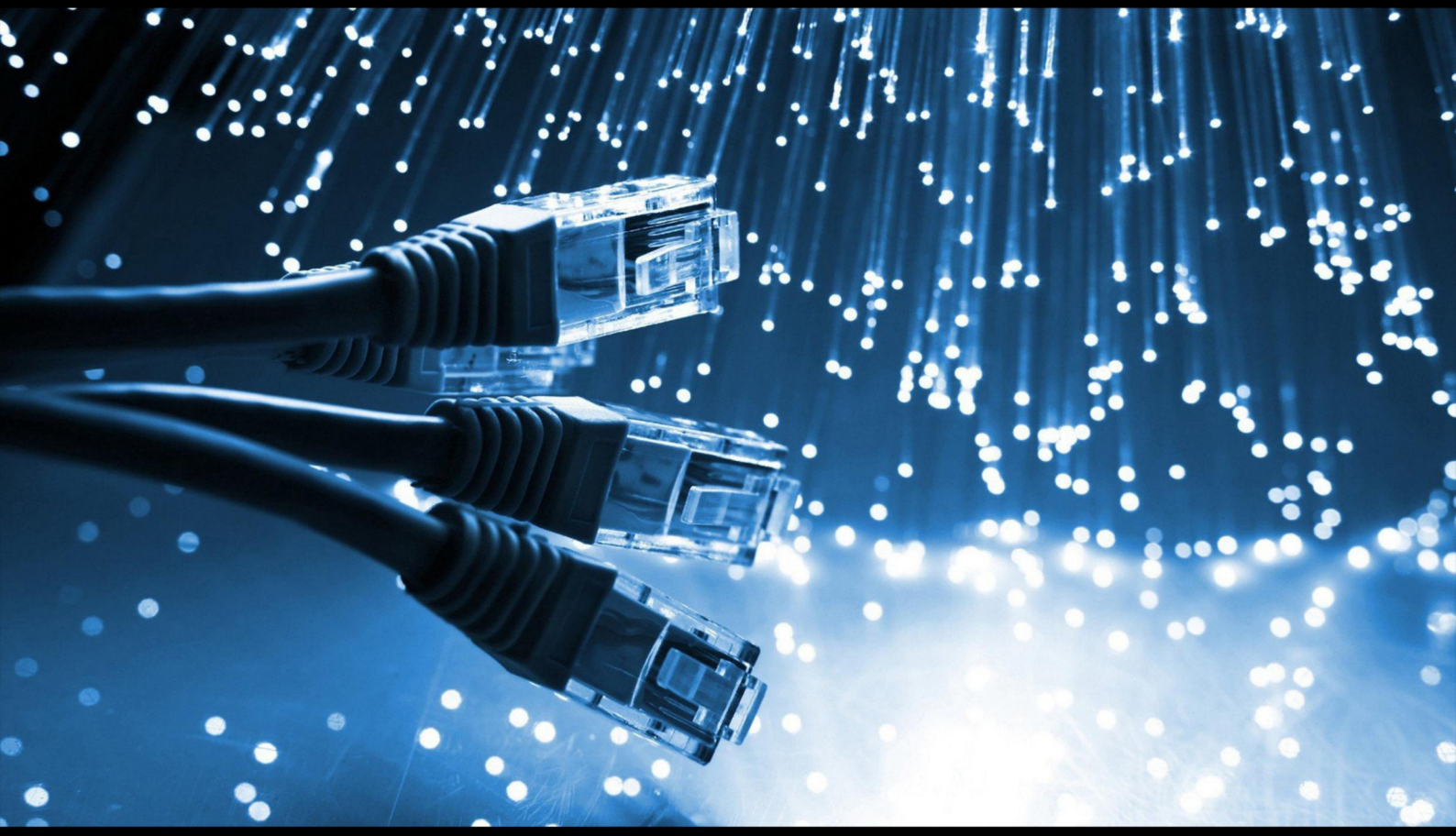
JCSI

Journal of Computer Sciences Institute

Volume 30/2024



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

Redaktorzy techniczni:
Beata Pańczyk
e-mail: b.panczyk@pollub.pl
Magdalena Zoła
e-mail: m.zola@pollub.pl

Recenzenci numeru:

dr inż. Małgorzata Plechawska-Wójcik
dr inż. Jacek Kęsik
dr inż. Elżbieta Miłosz
dr inż. Tomasz Szymczyk
dr inż. Tomasz Nowicki
dr inż. Marek Miłosz, prof. PL
dr inż. Dariusz Gutek
dr Marcin Barszcz

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 editors:
Beata Pańczyk
e-mail: b.panczyk@pollub.pl
Magdalena Zoła
e-mail: m.zola@pollub.pl

Reviewers:

Małgorzata Plechawska-Wójcik
Jacek Kęsik
Elżbieta Miłosz
Tomasz Szymczyk
Tomasz Nowicki
Marek Miłosz, prof. PL
Dariusz Gutek
Marcin Barszcz

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

Projekt okładki:
Marta Zbańska

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 30 (2024) 1-8
Received: 10 July 2023
Accepted: 3 November 2023

Analysis of data processing efficiency with use of Apache Hive and Apache Pig in Hadoop environment

Analiza efektywności przetwarzania danych z użyciem Apache Hive i Apache Pig w środowisku Hadoop

Mikołaj Skrzypczyński*, Piotr Muryjas

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

Abstract
The aim of this paper is the analysis of data processing efficiency with use of Apache Hive and Apache Pig in Hadoop environment. The analysis was based on comparison between both mentioned tools with use of large data set, represented by 28 million records. Research was provided with use of scripts and queries destined for Apache Hive and Apache Pig, and then executed 10 times on environment brought by created virtual machine. Those methods were performed on the same data sets for 16 times according to previously prepared research scenarios. As the conclusion, authors had observed that Apache Hive is more efficient tool, than Apache Pig.

Keywords: data processing; Apache Hive; Apache Pig, Hadoop

Streszczenie
Celem niniejszej pracy jest analiza efektywności przetwarzania danych z użyciem Apache Hive i Apache Pig w środowisku Hadoop. Analiza polegała na porównaniu pomiędzy obydwoma wspomnianymi narzędziami z użyciem dużych zbiorów danych, w formie 28 milionów rekordów. Badanie zostało przeprowadzone z użyciem skryptów i zapytań przeznaczonych dla Apache Hive oraz Apache Pig, a następnie wykonanie dziesięciokrotnie na środowisku dostarczonym dzięki utworzonej maszynie wirtualnej. Wymienione metody zostały uskutecznione na tych samych zbiorach danych 16 razy, zgodnie z uprzednio przygotowanymi scenariuszami badawczymi. W rezultacie autorzy zaobserwowali, iż Apache Hive jest bardziej efektywnym narzędziem, niż Apache Pig.

Słowa kluczowe: przetwarzanie danych; Apache Hive; Apache Pig; Hadoop

*Corresponding author
Email address: mikolaj.skrzypczynski@pollub.edu.pl (M. Skrzypczyński)

1. Wstęp

Ilość danych na świecie wzrasta z dnia na dzień, co wiąże się z coraz większym skomplikowaniem procesów związanych z ich przetwarzaniem. Istotnym aspektem w dostępie do danych jest efektywność korzystania z nich, w celu uzyskania zakładanych efektów przy określonej złożoności obliczeniowej, a co za tym idzie również czasowej. Odpowiedzią na te zagadnienia jest rozwiązanie Big Data, które to nastawione jest na wydajną pracę z dużymi zbiorami danych.

Celem niniejszej pracy jest analiza efektywności przetwarzania danych z użyciem Apache Hive i Apache Pig w środowisku Hadoop. Apache Hadoop jest środowiskiem mającym na względzie umożliwienie pracy na klastrach komputerów, w celu procesowania wielkich zestawów danych z wykorzystaniem złożonych narzędzi takich jak Apache Hive czy też Apache Pig.

Bazując na rosnącej popularności wyżej wymienionych rozwiązań, autorzy niniejszej pracy postanowili zbadać ich efektywność. Jest to kluczowy czynnik wpływający na wydajność korzystania z tych narzędzi. W efekcie badany aspekt jest istotną składową, w kwestii rozważania, której technologii użyć poza jej składnią zapytań czy też łatwością konfiguracji.

Do przeprowadzenia badań: dokonano przeglądu literatury, skonfigurowano środowisko, przygotowano scenariusze badawcze, a następnie ich implementację z wykorzystaniem skryptów Apache Pig i zapytań Apache Hive, podejmując zarówno aspekt teoretyczny jak i praktyczny zagadnienia.

2. Przegląd literatury

Niniejszy rozdział zawiera analizę literatury obejmującej tematykę narzędzi Big Data: Apache Hadoop, Apache Pig i Apache Hive.

W pracy pt. „*Analyzing Performance of Apache Pig and Apache Hive with Hadoop*” autorzy porównali narzędzia Apache Hive i Apache Pig pod względem wydajności i złożoności zapytań. Jako dane wejściowe dla badań wybrano duży zbiór danych zawierający informacje z ośrodków lekarskich w Stanach Zjednoczonych dotyczących złych praktyk wykonywanych przez lekarzy. Na ich podstawie autorzy próbowali dowiedzieć się, w jakich okolicznościach powinno się używać narzędzia Apache Pig, a w jakich Apache Hive. Narzędzia zostały porównane w wymiarach sposobu przechowywania danych, ekstrakcji i przekształcania danych oraz opłacalności użycia. Autorzy wykazali, że pod względem opłacalności użycia oba narzędzia są na tym samym poziomie. Natomiast w kwestii przechowywania danych Hive jest lepszym wyborem, a w przypadku ekstrakcji i przekształcania danych

lepszy jest Pig. Narzędzia zostały porównane także pod względami czasów wykonania oraz liczby linijek kodu dla każdego ze zdefiniowanych zapytań. Ostateczne wyniki przeprowadzonych eksperymentów pokazały, że Hive jest wydajniejszy, wygodniejszy i mniej skomplikowany w użyciu niż Pig [1].

Autorzy pracy „*Performance Analysis of ECG Big Data using Apache Hive and Apache Pig*" porównali wydajność Apache Pig i Apache Hive w kontekście analizy danych EKG pacjentów pochodzących ze szpitali. Celem badania było wykazanie, które narzędzie jest lepsze pod względem szybkości przetwarzania i optymalizacji czasu zapytań. Wynik badań miał za zadanie ułatwić koncernom medycznym mającym styczność z danymi EKG wybór narzędzia używanego do analizy danych. Zbadane zostały dwa aspekty, które mogą mieć wpływ na wynik badań, którymi są kolejno rozmiar zestawu danych oraz czas wykonywania zapytania w celu pozyskania określonej informacji. Przeprowadzone badania wykazały zależność między zbiorem danych, a czasem przetwarzania. Wraz ze wzrostem rozmiaru danych rośnie czas ich przetwarzania. Do porównania narzędzi Apache Pig i Apache Hive wybrano dwie próbki danych zawierających odpowiednio 5076 oraz 15230 rekordów. Czasy wykonania podobnych zapytań na mniejszym zbiorze prezentowały się następująco: dla Pig było to 3,00 sekundy, natomiast dla Hive 46,66 sekund. Natomiast dla większego zbioru danych było to odpowiednio 9,00 sekund oraz 55,69 sekund. Autorzy, wykazali, że czas wykonywania zapytań przez Apache Hive jest zdecydowanie wyższy w porównaniu do Apache Pig. Ostateczny werdykt określił Pig znacznie wydajniejszym narzędziem [2].

W artykule pt. „*Processing performance on Apache Pig, Apache Hive and MySQL cluster*" badacze porównują czas przetwarzania danych z użyciem narzędzi Hive, Pig i MySQL Cluster przy zbiorze danych charakteryzującym się prostą strukturą. Celem badania było określenie, które z narzędzi jest najwydajniejsze. Autorzy wykazali, że na niedrogim sprzęcie Hive jest znacznie szybszy od MySQL Cluster oraz jest w stanie obsługiwać obszerne zbiory danych. Natomiast dla wybranego zestawu danych Apache Pig wydaje się nieodpowiednim narzędziem. Według autorów Apache Pig znajduje lepsze zastosowanie dla złożonych zapytań i jeszcze większych zestawów danych niż użyty w badaniach. Eksperyment pokazał, że zapytania MySQL Cluster są szybsze niż zapytania Hive i Pig do pewnego momentu. Jednakże, kiedy liczba rekordów zestawów danych zwiększa się, to narzędzie Apache Hive staje się bardziej niezawodne. Jedną z przeszkód klastra MySQL jest to, że wraz ze wzrostem rozmiaru danych wymaga on więcej pamięci RAM do ich przetwarzania. Indeksowane kolumny są zawsze przechowywane w pamięci. Im większy rozmiar danych, tym więcej pamięci lub maszyny będzie potrzebować. Wyniki eksperymentu pokazały, że Apache Hive pokonuje wydajność MySQL Cluster i Apache Pig. Jednakże autorzy pracy zaznaczają, że wydajność narzędzi może ulec zmianie w zależności od klasy używanego sprzętu. Przy większej ilości pamięci RAM wydajność klastra MySQL mogłaby być większa i przewyższać inne narzędzia [3].

Wyniki każdej z wymienionych prac świadczą o tym, że nie ma ostatecznego zwycięzcy pośród badanej dwójki narzędzi, którymi są Pig i Hive. Wydajność narzędzi może być odmienna zależnie od dostępnego środowiska, wielkości zbiorów danych oraz złożoności tworzonych zapytań. Niniejsza praca oraz zawarte w niej wyniki badań mogą stanowić pomoc przy decyzji w wyborze odpowiedniego narzędzia do przetwarzania i analizy danych.

3. Narzędzia Big Data w ekosystemie Hadoop

W niniejszym rozdziale przedstawiono technologie i narzędzia Big Data powiązane z tematem badań, a dokładniej: środowisko Apache Hadoop, technologię MapReduce oraz narzędzia Pig i Hive.

3.1. Środowisko Apache Hadoop

Apache Hadoop jest otwarto-źródłowym szkieletem aplikacyjnym rozwijanym przez organizację Apache Software Foundation, której przeznaczeniem jest przetwarzanie i przechowywanie dużych zbiorów danych. Szkielet udostępnia możliwości rozproszonego przetwarzania Big Data w klastrach komputerów przy użyciu prostych modeli programowania. Zaprojektowano go w taki sposób, aby liczba serwerów stosowana w implementowanym rozwiązaniu mogła być skalowalna od pojedynczej do nawet tysięcy maszyn [4].

Projekt Apache Hadoop składa się z kilku modułów, z których każdy ma inne zastosowanie. Tymi modułami są: HDFS – rozproszony system plików, YARN – zarządca zasobów i zadań oraz MapReduce – model programowania służący do przetwarzania dużej ilości danych w sposób równoległy na którym został oparte narzędzia Apache Pig i Apache Hive [4].

Otwarto-źródłowość projektu przyczyniła się do powstania społeczności miłośników tej technologii oraz zwiększenia liczby osób zaangażowanych w rozwój projektu. Wokół projektu zaczęło pojawiać się wiele innych projektów i narzędzi powiązanych ze szkieletem aplikacyjnym Hadoop. Znaczna część powstałych rozwiązań na stałe zagościła w projekcie Hadoop i stała się jego integralną częścią. Zbiór narzędzi i projektów zbudowanych dookoła opisywanego środowiska nazywa się Hadoop Ekosystem. Wymieniony zbiór daje możliwość wykonywania jeszcze bardziej zaawansowanej analizy danych i pozwala na lepsze wykorzystanie potencjału samego środowiska Hadoop [5].

3.2. Apache MapReduce

Apache Hadoop MapReduce jest szkieletem aplikacyjnym umożliwiającym tworzenie aplikacji, które przetwarzają równolegle ogromne zbiory danych na dużych klastrach sprzętu komputerowego w sposób niezawodny i odporny na błędy.

Hadoop MapReduce zazwyczaj dzieli zbiór danych wejściowych na niezależne fragmenty, które są potem przetwarzane przez zadania mapowania w sposób

równoległy, następnie wyniki mapowania są sortowane i wprowadzane do zadań redukcji. Najczęściej zarówno dane wejściowe jak i wyjściowe są przechowywane w systemie plików HDFS [6].

Szkielet MapReduce zajmuje się planowaniem i monitorowaniem zadań oraz ponownym wykonywaniem, tych które się nie powiodły. MapReduce i HDFS działają na tych samych zestawach węzłów, co oznacza, że węzły przeznaczone do obliczeń i węzły magazynowe są tymi samymi. Taka konfiguracja pozwala na efektywne zaplanowanie zadań na węzłach, na których już znajdują się dane, co skutkuje bardzo wysoką łączną przepustowością w całym klastrze [6].

3.3. Apache Hive

Apache Hive to projekt, którego celem jest uproszczenie przetwarzania dużych zbiorów danych. Jest skierowany w szczególności dla użytkowników korzystających ze środowiska Hadoop oraz narzędzi typu business intelligence. Projekt dostarcza interfejs zbliżony do języka SQL dla technologii Hadoop MapReduce [7].

Rozwiązanie to wspiera właściwości ACID (ang. atomicity, consistency, isolation, durability) oraz indeksowanie, jednakże nie jest przeznaczone do zadań typu OLTP (ang. Online Transaction Processing). Język zapytań dostarczany z projektem nosi nazwę HQL (ang. Hive Query Language). Jego semantyka i funkcje przypominają standard języka SQL dla relacyjnych baz danych [7].

Struktura metadanych Hive pozwala na konstruowanie wysokopoziomowych struktur na szczycie systemu plików HDFS, złudnie podobnych do tabel w typowych rozwiązaniach magazynowych tabele odpowiadają katalogom danych w HDFS [7].

Praca z Hive nie wymaga wcześniejszego tworzenia schematów bazodanowych przed umieszczeniem danych w magazynie. Schematy metadanych Hive są zazwyczaj tworzone w sposób ad-hoc przy wczytywaniu zbiorów danych. Rozwiązanie zapewnia elastyczność i wydajność w pracy z danymi, a także dostarcza gotowy do użycia, zoptymalizowany i prosty w obsłudze model zapytań nie wymagający złożonych operacji programistycznych jak w przypadku MapReduce [7].

3.4. Apache Pig

Apache Pig jest otwarto-źródłową platformą przeznaczoną do analizy i przekształcania zbiorów danych. Na platformę składają się proceduralny język skryptowy Pig Latin, oraz środowisko uruchomieniowe. Narzędzie posiada nowatorskie środowisko do debugowania, którego przydatność może być szczególnie doceniona przy pracy z ogromnymi zbiorami danych [8].

Pig Latin zapewnia: wsparcie dla elastycznych zagnieżdżonych modeli danych, wsparcie dla funkcji definiowanych przez użytkownika oraz możliwość przeprowadzania operacji na plikach bez informacji o strukturze danych. Język ten został zaprojektowany tak, aby mógł obsługiwać wiele najpopularniejszych prostych typów danych takich jak ciągi znaków, liczby całkowite i zmiennoprzecinkowe oraz złożonych typów danych takich jak krotki, kolekcje krotek, mapy. Jego ideą jest umożliwienie intuicyjnego i łatwego modelowania przepływów danych i ich przekształcania [8].

Program napisany w Pig Latin składa się z sekwencji instrukcji opisujących operacje przetwarzania zbioru danych w sposób potokowy. Operacje te z pomocą kompilatora po uprzedniej analizie składni i poprawności kodu, są optymalizowane i konwertowane na zadania MapReduce, które następnie są uruchamiane. Pig umożliwia równoległe przetwarzanie ogromnych zbiorów danych oraz wykonywanie operacji ETL na danych pochodzących z różnych źródeł. Dane wynikowe zapisywane są w HDFS [2].

4. Środowisko badawcze

W celu oceny efektywności przetwarzania danych z użyciem Apache Hive i Apache Pig w środowisku Hadoop do przeprowadzenia badań wykorzystano jeden z dostępnych na rynku wariantów oprogramowania implementujących środowisko Hadoop – Cloudera. Środowisko Cloudera może być zainstalowane, uruchomione i konfigurowane na fizycznej stacji roboczej lub w postaci gotowej do użycia maszyny wirtualnej. Do badań wykorzystano maszynę wirtualną, którą uruchomiono w środowisku VMWare Workstation 17 Player [9]. Konfigurację stacji roboczej przedstawiono w Tabeli 1.

Tabela 1: Konfiguracja fizycznej maszyny

Właściwość	Opis
Model laptopa	Lenovo Thinkpad T14 Gen 3
System operacyjny	Windows 11 Pro 22H2
Środowisko wirtualizacji	VMware Workstation 17 Player
Procesor	Intel Core i7-1260P 2.10 GHz, 12 procesorów, 16 wątków
Pamięć RAM	2x16 GB, Kingston, DDR4, 3200 MHz
Dysk twardy	SK Hynix PC711 512GB SSD, PCIe GEN 3 x4.0 NVMe

Aby zmaksymalizować wydajność maszyny wirtualnej do jej konfiguracji przydzielono największą możliwą ilość zasobów dostępnych w maszynie hosta umożliwiających stabilną pracę. Cloudera Distributed Hadoop (CDH) to w pełni otwarto-źródłowa dystrybucja platformy Cloudera zawierająca środowisko Apache Hadoop oraz ważne narzędzia powiązane ze środowiskiem Hadoop w których skład wchodzą m.in. Apache Pig i Apache Hive [10]. Parametry maszyny wirtualnej uruchamiającej środowisko Cloudera Quickstart CDH VM 5.3 przedstawiono w Tabeli 2.

Tabela 2: Konfiguracja maszyny wirtualnej Cloudera Quickstart CDH VM 5.3

Właściwość	Opis
System operacyjny	CentOS 6.4 z zainstalowanym środowiskiem Cloudera CDH
Procesor	8 procesorów logicznych
Pamięć RAM	26 GB
Dysk	128 GB
System operacyjny	CentOS 6.4 z zainstalowanym środowiskiem Cloudera CDH
Procesor	8 procesorów logicznych

Na potrzeby badań wykorzystani darmowy zbiór danych „NYC Taxi Trips" udostępniany w portalu Maven Analytics [11] zawierający ponad 28 milionów rekordów z danymi przejazdów wszystkich zielonych taksówek w Nowym Jorku w latach od 2017 do 2020. Rekordy zawierają m.in. informacje dotyczące dat i godzin rozpoczęcia i zakończenia przejazdów, lokalizacji początkowej i końcowej, długości przejazdów, liczbie pasażerów i kosztach przejazdu.

Przed wykonaniem badań należało uprzednio dostosować środowisko Cloudera w maszynie wirtualnej do potrzeb eksperymentów. Uruchomienie maszyny wirtualnej z oprogramowaniem CDH nie uruchamia automatycznie interfejsu Cloudera Manager. W celu korzystania z jego funkcjonalności, należało z poziomu pulpitu użytkownika systemu wykonać skrypt uruchamiający interfejs. W celu dalszego przygotowania wirtualnej maszyny do przeprowadzenia badań należy wyłączyć narzędzia, które nie będą wymagane do zaimplementowania scenariuszy. Wyłączone zostały następujące technologie: Apache Flume, Apache HBase, Apache Hue, Apache Impala, Apache Oozie, ks_indexer, Apache Sentry, Apache Solr, Apache Spark, Apache Sqoop. Natomiast technologie, które pozostawiono uruchomionymi to: HDFS, Apache Hive, Apache Zookeeper, Apache Yarn. Zarządzanie stanem wyżej wymienionych usług odgrywa istotną rolę w wykorzystaniu zasobów, stąd zalecane jest ograniczenie liczby włączonych usług w celu zwiększenia wydajności ich działania.

5. Metoda badań

Badania zostały przeprowadzone w uprzednio przygotowanym oraz skonfigurowanym środowisku badawczym. Po czynnościach związanych z dostosowaniem fizycznej stacji roboczej i maszyny wirtualnej, do środowiska został wczytany zestaw danych. który posłużył do realizacji szesnastu scenariuszy badawczych zdefiniowanych dla obu testowanych narzędzi. Scenariusze badawcze zostały zaimplementowane w postaci zapytań HQL i skryptów Pig Latin. Przed wykonaniem skryptów założono, że istnieją już utworzone tabele z danymi w narzędziu Apache Hive.

Za pomocą metody badań zmierzone zostały czasy wykonania (w sekundach) skryptów zdefiniowanych dla treści scenariusza. Mierzono odcinek czasu od przekazania pików z implementacją scenariuszy do testowanych narzędzi, aż do zakończenia ich działania i zapisania wyników przetwarzania do plików.

Na potrzeby eksperymentu przygotowano katalog zawierający podkatalogi odpowiadające konkretnym scenariuszom badawczym. W każdym z podkatalogów umieszczono pliki z implementacjami scenariuszy narzędzi oraz skrypt uruchamiający je dziesięciokrotnie i wykonujący pomiary czasu. Wyniki każdej iteracji zapisywano do plików w podkatalogach odpowiadającym uruchomionemu narzędziu dla danego scenariusza. Po wykonaniu zaplanowanej sekwencji działań informacje o czasach wykonania poszczególnych prób zapisywano do pliku tekstowego w katalogu scenariusza. Zebrane z eksperymentów umieszczano ręcznie w pliku arkusza kalkulacyjnego, w którym wyliczano średni czas wykonania każdego ze scenariuszy badawczych. W kolejnym kroku wynik wyliczeń umieszczano w tabeli grupującej ostateczne wyniki dla scenariuszy.

6. Scenariusze badawcze

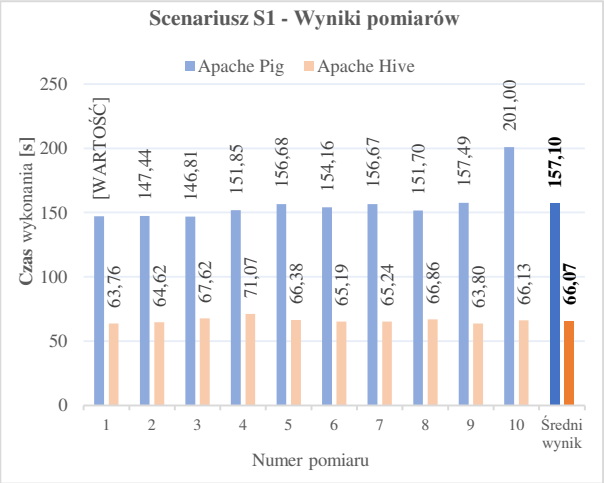
Na potrzeby przeprowadzenia badań w celu oceny efektywności przetwarzania danych z użyciem narzędzi Apache Hive i Apache Pig w środowisku Hadoop na podstawie wybranego zbioru danych przygotowano następujące scenariusze badawcze:

1. Wyznaczenie całkowitej liczby przejazdów taksówek dla poszczególnych miesięcy dla każdego roku (S1).
2. Wyznaczenie całkowitej kwoty płatności kartą kredytową dla każdego roku (S2).
3. Wyznaczenie całkowitej liczby przejazdów dla każdego rodzaju przejazdu, które były opłacone kartą kredytową (S3).
4. Wyznaczenie średniej liczby pasażerów w danym miesiącu w roku 2018 (S4).
5. Wyznaczenie średniej liczby pasażerów dla każdej godziny w ciągu doby w dniach roboczych w maju 2019 (S5).
6. Wyznaczenie pięciu dzielnic, w których kończyło się najwięcej przejazdów taksówką w weekendy w roku 2020 (S6).
7. Wyznaczenie średnich czasów podróży dla kwartałów w latach 2017-2020 (S7).
8. Wyznaczenie średniego kosztu przejazdu dla godzin dziennych (6.00- 21.59) i nocnych (22.00-5.59) dla każdego roku (S8).
9. Wyznaczenie trzech dni w roku 2017, w których średnie koszty przejazdów taksówką rozpoczętych w godzinach między godziną 16.00 a 17.00 były najmniejsze (S9).
10. Wyznaczenie całkowitej liczby przejazdów taksówką w latach 2017-2020 (S10).
11. Wyznaczenie maksymalnej i minimalnej kwoty napiwku dla każdego roku (S11).
12. Wyznaczenie średniej kwoty opłaty za przejazdu dla przejazdów z napiwkiem i bez napiwku w roku 2017 (S12).
13. Wyznaczenie średniej kwoty napiwku dla przejazdów w zależności od liczby pasażerów w godzinach dziennych 2020 roku (S13).
14. Wyznaczenie dla każdego roku średniej prędkości przejazdu taksówką na najpopularniejszej trasie przejazdu w każdym roku dla każdego roku i końcową ze wszystkich lat (S14).
15. Wyznaczenie średniej długości trasy przejazdu taksówką dla przejazdów opłaconych kartą kredytową i gotówką dla każdego roku (S15).
16. Wyznaczenie trzech dzielnic miasta, które mają największą różnicę w średnim koszcie przejazdu taksówką w ciągu dnia między godzinami szczytu (14-17), a godzinami poza szczytem (6-14 i 17-22) (S16).

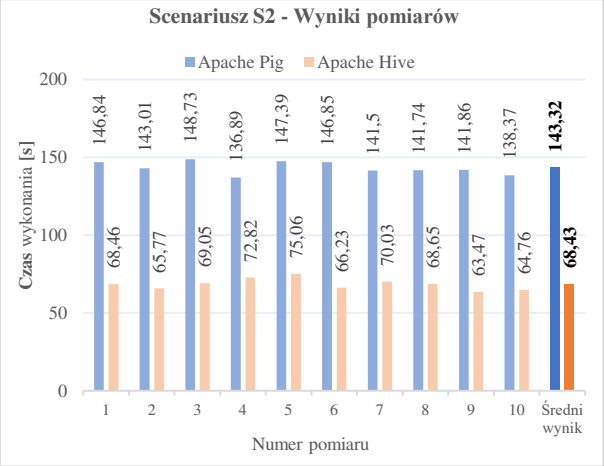
7. Wyniki badań

W niniejszym rozdziale zaprezentowano wyniki badań uzyskane na podstawie zastosowanej metody badań dla zdefiniowanych scenariuszy. Wyniki badań zostały zaprezentowane w postaci wykresów słupkowych zawierających informacje o czasach wykonywania zapytania odpowiednio dla każdego z badanych narzędzi.

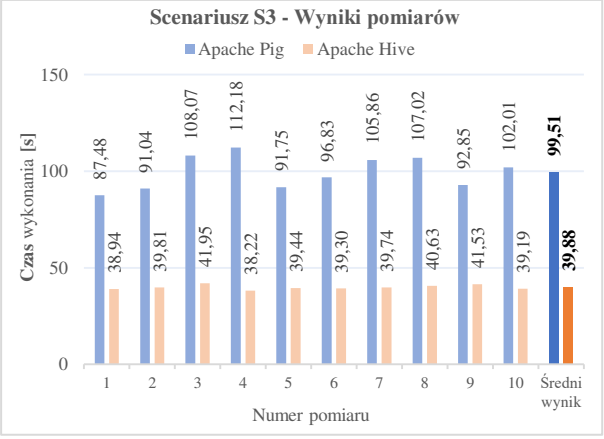
Rysunki 1-16 przedstawiają wyniki pomiarów i ich uśrednioną wartość dla scenariuszy S1-S16.



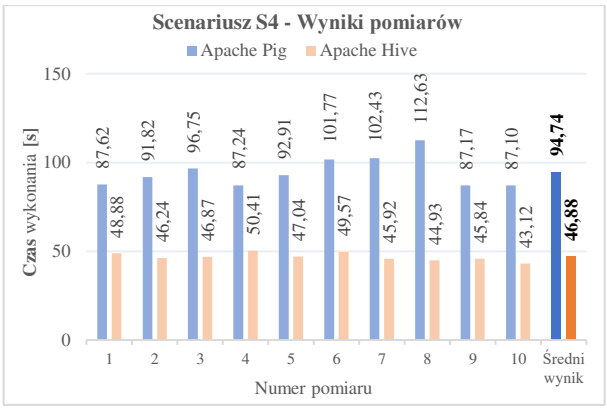
Rysunek 1: Scenariusz S1 – Wyniki pomiarów.



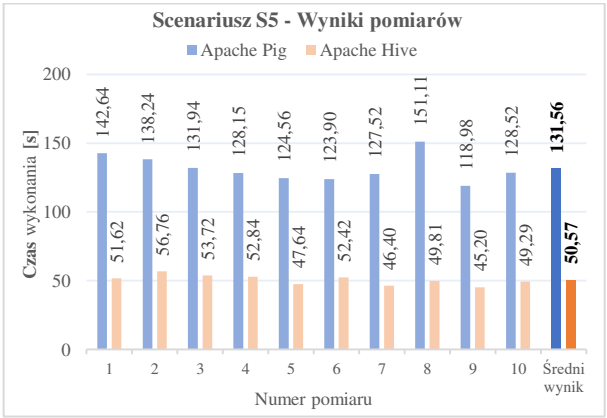
Rysunek 2: Scenariusz S2 – Wyniki pomiarów.



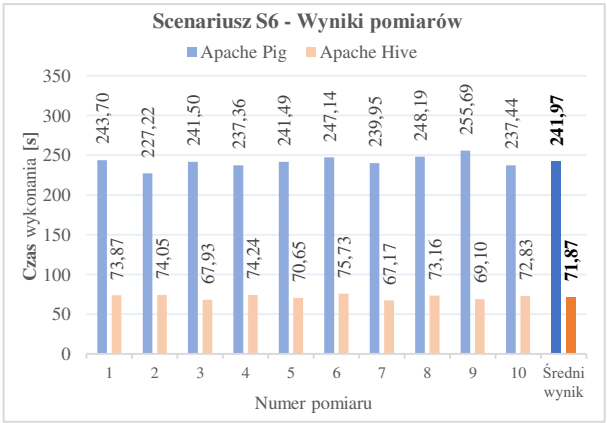
Rysunek 3: Scenariusz S3 – Wyniki pomiarów.



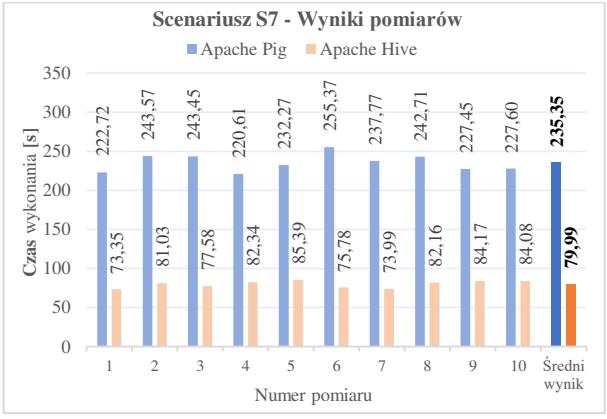
Rysunek 4: Scenariusz S4 – Wyniki pomiarów.



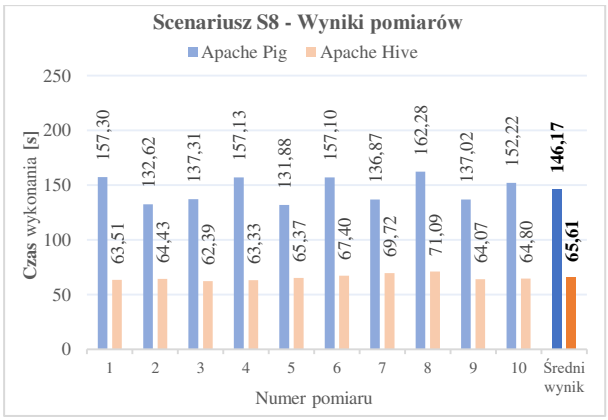
Rysunek 5: Scenariusz S5 – Wyniki pomiarów.



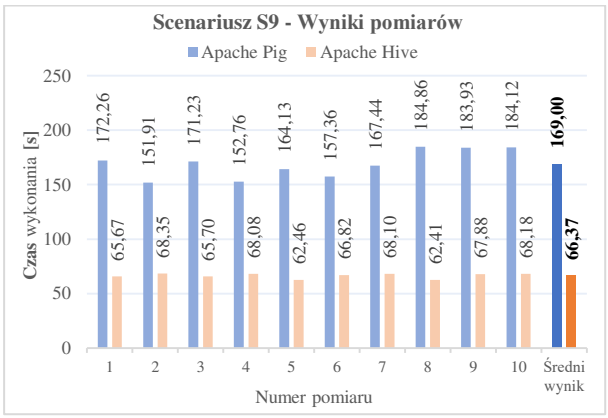
Rysunek 6: Scenariusz S6 – Wyniki pomiarów.



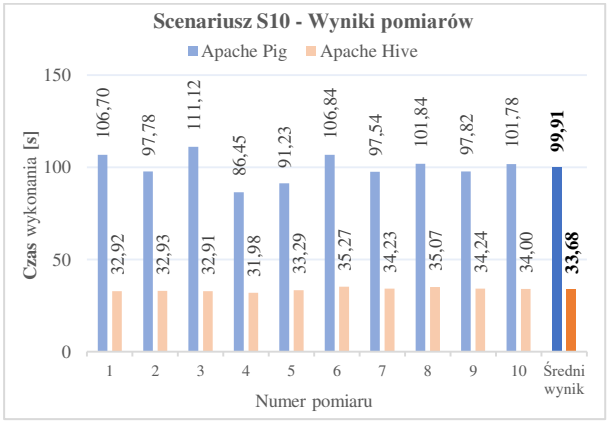
Rysunek 7: Scenariusz S7 – Wyniki pomiarów.



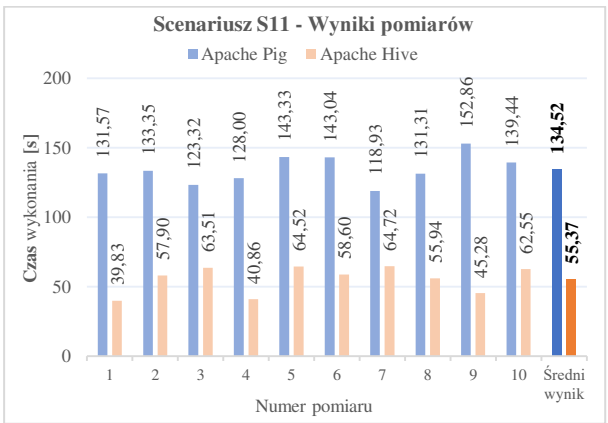
Rysunek 8: Scenariusz S8 – Wyniki pomiarów.



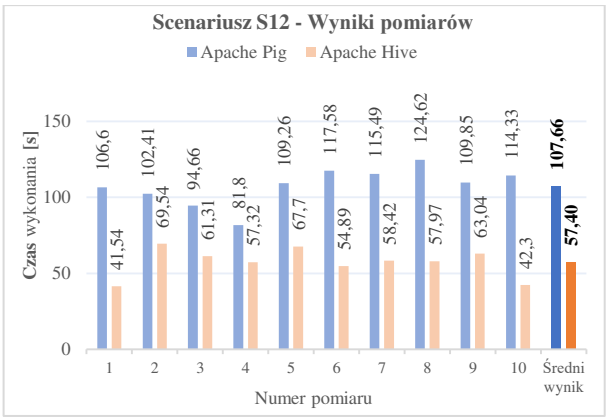
Rysunek 9: Scenariusz S9 – Wyniki pomiarów.



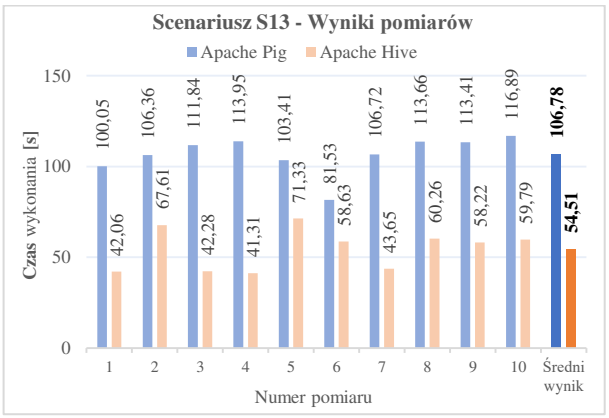
Rysunek 10: Scenariusz S10 – Wyniki pomiarów



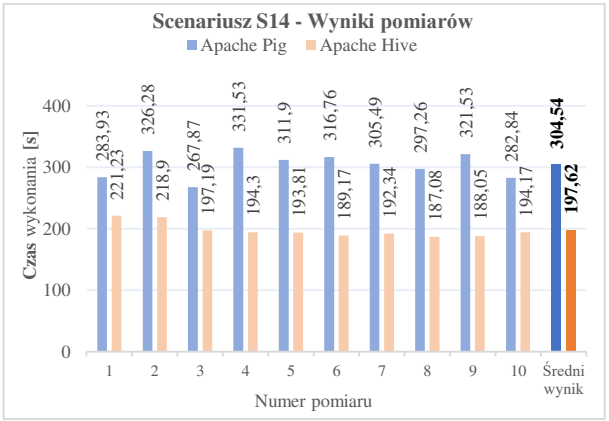
Rysunek 11: Scenariusz S11 – Wyniki pomiarów.



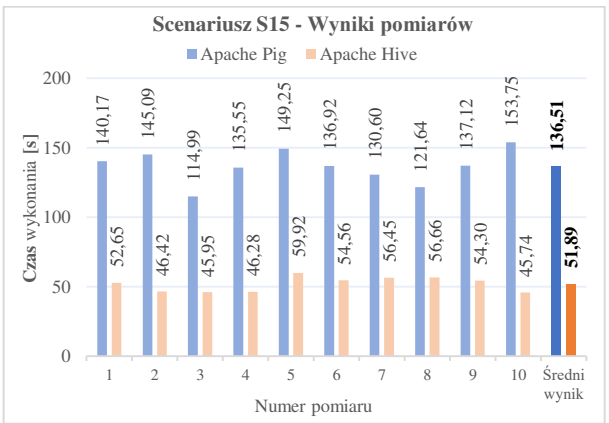
Rysunek 12: Scenariusz S12 – Wyniki pomiarów.



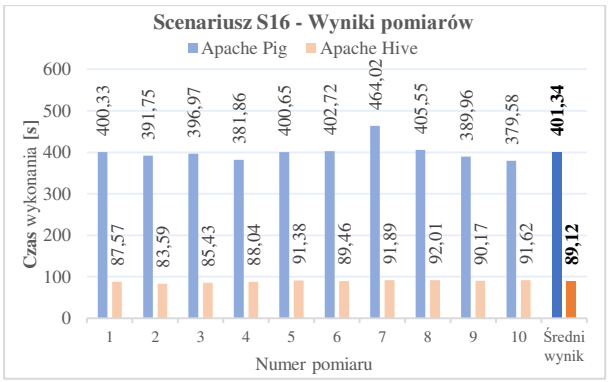
Rysunek 13: Scenariusz S13 – Wyniki pomiarów.



Rysunek 14: Scenariusz S14 – Wyniki pomiarów

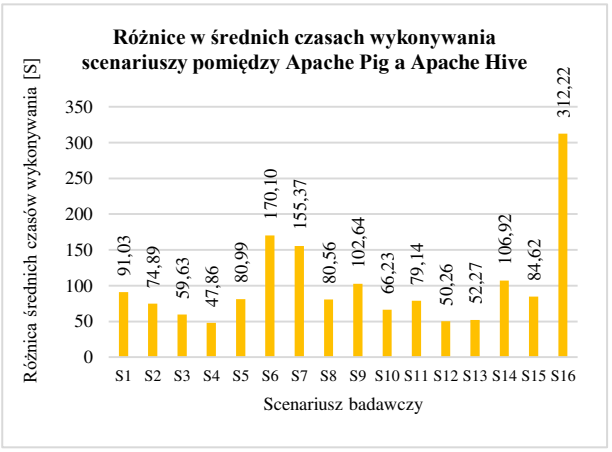


Rysunek 15: Scenariusz S15 – Wyniki pomiarów.



Rysunek 16: Scenariusz S16 – Wyniki pomiarów.

Rysunek 17 przedstawia różnice średnich czasów wykonywania scenariuszy pomiędzy narzędziami Apache Pig a Apache Hive.



Rysunek 17: Różnice w średnich czasach wykonywania scenariuszy pomiędzy Apache Pig a Apache Hive.

8. Dyskusja wyników

Analizując wyniki badań można zauważyć, że zdecydowaną przewagę w efektywnym przetwarzaniu danych w środowisku Apache Hadoop ma narzędzie Apache Hive, którego wyniki w każdym ze scenariuszy prezentują się znacznie lepiej niż dla Apache Pig. Dla większości badanych scenariuszy średni czas wykonywania scenariusza Pig był około dwu- lub trzykrotnie większy w porównaniu do czasu drugiego narzędzia.

Najmniejsza różnica w średnich czasach przetwarzania scenariuszy wyniosła 47,86 s (scenariusz S4), natomiast największa 312,22 s (scenariusz S16). Największa wartość współczynnika średniego czasu przetwarzania Apache Pig do drugiego narzędzia wyniosła 4,5 w scenariuszu S16, natomiast najmniejszy stosunek czasów wystąpił w scenariuszu S14, którego wartość była równa 1,5.

Najdłuższy średni czas wykonywania zapytania dla Hive wyniósł 197,62 s (scenariusz S14), a najkrótszy 33,68 s (scenariusz S10), w przypadku drugiego narzędzia czasy te wyniosły odpowiednio 401,34 s (scenariusz S16) oraz 94,74 s (scenariusz S4).

Pomimo tego, że narzędzie Pig nie okazało się być tak efektywnym jak Hive to istnieje prawdopodobieństwo, że możliwe jest osiągnięcie lepszych wyników z jego użyciem. Wpływ na efektywność przetwarzania może mieć konfiguracja środowiska badawczego tj. m.in.: ilość przydzielonych zasobów takich jak pamięć RAM i liczba procesorów, przepustowość sieci, prędkość odczytu dysków twardych lub liczba maszyn w klastrze biorąca udział w obliczeniach.

Z powodu braku dostępu do bardziej zaawansowanych technologicznie stacji roboczych badania zostały przeprowadzone na pojedynczej maszynie. Środowisko Cloudera w interfejsie Cloudera Manager komunikuje, że dla usług Hive, Yarn czy HDFS powinny istnieć co najmniej jeszcze dwa inne węzły w klastrze. Dodatkowym czynnikiem, który może mieć wpływ na wyniki jest m.in. wielkość zestawu danych.

Kolejnym, ale również ważnym elementem wpływającym na efektywność przetwarzania jest skomplikowanie uruchamianych zapytań i skryptów. Wyniki badań wykazują, że narzędzie Apache Pig przetwarzało dane najwolniej w przypadku bardziej skomplikowanych scenariuszy, w których pojawiały się operacje na danych takie jak: tworzenie podzapytania, więcej niż jedno złączenie tabel, filtrowanie danych czy sortowanie (scenariusze S6, S7, S16). Natomiast dobrze radziło sobie z prostymi zapytaniami, w których występowało grupowanie lub filtrowanie (scenariusze S4, S13). W przypadku Apache Hive czasy wykonania wszystkich scenariuszy oprócz najbardziej skomplikowanego (scenariusz S14) wyniosły mniej niż 90 sekund. Wyniki badań jednoznacznie wykazują, że Apache Hive jest narzędziem efektywniejszym w przetwarzaniu danych niż Apache Pig w skonfigurowanym środowisku badawczym z ekosystemem Apache Hadoop.

9. Wnioski

Celem niniejszej pracy była analiza efektywności przetwarzania danych za pomocą narzędzi Apache Hive i Apache Pig w środowisku Hadoop.

Wstępne badania narzędzi w obszarze Big Data wyróżniły środowisko Hadoop jako przodujące w rozwiązaniach przetwarzania i analizy dużych zbiorów danych. Wspomniane środowisko wyróżnia się szerokim zakresem różnorodnych narzędzi gotowych do użycia, otwarto-źródłowością oraz prężnie działającą społecznością rozwijającą oprogramowanie. Przegląd literatury ugruntował stwierdzenie iż, środowisko Hadoop oraz narzędzia Pig i Hive są flagowymi rozwiązaniami w przetwarzaniu wielkich ilości danych

Kolejnym etapem było merytoryczne przygotowanie środowiska badawczego w postaci wyboru próbki zbioru danych, zdefiniowania scenariuszy badawczych oraz predykcji optymalnej konfiguracji stacji roboczej, maszyny wirtualnej i oprogramowania Hadoop, biorąc pod uwagę sprzęt dostępny autorowi. Środowisko Cloudera CDH implementując Hadoop umożliwiło na sprawne stworzenie warunków koniecznych do przeprowadzenia badań.

Główną metodą badań było wykonanie zapytań HQL i skryptów Pig Latin dla każdego zdefiniowanego scenariusza badawczego na ówcześnie przygotowanym, tym samym zestawie danych. Celem metody badań było

zaobserwowanie wpływu wyboru narzędzia na wydajność procesowania w takich samych warunkach. Scenariusze badawcze cechowały się różnym stopniem złożoności, by przetestować możliwości obu narzędzi.

Wpływ na wiarygodność wyników odgrywało środowisko badawcze. Należy zaznaczyć, że z powodu braku dostępu do zaawansowanych stacji roboczych autor wykorzystał jedynie jedną maszynę. W przypadku wykorzystania klastra obciążenie obliczeniowe prezentowałoby się w odmienny sposób.

Wyniki badań jednogłośnie przemawiają na korzyść Apache Hive. Flagowymi scenariuszami uwydatniającymi różnice przetwarzania są scenariusze S4 i S16. Zauważono iż Hive, może być efektywniejszy od Pig nawet czterokrotnie. W pierwszym skrajnym scenariuszu stosunek długości średniego czasu wykonywania Pig do Hive uzyskał wartość 1.5, natomiast w drugim 4.5. Dodatkowo warto zaznaczyć, że średni czas wykonywania we wszystkich scenariuszach Hive, oprócz scenariusza S14, był mniejszy niż najmniejszy średni czas wykonywania ze wszystkich scenariuszy w Pig.

Podczas badań zmierzono się z wyzwaniami jakie postawiły: konfiguracja środowiska, czy też przechowywanie wyników mających wpływ na dalsze badania.

Działania podjęte w ramach pracy na polu teoretycznym jak i praktycznym umożliwiły osiągnięcie postawionego celu. Wyniki badań jednoznacznie wskazują Apache Hive jako bardziej efektywne narzędzie do przetwarzania dużej ilości danych w środowisku Hadoop.

Literatura

[1] K. Bansal, P. Chawla, P. Kurle, Analyzing Performance of Apache Pig and Apache Hive with Hadoop, International Conference On Engineering Vibration Communication and Information Processing (ICoEVCI), (2018) 41-51, https://doi.org/10.1007/978-981-13-1642-5_4

[2] M. Ahmad, S. Kanwal, M. Cheema, M. A. Habib, Performance Analysis of ECG Big Data using Apache Hive and Apache Pig, 2019 8th International Conference on Information and Communication Technologies (ICICT), (2019) 2-7, https://doi.org/10.1109/ICICT47744.2019.9001287

[3] A. Fuad, A. Erwin, H. P. Ipung, Processing performance on Apache Pig, Apache Hive and MySQL cluster, Proceedings of International Conference on Information, Communication Technology and System (ICTS), (2014) 297-302, https://doi.org/10.1109/ICTS.2014.7010600

[4] Dokumentacja techniczna technologii Apache Hadoop https://hadoop.apache.org/, [10.07.2023]

[5] K. Sitto, M. Presser, Field Guide to Hadoop: An Introduction to Hadoop, Its Ecosystem, and Aligned Technologies, O'Reilly Media, 2015

[6] Dokumentacja techniczna technologii MapReduce https://hadoop.apache.org/docs/stable/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html#Overview, [10.07.2023]

[7] D Dayong., Apache Hive Essentials Second Edition, Packt Publishing, 2015

[8] C. Swarna, Z. Ansari, Apache Pig-a data flow framework based on Hadoop Map Reduce. International Journal of Engineering Trends and Technology (IJETT), 50 (5) (2017) 271-275 https://doi.org/10.14445/22315381/IJETT-V50P244

[9] Środowisko wirtualizacji VMware Workstation 17 Player https://www.vmware.com/products/workstation-player/workstation-player-evaluation.html, [10.07.2023]

[10] Komponenty składowe środowiska Cloudera CDH https://www.cloudera.com/products/open-source/apache-hadoop/key-cdh-components.html, [10.07.2023]

[11] Zbiór danych testowych „NYC Taxi Trips Dataset" https://maven-datasets.s3.amazonaws.com/Taxi+Trips/NYC_Taxi_Trips.zip, [10.07.2023]

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 30 (2024) 9–13
Received: 7 September 2023
Accepted: 18 October 2023

Analysis of the application for the DFD authoring usage possibilities

Analiza możliwości wykorzystania aplikacji do tworzenia diagramów DFD

Marek Pieczykolan*, Marcin Badurowicz

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

Abstract
This article analyzes the possibility of using application to create data flow diagrams, by using QSEE Superlite and the application created in Flutter framework. The purpose of the article is to make comparative analysis of the time needed to create a data flow diagram, as well as to evaluate users in terms of interface transparency and freedom of action in each application. Two research hypotheses have been formulated: "DFD Maker enables faster creation of data flow diagrams compared to QSEE Superlite" and "DFD Maker has a more user-friendly interface compared to QSEE Superlite, allowing for easier creation of data flow diagrams", which were confirmed after analyzing the results of the conducted study.

Keywords: data flow diagrams; DFD; Flutter

Streszczenie
Artykuł dotyczy analizy możliwości wykorzystania aplikacji do tworzenia diagramów przepływu danych przy wykorzystaniu QSEE Superlite oraz aplikacji stworzonej we frameworku Flutter. Celem artykułu jest dokonanie analizy porównawczej czasu potrzebnego do stworzenia diagramu przepływu danych, a także oceny użytkowników pod kątem przejrzystości interfejsu oraz swobody działania w każdej z aplikacji. Postawiono dwie hipotezy badawcze: „DFD Maker pozwala na szybsze tworzenie diagramów przepływu danych w porównaniu do QSEE Superlite" oraz „DFD Maker ma przyjaźniejszy interfejs dla użytkownika w porównaniu do QSEE Superlite co pozwala na łatwiejsze tworzenie diagramów przepływu danych", które po analizie wyników, przeprowadzonych badań, zostały udowodnione.

Słowa kluczowe: diagramy przepływu danych; DFD; Flutter

*Corresponding author
Email address: marek.pieczykolan@pollub.edu.pl (M. Pieczykolan)

1. Wstęp

Ważnym aspektem przed rozpoczęciem procesu tworzenia oprogramowania jest odpowiedne zaprojektowanie systemu informatycznego. Wcześniejsza analiza wymagań, określenie reakcji systemu na różne zdarzenia pozwala na skrócenie późniejszego etapu, w którym system powstaje. Co przy obecnie dużej konkurencyjności rynku technologicznego, może okazać się kluczowe. Kosztowność rozwoju bardzo dużych projektów dodatkowo podkreśla jak ważnym aspektem jest metodyczne oraz stopniowe podejście do tego zagadnienia. Jednym z ważnych elementów przy procesie projektowaniu systemu informatycznego, szczególnie w biznesie, jest stworzenie diagramów przepływu danych, które pomogą w określeniu specyfikacji dla oprogramowania, a także ułatwią zdefiniowanie wymagań systemowych oraz rozpoznawanie i rozwiązywanie błędów [1].

Diagramy przepływu danych (ang. Data Flow Diagram - DFD) dzięki wizualnemu przestawieniu modeli logicznych, a także transformacji oraz przepływu danych, służą jako ustrukturyzowana metoda do analizy i projektowania. Poprzez modelowanie przepływu danych, diagramy, wspierają dekompozycję, która pomaga w zilustrowaniu funkcji oraz przepływów pomiędzy nimi. DFD zawierają charakterystyki takie jak, wsparcie etapu wymagań projektu oraz jego analizę, dopuszczenie procedur asynchronicznych oraz równoległych, tworzenie diagramów z adnotacjami. Ze względu na swoje przeznaczenie diagramy DFD nie mogą przedstawić kolejności wykonywania działań, dlatego nie są metodami, których można użyć do modelowania procesów jak i procedur [2].

Proces tworzenia diagramów przepływu danych wymaga przestrzegania pewnych zasad, które pozwolą zminimalizować liczbę błędów, a także pozwalają na poprawne sprawdzenie spójności pomiędzy zależnymi od siebie diagramami. Jest to ważne w kontekście czasochłonności całego procesu. Diagramy DFD można podzielić na trzy główne stopnie szczegółowości: diagram kontekstowy, diagram systemowy oraz diagram szczegółowy. Diagram kontekstowy który jest diagramem najwyższego poziomu, zawiera granice systemu informatycznego oraz pozwala na określenie obiorców oraz źródła danych. Następnym poziomem jest diagram systemowy zwany diagramem poziomu zerowego, zawiera on główne funkcje oraz magazyny danych, które są powiązane z funkcjami, zaprojektowanego systemu. Ostatnim stopniem jest diagram szczegółowy n elementowy, jego zadaniem jest przedstawienie przepływu danych w procesach. Podczas tworzenia DFD powinno się dekomponować system do

etapu, w którym będzie możliwe dostarczenie szczegółowego opisu procesu [3].

Celem niniejszego artykułu jest porównanie dwóch aplikacji do tworzenia diagramów przepływu danych. Pierwszą z nich jest QSEE Superlite, która umożliwia ona na tworzenie wielu rodzajów diagramów w tym DFD. Wszystkie badania zostaną przeprowadzone w stylu Metodologii analizy i projektowania systemów strukturalnych (ang. Structured Systems Analysis and Design Methodology - SSADM). Drugą jest aplikacja stworzona przy pomocy platformy Flutter nazwana DFD Maker. Kształt komponentów jest niemal identyczny jak w przypadku QSEE. W celu potwierdzenia hipotez, uczestnicy procesu analizy stworzą diagramy przepływu danych w dwóch aplikacjach, zostanie zmierzony czas potrzebny do stworzenia każdego z nich oraz uczestnicy wypełnią ankietę. Na tej podstawie zostały postawione niniejsze hipotezy:

H1. DFD Maker pozwala na szybsze tworzenie diagramów przepływu danych w porównaniu do QSEE Superlite.

H2. DFD Maker ma przyjaźniejszy interfejs dla użytkownika w porównaniu do QSEE Superlite co pozwala na łatwiejsze tworzenie diagramów przepływu danych.

2. Przegląd literatury

Środowisko naukowe dostarcza wiele artykułów opisujących zastosowania diagramów przepływu danych. Omawiają one nie tylko stronę teoretyczną, ale odnoszą się także do konkretnych przykładów użycia DFD przy wstępnym procesie planowania oprogramowania, prezentując wszelkie zalety jak i wady diagramów.

W pracy [4] zbadano zastosowanie diagramów przepływu danych w analizie systemowej dla procesu realizacji zamówień. Skupiono się w nim na stworzeniu modelu, który zobrazował występowanie kluczowych miejsc powodujących opóźnienia, a także ukazał najważniejsze etapy.

Autorzy w pracy [5] stworzyli model obrazujący zachodzące procesy w przedsiębiorstwie w postaci diagramów przepływu danych. Wykazano w niej jak wszechstronnym narzędziem są diagramy przepływu danych. Ich odpowiednie zastosowanie pozwala na zobrazowanie przepływu danych w taki sposób, aby określały one najsłabsze punkty w przepływie, pozwalając na wdrożenie zmian, które zniwelują owe punkty.

W pracy [6], w szczegółowy sposób opisano diagramy przepływu danych. Każdy z elementów został przedstawiony, a także dokładnie wytłumaczono zadania, które spełnia każdy z nich. Autor zaznaczył różnice pomiędzy poszczególnymi poziomami DFD, wyszczególniając podział na trzy kategorie: diagram kontekstowy; diagram systemowy; diagram szczegółowy, który zawiera niższe poziomy w modelu. Każda z kategorii została poza szczegółowym wytłumaczeniem zawiera rady, którymi należy się kierować podczas tworzenia diagramów przepływu danych.

W pracy [7] stworzono środowisko, które na podstawie diagramów przepływu danych w połączeniu z adnotacjami pozwala na przeprowadzenie symulacji. Symulacja obrazuje wydajność zaprojektowanego systemu, dając możliwość pełniejszego wykorzystania DFD. Wyniki wskazują jak przydatnym narzędziem mogą okazać się diagramy w planowaniu projektu przy odpowiednim ich wykorzystaniu.

W artykule [8] dostarczono rygorystycznych ram formalnych, które wskazują w jaki sposób mają być opisywane diagramy poprzez etykiety celu, a także podpisy prywatności. Jest to rozwinięcie możliwości jakie prezentują diagramy przepływu danych, ukazując jak wszechstronnym narzędziem mogą się one okazać i na jak wielu płaszczyznach mogą pomóc w projektowaniu oraz tworzeniu oprogramowania.

W pracy [9] ukazano sposób dołączania semantyki do diagramów przepływu danych. Przedstawiono sposób przekształcania DFD na sieci Petriego. Sposób ten przedstawia aspekty synchronizacji diagramów przepływu danych.

W pracy [10] przedstawiono zalety jakie niesie wykorzystanie diagramów przepływu danych w kontekście wygody użytkownika. Odpowiednie użycie DFD może zwiększyć komfort z korzystania z aplikacji, poprzez wcześniejsze dostarczenie informacji o przepływie danych pomiędzy poszczególnymi elementami oprogramowania. Dzięki wiedzy na temat przepływu informacji w aplikacji, można poprawić funkcjonalność, dostosowując elementy, w sposób najbardziej przyjazny użytkownikowi.

Autorzy w pracy [11], używają diagramów przepływu danych do ukazania wzorców użytkowania oraz celów bankowości internetowej. Zastosowanie DFD w artykule obrazuje jak wiele zalet niesie ze sobą użycie tego narzędzia. Najważniejszą zaletą jest ilość dostarczanych informacji w formie wizualnej, która ułatwia zrozumienie przepływu danych, a także w jaki sposób wykonywane są kolejne procesy, które tworzą całość oprogramowania bankowego. Diagramy przepływu danych dostarczyć mogą również informacji na temat potencjalnych zagrożeń oraz użyteczności w wizualizowanym modelu. Wczesne wykrycie zagrożeń może pozwolić na zmniejszenie ilości problemów, które napotkają programiści podczas implementacji funkcjonalności systemu informatycznego.

Przegląd literatury zaprezentował jak ważnym oraz przydatnym narzędziem są diagramy przepływu danych. Odpowiednie stworzenie modelu oprogramowania przy pomocy DFD może nie tylko pozwolić na lepsze zrozumienie funkcjonowania systemu informatycznego, ale również wskazać miejsca, które są szczególnie narażone na występowanie błędów. Dlatego ważnym jest użycie odpowiedniego narzędzia, które pozwoli na szybkie oraz intuicyjne tworzenie diagramów. Jest to szczególnie istotne w przypadku dużych

i skomplikowanych systemów, gdzie mogą wystąpić znaczące problemy wymagające wprowadzenia zmian w modelu.

3. Aplikacje testowe

W analizie danych oraz projektowaniu systemów informatycznych, kluczową rolę odgrywa wybór odpowiedniego narzędzia, które wspomoże proces tworzenia diagramów przepływu danych. Wraz z postępem technologii, który objawia się poprzez rosnącą wydajność sprzętu, wzrasta też poziom złożoności oprogramowania. Powoduje to potrzebę tworzenia bardziej skomplikowanych DFD. Przy dużym stopniu złożoności, ważne jest, aby narzędzie wspierało użytkownika w sposób pozwalający na jak największą optymalizację pracy, jak i ułatwienie jej. W badaniu zostały użyte dwie aplikacje, które pozwalają na tworzenie diagramów przepływu danych. Porównano ich możliwości, a także przyjazność dla użytkowników.

3.1. QSEE Superlite

Pierwszą aplikacją, której możliwości zostały sprawdzone jest QSEE Superlite w wersji 1.1.2. Aplikacja pozwala na tworzenie diagramów DFD w dwóch różnych stylach.

Pierwszym stylem jest Structured Systems Analysis and Design Method, diagramy w tym stylu obrazują przepływ danych pomiędzy procesami, zbiornikami danych, a także obiektami zewnętrznymi. Narzędzie multi-CASE wymusza stosowanie szeregu reguł, które blokują możliwość wystąpienia niespójności w modelu. Dzięki zapewnionej abstrakcji, umożliwiającej podzielenie systemu na zarządzalne segmenty, poruszanie się, a także analiza modelu jest prosta dla użytkownika.

Drugim stylem jest Real Time – Yourdon, który pozwala na modelowanie diagramów przepływu danych, ale dodatkowo wprowadza obsługę systemów czasu rzeczywistego. Styl także, jak poprzedni, bazuje na narzędziu multi-CASE, przez co również posiada reguły blokujące możliwość wystąpienia niespójności w modelu oraz wcześniej wymienione aspekty w stylu Structured Systems Analysis and Design Method.

W badaniu do analizy został wybrany styl Structured Systems Analysis and Design Method ze względu na swoją prostotę, a także większe podobieństwo diagramów do względem tych tworzonych w następnej aplikacji.

Aplikacja QSEE Superlite została wybrana ze względu na swoją popularność. Obecnie nadal jest używana do tworzenia diagramów DFD.

3.2. Aplikacja Flutterowa (DFD Maker)

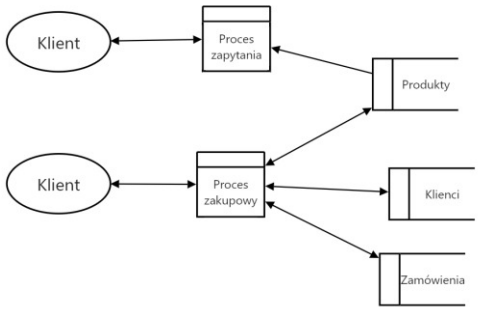
Flutter jest to framework otwartoźródłowy, stworzony oraz rozwijany przez firmę Google. Jako narzędzie wieloplatformowe pozwala na tworzenie aplikacji na wiele platform przy pomocy jednego kodu źródłowego. Obiektowy język programowania Dart, na którym jest oparty Flutter kompilowany jest do kodu natywnego dla każdej z platform, co przekłada się na zbliżoną wydajność do aplikacji stworzonych w językach natywnych [12]. Bogata biblioteka, tworzona nie tylko przez twórców platformy, pozwala na szybsze tworzenie aplikacji, a także ułatwia cały proces. Ze względu na swoje zalety ten właśnie framework posłużył jako narzędzie do stworzenia aplikacji do tworzenia diagramów DFD.

Stworzoną aplikację nazwano DFD Maker. Głównym założeniem podczas tworzenia aplikacji było zaadaptowanie udanych rozwiązań QSEE Superlite z jednoczesną poprawą interfejsu, miało to na celu przyśpieszenie oraz ułatwienie procesu tworzenia modelu przy pomocy diagramów przepływu danych. Elementy DFD wyglądem odpowiadają elementom zawartym w pierwszej aplikacji, różni się natomiast sposób umieszczania ich na płótnie. Umieszczenie zachodzi po przeciągnięciu elementu z paska bocznego, usuwa to konieczność każdorazowego klikania, co przy dużej liczbie elementów może skrócić czas potrzebny do tworzenia diagramów przepływu danych.

4. Metodyka badawcza

Celem artykułu było przeprowadzenie analizy możliwości wykorzystania aplikacji do tworzenia diagramów przepływu danych. Badanie przeprowadzono na dwóch aplikacjach QSEE Superlite oraz DFD Maker, na grupie dwudziestu uczestników. Uczestnikami badania byli absolwenci Politechniki Lubelskiej z kierunku informatyka. Znajomości podstaw diagramów DFD, a także obycie z aplikacjami służącymi do tworzenia owych diagramów, pozwoliła na dokładniejsze sprawdzenie aplikacji. Badanie podzielono na dwa scenariusze badawcze. Przed rozpoczęciem badania na podstawie opracowanych scenariuszy badawczych, każdy użytkownik otrzymywał czas, aby zapoznać się z działaniem sprawdzanej aplikacji. Podczas zapoznawania się osoby biorące udział w badaniu tworzyły proste diagramy.

Pierwszy scenariusz badawczy wymagał od użytkowników odwzorowania diagramu przepływu danych, przedstawionego na rysunku 1. Miało to na celu sprawdzenie szybkości tworzenia diagramów DFD w danej aplikacji. Zebrane na tej podstawie czasy posłużyły do obliczenia średniej oraz odchylenia standardowego dla każdej z aplikacji. Ze względu na czas potrzebny na zapoznanie się z każdą aplikacją odwzorowywany model jest prosty, co miało na celu skrócenie badań.

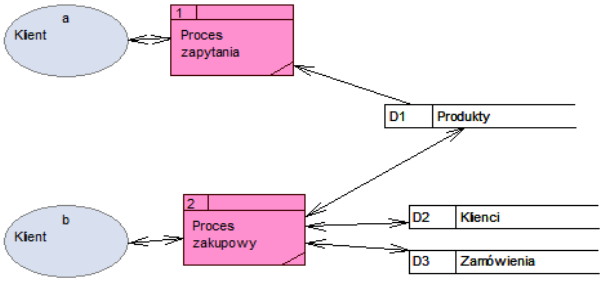


Rysunek 1: Odwzorowywany diagram przepływu danych.

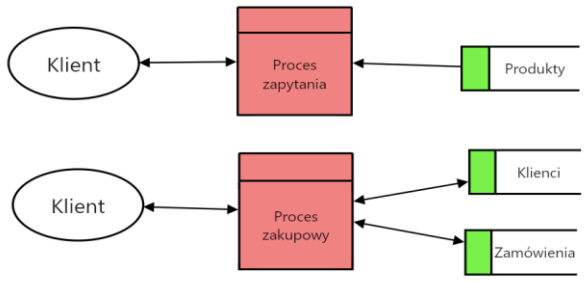
Drugi scenariusz badawczy pozwalał użytkownikom na stworzenie dowolnie wielu diagramów przepływu danych. Po zakończeniu całego procesu, użytkownicy ocenili przyjazność interfejsu, a także łatwość tworzenia DFD. Jak w poprzednim scenariuszu, tu również na podstawie zebranych wyników obliczono średnią, a także odchylenie standardowe dla każdej z aplikacji. Użytkownicy oceniali aplikacje w skali 1 do 10, gdzie 1 oznacza, iż aplikacja nie nadaje się do tworzenia diagramów, natomiast 10 oznacza uznanie aplikacji, za perfekcyjne narzędzie do tworzenia DFD.

5. Wyniki badań

Pierwszym badanym aspektem był czas potrzebny do odwzorowania diagramu przepływu danych. Wyniki dla średniego czasu oraz odchylenia standardowego zaprezentowano w Tabeli 1. Rysunek 2 oraz 3 obrazuje przykładowe odwzorowanie diagramu przepływu danych w każdej z aplikacji.



Rysunek 2: Diagram stworzony w aplikacji QSEE.

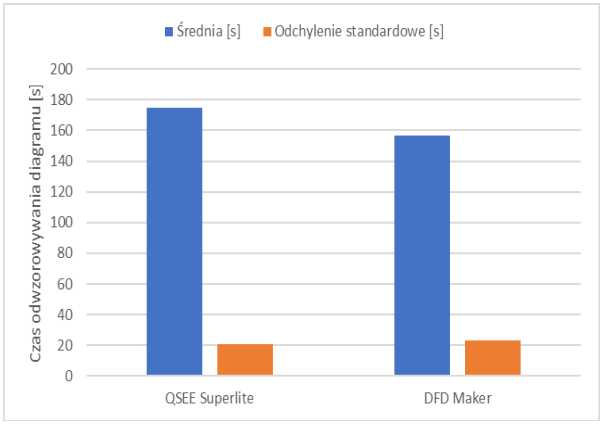


Rysunek 3: Diagram stworzony w aplikacji DFD Maker.

Tabela 1: Średni czas odwzorowywania diagramu przepływu danych przez badanych

Liczba prób: 20	Średni czas tworzenia [s] ± odchylenie standardowe
QSEE	**174,7 ± 20,82**
DFD Maker	**156,6 ± 22,897**

Zestawione wyniki dla pierwszego scenariusza badawczego przedstawiono również w formie wykresu na Rysunku 4. Przedstawiają one w sekundach średni czas potrzebny do odwzorowania diagramu przepływu danych.



Rysunek 4: Test średniego czasu potrzebnego do odwzorowania diagramu.

Drugim badanym aspektem była przejrzystość interfejsu, a także łatwość tworzenia diagramów przepływu danych. Wyniki przedstawiono w Tabeli 2 oraz 3.

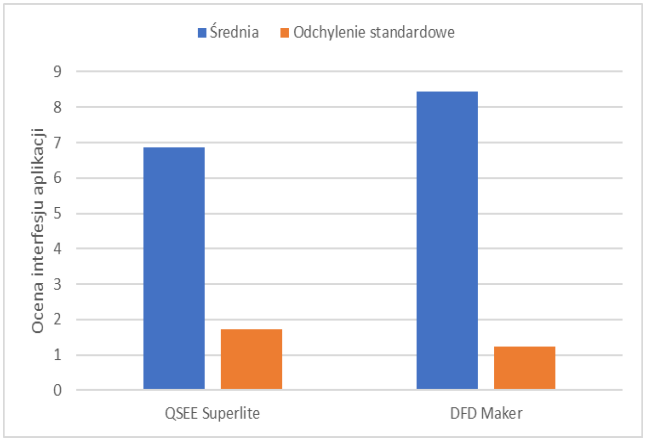
Tabela 2: Średnia ocena interfejsu aplikacji przez badanych

Liczba prób: 20	Średnia ocena aplikacji ± odchylenie standardowe
QSEE	**6,85 ± 1,725**
DFD Maker	**8,45 ± 1,234**

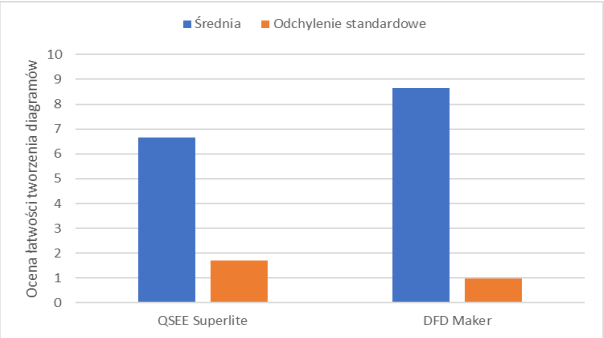
Tabela 3: Średnia ocena łatwości tworzenia diagramów

Liczba prób: 20	Średnia ocena aplikacji ± odchylenie standardowe
QSEE	**6,65 ± 1,694**
DFD Maker	**8,65 ± 0,988**

Zestawione wyniki dla drugiego scenariusza badawczego przedstawiono na Rysunkach 5 oraz 6.



Rysunek 5: Średnia ocena przejrzystości interfejsu przez użytkowników.



Rysunek 6: Średnia ocena łatwości tworzenia DFD.

6. Wnioski

W niniejszym artykule przedstawiono analizę możliwości wykorzystania aplikacji do tworzenia diagramów przepływu danych, a także omówiono zalety jakie płyną ze stosowania owych diagramów w projekcie systemu informatycznego. Wybrano dwie aplikacje do porównania, pierwszą z nich jest QSEE Superlite, drugą jest stworzona na potrzeby badania we Flutterze.

Badania, które zostały wykonane polegały na zmierzeniu czasu potrzebnego do odwzorowania diagramu przepływu danych, a także do zebrania ocen użytkowników na temat przejrzystości interfejsu oraz łatwości tworzenia DFD w każdej z aplikacji.

W przypadku badań pomiaru czasu, wyniki przedstawiają, iż DFD Maker pozwala na szybsze tworzenie diagramów, pomimo iż to QSEE Superlite było drugą aplikacją testowaną przez każdego użytkownika, co mogło pomóc QSEE, gdyż ponowne odtworzenie modelu jest szybsze. Pomimo tego dla zaproponowanego DFD różnica średnich czasów wynosi 18 sekund, co przy skomplikowanych modelach, jak i konieczności tworzenia wielu diagramów daje dużą oszczędność czasu.

Badania nad przejrzystością interfejsu, a także łatwością tworzenia diagramów przepływu danych dostarczają wyników, które obrazują, iż DFD Maker posiada przejrzystszy, a co za tym idzie przyjaźniejszy interfejs. Badani zostali poproszeni o ocenienie aplikacji w skali od 1 do 10, gdzie 1 oznaczało niemożliwość współpracy z narzędziem, a 10 stwierdzało, iż aplikacja jest idealnym narzędziem. Użytkownicy oceniali średnio interfejs DFD Makera wyżej o 1,6 punktu, przy średniej wynoszącej 8,45, natomiast QSEE Superlite osiągnęło średnią 6,85. Sama średnia ocena łatwości tworzenia diagramów również wypada korzystniej dla aplikacji opracowanej przez autora względem QSEE Superlite i wynosi 2 punkty. QSEE osiągnęło średnią 6,65 natomiast DFD Maker 8,65.

Literatura

[1] S. Adi, D. M. Kristin, Strukturisasi Entity Relationship Diagram dan Data Flow Diagram Berbasis Business Event-Driven, ComTech: Computer, Mathematics and Engineering Applications 5(1) (2014) 26-34 https://journal.binus.ac.id/index.php/comtech/article/view/2577.

[2] M. Nejad-Sattary, An Extended Data Flow Diagram Notation for Specification of Real-Time Systems 1990 https://openaccess.city.ac.uk/id/eprint/28541/.

[3] R. Ibrahim, S. Y. Yen, Formalization of the data flow diagram rules for consistency check, International Journal of Software Engineering & Applications 1(4) (2010) 95-111 https://arxiv.org/ftp/arxiv/papers/1011/1011.0278.pdf.

[4] M. Odlanicka-Poczobutt, E. Kulińska, Zastosowanie diagramu przepływu danych w analizie systemowej procesu realizacji zamówień klienta interco 1 (2015) 936-948 http://repolis.bg.polsl.pl/Content/36138/Zastosowanie%20diagramu_97076.pdf.

[5] A. Liaushuk, A. Jasiak, Method of organizing of the data flow in business entity on the basis of data flow diagram and the theory of queues, Research in Logistics & Production 9(1) (2019) 31-38 https://yadda.icm.edu.pl/baztech/element/bwmeta1.element.baztech-93d60794-6b31-45aa-9af2-e8f2957aecb8.

[6] Uniwersytet Kapsztadzki w Południowej Afryce [07.06.2023] https://www.cs.uct.ac.za/mit_notes/software/htmls/ch06.html.

[7] J. R. Warren, J. W. Stott, A. F. Norcio, Stochastic simulation of information systems designs from data flow diagrams, Journal of Systems and Software 18(2) (1992) 191-199 https://www.sciencedirect.com/science/article/abs/pii/0164121292901276.

[8] H. Alshareef, K. Tuma, S. Stucki, G. Schneider, R. Scandariato, Precise Analysis of Purpose Limitation in Data Flow Diagrams, Proceedings of the 17th International Conference on Availability, Reliability and Security (2022) 1-11 https://dl.acm.org/doi/abs/10.1145/3538969.3539010.

[9] P. D. Bruza, T. Van der Weide, The Semantics of Data Flow Diagrams, Nijmegen TR: University of Nijmegen, Department of Informatics, Faculty of Mathematics and Informatics 1989 https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=70cfc87f97d9aacdcac857f7222086081d965cd3.

[10] W. Wulandari A. D. Y. Widiantoro, Design Data Flow Diagram for Supporting the User Experience in Applications, Design Data Flow Diagram for Supporting the User Experience in Applications 25(2) (2017) 14-20 http://repository.unika.ac.id/17139/.

[11] R. Ganesh, G. Prabu, Determination of Internet Banking Usage and Purpose with Explanation of Data Flow Diagram and Use Case Diagram, International Journal of Management and Humanities 4(7) (2020) 52-58 https://www.ijmh.org/wp-content/uploads/papers/v4i7/G0674034720.pdf.

[12] S. Zindl, Flutter on Windows Desktop: a use case based study, Bachelor's thesis [07.06.2023] https://elib.uni-stuttgart.de/handle/11682/11740

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 30 (2024) 14–20
Received: 10 September 2023
Accepted: 1 October 2023

Comparative analysis of query execution speed using Entity Framework for selected database engines

Analiza porównawcza szybkości wykonywania zapytań za pomocą Entity Framework dla wybranych silników baz danych

Krzysztof Winiarczyk*, Rafał Stęgierski

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

Abstract
This article presents the comparative analysis of time efficiency while executing queries by object-relational mapping framework Entity Framework for the following database engines: Microsoft SQL Server, MySQL and PostgreSQL. Time measurements of obtaining object results from database queries were made by app created in C#. Queries referred to single or multiple tables linked by relationships (1:1, 1:n, m:n) and performed operations of reading, creating, updating and deleting data. Obtained results have been cleaned from outliers and trimmed means were given as final results. Different database engines obtained the shortest query execution times depending on record number and table structures.

Keywords: time efficiency; Entity Framework; database

Streszczenie
Artykuł przedstawia analizę porównawczą wydajności czasowej wykonywania zapytań za pomocą szkieletu mapowania obiektowo-relacyjnego Entity Framework dla następujących silników baz danych: Microsoft SQL Server, MySQL i PostgreSQL. Pomiaru czasu uzyskania obiektowych rezultatów zapytań do bazy danych dokonano przy pomocy aplikacji napisanej w języku C#. Zapytania dotyczyły jednej tabeli bądź kilku tabel połączonych relacjami (1:1, 1:n, m:n) oraz realizowały operacje odczytu, tworzenia, aktualizacji i usuwania danych. Uzyskane rezultaty oczyszczono z wartości odstających, a jako wyniki podano średnie ucinane. W zależności od liczby rekordów oraz struktury tabel różne silniki baz danych uzyskiwały najkrótsze czasy wykonania zapytań.

Słowa kluczowe: wydajność czasowa; Entity Framework; baza danych

*Corresponding author
Email address: **krzysztof.r.winiarczyk@gmail.com** (K. Winiarczyk)

1. Wstęp

Jednym z rozwiązań wykorzystywanym we współczesnych aplikacjach serwerowych do trwałego przechowywania informacji są relacyjne bazy danych. Do ich niewątpliwych zalet należą m. in.: zapewnianie bezpieczeństwa i integralności informacji oraz możliwość agregowania danych w celu uzyskania złożonych raportów.

W relacyjnych bazach danych przechowywanie informacji zorientowane jest na relacje jakie zachodzą pomiędzy konkretnymi encjami, natomiast duża część języków programowania (używanych współcześnie do tworzenia aplikacji internetowych) zorientowana jest obiektowo. Rozwiązaniem ułatwiającym przenoszenie danych pomiędzy bazą danych a aplikacją są szkielety programistyczne mapowania obiektowo-relacyjnego. Jednym z nich jest Entity Framework, który oferuje możliwość pisania zapytań do bazy danych bezpośrednio w kodzie programu, bez użycia języka SQL. Taka funkcjonalność usprawnia tworzenie aplikacji, stanowi ułatwienie dla programisty (który operuje jednym językiem) oraz poprawia przejrzystość kodu – logika biznesowa nie jest rozproszona pomiędzy kod aplikacji i skrypty SQL [1].

Szkielet programistyczny mapowania obiektowo-relacyjnego stanowi dodatkową warstwę, która pośredniczy między aplikacją i bazą danych. Zastosowanie tego rodzaju narzędzi ma negatywny wpływ na wydajność [2], która jest kluczowym elementem wytwarzanego oprogramowania. Istotną kwestią w przypadku projektowania nowych systemów informatycznych jest dobór odpowiednich dla danego zastosowania technologii, które po połączeniu pozwolą na stworzenie wydajnych rozwiązań.

Niniejsza praca została poświęcona porównaniu wydajności czasowej wykonywania zapytań przy użyciu narzędzia Entity Framework współpracującego z wybranymi silnikami baz danych.

2. Przegląd literatury

Optymalizacja aplikacji jest ważnym elementem procesu wytwórczego oprogramowania. Jednym ze sposobów na skrócenie czasu działania poszczególnych funkcjonalności aplikacji jest zmniejszenie czasu dostępu do danych poprzez użycie optymalnych metod łączenia się z bazą danych oraz wykorzystanie silników baz danych najbardziej odpowiednich dla rozpatrywanej aplikacji. Porównanie wydajności w kontekście baz danych jest popularnym tematem prac naukowych.

Celem artykułu [2] jest porównanie wydajności wykonywania zapytań dla trzech frameworków ORM dla .NET: Entity Framework Core 2.2, nHibernate 5.2.3 i Dapper 1.50.5. Dla każdego frameworka ORM autorzy przeprowadzili testy obejmujące zapytania typu insert, select, update i delete na tej samej bazie danych. Program utworzony do testowania jest aplikacją konsolową. Jako silnika bazodanowego autorzy użyli MSSQL Server 2018. Zebrane wyniki zostały podsumowane stwierdzeniem, że nie da się jednoznacznie wskazać najbardziej wydajnego frameworka, ani pod względem czasu wykonywania zapytań, ani pod względem użytej pamięci operacyjnej. W związku z czym wybór szkieletu ORM dla aplikacji (pod kątem wydajności) zależy od tego jakie zapytanie (typ operacji) wykonywany jest najczęściej.

Artykuł [3] zawiera porównanie wydajnościowe dwóch popularnych otwarto-źródłowych silników baz danych: MySQL (baza danych SQL) i CouchDB (baza danych NoSQL). Głównym celem publikacji jest przeprowadzenie analizy porównawczej wpływu systemu zarządzania bazą danych na wydajność aplikacji podczas wykonywania operacji typu CRUD. Autorzy rozpatrzyli cztery rozwiązania bazodanowe: MySQL w podejściu relacyjnym, MySQL w podejściu opartym na dokumentach oraz dwie struktury w CouchDB. Dokument w pierwszej strukturze, w CouchDB, jest prostym obiektem JSON zawierającym jedynie pary klucz-wartość (bez zagnieżdżonych obiektów), natomiast posiadającym odniesienie do innego dokumentu poprzez id. Dokument w drugiej strukturze jest złożonym obiektem JSON – posiadającym obiekty wielokrotnie zagnieżdżone, ale nieposiadającym odniesień do innych dokumentów. Struktura relacyjna bazy MySQL składa się z 5 tabel i 5 relacji jeden do wielu. Relacje można przedstawić następująco: kontynent (1:n) kraj, kraj (1:n) miasto, miasto (1:n) restauracja, miasto (1:n) hotel, hotel (1:n) restauracja. Struktura dla dokumentowego MySQL jest identyczna jak druga struktura bazy CouchDB. Każdy rodzaj operacji na danych został wykonany dla kilku liczności rekordów (1000, 10 000, 100 000, 1 000 000). Wynik dla rozwiązania, operacji i liczby rekordów stanowi średnia arytmetyczna z pięciu pomiarów czasu wykonania. W omówieniu wyników autorzy wskazali, że MySQL w podejściu relacyjnym wykazuje duże spadki wydajności przy zwiększaniu ilości danych (liczby rekordów). Lepszym rozwiązaniem okazuje się zastosowanie bazy CouchDB, natomiast najlepszą wydajnością czasową wyróżnia się MySQL w podejściu opartym na dokumentach. W podsumowaniu autorzy wskazują, że MySQL oparty na dokumentach jest bardzo dobrą alternatywą dla aplikacji przetwarzających duże ilości danych. Użycie drugiej struktury CouchDB pozwala osiągnąć lepszą wydajność czasową niż zastosowanie relacyjnego MySQL. Autorzy zwracają również uwagę na istotną kwestię, którą należy rozważyć przy projektowaniu aplikacji: czy zysk na wydajności czasowej jest ważniejszy niż normalizacja bazy danych i reguły ACID, które zostają odrzucone przy wyborze bazy zorientowanej na dokumenty.

Materiał konferencyjny [4] zawiera porównanie wydajności wykonywania pojedynczych poleceń SQL, poleceń multi-SQL oraz przygotowanych instrukcji SQL. Do celów publikacji zalicza się wskazanie optymalnych warunków do użycia przygotowanych zapytań. Po przeprowadzeniu analizy otrzymanych wyników autorzy wskazali, że pod względem wydajności, przygotowane instrukcje SQL nie powinny być używane do prostych zapytań. Zalecane jest natomiast wykorzystanie zapytań standardowych oraz zapytań multi-SQL. Autorzy wskazali również, na konieczność porównania korzyści dla bezpieczeństwa aplikacji z kosztami wydajnościowymi związanymi z użyciem preapred statements.

Materiał konferencyjny [5] zawiera porównanie wydajności Entity Framework i NHibernate – dwóch najpopularniejszych szkieletów mapowania obiektowo-relacyjnego dla platformy .NET Framework. Autorzy wykonali testy dla dwóch silników bazodanowych (MS SQL Server i PostgreSQL) oraz różnych języków zapytań (wyrażenia lambda i LINQ dla Enity Framework oraz HQL i Criteria API dla NHibernate). Uzyskane wyniki zostały porównane z zapytaniami wykonani za pomocą standardowego sposobu wykorzystującego SqlClient. W podsumowaniu autorzy wskazują, że w ogólności lepszą wydajnością cechowały się zapytania wykonywane przy użyciu SqlClient, jakkolwiek różnice wydajnościowe pomiędzy dobrze zaprojektowanym ORM a standardowym sposobem wykonywania zapytań nie były znaczące. Uzyskane rezultaty badań zostały podsumowane stwierdzeniem, że nie da się na ich podstawie jednoznacznie wskazać rekomendowanego pod względem wydajności szkieletu ORM, a wybór powinien być dokonywany uwzględniając wiele innych kryteriów (takich jak wspierane silniki baz danych, wsparcie grupowania i funkcji agregujących).

3. Cel badań

Celem badań jest porównanie wydajności czasowej wykonywania zapytań za pomocą Entity Framework dla wybranych silników baz danych.

4. Uwzględniane technologie

Przy analizie porównawczej uwzględniono następujące silniki baz danych: Microsoft SQL Server, MySQL oraz PostgreSQL. Program testowy został napisany przy użyciu języka C#, na platformie .NET.

4.1. Język C#, platforma .NET

C# jest wysoko-poziomowym językiem programowania ogólnego przeznaczenia. Jest silnie typowany i zorientowany obiektowo [6].

Platforma .NET to bezpłatne i otwartoźródłowe oprogramowanie pozwalające tworzyć i uruchamiać aplikacje napisane m. in. w języku C# [7].

Kod w C# jest kompilowany do kodu pośredniego (Intermediate Language – IL). Podczas wykonywania programu napisanego w języku C# środowisko uruchomieniowe (Common Language Runtime – CLR) prze-

prowadza kompilację Just-In-Time (JIT), aby przekonwertować kod IL do języka maszynowego [6].

4.2. Szkielet programistyczny Entity Framework Core

Entity Framework Core jest otwartoźródłowym szkieletem programistycznym mapowania obiektowo-relacyjnego dla platformy .NET. Narzędzie pozwala programiście pracować z bazą danych za pomocą obiektów i eliminuje konieczność pisania większości kodu odpowiadającego za dostęp do danych [8].

4.3. System zarządzania bazą danych Microsoft SQL Server Express

Microsoft SQL Server jest systemem zarządzania relacyjnymi bazami danych rozwijanym i dostarczanym przez firmę Microsoft. Wersją wolną do pobierania i rozpowszechniania jest SQL Server Express [9].

4.4. System zarządzania bazą danych MySQL

MySQL jest otwartoźródłowym systemem zarządzania bazami danych rozwijanym, dostarczanym i wspieranym przez Oracle Corporation [10]. Oprogramowanie jest dostępne na zasadach licencji GNU General Public License oraz płatnej licencji komercyjnej [10].

4.5. System zarządzania bazą danych PostreSQL

PostgreSQL jest darmowym i otwartoźródłowym systemem zarządzania bazami danych [11]. Początki systemu sięgają 1986 roku i projektu POSTGRES na Uniwersytecie Kolifornijskim w Berkeley. Oprogramowanie jest aktywnie rozwijane od ponad 35 lat [11].

5. Badania

W celu zapewnienia wiarygodnych wyników, po każdym przeprowadzeniu testu baza danych była usuwana, tworzona i wypełniana rekordami na nowo.

5.1. Środowisko testowe

Parametry komputera osobistego, na którym przeprowadzono eksperyment przedstawiono w Tabeli 1.

Tabela 1: Parametry komputera użytego do przeprowadzonego eksperymentu

Procesor	AMD Ryzen 7 4700U
Pamięć RAM	32GB 2666MHz
System operacyjny	Windows 11 Home 64-bitowy

W Tabeli 2 przedstawiono wersje i edycje systemów zarządzania relacyjnymi bazami danych użytych do przeprowadzenia eksperymentu.

Tabela 2: Wersje i edycje systemów zarządzania bazami danych użytych do przeprowadzonego eksperymentu

System zarządzania relacyjną bazą danych	Edycja	Wersja
Microsoft SQL Server 2022	Express Edition	16.0.1050.5
MySQL	Community Server - GPL	8.0.32
PostgreSQL	-	15.2

Wersje użytego oprogramowania, narzędzi i wybranych pakietów zostały przedstawione w Tabeli 3.

Tabela 3: Pozostałe wersje wykorzystanego oprogramowania

Program/narzędzie/pakiet	Wersja
C#	10.0
.NET	6.0
Microsoft.EntityFrameworkCore.Tools	7.0.5
Microsoft.EntityFrameworkCore.SqlServer	7.0.5
MySql.EntityFrameworkCore	7.0.2
Npgsql.EntityFrameworkCore.PostgreSQL	7.0.4

5.2. Scenariusze

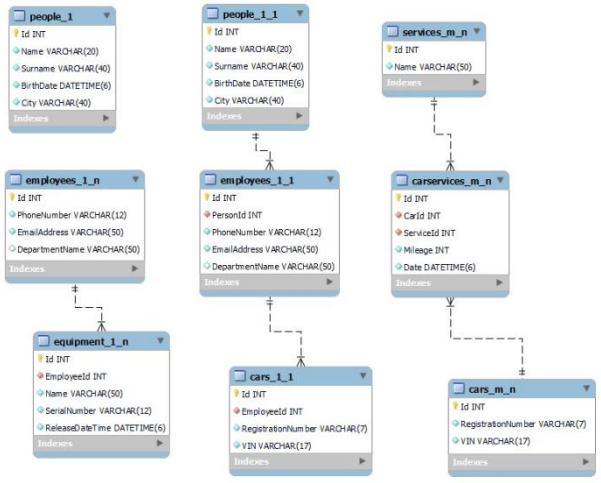
Scenariusze testowe obejmowały wykonanie zapytań realizujących operację odczytu, zapisu, aktualizacji i usuwania dla 1 000, 5 000, 10 000 i 100 000 rekordów w następujących przypadkach:

1. Zapytanie dotyczące jednej tabeli.
2. Zapytanie dotyczące trzech tabel (połączonych relacjami 1:1).
3. Zapytanie dotyczące dwóch tabel (połączonych relacjami 1:n).
4. Zapytanie dotyczące trzech tabel (połączonych relacjami 1:n w taki sposób, że rozbijają one relację m:n).

Każda operacja została wykonana 10 razy dla danego przypadku i liczby rekordów.

5.3. Struktura bazy danych

Do testów wykorzystano bazę danych przechowującą wygenerowane informacje na temat hipotetycznego przedsiębiorstwa. Diagram związków encji (ERD) przedstawiony jest na Rysunku 1.



Rysunek 1: Schemat ERD bazy danych.

Tabele Cars_1_1 i Cars_m_n są przykładem zduplikowania tabel dotyczących tych samych informacji. Specyficzna struktura bazy danych ma swoje uzasadnienie w obecności kluczy obcych. Gdyby tabela przechowująca informacje o samochodach (Cars) była połączona jednocześnie z tabelą przechowującą informacje o przeprowadzonych serwisach (CarServices) oraz tabelą przechowującą informacje o pracownikach (Em-

ployees) usunięcie rekordu z tabeli Cars spowodowałoby kaskadowe usunięcie rekordów z tabeli CarsServices i usunięcie rekordów albo zastąpienie wartościami null pól tych rekordów w tabeli Employees. Taki stan rzeczy powodowałby, że wyniki nie byłyby miarodajne.

Początkową liczbę rekordów dla poszczególnych tabel przedstawiono w Tabeli 4.

Tabela 4: Początkowe liczby rekordów w poszczególnych tabelach bazy danych

Tabela	Początkowa liczba rekordów
People_1	5 000 000
Cars_1_1	5 000 000
Employees_1_1	5 000 000
People_1_1	5 000 000
Employees_1_n	5 000 000
Equipment_1_n	15 000 000
Cars_m_n	5 000 000
CarServices_m_n	19 998 419
Services_m_n	17

6. Wyniki

Na przypadek testowy składają się 4 parametry: silnik bazy danych, struktura w bazie danych, liczba rekordów oraz rodzaj operacji (odczyt, zapis, aktualizacja, usuwanie). Dla każdego przypadku testowego uzyskano 10 wyników, które następnie zostały odfiltrowane za pomocą metody opierającej się na rozstępie międzykwartylowym. Oczyszczenie danych polegało na odrzuceniu wartości odstających, to znaczy takich, które nie spełniają warunku przedstawionego we wzorze (1)

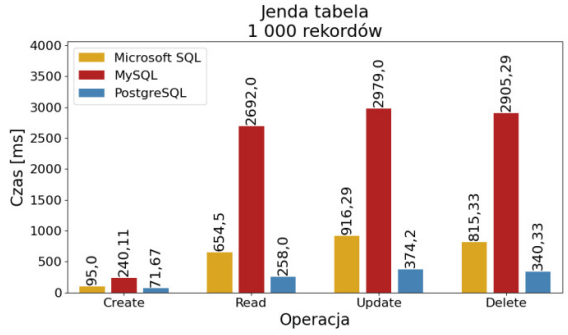
$$Q_1 - 1{,}5 * IQR \leq x \leq Q_3 + 1{,}5 * IQR \qquad (1)$$

gdzie x jest pojedynczym wynikiem, Q_1 jest pierwszym kwartylem obliczonym na podstawie wyników dla danego przypadku testowego, Q_3 jest trzecim kwartylem, a $IQR = Q_3 - Q_1$ jest rozstępem międzykwartylowym.

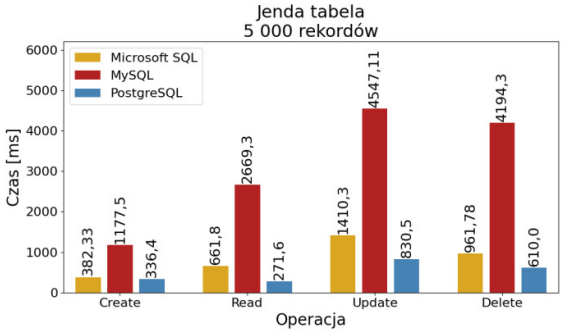
W celu obrazowego przedstawienia rozbieżności w czasach wykonania różnych operacji dla 3 silników baz danych utworzono wykresy słupkowe. Zaprezentowane na nich wyniki są średnimi ucinanymi z wyników jednostkowych dla danego przypadku testowego.

6.1. Operacje dla jednej tabeli

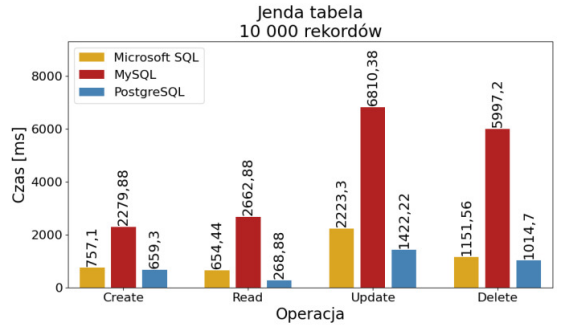
Średnie czasy wykonania operacji CRUD dotyczących jednej tabeli, dla rozpatrywanych silników baz danych, przedstawiono na Rysunkach 2-5.



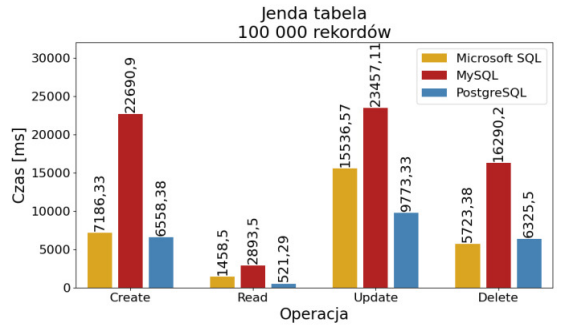
Rysunek 2: Średnie czasy wykonania operacji CRUD dla 1 000 rekordów z jednej tabeli.



Rysunek 3: Średnie czasy wykonania operacji CRUD dla 5 000 rekordów z jednej tabeli.



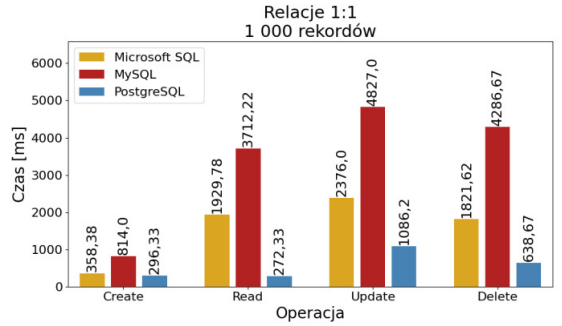
Rysunek 4: Średnie czasy wykonania operacji CRUD dla 10 000 rekordów z jednej tabeli.



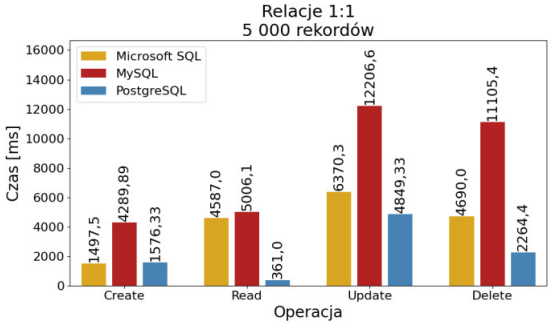
Rysunek 5: Średnie czasy wykonania operacji CRUD dla 100 000 rekordów z jednej tabeli.

6.2. Operacje dla trzech tabel połączonych relacjami 1:1

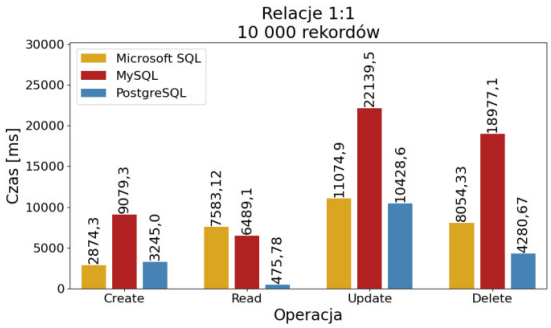
Średnie czasy wykonania operacji CRUD dotyczących trzech tabel połączonych relacjami 1:1, dla rozpatrywanych silników baz danych, przedstawiono na Rysunkach 6-9.



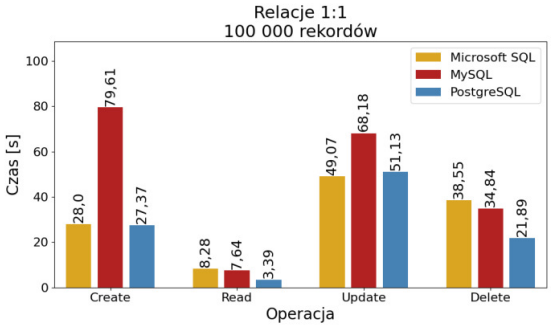
Rysunek 6: Średnie czasy wykonania operacji CRUD dla 1 000 rekordów z trzech tabel połączonych relacjami 1:1.



Rysunek 7: Średnie czasy wykonania operacji CRUD dla 5 000 rekordów z trzech tabel połączonych relacjami 1:1.



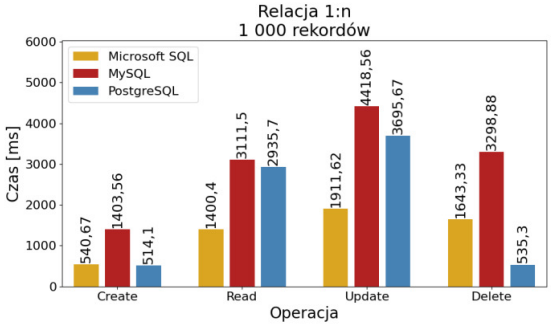
Rysunek 8: Średnie czasy wykonania operacji CRUD dla 10 000 rekordów z trzech tabel połączonych relacjami 1:1.



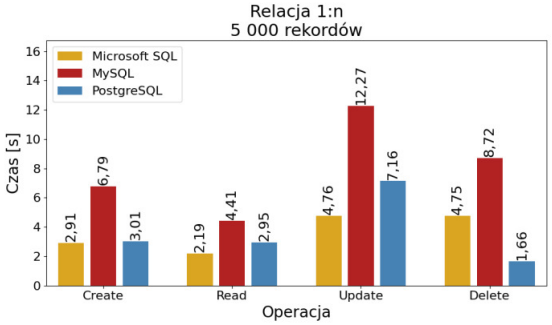
Rysunek 9: Średnie czasy wykonania operacji CRUD dla 100 000 rekordów z trzech tabel połączonych relacjami 1:1.

6.3. Operacje dla dwóch tabel połączonych relacjami 1:n

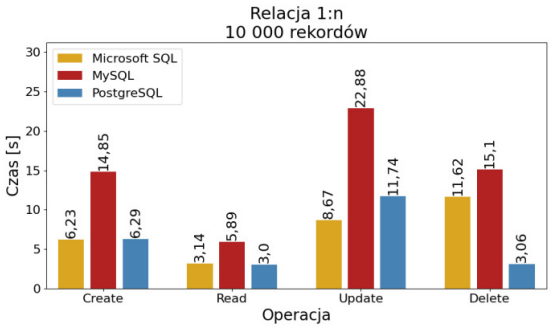
Średnie czasy wykonania operacji CRUD dotyczących dwóch tabel połączonych relacją 1:n, dla rozpatrywanych silników baz danych, przedstawiono na Rysunkach 10-13.



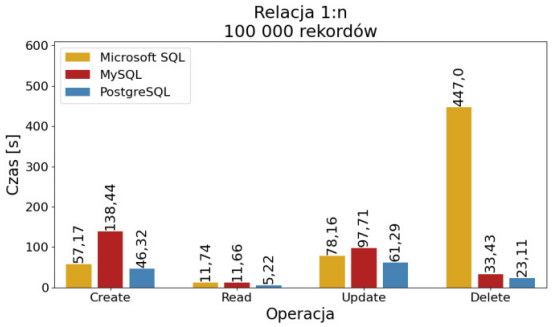
Rysunek 10: Średnie czasy wykonania operacji CRUD dla 1 000 rekordów z dwóch tabel połączonych relacją 1:n.



Rysunek 11: Średnie czasy wykonania operacji CRUD dla 5 000 rekordów z dwóch tabel połączonych relacją 1:n.



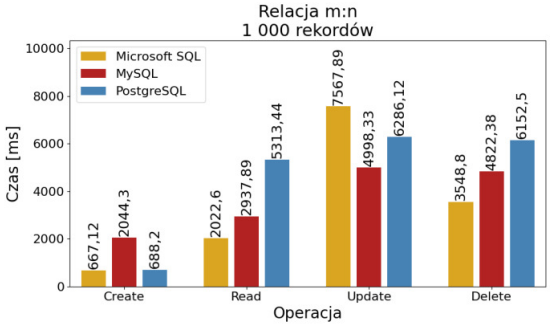
Rysunek 12: Średnie czasy wykonania operacji CRUD dla 10 000 rekordów z dwóch tabel połączonych relacją 1:n.



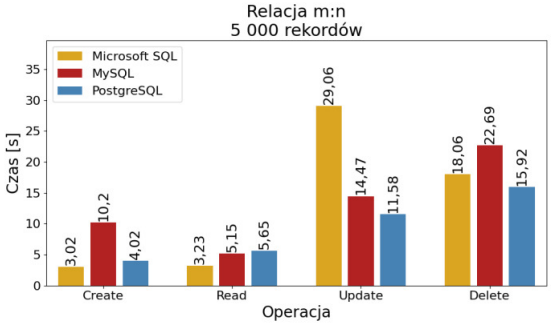
Rysunek 13: Średnie czasy wykonania operacji CRUD dla 100 000 rekordów z dwóch tabel połączonych relacją 1:n.

6.4. Operacje dla trzech tabel rozbijających relację m:n

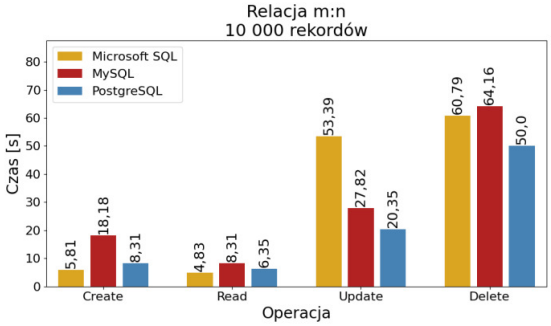
Średnie czasy wykonania operacji CRUD dotyczących trzech tabel połączonych relacjami 1:n (w sposób który rozbija relację m:n), dla rozpatrywanych silników baz danych, przedstawiono na Rysunkach 14-17.



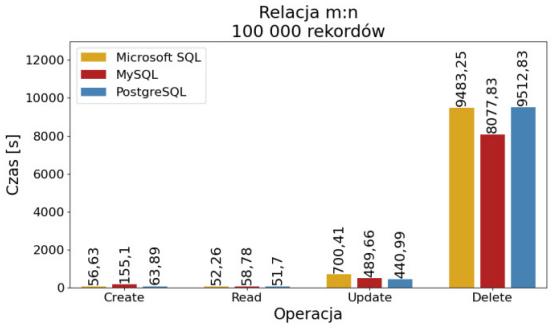
Rysunek 14: Średnie czasy wykonania operacji CRUD dla 1 000 rekordów z trzech tabel rozbijających relację m:n.



Rysunek 15: Średnie czasy wykonania operacji CRUD dla 5 000 rekordów z trzech tabel rozbijających relację m:n.



Rysunek 16: Średnie czasy wykonania operacji CRUD dla 10 000 rekordów z trzech tabel rozbijających relację m:n.



Rysunek 17: Średnie czasy wykonania operacji CRUD dla 100 000 rekordów z trzech tabel rozbijających relację m:n.

7. Wnioski

Dla operacji wykonywanych na rekordach jednej tabeli można wskazać, że najgorsze wyniki wydajności czasowej, niezależnie od liczby rekordów i rodzaju operacji, uzyskał silnik MySQL. Silnikiem z najlepszymi wynikami (poza przypadkiem testu operacji usuwania dla 100 000 rekordów) był PostgreSQL.

W przypadku trzech tabel połączonych relacjami 1:1 przewaga PostgreSQL jest nadal widoczna, przy czym wystąpiły 3 przypadki, w których ustąpił on technologii Microsoft SQL Server (operacje tworzenia dla 5 000 i 10 000 rekordów oraz operacja aktualizacji dla 100 000 rekordów). Silnik MySQL w większości przypadków osiągnął najgorsze wyniki – wyjątkami są tutaj operacje odczytu dla 10 000 rekordów oraz operacje odczytu i usuwania dla 100 000 rekordów, gdzie najwolniejszy był Microsoft SQL Server.

Rozpatrując wyniki czasowe dla dwóch tabel połączonych relacją 1:n można stwierdzić, że silnik MySQL uzyskał najgorsze wyniki poza przypadkami testowania operacji odczytu i usuwania dla 100 000 rekordów (najwolniejszy był wtedy Microsoft SQL Server). Trudno o wskazanie innych tendencji, ponieważ dla danego rodzaju operacji w zależności od liczby rekordów najszybszy okazywał się PostgreSQL albo Microsoft SQL Server.

Przypadek trzech tabel rozbijających relację m:n jest najbardziej zróżnicowany pod względem wyników osiąganych przez poszczególne silniki. Dla operacji tworzenia danych najszybszym okazał się Microsoft SQL Server, a najwolniejszym silnik MySQL. Dla operacji odczytu (poza testem dla 100 000 rekordów) najszybszy okazał się Microsoft SQL Server. Biorąc pod uwagę operację aktualizacji najwolniejszy okazał się Microsoft SQL Server, a najszybszy (poza testem dla 1 000 rekordów) PostgreSQL. Na temat operacji usuwania danych trudno jest wskazać konkretną tendencję.

Na podstawie przeprowadzonych badań można stwierdzić, że tworząc aplikację, w której do wykonywania zapytań używany jest Enitity Framework, a w strukturze bazy danych dominują pojedyncze tabele lub tabele połączone relacjami 1:1, jako silnik baz danych warto zastosować PostgreSQL, a unikać MySQL. Jeśli natomiast w strukturze bazy danych dominują tabele połączone relacjami 1:n, które nie rozbijają relacji m:n, kierując się wydajnością czasową należy unikać silnika MySQL, natomiast wybrać spośród technologii Microsoft SQL Server i PostgreSQL (zależnie od dominującej operacji na danych jaka będzie wykonywana przez system informatyczny). Jeżeli zaś w strukturze bazy danych dominują relacje m:n rozbite na dwie relacje 1:n wybór silnika bazy danych będzie zależał od rodzaju najczęściej wykonywanej operacji i liczby rekordów, dla których ta operacja będzie dotyczyła.

Literatura

[1] S. Cvetković, D. Janković, A Comparative Study of the Features and Performance of ORM Tools in a .NET Environment, Objects and Databases, Lecture Notes in Computer Science 6348 (2010) 147-158 https://doi.org/10.1007/978-3-642-16092-9_14.

[2] D. Zmaranda, L. Pop-Fele, C. Gyorödi, R. Gyorödi, G. Pecherle, Performance comparison of CRUD methods using NET object relational mappers: A case study, International Journal of Advanced Computer Science and Applications 11(1) (2020) 55-65, https://dx.doi.org/10.14569/ijacsa.2020.0110107.

[3] C.A. Győrödi, D.V. Dumşe-Burescu, D.R. Zmaranda, R.Ş. Győrödi, G.A. Gabor, G.D. Pecherle, Performance analysis of nosql and relational databases with couchdb and mysql for application's data storage, Applied Sciences (Switzerland) 10(23) (2020) 1-21 https://dx.doi.org/10.3390/app10238524.

[4] T. Seser, V. Plestina, F. Marjanica, Performance analysis of SQL prepared statements in CRUD operations. 7th International Conference on Smart and Sustainable Technologies, SpliTech (2022), https://dx.doi.org/10.23919/SpliTech55088.2022.9854303.

[5] A. Gruca, P. Podsiadło, Performance Analysis of .NET Based Object–Relational Mapping Frameworks, Beyond Databases, Architectures and Structures, Communications in Computer and Information Science 424 (2014) 40-49, https://dx.doi.org/10.1007/978-3-319-06932-6_5.

[6] A tour of the C# language, https://learn.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/, [17.06.2023]

[7] What is .NET? Introduction and overview, https://learn.microsoft.com/en-us/dotnet/core/introduction, [17.06.2023]

[8] Entity Framework Core, https://learn.microsoft.com/en-us/ef/core/, [17.06.2023]

[9] Microsoft SQL Server 2022 Licensing guide, https://download.microsoft.com/download/9/3/d/93d32de6-f268-45ed-ba25-2f9a6756b6af/SQL_Server_2022_Licensing_guide.pdf, [17.06.2023]

[10] What is MySQL?, https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html, [17.06.2023]

[11] PostgreSQL - about https://www.postgresql.org/about/, [17.06.2023]

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 30 (2024) 21–25
Received: 13 September 2023
Accepted: 18 October 2023

C++ and Kotlin performance on Android – a comparative analysis

Analiza porównawcza wydajności języków C++ i Kotlin na platformie Android

Grzegorz Zaręba*, Maciej Zarębski, Jakub Smołka

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

Abstract
The article discusses the comparison of C++ and Kotlin programming languages in a mobile environment. The authors performed a series of tests based on five selected algorithms: n-bodies, the n^{th} term of the Fibonacci sequence, reading and writing a file, and bubble sort for both small and large sets of values. The tests were carried out in a way that allowed to determine the performance of the Kotlin language both when it uses the Just-in-Time compilation mechanism and when it is not used. The research was carried out both on a physical mobile device and emulators. Although the C++ language outclassed its rival in most of the tests performed, Kotlin showed more than three times faster performance when bubble sorting on a small (20,000 values) array.

Keywords: programming language comparison; C++; Kotlin; Just-In-Time Compilation

Streszczenie
Artykuł porównuje języki programowania C++ i Kotlin w środowisku mobilnym. Autorzy wykonali serie testów w oparciu o pięć wybranych algorytmów: n-ciał, n-ty wyraz ciągu Fibonacciego, odczyt i zapis do pliku oraz sortowanie bąbelkowe dla małych oraz dużych zbiorów. Testy wykonano w sposób pozwalający określić wydajność języka Kotlin zarówno kiedy wykorzystuje on mechanizm kompilacji Just-in-Time, jak również gdy nie jest on używany. Badania przeprowadzono zarówno na fizycznym urządzeniu mobilnym, jak również emulatorach. Jakkolwiek język C++ zdeklasował rywala w większości wykonanych testów, Kotlin wykazał się ponad trzykrotnie większą szybkością działania przy sortowaniu bąbelkowym na małej (20 tysięcy wyrazów) tablicy.

Słowa kluczowe: porównanie języków programowania; Kotlin; C++; Kompilacja Just-In-Time

*Corresponding author
Email address: **grzegorz.zareba@pollub.edu.pl** (G. Zaręba)

1. Wstęp

Telefony komórkowe dwukrotnie już zrewolucjonizowały życie codzienne nieomal każdego człowieka – po raz pierwszy poprzez umożliwienie mu dzwonienia z dowolnego miejsca objętego zasięgiem sieci GSM, następnie zaś wraz z wejściem na rynek smartfonów oddając mu do ręki de facto uniwersalny i mobilny komputer z dostępem do Internetu. Największą popularnością obecnie cieszą się urządzenia z systemem Android do których należy niemalże 35% rynku [1]. Nie są one jednak idealne, ze względu na oparcie systemu o wirtualną maszynę Java, później zaś dołączenie doń języka Kotlin. Języki wspomniane nie cieszą popularnością w miejscach, w których priorytetyzowana jest szybkość wykonania operacji – w wypadku urządzeń mobilnych mogą to być operacje niskopoziomowe tudzież skomplikowane algorytmy, chociażby związane z kryptografią. Z myślą o takich zastosowaniach, Google umożliwiło kompilowanie dla urządzeń Android kodu języka C oraz C++ poprzez narzędzie zwane NDK (Native Development Kit).

Praca niniejsza jest próbą porównania wydajności tych dwóch języków o teoretycznie zgoła odmiennym charakterze. Miarą testów przeprowadzanych naprzemiennie jest czas realizacji zadanego algorytmu. Pytaniem na które chcą odpowiedzieć autorzy, jest zasadność przechodzenia przez skomplikowany proces wytwarzania aplikacji dla systemu Android (lub ich części) w C++, mając jako alternatywę stworzony przez JetBrains język Kotlin.

2. Cel i zakres badań

Celem badań jest porównanie wydajności kodu napisanego w językach C++ oraz Kotlin, uruchamianego w ramach jednolitej aplikacji działającej na maszynie wirtualnej Javy, w środowisku systemu Android. Porównanie opiera się na wynikach testów wydajnościowych, polegających na pomiarze czasu wykonania poszczególnych algorytmów, zaimplementowanych przez autorów w obu badanych językach.

Zakres badań obejmuje:

- zapoznanie się z istniejącą literaturą naukową,
- zaimplementowanie aplikacji testowej oraz algorytmów,
- wykonanie pomiarów wydajności zaimplementowanych algorytmów,
- analizę wyników.

3. Przegląd literatury

Jedną z prac traktujących o wydajności aplikacji mobilnych napisanych w różnych językach programowania jest [2]. Autorzy zaimplementowali prostą aplikację w środowisku systemu Android 2.2 w dwóch technolo-

giach. Jedną z nich był język Java, a drugą - stos technologiczny HTML + CSS + JavaScript w połączeniu ze szkieletem programistycznym PhoneGap. Jedną z istotnych obserwacji, poczynionych przez autorów omawianej pracy, jest fakt, że kod źródłowy niekompilujący się bezpośrednio do kodu bajtowego maszyny wirtualnej Java, musi zostać najpierw przetworzony w taki sposób, aby mógł być przez tę maszynę zrozumiany i wykonany. Z tej przyczyny wydajność aplikacji wykonanej w oparciu o szkielet PhoneGap była zauważalnie gorsza. Jako iż Kotlin jest kompilowany bezpośrednio do kodu bajtowego Javy, a C++ - nie, można domniemywać, że aspekt ten będzie miał również znaczenie przy porównaniu tychże języków.

Kolejnym artykułem który stanowił istotne dla autorów źródło wiedzy jest [3]. Użyte i opisane w nim algorytmy (jak chociażby N-bodies) znalazły zastosowanie w niniejszej pracy, przez wzgląd na łatwość implementacji i miarodajność wyniku, będącego de facto sumą operacji matematycznych, niewymagających zastosowania (jak choćby przy odczycie/zapisie do pliku) użycia specjalizwoanych bibliotek. Sam artykuł porównuje wydajność, zużycie zasobów oraz, co interesujące, zużycie prądu podczas realizacji algorytmu w kilkudziesięciu językach programowania. Wykonane badania ustanowiły sprawność czasową C++ i Javy względem języka C na odpowiednio: 1:1,56 oraz 1:1,89. Nie jest jednakoż jasnym jak uplasowałby się Kotlin, zwłaszcza uruchomiony w środowisku mobilnym, z myślą o którym został stworzony.

W doborze narzędzi istotna okazała się praca [4], pokazująca różnicę pomiędzy zastosowaniem NDK oraz bezpośredniej cross-kompilacji na ARM. Aby sprawdzić hipotezę, badacze przeprowadzili testy oparte o sześć algorytmów, między innymi Quicksort, Szybką transformację Fouriera czy Algorytm Djikstry. Artykuł wykazał wyższość rozwiązania NDK, wykazującego się zdecydowanie krótszymi czasami wykonania – w zależności od algorytmu od 27,6% aż do 114,4% szybciej od rozwiązania bezpośredniego.

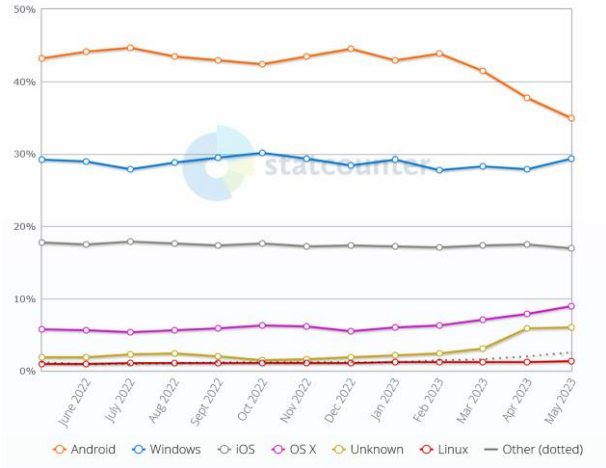
Kolejną godną uwagi publikacją, omawiającą problematykę wykorzystywania kodu C++ w aplikacjach dla systemu Android, jest prezentacja z wykładu [5]. Autor omawia w niej zalety oraz wady takiego podejścia, a także opisuje potencjalne problemy, na które natknąć może się programista w trakcie procesu budowania i uruchamiania swojej aplikacji wykorzystującej C++.

4. Omówienie technologii

Jako, iż tekst dotyczy wydajności kodu na platformie Android, większość zagadnień technologicznych, z którymi autorzy się zapoznali, jest związana z tym systemem.

4.1. System operacyjny Android

Android to otwartoźródłowy system operacyjny, oparty o jądro Linux, i rozwijany przez firmę Google. System jest wykorzystywany głównie na urządzeniach opartych o procesory ARM, takie jak smartfony, smartwatche, telewizory, czy komputery pokładowe w samochodach. Android to system operacyjny cieszący się dużą popularnością (Rysunek 1) - w poczet jego zalet zaliczyć można otwartoźródłowość, szeroką dostępność darmowych narzędzi wspomagających tworzenie oprogramowania, oraz możliwość wykorzystania wielu różnych języków, takich jak C++, Java, Kotlin, Dart, czy C# przy tworzeniu aplikacji. Testy opisane w niniejszej publikacji były przeprowadzane w środowisku Android 13, czyli najnowszej stabilnej wersji, dostępnej w momencie przygotowywania niniejszego artykułu.



Rysunek 1: Globalny udział rynkowy poszczególnych systemów operacyjnych w okresie kwiecień 2022 - kwiecień 2023 [1].

4.2. Język programowania C++

C++ to wieloparadygmatowy język programowania, bazujący na popularnym języku C. Jest on często stosowany tam, gdzie kluczowym jest osiągnięcie wysokiej wydajności kodu, na przykład przy obliczeniach związanych z grafiką trójwymiarową, bądź też w sytuacjach, w których programista potrzebuje zachować wysoki stopień kontroli nad sprzętem - przykładowo przy tworzeniu systemów operacyjnych [6]. C++ nie jest domyślnym językiem, zalecanym przez twórców systemu Android do tworzenia aplikacji na tej platformie. Mimo to, możliwe jest wykorzystanie kodu napisanego w tym języku poprzez zastosowanie narzędzi Android NDK, a w szczególności Java Native Interface - biblioteki umożliwiającej wywoływanie kodu C/C++ z poziomu kodu języka Java, bądź języków pochodzących od Javy, takich jak np. Kotlin [4].

4.3. Język programowania Kotlin

Kotlin to statycznie typowany język programowania działający na maszynie wirtualnej Javy, który jest głównie rozwijany przez JetBrains na potrzeby urządzeń z systemem Android, został on zresztą ogłoszony oficjalnym językiem programowania dla tychże na konferencji Google I/O 2019 [7]. Jedną z cech tego opartego o wirtualną maszynę Javy rozwiązania jest wykorzystanie kompilacji Just-in-time. Metoda ta kompiluje najczęściej wykonywane partie tworzonego kodu do postaci języka maszynowego. Skutkuje to skróceniem czasu wykonania. W języku wprowadzono ponadto elementy

nullpointer-safety, oraz uproszczono semantykę. Z perspektywy biznesowej największą zaletą języka jest jego interoperacyjność z zastępowaną przezeń Javą.

5. Metody badawcze

Aby porównać wydajność języków Kotlin i C++ na platformie Android, autorzy przygotowali aplikację testową, mającą służyć za środowisko do uruchamiania poszczególnych testów, rejestrowania czasów wykonania algorytmów testowych, oraz zapisywania zgromadzonych wyników. Szkielet aplikacji, logika odpowiadająca za obsługę interfejsu użytkownika, oraz za zbieranie i zapisywanie wyników zostały zaimplementowane w języku Kotlin. Testy przeprowadzono w środowisku systemu Android 13 na dwóch urządzeniach: maszynie wirtualnej dostarczonej przez środowisko programistyczne Android Studio, oraz fizycznym urządzeniu - smartfonie Samsung Galaxy A51.

Pomiary czasu wykonania testów zostały dokonane z użyciem funkcji „measureTimeMillis", pochodzącej ze standardowej biblioteki języka Kotlin. Funkcja ta przyjmuje jako parametr wskaźnik do funkcji, której czas wykonania ma zostać zmierzony. Uzyskany wynik zapisywany jest do zmiennej, którą następnie można na przykład wyświetlić bądź zapisać do pliku. Jest to prosty i skuteczny, a przy tym dostatecznie precyzyjny sposób na wykonanie pomiarów.

5.1. Zastosowane algorytmy

Dla każdego języka zaimplementowane zostało sześć testów, opartych na pięciu algorytmach:

- Sortowanie bąbelkowe
- Zapis pliku do pamięci telefonu
- Odczyt pliku z pamięci telefonu
- Symulacja n-ciał
- Obliczanie n-tego wyrazu ciągu Fibonacciego

Test oparty o sortowanie bąbelkowe został przeprowadzony w wersji ze zbiorem dwudziestu tysięcy elementów, oraz stu tysięcy elementów. Dla każdego z testów przygotowane zostały zestawy danych wejściowych, niezmienne pomiędzy kolejnymi uruchomieniami, tak aby zagwarantować powtarzalność testów.

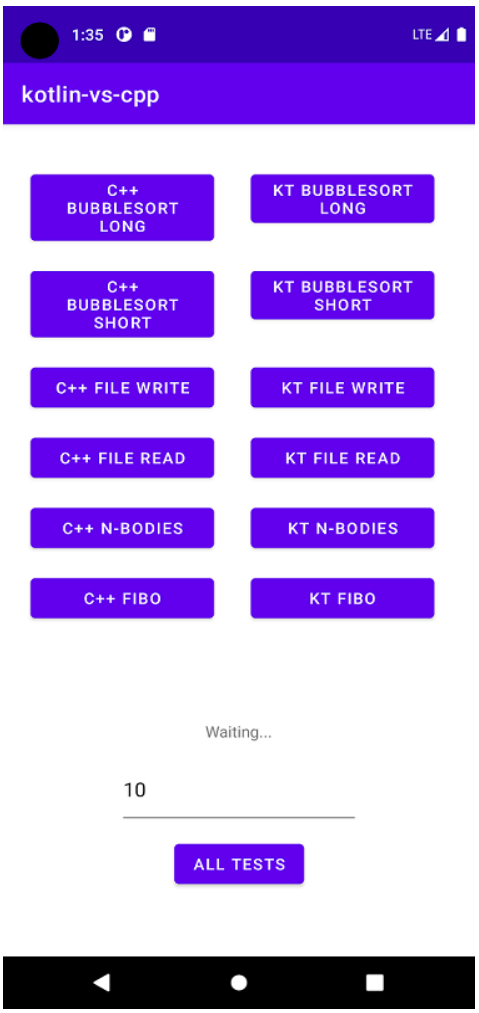
Test sortowania bąbelkowego polega na posortowaniu określonego zbioru danych. Sortowanie jest malejące, zaś przekazywany zbiór danych jest zawsze posortowany rosnąco. Sprawia to, że złożoność obliczeniowa algorytmu jest niezmienna dla każdego uruchomienia, co gwarantuje rzetelność wyników.

Test zapisu pliku polega na utworzeniu w określonej ścieżce pamięci telefonu pliku tekstowego, oraz zapisaniu do niego zadanego ciągu dwudziestu milionów znaków. Jeśli w momencie uruchomienia testu plik już istnieje, jest on usuwany przed rozpoczęciem pomiaru.

W ramach testu odczytu pliku, program ma za zadanie otworzyć uprzednio stworzony plik tekstowy i załadować jego zawartość, czyli ciąg dwudziestu milionów znaków, do zmiennej. Podobnie jak przy teście odczytu pliku, jeżeli w momencie uruchomienia testu plik nie istnieje, to jest on tworzony przed rozpoczęciem pomiaru.

Kolejny test, czyli symulacja n-ciał, to algorytm oparty na zasadach fizyki newtonowskiej, obliczający wielkości fizyczne opisujące ciała o zadanych parametrach, oraz ich zmiany wraz z upływem czasu. Algorytm zaimplementowany w ramach opisywanego testu symuluje pięć ciał niebieskich (Słońce, Neptun, Uran, Saturn, Jowisz) na przestrzeni dziesięciu tysięcy sekund, z krokiem czasowym wynoszącym jedną setną sekundy [3]. W ramach symulacji, przy każdym kroku czasowym obliczane jest położenie i prędkość każdego z ciał.

Ostatnim testem jest algorytm, obliczający w sposób iteracyjny wartość zadanego wyrazu ciągu Fibonacciego. Na potrzeby testu autorzy zdecydowali się każdorazowo znajdować wartość stumilionowego wyrazu ciągu.



Rysunek 2: Aplikacja testowa.

5.2. Aplikacja testowa

Interfejs graficzny aplikacji testowej przedstawiony został na rysunku 2. Aplikacja posiada jedną aktywność, z poziomu której można wywołać dowolny test, poprzez naciśnięcie odpowiednio opisanego przycisku. Ponadto możliwe jest wywołanie wszystkich testów za pomocą przycisku opisanego jako "All tests". Przycisk ten wywołuje wykonanie w losowej kolejności wszystkich zdefiniowanych w aplikacji testów określoną ilość razy - liczbę powtórzeń można podać w polu do wprowadzania liczb, znajdującym się powyżej.

5.3. Scenariusze testowe

W trakcie prowadzenia badań autorzy zauważyli, że mechanizm kompilacji just-in-time (JIT), wykorzystywany przez maszynę wirtualną Javy do optymalizowania kodu języka Kotlin, ma ogromny wpływ na wyniki eksperymentów. Aby uczynić prowadzone badania bardziej miarodajnymi, zdecydowano o przeprowadzeniu testów zgodnie z trzema różnymi scenariuszami. Należy dodać, że przez "pełny cykl testów", autorzy mają na myśli fakt pojedynczego wykonania każdego z sześciu przygotowanych testów dla obydwu języków - czyli przeprowadzenie łącznie dwunastu testów wraz z zapisem wyników.

Opracowano następujące scenariusze:

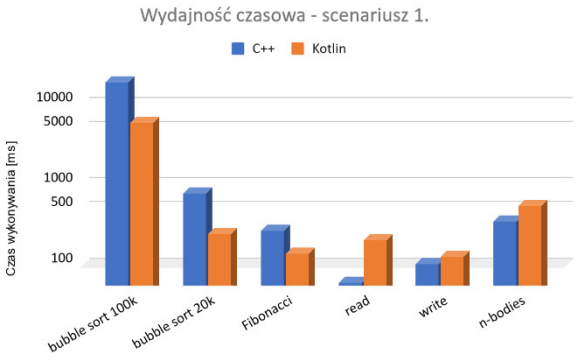
- Scenariusz nr 1 przewiduje stukrotne wykonanie pełnego cyklu testów, bez zamykania programu pomiędzy cyklami i przy standardowych ustawieniach środowiska Android, to jest z aktywnym mechanizmem kompilacji JIT.
- Scenariusz nr 2 również zakłada stukrotne wykonanie pełnego cyklu testów bez zamykania programu pomiędzy cyklami, ale w ramach tego scenariusza mechanizm kompilacji JIT jest nieaktywny.
- Scenariusz nr 3 zakłada trzynastokrotne wykonanie pełnego cyklu testów, przy czym po każdym cyklu aplikacja jest zamykana i uruchamiana ponownie. Kompilacja JIT jest aktywna.

6. Wyniki badań

Wszystkie badania zostały przeprowadzone na emulatorze systemu Android w wersji 13 (poziom API 33), dostarczonym przez środowisko Android Studio. Emulator uruchomiony był na systemie operacyjnym Windows 10, na komputerze opartym o architekturę x86 z następującymi podzespołami:

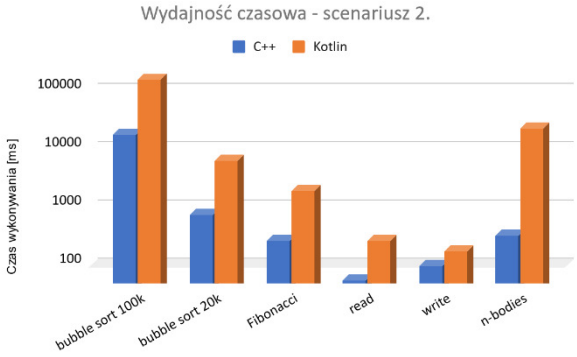
- procesor Inter Core i3 10100,
- 16 GB pamięci RAM DDR4,
- karta graficzna Nvidia GeForce 1060.

Należy dodać, że dla wszystkich trzech scenariuszy testowych, wydajność kodu C++ pozostała praktycznie niezmieniona. Główną różnicą pomiędzy scenariuszami był sposób wykorzystania mechanizmu kompilacji JIT, który ma wpływ tylko na wydajność kodu w języku Kotlin.



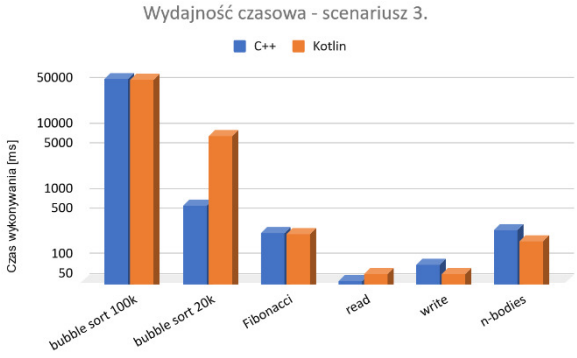
Rysunek 3: Wydajność czasowa w pierwszym scenariuszu.

W scenariuszu nr 1 Kotlin uzyskał najlepsze wyniki spośród wszystkich trzech scenariuszy (Rysunek 3). Okazał się być ponad dwukrotnie szybszy niż C++ zarówno w operacji sortowania bąbelkowego krótkiego i długiego zbioru, jak również w obliczaniu n-tego wyrazu ciągu Fibonacciego. Testy odczytu oraz zapisu do pliku, jak również symulacja n-ciał, wykonały się jednak blisko dwa razy szybciej w języku C++ niż w Kotlinie.



Rysunek 4: Wydajność czasowa w drugim scenariuszu.

W scenariuszu nr 2, którego wyniki widoczne są na powyższym rysunku (Rysunek 4) C++ okazał się być zdecydowanym zwycięzcą, pokonując Kotlina we wszystkich testach - wyłączenie mechanizmu kompilacji JIT drastycznie zmniejszyło wydajność Kotlina. Dla testów dotyczących sortowania bąbelkowego, Kotlin uzyskał ponad dwadzieścia razy gorszy wynik niż w scenariuszu nr 1, zaś dla symulacji n-ciał, wynik było około czterdzieści razy gorszy. Trzeba mieć na uwadze, że scenariusz ten raczej nie zaistniałby w warunkach normalnego użytkowania telefonu - wyłączenie mechanizmu kompilacji JIT nie ma praktycznego zastosowania poza prowadzeniem eksperymentów wydajnościowych.



Rysunek 5: Wydajność czasowa ostatniego scenariuszu.

W scenariuszu nr 3 Kotlin uzyskał wyniki zauważalnie gorsze niż w scenariuszu pierwszym, ale znacznie lepsze niż w drugim (Rysunek 5). Jak nietrudno zauważyć, różnica jest najbardziej widoczna w przypadku sortowania bąbelkowego, podczas gdy wyniki odczytu, zapisu, czy symulacji n-ciał pozostają zbliżone do wyników z pierwszego scenariusza.

7. Wnioski

Na podstawie wykonanych testów oraz analizy ich rezultatów można niemalże jednoznacznie wskazać wyższość języka C++ nad Kotlinem w dziedzinie szybkości działania, zwłaszcza w algorytmach bardziej złożonych. W zastosowaniach stosujących proste i repetetywne funkcje, a więc tam, gdzie ma zastosowanie mechanizm JIT, Kotlin wykazuje się nadspodziewaną szybkością, wykonując polecenia ponad dwukrotnie szybciej. Jakkolwiek całkowite zastąpienie Kotlina tudzież Javy przy tworzeniu aplikacji mobilnych nie jest zasadne, a wytwarzanie takowych aplikacji wiązałoby się z wykładniczo większym nakładem pracy, wskazanym i zasadnym jest zastępować język Kotlin wstawkami z C++, celem usprawnienia działania krytycznych elementów kodu, zwłaszcza tam, gdzie nie jest to związane ze współpracą z interfejsem użytkownika.

Literatura

[1] Globalne statystyki popularności wybranych systemów operacyjnych w latach 2022-2023, https://gs.statcounter.com/os-market-share, [28.06.2023]

[2] L. Corral, A. Sillitti, G. Succi. Mobile multiplatform development: An experiment for performance analysis, Procedia Computer Science 10 (2012) 736-743, https://doi.org/10.1016/j.procs.2012.06.094.

[3] R. Pereira, M. Couto, F. Ribeiro, R. Rua, J. Cunha, J. P. Fernandes, J. Saraiva, Energy Efficiency across Programming Languages: How Do Energy, Time, and Memory Relate, in: 10th ACM SIGPLAN International Conference (SLE'17), Vancouver, Canada, October 23–24, 2017.

[4] S. Lee, J. W. Jeon, Benchmarking Java application using JNI and native C application on Android, in: International Conference on Control, Automation and Systems (ICCAS), Gyeonggi-do, Korea (South), October 3-4, 2012, 1160-1163.

[5] M. Siggel, How to bring compute intensive C++ base apps to Android, in: Free and Open Source Conference 9 (FrOSCon) Sankt Augustin, Germany, August 23-24, 2014.

[6] What Is C++ Used For? https://www.codecademy.com/resources/blog/what-is-c-plus-plus-used-for/, [30.05.2023].

[7] Kotlin for Android, https://kotlinlang.org/docs/android-overview.html, [22.06.2023].

[8] N-body simulation – Wikipedia, https://en.wikipedia.org/wiki/N-body_simulation, [22.06.2023].

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 30 (2024) 26–30
Received: 25 September 2023
Accepted: 11 October 2023

Comparative analysis of NodeJs frameworks

Analiza porównawcza szkieletów programistycznych środowiska uruchomieniowego NodeJs

Bartłomiej Zima*, Marcin Barszcz

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

Abstract
The aim of the article is to compare two popular NodeJs frameworks. The analysis was performed for the ExpressJs and NestJs frameworks. Two proprietary applications supporting CRUD operations, implemented in these technologies, containing the same functionalities were used for the research. The comparison includes application performance, code metrics, documentation quality and completeness, and community support. The analysis showed that ExpressJs is minimalistic and suitable for less complex applications, while NestJs provides a standardized framework that allows the creation and development of large and complex projects.

Keywords: ExpressJs; NestJs; NodeJs; NodeJs frameworks

Streszczenie
Celem artykułu jest porównanie dwóch popularnych frameworków środowiska NodeJs. Analiza została przeprowadzona dla frameworków ExpressJs i NestJs. Do badań użyto dwóch autorskich aplikacji wspierających operacje typu CRUD, zaimplementowanych w tych technologiach, posiadających identyczne funkcjonalności. Porównanie obejmuje wydajność aplikacji, metryki kodu, jakość i kompletność dokumentacji oraz wsparcie społeczności. Analiza wykazała, że ExpressJs jest minimalistyczny i nadaje się do mniej złożonych aplikacji, podczas gdy NestJs zapewnia ustandaryzowaną strukturę pozwalającą na tworzenie i rozwijanie obszernych i skomplikowanych projektów.

Słowa kluczowe: ExpressJs; NestJs; NodeJs; szkielety programistyczne NodeJs

*Corresponding author
Email address: bartlomiej.zima@pollub-edu.pl (B. Zima)

1. Wstęp

Postęp w tworzeniu aplikacji internetowych, począwszy od momentu, gdy statyczne strony stworzone wyłącznie w języku HTML zaczęły rozwijać się w kierunku większej interaktywności z użytkownikiem, jest efektem wprowadzenia na rynek języka JavaScript,

Język ten powstał w 1995 roku i funkcjonował jako język skryptowy, działający po stronie klienta, dla przeglądarek internetowych [1], pozwalający na tworzenie interaktywnych stron. Elementami interaktywności były przykładowo walidowanie formularzy lub manipulowanie elementami HTML obecnymi na stronie. Z biegiem czasu wykonywanie wyłącznie synchronicznych operacji na stronach stało się kosztowne i znacznie spowalniające wydajność. W 2009 roku frustracja na rynku wywołana niezdolnością przetworzenia równoczesnych żądań liczonych w tysiącach osiągnęła takie rozmiary, że podjęto decyzję o znalezieniu rozwiązania [1].

Wówczas powstał NodeJs, czyli środowisko uruchomieniowe umożliwiające wykorzystywanie języka Javascript i Typescript (rozszerzenie JavaScript z wprowadzonymi typami danych) poza przeglądarką internetową. Technologia ta znacząco zwiększyła wydajność serwera wykorzystując pętlę zdarzeń (ang. Event loop) i operacje asynchroniczne wejścia i wyjścia, które służą do obsługi wielu zdarzeń jednocześnie [1].

Podobnie jak większość obecnie używanych technologii tego typu, NodeJs również doczekał się własnych szkieletów programistycznych (ang. Frameworks), pozwalających na składne i zorganizowane operowanie nim po stronie serwera. Opisywane frameworki mają następujące cechy i zastosowanie:

- ExpressJs – framework, powstały w 2010 roku w którym używany jest język Javascript. Oferuje zestaw funkcjonalności typu routing i oprogramowanie pośrednie Jest to framework elastyczny, nienarzucający ścisłych konwencji programistycznych, co daje dużą swobodę programistom. Jest podstawą dla wielu innych frameworków, włącznie z drugim poddanym analizie w niniejszym artykule – NestJs.
- NestJs – szkielet programistyczny, opracowany w 2017 roku który używa języka Typescript. W przeciwieństwie do języka Javascript, język ten pozwala na wykrycie błędów wynikających z typów danych już podczas kompilacji, a nie dopiero w trakcie działania programu. Ponadto NesJs posiada ściśle zdefiniowaną strukturę, inspirowaną technologią AngularJs [2]. Zawiera ona wbudowane wstrzykiwanie zależności (ang. Dependency injection), które pozwala na łatwiejsze zarządzanie zależnościami między modułami aplikacji. Sprawia to, że kod staje się łatwiejszy w utrzymaniu, testowaniu itp.

Celem artykułu jest przeprowadzenie analizy porównawczej tych dwóch frameworków, ze szczególnym

uwzględnieniem ich wydajności, jakości kodu, kompletności dokumentacji oraz wsparcia społeczności.

2. Metodyka i obiekt badań

W celu przeprowadzenia analizy porównawczej rozważanych w artykule frameworków opracowano dwie zaimplementowane od nowa aplikacje posiadające takie same funkcjonalności. Zwrócono szczególną uwagę na minimalizację wykorzystania bibliotek zewnętrznych, co podnosi miarodajność porównania. Podstawowymi elementami aplikacji były operacje typu CRUD. Do realizacji autoryzacji wykorzystano bibliotekę obsługującą JsonWebToken oraz bazę danych PostgreSQL.

Podczas implementacji aplikacji wykorzystano najpopularniejsze rozwiązania dostępne w obu frameworkach. Aby dokładniej zbadać wydajność aplikacji, ograniczono ilość kodu do minimum koniecznego do prawidłowego przetworzenia żądań. W przypadku autoryzacji wykorzystano nagłówek autoryzacyjny http.

Analizę porównawczą przeprowadzono na podstawie szeregu istotnych kryteriów, które stanowią kluczowe elementy oceny szkieletów programistycznych w środowisku uruchomieniowym NodeJs. Przeprowadzone badania koncentrowały się na następujących aspektach:

- Metryki kodu aplikacji – oceniono ilość linii kodu oraz plików niezbędnych do wykonania określonej operacji, zgodnie z potencjalną strukturą architektoniczną. Analiza tej metryki pozwoliła na ocenę stopnia skomplikowania kodu w poszczególnych szkieletach programistycznych.
- Czas i wydajność przetwarzania żądań http – przeprowadzono badania mające na celu zbadanie czasu odpowiedzi serwera dla aplikacji w różnych warunkach obciążenia, zarówno przy niewielkim, jak i znacznym obciążeniu. Analiza ta pozwoliła na ocenę wydajności poszczególnych szkieletów w kontekście obsługi żądań http. W celu oceny wydajności przetwarzania żądań http zastosowano narzędzie Apache JMeter, umożliwiające symulację różnych stopni obciążenia aplikacji.
- Jakość i kompletność dokumentacji – dokonano analizy oficjalnych dokumentacji frameworków pod kątem ich organizacji, przejrzystości i szczegółowości. Dokumentacja stanowi kluczowe źródło wiedzy podczas implementacji aplikacji, dlatego jej jakość ma istotny wpływ na proces programowania.
- Popularność i dostępność porad wśród społeczności – badania w zakresie popularności technologii zostały zrealizowane na podstawie statystyk z popularnych portali technologicznych. Popularność i dostępność porad wśród społeczności związane są z rozwojem Internetu i technologii używanych do tworzenia aplikacji. W wyniku tego rozwoju powstały społeczności skupiające entuzjastów, gdzie ludzie dzielą się wiedzą, pomagają sobie wzajemnie oraz dyskutują na temat napotkanych problemów technicznych, starając się je rozwiązać.

Wnikliwa analiza tych kryteriów pozwoliła na pełniejsze zrozumienie i ocenę szkieletów programistycznych w kontekście tworzenia aplikacji w środowisku NodeJs.

W niniejszych badaniach wykorzystano szereg kluczowych narzędzi i technologii, które odegrały istotną rolę w przeprowadzeniu analizy i testowania. Poniżej przedstawiono szczegółowe informacje na temat użytych wersji oprogramowania oraz parametrów sprzętu badawczego.

Wykorzystane wersje technologii:

- NodeJs: Wersja 18.12.1 – popularne środowisko uruchomieniowe oparte na silniku V8 JavaScript, stanowiło fundament aplikacji, umożliwiając dynamiczną i wydajną obsługę żądań.
- NestJs: Wersja 9.3.0 – to framework dla NodeJs, zapewnił skalowalność oraz strukturalność projektu, ułatwiając rozwój aplikacji poprzez zastosowanie wzorca architektonicznego.
- ExpressJs: Wersja 4.8.12 – to framework dla NodeJs, który posłużył do szybkiego tworzenia interfejsu API oraz zarządzania ścieżkami żądań.
- JMeter: Wersja 5.4.1 – to narzędzie do testowania wydajności aplikacji, było kluczowe w analizie obciążenia systemu oraz ocenie jego responsywności.
- PostgreSQL: Wersja 13.5 – to zaawansowany system zarządzania bazami danych, posłużył jako główna baza danych do przechowywania i zarządzania danymi aplikacji.

Parametry sprzętu badawczego:

- Rodzaj platformy: Komputer stacjonarny – został wybrany jako platforma badawcza, zapewniająca stabilność i wydajność podczas testowania aplikacji.
- Procesor: 13th Gen Intel(R) Core(TM) i7-13700K – wykorzystany procesor 13. generacji firmy Intel, charakteryzujący się wysoką mocą obliczeniową, był kluczowym elementem umożliwiającym szybkie przetwarzanie danych i zapytań.
- Pamięć RAM: 32GB – duża ilość pamięci RAM pozwoliła na płynne działanie aplikacji oraz manipulację dużymi ilościami danych podczas testów.
- System operacyjny: Windows 10 – wykorzystany system operacyjny Windows 10, popularny i szeroko używany, zapewnił stabilne środowisko pracy do przeprowadzenia eksperymentów i testów.

3. Dyskusja i analiza wyników

W poniższym rozdziale dokonano analizy zagadnień podjętych w niniejszym badaniu. Przedstawiono konteksty badawcze, w jakich omawiane szkielety programistyczne operują w środowisku uruchomieniowym NodeJs. Przeanalizowano również rezultaty przeprowadzonych badań w kontekście określonych kryteriów, dając pełny wgląd w porównawczą ocenę tychże szkieletów. Dokładnie przeanalizowano wpływ różnych czynników na wydajność i skalowalność i złożoność implementacji, a także zidentyfikowano mocne i słabe strony każdego szkieletu. Zawarto także omówienie

kwestii związanych z praktycznym wykorzystaniem szkieletów w projekcie aplikacji, uwzględniając aspekty takie jak łatwość użycia, społeczność wsparcia i dostępność dokumentacji. Analiza ta stanowi kluczowy etap, umożliwiający pełne zrozumienie i wykorzystanie wyników badań w kontekście praktycznego zastosowania szkieletów programistycznych w projektach oparty na NodeJs.

3.1 Metryki kodu aplikacji

NestJs jest szkieletem opartym na architekturze modułowej, co oznacza że aplikacja zostaje podzielona na niezależne moduły, z których każdy zawiera zbiór powiązanych komponentów, jak na przykład kontrolery, repozytoria lub modele. Taka struktura architektoniczna zapewnia skalowalność, ułatwia utrzymanie kodu i uniezależnia fragmenty systemu od siebie [3]. Z kolei ExpressJs nie narzuca żadnej konkretnej struktury aplikacji, dając deweloperowi swobodę projektowania architektury zgodnie z indywidualnymi preferencjami i potrzebami. Jednakże, w przypadku wykorzystania tego frameworka w większym projekcie, zaleca się zastosowanie jednego z wzorców architektonicznych, w celu zapewnienia łatwiejszej skalowalności [3].

Porównując kod aplikacji odpowiedzialny za obsługę żądania, w którym wymagane jest połączenie z bazą danych, w praktyce zrealizowano go na podstawie funkcji pobierającej wszystkie rekordy z tabeli w bazie danych.

W aplikacji ExpressJs, do pobrania danych z bazy PostgreSQL w praktyce wystarczy otworzyć połączenie do niej, podając odpowiednią konfigurację w pliku przetwarzającym żądanie http, wykonać zapytanie SQL i następnie zwrócić otrzymane dane użytkownikowi. Natomiast w NestJs potrzebne jest kilka dodatkowych kroków. Ze względu na silne typowanie tego frameworka, oprócz samej biblioteki do obsługi PostgreSQL, trzeba zainstalować bibliotekę umożliwiającą mapowanie obiektów na struktury bazodanowe (ang. Object-Relational-Mapping) [4]. Należy także zdefiniować konkretne encje w kodzie aplikacji, oznaczone adnotacjami @Entity(). Pola encji, jeśli nie stosujemy dodatkowej walidacji danych, należy oznaczyć adnotacjami @Column() i @PrimaryGeneratedColumn(). Oznaczają one odpowiednio zwykłe kolumny w bazie danych oraz kolumnę będącą kluczem głównym tabeli. Po nawiązaniu połączenia z bazą danych, repozytorium, operujące na danych zdefiniowanej encji, zostaje wstrzyknięte do serwisu. Ten serwis z kolei zostaje wstrzyknięty do odpowiedniego kontrolera, który obsługuje żądania HTTP. Kontroler użyje konkretnych funkcji serwisu, aby pobrać dane i następnie zwrócić je użytkownikowi.

Tabela 1 pokazuje liczbową różnicę wymaganych do implementacji jednej kompletnej operacji zasobów pomiędzy frameworkami. Aby zrealizować taką implementację w przypadku NestJs potrzebna jest względnie dużo większa ilość plików i linii kodu.

Tabela 1: Porównanie wymaganej ilości zasobów dla operacji GET na bazie danych

	Express	NestJs
Liczba plików	1	5
Łączna liczba linii kodu	16	53

3.2 Czas i wydajność przetwarzania żądań http

W celu zbadania średnich czasów przetwarzania niezautoryzowanych żądań poszczególnych metod http w symulacji jednego użytkownika, przeprowadzono 100 prób. Wyniki przedstawiono w Tabeli 2.

Tabela 2: Porównanie uśrednionych czasów odpowiedzi serwera dla niezautoryzowanych żądań typu CRUD

	GET	POST	PUT	DELETE
ExpressJs	37 ms	5 ms	14 ms	9 ms
NestJs	40 ms	11 ms	8 ms	40 ms

Porównując ExpressJs z NestJs, można zauważyć, że ExpressJs jest w stanie szybciej zwrócić odpowiedź w przypadku metod GET, POST, i DELETE, jednakże, jeżeli chodzi o metodę PUT, różnica w milisekundach między ExpressJs, a NestJs jest względnie największa. Tabela 3 przedstawia wyniki podobnych operacji, w tym przypadku żądania są autoryzowane.

Tabela 3: Porównanie uśrednionych czasów odpowiedzi serwera dla zautoryzowanych żądań typu CRUD

	GET	POST	PUT	DELETE
ExpressJs	24 ms	8 ms	15 ms	11 ms
NestJs	40 ms	14 ms	11 ms	44 ms

Żądania wymagające autoryzacji posiadają w większości dłuższe czasy odpowiedzi niż poprzednie, ale różnice pozostały relatywnie proporcjonalne. Można z tego wywnioskować, że w przypadku pojedynczych żądań, konieczność przetworzenia nagłówka autoryzacyjnego z JsonWebToken, nie wpływa znacznie na różnice czasu oczekiwania na odpowiedź serwera między frameworkami.

Przeprowadzono również szczegółowe testy wydajnościowe aplikacji, z podziałem na zautoryzowane i niezautoryzowane żądania http. Zasymulowano jednoczesne wysyłanie żądań łącznie dla 120 000 użytkowników w próbach po 40 000 każda. Uśrednione czasy odpowiedzi serwera, jak również procent żądań zakończonych niepowodzeniem zostały zawarte w Tabelach 4 i 5.

Tabela 4: Porównanie wyników testów obciążeniowych dla niezautoryzowanych metod typu CRUD

	ExpressJs		**NestJs**	
Metoda	Średni czas	% błędu	Średni czas	% błędu
GET	934 ms	22.5	1167 ms	28.75
POST	1056 ms	21.5	1211 ms	58.76
PUT	655 ms	65.40	327 ms	80.12
DELETE	610 ms	45.67	487 ms	97.37

Tabela 5: Porównanie wyników testów obciążeniowych dla zautoryzowanych metod typu CRUD

	ExpressJs		**NestJs**	
Metoda	Średni czas	% błędu	Średni czas	%błędu
GET	844 ms	30.48	221 ms	85.27
POST	19 ms	87.03	278 ms	86.6
PUT	838 ms	40.85	170 ms	87.22
DELETE	725 ms	42.45	318 ms	98.15

Tabele 4 i 5 wyraźnie pokazują, że ExpressJs jest bardziej odporny na obciążenia. Mimo wielu przypadków wyższego czasu oczekiwania na odpowiedź niż NestJs, potrafił procentowo przetworzyć znacznie więcej żądań od niego.

Co do NestJs można powiedzieć, że w przypadkach wymagających autoryzacji, framework jest wyjątkowo nieprzystosowany do dużego obciążenia.

Należy jednak pamiętać, że różnice w czasie odpowiedzi mierzonej w milisekundach, nawet jeśli wydają się duże między szkieletami programistycznymi, są praktycznie nieodczuwalne dla użytkownika końcowego.

3.3 Jakość i kompletność dokumentacji

W kontekście tworzenia aplikacji internetowych, poprawnie opracowana dokumentacja używanej technologii jest kluczowym elementem pomagającym zrozumieć używane funkcje i poprawnie je zaimplementować. Oba frameworki posiadają obszerne dokumentacje, które stanowią podstawowe źródło informacji i przewodników w procesie tworzenia systemu.

W przypadku NestJs dokumentacja jest bardzo szczegółowa i zorganizowana w sposób łatwy do nawigacji [4]. Została podzielona na sekcje tematyczne i zawiera przykłady użycia, co pomaga deweloperowi zrozumieć wykorzystanie danych funkcji w praktyce. Obejmuje szeroki zakres tematów, takich jak routing, autoryzacja i walidacja danych. Dokumentacja ta jest regularnie aktualizowana wraz z wprowadzaniem nowych wersji tego frameworka.

Dokumentacja ExpressJs również jest obszerna [5] i opisuje podstawowe koncepcje i funkcje frameworka. Jest łatwa do zrozumienia i zawiera przykłady kodu oraz opisuje różne techniki i wzorce, które można zastosować przy implementacji. Niemniej jednak dokumentacja ta jest mniej szczegółowa niż dokumentacja NestJs, więc znalezienie konkretnych informacji może zająć programiście więcej czasu.

Warto zaznaczyć, że oba szkielety programistyczne posiadają dokumentację dobrej jakości. NestJs wyróżnia się jako szkielet oferujący bardziej szczegółowe i kompleksowe informacje, podczas gdy ExpressJs dostarcza dokumentację, która jest łatwa do zrozumienia i zapewnia podstawowe informacje o frameworku. Co w przypadku minimalistycznego frameworka, można uznać za wystarczające. Dodatkowo w dokumentacji ExpresssJs można znaleźć wyjaśnienia pojęć którymi operuje, niekiedy również z graficznym zobrazowaniem i wyjaśnieniem procesów.

3.4 Popularność i dostępność porad wśród społeczności

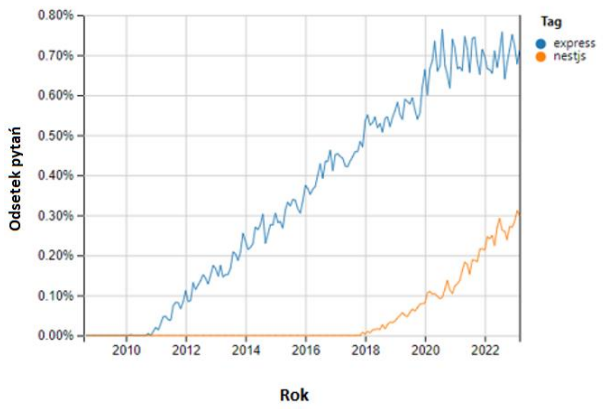
Do oceny popularności szkieletów programistycznych wykorzystano dane liczbowe zebrane ze społeczności programistycznej na platformie StackOverflow [6]. Jest to platforma działająca jak forum dyskusyjne, na której programiści zadają pytania dotyczące problemów technicznych, dzielą się wiedzą z innymi oraz uczestniczą w społeczności programistycznej. StackOverflow wykorzystuje tagi do kategoryzacji pytań, umożliwiając również wyszukiwanie konkretnych tagów i uzyskiwanie informacji o liczbie związanych z nimi wątków. Badania zostały przeprowadzone dnia 03.05.2023

- NestJs – tag [nestjs] posiada 11034 wątków, z czego 10 powstało w dniu przeprowadzenia badań [7].
- ExpressJs – tag [express] posiada 92969 wątków, przy czym 23 powstały w dniu przeprowadzania badań [8].

Ponadto istotnym wskaźnikiem jest liczba repozytoriów na platformie Github, która pełni funkcję bazy milionów projektów aplikacji internetowych. Platforma ta umożliwia również kategoryzację repozytoriów według technologii:

- NestJs – liczba repozytoriów oznaczonych tą technologią wynosiła 62 903 [9] w dniu przeprowadzenia badań
- ExpressJs – liczba repozytoriów oznaczonych tą technologią wynosiła 557 907 [10] w dniu przeprowadzenia badań

Rysunek 1 przedstawia wykres odsetka zadawanych pytań na StackOverflow dotyczących analizowanych frameworków. Obserwacje pokazują, że dla ExpressJs od roku 2010 do 2020 odsetek ten wyraźnie wzrastał. Jednak po roku 2020 zaczęły występować oscylacje między tendencją wzrostową a spadkową. Dla NestJs natomiast, od momentu jego wprowadzenia, trend ten niepodważalnie ukazuje wzrost.



Rysunek 1: Wykres procentowy zadawanych pytań na przestrzeni lat na platformie StackOverflow.

4. Wnioski

Po przeprowadzonej analizie porównawczej dwóch szkieletów programistycznych, można stwierdzić, że ExpressJs wyróżnia prostota i minimalistyczne podejście, co czyni go dobrym wyborem dla aplikacji o niskim stopniu skomplikowania. NestJs natomiast cechuje się bardziej strukturalnym podejściem opartym na architekturze modułowej, co dobrze sprawdzi się w bardziej złożonych projektach, zapewniając im lepszą skalowalność i łatwiejsze utrzymanie. Pomimo faktu, że wymaga on większej ilości zasobów takich jak czas programistów czy pamięć na dysku, do stworzenia jednej funkcjonalności niż ExpressJs, zapewnia im zorganizowaną formę i czyni kod bardziej zrozumiałym.

NestJs wyróżnia bogata i szczegółowa dokumentacja. Oferuje dużo informacji na temat wszystkich aspektów frameworka, zawiera przykłady kodu ułatwiające zrozumienie i implementację. ExpressJs również posiada czytelną dokumentację, jednakże nie tak szczegółową jak Nest.

Analizując liczbę wątków na dwóch popularnych platformach StackOverflow i Github, można stwierdzić że ExpressJs jest technologią bardziej popularną niż NestJs. Może to wynikać z faktu że został opracowany około 7 lat wcześniej [4]. Wykres przedstawiony na rysunku 1 pokazuje jednak, że NestJs zyskuje coraz większą popularność.

W przypadku dużego obciążenia aplikacji w jednym momencie, ExpressJs wykazuje się większą niezawodnością. Pomimo, że NestJs w wielu przypadkach był w stanie przetworzyć żądania znacznie szybciej, to wysoki procent żądań zakończonych niepowodzeniem sprawia, że w systemach narażonych na duże obciążenie nie jest on optymalnym wyborem.

Pod względem symulacji żądań pojedynczego użytkownika, frameworki nie różnią się znacząco. Jedynie w przypadku metody DELETE wyniki wykazują względnie największą różnicę. Warto zaznaczyć jednak, że w praktycznym użytkowaniu aplikacji te różnice są na tyle marginalne, że użytkownik końcowy nie odczuje ich znacząco.

Literatura

[1] R. Kempf, History of JavaScript, 2021 https://www.azion.com/en/blog/history-of-javascript/ , [06.05.2023]

[2] R. Benita, Clean Node.js Architecture – With NestJs and Typescript, https://betterprogramming.pub/clean-node-js-architecture-with-nestjs-and-typescript-34b9398d790f , [06.05.2023]

[3] S. Pasquali, Node.js Projektowanie, wdrażanie i utrzymywanie aplikacji, Helion, 2017

[4] Oficjalna dokumentacja technologii NestJs, https://docs.nestjs.com , [06.05.2023]

[5] Oficjalna dokumentacja technologii ExpressJs, https://expressjs.com , [06.05.2023]

[6] Oficjalna strona platformy StackOverflow: https://stackoverflow.com/ , [06.05.2023]

[7] Pytania otagowane słowem kluczowym „nestjs" na platformie StackOverflow, https://stackoverflow.com/questions/tagged/nestjs , [03.05.2023]

[8] Pytania otagowane słowem kluczowym „express" na platformie StackOverflow, https://stackoverflow.com/questions/tagged/express , [03.05.2023]

[9] Repozytoria NestJs na platformie Github, https://github.com/search?q=nestjs&type=repositories , [03.05.2023]

[10] Repozytoria ExpressJs na platformie Github, https://github.com/search?q=express&type=repositories&p=2 , [03.05.2023]

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 30 (2024) 31–38
Received: 28 September 2023
Accepted: 18 October 2023

User experience analysis in virtual museums

Analiza doświadczenia użytkownika w wirtualnych muzeach

Aleksandra Kobylska*, Mariusz Dzieńkowski

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

Abstract

The paper presents an analysis of user experience in virtual museums. The objects of interest are museums that offer a virtual walk which allows the user to visit and view exhibitions without leaving home. Among the selected objects for the study were the Museum of Auschwitz-Birkenau and the Malbork Castle Museum. The interfaces of these two virtual museums were subjected to eye tracking analysis using the Gazepoint GP3 HD eye tracker and an expert analysis using Nielsen heuristics. Additionally, a survey consisting of questions from the System Usability Scale and self-reported questions was then conducted to help gather information on usability and to collect opinions about online museums. The research group consisted of sixteen students from the Lublin University of Technology, who were presented with the same tasks to perform in the virtual museums. As a result of the research, it turned out that both according to the Nielsen heuristics and the System Usability Scale survey, the Auschwitz-Birkenau Museum was rated better, while in the eye tracking experiment both museums obtained similar results.

Keywords: virtual museum; user experience; eye tracking; Nielsen's heuristics; SUS

Streszczenie

Praca przedstawia analizę wrażeń użytkownika w wirtualnych muzeach. Przedmiotami zainteresowania są muzea, które oferują użytkownikowi wirtualny spacer umożliwiający zwiedzanie i oglądanie ekspozycji bez wychodzenia z domu. Wśród obiektów wybranych do badań znalazły się Muzeum Auschwitz-Birkenau oraz Muzeum Zamkowe w Malborku. Interfejsy tych dwóch wirtualnych muzeów zostały poddane analizie eyetrackingowej z użyciem urządzenia Gazepoint GP3 HD oraz analizie eksperckiej z wykorzystaniem heurystyk Nielsena. Dodatkowo przeprowadzono ankietę składającą się z pytań zawartych w Skali Użyteczności Systemu oraz pytań własnych, która pomogła w ocenie użyteczności oraz w zebraniu opinii na temat muzeów online. Grupę badawczą stanowiło szesnastu studentów Politechniki Lubelskiej, którym przedstawiono takie same zadania do wykonania w wirtualnych muzeach. W efekcie przeprowadzonych badań okazało się, że zarówno według heurystyk Nielsena jak i przeprowadzonej ankiety Skali Użyteczności Systemu lepiej zostało ocenione Muzeum Auschwitz-Birkenau, natomiast w eksperymencie eyetrackingowym oba muzea uzyskały podobne wyniki.

Słowa kluczowe: wirtualne muzeum; doświadczenie użytkownika; eyetracking; heurystyki Nielsena; SUS

*Corresponding author

Email address: aleksandra.kobylska@pollub.edu.pl (A.Kobylska)

1. Wstęp

W dzisiejszych czasach, wraz z postępem technologicznym, coraz więcej podstawowych segmentów życia codziennego zostało zautomatyzowanych i zinformatyzowanych. Rosnąca popularność wykorzystywania technologii komputerowych została spotęgowana również przez globalną pandemię COVID-19, która wymusiła na społeczeństwie szereg zmian w sposobie funkcjonowania. W odpowiedzi na to, wiele muzeów zdecydowało się przenieść swoje działania do przestrzeni cyfrowej, która umożliwiła organizowanie wirtualnych wycieczek po muzeach i podziwianie interaktywnych wystaw bez wychodzenia z domu. Dostosowanie się przemysłu muzealnego do zmieniających się okoliczności i rozwoju technologicznego sprawiło, że obcowanie z obiektami dziedzictwa kulturowego stało się wygodniejsze i łatwiej dostępne niż kiedykolwiek wcześniej.

Wirtualne muzea jako alternatywny sposób postrzegania historii i sztuki oferują zwiedzającym serię unikalnych przeżyć i emocji bez ograniczeń czasowych czy geograficznych. Jednakże, możliwość przeglądania dzieł sztuki z zacisza własnego domu nie jest jedynym czynnikiem kształtującym doświadczenia i odczucia użytkownika. Zagadnieniami równie znaczącymi, które mają wpływ na wywarcie pozytywnego wrażenia na zwiedzającym mogą być zarówno aspekty wizualne, takie jak: wysoka jakość prezentowanych obiektów czy sposób ich przedstawienia i oświetlenia jak i kwestie techniczne, wśród których można wymienić: przejrzysty i intuicyjny interfejs, szybkość ładowania eksponatów, a także łatwość poruszania się po obiekcie. Dodatkowo, trwające badania nad zwiększeniem interaktywności między przedmiotami muzealnymi a użytkownikiem sprawiają, że wirtualne zwiedzanie staje się jeszcze bardziej wciągające i interesujące [1].

2. Cel, zakres pracy i hipotezy badawcze

Celem pracy jest poznanie opinii użytkowników na tematy związane z wirtualnymi muzeami oraz analiza

doświadczenia użytkowników o dwóch wybranych muzeach: Muzeum Auschwitz-Birkenau oraz Muzeum Zamkowego w Malborku. Instytucje te pozwalają zainteresowanym na zapoznanie się ze swoimi zbiorami poprzez witrynę internetową.

Temat badawczy dotyczy aspektów związanych z analizą doświadczenia użytkownika pod kątem zrozumiałości i przejrzystości treści prezentowanych w interfejsie oraz wrażeń, jakie wywierają zwiedzane obiekty. Przeprowadzone badania i ocena otrzymanych wyników mogą pomóc w identyfikacji potencjalnych problemów lub zagadnień, których poprawa zwiększyłaby użyteczność i przyjemność korzystania z wirtualnych muzeów.

Zakres pracy obejmuje następujące kwestie:

- przegląd zagadnień związanych z podjętym tematem, ustalenie celu badań i sformułowanie hipotez,
- wybór obiektów badań oraz metod badawczych (eyetracking, SUS, analiza heurystyczna, autorska ankieta),
- zaprojektowanie eksperymentu, zorganizowanie grupy badawczej i przeprowadzenie badań,
- analiza i dyskusja wyników oraz wnioski.

W ramach pracy sformułowano dwie hipotezy badawcze:

H1: *Ludzie są zainteresowani odwiedzaniem wirtualnych muzeów, ale oczekują od nich wysokiej interaktywności oraz informacyjności.*

H2: *Interfejsy wirtualnych muzeów wymagają udoskonalania, aby osiągnąć wysoki poziom doświadczenia użytkownika.*

3. Przegląd literatury

Zaznajomienie się z najważniejszymi publikacjami dotyczącymi eyetrackingu oraz doświadczeń użytkowników w wirtualnych muzeach umożliwia zrozumienie i podsumowanie dotychczasowej wiedzy na wymienione tematy. W pierwszym artykule autorzy zwracają uwagę na ograniczenia związane z trudnością i kosztownością między innymi transportu oraz ubezpieczenia bezcennych i delikatnych obiektów muzealnych, które cieszą się największą popularnością [1]. Rozwiązaniem, które zostało zaproponowane w celu zwiększenia dostępności do takich ekspozycji jest wykorzystanie wirtualnej rzeczywistości (VR) i rzeczywistości rozszerzonej (AR). Tworzenie wirtualnych wystaw poprzez digitalizację przedmiotów muzealnych umożliwia technologia zastosowana w projekcie ARCO [2]. Narzędzia dostarczane przez ten system zostały przedstawione jako kompleksowe rozwiązanie, które pozwala projektantom wystaw na sprawne tworzenie cyfrowych reprezentacji obiektów muzealnych takich jak: obrazy, modele 3D, filmy, opisy oraz nagrania dźwiękowe. Autorzy podkreślają również, iż wykorzystanie X-VRML czyli wysokopoziomowego języka opartego na XML możliwe jest opracowanie szablonów ze zbiorem parametrów, dzięki czemu zdefiniowanie odpowiedniej reguły umożliwia użycie instancji obiektu w wielu przestrzeniach wystawienniczych.

Autorka drugiego artykułu bada wpływ różnych czynników na doświadczenie i zadowolenie odwiedzających muzea. Zagadnienie zostało zbadane w odniesieniu do społecznych, psychologicznych i środowiskowych aspektów wizyty [3]. Metodą badawczą wykorzystaną tutaj była obserwacja zachowania obserwatorów w muzeum w Birmingham. Następnie, uzyskane dane poddano analizie tematycznej, w wyniku której powtarzające się działania wśród podobnych jednostek lub grup stały się podstawą ostatecznej interpretacji zachowań. Pierwsza z wymienionych perspektyw podaje przykład wizyty w muzeum jako symbolu statusu, który świadczy o tym, że pewna grupa odwiedzających czerpie przyjemność z samego faktu wyjścia do muzeum. Traktowane jest ono wtedy jako miejsce, które inicjuje nowe tematy do rozmów i stawia czas spędzany w towarzystwie grupy ponad czerpanie indywidualnej satysfakcji z oglądania wystawy. Istotą podejścia psychologicznego jest zrozumienie motywacji, zaangażowania oraz uważności odwiedzających. Zwrócona zostaje również uwaga na odpowiednie otoczenie i przestrzeń muzealną, które w zależności od fizycznych i estetycznych aspektów mogą wpłynąć pozytywnie jak i negatywnie na doświadczenie odwiedzających.

Metody, które wyłoniły się z projektowania stron internetowych i aplikacji mobilnych mogą przyczynić się do poprawy doświadczeń użytkowników (UX) odwiedzających muzea. W pracy [4] autorka postanowiła zrozumieć doświadczenie zwiedzających muzeum poprzez wykorzystanie narzędzi z badań UX i użyteczności. W przeprowadzonych badaniach oceniała preferencje odwiedzających wśród czterech różnych nośników informacji w module wystawienniczym wykorzystując do tego śledzenie wzroku oraz ankietę. Wyniki pokazały, że interaktywne nośniki, takie jak tablety i ekrany, nawet jeśli nie są preferowane przez większość uczestników, po zaangażowaniu się w te nośniki, wchodzą w interakcję z nimi przez dłuższy czas. Wiedza ta może być wykorzystana przez kuratorów i menadżerów ekspozycji do poprawienia doświadczeń odwiedzających muzea.

Celem kolejnej pracy [5] było lepsze zrozumienie wizualnego poznania zwiedzających, aby poprawić i uatrakcyjnić odbiór wystaw fizycznych lub internetowych. W zrealizowanych badaniach śledzono i analizowano trajektorie ruchów oczu odwiedzających muzeum oraz badano ich związek z uczeniem się i poznawaniem, a także szukano klucza do wpływania na poznanie poprzez analizę sekwencji zachowań użytkownika podczas wyświetlania treści. Wyniki pokazują, że osoby zainteresowane wyświetlanymi treściami mają lepszą wydajność poznawczą, są zanurzone w czytaniu tekstu i w ich ruchach oczu widać wyraźne zmiany. Z kolei osoby niezainteresowane wyświetlaną treścią są rozproszone i często zwracają uwagę z powrotem na tytuł treści. W badaniu tym ruch gałek ocznych i fiksacje są wskaźnikami, które można wykorzystać jako punkt odniesienia dla przyszłego projektowania wyświetlaczy w celu poprawy skuteczności prezentowania informacji odwiedzającym.

4. Wykorzystywane metody badawcze

Zarówno w tej, jak i innych pracach badawczych, wybór odpowiednich narzędzi i technik jest kluczową kwestią, ponieważ w znacznym stopniu wpływa na jakość gromadzenia danych oraz sposób zrozumienia badanych aspektów. Wśród wykorzystywanych metod znalazły się: technika okulograficzna, heurystyki Nielsena, Skala Użyteczności Systemu (SUS) oraz ankieta autorska.

4.1. Technika eyetrackingowa

Technika eyetrackingowa służy do monitorowania, rejestracji i pomiaru ruchów gałek ocznych. Obecnie jest ona stosowana w różnorodnych obszarach, wśród których można wymienić: badania naukowe, medycynę, psychologię, rozrywkę, marketing czy też interakcję człowiek-komputer [6].

Aktywność skupienia uwagi na bodźcach wizualnych może przejawiać się poprzez różne rodzaje ruchów oczu. Są one wynikiem procesów, które polegają na wybieraniu i koncentrowaniu się na elementach wizualnych, które docierają do systemu wzrokowego [7]. Dlatego istotne staje się zrozumienie podstawowych cech i charakterystyk rodzajów ruchów oczu. Należą do nich fiksacje, czyli momenty w czasie, podczas których oczy utrzymują i skupiają wzrok na względnie stałym punkcie lub obiekcie oraz sakady - krótkotrwałe, gwałtowne i skokowe poruszenia oczami, które pozwalają na przemieszczenie wzorku z jednego punktu na drugi. W czasie tych ruchów nie następuje pobieranie informacji wizualnych [8].

Najpopularniejszymi formami, w których prezentowane są wyniki z badań okulograficznych są ścieżki wzroku i mapy cieplne. Ich definicje są następujące:

- ścieżki wzroku – pokazują drogę, jaką oczy użytkownika pokonują na ekranie w określonym czasie;
- mapy cieplne – graficzne reprezentacje obszarów na ekranie, które różnicując je kolorami, pokazują, jakie miejsca najbardziej przyciągają uwagę badanego.

4.2. Heurystyki Nielsena

Jacob Nielsen jako specjalista w dziedzinie użyteczności, wspólnie z Ralfem Molichem, zaproponował zestaw dziesięciu zasad dotyczących interakcji człowieka z maszyną, a dokładniej związanych z projektowaniem i analizowaniem interfejsów witryn internetowych. Zasady te zostały następnie nazwane heurystykami i wykorzystywane są do dziś w celu oceny spójności systemu oraz wykrywania ich potencjalnych błędów [9]. Mogą być również traktowane jako lista kontrolna, której poszczególne elementy zostały przedstawione w Tabeli 1 [10].

Tabela 1: Treści twierdzeń heurystyk Nielsena

Lp.	Treść heurystyki
1	Pokazuj status systemu.
2	Zachowaj zgodność pomiędzy systemem a rzeczywistością.
3	Daj użytkownikowi pełną kontrolę.
4	Trzymaj się standardów i zachowaj spójność.
5	Zapobiegaj błędom.
6	Pozwalaj wybierać zamiast zmuszać do pamiętania.
7	Zapewnij elastyczność i efektywność.
8	Dbaj o estetykę i umiar.
9	Zapewnij skuteczną obsługę błędów.
10	Zadbaj o pomoc i dokumentację.

Wśród zalet metody heurystycznej wymienić można prostotę i szybkość, jednak należy pamiętać, iż jej jakość zależy od wiedzy i doświadczenia osób przeprowadzających analizę.

4.3. Kwestionariusz – Skala Użyteczności Systemu

Skala Użyteczności Systemu to narzędzie umożliwiające pomiar użyteczności między innymi systemów internetowych za pomocą ankiety. Jest to ustandaryzowany kwestionariusz, który składa się z dziesięciu stwierdzeń, wśród których pozycje o numerach nieparzystych mają wydźwięk pozytywny, natomiast pozycje o numerach parzystych – negatywny [11]. Treści twierdzeń podlegających ocenie w ankiecie SUS zostały przedstawione w Tabeli 2.

Tabela 2: Treści twierdzeń Skali Użyteczności Systemu

Lp.	Treść twierdzenia
1	Myślę, że chciałbym często korzystać z tego systemu.
2	Uważam, że system jest niepotrzebnie skomplikowany.
3	Myślałem, że system jest łatwy w użyciu.
4	Myślę, że mógłbym korzystać z tego systemu, jednak potrzebowałbym wsparcia osoby technicznej.
5	Odkryłem, że różne funkcje w systemie są dobrze zintegrowane.
6	Uważam, że w tym systemie jest zbyt wiele niespójności.
7	Uważam, że większość osób będzie w stanie opanować system bardzo szybko.
8	Uważam, że system jest bardzo niewygodny w użyciu.
9	Czułem się bardzo pewnie korzystając z systemu.
10	Musiałem nauczyć się wielu rzeczy, zanim mogłem zacząć korzystać z tego systemu.

Skala pytań zawiera się od „Zdecydowanie się nie zgadzam” do „Zdecydowanie się zgadzam”. Wynik zostaje obliczony za pomocą przypisania punktów do poszczególnych wariantów odpowiedzi zgodnie ze schematem [12]. Następnym krokiem jest zsumowanie punktów i pomnożenie uzyskanej wartości przez 2,5, dzięki czemu możliwe będzie policzenie wskaźnika, którego wartość powinna znajdować się w przedziale od 0 do 100. Uznaje się, że wynik powyżej 68 punktów określa wartość powyżej średniej i jest traktowany jako dobra ocena. Natomiast, aby system został oceniony bardzo dobrze i znalazł się w górnych 10% wszystkich witryn powinien uzyskać co najmniej 80 punktów [13].

5. Metodyka badań

Badanie doświadczenia użytkowników w wirtualnych muzeach podzielone zostało na część eyetrackingową, ankietową oraz badanie użyteczności systemu z wykorzystaniem heurystyk Nielsena. Czas, w którym osoba

badana wykonywała dwa pierwsze etapy badania wynosił około 15 minut, natomiast ocena heurystyczna została wykonana zdalnie.

5.1. Obiekty badań

Wybranymi obiektami badań są dwa wirtualne muzea, które różnią się nie tylko tematyką, ale również możliwościami oferowanymi przez interfejs. Należą do nich Muzeum Auschwitz-Birkenau oraz Muzeum Zamkowe w Malborku (Tabela 3).

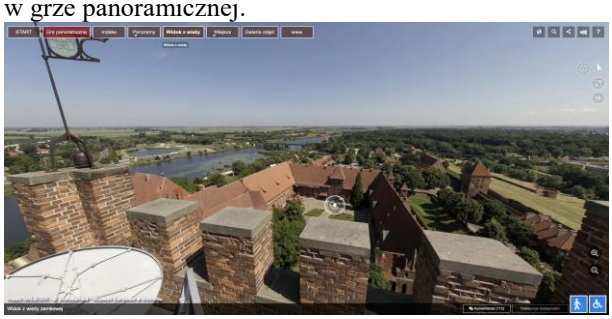
Tabela 3: Wybrane muzea z adresami do wirtualnego zwiedzania

Lp.	Nazwa muzeum	Adres wirtualnego muzeum
1	Muzeum Auschwitz-Birkenau	https://panorama.auschwitz.org
2	Muzeum Zamkowe w Malborku	https://muzeumzamkowewmalborku.wkraj.pl/html5/index.php?id=34844

Pierwsze wybrane muzeum jest wyjątkowo interesujące ze względu na ogromne znaczenie dla edukacji i upamiętnienia oraz przekazywania historii. Jego wirtualna forma (Rysunek 1) oferuje dostęp do unikalnych zdjęć i materiałów związanych z obozem i pomaga odkrywać to tragiczne miejsce, przybliżyć jego kontekst historyczny oraz oddać hołd pamięci ofiar.

Rysunek 1: Interfejs wirtualnego Muzeum Auschwitz-Birkenau.

Drugim wirtualnym muzeum jest Muzeum Zamkowe w Malborku (Rysunek 2), które za pomocą interaktywnych map umożliwia eksplorację zamkowych pomieszczeń, dziedzińców i korytarzy oraz oglądanie detali architektury i wystroju. Platforma tego muzeum wyróżnia się również możliwością wzięcia udziału w grze panoramicznej.



Rysunek 2: Interfejs wirtualnego Muzeum Zamkowego w Malborku.

5.2. Grupa badawcza

W badaniu eyetrackingowym oraz ankietowym wzięło udział 16 studentów drugiego roku informatyki na Politechnice Lubelskiej. Zdecydowana większość grupy badawczej składała się z mężczyzn w wieku od 21 do 23 lat (62,5% wszystkich osób). W badaniu brały udział dwie kobiety i trzy osoby powyżej 27 roku życia. Do badania użyteczności systemu z wykorzystaniem heurystyk Nielsena zaproszono dwóch studentów ostatniego roku informatyki drugiego stopnia, którzy mają wiedzę na temat projektowania i ergonomii interfejsów.

5.3. Stanowisko badawcze

Stanowisko eyetrackingowe to specjalnie przygotowane miejsce, które powinno być wyciszone, odpowiednio oświetlone i wyposażone w meble umożliwiające wygodne i nieograniczające usadzenie badanego oraz wyposażone w niezbędne urządzenia. Dlatego też, dedykowany obszar badawczy składał się z biurka, krzesła, laptopa z zamontowanym urządzeniem eyetrackingowym oraz podłączonej myszki (Rysunek 3). Stanowisko znajdowało się w jednej z sal budynku Centrum Technologii Informatycznych i Lingwistyki Technicznej CEN-TECH należącego do Politechniki Lubelskiej.

Rysunek 3: Stanowisko eyetrackingowe.

Eyetracker Gazepoint GP3 HD, czyli stacjonarny sprzęt do śledzenia ruchu gałek ocznych był umieszczony bezpośrednio pod ekranem laptopa. Ponadto jest on małych rozmiarów (235x45x47 mm) i waży 125 g, co sprawia, że jest kompaktowy i co ułatwia jego przenoszenie do różnych miejsc, w których przeprowadzane będą badania [14]. Eyetracker GP3 HD jest urządzeniem pracującym z częstotliwością próbkowania do 150 Hz, które oferuje wysoką rozdzielczość monitorowania i dokładność. Za pomocą kamer na podczerwień rejestruje dane dotyczące reakcji użytkowników na treści wyświetlane na monitorze, a jego wykorzystanie ograniczone jest do współpracowania z nowszymi wersjami systemu Windows i ekranów o rozdzielczości maksymalnie 24 cale. Jest to zarówno profesjonalny jak i wszechstronny sprzęt, który dzięki przystępnej cenie oraz wysokiej jakości jest wysoce popularny w dziedzinie eyetrackingu.

Laptopem wykorzystywanym w eksperymencie był Acer Nitro 5 wyposażony w procesor AMD Ryzen 7 5800H i pamięć RAM 32 GB z systemem operacyjnym Windows 10. Natomiast oprogramowaniem, które umożliwiło stworzenie projektu badania, przedstawienie

go uczestnikom oraz wygenerowanie danych i wyników było iMotions w wersji 9.1.

5.4. Przebieg badania

Czynności przygotowawcze i sam eksperyment obejmowały następujące etapy:

1. Określenie problemu badawczego i celu badań.
2. Wybranie obiektów badawczych czyli interfejsów dwóch wirtualnych muzeów.
3. Dobranie metod badawczych.
4. Sporządzenie scenariusza oraz przygotowanie materiałów wyświetlanych użytkownikom w części eyetrackingowej badania.
5. Opracowanie ankiety z wykorzystaniem kwestionariusza SUS oraz pytań własnych.
6. Przygotowanie stanowiska badawczego, a następnie projektu eksperymentu w oprogramowaniu iMotions.
7. Zaproszenie grupy badawczej i zapoznanie jej z tematem i przebiegiem badań.
8. Przeprowadzenie badania eyetrackingowego uwzględniającego przygotowanie uczestników i kalibrację.
9. Wypełnienie ankiety.
10. Wykonanie oceny użyteczności systemu na podstawie heurystyk Nielsena.
11. Zgromadzenie i dokładna analiza uzyskanych danych.
12. Sformułowanie wniosków i podsumowanie badań.

Eksperyment eyetrackingowy został przygotowany w programie iMotions. Składał się on z 9 poleceń z podziałem na zadania statyczne i dynamiczne (Tabela 4).

Tabela 4: Polecenia do wykonania w części eyetrackingowej badania

Lp.	Rodzaj zadania	Treść polecenia
z1	Statyczne	Znajdź ikonę powiększenia obrazu.
z2		Znajdź ikonę udostępniania.
z3		Znajdź ikonę umożliwiającą powiększenie widoku na pełny ekran.
z4		Znajdź informację, w jakim miejscu się teraz znajdujesz.
z5		Znajdź ikonę „Pomoc”.
z6	Dynamiczne	Przejdź do innego widoku/pomieszczenia za pomocą jednego z widocznych przycisków interaktywnych (przycisk strzałki przy obiektach/na drodze).
z7		Obróć widok na północ. (Pomocne znalezienie/naciśnięcie ikony kompasu).
z8		Zmień panoramę na jedną z proponowanych/gotowych.
z9		Otwórz mapę, a następnie zmień oglądany widok poprzez naciśnięcie wybranego punktu.

Pierwsza grupa poleceń polegała na odnalezieniu podanego w treści instrukcji elementu na statycznym obrazie z wirtualnego muzeum. Polecenia z drugiej grupy wymagały wykonania określonej czynności za pomocą funkcjonalności dostępnych w interfejsie. Zarówno kolejność zadań jak i muzea były odpowiednio wymieszane tak, aby zminimalizować zapamiętywanie rozmieszczenia elementów. Późniejszej analizie zostały poddane tylko polecenia statyczne, natomiast dynamiczne miały na celu lepsze zapoznanie uczestnika z możliwościami oferowanymi przez cyfrowe muzea.

Po wykonaniu badania eyetrackingowego, uczestnikowi badania przedstawiana była ankieta, która składała się z kwestionariusza SUS (osobnego dla każdego muzeum), ankiety własnej dotyczącej opinii na temat wirtualnych muzeów oraz dwóch pytań otwartych. Dzięki tym pytaniom możliwe było lepsze zrozumienie perspektywy i doświadczenia osób badanych, które zapoznały się z wirtualnymi muzeami i jednocześnie poznanie, co uważają za ich najlepsze i najsłabsze strony. Treści tych pytań były następujące:

1. Co uważasz za główną zaletę wirtualnych muzeów?
2. Co uważasz za główną wadę wirtualnych muzeów?

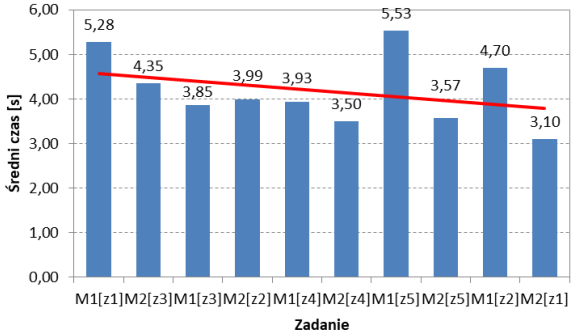
6. Wyniki badań

Przeprowadzenie obu części badań, czyli eyetrackingowych i ankietowych, pozwoliło na uzyskanie wyników dotyczących doświadczeń użytkowników w wirtualnych muzeach. Następnie, przedstawione zostały średnie ocen otrzymane z badania heurystycznego, które zostało przeprowadzone przez dwoje ekspertów.

6.1. Wyniki badania eyetrackingowego

Wyniki uzyskane z badania eyetrackingowego mają charakter jakościowy - obrazy z widocznymi mapami cieplnymi i ścieżkami wzroku oraz ilościowy w postaci czasów realizacji poszczególnych zadań przez użytkowników.

Na Rysunku 4 zestawiono średnie czasy uzyskane przez uczestników badania dla 5 zadań statycznych wykonywanych na obu muzeach.



Rysunek 4: Wykres średnich czasów dla każdego zadania z części eyetrackingowej badania.

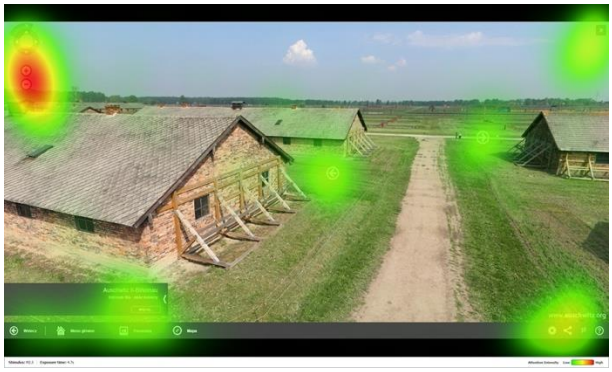
Zadania zostały przedstawione w takiej kolejności, w jakiej były wyświetlane użytkownikom, z pominięciem zadań dynamicznych, które nie zostały poddane analizie. Na wykresie poszczególne polecenia zostały oznaczone skrótami, gdzie M1 oznacza Muzeum Zamkowe w Malborku, natomiast M2 – Muzeum Auschwitz-Birkenau. Dodatkowo zadania zostały ponumerowane od z1 do z5 i były jednakowe dla obu interfejsów. Do obliczeń średnich wyniki powyżej 9 sekund

zostały wykluczone. Z dużym prawdopodobieństwem w tych przypadkach uczestnicy nie wykonali prawidłowo zadań, nie wiedzieli lub zapomnieli co mają zrobić.

Analizując dane z wykresu, można stwierdzić, że rzadko kiedy średnie czasy przekraczały 5 sekund, zazwyczaj wynosiły one 3-4 sekundy. Są to dobre wyniki i oznaczają, że odnajdywanie elementów na stronie odbywało się szybko i sprawnie. Linia trendu pokazuje również, że czas wykonywania kolejnych zadań się zmniejszał, co oznacza, iż mimo mieszania kolejności poleceń oraz muzeów użytkownicy przyzwyczajali się do interfejsów i zapamiętywali rozmieszczenia ich elementów.

Do przeprowadzenia analizy jakościowej użyto map cieplnych i ścieżek skanowania. Rysunki 5 i 6 przedstawiają mapy cieplne wygenerowane po wykonaniu przez wszystkich uczestników zadania z1 polegającego na znalezieniu ikony – narzędzia do powiększenia obrazu w obu interfejsach muzeów.

Rysunek 5: Mapa cieplna wygenerowana dla Muzeum Zamkowego w Malborku.

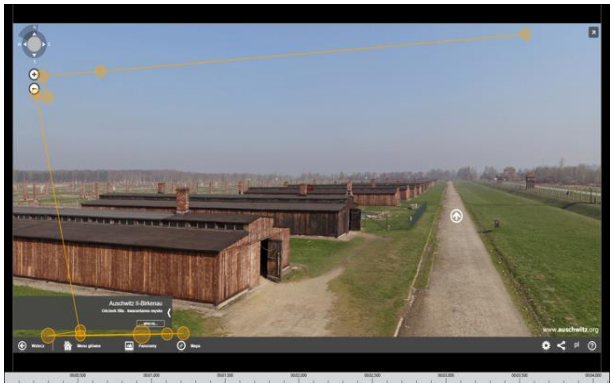


Rysunek 6: Mapa cieplna wygenerowana dla Muzeum Auschwitz-Birkenau.

Na pierwszym zrzucie ekranowym (Rysunek 5), widoczny jest jeden główny gorący, czerwony obszar w centralnej części bodźca i kilka żółto-zielonych. Drugi pod względem intensywności obszar znajdujący się w prawym, dolnym rogu to właściwy cel tego zadania. Wynika to z tego, że było to pierwsze zadanie i dlatego uczestnicy badań zapoznawali się z interfejsem i szukali właściwego narzędzia. Natomiast mapa cieplna przedstawiona na Rysunku 6 jest przykładem pomyślne wykonanego polecenia polegającego na odnalezieniu właściwej ikony. Jej symbol graficzny pokrywa gorący obszar w kolorze czerwonym. Oznacza to, że właśnie w tym miejscu najczęściej skupiał się wzrok użytkowników.

W przypadku ścieżek skanowania, program iMotions generuje wyniki osobno dla każdego użytkownika wykonującego określone zadanie w danym wirtualnym muzeum. Oznacza to, że w sumie analizie zostało poddane 10 obrazów ze ścieżkami wzroku dla każdego uczestnika badania (po 5 dla każdego interfejsu).

Rysunek 7: Przykład ścieżki wzroku uzyskanej na podstawie wirtualnego zwiedzania Muzeum Zamkowego w Malborku.



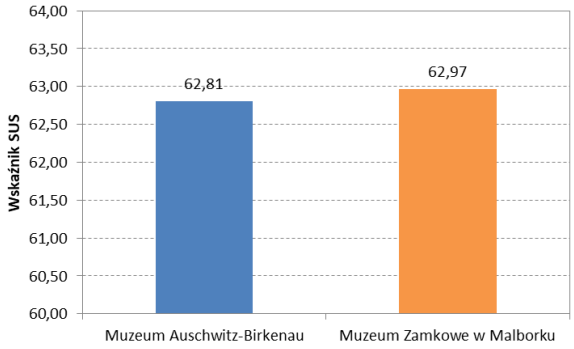
Rysunek 8: Przykład ścieżki wzorku uzyskanej na podstawie wirtualnego zwiedzania Muzeum Auschwitz-Birkenau.

Przykłady obrazów z oznaczonymi ruchami i zatrzymaniami uwagi wzrokowej zostały zaprezentowane na Rysunkach 7 oraz 8. Obraz ze ścieżką wzroku na Rysunku 7 prezentuje przykład, w którym użytkownik wykonując niewielką liczbę ruchów sakadycznych, w wyniku których sprawnie osiągnął cel, odnalazł wzrokiem ikonę umożliwiającej udostępnienie. Przykład ścieżki wzorku dla drugiego muzeum (Rysunek 8) przedstawia niepoprawnie wykonane zadanie. Badany prawdopodobnie nie zrozumiał polecenia lub za szybko przeszedł do kolejnej instrukcji, ponieważ jego wzrok nawet nie zbliżył się do obszaru, w którym znajduje się szukany element.

6.2. Wyniki badania ankietowego

Użytkownicy otrzymali dla każdego muzeum kwestionariusz Skali Użyteczności Systemu. Po wypełnieniu ankiet obliczono wskaźniki SUS dla każdego użytkownika, a następnie je uśredniono. Wyniki SUS dla obu muzeów, przedstawione na Rysunku 9, miały podobne poziomy i wahały się w granicach 62 - 63 punktów. Muzeum Zamkowe w Malborku uzyskało 62,81 punkty,

a Muzeum Auschwitz-Birkenau 62,97 punkty. Oba wyniki sytuują się poniżej 68 punktów – pułap, który wyznacza minimalny próg, od którego systemy określa się jako dobre. Oznacza to, że wyniki z tej części badań znajdują się poniżej progu czyli 68 punktów.



Rysunek 9: Wykres przedstawiający średnie wyniki SUS dla obu muzeów.

Dodatkowymi wynikami uzyskanymi z ankiety opracowanej przez autorów były średnie oceny wyliczone dla każdego z siedmiu pytań podejmujących istotne zagadnienia z punktu widzenia autorów tego artykułu. Ich treści nie odnosiły się do konkretnego interfejsu, ale do ogólnych kwestii związanych z wirtualnymi muzeami. Użytkownicy wyrażali swoje opinie posługując się skalą od 1 do 5, gdzie 1 oznaczało „Zdecydowanie się nie zgadzam", natomiast 5 – „Zdecydowanie się zgadzam". Treści zdań oraz uzyskane średnie oceny zostały przedstawione w Tabeli 5.

Tabela 5: Treści autorskich zagadnień dotyczących wirtualnych muzeów wraz z ich średnimi ocenami

Lp.	Zagadnienie	Średnia ocen
1	Uważam, że wirtualne muzea są bardzo dobrą alternatywą dla tradycyjnych muzeów.	3,25
2	W przyszłości z chęcią będę zwiedzał muzea w formie wirtualnej.	3,25
3	Uważam, że coraz więcej obiektów muzealnych powinno być dostępnych w formie wirtualnej.	4,16
4	Uważam, że wirtualne muzea dobrze oddają klimat wizyty w tradycyjnym muzeum.	2,25
5	Uważam, że łatwa przystępność to jedyna zaleta wirtualnych muzeów.	2,81
6	Myślę, że wirtualne muzea powinny wyróżniać się większą liczbą interaktywnych elementów.	4,25
7	Uważam, że spacer po wirtualnym muzeum dostarcza wielu informacji na temat oglądanych obiektów.	4,19

Ostatnia część badania ankietowego składała się z dwóch pytań otwartych, które miały na celu odnalezienie mocnych i słabych stron dotyczących wirtualnych muzeów. Grupa badawcza, która podczas badania eye-trackingowego zapoznała się z wybranymi interfejsami muzeów, wskazała mocne strony wirtualnych muzeów, do których należą:

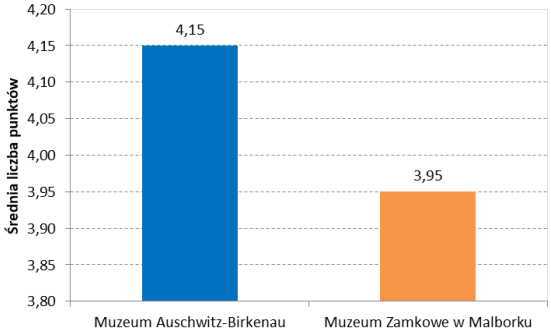
- możliwość zwiedzania niezależne od czasu (wirtualne muzea są „otwarte" całą dobę) i miejsca;
- dostępność czyli możliwość zwiedzenia muzeów znajdujących się nawet bardzo daleko;
- oszczędność czasu i pieniędzy, które tracone są w przypadku podróży do tradycyjnego muzeum;
- brak ograniczeń ze względu na pogodę;
- brak tłumów, które są częste w przypadku popularnych ekspozycji.

Z kolei do słabych stron wirtualnych muzeów zaliczają się:

- wrażenie, iż wirtualne muzeum przypomina grę komputerową;
- sztuczność i ograniczenie odczuwania muzealnej atmosfery, która wpływa na odbiór oglądanych obiektów;
- za duże różnice (np. rozmiar, kształt) między wirtualnymi obiektami, a rzeczywistymi.

6.3. Wyniki analizy heurystycznej

Informacje uzyskane z przeprowadzenia analizy heurystycznej interfejsów Muzeum Auschwitz-Birkenau oraz Muzeum Zamkowego w Malborku pomagają zrozumieć, w jakim stopniu dana witryna spełnia podstawowe zasady użyteczności.



Rysunek 10: Średnie oceny interfejsów muzeów według heurystyk Nielsena.

Zgromadzone dane są w postaci ocen każdej z heurystyk Nielsena i wystawione zostały przez dwie osoby mające wiedzę i doświadczenie na temat projektowania interfejsów oraz ich ewaluacji. Większość ocen miała wartości 4 lub 5, co przekłada się na uzyskanie dobrych wyników końcowych dla analizowanych serwisów wirtualnych muzeów. Wyniki te zostały uzyskane poprzez zsumowanie ocen dla danego muzeum, a następnie wyliczenie ich średnich. Wykres przedstawiający uzyskane rezultaty z analizy heurystycznej został przedstawiony na Rysunku 10.

7. Podsumowanie

Wprowadzone w czasie pandemii COVID-19 obostrzenia i zalecenia wpłynęły w znaczny sposób na sektor muzealny, który zmuszony był do zamknięcia, zawieszenia lub ograniczenia działalności większości placówek, co powodowało liczne trudności i straty finansowe. W odpowiedzi, wiele obiektów zdecydowało się na przeniesienie swojej działalności do świata cyfrowego,

w którym wirtualne muzea swoją atrakcyjnością i dostępnością zachęcają do eksplorowania ich zasobów.

W artykule podkreślono rolę eyetrackingu jako narzędzia wspierającego ocenę interfejsów i analizę doświadczeń użytkowników. Wykorzystano również kwestionariusz SUS oraz własną ankietę, które pozwoliły na zebranie ogólnych informacji odzwierciedlających opinie użytkowników na temat wirtualnych muzeów. Zastosowano również analizę ekspercką opierającą się na heurystykach Nielsena.

W efekcie przeprowadzonych badań możliwe było porównanie obu interfejsów muzeów pod kątem łatwości ich obsługi, szybkości odnajdywania elementów i szeroko pojętego doświadczenia ich użytkowników. Wyniki uzyskane z eksperymentu eyetrackingowego wskazują na to, iż zadania sprawniej wykonywane były w przypadku Muzeum Auschwitz-Birkenau. Analiza jakościowa uzyskanych map cieplnych i ścieżek wzroku przyniosła podobne rezultaty. Podczas realizacji zadań dla obu muzeów zdarzały się błędnie wykonywane polecenia.

Badanie oparte na kwestionariuszu SUS pozwoliło na stwierdzenie, iż mimo podobnych wyników wskaźnika SUS nieznacznie gorzej wypadło Muzeum Zamkowe w Malborku. W obu przypadkach poziomy tego wskaźnika były niezadowalające.

Z przeprowadzonej autorskiej ankiety wynika, że ankietowani widzą potrzebę wirtualizowania muzeów oraz że ta forma zwiedzania dostarcza wielu informacji na temat oglądanych obiektów. Poza tym respondenci zwracają uwagę na zwiększenie interaktywności wirtualnych wystaw. Wnioski te potwierdzają prawdziwość hipotezy H1. Z kolei respondenci zauważają jednocześnie, że wizyta w wirtualnym muzeum nie oddaje klimatu jaki panuje w tradycyjnym muzeum. Poza tym cyfrowa forma muzeów, stwarza atmosferę sztuczności, a obiekty wirtualne wyglądają inaczej niż obiekty rzeczywiste.

Mimo dobrych ocen i bardzo zbliżonych wyników, eksperci przeprowadzający badanie heurystyczne także lepiej ocenili interfejs Muzeum Auschwitz-Birkenau. W związku z tym z przeprowadzonych badań wynika, iż w porównaniu obu muzeów nieznacznie lepiej wypada wirtualne Muzeum Auschwitz-Birkenau niż Muzeum Zamkowe w Malborku. Pomimo tego, spoglądając na wszystkie uzyskane wyniki – te lepsze i te gorsze - można stwierdzić, że istnieje konieczność udoskonalania interfejsów wirtualnych muzeów w celu osiągnięcia wysokiego poziomu doświadczenia użytkownika. Ta konstatacja potwierdza słuszność hipotezy H2.

Literatura

[1] K. Walczak, W. Cellary, M. White, Virtual Museum Exhibitions, Computer 39(3) (2006) 93-95, https://doi.org/10.1109/MC.2006.108.

[2] Augmented Representation of Cultural Objects (ARCO), https://www.arco-web.org/index.html, [20.09.2023].

[3] C. Goulding, The museum environment and the visitor experience, European Journal of Marketing 34(3/4) (2000) 261-278, https://doi.org/10.1108/03090560010311849.

[4] P. P. P. Morantes, S. A. Peñarete, G. Arbelaez, M. Camargo, L. Dupont, Understanding Museum visitors' experience through an Eye-tracking study and a Living Lab approach, 2016 International Conference on Engineering, Technology and Innovation/IEEE International Technology Management Conference (ICE/ITMC), Trondheim, Norway (2016) 1-6, https://10.1109/ICE/ITMC39735.2016.9025900.

[5] Y-L Hsieh, M-F. Chen, W-J. Wang, Application of visitor eye movement information to museum exhibit analysis, Sustainability 14(11) (2022) 6932, https://doi.org/10.3390/su14116932.

[6] P. Majaranta, A. Bulling, Eye tracking and eye-based human–computer interaction, Advances in physiological computing (2014) 39-65, https://doi.org/10.1007/978-1-4471-6392-3_3.

[7] J. Młodkowski, Koncepcja uwagi wizualnej, Acta Universitatis Lodziensis. Folia Psychologica 12 (2008) 23-44.

[8] A. Stolińska, M. Andrzejewska, Metodologiczne aspekty stosowania techniki eye trackingowej w badaniach edukacyjnych, Przegląd Badań Edukacyjnych (Educational Studies Review) 1(24) (2017) 259-276, http://dx.doi.org/10.12775/PBE.2017.015.

[9] Ł. Rosicki, Zastosowanie heurystyk Jakoba Nielsena w optymalizacji serwisów internetowych na przykładzie witryny Uniwersytetu Ekonomicznego w Krakowie, Knowledge-Economy-Society (2018) 185-192.

[10] K. Nowak, D. Samolej, Analiza wybranych metod oceny użyteczności w procesie tworzenia aplikacji internetowych, Journal of Computer Sciences Institute 14 (2020) 88-93, https://doi.org/10.35784/jcsi.1582.

[11] J.R. Lewis, J. Sauro, Item benchmarks for the system usability scale, Journal of Usability Studies 13(3) (2018) 158-167.

[12] Skala Użyteczności Systemu, https://www.surveylab.com/pl/blog/skala-uzytecznosci-systemu-sus/, [20.09.2023].

[13] Measuring Perceived Usability with the SUS, NASA-TLX, and the Single Ease Question After Tasks and Usability Tests, https://www.nngroup.com/articles/measuring-perceived-usability/, [20.09.2023].

[14] GP3 HD Eye Tracker, https://www.gazept.com/product/gp3hd/, [20.09.2023].

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 30 (2024) 39–46
Received: 3 October 2023
Accepted: 9 October 2023

Analysis of user experience during interaction with automotive repair workshop websites

Analiza doświadczenia użytkownika podczas interakcji z serwisami internetowymi warsztatów samochodowych

Radosław Danielkiewicz*, Mariusz Dzieńkowski

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

Abstract
The study presented here focuses on analyzing and comparing the user experience of two automotive repair workshop website search engines. The research tools used were usability tests, user interviews and eye tracking. These were aimed at assessing the functionality, intuitiveness and overall user experience when using the analyzed sites. The results show that users encountered a variety of problems with navigation and search functions, and made suggestions for improving and expanding the website functions. Nevertheless, despite some differences in the evaluations of specific aspects of both services, there was little difference in the effectiveness of the set scenarios.

Keywords: user experience; usability testing; user interview; eye tracking

Streszczenie
Przedstawione badanie koncentruje się na analizie i porównaniu użytkowania dwóch wyszukiwarek serwisów warsztatowych. Wykorzystane narzędzia badawcze to testy użyteczności, wywiady z użytkownikami oraz eyetracking. Miały one na celu ocenę funkcjonalności, intuicyjności i ogólnego doświadczenia użytkowników podczas korzystania z analizowanych serwisów. Wyniki pokazują, że użytkownicy napotykali różnorodne problemy z nawigacją czy funkcjami wyszukiwania, a także zgłaszali sugestie dotyczące poprawy i rozbudowy funkcji serwisów. Niemniej jednak, pomimo pewnych różnic w ocenach poszczególnych aspektów obu serwisów, różnice w skuteczności realizacji zadanych scenariuszy były niewielkie.

Słowa kluczowe: doświadczenia użytkowników; testy użyteczności; wywiad z użytkownikiem; eyetracking

*Corresponding author
Email address: **radoslaw.danielkiewicz@pollub.edu.pl** (R. Danielkiewicz)

1. Wstęp

Analiza doświadczenia użytkownika jest kluczowym obszarem badań w dziedzinie projektowania interakcji człowiek-komputer. Obejmuje ona badanie interakcji użytkowników z różnego rodzaju systemami, aplikacjami czy serwisami w celu zrozumienia ich oczekiwań, potrzeb i satysfakcji. Analiza doświadczenia jest związana z zapewnieniem użytkownikom jak najlepszego interfejsu oraz optymalizacją procesów, co przekłada się na zwiększenie efektywności, satysfakcji i lojalności użytkowników.

W kontekście warsztatów samochodowych, serwisy internetowe odgrywają kluczową rolę w zapewnianiu klientom łatwego dostępu do informacji oraz realizacji usług. Przez platformy internetowe użytkownicy mogą składać zlecenia napraw, umawiać się na wizyty, sprawdzać statusy napraw czy też otrzymywać informacje o cenach usług. Serwisy te są projektowane w taki sposób, aby ułatwić interakcję między klientami a warsztatami samochodowymi, zwiększając tym samym efektywność procesu naprawczego.

Oczekiwania klientów wobec interfejsów stron internetowych firm warsztatowych stale rosną. Klienci oczekują intuicyjnego i przyjaznego dla użytkownika projektu, łatwej nawigacji, zrozumiałych formularzy oraz szybkiego dostępu do potrzebnych informacji. W odpowiedzi na te wymagania, warsztaty samochodowe muszą skupiać się na ciągłym doskonaleniu swoich serwisów internetowych, aby zapewnić użytkownikom jak najwyższy poziom satysfakcji.

W niniejszej pracy skupiono się na analizie doświadczenia użytkownika podczas interakcji z serwisami internetowymi warsztatów samochodowych. Celem jest zrozumienie, jak użytkownicy odbierają interfejsy dwóch serwisów oraz ocena intuicyjności obu serwisów. Przeprowadzone zostało badanie z wykorzystaniem dwóch głównych metod: testów użyteczności (ang. Usability Testing) oraz wywiadów z użytkownikami (ang. User Interviews). Testy użyteczności pozwolą ocenić efektywność i intuicyjność interfejsów, podczas gdy wywiady z użytkownikami dostarczą cennych informacji zwrotnych odnośnie ich doświadczeń, oczekiwań i sugestii.

Przy użyciu tych metod analizy można zidentyfikować mocne strony serwisów internetowych warsztatów samochodowych oraz obszary, które wymagają poprawy. Wyniki tych badań staną się cennym źródłem informacji dla firm tworzących aplikacje internetowe.

2. Przegląd literatury

W książce [1] autorstwa Jakoba Nielsena podjęto się zadania zbadania i opisania fundamentalnych aspektów

użyteczności witryn internetowych. Celem przeprowadzonej analizy była dogłębna eksploracja czynników decydujących o intuicyjności i przyjazności użytkownika danej strony. Wykorzystując różnorodne metody badawcze, w tym obserwacje bezpośrednie, ankiety i testy z użytkownikami, Nielsen szczegółowo zbadał setki różnorodnych interfejsów internetowych, dążąc do identyfikacji uniwersalnych zasad ich funkcjonowania. Kluczowym wnioskiem, który jest filarem całej pracy, jest postawienie na pierwszym miejscu prostoty w projektowaniu. Autor przekonuje, że w dobie nadmiaru informacji i złożoności technologicznych, użytkownicy poszukują interfejsów charakteryzujących się przejrzystością, intuicyjnością i brakiem niepotrzebnych elementów. Takie stanowisko wynika z wielu badań, które dowodzą, że witryny o przejrzystej i nieskomplikowanej strukturze cieszą się większą popularnością i lepszą oceną wśród użytkowników. W zakończeniu pracy Nielsen zaznacza, że regularne badania użyteczności powinny stanowić nieodłączny element każdego procesu projektowania witryn internetowych. Tylko dzięki nieustannemu dostosowywaniu się do ewoluujących potrzeb i preferencji użytkowników, możliwe jest tworzenie stron, które są nie tylko estetycznie atrakcyjne, ale przede wszystkim funkcjonalne i użyteczne.

W kolejnej pozycji książkowej [2] autorstwa Steve'a Kruga główny nacisk położony jest na analizę fundamentalnych aspektów użyteczności interfejsów internetowych z punktu widzenia zwykłego użytkownika. Autor, łącząc anegdoty z własnej praktyki zawodowej z gruntownymi badaniami, zastosował metodę "testów użyteczności na śniadanie", ilustrując w jaki sposób szybkie sesje oceny mogą efektywnie identyfikować problemy projektowe. Krug opiera swój sposób projektowania na przekonaniu, iż najbardziej efektywne strony i aplikacje to takie, które są intuicyjne dla użytkownika, eliminując potrzebę głębokich rozważań nad interakcją. Autor demonstruje, że w praktyce wielu użytkowników przegląda strony internetowe w sposób powierzchowny, a nie dogłębny, skanując treści w poszukiwaniu potrzebnych informacji. Podsumowując, "Don't Make Me Think" to zbiór cennych uwag dotyczących projektowania użytecznych interfejsów internetowych. Krug podkreśla, że zrozumienie ludzkich zachowań w cyfrowym środowisku interakcji jest kluczowe dla tworzenia stron i aplikacji, które w pełni zaspokajają potrzeby użytkowników.

W książce [3] czytelnik ma możliwość zanurzenia się w tematykę testowania użyteczności oraz metodologię oceny doświadczenia użytkownika (UX). Twórcy tej publikacji skupiają się na niezbędnych procesach i praktykach, które są kluczowe dla skutecznego testowania interfejsów, począwszy od fazy planowania, poprzez kreowanie testów, aż do analizy uzyskanych wyników. Rubin i Chisnell wnikliwie wyjaśniają, jak realizować testy użyteczności, jak dobierać adekwatnych uczestników, jak również jak analizować i interpretować dane, by doskonalić projekty. Jednym z istotnych elementów tej publikacji jest etyczne podejście do procesu testowania, co ma kluczowe znaczenie w kontakcie z uczestnikami. Przedstawiona pozycja oferuje całe spectrum narzędzi i technik niezbędnych do oceny i usprawnienia interfejsu użytkownika. Poprzez zgłębianie kompleksowych procesów testowania użyteczności, książka umożliwia zrozumienie reakcji użytkowników na różnorodne elementy interfejsu, potencjalnych problemów, z jakimi mogą się oni zetknąć oraz sposobów ich rozwiązania, aby zapewnić optymalne doświadczenie użytkownika.

W publikacji [4], autorstwa Jakoba Nielsena, zaprezentowano dogłębną analizę fundamentalnych technik i metod, które znajdują zastosowanie w dziedzinie inżynierii użyteczności. Nielsen detalicznie przybliża procesy, takie jak testy użyteczności, akcentując istotność obserwacji autentycznych użytkowników oraz metody ewaluacji ekspertów, które koncentrują się na ocenie interfejsów w świetle ustalonych zasad projektowania. Autor zwraca uwagę na analizę zadaniową, koncentrując się na rozumieniu i doskonaleniu zadań realizowanych przez użytkowników. W książce znaleźć można również praktyczne porady dotyczące prototypowania, zarówno w wersji papierowej, jak i zaawansowanej cyfrowej, a także omówienie roli scenariuszy użytkowania w kreowaniu interaktywnych doświadczeń. Nielsen podkreśla również znaczenie przestrzegania wytycznych projektowych i metryk użyteczności, służących do kwantyfikacji i oceny efektywności projektów pod kątem użyteczności. Reasumując całość, autor akcentuje niezbędność zrozumienia i uwzględnienia potrzeb użytkowników jako kluczowego elementu każdego przedsięwzięcia projektowego. W rezultacie, "Usability Engineering" stanowi nie tylko teoretyczne kompendium wiedzy, ale przede wszystkim praktyczne narzędzie dla specjalistów dążących do tworzenia produktów przyjaznych użytkownikowi.

W publikacji [5] autor podejmuje się analizę kwestii projektowania formularzy online oraz oddziaływania ich użyteczności na doświadczenia korzystających z nich osób. Przedstawione są strategie, metody oraz sprawdzone praktyki w dziedzinie kreowania formularzy internetowych, mające na celu zwiększenie ich użyteczności i generowanie pozytywnych doświadczeń u użytkowników. Głównym zamierzeniem książki jest zbadanie, w jaki sposób adekwatne projektowanie formularzy internetowych może wpływać na zadowolenie użytkowników, stopień konwersji oraz efektywność serwisów internetowych. Podstawą analizy autora są przegląd literatury i badanie rzeczywistych przypadków formularzy, gdzie ukazane są zarówno pozytywne, jak i negatywne przykłady projektowania formularzy online. Dodatkowo autor prezentuje różnorodne techniki i rozwiązania mogące przyczynić się do zwiększenia użyteczności formularzy. Kluczowym odkryciem jest wyodrębnienie głównych elementów mających wpływ na użyteczność formularzy online, takich jak upraszczanie procesu wypełniania, dostosowywanie formularzy do wymagań użytkowników oraz implementacja właściwych mechanizmów kontrolnych i walidacji danych. Adekwatne projektowanie formularzy online znacząco oddziałuje na doświadczenie użytkowników, co z kolei ma wpływ na powodzenie serwisów internetowych.

Zastosowanie rekomendowanych przez autora praktyk i metod może skutkować wzrostem zadowolenia użytkowników, poprawą konwersji oraz ogólnym podniesieniem użyteczności witryn. Projektanci stron internetowych powinni więc przywiązywać szczególną wagę do jakości formularzy, aby zagwarantować użytkownikom pozytywne doświadczenia.

3. Wykorzystane metody badawcze

Testy użyteczności, będące kluczowym elementem badania, miały na celu ocenę i zrozumienie, jak użytkownicy pracują z serwisami oraz jakie trudności mogą napotkać, korzystając z funkcjonalności rezerwacji online. Przez symulację realnych sytuacji użytkowania, testy te pozwoliły na zgłębienie i analizę doświadczeń użytkownika, co umożliwiło identyfikację potencjalnych obszarów do poprawy. Dodatkowo, przeprowadzone wywiady dostarczyły cennych informacji odnośnie percepcji i oczekiwań użytkowników w kontekście korzystania z omawianych serwisów. Wywiady te były nieocenionym źródłem kwalitatywnych danych, umożliwiających głębsze zrozumienie motywacji, potrzeb i problemów użytkowników. Razem, testy użyteczności i wywiady pozwoliły na holistyczne zrozumienie użyteczności serwisów, co miało bezpośredni wpływ na formułowanie wniosków i rekomendacji.

3.1. Testy użyteczności

Testy użyteczności stanowią kluczowy element w procesie projektowania interfejsów użytkownika. Definiując, testy użyteczności to zorganizowany proces, w którym użytkownicy końcowi korzystają z produktu lub usługi w celu zidentyfikowania obszarów do poprawy i optymalizacji. Głównym celem jest ocena, w jaki sposób rzeczywiste osoby używają systemu, a wyniki tych badań informują projektantów o możliwych usprawnieniach.

Testy użyteczności charakteryzują się zbieraniem danych w sposób bezpośredni, poprzez obserwację użytkowników i analizę ich interakcji z systemem. Są one przeprowadzane w kontrolowanych warunkach, zazwyczaj z użyciem scenariuszy zadań, które odwzorowują typowe działania i procesy. Umożliwia to identyfikację problemów użyteczności, takich jak trudności w nawigacji, niejasności w komunikacji czy błędy interakcji.

Zastosowanie testów użyteczności jest bardzo szerokie i obejmuje różne etapy projektowania. Mogą być one używane w fazie wstępnej, aby zidentyfikować potrzeby i oczekiwania użytkowników, w trakcie projektowania, aby iteracyjnie poprawiać prototypy, oraz po wdrożeniu, aby ocenić użyteczność gotowego produktu.

3.2. Wywiad z użytkownikiem

Wywiady z użytkownikami to kolejna technika badawcza w dziedzinie projektowania interaktywnego i doświadczenia użytkownika. Definiując, wywiady z użytkownikami polegają na bezpośrednich rozmowach z użytkownikami końcowymi, mających na celu zgłębienie ich potrzeb, oczekiwań, doświadczeń i motywacji.

Podczas przeprowadzania wywiadów z użytkownikami, badacz korzysta z zestawu zaplanowanych pytań, ale rozmowa jest na tyle elastyczna, że może się rozwijać w zależności od odpowiedzi uczestnika. To pozwala na głębokie zrozumienie perspektywy użytkownika i odkrycie aspektów, których nie można by zidentyfikować za pomocą bardziej skwantyfikowanych metod, takich jak ankiety.

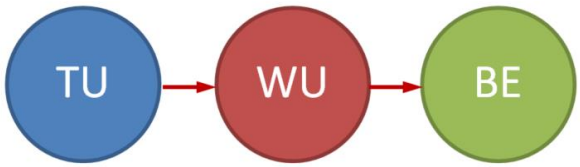
Wywiady z użytkownikami można przeprowadzać w różnych etapach procesu projektowego. Na wczesnych etapach mogą one pomóc zrozumieć kontekst użytkowania, cele i potrzeby użytkowników. W fazie projektowania i prototypowania, wywiady mogą dostarczyć wartościowej informacji zwrotnej i wglądu w interakcje użytkowników z produktem.

3.3. Eyetracking

Eyetracking to technika wykorzystywana m.in. w badaniu ergonomii i jakości interfejsów komputerowych. Umożliwia ona pomiar, rejestrację i analizę aktywności ocznej badanych osób. Uzyskane dane można poddać analizie zarówno ilościowej jak i jakościowej. W ramach tej pracy zrealizowano analizy jakościowe na podstawie ścieżek skanowania, obrazujących ruchy i zatrzymania uwagi na wyświetlanym bodźcu, w tym przypadku będącym serwisem internetowym.

4. Plan badania

Celem badania było zgromadzenie danych na temat korzystania z serwisów internetowych Warsztaty Yanosik oraz Motointegrator, oferujących identyczną funkcjonalność rezerwacji wizyt online w warsztatach samochodowych. Ta specyficzna funkcja stanowiła główny obiekt badań. Oba serwisy zastosowały różne podejścia do kwestii rezerwacji terminu naprawy. Użytkownicy przeszli przez scenariusze zadań, dostosowane do funkcjonalności rezerwacji wizyt online zarówno w serwisie Warsztaty Yanosik jak i Motointegratorze, odwzorowujący realną sytuację przygotowania samochodu do sezonu zimowego. Kolejne etapy przeprowadzonych badań przedstawia schemat na Rysunku 1.



Rysunek 1: Plan badania (TU – test użyteczności, WU - wywiad z użytkownikiem, BE – badanie eyetrackingowe).

4.1. Scenariusze testów użyteczności

W ramach badania doświadczeń użytkownika na platformach Warsztaty Yanosik oraz Motointegrator opracowano specyficzne scenariusze zadań. Celem tych scenariuszy jest zasymulowanie realnych sytuacji, w których użytkownicy mogą korzystać z serwisów, aby dokładniej zbadać ich użyteczność i interaktywność.

Symulowane zadanie jest oparte na wymianie opon. Scenariusz zadania dla obu serwisów jest analogiczny. Dla serwisu Warsztaty Yanosik wygląda on następująco:

Chcesz przygotować samochód do zbliżającego się sezonu zimowego i musisz wymienić w nim opony na zimowe. Skorzystaj z serwisu internetowego Warsztaty Yanosik w celu umówienia wizyty online w najlepszym (na podstawie opinii innych użytkowników) warsztacie w Lublinie lub okolicy. Warsztat musi specjalizować się w konkretnej marce samochodu oraz wykonywać usługę wymiany opon. Do ograniczenia listy wyników warsztatów wykorzystaj wyszukiwanie zaawansowane. Następnie wybierz warsztat i zapoznaj się z jego ofertą usług i wykonaj symulację rezerwacji wizyty.

Szczegółowy przebieg badania dla obu platform prezentuje się następująco:

Serwis Warsztaty Yanosik:

- Wpisanie lokalizacji Lublin w wyszukiwarkę warsztatów.
- Użycie filtra określającego markę serwisowanego samochodu.
- Użycie filtra usług warsztatowych aby zawęzić wyniki.
- Wybór najlepszego warsztatu na podstawie ocen i opinii użytkowników. Posortowanie warsztatów wg ocen malejąco.
- Rozpoczęcie procesu rezerwacji w wybranym warsztacie. Kliknięcie w przycisk "Umów wizytę".
- Wypełnienie formularza zgłoszeniowego: wybór marki i modelu pojazdu, podanie danych kontaktowych oraz opisu usterki.

Wykorzystanie powyższych zadań elementarnych pozwoliło na dokładną analizę oraz zdobywanie istotnych informacji dotyczących użyteczności omawianych serwisów, stanowiąc kluczowy element w procesie badawczym.

4.2. Wywiad z użytkownikiem - pytania

Wywiady zostały przeprowadzone w formie indywidualnych rozmów z użytkownikami, bezpośrednio po wykonaniu przez nich scenariuszy dla obu serwisów. Wywiady przeprowadzano w spokojnym otoczeniu, umożliwiając uczestnikom swobodne dzielenie się swoimi myślami i opiniami. Wszystkie odpowiedzi były zapisywane i później poddawane analizie, aby wyciągnąć z nich kluczowe wnioski i obserwacje.

Poniżej przedstawiono zestaw pytań, które zostały sformułowane w celu dogłębnego zrozumienia doświadczeń użytkowników po interakcji z oboma serwisami. Te konkretne kwestie skupiają się na użytkowalności, intuicyjności oraz ogólnych wrażeniach korzystania z serwisów:

1. Czy którykolwiek z serwisów wyróżnił się pozytywnie lub negatywnie?
2. Czy którakolwiek ze stron była trudniejsza w nawigacji? Jeśli tak, które elementy sprawiły Ci trudność?
3. Czy na którejkolwiek ze stron napotkałeś na elementy, które były dla Ciebie niejasne lub mylące?
4. Jakie były Twoje wrażenia dotyczące narzędzi wyszukiwania w obu serwisach? Czy któreś z nich było bardziej skuteczne niż drugie?
5. Czy w trakcie zgłaszania naprawy na którejkolwiek ze stron napotkałeś na trudności? Jeśli tak, jakie?
6. Jakie były Twoje wrażenia dotyczące formularza zgłoszeniowego w obu serwisach?
7. Jakie są Twoje wrażenia dotyczące narzędzi do filtrowania wyników w obu serwisach?
8. Czy któreś z narzędzi do filtrowania było bardziej intuicyjne lub skuteczne niż drugie?
9. Czy masz jakieś dodatkowe uwagi lub sugestie dotyczące tego, jak możemy poprawić każdy z serwisów?
10. Gdybyś miał wybrać jeden serwis do regularnego korzystania w przyszłości, który by to był i dlaczego?

Zestaw tych pytań miał na celu zidentyfikowanie mocnych i słabych stron obu serwisów z perspektywy użytkownika. Opinie i odpowiedzi uzyskane w trakcie wywiadów pomogą w dalszym doskonaleniu i optymalizacji funkcjonalności serwisów, tak aby jak najlepiej spełniały one potrzeby i oczekiwania użytkowników.

4.3. Stanowisko badawcze

Część eyetrackingowa badań została przeprowadzona w laboratorium Katedry Informatyki Politechniki Lubelskiej, wyposażonym w eyetracker podłączony do laptopa z dostępem do Internetu oraz z zainstalowanym specjalistycznym oprogramowaniem (Rysunek 2).

Model laptopa to Acer Nitro 5 wyposażony w procesor firmy AMD Ryzen 7 5800H z dedykowaną kartą graficzną marki NVIDIA serii GeForce RTX 3060. Urządzenie posiada 32 GB pamięci RAM i ma zainstalowany system operacyjny Windows 10. Ekran laptopa posiada przekątną o długości 15,6 cala i rozdzielczość Full HD z częstotliwością odświeżania 144 Hz.

Rysunek 2: Stanowisko badawcze: laptop Acer Nitro 5 oraz eyetracker Gazepoint GP3 HD [6].

Do przeprowadzenia badania użyto eyetrackera Gazepoint GP3 HD. Wymiary tego urządzenia to 23,5 x 4,5 x 4,7 cm, a jego waga - 125g. Połączenie eyetrackera z komputerem realizowane było za pomocą interfejsu USB, a przetwarzanie danych odbywało się na komputerze PC. Urządzenie to charakteryzuje się częstotliwością próbkowania wynoszącą 60Hz lub 150Hz. Dokładność urządzenia mieści się w zakresie od 0,5 do 1° [6]. Okulograf został ustawiony przy ekranie laptopa. Po-

zwala on na ruchy głowy w zakresie 35 x 22 cm. Oprócz rejestracji sakad – ruchów oczu i fiksacji – zatrzymań trwających dłużej niż 80 ms, eyetracker oferuje możliwość pomiaru zmian średnicy źrenic.

Eyetracker działał pod kontrolą platformy iMotions w wersji 9.1. Dzięki tej aplikacji możliwe było zaprojektowanie eksperymentu, jego przeprowadzenie oraz gromadzenie, analiza i prezentacja wyników [7].

4.4. Grupa badawcza

Grupa badawcza składała się z 20 osób o zróżnicowanym wieku, z najliczniejszą grupą w wieku 22-27 lat (55%). 90% uczestników posiadało własny samochód. Wszyscy respondenci nie mieli doświadczenia z rezerwacją wizyt online w warsztatach samochodowych. Wiedza uczestników z dziedziny mechaniki samochodowej była zróżnicowana: 45% badanych miało podstawową wiedzę, 35% średnią, a 10% było ekspertami. W kwestii preferencji formy rezerwacji, 45% nie miało wyraźnej preferencji, 25% preferowało rezerwację online, a 30% nie preferowało rezerwacji przez Internet. Te informacje są kluczowe dla pełnego zrozumienia i interpretacji wyników badania, mając na uwadze profil, doświadczenia i preferencje respondentów.

4.5. Obiekty badań

W erze cyfrowej, zrozumienie interakcji użytkowników z serwisami internetowymi stało się kluczowe dla sukcesu firm online, w tym warsztatów samochodowych. W niniejszym podrozdziale skoncentrowano się na przedstawieniu dwóch serwisów internetowych warsztatów samochodowych - Yanosik i Motointegrator. Poniżej zaprezentowano dokładne opisy każdego z badanych serwisów.

Serwis internetowy Warsztaty Yanosik [8] to rozszerzenie marki Yanosik, znanego z aplikacji do monitorowania ruchu drogowego. Serwis oferuje bogatą gamę usług związanych z konserwacją i naprawą pojazdów. Użytkownicy mają możliwość przeszukiwania warsztatów według lokalizacji, rodzaju usług oraz opinii innych kierowców, a także rezerwacji wizyty online.

Strona główna serwisu charakteryzuje się przejrzystością i czytelnością. Wyróżnione są trzy podstawowe opcje dla użytkowników ("Informacje", "Blog", "Kontakt") oraz trzy dla warsztatów ("Zaloguj się", "Wyróżnij warsztat", "Zarejestruj warsztat"). Strona oferuje także zaawansowane funkcje wyszukiwania, pozwalające użytkownikom na dokładne sprecyzowanie swoich potrzeb, w tym wybór marki samochodu i specyficznych usług.

Wyniki wyszukiwania prezentowane są w sposób unikalny, z listą warsztatów po lewej stronie i mapą ich lokalizacji po prawej. Interakcja z mapą jest bezpośrednio połączona z listą warsztatów, ułatwiając użytkownikowi orientację. Profil każdego warsztatu jest bogaty w informacje i zawiera sekcję z opiniami klientów, co pomaga w ocenie wiarygodności i jakości usług.

System rezerwacji wizyt został zaprojektowany z myślą o ułatwieniu procesu dla kierowców, umożliwiając rezerwację bez konieczności logowania się do serwisu. Taka struktura formularza rezerwacji gwarantuje, że warsztat otrzyma pełne informacje na temat oczekiwanej wizyty. Ogólnie rzecz biorąc, Warsztaty Yanosik to nowoczesna platforma, której intuicyjność, funkcjonalność i transparentność sprawiają, że cieszy się ona dużą popularnością wśród użytkowników.

Motointegrator [9] jest nowoczesną platformą online, której celem jest integracja świata motoryzacji z nowoczesnymi technologiami. Głównym zadaniem serwisu jest ułatwienie kierowcom znalezienia odpowiedniego warsztatu samochodowego i dostarczenie istotnych informacji z zakresu motoryzacji. Platforma ta oferuje szereg funkcji, takich jak zaawansowana wyszukiwarka warsztatów, spersonalizowane treści dla kierowców i właścicieli warsztatów oraz możliwość umawiania wizyt online.

Strona główna serwisu Motointegrator prezentuje pięć głównych zakładek: "Warsztaty", "Promocja", "Poradniki motoryzacyjne", "Dla warsztatów" oraz "Konto kierowcy", odpowiadając na różnorodne potrzeby użytkowników. Wyszukiwanie zaawansowane umożliwia dopasowanie warsztatu do indywidualnych potrzeb, a kategoria "Popularne" prezentuje najczęściej wybierane opcje lub warsztaty o najwyższych ocenach. Użytkownik może również wybierać warsztaty na podstawie dostępności, obsługiwanych pojazdów, udostępnianych udogodnień oraz przynależności do sieci serwisowej.

Prezentacja wyników wyszukiwania jest jednokolumnowa, zawierając kluczowe informacje o każdym warsztacie, takie jak zdjęcie, ocena, dane kontaktowe, krótki opis historii oraz ewentualne logo sieci serwisowej. Rezultaty wyszukiwania obejmują również reklamy, które są strategicznie rozmieszczone i zaprojektowane w sposób spójny z innymi wynikami. Dodatkowo, wyniki można wyświetlać na mapie, co ułatwia zlokalizowanie warsztatów w danym regionie, a punkty grupowe poprawiają czytelność dla obszarów o dużej liczbie warsztatów.

Platforma Motointegrator umożliwia użytkownikom intuicyjne wyszukiwanie i rezerwowanie usług warsztatów samochodowych. Na stronie głównej zaprezentowane są różnorodne kryteria wyszukiwania, takie jak popularność, lokalizacja czy specjalizacja warsztatu. Szczegółowy widok warsztatu oferuje obszerną prezentację, obejmującą zdjęcia, oceny, cennik, opisy usług oraz opinie klientów, umożliwiając użytkownikowi pełne zorientowanie się w ofercie i wiarygodności warsztatu.

Proces rezerwacji został podzielony na cztery etapy, w których użytkownik określa zakres naprawy, podaje informacje o pojeździe, wybiera termin oraz dostarcza dane kontaktowe. Formularz zgłoszeniowy jest dostosowany zarówno dla użytkowników z wiedzą techniczną, jak i dla tych, którzy nie są pewni charakteru usterki.

Motointegrator charakteryzuje się zaawansowanym systemem wyszukiwania i intuicyjnym procesem rezerwacji, ułatwiając komunikację między kierowcami a warsztatami i zapewniając wysoki standard obsługi klienta. Platforma jest dedykowana zarówno dla właści-

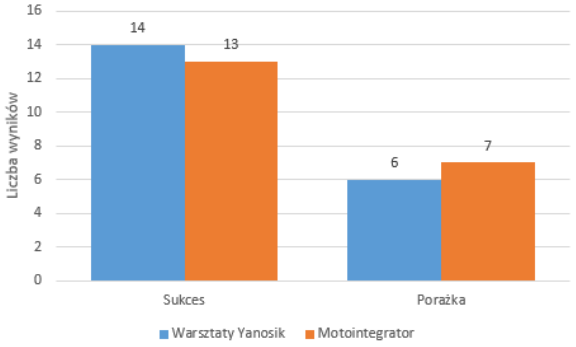
cieli pojazdów, jak i warsztatów samochodowych, oferując pełen zakres informacji i funkcjonalności niezbędnych do realizacji usług.

5. Wyniki

Zarówno testy użyteczności, jak i wywiady były niezwykle cennym źródłem informacji, umożliwiającym lepsze zrozumienie doświadczeń użytkowników, a także identyfikację mocnych i słabych stron obu serwisów.

5.1. Testy użyteczności

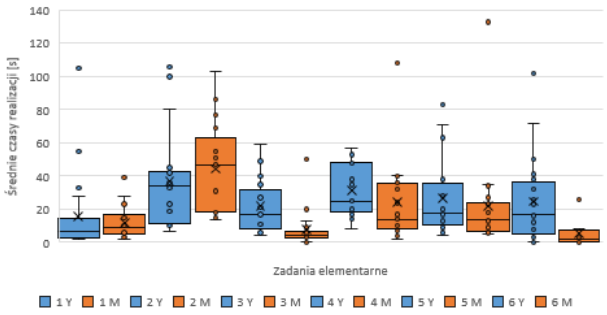
Testy te oceniały funkcjonalność, intuicyjność oraz ogólne doświadczenie użytkownika podczas korzystania z analizowanych serwisów warsztatowych. Badanie koncentrowało się na porównaniu serwisów pod względem sukcesu i porażki realizacji zadanego scenariusza, średniego czasu jego realizacji oraz średniego czasu zadania elementarnego. Zbadano, jak uczestnicy radzili sobie z poszczególnymi zadaniami, które elementy serwisów były dla nich problematyczne oraz które aspekty zostały przez nich szczególnie pozytywnie ocenione.



Rysunek 3: Współczynnik sukcesu i porażki zadań elementarnych ze scenariusza dla obu serwisów.

Rysunek 3 przedstawia porównanie efektywności użytkowania dwóch serwisów warsztatowych: Warsztaty Yanosik oraz Motointegrator, bazując na liczbie sukcesów i porażek zanotowanych w trakcie testów. Wykonanie pełnego scenariusza bez pominięcia jakiegokolwiek zadania elementarnego klasyfikowano jako sukces, natomiast pominięcie choć jednego zadania elementarnego traktowano jako porażkę. Analizując wyniki dla serwisu Warsztaty Yanosik, odnotowano 14 sukcesów i 6 porażek. W przypadku serwisu Motointegrator zaobserwowano 13 sukcesów i 7 porażek. Te dane sugerują, że wśród uczestników badania serwis Warsztaty Yanosik miał niewielką przewagę w zakresie skuteczności realizacji zadanych scenariuszy w porównaniu z serwisem Motointegrator.

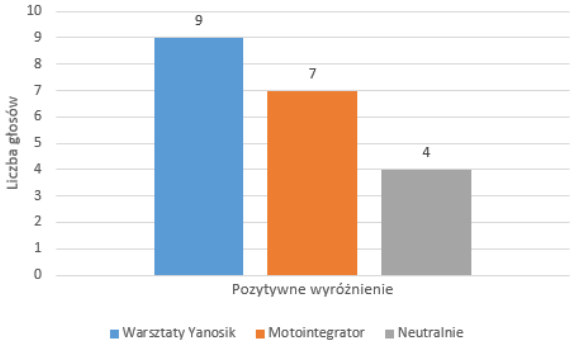
Rysunek 4 przedstawia średnie czasy realizacji zadań elementarnych dla dwóch serwisów warsztatowych: Warsztaty Yanosik oraz Motointegrator. Dane są wyrażone w sekundach i odnoszą się do konkretnych kroków w scenariuszu, z których każdy odpowiada kolejnemu zadaniu elementarnemu. Dla serwisu Warsztaty Yanosik średnie czasy w zadania od 1 do 6 oscylują w zakresie od 15,85 do 36,83 sekundy. W przypadku serwisu Motointegrator czasy są bardziej zróżnicowane i wartości zaczynają się od 7,80 sekundy a kończą na wartości 44,29 sekundy. Zadanie 7 jest analizowane osobno, ze względu na konieczność wprowadzenia danych przez użytkownika. Serwis Warsztaty Yanosik uzyskał znacznie niższy wynik 65,20 sekundy, jednak był to prosty jednoetapowy formularz zgłoszeniowy. W serwisie Motointegratorze czas realizacji wynosi 137,74 sekundy. Należy podkreślić, iż jest to znaczniej bardziej szczegółowy 4 etapowy formularz zgłoszeniowy niż formularz znajdujący się na stronie Warsztaty Yanosik. Porównując oba serwisy, można zauważyć średnie czasy krótsze w serwisie Motointegrator dla zadań 1,3,4,5 oraz 6. Serwis Warsztaty Yanosik uzyskał lepszy czas jedynie w zadaniu 2. Zestawiając ze sobą średnie czasy z zadania 7 widać prawie dwukrotnie krótszy czas dla serwisu Warsztaty Yanosik. Czas Motointegratora wypada wzorowo, ponieważ zawiera o 3 etapy więcej w formularzu zgłoszeniowym.



Rysunek 4: Średnie czasy realizacji zadań elementarnych.

5.2. Wywiad z użytkownikiem

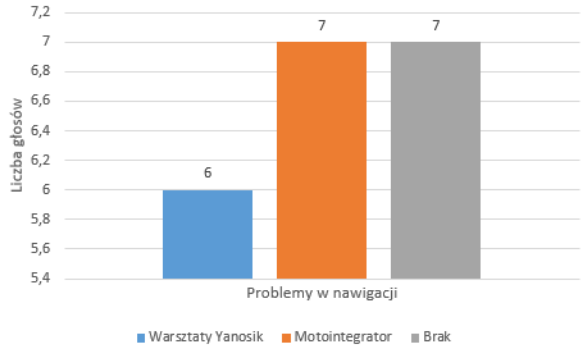
Z dziesięciu pytań, które zostały zadane uczestnikom badań, wybrano cztery, które dostarczają najbardziej wartościowe informacje. W odpowiedziach na pozostałych pytaniach nie zauważono znaczących różnic między badanymi serwisami, co skłoniło do skoncentrowania się na aspektach, które pozwoliły na głębszą analizę i porównanie.



Rysunek 5: Odpowiedzi na pytanie 1 – *Czy którykolwiek z serwisów wyróżnił się pozytywnie lub negatywnie?*

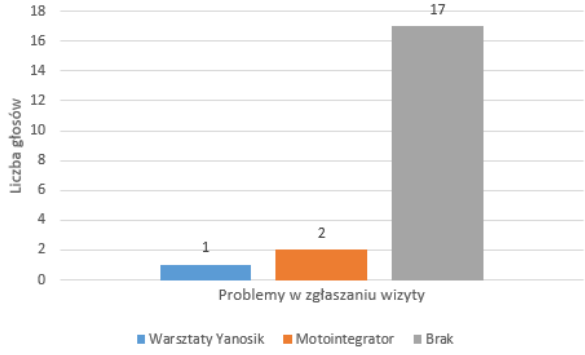
W analizowanym pytaniu dotyczącym pozytywnego lub negatywnego wyróżnienia się serwisu, wyniki przedstawia Rysunek 5: Warsztaty Yanosik zdobyły 9 pozytywnych ocen, podczas gdy Motointegrator uzyskał 7 pozytywnych opinii. Czterech ankietowanych nie wskazało jednoznacznie żadnego z serwisów, wybierając odpowiedź neutralną. Dane te zostały zaprezento-

wane na histogramie, gdzie kolorem niebieskim zaznaczono wyniki dla Warsztatów Yanosik, kolorem pomarańczowym dla Motointegratora, natomiast oceny neutralne reprezentował kolor szary.



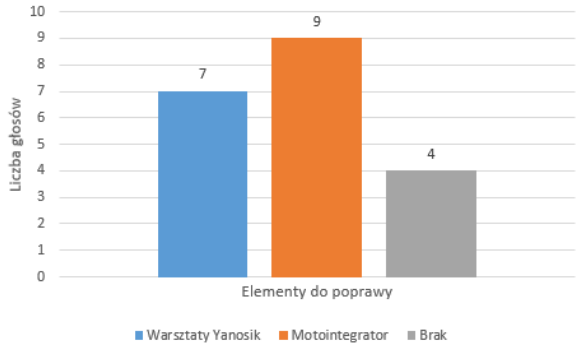
Rysunek 6: Odpowiedzi na pytanie 2 - *Czy którakolwiek ze stron była trudniejsza w nawigacji?*

Na Rysunku 6 w sposób ilościowy przedstawiono doświadczenia użytkowników w zakresie napotkanych problemów dotyczących nawigacji po obu serwisach. W serwisie Warsztaty Yanosik, sześć osób zgłosiło problemy nawigacyjne, wskazując na trudności z odnalezieniem pola marki samochodu w filtrze zaawansowanym, lokalizację przycisku otwierającego filtry zaawansowane oraz przejście do procesu umawiania wizyty. W serwisie Motointegrator, siedmiu ankietowanych doświadczyło trudności, głównie związanych z odnalezieniem filtrów oraz z gorszym układem elementów nawigacyjnych. Zauważono również, że siedem osób nie napotkało na problemy z nawigacją w żadnym z analizowanych serwisów, co również zostało zobrazowane na wykresie słupkowym.



Rysunek 7: Odpowiedzi na pytanie 5 - *Czy w trakcie zgłaszania naprawy na którejkolwiek ze stron napotkałeś na trudności?*

Na kolejnym Rysunku 7 ukazano ilość problemów, jakie użytkownicy napotkali podczas zgłaszania naprawy w analizowanych serwisach. W serwisie Warsztaty Yanosik zanotowano jedno zgłoszenie problemu, które dotyczyło funkcji wyszukiwania w zamkniętych polach formularza, działającej tylko po wpisaniu pojedynczego znaku. Z kolei w przypadku Motointegratora zidentyfikowano dwa problemy. Obydwa były związane z polem modelu auta, które oferowało użytkownikom bardzo obszerny wybór opcji. Dodatkowo na wykresie zaznaczono, że 17 respondentów nie doświadczyło żadnych problemów podczas korzystania z obu serwisów.



Rysunek 8: Odpowiedzi na pytanie 9 - *Czy masz jakieś dodatkowe uwagi lub sugestie dotyczące tego, jak możemy poprawić każdy z serwisów?*

Na kolejnym Rysunku 8 zaprezentowano liczbę zgłoszonych przez użytkowników elementów do poprawy w serwisach Yanosik i Motointegrator. Yanosik otrzymał siedem zgłoszeń, które obejmowały propozycje takie jak: dodanie terminu oddania auta do formularza, poprawienie pola filtrowania marki samochodu, poprawa wizualna strony, korekta napisu w pasku nawigacji, zwiększenie intuicyjności oraz wprowadzenie informacji o cenach w formularzu.

Z drugiej strony, serwis Motointegrator zebrał dziewięć zgłoszeń. Użytkownicy sugerowali poprawę mapki wyników, redukcję ilości informacji na stronie, przeniesienie filtrów do jednego miejsca, eliminację przycisku specjalizacji w danej marce, przeorganizowanie sekcji z reklamami, uproszczenie filtrów, automatyczne zatwierdzanie wybranych filtrów oraz dodanie filtra z promieniem odległości od danego miejsca. Dodatkowo cztery osoby nie wskazały elementów wymagających poprawy.

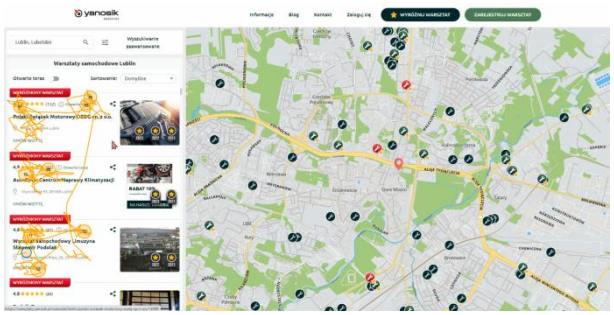
5.3. Eyetracking

W trzecim etapie badań użyto techniki eyetrackingowej. Użytkownicy mieli wykonać kilka analogicznych poleceń na dwóch testowanych serwisach. Zadania te dotyczyły: znalezienia konkretnej usługi warsztatowej przy wykorzystaniu zaawansowanej wyszukiwarki, użycia filtra związanego z marką samochodu, przesortowania wyników wg opinii innych użytkowników, wybór najlepszego warsztatu oraz sprawdzeniu czy dany warsztat ma w swojej ofercie usługę wulkanizacji.

W tym badaniu wzięło udział 5 uczestników. W wyniku otrzymano po 5 rejestracji dla każdego serwisu w postaci nagrań wideo obrazujących działania użytkowników w danym serwisie z nałożonymi na ten obraz ścieżkami skanowania składającymi się z linii (sakady) oraz okręgów (fiksacji). Rysunki 9 i 10 przedstawiają mały wycinek tych badań w postaci dwóch przykładowych ścieżek skanowania, które zostały poddane analizie jakościowej.

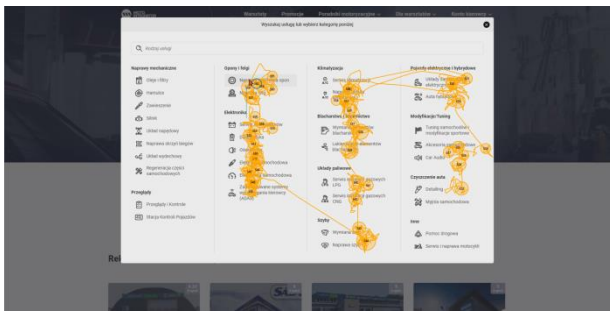
Na przedstawionym zrzucie ekranowym na Rysunku 9 obserwowano moment, w którym użytkownik dokonywał wyboru najlepszego warsztatu. Z tego fragmentu widać, że wzrok badanego koncentrował się głównie na

trzech pierwszych wynikach wyszukiwania. Użytkownik porównywał opinie między czołowymi warsztatami, dążąc do podjęcia właściwej decyzji.



Rysunek 9: Wybór najlepszego warsztatu wg opinii w serwisie Warsztaty Yanosik.

Na kolejnym zrzucie ekranowym (Rysunek 10) widać realizację zadnia polegającego na znalezieniu pożądanej przez użytkownika usługi, którą była wulkanizacja. W tym przypadku badany rozpoczął przeszukiwanie czterokolumnowej listy od końca. Wnikliwie, krok po kroku przeskanował wszystkie pozycje rozpoczynając od czwartej kolumny idąc od dołu do góry. Duża liczba elementów w każdej liście, pomimo zastosowania grupowania oraz przyjęcie przez badanego własnej strategii przeszukiwania przyczyniło się do wydłużenia czasu znalezienia szukanej opcji.



Rysunek 10: Przeszukiwanie listy oferowanych usług w serwisie Motointegrator.

6. Wnioski

Analiza została skupiona na formularzach wyszukiwarek warsztatowych. Na podstawie zaprezentowanych wyników i przeprowadzonych analiz można wysnuć kilka kluczowych wniosków. Po pierwsze, zauważalna jest niewielka przewaga serwisu Warsztaty Yanosik w zakresie skuteczności realizacji zadanych scenariuszy w porównaniu z serwisem Motointegrator. Różnice w średnim czasie realizacji zadań elementarnych wskazują na zróżnicowanie w użyteczności i efektywności interfejsów obu serwisów, co podkreśla znaczenie ciągłego doskonalenia i optymalizacji.

Analiza opinii użytkowników oraz doświadczeń z nawigacją po serwisach pokazuje, że obie platformy napotkały na trudności, lecz jednocześnie zdobyły zbliżoną ilość pozytywnych ocen. To sugeruje, że mimo pewnych problemów, użytkownicy generalnie pozytywnie oceniają oba serwisy.

Co więcej, zgłoszone problemy i propozycje poprawy wskazują na konkretne obszary, które wymagają interwencji i modyfikacji. Zarówno serwis Warsztaty Yanosik, jak i Motointegrator otrzymały różnorodne sugestie, co może być cennym źródłem informacji dla dalszego rozwoju i doskonalenia usług.

Podsumowując, mimo zauważalnych różnic i obszarów do poprawy, oba badane serwisy warsztatowe mają swoje mocne i słabe strony. Użytkownicy dostrzegali zarówno pozytywne aspekty, jak i elementy wymagające korekty, co świadczy o konieczności dalszej analizy i pracy nad optymalizacją funkcji i interfejsów obu serwisów.

Literatura

[1] J. Nielsen, Designing Web Usability: The Practice of Simplicity, New Riders Publishing, 1999.

[2] S. Krug, Don't Make Me Think: A Common Sense Approach to Web Usability, New Riders Publishing, 2000.

[3] J. Rubin, D. Chisnell, Handbook of Usability Testing: How to Plan, Design, and Conduct Effective Tests. Wiley Publishing, 2008.

[4] J. Nielsen, Usability Engineering, Morgan Kaufmann, 1993.

[5] L. Wroblewski, Web form design: filling in the blanks, Rosenfeld Media, 2008.

[6] Gazepoint GP3 HD, https://www.gazept.com/product/gp3hd/, [20.09.2023].

[7] iMotions, https://imotions.com, [24.09.2023].

[8] Serwis Warsztaty Yanosik, https://warsztaty.yanosik.pl/, [27.09.2023].

[9] Serwis Motointegrator, https://motointegrator.com/pl/pl, [27.09.2023].

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 30 (2024) 47–52
Received: 8 October 2023
Accepted: 18 November 2023

A comparative analysis of transitions generated using the Unity game development platform

Analiza porównawcza przejść generowanych przy użyciu platformy do tworzenia gier Unity

Marek Tabiszewski*

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

Abstract
This paper conducts a comparative analysis of transitions generated using the Unity engine. It selects fifteen animations featuring a humanoid character, introduces breaks in marker trajectories, and fills them with transitions generated by the game engine's animator. These transitions are then compared with the unmodified original character animation. The study compares animations by calculating the average deviation in bone rotation and position between the original and generated motion throughout the animation. The results show that the Unity engine excels in generating transitions for slow animations involving the lower body limbs, with the largest errors occurring in the bones at the extremities of the limbs.

Keywords: Unity; animation; character motion; animation quality

Streszczenie
W artykule przeprowadzono analizę porównawczą przejść generowanych przy użyciu silnika Unity. Do badań wyselekcjonowano piętnaście animacji postaci humanoidalnej, w których wprowadzono przerwy w trajektoriach markerów tak, aby można było zapełnić je przejściami wygenerowanymi przez animator użytego silnika gier, a następnie porównać przejścia z oryginalnym ruchem postaci pochodzącym z niezmodyfikowanej animacji. Animacje porównano obliczając średnie odchylenie rotacji oraz pozycji każdej z kości pomiędzy ruchem oryginalnym a wygenerowanym na przestrzeni całej animacji. Na podstawie otrzymanych wyników stwierdzono, że silnik Unity lepiej generuje przejścia pomiędzy animacjami, które są powolne i angażują dolne kończyny ciała oraz że największe błędy generują kości na końcach kończyn.

Słowa kluczowe: Unity; animacje; ruch postaci; jakość animacji

*Corresponding author
Email address: **Marek.tabiszewski@pollub.edu.pl** (M.Tabiszewski)

1. Wstęp

Naturalną tendencją gier wideo na przestrzeni lat jest ciągle polepszająca się ich jakość audiowizualna. Chcąc osiągnąć efekt jak najbardziej zbliżony do rzeczywistości stosuje się coraz bardziej szczegółowe modele, tekstury o coraz wyższej rozdzielczości oraz nowe, lepsze metody oświetlenia w pełni wykorzystując możliwości powszechnie dostępnych podzespołów komputerowych czy konsol [1]. Niemniej ważnym aspektem realizmu w grach jest jakość przedstawianych w nich animacji postaci. Istnieją prace [2,3] świadczące o tym, że wraz ze wzrostem jakości i szczegółowości przedstawianych modeli humadoidalnych, rośnie wrażliwość odbiorcy na ewentualne błędy bądź nienaturalność ich ruchu.

Współcześnie do jak najwierniejszego przedstawienia animacji humanoidalnych stosuje się nagrania w realizowane technologii motion capture. Ograniczenia technologiczne takie jak ograniczone miejsce i długość ujęcia, a także specyfika gier wideo, która często wymaga dynamicznego łączenia różnych ruchów postaci np. w reakcji na sterowanie gracza lub inne czynniki zewnętrzne, sprawia, że animacje tego typu muszą być ze sobą łączone aby otrzymać jeden płynny ruch. Takie połączenia, będące przejściem z jednej animacji w drugą są punktami narażonymi na wystąpienie nienaturalnych ruchów. Aby się przed nimi uchronić korzysta się z różnych algorytmów dobierających animacje i interpolujących pomiędzy nimi.

Zagadnienie generowania przejść pomiędzy nagraniami ruchu jest poruszane od końca lat dziewięćdziesiątych. Prace [4-8] opisują nowatorskie, jak na tamte czasy podejścia, które dawały zadowalające rezultaty, jednakże obliczenia potrzebne do wygenerowania ruchu trwały tak długo, że uniemożliwiały generację przejść w czasie rzeczywistym. Jednakże już w pracach [9-11] postęp technologiczny pozwalał na dynamiczne generowanie przejść pomiędzy nagraniami ruchu dla postaci sterowanej przez użytkownika w czasie rzeczywistym. Najnowszym podejściem do generowania przejść jest wykorzystanie sieci neuronowych. W pracach [12-14] osiągnięto niespotykane dotąd rezultaty przy użyciu dużo mniejszych zasobów obliczeniowych.

W kontekście przytoczonych przykładów podejść do problematyki generowania przejść pomiędzy nagraniami ruchu, środowisko Unity oferuje stosunkowo prymi-

tywne rozwiązanie [15], jednakże prostota jego użycia oraz możliwość ograniczonej konfiguracji czyni go dobrym narzędziem do nieskomplikowanych zastosowań. Wyniki analizy ruchów wygenerowanych za pomocą tego systemu przedstawione w tym artykule mogą być pomocne w wyodrębnieniu animacji, które mogą być szczególnie narażone na błędy.

2. Eksperyment

Jedynym sposobem na obiektywne zbadanie jakości generowanego przejścia niezależnego od czynników zależnych od użytych animacji takich jak liczba klatek na sekundę lub charakterystyka ruchu jest generowanie przejścia pomiędzy dwiema pozami i porównywanie powstałego ruchu z ruchem pomiędzy dwiema takimi samymi pozami ale pochodzącym z animacji.

2.1 Użyte oprogramowanie

W celu przeprowadzenia eksperymentu użyto następującego oprogramowania:

- Blender 2.92 - do przygotowania animacji
- Unity 2020.3.4f1 - do przeprowadzenia pomiarów
- JetBrains Rider 2021.2 - środowisko programistyczne
- LibreOffice Calc - analiza wyników

2.2 Przygotowanie materiału badawczego

Animacje użyte do przeprowadzenia badań pobrano z bazy danych nagrań motion capture Uniwersytetu Carnegie Mellon [16] i przekonwertowano do formatu FBX tak, aby można je było edytować w programie Blender. Krótki opis charakterystyki ruchu każdej z animacji przedstawiono na Tabeli 1.

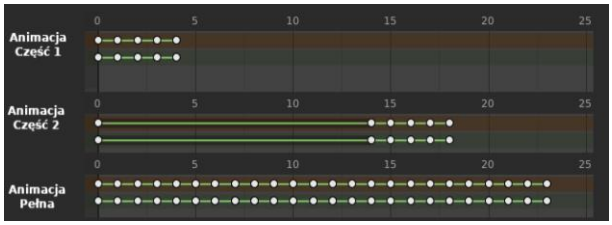
Tabela 1: Charakterystyki ruchu badanych animacji

Nazwa animacji	Opis charakterystyki ruchu
Crouch Jump	Przejście z pozycji kucającej do podskoku
Hands Forward	Podniesienie obu rąk do przodu
Hands Sides	Podniesienie obu rąk na boki
Hands Up	Podniesienie obu rąk do góry
Hour Check	Sprawdzenie godziny na zegarku ręcznym
Jumping Jack Begin	Przejście z pozycji neutralnej stojącej do pozycji wejściowej przy robieniu pajacyków
Jumping Jack End	Przejście z robienia pajacyków do pozycji neutralnej stojącej
Jump Begin	Wyskok do góry rozpoczynający się od neutralnej pozycji stojącej
Jump End	Lądowanie po wyskoku i przejście do pozycji neutralnej stojącej
Punch	Bokserski cios prawą ręką rozpoczynający się od gardy i kończący na gardzie
Side Step Begin	Przejście z pozycji neutralnej stojącej do kroku odstawno dostawnego
Side Step End	Przejście z kroku odstawno dostawnego do pozycji neutralnej stojącej
Step Up	Wejście na schodek rozpoczynające się od pozycji neutralnej stojącej
Walk Backward	Ciągły, trwający chód do tyłu
Walk Forward	Ciągły, trwający chód do przodu

Animacje zostały odpowiednio wycięte i ustawione tak, aby ruch był generowany z częstością sześćdziesięciu klatek na sekundę. Dla każdego badanego fragmentu ruchu przygotowano zestaw trzech animacji:

- animacji pełnej - wycinek trwający od piątej klatki przed badanym fragmentem do piątej klatki po badanym fragmencie
- animacji niepełnej część pierwsza - wycinek animacji zawierający 5 klatek przed badanym fragmentem
- animacji niepełnej część druga - wycinek animacji zmodyfikowany w taki sposób, że poza wyjściowa badanego fragmentu zostaje ustawiona i zachowana na czas równy czasowi trwania badanego fragmentu, następnie ruch kontynuowany jest przez pozostałe 5 klatek jak w animacji oryginalnej.

Na Rysunku 1 przedstawiono przykładowy wygląd osi czasu dla każdej animacji.

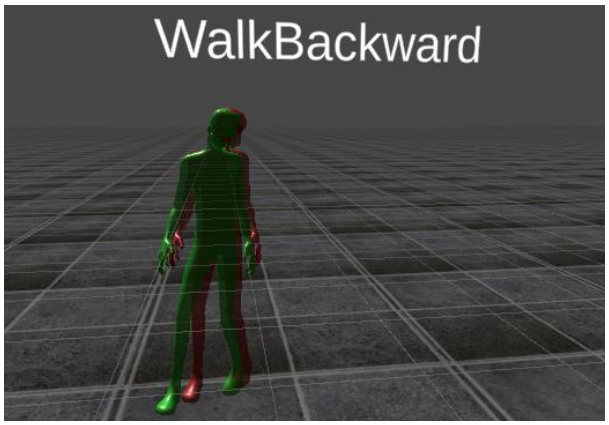


Rysunek 1: Oś czasu dla zestawu animacji w programie Blender.

Dodatkowe klatki otaczające badany fragment zostały dodane w celu upewnienia się, że animacja oryginalna i sekwencja animacji z wygenerowanym przejściem są prawidłowo zsynchronizowane. Przy prawidłowej synchronizacji we fragmentach poza generowanym przejściem różnice w pozycjach i rotacjach kości powinny być bardzo zbliżone do zera.

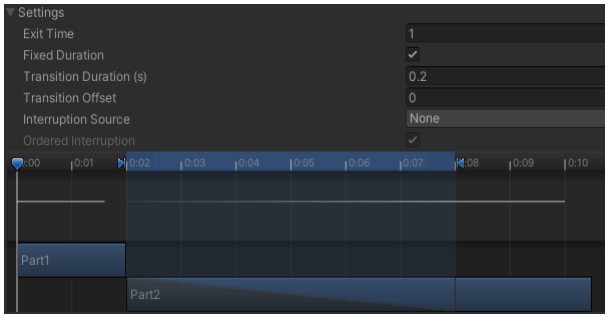
2.3 Przygotowanie środowiska badawczego w programie Unity

Tak przygotowane animacje wyeksportowano do Unity gdzie stworzono dla nich graficzne reprezentacje nazywane dalej awatarami. Przykładową parę awatarów pokazano na Rysunku 2.



Rysunek 2: Graficzna reprezentacja zestawu animacji, widoczne są niewielkie różnice powstałe w wyniku generowania przejścia.

Zielona postać prezentuje animację pełną natomiast czerwona sekwencje animacji niepełnej części pierwszej oraz części drugiej. Sposób konfiguracji sekwencji przedstawiono na Rysunku 3.



Rysunek 3: Widok osi czasu sekwencji animacji oraz jej ustawień w programie Unity.

W dolnej części rysunku widoczna jest oś czasu sekwencji. Podpisany ‘Part1’ przedstawia trwanie pierwszej części animacji niepełnej, następnie odgrywana jest część druga animacji niepełnej. We fragmencie podświetlonym na niebiesko fragment ‘Part2’ nie porusza postacią (zachowana jest postawa po badanym fragmencie) dzięki czemu silnik interpoluje pomiędzy początkową a końcową pozą badanego fragmentu co skutkuje generacją przejścia, następnie odgrywana jest końcowa część ruchu w celu sprawdzenia synchronizacji z animacją główną po wygenerowaniu przejścia.

2.4 Dokonywanie pomiarów

Procesem dokonywania pomiarów zarządza skrypt DataCollector. Jego główną funkcjonalność zaprezentowano na Listingu 1.

Listing 1. Metoda Update wykonywana co klatkę, zawarta w skrypcie DataCollector

```
private void Update()
{
    if (fullAvatarAnimationHandler.ImActive &&
        partialAvatarAnimationHandler.ImActive)
    {
        SyncRootBones();
        fullAnimationData.Add(
            item: fullAvatarAnimationHandler.GetFrameData());
        partialAnimationData.Add(
            item: partialAvatarAnimationHandler.GetFrameData());
        fullAvatarAnimationHandler.Animator // Animator
            .Update(Time.deltaTime);
        partialAvatarAnimationHandler.Animator // Animator
            .Update(Time.deltaTime);
    }
    else
    {
        SaveData();
        enabled = false;
    }
}
```

Co klatkę działania programu następuje synchronizacja pozycji i rotacji obu awatarów tak aby zniwelować błędy w pomiarach wynikające z przesunięcia całego modelu w wyniku animacji, następnie informacje o kościach są pobierane i zapisywane z każdego z awatarów co zaprezentowano na Listingu 2.

Listing 2. Metoda odpowiadająca za pobieranie danych o stanie szkieletu zawarta w skrypcie AvatarAnimationHandler

```
public List<BoneFrameData> GetFrameData() {
    List<BoneFrameData> bonesFrameData = new List<BoneFrameData>();
    foreach (var keyValuePair in bones) {
        Transform boneT = keyValuePair.Value;
        bonesFrameData.Add(
            item: new BoneFrameData(boneT.position,boneT.rotation));
    }
    return bonesFrameData;
}
```

W dalszej kolejności oba animatory są aktualizowane. Po zakończeniu pracy przynajmniej jednego z animatorów obliczane są różnice w pozycjach oraz kąty pomiędzy rotacjami kości awatara animacji oryginalnej i z przejściem, dla każdej klatki. Implementację tych obliczeń zaprezentowano na Listingach 3 oraz 4.

Listing 3. Metoda odpowiadająca za obliczenie kątów pomiędzy rotacjami kości

```
private void CompareRotations(ref string[,] initialTable,
                List<List<BoneFrameData>> animationDataA,
                List<List<BoneFrameData>> animationDataB) {
    for (int j = 1; j < animationDataA.Count + 1; j++)
        for (int k = 1; k < animationDataA[0].Count + 1; k++) {
                float difference =
                    Quaternion.Angle(
                        a: animationDataA[j - 1][k - 1].rot,
                        b: animationDataB[j - 1][k - 1].rot);
                initialTable[j, k] = difference.ToString(culture);
        }
}
```

Metoda CompareRotations przyjmuje jako parametry dane z dwóch badanych awatarów oraz referencję do tablicy dwuwymiarowej, która przetrzymuje wyniki obliczeń. Funkcja iteruje po każdej parze danych z dwóch awatarów, biorąc pod uwagę każdą kość w każdej klatce generowania przejścia i oblicza kąt pomiędzy rotacjami kości należących do dwóch awatarów, następnie wpisuje wynik do tablicy.

Listing 4. Metoda odpowiadająca za obliczenie różnicy pozycji pomiędzy rotacjami kości

```
private void ComparePositions(ref string[,] initialTable,
    List<List<BoneFrameData>> animationDataA,
    List<List<BoneFrameData>> animationDataB)
{
    for (int j = 1; j < animationDataA.Count + 1; j++)
        for (int k = 1; k < animationDataA[0].Count + 1; k++)
        {
            float difference = Vector3.Magnitude(
                vector: animationDataA[j - 1][k - 1].pos -
                animationDataB[j - 1][k - 1].pos);
            initialTable[j, k] = difference.ToString(culture);
        }
}
```

Metoda ComparePositions działa analogicznie do metody CompareRotations i oblicza różnicę pomiędzy pozycjami kości należących do dwóch awatorów poprzez obliczenie długości wektora będącego różnicą

wektorów pozycji porównywanych kości, następnie zapisuje wynik.

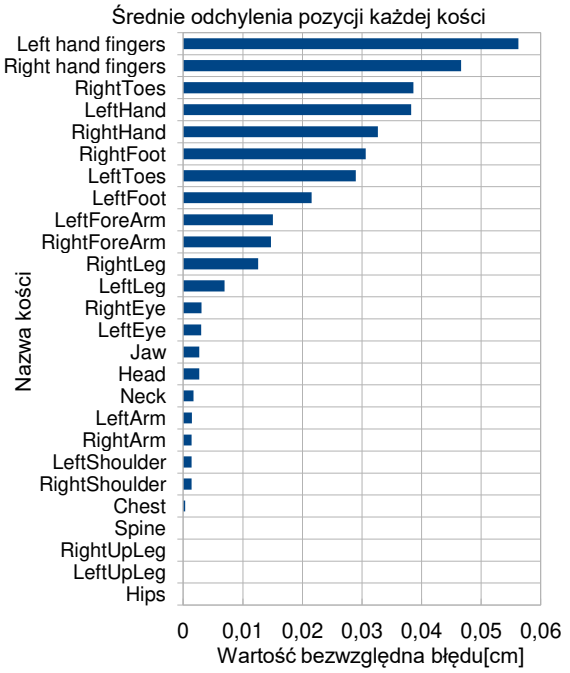
Otrzymane w ten sposób dane są parsowane do postaci tabelarycznej i zapisywane pliku CSV. Dla każdej z badanych animacji generowany jest oddzielny plik zawierający dane dotyczące rotacji oraz drugi zawierający dane dotyczące pozycji. Pliki importowane są do programu LibreOffice Calc, gdzie poddawane są analizie, oraz generowane są wykresy w celu graficznego przedstawienia wyników badań.

3. Wyniki badań

Podczas eksperymentu przeanalizowano 15 animacji przedstawiających różnorodne ruchy postaci humanoidalnej. Do analizy wykorzystano wartości bezwzględne błędów. Zostały one przeliczone na centymetry według instrukcji podanej przez twórców animacji źródłowych [17].

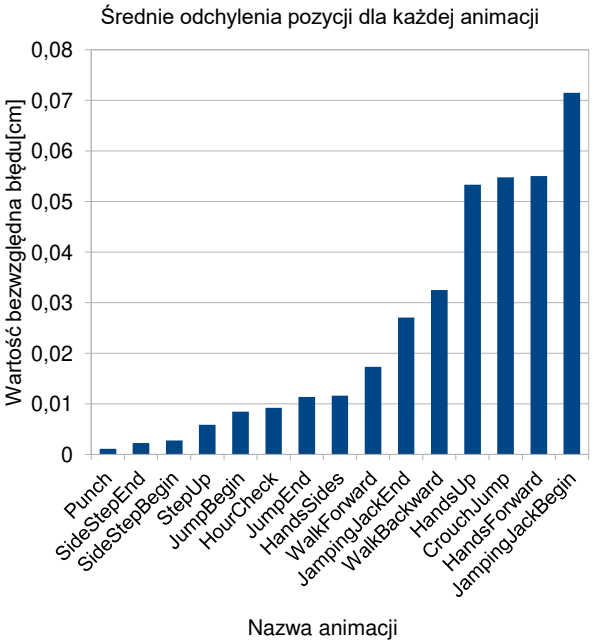
Zdecydowano się na pominięcie danych pochodzących z kości animujących części twarzy, ponieważ żadna z badanych animacji nie dotyczyła mimiki. Ponadto uśredniono dane pochodzące z kości palców u obu dłoni i potraktowano jako pojedyncze kości, po jednej na każdą dłoń, gdyż błędy generowane przez kości palców były bardzo zbliżone do siebie i ich liczba utrudniała analizę wyników. Wyniki przeprowadzonych badań zostały przedstawione w formie czterech porównań:

- porównania średnich odchyleń pozycji na klatkę dla każdej kości (Rysunek 4)
- porównania średnich odchyleń pozycji na klatkę dla całego szkieletu (Rysunek 5)
- porównania średnich odchyleń rotacji na klatkę dla każdej kości (Rysunek 6)
- porównania średnich odchyleń rotacji na klatkę dla całego szkieletu (Rysunek 7)



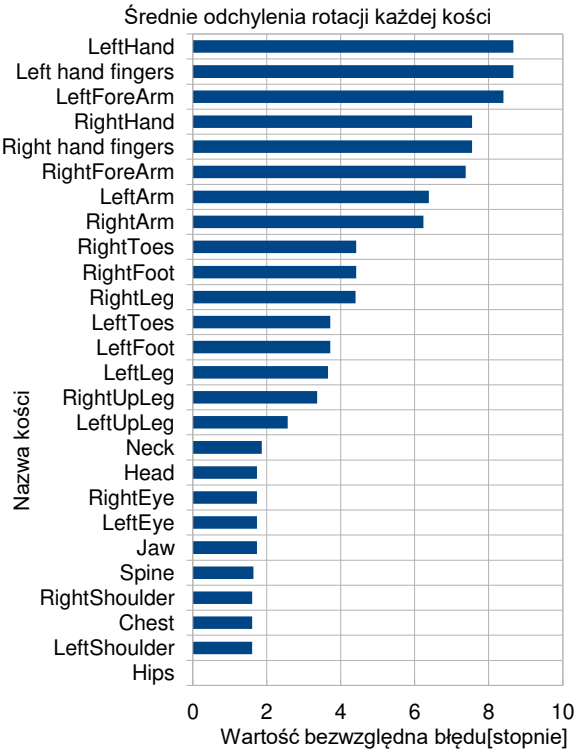
Rysunek 4. Wykres przedstawiający średnie odchylenia pozycji dla każdej kości.

Na Rysunku 4 widać, że największe odchylenia występują w kościach dłoni oraz stóp, natomiast najmniejsze w kościach tułowia i ud.



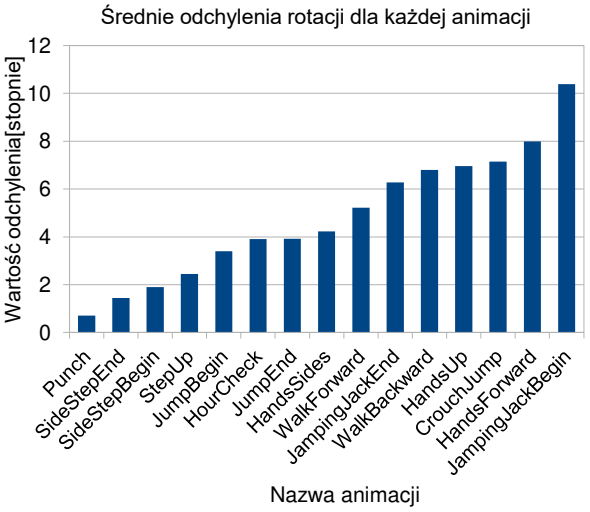
Rysunek 5. Wykres przedstawiający średnie odchylenia pozycji dla każdej animacji.

Po analizie średnich odchyleń dla pojedynczych animacji zaobserwowano, że największe błędy występują w animacjach, w których ruch jest dynamiczny i angażowane są górne kończyny postaci. Mniejsze błędy występują w animacjach powolnych, bądź angażujących tylko jedną kończynę (Rysunek 5).



Rysunek 6. Wykres przedstawiający średnie odchylenia rotacji dla każdej kości.

W przypadku odchyleń w rotacji zaobserwowano około dwukrotnie większe wartości błędów dla kości kończyn górnych niż dla kości kończyn dolnych. Najmniejsze wartości odchyleń, podobnie jak w przypadku odchyleń pozycji, zanotowano dla kości tułowia oraz głowy, zostało to zaprezentowane na Rysunku 6.



Rysunek 7. Wykres przedstawiający średnie odchylenia rotacji dla każdej animacji.

Na wykresie z Rysunku 7 można zauważyć, że występuje silna korelacja wartości błędów odchylenia pozycji z wartością błędów odchylenia rotacji. Animacje, które mają tendencję do generowania dużych błędów odchyleń pozycji, generują również duże błędy odchyleń rotacji.

4. Podsumowanie

Celem niniejszego artykułu była analiza porównawcza przejść generowanych przez silnik Unity w przypadku różnych rodzajów ruchów. Przeprowadzono badanie odchyleń rotacji oraz pozycji dla każdej kości awatarów użytych do badania. Wyniki przeanalizowano w kontekście porównania średnich błędów każdej animacji oraz w kontekście średnich błędów każdej kości.

Zauważono, że największe błędy generują ruchy angażujące górne kończyny postaci, oraz prezentują dużą dynamikę ruchu. Ponadto zaobserwowano tendencję do kumulowania się błędów w kościach, które występują na końcach kończyn (np. palców lub dłoni). Warto również zauważyć, że w przypadku badania błędów w kontekście całych animacji wystąpiła korelacja pomiędzy wartościami błędów rotacji oraz pozycji, czego nie można powiedzieć o kontekście pojedynczych kości.

Obserwacje wynikające z przeprowadzonego badania, pozwalają wyciągnąć wniosek, że silnik Unity lepiej generuje przejścia pomiędzy animacjami, które są powolne i angażują dolne kończyny ciała. Może to stanowić wstęp do bardziej szczegółowej analizy podjętego tematu.

Literatura

[1] M. Masuch, N. Röber, Game graphics beyond realism: Then, now and tomorrow, Level UP: digital games research conference (DIGRA), Faculty of Arts, University of Utrecht, 2004, http://www.digra.org/wp-content/uploads/digital-library/05150.48223.pdf.

[2] J.K. Hodgins, J.F. O'Brien, J. Tumblin, Perception of human motion with different geometric models, IEEE Transactions on Visualization and Computer Graphics 4 (4) (1998) 307-316, https://doi.org/10.1109/2945.765325.

[3] M. Oesker, H. Hecht, B. Jung, Psychological Evidence for Unconscious Processing of Detail in Real-time Animation of Multiple Characters, The Journal of Visualization and Computer Animation 11 (2) (2000) 105-112, https://doi.org/10.1002/1099-1778(200005)11:2<105::AID-VIS222>3.0.CO;2-Q.

[4] J. Lee, J. Chai, P.S. Reitsma, J.K. Hodgins, N.S. Pollard, Interactive control of avatars animated with human motion data, ACM Transactions on Graphics 21 (3) (2002) 491-500, https://doi.org/10.1145/566654.566607.

[5] C. Rose, B. Guenter, B. Bodenheimer, M.F. Cohen, Efficient generation of motion transitions using spacetime constraints, In Proceedings of the 23rd annual conference on Computer graphics and interactive techniques (1996) 147-154, https://doi.org/10.1145/237170.237229.

[6] T. Polichroniadis, N. Dodgson, Motion blending using a classifier system, In Proceedings of the 7th International Conference in Central Europe on Computer Graphics 1 (1999) 225-232, http://www.neildodgson.com/pubs/WSCG99.pdf.

[7] G. Ashraf, K.C. Wong, Generating consistent motion transition via decoupled framespace interpolation, Blackwell Publishers Ltd. Oxford, UK and Boston, USA, Computer Graphics Forum 19 (3) (2000) 447-456, https://doi.org/10.1111/1467-8659.00437.

[8] L. Kovar, M. Gleicher, Flexible automatic motion blending with registration curves, In Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation (SCA '03), Eurographics Association, Goslar, DEU, (2003) 214–224, http://dx.doi.org/10.2312/SCA03/214-224.

[9] M. Gleicher, H.J. Shin, L. Kovar, A. Jepsen, Snap-together motion: assembling run-time animations, ACM SIGGRAPH (2008) 1-9, https://doi.org/10.1145/641480.641515.

[10] V.B. Zordan, A. Majkowska, B. Chiu, M. Fast, Dynamic response for motion capture animation, ACM Transactions on Graphics 24 (3) (2005) 697-701, https://doi.org/10.1145/1073204.1073249.

[11] H.J. Shin, H.S. Oh, Fat graphs: constructing an interactive character with continuous controls, In Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation (2006) 291-298, http://dx.doi.org/10.2312/SCA/SCA06/291-298.

[12] D. Holden, T. Komura, J. Saito, Phase-functioned neural networks for character control, ACM Transactions on Graphics (TOG), 36 (4) (2017) 1-13, https://doi.org/10.1145/3072959.3073663.

[13] F. Gaisbauer, P. Fröhlich, J. Lehwald, P. Agethen, E. Rukzio, Presenting a Deep Motion Blending Approach for Simulating Natural Reach Motions, Eurographics (Posters) (2018) 5-6, http://dx.doi.org/10.2312/egp.20181010.

[14] F. Gaisbauer, J. Lehwald, J. Sprenger, E. Rukzio, Natural posture blending using deep neural networks, In Proceedings of the 12th ACM SIGGRAPH Conference

on Motion, Interaction and Games (2019) 1-6, https://doi.org/10.1145/3359566.3360052.

[15] Dokumentacja Unity dotycząca generowania przejść, https://docs.unity3d.com/Manual/class-Transition.html, [01.10.2023].

[16] Baza danych animacji używanych w badaniu, http://mocap.cs.cmu.edu/, [01.10.2023].

[17] Informacje dotyczące przekonwertowania jednostek miary w animacjach na centymetry, http://mocap.cs.cmu.edu/faqs.php, [01.10.2023].

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 30 (2024) 53–60
Received: 17 October 2023
Accepted: 4 December 2023

Comparative analysis of the performance of Unity and Unreal Engine game engines in 3D games

Analiza porównawcza wydajności silników Unity i Unreal Engine w grach 3D

Kamil Abramowicz*, Przemysław Borczuk

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

Abstract
The article compared the performance of the Unity and Unreal Engine game engines based on tests conducted on two nearly identical games. The research focused on frames per second, CPU usage, RAM, and GPU memory. The results showed that Unity achieved a better average frame rate. Unreal Engine required more RAM and GPU resources. Analyzing CPU load values revealed that on the first system, Unity demanded less CPU usage. However, on the second system, Unreal Engine used over 10 percentage points less CPU. The conclusions from the research partially confirm the hypothesis that Unity requires fewer computer resources, although in some cases, Unreal Engine may demand fewer CPU resources.

Keywords: game engine; Unity; Unreal Engine; performance

Streszczenie
W artykule porównano wydajność silników gier Unity i Unreal Engine na podstawie badań przeprowadzonych na dwóch bliźniaczo podobnych grach. Badania skupiły się na: liczbie klatek na sekundę, użyciu procesora (CPU), RAM i karty graficznej. Wyniki wykazały, że Unity osiągnął lepszą średnią liczbę klatek na sekundę. Unreal Engine wymagał większych zasobów pamięci RAM oraz karty graficznej. Przeanalizowane wartości obciążenia CPU pokazały, że na pierwszym stanowisku silnik Unity wymagał mniejszego użycia CPU. Natomiast na drugim stanowisku Unreal Engine wykorzystywał ponad 10 punktów procentowych CPU mniej. Wnioski z przeprowadzonych badań częściowo potwierdzają tezę, że Unity wymaga mniej zasobów komputera, choć w niektórych przypadkach Unreal Engine może wymagać mniej zasobów CPU.

Słowa kluczowe: silnik gier; Unity; Unreal Engine; wydajność

*Corresponding author
Email address: **kamil.abramowicz@pollub.edu.pl** (K. Abramowicz)

1. Wstęp

Silniki gier, dzięki swojej wszechstronności oraz możliwości ich użycia przy wielu różnych projektach, odgrywają znaczącą rolę podczas tworzenia gier. Dzięki nim deweloperzy mogą skupić się na konstruowaniu jak najlepszych aplikacji wykorzystując dostępne funkcje oraz rozwiązania używane w poprzednich produkcjach. Dostępne na rynku silniki znacznie przyspieszają rozwój projektów, jak również zwiększają jakość produktu. Narzędzia te są rozwijane wraz z upływem lat, co poprawia ich wydajność. Darmowy dostęp do środowisk ułatwiających urzeczywistnianie własnych pomysłów na gry przyciąga licznych deweloperów specjalizujących się w wytwarzaniu zasobów przeznaczonych właśnie dla takich środowisk.

Wszystkie wyżej wymienione zalety przyczyniają się do usprawnienia procesu tworzenia gier, umożliwiając rozwijanie zoptymalizowanych aplikacji w krótszym czasie. Duży wpływ na działanie gry mają efekty cząsteczkowe, które używane są przy np. wybuchach czy opadach płatków śniegu. Innym czynnikiem wpływającym na wydajność aplikacji jest renderowanie. Przy generowaniu obiektów graficznych złożonych z dużej liczby wielokątów gra może wymagać zbyt dużo zasobów komputera. Nie można jednak skupić się wyłącznie na optymalizacji jednego obszaru projektu i zignorować pozostałych, ponieważ w takim przypadku gra nie będzie działać wystarczająco dobrze.

Biorąc pod uwagę popularność silników gier, jak również ich możliwości, do analizy wybrane zostały dwa: Unity oraz Unreal Engine. Na potrzeby porównania w podanych środowiskach programistycznych napisana została gra 3D. Następnie w każdej z nich zmierzono zużycie zasobów komputerowych podczas wykonywania przez gracza akcji takich jak np. strzał z pistoletu, rzut granatem lub nawigacja postacią.

Celem niniejszej pracy było wieloaspektowe porównanie wydajności silników Unity i Unreal Engine oraz przeanalizowanie wymaganych zasobów komputera przez utworzoną w nich grę 3D. W badaniach skupiono się na obciążeniu sprzętu wywieranym przez grę podczas wykonywania przez gracza różnych akcji.

Zakres pracy obejmował wybór obiektów badawczych (silników gier), metodyki badań, a także mierzonych zasobów komputera. Zaprojektowana została trójwymiarowa gra, która została wdrożona w dwóch środowiskach w taki sposób, aby była jak najbardziej zbliżona w wybranych środowiskach. Kolejnym krokiem

było wykonanie badań oraz analiza wyników. Z przeprowadzonych działań wyciągnięto oraz sformułowano wnioski.

Na potrzeby badania postawiono następującą hipotezę badawczą: Gry 3D tworzone z pomocą Unity wymagają mniej zasobów komputera niż te zaprogramowane przy użyciu Unreal Engine.

2. Przegląd literatury

W trakcie projektowania i programowania gier bardzo ważna jest optymalizacja, aby gra działała wydajnie na jak największej liczbie urządzeń. Wiele prac skupia się głównie na zestawieniu silników pod względem ich funkcjonalności i możliwości. Przedmiotem pracy [1] było porównanie efektywności aplikacji opracowanych przy pomocy środowisk programistycznych Unity oraz CryEngine. Wytworzone aplikacje posiadały zbliżone poziomy rozgrywki, a także podobne skrypty generujące obiekty. Projekty pozwoliły na wygenerowanie określonej liczby obiektów obciążających silniki. Przeanalizowano zużycie pamięci RAM, wykorzystanie procesora, liczbę klatek na sekundę oraz czas generowania obiektów przez narzędzie. Wybrano cztery poziomy obciążeń - 1000, 5000, 7500 oraz 10000 wygenerowanych obiektów. Odczyt mierzonych parametrów dokonano po minucie od wygenerowania obiektów, pomiary powtórzono trzykrotnie. Generowane obiekty były rozmieszczane losowo w obrębie sceny, posiadały one właściwości fizyczne: masę, podatność na grawitację oraz kolizje. Na podstawie uzyskanych wyników wyciągnięto następujące wnioski: Unity potrzebuje mniejszej ilości zasobów do sprawnego działania, niezależnie od liczby obiektów wymagane zasoby dla CryEngine są zbliżone, silnik fizyki CryPhysics nie radzi sobie z dużą liczbą obiektów fizycznych.

Temat zestawienia ze sobą silników gier podjęty został w pracy [2], w której przeprowadzono analizę porównawczą środowisk CryEngine, Unreal Engine oraz Unity. Kryteriami porównania były możliwości techniczne oraz czynniki wpływające na popularność wśród użytkowników. Celem pracy było wykazanie słabych i mocnych stron oraz prezentacja różnic pomiędzy narzędziami. Badając różnice brano pod uwagę wiele czynników m.in. wieloplatformowość, języki programowania, modelowanie czy dokumentację. Wysnuto wnioski, że Unity jest najlepszym silnikiem dla początkujących, ponieważ posiada dobrą dokumentację, dostępnych jest wiele kursów i szablonów, wspiera wiele platform oraz jako jedyny z porównywanych narzędzi posiada dedykowany tryb 2D. Wadami tego środowiska są zaś najsłabsza jakość graficzna tworzonych na nim gier, niewielkie możliwości wbudowanych narzędzi do animacji, reżyserowania przerywników filmowych oraz dodawania efektów dźwiękowych. Unreal Engine posiada szeroki zakres opcji oraz edytorów, które mogą przytłoczyć początkującego użytkownika. Dzięki możliwości użycia Blueprint'ów Unreal Engine jest dobrym rozwiązaniem dla użytkowników bez umiejętności programowania. Silnik ten udostępnia dobry edytor materiałów, ułatwia tworzenie zaawansowanych animacji oraz posiada systemy wspierające implementacje sztucznej inteligencji. W większości eksperymentów gra zaimplementowana w tym środowisku była najmniej efektywna. Silnik CryEngine jest narzędziem, które nie jest polecane dla użytkowników bez doświadczenia w pracy nad grami, posiada on małą liczbę kursów oraz słabą dokumentację. Narzędzie to jest przeznaczone do produkcji gier typu FPS (strzelanka pierwszoosobowa, ang. First-person shooter). W testach wydajności gra zaimplementowana w CryEngine z najwyższymi ustawieniami graficznymi działała najbardziej optymalnie.

Innym sposobem porównania silników gier Unity i Unreal Engine jest znalezienie wspólnych części i porównanie ich z osobna tak jak zrobiono to w pracy [3]. Według autora wszystkie silniki gier składają się z silników dźwięku, fizyki, renderowania oraz animacji i sztucznej inteligencji. Moduł odpowiadający za renderowanie pozwala na wyświetlanie obiektów poprzez kontrolowanie karty graficznej. Gra potrzebuje także starannie dobranej lub skomponowanej muzyki, która jest wgrywana i dostosowywana przez silnik dźwięku. Natomiast silnik fizyki pozwala na zredukowanie czasu programowania każdej reguły fizyki, która jest zauważalna, ponieważ wystarczy uruchomić ją wybraną metodą klasy. Sztuczna inteligencja odpowiada za zachowania obiektów niekontrolowanych bezpośrednio przez gracza takich jak zwierzęta, ludzi itp. Niezbędną częścią nowoczesnych gier są animacje np. przeładowanie pistoletu czy poruszanie się postaci. Z badań wykonanych przez autorów wynika, że obydwa narzędzia są bardzo podobne z tym, że Unreal Engine ma np. lepsze efekty świetlne natomiast Unity produkuje lepszej jakości cienie. Wybór więc zależy od preferencji dewelopera.

W dzisiejszych czasach bardzo ważną częścią tworzenia gier jest zapewnienie jak najbardziej realistycznej grafiki współpracującej z jak najlepszą fizyką. Dodatkowe narzędzie, które pomaga nadać grze realizmu to efekty cząsteczkowe. Pozwalają one urzeczywistnić świat gry poprzez dodanie unoszących się pojedynczych płatków śniegu, odłamków powstających w wyniku wybuchów oraz wielu innych efektów. Zhang J. opisał w swojej pracy [4] proces tworzenia cząsteczek śniegu oraz jego optymalizację używając Unreal Engine 4. Silnik ten używa modułów opisujących każdy aspekt wytwarzanych cząsteczek. Wynik ustawień jest widoczny od razu po wprowadzeniu zmian ułatwiając osiągnięcie pożądanego zachowania i wyglądu. Optymalizacja efektów cząsteczkowych jest najczęściej wykonywana poprzez zredukowanie liczby systemów cząsteczek oraz ich zakresu działania, zmniejszenie liczby emiterów, liczby emitowanych cząsteczek i czasu ich istnienia. Lepszą wydajność można uzyskać także przy pomocy rzadszego obliczania kolizji lub innych zachowań cząsteczek. Kolejnym sposobem optymalizacji gry jest ustalenie odległości z jakiej cząsteczki stają się widoczne i nie wykonywać obliczeń, kiedy są one poza zasięgiem widoczności. Podsumowując, Unreal Engine znacząco ułatwia tworzenie efektów cząsteczkowych oraz ich optymalizację, pozwalając na zmniejszenie obciążenia urządzenia i osiągnięcie równowagi między

efektywnością a atrakcyjnością emitowanych cząsteczek.

W wielu przeanalizowanych pracach skupiono się na porównaniu ogólnych właściwości silników gier. Najlepszym przykładem jest praca [5], w której autorzy stworzyli tabelę porównawczą dla badanych narzędzi. W kolejnej pracy [6] na podstawie najbardziej znanych gier wyprodukowanych przy użyciu wybranych środowisk, przeprowadzono testy, które miały na celu ocenę efektywności silników gier. Badane w pracy gry zostały wyprodukowane w znacznym odstępie czasu, przez różne firmy, a także są to inne gatunki gier, dlatego w naszej pracy przyjęliśmy inną metodologię. Sposobem przyjętym do wykonania tej pracy był sposób pokazany w artykule [7] i [8], który polega na zbudowaniu bliźniaczych gier na dwóch silnikach, co eliminuje problemy z poprzedniej pracy. W pracy [9] autorzy przedstawili porównanie silników gier za pomocą tabel, w których zaznaczono różnice w wielu aspektach takich jak renderowanie, fizyka, a nawet platformy, na które można zbudować aplikację. Z kolei pozycja [10] skupia się na procesach teksturowania obiektów i określeniu najbardziej efektywnego sposobu pracy. Teksturowanie jest nieodzownym elementem procesu produkcji gier, lecz duża liczba tekstur o wysokiej rozdzielczości może niekorzystnie wpłynąć na działanie gry. Autor artykułu [11] wybiera czternaście silników gier i krótko opisuje każdy z nich. W porównaniu brane pod uwagę są aspekty takie jak platforma publikacji gry, koszt korzystania z narzędzia, języki programowania i wspierane platformy.

Autorzy niektórych prac badali tylko jeden wybrany aspekt silnika, przez co analiza narzędzi nie jest wyczerpująca. Dlatego w niniejszej pracy zdecydowano się na porównanie wydajności bliźniaczych gier utworzonych za pomocą dwóch silników.

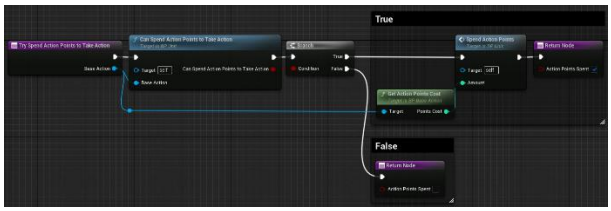
3. Metoda badawcza

Celem przeprowadzonych badań było porównanie wydajności silników gier Unity oraz Unreal Engine w zaprojektowanej strategicznej grze 3D. Gra w środowisku Unity została przygotowana przy użyciu języka programowania C#, podczas gdy w przypadku Unreal Engine skorzystano wyłącznie z Blueprint'ów - narzędzia do graficznego programowania, które pozwala stworzyć grę bez konieczności pisania kodu w języku C++. Wygląd przykładowej funkcji wykonanej z wykorzystaniem C# i Blueprint'ów widoczny jest na Listingach 1 i 2.

Listing 1: Wygląd przykładowej funkcji w C# (Unity)

```
public bool TrySpendActionPointsToTakeAction(BaseAction baseAction)
{
    if (CanSpendActionPointsToTakeAction(baseAction))
    {
        SpendActionPoints(baseAction.GetActionPointsCost());
        return true;
    }
    else
    {
        return false;
    }
}
```

Listing 2: Wygląd przykładowej funkcji w Blueprint'cie (Unreal Engine)



Do zmierzenia wydajności wykorzystane zostały narzędzia MSI Afterburner i RivaTuner Statistics Server, a mierzonymi wskaźnikami były: liczba klatek na sekundę, wykorzystanie procesora komputera, użycie pamięci RAM oraz procesora graficznego.

3.1. Obiekt badań

Obiektem badań była turowa gra strategiczna, w której użytkownik steruje postaciami na utworzonym poziomie. Obliczanie ścieżki poruszania się postaci wykonano za pomocą algorytmu A* Pathfinding [12]. Gracz ma także dostęp do innych rodzajów ruchów tj. przeczekania tury, strzału bronią palną, rzutu granatem, wykonania ataku z bliska, czy interakcji z drzwiami. Mapa odkrywana jest przed użytkownikiem wraz z otwarciem drzwi do kolejnych pomieszczeń. Dana gra została zaprogramowana na dwóch silnikach gier - Unity oraz Unreal Engine. Wykorzystano najnowszą wersję Unreal Engine 5.2.1 oraz najnowszą długo wspieraną odsłonę (LTS – ang. Long Term Support) Unity – 2021.3.18f1.

3.2. Stanowiska testowe

Zbudowane gry przetestowane zostały na dwóch komputerach, których parametry znajdują się w Tabeli 1. Na pierwszym stanowisku gra uruchomiona została w rozdzielczości 2560x1440 pikseli, na drugim zaś w rozdzielczości 1920x1080 pikseli.

Tabela 1: Stanowiska testowe

	Stanowisko nr 1 (S1)	Stanowisko nr 2 (S2)
System operacyjny	Windows 11	Windows 10
Procesor	Intel Core i7-13700K, 16 rdzeni (8 Performance + 8 Efficient)	Intel Core i5-9300HF, 4 rdzenie
Pamięć RAM	32 GB DDR4, 4000 MHz, CL 18-22-22-42	8 GB DDR4, 2400 MHz, CL 17-17-17-39
Karta graficzna	NVIDIA RTX 4080, 16 GB GDDR6X	NVIDIA GeForce GTX 1650, 4 GB GDDR5

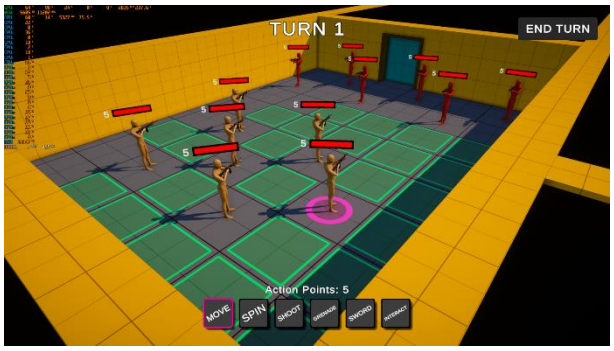
3.3. Przebieg testów

Dla porównania obu silników do tworzenia gier zostały zmierzone następujące zasoby:

- liczba klatek na sekundę,
- wykorzystanie procesora (CPU),
- średnie użycie pamięci RAM,
- średnie wykorzystanie pamięci karty graficznej (pamięć GPU).

Zasoby były mierzone podczas:

- braku aktywności ze strony użytkownika,
- poruszania się postaci (chodzenie jednostkami po mapie, otwieranie drzwi, ruch kamery),
- wykonywania funkcji ataku (strzał z broni palnej, rzut granatem, uderzenie mieczem).



Rysunek 1: Wygląd gry wykonanej w Unity, z widoczną nakładką programu mierzącego statystyki w lewym górnym rogu.

Rysunek 2: Wygląd gry wykonanej w Unreal Engine, z widoczną nakładką programu mierzącego statystyki w lewym górnym rogu.

Wykonywane badania zostały podzielone na 3 grupy, mianowicie: funkcje ataku, poruszania oraz bezruch. Badania podczas braku aktywności użytkownika polegały na uruchomieniu gry, odczekaniu 10 minut, aby wykonały się ewentualne procesy w tle np. kompilacja shaderów, i 15 minutowym pomiarze przez narzędzie do monitorowania zasobów. W testach poruszania postacią zawierały się: akcja poruszania, interakcja z drzwiami w celu odblokowania dostępu do kolejnych pomieszczeń, ruch kamery - brak ruchu kamery ogranicza dostęp gracza do części mapy, która znajduje się poza zasięgiem kamery, czasami konieczne jest obrócenie i przybliżenie kamery, aby wskazać drzwi. Pomiary grupy poruszania polegały na wykorzystaniu wszystkich dostępnych punktów akcji poprzez poruszanie postaciami i otwieranie drzwi. Użytkownik ma do dyspozycji 6 sojuszniczych jednostek, każda z pięcioma punktami akcji. W trakcie jednego uruchomienia gry wykonano 30 ruchów. Badanie powtórzono 10 razy, co dało łącznie 300 działań. Badania akcji strzału i rzutu granatem wyglądały podobnie do siebie. Wykorzystano wszystkie dostępne punkty, w ramach badanej akcji, dla każdej sojuszniczej jednostki do momentu ich wykorzystania - łącznie 300 strzałów oraz 300 rzutów granatem. Ostatnie testy wykonywania uderzenia mieczem przeprowadzono poprzez wcześniejsze podejście jednostkami tak, aby przeciwnicy byli w zasięgu. Następnie włączono monitorowanie zasobów i wykonano każdym sojusznikiem dwa ataki mieczem na sześciu przeciwnikach, co dało łącznie 12 uderzeń mieczem. Mimo dostępnych punktów akcji nie został ani jeden przeciwnik, na którym można by było wykonać uderzenie. Pomiary wykonano 10 razy - łącznie 120 ruchów. Wszystkie powyższe testy zostały wykonane na dwóch wariantach gry (Unity oraz Unreal Engine) przy pomocy dwóch stanowisk.

4. Analiza wyników badań

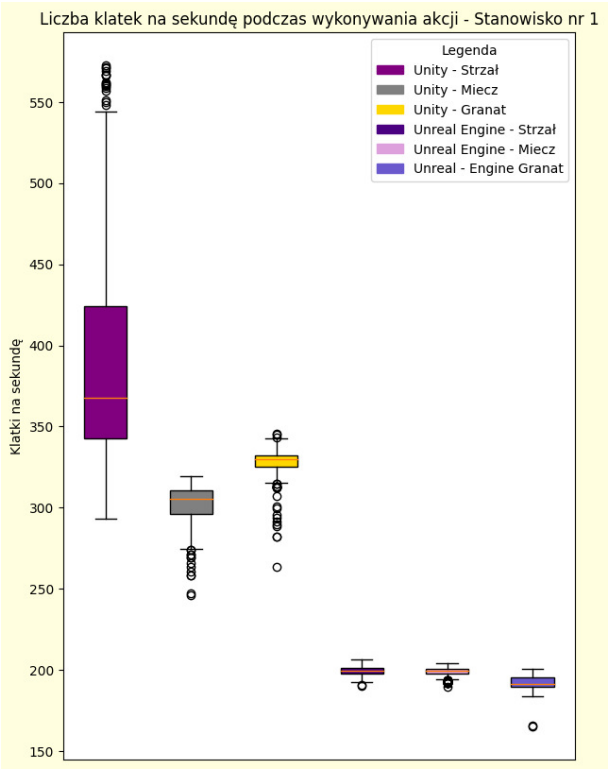
W celu wizualizacji zebranych danych zdecydowano się na wybór trzech różnych typów wykresów: pudełkowych, liniowych oraz słupkowych. Różnorodność tych diagramów umożliwiła dokładne przedstawienie i analizę wyników badań. Kierując się charakterystyką badanych zasobów komputera przydzielono do nich odpowiednie diagramy. Wykresy pudełkowe zostały wykorzystane jako narzędzie do przedstawienia zmian w liczbie klatek na sekundę (fps - ang. frames per second), co pozwoliło na zrozumienie rozkładu i zmienności tej istotnej metryki wydajności gier. Z kolei diagramy liniowe zostały zastosowane w przypadku analizy wykorzystania procesora, co umożliwiło obserwację dynamiki zużycia zasobu w trakcie badanych scenariuszy gry. Grafy słupkowe zostały użyte w kontekście analizy wykorzystania pamięci RAM oraz pamięci karty graficznej, co pozwoliło na klarowne i bezpośrednie przedstawienie zauważonych różnic w tych aspektach.

4.1. Analiza wydajności silników gier po kątem liczby klatek na sekundę

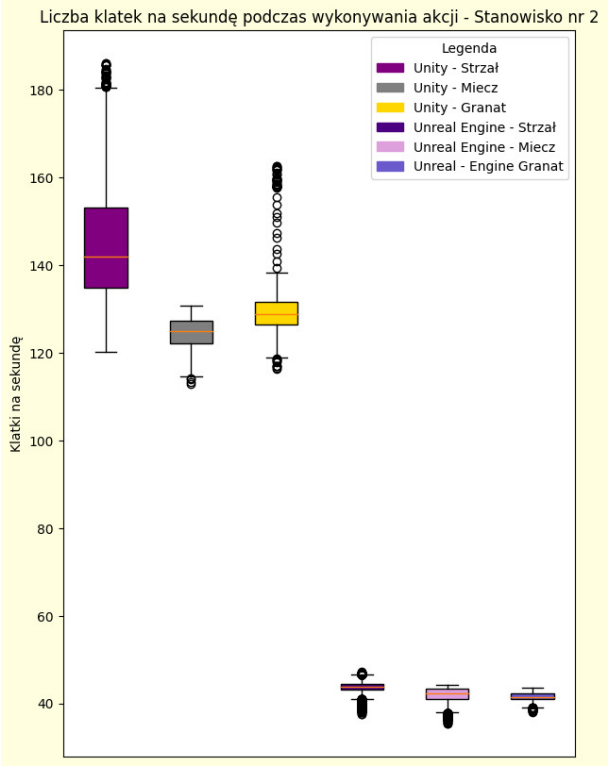
Liczba klatek na sekundę jest istotnym wskaźnikiem wydajności w grach komputerowych. Większa liczba klatek na sekundę oznacza płynniejszy obraz i poprawia komfort obcowania z grą. Jednakże istnieje pewien punkt, po osiągnięciu którego wzrost fps może przestać być zauważalny przez ludzkie oko, przy czym warto zaznaczyć, że ten punkt może różnić się w zależności od konkretnej osoby.

Analizując Rysunek 3, można zauważyć, że w przypadku stanowiska nr 1 (S1) gra wykorzystująca silnik Unity uzyskuje znacznie wyższe wartości. Porównując wyniki tych samych akcji na obu silnikach można zauważyć, że największa różnica median występuje w przypadku akcji strzału, gdzie Unity przewyższa Unreal Engine o 168 klatek. Warto również zaznaczyć, że największa różnica między pierwszym a trzecim kwartylem (Q1, Q3) dla Unreal Engine wyniosła niecałe 6 klatek dla ataku z użyciem granatu, podczas gdy dla Unity wynosiła aż 81 klatek w przypadku akcji strzału. Podsumowując, gra stworzona z użyciem Unity osiągała wyższe wartości fps, jednakże występowały pewne wahania w płynności w porównaniu do gry na Unreal Engine. Z pomiarów widocznych na Rysunku 4 dla stanowiska nr 2 (S2) wynikają podobne wnioski jak dla S1. W tym przypadku największa różnica pomiędzy wartościami środkowymi wyniosła 102 klatki, także dla akcji strzału. Największe odchylenie między pierwszym

a trzecim kwartylem wyniosło prawie 19 fps przy strzale dla Unity, podczas gdy dla Unreal Engine było ono równe 2,1 klatki. Dlatego wnioski są takie same jak dla S1.

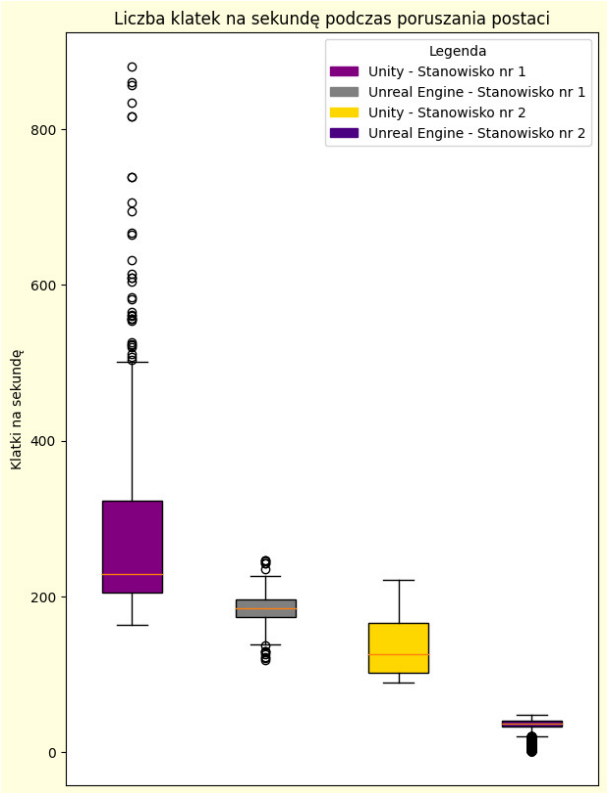


Rysunek 3: Wykresy pudełkowy dla testów funkcji ataku dla stanowiska nr 1.



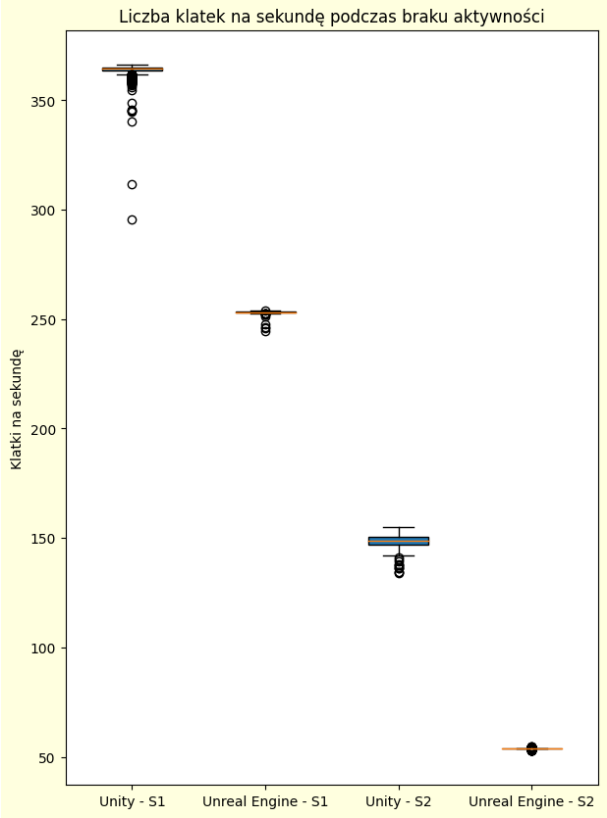
Rysunek 4: Wykresy pudełkowy dla testów funkcji ataku dla stanowiska nr 2.

Pomiary przeprowadzone podczas akcji z grupy poruszania, widoczne na Rysunku 5, wyraźnie wskazują na przewagę Unity nad konkurentem pod względem liczby wyświetlanych klatek na sekundę. Na stanowisku pierwszym różnica między pierwszym a trzecim kwartylem wyniosła prawie 120 klatek dla Unity oraz 23 dla Unreal Engine, przy wyższej medianie o 44 klatki na korzyść Unity. Testy przeprowadzone na drugim stanowisku potwierdzają przewagę silnika Unity. Szczególnie interesujące są różnice między Q1 i Q3 dla aplikacji działającej w środowisku Unreal Engine, wynoszące 23,2 fps dla S1 i 6,8 fps dla S2. Wartości te są wyższe w porównaniu z akcjami związanymi z atakiem, co może wynikać z faktu, że podczas tej akcji wykonywana jest bardzo duża liczba obliczeń. Blueprint'y, używane w Unreal Engine, działają na maszynie wirtualnej, która tłumaczy je na natywny kod języka C++, co może prowadzić do zwiększania czasu potrzebnego na wykonanie obliczeń i w rezultacie zmniejszenia liczby generowanych klatek.



Rysunek 5: Wykresy pudełkowe dla testów funkcji poruszania.

Wyniki pomiarów liczby klatek na sekundę uzyskane podczas uruchomionej gry, gdy użytkownik nie wykonywał żadnych czynności, zostały przedstawione na Rysunku 6. Warto zauważyć, że różnice między pierwszym a trzecim kwartylem nie przekraczały 2 klatek na sekundę dla żadnego ze stanowisk i silników. Spadki wartości były niewielkie, z wyjątkiem kilku obserwowanych spadków przy S1 i Unity. Dla S1 różnica median wyniosła ponad 110 klatek na korzyść Unity, podczas gdy dla S2 była niższa i wynosiła poniżej 100 fps, także na korzyść Unity.

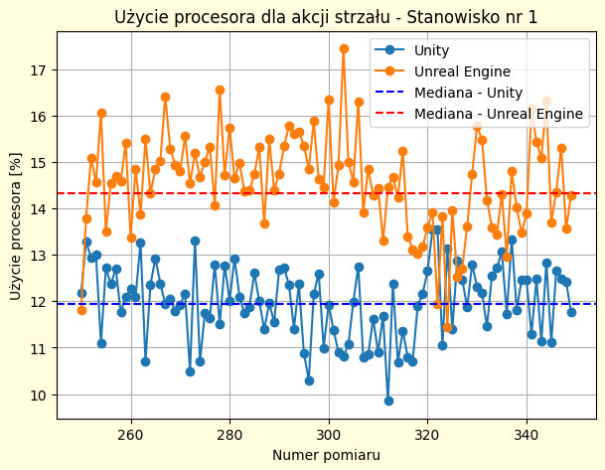


Rysunek 6: Wykresy pudełkowe dla testów bezczynności.

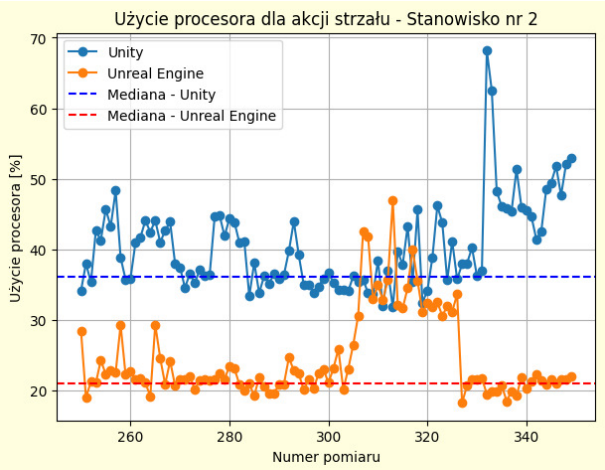
4.2. Analiza wydajności silników gier pod kątem zmian użycia procesora

Procesor (CPU) stanowi jedną z kluczowych jednostek obliczeniowych w komputerze i odgrywa niezwykle istotną rolę w zapewnieniu płynności i wydajności gier komputerowych. W niniejszym podrozdziale przeprowadzono analizę wydajności badanych silników gier pod kątem obciążenia procesora. Przedstawiono wykresy liniowe, które obrazują procentowe użycie procesora, pozwalając na lepsze zrozumienie, jak testowane silniki reagują na różne scenariusze gry. Warto zaznaczyć, że prezentowane na grafach mediany zostały obliczone na podstawie wszystkich zgromadzonych pomiarów, a w celu zachowania czytelności wykresów w pracy ograniczono przedstawienie wyników do pomiarów z zakresu od 250 do 350.

Wyniki otrzymane na stanowisku nr 1, przedstawione na Rysunku 7, wykazują niższe minimalne i maksymalne wartości użycia procesora w Unity w porównaniu do Unreal Engine podczas akcji strzału. Minimalna wartość wynosi mniej niż 10% dla Unity i około 11,4% dla Unreal Engine, podczas gdy maksymalna wartość wynosi 13,5% dla Unity i 17,4% dla Unreal Engine. Mediana dla Unreal Engine jest wyższa o ponad dwa punkty procentowe. Natomiast Rysunek 8 przedstawia wyniki dla stanowiska drugiego, gdzie Unity osiąga wyższe minimalne (około 31,8%) i maksymalne (około 68,2%) wartości w porównaniu do Unreal Engine (minimum około 18,2%; maksimum około 46,9%). Mediana uzyskana dla wyników silnika Unity również była wyższa o ponad 15 punktów procentowych.



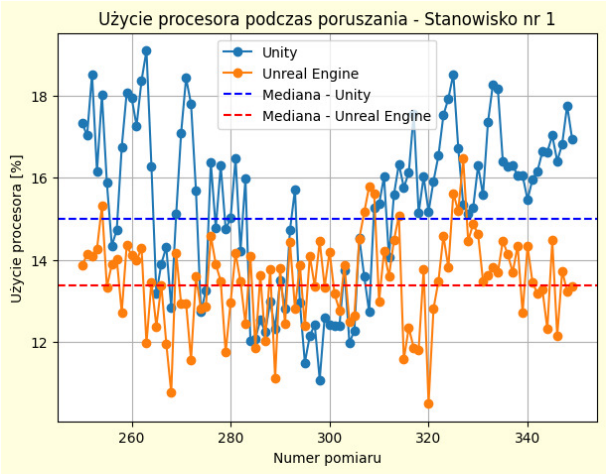
Rysunek 7: Wykresy liniowe testów strzału dla stanowiska nr 1.



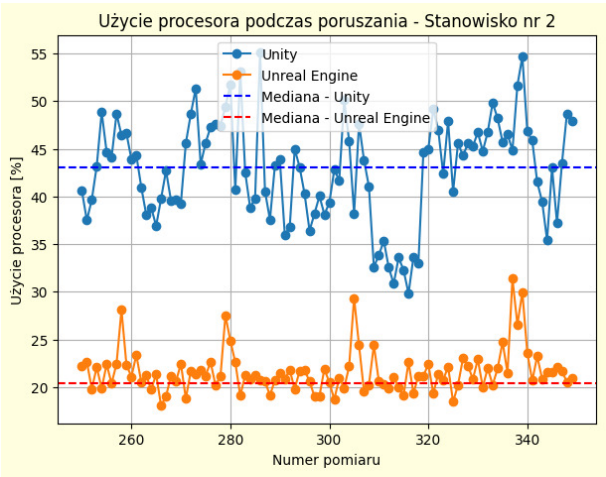
Rysunek 8: Wykresy liniowe testów strzału dla stanowiska nr 2.

Wyniki testów grupy poruszania na S1 widoczne są na Rysunku 9. W przypadku tych pomiarów Unity wykazywało większe obciążenie procesora. Minimalna wartość dla Unity (około 11,1%) była wyższa w porównaniu do konkurencyjnego silnika (minimum około 10,5%). Maksymalna wartość wynosiła około 19,1% dla Unity i około 16,5% dla Unreal Engine, a mediana była na poziomie około 15% dla Unity i około 13,4% dla Unreal Engine. Warto zauważyć, że dla stanowiska pierwszego, akcja poruszania była jedyną, przy której Unity wymagało większego użycia procesora niż Unreal Engine. Może to wynikać, tak jak w przypadku pomiarów fps tej akcji, z ograniczeń maszyny wirtualnej, która tłumaczy Blueprint'y na natywny język C++. Jeśli maszyna wirtualna ogranicza prędkość wykonania funkcji, to obliczenia wykonywane są wolniej i nie jest wymagana w krótszym czasie większa moc procesora. Silnik Unity zaś przy tej akcji osiągał wyższą liczbę klatek na sekundę co oznacza, że obliczenia były wykonywane w krótszym czasie i przy wyższym wykorzystaniu CPU. W przypadku S2 (Rysunek 10), Unity również osiąga wyższe wartości użycia CPU, niż Unreal Engine. Na stanowisku drugim, dla każdej akcji, odnotowano wyższe wartości środkowe w grze stworzonej przy pomocy Unity.

Podsumowując, na stanowisku nr 1, Unreal Engine zazwyczaj wykorzystywał procesor w większym stopniu niż Unity, natomiast na stanowisku drugim, Unreal Engine zawsze korzystał z mniejszej mocy procesora.



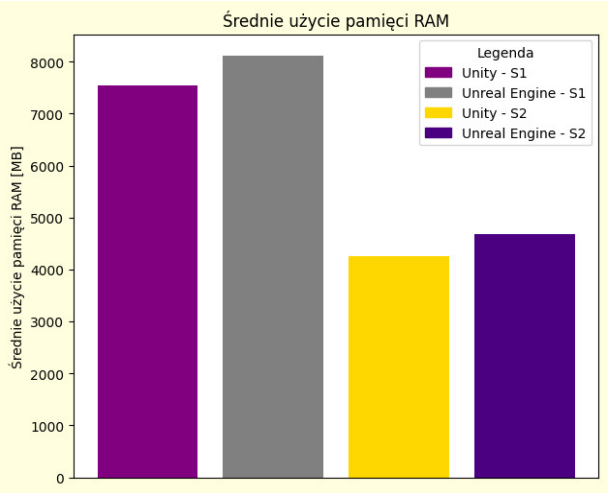
Rysunek 9: Wykresy liniowe testów poruszania dla stanowiska nr 1.



Rysunek 10: Wykresy liniowe testów poruszania dla stanowiska nr 2.

4.3. Analiza wydajności silników gier pod kątem użycia pamięci RAM

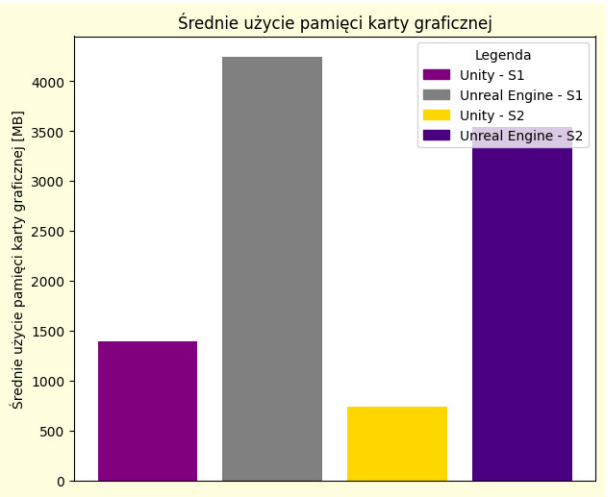
Pamięć RAM (ang. Random Access Memory) jest jednym z kluczowych komponentów w komputerach. Stanowi ona miejsce przechowywania danych aktualnie wykonywanych programów. Rysunek 11 przedstawia średnie użycie pamięci RAM przez stworzone gry. Wartości na wykresach zostały wyliczone na podstawie średnich ze wszystkich badań, ponieważ wartości wszystkich akcji dla danego silnika i stanowiska były podobne. Na stanowisku nr 1 Unreal Engine zużywał o 588 MB więcej pamięci w porównaniu do Unity. W przypadku drugiego stanowiska również wymagał więcej pamięci - o 409 MB. Analiza użycia pamięci RAM jednoznacznie wskazuje na mniejsze średnie użycie dla silnika Unity, aczkolwiek różnica ta nie jest znacząca.



Rysunek 11: Wykres słupkowy dla średniego użycia pamięci RAM.

4.4. Analiza wydajności silników gier pod kątem użycia pamięci karty graficznej

Karta graficzna (GPU) jest elementem komputera odpowiedzialnym za przetwarzanie i renderowanie obrazów oraz grafiki. Karta graficzna jest w stanie obsłużyć intensywne obliczenia związane z grafiką, co odciąża CPU i poprawia wydajność gier i aplikacji.



Rysunek 12: Wykres słupkowy dla średniego użycia pamięci karty graficznej.

Rysunek 12 przedstawia średnie użycie pamięci GPU w testowanych grach. Wartości przedstawione na wykresach wyliczono, podobnie jak w średnim użyciu pamięci RAM, poprzez wyciągnięcie średniej ze wszystkich badań. W przypadku S1 można zauważyć, że silnik Unity wykorzystywał 2844 MB pamięci GPU mniej niż Unreal Engine. Dla stanowiska nr 2 także odnotowano mniejsze o 2796 MB użycie pamięci GPU. Przeprowadzone badania wykazały ponad trzykrotnie większe użycie pamięci GPU przez silnik Unreal Engine.

Tak jak w przypadku analizy wyników dla zużycia pamięci RAM okazało się, że średnie wartości wykorzystania pamięci GPU są bardzo zbliżone dla testów wykonywanych na danej grze, więc wyliczono średnią z poprzednio wyliczonych średnich i wyniki przedsta-

wiono na powyższym wykresie oraz przeanalizowano. Na tej podstawie zauważono bardzo dużą różnicę w użyciu pamięci GPU na korzyść Unity.

5. Wnioski

Na podstawie analizy wydajności silników gier, przeprowadzonej w ramach niniejszej pracy, można wyciągnąć następujące wnioski:

1. Pomiar klatek na sekundę**:**
   a) silnik Unity osiągał wyższe średnie wartości klatek na sekundę przy każdej akcji na obu stanowiskach testowych,
   b) wyniki pomiarów wskazały mniejsze odchylenia standardowe (mniejsze wahania fps) dla gry napisanej w Unreal Engine,
2. Użycie procesora:
   a) na pierwszym stanowisku testowym silnik Unity wykazywał mniejsze użycie procesora w większości akcji, lecz różnica median nigdy nie przekraczała 4%,
   b) w przypadku S1 i akcji poruszania zmierzono wyższe użycie procesora dla Unity, co może wynikać z charakterystyki działania Blueprint'ów w konkurencyjnym silniku,
   c) na drugim stanowisku, przy każdej akcji, Unity wykazywało wyższe użycie procesora,
3. Użycie pamięci RAM:
   a) średnie użycie pamięci RAM było niższe zarówno na pierwszym, jak i drugim stanowisku dla silnika Unity,
   b) różnica w użyciu pamięci RAM między silnikami nie była znacząca,
4. Użycie pamięci GPU:
   a) średnie użycie pamięci GPU było niższe zarówno na pierwszym, jak i drugim stanowisku dla silnika Unity,
   b) różnica użycia pamięci GPU była ponad trzykrotnie wyższa dla Unreal Engine.

Wyniki badań częściowo potwierdzają postawioną we wstępie tezę, która twierdzi, że Unity wymaga mniej zasobów komputera niż Unreal Engine. Testy pamięci RAM i GPU potwierdzają mniejsze zapotrzebowanie na te zasoby przez silnik Unity. Jednak analiza użycia procesora wykazała, że na stanowisku nr 2 Unity wymagało większej mocy procesora. Dane zebrane podczas akcji poruszania na stanowisku pierwszym również wskazują, na wyższe użycie zasobów CPU przez Unity.

Literatura

[1] H. Żukowski, Comparison of 3D games' efficiency with use of CRYENGINE and Unity game engines, Journal of Computer Sciences Institute 13 (2019) 345–348.

[2] A. M. Barczak, H. Woźniak, Comparative Study on Game Engines, Studia Informatica, Systems and Information Technology 23(1-2) (2020) 5–24.

[3] H. A. J. Al Lawati, The Path of UNITY or the Path of UNREAL? A Comparative Study on Suitability for Game Development, Journal of Student Research (2020) 1–7.

[4] J. Zhang, Implementation and Optimization of Particle Effects based on Unreal Engine 4, Journal of Physics: Conference Series 1575 012187 (2020) 1–7.

[5] A. Pattrasitidecha, Comparison and evaluation of 3D mobile game engines, Department of Computer Science and Engineering Göteborg, Sweden, 2014.

[6] P. Mishra, U. Shrawankar, Comparison between Famous Game Engines and Eminent Games, International Journal of Interactive Multimedia and Artificial Intelligence 4 (2016) 69–77.

[7] P. Skop, Comparison of performance of game engines across various platforms, Journal of Computer Sciences Institute 7 (2018) 116–119.

[8] A. Šmíd, Comparison of unity and unreal engine, Czech Technical University in Prague (2017) 41–61.

[9] E. Christopoulou, S. Xinogalos, Overview and comparative analysis of game engines for desktop and mobile devices, International Journal of Serious Games 4(4) (2017) 21–36.

[10] A. Neppius, 3D Game Texturing: Comparative Analysis Between Hand-painted and PBR pipelines, South-Eastern Finland University of Applied Sciences, Finland, 2022.

[11] A. Andrade, Game engines: A survey, EAI Endorsed Transactions on Serious Games 2(6) (2015) 1–6.

[12] X. Cui, H. Shi, A*-based pathfinding in modern computer games, International Journal of Computer Science and Network Security 11(1) (2011) 125–130.

JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 30 (2024) 61–67
Received: 14 November 2023
Accepted: 17 December 2023

Classification Performance Comparison of BERT and IndoBERT on Self-Report of COVID-19 Status on Social Media

Porównanie wyników klasyfikacji BERT i IndoBERT w zakresie samodzielnego zgłaszania statusu COVID-19 w mediach społecznościowych

Irwan Budiman, Mohammad Reza Faisal*, Astina Faridhah, Andi Farmadi, Muhammad Itqan Mazdadi, Triando Hamonangan Saragih, Friska Abadi

Department of Computer Science, Lambung Mangkurat University, Banjarbaru 70714, Indonesia

Abstract
Messages shared on social media platforms like X are automatically categorized into two groups: those who self-report COVID-19 status and those who do not. However, it is essential to note that these messages cannot be a reliable monitoring tool for tracking the spread of the COVID-19 pandemic. The classification of social media messages can be achieved through the application of classification algorithms. Many deep learning-based algorithms, such as Convolutional Neural Networks (CNN) or Long Short-Term Memory (LSTM), have been used for text classification. However, CNN has limitations in understanding global context, while LSTM focuses more on understanding word-by-word sequences. Apart from that, both require a lot of data to learn. Currently, an algorithm is being developed for text classification that can cover the shortcomings of the previous algorithm, namely Bidirectional Encoder Representations from Transformers (BERT). Currently, there are many variants of BERT development. The primary objective of this study was to compare the effectiveness of two classification models, namely BERT and IndoBERT, in identifying self-report messages of COVID-19 status. Both BERT and IndoBERT models were evaluated using raw and preprocessed text data from X. The study's findings revealed that the IndoBERT model exhibited superior performance, achieving an accuracy rate of 94%, whereas the BERT model achieved a performance rate of 82%.

Keywords: Text classification; Covid-19 status; X; BERT; IndoBERT

Streszczenie
Wiadomości udostępniane na platformach mediów społecznościowych, takich jak X, są automatycznie dzielone na dwie grupy: te, które samodzielnie zgłaszają swój status COVID-19, i te, które tego nie robią. Należy jednak pamiętać, że komunikaty te nie mogą stanowić wiarygodnego narzędzia monitorowania umożliwiającego śledzenie rozprzestrzeniania się pandemii Covid-19. Klasyfikację komunikatów w mediach społecznościowych można osiągnąć poprzez zastosowanie algorytmów klasyfikacyjnych. Do klasyfikacji tekstu wykorzystano wiele algorytmów opartych na głębokim uczeniu się, takich jak konwolucyjne sieci neuronowe (CNN) czy pamięć długoterminowa (LSTM). Jednak CNN ma ograniczenia w rozumieniu kontekstu globalnego, podczas gdy LSTM koncentruje się bardziej na zrozumieniu sekwencji słowo po słowie. Poza tym oba wymagają dużej ilości danych do nauki. Obecnie opracowywany jest algorytm klasyfikacji tekstu, który może pokryć wady poprzedniego algorytmu, a mianowicie dwukierunkowych reprezentacji enkoderów z transformatorów (BERT). Obecnie istnieje wiele wariantów rozwoju BERT. Podstawowym celem tego badania było porównanie skuteczności dwóch modeli klasyfikacji, a mianowicie BERT i IndoBERT, w identyfikowaniu komunikatów samoopisowych na temat statusu COVID-19. Zarówno modele BERT, jak i IndoBERT oceniano przy użyciu surowych i wstępnie przetworzonych danych tekstowych z X. Wyniki badania wykazały, że model IndoBERT wykazał się doskonałą wydajnością, osiągając współczynnik dokładności na poziomie 94%. Natomiast model BERT osiągnął wskaźnik wydajności na poziomie 82%.

Słowa kluczowe: Klasyfikacja tekstu; Covid-19 status; X; BERT; IndoBERT

*Corresponding author
Email address: reza.faisal@ulm.ac.id (M. R. Faisal)

1. Introduction

Covid-19 is an ailment that arises from the Coronavirus, making its debut in Wuhan at the culmination of 2019. Subsequently, the first accounts of COVID-19 cases in Indonesia were disclosed in March 2020. Some prevalent indications of COVID-19 in individuals include heightened body temperature, coughing, difficulty in respiration, and diminished olfactory function. The COVID-19 pandemic, as of the conclusion of 2019, has registered an excess of 3 million documented cases across the globe, along with roughly 208,516 fatalities as of April 2020 [1]. This elevated level of mortality is attributable to a delay in promptly recognizing those afflicted with Covid-19 [2]. Consequently, those affected continue to engage in their customary activities, thereby facilitating the transmission of the virus to their close associates.

The worldwide dissemination of COVID-19 has evolved into a widespread pandemic, significantly impacting individuals across the globe. Throughout this global health crisis, social media, particularly the popular platform X, has emerged as a predominant means of disseminating information pertaining to the Covid-19 virus.

X has emerged as a widely embraced platform, boasting a staggering number of over 3.7 million active users who diligently disseminate approximately 10 million posts per diem [3]. Amongst this vast array of posts, one intriguing facet worth mentioning is the propensity for individuals to utilize this platform as a conduit for sharing insightful information about the COVID-19 crisis, including personal anecdotes relating to symptoms experienced or even instances of infection. Furthermore, the online community of social media enthusiasts has aptly utilized X as a platform for recounting the adverse impact that the pandemic has had on their respective families. Given its real-time nature, X lends itself seamlessly to monitoring the progression of the Covid-19 pandemic.

Identification of status of COVID-19 can be automated through the classification of self-report messages employing classification algorithms extensively developed by scholars in the field. Scholars have employed various techniques, such as Long Short-Term Memory (LSTM), Bi-directional LSTM (BiLSTM), word2vec, Support Vector Machines (SVM), and Convolutional Neural Networks (CNN), for text classification.

The classification of emotions in the text was undertaken in the study [4] utilizing the LSTM method, which yielded an accuracy score of 73.15%. In [3], the investigation focused on analysing emotions and sentiments in posts using BERT, resulting in an accuracy rate of 92%. In a subsequent study conducted by [5], the examination of emotions and sentiments deployed the BERT model, achieving a performance level of 92%. Furthermore, the researchers devised the IndoBERT model to classify Indonesian texts [6]. The findings of this study demonstrate that the IndoBERT model exhibits improved efficacy when applied to Indonesian texts with limited training data.

The BERT model has gained popularity among researchers in recent times. In contrast to other language models, BERT was developed as a pre-trained deep bidirectionally trained model using unlabeled text data. This was achieved by integrating left- and right-sided context layers [7]. Consequently, BERT models can be tailored to various machine learning tasks, such as classification and question answering, by simply incorporating a single layer [8].

However, BERT continues to possess limitations, specifically about the restricted availability of words in the Indonesian language. Consequently, numerous scholars have endeavoured to design BERT architectures by their respective languages. Presently, the BERT model has been made accessible in a multitude of languages, including Indonesian, exemplified by the IndoBERT [6]. Another study on text classification employed the IndoBERT model as a basis and conducted an experiment to investigate the impact of text preprocessing on classification performance [9].

Current literature on COVID-19 on social media predominantly focuses on sentiment analysis. Conversely, there is a dearth of research on text classification that aims to identify self-reported messages regarding COVID-19 status, and the available datasets primarily consist of texts in the English language [1], [10], [11].

A study was undertaken to compare the performance of BERT and IndoBERT models in identifying self-reported COVID-19 status from social media messages in Indonesian. The study involved training and testing each model using both raw and preprocessed text. The objective was to determine which model combination offers the highest classification performance.

2. Dataset

The datasets utilized in this investigation are derived from preceding investigations [12]. The dataset comprises 1000 messages sourced from X encompassing the keyword covid, as demonstrated in. The dataset is segregated into two classifications, namely, positive and negative. A positive message is defined as a message that, alongside the keyword COVID-19, exhibits one or more symptoms of COVID-19, and its connotation indicates that the individual expressing the complaint is presumed positive for COVID-19. The quantity of positive messages totals 500 messages. Examples of messages within the positive category can be observed in Table 1.

Table 1: Positive messages

Post	Translation	Class Label
doakan aku cepat sembuh dari covid	Pray for my speedy recovery from Covid	positive
Hari ini saya meriang, dan ketika periksa ke klinik ternyata saya positif covid	Today I had a fever, and when I went to the clinic it turned out I was positive for Covid	positive

Negative category messages are messages that do not pertain to positive COVID-19 cases. The total count of negative messages amounts to 500. Instances of messages belonging to negative categories are shown in Table 2.

Table 2: Negative message

Post	Translation	Class Label
Hari saya test swab namun hasilnya negatif covid	Today I had a swab test but the results were negative for Covid	negative
Nafas agak sesak, aku kira gejala covid ternyata tidak	Breathing a bit short, I thought it was a symptom of Covid but it wasn't	negative

3. Research Implementation

The various phases executed throughout this investigation can be visually observed and analyzed in a detailed manner by referencing the comprehensive illustration labelled as Figure 1.

The initial phase entails gathering datasets, as expounded upon in the preceding section. Subsequently, the text is subjected to preprocessing in order to achieve normalization through the utilization of tokenization, stemming, and stopword elimination techniques [13], [14], [15]. Tokenization is a procedure executed to fragment sentences into segments of lexemes, punctuation, and other significant articulations in agreement with the stipulations of the utilized linguistic system. Stemming is the procedure of transforming a term that possesses an inflection into its foundational term (radical structure) by eliminating affixes such as prefixes, suffixes, and confixes. Stopwords Elimination is the procedure of excluding lexemes that lack any import.

Additional preprocessing measures were undertaken by incorporating specific tokens, namely [CLS] at the commencement of the sentence, [SEP] at its conclusion, and [PAD] in instances of shorter sentences [16].

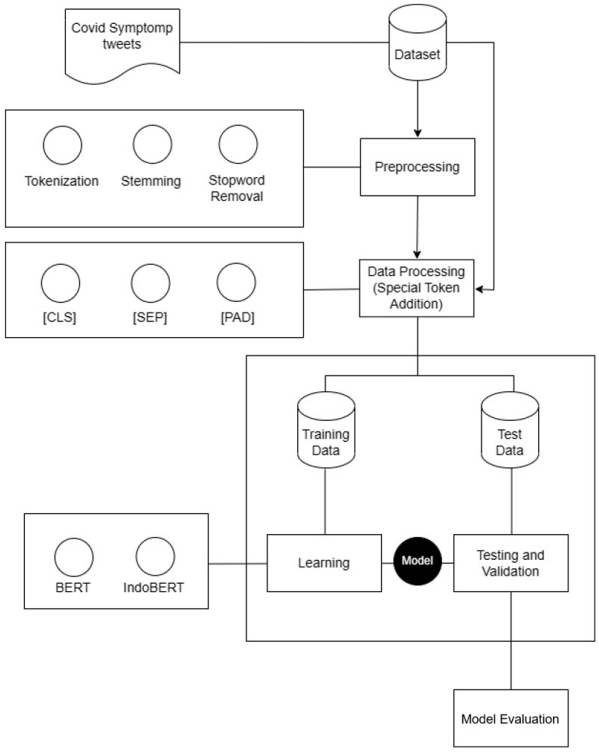


Figure 1: Research Flow.

The subsequent phase entails the division of the data utilizing the hold-out technique. This technique effectively separates the data into two distinct categories: training data and test data. The training data accounts for 80% of the composition, while the test data accounts for the remaining 20% [17], [18]. Following this, the learning phase commences, with the objective of constructing a classification model through the utilization of two methods, namely BERT and IndoBERT. As a result of this stage, four models are created, incorporating a combination of methods and techniques, as illustrated in Table 3. It is important to note that the hyperparameter values employed in both the BERT and IndoBERT methods remain consistent. Furthermore, the models that are constructed utilizing the column values of the preprocessing are of no significance. These models do not undergo any text data processing techniques such as stemming or stopword removal.

Table 3: List of Classification Models

No	Preprocessing	Classification Method	Hyper Parameter
1	No	BERT	Batch size=16 Learning rate=2e-5 Epoch=10
2	Yes		
3	No	IndoBert	
4	Yes		

The testing and validation phase carries out the anticipation of the class label for the test data through four model classifications that were subsequently established in the preceding stage. Additionally, the performance outcomes for the four models are contrasted and scrutinized at the model evaluation stage. The assessment of the performance of the classification model at this stage employs the employment of the confusion matrix to determine accuracy, specificity, and sensitivity [14]. The depiction of the confusion matrix can be observed in Table 4.

Table 4: Confusion matrix

		Predicted Class	
	Class	Positive	Negtive
Actual Class	Positive	True Positif (TP)	False Negative (FN)
	Negative	False Positive (FP)	True Negative (TN)

True Positive (TP) is an outcome where the model correctly predicts the positive class. True Negative (TN) is an outcome where the model correctly predicts the negative class. False Positive (FP) is an outcome where the model incorrectly predicts the positive class. False Negative (FN) is an outcome where the model incorrectly predicts the negative class. Those values can be used to calculate accuracy, sensitivity, and specificity classification performance values. The formula of the three classification performances can be seen below.

$$Accuracy = \frac{(TP+TN)}{(TP+FP+TN+FN)} \text{ x } 100\% \quad (1)$$

$$Sensitivity = \frac{(TP)}{(TP+FN)} \text{ x } 100\% \quad (2)$$

$$Specificity = \frac{(TN)}{(TN+FP)} \text{ x } 100\% \quad (3)$$

4. Results and Discussion

The results obtained from processing textual data in the dedicated token supplementary phase are presented in Table 5.

Table 5: Result of tokenization

	BERT	IndoBERT
Text	“Qadarullah”, “pak”, “suami”, “negatif”, “NS-1”, “positif”, “covid”, “,”, “plus”, “ku”, “jg”, “sama”, “positif”, “covid”, “.”, “Suami”, “cm”, “bergejala”, “3”, “hari”, “di”, “awal”, “,”, “demam”, “40”, “derajat”, “,”, “skrg” “pilek”, “ma”, “batuk2”, “.”, “Ku”, “baru”, “gejala”, “kemarin”, “,”, “demam”, “38”, “drjt”, “,”, “pusing”, “ma”, “linu2”, “badan”	
Tokenization results using each pre-train Vocabulary	'q', '##ada', '##rul', '##lah', 'pak', 'sua', '##mi', 'negatif', 'ns', '-', '1', 'positif', 'co', '##vid', ',', 'plus', 'ku', 'jg', 'sama', 'positif', 'co', '##vid', '.', 'sua', '##mi', 'cm', 'berge', '##jala', '3', 'hari', 'di', 'awal', '(', 'dem', '##am', '40', 'dera', '##jat', ')', 'sk', '##rg', 'pile', '##k', 'ma', 'batu', '##k', '##2', '.', 'ku', 'baru', 'ada', 'ge', '##jala', 'kem', '##arin', ',', 'dem', '##am', '38', 'dr', '##jt', ',', 'pus', '##ing', 'ma', 'lin', '##u', '##2', 'badan'	‘qadar’, ‘##ullah’, 'pak', 'suami', 'negatif', 'ns', '-', '1', 'positif', 'cov', '##id', ',', 'plus', 'ku', 'jg', 'sama', 'positif', 'cov', '##id', '.', 'suami', 'cm', 'berg', '##eja', '##la', '3', 'hari', 'di', 'awal', '(', 'demam', '40', 'derajat', ')', 'sk', '##r', '##g', 'pil', '##ek', 'ma', 'batuk', '##2', '.', 'ku', 'baru', 'ada', 'gejala', 'kemarin', ',', 'demam', '38', 'dr', '##j', '##t', ',', 'pusing', 'ma', 'lin', '##u', '##2', 'badan'

After the process of tokenization, there is an inclusion of specific tokens. These tokens are the [CLS] token, which is positioned at the beginning of the sentence, the [SEP] token, which is placed at the end of the sentence, and the [PAD] token, which is assigned to sentences that have a word count lower than the maximum number of words in a predefined sentence. Moreover, each token undergoes encoding based on the vocabulary index, and the outcomes of this encoding procedure are presented in Table 6.

Table 6: Result of special token addition

Special token addition	BERT	IndoBERT
	‘[CLS]’, ‘q’, ‘##ada’, ‘##rul’, ‘##lah’, ‘sua’, ‘##mi’, ‘negatif’, ‘ns’, ‘-‘, ‘1’, ‘positif’, ‘co’, ‘##vid’, ‘,’, ‘plus’, ‘ku’, ‘jg’, ‘sama’, ‘positif’, ‘co’, ‘##vid’, ‘.’, ‘sua’, ‘##mi’, ‘ge’, ‘##jala’, ‘3’, ‘hari’, ‘dem’, ‘##am’, ‘40’, ‘dera’, ‘##jat’, ‘,’, ‘pile’, ‘##k’, ‘ma’, ‘batu’, ‘##k’, ‘##2’, ‘.’, ‘ku’, ‘ge’, ‘##jala’, ‘kem’, ‘##arin’, ‘,’, ‘dem’, ‘##am’', ‘38’, ‘dr’, ‘##jt’, ‘,’, ‘pus’, ‘##ing’, ‘ma’, ‘lin’, ‘##u’, ‘##2’, ‘badan’, ‘[SEP]’, ‘[PAD]’, ..., ‘[PAD]’	‘[CLS]’, ‘qadar’, ‘##ullah’, ‘sua-mi’, ‘negatif’, ‘ns’, ‘-‘, ‘1’, ‘positif’, ‘cov’, ‘##vid’, ‘,’, ‘plus’, ‘ku’, ‘jg’, ‘positif’, ‘cov’, ‘##id’, ‘.’, ‘sua-mi’, ‘gejala’, ‘##la’, ‘3’, ‘hari’, demam’, ‘40’, ‘derajat’, ‘,’, ‘pil’, ‘##ek’, ‘ma’, ‘batuk’, ‘##2’, ‘.’, ‘ku’, ‘gejala’, ‘kemarin’, ‘,’, ‘demam’, ‘38’, ‘dr’, ‘##j’, ‘##t’, ‘,’, ‘pusing’, ‘ma’, ‘lin’, ‘##u’, ‘##2’, ‘badan’ ‘[SEP]’, ‘[PAD]’, ..., ‘[PAD]’

The subsequent stage involves executing an encoding procedure that transforms each token into an index. The outcomes are observable in Table 7.

Table 7: Result of encoding step

Encoding	BERT	IndoBERT
	101, 159, 11466, 31387, 16910, 10559, 10555, 81523, 28235, 118, 122, 66150, 10348, 41194, 117, 10608, 14886, 52817, 66150, 10348, 41194, 119, 10559, 10555, 50346, 29378, 124, 117, 10188, 11064, 10606, 75323, 15714, 78191, 10167, 10507, 19787, 10167, 10835, 119, 14123, 25463, 29378, 47304, 48499, 117, 10188, 11064, 11330,	2, 28216, 2421, 4425, 3778, 23803, 17, 21, 4590, 28912, 1528, 16, 8211, 1984, 23599, 4590, 28912, 1528, 18, 4425, 6004, 23, 1843, 16, 7322, 3828, 8033, 16, 4086, 1533, 2262, 14575, 952, 18, 1984, 6004, 4454, 16, 7322, 5885, 2455, 954, 930, 16, 12693, 2262, 4560, 943, 952, 2835, 3, 0, ..., 0

	11407, 30373, 117, 43955, 10285, 10507, 29715, 10136, 10835, 19605, 102, 0, ..., 0	

The first step involves generating an attention mask, which serves to differentiate between the word token's significance and the padding's insignificance. Examination of the outcomes is presented in Table 8.

Subsequently, the customized input is subsequently introduced into the BERT and IndoBERT networks, both of which consist of a series of 12-layer encoder transformers. Each encoder layer is composed of two sub-layers: the first one being a multi-head self-attention mechanism and the second one being a fully connected feed-forward network. Following the passage through all encoders, a vector output is generated for each token. However, only the vector output of the token [CLS] will be utilized as the vector input classifier.

Table 8: Result of attention mask

Attention mask	BERT	IndoBERT
	1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0,..., 0	1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, ..., 0

The datasets transformed into the resultant input vector configuration are subsequently separated into training and test data. Following that, a phase of learning, testing, and validation ensues. The outcomes about the performance of the classification model can be observed in Table 9.

Table 9: Performance of classification models

No	Preprocessing	Classification Method	Classification Performance
1	No	BERT	Accuracy=81.50% Specificity=91.09% Sensitivity=71.72%
2	Yes	BERT	Accuracy=82.00% Specificity=89.11% Sensitivity=74.75%
3	No	IndoBERT	Accuracy=89.50% Specificity=82.83% Sensitivity=96.04%
4	Yes	IndoBERT	Accuracy=94.00% Specificity=92.00% Sensitivity=96.00%

Comparisons regarding the performance of each classification model can be observed in Figure 2. Upon examining these results, it becomes apparent that model No. 4 outperforms the other models. Model No. 4 is a text-based model that has undergone preprocessing by applying stemming and stopword removal techniques.

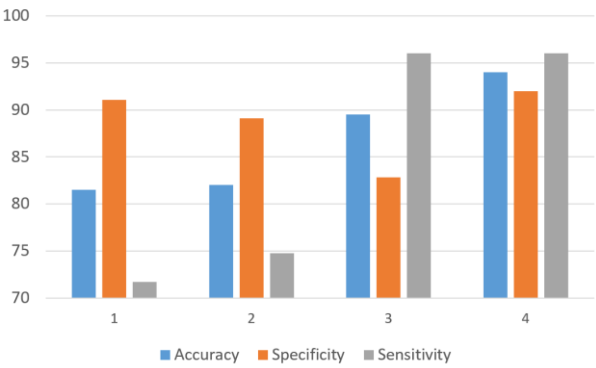


Figure 2: Comparison of classification models' performance.

Figure 3 shows a juxtaposition of the classification performance achieved by different classification methods. The outcomes manifest that models constructed utilizing the IndoBERT technique exhibit superior efficacy, as inferred from the accuracy and sensitivity metrics. Remarkably, the elevated sensitivity metric signifies the model's commendable aptitude in accurately discerning self-reported messages pertaining to COVID-19 status.

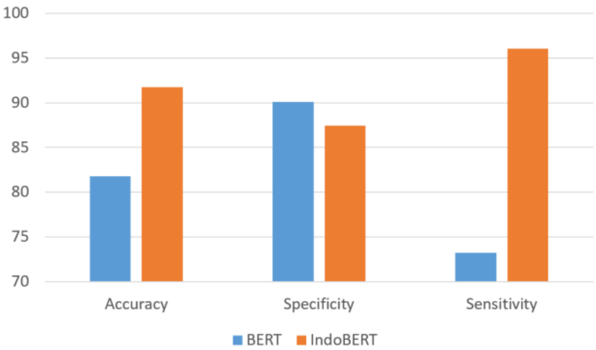


Figure 3. Comparison performance between BERT and IndoBERT models.

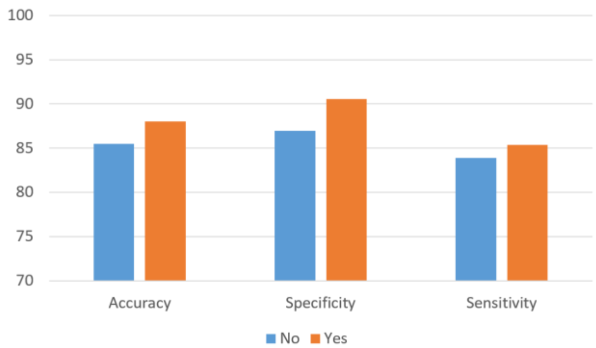


Figure 4. Effect of preprocessing on classification models' performance.

Figure 4 displays a comparison of the impact of preprocessing on the efficacy of classification models. This comparison reveals that preprocessing steps, such as stemming and stopword elimination, have the potential

to enhance the classification performance of BERT and IndoBERT-based models.

5. Conclusion

The findings of this investigation demonstrated that the execution of the IndoBERT-founded approach exhibited the capability to generate the most outstanding model for discerning self-reported COVID-19 status messages when compared to the classification model founded on BERT. Additional evidence presented in this analysis indicates that the preprocessing impact of stemming and stopword elimination can enhance the performance of identifying self-reported messages about COVID-19 status.

The self-report COVID-19 status message identification model, based on the IndoBERT model, demonstrated the highest level of performance. Its accuracy was recorded at 94%, with a specificity of 92% and a sensitivity of 96%. Conversely, the BERT-based model achieved an accuracy of 82%, a specificity of 89.11%, and a sensitivity of 74.75%. Employing the unpaired t-test on the test results, a two-tailed P value of 0.0488 was obtained, indicating a statistically significant difference. Thus, it can be concluded that the IndoBERT-based model exhibits significantly superior performance. On the other hand, the impact of text preprocessing on the classification model's performance was found to be insignificant. Using the paired t test technique, a two-tailed P value of 0.0528 was observed.

In the subsequent investigation, an evaluation will be conducted regarding additional BERT-derived methodologies, including ALBERT, Roberta, and DistilBERT. The primary objective entails constructing a model effectively discerning self-reported COVID-19 status within messages.

6. Acknowledgements

The computation time for the computer system in this study was furnished by the Data Science Lab, which ope-rates within the Computer Science Department of the Faculty of Mathematics and Natural Sciences at Lambung Mangkurat University. This investigation received financial backing from the Program Dosen Wajib Meneliti (PDWM) grant provided by PNBP Lambung Mangkurat University.

References

[1] T. Mackey, V. Purushothaman, J. Li, N. Shah, M. Nali, C. Bardier, B. Liang, M. Cai, R. Cuomo, Machine learning to detect self-reporting of symptoms, testing access, and recovery associated with COVID-19 on Twitter: retrospective big data infoveillance study, JMIR public health and surveillance, 6(2) (2020) 1-9, https://doi.org/10.2196/19509

[2] A. Z. Klein, A. Magge, K. O'Connor, J. I. Flores Amaro, D. Weissenbacher, and G. Gonzalez Hernandez, Toward using Twitter for tracking COVID-19: a natural language processing pipeline and exploratory data set, Journal of medical Internet research, 23 (1) (2021) 1-6, https://doi.org/10.2196/25314

[3] F. E. Ayo, O. Folorunso, F. T. Ibharalu, and I. A. Osinuga, Machine learning techniques for hate speech classification of Twiiter data: State-of-The-Art, future challenges and research directions, Computer Science Review, 38 (2020) 1-34, https://doi.org/10.1016/j.cosrev.2020.100311

[4] M. A. Riza, N. Charibaldi, U. Pembangunan, and N. Veteran, Emotion Detection in Twiter Social Media Using Long Short - Term Memory (LSTM) and Fast Text, 3 (1) (2021) 15–26, https://doi.org/10.25139/ijair.v3i1.3827

[5] A. Chiorrini, C. Diamantini, A. Mircoli, and D. Potena, Emotion and sentiment analysis of posts using BERT, In EDBT/ICDT Workshops, 3 (2021) 1-7

[6] B. Wilie, K. Vincentio, G.I. Winata, S. Cahyawijaya, X. Li, Z.Y. Lim, S. Soleman, R. Mahendra, P. Fung, S. Bahar, A. Purwarianti, IndoNLU: Benchmark and Resources for Evaluating Indonesian Natural Language Understanding, arXiv preprint arXiv:2009.05387, (2020) 1-15

[7] P. Ganesh, Y. Chen, X. Lou, M.A. Khan, Y. Yang, H. Sajjad, P. Nakov, D. Chen, M . Winslett, Compressing large-scale transformer-based models: A case study on BERT, Transactions of the Association for Computational Linguistics, 9 (2021) 1061–1080, https://doi.org/10.1162/tacl_a_00413

[8] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, BERT: Pre-training of deep bidirectional transformers for language understanding, In Proceedings of naacL-HLT, 1 (2019) 4171–4186

[9] F. Koto, A. Rahimi, J. H. Lau, and T. Baldwin, IndoLEM and IndoBERT: A Benchmark Dataset and Pre-trained Language Model for Indonesian NLP, COLING 2020 - 28th International Conference on Computational Linguistics, Proceedings of the Conference (2020) 757–770, http://dx.doi.org/10.18653/v1/2020.coling-main.66

[10] C. Menni, A.M. Valdes, M.B. Freidin, C.H. Sudre, L.H. Nguyen, D.A. Drew, S. Ganesh, T. Varsavsky, M.J. Cardoso, J.S. El-Sayed Moustafa, A. Visconti, Real-time tracking of self-reported symptoms to predict potential COVID-19, Nature medicine, 26 (7) (2020) 1037–1040, https://doi.org/10.1038/s41591-020-0916-2

[11] M. A. Al-garadi, Y. Yang, S. Lakamana, A. Sarker, A Text Classification Approach for the Automatic Detection of Twitter Posts Containing Self-reported COVID-19 Symptoms, Open Review, (2020) 1–5

[12] S. N. Sari, M. R. Faisal, D. Kartini, I. Budiman, Comparison of Feature Extraction with Supervised and Unsupervised Weighting in the Random Forest Algorithm for Monitoring Reports of COVID-19 Sufferers on Twitter, Jurnal Komputasi, 11 (1) (2023) 34–42, http://dx.doi.org/10.23960%2Fkomputasi.v11i1.6650

[13] M. R. Faisal, I. Budiman, F. Abadi, M. Haekal, D. T. Nugrahadi, A comparison of word embedding-based extraction feature techniques and deep learning models of natural disaster messages classification, Journal of Computer Sciences Institute, 27 (2023) 145–153, https://doi.org/10.35784/jcsi.3322

[14] M. Khairie, M. R. Faisal, R. Herteno, I. Budiman, F. Abadi, and M. I. Mazdadi, The Effect of Channel Size on Performance of 1D CNN Architecture for Automatic Detection of Self-Reported COVID-19 Symptoms on Twitter, in 2023 International Seminar on Intelligent Technology and Its Applications (ISITIA) (2023) 621–625. https://doi.org/10.1109/ISITIA59021.2023.10220444

[15] M. R. Faisal, I. Budiman, F. Abadi, D. T. Nugrahadi, M. Haekal, and I. Sutedja, Applying Features Based on Word Embedding Techniques to 1D CNN for Natural Disaster Messages Classification, 2022 5th International Conference on Computer and Informatics Engineering, IC2IE 2022, (2022) 192–197, https://doi.org/10.1109/IC2IE56416.2022.9970188

[16] G. A. Pradnyana, W. Anggraeni, E. M. Yuniarno, and M. H. Purnomo, Fine-Tuning IndoBERT Model for Big Five Personality Prediction from Indonesian Social Media, in 2023 International Seminar on Intelligent Technology and Its Applications (ISITIA) (2023) 93–98, https://doi.org/10.1109/ISITIA59021.2023.10221074

[17] M. F. Nafiz, D. Kartini, M. R. Faisal, F. Indriani, and T. Hamonangan, Automated Detection of COVID-19 Cough Sound using Mel-Spectrogram Images and Convolutional Neural Network, Jurnal Ilmiah Teknik Elektro Komputer dan Informatika (JITEKI), 9 (3) (2023) 535–548, http://dx.doi.org/10.26555/jiteki.v9i3.26374

[18] K. Y. Halim, D. T. Nugrahadi, M. R. Faisal, R. Herteno, and I. Budiman, Gender Classification Based on Electrocardiogram Signals Using Long Short Term Memory and Bidirectional Long Short Term Memory, Jurnal Ilmiah Teknik Elektro Komputer dan Informatika (JITEKI), 9 (3) (2023) 606–618, http://dx.doi.org/10.26555/jiteki.v9i3.26354