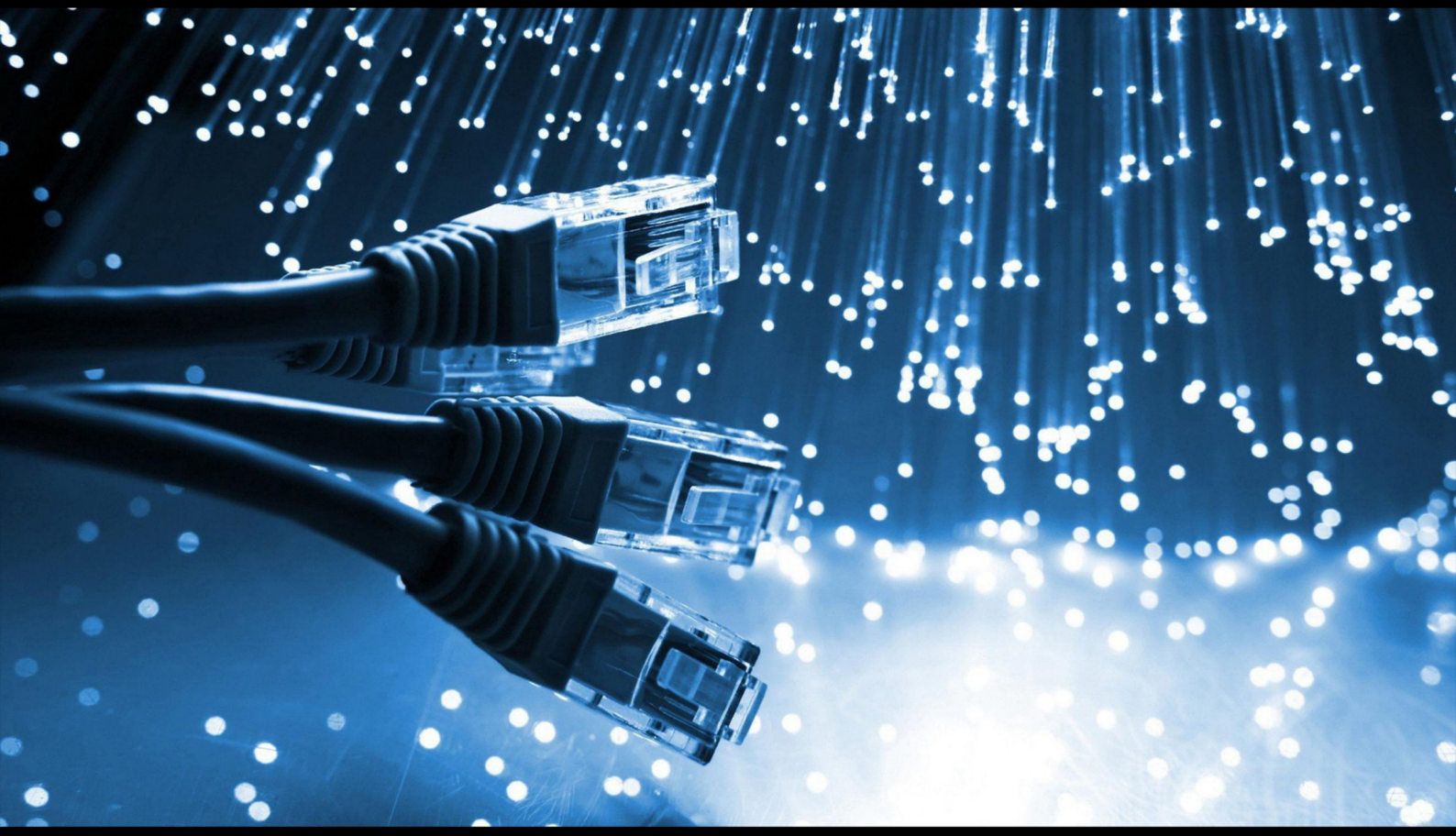
JCSI

Journal of Computer Sciences Institute

Volume 28/2023



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. Małgorzata Plechawska-Wójcik
dr inż. Piotr Muryjas
dr inż. Kamil Żyła
dr inż. Marcin Badurowicz
dr inż. Jacek Kęsik
dr inż. Maria Skublewska-Paszkowska
dr inż. Elżbieta Miłosz
dr inż. Marek Miłosz, prof. uczelni
dr inż. Maciej Pańczyk
dr inż. Jakub Smołka
dr inż. Krzysztof Dziedzic
dr inż. Tomasz Nowicki
dr inż. Grzegorz Kozieł
dr Mariusz Dzieńkowski
dr inż. Sylwester Korga
dr Marcin Barszcz

Skład komputerowy:
Anna Sałamacha
e-mail: a.salamacha@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:

Małgorzata Plechawska-Wójcik
Piotr Muryjas
Kamil Żyła
Marcin Badurowicz
Jacek Kęsik
Maria Skublewska-Paszkowska
Elżbieta Miłosz
Marek Miłosz
Maciej Pańczyk
Jakub Smołka
Krzysztof Dziedzic
Tomasz Nowicki
Grzegorz Kozieł
Mariusz Dzieńkowski
Sylwester Korga
Marcin Barszcz

Computer typesetting:
Anna Sałamacha
e-mail: a.salamacha@pollub.pl

Cover design:
Marta Zbańska

ISSN 2544-0764

Spis treści

Contents

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 28 (2023) 186-189
Received: 16 April 2023
Accepted: 29 June 2023

The Examination of SQL Queries Efficiency in Chosen IT System

Badanie wydajności zapytań SQL w wybranym systemie informatycznym

Krzysztof Barczak*

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

Abstract
Optimization of SQL queries is an important part of any system using a database. Valuable time can be gained by consciously making changes within the database. Therefore, several different optimization methods have been selected and resented in this article. The research was conducted on the MS SQL database engine. The execution times of SQL queries were carefully examined according to defined criteria, then performance optimization was done, followed by repeated tests to obtain results after the optimization. Finally, the results were compared through which conclusions were drawn.

Keywords: SQL query optimization; databases; testing; performance analysis

Streszczenie
Optymalizacja zapytań SQL jest istotnym elementem każdego systemu, wykorzystującego bazy danych. Świadomie dokonane zmiany w obrębie bazy danych pozwalają zyskać cenny czas. Dlatego też w niniejszym artykule wybrano i przedstawiono kilka różnych metod optymizacji, natomiast badania zostały przeprowadzone na silniku bazodanowym MS SQL. Czasy wykonania zapytań SQL dokładnie zmierzono według zdefiniowanych kryteriów, a następnie przeprowadzono optymalizację wydajnościową oraz powtórzono badania w celu ponownego określenia czasu wykonania zoptymalizowanych zapytań. Na podstawie porównanych wyników zostały wyciągnięte odpowiednie wnioski.

Słowa kluczowe: Optymalizacja zapytań; bazy danych; testowanie; analiza wydajności

*Corresponding author

Email address: krzysztof.barczak@pollub.edu.pl (K. Barczak)

1. Wstęp

Bazy danych występują wszędzie. W każdej firmie, która wykorzystuje dane muszą być one przechowywane w odpowiedni sposób. Zależnie od dostępności do systemów informatycznych, możliwości finansowych czy też chęci, pracownicy zauważają, w których obszarach program potrzebuje więcej czasu na odpowiedź z bazy danych, a gdzie wyniki zapytań są otrzymywane od razu. Jeżeli firmie zależy na prawidłowym korzystaniu z systemu powinna ona zwracać również uwagę na pojawiające się w nim problemy wydajnościowe.

Dzięki zmianom zapytań SQL można zaoszczędzić czas, co oczywiście przekłada się na zyski. Z doświadczenia autora wynika, że firmy, którym zależy na płynnym działaniu systemów informatycznych, osiągają lepsze wyniki. Dlatego tak ważne jest pojęcie optymalizacji zapytań SQL i utrzymania prawidłowego działania serwera. Należy uważać nawet na z pozoru rzadko występujące problemy funkcjonowania serwera bazy danych, takie jak brak miejsca na serwerze produkcyjnym. Skrócenie czasu zapytania zapobiega blokowaniu wykonania polecenia przez drugie (ang. deadlocks) oraz zwiększa wydajność działania aplikacji, co prowadzi do sytuacji „win-win", gdzie w tym samym czasie z aplikacji może korzystać więcej osób, bez konieczności rozbudowy serwera.

Wybranie odpowiednich kolumn podczas przygotowywania zapytań sprawia również, że czas wykonania zapytania skraca się. Ważnym aspektem są również indeksy. Umiejętne ich dobranie potrafi znacząco skrócić czas oczekiwania odpowiedzi na zapytanie. Należy mieć jednak na uwadze, że nadmiarowa liczba indeksowanych kolumn może negatywnie wpłynąć na pracę bazy danych. Konieczna jest więc analiza potrzeb informacyjnych i optymalne dobranie kolumn, które będą indeksowane.

Oczywiście równie istotnym czynnikiem są parametry serwera. Niewłaściwie dobrana konfiguracja sprzętowa i oprogramowanie potrafią znacząco obniżyć wydajność silnika bazodanowego jak i również aplikacji korzystającej z niego.

2. Przegląd literatury

Wybranie odpowiednich kolumn podczas definiowania zapytania sprawia, że czas jego wykonania skraca się. Dostępne są publikacje, w których dokonano analizy i przedstawiono zagadnienia odnoszące się do badań czasów wykonywania zapytań SQL.

Pierwszym przykładem jest artykuł naukowy „Analiza wydajności relacyjnych baz danych Oracle oraz MSSQL na podstawie aplikacji desktopowej" przygotowany przez Grzegorza Dziewita, Jakuba Korczyńskiego i Marię Skublewską-Paszkowską [1]. W artykule poruszona jest kwestia czasów prostych zapytań dla następujących rekordów: 10, 30, 50, 10000, 50000, 100000 dla danych tekstowych oraz numerycznych. Głównym celem badania było porównanie dwóch silników bazodanowych: MS SQL oraz Oracle. Wnioski okazały się następujące: baza Oracle szybciej wykonuje

operację SELECT, jednak baza MS SQL szybciej realizuje operacje UPDATE oraz INSERT.

Drugą, nieco starszą pozycją jest artykuł naukowy „Optymalizacja Microsoft SQL server przy współpracy z Microsoft Dynamic NAV" przygotowany przez Marcina Wocha i Piotra Łebkowskiego. Artykuł ten przedstawia narzędzia stworzone do wspierania administratorów systemu NAV. Narzędzia umożliwiają optymalizację kodu oraz monitorowanie systemu [2]. W tej publikacji potwierdzono iż prawidłowa administracja serwerem oraz systemem Navision pozwala na optymalizację zapytań oraz manipulacji danymi.

Kolejnym przykładem jest pozycja „Pro T-SQL 2019" [3]. Autor tej publikacji opisuje wiele aspektów pisania kodu T-SQL oraz jego optymalizacji. Jednym z nich jest optymalizacja logicznych odczytów (ang. Logical Reads). Jednym z wniosków jest poprawa wydajności wykonania kodu T-SQL, dzięki dobrym praktykom projektowym.

Warto również wspomnieć o badaniach przeprowadzonych na bazach nie SQL-owych [4]. Wybrana pozycja „Analiza prędkości wykonywania zapytań w wybranych bazach nie SQL-owych" przedstawia badania oraz analizę wyników czasów zapytań dla baz nierelacyjnych takich jak CouchDB i MongoDB. Głównym, generalnym wnioskiem jest stwierdzenie iż baza MongoDB jest wydajniejsza względem bazy CouchDB.

Powyższe pozycje literaturowe pokazują sposób badania czasów zapytań oraz wskazują na różne czynniki, które mogą zostać poddane optymalizacji w celu uzyskania krótszych czasów wykonania zapytań SQL. Za takie aspekty uważa się między innymi korzystanie z tabel tymczasowych, tworzenie indeksów czy wybieranie tylko potrzebnych danych. Podczas przeglądu literatury zauważono, iż brakuje badań przedstawiających optymalizację czasów wykonywania zapytań SQL dla bazy danych MS SQL Server.

3. Cel i zakres badań

Celem badań było przeprowadzenie analizy wpływu optymalizacji zapytań SQL na szybkość ich wykonywania oraz udowodnienie, iż różne metody optymalizacyjne potrafią skrócić czas wykonania zapytania. Z badań nad możliwościami optymalizacyjnymi zapytań SQL wyciągnięto odpowiednie wnioski, tak aby pomóc w doborze najbardziej optymalnych ustawień oraz konfiguracji serwera MS SQL. Przeprowadzona analiza wpływu optymalizacji zapytań SQL pozwoliła na postawienie następujących tez badawczych:

T1: Wybór tylko niezbędnych kolumn w zapytaniu SQL pozwala na zwiększenie wydajności zapytań.

T2: Poprzez usunięcie klauzuli „order by" nastąpi skrócenie czasu trwania zapytania SQL.

T3: Skrócenie czasu trwania zapytań SQL nastąpi poprzez dodanie odpowiednich indeksów.

T4: Rodzaj złączenia między tabelami wpływa na czas wykonania zapytań SQL.

4. Metodyka badawcza

Do przeprowadzenia badań wydajności zapytań SQL przygotowano aplikację na wzór znanego portalu zajmującego się sprzedażą samochodów [5]. Do wykonania aplikacji testowej wykorzystane zostały: szkielet programowania Symfony oraz baza danych Microsoft SQL Server Express. Aplikację przygotowano zgodnie ze wzorcem MVC – Model-View-Controller [6]. Stanowiskiem badawczym był laptop o parametrach podanych poniżej:

Procesor Intel Core i5-7300HQ 2.50Ghz, Windows 10 w wersji Home, oraz karta graficzna NVIDIA GeForce GTX 1050 oraz 16 GB RAM.

Każde przeprowadzone badanie składało się z 10 prób. Uzyskane wyniki następnie uśredniano oraz wprowadzono wybrane modyfikacje do zapytań, a badania powtórzono. Na koniec uzyskane wyniki zostały porównane, w celu określenia, która forma zapytania SQL jest bardziej wydajna. Przygotowana metodyka badań wykorzystywała stworzoną aplikację wraz z zapytaniami, które mogły zostać zoptymalizowane, w celu porównania czasów wykonania zoptymalizowanych zapytań SQL z zapytaniami pierwotnymi.

Należy jednak wspomnieć, iż wszystkie badania wykonano w środowisku bez działania dodatkowych usług oraz z wyłączonym programem antywirusowym, tak aby warunki każdego badania były do siebie maksymalnie zbliżone.

W ocenie optymalnej struktury kodu SQL zastosowano kryterium czasu, aby stwierdzić czy wprowadzone do kodu optymalizacje pozwalają jasno określić, kiedy zaproponowana zmiana kodu może być użyteczna.

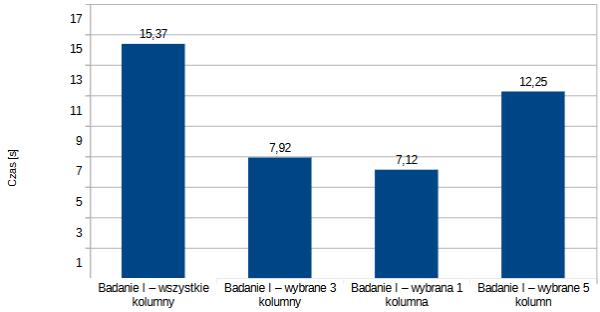
5. Wyniki badań

5.1. Wpływ liczby zwracanych kolumn

Zmniejszenie liczby wybranych kolumn w operacji typu SELECT to proces optymalizacji polegający na ograniczeniu wyboru kolumn tylko do tych koniecznych, w celu skrócenia czasu wykonania zapytania SQL przez silnik bazodanowy.

W badaniu zapytania o różnej strukturze przygotowano cztery próby, odpowiednio wybierając: wszystkie kolumny, pięć kolumn, trzy kolumny oraz finalnie jedną kolumnę.

Na Rysunku 1 przedstawiono wykres dla każdej grupy ze średnimi czasami wykonywania zapytań, które zostały wyrażone w sekundach.



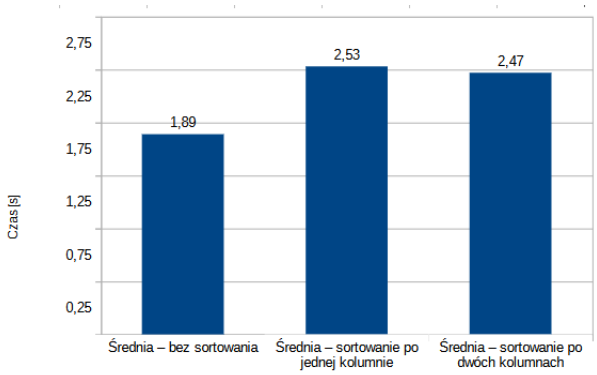
Rysunek 1: Wykres porównawczy uśrednionych czasów zapytań dla różnych ilości kolumn.

Różnica w otrzymanych czasach zapytań jest zauważalna. W powyższym badaniu średni czas wykonania zapytań dla 1000000 rekordów z całej bazy jest najdłuższy (około 2 razy dłuższy od zapytania, w którym wybrana została jedna kolumna) i wynosi 15,37 sekundy. Średni czas zapytania wybierającego tylko jedną kolumnę był najkrótszy i wynosił 7,12 sekundy. Dla trzech kolumn czas wyniósł 7,92 sekundy, zaś dla pięciu kolumn było to 12,25 sekundy. Na podstawie wyników tego badania można stwierdzić, że im większa liczba kolumn w definicji zapytania tym dłuższy czas oczekiwania na wynik jego wykonania.

5.2. Narzut czasowy spowodowany sortowaniem

Usunięcie klauzuli „order by" jest metodą optymalizacji, polegającą na wykluczeniu sortowania danych w kodzie, gdy nie jest ono faktycznie istotne. Silnik bazodanowy w przypadku użycia klauzuli „order by" musi dodatkowo przy zwracaniu wyników posortować dane w odpowiedniej kolejności. W miejscach, w których nie potrzeba posortowanych wyników nie powinno się stosować tej klauzuli.

Na Rysunku 2 przedstawiono wykres ze średnimi czasami wykonywania zapytań, dotyczących klauzuli order by, które zostały wyrażone w sekundach.

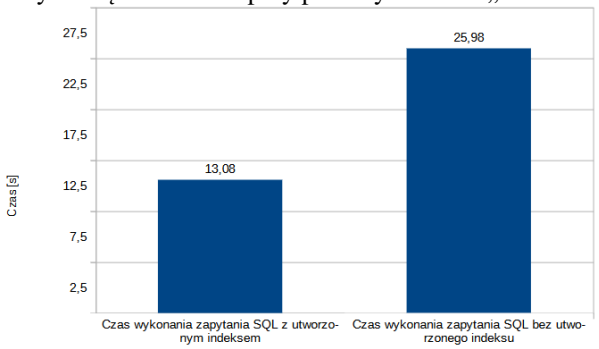


Rysunek 2: Wykres porównawczy uśrednionych czasów zapytań badania II.

Otrzymany wynik w badaniu wpływu klauzuli „order by" na czas zapytania SQL to różnica około 25% pomiędzy czasami realizacji zapytań bez klauzuli „order by", a zapytaniami SQL z tą klauzulą. W przypadku zapytania SQL z klauzulą sortującą istnieje pewność, że wyniki są uporządkowane we wskazanej kolejności. Natomiast w przypadku zapytania SQL bez klauzuli „order by" silnik bazodanowy zwraca nieposortowane wyniki. Jednakże jeżeli nie muszą być one posortowane w odpowiedniej kolejności, można skrócić czas potrzebny na wykonanie zapytania, poprzez brak użycia klauzuli „order by".

5.3. Wpływ indeksowania

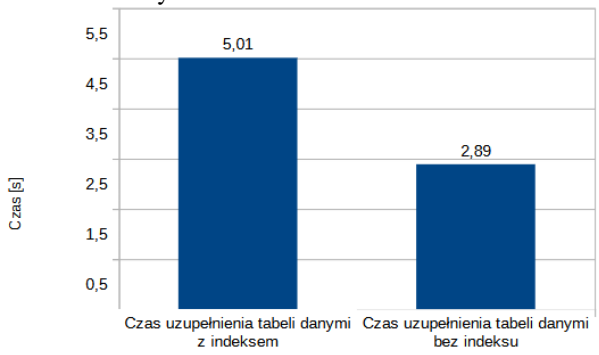
Indeksowanie danych w kolumnach, jest to metoda optymalizacji polegająca na zastosowaniu odpowiedniego indeksu na kolumnie, dzięki której dane są często wyszukiwane. Proces optymalizacji zapytań zaprezentowany w badaniu III polegał na dodaniu indeksu na kolumnie w wybranej tabeli, za pomocą której często odbywa się filtrowanie przy pomocy klauzuli „where".



Rysunek 3: Wykres porównawczy uśrednionych czasów dla zapytań SQL z utworzonym indeksem oraz bez.

Na Rysunku 3 przedstawiono wykres ze średnimi czasami wykonania zapytań z badania wpływu indeksu na czas trwania zapytania SQL. Wartości te zostały wyrażone w sekundach. Średni czas wykonywania zapytania dla polecenia „select", w przypadku gdy nie dodano indeksu wydłużył się o około 13 sekund przy 1000000 rekordów, co w tym przypadku przełożyło się na wzrost o 100% w porównaniu do zapytań, gdzie indeks był utworzony [7].

Jednakże, należy pamiętać o negatywnych aspektach indeksowania. Przy badaniu III zweryfikowano dodatkowo koszt utrzymania indeksów. Zbadano i porównano czasy potrzebne na wstawianie danych do tabeli. Na Rysunku 4 przedstawiono wykres ze średnimi czasami dodawania danych do tabeli.

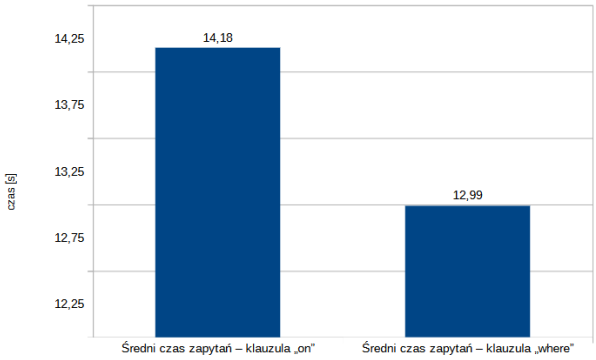


Rysunek 4: Wykres porównawczy czasów dodawania danych do tabeli.

Jak można zauważyć, czas uzupełnienia tabeli milionem wierszy z założonym indeksem jest dłuższy o ponad 2 sekundy niż do tej bez indeksu. Dlatego warto dobrze przemyśleć każdy przypadek z osobna i określić czy indeks jest konieczny.

5.4. Wpływ rodzaju złączenia

Rodzaj złączenia również może wpłynąć na czas trwania zapytania SQL. Badanie czwarte polegało na sprawdzeniu czasów wykonywania zapytań złożonych, w których łączono tabele przy pomocy klauzuli „on" oraz w klauzuli „where". Na Rysunku 5 przedstawiono wykres ze średnimi czasami wykonywania zapytań.



Rysunek 5: Wykres porównawczy uśrednionych czasów zapytań dla złączeń miedzy tabeli w klauzuli „on" oraz „where".

Uśrednione wyniki pomiędzy złączeniami nie są znaczące dla tego badania. Czasy wykonywania zapytań gdy tabele były połączone w klauzuli „on" są dłuższe o 9%, od tych z tabelami połączonymi w klauzuli „where". W tym przypadku, gdy dwie tabele zawierają po 1000000 rekordów jest to różnica około jednej sekundy.

6. Wnioski

Celem artykułu było zbadanie skuteczności zastosowania metod, których zadaniem jest optymalizacja zapytań SQL w środowisku bazodanowym MS SQL. Po przeanalizowaniu dostępnych publikacji naukowych z zakresu optymalizacji zapytań, wybrano cztery różne metody, pozwalające skrócić czas wykonywania zapytań SQL.

Na podstawie wyników badania wpływu liczby kolumn na czas trwania zapytania, można stwierdzić iż teza T1, czyli. „Wybór tylko niezbędnych kolumn w zapytaniu SQL pozwala na zwiększenie wydajności zapytań" jest prawdziwa. Poprzez ograniczenie liczby kolumn w zapytaniu SQL można zaoszczędzić niezbędny czas.

W przypadku tezy T2, tj. „Poprzez usunięcie klauzuli order by nastąpi skrócenie czasu trwania zapytania SQL", drobną zmianą, czyli usunięciem klauzuli można skrócić czas potrzebny na wykonanie zapytania SQL. Warto przejrzeć kod w celu odnalezienia zbędnych klauzuli sortujących.

Wyniki dla badania III potwierdzają postawioną wcześniej tezę T3, tj. „Skrócenie czasu trwania zapytań SQL nastąpi poprzez dodanie odpowiednich indeksów". Dodanie odpowiedniego indeksu pozwoli znacznie skrócić czas potrzebny na wykonanie zapytania „select". Niemniej jednak, należy pamiętać o negatywnym wpływie indeksu w kontekście na przykład dodawania danych do bazy.

Na podstawie wyników badania wpływu rodzaju złączenia tabel, można stwierdzić, iż teza T4, czyli: „Złączenia między tabelami mają wpływ na czasy wykonania zapytania SQL." jest prawdziwa. Można zaobserwować pewną różnicę w czasach realizacji zapytań złożonych SQL, gdy tabele są złączone w klauzulach „on" oraz „where". Skuteczniejszą metodą w tym badaniu okazało się złączenie tabel w klauzuli „where".

Literatura

[1] G. Dziewit, J. Korczyński, M. Skublewska-Paszkowska, Performance analysis of relational databases Oracle and MS SQL based on desktop application, Journal of Computer Sciences Institute 8 (2018) 263-269, https://doi.org/10.35784/jcsi.693.

[2] M. Woch, Optymalizacja Microsoft SQL server przy współpracy z Microsoft Dynamic NAV, Studia Informatica 28(2) (2007) 17-29.

[3] E. Noble, Pro T-SQL 2019, Apress, Berkeley, 2020.

[4] W. Bolesta, Analysis of query execution speed in the selected NoSQL databases, Journal of Computer Sciences Institute 7 (2018) 138-141, https://doi.org/10.35784/jcsi.662.

[5] Otomoto, https://www.otomoto.pl, [01.11.2022].

[6] Model MVC, https://vavatech.pl/technologie/architektura/mvc, [01.11.2022].

[7] SQL - INDEX, https://www.plukasiewicz.net/SQL, [01.11.2022].

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 28 (2023) 190-196
Received: 22 May 2023
Accepted: 4 July 2023

Comparative analysis of selected databases on the example of a proprietary web application

Analiza porównawcza wybranych baz danych na przykładzie autorskiej aplikacji internetowej

Łukasz Przychodzień*, Dominika Radwan, Grzegorz Kozieł

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

Abstract
Database performance is one of the most important factors affecting the usability of the system. Therefore, the authors of the article decided to examine 3 popular database systems: MySQL, MS SQL and PostgreSQL, analyzing their performance. For this purpose, a test application was prepared and Docker software was used to simulate different hardware parameters. Depending on the selected settings and the number of records, different results were obtained. For small data sets, the differences were almost imperceptible. They have drastically increased for large data sets. In this case, MySQL fared poorly, and MS SQL was the best. This means that the choice of the database is very important, and it is worth considering the available hardware, the amount of data and the queries performed.

Keywords: performance analysis; relational databases; DBMS

Streszczenie
Wydajność bazy danych jest jednym z najistotniejszych czynników wpływających na użyteczność systemu. Dlatego autorzy artykułu postanowili przebadać 3 popularne systemy bazodanowe: MySQL, MS SQL oraz PostgreSQL analizując ich wydajność. W tym celu przygotowano aplikację testową oraz wykorzystano oprogramowanie Docker, aby umożliwić symulację różnych parametrów sprzętowych. W zależności od wybranych ustawień oraz liczebności rekordów uzyskano inne wyniki. Dla małych zestawów danych różnice były niemal niezauważalne. Drastycznie zwiększyły się dla dużych zbiorów danych. W tym przypadku słabo wypadł MySQL, a najlepiej MS SQL. Oznacza to, że wybór bazy danych jest niezwykle istotny, a do jego podjęcia warto rozważyć dostępny sprzęt, liczebność danych oraz wykonywane zapytania.

Słowa kluczowe: analiza porównawcza; relacyjne bazy danych; SZDB

*Corresponding author
Email address: **lukasz.przychodzien@pollub.edu.pl** (Ł. Przychodzień)

1. Wstęp

W obecnych czasach bardzo trudno jest znaleźć aplikację czy stronę internetową, której działanie nie byłoby oparte na wykorzystaniu bazy danych w celu przechowywania informacji koniecznych do poprawnego funkcjonowania systemu. W znacznej większości przypadków wykorzystywane są do tego celu relacyjne bazy danych, przechowujące dane w uporządkowanych tabelach. Każda z tabel zawiera informacje na określony temat, taki jak dane klientów, zamówień czy pracowników. Dodatkowo poszczególne tabele połączone są relacjami pozwalającymi na uporządkowanie związków pomiędzy nimi.

Ważnym aspektem podczas projektowania aplikacji oraz doboru użytych technologii jest kwestia wydajności. Niezwykle istotnym jest, aby zapytania obsługiwane przez system bazodanowy wykonywały się wystarczająco szybko. Dodatkowo należy zwrócić uwagę na zróżnicowanie wielkości przechowywanych zbiorów danych w przypadku typowych aplikacji oraz możliwości wykorzystanej architektury serwerowej, a także kosztów związanych z jej zakupem.

Biorąc pod uwagę powyższe kwestie autorzy artykułu zdecydowali się przeprowadzić porównanie wydajności najpopularniejszych systemów bazodanowych. Dodatkowo, aby zweryfikować wpływ parametrów środowiska do przeprowadzenia testów wykorzystana zostanie technologia wirtualizacji w postaci kontenerów Docker. Ponadto przeprowadzane testy zostaną wykonane dla kilku zbiorów danych o różnej wielkości, dzięki czemu zbadana zostanie również skalowalność systemu bazodanowego.

2. Przegląd literatury

Aby lepiej poznać możliwości wybranych systemów bazodanowych, została przeprowadzona analiza artykułów o podobnym temacie powstałych na przestrzeni ostatnich lat.

W artykule “Performance comparison of relational databases SQL Server, MySQL and PostgreSQL using a web application and the Laravel framework” [1] autorzy przedstawili porównanie wydajności trzech silników bazodanowych: SQL Server, PostgreSQL oraz MySQL. Do badań wykorzystana została aplikacja napisana w języku PHP z użyciem frameworka Laravel. W badaniach skupiono się na najpopularniejszych operacjach bazodanowych, takich jak np. SELECT, INSERT, DELETE. Czas wykonania instrukcji na bazie

był mierzony z poziomu kodu PHP obliczając różnicę czasu po i przed wykonaniu instrukcji. Działanie każdej instrukcji sprawdzano kolejno na różnych liczbach rekordów badanej tabeli począwszy od 1 aż do 20000 rekordów. Dla małej liczby rekordów tj. do 1000 najszybszy okazał się MySQL, niewiele wolniejszy był PostgreSQL, z kolei znacznie wolniejszy okazał się MSSQL. W przypadku większej liczby rekordów (powyżej 1000) najszybszy okazał się PostgreSQL.

Artykuł „Performance analysis of relational databases Oracle and MS SQL based on desktop application" [2] opisuje porównanie wydajności relacyjnych baz danych Oracle oraz MS SQL. Do testów autorzy wykorzystali autorską aplikację napisaną w języku C# z użyciem technologii MVVM (Model-View-View-Model). Dziewit et al. badali czas odpowiedzi systemów bazodanowych w zależności od wykonywanych zapytań wywoływanych przez zewnętrzną (aplikacja pozasystemowa względem bazy danych) aplikację desktopową wspomagającą zarządzanie warsztatem samochodowym. Czasy wykonania poszczególnych zapytań zostały wyznaczone na podstawie statystyk wykonanych poleceń zapisywanych automatycznie przez systemy bazodanowe. Autorzy analizowali czasy wykonania operacji języka DML zarówno dla danych tekstowych, jak i binarnych. Badania wykazały, że baza MS SQL lepiej realizuje operacje insert i update niż baza Oracle, z kolei Oracle jest wydajniejszy w przypadku operacji typu select na danych binarnych.

Autorzy pracy pod tytułem "Analiza porównawcza baz danych" [3] przeprowadzili badania najpopularniejszych systemów zarządzania bazami danych: MySQL, PostgreSQL oraz Firebird. Analizowali oni szybkość wykonania zapytań, obciążenie procesora oraz pamięci na dysku z użyciem autorskiej aplikacji przygotowanej w oparciu o standardy TPC (Transaction Processing Performance Council). Aplikacja została napisana w języku Java. Testy wydajności przeprowadzono na dwóch komputerach o różnych parametrach technicznych.

Pod względem szybkości wykonania zapytań najlepsza okazała się baza danych MySQL. Identycznie wygląda sytuacja pod względem ilości zajmowanego miejsca w pamięci – najlepsze wyniki uzyskała baza MySQL. Wyniki badań obciążenia procesora wykazały zależność od rodzaju wykonywanej operacji. Podczas dodawania dużej liczby rekordów najmniejsze obciążenie procesora zaobserwowano dla silnika bazy Firebird.

W pracy [4] porównana została wydajność baz danych MySQL, PostgreSQL, MariaDB oraz H2. Autorzy wyznaczyli średni czas wykonania operacji dodania, aktualizacji, usuwania oraz wybierania danych. Każde badanie zostało wykonane 10 razy dla 1, 100, 2500, 5000, 7500 i 10000 rekordów, a następnie obliczony został czas średni. MySQL wykazała słabą wydajność podczas pracy z dużą ilością danych. 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. 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. PostgreSQL wykazał najlepsze wyniki dla operacji aktualizacji.

Innym rozwiązaniem bazodanowym cieszącym się coraz większą popularnością są bazy nierelacyjne. Artykuł „A comparison of database performance of MariaDB and MySQL with OLTP workload" opisuje porównanie relacyjnej bazy MySQL z nierelacyjnym systemem MariaDB [5]. Autorzy analizowali wydajność obu baz na przykładzie systemu klasy OLTP wspomagającego pracę lotniska. Ich pomiary wykazały znaczną przewagę bazy relacyjnej MySQL.

Podobne badania opisuje artykuł „Performance Evaluation for CRUD Operations in Asynchronously Replicated Document Oriented Database" [6]. Jego twórcy porównali wydajność 3 systemów NoSQL z 3 bazami relacyjnymi badając czasy wykonania operacji CRUD. W przypadku tej pracy lepsze wyniki osiągnęły rozwiązania nierelacyjne.

Zainteresowanie tematyką wydajności relacyjnych baz danych w ostatnich latach jest bardzo wysokie. Powstały liczne prace naukowe weryfikujące efektywność zarówno baz relacyjnych jak i nierelacyjnych, sprawdzające ich działanie w połączeniu z aplikacjami internetowymi oraz desktopowymi. Jednak nie znaleziono badań wykorzystujących framework Spring, jak również wirtualizację z użyciem środowiska Docker. Wykorzystanie konteneryzacji umożliwia przeprowadzenie testów wydajnościowych dla różnorodnych parametrów technicznych środowiska testowego oraz ich analizę w sposób niespotykany w żadnej z opisanych badań.

3. Relacyjne systemy baz danych

Za twórcę relacyjnych baz danych uznaje się Edgara Coda, który zaproponował to rozwiązanie w roku 1969, gdy pracował dla IBM-u. Cechą charakterystyczną baz relacyjnych jest przedstawienie przechowywanych danych w formie tabeli z wierszami i kolumnami [7]. Każda tabela zawiera w sobie rekordy danych (pojedynczy rekord zwany jest krotką) gdzie każda krotka jest definiowana przez typ atrybutu oraz unikalną dla tej krotki wartością.

Kolejną cechą wyróżniającą bazy relacyjne jest wykorzystanie relacji, czyli połączeń pomiędzy tabelami, co pozwala na łączenie tabel na wiele sposobów umożliwiając wyszukiwanie zaawansowanych połączeń pomiędzy danymi. Dodatkowo umożliwia to na ograniczenie redundancji przechowywanych danych. Każda tabela dzieli z inną tabelą co najmniej jeden atrybut, jeśli jest z nią w relacji 'jeden do jednego', 'wiele do jednego' albo 'wiele do wielu'.

3.1. MySQL

Stworzony przez szwedzką firmę MySQL AB w roku 1995, która została wykupiona przez Sun Microsys-

tems, obecnie funkcjonujący pod nazwą Oracle Corporation. Dziś jest jednym z najbardziej popularnych open sourcowych systemów zarządzania relacyjną bazą danych [8].

Podobnie jak inne relacyjne bazy danych, MySQL przechowuje dane w tabelach składających się z wierszy i kolumn. Użytkownicy mogą definiować dane, manipulować nimi, kontrolować je i wyszukiwać zapytania przy użyciu języka Structured Query Language, bardziej znanego jako SQL [9].

Wśród jego licznych zastosowań można wymienić pakiet oprogramowania służącego do rozwoju aplikacji internetowych znany pod akronimem LAMP (Linux, Apache, MySQL, PHP) oraz jego alternatywę, czyli pakiet LEMP. W przypadku tego drugiego serwer Apache został zastąpiony serwerem Nginx (czyt. engine-x, stąd litera E w skrócie).

3.2. PostgreSQL

Początki systemu PostgreSQL datuje się na lata 80. ubiegłego wieku. Powstał on na Uniwersytecie Kalifornijskim w Berkeley. Obecnie nad rozwojem języka czuwa grupa entuzjastów pod nazwą "PostgreSQL Global Development Group" [10]. PostgreSQL jest bazą obiektowo-relacyjną dystrybuowaną na darmowej licencji open source.

Dane przechowywane są w postaci tabel połączonych relacjami. Dodatkowo, PostgreSQL integruje obiektowe podejście do baz danych z relacyjnym pozwalając na wykorzystanie narzędzi znanych z relacyjnych baz danych, ale poszerzonych o możliwości związane z obiektowością. Podobnie jak w programowaniu obiektowym możemy stworzyć klasę oraz jej obiekty i klasę dziedziczącą po tej klasie.

Relacyjne właściwości bazy pozwalają z kolei na używanie strukturalnego języka zapytań znanego z relacyjnych baz danych do efektywnego i szybkiego przeszukiwania danych niedostępnego w obiektowych bazach danych. Dodatkowo PostgreSQL wspiera transakcje zgodne z zasadami ACID (Atomicity, Consistency, Isolation, Durability - ang. atomowość, spójność, izolacja, trwałość). Dzięki czemu gwarantuje poprawne przetwarzanie wykonywanych operacji [11].

3.3. MS SQL

Microsoft SQL Server to system zarządzania bazą danych, rozpowszechniany przez korporację Microsoft. Jest to główny produkt bazodanowy tej firmy oraz trzeci co do popularności najczęściej wybierany system bazodanowy na świecie. Głównym językiem zapytań wykorzystywanym w MS SQL jest Transact-SQL. Poza edycjami czysto komercyjnymi Microsoft udostępnia również edycje darmowe do dowolnego zastosowania takie jak Express i Developer [12].

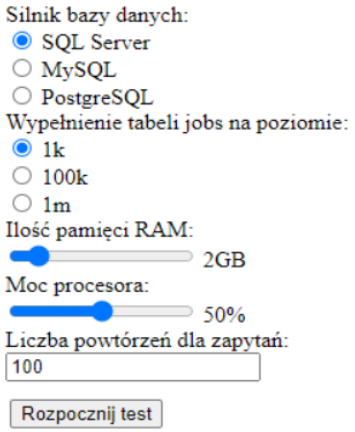
MS SQL, podobnie jak inne relacyjne bazy danych, przechowuje dane w tabelach, kolumnach i wierszach. Każdy wpis jest zdefiniowany unikalnym identyfikatorem. Od samego początku twórcy systemu za nadrzędny cel przyjmowali wydajność oraz niezawodność opracowywanej bazy danych. Równie istotna była skalowalność opracowywanego systemu, czyli jego dostosowanie do obsługi nawet bardzo dużych zbiorów danych.

Microsoft SQL Server jest szeroko stosowany we wdrożeniach korporacyjnych. Ponadto udostępnia on kilka narzędzi ETL (Extract, Transform and Load) oraz usługi raportowania, w których można dodawać, modyfikować i wyszukiwać dane przy użyciu ustandaryzowanego języka zapytań strukturalnych (SQL). MS SQL to rozwijająca się platforma danych używana w rozwiązaniach biznesowych i danych o znaczeniu krytycznym w środowisku lokalnym, w chmurze, czy na platformach hybrydowych [13].

4. Aplikacja testowa

Do przeprowadzenia testów przygotowana została autorska aplikacja napisana w oparciu o szkielet programistyczny Spring. Pozwoliła ona na zautomatyzowanie oraz przyspieszenie wykonywania pomiarów. Na rysunku 1 zaprezentowano ekran główny aplikacji. Do przeprowadzenia serii testów należy wybrać analizowaną bazę danych, ustawić odpowiednie parametry testowe oraz liczbę powtórzeń w serii oraz nacisnąć przycisk "Rozpocznij test".

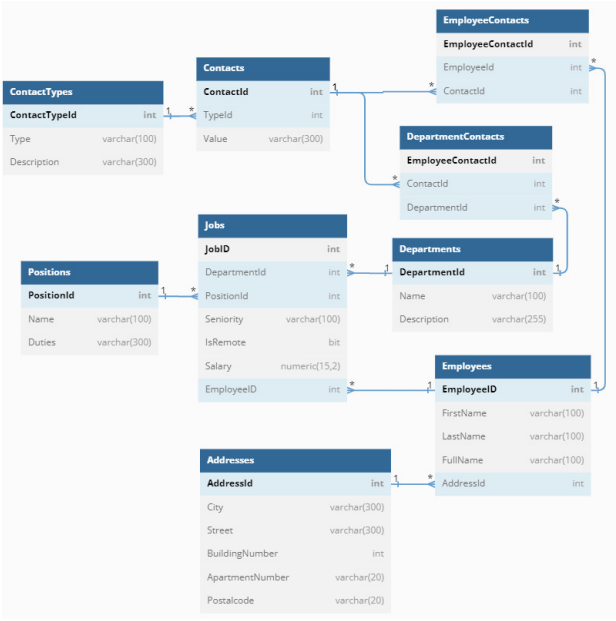
Aplikacja automatycznie uruchamia wybraną bazę danych na kontenerze, a następnie wykonuję serię zapytań według przygotowanych scenariuszy testowych, uwzględniając wybraną liczbę powtórzeń. Czas rozpoczęcia oraz zakończenia obsługi każdego z zapytań jest rejestrowany, a na ich podstawie wyliczany jest dokładny czas wykonania danej instrukcji. Zebrane w ten sposób pomiary następnie zapisywane są do pliku csv. Tak przygotowane dane zostały następnie opracowane z użyciem języka Python.



Rysunek 1: Ekran główny aplikacji.

4.1. Schemat bazy danych

Na potrzeby przeprowadzenia badań przygotowana została baza testowa o schemacie przedstawionym na rysunku 2. Zawiera ona informacje o pracownikach zatrudnionych w firmie, ich danych, umowach, czy wynagrodzeniu.



Rysunek 2: Schemat ERD bazy testowej.

5. Metodyka

Podstawową operacją realizowaną przez większość systemów bazodanowych jest pobieranie danych spełniających zadane warunki. W tabeli 1 przedstawiono zestawienie scenariuszy testowych wykorzystanych podczas testów. Każdy z opisywanych scenariuszy uwzględnia inny typ zapytania DQL (ang. Data Query Language - język do definiowania zapytań).

Tabela 1: Scenariusze testowe

Scenariusz	Kod SQL
I	SELECT c.Value FROM Contacts c WHERE c.Value LIKE '%@gmail.com';
II	SELECT TOP 5 Salary FROM Jobs j ORDER BY Salary DESC LIMIT 5;
III	SELECT c.Value, e.FullName FROM Contacts c JOIN EmployeeContacts ec ON c.ContactId = ec.ContactId JOIN Employees e ON ec.EmployeeId = e.EmployeeId;
IV	SELECT COUNT(DISTINCT j.EmployeeId), d.Name FROM Jobs j JOIN Departments d ON j.DepartmentId = d.DepartmentId GROUP BY d.DepartmentId, d.Name;
V	SELECT SUM(CASE WHEN j.IsRemote = 0 THEN 1 ELSE 0 END) AS office, SUM(CASE WHEN j.IsRemote = 1 THEN 1 ELSE 0 END) AS remote, d.Name FROM Jobs j JOIN Departments d ON j.DepartmentId = d.DepartmentId GROUP BY d.DepartmentId, d.Name;
VI	WITH CTE(remoteCNT, DepartmentId) AS (SELECT SUM(j.IsRemote), j.DepartmentId FROM Jobs j GROUP BY DepartmentId) SELECT d.Name, c.remoteCNT FROM cte c JOIN Departments d ON c.DepartmentId = d.DepartmentId;
VII	SELECT e.FullName FROM Employees e WHERE NOT EXISTS (SELECT 1 FROM EmployeeContacts ec WHERE ec.EmployeeId = e.EmployeeID)
VIII	SELECT e.FullName FROM Employees e WHERE EXISTS (SELECT 1 FROM EmployeeContacts ec WHERE ec.EmployeeId = e.EmployeeID)
IX	SELECT DISTINCT a.City FROM Addresses a;
X	SELECT j.EmployeeID, 'low income' AS ds FROM Jobs j WHERE j.Salary < 1000 UNION ALL SELECT j.EmployeeID, 'middle class' FROM Jobs j WHERE j.Salary BETWEEN 1000 AND 10000 UNION ALL SELECT j.EmployeeID, 'upper class' FROM Jobs j WHERE j.Salary > 10000
XI	SELECT City, Street, BuildingNumber, ApartmentNumber FROM Addresses WHERE City IN ('Warszawa', 'Kraków', 'Wrocław')
XII	SELECT City, Street, BuildingNumber, ApartmentNumber FROM Addresses WHERE City NOT IN ('Warszawa', 'Kraków', 'Wrocław')

Aby lepiej zrozumieć wpływ wielkości zbioru danych na wydajność analizowanego systemu testy zostały przeprowadzone z wykorzystaniem 3 zestawów danych o różnej liczbie rekordów opisanej w tabeli 2 z uwzględnieniem poszczególnych tabel bazy.

Za najbardziej znaczącą uznano tabelę Jobs, odzwierciedlającą umowy o pracę zawarte z pracownikami przedsiębiorstwa. Przechowuje ona informacje, o tym który pracownik pracuje na jakim stanowisku oraz do którego działu należy. Rozszerza ona dodatkowo te dane o informacje takie jak: tytuł pracownika w firmie, informację czy pracownik wykonuje swoje obowiązki zdalnie, a także o wysokość jego miesięcznej wypłaty. Dlatego też w dalszej części artykułu tabela Jobs nazywana jest tabelą główną, a liczba rekordów w niej przechowywanych jest równoznaczna do poziomu wypełnienia bazy.

Tabela 2. Liczebność rekordów w tabelach

Nazwa Tabeli	Liczba rekordów		
	Zestaw 1	Zestaw 2	Zestaw 3
Addresses	1000	100000	1000000
Employees	1000	100000	1000000
Jobs	1000	100000	1000000
Positions	1000	100000	1000000
Contacts	1100	100000	1000000
ContactTypes	3	3	3
Departments	100	100	100
DepartmentContacts	100	100	100
EmployeeContacts	1000	100000	1000000

Kolejnym istotnym czynnikiem analizowanym w badaniach jest wpływ parametrów środowiska uruchomieniowego na wydajność obsługi zapytań. W celu uwzględnienia go wykorzystane zostało oprogramowanie do konteneryzacji Docker. Tabela 3 zawiera opis 3 konfiguracji testowych użytych w badaniach oraz wartości 2 parametrów testowych dla każdej z tych konfiguracji.

Tabela 3. Konfiguracje parametrów testowych

Konfiguracja	Ilość pamięci RAM	Maksymalne dopuszczalne wykorzystanie CPU
Konfiguracja 1	2 GB	20%
Konfiguracja 2	4 GB	40%
Konfiguracja 3	6 GB	60 %

Parametry ustawiane były podczas uruchamiania kontenera. Do ustawienia wybranej wartości pamięci wykorzystano flagę --memory. Domyślnie kontener Dockerowy ma nieograniczony dostęp do zasobów

procesora maszyny, na której jest uruchomiony. Dlatego na potrzeby testów wykorzystano parametr --cpus, określający maksymalne wykorzystanie procesora przez uruchamiany kontener [14].

Do przeprowadzenia badań wykorzystany został komputer o następujących parametrach technicznych:

- Procesor: Intel i5 7300hq,
- GPU: NVIDIA GeForce GTX 1050 Mobile,
- RAM: 16 GB 3200 MHz,
- Dysk: SSD NVMe 3200 MB/s / 1900 MB/s.

6. Analiza wyników

W celu uniknięcia wpływu losowych zakłóceń na uzyskane wyniki, każde z badań przeprowadzono 100 razy. Uzyskane wyniki poddano obróbce, usuwając z nich po 3 wartości skrajne. Pozostałe wyniki zostały przeanalizowane i zaprezentowane w dalszej części artykułu.

6.1. Wyniki dla zbioru 1 000 rekordów

W przypadku najmniejszego zestawu danych testowych oraz najsłabszych parametrów sprzętowych wszystkie bazy uzyskały bardzo dobre wyniki, co pokazuje tabela 4. Różnica w wynikach pomiędzy systemami jest nieznaczna, jednak przy 9 z 12 badanych scenariuszy najlepiej wypadł PostgreSQL. Z kolei dla zapytań wykorzystujących instrukcje UNION ALL i NOT IN lepszy wynik uzyskał MySQL. MS SQL wygrał w scenariuszu z użyciem NOT EXISTS.

Tabela 4. Wyniki dla konfiguracji testowej 1

Scenariusz	Średni czas [ms]		
	MySQL	PostgreSQL	MS SQL
Scenariusz I	10.9	7.1	7.9
Scenariusz II	9.2	5.6	12.7
Scenariusz III	17.1	8.8	15.4
Scenariusz IV	19.3	7.8	15.3
Scenariusz V	14.2	7.5	10.9
Scenariusz VI	15.7	8.1	12.4
Scenariusz VII	8.5	7.7	7.1
Scenariusz VIII	16.7	9.5	23.6
Scenariusz IX	14.9	8.3	36.1
Scenariusz X	13.9	20.0	17.7
Scenariusz XI	17.9	7.0	11
Scenariusz XII	10.7	12	14

Tabela 5. Wyniki dla konfiguracji testowej 2

Scenariusz	Średni czas [ms]		
	MySQL	PostgreSQL	MS SQL
Scenariusz I	2.9	1.8	3.3
Scenariusz II	2.5	1.5	2.5
Scenariusz III	6.1	3.8	6.8
Scenariusz IV	4.2	2.4	3.7
Scenariusz V	3.7	2.0	3.1
Scenariusz VI	3.7	2.0	3.1
Scenariusz VII	3.1	2.0	5.9
Scenariusz VIII	4.3	3.0	6.4
Scenariusz IX	3.3	2.4	5.4
Scenariusz X	4.4	4.9	4.6
Scenariusz XI	3.2	2.7	3.2
Scenariusz XII	4.6	5.0	4.1

Ponownie dla najmniejszego zbioru danych, w przypadku konfiguracji 2 uzyskano bardzo zbliżone wyniki dla wszystkich analizowanych systemów, co obrazują wyniki zaprezentowane w tabeli 5. Przeciętny czas operacji we wszystkich analizowanych scenariuszach był rzędu kilku milisekund. Z niewielką przewagą w 10 analizowanych zapytaniach najwydajniejszy okazał się PostgreSQL.

Tabela 6. Wyniki dla konfiguracji testowej 3

Scenariusz	Średni czas [ms]		
	MySQL	PostgreSQL	MS SQL
Scenariusz I	2.5	1.6	2.4
Scenariusz II	2.0	1.4	2.0
Scenariusz III	4.6	3.5	5.0
Scenariusz IV	3.8	2.3	2.5
Scenariusz V	4.2	2.1	2.8
Scenariusz VI	3.2	1.9	2.4
Scenariusz VII	2.5	1.7	3.2
Scenariusz VIII	3.3	2.1	3.3
Scenariusz IX	2.8	1.9	2.9
Scenariusz X	3.6	3.8	2.2
Scenariusz XI	3.1	1.9	3.8
Scenariusz XII	4.7	3.5	4.8

Dla konfiguracji testowej 3, tak jak w przypadku poprzedniej, testy przy wypełnieniu bazy na poziomie 1000 rekordów zakończyły się wynikami rzędu kilku milisekund oraz niemal niezauważalnymi różnicami pomiędzy badanymi systemami. Ponownie najwydajniejszy okazał się w tym przypadku PostgreSQL, co można zaobserwować w tabeli 6.

6.2. Wyniki dla zbioru 100 000 rekordów

W przypadku drugiego zestawu danych testowych oraz konfiguracji 1 uzyskano bardziej zróżnicowane czasy pomiarów, co jest widoczne w tabeli 7. W znacznej większości scenariuszy najgorzej wypadł MySQL uzyskując znacząco większe czasy od pozostałych systemów. Bardzo dobrze radził sobie PostgreSQL, notując najlepsze wyniki w 8 z 12 przypadków testowych. W pozostałych 4 najlepiej wypadł MS SQL.

Tabela 7. Wyniki dla konfiguracji testowej 1

Scenariusz	Średni czas [ms]		
	MySQL	PostgreSQL	MS SQL
Scenariusz I	547.2	249.8	28.6
Scenariusz II	320.8	65.7	308.6
Scenariusz III	4621.0	1603.3	2031.8
Scenariusz IV	1363.8	559.1	600.4
Scenariusz V	1861.5	350.9	270.6
Scenariusz VI	1188.4	235.0	310.7
Scenariusz VII	912.5	380.7	736.6
Scenariusz VIII	1512.0	900.7	432.7
Scenariusz IX	589.2	250.7	450.6
Scenariusz X	1231.1	2485.9	99.6
Scenariusz XI	467.9	181.4	111.6
Scenariusz XII	1350.4	1042.6	39.9

Tabela 8. Wyniki dla konfiguracji testowej 2

Scenariusz	Średni czas [ms]		
	MySQL	PostgreSQL	MS SQL
Scenariusz I	216.3	98.3	3.1
Scenariusz II	123.7	32.7	86.9
Scenariusz III	2641.1	622.0	1102.8
Scenariusz IV	524.7	272.6	321.4
Scenariusz V	665.6	109.4	129.5
Scenariusz VI	434.8	98.0	147.4
Scenariusz VII	447.4	127.9	239.9
Scenariusz VIII	625.5	300.4	146.9
Scenariusz IX	272.5	69.2	131.0
Scenariusz X	635.7	683.9	30.8
Scenariusz XI	211.9	64.3	54.2
Scenariusz XII	626.3	480.7	4.3

Również dla tej konfiguracji testowej zwiększenie liczby rekordów testowych spowodowało wydłużenie czasów badanych operacji oraz zwiększenie różnic pomiędzy analizowanymi systemami, co widać w tabeli 8. Dla zbioru 100 000 oraz konfiguracji numer 2 w 11 z 12 scenariuszy testowych najgorszy wynik odnotował MySQL. W 6 najwydajniejszy okazał się PostgreSQL. Nietypowe wyniki uzyskano w scenariuszu X, gdzie PostgreSQL okazał się najwolniejszy, a znaczną przewagą wygrał MS SQL.

Przy wypełnieniu tabel na poziomie 100 000 oraz konfiguracji 3 bardzo słabe wyniki uzyskał MySQL, który zajął ostatnie miejsce w przypadku 10 analizowanych scenariuszy. Znacznie lepsze wyniki uzyskał PostgreSQL, który okazał się najwydajniejszy w 7 z 12 przeprowadzonych testów. Średnie czasy uzyskane w tym zestawieniu zaprezentowano w tabeli 9.

Tabela 9. Wyniki dla konfiguracji testowej 3

Scenariusz	Średni czas [ms]		
	MySQL	PostgreSQL	MS SQL
Scenariusz I	142	89	8
Scenariusz II	108	20	91
Scenariusz III	1682	518	994
Scenariusz IV	325	199	232
Scenariusz V	376	95	93
Scenariusz VI	237	63	106
Scenariusz VII	179	65	188
Scenariusz VIII	288	189	117
Scenariusz IX	119	57	116
Scenariusz X	275	540	27
Scenariusz XI	121	52	9
Scenariusz XII	137	33	35

6.3. Wyniki dla zbioru 1 000 000 rekordów

Zwiększenie rozmiaru bazy do poziomu 1 000 000 rekordów oraz wybranie parametrów technicznych z zestawu 1 podkreśliło różnice między analizowanymi systemami, co pokazują wyniki zanotowane w tabeli 10. Wyraźną przewagę uzyskał w tym przypadku MS SQL. Czasy operacji dla wszystkich analizowanych systemów uległy znacznemu wydłużeniu. Najgorsze wyniki uzyskała baza MySQL.

Tabela 10. Wyniki dla konfiguracji testowej 1

Scenariusz	Średni czas [s]		
	MySQL	PostgreSQL	MS SQL
Scenariusz I	8.6	3.9	0.003
Scenariusz II	3.4	1.6	2.5
Scenariusz III	98.6	22.9	6.2
Scenariusz IV	29.7	7.0	4.8
Scenariusz V	57.0	4.4	2.4
Scenariusz VI	22.8	3.1	3.4
Scenariusz VII	32.2	14.6	3.6
Scenariusz VIII	39.5	14.1	1.9
Scenariusz IX	6.8	2.5	1.8
Scenariusz X	24.0	27.2	1.0
Scenariusz XI	3.8	2.1	0.4
Scenariusz XII	14.1	11.7	0.03

W przypadku konfiguracji testowej 2 testy dla największego z rozpatrywanych zestawów danych ponownie podkreśliły różnice pomiędzy systemami, co jest widoczne w tabeli 11. Nieunikniony okazał się również wzrost czasu wykonania operacji względem mniejszych rozmiarów bazy, przy zachowaniu identycznych parametrów testowych. W przypadku 10 scenariuszy najszybszy okazał się MS SQL. Dla 9 rozpatrywanych zapytań najwolniejszy okazał się MySQL, dla 3 PostgreSQL.

Tabela 11. Wyniki dla konfiguracji testowej 2

Scenariusz	Średni czas [s]		
	MySQL	PostgreSQL	MS SQL
Scenariusz I	4.7	1.6	0.004
Scenariusz II	1.2	1.0	1.2
Scenariusz III	39.3	9.6	2.3
Scenariusz IV	9.5	2.7	1.8
Scenariusz V	19.0	2.3	0.9
Scenariusz VI	7.6	1.1	1.2
Scenariusz VII	14.7	9.4	1.2
Scenariusz VIII	21.5	7.2	0.6
Scenariusz IX	3.0	1.6	0.5
Scenariusz X	10.8	16.1	0.3
Scenariusz XI	1.1	1.1	0.2
Scenariusz XII	3.0	5.5	0.006

Wyniki w przypadku 3 konfiguracji testowej dla zbioru 1 000 000 rekordów ponownie pokazały, że baza MySQL bardzo słabo radzi sobie z dużą liczbą rekordów. Zajęła ona ostatnie miejsce w 10 z 12 rozpatrywanych scenariuszy. Z kolei najlepszy okazał się MS SQL, który był najszybszy dla 9 zapytań testowych. Wyniki uzyskane w tej serii testów zaprezentowano w tabeli 12.

Tabela 12. Wyniki dla konfiguracji testowej 3

Scenariusz	Średni czas [s]		
	MySQL	PostgreSQL	MS SQL
Scenariusz I	4.1	1.1	0.006
Scenariusz II	0.83	0.76	1.02
Scenariusz III	22.1	7.0	2.3
Scenariusz IV	7.7	1.6	1.9
Scenariusz V	10.7	1.6	1.0
Scenariusz VI	4.6	0.7	1.3
Scenariusz VII	7.5	4.1	1.4
Scenariusz VIII	10.9	3.7	0.8
Scenariusz IX	1.5	1.3	0.8
Scenariusz X	3.9	9.3	0.4
Scenariusz XI	0.87	0.61	0.14
Scenariusz XII	3.510	2.634	0.005

7. Podsumowanie

W niniejszym artykule dokonano porównania trzech relacyjnych baz danych: MySQL, MS SQL oraz PostgreSQL pod względem ich wydajności. W porównaniu uwzględniono wpływ parametrów technicznych środowiska testowego, znaczenie wielkości zbioru danych testowych oraz szereg różnorodnych scenariuszy testowych. Na tej podstawie można wyciągnąć następujące wnioski.

Nie istnieje jeden uniwersalny i najlepszy system bazodanowy. W zależności od testowanego zapytania, parametrów technicznych, czy rozmiaru przechowywanych danych testowych poszczególne bazy danych mogą osiągać lepsze lub gorsze wyniki.

Dla małych zbiorów danych różnice w uzyskiwanych czasach wynoszą około 1 ms lub nawet mniej. Jest to różnica, która z punktu widzenia człowieka nie jest zauważalna. W takiej sytuacji czas wykonania zapytań nie jest czynnikiem decydującym o wyborze systemu bazodanowego podczas tworzenia aplikacji.

Wówczas należy rozważyć dodatkowe czynniki takie jak dostępność programistów specjalizujących się w danym języku, ceny licencji, czy dostępne funkcjonalności rozważanych systemów.

W przypadku średnich baz danych widoczne jest duże zróżnicowanie wyników w zależności od analizowanego scenariusza oraz znaczna poprawa wydajności w przypadku poprawy parametrów technicznych. W takiej sytuacji warto rozważyć dostępne zasoby sprzętowe oraz konkretne zapytania, które najczęściej będą wykonywane przez tworzoną aplikację.

Podczas testów przeprowadzanych dla dużych zbiorów danych tak jak w przypadku zestawu 3 widać było wyraźną różnicę w wydajności badanych systemów. W większości analizowanych scenariuszy najlepsze czasy uzyskał MS SQL. Uzyskał on niewielką przewagę nad PostgreSQL. Najsłabiej wypadł w tym przypadku MySQL.

Uzyskane wyniki są dość oczekiwane oraz odpowiadają one wynikom przedstawionych w pracach innych autorów [1-4]. Dodatkowo zgodnie z przewidywaniami można zaobserwować, że w przypadku zwiększania wielkości zbioru danych testowych czasy wykonywania zapytań operacji zwiększają się. Analogicznie poprawa parametrów technicznych środowiska uruchomieniowego prowadzi do zwiększenia wydajności, w przypadku każdego z analizowanych systemów bazodanowych.

Literatura

[1] R. Wodyk, M. Skublwska-Paszkowska, Performance comparison of relational databases SQL Server, MySQL and PostgreSQL using a web application and the Laravel framework, Journal of Computer Sciences Institute 17 (2020) 358-364, https://doi.org/10.35784/jcsi.2279.

[2] G. Dziewit, J. Korczyński, M. Skublewska-Paszkowska, Performance analysis of relational databases Oracle and MS SQL based on desktop application, Journal of Computer Sciences Institute 8 (2018) 263-269, https://doi.org/10.35784/jcsi.693.

[3] S. Stets, G. Kozieł, Analiza porównawcza baz danych – praca dyplomowa magisterska, Politechnika Lubelska, Lublin 2019.

[4] K. Krocz, O. Kizun, M. Skublewska-Paszkowska, Analiza wydajności relacyjnych baz danych MySQL, PostgreSQL, MariaDB oraz H2 – praca dyplomowa magisterska, Politechnika Lubelska, Lublin 2020.

[5] S. Tongkaw, A. Tongkaw, A comparison of database performance of MariaDB and MySQL with OLTP workload, IEEE Conference on Open Systems, Langkawi 2016, 117-119.

[6] C. -O. Truica, F. Radulescu, A. Boicea and I. Bucur, Performance Evaluation for CRUD Operations in Asynchronously Replicated Document Oriented Database, 20th International Conference on Control Systems and Computer Science, Bucharest 2015, 191-196.

[7] P. Beynon-Davies, Systemy baz danych, Wydawnictwa Naukowo Techniczne, 2003.

[8] Strefa wiedzy Vavatech, https://vavatech.pl/technologie/bazy-danych/mysql, [10.02.2023].

[9] MySQL - Wikipedia, https://en.wikipedia.org/wiki/MySQL, [10.02.2023].

[10] Opis oprogramowania PostgreSQL, https://www.postgresql.org/about/, [12.02.2023].

[11] PostgreSQL - Amazon AWS, https://aws.amazon.com/rds/postgresql/what-is-postgresql/, [12.02.2023].

[12] Microsoft SQL Server - Wikipedia, https://pl.wikipedia.org/wiki/Microsoft_SQL_Server, [20.04.2023].

[13] Opis systemu MS SQL na portalu Atlantic, https://www.atlantic.net/vps-hosting/what-is-mssql/, [21.04.2023].

[14] Konfiguracja kontenera Docker, https://www.baeldung.com/ops/docker-memory-limit, [21.03.2023].

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 28 (2023) 197–203
Received: 24 May 2023
Accepted: 21 June 2023

Performance optimization of web applications using Qwik

Optymalizacja wydajności aplikacji internetowych z wykorzystaniem Qwik

Adam Lipiński*, Beata Pańczyk

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

Abstract
This article analyzes the performance of three frameworks - React.js, Next.js and Qwik - that offer different methods of rendering application views. The purpose of the study was to show whether the new Qwik framework allows for better application load times compared to the other frameworks. The study was conducted using 3 applications representing the same research content, referring to cases occurring in production environments. In order to assess the performance, the Google Lighthouse tool was used, thanks to which it was proved that it is impossible to unequivocally say that Qwik allows for better optimization of the application compared to other frameworks.

Keywords: optimization; React; Next; Qwik

Streszczenie
W niniejszym artykule przeprowadzono analizę wydajnościową trzech szkieletów programistycznych - React.js, Next.js oraz Qwik - oferujących różne metody renderowania widoków aplikacji. Celem badania było wykazanie, czy nowy szkielet Qwik pozwala na uzyskanie lepszych wyników czasów ładowania aplikacji, w porównaniu z pozostałymi szkieletami. Badanie przeprowadzono z wykorzystaniem 3 aplikacji reprezentujących tę samą treść badawczą, nawiązującą do przypadków występujących w środowiskach produkcyjnych. W celu oceny wydajności wykorzystano narzędzie Google Lighthouse, dzięki czemu dowiedziono, że nie da się jednoznacznie stwierdzić, aby Qwik pozwalał na lepszą optymalizację aplikacji w porównaniu z pozostałymi szkieletami.

Słowa kluczowe optymalizacja; React; Next; Qwik

*Corresponding author
Email address: adam.lipinski@pollub.edu.pl (A. Lipiński)

1. Introduction

Along with the development and popularization of modern technologies used to create the client part of websites and web applications, programmers were encouraged to transfer some operations to browsers. Increasingly, MPA (Multi Page Applications) were abandoned in favor of SPA (Single Page Applications) which allowed for increased smoothness of operation, as well as avoiding constant reloading of the page with each user's action. Initially, the idea was to upload a nearly empty HTML file and fill it with content using code that is executed when the page loads. This method was named CSR (Client Side Rendering) because all views were rendered on the client's browser side.

The increased amount of often computationally demanding code contributed to the search for new solutions in the field of rendering application views, as well as optimization in terms of their loading speed. This brought many changes in this area which made the SSR (Server Side Rendering) method more and more often used in comparison to CSR. This method differed significantly from Client Side Rendering because the initial view of the application sent in response to the client's request was rendered on the server. As a result, the application loading speed was significantly improved and the user could see its content faster. However, this is not without some losses - the user does see the page faster, but there is no possibility to interact with it as long as the hydration process is not completed.

Hydration is the process by which application code is loaded in the background when a view is displayed. In this process, all listeners for any events are attached to the static page and the code required when the application is run for the first time is executed. As it has already been mentioned, applications very often began to grow very large so the time between the first page view and the moment when the user could start using it was getting longer and longer.

Therefore, many developers began to look for a way to reduce the impact of hydration on page loading. There have been many approaches to this problem but one of the last ones that started to get community acknowledgement was the use of the resumability method. It was proposed by Misko Hevery, the creator of one of the currently most popular programming frameworks used to create web applications - Angular. This method assumes that operations on the server are suspended and the process is resumed in the same place but on the client side, without the need to re-create and download the entire application logic. It has been implemented in the constantly developed new Qwik framework which, according to people working on it, is to change the quality of web applications and solve the problem that the Internet has to face due to the hydration process [1].

Due to the fact that this technology is very young, there are practically no scientific sources on it. To fill this gap this article will examine how the Qwik framework handles the day-to-day cases CSR and SSR applications face. The collected results will concern application rendering times and delays. They will be compared with the results for the widely used React.js and Next.js frameworks which will make it possible to determine whether Qwik can really become a new direction in the development of web applications in the future.

2. Purpose and scope of the paper

The aim of the paper is to examine and compare three programming frameworks: Qwik, React.js, and Next.js. Frameworks will be compared based on application rendering times and delays.

The scope of the paper includes: analysis of literature on research oscillating in the subject of optimization of web applications, selection of research methods, development of research scenarios, preparation of applications in each of the analyzed frameworks, research implementation, data analysis, and formulation of conclusions.

For the purposes of the work, the following research hypotheses have been set which will be tested thanks to the performed analysis:

H1. The Qwik framework allows to reduce the loading time of the web application compared to the React.js and Next.js frameworks.
H2. The Qwik framework allows for faster rendering of applications with a different number of elements compared to the React.js and Next.js frameworks.
H3. The Qwik framework allows for more efficient loading of a large number of larger-sized resources compared to the React.js and Next.js frameworks.
H4. The Qwik framework allows for more efficient execution of more demanding code compared to the React.js and Next.js frameworks.

3. Literature review

Although the technology analyzed in this article is completely new, comparing the optimality and speed of programming frameworks is a frequent topic of scientific papers and publications.

In the research conducted in articles [2, 3] the authors tried to determine which of the analyzed frameworks - React.js, Next.js, and Vue.js - is better for tasks such as rendering a different number of elements in the DOM (Document Object Model) tree. By comparing several of the same applications written in different programming frameworks, the authors concluded that in most cases React.js performed the best in both studies.

In addition, the frameworks compared in publications [4-7] offered different methods of rendering application views. On this basis, the researchers wanted to indicate which of the methods allows for the best results in terms of response time for a specific test case. In most scenarios, rendering views on the server offered better performance results, compared to rendering on the client side.

Particularly noteworthy are the articles [8, 9] in which the researchers, more than on the frameworks themselves, focused on various techniques for optimizing web applications. In these papers, one application was profiled in terms of performance and then subjected to speed optimization methods. Thanks to that it was possible to indicate that selecting the appropriate rendering method can significantly affect the speed of the application.

A frequent topic of consideration was also the development of tools and test frameworks allowing for easier and more accurate profiling of applications as well as identifying areas where improvements could be made to increase the speed of their operation which was analyzed by researchers in articles [10-17]. In the majority of instances, the prepared test environments were characterized by high accuracy of results, compared to other widely used testing software. Moreover, the sheer multitude of publications on this subject proves the general interest in the subject of performance optimization.

Yet another approach to the topic of application optimization was adopted by researchers in a publication [18] in which they tried to check how adequate the results proposed by automatic performance testing tools are to the real, empirical feelings of users from using web applications. Their task was also to indicate which metrics best reflect the experience of using the application. The obtained results showed that the FCP (First Contentful Paint) time, which is an integral metric accessible within the Google Lighthouse performance assessment, very clearly translated into the actual perception of the website's performance.

Among the newer sources, the subject of application performance is also very often mentioned which can be read in the article [19] prepared by researchers from Google and Shopify. In this article, they described the advantages and disadvantages of various application rendering methods and also explained how the choice of technology, and thus the rendering method, affects performance as well as SEO (Search Engine Optimization). An important part of the article is a fragment devoted to the problem of hydration occurring in modern programming frameworks prepared for developing client parts of web applications that offer the possibility of rendering views on the server.

The last of the analyzed articles focused entirely on considerations very close to the subject of this thesis. In the paper [20] it was explained what the resumability method utilized by the Qwik framework is and how it works. The author emphasized the problems and limitations caused by the hydration process and conducted a discussion on how the mentioned method can affect their solution. The challenges faced by the use of this framework, and thus this method in production applications, were also identified.

4. Research method

For this study, three identical applications with a minimalist graphical interface were prepared in each tested

programming framework. The research is aimed at checking application rendering times for each of these technologies in various test cases. The analyzed times are: FCP (First Contentful Paint), TBT (Total Blocking Time), LCP (Largest Contentful Paint), and SI (Speed Index). The results were obtained through the utilization of data gathered from the Google Lighthouse tool. In order to ensure consistency and accuracy, caching of data in the browser was disabled for each of the conducted tests. Furthermore, 50 trials were undertaken for every test to increase reliability and validity of the results.

The prepared test cases performed in this study are as follows:

1. Rendering applications with a large number of elements in the DOM tree:
   a) 100 elements,
   b) 1000 elements,
   c) 10000 elements.
2. Rendering applications with a large number of resources of large sizes - images with a size of several MB:
   a) 100 images,
   b) 1000 images,
   c) 10000 images.
3. Rendering applications that perform expensive operations and processes data - downloading a large amount of external data then processing it and displaying it on the screen.

4.1. Testing platform

For the purpose of conducting performance audits, the following testing platform was used:

- web browser: Google Chrome 109.0.5414.75,
- OS: MacOS Monterey 13.1,
- CPU: Intel Core i5 1.4GHz quad-core,
- RAM: 16GB 2133 MHz LPDDR3,
- GPU: Intel Iris Plus Graphics 645 1536 MB,
- model: Macbook Pro 13 2020.

4.2. Description of the experiment

To achieve the research objective an experiment was designed for which the research scenarios presented in Tables 1-3 were determined**.**

Table 1: A research scenario for examining the rendering times of an application with a large number of elements in the DOM tree

Name: Rendering an application with a large number of elements in the DOM tree		
Aim: Verifying how a large number of elements in the DOM affects the rendering speed of the application		
Initial conditions: The application is ready to load, the number of elements in the tree is set to the initial value - 100		
Final conditions: App load time data has been saved		
Next steps		
No.	Description of activities	Expected result
1.	Loading the application	The application has loaded
2.	Saving the value of the loading times	The data has been saved
3.	The app is loaded another 49 times	The application has loaded and the results have been saved
4.	Changing the number of rendered items to 1000	The number of items has been changed
5.	Repeating points 1-3 for a new number of elements	The application has loaded and the results have been saved
6.	Changing the number of rendered items to 10000	The number of items has been changed
7.	Repeating points 1-3 for a new number of elements	The application has loaded and the results have been saved

Table 2: Research scenario for examining the rendering times of an application with a large number of displayed photos

Name: Rendering an application which loads large numbers of images of significant size		
Aim: Verifying how loading and displaying a large number of photos of significant sizes affects application rendering times		
Initial conditions: The application is ready to load, the number of photos to download is set to the initial value - 10		
Final conditions: App load time data has been saved		
Next steps		
No.	Description of activities	Expected result
1.	Loading the application	The application has loaded
2.	Saving the value of the loading times	The data has been saved
3.	The app is loaded another 49 times	The application has loaded and the results have been saved
4.	Changing the number of photos displayed to 100	The number of photos has been changed
5.	Repeating points 1-3 for a new number of photos	The application has loaded and the results have been saved
6.	Changing the number of photos displayed to 1000	The number of photos has been changed
7.	Repeating points 1-3 for a new number of photos	The application has loaded and the results have been saved

Table 3: Research scenario for testing the rendering times of an application that downloads and processes a large amount of data

Name: Rendering an application which performs expensive operations and data processing - downloading a large amount of external statistical data, then processing, sorting and displaying it on the screen in the form of a graph
Aim: Verifying how downloading and processing large amounts of external data affects application rendering speed
Initial conditions: The application is ready to load
Final conditions: App load time data has been saved

Next steps		
No.	Description of activities	Expected result
1.	Loading the application	The application has loaded
2.	Saving the value of the loading times	The data has been saved
3.	The app is loaded another 49 times	The application has loaded and the results have been saved

5. Results

The results of the audits conducted for the three prepared tests are presented in Tables 4-11. Additionally, Figures 1-7 showcase graphs that display the collected results, categorized into tests and frameworks.

It is important to mention that the LCP times were not measurable using the Google Lighthouse tool for the third test, and hence, they were not considered in the analysis of the application's performance for this particular test. The absence of outcomes concerning this metric could potentially be attributed to the particularity of this test.

Table 4 Mean metric values for test one - 100 elements

Metric	React	Next	Qwik
FCP [s]	1.40±0.02	1.11±0.04	0.80±0.00
LCP [s]	1.40±0.02	1.83±0.06	0.80±0.00
TBT [ms]	0.00±0.00	0.01±2.40	0.00±0.00
SI [s]	1.40±0.02	1.11±0.04	0.80±0.00

Table 5: Mean metric values for test one - 1000 elements

Metric	React	Next	Qwik
FCP [s]	1.40±0.00	1.11±0.03	0.90±0.00
LCP [s]	1.40±0.00	1.86±0.05	0.90±0.00
TBT [ms]	11.20±7.18	9.40±2.40	0.00±0.00
SI [s]	1.40±0.00	1.11±0.03	0.90±0.00

Table 6: Mean metric values for test one - 10000 elements

Metric	React	Next	Qwik
FCP [s]	1.40±0.00	1.78±0.05	1.31±0.06
LCP [s]	1.40±0.00	1.96±0.24	1.45±0.06
TBT [ms]	177.80±7.90	158.20±38.42	309.80±64.29
SI [s]	1.40±0.00	1.82±0.07	1.31±0.06

Table 7 Mean metric values for test two - 10 photos

Metric	React	Next	Qwik
FCP [s]	1.40±0.00	1.10±0.02	0.80±0.00
LCP [s]	8.25±0.78	5.32±0.74	7.61±0.76
TBT [ms]	449.99±346.34	22.80±29.97	176.98±336.80
SI [s]	2.26±0.16	1.12±0.05	2.12±0.50

Table 8 Mean metric values for test two - 100 photos

Metric	React	Next	Qwik
FCP [s]	1.40±0.00	1.10±0.01	0.80±0.00
LCP [s]	7.97±0.56	5.56±0.34	7.89±0.67
TBT [ms]	384.73±313.05	23.60±34.74	62.08±213.68
SI [s]	2.37±0.18	1.12±0.09	2.14±0.27

Table 9: Mean metric values for test two - 1000 photos

Metric	React	Next	Qwik
FCP [s]	1.40±0.00	1.40±0.00	0.90±0.01
LCP [s]	10.39±0.56	5.07±1.49	27.11±4.09
TBT [ms]	2.24±0.44	19.80±34.26	1.95±0.46
SI [s]	5.27±0.72	1.46±0.15	5.42±0.85

Table 10: Mean metric values for test three

Metric	React	Next	Qwik
FCP [s]	1.40±0.00	1.10±0.00	1.42±0.07
LCP [s]	N/A	N/A	N/A
TBT [ms]	109.35±6.80	130.80±12.09	89.40±28.24
SI [s]	5.88±1.48	1.43±0.05	12.15±4.76

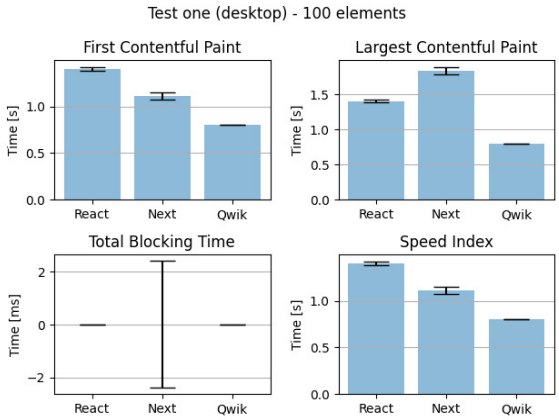


Figure 1: Mean metric values in graphical form obtained in test one for 100 elements.

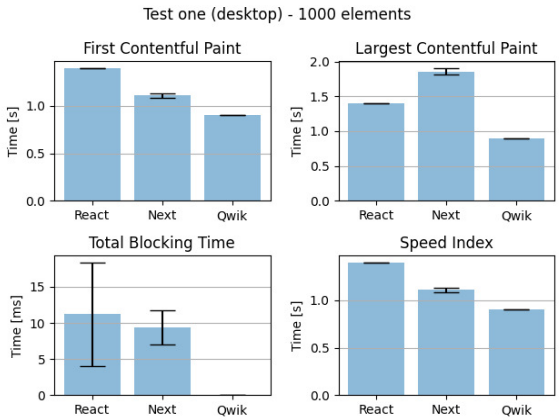


Figure 2: Mean metric values in graphical form obtained in test one for 1000 elements.

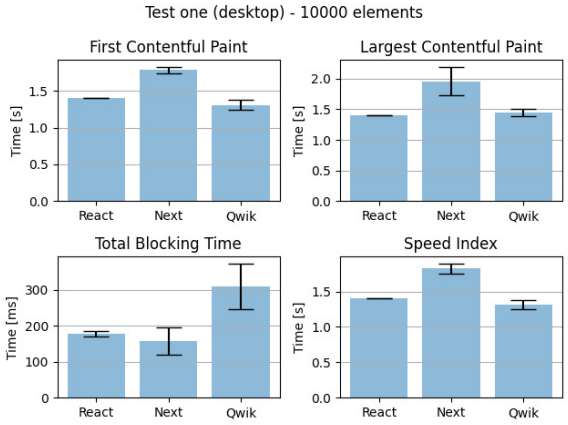


Figure 3: Mean metric values in graphical form obtained in test one for 10000 elements.

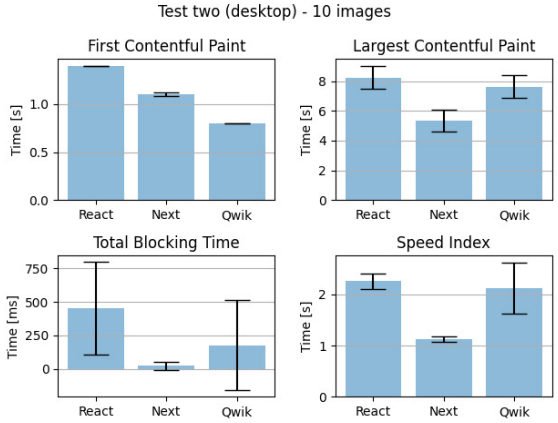


Figure 4: Mean metric values in graphical form obtained in test two for 10 photos.

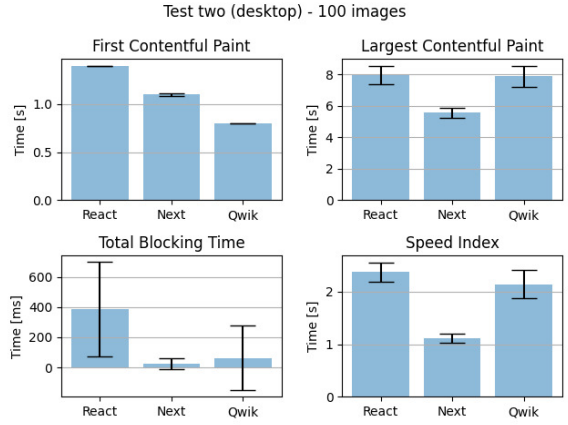


Figure 5: Mean metric values in graphical form obtained in test two for 100 photos.

Upon analyzing the results obtained from the research, it has become apparent that the metric values in individual tests differed significantly from one another. Furthermore, it is worth noting that in most cases, within a given study, analyzed technologies maintained the same trends, thereby providing consistent results, and the frameworks' ranking remained relatively stable throughout the study.

For test one Qwik had the best FCP and LCP times for each test case. Visually the content was also visible the fastest in the case of Qwik. For the time of blocking user interaction the greatest discrepancies can be observed - this time was the largest for Next for a small number of elements, for React.js for the average number of elements, and for Qwik it was the largest for the largest number of elements. Overall, Qwik is the best scorer according to Lighthouse's score calculation [21] but it's not a landslide victory.

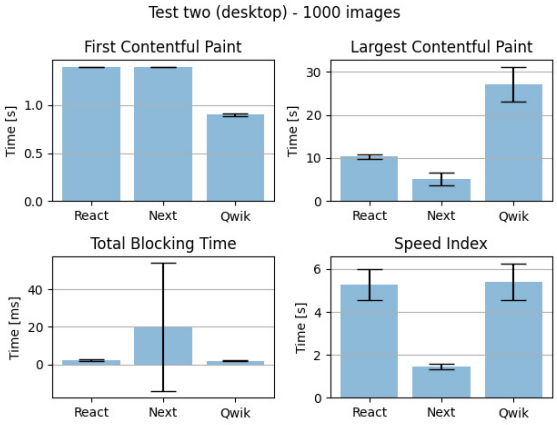


Figure 6: Mean metric values in graphical form obtained in test two for 1000 photos.

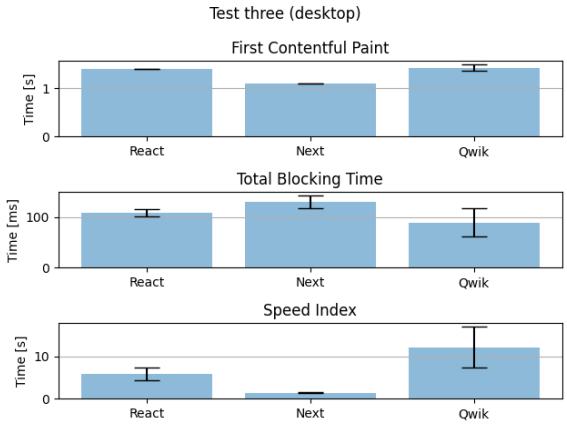


Figure 7: Mean metric values in graphical form obtained in test three.

In the second test once again the lowest FCP time was achieved by Qwik. It can be seen, however, that the situation has changed for LCP and the lowest values were obtained by Next.js while Qwik was classified at the end of the ranking. Next.js also had the best TBT results but was overtaken by Qwik and React.js for the large number of images displayed. For the SI metric, the best results were achieved by Next.js. Qwik and React have kept these times on a similar level with Qwik leading slightly. To sum up this study, and considering that Lighthouse's TBT and LCP [21] times are the most important for performance, Next.js performed significantly better than Qwik in this test while React.js performed the worst.

In the case of the third test, Qwik obtained the best values for the TBT metric but it also had the largest fluctuations of values as evidenced by the high value of

the standard deviation. However, it performed the worst in other metrics where Next.js took the lead and once again the standard deviation for the results obtained by Qwik was the largest.

6. Conclusions

The obtained test results show that each of the tests performed was a different challenge for the analyzed frameworks. The first test went the best, as evidenced by the fact that each of the frameworks obtained satisfactory results allowing for passing the Lighthouse audit with a high score of application optimization. As expected, the increase in the number of rendered elements resulted in an increase in application load times. The biggest surprise in this case, however, was the sudden jump in interaction blocking time for the Qwik framework for the largest number of elements which nevertheless did not stop it from showing that it was the best among the analyzed frameworks at the task of displaying a large number of elements in the DOM tree, thus confirming hypothesis H2.

In general, the results were the worst for the second test which was also characterized by the lowest convergence between the values from subsequent trials. In this case, Next.js showed the best of the frameworks, which may be the result of the use of the Image component which allows optimizing the displayed images [22]. Therefore, Qwik did not obtain results allowing to conclude that it has a chance to improve the performance of an application loading a large number of resources with increased sizes, thus refuting hypothesis H3.

The third test assumed some disadvantages in favor of the Qwik framework - the library used to display the graphs is a React library so Next.js and React.js can use it directly and Qwik must additionally convert the component used from it. Despite the apparent injustice, it must be borne in mind that this framework is new and if we want to use it in a production environment we cannot count on developers to create from scratch every library that has already been created in React.js, especially having the possibility to use it in an application written in Qwik. Therefore, this study made perfect sense as this is a typical case that can occur in a production environment. This probably influenced the result of the study which showed that the application created in Qwik performed worse than other applications, and thus caused hypothesis H4 to be rejected.

Summing up all tests, the application written in Qwik received better metrics results only in the first test so it was not possible to fully demonstrate that this framework allows for better application loading times which was the subject of hypothesis H1.

References

[1] Presentation by M. Hevery on “Qwik + Partytown: How to remove 99% of JavaScript from main thread” at WeAreDevelopers World Congress 2022, https://www.youtube.com/watch?v=0dC11DMR3fU, [13.06.2023].

[2] C. M. Novac, O. C. Novac, R. M. Sferle, M. I. Gordan, G. Bujdosó, C. M. Dindelegan, Comparative study of some applications made in the Vue.js and React.js frameworks, 16th International Conference on Engineering of Modern Electric Systems (EMES), (2021) 1-4, https://doi.org/10.1109/EMES52337.2021.9484149.

[3] Z. Dinku, React.js vs. Next.js, Metropolia University of Applied Sciences, (2022), https://www.theseus.fi/bitstream/handle/10024/750122/Dinku_Zerihun.pdf.

[4] A. Świątkowski, K. Ścibior, Comparative analysis of React, Next and Gatsby programming frameworks for creating SPA applications, Journal of Computer Sciences Institute, 24 (2022) 224-227, https://doi.org/10.35784/jcsi.2972.

[5] T. Fadhilah Iskandar, M. Lubis, T. Fabrianti Kusumasari, A. Ridho Lubis, Comparison between client-side and server-side rendering in the web development, IOP Conference Series: Materials Science and Engineering, 801 (2020) 1-6, https://doi.org/10.1088/1757-899X/801/1/012136.

[6] M. Hakim, Speed index and critical path rendering performance for isomorphic single page applications, Proceedings of the 16th Winona Computer Science Undergraduate Research Seminar, (2016) 41-46, https://cs.winona.edu/cs-website/current_students/Projects/CSConference/2016conference.pdf.

[7] N. K. SG, P. K. Madugundu, J. Bose, S. C. S. Mogali, A Hybrid Web Rendering Framework on Cloud, 2016 IEEE International Conference on Web Services (ICWS), IEEE, (2016) 602-608, https://doi.org/10.1109/ICWS.2016.83.

[8] F. Pavić, L. Brkić, Methods of Improving and Optimizing React Web-applications, 44th International Convention on Information, Communication and Electronic Technology (MIPRO), (2021) 1753-1758, https://doi.org/10.23919/MIPRO52101.2021.9596762.

[9] J. Väyrynen, Ensuring Availability of a Server-Side Rendered React Application: A Case Study, Aalto University, (2019), http://urn.fi/URN:NBN:fi:aalto-201905122998.

[10] A. M. Aladwani, An empirical test of the link between website quality and forward enterprise integration with web consumers, Business Process Management Journal, Emerald Publishing, 12 (2) (2006) 178-190, https://doi.org/10.1108/14637150610657521.

[11] A. M. Aladwani, P. C. Palvia, Developing and validating an instrument for measuring user-perceived web quality, Information & Management, Elsevier, 39 (6) (2002) 467-476, https://doi.org/10.1016/S0378-7206(01)00113-6.

[12] F. Almeida, J. Monteiro, The role of responsive design in web development, Webology, Webology Center, 14 (2) (2017) 48-651, http://www.webology.org/2017/v14n2/a157.pdf.

[13] G. Richards, A. Gal, B. Eich, J. Vitek, Automated construction of JavaScript benchmarks, ACM SIGPLAN, 46 (10) (2011) 677-693, https://doi.org/10.1145/2076021.2048119.

[14] H. Findel, J. Navon, A Test Environment for Web Single Page Applications (SPA), In Proceedings of the 11th International Conference on Web Information Systems and Technologies - WEBIST, (2015) 47-54, https://doi.org/10.5220/0005428000470054.

[15] H. Golestani, S. Mahlke, S. Narayanasamy, Characterization of Unnecessary Computations in Web Applications, IEEE International Symposium on Performance Analysis of Systems and Software, (2019) 11-21, https://doi.org/10.1109/ISPASS.2019.00010.

[16] K. Kiyokawa, Q. Jin, A Front-End Framework Selection Assistance System with Customizable Quantification Indicators Based on Analysis of Repository and Community Data, Big-Data-Analytics in Astronomy, Science, and Engineering, Lecture Notes in Computer Science, Springer, 13167 (2022) 41-55, https://doi.org/10.1007/978-3-030-96600-3_4.

[17] M. Kaluža, K. Troskot, B. Vukelić, Comparison of Front-End frameworks for web applications development, Journal of the Polytechnic of Rijeka, 6 (1) (2018) 261-282, https://doi.org/10.31784/zvr.6.1.19.

[18] L. Borzemski, M. Kędras, Measured vs. Perceived Web Performance, Information Systems Architecture and Technology: Proceedings of 40th Anniversary International Conference on Information Systems Architecture and Technology – ISAT 2019, ISAT 2019, Advances in Intelligent Systems and Computing, Springer, 1050 (2019) 285-301, https://doi.org/10.1007/978-3-030-30440-9_27.

[19] J. Miller, A. Osmani, Rendering on the Web, Google Developers, web.dev, (2019), https://web.dev/rendering-on-the-web/, [20.03.2023].

[20] R. Carniato, Resumable JavaScript with Qwik, DEV Community, (2022), https://dev.to/this-is-learning/resumable-javascript-with-qwik-2i29, [20.03.2023].

[21] How the Performance score is weighted - Lighthouse 10, https://developer.chrome.com/docs/lighthouse/performance/performance-scoring/#lighthouse-10, [25.04.2023].

[22] Next.js Documentation - Image Component and Image Optimization, https://nextjs.org/docs/basic-features/image-optimization, [25.04.2023].

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 28 (2023) 204–209
Received: 24 May 2023
Accepted: 25 June 2023

Analyze the effectiveness of ETL processes implemented using SQL and Apache HiveQL languages

Analiza efektywności procesów ETL realizowanych z użyciem języków SQL i Apache HiveQL

Krzysztof Damian Litka*

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

Abstract

In the era of digitization, where data is collected in ever-increasing quantities, efficient processing is required. The article analyzes the performance of SQL and HiveQL, for scenarios of varying complexity, focusing on the execution time of individual queries. The tools used in the study are also discussed. The results of the study for each language are summarized and compared, highlighting their strengths and weaknesses, as well as identifying their possible areas of application.

Keywords: ETL; SQL; HiveQL

Streszczenie

W dobie cyfryzacji, gdzie dane gromadzone są w coraz większych ilościach, wymagane jest ich efektywne przetwarzanie. W artykule dokonano analizy wydajności języka SQL i HiveQL, dla scenariuszy o zróżnicowanym stopniu złożoności, skupiając się na czasie wykonania poszczególnych zapytań. Omówiono także wykorzystane w badaniu narzędzia. Wyniki badań dla poszczególnych języków zostały zestawione i porównane, podkreślając ich mocne i słabe strony, a akże określając ich możliwe obszary zastosowań.

Słowa kluczowe: ETL; SQL; HiveQL

*Corresponding author

Email address: s99174@pollub.edu.pl (K. D. Litka)

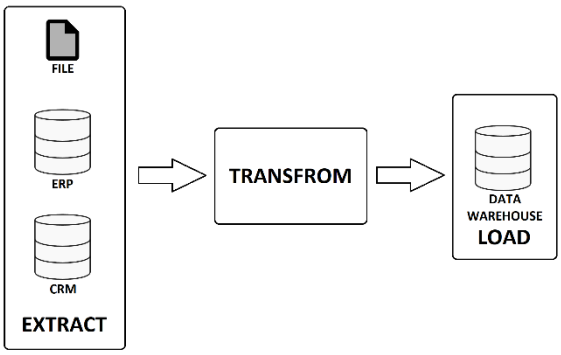
1. Wstęp

W dobie gwałtownego wzrostu ilości danych generowanych przez różne systemy informatyczne, coraz bardziej kluczowe staje się efektywne zarządzanie tymi danymi, ich analiza i przetwarzanie. To powoduje, że procesy ETL (Extract, Transform, Load), które służą do pobierania, transformacji i ładowania danych z różnych źródeł do magazynu danych, stają się coraz bardziej istotne. Kluczowym aspektem procesów ETL jest nie tylko skuteczność, ale także zdolność do szybkiego przetwarzania dużych ilości danych, przy minimalnym wykorzystaniu zasobów. SQL, będący standardowym językiem do zarządzania danymi przechowywanymi w relacyjnych bazach danych, jest powszechnie stosowany w tradycyjnych procesach ETL. Z drugiej strony, Apache HiveQL, będący częścią ekosystemu Hadoop i zapewniający interfejs zapytań podobnych do SQL, jest coraz częściej wykorzystywany w procesach ETL związanych z Big Data [2, 9].

Celem tego artykułu jest szczegółowe zbadanie i porównanie efektywności dwóch popularnych języków zapytań - SQL i HiveQL – w kontekście różnorodnych scenariuszy procesów ETL. Te scenariusze zostały zaprojektowane tak, aby odzwierciedlać typowe przypadki użycia w biznesie, obejmujące różne stopnie złożoności. Przez przeprowadzenie serii eksperymentów, porównana została wydajność tych dwóch języków, ale także przeanalizowano różne funkcje i ograniczenia związane z ich użyciem. W pracy tej zwrócono szczególną uwagę na wpływ różnych czynników, takich jak rozmiar danych wejściowych i złożoność zapytań, na efektywność procesów ETL. W tym kontekście, badania te stanowią istotny wkład do zrozumienia, jak te czynniki mogą wpływać na wybór odpowiedniego języka zapytań dla konkretnego zadania czy projektu. Ponadto, praca ta jest zakorzeniona w istniejącej literaturze teoretycznej dotyczącej procesów ETL oraz w literaturze dotyczącej wydajności narzędzi, które są wykorzystywane w tych procesach. Zdobyta wiedza teoretyczna posłużyła jako fundament do projektowania i przeprowadzania eksperymentów. Praca ta ma na celu nie tylko poszerzenie zrozumienia związku między teoretycznymi aspektami procesów ETL, a praktycznymi zastosowaniami, ale także dostarczenie konkretnych danych dotyczących wydajności SQL i HiveQL w różnych kontekstach [2, 8, 9, 12, 13].

2. Definicja ETL

Procesy ETL, to procesy związane z przetwarzaniem danych, które polegają na pobraniu danych z różnych źródeł, ich przekształceniu oraz załadowaniu do docelowego systemu przechowywania, co pozwala na ich łatwiejsze zarządzanie, analizowanie oraz udostępnianie. [2]. Schemat obrazujący proces ETL przedstawiono na Rysunku 1.



Rysunek 1: Schemat procesu ETL.

Ekstrakcja polega na pobieraniu danych z różnorodnych źródeł, takich jak bazy danych, pliki tekstowe, CRM, SCM, ERP i inne aplikacje biznesowe. Etap ten musi być skalowalny i elastyczny, aby dostosować się do zmieniających się wymagań. Transformacja przekształca dane, aby były zgodne z wymaganiami systemu docelowego. Proces ten może obejmować czyszczenie danych, konwersję typów danych, normalizację, mapowanie wartości, filtrowanie, agregację i scalanie danych. Ostatni etap, ładowanie, polega na przekazaniu danych do systemu docelowego, gdzie są one indeksowane, sortowane i przygotowane do analizy [7-9].

3. Opis środowiska badawczego

Procesy ETL można realizować za pomocą wielu narzędzi. W badaniu wykorzystane zostały takie platformy jak Hadoop, Apache Hive, Cloudera, SQL Server i SQL Server Management Studio. Oraz język SQL i HiveQL.

3.1. Narzędzia

1. Hadoop to otwartoźródłowy framework, stworzony do przetwarzania i przechowywania dużych zbiorów danych na klastrach komputerów. Składa się z dwóch głównych komponentów: Hadoop Distributed File System (HDFS) do przechowywania danych na wielu maszynach i MapReduce do przetwarzania danych w sposób równoległy i rozproszony [4, 13].
2. Apache Hive, opracowany przez Apache Software Foundation, jest narzędziem do tworzenia, implementacji i zarządzania magazynami danych na platformie Hadoop. Obsługuje przetwarzanie danych w sposób rozproszony na wielu węzłach, a także różne formaty danych [1].
3. Cloudera to dystrybucja Hadoop oferująca różne narzędzia do zarządzania i przetwarzania danych. Wśród nich znajdują się Cloudera Data Platform (CDP) do zarządzania danymi na różnych platformach, Cloudera Data Engineering (CDE) do tworzenia, wdrażania i zarządzania procesami ETL, a także Cloudera Manager do zarządzania i monitorowania klastrów Hadoop [3].
4. SQL Server, zaawansowany system zarządzania relacyjnymi bazami danych firmy Microsoft, używany jest do przechowywania, manipulowania i zarządzania danymi. Sprawdza się zarówno w procesach szybkiego przetwarzania transakcji (OLTP), jak i skomplikowanej analizy danych (OLAP) [2, 5, 8].
5. SQL Server Management Studio (SSMS) to narzędzie do zarządzania infrastrukturą SQL. Wykorzystywane jest do zarządzania danymi, indeksami, użytkownikami i ich rolami, a także do tworzenia zapytań, debugowania, tworzenia i zarządzania schematami oraz instancjami serwerów [2, 5, 8].

3.2. Języki

SQL (Structured Query Language) jest językiem używanym do interakcji z relacyjnymi bazami danych. Jego podstawową strukturą są zapytania, które służą do wykonywania określonych działań na danych. SQL jest podzielony na cztery główne podzbiory: DDL (Data Definition Language), DML (Data Manipulation Language), DQL (Data Query Language) i DCL (Data Control Language), każdy odpowiedzialny za różne operacje na danych, od definicji i modyfikacji struktury bazy danych, poprzez manipulację danymi, po zarządzanie uprawnieniami dostępu. Język SQL posiada również funkcje agregujące oraz różne słowa kluczowe i operatory, które umożliwiają tworzenie skomplikowanych zapytań [5, 6].

HiveQL, język zapytań wysokiego poziomu, został stworzony do współpracy z Apache Hive. Jest strukturalnie i składniowo podobny do SQL, chociaż nie obsługuje wszystkich funkcji SQL, takich jak operacje modyfikacji danych czy transakcje. Natomiast posiada unikalne funkcje, które go wyróżniają. Klauzule takie jak PARTITION BY i CLUSTER BY, które umożliwiają partycjonowanie i klastrowanie danych, są szczególnie ważne dla efektywnego przetwarzania dużych zbiorów danych, zgodnie z modelowaniem danych w ekosystemie Hadoop. HiveQL, oparty na MapReduce, umożliwia efektywne przetwarzanie dużych zbiorów danych w rozproszonych środowiskach [1, 10, 11].

3.3. Platforma systemowa i sprzętowa

Badania przeprowadzono w dwóch hermetycznych środowiskach testowych z użyciem maszyn wirtualnych VMware. Pierwsze środowisko skupiało się na języku SQL i wykorzystywało Microsoft SQL Server 2019 Express oraz SQL Server Management Studio do zarządzania komponentami SQL Server. Środowisko testowe zostało postawione na systemie operacyjnym Linux Red Hat 6 64-bit uruchomionym na maszynie wirtualnej ze 150 GB dysku twardego NVMe M2, 3-rdzeniowym procesorem o taktowaniu 3.8 GHz (5.1 GHz w boost) i 16 GB pamięci RAM o taktowaniu 3600 MHz. Natomiast drugie środowisko testowe utworzono dla HiveQL, języka zapytań wzorowanym na SQL, zaprojektowanym dla środowiska Hadoop. Wykorzystano tutaj dystrybucję Hadoop od Cloudera w wersji 2.5.0, która jest zoptymalizowana i łatwa w obsłudze. Środowisko to było uruchomione na identycznym sprzęcie co środowisko SQL. Obie konfiguracje umożliwiały efektywne przeprowadzenie badań i porównanie wydajności obu środowisk w procesach ETL.

4. Struktura bazy danych

W badaniu utworzona została baza danych typu Data Warehouse, jaką jest AdventureWorksDW. Baza została zaprojektowana zgodnie z modelem Star Schema. Baza składa się z tabel wymiarów (Dim) i faktów (Fact), które są powiązane relacjami. Tabele wymiarów zawierają atrybuty służące do analiz, np. informacje o klientach. Tabele faktów natomiast, zawierają metryki poddawane analizie, np. dane o sprzedaży. W środowisku SQL Server tabele wymiarów i faktów są połączone za pomocą kluczy, tworząc strukturę gwiazdy. W środowisku Hadoop, nie ma bezpośrednich połączeń pomiędzy tabelami, gdyż Hive nie obsługuje takich mechanizmów relacyjnych jak klucze główne i klucze obce [8].

Baza danych AdventureWorksDW, powstała na podstawie relacyjnej bazy danych AdventureWorks2019, przedstawiającej działalność fikcyjnej firmy. Zawiera ona szczegółowe dane dotyczące procesów biznesowych firmy, reprezentowane przez szereg powiązanych tabel.

5. Implementacja

5.1. Różnice w implementacji zapytań SQL i HiveQL

Z faktu, iż HiveQL bazuje na podstawach języka SQL, ich składnia w zapytaniach jest niemal identyczna. Jednakże przy implementacji zapytań, jakie wykorzystano w badaniu, występuj pewne różnice.

W SQL, aby uniknąć niejednoznaczności i błędów wynikających z użycia słów kluczowych jako nazw kolumn, umieszcza się nazwy kolumn w nawiasach kwadratowych []. Natomiast w ekosystemie Hadoop, używając HiveQL, nazwy kolumn w zapytaniach nie muszą być umieszczane w nawiasach kwadratowych [1, 2, 5, 8]. Przykład tej metody użycia przedstawiono na Listingu 1.

Listing 1: Użycie nawiasów kwadratowych w SQL

```
SELECT DISTINCT
    st.TerritoryID AS SalesTerritoryAlternateKey,
    st.[Name] AS SalesTerritoryRegion,
    st.CountryRegionCode AS SalesTerritoryCountry,
    st.[Group] AS SalesTerritoryGroup
FROM AdventureWorks2019.Sales.SalesTerritory AS st
```

W SQL podczas ładowania danych do tabeli, wykorzystuje się klauzulę INSERT INTO, natomiast HiveQl użyta musi być ta klauzula, poszerzona o słowo TABLE: INSERT INTO TABLE. Sposób w jaki dane są dodawane do tabeli może się różnić pomiędzy tymi dwoma językami [1, 2, 5, 8].

W SQL, do generowania unikalnych identyfikatorów dla wierszy (często dla kolumn będących kluczem głównym) używa się właściwości IDENTITY. SQL Server automatycznie zwiększa wartość w tej kolumnie, za każdym razem, gdy dodawany jest nowy wiersz. HiveQL nie ma natomiast takiej funkcji. W zastępstwie, wykorzysta została funkcja DENSE_RANK(), która przypisuje rangę dla każdego wiersza w partycji danych, sortując je na podstawie określonych kolumn, a przy pomocy klauzuli DISTINCT, otrzymane zapytania są unikalne. Zarówno SQL, jak i HiveQL korzystają z klauzuli DISTINCT do eliminowania powtarzających się wierszy [1, 2, 5, 8]. Listing 2 przedstawia przykład użycie DENSE_RANK() w HiveQL do wygenerowania unikalnej wartości.

Listing 2: Generowanie unikalnej wartości w HiveQL

```
INSERT INTO TABLE AdventureWorksDW.DimProduct
SELECT DISTINCT
    DENSE_RANK() OVER(ORDER BY p.ProductID) AS PoductKy,
```

Aktualna data i czas w języku SQL, są zwykle pobierane za pomocą funkcji GETDATE(). Działanie tego polecenia jest proste i zwraca aktualną datę i czas serwera SQL. Natomiast w wykorzystywanej w badaniu wersji HiveQL, dla uzyskania aktualnej daty i czasu, potrzebne jest wykorzystanie kombinacji dwóch funkcji: UNIX_TIMESTAMP() i FROM_UNIXTIME(). UNIX_TIMESTAMP() zwraca aktualny czas Unix (liczbę sekund od 1970-01-01 00:00:00 UTC), a FROM_UNIXTIME() konwertuje ten czas Unix na czytelny format daty [1, 2, 5, 8]. Przykład pobrania aktualnej daty w języku HiveQL przedstawiono na Listingu 3.

Listing 3: Pobranie aktualnej daty w HiveQL

```
WHERE edh.enddate IS NULL OR
edh.enddate >= date(from_unixtime(unix_timestamp()));
```

SQL zawiera wbudowane funkcje takie jak DATEPART(), DATENAME(), i DATEADD(), które są wygodne do manipulowania danymi daty i czasu. W HiveQL, takie funkcje nie są bezpośrednio dostępne. Dlatego, aby uzyskać takie same wyniki, wykorzystane zostały kombinacje funkcji, takich jak FROM_UNIXTIME(), UNIX_TIMESTAMP(), TO_DATE(), i CAST() [1, 2, 5, 8]. Przykład użycia tych funkcji w HiveQL, przedstawiono na Listingu 4. Ta część zapytania wyciąga z wartości w kolumnie „orderdate" kolejno numer dnia, numer miesiąca, numer dnia w roku oraz numer tygodnia w roku.

Listing 4: Manipulacja datą w HiveQl

```
CAST(FROM_UNIXTIME(UNIX_TIMESTAMP(orderdate, 'yyyy-MM-dd'),
                   'd') AS INT) as daynumberofmonth,
CAST(FROM_UNIXTIME(UNIX_TIMESTAMP(orderdate, 'yyyy-MM-dd'),
                   'D') AS INT) as daynumberofyear,
CAST(FROM_UNIXTIME(UNIX_TIMESTAMP(orderdate, 'yyyy-MM-dd'),
                   'w') AS INT) as weeknumberofyear
```

5.2. Metoda wykonania badań

W każdym z badanych scenariuszy, zapytanie zostało wykonane dziesięć razy w celu uzyskania wiarygodnej próbki czasu wykonania. W celu zminimalizowania wpływu innych czynników na wyniki, przed każdym wykonaniem zapytania, odpowiednie tabele zostały oczyszczone za pomocą polecenia TRUNCATE TABLE. Dodatkowo w środowisku SQL Server, pamięć podręczna i bufor danych zostały oczyszczone za pomocą poleceń DBCC FREEPROCCACHE i DBCC DROPCLEANBUFFERS, aby upewnić się, że żadne wcześniejsze zapytania nie wpłyną na wyniki. W przypadku HiveQL, wyczyszczone zostały tabele, ale z powodu braku funkcji do czyszczenia pamięci podręcznej i bufora danych, tego kroku nie wykonano [5].

5.3. Scenariusze badawcze

W scenariuszu pierwszy, wyekstrahowano dane do dwóch tabel wymiaru, pochodzące z 5 tabel z bazy danych AdventureWorks2019. W tabelach docelowych otrzymano kolejno 37 i 504 rekordy.

W scenariuszu drugim do ekstrakcji użyto 15 tabel z AdventureWorksDW i 1 tabeli, która została utworzono na początku tego scenariusza i osadzona w AdventureWorksDW. Dane zostały załadowane do 3 tabel wymiaru w AdventureWorksDW, które otrzymały kolejno: 683, 10 i 20058 rekordów.

W scenariuszu trzecim dane są pobierane z 19 tabel z AdventureWorks2019. Trzy docelowe tabele wymiaru umieszczone w AdventureWorksDW otrzymują kolejno: 290, 1530 i 1127 rekordów.

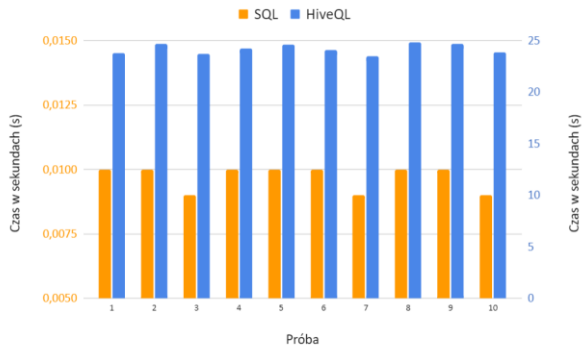
W scenariuszu czwartym uzupełniono tabelę faktu w AdventureWorksDW. Dane wyekstrahowano z 5 tabel z bazy danych AdventureWorks2019 i 5 tabel z AdventureWorksDW. Do tabeli faktu załadowano 7789440 rekordów.

W scenariuszu piątym również została utworzona tabela faktu. Dane do niej zostały pobrane z 7 tabel z bazy danych AdventureWorks2019 i 5 tabel z AdventureWorksDW. Do tabeli docelowej w wyniku zapytania załadowano 7730944 rekordów.

6. Wyniki

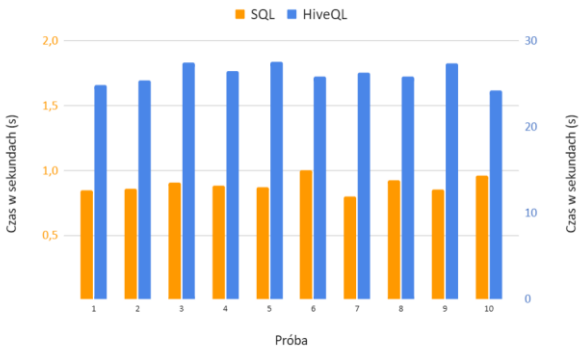
6.1. Analiza i porównanie wyników badań

W pierwszym scenariuszu, zapytanie SQL wykonuje się niezwykle szybko, potrzebując jedynie 0,01 sekundy, podczas gdy HiveQL wymaga znacznie więcej czasu - aż 24,2 sekundy. Jest to ogromna różnica, z HiveQL wykonującym to samo zapytanie aż 2420 razy wolniej niż SQL. Zestawienie wyników z tego scenariusza pokazuje Rysunek 2.



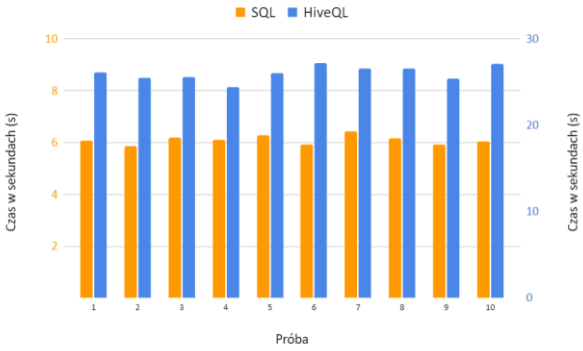
Rysunek 2: Zestawienie wyników z scenariusza pierwszego.

W drugim scenariuszu, SQL nadal przewyższa HiveQL, choć różnica nie jest już tak drastyczna. SQL kończy zadanie w ciągu 0,89 sekundy, podczas gdy HiveQL potrzebuje 26,13 sekundy do wykonania tego samego zadania. Mimo, iż różnica jest mniejsza, HiveQL jest wciąż około 29 razy wolniejszy. Zestawienie wyników z tego scenariusza pokazuje Rysunek 3.



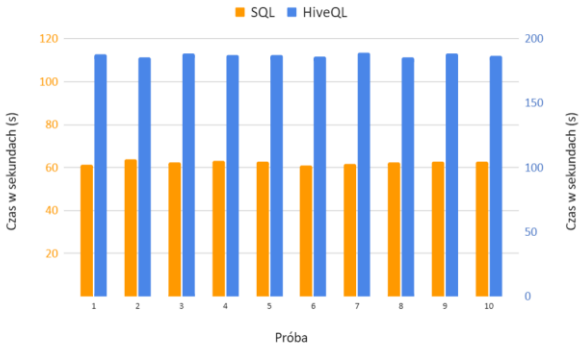
Rysunek 3: Zestawienie wyników scenariusza drugiego.

W trzecim scenariuszu, SQL nadal zdecydowanie przewyższa HiveQL. Zapytanie SQL kończy się w ciągu 6,1 sekundy, a zapytanie HiveQL wymaga 26,04 sekundy. W tym przypadku, HiveQL jest około 4 razy wolniejszy. Zestawienie wyników z tego scenariusza pokazuje Rysunek 4.



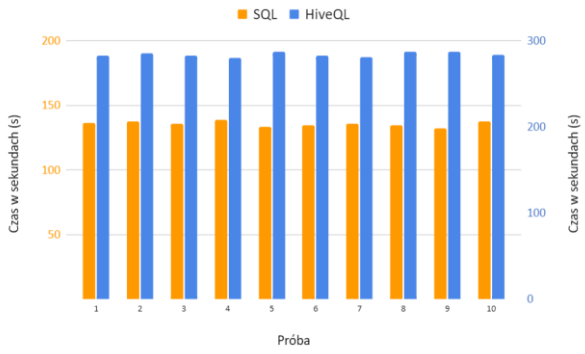
Rysunek 4: Zestawienie wyników z scenariusza trzeciego.

Czwarty scenariusz pokazuje jeszcze bardziej zauważalne różnice. SQL kończy zadanie w ciągu 62,3 sekundy, a HiveQL potrzebuje aż 187,33 sekundy na wykonanie tego samego zadania. To oznacza, że HiveQL jest około 3 razy wolniejszy. Zestawienie wyników z tego scenariusza pokazuje Rysunek 5.



Rysunek 5: Zestawienie wyników z scenariusza czwartego.

W scenariuszu piątym, SQL wykonuje zadanie w ciągu 135,66 sekundy, podczas gdy HiveQL wymaga 284,19 sekundy. HiveQL jest w tym przypadku około 2 razy wolniejszy. Zestawienie wyników z tego scenariusza pokazuje Rysunek 6.



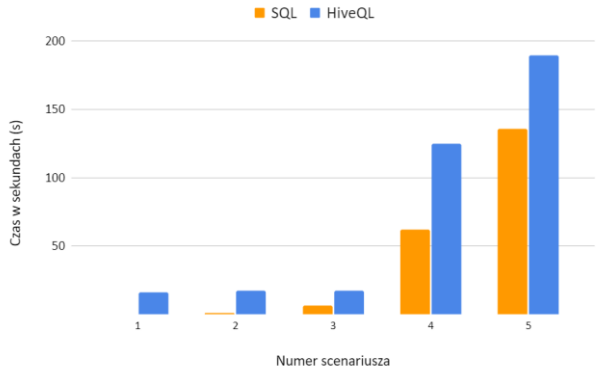
Rysunek 6: Zestawienie wyników z scenariusza piątego.

Dla każdego scenariusza wyliczono średnią z czasów otrzymanych z każdej próby poszczególnego scenariusza. Wyniki przedstawiono w Tabeli 1.

Tabela 1: Średni czas wykonania scenariuszy w 10 próbach

Scenariusz	Czas SQL (s)	Czas HiveQL (s)
1	0,01	24,2
2	0,89	26,13
3	6,1	26,04
4	62,3	187,33
5	135,66	284,19

Dane z Tabeli zostały zestawione w postaci wykresu, na Rysunku 7.



Rysunek 7: Porównanie średniego czasu wykonania scenariuszy dla SQL i HiveQL.

Podsumowując, wyniki testów wydajności jasno pokazują, że SQL jest znacznie szybszy w wykonaniu zapytań dla wszystkich pięciu scenariuszy. Różnice w wydajności między SQL, a HiveQL są szczególnie wyraźne dla prostszych zapytań, a złożoność zapytania wydaje się zwiększać wydajność HiveQL w porównaniu do SQL, choć nadal pozostaje on znacznie wolniejszy. Te wyniki podkreślają znaczenie dobrego zrozumienia narzędzi, które wybieramy do przetwarzania i analizy naszych danych, a także tego, jak złożoność zapytania może wpływać na wydajność tych narzędzi.

7. Dyskusja

W zarządzaniu danymi kluczową rolę pełnią procesy ETL, które umożliwiają przetwarzanie i analizę dużych zbiorów danych. Dwa popularne w tym zakresie języki to SQL i HiveQL, oba posiadające swoje unikalne zalety i wady.

SQL, będący stabilnym standardem branżowym, jest wszechstronny dzięki kompatybilności z wieloma systemami zarządzania bazami danych. Wykazuje efektywność w obsłudze różnych zapytań, szczególnie na mniejszych zestawach danych. Dodatkowo, doskonale integruje się z innymi technologiami i narzędziami analitycznymi. Wśród jego wad należy wymienić koszty związane z niektórymi systemami zarządzania bazami danych oraz potencjalne problemy z przetwarzaniem i analizą bardzo dużych zbiorów danych.

Z kolei HiveQL, część ekosystemu Hadoop, wyróżnia się skalowalnością umożliwiającą obsługę Big Data. Jest projektem open source, co minimalizuje bezpośrednie koszty, chociaż niektóre dystrybucje Hadoop mogą wiązać się z kosztami licencyjnymi. HiveQL jest podobny do SQL, co ułatwia naukę tego języka, ale może być trudniejszy w zarządzaniu i utrzymaniu, szczególnie dla osób bez doświadczenia. Przetwarzanie danych jest wolniejsze niż w SQL dla prostych zapytań i na mniejszych zestawach danych.

W kontekście operacji na pojedynczych wierszach, SQL umożliwia precyzyjne manipulowanie danymi dzięki funkcjom takim jak UPDATE, DELETE, INSERT, skomplikowanym funkcjom skalarnym oraz pętlom i instrukcjom warunkowym. HiveQL, zorientowany na przetwarzanie dużych zestawów danych w sposób równoległy, nie oferuje tych samych możliwości. W tym języku zmiany w danych są zazwyczaj dokonywane przez przetwarzanie całego zestawu danych, a wyniki są zapisywane do nowego zestawu danych.

W artykule Data Processing in Hive vs. SQL Server: A comparative analysis in the query performance, autorzy przeprowadzają badanie przy użyciu Hive oraz SQL Server. Badanie to wykazuje, że SQL Server jest wydajniejszy przy wykonywaniu operacji w czasie rzeczywistym. Natomiast Hive jest efektywniejszy pod względem czasu wykonania zapytania, wraz ze wzrostem rozmiaru danych do przetworzenia. Jak piszą autorzy, Hive jest odpowiedni przy przetwarzaniu dużej ilości rozległych zbiorów danych, które mogą obejmować nawet setki tysięcy maszyn [14].

8. Wnioski

Podczas testów wydajności, SQL okazał się szybszy we wszystkich pięciu scenariuszach testowych, szczególnie dla mniej skomplikowanych zapytań. Natomiast wydajność języka HiveQL, zaczęła wzrastać wraz ze złożonością zapytań oraz przy przetwarzaniu większej ilości danych. Przy wyższej złożoności zapytań, czas wykonania zapytania w HiveQL zaczyna się zbliżać do czasu

wykonania zapytania w SQL, co sugeruje, że HiveQL może być bardziej odpowiedni w kontekście przetwarzania bardzo dużych zbiorów, sięgających terabajtów, a nawet petabajtów danych Wybór między SQL a HiveQL uzależniony powinien być od specyfiki zadania i charakterystyki danych.

Literatura

[1] E. Capriolo, D. Wampler, J. Rutherglen, Programming Hive: Data Warehouse and Query Language for Hadoop, O'Reilly Media, 1st edition, 2012.

[2] J. Caserta, R. Kimball, The Data Warehouse ETL Toolkit., Wiley, 2004.

[3] Cloudera Data Platform, https://www.cloudera.com/products/cloudera-data-platform.html, [25.05.2023].

[4] J. Dean, S. Ghemawat, MapReduce: Simplified Data Processing on Large Clusters, Communications of the ACM 51(1) (2008) 107-113, https://doi.org/10.1145/1327452.1327492.

[5] B. Karwin, SQL Antipatterns: Avoiding the Pitfalls of Database Programming, Pragmatic Programmers LLC, The 1st edition 2017.

[6] P. Mellor, SQL and Relational Theory: How to Write Accurate SQL Code, O'Reilly Media Inc., 2011.

[7] B. Oliveira, O. Belo, J. Caldeira, A Systematic Literature Review on Big Data Extraction, Transformation and Loading (ETL), Proceedings of the 2021 Computing Conference Volume 2 held virtually (2021) 308-324, https://doi.org/10.1007/978-3-030-80126-7_24.

[8] A. Pelikant, Hurtownie danych. Od przetwarzania analitycznego do raportowania, Wydanie II, Helion, 2021.

[9] A. Simitsis, P. Vassiliadis, T. Sellis, Optimizing ETL processes in data warehouses, 21st International Conference on Data Engineering (ICDE'05), Tokyo, Japan (2005) 564-575, https://doi.org/10.1109/ICDE.2005.103.

[10] A. Thusoo, J. S. Sarma, N. Jain, Z. Shao, P. Chakka, N. Zhang, Hive - a Petabyte Scale Data Warehouse using Hadoop, 2010 IEEE 26th International Conference on Data Engineering (ICDE 2010), Long Beach, CA USA (2010) 996-1005, https://doi.org/10.1109/ICDE.2010.5447738.

[11] A. Thusoo, J. S. Sarma, N. Jain, Z. Shao, P. Chakka, S. Anthony, H. Liu, P. Wyckoff, R. Murthy, Hive: a warehousing solution over a map-reduce framework, Proceedings of the VLDB Endowment 2(2) (2009) 1626–1629, https://doi.org/10.14778/1687553.1687609.

[12] T. White, Hadoop: The definitive guide, O'Reilly Media Inc., 2012.

[13] P. C. Zikopoulos, C. Eaton, Understanding big data: Analytics for enterprise class Hadoop and streaming data, McGraw-Hill Osborne Media, 2011.

[14] N. Ahmed, S. Ahamed, J. I. Rahim, Data Processing in Hive vs. SQL Server: A comparative analysis in the query performance, 2017 IEEE 3rd International Conference on Engineering Technologies and Social Sciences, Bangkok, Thailand (2017) 1-5, https://doi.org/10.1109/icetss.2017.8324202.

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 28 (2023) 210–216
Received: 24 May 2023
Accepted: 17 June 2023

A comparative analysis of the performance of the relational database and the Hadoop environment in the context of analytical data processing

Analiza porównawcza wydajności relacyjnej bazy danych i środowiska Hadoop w kontekście analitycznego przetwarzania danych

Michał Zadrąg*

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

Abstract
The article presents a detailed comparative analysis of the performance of a Microsoft SQL Server relational database and an Apache Hadoop environment in the context of analytical data processing. The study was carried out by executing more than a dozen research scenarios with different queries on datasets of varying sizes. For each research scenario, the average query execution time on different datasets was compared. Based on the results, it was found that the average execution time of queries from the presented scenarios is significantly shorter in MS SQL Server than in Apache Hadoop.

Keywords: Apache Hadoop; SQL Server; relational database; OLAP

Streszczenie
Artykuł przedstawia szczegółową analizę porównawczą wydajności relacyjnej bazy danych Microsoft SQL Server i środowiska Apache Hadoop w kontekście analitycznego przetwarzania danych. Badanie zostało przeprowadzone poprzez wykonanie kilkunastu scenariuszy badawczych, w których zastosowano różne zapytania na zbiorach danych o zróżnicowanej wielkości. Dla każdego scenariusza badawczego porównywano średni czas wykonania zapytania na różnych zbiorach danych. Na podstawie otrzymanych wyników stwierdzono, że średni czas wykonania zapytań z przedstawionych scenariuszy jest znacznie krótszy w MS SQL Server niż w Apache Hadoop.

Słowa kluczowe: Apache Hadoop; SQL Server; relacyjna baza danych; OLAP

*Corresponding author
Email address: michal.zadrag@pollub.edu.pl (M. Zadrąg)

1. Wstęp

W dzisiejszych czasach dane są najcenniejszym zasobem. Sposób, w jaki są one przechowywane oraz przetwarzane jest bardzo istotny, ponieważ może on znacząco zwiększyć przewagę konkurencyjną danej firmy na rynku. Zgromadzone dane można poddać analitycznemu przetwarzaniu i dzięki temu uzyskać informacje pozwalające wyciągnąć wnioski na interesujący nas temat. Wraz z rozwojem technologii a w szczególności Internetu, wolumen danych generowanych oraz przetwarzanych przez systemy informatyczne znacznie się powiększył.

Analityczne przetwarzanie danych [1] (OLAP, Online Analytical Processing) to technologia pozwalająca na złożoną analizę dużego wolumenu danych. Systemy OLAP zostały zaprojektowane w celu ułatwienia wyodrębniania informacji analizy biznesowej z danych w wysoce wydajny sposób. Narzędzia OLAP umożliwiają tworzenie zaawansowanych raportów i analiz, co pozwala na lepsze zrozumienie danych i lepsze podejmowanie decyzji.

Celem niniejszego artykułu jest przeprowadzenie analizy porównawczej wydajności relacyjnej bazy danych Microsoft SQL Server i środowiska Apache Hadoop w kontekście analitycznego przetwarzania danych. Badanie to ma na celu zidentyfikowanie mocnych stron i słabych stron obu podejść, aby pomóc organizacjom w wyborze odpowiedniego narzędzia w zależności od specyfiki ich danych i wymagań analitycznych. Przyjętą hipotezą badawczą jest: Środowisko Hadoop jest wydajniejszym rozwiązaniem w analitycznym przetwarzaniu danych przy dużych wolumenach danych.

W rozdziale drugim przedstawiony zostanie przegląd literatury wykorzystany podczas tworzenia niniejszej pracy. Rozdział trzeci artykułu przedstawia szczegółowy opis relacyjnej bazy danych SQL Server i środowiska Apache Hadoop, omawia różnice w sposobie przechowywania i przetwarzania danych. Opis środowiska badawczego zawarty będzie w rozdziale numer cztery. Rozdział piąty przedstawi metody badawcze oraz zaprezentuje wybrane scenariusze badawcze, wykorzystane podczas badań. Wyniki otrzymane po wykonaniu scenariuszy badawczych, zostaną przedstawione w rozdziale szóstym. Na podstawie wyników tych eksperymentów dokonana zostanie analiza. Rozdział siódmy zawierał będzie podsumowanie oraz wyciągnięte wnioski.

2. Przegląd literatury

Autorzy w artykule [2] opisują czym według nich jest Big Data. Według nich jest to termin określający olbrzymie, różnorodne i posiadające złożoną strukturę dane. Problemem wynikającym z ilości tych danych jest

zdecydowanie sposób ich przechowywania, bardzo istotne jest, aby wybrać optymalny sposób. Autorzy zwracają również uwagę na to, że aby Big Data była użyteczna należy poddać ją szczegółowej analizie w celu wyciągnięcia wniosków, które pozwolą przekuć zdobytą wiedzę w konkretne działania. Na wyszczególnienie zasługuje przytoczona przez autorów statystyka obrazująca znaczący wzrost liczby generowanych danych: Do roku 2003 ludzkość wygenerowała 5 eksabajtów danych. W 2012 taki wolumen danych był generowany w ciągu dwóch dni. Na te dane składa się cała aktywność w internecie. Tradycyjna analiza danych nie jest w stanie przetworzyć danych z Big Data, dlatego wymagała ona kroku naprzód. Krok ten musiał uwzględniać trzy główne cechy wyróżniające Big Data od zwykłego zbioru danych. Autorzy artykułu zwracają uwagę na fakt, że Big Data spowodowała powstanie zupełnie nowej architektury jaką jest architektura MapReduce. Jako przykład implementacji przytaczają Apache Hadoop.

Artykuł [3] autorstwa Umeshwar Dayal oraz Surajit Chaudhuri, skupia się na tym, czym jest według autorów analityczne przetwarzanie danych OLAP (ang. Online analytical processing) w kontekście hurtowni danych. OLAP to metoda przetwarzania danych która pozwala na wielopoziomową i wielowymiarową analizę ogromnych wolumenów danych, dostarczając przy tym możliwość wizualizacji zbioru danych z różnych perspektyw. Głównymi operacjami w OLAP są: drill-down, umożliwia użytkownikom nawigację z wyższego poziomu szczegółowości do niższego poziomu szczegółowości danych; Roll-up, pozwala użytkownikom na nawigację z niższego poziomu szczegółowości do wyższego poziomu szczegółowości danych; Slice-and-dice, pozwala użytkownikom na selektywne przeglądanie podzbioru danych w oparciu o określone kryteria; Pivot, pozwala użytkownikom obracać dane, aby zobaczyć je z innej perspektywy; Drill-through, pozwala użytkownikom na dostęp do szczegółowych danych dla konkretnego punktu danych; Data Mining, pozwala użytkownikowi na wydobycie wiedzy z danych poprzez zastosowanie technik statystycznych, algorytmów uczenia maszynowego i innych metod; Agregacja, umożliwia użytkownikowi wykonywanie obliczeń na danych, takich jak suma, liczba, średnia itp. Autorzy podkreślają to, że zarządzanie hurtowniami danych oraz OLAP stawia nowe wyzwania przez użytkownikami, nadal nie są to technologie bez wad i istotne jest, aby cały czas pracować nad ich doskonaleniem.

Artykuł o tytule "HaoLap: a Hadoop based OLAP System for Big Data" [4] opisuje autorski system OLAP, z wykorzystaniem Apache Hadoop, dla Big Data. Autorzy opisują w artykule cały proces tworzenia systemu od opracowania architektury, aż po samą implementacje. Istotnym czynnikiem jest fakt, że nowo utworzony system został porównany z istniejącymi już systemami: Hive, HadoopDB, HBaseLattice oraz Olap4Cloud. Całe porównanie opierało się na porównaniu, ładowania, funkcji cube oraz roll–up i przechowywania w kontekście 13 klastrów. Na potrzeby eksperymentu autorzy stworzyli zbiór danych "OceanCube" który zawierał dane oceanograficzne. Wyniki porównania ładowania wykazały, że autorski HaoLap jest o 15 razy szybszy od Olap4Cloud i 16 razy szybszy od HBaseLattice. Drugim kryterium było całkowite zużycie czasu na funkcję cube, zgodnie z wynikami HaoLap wypadł lepiej od wszystkich systemów za wyjątkiem HBaseLattice. Najgorszy w zestawieniu okazało się rozwiązanie HadoopDB. Kolejnym testem była operacja roll–up, wyniki były tożsame z tymi z operacji kostkowania tj. HaoLao wypadł lepiej od wszystkich systemów za wyjątkiem HBaseLattice. Ostatnim testem była wielkość jaka jest potrzebna do przechowywania danych przez dane rozwiązanie. HaoLap i Hive wypadły w tym teście najlepiej, ponieważ oba te rozwiązania nie korzystają z żadnych metadanych. Olap4Cloud natomiast bazuje na HBase który musi przetrzymywać indeksy i metadane. HadoopDB z kolei opiera się na relacyjnej bazie danych.

3. Porównywane technologie

3.1. Microsoft SQL Server

SQL Server [5] jest relacyjnym systemem zarządzania bazą danych firmy Microsoft. Pierwsza wersja systemu pojawiła się w 1993 roku i obsługiwała ówczesny standard języka SQL (SQL – 89). Produkt firmy Microsoft korzysta zarówno z tradycyjnego języka zapytań SQL, jak i z rozszerzenia tego języka o nazwę Transact-SQL. T-SQL rozszerza możliwości zwykłego SQL o takie konstrukcje jak pętle, zmienne oraz instrukcje warunkowe. Dodatkowo istnieje możliwość tworzenia własnych funkcji, procedur składowanych, wyzwalaczy oraz kursorów. Według strony https://db-engines.com/ Microsoft SQL Server jest jednym z najpopularniejszych systemów zarządzania relacyjnymi bazami danych.

3.2. Apache Hadoop

Apache Hadoop [6] został stworzony w celu przetwarzania ogromnych zbiorów danych, nie jest to odpowiednie rozwiązanie dla aplikacji czasu rzeczywistego. Cały ekosystem składa się z kilku głównych modułów, które razem tworzą platformę rozproszonego przetwarzania danych:

- Hadoop Distributed File System (HDFS)
- Hadoop MapReduce
- Hadoop YARN (Yet Another Resource Negotiator)
- Hadoop Common

Hadoop Distributed File System [7] został zaprojektowany do niezawodnego przechowywania bardzo dużych wolumenów danych. Opiera się na zasadzie przechowywania plików na wielu maszynach. Hadoop MapReduce [8] jest implementacja paradygmatu MapReduce do przetwarzania dużych ilości danych. Hadoop YARN [9] to system zarządzania zasobami dla Hadoop, odpowiedzialny za zarządzanie zasobami klastra Hadoop i harmonogramowanie zadań do wykonania na tych zasobach. Hadoop Common jest zbiorem bibliotek będących podstawą dla innych modułów z ekosystemu Hadoopa, zapewnia ich podstawową funkcjonalność

oraz wsparcie co pomaga w zarządzaniu omawianym rozwiązaniem.

4. Opis środowiska

Badania zostały przeprowadzone na wirtualnych maszynach o takich samych parametrach, w celu wyeliminowania zniekształcenia wyników wynikającego z różnych parametrów maszyn, na których przeprowadzano test. Tabela 1 przedstawia specyfikację komputera, na którym zostały uruchomione omawiane środowiska wirtualne.

Tabela 1: Specyfikacja komputera do maszyn wirtualnych

Procesor	12th Gen Intel(R) Core(TM) i5-12500H 1.70 GHz
System operacyjny	Windows 10 Pro
Dysk	Kioxia 512GB SSD
RAM	32 GB

Dla każdego środowiska uruchomiona została osobna maszyna wirtualna przy pomocy Oracle VirtualBox w wersji 6.1. Każda instancja opierała się na dystrybucji systemu operacyjnego Linux, Red Hat i posiadała 16 GB pamięci RAM. Na jednej z maszyn wirtualnych została zainstalowana relacyjna baza danych Microsoft SQL Server, natomiast druga korzysta z gotowej dystrybucji Apache Hadoop o nazwie "Cloudera Quickstart VM".

5. Metody badawcze

W celu porównania wydajności, badanie przeprowadzone zostało w formie wykonania zapytań bazodanowych na odpowiednio przygotowanych bazach danych. Baza składa się z tabel wymiarów (Dim) i faktów (Fact), które są powiązane relacjami. Tabele wymiarów zawierają atrybuty służące do analiz. Tabele faktów natomiast, zawierają metryki poddawane analizie, np. dane o sprzedaży. Baza danych liczy jeden milion rekordów. Po każdym wykonaniu zapytania został zmierzony czas jego wykonania. Na podstawie czasów wykonania poszczególnych zapytań zostanie wyciągnięty średni czas wykonania zapytania dla danego zbioru danych. Każde zapytanie, zostało wykonane na wolumenie danych:

- 250 tysięcy rekordów (Zestaw danych numer 1),
- 500 tysięcy rekordów (Zestaw danych numer 2),
- 750 tysięcy rekordów (Zestaw danych numer 3),
- 1 milion rekordów (Zestaw danych numer 4).

Wielokrotne wykonanie zapytań pozwalało na poddanie wyników obróbce statystycznej. Natomiast różna ilość danych pozwala stwierdzić w jaki sposób zachowują się omawiane rozwiązania zależnie od ilości danych. Przygotowane zostały cztery scenariusze badawcze, na podstawie których zostały przeprowadzone badania. Scenariusze badawcze zostały zdefiniowane w ten sposób, aby sprawdzały możliwie wszystkie sposoby przetwarzania danych przez te rozwiązania. Scenariusz pierwszy zawierał definicje zapytania wykorzystującego polecenie ROLLUP. Omawiane zapytanie zwraca wyniki sprzedaży według kraju i roku, a następnie sumuje je na poziomie kraju i na całkowitym poziomie. Kod zapytania ze scenariusza pierwszego został umieszczony na Listingu 1.

Listing 1: Kod zapytania ze scenariusza pierwszego

```
SELECT
    COALESCE(CONVERT(varchar, g.COUNTRYNAME), 'All Countries') AS Country,
    COALESCE(CONVERT(varchar, dt.CALENDARYEAR), 'All Years') AS Year,
    COUNT(DISTINCT c.CUSTOMERKEY) AS NumCustomers,
    ROUND(SUM(os.TRANSACTIONPRICE * os.QUANTITY),2) AS Sales
FROM
    FactOnlineSales os
INNER JOIN
    DimCustomer c ON os.CUSTOMERKEY = c.CUSTOMERKEY
INNER JOIN
    DimDate dt ON os.ORDERDATEKEY = dt.DATEKEY
INNER JOIN
    DimGeography g on c.GEOGRAPHYKEY = g.GEOGRAPHYKEY
GROUP BY
     g.COUNTRYNAME,
     dt.CALENDARYEAR
WITH ROLLUP;
```

Scenariusz drugi zawierał definicje zapytania które zwraca liczbę zamówień złożonych w każdym regionie sprzedaży w styczniu 2020 roku. Ten przykład badawczy sprawdzał wpływ zawężenia wyników zapytania poprzez polecenie WHERE, na średni czas wykonania zapytania. Treść zapytania została przedstawiona na Listingu 2.

Listing 2: Kod zapytania ze scenariusza drugiego

```
SELECT
    st.SALESTERRITORYNAME,
    COUNT(*) AS 'Number of orders'
FROM
    dbo.FactOnlineSales os
JOIN
    dbo.DimSalesterritory st
ON
    os.SALESTERRITORYKEY = st.SALESTERRITORYKEY
JOIN
    dbo.DimDate dt
ON
    os.ORDERDATEKEY = dt.DATEKEY
WHERE
    dt.MONTHNUMBEROFYEAR = 1
    AND dt.CALENDARYEAR = 2020
GROUP BY
    st.SALESTERRITORYNAME
```

Scenariusz trzeci zawierał definicje zapytania które wykorzystywało podzapytanie. Taki test pozwolił określić wpływ wykorzystania podzapytania skorelowanego na średni czas wykonania zapytania. Treść zapytania została przedstawiona na Listingu 2.

Listing 3: Kod zapytania ze scenariusza trzeciego

```
SELECT
    c.CustomerKey AS [Customer ID],
    CONCAT(c.LASTNAME, ' ', c.FIRSTNAME) AS [Customer Name],
    dt.FULLDATE AS [Order Date]
FROM
    FactOnlineSales os
    JOIN DimDate dt ON os.ORDERDATEKEY = dt.DATEKEY
    JOIN DimCustomer c ON os.CUSTOMERKEY = c.CUSTOMERKEY
WHERE
    dt.MONTHNUMBEROFYEAR = 12 AND dt.CALENDARYEAR = 2019 AND
    NOT EXISTS (
        SELECT
            1
        FROM
            FactOnlineSales os2
            JOIN DimDate dt2 ON os2.ORDERDATEKEY = dt2.DATEKEY
        WHERE
            os2.CUSTOMERKEY = os.CUSTOMERKEY AND
            dt2.MONTHNUMBEROFYEAR = 1 AND dt2.CALENDARYEAR = 2020
    )
ORDER BY
    c.LASTNAME;
```

Scenariusz czwarty zawierał definicje zapytania, które korzystało z polecenia Zapytanie korzysta z operatora IN, które pozwala na sprawdzanie czy wartość zawiera się w podanym zbiorze. Zapytanie grupuje rezultaty oraz je sortuje. Listing 4 przedstawia treść zapytania omawianego w scenariuszu czwartym.

Listing 4: Kod zapytania ze scenariusza czwartego

```
SELECT
    dt.CALENDARYEAR AS [Year],
    p.PRODUCTSUBCATEGORYNAME AS [Product],
    COUNT(DISTINCT os.CUSTOMERKEY) AS [#Customers],
    ROUND(SUM(os.TRANSACTIONPRICE * os.QUANTITY), 2) AS [Total Value]
FROM
    FactOnlineSales os
    INNER JOIN DimProduct p ON os.PRODUCTKEY = p.PRODUCTKEY
    INNER JOIN DimDate dt ON os.ORDERDATEKEY = dt.DATEKEY
WHERE
    p.PRODUCTSUBCATEGORYNAME IN ('Mountain Frames', 'Road Frames', 'Touring Frames')
GROUP BY
    dt.CALENDARYEAR, p.PRODUCTSUBCATEGORYNAME
ORDER BY
    [Year], [Product]
```

Scenariusz piąty zawierał definicje zapytania, które korzystało z klauzuli GROUP BY oraz funkcji arytmetycznej SUM oraz ROUND. Zapytanie dodatkowo grupuje rezultaty oraz następnie je sortuje. Listing 5 przedstawia treść zapytania omawianego w scenariuszu piątym.

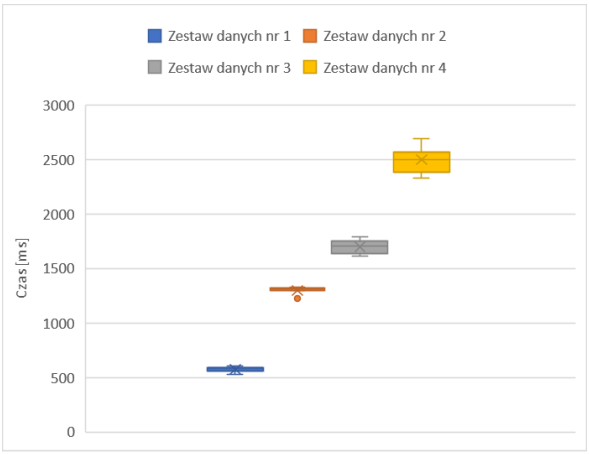
Listing 5: Kod zapytania ze scenariusza piątego

```
SELECT
    dt.CALENDARYEAR AS Year,
    ROUND(SUM(os.TRANSACTIONPRICE * os.QUANTITY), 2) AS 'OrderValue'
FROM
    FactOnlineSales os
    JOIN DimDate dt ON os.ORDERDATEKEY = dt.DATEKEY
GROUP BY
    dt.CALENDARYEAR
ORDER BY
    dt.CALENDARYEAR ASC
```

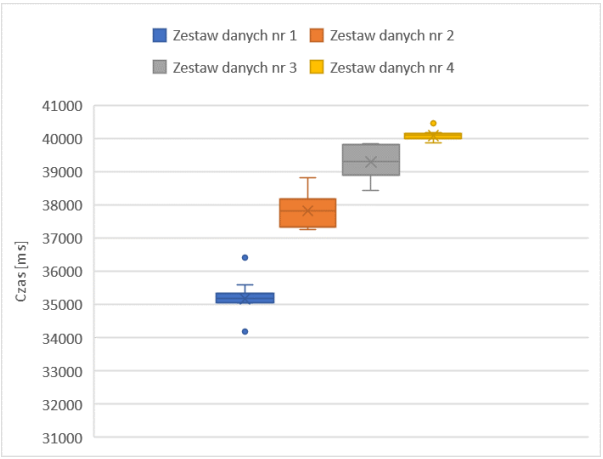
6. Wyniki

Każde zapytanie zostało wykonane sto razy na omawianym w rozdziale dotyczącym metodyki badań wolumenie danych. Czas wykonania zapytania został zmierzony przy użyciu SSMS (SQL Server Management Studio) w przypadku SQL Server, oraz przy użyciu Apache Hue w przypadku Apache Hadoop. Następnie z otrzymanych wyników została wyznaczona średnia oraz utworzone odpowiednie wykresy prezentujące otrzymane wyniki. W celu ujednolicenia wyników oraz ograniczenia wpływu zewnętrznego na wyniki, całe zbędne oprogramowanie zostało wyłączone na czas przeprowadzania badań. Dodatkowo został odrzucony pierwszy wynik, ponieważ czas wykonania tego zapytania był znacznie dłuższy od kolejnych, ponieważ silnik bazodanowy konstruował plan zapytania. Do przedstawienia wyników wykorzystano wykres pudełkowy, ponieważ wykres ten jest użytecznym narzędziem do analizy rozkładu danych, identyfikowania wartości odstających oraz porównywania rozkładów między różnymi grupami lub kategoriami. Dodatkowo wykorzystano wykresy kolumnowe do przedstawienia średnich czasów wykonania dla poszczególnych scenariuszy badawczych.

Wykres zaprezentowany na Rysunku 1 przedstawia rozkład wyników badania, na podstawie scenariusza pierwszego dla technologii SQL Server. Wykres przedstawiony na Rysunku 2 przedstawia rozkład wyników badania na podstawie scenariusza pierwszego dla technologii Apache Hadoop.

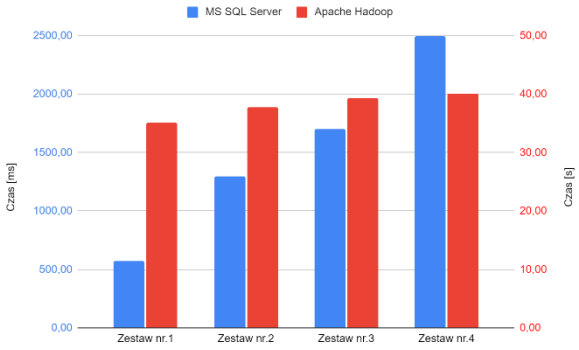


Rysunek 1: Wykres pudełkowy dla scenariusza pierwszego dla SQL Server.



Rysunek 2: Wykres pudełkowy dla scenariusza pierwszego dla Apache Hadoop.

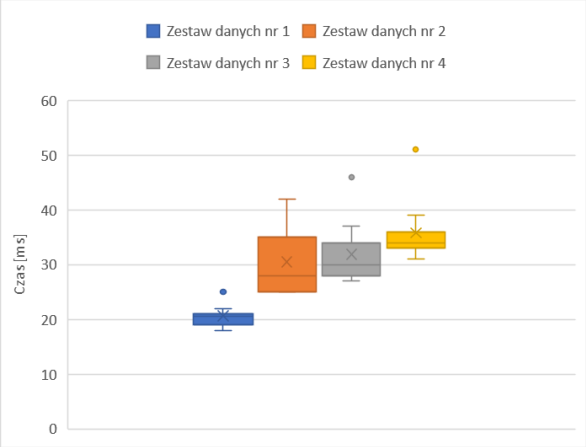
Rysunek 3 przedstawia wykres prezentujący średnie czasy wykonania zapytania, które zostało zdefiniowane i omówione w scenariuszu pierwszym na obu technologiach.



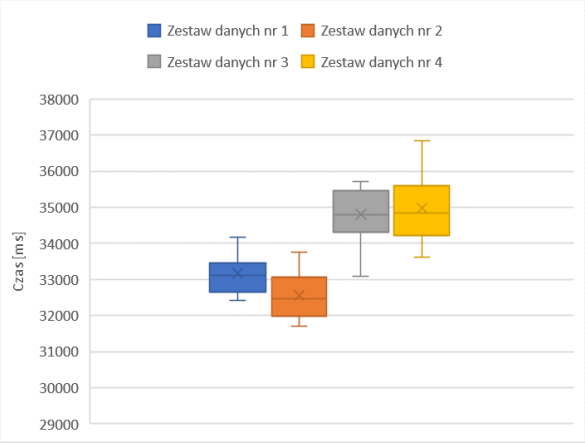
Rysunek 3: Średnie czasy wykonania scenariusza pierwszego.

Wykres przedstawiony na Rysunku 4 przedstawia rozkład wyników badania dla technologii Microsoft SQL Server. Wykres zaprezentowany na Rysunku 5 przedstawia rozkład wyników badania na podstawie

scenariusza drugiego dla technologii Apache Hadoop od firmy Oracle.

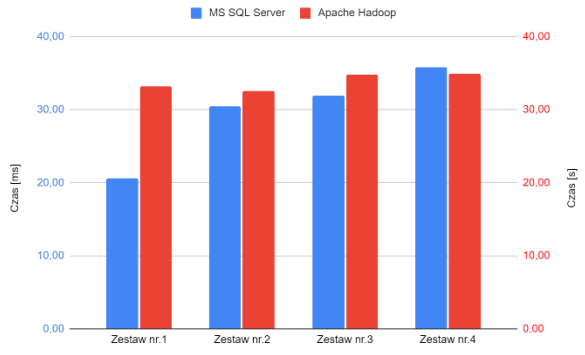


Rysunek 4: Wykres pudełkowy dla scenariusza drugiego dla SQL Server.



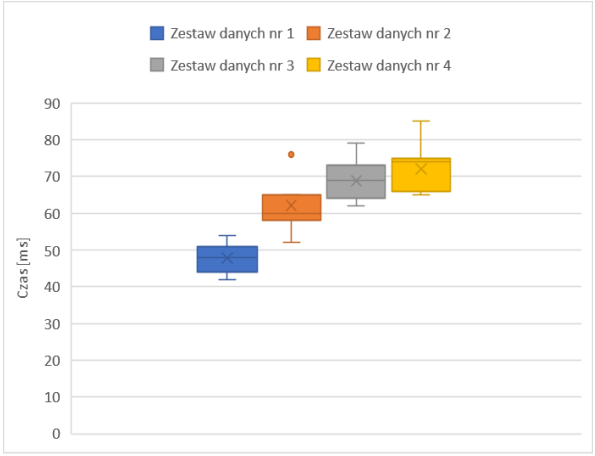
Rysunek 5: Wykres pudełkowy dla scenariusza drugiego dla Apache Hadoop.

Rysunek 6 przedstawia wykres prezentujący średnie czasy wykonania zapytania zdefiniowanego i omówionego w scenariuszu drugim, na obu technologiach.

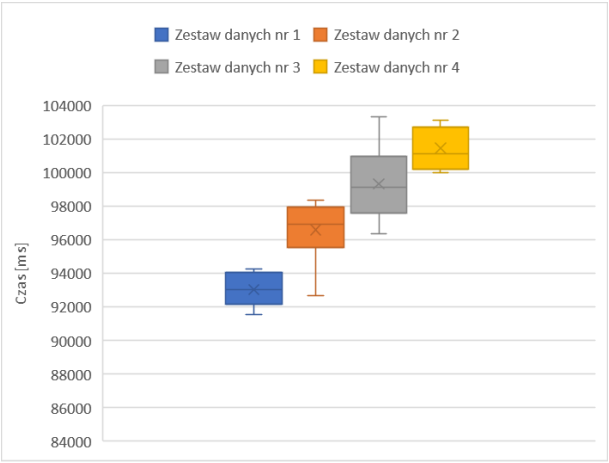


Rysunek 6: Średnie czasy wykonania scenariusza drugiego.

Wykres przedstawiony na Rysunku 7 przedstawia rozkład wyników badania na podstawie scenariusza trzeciego dla technologii Microsoft SQL Server. Wykres zaprezentowany na Rysunku 8 przedstawia rozkład wyników badania na podstawie scenariusza trzeciego dla technologii Apache Hadoop.

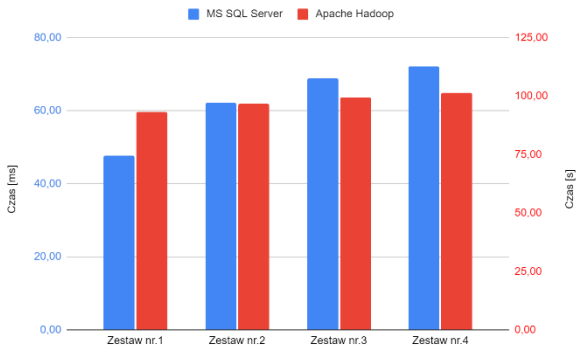


Rysunek 7: Wykres pudełkowy dla scenariusza trzeciego dla SQL Server.



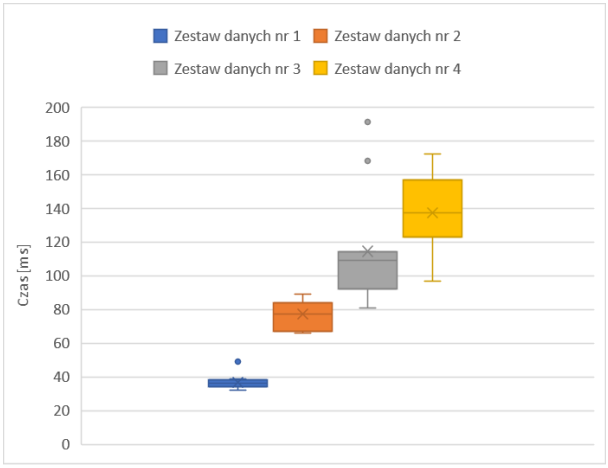
Rysunek 8: Wykres pudełkowy dla scenariusza trzeciego dla Apache Hadoop.

Rysunek 9 przedstawia wykres prezentujący średnie czasy wykonania zapytania ze scenariusza trzeciego.

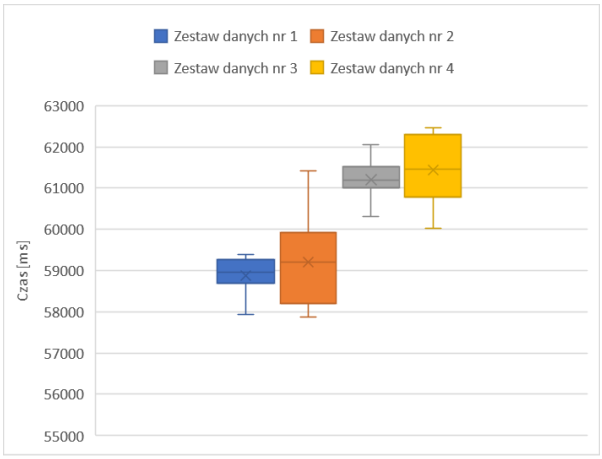


Rysunek 9: Średnie czasy wykonania scenariusza trzeciego.

Wykres zaprezentowany na Rysunku 10 przedstawia rozkład wyników badania na podstawie scenariusza czwartego dla technologii Microsoft SQL Server. Wykres przedstawiony na Rysunku 11 przedstawia rozkład wyników badania na podstawie scenariusza czwartego dla technologii Apache Hadoop.

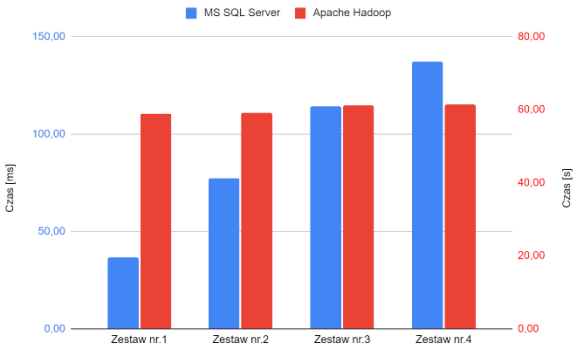


Rysunek 10: Wykres pudełkowy dla scenariusza czwartego dla SQL Server.



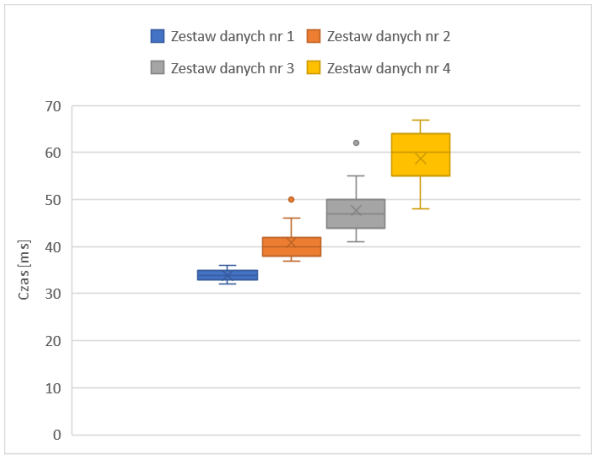
Rysunek 11: Wykres pudełkowy dla scenariusza czwartego dla Apache Hadoop.

Rysunek 12 przedstawia wykres prezentujący średnie czasy wykonania scenariusza czwartego na relacyjnej bazie danych SQL Server oraz technologii Apache Hadoop.

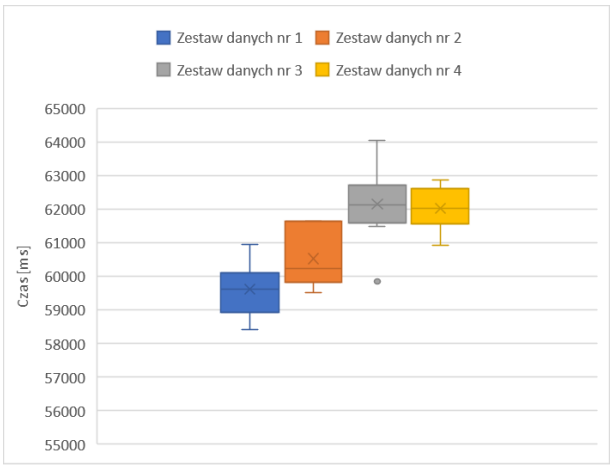


Rysunek 12: Średnie czasy wykonania scenariusza czwartego.

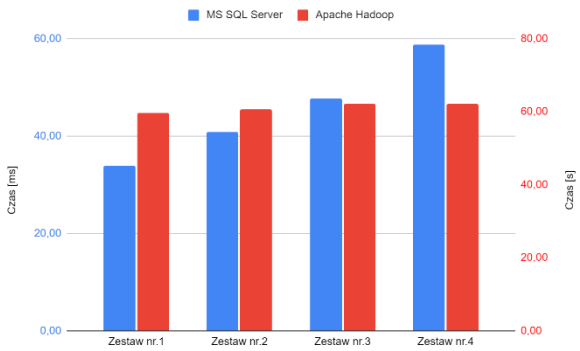
Wykres na Rysunku 13 przedstawia rozkład wyników badania na podstawie scenariusza czwartego dla technologii Microsoft SQL Server. Wykres na Rysunku 14 przedstawia rozkład wyników badania na podstawie scenariusza czwartego dla technologii Apache Hadoop Rysunek 15 przedstawia wykres prezentujący średnie czasy wykonania zapytania zdefiniowanego w scenariuszu piątym.



Rysunek 13: Wykres pudełkowy dla scenariusza piątego dla SQL Server.



Rysunek 14: Wykres pudełkowy dla scenariusza piątego dla Apache Hadoop.



Rysunek 15: Średnie czasy wykonania scenariusza piątego.

7. Wnioski

Na podstawie wyników uzyskanych ze scenariusza pierwszego można wywnioskować, że średni czas wykonania zapytania rośnie wraz z liczbą rekordów w zestawie danych dla obu rozwiązań. SQL Server był znacząco szybszy, ponieważ czasy wykonania zostały wyrażone w milisekundach, a nie w sekundach, jak w przypadku Apache Hadoop. W SQL Server średni

czas wykonania zapytania na najmniejszym wolumenie danych był ponad 4-krotnie mniejszy od czasu wykonania zapytania na największym wolumenie danych. Natomiast w przypadku Apache Hadoop wzrost czasu wykonania zapytania pomiędzy pierwszym a ostatnim zestawem danych wyniósł 14%, co jest znacznie lepszym rezultatem niż ten uzyskany przez rozwiązanie firmy Microsoft. Na podstawie wyników dla scenariusza pierwszego, można zauważyć, że dla zestawów danych nr 1 do nr 4, czas przetwarzania w Apache Hadoop jest znacznie dłuższy niż w MS SQL Server. Zatem, w kontekście tego scenariusza, MS SQL Server wykazuje wyższą wydajność i szybkość przetwarzania niż Apache Hadoop.

Wyniki uzyskane po wykonaniu scenariusza drugiego pokazują, iż w przypadku SQL Server, procentowa zależność między wzrostem liczby rekordów a średnim czasem wykonania zapytania prezentuje się następująco: dla zestawów danych 1 i 2 wynosi 48,06%, dla zestawów danych 2 i 3 wynosi 4,59%, natomiast dla zestawów danych 3 i 4 wynosi 12,22%. Te same zależności w przypadku Apache Hadoop prezentują się następująco: Zestaw 1 do zestaw 2: -1,8%, zestaw 2 do zestaw 3: 6,9% oraz zestaw 3 do zestaw 4: 0,5%. Wyniki pokazują, że procentowa różnica czasowa między zestawem 1 i 2 jest niewielka, natomiast różnica między zestawami 2 i 3 jest znaczna (6,9%). Różnica między zestawami 3 i 4 jest minimalna (0,5%). Takie wyniki pokazują, że Apache Hadoop jest mniej podatny na wzrost wolumenu danych niż SQL Server. Wyniki pokazują, że średni czas wykonania zapytania w MS SQL Server jest znacznie krótszy, dla wszystkich zestawów danych, niż w Apache Hadoop. Różnice pomiędzy średnimi czasami wykonania są bardzo duże.

Na podstawie wyników otrzymanych po wykonaniu scenariusza czwartego dla SQL Server, widać, że procentowy wzrost czasu wykonania zapytania między kolejnymi zestawami danych wynosi: zestaw 1 do zestaw 2: 29,9%, zestaw 2 do zestaw 3: 10,7% oraz zestaw 3 do zestaw 4: 4,8%. Natomiast w przypadku Apache Hadoop można zaobserwować, że różnice czasowe w średnim czasie wykonania wynoszą: zestaw 1 do zestaw 2: 3.85%, zestaw 2 do zestaw 3: 2,82% oraz zestaw 3 do zestaw 4: 2,14%. Wyniki pokazują, że średni czas wykonania zapytania ze scenariusza trzeciego w MS SQL Server jest trzy rzędy wielkości krótszy od średniego czasu wykonania w Apache Hadoop. Różnica jest zachowana dla każdego zbioru danych, co wskazuje, że SQL Server jest rozwiązaniem wydajniejszym pod względem czasu wykonania zapytania.

Na podstawie wyników otrzymanych po wykonaniu scenariusza czwartego, można zaobserwować, że średni czas wykonania zapytania w relacyjnej bazie danych SQL Server jest znacznie krótszy niż w Apache Hadoop. Ta różnica utrzymuje się dla każdego zestawu danych, co sugeruje, że SQL Server jest bardziej wydajnym rozwiązaniem pod względem czasu wykonania zapytania.

Na podstawie wyników z badania scenariusza piątego można zobaczyć, iż procentowa różnica między średnim czasem wykonania zapytania dla obu technologii wynosi kolejno: 175763,7%, 147888,5%, 130214,4% oraz 105555,9%. Różnice są bardzo duże, jednak zauważalna jest tendencja spadkowa.

Rozwiązanie SQL Server charakteryzowało się dużo mniejszym wpływem wielkości wolumenu danych na średni czas wykonania zapytania. Dodatkową cechą tej technologii były mniejsze wahania czasów wykonania między próbami. Jednak, jeśli chodzi o główny czynnik, czyli czas wykonania zapytania, to rozwiązanie SQL Server uzyskało znacznie korzystniejsze rezultaty.

Literatura

[1] P. O. Queiroz-Sousa, A. C. Salgado, A review on OLAP technologies applied to information networks, ACM Transactions on Knowledge Discovery from Data (TKDD) 14(1) (2019) 1–25, https://doi.org/10.1145/3370912.

[2] S. Sagiroglu, D. Sinanc, Big data: A review, Proceedings of the 2013 international conference on collaboration technologies and systems (CTS), IEEE, San Diego, CA, USA (2013) 42–47, https://doi.org/10.1109/CTS.2013.6567202.

[3] S. Chaudhuri, U. Dayal, An overview of data warehousing and OLAP technology, ACM Sigmod record 26(1) (1997) 65–74, https://doi.org/10.1145/248603.248616.

[4] J. Song, C. Guo, Z. Wang, Y. Zhang, G. Yu, J. M. Pierson, HaoLap: A Hadoop based OLAP system for big data, Journal of Systems and Software 102 (2015) 167–181, https://doi.org/10.1016/j.jss.2014.09.024.

[5] R. Stanek, Microsoft SQL Server. Brno: Computer Press, 2013. [12.02.2023]

[6] R. Kumar, B. B. Parashar, S. Gupta, Y. Sharma, N. Gupta, Apache hadoop, nosql and newsql solutions of big data, International Journal of Advance Foundation and Research in Science & Engineering (IJAFRSE) 1(6) (2014) 28–36, https://doi.org/10.13140/2.1.3454.9444.

[7] K. Shvachko, H. Kuang, S. Radia, R. Chansler, The hadoop distributed file system, Proceedings of the 2010 IEEE 26th symposium on mass storage systems and technologies (MSST), Ieee, Incline Village, NV, USA (2010) 1–10, https://doi.org/10.1109/MSST.2010.5496972

[8] J. Dittrich, J. A. Quiané-Ruiz, Efficient big data processing in Hadoop MapReduce, Proceedings of the VLDB Endowment 5(12) (2012) 2014-2015, https://doi.org/10.14778/2367502.2367562.

[9] V. K. Vavilapalli, A. C. Murthy, C. Douglas, S. Agarwal, M. Konar, R. Evans, E. Baldeschwieler, Apache hadoop yarn: Yet another resource negotiator., Proceedings of the 4th annual Symposium on Cloud Computing, Santa Clara California (2013) 1–16, https://doi.org/10.1145/2523616.2523633

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 28 (2023) 217–222
Received: 25 May 2023
Accepted: 20 June 2023

Performance comparison of Flutter platform GUI in web and native environments

Porównanie wydajności interfejsu graficznego platformy Flutter w środowisku webowym i natywnym

Juliusz Piskor*, Marcin Badurowicz

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

Abstract
This article compares the performance of the Flutter framework's graphical user interface (GUI) in a Windows platform native environment and in a web environment on the same device. The purpose of the article is to make a comparative analysis of screen building and rebuilding times in both environments. For the purpose of the paper, a test application has been created in which screens of different types of complexity are included. The stated theses: "The Flutter platform is less efficient in the web environment compared to the native environment in terms of view loading times." and "The Flutter platform is less efficient in a web environment compared to a native environment in terms of view rebuilding time" have been proven.

Keywords: performance comparison; multi-platform; GUI performance

Streszczenie
Artykuł dotyczy porównania wydajności interfejsu graficznego frameworka Flutter w środowisku natywnym na platformie Windows oraz na środowisku webowym, na tym samym urządzeniu. Celem artykułu jest dokonanie analizy porównawczej czasów budowy i przebudowy ekranów na obydwu środowiskach. Na potrzeby pracy została wykonana aplikacja testowa w której skład wchodzą ekrany o różnym typie złożoności. Postawione tezy: “Platforma Flutter jest mniej wydajna w środowisku webowym w porównaniu do środowiska natywnego pod względem czasu ładowania widoków.” oraz “Platforma Flutter jest mniej wydajna w środowisku webowym w porównaniu do środowiska natywnego pod względem czasu przebudowywania widoków” zostały udowodnione.

Słowa kluczowe: porównanie wydajności; wieloplatformowość; wydajność interfejsu graficznego

*Corresponding author
Email address: juliusz.piskor@pollub.edu.pl (J. Piskor)

1. Wstęp

Planowanie projektu informatycznego jest niezwykle ważnym etapem w procesie tworzenia oprogramowania [1]. W dzisiejszych czasach, kiedy rynek technologiczny jest bardzo konkurencyjny, kluczowe jest, aby projekt był nie tylko innowacyjny i funkcjonalny, ale także dostosowany do potrzeb użytkowników.

Właściwie zaplanowany projekt IT może zminimalizować koszty, przyspieszyć czas realizacji oraz przyczynić się do powodzenia biznesowego projektu. Dlatego też, uwzględnienie etapu planowania jest niezbędne dla zespołów programistycznych, aby projekt IT był zrealizowany w sposób efektywny i zgodny z oczekiwaniami klienta. W tym kontekście, wybór odpowiednich narzędzi, technologii oraz zasobów, w tym wybór platformy docelowej, stanowi kluczowe zagadnienie, które musi być uwzględnione już na etapie planowania projektu IT.

Kilkanaście lat temu głównymi pytaniami jakie zadawały sobie zespoły programistów podczas procesu planowania danego projektu były te, związane z wykorzystaniem technologii oraz, co za tym idzie, platform docelowych dla tworzonych projektów [2]. Jednym z problemów, który często pojawiał się w kontekście takiego wyboru, gdy okazał się on błędny w trakcie trwania projektu, było trzymanie się go przez twórców do samego końca, ze względu na nieopłacalność zmiany wyboru w trakcie pracy nad nim. Wynikało to z faktu, że zmiana narzędzi i technologii wiązała się z koniecznością ponoszenia dodatkowych kosztów związanych m.in. z przeprowadzeniem szkoleń dla zespołu programistycznego lub zakupem licencji na nowe oprogramowanie. Prowadzić mogło to do sytuacji w której menedżerowie projektów byli świadomi, że wcześniejszy wybór platformy docelowej był błędny, a mimo tego zmuszeni byli dokończyć dany projekt.

W odpowiedzi na przedstawione problemy coraz większą popularnością cieszą się technologie wieloplatformowe. Dzięki nim możliwe jest zmniejszenie wielkości zespołów programistycznych, redukcja czasu i zasobów potrzebnych do tworzenia projektu [3], a co kluczowe w kontekście poniższej pracy, możliwe jest jak największe przesunięcie w czasie decyzji o wyborze platformy dla oprogramowania, ponieważ decyzja ta może zapaść już po rozpoczęciu prac nad aplikacją. Jedną z takich technologii jest framework Flutter, który pozwala na kompilację aplikacji na platformy takie jak

Windows, Linux, MacOS, Android, iOS lub web przy pomocy jednej bazy kodowej.

Celem niniejszego artykułu jest zbadanie czasu ładowania i odpowiedzi interfejsu graficznego platformy Flutter w celu sprawdzenia, czy aplikacje desktopowe oraz webowe są w takim samym stopniu wydajne. Zostanie to sprawdzone na podstawie przygotowanej aplikacji na potrzeby badań. W celu potwierdzenia hipotez, aplikacja zostanie uruchomiona i przeanalizowana na systemie Windows oraz jako aplikacja webowa uruchamiana na tym samym urządzeniu. Na tej podstawie postawione zostały następujące hipotezy:

H1. Platforma Flutter jest mniej wydajna w środowisku webowym w porównaniu do środowiska natywnego pod względem czasu ładowania widoków,
H2. Platforma Flutter jest mniej wydajna w środowisku webowym w porównaniu do środowiska natywnego pod względem czasu przebudowywania widoków.

2. Przegląd literatury

W środowisku naukowym istnieje wiele artykułów dotyczących porównania wydajnościowego aplikacji na różnych platformach docelowych. Można znaleźć także prace, które bezpośrednio porównują ze sobą framework Flutter z konkurencyjnymi rozwiązaniami pod względem wydajności i responsywności. Analiza poniższych artykułów pomóc może głębiej przyjrzeć się wydajności frameworku oraz dowiedzieć się czego można od niego oczekiwać odnosząc się do jego wydajności.

W pracy [4] zbadano wydajność graficzną platformy Flutter oraz zestawiono wyniki z podobnymi badaniami opierających się na testowaniu platformy Cordova. Skupiono się tam głównie na urządzeniach mobilnych - nie brano pod uwagę platform natywnych oraz webowych. Wyniki badań przerosły oczekiwania twórców i udowodniły przewagę Fluttera nad starszymi mobilnymi technologiami wieloplatformowymi.

Podobne badania wykonano w pracy [5]. Skupiono się tam nad porównaniem Fluttera oraz jego bezpośredniej konkurencji - Reacta Native. Wykazano wówczas, że nieznacznie mniej klatek animacji gubionych jest przez framework od firmy Facebook. Autor podkreśla jednak, że o ile różnica jest zauważalna, to nie jest ona znacząca.

Autor pracy [6] przeanalizował responsywność aplikacji natywnych i porównał ją z aplikacjami webowymi. Przy wykorzystaniu autorskich aplikacji wykonał badania, które wskazały, że lepszym wyborem przy tworzeniu aplikacji, która będzie intensywnie wykorzystywać zasoby, będzie platforma natywna.

Jednym z wniosków jednak było to, że aplikacje webowe, które korzystają z zewnętrznych treści, np. poprzez wykorzystywanie API, a nie takie, które same przeprowadzają obliczenia mogą być bliskie w wydajności do swoich natywnych odpowiedników.

W pracy [7] wykonano podobne badania. Skupiono się jednak tam nie tylko na aspektach wydajności interfejsu - badano także wydajność sieciową oraz wykorzystywanie baterii urządzenia. Autorzy pracy wykazali, że około 70% badanych aplikacji jest w każdym aspekcie bardziej wydajna w formie natywnej w porównaniu do swojego odpowiednika webowego.

W pracy [8] z 2020 roku autorka dokonała porównania aplikacji stworzonych na platformach natywnych oraz przy użyciu frameworka Flutter. Analizie zostały poddane dane zużycia procesora podczas korzystania z aplikacji. Wniosek, który został wyciągnięty przez autorkę z tych badań jest taki, że Flutter jest idealnym rozwiązaniem jeśli chodzi o tworzenie małych i średnich aplikacji z myślą o wydajności, ale przy dużych ustępuje rozwiązaniom natywnym, choć jak sama podkreśla, zauważalny jest potencjał w prześcignięciu wydajnościowym nawet przy większych aplikacjach.

Z kolei w pracy [9] autorzy porównali ze sobą pod względem wydajnościowym platformy Flutter oraz Xamarin, który także jest jednym z konkurencyjnych rozwiązań. Początkowo postawiona hipoteza mówiąca o tym, że Xamarin jako bardziej dojrzała technologia będzie średnio działać lepiej, ale badania potwierdziły że Flutter w większości aspektów był tak samo dobry, a często lepszy od swojego starszego konkurenta. Jedynym polem na którym Xamarin wypadł lepiej był test zużycia pamięci RAM.

W pracy [10] autor porównał ze sobą autorskie aplikacje na platformę Windows. Jedna z nich była napisana w technologii Flutter, druga zaś, korzystając z bardziej klasycznego rozwiązania w produkcji aplikacji desktopowych - w technologii .NET. Badania wydajnościowe, które przeprowadził wskazały, że aplikacje korzystające z frameworka do Google okazały się znacznie lepiej radzące sobie jeśli chodzi o rozruch programu i zużycie pamięci ale w renderingu obrazów przy użyciu CPU ustępują innym rozwiązaniom.

Dokonany przegląd literatury jednoznacznie wskazał, że Flutter został dość dobrze zbadany wydajnościowo w kontraście do swojej konkurencji na rynku cross-platformowych technologii, jak i do natywnych rozwiązań na różnorakich platformach. Wyniki badań i stojące za nimi wnioski przeważnie podkreślały że Flutter przerasta oczekiwania w kontekście wydajności i widoczna jest jego przewaga w tym aspekcie nad innymi rozwiązaniami.

Wiele wskazuje na to, że w środowisku brakuje prac, które bezpośrednio zajmują się porównaniem wydajnościowym wspomnianego produktu z nim samym z kontekście platform przez niego obsługiwanych.

Flutter, jako platforma do tworzenia aplikacji zarówno internetowych, mobilnych, jak i na komputery stacjonarne, może zachowywać się inaczej na tych samych komputerach w kontekście środowiska wykonawczego.

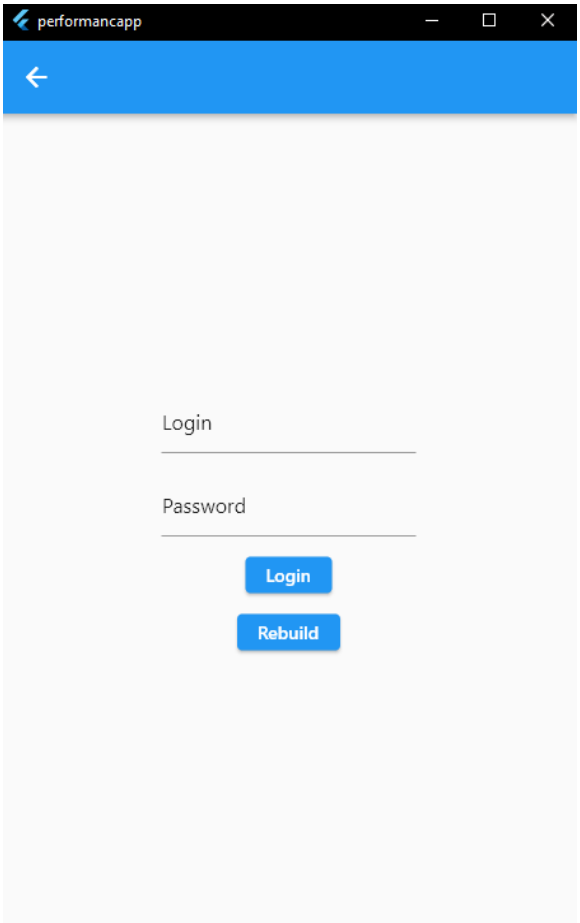
3. Implementacja aplikacji testowej

Na potrzeby badań wydajnościowych została wykonana aplikacja przy wykorzystaniu badanego frameworka.

Dzięki niej, możliwe będzie zbadanie czasu budowy i przebudowy dostępnej w niej ekranów. Projekt oparty został na Flutterze w wersji 3.7.0 umożliwiającej kompilację kodu do postaci zarówno webowej jak i desktopowej.

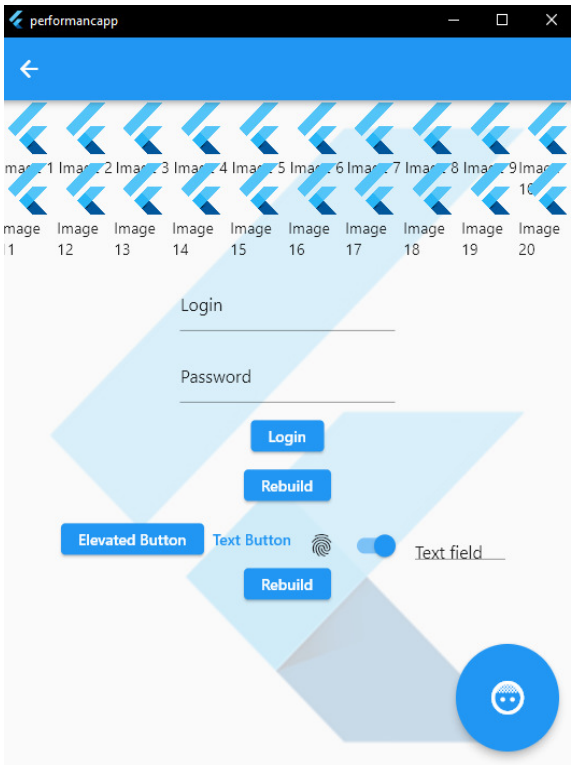
Aplikacja zawiera dwa ekrany, które będą poddane badaniom - ekran o mniejszym i większym poziomie skomplikowania. Dla upewnienia się że ekrany różnią się od siebie stopniem złożoności, a nie tylko są od siebie różne, zawierają one część wspólną – standardowy formularz logowania oraz przycisk, który pozwoli przebudować dany ekran na żądanie.

Ekran o mniejszej złożoności (Rysunek 1) został przygotowany zgodnie z wytycznymi optymalizacyjnymi dostępnymi w dokumentacji samego frameworku [11] i składa się tylko z formularza logowania oraz przycisku przebudowywania, wspomnianej wcześniej części wspólnej.



Rysunek 1: Interfejs ekranu o mniejszej złożoności.

Drugi z ekranów, który dostępny jest w aplikacji testowej, jest ten o wysokim skomplikowaniu (Rysunek 2). W jego skład, poza częścią wspólną, wchodzi szereg widgetów, których dodanie ma spowolnić czas budowy interfejsu graficznego. Wspomniane widgety zostały dobrane pod względem ich „ciężkości", co oznacza że powinny one mieć duży wpływ na szybkość budowy GUI. Przykładami takich widgetów mogą być Opacity lub ListView [12,13]. Znane są one ze swojej nieoptymalności i z reguły unikane są przy tworzeniu aplikacji komercyjnych.



Rysunek 2: Interfejs ekranu o większej złożoności.

4. Metodyka badawcza

Metodyka badawcza pozwalająca przeprowadzić badania dające odpowiedź na przedstawione wcześniej hipotezy składa się z ośmiu scenariuszy badawczych, które sprawdzą czas wstępnej budowy oraz późniejszej przebudowy graficznego interfejsu użytkownika przygotowanej aplikacji. Badania zostały przeprowadzone na dwóch ekranach i na dwóch platformach docelowych. Zależnie od scenariusza badawczego, zmierzony został czas wstępnej budowy ekranu po przejściu do niego z ekranu głównego, lub w przypadku drugiego scenariusza, czas przebudowy bo naciśnięciu odpowiedniego przycisku przebudowującego ekran.

Do określenia dokładnych czasów wykorzystane były narzędzia dostarczone, w przypadku aplikacji desktopowej, przez sam framework, a w przypadku aplikacji webowej, przez przeglądarkę internetową. Narzędzia o których mowa to odpowiednio: Dart DevTools oraz narzędzia deweloperskie do badania wydajności Google Chrome. Informacje w nich zawarte pokazały ile czasu zajmują poszczególne akcje wykonywane na aplikacji – w tym przebudowa i wstępna budowa ekranów.

Wszystkie próby badawcze były realizowane na następującym systemie:

- Procesor- Intel i7-4790k;
- Karta graficzna - Nvidia GeForce GTX 1070;
- RAM - DDR3 16 GB 1600 Mhz;

- Dysk - SSD Samsung EVO 1TB;
- System operacyjny - Windows 10 22H2.

Przeprowadzone testy były prowadzone w warunkach, której umożliwiły jak najbardziej dokładne pomiary, tj. bez aplikacji działających w tle i z odłączonym połączeniem internetowym. Dla większej dokładności wyników badania powtórzone po 20 razy dla każdego z przypadków. Odstęp czasowy pomiędzy każdym testem wynosił 5 minut.

5. Wyniki badań

Wyniki badań wydajnościowych zostały przedstawione w Tabelach 1,2,3 i 4. Pierwszym z badanych przypadków był czas pierwszej budowy ekranów.

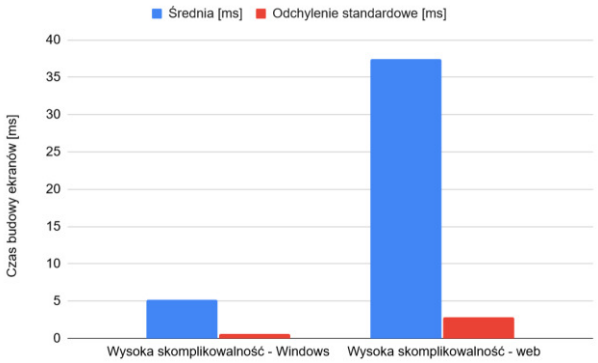
Tabela 1: Średni czas budowy ekranu o niskim skomplikowaniu na badanych platformach

Liczba prób: 20	Średni czas uruchomienia [ms] ± odchylenie standardowe
Windows	**1,87 ± 0,162**
Web	**21,47 ± 2,268**

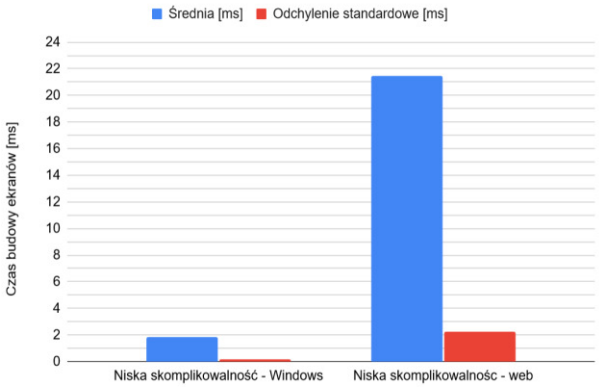
Tabela 2: Średni czas budowy ekranu o wysokim skomplikowaniu na badanych platformach

Liczba prób: 20	Średni czas uruchomienia [ms] ± odchylenie standardowe
Windows	**5,18 ± 0,598**
Web	**37,45 ± 2,873**

Zestawienie wyników badań wydajnościowych dla budowy ekranów dla poszczególnych widoków i platform zostało przedstawione na Rysunkach 3 i 4.



Rysunek 3: Testy wydajnościowe budowy ekranu o wysokim skomplikowaniu na badanych platformach.



Rysunek 4: Testy wydajnościowe budowy ekranu o niskim skomplikowaniu na badanych platformach.

Drugim z badanych przypadków był czas przebudowy każdego z ekranów.

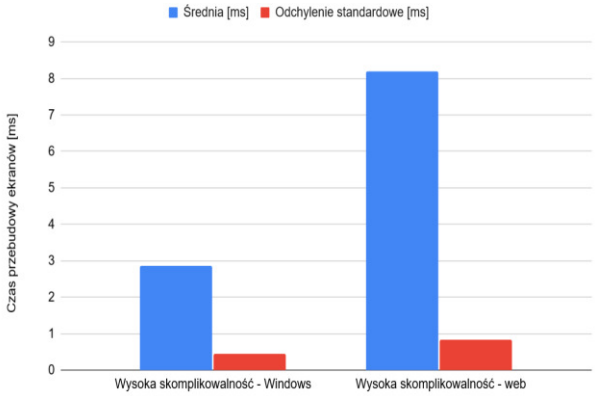
Tabela 3: Średni czas przebudowy ekranu o niskim skomplikowaniu na badanych platformach

Liczba prób: 20	Średni czas przebudowy [ms] ± odchylenie standardowe
Windows	**1,84 ± 0,303**
Web	**4,33 ± 0,889**

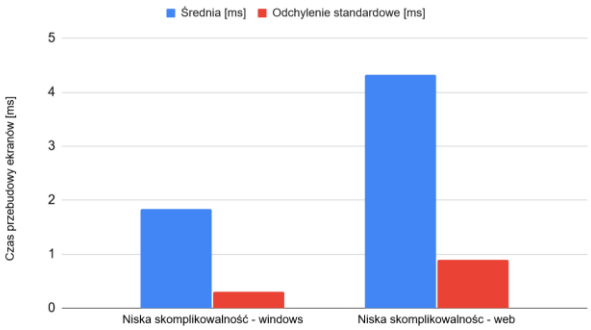
Tabela 4: Średni czas przebudowy ekranu o wysokim skomplikowaniu na badanych platformach

Liczba prób: 20	Średni czas przebudowy [ms] ± odchylenie standardowe
Windows	**2,86 ± 0,458**
Web	**8,2 ± 0,827**

Zestawienie wyników badań wydajnościowych dla przebudowy ekranów dla poszczególnych widoków i platform zostało przedstawione na Rysunkach 5 i 6.



Rysunek 5: Testy wydajnościowe przebudowy ekranu o wysokim skomplikowaniu na badanych platformach.



Rysunek 6: Testy wydajnościowe przebudowy ekranu o niskim skomplikowaniu na badanych platformach.

6. Wnioski

W przedstawionym artykule dokonano porównania wydajnościowego platformy Flutter w środowisku natywnym oraz webowym. Na potrzeby przeprowadzonych badań stworzono aplikację testową w której skład wchodziły ekrany o różnym typie skomplikowania. Wykonane badania polegały na zmierzeniu czasów wstępnej budowy oraz późniejszej przebudowy przygotowanych ekranów.

W przypadku badań obejmujących pomiar czasów wstępnej budowy ekranów, różnica w szybkości budowy zarówno ekranu o wyższym jak i niższym poziomie skomplikowania na badanych platformach była bardzo dobrze widoczna. Różnica ta była bardziej zauważalna w przypadku testów ekranu o niższym skomplikowaniu. Framework budował ten widok średnio 11,5 razy dłużej w środowisku webowym w porównaniu do natywnego. Nieznacznie mniejsza różnica widoczna była w przypadku ekranu o większym skomplikowaniu. Mowa tu o średnio nieco ponad 7-krotnie dłuższym czasie budowy na platformie webowej.

Podobną sytuację można zauważyć było w przypadku pomiarów czasów przebudowy ekranów, lecz zauważona różnica nie była już aż tak duża. Średni czas przebudowy ekranu o niższym skomplikowaniu był o ponad 2 razy większy w środowisku webowym w porównaniu do desktopowego. Jeśli chodzi o ekran o większym skomplikowaniu, tutaj różnica była nieznacznie większa – średnio niespełna 3 krotnie dłużej widok przebudowywał się na platformie webowej.

Przeprowadzone badania i uzyskane na ich podstawie wyniki pozwalają potwierdzić wcześniej postawione hipotezy H1 oraz H2. Warto zaznaczyć jednak, że różnice w czasie budowy i przebudowy ekranów były tym mniejsze, im bardziej skomplikowane były widoki. Niewykluczone jest, że w przypadku aplikacji komercyjnych, zwłaszcza takich, które zawierają skomplikowane struktury widgetów, różnica ta mogłaby być jeszcze mniej zauważalna, ale w celu potwierdzenia tego, potrzebne by były dodatkowe badania.

Ważnym do dodania jest także fakt, że o ile procentowo różnice pomiędzy czasami na różnych platformach są znaczące, to realne wartości dla przeciętnego użytkownika mogą być niezauważalne lub pomijane w odbiorze responsywności danej strony, czy też aplikacji.

Literatura

[1] A. M. Aladwani, IT project uncertainty, planning and success: An empirical investigation from Kuwait, Information Technology & People 15 (3) (2002) 210-226, https://doi.org/10.1108/09593840210444755.

[2] N. Shevtsiv, A. Striuk, Cross platform development vs native development, CEUR Workshop Proceedings 2832 (2021) 75-83, https://doi.org/10.31812/123456789%2F4428.

[3] L. Dagne, Flutter for cross-platform App and SDK development, Bachelor Thesis at Metropolia University of Applied Sciences, Helsinki, 2019, https://www.theseus.fi/bitstream/handle/10024/172866/Lukas%20Dagne%20Thesis.pdf.

[4] T. Tran, Flutter Native Performance and Expressive UI/UX, Bachelor Thesis at Metropolia University of Applied Sciences, Helsinki, 2022, https://www.theseus.fi/bitstream/handle/10024/336980/Thanh_Tran.pdf?sequence=2.

[5] J. Jagiełło, Performance comparison between react native and flutter, Bachelor Thesis at Umeå University, Umeå, 2019, https://www.diva-portal.org/smash/get/diva2:1349917/FULLTEXT01.pdf.

[6] A. Charland, B. Leroux, Mobile application development: web vs. native, Communications of the ACM, 54 (5) (2011) 49-53, https://doi.org/10.1145/1941487.1941504.

[7] Y. Ma, X. Liu, Y. Liu, Y. Liu, G. Huang, A tale of two fashions: An empirical study on the performance of native apps and web apps on android, IEEE Transactions on Mobile Computing 17 (5) (2018) 990-1003, https://doi.org/10.1109/TMC.2017.2756633.

[8] M. Olsson, A Comparison of Performance and Looks Between Flutter and Native Applications: When to prefer Flutter over native in mobile application development, Bachelor Thesis at Blekinge Institute of Technology, Karlskrona, 2020, https://www.diva-portal.org/smash/get/diva2:1442804/FULLTEXT01.pdf.

[9] Y. Rasmusson-Wright, S. Hedlund, Cross-platform Frameworks Comparison: Android Applications in a Cross-platform Environment, Xamarin Vs Flutter, Bachelor Thesis at Blekinge Institute of Technology, Karlskrona, 2021, https://www.diva-portal.org/smash/get/diva2:1568490/FULLTEXT01.pdf.

[10] S. Zindl, Flutter on Windows Desktop: a use case based study, Bachelor Thesis at Universität Stuttgart, Stuttgart, 2021, http://dx.doi.org/10.18419/opus-11723.

[11] Oficjalna dokumentacja Flutter - najlepsze praktyki, https://docs.flutter.dev/perf/best-practices, [19.05.2023].

[12] Gskinner, https://blog.gskinner.com/archives/2022/09/flutter-rendering-optimization-tips.html, [19.05.2023].

[13] Strona z wytycznymi Flutter cookbook, https://docs.flutter.dev/cookbook/lists/long-list, [19.05.2023].

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 28 (2023) 223–228
Received: 30 May 2023
Accepted: 20 June 2023

Usability analysis of banking service interfaces in Poland

Analiza użyteczności interfejsów serwisów bankowych w Polsce

Paulina Sułek* , Aleksandra Anna Walaszek

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

Abstract

This paper presents a comparative analysis of selected mobile banking applications: PKO BP, mBank, Pekao S.A., and Credit Agricole. The aim of the study is to obtain answers to the stated thesis and three specific research questions. In the initial stage, a review of scientific publications in the field of the topic was conducted. The research was carried out using an online survey and the cognitive walkthrough method with a modified list of usability testing. The obtained results revealed errors in the functioning of the analyzed service interfaces from the users' perspective and allowed for answering the research questions. The mobile applications of the discussed banks require minor modifications to maximize their usability.

Keywords: usability of interface; bank; service; mobile applications

Streszczenie

Niniejszy artykuł dotyczy analizy porównawczej wybranych aplikacji mobilnych banków: PKO BP, mBank, Pekao S.A. oraz Credit Agricole. Celem pracy jest uzyskanie odpowiedzi na zadaną tezę i trzy szczegółowe pytania badawcze. W początkowym etapie dokonano przeglądu publikacji naukowych z dziedziny tematyki pracy. Badania przeprowadzono z pomocą ankiety internetowej oraz metodą wędrówki poznawczej z wykorzystaniem zmodyfikowanej listy LUT. Otrzymane wyniki ujawniły błędy w funkcjonowaniu interfejsów analizowanych serwisów widziane z perspektywy użytkowników i pozwoliły udzielić odpowiedzi na zadane pytania. Aplikacje mobilne omawianych banków wymagają drobnych modyfikacji, tak aby ich użyteczność była na jak najwyższym poziomie.

Słowa kluczowe: użyteczność interfejsu; bank; serwis; aplikacje mobilne

*Corresponding author

Email address: paulina.sulek1@pollub.edu.pl (P. Sułek)

1. Wstęp

Dynamiczny rozwój cyfryzacji skutkuje nieodpartą potrzebą podążania za najnowszymi technologiami. Bankowość mobilna (ang. *Mobile Banking*) to jedna z najnowszych innowacji dla sektorów finansowych, która może mieć praktyczną wartość zarówno dla użytkowników, jak i banków [1,2]. Urządzenia mobilne, takie jak smartfony i tablety, używane są obok komputerów osobistych, a nawet zastępują je w niektórych aplikacjach. Banki coraz częściej inwestują w mobilność, udostępniając kanały mobilnej sieci i aplikacji mobilnych dla bankowości internetowej oraz oferując nowe usługi płatności mobilnych [3]. Identyfikacja najlepiej zaprojektowanych użyteczności interfejsów serwisów bankowych z punktu widzenia klienta indywidualnego w Polsce, jest głównym celem banków [4,5]. Klient indywidualny bankowości elektronicznej, z użytkownika najprostszych jej funkcji, przeradza się w użytkownika świadomego, widzącego plusy i minusy tej nowoczesnej formy komunikacji, potrafiącego też ocenić własne korzyści z posiadania zdalnego dostępu do konta oraz wybrać najlepszy bank dla siebie [6]. Interfejs użytkownika bankowości internetowej zapewnia klientom platformę do łączenia serwerów bankowych i przeprowadzania transakcji przez Internet [7]. Daje to ogromne możliwości zarówno klientom, jak i bankom. Bezpieczeństwo i prywatność to jedne z największych problemów, z którymi borykają się banki, aby przetrwać na globalnym, konkurencyjnym rynku bankowym. Od ergonomiczności środowiska oraz narzędzi komputerowych (zarówno softwarowych jak i sprzętowych) zależy zadowolenie użytkowników systemów [8].

Bez wątpienia bardzo ważnym elementem, na który warto zwrócić uwagę jest jakość interfejsów bankowych aplikacji mobilnych [9,10]. Celem artykułu [11] było przeprowadzenie analizy porównawczej jakości interfejsów mobilnych aplikacji bankowych na przykładzie aplikacji T-Mobile Usług Bankowych i mBanku. Badania miały ujawnić, które cechy interfejsu użytkownika wpływają w największym stopniu na efektywność wykonywanych przez użytkownika czynności. W trakcie badań brano pod uwagę czynniki takie jak bezpieczeństwo aplikacji [12], dostępność na platformy mobilne, szybkość wykonywania zadań, prostota obsługi, zrozumiałość, oprawę graficzną, komfort pracy oraz liczbę funkcjonalności. Użyte metody badawcze to badania ankietowe, testy korytarzowe, badania eyetrackingowe i badania eksperckie [13]. Wyniki zrealizowanych badań pokazują przewagę aplikacji T-Mobile Usługi bankowe nad aplikacją mBanku w kilku kategoriach prostoty obsługi oraz zrozumiałości. Natomiast w innych kategoriach, takich jak responsywność, liczba funkcjonalności czy dostępność wygrała aplikacja mBank. Ostateczny wynik aplikacji mBank, która uzyskała 85/100 punktów był lepszy od aplikacji T-Mobile Usługi Bankowe, która zdobyła 83/100 punktów. Wyniki dla

obu aplikacji były jednak bardzo zbliżone. Różnica wynosiła jedynie dwa punkty.

2. Przegląd publikacji naukowych

W artykule [14] skupiono się na rozwiązaniu problemów bezpieczeństwa w taki sposób, aby podejście do bankowości cyfrowej, biorąc pod uwagę ogólny obraz, w tym wszystkie podmioty oraz procesy zaangażowane w operacje, było bardziej kompleksowe. W artykule przyjęto interdyscyplinarne podejście i połączono kilka elementów (badania, działania informacyjne i edukację), które są zgodne z ogólnie przyjętymi zasadami cyberbezpieczeństwa. W maju 2020 r. przeprowadzono ocenę ośmiu największych polskich banków za pomocą ankiety. Proces oceny składał się z trzech faz: analizy danych zastanych (ang. *desk research*), działań praktycznych (tj. interakcji z serwisem internetowym i aplikacjami mobilnymi) oraz kontaktu z pracownikami banku do uzupełnienia brakujących danych. Pod względem zasad bezpieczeństwa, zwłaszcza w obszarze edukacji, zdecydowanie wyróżniał się mBank. Tempo wdrażania nowych technologii było tam jednak znacznie szybsze niż rozwój metod zabezpieczania danych i nie zawsze znajdowało odzwierciedlenie w otrzymywanych informacjach.

Artykuł [15] skupiał się na przedstawieniu struktury kryteriów oceny interfejsu wybranych mobilnych aplikacji bankowych w Polsce. Do badań zaproszono osoby korzystające z internetu oraz posiadające aktywne konto bankowe z dostępem do bankowości mobilnej na telefonie z system operacyjnym: Android, iOS lub Windows Phone. W badaniu wzięło udział 1525 respondentów. Badanie objęło 27 mobilnych aplikacji oferowanych przez najpopularniejsze banki. Dodatkowo zbadano cel poboczny, jakim było przedstawienie metody konwersji oraz prezentacja zasadniczych wyników z przeprowadzonych badań. W pracy ukazano jakie kryteria oceny oprogramowania użyto oraz uzasadniono ten wybór. Wyniki badań były jednoznaczne: w czołówce na pierwszym miejscu pojawiła się aplikacja mobilna od PKO BP, następnie znalazły się oprogramowania od Pekao S.A. i T-Mobile Usługi bankowe. W tym zestawieniu najgorzej wypadły aplikacje banku BPH oraz Alior Banku.

Rosnąca dynamika i wymagania obecnego rynku spowodowały, że osoby starsze często odmawiają korzystania z aplikacji bankowości mobilnej [16]. Mogą się czuć wykluczone, z powodu nie zaznajomienia z interfejsem aplikacji i przebiegami jej procesów. Należy pamiętać, że takie osoby stanowią ponad ¼ ludności w Polsce. W artykule [17] starano się ustalić, w jaki sposób jakość usług bankowości mobilnej wpływa na satysfakcję klientów, zwłaszcza w pokoleniu X i Y. Zastosowano tam podejście ilościowe z opisowym typem badań. Wywnioskowano, że jakość usług bankowości mobilnej ma pozytywny wpływ na zadowolenie klientów.

Przedstawienie praktycznego zastosowania testów użyteczności w rzeczywistym przedsięwzięciu informatycznym było celem artykułu [18]. Obiektem badań użyteczności prezentowanych w niniejszym tekście była główna witryna Banku Zachodniego WBK S.A. Wykorzystano trzy techniki badania użyteczności: analiza ekspercka oparta o heurystyki Nielsena, testy użyteczności zrealizowane za pomocą scenariuszy użycia oraz śledzenie działania użytkowników na stronie głównej serwisu. Jako krok wstępny wykonana została również tzw. analiza kontekstu użytkowania obejmująca m.in. analizę profilu użytkowników, ich potrzeb i typowych działań w serwisie.

Ujawnienie wpływu interpersonalnego na przyjęcie nowego interfejsu usługi było celem artykułu [19]. Metodą badania była ankieta, którą przeprowadzono, aby dodatkowo zweryfikować rolę wpływu interpersonalnego na przyjęcie innowacji usługowych. Gdy ludzie otrzymują relatywnie złożoną usługę świadczoną przez nowy interfejs usług, takich jak bankowość internetowa i zakupy online, bardzo ważną rolę odgrywają więzi międzyludzkie. W ten sposób możliwe jest osiągnięcie nowej wartości interfejsu usługi, jak obniżenie kosztów czy zwiększenie satysfakcji klienta.

Głównym celem artykułu [20] było wyłonienie najlepszych serwisów bankowości elektronicznej z punktu widzenia indywidualnego klienta w okresie przed kryzysem i po kryzysie w latach 2008, 2010/2011, 2013. Autorzy zastosowali własny, choć oparty na literaturze, zestaw kryteriów oceny punktowej oraz wybór usług elektronicznych wybranych banków. Przeprowadzono badanie jakości serwisów internetowych oferujących elektroniczny dostęp do usług najpopularniejszych wśród polskich klientów indywidualnych. Respondenci wypełnili tabele oceniające serwisy bankowości elektronicznej banków, w których mieli rachunki, przeprowadzając analizę i ocenę uzyskanych wyników. Autorzy zaimplementowali także metodę konwersji oraz zebrali informacje o wybranych kryteriach (ekonomiczne, funkcjonalne, technologiczne), a do oceny zastosowano typową skalę Likerta. W 2008 roku najlepszymi serwisami w rankingu był mBank (83,49%), a w 2011 roku: BZ WBK (90,28%) oraz Inteligo (87,89%). Potwierdziła się również teza autorów o nieadekwatności i swoistej powierzchowności standardowych, ujednoliconych, ilościowych metodologii stosowanych do oceny oraz selekcji serwisów bankowości elektronicznej.

Przegląd literatury wykazał, iż najlepszymi metodami do analizy użyteczności interfejsów są: zasady konstrukcji dobrego i ergonomicznego interfejsu zawarte w dziesięciu podstawowych heurystykach Jakoba Nielsena [21] oraz badanie metodą wędrówki poznawczej z wykorzystaniem listy LUT [8]. Dla zwiększenia poprawności badania, pytania z listy LUT należy odpowiednio zmodyfikować w odniesieniu do tematu pracy, natomiast te nie znajdujące zastosowania pominąć.

3. Cel i zakres pracy

Celem pracy jest przeprowadzenie analizy użyteczności serwisów bankowych w Polsce. Badanie ma na celu wyłonić najlepszy serwis, który pod każdym badanym aspektem wypada najlepiej. Przetestowano wybrane serwisy bankowe pod względem wydajności, bezpie-

czeństwa oraz funkcjonalności systemu. Eksperyment badawczy został wykonany z wykorzystaniem ankiety internetowej przeprowadzonej z podziałem na cztery wybrane do analizy banki, natomiast pozostałe banki zostały potraktowane zbiorczo jako kategoria *inne*. Dodatkowo przeprowadzono badanie metodą wędrówki poznawczej, aby uzyskać szerszy pogląd na użyteczność interfejsów serwisów. Wykorzystano dwie aplikacje z analizowanych banków, które posiadają wersję demonstracyjną, dostępną w formie mobilnej dla każdego użytkownika.

Teza: PKO BP posiada najbardziej użyteczny interfejs na tle pozostałych serwisów bankowych w Polsce.

Szczegółowe pytania badawcze:

1. Czy serwis bankowy PKO BP posiada najbardziej przejrzysty interfejs?
2. Czy korzystanie z serwisu bankowego PKO BP jest najbezpieczniejszą opcją z dostępnych w Polsce serwisów bankowych?
3. Czy serwis bankowy PKO BP jest najpopularniejszym serwisem wśród ankietowanych?

4. Plan Badań

Plan badawczy prezentuje się następująco:

1. Wybór metod analizy.
   a) Badanie ankietowe - ankieta internetowa.
   b) Badanie praktyczne metodą wędrówki poznawczej z wykorzystaniem zmodyfikowanej listy LUT.
2. Przygotowanie pytań do ankiety dotyczących użyteczności serwisów bankowych.
   a) Zebranie podstawowych danych o ankietowanym.
   b) Przygotowanie pytań o używany przez respondenta system bankowy.
3. Wybranie serwisów bankowych do analizy porównawczej.
   a) Wybór serwisów.
   b) Wybór narzędzi do badania praktycznego.
   c) Przygotowanie eksperymentu polegającego na wykonaniu akcji na wersjach demo wybranych serwisów bankowych.
4. Wybór osób do badania.
   a) Ustalenie liczebności grupy badawczej metodą ankiety (min 200 osób).
   b) Ustalenie liczebności grupy badawczej metodą wędrówki poznawczej z wykorzystaniem zmodyfikowanej listy LUT (min 5 osób).
5. Przebieg badania.
   a) Udostępnienie ankiety.
   b) Wykonanie badania metodą wędrówki poznawczej z wykorzystaniem zmodyfikowanej listy LUT.
6. Analiza i interpretacja wyników.
   a) Opis wyników.
   b) Opracowanie wykresów.
7. Wnioski.

4.1. Badanie ankietowe

Celem badania ankietowego było pozyskanie danych o ocenie interfejsów serwisów bankowych w Polsce przez ich aktywnych użytkowników. Ankieta przeprowadzona została drogą elektroniczną z wykorzystaniem narzędzia do tworzenia ankiet *Google forms*.

W pierwszej części ankiety zebrano od respondentów informacje o wieku, płci, wykształceniu oraz miejscu zamieszkania. Następnie zadano pytanie, który system bankowy jest uznawany przez nich jako główny. Dalsza część ankiety polegała na ocenie w skali od 1 (źle) do 5 (bardzo dobrze) zadanych pytań o użyteczności wskazanego systemu bankowego. Określenie najlepszego oraz najgorszego serwisu wyliczono poprzez podzielenie sum ocen pozytywnych (4 i 5) oraz negatywnych (1 i 2). Odpowiedzi o wartości 3 uznano za neutralne i nie uwzględniano ich w tym działaniu.

4.2. Badanie metodą wędrówki poznawczej z użyciem zmodyfikowanej listy LUT

Celem badania praktycznego było sprawdzenie czasu wykonania zadania, zliczenie liczby dotknięć ekranu (tapnięć) telefonu oraz liczby popełnionych błędów. Porównano dwie aplikacje mobilne banków przy użyciu ich wersji demonstracyjnych dostępnych dla każdego. Z uwagi na dostępność wersji demo, wybrano aplikację IKO banku PKO BP SA oraz PeoPay banku Pekao SA. Badanie przeprowadzono z użyciem dwóch smartfonów różniących się system operacyjnym (Android oraz iOS), z czego jedna badana osoba posługiwała się tylko jednym z nich. Dla uzyskania miarodajnych wyników, kolejność testowanych aplikacji oraz urządzeń zmieniano naprzemiennie.

Badanie metodą wędrówki poznawczej składało się z siedmiu zadań. Zadania zostały przygotowane w taki sposób, aby można było je wykonać w obu aplikacjach demonstracyjnych banków. Treści pytań sformułowano tak, by w obu serwisach podawać te same dane (np. PIN, imię i nazwisko, numer konta). Dzięki temu wpisywane informacje wymagały tej samej liczby tapnięć w ekran urządzenia.

Badane osoby po wykonaniu zadań metodą wędrówki poznawczej dla każdej aplikacji, miały za zadanie odpowiedzieć na pytania ze zmodyfikowanej listy LUT [8]. Uczestnicy wypełniali dokument badania od razu po zakończeniu pracy z danym serwisem.

5. Wyniki badań

Przedstawione wyniki badań dotyczą jedynie tezy oraz pytań szczegółowych podanych w rozdziale 3.

5.1. Badanie ankietowe

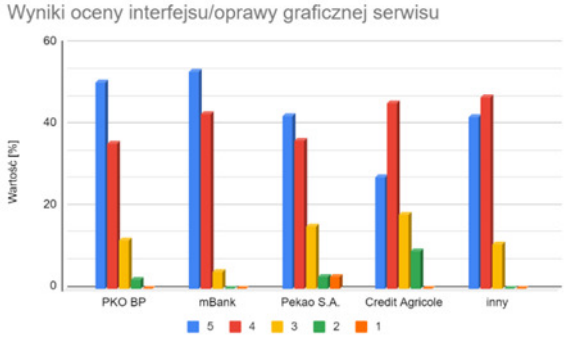
W badaniu ankietowym wzięło udział dwustu pięćdziesięciu respondentów. Większość respondentów stanowiły kobiety (172/250), natomiast mężczyźni to tylko 75 osób. Najczęściej wybieraną odpowiedzią dotyczącą wieku było: 20-35, ponieważ wybrało ją aż 155/250 respondentów. Następnie pod względem liczby odpowiedzi znalazła się opcja mniej niż 20, stanowiąca 1/4 wyniku. Trzy osoby wybrały zaś odpowiedź, która nie

identyfikuje ich z żadną płcią. Link do formularza udostępniono w gronie znajomych, rodziny i studentów. Skorzystano również z grup poświęconym rozwiązywaniu ankiet, do których należą ludzie z różnych środowisk. Zbieranie odpowiedzi trwało dwa tygodnie, od 8 do 22 marca 2023 roku.

Bardzo ważnym elementem w badaniu ankietowym było pytanie o to, którego banku aplikacji używa ankietowany. W odpowiedziach wybrane zostały cztery największe instytucje finansowe w Polsce: PKO BP, mBank, Pekao S.A. oraz Credit Agricole. Dodatkowo dodana została piąta opcja (inny), umożliwiająca respondentowi wpisanie nazwy banku, w którym posiadał konto, jeśli nie był klientem żadnego z tych powyżej wymienionych. Z możliwości kliknięcia w to pole skorzystano 64 razy, a najczęściej wpisywano tam: Alior Bank, ING Bank Śląski, Millenium Bank oraz Santander Bank Polska SA. Liderem został PKO BP, z którego usług korzystało blisko stu ankietowanych, co ukazuje Rysunek 1. Następny sklasyfikował się mBank z liczbą użytkowników 49. Również znaczący wynik osiągnął Pekao S.A., na który zagłosowano 33 razy. Natomiast tylko 1/3 tej wartości uzyskał Credit Agricole. Jeżeli uczestnik badania posiadał kilka zainstalowanych aplikacji bankowych, proszony był o zaznaczenie odpowiedzi z nazwą tej używanej najczęściej.

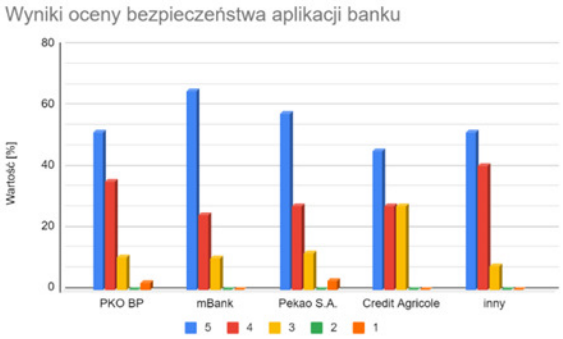
Rysunek 1: Pytanie jakiego banku aplikacje używa ankietowany.

Pierwsze szczegółowe pytanie w ankiecie dotyczyło oceny interfejsu/oprawy graficznej serwisu. Przedstawione na Rysunku 2 wyniki ukazują, że większość ankietowanych pozytywnie oceniło swój bank. Najlepszy wynik uzyskała aplikacja mBanku, z sumą procentową ocen 4 i 5 równą 96%. Jedynie Credit Agricole został oceniony słabiej ze znaczącą przewagą oceny 4.



Rysunek 2: Pytanie o interfejs/oprawę graficzną.

Według większości ankietowanych aplikacje mobilne banków, których są klientami, posiadają odpowiednie zabezpieczenia. Wśród użytkowników PKO BP i Pekao S.A. pojawiło się jednak kilka osób, które twierdzą inaczej. Banki zawarte w odpowiedzi inny okazały się najlepsze, ponieważ uzyskały największą sumę pozytywnych ocen oraz ani jednej negatywnej, co potwierdza Rysunek 3. Natomiast badanie to skupiało się na czterech największych instytucjach finansowych w Polsce i w tym zestawieniu wygranym został mBank, który osiągnął zbliżoną, lecz niższą o 2,4 procenta, wartość finalną.



Rysunek 3: Pytanie o bezpieczeństwo.

5.2. Badanie ankietowe

Badanie zrealizowano z udziałem dziesięciu uczestników, którzy pochodzili z różnych grup wiekowych (od 20 do 55 lat) oraz posiadali zróżnicowane doświadczenie odnośnie korzystania z aplikacji mobilnych banków. Realizacja doświadczenia przebiegała w środowisku domowym tak, aby nie wywierać presji i nie stresować dodatkowo osób go wykonujących.

Zadanie 1 polegające na zalogowaniu do systemu zostało wykonane bezproblemowo dla obu aplikacji, nikt nie popełnił błędu, a liczba tapnięć była taka sama u każdego uczestnika. Średni czas potrzebny na wykonanie tej czynności był niemalże identyczny, a różnica wyników to zaledwie o 0,32 sekundy więcej dla IKO. Następnie do wykonania było zrealizowanie przelewu, które sprawiło pewne problemy. Liczba tapnięć wahała się od 51 do 77 i była mniejsza dla aplikacji banku PKO BP. Natomiast odchylenie standardowe czasu wykonania zadania dla IKO wynosiło ± 30,27, a dla PeoPay 13,17. Kolejne polecenie dotyczyło sprawdzenia historii transakcji i podania daty oraz kwoty najstarszej operacji. W zadaniu tym lepsze wyniki uzyskał serwis PeoPay. Średnie wyniki i ich odchylenia uzyskały wysokie wartości z uwagi na nadmierne przewijanie zawartości ekranu aby upewnić się, że to już koniec listy. Zadanie 4, w którym uczestnik miał podać kwotę aktywnej lokaty 12 miesięcznej, było jednym z trudniejszych. Pomimo większej liczby kroków dla IKO, średnia liczba tapnięć była mniejsza niż dla PeoPay, ponieważ zadanie to można było wykonać posługując się skrótem przenoszącym do zakładki Lokaty dostępnym na ekranie głównym. Natomiast średni czas dla obu serwisów wypadł dość podobnie, lecz odchylenie różniło się o 100% w stosunku do siebie. Kolejne zadanie dotyczy-

ło sprawdzenia aktywnych kart, czynność ta została zrealizowana bez większych problemów. Lepsze wyniki uzyskała aplikacja IKO ze średnim czasem 5,20 sekundy. Wszystkie wyniki dla zadania szóstego w przypadku obu serwisów wypadły niemalże jednakowo. Uzyskanie kodu BLIK okazało się być łatwe, badani bez zastanowienia wiedzieli co mają kliknąć. Wyniki ostatniego zadania znacznie różniły się między sobą. Wynikało to z umiejscowienia przycisku prowadzącego do wylogowania (Tabela 1), który lepiej zaprojektowany został w aplikacji PeoPay należącej do banku Pekao S.A.

Tabela 1: Wyniki badania metodą wędrówki poznawczej

Wyniki badnia metodą wędrówki poznaczej		IKO	PeoPay
		Średni czas dla zadania, średnia liczba tapnięć, suma błędów	Średni czas dla zadania, średnia liczba tapnięć, suma błędów
Zadanie 1	Czas (sekundy):	5,86 ± 2,30	5,54 ± 1,83
	Liczba tapnięć:	5,0 ± 0,00	5,0 ± 0,00
	Liczba błędów:	0 ± 0,00	0 ± 0,00
Zadanie 2	Czas (sekundy):	63,81 ± 30,27	59,31 ± 13,17
	Liczba tapnięć:	60,2 ± 7,15	62,2 ± 6,63
	Liczba błędów:	7 ± 0,82	7 ± 0,67
Zadanie 3	Czas (sekundy):	20,52 ± 23,13	16,14 ± 12,72
	Liczba tapnięć:	8,0 ± 9,74	6,1 ± 5,69
	Liczba błędów:	7 ± 1,57	1 ± 0,32
Zadanie 4	Czas (sekundy):	18,85 ± 16,34	15,53 ± 8,67
	Liczba tapnięć:	5,7 ± 4,69	6,1 ± 3,84
	Liczba błędów:	6 ± 1,58	4 ± 0,84
Zadanie 5	Czas (sekundy):	5,20 ± 2,54	7,70 ± 3,29
	Liczba tapnięć:	1,9 ± 0,88	3,4 ± 0,70
	Liczba błędów:	0 ± 0,00	2 ± 0,42
Zadanie 6	Czas (sekundy):	3,98 ± 0,77	3,93 ± 0,61
	Liczba tapnięć:	1,0 ± 0,00	1,2 ± 0,63
	Liczba błędów:	0 ± 0,00	0 ± 0,00
Zadanie 7	Czas (sekundy):	10,00 ± 11,30	2,05 ± 1,54
	Liczba tapnięć:	4,7 ± 3,65	1,1 ± 0,32
	Liczba błędów:	4 ± 0,70	0 ± 0,00

Następnie wyodrębniono wyniki z badania przeprowadzonego za pomocą zmodyfikowanej listy LUT. Uzyskane oceny w głównej mierze okazały się pozytywne (to znaczy 5 i 4), zdarzały się czasem jednak niższe wartości typu 3 i 2. Pierwszy obszar *Nawigacja i struktura* lepiej o 0,1 punktu oceniono dla banku Pekao S.A., jego wartość wyniosła 4,70/5,00. *Komunikaty, feedback, pomoc dla użytkownika* uzyskały najlepszy wynik z wszystkich badanych stref. Tutaj również lepiej wypadła aplikacja PeoPay, tym razem różnica wynosiła tylko 0,05 punktu. Następny obszar o nazwie *Interfejs aplikacji* dotyczył layoutu i doboru barw. Okazał się on być lepszy, z wynikiem 4,88/5,00, w aplikacji IKO. W jednym pytaniu strefy *Treść podstron* odnotowano najniższą uzyskaną wartość równą dwa dla PeoPay. Mimo to średni wynik dla tego banku, w tym obszarze jest większy o aż 0,13 punktu. Ostatni podział o nazwie *Wprowadzanie danych* uzyskał zbliżone do siebie wyniki dla obu serwisów. Sumaryczny wynik dla wszystkich obszarów to 4,77/5,00 dla IKO oraz 4,82/5,00 dla PeoPay (Tabela 2).

Tabela 2: Wyniki badania za pomocą zmodyfikowanej listy LUT

Wyniki badania za pomocą zmodyfikowanej listy LUT		IKO		PeoPay	
		Średnia dla każdego pytania	Średnia dla każdego obszaru	Średnia dla każdego pytania	Średnia dla każdego obszaru
Nawigacja i struktura	Pytanie 1	4,30	4,60	4,50	4,70
	Pytanie 2	4,50		4,50	
	Pytanie 3	4,50		4,80	
	Pytanie 4	4,70		4,80	
	Pytanie 5	4,90		4,80	
	Pytanie 6	4,70		4,70	
	Pytanie 7	4,60		4,80	
Komunikaty, feedback, pomoc dla użytkownika	Pytanie 8	4,90	4,88	5,00	4,93
	Pytanie 9	4,90		5,00	
	Pytanie 10	4,90		4,90	
	Pytanie 11	4,80		4,80	
Interfejs aplikacji	Pytanie 12	4,80	4,85	4,70	4,75
	Pytanie 13	4,80		4,80	
	Pytanie 14	4,90		5,00	
	Pytanie 15	4,90		4,50	
Treść podstron	Pytanie 16	4,80	4,70	5,00	4,83
	Pytanie 17	4,70		4,70	
	Pytanie 18	4,60		4,80	
Wprowadzanie danych	Pytanie 19	4,80	4,82	4,70	4,88
	Pytanie 20	4,90		4,90	
	Pytanie 21	4,80		5,00	
	Pytanie 22	4,70		4,80	
	Pytanie 23	4,90		5,00	
	Wynik WUT	**dla IKO**	**4,77**	**dla PeoPay**	**4,82**

6. Wnioski

Niniejsza praca miała na celu przeprowadzenie analizy użyteczności serwisów bankowych w Polsce. Przeprowadzone badania pozwoliły zrealizować ustalony cel, obalić postawioną tezę oraz udzielić odpowiedzi na sformułowane pytania badawcze.

Pierwsze szczegółowe pytanie badawcze brzmiało: *Czy serwis bankowy PKO BP posiada najbardziej przejrzysty interfejs?*. Ostateczne wyniki okazały się być dość zaskakujące, ponieważ we wszystkich badaniach, IKO zajęło dopiero drugie miejsce. Patrząc jednak na oceny respondentów, aplikacja ta posiada dosyć przejrzysty interfejs. Dokładne wartości wyników są dość zbliżone, wystarczy więc jedynie dopracować niektóre elementy, aby system stał się najlepszym na polskim rynku. Podsumowując, pierwsza hipoteza badawcza została obalona.

Drugie szczegółowe pytanie badawcze brzmiało: *Czy korzystanie z serwisu bankowego PKO BP jest najbezpieczniejszą opcją z dostępnych w Polsce serwisów bankowych?*. Odpowiedź na nie znalazła się zapytaniu badania ankietowego, przedstawionym na Rysunku 3. Użytkownicy aplikacji IKO w znacznej większości czuli się bezpiecznie (91/93 ankietowanych zaznaczyło opcje 3, 4 lub 5, gdzie 5 to bardzo bezpiecznie). Natomiast porównując to z wynikami dla innych serwisów bankowych, nie był to najlepszy wynik. Bank PKO BP powinien zatem skupić się na podniesieniu poziomu bezpieczeństwa, aby osiągnąć jak najlepszą jakość usług. Podsumowując druga hipoteza badawcza również została obalona.

Trzecie szczegółowe pytanie badawcze brzmiało: *Czy serwis bankowy PKO BP jest najpopularniejszym serwisem wśród ankietowanych?*. Wyniki tego zapytania przedstawiają się podobnie do poprzedniego, a odpowiedź na nie znalazła się na Rysunku 1. Badanie

wykazało, iż aplikacja banku PKO BP jest najpopularniejsza w stosunku do trzech innych równie znanych i lubianych instytucji finansowych w Polsce. Jedynie dodatkowa odpowiedź inny, w której zawarte zostały wszystkie pozostałe banki prowadzące działalność w RP, oceniona została podobnie do IKO. Nadal jednak różnica pomiędzy nimi wyniosła ponad 10% ankietowanych. Trzecia hipoteza badawcza została zatem potwierdzona.

Otrzymane wyniki badań przeprowadzonych w pracy ujawniły błędy w funkcjonowaniu interfejsów analizowanych serwisów z perspektywy użytkowników. Mogą zatem stanowić wartościowe wsparcie dla organizacji udostępniających podobne aplikacje, umożliwiając programistom zwiększyć ich atrakcyjność.

Literatura

[1] Q. Zhou, F. J. Lim, H. Yu, G. Xu, X. Ren, D. Liu, X. Wang, X. Mai, H. Xu, A study on factors affecting service quality and loyalty intention in mobile banking, Journal of Retailing and Consumer Services 60 (2021) 1-8.

[2] V. V. Pshenichnikov, V. E. Krolivetskaya, A. V. Babkin, Bank running model's evolution on the wave of information and communication technology development, In 2018 IEEE International Conference "Quality Management, Transport and Information Security, Information Technologies" (IT&QM&IS), IEEE (2018, September) 709-713.

[3] G. Fenu, P. L. Pau, An analysis of features and tendencies in mobile banking apps, Procedia Computer Science 56 (2015) 26-33.

[4] W. Chmielarz, Comparative analysis of electronic banking services in selected banks in Poland in 2013, Prace Naukowe Uniwersytetu Ekonomicznego we Wrocławiu 316 (2013) 16-29.

[5] P. Iyappan, V. PrasannaVenkatesan, R. Amarnath, L. N. Mouhammed, A. Selvamani, An enhanced smart multi-banking integrated system—Service oriented approach, In 2012 Third International Conference on Computing, Communication and Networking Technologies (ICCCNT'12), IEEE (2012, July) 1-8.

[6] W. Chmielarz, M. Zborowski, Analiza jakości najpopularniejszych bankowych serwisów internetowych w Polsce w 2014 roku, Kwartalnik Naukowy Uczelni Vistula 1(47) (2016) 22-37.

[7] A. Ojeniyi, O. K. Alo, E. M. Oyetade, M. T. Ang, Y. K. Sanusi, Online banking user interface: Perception and attitude, In 2015 International Conference on Computer, Communications, and Control Technology (I4CT), IEEE (2015, April) 64-69.

[8] M. Miłosz, Ergonomia Systemów Informatycznych, Politechnika Lubelska, Lublin, 2014.

[9] A. Premchand, A. Choudhry, Open banking & APIs for transformation in banking, In 2018 international conference on communication, computing and internet of things (IC3IoT), IEEE (2018, February) 25-29.

[10] A. Ciunova-Shuleska, N. Palamidovska-Sterjadovska, J. Prodanova, What drives m-banking clients to continue using m-banking services?, Journal of Business Research 139 (2022) 731-739.

[11] M. Kurzyna, D. Matysiak, M. Miłosz, Analiza porównawcza jakości interfejsów mobilnego dostępu do usług wybranych banków, Journal of Computer Sciences Institute 5 (2017) 159-166.

[12] A. Kruzikova, L. Knapova, D. Smahel, L. Dedkova, V. Matyas, Usable and secure? User perception of four authentication methods for mobile banking, Computers & Security 115 (2022) 1-12.

[13] M. Kaczmarek, Użyteczność serwisów internetowych banków, Wydawnictwo Uniwersytetu Ekonomicznego w Poznaniu, Poznań, 2016.

[14] W. Wodo, P. Blaskiewicz, D. Stygar, N. Kuzma, Evaluating the security of electronic and mobile banking, Computer Fraud & Security, 2021(10) (2021) 8-14.

[15] M. R. Zborowski, K. Łuczak, Propozycja doboru składowych struktury kryteriów oceny jakości aplikacji mobilnych na przykładzie wybranych bankowych aplikacji mobilnych w Polsce, Annales Universitatis Mariae Curie-Skłodowska, sectio H–Oeconomia, 50(2) (2016) 183-202.

[16] E. Ubam, L. Hipiny, H. Ujir, User Interface/User Experience (UI/UX) Analysis & Design of Mobile Banking App for Senior Citizens: A Case Study in Sarawak, Malaysia, In 2021 International Conference on Electrical Engineering and Informatics (ICEEI), IEEE (2021, October) 1-6.

[17] R. Trialih, E. S. Astuti, D. F. Azizah, Y. T. Mursityo, M. D. Saputro, Y. A. Aprilian, A. S. Rizki, How mobile banking service quality influence customer satisfaction of generation x and y?, In 2018 International Conference on Information and Communication Technology Convergence (ICTC), IEEE (2018, October) 827-832.

[18] M. Szulc, A. Jarzębowicz, Optymalizacja witryny internetowej Banku Zachodniego WBK SA na podstawie testów użyteczności, XII Krajowa Konferencja Inżynierii Oprogramowania, Pomorskie Wydawnictwo Naukowo-Techniczne (2010, September) 329-336.

[19] H. Sang, Interpersonal Influence on Adoption of New Service Interface, In 2012 International Joint Conference on Service Sciences, IEEE (2012, May) 275-280.

[20] W. Chmielarz, M. Zborowski, Conversion method in comparative analysis of e-banking services in Poland, In International Conference on Business Informatics Research, Springer Berlin Heidelberg (2013, September) 227-240.

[21] Strona internetowa poświęcona heurystykom Nielsena, https://www.damianrams.pl/heurystyki-nielsena/, [29.05.2023].

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 28 (2023) 229–235
Received: 31 May 2023
Accepted: 20 June 2023

Comparative analysis of selected tools for test automation of web applications

Analiza porównawcza wybranych narzędzi do automatyzacji testów aplikacji webowych

Franciszek Wąsik*, Michał Pojęta*, Małgorzata Plechawska-Wójcik

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

Abstract
The paper concerns a comparison of selected tools for conducting automated unit tests of web applications. It focuses on the testing of server and client parts. The aim of the paper is to answer the questions: which tools work best in creating automatic tests of server and client applications. In the context of the server part, unittest and pytest – libraries based on the Python language – are compared. The comparison of the client part tools is carried out in the context of testing applications programmed with the Angular framework and pairs Jasmine and Jest together. The research is based on the results of test execution times of the prepared test applications. Analogous tests were programmed with each tool and repeated several times to obtain reliable results. The research showed that among the tools for testing server applications, unittest is the more efficient, while in the case of tools for testing client applications, Jasmine shows higher performance.

Keywords: comparative analysis; automated testing; performance of test automation tools

Streszczenie
Artykuł dotyczy porównania wybranych narzędzi do przeprowadzania automatycznych testów jednostkowych aplikacji internetowych. Skupia się on na testach części serwerowej i klienckiej. Celem pracy jest uzyskanie odpowiedzi na pytania: które narzędzia sprawdzają się najlepiej w tworzeniu automatycznych testów aplikacji serwerowych oraz klienckich. W kontekście części serwerowej porównaniu zostały poddane unittest oraz pytest – narzędzia oparte o język Python. Porównanie narzędzi części klienckiej jest przeprowadzone w kontekście testowania aplikacji zaprogramowanych z wykorzystaniem szkieletu programistycznego Angular i zestawia ze sobą Jasmine oraz Jest. Badania oparto o wyniki czasów wykonania testów przygotowanych aplikacji testowych. Analogiczne testy zaprogramowano z użyciem każdego z narzędzi i wielokrotnie powtórzono w celu uzyskania wiarygodnych wyników. Badania wykazały, że spośród narzędzi do testowania aplikacji serwerowych wydajniejszym jest unittest, natomiast w przypadku narzędzi do testowania aplikacji klienckich, Jasmine legitymuje się wyższą wydajnością.

Słowa kluczowe: analiza porównawcza; testy automatyczne; wydajność narzędzi do automatyzacji testów

*Corresponding author

Email address: **franciszek.wasik@pollub.edu.pl** (F. Wąsik), **michal.pojeta@pollub.edu.pl** (M. Pojęta)

1. Wstęp

Testowanie oprogramowania jest bardzo ważnym etapem jego produkcji. To czy system zostanie odpowiednio przetestowany, w dużym stopniu wpływa na jego dalszy cykl życia. Przeprowadzenie testów ma na celu wyłonienie niezgodności w tworzonym oprogramowaniu i pozwala uniknąć wielu problemów związanych z jego wdrożeniem i utrzymaniem [1]. Aby wytworzyć wysokiej jakości oprogramowanie, proces testowania powinien być dla jego twórców kluczowym aspektem [2].

Niniejsza praca traktuje o narzędziach służących do automatyzacji testów aplikacji internetowych. Możliwość automatyzacji przynosi wiele korzyści dla procesu testowania oprogramowania. Nie tylko pozwala na jego przyspieszenie, ale także na zwiększenie liczby wykonywanych testów [3]. Inną korzyścią wynikającą ze stosowania automatycznych testów, jest zmniejszenie zasobów ludzkich koniecznych do wykonania testów [4]. Co więcej, automatyzacja wpływa na zmniejszenie wydatków wynikających z konieczności przeprowadzania testów oprogramowania, które mogą stanowić znaczną część kosztów jego wytworzenia [5-6].

Jednym z kluczowych aspektów, które wpływają na wdrożenie automatyzacji jest zmniejszenie czasu trwania procesu testowania [7]. Powstało wiele publikacji, które analizują korzyści płynące z zastosowania automatyzacji testów. Artykuł “The Impacts of Test Automation on Software’s Cost, Quality and Time to Market” [8] skupia się na zmierzeniu wpływu automatyzacji na koszty, jakość i czas wytworzenia oprogramowania. Przeprowadzone badania jednoznacznie wskazują, że zastosowanie automatyzacji daje pozytywne rezultaty we wszystkich tych aspektach.

Rynek oferuje liczne rozwiązania, które pozwalają przeprowadzać zautomatyzowane testy. Wybór odpowiedniego narzędzia może być kluczowy, aby proces testowania przebiegał efektywnie. Próba wyłonienia najlepszego była podejmowana w wielu publikacjach. Jedną z nich jest “Comparative Study of Automated

Testing Tools: Selenium, Quick Test Professional and Testcomplete" [9]. Artykuł ten skupia się na trzech popularnych narzędziach do automatyzacji testów: Selenium, Testcomplete oraz Quick Test Professional. Autorzy porównują powyższe technologie pod względem kosztów licencji, wsparcia, czy też wymaganych umiejętności programistycznych użytkownika. Jednakże ostatecznie stwierdzają, że wybór narzędzia powinien być dostosowany do indywidualnych potrzeb danego projektu.

W kontekście testowania aplikacji opartych o JavaScript, czy TypeScript jednymi z popularniejszych narzędzi są Jasmine i Jest. Jedna z publikacji [10] porusza temat obu z nich i wskazuje, że Jest staje się coraz bardziej popularny i przewyższa liczbą pobrań konkurencyjne narzędzie Jasmine. Autorzy opisują, że zastąpili w swojej aplikacji Jasmine przez Jest i wskazują na jego zalety. Wśród nich wymieniają między innymi prostotę konfiguracji i samowystarczalność, rozumianą przez brak konieczności korzystania z dodatkowych środowisk do uruchamiania testów. Ponadto zwracają uwagę na bezproblemowość migracji z jednego narzędzia na drugie, która wynika z tego, że składnia testów, które były zaprogramowane w Jasmine, była niemal całkowicie kompatybilna z Jest.

Publikacja o tytule "Web based Automation Testing and Tools" [11] porusza większość podstawowych aspektów testowania aplikacji. Artykuł opisuje różne rodzaje testów, takie jak jednostkowe, integracyjne, systemowe, czy akceptacyjne. Praca porównuje także narzędzia do testowania aplikacji webowych – Selenium oraz TestNG. Ponadto artykuł zwraca uwagę na zasadność testowania aplikacji w procesie tworzenia oprogramowania, a także na to jak istotny jest odpowiedni dobór narzędzi oraz technik pisania testów.

Najpopularniejszym standardem pisania testów jest TDD (ang. Test Driven Development). Publikacja "Overview of the Test Driven Development Research Projects and Experiments" [12] omawia wspomniany standard na wielu płaszczyznach. Analizuje ona wyniki eksperymentów, przeprowadzonych między innymi przez firmy Microsoft oraz IBM, które miały na celu zbadanie wpływu TDD na proces tworzenia oprogramowania. Badania wykazały, że nie można jednoznacznie stwierdzić, jakoby TDD było lepszym podejściem testowania od tradycyjnego. Wykazano, że technika TDD dostarcza lepsze pokrycie testów, natomiast nie potwierdzono by jej zastosowanie skracało czas tworzenia oprogramowania.

Rozszerzeniem techniki TDD jest BDD (ang. Behaviour Driven Development). BDD łączy technikę TDD, a więc podejście polegające na pisaniu testów automatycznych przy dodawaniu funkcjonalności w ramach ciągłej integracji (ang. Continuous Integration, CI) podczas wytwarzania oprogramowania, oraz testy akceptacyjne użytkowników. Głównym założeniem BDD jest testowanie oprogramowania przy jednoczesnym spełnianiu oczekiwań klientów. Pracą, która kompleksowo charakteryzuje BDD jest "A Study of the Characteristics of Behaviour Driven Development" [13]. Według autorów głównymi cechami BDD są: wszechobecny język (ang. Ubiquitous Language), iteracyjny proces dekompozycji (ang. Iterative Decomposition Process), opis z historią użytkownika i szablonami scenariuszy (ang. Plain Text Description with User Story and Scenario Templates), zautomatyzowane testy akceptacyjne z regułami mapowania (ang. Automated Acceptance Testing with Mapping Rules), czytelny kod specyfikacji zorientowanej (ang. Readable Behaviour Oriented Specification Code) na zachowanie oraz zachowanie napędzane w różnych fazach (ang. Behaviour Driven at Different Phases).

Python jest obecnie jednym z najpopularniejszych języków programowania na świecie w wielu dziedzinach IT – od uczenia maszynowego, przez sztuczną inteligencję oraz Big Data, na programowaniu aplikacji internetowych kończąc [14]. Python dostarcza wiele gotowych bibliotek do testowania aplikacji internetowych. Książka "Python Unit Test Automation – Practical Techniques for Python Developers and Testers" [15] przedstawia opisy standardów pisania testów przy użyciu języka Python – na przykład TDD. Ponadto zawiera ona przegląd bibliotek takich jak: doctest, unittest, nose, nose2 czy pytest.

Inną pozycją traktującą o testowaniu w kontekście języka Python jest książka "Testing Python: Applying Unit Testing, TDD, BDD and Acceptance Testing" [16] autorstwa David'a Sale'a. Podobnie jak w przypadku [15], opracowanie to zawiera opisy bibliotek takich jak unittest, nose, doctest, pytest czy nose2, wraz z przykładowymi testami. Ponadto opisuje metody używane w poszczególnych bibliotekach (głównie unittest). Dodatkowo publikacja ta omawia narzędzia do zarządzania pakietami języka Python (pip), tworzenia wirtualnego środowiska (virtualenv) oraz kontroli wersji (SVN, git), a także przykłady ich użycia. Co równie istotne, książka David'a Sale'a porusza też temat tworzenia testów akceptacyjnych, przy użyciu szkieletu Robot, a także automatyzacji procesu uruchamiania i zarządzania testami poprzez narzędzie Jenkins, oraz pokrycia testów aplikacji narzędziem Coverage.

Książką podobną do poprzednich dwóch pozycji jest "Python Testing: Beginner's Guide" [17], której autorem jest Daniel Arbuckle. Analogicznie do wcześniej wspomnianych publikacji, zawiera ona opisy bibliotek służących do testowania w języku Python, wraz z przykładami. Co więcej, książka ta rozwija temat testowania, poświęcając więcej uwagi testowaniu integracyjnemu oraz testowaniu systemu. Podobnie jak książka David'a Sale'a, rozwja ona temat testów akceptacyjnych, natomiast w tym przypadku autor użył narzędzia Twill.

Jedną z najpopularniejszych bibliotek służących do testowania aplikacji jest biblioteka pytest. Książka "Python Testing with pytest" [18] jest kompletnym przeglądem tego narzędzia. Pozycja ta zawiera opis, konfigurację oraz przykłady użycia biblioteki, wraz ze wszystkimi funkcjonalnościami, technikami testowania oraz wtyczkami. Każdy rozdział obejmuje inną część biblioteki pytest, wraz z przykładami użycia oraz ćwiczeniami. Inną pozycją traktującą o podobnej tematyce do

wspomnianej książki jest "pytest Quick Start Guide: Write better Python code with simple and maintainable tests" [19].

Mając na uwadze, że w niniejszej analizie zostały porównane narzędzia unittest oraz pytest, ciekawą publikacją jest artykuł "Assessing the migration of testing frameworks in the Python ecosystem" [20]. Jest to publikacja badająca migrację szeregu aplikacji, projektów czy też systemów z narzędzia unittest do biblioteki pytest. Z badania wynika, że ze 100 projektów 34% używa obu narzędzi. Jednocześnie potwierdzona została teza o zmianie narzędzia na pytest. Wśród głównych powodów autorzy wymieniają takie czynniki jak łatwiejsza składnia i utrzymanie, elastyczność oraz ponowne użycie poszczególnych komponentów. Jednak należy zauważyć, że w podanym badaniu brak jakichkolwiek porównań pod kątem wydajności obu narzędzi.

2. Cel i zakres pracy

Celem pracy jest przeanalizowanie oraz porównanie wybranych narzędzi służących do automatyzacji testów aplikacji internetowych. Praca składa się z dwóch części – pierwsza jest poświęcona narzędziom do wykonywania testów aplikacji serwerowych, natomiast druga, tym do testów aplikacji klienckich. Wybrane narzędzia zostały przetestowane pod względem wydajności. Dane badawcze pozyskano poprzez zebranie czasów wykonania analogicznych testów jednostkowych napisanych z wykorzystaniem poszczególnych narzędzi. Dla części traktującej o narzędziach do testowania aplikacji serwerowych zostało wykorzystane otwarto-źródłowe API. W części skupiającej się na narzędziach służących do testowania aplikacji klienckich, testy zostały napisane w oparciu o przygotowaną aplikację testową.

Tezy:

1. Jasmine jest wydajniejszym narzędziem do wykonywania testów jednostkowych dla aplikacji napisanych z użyciem szkieletu programistycznego Angular.
2. unittest jest wydajniejszym narzędziem do wykonywania testów jednostkowych w kontekście języka Python.

Pytania badawcze:

1. Czy można wyłonić wydajniejsze od domyślnego narzędzie do automatyzacji testów jednostkowych dla aplikacji napisanych z użyciem Angular?
2. Jakie narzędzie sprawdza się najlepiej w tworzeniu automatycznych testów jednostkowych aplikacji zaprogramowanych z użyciem Angular?
3. Jakie narzędzie sprawdza się najlepiej w tworzeniu testów automatycznych serwerowych aplikacji webowych?

3. Plan eksperymentu

Na plan eksperymentu badawczego składają się następujące etapy:

1. Przygotowanie aplikacji, na podstawie których zostanie przeprowadzona analiza porównawcza
    a. Przygotowanie aplikacji testowej dla narzędzi do testowania części serwerowej
    b. Przygotowanie aplikacji testowej dla narzędzi do testowania części klienckiej
2. Przygotowanie środowisk dla poszczególnych narzędzi służących do przeprowadzania testów
3. Utworzenie scenariuszy testowych
4. Przeprowadzenie testów oraz zgromadzenie otrzymanych wyników
    a. Wykonanie serii testów w celu uzyskania wiarygodnego średniego czasu ich trwania
    b. Zapisanie czasów wykonania poszczególnych testów jednostkowych
5. Analiza otrzymanych wyników
6. Wnioski

W celu otrzymania wiarygodnych wyników i stabilnych wartości statystycznych, testy zostały wykonane w kilku seriach. Dla aplikacji serwerowej zostały one powtórzone 10-krotnie. Ta liczba powtórzeń pozwoliła na uzyskanie precyzyjnych wyników. Testy aplikacji klienckiej, ze względu na duże różnice w czasie wykonania testów pomiędzy badanymi narzędziami, powtórzono 20-krotnie. Dla takiej liczby wyniki zostały uznane za wiarygodne.

Testy dla części serwerowej i klienckiej zostały wykonane na innych urządzeniach. Parametry sprzętu, na którym przygotowano środowisko testowe dla części serwerowej prezentuje Tabela 1, natomiast w Tabeli 2 przedstawiono parametry sprzętu wykorzystanego przy części klienckiej.

Tabela 1: Parametry sprzętu do testów części serwerowej

Parametr	**Wartość**
Procesor	Intel Core i7-9750H
Pamięć RAM	8GB
Dysk	256GB SSD
Karta graficzna	NVIDIA GeForce GTX 1650
System operacyjny	Windows 11 Home

Tabela 2: Parametry sprzętu do testów części klienckiej

Parametr	**Wartość**
Procesor	Intel Core i7-7700HQ
Pamięć RAM	16GB
Dysk	512GB SSD
Karta graficzna	NVIDIA GeForce GTX 1050 Ti
System operacyjny	Windows 11 Home

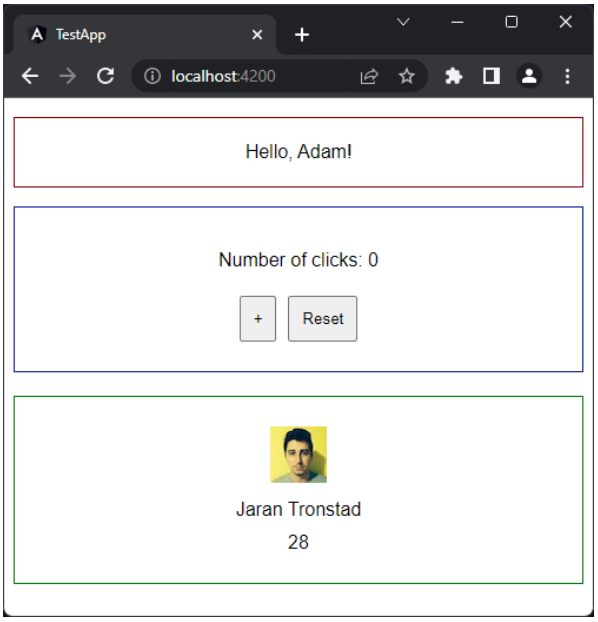
4. Aplikacje testowe

Na potrzeby przeprowadzenia badań zostały wykorzystane, specjalnie do tego celu przygotowane, aplikacje testowe. W oparciu o wspomniane aplikacje zaprogramowano testy jednostkowe z wykorzystaniem poddanych porównaniu narzędzi. Testy dla części serwerowej zostały zaprogramowane dla darmowego Api-Football w wersji 3.9.2. API to zwraca dane z zakresu tematyki piłki nożnej i pozwala pobrać historyczne dane dotyczące odbytych meczów, wyników, zespołów, stadionów, czy statystyk. Przykładowa odpowiedź z wykorzystanego API została przedstawiona na Rysunku 1.

```
    "get": "trophies",
    "parameters": {
        "player": "1100"
    },
    "errors": [],
    "results": 11,
    "paging": {
        "current": 1,
        "total": 1
    },
    "response": [
        {
            "league": "Community Shield",
            "country": "England",
            "season": "2022/2023",
            "place": "2nd Place"
        },
        {
            "league": "Bundesliga",
            "country": "Germany",
            "season": "2021/2022",
            "place": "2nd Place"
        },
```


Rysunek 1: Przykładowa odpowiedź z wykorzystanego API.

Dla części klienckiej zaprogramowano aplikację (Rysunek 2) z użyciem szkieletu Angular w wersji 15.2.6. Składa się ona z trzech niezależnych komponentów: GreetingComponent, ClickCounterComponent oraz UserComponent, które odpowiednio umożliwiają wyświetlenie powitania, obsługę licznika kliknięć oraz wyświetlenie danych użytkownika pobranych z zewnętrznego API.

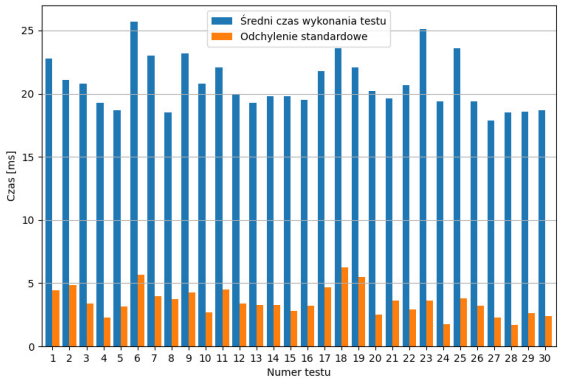


Rysunek 2: Kliencka aplikacja testowa.

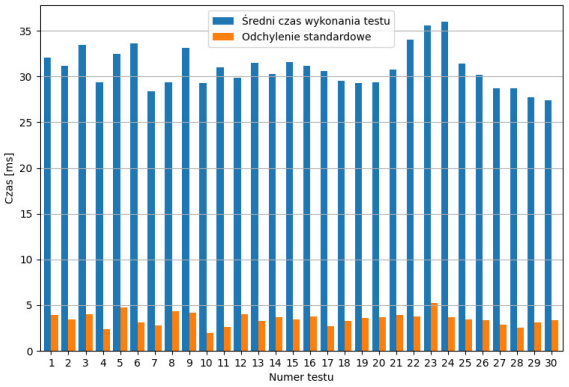
5. Wyniki badań

5.1. Wyniki badań narzędzi dla części serwerowej

Średnie czasy wykonania poszczególnych testów wraz z odchyleniami standardowymi dla biblioteki unittest zostały przedstawione na Rysunku 3, natomiast Rysunek 4 prezentuje te dane dla biblioteki pytest.



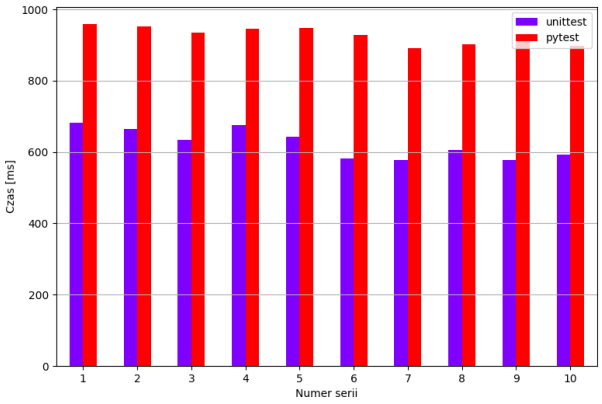
Rysunek 3: Średnie czasy wykonania oraz odchylenia standardowe dla testów napisanych z użyciem unittest.



Rysunek 4: Średnie czasy wykonania oraz odchylenia standardowe dla testów napisanych z użyciem pytest.

Analizując średnie czasy wykonania testów można stwierdzić, że testy napisane z wykorzystaniem unittest wykonywały się szybciej.

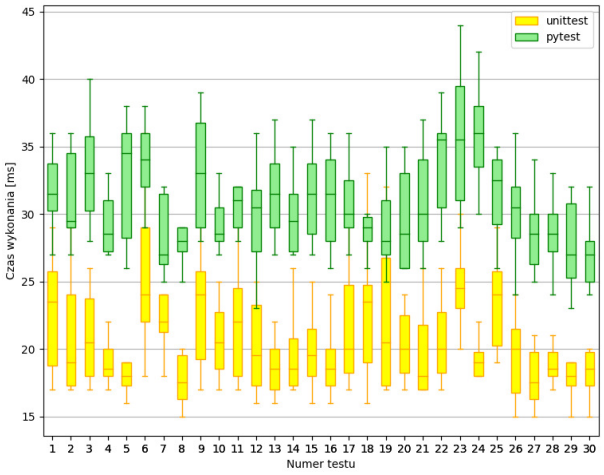
Wszystkie testy zostały powtórzone 10-krotnie, co pozwoliło uzyskać stabilne wartości statystyczne. Rysunek 5 prezentuje łączny czas wykonania wszystkich testów w kolejnych seriach. W każdej z nich unittest wypadł lepiej pod względem czasu wykonania, który średnio wynosił 624ms. Dla konkurencyjnej biblioteki pytest, średni czas potrzebny na wykonanie zestawu testów wyniósł 927ms.



Rysunek 5: Czasy wykonania wszystkich testów w poszczególnych seriach.

Na Rysunku 6 przedstawiono wykres pudełkowy przedstawiający czasy wykonania kolejnych testów dla poszczególnych bibliotek. Pokazuje on, że narzędzie

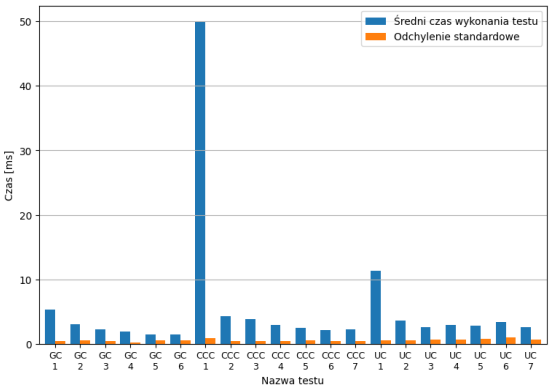
unittest jest bardziej wydajnym narzędziem niż pytest. Pomimo kilku wartości podobnych dla obu narzędzi, jak na przykład największy czas i najmniejszy czas wykonania danej serii dla przykładowego testu, można zauważyć trend, który jednoznacznie wskazuje narzędzie unittest jako bardziej wydajne. Ponadto na wykresie można zauważyć, że dla kilku testów (np. 12. oraz 23. test dla narzędzia pytest lub 18. test dla narzędzia unittest) różnica między maksymalnym a minimalnym czasem wykonania odbiega od różnic między tymi wartościami dla innych scenariuszy testowych.



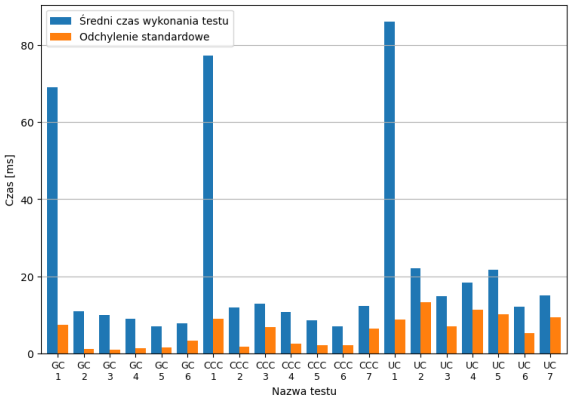
Rysunek 6: Rozkład czasu wykonania testów dla obu narzędzi.

5.2. Wyniki badań narzędzi dla części klienckiej

Średnie czasy wykonania oraz odchylenia standardowe poszczególnych testów dla narzędzi Jasmine oraz Jest przedstawiono odpowiednio na Rysunku 7 oraz 8. Wyniki zostały obliczone na podstawie 20 serii testów. Ta liczba powtórzeń pozwoliła na uzyskanie stabilnych wartości statystycznych i wiarygodnych wyników. Nazwy testów odpowiadają kolejnym testom napisanym dla poszczególnych komponentów aplikacji testowej. Testy *GC 1 – GC 6* są testami komponentu GreetingComponent, *CCC 1 – CCC 7* odpowiadają testom ClickCounterComponent, natomiast *UC 1 – UC 7* są testami UserComponent. Analizując średnie czasy wykonania poszczególnych testów można zauważyć, że w każdym przypadku narzędzie Jasmine było wielokrotnie szybsze od narzędzia Jest. Odchylenia standardowe są także znacznie mniejsze w przypadku Jasmine.

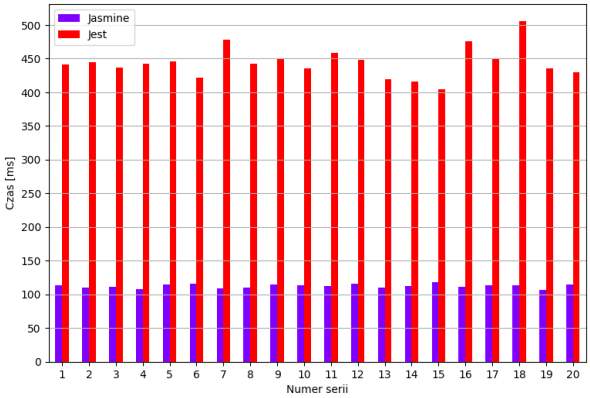


Rysunek 7: Średnie czasy wykonania oraz odchylenia standardowe dla testów napisanych z użyciem Jasmine.



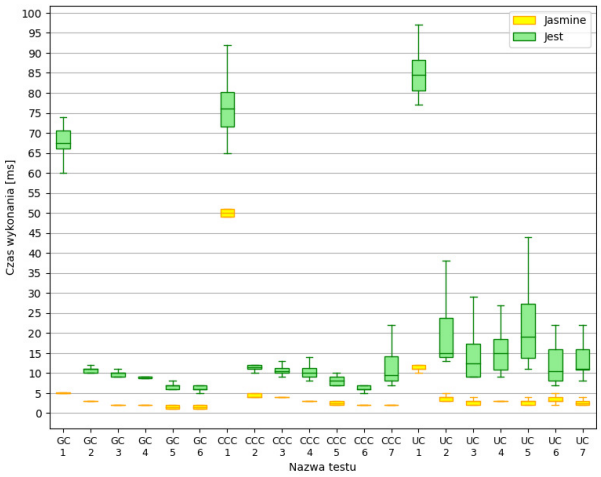
Rysunek 8: Średnie czasy wykonania oraz odchylenia standardowe dla testów napisanych z użyciem Jest.

Na Rysunku 9 przedstawiono czasy wykonania całego zestawu, składającego się z 20 testów, w poszczególnych seriach. Dla narzędzia Jasmine czasy te są bardzo podobne w każdej z serii i średnio wynoszą 113ms. W przypadku Jest można zaobserwować większą rozbieżność pomiędzy poszczególnymi wynikami, a średni czas potrzebny na wykonanie zestawu testów to 444ms.



Rysunek 9: Czasy wykonania wszystkich testów w poszczególnych seriach.

Rozkład czasu wykonania testów prezentuje wykres pudełkowy przedstawiony na Rysunku 10.



Rysunek 10: Rozkład czasu wykonania testów dla obu narzędzi.

Wykres ten potwierdza, że czasy wykonania testów dla narzędzia Jasmine są zauważalnie mniejsze. Ponadto pokazuje, że rozbieżność pomiędzy kolejnymi próbkami jest znacząco większa dla części testów wykonanych z wykorzystaniem narzędzia Jest. Można zatem stwierdzić, że jest ono mniej powtarzalne, jeśli chodzi o czas wykonywania się testów, od narzędzia Jasmine.

6. Wnioski

Niniejsza praca miała na celu porównanie ze sobą narzędzi służących do przeprowadzania automatycznych testów jednostkowych aplikacji internetowych. Porównane zostały narzędzia służące do testowania aplikacji serwerowych – unittest i pytest, oraz klienckich – Jasmine i Jest.

Przed przystąpieniem do realizacji eksperymentu zostały postawione tezy badawcze: *Jasmine jest wydajniejszym narzędziem do wykonywania testów jednostkowych dla aplikacji napisanych z użyciem szkieletu programistycznego Angular* oraz *unittest jest wydajniejszym narzędziem do wykonywania testów jednostkowych w kontekście języka Python*.

W celu przeprowadzenia eksperymentu badawczego przygotowano aplikacje testowe, w oparciu o które zostały zaprogramowane automatyczne testy jednostkowe. Czasy wykonania wspomnianych testów zostały zebrane i poddane analizie.

Po przeprowadzeniu eksperymentu badawczego, teza postawiona dla narzędzi do testów aplikacji klienckich znalazła swoje odzwierciedlenie w otrzymanych wynikach. Analiza czasów zebranych przez wielokrotne uruchomienie testów wykazała, że Jasmine jest znacznie wydajniejszym narzędziem niż Jest. Testy zaprogramowane z jego wykorzystaniem wykonywały się kilkukrotnie krócej, niż w przypadku konkurencyjnego narzędzia. Ponadto ich czasy wykonania były bardziej powtarzalne. Wyniki eksperymentu pozwalają odpowiedzieć na postawione pytania badawcze. Odpowiedź na pierwsze z nich – "*Czy można wyłonić wydajniejsze od domyślnego narzędzie do automatyzacji testów jednostkowych dla aplikacji napisanych z użyciem Angular?*" nie jest jednoznaczna. Wynika to z faktu, że z domyślnym narzędziem, którym jest Jasmine, zostało zestawione jedno konkurencyjne, a rynek oferuje też inne, mniej popularne narzędzia. Dlatego też w kontekście zestawionych w badaniu narzędzi odpowiedź na to pytanie jest negująca, jednakże nie można wykluczyć, że próba zestawienia Jasmine z innym narzędziem pozwoliłaby uzyskać odmienną odpowiedź. Drugie pytanie dotyczyło tego, które narzędzie sprawdza się najlepiej w tworzeniu automatycznych testów jednostkowych aplikacji zaprogramowanych z użyciem Angular. W tym przypadku badania wykazały, że pod względem wydajności lepiej sprawdza się Jasmine. Składnia testów dla obu narzędzi jest bardzo podobna, zatem konkurencyjny Jest nie wykazuje się widoczną przewagą w stosunku do domyślnego Jasmine. Można zatem wysnuć wniosek, że w kontekście aplikacji zaprogramowanych z wykorzystaniem szkieletu Angular, Jasmine sprawdza się lepiej.

Teza postawiona dla narzędzi do testów aplikacji serwerowych także została potwierdzona. Czasy wykonania testów stworzonych z wykorzystaniem biblioteki unittest były krótsze niż w przypadku pytest. Pomimo kilku omówionych przykładów, pokazujących duże różnice między maksymalnymi, a minimalnymi czasami wykonania poszczególnych testów, przeprowadzona analiza wyników jednoznacznie wskazuje narzędzie unittest jako bardziej wydajne. Nawiązując do postawionego pytania badawczego, dotyczącego tego jakie narzędzie sprawdza się najlepiej w tworzeniu testów automatycznych serwerowych aplikacji webowych, można stwierdzić, że wszystko zależy od przyjętych kryteriów. Jeśli najważniejszym kryterium jest wydajność, wtedy lepszym narzędziem jest biblioteka unittest. Jeśli głównymi czynnikami porównania jest prostota pisania testów, wtedy lepszym narzędziem jest pytest przez wzgląd na łatwiejszą składnię oraz obsługę testów.

Literatura

[1] M. E. Khan, F. Khan, Importance of Software Testing in Software Development Life Cycle, International Journal of Computer Science Issues (IJCSI) 11(2) (2014) 120-123.

[2] I. Bhatti, J. A. Siddiqi, A. Moiz, Z. A. Memon, Towards Ad hoc testing technique effectiveness in software testing life cycle, 2019 2nd International Conference on Computing, Mathematics and Engineering Technologies (iCoMET) (2019) 1-6.

[3] M. A. Umar, C. Zhanfang, A Study of Automated Software Testing: Automation Tools and Frameworks, International Journal of Computer Science Engineering 6 (2019) 217-225.

[4] P. Kunte, D. Mane, Automation Testing of Web based application with Selenium and HP UFT (QTP), International Research Journal of Engineering and Technology (IRJET) 6 (2017) 2579-2583.

[5] D. Raghuvanshi, Introduction to Software Testing, International Journal of Trend in Scientific Research and Development (IJTSRD) 4(3) (2020) 797-800.

[6] V. Garousi, M. V. Mäntylä, When and what to automate in software testing? A multi-vocal literature review, Information and Software Technology 76 (2016) 92-117.

[7] D. Ateşoğulları, A. Mishra, Automation testing tools: A comparative view, International Journal on Information Technologies & Security 12(4) (2020) 63-76.

[8] D. Kumar, K. K. Mishra, The impacts of test automation on software's cost, quality and time to market, Procedia Computer Science 79 (2016) 8-15.

[9] H. Kaur, G. Gupta, Comparative Study of Automated Testing Tools: Selenium, Quick Test Professional and Testcomplete, Int. Journal of Engineering Research and Applications 3(5) (2013) 1739-1743.

[10] E. Pernice, C. Albiston, R. Beeler, E. Chou, C. Fry, M. Shor, J. Spears, D. Speck, A. Thakur, S. West, Application Development in the Face of Evolving Web Technologies at the National Ignition Facility, 17th Int. Conf. on Acc. and Large Exp. Physics Control Systems (2019) 1052-1056.

[11] M. Sharma, R. Angmo, Web based automation testing and tools, International Journal of Computer Science and Information Technologies 5(1) (2014) 908-912.

[12] A. Bulajic, S. Sambasivam, R. Stojic, Overview of the test driven development research projects and experiments, Proceedings of Informing Science & IT Education Conference (InSITE) (2012) 165-187.

[13] C. Solis, X. Wang, A study of the characteristics of behaviour driven development, 37th EUROMICRO conference on software engineering and advanced applications (2011) 383-387.

[14] A. Rawat, A Review on Python Programming, International Journal of Research in Engineering, Science and Management 3(12) (2020) 8-11.

[15] A. Pajankar, Python Unit Test Automation: Practical Techniques for Python Developers and Testers, Apress, Nashik, 2017.

[16] D. Sale, Testing Python: Applying Unit Testing, TDD, BDD and Acceptance Testing, John Wiley & Sons, Chichester, 2014.

[17] D. Arbuckle, Python Testing: Beginner's Guide, Packt Publishing Ltd, Birmingham, 2010.

[18] B. Okken, Python Testing with pytest, Pragmatic Bookshelf, Raleigh, 2022.

[19] B. Oliveira, pytest Quick Start Guide: Write better Python code with simple and maintainable tests, Packt Publishing Ltd., Birmingham, 2018.

[20] L. A. Barbosa, Assessing the migration of testing frameworks in the Python ecosystem, Master Thesis, Universidade Federal de Minas Gerais, Belo Horizonte, 2022.

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 28 (2023) 236–241
Received: 2 June 2023
Accepted: 20 June 2023

Comparative analysis of methods for testing web applications

Analiza porównawcza sposobów testowania aplikacji internetowych

Tomasz Smyk*, Wojciech Superson, Małgorzata Plechawska-Wójcik

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

Abstract

The aim of the study was to conduct a comparative analysis of testing approaches for web applications in the two most popular architectures: monolithic and microservices. For the purpose of the study, the server-side of the application (backend) was implemented twice with identical functionalities for each of these architectures, allowing for a precise comparison of testing differences for the same program capabilities. The results revealed that the monolithic application was easier and faster to test. However, the microservices architecture requires more energy spent on testing, but allows better scalability and elasticity for independent teams to develop applications. Each of the examined architectures certainly has its own advantages and drawbacks. Furthermore, the conducted research indicates that unit tests require significantly less time to execute. However, when it comes to comprehensive code analysis, integration tests outperform unit tests by covering a substantial portion of the application's code with a single test. Nonetheless, the best comprehensive code analysis and protection against unwanted functional changes can be achieved by employing all known types of tests.

Keywords: unit testing; integration testing; microservices architecture; monolithic architecture

Streszczenie

Celem pracy była analiza porównawcza sposobów testowania aplikacji internetowych w dwóch najpopularniejszych architekturach, tj. monolitycznej oraz mikroserwisowej. Na potrzeby badania strona serwerowa aplikacji (*ang. backend*) została napisana dwukrotnie z identycznymi funkcjonalnościami w każdej z tych architektur, aby móc dokładnie zestawić różnice w testowaniu tych samych możliwości programu. Wyniki ukazały aplikację monolityczną, jako łatwiejszą i szybszą do testowaniu. Architektura mikroserwisowa natomiast wymaga większego nakładu pracy związanej z testowaniem, ale pozwala na uzyskanie większej skalowalności i elastyczność w rozwijaniu aplikacji przez niezależne zespoły. Każda z badanych architektur z pewnością ma swoje wady i zalety. Z przeprowadzonych badań wynika również, że testy jednostkowe potrzebują dużo mniej czasu na wykonanie, jednak jeśli chodzi o kompleksowość analizy kodu testy integracyjne zwyciężają, pokrywając jednym testem znaczną część kodu aplikacji. Najlepszą jednak kompleksowość analizy kodu i zabezpieczenie aplikacji przed niechcianymi zmianami funkcjonalności uzyskamy stosując wszystkie znane nam rodzaje testów.

Słowa kluczowe: testy jednostkowe; testy integracyjne; architektura mikroserwisowa; architektura monolityczna

*Corresponding author

Email address: **tomasz.smyk@pollub.edu.pl** (T. Smyk)

1. Wstęp

Celem niniejszej pracy jest analiza i porównanie sposobów testowania aplikacji internetowych. Przedstawione zostaną różne sposoby, którymi obecnie rynek komercyjny zwykł testować aplikacje przed, w trakcie i po wdrożeniu. Testowanie aplikacji znacząco różni się w zależności od zastosowanej architektury oraz czy jest to warstwa serwerowa aplikacji, czy też warstwa kliencka. Podstawowe pytanie badawcze pracy brzmi: które rodzaje testów są bardziej nastawione na wydajność wykonania, a które na kompleksową analizę aplikacji kosztem właśnie wydajności. Kolejnym pytaniem, które nasuwa się podczas analizy tematu jest czy obecnie popularny, mikroserwisowy, podział systemu aplikacji wpływa na dywersyfikację typów testów inaczej w przypadku klasycznego monolitycznego sposobu podziału aplikacji.

Każda z metod testowania aplikacji internetowych ma swoje wady, ale również posiada zalety. Skuteczność danego rodzaju testu może być zależna od wielu czynników - od stopnia skomplikowania systemu aż po stopień zintegrowania danej aplikacji z innymi. Celem niniejszej pracy jest przeprowadzenie analizy porównawczej sposobów testowania aplikacji internetowych. W ten sposób autorzy postarają się udowodnić, że testy kontraktowe pozwalają na lepsze i bardziej kompleksowe przetestowanie kodu aplikacji mikroserwisowych w systemie informatycznym w porównaniu do zwykłych testów integracyjnych.

2. Przegląd literatury

Obecnie standardem można nazwać tworzenie systemów informatycznych w architekturze mikroserwisów [1]. Jest to spowodowane skalowalnością, prostotą utrzymania, łatwością wdrażania poprawek itp. Natomiast niewątpliwym problemem tego sposobu tworzenia systemów informatycznych jest trudność testowania sposobu komunikacji [2, 3]. Obecnie standardem powoli stają się testy oparte o kontrakty komunikacji między

modułami systemu [4]. Jednocześnie, ciągle największy nacisk kładziony jest na testy jednostkowe i integracyjne, a im większa aplikacja tym bardziej rozkład testów jest zbliżony do klasycznego piramidowego znanego z aplikacji monolitycznych [5].

Obecnie popularność zyskuje budowanie aplikacji internetowych z gotowych komponentów, które zostały wcześniej przygotowane w taki sposób, by były niezależne i umożliwiały możliwie najprostsze rozwiązania programistyczne pod względem rozszerzalności kodu produkcyjnego tak, aby podłączony komponent można było z łatwością dostosować do wymagań klienta. Jest to niesamowite ułatwienie w porównaniu do archaicznych już modeli aplikacji opartych o architekturę monolityczną. Jednak, aby upewnić się o poprawności działania takiego nowego, dopiero, co podłączonego komponentu, należy do niego napisać odpowiednie testy, w taki sposób, aby po podłączeniu testy automatycznie zweryfikowały poprawność działania aplikacji w sposób automatyczny, a przynajmniej pół-automatyczny [2].

Taki trend tworzenia aplikacji z gotowych komponentów, czyli w architekturze mikroserwisowej zapanował z powodu możliwości zwinnego tworzenia komponentów do tych aplikacji, niezależności pisania jednego kodu produkcyjnego od drugiego, możliwości łatwego wdrażania aplikacji i w następstwie łatwych poprawek programistycznych a także skalowalności systemu zbudowanego z takich komponentów [1]. Takie zalety prowadzą jednak do konieczności zmiany postrzegania testowania aplikacji z zachowaniem klasycznej piramidy [3]. Przy architekturze klasycznego monolitu, testy powinny mieć rozkład prezentowany przez piramidę, jednak w architekturze mikroserwisowej powinno się skupiać się zdecydowanie na komunikacji wewnętrznej i zewnętrznej między serwisami, a testy jednostkowe zostawić na niezbędne pokrycie ścisłej logiki biznesowej aplikacji.

Nie należy jednak tutaj zaprzeczać, że testy jednostkowe dalej odgrywają kluczową rolę w procesie rozwoju oprogramowania [6]. Pozwalają one na niezależne, w pewien sposób hermetyczne, zapewnienie programisty o tym, że logika biznesowa, która zaimplementował w swojej aplikacji, jest zgodna z założeniami [7]. Tak napisane testy są swoistym zabezpieczeniem nie tylko przed błędem podczas pierwotnej implementacji wymagań, ale także podczas późniejszego rozwijania aplikacji, kiedy to inni programiści mogą upewnić się o poprawności napisanego przez siebie kodu w stosunku do już istniejącego w aplikacji [8]. Bez takich zapewnień ciężko jest tworzyć kod bez wad.

Mimo, że testy jednostkowe są tak niezaprzeczalnie ważne, to, jak zostało już wspomniane, nie gwarantują one niezawodności działania całego systemu informatycznego, szczególnie takiego, który opiera swoje działanie na komunikacji między swoimi komponentami. Uzupełnieniem testów jednostkowych o zapewnienie o poprawności komunikacji są testy integracyjne [9]. Pozwalają one przetestować przetwarzanie danych między modułami aplikacji lub między aplikacjami. Są one jednak wolniejsze od jednostkowych, bardziej kosztowne w utrzymaniu a także mniej czytelne dla programisty, co jest logiczne, jeśli weźmie się pod uwagę, że testują większe części aplikacji w porównaniu do testów jednostkowych. Istnieje jednak rodzaj testów integracyjnych, który nie jest dużo wolniejszy od testów jednostkowych, a który gwarantuje poprawność komunikacji między komponentami systemu. Są to testy kontraktowe, czyli testy swoistej umowy zawartej między 2 modułami systemu. Innymi słowy, test kontraktowy wykorzystuje zdefiniowany kontrakt, aby automatycznie zweryfikować, czy nie została złamana komunikacja między komponentami [9]. W efekcie otrzymuje się wydajny test, którego zaletą jest też możliwość wykonania po obu stronach komunikacji.

Internetowa treść sieci przekształciła się z prostych stron internetowych w złożone aplikacje. Aby zapewnić jakość oprogramowania i sprawność interfejsu użytkownika, aplikacja powinna być testowana za pomocą testów jednostkowych i integracyjnych [10]. Jednak modyfikacje aplikacji mogą prowadzić do niezamierzonych zmian wizualnych, które nie są wykrywane przez testy funkcjonalne. Można przetestować tę regresję wizualną, przechwytując stan aplikacji jako zrzut ekranu i porównując go z obrazami z poprzednich wersji aplikacji.

Można przyjąć, że jest mało testów sprawdzających wydajność kodu, a dużo testów jednostkowych sprawdzających logikę biznesową (funkcjonalność) [5]. Można to zauważyć, licząc wystąpienia w publicznych repozytoriach kodu wykorzystywanego do analizy wydajności kodu w popularnych szkieletach programistycznych służących do pisania testów w języku Java.

Obsługa różnych przeglądarek i ich różnych wersji jest jednym z głównych wyzwań dla wielu aplikacji internetowych [11,12]. Kod JavaScript uruchamiany w przeglądarce Safari może nie działać poprawnie w innych przeglądarkach, takich jak Internet Explorer, Firefox lub Google Chrome. To wyzwanie wynika z braku testów jednostkowych kodu JavaScript. Jednak za pomocą popularnych szkieletów programistycznych, takich jak Jasmine, YUI Test, QUnit i JsTestDriver [13], można skutecznie tworzyć i automatyzować testy JavaScript dla aplikacji internetowych [14].

Test Driven Development (TDD) [15] to metoda programowania, w której najpierw definiuje się testy jednostkowe dla kodu, a następnie implementuje się sam kod. Celem TDD jest zapewnienie jakości i wydajności kodu poprzez ciągłe testowanie go i dostosowywanie do wymagań. Podejście TDD polega na tym, że testowanie staje się głównym czynnikiem napędowym projektowania, dokumentacji, łatwości utrzymania i jakości kodu. W przypadku języków dynamicznych, na przykład JavaScript, gdzie nie otrzymuje się ostrzeżeń o błędach podczas kompilacji, testowanie staje się kluczowym elementem łączącym duże aplikacje.

W dzisiejszych czasach jednym z kluczowych czynników sukcesu każdej firmy jest jej widoczność w Internecie [16]. Dlatego tak ważne jest, aby każda firma posiadała odpowiednią stronę internetową, która może być np. sklepem internetowym. Projektowanie takich stron nie jest łatwe, ponieważ trudno zapewnić, że będą one działać poprawnie we wszystkich nowoczesnych przeglądarkach [17]. Również dla testerów oprogramowania stanowi to wyzwanie - muszą oni sprawdzić, czy strona działa poprawnie we wszystkich przeglądarkach. W artykule [18] autorzy skupili się na różnych narzędziach do automatyzacji testów dostępnych na rynku, zarówno tych otwartych, jak i płatnych. Porównują je pod kątem ich przydatności i wyciągają wnioski. Badanie to pokazuje, że skuteczne testowanie w różnych przeglądarkach jest możliwe dzięki wykorzystaniu narzędzi do automatyzacji.

3. Teoria testów aplikacji internetowych

Wykorzystywanie różnych rodzajów testów jest kluczowe dla prawidłowego zrozumienia procesu testowania oprogramowania. Jest wiele rodzajów testów dedykowanych różnym celom: testy jednostkowe, testy integracyjne, testy kontraktowe oraz inne testy, które są stosowane w procesie weryfikacji jakości oprogramowania.

Testy jednostkowe to rodzaj testów oprogramowania, które umożliwiają pozbycie się zależności wynikających z naturalnego sposobu pisania kodu produkcyjnego. Testy te skupiają się na testowaniu najmniejszych wydzielonych komponentów kodu, takich jak klasy w programowaniu obiektowym, w celu sprawdzenia poprawności realizacji logiki biznesowej zleconego zadania programistycznego. Testy jednostkowe są niezależne od innych komponentów systemu oraz kontekstu aplikacyjnego, co umożliwia ich szybkie wykonanie i łatwe zrozumienie. Ich wykonanie na etapie tworzenia oprogramowania pozwala na szybkie wykrycie i poprawienie błędów w kodzie, co prowadzi do oszczędności czasu i pieniędzy w dalszych fazach projektu. Ponadto, testy jednostkowe są ważne dla utrzymania jakości kodu oraz ułatwienia jego późniejszej rozbudowy, co umożliwia wprowadzenie zmian bez ryzyka powstawania błędów w innych częściach systemu.

Testy integracyjne są kluczowe dla weryfikacji poprawności interakcji pomiędzy poszczególnymi modułami lub komponentami, które tworzą całość systemu. Testy te są przeprowadzane w środowisku testowym, w którym stosowane są specjalnie przygotowane dane testowe, aby zapewnić spójność i niezmienną jakość wyników testów. W odróżnieniu od testów jednostkowych, które skupiają się na weryfikacji poprawności kodu każdego modułu oddzielnie, testy integracyjne są bardziej skomplikowane i czasochłonne. Jednakże, ich wartość jest nieoceniona, ponieważ pozwalają na wczesne wykrywanie i naprawianie błędów w interakcjach pomiędzy modułami, co minimalizuje ryzyko wystąpienia poważniejszych problemów w systemie w przyszłości. Testy integracyjne powinny być przeprowadzane systematycznie i regularnie, co znacznie zwiększa stabilność, niezawodność i wydajność systemu, a także zapewnia spójność danych i interakcji między różnymi modułami.

Testy kontraktowe są ważnym narzędziem testowania oprogramowania, które służy do zapewnienia poprawnej i spójnej interakcji pomiędzy różnymi modułami lub serwisami. Ich celem jest przetestowanie, czy kontrakty, czyli umowy między nimi, są zachowane i czy komunikacja między nimi jest zgodna z oczekiwaniami. Testy kontraktowe pozwalają na wczesne wykrycie błędów w kodzie oraz na szybsze znalezienie i rozwiązanie problemów związanych z integracją oprogramowania. Testy kontraktowe opierają się na zdefiniowaniu interfejsów API między modułami lub serwisami, a następnie na przetestowaniu, czy te interfejsy są zgodne z założeniami. W przypadku testów jednostkowych, programista testuje poszczególne funkcje lub metody w izolacji, podczas gdy testy kontraktowe uwzględniają całą sieć powiązań między różnymi modułami.

Ważne jest, aby testy kontraktowe były pisane w sposób czytelny i łatwy do zrozumienia dla innych członków zespołu. Dzięki temu testy kontraktowe mogą służyć nie tylko jako narzędzie testowania, ale również jako dokumentacja dla projektu. Testy kontraktowe są szczególnie przydatne w projektach, w których wiele zespołów pracuje nad różnymi modułami lub serwisami, a integracja między nimi jest kluczowa dla poprawnego funkcjonowania systemu jako całości. W takich projektach testy kontraktowe pozwalają na szybsze wykrycie i rozwiązanie problemów związanych z integracją.

Przetestowanie aplikacji monolitycznych zgodnie z dobrymi praktykami i nowoczesnymi technologiami może być nie tylko przyjemne, ale także satysfakcjonujące. Dzięki nowoczesnym narzędziom testowym i podejściu do testowania, deweloperzy są w stanie szybko przeprowadzić testy jednostkowe i integracyjne, co pozwala na wykrycie błędów i szybsze wprowadzenie poprawek.

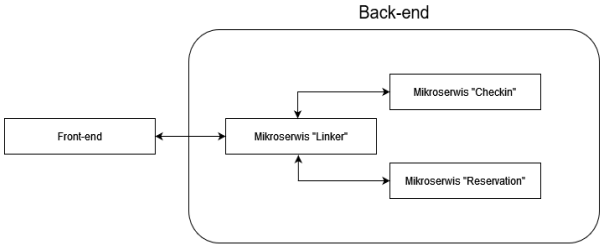
Testowanie jest także kluczowym elementem architektury mikroserwisowej. Wymaga ono podejścia, które uwzględnia testowanie integracyjne oraz testowanie jednostkowe każdego z mikroserwisów, ponieważ system składa się z wielu mniejszych usług. Testy kontraktowe są ważnym rodzajem testów w architekturze mikroserwisowej. Są one szczególnie istotne, gdy wiele mniejszych usług komunikuje się ze sobą za pomocą API.

4. Plan badań

Badanie miało na celu przeprowadzenie analizy porównawczej testowania aplikacji internetowych z uwzględnieniem różnych rodzajów testów, takich jak jednostkowe, integracyjne i kontraktowe. Jednak oprócz samego porównania wydajnościowego, skupiono się również na roli, jaką architektura aplikacji odgrywa w ilości i rodzaju testów pisanych podczas fazy rozwoju

aplikacji. Aplikacja miała na celu umożliwienie rezerwacji biletów na filmy (wybranie sali oraz miejsca na niej) oraz odprawę w kinie za pomocą kodu odprawy otrzymanego podczas rezerwacji.

W pierwszej aplikacji wykorzystano architekturę mikroserwisową, która składała się z trzech mikroserwisów: "linker", "reservation" i "checkin". Każdy z tych mikroserwisów miał architekturę Domain-Driven Design (DDD). Mikroserwis "linker" pełnił rolę pośrednika, komunikującego się zarówno z aplikacją front-end, jak i pozostałymi mikroserwisami. Pozostałe mikroserwisy, "reservation" i "checkin", nie komunikowały się bezpośrednio z zewnętrznym światem, udostępniając swoje API tylko dla "linkera". Opisana komunikację przedstawia Rysunek 1.



Rysunek 1: Graficzna reprezentacja komunikacji w aplikacji.

W drugiej aplikacji zaimplementowano tę samą funkcjonalność, ale jako aplikację monolityczną. Cała logika i funkcjonalność zostały skonsolidowane w jednym projekcie.

Przeprowadzenie porównawczej analizy tych dwóch aplikacji pozwoliło na zrozumienie wpływu architektury aplikacji na proces testowania. W przypadku aplikacji mikroserwisowej, testowanie było bardziej skomplikowane ze względu na rozproszenie logiki biznesowej i konieczność zarządzania komunikacją między mikroserwisami. Testy integracyjne miały kluczowe znaczenie, aby upewnić się, że poszczególne mikroserwisy komunikują się ze sobą poprawnie i spełniają oczekiwane wymagania.

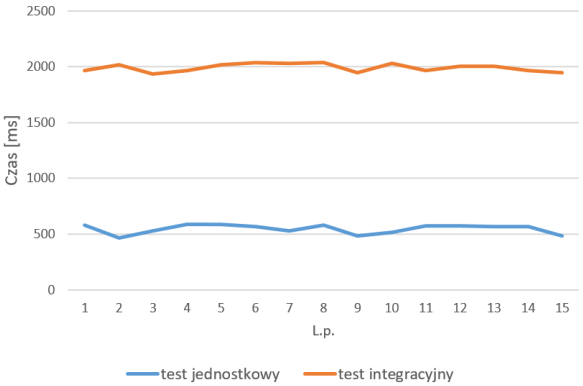
5. Analiza wyników badań

5.1. Wydajność różnych rodzajów testów

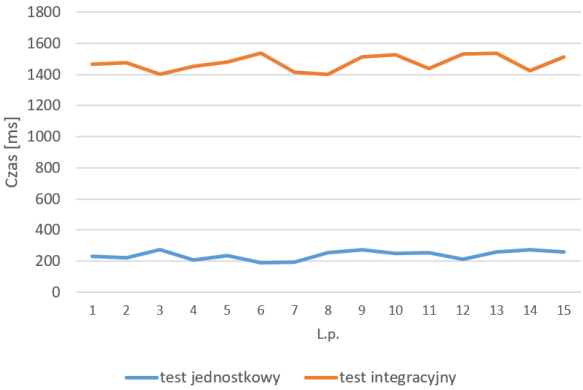
Przeprowadzona analiza dotycząca czasów wykonania różnych rodzajów testów w każdej z zaimplementowanych aplikacji w architekturze mikroserwiowej dała wyniki przedstawione graficznie na Rysunku 2.

Takie samo badanie zostało przeprowadzone w aplikacji w architekturze monolitycznej i dane przedstawione są na Rysunku 3.

W mikroserwisach stosunek czasu wykonania pojedynczego testu jednostkowego do czasu wykonania pojedynczego testu integracyjnego wynosi około 1:4, natomiast w monolicie stosunek ten oscylował w granicach 1:6. Oznaczałoby to, że w architekturze monolitycznej testy jednostkowe wykonują się szybciej, patrząc na dane testy integracyjne wykazują podobną tendencję. Wynika to oczywiście z faktu, że każdy z mikroserwisów musi przygotować kontekst frameworka Spring, skompilować kod osobno i wykonać testy w ‘świeżym’ środowisku. W aplikacji monolitycznej dzieje się to raz i wszystkie testy mogą zostać wykonane.



Rysunek 2: Graficzna reprezentacja czasów w milisekundach potrzebnych do wykonania pojedynczego testu w aplikacji w architekturze mikroserwisowej.



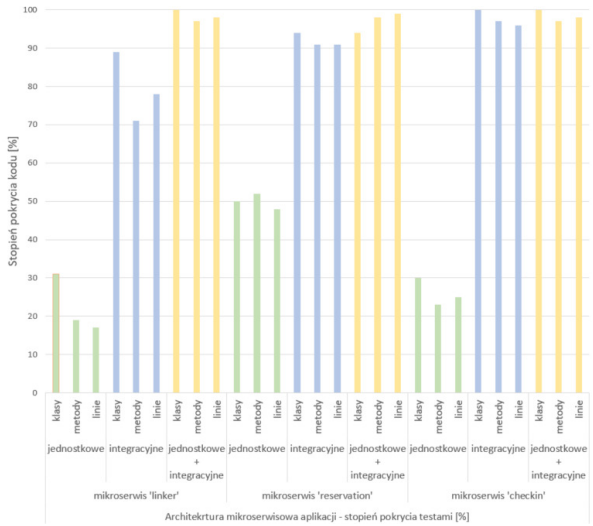
Rysunek 3: Graficzna reprezentacja czasów w milisekundach potrzebnych do wykonania pojedynczego testu w aplikacji w architekturze monolitycznej.

5.2. Kompleksowość analizy kodu

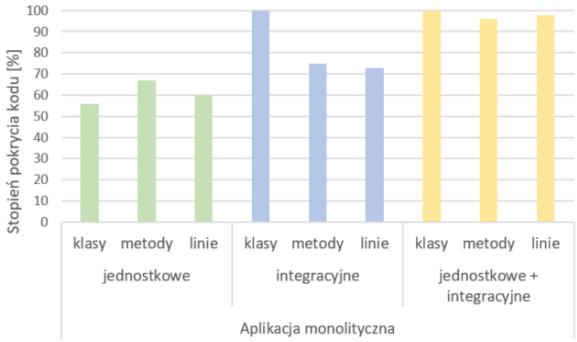
Analiza kompleksowości analizy kodu została przeprowadzona na podstawie stopnia pokrycia kodu testami. Wyniki badań przedstawiono na Rysunkach 4 oraz 5.

We wszystkich aplikacjach pokrycie kodu testami jest niemal stuprocentowe, co najlepiej pozwoliło ukazać, który rodzaj testów jest niezbędny. Mimo tego, że testy integracyjne ukazują większy stopień pokrycia kodu nie są idealne i nie można osiągnąć nimi wszystkiego. Testami jednostkowymi dałoby się oczywiście osiągnąć stuprocentowe pokrycie, jednak w tym przypadku lepiej sprawdziły się jako uzupełnienie do sprawdzania miejsc ciężko dostępnych przy testowaniu integracyjnym. Z analizy wyników można wywnioskować, że najlepszym wyjściem jest stosowanie obu rodzajów testów symultanicznie. Przeprowadzenie badań kompleksowości analizy kodu zostało wykonane przy użyciu narzędzia ‘Code

coverage' programu IntelliJ IDEA. Jest to narzędzie pokazujące miejsca, które już są pokryte testami oraz wyszczególniające miejsca, których testy na ten moment nie sprawdzają.



Rysunek 4: Graficzna reprezentacja stopnia pokrycia kodu w procentach w aplikacjach w architekturze mikroserwisowej.

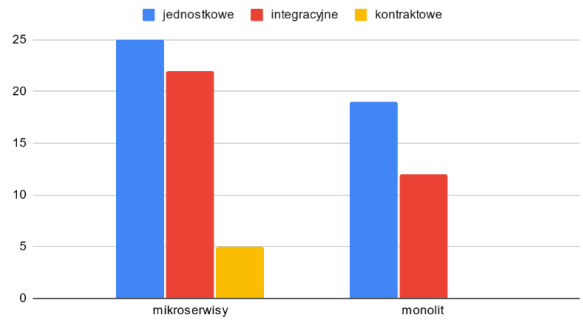


Rysunek 5: Graficzna reprezentacja stopnia pokrycia kodu w procentach w aplikacji w architekturze monolitycznej.

5.3. Zróżnicowanie testów pod względem architektury

Jak wspomniano wcześniej, realizując badania napisano aplikację dwukrotnie, raz w architekturze mikroserwisowej a raz w architekturze monolitycznej, jednak z identyczną funkcjonalnością. W ramach badań test kompleksowo przetestowano testami jednostkowymi, integracyjnymi i kontraktowymi. Następnie policzono wystąpienia każdego rodzaju testu w każdej architekturze. Wyniki przedstawia wykres na Rysunku 6.

Analizując przedstawiony wykres można zauważyć, że, mimo iż logika biznesowa w obu badanych architekturach jest identyczna, to ilość testów jest całkowicie inna. Architektura oparta o mikroserwisy ma zdecydowanie więcej testów jednostkowych i integracyjnych, a dodatkowo decydując się na tego rodzaju architekturę trzeba również wykonać testy kontraktowe, których w monolicie nie ma potrzeby pisać.



Rysunek 6: Graficzna reprezentacja ilości różnych rodzajów testach w aplikacji w dwóch różnych architekturach.

5.4. Porównanie testów kontraktowych i integracyjnych

Jako, że oba rodzaje testów - testy kontraktowe i integracyjne - służą zapewnieniu pewności w poprawności działania komunikacji między różnymi częściami systemu, aby odpowiedzieć na pytanie czy skuteczność testów kontraktowych jest większa w porównaniu z testami integracyjnymi, należy zawęzić zakres testowalności dla obu tych rodzajów testów. Można z całą pewnością stwierdzić, że zakres zastosowalności testów integracyjnych jest o wiele szerszy niż testów kontraktowych. Przykładem takiej komunikacji może być testowanie komunikacji między modułami lub między aplikacją a bazą danych. Testy kontraktowe nie sprawdzą takiego rodzaju komunikacji, nie były też tworzone z taką myślą. Zatem w celu poprawnego porównania autorzy tekstu zdecydowali się zawęzić oba rodzaje do jednego celu: sprawdzania programistycznego interfejsu aplikacji.

Podczas rozwoju systemu napisanych zostało kilka testów kontraktowych oraz kilka testów integracyjnych spełniających założenia. Oczywiście, należy mieć na uwadze, że ich realizacja następuje po obu stronach aplikacji całkowicie inaczej.

Po stronie dostawcy API test kontraktowy jest prostszy w napisaniu - wymaga tylko zdefiniowania testowych danych w pliku kontraktu. Test integracyjny natomiast zawiera logikę komunikacji, co utrudnia jego pisanie. Dodatkowo, właściwy, czyli wykonywalny, test kontraktowy jest automatycznie generowany z napisanego kontraktu i ma postać czytelną, bardzo podobną do tego, co w branży przyjęło się pisać w testach integracyjnych sprawdzających komunikację API. Natomiast test integracyjny można uruchomić w dowolnym momencie, tak jak każdy inny test, w przeciwieństwie do testu kontraktowego, który można uruchomić tylko w fazie budowania aplikacji, która musi się zakończyć, aby uzyskać rezultat wykonania wszystkich testów, i na tej podstawie, albo w logach budowania, sprawdzić czy dany test kontraktowy wykonał się pomyślnie.

Po stronie konsumenta API test kontraktowy jest o wiele łatwiejszy w utrzymaniu, nie wymaga zasobów oraz pracy nad utrzymaniem dostępności testowej usługi. Aby test integracyjny nie był zależny od testowej instancji dostarczyciela API, musi posiadać logikę, która faktycznie nie sprawdza implementacji komunikacji między mikroserwisami, a więc nie spełnia swojego podstawowego założenia. Staje się w takim wypadku testem sprawdzającym wewnętrzną komunikację aplikacji. W ten sposób powstaje luka w zabezpieczeniu systemu przed zmianami. Ma jednak przewagę nad testem kontraktowym w momencie, kiedy integracja następuje z zewnętrznym serwisem, który nie dostarcza definicji kontraktów. W tym momencie nie da się napisać testu kontraktowego, a test integracyjny jest jedynym wyjściem.

6. **Wnioski**

Testy integracyjne są zdecydowanie bardziej nastawione na kompleksową analizę funkcjonalności aplikacja w porównaniu do testów jednostkowych, które są skupione bardziej na pojedynczym uniezależnionym od kontekstu aplikacji procesie, przez co są zdecydowanie bardziej wydajne. Taki wniosek pozwala przyjąć podstawowe pytanie badawcze postawione w pracy. Architektura mikroserwisowa wymaga większego nakładu pracy związanej z testowaniem, ale pozwala na większą modularność i elastyczność w rozwijaniu aplikacji. Natomiast architektura monolityczna może być bardziej efektywna pod względem czasu wykonania testów, ale może ograniczać skalowalność i separację odpowiedzialności w systemie. Ostateczny wybór architektury zależy od konkretnych wymagań i preferencji projektowych. Z tego powodu można stwierdzić, że wykorzystana w systemie architektura wpływa dosyć znacznie na dywersyfikację rodzajów testów wykorzystanych w komponentach systemu. Jednocześnie, mając tylko na uwadze integrację między identycznymi warstwami dostępu, testy kontraktowe okazały się lepszym wyborem, jeśli tylko mogły być zastosowane. Stosowalność tego rodzaju testów jest ograniczona jednak dosyć mocno od integracyjnych, jednak porównując tylko zakres wspólnej zastosowalności testy kontraktowe okazują się lepszym wyborem.

Literatura

[1] J. P. Sotomayor, S. C. Allala, P. Alt, J. Phillips, T. M. King, P. J. Clarke, Comparison of runtime testing tools for microservices, Annual Computer Software and Applications Conference (COMPSAC) 43(2) (2019) 356-361.

[2] H. G. Gross, C. Atkinson, F. Barbier, Component integration through built-in contract testing, Component-based software quality, Lecture Notes in Computer 2693 (2003) 159-183.

[3] H. Fischer, Testing in microservice systems: a repository mining study on open-source systems using contract testing, GUPEA, Gothenburg, 2021.

[4] F. Selleby, Creating a Framework for Consumer-Driven Contract Testing of Java APIs, Bachelor's degree, Linköping University, Linköping, 2018.

[5] P. Stefan, V. Horky, L. Bulej, P. Tuma, Unit testing performance in java projects: Are we there yet?, Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering (2017) 401-412.

[6] R. Pawlak, Testowanie oprogramowania. Podręcznik dla początkujących, Helion, Gliwice, 2014.

[7] R. Dahiya, A. Shahid, Importance of Manual and Automation Testing, CS & IT Conference Proceedings 9(17) (2019) 6-13.

[8] G. Fink, F. Ido, JavaScript Unit Testing, Pro Single Page Application Development: Using Backbone, JS and ASP. Net, Apress, Berkeley, 2014.

[9] D. Raghuvanshi, Introduction to Software Testing, International Journal of Trend in Scientific Research and Development (IJTSRD) 4(3) (2020) 797-800.

[10] M. Vesikkala, Visual regression testing for web applications, Master's thesis, Aalto University, Espoo, 2014.

[11] H. Achkar, Model Based Testing of Web Applications, The Science Technicians' Association of New Zealand Conference (2010) 11-19.

[12] Z. Qian, M. Huaikou, Z. Hongwei, a practical web testing model for web application testing, 2007 third international IEEE conference on signal-image technologies and internet-based system (2007) 434-441.

[13] N. Antunes, M. Vieira, Penetration testing for web services, Computer 47(2) (2013) 30-36.

[14] H. Saleh, JavaScript Unit Testing, Packt Publishing, Mumbai 2013.

[15] T. Kleivane, Unit Testing with TDD in JavaScript, Master's thesis, Institutt for datateknikk og informasjonsvitenskap, Trondheim, 2011.

[16] B. Kaalra, and K. Gowthaman, Cross Browser Testing Using Automated Test Tools, International Journal of Advanced Studies in Computers, Science and Engineering 3(10) (2014) 7-13.

[17] P. Tonella, R. Filippo, Web Testing: a Roadmap for the Empirical Research, Seventh IEEE International Symposium on Web Site Evolution (2005) 63-70.

[18] H. V. Gamido, M. V. Gamido, Comparative review of the features of automated software testing tools, International Journal of Electrical and Computer Engineering (IJECE) 9(5) (2019) 4473-4480.

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 28 (2023) 242–247
Received: 2 June 2023
Accepted: 3 July 2023

Performance comparison of microservices written using reactive and imperative approaches

Porównanie wydajności mikroserwisów napisanych w oparciu o podejście reaktywne i imperatywne

Kacper Mochniej*, Marcin Badurowicz

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

Abstract

The purpose of this paper was to compare the performance of microservices based on reactive and imperative approaches. To accomplish this task, two microservice applications written in Java using the Spring programming framework were developed. The Spring Web and Spring Webflux modules were used for the conventional and reactive versions, respectively. During the tests, functionalities related to operations of retrieving and inserting records into the database, data processing and file transfer were invoked. The Gatling tool was used to conduct the tests. The tests showed that reactive microservices can be more efficient in particular when there are delays in communication with services or the database. Otherwise, it depends on the complexity of the operations being performed. Microservices based on the reactive paradigm also use less RAM compared to conventional counterparts.
Keywords: microservices; reactive programming; imperative programming

Streszczenie

Celem pracy było porównanie wydajności mikroserwisów opartych o podejście reaktywne i imperatywne. Aby wykonać zadanie, stworzono dwie aplikacje mikroserwisowe napisane w języku Java z użyciem szkieletu programowania Spring. Wykorzystane zostały moduły Spring Web oraz Spring Webflux odpowiednio dla wersji konwencjonalnej i reaktywnej. W trakcie badań wywoływane były funkcjonalności związane z operacjami pobierania i wstawiania rekordów do bazy danych, przetwarzania danych, przesyłania plików. Do przeprowadzenia testów wykorzystano narzędzie Gatling. Badania wykazały, że mikroserwisy reaktywne mogą być wydajniejsze w szczególności w przypadku występowania opóźnień w komunikacji z serwisami lub bazą danych. W innym razie jest to zależne od złożoności wykonywanych operacji. Mikroserwisy oparte o paradygmat reaktywny, wykorzystują również mniej pamięci RAM w porównaniu z konwencjonalnymi odpowiednikami.
Słowa kluczowe: mikroserwisy; programowanie reaktywne; programowanie imperatywne

*Corresponding author

Email address: **kacper.mochniej@pollub.edu.pl**

1. Introduction

A microservice is a small application which can be deployed, scaled and tested independently and has single responsibility. It can, for example, read data from a queue, execute small pieces of business logic. Such applications are easy to maintain, so the microservice approach has become very popular in enterprise IT. It was introduced in 2014 by J. Lewis and M. Fowler. As a result, applications began to be divided into smaller, cooperating components [1].

Reactive programming is focused on reacting to changes such as data values or events. It allows to program asynchronous and event-driven use cases much easier, without the need for a deep understanding of low-level computer processes and the need to define the complex interactions of state, particularly across thread and network boundaries. Reactive programming is useful in following scenarios:

- processing user events or signal changes,
- handling latency-bound I/O events,
- handling events pushed to the application [2].

Java-based Spring framework is one of the most popular solutions for microservices development. That's because it contains a lot of functionality that helps developers create both small and large projects. Along with Spring 5, the Spring WebFlux [3] module was released for creating reactive applications. It uses Project Reactor [4] library which is an implementation of Reactive Streams - standard for asynchronous stream processing with non-blocking back pressure adopted in Java 9. Spring WebFlux also provides support for non-servlet containers such as Netty or Undertow.

The purpose of the work is to compare the performance of microservices based on reactive and imperative approaches, considering:

- communication with the database,
- operations on data,
- communication between services.

The following hypotheses have been defined:

1. Reactive microservices are more efficient for data-intensive tasks than conventional ones,

2. Reactive microservices are more efficient for latency-bound operations compared to non-reactive ones,
3. Reactive microservices use less RAM than conventional ones.

2. Review of the literature

The analysis of the literature showed that the topic covered is still fresh, as there are not many works dedicated to it. In addition, the conclusions of the various works are inconsistent, with some showing that reactive applications are more efficient while others don't. The situation is similar for hardware resources - the results of some works show less RAM or CPU usage for a reactive application, while others for a conventional application.

In the work [5] the author analyzed the features and disadvantages of reactive programming compared to conventional programming using a containerized microservices-based online ticket store programmed both in reactive and non-reactive versions. Tests have shown that there is not much difference from the conventional approach, however reactive programming improves the software development process and the stability of software.

Article [6] compares the reactive and conventional approaches in Java Web application development. For this purpose applications were created both in reactive and non-reactive ways using Spring Boot framework. Query processing times, the use of environment resources, and how many queries can be handled correctly was checked. In addition, the lines of code required to create each application were analyzed to compare the time consumption of their implementation. Results show that the reactive application processes queries faster, uses less CPU and is more stable in the case of handling many simultaneous requests, but it's more time-consuming to create than imperative variant.

Work [7] evaluates the possibility of using reactive programming and R2DBC in Java to communicate with a relational database, it has been done by creating two applications with Spring Boot framework: the reactive and non-reactive one, which include appropriate API to connect with the database (R2DBC and JDBC). The database used in work is MySQL. The study shows that R2DBC is good "out of the box" without need to set specific parameters. However it seems to have slower select queries and BLOB's are not handled optimally.

In the work [8] the author compared reactive and non-reactive applications written in Spring Boot and Quarkus. The project aimed to provide information to decide what framework is preferred to use in which cases. Results show that in Spring Boot reactive applications use more hardware resources than in non-reactive ones unlike Quarkus. Also, the overall use of hardware resources is higher in Spring Boot.

In the article [9] authors share the experiences in building and adapting reactive systems to microservices architecture. They rewrote an existing application using microservices architecture to reactive system and compared performance of both variants. Results show that the performance improvement in reactive system is not dramatic, but there is a large increase in throughput.

The article [10] is a review of the state of the art of reactive microservices. The objective is to explore documents concerning reactive microservices, migrate microservice project to reactive variant, share experiences and evaluate project with the studied metrics. Authors chose the Restaurant Management system to migrate to reactive microservices using the Lagom framework. The implemented solution accomplished goals relative to maintainability, scalability, testability and monitorability. However, it was not possible to obtain reliable results on the performance, also some security vulnerabilities were detected.

The purpose of work [11] is to check the effects of reactive programming. Author compared synchronous and reactive Playtech BGT Sports content server written using Spring Boot framework. The results showed that the CPU usage is similar for both solutions. However, when the content provider transmitted data at 5 ms intervals, the reactive system had lower latency and 100% throughput.

3. Research methodology

The subject of the study is to compare performance of two microservice applications - one written in reactive approach and another in conventional approach. The application consists of 3 microservices each one performing a specific function.

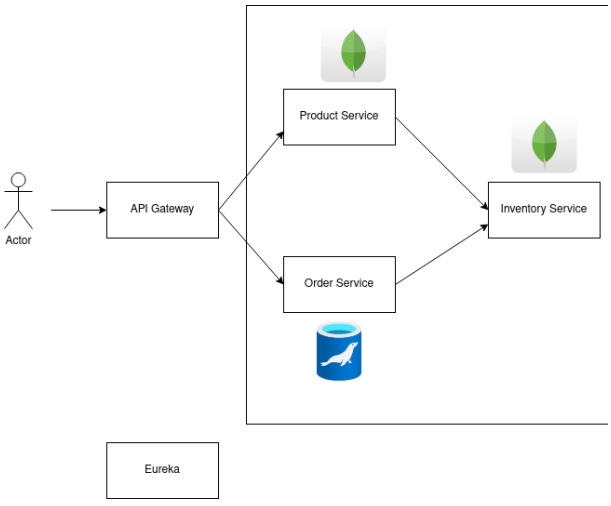


Figure 1: Application architecture.

ProductService is responsible for adding products to the catalog, as well as updating their inventory by connecting to InventoryService. It also allows for exporting, as well as multiple saving using a csv file. It uses a MongoDB database. The previously mentioned InventoryService has access to the inventory of products. It checks at the time of ordering whether the product is in stock, if so, it subtracts the quantity ordered from the current product stock, if not, it returns the product code. It also uses a MongoDB database. OrderService is used to place orders, connects to

InventoryService, uses MariaDB database. Average request processing times for a given scenario, number of instances and number of users were compared. The Gatling [12] tool was used to create and send requests to the applications. The operations that were used to perform the tests are:

- inserting records into the database,
- retrieving records from the database,
- object mapping,
- uploading files to server,
- downloading files from server,
- sending data between services.

This will be done by performing operations such as placing orders, adding, retrieving products or importing/exporting them via csv file. The tests were conducted for each functionality 3 times for both variants of the application with different numbers of microservice instances (1 and 3) and simultaneous requests to the application (100 and 3000). Running tests for different numbers of microservices instances was intended to verify whether the number of instances running and the chosen paradigm are related in terms of performance. The following table presents a detailed description of the scenarios.

Table 1: Test scenarios

Scenario	Description	Number of users
Placing incorrect orders	Sending 55 objects representing invalid orders to OrderService, then transfer to InventoryService , remap and validate, return the order codes to OrderService, and then return to the user.	100, 3000
Placing incorrect orders (single response delayed)	„Placing incorrect orders" scenario with changed logic of order validation - each order is sent individually, a delay of 100ms was added before returning the response from the service.	100
Updating stock	Updating the stock of a product using its id: check if a product with a given exists in the database, if so, change the quantity in stock	100, 3000
Product exports	Retrieving large number of records (271600) from database, map objects to rows, export to csv file	10
Product imports	Uploading the csv file with 18084 rows to the service, remap the row to an object and save it to the database	10
Adding a product	Inserting a single record into the database	100, 3000
Retrieving a product	Retrieving a record from the database	100, 3000
Delay 100 ms	Simulating a delay before returning the response from the ProductService	100, 3000
Barcode generation	Generating a barcode for the product	100, 3000

A platform with the following specifications was used:

- Processor: Intel Core i5 8250U 1.6-3.4GHz,
- RAM: 8GB DDR4,
- Drive: 256GB NVME SSD.

Both microservices and tests were running on the same machine. The environment configuration for running multiple instances was the same as for a single instance.

4. Research results

4.1. Application response times

The results of the tests are statistics of response times to requests including average times.

Table 2: Average response times for conventional application requests

Conventional variant (avg response time [ms])				
Scenario	1 instance		3 instances	
	100 users	3000 users	100 users	3000 users
Placing incorrect orders	537	3711	537	4228
Placing incorrect orders (concurrent internal requests with delay)	6667	-	6649	-
Updating stock	488	3263	522	3263
Adding product	350	2163	351	2319
Fetching product	338	2483	339	2395
Delay 100ms	327	2100	310	2006
Generate barcode for product	342	2907	369	3460
	10 users (working on large datasets)			
Import products	2337		2686	
Export products	15375		16573	

Based on Tables 2 and 3, it can be seen that in most cases the reactive application responds to requests in comparable or worse time than the conventional counterpart. In the test of placing incorrect orders, the reactive application performed much worse. In the case of 100 simultaneous users and 1 instance, the difference is about 46%, for 3 instances the difference is 38%. For 3000 users the conventional application is more than 3 times faster for both 1 and 3 instances. In the second scenario the same functionality is used but it's adjusted to take advantage of the strengths of the non-blocking http client. This time the reactive application was 2 times faster for 100 concurrent users for 1 and 3 instances. "Updating stock" scenario focuses more on exploring performance for inter-service communication alone, without costly stream operations. For 1 instance

results are similar for both 100 and 3,000 users the difference is just over 10% in favor of the conventional version, for 3 instances the difference is greater 13% for 100 users and 23% for 3000 users. In the product addition test, the reactive application achieved better results by being 26% faster for 100 users for 1 instance and 27% for 3 instances and for 3,000 users 10% and 15% respectively. In the product fetching test results were similar, the difference in favor of the reactive version is small and within the limit of measurement error. The latency simulation test showed for 100 users a 21%, and for 3000 users a 12% performance advantage for 1 instance. However for 3 instances results were close. For generating a barcode test for 100 users, the reactive variant was 21% faster for 1 instance, however, for the rest of the cases the results are similar. The last 2 tests were performed for only 10 users, as large datasets were used and it was not possible to perform these tests for more users for performance reasons. Results for products import were very similar, however in the export test the conventional variant was 12% faster for 1 instance and 16% for 3 instances.

Table 3: Average response times for reactive application requests

Reactive variant (avg response time [ms])				
Scenario	1 instance		3 instances	
	100 users	3000 users	100 users	3000 users
Placing incorrect orders	992	11070	867	11077
Placing incorrect orders (concurrent internal requests with delay)	3102	-	3363	-
Updating stock	549	3727	601	4227
Adding product	260	1936	255	1963
Fetching product	322	2321	317	2356
Delay 100ms	259	1845	314	1817
Generate barcode for product	271	2890	338	3266
	10 users (working on large datasets)			
Import products	2311		2582	
Export products	17505		19640	

4.2. RAM usage

The charts in this chapter present the RAM consumption of individual microservices before and after testing for the reactive and conventional versions.

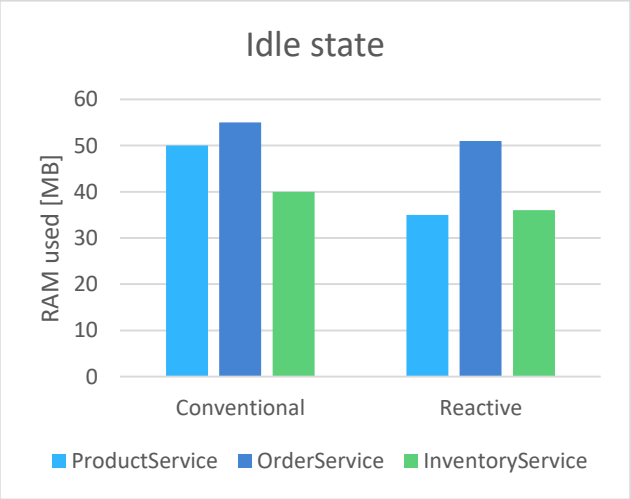


Figure 2: RAM used by microservices in the idle state.

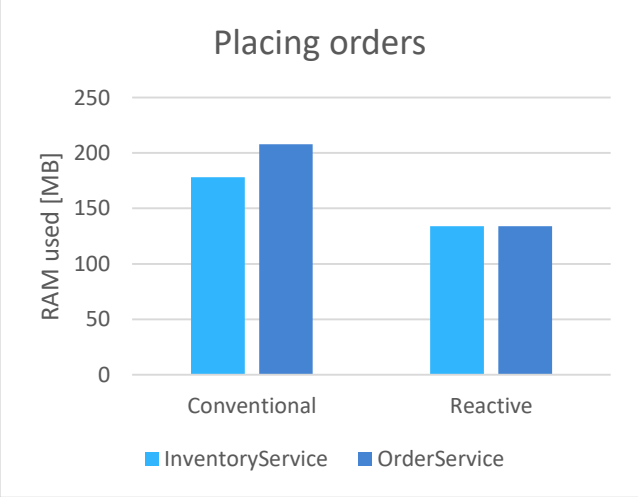


Figure 3: RAM used by microservices in placing incorrect orders scenario.

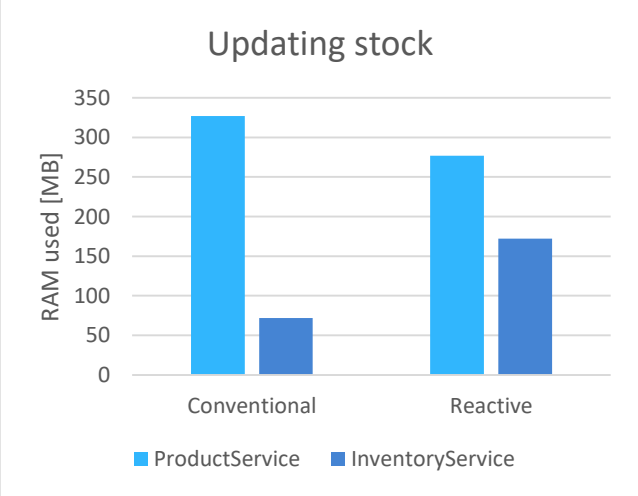


Figure 4: RAM used by microservices in updating stock scenario.

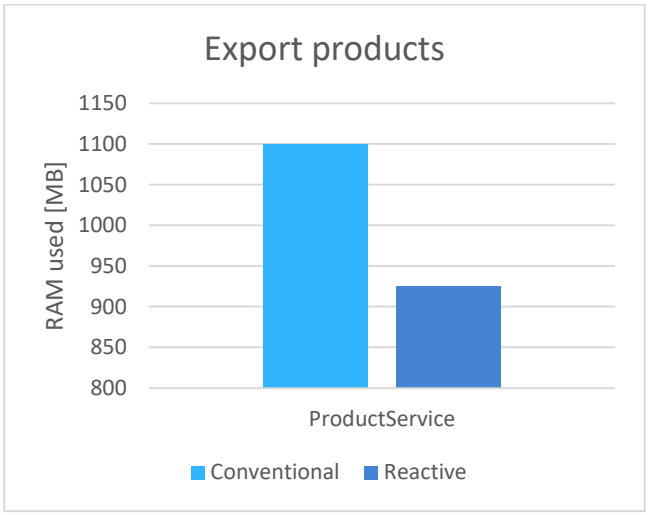


Figure 5: RAM used by microservices in export products scenario.

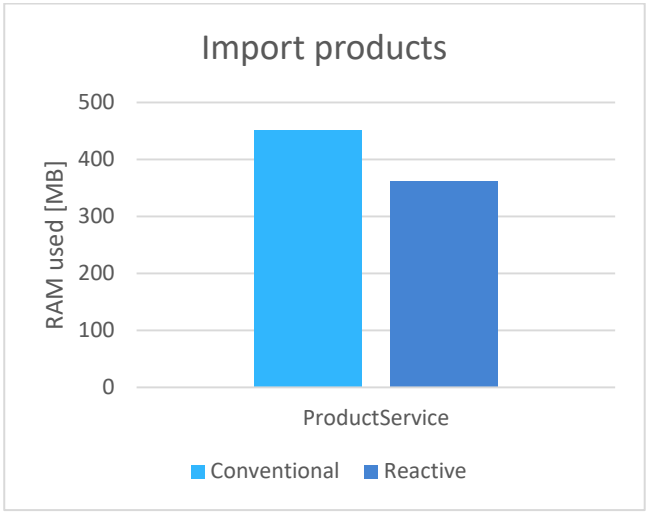


Figure 6: RAM used by microservices in import products scenario.

From the diagrams, you can see that the reactive application used less RAM for most cases. In the idle state reactive ProductService used 30% less RAM, OrderService 7% and InventoryService 10%. After placing incorrect orders reactive OrderService used 36% less RAM and InventoryService 23%. A bit different result is seen with the update stock test: reactive ProductService used 15% less RAM, while InventoryService used 58% more. In the adding product test the reactive variant obtained a better result by 13%. The biggest difference was in the fetching product test, reactive service used 61% less memory. In the generating barcode test, the reactive application used 57% less RAM. For the scenario of import products, the reactive variant was better by 20%. In last test – products export, there was an 16% advantage for the reactive variant.

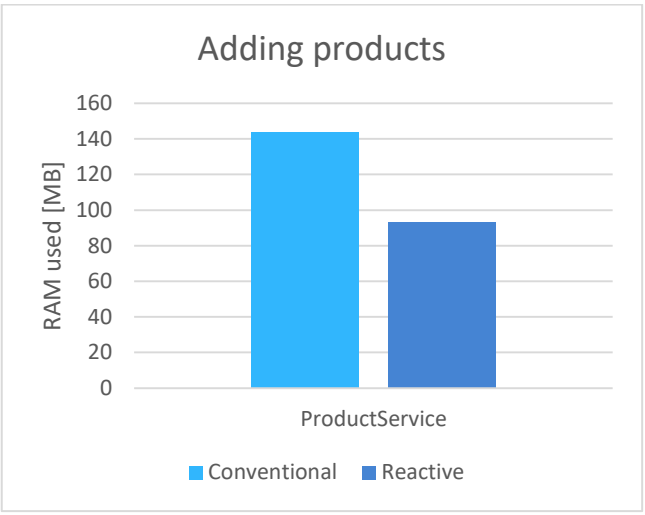


Figure 7: RAM used by microservices in adding products scenario.

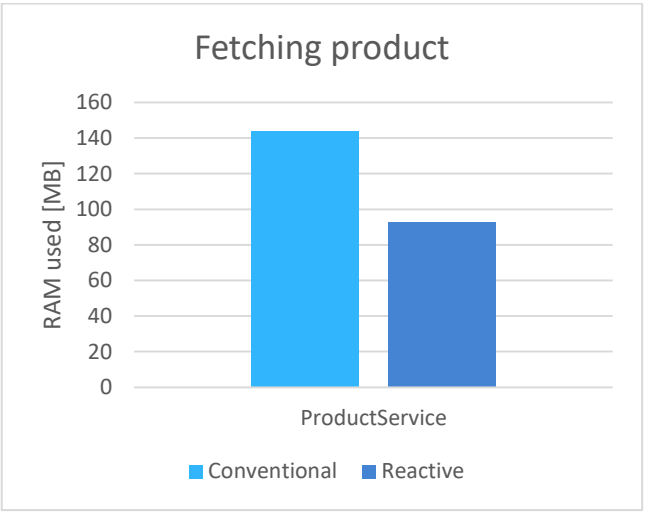


Figure 8: RAM used by microservices in fetching products scenario.

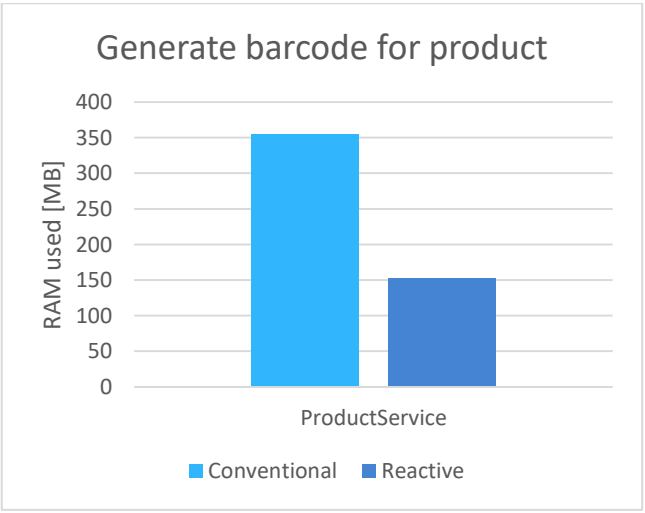


Figure 9: RAM used by microservices in generate barcode for product scenario.

5. Summary and conclusions

Despite the fact that reactive streams process data asynchronously, it wasn't possible to observe a performance advantage for operations on large data sets. Especially for CPU-intensive tasks, the reactive application performed significantly worse than the conventional one. This could mean that operations on reactive streams have higher complexity and take longer than the corresponding imperative code. This would be indicated by the results of the barcode generation test, where the results perform gently better. It is also a CPU-intensive task, but the operations are performed in a blocking style. This is also noticeable when comparing the tests of placing incorrect orders and updating stock, in both tests data is exchanged between services, but in the latter there are far fewer data operations and the results of the reactive version perform much better here. Two tests (adding a single product and I/O operations) showed the advantage of the reactive variant due to the non-blocking nature of I/O operations for this solution. The surprise, however, is that this gain is not apparent for retrieving a record from the database. As a result, it is difficult to say with certainty which approach provides better performance. Based on measurements for individual tests, one can conclude that for typical purposes (communication with other services, returning data) reactive services are less efficient. In reality, however, this depends on many different factors such as the specification of the server, application design, the database used, so for similar scenarios the results under different conditions can be quite different. This is evident by comparing the results obtained from various works. What's more, all the services and databases were running on the same machine, so there were no delays in the connections between them, which reactive application handles better because it doesn't block the thread, but sends another request. This was confirmed in the ordering test, where order data was sent one at a time, and in a test simulating the delay in processing a request. Therefore, it is important that before deciding on a reactive system, careful consideration should be given to whether the choice is appropriate under the circumstances. Performance also depends on the implementation of the reactive paradigm, in this case, the Spring WebFlux framework and the Java language were used, the results would be quite different when using other development tools. Based on the above results, it is difficult to clearly determine whether the number of instances and the chosen approach are related in terms of performance, sometimes the gain was greater for the reactive application, other times vice versa.

In the tests conducted, reactive services used less RAM in most cases. This is made possible by using an event loop model that takes care of calling the corresponding request and response handling functions. It runs in the background and does not block the main thread; instead, it moves on to the next request, and when the first request is ready, it resumes processing. This also greatly reduces the number of threads created by a reactive application.

References

[1] J. Thönes, Microservices, IEEE Software 32(1) (2015) 113-116.

[2] T. Nurkiewicz, B. Christensen, Reactive Programming with RxJava: Creating asynchronous, event based applications, 1st Edition, O'Reilly Media , 2016.

[3] Spring WebFlux Documentation, https://docs.spring.io/spring-framework/reference/web/webflux.html#webflux, [27.05.2023].

[4] Project Reactor webpage, https://projectreactor.io/, [27.05.2023].

[5] P. Dakowitz, Comparing reactive and conventional programming of Java based microservices in containerized environments, Master thesis, Haw Hamburg, 2018.

[6] S. Iwanowski, G. Kozieł, Comparative analysis of reactive and imperative approach in Java Web application development, Journal of Computer Sciences Institute 24 (2022) 242-249.

[7] K. Dahlin, An evaluation of Spring Webflux with focus on built in SQL features, Master thesis, Mid Sweden University, 2020.

[8] A. Nordlund, N.Nordstrom, Reactive vs Non-reactive Java Framework, Bachelor thesis, Mid Sweden University, 2022.

[9] A. Sim, O. Barus, F. Jaya, Lessons Learned In Applying Reactive System In Microservices, Journal of Physics: Conf. Series 1175 (2019) 1-6.

[10] G. Hochbergs, Reactive Programming and its effect on performance and the development process, Master thesis, Lund University, 2017.

[11] J. Ferreira, Reactive Microservices An Experiment, Master thesis, Polytechnic of Porto, 2022.

[12] Gatling webpage, https://gatling.io/ , [27.05.2023]

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 28 (2023) 248–255
Received: 6 June 2023
Accepted: 21 June 2023

Comparative analysis of live sports streaming services

Analiza porównawcza serwisów transmitujących wydarzenia sportowe

Emilia Skiba*

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

Abstract
In the current era of dynamic development of internet television, there is significant competition among providers of sports transmission services. In an effort to stand out, manufacturers introduce new functionalities, often disregarding the limitations associated with customers' access to high-speed internet. The aim of this article is to conduct a comparative analysis of sports event streaming services available in the Polish market, taking into account their performance under various network conditions. A survey was conducted among a specified research group, and a technical evaluation of the internet and mobile applications of three services was carried out. Both approaches revealed that Polsat Box Go is the service that performs best under any network conditions.

Keywords: comparative analysis; live streaming

Streszczenie
W obecnych czasach dynamicznego rozwoju telewizji internetowej, można zaobserwować znaczną konkurencję między dostawcami usług transmisji sportowych. Chcąc się wyróżnić, producenci wprowadzają nowe funkcjonalności, często jednak nie biorąc pod uwagę ograniczeń związanych z dostępem klientów do szybkiego Internetu. Celem niniejszego artykułu jest przeprowadzenie analizy porównawczej serwisów transmitujących wydarzenia sportowe dostępnych na polskim rynku, z uwzględnieniem ich działania w różnych warunkach sieciowych. Przeprowadzone zostało badanie ankietowe wśród określonej grupy badawczej oraz badanie techniczne aplikacji internetowych i mobilnych trzech serwisów. Oba te podejścia wykazały, że serwisem, który najlepiej radzi sobie w każdych warunkach sieciowych, jest Polsat Box Go.

Słowa kluczowe: analiza porównawcza; transmisja na żywo

*Corresponding author
Email address: emilia.skiba@pollub.edu.pl (E. Skiba)

1. Wstęp

Wraz z narastającą popularnością telewizji internetowej, wiele przedsiębiorstw, które dostarczały transmisje wydarzeń sportowych w tradycyjny sposób, zdecydowało się wprowadzić usługę przekazu przez Internet. Ta dynamiczna zmiana stała się niezwykle popularnym rozwiązaniem, stopniowo wypierając tradycyjne metody oglądania sportu. Wchodzące na rynek nowe firmy dostrzegły potencjał tego trendu i koncentrują swoje działania na dostarczaniu transmisji za pośrednictwem przeglądarek internetowych oraz aplikacji mobilnych. Ten ewolucyjny kierunek przynosi wiele korzyści dla konsumentów. Przede wszystkim, transmisje przez Internet umożliwiają dostęp do wydarzeń sportowych w dowolnym miejscu i czasie, co sprawia, że oglądanie sportu staje się bardziej elastyczne i dostosowane do potrzeb widza. Ponadto, możliwość korzystania z różnorodnych platform i urządzeń, takich jak smartfony, tablety czy smart telewizory, znacznie rozszerza dostępność transmisji, co przyciąga szerokie grono odbiorców.

Producenci, zdając sobie sprawę z rosnącego znaczenia internetowej transmisji sportowej, inwestują w rozwój zaawansowanych technologii, aby zapewnić użytkownikom jak najlepsze doświadczenie. W różny sposób przystosowali swoje serwisy do tego, aby niezależnie od jakości połączenia internetowego klienta, ich transmisja działała płynnie. Najpopularniejszym rozwiązaniem okazała się technika ABR (ang. *Adaptive Bit Rate*)[1]. Jest to używana w strumieniowaniu wideo i audio technologia, która dostosowuje jakość strumienia w czasie rzeczywistym, w zależności od bieżących warunków sieciowych i urządzenia odbiorczego. Serwer strumieniowy monitoruje jakość połączenia sieciowego i wydajność odbiorcy i na tej podstawie podejmuje decyzję o wyborze najodpowiedniejszego wariantu strumienia. Może to obejmować zmianę rozdzielczości i formatu kompresji w trakcie odtwarzania, aby zapewnić płynne i nieprzerwane doświadczenie odbiorcy. W ten sposób zapewnione ma zostać dostosowanie jakości transmisji do szybkości łącza internetowego klienta bez zakłóceń.

Producenci chcąc wyróżnić się na rynku, oferują różne nowoczesne funkcje w swoich serwisach. Często przez pogoń za jak najnowocześniejszymi funkcjonalnościami, liderzy wśród dostawców transmisji na żywo zaniedbują fakt, że nie wszyscy użytkownicy mają dostęp do szybkiego Internetu. Wprowadzanie nowych funkcji wymaga solidnego fundamentu, który obejmuje brak zakłóceń oraz odpowiednią jakość, dostosowaną do parametrów połączenia internetowego. Według aktualnych danych, średnia wartość prędkości pobierania dla łącza stacjonarnego w Polsce wynosi 60,9 Mbps, natomiast średnia prędkość łącza mobilnego wynosiła prawie dwukrotnie mniej [2]. Badania wykazały, że realna

prędkość łącza na niektórych obszarach wiejskich notowała nawet minimalne wartości 3,9 Mbps. Oznacza to, że istnieją w Polsce miejsca, w których dostęp do Internetu jest ograniczony, co powinno być uwzględniane podczas tworzenia nowych technologii.

W celu przeprowadzenia analizy porównawczej serwisów transmitujących wydarzenia sportowe na żywo dostępnych na polskim rynku, kluczowym aspektem jest porównanie dostosowania płynności transmisji do niższych średnich wartości prędkości pobierania. Wielu dostawców nie oferuje funkcji zmiany jakości, w zamian oferując wbudowaną opcję dostosowania jej do prędkości łącza, dlatego bezawaryjność działania tej funkcjonalności jest niezwykle istotna. W związku z tym, podczas analizy jakości transmisji, przeprowadzone zostały testy, które pozwoliły na określenie, jak różne serwisy radzą sobie z różnymi warunkami sieciowymi. Było to możliwe dzięki wykorzystaniu specjalnych narzędzi, takich jak oprogramowanie do testowania prędkości łącza internetowego oraz oprogramowanie umożliwiające ograniczanie prędkości łącza internetowego.

Aby lepiej rozpoznać się w tym, na czym najbardziej zależy klientom serwisów udostępniających transmisje na żywo, przeprowadzona została ankieta wśród grupy badawczej. Jej zadaniem było zebranie informacji na temat tego, jakie funkcjonalności i aspekty transmisji są dla jej odbiorców najważniejsze. Ponadto, w przeprowadzonej ankiecie znalazły się pytania związane z niedogodnościami technicznymi, które użytkownicy doświadczają podczas korzystania z usług serwisów, a także pytania dotyczące działania platformy oraz udostępnionych przez producenta aplikacji mobilnych. Podejście to umożliwiło uzyskanie kompleksowej oceny serwisów transmitujących wydarzenia sportowe, uwzględniającej nie tylko aspekt techniczny, ale również zadowolenie użytkowników z ich działania.

2. Przegląd literatury

Na polskim rynku serwisów oferujących transmisję wydarzeń sportowych zaobserwować można dynamiczny rozwój. Wciąż pojawiają się nowe, światowe firmy, które z powodzeniem zdobyły już swoją pozycję na arenie międzynarodowej. Jednakże, warto zauważyć, że w kontekście badań naukowych dotyczących tego zagadnienia, porównania między tymi serwisami nie zostały jeszcze dostatecznie zgłębione. Istnieją artykuły naukowe, które łączą ze sobą tematykę informatyki i sportu. Wiele z nich skupia się głównie na marketingowym podejściu do transmisji sportu, ale istnieją również takie, które proponują nowe podejścia do transmisji.

Dostępne są artykuły naukowe, których rezultaty mogą przyczyniać się do rozwoju transmisji wydarzeń sportowych na żywo. Y. A. Reznik w swojej pracy koncentruje się na analizie zachowania systemów transmisji wideo z uwzględnieniem adaptacji do zmienności przepustowości sieci oraz rozmiarów odtwarzacza [3]. Głównym celem jest opracowanie matematycznych formuł, umożliwiających ocenę średnich parametrów wydajności takich jak średnie zużycie przepustowości czy średnią jakość dostarczana przez system. Opracowane wyrażenia są wykorzystane do badania granic wydajności osiągalnych poprzez adaptacyjną transmisję oraz do sformułowania kilku problemów optymalizacyjnych z nimi związanych. Rezultaty tej pracy mają zastosowanie w dziedzinie transmisji na żywo w sporcie, gdzie adaptacyjna transmisja jest kluczowym aspektem zapewnienia wysokiej jakości dostarczanego strumienia wideo w zmiennych warunkach sieciowych i przy różnych rozmiarach odtwarzaczy.

Y. Li i pozostali autorzy w swojej pracy omówili wyzwania związane z algorytmami adaptacyjnego kodowania przepływności w transmisji wideo, które są szeroko stosowane w usługach transmisji na żywo, takich jak transmisje sportowe [4]. Autorzy proponują szkielet transmisji wideo na żywo oparty na protokole HTTP/2, który umożliwia pomijanie klatek w celu zmniejszenia opóźnień. Przedstawiają również model QoE i formułują problem optymalizacji, dążąc do zapewnienia wysokiej jakości transmisji wideo. Przeprowadzone eksperymenty oceniają proponowaną metodę w kontekście ogólnych algorytmów adaptacyjnego kodowania przepływności. W rezultacie, zaproponowana metoda osiąga porównywalną wydajność przy minimalnych stratach jakości. Ten artykuł może dostarczać wglądu w technologiczne aspekty transmisji na żywo w sporcie, szczególnie w zakresie optymalizacji jakości transmisji w celu zapewnienia jak najlepszego doświadczenia dla użytkowników.

N. Barman i M. G. Martini w swoim artykule poruszyli kwestie jakości dostarczanej przez usługi wideo oraz modelowania jakości doświadczenia użytkownika w kontekście dostarczania transmisji na żywo [5]. Analizowanie jakości dostarczanej przez usługi transmisji wideo jest istotne, ponieważ zapewnia wytyczne dotyczące zapewnienia optymalnego doświadczenia użytkownikowi, zwłaszcza w kontekście transmisji na żywo wydarzeń sportowych.

Zastosowanie technologii w sporcie może wpłynąć na to jak będzie rozwijać się rynek serwisów transmitujących wydarzenia sportowe. W swojej pracy, T. Kumano i pozostali autorzy, zaproponowali rozwój technologii, która tworzy automatyczny komentarz sportowy na podstawie danych z transmisji wydarzeń sportowych [6]. Metoda ta wykorzystuje syntetyczną mowę, aby przekazać obiektywną sytuację w grze. Poprzez przygotowanie szablonów komentarzy dla różnych wydarzeń sportowych, metoda ta może być stosowana w przypadku różnorodnych dyscyplin sportowych. Wyniki oceny subiektywnej wskazują, że dzięki tej technologii odbiorcy mogą lepiej zrozumieć przebieg wydarzeń sportowych. Automatyczny komentarz sportowy może przyczynić się do poprawy jakości transmisji na żywo. Widzowie, którzy mają lepsze zrozumienie gry i sytuacji, mogą bardziej docenić jakość usługi transmisji i być bardziej zadowoleni z doświadczenia. To może przekładać się na lojalność użytkowników i długoterminowy rozwój serwisów transmitujących wydarzenia sportowe.

K. Bilal, A. Erbad i M. Hefeeda w swojej pracy proponują nowy system, nazwany CMVCS (ang. *Cloud-based Multi-View Crowdsourced Streaming*), który zbiera indywidualne strumienie wideo i łączy je w wielowidokowe nagrania, pozwalając widzom oglądać wydarzenie z różnych perspektyw [7]. Optymalna alokacja zasobów i zapewnienie wysokiej jakości doświadczenia widza są kluczowe w transmisji sportu, a opisane w artykule rozwiązania mogą mieć zastosowanie w poprawie jakości i zróżnicowaniu oglądanych perspektyw w trakcie transmisji na żywo.

Udostępnianie transmisji wydarzeń sportowych w Internecie wiąże się z zagrożeniami kradzieży przechwytywania transmisji i udostępniania jej w legalnych źródłach. K. K. Jakkur Patalappa i S. M. Chandramouli w swojej pracy przeanalizowali problem nielegalnych usług transmisji na żywo i ich wpływ na użytkowników. Wskazuje na zagrożenia związane z tymi usługami, takie jak szkodliwe oprogramowanie, śledzenie użytkowników i nieuczciwe praktyki reklamowe. Ponadto, artykuł podkreśla konieczność ochrony widzów przed takimi nielegalnymi usługami i proponuje metody identyfikacji i ścigania stron internetowych, które prowadzą nielegalne transmisje. Artykuł pomaga w budowaniu świadomości na temat zagrożeń związanych z nielegalnymi transmisjami na żywo i zachęcaniu do korzystania z legalnych i bezpiecznych usług transmisyjnych.

3. Cel i zakres pracy

Celem pracy jest przeprowadzenie analizy serwisów transmitujących wydarzenia sportowe na żywo, dostępnych obecnie na polskim rynku. Celem jest wybranie serwisu najbardziej dostosowanego do potrzeb użytkowników oraz najlepiej przystosowanego do działania w każdych warunkach sieciowych. Badanie zostało przeprowadzone poprzez ankietę w gronie użytkowników korzystających z tego typu serwisów oraz poprzez testy jakości i płynności transmisji względem warunków sieci.

Zakres pracy obejmuje przegląd literatury związanej z wybraną tematyką badawczą, wybór obiektów badawczych, wybór metod badania, utworzeniu planu badań, przeprowadzenie badania ankietowego i badania technicznego oraz analizę wyników i wyciągnięcie wniosków.

Na potrzeby badania sformułowano szczegółowe pytania badawcze:

1. Który serwis jest najlepiej oceniany przez jego użytkowników?
2. Który serwis jest najpopularniejszy?
3. Który serwis posiada najlepiej działającą w każdych warunkach sieciowych aplikację mobilną?
4. Który serwis działa najlepiej przez przeglądarkę internetową w każdych warunkach sieciowych?

4. Obiekty badawcze

Kryterium wybrania obiektów do badań była dostępność i popularność na polskim rynku. Każdy serwis posiada różne funkcjonalności, jednak wybrane obiekty badawcze zostały one dobrane w ten sposób, aby można było porównać je w tych samych kategoriach. Do badania wybrane zostały serwisy Canal+ Online, Viaplay oraz Polsat Box Go.

4.1. Serwis Canal+ Online

Pierwszym obiektem badań jest platforma Canal+ Online. Jest to jeden z największych i najpopularniejszych serwisów internetowych, oferujących transmisje wydarzeń sportowych na żywo. Canal+ Online stanowi integralną część grupy Canal+, która od długiego czasu pełni rolę lidera w dziedzinie dystrybucji sportowych transmisji na globalną skalę. Serwis ten umożliwia swoim użytkownikom dostęp do zróżnicowanej gamy wydarzeń sportowych, wykraczając poza podstawowe oferty. Ponadto, platforma ta zapewnia użytkownikom dostęp do różnorodnych funkcji, takich jak przeglądanie archiwum transmisji, równoczesne korzystanie z wielu urządzeń. Producent oferuje użytkownikom w ramach abonamentu korzystanie z aplikacji internetowej oraz aplikacji na urządzenia mobilne z systemem Android lub iOS, komputery z systemem operacyjnym Windows lub Mac OS, oraz na wybrane telewizory Smart TV.

4.2. Serwis Viaplay

Viaplay to jedna z największych i najpopularniejszych platform streamingowych w Europie, oferująca bogaty wybór treści filmowych, serialowych oraz sportowych na żywo. Od lipca 2018 roku właścicielem serwisu jest Viaplay Group (dawniej Nordic Entertainment Group), które od lat jest liderem w dziedzinie rozrywki na całym świecie. Na polski rynek serwis został wprowadzony w 2021 roku. Serwis zapewnia użytkownikom dostęp do szerokiej gamy wydarzeń sportowych na żywo. Ponadto, oferuje wiele funkcjonalności, takich jak dostęp do archiwum transmisji, możliwość oglądania na wielu urządzeniach jednocześnie, dostępność transmisji studyjnych z ekspertami i magazynów sportowych. Producent oferuje aplikacje na wiele urządzeń, w tym komputery, urządzenia mobilne z systemem iOS i Android, telewizory Smart TV oraz konsole do gier.

4.3. Serwis Polsat Box Go

Polsat Box Go jest serwisem internetowym oferującym bogatą ofertę wydarzeń sportowych. Jest częścią grupy Polsat, która od lat jest liderem w branży mediowej w Polsce i oferuje wiele atrakcyjnych treści dla swoich użytkowników. Polsat Box Go oferuje nie tylko możliwość oglądania transmisji na żywo, ale również dostęp do archiwum wydarzeń sportowych oraz programów i magazynów sportowych. Platforma ta umożliwia swoim klientom również korzystanie z funkcjonalności umożliwiające oglądanie na wielu urządzeniach jednocześnie. Producent oferuje aplikację na urządzenia mobilne z systemem Android, iOS i Harmony OS oraz na wybrane telewizory Smart TV i dekodery Polsat Box.

4.4. Porównanie cen abonamentu, ofert serwisów oraz ocen aplikacji mobilnych

W Tabeli 1 przedstawione zostało porównanie cen abonamentu oraz ofert sportowych badanych serwisów.

Tabela 1: Porównanie cen abonamentu, ofert oraz ocen aplikacji mobilnych badanych serwisów (dane z dnia 06.06.2023)

Serwis	Canal+ Online	Viaplay	Polsat Box Go
Regularna miesięczna cena abonamentu	49zł	55zł	50zł
Dostępne transmisje	kanały sportowe, piłka nożna, koszykówka, piłka ręczna, sporty walki, sporty motorowe, tenis, lekkoatletyka, kolarstwo, rugby, sporty zimowe	piłka nożna, sporty motorowe, sporty walki, gale sztuk walki, tenis, baseball, żużel, hokej, piłka ręczna, golf, koszykówka, rugby, rzutki, bilard	kanały sportowe, piłka nożna, siatkówka, skoki narciarskie, sporty walki
Ocena aplikacji mobilnej (Sklep Play)	4,2	3,6	4,8

5. Plan badań

W celu przeprowadzenia analizy, wykorzystano dwie różne metody badawcze. Pierwszą z nich było przeprowadzenie ankiety w celu zebrania danych dotyczących badanych serwisów. Drugą metodą były badania techniczne, które skupiały się na zbadaniu dostosowania jakości i płynności transmisji do jakości połączenia internetowego. Wykorzystanie obu tych metod miało na celu uzyskanie kompleksowej oceny serwisów oraz uzyskanie charakteru interdyscyplinarnego pracy, łącząc w sobie elementy z dziedziny informatyki oraz badań marketingowych.

5.1. Badanie ankietowe

W ramach przygotowania badania ankietowego, pierwszym etapem było wyznaczenie grupy badawczej. Grupa ta została skonkretyzowana na osoby, które oglądają transmisje sportu w Internecie. Następnie przeprowadzono anonimową ankietę, która zawierała pytania dotyczące preferowanego serwisu przez ankietowanych oraz sposobu korzystania z niego. Dodatkowe pytania obejmowały częstotliwość korzystania z serwisu, ocenę działania aplikacji, adekwatność jakości transmisji do jakości połączenia internetowego, ocenę jakości własnego dostępu do Internetu, satysfakcję z kosztu abonamentu serwisu oraz częstotliwość występowania przerw w transmisji.

Istotnym elementem sporządzenia ankiety było przygotowanie pytania, zawierającego listę funkcjonalności oferowanych przez wszystkie badane serwisy transmitujące wydarzenia sportowe. Celem tego pytania było uzyskanie oceny ważności tych funkcjonalności przez ankietowanych. Dzięki temu możliwe było stworzenie ostatecznej listy funkcji, uwzględniającej najważniejsze aspekty z perspektywy użytkowników.

Kolejnym krokiem było powtórzenie pytania, jednak tym razem ankietowani zostali poproszeni o ocenę tych samych funkcjonalności związanych z wybranym przez nich serwisem. Dzięki przeprowadzonym krokom, uzyskano kompleksową ocenę analizowanych serwisów.

5.2. Badanie techniczne

W ramach badania technicznego przeprowadzone zostały testy transmisji 30 wydarzeń sportowych, co składało się na łączną liczbę 10 testów dla każdego z trzech badanych serwisów. Każdy test składał się z 5 transmisji przeprowadzonych na jednym z dwóch urządzeń - komputerze z systemem Windows 11 oraz tablecie z systemem Android w wersji 11, wyposażonym w ekran o rozdzielczości Full HD. Celem tych badań było porównanie jakości oraz płynności transmisji w zależności od różnych prędkości pobierania, przy uwzględnieniu zarówno aplikacji internetowej, jak i mobilnej. W ramach przeprowadzonych testów, uwzględniono zarówno popularne transmisje, które cieszyły się największym zainteresowaniem w ciągu dnia, jak i mniej popularne. Zostały one tak dobrane, aby zapewnić wszechstronne badanie, badając działanie każdego z badanych serwisów w różnych warunkach obciążenia serwerów. Tym samym, testowane wydarzenia stanowiły reprezentację różnych scenariuszy, które pozwoliły ocenić jakość transmisji w różnych warunkach eksploatacji. Następnie, w trakcie trwania każdej transmisji, w określonym momencie po upływie 5 minut, przeprowadzano kontrolowane zmniejszenie prędkości pobierania na badanych urządzeniach. W trakcie trwania każdego okresu transmisji, przy każdym etapie redukcji prędkości, dokładnie monitorowano i rejestrowano różne parametry, takie jak jakość automatyczna transmisji, opóźnienie transmisji względem wydarzenia na żywo, łączna liczba zawieszeń transmisji oraz opóźnienie przy najwyższej jakości transmisji.

W celu monitorowania prędkości pobierania na obu urządzeniach, skorzystano z trzech dedykowanych do tego technologii. Pierwszą z nich była aplikacja Speed Test [9], która umożliwiła ciągłe monitorowanie prędkości pobierania w ramach jednego badania. Program ten został użyty zarówno podczas testów na laptopie, jak i na tablecie. W celu ograniczenia prędkości pobierania wykorzystane zostały dwie aplikacje. Na laptopie wykorzystano aplikację NetLimiter [10], która umożliwia ustawienie maksymalnej przepustowości dla dowolnej aplikacji używanej na urządzeniu oraz udostępnia wykres natężenia ruchu sieciowego w czasie rzeczywistym. Na tablecie wykorzystana została aplikacja Throttly [11], dostępna w dedykowanym dla urządzeń z systemem Android sklepie. Aplikacja umożliwia symulowanie różnych warunków sieciowe, oferując ograniczanie przepustowości do wybranych wartości.

6. Wyniki

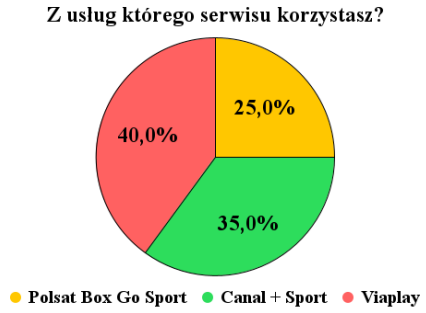
Po przeprowadzeniu badań eksperymentalnych, przystąpiono do szczegółowej analizy uzyskanych wyników. Ze względu na rozległość i kompleksowość przeprowa-

dzonych eksperymentów, przedstawione zostały jedynie wybrane, lecz reprezentatywne rezultaty.

6.1. Wyniki badania ankietowego

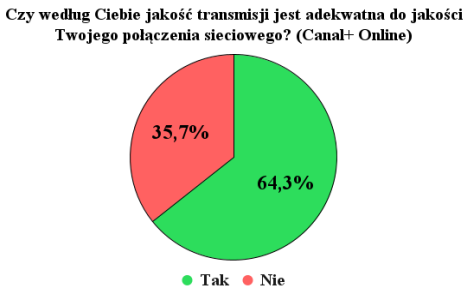
W ramach przeprowadzonej ankiety zebrano łącznie 40 odpowiedzi, które zostały poddane szczegółowej analizie. W celu analizy uzyskanych odpowiedzi, utworzone zostały wykresy, które wspomagają interpretację wyników.

Wyniki pokazały, że największą popularnością cieszy się serwis Viaplay, jednak wartości procentowe są mocno do siebie zbliżone. Zostało to przedstawione na Rysunku 2.



Rysunek 2: Wykres przedstawiający popularność serwisów transmitujących wydarzenia sportowe.

Głównym celem pracy było określenie, czy jakość transmisji jest adekwatna do jakości połączenia internetowego, dlatego ważnym wynikiem uzyskanym podczas badania są wykresy, które przedstawione zostały na Rysunkach 3-5.



Rysunek 3: Wykres przedstawiający zadowolenie klientów serwisu Canal+ Online z jakości transmisji.

Rysunek 4: Wykres przedstawiający zadowolenie klientów serwisu Viaplay z jakości transmisji.

Rysunek 5: Wykres przedstawiający zadowolenie klientów serwisu Polsat Box Go z jakości transmisji.

Użytkownicy serwisu Canal+ Online w większości wykazują zadowolenie z jakości transmisji w stosunku do jakości swojego połączenia sieciowego. Klienci Viaplay wyrażają surowsze opinie na temat tej funkcjonalności. Według połowy z nich, jakość transmisji nie spełnia ich oczekiwań. Z kolei osoby korzystające z usług serwisu Polsat Box Go zgadzają się, że jakość transmisji dostarczanych przez tego dostawcę jest w pełni adekwatna do jakości ich połączenia sieciowego.

W wyniku ankiety, przeprowadzonej w celu ustalenia priorytetowych funkcjonalności serwisów, udało się uszeregować funkcjonalności, które były dla respondentów najważniejsze. Taka metoda umożliwia dokładniejsze zrozumienie, które funkcjonalności są kluczowe, a które mają mniejsze znaczenie dla ogólnej oceny. Tabela 2 zawiera wyniki wraz z przypisanymi do funkcji wagami, co pozwala na lepsze zrozumienie preferencji użytkowników i ich znaczenia.

Tabela 2: Funkcjonalności oraz wyznaczone dla nich wagi

Nr.	Funkcjonalność	Waga
1	Wysoka jakość transmisji na żywo	9
2	Dostosowanie jakości transmisji do jakości połączenia sieciowego	8
3	Dostępność transmisji na wielu urządzeniach (telewizor, tablet, smartfon itp.)	7
4	Brak opóźnień	6
5	Atrakcyjna cena abonamentu	5
6	Duży wybór dyscyplin sportowych	4
7	Wysokiej jakości komentarz	3
8	Możliwość dostępu do archiwum wydarzeń sportowych	2
9	Dodatkowe atrakcje, takie jak studio z ekspertami	1

Korzystając z uzyskanych wag jako mnożników punktacji, istnieje możliwość finalnej oceny serwisów z uwzględnieniem stopnia ważności funkcji. Średnie oceny serwisów uzyskane przez ankietę, zostały przedstawione w Tabeli 3.

Tabela 3: Średnie oceny funkcjonalności serwisów

Nr. funkcjonalności	Wynik ± SD		
	Canal+ Online	Viaplay	Polsat Box Go
1	3,4 ± 1,3	3,1 ± 1,1	4,3 ± 0,8
2	3,1 ± 1,3	3,1 ± 1,4	4,3 ± 0,8
3	3,5 ± 1,3	4,3 ± 0,9	4,5 ± 0,8
4	3,2 ± 0,9	3,7 ± 1,4	4,2 ± 0,8
5	2,8 ± 1,1	2,4 ± 1,0	3,4 ± 0,7
6	3,4 ± 0,6	3,6 ± 1,3	3,6 ± 0,8
7	3,8 ± 0,7	3,4 ± 1,2	3,7 ± 0,9
8	3,3 ± 1,1	3,4 ± 1,4	3,4 ± 0,8
9	3,6 ± 0,5	3,3 ± 1,2	3,5 ± 0,7

Po przypisaniu punktów (3 za pierwsze miejsce w kategorii, 2 za drugie miejsce i 1 za trzecie miejsce) oraz uwzględnieniu wag dla poszczególnych kategorii, utworzona została ostateczna Tabela 4, która przedstawia miejsca przyznane badanym serwisom w wyniku przeprowadzonego badania ankietowego.

Tabela 4: Wyniki badania ankietowego

Serwis	Canal+ Online	Viaplay	Polsat Box Go
Punkty	75	78	131
Miejsce	3	2	1

W rezultacie przeprowadzonego badania ankietowego udało się wyłonić bezsprzecznie najlepszy serwis, który zdecydowanie przewyższał pozostałe dwie badane platformy. Ankietowani wybrali Polsat Box Go jako serwis najlepiej dostosowany do transmisji wydarzeń sportowych.

6.2. Wyniki badania technicznego

Badanie techniczne zostało przeprowadzone dla sześciu różnych prędkości pobierania, które wynosiły odpowiednio: 180 Mbps, 30 Mbps, 15 Mbps, 10 Mbps, 5 Mbps oraz 1 Mbps. W celu ułatwienia interpretacji danych, dokonano numerycznej reprezentacji wartości jakości transmisji. Skala oceny jakości została ustalona na pięć stopni, gdzie 5 oznacza najwyższą jakość Full HD, 4 - jakość wysoką HD, 3 -jakość średnią, 2 - jakość niską, a 1 - jakość najniższą.

Badania na dwóch urządzeniach zostały rozdzielone z powodu braku funkcjonalności zmiany jakości dla aplikacji mobilnych. W Tabeli 5 przedstawione zostały obliczone średnie wyniki badania aplikacji internetowej dla trzech badanych serwisów w zależności od prędkości pobierania.

Tabela 5: Średnie wyniki badania technicznego wykonanego na aplikacjach internetowych badanych serwisów

Kategoria	Prędkość pobierania [Mbps]	Canal+ Online	Viaplay	Polsat Box Go
Średnia jakość automatyczna transmisji	180	5	4	5
	30	4	4	4
	15	3	4	4
	10	3	4	4
	5	3	3	3
	1	2	2	2
	Średnia ± SD	3,3 ± 1,0	3,5 ± 0,8	3,7 ± 1,0
Średnie opóźnienie [s]	180	25	17	23
	30	25	16	24
	15	24	19	24
	10	26	18	25
	5	26	20	28
	1	39	24	30
	Średnia ± SD	27,5 ± 5,7	19 ± 2,8	25,7 ± 2,7
Średnie opóźnienie przy najwyższej jakości transmisji względem transmisji przy najwyższej jakości i najszybszej prędkości pobierania [s]	180	Brak funkcji zmiany jakości	-	-
	30		2	1
	15		4	1
	10		4	3
	5		12	7
	1		-	-
	Średnia ± SD	-	5,5 ± 4,4	3 ± 2,8
Łączna liczba zawieszeń transmisji	180	1	0	0
	30	0	0	0
	15	0	1	2
	10	0	0	0
	5	0	4	2
	1	1	0	0
	Łącznie	2	5	4

Uzyskane z ankiety wagi funkcjonalności zostały wykorzystane do oceny serwisów w analizie technicznej. Wagi kategorii zostały odpowiednio przeliczone na badane funkcjonalności w kontekście badania technicznego. Łączna liczba zawieszeń transmisji została zinterpretowana jako odpowiednik kategorii dostosowania jakości transmisji do stabilności połączenia internetowego, ponieważ na tę wartość ma wpływ wybór automatycznej jakości, który odzwierciedla płynność transmisji. W Tabeli 6 znajdują się finalne wyniki badania technicznego aplikacji internetowych badanych serwisów.

Tabela 6: Finalne wyniki badania technicznego aplikacji internetowych badanych serwisów i przypisane im miejsca

Kategoria	**Waga**	**Canal+ Online**	**Viaplay**	**Polsat Box Go**
Średnia jakość automatyczna transmisji	3	1	2	3
Średnie opóźnienie [s]	1	1	3	2
Średnie opóźnienie przy najwyższej jakości transmisji względem transmisji przy najwyższej jakości i najszybszej prędkości pobierania [s]	1	1	2	3
Łączna liczba zawieszeń	2	3	1	2
Suma		6	7	18
Miejsce		3	2	1

Analogicznie wykonane zostało badanie techniczne aplikacji mobilnych dedykowanych dla systemu Android. Średnie wyniki badania w zależności od prędkości pobierania, zostały umieszczone w Tabeli 7.

Tabela 7: Średnie wyniki badania technicznego wykonanego na aplikacjach mobilnych badanych serwisów na system Android

	Prędkość pobierania [Mbps]	**Canal+ Online**	**Viaplay**	**Polsat Box Go**
Średnia jakość automatyczna transmisji	180	3	4	4
	30	3	4	4
	15	2	4	4
	10	2	3	3
	5	2	3	3
	1	1	2	2
	Średnia ± SD	2,2 ± 0,8	3,3 ± 0,8	3,3 ± 0,8
Średnie opóźnienie [s]	180	54	6	24
	30	55	7	26
	15	55	8	27
	10	54	7	25
	5	54	7	26
	1	58	13	30
	Średnia ± SD	55,0 ± 1,5	8,0 ± 2,5	26,3 ± 2,1
Łączna liczba zawieszeń transmisji	180	0	0	0
	30	1	0	0
	15	1	0	2
	10	0	0	0
	5	5	1	0
	1	2	3	0
	Łącznie	9	4	2

Również w tym przypadku wykorzystane zostały uzyskane z ankiety wagi funkcjonalności przeliczone na badane funkcjonalności w kontekście badania technicznego. W Tabeli 8 znajdują się finalne wyniki badania technicznego aplikacji mobilnych badanych serwisów, dedykowanych na system Android.

Tabela 8: Średnie wyniki badania technicznego wykonanego na aplikacjach mobilnych badanych serwisów na system Android

Kategoria	**Waga**	**Canal+ Online**	**Viaplay**	**Polsat Box Go**
Średnia jakość automatyczna transmisji	3	1	3	3
Średnie opóźnienie	1	1	3	2
Łączna liczba zawieszeń	2	1	2	3
Suma		6	16	17
Miejsce		3	2	1

W wyniku przeprowadzonych badań udało się zidentyfikować serwis, który osiągnął najlepsze wyniki we wszystkich kategoriach. Można jednoznacznie stwierdzić, że Polsat Box Go jest najlepiej dostosowanym serwisem do udostępniania transmisji wydarzeń sportowych. Zarówno w badaniu ankietowym jak

i w badaniach technicznych, najlepiej przystosował do działania potrzebne funkcjonalności. Punktacja każdego serwisu została przedstawiona w Tabeli 9.

Tabela 9: Połączone wyniki obu badań

Kategoria	Canal+ Online	Viaplay	Polsat Box Go
Miejsce, które uzyskał serwis w badaniu ankietowe	3	2	1
Miejsce, które uzyskał serwis w badaniu technicznym aplikacji internetowej	3	2	1
Miejsce, które uzyskał serwis w badaniu technicznym aplikacji mobilnej na system Android	3	2	1
Miejsce w ogólnym rankingu	3	2	1

7. Wnioski

Analiza wyników pozwoliła na stwierdzenie, że serwis Polsat Box Go osiągnął najlepsze rezultaty zarówno w badaniu ankietowym, jak i w badaniu technicznym. Warto zauważyć, że serwis ten jest polskim przedstawicielem na rynku, który charakteryzuje się silną konkurencją i ciągłym pojawianiem się nowych zagranicznych serwisów. Mimo to, Polsat Box Go odniósł największy sukces zarówno wśród respondentów biorących udział w badaniu ankietowym, jak i wśród analizowanych parametrów technicznych.

Przeprowadzone badania dostarczyły odpowiedzi na wcześniej postawione pytania badawcze:

1. Który serwis jest najlepiej oceniany przez jego użytkowników? - Na to pytanie odpowiedź jest jednoznaczna, ponieważ serwis Polsat Box Go uzyskał zdecydowanie najwięcej punktów w przeprowadzonym badaniu ankietowym.
2. Który serwis jest najpopularniejszy? - Zgodnie z danymi uzyskanymi w badaniu ankietowym, wynika, że serwis Viaplay otrzymał największą liczbę odpowiedzi, co czyni go najczęściej wybieranym serwisem do transmisji wydarzeń sportowych na polskim rynku. Jest to związane z brakiem odpowiednika telewizyjnego serwisu oraz monopolem na transmisję wielu sportów.
3. Który serwis posiada najlepiej działającą w każdych warunkach sieciowych aplikację mobilną? - W wyniku badania technicznego zostało wyłonione, że Polsat Box Go posiada najlepszą aplikację mobilną.
4. Który serwis działa najlepiej przez przeglądarkę internetową w każdych warunkach sieciowych? – Najlepiej działającym w każdych warunkach sieciowych przez przeglądarkę internetową serwisem okazał się Polsat Box Go.

Po przeprowadzeniu badań i analizie ich wyników, autor pracy dostrzega potencjalne obszary rozwoju dalszych badań. Aby dokonać bardziej dogłębnej analizy polskiego rynku dostawców internetowych transmisji sportowych, warto rozważyć badanie wszystkich wersji dostarczanych przez producentów aplikacji na wszystkie systemy oraz analizę interfejsów każdej z nich. Pozwoliłoby to wskazać również serwis z najlepiej przystosowanym interfejsem.

Literatura

[1] A. T. Tran, N. N. Dao, S. Cho, Bitrate Adaptation for Video Streaming Services in Edge Caching Systems, IEEE Access 8 (2020) 135844-135852.

[2] K. Janc, W. Jurkowski, Spatial differentiation of the Internet quality in terms of digital divide in Poland, Transport Geography Papers of Polish Geographical Society 25 (2022) 73-84.

[3] Y. A. Reznik, Average Performance of Adaptive Streaming, Data Compression Conference, IEEE (2021) 263-272.

[4] Y. Li, S. Wang, X. Zhang, C. Zhou, S. Ma, High Efficiency Live Video Streaming With Frame Dropping, IEEE International Conference on Image Processing (2020) 1226-1230.

[5] N. Barman, M. G. Martini, QoE Modeling for HTTP Adaptive Video Streaming - A Survey and Open Challenges, IEEE Access 7 (2019) 30831-30859.

[6] T. Kumano, M. Ichiki, K. Kurihara, H. Kaneko, T. Komori, T. Shimizu, N. Seiyama, A. Imai, H. Sumiyoshi, T. Takagi, Generation of Automated Sports Commentary from Live Sports Data, IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (2019) 1-4.

[7] K. Bilal, A. Erbad, M. Hefeeda, Crowdsourced Multi-View Live Video Streaming using Cloud Computing, IEEE Access 5 (2017) 12635-12647.

[8] K. K. Jakkur Patalappa, S. M. Chandramouli, Exploring Ecosystem of Free Illegal Live Streaming Services and Its Price on Legitimate Services, IEEE International Conference on Mobile Networks and Wireless Communications (2021) 1-8.

[9] Aplikacja do testowania prędkości Internetu, https://www.speedtest.pl/, [29.05.2023].

[10] Aplikacja do ograniczania prędkości Internetu na komputery, https://www.netlimiter.com/, [29.05.2023].

[11] Aplikacja do ograniczania prędkości Internetu na urządzenia mobilne, https://apkpure.com/throttly-network-tools/me.twocities.throttly, [29.05.2023].

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 28 (2023) 256–263
Received: 8 June 2023
Accepted: 29 June 2023

Comparative analysis of Angular and React frameworks

Analiza porównawcza szkieletów programistycznych Angular i React

Sylwester Skrzypiec*, Małgorzata Plechawska-Wójcik

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

Abstract
This thesis aimed to examine the Angular framework and React library in terms of performance on three popular web browsers. To achieve this, two similar test applications were implemented to create user interfaces, along with a server application for database communication and the aforementioned programming frameworks. Subsequently, tests were conducted to assess the time required for downloading and displaying images, records, and their deletion. In addition to performance results, a literature review was conducted to provide a better understanding of the current state of knowledge in the field, and individual features distinguishing the respective framework or library were described. Furthermore, the obtained data was analyzed for statistical differences using statistical tests.

Keywords: Angular; React; performance; SPA; benchmarking

Streszczenie
Niniejsza praca dyplomowa miała na celu sprawdzenie szkieletu Angular oraz biblioteki React pod kątem wydajności na trzech popularnych przeglądarkach internetowych. W związku z tym zaimplementowano dwie podobne aplikacje testowe tworzące interfejsy użytkownika oraz jedną aplikację serwerową do komunikacji z bazą danych i wymienionymi szkieletami programistycznymi. Następnie zostały wykonane testy sprawdzające czas pobierania i wyświetlania danych oraz ich usuwania. Poza wynikami wydajności przeprowadzono przegląd literatury w celu lepszego zobrazowania obecnego stanu wiedzy w danym temacie oraz opisano poszczególne elementy wyróżniające dany szkielet lub bibliotekę. Sprawdzono również otrzymane dane wynikowe pod względem różnic statystycznych korzystając z testów statystycznych.

Słowa kluczowe: Angular; React; wydajność; SPA; analiza porównawcza

*Corresponding author
Email address: sylwester.skrzypiec@pollub.edu.pl (S. Skrzypiec)

1. Wstęp

W obecnych czasach tworzenie nowoczesnych systemów informatycznych ciągle stawia przed programistami nowe wyzwania, żeby sprostać rosnącym oczekiwaniom użytkowników. Żeby zaoszczędzić ich czas oraz ułatwić sobie pracę, w obecnych rozwiązaniach stosuje się architekturę SPA (*ang. Single Page Application*), która wypierając poprzednią tzn. MPA (*ang. Multi Page Application*) pozwoliła na wdrożenie nowszych, znacznie skuteczniejszych rozwiązań [1, 2, 3]. W odróżnieniu od poprzedniej, SPA pobiera wszelkie zasoby tylko przy pierwszym wejściu na daną stronę, a następnie reagując na działania użytkownika, podmienia elementy DOM (*ang. Document Object Model*) korzystając z javascript'u. [4, 5].

Język javascript pomimo swojej wszechstronności oraz popularności [6] razem z CSS (*ang. Cascading Style Sheets*) oraz HTML (ang. *HyperText Markup Language*) coraz rzadziej występuje w sposób natywny w implementacji nowoczesnych aplikacji internetowych i coraz częściej jest zastępowany przez swój nadzbiór, czyli typescript [7]. Plik typescript'owy korzysta z lekko zmienionej składni oraz są na niego nałożone dodatkowe reguły syntaktyczne, a w momencie kompilacji jest tłumaczony na język javascript, dzięki temu przeglądarki nie mają problemu z jego zrozumieniem [8]. Rozszerzenie takie pozwoliło wyeliminować niedoskonałości w postaci braku kontroli nad typami zmiennych oraz dołożyło nowe możliwości w asynchronicznej komunikacji z serwerem. Programiści oprócz wspomnianej zmiany, zaczęli również korzystać z tak zwanych szkieletów programistycznych. Obecnie topowe miejsca zajmują dwa z nich: Angular oraz React [9].

Angular jest szkieletem rozwijanym przez firmę Google oferując deweloperowi wszelkie narzędzia do tworzenia aplikacji internetowych [10]. Używa on swoich rozwiązań w postaci wstrzykiwania zależności, bibloteki „Material" do oprawy graficznej interfejsu oraz zapewnia czysty oraz testowalny kod. Poza tym umożliwia sprawne zarządzanie dostępem do poszczególnych stron, dzięki użyciu komponentów jest możliwe używanie tych samych szablonów HTML w wielu miejscach, przez co same aplikacje są bardziej otwarte na rozbudowę oraz pozwalają na bezkonfliktowe zmiany.

React jest rozwijany przez firmę Facebook oraz w porównaniu do Angular'a jest bibloteką javascript'u a nie szkieletem programistycznym [11]. Jego składnia opiera się na JSX *(ang. Javascript Extensible Markup Language*), które podobnie jak typescript jest rozszerzeniem javascript'u [12]. Takie rozwiązanie oferuje przyspieszoną kompilację kodu oraz czyni kod bardziej czytelnym i łatwiejszym do utrzymania [13].

W związku z szeroką ofertą dostępnych narzędzi w omówionych wyżej rozwiązaniach, zdecydowano się porównać oba szkielety pod kątem wydajności w pobie-

raniu oraz wyświetlaniu danych na przeglądarce Google Chrome.

2. Przegląd literatury

W celu lepszego zrozumienia badanego problemu, przed przystąpieniem do badań został wykonany przegląd literatury nawiązującej do opracowanego tematu.

Pierwszym analizowanym tekstem została praca dyplomowa pt. „Evaluation of Front-end JavaScript Frameworks for Master Data Management Application Development" [14]. Dotyczy ona porównania szkieletów programistycznych do tworzenia interfejsu użytkownika w aspekcie zarządzania danymi podstawowymi (*ang. Master Data Management, MDM*). Autor analizuje wszelkie możliwości szkieletów, architektury, dane statystyczne oraz bada je pod kątem szybkości wykonywania operacji typu CRUD (*ang. create, read, update, delete, tłum. utwórz, odczytaj, aktualizuj, usuń*). Analizując wyniki badań oraz wnioski pracy odkryto, że najlepszym wyborem do stworzenia aplikacji typu MDM jest Angular ze względu na przewagę nad React i Vue w kwestiach popularności, szybkości kompilacji czy możliwości dostosowywania języka interfejsu graficznego do użytkownika.

Inną przydatną pozycją literaturową okazała się praca dyplomowa pt. „Development of an evaluation model for client-side JavaScript Frameworks" [15]. Autor skupia się na przedstawieniu kryteriów oraz oceny poszczególnych szkieletów, żeby wspomóc programistów w ich wyborze. Do porównania przyjęto trzy najpopularniejsze z nich tzn. Angular, React oraz Vue. W kwestii kryteriów oceny przyjęto wszelkie istotne kwestie związane z wytwarzaniem oprogramowania, czyli łatwością implementacji wymagań funkcjonalnych na przykład możliwość wykorzystywania komponentów do budowy interfejsu użytkownika. Do innych ważnych czynników zaliczono wymagania niefunkcjonalne jak na przykład kwestia bezpieczeństwa, stabilności lub wydajności. Poza przeprowadzonymi badaniami, autor opisał również archtiektury szkieletów oraz opisał możliwości poszczególnych szkieletów. Analizując przedstawione wnioski ustalono, że pierwsze miejsce zajeli React oraz Vue z wynikiem 23/28 punktów. Angular otrzymał 22/28, więc można stwierdzić, że wszystkie trzy możliwości z pewnością nadają się do implementacji interfejsu użytkownika i pomimo występujących różnic w dostępnych narzędziach, poradzą sobie z wszelkimi wymaganiami.

Autorzy pracy dyplomowej pt. „JavaScript Frameworks A qualitative evaluation and comparison of the dominant factors in Angular and React" [16] skoncentrowali się na udzieleniu pomocy początkującym programistom w wyborze szkieletu programistycznego między Angular'em a React'em, odpowiedniego do ich potrzeb. Na początku, zostały opisane poszczególne narzędzia dostępne w szkieletach oraz sam bazowy język Javascript. Następnie poprzez zastosowanie cykli kolba, jako modelu uczenia się przez doświadczenie, poddano ocenie oba szkielety. Uczenie podzielono na cztery okresy sumujące się do czasu 14 dni, gdzie w każdym okresie opisano trudności występujące podczas nauki oraz ogólne subiektywne wnioski towarzyszące implementacji. Analizując punkty przyznane w kategoriach uczenia, najlepszy okazał się Angular.

3. Cel i zakres badań

Celem prowadzonych badań było porównanie wydajnościowe szkieletów programistycznych Angular oraz React. Zakres badań obejmował analizę czasową przetwarzania danych pobranych z serwera w celu potwierdzenia lub negacji tezy oraz odpowiedzi na pytania badawcze. Opracowano następującą tezę:

Szkielet programistyczny Angular jest szybszy w przetwarzaniu danych względem bibloteki React.
Pytania badawcze:

Q1: Czy szkielet Angular szybciej pobierze i wyświetli odpowiednio 1 000, 5 000, 10 000 rekordów niż React?

Q2: Czy React szybciej usunie odpowiednio 100, 500, 1 000 rekordów niż Angular?

Q3: Czy Angular szybciej pobierze i wyświetli 50, 100, 200 zdjęć w formacie .JPG?

4. Plan badań

Przed przystąpieniem do realizacji badań zostały zaimplementowane dwie aplikacje internetowe wykonane w szkieletach Angular i React oraz jedna aplikacja serwerowa obsługująca REST API (*ang. Representational state transfer application programming interface*) do obsługi żądań związanych z pobraniem i wysyłaniem danych.

Testy wydajnościowe opisane poniżej w celu lepszego sprawdzenia wydajności szkieletów, zostały wykonane na przeglądarkach: Google Chrome, Mozilla Firefox oraz Microsoft Edge.
Eksperyment zakładał następujące etapy badań:

1. Wykonanie badań:
   a. wykonanie dwudziestu pomiarów pobrania 1 000, 5 000, 10 000 rekordów z użyciem Angular'a na każdej z trzech przeglądarek,
   b. wykonanie dwudziestu pobrań 1 000, 5 000, 10 000 rekordów z użyciem React'a na każdej z trzech przeglądarek,
   c. wykonanie dwudziestu pomiarów usunięcia 100, 500, 1 000 rekordów przy użyciu Angular'a na każdej z trzech przeglądarek,
   d. wykonanie dwudziestu pomiarów usunięcia 100, 500, 1 000 rekordów z użyciem React'a na każdej z trzech przeglądarek,
   e. wykonanie dwudziestu pobrań i wyświetleń 50, 100, 200 zdjęć z użyciem Angular'a na każdej z trzech przeglądarek,
   f. wykonanie dwudziestu pobrań i wyświetleń 50, 100, 200 zdjęć z użyciem React'a na każdej z trzech przeglądarek.
2. Analiza popularności oraz dostępu do materiałów obu szkieletów:
   a. sprawdzenie liczby dostępnych materiałów bazując na słowach kluczowych związanych z danym szkieletem,

   b. analiza rankingu popularności według społeczności oraz dostępnych ankiet.
3. Analiza wyników badań
   a. przedstawienie danych w formie wykresów, tabel,
   b. wykonanie testów statystycznych i analiza ich wyników.
4. Wnioski

Specyfikacja urządzenia wykorzystanego do przeprowadzenia testów wydajnościowych została podana w Tabeli numer 1.

Tabela 1: Specyfikacja urządzenia

Komponent	Specyfikacja
System operacyjny	Windows 10 Home Wersja 22H2 x64
Procesor	Intel Core i7-7700HQ CPU 2.80 GHz
Pamięć RAM	DDR4 2400MHz 16GB
Karta graficzna	GeForce MX150, GDDR5 2 GB
Dysk	HDD SATA 5400 obr. 1TB

5. Opis wykorzystanych szkieletów

W tworzeniu aplikacji internetowych często używa się szkieletów programistycznych (*ang. frameworks*). Dzięki nim praca jest szybsza, ponieważ udostępniają gotowe narzędzia i komponenty. Szkielety pozwalają na stylizację elementów, walidację formularzy oraz manipulację elementami DOM. Przy tworzeniu aplikacji bez szkieletów programiści muszą samodzielnie zadbać o skalowalność i bezpieczeństwo systemu. Szkielety programistyczne pomagają w ochronie przed atakami takimi jak XSS, CSRF i SQL Injection, co ułatwia wprowadzanie zabezpieczeń do aplikacji [17, 18, 19].

Ogólne różnice pomiędzy badanymi szkieletami zostały przedstawione w Tabeli numer 2.

Tabela 2: Wykaz różnic pomiędzy badanymi szkieletami

Różnica	Angular	React
Firma utrzymująca	Google	Facebook
Język oraz rozszerzenia	Typescript	Javascript/Typescript/JSX
Wiązanie danych	Jedno lub dwukierunkowe	Jednokierunkowe
Typ szkieletu	Szkielet programistyczny	Biblioteka
Łatwość uczenia	Trudny	Średni
Elastyczność	Niska	Wysoka
Dodatkowe narzędzia	Większość wbudowanych	Potrzeba dodatkowej instalacji
Styl architektury	MVC	Otwarta
Społeczność	Duża	Ogromna

5.1. Szkielet Angular

Historia Angular'a sięga roku 2009, kiedy to dwóch programistów pracujących w Google stworzyło projekt o nazwie AngularJS [20]. Po wewnętrznym wdrożeniu projektu w Google, w 2012 roku AngularJS został udostępniony, jako szkielet open-source, zyskując duże poparcie społeczności programistów. Jednak wraz z pojawieniem się bardziej złożonych interfejsów użytkownika, AngularJS zaczął mieć problemy z wydajnością. W odpowiedzi na te wyzwania, Google postanowiło opracować nową wersję szkieletu o nazwie "Angular", napisaną w języku TypeScript.

TypeScript, będący nadzbiorem JavaScriptu, wprowadza statyczne typowanie zmiennych oraz obiektowość, co eliminuje przypadkowe błędy i ułatwia zarządzanie kodem aplikacji [7]. Architektura Angular'a opiera się na wzorcu projektowym Model-Widok-Kontroler (*MVC, ang. Model-View-Controller*), co umożliwia lepszą strukturę i skalowalność kodu [20]. Warstwa modelu jest zarządzana za pomocą serwisów, które komunikują się z API (*ang. Application Programming Interface, tłum. Interfejs Programowania Aplikacji*), w celu pobierania i wysyłania danych. Warstwa widoku składa się z szablonów i dyrektyw, które umożliwiają tworzenie i modyfikację elementów DOM. Kontroler pełni rolę interfejsu, łącząc model i widok, obsługując przetwarzanie danych i walidację [21].

Dzięki podziałowi na modele, widoki i kontrolery, Angular zapewnia lepsze utrzymanie kodu, łatwiejsze testowanie i skalowalność. Komponenty są hierarchicznie osadzane w głównym pliku "app.component.ts", a moduły z własnymi ścieżkami trasowania umożliwiają wyświetlanie odpowiednich komponentów w zależności od nawigacji użytkownika. Każdy komponent składa się z plików: komponentu, styli, szablonu i testów jednostkowych. Angular oferuje również testy jednostkowe oraz testy E2E, które pozwalają sprawdzić działanie aplikacji w różnych scenariuszach.

Podsumowując, Angular jest potężnym narzędziem do tworzenia zaawansowanych interfejsów użytkownika, oferującym wiele gotowych narzędzi, które przyspieszają rozwój projektów. Znajduje często zastosowanie w projektach komercyjnych i jest ciągle rozwijany przez swojego producenta.

5.2. Biblioteka React

React to biblioteka do tworzenia interfejsów użytkownika (UI). Została stworzona w 2011 roku przez inżynierów z firmy Facebook, aby rozwiązać problemy z implementacją skomplikowanych UI [12]. React umożliwia tworzenie aplikacji internetowych oraz aplikacji mobilnych za pomocą narzędzia React Native. Jest to biblioteka open-source rozwijana przez Facebooka.

Głównym konceptem React jest tworzenie wirtualnego drzewa DOM, które reprezentuje rzeczywiste drzewo DOM w przeglądarce [12, 22]. Biblioteka porównuje oba drzewa i aktualizuje tylko te elementy, które uległy zmianie, co sprawia, że jest bardzo wydajny. Można go używać z różnymi architekturami, takimi jak MVC lub MVVM, gdzie MVVM (*ang. Model-*

View-ViewModel, tłum. Model-Widok-WidokModelu) dodaje warstwę widoku modelu, izolując logikę biznesową od warstwy prezentacji.

React wykorzystuje JavaScript, jako główny język programowania, ale od niedawna można również używać TypeScript. Biblioteka oferuje różne "uchwyty stanu" (*ang. state hooks*), takie jak useState, useContext i useReducer, które umożliwiają zarządzanie stanem komponentów [23]. Stan jest magazynem danych komponentów i może się zmieniać w zależności od kontekstu, co pozwala na dynamiczne renderowanie komponentów.

Ważnym elementem React jest JSX, które jest rozszerzeniem składni JavaScript. JSX pozwala na definiowanie elementów UI w sposób podobny do HTML i jest łatwiejszy do czytania i modyfikacji niż czysty JavaScript. JSX zapewnia również bezpieczeństwo, ponieważ przed wyświetleniem elementu React stosuje "znaki ucieczki", aby uniknąć wstrzykiwania niepożądanego kodu (atak XSS [17]).

W testowaniu, React korzysta z szkieletu testowego o nazwie "Jest", który umożliwia testowanie komponentów. Można również użyć biblioteki React Testing Library, która skupia się na testowaniu interakcji użytkownika z aplikacją poprzez testowanie drzewa DOM [23].

Podsumowując, React to również świetne narzędzie do tworzenia zaawansowanych interfejsów użytkownika. Posiada wiele funkcji i narzędzi, które ułatwiają tworzenie skalowalnych i bezpiecznych aplikacji internetowych. Jest ciągle rozwijany przez społeczność programistyczną i może być wykorzystywany w projektach komercyjnych.

5.3. Inne wykorzystane narzędzia

W ramach przeprowadzonych badań wykorzystano również inne narzędzia istotne dla działania aplikacji internetowych. Warto wspomnieć o bazie danych, zapleczu technicznym oraz architekturze REST API, które są nieodzownymi elementami aplikacji internetowych.

Jednym z najczęściej wykorzystywanych narzędzi jest szkielet Spring Boot. Został on stworzony w 2014 roku, jako rozszerzenie szkieletu Java Spring, mające na celu ułatwienie konfiguracji i zapewnienie większej modularności [24]. Spring Boot umożliwia tworzenie zaplecza technicznego dla aplikacji internetowych oraz mobilnych. Posiada wsparcie dla tworzenia mikro usług oraz implementacji CMS'ów, co sprawia, że jest popularny wśród programistów.

Do przechowywania danych wykorzystano bazę danych MySQL. Jest to relacyjna baza danych, która oferuje wiele funkcjonalności, takich jak transakcje, indeksy czy widoki [25]. Poza bazodanowymi funkcjonalnościami zapewnia również narzędzie do graficznej budowy schematu relacyjnej bazy danych oraz możliwość wygenerowania kodu SQL ze schematu lub odwrotnie. MySQL jest jednym z najpopularniejszych rozwiązań i charakteryzuje się prostą konfiguracją, co ułatwia jego wykorzystanie w małych projektach.

W kontekście komunikacji między elementami systemu informatycznego, zastosowano architekturę REST API [26]. REST API wykorzystuje protokół HTTP do komunikacji między klientem a serwerem. Dane są przekazywane w postaci zasobów, najczęściej w formacie JSON (*ang. JavaScript Object Notation, tłum. Notacja Obiektu JavaScript*). Komunikacja odbywa się za pomocą standardowych metod HTTP, takich jak GET, POST, PUT i DELETE [26]. REST API zapewnia jednolity interfejs oraz zasady, które umożliwiają skalowalność i łatwą zmienność systemu.

Wykorzystanie tych narzędzi przynosi wiele korzyści w procesie tworzenia aplikacji internetowych. Spring Boot zapewnia elastyczność i skalowalność, baza danych MySQL umożliwia przechowywanie i manipulację danymi, a architektura REST API zapewnia spójną komunikację między elementami systemu.

6. Wyniki badań

Podczas wykonywania badań nałożono szczególny nacisk, aby urządzenie, na którym zostały one wykonane było obciążone innymi usługami oprócz tych wymaganych do przeprowadzenia testów. Ze względów ograniczających dostępne zasoby, każde wywołanie zapytania do API zostało opóźnione o 5 ms, sumaryczny czas opóźniający został następnie odjęty od końcowego. Dla obu szkieletów przeprowadzono testy dwudziestokrotnie dla wszystkich przypadków na trzech popularnych przeglądarkach. Następnie obliczono podstawowe dane statystyczne w formie średniej oraz odchylenia standardowego. Poza wymienionymi danymi statystycznymi wykorzystano również testy statystyczne szerzej omówione w dalszych podrozdziałach.

6.1. Wyniki testów wydajnościowych

W Tabeli numer 3 zostały zamieszczone średnie wyniki wraz z uwzględnieniem odchylenia standardowego uzyskane dla poszczególnych badanych operacji, uzyskanych na przeglądarce Google Chrome.

Tabela 3: Wyniki testów na przeglądarce Google Chome

Próba	Angular [ms]	React [ms]
Pobranie i wyświetlenie 1 000 rekordów	10 258 ± 973	18 482 ± 1 630
Pobranie i wyświetlenie 5 000 rekordów	142 536 ± 13 163	491 189 ± 20 470
Pobranie i wyświetlenie 10 000 rekordów	603 473 ± 30 770	1 928 329 ± 79 178
Pobranie i wyświetlenie 50 zdjęć	1 832 ± 270	3 418 ± 322
Pobranie i wyświetlenie 100 zdjęć	3 829 ± 372	8 862 ± 327
Pobranie i wyświetlenie 200 zdjęć	5 725 ± 693	26 370 ± 2 494
Usunięcie 100 rekordów	598 ± 89	438 ± 36

Usunięcie 500 rekordów	6 203 ± 320	8 758 ± 247
Usunięcie 1 000 rekordów	18 431 ± 994	32 842 ± 805

W Tabeli numer 4 przedstawiono analogicznie wyniki testów wydajnościowych, wykonanych na przeglądarce Mozilla Firefox.

Tabela 4: Wyniki testów na przeglądarce Mozilla Firefox

Próba	Angular [ms]	React [ms]
Pobranie i wyświetlenie 1 000 rekordów	13 881 ± 389	25 409 ± 679
Pobranie i wyświetlenie 5 000 rekordów	195 851 ± 15 109	524 966 ± 23 173
Pobranie i wyświetlenie 10 000 rekordów	795 304 ± 81 189	2 206 582 ± 70 321
Pobranie i wyświetlenie 50 zdjęć	2 262 ± 243	3 602 ± 165
Pobranie i wyświetlenie 100 zdjęć	4 199 ± 197	8 590 ± 248
Pobranie i wyświetlenie 200 zdjęć	6 919 ± 928	22 978 ± 1 228
Usunięcie 100 rekordów	1 084 ± 16	1 121 ± 80
Usunięcie 500 rekordów	5 740 ± 197	6 722 ± 113
Usunięcie 1 000 rekordów	15 866 ± 659	25 486 ± 819

W przypadku testów wykonanych na przeglądarce Microsoft Edge, wyniki zostały przedstawione w Tabeli numer 5.

Tabela 5: Wyniki testów na przeglądarce Microsoft Edge

Próba	Angular [ms]	React [ms]
Pobranie i wyświetlenie 1 000 rekordów	10 704 ± 1 387	18 472 ± 1 710
Pobranie i wyświetlenie 5 000 rekordów	128 164 ± 1 805	479 991 ± 7 871
Pobranie i wyświetlenie 10 000 rekordów	536 469 ± 6 279	1 984 726 ± 32 450
Pobranie i wyświetlenie 50 zdjęć	2 961 ± 574	4 258 ± 287
Pobranie i wyświetlenie 100 zdjęć	4 310 ± 398	10 304 ± 426
Pobranie i wyświetlenie 200 zdjęć	7 432 ± 755	28 742 ± 945
Usunięcie 100 rekordów	738 ± 111	492 ± 61
Usunięcie 500 rekordów	6 803 ± 611	8 637 ± 333
Usunięcie 1 000 rekordów	23 892 ± 778	31 682 ± 663

Podczas porównywania wyników, warto zauważyć wysoce nieliniowy charakter uzyskanych danych, który jest związany ze zwiększeniem obciążenia spowodowanego liczbą przetwarzanych danych.

Podsumowując wyniki uzyskane na poszczególnych przeglądarkach dla danych typów badanych operacji zdecydowano się na dodatkowe przedstawienie w formie tabeli szkieletów, które szybciej poradziły sobie z zadaniem. Obserwując wyniki pokazane w Tabeli 6 można zauważyć znaczną przewagę szkieletu Angular (kolor czerwony) nad biblioteką React (kolor zielony) w zdecydowanej większości badanych operacji. Bazując na średnich wynikach pojedynczych operacji zestawionych w Tabelach 3 – 5, podano szkielet, który szybciej poradził sobie z wykonaniem zadanego zadania. W 25/27 operacji Angular okazał się lepszy pod względem czasowym.

Tabela 6: Wykaz sumaryczny testów wydajnościowych

	Chrome	Firefox	Edge
Pobranie i wyświetlenie 1 000 rekordów	Angular	Angular	Angular
Pobranie i wyświetlenie 5 000 rekordów	Angular	Angular	Angular
Pobranie i wyświetlenie 10 000 rekordów	Angular	Angular	Angular
Pobranie i wyświetlenie 50 zdjęć	Angular	Angular	Angular
Pobranie i wyświetlenie 100 zdjęć	Angular	Angular	Angular
Pobranie i wyświetlenie 200 zdjęć	Angular	Angular	Angular
Usunięcie 100 rekordów	React	Angular	React
Usunięcie 500 rekordów	Angular	Angular	Angular
Usunięcie 1 000 rekordów	Angular	Angular	Angular
Ocena punktowa	8:1	9:0	8:1
Sumarycznie	25:2		

6.2. Wyniki testów statystycznych

Jak wspomniano poza wykorzystaniem podstawowych danych statystycznych zdecydowano się na użycie testów statystycznych. Wykorzystane testy pozwolą potwierdzić lub odrzucić hipotezę domyślną - h0, mówiącą czy istnieje statystycznie istotna różnica pomiędzy badanymi grupami próbek.

W związku z decyzją o wykorzystaniu testów t-student'a, Mann'a-Whitney'a w przypadku porównania identycznych operacji wykonanych dla tej samej przeglądarki wykorzystano testy Shapiro-Wilk'a oraz Levene'a w celu sprawdzenia zgodności danych z rozkładem

normalnym oraz homogeniczności wariancji (warunek skorzystania z testu t-student'a) [27]. Drugim badaniem z wykorzystaniem testów Anova oraz Kruskal'a-Wallis'a było porównanie danych uzyskanych dla jednego szkieletu na wszystkich przeglądarkach [28]. Również w tym przypadku posiłkowano się danymi z testów Shapiro-Wilk'a oraz Levene'a w celu sprawdzenia warunków koniecznych do użycia testu Anova [29]. Ustalono stały poziom istotności dla wszystkich testów na poziomie $\alpha = 0.05$.

W Tabeli numer 7 zostały przedstawione wyniki testów t-student'a (kolor zielony) oraz Mann'a-Whitney'a. Testy wykonywane były w zależności od wartości p-value uzyskanych z testów Shapiro-Wilk'a oraz Levene'a wykonanych na poszczególnych grupach danych. Warto przypomnieć, że w przypadku tego testu porównywane były wyniki uzyskane dla identycznych operacji z wykorzystaniem osobnych szkieletów.

Tabela 7: Wyniki testów t-student'a oraz Mann'a-Whitney'a

Operacja	Chrome [p-value]	Firefox [p-value]	Edge [p-value]
Wyświetlenie 1000 rekordów	6,796E-08	6,796E-08	6,786E-08
Wyświetlenie 5000 rekordów	6,796E-08	6,796E-08	6,796E-08
Wyświetlenie 10 000 rekordów	6,796E-08	6,796E-08	6,796E-08
Wyświetlenie 50 zdjęć	7,451E-19	6,786E-08	9,160E-08
Wyświetlenie 100 zdjęć	6,786E-08	2,553E-39	1,721E-34
Wyświetlenie 200 zdjęć	6,796E-08	6,796E-08	6,796E-08
Usunięcie 100 rekordów	1,425E-07	1,464E-03	2,556E-07
Usunięcie 500 rekordów	6,796E-08	6,796E-08	4,533E-07
Usunięcie 1 000 rekordów	5,526E-36	1,307E-32	6,796E-08

Poza porównaniem między-szkieletowym, zdecydowano również o sprawdzenie szkieletów pod kątem zmiany przeglądarki. W Tabeli numer 7 przedstawiono wyniki testów Anova (kolor zielony) oraz Kruskal'a-Wallis'a. Testy wykonywano również warunkowo jak w przypadku wyników z Tabeli numer 8.

Tabela 8: Wyniki testów Anova oraz Kruskal'a-Wallis'a

Operacja	Angular [p-value]	React [p-value]
Wyświetlenie 1000 rekordów	1,278E-08	2,829E-09
Wyświetlenie 5000 rekordów	9,265E-12	1,908E-07
Wyświetlenie 10 000 rekordów	1,412E-11	1,002E-09
Wyświetlenie 50 zdjęć	2,296E-08	3,738E-09
Wyświetlenie 100 zdjęć	2,586E-04	3,527E-10
Wyświetlenie 200 zdjęć	5,216E-08	8,497E-11
Usunięcie 100 rekordów	1,019E-10	3,840E-10
Usunięcie 500 rekordów	7,756E-09	1,401E-09
Usunięcie 1 000 rekordów	5,266E-12	5,115E-37

7. Porównanie statystycznych danych popularnościowych

Poza testami wydajnościowymi i statystycznymi, przeanalizowano rankingi, wsparcie społeczności i dostępną dokumentację. Sprawdzenie wsparcia społeczności na popularnym forum StackOverflow jest ważne, ponieważ programiści często szukają tam rozwiązań dla swoich problemów. Sprawdzając liczbę tematów pod słowem kluczowym „reactjs", znaleziono 456 082 tematów, z drugiej strony pod etykietą „angular" 295 979 (z dnia 29.05.2023).

W przypadku chęci sprawdzenia liczby wyników wyszukiwania dotyczących dokumentacji, artykułów i stron poświęconych danemu szkieletowi programistycznemu, można wykorzystać licznik wbudowany do przeglądarki. Przykładowo, fraza "Angular documentation" generuje 151 000 000 wyników, podczas gdy fraza "React documentation" generuje 461 000 000 wyników (z dnia 29.05.2023).

Według ankiety StateOfJs przeprowadzonej w 2022 roku i obejmującej 39 472 odpowiedzi, React cieszy się największą popularnością wśród programistów szkieletów używanych do tworzenia interfejsu użytkownika w kategorii chęci dalszego użycia [30]. Od 2016 roku widać wyraźną przewagę Reacta, który od wyniku 52,7% wzrósł do obecnego poziomu 81,8%. Angular zajmuje drugie miejsce, choć jego popularność powoli spada na rzecz szkieletu Vue.js.

8. Wnioski

Analizując powyższą pracę można zauważyć, że oba szkielety programistyczne są niezwykle użytecznymi narzędziami przy tworzeniu zaawansowanego interfejsu użytkownika. Tak jak było to wspomniane w poprzednich rozdziałach, obecne aplikacje internetowe wymagają obsługi skomplikowanej logiki biznesowej oraz asynchronicznego przetwarzania danych. Poza tym, programiści zaczęli zwracać szczególną uwagę na czystość kodu oraz jak najwyższy poziom skalowalności pomagający sprostać ciągłemu rozwojowi aplikacji.

Pomimo licznych różnic, można zauważyć, że zarówno React jak i Angular są wszechstronnymi narzędziami i mogą zostać wykorzystane w celu rozwiązania praktycznie każdego problemu w kwestii wytwarzania interfejsu użytkownika. Największą ich różnicą jest sposób implementacji poszczególnych funkcjonalności. React korzysta ze swojego JSX wstrzykiwanego wraz z zawartym w nim HTML do wirtualnego drzewa DOM, natomiast Angular jest w stanie dodawać i modyfikować kontent przy pomocy różnego rodzaju dyrektyw.

Analizując wyniki poszczególnych testów wydajnościowych możemy zauważyć, że poza operacją usuwania 100 rekordów na przeglądarkach Chrome oraz Edge, w reszcie przypadków zwycięzcą okazał się Angular. W przypadku zwiększania liczby przetwarzanych danych czas rósł nieliniowo dla obu szkieletów. Poza porównywaniem między sobą Angulara i React, warto spojrzeć również w kontekście zachowania się danego narzędzia na poszczególnej przeglądarce. Angular najlepiej poradził sobie z przetwarzaniem obrazów na przeglądarce Chrome, natomiast React na Firefox. W przypadku pobierania rekordów, oba szkielety uzyskały najlepsze wyniki na przeglądarce Edge, natomiast dla usuwania rekordów na przeglądarce Firefox.

Testy statystyczne wykazały, że pomimo częstej zgodności z rozkładem normalnym lub spełnienia warunków homogeniczności wariancji, wyniki porównywane pomiędzy szkieletami lub wewnętrznie pomiędzy przeglądarkami wykazały w każdym przypadku statystycznie istotne różnice.

Obserwując dostępne statystyki trendów chęci użycia szkieletu, widać, że React silnie dominuje opinię programistów począwszy od 2016 r. Zaraz za nim pojawia się Angular, którym deweloperzy interesują się coraz mniej. Można z tego wywnioskować, że istnieje możliwość dalszego wzrostu użycia na rynku biblioteki React oraz konsekwentny spadek popularności szkieletu Angular.

Powracając do postawionej na początku tezy - „Szkielet programistyczny Angular jest szybszy w przetwarzaniu danych względem biblioteki React", można stanowczo powiedzieć, że została ona potwierdzona ze względu na wyniki testów wydajności. Poza potwierdzeniem głównej tezy można również odpowiedzieć na zadane pytania badawcze:

- Q1: Czy szkielet Angular szybciej pobierze i wyświetli odpowiednio 1 000, 5 000, 10 000 rekordów niż biblioteka React?
- Q2: Czy biblioteka React szybciej usunie odpowiednio 100, 500, 1 000 rekordów niż szkielet Angular?
- Q3: Czy szkielet Angular szybciej pobierze i wyświetli 50, 100, 200 zdjęć w formacie .JPG?

W przypadku Q1 oraz Q3 Angular lepiej poradził sobie z przetwarzaniem wszystkich grup danych, zatem na pytania można odpowiedzieć twierdząco. Dla Q2 pomimo przewagi Reacta w kwestii usuwania 100 rekordów na dwóch przeglądarkach, w pozostałych przypadkach dominował Angular osiągając lepsze czasy, zatem odpowiedź na pytanie jest negatywna.

Podsumowując powyższy artykuł, warto pamiętać o fakcie, iż przeprowadzone testy wydajnościowe miały specyficzny charakter w postaci sprawdzenia wydajności renderowania w ekstremalnych warunkach. Rekomendacjami do dalszych badań mogłyby być:

- wykonanie badań z wykorzystaniem dodatkowego szkieletu,
- wykonanie badań na większej liczbie rekordów oraz zdjęć, pobierając dane pojedynczo, a następnie renderując za jednym razem po zakończeniu zapytań do serwera.

Literatura

[1] K. Bielak, B. Borek, M. Plechawska-Wójcik, Web application performance analysis using Angular, React and Vue.js frameworks, Journal of Computer Sciences Institute 23 (2022) 77-83, https://doi.org/10.35784/jcsi.2827.

[2] E. Petukhova, Sitecore JavaScript Services Framework Comparison, praca magisterska, Åbo Akademi University, Turku 2019.

[3] C. L. Mariano, Benchmarking JavaScript Frameworks, praca magisterska, Technological University Dublin, Dublin 2017.

[4] J. Kalinowska, B. Pańczyk, Comparison of tools for creating SPA applications using the examples of Angular2 and React, Journal of Computer Sciences Institute 10 (2019) 1-4, https://doi.org/10.35784/jcsi.183.

[5] R. Ferguson, JavaScript and Application Frameworks: Angular. In: Beginning JavaScript, Apress, Berkeley, 2019.

[6] E. Wohlgethan, Supporting web development decisions by comparing three major javascript frameworks: Angular, react and vue.Js, praca licencjacka, Hamburg University of applied sciences, Hamburg 2018.

[7] B. Grynhaus, J. Hudgens, R.Hunte, M. Morgan, W. Stefanovski, TypeScript na warsztacie. Praktyczny przewodnik pisania efektywnego kodu, Helion, 2022.

[8] E. Molin, Comparison of single-page application frameworks. A method of how to compare Single-Page Application frameworks written in JavaScript, praca magisterska, KTH Royal Institute of Technology, Stockholm 2016.

[9] S. Mousavi, Maintainability Evaluation of Single Page Application Frameworks : Angular2 vs. React, praca licencjacka, Linnaeus University, Kalmar 2017.

[10] Y. Fain, A. Moiseev, Angular. Programowanie z użyciem języka TypeScript, Wydanie II, Helion, 2019.

[11] D. B. Duldulao, Practical Enterprise React: Become an Effective React Developer in Your Team, Apress, 2021.

[12] C. Gackenheimer, Introduction to React, Apress, Berkeley, 2015.

[13] A. Fedosejev, React. js essentials, Packt Publishing Ltd, Birmingham, 2015.

[14] J. Voutilainen, Evaluation of Front-end JavaScript Frameworks for Master Data Management Application Development, praca licencjacka, Metropolia University of Applied Sciences, Helsinki 2017.

[15] S. Retzius, E. Sundholm, Development of an evaluation model for client-side JavaScript Frameworks, praca magisterska, Linköping University, Linköping 2022.

[16] A. Lassen, JavaScript Frameworks A qualitative evaluation and comparison of the dominant factors in Angular and React, praca magisterska, Roskilde University, Roskilde 2020.

[17] S. D. Ankush, XSS attack prevention using DOM based filtering API, praca magisterska, National Institute of Technology Rourkela, Rourkela 2014.

[18] X. Lin, P. Zavarsky, R. Ruhl, D. Linskog, Threat modeling for CSRF attacks, International Conference on Computational Science and Engineering 3 (2009) 486-491, https://doi.org/10.1109/CSE.2009.372.

[19] L. K. Shar, H. B. K. Tan, Defeating SQL injection, Computer 46(3) (2012) 69-77, https://doi.org/10.1109/MC.2012.283.

[20] A. Bhaskar, A. E.Manjunath, An Interpretation and Anatomization of Angular: A Google Web Framework, International Research Journal of Engineering and Technology (IRJET) 7(05) (2020) 7613-7619.

[21] K. Simkhada, Transitioning Angular 2 User Interface (UI) intoReact, praca licencjacka, Metropolia University of Applied Sciences, Helsinki 2017.

[22] E. Saks, JavaScript Frameworks: Angular vs React vs Vue, praca licencjacka, Haaga-Helia University of Applied Sciences, Helsinki 2019.

[23] A. Mardan, React Quickly: Painless web apps with React, JSX, Redux, and GraphQL, Manning, 2017.

[24] C. E. d. Oliveira, G. L. Turnquist, A. Antonov, Developing Java Applications with Spring and Spring Boot: Practical Spring and Spring Boot Solutions for Building Effective Applications, Packt Publishing, Wielka Brytania, 2018.

[25] Y. Li, S. Manoharan, A performance comparison of SQL and NoSQL databases, 2013 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM), IEEE 8-13 (2013) 15-19, https://doi.org/10.1109/PACRIM.2013.6625441.

[26] R. Richards, Representational State Transfer (REST). In: Pro PHP XML and Web Services, Apress, Berkeley, 2006.

[27] S. M. Karadimitriou, E. Marshall, C. Knox, Mann-whitney u test, Sheffield Hallam University, Sheffield, 2018.

[28] T. J. Cleophas, A. H. Zwinderman, Non-parametric tests for three or more samples (Friedman and Kruskal-Wallis). Clinical data analysis on a pocket calculator: understanding the scientific methods of statistical reasoning and hypothesis testing, Springer (2016) 193-197, http://dx.doi.org/10.1007/978-3-319-27104-0_34.

[29] P. Mishra, U. Singh, C. M. Pandey,P. Mishra, G. Pandey, Application of student's t-test, analysis of variance, and covariance, Annals of cardiac anaesthesia, PUBMED 22(4) (2019) 407-411, https://doi.org/10.4103/aca.aca_94_19.

[30] Ankieta z 2022 r. wykonana przez StateOfJs, dotycząca szkieletów programistycznych używanych do tworzenia interfejsu użytkownika, https://2022.stateofjs.com/en-US/libraries/front-end-frameworks, [29.05.2023].

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 28 (2023) 264–272
Received: 14 June 2023
Accepted: 3 July 2023

Performance analysis of databases created in virtualized and containerized environment

Analiza wydajności baz danych utworzonych w zwirtualizowanym i skonteneryzowanym środowisku

Zygmunt Łata*, Maria Skublewska-Paszkowska

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

Abstract
Database systems are currently implemented on different environments - containerized and virtualized. This paper aimed to compare the performance of relational databases running on these two environments. Therefore, two research hypotheses were formulated. The first hypothesis assumed that databases running on Docker containers are more efficient than those on virtual machines. The second one assumed that the Oracle is the most efficient database regardless of which environment it was run on. MySQL, PostgreSQL, Microsoft SQL Server and Oracle databases were tested. The study measured the execution times of INSERT, UPDATE, DELETE and SELECT queries. Each test was repeated 100 times. It was stated that databases running on Docker containers outperform instances running on virtual machines. Furthermore, it was found that the PostgreSQL database have a definite advantage in performance over the rest, analysed databases, which allowed the second hypothesis to be rejected.

Keywords: performance comparison; relational databases; Docker containers; virtual machines

Streszczenie
Systemy bazodanowe są obecnie implementowane na różnych środowiskach - skonteneryzowanym oraz zwirtualizowanym. Artykuł ma na celu porównać wydajność relacyjnych baz danych uruchomionych w tych dwóch środowiskach. W związku z tym sformułowano dwie hipotezy badawcze. Pierwsza z nich zakładała, że bazy uruchomione na kontenerach Docker są wydajniejsze od tych na maszynach wirtualnych. W drugiej hipotezie przyjęto, że Oracle jest najwydajniejszą bazą niezależnie w jakim środowisku została uruchomiona. Badaniom poddano bazy MySQL, PostgreSQL, Microsoft SQL Server oraz Oracle. W ramach badań zmierzono czasy wykonywania zapytań INSERT, UPDATE, DELETE oraz SELECT. Każdy test został powtórzony 100 razy. Wykazano, że bazy uruchomione na kontenerach Docker przewyższają pod względem wydajności instancje działające na maszynach wirtualnych. Ponadto stwierdzono, że baza PostgreSQL ma zdecydowaną przewagę w wydajności nad pozostałymi, analizowanymi bazami, co pozwoliło odrzucić drugą hipotezę.

Słowa kluczowe: porównanie wydajności; relacyjne bazy danych; kontenery Docker; maszyny wirtualne

*Corresponding author
Email address: zygmunt.lata@pollub.edu.pl (Z. Łata)

1. Wstęp

Zdecydowana większość nowoczesnych aplikacji i systemów wykorzystuje bazy danych do przechowywania niezbędnych informacji w sposób trwały i usystematyzowany oraz do wykonywania na nich podstawowych operacji - zapis, edycja, usuwanie i odczyt. Na przestrzeni lat bazy danych były stale rozwijane i udoskonalane przez programistów, aby sprostać rosnącym potrzebom na rynku, takim jak szybszy dostęp i przetwarzanie danych. Istnieje wiele rodzajów baz danych rozróżnianych ze względu na model danych m.in. relacyjne, obiektowe, NoSQL (nierelacyjne), grafowe oraz dokumentowe (w formacie JSON) [1]. Relacyjne bazy danych wciąż są często stosowane, ponieważ mają swoje zalety i zajmują wysokie pozycje w rankingach popularności [2]. Najpopularniejszymi obecnie wśród nich są Oracle, MySQL, MS SQL Server (Microsoft) oraz PostgreSQL.

Współcześnie wiele firm wybiera wdrażanie baz danych w chmurze jako usługi (ang. Database as a Service, DBaaS), niekiedy wiąże się to z wysokimi kosztami eksploatacji takiego typu bazy i zarządzania nią, gdyż w przypadku takiego modelu przedsiębiorstwo nabywa dostęp do bazy na zasadzie płatnej subskrypcji z dostawcą chmury [3]. Dlatego też znaczna część organizacji decyduje się na utworzenie lokalnej infrastruktury (serwera), na którym wdrażana jest baza danych wykorzystując przy tym technologie wirtualizacji, która pozwala znacznie polepszyć wykorzystanie zasobów sprzętowych fizycznej infrastruktury.

Wirtualizacja pozwala za pomocą oprogramowania utworzyć odizolowaną warstwę sprzętu komputerowego [4]. W takim środowisku zasoby sprzętowe pojedynczego komputera takie jak: procesory, pamięć operacyjna, pamięć masowa mogą zostać rozłożone na wiele maszyn wirtualnych (ang. Virtual Machine, VM), które posiadają własny system operacyjny (ang. Operating System, OS) i działają jako niezależne urządzenia, wykorzystując przy tym jedynie część zasobów sprzętowych bazowego komputera (hosta). Takie podejście

pozwala na utworzenie oddzielnych instancji maszyn wirtualnych z zainstalowanymi i skonfigurowanymi na nich określonymi serwerami baz danych, do których następnie będzie można uzyskać dostęp z zewnątrz za pomocą odpowiednich żądań.

Konteneryzacja jest równie popularną technologią wykorzystywaną do wdrażania różnych usług, takich jak serwery aplikacji i bazy danych [5]. Podobnie jak wirtualizacja, odseparowuje warstwę sprzętu komputerowego od komputera hosta, w takim podejściu utworzone środowisko współdzieli zasoby sprzętowe i rozdziela je na wiele oddzielnych instancji, zwanych kontenerami. Kontenery nie posiadają własnego systemu operacyjnego, współdzielą one jądro systemu operacyjnego komputera bazowego i zawierają wyłącznie biblioteki systemu operacyjnego wraz z zależnościami wymaganymi do uruchomienia danej usługi. Oferują lepszą wydajność w wykorzystaniu zasobów sprzętowych niż maszyny wirtualne oraz cechują się lepszą przenośnością, ponieważ działają w jednakowy sposób w dowolnej infrastrukturze. Taka koncepcja pozwala na uruchamianie kontenerów jako oddzielnych instancji zawierających kompletne środowisko bazodanowe, najczęściej poprzez wskazanie określonego obrazu bazy danych, które są ogólnie dostępne w publicznym repozytorium DockerHub. Kontener utworzony w taki sposób można następnie bez żadnych przeszkód wdrożyć w chmurze.

Bazując na opisanych powyżej technologiach wdrażania usług, celem artykułu jest porównanie wydajności relacyjnych baz danych utworzonych w zwirtualizowanym i skonteneryzowanym środowisku.

2. Przegląd literatury

Istnieje wiele prac naukowych poświęconych tematyce porównującej wydajność rozwiązań bazodanowych oraz środowisk, w których zostają uruchomione. Autorzy stosują w nich różnorodne metody badawcze pozwalające ocenić i porównać wydajność wybranych przez nich systemów bazodanowych.

Ze względu na charakter pracy, artykułem otwierającym przegląd jest „*An Introduction to Docker and Analysis of its Performance*" [6], który charakteryzuje środowisko skonteneryzowane Docker i jego wydajność. Opisane zostały w nim główne elementy składające się na platformę Docker w porównaniu z technologią wirtualizacji i innymi dostępnymi technologiami konteneryzacji. Oprócz tego zwrócono również uwagę na zalety kontenerów Docker, takie jak: przenośność, szybkość, skalowalność, szybkie dostarczanie oraz gęstość, jak również na ich wady, takie jak: brak pełnej wirtualizacji, brak wsparcia dla starszych komputerów oraz niedostarczone w chwili obecnej rozwiązania związane z bezpieczeństwem kontenerów. Artykuł wykazuje, że kontenery Docker są szybsze w działaniu niż maszyny wirtualne i mniej obciążają zasoby.

W artykule „*Comparison of MySQL, MSSQL, PostgreSQL, Oracle databases performance, including virtualization*" [7] przeprowadzono porównanie wydajności relacyjnych DBMS w środowisku skonteneryzowanym. Badanymi systemami były MySQL, PostgreSQL, MS SQL Server i Oracle. Każdy z nich został wdrożony jako oddzielna instancja kontenera Docker z domyślnymi ustawieniami, wykorzystując oficjalne dystrybucje obrazów z DockerHub. Przeprowadzono serię eksperymentów z użyciem aplikacji webowej z zaimplementowanymi testami do pomiarów czasowych wykonywania określonych operacji. Eksperymenty obejmowały zapytania SELECT dla trzech różnych rozmiarów zestawów danych. Badania przeprowadzono dla czterech konstrukcji zapytania SELECT: z klauzulą ORDER BY lub WHERE (jednym warunkiem), z podzapytaniem skorelowanym i klauzulą ORDER BY oraz z funkcjami agregującymi i wielokrotnymi złączeniami wewnętrznymi INNER JOIN. Każdy scenariusz został powtórzony 100 razy. Na podstawie analizy otrzymanych wyników autorzy stwierdzili, że najbardziej wydajnym systemem okazał się Oracle. Potwierdza to słuszność hipotezy postawionej przez autorów, ponieważ uzyskał on najlepsze wyniki w prawie każdym scenariuszu badawczym, za wyjątkiem czwartego scenariusza, w którym wydajniejszym systemem okazał się MS SQL Server. Co więcej autorzy potwierdzili swoje przypuszczenia, że czas wykonywania zapytań wybierających dane z bazy jest tym większy, im większy jest zestaw przeszukiwanych danych w bazie.

Artykuł „*Comparison of query performance in relational a non-relation databases*" [8] porównuje wydajność dwóch różnych typów DBMS – relacyjnych i nierelacyjnych. W artykule zostały scharakteryzowane bazy danych SQL oraz NoSQL, zwracając uwagę na różnice pomiędzy nimi, ich cechy, zasadę działania oraz architekturę. Badania przeprowadzono na relacyjnych bazach danych: MySQL, MS SQL Server, Oracle oraz na nierelacyjnych: MongoDB, GraphQL, Apache Cassandra i Redis. Scenariusze skupiały się na wykonywaniu pomiarów czasu operacji wybierania danych oraz manipulacji nimi - wstawiania, aktualizowania oraz usuwania danych. Testy przeprowadzone zostały dla dwóch zestawów danych liczących po 10 000 i 100 000 rekordów. Wyniki jednoznacznie ilustrują przewagę nierelacyjnych baz nad relacyjnymi pod względem uzyskanych czasów poszczególnych operacji. Stosunek wydajności nierelacyjnych do relacyjnych baz wynosił odpowiednio: 1:15 dla wstawiania danych, 1:9 dla aktualizacji danych, 1:6 dla usuwania danych oraz 1:3 dla wybierania danych. MS SQL Server osiągnął najwyższą wydajność wśród relacyjnych baz, natomiast wśród nierelacyjnych najwydajniejsza okazała się MongoDB.

W artykule „*Microsoft SQL Server and Oracle: Comparative Performance Analysis*" [9] autorzy porównują wydajność dwóch relacyjnych baz danych: MS SQL Server i Oracle. Przeprowadzili testy, dokonując pomiaru czasów wykonania zapytań SELECT dla pojedynczej tabeli oraz dla wielu scalonych tabel za pomocą złączeń wewnętrznych INNER JOIN. Wykonywane zapytania podzielono na: zapytanie proste, z klauzulą WHERE lub ORDER BY. Przy ocenie wydajności uwzględniano również wymagania systemowe i sprzętowe, bezpieczeństwo oraz podstawowe cechy baz da-

nych: obsługiwane systemy operacyjne, języki programowania, składnia, interfejsy oraz języki SQL. Na podstawie analizy wyników autorzy wskazali na przewagę bazy Oracle, ponieważ jest wspierana przez więcej systemów operacyjnych i języków programowania. MS SQL Server przeważa tylko pod kątem używanego języka zapytań T-SQL, który ma prostszą składnią niż PL-SQL używany przez Oracle. W przypadku wymagań systemowych i sprzętowych nie zaobserwowano większych różnic, poza tym, że MS SQL Server wymagał dwukrotnie mniejszego zużycia pamięci RAM. W kwestii bezpieczeństwa Oracle oferuje silniejsze zabezpieczenia, jest bardziej odporny na błędy i uszkodzenia danych. Pod względem czasów wykonywania zapytań SELECT dużą przewagę miał MS SQL Server. Mimo to autorzy nie określają jednoznacznie, który z systemów jest wydajniejszy.

Artykuł „*Huge and Real-Time Database Systems: A Comparative Study and Review for SQL Server 2016, Oracle 12c & MySQL 5.7 for Personal Computer*" [10] porównuje wydajność relacyjnych baz danych: MySQL, MS SQL Server oraz Oracle. Przy analizie wydajności posłużono się pomiarami czasu wykonywanych operacji bazodanowych: aktualizacji kolumny, kopiowania rekordów z jednej tabeli i wstawiania ich do drugiej oraz wybierania danych. Zapytanie SELECT podzielono na: zapytanie proste, z klauzulą WHERE, ORDER BY lub GROUP BY oraz dla wielu scalonych tabel za pomocą wielokrotnych złączeń JOIN. Zestaw danych testowych liczył od 300 000 do 400 000 rekordów. Na podstawie otrzymanych wyników stwierdzono, że najwydajniejszą bazą pod względem szybkości wykonywanych operacji okazał się być MySQL.

Artykuł „*A Comparative Study on the Performance of the Top DBMS Systems*" [11] przedstawia porównanie wydajności relacyjnych systemów bazodanowych MySQL, IBM DB2, Oracle, MS SQL Server oraz MS Access. Testy opierały się na pomiarach czasu wykonywania zapytań UPDATE, DELETE i SELECT. Zapytanie SELECT podzielono na: zapytanie proste, z klauzulą ORDER BY lub WHERE z wieloma warunkami, z wieloma złączeniami wewnętrznymi INNER JOIN, z funkcjami agregującymi bez warunków lub z dodatkowymi warunkami po klauzuli HAVING, z wewnętrznie zagnieżdżonymi zapytaniami SELECT oraz zapytanie łączące wszystkie poprzednie w jedno atomowe. Zestaw danych testowych składał się z 1 000 000 rekordów. Ocenie podlegało również zużycie zasobów: procesora, pamięci RAM, pamięci wirtualnej oraz liczby zajętych wątków. Na podstawie przeprowadzonych eksperymentów stwierdzono, że IBM DB2 jest najwydajniejszym systemem bazodanowym pod względem szybkości wykonywania zapytań. Bezpośrednio za nim znajdują się: MS SQL Server, MySQL oraz Oracle. Natomiast ostatnie miejsce zajął MS Access charakteryzujący się najdłuższym czasem spośród badanych baz danych. Mimo to, zauważono, że baza MS Access wykazywała się najmniejszym zużyciem procesora oraz pamięci.

3. Cel i hipotezy badawcze

Artykuł jest efektem przeprowadzonych badań, których celem była ocena wydajności relacyjnych baz danych w dwóch różnych środowiskach - zwirtualizowanym i skonteneryzowanym. Narzędziem używanym do utworzenia środowisk zwirtualizowanych będzie Oracle VM VirtualBox, natomiast dla środowisk skonteneryzowanych oprogramowanie Docker Desktop. Każdy z systemów zostanie wdrożony jako oddzielna usługa na maszynie wirtualnej i na kontenerze Docker. Badanymi bazami będą MySQL, PostgreSQL, MS SQL Server i Oracle. Do oceny wydajności posłużą metryki oparte na pomiarach czasu wykonywania zapytań SQL typu INSERT, UPDATE, DELETE oraz SELECT.

Na podstawie przeglądu literatury postawione zostały następujące hipotezy badawcze:

H1: Bazy danych uruchomione na kontenerach Docker są wydajniejsze niż bazy uruchomione na maszynach wirtualnych pod względem szybkości wykonywania zapytań SQL.

H2: Baza danych Oracle jest najwydajniejsza pod kątem szybkości wykonywania zapytań SQL wśród badanych baz, niezależnie od środowiska, w którym została uruchomiona.

4. Metodyka badawcza

4.1. Stanowisko badawcze

Stanowiskiem badawczym, na którym zostały przeprowadzone eksperymenty jest fizyczna maszyna Lenovo Legion Y540, pełniąca rolę gospodarza (hosta) z zainstalowanym 64-bitowym systemem operacyjnym Windows 11 Home w wersji 22H2, kompilacji 22621.1265. W Tabeli 1 znajduje się pełna specyfikacja techniczna urządzenia testowego.

Tabela 1: Specyfikacja techniczna fizycznej maszyny gospodarza

Procesor	Intel(R) Core(TM) i5-9300HF CPU @ 2.40GHz, 2400 MHz, Rdzenie: 4, Procesory logiczne: 8
Dysk	Model: INTEL SSDPEKNW010T9, Typ: SSD (ang. Solid State Drive), Pojemność: 954 GB
Pamięć RAM	32 GB (DDR4, 2666MHz)

Środowiska zwirtualizowane z określonymi serwerami baz danych uruchomiono na maszynach wirtualnych za pomocą narzędzia Oracle VM VirtualBox w wersji 7.0.6. Bazy MySQL, PostgreSQL oraz MS SQL Server wykorzystywały taki sam obraz systemu operacyjnego Ubuntu w wersji 20.04.5 LTS (Focal Fossa), używany jako serwer. Natomiast baza Oracle wykorzystywała obraz systemu operacyjnego Oracle Linux w wersji 8.7 (R8-U7-x86_64), również w postaci serwerowej. Na każdej maszynie wirtualnej zainstalowano i skonfigurowano konkretną wersję serwera bazodanowego zgodnie z instrukcjami zawartymi na oficjalnych stronach dokumentacji twórców. W Tabeli 2 przedstawiono przydzielone zasoby dla maszyn wirtualnych.

Tabela 2: Zasoby sprzętowe przydzielone maszynom wirtualnym

Liczba rdzeni procesora (wirtualnych procesorów)	2
Dysk	Kontroler: SATA (ang. Serial ATA), Typ: VDI (ang. Virtual Disk Image), Pojemność: 30 GB, Dyna-

	miczny sposób przydzielania pamięci
Pamięć RAM	4 GB

W celu ujednolicenia tworzonych środowisk bazodanowych oraz uwiarygodnienia wyników eksperymentów na maszynie gospodarza korzystającej z systemu operacyjnego Windows został zainstalowany WSL (ang. Windows Subsystem for Linux) pozwalający na uruchamianie dystrybucji systemów operacyjnych bazujących na systemie Linux [12]. Używając narzędzia WSL zainstalowano dystrybucję systemu operacyjnego Ubuntu w wersji 20.04.5 LTS (Focal Fossa) zapewniając tym samym zgodność pod kątem systemów operacyjnych z maszynami wirtualnymi. Domyślną wersję WSL zaktualizowano do wersji WSL 2 wprowadzając tym samym znaczne usprawnienia takie jak wbudowane jądro Linux (ang. Linux kernel) oraz pełną kompatybilność wywołań systemowych zwiększając przy tym wydajność uruchamianych dystrybucji [13].

Środowiska skonteneryzowane z usługami bazodanowymi uruchomiono na kontenerach Docker za pomocą oprogramowania Docker Desktop w wersji 4.17.0 (99724). Narzędzie to używa silnika Docker w wersji 20.10.23 i zostało skonfigurowane tak aby korzystało z silnika WSL 2, zwiększając jednocześnie wydajność niż przy domyślnym silniku Hyper-V. Kontenery Docker zostały uruchomione z domyślnymi ustawieniami zgodnie z przedstawioną dokumentacją dla oficjalnych obrazów baz danych udostępnionych przez twórców na publicznym repozytorium obrazów DockerHub. Kontenery zdefiniowano za pomocą oddzielnych plików Compose, co znacznie uprościło proces uruchamiania i zarządzania nimi. Porty publiczne, na których zostały wystawione kontenery z bazami danych były zgodne z odpowiadającymi im portami na maszynie gospodarza. Do każdego kontenera podmontowano wolumin z ustawionymi opcjami praw dostępu do katalogu roboczego kontenera - odczytu z, zapisu do (ang. read-write, rw). Utworzony w ten sposób wolumin powiązywał katalog roboczy wdrażanego serwera bazodanowego z systemem plików maszyny gospodarza, co pozwoliło na zachowanie stanu bazy danych - jej struktury oraz przechowywanych danych.

Ponadto utworzony został plik konfiguracyjny o rozszerzeniu .wslconfig pozwalający na zmianę domyślnych ustawień przydzielonych zasobów sprzętowych dla wszystkich dystrybucji uruchamianych na WSL 2 [14]. W Tabeli 3 przedstawiono przydzielone zasoby sprzętowe zgodnie z plikiem konfiguracyjnym .wslconfig dla platformy Docker Desktop, wprowadza to ograniczenia dla kontenerów Docker uruchamianych na wszystkich dystrybucjach działających na WSL 2. Widoczny przydział pokrywa się z limitami wskazanymi dla maszyn wirtualnych.

Tabela 3: Zasoby sprzętowe przydzielone platformie Docker Desktop

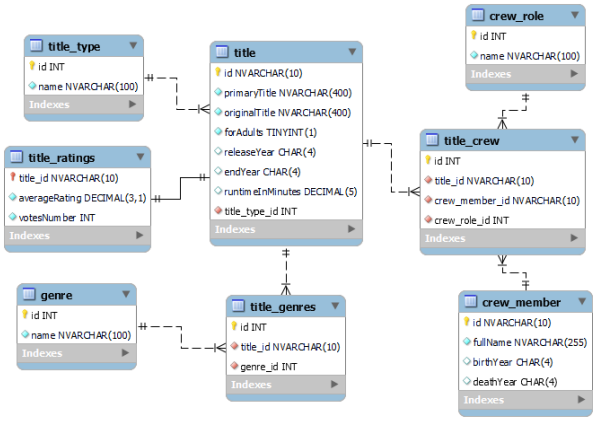
Liczba rdzeni procesora (wirtualnych procesorów)	2
Pamięć RAM	4 GB
Pamięć SWAP	1 GB

4.2. Bazy danych

W eksperymencie porównywano ze sobą następujące systemy bazodanowe:

- MySQL w wersji 8.0.32;
- PostgreSQL w wersji 15.2;
- MS SQL Server w wersji 2022 (16.0.4003.1);
- Oracle XE (Express Edition) w wersji 21c (21.3.0.0) – Production.

Schemat widoczny na rysunku 1 jest modyfikacją koncepcji bazy danych udostępnionej przez serwis IMDb, w której przedstawiony został realny zbiór danych na temat tytułów filmowych, telewizyjnych, seriali oraz powiązanych z nimi celebrytów na całym świecie [15].



Rysunek 1: Struktura bazy danych.

W badaniach zastosowano realne zasoby udostępnione przez serwis IMDb, które ze względu na wprowadzone modyfikacje schematu wymagały ich spreparowania. Opracowane w ten sposób dane wyodrębniono na cztery zestawy z rosnącą liczbą rekordów. Wszystkie bazy wypełniono jednakową treścią, oznaczało to, że początkowe dane pozostawały niezmienne, natomiast do poprzedniej puli danych dodawano liczbę rekordów określoną dla konkretnego zestawu danych do tabeli *title* i do powiązanych z nią tabel. W Tabeli 4 przedstawiono przydział liczby rekordów dla poszczególnych zestawów testowych.

Tabela 4: Przydział liczby rekordów dla poszczególnych zestawów danych

Tabela	Zestaw danych nr 1	Zestaw danych nr 2	Zestaw danych nr 3	Zestaw danych nr 4
title	1 000	10 000	100 000	1 000 000
title_genres	1 848	17 171	174 023	1 603 316
crew_member	842	12 971	230 403	1 346 385
title_crew	4 018	72 109	1 013 876	7 750 390
title_ratings	1 000	10 000	100 000	1 000 000

4.3. Metryki pomiaru

Badania przeprowadzone zostały dla wszystkich systemów bazodanowych wymienionych w podrozdziale 4.2, które to zostały zaimplementowane w dwóch różnych środowiskach - zwirtualizowanym oraz skonteneryzowanym. W trakcie trwania każdego eksperymentu uruchomiona była jedynie pojedyncza instancja bazy danych. Testy opierały się na pomiarach czasów wykonywania konkretnych operacji bazodanowych, które zostały powtórzone cyklicznie 100 razy, aby osiągnąć jak najbardziej wiarygodne i prawidłowe wyniki. Eksperymenty zostały podzielone na główne scenariusze ze względu na wykonywany rodzaj zapytania INSERT, UPDATE, DELETE oraz SELECT. Każdy z nich został powtórzony dla kolejnych zestawów danych o zmieniającej się liczbie rekordów przedstawionych w Tabeli 4.

Przy zestawieniu wszystkich pomiarów, pierwszy z nich należało odrzucić, ponieważ znacznie odbiegał od reszty ze względu na dłuższy czas wykonywania zapytania. Sytuacja ta spowodowana była przez brak wcześniejszej analizy składniowej określonego w scenariuszu zapytania, co jednocześnie przekładało się na brak zbudowanego planu wykonania zapytania. W rezultacie pierwsze wykonanie określonego zapytania w scenariuszu poddawane było przez system bazodanowy analizie składniowej. W dalszej kolejności uruchamiany był optymalizator zapytań, który tworzył możliwe plany wykonania zapytania, po czym obliczał ich przybliżone koszty. Następnie wybierał on plan o najniższym możliwym koszcie i zapisywał go, a potem przekazywał go do kolejnego etapu, w którym było wykonywane zapytanie na podstawie tak skonstruowanego planu [17].

Lista kroków, jakie zrealizowano podczas dokonywania pomiaru czasu wykonywania zapytania SQL została zaprezentowana w Tabeli 5.

Tabela 5: Lista czynności realizowanych w trakcie pomiaru czasu wykonywania zapytania SQL

Numer realizowanego kroku	Wykonywana czynność
1	Pomiar czasu przed wykonaniem zapytania SQL
2	Wykonanie określonego zapytania SQL
3	Pomiar czasu po wykonaniu zapytania SQL
4	Obliczenie różnicy pomiędzy pomiarami przed i po wykonaniu zapytania SQL według następującego wzoru: czas_wykonania = (czas_po – czas_przed) * 1000
5	Wyczyszczenie pamięci podręcznej (ang. cache) zapytań, bazy danych

Wyniki pomiarów zapisywano do oddzielnej ramki danych (ang. DataFrame). Po przeprowadzeniu eksperymentu dla konkretnego scenariusza i uzupełnieniu danych w ramce danych została ona wyeksportowana do pliku .csv. Nazwa utworzonego pliku była zgodna z następującym wzorem – scenariusz_X_zestaw_danych_Y.csv, który składał się z dwóch oddzielnych członów. Litera X odnosiła się do realizowanego obecnie scenariusza badań, natomiast litera Y wskazywała na aktualnie wybrany zestaw danych.

4.4. Scenariusze

Pierwszy scenariusz opierał się na wykonaniu operacji INSERT, która polegała na wstawieniu pojedynczego rekordu do tabeli *title*, dane wstawiane były rekord po rekordzie. Budowa zapytania została przedstawiona na listingu 1.

Listing 1: Zapytanie SQL typu INSERT wstawiające pojedynczy rekord do tabeli *title*

```
INSERT INTO title (
    id, primaryTitle, originalTitle, forAdults,
    releaseYear, endYear, runtimeInMinutes,
    title_type_id
)
VALUES (
    :id, :primaryTitle, :originalTitle, :forAdults,
    :releaseYear, :endYear, :runtimeInMinutes,
    :title_type_id
);
```

Drugi scenariusz opierał się na wykonaniu operacji DELETE, która polegała na usunięciu pojedynczego rekordu z tabeli *title_ratings* przy zastosowaniu klauzuli WHERE z jednym warunkiem filtrowania. Składnię zapytania przedstawiono na listingu 2.

Listing 2: Zapytanie SQL typu DELETE usuwające pojedynczy rekord z tabeli *title_ratings* z jednym warunkiem filtrowania

```
DELETE FROM title_ratings
WHERE title_id = 'tt0000514';
```

Trzeci scenariusz opierał się na wykonaniu operacji UPDATE, która polegała na modyfikacji pojedynczego rekordu w tabeli *title_ratings* przy zastosowaniu klauzuli WHERE z jednym warunkiem filtrowania. Wartość w kolumnie *averageRating* jest wypełniana wygenerowaną liczbą pseudolosową (zmiennoprzecinkową) z przedziału (5, 10), zaokrąglaną do 1 miejsca po przecinku. Zapytanie zostało zilustrowane na listingu 3.

Listing 3: Zapytanie SQL typu UPDATE modyfikujące pojedynczy rekord w tabeli *title_ratings* z jednym warunkiem filtrowania

```
UPDATE title_ratings
SET averageRating = ROUND((RAND() * 5) + 5, 1)
WHERE title_id = 'tt0000264';
```

W przypadku scenariuszy nr 1-3 konieczne było sekwencyjne wykonanie następujących czynności w celu przywrócenia stanu bazy danych do stanu sprzed wykonania zapytania:

- dezaktywacja/usunięcie kluczy obcych, które są powiązane z tabelą (opcjonalnie);
- usunięcie wszystkich danych z tabeli;
- wstawienie danych do tabeli, odpowiednio dla badanego zestawu danych;
- aktywacja/utworzenie kluczy obcych, które są powiązane z tabelą (opcjonalnie).

W scenariuszu nr 1, czynności były realizowane po zakończeniu wszystkich iteracji, tj. po wykonaniu

wszystkich zapytań i zebraniu wszystkich pomiarów. Natomiast dla scenariuszy nr 2-3 czynności były wykonywane po każdej iteracji, tzn. po wykonaniu zapytania i dokonania zapisu pomiaru.

Czwarty scenariusz opierał się na wykonaniu operacji SELECT i został podzielony na dwa przypadki.

Pierwszy przypadek A polegał na selekcji rekordów z tabeli *title* przy zastosowaniu klauzuli WHERE z pojedynczym warunkiem filtrowania. Wybierane są wszystkie rekordy, w których wartość kolumny *title_type_id* wynosi 1. W rezultacie zwracane są wszystkie tytuły, w których typ tytułu został przypisany jako film. Nałożony został również limit zwracanych rekordów przy pomocy klauzuli LIMIT, maksymalna liczba zwracanych rekordów została zawężona do 100. Oprócz tego dane zostały posortowane rosnąco względem wartości w kolumnie *id* poprzez wykorzystanie klauzuli ORDER BY z opcją ASC. Budowa zapytania została przedstawiona na listingu 4.

Listing 4: Zapytanie SQL typu SELECT wybierające dane z tabeli *title* z pojedynczym warunkiem filtrowania, limitem zwracanych rekordów i sortowaniem po jednej kolumnie

```
SELECT * FROM title
WHERE title_type_id = 1
ORDER BY id ASC
LIMIT 100;
```

Drugi przypadek B polegał na selekcji rekordów ze scalonych ze sobą dwóch tabel poprzez wykorzystanie klauzuli INNER JOIN. Tabela *title_genres* została połączona złączeniem wewnętrznym z tabelą *genre*. W rezultacie zwracane są rekordy zawierające identyfikator, nazwę gatunku oraz liczbę tytułów o określonym gatunku. W tym celu została użyta pojedyncza funkcja agregująca COUNT, która zwróciła liczbę rekordów na podstawie kolumny *title*_id. Funkcja agregująca wykorzystywała pogrupowane dane według kolumn *genre_id* oraz name przy pomocy klauzuli GROUP BY. Ponadto dane zostały posortowane rosnąco względem wartości w kolumnie *genre_id* przy pomocy klauzuli ORDER BY z opcją ASC. Składnię zapytania zilustrowano na listingu 5.

Listing 5: Zapytanie SQL typu SELECT wybierające dane z tabeli *title_genres* z pojedynczą funkcją agregacji, pojedynczym złączeniem wewnętrznym, grupowaniem danych i sortowaniem po jednej kolumnie

```
SELECT
    tg.genre_id as genre_id,
    g.name as genre_name,
    COUNT(tg.title_id) as titles_counter
FROM title_genres tg
INNER JOIN genre g on tg.genre_id = g.id
GROUP BY tg.genre_id, g.name
ORDER BY genre_id ASC;
```

5. Wyniki

Wyniki dotyczące scenariusza badawczego nr 1, w którym mierzony był czas wykonywania instrukcji INSERT polegającej na wstawieniu pojedynczego rekordu do tabeli *title* zostały przedstawione w Tabeli 6.

Wyniki dotyczące scenariusza badawczego nr 2, w którym badany był czas wykonywania instrukcji DELETE usuwającej pojedynczy rekord z tabeli *title_ratings* spełniający określony warunek zostały przedstawione w Tabeli 7.

Tabela 6: Średnie czasy wykonywania zapytań dla scenariusza badawczego nr 1

Baza danych	Średni czas wykonywania zapytań bazodanowych (ms)			
	Zestaw danych nr 1	Zestaw danych nr 2	Zestaw danych nr 3	Zestaw danych nr 4
MySQL kontener	4,059	4,080	4,578	4,410
MySQL maszyna wirtualna	9,047	9,986	9,040	9,382
PostgreSQL kontener	1,665	1,664	1,660	1,655
PostgreSQL maszyna wirtualna	2,549	2,747	2,562	2,344
Microsoft SQL Server kontener	7,404	7,839	11,835	13,978
Microsoft SQL Server maszyna wirtualna	9,000	13,447	18,651	20,595
Oracle kontener	9,505	10,072	15,023	14,218
Oracle maszyna wirtualna	10,628	10,658	10,223	11,314

Tabela 7: Średnie czasy wykonywania zapytań dla scenariusza badawczego nr 2

Baza danych	Średni czas wykonywania zapytań bazodanowych (ms)			
	Zestaw danych nr 1	Zestaw danych nr 2	Zestaw danych nr 3	Zestaw danych nr 4
MySQL kontener	4,107	4,804	4,502	4,871
MySQL maszyna wirtualna	9,332	9,046	9,219	11,429
PostgreSQL kontener	1,893	2,142	3,687	3,200
PostgreSQL maszyna wirtualna	2,668	2,743	4,653	3,998
Microsoft SQL Server kontener	11,883	18,183	75,155	308,270
Microsoft SQL Server maszyna wirtualna	16,745	22,428	80,450	364,955
Oracle kontener	9,478	8,797	16,369	13,422
Oracle maszyna wirtualna	10,539	9,575	17,543	23,677

Wyniki uzyskane w Tabeli 6 odnoszące się do scenariusza badawczego nr 1, pokazują, że najlepszym wydajnościowo systemem bazodanowym pod względem szybkości wykonywania zapytań SQL typu INSERT okazał się system PostgreSQL. Dla wszystkich zestawów danych średni czas wykonywania zapytań w przypadku bazy PostgreSQL na kontenerze Docker wynosił 1,7 ms, dlatego też zajął on pierwsze miejsce wśród badanych systemów bazodanowych. Odnosząc się do wyników uzyskanych w pierwszym scenariuszu dla zestawu 4, ostatnie miejsce z najdłuższym czasem wynoszącym 20,6 ms zajął MS SQL Server na maszynie wirtualnej.

Otrzymane wyniki pomiarów w Tabeli 7 dotyczące scenariusza badawczego nr 2, obrazują, że najwydajniejszą bazą danych dla wszystkich zestawów danych jest PostgreSQL osiągając wyniki bliskie 2-5 ms. Równie wydajną bazą danych okazuje się być baza MySQL, osiągając podobne wyniki. Najgorszą natomiast przy tego rodzaju operacjach okazała się być baza MS SQL Server. W początkowej fazie eksperymentu system nieznacznie odbiegał od pozostałych, jednakże wraz z przekroczeniem określonego progu 10 000 rekordów czas wykonywania zapytania drastycznie wzrastał. Dla zestawu 4 uzyskano wyniki w przedziale 308-365 ms.

Wyniki dotyczące scenariusza badawczego nr 3, w którym sprawdzany był czas wykonywania instrukcji UPDATE modyfikującej pojedynczy rekord z tabeli *title_ratings* spełniający określony warunek zostały przedstawione w Tabeli 8.

Tabela 8: Średnie czasy wykonywania zapytań dla scenariusza badawczego nr 3

Baza danych	Średni czas wykonywania zapytań bazodanowych (ms)			
	Zestaw danych nr 1	Zestaw danych nr 2	Zestaw danych nr 3	Zestaw danych nr 4
MySQL kontener	4,135	4,626	4,486	5,086
MySQL maszyna wirtualna	9,506	8,699	9,357	12,791
PostgreSQL kontener	1,922	2,179	3,817	3,381
PostgreSQL maszyna wirtualna	3,161	3,079	4,790	4,092
Microsoft SQL Server kontener	13,312	19,109	78,860	309,793
Microsoft SQL Server maszyna wirtualna	18,797	25,219	84,461	371,181
Oracle kontener	48,591	56,526	63,944	59,614
Oracle maszyna wirtualna	50,727	54,686	60,560	62,214

Na podstawie uzyskanych wyników w Tabeli 8 odnoszących się do scenariusza badawczego nr 3, stwierdzono, że najkrótszy czas wykonywania zapytań dla wszystkich zestawów danych osiągnął PostgreSQL z czasem rzędu 2-5 ms. Inną równie wydajną bazą okazuje się być MySQL, osiągając wynik na poziomie 4-13 ms. Różnica czasu wykonywania zapytań pomiędzy PostgreSQL, a MySQL była niewielka i wynosiła kilka milisekund. Najgorzej dla zestawu 1 oraz 2 wypadła baza Oracle z czasem wynoszącym około 48,6- 56,5 ms. Podczas gdy dla zestawu 3 oraz 4 najmniej wydajną okazuje się być baza MS SQL Server z wynikiem nawet 372 ms.

Wyniki dotyczące scenariusza badawczego nr 4 zostały przedstawione w Tabelach 9-10

Tabela 9: Średnie czasy wykonywania zapytań dla scenariusza badawczego nr 4 przypadku A

Baza danych	Średni czas wykonywania zapytań bazodanowych (ms)			
	Zestaw danych nr 1	Zestaw danych nr 2	Zestaw danych nr 3	Zestaw danych nr 4
MySQL kontener	1,399	1,667	1,745	3,200
MySQL maszyna wirtualna	1,646	1,903	1,779	2,022
PostgreSQL kontener	1,083	1,297	1,292	2,341
PostgreSQL maszyna wirtualna	1,054	2,010	1,813	2,059
Microsoft SQL Server kontener	6,199	7,011	15,627	30,355
Microsoft SQL Server maszyna wirtualna	10,923	15,138	126,430	432,398
Oracle kontener	11,363	12,037	16,668	26,359
Oracle maszyna wirtualna	12,162	18,409	70,055	174,274

Tabela 10: Średnie czasy wykonywania zapytań dla scenariusza badawczego nr 4 przypadku B

Baza danych	Średni czas wykonywania zapytań bazodanowych (ms)			
	Zestaw danych nr 1	Zestaw danych nr 2	Zestaw danych nr 3	Zestaw danych nr 4
MySQL kontener	3,317	18,700	223,249	6995,514
MySQL maszyna wirtualna	4,067	19,881	266,028	8401,838
PostgreSQL kontener	0,912	0,829	0,809	1,639
PostgreSQL maszyna wirtualna	1,117	1,056	0,875	257,710
Microsoft SQL Server kontener	7,744	9,065	30,143	211,049
Microsoft SQL Server maszyna wirtualna	12,832	13,570	47,429	341,674
Oracle kontener	16,509	14,566	44,437	280,221

Oracle maszyna wirtualna	15,730	21,188	44,750	166,475

W czwartym scenariuszu dla przypadku A mierzony był czas wykonywania instrukcji SELECT wybierającej dane z tabeli *title* z pojedynczym warunkiem filtrowania, limitem i sortowaniem. Analizując otrzymane wyniki w Tabeli 9, zauważono, że dwie bazy danych wyróżniały się na tle pozostałych, były to bazy PostgreSQL i MySQL. Średnie czasy dla bazy PostgreSQL i MySQL nie różniły się znacznie od siebie, oscylowały w granicach 1-3,2 ms. Porównując ze sobą uzyskane czasy dla zestawu 4, zdecydowaną przewagę w wydajności nad pozostałymi, analizowanymi bazami danych miała baza MySQL na maszynie wirtualnej. Kolejno po niej znaj-dowały się odpowiednio bazy PostgreSQL na maszynie wirtualnej, PostgreSQL na kontenerze Docker i MySQL na kontenerze Docker. Dla wcześniejszych zestawów od 1 do 3 przewagę nad maszynami wirtualnymi miały kontenery Docker. Najgorszym wyborem dla zestawu 4 okazała się być baza MS SQL Server na maszynie wirtualnej zajmująca ostatnie miejsce, dla której czas wyniósł około 432 ms.

W czwartym scenariuszu dla przypadku B mierzony był czas wykonywania instrukcji SELECT wybierającej dane z tabeli *title_genres* z pojedynczą funkcją agregacji, pojedynczym złączeniem wewnętrznym, grupowaniem danych i sortowaniem. Według uzyskanych wyników w Tabeli 10, najlepszą okazuje się baza PostgreSQL na kontenerze Docker osiągając czas rzędu 1,6 ms dla największego zestawu danych. W zestawieniu z pozostałymi systemami można zauważyć, że różnica pomiędzy czasami dla zestawu 4 była bardzo wysoka i wynosiła ponad 100 ms. Najgorzej w tym zestawieniu wypadła baza danych MySQL na kontenerze Docker i na maszynie wirtualnej z uwagi na jej bardzo oddalony od tej granicy wynik, który wyniósł niemalże 7-8,4 s.

6. Podsumowanie

Reasumując, celem artykułu było porównanie ze sobą wydajności systemów bazodanowych uruchomionych w środowisku skonteneryzowanym oraz zwirtualizowanym. Badanymi systemami bazodanowymi były MySQL, PostgreSQL, Microsoft SQL Server i Oracle. Przy ocenie wydajności posłużono się średnimi czasami wykonywanych zapytań SQL typu INSERT, UPDATE, DELETE oraz SELECT. Zdefiniowano dwie hipotezy badawcze. Pierwsza hipoteza H1 zakładała, że bazy danych uruchomione na kontenerach są wydajniejsze od baz uruchomionych na maszynach wirtualnych. W drugiej hipotezie H2 ustalono, że Oracle jest najwydajniejszy spośród testowanych baz danych niezależnie w jakim środowisku został uruchomiony. Na podstawie wyników przedstawionych w Tabelach 6-10 hipoteza H1 została potwierdzona. Podczas gdy hipoteza H2 została odrzucona, gdyż wykazano, że PostgreSQL charakteryzował się zdecydowanie większą wydajnością pod względem szybkości wykonywania zapytań spośród analizowanych systemów bazodanowych. Porównywalne wyniki do PostgreSQL uzyskała baza MySQL, osiągająca nieznacznie dłuższe czasy wykonywania zapytań. Z kolei system Oracle znajdował się na trzecim, bądź też ostatnim miejscu w zależności od badanego scenariusza. Podsumowując, uzyskane wyniki różnią się nieco od dotychczasowych badań dotyczących omawianych problemów. W obecnej chwili wykazano, że najlepszą bazą jest PostgreSQL, a po niej osiągając zbliżone wyniki znajduje się baza MySQL. Różnice te mogą wynikać z nieustannego postępu technologicznego oraz zmian wprowadzanych w systemach bazodanowych, konfiguracji baz danych, wyboru środowisk, na których są uruchomione, ustawień połączenia oraz języka, w jakim została napisana aplikacja.

Literatura

[1] What is a database? – Simple definition by Oracle, https://www.oracle.com/pl/database/what-is-database/, [02.2023].

[2] Complete ranking of relational DBMS, https://db-engines.com/en/ranking/relational+dbms, [02.2023].

[3] Database as a Service (DBaaS) by TechTarget, https://www.techtarget.com/searchdatamanagement/definition/database-as-a-service-DBaaS, [02.2023].

[4] IBM topics series – What is virtualization?, https://www.ibm.com/topics/virtualization, [02.2023].

[5] IMB topics series – What is containerization?, https://www.ibm.com/topics/containerization, [02.2023].

[6] B.B. Rad, H.J. Bhatti, M. Ahmadi, An Introduction to Docker and Analysis of its Performance, International Journal of Computer Science and Network Security (IJCSNS) 17(3) (2017) 228-235.

[7] R. Klewek, W. Truskowski, M. Skublewska-Paszkowska, Comparison of MySQL, MSSQL, PostgreSQL, Oracle databases performance, including virtualization, Journal of Computer Sciences Institute (JCSI) 16 (2020) 279-284.

[8] R. Čerešňák, M. Kvet, Comparison of query performance in relational a non-relation databases, Transportation Research Procedia 40 (2019) 170–177.

[9] M. Ilić, L. Kopanja, D. Zlatković, M. Trajković, D. Ćurguz, Microsoft SQL Server and Oracle: Comparative Performance Analysis, The 7th International Conference on Knowledge Management and Informatics, Vrnjačka Banja, 7(poz. 5) (2021) 33-40.

[10] K. Islam, K. Ahsan, S. A. K. Bari, M. Saeed, S. Asim, Huge and Real-Time Database Systems: A Comparative Study and Review for SQL Server 2016, Oracle 12c & MySQL 5.7 for Personal Computer, Journal of Basic & Applied Sciences 13 (2017) 481-490.

[11] Y. Bassil, A Comparative Study on the Performance of the Top DBMS Systems, Journal of Computer Science & Research 1(1) (2012) 20-31.

[12] What is a WSL 2?, https://learn.microsoft.com/en-us/windows/wsl/about#what-is-wsl-2, [03.2023].

[13] Comparing WSL versions, https://learn.microsoft.com/en-us/windows/wsl/compare-versions, [03.2023].

[14] Settings configuration in WSL – Microsoft manual, https://learn.microsoft.com/en-us/windows/wsl/wsl-config, [03.2023].

[15] Specification of IMDb non-commercial datasets, https://developer.imdb.com/non-commercial-datasets/, [03.2023].

[16] Lecture about query optimization – Section 10.2, https://edu.pjwstk.edu.pl/wyklady/szb/scb/rW10.htm, [03.2023].

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 28 (2023) 273–278
Received: 15 June 2023
Accepted: 3 July 2023

A comparative analysis of non-relational databases in e-commerce applications

Analiza porównawcza nierelacyjnych baz danych w zastosowaniach e-commerce

Grzegorz Rożek*, Kacper Saweczko*, Małgorzata Plechawska-Wójcik

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

Abstract
In this article, a comparative analysis of non-relational databases was conducted to determine the best database for e-commerce systems. The research thesis was "MongoDB is the best choice for e-commerce applications. Non-relational systems such as MongoDB and Apache Cassandra were used for the study and the results were compared with a relational PostgreSQL database. The main research criterion was performance testing of several types of queries based on execution time. To implement the research, typical e-commerce databases were created and then tested in a .NET test application created by authors. In addition, the difference in community support between non-relational and relational systems was determined. The research showed that MongoDB is best suited for e-commerce systems.

Keywords: non-relational database; MongoDB; Cassandra; e-commerce

Streszczenie
W tym artykule przeprowadzono analizę porównawczą nierelacyjnych baz danych w celu wyłonienia najlepszej bazy do systemów e-commerce. Postawiono tezę badawczą „MongoDB jest najlepszym wyborem w zastosowaniach e-commerce. Do badań wykorzystano systemy nierelacyjne takie jak MongoDB oraz Apache Cassandra, a wyniki zestawiono z relacyjną bazą PostgreSQL. Głównym kryterium badawczym były testy wydajności kilku rodzajów zapytań na podstawie czasu realizacji. Do realizacji badań zostały stworzone typowe dla e-commerce bazy danych, które następnie poddano testom w stworzonej aplikacji testowej w środowisku .NET. Dodatkowo określono różnicę we wsparciu społeczności pomiędzy systemami nierelacyjnymi a relacyjnymi. Badania wykazały, że do systemów e-commerce najlepiej przystosowane jest MongoDB.

Słowa kluczowe: nierelacyjna baza danych; MongoDB; Cassandra; handel elektroniczny

*Corresponding authors
Email address: grzegorz.rozek@pollub.edu.pl,kacper.saweczko@pollub.edu.pl

1. Wstęp

Handel elektroniczny [1], nazywany także e-handlem albo e-commerce, odgrywa coraz większą rolę w dzisiejszym globalnym systemie gospodarczym. Obejmuje on sprzedaż i zakup produktów oraz usług za pośrednictwem Internetu i innych elektronicznych środków komunikacji. Rozwój technologii i wzrost popularności Internetu przyczyniły się do dynamicznego rozwoju handlu elektronicznego na całym świecie. E-commerce oferuje wiele korzyści zarówno dla konsumentów, jak i dla przedsiębiorstw. Klienci mogą łatwo porównywać produkty i ceny, robić zakupy o dowolnej porze oraz mieć dostęp do szerokiego wyboru towarów z całego świata. Umożliwia on również wygodne płatności online, co eliminuje konieczność fizycznego udawania się do sklepu i płacenia gotówką. Klienci mają do dyspozycji różne opcje dostawy, w tym dostawy do domu lub odbioru w punkcie odbioru. Coraz większa popularność internetowych serwisów e-commerce wymaga wybrania odpowiedniej bazy danych, która zapewni skalowalność oraz bezpieczny i szybki dostęp do danych dla cały czas zwiększającej się liczby klientów.

Celem pracy jest przeprowadzenie analizy porównawczej nierelacyjnych baz danych w zastosowaniach e-commerce oraz porównanie wyników z wybraną bazą relacyjną. Analiza skupia się na porównaniu architektur omawianych baz oraz przetestowaniu ich wydajności. Pomoże to wskazać, który z wybranych systemów nierelacyjnych baz danych jest najbardziej odpowiedni dla systemów e-commerce. Dodatkowo wyniki analizy zostaną zestawione z bazą relacyjną, aby porównać różnice w ich działaniu przy liczbie sięgającej 1000 jednoczesnych użytkowników. W artykule postawiono tezę badawczą „MongoDB jest najlepszym wyborem w zastosowaniach e-commerce.” Dodatkowo zweryfikowane zostaną następujące pomocnicze pytania badawcze:

- Czy baza MongoDB jest bardziej przystosowana do zastosowań e-commerce niż Cassandra?
- Czy nierelacyjne bazy danych są wydajniejsze niż relacyjne w systemach e-commerce?
- Czy wybrane bazy nierelacyjne mają większe wsparcie społeczności niż wybrany system relacyjny?

2. Przegląd literatury

W wielu artykułach poruszona została kwestia wydajności i zastosowań różnych baz danych.

W artykule [2] przedstawiona została struktura i składniki baz danych dla systemów e-commerce w świecie rzeczywistym. Opisano szczegółowy łańcuch wartości e-commerce oraz wymagania dotyczące danych. Na potrzeby artykułu przeanalizowano łańcuch składający się z ośmiu procesów biznesowych. Następnie przedstawiono schemat bazy danych dla e-commerce w notacji UML. Analiza projektu skupiła się na podstawach sprawnego działania systemów przetwarzania transakcji e-commerce. Wynika z niego, że z punktu widzenia projektowania baz danych interesującym problemem badawczym jest to, jakie struktury baz danych są potrzebne, aby najefektywniej wspierać dostosowanie i personalizację. Na przykład, jak i jakie dane trzeba przechwycić, aby zbudować hurtownię internetową do personalizacji, a następnie, jak komunikować się z użytkownikami systemów**.**

W badaniu [3], aby zbudować udany i skalowalny sklep internetowy użyto dokumentowej bazy danych do włączenia przepływu pracy e-commerce. Projekt i strategia pracy e-commerce została oparta o MongoDB, która jest szeroko stosowaną bazą danych NoSQL do przetwarzania aplikacji biznesowych na dużą skalę. Autorzy zaczęli od analizy wydajności MongoDB, następnie opracowali schemat bazy danych dla systemu e-commerce. Przeanalizowali oni także elementy, które są najbardziej istotne podczas wymiany informacji pomiędzy systemem a klientem. Zorientowana na dokumenty baza danych NoSQL, taka jak MongoDB, ma wiele zalet w porównaniu z relacyjnymi bazami danych, np. może zawierać tysiące rekordów danych z odpowiednim elastycznym modelem. Również tego rodzaju modele danych, takie jak modele projektowania schematów, są przydatne do dalszego rozwoju nowych systemów przyjętych zasobów.

W pracy [4] autorzy opracowali badanie porównawcze, w którym oceniają wydajność dwóch bardzo rozpowszechnionych w tej dziedzinie baz danych: MySQL jako relacyjnej bazy danych oraz MongoDB jako bazy NoSQL. Do realizacji tej konfrontacji autorzy wykorzystali Yahoo! CloudServing Benchmark - YCSB. Miało to na celu udzielenie niezbędnej pomocy i wsparcia zainteresowanym podmiotom z branży Big Data i Cloud Computing w podjęciu decyzji o wyborze najlepszego rozwiązania bazodanowego, jakie powinno zostać przyjęte w ich firmach. Po analizie wyników eksperymentalnych autorzy doszli do wniosku, że wybór przyjmowanego rozwiązania zależy od zestawu parametrów takich jak wielkość środowiska, charakter i szacowana częstotliwość wykonywanych operacji.

W artykule [5] sprawdzono piętnaście kategorii baz danych NoSQL, aby poznać cechy każdej z nich. Zaproponowano pewne zasady i przykłady, które pozwolą wybrać odpowiednią bazę danych NoSQL dla różnych branż. Autorzy opisali cechy baz danych NoSQL. Na tej podstawie starali się wybrać najbardziej efektywny dla każdego rodzaju system. Podsumowując, jeśli firma porzuca relacyjną bazę danych i przechodzi na bazę NoSQL, musi rozważyć cechy danych firmy, aby znaleźć odpowiednią bazę danych. Dane transakcyjne branży e-commerce często muszą być powiązane, odpowiednią kategorią NoSQL jest szeroka rodzina kolumn, a Apache HBase jest dobrym wyborem.

Przeprowadzone badania w artykule [6] skupiły się na identyfikacji najbardziej wydajnego systemu przechowywania danych pod względem czasu odpowiedzi, porównując dwa najbardziej reprezentatywne systemy bazodanowe z dwóch kategorii (NoSQL i relacyjnych), tj. MongoDB oraz PostgreSQL. Ocena oparta jest na rzeczywistych, biznesowych scenariuszach i ich kolejnych zapytaniach jak również infrastruktur bazowych i kończy się na potwierdzeniu wyższości PostgreSQL. W szczególności baza ta jest czterokrotnie szybsza pod względem czasu odpowiedzi w większości przypadków. Zaobserwowano również, że średni czas odpowiedzi jest zmniejszony o połowę przy użyciu indeksów niemal we wszystkich przypadkach, podczas gdy redukcja jest znacznie mniejsza w przypadku PostgreSQL.

W badaniu [7] przedstawiono przypadek, który opierał się na dwóch stronach internetowych e-commerce. W trakcie badania połączono dwie różne bazy danych przy użyciu usług internetowych do pobierania danych. Wszystkie komponenty scalające były przechowane na usługach chmurowych. Każdy proces wsadowy w systemie synchronizacji odbywał się online i automatycznie wykorzystywał narzędzie cron. W opisywanym badaniu przedstawiono wyniki testów integracji danych na usługach webowych oraz pomiar czasu wykonania systemu synchronizacji. Po przetestowaniu dwóch użytych serwisów internetowych okazało się, że wszystkie przeszły test integracji danych. Następnie dokonano pomiarów na systemie synchronizacji patrząc na wyświetlany czas wykonania. Wykorzystano od 100 do 1000 danych i różne warunki. Uzyskano czas wykonania od 4 do 5 sekund dla każdej danej w pomiarze. Wyniki badań wskazały, że integracja danych z dwóch różnych baz danych z wykorzystaniem usług internetowych i systemów synchronizacji jest znośna i odpowiednia dla ilości danych poniżej 10000 rekordów przy stanie bazy już wypełnionej.

W artykule [8] przedstawiono taksonomię dotyczącą systemów NoSQL. Używając tej perspektywy, dokonano porównań różnych systemów NoSQL używając wielu aspektów, w tym architektury systemu, modelu danych, języka zapytań, API klienta, skalowalności i dostępności. Pogrupowano obecne systemy NoSQL ze względu na ich model danych: klucz-wartość, rodzina kolumn, dokument, graf, natywny XML. Opisano również scenariusze zastosowań dla każdej kategorii, aby pomóc w wyborze odpowiedniego systemu NoSQL dla danej aplikacji.

W pracy [9] autorzy zaproponowali koncepcję systemu e-commerce, który w procesie tworzenia rekomendacji, gromadzi i wykorzystuje dane pochodzące z profili społecznościowych swoich użytkowników. Takie podejście do modelowania architektury zostało opracowane w ramach projektu aplikacji internetowej typu mashup, która integruje się z API Facebooka. Opisane zostało, jakie dane można pozyskać z Facebooka, zaproponowano sposób ich przechowywania oraz wska-

zano, w jaki sposób informacje z profilu użytkownika mogą poprawić efektywność systemu rekomendacji handlu elektronicznego.

W badaniach z artykułu [10] omówiono różne aspekty MongoDB. Przedstawiono zalety korzystania z systemu, a następnie dokonano zestawienia najpopularniejszych zagadnień wykorzystujących tą bazę danych. Przeanalizowano główne problemy systemu takie jak konstrukcja schematu bazy danych czy też brak niezawodności. Wyniki tego badania otworzyły nowe drogi dla przyszłych badań nad wydajnością dostępu do danych kiedy występują hotspoty w danych czyli sektory, w których występuje wiele żądań dostępu w określonym momencie. Założono, że dostęp do wszystkich danych będzie uzyskiwany przy użyciu tych samych wzorców.

W artykule [11] dokonano porównania cech wyróżniających narzędzia relacyjnych i nierelacyjnych bazy danych. Zestawiono kluczowe aspekty największych relacyjnych systemów baz danych takich jak Oracle i MySQL. W kwestii nierelacyjnych systemów dokonano klasyfikacji na podstawie sposobu organizacji danych. Czynnikami podlegającymi porównaniu były np.: przepustowość, skalowalność, łatwości manipulacji danymi czy duplikacja danych.

Alternatywne wykorzystanie nierelacyjnej bazy Cassandra przedstawiono w [12]. Badania skupiły się na utrzymaniu i magazynowaniu danych strumieniowych z różnych platform.

W artykule [13] przedstawiono zintegrowany 8-procesowy łańcuch wartości potrzebny systemowi e-commerce i związane z nim dane na każdym etapie łańcucha wartości.

Przegląd literatury okazał się pomocny do wyboru metod badawczych, sposobu przeprowadzenia badań i wizualizacji wyników.

3. Wykorzystane systemy bazodanowe

3.1. Baza MongoDB

Baza MongoDB [14] jest to stworzony w języku C++ magazyn danych NoSQL. Główną strukturą danych są kolekcje, które zawierają dokumenty. Dokumenty posiadają unikalny klucz specjalny o nazwie _id, który najczęściej jest typu "ObjectId", a którego używa się do jednoznacznej identyfikacji dokumentów. Dokumenty MongoDB poddawane są serializacji jako obiekty JSON i przechowywane wewnętrznie przy użyciu binarnego kodowania BSON. Najpopularniejszymi formatami półplików, których styl przyjmują dokumenty to: XML, JSON, YAML i CSV. Do składowania danych wykorzystuje się zagnieżdżanie dokumentów lub referencje do innych dokumentów na żądanie użytkownika. **S**ystem ten jest aktualnie najpopularniejszym systemem wybieranym do rozwiązań NoSQL, ponieważ charakteryzuje się dużą skalowalnością i elastycznością w zakresie przechowywania i przetwarzania danych. W chwili obecnej bazę MongoDB najczęściej uruchamia się w chmurze obliczeniowej ale do dyspozycji użytkowników są również narzędzia takie jak aplikacja kliencka Compass.

3.2. Baza Apache Cassandra.

Apache Cassandra[15] jest to produkt open source stworzony przez firmę Facebook. Podobnie jak MongoDB jest to rozwiązanie NoSQL, jednak Cassandre wyróżnia to, że jest rozproszona i przeznaczona do pracy na dużej ilości serwerów. Cassandra jest kolumnową bazą danych, zaprojektowaną do gromadzenia dużej ilości danych. Posiada odpowiednie narzędzia do utrzymania klasy niezawodności takie jak silnik BigTable oraz Dynamo wspomagające przechowanie i replikację. Główne zalety tej bazy danych to: pełna replikacja bazy danych, niskie opóźnienia przy wysokiej dostępności, duża elastyczność schematu czy rozwój klastra. Do wykonywania zapytań wykorzystywany jest wewnętrzny język CQL (Cassandra Query Language). Do wad tego rozwiązania można zaliczyć takie aspekty systemu jak: problemy z zarządzaniem pamięcią JVM oraz brak obsługi agregatów. Organizacja danych opiera się o klaster węzłów.

3.3. Baza PostgreSQL

Baza PostgreSQL to jedyna relacyjna baza w zestawieniu systemów bazodanowych wybranych przez autorów pracy. Jest to system obiektowo-relacyjny będący produktem open source [16]. PostgreSQL zapewnia wysoką ochronę integralności danych i odporność na błędy. Podobnie jak w większości relacyjnych baz danych, podstawową strukturą do przechowywania danych jest tabela. Do manipulacji danymi służy język SQL, jednak można również używać funkcji i procedur w języku PL/pgSQL. Integralność danych zapewniają klucze główne i obce definiowane na poziomie tabeli. Dostępne są trzy rodzaje replikacji danych: synchroniczna, asynchroniczna i logiczna. Baza posiada dobrze rozwinięty system kontroli dostępu, szyfrowanie i uwierzytelnianie wieloczynnikowe. Na chwilę obecną najbardziej znane społeczności wady PostgreSQL to porównywalnie niska prędkość odczytu oraz brak dostępności na wszystkich hostach.

4. Plan badań

Do badań utworzone zostały trzy niezależne bazy danych w wybranych systemach. W celu weryfikacji postawionej tezy i pytań badawczych stworzona została aplikacja testowa w środowisku .NET, która symuluje działanie systemu e-commerce z wykorzystaniem stworzonych baz danych. Aplikacja pozwala na rejestrację użytkowników oraz zarządzanie produktami i zamówieniami klientów. Zastosowany został wzorzec projektowy repozytorium, aby w zależności od potrzeb wykorzystać właściwą bazę danych. W przypadku bazy PostgreSQL oprócz automatycznie indeksowanych kluczy głównych dodany został unikalny indeks na kolumnie email w tabeli użytkownika. Natomiast dla bazy Cassandra tabele zostały zaprojektowane biorąc pod uwagę zapytania wymagane przez aplikację. Oznacza to, że jeżeli potrzebne były dane według id oraz nazwy, utworzono dwie tabele powielające dane. W pierwszej tabeli kluczem partycji było pole id, a w drugiej pole nazwa. Zastosowany został także indeks pomocniczy dla tabeli

zamówień na kolumnie ze statusem. Do wykonywania operacji na bazie bezpośrednio z aplikacji wykorzystano wspierane przez twórców biblioteki: Entity Framework Core, DataStax C# Driver i MongoDB C# Driver. W operacjach korzystano ze składni zapytań LINQ [17], która jest tłumaczona na odpowiednie zapytania na bazie, przez wykorzystane biblioteki. Za pomocą biblioteki BenchmarkDotNet [18] przeprowadzone zostały następujące testy wydajności systemów nierelacyjnych dla następujących operacji występujących w systemie e-commerce:

- dodawanie użytkownika, produktu i zamówienia,
- wyszukiwanie użytkowników, zamówień i produktów,
- usuwanie użytkownika, produktu i zamówienia,
- aktualizacja produktu i zamówienia.

Dla zapewnienia wiarygodności testów zostały one powtórzone 1000 razy dla każdego systemu bazodanowego.

Do przedstawienia poziomu wsparcia społeczności zostały wykorzystane platformy GitHub i Stack Overflow. Techniką pozwalającą określić zainteresowanie daną tematyką jest analiza popularności polegająca na zliczeniu rozpoczętych dyskusji na temat systemów bazodanowych na forum Stack Overflow oraz liczby gwiazdek repozytorium na GitHub.

4.1. Struktury badanych danych

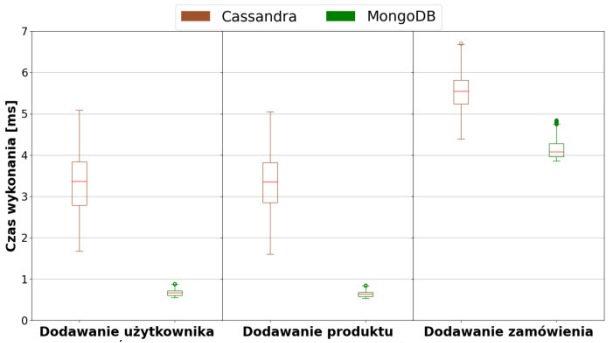
Struktura danych opiera się na trzech głównych encjach: użytkownika, produktu i zamówienia. W przypadku bazy PostgreSQL, aby nie duplikować danych zastosowano normalizację. Powstały w ten sposób następujące tabele: adres, który jest w relacji z użytkownikiem i zamówieniem, kategoria produktu związana z tabelą produktów oraz szczegóły płatności, metoda płatności, status płatności, status zamówienia, sposób dostawy, będące w relacji z tabelą zamówienia. W przypadku użytych baz NoSQL zastosowano denormalizację. Wymagane dane umieszczono bezpośrednio w głównych encjach.

Operacja zwracania zamówień zawiera również informacje o powiązanych z nimi produktami. Ponadto usunięcie użytkownika lub produktu nie wpływa na rekordy w tabeli z zamówieniami.

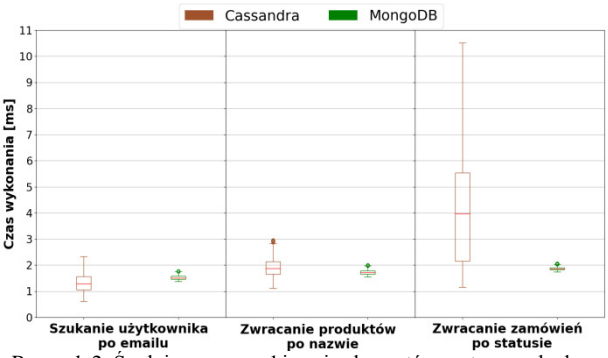
5. Wyniki badań

5.1. Analiza wyników wydajności

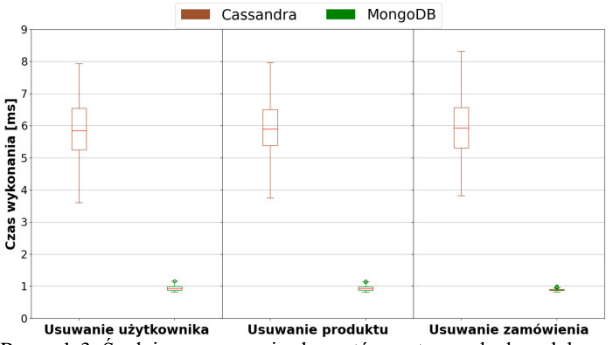
Na Rysunkach 1-4 przedstawiono wykresy pudełkowe ilustrujące różnice w czasach wykonywania operacji na bazach nierelacyjnych za pośrednictwem aplikacji testowej. W większości przypadków występuje sytuacja, gdzie MongoDB okazuje się szybsza od Cassandry. Różnice średnich czasów wynoszą od około milisekundy do niecałych dwóch milisekund w przypadku operacji dodawania. Największa nieregularność występuje przy zapytaniach wyszukujących. Na uwagę zasługuje operacja zwracania produktów po nazwie gdzie Cassandra okazała się lepsza o 0.17 ms. Warto również nadmienić, że największa różnica w czasie występuje podczas operacji zwracania wszystkich zamówień i wynosi około 22 ms. W przypadku operacji usuwania MongoDB jest o około 5 ms szybsze. Podczas aktualizacji Cassandra, wypada około 5-6 ms gorzej niż MongoDB.



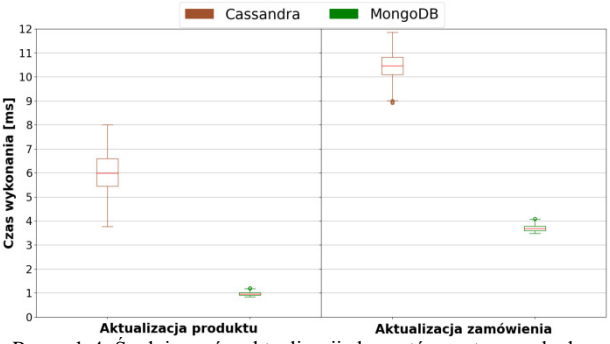
Rysunek 1: Średni czas dodawania nowych elementów systemu do badanych baz nierelacyjnych.



Rysunek 2: Średni czas wyszukiwania elementów systemu w badanych bazach nierelacyjnych.



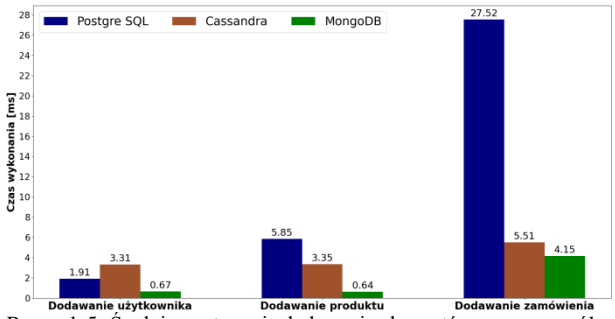
Rysunek 3: Średni czas usuwania elementów systemu z badanych baz nierelacyjnych.



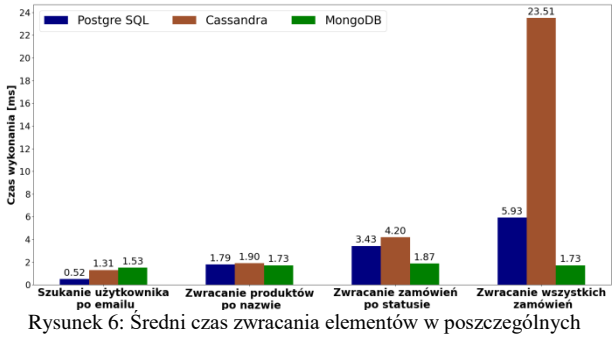
Rysunek 4: Średni czasów aktualizacji elementów systemu w badanych bazach nierelacyjnych.

Na Rysunkach 4-8 zobrazowano zestawienie średnich czasów wszystkich wybranych systemów bazodanowych. Baza PostgreSQL nie wypada lepiej niż MongoDB za wyjątkiem operacji wyszukania użytkownika

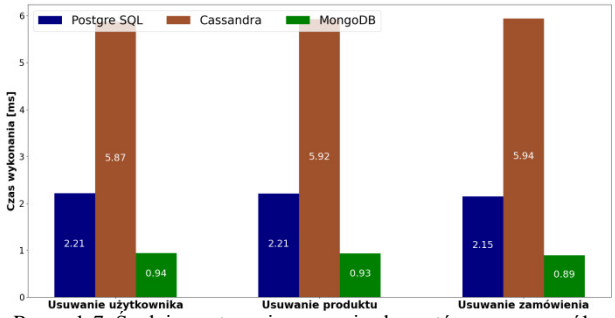
przy pomocy adresu email. Natomiast w większej liczbie testów system ten okazał się być szybszy niż Cassandra. Największe różnice w średnim czasie między systemami nierelacyjnymi a PostgreSQL występują w przypadku operacji na zamówieniach takich jak dodawanie i aktualizacja.



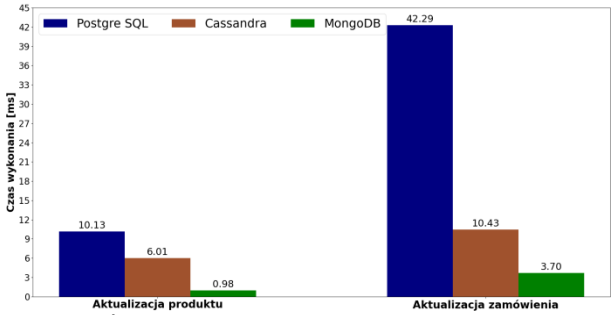
Rysunek 5: Średni czas trwania dodawania elementów w poszczególnych bazach danych.



Rysunek 6: Średni czas zwracania elementów w poszczególnych bazach danych.



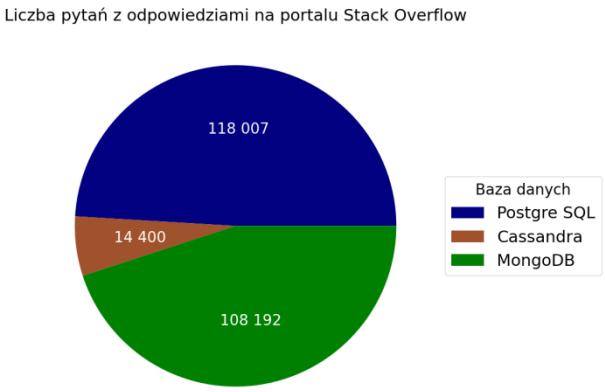
Rysunek 7: Średni czas trwania usuwania elementów w poszczególnych bazach danych.



Rysunek 8: Średni czas trwania aktualizacji elementów w poszczególnych bazach danych.

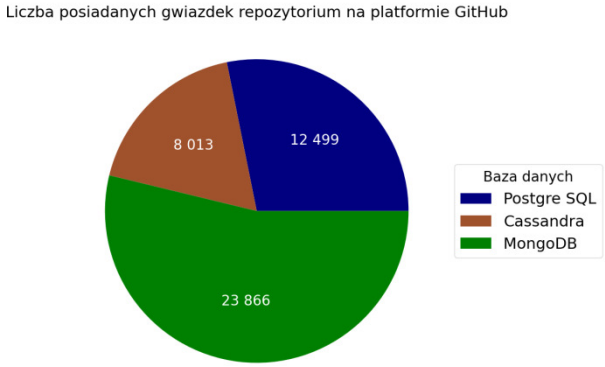
5.2. Analiza wsparcia społeczności

Na forum Stack Overflow [19], które jest największą społecznością przeznaczoną do dzielenia się wiedzą przez deweloperów, sprawdzono dla każdej badanej bazy liczbę pytań, które posiadają odpowiedź od użytkowników na temat danej technologii. Na Rysunku 9 można zauważyć, że najwięcej pytań, na które udzielono odpowiedzi posiada relacyjna baza PostgreSQL – ok. 118 tys. Niewiele mniej jest w przypadku MongoDB – ok. 108 tys. Natomiast zdecydowanie najmniej posiada Cassandra: ok. 14 tys.



Rysunek 9: Zestawienie liczby pytań z odpowiedziami na Stack Overflow wybranych systemów bazodanowych na dzień 08.06.2023r.

Z kolei w serwisie GitHub [20] sprawdzono posiadaną liczbę gwiazdek przez oficjalne repozytoria. Stanowią one wyraz uznania społeczności i wpływają na ranking repozytorium na stronie. Na przedstawionym Rysunku 10 zdecydowanym faworytem jest baza MongoDB, która posiada ok. 24 tys. gwiazdek. Drugie miejsce zajmuje PostgreSQL, które posiada ich ok. 12 tys. Na ostatnim miejscu znajduje się Cassandra z 8 tys. gwiazdek.



Rysunek 10: Zestawienie liczby gwiazdek repozytorium na platformie GitHub dla wybranych systemów bazodanowych na dzień 08.06.2023r.

6. Wnioski

Wykresy na rysunkach 1-8 wizualizujące przeprowadzone badania jednoznacznie potwierdzają postawioną tezę. MongoDB jest najlepiej przystosowaną bazą danych do systemów e-commerce. Baza ta jest najszybsza w 11 na 12 badanych operacjach. Większość wybranych do badań operacji wykonuje średnio poniżej 1 ms, a najwolniej pojedynczą operację wykonuje nieco powyżej 4ms. To doskonały wynik przy 1000 prób.

Różnica w wydajności pomiędzy systemami relacyjnymi a nierelacyjnymi najbardziej zauważalna jest

w przypadku operacji aktualizujących. W zależności od operacji można stwierdzić, że PostgreSQL cechuje podobna wydajność do Cassandry. Natomiast PotgreSQL okazało się lepsze niż MongoDB tylko w jednej z 12 badanych operacji. Społeczność udzielająca się na wybranych platformach prezentuje porównywalną aktywność w przypadku wybranych do badań baz nierelacyjnych i relacyjnych. Dzięki zebranym danym można stwierdzić, że wsparcie jest na bardzo podobnym poziomie. Warto nadmienić jednak, że bazy nierelacyjne są stosunkowo młode w porównaniu do relacyjnych. Porównywalny poziom dyskusji na najpopularniejszym forum świadczy o wysokiej dynamice rozwoju tych rozwiązań.

Badania można kontynuować pod kątem bezpieczeństwa wybranych baz danych w systemach e-commerce. Bezpieczeństwo danych jest ważnym aspektem przy projektowaniu systemów więc może to być ciekawy kierunek. Można również porównać skalowalność przy jednoczesnym korzystaniu przez wielu użytkowników.

Literatura

[1] A. Molla, P.S. Licker,E-commerce systems success: An attempt to extend and respecify the Delone and MacLean model of IS success, Journal of Electronic Commerce Research 2 (2001) 131-141.

[2] I. Y. Song, K. Y. Whang, Database design for real-world e-commerce systems, EEE Data Eng. Bull. 23.1 (2000) 23-28.

[3] D. Ramesh, E. Khosla, S. N. Bhukya, Inclusion of e-commerce workflow with NoSQL DBMS: MongoDB document store, IEEE international conference on computational intelligence and computing research(ICCIC) (2016) 1-5.

[4] H. Matallah, G. Belalem, K. Bouamrane, Comparative study between the MySQL relational database and the MongoDB NoSQL database, International Journal of Software Science and Computational Intelligence (IJSSCI) 13(3) (2021) 38-63.

[5] J. K. Chen, W. Z. Lee, An Introduction of NoSQL Databases based on their categories and application industries, Algorithms 12(5) (2019) 1-16, https://doi.org/10.3390/a12050106.

[6] A. Makris, K. Tserpes, G. Spiliopoulos, D. Anagnostopoulos, Performance Evaluation of MongoDB and PostgreSQL for spatio-temporal Data, EDBT/ICDT Workshops2019 Joint Conference Workshops, Lisbon, Portugal (2019) 1-8.

[7] A. J. Maulidin, F. Renaldi, F. R. Umbara, Online Integration of SQL and No-SQL Databases using RestAPIs: A Case on 2 furniture e-Commerce Sites, 2020 3rd International Conference on Computer and Informatics Engineering (IC2IE) (2020) 261-266.

[8] V. N. Gudivada, S. Jothilakshmi, D. Rao, Data management issues in big data applications, ALLDATA 15 (2015) 16-21.

[9] D. Fijałkowski, R. Zatoka, An architecture of a Web recommender system using social network user profiles for e-commerce, 2011 Federated Conference on Computer Science and Information Systems (FedCSIS) (2011) 287-290.

[10] A. Chauhan, A Review on Various Aspects of MongoDb Databases, International Journal of Engineering Research & Technology (IJERT) 8(5) (2019) 90-92.

[11] N. Jatana, S. Puri, M. Ahuja, I. Kathuria, D. Gosain, A survey and comparison of relational and non-relational database, International Journal of Engineering Research & Technology 1(6) (2012) 1-5.

[12] A. Kumar, G. Vijaya. Streaming data analysis using apache cassandra and zeppelin. IJISET-International Journal of Innovative Science, Engineering & Technology 3 (2016) 8-15.

[13] SONG, Il-Yeol; WHANG, Kyu-Young. Database design for real-world e-commerce systems. IEEE Data Eng. Bull. 23.1 (2000) 23-28.

[14] What is MongoDB - MongoDB documentation, https://www.mongodb.com/docs/manual, [08.06.2023].

[15] Apache Cassandra Documentation – overview, http://cassandra.apache.org/doc/latest/cassandra/architecture/overview.html, [08.06.2023].

[16] PostgreSQL - About, https://www.postgresql.org/about, [08.06.2023].

[17] LINQ – overview https://learn.microsoft.com/en-us/dotnet/csharp/linq/ [26.06.2023].

[18] BenchmarkDotNet overview, https://benchmarkdotnet.org/, [26.06.2023].

[19] StackOverflow – forum programistyczne, https://stackoverflow.com, [08.06.2023].

[20] GitHub – serwis internetowy, https://github.com, [08.06.2023].

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 28 (2023) 279–284
Received: 17 June 2023
Accepted: 3 July 2023

Analysis of how universal design principles impact on the perception of virtual museum interfaces

Analiza wpływu zastosowania projektowania uniwersalnego na postrzeganie interfejsów wirtualnych muzeów

Dawid Nicpoń, Weronika Wach*, Maria Skublewska-Paszkowska

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

Abstract
The research presented in this paper aims to analyse the impact of using universal design in implementation of virtual museum interfaces to user perception. The experiment uses collaboration with the Museum of the History of the City of Lublin to create the web application which presents virtual exhibits. The application was based on the React JavaScript framework, which enabled the creation of a basic web interface and the React 360 to generate the three-dimensional view. During research, the implemented application was compared with the muzeumpuck.wkraj.pl website, which does not conform to the universal design principles. The main research method consisted of eye-tracking technology and the LUT survey. The analysis of the results shows the interface which follows the principles of universal design was easier to navigate. The time of searching for specific elements on that interface was shorter than in other applications as well.
Keywords: universal design; virtual museum; eye tracking; LUT survey

Streszczenie
Badania przedstawione w niniejszym artykule mają na celu analizę wpływu zastosowania projektowania uniwersalnego w implementacji wirtualnych interfejsów muzealnych na percepcję użytkowników. W celu stworzenia aplikacji internetowej, która prezentuje wirtualne eksponaty, nawiązano współpracę z Muzeum Historii Miasta Lublina. Aplikacja została oparta na javascriptowym szkielecie programistycznym React, który umożliwił stworzenie podstawowego interfejsu webowego oraz React 360 do wygenerowania trójwymiarowego widoku. W trakcie badań porównano zaimplementowaną aplikację z serwisem muzeumpuck.wkraj.pl, który nie spełnia zasad projektowania uniwersalnego. Główną metodą badawczą była technologia eye-trackingowa oraz lista kontrolna LUT. Z analizy wyników wynika, że interfejs zgodny z zasadami projektowania uniwersalnego był łatwiejszy w nawigacji. Czas wyszukiwania poszczególnych elementów na tym interfejsie był również krótszy niż w innej aplikacji.
Słowa kluczowe: projektowanie uniwersalne; wirtualne muzeum; eye tracking; ankieta LUT

*Corresponding author
Email address: weronika.mroz.it@gmail.com (W. Wach)

1. Introduction

At present, the majority of cultural facilities are working on expanding their virtual offer by adding interactive elements to the website interfaces. One of such elements is a virtual walk which allows you to get interactive with the exhibitions and attractions of the museum. Users willingly take advantage of the possibility of three-dimensional navigation around the premises of the facility and finding interesting facts about the exhibits [1].

Each interface differs in visual and implementation features. Currently, there is a large pool of technologies that enable the creation of virtual views. Depending on the needs, such solutions can offer a number of additional functionalities and facilitations, such as changing the contrast, using of an virtual guide or zooming in information about the exhibits after hovering over them.

2. The purpose of the work and hypotheses

This study aims to analyse the impact of applying the principles of Universal Design (UD) in the implementation of a virtual museum interface on the user's perception. The research is based on checking the user's reaction time and the ease of navigating the application interface. Based on the literature review, the following hypotheses were put forward:

H1: The interface designed in accordance with the UD pattern, improves the user's perception of the quality of the virtual museum interface.

H2: The perception of interface elements differs depending on gender.

3. Literature review

Universal Design is implemented in many areas of life, including in the GUI design. Many studies can be found on the topics covered in this study.

Articles [1-3] explored how to create a virtual museum application using different technologies. The authors are searching for the best solutions that will allow for a uniform architecture of the application, and at the same time the largest and most realistic possibilities of interaction with the website and virtual exhibits.

The aim of the research presented in [4-7] was to examine the perception of exhibits by visitors. The

authors tried to identify a general pattern of interpretation of the images. As a result of the experiments, it was shown that despite the inevitable variability between subjects, there were common basic patterns of fixation of gaze, but due to the wide variety of results, the researchers failed to emerge a general pattern in which works of art are perceived.

In the study [8], the author discuss the importance of using UD in the process of implementing solutions. She pays attention to the impact of UD principles on the accessibility and usability of applications.

Articles [9-10] discuss user expectations for virtual museum interfaces. The authors focus on adapting the interface in such a way that it can reflect the user experience in the real world as faithfully as possible.

Based on the literature review conducted, no studies were found on the impact of the use of UD in virtual museum interfaces on users' perceptions and whether perceptions of interfaces in virtual museums differ by gender.

4. The interface project and implementation

The experiment described in this paper required the creation of a virtual museum interface, which was created in cooperation with the local museum - the Museum of the History of the City of Lublin. The interface was implemented using the React framework [11], which allowed the creation of a main website on which a three-dimensional application was embedded.

The second element of the interface is a three-dimensional application, which was created using the React 360 framework [12]. The website uses panoramic photos to create a 360° view which is the main background of the application. The structure of the view is created on the basis of assigning coordinates for the places of occurrence of given elements, such as, for example, arrows moving around the view or additional close-ups of elements imitating the zoom. The Figure 1 shows a snippet of code that creates the home page of a three-dimensional view.

```
<html>
  <head>
    <title>Muzeum Historii Miasta Lublina</title>
    <style>body { margin-top: 20; }</style>
    <link rel="stylesheet" href="style.css">
    <meta name="viewport" content="width=device-width, initial-scale=1, user-scalable=no">
  </head>
  <body>
    <div id="container">
      <header class="site-header">
        <div class="container">
            <div class="site-header-inner">
                <nav id="topnav">
                    <a id="logo" class="nav-link" href="home.html">Muzeum Historii Miasta Lublina</a>

                    <a class="nav-link" href="opis.html">Opis Muzeum</a>
                    <a class="nav-link" href="historia.html">Historia Muzeum</a>
                    <a class="nav-link" href="wizyta.html">Wizyta</a>
                    <a class="nav-link" href="index.html">Wirtualna wycieczka</a>
                    <a class="nav-link" href="kontakt.html">Kontakt</a>
                    <a class="nav-link" href="#"></a>
                </nav>
            </div>
        </div>
    </header>
    </div>
    <script src="./client.bundle?platform=vr"></script>
    <script>
      React360.init(
        'index.bundle?platform=vr&dev=true',
        document.getElementById('container'),
        {
          assetRoot: 'static_assets/',
        }
      );
    </script>
  </body>
</html>
```

Figure 1: Implementation code of main page of a three-dimensional view.

The final appearance of the 3D application is shown in Figure 2.

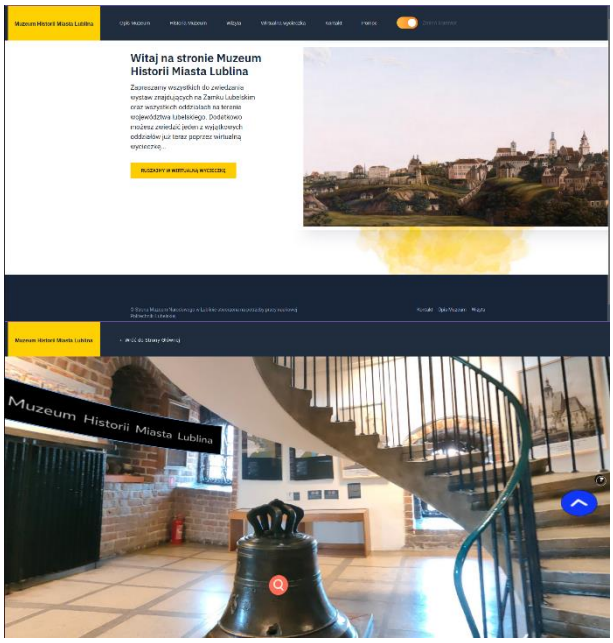


Figure 2: The final view of a three-dimensional application.

5. Research methods

The experiment was conducted based on two methodologies:
- examination using an eye-tracking device;
- examination using the Lublin University of Technology (LUT) survey [13].

Each of the methods required the use of a separate research scenario.

5.1. Research groups

Twelve participants, including 5 men and 7 women with experience in using websites took part in the study. They, regardless of gender, were asked to complete the given tasks according to the scenarios in the eye-tracker study and to evaluate the applications using the LUT survey.

5.2. Eye-tracking study

The first stage of the experiment involved the eye-tracker equipment [14]. The test stand consisted of a monitor and a stationary eye-tracking device attached to it with the specifications specified in Table 1.

Table 1: Eye-tracker specifications

Accuracy of the viewing angle	0.5 - 1.0 degrees
Sampling rate	60 Hz or 150 Hz
Calibration	5- or 9-point
Permissible head movement	35 cm (horizontal) x 22 cm (vertical)
Depth of head movement	±15 cm
Physical parameters (dimensions)	235 x 45 x 47 mm

The participants were divided by gender. Users were asked to perform specific actions on the user interface of the virtual museum website in accordance with re-

search scenarios. For this purpose, each of the existing websites [15] and created for the purpose of the study, was tested in two variants.

The following were tested:

- time to perform tasks from the research scenario;
- interface elements on which the eyes of the subjects were focused.

A single scenario included tasks such as entering search data and finding an element on a page. An exemplary usability test scenario is presented in Table 2.

Table 2: Sample research scenario

Name: Analysis of the speed of locating interface elements.		
Aim of the research: Verification of the impact of the arrangement of elements in the user interface on the speed of locating.		
Initial conditions: Presenting the user with successive views.		
Postconditions: The data collected from the eye-tracker was saved after the completion of the scenario.		
Study participants: 12		
List of tasks:		
No.	**Description**	**Result**
1	Launching the interface	The user launches the interface
2	Locate the virtual walk view button	The user finds the button and presses it
3	Locating object from the attached photo	The user moves around the view and finds the designated object
4	Locating an icon with arrow	The user moves view and finds icon
5	Exit virtual walk view	The user locates the exit button and leaves the view

5.3. Research using the LUT Survey

The next stage was to conduct a study using the LUT questionnaire [14]. Users assessed the quality of the website interface on the basis of a specially prepared survey. For the purpose of the study, 4 areas consisting of 8 sub-areas were selected. The questions were rated on a scale of 1 to 5, with 1 being the worst and 5 being the best. Each participant was asked to complete a questionnaire after checking the website. Aspects such as the ease of finding elements on the website, ease of navigation between sections and the processes that must be performed on the website in order to obtain the appropriate result in the form of receiving specific information on the website were assessed. The exact content of the LUT questionnaire is presented in Table 3.

Table 3: Scope of researched areas – the LUT survey, developed on the basis of [14]

Area	Sub-area	Question
Navigation and structure	Ease of navigation	Is access to all sections of the application easy and intuitive?
		Is access to all the functions of the application easy and intuitive?
	Information structure	Is the structure of the information well thought out?
		Is the information structure consistent?
		Is the structure of the information understandable to the user?
Messages, feedback, user support	Feedback and help	Is the help content available to the average user?
		Is the help content understandable to the average user?
		Are the presented hints or solutions to problems possible to perform by an ordinary user?
Application interface	Layout	Is the layout legible?
		Is the layout adapted to different resolutions?
		Is the layout adapted to mobile devices?
		Is the layout consistent?
		Does the layout support the implementation of tasks?
	Choice of colours	Is the contrast between the text and the background adequate?
		Does the choice of colours allow the use of the application by people with colour vision disorders?
		Does the choice of colours allow you to use the application with the use of various types of displays?
Content of subpages	Texts	Are they understandable to the user?
	Nomenclature	Is the naming used in the app consistent?
		Is the naming used in the app understandable?
	Labels	Do the labels used in the interface provide enough information?
		Do the interface elements have the necessary labels?

Participants focused on LUT areas such as "Navigation and Structure", "Messages, Feedback, User Support", "Application Interface", "Page Content" with each question scaled from 1 to 5, with 1 being the worst score and 5 the best. Working with both services, users were expected to perform the same set of tasks.

After completing all tasks, the users received questionnaires to evaluate their experience. An example scenario of testing the service before completing the survey:

1. Determining where the user is currently located.
2. Locating individual sections of the interface: (History of the object, Help, Contact, Virtual museum).
3. Testing the buttons and links in each section.

In order to analyze the obtained results, the expert method with the use of lists and the processing of the experimental results were used.

6. Results

6.1. The eye-tracker study result

During this study, the participants performed the same scenario for the created interface and the existing interface [15]. Interaction times of the subjects and their focus on specific elements were measured. The time statistics are presented in Table 4 and Table 5, taking into account the maximum total time, the minimum total time, the average time, the standard deviation and the median for three groups: all participants, women and men.

Table 4: Test time results of the created interface compatible with UD

	Maximum total time(s)	Minimum total time (s)	Average time (s)	Standard deviation (s)	Median (s)
All	17.44	7.14	11.76	3.21	10.32
Women	16.98	8.68	11.82	3.36	9.64
Men	17.44	7.14	11.73	3.14	10.64

Table 5: Test time results of the interface incompatible with UD

	Maximum total time(s)	Minimum total time (s)	Average time (s)	Standard deviation (s)	Median (s)
All	43.61	19.44	30.41	6.95	28.77
Women	43.61	19.44	30.03	7.54	27.98
Men	41.51	24.2	30.93	5.99	29.57

Additionally, the result data is presented in boxplots in Figure 3 and Figure 4.

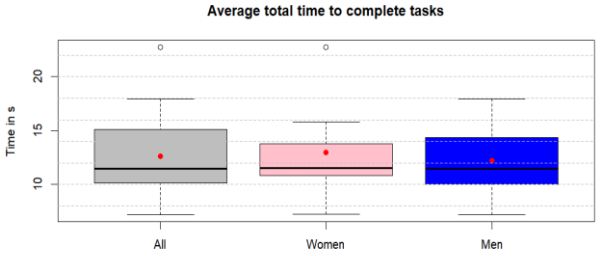


Figure 3: Average total time to complete tasks for the created interface, complying with UD.

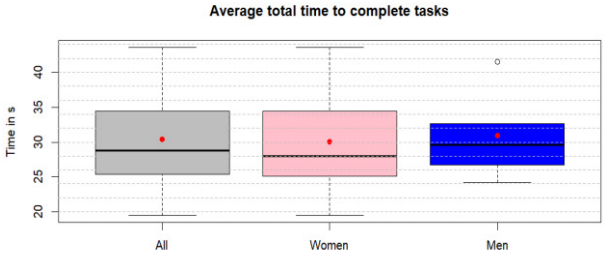
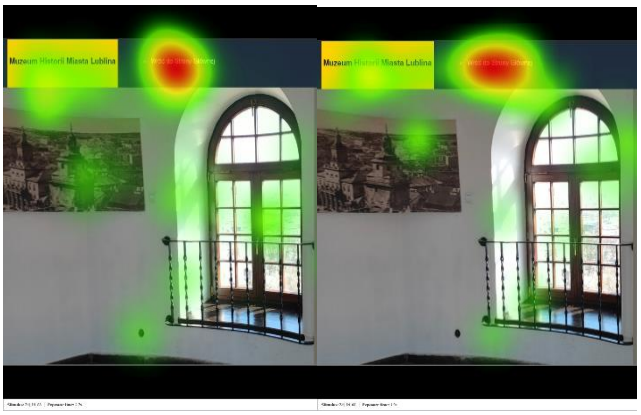


Figure 4: Average total time to complete tasks for the interface not complying with UD.

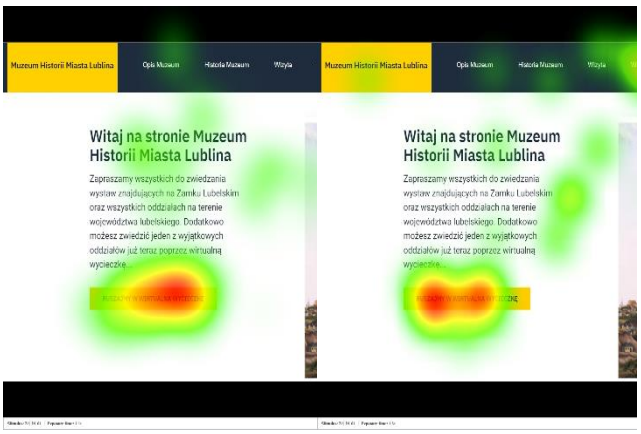
Boxplots (Fig. 3 and Fig. 4) show the dispersion of statistical data of the tested time samples, the red dot marks the average times for a given group, while the horizontal thick line shows the median time. The following conclusions were drawn on the basis of the presented time results.

The average times for each of the three groups are very similar across the same interface. The interface created in accordance with the principles of UD allows for a faster user response than non-UD interface. The average time to complete a task is 18.65s longer for the interface that does not follow the principles of UD which is more than 2 times slower than in the case of the interface that complies with these principles. Additionally, it can be seen that men's minimum task completion time is more than 3 times greater for a non-UD application. All time statistics are much lower for the application created in accordance with UD principles.



a) eye focus for men on the interface

b) eye focus for women on the interface



c) male eye focus on the virtual museum homepage

d) female eye focus on the virtual museum homepage

Figure 5: Heatmaps a), b), c), d) depending on gender of subjects.

Another aspect examined using the eye-tracker was the perception of the interface by women and men. H2 assumed that the perception would differ depending on the gender of the subjects. In order to test the hypothesis based on the study of navigating the interface that meets the principles of UD heatmaps were created that reflect the focus of the eyes on individual views of the subjects. The results are presented in Figure 5.

Based on the analysed heatmaps, it can be seen that the focus of sight differs depending on the gender of the subjects. The most frequently focused areas are marked in red, as the frequency of eye focus decreases, the colour gradually changes to green. In most of the heatmaps studied, female gaze was more diffused than male gaze.

Although there are slight differences in eye focus, they are not clear enough to be able to define on their basis whether the examined person is a man or a woman.

6.2. The LUT survey result

During the LUT survey, the participants assessed the interface features by answering questions to the given subareas on a scale of 1 to 5. Based on their assessments, the WUP index [14] was calculated, which is the average of the subareas and areas specified in Table 6.

Table 6: Calculations of the WUP indicator based on the LUT survey

Area	Subarea	Avg for interface not complying with UD		Avg for created interface, complying with UD	
Navigation and structure	Easy to navigate	3.25	3.07	5	4.903
	Information structure	2.889		4.806	
Messages, feedback, user help	Feedback and help	2.604	2.604	4.806	4.806
Application interface	Layout	2.883	2.859	4.9	4.839
	Choice of colours	2.833		4.778	
Content of subpages	Texts	2.917	3.07	4.917	4.875
	Nomenclature	3.25		4.958	
	Labels	3.042		4.75	
Total average		2.901		4.856	

Based on the obtained results (Table 6), it can be concluded that the interface that is not based on the principles of UD received definitely worse ratings from users. It caused some difficulties in using and finding the functionality. The weakest subarea is feedback and help, which turned out to be insufficient according to the users' requirements. On the other hand, the interface supporting the principles of UD received very high and fairly uniform results, which suggests that each of the sub-areas satisfies the requirements of users to a sufficient extent.

7. Discussion and summary

The purpose of this paper was to conduct a comparative analysis of two websites that present virtual exhibits. On the basis of the research methods taken into account, the authors had the opportunity to verify two formulated hypotheses. The first of them concerned the impact of UD on improving the perception of the virtual museum interface by the user, and the second was related to the influence of gender on the way of using the interface. Two methods were used in the study.

The results of the eye-tracking test showed that an interface designed in accordance with the principles of UD allowed users to complete the tasks indicated in the scenario faster, compared to an interface that did not comply with these principles. The time difference was about 61.5%. This result underscores the great importance of UD in increasing website accessibility and a positive user experience.

The LUT survey provided information on the perception of the interface of the virtual museum website by individual users. Areas such as navigation and structure, messages and feedback, application interface and page content were assessed. Participants rated these areas on a scale of 1 to 5. Also in this aspect a large impact of the presence of UD on the results obtained can be seen. The non-UD website scored significantly lower than UD website. The difference was about 67% in favour of the latter. On this basis, we can conclude that the time studies confirm the assumptions of hypothesis H1.

To verify the second gender hypothesis, an eye-tracker study was also used. In the presented graphs (Figure 3 and Figure 4) and heatmaps (Figure 5) it can be seen that the differences in the time of performing tasks and the way of perceiving the interface differ slightly depending on gender. The authors of this paper were unable to distinguish significant differences between the behaviour of users of both sexes and the time of execution of the scenarios assigned to them. It can therefore be concluded that gender does not matter in the perception of website interfaces. That means that H2 cannot be confirmed by the results of the above studies.

Bibliography

[1] P. Petridis, M. White, N. Mourkousis, F. Liarokapis, M. Sifniotis, A. Basu, C.Gatzidis, Exploring and Interacting with Virtual Museums, In Proceedings of the 33rd Annual Conference of Computer Applications and Quantitative Methods in Archaeology (2005) 73-82.

[2] U. Klentien, Development of virtual museum model for youth, International Journal of Information and Education Technology 12 (2022) 313-317.

[3] S. Jangra, G. Singh, A. Mantri, B. Sharma, Adoption of Virtual Reality in Cultural Heritage and Museum Exhibition, International Conference on Communication and Electronics Systems 7 (2022) 1631-1637.

[4] R. Q. Quiroga, C. Pedreira, How Do We See Art: An Eye-Tracker Study, Frontiers in Human Neuroscience 5 (2011) 98-106.

[5] A. Ylitalo, A. Särkkä, P. Guttorp, What we look at in paintings: A comparison between experienced and inexperienced art viewers, The Annals of Applied Statistics 10(2) (2016) 549-574.

[6] V. Yanulevskaya, J. Uijlings, E. Bruni, A. Sartori, E. Zamboni, F. Bacci, D. Melcher, N. Sebe, In the eye of the beholder: Employing statistical analysis and eye tracking for analyzing abstract paintings, Proceedings of the 20th ACM International Conference on Multimedia (2012) 349-358.

[7] R. Pierdicca, M. Paolanti, S. Naspetti, S. Mandolesi, R. Zanoli, E. Frontoni, User-centered predictive model for improving cultural heritage augmented reality applications: An HMM-based approach for eye-tracking data, Journal of Imaging 4 (2018) 101-114.

[8] D. A. Riley-Huff, Web Accessibility and Universal Design A Primer on Standards and Best Practices for Libraries, Chapter 4, Library Technology Reports 48 (2012) 29-35.

[9] M. S. B. A. Ghani, S. N. B. W. Shamsuddin, A Systematic Literature Review: User Experience (UX) Elements in Digital Application for Virtual Museum, International Journal of Advanced Trends in Computer Science and Engineering 9 (2020) 2801-2807.

[10] S. Pescarin, Museums and Virtual Museums in Europe: reaching expectations, SCIentific RESearch and Information Technology 4 (2014) 131-140.

[11] Framework React documentation with brief configuration, https://legacy.reactjs.org/docs/getting-started.html, [17.05.2023].

[12] Framework React360 documentation with brief configuration, https://github.com/facebookarchive/react-360, [17.05.2023].

[13] M. Miłosz, Ergonomia systemów informatycznych, Biblioteka Cyfrowa Politechniki Lubelskiej, Lublin (2014).

[14] Gazepoint GP3 HD Eye Tracker, https://www.gazept.com/product/gp3hd/, [23.05.2023].

[15] Museum of Puck Country, https://muzeumpuck.wkraj.pl/html5/index.php?id=86194, [14.06.2023].

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 28 (2023) 285–290
Received: 19 June 2023
Accepted: 4 July 2023

An accessibility analysis of websites of selected types of universities

Analiza dostępności stron internetowych wybranych rodzajów uczelni wyższych

Maciej Banaszak*, Mariusz Dzieńkowski

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

Abstract
The aim of the study is to analyze and evaluate the accessibility of websites of various types of selected universities in Poland, taking into account the type of the university. Accordingly, a study consisting of two stages was performed. The first stage consisted in conducting an expert analysis using a proprietary checklist to verify whether a given website contains selected accessibility features. The second stage consisted in an audit using automated tools that checked various aspects of the application associated with accessibility. Thirty websites were used as research material, each belonging to one of 6 types of higher-education institutions. The results obtained from the first part of the survey show that university websites are equipped with standard accessibility tools, regardless of which group they belong. The second part of the survey showed that the greatest discrepancies between the different types of universities occurred in the aspect of website performance.

Keywords: accessibility; usability; automated accessibility testing tools; checklist

Streszczenie
Celem artykułu jest analiza i ocena dostępności serwisów internetowych wybranych uczelni wyższych w Polsce, z uwzględnieniem rodzaju uczelni. W związku z tym zrealizowano badania składające się z dwóch etapów. Pierwszy etap polegał na przeprowadzeniu analizy eksperckiej przy wykorzystaniu autorskiej listy kontrolnej, która miała sprawdzić czy dany serwis zawiera wybrane funkcje dostępności. Drugi etap polegał na przeprowadzeniu audytu za pomocą automatycznych narzędzi, które sprawdzały różne aspekty aplikacji związane z dostępnością. Jako materiału badawczego użyto 30 serwisów, z których każdy należał do jednego z 6 rodzajów uczelni. Wyniki uzyskane z pierwszej części badania pokazują, że strony internetowe szkół wyższych są wyposażone w standardowe narzędzia dostępności, niezależnie do której grupy należą. Druga część badań wykazała, że największe rozbieżności między poszczególnymi typami uczelni wystąpiły w aspekcie wydajności serwisów.

Słowa kluczowe: dostępność; użyteczność, automatyczne narzędzia do badania dostępności; lista kontrolna

*Corresponding author
Email address: maciej.banaszak@pollub.edu.pl (M. Banaszak)

1. Wstęp

Dostępność cyfrowa (ang. digital accessibility) oznacza tworzenie serwisów internetowych, aplikacji mobilnych, multimediów i usług elektronicznych w sposób uniwersalny, tak aby były one dostępne dla jak największej grupy użytkowników [1]. Jest to szczególnie ważne przy tworzeniu stron internetowych podmiotów publicznych takich jak państwowe uczelnie wyższe.

Ze stron uczelni korzystają między innymi studenci, kandydaci na studia oraz pracownicy. Część z tych osób jest dotknięta różnego rodzaju niepełnosprawnościami, przez co osoby te mają specjalne wymagania, które są stawiane serwisom oraz aplikacjom mobilnym i internetowym [2-4]. W celu umożliwienia takim osobom wygodnego korzystania z aplikacji www kwestie dostępności, wydajności i użyteczności powinny być uwzględnione już na etapie projektowania.

Istnieje wiele sposobów na zapewnienie dostępności w aplikacjach internetowych [5]. Jednym z nich jest umożliwienie użytkownikom wyświetlania stron za pomocą różnych przeglądarek internetowych oraz urządzeń mobilnych. Kolejnym jest dostarczenie na stronach www popularnych narzędzi, które pozwalają co najmniej na dostosowanie kontrastu i zmianę wielkości czcionki. Ważne jest również to, żeby strony były wyposażone w wygodną wyszukiwarkę, narzędzie do zmiany języka oraz możliwość wyświetlenia mapy strony.

Do badania poziomu dostępności oraz wykrywania błędów związanych z dostępnością stron/aplikacji internetowych można wykorzystać automatyczne narzędzia, które z łatwością można znaleźć w Internecie. Przykładami mogą być: Accessibility Evaluator (ACE), AChecker, czy Functional Accessibility Evaluator (FAE) [6]. Oprócz określenia poziomu dostępności narzędzia te pokazują, które elementy mają problemy z danym aspektem i podpowiadają w jaki sposób można określone wady naprawić. Inną metodą oceny dostępności jest analiza ekspercka realizowana za pomocą list kontrolnych, będących zestawem pytań/stwierdzeń, dotyczących różnych aspektów systemu i ocenianych przez ekspertów.

Celem artykułu jest analiza dostępności serwisów internetowych wybranych uczelni wyższych w Polsce. Uczelnie pogrupowano według ich rodzaju oraz przea-

nalizowano i porównano poziom dostępności tych grup, z których każda składała się z pięciu serwisów www.

2. Przegląd literatury

Dostępność internetowa jest obecnie bardzo ważnym aspektem serwisów, aplikacji internetowych i mobilnych. Jej zapewnienie sprawia, że aplikacja może być stosowana przez szerokie grono użytkowników oraz osoby z niepełnosprawnościami. Podmioty publiczne tj. państwowe uczelnie wyższe są prawnie zobligowane do zapewnienia dostępności na swoich portalach internetowych. Udostępniane oprogramowanie musi posiadać nie tylko opcje dostosowania wielkości czcionki i kontrastu, ale także musi spełniać wiele innych wymogów dotyczących odpowiedniego formatowania tekstu czy obsługi za pomocą klawiatury [7].

Problematyka dostępności aplikacji internetowych była podejmowana w wielu pracach. Autorzy artykułu [8] badali poziom dostępności stron uczelni wyższych w Lublinie i porównali je z dwiema innymi uczelniami w Polsce. Do badań została wykorzystana lista kontrolna oraz automatyczne narzędzia badania dostępności. Badacze stwierdzili, że poziom dostępności serwisów uczelni wyższych w Lublinie jest niski. Powtarzające się problemy dotyczyły braku deklaracji dostępności, nieumieszczania zeskanowanych dokumentów w formie plików PDF oraz braku dostosowania serwisu do urządzeń mobilnych.

W artykule [9] również wykorzystano listę kontrolną i automatyczne narzędzia badania dostępności do wyznaczenia wskaźników ogólnego poziomu dostępności serwisów gmin województwa lubelskiego. Po przeprowadzeniu badań wskaźnik dla analizy automatycznej wyniósł 74,92%, natomiast dla eksperckiej 45,99%. W podsumowaniu autorzy zwrócili uwagę na to, że oba wskaźniki były niższe od początkowo zakładanych wartości, co pokazało, że w sferze dostępności stron internetowych jednostek publicznych województwa lubelskiego jest wiele aspektów, które wymagają poprawy.

Badaniem dostępności zajmowali się również Irańscy naukowcy. W artykule [10] zbadana została dostępność stron internetowych uniwersytetów medycznych w Iranie. Przed przeprowadzeniem badań uczelnie zaklasyfikowano do jednej z trzech grup, biorąc pod uwagę poziom ich kształcenia. Następnie wykonano testy serwisów za pomocą dwóch automatycznych narzędzi do badania dostępności: AChecker i Functional Accessibility Evaluator (FAE) [11]. Na koniec, sprawdzono istotność różnic w poziomach dostępności między poszczególnymi grupami, wykorzystując do tego celu statystyczny test Kruskala-Wallisa [12]. Okazało się, że spośród 50 badanych stron tylko dwie nie miały problemów z zapewnieniem odpowiedniego poziomu dostępności.

3. Metody badawcze

W ramach tej pracy opracowano i przeprowadzono trzyetapowy eksperyment polegający na zbadaniu dostępności stron internetowych uczelni wyższych. Na początku dokonano wyboru i klasyfikacji serwisów. Następnie wykonano analizę ekspercką, polegającą na tym, iż specjalista - osoba zaznajomiona z tematyką dostępności cyfrowej, wykonała audyt tych stron - stanowiących materiał badawczy oraz przyznawała punkty w kategoriach wymienionych w Tabeli 1.

Tabela 1: Lista kontrolna do weryfikacji wybranych funkcji dostępności

Wielkość czcionki	0	brak możliwości zmiany wielkości czcionki na stronie
	1	możliwość zmiany wielkości czcionki na stronie
Kontrast	0	brak możliwości zmiany kontrastu na stronie
	1	możliwość zmiany wielkości czcionki na stronie
Język	0	strona jest dostępna tylko w jednej wersji językowej
	1	strona jest dostępna w co najmniej dwóch wersjach językowych
Wyszukiwarka	0	brak wyszukiwarki na stronie
	1	obecność wyszukiwarki na stronie
Mapa strony	0	brak mapy strony w serwisie
	1	obecność mapy strony w serwisie
Animacje	-1	brak możliwości zatrzymania animacji na stronie
	0	brak animacji na stronie
	1	możliwość zatrzymania animacji na stronie
Pliki do pobrania	0	brak informacji o formacie i wielkości w tytułach plików możliwych do pobrania
	1	tytuły plików możliwych do pobrania zawierają informacje o formacie i ich wielkości

W drugiej części pracy skorzystano z automatycznych narzędzi: Utilitia [13], Google Mobile-Friendly Test [14] oraz Google PageSpeed Insights [15]. Narzędzia te badają różne aspekty składające się na szeroko pojętą dostępność, włączając w nią użyteczność i wydajność. Walidator Utilitia jest najbardziej uniwersalnym narzędziem, badającym strony pod względem m.in. poprawności kodu html i css. Analizy przeprowadzane przez Google MobileFriendly Test skupiają się na sprawdzeniu dostosowania stron do urządzeń mobilnych, natomiast głównym zadaniem testów przeprowadzanych przez Google PageSpeed Insights jest wydajność stron. Po przeprowadzeniu pierwszej i drugiej części eksperymentu uzyskano szereg wskaźników, które przeliczono na wartości procentowe, aby ułatwić ich porównywanie. W ostatniej części eksperymentu przeprowadzono badanie za pomocą narzędzia accessiBe [16], które testuje zgodność stron z wytycznymi Web Content Accessibility Guidelines (WCAG) na poziomie 2.1 [17].

3.1. Materiał badawczy

Materiał badawczy obejmował 30 wybranych serwisów www uczelni wyższych w Polsce. Przed przystąpieniem do badań uczelnie wyższe zakwalifikowano według ich typu do jednej z sześciu grup:

1. uniwersytety (UU),
2. uczelnie techniczne (UT),
3. uczelnie ekonomiczne (UE),
4. uczelnie pedagogiczne (UP),
5. uczelnie rolnicze/przyrodnicze (UR),
6. uczelnie wychowania fizycznego (UF).

Do każdej grupy dobrano po 5 serwisów uczelni, które następnie poddano analizom.

3.2. Stanowisko badawcze

Eksperyment został zrealizowany na komputerze mobilnym skonfigurowanym w ten sposób, aby możliwe było płynne i bezproblemowe działanie najnowszej wersji przeglądarki internetowej Mozilla Firefox. Laptop był włączony do sieci za pomocą przewodu Ethernet, by uniknąć opóźnień i przerw, które mogłyby wystąpić przy połączeniu bezprzewodowym. Dokładna specyfikacja stanowiska badawczego została przedstawiono w Tabeli 2.

Tabela 2: Specyfikacja stanowiska badawczego

Model laptopa	Asus a15 FA 506II
Procesor	AMD Ryzen 5 4600H
Pamięć RAM	16 GB (DDR4, 3200MHz)
Ekran	15,6'' LED
Karta graficzna	AMD Radeon Graphics 1500 MHz
Karta sieciowa	Lan Rj-45 1 Gb/s
Dysk	SSD M.2 NVMe PCIe 3.0 512 GB
System operacyjny	Windows 10 Home 22H2
Przeglądarka	Mozilla Firefox 109.0

4. Wyniki badań

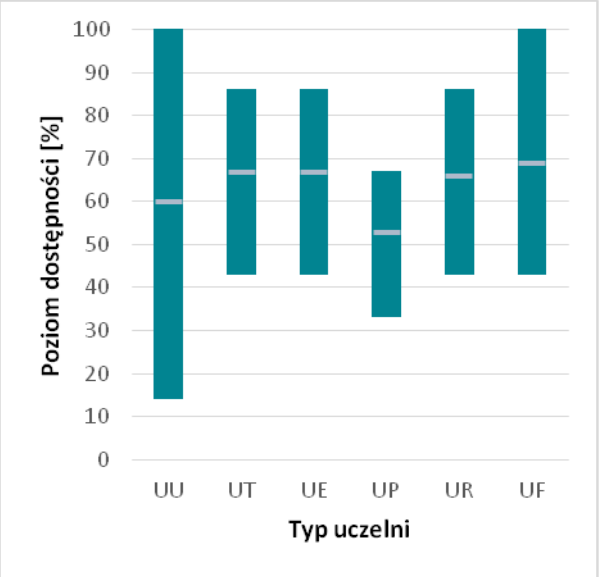
Po zrealizowaniu badań otrzymano wyniki, które znormalizowano, a następnie uśredniono i uogólniono do postaci pięciu wskaźników reprezentujących:

- poziom dostępności - otrzymany na podstawie analizy eksperckiej,
- poziom dostępności - obliczony za pomocą walidatora Utilitia, określający spełnienie konkretnych wytycznych dotyczących standardu WCAG 2.0 i poprawność kodu HTML i CSS,
- poziom dostosowania serwisów do urządzeń mobilnych - wyznaczony za pomocą narzędzia Google Mobile Friendly Test,
- poziom zgodności stron z wytycznymi WCAG 2.1 - określony za pomocą narzędzia accessiBe,
- poziom wydajności stron - zmierzony przy pomocy Google PageSpeed Insights.

4.1. Analiza ekspercka

Pierwsza część badań skupiała się na przeglądzie zastosowanych narzędzi dostępności, w jakie były wyposażone strony uczelni. Ta część badań została zrealizowana za pomocą przygotowanej do tego celu listy kontrolnej. Na Rysunku 1 przedstawiono uśrednione wyniki dostępności serwisów dla poszczególnych typów szkół wyższych. Oś pionowa reprezentuje średni poziom dostępności wyrażony w procentach, wyznaczony na podstawie kryteriów, które zostały wzięte pod uwagę przy konstruowaniu listy kontrolnej. Oprócz średniej wykres przedstawia wartości minimalne i maksymalne.

Według Rysunku 1 średni poziom dostępności dla wszystkich badanych uczelni wyniósł 63,67%. Na skutek dokonanych uśrednień różnice między poszczególnymi rodzajami uczelni są niewielkie. Najlepiej wypadły uczelnie wychowania fizycznego ze średnią 69%, nieco gorzej uczelnie techniczne i ekonomiczne (po 67%), a najsłabiej uniwersytety i uczelnie pedagogiczne z wynikami odpowiednio 60% i 53%. Wyniki dwóch serwisów uniwersytetów (Uniwersytet Mikołaja Kopernika w Toruniu – 14%, Uniwersytet Gdański – 29%) zaniżyły średnią uniwersytetów, pomimo, że pozostałe dwa (Uniwersytet Marii Curie-Skłodowskiej w Lublinie i Uniwersytet Wrocławski) otrzymały noty maksymalne.



Rysunek 1: Dostępność cyfrowa serwisów www poszczególnych grup uczelni wyznaczona na podstawie analizy eksperckiej.

4.2. Walidator Utilitia

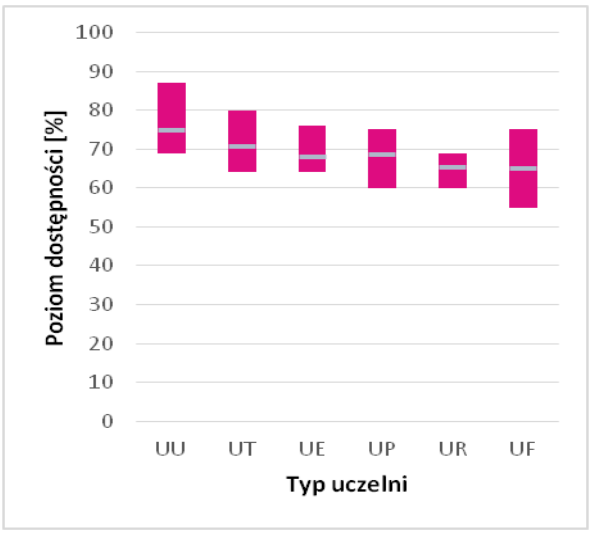
W drugiej części badania przeprowadzono audyt stron www za pomocą narzędzia Utilitia, które daje wynik w postaci liczby punktów w przedziale od 0 do 10. Na Rysunku 2 przedstawiono średni procentowy poziom dostępności wyznaczony dla każdej grupy uczelni.

W tym badaniu poziom dostępności dla wszystkich badanych uczelni wyniósł 68,83%. Tutaj także wystąpiły małe różnice w poziomie dostępności między poszczególnymi grupami uczelni. W tym porównaniu najlepiej wypadły serwisy uniwersytetów (75%), a najgorzej serwisy uczelni rolniczych i uczelni wychowania fizycznego (około 65%).

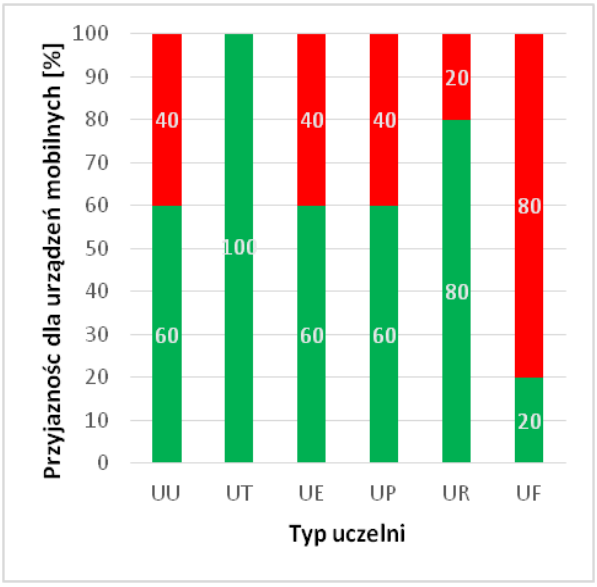
4.3. Google Mobile Friendly Test

W celu sprawdzenia dostosowania serwisów do urządzeń mobilnych użyto narzędzia Google Mobile Friendly Test, które daje wynik pozytywny, jeśli strona jest przystosowana do urządzeń mobilnych, a negatywny, jeśli strona nie wyświetla się poprawnie na tych urzą-

dzeniach. Na Rysunku 3 przedstawiono poziom przyjazności serwisów www dla urządzeń mobilnych pogrupowanych w zależności od ich. Kolorem zielonym zaznaczono procent uczelni, które uzyskały wyniki pozytywne, a czerwonym negatywne.



Rysunek 2: Poziom dostępności serwisów uczelni wg rodzaju uczelni wyznaczony za pomocą walidatora Utilitia.



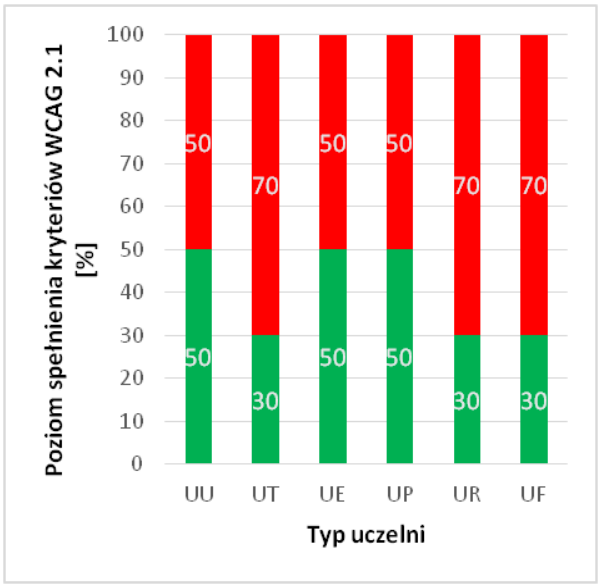
Rysunek 3: Poziom przyjazności dla urządzeń mobilnych serwisów www pogrupowanych w zależności od typu uczelni.

Średni poziom przyjazności serwisów dla urządzeń mobilnych pogrupowanych według rodzaju uczelni wyniósł 63,33%. W tym porównaniu bardzo źle wypadły uczelnie wychowania fizycznego, z których tylko jedna spełniała to kryterium. Z kolei najlepiej wypadły uczelnie techniczne, których wszystkie serwisy okazały się przyjazne dla urządzeń mobilnych.

4.4. WCAG 2.1

W kolejnej części badań przeprowadzono audyt stron narzędziem accessiBe pod kątem ich zgodności z wytycznymi Web Content Accessibility Guidelines w wersji 2.1. Na Rysunku 4 przedstawiono procentowy poziom spełnienia kryteriów WCAG 2.1 przez serwisy uczelni należące do poszczególnych grup uczelni.

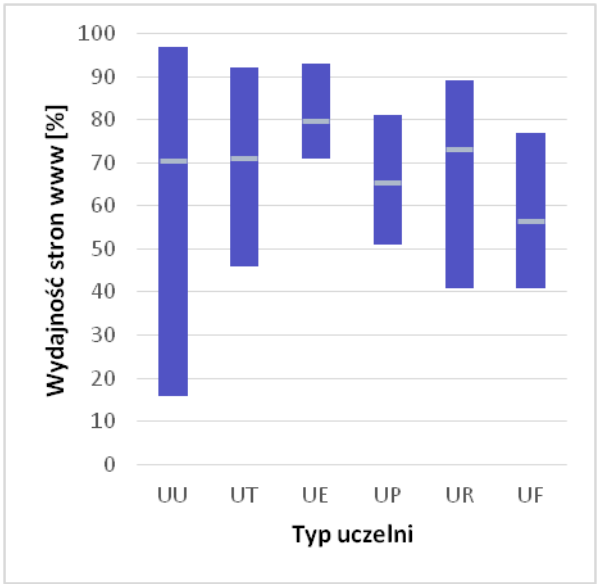
Średni poziom pokrycia kryteriów przez serwisy uczelni pogrupowanych względem rodzaju uczelni był niski i wyniósł 40%. Większość badanych serwisów (22 z 30) tylko częściowo spełnia wytyczne WCAG 2.1. Ostatecznie tylko jedna uczelnia spełniła wszystkie kryteria WCAG 2.1. Był to Uniwersytet Pedagogiczny im. Komisji Edukacji Narodowej w Krakowie.



Rysunek 4: Poziom spełnienie kryteriów WCAG 2.1 przez serwisy uczelni pogrupowane według typu uczelni.

4.5. Google PageSpeed Insights

Narzędzie Google PageSpeed Insights wykorzystano w celu określenia wydajności badanych stron. Po przeprowadzeniu testów obliczono średnie wartości w każdej grupie i przedstawiono je w postaci procentowej. Na Rysunku 5 pokazano wydajność serwisów www dla poszczególnych typów uczelni.



Rysunek 5: Wydajność serwisów www według typów uczelni wyznaczona na podstawie narzędzia Google PageSpeed Insights.

Średnia szybkość działania stron dla wszystkich badanych uczelni wyniosła 69,40%. Zaobserwowano duże rozbieżności między średnimi poszczególnych grup uczelni, które wahają się w przedziale 56,4% – 79,6%. Szczególnie duże różnice wystąpiły w szybkości działania stron uniwersytetów (różnica 81%), a nieco mniejsze w przypadku uczelni technicznych (różnica 46%) i uczelni rolniczych (różnica 48%). Najwyższe wskaźniki wydajności osiągnęły strony uczelni ekonomicznych, ze średnią 79,6%.

5. Wnioski

Na podstawie zaprezentowanych wyników nasuwają się następujące wnioski. Średni poziom dostępności stron wszystkich badanych uczelni uzyskany w wyniku przeprowadzenia analizy eksperckiej za pomocą listy kontrolnej wyniósł 63,67%. Oznacza to, że istnieje jeszcze duże pole do działania w kierunku poprawy tego wskaźnika. Najwyższy wynik osiągnęły uczelnie wychowania fizycznego (69%), a najniższy uczelnie przyrodnicze i rolnicze (53%). Ponadto analiza ekspercka pokazała, że uczelnie na swoich stronach internetowych stosują typowe narzędzia dostępności oraz że uczelnie modernizują swoje serwisy i uwzględniają przy tym kwestie dostępności. Zdarzają się również uczelnie, które już w tym momencie stosują wymogi prawa oraz w swoich serwisach i aplikacjach posiadają szeroką gamę narzędzi dostępności.

Analizy automatycznie wykonane przy użyciu narzędzi Utilitia, Google Mobile-Friendly Test oraz accessiBe dały zróżnicowane i niejednoznaczne wyniki. Według walidatora Utilitia najwyższy poziom dostępności uzyskały uniwersytety (75%), a najniższy uczelnie przyrodnicze i rolnicze oraz wychowania fizycznego (65%). Testy dostosowania serwisów do urządzeń mobilnych wykazały, że najlepszy wynik osiągnęły uczelnie techniczne (100%), a najniższy uczelnie wychowania fizycznego (20%). Kolejna analiza pokazała, że największą zgodność z kryteriami WCAG 2.1 miały uniwersytety, uczelnie ekonomiczne i pedagogiczne (50%), a najniższą uczelnie techniczne, rolnicze i przyrodnicze oraz wychowania fizycznego.

Szybkość działania serwisów ma istotny wpływ na zadowolenie użytkowników. Porównując serwisy różnych typów uczelni, można zauważyć dość duże różnice występujące w wydajności stron www. W tym porównaniu strony uczelni ekonomicznych uzyskały najlepszy wynik, podczas gdy strony uczelni wychowania fizycznego osiągnęły najniższą wydajność. Płynie z tego wniosek, że uczelnie powinny dbać o wydajność swoich stron internetowych, aby szybko się wczytywały i płynnie działały.

Analiza dostępności stron internetowych uczelni wykazała, że spełnienie wymogów dostępności to proces ciągły. Konieczne jest regularne monitorowanie, ocenianie i aktualizowanie stron i aplikacji, aby zapewnić im zgodność z obowiązującymi standardami dostępności. Również deklaracje dostępności powinny szczegółowo i precyzyjnie przedstawiać elementy, które wymagają poprawy w zakresie dostępności oraz powinny wskazywać perspektywę czasową ich dostosowania.

Podsumowując, analiza dostępności stron internetowych uczelni wyższych wykazała potrzebę kontynuowania działań mających na celu poprawę dostępności, wydajności i użyteczności tych stron. Zapewnienie dostępności dla wszystkich użytkowników, w tym osób z niepełnosprawnościami, powinno być priorytetem dla uczelni, aby zapewnić równość szans i pełne uczestnictwo w edukacji oraz dostęp do informacji oferowanych na stronach a także w aplikacjach internetowych uczelni.

Literatura

[1] Czym jest dostępność cyfrowa, https://utilitia.pl/czym-jest-dostepnosc-cyfrowa/, [11.12.2022].

[2] Dyrektywa Parlamentu Europejskiego I Rady (UE) 2016/2102 z dnia 26 października 2016 r. w sprawie dostępności stron internetowych i mobilnych aplikacji organów sektora publicznego (Dziennik Urzędowy Unii Europejskiej L 327/1,2.12.2016).

[3] Konieczny Dostępność cyfrowa. Omówienie wymogów dostępności cyfrowej dla podmiotów publicznych, https://www.gov.pl/web/dostepnosc-cyfrowa/omowienie-wymogow-dostepnosci-cyfrowej-dla-podmiotow-publicznych, [11.12.2022].

[4] Ustawa z dnia 4 kwietnia 2019 r. o dostępności cyfrowej stron internetowych i aplikacji mobilnych podmiotów publicznych (Dz. U. 2019 poz. 848).

[5] D. Paszkiewicz, J. Dębski, Dostępność serwisów internetowych. Dobre praktyki w projektowaniu serwisów internetowych dostępnych dla osób z różnymi niepełnosprawnościami, Stowarzyszenie Przyjaciół Integracji, Warszawa, 2013, https://www.researchgate.net/publication/284182660_Dostepnosc_serwisow_internetowych.

[6] Accessibility Evaluator (ACE), https://ace.accessibe.com/, [11.12.2022].

[7] W3C, Supported States and Properties, https://www.w3.org/WAI/fundamentals/accessibility-intro/, [11.12.2022].

[8] W. Stasiak, M. Dzieńkowski, Accessibility assessment of selected university websites, Journal of Computer Sciences Institute 19 (2021) 81-88, https://doi.org/10.35784/jcsi.2462.

[9] M. Bednarczyk, M. Dzieńkowski, Evaluation of the availability of websites of communes in the Lubelskie Province, Journal of Computer Sciences Institute 19 (2021) 114-120, https://doi.org/10.35784/jcsi.2618.

[10] S. Rahmatizadeh, S. Valizadeh-Haghi, Monitoring for accessibility in medical university websites: Meeting the needs of people with disabilities, Journal of Accessibility and Design for All, 8 (2), (2018) 102-124, http://www.jacces.org/index.php/jacces/article/view/150/201.

[11] AChecker, Functional Accessibility Evaluator (FAE), https://fae.disability.illinois.edu/anonymous/?Anonymous%20Report=/, [11.12.2022].

[12] E. Ostertagová, O. Ostertag, J. Kováč, Methodology and Application of the Kruskal-Wallis Test, Applied Mechanics and Materials 611 (2014) 115-120 https://doi.org/10.4028/www.scientific.net/AMM.611.115.

[13] Walidator utilitia, https://validator.utilitia.pl/, [11.12.2022].

[14] Google Mobile-Friendly Test, https://search.google.com/test/mobile-friendly, [11.12.2022].

[15] Goggle PageSpeed Insights, https://pagespeed.web.dev/?hl=pl, [11.12.2022].

[16] accessiBe, https://accessibe.com/accessscan, [11.12.2022].

[17] J. Dębski, WCAG 2.1 w skrócie, https://www.gov.pl/web/dostepnosc-cyfrowa/wcag-21-w-skrocie, [11.12.2022].

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 28 (2023) 291–295
Received: 19 June 2023
Accepted: 3 July 2023

Impact of changes in graphics setting on performance in selected video games

Wpływ zmian ustawień graficznych na wydajność w wybranych grach komputerowych

Kamil Szafran*, Małgorzata Plechawska-Wójcik

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

Abstract
Computer games, especially more advanced ones, allow users to change graphic settings to improve the overall appearance of the game or to achieve smoother gameplay. The aim of the paper is conducting research to determine whether changes in graphic settings affect computer performance. Parameters such as temperature and usage of graphics cards and processors or frames per second were examined. Three different computer games and three computer stations were used to conduct the tests. Two research hypotheses were formulated: "Changing graphic settings in games affects computer performance" and "The change in shadow quality in games has the greatest impact on the tested parameters." The obtained results confirmed both of these hypotheses.

Keywords: computer games performance; Unreal Engine 4; graphics settings

Streszczenie
Gry komputerowe, zwłaszcza te bardziej rozbudowane, pozwalają użytkownikom na zmianę ustawień graficznych w celu poprawy wyglądu całej gry lub dążenia do płynniejszej rozgrywki. Celem artykułu jest przeprowadzenie badań mających na celu sprawdzenie czy zmiany ustawień graficznych będą wpływać na wydajność komputerów. Sprawdzano takie parametry jak temperatura, wykorzystanie kart graficznych i procesorów lub liczbę klatek na sekundę. Do przeprowadzenia testów użyto trzech różnych gier komputerowych oraz trzech komputerów. Postawiono dwie hipotezy badawcze: „Zmiana ustawień graficznych w grach ma wpływ na wydajność komputera" oraz „Największy wpływ na badane parametry ma zmiana jakości cieni w grach". Otrzymane wyniki potwierdziły obie te hipotezy.

Słowa kluczowe: wydajność gier komputerowych; Unreal Engine 4; ustawienia graficzne

*Corresponding author
Email address: kamsin21011999@gmail.com (K. Szafran)

1. Wstęp

Branża gier komputerowych, będąca jedną z największych branż rozrywkowych, rozwija się wraz z postępem technologicznym sprzętu komputerowego [1]. Tworzone gry oferują coraz więcej treści, lepszą grafikę i są coraz bardziej skomplikowane. Jednak dostęp do nowego sprzętu jest utrudniony z powodu wysokich cen i braku dostępności na rynku, co prowadzi do zmniejszenia innowacyjności i wydajności używanych komputerów. Mimo to, twórcy gier nie przestają tworzyć coraz bardziej wymagających produkcji. Dlatego starają się optymalizować gry poprzez ustawienia graficzne, które pozwalają użytkownikom dostosować wygląd i płynność rozgrywki.

Jednym z narzędzi, dzięki któremu powstają gry komputerowe, jest silnik Unreal Engine 4 [2]. Jest to zaawansowane narzędzie do tworzenia wizualnie imponujących i technologicznie zaawansowanych produkcji. Unreal Engine 4 oferuje szeroki zakres narzędzi i funkcji, umożliwiając programistom i twórcom gier tworzenie realistycznych światów, zaawansowanych efektów graficznych i dynamicznej fizyki. Silnik ten jest wykorzystywany przez wiele znanych firm i twórców gier, dzięki czemu powstało wiele popularnych tytułów na różne platformy. Dzięki swojej elastyczności i możliwości personalizacji, Unreal Engine 4 jest ceniony przez branżę gier komputerowych jako narzędzie o ogromnym potencjale twórczym.

Niniejszy artykuł ma na celu zbadanie wpływu zmian ustawień graficznych w grach komputerowych, powstałych dzięki Unreal Engine 4. Testy wydajnościowe przeprowadzono na trzech jednostkach komputerowych. Do wykonania badań wybrano trzy gry: Final Fantasy VII Remake: Intergrade [3], Star Wars Jedi: Fallen Order [4] oraz The Outer Worlds [5]. Zbadano wybrane parametry, poddano je analizie i otrzymano wyniki w postaci wykresów oraz tabel.

Na podstawie omawianego zagadnienia postawiono następujące dwie hipotezy badawcze: „Zmiana ustawień graficznych w grach ma wpływ na wydajność komputera" oraz „Największy wpływ na badane parametry ma zmiana jakości cieni w grach".

Nie znaleziono artykułów, które poruszają dokładnie taką samą tematykę jak dany artykuł. Większość innych artykułów sprawdza np. konkretne funkcje silnika we własnych projektach [6] albo porównuje ogólnie wybrane silniki gier [7]. Niewiele jest prac sprawdzających produkty lub gry, które powstały dzięki nim. Mimo tego zwrócono uwagę na duże możliwości pod względem grafiki dla gier, które są stworzone w silniku Unreal

Engine. Literatura wskazuje również, że Unreal Engine wykorzystuje najwięcej zasobów komputera, kosztem produkowania najlepszych efektów pod względem wizualnym oraz graficznym [8].

2. Metodyka badań

Przeprowadzenie badań poprzez wykonanie testów wydajnościowych odbyło się za pomocą dwóch narzędzi: MSI Afterburner [9] oraz Riva Tuner Statistics Server (RTSS) [10]. Dzięki nim można było monitorować wybrane parametry komputera w czasie rzeczywistym co jest widoczne na Rysunku 1. Możliwe było również zapisywanie wyników do plików.

Rysunek 1: Wygląd sprawdzania danych podczas rozgrywki.

Dzięki tym dwóm narzędziom zbadano następujące parametry (liczone w następujących jednostkach):

- temperatura karty graficznej (w °C),
- wykorzystanie karty graficznej (w %),
- wykorzystanie pamięci karty (w MB),
- taktowanie rdzenia karty graficznej (w MHz),
- taktowanie pamięci karty graficznej (w MHz),
- temperatura procesora (w °C),
- wykorzystanie procesora (w %),
- taktowanie procesora (w MB),
- wykorzystanie pamięci RAM (w MB),
- liczba klatek na sekundę (w FPS – ang. frames per second).

Do przeprowadzenia testów wydajnościowych wykorzystano trzy różne jednostki komputerowe, o różnych parametrach, przedstawione w Tabeli 1.

Tabela 1: Parametry komputerów wykorzystanych do przeprowadzenia badań

	Komputer 1	Komputer 2	Komputer 3
CPU	Intel Core i5-6300HQ 2.3 GHz 4-rdzeniowy	AMD Ryzen 7 1700x 3.4 GHz, 8-rdzeniowy	Inter Core i7-11700K, 3.6 GHz, 8-rdzeniowy
RAM	8 GB, 2133 MHz	16 GB, 3400 MHz	64 GB, 3200 MHz
OS	Windows 10	Windows 11	Windows 11
GPU	NVIDIA GeForce GTX 960M	NVIDIA GeForce GTX 1060	NVIDIA GeForce RTX 3070
Dysk	Samsung 870 EVO 1TB	Serial ATAWDC 500 GB	Kingstron SNSV 1TB

Każda z tych gier pozwalała użytkownikom na wybranie ustawień graficznych w różnych kombinacjach. Pierwsza z nich oferuje 4 różne kombinacje, natomiast pozostałe dwie oferują 27 różnych kombinacji ustawień graficznych (Tabela 2). Powodem dla różnych liczb kombinacji jest to, że gra Final Fantasy VII Remake: Intergrade jest konwersją gry konsolowej na grę komputerową i oferuje mniejszy wybór ustawień graficznych. Natomiast pozostałe dwie gry były pierwotnie zaprojektowane na komputery stacjonarne, przez co mają większą możliwość dostosowywania ustawień graficznych ze względu na większą różnorodność jednostek komputerowych. Testy wydajnościowe przeprowadzano na każdym komputerze, na każdej grze po trzy lub pięciokrotnie. Przechodzono wybrany fragment danej gry, trwający od trzech do pięciu minut i wszystkie wyniki zapisywano do plików .hml, których dane następnie wczytywano do plików programu Excel.

Tabela 2: Ustawienia graficzne dla Final Fantasy VII

Nazwa gry	Możliwe ustawienia graficzne	Liczba wykonywanych testów wydajnościowych
Final Fantasy VII Remake: Intergade	- Jakość cieni: niskie / wysokie - Jakość tekstur: niskie / wysokie	Pięć trwających 3-5 min.
Star Wars Jedi: Fallen Order oraz The Outer Worlds	- Jakość cieni: średnie / wysokie / epickie - Jakość tekstur: średnie / wysokie / epickie - Jakość efektów cząsteczkowych: średnie / wysokie / epickie	Trzy trwające 3-4 min.

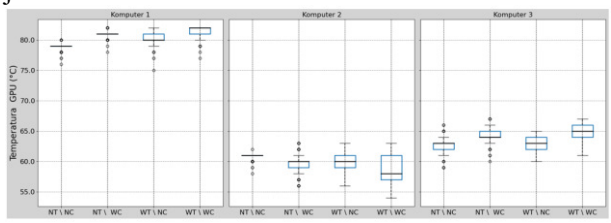
3. Analiza wyników badań

Po wykonaniu badań, wszystkie zebrane dane poddano testom statystycznym. Sprawdzono czy istnieją statystycznie istotne różnice pomiędzy kolejnymi wynikami badań. Najpierw przeprowadzono test statystyczny Kruskala-Walisa [11], jako że jest to nieparametryczny test statystyczny używany do porównywania median różnych grup, który pozwala na stwierdzenie, czy różnice między badanymi grupami są istotne. Po otrzymaniu wyników, wykonano dodatkowy test statystyczny post-hoc, mianowicie test Dunn'a [12]. Zdecydowano się na niego, ponieważ pozwala na porównanie par grupowych bez założenia normalności rozkładu danych. Dodatkowo, test Dunn'a chroni przed popełnieniem błędów typu I. Dzięki tym testom można było ustalić, że zmiana ustawień graficznych nie powoduje żadnych (albo ledwie zauważalnych) zmian dla następujących parametrów: taktowanie rdzenia karty graficznej, taktowanie pamięci karty graficznej oraz taktowanie procesora. Z

tego powodu nie poddano dalszej analizie owych parametrów.

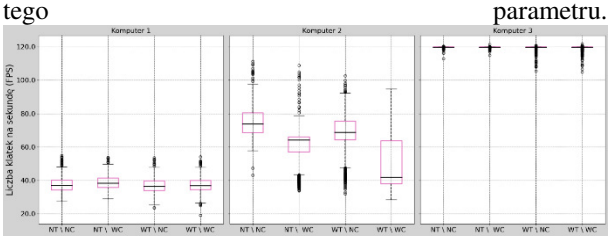
Z otrzymanych wyników dla pozostałych badanych parametrów z wykonanych testów wydajnościowych utworzono wykresy pudełkowe. Jest to graficzna reprezentacja rozkładu danych numerycznych, która przedstawia informacje takie jak mediana, kwartyle, wartości skrajne i ewentualne odstępstwa. Jego główna idea polega na wizualizacji centralnych tendencji i rozproszenia danych, co pozwala na szybkie porównanie wielu zestawów danych lub identyfikację nietypowych wartości w danym zbiorze. Porównują każdy zbadany parametr dla każdej gry, dla każdego komputera.

Rysunek 2 przedstawia wykres danych jaki był utworzony dla gry Final Fantasy VII Remake: Intergrade, dla temperatury karty graficznej. Liczba możliwych kombinacji ustawień graficznych była równa cztery. Pozwoliło to na przedstawienie wykresów pudełkowych dla wszystkich trzech komputerów, na których przeprowadzono testy wydajnościowe. Osie y zawierają wartości parametrów. Wartości na osi x w przypadku gry Final Fantasy VII Remake: Intergrade oznaczają wybrane kombinacje ustawień graficznych. Oznaczenia są następujące: NT – niska jakość tekstur, WT – wysoka jakość tekstur, NC – niska jakość cieni, WC – wysoka jakość cieni.



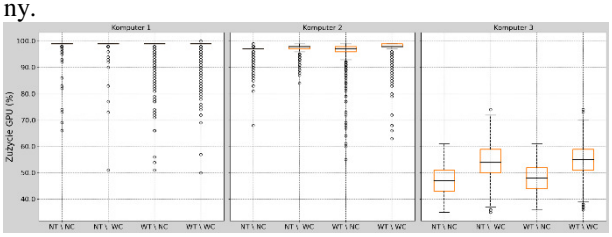
Rysunek 2: Temperatura karty graficznej dla gry Final Fantasy VII.

Widoczny jest wzrost wartości median, podczas zmiany ustawień graficznych z niższych na wyższe. W tym przypadku zmiana jakości cieni wpływa w większym stopniu na wyniki niż zmiana jakości tekstur. Podobna sytuacja zachodzi dla parametru, który zliczał liczbę klatek na sekundę. Rysunek 3 przedstawia wykresy utworzone na podstawie zebranych danych dla tego parametru.



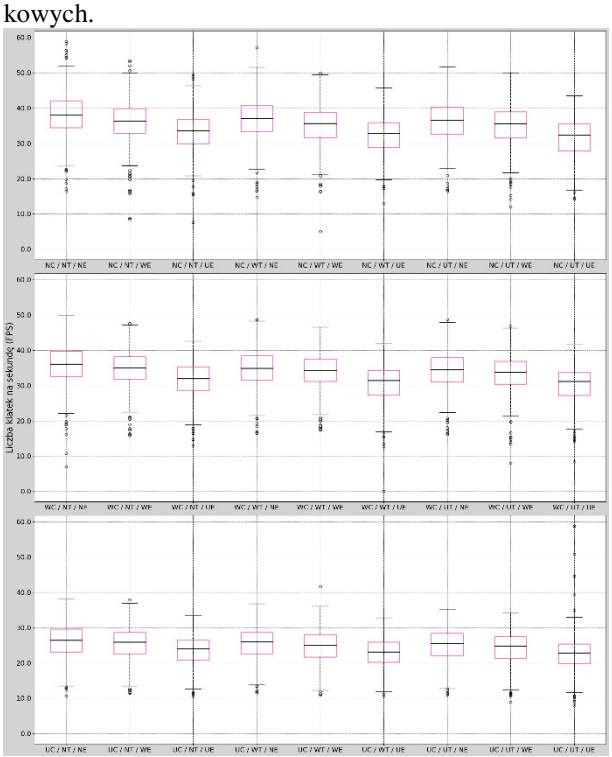
Rysunek 3: Liczba klatek na sekundę dla gry Final Fantasy VII.

Widoczny jest spadek wartości median, wskazujący na osiąganie coraz gorszych wyników wraz ze zmianą ustawień. Czasami występowały przypadki, w których ustalenie, czy zmiana ustawień graficznych ma wpływ na badany parametr była utrudniona lub nie możliwa. Przykładem jest ostatnia sekcja w Rysunku 3, gdzie zostały przedstawione wyniki dla komputera 3. W tym przypadku jedynie za pomocą odstępstw od wykresu pudełkowego można zauważyć zmiany w wynikach. Podobna sytuacja zaszła również podczas przedstawiania wyników z badania wykorzystania karty graficznej (Rysunek 4), gdzie także jedynie odczyt odstępstw w dwóch pierwszych sekcjach może wskazywać na zmiany.



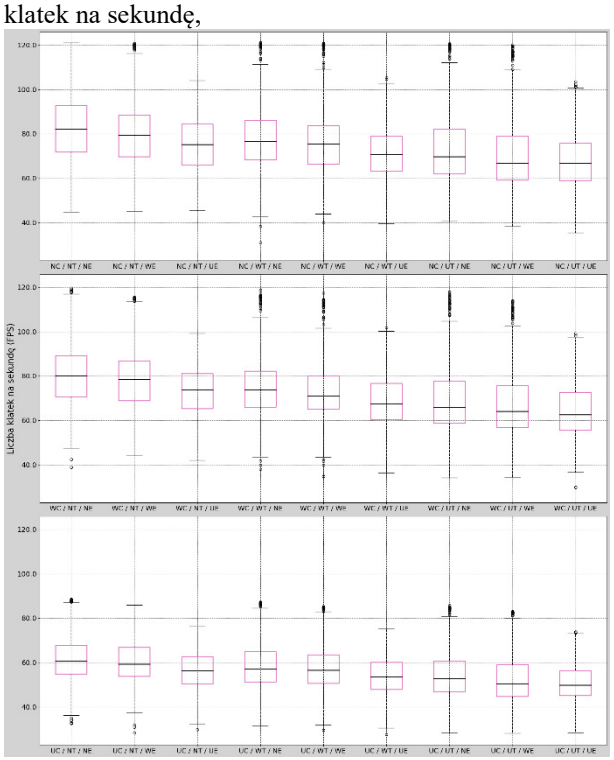
Rysunek 4: Wykorzystanie karty graficznej dla gry Final Fantasy VII.

Przedstawianie wyników w postaci wykresów pudełkowych dla wszystkich trzech komputerów na jednym wykresie było możliwe jedynie dla gry Final Fantasy VII Remake: Intergrade. W przypadku dwóch pozostałych gier zdecydowano się na pokazywanie wyników na jednym wykresie tylko dla jednego komputera dla danego parametru. Powodem była przejrzystość przedstawionych wykresów. Pozostałe dwie gry mają tych kombinacji 27, dlatego na rysunkach wyniki dla poszczególnych kombinacji parametrów umieszczono w rzędach (Rysunek 5). Oś y zawiera wartości badanych parametrów. Oś x oznacza wybrane kombinacje ustawień graficznych. Oznaczenia są takie same jak w przypadku wykresów z gry Final Fantasy VII, ale dochodzą jeszcze następujące skróty: UT – ultra jakość tekstur, UC – ultra jakość cieni. NE – niska jakość efektów cząsteczkowych, WE – wysoka jakość efektów cząsteczkowych, UE – najwyższa jakość efektów cząsteczkowych.



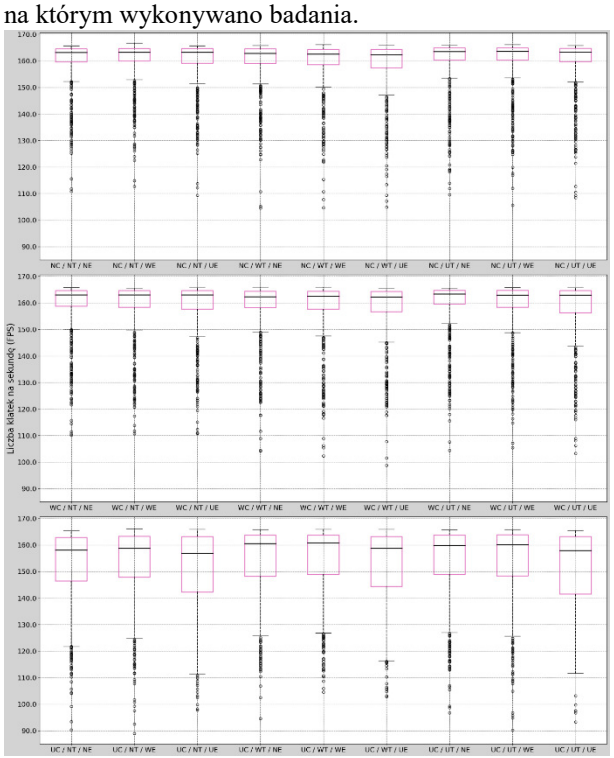
Rysunek 5: Liczba klatek na sekundę dla gry The Outer Worlds, dla komputera 1.

Rysunek 6 to również przedstawienie wyników dla liczby klatek na sekundę dla gry The Outer Worlds, ale już dla komputera 2. Ponownie przedstawiono liczbę klatek na sekundę,



Rysunek 6: Liczba klatek na sekundę dla gry The Outer Worlds, dla komputera 2.

Porównywano wyniki dla każdego komputera, więc Rysunek 7 przedstawia także badany parametr, czyli liczbę klatek na sekundę, ale dla ostatniego komputera na którym wykonywano badania.



Rysunek 7: Liczba klatek na sekundę dla gry The Outer Worlds, dla komputera 3.

Wszystkie wykresy zostały następnie poddane analizie oraz stworzono system oceniania wpływu zmian ustawień graficznych na wydajność. Polegał on na sprawdzaniu każdego wykresu i oceny czy zmiana wybranego ustawienia graficznego wpływała na badany parametr. Jeżeli tak, to oceanio go w skali od 1 do 3, gdzie 1 oznaczało wpływ w najmniejszym stopniu, a 3 w najwyższym. Jeżeli zmiana danego ustawienia graficznego nie wpływała w żaden sposób na badany parametr wystawiano w tym przypadku ocenę 0. Jeżeli natomiast nie można było dokładnie wskazać, które ustawienie graficzne wpływa najbardziej na dany parametr przydzielano ocenę 2 każdemu z nich. Postąpiono tak z każdym z siedmiu parametrów, dla każdej z trzech gier, dla każdego z trzech komputerów. Pod koniec, oceny te zsumowano i porównano ze sobą. Przykładowe wyniki po ocenie ukazane są w Tabeli 4 oraz 5. Pierwszy wiersz jest przeznaczony dla danych parametrów które kolejno są oznaczone cyframi od 1 do 7. Oznaczają kolejno: temperatura karty graficznej, wykorzystanie karty graficznej, wykorzystanie pamięci karty, temperatura procesora, wykorzystanie procesora, wykorzystanie pamięci RAM, liczba klatek na sekundę.

Tabela 3: Wyniki oceny dla gry The Outer Worlds, dla komputera 2

	1.	2.	3.	4.	5.	6.	7.	Suma
Cienie	0	2	1	2	3	2	3	13
Tekstury	0	2	3	2	2	2	1	12
Efekty cząsteczkowe	0	2	2	2	1	2	2	11

Tabela 4: Wyniki oceny dla gry Final Fantasy VII, dla komputera 3

	1.	2.	3.	4.	5.	6.	7.	Suma
Cienie	2	2	1	2	2	1	0	10
Tekstury	1	1	2	1	1	2	0	8

Ostatecznie wyniki z systemu oceniania prezentują się w następujący sposób. Tabela 6 przedstawia wyniki zebrane dla gry Final Fantasy VII Remake: Intergrade, Tabela 7 przedstawia wyniki dla gry Star Wars Jedi: Fallen Order, a Tabela 8 przedstawia wyniki dla gry The Outer Worlds.

Tabela 5: Suma punktów z oceny gry Final Fantasy VII

	Komputer 1	Komputer 2	Komputer 3
Cienie	9	11	10
Tekstury	9	10	8

Z Tabeli 6 odczytać można, że zmiana jakości cieni ma większy wpływ na badane parametry niż zmiana

jakości tekstur podczas rozgrywki w Final Fantasy VII Remake: Intergrade. Możliwe było jedynie porównanie dwóch możliwych ustawień graficznych. W przypadku gier Star Wars Jedi: Fallen Order oraz The Outer Worlds dochodziła trzecia opcja: jakość efektów cząsteczkowych.

Tabela 6: Suma punktów z oceny gry Star Wars Jedi

	Komputer 1	Komputer 2	Komputer 3
Cienie	9	13	12
Tekstury	11	12	12
Efekty cząsteczkowe	10	11	12

Tabela 7 przedstawia wyniki oceny wpływu zmian ustawień graficznych dla gry Star Wars Jedi: Fallen Order. Odczyt wyników nie pozwala jednoznacznie stwierdzić, zmiana którego z ustawień graficznych najbardziej wpływa na wydajność komputerów. Jedynie można stwierdzić, że zmiana jakości cieni oraz tekstur wpływa w niewielkim stopniu bardziej na wydajność niż zmiana jakości efektów cząsteczkowych.

Tabela 7: Suma punktów z oceny gry The Outer Worlds

	Komputer 1	Komputer 2	Komputer 3
Cienie	14	11	19
Tekstury	12	12	12
Efekty cząsteczkowe	10	7	11

Wyniki z Tabeli 8 tym razem wskazują, że największy wpływ na badane parametr miała zmiana jakości cieni podczas rozgrywki w The Outer Worlds.

4. Wnioski

W niniejszym artykule zbadano wpływ zmian ustawień graficznych w wybranych grach komputerowych. Wybrano trzy gry komputerowe, trzy jednostki komputerowe, odpowiednie narzędzia do badań i wykonano testy wydajnościowe. Po analizie wyników i ocenieniu każdego z możliwych badanych ustawień graficznych można stwierdzić, że zmiana ustawień graficznych w grach komputerowych ma wpływ na wydajność komputerów. Największy wpływ ma zmiana jakości cieni. Kolejnym pod względem stopnia wpływu na wydajność jest zmiana jakości tekstur, a na końcu znajduje się zmiana efektów cząsteczkowych.

Postawione hipotezy, czyli „Zmiana ustawień graficznych w grach ma wpływ na wydajność komputera" oraz „Największy wpływ na badane parametry ma zmiana jakości cieni w grach" zostały udowodnione na podstawie przeprowadzonych badań.

Literatura

[1] A New Era of Engagement in Media & Entertainment, How varying generations engage with the digital and physical world, https://newzoo.com/resources/trend-reports/a-new-era-of-engagement-in-media-entertainment-industry-report, [15.06.2023].

[2] Unreal Engine 4, https://docs.unrealengine.com/4.27/en-US/, [05.06.2023].

[3] Final Fantasy VII Remake: Intergrade, https://ffvii-remake-intergrade.square-enix-games.com/en-us/, [05.06.2023].

[4] Star Wars Jedi: Fallen Order, https://www.ea.com/pl-pl/games/starwars/jedi/jedi-fallen-order, [05.06.2023].

[5] The Outer Worlds, https://outerworlds.obsidian.net/en, [05.06.2023].

[6] J. Zhang, Implementation and Optimization of Particle Effects based on Unreal Engine 4, Journal of Physics: Conference Series, 1575 (2020) 1-7, http://dx.doi.org/10.1088/1742-6596/1575/1/012187.

[7] A. M. Ciekanowska, A. K. Kiszczak-Gliński, K. Dziedzic, Analiza porównawcza wydajności silników Unity i Unreal Engine w aspekcie tworzenia wirtualnych pokazów modeli pochodzących ze skanowania 3D, Journal of Computer Sciences Institute, 20 (2021) 247-253.

[8] A. Barczak, H. Woźniak, Comparative Study on Game Engines, Studia Informatica. System and information technology, 23 (2019) 1-2.

[9] MSI Afterburner, https://www.msi.com/Landing/afterburner/graphics-cards, [15.06.2023].

[10] Riva Tuner Statistics Server, https://rivatuner.net/, [15.06.2023].

[11] W. H. Kruskall, W. A. Wallis, Use of ranks in one-criterion variance analysis, Journal of the American statistical Association, 47 (1952) 583-621.

[12] O. J. Dunn, Multiple comparisons among means, Journal of the American statistical association, 56 (1961) 52-64.

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 28 (2023) 296–303
Received: 19 June 2023
Accepted: 29 June 2023

The comparative performance analysis of selected relational database systems

Analiza porównawcza wydajności wybranych relacyjnych systemów baz danych

Szymon Schab*

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

Abstract
The objective of this study was to carry out a performance analysis of the following database systems: MySQL, PostgreSQL and Microsoft SQL Server. For this purpose scripts were used to measure execution times of selecting, updating and inserting data. Furthermore, three data sets were utilized consisting of 100, 1 000 and 10 000 rows. The experiment included nine cases depending on the query type and the data set. For each case, thirty five test trials were conducted while first five trials were ignored i.a. because of cache storage. The statistical test was performed for the results and the trials in which the DBMS achieved best times were counted. For each case best systems were acknowledged and the most efficient system of the experiment was determined along with systems for each operation type.

Keywords: performance analysis; MySQL; PostgreSQL; Microsoft SQL Server

Streszczenie
Celem pracy było przeprowadzenie analizy wydajnościowej trzech relacyjnych systemów zarządzania bazami danych: MySQL, PostgreSQL i Microsoft SQL Server. W badaniu wykorzystano skrypty mierzące długości czasów operacji wstawiania, aktualizacji i zwracania danych, a także trzy zestawy danych liczące kolejno 100, 1 000 i 10 000 rekordów. Eksperyment składał się z dziewięciu przypadków uwzględniających rodzaj operacji i wariant zestawu danych, dla których wykonano po trzydzieści pięć prób, przy czym pierwsze pięć prób pominięto m.in. ze względu na kwestie przechowywania danych w pamięci podręcznej. Otrzymane wyniki sprawdzono pod kątem istotności różnic, a następnie dla każdego z przypadków zliczono liczbę prób, w których oprogramowania uzyskały najlepsze wyniki. Na końcu wskazano i policzono najlepsze systemy dla poszczególnych przypadków i wyznaczono najwydajniejszy system dla całego badania oraz systemy dla testowanych rodzajów operacji.

Słowa kluczowe: analiza wydajności; MySQL; PostgreSQL; Microsoft SQL Server

*Corresponding author
Email address: **szymon.schab@pollub.edu.pl** (S. Schab)

1. Wstęp

Współczesne systemy bazodanowe są podstawą wielu systemów informatycznych. Rynek oferuje mnóstwo narzędzi pozwalających na zarządzanie systemami bazodanowymi, spośród których jednymi z najpopularniejszych stały się relacyjne systemy baz danych. Rosnące zapotrzebowanie na takie programy spowodowało, że powstało ich wiele proponując tym samym zróżnicowane poziomy wydajności pracy. W pracach [1-13] skupiona została uwaga na efektywności systemów DBMS uzyskanej w testach wykorzystujących m.in. zapytania SQL. Otrzymane wyniki pozwoliły stwierdzić, które oprogramowanie i w jakich scenariuszach sprawdziło się najlepiej.

Zdolność platformy do przystosowania się do zwiększonych wymagań przetwarzania danych odgrywa kluczową rolę w podejmowaniu decyzji, dlatego celem pracy [14] było zbadanie wydajności dwóch silników relacyjnych baz danych - MySQL InnoDB i Microsoft SQL Server 2012 dla środowisk DSS. Przeprowadzone eksperymenty pozwoliły ocenić osiągi i przydatność badanych silników do małych i średniej wielkości środowisk wspomagania decyzji (DSS). W testach użyto programu Star Schema Benchmark, który wykorzystywał przebiegi zapytań (ang. query flight), z których każdy składał się z zapytań o różnej selektywności. Query Flight Q1 wykonywał łączenie *join* między tabelą Lineorder a tabelą Date Dimension, Query Flight Q2 łączył tabele Lineorder, Date, Part i Supplier, a także agregował oraz sortował dane według roku i marki, Query Flight Q3 zwracał całkowity przychód dla transakcji Lineorder w danym regionie w określonym czasie, Query Flight Q4 łączył wszystkie tabele i zwracał łączny zysk pogrupowany według roku i nacji. W badaniu użyty został program Star Schema Benchmark w celu utworzenia zestawów danych o rozmiarach 1 GB, 3 GB, 6 GB, 12 GB i 24 GB. Przy użyciu programu DBgen wygenerowano tabele z 6, 18, 36, 64, 71, 144 milionami rekordów. Następnie przetestowano zapytania SSB dla różnych zestawów danych, każdy z 3 uruchomieniami i obliczono średni czas wykonania zapytania. Dla zestawów danych do 12 GB, MySQL InnoDB uzyskał czasy zapytań, które można uznać za akceptowalne, ale w przypadku eksploracji i analizy większych zbiorów danych pomiędzy tymi dwoma DBMS, lepszym silnikiem okazał się Microsoft SQL Server 2012.

2. Testowane technologie

MySQL jest darmowym systemem zarządzania relacyjnymi bazami danych wydanym w 1995 roku przez szwedzką firmę MySQL AB. System ten pomaga m.in. stworzyć i wdrożyć w projekcie relacyjną bazę danych, wygenerować jej kopie zapasowe czy sprawdzić integralność bazy. Omawiany program bazuje na koncepcji „open-source" pozwalającą każdemu na wgląd oraz ingerencję w kod źródłowy oprogramowania. Sam MySQL korzysta z języka SQL (ang. Structured Query Language), a dane zarządzane są w sposób zgodny z założeniami relacyjnego modelu baz danych.

Microsoft SQL Server to system zarządzania bazami danych wydany w 1989 roku przez firmę Microsoft. System ten jest stosowany na szeroką skalę przez przedsiębiorstwa różnego rodzaju, w tym przez wiele firm informatycznych. Oparty jest na języku SQL, choć sam implementuje Transact-SQL. MS SQL Server można uruchomić na następujących platformach: Windows, Linux i Docker.

PostgreSQL to system zarządzania bazami danych wydany w 1989 przez grupę specjalistów na Uniwersytecie Kalifornijskim w Berkeley. Został on stworzony na takie platformy jak Unix, Linux, Windows oraz środowiska chmurowe np. Microsoft Azure i Amazon Web Services. Uważany jest za jeden z najbardziej rozwiniętych systemów na rynku. Oprogramowanie to bazuje na języku SQL, ale także obsługuje język procedur składowanych PL/pgSQL. PostgreSQL jest programem utworzonym zgodnie z koncepcją „open-source" zapewniającą otwarty dostęp do kodu źródłowego.

3. Cel i zakres pracy

Celem pracy było przeprowadzenie analizy wydajnościowej następujących relacyjnych systemów zarządzania bazami danych: MySQL, PostgreSQL i Microsoft SQL Server. Badanie uwzględniało serię testów wykorzystujących skrypty mierzące wartości czasów poszczególnych rodzajów operacji. Weryfikowane typy zapytań to: wstawianie rekordu (INSERT), aktualizacja rekordu (UPDATE) i zwrócenie konkretnych wierszy (SELECT). Eksperyment przeprowadzono dla każdego systemu baz danych. W celu otrzymania kompleksowej i wiarygodnej oceny wydajności zastosowano zestawy danych zawierające odpowiednio 100, 1 000 i 10 000 rekordów. Każdy z tych zestawów uwzględniono we wszystkich rodzajach operacji.

Teza pracy zakłada, że system bazodanowy Microsoft SQL Server jest najwydajniejszy ze wszystkich testowanych systemów. W pracy sformułowano także pytania badawcze mające za zadanie uzyskać konkretną wiedzę w ramach określonych zagadnień. Pytania te prezentowały się następująco:

1. Który z testowanych systemów baz danych jest najwydajniejszy w przypadku zapytania zwracającego rekordy?
2. Który z testowanych systemów baz danych jest najwydajniejszy w przypadku zapytania aktualizującego rekordy?
3. Który z testowanych systemów baz danych jest najwydajniejszy w przypadku zapytania wstawiającego rekordy?

4. Plan badań

Plan eksperymentu badawczego składał się z następujących etapów:

1. Przygotowanie środowiska testowego:
   a) Zainstalowanie i skonfigurowanie systemów zarządzania bazami danych MySQL, PostgreSQL i Microsoft SQL Server.
   b) Wygenerowanie trzech baz danych wraz z tabelami o jednakowej strukturze.
   c) Zaimportowanie danych testowych do tabel w bazach danych.
   d) Przygotowanie skryptów wykonujących operacje wstawiania (insert), aktualizacji (update) i zwracania rekordów (select) dla każdego z systemów bazodanowych.
2. Przeprowadzenie badań:
   a) Wykonanie operacji wstawiania danych i zmierzenie czasów dla trzech grup rekordów liczących odpowiednio 100, 1 000 i 10 000 wierszy.
   b) Wykonanie operacji aktualizacji danych i zmierzenie czasów dla trzech grup rekordów liczących odpowiednio 100, 1 000 i 10 000 wierszy.
   c) Wykonanie operacji zwracania danych i zmierzenie czasów dla trzech grup rekordów liczących odpowiednio 100, 1 000 i 10 000 wierszy. Operacja *select r*ealizowana jest z uwzględnieniem projekcji oraz wyszukiwania danych z użyciem klauzuli *where* i podzapytania.
3. Analiza wyników:
   a) Przeprowadzenie testu statystycznego dla zestawów wyników.
   b) Wskazanie i zliczenie zwycięzców dla poszczególnych przypadków oraz wyznaczenie zwycięzcy dla całości eksperymentu i testowanych rodzajów operacji.
   c) Wyciągnięcie wniosków dotyczących wydajności testowanych systemów w kontekście przeprowadzonych operacji.

Dla wszystkich systemów bazodanowych ustalono dziewięć przypadków testowych uwzględniających rodzaj operacji i wariant zestawu danych. Dla każdego przypadku zrealizowano trzydzieści pięć prób, przy czym pierwsze pięć prób zostało pominiętych ze względu na kwestie optymalizacji bazy danych i kwestie przechowywania danych w pamięci podręcznej. W dalszej kolejności zestawiono ze sobą rezultaty otrzymane przez systemy dla poszczególnych przypadków i wykorzystano odpowiedni test statystyczny do zbadania istotności różnic zestawów wyników. Następnie dla każdego z przypadków zliczono liczbę prób, w których oprogramowania uzyskały najlepsze wyniki oraz wskazano i policzono zwycięzców dla tych przypadków.

Przygotowanie systemów bazodanowych odbyło się na komputerze o specyfikacji przedstawionej w Tabeli 1. Konfigurację serwerów przedstawiono w Tabeli 2.

Znajdują się tutaj takie parametry jak rozmiar bufora czy maksymalna liczba połączeń. Ten pierwszy wskazuje na rozmiar w pamięci zarezerwowanej w celu zapisywania w pamięci podręcznej danych tabel oraz indeksów i ich odczytu. W przypadku MS SQL Server wspomniany rozmiar jest dobierany przez system bazodanowy. Maksymalna liczba połączeń odpowiada za największą liczbę jednoczesnych połączeń na serwerze.

Tabela 1: Specyfikacja komputera do badań

rodzaj podzespołu	podzespół
procesor	Intel Core i7-4790k, 4,00 GHz, 8 MB, 4 rdzenie, architektura 64-bitowa
pamięć RAM	Mushkin Stealth, 8 GB, 1 600 MHz, CL 9, DDR3
dysk	Samsung 870 EVO, 500 GB, SSD
system operacyjny	Windows 10 Pro, wersja 64-bitowa

Tabela 2: Konfiguracja serwerów

	MySQL	MS SQL Server	PostgreSQL
nazwa i wersja serwera	MySQL Community Server – GPL 8.0.32	SQL Server 2022 (RTM) - 16.0.1000.6	PostgreSQL 15.3
port	3306	1434	5432
adresy	wszystkie	127.0.0.1	wszystkie
rozmiar bufora	128 MB	-	128 MB
maksymalna liczba połączeń	151	32 767	100

4. Środowisko testowe

4.1. Baza danych

Na wszystkich testowanych systemach baza danych opierała się o ten sam schemat przedstawiający model dla sklepu internetowego zajmującego się technologiami konsumenckimi. Struktura bazy zawierała trzynaście tabel, a każda z tabel skupiała się na konkretnym aspekcie dotyczącym wspomnianego biznesu, np. tabela „customer_credit_debit_card" zbierała informacje na temat zapisanych przez użytkownika kartach płatniczych (kredytowych i debetowych), a tabela „customers" na temat danych klientów. Jeśli chodzi o pozostałe obiekty to tabela „orders" zawierała dane o zamówieniach, „order_items" o elementach zamówienia, „products" o produktach, „customer_address" o zapisanych przez użytkownika adresach, „invoices" o fakturach, „receipts" o paragonach, „shipments" o wysyłkach, „shipment_items" o elementach wysyłki, „product_categories" o kategoriach produktów, „employees" o pracownikach, a „payments" o płatnościach. W tabelach uwzględniono takie indeksy jak klucz podstawowy, klucz obcy i wartość unikalna. Dane zaimportowane do bazy danych zostały wygenerowane przy użyciu internetowego narzędzia Mockaroo [15]. Program ten pozwolił na utworzenie rekordów dla tabel zgodnie z obowiązującym schematem. Przy generowaniu danych zostały uwzględnione takie aspekty jak możliwość wstawienia wartości NULL do kolumny oraz typy danych.

4.2. Skrypty

Struktura kodu skryptów była bardzo podobna dla wszystkich trzech systemów, a programy zwracały ten sam rodzaj informacji, tj. tabelę z numerami prób i wartościami czasów dla wskazanego zestawu danych i typu polecenia.

Program opierał się na dwóch pętlach: zewnętrznej kontrolującej liczbę prób i wewnętrznej kontrolującej liczbę wykonań polecenia. Pomiary przeprowadzono w taki sposób, że zmienna „@start_" została umieszczona tuż przed fragmentem kodu testowanej operacji, a zmienna „@end_" zaraz po nim. Obu tym zmiennym przyporządkowano wartości aktualnego czasu mierzonego w sekundach z dokładnością do sześciu miejsc po przecinku. Potem obliczono różnicę i przypisano ją zmiennej „@singleQueryDuration", której wartość dodano do zmiennej „@totalDuration". Po wyjściu z pętli wewnętrznej następowało wprowadzenie uzyskanego rezultatu do tymczasowej tabeli i wyzerowanie zmiennych. Po wykonaniu wszystkich prób program wyświetlił tabelę z wynikami. Co do poleceń, instrukcja SELECT zwracała tylko jeden rekord z informacjami o kliencie, którego zamówienia:

- były o wartości przynajmniej 5 000 USD,
- zostały już dostarczone,
- zostały opłacone przy użyciu karty kredytowej lub debetowej,
- mają paragon z datą wynoszącą najwcześniej 2023-01-15,
- składały się tylko z nadal dostępnych produktów.

W przypadku polecenia aktualizującego rekordy (Listing 1) jedyną zmianą w skrypcie było zastąpienie fragmentu z poleceniem „SELECT" kodem z instrukcją „UPDATE". Instrukcja ta zmniejszała ceny i zwiększała liczbę produktów należących do tej samej kategorii. Podobnie wyglądało przygotowanie skryptu wstawiającego dane. Po zamianie odpowiednich instrukcji program poprawnie zmierzył długości czasów dla polecenia „INSERT", które wprowadzało do tabeli „order_items" jeden rekord z informacjami dla konkretnego zamówienia.

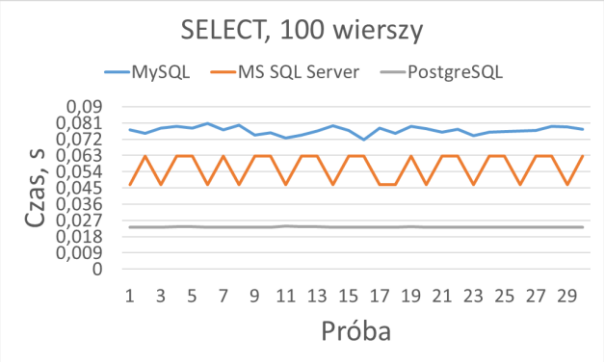
5. Rezultaty

5.1. Zwracanie danych

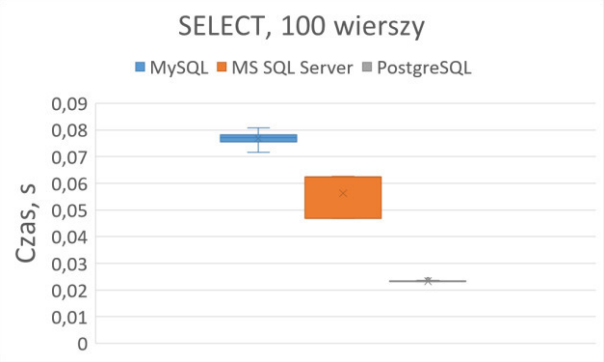
Na Rysunku 1 znajduje się wykres czasów uzyskanych przez trzy systemy bazodanowe dla pierwszego przypadku, czyli dla operacji zwracania 100 rekordów. Tutaj zwycięzcą okazał się PostgreSQL osiągając znacznie mniejsze wartości od konkurencji. Różnica między MS SQL Server wynosiła około dwukrotności długości czasów PostgreSQL, a dla MySQL ponad trzykrotność długości czasów. Zwycięskiemu systemowi udało się utrzymać najlepszą stabilność w swoich wynikach (Rysunek 2). Świadczy o tym nie tylko brak wartości odstających, ale również najwęższa część pudełkowa wykresu, najbliższe położenie mediany i kwartyli, a także najmniejsza różnica pomiędzy wartością maksymalną i minimalną.

Listing 1: Fragment kodu programu mierzącego czasy operacji aktualizacji danych dla systemu MS SQL Server

```
SET @start_ = DATEDIFF_BIG(MICROSECOND,'1970-01-01 00:00:00', SYSDATETIME()) / CAST(1000000 AS DECIMAL(30,6))
UPDATE products
    SET products.product_price = products.product_price * 0.99,
        products.product_quantity_in_stock = products.product_quantity_in_stock + 5
    FROM products
    INNER JOIN product_categories ON
    products.product_categories_id_product_categories = product_categories.id_product_categories
    WHERE product_categories.id_product_categories = 3
SET @end_ = DATEDIFF_BIG(MICROSECOND,'1970-01-01 00:00:00', SYSDATETIME()) / CAST(1000000 AS DECIMAL(30,6))
SET @singleQueryDuration = @end_ - @start_
SET @totalDuration = @totalDuration + @singleQueryDuration
```



Rysunek 1: Wykres liniowy dla operacji zwracania 100 rekordów.

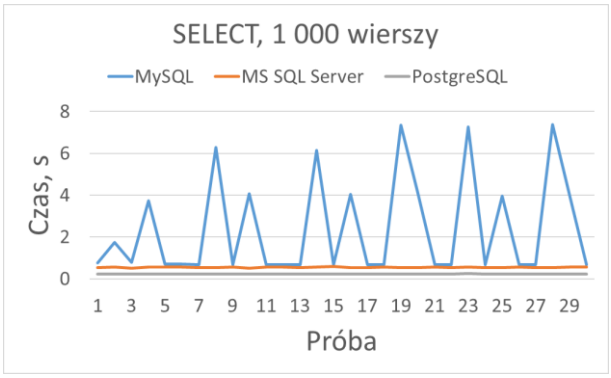


Rysunek 2: Wykres pudełkowy dla operacji zwracania 100 rekordów.

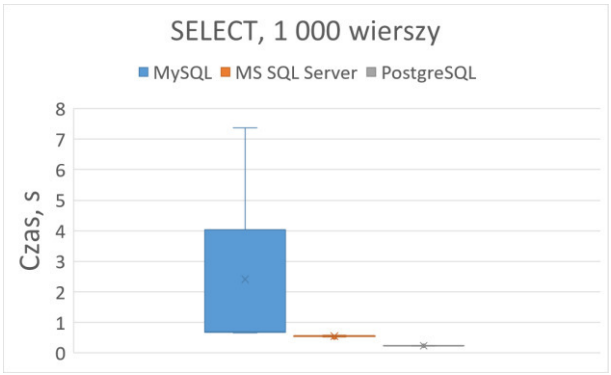
Na Rysunku 3 przedstawiono wykres czasów dla drugiego przypadku - operacji zwracania 1 000 rekordów. Ponownie najlepsze rezultaty otrzymał PostgreSQL. Wartości w tym przypadku były przynajmniej dwukrotnie mniejsze od czasów dla systemu MS SQL Server, a także systemu MySQL. Warto zauważyć poziom stabilności wyników dla PostgreSQL i MS SQL Server (Rysunek 4), które prezentują się bardzo dobrze w porównaniu z ogromną niestabilnością w rezultatach uzyskanych przez MySQL.

Rysunek 5 ilustruje wykres długości czasów trwania prób dla trzeciego przypadku, czyli dla operacji zwracania 10 000 rekordów. PostgreSQL znów zajął pierwsze miejsce, podczas gdy MS SQL Server zajął drugie, a MySQL trzecie. Różnice między rezultatami systemu PostgreSQL a MS SQL Server są ponad dwukrotne, zaś dla MySQL'a są one ponad ośmiokrotne. Stabilność wyników (Rysunek 6) prezentuje się najlepiej dla PostgreSQL i MS SQL Server, a najgorzej dla MySQL.

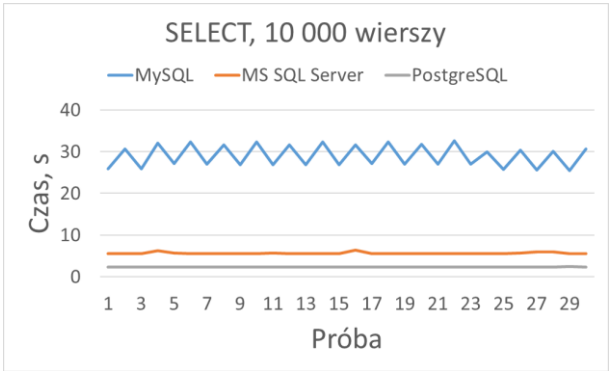
W trzech pierwszych przypadkach badania zaobserwowano, że MySQL zwykle osiągał najgorsze wartości, podczas gdy MS SQL Server znajdował się w środku rankingu, a PostgreSQL zostawał liderem. Wspomnianą zależność zauważono także w temacie stabilności systemów. Najstabilniejsze wartości osiągał PostgreSQL, a najmniej stabilne MySQL.



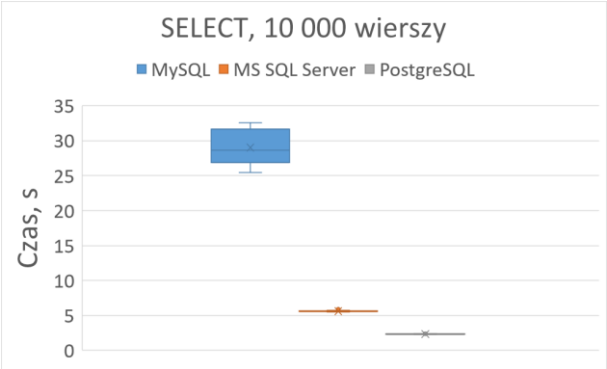
Rysunek 3: Wykres liniowy dla operacji zwracania 1 000 rekordów.



Rysunek 4: Wykres pudełkowy dla operacji zwracania 1 000 rekordów.



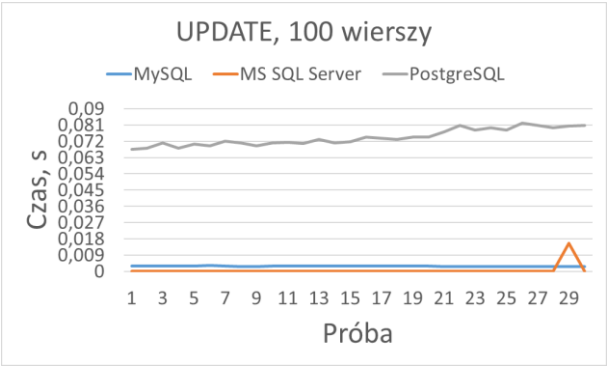
Rysunek 5: Wykres liniowy dla operacji zwracania 10 000 rekordów.



Rysunek 6: Wykres pudełkowy dla operacji zwracania 10 000 rekordów.

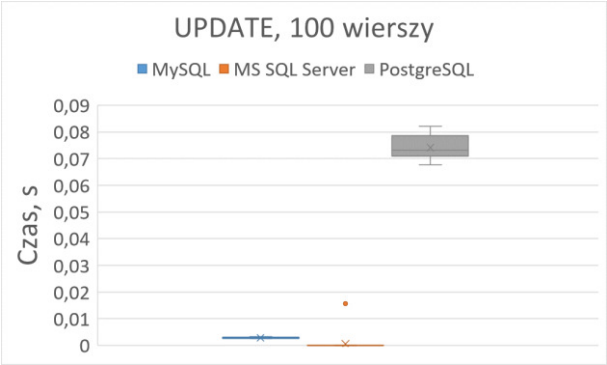
5.2. Aktualizacja danych

Aktualizacja danych to kolejny rodzaj operacji testowany na bazach. Dla przypadku ze stoma wierszami (Rysunek 7) zwycięzcą okazał się system MS SQL Server, który dla ustalonej precyzji do sześciu miejsc po przecinku uzyskiwał wyniki równe 0. Na drugim miejscu znalazł się MySQL, który swoimi rezultatami pokazał sprawność dla tego typu polecenia. Najgorszymi długościami czasów charakteryzował się PostgreSQL. Co ciekawe, czasy dla tego systemu rosły wraz z kolejnymi próbami. Działo się tak, dlatego, że PostgreSQL, przy aktualizacji danych, tworzy nowy wiersz i usuwa stary. Niestety, przy wielokrotnym aktualizowaniu dużej liczby rekordów oryginalne rekordy są nadal przechowywane w bazie, co powoduje zwiększenie jej rozmiaru, a tym samym wydłużenie wykonania zapytania. Różnice między wartościami MS SQL Server i MySQL wynosiły około 0,0028 s, a między MS SQL Server a PostgreSQL około 0,074 s. Największą stabilnością wyników odznaczyły się systemy MS SQL Server i MySQL, a najmniejszą PostgreSQL (Rysunek 8).

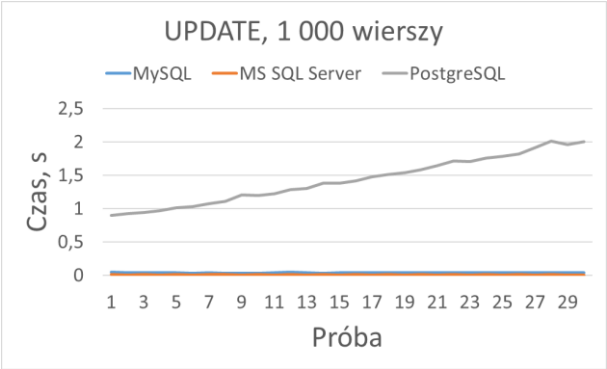


Rysunek 7: Wykres liniowy dla operacji aktualizacji 100 rekordów.

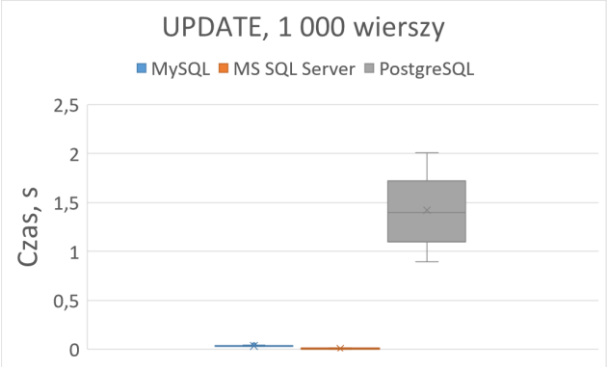
Ranking dla przypadku z 1 000 wierszy wyglądał podobnie jak poprzednio (Rysunek 9). Najkrótszymi czasami wykazał się MS SQL Server, zaraz po nim uplasował się MySQL, a na końcu znalazł się PostgreSQL. MS SQL Server uzyskał część wyników równą zero, a różnice w wartościach między tym systemem a MySQL były na poziomie około 0,028 s, podczas gdy dla PostgreSQL były one równo około 1,42 s. W kwestii stabilności najlepszymi rezultatami odznaczyły się MS SQL Server i MySQL (Rysunek 10).



Rysunek 8: Wykres pudełkowy dla operacji aktualizacji 100 rekordów.



Rysunek 9: Wykres liniowy dla operacji aktualizacji 1 000 rekordów.

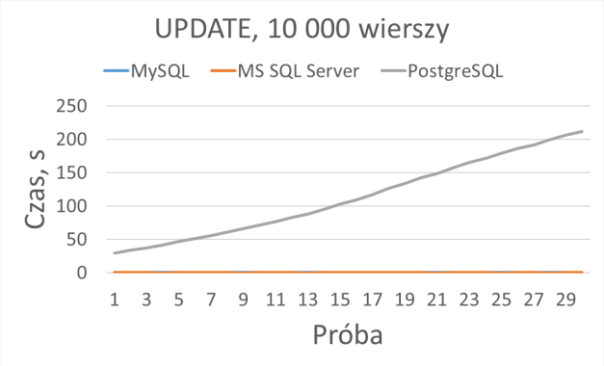


Rysunek 10: Wykres pudełkowy dla operacji aktualizacji 1 000 rekordów.

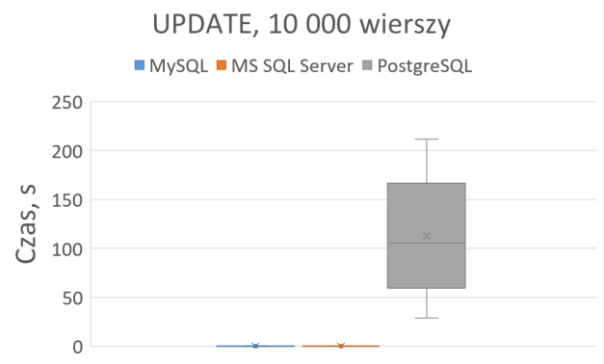
Kolejnym badanym przypadkiem była aktualizacja 10 000 wierszy. Tutaj wykres wyglądał podobnie do wykresów dla poprzednich przypadków (Rysunek 11). Powtórnie zwycięzcą okazał się MS SQL Server, MySQL zajął drugie miejsce, a PostgreSQL trzecie. Różnice w długościach czasów między systemem MS SQL Server a MySQL były niewielkie (około 0,2 s), a między MS SQL Server a PostgreSQL sięgały one kilkudziesięciu sekund. Wykres systemu PostgreSQL odznaczył się trendem wzrostowym, a w kwestii stabilności wyników najlepiej poradziły sobie MySQL i MS SQL Server (Rysunek 12).

Ranking wydajności dla aktualizacji rekordów był zbieżny dla trzech zestawów danych. MS SQL Server osiągał najlepsze wyniki, MySQL plasował się na drugiej pozycji, a PostgreSQL był na końcu. Ponadto, różnice w rezultatach były najmniejsze między MS SQL Server a MySQL, podczas gdy wyniki dla PostgreSQL

odstawały znacząco od konkurencji. Jeśli chodzi o stabilność uzyskanych wartości najlepsze okazały się systemy MS SQL Server i MySQL.



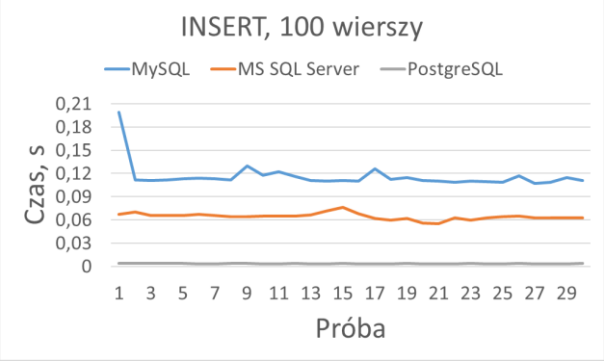
Rysunek 11: Wykres liniowy dla aktualizacji 10 000 rekordów.



Rysunek 12: Wykres pudełkowy dla aktualizacji 10 000 rekordów.

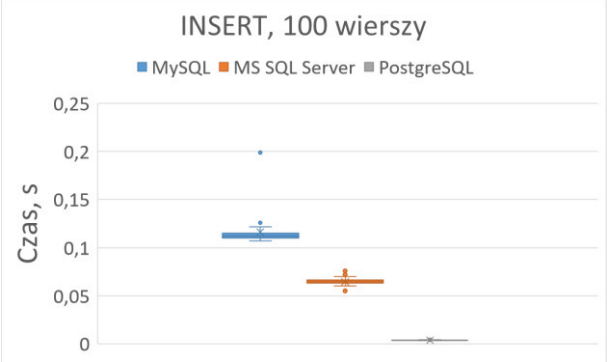
5.3. Wstawianie danych

Następnym badanym typem polecenia było wstawianie danych. Dla przypadku ze 100 wierszami (Rysunek 13) najlepsze rezultaty osiągnął PostgreSQL, a najgorsze MySQL. MS SQL Server zajął drugie miejsce. Różnice w wynikach były całkiem spore, ponieważ dla PostgreSQL i MS SQL Server oscylowały na poziomie 0,06 s, a dla PostgreSQL a MySQL wynosiły średnio 0,11 s. W kwestii stabilności rezultatów najlepszy okazał się PostgreSQL, podczas gdy MS SQL Server i MySQL cechowały większe dysproporcje w uzyskanych czasach (Rysunek 14).

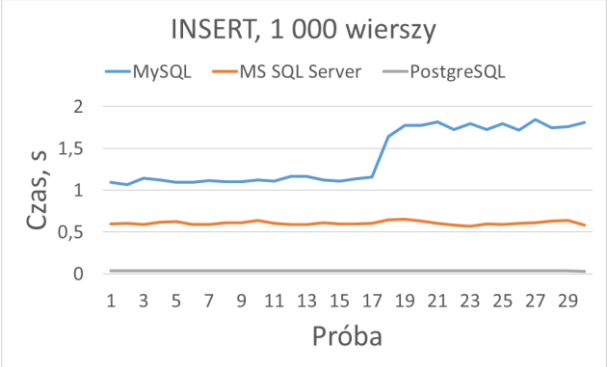


Rysunek 13: Wykres liniowy dla operacji wstawiania 100 rekordów.

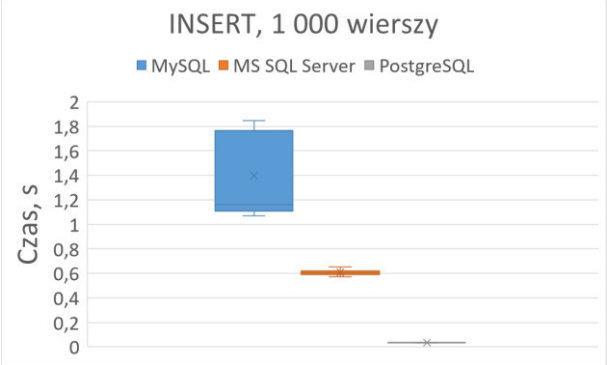
Po wstawieniu 1 000 rekordów miejsca na podium wyglądały identycznie jak poprzednio (Rysunek 15). PostgreSQL został zwycięzcą, MS SQL Server znalazł się zaraz po nim, a na końcu uplasował się MySQL. Różnice między wynikami ponownie były dość znaczące. Dla PostgreSQL i MS SQL Server były one równe około 0,57 s, a dla PostgreSQL i MySQL 1,36 s. Najlepszą stabilnością rezultatów mogły pochwalić się PostgreSQL i MS SQL Server (Rysunek 16). Natomiast MySQL po siedemnastej próbie zauważalnie wydłużył czasy wstawiania wierszy.



Rysunek 14: Wykres pudełkowy dla operacji wstawiania 100 rekordów.



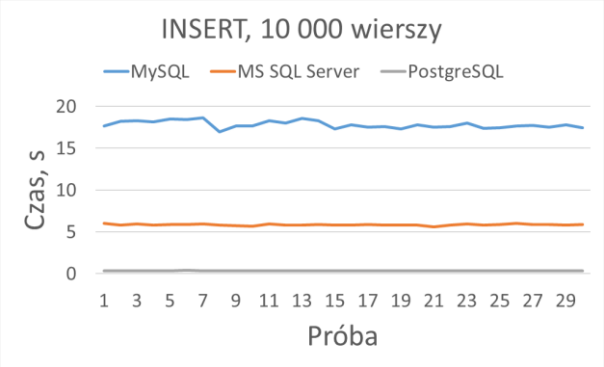
Rysunek 15: Wykres liniowy dla operacji wstawiania 1 000 rekordów.



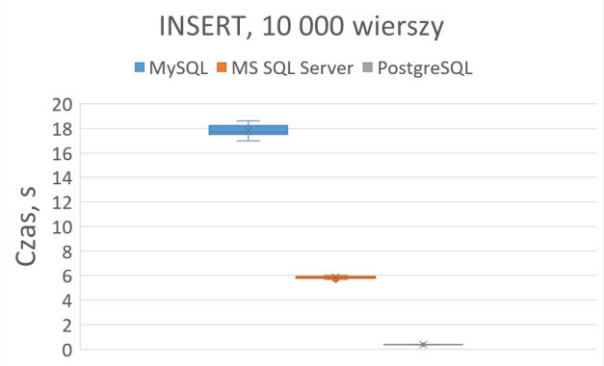
Rysunek 16: Wykres pudełkowy dla operacji wstawiania 1 000 rekordów.

Wstawienie 10 000 rekordów nie spowodowała zmian na podium (Rysunek 17). Najwydajniejszy znowu okazał się PostgreSQL, drugie miejsce zajął MS SQL Server, a trzecie MySQL. Różnice w wynikach między PostgreSQL a MS SQL Server były na poziomie około 5,48 s., a między PostgreSQL a MySQL wynosiły one około 17,83 s. W przypadku stabilności czasów

(Rysunek 18) powtórnie najlepsze były systemy PostgreSQL i MS SQL Server.



Rysunek 17: Wykres liniowy dla operacji wstawiania 10 000 rekordów



Rysunek 18: Wykres pudełkowy dla operacji wstawiania 10 000 rekordów.

Ranking systemów dla wstawiania wierszy wyglądał identycznie dla wszystkich trzech zestawów danych. PostgreSQL osiągał najkrótsze czasy, podczas gdy MS SQL Server i MySQL zajmowali odpowiednio drugie i trzecie miejsce. Różnice między wynikami były dość spore zarówno dla PostgreSQL i MS SQL Server jak i dla PostgreSQL i MySQL. Pod względem stabilności rezultatów najlepsze okazały się PostgreSQL i MS SQL Server.

6. Analiza i wyniki badania

W celu wyznaczenia najbardziej wydajnego systemu zarządzania bazami danych należało wykonać test statystyczny udowadniający istnienie statystycznie istotnych różnic między trzema zestawami danych. Zdecydowano się skorzystać z testu Kruskala-Wallisa [16] oraz internetowego narzędzia o nazwie „Online Web Statistical Calculators ..... for Categorical Data Analysis" [17], które pozwoliło to sprawdzić. Otrzymane rezultaty potwierdziły niezależność trzech prób od siebie dla wszystkich dziewięciu przypadków. Dla układu z zapytaniem SELECT i 100 wierszami (Tabela 3) wartość p-value wyniosła dużo mniej niż założony poziom istotności 0,05, dlatego odrzucono hipotezę zerową mówiącą o tym, że wszystkie próbki pochodzą z tego samego rozkładu na rzecz hipotezy alternatywnej, która zakładała, że jedna lub więcej próbek pochodzi z innego rozkładu. Dodatkowo, przeprowadzono testy post-hoc Dunna [18] w celu stwierdzenia pomiędzy którymi grupami pojawiły się istotne różnice (Tabela 4). Wszystkie wartości osiągnęły poziom niższy niż 0,05, zatem uznano, że wszystkie trzy zestawy danych są od siebie różne.

Tabela 3: Wynik testu Kruskala-Wallisa dla przypadku z zapytaniem SELECT i 100 wierszami

statystyka chi-kwadrat	stopnie swobody	wartość p-value
79,15	2	$6{,}5 \cdot 10^{-18}$

Tabela 4: Test post-hoc Dunna dla przypadku z zapytaniem SELECT i 100 wierszami

	MS SQL Server	MySQL
MySQL	0,000017	
PostgreSQL	0,000017	$1{,}73 \cdot 10^{-18}$

Wskazanie zwycięzcy dla każdego z przypadków opierało się na zliczeniu najkrótszych długości czasów dla wszystkich prób. System, który w większości prób osiągnął najlepsze wartości zostawał zwycięzcą dla danego przypadku. Uzyskane w ten sposób wyniki wykorzystano do utworzenia porównania wydajności systemów zarządzania bazami danych (Tabela 5). PostgreSQL okazał się być najszybszy aż w sześciu przypadkach, a MS SQL Server w trzech. MySQL nie zajął pierwszego miejsca ani razu. Tym samym system PostgreSQL został wskazany jako najwydajniejszy dla całego badania. Na tej podstawie należy odrzucić tezę pracy typującą MS SQL Server jako zwycięzcę. Rezultaty badania pozwoliły również odpowiedzieć na postawione pytania badawcze. Dla zapytania zwracającego i wstawiającego rekordy najszybszy był PostgreSQL, a dla zapytania aktualizującego wiersze MS SQL Server.

Tabela 5: Porównanie systemów bazodanowych

Przypadek	MySQL	MS SQL Server	PostgreSQL
SELECT, 100 wierszy	-	-	+
SELECT, 1 000 wierszy	-	-	+
SELECT, 10 000 wierszy	-	-	+
UPDATE, 100 wierszy	-	+	-
UPDATE, 1 000 wierszy	-	+	-
UPDATE, 10 000 wierszy	-	+	-
INSERT, 100 wierszy	-	-	+
INSERT, 1 000 wierszy	-	-	+
INSERT, 10 000 wierszy	-	-	+

7. Podsumowanie i wnioski

Celem pracy było przeprowadzenie analizy wydajnościowej trzech relacyjnych systemów zarządzania bazami danych: MySQL, PostgreSQL i Microsoft SQL Server. Badanie polegało na wykonaniu testów wykorzystujących wcześniej utworzone skrypty mierzące długości czasów operacji. Analizowane typy zapytań składały się z polecenia zwracającego dane (select), polecenia aktualizującego dane (update) i polecenia

wstawiającego dane (insert). Eksperyment przeprowadzono z użyciem trzech zestawów danych obejmujących kolejno 100, 1 000 i 10 000 rekordów. W testach uwzględniono dziewięć przypadków powstałych w oparciu o rodzaj polecenia i wariant zestawu danych. Schemat bazy danych zawierał trzynaście tabel i przedstawiał model dla sklepu internetowego zajmującego się technologiami konsumenckimi. W badaniu wykonano 35 prób dla każdego z przypadków, przy czym pierwsze pięć prób pominięto ze względu na kwestie optymalizacji bazy danych i kwestie przechowywania danych w pamięci podręcznej. Najwydajniejszym oprogramowaniem okazał się PostgreSQL. Osiągnął on najlepsze rezultaty dla sześciu przypadków. MS SQL Server wygrał tylko w trzech przypadkach, a MySQL nie zwyciężył ani razu. Odpowiedziano również na pytania badawcze postawione w pracy. PostgreSQL był najszybszy dla zapytania zwracającego i wstawiającego rekordy, a MS SQL Server dla zapytania aktualizującego wiersze.

Na podstawie przeprowadzonych badań sformułowano następujące wnioski:

1. PostgreSQL charakteryzował się bardzo dobrym poziomem wydajności i dobrą stabilnością dla operacji zwracania i wstawiania danych.
2. Microsoft SQL Server charakteryzował się bardzo dobrym poziomem wydajności dla operacji aktualizacji danych.
3. Microsoft SQL Server wyróżnił się dobrym poziomem wydajności i stabilności dla wszystkich testowanych rodzajów operacji.
4. MySQL cechowała zauważalna niestabilność w wydajności w przypadku zwracania większej ilości danych.
5. Nieprzerwane i wielokrotne aktualizowanie rekordów w systemie PostgreSQL spowodowało wydłużenie czasu wykonania tej operacji.

Badanie pozostawiło wiele obszarów, w których mogłoby znaleźć swój dalszy ciąg. Takie kwestie jak identyfikacja tzw. „wąskich gardeł" czy niewydajne indeksowanie to część niesprawdzonych aspektów wpływających na wydajność systemów zarządzania bazami danych. Ewentualna kontynuacja badania mogłaby pomóc wyjaśnić m.in. przyczyny niestabilności testowanych systemów oraz ich zachowania, a także pozwoliłaby znaleźć sposoby zwiększenia ich wydajności.

Literatura

[1] M. Grudzień, K. Korgol, D. Gutek, Porównanie możliwości wykorzystania oraz analiza wydajności baz danych na systemach mobilnych, praca magisterska, Politechnika Lubelska, Lublin, 2016.

[2] R. Kleweka, W. Truskowski, M. Skublewska-Paszkowska, Porównanie wydajności baz danych MySQL, MSSQL, PostgreSQL oraz Oracle z uwzględnieniem wirtualizacji, praca magisterska, Politechnika Lubelska, Lublin, 2020.

[3] K. Lachewicz, Analiza wydajności systemów bazodanowych: MySQL, MS SQL, PostgreSQL w kontekście aplikacji internetowych, praca magisterska, Politechnika Lubelska, Lublin, 2020.

[4] S. Stets, G. Kozieł, Analiza porównawcza wydajności baz danych pracujących pod kontrolą systemu Windows, praca magisterska, Politechnika Lubelska, Lublin, 2019.

[5] S. Kulshrestha, S. Sachdeva, Performance comparison for data storage - Db4o and MySQL databases, 2014 Seventh International Conference on Contemporary Computing (IC3) (2014) 166-170.

[6] R. Poljak, P. Pošcić, D. Jakšić, Comparative Analysis of the Selected Relational Database Management Systems, 2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO) (2017) 1496-1500.

[7] R. Kleweka, W. Truskowski, M. Skublewska-Paszkowska, Porównanie wydajności baz danych MySQL, MSSQL, PostgreSQL oraz Oracle z uwzględnieniem wirtualizacji, Journal of Computer Sciences Institute 16 (2020) 279-284.

[8] Y. Abubakar, Benchmarking popular open source RDBMS: a performance evaluation for IT professionals, International Journal of Advanced Computer Technology (IJACT) 3 (2014) 39-44.

[9] S. Tongkaw, A. Tongkaw, A comparison of database performance of MariaDB and MySQL with OLTP workload, 2016 IEEE Conference on Open Systems (ICOS) (2016) 117-119.

[10] M. -G. Jung, S. -A. Youn, J. Bae, Y. -L. Choi, A Study on Data Input and Output Performance Comparison of MongoDB and PostgreSQL in the Big Data Environment, 2015 8th International Conference on Database Theory and Application (DTA) (2015) 14-17.

[11] M. M. Eyada, W. Saber, M. M. El Genidy, F. Amer, Performance Evaluation of IoT Data Management Using MongoDB Versus MySQL Databases in Different Cloud Environments, IEEE Access 8 (2020) 110656-110668.

[12] H. Fatima, K. Wasnik, Comparison of SQL, NoSQL and NewSQL databases for internet of things, 2016 IEEE Bombay Section Symposium (IBSS) (2016) 1-6.

[13] M. Meekyung, Experiments of Search Query Performance for SQL-Based Open Source Databases, International Journal of Internet, Broadcasting and Communication 10 (2018) 31-38.

[14] R. Almeida, P. Furtado, J. Bernardino, Performance Evaluation MySQL InnoDB and Microsoft SQL Server 2012 for Decision Support Environments, Proceedings of the Eighth International C* Conference on Computer Science & Software Engineering (2015) 56-62.

[15] Generator makiet danych „Mockaroo", https://www.mockaroo.com, [10.05.2023].

[16] W. H. Kruskal, W. A. Wallis, Use of Ranks in One-Criterion Variance Analysis, Journal of the American Statistical Association 47 (1952) 583-621.

[17] Internetowy kalkulator testów statystycznych „Online Web Statistical Calculators ..... for Categorical Data Analysis", https://astatsa.com/KruskalWallisTest/, [13.06.2023].

[18] O. J. Dunn, Multiple Comparisons Using Rank Sums, Technometrics 6 (1964) 241-252.