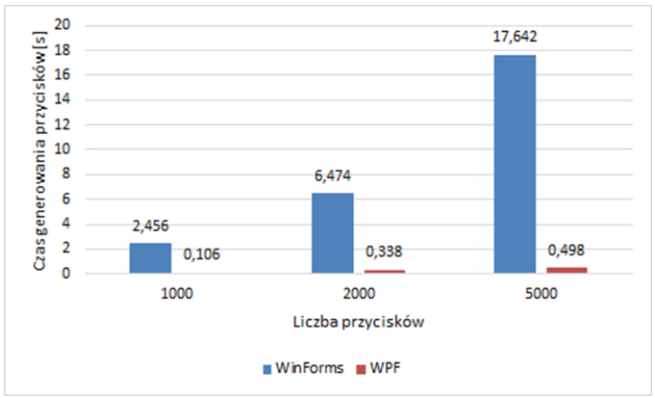
Rys. 1. Średni czas generowania Pól tekstowych



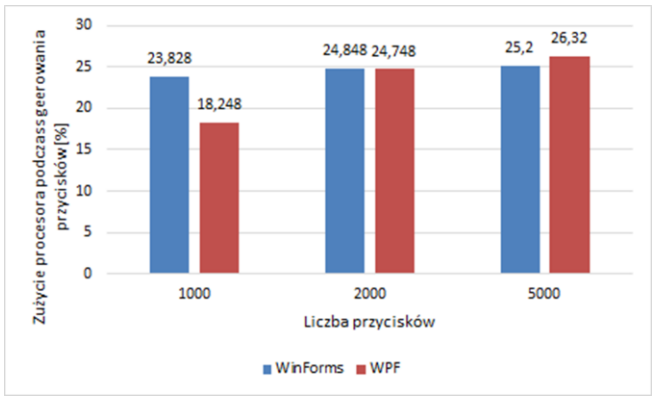
Rys. 2. Średnie zużycie procesora przy generowaniu Pól tekstowych



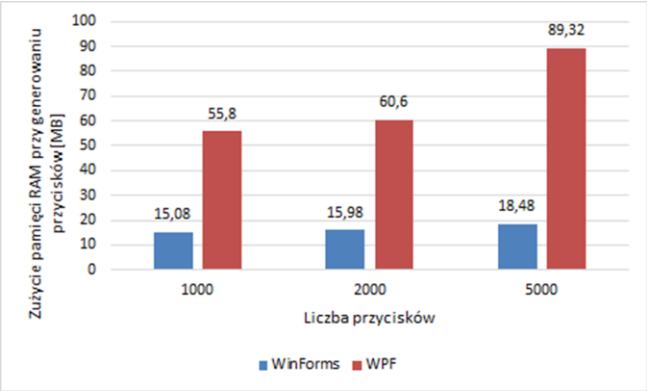
Rys. 3. Średnie zużycie pamięci RAM przy generowaniu Pól tekstowych



Rys. 4. Średnie czas generowania przycisków



Rys. 5. Średnie zużycie procesora przy generowaniu przycisków



Rys. 6. Średnie zużycie pamięci RAM przy generowaniu przycisków

Ostatnie wyniki dotyczą wyświetlania 1000 elementowej listy w komponencie ListView przy uruchomieniu aplikacji, gdzie sprawdzono czas wygenerowania listy oraz zużycie procesora i pamięci RAM. Uzyskane wyniki zgromadzono w tabeli 5.

Tabela 5. Średnie wyniki przy wyświetleniu 1000 elementowej listy w widoku listy (ListView) podczas uruchomienia aplikacji

Nazwa technologii	**Obciążenie procesora przez aplikację (%)**	**Obciążenie pamięci RAM przez aplikację (MB)**	**Czas wygenerowania listy w ListView (s)**
Średnia dla WPF	4,22	40,44	0,0136
Odchylenie standardowe dla WPF	±2,94	±0,17	±0,0
Średnia dla WinForms	22,88	13,7	0,0136
Odchylenie standardowe dla WinForms	±0,12	±0,0	±0,0

7. Dyskusja wyników

Na podstawie tabeli 2 wyraźnie widać przewagę WinForms nad WPF. Zużycie procesora przez WPF jest o 1/3 większy niż w WinForms i aż dwa razy większe jest zużycie pamięci RAM.

W tabeli, 3 jaki i 4 można zaobserwować większe zużycie pamięci RAM i procesora przez WPF. O ile przy zużyciu procesora różnica jest niewielka i wynosi tylko kilka procent, to przy pamięci RAM jest ona ponad trzykrotna. Jeśli chodzi o czas wykonania tej operacji to iw jednym i drugim przypadku lepiej wypadł WPF. WinForms przegrał tylko, jeśli chodzi o czas operacji dla przycisku i pola tekstowego wynosi 0,22s. W podobnych badaniach, jeśli chodzi o czas i zużycie RAM można zaobserwować podobne wyniki, jednak tam uwzględniono tylko jedną kontrolkę Pola tekstowego i nie wzięto pod uwagę zużycia procesora [13].

Badanie generowania kontrolek przy uruchomionym programie ukazują zużycie pamięci RAM, procesora i czasu operacji w zależności od liczby generowanych kontrolek. Na rysunku 1 i 4 widać przewagę technologii WPF w stosunku do WinForms. Przy 1000 kontrolkach WinForms jest wolniejszy w generowaniu tylko o 2 sekundy w przypadku generowania pola tekstowego i o 4 sekundy dla przycisku. Następnym obszarem porównania jest zużycie procesora. W tym przypadku to WinForms jest lepszy przy generowaniu Pola tekstowego. Zużycie zostaje prawie niezmienne w porównaniu do WPF, gdzie wyraźnie widać, że wraz ze wzrostem liczby generowanych ikonek zużycie procesora rośnie. Ciekawym przypadkiem jest generowanie przycisku, gdzie można zaobserwować przewagę technologii WPF dla 1000 kontrolek. Następnie przy 2000 kontrolek zużycie procesora prawie się zrównało. Na końcu dla 5000 kontrolek to WinForms okazuje się lepszy. W WinForms zużycie niemal pozostaje na tym samym poziomie nieznacznie wzrastając jak dla Pola tekstowego. WPF podobnie jak dla Pola tekstowego widać wzrost zużycia procesora w zależności od liczby wygenerowanych komponentów. Na rysunku 3 i 6 przedstawione zostało zużycia procesora. Widoczna jest przewaga technologii WinForms. Niezależnie od liczby wygenerowanych komponentów wzrost zużycia jest dosyć mały w porównaniu do wzrostów zużycia przez WPF. WPF zużywa dużo więcej pamięci oraz posiada większą różnicę między generowanymi kontrolkami w stosunku do WinForms.

W tabeli 5 można zaobserwować przewagę WPF pod względem mniejszego zużycia procesora, a WinForms ma mniejsze zużycia pamięci RAM. Natomiast czas operacji jest równy dla obu technologii.

WPF osiągnął lepsze wyniki przy zużyciu procesora dla generowaniaprzycisku 1000 i 2000 razy w porównaniu do generowania kontrolki Pola tekstowego. Może to być spowodowane pracą środowiska testowego lub czynnikami niezależnymi od aplikacji.

8. Wnioski

Praca aplikacji w dużej mierze zależy od środowiska, na którym jest uruchamiana. Generowanie dużej liczby kontrolek w aplikacji zawsze obciąża słabsze środowisko uruchomieniowe. Dlatego większość aplikacji desktopowych rezygnuje z dużej liczby kontrolek wyświetlanych w jednym oknie. Zamiast tego wybiera się mniejszą liczbę kontrolek i możliwości przejścia między oknami aplikacji.

Na podstawie przeprowadzonych badań można wywnioskować, że WinForms sprawdza się lepiej pod względem zużycia pamięci RAM i procesora. Zużycie ich jest stosunkowo małe w porównaniu do WPF. Podczas generowania komponentów w WinFormsmniej obciąża środowisko uruchomieniowe. Obciążenie to utrzymuje się na podobnym poziomie niezależnie od ich liczby. WPF, natomiast sprawdza się lepiej, jeśli chodzi o szybkość. We wszystkich badanych operacjach, oprócz wczytywania listy, był szybszy od WinForms. Podsumowując badania, WinForms znacznie lepiej sprawdzałby się w aplikacjach, w których zależy użytkownikowi na mniejszym zużyciu procesora, jak i pamięci RAM. WPF sprawdza się lepiej, jeśli w aplikacji chodzi o szybkość wykonania danego zadania.

Literatura

[1] Nora Georgieva, Survey Report: Who is the .NET Developer of 2016?, https://www.telerik.com/blogs/survey-report-the-dotnet-developer-of-2016 [06.10.19]

[2] Developer Survey Results 2018, https://insights.stackoverflow.com/survey/2018/ [06.10.2019]
[3] Developer Survey Results 2017, https://insights.stackoverflow.com/survey/2017/ [06.10.2019]
[4] Anurag Misra, Use of Windows Presentation Foundation and Windows Forms in Windows Application Programming, Volume 7, No. 7, Nov-Dec 2016 International Journal of Advanced Research in Computer Science
[5] Winforms vs WPF, https://www.educba.com/winforms-vs-wpf/ [06.10.2019]
[6] WPF vs. WinForms, https://www.wpf-tutorial.com/pl/2/o-wpf/wpf-vs-winforms/ [06.10.2019]
[7] Windows Presentation Foundation (WPF), Margaret Rouse, https://searchwindevelopment.techtarget.com/definition/Windows-Presentation-Foundation [06.10.2019]
[8] Wei-Meng Lee, An Overview of Windows Presentation Foundation, Codemagazine 03/04 2006,
[9] Paul D. Sheriff, Why Use WPF?, Codemagazine 11/12 2009
[10] Sells, Chris, Windows Forms Programming in C# (1st ed.). Addison-Wesley Professional. p. xxxviiii.
[11] https://www.worddisk.com/wiki/WinForms/ [06.10.2019]
[12] Mark Russinovich, Process Explorer v16.30, https://docs.microsoft.com/en-us/sysinternals/downloads/process-explorer [06.10.2019]
[13] yashsoman, Window Form Controls v/s WPF Controls Memory Comparison, http://www.codeproject.com/Articles/1038077/Window-Form-Controls-v-s-WPF-Controls-MemoryCompa [06.10.2019]

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 14 (2020) 31-36
Wysłane: 2019-10-14
Przyjęte: 2019-10-18

Wpływ języka programowania aplikacji chmurowej na wydajność jej implementacji w wybranych środowiskach serverless

Krzysztof Bezrąk*, Sławomir Przyłucki
Politechnika Lubelska, Katedra Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. Ostatnie lata rozwoju technologii chmurowych przyniosły gwałtowny wzrost zainteresowania rozwiązaniami określanymi jako systemy bezserwerowe. Ich wydajność, a tym samym przydatność w potencjalnych zastosowaniach, jest silnie uzależniona od sposobu implementacji programowej konkretnych zadań. W artykule poddano analizie wpływ wybranych, obecnie najpopularniejszych, języków programowania na wydajność testowej infrastruktury bezserwerowej uruchomionej w środowisku zarządzanym przez system Kubernetes. Zgromadzone dane posłużyły do sformułowania wniosków dotyczących przydatności poszczególnych języków w warunkach zróżnicowanych obciążeń systemu bezserwerowego.

Słowa kluczowe: Kubernetes; serverless; Kubeless

*Autor do korespondencji.

Adres e-mail: krzysztof.bezrak@pollub.edu.pl

Impact of the cloud application programming language on the performance of its implementation in selected serverless environments

Krzysztof Bezrąk*, Sławomir Przyłucki
Deaprtment of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract. Recent years of cloud technology development have brought a sharp increase in interest in solutions known as serverless systems. Their performance, and thus usefulness in potential applications, strongly depends on the method of program implementation of specific tasks. The article analyzes the impact of selected, currently the most popular, programming languages on the performance of the serverless test infrastructure running in an environment managed by the Kubernetes system. The collected data were used to formulate conclusions regarding the suitability of individual languages in the conditions of varying serverless system loads.

Keywords: Kubernetes; serverless; Kubeless

*Corresponding author.

E-mail address/addresses: krzysztof.bezrak@pollub.edu.pl

1. Wstęp

Systemy chmurowe stały się nieodłącznym komponentem współczesnych systemów informatycznych i są tym samym powszechnie wykorzystywane jako środowiska programistyczne zarówno w dużych korporacjach, jak i małych firmach informatycznych. Dla tych drugich jednak często koszty rozwiązań chmurowych oraz trudność ich wdrożenia stanowiły ogromną barierę, co było jednym z powodów dla których powstała technologia systemów bezserwerowych (ang. serverless). Dostarcza ona skalowalne środowiska uruchomieniowe aplikacji, w których płaci się tylko za wykorzystane przez nią zasoby i czas jej działania. Czynnikiem bezpośrednio wpływającymi na powyższe parametry jest sposób implementacji programowej danego zadania. Ponieważ w chwili obecnej wybór języków jest uzależniony od wielu czynników i nie ogranicza się do jednego, dedykowanego rozwiązania, celowym jest identyfikacja wpływu wyboru języka na wydajność systemów serverless w różnych scenariuszach ich użycia.

2. Cel i teza

Celem artykułu jest zbadanie wpływu języka programowania danej aplikacji napisanej w technologii serverless na jej wydajność. Na jego potrzeby zostały przygotowane zbliżone do siebie aplikacje wykonujące podobny algorytm w wybranych językach programowania. W ramach badania zostały przeprowadzone testy polegające na uruchamianiu tychże aplikacji i pomiarze czasów ich odpowiedzi przy różnych scenariuszach obciążenia. Analiza wyników powyższych testów w powiązaniu z wiedzą teoretyczną na temat języków programowania i środowisk uruchomieniowych aplikacji umożliwiła wyciągnięcie i uogólnienie wniosków w obszarze zdefiniowanym celem badawczym.

Badanie zostało przeprowadzone na serwerze z zainstalowanym systemem Kubernetes, w którym zostanie uruchomione środowisko Kubeless. System Kubernetes został wybrany ze względu na wiodącą rolę w zakresie zarządzania skonteneryzowanymi aplikacjami, a takimi są aplikacje serverless i rosnącą popularność [1]. Kubeless jest jednym z rozwiązań umożliwiających obsługę aplikacji serverless w systemie Kubernetes, jego wybór uwarunkowała największa

ilość wspieranych języków programowania w porównaniu z konkurencją, a także wsparcie komercyjne zapewniane przez firmę Bitnami. Serwer w miarę możliwości nie będzie zawierał dodatkowego oprogramowania poza tym, które będzie potrzebne dla systemu Kubernetes aby wyeliminować czynniki mogące potencjalnie wpłynąć na wyniki pomiarów. Serwer będzie podłączony do Internetu i do sieci lokalnej.

Na powyższym serwerze umieszczone zostaną aplikacje serverless napisane w poniższych językach programowania:

- Golang,
- Java,
- JavaScript,
- Python.

Języki te zostały wybrane ze względu na popularność w ostatnim czasie, szczególnie w platformach serverless. Różnią się też od siebie pod względem architektury - JavaScript i Python to języki skryptowe, interpretowalne, Golang kompiluje się statycznie do postaci kodu maszynowego, a Java kompiluje się do kodu bajtowego wykonywanego przez maszynę wirtualną (JVM). Wszystkie są też wspierane przez wybraną platformę serverless – Kubeless.

W tej samej sieci lokalnej co serwer będzie podłączony drugi komputer z którego będą wykonywane pomiary. Wynika to z potrzeby zminimalizowania ewentualnego, negatywnego wpływu innych aktywnych urządzeń i powiązanych z nimi usług na rezultaty pomiarów. Pomiary będą wykonywane przy pomocy narzędzia Apache Benchmark (ab), które służy do mierzenia wydajności aplikacji komunikujących się poprzez protokół HTTP. O jego wyborze zadecydowała możliwość definiowania ilości równoczesnych połączeń z serwerem a także sposób prezentowania rezultatów pomiaru.

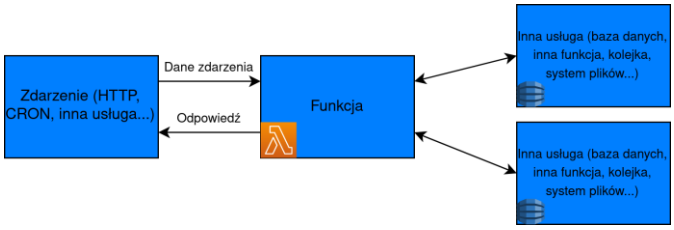
3. Systemy bezserwerowe

3.1. Cechy charakterystyczne systemów bezserwerowych

Systemy bezserwerowe dostarczają usługi serwerowe rozliczane na podstawie ich rzeczywistego zużycia, uruchamiane na serwerach dostawcy tychże usług. Cechują się one niskim nakładem pracy związanej z zarządzaniem infrastrukturą, wysoką skalowalnością i dostępnością. Takie podejście tworzy alternatywę dla standardowych wdrożeń aplikacji, w których poza napisaniem kodu należy także stworzyć i utrzymać środowisko na którym będzie ona uruchomiona. Przeważnie dostawcy takich usług rozliczają klienta bazując na pojedynczych uruchomieniach aplikacji, dzięki czemu nie trzeba płacić za ciągłe utrzymywanie serwerów gdy nie są wykorzystywane. Systemy serverless mają zastosowanie szczególnie w procesach, które są uruchamiane na podstawie określonego wyzwalacza, np. wysłanie powiadomienia na e-mail po wysyłce formularza kontaktowego, czy cotygodniowe usuwanie nieaktywnych użytkowników systemu [2].

3.2. Architektura systemów bezserwerowych

Typowa architektura systemu bezserwerowego opiera się na komponentach wykorzystywanych w innych usługach chmurowych, najczęściej od tego samego dostawcy, takimi jak baza danych, kolejka wydarzeń czy system plików. Do każdego wywołania funkcji napisanej w architekturze bezserwerowej jest przekazywany obiekt zawierający dane przekazane przez jej wyzwalacz, którym może być żądanie HTTP czy inna usługa (np. dodanie nowego użytkownika do bazy danych może wywołać funkcję wysyłki maila powitalnego). Przykład takiej architektury przedstawiony jest na rysunku 1.



Rys. 1. Przykładowa architektura systemu bezserwerowego

W odróżnieniu od tradycyjnej architektury monolitycznej, łatwo jest określić źródło wywołujące daną funkcję, co może pomóc w określeniu procesów biznesowych jakie dana funkcja ma spełniać.

3.3. Przegląd języków programowania

W procesie tworzenia systemów bezserwerowych często wykorzystywane są języki interpretowalne, takie jak Python czy JavaScript, głównie ze względu na ich rosnącą popularność wśród programistów i szybkość wdrażania zmian . W przypadku potrzeby wydania nowej wersji funkcji nie jest wymagana kompilacja kodu źródłowego, co upraszcza proces uruchamiania danej funkcji.

Język Python jest często wykorzystywany do analizy danych i uczenia maszynowego. W przypadku takich aplikacji architektura systemów bezserwerowych może w znaczącym stopniu zmniejszyć koszty związane z zapotrzebowaniem na wysoką moc obliczeniową. Najpopularniejsze środowisko uruchomieniowe JavaScript, Node.js, jest natomiast wydajne w komunikacji asynchronicznej z zewnętrznymi zasobami, a w repozytorium pakietów można znaleźć dużą ilość gotowych integracji z popularnymi usługami chmurowymi, co również czyni go dobrym kandydatem do wykorzyatania w środowiska bezserwerowych.

Innym przykładem języka programowania stosowanego w systemach bezserwerowych jest Golang. Jest to język komplilowalny do postaci kodu maszynowego, dzięki czemu jest wydajny w zadaniach wymagających dużej mocy obliczeniowej. Czas wykonania jest w systemach bezserwerowych istotny, gdyż rozliczane są wykorzystane zasoby w trakcie wykonywania funkcji. Ze względu na stałą popularność w zastosowaniach serwerowych, w takich systemach popularna jest również Java. Jednym z powodów

jest łatwość przepisywania starych, monolitycznych systemów informatycznych na architekturę serverless.

4. System Kubernetes

4.1. Znaczenie i rola kontenerów programowych

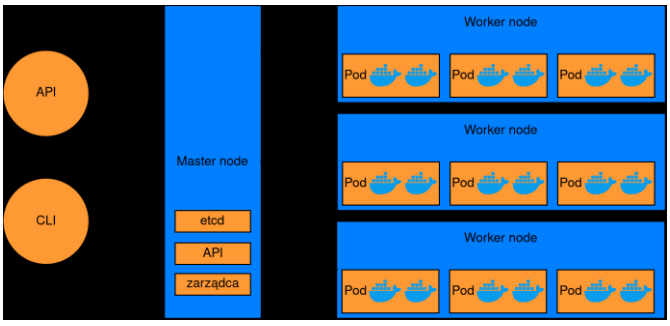
Kontenery programowe oferują izolację usług i aplikacji realizowaną na poziomie systemu operacyjnego. Zapewniają dzięki temu bezpieczeństwo i skalowalność, przy zachowaniu wydajności dorównującej systemowi gospodarza dzięki zastosowaniu lekkiej wirtualizacji.

Najpopularniejszą technologią wykorzystującą kontnerery programowe jest Docker, który cechuje się tworzeniem obrazów aplikacji zawierających ich środowisko uruchomieniowe, w tym system operacyjny, opisywanych za pomocą pliku tekstowego (Dockerfile). Za pomocą tego pliku można tworzyć odtwarzalne i przenośne kompilacje takich obrazów, które mogą być później uruchamiane i połączone ze sobą w wirtualnych sieciach. Dzięki zastosowanej architekturze pomiędzy kontenerami i systemem operacyjnym są współdzielone jedynie jądro systemu i zasoby sprzętowe, co pozwala na osiągnięcie wysokiej wydajności [3].

4.2. System Kubernetes

Najczęściej jako alternatywę do architektury monolitycznej stosuje się obecnie architekturę opartą o oddzielne mikroserwisy, które realizują pojedynczą potrzebę biznesową [4]. Są one wdrażane przez osobne kontenery, które muszą się ze sobą komunikować, co przy większej ilości kontenerów i potrzebie wykorzystania wielu fizycznych serwerów może stanowić kłopot.

Kubernetes powstał w celu rozwiązania tych problemów. Stanowi on rolę nadzorcy nad kontenerami tworząc dla nich izolowane, wirtualne sieci, niezależnie od serwerów na których są uruchamiane. Zarządza ich cyklem życia, monitoruje zużycie zasobów i pozwala na wzajemną komunikację, co czyni go kompleksowym narzędziem do zarządzania nowoczesnymi systemami informatycznymi jak systemy mikroserwisowe czy bezserwerowe [5].

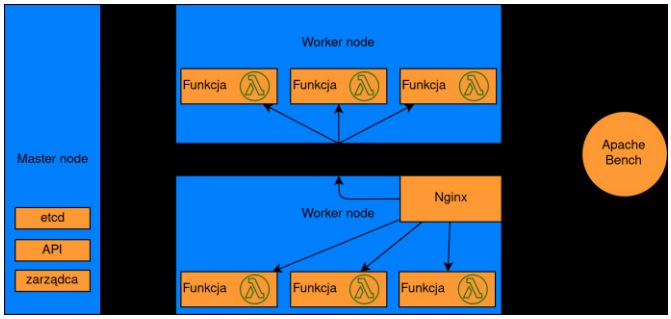


Rys. 2. Architektura systemu Kubernetes

Na rysunku 2 przedstawiona jest typowa architektura systemu Kubernetes. Podstawowym jej elementem jest węzeł główny (ang. master node), który jest zarządcą całego systemu. Zawiera on między innymi moduł etcd, który przechowuje dane konfiguracyjne systemu, interfejs REST API do komunikacji zewnętrznej i wewnętrznej oraz zarządcę przydzielającego kontenery do poszczególnych serwerów. Do niego podłączone są węzły robocze (ang. worker node), na których uruchomione są kontenery ulokowane w jednostki nazwane podami.

5. Testy wydajności systemu bezserwerowego

Na komputerze stacjonarnym z zainstalowanym systemem Ubuntu 19.04 zostało utworzone środowisko testowe. Składa się ono z klastra Kubernetes uruchomionego w oparciu o narzędzie kind. Klaster ten zawiera jeden węzeł główny i dwa węzły robocze. Na nim zostało uruchomione środowisko serverless w postaci frameworka Kubeless, serwer HTTP Nginx udostępniający funkcje na zewnątrz klastra, oraz moduł metrics-server służący do pomiaru zużyciu zasobów na potrzeby autoskalowania. Do pomiaru wydajności został użyty program Apache Benchmark, uruchamiany na systemie gospodarza. Rysunek 3 ilustruje architekturę środowiska testowego.



Rys. 3. Przykładowa architektura systemu bezserwerowego

5.1. Aplikacja testowa

Aplikacja testowa ma za zadanie wyznaczenie *n*-tego wyrazu ciągu Fibonacciego. Ciąg Fibonacciego jest wyrażony następującym wzorem:

$$F_n = \begin{cases} 0 & \text{dla } n=0 \\ 1 & \text{dla } n=1 \\ F_{n-1}+F_{n-2} & \text{dla } n>1 \end{cases} \quad (1)$$

Na potrzeby tego zadania został wyznaczony algorytm, który w sposób rekurencyjny znajduje podany wyraz ciągu. Schemat algorytmu został przedstawiony poniżej:

Przykład 1. Algorytm wyliczania n-tego wyrazu ciągu Fibonacciego

```
procedure Fibonacci(n)
  if n > 1 then return Fibonacci(n − 1) + Fibonacci(n − 2)
  else if n = 1 then return 1
  return 0
```

Algorytm ten zostanie zaimplementowany we wszystkich wspomnianych poprzednio językach programowania. Aplikacje będą otrzymywać parametr *n* za pomocą zdarzenia („event”) jako parametr *n*, a jako wynik będą zwracać rezultat działania algorytmu.

Algorytm ten został wybrany ze względu na złożoność obliczeniową proporcjonalną do wartości parametru *n*, która pozwoli zarówno na zbadanie działania aplikacji przy minimalnej ilości obliczeń (np. dla *n = 0*), jak również przy wysokim obciążeniu procesora (np. dla *n = 40*). Jest on także niezłożony i prosty w implementacji w każdym z badanych w ramach tej pracy języków programowania.

Przykład implementacji tego algorytmu w aplikacji serverless dla języka JavaScript można zobaczyć poniżej.

Przykład 2. Aplikacja testowa w języku JavaScript

```
function fib(n) {
  if (n < 2) {
    return n;
  } else {
    return fib(n - 1) + fib(n - 2);
  }
}

module.exports = {
  handler: (event, context) => {
    var n = parseInt(event.data.n, 10);
    return fib(n);
  },
};
```

W powyższej aplikacji można zwrócić uwagę na dwie części: najpierw jest zdefiniowana funkcja implementująca opisywany wcześniej algorytm. Następnie mamy obsługę zdarzenia („eventu") wywołanego za pomocą wyzwalacza („triggera"). Ze zdarzenia jest wyznaczany parametr *n*, który jest konwertowany do typu liczbowego metodą *parseInt* (na wypadek gdyby był zdefiniowany jako ciąg znaków). Następnie zostaje on przekazany do funkcji obliczającej *n*-ty wyraz ciągu Fibonacciego, której wynik jest zwracany jako odpowiedź aplikacji na dane zdarzenie.

5.2. Scenariusze testów

W ramach badań zostały wykonane trzy scenariusze testowe skupiające się na różnych obszarach analizy działania środowiska serverless. W ramach każdego z nich uruchomiono funkcje napisane w zdefiniowanych wcześniej językach poddawanych analizie i rejestrowano czas ich odpowiedzi. Funkcje były wywoływane poprzez protokół HTTP za pomocą narzędzia Apache Benchmark. W ramach testów zmieniano ilość replik funkcji oraz liczbę N dla której jest wyliczany wyraz ciągu Fibonacciego. Apache Benchmark wykonywał zapytania 1000 razy i ze zgromadzonych wyników została wyliczona średnia arytmetyczna. Każdy z testów był powtarzany trzykrotnie ze zmienną liczbą równoczesnych zapytań do serwera - odpowiednio z jednym, pięcioma i dwudziestoma pięcioma zapytaniami.

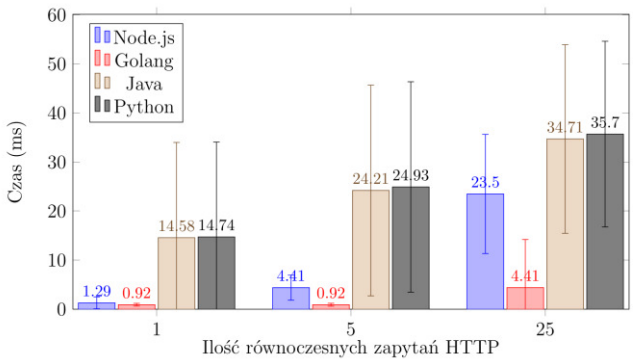
Pierwszy test był przeprowadzony dla jednej repliki każdej z funkcji i parametru N = 1. Zgodnie z zaimplementowanym algorytmem wyliczania n-tego wyrazu ciągu Fibonacciego nie wymagał on rekurencyjnych wywołań funkcji, a w rezultacie funkcje powinny zwrócić 1 bez wykonywania obliczeń. Dzięki temu testowi określono wpływ samego środowiska w którym uruchamiana jest funkcja na całkowity czas jej wykonywania, (ponieważ funkcja nie wykonywała żadnych obliczeń).

Drugi test również wykorzystywał jedną replikę funkcji, natomiast parametr N był równy 25. Pozwoliło to zaobserwować działanie funkcji przy dużym obciążeniu procesora, ponieważ w ramach jednego wywołania niezbędne było rekurencyjne, wielokrotne odwołanie do funkcji, co bezpośrednio wynika z zastosowanego algorytmu. Parametr N dla tego testu został wyznaczony manualnie, weryfikując czy funkcje obciążają procesor w wystarczająco dużym stopniu.

Trzeci test łączył dwa poprzednie testy, a zmieniana była ilość replik do 3 i 10. Rezultatem tego testu są 4 wykresy, czyli powtórzenie pierwszego testu dla trzech i dziesięciu replik, oraz analogiczne powtórzenie drugiego testu dla trzech i dziesięciu replik. Dzięki temu określony został wpływ skalowania ilości replik funkcji na ich wydajność wyrażoną poprzez pełen czas odpowiedzi serwera.

5.3. Wyniki testów

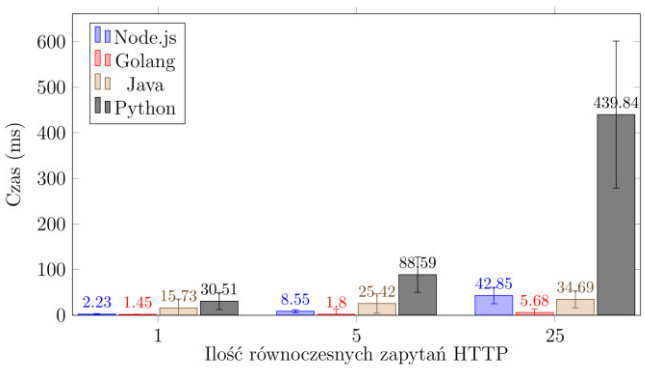
W pierwszym teście dla każdego przypadku testowego wszystkie zapytania HTTP wykonane przez Apache Benchmark się powiodły. Test opiera się na wykonaniu najprostszego możliwego scenariusza dla jednej repliki każdej z funkcji. Oczekiwanym rezultatem były średnie czasy odpowiedzi o wartościach dużo niższych niż w teście drugim. Na rysunku 4 przedstawione są rezultaty pierwszego testu.



Rys. 4. Rezultaty testów dla jednej repliki funkcji i niskiego obciążenia procesora

Otrzymane wyniki wskazują, że ilość równoczesnych zapytań HTTP ma proporcjonalny wpływ na średni czas wykonania zapytania w przypadku Node.js i Golang, ale dla funkcji w języku Java i Python wyniki są istotnie gorsze, co sugeruje negatywny wpływ tych języków na wydajność testowego systemu.

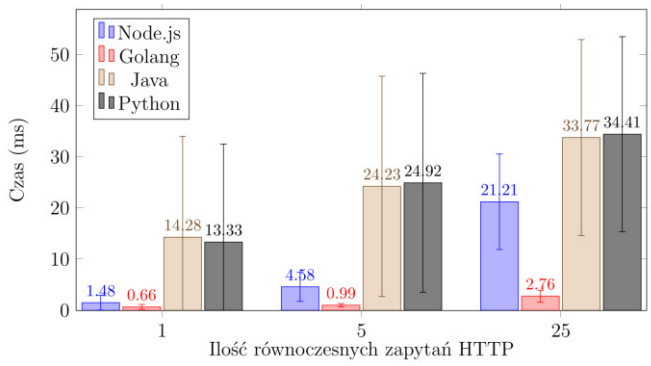
Drugi test został wykonywany w analogiczny sposób co pierwszy. Zmieniła się tylko liczba N przekazywana do funkcji, co powinno mieć wpływ na obciążenie procesora, a tym samym na czasy odpowiedzi funkcji. Ten test również się powiódł i jego wyniki są pokazane na rysunku 5.



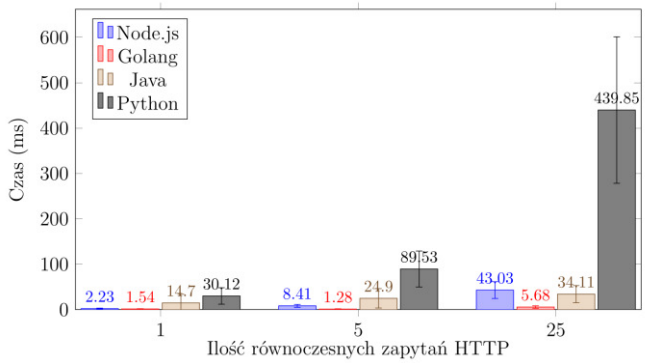
Rys. 5. Rezultaty testów dla jednej repliki funkcji i wysokiego obciążenia procesora

Z otrzymanych rezultatów tego testu wynika, że funkcja napisana w języku Python przy większym obciążeniu procesora jest nawet dwunastokrotnie wolniejsza od funkcji napisanej w języku Java. W przypadku Go natomiast nie zaobserwowano tak dużego wpływu dodatkowych obliczeń spowodowanych większym parametrem N w porównaniu do innych języków. Jest to prawdopodobnie spowodowane specyfiką tego języka, gdyż jest kompilowany do kodu w postaci binarnej.

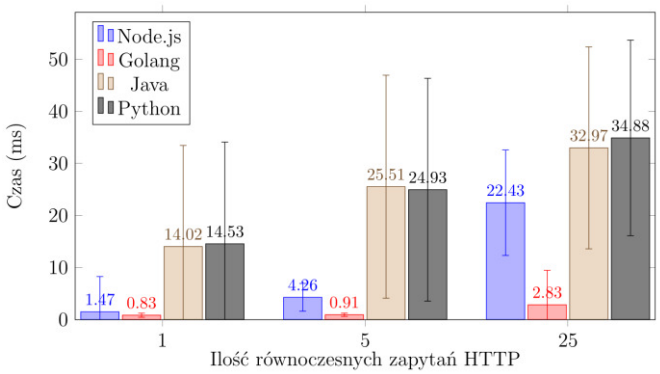
W przypadku trzeciego testu uwaga została poświęcona skalowalności funkcji w przypadku uruchamiania wielu jej instancji. Test wykonał się pomyślnie, a jego rezultaty są przedstawione na rysunkach od 6 do 9.



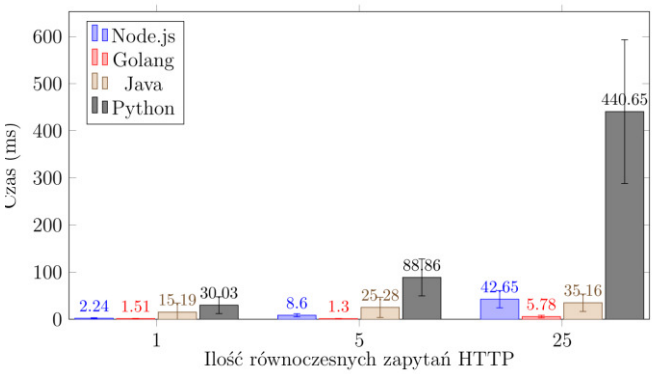
Rys. 6. Rezultaty testów dla trzech replik funkcji i niskiego obciążenia procesora



Rys. 7. Rezultaty testów dla trzech replik funkcji i wysokiego obciążenia procesora



Rys. 8. Rezultaty testów dla dziesięciu replik funkcji i niskiego obciążenia procesora



Rys. 9. Rezultaty testów dla dziesięciu replik funkcji i wysokiego obciążenia procesora

Na podstawie powyższych rezultatów można stwierdzić, że we wszystkich przypadkach zarejestrowano największe wartości czasów potrzebnych na wykonanie funkcji w języku Python. W przypadku Node.js i Java ilość replik miała bardzo mały wpływ na czas odpowiedzi, natomiast dla Go widać krótsze czasy odpowiedzi przy zwiększaniu ilości replik.

6. Wnioski

W celu końcowej weryfikacji opracowane funkcje zostały uruchomione na tym samym komputerze, na którym zaimplementowano środowisko testowe, ale tym razem bez środowiska serverless. Pozwoliło to na porównanie ich wydajności z pominięciem wszystkich czynników powiązanych z systemem chmurowym. Funkcja napisana w języku Golang skompilowana do postaci binarnej wykonywała się najszybciej. Z kolei, o 25% wolniej od niej wykonywała się ta funkcja w języku Java skompilowana do kodu bajtowego i uruchamiana w maszynie wirtualnej JVM. Funkcja w języku JavaScript wykonywana w środowisku Node.js była 3x wolniejsza od Golang. Python okazał się najwolniejszym ze wszystkich funkcji wykonując kod 35x wolniej niż Golang. Te porównawcze wyniki pokrywają się z charakterystyką testowanych języków i odzwierciedlają ich wydajność.

Patrząc na rezultaty testów widać, że najszybciej wykonywały się zawsze funkcje napisane w języku Golang. Zmiana parametrów miała też w przypadku tego języka

proporcjonalny wpływ na rezultaty, a ich odchylenie standardowe było małe. Oznacza to, że jego zachowanie w środowisku serverless jest stabilne i przewidywalne, jednocześnie będąc zdecydowanie szybsze od pozostałych testowanych języków. Język ten był również najszybszy w testach wykonanych bez środowiska serverless.

Node.js w wynikach testów zwykle znajdował się na drugiej pozycji, za wyjątkiem testów z parametrem N=25 i ilością równoczesnych zapytań równą 25, w których spadał na trzecią pozycję. Można z tego wywnioskować, że nie jest najlepszym wyborem do implementacji funkcji pod dużym obciążeniem, które wykonują intensywne obliczenia. W pozostałych testach nie odstawał on w znacznym stopniu od języka Golang. Środowisko serverless może mieć znaczący wpływ na wyniki pod obciążeniem, zważywszy na fakt, że rezultaty testów porównawczych bez środowiska serverless nie wykazywały tak dużych różnic. Wyniki czasów odpowiedzi funkcji zaimplementowanej w tym języku również nie charakteryzowały się dużym odchyleniem standardowym.

W przypadku funkcji napisanej w języku Java zaobserwowano, że pomiędzy wynikami z testów pierwszego i drugiego nie ma dużych różnic, w odróżnieniu od pozostałych języków. Jedną z możliwych przyczyn takiego stanu rzeczy jest długi czas potrzebny na samo uruchomienie funkcji w środowisku serverless, przy jednoczesnym szybkim wykonywaniu obliczeń, których czas w rezultacie jest niezauważalny. Potwierdzają to także rezultaty porównawczych wyników testów bez środowiska serverless. Każde zapytanie przy jednym równoczesnym połączeniu zajmowało średnio minimum 15 milisekund, rosnąc razem z liczbą równoczesnych zapytań, co może sugerować, że Java w środowisku serverless może mieć zastosowanie dla rzadko wykonywanych funkcji które wykonują zaawansowane obliczenia. Uzyskane wyniki testów cechują się też w tym przypadku dużym odchyleniem standardowym.

Funkcja napisana w języku Python była najwolniejsza w niemal wszystkich przypadkach testowych. Przy małej ilości obliczeń czasy są zbliżone do Javy, ale przy dużym obciążeniu procesora Python jest wielokrotnie wolniejszy od reszty badanych języków. Rezultaty otrzymane pod dużym obciążeniem pokrywają się z rezultatami porównawczych wyników testów bez środowiska serverless. Odchylenie standardowe w przypadku aplikacji w języku Python, podobnie jak w języku Java, było bardzo duże, co podkreśla niestabilność zachowania w środowisku serverless. Biorąc zatem pod uwagę czas wykonywania funkcji można dojść do wniosku, że w gronie badanych języków jest on najgorszym wyborem dla środowiska serverless.

Literatura

[1] Everything You Need to Know about Containers, Part III: Orchestration with Kubernetes, https://www.linuxjournal.com/content/everything-you-need-know-about-containers-part-iii-orchestration-kubernetes [14.10.2019]

[2] What Is Serverless Computing, https://www.cloudflare.com/learning/serverless/what-is-serverless/ [10.10.2019]

[3] Jarosław Krochmalski, Docker: projektowanie i wdrażanie aplikacji, Wydawnictwo Helion (2017), 16-18

[4] Łukasz Wróbel, Trzy powody, dla których warto przejść z architektury monolitycznej na mikroserwisową, PC World Komp-Wiadomości, 2018.

[5] Jonathan Baier, Getting Started with Kubernetes, Packt Publishing (2015), 6-7

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 14 (2020) 37-42
Wysłane: 2019-10-10
Przyjęte: 2019-10-18

Zastosowanie maszyny wektorów nośnych w sterowaniu sygnalizacją świetlną

Artur Całuch*, Adam Cieślikowski*, Małgorzata Plechawska-Wójcik
Politechnika Lubelska, Katedra Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. Niniejszy artykuł przedstawia proces dostosowania parametrów modelu maszyny wektorów nośnych, który posłuży do zbadania wpływu wartości parametru długości cyklu sygnalizacji świetlnej na jakość ruchu. Badania przeprowadzono z użyciem danych pozyskanych w trakcie przeprowadzonych symulacji w autorskim symulatorze ruchu ulicznego. W artykule przedstawiono i omówiono wyniki poszukiwania optymalnej wartości parametru długości cyklu sygnalizacji świetlnej.

Słowa kluczowe: uczenie maszynowe; symulator ruchu ulicznego; maszyna wektorów nośnych

*Autor do korespondencji.

Adresy e-mail: arturcaluch@gmail.com, a.cieslikowski95@gmail.com

Application of support vector machine in a traffic lights control

Artur Całuch*, Adam Cieślikowski*, Małgorzata Plechawska-Wójcik
Department of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract. This article presents the process of adapting support vector machine model's parameters used for studying the effect of traffic light cycle length parameter's value on traffic quality. The survey is carried out using data collected during running simulations in author's traffic simulator. The article shows results of searching for optimum traffic light cycle length parameter's value.

Keywords: machine learning; traffic simulator; support vector machine

*Corresponding author.

E-mail addresses: arturcaluch@gmail.com, a.cieslikowski95@gmail.com

1. Wprowadzenie

Proces tworzenia infrastruktury drogowej jest niezwykle skomplikowany. Jednym z najistotniejszych etapów jest zaplanowanie szeroko pojętej organizacji ruchu, na którą, w głównej mierze, składa się sygnalizacja świetlna. Ma ona największe znaczenie w kontekście komfortu korzystania z powstającej infrastruktury - źle skonfigurowana będzie miała negatywny wpływ na jakość ruchu. Ustawienie sygnalizacji, utworzone na etapie planowania, w większości przypadków nie zdaje egzaminu, ponieważ nie można przewidzieć warunków drogowych bez obserwacji ruchu i analizy zebranych w ten sposób danych. Z pomocą przychodzą wszelkiego rodzaju symulatory oraz algorytmy uczenia maszynowego, z pomocą których można stworzyć bardzo wydajne konfiguracje sygnalizacji świetlnej.

W niniejszym artykule przedstawiono przegląd istniejących rozwiązań, proces adaptacji modeli maszyny wektorów nośnych, które odpowiadały za dostosowanie optymalnej konfiguracji sygnalizacji świetlnej w autorskim symulatorze ruchu ulicznego oraz wyniki badań nad optymalizacją wspomnianej konfiguracji.

1.1. Przegląd literatury

Autor artykułu "Q-Learning for Adaptive Traffic Signal Control Based on Delay Minimization Strategy" skupił się na porównaniu statycznie ustawionego cyklu sygnalizacji świetlnej do zarządzanego przez algorytm Q-Learning [7]. Jest to algorytm uczenia maszynowego, który opiera się na zasadzie uczenia przez wzmacnianie, z tą różnicą, że nie posiada modelu dla środowiska (źródła danych) oraz nie wymaga nadzorowania procesu uczenia. Algorytm samodzielnie nabiera doświadczenia, potrafi dynamicznie dostosować się do zmian zachodzących w środowisku. Poprzez użycie algorytmu Q-Learning, podczas badań wykonywanych przez autorów artykułu, udało się zredukować oczekiwanie podczas przejazdów przez skrzyżowanie o średnio 21 sekund na cykl świateł dla samochodu, co przełożyło się na o 11 minut krótszy przejazd w stosunku do statycznie ustawionych cykli sygnalizacji świetlnej, gdzie przejazd zajmował godzinę (18,3% krócej).

W artykule "Urban Traffic Control Based on Learning Agents" autorzy postanowili wykorzystać algorytm uczenia ze wzmocnieniem w symulatorze ruchu pojazdów [9]. Dla porównania wykonano symulacje ze stałym ustawieniem sygnalizatorów świetlnych oraz z wykorzystaniem algorytmu Q-Learning, czego wynikiem były podobne rezultaty w przypadku stałego poziomu natężenia ruchu, a dla ruchu losowego - nieco ponad 10% krótsze czasy postoju.

Rozszerzoną wersję badań z poprzednich artykułów przedstawili Samah El-Tantawy i Baher Abdulhai w pracy dotyczącej koordynacji sygnalizacji świetlnej na wielu połączonych ze sobą skrzyżowaniach [2]. Wykorzystano do tego celu bezmodelowe, wieloagentowe uczenie ze wzmacnianiem (wieloagentowy Q-Learning). Efektem

przeprowadzonych badań było uzyskanie tzw. "zielonego czasu" (ang. green time), czyli czasu trwania zielonego światła, zgodnego z zapotrzebowaniem ze strony użytkowników ruchu. Analizując przedstawione przez autorów wykresy można zauważyć jak bardzo podobne, w przeciwieństwie do czasu ustawionego statycznie, jest przydzielenie "zielonego czasu" wyznaczonego za pomocą algorytmu uczącego do zapotrzebowania na zielone światło.

Rozwiązanie przedstawione w artykule "A group-based traffic signal control with adaptive learning ability" wykorzystuje wieloagentowe uczenie przez wzmacnianie. Porównywany jest do istniejącego genetycznego algorytmu kontroli i pokazuje wyższość koncepcji opisanej przez autorów dzięki swojej możliwości adaptacji [4].

Przykładem wykorzystania skoordynowanego algorytmu Q- Learning jest badanie opisane w artykule "Traffic Light Control in Non-stationary Environments based on Multi Agent Q-learning" [1]. Badano efektywność tego algorytmu w autorskim symulatorze, gdzie, jak w większości wymienionych powyżej prac badawczych, wybierano drogę otrzymującą zielone światło na podstawie długości pojazdów oczekujących na światłach oraz liczby nadjeżdżających samochodów. Wynikiem było stwierdzenie, że dla zwiększającego się natężenia pojazdów, użycie dynamicznej kontroli sygnalizacją jest coraz bardziej efektywne.

Artykuł "Multiagent Reinforcement Learning for Urban Traffic Control Using Coordination Graphs", autorstwa Kuyer L., Whiteson S., Bakker B. oraz Vlassis N., pokazuje rozszerzenie algorytmu wieloagentowego uczenia przez wzmacnianie poprzez dodanie koordynacji pomiędzy agentami i zwraca uwagę na poprawę działania w porównaniu do poprzedniej, nieskoordynowanej wersji [5].

Ciekawe podejście przedstawiono w pracy pod tytułem "Intelligent Traffic Light Control Using Distributed Multi-agent Q Learning" [6]. Autorzy postanowili stworzyć rozwiązanie oparte nie tylko na analizie ruchu pojazdów, ale włączyć do niej również ruch pieszych, co miałoby stanowić podstawę kierowanie ruchem w inteligentnym mieście. Przedstawione wyniki na obecnym poziomie prac deklasują istniejące rozwiązania, choć sami autorzy deklarują podjęcie pracy w ulepszaniu algorytmu.

Praca badawcza pod tytułem "Coordinated Deep Reinforcement Learners for Traffic Light Control" przedstawia użycie Q-Learning w głębokiej konwolucyjnej sieci neuronowej [10]. Jest to jedynie próba wykorzystania takiego podejścia, jednak jak wynika z badań, potrzeba znacznie więcej pracy aby zapewnić niezawodność głębokiego uczenia przez wzmacnianie, gdyż nie jest ono stabilne. Jednak udowodnione zostało, że algorytm ten przewyższa swojego poprzednika (wieloagentowe uczenie przez wzmacnianie), pomijając wiele problemów oraz pozwala na szybsze i bardziej skalowalne uczenie.

„Traffic Light Control Using Deep Policy-Gradient and Value-Function Based Reinforcement Learning" [8] to praca, która przedstawia to samo podejście do problemu sterowania ruchem, co poprzednio wspomniany artykuł. Skupiono się na polityce zachowań oraz wartościach uzyskanych z analizy migawek pobranych z symulatora ruchu w postaci obrazu, które analizują zaproponowanymi przez siebie algorytmami głębokiego uczenia przez wzmacnianie: polityka gradientów i oparte na funkcji wartości nagrody. W porównaniu do sztywno ustawionych sygnalizacji świetlnych użycie obu algorytmów skutkowało znacznie mniejszą średnią liczbą samochodów oczekujących na przejazd przez skrzyżowanie.

Autorzy artykułu "Adaptive Traffic Signal Control: Deep Reinforcement Learning Algorithm with Experience Replay and Target Network" przedstawili własne rozwiązanie problemu sterowania sygnalizacją świetlną. Zwrócili uwagę na fakt, że w większości przypadków stosuje się podejście, w którym pierwszeństwo ma najdłuższa kolejka samochodów, a pomijane są dane takie jak prędkość każdego pojazdu czy jego pozycja. Uzyskana płynność ruchu w porównaniu do algorytmu "najdłuższej kolejki" jest lepsza o 47% i aż 86% do statycznego ustawienia świateł [3].

1.2. Cel badań

Sposoby zarządzania sygnalizacją świetlną przedstawione w przeglądzie literatury opierają się na ciągłej analizie ruchu ulicznego. W artykule podjęto się analizy sytuacji, w której dany system dostosowujemy tylko jeden raz, na podstawie danych zebranych w trakcie pewnego okresu obserwacji. Maszyna wektorów nośnych, ze względu na swoje założenia oraz sposób działania, odnajduje swoje zastosowanie w rozwiązywaniu tego typu problemów. Niniejszy artykuł skupia się na doborze odpowiednich parametrów trenowanych modeli opartych o algorytm maszyny wektorów nośnych. Kolejnym obszarem, który poddano badaniom, była optymalizacja konfiguracji sygnalizacji świetlnej w stworzonym symulatorze na podstawie obserwacji jakości ruchu z pomocą utworzonych modeli maszyny wektorów nośnych. Jakość ruchu definiowana jest przez trzy cechy: średnia prędkość przejazdu, średni czas postoju i średnia liczba wypadków. Hipoteza badawcza, związana z procesem optymalizacji konfiguracji sygnalizacji świetlnej, postawiona w tej pracy, brzmi następująco: "Czy długość cyklu sygnalizacji świetlnej ma wpływ na jakość ruchu ulicznego?".

1.3. Zakres artykułu

W rozdziale drugim opisano obiekt badań oraz autorski symulator ruchu ulicznego.

Rozdział trzeci zawiera charakterystykę metody wykorzystanej w przeprowadzonych badaniach.

Rozdział czwarty prezentuje wyniki dostosowania modelu wykorzystanego w procesie optymalizacji parametru długości cyklu sygnalizacji świetlnej.

W rozdziale piątym omówiono uzyskane wyniki badań, natomiast w rozdziale szóstym przedstawiono wnioski powstałe w trakcie badań.

2. Opis obiektu badań

Parametr długości cyklu sygnalizacji świetlnej wydłuża czas trwania światła zielonego o zadaną wartość, co przekłada

się na przedłużenie całego cyklu. Minimalną wartość, jaką może przyjąć parametr to 5 sekund (mniejsze wartości uniemożliwiłyby przejazd przez skrzyżowanie).

Jakość ruchu rozumiana jest jako ogół warunków na drogach składających się na płynność przejazdu. Jest definiowana na podstawie współczynników:

- średniej prędkości przejazdu,
- średniego czasu postoju,
- średniej liczby kolizji.

Parametr długości cyklu sygnalizacji świetlnej jest kluczowym elementem autorskiego symulatora, który ma wpływ na jakość ruchu ulicznego.

2.1. Symulator ruchu ulicznego

Zadaniem systemu jest symulowanie ruchu pojazdów oraz pieszych na dwuwymiarowej makiecie fragmentu miasta. Uczestnicy ruchu poruszają się z punktu A do punktu B wraz z obowiązującymi zasadami ruchu drogowego, takimi jak: ustępowanie pierwszeństwa, zatrzymywanie się na czerwonym świetle czy przepuszczanie pieszych na przejściach. Symulator uwzględnia również poziom przyspieszania oraz maksymalnej prędkości z jaką poruszają się uczestnicy ruchu, tzn. przy dozwolonej prędkości 50km/h będzie osiągana prędkość oscylująca w jej granicach, odzwierciedlając sposób jazdy kierującego. Symulator, dla lepszego przedstawienia rzeczywistości, uwzględnia uczestników łamiących przepisy ruchu drogowego np. dopuszczających się znacznego przekroczenia prędkości lub nie zatrzymywania pojazdu w wyznaczonych miejscach. Ostatnim zadaniem symulatora jest uwzględnienie wypadków drogowych. W chwili ich wystąpienia ma miejsce czasowa blokada danej części drogi, na której zdarzył się wypadek.

3. Metoda badawcza

Plan badań zakłada przeprowadzenie symulacji w trzech różnych wariantach:

- poziom zagrożenia 0% - tryb bezpieczny, brak uczestników ruchu łamiących przepisy ruchu drogowego,
- poziom zagrożenia 50% - połowa uczestników ruchu łamie przepisy ruchu drogowego,
- poziom zagrożenia 100% - wszyscy uczestnicy łamią przepisy ruchu drogowego.

Symulacje zostaną wykonane dla:

- parametru długości cyklu sygnalizacji świetlnej o wartościach 5-15, oraz 20, 25 i 30 sekund,
- liczby samochodów wynoszącej 100 dla danych trenujących i 50 samochodów dla danych testowych,
- liczby pieszych: 50.

Do zebrania danych trenujących, w każdej konfiguracji, zostaną przeprowadzone dwie symulacje, a dla danych testowych jedna. Przełoży się to na maksymalnie 680 próbek dla zbioru trenującego i 476 próbek dla zbioru testowego. Samochody mogą poruszać się po 34 drogach. Każda z nich posiada dwa pasy ruchu. W analizie nie będą brane pod uwagę drogi na których samochód rozpoczyna, bądź kończy trasę. W rezultacie 21 dróg jest poddanych analizie. Dla dróg dwukierunkowych dane z pasów ruchu zbierane są osobno (różny kierunek ruchu), a dla jednokierunkowych razem, ponieważ możliwa jest zmiana pasa ruchu, a dane zbierane są dla całej długości drogi - powodowałoby to odrzucenie dużej ilości danych. Uwzględnienie kierunku ruchu daje 34 analizowane elementy. Ze względu na losowość wybranej trasy może wydarzyć się sytuacja, w której po niektórych drogach i kierunku, w danej konfiguracji, nie przejedzie żaden samochód, czego wynikiem będzie nieznaczne zmniejszenie liczby próbek. Zebrane dane zostaną zsumowane w obrębie każdego elementu, dla poszczególnych wartości parametru długości cyklu sygnalizacji świetlnej, i zapisane do pliku wraz z liczbą wyników. Następnie zostaną obliczone średnie wartości współczynników (średnia prędkość, średni czas przejazdu i średnia liczba wypadków) w zakresie poszczególnych czasów, dla każdej drogi i kierunku ruchu.

Dla każdego ze zbiorów wartości badanych parametrów zostaną wyznaczone po cztery wartości:

- minimalna - kres dolny zbioru,
- kwartyl pierwszy,
- kwartyl trzeci,
- maksymalna - kres górny zbioru.

Tak przetworzone dane umożliwią podział każdego z badanych zbiorów na trzy klasy:

- dla zbioru średniej liczby kolizji i zbioru średniego czasu postoju na drodze:
  - "najlepsza", gdy zawiera się w przedziale od wartości minimalnej do kwartyla pierwszego,
  - "średnia", gdy zawiera się w przedziale od kwartyla pierwszego do kwartyla trzeciego,
  - "najgorsza", gdy zawiera się w przedziale od kwartyla trzeciego do wartości maksymalnej.
- dla zbioru średniej prędkości:
  - "najlepsza", gdy zawiera się w przedziale od kwartyla trzeciego do wartości maksymalnej,
  - "średnia", gdy zawiera się w przedziale od kwartyla pierwszego do kwartyla trzeciego,
  - "najgorsza", gdy zawiera się w przedziale od wartości minimalnej do kwartyla pierwszego.

Podane wartości obliczane są oddzielnie dla każdego identyfikatora drogi. Dane te zostaną wykorzystane do trenowania modeli - każdy model odpowiadać będzie za klasyfikowanie poszczególnego parametru i identyfikatora drogi. Modele zostaną stworzone w oparciu o model maszyny wektorów nośnych (SVM). Został on odpowiednio dopasowany dla analizowanego zbioru danych - złożonego, ale o niedużym rozmiarze (mniej niż 700 próbek).

Trenowanie modeli wykonane zostanie przy pomocy klasyfikatora wektorów nośnych (SVC) z parametrami:

- kernel - jądro klasyfikatora, np. liniowe,
- C - oznacza stopień tolerancji na błędną klasyfikację próbki; im większa wartość, tym mniejsza tolerancja,
- gamma - definiuje wartość wpływu poszczególnej próbki,

- degree - stopień wielomianu jądra wielomianowego.

Trenowanie modeli zostanie przeprowadzone z różnymi wartościami podanych parametrów. Każdy z modeli zostanie przetestowany celem wyłonienia zestawu parametrów klasyfikatora dającego najlepsze wyniki.

Wpływ długości świateł zostanie określony w następujący sposób:

- dla każdego parametru długości cyklu sygnalizacji świetlnej obliczona będzie suma wyników klasyfikacji wszystkich trzech parametrów wszystkich dróg (dla dróg bez wyników, istnieje możliwość, że w danej symulacji po danej drodze nie przejedzie żaden pojazd, przyjęty zostanie klasyfikator "średnia"),
- powstałe sumy wyników klasyfikacji zostaną poddane analizie, w wyniku której będzie można jednoznacznie określić, czy parametr długości świateł ma realny wpływ na jakość ruchu ulicznego.

4. Prezentacja rezultatów badań

Ze względu na dwa etapy badań przedstawienie rezultatów podzielono na dwie części: dostosowanie modelu oraz dostosowanie wartości parametru sterującego sygnalizacją świetlną.

4.1. Dostosowanie modelu

Parametry zostały dopasowane w następującej kolejności:

1) degree

Na początku wybrano najlepszą wartość *degree* dla jądra wielomianowego. Miało to na celu wyłonienie modelu wielomianowego o najwyższej precyzji, aby móc porównywać jego najlepszy wariant z innymi jądrami. Model odpowiedzialny za średnią prędkość uzyskiwał najdokładniejsze wyniki dla wartości parametru *degree* równej 7. Średni czas oczekiwania na drodze był wyznaczany najprecyzyjniej przez model z wartością parametru *degree* równą 9. Modele dla średniej liczby kolizji otrzymywały wyniki z podobną dokładnością. Piąty stopień wielomianu uzyskał najlepszy wynik i zostanie wykorzystany dla jądra wielomianowego w dalszej części badań.

2) gamma

Parametr *gamma* jest używany przez modele z jądrem wielomianowym oraz radialnym, dlatego jądro liniowe zostało pominięte w tym kroku.

Dla jądra wielomianowego model średniej prędkości oraz średniego czasu oczekiwania na drodze był najdokładniejszy dla *gamma* równego 1. Model średniej liczby kolizji uzyskiwał wyniki na tym samym poziomie bez względu na wartość parametru *gamma*. Do dalszych badań, w celu ujednolicenia tego parametru dla wszystkich modeli, ustawiono jego wartość na 1. Zabieg ten nie wpłynął na dokładność wyników.

Dla jądra radialnego model średniej prędkości uzyskał najwyższy wynik, gdy wartość *gamma* wyniosła 0,75 i 1 - dla kolejnych etapów selekcji parametrów przyjęto wartość 1. Model średniego czasu oczekiwania na drodze okazał się najdokładniejszy dla wartości *gamma* równej 1.5. *Gamma* nie miała wpływu na wyniki klasyfikacji modelu średniej liczby kolizji. Dla każdej wartości otrzymano maksymalny wynik - do dalszych badań przyjęto wartość 1.

3) kernel

Jądro radialne okazało się bezkonkurencyjne. Uzyskane wyniki modeli z tym jądrem były najwyższe.

4) C

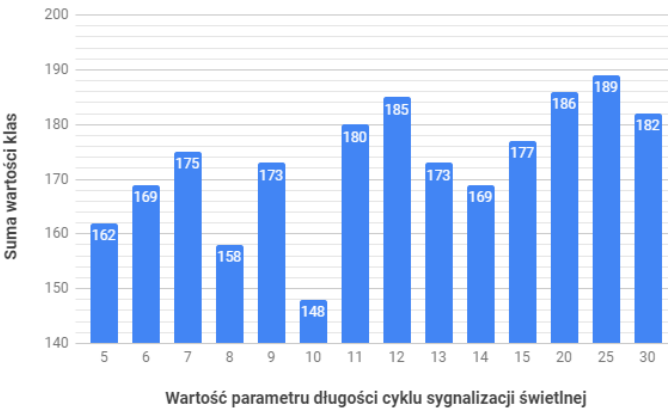
Model średniej prędkości klasyfikował dane z największą dokładnością dla parametru *C* równego 1 000. Model średniego czasu oczekiwania na drodze wykazywał najlepsze wyniki, wyróżniające się tle pozostałych, gdy parametr był równy 10 000 000. Dla modelu średniej liczby kolizji zwiększenie parametru C spowodowało zmniejszenie dokładności. Wartości 1 000 i 10 000 pozwalały na uzyskanie maksymalnego wyniku. Bazując na tendencji spadku precyzji wraz ze wzrostem parametru C, wybrano wartość 1 000.

4.2. Dostosowanie wartości parametru długości cyklu sygnalizacji świetlnej

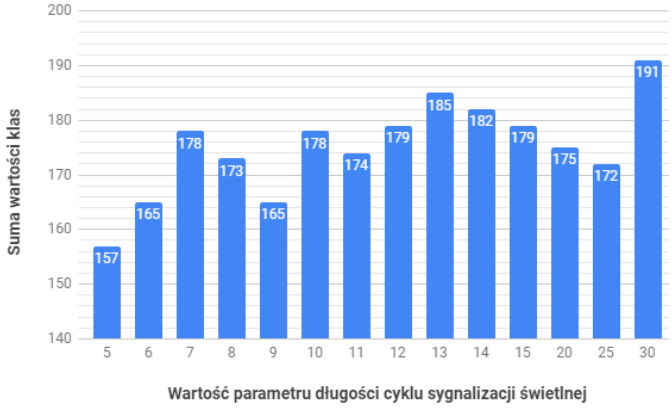
Dla każdego parametru długości cyklu sygnalizacji świetlnej obliczona została suma wyników klasyfikacji wszystkich trzech parametrów wszystkich dróg, oddzielnie dla badanych poziomów zagrożenia. Suma wartości klas została wyznaczona w następujący sposób:

1) Próbki zostały pobrane ze zbiorów danych reprezentujących kolejny poziom.
2) Jeżeli dana próbka danych została sklasyfikowana przez model jako:
- najlepsza - otrzymuje wartość 1,
- średnia - otrzymuje wartość 2,
- najgorsza - otrzymuje wartość 3.
3) Wartości próbek zostały zsumowane w obrębie każdej wartości parametru długości cyklu sygnalizacji świetlnej.

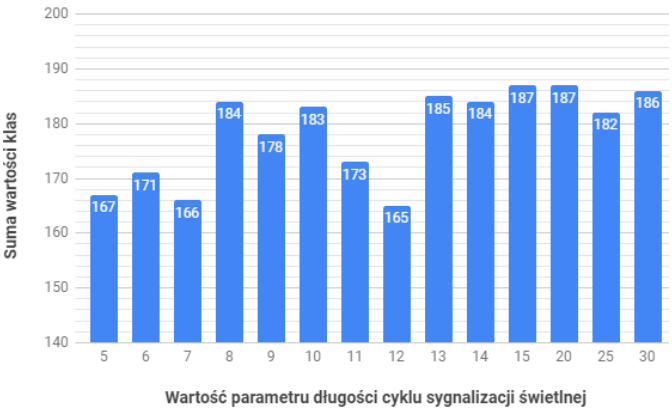
Wyniki tej operacji przedstawiono na rysunkach 2, 3, 4. Jakość ruchu ulicznego jest odwrotnie proporcjonalna do sumy wartości klas.



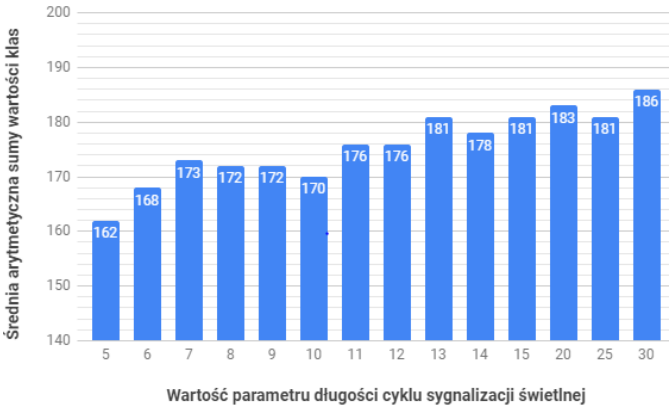
Rys. 2. Wpływ parametru długości cyklu sygnalizacji świetlnej na jakość ruchu ulicznego dla poziomu zagrożenia wynoszącego 0%.



Rys. 3. Wpływ parametru długości cyklu sygnalizacji świetlnej na jakość ruchu ulicznego dla poziomu zagrożenie wynoszącego 50%.



Rys. 4. Wpływ parametru długości cyklu sygnalizacji świetlnej na jakość ruchu ulicznego dla poziomu zagrożenie wynoszącego 100%.



Rys. 5. Średnia arytmetyczna z sumy wartości klas dla wszystkich poziomów zagrożenia.

5. Dyskusja wyników

Analizując wykresy przedstawione na rysunkach 3-5, można stwierdzić, że nie można dopasować jednoznacznego wzorca do zależności pomiędzy parametrem długości cyklu sygnalizacji świetlnej, a jakością ruchu. Mimo tendencji wzrostu sumy wartości klas wraz ze zwiększeniem parametru długości cyklu sygnalizacji świetlnej we wszystkich badanych poziomach zagrożenia, najniższa wartość tego parametru nie zawsze wiąże się z najmniejszą wartością sumy wartości klas. Powodem tego jest duża losowość w przeprowadzanych symulacjach. Jest dużo czynników, które mają wpływ na przebieg symulacji, na przykład miejsce wypadku, prędkość pojazdu czy natężenie ruchu na danym odcinku. Z tego powodu sumy wartości klas wszystkich poziomów zagrożenia zostały uśrednione i przedstawione na rysunku 5.

Uśrednienie wartości pozwoliło na ograniczenie skutków losowości przy zbieraniu danych. Odczytując powyższy wykres można z całą pewnością potwierdzić zależność pomiędzy parametrem długości cyklu sygnalizacji świetlnej a jakością ruchu ulicznego tj. wraz ze wzrostem parametru długości cyklu sygnalizacji świetlnej spada jakość ruchu ulicznego. Można stwierdzić, że najbardziej optymalną wartością dla parametru długości cyklu sygnalizacji świetlnej była wartość 5, czyli wartość najmniejsza.

6. Wnioski

W wyniku przeprowadzonych badań otrzymano optymalny zestaw parametrów dla modeli wytrenowanych w sprawdzania wpływu wartości parametru długości cyklu na jakość ruchu. Analiza przeprowadzona na podstawie wyników zebranych i zaprezentowanych w postaci wykresów doprowadziła do ustalenia następujących wniosków:

1) mimo niewielkich różnic pomiędzy wartościami w otrzymanych rezultatach dla każdego poziomu zagrożenia konieczne było przeanalizowanie wyników zbiorczo w postaci średniej arytmetycznej każdej wartości parametru wyciągniętej ze wszystkich badanych poziomów zagrożenia w celu określenia prawidłowości zachodzących w przedmiocie badań,
2) parametr długości cyklu sygnalizacji świetlnej ma niewielki wpływ na jakość ruchu ulicznego,
3) najniższa wartość parametru długości cyklu sygnalizacji świetlnej okazała się najlepsza,
4) wydłużenie cyklu sygnalizacji świetlnej, z małymi wyjątkami, wiązało się z obniżeniem jakości ruchu,
5) charakter i zróżnicowanie próbek danych ma bardzo duży wpływ na parametryzację modeli implementujących algorytmy uczenia maszynowego oraz przebieg ich dostrajania: im dane są mniej skomplikowane, tym proces trenowania i dostosowywania modelu jest krótszy i mniej złożony,
6) nieprawidłowe dobranie wartości parametrów skutkuje uzyskaniem niepożądanych wyników oraz wydłużeniem procesu trenowania do niebotycznych długości.

Podsumowując wykonane badania można stwierdzić, że hipoteza: “Czy długość cyklu sygnalizacji świetlnej ma wpływ na jakość ruchu ulicznego?” została potwierdzona. Analizując wyniki badań można zauważyć tendencję spadku jakości ruchu wraz ze wzrostem parametru długości cyklu sygnalizacji świetlnej - im cykl jest dłuższy, tym bardziej negatywnie wpływa na ruch uliczny.

Warto zaznaczyć, że duży wpływ na wyniki badań mógł mieć losowy charakter zebranych próbek danych tworzących zbiory: trenujący oraz wykorzystany w analizie problemu.

Literatura

[1] Abdoos M., Mozayani N., Bazzan A. L. C.,Traffic Light Control in Non-stationary Environments based on Multi Agent Q-learning, 14th International IEEE Conference on Intelligent Transportation Systems (ITSC), s. 1580 - 1585, 2011

[2] El-Tantawy S., Abdulhai B., An Agent-Based Learning Towards Decentralized and Coordinated Traffic Signal Control, 13th International IEEE Conference on Intelligent Transportation Systems, s. 665 - 670, 2010

[3] Gao J., Shen Y., Liu J., Ito M., Shiratori S., Adaptive Traffic Signal Control: Deep Reinforcement Learning Algorithm with Experience Replay and Target Network, arXiv:1705.02755, 2017

[4] Jin J., Ma X., A group-based traffic signal control with adaptive learning ability, Engineering applications of artificial intelligence, s. 282-293, 2017

[5] Kuyer L., Whiteson S., Bakker B., Vlassis N., Multiagent Reinforcement Learning for Urban Traffic Control Using Coordination Graphs, Obrady ECML/PKDD, Antwerp, Belgia, s.656–671,2008

[6] Liu Y., Liu L., Chen W., Intelligent Traffic Light Control Using Distributed Multi-agent Q Learning, IEEE 20th International Conference on Intelligent Transportation Systems (ITSC), Październik 2017

[7] Lu S., Liu X., Dai S., Q-Learning for Adaptive Traffic Signal Control Based on Delay Minimization Strategy, 2008 IEEE International Conference on Networking, Sensing and Control, s. 687-691, 2008

[8] Mousav S. S., Schukat M., Howley E., Traffic Light Control Using Deep Policy-Gradient and Value-Function Based Reinforcement Learning, IET Intelligent Transport System, vol.11 No.7, s. 417-423, Wrzesień 2017

[9] Pierre-Luc G., Desjardins C., Laumonier J., Chaib-draa B., Urban Traffic Control Based on Learning Agents, 2007 IEEE Intelligent Transportation Systems Conference, s. 916-921, 2007

[10] van der Pol E. Oliehoek F. A., Coordinated Deep Reinforcement Learners for Traffic Light Control, Artykuły naukowe Uniwersytetu Amsterdamskiego, 2016

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 14 (2020) 43-47
Wysłane: 2019-10-16
Przyjęte: 2019-10-22

Porównanie systemów mapowania obiektowo relacyjnego greenDao i Room

Maciej Lewinski*
Politechnika Lubelska, Katedra Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. Poniższy artykuł skupia się na porównaniu dwóch rozwiązań ORM dla systemu Android: greenDao i Room. Zestawiono sposób użycia tych frameworków. Dokonano pomiaru czasu wykonania operacji wstawienia, aktualizacji, pobierania i usuwania danych z bazy. W graficznej i tabelarycznej formie przedstawiono wyniki. Dokonano ich analizy i zaprezentowano wnioski.

Słowa kluczowe: Android; ORM; greenDao; Room

*Autor do korespondencji.

Adres e-mail: lewinski.maciej1@gmail.com

Comparison of GreenDao and Room ORM Systems

Maciej Lewinski*
Department of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract. This paper focuses on comparison of two ORM libraries for Android: greenDao and Room. Ways of using these frameworks are presented. Time consumption of creating, reading, updating and deleting records are measured. Tables and charts containing results are presented. Finally analysis and conclusions are made.

Keywords: Android; ORM; greenDao; Room

*Corresponding author.

E-mail address: lewinski.maciej1@gmail.com

1. Wstęp

W dzisiejszych czasach urządzenia z systemem Android stały się elementem codziennego życia. Zgodnie ze statystykami prowadzonymi przez *statcounter GlobalStats* 76.23% wszystkich urządzeń mobilnych korzysta z systemu Android (stan na sierpień 2019) [1]. Dla porównania drugi najbardziej popularny system iOS posiadał udział w rynku na poziomie 22.17%.

Właściwie każda aplikacja zasilana jest danymi z bazy danych. Otrzymywane dane muszą być w odpowiedni sposób przystosowane w celu użycia w aplikacji. Proces ten nazywa się mapowaniem obiektowo-relacyjnym (ang. Object-Relational Mapping).

Tworzenie kodu programu, który jest odpowiedzialny za połączenie z bazą danych i wykonywanie na niej operacji jest schematyczne. Pojawiło się wiele frameworków, które wyręczają programistę z pisania powtarzalnych elementów. Użycie ich zmniejsza nakład pracy i ilość kodu minimalizując ryzyko błędów.

Systemy ORM różnią się między sobą. Można je porównywać pod względem ilości linii kodu, który trzeba napisać, żeby je użyć czy wspieranych systemów baz danych. W tym artykule porównywane są dwa z nich greenDao i Room pod względem wydajności i podstawowego sposobu użycia.

2. Przegląd literatury

Użyteczność systemów ORM motywuje autorów do tworzenia prac na ten temat. Jednym z najbardziej popularnych frameworków stworzonych dla języka Java jest Hibernate. W artykule „Efficient Implement of ORM (Object/Relational Mapping) Use in J2EE Framework: Hibernate" [2] poruszany jest problem efektywnego używania tego systemu. Autor przedstawia różne elementy rozwiązania Hibernate i ich sposób użycia. System Android korzysta jednak z bazy SQLite, co wyklucza korzystanie tego systemu. Urządzenia mobilne niosą również dodatkowe wyzwania. W artykule „Understanding the Energy, Performance, and Programming Effort Trade-Offs of Android Persistence Frameworks" porównane zostały rozwiązania ORM skupiając się na wykorzystaniu baterii, obciążeniu bazy danych i czas wykonywania zapytań [3].

Producenci frameworków również tworzą testy sprawności w celu promowania swoich rozwiązań. Firma greenRobot opublikowała porównanie systemów ORM [4]. Zestawiono w nim czasy wykonań operacji CRUD (create, read, update, delete). Zgodnie z tym zestawieniem najszybszy jest framework greenDao. Nie zawarto w nim jednak systemu Room.

Tworząc testy sprawności uruchamiane na telefonach lub tabletach należy mieć na uwadze specyfikę tych urządzeń. W artykule „Benchmarking on Android" [5] poruszone zostały kwestie takie jak oszczędzanie energii, aktualizacja innych

aplikacji i tryb samolotowy oraz ich wpływ na przeprowadzane testy. Należy dbać, aby testy były jak najbardziej obiektywne.

3. Opis obiektu badań

W tej pracy zbadano działanie dwóch frameworków greenDao i Room. Oba dostarczają rozwiązania implementujące mapowanie obiektowo-relacyjne przeznaczone dla systemu Android.

Główną klasą używaną podczas tworzenia aplikacji dla tego systemu jest Activity [6]. Zawiera ona metody, które są wywoływane podczas zmiany stanów aktywności [7]. Są one odpowiedzią na działania użytkownika takie jak naciśnięcie przycisku wstecz czy zamknięcie aplikacji [8]. Aktywności są odpowiedzialne przede wszystkim za zarządzanie widokiem. Za połączenie z bazą danych czy mapowanie obiektowo-relacyjne powinny być odpowiedzialne inne klasy.

GreenDao jest systemem, który używa adnotacji w celu odpowiedniego oznaczenia klas i ich składowych, które powinny zostać użyte [9]. Następujące klasy są podstawą użycia greenDao: DaoMaster, DaoSession, XYZDao, XYZEntity.

DaoMaster jest klasą, która jest punktem dostępowym do systemu greenDao. Ta główna klasa zarządza innymi, które są odpowiedzialne za dostęp do danych danego schematu. Jedną ze składowych klasy DaoMaster jest obiekt bazy danych. Zawiera ona również statyczne metody do tworzenia i usuwania tabel. W postaci klas wewnętrznych dostarcza również implementacje klasy abstrakcyjnej SQLiteOpenHelper: OpenHelper i DevOpenHelper.

DaoSession odpowiedzialne jest za zarządzanie wszystkimi obiektami klas dostępu do bazy, czyli klas DAO (ang. Data Access Object). Za pomocą obiektów tej klasy możemy wywołać również podstawowe metody dostępu do danych.

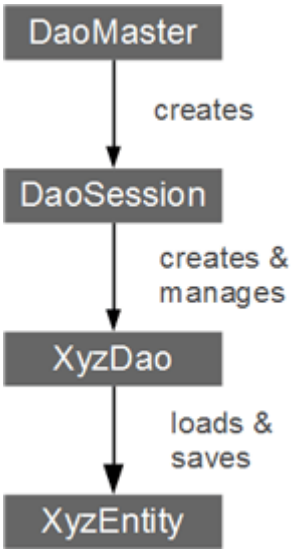
GreenDao dla każdej klasy encji tworzy klasę dostępu do danych. Odpowiedzialne są one za zapisywanie danych i wykonywanie bardziej skomplikowanych zapytań do bazy.

Klasy encji reprezentują tabele w bazie danych. Zazwyczaj jeden obiekt odpowiada jednemu rekordowi. Te klasy oznaczane są adnotacją *@Entity*.

Rysunek 1 ilustruje zależności pomiędzy komponentami systemu greenDao.

System Room tworzy poziom abstrakcji używany do łatwego dostępu do bazy SQLite [10]. Pozwala to na odseparowanie logiki biznesowej aplikacji od części odpowiedzialnej za pobieranie i zapisywanie danych. Jest to oficjalnie zalecany framework dla systemu Android do pracowania z lokalną bazą danych.

System ORM Room składa się z trzech podstawowych elementów, a mianowicie *Database*, *Entity* i *Dao*.



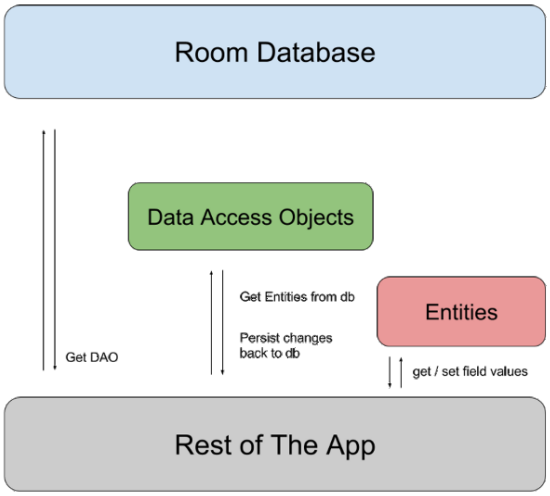
Rys. 1. Zależności między klasami systemu greenDao [9]

Klasa oznaczona adnotacją *@Database* i rozszerzająca *RoomDatabase* jest punktem wyjściowym do użycia systemu Room. Dostarcza ona połączenie do bazy danych. Obiekt tej klasy można uzyskać wywołując statyczną metodę *databaseBuilder()* klasy *Room*.

Klasy rodzaju *Entity* są definicjami tabel w bazie danych. Oznaczane są adnotacją *@Entity*. Ich instancje reprezentują wiersze danych.

W celu użycia obiektów DAO programista musi zdefiniować interfejs oznaczony adnotacją *@Dao*. System Room sam wygeneruje klasę go implementującą.

Rysunek 2 ilustruje zależności między komponentami systemu Room.



Rys. 2. Zależności między komponentami frameworka Room [10]

4. Metody badawcze

W celu zbadania szybkości działania wybranych systemów ORM zbudowano aplikację wykonującą te same operacje w bazie danych za pomocą różnych frameworków.

W tabeli 1 przedstawiono użyte wersje badanych systemów ORM.

Tabela 1. Wersje użytych systemów ORM

Nazwa systemu	Wersja
Room	2.1.0
greenDao	3.2.2

4.1. Wstawianie rekordów

Pierwszy test polegał na pomiarze czasu wstawiania rekordów do bazy. Klasy reprezentujące encje zawierały pola wszystkich typów prostych i odpowiadających im typów referencyjnych. Przed rozpoczęciem wstawiania tworzono listę wypełnioną odpowiednią ilością gotowych do wstawienia rekordów. Pola prymitywnych typów liczbowych były inicjalizowane minimalnymi wartościami, jakie mogły zawierać, pola referencyjne zainicjalizowano wartością *null*, a pole typu *boolean* zawierało wartość *false*. Przed startem pomiaru inicjalizowano również połączenie z bazą danych. Wypełniana tabela była pusta.

4.2. Aktualizowanie danych w bazie

Kolejną operacją, która była mierzona było aktualizowanie danych w bazie. Użyto tabeli, która została zastosowana do pomiaru czasu wstawiania. Użyto takiej liczby rekordów, jaka została dodana w poprzedzającym badaniu. Przed rozpoczęciem pomiaru wypełniono danymi listę obiektów klas reprezentujących encje. Pola prymitywnych typów liczbowych były inicjalizowane maksymalnymi wartościami, jakie mogły zawierać, pola referencyjne zainicjalizowano wartością *null*, a pole typu *boolean* zawierało wartość *true*.

4.3. Pobieranie rekordów z bazy

Ten test polegał na pomiarze czasu pobierania uprzednio zaktualizowanych danych. Indeksy wierszy były kolejnymi liczbami naturalnymi zaczynając od liczby jeden. Wykonywano pojedyncze zapytania do bazy danych. Podając wartość *id* rekordu wczytywano go do obiektu. Następnie wywoływano wszystkie gettery, aby zapobiec ewentualnemu odroczonemu wczytywaniu.

4.4. Usuwanie danych z bazy

W tym teście mierzono czas pojedynczego usuwania rekordów z bazy. Tabela, z której korzystano zawierała dane, które były pobierane w poprzednim badaniu. Dla każdej wartości *id* z osobna wywoływano metodę usuwania.

5. Wyniki badań

Wszystkie pomiary wykonano dla 250, 500, 1 000, 2 000, 5 000, 10 000, 20 000 rekordów. Testy w każdej konfiguracji powtórzono pięciokrotnie. Użytym urządzeniem był telefon Lenovo C2, posiadający 1 GB RAM. Zainstalowanym oprogramowaniem był system Android 6.0.

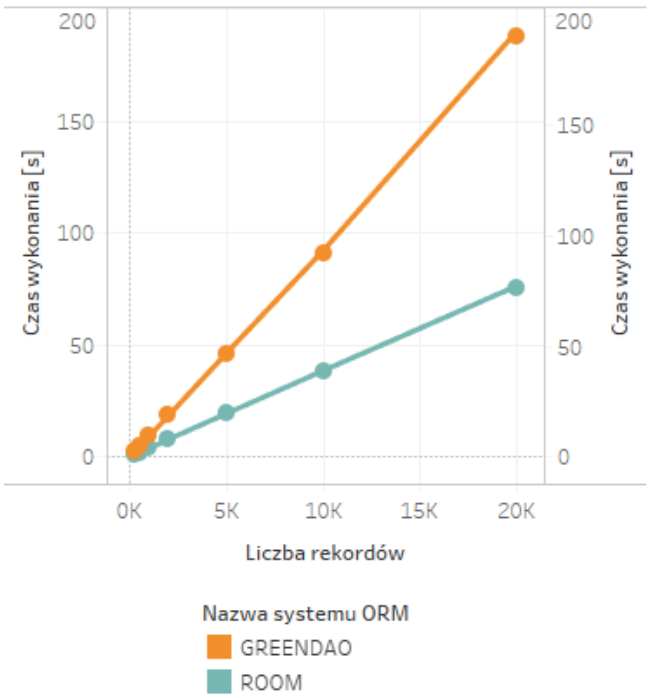
5.1. Wstawianie rekordów

Przedstawiony na rysunku 3 wykres bardzo wyraźnie wskazuje na liniową zależność między czasem wykonywania a liczbą rekordów. Jasno pokazuje on, że system greenDAO działał znacznie wolniej niż framework Room.

W tabeli 2 ujęto dokładne czasy wykonywania. Bez względu na liczbę rekordów system Room wykonywał tę operację przynajmniej dwa razy szybciej niż greenDAO.

Ostatecznie oficjalny framework dla systemu Android potrzebował średnio 0,3879 ms na wstawienie jednego rekordu, podczas, gdy ten publikowany przez greenRobot wymagał 0,9422 ms. Tak więc ten drugi potrzebował o 143 % więcej czasu.

Wstawienie 250 rekordów zajęło systemowi Room średnio 1,1 s. Jest to czas dłuższy niż 1 s, która jest graniczna wartością zapewniajaca użytkownikowi poczucie płynnego działania aplikacji. Oznacza to, że oczekiwanie na wstawienie takiej ilości rekordów może powodować dyskomfort.



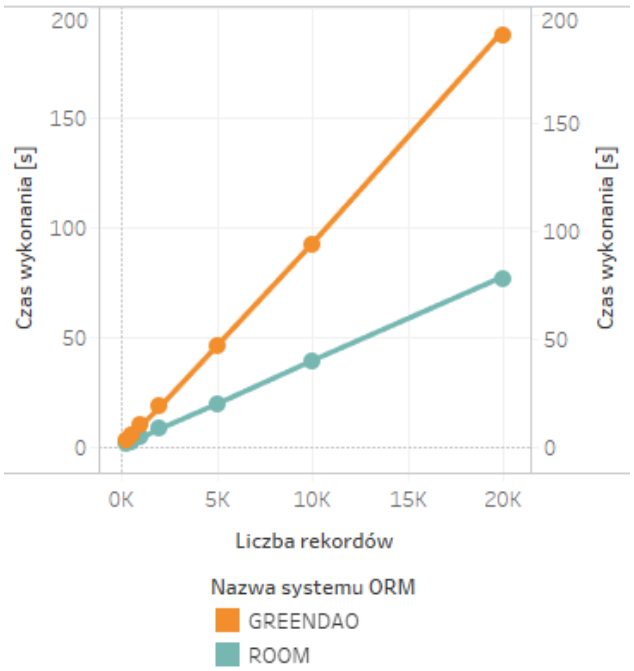
Rys. 3. Wykres zależności czasu wykonania operacji wstawiania od liczby rekordów

Tabela 2. Średni czas wykonania operacji wstawienie w zależności od liczby rekordów i systemu ORM

Liczba rekordów	Room	greenDao
250	1,1 s	2,4 s
500	2,2 s	4.9 s
1 000	4,0 s	9.6 s
2 000	8,1 s	18,8 s
5 000	19,6 s	46,7 s
10 000	38,8 s	92,5 s
20 000	76,5 s	190,3 s

5.2. Aktualizowanie rekordów

Czas aktualizacji jednego rekordu za pomocą systemu Room wynosił średnio 3,931 ms. Jest to wartość o 58 % niższa od wyniku frameworka greenDao, który potrzebował ok 9,415 ms. Tabela 3 pokazuje, że czasochłonność operacji operacji aktualizacji jest prawie taka jak operacji wstawiania. Posiada również wręcz identyczna liniową charakterystykę zależności od ilości rekordów, co widoczne jest na umieszczonym na rysunku 4 wykresie.



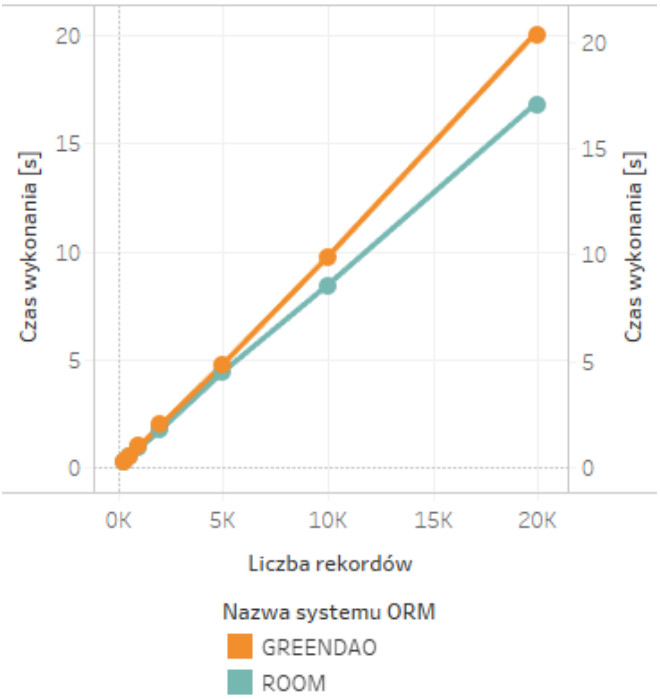
Rys. 4. Wykres zależność czasu wykonania operacji aktualizacji od liczby rekordów

Tabela 3. Średni czas wykonania operacji aktualizacji w zależności od liczby rekordów i systemu ORM

Liczba rekordów	Room	greenDao
250	1,1 s	2,6 s
500	2,0 s	4,8 s
1 000	4,1 s	9,5 s
2 000	8,3 s	18,3 s
5 000	19,7 s	46,3 s
10 000	39,4 s	93,0 s
20 000	77,8 s	190,2 s

5.3. Pobieranie rekordów

Wyniki pomiarów pokazują, że pobieranie rekordów z bazy danych jest operacją znacznie szybszą niż ich wstawianie czy aktualizacja. W tej kategorii system Room okazał się być również szybszy. Średnio pobranie jednego rekordu zajmowało temu frameworkowi 0,8601 ms. System greenDAO potrzebował 16 % więcej czasu, a mianowicie 0,9988 ms. Tak mała różnica w szczególności dla bardzo krótkich okresów czasu jest dla użytkownika praktycznie niezauważalna. Dokładne wartości zostały zaprezentowane w tabeli 4. Wykres przedstawiony na rysunku 5 pokazuje liniową zależność między czasem wykonania operacji pobierania a ilością rekordów.



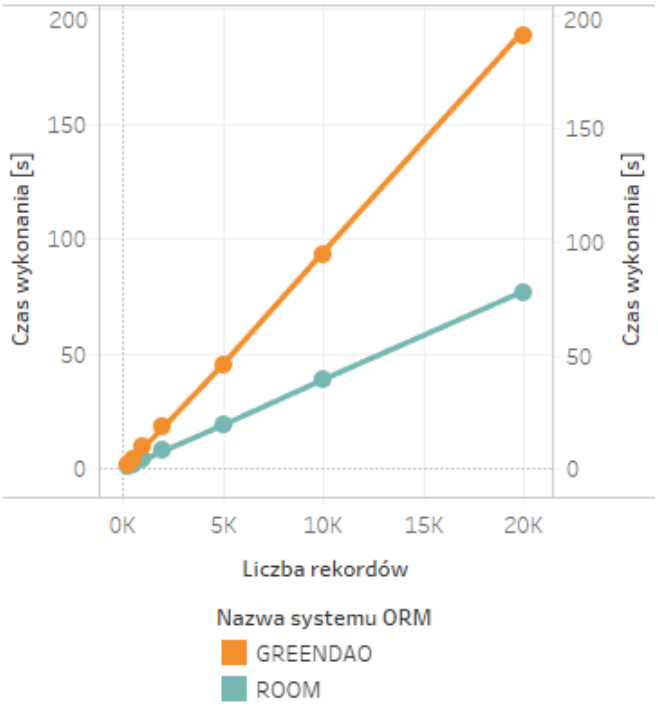
Rys. 5. Wykres zależności czasu wykonania operacji pobierania od liczby rekordów

Tabela 4. Średni czas wykonania operacji pobierania w zależności od liczby rekordów i systemu ORM

Liczba rekordów	Room	greenDao
250	0,22 s	0,27 s
500	0,49 s	0,52 s
1 000	0,89 s	0,99 s
2 000	1,75 s	1,97 s
5 000	4,46 s	4,82 s
10 000	8,50 s	9,82 s
20 000	17,01 s	20,31 s

5.4. Usuwanie rekordów

Umieszczony na rysunku 6 wykres pokazuje liniową zależność między czasem usuwania a liczbą rekordów. Wiedząc, że taka jest relacja można szacować czas wykonywania dla innej ilości danych. Pod względem usuwania danych szybszy ponownie okazał się system Room. Średni czas usuwania jednego rekordu przez ten framework wyniósł 3,913 ms. System greenDao potrzebował średnio na jeden rekord 9,468 ms, czyli o 142 % więcej. Tabela 5 pokazuje, że różnice w wykonywaniu tej operacji są znaczące.



Rys. 6. Wykres zależności czasu wykonania operacji usuwania od liczby rekordów

Tabela 5. Średni czas wykonania operacji usuwania w zależności od liczby rekordów i systemu ORM

Liczba rekordów	Room	greenDao
250	1,1 s	2,4 s
500	2,0 s	4,7 s
1 000	4,2 s	9,6 s
2 000	8,0 s	18,4 s
5 000	19,4 s	46,0 s
10 000	39,2 s	94,8 s
20 000	77,6 s	191,1 s

6. Analiza wyników i wnioski

Użycie obu frameworków jest bardzo podobne. Zawarcie ich w projekcie polega na dodaniu odpowiednich zależności w pliku Gradle. Zarówno system greenDao jak i Room używa adnotacji w celu oznaczenia mapowanych klas.

Znacząca różnica występuje jednak w sposobie użycia klas DAO. System Room umożliwia pisanie zapytań SQL, które są walidowane w czasie kompilowania projektu. greenDao pozwala na pisanie tzn. "surowych" zapytań lecz niestety nie są one sprawdzane podczas kompilacji. Typowym podejściem wspieranym przez system dostarczony przez greenRobot jest używanie budowniczego zapytań. Taka możliwość nie jest dostarczana przez system Room.

Framework dostarczany przez twórców systemu Android jest więc bardziej odpowiedni dla osób, które głównie korzystają z pisania zapytań w języku SQL. Walidacja podczas kompilacji znacząco ułatwia znajdowanie i naprawę błędów. Programiści, którzy nie znają SQL znaczanie szybciej odnajda się w używaniu systemu greenDAO, gdyż używanie klasy *QueryBuilder* będzie dla nich wygodniejsze.

Jeżeli dla tworzonej aplikacji osiągi czasowe są decydujące, to użycie systemu Room będzie zdecydowanie lepsze. Oferuje on znacznie szybsze wykonywanie operacji na bazie danych. Czasy pobierania były zbliżone, lecz wstawianie, aktualizacja i usuwanie danych było co najmniej dwa razy szybsze.

Literatura

[1] https://gs.statcounter.com/os-market-share/mobile/worldwide [30.09.2019]

[2] Chuanlong Xia, Guangcan Yu, Meng Tang: Efficient Implement of ORM (Object/Relational Mapping) Use in J2EE Framework: Hibernate. 2009 IEEE International Conference on Computational Intelligence and Software Engineering

[3] J. Pu, Z. Song, E. Tilevich, Understanding the Energy, Performance, and Programming Effort Trade-Offs of Android Persistence Frameworks, 2016 IEEE 24th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems

[4] http://greenrobot.org/android/android-orm-performance-2016 [14.10.2019]

[5] http://greenrobot.org/android/benchmarking-on-android/ [11.10.2019]

[6] Allen Grant, Beginning Android, Apress, 2015

[7] MacLean Dave, Komatineni Satya, Allen Grant, Pro Android 5, Apress, 2015

[8] https://developer.android.com/guide/components/activities/activity-lifecycle [05.10.2019]

[9] http://greenrobot.org/greendao/documentation/introduction/ [04.10.2019]

[10] https://developer.android.com/training/data-storage/room [04.10.2019]

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 14 (2020) 48-54
Wysłane: 2019-10-17
Przyjęte: 2019-10-28

Ocena świadomości studentów informatyki w zakresie bezpieczeństwa komunikatorów internetowych

Paweł Stręciwilk*, Grzegorz Kozieł
Politechnika Lubelska, Katedra Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. Komunikatory internetowe, choć obecne z nami już od dziesięcioleci, zyskały dużą popularność w tej dekadzie. Dziś są najczęściej wykorzystywaną formą komunikacji, szczególnie wśród młodych ludzi. Jednak tam, gdzie są przesyłane informacje, istnieje również ryzyko ich przechwycenia, manipulacji czy udaremnienia przekazu. Ważną kwestią stało się zatem bezpieczeństwo komunikatorów internetowych, które powinny chronić nie tylko wrażliwe dane swoich użytkowników, ale także ich prywatność. Niemniej ważną kwestią jest także świadomość użytkowników, którzy z tego oprogramowania korzystają. Czy są oni świadomi zagrożenia wysyłania poufnych informacji tą drogą? Czy sami odpowiednio zabezpieczają się przed ewentualną próbą włamania? Artykuł powstał w celu zaprezentowania oceny świadomości użytkowników w zakresie bezpieczeństwa komunikatorów internetowych.

Słowa kluczowe: komunikacja; bezpieczeństwo; komunikatory internetowe

*Autor do korespondencji.

Adresy e-mail: pawel.streciwilk@pollub.edu.pl, g.koziel@pollub.pl

An Assessment of IT Students' Awareness in the Field of Instant Messengers Security

Paweł Stręciwilk*, Grzegorz Kozieł
Department of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract. Instant Messengers, though being around for decades, only in recent years they have managed to gain much more popularity. Today, they are the most common way of communicating, especially among young people. However, where information is being transferred there is a potential risk of it being intercepted, manipulated or disrupted. Thus, Security of Instant Messengers became a matter of high importance comprising of not only protecting the users' sensitive information but also privacy. Also important is the awareness of the users. Are they aware of the dangers that come with directing sensitive information in this manner? Are they taking the necessary steps to shield themselves from breach attempts? This article was written to present and evaluate the awareness of IT students in the field of instant messengers security.

Keywords: communication; security; instant messengers

*Corresponding author.

E-mail addresses: pawel.streciwilk@pollub.edu.pl, g.koziel@pollub.pl

1. Wstęp

Komunikacja z użyciem komunikatorów internetowych staje się obecnie standardem. Dotyczy to zarówno sfery prywatnej, jak również zawodowej. W sferze prywatnej prym wiedzie Messenger – komunikator opracowany przez Facebook [1]. W sferze zawodowej komunikator zazwyczaj narzucany jest przez firmę, w jakiej użytkownik podejmuje zatrudnienie. Powodem jest konieczność posiadania jednej platformy komunikacji pozwalającej wszystkim pracownikom na łatwą komunikację. Nie mniej istotne jest również bezpieczeństwo oferowane przez zastosowane oprogramowanie[1]. Istotne jest jednak pytanie czy wśród użytkowników prywatnych świadomość bezpieczeństwa jest równie rozwinięta. Aby odpowiedzieć na to pytanie postawiono 3 hipotezy robocze:

H1. Użytkownicy związani z dziedziną informatyki mają wysoki poziom świadomości w sferze bezpieczeństwa komunikatorów internetowych.

H2. Użytkownicy posiadający wysoki poziom wiedzy w dziedzinie bezpieczeństwa wybierają komunikatory ze względu na oferowane przez nie zabezpieczenia.

H3. Dla użytkowników prywatnych podczas używania komunikatorów liczy się przede wszystkim prywatność.

W celu zweryfikowania tych hipotez przeprowadzono ankietę na grupie stu osób. Wszystkie z nich były związane z branżą informatyczną. Większość, bo aż 88 osób, była studentami trzeciego roku informatyki na Politechnice Lubelskiej. Grupa została dodatkowo uzupełniona o 12 osób, które edukację nadal kontynuują lub już ją zakończyły i wciąż rozwijają się w branży. Ankieta miała na celu sprawdzenie jaką świadomość, preferencje i doświadczenia mieli użytkownicy z bezpieczeństwem komunikatorów internetowych.

1.1. Przegląd literatury

Publikacja „Guide to Chat Apps” [1] stanowiła doskonałą drogę, by lepiej zrozumieć naturę komunikatorów internetowych, kierunek ich rozwoju i pozycję we współczesnym świecie. Pokazuje jak ważna jest dziś ta forma komunikacji, zwłaszcza wśród młodszego pokolenia. Pozycja Holta, Freilicha i Chermaka [2] pomogła z kolei zidentyfikować współczesne zagrożenia, na jakie narażeni są użytkownicy. Dziś hakerzy atakują już nie tylko kierowani chęcią szybkiego zarobku, ale przyświecają im także szkodliwe idee – takie jak ekstremizm popychający do terroryzmu. Targetowanie przez nich swoich ofiar daje dodatkowe powody, by szczególnie dbać o swoje bezpieczeństwo i prywatność. Lektura pozycji Mitnicka, Simona oraz Wozniaka [3] daje wgląd w jedną z najsłabszych ogniw bezpieczeństwa niemal każdego systemu – element ludzki. Można wysunąć dość jasny wniosek, że każdy system jest tak bezpieczny jak społeczność go tworząca. Zainspirowało to stworzenie ankiety wśród użytkowników, by sprawdzić czy ich świadomość w zakresie bezpieczeństwa jest wystarczająca, by nie narażać systemu (a więc także innych użytkowników) na luki ze swojej strony. Powstanie artykułu zostało poprzedzone także odpowiednim przygotowaniem z podstawowych pojęć z dziedziny kryptografii, by móc lepiej zrozumieć mechanizmy stojące za szyfrowaniem przekazywanych informacji i ewentualne problemy, jakie te może napotkać [4]. Pozostałe pozycje [5-10] stanowiły lektury uzupełniające, które pozwoliły lepiej przygotować się do omówionego tematu, poprzez spojrzenie na problem bezpieczeństwa w informatyce z różnych perspektyw.

2. Porównanie komunikatorów

Rynek komunikatorów internetowych nie może narzekać na pustki czy stagnację. Użytkownicy mają do wyboru niezliczone ilości komunikatorów – zależnie od potrzeb na pewno znajdzie się pozycja, która powinna je spełnić. Oczywiście nie sposób opisać je wszystkie, więc na podstawie przeprowadzonej ankiety zawężono wybór, a następnie krótko opisano kilka z nich. Wybrano Facebook Messenger, WhatsApp, Telegram, Discord oraz Skype. Wszystkie posiadają unikalne dla siebie cechy. Facebook Messenger jest komunikatorem wywodzącym się z największej sieci społecznościowej na świecie, Facebooka, i w związku z tym cieszy się ogromną bazą użytkowników. WhatsApp jest jedną z pierwszych aplikacji, która przebiła się do świadomości użytkowników (dziś mówimy o liczbie ponad 1 miliarda pobrań) i przyśpieszyła rozwój rynku komunikatorów internetowych. Telegram jest aplikacją konkurencyjną dla WhatsAppa, stawiającą głównie na bezpieczeństwo i ochronę prywatności. Bardzo szybko zyskujący na popularności Discord jest aplikacją przeznaczoną dla bardzo konkretnej grupy odbiorców, wokół której rozwija swoje funkcjonalności. Skype natomiast może pochwalić się silnym zakorzenieniem wśród świadomości użytkowników. Wszystkie te aplikacje oczywiście różnią się nie tylko funkcjonalnością, ale również podejściem ich twórców do zagadnień bezpieczeństwa oraz prywatności swoich użytkowników.

Tabela 1. Wykorzystywane przez komunikatory szyfrowanie

Nazwa komunikatora	**Szyfrowanie**	
	Podstawowe algorytmy	**End-to-end**
Messenger	Tak	Opcjonalne
WhatsApp	Nie	Tak
Telegram	Tak	Opcjonalne
Discord	Brak informacji	Nie
Skype	Tak (nie zawsze)	Nie

Komunikatory te zostały poddane analizie bezpieczeństwa i prywatności na podstawie ogólnie dostępnych informacji na ich temat – zaczynając na tych udostępnionych przez samych twórców, na artykułach o złamanych zabezpieczeniach kończąc. Najważniejsze uwagi zostały zaprezentowane w Tabeli 2.

Tabela 2. Wyniki szczegółowej analizy poszczególnych komunikatorów.

Nazwa	**Uwagi**
Messenger	Ogromna baza użytkowników. Duża ilość zbieranych o użytkownikach informacji. Skanowanie wiadomości w sposób automatyczny oraz okazjonalne czytanie ich przez pracowników firmy. Dobre zabezpieczenie na etapie logowania.
WhatsApp	Dobra edukacja użytkowników na temat bezpieczeństwa. Obowiązkowe szyfrowanie end-to-end. Brak jakiegokolwiek szyfrowania kopii zapasowej historii rozmów. Długi czas reakcji na krytyczne błędy bezpieczeństwa.
Telegram	Decentralizacja przechowywania kluczy szyfrujących dane w chmurze. Możliwość usuwania wiadomości po stronie rozmówcy. Szybka reakcja na krytyczne błędy bezpieczeństwa. Wysoki poziom prywatności. Liczne funkcjonalności komunikatora wpływające na bezpieczeństwo – np. brak możliwości wykonania zrzutów ekranu, opcja samozniszczenia konta, dodatkowy ekran blokujący komunikator). Ograniczona dostępność w niektórych regionach.
Discord	Szczątkowe informacje na temat bezpieczeństwa. Funkcje prywatności ograniczone do ustawień wewnątrz grupy i trybu streamowania. Rozwój oprogramowania dyktowany przypodobaniem się graczom.
Skype	Brak szyfrowania połączeń podczas używania sieci publicznych. Brak szyfrowania gromadzonych plików lokalnych. Brak funkcjonalności zwiększających bezpieczeństwo i prywatność.

Analiza ta pozwoli na lepsze zrozumienie wartości, jakimi kierowali się ankietowani podczas wyboru używanych przez siebie komunikatorów – a tym samym wyników przeprowadzonej ankiety.

3. Badanie

3.1. Metodyka badań

Badanie zostało podzielone na dwie części. Pierwszą z nich jest przedstawiona już analiza komunikatorów internetowych, drugą z kolei przeprowadzenie ankiety badającej kilka aspektów związanych z bezpieczeństwem. Obie części uzupełniają się, by możliwe było wyciągnięcie jak najlepszych wniosków.

3.2. Dobór grupy badawczej

Grupa badawcza stanowiła 100 osób, 88 z nich to studenci Politechniki Lubelskiej trzeciego roku informatyki. Grupa ta została dodatkowo uzupełniona o 12 osób związanych z branżą informatyczną, by sprawdzić, czy pojawią się ewentualne odchyły w uzyskanych wynikach. Założono, że wybrana grupa powinna być bardziej świadoma istniejących zagrożeń aniżeli przeciętni użytkownicy, na co dzień niezwiązani w żaden sposób z informatyką.

3.3. Przebieg badań

Przeprowadzona ankieta pokryła kilka kwestii związanych z bezpieczeństwem komunikatorów internetowych. Użytkowników zapytano o:

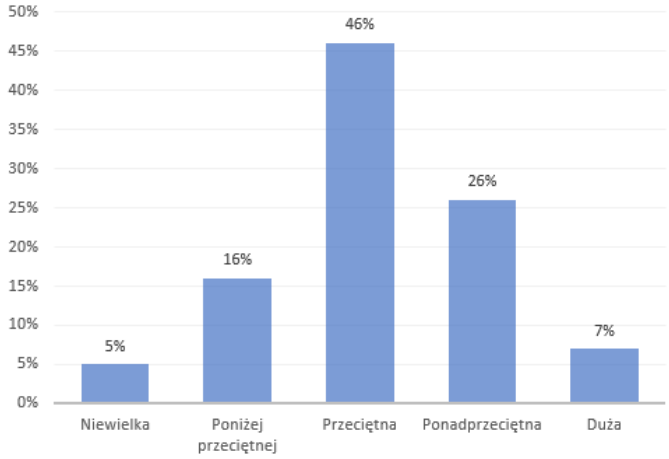
- wiedzę w zakresie cyberbezpieczeństwa,
- ich preferencje względem używanych komunikatorów, co wpłynęło na ich wybór,
- przywiązywanie uwagi do bezpieczeństwa komunikacji,
- podejście do zapewnienia prywatności,
- jak sami dbają o swoje bezpieczeństwo,
- doświadczenia związane z naruszeniem tego bezpieczeństwa,
- co postrzegają jakie największe zagrożenie dla swojego bezpieczeństwa.

Wyniki zostały następnie przeanalizowane. Wzięto pod uwagę zarówno pojedyncze odpowiedzi oraz wyciągnięto wnioski z krzyżowania wybranych pytań.

4. Dyskusja wyników

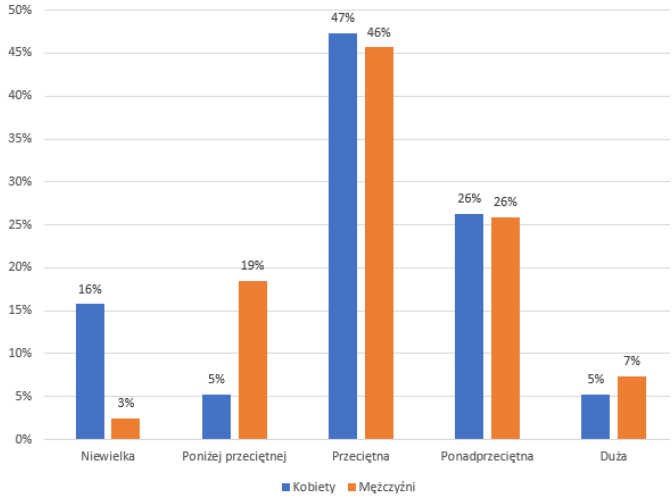
Ankieta została przeprowadzona w środowisku, któremu zagadnienia z dziedziny kryptologii nie powinny być trudne do przyswojenia. Zapytano ich w takim razie, jak oceniają swoją wiedzę o cyberbezpieczeństwie. Można było odpowiedzieć w pięciostopniowej skali – od wiedzy niewielkiej do dużej. Wyniki zaprezentowano na Rys.1.

Jak widać zdecydowana większość (46%) określa swoją widzę jako przeciętną – dając sobie pewne miejsce na uzupełnienie braków. Pośród pozostałych użytkowników 26% deklaruje, że ich wiedza jest ponadprzeciętna, a 7% mówi wręcz o dużej wiedzy. Szczególnie ta grupa była bardzo pomocna w ocenieniu, które komunikatory zasługują na uwagę pod względem bezpieczeństwa. Dodatkowo 16% oceniło swoją wiedzę na poniżej przeciętną, a 5% na niewielką – dalsza analiza wyników pokaże czy nie jest to dyktowane dozą ostrożności w tym temacie.



Rys, 1. Deklarowana wiedza o cyberbezpieczeństwie

Naturalnym następstwem byłoby także zbadanie korelacji, między różnymi grupami badanych – ta jest jednak bardzo jednolita. Szczególnie jeżeli chodzi o wiek badanych wahający się między 21 a 23 lata, z kilkoma wyjątkami. Na Rys.2. można jednak sprawdzić jak deklarowana wiedza wygląda przy reprezentacji różnych płci.

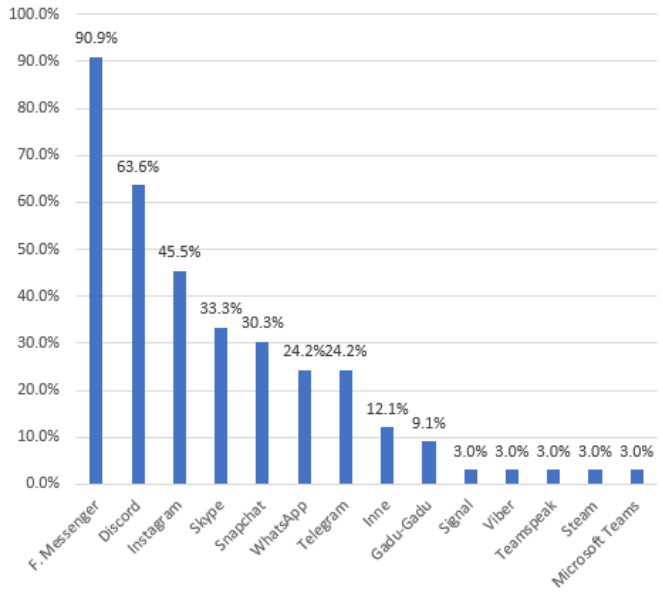


Rys. 2. Deklarowana wiedza wśród kobiet i mężczyzn

Kobiety i mężczyźni są dość zgodni jeżeli chodzi o ocenę swojej wiedzy o cyberbezpieczeństwie. Jedyna niewielka różnica występuje przy deklarowaniu wiedzy niewielkiej i poniżej przeciętnej – nie mając przy tym większego wpływu na całość. Można to tłumaczyć faktem, iż mimo podziału na płeć wszystkie te osoby dzielą wspólną pasję, rozwijając się na uczelni na kierunku Informatyki, której to kryptologia jest nieodłącznym elementem. Osoby takie chętnie również

adaptują nowe technologie, co może skutkować w zgłębianie się w szczegóły działania wybieranych rozwiązań.

Wracając do samych komunikatorów, biorąc pod uwagę jedynie osoby o ponadprzeciętnej i dużej wiedzy o cyberbezpieczeństwie możemy sprawdzić na Rys.3. jakich komunikatorów używa ta grupa i sprawdzić czy pojawia się tendencja do komunikatorów uznawanych za bezpieczne.



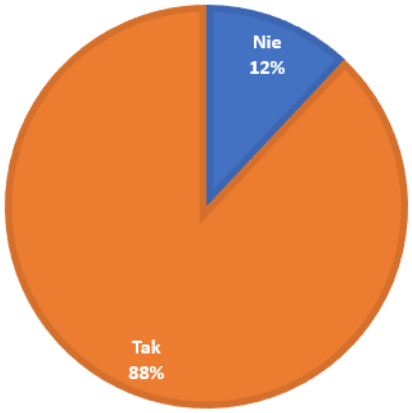
Rys.3. Komunikatory używane przez osoby o ponadprzeciętnej i dużej wiedzy o cyberbezpieczeństwie.

Wykres jasno pokazuje, że duża wiedza nie ma większego wpływu na wybór używanych komunikatów. Wysokie pozycje Facebook Messengera, Snapchata i Instagrama można jednak tłumaczyć nie tyle chęcią komunikacji, co posiadania konta na portalu społecznościowym. Pamiętać należy także, że portale te posiadają dużą bazę użytkowników, którzy mogą być niedostępni nigdzie indziej. Jednak wysoka świadomość zagrożenia może nadal w dużym stopniu wpłynąć na sposób w jaki korzystają z nich użytkownicy.

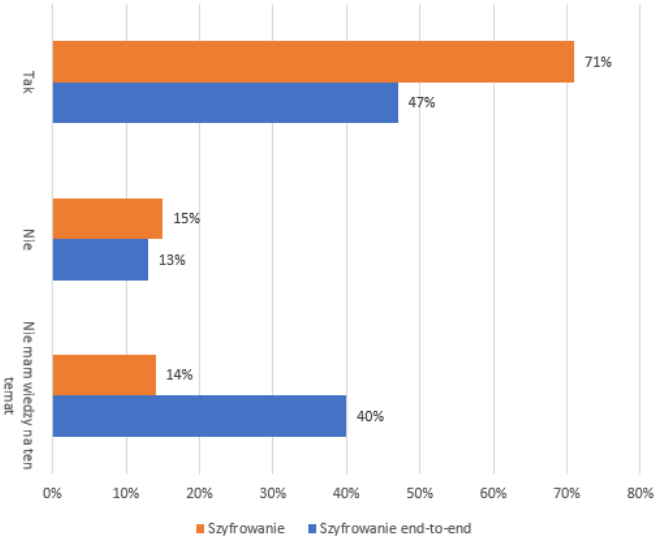
By rozwinąć tę kwestię i poszukać odpowiedzi gdzie indziej, zapytano czy użytkownicy zwracają uwagę na swoje bezpieczeństwo korzystając z owych komunikatorów, a tym samym są świadomi istniejącego zagrożenia.

Diagram widoczny na Rys.4. może budzić optymizm, biorąc pod uwagę, że aż 88% ankietowanych zwraca uwagę na bezpieczeństwo. Można więc założyć, że mimo że wiedza o cyberbezpieczeństwie nie wpływa bezpośrednio na wybór komunikatora, to wpływa na sposób ich używania.

Uzupełniającym pytaniem będzie, czy użytkownikom zależy na szyfrowaniu oraz szyfrowaniu end-to-end. Przedstawia to Rys. 5.



Rys. 4. Procent wszystkich użytkowników zwracających uwagę na bezpieczeństwo komunikacji.



Rys. 5. Procent użytkowników, którym zależy na szyfrowaniu.

Biorąc pod uwagę duży procent użytkowników, którzy zwracają uwagę na bezpieczeństwo oraz równie duże zapotrzebowanie na szyfrowanie wiadomości można wyciągnąć pewne wnioski. Mimo deklarowania przez większość przeciętnej wiedzy na temat cyberbezpieczeństwa jest ona wystarczająca, by świadomie korzystać z komunikatorów internetowych i skutecznie zapobiegać zagrożeniom.

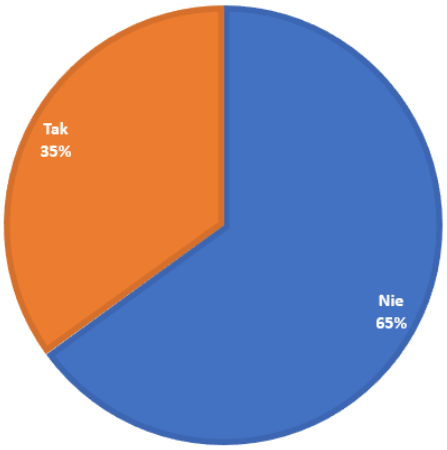
Można więc częściowo zweryfikować hipotezę H1. Użytkownicy związani z branżą informatyczną mają dużą świadomość zagrożenia. Nie przekłada się to jednak na posiadanie specjalistycznej wiedzy z tej dziedziny, czego dowodzi m.in. 40% użytkowników, którzy nie mają wiedzy na temat szyfrowania end-to-end – jednego z podstawowych zagadnień w bezpieczeństwie komunikatorów internetowych.

Pamiętając, że użytkownicy o dużej wiedzy o cyberbezpieczeństwie korzystają z tych samych komunikatorów co reszta grupy, warto zadać sobie pytanie co kieruje ich wyborem.

Zapytano użytkowników o powody, z których korzystają z wybranych rozwiązań. Oto wybrane odpowiedzi:

- „Bezpieczeństwo, funkcjonalność, design”,
- „Znajomi ich używali”,
- „Funkcjonalność”,
- „Nacisk otoczenia ” (przetłumaczone z „peer pressure”),
- „Prywatność, bezpieczeństwo, łatwość użycia”,
- „Bezpieczeństwo, naklejki, wygoda”,
- „Coraz mniej osób korzystało ze starego komunikatora, przeniosłem się tam, gdzie miałem najwięcej znajomych – dodatkowo miał fajne ficzery.”,
- „Ilość znajomych korzystających z danego rozwiązania, oferowana funkcjonalność, bezpieczeństwo rozmów”,
- „Ilość znajomych korzystających z komunikatora”,
- „Duża ilość znajomych”.

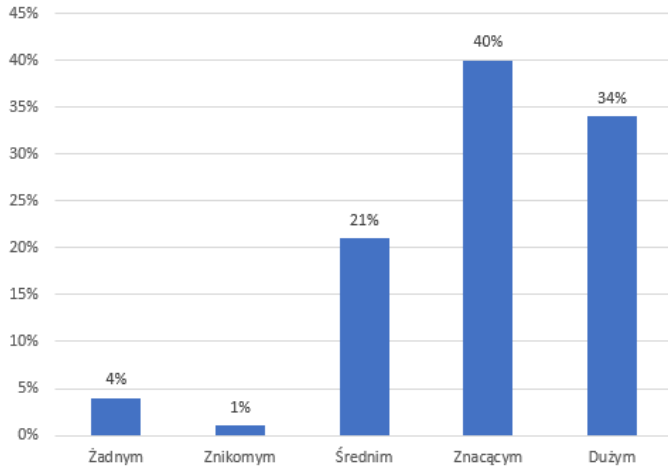
Jak widać, kryteria są dość uniwersalne – liczba funkcjonalności, prywatność, bezpieczeństwo, ale przede wszystkim liczba znajomych korzystająca już z danego rozwiązania. To dowodzi, czemu to właśnie komunikatory wywodzące się z sieci społecznościowych mają największą liczbę użytkowników. Uzupełniającym pytaniem było zapytanie wprost, czy użytkownicy kierują się bezpieczeństwem komunikatora podczas jego wyboru.



Rys. 6. Procent użytkowników kierujących się bezpieczeństwem podczas wyboru komunikatora.

Dla aż 65% użytkowników bezpieczeństwo nie jest priorytetem podczas tego wyboru. Mimo świadomości zagrożenia, to inne czynniki wpływają na używane komunikatory. Pozwala to obalić hipotezę H2.

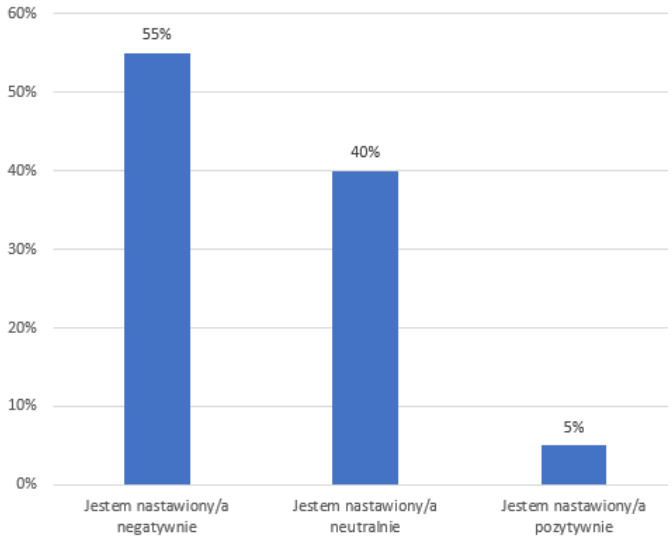
Ostatnią do omówienia kwestią jest prywatność. Mimo iż pojęcie prywatności jest bardzo bliska pojęciu bezpieczeństwa w przypadku komunikatorów internetowych, należy zaznaczyć, że nie każdy gwarantujący bezpieczeństwo komunikator, gwarantuje także prywatność. Ma to związek z przyjętą przez twórców polityką. Zapytano więc użytkowników jak bardzo cenią sobie prywatność. Przyjęto pięciostopniową skalę – od prywatności nie będącą istotną w żaden sposób do dużego znaczenia. Wynik widoczny jest na Rys. 7.



Rys. 7. Stopień w jakim użytkownicy cenią sobie swoją prywatność.

Jak widać nie jest to kwestia użytkownikom obojętna. Jedynie nieliczni ankietowani odpowiedzieli, że prywatność jest dla nich w żaden lub znikomy sposób istotna. Większość ankietowanych uważa, że prywatność jest dla nich aspektem znaczącym (40%) lub bardzo ważnym (34%). Z kolei 21% użytkowników ceni sobie swoją prywatność w średnim stopniu.

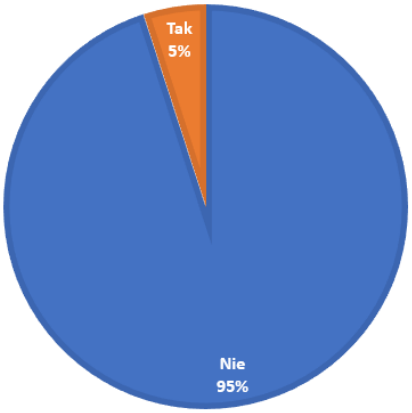
Głównym zagrożeniem dla prywatności jest oczywiście dostęp do wiadomości przez inne osoby. Czasem jednak to dostawcy oprogramowania korzystają z dostępu do treści rozmów w różnych celach: moderacja, analiza w celu dobierania reklam itd. W większości przypadków dzieje się to z wykorzystaniem zautomatyzowanych rozwiązań. Jakie zdanie mają użytkownicy na temat analizy treści ich konwersacji możemy zobaczyć na Rys. 8.



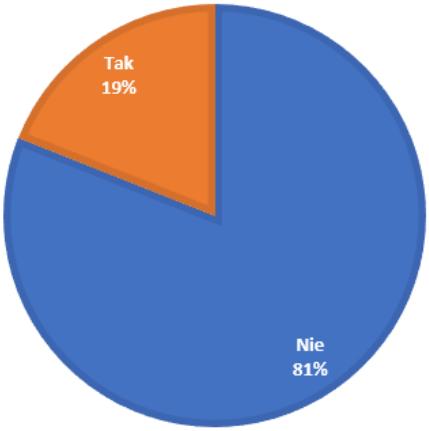
Rys. 8. Podejście użytkowników do zautomatyzowanej analizy treści.

Użytkownicy bardzo niechętnie podchodzą do tego typu praktyk. Ponad połowa ankietowanych – 55% – jest negatywnie nastawiona do analizy treści ich prywatnych wiadomości. Tylko 5% ma na ten temat pozytywne zdanie.

W celu uzupełnienia wyników zapytano użytkowników, czy zrezygnowaliby z prywatności na rzecz lepszego dopasowania materiałów marketingowych (Rys. 9) oraz pomocy służbom bezpieczeństwa narodowego (Rys. 10).



Rys. 9. Procent użytkowników, którzy byliby skłonni zrezygnować z prywatności na rzecz lepszego dopasowania materiałów marketingowych.



Rys. 10. Procent użytkowników, którzy byliby skłonni zrezygnować z prywatności na rzecz bezpieczeństwa narodowego.

Przedstawione wyniki nie pozostawiają pola do dyskusji. Użytkownicy bardzo niechętnie rezygnują ze swojej prywatności. Jest to dla niech znacznie bardziej wrażliwy temat niż samo bezpieczeństwo komunikacji (które samo w sobie może oczywiście bardzo negatywnie odbić się na prywatności). Jedynie 5% ankietowanych byłaby skłonna zrezygnować z prywatności, by otrzymywać lepiej dopasowane materiały marketingowe, zbudowane na bazie wysyłanej przez siebie treści. Wszyscy pozostali nie godzili się na takie rozwiązanie. Podobną tendencję można zaobserwować przy rezygnacji z prywatności na rzecz bezpieczeństwa narodowego. Tutaj już nieco więcej – 19% ankietowanych byłaby skłonna zrezygnować z prywatności, by zapewnić służbom bezpieczeństwa odpowiednie informacje. Nadal 81% jest przeciwna temu pomysłowi, ponad wszystko ceniąc swoją prywatność. Pozwala to w pełni zweryfikować tezę H3.

5. Wnioski

Wspierając przeprowadzone badanie, artykuł miał na celu ocenę świadomości studentów informatyki w zakresie bezpieczeństwa komunikatorów internetowych. Artykuł odpowiedział na 3 postawione przed nim hipotezy. Dwie z nich zostały zweryfikowane, jedna z kolei obalona. Dodatkowo pozwolił na wysunięcie kilku wniosków, które mogą być przydatne zarówno dla użytkowników jak i twórców oprogramowania.

Przede wszystkim rynkiem komunikatorów internetowych steruje ich popularność i bardzo ciężko przełamać tę barierę alternatywnym podejściem. Duża baza użytkowników jest głównym powodem używania konkretnych rozwiązań, stąd też komunikatory wywodzące się z sieci społecznościowych cieszą się bardzo dużym zainteresowaniem. Bezpieczeństwo schodzi na drugi plan. Nawet świadomi użytkownicy rzadko kierują się tym kryterium podczas wyboru.. Wybierają komunikatory, na których już są ich znajomi, tym samym jeszcze bardziej powiększając ich bazę użytkowników.

Duża wiedza o cyberbezpieczeństwie może jednak pozytywnie wpłynąć na sposób korzystania z komunikatorów internetowych oraz oczekiwania wobec nich. Osoby takie mogą wpłynąć pozytywnie na bezpieczeństwo całej swojej sieci kontaktów, poprzez odpowiednie edukowanie swoich rozmówców, kiedy oprogramowanie nie robi tego dostatecznie dobrze. Może to skutkować tworzeniem się mniejszych sieci kontaktów, które będą opierać swoją komunikację na bezpieczniejszych rozwiązaniach i powoli zyskiwać nowych członków.

Niezwykle ważnym elementem dla użytkowników jest poszanowanie ich prywatności. Zdecydowana większość z nich negatywnie wypowiada się na temat jej naruszenia, nawet w takich celach jak zwiększenie bezpieczeństwa narodowego. Jednak mimo to, to właśnie Facebook Messenger (uznany w analizie za najmniej szanujący prywatność) jest najczęściej używanym komunikatorem wśród badanej grupy.

Należy także zaznaczyć, że żaden z ankietowanych nie jest ograniczony do wyboru jedynie jednego komunikatora. Zależnie od potrzeb i sieci znajomych może on korzystać z różnych rozwiązań jednocześnie.

Na koniec można wyciągnąć pewną lekcję. By zapewnić sobie bezpieczeństwo podczas używania komunikatorów internetowych, nie wystarczy jedynie wybór dobrego oprogramowania. Użytkownik oraz jego rozmówca muszą być w pełni świadomi ewentualnych zagrożeń i skutecznie im zapobiegać – w trosce o obu z nich.

Literatura

[1] Barot T., Oren, E., „Guide to chat apps.”, 2015, Tow Center for Digital Journalism, Columbia University,

[2] Holt T., Freilich J., Chermak S., „Exploring the Subculture of Ideologically Motivated Cyber-Attackers”, 2017,

[3] Mitnick K., Simon W., Wozniak S., „Sztuka podstępu. Łamałem ludzi, nie hasła.”, 2001, Wydawnictwo Helion,

[4] Karbowski M. „Podstawy Kryptografii.", 2006, Wydawnictwo Helion,

[5] Erickson J., „Hacking: The Art Of Exploitation, 2nd Edition", 2008, No Starch Press,US,

[6] Stallings W., Brown L., „Computer Security: Principles and Practice, Global Edition", 2018, Pearson Education Limited,

[7] Easttom W., „Computer Security Fundamentals", 2016, Pearson Education (US),

[8] Goodrich M., Tamassia R., „Introduction to Computer Security: Pearson New International Edition", 2013, Pearson Education Limited,

[9] Gunasekera S., „Android Apps Security", 2012. Apress,

[10] Szeliga M., Mendrala D., „Bezpieczeństwo Twojego komputera", 2004 , Wydawnictwo Helion.

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 14 (2020) 55-58
Wysłane: 2019-10-14
Przyjęte: 2019-10-23

Metody wyznaczania wskaźników podobieństwa ruchu trójwymiarowego

Piotr Flisiak*, Marcin Kuszyk

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

Streszczenie. Praca zawiera sposoby porównywania ruchów trójwymiarowych, zapisanych w pliku C3D za pomocą korelacji liniowej Pearsona i błędu średnio-kwadratowego. Została zbadana także ich dokładność. Ruchy osoby ćwiczącej na ergometrze na dystansie 500 metrów zostały zarejestrowane technologią akwizycji ruchu. W artykule została przedstawiona utworzona aplikacja, użyta do analizy.

Słowa kluczowe: Porównywanie ruchu; Ruch trójwymiarowy; Algorytmy podobieństwa ruchu.

*Autor do korespondencji.

Adres e-mail: piotr.flisiak.it@gmail.com

Methods of determining indicators of similarity of 3D motion

Piotr Flisiak*, Marcin Kuszyk

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

Abstract. This paper includes methods for comparing three-dimensional movements with data saved in a C3D file usingPearson's linear correlation algorithms and a mean-quadratic error. Their accuracy was also examined. The movements of a person rowing on an ergometer at a distance of 500 meters were recorded by motion acquisition technology. The article presents the created application used for analysis.

Keywords: Comparing moves; Three-dimensional movement; Movement similarity algorithms.

*Corresponding author.

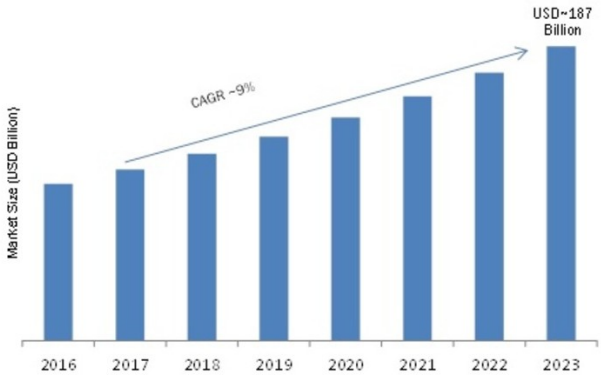
E-mail address: piotr.flisiak.it@gmail.com

1. Wstęp

Ruch trójwymiarowy jest podstawą w życiu człowieka, dlatego też od dawna ludzie pracowali nad jego jak najbardziej dokładnym opisem, aby móc zrozumieć jego charakterystykę i jak najlepiej go wykorzystać. Obecnie w świecie komputerowym coraz dokładniej symulowany jest ten ruch wykorzystując znane światu prawa fizyki i nowe technologie, jedną z nich jest metoda przechwytywania ruchu motioncapture. Motion capture (mocap) to próbkowanie i rejestrowanie ruchu ludzi, zwierząt i nieożywionych obiektów jako dane 3D. Dane mogą być wykorzystane do badania ruchu lub do animowania życia modeli komputerowych 3D. Ponieważ większość aplikacji mocap wymaga obecnie specjalnego sprzętu, nadal istnieje ograniczona liczba firm, szkół i organizacji korzystających z technologii mocap [1]. Zostaje jednak pytanie jak sprawdzić czy dany ruch jest prawidłowy i powtarzalny. W dalszej części artykuły zostanie przedstawiona odpowiedź na to pytanie. W tym celu napisano aplikację analizującą dwa algorytmy porównywania ruchu na plikach C3D.

2. Zastosowanie i rynek

Na początku należy określić przeznaczenie danego algorytmu, gdyż nie do każdego celu będzie pasował dany algorytm porównujący ruch. W medycynie na ten przykład mniej będzie ważna sama pozycja danego znacznika a bardziej kąt pomiędzy nimi aby sprawdzić np. zakres ruchu [2]. W sporcie natomiast korzystać się będzie zarówno z kąta, jak i z samej pozycji znacznika ponieważ ważna będzie też odległość jaką pokona sportowiec zasięg kończyn, czy pozycja wykorzystywanych przedmiotów[3]. Inne podejście należy zastosować jeśli będzie trzeba porównać ruch przedmiotów, gdyż w sporej części przypadków ważna będzie raczej pozycja znaczników względem siebie lub w przestrzeni, a mniej ważny będzie kąt między nimi. Dlatego najpierw trzeba się zastanowić, co będzie porównywane, a następnie trzeba znać cel porównywania ruchów, aby odpowiednio dobrać do niego metodę rejestracji ruchu i jego porównywania.



Rys. 1. Prognoza rynku Motion Capture [4]

Według opisu raportu, w 2023 roku rynek systemów Motion Capture wyniesie około 187 miliardów dolarów, a współczynnik wzrostu (CAGR) miedzy 2017 a 2023 około 9% (Rys. 1).

3. Aplikacja i narzędzia

Aplikacja wykorzystana w badaniu została napisana w języku C# z wykorzystaniem, biblioteki C3D.NET oraz technologii WPF (Windows Platform Forms). Program ma za zadanie obliczać współczynniki korelacji Pearsona oraz błędu średnio-kwadratowego dla wczytanego pliku C3D.

C3D.NET jest darmową biblioteką zawierającą klasy i metody do obróbki plików C3D. Jest rozpowszechniana na zasadzie licencji MIT [5].

Projekt pliku C3D był pierwotnie napędzany potrzebą wygodnego i wydajnego formatu do przechowywania danych zebranych w środowisku przechwytywania ruchu. Format przechowuje współrzędne 3D i dane liczbowe dla każdej próby pomiarowej, ze wszystkimi różnymi parametrami opisującymi dane, w jednym pliku. Eliminuje to potrzebę składowania danych o ruchu z dodatkowymi notatkami i informacjami testowymi (które zwykle są oddzielane w dalszym procesie) [6].

4. Sposoby porównywania

Istnieje wiele sposobów porównywania do siebie ruchów, w literaturze jednak najczęściej pojawiały się tylko 3 z nich. Najczęściej korzystano z najprostszego, czyli porównywania pozycji każdego punktu. Jest to najprostszy sposób, jednakże jest on mało miarodajny i przydatny tylko w dokładnie określonych warunkach. Za przykład może posłużyć pływanie na ergometrze, gdzie ruch był porównywany w odniesieniu do stabilnego punktu, co umożliwiło wybranie klatek, w których punkt znajdował się najbliżej statycznego markera i wycięcia odpowiednich faz ruchu [7].

Lepszym rozwiązaniem jest zastosowanie algorytmu obliczania błędu średnio-kwadratowego. Jest to sposób bardzo podobny do poprzedniego, jednakże uwydatnia on bardziej różnicę w ruchach. W przeciwieństwie do poprzedniej metody, jako wynik zwracana jest nie tablica a konkretna liczba, dokładniej mówiąc średnia z kwadratu różnicy obu analizowanych ruchów. Daje to możliwość automatycznego wykorzystania wyniku porównania w aplikacji, co rozszerza możliwości jego zastosowania [8]. Wzór prezentuje się następująco:

$$\frac{1}{n}\sum_{i=1}^{n}(x_i - y_i)^2 \qquad (1)$$

gdzie:

- x_i, y_i – i-te wartości obserwacji z populacji X i Y,
- X- pierwszy zbiór danych (ruch wzorcowy),
- Y-drugi zbiór danych (ruch porównywany),
- n - liczba obserwacji (X i Y mają tyle samo obserwacji).

Ostatnim z omawianych sposobów porównywania ruchu będzie współczynnik korelacji liniowej Pearsona. Jest to inaczej współczynnik określający liniową zależność zmiennych losowych, który znalazł dobre zastosowanie wporównywaniu ruchu. Tutaj także wynikiem jest konkretna liczba, dając możliwość wykorzystania jego na przykład w aplikacjach reagujących na ruch człowieka [9]. Wzór na współczynnik korelacji liniowej Pearsona:

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \acute{x})(y_i - \acute{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \acute{x})^2 \sum_{i=1}^{n}(y_i - \acute{y})^2}} \qquad (2)$$

gdzie:

- x_i, y_i - i-te wartości obserwacji z populacji X i Y,
- $\acute{x}$, $\acute{y}$- średnie z populacji X i Y,
- X- pierwszy zbiór danych (ruch z którym porównujemy),
- Y-drugi zbiór danych (ruch który porównujemy).

5. Implementacja algorytmów obliczających współczynniki

Algorytm obliczający błąd średnio kwadratowego okazał się dużo łatwiejszy do zaimplementowania i krótszy od korelacji liniowej Pearsona, która posiada dość skomplikowany wzór. Czas wykonania obliczeń był prawie trzykrotnie krótszy w przypadku błędu średnio kwadratowego w stosunku do korelacji liniowej Pearsona gdzie wynosił

Przykład 1. Algorytm obliczania współczynnika błędu średnio-kwadratowego

```
for (Int32 i = 0; i <liczbaKlatek; i++)
{
    //Wczytanie klatek
    C3DFrame frame1 = file1.AllFrames[i];
    C3DFrame frame2 = file2.AllFrames[i];
    for (Int32 j = 0; j <liczbaPunktow; j++)
    {
        //Wczytanie punktów w klatce
        C3DPoint3DData Point3D1 = frame1.Point3Ds[j];
        C3DPoint3DData point3D2 = frame2.Point3Ds[j];

        //Obliczanie sumy kwadratu różnicy punktów
        wskaznik += (Point3D1.X - point3D2.X) *
(Point3D1.X - point3D2.X);
        wskaznik += (Point3D1.Y - point3D2.Y) *
(Point3D1.Y - point3D2.Y);
        wskaznik += (Point3D1.Z - point3D2.Z) *
(Point3D1.Z - point3D2.Z);
    }
}
//Wyliczenie wskaźnika błędu średnio-kwadratowego
wskaznik = wskaznik / (liczbaPunktow*liczbaKlatek*3);
```

Opis użytych zmiennych w obu algorytmach (przykład 1 i przykład 2):

- file1,file2-porównywane pliki zawierające analogiczne ruchy osoby ćwiczącej na ergometrze,
- frame1,frame2 - pojedyncze klatki porównywanych plików,
- liczbaKlatek - liczba klatek obu plików (w przypadku różnej liczby klatek w plikach jest brana pod uwagę liczba klatek krótszego pliku),
- liczbaPunktow- liczba punktów w pliku.

Przykład 2. Algorytm obliczania współczynnika błędu średnio-kwadratowego korelacji liniowej Pearsona

```
// Obliczenie średniej dla każdego wymiaru w punktach
for (Int32 i = 0; i <liczbaKlatek; i++)
{
    C3DFrame frame1 = file1.AllFrames[i];
    C3DFrame frame2 = file2.AllFrames[i];
    for (Int32 j = 0; j <liczbaPunktow; j++)
    {
        C3DPoint3DData Point3D1 = frame1.Point3Ds[j];
        C3DPoint3DData Point3D2 = frame2.Point3Ds[j];
        //Średnia punktu we wszystkich klatkach
        x1sr[j] += Point3D1.X / liczba;
        y1sr[j] += Point3D1.Y / liczba;
        z1sr[j] += Point3D1.Z / liczba;
        x2sr[j] += Point3D2.X / liczba;
        y2sr[j] += Point3D2.Y / liczba;
        z2sr[j] += Point3D2.Z / liczba;
    }
}
//Obliczenie wskaźnika
for (Int32 i = 0; i <liczbaKlatek; i++)
{
```

```
    C3DFrame frame1 = file1.AllFrames[i];
    C3DFrame frame2 = file2.AllFrames[i];
    for (Int32 j = 0; j <liczbaPunktow; j++)
    {
        C3DPoint3DData Point3D1 = frame1.Point3Ds[j];
        C3DPoint3DData Point3D2 = frame2.Point3Ds[j];
//Obliczanie sumy z licznika we wzorze korelacji liniowej
//Pearsona
        suma += (Point3D1.X - x1sr[j]) * (Point3D2.X -
x2sr[j]) + (Point3D1.Y - y1sr[j]) * (Point3D2.Y - y2sr[j]) +
(Point3D1.Z - z1sr[j]) * (Point3D2.Z - z2sr[j]);
//Obliczanie pierwszej sumy z mianownika we wzorze
//korelacji liniowej Pearsona
        suma1kwadrat += Math.Pow((Point3D1.X -
x1sr[j]),2) + Math.Pow((Point3D1.Y - y1sr[j]), 2) +
Math.Pow((Point3D1.Z - z1sr[j]), 2);
//Obliczanie drugiej sumy z mianownika we wzorze korelacji
//liniowej Pearson
        suma2kwadrat += Math.Pow((Point3D2.X -
x2sr[j]),2) + Math.Pow((Point3D2.Y - y2sr[j]), 2) +
Math.Pow((Point3D2.Z - z2sr[j]), 2);
    }
}
// Wyliczenie wskaźnika korelacji liniowej Pearsona
wskaznik = suma / Math.Sqrt(suma1kwadrat *
suma2kwadrat);
```

6. Badanie

Badanie wykonano z użyciem jednego pliku, z którego zostały wycięte dwa analogiczne ruchy pływania na ergometrze w celu zaobserwowania zachowania algorytmu dla ruchu o takim samym zakresie. Pliki te zostały zapisane w formacie c3d. Następnie porównano pierwsze 10 powtórzeń ćwiczenia z kolejnymi 10 powtórzeniami , a także pierwszy cykl z wprowadzonymi różnymi zakłóceniami, aby sprawdzić odporność algorytmów na możliwe błędy w rejestracji ruchu. Zakłóceń dokonano przez wprowadzenie w kopii ruchu przesunięcia o 5 i o 100 jednostek.

Wyniki działania programu porównującego zapisano w poniższej tabeli 1 w celu lepszej interpretacji.

1) Zakres wskaźnika;
2) Liczba klatek w porównywanym ruchu;
3) Porównanie 10 pierwszych powtórzeń ćwiczenia wykonanego na ergometrze z 10 kolejnymi;
4) Porównanie z tym samym ruchem lecz z wprowadzonymi zakłóceniami:
   a) zakłócenie 4 klatek rozłożonych równomiernie w ruchu
      i. zwiększenie wartości punktów o 5;
      ii. zmniejszenie wartości punktów o 5;
      iii. naprzemienne zwiększenie i zmniejszenie o 5;
      iv. zwiększenie o 100;
      v. zmniejszenie o 100;
      vi. naprzemienne zwiększenie i zmniejszenie o 100;
   b) zakłócenie 20 klatek rozłożonych równomiernie w ruchu
      i. zwiększenie o 5;
      ii. zmniejszenie o 5;
      iii. naprzemienne zwiększenie i zmniejszenie o 5;
      iv. zwiększenie o 100;
      v. zmniejszenie o 100;
      vi. naprzemienne zwiększenie i zmniejszenie o 100;
5) Czas działania w milisekundach;
6) Liczba linii kodu;

Tabela 1. Porównanie algorytmów dla ruchów o klatkach

Metoda porównania algorytmów			Korelacja liniowa Pearsona	Błąd średnio-kwadratowy
1			Od -1 do 1	Od 0 do ∞
2			2982	
3			-0,8282	86783,68
4	a	i.	0,99999929681	0,0335345407
		ii.	0,99999929683	0,0335345406
		iii.	0,99999929584	0,0335345402
		iv.	0,99971875	13,4138162
		v.	0,99971893	13,4138161
		vi.	0,99971847	13,4138,163
	b	i.	0,99999650291	0,167672,702
		ii.	0,99999650283	0,167672,703
		iii.	0,99999647922	0,167672,701
		iv.	0,9986044	67,069081137
		v.	0,9986037	67,069081171
		vi.	0,9985944	67,069081170
5			77,0696	28,0354
6			31	14

7. Wnioski

Algorytm obliczania korelacji liniowej Pearsona można łatwiej wykorzystać w aplikacjach, gdyż jego wyniki można przekonwertować na wartości procentowe. Dodatkowo algorytm wykrywa, czy ruch jest odbiciem lustrzanym przy ujemnych wartościach wskaźnika, co również może znaleźć ciekawe zastosowanie.

Z kolei algorytm wyliczania współczynnika błędu średnio-kwadratowego pokazuje za to w bardziej widoczny sposób wszelkie wychylenia porównywanego ruchu w stosunku do głównego ruchu. Współczynnik ten jest jednak mało przejrzysty ze względu na brak górnej granicy jego wartości. Jego obliczenie jest za to szybsze o około 49 ms, a sam algorytm prostszy od korelacji liniowej Pearsona, dzięki czemu łatwo go zamieścić w aplikacji.

Literatura

[1] M. Kitagawa, B. Windsor. MoCap for Artists: Workflow and Techniques for Motion Capture 1st Edition, 2008.

[2] W. Chwała, P. Maciejasz. Inżynieria biomedyczna Księga współczesnej wiedzy tajemnej w wersji przystępnej i przyjemnej. Uczelniane Wydawnictwa Naukowo-Dydaktyczne AGH, Kraków 2008.

[3] W. Chwała, P. Maciejasz. Fascynacja ruchem – technika Motion Capture, Podstawy inżynierii biomedycznej. Wydawnictwo AGH, Kraków, 2009.

[4] 3D Motion Capture market forecast https://www.marketresearchfuture.com/reports/3d-motion-capture-system-market-3026 [07.10.2019].

[5] C3D.NET libary https://archive.codeplex.com/?p=c3d [07.10.2019]

[6] About C3D file https://www.c3d.org/ [07.10.2019].

[7] K. Kowalczyk, M. Skublewska-Paszkowska. Metoda docinania faz ruchu podczas wiosłowania na ergometrze na podstawie danych trójwymiarowych. Czasopismo Journal of Computer Sciences Institute, nr 5, s. 155-158, 2017.

[8] T. Cloete. Benchmarking full-body inertial motion capture for clinical gait analysis. Stellenbosch University, 2009.

[9] K. Zaborska., K. Jochymczyk-Woźniak, P. Wodarski, A. Bieniek, A.Porównanie systemów przestrzennej analizy ruchu na przykładzie systemów optycznych i akcelerometrycznych. Czasopismo Aktualne Problemy Biomechaniki, nr 9, s 129--134, 2015.

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 14 (2020) 59-64
Wysłane: 2019-10-20
Przyjęte: 2019-10-29

Analiza porównawcza wydajności frameworków Angular oraz Vue.js

Roman Baida*, Maksym Andriienko*, Małgorzata Plechawska-Wójcik
Politechnika Lubelska, Katedra Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. Celem artykułu jest porównanie wydajności popularnych frameworków JavaScriptowych Angular i Vue.js w kontekście tworzenia gier oraz wybór lepszego z nich. Kryteria porównawcze są następujące: czas wymiany danych z serwerem oraz renderowania różnych komponentów aplikacji, ilość zajmowanej pamięci podczas odświeżania informacji o przebiegu gry i przywróceniu użytkownika do bieżącej gry, stopień obciążenia przeglądarki oraz rozmiar plików końcowych. Na podstawie wyników z przeprowadzonych badań można stwierdzić, że bardziej wydajny jest framework Vue.js.

Słowa kluczowe: Angular; Vue; analiza porównawcza; gra przeglądarkowa

*Autorzy do korespondencji.

Adresy e-mail: r.baida@pollub.edu.pl, maksym.andriienko@pollub.edu.pl

Perfomance analysis of frameworks Angular and Vue.js

Roman Baida*, Maksym Andriienko*, Małgorzata Plechawska-Wójcik
Department of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract. The aim of the paper is to compare the performance of popular JavaScript frameworks Angular and Vue.js in the context of game development. The comparative criteria are as follows: time of data exchanging with server and rendering of various application components, memory consumption during refreshing the current game information and restoring the user to the current game, browser load level and size of the final application files. The test results show that the Vue.js framework is more efficient.

Keywords: Angular; Vue; performance analysis; browser game

*Corresponding authors.

E-mail addresses: r.baida@pollub.edu.pl, maksym.andriienko@pollub.edu.pl

1. Wstęp

W dzisiejszych czasach aplikacje oraz strony internetowe rzadko są tworzone bez zastosowania frameworków, a tylko i wyłącznie za pomocą zwykłych języków programowania, takich jak JavaScript, PHP, CSS. Wraz z rozwojem technologii rozwijają się także strony internetowe. Strony statyczne zostały zastąpione stronami dynamicznymi z dużą liczbą elementów interaktywnych, które mogą się zmieniać wielokrotnie w bardzo krótkim czasie. Do tworzenia takich stron potrzebne są frameworki [1], które pomagają programistom poradzić sobie ze złożonością takich projektów. Istnieje wiele różnych frameworków JavaScriptowych, ale na potrzeby niniejszego artykułu zostały porównane tylko dwa z nich, Angular i Vue.js. Angular, który został utworzony przez firmę Google, jest jednym z najpopularniejszych frameworków JavaScriptowych [2]. Każdego roku framework Vue.js staje się coraz bardziej popularny. Został on utworzony został przez Evana You [3], który w przeszłości pracował w Google, wykorzystując AngularJS. Vue.js powstał jako framework, który zawiera w sobie najlepsze cechy frameworka Angular, skupując się na prostocie pisania kodu i małej ilości zajmowanego miejsca plików wykonywalnych na dysku.

Celem badania jest analiza wydajności frameworków JavaScriptowych Angular oraz Vue.js. Porównanie wydajności frameworków będzie polegać na porównaniu średniego czasu wysyłania żądań do serwera i renderowania komponentów stron oraz ilości zajmowanej pamięci podczas wykonywania tych działań. W tym celu stworzona została aplikacja internetowa implementująca grę planszową Munchkin. Na podstawie wyników badań zostanie wykonana analiza porównawcza tych frameworków.

Postawiono następującą tezę:

Framework Vue.js jest lepszym środowiskiem programistycznym do budowy aplikacji internetowych dla początkującego programisty niż framework Angular ze względu na łatwość nauczenia się go oraz wydajność przy założeniu, że tworzona aplikacja internetowa ma średni poziom złożoności.

2. Przegląd literatury

Po przeanalizowaniu dostępnych materiałów można wywnioskować, że większość prac związanych z frameworkami Angular i Vue.js polega na analizie technicznej tych frameworków lub ma za cel charakterystykę tych frameworków.

W artykule *Speed Performance Comparison of JavaScript MVC Frameworks* [4] autor przedstawił analizę wydajności ośmiu frameworków. Przeprowadzone badania polegały na

mierzeniu czasu tworzenia, modyfikowania i usuwania elementów HTML na stronie internetowej. Porównanie wyników tych badań pokazuje, który z frameworków jest najszybszy. Porównując średnie wyniki czasowe, Vue.js jest szybszy od Angular tylko w przypadku usuwania elementów HTML. We wnioskach autor stwierdza, że Angular posiada najlepsze wyniki i zajmuje pierwsze miejsce w prawie wszystkich przeprowadzonych testach. Vue jest jednym z najszybszych frameworków w przypadku tworzenia i modyfikowania elementów, chociaż jest i tak wolniejszy od Angular w wykonywaniu tego typu operacji.

Głównym celem artykułu *Sitecore JavaScript Services Framework Comparison* [5] jest określenie frameworku, który najlepiej nadawałby się do stworzenia serwisu w systemie zarządzania treścią Sitcore. Do wyznaczenia najodpowiedniejszej technologii, autor dokonuje porównania takich metryk jak liczba linii kodu źródłowego, ilość miejsca na dysku zajmowana przez pliki wykonywalne, złożoność pisania kodu, ograniczenia architektury wprowadzane przez framework i wydajność. Na podstawie tych badań wysunięto wniosek, że w celu stworzenia tego typu aplikacji, najlepiej użyć biblioteki React. Autor zauważa również, że framework Vue.js ma ogromne perspektywy na pokonanie React, ze względu na niski próg wejścia, minimalną konfigurację podstawową oraz łatwość rozszerzenia architektury kodu. W przypadku złożonego i bogatego w funkcjonalność projektu Angular jest głównym konkurentem React ze względu na elastyczność kodu i szybkość renderowania stron.

W artykule *Comparison of front-end frameworks for web applications development* [6] dokonano porównania frameworków Angular, Vue.js i React w celu wyznaczenia, który z nich najbardziej nadaje się do tworzenia aplikacji typu SPA (*ang. Single Page Application*), który do tworzenia aplikacji typu MPA (*ang. Multi Page Application*) lub do tworzenia obu rodzajów aplikacji. Przy porównaniu wybranych frameworków, autor porusza takie zagadnienia, jak możliwość generowania strony po stronie serwera, możliwość napisania i zarządzania dużą ilością złożonego kodu oraz możliwości frameworka w przypadku kompresji i minimalizacji plików źródłowych. W wyniku przeprowadzonych badań autor wywnioskuje, że w celu tworzenia MPA najbardziej nadaje się Vue.js, zaś Angular jest najgorszym wyborem ze względu na jego rozmiar i złożoność, jak i również jego nieprzystosowanie do sterowania małymi częściami struktury DOM. W przypadku SPA Angular, zdaniem autora, jest najlepszym wyborem. Na końcu autor stwierdza, że Vue.js jest dobrym wyborem zarówno w przypadku tworzenia SPA, jak i MPA.

Po przeanalizowaniu publikacji naukowych dotyczących tematu porównania frameworków JavaSctiptowych można stwierdzić, że większość z badań przeprowadzonych w tych pracach polega na porównaniu czasu wykonania prostych operacji po stronie klienta. Nie udało się znaleźć prac naukowych, które dokonywałyby porównania czasów wymiany informacją z serwerem albo, na przykład, renderowania całych stron lub poszczególnych ich elementów.

3. Obiekty badań

3.1. Angular 6

Angular 6 jest nową wersją popularnego frameworka do tworzenia dynamicznych stron internetowych typu SPA. Ten framework jest stworzony i wspierany przez firmę Google. Pierwsza wersja framework'u bazowała na języku JavaScript, a od wersji 2+ bazuje na języku TypeScript [7].

Zaletami szkieletu Angular są:

- łatwość pisania kodu aplikacji,
- dwustronne powiązanie zmiennych kodu TWDP (*ang. Two-way data binding*),
- skalowalność,
- przydatność do pracy z interfejsem Rest API,
- podział na komponenty.

Wadami szkieletu Angular są:

- wyższy próg wejścia ze względu na Observable (RxJS) i Dependency Injection [9],
- wymagane jest poświęcenie dużej ilości czasu na optymalizacje w celu uzyskania szybkiego i poprawnego kodu,
- trudność realizacji dynamicznego tworzenia komponentów,
- duży rozmiar zbudowanego kodu ze względu na zbędne funkcję frameworku,
- jest wspierany przez firmę Google, która gwarantuje częste aktualizację i długi rozwój frameworku.

Angular umożliwia tworzenie dużych i złożonych pod względem logiki biznesowej aplikacji internetowych. Angular po wersji AngularJS został stworzony całkowicie na nowo [8]. W wyniku tego, framework ten zyskał większe możliwości skalowania, a jego kod wykonuje się szybciej w porównaniu ze starszą wersją.

W porównaniu do Vue.js, framework Angular posiada więcej możliwości pracy z interfejsem Rest API. Na przykład, umożliwia dokonanie asynchronicznej modyfikacji danych lub monitorowanie zmiany stanów zmiennych za pomocą Observable (RxJS) lub Subject.

3.2. Vue.js

Vue.js jest frontendową biblioteką, która została napisana w języku JavaScript. Biblioteka ta umożliwia łatwe tworzenie aplikacji internetowych opartych o MVVM (*ang. Model-View View Model*). Każda aplikacja napisana na podstawie Vue.js składa się z komponentów, które zawierają kod JavaScript do wykonania w obrębie komponentu, kod stylów CSS i kod HTML warstwy widoku.

Główne zalety Vue.js to:

- niski próg wejścia,
- wydajność uzyskana dzięki wirutalnemu DOM,
- wysoka elastyczność,

- prosta obsługa,
- minimalne rozmiary zbudowanego kodu,
- wysoka szybkość wykonywanego kodu i renderowania stron aplikacji.

Wadami szkieletu Vue.js są:

- realizacja dwustronnego powiązania zmiennych kodu TWDP wymaga implementacji dodatkowych funkcji *computed* lub *watch*,
- trudność rozszerzenia gotowej aplikacji,
- tworzony głównie przez jednego programistę, przy współpracy z niezależną społecznością programistyczną, co powoduje rzadkie aktualizację oraz nie zapewnia długoterminowego rozwoju frameworku.

Praca z Vue.js rozpoczyna się od stworzenia nowej instancji *new Vue* przy pomocy konstruktora. Umożliwia to obserwowanie zmian, które występują w danym komponencie. Następnie instancja Vue przyjmuje zmienną *template*, z której będzie ona korzystać w celu pobrania widoku do renderowania. W atrybucie *data* są przechowywane wszystkie zmienne komponentu, którego stan śledzi Vue.

Komponenty, które zasadniczo są częściami interfejsu użytkownika, pomagają w rozwinięciu podstawowych elementów HTML i implementowaniu kodu wielokrotnego użytku. Na etapie projektowania aplikacja zostaje podzielona na niezależne części w celu uzyskania drzewiastej struktury komponentów, ale ze względu na to, że kod HTML, CSS i JavaScript znajduje się w jednym pliku, rozszerzenie gotowej aplikacji internetowej jest trudniejsze w porównaniu do Angular.

4. Metoda badań

Porównanie wydajności frameworków będzie polegać na porównaniu średniego czasu wysyłania żądań do serwera oraz renderowania komponentów. Każde badanie zostało przeprowadzone dziesięć razy, a z otrzymanych wyników został obliczony wynik średni. W tym celu została stworzona aplikacja internetowa implementująca grę planszową Munchkin. Aplikacja ta wspiera grę wieloosobową, zarządzanie kontem użytkownika, tworzenie pokojów do gry.

Rozgrywka w grze Munchkin polega na użyciu dwóch talii kart, a cały proces tej gry polega na zbieraniu i wymianie pewnych kart pomiędzy graczami oraz rozegraniu tych kart we właściwych momentach gry. Aplikacja internetowa zawiera wszystkie niezbędne karty, a powiadomienie użytkowników o akcjach z nimi związanych odbywa się za pomocą okien typu pop-up. Dlatego jednymi z najważniejszych badaniami są badania określające czas renderowania kart oraz okien typu pop-up.

Do testowania i porównania możliwości wybranych frameworków zostało wykorzystane narzędzie Google Chrome DevTools [10], które pomaga przeglądać i debugować pliki HTML, CSS oraz witryny JavaScript. Pomaga ono również w sprawdzaniu ruchu sieciowego pobieranego przez witrynę, sprawdzeniu szybkości witryny, analizie wydajności środowiska wykonawczego.

Badania zostały przeprowadzone na komputerze HP EliteBook 840 G3. Komputer ten posiada procesor czterordzeniowy Intel Core i5-6300U o częstotliwości ~2.5GHz oraz ma zainstalowaną pamięć RAM DDR4 8GB. Systemem operacyjnym jest Windows 10 Enterprise 64-bit (10.0). Do badań użyto przeglądarki Google Chrome w wersji 77. Serwery aplikacji zostały uruchomione lokalnie. Część backendowa została uruchomiona za pomocą .Net Core CLI (*ang. Command-Line Interface*), a części frontendowe Angular i Vue.js zostały, odpowiednio, uruchomione przy użyciu Angular CLI oraz Vue CLI.

Kryteria według których zostaną porównane obrane frameworki są następujące:

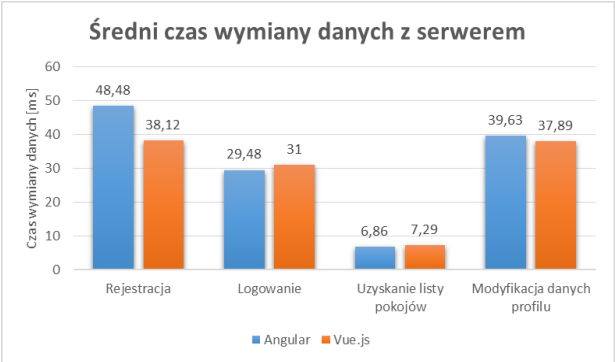
- czas wymiany danych z serwerem,
- czas renderowania komponentów stron,
- czas renderowania kart,
- renderowanie okien typu pop-up
- przywrócenie użytkownika do bieżącej gry,
- odświeżanie informacji o przebiegu gry,
- stopień obciążenia przeglądarek,
- rozmiar plików końcowych.

5. Wyniki badań

5.1. Czas wymiany danych z serwerem

Badanie to zostało przeprowadzone na takich funkcjach aplikacji jak: rejestracja, logowanie, pobieranie listy pokojów dostępnych do gry oraz modyfikacja danych profilu użytkownika. Badane funkcje dokonują wymiany danymi z serwerem, a po otrzymaniu odpowiedzi odświeżają bieżącą stronę lub powiadamiają użytkownika o uzyskanym błędzie. Podczas przeprowadzenia tych badań, liczony był czas od momentu wywołania akcji do wysyłania danych na serwer oraz od momentu odebrania danych do uzyskania reakcji. Czas transferu danych przez sieć nie był uwzględniony.

Na rysunku 1 przedstawiono diagram słupkowy. Oś rzędnych reprezentuje czas w milisekundach, a oś odciętych wskazuje na badaną funkcję. Różnica między średnim czasem wykonywania funkcji dla obu frameworkach jest dość mała. Minimalna średnia różnica wynosi 0,5 milisekundy dla komponentu listy dostępnych pokojów do gry, natomiast maksymalna stanowi 10 milisekund dla komponentu rejestracji. Spowodowane to jest tym, że do komunikacji sieciowej, Angular korzysta ze swojego modułu HttpClient, a Vue.js korzysta z modułu Axios, przy czym HttpClient jest stworzony na bazie Axios. Z tego powodu wyniki są bardzo podobne. Z wyników badań można wywnioskować, że główna różnica polega na czasie wykonania żądań typu GET, gdzie Angular jest wolniejszy.

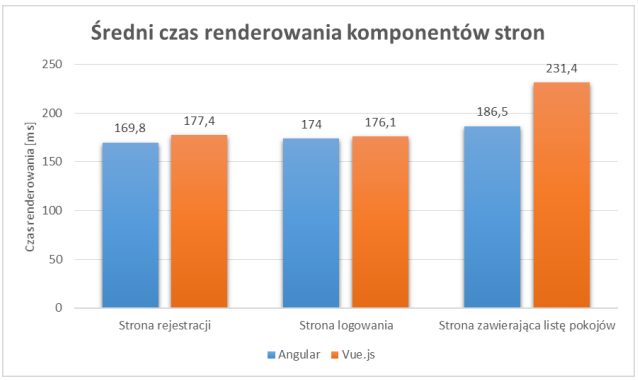


Rys. 1. Diagram pokazujący czas średni wywołania różnych funkcji wybranymi frameworkami

5.2. Czas renderowania komponentów stron

W tym badaniu będą porównywane identyczne funkcje napisane w Angular i Vue.js. To badanie potrzebne jest do porównywania czasów renderowania oddzielnych komponentów aplikacji internetowej. Zostało ono przeprowadzone na następujących komponentach aplikacji: komponent logowania, komponent rejestracji, komponent w którym odbywa się przebieg gry.

Na rysunku 2 przedstawiono diagram słupkowy. Oś rzędnych reprezentuje czas w milisekundach, a oś odciętych wskazuje na badaną funkcję. Różnica między średnim czasem renderowania stron jest bardzo mała. Na podstawie wykresu można wywnioskować, że Vue.js ma dłuższy czas renderowania komponentu zawierającego listę dostępnych pokoi dla gry w porównaniu do Angular. Ta różnica jest spowodowana tym, że Vue.js tworzy instancje do monitorowania bieżącego stanu zmiennej zawierającej listę, co powoduje wydłużenie czasu renderowania. Żeby zwiększyć szybkość renderowania listy pokojów do gry trzeba użyć specjalnego modyfikatora *v-once*, która nie pozwala ustawiać dla zmiennej listy instancję monitorowania. W takim przypadku lista zostanie wyrenderowana jednokrotnie.

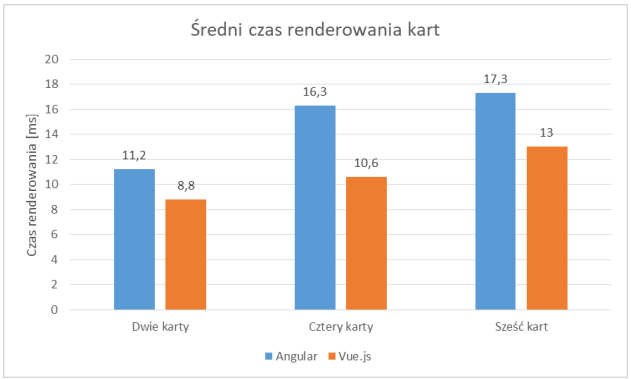


Rys. 2. Diagram pokazujący czas średni renderowania stron wybranymi frameworkami

5.3 Czas renderowania kart

Wyniki tego badania pokazują, w jakim stopniu renderowanie kart obciąża zasoby przeglądarki. Karty są wyświetlane w przypadku ich wyciągnięcia z talii, gdy użytkownik chce zobaczyć karty, które posiada inny gracz lub podczas innych akcji. Jest to bardzo istotne badanie, które pokaże, kompilator którego frameworku jest lepszy pod względem optymalizacji zasobów używanych aplikacją.

Rysunek 3 przedstawia diagram słupkowy. Oś rzędnych reprezentuje czas w milisekundach, a oś odciętych wskazuje na liczbę kart renderowanych w każdym badaniu. Z wyników badań średniego czasu renderowania kart do gry można wywnioskować, że kompilator optymalizacji plików dodatkowych oraz zdjęć jest lepszy w frameworku Vue.js, gdzie dla aplikacji testowej średni czas jest o ~4,2 milisekundy krótszy od czasu renderowania kart w Angular.



Rys. 3. Średni czas renderowania różnej liczby kart

5.4. Renderowanie okien typu pop-up

Ponieważ prawie każda aplikacja internetowa implementuje okna typu pop-up, badanie to jest jednym z najważniejszych w niniejszym artykule. Pozwoli ono wyznaczyć ilość zajmowanej pamięci oraz czas potrzebny do renderowania takich okien. Na potrzeby artykułu zostało przebadane okno, które powstaje podczas akcji sprzedaży niepotrzebnych kart. Analiza wyników tego badania pokaże, który framework lepiej radzi z renderowaniem okien typu pop-up.

Okna typu pop-up stanowią ważną część aplikacji internetowej, ponieważ wyświetlają one ważną informację o przebiegu gry. W zakresie jednej rundy składającej się z czterech etapów, okno typu pop-up pojawia się co najmniej dwa lub trzy razy, a w niektórych przypadkach ich liczba wzrasta wielokrotnie. Dlatego ważne jest to, ile czasu zajmuje renderowanie jednego takiego okna oraz ile pamięci zostaje wykorzystane przy jego pojawieniu się.

W tabeli 1 znajdują się wyniki badań określających średni czas renderowania okien typu pop-up.

Tabela 1. Czas renderowania okien typu pop-up w milisekundach

Framework	Angular	Vue.js
Wartość średnia	55,5	58,4

Tabela 2 pokazuje użytą pamięć w kilobajtach. Ilość pamięci używanej frameworkiem Vue.js jest o trzy razy mniejsza w porównaniu do Angular.

Tabela 2. Ilość zajmowanej pamięci podczas renderowania okien typu pop-up w kilobajtach

Framework	Angular	Vue.js
Wartość średnia	179,9	50,69

Z wyników badań średniego czasu renderowania i wywołania okien typu pop-up wynika, że czas średni jest bardzo podobny i nie ma dużej różnicy, ale przy badaniu pamięci potrzebnej do renderowania widoczne jest to, że Vue.js potrzebuje faktycznie 3 razy mniej pamięci niż Angular. Jest to związane z tym, że we frameworku Vue.js został zaimplementowany lepszy sposób modyfikacji DOM strony, w odróżnieniu do Angular.

5.5. Przywrócenie użytkownika do bieżącej gry

Badanie to umożliwia wyznaczenie czasu potrzebnego na powrót do aktualnego stanu gry użytkownikowi, który przez przypadek zamknął okienko z grą, stracił połączenie z serwerem lub odświeżył stronę. Wtedy, po przejściu na stronę z listą pokojów, użytkownik automatycznie zostaje przywrócony do gry, w której on uczestniczy, a informacja o grze aktualizuje się do aktualnego stanu. Wyniki badania posłużą do określenia, który framework szybciej podłącza się do WebSocket, pobiera dużą ilość danych od serwera oraz renderuje niezbędne komponenty stron.

Tabela 3. Czas przywrócenia użytkownika do bieżącej gry

Framework	Angular	Vue.js
Wartość średnia	88,7	51,3

Tabela 4. Ilość zajmowanej pamięci podczas przywrócenia użytkownika do bieżącej gry w kilobajtach

Framework	Angular	Vue.js
Wartość średnia	53,45	51,55

Jak widać z tabel wyżej (tab. 3 oraz tab. 4), pamięć potrzebna do przywrócenia użytkownika do bieżącej gry w Angular i we Vue.JS jest faktycznie jednakowa, ale czas potrzebny na to działanie w przypadku Angular jest o 50% dłuższy. Jest to przede wszystkim związane z pracą modułu odpowiadającego za dwustronnie przesyłanie danych w zmiennej, gdzie w Angular dana funkcja jest zbyt rozbudowana.

5.6. Odświeżanie informacji o przebiegu gry

To badanie umożliwia wyznaczenie lepszego frameworku pod względem pobierania informacji z serwera i wyrenderowania niezbędnych danych. Na końcu każdej tury oraz podczas wykonywania różnych akcji, serwer wysyła każdemu graczowi informację o aktualnym przebiegu gry w czasie rzeczywistym. Odświeża się bieżący etap rundy, informacja o graczach oraz ich kartach.

Tabela 5. Czas odświeżania informacji o przebiegu gry

Framework	Angular	Vue.js
Wartość średnia	47	31,7

Tabela 6. Ilość zajmowanej pamięci podczas odświeżania informacji o przebiegu gry w kilobajtach

Framework	Angular	Vue.js
Wartość średnia	43,85	43,32

Wyniki tego badania odpowiadają wynikom poprzedniego badania "Przywrócenie użytkownika do bieżącej gry". Z tych tabel (tab. 5 oraz tab. 6) można wywnioskować, że pamięć potrzebna do przywrócenia użytkownika do bieżącej gry w Angular i we Vue.JS jest faktycznie jednakowa, ale czas potrzebny na to działanie w przypadku Angular jest dwukrotnie dłuższy. Jest to głównie związane z pracą modułu odpowiadającego za dwustronnie przesyłanie danych w zmiennej, gdzie w Angular dana funkcja jest zbyt rozbudowana.

5.7. Stopień obciążenia przeglądarek

Badanie to pozwala zweryfikować, ile pamięci wykorzystuje aplikacja. Pamięć podręczna przeglądarki była czyszczona przed każdą kolejną próbą. Potrzebne to jest do określenia frameworku, który obciąża przeglądarkę w mniejszym stopniu. Stopień obciążenia przeglądarek był mierzony podczas uruchomienia aplikacji oraz w przeciągu jednej godziny z próbami co 30 minut. Po skończeniu badań nie wykryto dużych różnic w obciążeniu zależnych od czasu życia aplikacji.

Kod aplikacji i wszystkie pliki graficzne pobierane są przy pierwszym uruchomieniu aplikacji w celu zwiększenia jej wydajności. Dlatego stopień obciążenia przez cały czas jest niemal jednakowy. Nieznaczna różnica rzędu 5-10 kB jest spowodowana sytuacją w przebiegu gry, w której każdy z graczy posiada dużą ilość kart.

Tabela 7. Stopień obciążenia pamięci przeglądarką w zależności od czasu życia aplikacji w kilobajtach

	Angular	**Vue.js**
Uruchomienie	140,9	73
Półgodziny	146,2	77,8
Godzina	151,4	82,5

Jak widać w tabeli 7, średnia ilość pamięci wykorzystanej podczas uruchomienia aplikacji dla Angular wynosi 140,9 kB, a dla Vue.JS 73 kB. Jest to przede wszystkim związane z tym, że framework Angular jest bardziej rozbudowanym i posiada wiele dodatkowych modułów, które od razu zaczynają działać, nawet, wtedy, gdy nie jest to potrzebne. Inną przyczyną uzyskania takich wyników jest to, że framework Vue.js jest nowszy w porównaniu z Angular i jego optymalizacja kodu jest lepsza. To badanie pokazuje, że przy uruchomieniu na urządzeniach mobilnych badanej aplikacji, większe ryzyko zawieszenia urządzenia występuje w przypadku frameworka Angular.

5.8. Rozmiar plików końcowych

Rozmiar plików końcowych został zbadany przy użyciu wbudowanych narzędzi systemu Windows, po zbudowaniu paczki aplikacji internetowej odpowiedniego frameworku. Na podstawie otrzymanych wyników rozmiaru zbudowanej paczki aplikacji można wywnioskować, dla którego frameworku aplikacja zostanie szybciej załadowana przez urządzenie.

Po zbudowaniu aplikacji internetowej gry Munchkin, rozmiar plików końcowych w przypadku użycia Angular wynosi 31,7 MB, a w przypadku użycia Vue.js wynosi 22,1 MB. Rozmiar plików końcowych przy użyciu frameworku Vue.js jest prawie o 10 MB mniejszy. Tak wielka różnica jest powodowana tym, że Angular jest bardziej rozbudowany w porównaniu do Vue.js. Na przykład, Angular posiada bardzo rozbudowany system kontroli życia aplikacji, co powoduje zwiększenie rozmiaru plików końcowych oraz wydłuża czas wykonywania dowolnej funkcji. To, że Vue.js ma mniejszy rozmiar plików końcowych może być spowodowane tym, że Vue.js jest nowszym frameworkiem w porównaniu do Angular i twórcy Vue.js uwzględnili wszystkie tak zwane „wąskie gardła" oraz zbędne funkcje, które występują w Angular i dokonali w stosunku do nich zmian i optymalizacji lub całkowicie z nich zrezygnowali. Dlatego Vue.JS jest szybszy od Angular i wykorzystuje mniej pamięci.

6. Wnioski

Analizując wyniki przeprowadzonych badań można stwierdzić, że szybkość pracy tych dwóch frameworków jest bardzo podobna. Prawdopodobnie związane to jest z tym, że Vue.js rozpoczął swoje istnienie jako odnoga Angular, i jest on napisany w bardzo podobnym do Angular stylu. Dzięki użyciu dobrych technik programowania i uwzględnieniu problemów, które posiada Angular, programistom Vue.js udało się zmniejszyć rozmiar kodu wykonywalnego projektu o 50%. I chociaż na początku te dwa frameworki były bardzo do siebie podobne, Angular w wersji drugiej został napisany w TypeScript, a Vue.js nadal pozostaje rozwijany za pomocą HTML i JavaScript.

Ze względu na te uwagi można wywnioskować, że framework Angular jest bardziej rozbudowany i przez to trudniejszy do nauki, ale dzięki twardej typizacji Angular jest bardzo przydatny w przypadku tworzeniu dużych projektów. Vue.js posiada mniej możliwości i wydaje się być lepszym wyborem dla początkujących programistów, ale jednocześnie jest on bardziej elastycznym frameworkiem niż Angular i może zostać użyty do napisania zarówno dużych, jak i małych projektów.

Po przeanalizowaniu otrzymanych wyników można wywnioskować, że dla początkującego programisty framework Vue.js stanowi lepsze rozwiązanie do tworzenia aplikacji internetowych, dlatego że jest on łatwiejszy do nauki, potrzebuje mniej zasobów komputerowych do uruchomienia i działania napisanej za jego pomocą aplikacji internetowej i jest szybszy w przypadku projektu o niewielkim stopniu złożoności. Warto zauważyć, że wraz ze wzrostem doświadczenia zawodowego programisty i przy tworzeniu dużych projektów framework Angular może stać się bardziej przydatny. Pomimo faktu, że TypeScript, w którym jest napisany Angular, jest bardziej złożony i trudniejszy do nauki, rozbudowanie i wsparcie aplikacji stworzonej przy jego pomocy jest łatwiejsze. Na podstawie wyżej wymienionych faktów można stwierdzić, iż teza mówiąca że *framework Vue.js jest lepszym środowiskiem programistycznym do budowy aplikacji internetowych dla początkującego programisty niż framework Angular ze względu na łatwość nauczenia się go oraz wydajność przy założeniu, że tworzona aplikacja internetowa ma średni poziom złożoności* jest prawdziwa.

Literatura

[1] https://fructcode.com/ru/blog/features-of-popular-frameworks-html-css-php-and-python-frameworks/, Framework — ważne narzędzie programistyczne, [24.07.2019]

[2] https://divante.com/blog/top-10-popular-javascript-frameworks-2019/, Top 10 most popular JavaScript frameworks in 2019, [24.07.2019]

[3] https://en.wikipedia.org/wiki/Vue.js, Vue.js, [27.07.2019]

[4] Svensson A.: Speed Performance Comparison of JavaScript MVC Frameworks. Blekinge Institute of Technology, 2015

[5] Petukhova E.: Sitecore JavaScript Services Framework Comparison. Åbo Akademi University, 2019

[6] Kaluža M., Troskot K., Vukelić B.: Comparison of front-end frameworks for web applications development. Zbornik Veleučilišta u Rijeci, 2018

[7] Seshadri S.: Angular: Up and Running: Learning Angular, Step by Step. O'Reilly Media, 2018

[8] https://habr.com/ru/post/320014/, AngularJS vs Angular, [25.07.2019]

[9] Wilken J.: Angular in Action. Manning Publications, 2018

[10] https://developers.google.com/web/tools/chrome-devtools, Chrome Dev Tools, [12.08.2019]

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 14 (2020) 65-72
Wysłane: 2019-10-23
Przyjęte: 2019-10-30

Analiza możliwości zastosowania gry typu serious game do nauki postępowania podczas udzielania pierwszej pomocy

Klaudia Zaborek*, Małgorzata Plechawska-Wójcik
Politechnika Lubelska, Katedra Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. Celem niniejszej pracy jest analiza możliwości zastosowania gry typu serious game do nauki postępowania podczas udzielania pierwszej pomocy. Zakres pracy zawiera zaprojektowanie i zaimplementowanie gry oraz przeprowadzenie badań i analizę wyników. Przedmiotem badań w pracy jest sprawdzenie, czy wykorzystanie gry typu Serious Game umożliwia zdobycie i utrwalenie wiedzy z zakresu pierwszej pomocy.

Słowa kluczowe: poważna gra; Unity; pierwsza pomoc

*Autor do korespondencji.

Adres e-mail: zaborekklaudia@gmail.com

Possibility analysis of applying serious game to learn the first aid procedures

Klaudia Zaborek*, Małgorzata Plechawska-Wójcik
Department of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract. The purpose of this work is to analyze the possibility of using serious games to learn the first aid procedures. This will be done based on research results. The scope of the work includes designing and implementing the game as well as research and analysis of results. The subject of research is to check whether the use of a serious game allows players to acquire and consolidate knowledge in the field of first aid.

Keywords: serious game; Unity; first aid

*Corresponding author.

E-mail address: zaborekklaudia@gmail.com

1. Wstęp

Serious Game jest to szybko rozwijający się gatunek gier. Są to gry, których głównym celem nie jest rozrywka. Przymiotnik „poważny'' wskazuje, że zostały zaprojektowane w celu zdobycia lub poszerzenia zasobu wiedzy gracza w jakiejś konkretnej dziedzinie. Znajdują zastosowanie w różnych branżach jako symulatory, które pozwalają przećwiczyć i utrwalić umiejętności.

Tematyka pierwszej pomocy jest to wiedza, którą każdy posiada w jakimś zakresie. Poprawne udzielenie pierwszej pomocy zapewnia więcej czasu dla poszkodowanego w celu ochrony życia lub zdrowia do przyjazdu specjalistycznej pomocy. Jest to główny powód dlaczego warto upowszechniać zasady postępowania podczas udzielania pierwszej pomocy.

Problem pojawia się, kiedy ludzie w przypadku spotkania osoby nieprzytomnej decydują się nie udzielać pierwszej pomocy. Jako pretekst stawiają strach przed zaszkodzeniu poszkodowanemu, nadzieją, że pomocy udzieli ktoś inny, bądź obawą przed odpowiedzialnością karną. W takim przypadku tym bardziej należy zaznajamiać ludzi z odpowiednimi schematami postępowania w nagłych wypadkach.

Próbą rozwiązania tego problemu może być gra typu Serious Game, której celem rozgrywki jest rozwijanie i utrwalanie umiejętności z konkretnej dziedziny. W przypadku pierwszej pomocy jest to pomoc w budowaniu właściwych odruchów w sytuacjach kryzysowych.

2. Przegląd literatury

Gry typu serious game (w tłumaczeniu; „poważne gry") w kontekście gier komputerowych wydają się być nowym zjawiskiem. Temat ten w literaturze naukowej pojawił się dopiero w drugiej połowie dwudziestego wieku. Początkowo termin ten jednak dotyczył gier karcianych i planszowych, a dopiero z czasem zaczął on być używany do gier komputerowych.

2.1. Pojęcie „poważna gra''

W literaturze pojawia się kilka podejść do zdefiniowania terminu poważnej gry, w znaczeniu, którego używamy w dniu dzisiejszym.

Pierwszą formalną próbą zdefiniowania pojęcia wydaje się być wprowadzona przez Clark C. Abt [1] w wydanej przez niego książce w 1970 roku. Abt w swojej książce porównał zwykłe gry, których głównym celem jest rozrywka, do gier wyspecjalizowanych (poważnych), których celem jest szkolenie i edukacja. Przedstawił symulacje i gry jako element, który może poprawić edukację zarówno w szkole jak i poza nią – tutaj jako przykład podał grę militarną, której zadaniem było szkolenie oficerów wojskowych. Inne

przykłady, którymi się posłużył, są to gry oparte na „piórze i papierze'' – głównie gry karciane i planszowe – branża gier komputerowych wtedy jeszcze się nie wykształtowała. Abt zdefiniował pojęcie poważnej gry jako „zredukowana do jej formalnej istoty gra to aktywność pomiędzy dwoma lub więcej niezależnymi decydentami dążącymi do osiągnięcia swoich celów w jakimś ograniczonym kontekście. [...] Do czynienia z serious games mamy wtedy gdy sensem tych gier jest wyraźnie i starannie przemyślany cel edukacyjny i nie są przeznaczone do grania w celach rozrywkowych." [1]. Mówiąc prościej poważna gra to taka gra, której wymiar rozrywkowy nie jest głównym celem.

Następną osobą, która podjęła się tematu jakim są poważne gry był Michael Zyda [2], który nieco rozwinął definicje. Według autora gra jest to: „fizyczna lub psychiczna rywalizacja tworzona według określonych zasad, której celem jest zabawianie i nagradzanie uczestnika", a gra wideo: „rywalizacja o podłożu psychicznym, której celem jest gra z komputerem rozgrywana według pewnych zasad; służy ona zabawie, rekreacji, lub wygrywaniu nagród". Zyda zdefiniował pojęcie poważnej gry jako „rywalizacja umysłowa, rozgrywana z komputerem, w oparciu o określone zasady, która wykorzystuje rozrywkę, aby promować i osiągać rządowe lub korporacyjne cele związane z treningiem umiejętności, edukacją, zdrowiem, polityką publiczną i komunikacją strategiczną.". Kwestia rozrywki, choć nadal nie jest pierwszoplanowa, zyskała z czasem nieco na znaczeniu – u Abta występowała jako korzyść uboczna, u Zydy jest już narzędziem.

Podsumowując, poważna gra to taka gra, które za główny cel nie przyjmuje rozrywki, jednak nie oznacza, że będzie jej pozbawiona. Przedstawiają poważne problemy, stymulując przy tym rzeczywistość i sprawdzają, jak dana osoba zachowałaby się w konkretnej sytuacji. Mają przedstawiać praktyczne zastosowania i służą jakiemuś celowi.

2.2. Zastosowanie

Istnieją publikacje, które przedstawiają konkretne przykłady zastosowania poważnych gier. Wiele z nich dotyczy możliwości użycia ich w medycynie. Mogą być one skierowane do różnych grup społecznych. Przykładem jest gra edukacyjna dla dzieci i młodzieży, której celem jest wyćwiczenie nawyku zdrowego odżywiania [3], a dla osób w podeszłym wieku gra zaprojektowana z myślą o poprawie zdrowia fizycznego i psychicznego [4]. Poważne gry mogą również koncentrować się wokół rozwiązania konkretnego problemu zdrowotnego. Przykładem są różnego rodzaju gry zaprojektowane specjalnie do celów terapeutycznych i uzupełnienia psychoterapii [5], albo leczenia depresji [6]. W kategorii poważnych gier zdrowotnych znajdują się również rozwiązania stworzone z myślą o personelu medycznym, które można wykorzystać do celów edukacyjnych, szkoleniowych lub treningowych.

Przykładem gier stworzonych z myślą o personelu medycznym są to symulacje sal operacyjnych. Przegląd „Systematic review of serious games for medical education and surgical skills training'' [7] analizuje wiele poważnych gier z tej kategorii. Podzielono je na dwie kategorie: te opracowane dla konkretnych celów edukacyjnych oraz gry komercyjne przydatne również do rozwijania umiejętności istotnych dla personelu medycznego. Niektóre skupiają się na odkrywaniu przyczyn występowania niepowodzeń mającą związek z niewłaściwą komunikacją między chirurgami, pielęgniarkami oraz pacjentami. Drugie z kolei skupiają się na tworzeniu symulacji do szkolenia przyszłych chirurgów, których celem jest zmniejszenie występowania błędu medycznego. Przegląd zaznacza, że żadna z wymienionych tam gier nie przeszła pełnego procesu walidacji w celu użycia. Nie przekreślają jednak możliwości stosowania gier do szkolenia technicznych umiejętności chirurgicznych, ale wymagają one większej walidacji przed włączeniem do programów nauczania.

2.3. Inne metody nauczania

Istnieje wiele innych metod nauczania pierwszej pomocy [8]. Można je podzielić na teoretyczne oraz praktyczne.

Jedną z nich jest wykład, w którym dla grupy ludzi przeprowadza się teoretyczną prelekcję na temat zasad udzielania pierwszej pomocy. Metoda nie należy do idealnej, ponieważ brakuje tutaj praktycznego przećwiczenia umiejętności.

Inną metodą jest kurs w postaci pokazu, który polega na zademonstrowaniu oraz omówieniu czynności przez instruktora bądź w prezentacji multimedialnej lub filmie. Taki sposób nauki wydaje się być atrakcyjniejszą formą przekazu, jednak nadal brakuje w nim ćwiczeń praktycznych.

Kolejną metodą jest kurs w postaci symulacji, w którym uczestnicy mogą przećwiczyć praktyczne umiejętności pierwszej pomocy na manekinie. Sposób ten oferuje przekazanie wiedzy zarówno w sposób teoretyczny jak i praktyczny. Realistyczne symulacje wpływają na lepsze jej zapamiętanie przez uczestnika.

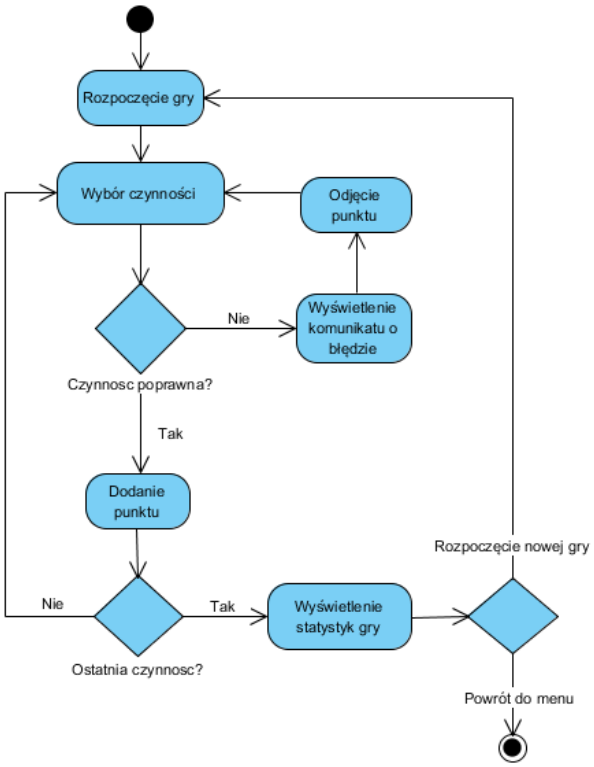
Metodą, która sprawdza się podczas nauki zasad u dzieci jest inscenizacja. Realizowana jest w formie zabawy, w której to dzieci wcielają się w rolę osób poszkodowanych lub ratujących. Odtwarzanie zachowań powoduje przeżycie emocji, a tym samym powoduje lepsze zapamiętywanie ćwiczeń.

3. Projekt gry

Zadaniem gry jest nauczenie potencjalnych graczy podstawowych zasad udzielania pierwszej pomocy. Platformą docelową jest komputer z systemem operacyjnym Windows.

Rozgrywka przedstawia nieprzytomną osobę leżącą na ziemi w parku. Zadaniem gracza jest udzielenie jej pierwszej pomocy. Zależnie od wybranego scenariusza, czynnością tą będzie ułożenie poszkodowanego w pozycji bocznej bezpiecznej lub przeprowadzenie resuscytacji krążeniowo-oddechowej (RKO).

Podczas rozgrywki na ekranie w formie tekstowej są wyświetlane instrukcje, które gracz powinien wykonać. Samą rozgrywkę można podzielić na segmenty, w których to gracz musi podjąć jakąś decyzję. Taką decyzją może być odpowiedź na pytanie, bądź wybranie odpowiedniego obiektu z sceny gry. Każdy segment posiada tylko jedną poprawną odpowiedź oraz kilka błędnych. Poprawne decyzje powodują przyznanie punktu oraz postęp rozgrywki, a błędne odejmują punkt i wymuszają wybór innej odpowiedzi w danym kroku. Gracz na koniec dostaje podsumowanie z wynikiem i informacją o liczbie popełnionych błędów wraz z ich opisem. Do zobrazowania przebiegu rozgrywki stworzono diagram aktywności. Przedstawia go rysunek 1.



Rys. 1. Diagram aktywności – przebieg rozgrywki

Budowa gry opiera się głównie na GameObjects [9]. Są to podstawowe obiekty w Unity [10], które reprezentują postać bądź scenerię. Działają jako pojemnik na komponenty, które wdrażają zachowanie w grze.

Rozgrywkę gry można podzielić na trzy elementy: sprawdzenie stanu poszkodowanego, sprowadzenie pomocy oraz zależnie od trybu ułożenie poszkodowanego w pozycji bocznej lub przeprowadzenie RKO. Każdy z tych elementów jest pojemnikiem GameObjects, który składa się z kolejnych GameObjects - segmentów. Domyślnie wszystkie segmenty są nieaktywne oprócz odpowiadającego za rozpoczęcie gry. Fragment struktury gry przedstawia rysunek 2, który obrazuje budowę pozycji bocznej z wszystkimi jej obiektami.

Rys. 2. Segmenty pozycji bocznej

4. Realizacja badań

Celem pracy jest analiza wpływu gry typu poważna gra na proces nauki zasad udzielania pierwszej pomocy. Zrealizowano to w oparciu o wyniki badań, które przeprowadzono w ramach pracy. Analiza wyników ma za zadanie udowodnić poprawność tezy, że gra z gatunku poważna gra może pomóc w zdobyciu lub polepszeniu wiedzy z zakresu pierwszej pomocy.

Do przeprowadzenia badań zastosowano metodę CAWI (Computer Assisted Web Interviews). Polega ona na przeprowadzeniu wywiadu, w której respondent wypełnia ankietę online. Zebrane dane są w formie elektronicznej co zapewnia szybki do nich dostęp. Umożliwia to również podgląd wyników badań w trakcie ich trwania. Obecnie praktycznie każdy ma dostęp do Internetu dzięki czemu metodą CAWI łatwiej dotrzeć do wybranej grupy badawczej.

Badania zostały przeprowadzone z zastosowaniem anonimowej ankiety. Osoby biorące udział w badaniu zostały poproszone o przejście obu scenariuszy w autorskiej grze, a następnie o wypełnienie ankiety na jej temat. Ankieta składa się z 20 pytań.

Treść ankiety można podzielić na trzy kategorie:

- pytania określające respondenta,
- pytania badające przekazaną wiedzę,
- pytania badające interfejs użytkownika.

Pytania określające respondenta mają za zadanie określić wiek, płeć, wykształcenie oraz doświadczenie w udzielaniu pierwszej pomocy.

Zadaniem kolejnej grupy pytań jest sprawdzenie wiedzy nabytej w grze. Znajdują się tutaj pytania o subiektywną opinie respondenta na temat jego wiedzy oraz pytania kontrolne. Przykładowym pytaniem kontrolnym jest ile uciśnięć klatki należy wykonać podczas resuscytacji krążeniowo-oddechowej. W tej grupie pojawiła się również pytanie sprawdzające czas spędzony w grze.

Ostatnią grupą są pytania sprawdzające wykonanie graficznego interfejsu użytkownika (GUI) - nawigowanie, intuicyjność, zrozumienie pojawiających się komunikatów oraz czytelność.

Do analizy interfejsu zastosowano wędrówkę poznawczą [11]. Jest to metoda oceny użyteczności interfejsu. Kładzie nacisk na badaniu płynności procesu zamiast oceny poszczególnych ekranów interfejsu. Analityk podczas badania rozważa i ocenia zagadnienia typu: czy gracz dostrzega elementy interfejsu, z którymi może wejść w interakcję; czy poprawnie połączy akcję z elementem otoczenia.

Założeniem dla grupy badawczej jest, że w jej skład wchodzą osoby niezwiązane z medycyną. Osoby te posiadają małe lub żadne doświadczenie w udzielaniu pierwszej pomocy. Docelowo grupa ma się składać z 30 osób.

Do przeprowadzenia ankiety skorzystano z darmowego formularza Google. Pozwala on zarządzać ankietą poprzez przeglądarkę internetową. Oferuje stworzenie pytań z różnymi typami odpowiedzi - jedno lub wielokrotnego wyboru. Formularze Google oferują przedstawienie wyników w formie graficznej oraz umożliwiają wyeksportowanie odpowiedzi do Excel.

5. Wyniki badań

5.1. Grupa badawcza

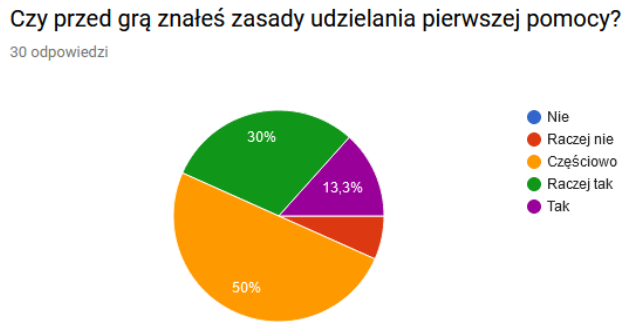
Badanie zostało przeprowadzone na 30 osobach. Średnia wieku respondentów wynosi 24 lat. Najmłodsza osoba miała 19 lat, a najstarsza 33. Ze względu na płeć to większość respondentów stanowili mężczyźni, bo aż 70%. Spośród ankietowanych tylko jedna osoba posiada wykształcenie niższe niż średnie, a większość ankietowanych posiada wykształcenie wyższe - 46,7%.

Z osób biorących udział w badaniu 40% zadeklarowało brak jakiegokolwiek doświadczenia w udzielaniu pierwszej pomocy. Pozostała część respondentów posiada przynajmniej małe doświadczenie.

Rys. 3. Doświadczenie respondentów w udzielaniu pierwszej pomocy

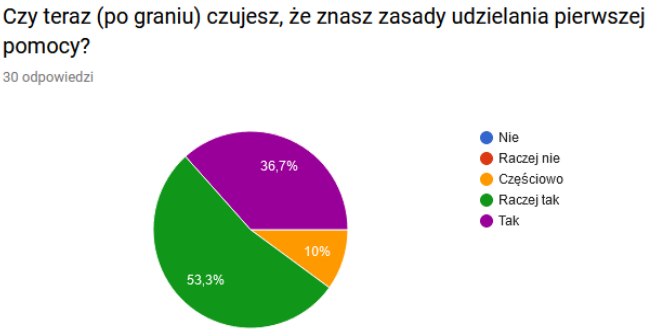
5.2. Wartość edukacyjna

Większość respondentów przyznała, że przed zagraniem w grę znała przynajmniej częściowo zasady udzielania pierwszej pomocy, stanowią oni aż 50% całości (rys. 4). Jedynie 6,7% odpowiedziała „Raczej nie". Pozostała część ankietowanych stwierdziła, że „raczej znają'' bądź „znają'' zasady udzielania pierwszej pomocy.



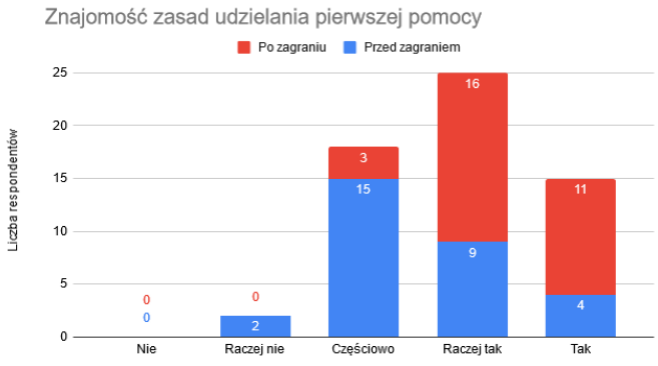
Rys. 4. Znajomość zasad udzielania pierwszej pomocy - przed zagraniem

Przed zagraniem w grę większość respondentów znała przynajmniej częściowo zasady udzielania pierwszej pomocy, stanowią oni aż 50% całości (rys. 5). Jedynie 6,7% odpowiedziała „Raczej nie". Pozostała część ankietowanych stwierdziła, że „raczej znają'' bądź „znają'' zasady udzielania pierwszej pomocy.



Rys. 5. Znajomość zasad udzielania pierwszej pomocy - po zagraniu

Na to samo pytanie po zagraniu w grę już większość ankietowanych zadeklarowała że „raczej zna'' bądź „zna'' zasady udzielania pierwszej pomocy (rys. 6). Jedynie 10% ankietowanych po zagraniu czuje, że nadal tylko częściowo zna zasady. Nikt z respondentów nie odpowiedział na to pytanie negatywne. Oznacza to, że według opinii respondentów gra pozytywnie wpłynęła na poprawę ich wiedzy w tym zakresie.



Rys. 6. Znajomość zasad udzielania pierwszej pomocy – przed oraz po zagraniu

Przeprowadzony został również test T-Studenta dla prób zależnych przy użyciu wyżej podanych danych. Dla poziomu

istotności alpha = 0.05 wartość p wynosi .000454. Oznacza to, że wynik jest istotny przy p <0,05.

Ocena znajomości zasad udzielania pierwszej pomocy jest tylko subiektywną opinią respondentów. Na potrzeby badań zadano respondentom pięć pytań sprawdzających wiedzę przekazaną w grze. Poprawna odpowiedź była warta jeden punkt, czyli łącznie można było ich zdobyć pięć. Większość pytań zawierało 4 odpowiedzi, w tym tylko jedną poprawną - wyjątkiem było pytanie o kolejność czynności przy przeprowadzaniu RKO.

Pierwszym pytaniem było podanie numeru alarmowego. Wszyscy ankietowani odpowiedzieli na to pytanie poprawnie.

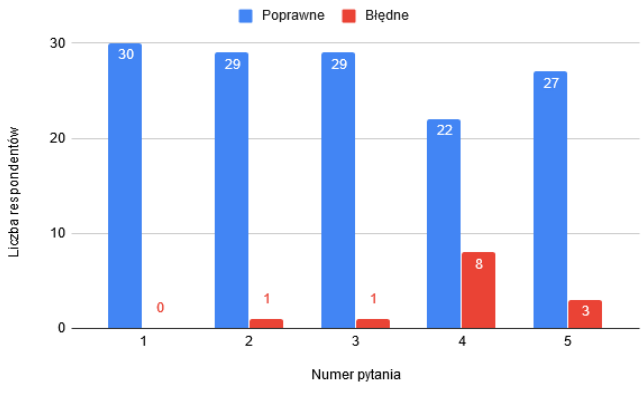
Kolejnym pytaniem było podanie liczby uciśnięć klatki piersiowej jaką należy wykonać podczas RKO. Tylko jedna osoba odpowiedziała na to pytanie błędnie.

Następnym pytaniem było podanie liczby oddechów ratunkowych, którą należy wykonać podczas RKO. Tutaj również tylko jedna osoba odpowiedziała błędnie. Była to inna osoba niż w poprzednim pytaniu.

Czwartym pytaniem było podanie w jakiej kolejności należy wykonać czynności podczas RKO. Jest to pytanie, które sprawiło najwięcej problemu respondentom. Błędnie odpowiedziało na nie 9 ankietowanych, którzy stanowi 26,7% całości.

Ostatnim pytaniem było podanie kolejności czynności jakich należy wykonać, aby ułożyć poszkodowanego w pozycji bocznej bezpiecznej. Na to pytanie odpowiedziały 3 osoby błędnie, którzy stanowią 10% całości.

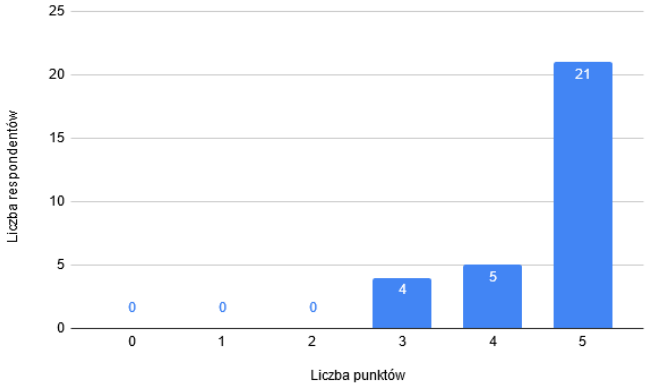
Rysunek 7 przedstawia stosunek błędnych i poprawnych odpowiedzi dla każdego pytania.



Rys. 7. Wykres przedstawiający stosunek błędnych i poprawnych odpowiedzi

Większość ankietowanych odpowiedziała poprawnie na wszystkie pytania, stanowią oni aż 70% całości (rys. 8). Średnia zdobytych punktów wynosi 4,56. Nikt z respondentów nie odpowiedział na wszystkie pytania błędnie. Każdy z ankietowanych odpowiedział co najmniej na trzy pytania prawidłowo. Korelacja między liczbą punktów, a subiektywną opinią na temat wiedzy posiadanej po zagraniu wynosi 0,61 co wskazuje na umiarkowaną zależność. Jest ona dodatnia co oznacza, że wraz ze wzrostem jednej danej rośnie również druga.

Dane przedstawiają stan wiedzy uczestników na temat pierwszej pomocy w pozytywnym świetle. Wyniki przedstawiają jednak stan wiedzy respondentów jedynie po zagraniu. Nie można przez to stwierdzić, czy gra wpłynęła na zwiększenie wiedzy. W celu zbadania tego należałoby przeprowadzić kolejne badania, w których również sprawdzany jest stan wiedzy uczestników przed zagraniem w grę.



Rys. 8. Wykres przedstawiający liczbę zdobytych punktów przez respondentów

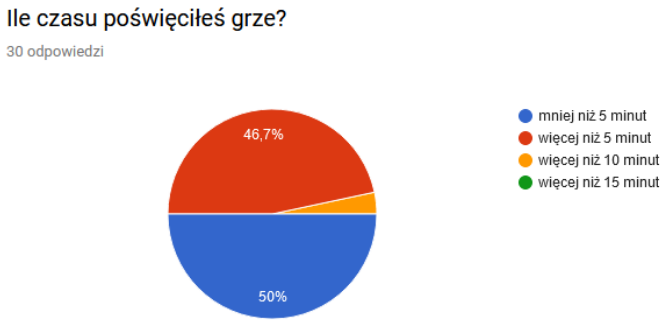
Respondenci zostali również zapytani o czas poświęcony grze. Połowa ankietowanych spędziła mniej niż 5 minut na graniu, a druga połowa więcej niż 5 minut, w tym jedna osoba zadeklarowała, że spędziła więcej niż 10 minut. Czas spędzony w grze przedstawia rysunek 8. Na podstawie powyższych wyników można założyć, że ok. 5 minutowa rozgrywka jest wystarczająca do przyswojenia wiedzy. Z drugiej strony korelacja między czasem spędzonym w grze, a liczbą zdobytych punktów wynosi 0,04, co wskazuje na brak związku między danymi.

Gra została oceniona przez większość respondentów za przydatną, bądź bardzo przydatną co stanowi aż 83,3% (rys. 10). Średnia wynosi 4,1 co odpowiada ocenie „przydatna''. Jedna osoba określiła grę za „raczej nieprzydatną'' co stanowi 3,3% całości. W oparciu o głosy graczy można powiedzieć, że gra daje możliwości edukacyjne.

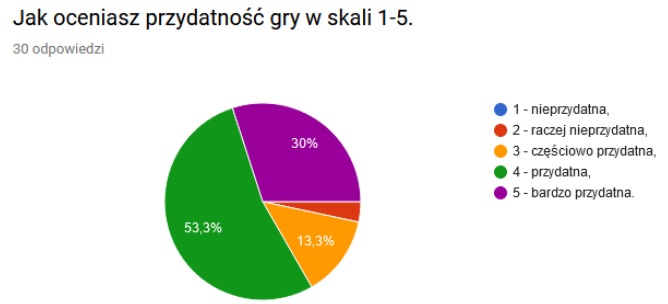
Respondentom zadano także pytanie dotyczące wpływu gry na ich umiejętność poprawnego udzielenia pierwszej pomocy. Większość ankietowanych zadeklarowała pozytywnie - odpowiedzi „raczej tak'' i „tak'' stanowią 80% całości (rys. 11). Tylko dwie osoby odpowiedziały „raczej nie''.

Korelacja między liczbą zdobytych punktów, a wpływie gry na umiejętności respondentów wynosi 0,51. Zależność między danymi jest dość istotna. Między wcześniejszym doświadczeniem, a wpływem gry korelacja wynosi już tylko 0,13 co pokazuje praktycznie brak związku między danymi.

Wskazuje to, że gra wpływa w pozytywny sposób na pewność zdobytych umiejętności bądź ich utrwalenie.

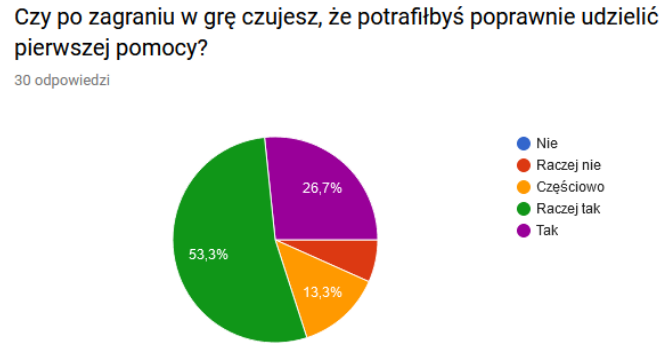


Rys. 9. Czas spędzony w grze



Rys. 10. Ocena przydatności

Zdecydowana większość ankietowanych zadeklarowała chęć ponownego zagrania w grę w przypadku dodania nowych scenariuszy, bo aż 93,3% (rys. 12). Jedynie dwie osoby odpowiedziały „nie''. Pokazuje to, że taka forma (poważna gra) przyswajania informacji cieszy się dużym zainteresowaniem.



Rys. 11. Wpływ gry na umiejętności udzielania pierwszej pomocy

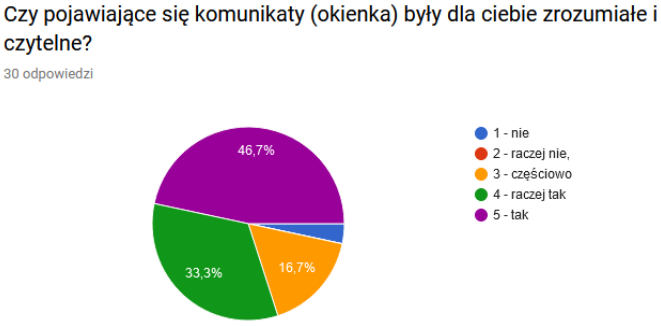
5.3. Wykonanie gry

Większa część ankietowanych określiła nawigowanie po grze jako łatwe i intuicyjne - odpowiedzi „raczej tak'' oraz „tak'' stanowią 53,3% całości. Dla 10% ankietowanych nawigowanie po grze sprawiało trudności.

Większość ankietowanych określiła komunikaty zrozumiałe oraz czytelne, bo dla aż 80%. Jedynie jedna osoba odpowiedziała przecząco. Dla wszystkich respondentów nazewnictwo użyte w grze było zrozumiałe. Wyniki przedstawiają rysunki 14 i 15.

Rys. 12. Zainteresowanie dodatkowymi scenariuszami

Rys. 13. Ocena nawigowania po grze



Rys. 14. Ocena komunikatów

Rys. 15. Ocena nazewnictwa

Respondentom zostało także zadane pytanie czy w grze napotkali moment, w którym nie wiedzieli co następnie należy

wykonać (rys. 16). Twierdząco na to pytanie odpowiedziało 30% ankietowanych.

Rys. 16. Zainteresowanie dodatkowymi scenariuszami

Respondenci mieli możliwość napisania swoich uwag dotyczących gry. Kilku z nich skorzystało z takiej okazji. Większość uwag dotyczyła lepszego oznaczenia obiektów, z którymi można wejść z interakcję. Propozycjami rozwiązania problemu było oprócz podpisania również podświetlenie elementu po najechaniu na niego myszką. Drugą powtarzającą się uwagą są mało widoczne napisy w niektórych momentach.

5.4. Wnioski

Na podstawie zebranych danych z ankiety można stwierdzić, że gra posiada potencjał edukacyjny. Gra pozwoliła uczestnikom na zweryfikowanie swojej dotychczasowej wiedzy. Subiektywna opinia respondentów określająca ich poziom wiedzy z zakresu pierwszej pomocy uległa poprawie po zagraniu w grę. Pytania kontrolne przedstawiły stan wiedzy ankietowanych w pozytywnym świetle. Udowadnia to tezę, że gra wpłynęła na zdobycie wiedzy tylko częściowo. W tym celu konieczne byłoby przeprowadzenie kolejnych badań, w których należałoby sprawdzić wiedzę respondentów zarówno przed, jak i po zagraniu w grę.

Ocena przydatności gry oraz chęć zagrania w inne scenariusze pokazują, że poważna gra jako forma przekazywania wiedzy cieszy się dużym zainteresowaniem. Według ankietowanych wpłynęła ona również w pozytywny sposób na ich pewność co do posiadanych umiejętności z tego zakresu. Potwierdza to, że pomysł wykonania takiej gry okazał się słuszny.

Badania wykazały również kilka elementów koniecznych do poprawy. Chodzi tutaj przede wszystkim o przedstawienie graficzne gry. Część respondentów zgłosiła problemy z nawigowaniem po grze oraz czytelnością czcionki. Są to jednak błędy wyłącznie kosmetyczne.

Rozwiązaniem czytelności może być zwiększenie kontrastu pomiędzy tłem, a czcionką. Kolejnym krokiem może być również zwiększenie rozmiaru czcionki. Wyskakujące komunikaty również niektórym respondentom sprawiły problem z ich odczytaniem. Jeden z respondentów wpisał w uwagach, że kliknięcie w komunikat, aby go zamknąć nie jest intuicyjne. Dodanie symbolu „x'', który w większości programów komputerowych spełnia rolę zamknięcia powinno rozwiązać problem. Rysunek 7.18 przedstawia propozycję poprawy.

Metoda wędrówki poznawczej wykazała, że respondenci największy problem mieli z znalezieniem obiektów, z którymi można wejść w interakcję. Sam podpis po najechaniu na nie myszką był niewystarczający. Najlepszym rozwiązaniem byłoby, aby wszystkie elementy do interakcji odznaczały od reszty bez konieczności ich zaznaczenia. Wprowadzić to można za pomocą zmiany koloru (przykład - rys. 17), delikatnego efektu poświaty, ewentualnie wprowadzić podpisanie elementów na stałe, a nie tylko po oznaczeniu ich kursorem. Takie rozwiązanie wiązałoby się z koniecznością dodania więcej elementów błędnych, aby zachować poziom trudności gry.

Inną propozycją poprawy nawigowania po grze jest zmiana oznaczenia obiektu, na którym znajduje się kursor. Dodatkowo do podpisu można otoczyć cały element poświatą, aby gracz dokładnie widział, który obiekt wybiera. Efekt ten można również zastosować do wyróżnienia wszystkich elementów możliwych do interakcji. Kolor poświaty mógłby posłużyć do odróżnienia stanu obiektu np. zielony - dostępny do interakcji, szary - w tym momencie nie można z nim wejść w interakcję, żółty - aktualnie na nim znajduje się kursor myszki.

Rys. 17. Scena z zmienionymi czcionkami

Rys. 18. Poprawa widoczności obiektów za pomocą zmiany koloru

Istnieją inne potencjalne formy nauki pierwszej pomocy. Dzielą się one na teoretyczne oraz praktyczne. Teoretyczne

skupiają się na przekazaniu teorii. Brakuje w nich jednak możliwości przećwiczenia zdobytych umiejętności. W praktycznych kładziony jest większy nacisk na ćwiczenia, ale jednocześnie jest przekazywana wiedza teoretyczna.

Literatura

[1] Abt, C. (1970). Serious Games. Viking Press, USA.

[2] Zyda, M. (2005). From visual simulation to virtual reality to games. Computer, 38(9), 25-32.

[3] Anderson, D. M. (2002, November). Hungry Red Planet: Children's multimedia nutrition simulation software. In The 130th Annual Meeting of APHA.

[4] Kang, K. K., Kim, J. A., & Kim, D. (2009). Development of a sensory gate–ball game system for the aged people. The Visual Computer, 25(12), 1073.

[5] Horne-Moyer, H. L., Moyer, B. H., Messer, D. C., & Messer, E. S. (2014). The use of electronic games in therapy: a review with clinical implications. Current psychiatry reports, 16(12), 520.

[6] Fleming T., Cheek C., Merry S., Thabrew H., Bridgman H., et al. (2014). Serious games for the treatment or prevention of depression: A systematic review. Journal of Psychopathology and Clinical Psychology, vol. 19, no 3, 227-242.

[7] Graafland, M., Schraagen, J. M., & Schijven, M. P. (2012). Systematic review of serious games for medical education and surgical skills training. British journal of surgery, 99(10), 1322-1330.

[8] Leszczyński, P. (2012). Metody nauczania resuscytacji. J Publ Health Nurs Med Rescue, 4, 50-55.

[9] https://docs.unity3d.com/Manual/class-GameObject.html [15.10.2019]

[10] Unity – Game Engine. https://unity3d.com/unity [15.10.2019]

[11] Borys Plechawska-Wójcik, M., & Plechawska-Wójcik, M. (2013). Badanie użyteczności oraz dostępności interfejsu w aplikacjach mobilnych. Nierówności społeczne a wzrost gospodarczy, (35), 63-78.

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 14 (2020) 73-77
Wysłane: 2019-10-14
Przyjęte: 2019-10-23

Porównanie wydajności narzędzi do tworzenia interfejsu aplikacji typu SPA na przykładzie React i Vue.js

Krzysztof Boczkowski*, Beata Pańczyk
Politechnika Lubelska, Katedra Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. W artykule przeprowadzono analizę wydajności aktualnie stosowanych narzędzi do tworzenia interfejsu aplikacji typu SPA. Badanie przeprowadzono z wykorzystaniem dwóch aplikacji, o takiej samej funkcjonalności, zaimplementowanych w React i Vue.js. Do pomiaru wykorzystano narzędzia deweloperskie dostępne w przeglądarkach internetowych oraz odpowiednie implementacje własnych metod.

Słowa kluczowe: React; Vue.js; wydajność; JavaScript

*Autor do korespondencji.

Adres e-mail: krzysztof.boczkowski@pollub.edu.pl

Comparison of the performance of tools for creating a SPA application interface - React and Vue.js

Krzysztof Boczkowski*, Beata Pańczyk
Department of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract. The article analyses the performance of currently used tools for creating a SPA application interface. The study was conducted using two applications with the same functionality, implemented in React and Vue.js. The tools available in web browsers and appropriate implementations of own methods were used to measure SPA performance.

Keywords: React; Vue.js; performance; JavaScript

*Corresponding author.

E-mail address: krzysztof.boczkowski@pollub.edu.pl

1. Wstęp

Obecnym trendem w budowaniu stron internetowych jest tworzenie aplikacji typu SPA (Single Page Application). SPA to aplikacja uruchamiana w przeglądarce internetowej, której ideą jest stworzenie interfejsu użytkownika o funkcjonalności zbliżonej do natywnych aplikacji systemowych [1].

Ten typ aplikacji charakteryzuje się wieloma właściwościami, dającymi przewagę nad zwykłą stroną internetową:

- aplikacja typu SPA jest wyświetlana jak zwykła aplikacja, lecz w przeglądarce aplikacje takie aktualizują widok dynamicznie, fragmentami. Każda akcja powoduje natychmiastową zmianę w wyświetlanym interfejsie. Jest to możliwe, ponieważ cała logika oraz elementy interfejsu są realizowane przez kod JavaScript [1].
- dużą zaletą jest szybsza komunikacja z serwerem - wszystkie zapytania do serwera potrzebują mniej czasu na oczekiwanie na odpowiedź. Pobierane są tylko potrzebne dane, zwykle w postaci obiektu JSON (JavaScript Object Notation) [1].

Wydajność, obok funkcjonalności, jest jedną z najważniejszych cech aplikacji. Na jej podstawie budowana jest satysfakcja użytkownika korzystającego z danego oprogramowania. Użytkownicy oczekują, że wczytywana strona wyświetli się nie dłużej niż po 2 sekundach. Jeśli wczytywanie trwa dłużej niż 3 sekundy to prawie 40% użytkowników rezygnuje z dalszego przeglądania witryny [2].

Przeglądarki internetowe podczas budowania i wyświetlania gotowego widoku aplikacji SPA realizują szereg procesów. Pierwszym z nich jest interpretacja języka JavaScript, który jest wykorzystywany do tworzenia interfejsu użytkownika. W kolejnych fazach są analizowane i obliczane style, budowany jest układ strony na bazie znaczników HTML, rysowany interfejs i na końcu ulega on komponowaniu [3].

Istnieje wiele narzędzi online umożliwiających zbadanie wydajności aplikacji internetowych m.in. PageSpeed Insights czy Webpagetest.org. Przeglądarki internetowe również posiadają wbudowane narzędzia do badania wydajności. Są to narzędzia deweloperskie pozwalające na wgląd w każdy element strony i zmierzenie czasów wykonywania poszczególnych procesów [4,5].

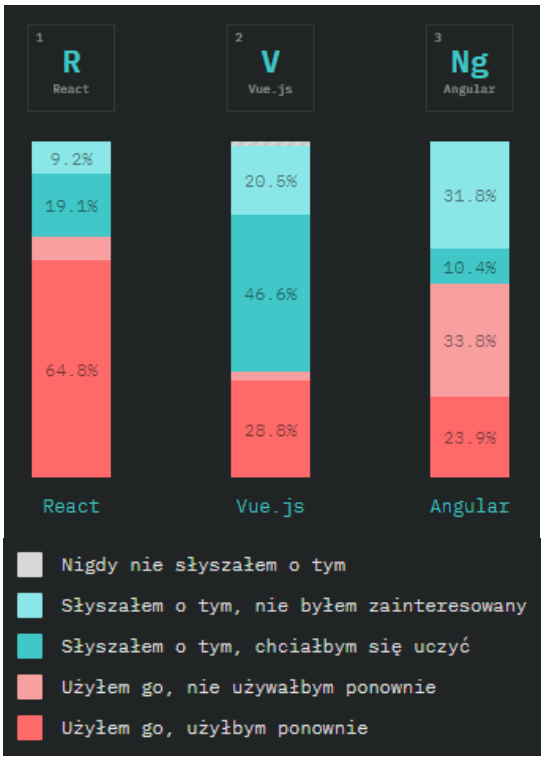
Język JavaScript posiada wbudowane narzędzia do badania wydajności kodu. Do nich zaliczamy m.in. interfejs ***performance*** dostarczający dostęp do informacji wydajnościowych wyświetlanej strony, czy interfejs ***console,*** który oprócz standardowej funkcji zapisu tekstu do konsoli

narzędzi deweloperskich umożliwia odczyt czasu pomiędzy wywołaniami metod ***time*** oraz ***timeEnd*** [6].

Celem badań przeprowadzonych w niniejszym artykule jest porównanie czasów wczytywania interfejsu użytkownika z poziomu kodu JavaScript aplikacji SPA tworzonej za pomocą biblioteki React i Vue.js, oraz wskazanie efektywniejszego rozwiązania.

2. Aktualne narzędzia do tworzenia aplikacji typu SPA

React i Vue.js należą do grona najpopularniejszych obecnie narzędzi do tworzenia interfejsów graficznych aplikacji typu SPA. Według badań przeprowadzonych dla potrzeb serwisu StateOfJs na grupie 20000 programistów z całego świata, w 2018 roku porównywane narzędzia zajęły dwa pierwsze miejsca (rys. 1) [7]. 64.8% respondentów chciałoby ponownie użyć React do budowy swoich aplikacji, w przypadku Vue było to 28.8% badanych.



Rys. 1. Wyniki badań satysfakcji programistów z 2018 roku [3]

2.1. React

React jest biblioteką do tworzenia interfejsów graficznych dynamicznych aplikacji internetowych, opartą o wirtualne drzewo DOM (Document Object Model). Za jego powstanie odpowiada Facebook, a dokładniej pracujący w nim programista Jordan Walke, który w 2012 napisał bibliotekę FaxJS - podwalinę pod React [8].

W React budowanie interfejsu opiera się o tworzenie komponentów wielokrotnego użytku, które są opisywane za pomocą JSX (JavaScript XML). To mieszanina kodu JavaScript oraz HTML (HyperText Markup Language), która pozwala na wykorzystywanie w kodzie znaczników HTML wraz z logiką. Każdy z komponentów posiada swój cykl życia składający się z etapu montowania, aktualizacji oraz usuwania. Każdy etap posiada odzwierciedlenie w odpowiednio nazwanych metodach, które umożliwiają reakcję na zmiany związane z cyklem życia komponentu [9].

2.2. Vue.js

Vue.js, podobnie jak React, jest narzędziem opartym o wirtualny DOM, do budowy warstwy wizualnej aplikacji. Jego twórcą jest Evan You, który jako pracownik Google wykorzystywał w pracy inne narzędzie - Angular. Zapragnął on stworzyć taki framework, który będzie wykorzystywał to co najlepsze w Angular oraz będzie jednocześnie łatwiejsze do wykorzystania i lżejsze. Konsekwencją tego było powstanie narzędzia Vue, opublikowanego w 2014 roku na GitHub [10].

Interfejs budowany jest tutaj z komponentów, które w swojej implementacji wykorzystują szablony HTML i logikę napisaną w JavaScript. Do przypisywania funkcjonalności i danych służą dyrektywy (podobnie jak w Angular). Analogicznie jak w React, każdy z komponentów posiada swój cykl życia i możliwa jest reakcja na jego zmianę [11].

3. Metoda badań

Do przeprowadzenia badań stworzono dwie aplikacje w języku z takim samym interfejsem użytkownika i funkcjonalnością, z wykorzystaniem opisanych w rozdziale 2 narzędzi języka JavaScript. W obu przypadkach zaimplementowano tabelę z możliwością zmiany liczby wyświetlanych wierszy. Dane do tabeli w postaci pliku JSON zostały wygenerowane za pomocą biblioteki faker.js jako tablica elementów (Przykład 1).

Przykład 1. Skrypt generujący dane testowe

```
const faker = require('faker');
const fs = require('fs');
const mockData = [];
for(let i=0; i<10000; i++){
  mockData.push({
    no: i + 1,
    firstName: faker.name.firstName(),
    lastName: faker.name.lastName(),
    hours: faker.random.number(24) + 160,
    jobTitle: faker.name.jobTitle(),
  });
}
fs.writeFile(
  `./public/report-data-mock.json`,
  JSON.stringify(mockData, null, 2),
  `utf-8`,
  (error) => {
    error
      ? console.log(error)
      : console.log(`generating finished`);
  }
);
```

Obie aplikacje uruchamiano w dwóch przeglądarkach internetowych:

- Google Chrome - v. 74.0.3729,
- Mozilla Firefox - v. 67.0.1.

Każda z przeglądarek posiada własny silnik renderowania, co może mieć wpływ na różnice w wydajności.

Testy wykonano według dwóch scenariuszy testowych:

- **Scenariusz 1**: czas renderowania komponentu tabeli dla 100, 500, 2500 i 10000 wierszy z wykonaniem pomiarów przy pomocy narzędzi deweloperskich przeglądarek. Mierzony jest czas od momentu wykrycia akcji kliknięcia myszy na przycisku nawigującym do ekranu z tabelą, do momentu zamontowania komponentu w drzewie DOM.
- **Scenariusz 2:** czas renderowania komponentu tabeli dla 100, 500, 2500 i 10000 wierszy z wykonaniem pomiarów za pomocą implementacji w metodzie cyklu życia komponentu. Ten scenariusz zakłada zbadanie wydajności przy pomocy obiektu ***console*** i jego metod: ***time*** oraz ***timeEnd*** (przykład 2 i 3).

Przykład 2. Implementacja testu wydajności w Vue

```
export default class Report extends Vue {
  constructor(props: any) {
    super(props);
    console.time("report screen render");
  }
  mounted() {
    console.timeEnd("report screen render");
  }
```

Przykład 3. Implementacja testu wydajności w komponencie React

```
class Report extends React.Component<any> {
  constructor(props: any) {
    super(props);
    console.time("report screen render");
  }
  componentDidMount() {
    console.timeEnd("report screen render");
  }
```

Podane metody obiektu ***console*** umożliwiają obliczenie czasu pomiędzy ich wywołaniami. Pozwalają na wprowadzenie identyfikatora do argumentu funkcji, który służy do wyznaczenia bloku kodu jaki będzie poddany pomiarom [12].

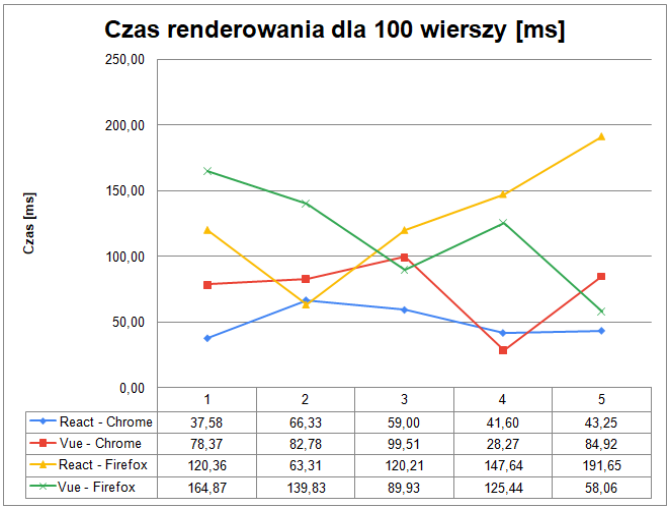
4. Platforma testowa

Badania wykonano na sprzęcie o specyfikacji:

- Procesor: Intel Core i7-7500u @ 3.5GHz, 4 rdzenie, 4 wątki;
- Pamięć: 8192MB LPDDR3 1866MHz;
- Dysk: SSD 256GB NVMe;
- System operacyjny: Antergos Linux (kernel wer. 5.1.6-arch1-1-ARCH);
- Ekran: 1920x1080;
- Układ graficzny: Intel Iris HD Graphics 620;
- Model: Lenovo Ideapad 710s-13ikb.

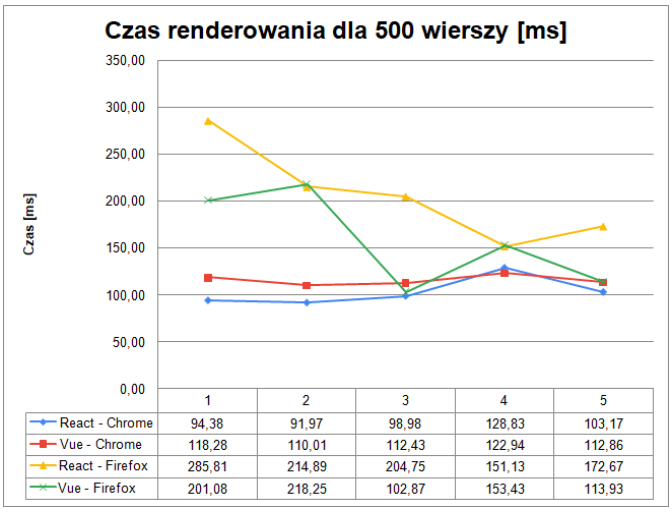
5. Wyniki badań

Wyniki testów wykonanych zgodnie z pierwszym scenariuszem przedstawiają rysunki 2-4. Rysunki 5-7 prezentują wyniki testów przeprowadzonych w drugim scenariuszu.



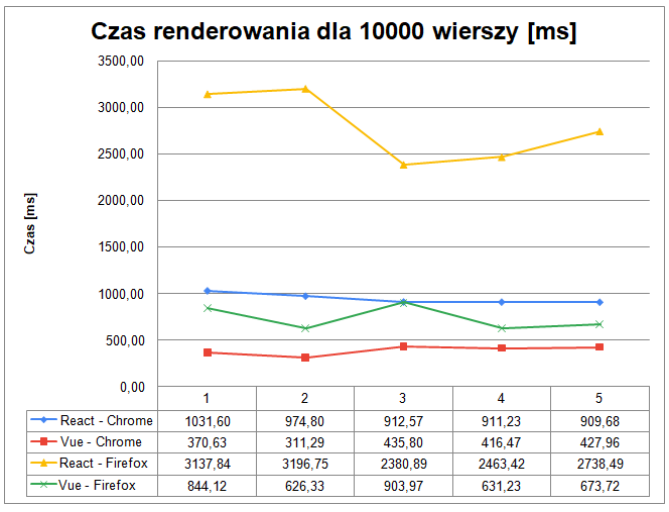
	1	2	3	4	5
React - Chrome	37,58	66,33	59,00	41,60	43,25
Vue - Chrome	78,37	82,78	99,51	28,27	84,92
React - Firefox	120,36	63,31	120,21	147,64	191,65
Vue - Firefox	164,87	139,83	89,93	125,44	58,06

Rys. 2. Wyniki pomiarów dla 100 wierszy (Scenariusz 1)



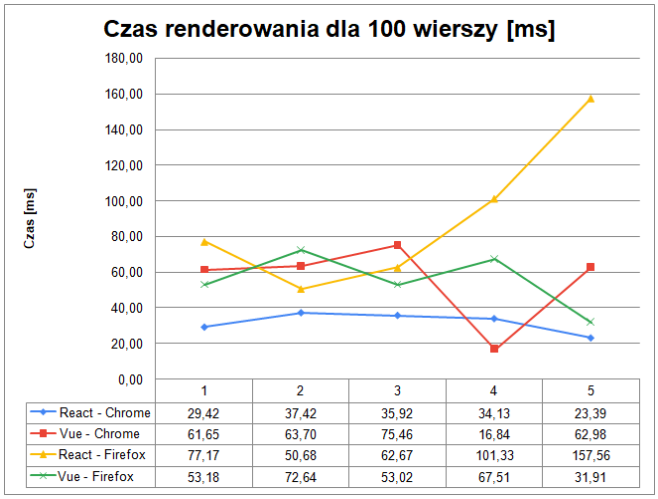
	1	2	3	4	5
React - Chrome	94,38	91,97	98,98	128,83	103,17
Vue - Chrome	118,28	110,01	112,43	122,94	112,86
React - Firefox	285,81	214,89	204,75	151,13	172,67
Vue - Firefox	201,08	218,25	102,87	153,43	113,93

Rys. 3. Wyniki pomiarów dla 500 wierszy (Scenariusz 1)



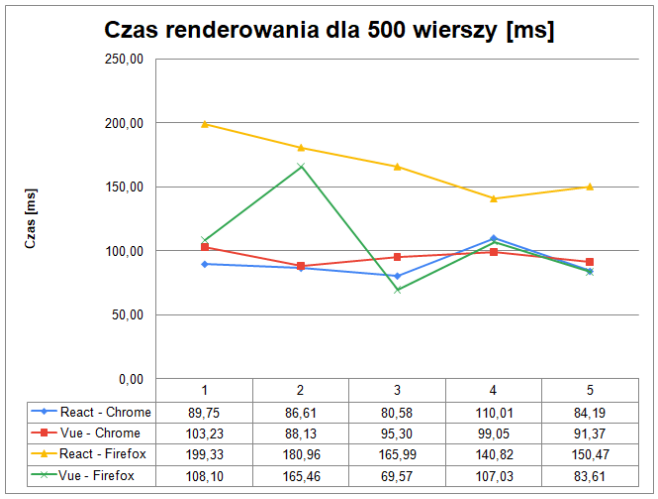
	1	2	3	4	5
React - Chrome	1031,60	974,80	912,57	911,23	909,68
Vue - Chrome	370,63	311,29	435,80	416,47	427,96
React - Firefox	3137,84	3196,75	2380,89	2463,42	2738,49
Vue - Firefox	844,12	626,33	903,97	631,23	673,72

Rys. 4. Wyniki pomiarów dla 100000 wierszy (Scenariusz 1)



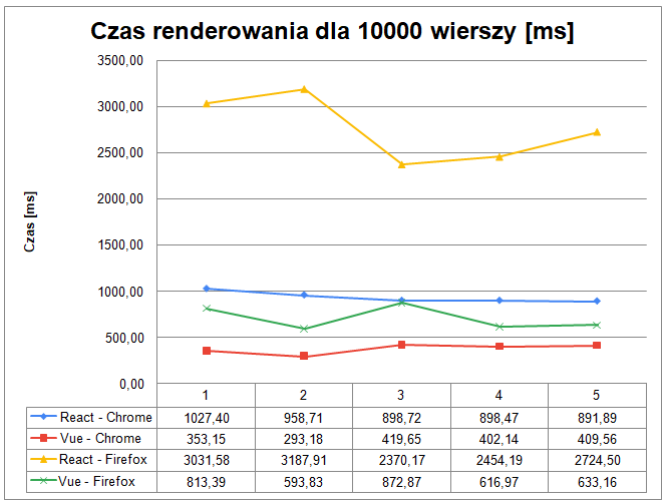
	1	2	3	4	5
React - Chrome	29,42	37,42	35,92	34,13	23,39
Vue - Chrome	61,65	63,70	75,46	16,84	62,98
React - Firefox	77,17	50,68	62,67	101,33	157,56
Vue - Firefox	53,18	72,64	53,02	67,51	31,91

Rys. 5. Wyniki pomiarów dla 100 wierszy (Scenariusz 2).



	1	2	3	4	5
React - Chrome	89,75	86,61	80,58	110,01	84,19
Vue - Chrome	103,23	88,13	95,30	99,05	91,37
React - Firefox	199,33	180,96	165,99	140,82	150,47
Vue - Firefox	108,10	165,46	69,57	107,03	83,61

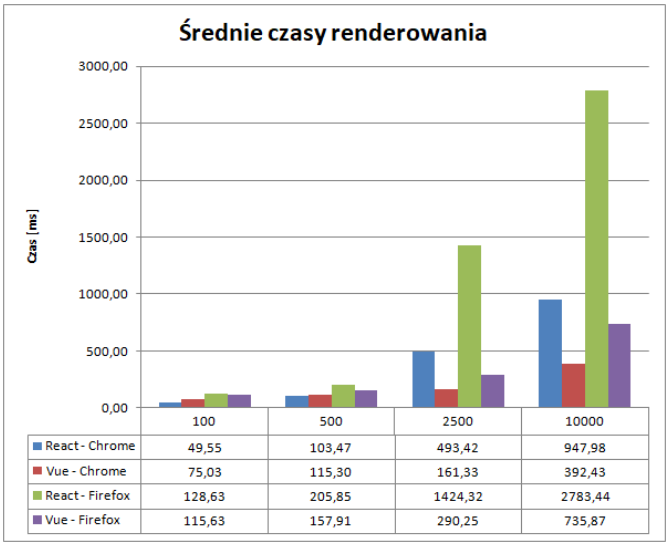
Rys. 6. Wyniki pomiarów dla 500 wierszy (Scenariusz 2).



	1	2	3	4	5
React - Chrome	1027,40	958,71	898,72	898,47	891,89
Vue - Chrome	353,15	293,18	419,65	402,14	409,56
React - Firefox	3031,58	3187,91	2370,17	2454,19	2724,50
Vue - Firefox	813,39	593,83	872,87	616,97	633,16

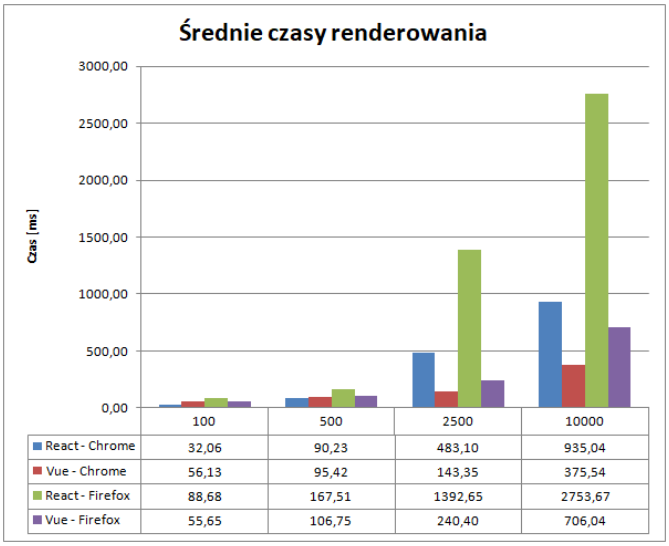
Rys. 7. Wyniki pomiarów dla 10000 wierszy (Scenariusz 2).

Średnie czasy uzyskane w wyniku pomiarów zestawiono na rysunkach 8 i 9.



	100	500	2500	10000
React - Chrome	49,55	103,47	493,42	947,98
Vue - Chrome	75,03	115,30	161,33	392,43
React - Firefox	128,63	205,85	1424,32	2783,44
Vue - Firefox	115,63	157,91	290,25	735,87

Rys. 8. Średnie czasy renderowania komponentu dla scenariusza 1.



	100	500	2500	10000
React - Chrome	32,06	90,23	483,10	935,04
Vue - Chrome	56,13	95,42	143,35	375,54
React - Firefox	88,68	167,51	1392,65	2753,67
Vue - Firefox	55,65	106,75	240,40	706,04

Rys. 9. Średnie czasy renderowania komponentu dla scenariusza 2.

6. Analiza wyników

Na podstawie uzyskanych wyników renderowania 100 wierszy nie można jednoznacznie stwierdzić, które z narzędzi w tym przypadku jest efektywniejsze. Zarówno pomiary za pomocą narzędzi deweloperskich w przeglądarce (Rys. 2), jak i te z wykorzystaniem metod cyklu życia komponentu (Rys. 5), posiadają duże odchylenia. Najlepiej widać to w przypadku przeglądarki Firefox, gdzie pierwsze pomiary dla aplikacji React są większe, niż w przypadku Vue. Bardziej wyrównane wyniki uzyskano w przeglądarce Chrome, w której lepszy okazuje się React. Należy jednak zauważyć, że na dość długi czas tworzenia widoku nie składa się jedynie montowanie i renderowanie testowanego komponentu. Różnica w czasach pomiędzy narzędziami deweloperskimi (gdzie mierzono czas od momentu wykrycia akcji myszki), a pomiarem na podstawie cyklu życia komponentu, jest zauważalna i stanowi dużą część procesu.

W przypadku tworzenia widoku z 500 wierszami, pomiary w Chrome są zbliżone (rysunek 3 i 6). W Firefox widać

większe różnice - Vue okazuje się wydajniejszy. Średni czas renderowania dla scenariusza 1 dla Vue wynosi 157,91 ms, a React potrzebuje już 205.85 ms na obsługę akcji kliknięcia myszą i zbudowanie widoku (Rys. 8). Czas renderowania komponentu w React, stanowi ponad 130% czasu w Vue. Stosunek ten zaczyna wzrastać w przypadku renderowania samego komponentu (Scenariusz 2) do 160% (Rys. 9).

Podczas renderowania 10000 wierszy jest już zauważalna znacząca różnica pomiędzy React i Vue. Widać też różnice pomiędzy przeglądarkami. Chrome wydaje się być stabilniejsza i lepiej zoptymalizowana. Pomiary dla 10000 wierszy charakteryzują się mniejszym procentowym błędem pomiarowym (Rys. 3,4,7,8) i wyraźniej widać różnicę w pomiarach.

7. Wnioski

Wyniki badań pokazują różnice w wydajności pomiędzy narzędziami w wyświetlaniu bardzo rozbudowanych interfejsów. React przewyższa nieznacznie wydajnością Vue w przypadku komponentów z małą liczbą wierszy. Ulega to jednak zmianie jeśli budowany widok ma większą liczbę elementów. Wtedy Vue.js jest wydajniejsze.

Różnice między przeglądarkami są wyraźne. Na podstawie wyników pomiarów aplikacji testowych, w przypadku Chrome silnik uruchomieniowy języka JavaScript jest wydajniejszy i jego kod jest bardziej optymalny w stosunku do jego odpowiednika w Firefox.

Aplikacje powstałe na potrzeby badań, dobrze pokazują, że duży wpływ na wydajność ma liczba prezentowanych danych na stronie, ale również przeglądarka, z której korzysta końcowy użytkownik. Dobrym zwyczajem programistów front-endu jest tworzenie widoków, które nie wyświetlają zbyt wielu informacji jednorazowo. Wówczas różnice w czasie renderowania komponentów nie będą zbyt wielkie w różnych przeglądarkach, niezależnie od tego, czy aplikacja SPA korzysta z React czy Vue.js.

Literatura

[1] E. Scott, SPA Design and Architecture: Understanding Single Page Web Applications, Manning Publications, 2015.

[2] J. Wagner, Web Performance in Action: Building Faster Web Pages, Manning Publications, 2017.

[3] https://developers.google.com/web/fundamentals/performance/rendering [20.10.2019]

[4] https://crystallize.com/blog/frontend-performance-measuring-and-kpis [20.10.2019]

[5] https://developers.google.com/web/tools/chrome-devtools/evaluate-performance/ [20.10.2019]

[6] https://ourcodeworld.com/articles/read/144/measuring-the-performance-of-a-function-with-javascript-using-browser-tools-or-creating-your-own-benchmark [20.10.2019]

[7] http://2018.stateofjs.com/front-end-frameworks/overview/ [08.10.2019]

[8] http://www.education-ecosystem.com/guides/programming/react-js/history [08.10.2019]

[9] http://pl.reactjs.org/docs/getting-started.html [08.10.2019]

[10] http://www.freecodecamp.org/news/between-the-wires-an-interview-with-vue-js-creator-evan-you-e383cbf57cc4/ [08.10.2019]

[11] http://vuejs.org/v2/guide/ [08.10.2019]

[12] Ch. Adams, Mastering JavaScript High Performance, Packt Publishing, 2015.

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 14 (2020) 78-81
Wysłane: 2019-10-17
Przyjęte: 2019-10-23

Analiza porównawcza technologii widoków dla aplikacji Spring

Vadym Borys*, Roman Slezenko*, Beata Pańczyk

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

Streszczenie. Celem artykułu jest wybór, możliwie najbardziej wydajnych narzędzi do tworzenia interfejsu użytkownika dla aplikacji Spring.W artykuleprzeprowadzono porównanie dla 4 wybranych technologii widoków: JSP, Thymeleaf, Wicket i Angular. Testy wydajności czasowej i pamięciowej zostały przeprowadzone z wykorzystaniem Rest API w Spring. Wyniki testów pozwoliły wskazaćnajlepsze rozwiązania.

Słowa kluczowe: wydajność; Spring; Angular; JSP; Thymeleaf; Wicket

*Autor do korespondencji.

Adres e-mail: bublik.drdrdr@gmail.com

Comparative analysis of view technologies for the Spring application

Vadym Borys*, Roman Slezenko*, Beata Pańczyk

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

Abstract. The goal of the article is to choose the most efficient user interface creation tools possible for Spring. The study compares 4 selected view technologies: JSP, Thymeleaf, Wicket and Angular. Time and memory performance tests were carried out using Rest API in Spring. Test results allowed to identify the best solutions.

Keywords: performance; Spring; Angular; JSP; Thymeleaf; Wicket

*Corresponding author.

E-mail address: bublik.drdrdr@gmail.com

1. Wstęp

Oprócz takich kryteriów jak bezpieczeństwo, czytelność kodu, łatwość konfiguracji oraz rozwoju aplikacji, bardzo ważną cechą każdej wykorzystywanej technologii jest wydajność. Każda technologia typu front-end służy do wygenerowania strony internetowej, przesłaniu danych do przeglądarki oraz wyświetleniu treści po stronie użytkownika. Różne technologie mają różny wpływ na obciążenie procesora i wykorzystywanie pamięci operacyjnej (zarówno po stronie serwera, jak i po stronie klienta).

REST (ang. Representational State Transfer) - to styl architektury oprogramowania dla różnych systemów rozproszonych, takich jak WWW, który jest zwykle używany do budowania usług sieciowych [1]. Najważniejszą zaletą serwisu REST jest to, że każdy system może z nim współpracować. Spring z kolei to najpopularniejszy szkielet aplikacji w języku Java, który pozwala również na szybkie tworzenie wysokiej jakości kodu nowoczesnych aplikacji webowych [2, 3]. Do pomiaru obciążenia procesora oraz innych komponentów sprzętowych służy Spring BootActuator, który otwiera punkty wejściowe (ang. endpoints) dla komunikacji za pomocą REST i pozwala na zdalne sprawdzenie stanu uruchomionej aplikacji oraz jej statystyk [4].

Tematem podjętych badań jest porównanie narzędzi do tworzenia widoków (typu front-end), stosowanych dla aplikacji Spring. W artykule zostaną przedstawione cztery technologie do tworzenia interfejsu graficznego: JSP [5] i Wicket[6] oraz nowszy Thymeleaf[7] i Angular [8].Podobne badania były prezentowane m.in. w artykułach [9, 10], gdzie porównywano jedynie po dwa narzędzia. W literaturze nie znaleziono porównań dla opisanych w niniejszym artykule czterech technologii.

2. Cel badań

Celem badań jest wskazanie najbardziej wydajnych technologii tworzenia widoków współpracujących z Rest API w Spring.

Postawiono następujące tezy badawcze:

Angular jest najlepszą, pod względem wydajności czasowej,przy małym wykorzystaniu pamięci RAM, technologią do tworzenia widoku dla Rest API w Spring.

Thymeleaf jest bardzo dobrym narzędziem do tworzenia małych serwisów.

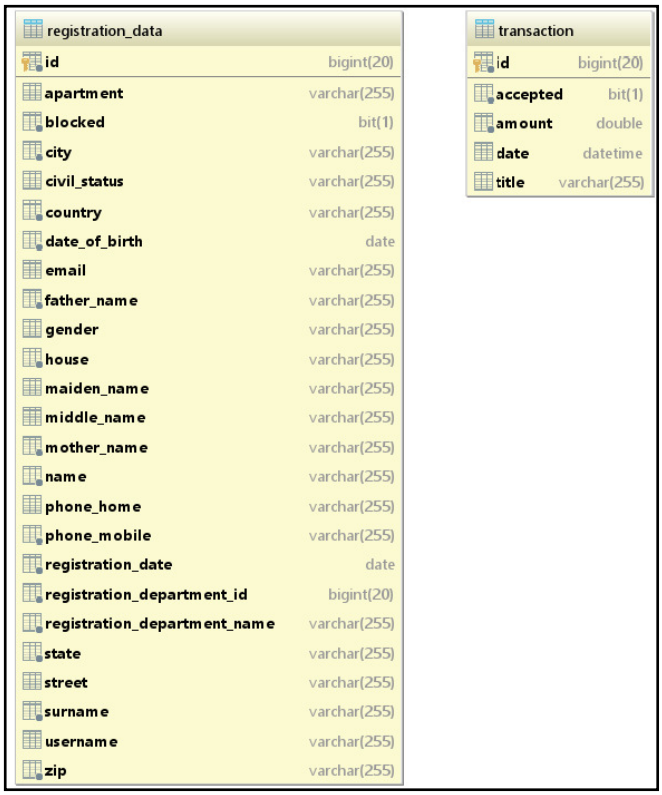
3. Metoda badań

Analizę wydajności czasowej i obciążenia pamięci wykonano za pomocą aplikacji testowej, dla której zaimplementowano cztery różne technologie generowania widoków.

Testowana aplikacja po stronie serwera (back-end) spełnia wymagania pozwalające na wykorzystanie różnych elementów interfejsu użytkownika, którymi są listy, rozbudowane formularze z walidacją oraz duże tabele do prezentacji raportów. Serwerowa część aplikacji testowej (wykonana w Spring,jako część wspólna dla wszystkich testowanych technologii widoku, zapewnia podstawowe mechanizmy dla komunikacji, autoryzacji oraz możliwości dynamicznego ładowania stron.

3.1. Aplikacja testowa

W celu symulowania realistycznych warunków działania aplikacji wykorzystano bazę danych MySQL [11] oraz bibliotekę Hibernate [12], która automatycznie generuje tabele i relacje między nimi na podstawie klas POJO oraz pozwala zarządzać danymi w oparciu o obiekty tych klas. Struktura bazy danych jest przedstawiona na rys. 1. Dla ułatwienia przeprowadzenia badań zostały stworzone klasy pomocnicze oraz konfiguracyjne. Przykład 1 przedstawia kod konfiguracyjny klasy GenerationExecutor, który podczas uruchamiania aplikacji generuje odpowiedni zbiór danych, wykorzystany później w celu wyświetlenia wybranych informacji na stronie. Do pomiaru zajętości pamięci RAM przez aplikację po stronie serwera wykorzystano Spring Actuator i klasę MemoryReader (przykład 2).



Rys. 1. Struktura bazy danych dla aplikacji testowej

Przykład 1. Kod klasy GenerationExecutor generującej przykładowe dane

```
@Service
@RequiredArgsConstructor
public class GenerationExecutor {
    @Value("${data.generate-on-start}")
    private booleangenerationEnabled;
    private final
RegistrationDataGeneratorregistrationDataGenerator;
    private final
TransactionGeneratortransactionGenerator;
    @PostConstruct
    public void generateData() {
        if (generationEnabled) {
        //liczba użytkowników
            registrationDataGenerator.generate(50);
        //liczba transakcji
            transactionGenerator.generate(20000);
        }}}
```

Przykład 2. Kod klasy MemoryReader monitorującej wykorzystywanie pamięci RAM przez aplikację back-end

```
@Service
@RequiredArgsConstructor
public class MemoryReader {

    private final RestTemplate restTemplate;

    public long getMemory(String property) {
        return
(long)(((Double)((Map)((List)this.restTemplate
.getForObject("http://localhost:8080/actuator/metrics/"
+ property, Map.class).get("measurements"))
                .get(0)).get("value"))/(1024*1024));
}
    @Scheduled(fixedRate = 1000)
    public void printMemory() {
System.out.println(getMemory("jvm.memory.used"));
}
}
```

3.2. Konfiguracja środowiska

Badania przeprowadzono na systemach Windows 10 oraz Mac OS Mojave 10.14. Aplikacja back-end działała na serwerzeTomcat.

Parametry komputerów, na których przeprowadzono badania,zostały przedstawione w tabeli 1.

Tabela 1. Parametry komputerów wykorzystanych do badań

Komputer 1	Komputer 2
• Windows 10 Pro 64-bit, • processor: Intel(R) Core(TM) i7 2.6(3.48)GHz, • pamięć RAM: 8 GB, • dysk SSD: 256 GB.	• Mac OS Mojave 10.14, • processor: Intel(R) Core(TM) i7 2,7GHz, • pamięć RAM: LPDDR3 16 GB, • dysk SSD: 512 GB.

3.3. Kryteria i scenariusze badawcze

Wybrane technologie zbadano według następujących kryteriów:

- czas ładowania strony w przeglądarce internetowej;
- obciążenie procesora po stronie serwera;
- wykorzystywanie pamięci RAM;
- inne cechy technologii: dostępność dokumentacji i przykładów, debugowanie, ograniczenia i braki w funkcjonalnościach.

Badania wykonano w oparciu o następujące scenariusze:

- S1 - załadowanie strony zawierającej formularz do tworzenia użytkownika, składający się z dużej liczby różnych elementów, ale bez załadowania danych formularza;
- S2 - wyświetlenie listy elementów w postaci tabeli (mała liczba wierszy);
- S3 - wyświetlenie listy powtarzających się elementów z różnymi danymi w postaci tabeli (10000 wierszy). Test głównie służy do wyodrębniania technologii, które najbardziej efektywnie wykorzystuje przepustowość połączenia inajszybszego wyświetlenia strony.
- S4 - przejście na stronę edycji użytkownika, aktualizacja danych w formularzu i przesłanie zmienionych danych do serwera;

- S5 - przełączenie pomiędzy stronami z małą tabelą w jednej sesji bez czyszczenia pamięci podręcznej a w przypadku testowania technologii Angular, bez przeładowania strony;
- S6 - scenariusz 5 z wykorzystaniem 10000 wierszy w tabeli;
- S7 - przejście na stronę edycji użytkownika, zmiana danych w formularzu i jego zatwierdzenie w jednej sesji.

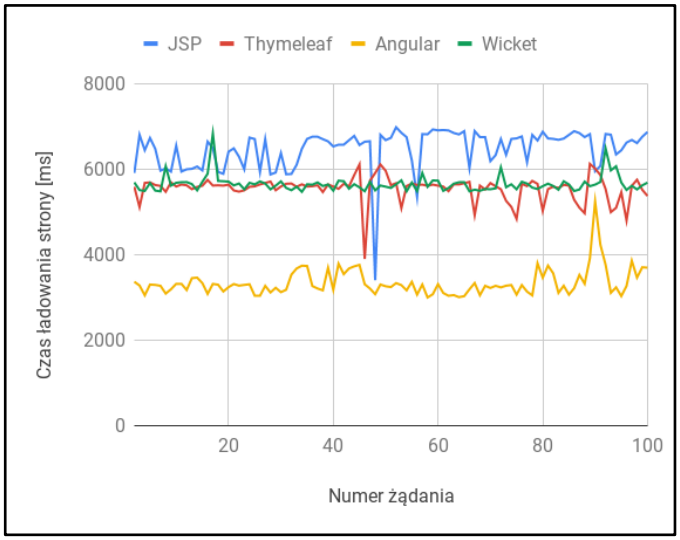
Przed każdym pomiarem dla scenariuszy 1-4 była czyszczona pamięć podręczna, pliki ciasteczek a następnie uruchamiano ponownie emulator przeglądarki. Wszystkie pomiary powtarzano po 100 razy.

Dla testowania front-endu wykorzystano narzędzie developerskie Selenium, pozwalające w łatwy sposób napisać i skonfigurować testy automatyczne, symulujące różne aktywności użytkownika. Jedna z kluczowych zalet Selenium jest obsługa wykonania testów automatycznych na różnych przeglądarkach internetowych.

4. Wyniki badań

4.1. Wydajność czasowa

Różnicę w wydajności badanych technologii najlepiej można ocenić na przykładzie badania zgodnie ze scenariuszem3. Test ten posłużył do wyboru technologii, które najefektywniej wykorzystują przepustowość połączenia do wyświetlenia strony (100 żądań). Rysunek 2 prezentuje czasy załadowania stron a średnie wyniki zestawiono w tabeli 2.

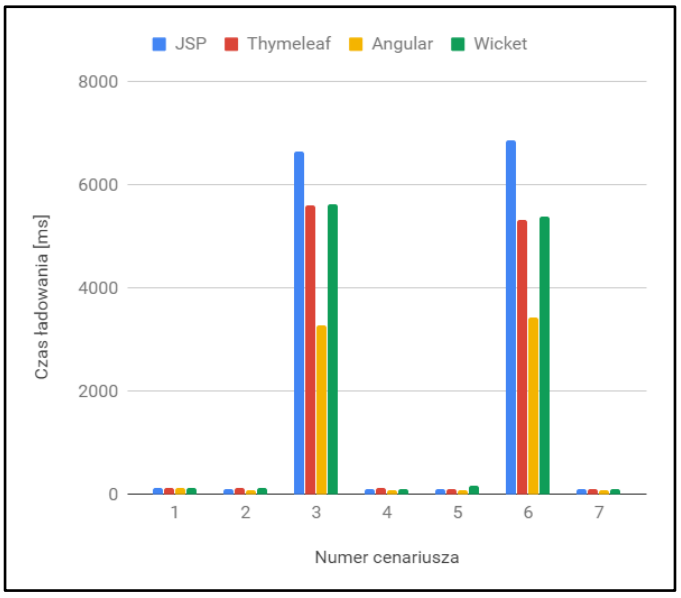


Rys. 2. Czasy wykonania 100 żądań dla S3

Tabela 2. Zagregowane dane pomiarów dla S3(zielone tło oznacza najlepszy wynik)

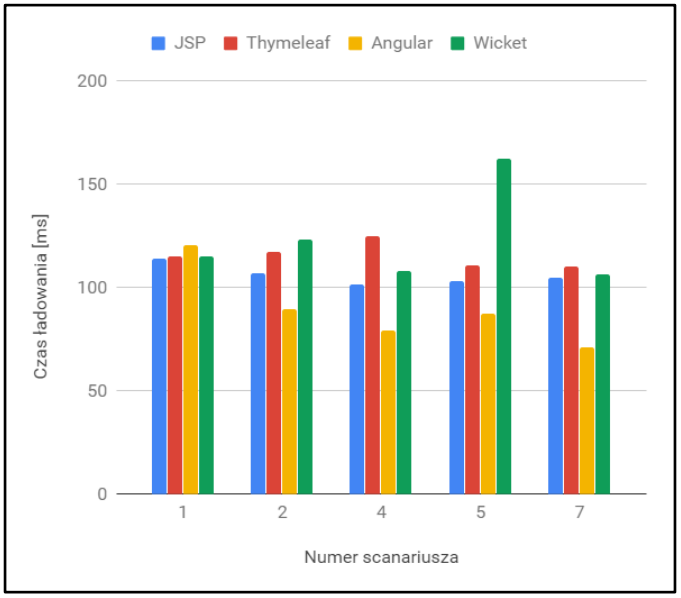
	JSP	Thymeleaf	Angular	Wicket
Minimum [ms]	3400	3899	2997	5467
Maksimum [ms]	6976	6120	5238	6827
Średnia [ms]	6467.46	5539.23	3339.18	5649.18
Mediana [ms]	6657.5	5606.5	3273	5634.5

Porównanie czasów ładowania strony dla wszystkich scenariuszy i badanych technologii jest przedstawione na rysunku3.



Rys.3. Porównanie średnich czasów wykonania żądań dla wszystkich scenariuszy

Ze względu na dużo większe wartości pomiarów czasu ładowania strony w przypadku scenariuszy z dużą liczbą elementów (3 oraz 6),wykres na rysunku 3 nie daje możliwości lepszego porównania czasów w pozostałych scenariuszach. Z tego powodu wyniki dla S1, S2, S4, S5 i S7 przedstawiono oddzielnie na rysunku 4.



Rys. 4. Średni czas wykonania żądań przez badane technologie dla scenariuszyS1, S2, S4, S5 i S7

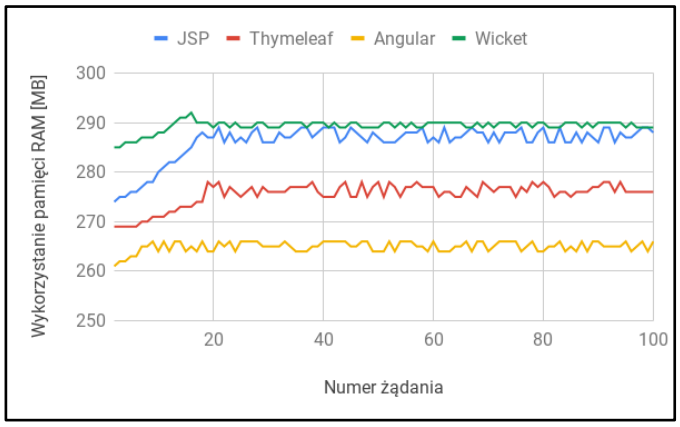
4.2. Obciążenie procesora po stronie serwera

Aplikacja Spring tworzy nowy wątek dla każdego klienta, czyli przy dużym obciążeniu w pełni są wykorzystywane zasoby procesora. Przy wszystkich opracowanych testach dla badań różnych narzędzi typu front-end używano maksymalną ilość zasobów komputera. Oznacza to, że podczas przeprowadzenia każdego badania wykorzystywano sto procent obciążenia procesora. Natomiast od strony serwera w Spring działa to w następujący sposób: system wykorzystuje wielowątkowość i każde żądanie jest obsługiwane w odrębnym wątku. Większa wydajność

konkretnej technologii front-end powoduje możliwość obsłużenia większej liczby zapytań. Ten sposób pozwolił na wskazanie wydajniejszej technologii pod względem obciążenia procesora.

4.3. Wykorzystywanie pamięci RAM

Pomiary wykorzystania pamięci RAM aplikacji działającej po stronie serwera zostały zrealizowane za pomocą Spring Actuator. Pomiary przeprowadzono podczas uruchomionego testu dla każdej badanej technologii. Porównanie wykorzystania pamięci RAM dla czterech technologii jest przedstawione na rys. 5, natomiast tabela 3 prezentuje zagregowane dane z pomiarów. Najlepsze wyniki w tym badaniu uzyskał Angular.



Rys. 5. Wykorzystanie pamięci RAM podczas wykonywania żądań do aplikacji Spring przez poszczególne technologie front-end

Tabela 3. Zagregowane dane pomiarów wykorzystywania pamięci RAM przez aplikację back-end (zielone tło oznacza najlepszy wynik)

	JSP	Thymeleaf	Angular	Wicket
Minimum [MB]	274	269	261	285
Maksimum [MB]	289	278	266	292
Średnia [MB]	286.16	275.53	264.97	289.31
Mediana [MB]	287	276	265	290

4.4. Inne cechy

Subiektywna ocena niemierzalnych cech technologii widoków została zestawiona w tabeli 4. Każde kryterium oceniono w skali od 1 do 5, gdzie większa liczba oznacza lepszą ocenę.

Tabela 4. Oceny różnych właściwości badanych technologii (zielonym kolorem są zaznaczone najwyższe oceny, czerwonym – najniższe)

	JSP	Thymeleaf	Angular	Wicket
Próg wejścia	4	5	4	2
Składnia języka	3	5	5	3
Debugowanie	3	3	5	2
Dostępne funkcjonalności	3	4	5	3
Wymaganawiedza	4	5	3	2
Wsparcie	4	4	5	1
Rozwój	3	3	5	1
Suma punktów	**24**	**29**	**32**	**14**
Język technologii	HTML + Java	HTML + znaczniki	Typescript + HTML	Java + HTML

5. Wnioski

Wobec braku oficjalnej informacji o porównaniu technologii analizowanych w niniejszej pracy, oraz braku innych publikacji na ten temat, nie ma możliwości porównania wyników własnych z wynikami z innych źródeł. Są dostępne jedynie nieoficjalne i niezatwierdzone źródła w postaci blogów oraz dyskusji w serwisach społecznościowych, wśród których często znajdują się sprzeczne wnioski, które nie mogą być traktowane jako badania naukowe.

Tezy postawione w pracy zostały potwierdzone. Szkielet aplikacji webowych Angular okazał się najbardziej efektywny, prawie we wszystkich kategoriach. Natomiast technologia Wicket nie spełniła oczekiwań i miała najgorsze wyniki. Pozostałe technologie - JSP oraz Thymeleaf pokazały nieistotną różnicę w wydajności (tabela 2, tabela 3, rys. 3, rys.4).

Thymeleaf okazał się bardzo dobrym narzędziem do tworzenia mniejszych serwisów (tabela 4), jest łatwy do obsługi na starszych urządzeniach oraz przeglądarkach, które wspierają starsze wersje języka JavaScript. Dodatkowo jest on jedną z najlepszych technologii dla migracji starszych systemów z JSP i innych technologii, bez wprowadzania dużych zmian do kodu źródłowego. W przypadku starych systemów nie spowoduje to wystąpienia dużej liczby błędów, jak na przykład w przypadku przerobienia całego systemu na architekturę REST API i konfigurowania serwerów dla dodatkowej aplikacji Angular.

Literatura

[1] Sanjay Patni: Pro RESTful APIs: Design, Build and Integrate with REST,JSON, XML and JAX-RS 1st ed. Edition. Apress 2017.

[2] Amuthan G : Spring MVC Beginner's Guide Paperback . Packt Publishing - ebooks Account 2014

[3] Craig Walls: Spring in Action - Fifth Edition. Manning Publications 2018.

[4] https://www.callicoder.com/spring-boot-actuator [13.10.2019]

[5] Hans Bergsten: JavaServer Pages - 3rd Edition. O'Reilly Media 2003.

[6] Michael Good: Thymeleaf with Spring Boot: An easy to follow guide. Amazon Digital Services LLC 2018

[7] Igor Vaynberg: Apache Wicket Cookbook. Packt Publishing 2011.

[8] Jeremy Wilken: Angular in Action 1st Edition. Manning Publications 1 edition 2018.

[9] https://siftery.com/product-comparison/angularjs-vs-thymeleaf, AngularJS vs Thymeleaf - Product Comparison [26.03.2019].

[10] https://siftery.com/product-comparison/angularjs-vs-apache-wicket, Angular vs Wicket – Product comparsion [27.03.2019].

[11] https://www.mysql.com/about, oficjalna strona MySQL [10.10.2019].

[12] https://hibernate.org/orm, oficjalna strona Hibernate ORM [10.10.2019].

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 8 (2019) 82-87
Wysłane: 2019-10-10
Przyjęte: 2019-10-29

Analiza wydajnościowa wybranych narzędzi do budowy aplikacji Single Page Application

Yehor Timanovskyi*, Małgorzata Plechawska-Wójcik
Politechnika Lubelska, Katedra Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. W artykule przedstawiono analizę wydajnościową z wykorzystaniem wybranych narzędzi do budowy Single Page Application. Do oceny wydajności aplikacji testowej używana została przeglądarka Google Chrome z narzędziem DevTools. Całkowita liczba wszystkich testów wynosiła 112. W ramach przeprowadzonego badania powstała aplikacja testowa z wykorzystaniem różnych szkieletów JavaScript - szkieletu Angular i szkieletu Vue.js.

Słowa kluczowe: SPA; szkielet JavaScript; Angular; Vue.js; wydajność

*Autor do korespondencji.

Adres e-mail: yehor.timanovskyi@pollub.edu.pl

Performance analysis of selected tools for building a Single Page Application

Yehor Timanovskyi*, Małgorzata Plechawska-Wójcik
Department of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract. The article presents analysis of the performance of selected tools to build a Single Page Application. Chrome browser with DevTools tool was used to evaluate the performance of the test application. The total number of all tests was 112. As part of the study, a test application was created using different JavaScript frameworks - the Angular framework and the Vue.js framework.

Keywords: SPA; frameworks JavaScript; Angular; Vue.js; performance

*Corresponding author.

E-mail address: yehor.timanovskyi@pollub.edu.pl

1. Wstęp

Aplikacje internetowe są wykorzystywane jako skuteczne rozwiązanie dla szerokiej gamy zadań biznesowych. W ostatnich latach aplikacje internetowe rozwijają się niezwykle szybko, stopniowo wypierając stacjonarne rozwiązania i stając się najważniejszym elementem biznesu w dzisiejszym świecie.

Jednostronicowe aplikacje (ang. *Single Page Applications*, zwane dalej SPA) są to aplikacje internetowe, których składniki są pobierane raz na jednej stronie i aktualizowane w razie potrzeby. W miarę wzrostu popularności jednostronicowych aplikacji, duża część przetwarzania danych jest skoncentrowana po stronie klienta.

Szkielety (ang. *frameworks*) są tworzone przede wszystkim, aby języki programowania (na przykład, *JavaScript*) ewoluowały. Szkielety to specjalne narzędzia, definiowane jako duże zestawy bibliotek wielokrotnego użytku do implementacji podstawowej struktury aplikacji. Szkielety JavaScript są używane do rozszerzenia możliwości programisty i upraszczania programowania poprzez dostarczanie zestawu gotowych komponentów dla bardziej skomplikowanych funkcjonalności. Korzystanie ze szkieletów JavaScript za pomocą natywnego JavaScript sprawia, że proces tworzenia interfejsu jest w dłuższej perspektywie szybszy.

Głównym problemem stojącym przed programistami jest odpowiedni dobór szkieletu lub języka do wykonywania określonej pracy. Dlatego w artykułe poruszono zagadnienie analizy wydajnościowej szkieletów JavaScript. Wyniki badania pozwolą ustalić, czy jest szkielet Vue.js jest bardziej wydajny od szkieletu Angular.

2. Cel i teza

Celem niniejszego artykułu jest bezpośrednie porównanie wydajności szkieletów JavaScript Vue.js oraz Angular.

Za tezę badawczą przyjęto:

Szkielet Vue.js jest bardziej wydajny od szkieletu Angular.

3. Przegląd literatury

Artykuł [1] poświęcony jest wydajnościowej analizie trzech współczesnych szkieletów JavaScript używanych w kontekście JSS (ang. *Sitecore JavaScript Services*). Szczegółowo opisana została architektura wzorców Model-Widok-Kontroler (ang. *Model-View-Controller*), MVP (ang. *ModelView-Presenter*) i MVVM (ang. *Model-View-ViewModel*).

W artykule [2] została opisana historia JavaScriptu oraz technologia ReactJS / Native wraz z ich zaletami i wadami.

Autorzy opisują również jak technologie React integrują się z innymi rozwiązaniami opartymi o język JavaScript.

Celem badań przedstawionych w artykule [3] jest zidentyfikowanie i zrozumienie czynników, które wpływają na wybór frameworka JavaScript spośród innych dostępnych na rynku. Badania realizowane są, aby odpowiedzieć na następujące pytanie badawcze: «Jakie czynniki prowadzą do podjęcia decyzji o wyborze konkretnego szkieletu JavaScript?».

Każdy z artykułów publikowanych [4], [5], [6] przedstawia poszczególne badania, dotyczące wspólnego tematu - porównania szkieletów JavaScript. W badaniach tych zostały podane szczegółowe informacje, których wyniki pozwoliły na ilościową ocenę pomiarów wskaźników porównawczych poprzez przeprowadzenie różnych eksperymentów. Autorzy opisują wpływ środowiska, w którym została uruchomiona aplikacja testowa, która korzysta ze szkieletu JavaScript.

Autor artykułu [7] przedstawia dokładny opis wykorzystania nowoczesnej architektury aplikacji internetowych, różnice między aplikacjami wielostronicowymi, a aplikacjami jednostronicowymi oraz wyjaśnia zasadę działania SPA.

Przegląd literatury pozwolił ocenić szkielety z różnych stron, zwrócić uwagę na popularne kryteria oceny szkieletów. Popularne kryteria to wydajność i rozmiar aplikacji.

4. Opis narzędzi badawczych

Obecnie JavaScript jest najbardziej popularnym językiem programowania aplikacji klienckich, dlatego otrzymał szerokie uznanie całej społeczności programistów. Korzystanie ze szkieletów JavaScript staje się coraz bardziej popularne, co uzasadnia fakt, że tworzenie szkieletów zmniejsza obciążenie procesu tworzenia aplikacji internetowych.

Analizując dane z Internetu, można zauważyć, że istnieją setki platform do tworzenia aplikacji internetowych. Każdy szkielet ma wiele atrakcyjnych funkcji i dodatków, dlatego wybór odpowiedniego jest dość trudny, a niewłaściwa decyzja może być główną przyczyną porażki projektu. Często pomijanym aspektem jest wydajność środowiska JavaScript, co jest istotne szczególnie w przypadku przedsiębiorstw tworzących złożone aplikacje.

4.1 Wady i zalety korzystania ze szkieletów JavaScript

Podczas wyboru szkieletów mogą wystąpić pewne trudności związane ze zdefiniowaniem zadań, które może wykonywać, dlatego należy ostrożnie podejść do kwestii wyboru i rozważyć wszystkie wady i zalety poszczególnych rozwiązań.

Poniżej znajdują się zalety korzystania ze szkieletów JavaScript [8]:

1) Elastyczność i skalowanie — elastyczne rozwiązanie niestandardowych zadań i możliwość dalszego rozszerzenia funkcjonalności poprzez połączenie bibliotek stron trzecich lub oddzielnych klas.
2) Zastosowanie architektury Model-Widok-Kontroler (ang. Model-View-Controller, zwany dalej MVC), znacznie poprawia funkcjonalność i elastyczność projektu. Jest to przydatne podczas projektowania małych aplikacji, ale niezbędne do prawidłowej funkcjonalności większych projektów.
3) Bezpieczeństwo. Szkielety mają wbudowane zabezpieczenia przed podstawowymi atakami (przykładowo, SQL injection).
4) Szkielet pozwala skoncentrować się na rozwiązywaniu problemów architektonicznych, a nie podstawowych, jak w projektowaniu bez jego użycia.
5) Jakość materiałów źródłowych. Na etapie tworzenia projektu szkielet pomaga pozbyć się typowych błędów, ponieważ istnieją pewne podstawowe zasady korzystania ze szkieletów.

Główną zaletą szkieletów, jest to, że są one idealne do tworzenia skalowalnych, unikalnych projektowo stron internetowych. Żaden duży projekt nie został rozwinięty przy użyciu gotowych CMS.

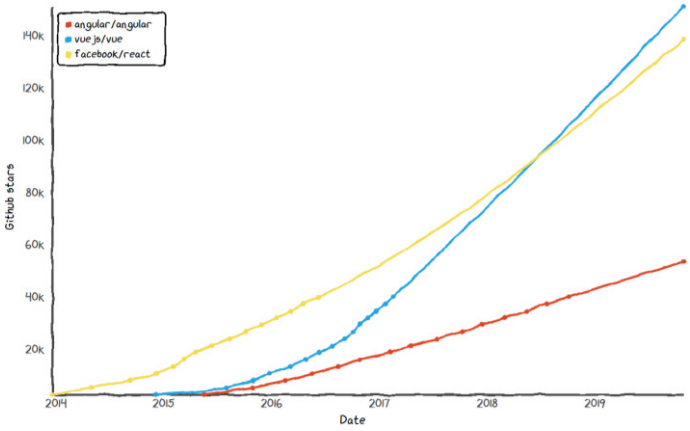
Wady korzystania ze szkieletów są dosyć umowne i niewielkie w porównaniu z korzyściami [8]:

1) Trudności w obsłudze.
2) Cena opracowania — koszt standardowego serwisu wykonanego na szkielecie od podstaw będzie wyższy niż na gotowych CMS, ponieważ zajmuje to kilka razy więcej czasu na opracowanie.
3) Trudności w procesie uczenia się.
4) Brak gotowych modułów i komponentów, które mógłby zainstalować klient. Wszystkie modyfikacje należy zgłaszać do deweloperów.

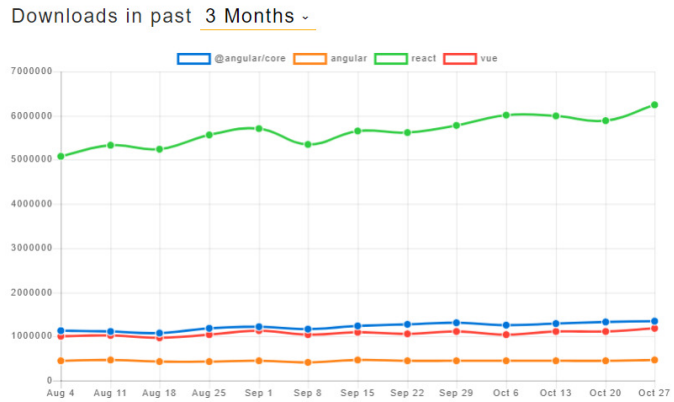
Opierając się na takich źródłach internetowych jak HotFrameworks i GitHub, statystyki dotyczące pobierania NPM [9], Stack Overflow, Google Trends [10], Developer Framework Satisfaction, można określić najbardziej popularne szkielety.

Dla porównania popularności szkieletów, jedną z pierwszych stron internetowych, które muszą być wymienione jest GitHub. Z punktu widzenia statystyki, system gwiazdkowy GitHub jest najbardziej przydatnym narzędziem do efektywnego wyszukiwania potrzebnego szkieletu (rys. 1).

Innym źródłem oceny pozycji szkieletów są statystyki pobierania dotyczące NPM (ang. *Node Package Manager*) [11]. Ponieważ NPM jest najpopularniejszym miejscem do instalowania bibliotek, jego liczby pobrań mogą dostarczyć wiarygodnych informacji na temat tego, jak dobrze technologia jest postrzegana i wykorzystywana przez społeczność (rys 2).



Rys. 1. Statystyka GitHub stars szkieletów [11]



Rys 2. Statystyka pobierania szkieletów [9]

W ramach przeprowadzonej analizy powyżej wymienionych źródeł można wyróżnić dwa najczęściej używane szkielety: React i Angular. Do późniejszego badania zdecydowano się użyć Angular oraz Vue.js, który nie tylko zyskuje na popularności, ale również jest uważany za jeden z najbardziej obiecujących.

4.2 Szkielet Angular

Angular to popularna platforma programistyczna umożliwiająca tworzenie aplikacji mobilnych i stacjonarnych [12]. Został on pierwotnie stworzony przez pracowników Google Misko Hevery i Adama Abronsa w 2008 roku.

Szkielet jest zbudowany w całości w języku TypeScript i użycie go w rozwoju JavaScript jest zalecane, choć nie obowiązkowe. Język TypeScript to nadzbiór JavaScript, który dodaje opcjonalne pisanie statyczne do JavaScript oraz został pierwotnie wprowadzony i jest nadal obsługiwanym przez Microsoft oprogramowaniem typu open-source.

Angular, podobnie jak wiele innych szkieletów, jest oparty na komponentach. Oznacza to, że komponenty są głównymi elementami składowymi - mogą wyświetlać informacje, renderować szablony i wykonywać działania na danych.

Powiązanie danych w ramach komponentu jest również godne uwagi. Zasadniczo dotyczy to wymiany danych między widokiem (szablon HTML) a modelem (plikiem TypeScript).

Routing odgrywa ważną rolę dla użyteczności SPA. Definiując trasy w osobnym pliku lub module, Angular obsługuje logikę, dla której składnik powinien być wyświetlany w zależności od aktualnie aktywnej ścieżki adresu URL.

W tym badaniu jest wykorzystany i oceniany Angular w wersji 6.4.7.

4.3 Szkielet Vue.js

Vue.js to innowacyjna platforma JavaScript typu open-source. Szkielet jest łatwo dostępny, wszechstronny i produktywny [13]. Wydany w 2014 r. przez Ewana You, byłego pracownika Google, który miał duże doświadczenie w pracy z AngularJS. Jednak wersja 1.0.0 pojawiła się dopiero w październiku 2015 r.

Vue nazywany jest często środowiskiem progresywnym, którego można używać do tworzenia interfejsów użytkownika. Chociaż nie był ściśle powiązany z szablonem Model-ViewViewModel (MVVM), został częściowo zainspirowany zasadami projektowania.

Vue korzysta z wielu przydatnych funkcji React i Angular, takich jak wirtualny DOM (ang. *Document Object Model*, zwany dalej DOM) i szablony. Wiele z tych funkcji zostało zaimplementowanych w jeszcze bardziej złożony sposób. Skalowalność jest jedną z głównych zalet. Jedną z cech Vue jest to, że jest w pełni rozwijany przez społeczność open source, a nie duże przedsiębiorstwo.

Podobnie jak Angular, Vue używa dyrektyw w swoich szablonach. Są one wykorzystywane do wiązania danych, obsługi zdarzeń i nie tylko oraz wizualnie oznaczone prefiksem *v*.

Rekwizyty Vue przypominają React. Aby wysłać dane z komponentów nadrzędnych do podrzędnych, jedną z opcji jest użycie rekwizytów.

W tym badaniu jest wykorzystany i oceniany Vue.js w wersji 2.6.10.

5. Plan badań

W SPA, gdzie stosowane są narzędzia JavaScript, szybkość reagowania i interakcji z klientem w dużej mierze zależy od wydajności platformy. W związku z tym wydajność szkieletów jest ważnym aspektem, który powinien być brany pod uwagę przy ocenie. Do oceny wydajności aplikacji testowej będzie używana przeglądarka Google Chrome z narzędziem DevTools [14].

Do weryfikacji wydajności będą wykorzystywane dwie strony aplikacji:

1) strona z listą dostępnych treningów wybranej kategorii (rys. 3);
2) strona ustawień użytkownika (rys. 4 – rys. 5).

Wykorzystane zostaną podane strony, ponieważ istnieje możliwość wyświetlenia nieograniczonej liczby treningów w pierwszym przypadku, i historii treningów w innym.

Zdecydowano się użyć cztery scenariusze badania, które różnią się liczbą treningów: 5, 100, 1000, 10000.

Biorąc pod uwagę, że termin „Wydajność" jest kompleksowym i wspólnym kryterium, w badaniu wykorzystane zostaną konkretne metryki. Poniżej znajduje się lista wybranych metryk. Wszystkie one mierzone w milisekundach (ms).

1) czas ładowania HTML (ang. *Loading time*),
2) całkowity czas ładowania strony (ang. Total time),
3) czas działania procesora graficznego (ang. *Graphics Processing Unit*, zwany dalej GPU).

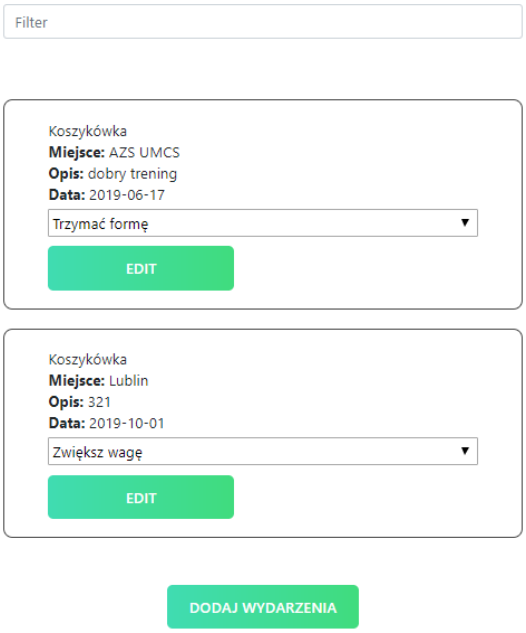
Porównanie się odbędzie w następujący sposób:

1) Do każdego szkieletu i scenariusza test będzie uruchomiony pięć razy, aby znaleźć wartość średnią i 2 razy, aby porównać tzw. trottling (ang. *Throttling*, zwany dalej trottling). Termin ten odnosi się do zjawiska polegającego na obniżeniu taktowania karty graficznej (GPU). Narzędzie DevTools wspiera czterokrotny i sześciokrotny poziom obniżenia taktowania. Odpowiednio, będzie wykorzystany każdy poziom do każdego scenariusza. Całkowita liczba wszystkich testów wynosi 112. Należy zauważyć, że scenariusze do porównania trottlingu będą używane do zrozumienia ograniczeń przepustowości aplikacji w celu określenia niezawodności systemu podczas ekstremalnych obciążeń (maksymalnej liczbie DOM elementów). Również dlatego, żeby odpowiedzieć na pytania o wystarczającej wydajności systemu w przypadku, gdy bieżące obciążenie znacznie przekracza przewidywaną maksymalną liczbę.
2) Wszystkie uzyskane dane zostaną wprowadzone do każdego szkieletu i osobnej tabeli, a następnie dla każdej metryki obliczona wartość średnia. Po otrzymaniu wartości średniej dane będą zgrupowane do jednej tabeli. Również metryki każdego szkieletu zostaną przedstawione w postaci wykresów.

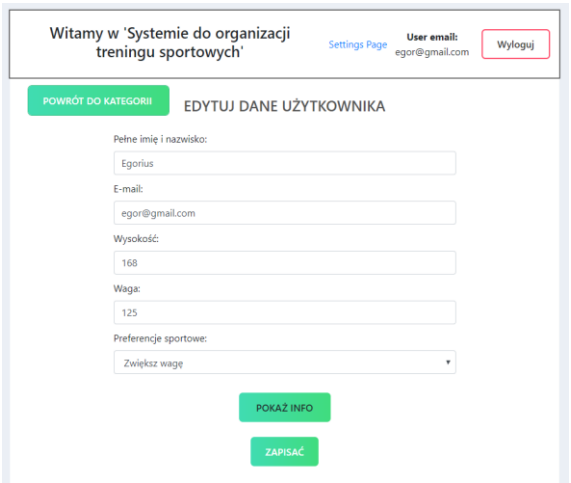
6. Aplikacja badawcza

Na potrzeby badania zrealizowano aplikację testową: system do organizacji treningów sportowych. Zadaniem systemu jest dostarczenie użytkownikowi optymalnego treningu dobranego ze względu na jego potrzeby i wymagania. Aplikacja oferuje kompleksowe formy treningów, pozwalające uzyskać dobrą formę sportową. Ponadto system umożliwia dokonanie oceny stanu fizycznego użytkownika, monitorowanie skuteczności treningu, a także dobór optymalnych ćwiczeń w celu uzyskania pożądanego efektu.

Do przeprowadzenia analizy zostały stworzone dwie aplikacje z maksymalnie podobnym UI. Każda z nich zawiera trzy główne moduły: logowanie i rejestracja, ustawienia użytkownika, stworzenie i wyświetlanie listy treningów. Na rysunkach 3 – 5 przedstawione są strony, które będą potrzebne podczas analizy wydajności.



Rys. 3. Strona z listą dostępnych treningów wybranej kategorii



Rys. 4. Formularz edycji danych strony ustawień użytkownika

Rys. 5. Lista ukończonych i przyszłych treningów strony ustawień użytkownika

7. Realizacja badań

Przed rozpoczęciem analizy wydajnościowej została przeprowadzona implementacyjna analiza porównawcza. Wyniki analizy porównawczej pozwalają stwierdzić, że projekt stworzony za pomocą szkieletu Vue.js jest mniejszy o 41 kB od projektu zrealizowanego przy użyciu szkieletu Angular. Dla tworzenia małych projektów otrzymana różnica nie jest zasadnicza. Jednak, w przypadku dużych aplikacji - każdy megabajt danych jest istotny. Należy zauważyć, że im mniejszy rozmiar gotowej aplikacji, tym szybciej się ładuje strona, i tym mniej czasu potrzeba na analizę w przeglądarce.

Implementacyjna analiza zawiera jeszcze jedną metrykę - liczba linii kodu. Całkowita liczba linii kodu użytych w aplikacji z wykorzystaniem szkieletu Angular wynosi 1801, do szkieletu Vue.js – 1558.

Podczas prowadzenia badań wydajnościowych ważnym aspektem, który bezpośrednio wpływa na ocenę, jest moc obliczeniowa komputera. Testy zostały uruchomione na komputerze o następujących parametrach:

1) procesor - Intel Core i7 - 4400U;
2) pamięć RAM - 12 GB;
3) dysk - 256GB SSD;
4) karta graficzna - Intel HD Graphics 4400;
5) system operacyjny - Windows 10 Pro.

W trakcie badań okazało, że obliczeniowa moc komputera nie pozwala na renderowanie 10000 treningów na jednej stronie, dlatego zdecydowano się zmniejszyć liczbę do tego scenariusza dwukrotnie.

Poniżej przedstawiono wyniki analizy wydajności dla każdej ze stron w postaci tabel (tabela 1 - 2). Im krótszy czas, tym lepiej działa aplikacja. Do wygodnego przeglądania tabel kolorem żółtym zaznaczono szkielet Angular. Niebieskim kolorem zaznaczona kolumna szkieletu Vue.js. Pomarańczowym kolorem zaznaczona kolumna różnicy procentowej pomiędzy szkieletami Vue.js a Angular.

Porównanie wyników odbywa się między scenariuszami na podstawie podania wartości średniej, odchylenia standardowego, różnicy procentowej każdej metryki bez uwzględniania wartości zerowych.

Po przeprowadzeniu wszystkich 112 testów zwrócono uwagę na to, jak trottling wpływa na wyniki. Wybrany wskaźnik (4x lub 6x) w porównaniu do testów bez wykorzystania trottlingu, zwiększa wyniki średnio o 37% - 56% dla 4x, i odpowiednio 125% - 410% dla 6x.

W niektórych testach z powodu niewystarczającej mocy komputera wyniki nie zostały przedstawione w narzędziu DevTools, ale wyświetlone na stronie z poprawną liczbą treningów. W takim przypadku dla takiego testu do tabeli podano wartość 0.

Na podstawie uzyskanych wyników, można powiedzieć, że najczęściej takie testy zostały uruchomione do scenariusza nr 4. W przypadku otrzymania procentowej wartości ujemnej to oznacza, że szkielet Vue.js jest bardziej wydajny w porównaniu do szkieletu Angular.

Analizując wyniki badań, można stwierdzić, że dla metryk "Czas działania procesora graficznego", "Całkowity czas ładowania strony" szkielet Vue.js wykazuje lepszą wydajność.

Tabela 1. Wyniki analizy strony z listą dostępnych treningów [ms]

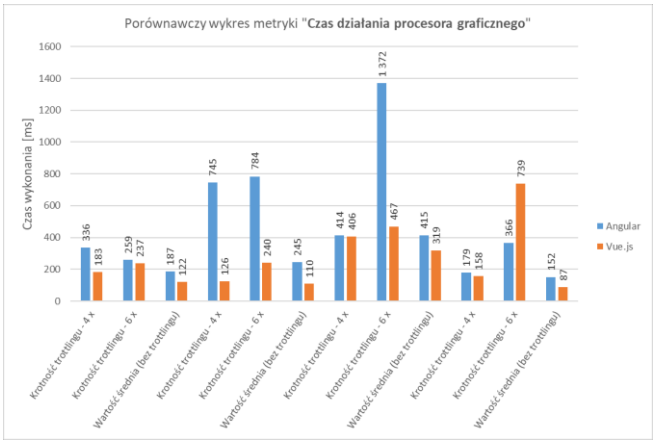
Scenariusz		Metryki								
		Czas ładowania HTML			Całkowity czas ładowania strony			Czas działania procesora graficznego		
		Angular	Vue.js	Różnica [%]	Angular	Vue.js	Różnica [%]	Angular	Vue.js	Różnica [%]
№ 1 (5 treningów)	Krotność trottlingu - 4 x	145,0	169,0	17%	18954,0	6061,0	-68%	336,0	183,0	-46%
	Krotność trottlingu - 6 x	248,0	243,0	-2%	27467,0	8023,0	-71%	259,0	237,0	-8%
	Wartość średnia (bez trottlingu)	44,0	42,2	-4%	5258,3	1427,4	-73%	187,0	122,2	-35%
№ 2 (100 treningów)	Krotność trottlingu - 4 x	189,0	163,0	-14%	30408,0	5071,0	-83%	745,0	126,0	-83%
	Krotność trottlingu - 6 x	332,0	276,0	-17%	40938,0	8028,0	-80%	784,0	240,0	-69%
	Wartość średnia (bez trottlingu)	42,4	48,0	13%	5033,8	1702,0	-66%	245,4	110,4	-55%
№ 3 (1000 treningów)	Krotność trottlingu - 4 x	150,0	170,0	13%	21927,0	16327,0	-26%	414,0	406,0	-2%
	Krotność trottlingu - 6 x	242,0	324,0	34%	68559,0	29931,0	-56%	1372,0	467,0	-66%
	Wartość średnia (bez trottlingu)	44,2	52,0	18%	8038,8	4268,2	-47%	414,6	318,8	-23%
№ 4 (5000 treningów)	Krotność trottlingu - 4 x	0,0	0,0	0%	24815,0	15094,0	-39%	179,0	158,0	-12%
	Krotność trottlingu - 6 x	0,0	0,0	0%	43091,0	125709,0	192%	366,0	739,0	102%
	Wartość średnia (bez trottlingu)	0,0	0,0	0%	7704,4	5179,8	-33%	152,0	87,2	-43%

Tabela 2. Wyniki analizy strony ustawień użytkownika [ms]

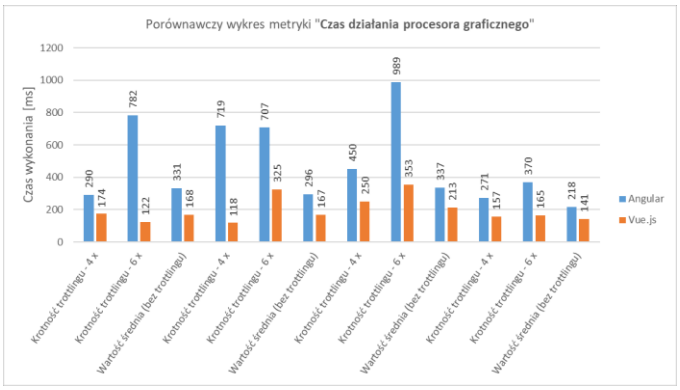
Scenariusz		Metryki								
		Czas ładowania HTML			Całkowity czas ładowania strony			Czas działania procesora graficznego		
		Angular	Vue.js	Różnica [%]	Angular	Vue.js	Różnica [%]	Angular	Vue.js	Różnica [%]
№ 1 (5 treningów)	Krotność trottlingu - 4 x	162,0	176,0	9%	17178,0	4209,0	-75%	290,0	174,0	-40%
	Krotność trottlingu - 6 x	180,0	271,0	51%	28234,0	6366,0	-77%	782,0	122,0	-84%
	Wartość średnia (bez trottlingu)	41,6	44,8	8%	4912,0	1778,0	-64%	331,2	168,2	-49%
№ 2 (100 treningów)	Krotność trottlingu - 4 x	114,0	178,0	56%	19252,0	4171,0	-78%	719,0	118,0	-84%
	Krotność trottlingu - 6 x	231,0	315,0	36%	31935,0	7264,0	-77%	707,0	325,0	-54%
	Wartość średnia (bez trottlingu)	47,4	44,4	-6%	5116,0	1973,0	-61%	296,0	166,8	-44%
№ 3 (1000 treningów)	Krotność trottlingu - 4 x	184,0	169,0	-8%	22640,0	6155,0	-73%	450,0	250,0	-44%
	Krotność trottlingu - 6 x	248,0	283,0	14%	48300,0	9126,0	-81%	989,0	353,0	-64%
	Wartość średnia (bez trottlingu)	46,2	45,0	-3%	7373,0	3121,6	-58%	336,8	213,2	-37%
№ 4 (5000 treningów)	Krotność trottlingu - 4 x	160,0	165,0	3%	25561,0	7530,0	-71%	271,0	157,0	-42%
	Krotność trottlingu - 6 x	0,0	0,0	0%	41478,0	10639,0	-74%	370,0	165,0	-55%
	Wartość średnia (bez trottlingu)	61,6	42,0	-32%	9871,8	4472,8	-55%	217,8	140,6	-35%

Metryka czas działania procesora graficznego została przedstawiona w postaci wykresów słupkowych (rys. 6 - 7). Na wykresach widać, że z 32 przeprowadzonych testów

wyłącznie w 2 szkielet Vue.js pokazuje gorsze wyniki, co wskazują na widoczną przewagę tego szkieletu.



Rys. 6. Porównawczy wykres metryki „*Czas działania procesora graficznego*" strony z listą dostępnych treningów [ms]



Rys. 7. Porównawczy wykres metryki „*Czas działania procesora graficznego*" strony ustawień użytkownika [ms]

8. Wnioski

Cele postawione w artykule zostały spełnione. Aby zrealizować badania została stworzona aplikacja sportowa. Program spełnia wszystkie wymagania funkcjonalne i niefunkcjonalne. Do tworzenia aplikacji wykorzystano narzędzia w najnowszych wersjach.

Do weryfikacji wydajności zostały wykorzystywane dwie strony aplikacji i cztery scenariusze badania, które różnią się liczbą treningów. Całkowita liczba wszystkich testów wynosiła 112.

Zarówno Vue.js jak i Angular to bardzo wydajne narzędzia, które ułatwiają tworzenie SPA. Wyniki przeprowadzonego badania pozwalają na potwierdzenie tezy o tym, że Vue.js jest bardziej wydajny w porównaniu do szkieletu Angular.

Literatura

[1] Petukhova, E. (2019). Sitecore JavaScript Services Framework Comparison.

[2] Boaler, S. (2019). ReactJS a Viable Career Option?.

[3] Pano, A., Graziotin, D., & Abrahamsson, P. (2018). Factors and actors leading to the adoption of a JavaScript framework. Empirical Software Engineering, 23(6), 3503-3534.

[4] Nägele, T., Hooman, J., Zigterman, R., & Brul, M. (2015). Client-side performance profiling of JavaScript for web applications. Universidad de Radbound.

[5] Ferreira, J. (2018). A JavaScript Framework Comparison Based on Benchmarking Software Metrics and Environment Configuration.

[6] Mariano, C. L. (2017). Benchmarking javascript frameworks.

[7] Voutilainen, J. (2017). Evaluation of Front-end JavaScript Frameworks for Master Data Management Application Development.

[8] Проценко, М. М. (2016). Аналіз фреймворків як засобів розробки Web-додатків. Міжнародний науковий журнал, (6 (1)), 86-89.

[9] Npm trends, https://www.npmtrends.com/@angular/core-vs-angular-vs-react-vs-vue [09.2019]

[10] Google Trends, https://trends.google.pl/trends/explore cat=31&date=2016-03-01%202019-03-01&q=React,Angular,Vue [08. 2019]

[11] Star history, http://www.timqian.com/star-history/#angular/angular&vuejs/vue&facebook/react [08. 2019]

[12] Angular, https://angular.io/docs [08.2019]

[13] Vue.js, https://vuejs.org [08.2019]

[14] Chrome DevTools, https://developers.google.com/web/tools/chrome-devtools [09.2019]

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 14 (2020) 88-93
Wysłane: 2019-10-24
Przyjęte: 2019-10-31

Analiza wybranych metod oceny użyteczności w procesie tworzenia aplikacji internetowych

Krzysztof Nowak*, Daniel Samolej
Politechnika Lubelska, Katedra Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. Artykuł przedstawia analizę wykorzystania testów A/B, metody KLM, heurystyk Nielsena oraz badań na użytkownikach, w procesie tworzenia autorskiej gry przeglądarkowej typu multiplayer. Badania przeprowadzono w trakcie tworzenia aplikacji, na różnych etapach jej powstawania. Celem badań było zweryfikowanie korzyści płynących z zastosowania, wspomnianych metod.

Słowa kluczowe: użyteczność; KLM; Nielsen

*Autor do korespondencji.

Adres e-mail: krzysztof376@gmail.com

Analysis of selected usability assessment methods in the process of creating web applications

Krzysztof Nowak*, Daniel Samolej
Department of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract. The article presents the analysis of the use of A/B tests, KLM method, Nielsen heuristics and user research in the process of creating a proprietary multiplayer browser game. The research was carried out during application development, at various stages of its creation. The aim of the study was to verify the benefits of using these methods.

Keywords: usability; KLM; Nielsen

*Corresponding author.

E-mail address: krzysztof376@gmail.com

1. Wstęp

Szybki rozwój technologiczny na świecie, owocuje coraz większą liczbą powszechnie dostępnych aplikacji internetowych. Ze względu na dużą konkurencyjność tychże aplikacji, bardzo ważne stało się ich odpowiednie zaprojektowanie. Mając dużą liczbę alternatywnych rozwiązań, użytkownik może bardzo szybko porzucić daną aplikację na rzecz innej, istotnym elementem każdej aplikacji, stała się więc, jej użyteczność. Spowodowało to pojawienie się szeregu metod do jej oceny, które mogą być świetnym sposobem na poprawę aspektów czytelności i ergonomii interfejsów w tworzonych aplikacjach.

W poniższym artykule przedstawione zostaną wyniki badań dotyczących analizy wykorzystania wybranych metod oceny użyteczności w procesie tworzenia aplikacji internetowej. Metody te to: testy A/B, metoda KLM, heurystyki Nielsena oraz badania na użytkownikach. Aby sprawdzić wpływ tych metod na proces tworzenia aplikacji została stworzona gra przeglądarkowa typu multiplayer. Podczas jej tworzenia zostały wykorzystane wyżej wspomniane metody.

2. Cel badań

Celem badań było zweryfikowanie korzyści płynących z zastosowania, wspomnianych we wstępie metod, w procesie tworzenia aplikacji.

W artykule postawiono następującą tezę:

Wykorzystanie testów A/B, metody KLM, heurystyk Nielsena oraz badań na użytkownikach, w procesie tworzenia aplikacji, pozwala na znaczne zmniejszenie liczby błędów w interfejsie.

3. Przegląd literatury

Metody oceny użyteczności służą podnoszeniu jakości interfejsów graficznych. Wśród często stosowanych metod wymienia się m.in. testy A/B, Keystroke-Level Model, ocenę heurystyczną, czy badania użytkowników.

Wśród najefektywniejszych metod wymienia się badania z udziałem użytkowników. Mogą być one przeprowadzone przy niewielkich kosztach – jak wspomina Jakob Nielsen w artykule „Why You Only Need to Test with 5 Users" [1], wystarczy 5 badanych osób, aby wykryć większość problemów z interfejsem. Wnioskuje on również, że przeprowadzanie kilku mniejszych badań przynosi lepsze efekty, niż przeprowadzenie jednego większego badania. Nie wymaga ono również stosowania drogiego sprzętu - wystarczy podstawowe stanowisko komputerowe [10].

Kolejną często stosowaną metodą jest Keystroke-Level Model. Jest ona stosowana do oszacowania czasu wykonania zadania przy pomocy klawiatury i myszki [6]. Stosowanie jej polega na stworzenie dwóch wersji tego samego rozwiązania,

a następnie rozbicie ich na elementarne akcje, takie jak wciśnięcie klawisza, czy przesunięcie kursora myszki, w celu wyliczenia czasu wykonania każdego z nich [7]. Rozwiązanie z krótszym czasem wykonania uznawane jest jako lepsze. Jak wspomniano w artykułach „Using the Keystroke-Level Model to Estimate Execution Times" [2] oraz „The Keystroke-Level Model for User Performance Time with Interactive Systems" [4], rzeczywiste czasy wykonywanych akcji mogą się jednak różnić pomiędzy użytkownikami, w zależności od stopnia zaawansowania.

Wśród innych często stosowanych metod jest ocena heurystyczna. Polega ona na ocenie systemu przez ekspertów na podstawie opracowanych zasad. Jakob Nielsen w jednym ze swoich artykułów, zatytułowanym „How to Conduct a Heuristic Evaluation" [3] opisuje, że optymalnym rozwiązaniem jest analiza interfejsu przez czterech ekspertów, gdyż pozwala to na wykrycie większości problemów. Jednak taką analizę może przeprowadzić nawet jedna osoba, co pozwoli na znalezienie około 35% błędów [9].

Kolejną często wykorzystywaną metodą są testy A/B. Ich stosowanie polega na przygotowaniu dwóch wariantów produktu lub prototypu, które będą się nieznacznie różnić [5][8]. Następnie są one prezentowane użytkownikom i na podstawnie ich opinii wybierane są najlepsze rozwiązania.

4. Procedura badawcza

W celu potwierdzenia postawionej we wstępie hipotezy przeprowadzony został szereg badań na różnych etapach tworzenia aplikacji. Ze względu na istotną kolejność ich zastosowania podczas jej tworzenia, cały proces został omówiony poniżej.

1. Przeprowadzenie testów A/B dotyczących układu interfejsu aplikacji

W zaprojektowanej aplikacji można wyróżnić dwa główne widoki. Jest to widok strony przed zalogowaniem oraz widok strony po zalogowaniu. Dla każdego widoku zostały stworzone po dwa prototypy, które następnie zostały zaprezentowane 5. różnym osobom. Uczestnicy mieli za zadanie wybrać opcję, która ich zdaniem jest lepsza.

2. Stworzenie prototypu aplikacji

Dla wybranej, w poprzednim etapie, wersji układu został stworzony prototyp przedstawiający funkcjonalności aplikacji.

3. Wykorzystanie metody KLM

Po zbudowaniu prototypu gry, zostały zaprojektowane różne rozwiązania dla wybranych funkcjonalności. Następnym krokiem było wybranie zaprojektowanych rozwiązań na podstawie wyników otrzymanych po zastosowaniu metody KLM.

4. Implementacja gry

W tym miejscu przystąpiono do implementacji gry, zgodnie z projektem oraz rozwiązaniami wypracowanymi w poprzednich etapach.

5. Wykorzystanie heurystyk Nielsena

Mając finalną wersję aplikacji wykorzystano heurystki Nielsena w celu wykrycia potencjalnych błędów. Istotnym elementem na tym etapie był brak wprowadzenia poprawek dla błędów wskazywanych przez heurystyki Nielsena. Fakt ten został podyktowany chęcią ustalenia czy błędy wykryte tą metodą zostaną zauważone podczas badań na użytkownikach.

6. Badania na użytkownikach

W celu sprawdzenia ogólnego odbioru aplikacji, wykrycia ewentualnych błędów oraz zweryfikowania rozwiązań wypracowanych przy zastosowaniu poprzednich metod, zostały przeprowadzone badania na użytkownikach. Była to grupa 5 osób, która po raz pierwszy miała styczność z aplikacją wykorzystywaną do testów. Uczestnicy zostali poinformowani o typie aplikacji z jaką mają do czynienia, jednak nie zostały im wyjaśnione zasady rozgrywki. Zostało to podyktowane chęcią sprawdzenia, czy interfejs jest odpowiednio przystosowany dla nowych użytkowników. Wszyscy uczestnicy badania mieli za zadanie wykonać szereg czynności, według podanego poniżej scenariusza.

Zadanie 1

a) Zarejestruj się
b) Dodaj opis oraz zdjęcie do swojego profilu w grze
c) Zmień hasło do swojego konta

Zadanie 2

a) Sprawdź co się dzieje w budynku: Centrum
b) Rozbuduj wybrany przez siebie budynek
c) Stwórz 10 mieczników
d) Zmień nazwę swojej osady
e) Sprawdź ogólny stan posiadania jednostki typu miecznik (we wszystkich osadach łącznie)
f) Zaatakuj osadę innego gracza
g) Odczytaj raport z przeprowadzonego ataku
h) Zobacz co się dzieje w twojej drugiej osadzie

Zadanie 3

a) Sprawdź profil swojego sojuszu
b) Ustal kto jest założycielem sojuszu oraz kto ma uprawnienia administratora
c) Opuść sojusz

Zadanie 4

a) Zmień opis oraz zdjęcie swojego sojuszu
b) Nadaj uprawnienia administratora wybranemu członkowi który takich uprawnień nie posiada
c) Zaproś gracza, który jest 2-gi w rankingu ogólnym, do swojego sojuszu
d) Przyjmij jednego z graczy, który wysłał prośbę o dołączenie do twojego sojuszu
e) Rozwiąż sojusz

Zadania 2 i 3 zostały przeprowadzone na specjalnie przygotowanym koncie, na którym: gracz posiadał 2 lokacje,

należał do sojuszu jako zwykły członek, posiadał w dzienniku kilka archiwalnych raportów z bitew. Zadanie 4 zostało przeprowadzone na koncie podobnym do tego wykorzystanego przy zadaniach 2 i 3 z tą różnicą, że na tym koncie gracz miał uprawnienia administratora w sojuszu do którego należał.

7. Implementacja odrzuconych układów interfejsu

Na tym etapie zostały zaimplementowane układy interfejsu, odrzucone w testach A/B, w celu przeprowadzenia weryfikacji o której mowa w następnym punkcie.

8. Weryfikacja wyboru układu w wersji prototypowej na finalnej wersji aplikacji

Badanie to polegało na zaprezentowaniu układów aplikacji, opisanych w punkcie 1, w wersji finalnej użytkownikom, którzy brali udział w wyborze wersji prototypowej. Etap ten został przeprowadzony w celu potwierdzenia wyboru układów w wersji prototypowej.

9. Implementacja alternatywnych rozwiązań funkcjonalności

Na tym etapie zaimplementowano rozwiązania odrzucone po wykorzystaniu metody KLM.

10. Porównanie odrzuconych i zaimplementowanych rozwiązań

Etap ten został przeprowadzony w celu potwierdzenia „trafności" rozwiązań wybranych metodą KLM. Badania polegały na prezentacji obu wersji rozwiązań, grupie użytkowników, którzy brali udział w badaniach w punkcie 6.

5. Wyniki badań

5.1. Testy A/B

Wersje układów zostały oznaczone odpowiednio A i B. Zestawienie wyborów użytkowników dotyczące układu interfejsu gry, przedstawia tabela 1 oraz tabela 2.

Tabela 1. Wybory układu przed zalogowaniem

	Użytkownik					Podsumowanie
	1	2	3	4	5	
Prototyp	B	B	B	A	B	B
Wersja finalna	A	A	A	A	B	A

Tabela 2. Wybory układu gry (po zalogowaniu)

	Użytkownik					**Podsumowanie**
	1	2	3	4	5	
Prototyp	B	A	A	B	B	B
Wersja finalna	B	A	B	B	B	B

5.2. Metoda KLM

Pierwsze i drugie rozwiązanie dla wybranych funkcjonalności zostały oznaczone odpowiednio A i B. Wyniki otrzymane metodą KLM, przedstawia tabela 3. Wybory użytkowników, otrzymane podczas weryfikacji metody KLM przedstawia tabela 4. Zestawienie tych wyborów z wynikami metody KLM oraz rozwiązań zaimplementowanych na jej podstawie, prezentuje tabela 5.

Tabela 3. Wyniki metody KLM

Funkcjonalność	**Rozwiązanie**	**Czas [s]**
1	A	11,46
	B	11,46
2	A	12,62
	B	15,12
3	A	17,84
	B	12,84
4	A	5,00
	B	5,00

Tabela 4. Wybory użytkowników

Funkcjonalność	**Użytkownik**				
	1	2	3	4	5
1	B	B	B	B	B
2	B	B	A	A	A
3	B	A	A	A/B	B
4	B	B	B	B	B

Tabela 5. Wyniki badań weryfikujących metodę KLM

Funkcjonalność	**Użytkownicy**	**KLM**	**Implementacja**
1	B	A/B	A
2	A	A	A
3	A/B	B	B
4	B	A/B	A

5.3. Heurystki Nielsena

Wyniki analizy aplikacji przeprowadzone tą metodą, przedstawia tabela 6. Podczas badań na użytkownikach, żaden z nich nie zwrócił uwagi na problemy prezentowane w tabeli 6.

Tabela 6. Wyniki oceny heurystycznej

Lp.	**Problem**	**Naruszane heurystyki**
1.	Brak informacji o liczbie możliwych do utworzenia jednostek	5. Zapobiegaj błędom
2.	Zbędna informacja o niedostępnych jednostkach	8. Dbaj o estetykę i umiar
3.	Wymiana surowca na taki sam rodzaj	5. Zapobiegaj błędom
4.	Brak okna dialogowego z potwierdzeniem usunięcia członka sojuszu	5. Zapobiegaj błędom
5.	W całej aplikacji brakuje informacji, o miejscu w którym znajduję się użytkownik	1. Pokazuj status systemu

5.4. Badania na użytkownikach

Wyniki badań na użytkownikach przedstawia tabela 7 oraz tabela 8.

Tabela 7. Zestawienie pomyłek popełnianych przez użytkowników

Zadania		Użytkownicy				
		1	2	3	4	5
1	a					
	b					
	c					
2	a					
	b					
	c					
	d					
	e					
	f					
	g					
	h					
3	a					
	b					
	c					
4	a					
	b					
	c					
	d					
	e					

Legenda

	Brak pomyłek
	Wystąpiły pomyłki, ale zadanie zostało ukończone
	Liczba pomyłek, która zaowocowała pominięciem zadania

Tabela 8. Czasy wykonywania zadań przez użytkowników

Zadanie	Czas [s] wykonania zadania dla poszczególnych użytkowników			
	1	2	3	4
1	93	145	150	80
2	374	350	433	254
3	115	46	84	48
4	125	190	125	170

Podczas badania zanotowano szereg spostrzeżeń, których podsumowanie prezentuje się następująco.

Spostrzeżenia badającego:

- Użytkownicy mieli wyraźny problem z przełączeniem się do odpowiedniego widoku w grze. Głównie w zadaniu 2, w podpunktach: a, c, d, e, f i g, oraz w zadaniu 4c. Opisana sytuacja bardzo często powtarzała się dla kilku użytkowników.
- Problemy z odnalezieniem funkcjonalności w danym widoku, występowały sporadycznie. Pojawiły się one w zadaniach: 1a, 2e, 4a. Były to jednak jednorazowe przypadki.
- Wśród użytkowników często pojawiał się problem ze skojarzeniem danej strony w grze z funkcjonalnością wymaganą do ukończenia zadania. Dotyczy to głównie zadania 2d, gdzie problem powtórzył się aż u 3 użytkowników. Podobna sytuacja występowała w zadaniu 2f, gdzie opisany problem wystąpił u 2 badanych osób. Pozostałe przypadki wystąpienia problemu, to sytuacje dotyczące tylko jednej z badanych osób.

Spostrzeżenia użytkowników

Wszystkie poniżej przedstawione sugestie użytkowników to spostrzeżenia unikalne – nie powtarzające się wśród pozostałych badanych:

- W miejscu wpisywania hasła, mogłaby istnieć opcja jego chwilowego podglądu. Pozwoliłoby to szybką korektę błędnie wpisanego hasła, zamiast konieczności jego ponownego wpisywania.
- Ikony oznaczające poszczególne uprawnienia członków sojuszu mogłyby być większe.
- Opcja pozwalająca na przełączenie się do rankingu graczy mogłaby być w menu głównym.
- W widoku budynku, mógłby istnieć przycisk, przełączający do widoku w którym można rozbudowywać budynki.
- Przycisk otwierający opcję edytowania uprawnień członków sojuszu mógłby być większy oraz jego podświetlenie po najechaniu na niego powinno być wyraźniejsze.

6. Analiza wyników

6.1. Testy A/B

Przy wyborze układu strony przed zalogowaniem w wersji prototypowej, zdecydowanie wygrała opcja B (4 na 5 użytkowników) jednak przy wyborze w wersji finalnej sytuacja się odwróciła. Widząc gotową wersję 4 na 5 użytkowników wybrałoby opcję A. Ciekawy jest również fakt, że tylko 2 na 5 użytkowników podtrzymało swój wybór prototypu na wersji finalnej. Warto również dodać, że były to dwie różne wersje układu.

Przy wyborze prototypu układu gry 3 na 5 użytkowników wybrało opcję B. Wybór tej wersji potwierdził się podczas wyboru wersji finalnej, gdyż wszyscy Ci trzej użytkownicy podtrzymali swój wybór. Dodatkowo widok finalnego układu w wersji B przekonał jednego z użytkowników, który w wersji prototypowej wybrał opcję A. Tylko jeden z użytkowników pozostał przy wersji A.

6.2. Metoda KLM

Wybory użytkowników, zebrane podczas przeprowadzonego badania weryfikującego wykorzystanie metody KLM, tylko w połowie pokrywają się z tymi wybranymi dzięki metodzie KLM. Istotne jest to, że niezgodność wyboru użytkowników oraz metody KLM dotyczy rozwiązań, które według metody KLM mają taki same czasy i wybór odpowiedniego rozwiązania został dokonany przez twórców aplikacji. W niektórych przypadkach użytkownicy byli bardziej przychylni wersji która według metody KLM jest dłuższa.

Użytkownicy mieli możliwość wyboru obu rozwiązań, jeśli te odpowiadałyby im w równym stopniu. Podczas badań

tylko raz padło takie stwierdzenie. Zdecydowana większość opowiadała się po stronie jednego z rozwiązań.

6.3. Heurystyki Nielsena

Najczęściej pojawiającym się problemem, wykrytym tą metodą, jest brak zapobiegania błędom w niektórych miejscach aplikacji. Jest to istotny problem, gdyż pomimo braku jego wykrycia podczas badań na użytkownikach, może on w znaczący sposób wpływać na frustrację graczy w momencie jego wystąpienia.

6.4. Badania na użytkownikach

Badania pozwoliły na wykrycie szeregu problemów z którymi nowi użytkownicy mogą mieć problemy. Pozwoliły one również na odkrycie kilku nowych elementów, które mogłyby usprawnić interfejs aplikacji. Pomimo błędów popełnianych przez badane osoby, nie zaobserwowano żadnego elementu w interfejsie, który uniemożliwiłby wykonanie któregokolwiek z zadań wszystkim użytkownikom.

Podczas badań w części dotyczącej strony gry, najistotniejszy okazał się problem z przełączaniem do odpowiedniego widoku w grze, który zawierał funkcjonalności wymagane do ukończenia zadania. Wśród użytkowników pojawiały się również problemy ze skojarzeniem danego widoku jako miejsca zawierającego daną funkcjonalność. Problemy te w dużym stopniu były spowodowane pierwszym zetknięciem z aplikacją oraz z nieznajomości mechaniki rozgrywki. Po zakończeniu badania, użytkownikom umożliwiono swobodne zapoznanie się z grą i testowanie zastosowanych w niej mechanik. Po takim zapoznaniu większość użytkowników stwierdzało, że spora część mechanik i rozwiązań w niej zastosowanych jest spójna i logiczna, a popełniane wcześniej błędy wynikają z braku doświadczenia z mechaniką rozgrywki. Wszyscy stwierdzili, że gdyby mieliby możliwość przejścia samouczka dotyczącego rozgrywki, zadania postawione im w trakcie badań byłyby znacznie prostsze.

Podczas badań na użytkownikach, w części dotyczącej strony przed zalogowaniem, nie stwierdzono istotnych problemów. Aspekt ten może wynikać z faktu iż każda osoba biorąca udział w badaniach miała już styczność z tym elementem aplikacji internetowych.

7. Wnioski

Badania weryfikacyjne na użytkownikach pokazały, że wybór układu interfejsu na wersji prototypowej nie zawsze pokrywa się z wyborem, dokonanym przez użytkowników, na wersji finalnej. Kolejnym istotnym elementem, który wykazały przeprowadzone badania jest fakt, iż niektóre z rozwiązań wypracowanych za pomocą metody KLM, nie pokrywają się w tej kwestii z preferencjami użytkowników. Heurystyki Nielsena pozwoliły na wykrycie kilku problemów, które zawierała stworzona aplikacja, jednak wykryte problemy pozostały niezauważane przez użytkowników. Badania na użytkownikach potwierdziły iż niektóre z rozwiązań, zaimplementowane na podstawie wyników otrzymanych metodą KLM, mogą być nietrafione.

Nawiązując do tezy badawczej należy stwierdzić, że wykorzystanie wymienionych w niej metod oceny użyteczności pozwoliłoby na znaczne ograniczenie liczby błędów w interfejsie. Metody te są w stanie znacznie usprawnić proces tworzenia aplikacji oraz niwelują liczbę popełnianych błędów. Ważne jest jednak odpowiednie ich wykorzystanie. Istotne jest również stosowanie kilku wymienionych metod jednocześnie.

W testach A/B dotyczących wyboru układu strony w wersji prototypowej istotne jest jak najdokładniejsze zaprezentowanie wizji finalnej wersji tego układu. Metoda KLM pozwala na odpowiedni wybór rozwiązania tylko wtedy gdy scenariusze badanego rozwiązania znacznie się różnią. W przypadku gdy oba scenariusze są podobne lub wręcz identyczne, najlepiej jest zaprezentować dane rozwiązanie kilku użytkownikom, aby móc wybrać to które ich zdaniem jest najlepsze. Heurystyki Nielsena pozwalają ocenić spójność całego systemu oraz wykryć jego niedopracowane elementy. Metoda ta pozwala też na wykrycie elementów, których brak, może być niezauważony przez użytkowników, a których istnienie mogłoby im znacznie uprościć posługiwanie się aplikacją.

Z przeprowadzonych badań można wywnioskować, że najlepszą metodą oceny użyteczności są badania na użytkownikach, gdyż to one pozwalają na ostateczną weryfikację zastosowanych rozwiązań. To właśnie dla użytkownika jest tworzony interfejs aplikacji i to on będzie się nim posługiwał, nie powinien więc zaskakiwać fakt, że to właśnie badania na użytkownikach pozwalają na najlepszą ocenę użyteczności.

Literatura

[1] Why You Only Need to Test with 5 Users, https://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/, [29.10.2019]

[2] Using the Keystroke-Level Model to Estimate Execution Times, ftp://www.infomus.org/pub/PPM/Publications/paper-L18-KeystrokeLevelModel.pdf, [29.10.2019]

[3] How to Conduct a Heuristic Evaluation, http://www.ingenieriasimple.com/usabilidad/HeuristicEvaluation.pdf, [29.10.2019]

[4] The Keystroke-Level Model for User Performance Time with Interactive Systems, https://www.researchgate.net/profile/Stuart_Card/publication/220426122_The_Keystroke-Level_Model_for_User_Performance_Time_with_Interactive_Systems/links/09e4150e317b6ce554000000/The-Keystroke-Level-Model-for-User-Performance-Time-with-Interactive-Systems.pdf, [29.10.2019]

[5] On the Complexity of A/B Testing, http://proceedings.mlr.press/v35/kaufmann14.pdf, [29.10.2019]

[6] Kieras D., Using the Keystroke-Level Model to Estimate Execution Times, University of Michigan, 2001

[7] Sauro J., Estimating productivity: Composite operators for keystroke level modeling, Berlin Heidelberg: Springer-Verlag, 2009.

[8] An ultimate guide to A/B testing on prototypes, https://uxplanet.org/an-ultimate-guide-to-a-b-testing-on-pre-live-apps-4bd57679e8cc, [29.10.2019]

[9] Ocena heurystyczna, http://hci.pjwstk.edu.pl/index.php?page=ocena-heurystyczna, [27.10.2019]

[10] Wybrane metody oceny użyteczności stron i aplikacji internetowych, htpp://krainabiznesu.pl/wp-content/uploads/sites/2/2014/04/White_Paper-Wybrane_metody_oceny_uzytecznozci_stron_i_aplikacji_internetowych.pdf, [27.10.2019]

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 14 (2020) 94-100
Wysłane: 2019-10-19
Przyjęte: 2019-10-29

Analiza wydajności systemów bazodanowych: MySQL, MS SQL, PostgreSQL w kontekście aplikacji internetowych

Katarzyna Lachewicz*

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

Streszczenie. Głównym celem niniejszego artykułu jest weryfikacja, który z trzech systemów bazodanowych: MySQL, MS SQL, PostgeSQL, jest najbardziej wydajny w kontekście aplikacji internetowych. W artykule zawarto informacje dotyczące wykorzystanych systemów bazodanowych, jednakże kluczowym elementem artykułu są badania wydajności baz danych. Zostały one wykonane w oparciu o aplikację, której najważniejszym zadaniem jest wykonywanie zapytań do bazy danych. Program został zbudowany w oparciu o najnowsze technologie, takie jak framework Spring, biblioteka Hibernate oraz interfejs JDBC.

Słowa kluczowe: MySQL; MS SQL; PostgreSQL; wydajność baz danych

*Autor do korespondencji.

Adres e-mail: katarzyna.lachewicz@pollub.edu.pl

Performance analysis of selected database systems: MySQL, MS SQL, PostgerSQL in the context of web applications

Katarzyna Lachewicz*

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

Abstract. The main purpose of this article is to check which database: MySQL, MS SQL, PostgerSQL is the most efficient for Internet applications. This work contains information about the databases used, but the most important part of this article is database performance research. They are based on an application whose main task was database queries. The program was created based on new technologies, such as the Spring framework, the Hibernate library and JDBC Interface.

Keywords: MySQL; MS SQL; PostgreSQL; database performance

*Corresponding author.

E-mail address: katarzyna.lachewicz@pollub.edu.pl

1. Wprowadzenie

Rozpowszechnienie dostępu do Internetu umożliwiło korzystanie z ogromnych zasobów wiedzy i nowych funkcjonalności programów komputerowych. Powstała dzięki temu zupełnie nowa generacja programów, tzw. aplikacji webowych, które dzięki wykorzystaniu przeglądarki internetowej, oferowały analogiczne funkcjonalności jak aplikacje desktopowe. Zasadniczą zaletą aplikacji internetowych jest brak konieczności instalowania oprogramowania, a także dostęp do danych i funkcjonalności w dowolnym miejscu i czasie. Oprogramowanie, które wykorzystuje jako swoje środowisko Internet, stwarza możliwość przechowywania danych a także ich przetwarzania i modyfikowania [1]. Wybór takich rozwiązań wymaga gromadzenia znacznych ilości danych, co wiąże się z koniecznością stosowania różnego rodzaju bazy danych.

Wykorzystywanie baz danych w aplikacjach internetowych jest niezwykle istotne, gdyż dzięki nim, możliwa jest komunikacja serwera z użytkownikiem. Najczęściej odbywa się ona za pomocą formularzy, które pozwalają użytkownikowi dodawać oraz modyfikować dane. Ponadto możliwe jest również usuwanie i wyświetlanie odpowiednich zestawów danych. Warto zwrócić uwagę, iż na jakość aplikacji internetowych ogromny wpływ ma szybkość wykonywanych zapytań do bazy danych. Takie operacje decydują, czy użytkownik jest zadowolony z aplikacji, a co za tym idzie czy będzie z niej korzystał. Opóźnione reakcje aplikacji, zbyt długie oczekiwanie na ładowanie się strony lub wykonywanie operacji w bazie danych może zniechęcić użytkownika, co grozi zastosowaniem innej, konkurencyjnej aplikacji. Wybór odpowiedniej bazy danych nie jest prosty, gdyż należy uwzględnić szereg czynników i parametrów takich jak szybkość wykonywania zapytań, metody implementacji bardziej skomplikowanych rozwiązań. O ostatecznym wyborze konkretnego systemu bazodanowego decyduje przeznaczenie aplikacji i przyjęte priorytety przy jej projektowaniu.

Obecnie, w dobie społeczeństwa informacyjnego oraz gospodarki opartej na wiedzy, ogromną rolę odgrywają metody gromadzenia i przetwarzania narastających każdego dnia coraz większych ilości danych. Analiza zgromadzonych danych jest źródłem wiedzy i rozwoju. Wydajność systemów bazodanowych odgrywa decydującą rolę w niemal wszystkich aspektach działania aplikacji internetowych. Zaniedbania i niedociągnięcia w tym zakresie skutkują chociażby tak niepożądanym wydłużeniem czasu ładowania strony czy też opóźnieniem potwierdzenia płatności. W komercyjnych zastosowaniach czas i wydajność posiadają decydujące znaczenie, dlatego też wybór optymalnej bazy danych decyduje o sukcesie danego serwisu.

2. Cel badań

Celem artykułu jest analiza wydajności trzech wybranych systemów bazodanowych w kontekście aplikacji internetowej. Porównanie wyników analizy pozwoli określić, który system bazodanowy działa najlepiej przy tego typu aplikacjach. Badania zostaną przeprowadzone według określonych scenariuszy, które prezentują sposób wykorzystania prezentowanych technologii.

Przeprowadzone badanie będzie wykonywane przy użyciu programu umożliwiającego symulacje obciążenie serwera - JMeter. Dzięki powyższemu rozwiązaniu, możliwa będzie weryfikacja szybkości wykonywania zapytań dla poszczególnych baz danych. Przedmiotowe porównanie stanowić będzie istotną pomoc przy wyborze konkretnego systemu bazodanowego stosownie do przyjętych wcześniej priorytetów - szybkości bądź stabilności działania. Badania zostały oparte na aplikacji wykorzystującej bibliotekę Hibernate, interfejs JDBC oraz badane bazy danych.

Dla opisywanego artykułu, została sformułowana następująca hipoteza: „Systemy bazodanowe: MySQL, MS SQL i PostgreSQL, są jednakowo wydajne w kontekście aplikacji internetowych". Poprawność zaprezentowanej tezy zostanie sprawdzona na podstawie wykonanych badań.

3. Przegląd literatury

Bazy danych pełnią coraz większą rolę w życiu każdego człowieka. Z uwagi na upowszechnienie się baz danych, istotne znaczenie nabiera ich wydajność. Dlatego też literatura naukowa dotycząca systemów bazodanowych, ich zastosowania, wdrożenia, jak również wydajności - jest liczna. Poniżej zaprezentowany został wybór prac naukowych o przedmiotowej tematyce.

W artykule „Comparative analysis of NoSQL (MongoDB) with MySQL Database" [2] autorstwa Lokesh Kumar, Shalini Rajawat oraz Krati Joshi, przedstawiono sposób zastąpienia relacyjnych baz danych przy wykorzystaniu nierelacyjnych systemów – NoSQL Autorzy publikacji wykorzystali technologie takie jak MySQL oraz MongoDB, dokonując porównania ich wydajności a także łatwości zastosowania w aplikacjach z wykorzystaniem języka PHP. Podsumowując informacje zawarte w artykule należy wskazać, iż warto wykorzystywać nierelacyjne bazy danych w sytuacjach, gdy konieczne jest utworzenie aplikacji wykorzystującej bardzo duże ilości danych oraz wykonującej znaczną liczbę zapytań, takich jak wyszukiwanie czy operacje na danych.

Kolejną pracą naukową poświęconym opisywanej tematyce jest „Performance comparison for data storage - Db4o and MySQL databases" [3], której autorami są Sudhanshu Kulshrestha oraz Shelly Sachdeva. Autorzy przeprowadzili analizę porównawczą wydajności różnych systemów bazodanowych w aplikacjach dotyczących opieki zdrowotnej w Indiach. W pracy przeanalizowano trendy dotyczące systemów bazodanowych w opisywanym kraju, a następnie dokonano porównania bazy danych relacyjnej, takiej jak MySQL z obiektową – DB4o, pod względem wykorzystania przestrzeni dyskowej oraz trwałości danych. W rezultacie przeprowadzonych badań stwierdzono, że krótszy czas zapisu danych uzyskała baza danych obiektowa – DB4o, jednocześnie wykorzystując więcej przestrzeni dyskowej od relacyjnej - MySQL.

Następną omawianą publikacją naukową jest „Performance Comparison of Relational Database with Object Database (DB4o)" [4], autorstwa K. E. Roopak, K. S. Swati Rao, S. Ritesh oraz Satyadhyan Chickerur. Artykuł opisuje badania z wykorzystaniem systemów bazodanowych takich jak Db4o i MySQL, na bardzo dużej bazie danych lotniska. Omówione zostały aspekty dotyczące pisania zapytań oraz wydajność prezentowanych baz danych. W wyniku przeprowadzonych badań uznano, iż w opisywanych warunkach – dla obszernej bazy danych lotniska - bardziej wydajnym systemem bazodanowym jest obiektowy system DB4o.

W artykule „A Study on Data Input and Output Performance Comparison of MongoDB and PostgreSQL in the Big Data Environment" [5], autorzy: Min-Gyue Jung, Seon-A Youn, Jayon Bae oraz Yong-Lak Choi, porównali wydajność baz danych relacyjnych i nierelacyjnych. Analizę porównawczą dokonano w oparciu o systemy PostgreSQL i MongoDB, przy wykorzystaniu bardzo dużych ilości danych. Ponadto w artykule opisano również projekt zwiększenia funkcjonalności dla nierelacyjnych baz danych. Podsumowując, przy nieustrukturyzowanym modelu danych, wykorzystanie MongoDB pozwala na odnotowanie większej wydajności. Natomiast przy ustrukturyzowanym modelu, PostgreSQL uzyskuje wyższą wydajność.

Artykuł „Analiza wydajności relacyjnych baz danych Oracle oraz MSSQL na podstawie aplikacji desktopowej" [6], której autorami są Grzegorz Dziewit, Jakub Korczyński oraz Maria Skublewska-Paszkowska, zawiera porównanie wydajności baz danych Oracle i MS SQL w aplikacjach desktopowych. Co istotne, artykuł zawiera informacje dotyczące metodyki wykorzystywanej podczas porównywania relacyjnych systemów bazodanowych w zakresie średniego czasu wykonywania poszczególnych zapytań. Opracowany przez badaczy system badań wydajnościowych umożliwia określenie, który system jest lepszy w zależności od funkcjonalności aplikacji. W artykule udowodnione zostało, iż baza danych MS SQL lepiej wykonuje operacje INSERT oraz UPDATE, natomiast w przypadku zapytań wyszukiwania danych, krótszy czas wykonania otrzymał system Oracle.

Następny omawiany artykuł - „NoSQL real-time database performance comparison" [7], to praca autorstwa Diogo Augusto Pereira, Wagner Ourique de Morais, Edison Pignaton de Freitas. W powyższym opracowaniu dokonano porównania cech, funkcjonalności oraz wydajności trzech nierelacyjnych systemów bazodanowych: Couchbase, MongoDB oraz RethinkDB. Testy obejmowały zarówno jedno jaki i wielowątkowe zapytania. Zaprezentowane wyniki wskazują, że najlepszą wydajność uzyskał Couchbase, który zdobył najlepsze wyniki w większości przeprowadzonych testów.

Problematyki wydajności baz danych dotyczy również praca „A Comparison of Performance Between MSSQL Server and MongoDB for Telco Subscriber Data Management" [8], przygotowana przez Aaron Nichie,

Heung - Seo Koo. W publikacji ujęto porównanie wydajności bazy danych MS SQL i MongoDB, w kontekście wydajności w różnych fazach przepływu danych. W rezultacie prowadzonych badań stwierdzono, że baza MongoDB posiadała lepszą wydajność względem MS SQL.

Publikacja „Alternatives to relational database: Comparison of NoSQL and XML approaches for clinical data storage" [9], opracowana została przez Author Ken Ka-Yin Lee, Wai-Choi Tang, Kup-Sze Choi. W powyższej pracy ukazano badania naukowe bazujące na bazach danych NoSQL i XML, w celu zweryfikowania ich przydatności w przypadku danych klinicznych ustrukturyzowanych. Wykonane badania wskazują, iż baza nierelacyjna posiada lepsze wyniki pod względem szybkości wykonywania zapytań, natomiast baza XML wygrywa pod względem skalowalności, rozszerzalności i elastyczności.

Ostatnim omówionym artkułem jest „Performance Analysis for NoSQL and SQL" [10], którego autorami są Megha Katkar, Shah Kutchhi i Anchor Kutchhi. W powyższej pracy naukowej opisano badanie wydajności baz SQL i NoSQL. Wyniki przedstawione w artykule wskazują, iż spośród wszystkich systemów SQL i NoSQL najlepsze wyniki uzyskała baza danych nierelacyjna – FoundationDB. Dodatkowo autorzy zauważyli, że dla wszystkich systemów wyniki wydajności różnią się w zależności od typu operacji jaka miała być wykonana.

Wszystkie przedstawione powyżej artykuły naukowe odnoszą się do wydajności różnorodnych systemów bazodanowych, co wprost wskazuje na znaczne zainteresowania badaczy tą problematyką. Kwestie te są niezmiernie istotne, ponieważ obecnie niemal wszystkie aplikacje wykorzystują bazy danych. W konsekwencji ich wydajność jest kluczowym elementem do budowy aplikacji spełniającej wysokie wymagania użytkowania.

4. Opis obiektu badań

4.1. MySQL

MySQL to system bazodanowy rozwijany przez firmę Oracle, przeznaczony do zarządzania relacyjnymi bazami danych. Umożliwia przechowywanie, wyszukiwanie a także sortowanie i odczytywanie obszernych zbiorów danych. Co istotne, wskazane czynności może wykonywać nie jeden a wielu użytkowników jednocześnie - tzw. wielodostępność. Tą istotną cechę uzyskano dzięki zastosowaniu systemu zapytań, korzystającego z popularnego standardu języka SQL. Istnieje wiele rozszerzeń tego języka dodających nowe elementy i funkcjonalności. W konsekwencji wraz z rozwojem języka możliwe jest tworzenie coraz bardziej skomplikowanych zapytań.

Początki systemu MySQL datuje się na 1979 r. [11], jednakże publiczne udostępnienie nastąpiło w 1996 r. Oprogramowanie jest ciągle udoskonalane. Najnowsza wersja systemu została wydana 25 kwietnia 2019 roku.

4.2. MS SQL

MS SQL, czyli Microsoft SQL Server to system zarządzania bazami danych, rozpowszechniony przez Microsoft. Jego zasadniczą zaletą jest rozszerzenie języka SQL – Transact-SQL. MS SQL jest systemem bazodanowym typu klient – serwer. W 1989 roku została wydana pierwsza wersja systemu - Ashton Tate/Microsoft SQL Server 1.0 [12]. Nad projektem pracowała firma Micorsoft, Sybase oraz w późniejszym czasie Ashton Tate.

Najważniejszym elementem systemu jest silnik baz danych, którego zadaniem jest przechowywanie i przetwarzanie danych, a następnie zarządzanie nimi z wykorzystaniem modelu relacyjnego. W tym przypadku wykorzystywany jest system zarządzania danymi XML. Silnik bazodanowy umożliwia synchronizację i spójność danych [12].

Microsoft SQL Server umożliwia wykorzystanie operacji zgodnych z językiem zapytań SQL. Pozwala to na wykonywanie podstawowych operacji na bazach danych takich jak dodawanie nowych rekordów, wyświetlanie, edytowanie danych oraz ich usuwanie. Dodatkowo opisywany system umożliwia wykorzystanie bardziej skomplikowanych zapytań, dzięki wykorzystaniu Transact-SQL. Język ten rozszerza język SQL, dając programiście możliwość wykonywania bardziej skomplikowanych i złożonych instrukcji [12].

4.3. PostgreSQL

PostgreSQL to jeden z najpopularniejszych systemów bazodanowych umożliwiający zarządzanie relacyjnymi bazami danych. Prace rozpoczęto w 1977 roku w Berkeley na Uniwersytecie Kalifornijskim, a wstępna nazwa tego projektu to Ingres. W 1986 roku opublikowana została pierwsza wersja systemu. Po kilku latach zyskał on ogromną popularność, co umożliwiło dalsze rozwinięcie projektu.

Opisywana baza danych jest ciągle usprawniana, dzięki czemu stała się jednym z najbardziej funkcjonalnych, a zarazem niezawodnych systemów relacyjnych baz danych [13]. Opisywany system jest wciąż aktualizowany, co można zauważyć patrząc na najnowszą wersję PostgreSQL 11.5 wydaną w 8 sierpnia 2019 roku [14].

Jedną z najważniejszych cech PostgreSQL jest jego wielozadaniowość. Ponadto w razie konieczności wykonania niestandardowych procedur, możliwe jest programowe rozszerzanie za pomocą procedur przechowywanych [15].

5. Opis przebiegu badań

5.1. Opis aplikacji

Głównym celem artykułu jest sprawdzenie wydajności trzech systemów bazodanowych: MySQL, MS SQL oraz PostgreSQL. W konsekwencji kluczowe jest połączenie aplikacji z powyższymi bazami. Aplikacja składa się głównie z funkcji wykonywanych na bazach, dlatego też oprócz prostych instrukcji, dodane zostały również zapytania bardziej skomplikowane, dostosowane do celu badań.

Program został zbudowany w języku Java wykorzystując framework Spring, co umożliwia łatwiejsze tworzenie złożonych projektów aplikacji internetowych w języku Java. Dodatkowo wykorzystano bibliotekę Hibernate ułatwiającą dostęp do danych, poprzez zapewnienie odpowiedniej komunikacji między systemem bazodanowym a aplikacją

oraz interfejs JDBC pozwalający również na wykonywanie zapytań do bazy danych. Ponadto do badań wykorzystano narzędzia umożliwiające pracę z bazami danych jak phpMyAdmin, który umożliwia zarządzanie bazą danych MySQL. Dla MS SQL zastosowano Microsoft SQL Server Management Studio, a w przypadku PostgreSQL – pgAdmin.

W aplikacji zostały zrealizowane zapytania do bazy danych, takie jak wprowadzanie danych, ich wyświetlanie, usuwanie oraz edytowanie. Dodatkowo wykonane zostały bardziej skomplikowane zapytania, mające na celu sprawdzenie wydajności baz danych względem wykorzystanych komend SQL-owych. Ponadto zostały przetestowane różne rodzaje powiązania między tabelami a także wydajność baz danych z wykorzystaniem podzapytania.

5.2. Opis metody badawczej

W celu wykonania jak najdokładniejszych badań wydajnościowych baz danych, wybór scenariuszy testowych został wybrany w taki sposób, aby przetestowane zostały wszystkie podstawowe funkcjonalności bazodanowe wykorzystywane w aplikacjach internetowych. Wybrane operacje to między innymi wyświetlanie danych, dodawanie i edycja pojedynczych rekordów oraz usuwanie wybranych wiersz.

Tabela 1. Opis scenariuszy badania dla operacji na pojedynczych tabelach

Opis scenariusza	Tabela bazy danych	Liczba rekordów w tabeli	Powiązanie z innymi tabelami?
Wyszukiwanie pojedynczych rekordów	Wypożyczenia	6000	brak
Wyświetlanie zbioru danych	Książki	1000	brak
Wyświetlanie zbioru danych z komendą „is null”	Książki	1000	brak
Dodawanie pojedynczych rekordów	Wypożyczenia	6000	brak
Usuwanie pojedynczych rekordów	Wypożyczenia	6000	brak
Edycja danych wybranego rekordu	Książki	1000	brak

Wszystkie scenariusze zostały wykonane dla pojedynczych jak i dla powiązanych ze sobą tabel. Dlatego też scenariusze podzielono na dwie serie. Pierwsza dotyczy pojedynczej tabeli, natomiast druga powiązanych encji. Dodatkowo, warto zwrócić uwagę, iż wszystkie opisane poniżej scenariusze zostały wykorzystane przy badaniach w aplikacji internetowej z biblioteką Hibernate oraz interfejsem JDBS, a także w przypadku testów wykonanych bezpośrednio na bazie danych.

Weryfikacja wydajności bazy danych przy wykorzystaniu scenariuszy badań z obu serii pozwoli na określenie, które systemy są optymalne dla różnej złożoności zapytań. W tabelach 1 i 2 zawarto takie informację jak: numer i opis scenariusza, tabele bazy danych jakie wykorzystywane zostały przy zapytaniu, liczba rekordów w każdej tabeli oraz informacja czy wykorzystane zostało powiązanie między encjami.

Tabela 2. Opis scenariuszy badania dla operacji na powiązanych tabelach

Opis scenariusza	Tabela bazy danych	Liczba rekordów w tabeli	Powiązanie z innymi tabelami?
Wyszukiwanie z dwóch powiązanych tabel	Książki	1000	tak
	Kategorie	30	
Wyszukiwanie z czterech powiązanych tabel	Użytkownicy	500	tak
	Kategorie	30	
	Książki	1000	
	Wypożyczenia	6000	
Wyszukiwanie z podzapytaniem	Użytkownicy	500	tak
	Wydawnictwo	300	
	Książki	1000	
	Wypożyczenia	6000	
Wyszukiwanie z warunkiem i podzapytaniem zawierającym powiązane tabele	Użytkownicy	500	tak
	Kategorie	30	
	Książki	1000	
	Wypożyczenia	6000	
	Autorzy	300	
Wyszukiwanie z innym typem złączenia tabel	Wypożyczenia	6000	tak
	Książki	1000	
	Kategorie	30	
Wyszukiwanie z komendą „is null”	Książki	1000	tak
	Kategorie	30	

W przypadku badania wydajności aplikacji internetowej wykorzystano kryterium, według którego dla każdego scenariusza obliczony został średni czas wykonania danego zapytania oraz najkrótszy i najdłuższy czas odnotowany w badaniu. Ponadto dla każdego scenariusza wyliczono odchylenie standardowe oraz medianę.

5.3. Narzędzie wykorzystane do testowania

Do przeprowadzenia badań wydajnościowych wykorzystany został program Apache JMeter. Służy on do wykonywania testów funkcyjnych oraz obciążających, pozwalających określić maksymalną wielkość ruchu, który jest w stanie obsłużyć dany serwer. Program został zaprojektowany przez Stefano Mazzocchi z Apache Software Foundation w celu testowania wydajności Apache Jserv [16]. Następnie narzędzie zostało przeprojektowane oraz wyposażone w interfejs graficzny, co umożliwiło stosowanie go do testów funkcyjnych.

JMeter pozwala na przetestowanie zasobów takich jak pliki, bazy danych, serwery FTP oraz języki umożliwiające programowania aplikacji internetowych: Java, PHP, ASP.NET [16]. Umożliwia również symulowanie ruchu na serwerze oraz analizowanie wydajności pod wpływem obciążenia.

6. Prezentacja rezultatów badań

W celu wykonania jak najdokładniejszych badań, każdy scenariusz przeprowadzony z wykorzystaniem aplikacji internetowej, został powtórzony 1000 razy. Dzięki temu możliwe było uzyskanie bardziej precyzyjnych średnich czasów wykonania zapytania.

6.1. Badania na pojedynczych tabelach z wykorzystaniem aplikacji z biblioteką Hibernate

W tabeli 3 zestawiono wyniki średnich czasów dla poszczególnych scenariuszy. Każde badanie zostało wykonane dla trzech systemów bazodanowych: MySQL, MS SQL i PostgreSQL.

Tabela 3. Wyniki średnich czasów dla poszczególnych scenariuszy w przypadku testów na pojedynczej tabeli

Scenariusz	MySQL	MS SQL	PostgreSQL
Wyszukiwanie pojedynczych rekordów	5,2 ms	5,2 ms	4,7 ms
Wyszukiwania zbioru danych	5,3 ms	5,7 ms	6,5 ms
Wyświetlanie zbioru danych z komendą "is null"	6,2 ms	6,0 ms	5,8 ms
Dodawanie pojedynczych rekordów	8,3 ms	8,4 ms	7,0 ms
Usuwanie pojedynczych rekordów	6,3 ms	6,9 ms	6,8 ms
Edycja danych wybranego rekordu	7,3 ms	7,4 ms	7,7 ms

Dane z tabeli 3 wskazują, iż w przypadku prostych zapytań na bazie danych, czyli takich, które wykonywane są na pojedynczej tabeli, wyniki są bardzo zbliżone dla zaprezentowanych baz danych. MySQL uzyskał najniższe wyniki w przypadku wyszukiwania pojedynczych rekordów, zbiorów danych oraz usuwania i edytowania pojedynczych danych. Natomiast system bazodanowy PostgreSQL zdobył najlepszy wynik dla scenariusza wyświetlania zbioru danych z wykorzystaniem komendy SQL „is null" oraz dla dodawania pojedynczych rekordów

Warto nadmienić, iż dane przedstawione dla pojedynczej tabeli są bardzo zbliżone dla wszystkich systemów bazodanowych. Dodatkowo, średnie czasy uzyskane dla przedstawionych scenariuszy nie przekraczają 8,5 ms, co oznacza optymalną odpowiedź serwisu bazodanowego.

6.2. Badania na tabelach powiązanych z wykorzystaniem aplikacji z biblioteką Hibernate

Kolejny zestaw badań polega na wykonywaniu zapytań wyszukiwania danych z wykorzystaniem powiązania tabel w aplikacji z wykorzystaniem biblioteki Hibernate. W tym przypadku sprawdzona została wydajność baz danych przy wykorzystaniu zapytań zawierających instrukcje warunkowe, podzapytania oraz powiązanie z większą ilością tabel. Wyniki dla wszystkich scenariuszy zostały zaprezentowane w tabeli 4.

Najdłuższe średnie czasy wykonywania zapytań zostały odnotowane w przypadku scenariusza wyszukiwania danych z czterech powiązanych tabel. W tym przypadku baza danych MySQL uzyskała wynik 114,3 ms, MS SQL – 21,0 ms a PostgreSQL 16,5 ms. Z powyższego zestawienia wynika, iż najlepiej poradziła sobie baza danych PostgreSQL.

Natomiast najkrótsze czasy odnotowane zostały dla wyszukiwania danych z instrukcją warunkową „is null". W przypadku bazy MySQL wyliczony został średni czas 7,4 ms, dla MS SQL – 5,5 ms, a PostgreSQL – 4,2 ms. W tym przypadku również system bazodanowy PostgreSQL uzyskał najlepsze wyniki, jednakże różnice względem pozostałych baz danych są niewielkie.

Tabela 4. Wyniki średnich czasów dla poszczególnych scenariuszy w przypadku testów na tabelach powiązanych

Scenariusz	MySQL	MS SQL	PostgreSQL
Wyszukiwanie z dwóch powiązanych tabel	13,3 ms	6,4 ms	6,1 ms
Wyszukiwanie z czterech powiązanych tabel	114,3 ms	21,0 ms	16,5 ms
Wyszukiwanie z podzapytaniem	51,5 ms	19,8 ms	15,3 ms
Wyszukiwanie z warunkiem i podzapytaniem zawierającym powiązane tabele	86,1 ms	12,6 ms	7,4 ms
Wyszukiwanie z innym typem złączenia tabel	45,8 ms	6,2 ms	4,7 ms
Wyszukiwanie w komendą „is null"	7,4 ms	5,5 ms	4,2 ms

Warto zwrócić uwagę, iż w przypadku wszystkich scenariuszy z serii badań wykonanych na powiązanych tabelach, najkrótsze średnie czasy uzyskała baza danych PostgreSQL. Dodatkowo warto nadmienić, iż w tym przypadku odnotowano wszystkie średnie czasy dla tego systemu bazodanowego poniżej 17 ms, co jest optymalnym wynikiem.

W zaprezentowanych badaniach w tabeli 4 najgorzej poradziła sobie baza danych MySQL, której średnie czasy dochodzą do wartości 114,3 ms. Wyniki te są dużo wyższe w porównaniu do pozostałych baz danych. W przypadku systemu MS SQL, najdłuższy średni czas to 21,0 ms, co oznacza, iż opisywana baza również uzyskuje lepsze wyniki niż MySQL.

Podsumowując, z badaniami opartymi na powiązanych tabelach najlepiej poradził sobie system bazodanowy PostgreSQL, który uzyskał najniższe wyniki we wszystkich testach opisanych w tabeli 4.

6.3. Badania na pojedynczych tabelach z wykorzystaniem aplikacji z interfejsem JDBC

Badania zaprezentowane w tabeli 5 przedstawiają wyniki zapytań wykonanych na pojedynczej tabeli z wykorzystaniem aplikacji internetowej z interfejsem JDBC.

Tabela 5. Wyniki średnich czasów dla poszczególnych scenariuszy w przypadku testów na pojedynczej tabeli

Scenariusz	MySQL	MS SQL	PostgreSQL
Wyszukiwanie pojedynczych rekordów	4,3 ms	4,7 ms	3,8 ms
Wyszukiwania zbioru danych	3,8 ms	4,1 ms	3,3 ms
Wyświetlanie zbioru danych z komendą „is null"	3,6 ms	3,8 ms	3,0 ms
Dodawanie pojedynczych rekordów	3,8 ms	5,2 ms	4,4 ms
Usuwanie pojedynczych rekordów	3,2 ms	5,4 ms	2,9 ms
Edycja danych wybranego rekordu	3,3 ms	4,8 ms	3,9 ms

Analizując zaprezentowane dane można zauważyć, iż baza PostgreSQL najlepiej poradziła sobie w przypadku wyszukiwania pojedynczych rekordów, zbiorów danych a także wyszukiwania z wykorzystaniem komendy „is null" oraz usuwaniem pojedynczych rekordów. Natomiast w przypadku dodawania i edycji pojedynczych danych, najkrótsze średnie czasy odnotowano dla systemu MySQL.

Wyniki zaprezentowane w tabeli 5 dla wszystkich systemów są bardzo zbliżone i nie przekraczają 5,2 ms. Jednak warto zwrócić uwagę, iż najdłuższe średnie czasy we wszystkich scenariuszach uzyskała baza MS SQL.

6.4. Badania na tabelach powiązanych z wykorzystaniem aplikacji z interfejsem JDBC

W tabeli 6 zaprezentowane zostały wyniki badania wykonane na aplikacji z interfejsem JDBC dla tabel powiązanych.

Baza danych, która uzyskała najkrótsze średnie czasy dla wszystkich scenariuszy to PostgreSQL. Czasy wykonania zapytań mieszczą się w zakresie od 2,9 ms do 7,2 ms. Kolejną bazą, która uzyskała wyniki od 4,1 ms do 12,8 ms to MS SQL. Natomiast najgorzej poradziła sobie baza MySQL z wynikami od 4,3 ms do 81,1 ms.

Tabela 6. Wyniki średnich czasów dla poszczególnych scenariuszy w przypadku testów na tabelach powiązanych

Scenariusz	MySQL	MS SQL	PostgreSQL
Wyszukiwanie z dwóch powiązanych tabel	8,4 ms	4,8 ms	3,3 ms
Wyszukiwanie z czterech powiązanych tabel	81,1 ms	12,8 ms	6,6 ms
Wyszukiwanie z podzapytaniem	38,3 ms	12,4 ms	7,2 ms
Wyszukiwanie z warunkiem i podzapytaniem zawierającym powiązane tabele	68,4 ms	11,2 ms	4,9 ms
Wyszukiwanie z innym typem złączenia tabel	47,2 ms	6,0 ms	4,3 ms
Wyszukiwanie w komendą „is null"	4,4 ms	4,1 ms	2,9 ms

Zapytania, które uzyskały najdłuższe średnie czasy realizacji to wyszukiwanie z czterech powiązanych tabel oraz wyszukiwanie z podzapytaniem. Warto również zauważyć, iż wyniki w opisanym badaniu są gorsze niż w przypadku wykorzystania Hibernate. Oznacza to, iż wykorzystanie tej biblioteki powoduje wykonywanie dodatkowych operacji w tle, a tym samym zmniejszana jest wydajność systemu.

6.5. Badania na tabelach powiązanych wykonane bezpośrednio na bazie danych

Tabela 7 zawiera wyniki badań wykonanych bezpośrednio na bazach danych, na których dodatkowo zastosowano czyszczenie pamięci cache. Testy wykonane zostały z wykorzystaniem scenariuszy jak w poprzednich punktach. Każde badanie zostało powtórzone 10 razy.

Po analizie uzyskanych danych można stwierdzić, iż po wyczyszczeniu pamięci cache najdłuższe średnie czasy uzyskała baza PostgreSQL. Wyniki wahają się od 65,9 ms do 283,5 ms. Natomiast najkrótsze czasy w większości zapytań uzyskała baza MS SQL z wynikami od 10,6 ms do 41,7 ms.

Tabela 7. Wyniki średnich czasów dla poszczególnych scenariuszy w przypadku testów na pojedynczej tabeli

Scenariusz	MySQL	MS SQL	PostgreSQL
Wyszukiwanie pojedynczych rekordów	18 ms	15,3 ms	261,5 ms
Wyszukiwania zbioru danych	12,1 ms	41,7 ms	283,5 ms
Wyświetlanie zbioru danych z komendą „is null"	18,2 ms	18,1 ms	258,2 ms
Dodawanie pojedynczych rekordów	56,5 ms	10,6 ms	74,3 ms
Usuwanie pojedynczych rekordów	34,1 ms	15,3 ms	80,3 ms
Edycja danych wybranego rekordu	40,5 ms	15,6 ms	65,9 ms

6.6. Badania dla tabel powiązanych wykonane bezpośrednio na bazie danych

Wyniki badań jakie uzyskano dla bazy danych w przypadku powiązanych encji zawarte zostały w tabeli 8. Wskazują one, iż najkrótsze średnie czasy dla wszystkich zapytań uzyskała baza MS SQL z wynikami od 15 ms do 76,1 ms. System bazodanowy MySQL uzyskał lepsze wyniki od PostgreSQL w przypadku zapytań wyszukiwania z dwóch powiązanych tabel, z wykorzystaniem podzapytania oraz z zastosowaniem komendy „is null". Natomiast dla PostgreSQL odnotowano niższe wyniki dla wyszukiwania z wykorzystaniem czterech powiązanych tabel, z zastosowaniem innych typów złączeń oraz wyszukiwanie z podzapytaniem i powiązanymi encji.

Tabela 8. Wyniki średnich czasów dla poszczególnych scenariuszy w przypadku testów na tabelach powiązanych

Scenariusz	MySQL	MS SQL	PostgreSQL
Wyszukiwanie z dwóch powiązanych tabel	114,6 ms	47,7 ms	262,7 ms
Wyszukiwanie z czterech powiązanych tabel	602,8 ms	39,9 ms	264,7 ms
Wyszukiwanie z podzapytaniem	285,6 ms	34,3 ms	302 ms
Wyszukiwanie z warunkiem i podzapytaniem zawierającym powiązane tabele	480,8 ms	76,1 ms	306,3 ms
Wyszukiwanie z innym typem złączenia tabel	443,7 ms	27,5 ms	305,4 ms
Wyszukiwanie w komendą „is null"	35,1 ms	19,2 ms	304,9 ms

Na podstawie wyników przeprowadzonych badań należy stwierdzić, że po oczyszczeniu pamięci ceche, najbardziej wydajnym systemem bazodanowym jest MS SQL. Warto zwrócić również uwagę, iż MySQL radzi sobie lepiej w porównaniu z PostgreSQL w przypadku prostszych zapytań, natomiast baza danych PostgreSQL w bardziej złożonych.

7. Wnioski

Głównym celem niniejszego artykułu było przeprowadzenie badań wydajności trzech systemów

bazodanowych a także określenie, który z nich najlepiej wykorzystać podczas budowy aplikacji internetowych.

Analizując dane zaprezentowane w rozdziale „Badania", należy wskazać, iż w przypadku aplikacji internetowych, najbardziej wydajnym systemem bazodanowym jest PostgreSQL. Przedmiotowa baza danych zdecydowanie lepiej radzi sobie z bardziej skomplikowanymi zapytaniami, które wykorzystują powiązanie tabel, zarówno dla aplikacji wykorzystującej bibliotekę Hibernate jak i interfejs JDBC. Warto również nadmienić, iż najdłuższe średnie czasy realizacji zapytań jakie uzyskano dla tej bazy danych to 16,5 ms, przy czym większość zapytań była realizowana w czasie poniżej 8 ms. Wyniki te są dużo korzystniejsze niż w przypadku pozostałych systemów bazodanowych. Natomiast najgorzej z przedstawionych baz danych poradził sobie MySQL, którego średnie czasy wykonywania zapytań dochodziły do 114,3 ms. W przypadku zapytań wykorzystujących pojedynczą tabele, średnie czasy opisywanych systemów są bardzo do siebie zbliżone. Warto jednak zwrócić uwagę, iż w przypadku aplikacji internetowych często wykorzystywane są bardziej skomplikowane zapytania. Dodatkowo w artykule udowodniono, iż lepszą wydajność systemów bazodanowych uzyskała aplikacja, która wykorzystała interfejs JDBC. Średnie czasy dla wszystkich zapytań były niższe, od tych z zastosowaniem biblioteki Hibernate.

Ponadto, wykonane zostały badania na bazach danych z wykorzystaniem czyszczenia pamięci cache, w celu zweryfikowania, który z systemów bazodanowych jest najbardziej wydajny. W wyniku badań wykonanych bezpośrednio na bazach danych, optymalnym systemem okazał się MS SQL. Otrzymany rezultat wskazuje na to, iż opisana baza danych w mniejszym stopniu bazuje na danych przechowywanych w pamięci podręcznej, w porównaniu do pozostałych. Takie rozwiązanie wynika ze sposobu optymalizacji zapytań jaki został wykorzystany przy budowie wybranego systemu bazodanowego.

Wydajność baz danych jest niezwykle istotnym aspektem podczas tworzenia programu, ponieważ to ona decyduje o szybkości działania aplikacji, co w konsekwencji przekłada się na zadowolenie klientów. Badania przeprowadzone w ramach artykułu, mają na celu weryfikację wydajności kilku najbardziej popularnych systemów bazodanowych pod kątem zastosowań w aplikacji internetowych.

Teza, która została postawiona w artykule: „Systemy bazodanowe: MySQL, MS SQL i PostgreSQL, są jednakowo wydajne w kontekście aplikacji internetowych" została obalona. W wyniku analizy badań przeprowadzonych na aplikacji z wykorzystaniem biblioteki Hibernate oraz interfejsu JDBC, najbardziej wydajnym systemem okazał się PostgreSQL.

Odpowiedni dobór technologii znacznie usprawnia działanie programu, co z kolei może być kluczowe dla klientów i tym samym przesądzić o powodzeniu biznesowym aplikacji. W przypadku aplikacji internetowych, najbardziej wydajnym systemem bazodanowym okazał się PostgreSQL. System ten sprawnie i efektywnie radzi sobie z zarówno prostszymi jak i bardziej rozbudowanymi zapytaniami do bazy danych, co wskazuje na jego szczególną przydatność w zastosowaniach związanych z tworzeniem aplikacji internetowych.

Literatura

[1] Marek Miłosz (red.): Aplikacje internetowe – od teorii do praktyki, 2018.

[2] Lokesh Kumar, dr. Shalini Rajawat, Krati Joshi: Comparative analysis of NoSQL (MongoDB) with MySQL Database, 2015.

[3] Sudhanshu Kulshrestha, Shelly Sachdeva, Performance comparison for data storage - Db4o and MySQL databases, 2014.

[4] Roopak K.E., Swati Rao K.S., Ritesh S., Satyadhyan: Performance Comparison of Relational Database with Object Database (DB4o), 2013

[5] Min-Gyue Jung, Seon-A Youn, Jayon Bae, Yong-Lak Choi: A Study on Data Input and Output Performance Comparison of MongoDB and PostgreSQL in the Big Data Environment, 2015.

[6] Grzegorz Dziewit, Jakub Korczyński, Maria Skublewska-Paszkowska: Analiza wydajności relacyjnych baz danych Oracle oraz MSSQL na podstawie aplikacji desktopowe, 2018.

[7] Diogo Augusto Pereira, Wagner Ourique de Morais, Edison Pignaton de Freitas: NoSQL real-time database performance comparison, 2017.

[8] Aaron Nichie, Heung-Seo Koo: A Comparison of Performance Between MSSQL Server and MongoDB for Telco Subscriber Data Management, 2016.

[9] Ken Ka-Yin Lee, Wai-Choi Tang, Kup-Sze Choi: Alternatives to relational database: Comparison of NoSQL and XML approaches for clinical data storage, 2013.

[10] Megha Katkar, Shah and Anchor Kutchhi: Performance Analysis for NoSQL and SQL, 2015.

[11] MySQL, http://vavatech.pl/technologie/bazy-danych/mysql [30.08.2019].

[12] Adam Pelikant, MS SQL Server. Zaawansowane metody programowania, 2014.

[13] PostgreSQL, http://vavatech.pl/technologie/bazy-danych/postgresql [30.08.2019].

[14] PostgreSQL 12 Released!, https://www.postgresql.org/docs/11/release-11-5.html [30.09.2019].

[15] Porównanie relacyjnych SZBD: SQLite, MySQL, PostgreSQL, https://hostovita.pl/blog/porownanie-relacyjnych-systemow-zarzadzania-bazami-danych-sqlite-mysql-postgresql/ [30.08.2019].

[16] JMeter – narzędzie testera, http://2016.testwarez.pl/jmeter-narzedzie-testera/ [10.09.2019].

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 14 (2020) 101-108
Wysłane: 2019-10-21
Przyjęte: 2019-10-28

Metody wytwarzania realistycznych pomieszczeń – skanowanie 3D oraz modelowanie 3D

Aleksandra Salwierz*, Tomasz Szymczyk
Politechnika Lubelska, Katedra Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. Artykuł przedstawia dwie współczesne metody wytwarzania realistycznych pomieszczeń 3D. Porównanie obejmuje skanowanie 3D za pomocą skanera FARO Focus 3D X330 oraz modelowanie 3D w programie Blender 2.8. Analiza sposobów generowania realistycznych pomieszczeń może okazać się przydatna w wielu dziedzinach np.: w architekturze, druku 3D, grach, wizualizacjach, kryminalistyce, inżynierii wstecznej czy dokumentacji zabytków. Artykuł opisuje proces generowania wybranego pomieszczenia dla każdej z metod. Dokonano porównania obu rozwiązań pod kątem kosztów, dokładności oraz stopnia oddania rzeczywistości. Dodatkowo opisano napotkane problemy oraz wskazano ich możliwe źródła. Oceniono stopień przydatności oraz opłacalności obu metod. Przeprowadzono również badania dotyczące stopnia immersyjności wizualizacji poszczególnych metod w wirtualnej rzeczywistości.

Słowa kluczowe: skanowanie; modelowanie; 3D; architektura

*Autor do korespondencji.

Adres e-mail: aleksandra.salwierz@pollub.edu.pl

Methods of creating realistic spaces – 3D scanning and 3D modelling

Aleksandra Salwierz*, Tomasz Szymczyk
Department of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract. Article shows two modern methods of creating realistic 3D spaces. The comparison includes 3D scanning with FARO Focus 3D X330 and 3D modelling in Blender 2.8. Analysis of methods for creating realistic 3D spaces can be useful in many fields e.g.: architecture, 3D printing, games industry, visualization, criminalistics, reverse engineering or monument documentation. The paper also describes process of generating a chosen space for each method. Each of the two approaches is assessed in terms of the expenses, precision and degree of reflecting reality.. Article includes an analysis of encountered problems and their possible sources. The paper also evaluate usefulness and profitability for each method. A research was carried out and focused on degree of immersion for VR visualizations depending on the used method.

Keywords: scanning; modelling; 3D; architecture

*Corresponding author.

E-mail address: aleksandra.salwierz@pollub.edu.pl

1. Wstęp

Współczesna grafika 3D znajduje zastosowanie w wielu dziedzinach np.: w architekturze, dokumentacji zabytków, zwiedzaniu miejsc [1], druku 3D, grach, wizualizacjach, kryminalistyce, przemyśle, medycynie [2], edukacji [3, 4] czy inżynierii odwrotnej. Ciągły rozwój technologiczny sprawił, że pojawia się coraz wydajniejsze oprogramowanie oraz sprzęt pozwalający na efektowniejsze tworzenie modeli. W przeciwieństwie do tradycyjnych metod generowania modeli 3D coraz częściej mówi się o skanowaniu 3D.

Wartość światowego rynku skanerów 3D zwiększa się w bardzo szybkim i trudnym do przewidzenia tempie. Według prognoz od Allied Market Research z lat 2013-2020 [5] miał on rokrocznie rosnąć o ponad 12%, aby ostatecznie osiągnąć wartość 5 mld USD. Obecnie to samo źródło podaje, że w roku 2017 wartość rynku skanerów 3D wyniosła 8 mld USD, co bardzo daleko wykracza poza początkowe prognozy. Aktualnie Allied Market Research przewiduje średni coroczny wzrost o 26% aż do 53 mld USD w 2025 [6]. Dodatkowo warto wspomnieć, że 2/3 odbiorców skanerów 3D znajduje się w Ameryce Północnej (40%) oraz Europie. Skanowanie 3D zaczyna pojawiać się jako alternatywa dla żmudnego i czasochłonnego procesu modelowania. Niestety potrafi być ono bardzo drogie.

Mimo wszystko rynkowi skanowania 3D sporo brakuje, żeby osiągnąć wartości porównywalne do rynku mapowania i modelowania 3D. Według Mordor Intelligence jego wartość w 2018 została wyceniona na prawie 9 bilionów USD z przewidywanym wzrostem rocznym wynoszącym ponad 20% [7].

Przyglądając się porównywanym procesom już na wstępie można zauważyć ogromne różnice dotyczące kosztów, jakie należy ponieść przy rozpoczynaniu pracy w danej branży. Skanowanie 3D wymaga znacznych nakładów finansowych na sprzęt i oprogramowanie, a dostępność dodatkowych szkoleń czy kursów jest o wiele mniejsza. Modelowanie 3D oferuje szeroki wachlarz rozwiązań i programów, duża część z nich jest darmowa. Proces nauki tej technologii jest bardzo ułatwiony dzięki bezpłatnym materiałom udostępnionym w Internecie, a także dużej ilości specjalistycznych kursów o różnym poziomie zaawansowania.

2. Technologie oraz narzędzia

Na Rysunku 1 przedstawiono wybrano pomieszczenie, które będzie poddane replikacji w wirtualnej rzeczywistości.

Rys. 1. Pomieszczenie wybrane do przeprowadzenia analizy – widok 360°.

Do wykonania analizy potrzebna była szczegółowa dokumentacja metryczna i fotograficzna. Pomiary zebrano z dokładnością do 1 cm i użyto do nich standardowej metrówki. Zdjęcia służące za referencję do modeli zostały wykonane za pomocą aparatu TrueDepth 12 MP z obiektywem szerokokątnym.

2.1. Stanowisko komputerowe

Wszelkie prace związane z tworzeniem, edycją oraz optymalizacją modeli zostały wykonane na stanowisku komputerowym o następujących parametrach:

- procesor: Intel Core i7 6700K, 4 GHz kolejna linia wypunktowania;
- karta graficzna: GeForce GTX 1070 ARMOR OC 8 GB;
- płyta główna: MSI Z170A MPOWER Gaming Titanium;
- OS: Microsoft Windows Pro 8 x64;
- RAM: G.Skill Ripjaws V DDR4m 16 GB, 3200MHz, CL16;
- SSD: Samsung 860 Pro 256 GB.

2.2. Skaner FARO Focus 3D X330 oraz SCENE 2019

Proces skanowania 3D polega na analizie rzeczywistości w celu wygenerowania danych pozwalających na jej opis. Może on dotyczyć zarówno pojedynczych obiektów, jak i całych przestrzeni. Zgromadzone dane są następnie przetwarzane i wykorzystywane do skonstruowania modeli 3D umożliwiających jak najwierniejsze odwzorowanie otoczenia. Skan został wykonany za pomocą skanera FARO Focus 3D X330. Jest to urządzenie z bardzo dużym zasięgiem doskonale sprawdzającym się przy skanowaniu budynków. Skaner nadaje się do skanowania różnego rodzaju przestrzeni zarówno wewnątrz, jak i na zewnątrz.

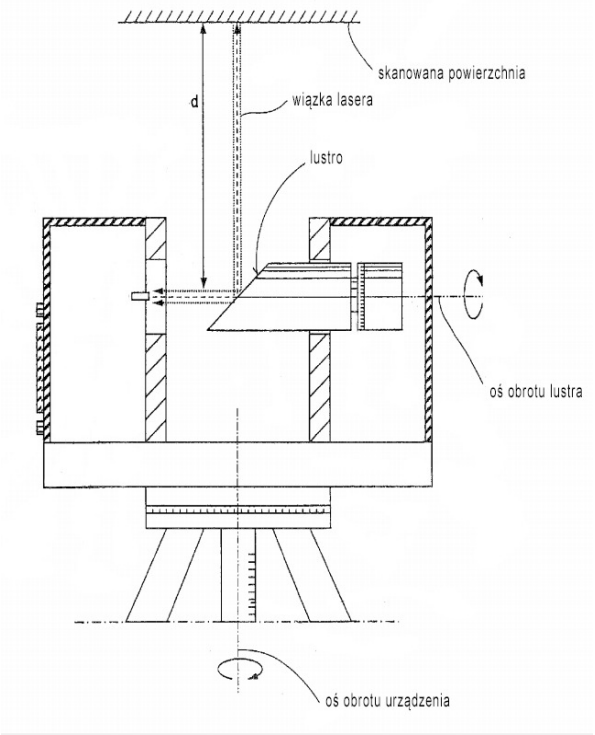
Rys. 2. Zdjęcie przedstawiające skanera Faro Focus 3D X330 [8]

Parametry skanera FARO Focus 3D X330 przedstawiono w Tabeli 1.

Tabela 1. Parametry skanera FARO Focus 3D X330 [8]

Parametr	Wartość
Precyzja	± 2 mm
Zasięg	od 0,6 m do 330 ma
Szybkość pomiaru	do 976 000 punktów/s
Laser	klasa 1
Waga	5,2 kg
Rozmiar	240 x 200 x 100 mm
Sterowanie	wyświetlacz dotykowy
Dodatkowe	GPS, kompas, wysokościomierz, kompensator dwuosiowy

Po ustawieniu sprzętu w skanowanej przestrzeni i rozpoczęciu procesu skanowania głowica lasera zaczyna wykonywać powolne obroty pionowe. W tym samym czasie rotor znajdujący się wewnątrz urządzenia obraca w osi pionowej lustro odbijające pod kątem wiązkę lasera. Obraca się on bardzo szybko w skutek czego wytwarza się pionowa smuga światła, która powoli przemieszcza się i skanuje środowisko wokół.



Rys. 3. Schemat działania skanera Faro Focus 3D X330 [9]

Emitowany promień lasera odbija się od punktów w skanowanej przestrzeni i wraca do urządzenia. Następnie jest on analizowany i mierzony. Chmura punktów tworzy model 3D w 360º. Proces jest przedstawiony na Rysunku nr 3. Oprócz chmury punktów skaner rejestruje również zdjęcia.

Po wykonaniu skanów 3D dokonuje się ich postprodukcji w programie SCENE 2019. Otrzymane skany są zazwyczaj łączone w całość za pomocą danych pobranych z wysokościomierza oraz systemu GPS. Zeskanowane punkty

mają kolory oddające wiernie rzeczywistość. Po wykonaniu rejestracji skanów oraz ich połączeniu można eksplorować skan.

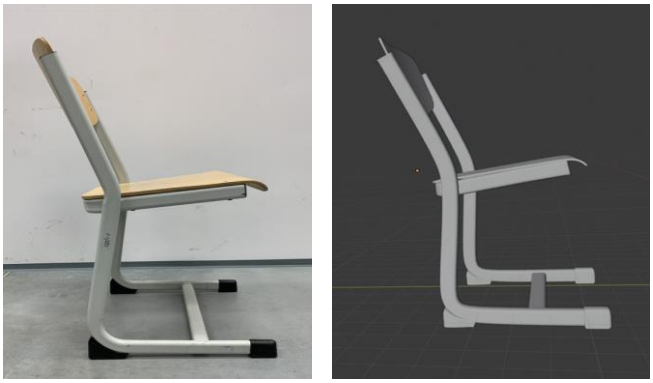
3. Proces modelowania 3D

Modelowanie 3D jest procesem polegającym na stworzeniu matematycznej interpretacji powierzchni dowolnego obiektu. Płaszczyzna jest przedstawiona w trójwymiarze. Efektem procesu modelowania 3D jest model 3D. Do tworzenia i edycji obiektów 3D wykorzystuje się specjalistyczne oprogramowanie nazywane modelerem. Wszystkie modele zostały stworzone w programie Blender 2.8. To darmowe oprogramowanie charakteryzuje się bardzo wysoką jakością oraz ogromnymi możliwościami. Dodatkowo bardzo dobrze sprawdza się w przypadku projektów architektonicznych.

3.1. Tworzenie modeli

Przedmioty w pokoju pogrupowano pod względem efektowności i istotności dla odtwarzanego modelu. Przed rozpoczęciem prac nad modelowaniem pomieszczenia zdecydowano się na pominięcie części elementów.

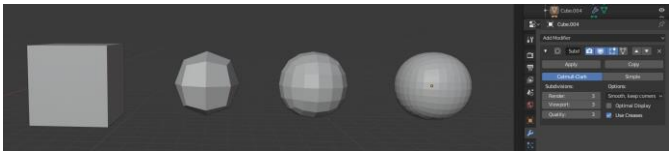
a) b)



Rys. 4. Porównanie dokładności wykonania modelu:

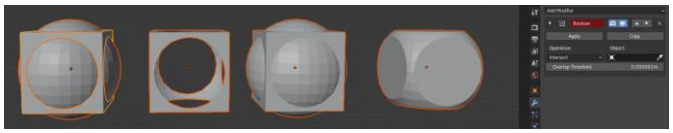
a) zdjęcie referencyjne krzesła
b) model stworzony w programie Blender 2.8

Modele były tworzone poprzez operacje na podstawowych bryłach. Siatka była edytowana ręcznie lub przy użyciu wbudowanych narzędzi. Wykorzystywano głównie dwa modyfikatory: Subdivision Surface oraz Boolean. Pierwszy z nich jest odpowiedzialny za dzielenie ścian w siatce na mniejsze. Dzięki modyfikatorowi Subdivision Surface uzyskano gładkie powierzchnie pracując jednocześnie na prostych, posiadających niewiele krawędzi siatkach. Przykładowe zastosowanie tego modyfikatora można zaobserwować na Rysunku 5.



Rys. 5. Działanie modyfikatora Subdivision Surface

Drugiem często wykorzystywanym narzędziem był modyfikator Boolean. Pozwala on na wykonanie operacji na siatce, które normalnie byłyby zbyt skomplikowane do wykonania. Narzędzie wykorzystuje do tego 3 operacje: suma, różnica oraz część wspólna. Zasada działania modyfikatora Boolean została przedstawiona na Rysunku 6.



Rys. 6. Działanie modyfikatora Boolean

Następnie wykonano proces optymalizacji modeli przy użyciu narzędzia Planar Decimate. Modyfikator Decimate pozwala na zredukowanie ilości wierzchołków i ścian przy jednocześnie minimalnej zmianie kształtu. Nadaje się idealnie do siatek, które były rzeźbione lub tworzone z dużą ilością modyfikatorów typu Subdivision Surface.

Proces optymalizacji pozwolił na drastyczne zmniejszenie ilości krawędzi w siatce przy jednoczesnym zachowaniu wysokiej jakości modeli. Efekt tego procesu prezentuje Tabela 2.

Tabela 2. Parametry modelu przed i po optymalizacji

Przed optymalizacją	Liczba wierzchołków	5 653 388
	Liczba ścian	5 512 948
	Liczba trójkątów	11 115 908
	Zużycie pamięci	1,74 GB
Po optymalizacji	Liczba wierzchołków	514 429
	Liczba ścian	255 467
	Liczba trójkątów	906 970
	Zużycie pamięci	263,9 MB

Modele zostały scalone oraz wyeksportowane do formatu **.ply*.

4. Proces skanowania 3D

Skany zostały wykonane za pomocą skanera FARO Focus 3D X330. Wierne odwzorowanie pomieszczenia wymagało wykonania procesu skanowania aż 12 razy, każdy trwał około 5 minut. Szczegółowe parametry skanowania zostały przedstawione w Tabeli 3.

Tabela 3. Parametry procesu skanowania

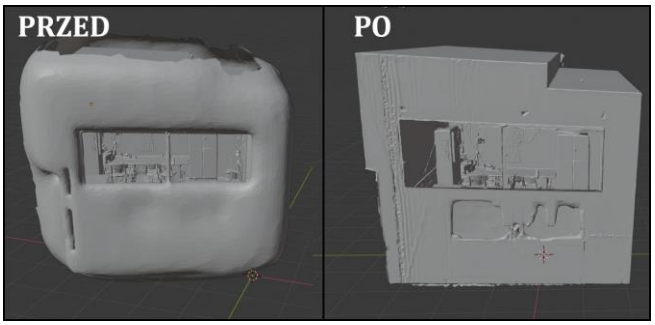
Parametr	**Wartość**
Nazwa profilu	Wewnątrz do 10 m
Rozdzielczość [MPkty]	11.1 (1/8)
Jakość	3x
Poziomo	0º do 360º
Pionowo	-60º do 90º
Kolor	Włączony
Czas skanowania [mm:ss]	05:06
Rozmiar skanu [MB]	69,31
Rozmiar skanu [Pkt]	5143 x 2169

Pliki zostały zaimportowane do programu SCENE 2019, a następnie przeszły proces przetwarzania oraz automatycznej rejestracji. Następnie możliwe było obejrzenie wygenerowanej

chmury punktów. Przekrój chmury punktów został zaprezentowany na Rysunku 7.

Rys. 7. Chmura punktów reprezentująca skanowane pomieszczenie

Chmura punktów została wyeksportowana jako siatka trójkątów w formacie **.ply*. Proces był bardzo długotrwały względu na częste awarie programu SCENE 2019. Na wyeksportowanym modelu pojawiły się problemy, które wymagały dodatkowej pracy w programie Blender 2.8. Należało usunąć dodatkowe krawędzie i usunąć powstałe artefakty. Efekt tej czynności jest widoczny na Rysunku 8.



Rys. 8. Efekt dodatkowej obróbki skanu w programie Blender 2.8

5. Badanie stopnia immersyjności

W ramach artykułu przeprowadzono badania z użyciem technologii wirtualnej rzeczywistości. Testy miały na celu zbadanie stopnia immersyjności wizualizacji pomieszczenia w zależności od metody jej wytworzenia. Źródłem analizy były wyniki ankiety.

5.1. Grupa badawcza

W badaniu wzięła udział zróżnicowana grupa osób składająca się z trzynastu kobiet oraz siedemnastu mężczyzn. Badani znajdowali się w przedziale wiekowym 19-65 lat oraz posiadali różny stopień wykształcenia.

5.2. Scenariusze badawcze oraz przebieg badania

Przed przystąpieniem do badania każdy z uczestników dokładnie zapoznał się z materiałem referencyjnym w postaci fotografii odwzorowywanego pomieszczenia. Podczas badania akietowani mieli możliwość dowolnej eksploatacji scen w wirtualnej rzeczywistości. Przykład sposobu badania immersji w VR pokazano na Rysunku 9.

Rys. 9. Zdjęcie ochotnika podczas badania z wykorzystaniem platformy Oculus Rift S

Scenariusz badań przewidywał uruchomienie dwóch różnych aplikacji VR zawierających kolejno:

- prezentacje modelu wygenerowanego na podstawie procesu skanowania 3D oraz prezentacje modelu stworzonego w programie Blender 2.8,Karta graficzna: GeForce GTX 1070 ARMOR OC 8 GB
- prezentacje zdjęcia 360° stworzonego za pomocą skanera FARO.

Aplikacje zostały uruchomione na zestawie Oculus Rift S. W skład zestawu wchodzą gogle do wyświetlania wirtualnej rzeczywistości oraz dwa kontrolery dotykowe. Poruszanie po wirtualnej makiecie odbywało się za pomocą joysticków znajdujących się na kontrolerach dotykowych Oculus. Lewy joystick odpowiadał za swobodne przemieszczanie się w przestrzeni, a prawy za obrót kamery. Sprzęt został zaprezentowany na Rysunku 10.

Rys. 10. Zestaw Oculus Rift S wykorzystany do badań [10]

Po zakończeniu badania uczestnicy zostali poproszeni o uzupełnienie ankiety. Ankieta składała się z trzech sekcji. Pierwsza z nich zawierała wstępne pytania dotyczące wieku, płci oraz wykształcenia uczestnika.

Kolejna sekcja ankiety dotyczyła poszczególnych metod generowania pomieszczeń 3D. Uczestnicy kolejno oceniali aplikacje prezentujące pomieszczenie stworzone na bazie skanu 3D, wymodelowane w Blenderze 2.8 oraz stworzone na bazie zdjęcia 360°. Ankietowani oceniali aspekty takie jak

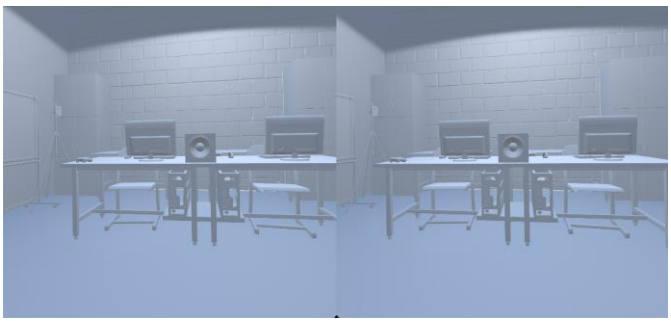
podobieństwo, jakość obrazu, przestrzenność obrazu, swobodę poruszania, ilość i jakoś szczegółów. Wystawili również ogólną ocenę, która podsumowywała wrażenia z danej wizualizacji. Odpowiedzi na pytania były udzielane w pięciostopniowej skali.

Ostatnia sekcja ankiety dotyczyła technologii wirtualnej rzeczywistości. Pojawiły się w niej pytania dotyczące wcześniejszych doświadczeń z technologią VR oraz jej przydatności. Ostatnie pytanie dotyczyło komfortu użytkowania gogli VR oraz ewentualnego dyskomfortu, jakiego mógł doświadczyć użytkownik. Pojawienie się tego pytania jest związane z możliwością wystąpienia choroby wirtualnej powodującej między innymi mdłości, zawroty i bóle głowy.

5.3. Opis aplikacji

Na potrzeby badań stworzono dwie aplikacje w środowisku Unity. Projekty zostały dostosowane do współpracy z zestawem Oculus Rift S. Pierwsza aplikacja pozwala użytkownikowi na eksplorację modeli pomieszczeń wygenerowanych za pomocą procesu skanowania 3D (Rysunek 11) oraz procesu modelowania 3D (Rysunek 12). Po uruchomieniu aplikacji oczom użytkownika ukazuje się scena zawierająca wygenerowane pomieszczenie.

Rys. 11. Zrzut ekranu aplikacji VR prezentującej model stworzony na bazie procesu skanowania 3D



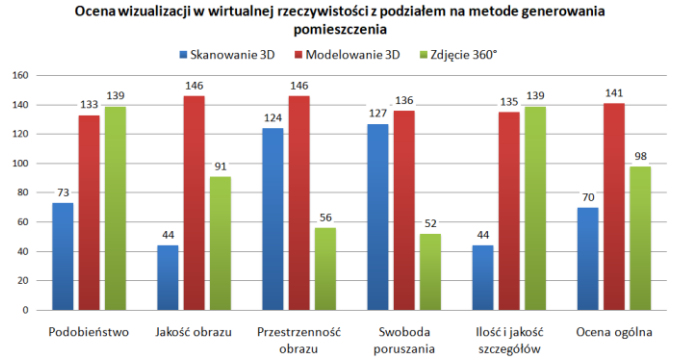
Rys. 12. Zrzut ekranu aplikacji VR prezentującej model stworzony w Blenderze 2.8

Ostatnia aplikacja służyła do wyświetlania w wirtualnej rzeczywistości zdjęcia 360°. Zdjęcie zostały wykonane za pomocą skanera FARO i powstało podczas procesu skanowania. W tej wersji aplikacji użytkownicy nie mają możliwości swobodnego przemieszczania się po scenie, a jedynie obejrzeć scenę z jednego konkretnego punktu. Zrzut ekranu aplikacji VR prezentującej zdjęcie 360° został przedstawiony na Rysunku 13.

Rys. 13. Zrzut ekranu aplikacji VR prezentującej zdjęcie 360°

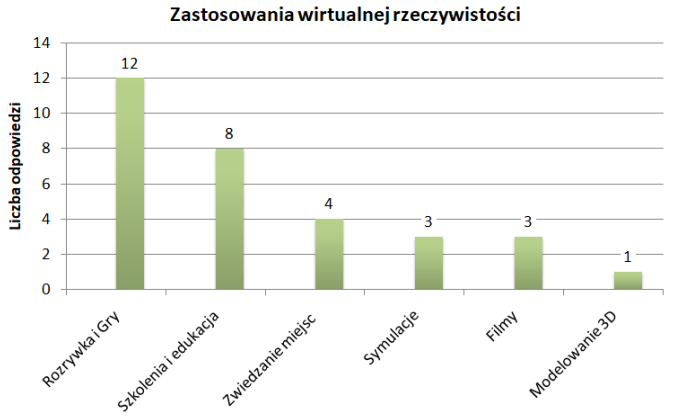
5.4. Analiza wyników badań

Bazą do analizy immersyjności była ankieta wypełniana na podstawie wirtualnej projekcji. Rezultatem ankiety były odpowiedzi uczestników badania na pytania zamknięte, otwarte oraz pytania z zastosowaniem skali odpowiedzi od 1 (bardzo słabo) do 5 (bardzo dobrze). Ankietowani najlepiej ocenili prezentację modelu wygenerowanego na podstawie procesu modelowania 3D, a najgorzej prezentację na podstawie skanu 3D (Rysunek 14).



Rys. 14. Wykres przedstawiający ocenę wizualizacji w wirtualnej rzeczywistości - podział na metody generowania pomieszczenia

Dwudziestu ankietowanych wymieniło swoje spostrzeżenia na temat zastosowań dla wirtualnej rzeczywistości. Ich odpowiedzi zostały posegregowane oraz przedstawione na wykresie znajdującym się na Rysunku 15.



Rys. 15. Zastosowania technologi wirtualnej rzeczywistości według przebadanych osób

Dodatkowo przeprowadzono analizę statystyczną zdobytych wyników biorąc pod uwagę oceny ogólne

uczestników ankiety. Wyznaczono wartości maksymalne, minimalne, medianę oraz średnią ocen. Wyniki zostały zaprezentowane w Tabeli 4.

$$\overline{x}_n = \frac{x_1 + x_2 + \ldots + x_n}{n} = \frac{\sum_{i=1}^{n} x_i}{n} \quad (1)$$

Tabela 4. Analiza statystyczna ocen ogólnych z odpowiedzi ankietowanych

	Skanowanie 3D	Modelowanie 3D	Zdjęcie 360°
Maksimum	1	4	2
Minimum	3	5	4
Średnia	2,33	4,6	3,26
Mediana	2	5	3

Warto wspomnieć, że badanie nie obyło się bez objawów choroby wirtualnej. Informacje o liczbie dolegliwości podczas korzystania z okularów VR zostały zaprezentowane na Rysunku 16. Niepokojące objawy odczuło trzynastu uczestników badania, a pięciu z nich skarżyło się na więcej niż jeden symptom.

Rys. 16. Liczba dolegliwości, które wystąpiły u trzynastu badanych podczas korzystania z okularów VR

6. Koszty

Omawiając koszty projektu pod uwagę zostały wzięte następujące czynniki: koszt pracy ludzkiej, koszt sprzętu (koszt stanowiska do modelowania 3D, akwizycji i postprodukcji, koszt skanera FARO Focus 3D X330, koszt stanowiska do projekcji wirtualnej) oraz koszt oprogramowania.

Proces skanowania trwał łącznie około 14 godzin 30 minut. Wykorzystując najnowsze oprogramowanie SCENE 2019 czas przetwarzania skanów wyniósł łącznie 12 godzin 15 minut. Pomimo bardzo dobrego stanowiska komputerowego i odpowiedniej konfiguracji oprogramowanie działało bardzo niestabilnie. Należy również wspomnieć, że korzystając z poprzednich wersji programu ten czas mógłby się znacznie wydłużyć. W przeciwieństwie do poprzedników oprogramowanie SCENE 2019 posiada intuicyjny interfejs, a dodatkowo większa część postprodukcji jest wykonywana automatycznie.

Pomimo znacznej automatyzacji procesu obróbki skanów 3D program SCENE 2019 nie uzyskano zadowalających wyników, a model wymagał dodatkowej pracy urzeczywistniającej skany. Obróbka została wykonana w programie Blender 2.8. Polegała ona na przycinaniu sceny, usuwaniu artefaktów i dorysowywaniu elementów.

Proces dodatkowej obróbki w programie Blender 2.8 mógłby zostać pominięty w przypadku zaprzestania procesu postprodukcji na etapie wygenerowania chmury punktów. Prezentacja wirtualnej wizualizacji osobom trzecim wymagałaby wtedy wykupienia dodatkowej usługi – udostępniania plików w chmurze FARO. Skanowanie 3D oprócz bardzo dobrego stanowiska komputerowego wymaga drogiego sprzętu. Koszty skanerów 3D potrafią wynosić kilka, kilkanaście lub nawet kilkaset tysięcy złotych. Nowy skaner FARO Focus 3D X330 w momencie tworzenia analizy kosztował 102 960,00 zł (kurs dolara amerykańskiego z dnia 23.09.2019).

Koszt stanowiska komputerowego w momencie jego zakupu wynosił 7103,04 zł. Zestaw komputerowy został wykorzystany do procesu skanowania 3D (akwizycja danych oraz postprodukcja skanów), procesu modelowania 3D w programie Blender 2.8 oraz projekcji wirtualnej.

Proces ręcznego modelowania pomieszczenia zajął łącznie 10 godzin oraz 30 minut. Na koszt pracy ludzkiej dotyczącej procesu modelowania 3D składają się następujące aspekty: czas potrzebny na wykonanie dokumentacji fotograficznej, czas poświęcony na obróbkę zdjęć,czas modelowania,czas wyszukiwania gotowych obiektów w Internecie orazczas poświęcony na optymalizację siatki obiektów.

Oprogramowanie Blender 2.8 jest darmowe. Program jest całkowicie autonomicznym rozwiązaniem. Pozwala na modelowanie, rzeźbienie, teksturowanie oraz renderowanie modeli. Dodatkowo Blender 2.8 ma o wiele mniejsze wymagania niż oprogramowanie SCENE 2019.

Koszt projekcji wirtualnej zeskanowanego pomieszczenia można oszacować na dwa różne sposoby. Pierwszy z nich to wykorzystanie zestawu Oculus Rift S, którego koszt na czas pisania pracy wynosi 1962,13 zł (kurs euro z dnia 23.09.2019). W tej cenie wliczony jest support w postaci gotowych bibliotek i wtyczek znacznie ułatwiających pracę.

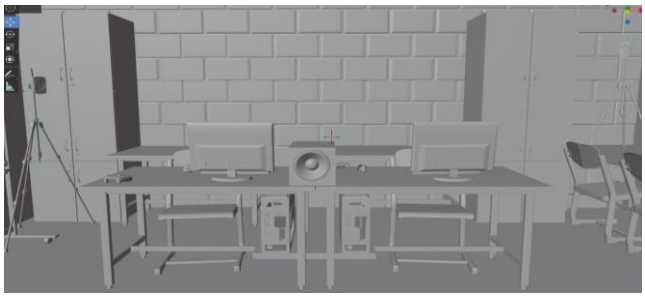
Drugi sposób to wykorzystanie budżetowych, pasywnych okularów 3D. Koszt takiego sprzętu waha się zazwyczaj w granicach od 30,00 do 100,00 zł. Dostępne są również znacznie droższe konstrukcje. W tym rozwiązaniu znacznie wzrasta czas oprogramowania aplikacji VR.

Aplikacja VR została stworzona w darmowej wersji środowiska Unity. Istnieje opcja zakupu usługi Unity Pro, której koszt na czas pisania pracy wynosił 495,00 zł (kurs dolara amerykańskiego z dnia 23.09.2019). Zapewnia ona wsparcie dla freelancerów oraz profesjonalnych zespołów zajmujących się tworzeniem gier 2D, 3D, VR i AR.

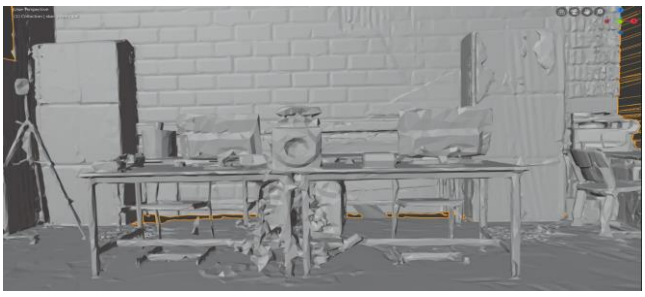
7. Podsumowanie

Na podstawie procesu skanowania 3D, modelowania 3D oraz badania z wykorzystaniem technologii wirtualnej rzeczywistości można sformułować następujące wnioski. Wpływ na jakość skanu ma bardzo dużo aspektów [12, 13, 14] takich jak: rodzaj oświetlenia (sztuczne, naturalne), kolor powierzchni skanowanej powierzchni [7], tekstura skanowanej powierzchni [10], rodzaj i kolor użytego lasera, transparentności powierzchni skanowanej powierzchni oraz stopień odbicia światła. Dodatkowo każdy z obecnych dostępnie na rynku skanerów posiada problem z błyszczącymi czy transparentnymi przedmiotami. Aby umożliwić skanowanie takich obiektów należy je wcześniej odpowiednio przygotować. Jednym ze sposobów jest pokrycie powierzchni matową farbą, pudrem lub talkiem. Obiekt może zostać również zmatowiony np. za pomocą papieru ściernego. W przypadku skanowanego pomieszczenia zabrakło logicznych przesłanek do wspomnianych zabiegów. Korzystanie z pudru czy talku zostało wykluczone z powodu znajdującego się w pokoju sprzętu komputerowego i niebezpieczeństwa zanieczyszczenia go pyłem. Duże znaczenie na końcowy efekt skanowania ma też kolor obiektu [11], jednak jest to zależne od barwy strumienia lasera. Wszlekiego rodzaju zniekształcenia są najmocniej widoczne w przypadku czarnych oraz zielonych powierzchni [13]. Aby osiągnąć jak najlepsze wyniki w skanowaniu 3D najbardziej pożądanym kolorem obiektu jest szary [13].

a)



b)



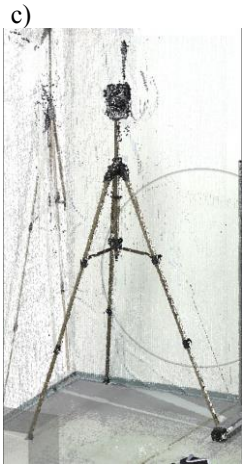
Rys. 17. Porównanie dwóch metod generowania realistycznych pomieszczeń 3D:
a) ręcznie wykonanego modelu 3D
b) modelu wygenerowanego na podstawie skanu

Autorzy prac naukowych wskazują również, że skanery mają duże problemy ze skanowaniem powierzchni transparentnych oraz mocno odbijających światło [14]. To stwierdzenie potwierdza się również w przypadku skanera FARO Focus 3D X330. W wygenerowanym modelu znajduje się o wiele więcej tego typu błędów, ilekroć laser napotyka mocno błyszczącą lub transparentną powierzchnię.

Proces skanowania 3D jest na ten moment dość dopracowanym rozwiązaniem, ale proces generowania modeli 3D z chmury punktów wymaga wielu usprawnień. Siatka wygenerowanego pomieszczenia była bardzo zniekształcona, pełna ubytków i nadprogramowych krawędzi. Stworzony obiekt 3D przedstawiał rzeczywiste pomieszczenie w bardzo prymitywny, wręcz karykaturalny sposób. Chmura punktów, która powstała w procesie skanowania o wiele lepiej i dokładniej odzwierciedlała rzeczywistość.

a) b)

c) d)



Rys. 18. Dokładność odwzorowania głośnika w wirtualnej przestrzeni:
a) zdjęcie referencyjne,
b) model stworzony w programie Blender 2.8,
c) chmura punktów utworzona podczas procesu skanowania 3D,
d) model wygenerowany oprogramowaniem SCENE 2019.

Podczas analizy kosztów projektów stwierdzono, że efekty uzyskane przy pomocy skanowania 3D są niezadowalające. Przyglądając się manualnie stworzonym modelom 3D możemy zauważyć, że praca grafika 3D charakteryzuje się o wiele większą dokładnością, immerysjnością, poprawnością oraz dbałością strukturą. Oba projekty zajęły porównywalną ilość czasu jednak proces skanowania 3D wymaga bardzo dużych nakładów finansowych na wydajne stanowisko komputerowe, sprzęt do skanowania oraz specjalistyczne oprogramowanie z pakietem dodatkowo płatnych usług. Dodatkowo oprogramowanie SCENE 2019 nie jest na tyle dopracowanym i autonomicznym rozwiązaniem, aby wygenerować zadowalający jakością model 3D. Obiekt musiał przejść dodatkowy proces postprodukcji w programie Blender

2.8. Jedynym plusem przemawiającym za procesem skanowania są wyraźnie mniejsze pliki, jednak nie są to znaczące różnice, dyskwalifikujące w jakikolwiek sposób modelowanie 3D.

Podczas badań z użyciem technologii wirtualnej rzeczywistości przeanalizowano immersyjność stworzonych pomieszczeń. Najlepszą ocenę ogólną uzyskała wizualizacja na bazie stworzonych manualnie modeli. Modelowanie 3D okazało się niewiele gorsze od zdjęć 360° tylko w dwóch przypadkach: podobieństwa oraz liczby i jakości szczegółów. Skanowanie 3D zostało ocenione najgorzej i jedyne kategorie, w których oceny badanych są zbliżone do modelowania 3D to przestrzenność obrazu i swoboda poruszania się.

Sześćdziesiąt procent ankietowanych nie miało wcześniej styczności z technologią wirtualnej rzeczywistości, jednak wszyscy jednogłośnie stwierdzili, że jest ona przydatna. Dodatkowo dwudziestu ankietowanych wymieniło swoje spostrzeżenia na temat zastosowań dla wirtualnej rzeczywistości. Najczęściej pojawiały się sugestie dotyczące rozrywki i gier komputerowych, takiej odpowiedzi udzieliło dwunastu ankietowanych. Oprócz tego wymieniono szkolenia i edukację (osiem osób), zwiedzanie miejsc (cztery osoby), symulacje oraz filmy (trzy osoby), a także modelowanie 3D (jedna osoba).

Trzynastu ankietowanych odczuło negatywne skutki korzystania z gogli VR. Dodatkowo jedna z ankietowanych osób przyznała się, że podczas korzystania z okularów do wirtualnej rzeczywistości towarzyszyło jej uczucie niepokoju i lęku. Kolejna z nich opisywała „wrażenie wpadania w przepaść, zapadania się". Wśród osób z objawami choroby wirtualnej pojawiły się trzy osoby, które już wcześniej korzystały z aplikacji VR. Dwie z nich przyznały, że po raz kolejny odczuwają negatywne efekty korzystania z okularów do wirtualnej rzeczywistości. Nieprzyjemne objawy pojawiały się najczęście w momencie poruszania się po wirtualnej makiecie. Ankietowani najczęściej skrażyli się na ogólny dyskomfort (osiem osób). Pojawiły się również zawroty głowy oraz zmęczenie oczu (cztery osoby), a także ból głowy i mdłości (jedna osoba).

Porównując obydwie metody, w skanowanym pomieszczeniu, należy stwierdzić, że wybór modelowania 3D wydaje się być słuszniejszy zwłaszcza pod względem większej efektowności w generowaniu obiektów 3D. Rozwój technologii skanowania 3D może w przyszłości zaowocować lepszą dostępnością oraz lepszymi technologiami generowania obiektów 3D na bazie skanów. Obecnie dobre rezultaty wymagają ogromnych nakładów finansowych na sprzęt i specjalistyczne oprogramowanie. W większości przypadków modelowanie 3D okazuje się o wiele korzystniejszą i lepszą opcją. Proces skanowania 3D może stanowić doskonałe wsparcie dla pracy grafika 3D, ponieważ wygenerowane na bazie skanu 3D pliki dostarczają ogólnych informacji dotyczących wymiarów czy rozmieszczenia elementów. Na ten moment skanowanie 3D może znaleźć zastosowanie w niszowych, mocno wyspecjalizowanych dziedzinach takich jak np.: kryminalistyka czy dokumentacja zabytków.

Literatura

[1] Szymczyk T.: *Presentation of the most interesting geographical places using virtual reality technology*, 12th International Technology, Education and Development Conference, 2018.

[2] Plakhotniuk I., Popko M., Szymczyk T.: *Health Aspect of Immersion in VR. Virtual Disease – Factor or Myth*, 13th International Technology, Education and Development Conference, 2019.

[3] Szymczyk T.: *Make learning more interesting by using virtual reality*, 9th International Conference on Education and New Learning Technologies, 2017.

[4] Szymczyk T., Skulimowski S.: *The use of virtual and agumented reality in the teaching process*, 11th International Technology, Education and Development Conference, 2017.

[5] *Global 3D Scanning Market is Expected to Reach $4.9 Billion, by 2020 - Allied Market Research* https://www.prnewswire.com/news-releases/global-3d-scanning-market-is-expected-to-reach-49-billion-by-2020---allied-market-research-268876941.html [15.04.2019]

[6] *3D Scanning Market - Global Opportunity Analysis and Industry Forecast, 2018-2025,* https://www.alliedmarketresearch.com/3D-scanning-market [15.04.2019]

[7] *3D mapping and 3D modelling market- growth, trends and forecast (2019-2024),* https://www.mordorintelligence.com/industry-reports/3d-mapping-and-3d-modelling [15.04.2019]

[8] *Dane dotyczące skanera FARO Focus 3D X330 oraz oprogramowania SCENE 2019*, https://www.faro.com/pl-pl/produkty/budownictwo-bim-cim/faro-focus/focus3d-pliki-do-pobrania/ [20.09.2019]

[9] *Dokumenty patentowe skanera 3D firmy FARO*, https://worldwide.espacenet.com/searchResults?ST=singleline&locale=en_EP&submitted=true&DB=&query=faro&Submit=Search [20.04.2019]

[10] *Informacje dotyczące platformy Oculus oraz zestawu Oculus Rift S*, https://www.oculus.com/rift-s/ [20.09.2019]

[11] *Clark J., Robson S.: Accuracy of measurements madewith a Cyrax 2500 Laser Scanner against Surfaces of known colour,* Survey Review, Nr 37/2004.

[12] Amiri Parian J.: *Integrated Laser Scanner and Intensity Image Calibration and Accuracy Assessment*, International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences, Nr 36/2005.

[13] Lemeš S.: *Influences of surface parameter on laser 3D scanning*, University of Zenica, 2010.

[14] Lichti D.: *The effects of reflecting surface material properties on timeof-flight laser scanner measurements*, Symposium on Geospatial Theory, Processing and Applications, Ottawa 2002

[15] *Informacje dotyczące platformy Oculus oraz zestawu Oculus Rift S*, https://www.oculus.com/rift-s/ [20.09.2019]